Escuela de Doctorado
y Estudios de Posgrado
Universidad de La Laguna

Máster Universitario en Ingeniería Industrial

Trabajo Fin de Máster

Sistema de monitorización de variables físicas para el uso de actividad deportiva

Autor: Gabriel González Rial
Tutor: Oswaldo Bernabé González Hernández

8 de marzo de 2022

La publicación de este Trabajo Fin de Máster solo implica que el estudiante ha obtenido al menos la nota mínima exigida para superar la asignatura correspondiente, no presupone que su contenido sea correcto, aunque si aplicable. En este sentido, la ULL no posee ningún tipo de responsabilidad hacia terceros por la aplicación total o parcial de los resultados obtenidos en este trabajo. También pone en conocimiento del lector que, según la ley de protección intelectual, los resultados son propiedad intelectual del alumno, siempre y cuando se haya procedido a los registros de propiedad intelectual o solicitud de patentes correspondientes con fecha anterior a su publicación.

Índice

Página

Resumen vi

Abstract vii

1. Introducción **3**

1.1. Introduction 6

2. Estado del arte **9**

2.1. Análisis del mercado y sensores asociados 9

2.2. Análisis de artículos académicos 12

2.3. Oportunidad de innovación 15

3. Descripción del *hardware* **17**

4. Fundamentos teóricos **23**

4.1. Funcionamiento del MPU-6050 23

4.2. *Bluetooth Low Energy* (BLE) 24

4.2.1. Servicios y características 26

4.2.2. Seguridad y emparejamiento del BLE 27

4.3. Protocolo SMTP para el envío de correos electrónicos 28

4.4. Servidor HTTP 31

4.4.1. Petición GET 32

4.4.2. Petición POST 33

4.4.3. Otras peticiones de un servidor web 35

5. Desarrollo del código **37**

5.1. Funcionamiento práctico 37

5.2. Código embebido para el nodo situado en la muñeca 44

5.3. Código embebido para el nodo situado en el brazo 48

6. Funcionalidades adicionales **56**

6.1. Biblioteca NimBLE para Arduino 56

6.2. Aplicación Android/iOS 58

6.3. *Sleep Mode* 58

6.4. Corrección de posturas 59

6.5. Ejercicios del tronco inferior 59

7. Validación y verificación **62**

8. Conclusiones y potenciales aplicaciones **66**

8.1. Conclusions and potential applications 67

Referencias bibliográficas **70**

Anexos **71**

Código correspondiente al nodo inferior 71

Código correspondiente al nodo superior 138

Esquemáticos y *layout* del sistema 169

Vistas del encapsulado del sistema 175

Encapsulado del nodo inferior 175

Encapsulado del nodo superior 177

Datasheets de los componentes 179

Índice de figuras

1. Sistema creado para el registro de picos musculares. Fuente: [19]] 13
2. Funcionamiento del dispositivo GIFT. Fuente: [20] 14
3. Encapsulado del MPU-6050 18
4. Encapsulado del módulo ESP32-WROOM-32D 18
5. Encapsulado del sensor de frecuencia cardíaca 19
6. Diagrama de bloques de la PCB diseñada 20
7. Diseño final del *hardware* implementado 21
8. Distribución de los ejes del MPU-6050. *Fuente: Invensense* 24
9. Esquema de funcionamiento del BLE 27
10. Ejemplo de una cabecera HTTP. Fuente: Ionos 33
11. Pantalla de carga del subsistema principal 37
12. Pantalla de espera del nodo de la muñeca 37
13. Pantalla principal de la *app* 38
14. Inicio de la detección de movimientos 39
15. Serie detectada durante la sesión 40
16. Pantalla de detección de ejercicio 40
17. Pantalla de finalización de entrenamiento 41
18. Pantalla de configuración de la aplicación 41
19. Pantalla de acceso a la base de datos de ThingSpeak 42
20. Análisis del entrenamiento previo visto desde la *app* 43
21. Formato de correo electrónico enviado al finalizar la sesión 43
22. Representación de los lados escogidos para el acelerómetro en el nodo principal 45
23. Procedimiento para la realización del *curl* de bíceps. Fuente: *Outlift* 47
24. Procedimiento para la realización del *press* de hombros. Fuente: *Outlift* . . . 50
25. Pantalla de edición de bloques para la creación de una app 58

Índice de tablas

1. Características del módulo MPU-6050 17
2. Características del ESP32-WROOM-32 18
3. Estructura de mensajes en el protocolo SMTP 30
4. Bibliotecas utilizadas en el sensor de la muñeca 44
5. Relación entre las posiciones del subsistema de la muñeca y sus aceleraciones 46
6. Estados equivalentes a las posiciones del antebrazo y número de ejercicios para cada grupo muscular 47
7. Codificación de mensajes del nodo de la muñeca 48
8. Relación entre las posiciones del subsistema del brazo y sus aceleraciones . . 49
9. Estados equivalentes a las posiciones del húmero y número de ejercicios para cada grupo muscular 51
10. Codificación de mensajes del nodo del húmero 51
11. Códigos de comunicación entre nodos 52
12. Condiciones de ambos nodos para la detección de los ejercicios (I) 53
13. Condiciones de ambos nodos para la detección de los ejercicios (II) 54
14. Comparación de ocupación del almacenamiento del programa para las librerías Arduino BLE y NimBLE-Arduino 56
15. Códigos informativos enviados por la aplicación e interpretados por el sistema 57
16. Condiciones de ambos nodos para la detección de los ejercicios de tronco inferior 60
17. Pruebas realizadas para el entrenamiento de bíceps 62
18. Pruebas realizadas para el entrenamiento de tríceps 62
19. Pruebas realizadas para el entrenamiento de espalda 63
20. Pruebas realizadas para el entrenamiento de pecho 63
21. Pruebas realizadas para el entrenamiento de hombros 64
22. Pruebas realizadas para el entrenamiento de piernas 64

Resumen

Durante el presente proyecto se acomete el procedimiento de creación de un sistema de detección de movimientos de cara a la actividad deportiva, para la monitorización de ejercicios en el entorno del *fitness*. En él, se recorren los diferentes aspectos teóricos que conforman las metodologías de comunicación utilizadas, junto con la descripción del *hardware* implementado y el desarrollo del código que permiten alcanzar el propósito final.

El sistema está compuesto por dos *Printed Circuit Boards* (en adelante, PCBs), con el mismo diseño, y con las únicas diferencias de la existencia de un sensor de frecuencia cardíaca y una pantalla para uno de los nodos. Los componentes principales en los que se basa el algoritmo son un giroscopio más acelerómetro del fabricante Invensense (MPU-6050), que se encargará de la detección de la aceleración y la orientación de cada subsistema y un módulo *WiFi+Bluetooth* de la compañía Espressif (ESP-32-WROOM32D), que realizará las tareas de comunicación entre ambas PCBs mediante protocolo *Bluetooth Low Energy*, así como la conexión que se establece con la aplicación nativa del sistema y el envío mediante protocolo *WiFi* del correo electrónico correspondiente al final de cada sesión con los datos recogidos. El epicentro del sistema, conformado por estos dos circuitos integrados, se sustenta en otros componentes como un gestor de baterías, conversores de voltaje y elementos pasivos. El algoritmo utilizado se crea en lenguaje de programación C++ con la IDE de Arduino, por lo que se utilizan diferentes bibliotecas propias del programa, teniendo una estructura similar para ambos subsistemas, pero con varias funciones adicionales para el nodo principal, que se sitúa en la muñeca. Los movimientos registrados corresponden a ejercicios del tronco superior del cuerpo, por lo que el segundo nodo se coloca en la parte alta del brazo. Esta detección se basa principalmente en el registro de un punto inicial, un punto final, y la aceleración en el recorrido del movimiento, si lo hubiera, además de otros condicionantes. Al mismo tiempo, ambos puntos de análisis consultan la posición de su homónimo para determinar el ejercicio que se está realizando, agrupando los datos obtenidos en series compuestas por las repeticiones hechas. Adicionalmente se incorporan algunos ejercicios del tronco inferior manteniendo el esquema establecido, junto a otros añadidos en fase temprana.

La aplicación diseñada para la gestión del sistema cuenta con diferentes funciones, tales como el inicio de una sesión de entrenamiento, el análisis de las sesiones anteriores mediante la creación de un servidor HTTP o el acceso a la base de datos mediante el portal ThingSpeak, además de una pestaña de configuración con diferentes parámetros. De esta manera, el usuario puede controlar el sistema de forma remota como si de un producto comercial se tratara.

Tras las pruebas realizadas para los diferentes grupos musculares que se programan, se consigue un acierto del 96,62 %, lo que establece un mecanismo innovador y eficaz respecto al análisis realizado para el mercado y los artículos de investigación referentes al tema que se propone.

Abstract

During the present project, the process involved in creating a movement detection system oriented to the monitoring of exercises in the fitness environment is developed. In it, the theoretical aspects of the communication methods implemented are detailed, with the description of the hardware and the explanation of the code that allows reaching the final purpose.

The system is composed of two Printed Circuit Boards (PCBs), with the same design and with the only differences of the existence of a heart-rate sensor and a display for one of the nodes. The main components on which the algorithm is based are: a gyroscope + accelerometer of the manufacturer Invensense (MPU-6050), in charge of the detection of the acceleration and the orientation of each subsystem, and a WiFi+Bluetooth module created by the company Espressif (ESP-32-WROOM-32D), that will do the communication tasks between both PCBs through Bluetooth Low Energy protocol, in addition to the connection that is established with the native application of the system and the sending through WiFi protocol of a corresponding e-mail at the end of the session with the recollected data. The epicenter of the system, composed of these two integrated circuits, is supported by other components such as a battery manager, voltage converters, and passive elements. The used algorithm is created in C++ language with the Arduino IDE, so different own libraries of the program are implemented, having a similar structure for both subsystems but with some additional functions for the primary node that is situated on the wrist. The registered moves are oriented to exercise focused on the high part of the body, so the second node is located on the superior part of the arm. This detection is based on the register of an initial point of analysis, a final point, and acceleration. At the same time, both nodes check the position of the other one to determine the exercise that is happening, grouping the obtained data in series composed by the repetitions done. In addition, some exercises of the low part of the body are implemented, keeping the established scheme with some extra functions in the early stages.

The designed application for the management of the system has different functions: the start of a new workout, the review of previous sessions through creating an HTTP server, access to the database through the ThingSpeak portal, and a configuration tab with different parameters. In this way, the user can control the system remotely like a commercial product.

After the test is done for the different muscle groups that are programmed, the system reaches a success percentage of 96,62 %, so an innovative and effective mechanism is set with regard to the review done of the market and research papers of the topic that is proposed.

Introducción

1. Introducción

En la denominada "revolución tecnológica" que se vive actualmente, donde lo digital está ampliando sus fronteras y ocupando la vida de millones de personas, existe un tipo de tecnología que se expande exponencialmente desde el inicio de la década pasada: el Internet de las Cosas (*Internet of Things*, IoT). Varios ejemplos se pueden encontrar en lo cotidiano, como elementos de seguridad en el hogar, ya sean cámaras, sensores de movimiento o sensores de apertura de puertas, o componentes "vestibles" denominados *wearables*, como relojes, auriculares *true wireless* o pulseras de actividad.

La definición de esta tecnología podría ser la interconexión de elementos cotidianos que generan un servicio agregado, y prueba de su impacto en el mercado actual son los diferentes estudios [1] en los que se ve involucrada, donde se detalla un avance de dispositivos conectados en la década que comprende desde 2015, que registraba una cantidad de 15,41 billones de sistemas, hasta el 2025, con una estimación de 75,44 billones, por lo que la tasa se quintuplicará en tan sólo diez años.

Fruto de las motivaciones personales ligadas a la vida deportiva, se decidió realizar un estudio comparativo entre las diferentes propuestas que se presentan en dicho sector utilizando la tecnología previamente mencionada. Bien es conocida la gran variedad de pulseras de actividad y relojes con capacidades de medición de parámetros de la salud orientados al mundo del deporte. Estos dispositivos son capaces, a día de hoy, de estimar características biomédicas tales como la frecuencia cardíaca o el nivel de oxígeno en sangre, y otras relacionadas con la actividad física, como distancias recorridas en deportes aeróbicos o ritmo medio de desplazamiento. El sistema de detección de estos parámetros en el entorno deportivo es siempre el mismo: el usuario, mediante una interfaz destinada para ello, se comunica con el dispositivo correspondiente para informar del inicio de una nueva sesión de entrenamiento y, una vez finalizado, realiza el mismo procedimiento.

Este mecanismo es válido en la mayoría de deportes comúnmente conocidos, especialmente los aeróbicos, pero existe un tipo de ejercicio que mueve, únicamente en Europa, a más de 60 millones de personas al año: el *fitness* [2]. Para este deporte se puede tomar la variación de pulsaciones que se generan en un entrenamiento, y así poder realizar un análisis de la resistencia a lo largo del tiempo para un mismo tipo de sesión. Además, en determinadas rutinas se pueden aplicar ejercicios de carrera o bicicleta, donde se valora la distancia recorrida. Sin embargo, existen otros parámetros que adquieren una mayor relevancia a la hora de estudiar una sesión de *fitness*: el número de ejercicios que se realizan, cuántas series de cada ejercicio es capaz de aguantar la musculatura y el avance en los pesos que se utilizan. Si el mecanismo de detección de variables que se utiliza convencionalmente se aplica a estos parámetros, el resultado es tedioso y poco efectivo, ya que el usuario debería señalar al *smartwatch* o pulsera de actividad cada serie que va a realizar, con una estimación media de 20 series por sesión.

El objetivo del presente proyecto es resolver dicha problemática mediante un sistema que detecte el ejercicio que se va a realizar, calcule las repeticiones y las agrupe en series, sin que el usuario tenga que informar al dispositivo más que el inicio de un nuevo *workout* (entrenamiento). Además, se busca un sistema lo más simple posible para ajustarse a la comodidad del individuo, acompañándolo de un aplicación nativa que permita la navegación a través de una interfaz donde visualizar los resultados registrados, junto a otras funciones como la comparativa de los entrenamientos realizados para un mismo grupo muscular a través de una base de datos o la corrección de posturas en tiempo real.

Para alcanzar el propósito final de dicho sistema se ha diseñado un *hardware* específico y se ha realizado la programación en el lenguaje C++ adaptado al IDE de Arduino, permitiendo así la detección de ejercicios para la parte superior del cuerpo (espalda, pecho, hombros, bíceps y tríceps) y algunos para la parte inferior. Además, se proporcionan funcionalidades extras como el registro de datos, el envío mediante correo electrónico de los parámetros registrados y la corrección de posturas basada en lógica difusa.

El presente documento recoge todo lo relacionado con el desarrollo del proyecto. En primer lugar se realiza un estado del arte, que se localiza en el capítulo 2, donde se analiza el estado actual del mercado en el entorno de los *wearables* y su aplicación al mundo del *fitness*, para hacer hincapié también en los estudios y artículos de carácter académico aplicados en el sector, con el fin de verificar que el proyecto cuenta con una capacidad innovadora y resolutiva ante un problema existente. En el tercer capítulo se detallan los componentes a nivel de *hardware* que se utilizan en el diseño final, profundizando en las características eléctricas y en detalles concretos de cada elemento. Se añaden determinados fundamentos teóricos en el capítulo 4, que permiten una mayor comprensión de la utilización de protocolos y mediciones, para avanzar durante el quinto capítulo en el desarrollo del algoritmo y el funcionamiento global del dispositivo y de la interfaz. Durante el capítulo seis se entra en detalle en aquellas funcionalidades extras que aporta el sistema más allá del propósito básico. En el séptimo capítulo se realiza una validación del mecanismo y se localizan los principales errores, buscando una solución que permita suavizarlos o erradicarlos en un futuro, seguido de las conclusiones del proyecto. Por último, se incorporan en los dos últimos capítulos la bibliografía utilizada, junto a los anexos, entre los que se encuentran las hojas de datos de los componentes utilizados, los esquemáticos y el *layout* del *hardware* implementado, los planos referentes al encapsulado de las PCBs y el código empleado.

1.1. Introduction

In the called technological revolution that is happening nowadays, where the digital sector is growing and being present in the life of millions of people, there is a type of technology that has been expanding exponentially from the beginning of the last decade: the Internet of Things (IoT). Some examples can be found in an average day, like security elements at home (cameras, motion sensors or door sensors), or wearables like smartwatches, true wireless headphones, or activity bracelets.

The definition of this technology could be the interconnection of daily elements that generate an aggregated service. A proof of its impact in the market can be the different studies [1] where it is involved, where an advance of connected devices between the year 2015 and 2025 is detailed, with an estimation of 75.44 billion devices at that year, five times higher than a decade before.

Product of personal ambitions oriented to sport, a comparative study between the different proposals presented in the sector using the mentioned technology was made: is well known the variety of activity bracelets and smartwatches with the capability of measure health parameters oriented to the sports world. These devices are capable of estimating biomedical characteristics such as the heart rate or the blood oxygen level and others related to physical activity like the traveled distance in aerobic sports or an average speed. The detection system of these parameters in the sports environment is always the same: the user, through a customized interface, communicates to the corresponding device to order the start of a new workout session, and when it is finished, the person does the same procedure.

This mechanism is valid for the majority of known sports, especially aerobics. Still, more than 60 million people practice a type of exercise per year only in Europe: fitness. For this sport, the heart rate generated during a session can be taken to analyze the pulmonary resistance over time for the same characteristics of the training. Moreover, in some routines, it can be applied running exercises or bicycle, where the traveled distance can be evaluated. However, two parameters have more relevance when a session of fitness is studied: the quantity of exercises that are done, how many series of each exercise is capable of resisting the muscle group, and the advance of the weights that are used. If the variables detection mechanism that is used conventionally is applied to these parameters, the result is tedious and presents low effectiveness because the user would have to indicate the smartwatch or activity bracelet the series that is going to do, with an estimate of twenty series per session.

The present project aims to resolve that problem through a system capable of detecting the exercise that is going to do, calculating the repetitions, and grouping them in series, without the user's interaction beyond indicating the start of a session. In addition, the project looks for the most straightforward possible system to adjust for every person, with the creation of a native application that allows through an interface the review of the registered results and other functions like the comparative between sessions for the same muscle groups through a database or the posture correction in real-time.

To reach the final goal of the system-specific hardware has been designed, and the programming in C++ language adapted to Arduino IDE has been made for it, allowing the detection of exercises for the high part of the body (back, chest, shoulders, biceps, and triceps) and some for the low part. Moreover, some extra functionalities like the register of date, the sending through an email of the registered parameters, and correcting the posture based on fuzzy logic are created.

The present document recollects all the parts related to the development of the project. First of all, a state-of-art is done located in chapter 2, where the actual state of the market in wearables environment is analyzed, and its application in the fitness world; also focusing on the studies and papers with academic purposes applied to this sector checking that the project has an innovative capability and to resolve an existing problem. In the third chapter, all the components involved in the final hardware are detailed, going deeper into electric characteristics and concrete details of each element. Some theoretical concepts are added in chapter 4 that allow a better comprehension of the use of protocols and measurements to develop the algorithm and the global working of the algorithm through the device and the interface in chapter 5. During the sixth chapter, the extra implemented functionalities are detailed, and chapter seven includes the validation of the mechanism and the location of the main errors for the correction in the future, following the conclusions of the project. The last part is done for the bibliography and attachments, where the datasheets of the used components are located, with the schematics and layout of the implemented hardware, the drawings for the packaging of the PCBs, and the final code.

Estado del arte

2. Estado del arte

Para la creación de un proyecto que cumpla con un objetivo de innovación, y que permita solucionar una problemática existente en el mercado actual, se realiza una investigación acerca del estado de este y de las posibles alternativas que surgen en el mundo del I+D+i.

2.1. Análisis del mercado y sensores asociados

Dentro de la gran variedad que existe en la gama de la denominada "tecnología invisible" aparecen una serie de sensores que se repiten para la mayoría de casos que se presenta. En las pulseras de actividad y relojes inteligentes, que son los que principalmente se ajustan al actual proyecto, se suele encontrar un sensor de frecuencia cardíaca. Es el caso de la *Xiaomi Mi Band 6* [3], la *Honor Band 5* [4] o la *Huawei Band 4 Pro* [5], que son los principales referentes del sector en la gama de entrada de las pulseras de actividad. Además, este tipo de sensor se encuentra en dispositivos de mayor coste económico, como el *Apple Watch Series 6* [6], el *Oppo Watch* [7] o el *Samsung Galaxy Active 2* [8], siendo estos últimos los buques insignia en términos de *wearables* que aportan estas tres de las más importantes marcas tecnológicas del planeta. Estos últimos, además, incluyen otros sensores de mayor envergadura tanto para el ámbito sanitario como para el común, como medidores de presión arterial, electrocardiograma (ECG), sensores de luz y ruido ambiental o la brújula. Esto se aplica para el resto de marcas del sector que orientan sus productos a un mercado generalista, lejos de objetivos poblacionales concretos.

Para este tipo de nicho de mercado se aplican una serie de sensores dedicados que permiten una experiencia deportiva más óptima y segura. Prueba de esto son los medidores de presión atmosférica y temperatura, que permiten la visualización en tiempo real de estos parámetros sin requerir de una conexión a internet como en los dispositivos convencionales. Además, es común encontrar sensores barométricos en este tipo de productos, por ejemplo, en el *Suunto Ambit3 Peak* [9] o el *Casio SGW-1000-2ber* [10], dos *smartwatches* de alto poder adquisitivo que están orientados al uso en senderismo.

El mecanismo de registro de variables en función al deporte que se realiza es similar para todos aquellos dispositivos de ámbito generalista, y las pequeñas diferencias que se encuentran entre aquellos de diferentes rangos de precio no aportan un valor diferencial para el propósito del proyecto que se presenta.

Para la gama de entrada, las pulseras de actividad son capaces de monitorizar hasta 30 tipos de deporte, como es el caso de la *Xiaomi Mi Band 6*. El dispositivo es capaz de diferenciar, en función a estos, los momentos en los que se trabaja de forma intensa o más relajada, utilizando el sensor de frecuencia cardíaca como referencia. Esta última versión de la pulsera incluye un modo "Gimnasio Interior", que se entiende como *fitness*, exportando de igual manera un estudio de la actividad en función a los latidos del corazón, junto a un resumen de calorías consumidas, por lo que no se tiene constancia de una detección de ejercicios como tal. Este ejemplo se puede extrapolar al resto de *wearables* que se encuentran en esa gama de precio, sin importar el tipo de marca, como es el caso de la *Yamay Fitness Tracker* [11] o la *Honor Band 5*. La más completa es la *Huawei Band 6* [12], aunque también de mayor coste, que incorpora un sensor de oxígeno en sangre y la capacidad de registrar hasta 96 tipos de deporte diferente, aunque el registro para el uso de gimnasio se ve limitado a lo visto anteriormente.

Si se busca en un entorno de mayor poder adquisitivo, se encuentran los dispositivos mencionados anteriormente de la marca *Apple*, *Samsung*, *Oppo* o la propia *Huawei*. El *smartwatch* que presenta Apple es capaz de detectar un entrenamiento de forma automática en función al movimiento del usuario y el aumento de las pulsaciones en un período de 10 minutos, así como de programar los entrenamientos para evitar el inicio manual de forma repetida. El número de ejercicios detectables se reduce a 15, y el mecanismo se resume a lo visto hasta ahora: un análisis del tiempo de entrenamiento, pulsaciones medias y máximas, número de calorías consumidas activas y número de calorías consumidas totales. Para determinados ejercicios aérobicos se incluye la posibilidad de determinar la distancia recorrida. En el caso del *fitness*, sólo se obtiene lo básico, por lo que no existe ningún sistema de detección de ejercicios concretos. El *Apple Watch Series 6* es uno de los relojes más avanzados del sector, por lo que, tal y como se comprueba, es de esperar que otros modelos como el *Samsung Galaxy Active 2*, el *Garmin Venu* o el *Huawei Watch 3 Series* adopten un registro y exportación similar o inferior a lo que se encuentra en el modelo de la manzana mordida.

Es por ello que, a nivel de usuario medio, no se encuentra un dispositivo, en ninguna gama de precios, que se ajuste al propósito que se desea presentar en en el documento actual. Por tanto, se deberá realizar un estudio de aquellos productos dedicados exclusivamente al entorno del *fitness*.

El *Atlas Multitrainer 3* [13] es el sucesor natural de la *Atlas Wristband 2*, un *wearable* que presume de un *tracking* del ejercicio eficaz y que promete facilitar la recuperación post-entreno. Entre sus cualidades, además, está la de detectar repeticiones de los ejercicios del *workout* que se realiza. La manera en la que funciona esto se resume en una comunicación previa, por parte del usuario, de los ejercicios que se van a ejecutar, junto a las repeticiones correspondientes a cada serie, y el número de series de cada uno. El reloj, que cuenta con un acelerómetro, cuenta las repeticiones utilizando los cambios de aceleración y los añade al registro. Sin embargo, no cuenta con un sistema de detección de estos ejercicios, por lo que el individuo debe señalar antes de cada sesión la rutina a realizar.

Otro de los productos que se pueden encontrar es la *PUSH Band* [14], una banda que se sitúa en el antebrazo y que comparte muchas características con el *Atlas Multitrainer*. Incluye la monitorización de la potencia, la velocidad y el trabajo realizado durante la sesión de entrenamiento, y un portal web para acceder a los datos. Además, promete hasta un total de detección de 250 ejercicios. Sin embargo, esta detección de ejercicios no es más que lo visto anteriormente, por lo que se trata de una detección de repeticiones de ejercicios previamente programados por el usuario, no un sistema que detecte de forma autónoma los ejercicios realizados.

El tercero en discordia es el dispositivo denominado *Beast* [15], una banda que se coloca en la muñeca y que se conecta a un elemento magnético que se adhiere a las barras donde se colocan los pesos. Cuenta con un *app* donde se puede consultar los registros y donde se visualiza la monitorización de las repeticiones. El sistema es el mismo que los otros dispositivos mencionados, una detección que depende de una preprogramación por parte del usuario, rompiendo con la fluidez y naturalidad que requiere un entrenamiento.

Por último, está el dispositivo multifuncional *Gymwatch* [16] y *Athos* [17]. El primero es una banda que se puede situar en cualquier extremidad, aportando además una completa aplicación que permite la incorporación de nuevos ejercicios a la hoja de ruta. Cuenta con un botón que habilita un contador hasta el inicio de la rutina. No es un producto orientado a la sesión de entrenamiento como tal, sino al cuidado de la integridad del individuo, y prueba de esto es que realiza un *feedback* con la calidad de los movimientos que se realizan para evitar lesiones y mejorar la ejecución, midiendo a su vez la tensión muscular. La detección de repeticiones repite el esquema que se encontraba hasta ahora. Sin embargo, este producto ha quedado obsoleto a día de hoy y no dispone de ningún sucesor previsto. El segundo son dos prendas deportivas, una malla y una camiseta, que cuentan con 20 sensores capaces de analizar la actividad muscular durante el ejercicio físico. No está orientado a la detección de movimiento o repeticiones, únicamente a la detección de ritmo cardíaco, respiración y capacidades musculares, por lo que su objetivo es el cuidado personal. Durante la sesión de entrenamiento sólo se muestra el esfuerzo realizado por cada grupo muscular sin importar el ejercicio, ya que tampoco cuenta con una programación previa.

2.2. Análisis de artículos académicos

Se demuestra, por tanto, que no se encuentran alternativas en el mercado que realicen una detección similar a lo que se pretende implementar en el presente proyecto, ya sea a nivel general como a productos orientados al sector del *fitness*. Es por ello que se realiza una investigación poniendo el foco en estudios y artículos de carácter académico con una fecha de publicación posterior a 2016.

Es en ese año cuando se crea un sistema de origen estadounidense denominado *StayFit* [18] y cuyos autores son Maram Maheedhar, Aman Gaurav, Vivek Jilla, Vijay N Tiwari y Rangavittal Narayanan. Para la creación del producto, mezclan conceptos de potencia y capacidad cardiorrespiratoria, que dan lugar a la detección de las repeticiones de ocho ejercicios diferentes para los grupos musculares de mayor tamaño, alcanzando una sensibilidad del 96 % y una precisión del 95 %, que dan como resultado un sistema similar a lo que se desea implementar en el presente proyecto mediante la programación de un reloj inteligente (*Samsung Galaxy Gear S2*), que opera a 25 Hz, con la desventaja de la limitación que requiere el utilizar un único punto de análisis. El número de ejercicios es reducido, y por tanto, inaplicable en un entorno real.

En el año 2017, Chaeolhyo Lee presentó en la Novena Conferencia de Redes del Futuro una algoritmo capaz de medir los valores picos de resistencia que genera la musculatura correspondiente en las actividades físicas propuestas [19], aunque sin entrar en términos de detección de ejercicios, ya que se conectan sensores en las diferentes partes del cuerpo y se mide la tensión muscular y la aceleración en los movimientos, detectando picos positivos y negativos, por lo que no se ajusta a lo que se desea alcanzar en el proyecto actual. El test práctico realizado para este sistema se puede visualizar en la figura 1.

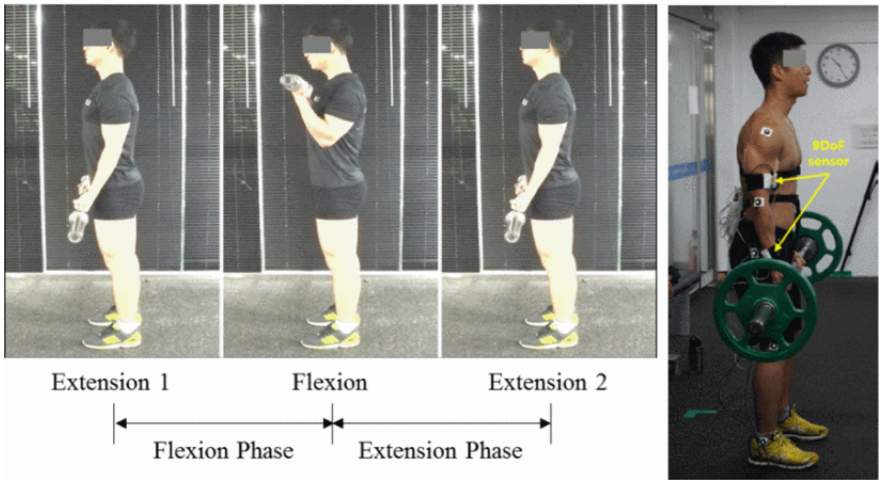


Figura 1: Sistema creado para el registro de picos musculares. Fuente: [19]]

Un año más tarde se presenta un dispositivo denominado *GIFT* [20], un guante destinado al *fitness indoor* que realiza una detección automática del ejercicio en función a la presión que se registra en los diferentes puntos de la mano (figura 2). Sus autores, Elder A. H. Akpa, Masashi Fujiwara, Yutaka Arakawa, Hirohiko Suwa y Keiichi Yasumoto, indican una precisión del 87 % para los diez ejercicios que registraron, conectando el guante a una unidad de muestreo de datos que recoge la presión en cada sensor y realiza una comparación con otros ejercicios. Al igual que en el primer artículo mencionado, al tener un único punto de análisis, se limita la detección de una gran cantidad de ejercicios, a pesar de la futura inclusión de un acelerómetro que se plantea en el artículo.

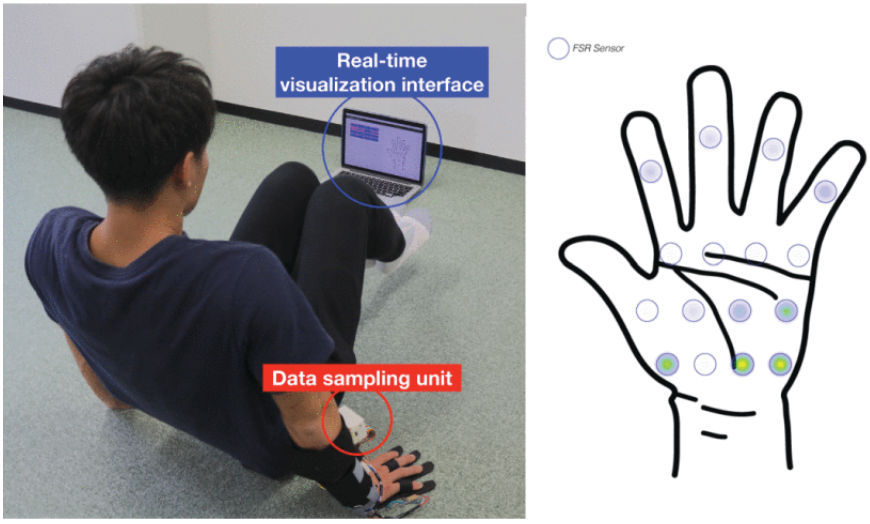


Figura 2: Funcionamiento del dispositivo GIFT. Fuente: [20]

En junio de 2019, se presenta un sistema de análisis en tiempo real de entrenamientos de *fitness* utilizando *machine learning* combinado con visión por ordenador [21]. Sus autores, Amit Nagarkoti, Revan Teotia, Amith K. Mahale y Pankaj K. Das realizan una comparativa del movimiento que realiza un sujeto utilizando una grabación y una serie de vectores que unen las diferentes extremidades con el torso de este. El sistema está orientado a la corrección de posturas de cara a la mejora de la ejecución del ejercicio y a la reducción de lesiones ocasionadas por estos. No existe detección automática, sino una comparación entre los movimientos de un profesional con el usuario. Esto no se ajusta al propósito del presente proyecto y, además, no resulta práctico de cara a un entorno real.

2.3. Oportunidad de innovación

Se finaliza, por tanto, el análisis del mercado, junto a los artículos académicos que se han publicado relacionados con el *fitness*, donde se puede concluir que no existe un sistema como el que se pretende implementar en el presente proyecto. Es cierto que existen dispositivos capaces de realizar una detección de repeticiones o de ejercicios, pero presentan varios problemas. El primero, que requieren de una programación previa, sobre todo en el ámbito comercial, por lo que no resulta cómodo y limita la fluidez que debe tener un entrenamiento de estas características. El segundo, que los dispositivos que se han creado para la detección de ejercicios se han testeado con un número muy reducido de estos, utilizando un único punto de análisis, lo que limita en gran medida la exportación al mundo real, donde muchos se deben detectar desde otro punto, con la ayuda de la información que se recoge en otro u otros sensores. A pesar de los altos porcentajes que pueden presentar, no se puede extrapolar a una sesión real en un futuro, mientras sólo se tome un punto de detección. Además, el campo que se estudia se encuentra en una fase muy temprana debida a la escasez de artículos publicados en este sector.

El presente proyecto, por tanto, puede solucionar la problemática existente. Por una parte, cuenta con un sistema autónomo que detecta los ejercicios sin necesidad de una programación previa, únicamente con la orden del inicio del entrenamiento. Por otra, al contar con dos puntos de análisis, se abre el abanico de registro de ejercicios, hasta un total de 36, aportando un valor diferencial respecto a los artículos publicados hasta ahora.

Descripción del hardware

3. Descripción del *hardware*

Previa implementación del *software* necesario para la detección de ejercicios y otros parámetros, se realiza el diseño completo del *hardware* utilizando la plataforma *Altium Designer*. Para el objetivo del proyecto, se debe tener en cuenta que se precisa de un acelerómetro y giroscopio, para determinar la aceleración y orientación de los puntos de estudio en cada instante de tiempo. Además, se requiere de un circuito integrado que permita establecer la comunicación *Bluetooth* entre los diferentes puntos, de tal manera que se estén enviando continuamente mensajes con la posición de estos. Es de utilidad incorporar una pequeña pantalla que sirva de ayuda visual ante problemas cotidianos de comunicación entre nodos o para visualizar datos como el tiempo de entrenamiento. Además, se instala un sensor de frecuencia cardíaca, junto a otros componentes menos visibles en la práctica, como convertidores de voltajes, elementos pasivos o un convertidor de protocolos USB-UART. Por tanto, se cuentan con los siguientes componentes principales:

- **MPU-6050**: este circuito integrado de tipo *smart-sensor* consta de un acelerómetro y un giroscopio, de tres ejes cada uno, por lo que se tienen 6 grados de libertad. La función de este módulo es la de detectar la posición del antebrazo o el brazo de forma discreta en cada instante de tiempo. Cuenta con unas dimensiones en milímetros de $4 \times 4 \times 0{,}9$ (alto $\times$ ancho $\times$ espesor), perteneciente al fabricante *Invensense*. Cuenta con un amplio abanico de tensiones a las que es capaz de funcionar. Sus principales características se pueden observar en la tabla 1, mientras que en la figura 3 se puede visualizar el encapsulado del componente:

Característica	**Valor**
Voltaje de alimentación, Vdd	-0.5 V a 6 V
Voltaje de entrada, Vlogic	-0.5 a Vdd + 0.5 V
Aceleración en cualquier eje	10000 g para 0.2 ms
Rango operativo de temperatura	-40ºC a +105ºC
Rango de temperatura de almacenamiento	-40ºC a +125ºC

Tabla 1: Características del módulo MPU-6050

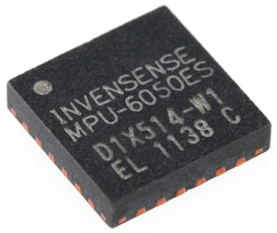


Figura 3: Encapsulado del MPU-6050

- **ESP32-WROOM-32D**: módulo *WiFi* + *Bluetooth* que se utilizará para la comunicación entre ambos nodos y el *smartphone* correspondiente. Diseñado y fabricado por *Espressif*, cuenta con los protocolos 802.11 b/g/n para el caso del *WiFi* y la versión 4.2 de *Bluetooth*. Además, incorpora compatibilidad para tarjetas SD, protocolo UART, puerto serie y leds, teniendo unas dimensiones de $18\pm0{,}2\,\text{mm}\times25{,}5\pm0{,}2\,\text{mm}\times3{,}1\pm0{,}15\,\text{mm}$. En la tabla 2 se observan algunas de las características más importantes del módulo, además de el encapsulado general del módulo que se muestra en la figura 4:

Item	**Especificación**
Voltaje de operación	2.7 V a 3.6 V
Corriente de operación	Media: 80 mA
Seguridad Wi-Fi	WPA/WPA2/WPA2-Enterprise/WPS
Protocolo Bluetooth	Bluetooth 4.2 BR/EDR y especificación BLE
Rango operativo de temperatura	-40ºC a +85ºC

Tabla 2: Características del ESP32-WROOM-32

Figura 4: Encapsulado del módulo ESP32-WROOM-32D

- **MAX30102EFD+**: este módulo (Figura 5) hace referencia a un sensor de frecuencia cardíaca, que va a estar destinado al registro de este parámetro durante la sesión de entrenamiento, para posteriormente establecer un análisis de los mínimos, máximos y frecuencia media. Necesita de una tensión mínima de 1,8 V y una máxima de 2,0 V. En el presente proyecto, el sensor de pulso estará integrado en el subsistema de la muñeca, con unas dimensiones de $5{,}6 \times 3{,}3 \times 1{,}55$ mm^3, realizándose únicamente la implementación a nivel de *hardware* de dicho módulo, limitándose su programación a futuribles proyectos que envuelvan al sistema.

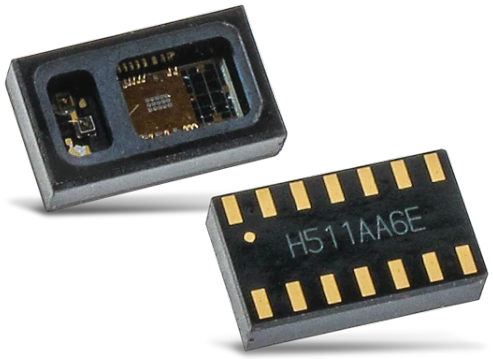


Figura 5: Encapsulado del sensor de frecuencia cardíaca

Además, se integran una serie de componentes que serán necesarios para el correcto funcionamiento de los anteriormente mencionados. Estos son:

- **TP4056**: gestor de baterías que se encarga de recargar la batería de litio-polímero (LiPo) para el uso continuado del sistema de manera inalámbrica. Tiene un rango de funcionamiento de -4 V a 8 V, aunque el uso típico es de 5 V, y un rango operativo de temperatura de $-40\,^{\circ}$C a $85\,^{\circ}$C.

- **TX0102DCTR**: este circuito integrado realiza la función de traslación de voltaje de forma bidireccional. Conecta los puertos serie del MPU-6050, del ESP32-WROOM y del *display*, que funcionan a 3,3 V, y del MAX30102EFD+, que funciona a 1,8 V.

- **TLV75533PDBVR**: convertidor de tensión que reduce la señal de 4,2 V a 3,3 V, necesarios para el funcionamiento del acelerómetro, de la pantalla y del módulo *WiFi* y *Bluetooth*. Cuenta con un rango de operación de $-0{,}3$ V a 6 V y un rango operativo de temperatura de $-40\,^{\circ}$C a $150\,^{\circ}$C.

- **NCP612SQ30T2G**: convertidor de tensión que se encargará de reducir la tensión de 3,3 V a 1,8 V, que es el voltaje de funcionamiento del sensor de frecuencia cardíaca. Su rango de temperatura va desde los $-40\,°C$ a los $85\,°C$ y permite un rango operativo de voltajes de 0 a 6,0 V, aunque su funcionamiento como tal debe estar en un margen de $\pm 0{,}3 V_{IN}$.

En el capítulo referente a los Anexos se adjuntan las hojas de datos de los componentes utilizados para la creación del proyecto. Como se puede observar, se tienen cuatro ramas principales de voltaje. La primera es la que funciona a 5,0 V, necesaria para la conexión USB y el funcionamiento del conversor USB a protocolo UART (CP2012). Tras esto, se convierte el voltaje a 4,2 V, ya que será la máxima tensión de funcionamiento a las que trabajan las baterías de litio-polímero, para posteriormente reducirlo a 3,3 V. Es aquí donde se encuentra todo el eje central del *hardware*, ya que es el voltaje de funcionamiento del acelerómetro y giroscopio (MPU-6050), del módulo *WiFi* + *Bluetooth* (ESP32-WROOM) y del *display* que se incorpora en el diseño. La última rama es la de 1,8 V, necesaria para el uso del sensor de frecuencia cardíaca (MAX30102EFD+). En la figura 6 se puede visualizar un esquema con los principales elementos que se integran en el sistema.

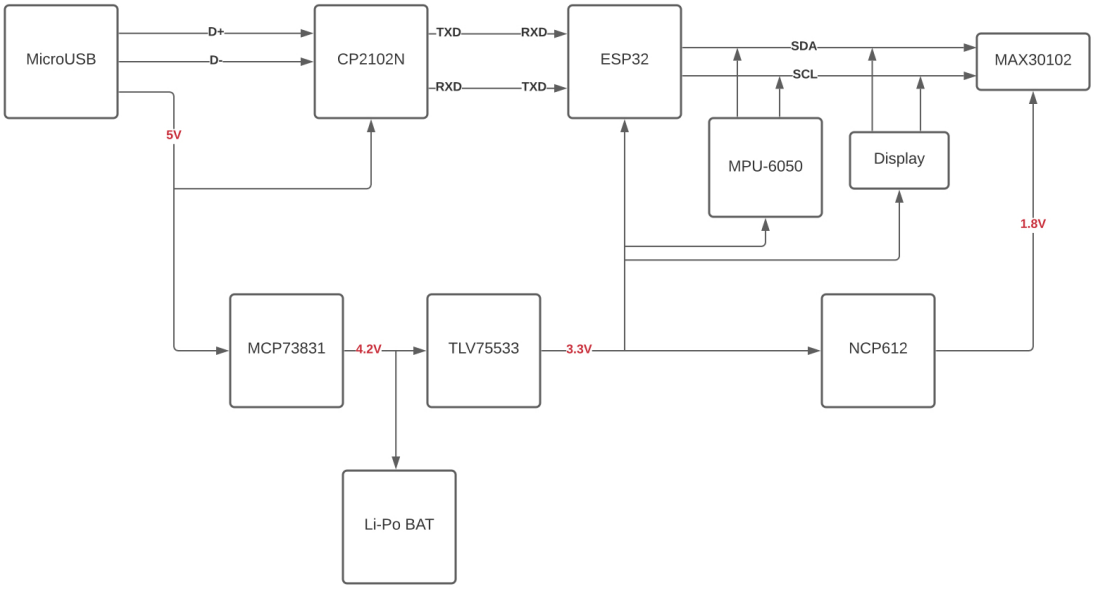


Figura 6: Diagrama de bloques de la PCB diseñada

En la figura 7 se puede observar el diseño *hardware* final con todos los componentes implementados en las PCBs, donde cada una tiene unas dimensiones en milímetros de 49×32.

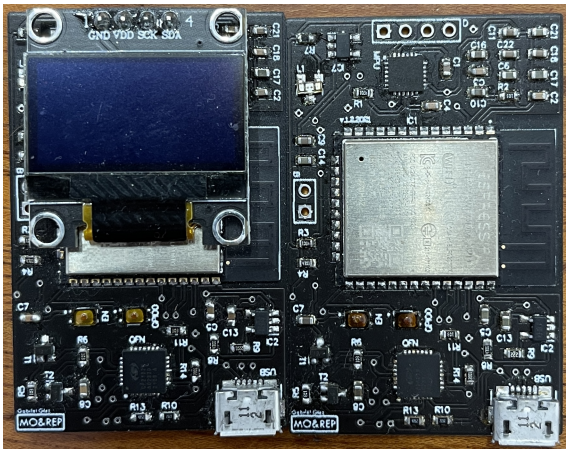


Figura 7: Diseño final del *hardware* implementado

Fundamentos teóricos

4. Fundamentos teóricos

Durante el presente capítulo se realiza una descripción exhaustiva de los principales fundamentos teóricos en los que se basan los elementos del proyecto, entre los que se encuentran los mecanismos del acelerómetro y giroscopio, los esquemas en los que se basa el *Bluetooth* de baja energía (BLE) y el protocolo SMTP utilizado para el envío de correos electrónicos, así como la estructura de un servidor HTTP.

4.1. Funcionamiento del MPU-6050

La aceleración se define como la variación de la velocidad por unidad de tiempo, representada matemáticamente como:

$$a = \frac{dv}{dt} \tag{1}$$

Si se aplica la segunda ley de Newton, que se define como el producto de la masa de un cuerpo por la aceleración a la que se ve sometido, se obtiene el funcionamiento de un acelerómetro.

$$F = m \cdot a \tag{2}$$

Estos contienen en su interior un sistema microelectromecánico (mEMS), que actúa de una manera similar a la de un resorte. El dispositivo es capaz de detectar la aceleración de la gravedad, por lo que nunca se mostrará un valor nulo de movimiento. Además, debido a las capacidades propias del acelerómetro, se pueden calcular de manera indirecta otros parámetros como la velocidad o el desplazamiento.

Por otro lado, la velocidad angular se define de manera analítica como la diferencia de desplazamiento angular en un determinado instante de tiempo, por lo que se puede identificar de forma empírica como la velocidad a la que gira un cuerpo alrededor de su eje.

$$\omega = \frac{d\theta}{dt} \tag{3}$$

El sistema microelectromecánico que se encuentra en componentes como el MPU-6050 utiliza el efecto Coriolis [21] para hallar este parámetro, permitiendo la medición indirecta del desplazamiento angular.

Sumando ambos conceptos se puede visualizar la idea del funcionamiento del IMU implementado, el MPU-6050. En la figura 8 se muestra la distribución de los ejes en los que se calculan las diferentes aceleraciones lineales y angulares y su relación posicional con el módulo.

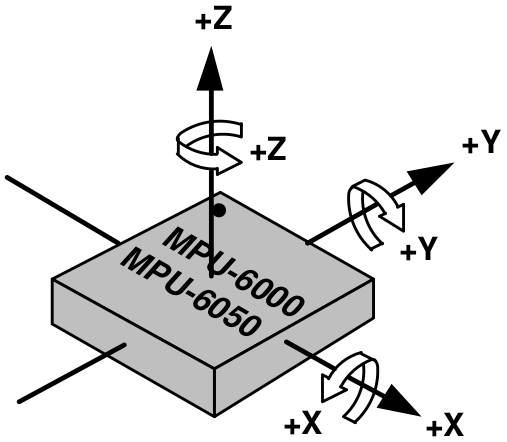


Figura 8: Distribución de los ejes del MPU-6050. *Fuente: Invensense*

Durante el proyecto que se presenta se implementa el módulo mencionado para determinar las posiciones que adoptan los diferentes puntos de análisis para su posterior comunicación vía *Bluetooth*.

4.2. *Bluetooth Low Energy* (BLE)

El *Bluetooth* de baja energía o BLE es un subconjunto del estándar *Bluetooth* v.4.0. Dispone de un grupo de protocolos respecto al modelo OSI orientada a conexiones sencillas para aplicaciones que requieren muy poca potencia, lo cual lo hace perfecto para el entorno del Internet de las Cosas, ya que su mayor uso engloba dispositivos que funcionan a base de pilas o baterías, como el ejemplo que se presenta.

Cuenta con una amplia aceptación y compatibilidad con los diferentes sistemas operativos que dominan el mercado, tanto en dispositivos móviles como en *wearables* u ordenadores. Siguiendo el esquema OSI mencionado anteriormente, se pueden identificar los diferentes protocolos según la capa a la que haga referencia:

- La **capa física** define el cableado y circuitería que conforma el sistema de comunicación. En ella se realizan los procesos de modulación y demodulación de las señales analógicas para convertirlas a señales digitales. La tecnología BLE permite el uso de 40 canales de 2 MHz en la banda ISM de 2,4 GHz. Dicho estándar realiza la técnica conocida como *frequency hopping*, que se traduce en saltos de frecuencia pseudoaleatorios entre los diversos canales que se mencionan, lo que crea un alto nivel de fiabilidad de cara a posibles interferencias.

- En la **capa de enlace** se gestionan determinadas características temporales, como la verificación de mensajes y reenvío de erróneos recibidos, el filtrado de direcciones, etc. Cuenta con una identificación de papeles o roles por dispositivo en el proceso de comunicación, que son el *Advertiser*, el *Scanner*, el *Master* y el *Slave*.

- En la **capa de red** se encuentra la HCI o *Host Controller Interface*, que permite la comunicación entre un servidor o *host* y un controlador mediante una interfaz serie. De esta manera se define como el conjunto de comandos y eventos para llevar a cabo el proceso de interacción entre ambos elementos.

- En la **capa de transporte** se encuentra el *Logic Link Control and Adaptation Protocol* o L2CAP, que tiene como objetivo el proceso de multiplexación (el envío de paquetes provenientes de niveles superiores en el modelo OSI para encapsularlos en paquetes estándar BLE, y viceversa), además de la fragmentación y recombinación. De esta manera, los paquetes que provienen de la capa de aplicación ocupando una cantidad muy alta de bytes, son fragmentados ajustándose a la unidad máxima de transferencia de paquetes que requiere el estándar (27 bytes de carga útil o *payload*). La capa L2CAP, además, se encarga de permitir el acceso y dar soporte al protocolo ATT (*Attribute Protocol*), que permite el intercambio de información utilizando una arquitectura servidor-cliente basada en "atributos", y el SMP o *Security Manager Protocol*, que proporciona un *framework* para generar y distribuir claves de seguridad entre los diferentes dispositivos.

- Para la **capa de sesión** se encuentran paralelamente las capas GAP (*Generic Access Profile*) y GATT (*Generic Attribute Profile*). La primera permite la visibilización de un dispositivo respecto al resto de estos, determinando cómo pueden interactuar entre ellos. Para la realización de este propósito la capa establece una serie de normas y conceptos para estandarizar las operaciones de niveles inferiores como:

  - Roles de interacción
  - Modos de operación y transición entre dispositivos
  - Procedimientos para el establecimiento de la comunicación
  - Modos de seguridad y procedimientos

  Por otro lado, la capa GATT realiza la función de definir cómo se transfiere la información entre dos dispositivos BLE. Esto se realiza una vez se ha determinado el establecimiento de la comunicación. [22]

La principal diferencia entre el BLE y el *Bluetooth* convencional reside en la cantidad de energía que se requiere para la utilización de uno u otro. El *Bluetooth* clásico se diseñó para la transmisión continua de datos, por lo que se precisa de una fuente generosa de batería, careciendo de utilidad en dispositivos de menor tamaño.

Ambos operan en la banda ISM de los 2,4 GHz. Sin embargo, el BLE permanece en modo suspensión de manera constante, salvo cuando se inicia la conexión, lo que tarda unos pocos milisegundos, a diferencia de los más de 100 milisegundos que tardaría el *Bluetooth* clásico. Para el uso de menor energía los dispositivos involucrados pueden ser centrales, como *smartphones*, *tablets* u ordenadores, capaces de ejecutar un número mayor de procesos, encargándose de controlar los periféricos, que suelen identificarse como sensores capaces de registrar datos y enviarlos a los dispositivos centrales, por lo que estos últimos son los encargados de procesarlos convenientemente, reduciendo la energía requerida por los periféricos.

4.2.1. Servicios y características

Las transacciones GATT en BLE están basados en objetos conjuntos denominados perfiles, servicios y características. En la figura 9 se puede observar como está distribuido el funcionamiento del *Bluetooth* de baja energía.

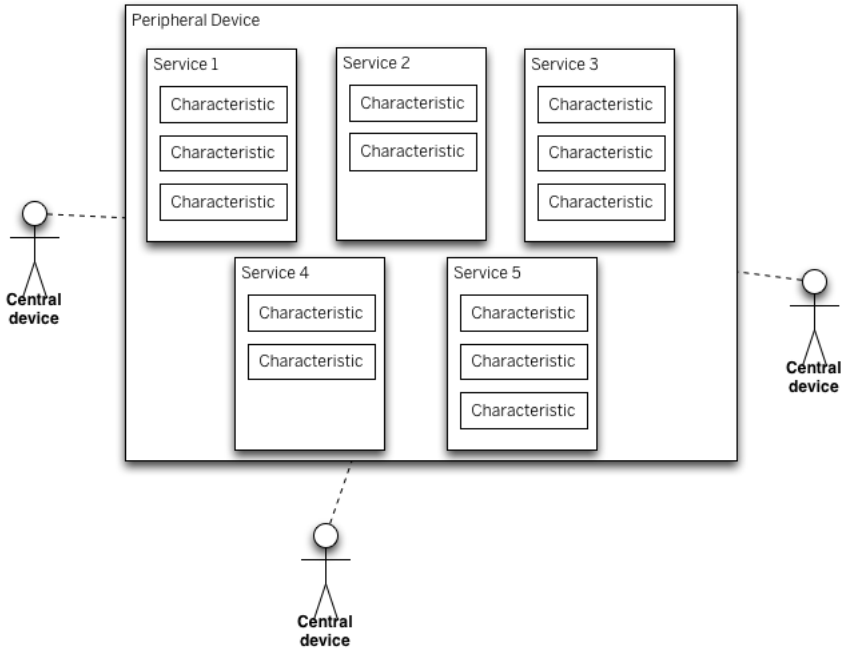


Figura 9: Esquema de funcionamiento del BLE

Un conjunto predefinido de servicios forman un perfil, y cada servicio se utiliza para dividir datos en entidades lógicas, conteniendo porciones específicas de datos llamados características. Un servicio, por tanto, podrá tener varias de dichas características, que se diferenciarán entre sí por un ID numérico denominado UUID, formados por 16 bits en el caso de los servicios BLE que se utilizan oficialmente, o 128 bits para aquellos personalizados. Las características son el concepto más bajo en las transacciones GATT, ya que encapsulan un único tipo de dato, aunque puede contener un vector en él [23].

4.2.2. Seguridad y emparejamiento del BLE

Las dos principales características que presenta el BLE en términos de seguridad son la autenticación y el cifrado de 128 bits, además de los saltos de frecuencia, que permiten la corrección de errores hacia adelante (FEC, *Forward Error Correction*), por lo que el receptor corrige errores de transmisión sin necesidad de reenvíos.

Una vez realizada la verificación en la conexión de dos o más dispositivos, la conexión es realmente segura, aunque, para ello, se precisa de una serie de pasos de cara al emparejamiento.

- La primera etapa consta de un intercambio de información básica en los que cobra importancia las capacidades de los integrantes de la red, para encontrar el procedimiento óptimo de conexión. Esta etapa no conlleva ningún tipo de encriptación.

- La segunda etapa de emparejamiento está orientada a la generación e intercambio de claves, siendo esta la fase de mayor vulnerabilidad del proceso, ya que las conexiones pueden ser manipuladas. Aquí aparecen métodos de seguridad para ejecutar el procedimiento de manera fiable, como el algoritmo *Diffie-Hellman* para la generación de claves, introducido en la versión 4.2 del BLE. Muchos de los dispositivos pertenecientes al IoT no cuentan con este tipo de seguridad, por lo que son susceptibles a escuchas pasivas o al conocido *Man-In-The-Middle*.

- En la última fase del proceso los dispositivos almacenan los datos de autenticación que se intercambiaron previamente, lo que permite el emparejamiento en casos posteriores al quedarse guardados dichos datos.

Los cuatro métodos principales de emparejamiento en la tecnología BLE son los denominados *Just Works*, *Out of Band*, claves de paso y comparación numérica.

En el presente proyecto se utiliza la tecnología *Bluetooth Low Energy* para la comunicación constante de los diferentes nodos que se utilizan durante el ejercicio, enviando y reconociendo las posiciones que ocupa cada uno en cada instante de tiempo, así como la conexión y emparejamiento con el dispositivo móvil correspondiente.

4.3. Protocolo SMTP para el envío de correos electrónicos

Al finalizar cada sesión de entrenamiento se envía un correo electrónico con el análisis de lo realizado. Para ello, se precisa del protocolo SMTP (*Simple Mail Transfer Protocol*), que utiliza los estándares RFC 821, 2821 y 5321 [22].

Dicho protocolo se encuentra directamente relacionado con otros comunes como el POP3, que usa el estándar RFC 1939, o el IMAP, que aplica el RFC 3501. La diferencia entre estos radica en el propósito de cada uno, ya que el SMTP suele utilizarse para correos electrónicos de salida, mientras que POP3 e IMAP se usan para correo entrante.

Para la transmisión de mensajes se definen una serie de normas, a modo de rutina, que conforman el estándar comunicativo. Estas son:

- Se requiere un origen, que se identifica como el cliente del correo emisor.
- El correo navega por internet a través de la conexión ISP.
- Se realiza una autenticación en el servidor SMTP llegando al servidor correspondiente.
- Cuando el servidor procesa el correo lo envía a los receptores de destino definidos.

En un cliente de correo convencional, el servidor de destino (*mail exchanger*) almacena la información, para que el propio cliente la recoja mediante protocolos POP3 o IMAP, enviándolo posteriormente a los destinatarios. En el popular ejemplo del *webmail*, el correo llega al servidor, que se encuentra conectado a internet, para ser visualizado a través del navegador o de la interfaz correspondiente.

El SMTP utiliza el puerto 25 como estándar, aunque el 465 se encuentra en un auge de popularidad de uso, siendo el utilizado para el presente proyecto. Además, otras plataformas orientadas al *marketing* como *Mailrelay* usan los puertos 125 o 2525. La diferencia entre los mencionados radica en la seguridad: los puertos 25 y 587 se encuentran por defecto en el protocolo SMTP como desencriptados. En todos se necesita una autenticación con usuario y contraseña. Sin embargo, el 465 añade una conexión cifrada SSL/TTS similar a la que se utiliza en el protocolo HTTPS. Esto no implica una seguridad absoluta e inquebrantable, pero sí que añade un nivel superior de seguridad respecto al resto de puertos.

Como se ha mencionado anteriormente, la principal desventaja que se encuentra en el protocolo SMTP es la falta de seguridad en determinados puertos utilizados, por lo que se emplea comúnmente para correos fraudulentos, que permite enviar un correo cualquiera desde cualquier dirección a cualquier destinatario.

Por otro lado, se debe mencionar la facilidad y lo básico que resulta el protocolo, a modo de ventaja. Los mensajes están basados en codificación ASCII. Las líneas se intercambian entre el cliente y el servidor, y llevan un orden estricto. En la tabla 3 se incluyen los diferentes comandos y respuestas que se plantean durante la conexión entre servidor y cliente:

Cliente/Servidor	**Mensaje**
Servidor	220 SMTP
Cliente	HELO miequipo.midominio.com
Servidor	250 Hello, please to meet you
Cliente	MAIL FROM: yo@midominio.com
Servidor	250 Ok
Cliente	RCPT TO: destinatario@sudominio.com
Servidor	250 Ok
Cliente	DATA
Servidor	354 End data with "."
Cliente	Subject: Campo de asunto
Cliente	From: yo@midominio.com
Cliente	To: destinatario@sudominio.com
Cliente	Hola,
Cliente	Esto es una prueba.
Cliente	Adiós
Cliente	.
Servidor	250 Ok: queued as 12345
Cliente	QUIT
Servidor	221 Bye

Tabla 3: Estructura de mensajes en el protocolo SMTP

Como se puede observar, los mensajes salientes del servidor se componen de un número y una cadena de texto, donde la mayor relevancia la acoge el número. En determinados casos, como en el 250, se verifica el envío del correo, siendo el cliente el que traduce dicha información. La comunicación comienza con la detección de la conexión por parte del servidor en el puerto correspondiente, devolviendo un 220. El cliente responde con un "HELO", seguido de la información propia del dispositivo con el que se trabaja. Una vez el servidor confirma que ha recibido dicha información, el cliente envía el correo emisor y el correo destinatario, con confirmaciones de recepción por parte del servidor. Posteriormente se envían todos los datos necesarios para el envío del correo, y el propio cliente da la orden de desconexión.

Para casos en los que existen errores de envío, el servidor emite unos comandos con nomenclatura 5XX, siendo X un número entero del 0 al 9, para que el cliente lo interprete.

4.4. Servidor HTTP

Una de las opciones que se implementan en el proyecto que se ocupa es el acceso a la última sesión realizada, haciendo uso de la memoria EEPROM del sistema. Para ello, se genera un servidor HTTP desde una dirección IP local donde se implementa la información requerida.

Un servidor web o servidor HTTP es una aplicación informática utilizada para la distribución de contenido web en redes internas o en internet, encargándose de proporcionar los datos para la visualización de la información. El funcionamiento reside en la renderización de un código recibido a través de un navegador web utilizando generalmente el protocolo HTTP, basado a su vez en los protocolos de red IP y TCP, para la transmisión de datos durante la comunicación, representando en el modelo OSI la capa de aplicación.

Existen dos tipos de servidor web en función a la estructura y el proceso de envío de los archivos: los estáticos y los dinámicos. En el primero dichos archivos se envían mediante el servidor de la misma forma que se almacenan, directamente hacia el navegador. Por otro lado, los dinámicos cuentan con una aplicación servidor y una base de datos, necesitando más recursos que el servidor estático debido a la complejidad del esquema establecido. La aplicación, por tanto, se encarga de actualizar los archivos almacenados previo al envío que se realiza por el servidor HTTP.

En la práctica, el servidor web se ejecuta en un dispositivo y se mantiene a la espera de recibir peticiones por parte de un cliente a través de un navegador web. Una vez las recibe son respondidas mediante una página web que se reproduce a través del navegador. El ejemplo más básico es la petición HTTP que se realiza al servidor con la URL a la que se desea acceder, mostrando mediante código HTML la página correspondiente, donde el cliente interpreta dicho código y lo muestra por pantalla [23]. Además de la transferencia de un código HTML, estos servidores son capaces de entregar aplicaciones web, las cuales se corresponden a códigos que se ejecutan bajo una petición o respuesta HTTP. Estas aplicaciones pueden ser:

- Aplicaciones en el lado del cliente. Es el cliente el que se encarga de ejecutar las aplicaciones en el dispositivo del usuario, utilizando comúnmente lenguajes como Java y Javascript.
- Aplicaciones en el lado del servidor. El servidor ejecuta las aplicaciones generando un código HTML y enviándolo al cliente mediante el protocolo HTTP. Este último tipo de aplicación es el que se gestiona en el presente proyecto.

Un servidor HTTP requiere de diferentes métodos de petición, los cuales indican la acción que se quiere realizar para un recurso determinado, denominados *HTTP verbs*. Estos se implementan mediante semánticas diferentes a pesar de contar con características similares entre varios de ellos. Los principales *HTTP verbs* son la petición GET y la petición POST.

4.4.1. Petición GET

La petición GET aparece, por ejemplo, al escribirse una dirección URL en el navegador web. Al realizarse, el navegador conecta con el servidor enviándole dicha petición con la forma "GET /index.php". El archivo "index.php" se representa con la página de inicio del sitio web, lo que el servidor envía como respuesta al navegador. De la misma forma se ejecutan los archivos HTML, emitiendo de la misma forma la petición ("GET /test.html"). Estas peticiones pueden incluir un mayor número de requisitos interpretables por el servidor. Si el navegador plantea una petición "GET /search?platform=Windows&category=office", el servidor interpreta tres secuencias. La primera será la interrogación (?), que indica el inicio de los parámetros de la petición, donde posteriormente se solicita que el parámetro "platform" tenga un valor igual a "Windows" y (&) "category" sea igual a "office".

A nivel de infrastructura se le asigna, por norma general, el puerto TCP 80 al protocolo HTTP. El método GET solicita una representación de un recurso específico, por lo que sólo se recuperan datos. Cuando un navegador realiza una o varias peticiones, se produce un *socket* con un servidor existente en una dirección IP mediante TCP. Las direcciones DNS, las cuales son alfanuméricas, se convierten en numéricas, y se solicita al servidor las direcciones IP correspondientes, en caso de que no exista ya una regla de base de datos DNS. De esta manera, se crea una nueva regla y se almacena la IP, junto a su relación con la DNS correspondiente. Tras esto, se realiza una petición a la base de datos para recuperar los valores de la regla y posteriormente se produce un *socket* con la dirección IP mediante TCP. Es entonces cuando se crea la petición GET con la URL, un *flag*, una prioridad y un método, el cual es el propio GET. Se realiza la apertura de caché y se ejecuta la petición, donde se leen las cabeceras HTTP y el cuerpo, denominado *HTTP transaction*. Finalmente se ejecuta una consulta en el caché sobre la existencia de una entrada asociada al recurso solicitado, lo cual tendrá una respuesta booleana donde la variable, denominada "created", tomará valor *true* si existiera el recurso, y *false* en caso contrario. En caso positivo se muestra la información procesada y se comprueba que sea correcta.

4.4.2. Petición POST

Para el envío al servidor web de grandes paquetes de datos como imágenes o formularios se necesita de un método más potente respecto al GET, ya que no se pueden escribir todos estos datos en la barra de direcciones de un navegador. Es entonces cuando se aplica la petición POST, adjuntándose a la cabecera HTTP. Un ejemplo se puede ver en la figura 10:

```
<html>
<body>
<form action="newsletter.php" method="post">
Name: <input type="text" name="name"><br>
E-mail: <input type="text" name="email"><br>
<input type="submit">
</form>
</body>
</html>
```

Figura 10: Ejemplo de una cabecera HTTP. Fuente: Ionos

En el caso de la petición POST, se requiere el envío de una entidad a un recurso específico, causando frecuentemente una modificación en el estado del servidor, lo que incluye posibles efectos secundarios. Para el funcionamiento de este, los datos que se deben enviar al servidor se adjuntan al cuerpo de la petición con las cabeceras HTTP asignadas, que se corresponden al tipo de petición. En la petición se incluye la cabecera "application/x-www-form-urlencoded", lo que indica el formato o el tipo de codificación que se envía, separando las variables mediante el símbolo &. La estructura generalizada para la petición POST es la siguiente:

- *Petition Type*. Determina el tipo de petición HTTP.
- *Referer*. Especifica la URL de origen.
- *Content-Length*. Especifica la longitud en bytes de los datos enviados en el cuerpo de la petición.
- *Origin*. Determina la URL principal.
- *User-Agent*. Referencia al identificador del navegador web que realizó la petición.
- *Content-Type*. Marca el formato o MIME (*Multipurpose Internet Mail Extensions* de los datos enviados en el cuerpo de la petición.

- *Accept*. Especifica el MIME que se espera obtener en la respuesta.
- *Accept-Lenguage*. Indica el lenguaje que debe tener el código de la respuesta.
- *Accept-Charset*. Especifica la codificación esperada en la respuesta.
- *Cookie*. Determina el identificador la sesión en la petició, que deriva en una *cookie*.
- *Accept-encoding*. Determina el tipo de codificación o compresión esperada en la respuesta (opcional).

La respuesta, por su parte, tendrá una estructura con los siguientes parámetros.

- *HTTP version & date*.
- *Date*.
- *Server*.
- *Expires*.
- *Cache-control*.
- *Pragma*.
- *Content-length*.
- *Content-type*.
- *Keep-Alive*.
- *Connection*.
- *X-Powered by*.

4.4.3. Otras peticiones de un servidor web

Además de las ya mencionadas peticiones GET y POST, existen otras adicionales que se utilizan a lo largo del funcionamiento de un servidor HTTP, que solicitan acciones específicas para un recurso. Estas son:

- HEAD.
- PUT.
- DELETE.
- CONNECT.
- OPTIONS.
- TRACE.
- PATCH.

Desarrollo del código

5. Desarrollo del código

5.1. Funcionamiento práctico

A la hora de llevar a la práctica el sistema diseñado, el usuario accede a la aplicación *Android/iOS* diseñada específicamente para el proyecto, tal y como se puede ver en la figura 13. Además, el propio dispositivo muestra por pantalla la instrucción correspondiente (figura 12), previa aparición del logo que acompaña al sistema (figura 11).

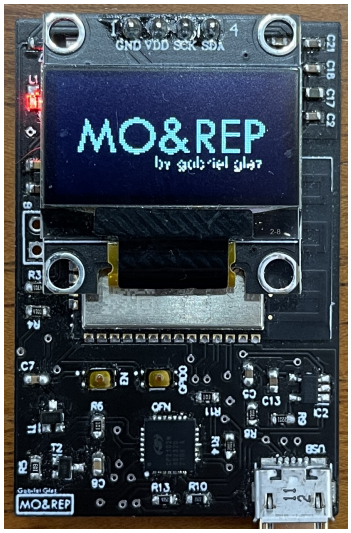


Figura 11: Pantalla de carga del subsistema principal

Figura 12: Pantalla de espera del nodo de la muñeca

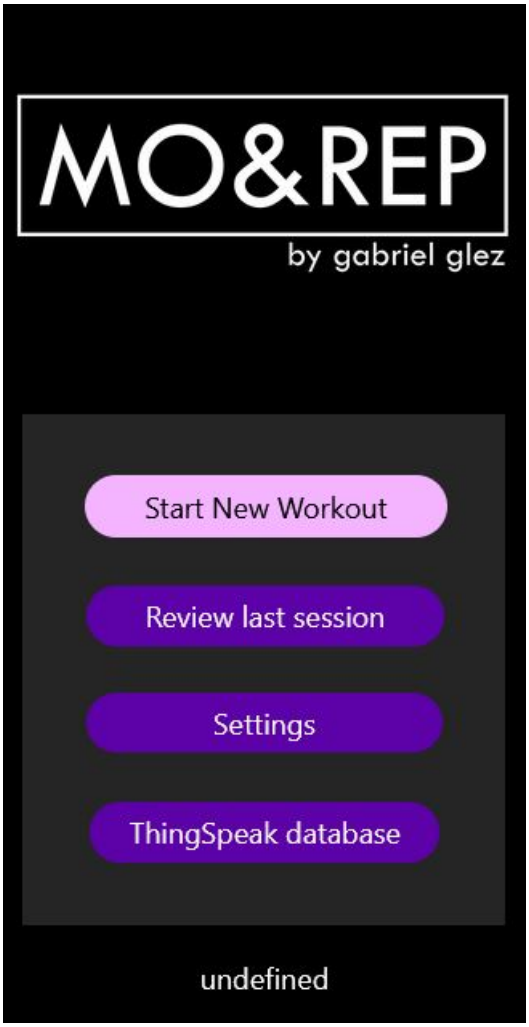


Figura 13: Pantalla principal de la *app*

En la pantalla principal aparecen varias opciones:

- La primera opción ordena el inicio de un nuevo *workout*. Cuando se selecciona dicho botón, el sistema inicia la tarea que se corresponde con el acelerómetro y giroscopio, detectando la posición del brazo y del antebrazo y codificando la sincronización de ambos.
- La segunda alternativa es la revisión del entrenamiento realizado durante la sesión anterior mediante la creación de un servidor HTTP, haciendo uso de la memoria EEPROM integrada en el ESP32.
- La tercera opción permite al usuario configurar parámetros como el envío del correo electrónico al finalizar la rutina.

- El último botón abre una ventana con la que acceder directamente al portal de *ThingSpeak*, lo que permite la visualización de la base de datos en la propia aplicación.

En este punto del proceso el emparejamiento entre ambos subsistemas ya se ha realizado. Por tanto, si el usuario decide seleccionar la primera opción mencionada anteriormente, se sincronizan ambos y comienza el proceso de detección del ejercicio. Durante la sesión de entrenamiento se puede ir observando en la pantalla del nodo inferior las series analizadas hasta el momento, incluyendo las repeticiones realizadas, así como el tiempo transcurrido, tal y como se muestra en las figuras 14 y 15.

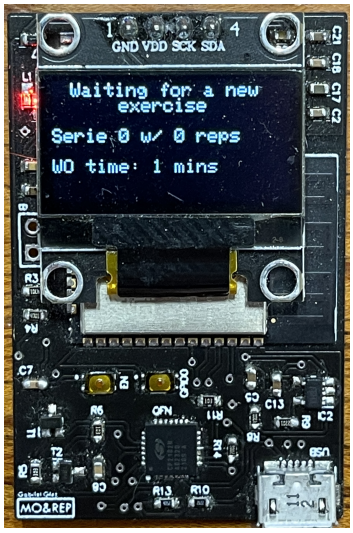


Figura 14: Inicio de la detección de movimientos

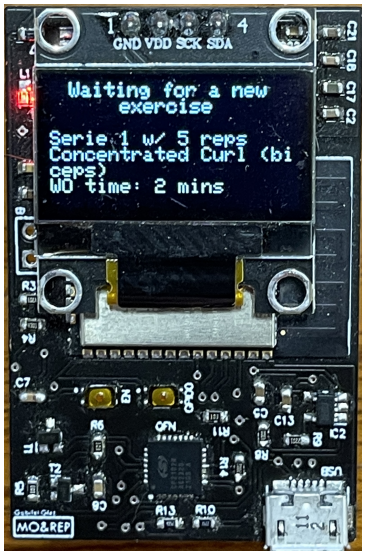


Figura 15: Serie detectada durante la sesión

Durante el transcurso del ejercicio, se podrá leer en la pantalla del nodo inferior la detección que está realizando el sistema en función a los movimientos del usuario, representado en la figura 16.

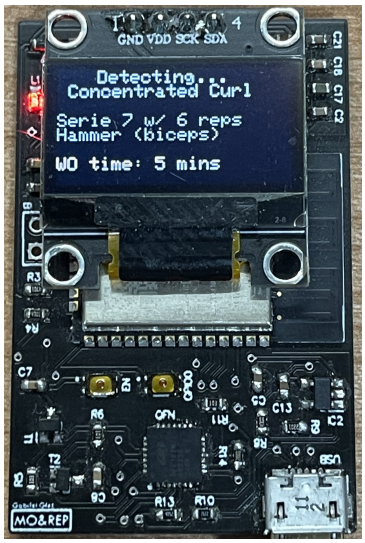


Figura 16: Pantalla de detección de ejercicio

Una vez finalizado el *workout*, se puede observar en la pantalla del sistema las instrucciones de finalización, que se resumen en el acceso al correo electrónico para el análisis del entrenamiento realizado y el acceso a la misma información a través de la aplicación, que tendrá la estructura visible en la figura 17, coincidiendo así con la revisión mencionada en el segundo punto.

Figura 17: Pantalla de finalización de entrenamiento

En la figura 18 se pueden observar los parámetros configurables en el sistema, entre los que se encuentran el envío del correo electrónico bajo el protocolo SMTP al finalizar la sesión, la activación del dispositivo tras el modo *sleep* y la opción de detectar ejercicios del tronco inferior.

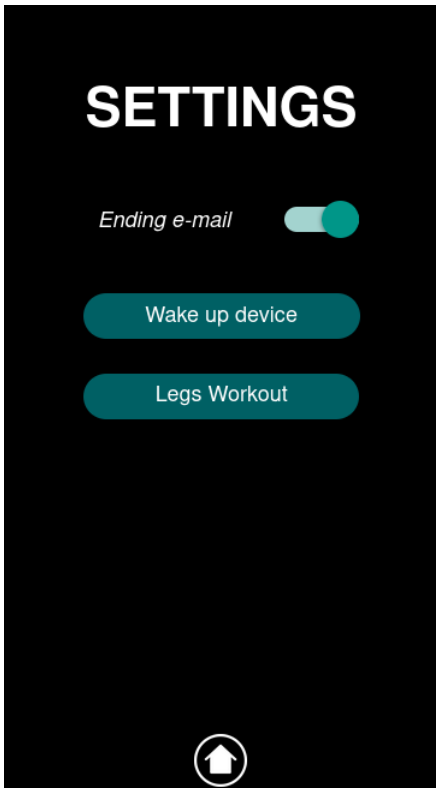


Figura 18: Pantalla de configuración de la aplicación

Además, es posible acceder en tiempo real a la base de datos de *ThingSpeak* para visualizar la evolución del número de series realizadas para cada grupo muscular, junto a la fecha y hora a la que se registraron, tal y como se puede observar en la figura 19.

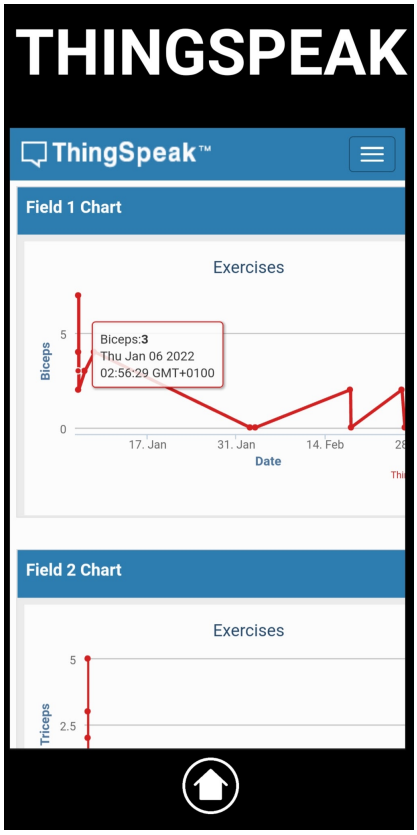


Figura 19: Pantalla de acceso a la base de datos de ThingSpeak

En las figura 20 y 21 se representan las estructuras correspondientes para el análisis de la sesión previa almacenada en la EEPROM del ESP32 y del correo electrónico enviado al finalizar el entrenamiento.

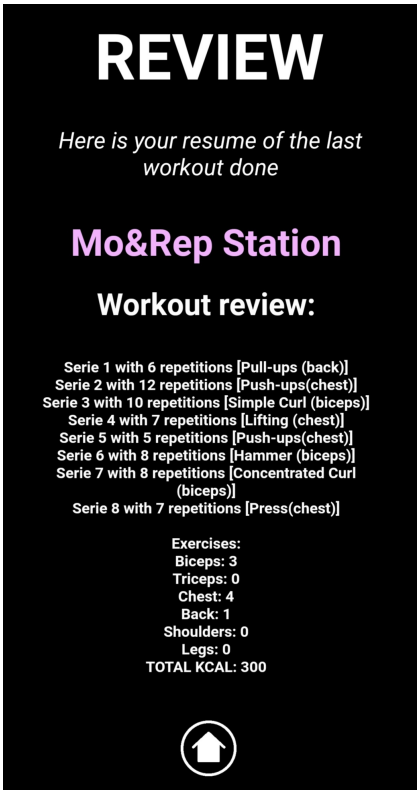


Figura 20: Análisis del entrenamiento previo visto desde la *app*

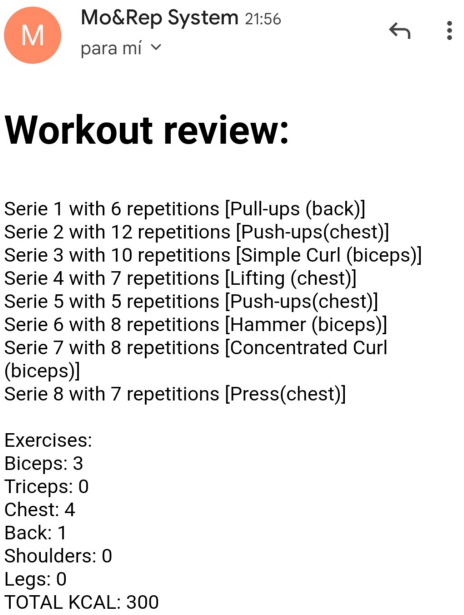


Figura 21: Formato de correo electrónico enviado al finalizar la sesión

5.2. Código embebido para el nodo situado en la muñeca

En primer lugar se desarrollará el entramado del algoritmo correspondiente al subsistema de la muñeca, debido a la cantidad de ejercicios detectables en este nodo, junto con la responsabilidad adicional de ejercer como servidor *Bluetooth* del sistema en su conjunto, ocupando tareas esenciales como el envío del correo electrónico y el registro de datos.

Tal y como se mencionó al principio de la memoria, los códigos se desarrollan en el IDE de Arduino, lo que implica el uso de bibliotecas propias para este software. En el caso que se ocupa se recogen en la tabla 4:

Librería	**Descripción**
Arduino.h	Libería básica del software implementado
Wire.h	Comunicación con dispositivos que utilizan el puerto serie (I^2C)
Adafruit_GFX.h	Uso de la pantalla conectada en el puerto serie
Adafruit_SSD1306.h	Definición de las funciones que se aplican en la pantalla
I2Cdev.h	Aplicación de funciones para dispositivos conectados en el puerto serie y el ESP32
MPU6050.h	Funciones del circuito tipo *smart-sensor* que cuenta con un acelerómetro y un giroscopio
String.h	Uso de cadenas de texto
WiFi.h	Uso del módulo WiFi + Bluetooth ESP32
EEPROM.h	Gestión de la EEPROM incorporada en el ESP32
Separador.h	Libería diseñada por el portal "Ioticos" para el tratamiento de cadenas de caracteres con separadores
ESP_Mail_Client.h	Comandos requeridos para el envío del correo electrónico mediante protocolo SMTP
ThingSpeak.h	Funciones de acceso a la base de datos
Secrets.h	Almacenamiento de claves de acceso y datos esenciales
MAX30105.h	Aplicación del algoritmo para el sensor de frecuencia cardíaca
Spo2_algorithm.h	Uso del medidor de oxígeno en sangre que incorpora el MAX30102EFD+
HeartRate.h	Complemento para las medidas biomédicas
NimBLEDevice.h	Comunicación BLE utilizando un 50 % de recursos del ESP32 respecto a la librería original
TaskScheduler.h	Gestión de tareas realizadas en paralelo

Tabla 4: Bibliotecas utilizadas en el sensor de la muñeca

Una vez declaradas las bibliotecas, se procede a la definición de variables que no hacen referencia a información confidencial de cara al acceso de la comunicación, tales como variables del sistema de detección de ejercicios y medición de datos biomédicos o punteros a características BLE. Tras esto, se detallan una serie de funciones propias del *Bluetooth* de baja energía, así como la codificación en lenguaje hexadecimal del logo del sistema.

Es entonces cuando se declara la tarea principal: la detección de ejercicios en el entorno de *fitness*. Utilizando el giroscopio y acelerómetro (MPU-6050), se realiza la función primordial del sistema de forma cíclica cada 200 ms, suficiente para el registro de cambios de posición.

El subsistema de la muñeca, que actúa como *master*, recibe la orden del inicio del entrenamiento, que se ve reflejada en la variable "workoutctrl". Cuando esta toma el valor entero "1" se activa la sincronización de ambos subsistemas. Para ello, el nodo principal modifica la característica diseñada en la comunicación BLE ("Wrist"). Esta característica siempre tendrá un encabezado con la terminología "WM_", independientemente del momento en el que se sitúe la sesión de entrenamiento, por lo que, para el inicio de este, está formada por una cadena de texto con valor "WM_1". Es aquí cuando "workoutctrl" se reinicia para evitar un *bug* crítico donde la característica de la muñeca siempre tiene valor 1. Será en este punto donde el nodo del brazo, que actúa como *slave*, lee el valor obtenido y escribe en su propia característica ("Arm") el mismo dato con el encabezado "AM_". Debido a que la función lectura en ambos nodos se produce cada 333 ms, el *delay* obtenido en el proceso es muy pequeño y no afecta al correcto funcionamiento del sistema.

Una vez programado el proceso de sincronización entre ambos nodos, se detallan las particularidades del subsistema de la muñeca. Para definir qué ejercicios se está realizando se aplica un mecanismo de diferenciación de posiciones basados en un dado, donde cada lado de este corresponde a una posición del antebrazo. En la figura 22 se muestra la estructura que se menciona. A su vez, la tabla 5 muestra las posiciones a las que se hace referencia, junto con las aceleraciones que se deben recoger en el eje correspondiente.

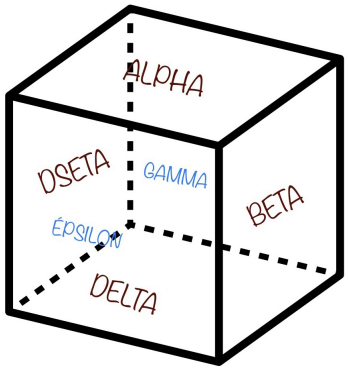


Figura 22: Representación de los lados escogidos para el acelerómetro en el nodo principal

Posición	**Aceleración**
Alpha	Eje Z >7 m/s^2
Delta	Eje Z <-7 m/s^2
Beta	Eje Y >7 m/s^2
Dseta	Eje Y <-7 m/s^2
Épsilon	Eje X >7 m/s^2
Gamma	Eje X <-7 m/s^2

Tabla 5: Relación entre las posiciones del subsistema de la muñeca y sus aceleraciones

Para el resto de valores comprendidos entre las aceleraciones destacadas se establece una posición trivial bajo el nombre "Half way", que no produce ninguna modificación en el mecanismo de detección.

El algoritmo parte de dichas posiciones, las cuales se integran en una máquina de ocho estados, donde seis equivalen a "Alpha", "Delta", "Beta", "Dseta", "Épsilon" y "Gamma", quedando dos adicionales que serán un estado inicial, necesario para reinicializar determinadas variables, y un estado final donde se determina las repeticiones que se han realizado para una serie concreta, almacenando en el registro el ejercicio realizado para comenzar con el proceso de nuevo.

En el presente proyecto se diferencian dos tipos de registro: el que se realiza mediante el subsistema de la muñeca y el que se realiza con el nodo del brazo. Esto se debe a las variaciones nulas en algunos movimientos para un determinado nodo. Un ejemplo de ejercicios que se detectan desde la muñeca es un *curl* concentrado de bíceps. Tal y como se ve en la figura 23, la parte superior del brazo se mantiene estática durante todo el movimiento correspondiente al ejercicio, mientras que es el antebrazo el que se desplaza. Para este caso, el subsistema de la muñeca parte desde una posición "Épsilon" y finaliza en una posición "Alpha", mientras que el nodo superior se mantiene estable.

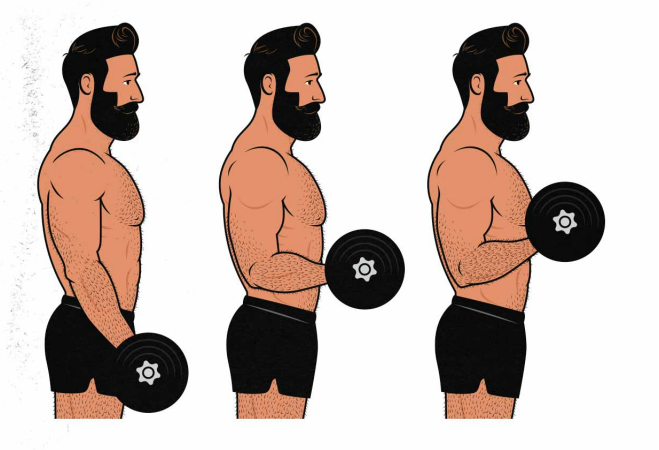


Figura 23: Procedimiento para la realización del *curl* de bíceps. Fuente: *Outlift*

Un total de 17 ejercicios recogidos se registran mediante el subsistema que actúa como *master*, aunque hay otros 13 para los que se requiere del *slave*, como se menciona posteriormente en el punto 5.3. En la tabla 6 se recoge la relación entre las posiciones del nodo inferior, el estado al que pertenecen y la cantidad de ejercicios en los que intervienen para los cinco grupos musculares de la parte superior del cuerpo que se engloban en el presente proyecto: pecho, espalda, bíceps, tríceps y hombros.

Posición	**Estado**	**Pecho**	**Espalda**	**Bíceps**	**Tríceps**	**Hombros**
Alpha	2	1	0	2	2	0
Delta	6	1	2	1	3	2
Beta	5	1	1	1	1	1
Dseta	7	0	0	1	0	0
Épsilon	4	2	1	3	7	3
Gamma	3	2	3	4	5	2

Tabla 6: Estados equivalentes a las posiciones del antebrazo y número de ejercicios para cada grupo muscular

Cabe recordar que las características implementadas en el protocolo *Bluetooth* de baja energía se van modificando a medida que se realizan los movimientos, a modo de mensajes que se codifican entre ambos subsistemas. Durante la ejecución del ejercicio, la característica del nodo inferior detalla el inicio y el final del movimiento de este. En la tabla 7 se determinan los códigos utilizados para la transmisión de información en el caso del subsistema de la muñeca.

Posición inicial	Estado inicial	Posición final	Estado final	Aceleración (m/s^2)	Codificación
Alpha	2	Gamma	3	-	WM_23
Épsilon	4	Épsilon	4	$\triangle ay > 2$	WM_04A
Épsilon	4	Épsilon	4	$\triangle ay \approx 0$	WM_04Q
Gamma	3	Gamma	3	$\triangle ax \approx 0$	WM_03Q
Gamma	3	Gamma	3	$\triangle ax > 2$	WM_03A
Beta	5	Beta	5	-	WM_05
Delta	6	Delta	6	-	WM_06

Tabla 7: Codificación de mensajes del nodo de la muñeca

Una vez finalizado el entrenamiento, se desmarca el botón correspondiente, ordenando así que la sesión ha terminado. Esto se identifica en el algoritmo con la variable mencionada anteriormente "workoutctrl", que toma el valor entero "2", lo que provoca la suspensión de la tarea de detección de posición. La característica perteneciente al *master* toma la codificación "WM_96", que interpretará el nodo superior para detener su proceso equivalente. Es aquí donde se ordena el envío del correo electrónico utilizando el protocolo SMTP, la carga del registro a la base de datos *ThingSpeak* y el guardado automático en la memoria EEPROM del sistema. En la pantalla del nodo inferior se mostrará el mensaje de finalización mostrado anteriormente, con las instrucciones correspondientes.

Más allá de esto se encuentran las funciones *setup()*, donde se inician diferentes componentes como la conexión *Bluetooth*, el puerto serie con la pantalla y el giroscopio más acelerómetro, o la lectura de la EEPROM. La función *FinishWorkout()* corresponde al envío del correo electrónico, al registro y almacenamiento de la información y lo relacionado con la base de datos, mientras que *ReadingEEPROM()* realiza la lectura de la sesión anterior, junto a la creación del servidor HTTP mediante la función *webServer()*.

5.3. Código embebido para el nodo situado en el brazo

El esquema que se sigue para el nodo superior es similar, aunque añade simplicidad respecto al nodo de la muñeca, ya que se rescinde de parámetros como la creación del servidor BLE o el envío del correo electrónico bajo el protocolo SMTP. Además, el número de ejercicios registrados desde este punto es de 13, lo que conforma una reducción significativa en términos de memoria *flash* y operaciones en tiempo real.

De la misma forma que en el apartado anterior, se va a diferenciar dos tipos de tareas que se realizan cíclicamente durante el transcurso del entrenamiento: la modificación de la característica correspondiente a la comunicación *Bluetooth* del subsistema ("Arm"), y la detección de la posición y el movimiento de este.

Como se ha mencionado anteriormente, el encabezado del mensaje que se utiliza en este nodo es "AM_" seguido de un código numérico que indica el recorrido que realiza durante la serie. Sin embargo, se requiere de un segundo encabezado para los ejercicios detectados desde este punto, ya que se debe enviar la información al subsistema *master* para el registro y la agrupación de los datos obtenidos. Este último encabezado será "EX_" seguido del codigo correspondiente del ejercicio realizado y el número de repeticiones realizadas para dicha serie.

Partiendo de esto, se definen seis posiciones posibles diferenciables en el nodo superior, las mismas que para la muñeca, con la diferencia de la posición "Eta", que no se aplica en ninguno de los ejercicios programados, por lo que su existencia es meramente simbólica y sujeta a la adición de más movimientos. Las diferentes caras del dado se pueden visualizar en la tabla 8 que se adjunta a continuación:

Posición	**Aceleración**
Theta	Eje Z <-7 m/s^2
Iota	Eje Z >7 m/s^2
Kappa	Eje X <-7 m/s^2
Eta	Eje X >7 m/s^2
Lambda	Eje Y <-7 m/s^2
Mi	Eje Y >7 m/s^2

Tabla 8: Relación entre las posiciones del subsistema del brazo y sus aceleraciones

De la misma manera que se realiza para el nodo inferior, se incorpora una posición "Half Way" que no aporta nada en el registro, y únicamente se utiliza como puente hacia otra posición.

En este nodo se detecta un total de 13 ejercicios en los que principalmente se localiza una variación en la parte superior del brazo, junto a un período de inmovilidad por parte del antebrazo. Un ejemplo de esto es el conocido *press* de hombros que se puede visualizar en la figura 24.

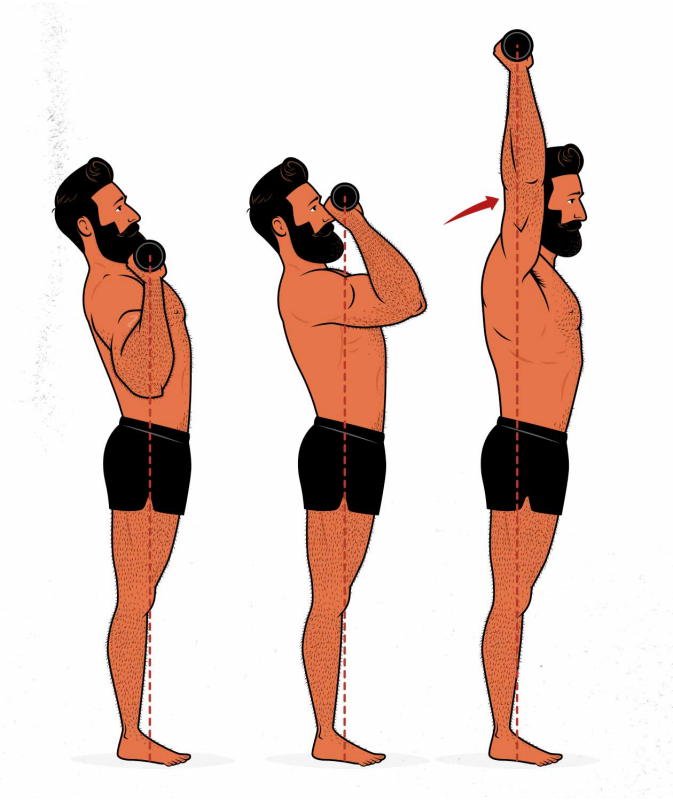


Figura 24: Procedimiento para la realización del *press* de hombros. Fuente: *Outlift*

Conociendo el mecanismo que se ha empleado para ambos subsistemas, se puede observar que la orientación del antebrazo no varía durante el desarrollo completo del ejercicio, mientras que es la parte superior la que sufre un cambio respecto a su posición inicial. Por tanto, el nodo inferior se mantiene en posición "Gamma" durante todo el transcurso de la serie, enviando un código "WM_03A" a la característica del antebrazo ("Wrist"), mientras que el nodo superior partirá desde una posición "Mi" y finalizará en "Kappa". El sistema está diseñado para que no existan conflictos entre ejercicios, por lo que el subsistema del húmero detecta el ejercicio que se está realizando y, una vez finalizado, codifica su característica ("Arm") con el valor "EX_24_X", donde 24 es la posición para *press* de hombros del *array* que incorpora todos los ejercicios programados y X corresponde al número de repeticiones realizadas, añadiendo una serie más en el registro. Esta información es leída por el *master*, que lo añade al resumen e inicia una nueva sincronización.

En la tabla 9 que se adjunta se relaciona, de igual manera que en el apartado anterior, cada posición implementada con el estado al que pertenece en la máquina de estados y el número de ejercicios en los que interviene para cada grupo muscular.

Posición	**Estado**	**Pecho**	**Espalda**	**Bíceps**	**Tríceps**	**Hombros**
Theta	3	2	1	0	0	1
Iota	5	2	2	3	7	4
Kappa	2	3	4	0	0	4
Lambda	6	1	2	0	2	0
Mi	4	4	4	1	6	2

Tabla 9: Estados equivalentes a las posiciones del húmero y número de ejercicios para cada grupo muscular

Cabe destacar que la máquina de estados que se implementa en este nodo cuenta con un total de siete estados, uno menos que el nodo inferior, debido a la posición que no tiene relevancia en el proceso.

Se incorpora en la tabla 10, además, la lista de códigos que pueden aparecer en la característica *Bluetooth* del húmero debidos al comportamiento durante el movimiento general.

Posición inicial	**Estado inicial**	**Posición final**	**Estado final**	**Aceleración (m/s^2)**	**Codificación**
Kappa	2	Kappa	2	-	AM_02
Iota	5	Kappa	2	-	AM_52
Iota	5	Theta	3	-	AM_53
Iota	5	Mi	4	-	AM_54
Iota	5	Lambda	6	-	AM_56
Iota	5	Iota	5	-	AM_05
Mi	4	Kappa	2	-	AM_42
Mi	4	Iota	5	-	AM_45
Mi	4	Mi	4	$\triangle az \approx 0$	AM_04Q
Mi	4	Mi	4	$\triangle az > 2{,}5$	AM_04A
Theta	3	Iota	5	-	AM_35
Theta	3	Theta	3	-	AM_03

Tabla 10: Codificación de mensajes del nodo del húmero

Además, se implementan dos posibles opciones de cara a la finalización de la detección del ejercicio y el inicio de uno nuevo:

- Se crea un variable denominada "a_total" que equivale a la suma de los valores absolutos de las aceleraciones en cada eje al cuadrado. En el análisis experimental se verificó que un ejercicio tiene un margen de aceleraciones de 7 a 20 m/s^2 para esta variable. Para darle un coeficiente de seguridad fiable, se establece un valor de 40 m/s^2, identificado como una agitación en el nodo de la muñeca, a modo de reinicio manual. Por tanto, si el subsistema detecta que el valor que toma "a_total" es mayor a 40 m/s^2, el algoritmo se trasladará directamente al estado 8, donde se registra lo realizado y se reinicia la detección.

- Por otro lado, un momento donde ambos nodos se mantienen estáticos durante más de 12 segundos, sin cambiar de posición en ningún instante. Esto se produce comúnmente en etapas de descanso entre ejercicios, lo que permite un reinicio automático de la detección.

La tabla 11 muestra una serie de códigos genéricos para la comunicación entre ambos nodos, relacionados con lo visto anteriormente, mientras que las tablas 12 y 13 profundizan en detalle las condiciones y los mensajes que se transmiten para cada uno de los ejercicios propuestos.

Orden	**Emisor**	**Receptor**	**Codificación**
Subsistema preparado	Ambos	Ambos	WM_1 AM_1
Fin de la sesión	Antebrazo	Húmero	WM_96
Adición de una serie	Antebrazo	Húmero	WM_97
Inicio manual	Antebrazo	Húmero	WM_98
Tiempo >12 segs	Ambos	Ambos	WM_99 AM_99

Tabla 11: Códigos de comunicación entre nodos

Ejercicio	Número	Grupo muscular	Nodo inferior	Nodo superior	Condición N.I.	Condición N.S.	Código N.I.	Código N.S.
Curl simple	Exercise[1]	Bíceps	Alpha-Gamma	Mi-Mi	Estado inicial: 2 Cambio de estado (2-3)	Estado fijo: 4	Ninguno	AM_04Q
Hammer	Exercise[2]	Bíceps	Épsilon-Beta	Iota-Iota	Estado inicial: 4 Cambio de estado (4-5)	Estado fijo: 5	Ninguno	AM_05
Curl concentrado	Exercise[3]	Bíceps	Épsilon-Alpha	Iota-Iota	Estado inicial: 4 Cambio de estado (4-2)	Estado fijo: 5	Ninguno	AM_05
Curl inverso	Exercise[4]	Bíceps	Épsilon-Delta	Iota-Iota	Estado inicial: 4 Cambio de estado (4-6)	Estado fijo: 5	Ninguno	AM_05
Extensión c/cuerda	Exercise[5]	Tríceps	Gamma-Épsilon	Mi-Iota	Estado inicial: 3 Cambio de estado (3-4)	Cambio de estado (4-5)	Ninguno	AM_45
Extensión c/barra	Exercise[6]	Tríceps	Gamma-Épsilon	Iota-Iota	Estado inicial: 3 Cambio de estado (3-4) Estado intermedio: 6	Cambio de estado (4-5)	Ninguno	AM_05
Extensión inversa	Exercise[7]	Tríceps	Gamma-Épsilon	Iota-Iota	Estado inicial: 3 Cambio de estado (3-4) Estado intermedio: 2	Cambio de estado (4-5)	Ninguno	AM_05
Fondos	Exercise[10]	Tríceps	Épsilon-Épsilon	Lambda-Iota	Estado fijo: 4 Tríceps >Pecho $\triangle$ay $\approx$ 0 m/s^2	Cambio de estado (6-5) Estado inicial: 6	WM_04Q	Ninguno
Flexiones diamante	Exercise[30]	Tríceps	Épsilon-Delta	Iota-Lambda	Estado inicial: 4 Cambio de estado (4-6)	Cambio de estado (5-6)	Ninguno	AM_56
Press	Exercise[24]	Hombros	Gamma-Gamma	Mi-Kappa	Estado fijo: 3	Cambio de estado (4-2) Pecho >Hombros Estado inicial: 4	WM_03A	Ninguno
Apertura (acostado)	Exercise[12]	Pecho	Alpha-Gamma	Mi-Kappa	Estado inicial: 2 Cambio de estado (2-3)	Cambio de estado (4-2)	Ninguno	AM_42
Apertura(erguido)	Exercise[13]	Pecho	Beta-Beta	Theta-Theta	Estado fijo: 5 Estado inicial: 5 $\triangle$az >2.5 m/s^2	Estado fijo: 3	Ninguno	AM_03
Empuje	Exercise[14]	Pecho	Delta-Delta	Theta-Theta	Estado fijo: 6 Estado inicial: 6 $\triangle$ay >2.5 m/s^2	Estado fijo: 3	Ninguno	AM_03
Flexiones	Exercise[15]	Pecho	Épsilon-Épsilon	Iota-Lambda	Estado fijo: 4 Tríceps <Pecho $\triangle$ay $\approx$ 0 m/s^2	Cambio de estado (5-6) Estado inicial: 5	WM_04Q	Ninguno
Levantamiento	Exercise[16]	Pecho	Épsilon-Alpha	Iota-Mi	Estado inicial: 4 Cambio de estado (4-2)	Cambio de estado (5-4)	Ninguno	AM_54 | AM_45
Dominadas	Exercise[18]	Espalda	Gamma-Gamma	Kappa-Mi	Estado fijo: 3 $\triangle$ay $\approx$ 0 m/s^2	Cambio de estado (2-4) Estado inicial: 2	WM_03Q	Ninguno
Jalón al pecho (abierto)	Exercise[19]	Espalda	Gamma-Gamma	Kappa-Mi	Estado fijo: 3 $\triangle$ay <-2.5 m/s^2	Cambio de estado (2-4) Estado inicial: 2	WM_03A	Ninguno
Press hombros	Exercise[24]	Hombros	Gamma-Gamma	Mi-Kappa	Estado fijo: 3	Cambio de estado (4-2) Hombros >Pecho Estado inicial: 4	WM_03A	Ninguno

Tabla 12: Condiciones de ambos nodos para la detección de los ejercicios (I)

Ejercicio	Número	Grupo muscular	Nodo inferior	Nodo superior	Condición N.I.	Condición N.S.	Código N.I.	Código N.S.
Apertura	Exercise[25]	Hombros	Delta-Delta	Theta-Theta	Estado inicial: 3 Estado fijo: 3 $\triangle$az <-2.5 m/s^2	Estado fijo: 3	Ninguno	AM_03
Elevación lateral	Exercise[27]	Hombros	Épsilon-Beta	Iota-Mi	Cambio de estado (4-5)	Cambio de estado (5-4)	Ninguno	AM_54
Vuelo	Exercise[28]	Hombros	Épsilon-Delta	Iota-Kappa	Estado inicial: 4 Cambio de estado (4-6)	Cambio de estado (5-2)	Ninguno	AM_52
Jalón al pecho (cerrado)	Exercise[20]	Espalda	Gamma-Gamma	Kappa-Iota	Estado fijo: 3 $\triangle$ay <-2.5 m/s^2	Cambio de estado (2-5) Estado inicial: 2	WM_03A	Ninguno
Atracción c/cuerda	Exercise[21]	Hombros	Delta-Delta	Theta-Mi	Estado fijo: 6	Cambio de estado (3-4) Estado inicial: 3	WM_06	Ninguno
Remo c/polea (V)	Exercise[22]	Espalda	Beta-Beta	Mi-Mi	Estado inicial: 5 Estado fijo: 5 $\triangle$ay <-2.5 m/s^2	Estado fijo: 4	Ninguno	AM_04Q
Remo c/polea (H)	Exercise[23]	Espalda	Delta-Delta	Theta-Theta	Estado inicial: 6 Estado fijo: 6 $\triangle$ay >2.5 m/s^2	Estado fijo: 2	Ninguno	AM_03
Remo al cuello	Exercise[17]	Hombros	Épsilon-Delta	Iota-Theta	Estado inicial: 4 Cambio de estado (4-6)	Cambio de estado (5-3)	Ninguno	AM_53
Pull-over	Exercise[32]	Espalda	Delta-Épsilon	Theta-Iota	Cambio de estado (6-4) Estado inicial: 6	Cambio de estado (3-5) Espalda >Pecho	Ninguno	AM_35
Apertura inferior	Exercise[33]	Pecho	Delta-Épsilon	Theta-Iota	Cambio de estado (6-4) Estado inicial: 6	Cambio de estado (3-5) Pecho >Espalda	Ninguno	AM_35
Remo horizontal	Exercise[31]	Espalda	Épsilon-Épsilon	Lambda-Iota	Estado fijo: 4	Estado inicial: 6 $\triangle$ay>2.5 m/s^2	WM_04A	Ninguno
Extensión vertical	Exercise[29]	Tríceps	Dseta-Gamma	Theta-Theta	Estado inicial: 7 Cambio de estado (7-3)	Ninguno	Ninguno	Ninguno

Tabla 13: Condiciones de ambos nodos para la detección de los ejercicios (II)

Funcionalidades adicionales

6. Funcionalidades adicionales

En el presente capítulo se detallan las funcionalidades que se añaden de forma adicional al cómputo general del sistema, orientadas a una mayor integración de cara al usuario y al registro de componentes que complementan al análisis de una sesión de gimnasio.

6.1. Biblioteca NimBLE para Arduino

El módulo ESP32 que se integra en el dispositivo cuenta con unas bibliotecas oficiales para el IDE de Arduino. Estas permiten la programación de los módulos para el uso de comunicación *Bluetooth* y, más específicamente, *Bluetooth* de baja energía, lo que facilita la interacción con el microcontrolador desde un inicio, ya que se adjuntan códigos de ejemplo de fácil sustitución, explicaciones de la sistemática a seguir por parte del fabricante y tutoriales oficiales para el buen uso del algoritmo.

Sin embargo, se ha demostrado a lo largo del tiempo que estas bibliotecas ocupan un porcentaje mayoritario de la memoria *flash* con las que cuentan los ESP32, alcanzando cantidades del 80 % para códigos simples y ejemplos clásicos. Esto produce un ínfimo margen de maniobra de cara a desarrollar otras tareas que se ejecuten en paralelo con la comunicación entre módulos, como es el proyecto que se presenta. Es por ello que se decide aplicar la librería NimBLE diseñada por el autor H2Zero [21].

La biblioteca NimBLE se caracteriza por la reducción en un 50 % de espacio para la memoria *flash* para la misma funcionalidad, y el uso de 100 kb menos de memoria RAM. Además, es 100 % compatible con todos los controladores para los que se puede desarrollar la biblioteca original de Arduino. En la tabla 14 se puede observar una comparación de la ocupación de memoria para ambas bibliotecas, demostrando la reducción a la mitad de los recursos requeridos:

Ejemplo	**Librería Arduino BLE**	**Librería NimBLE-Arduino**
BLE_client	58 % (*Heap* libre = 171548)	29 % (*Heap* libre = 270336)
BLE_notify	57 % (*Heap* libre = 173300)	28 % (*Heap* libre = 269792)

Tabla 14: Comparación de ocupación del almacenamiento del programa para las librerías Arduino BLE y NimBLE-Arduino

En el presente proyecto se utiliza un único servicio denominado "ActionsService", que contiene tres características:

- La primera muestra la posición del nodo inferior ("Wrist"), con UUID "d6597e78-c992-4f43-9861-719b78932fa6", teniendo propiedades de lectura encriptada, escritura y notificación.
- La segunda detalla el movimiento del nodo del húmero ("Arm"), con UUID "84d8a625-08f9-49b3-82fb-d85daffc3c39" y las mismas propiedades que la primera.
- La tercera y última es una característica de control de la aplicación ("InfoBLE"), lo que permitirá enlazar los comandos de la *app* con sus variables en el código del sistema. Tendrá UUID "beb5483e-36e1-4688-b7f5-ea07361b26a8" y propiedades básicas de lectura, escritura y notificación.

Tal y como se menciona en el apartado 5 del presente proyecto, las características de la muñeca y el nodo superior envían constantemente mensajes con el encabezado "WM_", "AM_" y "EX_", siendo el primer código para el subsistema principal, mientras que los otros dos pertenecen al *slave*. Sin embargo, la característica informativa cuenta con diferentes mensajes que se interpretan en el sistema provenientes de la aplicación móvil. Esta codificación se ve representada en la tabla 15:

Código	**Interpretación**
Reminder	Acceso a los datos almacenados en la EEPROM con la última sesión guardada
Ready	Emparejamiento exitoso entre la aplicación y el dispositivo
Email	Activación o desactivación del correo electrónico enviado al final del entrenamiento
WorkoutOn	Inicio de la detección de ejercicios
WorkoutOff	Finalización de la sesión de entrenamiento
LegsOn	Activación de la detección de ejercicios pertenecientes al tronco inferior
LegsOff	Desactivación de la detección de ejercicios de piernas
WakeUp	Activación del dispositivo tras el modo *sleep*
Sleep	Entrada al modo hibernación del dispositivo

Tabla 15: Códigos informativos enviados por la aplicación e interpretados por el sistema

6.2. Aplicación Android/iOS

Para el desarrollo de la aplicación funcional en los sistemas operativos móviles *Android* e *iOS* se utiliza la plataforma *web* denominada *Thunkable*, que permite la creación de *apps* nativas en un sólo *framework* de una manera intuitiva y práctica, con funciones *live* orientadas a la puesta en práctica constante mediante sincronización entre el servidor de la plataforma y su *app* correspondiente. Además, cuenta con sistemas avanzados de lectura de características *Bluetooth*, además de componentes básicos como botones, *switches*, etiquetas, *timers* o controles.

Tal y como se ha visto en el punto 5.1, se programan cuatro pantallas con sus respectivas opciones, lo que permite al usuario un manejo cómodo y rápido del sistema de cara al control del dispositivo, así como la visualización de los parámetros detectados a lo largo de las sesiones. En la figura 25 se puede observar el panel de edición de los bloques correspondientes a cada elemento de la pantalla.

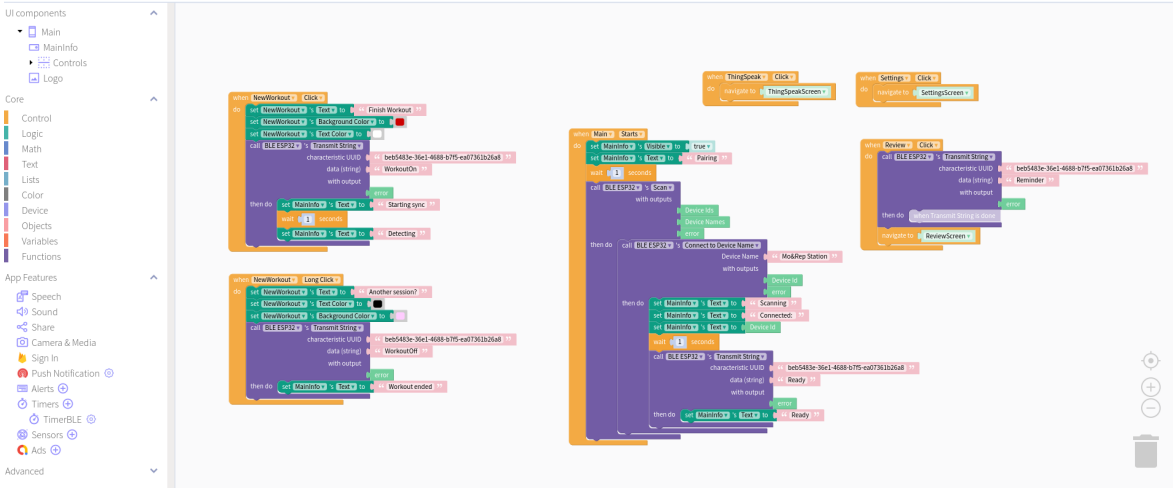


Figura 25: Pantalla de edición de bloques para la creación de una app

6.3. *Sleep Mode*

De cara a mejorar la autonomía de los dispositivos, se programa una función dedicada a la reducción de la frecuencia de reloj del microcontrolador. Este funciona de forma activa a 240 MHz, pudiendo reducirse hasta los 10 MHz. Sin embargo, para despertarlo de nuevo se necesita de la aplicación, que trabaja a través de *Bluetooth*, y este protocolo está limitado a los 80 MHz.

Es por ello que, una vez detecta que no se está realizando ningún entrenamiento, mediante las variables "workoutctrl" y un tiempo estimado de 30 minutos, el sistema limita su frecuencia a 80 MHz, y se desactivan las tareas correspondientes al MPU-6050 (acelerómetro). De esta manera, sólo se ejecuta de forma cíclica la lectura del servicio de BLE, reduciendo la carga de trabajo del microcontrolador y aumentando la autonomía de la batería.

6.4. Corrección de posturas

A modo de testeo y programación en versión *alpha*, se implementa un sistema que compara las aceleraciones y orientaciones medias durante el desarrollo de un ejercicio con los valores ideales que deberían tener estas. Se hace una estimación y se determina que la inclinación de los movimientos no debe sobrepasar un el 50 % de las ideales. Matemáticamente, se puede considerar como límite 1.5 veces el valor de la orientación ideal para el nodo inferior, que corresponde con la muñeca. Este sistema está diseñado para la futura implementación de un nodo central colocado en el torso, que permitirá corregir posturas de la espalda a la hora de realizar una sesión de entrenamiento, ya que es la parte del cuerpo más susceptible a lesiones y con mayor índice de gravedad.

La comparación se realiza una vez finalizada la serie, indicándose un mensaje en el *display* del nodo de la muñeca con el texto "Corrige tu postura".

6.5. Ejercicios del tronco inferior

Se estudia el desarrollo del sistema diseñado en ejercicios del tronco inferior, siendo un total de 6, entre los que destacan sentadillas, *curl* de cuádriceps o peso muerto.

El funcionamiento es el mismo que para el tronco superior, ya que se coloca uno de los nodos en el cuádriceps y otro por debajo de la rodilla, ambos por la cara interna de la pierna, para evitar presiones y golpes con las máquinas propias de un gimnasio. Cabe destacar que, para el nodo *slave*, la máquina de estados tendrá 8 estados, al igual que el *master*, ya que se hace uso de la posición "Eta", que ocupa el estado 7. En la tabla 16 se refleja, tal y como se hizo con el tronco superior, las condiciones a las que están sometidos los nodos durante el transcurso de cada ejercicio.

Ejercicio	Número	Grupo muscular	Nodo inferior	Nodo superior	Condición N.I.	Condición N.S.	Código N.I.	Código N.S.
Sentadillas	Exercise[1]	Piernas	Eta-Eta	Épsilon-Beta	Estado fijo: 7	Cambio de estado(4-5) Estado inicial: 4	AM_07	Ninguno
Curl de cuádriceps	Exercise[2]	Piernas	Eta-Mi	Beta-Beta	Cambio de estado(7-4) Estado inicial: 7	Estado fijo: 5	Ninguno	WM_05
Peso muerto	Exercise[3]	Piernas	Eta-Eta	Beta-Épsilon	Estado fijo: 7	Cambio de estado(5-4) Estado inicial: 5	AM_07	Ninguno
Prensa	Exercise[4]	Piernas	Kappa-Kappa	Gamma-Dseta	Estado fijo: 2	Cambio de estado(3-7) Estado inicial: 3	AM_02	Ninguno
Curl de femoral	Exercise[5]	Piernas	Mi-Eta	Beta-Beta	Cambio de estado(4-7) Estado inicial: 4	Estado fijo: 4	Ninguno	WM_05
Curl de femoral 2	Exercise[6]	Piernas	Lambda-Kappa	Dseta-Dseta	Cambio de estado(6-2) Estado inicial: 6	Estado fijo: 7	Ninguno	WM_07

Tabla 16: Condiciones de ambos nodos para la detección de los ejercicios de tronco inferior

Validación y verificación

7. Validación y verificación

Se realizan una serie de pruebas en entornos controlados. Se colocan las respectivas PCBs en la prenda personalizada y se ejecuta el proceso para la detección de ejercicios como se ha mencionado en el capítulo 5. Tras esto, se ejecutan una serie de ejercicios y series para comprobar la eficacia del sistema. El resultado de estas pruebas se adjuntan en las tablas 17, 18, 19, 20, 21 y 22:

Entreno	Ejercicios	Series reales	Reps reales	Series detectadas	Reps detectadas	% acierto
1	Curl concentrado	4	12	4	12	100 %
			10		10	
			8		8	
			6		6	
	Martillo	4	12	4	12	100 %
			10		10	
			8		8	
			6		6	
	Curl inverso	4	12	4	12	97,22 %
			10		10	
			8		8	
			6		5	
	Curl simple	4	12	4	12	100 %
			10		10	
			8		8	
			6		6	

Tabla 17: Pruebas realizadas para el entrenamiento de bíceps

Entreno	Ejercicios	Series reales	Reps reales	Series detectadas	Reps detectadas	% acierto
2	Extensión inversa	4	12	4	11	94,44 %
			10		11	
			8		8	
			6		6	
	Extensión con barra	4	12	4	12	100 %
			10		10	
			8		8	
			6		6	
	Extensión vertical	4	12	4	12	100 %
			10		10	
			8		8	
			6		6	
	Fondos	4	12	4	12	100 %
			10		10	
			8		8	
			6		6	

Tabla 18: Pruebas realizadas para el entrenamiento de tríceps

Entreno	Ejercicios	Series reales	Reps reales	Series detectadas	Reps detectadas	% acierto
3	Dominadas	4	12	4	12	100 %
			10		10	
			8		8	
			6		6	
	Jalón al pecho	4	12	4	10	91,67 %
			10		10	
			8		7	
			6		6	
	Remo horizontal	4	12	4	10	94,44 %
			10		10	
			8		7	
			6		6	
	Pull-over	4	12	4	11	91,67 %
			10		10	
			8		7	
			6		5	

Tabla 19: Pruebas realizadas para el entrenamiento de espalda

Entreno	Ejercicios	Series reales	Reps reales	Series detectadas	Reps detectadas	% acierto
4	Press	4	12	4	11	88,89 %
			10		9	
			8		6	
			6		6	
	Levantamiento	4	12	4	12	100 %
			10		10	
			8		8	
			6		5	
	Apertura	4	12	4	12	100 %
			10		10	
			8		8	
			6		6	
	Flexiones	4	12	4	12	100 %
			10		10	
			8		8	
			6		6	

Tabla 20: Pruebas realizadas para el entrenamiento de pecho

Entreno	Ejercicios	Series reales	Reps reales	Series detectadas	Reps detectadas	% acierto
5	Vuelo	4	12	4	11	91,67 %
			10		10	
			8		8	
			6		4	
	Press	4	12	4	12	94,44 %
			10		10	
			8		7	
			6		5	
	Remo al cuello	4	12	4	12	100 %
			10		10	
			8		8	
			6		6	
	Elevación lateral	4	12	4	12	91,67 %
			10		9	
			8		6	
			6		6	

Tabla 21: Pruebas realizadas para el entrenamiento de hombros

Entreno	Ejercicios	Series reales	Reps reales	Series detectadas	Reps detectadas	% acierto
6	Sentadilla	4	12	4	12	100 %
			10		10	
			8		8	
			6		6	
	Curl de cuádriceps	4	12	4	11	94,44 %
			10		10	
			8		8	
			6		5	
	Curl de femoral	4	12	4	11	91,67 %
			10		8	
			8		8	
			6		6	

Tabla 22: Pruebas realizadas para el entrenamiento de piernas

Haciendo un porcentaje de acierto total del 96,62 %. Cabe destacar que el uso de tecnología BLE produce una pequeña latencia a la hora de leer las características y escribirlas al mismo tiempo, lo que perjudica en algunos casos la detección de la primera repetición. Se realizaron pruebas utilizando una *mesh WiFi* con la misma metodología, y al tener mayor capacidad se producía un funcionamiento en tiempo real más eficaz que el que se produce con *Bluetooth Low Energy*. Además, el sistema tiene dificultades con aquellos ejercicios que dependen del movimiento en un único eje donde no se modifica las posiciones iniciales y finales. Sin embargo, es muy eficaz en aquellos ejercicios articulados, tales como *curl* de bíceps, dominadas o extensión vertical. Igualmente, el sistema implementado supera a alternativas similares, tanto en el mercado, como en el ámbito de la investigación, ya que se comporta de manera autónoma registrando una amplia variedad de ejercicios con un porcentaje de acierto muy elevado.

Conclusiones y potenciales aplicaciones

8. Conclusiones y potenciales aplicaciones

En el presente documento se ha desarrollado la creación de un sistema de detección de movimientos y parámetros biomédicos orientados al sector deportivo, y en concreto, el *fitness*, resumido en la aplicación de un sensor capaz de hallar las aceleraciones y orientación de un nodo colocado en la parte inferior del brazo, y la comunicación de estos valores con un homónimo situado a la zona alta de la misma extremidad. Mediante la creación de una interfaz integrada en una aplicación para dispositivos móviles, se recogen las repeticiones realizadas para un conjunto de 36 ejercicios preprogramados, agrupándose en series que se registran y se almacenan para el análisis posterior de la sesión de entrenamiento.

En primer lugar se ha realizado un estudio de mercado, así como del ámbito de la investigación, para buscar el punto de innovación que requiere el proyecto que se presenta, tras lo cual se alcanza la conclusión de haber desarrollado un sistema diferenciador en términos de amplitud de ejercicios y practicidad de uso respecto a lo visto hasta el momento. Una vez finalizado este apartado, se destacan los componentes en términos de *hardware* en los que se basa el dispositivo, haciendo hincapié en sus principales circuitos integrados (MPU-6050 y ESP32-WROOM-32D), dando el salto posteriormente a los fundamentos teóricos escondidos tras el funcionamiento de cada apartado relevante, entre los que se destacan el protocolo *Bluetooth Low Energy*, SMTP para el envío de correos electrónicos, el servidor HTTP y el funcionamiento del IMU incorporado. Se continúa con la explicación en detalle del procedimiento que realiza el algoritmo, profundizando en la codificación de los mensajes y en la distribución de la detección de los ejercicios programados en los nodos correspondientes, adjuntando tablas y figuras que los ejemplifican y que sintetizan la información redactada. La validación de los datos obtenidos en entornos controlados, con un porcentaje de acierto del 96,62 % y la descripción de las posibles mejoras a implementar vienen precedidas por las funcionalidades adicionales que se incorporan, donde se mencionan las plataformas, bibliotecas y funciones que complementan al código principal.

De cara a una línea futura de desarrollo, se considera la implementación de mecanismos de *machine learning* para una detección más precisa de ejercicios y rutinas, y la posibilidad de incrementar el número de estos mediante ejemplos personalizados contrastados con una base de datos. Esto se puede representar con el estudio por parte del algoritmo de los grupos musculares que se trabajan en función al día de la semana, algo que es muy popular entre los practicantes del *fitness*. A partir de esto, se podrían implementar probabilidades en la realización de otros ejercicios o en el trabajo de otros músculos, afinando la detección de las sesiones. De cara a la corrección de posturas, un sistema de mayor eficacia tendría que tener en cuenta cada ejercicio de forma individual y detallar cómo se debe corregir la postura para los tres nodos propuestos (antebrazo, brazo y espalda) basándose en el sistema que se implementa de forma experimental en el presente proyecto.

Más allá del sector deportivo, el uso de un circuito integrado como el MPU-6050, capaz de medir aceleración y orientación podría ser útil para determinados colectivos en el ámbito sanitario. Un ejemplo es la detección de ataques repentinos en personas que sufran enfermedades como el Parkinson. Utilizándolo con el sistema de frecuencia cardíaca, que podría comprobar un aumento de esta de forma instantánea, podría evaluar el riesgo de la situación y conectarse de forma remota al servidor del centro médico más cercano, o incluso podría tener la capacidad de ir administrando, junto a la ayuda de un profesional, las dosis correspondientes a la medicación que regula estos ataques, ya que se detallaría de una forma exacta la frecuencia a la que surgen estos.

8.1. Conclusions and potential applications

In the present document, the creation of a movements detection system and biomedical parameters oriented to the sports sector, specifically, fitness, is developed, summarized in the application of a sensor capable of detecting the accelerations and orientation of a node situated in the low part of the arm, and the communication of these values with the same structure located in the high part of it. Through the creation of an integrated interface inside a smartphone app, the repetitions for 36 different preprogrammed exercises are recollected and grouped in series that are recorded and stored for the consequent analysis of the workout session.

First of all, it has been done a study for the global market, in addition to the research sector, to look for the point of innovation that practices the project that is presented. After this, the conclusion to do a different system in terms of variety of exercises and practicality is reached, and once finished this part, all the components related to the hardware are highlighted, focusing on the main integrated circuits of the device (MPU-6050 and ESP32-WROOM-32D), taking the leap to the theoretical fundaments that are below each protocol that is used, like Bluetooth Low Energy or SMTP for the sending of an email at the end of the session. Also, the physical concepts of the used IMU are established. Next to it, the code is explained with details, including the practical process of the algorithm, the codification of the messages, and the distribution of the detection of the exercises in both nodes, attaching tables and figures that work as examples and help to comprehend the written information. The validation of the obtained values in controlled environments, with a success rate of 96,62 % and the description of the possible improvements, is preceded by the additional functions that are implemented, where the platforms, libraries, and complementary functions of the main code are mentioned.

Facing a future development line, the implementation of mechanisms based on deep learning are considered for more accurate detection of exercises and routines and the possibility of increasing the quantity of these through customized examples contrasted with a database. This can be represented with the study of the algorithm of the muscle groups that are worked depending on the day of the week, something that is so popular among practitioners of fitness. From this, it could be implemented probabilities doing other exercises or the work of other muscles, increasing the accuracy of the detections. Regarding fixing the postures, a more efficient system would have to consider each exercise and detail how the posture can be fixed for the two (or three) different nodes, based on the system implemented in the present project.

Beyond the sports sector, the use of an integrated circuit such as the MPU-6050, capable of measuring acceleration and orientation, could be helpful for determining collectives in the health field. An example is the detection of sudden attacks in people that suffer from diseases like Parkinson's. Using it with the heart-rate sensor, that could check an increase of this instantly; it could be evaluated the risk of the situation and connect remotely to the server of the closest medical center, or even could have the capability of manage, with the help of a professional, the corresponding doses to the medication that controls these attacks, because the frequency of these could be detailed.

Referencias

[1] «Internet de las cosas: cuando todo está conectado,» 2020. dirección: https://www.lavanguardia.com/vida/junior-report/20190301/46752655177/internet-cosas-dispositivos-conectados-iot.html.

[2] «Europa sigue enganchada al mundo fitness,» 2017. dirección: https://www2.deloitte.com/es/es/pages/life-sciences-and-healthcare/articles/europa-enganchada-fitness.html.

[3] «Xiaomi Mi Smart Band 6, análisis: tan recomendable e imperfecta como siempre,» 2021. dirección: https://www.xataka.com/analisis/xiaomi-mi-smart-band-6-analisis-caracteristicas-precio-especificaciones.

[4] «Honor Band 5, análisis: una pulsera todoterreno y versátil que consolida a Honor como referente en calidad precio,» 2019. dirección: https://www.xataka.com/analisis/honor-band-5-analisis-caracteristicas-precio-especificaciones.

[5] «Huawei Band 4 Pro: características y especificaciones,» 2021. dirección: https://consumer.huawei.com/es/wearables/band4-pro/.

[6] «Explicamos los sensores del Apple Watch y en qué modelos están,» 2020. dirección: https://lamanzanamordida.net/tutoriales/apple-watch/sensores-apple-watch-explicados/.

[7] «OPPO Watch, análisis: el primer smartwatch de OPPO con Wear OS llega presumiendo de calidad de construcción,» 2020. dirección: https://www.xataka.com/analisis/oppo-watch-analisis-caracteristicas-precio-especificaciones.

[8] «Samsung Galaxy Watch Active 2, análisis: el que debió ser el primer (y quizás único) Active,» 2020. dirección: https://www.xataka.com/relojes-inteligentes/samsung-galaxy-watch-%20active-2-analisis-caracteristicas-precio-especificaciones.

[9] «Suunto Ambit3, todo sobre el reloj más avanzado de Suunto,» 2020. dirección: https://www.palabraderunner.com/suunto-ambit3/.

[10] «Casio SGW-1000-2BER, características y especificaciones,» 2020. dirección: https://www.casio-europe.com/es/productos/relojes/collection/sgw-1000-2ber/.

[11] «Review Yamay Fitness Tracker,» 2020. dirección: https://pulserasdeactividad.net/yamay-fitness-tracker-review/.

[12] «Huawei Band 6, análisis: érase una vez una pulsera que quería ser smartwatch (y se quedó a medio camino),» 2021. dirección: https://www.xataka.com/analisis/huawei-band-6-analisis-caracteristicas-precio-especificaciones.

[13] «Atlas Multi-Trainer 3: Un smartwatch deportivo para el gimnasio,» 2019. dirección: https://www.smartwatchzone.net/atlas-multi-trainer-3-reloj-gimnasio/.

[14] «Push band, análisis: ¿tu compañero ideal para levantar pesas en el gimnasio?,» 2016. dirección: https://www.xataka.com/analisis/push-band-analisis-tu-companero-ideal-para-levantar-pesas-en-el-gimnasio.

[15] «PONEMOS A PRUEBA EL BEAST SENSOR,» 2016. dirección: https://powerexplosive.com/ponemos-a-prueba-el-beast-sensor/.

[16] «Gymwatch, el cuantificador optimizado para utilizar en el gimnasio,» 2014. dirección: https://www.xataka.com/otros/gymwatch-el-cuantificador-perfecto-para-utilizar-en-el-gimnasio.

[17] «Athos: la ropa que asiste tu entrenamiento,» 2013. dirección: https://www.vitonica.com/equipamiento/athos-la-ropa-que-asiste-tu-entrenamiento.

[18] M. Maheedhar, A. Gaurav, V. Jilla, V. N. Tiwari y R. Narayanan, «StayFit: A wearable application for Gym based power training,» IEEE Conference Publication, 2016. dirección: https://ieeexplore.ieee.org/document/7592166.

[19] C. Lee, «Movement detection and analysis of resistance exercises for smart fitness platform,» IEEE Conference Publication, 2017. dirección: https://ieeexplore.ieee.org/document/7993818.

[20] E. A. Akpa, M. Fujiwara, Y. Arakawa, H. Suwa y K. Yasumoto, «GIFT: Glove for Indoor Fitness Tracking System,» IEEE Conference Publication, 2018. dirección: https://ieeexplore.ieee.org/document/8480211.

[21] A. Nagarkoti, R. Teotia, A. K.Mahale y P. K.Das, «Realtime Indoor Workout Analysis Using Machine Learning & Computer Vision,» IEEE Conference Publication, 2019. dirección: https://ieeexplore.ieee.org/document/8856547.

[22] «Protocolo SMTP: Cómo Se Envían y Reciben Los Emails A Través de Internet,» 2020. dirección: https://www.xatakamovil.com/conectividad/protocolo-smtp-como-se-envian-y-reciben-los-emails-a-traves-de-internet.

[23] «Servidor web,» 2022. dirección: https://es.wikipedia.org/wiki/Servidor_web.

Anexos

Código correspondiente al nodo inferior

```
#include <ESP_Mail_Client.h>
#include <Arduino.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include "I2Cdev.h"
#include "MPU6050.h"
#include "string.h"
#include <EEPROM.h>
#include <Separador.h>
#include "ThingSpeak.h"
#include "secrets.h"
#include "MAX30105.h"
#include "spo2_algorithm.h"
#include "heartRate.h"
#include <WiFi.h>
#include <WebServer.h>
#include <NimBLEDevice.h>
#include <TaskScheduler.h>

Separador s;

//MAX30102EFD+
MAX30105 particleSensor;

const byte RATE_SIZE = 4;
byte rates[RATE_SIZE];
byte rateSpot = 0;
long lastBeat = 0;

float beatsPerMinute;
int beatAvg;

Scheduler               MoRepScheduler;
```

```
35
36 //WebServer
37 WiFiServer server(80);
38 unsigned long currentTime = millis();
39 unsigned long previousTime = 0;
40 const long timeoutTime = 2000;
41 String head;
42 bool localip = false;
43
44 //BLE
45 static NimBLEServer* pServer;
46 #define CHARACTERISTIC_UUID_WRIST "d6597e78-c992-4f43-9861-719
      b78932fa6"
47 #define CHARACTERISTIC_UUID_ARM "84d8a625-08f9-49b3-82fb-d85daffc3c39
      "
48 #define CHARACTERISTIC_UUID_INFO "beb5483e-36e1-4688-b7f5-
      ea07361b26a8"
49 NimBLEService* pSvc;
50 std::string valueW = "WM_0";
51 std::string valueA = "AM_0";
52 std::string header;
53
54 //SMTP
55 SMTPSession smtp;
56 void smtpCallback(SMTP_Status status);
57
58 //ThingSpeak
59 unsigned long myChannelNumber = SECRET_CH_ID;
60 const char * myWriteAPIKey = SECRET_WRITE_APIKEY;
61 int keyIndex = 0;
62 WiFiClient client;
63 int httpCode;
64
65 //I2C Chain
66 void scan_i2c();
67 byte i2c_address[5];
68 unsigned int nDevices = 0;
69
```

```
70  Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
       OLED_RESET);
71
72  MPU6050 mpu(0x68);
73  int16_t ax, ay, az;
74  int16_t gx, gy, gz;
75  float ang_x, ang_y, ang_z;
76  float ang_x_prev[20], ang_y_prev[20], ang_z_prev[20];
77  float accel_ang_x;
78  float accel_ang_y;
79  float accel_ang_z;
80
81  unsigned long t_prev;
82  float dt;
83  int j = 1;
84  int k = 0;
85  unsigned long tmove;
86  unsigned long tmove_prev;
87  unsigned long t_control;
88  int mins = 0;
89  unsigned long StartTime, FinishedTime, ElapsedTime, Over;
90
91  struct myStruct {
92    int Series [30] = {0};
93    int Exercise [35] = {0};
94    String Ex[100];
95    int MuscGroup[5] = {0};
96    int CountSeries[10] = {0};
97    int kcal = 0;
98  };
99
100 myStruct WORKOUT;
101
102 String WristPosition;
103 int WristPos;
104 int IniState = 1;
105 int state = 1;
106 int state_prev = 1;
```

```
107 bool Push = 0;
108 bool Pull = 0;
109 int IniMove = 0;
110 float ax_m_s2_prev[10];
111 float ay_m_s2_prev[10];
112 float az_m_s2_prev[10];
113 float angx_av = 0;
114 float angy_av = 0;
115 float angz_av = 0;
116 float ax_av, ay_av, az_av = 0;
117 float accel_av[2] = {0};
118 float accel_id[5] = {0};
119 int it = 0;
120 bool order = false;
121 int workoutctrl = 0;
122 std::string Review = "";
123 String Reminder = "";
124 std::string Analysis = "";
125
126 int eeaddress = 0;
127 bool mail = true;
128 int journey;
129 std::string WristMove = "";
130 std::string ArmMove;
131 int ArmEx;
132 int ArmRep;
133 int ExDisp;
134 bool env;
135
136 //Functions
137 void Postures();
138 void FinishWorkout();
139 void ReadingEEPROM();
140 void webServer();
141
142 class ServerCallbacks: public NimBLEServerCallbacks {
143     void onConnect(NimBLEServer* pServer) {
144         Serial.println("Client connected");
```

```
145         Serial.println("Multi-connect support: start advertising");
146         NimBLEDevice::startAdvertising();
147     };
148     void onConnect(NimBLEServer* pServer, ble_gap_conn_desc* desc) {
149         Serial.print("Client address: ");
150         Serial.println(NimBLEAddress(desc->peer_ota_addr).toString().
               c_str());
151         pServer->updateConnParams(desc->conn_handle, 24, 48, 0, 60);
152     };
153     void onDisconnect(NimBLEServer* pServer) {
154         Serial.println("Client disconnected - start advertising");
155         NimBLEDevice::startAdvertising();
156     };
157     void onMTUChange(uint16_t MTU, ble_gap_conn_desc* desc) {
158         Serial.printf("MTU updated: %u for connection ID: %u\n", MTU,
                desc->conn_handle);
159     };
160
161
162     uint32_t onPassKeyRequest(){
163         Serial.println("Server Passkey Request");
164         return 123456;
165     };
166
167     bool onConfirmPIN(uint32_t pass_key){
168         Serial.print("The passkey YES/NO number: ");Serial.println(
               pass_key);
169         return true;
170     };
171
172     void onAuthenticationComplete(ble_gap_conn_desc* desc){
173         if(!desc->sec_state.encrypted) {
174             NimBLEDevice::getServer()->disconnect(desc->conn_handle);
175             Serial.println("Encrypt connection failed - disconnecting
                    client");
176             return;
177         }
178         Serial.println("Starting BLE work!");
```

```
179     };
180 };
181
182 class CharacteristicCallbacks: public NimBLECharacteristicCallbacks {
183     void onRead(NimBLECharacteristic* pCharacteristic){
184     };
185     void onWrite(NimBLECharacteristic* pCharacteristic) {
186     };
187     void onNotify(NimBLECharacteristic* pCharacteristic) {
188     };
189
190     void onStatus(NimBLECharacteristic* pCharacteristic, Status
           status, int code) {
191         String str = ("Notification/Indication status code: ");
192         str += status;
193         str += ", return code: ";
194         str += code;
195         str += ", ";
196         str += NimBLEUtils::returnCodeToString(code);
197         Serial.println(str);
198     };
199
200     void onSubscribe(NimBLECharacteristic* pCharacteristic,
           ble_gap_conn_desc* desc, uint16_t subValue) {
201         String str = "Client ID: ";
202         str += desc->conn_handle;
203         str += " Address: ";
204         str += std::string(NimBLEAddress(desc->peer_ota_addr)).c_str
               ();
205         if(subValue == 0) {
206             str += " Unsubscribed to ";
207         }else if(subValue == 1) {
208             str += " Subscribed to notfications for ";
209         } else if(subValue == 2) {
210             str += " Subscribed to indications for ";
211         } else if(subValue == 3) {
212             str += " Subscribed to notifications and indications for
                   ";
```

```
213         }
214         str += std::string(pCharacteristic->getUUID()).c_str();
215 
216         Serial.println(str);
217     };
218 };
219 
220 class DescriptorCallbacks : public NimBLEDescriptorCallbacks {
221     void onWrite(NimBLEDescriptor* pDescriptor) {
222         std::string dscVal((char*)pDescriptor->getValue(),
               pDescriptor->getLength());
223     };
224 
225     void onRead(NimBLEDescriptor* pDescriptor) {
226     };
227 };
228 
229 static DescriptorCallbacks dscCallbacks;
230 static CharacteristicCallbacks chrCallbacks;
231 
232 NimBLECharacteristic* pWristCharacteristic;
233 NimBLECharacteristic* pArmCharacteristic;
234 NimBLECharacteristic* pInfoCharacteristic;
235 
236 const unsigned char LogoMoRep [] PROGMEM = {
237   0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0
      x00, 0x00, 0x00, 0x00, 0x00,
238   0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0
      x00, 0x00, 0x00, 0x00, 0x00,
239   0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0
      x00, 0x00, 0x00, 0x00, 0x00,
240   0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0
      x00, 0x00, 0x00, 0x00, 0x00,
241   0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0
      x00, 0x00, 0x00, 0x00, 0x00,
242   0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0
      x00, 0x00, 0x00, 0x00, 0x00,
```

```
243     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0
            x00, 0x00, 0x00, 0x00, 0x00,
244     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0
            x00, 0x00, 0x00, 0x00, 0x00,
245     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0
            x00, 0x00, 0x00, 0x00, 0x00,
246     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0
            x00, 0x00, 0x00, 0x00, 0x00,
247     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0
            x00, 0x00, 0x00, 0x00, 0x00,
248     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0
            x00, 0x00, 0x00, 0x00, 0x00,
249     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0
            x00, 0x00, 0x00, 0x00, 0x00,
250     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0
            x00, 0x00, 0x00, 0x00, 0x00,
251     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0
            x00, 0x00, 0x00, 0x00, 0x00,
252     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0
            x00, 0x00, 0x00, 0x00, 0x00,
253     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0
            x00, 0x00, 0x00, 0x00, 0x00,
254     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0
            x00, 0x00, 0x00, 0x00, 0x00,
255     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0
            x00, 0x00, 0x00, 0x00, 0x00,
256     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0
            x00, 0x00, 0x00, 0x00, 0x00,
257     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0
            x00, 0x00, 0x00, 0x00, 0x00,
258     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0
            x00, 0x00, 0x00, 0x00, 0x00,
259     0x00, 0x00, 0x00, 0x00, 0x00, 0x1c, 0x00, 0x01, 0xc0, 0x00, 0
            x00, 0x00, 0x00, 0x00, 0x00,
260     0x00, 0x00, 0x80, 0x10, 0x00, 0xff, 0x80, 0x07, 0xe0, 0x03, 0
            xf8, 0x0f, 0xfc, 0x7f, 0x00, 0x00,
261     0x00, 0x00, 0xc0, 0x18, 0x03, 0xff, 0xe0, 0x0f, 0xf0, 0x03, 0
            xfe, 0x0f, 0xfc, 0x7f, 0xc0, 0x00,
```

```
262     0x00, 0x00, 0xc0, 0x18, 0x07, 0xc0, 0xf0, 0x0c, 0x38, 0x03, 0x0f, 0
            x0c, 0x00, 0x71, 0xe0, 0x00,
263     0x00, 0x00, 0xe0, 0x38, 0x0f, 0x00, 0x78, 0x0c, 0x38, 0x03, 0x07, 0
            x0c, 0x00, 0x70, 0x70, 0x00,
264     0x00, 0x01, 0xe0, 0x38, 0x0e, 0x00, 0x3c, 0x0c, 0x38, 0x03, 0x03, 0
            x0c, 0x00, 0x70, 0x70, 0x00,
265     0x00, 0x01, 0xe0, 0x7c, 0x1c, 0x00, 0x1c, 0x0e, 0x70, 0x03, 0x03, 0
            x0c, 0x00, 0x70, 0x70, 0x00,
266     0x00, 0x01, 0xf0, 0x7c, 0x1c, 0x00, 0x0c, 0x07, 0xe0, 0x03, 0x03, 0
            x0c, 0x00, 0x70, 0x70, 0x00,
267     0x00, 0x01, 0xb0, 0x6c, 0x18, 0x00, 0x0e, 0x03, 0xc0, 0x03, 0x07, 0
            x0f, 0xfc, 0x70, 0xe0, 0x00,
268     0x00, 0x03, 0xb8, 0xec, 0x18, 0x00, 0x0e, 0x07, 0xc0, 0x03, 0xfe, 0
            x0f, 0xfc, 0x7f, 0xe0, 0x00,
269     0x00, 0x03, 0x18, 0xce, 0x18, 0x00, 0x0e, 0x1f, 0xe0, 0x03, 0xfc, 0
            x0c, 0x00, 0x7f, 0x80, 0x00,
270     0x00, 0x03, 0x19, 0xc6, 0x18, 0x00, 0x0e, 0x3c, 0xf1, 0x03, 0xf0, 0
            x0c, 0x00, 0x7c, 0x00, 0x00,
271     0x00, 0x03, 0x1d, 0x86, 0x1c, 0x00, 0x0c, 0x38, 0x7b, 0x83, 0x70, 0
            x0c, 0x00, 0x70, 0x00, 0x00,
272     0x00, 0x07, 0x0f, 0x86, 0x1c, 0x00, 0x1c, 0x70, 0x3f, 0x03, 0x38, 0
            x0c, 0x00, 0x70, 0x00, 0x00,
273     0x00, 0x06, 0x0f, 0x87, 0x0e, 0x00, 0x3c, 0x70, 0x1e, 0x03, 0x1c, 0
            x0c, 0x00, 0x70, 0x00, 0x00,
274     0x00, 0x06, 0x07, 0x03, 0x07, 0x00, 0x78, 0x70, 0x1e, 0x03, 0x1c, 0
            x0c, 0x00, 0x70, 0x00, 0x00,
275     0x00, 0x0e, 0x07, 0x03, 0x07, 0xc0, 0xf0, 0x38, 0x3f, 0x03, 0x0e, 0
            x0c, 0x00, 0x70, 0x00, 0x00,
276     0x00, 0x0e, 0x06, 0x03, 0x83, 0xff, 0xe0, 0x3f, 0xf3, 0x83, 0x07, 0
            x0f, 0xfc, 0x70, 0x00, 0x00,
277     0x00, 0x0c, 0x02, 0x03, 0x80, 0xff, 0x80, 0x0f, 0xe1, 0xc3, 0x07, 0
            x0f, 0xfc, 0x70, 0x00, 0x00,
278     0x00, 0x00, 0x00, 0x00, 0x00, 0x1c, 0x00, 0x03, 0x80, 0x00, 0x00, 0
            x00, 0x00, 0x00, 0x00, 0x00,
279     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0
            x00, 0x00, 0x00, 0x00, 0x00,
280     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x02, 0x00, 0
            x02, 0x00, 0x80, 0x00, 0x00,
```

```
281     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x02, 0x01, 0
            x02, 0x00, 0x80, 0x00, 0x00,
282     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x40, 0x04, 0x22, 0x84, 0
            x22, 0x08, 0x89, 0xc0, 0x00,
283     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x2a, 0x0b, 0x4a, 0x49, 0
            x4a, 0x12, 0x84, 0x40, 0x00,
284     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x26, 0x11, 0x4a, 0x01, 0
            x7a, 0x02, 0xbc, 0x80, 0x00,
285     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x24, 0x0b, 0x4a, 0x41, 0
            x42, 0x12, 0xa4, 0x80, 0x00,
286     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x44, 0x05, 0x20, 0x80, 0
            x20, 0x0a, 0x08, 0x00, 0x00,
287     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0a, 0x00, 0x00, 0
            x00, 0x1c, 0x00, 0x00, 0x00,
288     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0
            x00, 0x00, 0x00, 0x00, 0x00,
289     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0
            x00, 0x00, 0x00, 0x00, 0x00,
290     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0
            x00, 0x00, 0x00, 0x00, 0x00,
291     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0
            x00, 0x00, 0x00, 0x00, 0x00,
292     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0
            x00, 0x00, 0x00, 0x00, 0x00,
293     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0
            x00, 0x00, 0x00, 0x00, 0x00,
294     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0
            x00, 0x00, 0x00, 0x00, 0x00,
295     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0
            x00, 0x00, 0x00, 0x00, 0x00,
296     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0
            x00, 0x00, 0x00, 0x00, 0x00,
297     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0
            x00, 0x00, 0x00, 0x00, 0x00,
298     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0
            x00, 0x00, 0x00, 0x00, 0x00,
299     0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0
            x00, 0x00, 0x00, 0x00, 0x00,
```

```
300   0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0
         x00, 0x00, 0x00, 0x00, 0x00
301 };
302
303
304 Task taskMPULegs(TASK_SECOND / 5, TASK_FOREVER, []() {
305   if (workoutctrl == 1){
306     order = true;
307     WristMove = "1";
308     workoutctrl = 0;
309   }
310   if (order == true) {
311     if (tmove - t_prev > 1500){
312       t_prev = tmove;
313     }
314
315   float ax_m_s2, ay_m_s2, az_m_s2;
316   mpu.getAcceleration(&ax, &ay, &az); //raw values of accelerations
317   mpu.getRotation(&gx, &gy, &gz); //raw values of orientations
318   dt = (tmove - t_prev) / 1000.0;
319   tmove = millis();
320   ax_m_s2 = ax * (9.81 / 16384.0);
321   ay_m_s2 = ay * (9.81 / 16384.0);
322   az_m_s2 = az * (9.81 / 16384.0);
323   accel_ang_x = atan(ax / sqrt(pow(ay, 2) + pow(az, 2))) * (180.0 /
         3.14);
324   accel_ang_y = atan(ay / sqrt(pow(ax, 2) + pow(az, 2))) * (180.0 /
         3.14);
325   accel_ang_z = atan(az / sqrt(pow(ax, 2) + pow(ay, 2))) * (180.0 /
         3.14);
326   ang_x = 0.98 *  (gx / 131) * dt + 0.02 * accel_ang_x;
327   ang_y = 0.98 *  (gy / 131) * dt + 0.02 * accel_ang_y;
328   ang_z = 0.98 *  (gz / 131) * dt + 0.02 * accel_ang_z;
329   double a_total = abs(ax_m_s2) + abs(ay_m_s2) + abs(az_m_s2);
330   mins = tmove/60000;
331   accel_av[0] = ax_av; accel_av[1] = ay_av; accel_av[2] = az_av;
332
333
```

```
334   if (0.6 > dt > 0.5){
335     ang_x_prev[it] = ang_x;
336     ang_y_prev[it] = ang_y;
337     ang_z_prev[it] = ang_z;
338     angx_av = (angx_av + ang_x_prev[it])/(it+1);
339     angy_av = (angy_av + ang_y_prev[it])/(it+1);
340     angz_av = (angz_av + ang_z_prev[it])/(it+1);
341     ax_m_s2_prev[it] = ax_m_s2;
342     ay_m_s2_prev[it] = ay_m_s2;
343     az_m_s2_prev[it] = az_m_s2;
344     ax_av = (ax_av + ax_m_s2_prev[it])/(it+1);
345     ay_av = (ay_av + ay_m_s2_prev[it])/(it+1);
346     az_av = (az_av + az_m_s2_prev[it])/(it+1);
347     it++;
348   }
349
350   else if (it > 5){
351     it = 0;
352   }
353
354   if (az_m_s2 > 7) {
355     WristPosition = "Alpha";
356     WristPos = 1;
357   }
358   else if (az_m_s2 < -7) {
359     WristPosition = "Delta";
360     WristPos = 2;
361   }
362   else if (ax_m_s2 > 7) {
363     WristPosition = "Beta";
364     WristPos = 4;
365   }
366   else if (ax_m_s2 < -7) {
367     WristPosition = "Dseta";
368     WristPos = 3;
369   }
370   else if (ay_m_s2 < -7) {
371     WristPosition = "Epsilon";
```

```
372       WristPos = 5;
373     }
374     else if (ay_m_s2 > 7) {
375       WristPosition = "Gamma";
376       WristPos = 6;
377     }
378     else {
379       WristPosition = "Half way";
380       WristPos = 0;
381     }
382
383     switch (state) {
384       case (1): //State detection
385         if (ArmMove == "1" || header == "EX"){
386           IniMove = 0;
387           Push = 0;
388           Pull = 0;
389           WristMove = "1";
390         if (WristMove == "1" && ArmMove == "1"){
391           display.clearDisplay();
392           display.setTextSize(1.5);
393           display.setTextColor(WHITE);
394           display.setCursor(10, 1);
395           display.println("Waiting for a new        exercise");
396           display.display();
397         }
398         if (WristPosition == "Alpha") {
399           state = 2;
400           IniState = 2;
401           state_prev = 1;
402           tmove_prev = tmove;
403         }
404         if (WristPosition == "Gamma") {
405           IniState = 3;
406           state = 3;
407           state_prev = 1;
408           tmove_prev = tmove;
409         }
```

```
410         if (WristPosition == "Epsilon") {
411           state = 4;
412           IniState = 4;
413           state_prev = 1;
414           tmove_prev = tmove;
415         }
416         if (WristPosition == "Beta") {
417           state = 5;
418           IniState = 5;
419           state_prev = 1;
420           tmove_prev = tmove;
421         }
422         if (WristPosition == "Delta") {
423           state = 6;
424           IniState = 6;
425           state_prev = 1;
426           tmove_prev = tmove;
427         }
428         if (WristPosition == "Dseta") {
429           state = 7;
430           IniState = 7;
431           state_prev = 1;
432           tmove_prev = tmove;
433         }
434         else if (WORKOUT.Series[j] > 3) {
435           j++;
436           WristMove = "97";
437         }
438         else if (ArmMove == "97" && env == true) {
439           if (ArmEx == 2){
440             WORKOUT.Ex[j] = "Curl quadriceps(Legs)";
441             WORKOUT.MuscGroup[5]++;
442           }
443           if (ArmEx == 5 || ArmEx == 6){
444             WORKOUT.Ex[j] = "Curl femoral (Legs)";
445             WORKOUT.MuscGroup[5]++;
446           }
447           WORKOUT.Series[j] = ArmRep;
```

```
448           j++;
449           env = false;
450         }
451         }
452             break;
453
454       case (2): //Alpha
455        if (WristPos == 2){
456         state = 6;
457         state_prev = 2;
458         tmove_prev = tmove;
459        }
460        if (WristPos == 3){
461         state = 7;
462         state_prev = 2;
463         tmove_prev = tmove;
464        }
465        if (WristPos == 4){
466         state = 5;
467         state_prev = 2;
468         tmove_prev = tmove;
469        }
470        if (WristPos == 5){
471         state = 4;
472         state_prev = 2;
473         tmove_prev = tmove;
474        }
475        if (WristPos == 6){
476         state = 3;
477         state_prev = 2;
478         tmove_prev = tmove;
479        }
480        if (tmove - tmove_prev > 12000){
481           WristMove = "99";
482           if (ArmMove == "99"){
483             state = 1;
484             t_control = tmove;
485           }
```

```
486       }
487       if (a_total > 40) {
488        state = 8;
489        WristMove = "98";
490        tmove_prev = tmove;
491       }
492
493            break;
494
495      case (3): //Gamma
496       if (WristPos == 2){
497        state = 6;
498        state_prev = 3;
499        tmove_prev = tmove;
500       }
501       if (WristPos == 3){
502        if ((ArmMove == "02" || ArmMove == "99") && IniState == 3){
503          WORKOUT.Exercise[4]++;
504          display.clearDisplay();
505          display.setTextSize(1.5);
506          display.setTextColor(WHITE);
507          display.setCursor(10, 1);
508          display.print("  Detecting... \n Push\n");
509          display.display();
510        }
511        state = 7;
512        state_prev = 3;
513        tmove_prev = tmove;
514       }
515       if (WristPos == 4){
516        state = 5;
517        state_prev = 3;
518        tmove_prev = tmove;
519       }
520       if (WristPos == 5){
521        state = 4;
522        state_prev = 2;
523        tmove_prev = tmove;
```

```
524       }
525       if (WristPos == 1){
526        state = 2;
527        state_prev = 3;
528        tmove_prev = tmove;
529       }
530       if (tmove - tmove_prev > 12000){
531        WristMove = "99";
532        if (ArmMove == "99"){
533          state = 8;
534          t_control = tmove;
535        }
536       }
537       if (a_total > 40) {
538        state = 8;
539        WristMove = "98";
540        tmove_prev = tmove;
541       }
542            break;
543
544      case (4): //  psilon
545       if (WristPos == 2){
546        state = 6;
547        state_prev = 4;
548        tmove_prev = tmove;
549       }
550       if (WristPos == 3){
551        state = 7;
552        state_prev = 4;
553        tmove_prev = tmove;
554       }
555       if (WristPos == 4){
556        if ((ArmMove == "07" || ArmMove == "99") && IniState == 4){
557          WORKOUT.Exercise[1]++;
558          display.clearDisplay();
559          display.setTextSize(1.5);
560          display.setTextColor(WHITE);
561          display.setCursor(10, 1);
```

```
562           display.print("  Detecting... \n Squats\n");
563           display.display();
564         }
565         state = 5;
566         state_prev = 4;
567         tmove_prev = tmove;
568        }
569        if (WristPos == 1){
570         state = 2;
571         state_prev = 4;
572         tmove_prev = tmove;
573        }
574        if (WristPos == 6){
575         state = 3;
576         state_prev = 4;
577         tmove_prev = tmove;
578        }
579        if (tmove - tmove_prev > 12000){
580         WristMove = "99";
581         if (ArmMove == "99"){
582           state = 8;
583         }
584        }
585        if (a_total > 40) {
586         state = 8;
587         WristMove = "98";
588         tmove_prev = tmove;
589        }
590             break;
591
592      case (5): //Beta
593        if (WristPos == 2){
594         state = 6;
595         state_prev = 5;
596         tmove_prev = tmove;
597        }
598        if (WristPos == 3){
599         state = 7;
```

```
600         state_prev = 5;
601         tmove_prev = tmove;
602       }
603       if (WristPos == 1){
604         state = 2;
605         state_prev = 5;
606         tmove_prev = tmove;
607       }
608       if (WristPos == 5){
609         if ((ArmMove == "07" || ArmMove == "99") && IniState == 5){
610           WORKOUT.Exercise[3]++;
611           display.clearDisplay();
612           display.setTextSize(1.5);
613           display.setTextColor(WHITE);
614           display.setCursor(10, 1);
615           display.print("  Detecting... \n Deadlift\n");
616           display.display();
617 
618         }
619         state = 4;
620         state_prev = 5;
621         tmove_prev = tmove;
622       }
623       if (WristPos == 6){
624         state = 3;
625         state_prev = 5;
626         tmove_prev = tmove;
627       }
628       if (tmove - tmove_prev > 2000 && tmove - tmove_prev < 12000) {
629         WristMove = "05";
630         if (ExDisp == 1){
631           if (ArmMove == "74"){
632             display.clearDisplay();
633             display.setTextSize(1.5);
634             display.setTextColor(WHITE);
635             display.setCursor(10, 1);
636             display.print("  Detecting... \n Legs extension\n");
637             display.display();
```

```
638           }
639           if (ArmMove == "47"){
640             display.clearDisplay();
641             display.setTextSize(1.5);
642             display.setTextColor(WHITE);
643             display.setCursor(10, 1);
644             display.print("  Detecting... \n Curl Femoral\n");
645             display.display();
646           }
647         }
648        }
649        if (tmove - tmove_prev > 12000){
650         WristMove = "99";
651         if (ArmMove == "99"){
652           state = 8;
653         }
654        }
655        if (a_total > 40) {
656         state = 8;
657         WristMove = "98";
658         tmove_prev = tmove;
659        }
660            break;
661
662       case (6): //Delta
663        if (WristPos == 1){
664         state = 2;
665         state_prev = 6;
666         tmove_prev = tmove;
667        }
668        if (WristPos == 3){
669         state = 7;
670         state_prev = 6;
671         tmove_prev = tmove;
672        }
673        if (WristPos == 4){
674         state = 5;
675         state_prev = 6;
```

```
676         tmove_prev = tmove;
677        }
678        if (WristPos == 5){
679         state = 4;
680         state_prev = 6;
681         tmove_prev = tmove;
682        }
683        if (WristPos == 6){
684         state = 3;
685         state_prev = 6;
686         tmove_prev = tmove;
687        }
688        if (tmove - tmove_prev > 12000){
689         WristMove = "99";
690         if (ArmMove == "99"){
691           state = 8;
692         }
693        }
694        if (a_total > 40) {
695         state = 8;
696         WristMove = "98";
697         tmove_prev = tmove;
698        }
699             break;
700
701      case (7): //Dseta
702        if (WristPos == 2){
703         state = 6;
704         state_prev = 7;
705         tmove_prev = tmove;
706        }
707        if (WristPos == 1){
708         state = 2;
709         state_prev = 7;
710         tmove_prev = tmove;
711        }
712        if (WristPos == 4){
713         state = 5;
```

```
714         state_prev = 7;
715         tmove_prev = tmove;
716        }
717        if (WristPos == 5){
718         state = 4;
719         state_prev = 7;
720         tmove_prev = tmove;
721        }
722        if (WristPos == 6){
723         state = 3;
724         state_prev = 7;
725         tmove_prev = tmove;
726        }
727        if (tmove - tmove_prev > 2000 && tmove - tmove_prev < 12000) {
728         WristMove = "07";
729         if (ExDisp == 1){
730           display.clearDisplay();
731           display.setTextSize(1.5);
732           display.setTextColor(WHITE);
733           display.setCursor(10, 1);
734           display.print("  Detecting... \n Curl Femoral\n");
735           display.display();
736         }
737        }
738        if (tmove - tmove_prev > 12000){
739         WristMove = "99";
740         if (ArmMove == "99"){
741           state = 8;
742         }
743        }
744        if (a_total > 40) {
745         state = 8;
746         WristMove = "98";
747         tmove_prev = tmove;
748        }
749            break;
750
751      case (8): // Definition of moves
```

```
752         if (WORKOUT.Exercise[1] < 3) {
753           WORKOUT.Exercise[1] = 0;
754         }
755         if (WORKOUT.Exercise[3] < 3) {
756           WORKOUT.Exercise[1] = 0;
757         }
758         if (WORKOUT.Exercise[4] < 3) {
759           WORKOUT.Exercise[1] = 0;
760         }
761         if (WORKOUT.Exercise[1] > 3) {
762           WORKOUT.Series[j] = WORKOUT.Exercise[1];
763           WORKOUT.Ex[j] = "Squats (Legs)";
764           WORKOUT.Exercise[1] = 0;
765           WORKOUT.MuscGroup[5]++;
766         }
767         if (WORKOUT.Exercise[3] > 3) {
768           WORKOUT.Series[j] = WORKOUT.Exercise[3];
769           WORKOUT.Ex[j] = "Deadlift (Legs)";
770           WORKOUT.Exercise[3] = 0;
771           WORKOUT.MuscGroup[5]++;
772         }
773         if (WORKOUT.Exercise[4] > 3) {
774           WORKOUT.Series[j] = WORKOUT.Exercise[4];
775           WORKOUT.Ex[j] = "Push (Legs)";
776           WORKOUT.Exercise[4] = 0;
777           WORKOUT.MuscGroup[5]++;
778         }
779         env = true;
780         t_control = tmove;
781         state = 1;
782             break;
783       }
784
785       display.setTextSize(1.9);
786       display.setTextColor(WHITE);
787       display.print ("\nSerie ");
788       display.print (j-1);
789       display.print (" w/ ");
```

```
790     display.print (WORKOUT.Series[j-1]);
791     display.print (" reps\n");
792     display.println (WORKOUT.Ex[j-1]);
793     display.print ("WO time: ");
794     display.print (mins);
795     display.print (" mins");
796     display.print ("\n\n");
797     display.display();
798
799     Serial.print("valueW: ");
800     Serial.print(valueW.c_str());
801     Serial.print("\n");
802     Serial.print("valueA: ");
803     Serial.print(valueA.c_str());
804     Serial.print("\n");
805     }
806
807     if (workoutctrl == 2){
808       order = false;
809       WristMove = "96";
810       display.clearDisplay();
811       display.setTextSize(1.95);
812       display.setTextColor(WHITE);
813       display.setCursor(10, 1);
814       display.println ("  WORKOUT DONE!");
815       display.setTextSize(1.7);
816       display.println ("");
817       display.println ("Check your email, \nreview your workout  in the
              app, \nor start a new       workout");
818       display.display();
819       float tm = tmove;
820       WORKOUT.kcal = (tm / 3600000) * 450;
821       FinishWorkout();
822       WORKOUT.Series [30] = {0};
823       WORKOUT.MuscGroup[5] = {0};
824       WORKOUT.kcal = 0;
825       tmove = 0;
826       tmove_prev = 0;
```

```
827       t_control = 0;
828    }
829  });
830
831  Task taskMPU(TASK_SECOND / 5, TASK_FOREVER, []() {
832    if (workoutctrl == 1){
833      order = true;
834      WristMove = "1";
835      workoutctrl = 0;
836    }
837    if (order == true) {
838      if (tmove - t_prev > 1500){
839        t_prev = tmove;
840      }
841
842    float ax_m_s2, ay_m_s2, az_m_s2;
843    mpu.getAcceleration(&ax, &ay, &az);
844    mpu.getRotation(&gx, &gy, &gz);
845    dt = (tmove - t_prev) / 1000.0;
846    tmove = millis();
847    ax_m_s2 = ax * (9.81 / 16384.0);
848    ay_m_s2 = ay * (9.81 / 16384.0);
849    az_m_s2 = az * (9.81 / 16384.0);
850    accel_ang_x = atan(ax / sqrt(pow(ay, 2) + pow(az, 2))) * (180.0 /
         3.14);
851    accel_ang_y = atan(ay / sqrt(pow(ax, 2) + pow(az, 2))) * (180.0 /
         3.14);
852    accel_ang_z = atan(az / sqrt(pow(ax, 2) + pow(ay, 2))) * (180.0 /
         3.14);
853    ang_x = 0.98 *  (gx / 131) * dt + 0.02 * accel_ang_x;
854    ang_y = 0.98 *  (gy / 131) * dt + 0.02 * accel_ang_y;
855    ang_z = 0.98 *  (gz / 131) * dt + 0.02 * accel_ang_z;
856    double a_total = abs(ax_m_s2) + abs(ay_m_s2) + abs(az_m_s2);
857    mins = tmove/60000;
858    accel_av[0] = ax_av; accel_av[1] = ay_av; accel_av[2] = az_av;
859
860
861    if (0.6 > dt > 0.5){
```

```
862       ang_x_prev[it] = ang_x;
863       ang_y_prev[it] = ang_y;
864       ang_z_prev[it] = ang_z;
865       angx_av = (angx_av + ang_x_prev[it])/(it+1);
866       angy_av = (angy_av + ang_y_prev[it])/(it+1);
867       angz_av = (angz_av + ang_z_prev[it])/(it+1);
868       ax_m_s2_prev[it] = ax_m_s2;
869       ay_m_s2_prev[it] = ay_m_s2;
870       az_m_s2_prev[it] = az_m_s2;
871       ax_av = (ax_av + ax_m_s2_prev[it])/(it+1);
872       ay_av = (ay_av + ay_m_s2_prev[it])/(it+1);
873       az_av = (az_av + az_m_s2_prev[it])/(it+1);
874       it++;
875     }
876
877     else if (it > 10){
878       it = 0;
879     }
880
881     if (az_m_s2 > 7) {
882       WristPosition = "Alpha";
883       WristPos = 1;
884     }
885     else if (az_m_s2 < -7) {
886       WristPosition = "Delta";
887       WristPos = 2;
888     }
889     else if (ax_m_s2 > 7) {
890       WristPosition = "Beta";
891       WristPos = 4;
892     }
893     else if (ax_m_s2 < -7) {
894       WristPosition = "Dseta";
895       WristPos = 3;
896     }
897     else if (ay_m_s2 < -7) {
898       WristPosition = "Epsilon";
899       WristPos = 5;
```

```
900     }
901     else if (ay_m_s2 > 7) {
902       WristPosition = "Gamma";
903       WristPos = 6;
904     }
905     else {
906       WristPosition = "Half way";
907       WristPos = 0;
908     }
909 
910     switch (state) {
911       case (1): //State detection
912         if (ArmMove == "1" || header == "EX"){
913           IniMove = 0;
914           Push = 0;
915           Pull = 0;
916           WristMove = "1";
917         if (WristMove == "1" && ArmMove == "1"){
918           display.clearDisplay();
919           display.setTextSize(1.5);
920           display.setTextColor(WHITE);
921           display.setCursor(10, 1);
922           display.println("Waiting for a new        exercise");
923           display.display();
924         }
925         if (WristPosition == "Alpha") {
926           state = 2;
927           IniState = 2;
928           state_prev = 1;
929           tmove_prev = tmove;
930         }
931         if (WristPosition == "Gamma") {
932           IniState = 3;
933           state = 3;
934           state_prev = 1;
935           tmove_prev = tmove;
936         }
937         if (WristPosition == "Epsilon") {
```

```
938           state = 4;
939           IniState = 4;
940           state_prev = 1;
941           tmove_prev = tmove;
942         }
943         if (WristPosition == "Beta") {
944           state = 5;
945           IniState = 5;
946           state_prev = 1;
947           tmove_prev = tmove;
948         }
949         if (WristPosition == "Delta") {
950           state = 6;
951           IniState = 6;
952           state_prev = 1;
953           tmove_prev = tmove;
954         }
955         if (WristPosition == "Dseta") {
956           state = 7;
957           IniState = 7;
958           state_prev = 1;
959           tmove_prev = tmove;
960         }
961         else if (WORKOUT.Series[j] > 3) {
962           j++;
963           WristMove = "97";
964           Postures();
965         }
966         else if (header == "EX" && env == true) {
967           if (ArmEx == 10){
968             WORKOUT.Ex[j] = "Push-ups(triceps)";
969             WORKOUT.MuscGroup[1]++;
970           }
971           if (ArmEx == 11){
972             WORKOUT.Ex[j] = "Press(chest)";
973             WORKOUT.MuscGroup[2]++;
974           }
975           if (ArmEx == 15){
```

```
976             WORKOUT.Ex[j] = "Push-ups(chest)";
977             WORKOUT.MuscGroup[2]++;
978           }
979           if (ArmEx == 18){
980             WORKOUT.Ex[j] = "Pull-ups (back)";
981             WORKOUT.MuscGroup[3]++;
982           }
983           if (ArmEx == 19){
984             WORKOUT.Ex[j] = "Open pull (back)";
985             WORKOUT.MuscGroup[3]++;
986           }
987           if (ArmEx == 20){
988             WORKOUT.Ex[j] = "Close pull (back)";
989             WORKOUT.MuscGroup[3]++;
990           }
991           if (ArmEx == 21){
992             WORKOUT.Ex[j] = "Pull w/rope (shoulders)";
993             WORKOUT.MuscGroup[4]++;
994           }
995           if (ArmEx == 24){
996             WORKOUT.Ex[j] = "Press (shoulders)";
997             WORKOUT.MuscGroup[4]++;
998           }
999           if (ArmEx == 31){
1000            WORKOUT.Ex[j] = "Horizontal Scull (back)";
1001            WORKOUT.MuscGroup[3]++;
1002          }
1003          WORKOUT.Series[j] = ArmRep;
1004          j++;
1005          env = false;
1006        }
1007        }
1008            break;
1009
1010      case (2): //Alpha
1011        if (WristPos == 6) {
1012          if (IniState == 4 && state_prev == 3 && (ArmMove == "05" ||
                ArmMove == "99")) {
```

```
1013            WORKOUT.Exercise[3]++;
1014            display.clearDisplay();
1015            display.setTextSize(1.5);
1016            display.setTextColor(WHITE);
1017            display.setCursor(10, 1);
1018            display.print("  Detecting... \n Concentrated Curl\n");
1019            display.display();
1020          }
1021          if (IniState == 2 && (ArmMove == "04Q" || ArmMove == "99")) {
1022            WORKOUT.Exercise[1]++;
1023            display.clearDisplay();
1024            display.setTextSize(1.5);
1025            display.setTextColor(WHITE);
1026            display.setCursor(10, 1);
1027            display.print("  Detecting... \n    Simple curl\n");
1028            display.display();
1029          }
1030          if (IniState == 2 && (state_prev == 1 || state_prev == 3) &&
                 ArmMove == "42") {
1031            WORKOUT.Exercise[12]++;
1032            display.clearDisplay();
1033            display.setTextSize(1.5);
1034            display.setTextColor(WHITE);
1035            display.setCursor(10, 1);
1036            display.print("  Detecting... \n Opening on bench\n");
1037            display.display();
1038          }
1039            tmove_prev = tmove;
1040            state_prev = 2;
1041            WristMove = "23";
1042            state = 3;
1043        }
1044        if (WristPos == 5) {
1045         if (IniState == 3 && state_prev == 4 && (ArmMove == "05" ||
                ArmMove == "99")) {
1046            WORKOUT.Exercise[7]++;
1047            display.clearDisplay();
1048            display.setTextSize(1.5);
```

```
1049          display.setTextColor(WHITE);
1050          display.setCursor(10, 1);
1051          display.print("  Detecting... \n Inverse Extension\n");
1052          display.display();
1053        }
1054        tmove_prev = tmove;
1055        state_prev = 2;
1056        state = 4;
1057       }
1058       if (WristPos == 4 || WristPos == 3 || WristPos == 2 || a_total >
               30) {
1059          tmove_prev = tmove;
1060          state = 8;
1061          WristMove = "98";
1062       }
1063       if (tmove - tmove_prev > 12000){
1064          WristMove = "99";
1065          if (ArmMove == "99"){
1066            state = 1;
1067            t_control = tmove;
1068          }
1069       }
1070            break;
1071
1072      case (3): //Gamma
1073        if ((ay_m_s2_prev[it] - ay_av < -2) && ay_m_s2_prev[it] != 0 &&
               az_av > -2 && IniState == 3) {
1074          Push = 1;
1075          tmove_prev = tmove;
1076          if (IniMove == 0) {
1077            IniMove = 1;
1078          }
1079        }
1080        if ((ay_m_s2_prev[it] - ay_av > 2) && ay_m_s2_prev[it] != 0 &&
              az_av > -2 && IniState == 3) {
1081          Pull = 1;
1082          tmove_prev = tmove;
1083          if (IniMove == 0) {
```

```
1084            IniMove = 2;
1085          }
1086        }
1087        if (Push == 1 && Pull == 1 && IniMove == 1) {
1088          Push = 0;
1089          Pull = 0;
1090        }
1091        if (Push == 1 && Pull == 1 && IniMove == 2) {
1092          Push = 0;
1093          Pull = 0;
1094        }
1095
1096        if (WristPos == 1) {
1097          if (IniState == 3 && (ArmMove == "05" || ArmMove == "99")) {
1098            WORKOUT.Exercise[7]++;
1099            display.clearDisplay();
1100            display.setTextSize(1.5);
1101            display.setTextColor(WHITE);
1102            display.setCursor(10, 1);
1103            display.print("  Detecting... \n Inverse Extension\n");
1104            display.display();
1105          }
1106            state = 2;
1107            state_prev = 3;
1108            tmove_prev = tmove;
1109        }
1110        if (WristPos == 2) {
1111          tmove_prev = tmove;
1112          state_prev = 3;
1113          state = 6;
1114        }
1115        if (WristPos == 3) {
1116          tmove_prev = tmove;
1117          state_prev = 3;
1118          state = 7;
1119        }
1120        if (WristPos == 4) {
1121          tmove_prev = tmove;
```

```
1122          state_prev = 3;
1123          state = 5;
1124        }
1125        if (WristPos == 5) {
1126          if (IniState == 3) {
1127            if ((state_prev == 1 || state_prev == 6) && (ArmMove == "05
                  " || ArmMove == "99")) {
1128              WORKOUT.Exercise[6]++;
1129              display.clearDisplay();
1130              display.setTextSize(1.5);
1131              display.setTextColor(WHITE);
1132              display.setCursor(10, 1);
1133              display.print("  Detecting... \n Extension w/bar\n");
1134              display.display();
1135            }
1136            if ((state_prev == 1 || state_prev == 2) && (ArmMove == "05
                  " || ArmMove == "99")) {
1137              WORKOUT.Exercise[7]++;
1138              display.clearDisplay();
1139              display.setTextSize(1.5);
1140              display.setTextColor(WHITE);
1141              display.setCursor(10, 1);
1142              display.print("  Detecting... \n Inverse Extension\n");
1143              display.display();
1144            }
1145          }
1146          tmove_prev = tmove;
1147          state_prev = 3;
1148          state = 4;
1149        }
1150        if (tmove - tmove_prev > 2000 && tmove - tmove_prev < 12000) {
1151         if (abs(ax_m_s2_prev[it]) < 5 && ax_m_s2_prev[it] != 0 && az_av
              > -2) {
1152          WristMove = "03Q";
1153          if (ExDisp == 1){
1154            display.clearDisplay();
1155            display.setTextSize(1.5);
1156            display.setTextColor(WHITE);
```

```
1157            display.setCursor(10, 1);
1158            display.print("  Detecting... \n Pull-ups\n");
1159            display.display();
1160          }
1161        }
1162        if (abs(ax_m_s2_prev[it]) > 5 && az_av > -2) {
1163          WristMove = "03A";
1164          if (ExDisp == 1){
1165            if (ArmMove == "42"){
1166              if(WORKOUT.MuscGroup[2] > WORKOUT.MuscGroup[3] && WORKOUT
                     .MuscGroup[2] > WORKOUT.MuscGroup[4]){
1167                display.clearDisplay();
1168                display.setTextSize(1.5);
1169                display.setTextColor(WHITE);
1170                display.setCursor(10, 1);
1171                display.print("  Detecting... \n Press bench\n");
1172                display.display();
1173              }
1174              if (WORKOUT.MuscGroup[3] > WORKOUT.MuscGroup[2] &&
                     WORKOUT.MuscGroup[3] > WORKOUT.MuscGroup[4]){
1175                display.clearDisplay();
1176                display.setTextSize(1.5);
1177                display.setTextColor(WHITE);
1178                display.setCursor(10, 1);
1179                display.print("  Detecting... \n Open pull\n");
1180                display.display();
1181              }
1182            }
1183            if (ArmMove == "52"){
1184              display.clearDisplay();
1185              display.setTextSize(1.5);
1186              display.setTextColor(WHITE);
1187              display.setCursor(10, 1);
1188              display.print("  Detecting... \n Close pull\n");
1189              display.display();
1190            }
1191          }
1192        }
```

```
1193        }
1194
1195        if (tmove - tmove_prev > 12000){
1196         WristMove = "99";
1197         if (ArmMove == "99"){
1198           state = 8;
1199           t_control = tmove;
1200         }
1201        }
1202        if (a_total > 40) {
1203         state = 8;
1204         WristMove = "98";
1205         tmove_prev = tmove;
1206        }
1207             break;
1208
1209       case (4): // psilon
1210         if (WristPos == 4) {
1211           if (IniState == 4 && (ArmMove == "05" || ArmMove == "99")) {
1212             WORKOUT.Exercise[2]++;
1213             display.clearDisplay();
1214             display.setTextSize(1.5);
1215             display.setTextColor(WHITE);
1216             display.setCursor(10, 1);
1217             display.print("  Detecting... \n   Hammer\n");
1218             display.display();
1219           }
1220           if (IniState == 4  && (ArmMove == "54" || ArmMove == "45")) {
1221             WORKOUT.Exercise[27]++;
1222             display.clearDisplay();
1223             display.setTextSize(1.5);
1224             display.setTextColor(WHITE);
1225             display.setCursor(10, 1);
1226             display.print("  Detecting... \n Vertical lifting\n");
1227             display.display();
1228           }
1229             tmove_prev = tmove;
1230             state_prev = 4;
```

```
1231            state = 5;
1232          }
1233          if (WristPos == 2) {
1234            if (IniState == 4 && ArmMove == "52") {
1235              WORKOUT.Exercise[17]++;
1236              display.clearDisplay();
1237              display.setTextSize(1.5);
1238              display.setTextColor(WHITE);
1239              display.setCursor(10, 1);
1240              display.print("  Detecting... \n Scull to neck\n");
1241              display.display();
1242            }
1243            if (IniState == 4 && ArmMove == "53") {
1244              WORKOUT.Exercise[28]++;
1245              display.clearDisplay();
1246              display.setTextSize(1.5);
1247              display.setTextColor(WHITE);
1248              display.setCursor(10, 1);
1249              display.print("  Detecting... \n Flight\n");
1250              display.display();
1251            }
1252            if (IniState == 4 && (ArmMove == "05" || ArmMove == "99")) {
1253              WORKOUT.Exercise[4]++;
1254              display.clearDisplay();
1255              display.setTextSize(1.5);
1256              display.setTextColor(WHITE);
1257              display.setCursor(10, 1);
1258              display.print("  Detecting... \n Inverse Curl\n");
1259              display.display();
1260            }
1261            if (IniState == 4 && ArmMove == "56") {
1262              WORKOUT.Exercise[30]++;
1263              display.clearDisplay();
1264              display.setTextSize(1.5);
1265              display.setTextColor(WHITE);
1266              display.setCursor(10, 1);
1267              display.print("  Detecting... \n Diamond Push-ups\n");
1268              display.display();
```

```
1269          }
1270          tmove_prev = tmove;
1271          state_prev = 4;
1272          state = 6;
1273        }
1274        if (WristPos == 1) {
1275          if (IniState == 4 && (state_prev == 1 || state_prev == 3 ||
                 state_prev == 2) && (ArmMove == "05" || ArmMove == "99"))
                 {
1276            WORKOUT.Exercise[3]++;
1277            display.clearDisplay();
1278            display.setTextSize(1.5);
1279            display.setTextColor(WHITE);
1280            display.setCursor(10, 1);
1281            display.print("  Detecting... \n Concentrated Curl\n");
1282            display.display();
1283          }
1284          if (IniState == 4 && (state_prev == 1 || state_prev == 2) &&
                 (ArmMove == "54" || ArmMove == "45")) {
1285            WORKOUT.Exercise[16]++;
1286            display.clearDisplay();
1287            display.setTextSize(1.5);
1288            display.setTextColor(WHITE);
1289            display.setCursor(10, 1);
1290            display.print("  Detecting... \n      Lifting");
1291            display.display();
1292          }
1293          tmove_prev = tmove;
1294          state_prev = 4;
1295          state = 2;
1296        }
1297        if (WristPos == 6) {
1298          if (WORKOUT.Exercise[3] > 1) {
1299            WORKOUT.Exercise[3]++;
1300            display.clearDisplay();
1301            display.setTextSize(1.5);
1302            display.setTextColor(WHITE);
1303            display.setCursor(10, 1);
```

```
1304            display.print("  Detecting... \n Concentrated Curl\n");
1305            display.display();
1306          }
1307          if (WORKOUT.Exercise[4] > 1) {
1308            WORKOUT.Exercise[4]++;
1309            display.clearDisplay();
1310            display.setTextSize(1.5);
1311            display.setTextColor(WHITE);
1312            display.setCursor(10, 1);
1313            display.print("  Detecting... \n Inverse Curl\n");
1314            display.display();
1315          }
1316          tmove_prev = tmove;
1317          state_prev = 4;
1318          state = 3;
1319        }
1320        if (tmove - tmove_prev > 2000 && tmove - tmove_prev < 12000) {
1321          if ((ay_m_s2_prev[it] > 2.5) && ay_m_s2_prev[it] != 0 &&
                az_av > -2){
1322            WristMove = "04A";
1323            if (ExDisp == 1){
1324              display.clearDisplay();
1325              display.setTextSize(1.5);
1326              display.setTextColor(WHITE);
1327              display.setCursor(10, 1);
1328              display.print("  Detecting... \n Horizontal Scull\n");
1329              display.display();
1330            }
1331          }
1332          if ((ay_m_s2_prev[it] < 2.5) && ay_m_s2_prev[it] != 0 &&
                az_av > -2){
1333            WristMove = "04Q";
1334            if (ExDisp == 1){
1335              display.clearDisplay();
1336              display.setTextSize(1.5);
1337              display.setTextColor(WHITE);
1338              display.setCursor(10, 1);
1339              display.print("  Detecting... \n Push-ups\n");
```

```
1340              display.display();
1341            }
1342          }
1343        }
1344        if (WristPos == 3 || a_total > 40) {
1345          tmove_prev = tmove;
1346          WristMove = "98";
1347          state = 8;
1348        }
1349        if (tmove - tmove_prev > 12000){
1350          WristMove = "99";
1351          if (ArmMove == "99"){
1352            state = 8;
1353            t_control = tmove;
1354          }
1355        }
1356            break;
1357
1358      case (5): //Beta
1359        if (WristPos == 5) {
1360          if (IniState == 3 && (ArmMove == "45" || ArmMove == "54")) {
1361            WORKOUT.Exercise[5]++;
1362            display.clearDisplay();
1363            display.setTextSize(1.5);
1364            display.setTextColor(WHITE);
1365            display.setCursor(10, 1);
1366            display.print("  Detecting... \n Rope Extension\n");
1367            display.display();
1368          }
1369          tmove_prev = tmove;
1370          state_prev = 5;
1371          state = 4;
1372        }
1373        if (WristPos == 6) {
1374          tmove_prev = tmove;
1375          state_prev = 5;
1376          state = 3;
1377        }
```

```
1378        if (((ay_m_s2_prev[it] - ay_av) > 1.5) && ay_m_s2_prev[it] != 0
               && ax_av > -1 && az_av > -1 && IniState == 5) {
1379         Push = 1;
1380         tmove_prev = tmove;
1381         if (IniMove == 0) {
1382           IniMove = 1;
1383         }
1384        }
1385        if (((ay_m_s2_prev[it] - ay_av) < -1.5) && ay_m_s2_prev[it] != 0
                && ax_av > -1 && az_av > -1 && IniState == 5) {
1386         Pull = 1;
1387         tmove_prev = tmove;
1388         if (IniMove == 0) {
1389           IniMove = 2;
1390         }
1391        }
1392        if (((az_m_s2_prev[it] - az_av) > 1.5) && az_m_s2_prev[it] != 0
               && ay_av > 5 && ax_av > -1 && IniState == 5) {
1393         Push = 1;
1394         tmove_prev = tmove;
1395         if (IniMove == 0) {
1396           IniMove = 1;
1397         }
1398        }
1399        if (((az_m_s2_prev[it] - az_av) < -1.5) && az_m_s2_prev[it] != 0
                && ay_av > 5 && ax_av > -1 && IniState == 5) {
1400         Pull = 1;
1401         tmove_prev = tmove;
1402         if (IniMove == 0) {
1403           IniMove = 2;
1404         }
1405        }
1406        if (Push == 1 && Pull == 1 && IniMove == 1) {
1407         if (ArmMove == "03" || (ArmMove == "99" && WORKOUT.Exercise[13]
                > 1)) {
1408           WORKOUT.Exercise[13]++;
1409           display.clearDisplay();
1410           display.setTextSize(1.5);
```

```
1411          display.setTextColor(WHITE);
1412          display.setCursor(10, 1);
1413          display.print("  Detecting... \n Opening on chair\n");
1414          display.display();
1415        }
1416        if (ArmMove == "410"  || (ArmMove == "99" && WORKOUT.Exercise
                [22] > 1)) {
1417          WORKOUT.Exercise[22]++;
1418          display.clearDisplay();
1419          display.setTextSize(1.5);
1420          display.setTextColor(WHITE);
1421          display.setCursor(10, 1);
1422          display.print("  Detecting... \n Scull\n");
1423          display.display();
1424        }
1425        Push = 0;
1426        Pull = 0;
1427        state_prev = 5;
1428        state = 5;
1429       }
1430       if (WristPos == 1 || WristPos == 3 || WristPos == 2 || a_total >
             40) {
1431        tmove_prev = tmove;
1432        WristMove = "98";
1433        state = 8;
1434       }
1435       else if (tmove - tmove_prev > 2000 && tmove - tmove_prev <
             12000) {
1436        WristMove = "05";
1437       }
1438       if (tmove - tmove_prev > 12000){
1439        WristMove = "99";
1440        if (ArmMove == "99"){
1441          state = 8;
1442        }
1443       }
1444            break;
1445
```

```
1446      case (6): //Delta
1447        if (WristPos == 5) {
1448          if (IniState == 3 && state_prev == 3 && (ArmMove == "05" ||
                ArmMove == "99")) {
1449            WORKOUT.Exercise[6]++;
1450            display.clearDisplay();
1451            display.setTextSize(1.5);
1452            display.setTextColor(WHITE);
1453            display.setCursor(10, 1);
1454            display.print("  Detecting... \n Extension w/bar\n");
1455            display.display();
1456          }
1457          if (IniState == 6 && (ArmMove == "35" || ArmMove == "53")) {
1458            if (WORKOUT.MuscGroup[3] > WORKOUT.MuscGroup[2]){
1459              WORKOUT.Exercise[32]++;
1460              display.clearDisplay();
1461              display.setTextSize(1.5);
1462              display.setTextColor(WHITE);
1463              display.setCursor(10, 1);
1464              display.print("  Detecting... \n Pull-over\n");
1465              display.display();
1466            }
1467            else{
1468              WORKOUT.Exercise[33]++;
1469              display.clearDisplay();
1470              display.setTextSize(1.5);
1471              display.setTextColor(WHITE);
1472              display.setCursor(10, 1);
1473              display.print("  Detecting... \n Opening Down\n");
1474              display.display();
1475            }
1476          }
1477          tmove_prev = tmove;
1478          state_prev = 6;
1479          state = 4;
1480        }
1481        if (WristPos == 6) {
1482          if (IniState == 4 && (ArmMove == "05" || ArmMove == "99")) {
```

```
1483          WORKOUT.Exercise[4]++;
1484          display.clearDisplay();
1485          display.setTextSize(1.5);
1486          display.setTextColor(WHITE);
1487          display.setCursor(10, 1);
1488          display.print("  Detecting... \n Inverse Curl\n");
1489          display.display();
1490        }
1491        tmove_prev = tmove;
1492        state_prev = 6;
1493        state = 3;
1494      }
1495      if (WristPos == 1 || WristPos == 3 || WristPos == 4 || a_total >
            30){
1496        tmove_prev = tmove;
1497        WristMove = "98";
1498        state = 8;
1499      }
1500      if (tmove - tmove_prev > 12000){
1501        WristMove == "99";
1502        if (ArmMove == "99"){
1503          state = 1;
1504        }
1505      }
1506      if ((ax_m_s2_prev[it] - ax_av < -1.5) && az_av > 0 && IniState
           == 6) {
1507        Push = 1;
1508        if (IniMove == 0) {
1509          IniMove = 1;
1510        }
1511        tmove_prev = tmove;
1512      }
1513      if ((ax_m_s2_prev[it] - ax_av > 1.5) && az_av > 0 && IniState ==
            6) {
1514        Pull = 1;
1515        if (IniMove == 0) {
1516          IniMove = 2;
1517        }
```

```
1518        tmove_prev = tmove;
1519       }
1520       if ((az_m_s2_prev[it] - az_av > 1.5) && ax_av > 0 && IniState ==
               6) {
1521        Pull = 1;
1522        if (IniMove == 0) {
1523          IniMove = 3;
1524        }
1525        tmove_prev = tmove;
1526       }
1527       if ((az_m_s2_prev[it] - az_av < -1.5) && ax_av > 0 && IniState
              == 6) {
1528        Push = 1;
1529        if (IniMove == 0) {
1530          IniMove = 4;
1531        }
1532        tmove_prev = tmove;
1533       }
1534       if (Push == 1 && Pull == 1){
1535        if (ArmMove == "03" || ArmMove == "99"){
1536          if (IniMove == 1 || WORKOUT.Exercise[14] > 2) {
1537            WORKOUT.Exercise[14]++;
1538            display.clearDisplay();
1539            display.setTextSize(1.5);
1540            display.setTextColor(WHITE);
1541            display.setCursor(10, 1);
1542            display.print("  Detecting... \n Push\n");
1543            display.display();
1544            state = 3;
1545          }
1546          if (IniMove == 3 || WORKOUT.Exercise[25] > 2) {
1547            WORKOUT.Exercise[25]++;
1548            display.clearDisplay();
1549            display.setTextSize(1.5);
1550            display.setTextColor(WHITE);
1551            display.setCursor(10, 1);
1552            display.print("  Detecting... \n Opening\n");
1553            display.display();
```

```
1554            state = 3;
1555          }
1556        }
1557        state = 3;
1558       }
1559       else if (tmove - tmove_prev > 2000 && tmove - tmove_prev <
             12000) {
1560        WristMove = "06";
1561          if (ExDisp == 1){
1562            display.clearDisplay();
1563            display.setTextSize(1.5);
1564            display.setTextColor(WHITE);
1565            display.setCursor(10, 1);
1566            display.print("  Detecting... \n Pull w/rope\n");
1567            display.display();
1568          }
1569       }
1570            break;
1571
1572      case (7): //Dseta
1573        if (WristPos == 6) {
1574          if (IniState == 7 && (ArmMove == "02" || ArmMove == "99")) {
1575            WORKOUT.Exercise[29]++;
1576            display.clearDisplay();
1577            display.setTextSize(1.5);
1578            display.setTextColor(WHITE);
1579            display.setCursor(10, 1);
1580            display.print("  Detecting... \n Vertical Extension\n");
1581            display.display();
1582          }
1583          tmove_prev = tmove;
1584          state_prev = 7;
1585          state = 3;
1586        }
1587        if (WristPos == 4 || WristPos == 1 || WristPos == 5 || WristPos
               == 2 || a_total > 40) {
1588          tmove_prev = tmove;
1589          WristMove = "98";
```

```
1590          state = 8;
1591        }
1592        if (tmove - tmove_prev > 12000){
1593          WristMove = "99";
1594          if (ArmMove == "99"){
1595            state = 1;
1596          }
1597        }
1598            break;
1599
1600      case (8): // Definition of moves
1601        if (WORKOUT.Exercise[1] < 3) {
1602          WORKOUT.Exercise[1] = 0;
1603        }
1604        if (WORKOUT.Exercise[2] < 3) {
1605          WORKOUT.Exercise[2] = 0;
1606        }
1607        if (WORKOUT.Exercise[3] < 3) {
1608          WORKOUT.Exercise[3] = 0;
1609        }
1610        if (WORKOUT.Exercise[4] < 3) {
1611          WORKOUT.Exercise[4] = 0;
1612        }
1613        if (WORKOUT.Exercise[5] < 3) {
1614          WORKOUT.Exercise[5] = 0;
1615        }
1616        if (WORKOUT.Exercise[6] < 3) {
1617          WORKOUT.Exercise[6] = 0;
1618        }
1619        if (WORKOUT.Exercise[7] < 3) {
1620          WORKOUT.Exercise[7] = 0;
1621        }
1622        if (WORKOUT.Exercise[12] < 3) {
1623          WORKOUT.Exercise[12] = 0;
1624        }
1625        if (WORKOUT.Exercise[13] < 3) {
1626          WORKOUT.Exercise[13] = 0;
1627        }
```

```
1628        if (WORKOUT.Exercise[14] < 3) {
1629          WORKOUT.Exercise[14] = 0;
1630        }
1631        if (WORKOUT.Exercise[16] < 3) {
1632          WORKOUT.Exercise[16] = 0;
1633        }
1634        if (WORKOUT.Exercise[17] < 3) {
1635          WORKOUT.Exercise[17] = 0;
1636        }
1637        if (WORKOUT.Exercise[19] < 3) {
1638          WORKOUT.Exercise[19] = 0;
1639        }
1640        if (WORKOUT.Exercise[22] < 3) {
1641          WORKOUT.Exercise[22] = 0;
1642        }
1643        if (WORKOUT.Exercise[23] < 3) {
1644          WORKOUT.Exercise[23] = 0;
1645        }
1646        if (WORKOUT.Exercise[25] < 3) {
1647          WORKOUT.Exercise[25] = 0;
1648        }
1649        if (WORKOUT.Exercise[28] < 3) {
1650          WORKOUT.Exercise[28] = 0;
1651        }
1652        if (WORKOUT.Exercise[27] < 3) {
1653          WORKOUT.Exercise[27] = 0;
1654        }
1655        if (WORKOUT.Exercise[29] < 3) {
1656          WORKOUT.Exercise[29] = 0;
1657        }
1658        if (WORKOUT.Exercise[30] < 3) {
1659          WORKOUT.Exercise[30] = 0;
1660        }
1661        if (WORKOUT.Exercise[32] < 3) {
1662          WORKOUT.Exercise[32] = 0;
1663        }
1664        if (WORKOUT.Exercise[33] < 3) {
1665          WORKOUT.Exercise[33] = 0;
```

```
1666        }
1667        if (WORKOUT.Exercise[1] > 3) {
1668          WORKOUT.Series [j] = WORKOUT.Exercise[1];
1669          WORKOUT.Ex[j] = "Simple Curl (biceps)";
1670          WORKOUT.Exercise[1] = 0;
1671          WORKOUT.MuscGroup[0]++;
1672          accel_id[0] = 0;
1673          accel_id[1] = 0;
1674          accel_id[2] = 9;
1675          accel_id[3] = 0;
1676          accel_id[4] = 9;
1677          accel_id[5] = 0;
1678        }
1679        if (WORKOUT.Exercise[2] > 3) {
1680          WORKOUT.Series[j] = WORKOUT.Exercise[2];
1681          WORKOUT.Exercise[2] = 0;
1682          WORKOUT.Ex[j] = "Hammer (biceps)";
1683          WORKOUT.MuscGroup[0]++;
1684          accel_id[0] = 0;
1685          accel_id[1] = -9;
1686          accel_id[2] = 0;
1687          accel_id[3] = 9;
1688          accel_id[4] = 0;
1689          accel_id[5] = 0;
1690        }
1691        if (WORKOUT.Exercise[3] > 3) {
1692          WORKOUT.Series [j] = WORKOUT.Exercise[3];
1693          WORKOUT.Ex[j] = "Concentrated Curl (biceps)";
1694          WORKOUT.Exercise[3] = 0;
1695          WORKOUT.MuscGroup[0]++;
1696          accel_id[0] = 0;
1697          accel_id[1] = -9;
1698          accel_id[2] = 0;
1699          accel_id[3] = 0;
1700          accel_id[4] = 0;
1701          accel_id[5] = 9;
1702        }
1703        if (WORKOUT.Exercise[4] > 3) {
```

```
1704          WORKOUT.Series[j] = WORKOUT.Exercise[4];
1705          WORKOUT.Exercise[4] = 0;
1706          WORKOUT.Ex[j] = "Inverse Curl (biceps)";
1707          WORKOUT.MuscGroup[0]++;
1708          accel_id[0] = 0;
1709          accel_id[1] = -9;
1710          accel_id[2] = 0;
1711          accel_id[3] = 0;
1712          accel_id[4] = 9;
1713          accel_id[5] = 0;
1714        }
1715        if (WORKOUT.Exercise[5] > 3) {
1716          WORKOUT.Series[j] = WORKOUT.Exercise[5];
1717          WORKOUT.Exercise[5] = 0;
1718          WORKOUT.Ex[j] = "Rope Extension (triceps)";
1719          WORKOUT.MuscGroup[1]++;
1720          accel_id[0] = 0;
1721          accel_id[1] = 9;
1722          accel_id[2] = 0;
1723          accel_id[3] = 0;
1724          accel_id[4] = -9;
1725          accel_id[5] = 0;
1726        }
1727        if (WORKOUT.Exercise[6] > 3) {
1728          WORKOUT.Series[j] = WORKOUT.Exercise[6];
1729          WORKOUT.Exercise[6] = 0;
1730          WORKOUT.Ex[j] = "Extension w/bar (triceps)";
1731          WORKOUT.MuscGroup[1]++;
1732          accel_id[0] = 0;
1733          accel_id[1] = 9;
1734          accel_id[2] = 0;
1735          accel_id[3] = 0;
1736          accel_id[4] = -9;
1737          accel_id[5] = 0;
1738        }
1739        if (WORKOUT.Exercise[7] > 3) {
1740          WORKOUT.Series[j] = WORKOUT.Exercise[7];
1741          WORKOUT.Exercise[7] = 0;
```

```
1742          WORKOUT.Ex[j] = "Inverse Extension (triceps)";
1743          WORKOUT.MuscGroup[1]++;
1744          accel_id[0] = 0;
1745          accel_id[1] = 9;
1746          accel_id[2] = 0;
1747          accel_id[3] = 0;
1748          accel_id[4] = -9;
1749          accel_id[5] = 0;
1750        }
1751        if (WORKOUT.Exercise[12] > 3) {
1752          WORKOUT.Series[j] = WORKOUT.Exercise[12];
1753          WORKOUT.Exercise[12] = 0;
1754          WORKOUT.Ex[j] = "Opening on bench (chest)";
1755          WORKOUT.MuscGroup[2]++;
1756          accel_id[0] = 0;
1757          accel_id[1] = 0;
1758          accel_id[2] = 9;
1759          accel_id[3] = 0;
1760          accel_id[4] = 9;
1761          accel_id[5] = 0;
1762        }
1763        if (WORKOUT.Exercise[13] > 3) {
1764          WORKOUT.Series[j] = WORKOUT.Exercise[13];
1765          WORKOUT.Exercise[13] = 0;
1766          WORKOUT.Ex[j] = "Opening on chair (chest)";
1767          WORKOUT.MuscGroup[2]++;
1768          accel_id[0] = 9;
1769          accel_id[1] = 0;
1770          accel_id[2] = 0;
1771          accel_id[3] = 9;
1772          accel_id[4] = 0;
1773          accel_id[5] = 0;
1774        }
1775        if (WORKOUT.Exercise[14] > 3) {
1776          WORKOUT.Series[j] = WORKOUT.Exercise[14];
1777          WORKOUT.Exercise[14] = 0;
1778          WORKOUT.Ex[j] = "Push (chest)";
1779          WORKOUT.MuscGroup[2]++;
```

```
1780          accel_id[0] = 0;
1781          accel_id[1] = 0;
1782          accel_id[2] = -9;
1783          accel_id[3] = 0;
1784          accel_id[4] = 0;
1785          accel_id[5] = -9;
1786        }
1787        if (WORKOUT.Exercise[16] > 3) {
1788          WORKOUT.Series[j] = WORKOUT.Exercise[16];
1789          WORKOUT.Exercise[16] = 0;
1790          WORKOUT.Ex[j] = "Lifting (chest)";
1791          WORKOUT.MuscGroup[2]++;
1792          accel_id[0] = 0;
1793          accel_id[1] = -9;
1794          accel_id[2] = 0;
1795          accel_id[3] = 0;
1796          accel_id[4] = 0;
1797          accel_id[5] = 9;
1798        }
1799        if (WORKOUT.Exercise[17] > 3) {
1800          WORKOUT.Series[j] = WORKOUT.Exercise[17];
1801          WORKOUT.Exercise[17] = 0;
1802          WORKOUT.Ex[j] = "Scull to neck (chest)";
1803          WORKOUT.MuscGroup[2]++;
1804          accel_id[0] = 0;
1805          accel_id[1] = -9;
1806          accel_id[2] = 0;
1807          accel_id[3] = 0;
1808          accel_id[4] = 0;
1809          accel_id[5] = -9;
1810        }
1811        if (WORKOUT.Exercise[19] > 3) {
1812          WORKOUT.Series[j] = WORKOUT.Exercise[19];
1813          WORKOUT.Exercise[18] = 0;
1814          WORKOUT.Ex[j] = "Pull (back)";
1815          WORKOUT.MuscGroup[3]++;
1816          accel_id[0] = 0;
1817          accel_id[1] = 9;
```

```
1818          accel_id[2] = 0;
1819          accel_id[3] = 0;
1820          accel_id[4] = 9;
1821          accel_id[5] = 0;
1822        }
1823        if (WORKOUT.Exercise[22] > 3) {
1824          WORKOUT.Series[j] = WORKOUT.Exercise[22];
1825          WORKOUT.Exercise[22] = 0;
1826          WORKOUT.Ex[j] = "Scull (back)";
1827          WORKOUT.MuscGroup[3]++;
1828          accel_id[0] = 9;
1829          accel_id[1] = 0;
1830          accel_id[2] = 0;
1831          accel_id[3] = 9;
1832          accel_id[4] = 0;
1833          accel_id[5] = 0;
1834        }
1835        if (WORKOUT.Exercise[23] > 3) {
1836          WORKOUT.Series[j] = WORKOUT.Exercise[23];
1837          WORKOUT.Exercise[23] = 0;
1838          WORKOUT.Ex[j] = "Horizontal Low Scull (back)";
1839          WORKOUT.MuscGroup[3]++;
1840          accel_id[0] = 0;
1841          accel_id[1] = 0;
1842          accel_id[2] = 9;
1843          accel_id[3] = 0;
1844          accel_id[4] = 9;
1845          accel_id[5] = 0;
1846        }
1847        if (WORKOUT.Exercise[25] > 3) {
1848          WORKOUT.Series[j] = WORKOUT.Exercise[25];
1849          WORKOUT.Exercise[25] = 0;
1850          WORKOUT.Ex[j] = "Opening (shoulders)";
1851          WORKOUT.MuscGroup[3]++;
1852          accel_id[0] = 0;
1853          accel_id[1] = 0;
1854          accel_id[2] = -9;
1855          accel_id[3] = 0;
```

```
1856          accel_id[4] = 0;
1857          accel_id[5] = -9;
1858        }
1859        if (WORKOUT.Exercise[28] > 3) {
1860          WORKOUT.Series[j] = WORKOUT.Exercise[28];
1861          WORKOUT.Exercise[28] = 0;
1862          WORKOUT.Ex[j] = "Flight (shoulders)";
1863          WORKOUT.MuscGroup[4]++;
1864          accel_id[0] = 0;
1865          accel_id[1] = -9;
1866          accel_id[2] = 0;
1867          accel_id[3] = 0;
1868          accel_id[4] = 0;
1869          accel_id[5] = -9;
1870        }
1871        if (WORKOUT.Exercise[27] > 3) {
1872          WORKOUT.Series[j] = WORKOUT.Exercise[27];
1873          WORKOUT.Exercise[27] = 0;
1874          WORKOUT.Ex[j] = "Vertical Lifting (shoulders)";
1875          WORKOUT.MuscGroup[4]++;
1876          accel_id[0] = 0;
1877          accel_id[1] = -9;
1878          accel_id[2] = 0;
1879          accel_id[3] = 9;
1880          accel_id[4] = 0;
1881          accel_id[5] = 0;
1882        }
1883        if (WORKOUT.Exercise[29] > 3) {
1884          WORKOUT.Series[j] = WORKOUT.Exercise[29];
1885          WORKOUT.Exercise[29] = 0;
1886          WORKOUT.Ex[j] = "Vertical Extensions (triceps)";
1887          WORKOUT.MuscGroup[1]++;
1888          accel_id[0] = -9;
1889          accel_id[1] = 0;
1890          accel_id[2] = 0;
1891          accel_id[3] = 0;
1892          accel_id[4] = 9;
1893          accel_id[5] = 0;
```

```
1894        }
1895        if (WORKOUT.Exercise[30] > 3) {
1896          WORKOUT.Series[j] = WORKOUT.Exercise[30];
1897          WORKOUT.Exercise[30] = 0;
1898          WORKOUT.Ex[j] = "Diamond Push-ups (triceps)";
1899          WORKOUT.MuscGroup[1]++;
1900          accel_id[0] = 0;
1901          accel_id[1] = -9;
1902          accel_id[2] = 0;
1903          accel_id[3] = 0;
1904          accel_id[4] = 0;
1905          accel_id[5] = -9;
1906        }
1907        if (WORKOUT.Exercise[32] > 3) {
1908          WORKOUT.Series[j] = WORKOUT.Exercise[32];
1909          WORKOUT.Exercise[32] = 0;
1910          WORKOUT.Ex[j] = "Pull-over (back)";
1911          WORKOUT.MuscGroup[3]++;
1912          accel_id[0] = 0;
1913          accel_id[1] = 0;
1914          accel_id[2] = -9;
1915          accel_id[3] = 0;
1916          accel_id[4] = -9;
1917          accel_id[5] = 0;
1918        }
1919        if (WORKOUT.Exercise[33] > 3) {
1920          WORKOUT.Series[j] = WORKOUT.Exercise[33];
1921          WORKOUT.Exercise[33] = 0;
1922          WORKOUT.Ex[j] = "Opening down (back)";
1923          WORKOUT.MuscGroup[3]++;
1924          accel_id[0] = 0;
1925          accel_id[1] = 0;
1926          accel_id[2] = -9;
1927          accel_id[3] = 0;
1928          accel_id[4] = -9;
1929          accel_id[5] = 0;
1930        }
1931        display.clearDisplay();
```

```
1932        display.setTextSize(1.5);
1933        display.setTextColor(WHITE);
1934        display.setCursor(10, 1);
1935        display.println("   Registering reps");
1936        display.display();
1937        env = true;
1938        t_control = tmove;
1939        state = 1;
1940            break;
1941    }
1942
1943    display.setTextSize(1.9);
1944    display.setTextColor(WHITE);
1945    display.print ("\nSerie ");
1946    display.print (j-1);
1947    display.print (" w/ ");
1948    display.print (WORKOUT.Series[j-1]);
1949    display.print (" reps\n");
1950    display.println (WORKOUT.Ex[j-1]);
1951    display.print("\n");
1952    display.print ("WO time: ");
1953    display.print (mins);
1954    display.print (" mins");
1955    display.print("\n");
1956    display.display();
1957
1958    Serial.print("valueW: ");
1959    Serial.print(valueW.c_str());
1960    Serial.print("\n");
1961    Serial.print("valueA: ");
1962    Serial.print(valueA.c_str());
1963    Serial.print("\n");
1964    }
1965
1966    if (workoutctrl == 2){
1967      order = false;
1968      WristMove = "96";
1969      float tm = tmove;
```

```
1970      WORKOUT.kcal = round((tm / 3600000) * 450);
1971      Serial.print("tmove: ");
1972      Serial.print(tmove);
1973      Serial.print("\n");
1974      Serial.print("KCAL: ");
1975      Serial.print(WORKOUT.kcal);
1976      Serial.print("\n");
1977
1978      display.clearDisplay();
1979      display.setTextSize(1.95);
1980      display.setTextColor(WHITE);
1981      display.setCursor(10, 1);
1982      display.println ("  WORKOUT DONE!");
1983      display.setTextSize(1.7);
1984      display.println ("");
1985      display.println ("Check your email, \nreview your workout  in the
              app, \nor start a new       workout");
1986      display.display();
1987      FinishWorkout();
1988      FinishedTime = millis()-tmove;
1989      tmove = 0;
1990      tmove_prev = 0;
1991      t_control = 0;
1992      WORKOUT.Series [30] = {0};
1993      WORKOUT.MuscGroup[5] = {0};
1994      WORKOUT.kcal = 0;
1995      workoutctrl = 0;
1996    }
1997  });
1998
1999  Task taskBLE(TASK_SECOND / 3, TASK_FOREVER, []() {
2000    if(pServer->getConnectedCount()) {
2001      NimBLEService* pSvc = pServer->getServiceByUUID((uint16_t)0x1826)
             ;
2002      NimBLECharacteristic* pArm = pSvc->getCharacteristic(
             CHARACTERISTIC_UUID_ARM);
2003      valueW = "WM_"+ WristMove;
2004      pWristCharacteristic->setValue(valueW);
```

```
2005      valueA = pArmCharacteristic->getValue();
2006      header = valueA.substr(0, 2);//Decoding of messages
2007        if (header == "AM"){
2008          ArmMove = s.separa(valueA.c_str(), '_', 1).c_str();
2009          ExDisp = s.separa(valueA.c_str(), '_', 2).toInt();
2010        }
2011        if (header == "EX"){
2012          ArmEx = s.separa(valueA.c_str(), '_', 1).toInt();
2013          ArmRep = s.separa(valueA.c_str(), '_', 2).toInt();
2014        }
2015    }
2016    std::string infoBLE = pInfoCharacteristic->getValue();
2017    if (infoBLE == "Reminder"){
2018      Serial.println(Reminder);
2019      webServer();
2020      if (localip == true){
2021        pInfoCharacteristic->setValue(WiFi.localIP());
2022      }
2023    }
2024    if (infoBLE == "Ready"){
2025      Serial.println(infoBLE.c_str());
2026      pInfoCharacteristic->setValue("Start");
2027      display.clearDisplay();
2028      display.setTextSize(1.95);
2029      display.setTextColor(WHITE);
2030      display.setCursor(10, 1);
2031      display.println ("WELCOME TO MO&REP");
2032      display.setTextSize(1.7);
2033      display.println ("   by Gabriel Glez\n\n");
2034      display.println("Waiting instructions");
2035      display.display();
2036    }
2037    if (infoBLE == "Email"){
2038      mail = not(mail);
2039      pInfoCharacteristic->setValue("Settings");
2040    }
2041    if (infoBLE == "WorkoutOn"){
2042      workoutctrl = 1;
```

```
2043      pInfoCharacteristic->setValue("SessionOn");
2044      Serial.println(infoBLE.c_str());
2045    }
2046    if (infoBLE == "WorkoutOff"){
2047      workoutctrl = 2;
2048      pInfoCharacteristic->setValue("SessionOff");
2049    }
2050    if (infoBLE == "SessionOff"){
2051      FinishedTime = millis()-tmove;
2052    }
2053    if (infoBLE == "LegsOn"){
2054      Serial.println("LegsOn");
2055      pInfoCharacteristic->setValue("Ready");
2056      taskMPU.disable();
2057      taskMPULegs.enable();
2058    }
2059    if (infoBLE == "LegsOff"){
2060      pInfoCharacteristic->setValue("Ready");
2061      taskMPU.enable();
2062      taskMPULegs.disable();
2063    }
2064    if ((workoutctrl == 0 && (FinishedTime/1000) > 10) || infoBLE == "
          Sleep"){
2065      pInfoCharacteristic->setValue("Sleep");
2066      taskMPU.disable();
2067      taskMPULegs.disable();
2068      taskBLE.setInterval(TASK_SECOND*2);
2069      setCpuFrequencyMhz(80);
2070      display.clearDisplay();
2071      display.drawBitmap(0, 0, LogoMoRep, 128, 64, WHITE);
2072      display.display();
2073      FinishedTime = millis()-tmove;
2074      if (FinishedTime/1000 > 30 && infoBLE == "Sleep"){
2075        taskMPU.disable();
2076        taskMPULegs.disable();
2077        taskBLE.setInterval(10*TASK_SECOND);
2078        setCpuFrequencyMhz(80);
2079        display.clearDisplay();
```

```
2080      }
2081    }
2082
2083    else if (infoBLE == "WakeUp"){
2084      setCpuFrequencyMhz(240);
2085      taskMPU.enable();
2086      taskBLE.setInterval(TASK_SECOND/3);
2087      pInfoCharacteristic->setValue("Awaken");
2088      FinishedTime = 0;
2089      display.clearDisplay();
2090      display.setTextSize(1.95);
2091      display.setTextColor(WHITE);
2092      display.setCursor(10, 1);
2093      display.println ("WELCOME TO MO&REP");
2094      display.setTextSize(1.7);
2095      display.println ("   by Gabriel Glez\n\n");
2096      display.println("Waiting instructions");
2097      display.display();
2098    }
2099  });
2100
2101  Task taskwebServer(TASK_SECOND*5, TASK_FOREVER, []() {
2102    WiFiClient client = server.available();
2103    if (client) {
2104      currentTime = millis();
2105      previousTime = currentTime;
2106      String currentLine = "";
2107      while (client.connected()) {
2108        currentTime = millis();
2109        if (client.available()) {
2110          char c = client.read();
2111          Serial.write(c);
2112          head += c;
2113          if (c == '\n') {
2114            if (currentLine.length() == 0) {
2115              client.println("HTTP/1.1 200 OK");
2116              client.println("Content-type:text/html");
2117              client.println("Connection: close");
```

```
2118              client.println();
2119
2120              client.println("<head><meta name=\"viewport\" content=\"
                     width=device-width, initial-scale=1\">");
2121              client.println("<body bgcolor = black><h1><font color = #
                     F5B4FF><center>Mo&Rep Station</center></font></h1>");
2122              client.println("<h1><font size = 2 color = white><center>
                     ");
2123              client.println("<meta name=\"viewport\" content=\"width=
                     device-width, initial-scale=1\">");
2124              client.print(Reminder.c_str());
2125              client.print("</center></font></body>");
2126              break;
2127            } else {
2128              currentLine = "";
2129            }
2130          } else if (c != '\r') {
2131            currentLine += c;
2132          }
2133        }
2134      }
2135      head = "";
2136      client.stop();
2137      Serial.println("Client disconnected.");
2138      Serial.println("");
2139    }
2140  });
2141
2142  void Postures(){
2143    for (int i = 0; i < sizeof(accel_av)/sizeof(accel_av[0]); i++){
2144      if (accel_av[i] >= abs(1+3*(accel_id[i]+accel_id[i+2])/4)){
2145        display.clearDisplay();
2146        display.setTextSize(1.95);
2147        display.setTextColor(WHITE);
2148        display.setCursor(10, 1);
2149        display.print ("Fix your posture in ");
2150        display.print(WORKOUT.Ex[j-1]);
2151        display.display();
```

```
2152      }
2153    }
2154  }
2155
2156
2157  void setup() {
2158    Serial.begin(115200);
2159    EEPROM.begin(4096);
2160    Wire.begin();
2161    scan_i2c();
2162
2163    NimBLEDevice::init("Mo&Rep Station");
2164    NimBLEDevice::setSecurityAuth(/*BLE_M_PAIR_AUTHREQ_BOND |
          BLE_SM_PAIR_AUTHREQ_MITM |*/ BLE_SM_PAIR_AUTHREQ_SC);
2165    pServer = NimBLEDevice::createServer();
2166    pServer->setCallbacks(new ServerCallbacks());
2167    NimBLEService* pActionsService = pServer->createService((uint16_t)0
          x1826);
2168    pWristCharacteristic = pActionsService->createCharacteristic(
2169                                            CHARACTERISTIC_UUID_WRIST,
2170                                            NIMBLE_PROPERTY::READ |
2171                                            NIMBLE_PROPERTY::WRITE |
2172                                            NIMBLE_PROPERTY::NOTIFY |
2173                                            NIMBLE_PROPERTY::READ_ENC
2174                                            );
2175    pArmCharacteristic = pActionsService->createCharacteristic(
2176                                           CHARACTERISTIC_UUID_ARM,
2177                                           NIMBLE_PROPERTY::READ |
2178                                           NIMBLE_PROPERTY::WRITE |
2179                                           NIMBLE_PROPERTY::NOTIFY |
2180                                           NIMBLE_PROPERTY::READ_ENC
2181                                           );
2182    pInfoCharacteristic = pActionsService->createCharacteristic(
2183                                          CHARACTERISTIC_UUID_INFO,
2184                                          NIMBLE_PROPERTY::READ |
2185                                          NIMBLE_PROPERTY::WRITE |
2186                                          NIMBLE_PROPERTY::NOTIFY
2187                                          );
```

```
2188    pWristCharacteristic->setValue("WM_");
2189    pWristCharacteristic->setCallbacks(&chrCallbacks);
2190    pArmCharacteristic->setValue("AM_");
2191    pArmCharacteristic->setCallbacks(&chrCallbacks);
2192    pInfoCharacteristic->setValue("NONE");
2193    pInfoCharacteristic->setCallbacks(&chrCallbacks);
2194    pActionsService->start();
2195    NimBLEAdvertising* pAdvertising = NimBLEDevice::getAdvertising();
2196    pAdvertising->addServiceUUID(pActionsService->getUUID());
2197
2198    pAdvertising->setScanResponse(false);
2199    pAdvertising->start();
2200
2201    ReadingEEPROM();
2202    display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
2203    mpu.initialize();
2204    if (mpu.testConnection()){
2205      Serial.println("Accelerometer Detected");
2206    }
2207    else{
2208      Serial.println("Error in Accelerometer");
2209    }
2210
2211    display.clearDisplay();
2212    display.drawBitmap(0, 0, LogoMoRep, 128, 64, WHITE);
2213    display.display();
2214
2215    MoRepScheduler.addTask(taskMPU);
2216    taskMPU.enable();
2217    MoRepScheduler.addTask(taskBLE);
2218    taskBLE.enable();
2219    MoRepScheduler.addTask(taskMPULegs);
2220    MoRepScheduler.addTask(taskwebServer);
2221  }
2222
2223  void loop() {
2224    MoRepScheduler.execute();
2225  }
```

```
2226
2227 void FinishWorkout() {
2228   int n = 0;
2229
2230   Review += "<html>";
2231   Review += "<body>";
2232   Review += "<h1>Workout review:</h1>";
2233   Review += "<br>";
2234   for (k = 1; k <= j-1; k++){
2235     Analysis += std::string("Serie ") + String(k,DEC).c_str() + std::
          string(" with ") + String(WORKOUT.Series[k],DEC).c_str() + std
          ::string(" repetitions [") +  WORKOUT.Ex[k].c_str()
2236     + std::string("]<br>");
2237   }
2238
2239   Review += Analysis + std::string(" <br>") + std::string ("Exercises
       : <br>Biceps: ") + String(WORKOUT.MuscGroup[0],DEC).c_str()
2240           + std::string ("<br>Triceps: ") + String(WORKOUT.MuscGroup
                [1],DEC).c_str() + std::string ("<br>Chest: ") + String(
                WORKOUT.MuscGroup[2],DEC).c_str() + std::string ("<br>
                Back: ")
2241           + String(WORKOUT.MuscGroup[3],DEC).c_str() + std::string ("<
                br>Shoulders: ") + String(WORKOUT.MuscGroup[4],DEC).c_str
                () + std::string ("<br>Legs: ") + String(WORKOUT.
                MuscGroup[5],DEC).c_str()
2242           + std::string ("<br>TOTAL KCAL: "); + String(WORKOUT.kcal,
                DEC).c_str();
2243   Review += "</p>";
2244   Review += "</body>";
2245   Review += "</html>";
2246
2247   for (int m = 0; m <= int(Review.length()); m++){
2248     EEPROM.write(m, Review[m]);
2249   }
2250
2251     EEPROM.commit();
2252
2253   WiFi.begin(SECRET_SSID, SECRET_PASS);
```

```
2254     while (WiFi.status() != WL_CONNECTED){
2255      Serial.print(".");
2256      delay(200);
2257     }
2258
2259    WiFi.mode(WIFI_STA);
2260
2261    if (mail == true){
2262      smtp.debug(1);
2263      smtp.callback(smtpCallback);
2264      ESP_Mail_Session session;
2265      session.server.host_name = SMTP_HOST;
2266      session.server.port = SMTP_PORT;
2267      session.login.email = AUTHOR_EMAIL;
2268      session.login.password = AUTHOR_PASSWORD;
2269      session.login.user_domain = "moandrepsystem.net";
2270
2271      SMTP_Message message;
2272
2273      message.sender.name = "Mo&Rep System";
2274      message.sender.email = AUTHOR_EMAIL;
2275      message.subject = "Workout Session Review";
2276      message.addRecipient("Gabriel", "gabriel.glez.rial@gmail.com");
2277
2278      message.html.content = Review.c_str();
2279      message.html.charSet = "utf-8";
2280      message.html.transfer_encoding = Content_Transfer_Encoding::
            enc_7bit;
2281      message.priority = esp_mail_smtp_priority::
            esp_mail_smtp_priority_low;
2282      message.response.notify = esp_mail_smtp_notify_success |
            esp_mail_smtp_notify_failure | esp_mail_smtp_notify_delay;
2283      message.addHeader("Message-ID: <abcde.fghij@gmail.com>");
2284
2285      if (!smtp.connect(&session))
2286      return;
2287
2288      if (!MailClient.sendMail(&smtp, &message))
```

```
2289      Serial.println("Error sending Email, " + smtp.errorReason());
2290
2291      ESP_MAIL_PRINTF("Free Heap: %d\n", MailClient.getFreeHeap());
2292    }
2293
2294    ThingSpeak.begin(client);
2295    for (int i = 0; i < 5; i++){
2296      ThingSpeak.setField(i+1, WORKOUT.MuscGroup[i]);
2297    }
2298
2299    httpCode = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
2300
2301    Serial.print(httpCode);
2302    if (httpCode == 200) {
2303      Serial.println("Channel write successful.");
2304    }
2305    else {
2306      Serial.println("Problem writing to channel. HTTP error code " +
           String(httpCode));
2307    }
2308  }
2309
2310  void scan_i2c() {
2311    byte error, address;
2312    Serial.println("Scanning...");
2313    nDevices = 0;
2314    for (address = 1; address < 127; address++)
2315    {
2316      Wire.beginTransmission(address);
2317      error = Wire.endTransmission();
2318      if (error == 0)
2319      {
2320        Serial.print("I2C Device in ADDRESS 0x");
2321        if (address < 16)
2322          Serial.print("0");
2323        i2c_address[nDevices] = address;
2324        Serial.print(address, HEX);
2325        Serial.println("  !");
```

```
2326        nDevices++;
2327      }
2328      else if (error == 4)
2329      {
2330        Serial.print("Unknown error in ADDRESS 0x");
2331        if (address < 16)
2332          Serial.print("0");
2333        Serial.println(address, HEX);
2334      }
2335    }
2336    if (nDevices == 0)
2337      Serial.println("No I2C Device connected\n");
2338    else
2339      Serial.println("Done\n");
2340    delay(1000);
2341  }
2342
2343  void ReadingEEPROM(){
2344    for (int addr = 0; addr <= EEPROM_SIZE; addr++){
2345            Reminder += char(EEPROM.read(addr));
2346    }
2347    Reminder = Reminder.c_str();
2348  }
2349
2350  void smtpCallback(SMTP_Status status){
2351    /* Print the current status */
2352    Serial.println(status.info());
2353
2354    /* Print the sending result */
2355    if (status.success())
2356    {
2357      Serial.println("----------------");
2358      ESP_MAIL_PRINTF("Message sent success: %d\n", status.
          completedCount());
2359      ESP_MAIL_PRINTF("Message sent failled: %d\n", status.failedCount
          ());
2360      Serial.println("----------------\n");
2361      struct tm dt;
```

```
2362
2363      for (size_t i = 0; i < smtp.sendingResult.size(); i++)
2364      {
2365        /* Get the result item */
2366        SMTP_Result result = smtp.sendingResult.getItem(i);
2367        time_t ts = (time_t)result.timestamp;
2368        localtime_r(&ts, &dt);
2369
2370        ESP_MAIL_PRINTF("Message No: %d\n", i + 1);
2371        ESP_MAIL_PRINTF("Status: %s\n", result.completed ? "success" :
              "failed");
2372        ESP_MAIL_PRINTF("Date/Time: %d/%d/%d %d:%d:%d\n", dt.tm_year +
              1900, dt.tm_mon + 1, dt.tm_mday, dt.tm_hour, dt.tm_min, dt.
              tm_sec);
2373        ESP_MAIL_PRINTF("Recipient: %s\n", result.recipients);
2374        ESP_MAIL_PRINTF("Subject: %s\n", result.subject);
2375      }
2376      Serial.println("----------------\n");
2377    }
2378  }
2379
2380  void webServer(){
2381    Serial.print("Connecting to ");
2382    Serial.println(SECRET_SSID);
2383    WiFi.begin(SECRET_SSID, SECRET_PASS);
2384    while (WiFi.status() != WL_CONNECTED) {
2385      delay(500);
2386      Serial.print(".");
2387    }
2388    // Print local IP address and start web server
2389    Serial.println("");
2390    Serial.println("WiFi connected.");
2391    Serial.println("IP address: ");
2392    Serial.println(WiFi.localIP());
2393    localip = true;
2394    server.begin();
2395    taskwebServer.enable();
2396  }
```

Código correspondiente al nodo superior

```
#include <Arduino.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include "I2Cdev.h"
#include "MPU6050.h"
#include "string.h"
#include "Separador.h"
#include <NimBLEDevice.h>
#include <TaskScheduler.h>

Scheduler MoRepScheduler;

//I2C Chain
void scan_i2c();
byte i2c_address[5];
unsigned int nDevices = 0;

Separador s;

//MPU-6050
MPU6050 mpu(0x68);
int16_t ax, ay, az;
int16_t gx, gy, gz;
float ang_x, ang_y, ang_z, ang_disp;
float ang_x_prev[20], ang_y_prev[20], ang_z_prev[20];
String ArmPosition;
int ArmPos = 0;//Standing Arm Position
unsigned long tiempo_prev;
float dt;//Integration Step
int j = 1;
unsigned long tmove;
unsigned long tmove_prev;
int IniState = 1;
int state = 1;
int state_prev = 1;
```

```
37 int accel_ang_x;
38 int accel_ang_y;
39 int accel_ang_z;
40 float ax_m_s2_prev[5];
41 float ay_m_s2_prev[5];
42 float az_m_s2_prev[5];
43 float axv = 0;
44 float ayv = 0;
45 float azv = 0;
46 float angx_av = 0;
47 float angy_av = 0;
48 float angz_av = 0;
49 float ax_av, ay_av, az_av = 0;
50 float accel_av[2] = {0};
51 float accel_id[5] = {0};
52 int it = 0;
53 int n = 0;
54 bool order = false;
55
56 std::string valueW = "";
57 std::string valueA = "AM_1";
58 std::string WristMove = "";
59
60 struct myStruct {
61   int Series [30] = {0};
62   int Exercise [35] = {0};
63   int Ex[35];
64   int MuscGroup[5] = {0};
65 };
66
67 myStruct WORKOUT;
68
69 //BLE
70 static BLEUUID serviceUUID((uint16_t)0x1826);
71 static BLEUUID CHARACTERISTIC_UUID_WRIST ("d6597e78-c992-4f43
      -9861-719b78932fa6");
72 static BLEUUID CHARACTERISTIC_UUID_ARM ("84d8a625-08f9-49b3-82fb-
      d85daffc3c39");
```

```
73 static BLEUUID CHARACTERISTIC_UUID_INFO ("beb5483e-36e1-4688-b7f5-
      ea07361b26a8");
74
75 static boolean doConnect = false;
76 static boolean connected = false;
77 static boolean doScan = false;
78 static BLERemoteCharacteristic* pWristChar;
79 static BLERemoteCharacteristic* pArmChar;
80 static BLERemoteCharacteristic* pInfoChar;
81 static BLEAdvertisedDevice* myDevice;
82
83 static void notifyCallback(
84   BLERemoteCharacteristic* pBLERemoteCharacteristic,
85   uint8_t* pData,
86   size_t length,
87   bool isNotify) {
88     Serial.print("Notify callback for characteristic ");
89     Serial.print(pBLERemoteCharacteristic->getUUID().toString().c_str
         ());
90     Serial.print(" of data length ");
91     Serial.println(length);
92     Serial.print("data: ");
93     Serial.println((char*)pData);
94 }
95
96 class MyClientCallback : public BLEClientCallbacks {
97   void onConnect(BLEClient* pclient) {
98   }
99
100   void onDisconnect(BLEClient* pclient) {
101     connected = false;
102     Serial.println("onDisconnect");
103   }
104
105   uint32_t onPassKeyRequest(){
106     Serial.println("Client PassKeyRequest");
107     return 123456;
108   }
```

```
109   bool onConfirmPIN(uint32_t pass_key){
110     Serial.print("The passkey YES/NO number: ");Serial.println(
          pass_key);
111     return true;
112   }
113
114   void onAuthenticationComplete(ble_gap_conn_desc desc){
115     Serial.println("Starting BLE work!");
116   }
117 };
118
119 bool connectToServer() {
120     Serial.print("Forming a connection to ");
121     Serial.println(myDevice->getAddress().toString().c_str());
122
123     BLEClient*  pClient  = BLEDevice::createClient();
124     Serial.println(" - Created client");
125
126     pClient->setClientCallbacks(new MyClientCallback());
127
128     pClient->connect(myDevice);
129     Serial.println(" - Connected to server");
130
131     BLERemoteService* pRemoteService = pClient->getService(
          serviceUUID);
132     if (pRemoteService == nullptr) {
133       Serial.print("Failed to find our service UUID: ");
134       Serial.println(serviceUUID.toString().c_str());
135       pClient->disconnect();
136       return false;
137     }
138     Serial.println(" - Found our service");
139
140     pWristChar = pRemoteService->getCharacteristic(
          CHARACTERISTIC_UUID_WRIST);
141     pArmChar = pRemoteService->getCharacteristic(
          CHARACTERISTIC_UUID_ARM);
```

```
142     pInfoChar = pRemoteService->getCharacteristic(
           CHARACTERISTIC_UUID_INFO);
143     if (pWristChar == nullptr) {
144       Serial.print("Failed to find our characteristic UUID: ");
145       Serial.println(CHARACTERISTIC_UUID_WRIST.toString().c_str());
146       pClient->disconnect();
147       return false;
148     }
149     Serial.println(" - Found our characteristic");
150
151     if(pWristChar->canRead()) {
152       std::string valueW = pWristChar->readValue();
153       Serial.print("The characteristic value was: ");
154       Serial.println(valueW.c_str());
155     }
156
157     if(pWristChar->canNotify())
158       pWristChar->registerForNotify(notifyCallback);
159
160     if (pArmChar == nullptr) {
161       Serial.print("Failed to find our characteristic UUID: ");
162       Serial.println(CHARACTERISTIC_UUID_ARM.toString().c_str());
163       pClient->disconnect();
164       return false;
165     }
166     Serial.println(" - Found our characteristic");
167
168     if(pArmChar->canRead()) {
169       std::string valueA = pArmChar->readValue();
170       Serial.print("The characteristic value was: ");
171       Serial.println(valueA.c_str());
172     }
173
174     if(pArmChar->canNotify())
175       pArmChar->registerForNotify(notifyCallback);
176
177     if (pInfoChar == nullptr) {
178       Serial.print("Failed to find our characteristic UUID: ");
```

```
179       Serial.println(CHARACTERISTIC_UUID_INFO.toString().c_str());
180       pClient->disconnect();
181       return false;
182     }
183     Serial.println(" - Found our characteristic");
184
185     if(pInfoChar->canRead()) {
186       std::string infoBLE = pInfoChar->readValue();
187       Serial.print("The characteristic value was: ");
188       Serial.println(infoBLE.c_str());
189     }
190
191     if(pInfoChar->canNotify())
192       pInfoChar->registerForNotify(notifyCallback);
193
194     connected = true;
195     return true;
196 }
197
198 class MyAdvertisedDeviceCallbacks: public
       BLEAdvertisedDeviceCallbacks {
199   void onResult(BLEAdvertisedDevice* advertisedDevice) {
200     Serial.print("BLE Advertised Device found: ");
201     Serial.println(advertisedDevice->toString().c_str());
202
203     if (advertisedDevice->haveServiceUUID() && advertisedDevice->
          isAdvertisingService(serviceUUID)) {
204       BLEDevice::getScan()->stop();
205       myDevice = advertisedDevice;
206       doConnect = true;
207       doScan = true;
208     }
209   }
210 };
211
212 Task taskMPULegs(TASK_SECOND / 5, TASK_FOREVER, []() {
213   if (WristMove == "1") {
214     order = true;
```

```
215     }
216
217     if (order == true){
218       float ax_m_s2;
219       float ay_m_s2;
220       float az_m_s2;
221       mpu.getAcceleration(&ax, &ay, &az);
222       mpu.getRotation(&gx, &gy, &gz);
223       dt = (millis() - tiempo_prev) / 1000.0;
224       tiempo_prev = millis();
225       tmove = millis();
226       ax_m_s2 = ax * (9.81 / 16384.0);
227       ay_m_s2 = ay * (9.81 / 16384.0);
228       az_m_s2 = az * (9.81 / 16384.0);
229       ang_x = 0.98 *  (gx / 131) * dt + 0.02 * accel_ang_x;
230       ang_y = 0.98 *  (gy / 131) * dt + 0.02 * accel_ang_y;
231       ang_z = 0.98 *  (gz / 131) * dt + 0.02 * accel_ang_z;
232       accel_ang_x = atan(ax / sqrt(pow(ay, 2) + pow(az, 2))) * (180.0 /
              3.14);
233       accel_ang_y = atan(ay / sqrt(pow(ax, 2) + pow(az, 2))) * (180.0 /
              3.14);
234       accel_ang_z = atan(az / sqrt(pow(ax, 2) + pow(ay, 2))) * (180.0 /
              3.14);
235
236       if (0.6 > dt > 0.5){
237         ax_m_s2_prev[n] = ax_m_s2;
238         ay_m_s2_prev[n] = ay_m_s2;
239         az_m_s2_prev[n] = az_m_s2;
240         axv = axv + ax_m_s2_prev[n];
241         ayv = ayv + ay_m_s2_prev[n];
242         azv = azv + az_m_s2_prev[n];
243         n++;
244       }
245       if (n == 5){
246         n = 0;
247       }
248
249       axv = axv/(n+1);
```

```
250       ayv = ayv/(n+1);
251       azv = azv/(n+1);
252
253       if (az_m_s2 > 7) {
254         ArmPosition = "Eta";
255         ArmPos = 1;
256       }
257       else if (az_m_s2 < -7) {
258         ArmPosition = "Theta";
259         ArmPos = 3;
260       }
261       else if (ax_m_s2 > 7) {
262         ArmPosition = "Mi";
263         ArmPos = 4;
264       }
265       else if (ax_m_s2 < -7) {
266         ArmPosition = "Lambda";
267         ArmPos = 6;
268       }
269       else if (ay_m_s2 < -7) {
270         ArmPosition = "Iota";
271         ArmPos = 5;
272       }
273       else if (ay_m_s2 > 7) {
274         ArmPosition = "Kappa";
275         ArmPos = 2;
276       }
277       else {
278         ArmPosition = "Half way";
279         ArmPos = 0;
280       }
281
282       switch (state) {
283         case (1): //State detection
284           if (WristMove == "1") {
285             valueA = "AM_1";
286             if (ArmPosition == "Kappa") {
287               IniState = 2;
```

```
288             state = 2;
289             tmove_prev = tmove;
290           }
291           if (ArmPosition == "Iota") {
292             state = 5;
293             IniState = 5;
294             tmove_prev = tmove;
295           }
296           if (ArmPosition == "Mi") {
297             state = 4;
298             IniState = 4;
299             tmove_prev = tmove;
300           }
301           if (ArmPosition == "Theta") {
302             state = 3;
303             IniState = 3;
304             tmove_prev = tmove;
305           }
306           if (ArmPosition == "Lambda") {
307             state = 6;
308             IniState = 6;
309             tmove_prev = tmove;
310           }
311           if (ArmPosition == "Eta"){
312             state = 7;
313             IniState = 7;
314             tmove_prev = tmove;
315           }
316           else if (WristMove == "97") {
317             j++;
318           }
319         }
320         break;
321
322     case (2): //Kappa
323       if (ArmPos == 3) {
324         state = 3;
325         tmove_prev = tmove;
```

```
326           state_prev = 2;
327         }
328         if (ArmPos == 4) {
329           state = 4;
330           tmove_prev = tmove;
331           state_prev = 2;
332         }
333         if (ArmPos == 5) {
334           state = 5;
335           tmove_prev = tmove;
336           state_prev = 2;
337         }
338         if (ArmPos == 6) {
339           state = 6;
340           tmove_prev = tmove;
341           state_prev = 2;
342         }
343         if (ArmPos == 1) {
344           state = 7;
345           tmove_prev = tmove;
346           state_prev = 2;
347         }
348         if (tmove - tmove_prev > 2000 && tmove - tmove_prev < 12000) {
349           valueA = "AM_02";
350         }
351         if (tmove - tmove_prev > 12000) {
352           valueA = "AM_99";
353         }
354         if ((WristMove == "99" && valueA == "AM_99") || WristMove == "
              98") {
355           state = 8;
356           tmove_prev = tmove;
357         }
358         break;
359
360       case (3): //Theta
361         if (ArmPos == 4) {
362           state = 4;
```

```
363           tmove_prev = tmove;
364           state_prev = 3;
365         }
366         if (ArmPos == 5) {
367           state = 5;
368           tmove_prev = tmove;
369           state_prev = 3;
370         }
371         if (ArmPos == 6) {
372           state = 6;
373           tmove_prev = tmove;
374           state_prev = 3;
375         }
376         if (ArmPos == 2) {
377           tmove_prev = tmove;
378           state = 2;
379           state_prev = 3;
380         }
381         if (ArmPos == 1) {
382           state = 7;
383           tmove_prev = tmove;
384           state_prev = 3;
385         }
386         if (tmove - tmove_prev > 12000) {
387           valueA = "AM_99";
388         }
389         if ((WristMove == "99" && valueA == "AM_99") || WristMove == "
              98") {
390           state = 8;
391           tmove_prev = tmove;
392         }
393         break;
394       case (4): //Mi
395         if (ArmPos == 2) {
396           state = 2;
397           tmove_prev = tmove;
398           state_prev = 4;
399         }
```

```
400         if (ArmPos == 3) {
401           state = 3;
402           tmove_prev = tmove;
403           state_prev = 4;
404         }
405         if (ArmPos == 1) {
406           state = 7;
407           tmove_prev = tmove;
408           state_prev = 4;
409           if (WristMove == "05" && IniState == 4){
410             valueA = "AM_47_1";
411             WORKOUT.Exercise[5]++;
412           }
413         }
414         if (ArmPos == 5) {
415           state = 5;
416           tmove_prev = tmove;
417           state_prev = 4;
418         }
419         if (ArmPos == 6) {
420           state = 6;
421           tmove_prev = tmove;
422           state_prev = 4;
423         }
424         if (tmove - tmove_prev > 12000) {
425           valueA = "AM_99";
426         }
427         if ((WristMove == "99" && valueA == "AM_99") || WristMove == "
              98") {
428           state = 8;
429           tmove_prev = tmove;
430         }
431         break;
432
433       case (5): //Iota
434         if (ArmPos == 3) {
435           state = 3;
436           tmove_prev = tmove;
```

```
437           state_prev = 5;
438         }
439         if (ArmPos == 4) {
440           state = 4;
441           tmove_prev = tmove;
442           state_prev = 5;
443         }
444         if (ArmPos == 2) {
445           state = 2;
446           tmove_prev = tmove;
447           state_prev = 5;
448         }
449         if (ArmPos == 1) {
450           state = 7;
451           tmove_prev = tmove;
452           state_prev = 5;
453         }
454         if (ArmPos == 6) {
455           state = 6;
456           tmove_prev = tmove;
457           state_prev = 5;
458         }
459         if (tmove - tmove_prev > 12000) {
460           valueA = "AM_99";
461         }
462         if ((WristMove == "99" && valueA == "AM_99") || WristMove == "
               98") {
463           state = 8;
464           tmove_prev = tmove;
465         }
466         break;
467
468       case (6): //Lambda
469       if (ArmPos == 2) {
470           state = 2;
471           tmove_prev = tmove;
472           state_prev = 6;
473           if (WristMove == "07" && IniState == 6){
```

```
474             WORKOUT.Exercise[6]++;
475             valueA = valueA + "_1";
476           }
477         }
478         if (ArmPos == 3) {
479           state = 3;
480           tmove_prev = tmove;
481           state_prev = 6;
482         }
483         if (ArmPos == 4) {
484           state = 4;
485           tmove_prev = tmove;
486           state_prev = 6;
487         }
488         if (ArmPos == 5) {
489           state = 5;
490           tmove_prev = tmove;
491           state_prev = 6;
492         }
493         if (ArmPos == 6) {
494           state = 6;
495           tmove_prev = tmove;
496           state_prev = 6;
497         }
498         if (ArmPos == 1) {
499           state = 7;
500           tmove_prev = tmove;
501           state_prev = 6;
502         }
503         if (tmove - tmove_prev > 12000) {
504           valueA = "AM_99";
505         }
506         if ((WristMove == "99" && valueA == "AM_99") || WristMove == "
               98") {
507           state = 8;
508           tmove_prev = tmove;
509         }
510         break;
```

```
511
512     case (7): //Eta
513       if (ArmPos == 3) {
514         state = 3;
515         tmove_prev = tmove;
516         state_prev = 7;
517       }
518       if (ArmPos == 4) {
519         state = 4;
520         tmove_prev = tmove;
521         if (WristMove == "05" && IniState == 7){
522           valueA = "AM_74_1";
523           WORKOUT.Exercise[2]++;
524         }
525       }
526       if (ArmPos == 5) {
527         state = 5;
528         tmove_prev = tmove;
529         state_prev = 7;
530       }
531       if (ArmPos == 6) {
532         state = 6;
533         tmove_prev = tmove;
534         state_prev = 7;
535       }
536       if (tmove - tmove_prev > 2000 && tmove - tmove_prev < 12000) {
537         valueA = "AM_07";
538       }
539       if (tmove - tmove_prev > 12000) {
540         valueA = "AM_99";
541       }
542       if ((WristMove == "99" && valueA == "AM_99") || WristMove == "
             98") {
543         state = 8;
544         tmove_prev = tmove;
545       }
546       break;
547
```

```
548       case (8): //Exercise detection
549         if (WORKOUT.Exercise[2] < 3) {
550           WORKOUT.Exercise[2] = 0;
551         }
552         if (WORKOUT.Exercise[5] < 3) {
553           WORKOUT.Exercise[11] = 0;
554         }
555         if (WORKOUT.Exercise[6] < 3) {
556           WORKOUT.Exercise[6] = 0;
557         }
558         if (WORKOUT.Exercise[2] > 3) {
559           WORKOUT.Series [j] = WORKOUT.Exercise[2];
560           WORKOUT.Ex[j] = 2;
561           WORKOUT.Exercise[2] = 0;
562           WORKOUT.MuscGroup[5]++;
563         }
564         if (WORKOUT.Exercise[5] > 3) {
565           WORKOUT.Series [j] = WORKOUT.Exercise[5];
566           WORKOUT.Ex[j] = 5;
567           WORKOUT.Exercise[5] = 0;
568           WORKOUT.MuscGroup[5]++;
569         }
570         if (WORKOUT.Exercise[6] > 3) {
571           WORKOUT.Series [j] = WORKOUT.Exercise[6];
572           WORKOUT.Ex[j] = 6;
573           WORKOUT.Exercise[6] = 0;
574           WORKOUT.MuscGroup[5]++;
575         }
576         if (WORKOUT.Series[j] > 3) {
577           valueA = std::string ("EX_") + String(WORKOUT.Ex[j]).c_str()
                 + std::string ("_") + String(WORKOUT.Series[j]).c_str();
578           j++;
579         }
580         else{
581           valueA = "AM_1";
582         }
583         state = 1;
584         break;
```

```
585     }
586
587     if (WristMove == "96") {
588       order = false;
589       }
590     }
591   });
592
593   Task taskMPU(TASK_SECOND / 5, TASK_FOREVER, []() {
594     if (WristMove == "1") {
595       order = true;
596     }
597
598     if (order == true){
599       float ax_m_s2;
600       float ay_m_s2;
601       float az_m_s2;
602       mpu.getAcceleration(&ax, &ay, &az);
603       mpu.getRotation(&gx, &gy, &gz);
604       dt = (millis() - tiempo_prev) / 1000.0;
605       tiempo_prev = millis();
606       tmove = millis();
607       ax_m_s2 = ax * (9.81 / 16384.0);
608       ay_m_s2 = ay * (9.81 / 16384.0);
609       az_m_s2 = az * (9.81 / 16384.0);
610       ang_x = 0.98 *  (gx / 131) * dt + 0.02 * accel_ang_x;
611       ang_y = 0.98 *  (gy / 131) * dt + 0.02 * accel_ang_y;
612       ang_z = 0.98 *  (gz / 131) * dt + 0.02 * accel_ang_z;
613       accel_ang_x = atan(ax / sqrt(pow(ay, 2) + pow(az, 2))) * (180.0 /
              3.14);
614       accel_ang_y = atan(ay / sqrt(pow(ax, 2) + pow(az, 2))) * (180.0 /
              3.14);
615       accel_ang_z = atan(az / sqrt(pow(ax, 2) + pow(ay, 2))) * (180.0 /
              3.14);
616
617       if (0.6 > dt > 0.5){
618         ax_m_s2_prev[n] = ax_m_s2;
619         ay_m_s2_prev[n] = ay_m_s2;
```

```
620         az_m_s2_prev[n] = az_m_s2;
621         axv = axv + ax_m_s2_prev[n];
622         ayv = ayv + ay_m_s2_prev[n];
623         azv = azv + az_m_s2_prev[n];
624         n++;
625       }
626       if (n == 5){
627         n = 0;
628       }
629
630       axv = axv/(n+1);
631       ayv = ayv/(n+1);
632       azv = azv/(n+1);
633
634       if (az_m_s2 > 7) {
635         ArmPosition = "Eta";
636         ArmPos = 1;
637       }
638       else if (az_m_s2 < -7) {
639         ArmPosition = "Theta";
640         ArmPos = 3;
641       }
642       else if (ax_m_s2 > 7) {
643         ArmPosition = "Mi";
644         ArmPos = 4;
645       }
646       else if (ax_m_s2 < -7) {
647         ArmPosition = "Lambda";
648         ArmPos = 6;
649       }
650       else if (ay_m_s2 < -7) {
651         ArmPosition = "Iota";
652         ArmPos = 5;
653       }
654       else if (ay_m_s2 > 7) {
655         ArmPosition = "Kappa";
656         ArmPos = 2;
657       }
```

```
658     else {
659       ArmPosition = "Half way";
660       ArmPos = 0;
661     }
662 
663     switch (state) {
664       case (1): //State detection
665         if (WristMove == "1") {
666           valueA = "AM_1";
667           if (ArmPosition == "Kappa") {
668             IniState = 2;
669             state = 2;
670             tmove_prev = tmove;
671           }
672           if (ArmPosition == "Iota") {
673             state = 5;
674             IniState = 5;
675             tmove_prev = tmove;
676           }
677           if (ArmPosition == "Mi") {
678             state = 4;
679             IniState = 4;
680             tmove_prev = tmove;
681           }
682           if (ArmPosition == "Theta") {
683             state = 3;
684             IniState = 3;
685             tmove_prev = tmove;
686           }
687           if (ArmPosition == "Lambda") {
688             state = 6;
689             IniState = 6;
690             tmove_prev = tmove;
691           }
692           else if (WristMove == "97") {
693             j++;
694           }
695         }
```

```
696           break;
697
698       case (2): //Kappa
699         if (ArmPos == 3) {
700           state = 3;
701           tmove_prev = tmove;
702         }
703         if (ArmPos == 4) {
704           if (IniState == 2 && (WristMove == "03Q" || (WristMove == "99
                " && WORKOUT.Exercise[18] > 1)) && WORKOUT.Exercise[19] ==
                 0) {
705             WORKOUT.Exercise[18]++;
706             valueA = valueA + "_1";
707           }
708           if (IniState == 4 && (WristMove == "03A" || (WristMove == "99
                " && WORKOUT.Exercise[19] > 1)) && WORKOUT.Exercise[18] ==
                 0) {
709             WORKOUT.Exercise[19]++;
710             valueA = valueA + "_1";
711           }
712           state = 4;
713           tmove_prev = tmove;
714         }
715         if (ArmPos == 5) {
716           state = 5;
717           tmove_prev = tmove;
718         }
719         if (ArmPos == 6) {
720           state = 6;
721           tmove_prev = tmove;
722         }
723         if (ArmPos == 1) {
724           tmove_prev = tmove;
725         }
726         if (tmove - tmove_prev > 2000 && tmove - tmove_prev < 12000) {
727           valueA = "AM_02";
728         }
729         if (tmove - tmove_prev > 12000) {
```

```
730           valueA = "AM_99";
731         }
732         if ((WristMove == "99" && valueA == "AM_99") || WristMove == "
               98") {
733           state = 7;
734           tmove_prev = tmove;
735         }
736         break;
737 
738       case (3): //Theta
739         if (ArmPos == 4) {
740           state = 4;
741           tmove_prev = tmove;
742           if (WristMove == "06" && IniState == 3){
743             WORKOUT.Exercise[21]++;
744             valueA = valueA + "_1";
745           }
746         }
747         if (ArmPos == 5) {
748           state = 5;
749           tmove_prev = tmove;
750           valueA = "AM_35";
751         }
752         if (ArmPos == 6) {
753           state = 6;
754           tmove_prev = tmove;
755         }
756         if (ArmPos == 2) {
757           valueA = "AM_32";
758           tmove_prev = tmove;
759           state = 2;
760         }
761         if (ArmPos == 1) {
762           tmove_prev = tmove;
763         }
764         if (tmove - tmove_prev > 2000 && tmove - tmove_prev < 12000) {
765           valueA = "AM_03";
766         }
```

```
767         if (tmove - tmove_prev > 12000) {
768           valueA = "AM_99";
769         }
770         if ((WristMove == "99" && valueA == "AM_99") || WristMove == "
               98") {
771           state = 7;
772           tmove_prev = tmove;
773         }
774         break;
775       case (4): //Mi
776         if (ArmPos == 2) {
777           valueA = "AM_42";
778           if (IniState == 4){
779             if (WristMove == "23") {
780               WORKOUT.Exercise[12]++;
781               valueA = valueA + "_1";
782             }
783             if (WristMove == "03A" || (WristMove == "99" || WristMove
                  == "03Q" && (WORKOUT.Exercise[24] > 1 || WORKOUT.
                  Exercise[11] > 1))){
784               if (WORKOUT.MuscGroup[4] > WORKOUT.MuscGroup[2]){
785                 WORKOUT.Exercise[24]++;
786                 valueA = valueA + "_1";
787               }
788               if (WORKOUT.MuscGroup[4] < WORKOUT.MuscGroup[2]){
789                 WORKOUT.Exercise[11]++;
790                 valueA = valueA + "_1";
791               }
792             }
793           }
794           state = 2;
795           tmove_prev = tmove;
796         }
797         if (ArmPos == 3) {
798           state = 3;
799           tmove_prev = tmove;
800         }
801         if (ArmPos == 1) {
```

```
802           tmove_prev = tmove;
803         }
804         if (ArmPos == 5) {
805           valueA = "AM_45";
806           if (IniState == 4 && WristMove == "05") {
807             WORKOUT.Exercise[13]++;
808             valueA = valueA + "_1";
809           }
810           state = 5;
811           tmove_prev = tmove;
812         }
813         if (ArmPos == 6) {
814           state = 6;
815           valueA = "AM_46";
816           tmove_prev = tmove;
817         }
818         if (tmove - tmove_prev > 2000 && tmove - tmove_prev < 12000) {
819           if ((az_m_s2_prev[n] - azv  <  2.5) && az_m_s2_prev[n] != 0)
                 {
820             valueA = "AM_04Q";
821             tmove_prev = tmove;
822           }
823           if ((az_m_s2_prev[n] - azv > 2.5) && az_m_s2_prev != 0 &&
                  ax_m_s2 > -2) {
824             valueA = "AM_04A";
825             tmove_prev = tmove;
826           }
827         }
828         if (tmove - tmove_prev > 12000) {
829           valueA = "AM_99";
830         }
831         if ((WristMove == "99" && valueA == "AM_99") || WristMove == "
              98") {
832           state = 7;
833           tmove_prev = tmove;
834         }
835         break;
836
```

```
837     case (5): //Iota
838       if (ArmPos == 3) {
839         state = 3;
840         tmove_prev = tmove;
841         valueA = "AM_53";
842       }
843       if (ArmPos == 4) {
844         state = 4;
845         valueA = "AM_54";
846         tmove_prev = tmove;
847       }
848       if (ArmPos == 2) {
849         valueA = "AM_52";
850         if (WristMove == "03A" || (WristMove == "99" && WORKOUT.
               Exercise[20] > 1)){
851           WORKOUT.Exercise[20]++;
852           valueA = valueA + "_1";
853         }
854         state = 2;
855         tmove_prev = tmove;
856       }
857       if (ArmPos == 1) {
858         tmove_prev = tmove;
859       }
860       if (tmove - tmove_prev > 2000 && tmove - tmove_prev < 12000) {
861         valueA = "AM_05";
862       }
863       if (tmove - tmove_prev > 12000) {
864         valueA = "AM_99";
865       }
866       if (ArmPos == 6) {
867         if (IniState == 5){
868           valueA = "AM_56";
869           if ((WristMove == "04Q" || (WristMove == "99" && (WORKOUT.
                 Exercise[10] > 1 || WORKOUT.Exercise[15] > 1))) &&
                 WORKOUT.Exercise[31] == 0) {
870             if (WORKOUT.MuscGroup[1] > WORKOUT.MuscGroup[2]) {
871               WORKOUT.Exercise[10]++;
```

```
872                valueA = valueA + "_1";
873              }
874              else {
875                WORKOUT.Exercise[15]++;
876                valueA = valueA + "_1";
877              }
878            if ((WristMove == "04A" || ((WristMove == "04Q" ||
                  WristMove == "99") && WORKOUT.Exercise[31] > 1)) &&
                  WORKOUT.Exercise[10] == 0 && WORKOUT.Exercise[15] == 0){
879              WORKOUT.Exercise[31]++;
880              valueA = valueA + "_1";
881            }
882            }
883          }
884          state = 6;
885          tmove_prev = tmove;
886        }
887        if ((WristMove == "99" && valueA == "AM_99") || WristMove == "
              98") {
888          state = 7;
889          tmove_prev = tmove;
890        }
891        break;
892
893      case (6): //Lambda
894        if (ArmPos == 3) {
895          state = 3;
896          tmove_prev = tmove;
897        }
898        if (ArmPos == 4) {
899          state = 4;
900          tmove_prev = tmove;
901        }
902        if (ArmPos == 5) {
903          state = 5;
904          tmove_prev = tmove;
905        }
906        if (ArmPos == 6) {
```

```
907           state = 6;
908           tmove_prev = tmove;
909         }
910         if (ArmPos == 1) {
911           tmove_prev = tmove;
912         }
913         if (tmove - tmove_prev > 2000 && tmove - tmove_prev < 12000) {
914           valueA = "AM_06";
915         }
916         if (tmove - tmove_prev > 12000) {
917           valueA = "AM_99";
918         }
919         if ((WristMove == "99" && valueA == "AM_99") || WristMove == "
               98") {
920           state = 7;
921           tmove_prev = tmove;
922         }
923         break;
924 
925       case (7): //Exercise detection
926         if (WORKOUT.Exercise[10] < 3) {
927           WORKOUT.Exercise[10] = 0;
928         }
929         if (WORKOUT.Exercise[11] < 3) {
930           WORKOUT.Exercise[11] = 0;
931         }
932         if (WORKOUT.Exercise[15] < 3) {
933           WORKOUT.Exercise[15] = 0;
934         }
935         if (WORKOUT.Exercise[18] < 3) {
936           WORKOUT.Exercise[18] = 0;
937         }
938         if (WORKOUT.Exercise[19] < 3) {
939           WORKOUT.Exercise[19] = 0;
940         }
941         if (WORKOUT.Exercise[20] < 3) {
942           WORKOUT.Exercise[20] = 0;
943         }
```

```
944         if (WORKOUT.Exercise[21] < 3) {
945           WORKOUT.Exercise[21] = 0;
946         }
947         if (WORKOUT.Exercise[24] < 3) {
948           WORKOUT.Exercise[24] = 0;
949         }
950         if (WORKOUT.Exercise[31] < 3) {
951           WORKOUT.Exercise[31] = 0;
952         }
953         if (WORKOUT.Exercise[10] > 3) {
954           WORKOUT.Series [j] = WORKOUT.Exercise[10];
955           WORKOUT.Ex[j] = 10;
956           WORKOUT.Exercise[10] = 0;
957           WORKOUT.MuscGroup[1]++;
958         }
959         if (WORKOUT.Exercise[11] > 3) {
960           WORKOUT.Series [j] = WORKOUT.Exercise[11];
961           WORKOUT.Ex[j] = 11;
962           WORKOUT.Exercise[11] = 0;
963           WORKOUT.MuscGroup[2]++;
964         }
965         if (WORKOUT.Exercise[15] > 3) {
966           WORKOUT.Series [j] = WORKOUT.Exercise[15];
967           WORKOUT.Ex[j] = 15;
968           WORKOUT.Exercise[15] = 0;
969           WORKOUT.MuscGroup[2]++;
970         }
971         if (WORKOUT.Exercise[18] > 3) {
972           WORKOUT.Series[j] = WORKOUT.Exercise[18];
973           WORKOUT.Exercise[18] = 0;
974           WORKOUT.Ex[j] = 18;
975           WORKOUT.MuscGroup[3]++;
976         }
977         if (WORKOUT.Exercise[19] > 3) {
978           WORKOUT.Series[j] = WORKOUT.Exercise[19];
979           WORKOUT.Exercise[19] = 0;
980           WORKOUT.Ex[j] = 19;
981           WORKOUT.MuscGroup[3]++;
```

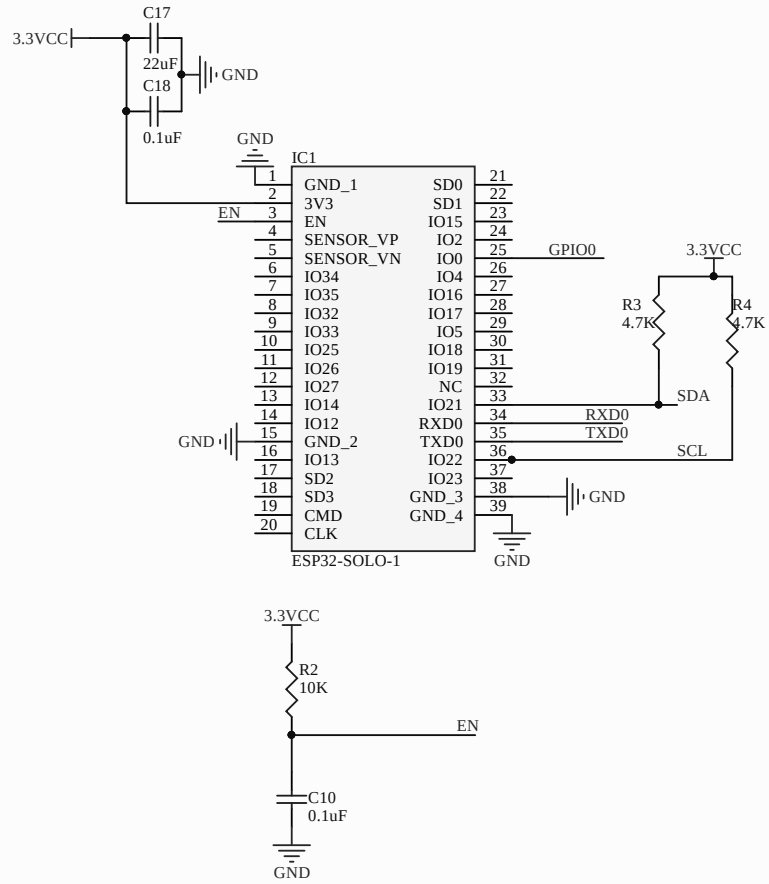
```
982         }
983         if (WORKOUT.Exercise[20] > 3) {
984           WORKOUT.Series[j] = WORKOUT.Exercise[19];
985           WORKOUT.Exercise[20] = 0;
986           WORKOUT.Ex[j] = 20;
987           WORKOUT.MuscGroup[3]++;
988         }
989         if (WORKOUT.Exercise[21] > 3) {
990           WORKOUT.Series[j] = WORKOUT.Exercise[21];
991           WORKOUT.Exercise[21] = 0;
992           WORKOUT.Ex[j] = 21;
993           WORKOUT.MuscGroup[4]++;
994         }
995         if (WORKOUT.Exercise[24] > 3) {
996           WORKOUT.Series [j] = WORKOUT.Exercise[24];
997           WORKOUT.Ex[j] = 24;
998           WORKOUT.Exercise[24] = 0;
999           WORKOUT.MuscGroup[4]++;
1000        }
1001       if (WORKOUT.Exercise[31] > 3) {
1002          WORKOUT.Series[j] = WORKOUT.Exercise[31];
1003          WORKOUT.Exercise[31] = 0;
1004          WORKOUT.Ex[j] = 31;
1005          WORKOUT.MuscGroup[3]++;
1006        }
1007        if (WORKOUT.Series[j] > 3) {
1008          valueA = std::string ("EX_") + String(WORKOUT.Ex[j]).c_str()
                 + std::string ("_") + String(WORKOUT.Series[j]).c_str();
1009          j++;
1010        }
1011        else{
1012          valueA = "AM_1";
1013        }
1014        state = 1;
1015        break;
1016    }
1017
1018    if (WristMove == "96") {
```

```
1019      order = false;
1020      }
1021    }
1022
1023  });
1024
1025  Task taskBLE(TASK_SECOND / 3.5, TASK_FOREVER, []() {
1026    if (doConnect == true){
1027      if (connectToServer()){
1028        Serial.println("Connected");
1029      }
1030      else{
1031        Serial.println("Not connected");
1032      }
1033     doConnect = false;
1034    }
1035
1036    if(connected){
1037      /* Set the value */
1038      pArmChar->writeValue(valueA);
1039      valueW = pWristChar->readValue();
1040      std::string infoBLE = pInfoChar->readValue();
1041      std::string header = valueW.substr(0, 2);//Decoding of messages
1042      if (header == "WM"){
1043        WristMove = s.separa(valueW.c_str(), '_', 1).c_str();
1044      }
1045      if (infoBLE == "LegsOn"){
1046        taskMPU.disable();
1047        taskMPULegs.enable();
1048      }
1049      if (infoBLE == "Sleep"){
1050        taskMPU.disable();
1051        taskMPULegs.disable();
1052        taskBLE.setInterval(10*TASK_SECOND);
1053        setCpuFrequencyMhz(80);
1054      }
1055      if (infoBLE == "WakeUp"){
1056        setCpuFrequencyMhz(240);
```

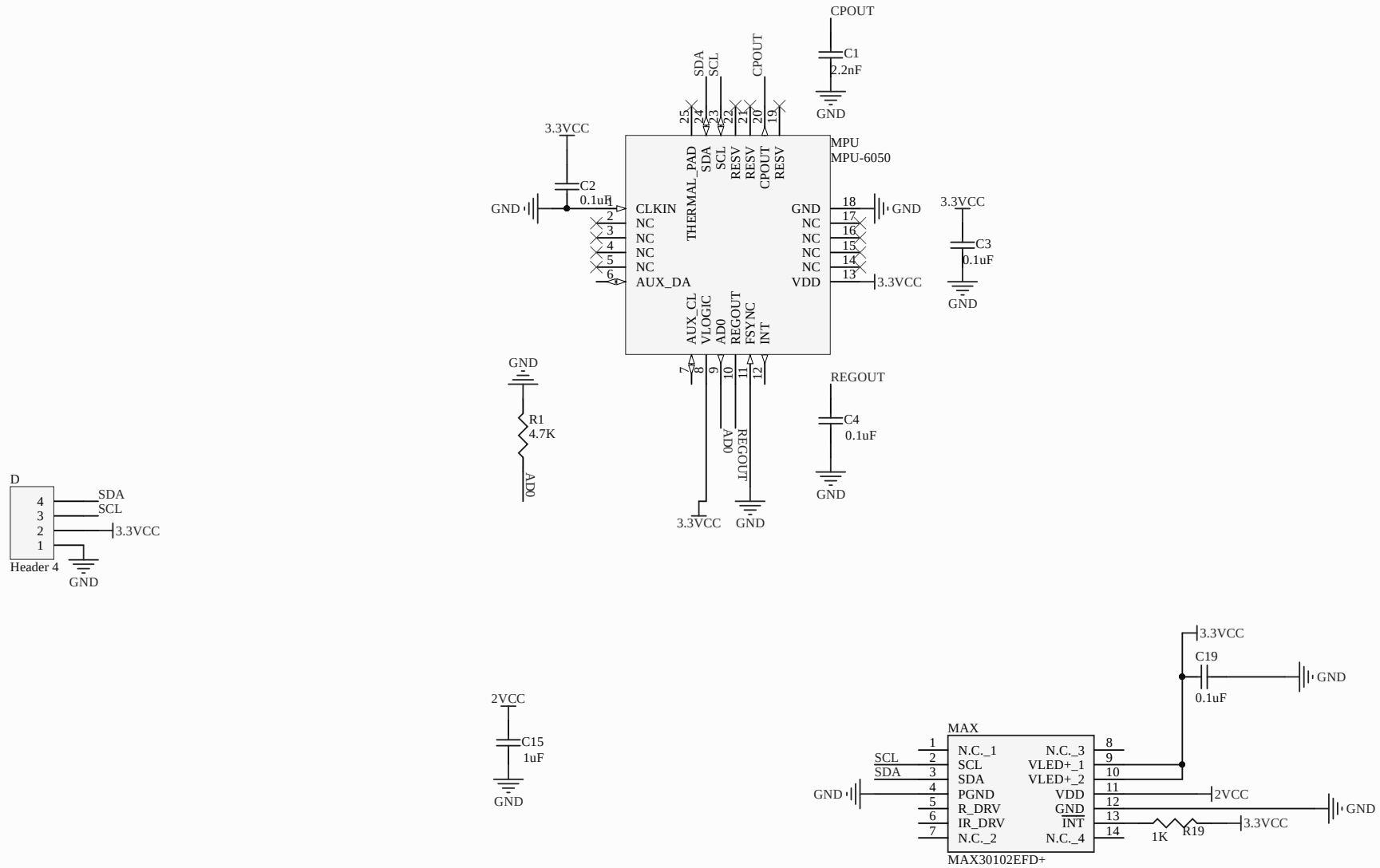
```
1057        taskMPU.enable();
1058        taskBLE.setInterval(TASK_SECOND/3);
1059      }
1060      if (infoBLE == "LegsOff"){
1061        taskMPU.enable();
1062        taskMPULegs.disable();
1063      }
1064    }
1065  });
1066
1067  void setup() {
1068    Serial.begin(115200);
1069    Wire.begin();
1070    BLEDevice::init("");
1071    MoRepScheduler.addTask(taskMPU);
1072    MoRepScheduler.addTask(taskBLE);
1073    MoRepScheduler.addTask(taskMPULegs);
1074    taskMPU.enable();
1075    taskBLE.enable();
1076    scan_i2c();
1077    mpu.initialize();
1078
1079    if (mpu.testConnection()){
1080      Serial.println("Accelerometer Detected");
1081    }
1082    else{
1083      Serial.println("Error in Accelerometer");
1084    }
1085
1086    BLEScan* pBLEScan = BLEDevice::getScan();
1087    pBLEScan->setAdvertisedDeviceCallbacks(new
          MyAdvertisedDeviceCallbacks());
1088    pBLEScan->setInterval(1349);
1089    pBLEScan->setWindow(449);
1090    pBLEScan->setActiveScan(true);
1091    pBLEScan->start(5, false);
1092  }
1093
```

```
1094 void loop() {
1095   MoRepScheduler.execute();
1096 }
1097
1098 void scan_i2c() {
1099   byte error, address;
1100   Serial.println("Scanning...");
1101   nDevices = 0;
1102   for (address = 1; address < 127; address++)
1103   {
1104     Wire.beginTransmission(address);
1105     error = Wire.endTransmission();
1106     if (error == 0)
1107     {
1108       Serial.print("I2C Device in ADDRESS 0x");
1109       if (address < 16)
1110         Serial.print("0");
1111       i2c_address[nDevices] = address;
1112       Serial.print(address, HEX);
1113       Serial.println("  !");
1114       nDevices++;
1115     }
1116     else if (error == 4)
1117     {
1118       Serial.print("Unknown error in ADDRESS 0x");
1119       if (address < 16)
1120         Serial.print("0");
1121       Serial.println(address, HEX);
1122     }
1123   }
1124   if (nDevices == 0)
1125     Serial.println("No I2C Device connected\n");
1126   else
1127     Serial.println("Done\n");
1128   delay(1000);
1129 }
```

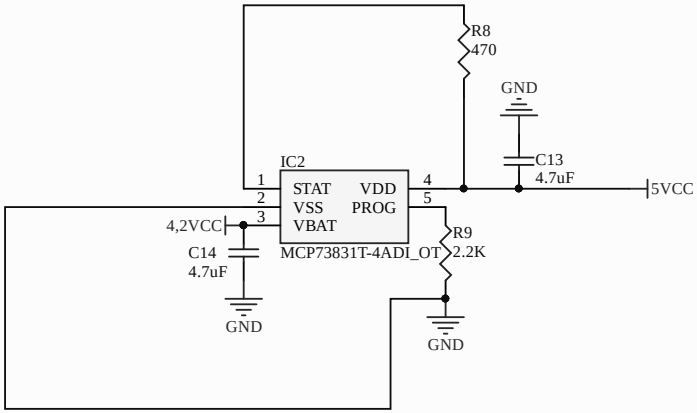
Esquemáticos y *layout* del sistema

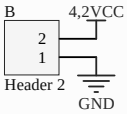


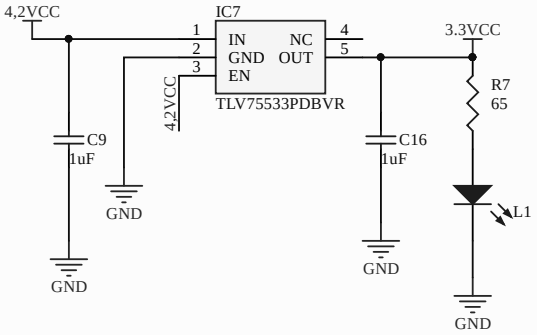
Title		
Size: A4	Number	Revision
Date: 14/09/2021		Sheet of
File: C:\Users\..\ESP32.SchDoc		Drawn By:

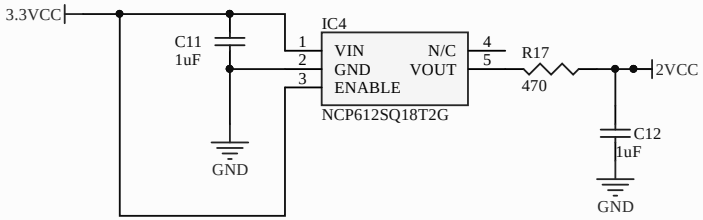


Title		
Size: A4	Number	Revision
Date: 14/09/2021		Sheet of
File: C:\Users\..\MPU6050.SchDoc		Drawn By:

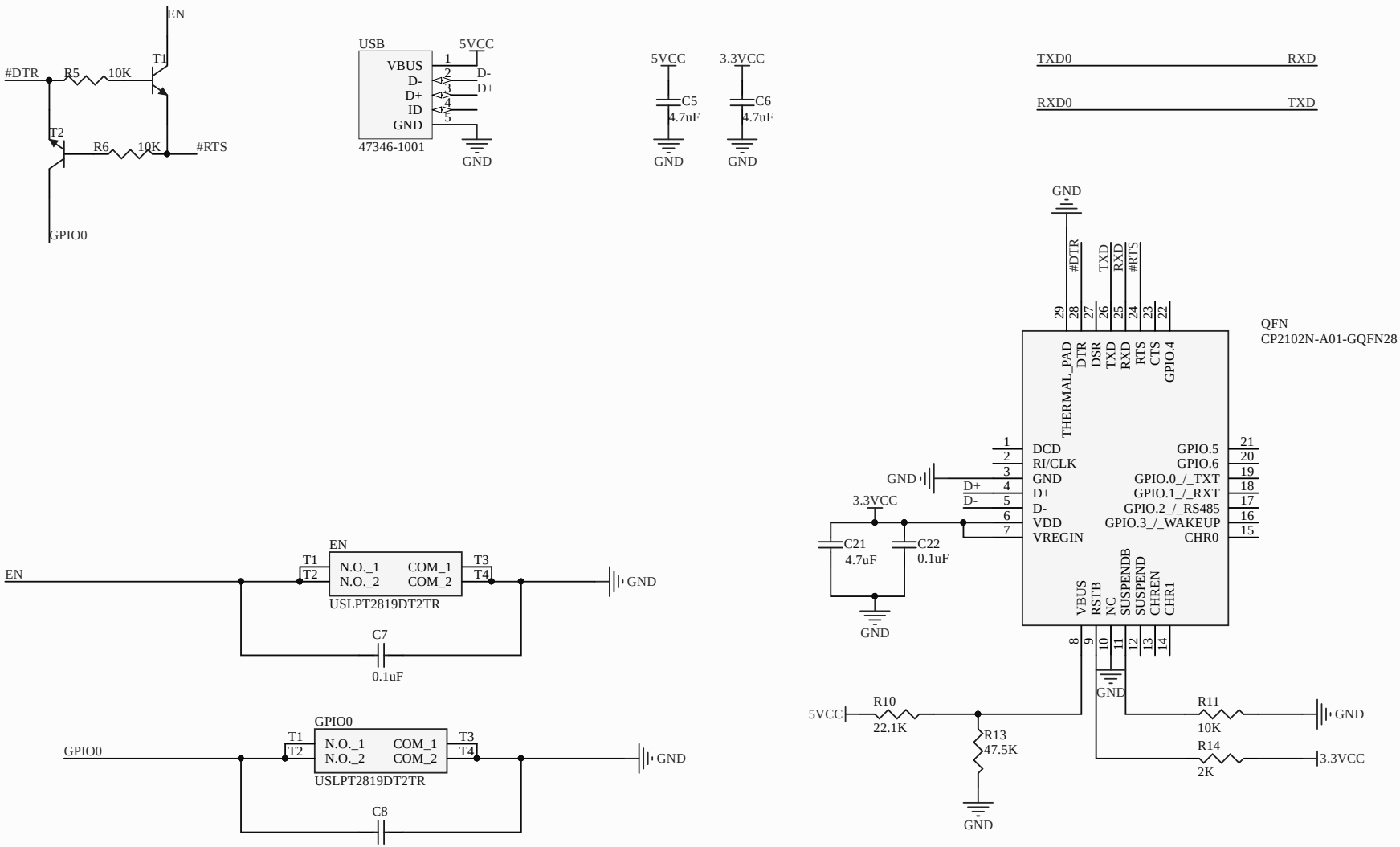








Title		
Size: A4	Number	Revision
Date: 14/09/2021		Sheet of
File: C:\Users\...\PWRSUPPLY.SchDoc		Drawn By:

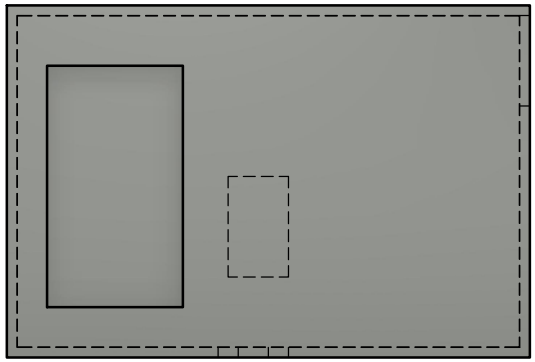


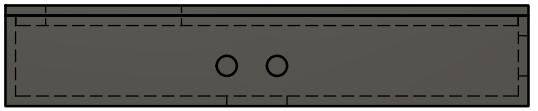
Title		
Size: A4	Number	Revision
Date: 14/09/2021		Sheet of
File: C:\Users\..\USBTOUART.SchDoc		Drawn By:

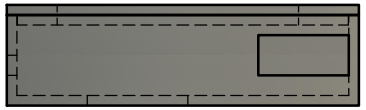
C21
C18
C17
C2
C22
C6
R2
C11
C16
C3
C10
C4
IC1
MPU
R1
v.1.2.2021
C7
L1
R7
C9
C14
B1
R3
C5
IC2
USB
R9
C13
C8
R8
R11
R14
R10
GPIO0
QFN
EN
R12
R6
C8
T2
T1
C7
R5
Gabriel Glez
MO&REP

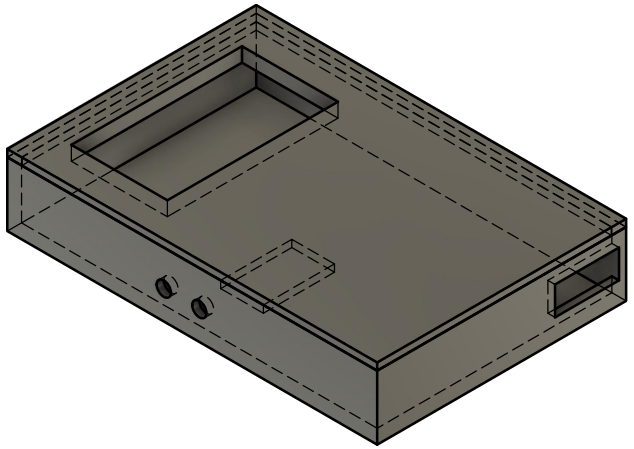
Vistas del encapsulado del sistema

Encapsulado del nodo inferior

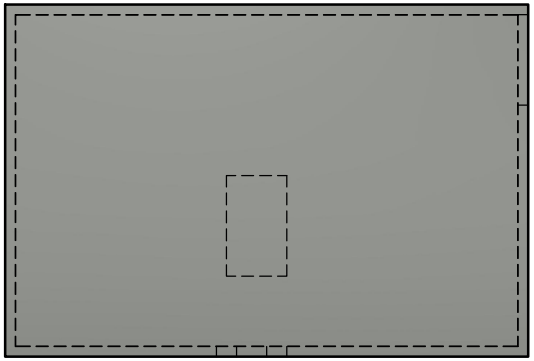


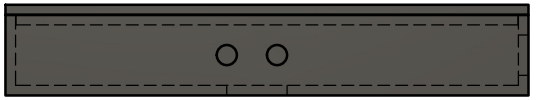


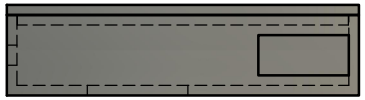


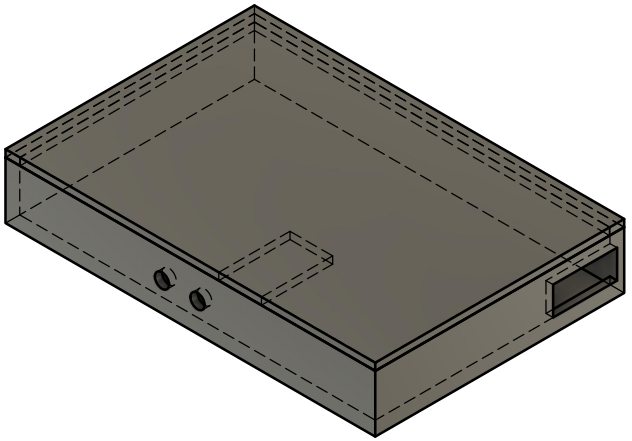


Encapsulado del nodo superior









Datasheets de los componentes

ESP32-WROOM-32

Datasheet

Version 2.9

Espressif Systems

About This Document

This document provides the specifications for the ESP32-WROOM-32 module.

Revision History

For revision history of this document, please refer to the last page.

Documentation Change Notification

Espressif provides email notifications to keep customers updated on changes to technical documentation. Please subscribe at www.espressif.com/en/subscribe.

Certification

Download certificates for Espressif products from www.espressif.com/en/certificates.

Disclaimer and Copyright Notice

Contents

1 Overview 1

2 Pin Definitions 3
2.1 Pin Layout 3
2.2 Pin Description 3
2.3 Strapping Pins 4

3 Functional Description 6
3.1 CPU and Internal Memory 6
3.2 External Flash and SRAM 6
3.3 Crystal Oscillators 6
3.4 RTC and Low-Power Management 7

4 Peripherals and Sensors 8

5 Electrical Characteristics 9
5.1 Absolute Maximum Ratings 9
5.2 Recommended Operating Conditions 9
5.3 DC Characteristics (3.3 V, 25 °C) 9
5.4 Wi-Fi Radio 10
5.5 BLE Radio 11
5.5.1 Receiver 11
5.5.2 Transmitter 11
5.6 Reflow Profile 12

6 Schematics 13

7 Peripheral Schematics 14

8 Physical Dimensions 16

9 Recommended PCB Land Pattern 17

10 Learning Resources 18
10.1 Must-Read Documents 18
10.2 Must-Have Resources 18

Revision History 19

List of Tables

1	ESP32-WROOM-32 Specifications	1
2	Pin Definitions	3
3	Strapping Pins	5
4	Absolute Maximum Ratings	9
5	Recommended Operating Conditions	9
6	DC Characteristics (3.3 V, 25 °C)	9
7	Wi-Fi Radio Characteristics	10
8	Receiver Characteristics – BLE	11
9	Transmitter Characteristics – BLE	11

List of Figures

1 ESP32-WROOM-32 Pin Layout (Top View) 3
2 Reflow Profile 12
3 ESP32-WROOM-32 Schematics 13
4 ESP32-WROOM-32 Peripheral Schematics 14
5 Discharge Circuit for VDD33 Rail 14
6 Reset Circuit 15
7 Physical Dimensions of ESP32-WROOM-32 16
8 Recommended PCB Land Pattern 17

1. Overview

ESP32-WROOM-32 is a powerful, generic Wi-Fi+BT+BLE MCU module that targets a wide variety of applications, ranging from low-power sensor networks to the most demanding tasks, such as voice encoding, music streaming and MP3 decoding.

At the core of this module is the ESP32-D0WDQ6 chip*. The chip embedded is designed to be scalable and adaptive. There are two CPU cores that can be individually controlled, and the CPU clock frequency is adjustable from 80 MHz to 240 MHz. The user may also power off the CPU and make use of the low-power co-processor to constantly monitor the peripherals for changes or crossing of thresholds. ESP32 integrates a rich set of peripherals, ranging from capacitive touch sensors, Hall sensors, SD card interface, Ethernet, high-speed SPI, UART, I²S and I²C.

Note:

* For details on the part numbers of the ESP32 family of chips, please refer to the document *ESP32 Datasheet*.

The integration of Bluetooth, Bluetooth LE and Wi-Fi ensures that a wide range of applications can be targeted, and that the module is all-around: using Wi-Fi allows a large physical range and direct connection to the Internet through a Wi-Fi router, while using Bluetooth allows the user to conveniently connect to the phone or broadcast low energy beacons for its detection. The sleep current of the ESP32 chip is less than 5 μA, making it suitable for battery powered and wearable electronics applications. The module supports a data rate of up to 150 Mbps, and 20 dBm output power at the antenna to ensure the widest physical range. As such the module does offer industry-leading specifications and the best performance for electronic integration, range, power consumption, and connectivity.

The operating system chosen for ESP32 is freeRTOS with LwIP; TLS 1.2 with hardware acceleration is built in as well. Secure (encrypted) over the air (OTA) upgrade is also supported, so that users can upgrade their products even after their release, at minimum cost and effort.

Table 1 provides the specifications of ESP32-WROOM-32.

Table 1: ESP32-WROOM-32 Specifications

Categories	Items	Specifications
Certification	RF certification	FCC/CE-RED/IC/TELEC/KCC/SRRC/NCC
	Wi-Fi certification	Wi-Fi Alliance
	Bluetooth certification	BQB
	Green certification	RoHS/REACH
Test	Reliablity	HTOL/HTSL/uHAST/TCT/ESD
Wi-Fi	Protocols	802.11 b/g/n (802.11n up to 150 Mbps)
		A-MPDU and A-MSDU aggregation and 0.4 μs guard interval support
	Frequency range	2.4 GHz ~ 2.5 GHz
Bluetooth	Protocols	Bluetooth v4.2 BR/EDR and BLE specification
	Radio	NZIF receiver with –97 dBm sensitivity
		Class-1, class-2 and class-3 transmitter
		AFH

Categories	Items	Specifications
	Audio	CVSD and SBC
Hardware	Module interfaces	SD card, UART, SPI, SDIO, I^2C, LED PWM, Motor PWM, I^2S, IR, pulse counter, GPIO, capacitive touch sensor, ADC, DAC
	On-chip sensor	Hall sensor
	Integrated crystal	40 MHz crystal
	Integrated SPI flash	4 MB
	Operating voltage/Power supply	3.0 V ~ 3.6 V
	Operating current	Average: 80 mA
	Minimum current delivered by power supply	500 mA
	Recommended operating temperature range	–40 °C ~ +85 °C
	Package size	(18.00±0.10) mm × (25.50±0.10) mm × (3.10±0.10) mm
	Moisture sensitivity level (MSL)	Level 3

2. Pin Definitions

2.1 Pin Layout

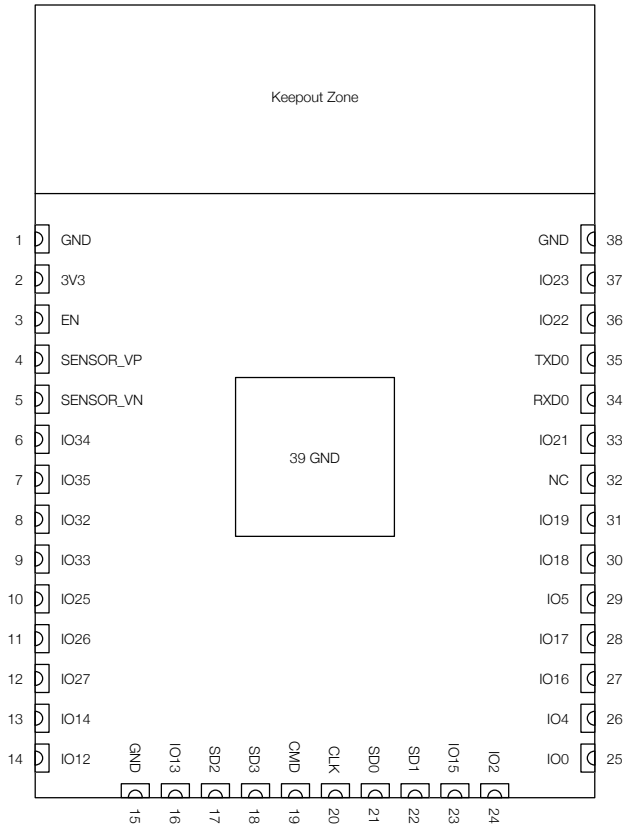


Figure 1: ESP32-WROOM-32 Pin Layout (Top View)

2.2 Pin Description

ESP32-WROOM-32 has 38 pins. See pin definitions in Table 2.

Table 2: Pin Definitions

Name	No.	Type	Function
GND	1	P	Ground
3V3	2	P	Power supply
EN	3	I	Module-enable signal. Active high.
SENSOR_VP	4	I	GPIO36, ADC1_CH0, RTC_GPIO0
SENSOR_VN	5	I	GPIO39, ADC1_CH3, RTC_GPIO3
IO34	6	I	GPIO34, ADC1_CH6, RTC_GPIO4
IO35	7	I	GPIO35, ADC1_CH7, RTC_GPIO5
IO32	8	I/O	GPIO32, XTAL_32K_P (32.768 kHz crystal oscillator input), ADC1_CH4, TOUCH9, RTC_GPIO9
IO33	9	I/O	GPIO33, XTAL_32K_N (32.768 kHz crystal oscillator output), ADC1_CH5, TOUCH8, RTC_GPIO8

Name	No.	Type	Function
IO25	10	I/O	GPIO25, DAC_1, ADC2_CH8, RTC_GPIO6, EMAC_RXD0
IO26	11	I/O	GPIO26, DAC_2, ADC2_CH9, RTC_GPIO7, EMAC_RXD1
IO27	12	I/O	GPIO27, ADC2_CH7, TOUCH7, RTC_GPIO17, EMAC_RX_DV
IO14	13	I/O	GPIO14, ADC2_CH6, TOUCH6, RTC_GPIO16, MTMS, HSPICLK, HS2_CLK, SD_CLK, EMAC_TXD2
IO12	14	I/O	GPIO12, ADC2_CH5, TOUCH5, RTC_GPIO15, MTDI, HSPIQ, HS2_DATA2, SD_DATA2, EMAC_TXD3
GND	15	P	Ground
IO13	16	I/O	GPIO13, ADC2_CH4, TOUCH4, RTC_GPIO14, MTCK, HSPID, HS2_DATA3, SD_DATA3, EMAC_RX_ER
SHD/SD2*	17	I/O	GPIO9, SD_DATA2, SPIHD, HS1_DATA2, U1RXD
SWP/SD3*	18	I/O	GPIO10, SD_DATA3, SPIWP, HS1_DATA3, U1TXD
SCS/CMD*	19	I/O	GPIO11, SD_CMD, SPICS0, HS1_CMD, U1RTS
SCK/CLK*	20	I/O	GPIO6, SD_CLK, SPICLK, HS1_CLK, U1CTS
SDO/SD0*	21	I/O	GPIO7, SD_DATA0, SPIQ, HS1_DATA0, U2RTS
SDI/SD1*	22	I/O	GPIO8, SD_DATA1, SPID, HS1_DATA1, U2CTS
IO15	23	I/O	GPIO15, ADC2_CH3, TOUCH3, MTDO, HSPICS0, RTC_GPIO13, HS2_CMD, SD_CMD, EMAC_RXD3
IO2	24	I/O	GPIO2, ADC2_CH2, TOUCH2, RTC_GPIO12, HSPIWP, HS2_DATA0, SD_DATA0
IO0	25	I/O	GPIO0, ADC2_CH1, TOUCH1, RTC_GPIO11, CLK_OUT1, EMAC_TX_CLK
IO4	26	I/O	GPIO4, ADC2_CH0, TOUCH0, RTC_GPIO10, HSPIHD, HS2_DATA1, SD_DATA1, EMAC_TX_ER
IO16	27	I/O	GPIO16, HS1_DATA4, U2RXD, EMAC_CLK_OUT
IO17	28	I/O	GPIO17, HS1_DATA5, U2TXD, EMAC_CLK_OUT_180
IO5	29	I/O	GPIO5, VSPICS0, HS1_DATA6, EMAC_RX_CLK
IO18	30	I/O	GPIO18, VSPICLK, HS1_DATA7
IO19	31	I/O	GPIO19, VSPIQ, U0CTS, EMAC_TXD0
NC	32	-	-
IO21	33	I/O	GPIO21, VSPIHD, EMAC_TX_EN
RXD0	34	I/O	GPIO3, U0RXD, CLK_OUT2
TXD0	35	I/O	GPIO1, U0TXD, CLK_OUT3, EMAC_RXD2
IO22	36	I/O	GPIO22, VSPIWP, U0RTS, EMAC_TXD1
IO23	37	I/O	GPIO23, VSPID, HS1_STROBE
GND	38	P	Ground

Notice:

* Pins SCK/CLK, SDO/SD0, SDI/SD1, SHD/SD2, SWP/SD3 and SCS/CMD, namely, GPIO6 to GPIO11 are connected to the integrated SPI flash integrated on the module and are not recommended for other uses.

2.3 Strapping Pins

ESP32 has five strapping pins, which can be seen in Chapter 6 Schematics:

- MTDI
- GPIO0
- GPIO2
- MTDO
- GPIO5

Software can read the values of these five bits from register "GPIO_STRAPPING".

During the chip's system reset release (power-on-reset, RTC watchdog reset and brownout reset), the latches of the strapping pins sample the voltage level as strapping bits of "0" or "1", and hold these bits until the chip is powered down or shut down. The strapping bits configure the device's boot mode, the operating voltage of VDD_SDIO and other initial system settings.

Each strapping pin is connected to its internal pull-up/pull-down during the chip reset. Consequently, if a strapping pin is unconnected or the connected external circuit is high-impedance, the internal weak pull-up/pull-down will determine the default input level of the strapping pins.

To change the strapping bit values, users can apply the external pull-down/pull-up resistances, or use the host MCU's GPIOs to control the voltage level of these pins when powering on ESP32.

After reset release, the strapping pins work as normal-function pins.

Refer to Table 3 for a detailed boot-mode configuration by strapping pins.

Table 3: Strapping Pins

Voltage of Internal LDO (VDD_SDIO)					
Pin	Default	3.3 V		1.8 V	
MTDI	Pull-down	0		1	
Booting Mode					
Pin	Default	SPI Boot		Download Boot	
GPIO0	Pull-up	1		0	
GPIO2	Pull-down	Don't-care		0	
Enabling/Disabling Debugging Log Print over U0TXD During Booting					
Pin	Default	U0TXD Active		U0TXD Silent	
MTDO	Pull-up	1		0	
Timing of SDIO Slave					
Pin	Default	Falling-edge Sampling Falling-edge Output	Falling-edge Sampling Rising-edge Output	Rising-edge Sampling Falling-edge Output	Rising-edge Sampling Rising-edge Output
MTDO	Pull-up	0	0	1	1
GPIO5	Pull-up	0	1	0	1

Note:

- Firmware can configure register bits to change the settings of "Voltage of Internal LDO (VDD_SDIO)" and "Timing of SDIO Slave" after booting.
- The module integrates a 3.3 V SPI flash, so the pin MTDI cannot be set to 1 when the module is powered up.

The strapping pins need a setup and hold time before and after the EN signal goes high. For details please refer to Section Strapping Pins in *ESP32 Datasheet*.

3. Functional Description

This chapter describes the modules and functions integrated in ESP32-WROOM-32.

3.1 CPU and Internal Memory

ESP32-D0WDQ6 contains two low-power Xtensa® 32-bit LX6 microprocessors. The internal memory includes:

- 448 KB of ROM for booting and core functions.
- 520 KB of on-chip SRAM for data and instructions.
- 8 KB of SRAM in RTC, which is called RTC FAST Memory and can be used for data storage; it is accessed by the main CPU during RTC Boot from the Deep-sleep mode.
- 8 KB of SRAM in RTC, which is called RTC SLOW Memory and can be accessed by the co-processor during the Deep-sleep mode.
- 1 Kbit of eFuse: 256 bits are used for the system (MAC address and chip configuration) and the remaining 768 bits are reserved for customer applications, including flash-encryption and chip-ID.

3.2 External Flash and SRAM

ESP32 supports multiple external QSPI flash and SRAM chips. More details can be found in Chapter SPI in the *ESP32 Technical Reference Manual*. ESP32 also supports hardware encryption/decryption based on AES to protect developers' programs and data in flash.

ESP32 can access the external QSPI flash and SRAM through high-speed caches.

- The external flash can be mapped into CPU instruction memory space and read-only memory space simultaneously.
  - When external flash is mapped into CPU instruction memory space, up to 11 MB + 248 KB can be mapped at a time. Note that if more than 3 MB + 248 KB are mapped, cache performance will be reduced due to speculative reads by the CPU.
  - When external flash is mapped into read-only data memory space, up to 4 MB can be mapped at a time. 8-bit, 16-bit and 32-bit reads are supported.
- External SRAM can be mapped into CPU data memory space. Up to 4 MB can be mapped at a time. 8-bit, 16-bit and 32-bit reads and writes are supported.

ESP32-WROOM-32 integrates a 4 MB SPI flash, which is connected to GPIO6, GPIO7, GPIO8, GPIO9, GPIO10 and GPIO11. These six pins cannot be used as regular GPIOs.

3.3 Crystal Oscillators

The module uses a 40-MHz crystal oscillator.

3.4 RTC and Low-Power Management

With the use of advanced power-management technologies, ESP32 can switch between different power modes.

For details on ESP32's power consumption in different power modes, please refer to section "RTC and Low-Power Management" in *ESP32 Datasheet*.

4. Peripherals and Sensors

Please refer to Section Peripherals and Sensors in *ESP32 Datasheet*.

Note:
External connections can be made to any GPIO except for GPIOs in the range 6-11. These six GPIOs are connected to the module's integrated SPI flash. For details, please see Section 6 Schematics.

5. Electrical Characteristics

5.1 Absolute Maximum Ratings

Stresses beyond the absolute maximum ratings listed in Table 4 below may cause permanent damage to the device. These are stress ratings only, and do not refer to the functional operation of the device that should follow the recommended operating conditions.

Table 4: Absolute Maximum Ratings

Symbol	Parameter	Min	Max	Unit
VDD33	Power supply voltage	–0.3	3.6	V
I_{output} [1]	Cumulative IO output current	-	1,100	mA
T_{store}	Storage temperature	–40	150	°C

1. The module worked properly after a 24-hour test in ambient temperature at 25 °C, and the IOs in three domains (VDD3P3_RTC, VDD3P3_CPU, VDD_SDIO) output high logic level to ground. Please note that pins occupied by flash and/or PSRAM in the VDD_SDIO power domain were excluded from the test.
2. Please see Appendix IO_MUX of *ESP32 Datasheet* for IO's power domain.

5.2 Recommended Operating Conditions

Table 5: Recommended Operating Conditions

Symbol	Parameter	Min	Typical	Max	Unit
VDD33	Power supply voltage	3.0	3.3	3.6	V
I_{VDD}	Current delivered by external power supply	0.5	-	-	A
T	Operating temperature	–40	-	85	°C

5.3 DC Characteristics (3.3 V, 25 °C)

Table 6: DC Characteristics (3.3 V, 25 °C)

Symbol	Parameter		Min	Typ	Max	Unit
C_{IN}	Pin capacitance		-	2	-	pF
V_{IH}	High-level input voltage		0.75×VDD[1]	-	VDD[1]+0.3	V
V_{IL}	Low-level input voltage		–0.3	-	0.25×VDD[1]	V
I_{IH}	High-level input current		-	-	50	nA
I_{IL}	Low-level input current		-	-	50	nA
V_{OH}	High-level output voltage		0.8×VDD[1]	-	-	V
V_{OL}	Low-level output voltage		-	-	0.1×VDD[1]	V
I_{OH}	High-level source current (VDD[1] = 3.3 V, V_{OH} >= 2.64 V, output drive strength set to the maximum)	VDD3P3_CPU power domain [1, 2]	-	40	-	mA
		VDD3P3_RTC power domain [1, 2]	-	40	-	mA
		VDD_SDIO power domain [1, 3]	-	20	-	mA

Symbol	Parameter	Min	Typ	Max	Unit
I_{OL}	Low-level sink current (VDD[1] = 3.3 V, V_{OL} = 0.495 V, output drive strength set to the maximum)	-	28	-	mA
R_{PU}	Resistance of internal pull-up resistor	-	45	-	kΩ
R_{PD}	Resistance of internal pull-down resistor	-	45	-	kΩ
V_{IL_nRST}	Low-level input voltage of CHIP_PU to power off the chip	-	-	0.6	V

Notes:

1. Please see Appendix IO_MUX of *ESP32 Datasheet* for IO's power domain. VDD is the I/O voltage for a particular power domain of pins.
2. For VDD3P3_CPU and VDD3P3_RTC power domain, per-pin current sourced in the same domain is gradually reduced from around 40 mA to around 29 mA, V_{OH}>=2.64 V, as the number of current-source pins increases.
3. Pins occupied by flash and/or PSRAM in the VDD_SDIO power domain were excluded from the test.

5.4 Wi-Fi Radio

Table 7: Wi-Fi Radio Characteristics

Parameter	Condition	Min	Typical	Max	Unit
Operating frequency range [note1]	-	2412	-	2484	MHz
Output impedance [note2]	-	-	*note 2*	-	Ω
TX power [note3]	11n, MCS7	12	13	14	dBm
	11b mode	17.5	18.5	20	dBm
Sensitivity	11b, 1 Mbps	-	–98	-	dBm
	11b, 11 Mbps	-	–89	-	dBm
	11g, 6 Mbps	-	–92	-	dBm
	11g, 54 Mbps	-	–74	-	dBm
	11n, HT20, MCS0	-	–91	-	dBm
	11n, HT20, MCS7	-	–71	-	dBm
	11n, HT40, MCS0	-	–89	-	dBm
	11n, HT40, MCS7	-	–69	-	dBm
Adjacent channel rejection	11g, 6 Mbps	-	31	-	dB
	11g, 54 Mbps	-	14	-	dB
	11n, HT20, MCS0	-	31	-	dB
	11n, HT20, MCS7	-	13	-	dB

1. Device should operate in the frequency range allocated by regional regulatory authorities. Target operating frequency range is configurable by software.
2. For the modules that use IPEX antennas, the output impedance is 50 Ω. For other modules without IPEX antennas, users do not need to concern about the output impedance.
3. Target TX power is configurable based on device or certification requirements.

5.5 BLE Radio

5.5.1 Receiver

Table 8: Receiver Characteristics – BLE

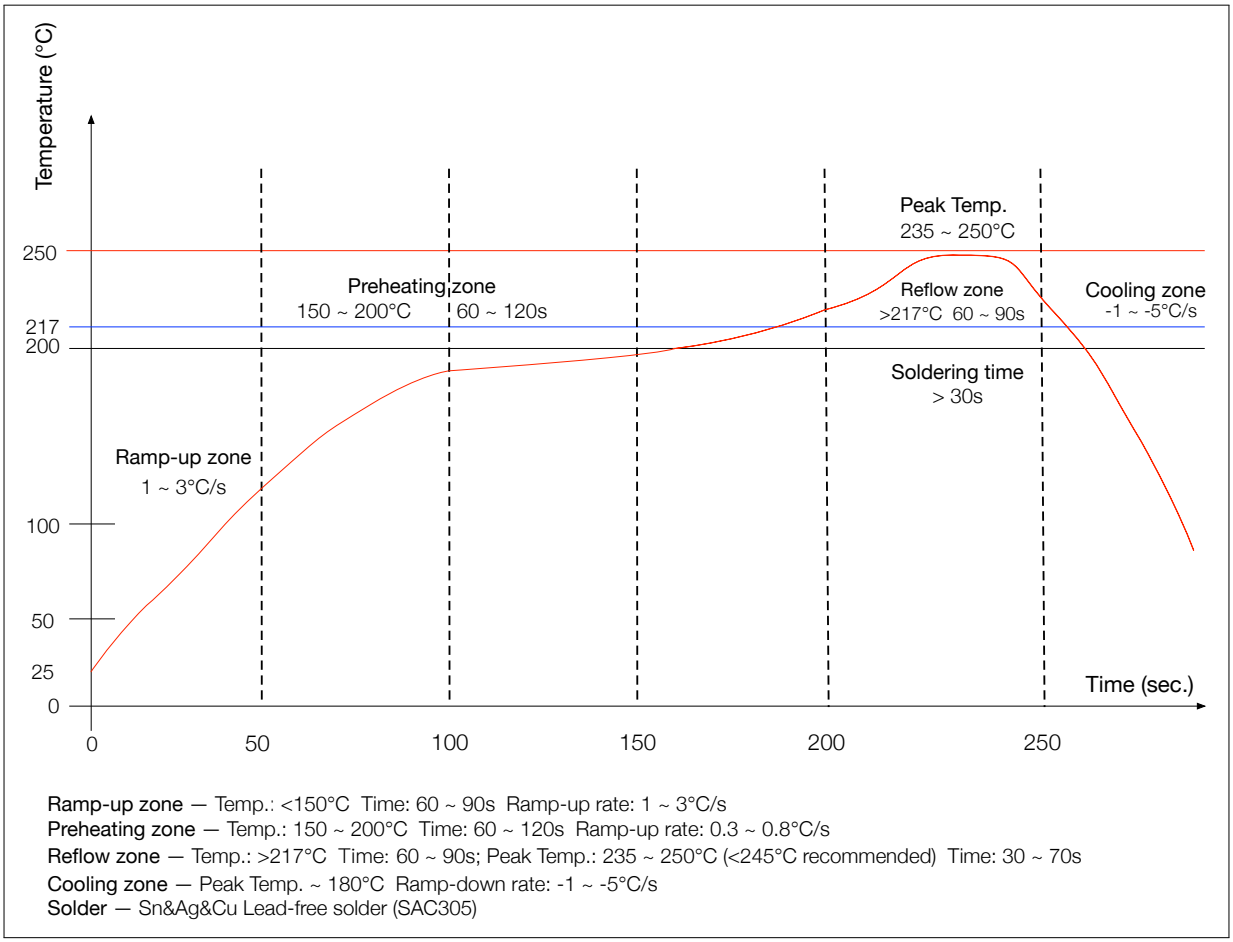
Parameter	Conditions	Min	Typ	Max	Unit
Sensitivity @30.8% PER	-	-	–97	-	dBm
Maximum received signal @30.8% PER	-	0	-	-	dBm
Co-channel C/I	-	-	+10	-	dB
Adjacent channel selectivity C/I	F = F0 + 1 MHz	-	–5	-	dB
	F = F0 – 1 MHz	-	–5	-	dB
	F = F0 + 2 MHz	-	–25	-	dB
	F = F0 – 2 MHz	-	–35	-	dB
	F = F0 + 3 MHz	-	–25	-	dB
	F = F0 – 3 MHz	-	–45	-	dB
Out-of-band blocking performance	30 MHz ~ 2000 MHz	–10	-	-	dBm
	2000 MHz ~ 2400 MHz	–27	-	-	dBm
	2500 MHz ~ 3000 MHz	–27	-	-	dBm
	3000 MHz ~ 12.5 GHz	–10	-	-	dBm
Intermodulation	-	–36	-	-	dBm

5.5.2 Transmitter

Table 9: Transmitter Characteristics – BLE

Parameter	Conditions	Min	Typ	Max	Unit
RF transmit power	-	-	0	-	dBm
Gain control step	-	-	3	-	dBm
RF power control range	-	–12	-	+9	dBm
Adjacent channel transmit power	F = F0 ± 2 MHz	-	–52	-	dBm
	F = F0 ± 3 MHz	-	–58	-	dBm
	F = F0 ± > 3 MHz	-	–60	-	dBm
$\Delta f1_{avg}$	-	-	-	265	kHz
$\Delta f2_{max}$	-	247	-	-	kHz
$\Delta f2_{avg}/\Delta f1_{avg}$	-	-	–0.92	-	-
ICFT	-	-	–10	-	kHz
Drift rate	-	-	0.7	-	kHz/50 μs
Drift	-	-	2	-	kHz

5.6 Reflow Profile



Ramp-up zone — Temp.: <150°C Time: 60 ~ 90s Ramp-up rate: 1 ~ 3°C/s
Preheating zone — Temp.: 150 ~ 200°C Time: 60 ~ 120s Ramp-up rate: 0.3 ~ 0.8°C/s
Reflow zone — Temp.: >217°C Time: 60 ~ 90s; Peak Temp.: 235 ~ 250°C (<245°C recommended) Time: 30 ~ 70s
Cooling zone — Peak Temp. ~ 180°C Ramp-down rate: -1 ~ -5°C/s
Solder — Sn&Ag&Cu Lead-free solder (SAC305)

Figure 2: Reflow Profile

6. Schematics

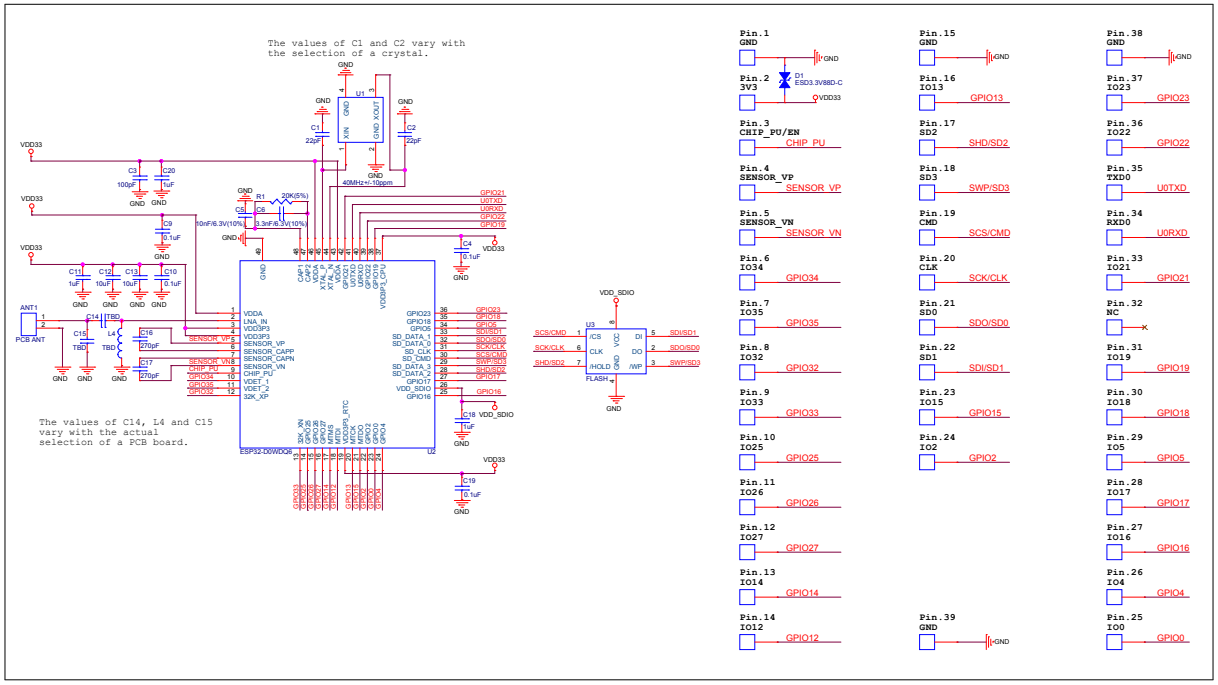


Figure 3: ESP32-WROOM-32 Schematics

7. Peripheral Schematics

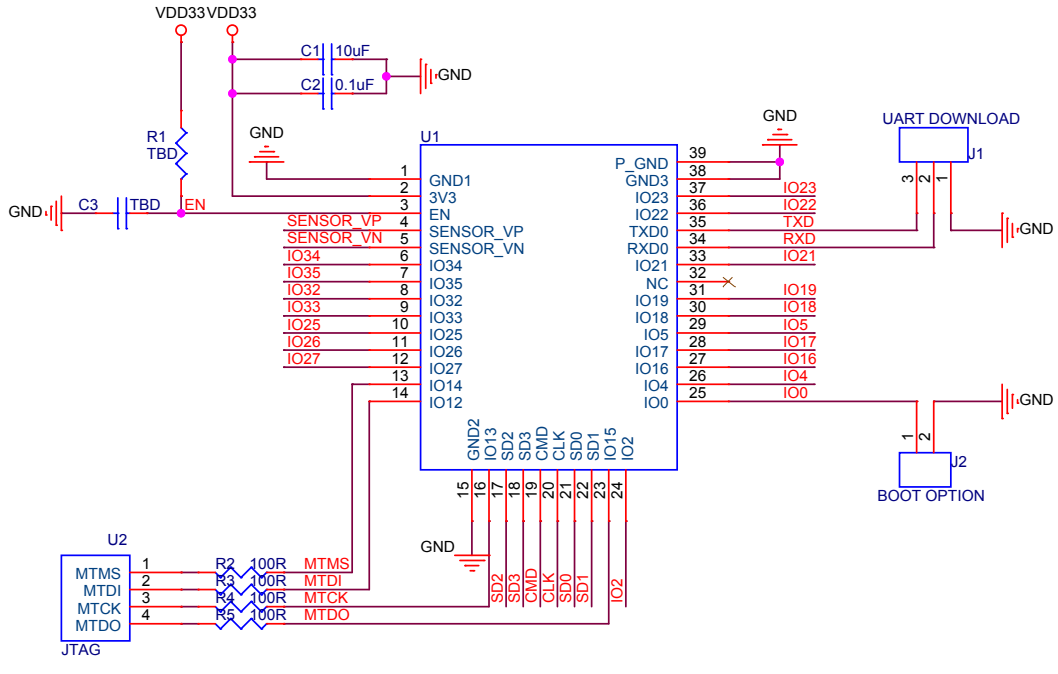


Figure 4: ESP32-WROOM-32 Peripheral Schematics

Note:

- Soldering Pad 39 to the Ground of the base board is not necessary for a satisfactory thermal performance. If users do want to solder it, they need to ensure that the correct quantity of soldering paste is applied.
- To ensure the power supply to the ESP32 chip during power-up, it is advised to add an RC delay circuit at the EN pin. The recommended setting for the RC delay circuit is usually R = 10 kΩ and C = 0.1 μF. However, specific parameters should be adjusted based on the power-up timing of the module and the power-up and reset sequence timing of the chip. For ESP32's power-up and reset sequence timing diagram, please refer to Section *Power Scheme* in *ESP32 Datasheet*.

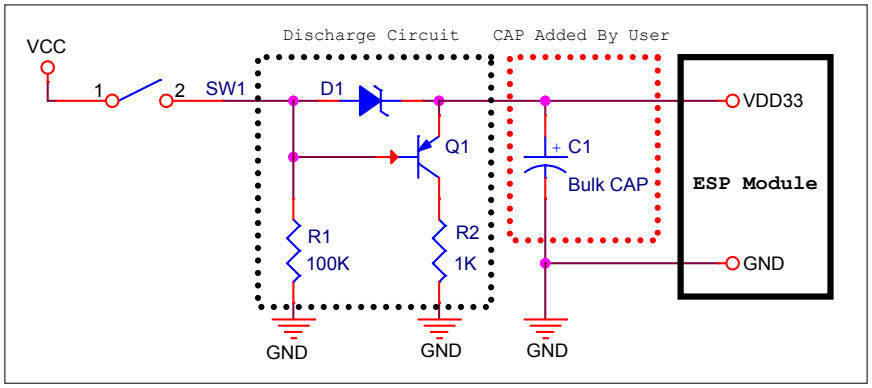


Figure 5: Discharge Circuit for VDD33 Rail

Note:

The discharge circuit can be applied in scenarios where ESP32 is powered on and off repeatedly by switching the power rails, and there is a large capacitor on the VDD33 rail. For details, please refer to Section *Power Scheme* in *ESP32 Datasheet*.

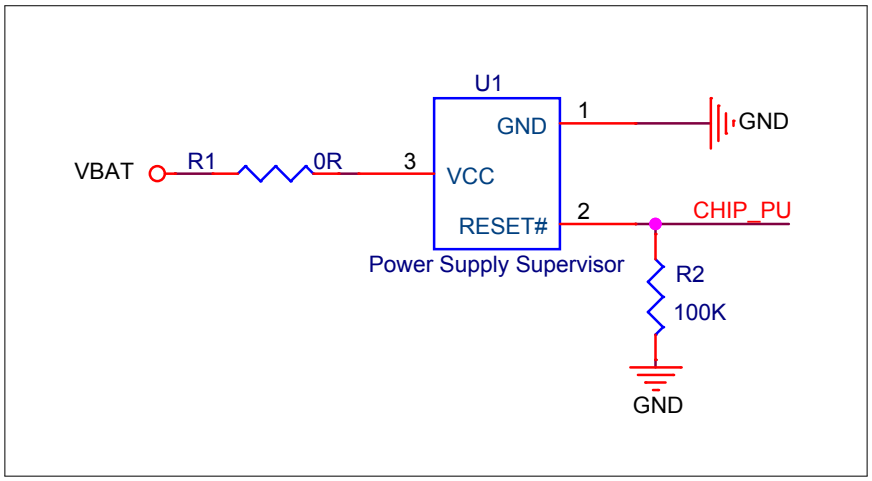


Figure 6: Reset Circuit

Note:

When battery is used as the power supply for ESP32 series of chips and modules, a supply voltage supervisor is recommended to avoid boot failure due to low voltage. Users are recommended to pull CHIP_PU low if the power supply for ESP32 is below 2.3 V.

8. Physical Dimensions

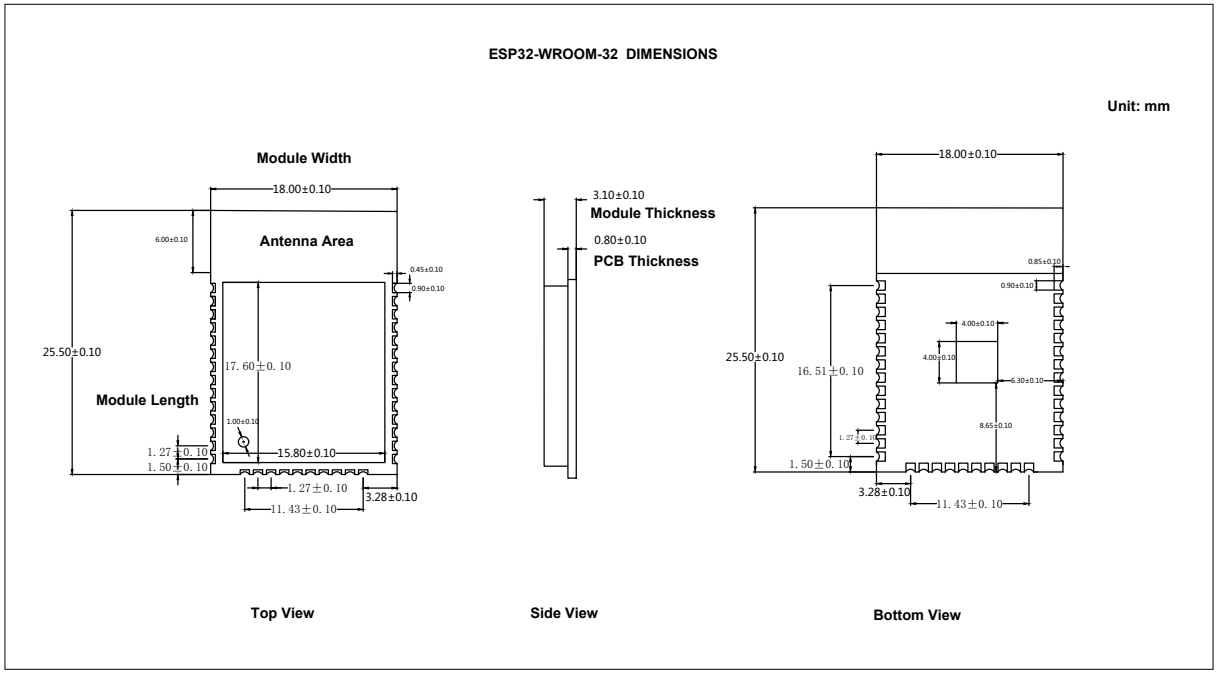


Figure 7: Physical Dimensions of ESP32-WROOM-32

9. Recommended PCB Land Pattern

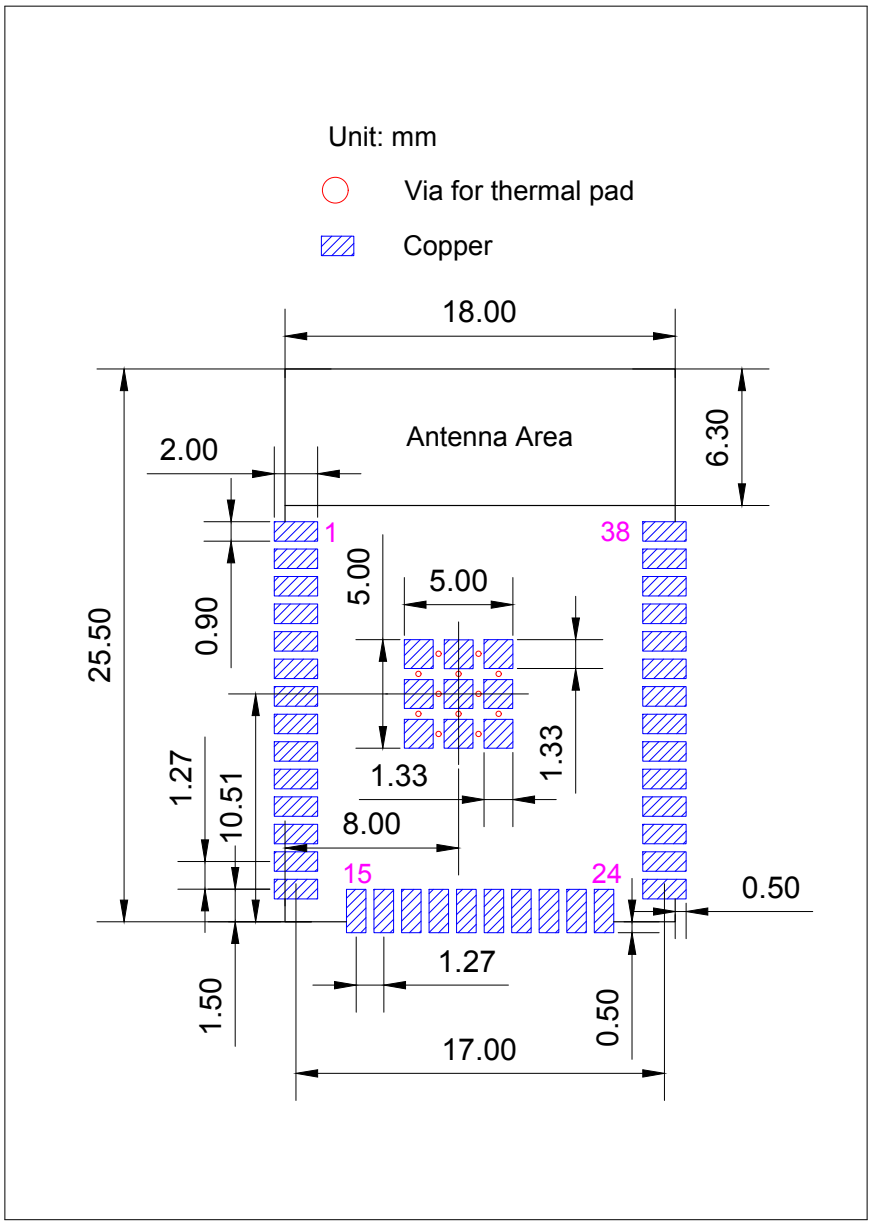


Figure 8: Recommended PCB Land Pattern

10. Learning Resources

10.1 Must-Read Documents

The following link provides documents related to ESP32.

- *ESP32 Datasheet*
  This document provides an introduction to the specifications of the ESP32 hardware, including overview, pin definitions, functional description, peripheral interface, electrical characteristics, etc.
- *ESP-IDF Programming Guide*
  It hosts extensive documentation for ESP-IDF ranging from hardware guides to API reference.
- *ESP32 Technical Reference Manual*
  The manual provides detailed information on how to use the ESP32 memory and peripherals.
- ESP32 Hardware Resources
  The zip files include the schematics, PCB layout, Gerber and BOM list of ESP32 modules and development boards.
- *ESP32 Hardware Design Guidelines*
  The guidelines outline recommended design practices when developing standalone or add-on systems based on the ESP32 series of products, including the ESP32 chip, the ESP32 modules and development boards.
- *ESP32 AT Instruction Set and Examples*
  This document introduces the ESP32 AT commands, explains how to use them, and provides examples of several common AT commands.
- *Espressif Products Ordering Information*

10.2 Must-Have Resources

Here are the ESP32-related must-have resources.

- ESP32 BBS
  This is an Engineer-to-Engineer (E2E) Community for ESP32 where you can post questions, share knowledge, explore ideas, and help solve problems with fellow engineers.
- ESP32 GitHub
  ESP32 development projects are freely distributed under Espressif's MIT license on GitHub. It is established to help developers get started with ESP32 and foster innovation and the growth of general knowledge about the hardware and software surrounding ESP32 devices.
- ESP32 Tools
  This is a webpage where users can download ESP32 Flash Download Tools and the zip file "ESP32 Certification and Test".
- ESP-IDF
  This webpage links users to the official IoT development framework for ESP32.
- ESP32 Resources
  This webpage provides the links to all available ESP32 documents, SDK and tools.

Revision History

Date	Version	Release notes
2019.09	V2.9	• Changed the supply voltage range from 2.7 V ~ 3.6 V to 3.0 V ~ 3.6 V; • Added Moisture sensitivity level (MSL) 3 in Table 1 *ESP32-WROOM-32 Specifications*; • Added notes about "Operating frequency range" and "TX power" under Table 7 *Wi-Fi Radio Characteristics*; • Updated Section 7 *Peripheral Schematics* and added a note about RC delay circuit under it; • Updated Figure 8 *Recommended PCB Land Pattern*.
2019.01	V2.8	Changed the RF power control range in Table 9 from –12 ~ +12 to –12 ~ +9 dBm.
2018.10	V2.7	Added "Cumulative IO output current" entry to Table 4: Absolute Maximum Ratings; Added more parameters to Table 6: DC Characteristics.
2018.08	V2.6	• Added reliability test items the module has passed in Table 1: ESP32-WROOM-32 Specifications, and removed software-specific information; • Updated section 3.4: RTC and Low-Power Management; • Changed the module's dimensions from (18±0.2) mm x (25.5 ±0.2) mm x (3.1±0.15) mm to (18.00±0.10) mm x (25.50±0.10) mm x (3.10±0.10) mm; • Updated Figure 8: Physical Dimensions; • Updated Table 7: Wi-Fi Radio.
2018.06	V2.5	• Changed the module name to ESP32-WROOM-32; • Deleted Temperature Sensor in Table 1: ESP32-WROOM-32 Specifications; • Updated Chapter 3: Functional Description; • Added Chapter 8: Recommended PCB Land Pattern; Changes to electrical characteristics: • Updated Table 4: Absolute Maximum Ratings; • Added Table 5: Recommended Operating Conditions; • Added Table 6: DC Characteristics; • Updated the values of "Gain control step", "Adjacent channel transmit power" in Table 9: Transmitter Characteristics - BLE.
2018.03	V2.4	Updated Table 1 in Chapter 1.
2018.01	V2.3	Deleted information on LNA pre-amplifier; Updated section 3.4 RTC and Low-Power Management; Added reset circuit in Chapter 7 and a note to it.
2017.10	V2.2	Updated the description of the chip's system reset in Section 2.3 Strapping Pins; Deleted "Association sleep pattern" in Table "Power Consumption by Power Modes" and added notes to Active sleep and Modem-sleep; Updated the note to Figure 4 Peripheral Schematics; Added discharge circuit for VDD33 rail in Chapter 7 and a note to it.
2017.09	V2.1	Updated operating voltage/power supply range updated to 2.7 ~ 3.6V; Updated Chapter 7.
2017.08	V2.0	Changed the sensitivity of NZIF receiver to -97 dBm in Table 1; Updated the dimensions of the module; Updated Table "Power Consumption by Power Modes" Power Consumption by Power Modes, and added two notes to it;

Date	Version	Release notes
		Updated Table 4, 7, 8, 9; Added Chapter 8; Added the link to certification download.
2017.06	V1.9	Added a note to Section 2.1 Pin Layout; Updated Section 3.3 Crystal Oscillators; Updated Figure 3 ESP-WROOM-32 Schematics; Added Documentation Change Notification.
2017.05	V1.8	Updated Figure 1 Top and Side View of ESP32-WROOM-32 (ESP-WROOM-32).
2017.04	V1.7	Added the module's dimensional tolerance; Changed the input impedance value of 50Ω in Table 7 Wi-Fi Radio Characteristics to output impedance value of 30+j10 Ω.
2017.04	V1.6	Added Figure 2 Reflow Profile.
2017.03	V1.5	Updated Section 2.2 Pin Description; Updated Section 3.2 External Flash and SRAM; Updated Section 4 Peripherals and Sensors Description.
2017.03	V1.4	Updated Chapter 1 Preface; Updated Chapter 2 Pin Definitions; Updated Chapter 3 Functional Description; Updated Table Recommended Operating Conditions; Updated Table 7 Wi-Fi Radio Characteristics; Updated Section 5.6 Reflow Profile; Added Chapter 10 Learning Resources.
2016.12	V1.3	Updated Section 2.1 Pin Layout.
2016.11	V1.2	Added Figure 7 Peripheral Schematics.
2016.11	V1.1	Updated Chapter 6 Schematics.
2016.08	V1.0	First release.

MPU-6000 and MPU-6050
Product Specification
Revision 3.4

CONTENTS

1 REVISION HISTORY 5
2 PURPOSE AND SCOPE 6
3 PRODUCT OVERVIEW 7
3.1 MPU-60X0 OVERVIEW 7
4 APPLICATIONS 9
5 FEATURES 10
5.1 GYROSCOPE FEATURES 10
5.2 ACCELEROMETER FEATURES 10
5.3 ADDITIONAL FEATURES 10
5.4 MOTIONPROCESSING 11
5.5 CLOCKING 11
6 ELECTRICAL CHARACTERISTICS 12
6.1 GYROSCOPE SPECIFICATIONS 12
6.2 ACCELEROMETER SPECIFICATIONS 13
6.3 ELECTRICAL AND OTHER COMMON SPECIFICATIONS 14
6.4 ELECTRICAL SPECIFICATIONS, CONTINUED 15
6.5 ELECTRICAL SPECIFICATIONS, CONTINUED 16
6.6 ELECTRICAL SPECIFICATIONS, CONTINUED 17
6.7 I^2C TIMING CHARACTERIZATION 18
6.8 SPI TIMING CHARACTERIZATION (MPU-6000 ONLY) 19
6.9 ABSOLUTE MAXIMUM RATINGS 20
7 APPLICATIONS INFORMATION 21
7.1 PIN OUT AND SIGNAL DESCRIPTION 21
7.2 TYPICAL OPERATING CIRCUIT 22
7.3 BILL OF MATERIALS FOR EXTERNAL COMPONENTS 22
7.4 RECOMMENDED POWER-ON PROCEDURE 23
7.5 BLOCK DIAGRAM 24
7.6 OVERVIEW 24
7.7 THREE-AXIS MEMS GYROSCOPE WITH 16-BIT ADCS AND SIGNAL CONDITIONING 25
7.8 THREE-AXIS MEMS ACCELEROMETER WITH 16-BIT ADCS AND SIGNAL CONDITIONING 25
7.9 DIGITAL MOTION PROCESSOR 25
7.10 PRIMARY I^2C AND SPI SERIAL COMMUNICATIONS INTERFACES 25
7.11 AUXILIARY I^2C SERIAL INTERFACE 26

7.12 SELF-TEST 27
7.13 MPU-60X0 SOLUTION FOR 9-AXIS SENSOR FUSION USING I2C INTERFACE 28
7.14 MPU-6000 USING SPI INTERFACE 29
7.15 INTERNAL CLOCK GENERATION 30
7.16 SENSOR DATA REGISTERS 30
7.17 FIFO 30
7.18 INTERRUPTS 30
7.19 DIGITAL-OUTPUT TEMPERATURE SENSOR 31
7.20 BIAS AND LDO 31
7.21 CHARGE PUMP 31
8 PROGRAMMABLE INTERRUPTS 32
9 DIGITAL INTERFACE 33
9.1 I2C AND SPI (MPU-6000 ONLY) SERIAL INTERFACES 33
9.2 I2C INTERFACE 33
9.3 I2C COMMUNICATIONS PROTOCOL 33
9.4 I2C TERMS 36
9.5 SPI INTERFACE (MPU-6000 ONLY) 37
10 SERIAL INTERFACE CONSIDERATIONS (MPU-6050) 38
10.1 MPU-6050 SUPPORTED INTERFACES 38
10.2 LOGIC LEVELS 38
10.3 LOGIC LEVELS DIAGRAM FOR AUX_*VDDIO* = 0 39
11 ASSEMBLY 40
11.1 ORIENTATION OF AXES 40
11.2 PACKAGE DIMENSIONS 41
11.3 PCB DESIGN GUIDELINES 42
11.4 ASSEMBLY PRECAUTIONS 43
11.5 STORAGE SPECIFICATIONS 46
11.6 PACKAGE MARKING SPECIFICATION 46
11.7 TAPE & REEL SPECIFICATION 47
11.8 LABEL 48
11.9 PACKAGING 49
11.10 REPRESENTATIVE SHIPPING CARTON LABEL 50
12 RELIABILITY 51
12.1 QUALIFICATION TEST POLICY 51

12.2 QUALIFICATION TEST PLAN 51

13 ENVIRONMENTAL COMPLIANCE.....52

1 Revision History

Revision Date	Revision	Description
11/24/2010	1.0	Initial Release
05/19/2011	2.0	For Rev C parts. Clarified wording in sections (3.2, 5.1, 5.2, 6.1-6.4, 6.6, 6.9, 7, 7.1-7.6, 7.11, 7.12, 7.14, 8, 8.2-8.4, 10.3, 10.4, 11, 12.2)
07/28/2011	2.1	Edited supply current numbers for different modes (section 6.4)
08/05/2011	2.2	Unit of measure for accelerometer sensitivity changed from LSB/m*g* to LSB/*g*
10/12/2011	2.3	Updated accelerometer self test specifications in Table 6.2. Updated package dimensions (section 11.2). Updated PCB design guidelines (section 11.3)
10/18/2011	3.0	For Rev D parts. Updated accelerometer specifications in Table 6.2. Updated accelerometer specification note (sections 8.2, 8.3, & 8.4). Updated qualification test plan (section 12.2).
10/24/2011	3.1	Edits for clarity Changed operating voltage range to 2.375V-3.46V Added accelerometer Intelligence Function increment value of 1m*g*/LSB (Section 6.2) Updated absolute maximum rating for acceleration (any axis, unpowered) from 0.3ms to 0.2ms (Section 6.9) Modified absolute maximum rating for Latch-up to Level A and ±100mA (Section 6.9, 12.2)
11/16/2011	3.2	**Updated self-test response specifications for Revision D parts dated with date code 1147 (YYWW) or later.** Edits for clarity Added Gyro self-test (sections 5.1, 6.1, 7.6, 7.12) Added Min/Max limits to Accel self-test response (section 6.2) Updated Accelerometer low power mode operating currents (Section 6.3) Added gyro self test to block diagram (section 7.5) Updated packaging labels and descriptions (sections 11.8 & 11.9)
5/16/2012	3.3	Updated Gyro and Accelerometer self test information (sections 6.1, 6.2, 7.12) Updated latch-up information (Section 6.9) Updated programmable interrupts information (Section 8) Changed shipment information from maximum of 3 reels (15K units) per shipper box to 5 reels (25K units) per shipper box (Section 11.7) Updated packing shipping and label information (Sections 11.8, 11.9) Updated reliability references (Section 12.2)
8/19/2013	3.4	Updates section 4

2 Purpose and Scope

This product specification provides advanced information regarding the electrical specification and design related information for the MPU-6000™ and MPU-6050™ MotionTracking™ devices, collectively called the MPU-60X0™ or MPU™.

Electrical characteristics are based upon design analysis and simulation results only. Specifications are subject to change without notice. Final specifications will be updated based upon characterization of production silicon. For references to register map and descriptions of individual registers, please refer to the MPU-6000/MPU-6050 Register Map and Register Descriptions document.

The self-test response specifications provided in this document pertain to Revision D parts with date codes of 1147 (YYWW) or later. Please see Section 11.6 for package marking description details.

3 Product Overview

3.1 MPU-60X0 Overview

MotionInterface™ is becoming a "must-have" function being adopted by smartphone and tablet manufacturers due to the enormous value it adds to the end user experience. In smartphones, it finds use in applications such as gesture commands for applications and phone control, enhanced gaming, augmented reality, panoramic photo capture and viewing, and pedestrian and vehicle navigation. With its ability to precisely and accurately track user motions, MotionTracking technology can convert handsets and tablets into powerful 3D intelligent devices that can be used in applications ranging from health and fitness monitoring to location-based services. Key requirements for MotionInterface enabled devices are small package size, low power consumption, high accuracy and repeatability, high shock tolerance, and application specific performance programmability – all at a low consumer price point.

The MPU-60X0 is the world's first integrated 6-axis MotionTracking device that combines a 3-axis gyroscope, 3-axis accelerometer, and a Digital Motion Processor™ (DMP) all in a small 4x4x0.9mm package. With its dedicated I^2C sensor bus, it directly accepts inputs from an external 3-axis compass to provide a complete 9-axis MotionFusion™ output. The MPU-60X0 MotionTracking device, with its 6-axis integration, on-board MotionFusion™, and run-time calibration firmware, enables manufacturers to eliminate the costly and complex selection, qualification, and system level integration of discrete devices, guaranteeing optimal motion performance for consumers. The MPU-60X0 is also designed to interface with multiple non-inertial digital sensors, such as pressure sensors, on its auxiliary I^2C port. The MPU-60X0 is footprint compatible with the MPU-30X0 family.

The MPU-60X0 features three 16-bit analog-to-digital converters (ADCs) for digitizing the gyroscope outputs and three 16-bit ADCs for digitizing the accelerometer outputs. For precision tracking of both fast and slow motions, the parts feature a user-programmable gyroscope full-scale range of ±250, ±500, ±1000, and ±2000°/sec (dps) and a user-programmable accelerometer full-scale range of ±2*g*, ±4*g*, ±8*g*, and ±16*g*.

An on-chip 1024 Byte FIFO buffer helps lower system power consumption by allowing the system processor to read the sensor data in bursts and then enter a low-power mode as the MPU collects more data. With all the necessary on-chip processing and sensor components required to support many motion-based use cases, the MPU-60X0 uniquely enables low-power MotionInterface applications in portable applications with reduced processing requirements for the system processor. By providing an integrated MotionFusion output, the DMP in the MPU-60X0 offloads the intensive MotionProcessing computation requirements from the system processor, minimizing the need for frequent polling of the motion sensor output.

Communication with all registers of the device is performed using either I^2C at 400kHz or SPI at 1MHz (MPU-6000 only). For applications requiring faster communications, the sensor and interrupt registers may be read using SPI at 20MHz (MPU-6000 only). Additional features include an embedded temperature sensor and an on-chip oscillator with ±1% variation over the operating temperature range.

By leveraging its patented and volume-proven Nasiri-Fabrication platform, which integrates MEMS wafers with companion CMOS electronics through wafer-level bonding, InvenSense has driven the MPU-60X0 package size down to a revolutionary footprint of 4x4x0.9mm (QFN), while providing the highest performance, lowest noise, and the lowest cost semiconductor packaging required for handheld consumer electronic devices. The part features a robust 10,000*g* shock tolerance, and has programmable low-pass filters for the gyroscopes, accelerometers, and the on-chip temperature sensor.

For power supply flexibility, the MPU-60X0 operates from VDD power supply voltage range of 2.375V-3.46V. Additionally, the MPU-6050 provides a VLOGIC reference pin (in addition to its analog supply pin: VDD), which sets the logic levels of its I^2C interface. The VLOGIC voltage may be 1.8V±5% or VDD.

The MPU-6000 and MPU-6050 are identical, except that the MPU-6050 supports the I^2C serial interface only, and has a separate VLOGIC reference pin. The MPU-6000 supports both I^2C and SPI interfaces and has a single supply pin, VDD, which is both the device's logic reference supply and the analog supply for the part. The table below outlines these differences:

Primary Differences between MPU-6000 and MPU-6050

Part / Item	MPU-6000	MPU-6050
VDD	2.375V-3.46V	2.375V-3.46V
VLOGIC	n/a	1.71V to VDD
Serial Interfaces Supported	I^2C, SPI	I^2C
Pin 8	/CS	VLOGIC
Pin 9	AD0/SDO	AD0
Pin 23	SCL/SCLK	SCL
Pin 24	SDA/SDI	SDA

4 Applications

- *BlurFree*™ technology (for Video/Still Image Stabilization)
- *AirSign*™ technology (for Security/Authentication)
- *TouchAnywhere*™ technology (for “no touch” UI Application Control/Navigation)
- *MotionCommand*™ technology (for Gesture Short-cuts)
- Motion-enabled game and application framework
- InstantGesture™ iG™ gesture recognition
- Location based services, points of interest, and dead reckoning
- Handset and portable gaming
- Motion-based game controllers
- 3D remote controls for Internet connected DTVs and set top boxes, 3D mice
- Wearable sensors for health, fitness and sports
- Toys

5 Features

5.1 Gyroscope Features

The triple-axis MEMS gyroscope in the MPU-60X0 includes a wide range of features:

- Digital-output X-, Y-, and Z-Axis angular rate sensors (gyroscopes) with a user-programmable full-scale range of ±250, ±500, ±1000, and ±2000°/sec
- External sync signal connected to the FSYNC pin supports image, video and GPS synchronization
- Integrated 16-bit ADCs enable simultaneous sampling of gyros
- Enhanced bias and sensitivity temperature stability reduces the need for user calibration
- Improved low-frequency noise performance
- Digitally-programmable low-pass filter
- Gyroscope operating current: 3.6mA
- Standby current: 5µA
- Factory calibrated sensitivity scale factor
- User self-test

5.2 Accelerometer Features

The triple-axis MEMS accelerometer in MPU-60X0 includes a wide range of features:

- Digital-output triple-axis accelerometer with a programmable full scale range of ±2*g*, ±4*g*, ±8*g* and ±16*g*
- Integrated 16-bit ADCs enable simultaneous sampling of accelerometers while requiring no external multiplexer
- Accelerometer normal operating current: 500µA
- Low power accelerometer mode current: 10µA at 1.25Hz, 20µA at 5Hz, 60µA at 20Hz, 110µA at 40Hz
- Orientation detection and signaling
- Tap detection
- User-programmable interrupts
- High-G interrupt
- User self-test

5.3 Additional Features

The MPU-60X0 includes the following additional features:

- 9-Axis MotionFusion by the on-chip Digital Motion Processor (DMP)
- Auxiliary master I^2C bus for reading data from external sensors (e.g., magnetometer)
- 3.9mA operating current when all 6 motion sensing axes and the DMP are enabled
- VDD supply voltage range of 2.375V-3.46V
- Flexible VLOGIC reference voltage supports multiple I^2C interface voltages (MPU-6050 only)
- Smallest and thinnest QFN package for portable devices: 4x4x0.9mm
- Minimal cross-axis sensitivity between the accelerometer and gyroscope axes
- 1024 byte FIFO buffer reduces power consumption by allowing host processor to read the data in bursts and then go into a low-power mode as the MPU collects more data
- Digital-output temperature sensor
- User-programmable digital filters for gyroscope, accelerometer, and temp sensor
- 10,000 *g* shock tolerant
- 400kHz Fast Mode I^2C for communicating with all registers
- 1MHz SPI serial interface for communicating with all registers (MPU-6000 only)
- 20MHz SPI serial interface for reading sensor and interrupt registers (MPU-6000 only)

- MEMS structure hermetically sealed and bonded at wafer level
- RoHS and Green compliant

5.4 MotionProcessing

- Internal Digital Motion Processing™ (DMP™) engine supports 3D MotionProcessing and gesture recognition algorithms
- The MPU-60X0 collects gyroscope and accelerometer data while synchronizing data sampling at a user defined rate. The total dataset obtained by the MPU-60X0 includes 3-Axis gyroscope data, 3-Axis accelerometer data, and temperature data. The MPU's calculated output to the system processor can also include heading data from a digital 3-axis third party magnetometer.
- The FIFO buffers the complete data set, reducing timing requirements on the system processor by allowing the processor burst read the FIFO data. After burst reading the FIFO data, the system processor can save power by entering a low-power sleep mode while the MPU collects more data.
- Programmable interrupt supports features such as gesture recognition, panning, zooming, scrolling, tap detection, and shake detection
- Digitally-programmable low-pass filters
- Low-power pedometer functionality allows the host processor to sleep while the DMP maintains the step count.

5.5 Clocking

- On-chip timing generator ±1% frequency variation over full temperature range
- Optional external clock inputs of 32.768kHz or 19.2MHz

6 Electrical Characteristics

6.1 Gyroscope Specifications

VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
GYROSCOPE SENSITIVITY						
Full-Scale Range	FS_SEL=0		±250		º/s	
	FS_SEL=1		±500		º/s	
	FS_SEL=2		±1000		º/s	
	FS_SEL=3		±2000		º/s	
Gyroscope ADC Word Length			16		bits	
Sensitivity Scale Factor	FS_SEL=0		131		LSB/(º/s)	
	FS_SEL=1		65.5		LSB/(º/s)	
	FS_SEL=2		32.8		LSB/(º/s)	
	FS_SEL=3		16.4		LSB/(º/s)	
Sensitivity Scale Factor Tolerance	25°C	-3		+3	%	
Sensitivity Scale Factor Variation Over Temperature			±2		%	
Nonlinearity	Best fit straight line; 25°C		0.2		%	
Cross-Axis Sensitivity			±2		%	
GYROSCOPE ZERO-RATE OUTPUT (ZRO)						
Initial ZRO Tolerance	25°C		±20		º/s	
ZRO Variation Over Temperature	-40°C to +85°C		±20		º/s	
Power-Supply Sensitivity (1-10Hz)	Sine wave, 100mVpp; VDD=2.5V		0.2		º/s	
Power-Supply Sensitivity (10 - 250Hz)	Sine wave, 100mVpp; VDD=2.5V		0.2		º/s	
Power-Supply Sensitivity (250Hz - 100kHz)	Sine wave, 100mVpp; VDD=2.5V		4		º/s	
Linear Acceleration Sensitivity	Static		0.1		º/s/*g*	
SELF-TEST RESPONSE						
Relative	Change from factory trim	-14		14	%	1
GYROSCOPE NOISE PERFORMANCE	**FS_SEL=0**					
Total RMS Noise	DLPFCFG=2 (100Hz)		0.05		º/s-rms	
Low-frequency RMS noise	Bandwidth 1Hz to10Hz		0.033		º/s-rms	
Rate Noise Spectral Density	At 10Hz		0.005		º/s/ √ Hz	
GYROSCOPE MECHANICAL FREQUENCIES						
X-Axis		30	33	36	kHz	
Y-Axis		27	30	33	kHz	
Z-Axis		24	27	30	kHz	
LOW PASS FILTER RESPONSE						
	Programmable Range	5		256	Hz	
OUTPUT DATA RATE						
	Programmable	4		8,000	Hz	
GYROSCOPE START-UP TIME	**DLPFCFG=0**					
ZRO Settling (from power-on)	to ±1º/s of Final		30		ms	

1. Please refer to the following document for further information on Self-Test: *MPU-6000/MPU-6050 Register Map and Descriptions*

6.2 Accelerometer Specifications

VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS	NOTES
ACCELEROMETER SENSITIVITY						
Full-Scale Range	AFS_SEL=0		±2		*g*	
	AFS_SEL=1		±4		*g*	
	AFS_SEL=2		±8		*g*	
	AFS_SEL=3		±16		*g*	
ADC Word Length	Output in two's complement format		16		bits	
Sensitivity Scale Factor	AFS_SEL=0		16,384		LSB/*g*	
	AFS_SEL=1		8,192		LSB/*g*	
	AFS_SEL=2		4,096		LSB/*g*	
	AFS_SEL=3		2,048		LSB/*g*	
Initial Calibration Tolerance			±3		%	
Sensitivity Change vs. Temperature	AFS_SEL=0, -40°C to +85°C		±0.02		%/°C	
Nonlinearity	Best Fit Straight Line		0.5		%	
Cross-Axis Sensitivity			±2		%	
ZERO-G OUTPUT						
Initial Calibration Tolerance	X and Y axes		±50		m*g*	1
	Z axis		±80		m*g*	
Zero-G Level Change vs. Temperature	X and Y axes, 0°C to +70°C		±35			
	Z axis, 0°C to +70°C		±60		m*g*	
SELF TEST RESPONSE						
Relative	Change from factory trim	-14		14	%	2
NOISE PERFORMANCE						
Power Spectral Density	@10Hz, AFS_SEL=0 & ODR=1kHz		400		µ*g*/ √ Hz	
LOW PASS FILTER RESPONSE						
	Programmable Range	5		260	Hz	
OUTPUT DATA RATE						
	Programmable Range	4		1,000	Hz	
INTELLIGENCE FUNCTION INCREMENT			32		mg/LSB	

1. Typical zero-g initial calibration tolerance value after MSL3 preconditioning
2. Please refer to the following document for further information on Self-Test: *MPU-6000/MPU-6050 Register Map and Descriptions*

6.3 Electrical and Other Common Specifications

VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	Units	Notes
TEMPERATURE SENSOR						
Range			-40 to +85		°C	
Sensitivity	Untrimmed		340		LSB/ºC	
Temperature Offset	35°C		-521		LSB	
Linearity	Best fit straight line (-40°C to +85°C)		±1		°C	
VDD POWER SUPPLY						
Operating Voltages		2.375		3.46	V	
Normal Operating Current	Gyroscope + Accelerometer + DMP		3.9		mA	
	Gyroscope + Accelerometer (DMP disabled)		3.8		mA	
	Gyroscope + DMP (Accelerometer disabled)		3.7		mA	
	Gyroscope only (DMP & Accelerometer disabled)		3.6		mA	
	Accelerometer only (DMP & Gyroscope disabled)		500		µA	
Accelerometer Low Power Mode Current	1.25 Hz update rate		10		µA	
	5 Hz update rate		20		µA	
	20 Hz update rate		70		µA	
	40 Hz update rate		140		µA	
Full-Chip Idle Mode Supply Current			5		µA	
Power Supply Ramp Rate	Monotonic ramp. Ramp rate is 10% to 90% of the final value			100	ms	
VLOGIC REFERENCE VOLTAGE	MPU-6050 only					
Voltage Range	VLOGIC must be ≤VDD at all times	1.71		VDD	V	
Power Supply Ramp Rate	Monotonic ramp. Ramp rate is 10% to 90% of the final value			3	ms	
Normal Operating Current			100		µA	
TEMPERATURE RANGE						
Specified Temperature Range	Performance parameters are not applicable beyond Specified Temperature Range	-40		+85	°C	

6.4 Electrical Specifications, Continued

VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

PARAMETER	CONDITIONS	MIN	TYP	MAX	Units	Notes
SERIAL INTERFACE						
SPI Operating Frequency, All Registers Read/Write	MPU-6000 only, Low Speed Characterization		100 ±10%		kHz	
	MPU-6000 only, High Speed Characterization		1 ±10%		MHz	
SPI Operating Frequency, Sensor and Interrupt Registers Read Only	MPU-6000 only		20 ±10%		MHz	
I^2C Operating Frequency	All registers, Fast-mode			400	kHz	
	All registers, Standard-mode			100	kHz	
I^2C ADDRESS	AD0 = 0		1101000			
	AD0 = 1		1101001			
DIGITAL INPUTS (SDI/SDA, AD0, SCLK/SCL, FSYNC, /CS, CLKIN)						
V_{IH}, High Level Input Voltage	MPU-6000	0.7*VDD			V	
	MPU-6050	0.7*VLOGIC			V	
V_{IL}, Low Level Input Voltage	MPU-6000			0.3*VDD	V	
	MPU-6050			0.3*VLOGIC	V	
C_I, Input Capacitance			< 5		pF	
DIGITAL OUTPUT (SDO, INT)						
V_{OH}, High Level Output Voltage	R_{LOAD}=1MΩ; MPU-6000	0.9*VDD			V	
	R_{LOAD}=1MΩ; MPU-6050	0.9*VLOGIC			V	
V_{OL1}, LOW-Level Output Voltage	R_{LOAD}=1MΩ; MPU-6000			0.1*VDD	V	
	R_{LOAD}=1MΩ; MPU-6050			0.1*VLOGIC	V	
$V_{OL.INT1}$, INT Low-Level Output Voltage	OPEN=1, 0.3mA sink Current			0.1	V	
Output Leakage Current	OPEN=1		100		nA	
t_{INT}, INT Pulse Width	LATCH_INT_EN=0		50		µs	

6.5 Electrical Specifications, Continued

Typical Operating Circuit of Section 7.2, VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

Parameters	Conditions	Typical	Units	Notes
Primary I^2C I/O (SCL, SDA)				
V_{IL}, LOW-Level Input Voltage	MPU-6000	-0.5 to 0.3*VDD	V	
V_{IH}, HIGH-Level Input Voltage	MPU-6000	0.7*VDD to VDD + 0.5V	V	
V_{hys}, Hysteresis	MPU-6000	0.1*VDD	V	
VIL, LOW Level Input Voltage	MPU-6050	-0.5V to 0.3*VLOGIC	V	
VIH, HIGH-Level Input Voltage	MPU-6050	0.7*VLOGIC to VLOGIC + 0.5V	V	
Vhys, Hysteresis	MPU-6050	0.1*VLOGIC	V	
V_{OL1}, LOW-Level Output Voltage	3mA sink current	0 to 0.4	V	
I_{OL}, LOW-Level Output Current	V_{OL} = 0.4V	3	mA	
	V_{OL} = 0.6V	5	mA	
Output Leakage Current		100	nA	
t_{of}, Output Fall Time from V_{IHmax} to V_{ILmax}	C_b bus capacitance in pF	20+0.1C_b to 250	ns	
C_I, Capacitance for Each I/O pin		< 10	pF	
Auxiliary I^2C I/O (AUX_CL, AUX_DA)	**MPU-6050: *AUX_VDDIO*=0**			
V_{IL}, LOW-Level Input Voltage		-0.5V to 0.3*VLOGIC	V	
V_{IH}, HIGH-Level Input Voltage		0.7*VLOGIC to VLOGIC + 0.5V	V	
V_{hys}, Hysteresis		0.1*VLOGIC	V	
V_{OL1}, LOW-Level Output Voltage	VLOGIC > 2V; 1mA sink current	0 to 0.4	V	
V_{OL3}, LOW-Level Output Voltage	VLOGIC < 2V; 1mA sink current	0 to 0.2*VLOGIC	V	
I_{OL}, LOW-Level Output Current	V_{OL} = 0.4V	1	mA	
	V_{OL} = 0.6V	1	mA	
Output Leakage Current		100	nA	
t_{of}, Output Fall Time from V_{IHmax} to V_{ILmax}	C_b bus capacitance in pF	20+0.1C_b to 250	ns	
C_I, Capacitance for Each I/O pin		< 10	pF	

6.6 Electrical Specifications, Continued

Typical Operating Circuit of Section 7.2, VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

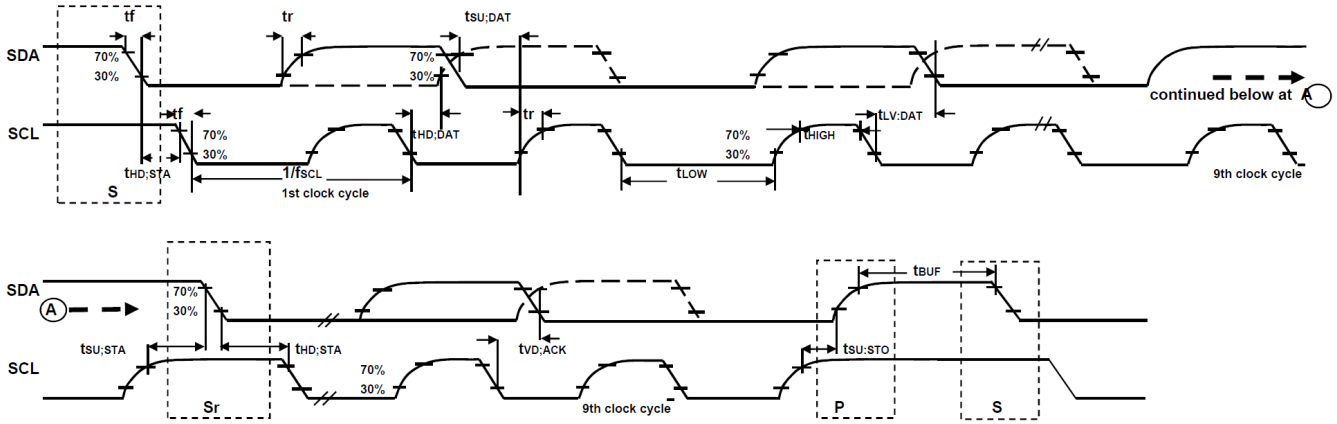
Parameters	Conditions	Min	Typical	Max	Units	Notes
INTERNAL CLOCK SOURCE	**CLK_SEL=0,1,2,3**					
Gyroscope Sample Rate, Fast	DLPFCFG=0 SAMPLERATEDIV = 0		8		kHz	
Gyroscope Sample Rate, Slow	DLPFCFG=1,2,3,4,5, or 6 SAMPLERATEDIV = 0		1		kHz	
Accelerometer Sample Rate			1		kHz	
Clock Frequency Initial Tolerance	CLK_SEL=0, 25°C	-5		+5	%	
	CLK_SEL=1,2,3; 25°C	-1		+1	%	
Frequency Variation over Temperature	CLK_SEL=0		-15 to +10		%	
	CLK_SEL=1,2,3		±1		%	
PLL Settling Time	CLK_SEL=1,2,3		1	10	ms	
EXTERNAL 32.768kHz CLOCK	**CLK_SEL=4**					
External Clock Frequency			32.768		kHz	
External Clock Allowable Jitter	Cycle-to-cycle rms		1 to 2		µs	
Gyroscope Sample Rate, Fast	DLPFCFG=0 SAMPLERATEDIV = 0		8.192		kHz	
Gyroscope Sample Rate, Slow	DLPFCFG=1,2,3,4,5, or 6 SAMPLERATEDIV = 0		1.024		kHz	
Accelerometer Sample Rate			1.024		kHz	
PLL Settling Time			1	10	ms	
EXTERNAL 19.2MHz CLOCK	**CLK_SEL=5**					
External Clock Frequency			19.2		MHz	
Gyroscope Sample Rate	Full programmable range	3.9		8000	Hz	
Gyroscope Sample Rate, Fast Mode	DLPFCFG=0 SAMPLERATEDIV = 0		8		kHz	
Gyroscope Sample Rate, Slow Mode	DLPFCFG=1,2,3,4,5, or 6 SAMPLERATEDIV = 0		1		kHz	
Accelerometer Sample Rate			1		kHz	
PLL Settling Time			1	10	ms	

6.7 I²C Timing Characterization

Typical Operating Circuit of Section 7.2, VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C

Parameters	Conditions	Min	Typical	Max	Units	Notes
I²C TIMING	**I²C FAST-MODE**					
f_{SCL}, SCL Clock Frequency				400	kHz	
$t_{HD.STA}$, (Repeated) START Condition Hold Time		0.6			µs	
t_{LOW}, SCL Low Period		1.3			µs	
t_{HIGH}, SCL High Period		0.6			µs	
$t_{SU.STA}$, Repeated START Condition Setup Time		0.6			µs	
$t_{HD.DAT}$, SDA Data Hold Time		0			µs	
$t_{SU.DAT}$, SDA Data Setup Time		100			ns	
t_r, SDA and SCL Rise Time	C_b bus cap. from 10 to 400pF	20+0.1C_b		300	ns	
t_f, SDA and SCL Fall Time	C_b bus cap. from 10 to 400pF	20+0.1C_b		300	ns	
$t_{SU.STO}$, STOP Condition Setup Time		0.6			µs	
t_{BUF}, Bus Free Time Between STOP and START Condition		1.3			µs	
C_b, Capacitive Load for each Bus Line			< 400		pF	
$t_{VD.DAT}$, Data Valid Time				0.9	µs	
$t_{VD.ACK}$, Data Valid Acknowledge Time				0.9	µs	

Note: Timing Characteristics apply to both Primary and Auxiliary I²C Bus

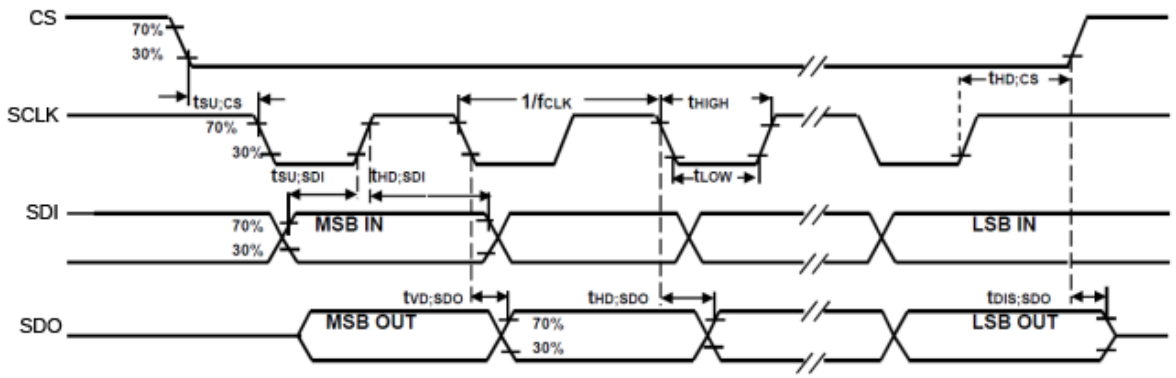


I²C Bus Timing Diagram

6.8 SPI Timing Characterization (MPU-6000 only)

Typical Operating Circuit of Section 7.2, VDD = 2.375V-3.46V, VLOGIC (MPU-6050 only) = 1.8V±5% or VDD, T_A = 25°C, unless otherwise noted.

Parameters	Conditions	Min	Typical	Max	Units	Notes
SPI TIMING						
f_{SCLK}, SCLK Clock Frequency				1	MHz	
t_{LOW}, SCLK Low Period		400			ns	
t_{HIGH}, SCLK High Period		400			ns	
$t_{SU.CS}$, CS Setup Time		8			ns	
$t_{HD.CS}$, CS Hold Time		500			ns	
$t_{SU.SDI}$, SDI Setup Time		11			ns	
$t_{HD.SDI}$, SDI Hold Time		7			ns	
$t_{VD.SDO}$, SDO Valid Time	C_{load} = 20pF			100	ns	
$t_{HD.SDO}$, SDO Hold Time	C_{load} = 20pF	4			ns	
$t_{DIS.SDO}$, SDO Output Disable Time				10	ns	



SPI Bus Timing Diagram

6.9 Absolute Maximum Ratings

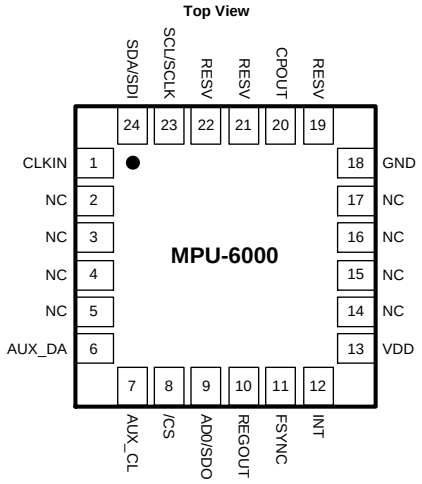
Stress above those listed as "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only and functional operation of the device at these conditions is not implied. Exposure to the absolute maximum ratings conditions for extended periods may affect device reliability.

Parameter	Rating
Supply Voltage, VDD	-0.5V to +6V
VLOGIC Input Voltage Level (MPU-6050)	-0.5V to VDD + 0.5V
REGOUT	-0.5V to 2V
Input Voltage Level (CLKIN, AUX_DA, AD0, FSYNC, INT, SCL, SDA)	-0.5V to VDD + 0.5V
CPOUT (2.5V ≤ VDD ≤ 3.6V)	-0.5V to 30V
Acceleration (Any Axis, unpowered)	10,000g for 0.2ms
Operating Temperature Range	-40°C to +105°C
Storage Temperature Range	-40°C to +125°C
Electrostatic Discharge (ESD) Protection	2kV (HBM); 250V (MM)
Latch-up	JEDEC Class II (2),125°C ±100mA

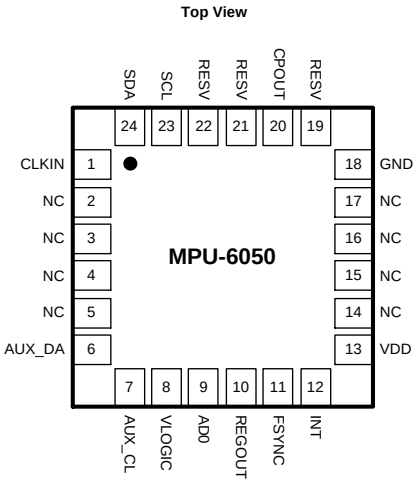
7 Applications Information

7.1 Pin Out and Signal Description

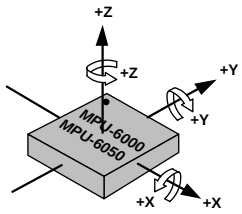
Pin Number	MPU-6000	MPU-6050	Pin Name	Pin Description
1	Y	Y	CLKIN	Optional external reference clock input. Connect to GND if unused.
6	Y	Y	AUX_DA	I^2C master serial data, for connecting to external sensors
7	Y	Y	AUX_CL	I^2C Master serial clock, for connecting to external sensors
8	Y		/CS	SPI chip select (0=SPI mode)
8		Y	VLOGIC	Digital I/O supply voltage
9	Y		AD0 / SDO	I^2C Slave Address LSB (AD0); SPI serial data output (SDO)
9		Y	AD0	I^2C Slave Address LSB (AD0)
10	Y	Y	REGOUT	Regulator filter capacitor connection
11	Y	Y	FSYNC	Frame synchronization digital input. Connect to GND if unused.
12	Y	Y	INT	Interrupt digital output (totem pole or open-drain)
13	Y	Y	VDD	Power supply voltage and Digital I/O supply voltage
18	Y	Y	GND	Power supply ground
19, 21	Y	Y	RESV	Reserved. Do not connect.
20	Y	Y	CPOUT	Charge pump capacitor connection
22	Y	Y	RESV	Reserved. Do not connect.
23	Y		SCL / SCLK	I^2C serial clock (SCL); SPI serial clock (SCLK)
23		Y	SCL	I^2C serial clock (SCL)
24	Y		SDA / SDI	I^2C serial data (SDA); SPI serial data input (SDI)
24		Y	SDA	I^2C serial data (SDA)
2, 3, 4, 5, 14, 15, 16, 17	Y	Y	NC	Not internally connected. May be used for PCB trace routing.



QFN Package
24-pin, 4mm x 4mm x 0.9mm

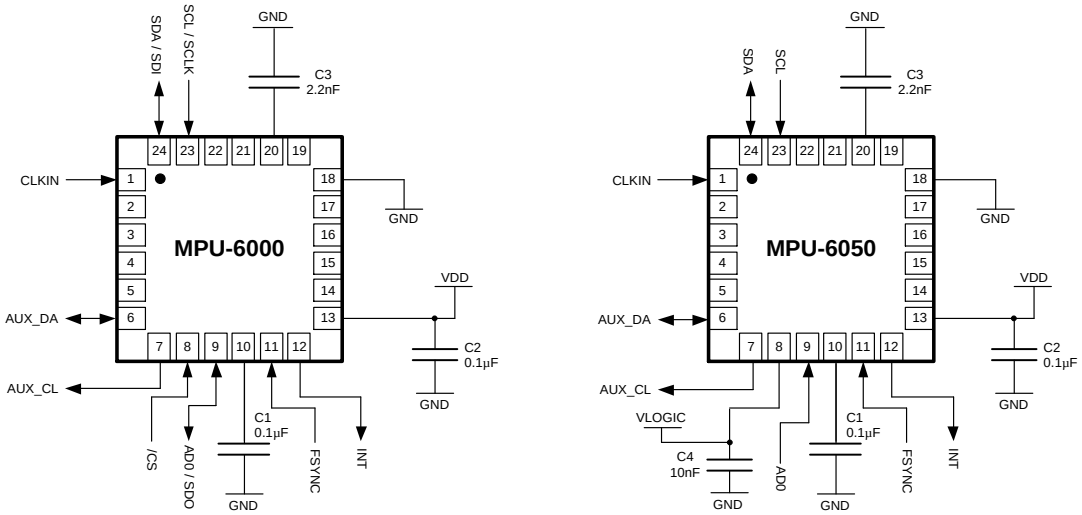


QFN Package
24-pin, 4mm x 4mm x 0.9mm



Orientation of Axes of Sensitivity and Polarity of Rotation

7.2 Typical Operating Circuit



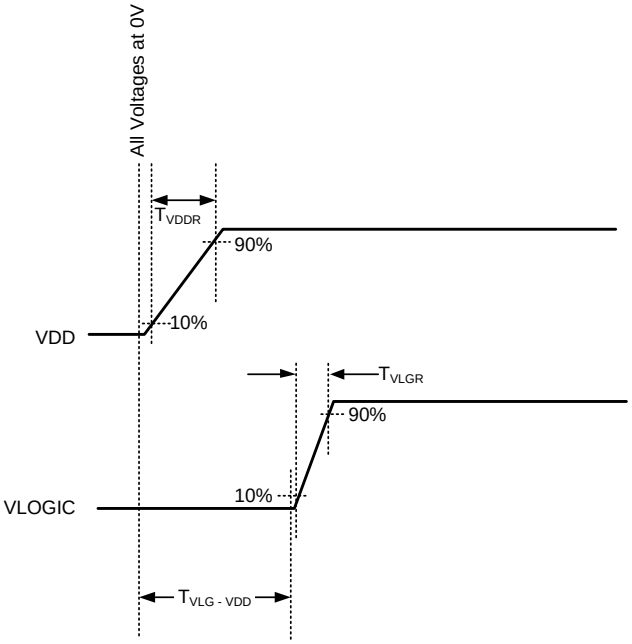
Typical Operating Circuits

7.3 Bill of Materials for External Components

Component	Label	Specification	Quantity
Regulator Filter Capacitor (Pin 10)	C1	Ceramic, X7R, 0.1µF ±10%, 2V	1
VDD Bypass Capacitor (Pin 13)	C2	Ceramic, X7R, 0.1µF ±10%, 4V	1
Charge Pump Capacitor (Pin 20)	C3	Ceramic, X7R, 2.2nF ±10%, 50V	1
VLOGIC Bypass Capacitor (Pin 8)	C4*	Ceramic, X7R, 10nF ±10%, 4V	1

* MPU-6050 Only.

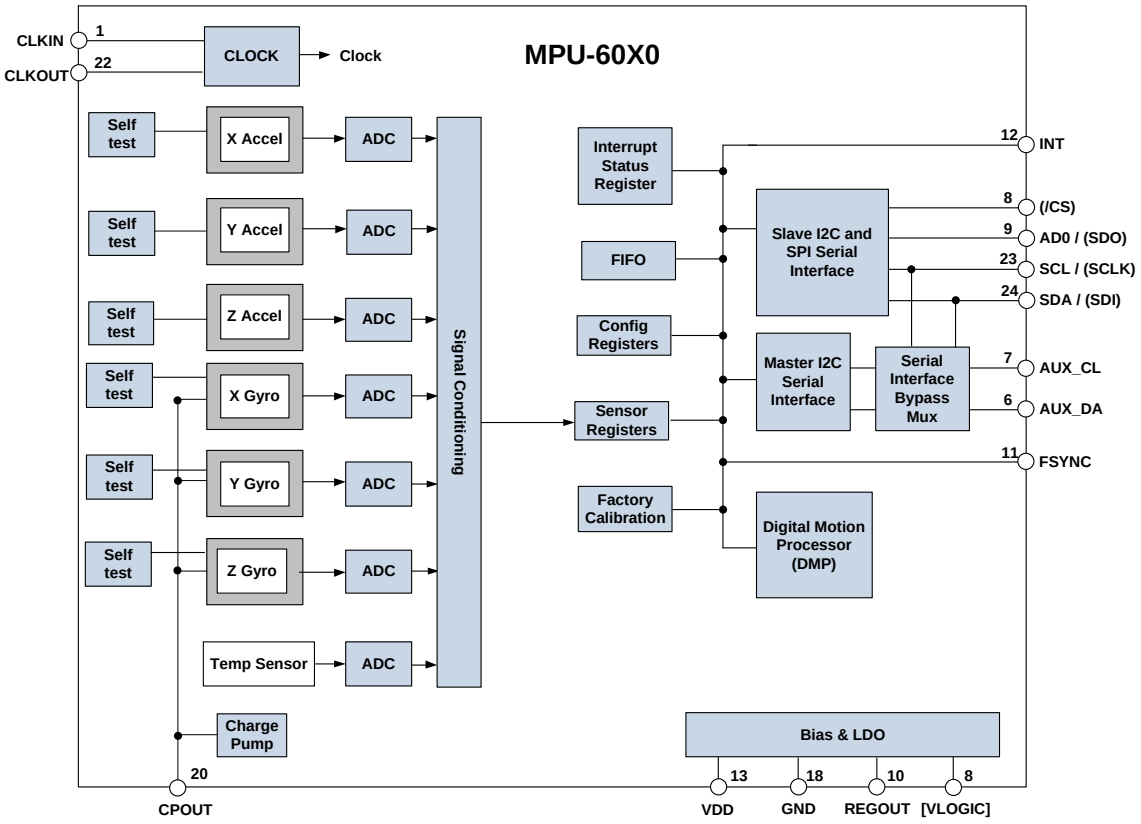
7.4 Recommended Power-on Procedure



Power-Up Sequencing

1. VLOGIC amplitude must always be ≤VDD amplitude
2. T_{VDDR} is VDD rise time: Time for VDD to rise from 10% to 90% of its final value
3. T_{VDDR} is ≤100ms
4. T_{VLGR} is VLOGIC rise time: Time for VLOGIC to rise from 10% to 90% of its final value
5. T_{VLGR} is ≤3ms
6. $T_{VLG-VDD}$ is the delay from the start of VDD ramp to the start of VLOGIC rise
7. $T_{VLG-VDD}$ is ≥0
8. VDD and VLOGIC must be monotonic ramps

7.5 Block Diagram



Note: Pin names in round brackets ( ) apply only to MPU-6000
Pin names in square brackets [ ] apply only to MPU-6050

7.6 Overview

The MPU-60X0 is comprised of the following key blocks and functions:

- Three-axis MEMS rate gyroscope sensor with 16-bit ADCs and signal conditioning
- Three-axis MEMS accelerometer sensor with 16-bit ADCs and signal conditioning
- Digital Motion Processor (DMP) engine
- Primary I^2C and SPI (MPU-6000 only) serial communications interfaces
- Auxiliary I^2C serial interface for 3rd party magnetometer & other sensors
- Clocking
- Sensor Data Registers
- FIFO
- Interrupts
- Digital-Output Temperature Sensor
- Gyroscope & Accelerometer Self-test
- Bias and LDO
- Charge Pump

7.7 Three-Axis MEMS Gyroscope with 16-bit ADCs and Signal Conditioning

The MPU-60X0 consists of three independent vibratory MEMS rate gyroscopes, which detect rotation about the X-, Y-, and Z- Axes. When the gyros are rotated about any of the sense axes, the Coriolis Effect causes a vibration that is detected by a capacitive pickoff. The resulting signal is amplified, demodulated, and filtered to produce a voltage that is proportional to the angular rate. This voltage is digitized using individual on-chip 16-bit Analog-to-Digital Converters (ADCs) to sample each axis. The full-scale range of the gyro sensors may be digitally programmed to ±250, ±500, ±1000, or ±2000 degrees per second (dps). The ADC sample rate is programmable from 8,000 samples per second, down to 3.9 samples per second, and user-selectable low-pass filters enable a wide range of cut-off frequencies.

7.8 Three-Axis MEMS Accelerometer with 16-bit ADCs and Signal Conditioning

The MPU-60X0's 3-Axis accelerometer uses separate proof masses for each axis. Acceleration along a particular axis induces displacement on the corresponding proof mass, and capacitive sensors detect the displacement differentially. The MPU-60X0's architecture reduces the accelerometers' susceptibility to fabrication variations as well as to thermal drift. When the device is placed on a flat surface, it will measure 0*g* on the X- and Y-axes and +1*g* on the Z-axis. The accelerometers' scale factor is calibrated at the factory and is nominally independent of supply voltage. Each sensor has a dedicated sigma-delta ADC for providing digital outputs. The full scale range of the digital output can be adjusted to ±2*g*, ±4*g*, ±8*g*, or ±16*g*.

7.9 Digital Motion Processor

The embedded Digital Motion Processor (DMP) is located within the MPU-60X0 and offloads computation of motion processing algorithms from the host processor. The DMP acquires data from accelerometers, gyroscopes, and additional 3rd party sensors such as magnetometers, and processes the data. The resulting data can be read from the DMP's registers, or can be buffered in a FIFO. The DMP has access to one of the MPU's external pins, which can be used for generating interrupts.

The purpose of the DMP is to offload both timing requirements and processing power from the host processor. Typically, motion processing algorithms should be run at a high rate, often around 200Hz, in order to provide accurate results with low latency. This is required even if the application updates at a much lower rate; for example, a low power user interface may update as slowly as 5Hz, but the motion processing should still run at 200Hz. The DMP can be used as a tool in order to minimize power, simplify timing, simplify the software architecture, and save valuable MIPS on the host processor for use in the application.

7.10 Primary I^2C and SPI Serial Communications Interfaces

The MPU-60X0 communicates to a system processor using either a SPI (MPU-6000 only) or an I^2C serial interface. The MPU-60X0 always acts as a slave when communicating to the system processor. The LSB of the of the I^2C slave address is set by pin 9 (AD0).

The logic levels for communications between the MPU-60X0 and its master are as follows:

- MPU-6000: The logic level for communications with the master is set by the voltage on VDD
- MPU-6050: The logic level for communications with the master is set by the voltage on VLOGIC

For further information regarding the logic levels of the MPU-6050, please refer to Section 10.

7.11 Auxiliary I²C Serial Interface

The MPU-60X0 has an auxiliary I²C bus for communicating to an off-chip 3-Axis digital output magnetometer or other sensors. This bus has two operating modes:

- I²C Master Mode: The MPU-60X0 acts as a master to any external sensors connected to the auxiliary I²C bus
- Pass-Through Mode: The MPU-60X0 directly connects the primary and auxiliary I²C buses together, allowing the system processor to directly communicate with any external sensors.

Auxiliary I²C Bus Modes of Operation:

- I²C Master Mode: Allows the MPU-60X0 to directly access the data registers of external digital sensors, such as a magnetometer. In this mode, the MPU-60X0 directly obtains data from auxiliary sensors, allowing the on-chip DMP to generate sensor fusion data without intervention from the system applications processor.

  For example, In I²C Master mode, the MPU-60X0 can be configured to perform burst reads, returning the following data from a magnetometer:

  - X magnetometer data (2 bytes)
  - Y magnetometer data (2 bytes)
  - Z magnetometer data (2 bytes)

  The I²C Master can be configured to read up to 24 bytes from up to 4 auxiliary sensors. A fifth sensor can be configured to work single byte read/write mode.

- Pass-Through Mode: Allows an external system processor to act as master and directly communicate to the external sensors connected to the auxiliary I²C bus pins (AUX_DA and AUX_CL). In this mode, the auxiliary I²C bus control logic (3rd party sensor interface block) of the MPU-60X0 is disabled, and the auxiliary I²C pins AUX_DA and AUX_CL (Pins 6 and 7) are connected to the main I²C bus (Pins 23 and 24) through analog switches.

  Pass-Through Mode is useful for configuring the external sensors, or for keeping the MPU-60X0 in a low-power mode when only the external sensors are used.

  In Pass-Through Mode the system processor can still access MPU-60X0 data through the I²C interface.

Auxiliary I²C Bus IO Logic Levels

- MPU-6000: The logic level of the auxiliary I²C bus is VDD
- MPU-6050: The logic level of the auxiliary I²C bus can be programmed to be either VDD or VLOGIC

For further information regarding the MPU-6050's logic levels, please refer to Section 10.2.

7.12 Self-Test

Please refer to the MPU-6000/MPU-6050 Register Map and Register Descriptions document for more details on self test.

Self-test allows for the testing of the mechanical and electrical portions of the sensors. The self-test for each measurement axis can be activated by means of the gyroscope and accelerometer self-test registers (registers 13 to 16).

When self-test is activated, the electronics cause the sensors to be actuated and produce an output signal. The output signal is used to observe the self-test response.

The self-test response is defined as follows:

Self-test response = Sensor output with self-test enabled – Sensor output without self-test enabled

The self-test response for each accelerometer axis is defined in the accelerometer specification table (Section 6.2), while that for each gyroscope axis is defined in the gyroscope specification table (Section 6.1).

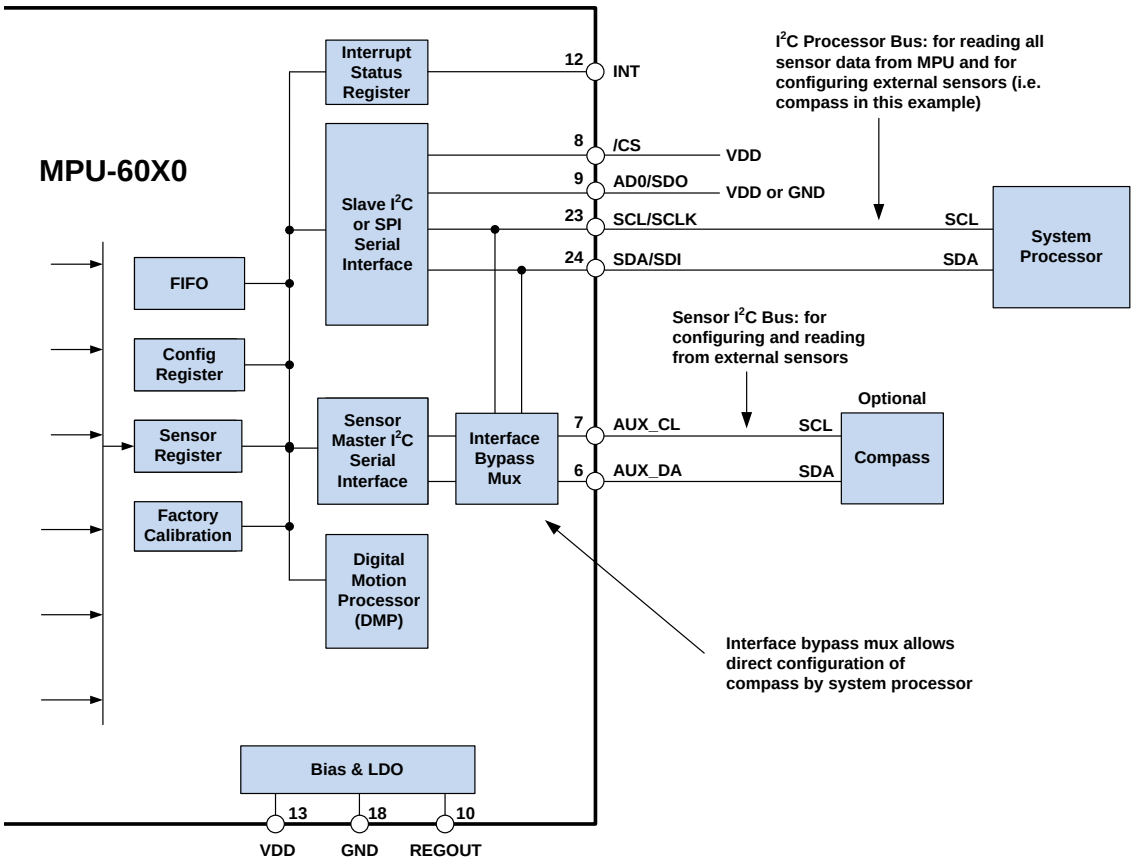
When the value of the self-test response is within the min/max limits of the product specification, the part has passed self test. When the self-test response exceeds the min/max values, the part is deemed to have failed self-test. Code for operating self test code is included within the MotionApps software provided by InvenSense.

7.13 MPU-60X0 Solution for 9-axis Sensor Fusion Using I²C Interface

In the figure below, the system processor is an I²C master to the MPU-60X0. In addition, the MPU-60X0 is an I²C master to the optional external compass sensor. The MPU-60X0 has limited capabilities as an I²C Master, and depends on the system processor to manage the initial configuration of any auxiliary sensors. The MPU-60X0 has an interface bypass multiplexer, which connects the system processor I²C bus pins 23 and 24 (SDA and SCL) directly to the auxiliary sensor I²C bus pins 6 and 7 (AUX_DA and AUX_CL).

Once the auxiliary sensors have been configured by the system processor, the interface bypass multiplexer should be disabled so that the MPU-60X0 auxiliary I²C master can take control of the sensor I²C bus and gather data from the auxiliary sensors.

For further information regarding I²C master control, please refer to Section 10.



7.14 MPU-6000 Using SPI Interface

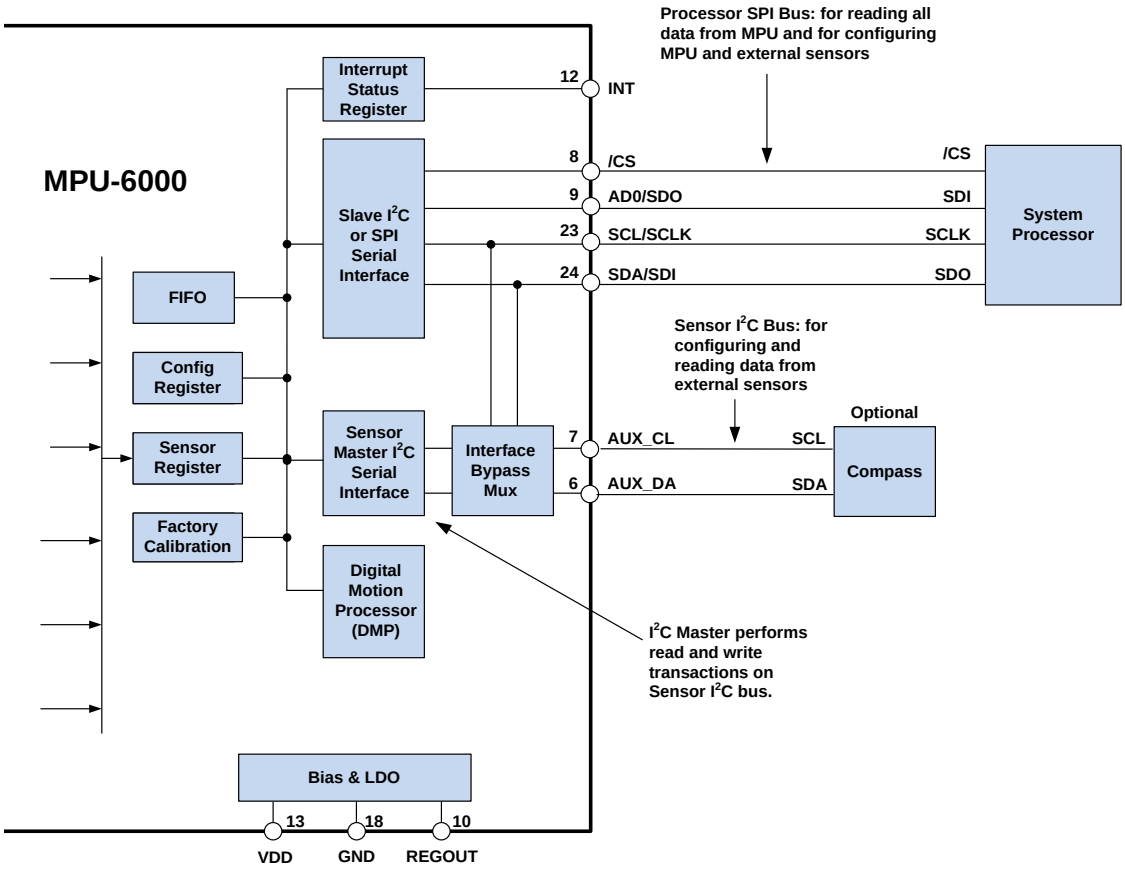
In the figure below, the system processor is an SPI master to the MPU-6000. Pins 8, 9, 23, and 24 are used to support the /CS, SDO, SCLK, and SDI signals for SPI communications. Because these SPI pins are shared with the I^2C slave pins (9, 23 and 24), the system processor cannot access the auxiliary I^2C bus through the interface bypass multiplexer, which connects the processor I^2C interface pins to the sensor I^2C interface pins.

Since the MPU-6000 has limited capabilities as an I^2C Master, and depends on the system processor to manage the initial configuration of any auxiliary sensors, another method must be used for programming the sensors on the auxiliary sensor I^2C bus pins 6 and 7 (AUX_DA and AUX_CL).

When using SPI communications between the MPU-6000 and the system processor, configuration of devices on the auxiliary I^2C sensor bus can be achieved by using I^2C Slaves 0-4 to perform read and write transactions on any device and register on the auxiliary I^2C bus. The I^2C Slave 4 interface can be used to perform only single byte read and write transactions.

Once the external sensors have been configured, the MPU-6000 can perform single or multi-byte reads using the sensor I^2C bus. The read results from the Slave 0-3 controllers can be written to the FIFO buffer as well as to the external sensor registers.

For further information regarding the control of the MPU-60X0's auxiliary I^2C interface, please refer to the MPU-6000/MPU-6050 Register Map and Register Descriptions document.



7.15 Internal Clock Generation

The MPU-60X0 has a flexible clocking scheme, allowing a variety of internal or external clock sources to be used for the internal synchronous circuitry. This synchronous circuitry includes the signal conditioning and ADCs, the DMP, and various control circuits and registers. An on-chip PLL provides flexibility in the allowable inputs for generating this clock.

Allowable internal sources for generating the internal clock are:

- An internal relaxation oscillator
- Any of the X, Y, or Z gyros (MEMS oscillators with a variation of ±1% over temperature)

Allowable external clocking sources are:

- 32.768kHz square wave
- 19.2MHz square wave

Selection of the source for generating the internal synchronous clock depends on the availability of external sources and the requirements for power consumption and clock accuracy. These requirements will most likely vary by mode of operation. For example, in one mode, where the biggest concern is power consumption, the user may wish to operate the Digital Motion Processor of the MPU-60X0 to process accelerometer data, while keeping the gyros off. In this case, the internal relaxation oscillator is a good clock choice. However, in another mode, where the gyros are active, selecting the gyros as the clock source provides for a more accurate clock source.

Clock accuracy is important, since timing errors directly affect the distance and angle calculations performed by the Digital Motion Processor (and by extension, by any processor).

There are also start-up conditions to consider. When the MPU-60X0 first starts up, the device uses its internal clock until programmed to operate from another source. This allows the user, for example, to wait for the MEMS oscillators to stabilize before they are selected as the clock source.

7.16 Sensor Data Registers

The sensor data registers contain the latest gyro, accelerometer, auxiliary sensor, and temperature measurement data. They are read-only registers, and are accessed via the serial interface. Data from these registers may be read anytime. However, the interrupt function may be used to determine when new data is available.

For a table of interrupt sources please refer to Section 8.

7.17 FIFO

The MPU-60X0 contains a 1024-byte FIFO register that is accessible via the Serial Interface. The FIFO configuration register determines which data is written into the FIFO. Possible choices include gyro data, accelerometer data, temperature readings, auxiliary sensor readings, and FSYNC input. A FIFO counter keeps track of how many bytes of valid data are contained in the FIFO. The FIFO register supports burst reads. The interrupt function may be used to determine when new data is available.

For further information regarding the FIFO, please refer to the MPU-6000/MPU-6050 Register Map and Register Descriptions document.

7.18 Interrupts

Interrupt functionality is configured via the Interrupt Configuration register. Items that are configurable include the INT pin configuration, the interrupt latching and clearing method, and triggers for the interrupt. Items that can trigger an interrupt are (1) Clock generator locked to new reference oscillator (used when switching clock

sources); (2) new data is available to be read (from the FIFO and Data registers); (3) accelerometer event interrupts; and (4) the MPU-60X0 did not receive an acknowledge from an auxiliary sensor on the secondary I^2C bus. The interrupt status can be read from the Interrupt Status register.

For further information regarding interrupts, please refer to the MPU-60X0 Register Map and Register Descriptions document.

For information regarding the MPU-60X0's accelerometer event interrupts, please refer to Section 8.

7.19 Digital-Output Temperature Sensor

An on-chip temperature sensor and ADC are used to measure the MPU-60X0 die temperature. The readings from the ADC can be read from the FIFO or the Sensor Data registers.

7.20 Bias and LDO

The bias and LDO section generates the internal supply and the reference voltages and currents required by the MPU-60X0. Its two inputs are an unregulated VDD of 2.375 to 3.46V and a VLOGIC logic reference supply voltage of 1.71V to VDD (MPU-6050 only). The LDO output is bypassed by a capacitor at REGOUT. For further details on the capacitor, please refer to the Bill of Materials for External Components (Section 7.3).

7.21 Charge Pump

An on-board charge pump generates the high voltage required for the MEMS oscillators. Its output is bypassed by a capacitor at CPOUT. For further details on the capacitor, please refer to the Bill of Materials for External Components (Section 7.3).

8 Programmable Interrupts

The MPU-60X0 has a programmable interrupt system which can generate an interrupt signal on the INT pin. Status flags indicate the source of an interrupt. Interrupt sources may be enabled and disabled individually.

Table of Interrupt Sources

Interrupt Name	Module
FIFO Overflow	FIFO
Data Ready	Sensor Registers
I^2C Master errors: Lost Arbitration, NACKs	I^2C Master
I^2C Slave 4	I^2C Master

For information regarding the interrupt enable/disable registers and flag registers, please refer to the MPU-6000/MPU-6050 Register Map and Register Descriptions document. Some interrupt sources are explained below.

9 Digital Interface

9.1 I²C and SPI (MPU-6000 only) Serial Interfaces

The internal registers and memory of the MPU-6000/MPU-6050 can be accessed using either I²C at 400 kHz or SPI at 1MHz (MPU-6000 only). SPI operates in four-wire mode.

Serial Interface

Pin Number	MPU-6000	MPU-6050	Pin Name	Pin Description
8	Y		/CS	SPI chip select (0=SPI enable)
8		Y	VLOGIC	Digital I/O supply voltage. VLOGIC must be ≤ VDD at all times.
9	Y		AD0 / SDO	I²C Slave Address LSB (AD0); SPI serial data output (SDO)
9		Y	AD0	I²C Slave Address LSB
23	Y		SCL / SCLK	I²C serial clock (SCL); SPI serial clock (SCLK)
23		Y	SCL	I²C serial clock
24	Y		SDA / SDI	I²C serial data (SDA); SPI serial data input (SDI)
24		Y	SDA	I²C serial data

Note:

To prevent switching into I²C mode when using SPI (MPU-6000), the I²C interface should be disabled by setting the *I2C_IF_DIS* configuration bit. Setting this bit should be performed immediately after waiting for the time specified by the "Start-Up Time for Register Read/Write" in Section 6.3.

For further information regarding the *I2C_IF_DIS* bit, please refer to the MPU-6000/MPU-6050 Register Map and Register Descriptions document.

9.2 I²C Interface

I²C is a two-wire interface comprised of the signals serial data (SDA) and serial clock (SCL). In general, the lines are open-drain and bi-directional. In a generalized I²C interface implementation, attached devices can be a master or a slave. The master device puts the slave address on the bus, and the slave device with the matching address acknowledges the master.

The MPU-60X0 always operates as a slave device when communicating to the system processor, which thus acts as the master. SDA and SCL lines typically need pull-up resistors to VDD. The maximum bus speed is 400 kHz.

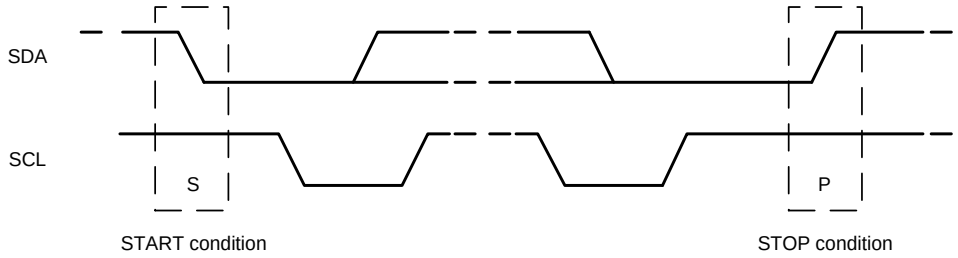
The slave address of the MPU-60X0 is b110100X which is 7 bits long. The LSB bit of the 7 bit address is determined by the logic level on pin AD0. This allows two MPU-60X0s to be connected to the same I²C bus. When used in this configuration, the address of the one of the devices should be b1101000 (pin AD0 is logic low) and the address of the other should be b1101001 (pin AD0 is logic high).

9.3 I²C Communications Protocol

START (S) and STOP (P) Conditions

Communication on the I²C bus starts when the master puts the START condition (S) on the bus, which is defined as a HIGH-to-LOW transition of the SDA line while SCL line is HIGH (see figure below). The bus is considered to be busy until the master puts a STOP condition (P) on the bus, which is defined as a LOW to HIGH transition on the SDA line while SCL is HIGH (see figure below).

Additionally, the bus remains busy if a repeated START (Sr) is generated instead of a STOP condition.

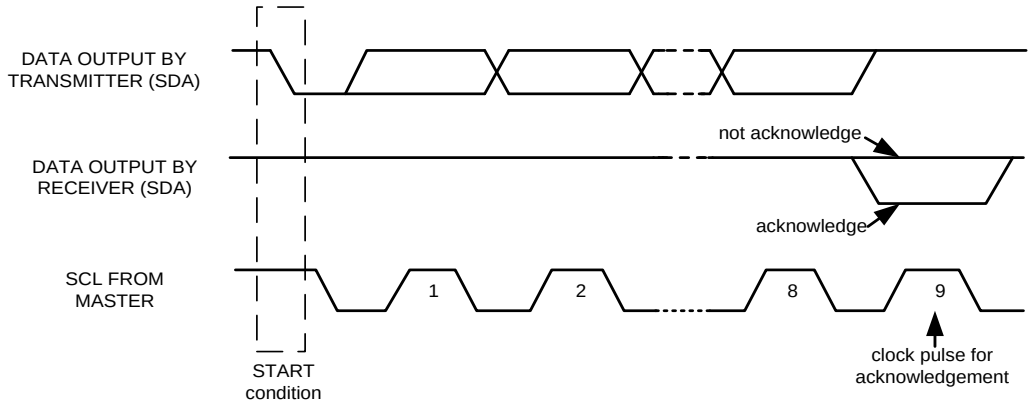


START and STOP Conditions

Data Format / Acknowledge

I^2C data bytes are defined to be 8-bits long. There is no restriction to the number of bytes transmitted per data transfer. Each byte transferred must be followed by an acknowledge (ACK) signal. The clock for the acknowledge signal is generated by the master, while the receiver generates the actual acknowledge signal by pulling down SDA and holding it low during the HIGH portion of the acknowledge clock pulse.

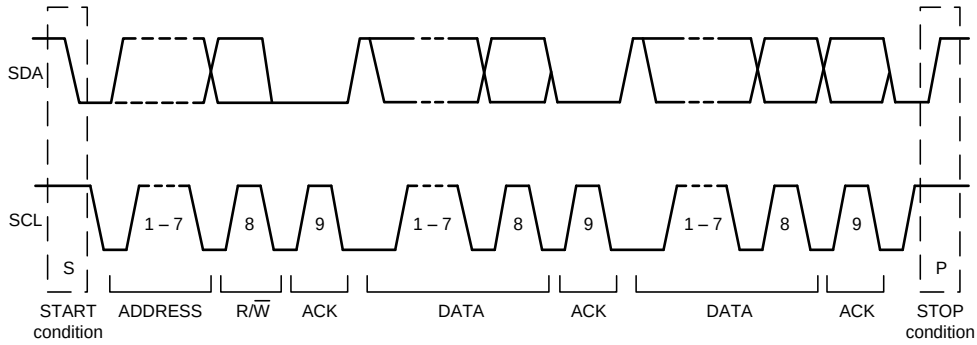
If a slave is busy and cannot transmit or receive another byte of data until some other task has been performed, it can hold SCL LOW, thus forcing the master into a wait state. Normal data transfer resumes when the slave is ready, and releases the clock line (refer to the following figure).



Acknowledge on the I^2C Bus

Communications

After beginning communications with the START condition (S), the master sends a 7-bit slave address followed by an 8th bit, the read/write bit. The read/write bit indicates whether the master is receiving data from or is writing to the slave device. Then, the master releases the SDA line and waits for the acknowledge signal (ACK) from the slave device. Each byte transferred must be followed by an acknowledge bit. To acknowledge, the slave device pulls the SDA line LOW and keeps it LOW for the high period of the SCL line. Data transmission is always terminated by the master with a STOP condition (P), thus freeing the communications line. However, the master can generate a repeated START condition (Sr), and address another slave without first generating a STOP condition (P). A LOW to HIGH transition on the SDA line while SCL is HIGH defines the stop condition. All SDA changes should take place when SCL is low, with the exception of start and stop conditions.



Complete I²C Data Transfer

To write the internal MPU-60X0 registers, the master transmits the start condition (S), followed by the I²C address and the write bit (0). At the 9th clock cycle (when the clock is high), the MPU-60X0 acknowledges the transfer. Then the master puts the register address (RA) on the bus. After the MPU-60X0 acknowledges the reception of the register address, the master puts the register data onto the bus. This is followed by the ACK signal, and data transfer may be concluded by the stop condition (P). To write multiple bytes after the last ACK signal, the master can continue outputting data rather than transmitting a stop signal. In this case, the MPU-60X0 automatically increments the register address and loads the data to the appropriate register. The following figures show single and two-byte write sequences.

Single-Byte Write Sequence

Master	S	AD+W		RA		DATA		P
Slave			ACK		ACK		ACK	

Burst Write Sequence

Master	S	AD+W		RA		DATA		DATA		P
Slave			ACK		ACK		ACK		ACK	

To read the internal MPU-60X0 registers, the master sends a start condition, followed by the I^2C address and a write bit, and then the register address that is going to be read. Upon receiving the ACK signal from the MPU-60X0, the master transmits a start signal followed by the slave address and read bit. As a result, the MPU-60X0 sends an ACK signal and the data. The communication ends with a not acknowledge (NACK) signal and a stop bit from master. The NACK condition is defined such that the SDA line remains high at the 9^{th} clock cycle. The following figures show single and two-byte read sequences.

Single-Byte Read Sequence

Master	S	AD+W		RA		S	AD+R			NACK	P
Slave			ACK		ACK			ACK	DATA		

Burst Read Sequence

Master	S	AD+W		RA		S	AD+R			ACK		NACK	P
Slave			ACK		ACK			ACK	DATA		DATA		

9.4 I^2C Terms

Signal	Description
S	Start Condition: SDA goes from high to low while SCL is high
AD	Slave I^2C address
W	Write bit (0)
R	Read bit (1)
ACK	Acknowledge: SDA line is low while the SCL line is high at the 9^{th} clock cycle
NACK	Not-Acknowledge: SDA line stays high at the 9^{th} clock cycle
RA	MPU-60X0 internal register address
DATA	Transmit or received data
P	Stop condition: SDA going from low to high while SCL is high

9.5 SPI Interface (MPU-6000 only)

SPI is a 4-wire synchronous serial interface that uses two control lines and two data lines. The MPU-6000 always operates as a Slave device during standard Master-Slave SPI operation.

With respect to the Master, the Serial Clock output (SCLK), the Serial Data Output (SDO) and the Serial Data Input (SDI) are shared among the Slave devices. Each SPI slave device requires its own Chip Select (/CS) line from the master.

/CS goes low (active) at the start of transmission and goes back high (inactive) at the end. Only one /CS line is active at a time, ensuring that only one slave is selected at any given time. The /CS lines of the non-selected slave devices are held high, causing their SDO lines to remain in a high-impedance (high-z) state so that they do not interfere with any active devices.

SPI Operational Features

1. Data is delivered MSB first and LSB last
2. Data is latched on the rising edge of SCLK
3. Data should be transitioned on the falling edge of SCLK
4. The maximum frequency of SCLK is 1MHz
5. SPI read and write operations are completed in 16 or more clock cycles (two or more bytes). The first byte contains the SPI Address, and the following byte(s) contain(s) the SPI data. The first bit of the first byte contains the Read/Write bit and indicates the Read (1) or Write (0) operation. The following 7 bits contain the Register Address. In cases of multiple-byte Read/Writes, data is two or more bytes:

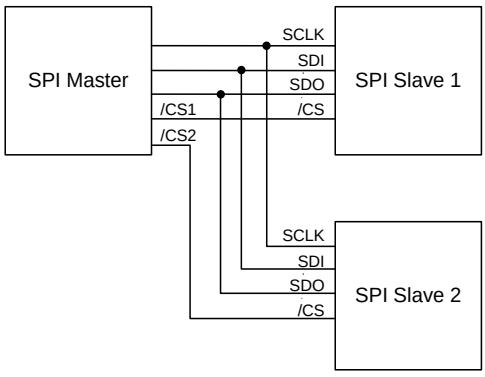
SPI Address format

MSB							**LSB**
R/W	A6	A5	A4	A3	A2	A1	A0

SPI Data format

MSB							**LSB**
D7	D6	D5	D4	D3	D2	D1	D0

6. Supports Single or Burst Read/Writes.



Typical SPI Master / Slave Configuration

10 Serial Interface Considerations (MPU-6050)

10.1 MPU-6050 Supported Interfaces

The MPU-6050 supports I^2C communications on both its primary (microprocessor) serial interface and its auxiliary interface.

10.2 Logic Levels

The MPU-6050's I/O logic levels are set to be VLOGIC, as shown in the table below. AUX_VDDIO must be set to 0.

I/O Logic Levels vs. *AUX_VDDIO*

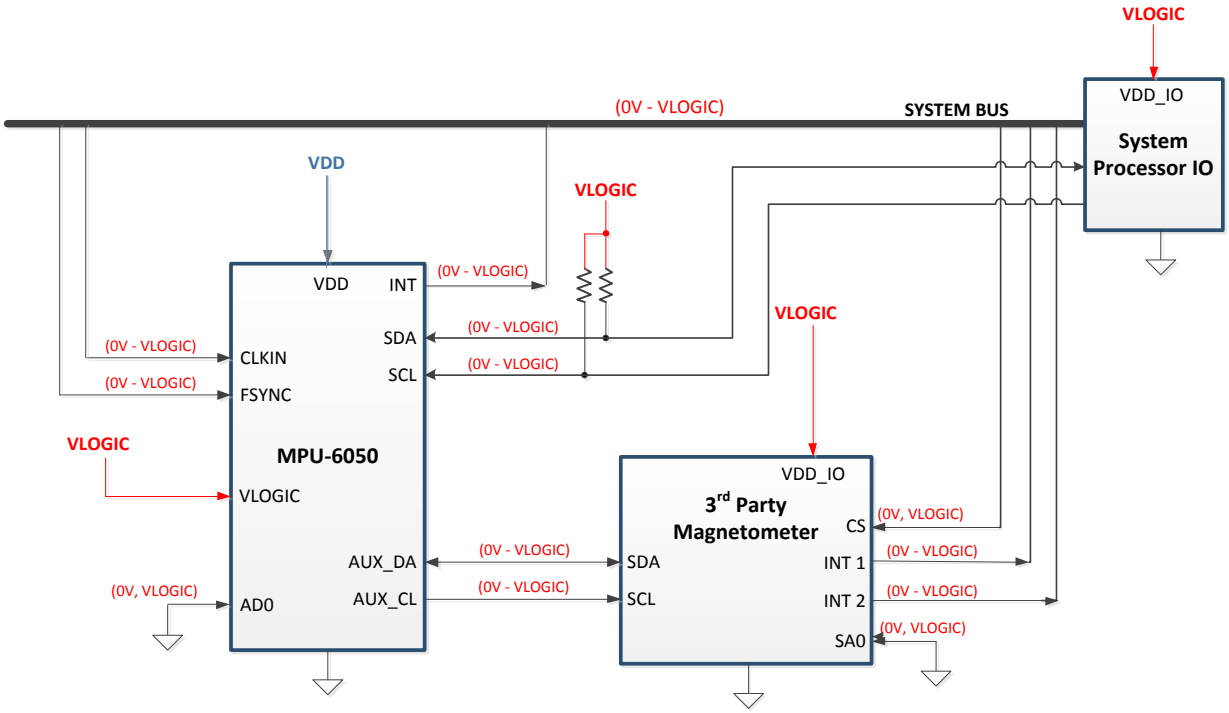
AUX_VDDIO	**MICROPROCESSOR LOGIC LEVELS** (Pins: SDA, SCL, AD0, CLKIN, INT)	**AUXILLARY LOGIC LEVELS** (Pins: AUX_DA, AUX_CL)
0	VLOGIC	VLOGIC

Note: The power-on-reset value for *AUX_VDDIO* is 0.

When *AUX_VDDIO* is set to 0 (its power-on-reset value), VLOGIC is the power supply voltage for both the microprocessor system bus and the auxiliary I^2C bus, as shown in the figure of Section 10.3.

10.3 Logic Levels Diagram for AUX_*VDDIO* = 0

The figure below depicts a sample circuit with a third party magnetometer attached to the auxiliary I^2C bus. It shows logic levels and voltage connections for *AUX_VDDIO* = 0. Note: Actual configuration will depend on the auxiliary sensors used.



I/O Levels and Connections for *AUX_VDDIO* = 0

Notes:

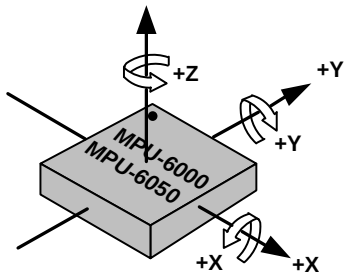
1. AUX_VDDIO determines the IO voltage levels of AUX_DA and AUX_CL (0 = set output levels relative to VLOGIC)
2. All other MPU-6050 logic IOs are referenced to VLOGIC.

11 Assembly

This section provides general guidelines for assembling InvenSense Micro Electro-Mechanical Systems (MEMS) gyros packaged in Quad Flat No leads package (QFN) surface mount integrated circuits.

11.1 Orientation of Axes

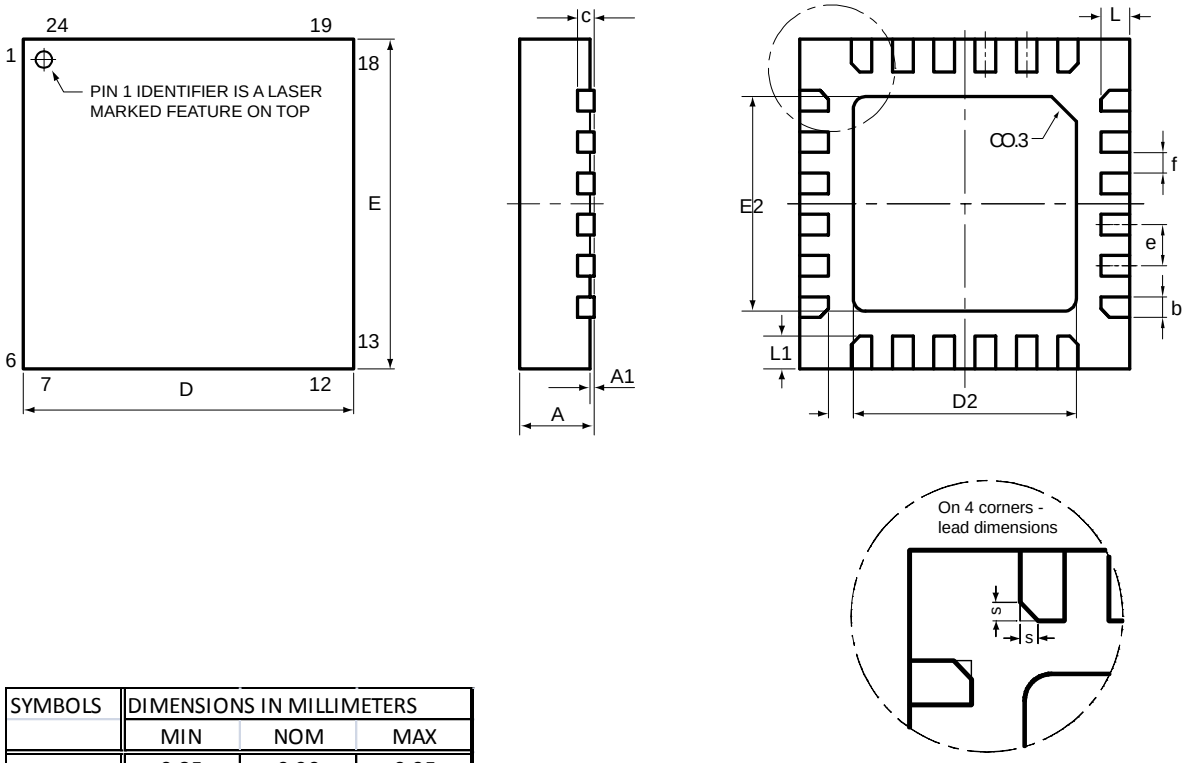
The diagram below shows the orientation of the axes of sensitivity and the polarity of rotation. Note the pin 1 identifier (•) in the figure.



Orientation of Axes of Sensitivity and Polarity of Rotation

11.2 Package Dimensions

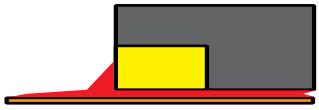
24 Lead QFN (4x4x0.9) mm NiPdAu Lead-frame finish



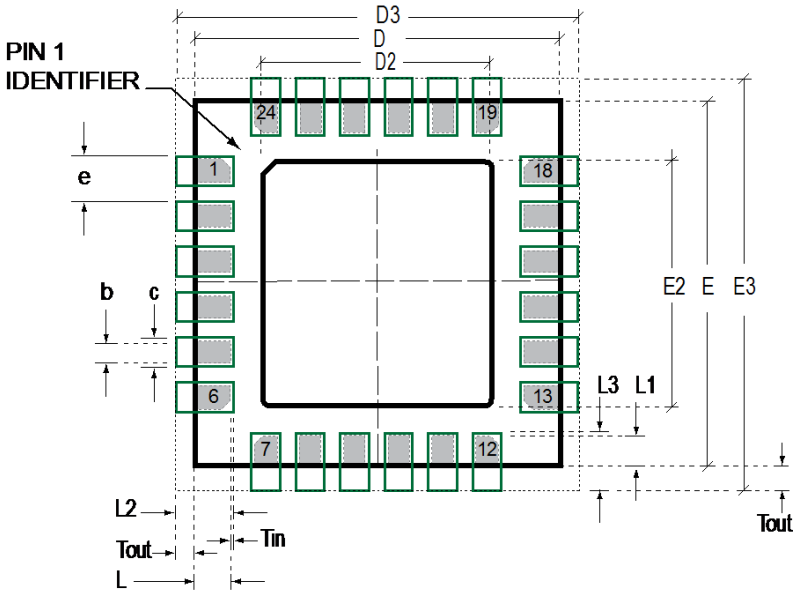
SYMBOLS	DIMENSIONS IN MILLIMETERS		
	MIN	NOM	MAX
A	0.85	0.90	0.95
A1	0.00	0.02	0.05
b	0.18	0.25	0.30
c	---	0.20 REF	---
D	3.90	4.00	4.10
D2	2.65	2.70	2.75
E	3.90	4.00	4.10
E2	2.55	2.60	2.65
e	---	0.50	---
f (e-b)	---	0.25	---
K	0.25	0.30	0.35
L	0.30	0.35	0.40
L1	0.35	0.40	0.45
s	0.05	---	0.15

11.3 PCB Design Guidelines

The Pad Diagram using a JEDEC type extension with solder rising on the outer edge is shown below. The Pad Dimensions Table shows pad sizing (mean dimensions) recommended for the MPU-60X0 product.



JEDEC type extension with solder rising on outer edge



PCB Layout Diagram

SYMBOLS	DIMENSIONS IN MILLIMETERS	NOM
Nominal Package I/O Pad Dimensions		
e	Pad Pitch	0.50
b	Pad Width	0.25
L	Pad Length	0.35
L1	Pad Length	0.40
D	Package Width	4.00
E	Package Length	4.00
D2	Exposed Pad Width	2.70
E2	Exposed Pad Length	2.60
I/O Land Design Dimensions (Guidelines)		
D3	I/O Pad Extent Width	4.80
E3	I/O Pad Extent Length	4.80
c	Land Width	0.35
Tout	Outward Extension	0.40
Tin	Inward Extension	0.05
L2	Land Length	0.80
L3	Land Length	0.85

PCB Dimensions Table (for PCB Lay-out Diagram)

11.4 Assembly Precautions

11.4.1 Gyroscope Surface Mount Guidelines

InvenSense MEMS Gyros sense rate of rotation. In addition, gyroscopes sense mechanical stress coming from the printed circuit board (PCB). This PCB stress can be minimized by adhering to certain design rules:

When using MEMS gyroscope components in plastic packages, PCB mounting and assembly can cause package stress. This package stress in turn can affect the output offset and its value over a wide range of temperatures. This stress is caused by the mismatch between the Coefficient of Linear Thermal Expansion (CTE) of the package material and the PCB. Care must be taken to avoid package stress due to mounting.

Traces connected to pads should be as symmetric as possible. Maximizing symmetry and balance for pad connection will help component self alignment and will lead to better control of solder paste reduction after reflow.

Any material used in the surface mount assembly process of the MEMS gyroscope should be free of restricted RoHS elements or compounds. Pb-free solders should be used for assembly.

11.4.2 Exposed Die Pad Precautions

The MPU-60X0 has very low active and standby current consumption. The exposed die pad is not required for heat sinking, and should not be soldered to the PCB. Failure to adhere to this rule can induce performance changes due to package thermo-mechanical stress. There is no electrical connection between the pad and the CMOS.

11.4.3 Trace Routing

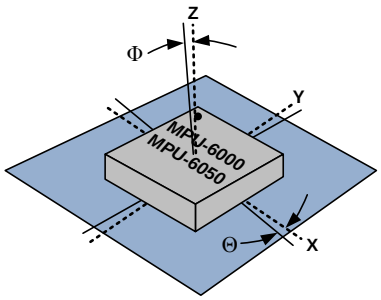
Routing traces or vias under the gyro package such that they run under the exposed die pad is prohibited. Routed active signals may harmonically couple with the gyro MEMS devices, compromising gyro response. These devices are designed with the drive frequencies as follows: X = 33±3Khz, Y = 30±3Khz, and Z=27±3Khz. To avoid harmonic coupling don't route active signals in non-shielded signal planes directly below, or above the gyro package. Note: For best performance, design a ground plane under the e-pad to reduce PCB signal noise from the board on which the gyro device is mounted. If the gyro device is stacked under an adjacent PCB board, design a ground plane directly above the gyro device to shield active signals from the adjacent PCB board.

11.4.4 Component Placement

Do not place large insertion components such as keyboard or similar buttons, connectors, or shielding boxes at a distance of less than 6 mm from the MEMS gyro. Maintain generally accepted industry design practices for component placement near the MPU-60X0 to prevent noise coupling and thermo-mechanical stress.

11.4.5 PCB Mounting and Cross-Axis Sensitivity

Orientation errors of the gyroscope and accelerometer mounted to the printed circuit board can cause cross-axis sensitivity in which one gyro or accel responds to rotation or acceleration about another axis, respectively. For example, the X-axis gyroscope may respond to rotation about the Y or Z axes. The orientation mounting errors are illustrated in the figure below.



Package Gyro & Accel Axes (– – –) Relative to PCB Axes (——) with Orientation Errors (Θ and Φ)

The table below shows the cross-axis sensitivity as a percentage of the gyroscope or accelerometer's sensitivity for a given orientation error, respectively.

Cross-Axis Sensitivity vs. Orientation Error

Orientation Error (θ or Φ)	Cross-Axis Sensitivity (sinθ or sinΦ)
0º	0%
0.5º	0.87%
1º	1.75%

The specifications for cross-axis sensitivity in Section 6.1 and Section 6.2 include the effect of the die orientation error with respect to the package.

11.4.6 MEMS Handling Instructions

MEMS (Micro Electro-Mechanical Systems) are a time-proven, robust technology used in hundreds of millions of consumer, automotive and industrial products. MEMS devices consist of microscopic moving mechanical structures. They differ from conventional IC products, even though they can be found in similar packages. Therefore, MEMS devices require different handling precautions than conventional ICs prior to mounting onto printed circuit boards (PCBs).

The MPU-60X0 has been qualified to a shock tolerance of 10,000*g*. InvenSense packages its gyroscopes as it deems proper for protection against normal handling and shipping. It recommends the following handling precautions to prevent potential damage.

- Do not drop individually packaged gyroscopes, or trays of gyroscopes onto hard surfaces. Components placed in trays could be subject to *g*-forces in excess of 10,000*g* if dropped.
- Printed circuit boards that incorporate mounted gyroscopes should not be separated by manually snapping apart. This could also create *g*-forces in excess of 10,000*g.*
- Do not clean MEMS gyroscopes in ultrasonic baths. Ultrasonic baths can induce MEMS damage if the bath energy causes excessive drive motion through resonant frequency coupling.

11.4.7 ESD Considerations

Establish and use ESD-safe handling precautions when unpacking and handling ESD-sensitive devices.

- Store ESD sensitive devices in ESD safe containers until ready for use. The Tape-and-Reel moisture-sealed bag is an ESD approved barrier. The best practice is to keep the units in the original moisture sealed bags until ready for assembly.

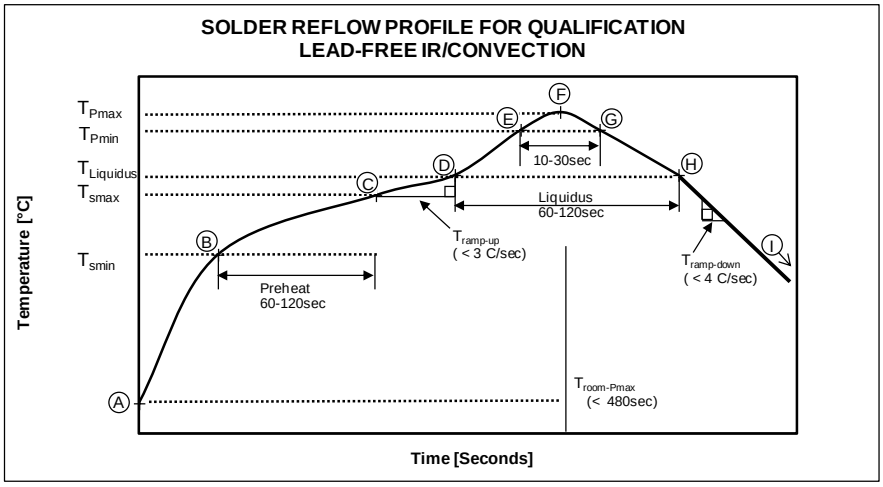
Restrict all device handling to ESD protected work areas that measure less than 200V static charge. Ensure that all workstations and personnel are properly grounded to prevent ESD.

11.4.8 Reflow Specification

Qualification Reflow: The MPU-60X0 was qualified in accordance with IPC/JEDEC J-STD-020D.1. This standard classifies proper packaging, storage and handling in order to avoid subsequent thermal and mechanical damage during the solder reflow attachment phase of PCB assembly.

The qualification preconditioning process specifies a sequence consisting of a bake cycle, a moisture soak cycle (in a temperature humidity oven), and three consecutive solder reflow cycles, followed by functional device testing.

The peak solder reflow classification temperature requirement for package qualification is (260 +5/-0°C) for lead-free soldering of components measuring less than 1.6 mm in thickness. The qualification profile and a table explaining the set-points are shown below:



Temperature Set Points Corresponding to Reflow Profile Above

Step	Setting	CONSTRAINTS		
		Temp (°C)	Time (sec)	Max. Rate (°C/sec)
A	T_{room}	25		
B	T_{Smin}	150		
C	T_{Smax}	200	$60 < t_{BC} < 120$	
D	$T_{Liquidus}$	217		$r_{(TLiquidus-TPmax)} < 3$
E	T_{Pmin} [255°C, 260°C]	255		$r_{(TLiquidus-TPmax)} < 3$
F	T_{Pmax} [260°C, 265°C]	260	$t_{AF} < 480$	$r_{(TLiquidus-TPmax)} < 3$
G	T_{Pmin} [255°C, 260°C]	255	$10< t_{EG} < 30$	$r_{(TPmax-TLiquidus)} < 4$
H	$T_{Liquidus}$	217	$60 < t_{DH} < 120$	
I	T_{room}	25		

Notes: Customers must never exceed the Classification temperature (T_{Pmax} = 260°C).
All temperatures refer to the topside of the QFN package, as measured on the package body surface.

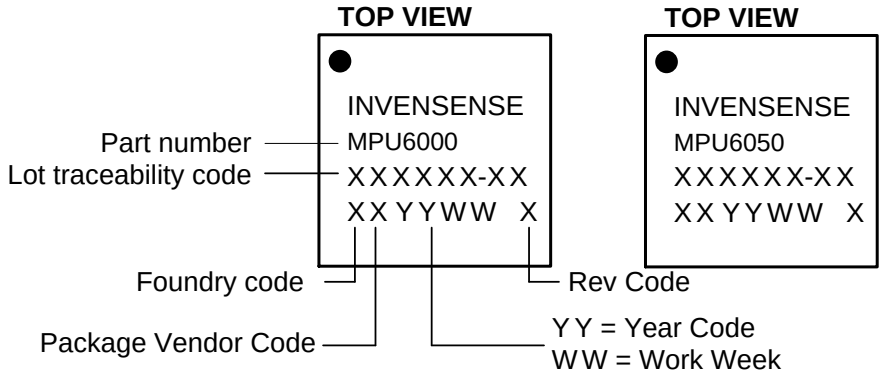
Production Reflow: Check the recommendations of your solder manufacturer. For optimum results, use lead-free solders that have lower specified temperature profiles (Tp_{max} ~ 235°C). Also use lower ramp-up and ramp-down rates than those used in the qualification profile. Never exceed the maximum conditions that we used for qualification, as these represent the maximum tolerable ratings for the device.

11.5 Storage Specifications

The storage specification of the MPU-60X0 conforms to IPC/JEDEC J-STD-020D.1 Moisture Sensitivity Level (MSL) 3.

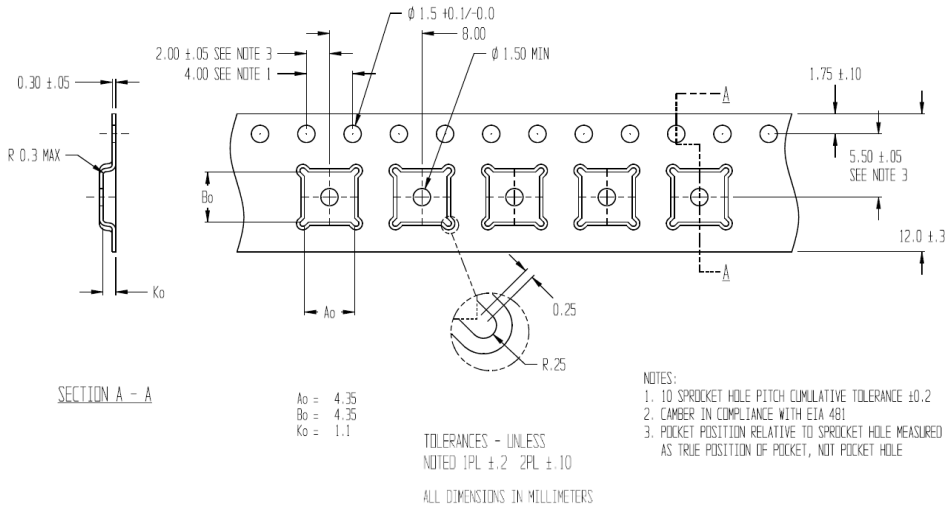
Calculated shelf-life in moisture-sealed bag	12 months -- Storage conditions: <40°C and <90% RH
After opening moisture-sealed bag	168 hours -- Storage conditions: ambient ≤30°C at 60%RH

11.6 Package Marking Specification

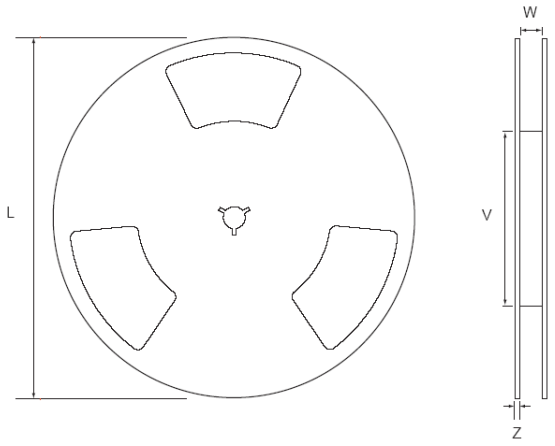


Package Marking Specification

11.7 Tape & Reel Specification



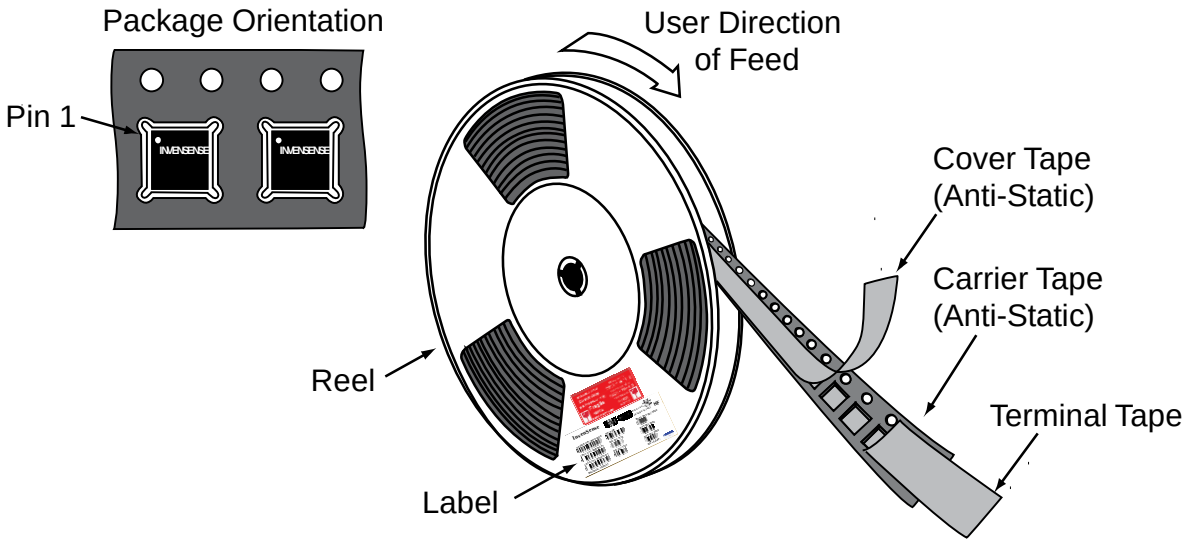
Tape Dimensions



Reel Outline Drawing

Reel Dimensions and Package Size

PACKAGE SIZE	REEL (mm)			
	L	V	W	Z
4x4	330	102	12.8	2.3



Tape and Reel Specification

Reel Specifications

Quantity Per Reel	**5,000**
Reels per Box	**1**
Boxes Per Carton (max)	**5**
Pcs/Carton (max)	**25,000**

11.8 Label

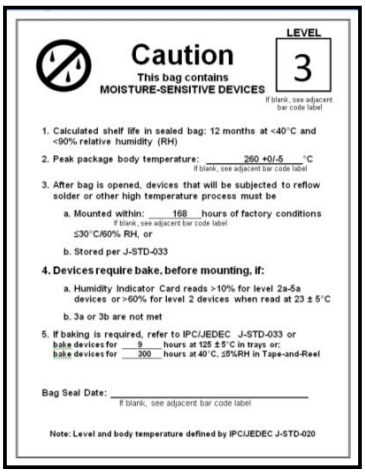
Barcode Label

Location of Label on Reel

11.9 Packaging

REEL – with Barcode & Caution labels

Vacuum-Sealed Moisture Barrier Bag with ESD, MSL3, Caution, and Barcode Labels



MSL3 Label

Caution Label

ESD Label

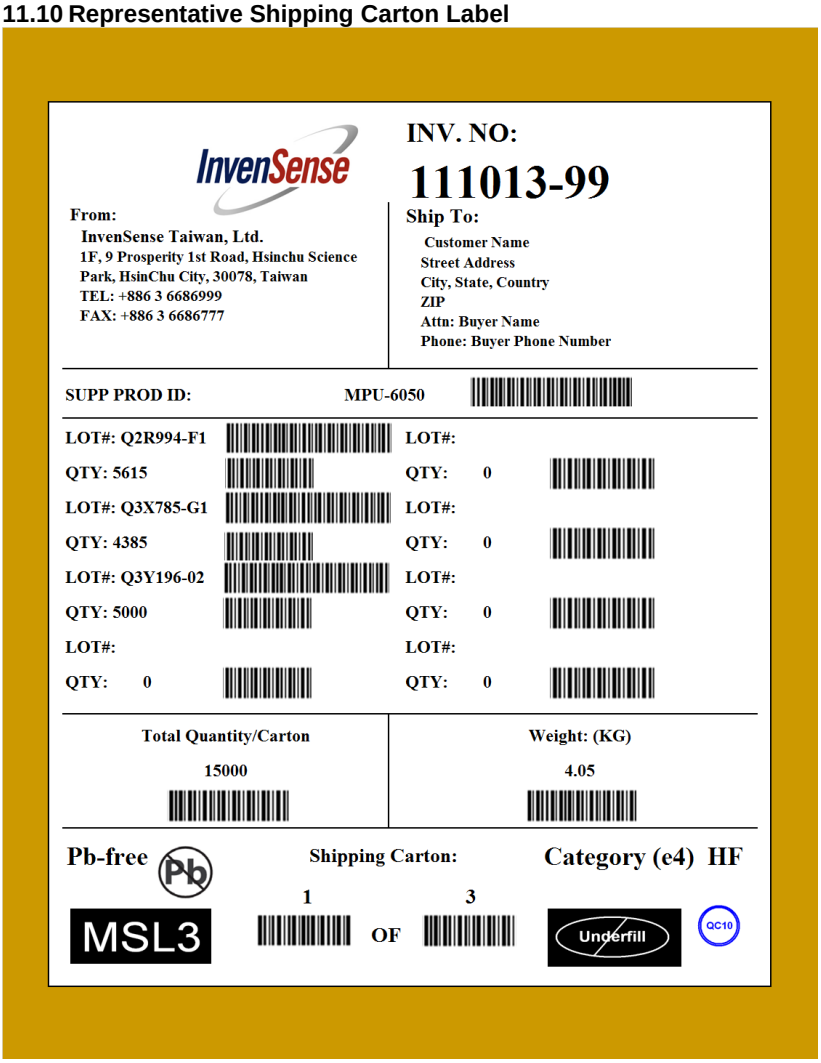
Inner Bubble Wrap

Pizza Box

Pizza Boxes Placed in Foam-Lined Shipper Box

Outer Shipper Label

11.10 Representative Shipping Carton Label



12 Reliability

12.1 Qualification Test Policy

InvenSense's products complete a Qualification Test Plan before being released to production. The Qualification Test Plan for the MPU-60X0 followed the JESD47I Standards, "Stress-Test-Driven Qualification of Integrated Circuits," with the individual tests described below.

12.2 Qualification Test Plan

Accelerated Life Tests

TEST	Method/Condition	Lot Quantity	Sample / Lot	Acc / Reject Criteria
(HTOL/LFR) High Temperature Operating Life	JEDEC JESD22-A108D, Dynamic, 3.63V biased, Tj>125°C [read-points 168, 500, 1000 hours]	3	77	(0/1)
(HAST) Highly Accelerated Stress Test (1)	JEDEC JESD22-A118A Condition A, 130°C, 85%RH, 33.3 psia. unbiased, [read-point 96 hours]	3	77	(0/1)
(HTS) High Temperature Storage Life	JEDEC JESD22-A103D, Cond. A, 125°C Non-Bias Bake [read-points 168, 500, 1000 hours]	3	77	(0/1)

Device Component Level Tests

TEST	Method/Condition	Lot Quantity	Sample / Lot	Acc / Reject Criteria
(ESD-HBM) ESD-Human Body Model	JEDEC JS-001-2012, (2KV)	1	3	(0/1)
(ESD-MM) ESD-Machine Model	JEDEC JESD22-A115C, (250V)	1	3	(0/1)
(LU) Latch Up	JEDEC JESD-78D Class II (2), 125°C; ±100mA	1	6	(0/1)
(MS) Mechanical Shock	JEDEC JESD22-B104C, Mil-Std-883, Method 2002.5, Cond. E, 10,000*g's*, 0.2ms, ±X, Y, Z – 6 directions, 5 times/direction	3	5	(0/1)
(VIB) Vibration	JEDEC JESD22-B103B, Variable Frequency (random), Cond. B, 5-500Hz, X, Y, Z – 4 times/direction	3	5	(0/1)
(TC) Temperature Cycling (1)	JEDEC JESD22-A104D Condition G [-40°C to +125°C], Soak Mode 2 [5'], 1000 cycles	3	77	(0/1)

Board Level Tests

TEST	Method/Condition	Lot Quantity	Sample / Lot	Acc / Reject Criteria
(BMS) Board Mechanical Shock	JEDEC JESD22-B104C,Mil-Std-883, Method 2002.5, Cond. E, 10000*g's*, 0.2ms, +-X, Y, Z – 6 directions, 5 times/direction	1	5	(0/1)
(BTC) Board Temperature Cycling (1)	JEDEC JESD22-A104D Condition G [-40°C to +125°C], Soak mode 2 [5'], 1000 cycles	1	40	(0/1)

(1) Tests are preceded by MSL3 Preconditioning in accordance with JEDEC JESD22-A113F

13 Environmental Compliance

The MPU-6000/MPU-6050 is RoHS and Green compliant.

The MPU-6000/MPU-6050 is in full environmental compliance as evidenced in report HS-MPU-6000, Materials Declaration Data Sheet.

Environmental Declaration Disclaimer:

InvenSense believes this environmental information to be correct but cannot guarantee accuracy or completeness. Conformity documents for the above component constitutes are on file. InvenSense subcontracts manufacturing and the information contained herein is based on data received from vendors and suppliers, which has not been validated by InvenSense.

Click here for production status of specific part numbers.

MAX30102

High-Sensitivity Pulse Oximeter and Heart-Rate Sensor for Wearable Health

General Description

The MAX30102 is an integrated pulse oximetry and heart-rate monitor module. It includes internal LEDs, photodetectors, optical elements, and low-noise electronics with ambient light rejection. The MAX30102 provides a complete system solution to ease the design-in process for mobile and wearable devices.

The MAX30102 operates on a single 1.8V power supply and a separate 3.3V power supply for the internal LEDs. Communication is through a standard I²C-compatible interface. The module can be shut down through software with zero standby current, allowing the power rails to remain powered at all times.

Applications

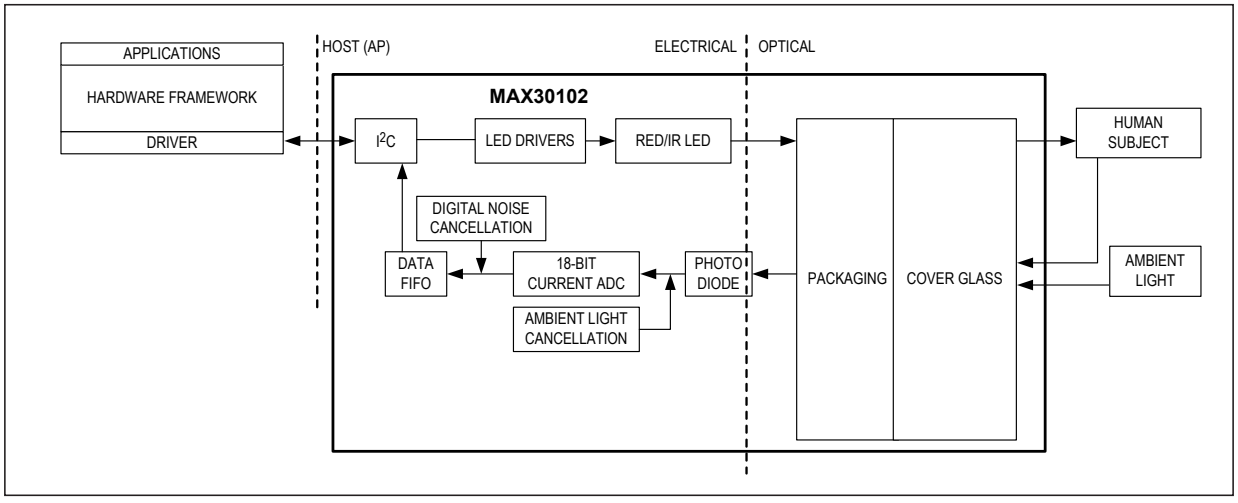
- Wearable Devices
- Fitness Assistant Devices
- Smartphones
- Tablets

Benefits and Features

- Heart-Rate Monitor and Pulse Oximeter Sensor in LED Reflective Solution
- Tiny 5.6mm x 3.3mm x 1.55mm 14-Pin Optical Module
  - Integrated Cover Glass for Optimal, Robust Performance
- Ultra-Low Power Operation for Mobile Devices
  - Programmable Sample Rate and LED Current for Power Savings
  - Low-Power Heart-Rate Monitor (< 1mW)
  - Ultra-Low Shutdown Current (0.7µA, typ)
- Fast Data Output Capability
  - High Sample Rates
- Robust Motion Artifact Resilience
  - High SNR
- -40°C to +85°C Operating Temperature Range

***Ordering Information* appears at end of data sheet.**

System Diagram



19-7740; Rev 1; 10/18

Absolute Maximum Ratings

V_{DD} to GND .....-0.3V to +2.2V
GND to PGND.....-0.3V to +0.3V
V_{LED+} to PGND.....-0.3V to +6.0V
All Other Pins to GND .....-0.3V to +6.0V
Output Short-Circuit Current Duration.....Continuous
Continuous Input Current into Any Terminal .....±20mA
ESD, Human Body Model (HBM).....2.5kV
Latchup Immunity.....±250mA

Continuous Power Dissipation (T_A = +70°C)
OESIP (derate 5.5mW/°C above +70°C) 440mW
Operating Temperature Range..... -40°C to +85°C
Junction Temperature.....+90°C
Soldering Temperature (reflow) .....+260°C
Storage Temperature Range..... -40°C to +105°C

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only; functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Package Information

PACKAGE TYPE: 14 OESIP	
Package Code	F143A5MK+1
Outline Number	**21-1048**
Land Pattern Number	**90-0602**
THERMAL RESISTANCE, FOUR-LAYER BOARD	
Junction to Ambient (θ_{JA})	180°C/W
Junction to Case (θ_{JC})	150°C/W

Package thermal resistances were obtained using the method described in JEDEC specification JESD51-7, using a four-layer board. For detailed information on package thermal considerations, refer to **www.maximintegrated.com/thermal-tutorial**.

For the latest package outline information and land patterns (footprints), go to **www.maximintegrated.com/packages**. Note that a "+", "#", or "-" in the package code indicates RoHS status only. Package drawings may show a different suffix character, but the drawing pertains to the package regardless of RoHS status.

Electrical Characteristics

(V_{DD} = 1.8V, V_{LED+} = 5.0V, T_A = -40°C to +85°C, unless otherwise noted. Typical values are at T_A = +25°C) (Note 1)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
POWER SUPPLY						
Power-Supply Voltage	V_{DD}	Guaranteed by RED and IR count tolerance	1.7	1.8	2.0	V
LED Supply Voltage V_{LED+} to PGND	V_{LED+}	Guaranteed by PSRR of LED driver	3.1	3.3	5.0	V
Supply Current	I_{DD}	SpO_2 and HR mode, PW = 215µs, 50sps		600	1200	µA
		IR only mode, PW = 215µS, 50sps		600	1200	
Supply Current in Shutdown	I_{SHDN}	T_A = +25°C, MODE = 0x80		0.7	10	µA

Electrical Characteristics (continued)

(V_{DD} = 1.8V, V_{LED+} = 5.0V, T_A = -40°C to +85°C, unless otherwise noted. Typical values are at T_A = +25°C) (Note 1)

PARAMETER	SYMBOL	CONDITIONS		MIN	TYP	MAX	UNITS
PULSE OXIMETRY/HEART-RATE SENSOR CHARACTERISTICS							
ADC Resolution					18		bits
Red ADC Count (Note 2)	REDC	LED1_PA = 0x0C, LED_PW = 0x01, SPO2_SR = 0x05, ADC_RGE = 0x00			65536		Counts
IR ADC Count (Note 2)	IRC	LED2_PA = 0x0C, LED_PW = 0x01, SPO2_SR = 0x05 ADC_RGE = 0x00			65536		Counts
Dark Current Count	LED_DCC	LED1_PA = LED2_PA = 0x00, LED_PW = 0x03, SPO2_SR = 0x01 ADC_RGE = 0x02			30	128	Counts
					0.01	0.05	% of FS
DC Ambient Light Rejection	ALR	ADC counts with finger on sensor under direct sunlight (100K lux), ADC_RGE = 0x3, LED_PW = 0x03, SPO2_SR = 0x01	Red LED		2		Counts
			IR LED		2		Counts
ADC Count—PSRR (V_{DD})	$PSRRV_{DD}$	1.7V < V_{DD} < 2.0V, LED_PW = 0x01, SPO2_SR = 0x05			0.25	1	% of FS
		Frequency = DC to 100kHz, 100mV_{P-P}			10		LSB
ADC Count—PSRR (LED Driver Outputs)	$PSRR_{LED}$	3.1V < V_{LED+}, < 5.0V, LED1_PA = LED2_PA = 0x0C, LED_PW = 0x01, SPO2_SR = 0x05			0.05	1	% of FS
		Frequency = DC to 100kHz, 100mV_{P-P}			10		LSB
ADC Clock Frequency	CLK			10.32	10.48	10.64	MHz
ADC Integration Time	INT	LED_PW = 0x00			69		µs
		LED_PW = 0x01			118		
		LED_PW = 0x02			215		
		LED_PW = 0x03			411		
Slot Timing (Timing Between Sequential Channel Samples; e.g., Red Pulse Rising Edge To IR Pulse Rising Edge)	INT	LED_PW = 0x00			427.1		µs
		LED_PW = 0x01			524.7		
		LED_PW = 0x02			720.0		
		LED_PW = 0x03			1106.6		
COVER GLASS CHARACTERISTICS (Note 3)							
Hydrolytic Resistance Class		Per DIN ISO 719			HGB 1		

Electrical Characteristics (continued)

(V_{DD} = 1.8V, V_{LED+} = 5.0V, T_A = -40°C to +85°C, unless otherwise noted. Typical values are at T_A = +25°C) (Note 1)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
IR LED CHARACTERISTICS (Note 3)						
LED Peak Wavelength	λ_P	I_{LED} = 20mA, T_A = +25°C	870	880	900	nm
Full Width at Half Max	$\Delta\lambda$	I_{LED} = 20mA, T_A = +25°C		30		nm
Forward Voltage	V_F	I_{LED} = 20mA, T_A = +25°C		1.4		V
Radiant Power	P_O	I_{LED} = 20mA, T_A = +25°C		6.5		mW
RED LED CHARACTERISTICS (Note 3)						
LED Peak Wavelength	λ_P	I_{LED} = 20mA, T_A = +25°C	650	660	670	nm
Full Width at Half Max	$\Delta\lambda$	I_{LED} = 20mA, T_A = +25°C		20		nm
Forward Voltage	V_F	I_{LED} = 20mA, T_A = +25°C		2.1		V
Radiant Power	P_O	I_{LED} = 20mA, T_A = +25°C		9.8		mW
PHOTODETECTOR CHARACTERISTICS (Note 3)						
Spectral Range of Sensitivity	λ (QE > 50%)	QE: Quantum Efficiency	600		900	nm
Radiant Sensitive Area	A			1.36		mm^2
Dimensions of Radiant Sensitive Area	L x W			1.38 x 0.98		mm x mm
INTERNAL DIE TEMPERATURE SENSOR						
Temperature ADC Acquisition Time	T_T	T_A = +25°C		29		ms
Temperature Sensor Accuracy	T_A	T_A = +25°C		±1		°C
Temperature Sensor Minimum Range	T_{MIN}			-40		°C
Temperature Sensor Maximum Range	T_{MAX}			85		°C
DIGITAL INPUT CHARACTERISTICS: SCL, SDA						
Input High Voltage	V_{IH}	V_{DD} = 2V	0.7 x V_{DD}			V
Input Low Voltage	V_{IL}	V_{DD} = 2V			0.3 x V_{DD}	V
Hysteresis Voltage	V_H			0.2		V
Input Leakage Current	I_{IN}	V_{IN} = GND or V_{DD} (STATIC)		±0.05	±1	µA
DIGITAL OUTPUT CHARACTERISTICS: SDA, $\overline{INT}$						
Ouput Low Voltage	V_{OL}	I_{SINK} = 6mA			0.2	V

Electrical Characteristics (continued)

(V_{DD} = 1.8V, V_{LED+} = 5.0V, T_A = -40°C to +85°C, unless otherwise noted. Typical values are at T_A = +25°C) (Note 1)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
I²C TIMING CHARACTERISTICS (SDA, SDA, $\overline{INT}$) (Note 3)						
I²C Write Address				AE		Hex
I²C Read Address				AF		Hex
Serial Clock Frequency	f_{SCL}		0		400	kHz
Bus Free Time Between STOP and START Conditions	t_{BUF}		1.3			µs
Hold Time (Repeated) START Condition	$t_{HD;STA}$		0.6			µs
SCL Pulse-Width Low	t_{LOW}		1.3			µs
SCL Pulse-Width High	t_{HIGH}		0.6			µs
Setup Time for a Repeated START Condition	$t_{SU;STA}$		0.6			µs
Data Hold Time	$t_{HD;DAT}$		0		900	ns
Data Setup Time	$t_{SU;DAT}$		100			ns
Setup Time for STOP Condition	$t_{SU;STO}$		0.6			µs
Pulse Width of Suppressed Spike	t_{SP}		0		50	ns
Bus Capacitance	C_B				400	pF
SDA and SCL Receiving Rise Time	t_R		20 + 0.1C_B		300	ns
SDA and SCL Receiving Fall Time	t_{RF}		20 + 0.1C_B		300	ns
SDA Transmitting Fall Time	t_{TF}				300	ns

Note 1: All devices are 100% production tested at T_A = +25°C. Specifications over temperature limits are guaranteed by Maxim Integrated's bench or proprietary automated test equipment (ATE) characterization.

Note 2: Specifications are guaranteed by Maxim Integrated's bench characterization and by 100% production test using proprietary ATE setup and conditions.

Note 3: Guaranteed by design and characterization. Not tested in final production.

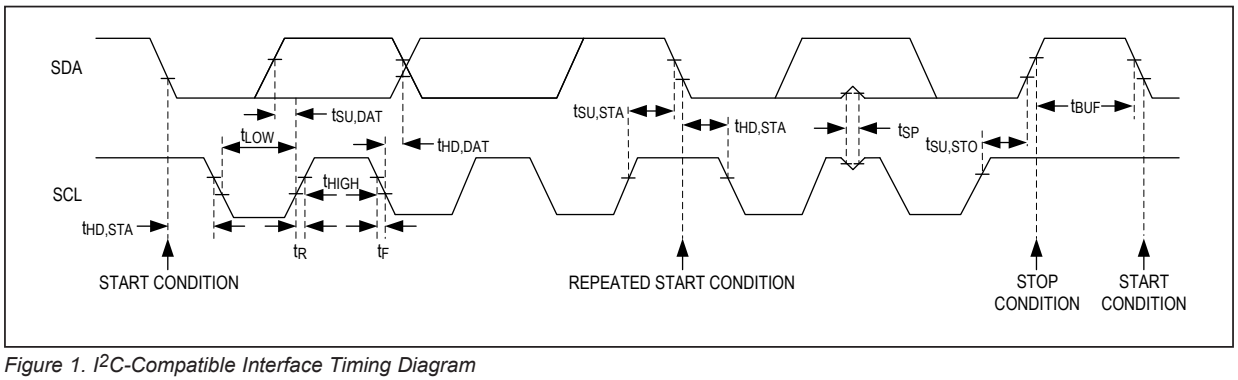
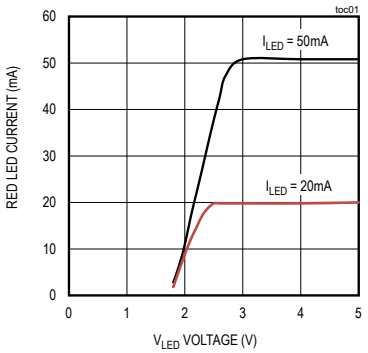


Figure 1. I²C-Compatible Interface Timing Diagram

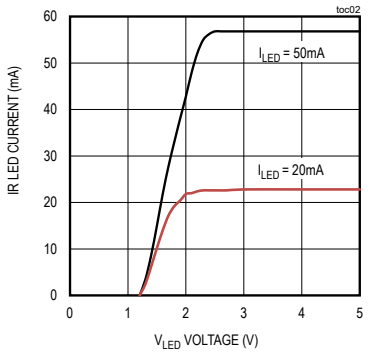
Typical Operating Characteristics

(V_{DD} = 1.8V, V_{LED+} = 5.0V, T_A = +25°C, $\overline{RST}$, unless otherwise noted.)

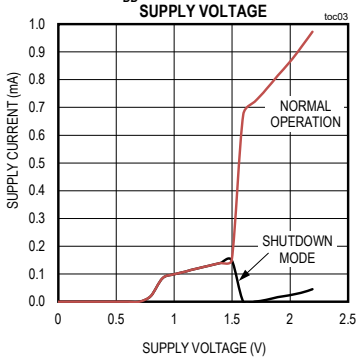
RED LED SUPPLY HEADROOM



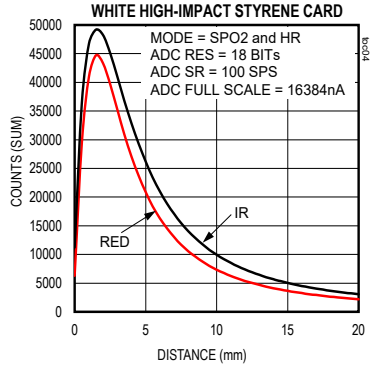
IR LED SUPPLY HEADROOM



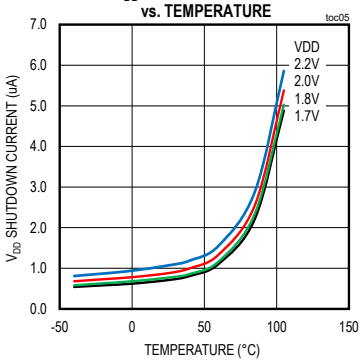
V_{DD} SUPPLY CURRENT vs. SUPPLY VOLTAGE



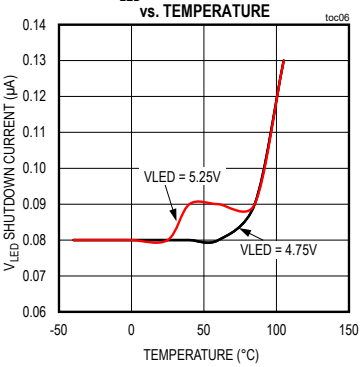
DC COUNTS vs. DISTANCE FOR WHITE HIGH-IMPACT STYRENE CARD



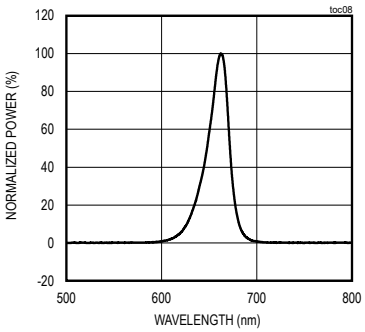
V_{DD} SHUTDOWN CURRENT vs. TEMPERATURE



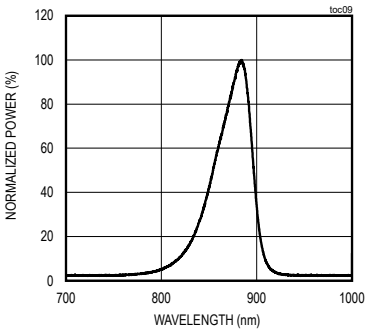
V_{LED} SHUTDOWN CURRENT vs. TEMPERATURE



RED LED SPECTRUM AT T_A = +30°C



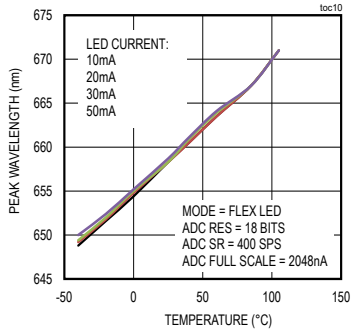
IR LED SPECTRUM AT T_A = +30°C



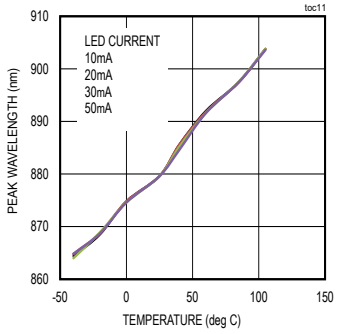
Typical Operating Characteristics (continued)

(V_{DD} = 1.8V, V_{LED+} = 5.0V, T_A = +25°C, $\overline{RST}$, unless otherwise noted.)

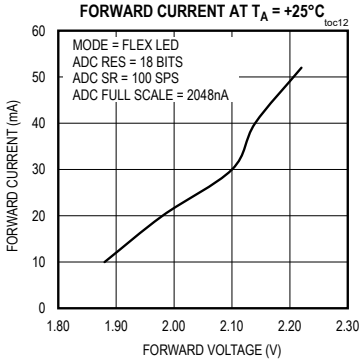
RED LED PEAK WAVELENGTH vs. TEMPERATURE



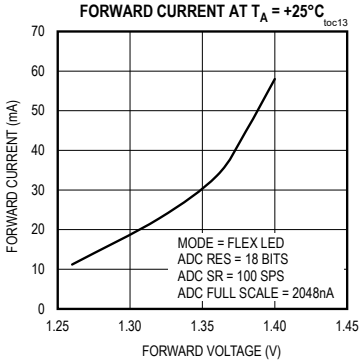
IR LED PEAK WAVELENGTH vs. TEMPERATURE



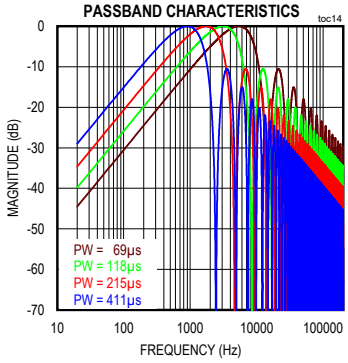
RED LED FORWARD VOLTAGE vs. FORWARD CURRENT AT T_A = +25°C



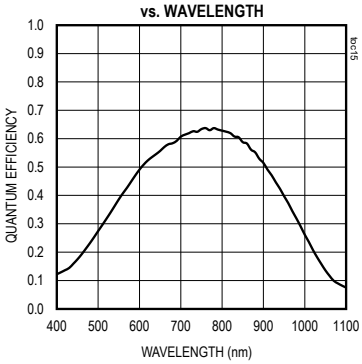
IR LED FORWARD VOLTAGE vs. FORWARD CURRENT AT T_A = +25°C



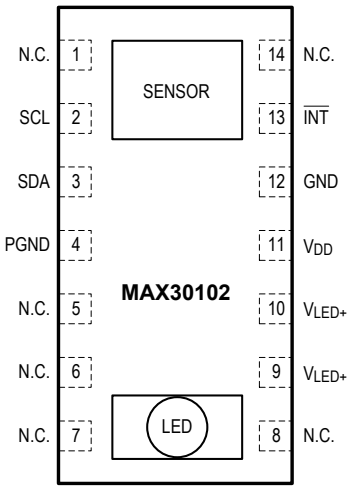
AMBIENT LIGHT CANCELLATION PASSBAND CHARACTERISTICS



PHOTODIODE QUANTUM EFFICIENCY vs. WAVELENGTH



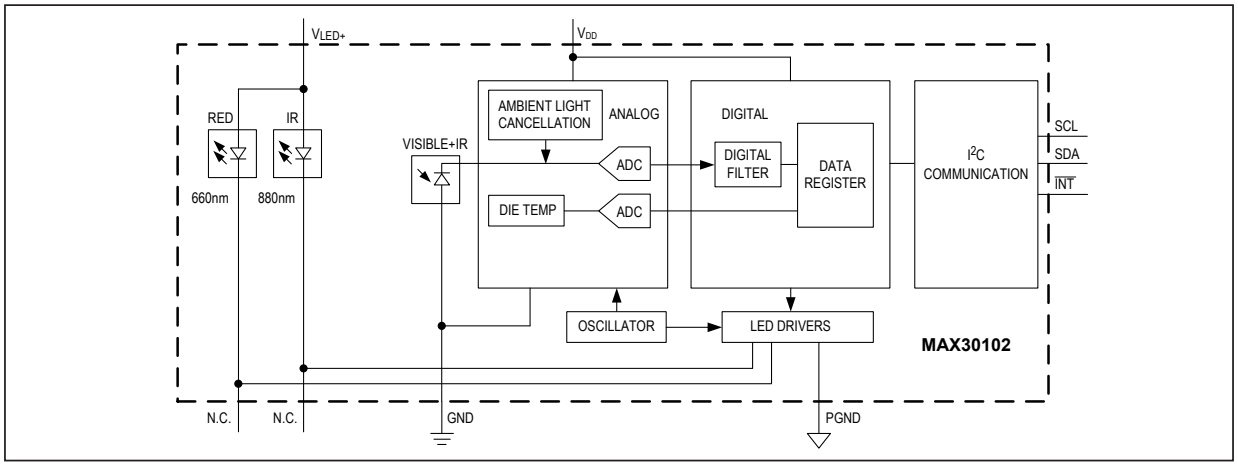
Pin Configuration



Pin Description

PIN	NAME	FUNCTION
1, 5, 6, 7, 8, 14	N.C.	No Connection. Connect to PCB pad for mechanical stability.
2	SCL	I^2C Clock Input
3	SDA	I^2C Data, Bidirectional (Open-Drain)
4	PGND	Power Ground of the LED Driver Blocks
9	V_{LED+}	LED Power Supply (anode connection). Use a bypass capacitor to PGND for best performance.
10	V_{LED+}	
11	V_{DD}	Analog Power Supply Input. Use a bypass capacitor to GND for best performance.
12	GND	Analog Ground
13	$\overline{INT}$	Active-Low Interrupt (Open-Drain). Connect to an external voltage with a pullup resistor.

Functional Diagram



Detailed Description

The MAX30102 is a complete pulse oximetry and heart-rate sensor system solution module designed for the demanding requirements of wearable devices. The device maintains a very small solution size without sacrificing optical or electrical performance. Minimal external hardware components are required for integration into a wearable system.

The MAX30102 is fully adjustable through software registers, and the digital output data can be stored in a 32-deep FIFO within the IC. The FIFO allows the MAX30102 to be connected to a microcontroller or processor on a shared bus, where the data is not being read continuously from the MAX30102's registers.

SpO_2 Subsystem

The SpO_2 subsystem of the MAX30102 contains ambient light cancellation (ALC), a continuous-time sigma-delta ADC, and a proprietary discrete time filter. The ALC has an internal Track/Hold circuit to cancel ambient light and increase the effective dynamic range. The SpO_2 ADC has programmable full-scale ranges from 2µA to 16µA. The ALC can cancel up to 200µA of ambient current.

The internal ADC is a continuous time oversampling sigma-delta converter with 18-bit resolution. The ADC sampling rate is 10.24MHz. The ADC output data rate can be programmed from 50sps (samples per second) to 3200sps.

Temperature Sensor

The MAX30102 has an on-chip temperature sensor for calibrating the temperature dependence of the SpO_2 subsystem. The temperature sensor has an inherent resolution of 0.0625°C.

The device output data is relatively insensitive to the wavelength of the IR LED, where the Red LED's wavelength is critical to correct interpretation of the data. An SpO_2 algorithm used with the MAX30102 output signal can compensate for the associated SpO_2 error with ambient temperature changes.

LED Driver

The MAX30102 integrates Red and IR LED drivers to modulate LED pulses for SpO_2 and HR measurements. The LED current can be programmed from 0 to 50mA with proper supply voltage. The LED pulse width can be programmed from 69µs to 411µs to allow the algorithm to optimize SpO_2 and HR accuracy and power consumption based on use cases.

Register Maps and Descriptions

REGISTER	B7	B6	B5	B4	B3	B2	B1	B0	REG ADDR	POR STATE	R/W
STATUS											
Interrupt Status 1	A_FULL	PPG_ RDY	ALC_ OVF					PWR_ RDY	0x00	0X00	R
Interrupt Status 2							DIE_TEMP _RDY		0x01	0x00	R
Interrupt Enable 1	A_FULL_ EN	PPG_ RDY_EN	ALC_ OVF_EN						0x02	0X00	R/W
Interrupt Enable 2							DIE_TEMP _RDY_EN		0x03	0x00	R/W
FIFO											
FIFO Write Pointer				FIFO_WR_PTR[4:0]					0x04	0x00	R/W
Overflow Counter				OVF_COUNTER[4:0]					0x05	0x00	R/W
FIFO Read Pointer				FIFO_RD_PTR[4:0]					0x06	0x00	R/W
FIFO Data Register	FIFO_DATA[7:0]								0x07	0x00	R/W
CONFIGURATION											
FIFO Configuration	SMP_AVE[2:0]			FIFO_ ROLL OVER_EN	FIFO_A_FULL[3:0]				0x08	0x00	R/W
Mode Configuration	SHDN	RESET				MODE[2:0]			0x09	0x00	R/W
SpO_2 Configuration	0 (Reserved)	SPO2_ADC_RGE [1:0]		SPO2_SR[2:0]			LED_PW[1:0]		0x0A	0x00	R/W
RESERVED									0x0B	0x00	R/W
LED Pulse Amplitude	LED1_PA[7:0]								0x0C	0x00	R/W
	LED2_PA[7:0]								0x0D	0x00	R/W
RESERVED									0x0E	0x00	R/W
RESERVED									0x0F	0x00	R/W
Multi-LED Mode Control Registers		SLOT2[2:0]				SLOT1[2:0]			0x11	0x00	R/W
		SLOT4[2:0]				SLOT3[2:0]			0x12	0x00	R/W

Register Maps and Descriptions (continued)

REGISTER	B7	B6	B5	B4	B3	B2	B1	B0	REG ADDR	POR STATE	R/W
RESERVED									0x13–0x17	0xFF	R/W
RESERVED									0x18-0x1E	0x00	R
DIE TEMPERATURE											
Die Temp Integer	TINT[7:0]								0x1F	0x00	R
Die Temp Fraction					TFRAC[3:0]				0x20	0x00	R
Die Temperature Config								TEMP_EN	0x21	0x00	R/W
RESERVED									0x22–0x2F	0x00	R/W
PART ID											
Revision ID	REV_ID[7:0]								0xFE	0xXX*	R
Part ID	PART_ID[7]								0xFF	0x15	R

**XX denotes a 2-digit hexadecimal number (00 to FF) for part revision identification. Contact Maxim Integrated for the revision ID number assigned for your product.*

Interrupt Status (0x00–0x01)

REGISTER	B7	B6	B5	B4	B3	B2	B1	B0	REG ADDR	POR STATE	R/W
Interrupt Status 1	A_FULL	PPG_RDY	ALC_OVF					PWR_ RDY	0x00	0X00	R
Interrupt Status 2							DIE_ TEMP_RDY		0x01	0x00	R

Whenever an interrupt is triggered, the MAX30102 pulls the active-low interrupt pin into its low state until the interrupt is cleared.

A_FULL: FIFO Almost Full Flag

In SpO_2 and HR modes, this interrupt triggers when the FIFO write pointer has a certain number of free spaces remaining. The trigger number can be set by the FIFO_A_FULL[3:0] register. The interrupt is cleared by reading the Interrupt Status 1 register (0x00).

PPG_RDY: New FIFO Data Ready

In SpO_2 and HR modes, this interrupt triggers when there is a new sample in the data FIFO. The interrupt is cleared by reading the Interrupt Status 1 register (0x00), or by reading the FIFO_DATA register.

ALC_OVF: Ambient Light Cancellation Overflow

This interrupt triggers when the ambient light cancellation function of the SpO_2/HR photodiode has reached its maximum limit, and therefore, ambient light is affecting the output of the ADC. The interrupt is cleared by reading the Interrupt Status 1 register (0x00).

PWR_RDY: Power Ready Flag

On power-up or after a brownout condition, when the supply voltage V_{DD} transitions from below the undervoltage lockout (UVLO) voltage to above the UVLO voltage, a power-ready interrupt is triggered to signal that the module is powered-up and ready to collect data.

DIE_TEMP_RDY: Internal Temperature Ready Flag

When an internal die temperature conversion is finished, this interrupt is triggered so the processor can read the temperature data registers. The interrupt is cleared by reading either the Interrupt Status 2 register (0x01) or the TFRAC register (0x20).

The interrupts are cleared whenever the interrupt status register is read, or when the register that triggered the interrupt is read. For example, if the SpO_2 sensor triggers an interrupt due to finishing a conversion, reading either the FIFO data register or the interrupt register clears the interrupt pin (which returns to its normal HIGH state). This also clears all the bits in the interrupt status register to zero.

Interrupt Enable (0x02-0x03)

REGISTER	B7	B6	B5	B4	B3	B2	B1	B0	REG ADDR	POR STATE	R/W
Interrupt Enable 1	A_FULL_EN	PPG_RDY_EN	ALC_OVF_EN						0x02	0X00	R/W
Interrupt Enable 2							DIE_TEMP_RDY_EN		0x03	0x00	R/W

Each source of hardware interrupt, with the exception of power ready, can be disabled in a software register within the MAX30102 IC. The power-ready interrupt cannot be disabled because the digital state of the module is reset upon a brownout condition (low power supply voltage), and the default condition is that all the interrupts are disabled. Also, it is important for the system to know that a brownout condition has occurred, and the data within the module is reset as a result.

The unused bits should always be set to zero for normal operation.

FIFO (0x04–0x07)

REGISTER	B7	B6	B5	B4	B3	B2	B1	B0	REG ADDR	POR STATE	R/W
FIFO Write Pointer				FIFO_WR_PTR[4:0]					0x04	0x00	R/W
Over Flow Counter				OVF_COUNTER[4:0]					0x05	0x00	R/W
FIFO Read Pointer				FIFO_RD_PTR[4:0]					0x06	0x00	R/W
FIFO Data Register	FIFO_DATA[7:0]								0x07	0x00	R/W

FIFO Write Pointer

The FIFO Write Pointer points to the location where the MAX30102 writes the next sample. This pointer advances for each sample pushed on to the FIFO. It can also be changed through the I^2C interface when MODE[2:0] is 010, 011, or 111.

FIFO Overflow Counter

When the FIFO is full, samples are not pushed on to the FIFO, samples are lost. OVF_COUNTER counts the number of samples lost. It saturates at 0x1F. When a complete sample is “popped” (i.e., removal of old FIFO data and shifting the samples down) from the FIFO (when the read pointer advances), OVF_COUNTER is reset to zero.

FIFO Read Pointer

The FIFO Read Pointer points to the location from where the processor gets the next sample from the FIFO through the I^2C interface. This advances each time a sample is popped from the FIFO. The processor can also write to this pointer after reading the samples to allow rereading samples from the FIFO if there is a data communication error.

FIFO Data Register

The circular FIFO depth is 32 and can hold up to 32 samples of data. The sample size depends on the number of LED channels (a.k.a. channels) configured as active. As each channel signal is stored as a 3-byte data signal, the FIFO width can be 3 bytes or 6 bytes in size.

The FIFO_DATA register in the I2C register map points to the next sample to be read from the FIFO. FIFO_RD_PTR points to this sample. Reading FIFO_DATA register, does not automatically increment the I2C register address. Burst reading this register, reads the same address over and over. Each sample is 3 bytes of data per channel (i.e., 3 bytes for RED, 3 bytes for IR, etc.).

The FIFO registers (0x04–0x07) can all be written and read, but in practice only the FIFO_RD_PTR register should be written to in operation. The others are automatically incremented or filled with data by the MAX30102. When starting a new SpO_2 or heart rate conversion, it is recommended to first clear the FIFO_WR_PTR, OVF_COUNTER, and FIFO_RD_PTR registers to all zeroes (0x00) to ensure the FIFO is empty and in a known state. When reading the MAX30102 registers in one burst-read I2C transaction, the register address pointer typically increments so that the next byte of data sent is from the next register, etc. The exception to this is the FIFO data register, register 0x07. When reading this register, the address pointer does not increment, but the FIFO_RD_PTR does. So the next byte of data sent represents the next byte of data available in the FIFO.

Reading from the FIFO

Normally, reading registers from the I2C interface autoincrements the register address pointer, so that all the registers can be read in a burst read without an I2C start event. In the MAX30102, this holds true for all registers except for the FIFO_DATA register (register 0x07).

Reading the FIFO_DATA register does not automatically increment the register address. Burst reading this register reads data from the same address over and over. Each sample comprises multiple bytes of data, so multiple bytes should be read from this register (in the same transaction) to get one full sample.

The other exception is 0xFF. Reading more bytes after the 0xFF register does not advance the address pointer back to 0x00, and the data read is not meaningful.

FIFO Data Structure

The data FIFO consists of a 32-sample memory bank that can store IR and Red ADC data. Since each sample consists of two channels of data, there are 6 bytes of data for each sample, and therefore 192 total bytes of data can be stored in the FIFO.

The FIFO data is left-justified as shown in Table 1; in other words, the MSB bit is always in the bit 17 data position regardless of ADC resolution setting. See Table 2 for a visual presentation of the FIFO data structure.

Table 1. FIFO Data is Left-Justified

ADC Resolution	FIFO_DATA[17]	FIFO_DATA[16]	…	FIFO_DATA[12]	FIFO_DATA[11]	FIFO_DATA[10]	FIFO_DATA[9]	FIFO_DATA[8]	FIFO_DATA[7]	FIFO_DATA[6]	FIFO_DATA[5]	FIFO_DATA[4]	FIFO_DATA[3]	FIFO_DATA[2]	FIFO_DATA[1]	FIFO_DATA[0]
18-bit																
17-bit																
16-bit																
15-bit																

FIFO Data Contains 3 Bytes per Channel

The FIFO data is left-justified, meaning that the MSB is always in the same location regardless of the ADC resolution setting. FIFO DATA[18] – [23] are not used. Table 2 shows the structure of each triplet of bytes (containing the 18-bit ADC data output of each channel).

Each data sample in SpO_2 mode comprises two data triplets (3 bytes each), To read one sample, requires an I^2C read command for each byte. Thus, to read one sample in SpO_2 mode, requires 6 I^2C byte reads. The FIFO read pointer is automatically incremented after the first byte of each sample is read.

Write/Read Pointers

Write/Read pointers are used to control the flow of data in the FIFO. The write pointer increments every time a new sample is added to the FIFO. The read pointer is incremented every time a sample is read from the FIFO. To reread a sample from the FIFO, decrement its value by one and read the data register again.

The FIFO write/read pointers should be cleared (back to 0x00) upon entering SpO_2 mode or HR mode, so that there is no old data represented in the FIFO. The pointers are automatically cleared if V_{DD} is power-cycled or V_{DD} drops below its UVLO voltage.

BYTE 1							FIFO_ DATA[17]	FIFO_ DATA[16]
BYTE 2	FIFO_ DATA[15]	FIFO_ DATA[14]	FIFO_ DATA[13]	FIFO_ DATA[12]	FIFO_ DATA[11]	FIFO_ DATA[10]	FIFO_ DATA[9]	FIFO_ DATA[8]
BYTE 3	FIFO_ DATA[7]	FIFO_ DATA[6]	FIFO_ DATA[5]	FIFO_ DATA[4]	FIFO_ DATA[3]	FIFO_ DATA[2]	FIFO_ DATA[1]	FIFO_ DATA[0]

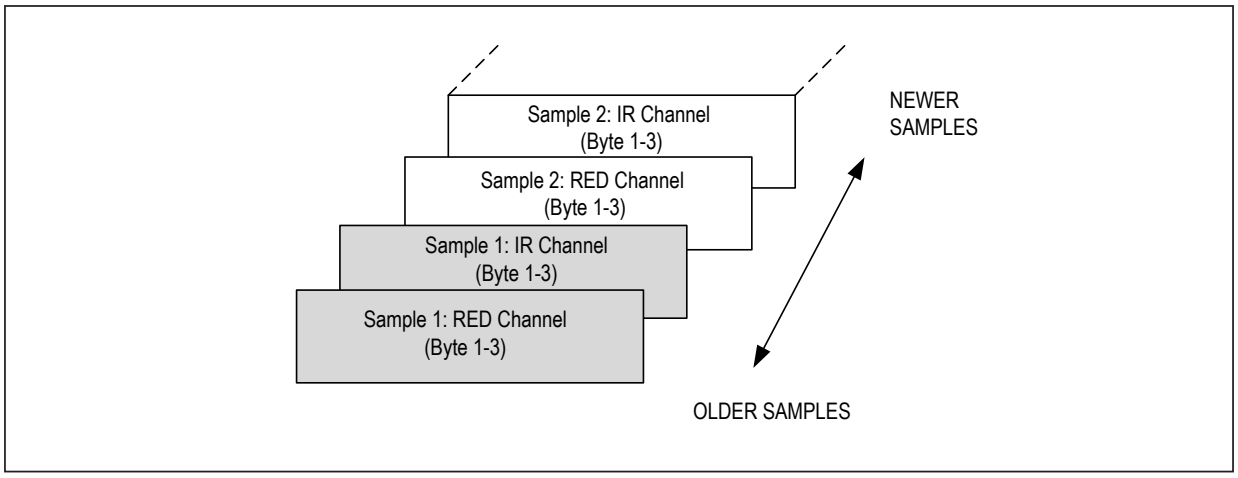


Figure 2. Graphical Representation of the FIFO Data Register. It shows IR and Red in SpO_2 Mode.

Pseudo-Code Example of Reading Data from FIFO

First transaction: Get the FIFO_WR_PTR:

```
START;
Send device address + write mode
Send address of FIFO_WR_PTR;
REPEATED_START;
Send device address + read mode
Read FIFO_WR_PTR;
STOP;
```

The central processor evaluates the number of samples to be read from the FIFO:

```
NUM_AVAILABLE_SAMPLES = FIFO_WR_PTR – FIFO_RD_PTR
(Note: pointer wrap around should be taken into account)
NUM_SAMPLES_TO_READ = < less than or equal to NUM_AVAILABLE_SAMPLES >
```

Second transaction: Read `NUM_SAMPLES_TO_READ` samples from the FIFO:

```
START;
Send device address + write mode
Send address of FIFO_DATA;
REPEATED_START;
Send device address + read mode
for (i = 0; i < NUM_SAMPLES_TO_READ; i++) {
Read FIFO_DATA;
Save LED1[23:16];
Read FIFO_DATA;
Save LED1[15:8];
Read FIFO_DATA;
Save LED1[7:0];
Read FIFO_DATA;
Save LED2[23:16];
Read FIFO_DATA;
Save LED2[15:8];
Read FIFO_DATA;
Save LED2[7:0];
Read FIFO_DATA;
}
STOP;
START;
Send device address + write mode
Send address of FIFO_RD_PTR;
Write FIFO_RD_PTR;
STOP;
```

Third transaction: Write to FIFO_RD_PTR register. If the second transaction was successful, FIFO_RD_PTR points to the next sample in the FIFO, and this third transaction is not necessary. Otherwise, the processor updates the FIFO_RD_PTR appropriately, so that the samples are reread.

FIFO Configuration (0x08)

REGISTER	B7	B6	B5	B4	B3	B2	B1	B0	REG ADDR	POR STATE	R/W
FIFO Configuration	SMP_AVE[2:0]			FIFO_ROLLOVER_EN	FIFO_A_FULL[3:0]				0x08	0x00	R/W

Bits 7:5: Sample Averaging (SMP_AVE)

To reduce the amount of data throughput, adjacent samples (in each individual channel) can be averaged and decimated on the chip by setting this register.

Table 3. Sample Averaging

SMP_AVE[2:0]	NO. OF SAMPLES AVERAGED PER FIFO SAMPLE
000	1 (no averaging)
001	2
010	4
011	8
100	16
101	32
110	32
111	32

Bit 4: FIFO Rolls on Full (FIFO_ROLLOVER_EN)

This bit controls the behavior of the FIFO when the FIFO becomes completely filled with data. If FIFO_ROLLOVER_EN is set (1), the FIFO address rolls over to zero and the FIFO continues to fill with new data. If the bit is not set (0), then the FIFO is not updated until FIFO_DATA is read or the WRITE/READ pointer positions are changed.

Bits 3:0: FIFO Almost Full Value (FIFO_A_FULL)

This register sets the number of data samples (3 bytes/sample) remaining in the FIFO when the interrupt is issued. For example, if this field is set to 0x0, the interrupt is issued when there is 0 data samples remaining in the FIFO (all 32 FIFO words have unread data). Furthermore, if this field is set to 0xF, the interrupt is issued when 15 data samples are remaining in the FIFO (17 FIFO data samples have unread data).

FIFO_A_FULL[3:0]	EMPTY DATA SAMPLES IN FIFO WHEN INTERRUPT IS ISSUED	UNREAD DATA SAMPLES IN FIFO WHEN INTERRUPT IS ISSUED
0x0h	0	32
0x1h	1	31
0x2h	2	30
0x3h	3	29
…	…	…
0xFh	15	17

Mode Configuration (0x09)

REGISTER	B7	B6	B5	B4	B3	B2	B1	B0	REG ADDR	POR STATE	R/W
Mode Configuration	SHDN	RESET				MODE[2:0]			0x09	0x00	R/W

Bit 7: Shutdown Control (SHDN)

The part can be put into a power-save mode by setting this bit to one. While in power-save mode, all registers retain their values, and write/read operations function as normal. All interrupts are cleared to zero in this mode.

Bit 6: Reset Control (RESET)

When the RESET bit is set to one, all configuration, threshold, and data registers are reset to their power-on-state through a power-on reset. The RESET bit is cleared automatically back to zero after the reset sequence is completed.

Note: Setting the RESET bit does not trigger a PWR_RDY interrupt event.

Bits 2:0: Mode Control

These bits set the operating state of the MAX30102. Changing modes does not change any other setting, nor does it erase any previously stored data inside the data registers.

Table 4. Mode Control

MODE[2:0]	MODE	ACTIVE LED CHANNELS
000	Do not use	
001	Do not use	
010	Heart Rate mode	Red only
011	SpO_2 mode	Red and IR
100–110	Do not use	
111	Multi-LED mode	Red and IR

SpO_2 Configuration (0x0A)

REGISTER	B7	B6	B5	B4	B3	B2	B1	B0	REG ADDR	POR STATE	R/W
SpO_2 Configuration		SPO2_ADC_RGE[1:0]		SPO2_SR[2:0]			LED_PW[1:0]		0x0A	0x00	R/W

Bits 6:5: SpO_2 ADC Range Control

This register sets the SpO_2 sensor ADC's full-scale range as shown in Table 5.

Table 5. SpO_2 ADC Range Control (18-Bit Resolution)

SPO2_ADC_RGE[1:0]	LSB SIZE (pA)	FULL SCALE (nA)
00	7.81	2048
01	15.63	4096
10	31.25	8192
11	62.5	16384

Bits 4:2: SpO_2 Sample Rate Control

These bits define the effective sampling rate with one sample consisting of one IR pulse/conversion and one Red pulse/conversion.

The sample rate and pulse width are related in that the sample rate sets an upper bound on the pulse width time. If the user selects a sample rate that is too high for the selected LED_PW setting, the highest possible sample rate is programmed instead into the register.

Table 6. SpO_2 Sample Rate Control

SPO2_SR[2:0]	SAMPLES PER SECOND
000	50
001	100
010	200
011	400
100	800
101	1000
110	1600
111	3200

See Table 11 and Table 12 for Pulse Width vs. Sample Rate information.

Bits 1:0: LED Pulse Width Control and ADC Resolution

These bits set the LED pulse width (the IR and Red have the same pulse width), and therefore, indirectly sets the integration time of the ADC in each sample. The ADC resolution is directly related to the integration time.

Table 7. LED Pulse Width Control

LED_PW[1:0]	PULSE WIDTH (µs)	ADC RESOLUTION (bits)
00	69 (68.95)	15
01	118 (117.78)	16
10	215 (215.44)	17
11	411 (410.75)	18

LED Pulse Amplitude (0x0C–0x0D)

REGISTER	B7	B6	B5	B4	B3	B2	B1	B0	REG ADDR	POR STATE	R/W
LED Pulse Amplitude	LED1_PA[7:0]								0x0C	0x00	R/W
	LED2_PA[7:0]								0x0D	0x00	R/W

These bits set the current level of each LED as shown in Table 8.

Table 8. LED Current Control

LEDx_PA [7:0], RED_PA[7:0], or IR_PA[7:0]	TYPICAL LED CURRENT (mA)*
0x00h	0.0
0x01h	0.2
0x02h	0.4
…	…
0x0Fh	3.0
…	…
0x1Fh	6.2
…	…
0x3Fh	12.6
…	…
0x7Fh	25.4
…	…
0xFFh	51.0

**Actual measured LED current for each part can vary widely due to the trimming methodology.*

Multi-LED Mode Control Registers (0x11–0x12)

REGISTER	B7	B6	B5	B4	B3	B2	B1	B0	REG ADDR	POR STATE	R/W
Multi-LED Mode Control Registers		SLOT2[2:0]				SLOT1[2:0]			0x11	0x00	R/W
		SLOT4[2:0]				SLOT3[2:0]			0x12	0x00	R/W

In multi-LED mode, each sample is split into up to four time slots, SLOT1 through SLOT4. These control registers determine which LED is active in each time slot, making for a very flexible configuration.

Table 9. Multi-LED Mode Control Registers

SLOTx[2:0] Setting	WHICH LED IS ACTIVE	LED PULSE AMPLITUDE SETTING
000	None (time slot is disabled)	N/A (Off)
001	LED1 (Red)	LED1_PA[7:0]
010	LED2 (IR)	LED2_PA[7:0]
011	None	N/A (Off)
100	None	N/A (Off)
101	Reserved	Reserved
110	Reserved	Reserved
111	Reserved	Reserved

Each slot generates a 3-byte output into the FIFO. One sample comprises all active slots, for example if SLOT1 and SLOT2 are non-zero, then one sample is 2 x 3 = 6 bytes.

The slots should be enabled in order (i.e., SLOT1 should not be disabled if SLOT2 is enabled).

Temperature Data (0x1F–0x21)

REGISTER	B7	B6	B5	B4	B3	B2	B1	B0	REG ADDR	POR STATE	R/W
Die Temp Integer	TINT[7]								0x1F	0x00	R
Die Temp Fraction					TFRAC[3:0]				0x20	0x00	R
Die Temperature Config								TEMP_EN	0x21	0x00	R/W

Temperature Integer

The on-board temperature ADC output is split into two registers, one to store the integer temperature and one to store the fraction. Both should be read when reading the temperature data, and the equation below shows how to add the two registers together:

$$T_{MEASURED} = T_{INTEGER} + T_{FRACTION}$$

This register stores the integer temperature data in 2's complement format, where each bit corresponds to 1°C.

Table 10. Temperature Integer

REGISTER VALUE (hex)	TEMPERATURE (°C)
0x00	0
0x01	+1
...	...
0x7E	+126
0x7F	+127
0x80	-128
0x81	-127
...	...
0xFE	-2
0xFF	-1

Temperature Fraction

This register stores the fractional temperature data in increments of 0.0625°C. If this fractional temperature is paired with a negative integer, it still adds as a positive fractional value (e.g., -128°C + 0.5°C = -127.5°C).

Temperature Enable (TEMP_EN)

This is a self-clearing bit which, when set, initiates a single temperature reading from the temperature sensor. This bit clears automatically back to zero at the conclusion of the temperature reading when the bit is set to one.

Applications Information

Sample Rate and Performance

The maximum sample rate for the ADC depends on the selected pulse width, which in turn, determines the ADC resolution. For instance, if the pulse width is set to 69µs then the ADC resolution is 15 bits, and all sample rates are selectable. However, if the pulse width is set to 411µs, then the samples rates are limited. The allowed sample rates for both SpO_2 and HR Modes are summarized in the Table 11 and Table 12.

Power Considerations

The LED waveforms and their implication for power supply design are discussed in this section.

The LEDs in the MAX30102 are pulsed with a low duty cycle for power savings, and the pulsed currents can cause ripples in the V_{LED+} power supply. To ensure these pulses do not translate into optical noise at the LED outputs, the power supply must be designed to handle these. Ensure that the resistance and inductance from the power supply (battery, DC/DC converter, or LDO) to the pin is much smaller than 1Ω, and that there is at least 1µF of power supply bypass capacitance to a good ground plane. The capacitance should be located as close as physically possible to the IC.

Table 11. SpO_2 Mode (Allowed Settings)

SAMPLES PER SECOND	PULSE WIDTH (µs)			
	69	118	215	411
50	O	O	O	O
100	O	O	O	O
200	O	O	O	O
400	O	O	O	O
800	O	O	O	
1000	O	O		
1600	O			
3200				
Resolution (bits)	15	16	17	18

Table 12. HR Mode (Allowed Settings)

SAMPLES PER SECOND	PULSE WIDTH (µs)			
	69	118	215	411
50	O	O	O	O
100	O	O	O	O
200	O	O	O	O
400	O	O	O	O
800	O	O	O	O
1000	O	O	O	O
1600	O	O	O	
3200	O			
Resolution (bits)	15	16	17	18

In the Heart Rate mode, only the Red LED is used to capture optical data and determine the user's heart rate and/or photoplethysmogram (PPG).

SpO_2 Temperature Compensation

The MAX30102 has an accurate on-board temperature sensor that digitizes the IC's internal temperature upon command from the I^2C master. The temperature has an effect on the wavelength of the red and IR LEDs. While the device output data is relatively insensitive to the wavelength of the IR LED, the red LED's wavelength is critical to correct interpretation of the data.

Table 13 shows the correlation of red LED wavelength versus the temperature of the LED. Since the LED die heats up with a very short thermal time constant (tens of microseconds), the LED wavelength should be calculated according to the current level of the LED and the temperature of the IC. Use Table 13 to estimate the temperature.

Red LED Current Settings vs. LED Temperature Rise

Add the temperature rise to the module temperature reading to estimate the LED temperature and output wavelength. The LED temperature estimate is valid even with very short pulse widths, due to the fast thermal time constant of the LED.

Interrupt Pin Functionality

The active-low interrupt pin pulls low when an interrupt is triggered. The pin is open-drain, which means it normally requires a pullup resistor or current source to an external voltage supply (up to +5V from GND). The interrupt pin is not designed to sink large currents, so the pullup resistor value should be large, such as 4.7kΩ.

Table 13. RED LED Current Settings vs. LED Temperature Rise

RED LED CURRENT SETTING	RED LED DUTY CYCLE (% OF LED PULSE WIDTH TO SAMPLE TIME)	ESTIMATED TEMPERATURE RISE (ADD TO TEMP SENSOR MEASUREMENT) (°C)
00000001 (0.2mA)	8	0.1
11111010 (50mA)	8	2
00000001 (0.2mA)	16	0.3
11111010 (50mA)	16	4
00000001 (0.2mA)	32	0.6
11111010 (50mA)	32	8

Timing for Measurements and Data Collection

Slot Timing in Multi-LED Modes

The MAX30102 can support two LED channels of sequential processing (Red and IR). Table 14 below displays the four possible channel slot times associated with each pulse width setting. Figure 3 shows an example of channel slot timing for a SpO_2 mode application with a 1kHz sample rate.

Table 14. Slot Timing

PULSE-WIDTH SETTING (µs)	CHANNEL SLOT TIMING (TIMING PERIOD BETWEEN PULSES) (µs)	CHANNEL-CHANNEL TIMING (RISING EDGE-TO-RISING EDGE) (µs)
69	358	427
118	407	525
215	505	720
411	696	1107

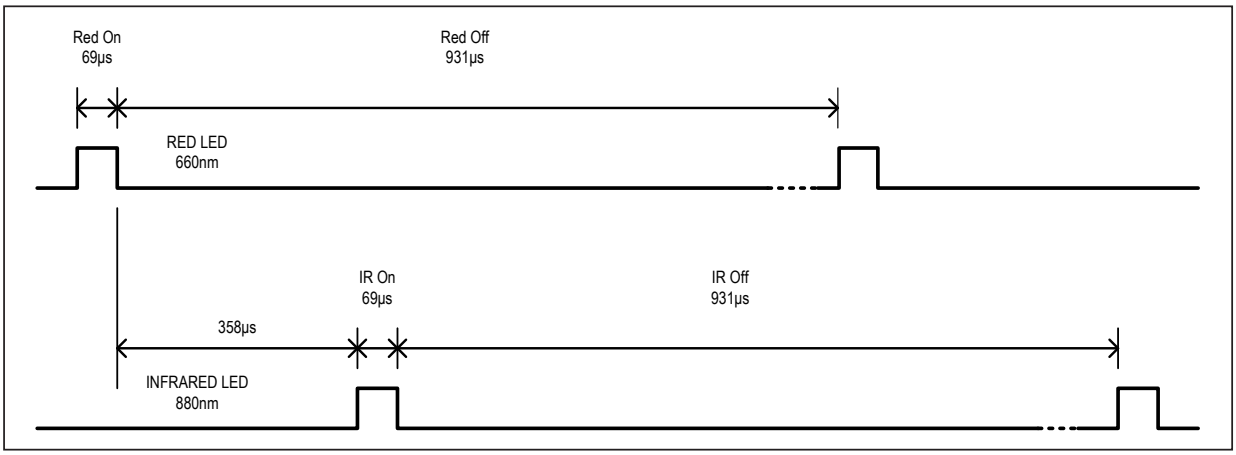


Figure 3. Channel Slot Timing for the SpO_2 Mode with a 1kHz Sample Rate

Timing in SpO_2 Mode

The internal FIFO stores up to 32 samples, so that the system processor does not need to read the data after every sample. The temperature does not need to be sampled very often–once a second or every few seconds should be sufficient.

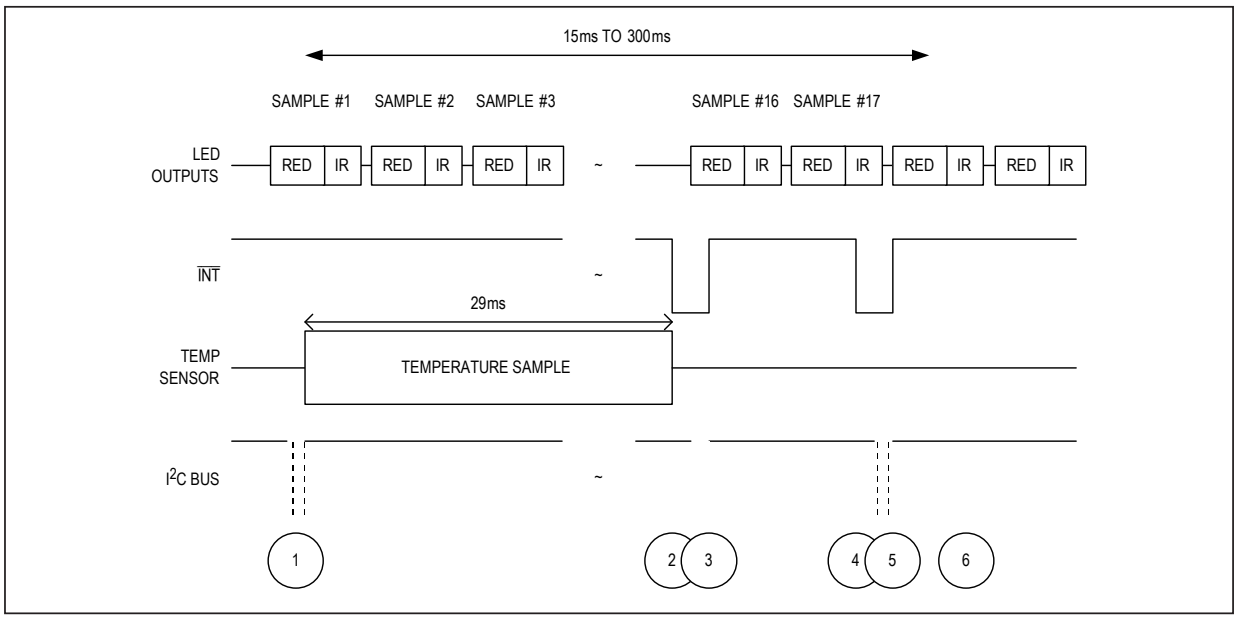


Figure 4. Timing for Data Acquisition and Communication When in SpO_2 Mode

Table 15. Events Sequence for Figure 4 in SpO_2 Mode

EVENT	DESCRIPTION	COMMENTS
1	Enter into SpO_2 Mode. Initiate a Temperature measurement.	I^2C Write Command sets MODE[2:0] = 0x03 and A_FULL_EN. Then to enable and initiate a single temperature measurement, set TEMP_EN and DIE_TEMP_RDY_EN.
2	Temperature Measurement Complete, Interrupt Generated	DIE_TEMP_RDY interrupt triggers, alerting the central processor to read the data.
3	Temp Data is Read, Interrupt Cleared	
4	FIFO is Almost Full, Interrupt Generated	Interrupt is generated when the FIFO almost full threshold is reached.
5	FIFO Data is Read, Interrupt Cleared	
6	Next Sample is Stored	New Sample is stored at the new read pointer location. Effectively, it is now the first sample in the FIFO.

Timing in HR Mode

The internal FIFO stores up to 32 samples, so that the system processor does not need to read the data after every sample. In HR mode (Figure 5), unlike in SpO_2 mode, temperature information is not necessary to interpret the data. The user can select either the red LED or the infrared LED channel for heart rate measurements.

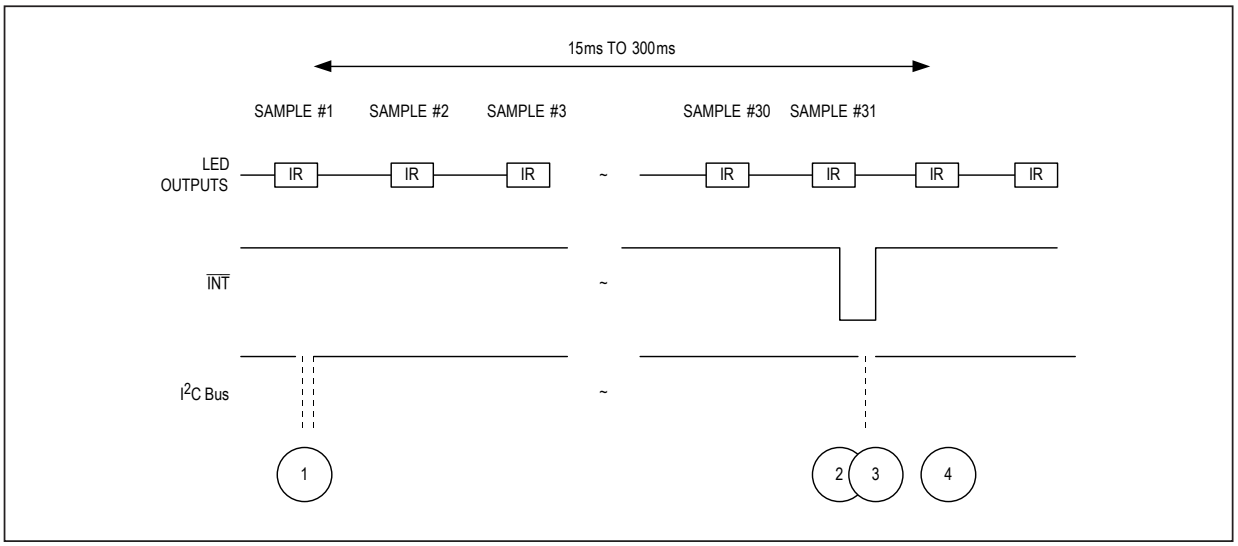


Figure 5. Timing for Data Acquisition and Communication When in HR Mode

Table 16. Events Sequence for Figure 5 in HR Mode

EVENT	DESCRIPTION	COMMENTS
1	Enter into Mode	I²C Write Command sets MODE[2:0] = 0x02. Mask the A_FULL_EN Interrupt.
2	FIFO is Almost Full, Interrupt Generated	Interrupt is generated when the FIFO has only one empty space left.
3	FIFO Data is Read, Interrupt Cleared	
4	Next Sample is Stored	New sample is stored at the new read pointer location. Effectively, it is now the first sample in the FIFO.

Power Sequencing and Requirements

Power-Up Sequencing

Figure 6. shows the recommended power-up sequence for the MAX30102.

It is recommended to power the V_{DD} supply first, before the LED power supplies (V_{LED+}). The interrupt and I²C pins can be pulled up to an external voltage even when the power supplies are not powered up.

After the power is established, an interrupt occurs to alert the system that the MAX30102 is ready for operation. Reading the I²C interrupt register clears the interrupt, as shown in Figure 6.

Power-Down Sequencing

The MAX30102 is designed to be tolerant of any power supply sequencing on power-down.

I²C Interface

The MAX30102 features an I²C/SMBus-compatible, 2-wire serial interface consisting of a serial data line (SDA) and a serial clock line (SCL). SDA and SCL facilitate communication between the MAX30102 and the master at clock rates up to 400kHz. Figure 1 shows the 2-wire interface timing diagram. The master generates SCL and initiates data transfer on the bus. The master device writes data to the MAX30102 by transmitting the proper slave address followed by data. Each transmit sequence is framed by a START (S) or REPEATED START (Sr) condition and a STOP (P) condition. Each word transmitted to the MAX30102 is 8 bits long and is followed by an acknowledge clock pulse. A master reading data from the MAX30102 transmits the proper slave address followed by a series of nine SCL pulses.

The MAX30102 transmits data on SDA in sync with the master-generated SCL pulses. The master acknowledges receipt of each byte of data. Each read sequence is framed by a START (S) or REPEATED START (Sr) condition, a not acknowledge, and a STOP (P) condition. SDA operates as both an input and an open-drain output. A pullup resistor, typically greater than 500Ω, is required on SDA. SCL operates only as an input. A pullup resistor, typically greater than 500Ω, is required on SCL if there are multiple masters on the bus, or if the single master has an open-drain SCL output. Series resistors in line with SDA and SCL are optional. Series resistors protect the digital inputs of the MAX30102 from high voltage spikes on the bus lines and minimize crosstalk and undershoot of the bus signals.

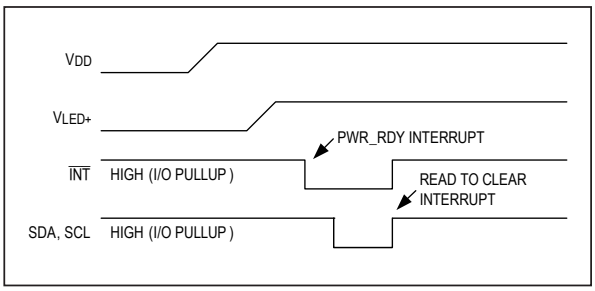


Figure 6. Power-Up Sequence of the Power Supply Rails

Bit Transfer

One data bit is transferred during each SCL cycle. The data on SDA must remain stable during the high period of the SCL pulse. Changes in SDA while SCL is high are control signals. See the *START and STOP Conditions* section.

START and STOP Conditions

SDA and SCL idle high when the bus is not in use. A master initiates communication by issuing a START condition. A START condition is a high-to-low transition on SDA with SCL high. A STOP condition is a low-to-high transition on SDA while SCL is high (Figure 7). A START condition from the master signals the beginning of a transmission to the device. The master terminates transmission, and frees the bus, by issuing a STOP condition. The bus remains active if a REPEATED START condition is generated instead of a STOP condition.

Early STOP Conditions

The MAX30102 recognizes a STOP condition at any point during data transmission except if the STOP condition occurs in the same high pulse as a START condition. For proper operation, do not send a STOP condition during the same SCL high pulse as the START condition.

Slave Address

A bus master initiates communication with a slave device by issuing a START condition followed by the 7-bit slave ID. When idle, the MAX30102 waits for a START condition followed by its slave ID. The serial interface compares each slave ID bit by bit, allowing the interface to power down and disconnect from SCL immediately if an incorrect slave ID is detected. After recognizing a START condition followed by the correct slave ID, the MAX30102 is programmed to accept or send data. The LSB of the slave ID word is the read/write (R/$\overline{W}$) bit. R/W indicates whether the master is writing to or reading data from the MAX30102 (R/$\overline{W}$ = 0 selects a write condition, R/$\overline{W}$ = 1 selects a read condition).

After receiving the proper slave ID, the MAX30102 issues an ACK by pulling SDA low for one clock cycle.

The MAX30102 slave ID consists of seven fixed bits, B7–B1 (set to 0b1010111). The most significant slave ID bit (B7) is transmitted first, followed by the remaining bits. Table 17 shows the possible slave IDs of the device.

Acknowledge

The acknowledge bit (ACK) is a clocked 9th bit that the MAX30102 uses to handshake receipt each byte of data when in write mode (Figure 8). The MAX30102 pulls down SDA during the entire master-generated 9th clock pulse if the previous byte is successfully received. Monitoring ACK allows for detection of unsuccessful data transfers. An unsuccessful data transfer occurs if a receiving device is busy or if a system fault has occurred. In the event of an unsuccessful data transfer, the bus master retries communication. The master pulls down SDA during the 9th clock cycle to acknowledge receipt of data when the MAX30102 is in read mode. An acknowledge is sent by the master after each read byte to allow data transfer to continue. A not-acknowledge is sent when the master reads the final byte of data from the MAX30102, followed by a STOP condition.

Write Data Format

For the write operation, send the slave ID as the first byte followed by the register address byte and then one or more data bytes. The register address pointer increments automatically after each byte of data received, so for example the entire register bank can be written by at one time. Terminate the data transfer with a STOP condition. The write operation is shown in Figure 9.

The internal register address pointer increments automatically, so writing additional data bytes fill the data registers in order.

Table 17. Slave ID Description

B7	B6	B5	B4	B3	B2	B1	B0	WRITE ADDRESS	READ ADDRESS
1	0	1	0	1	1	1	R/$\overline{W}$	0xAE	0xAF

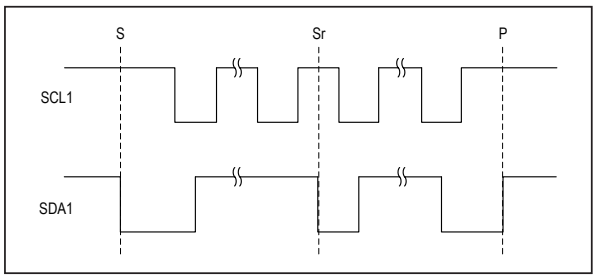


Figure 7. START, STOP, and REPEATED START Conditions

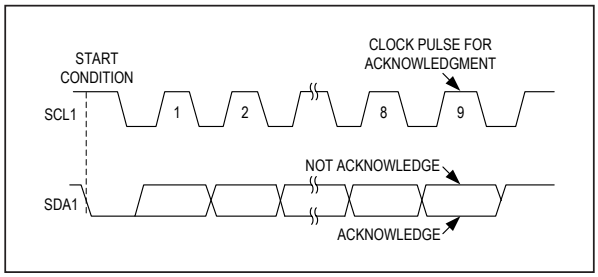


Figure 8. Acknowledge

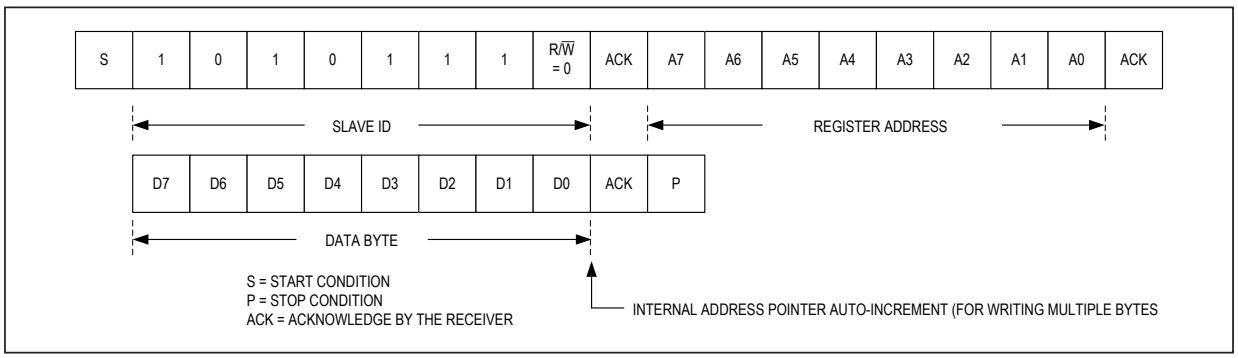


Figure 9. Writing One Data Byte to the MAX30102

Read Data Format

For the read operation, two I2C operations must be performed. First, the slave ID byte is sent followed by the I2C register that you wish to read. Then a REPEAT START (Sr) condition is sent, followed by the read slave ID. The MAX30102 then begins sending data beginning with the register selected in the first operation. The read pointer increments automatically, so the device continues sending data from additional registers in sequential order until a STOP (P) condition is received. The exception to this is the FIFO_DATA register, at which the read pointer no longer increments when reading additional bytes. To read the next register after FIFO_DATA, an I2C write command is necessary to change the location of the read pointer.

Figure 10 and Figure 11 show the process of reading one byte and multiple bytes of data.

An initial write operation is required to send the read register address.

Data is sent from registers in sequential order, starting from the register selected in the initial I2C write operation. If the FIFO_DATA register is read, the read pointer will not automatically increment, and subsequent bytes of data will contain the contents of the FIFO.

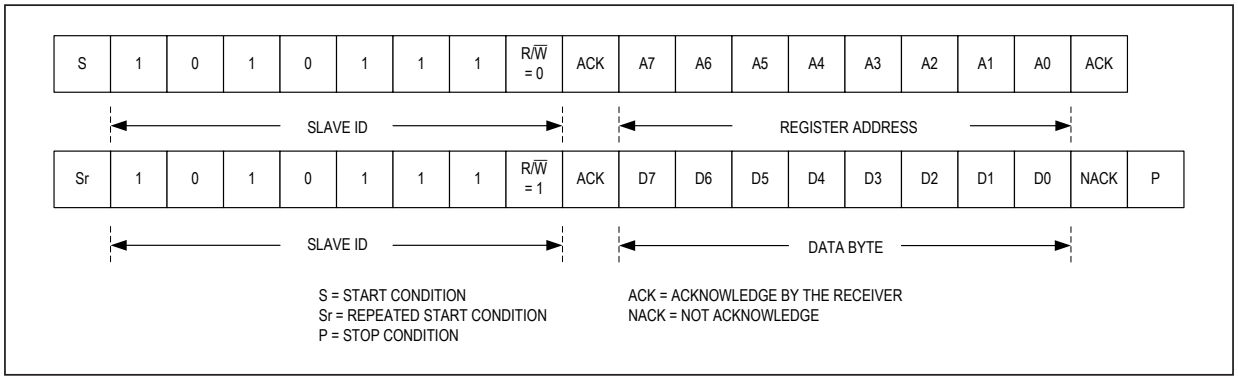


Figure 10. Reading One Byte of Data from MAX30102

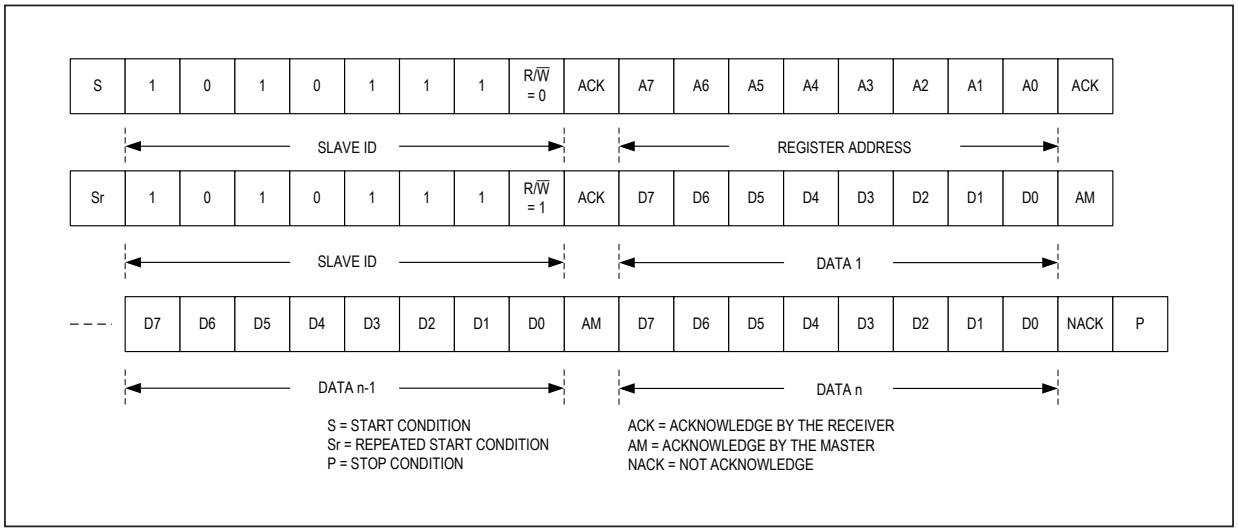
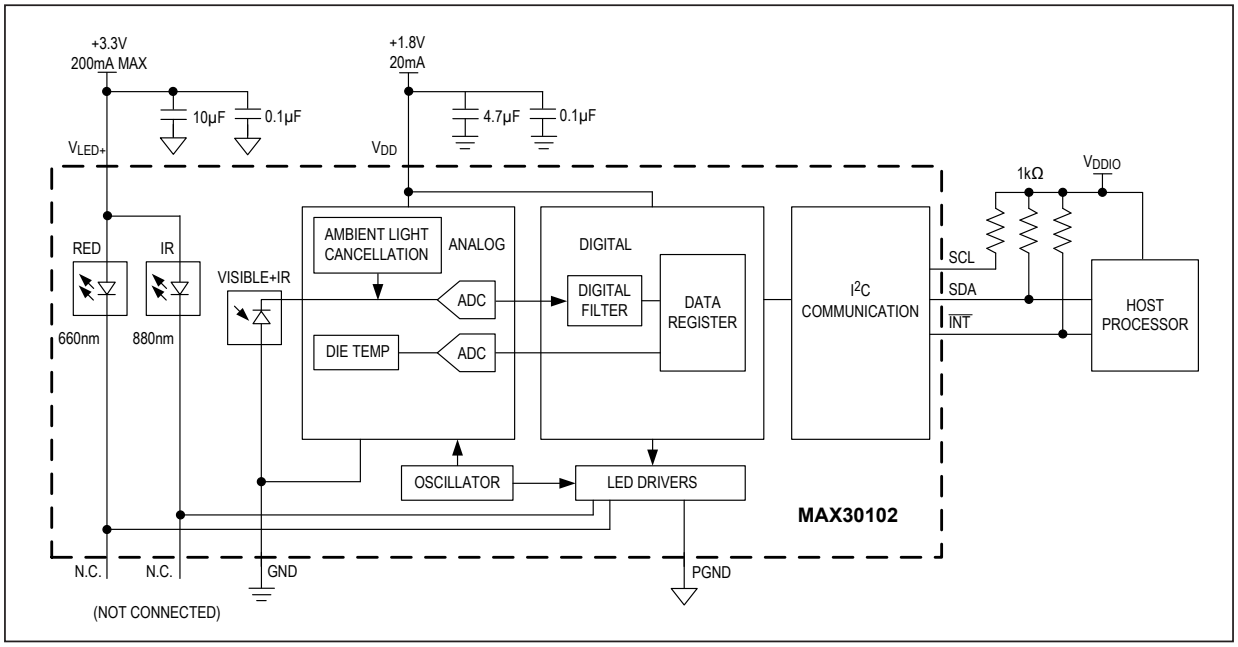


Figure 11. Reading Multiple Bytes of Data from the MAX30102

Typical Application Circuit



Ordering Information

PART	TEMP RANGE	PIN-PACKAGE
MAX30102EFD+T	-40°C to +85°C	14-Lead OESIP (0.8mm Pin Pitch)

+Denotes lead(Pb)-free/RoHS-compliant package.

T = Tape and reel.

Revision History

REVISION NUMBER	REVISION DATE	DESCRIPTION	PAGES CHANGED
0	9/15	Initial release	—
1	10/18	Updated the *General Description, Applications, Absolute Maximum Ratings, Electrical Characteristics, Pin Description, Timing in SpO_2 Mode, Power-Up Sequencing* sections; updated the *System Diagram, Pin Configuration, and Functional Diagram*; updated the *Register Map*, Interrupt Status (0x00–0x01), Interrupt Enable (0x02–0x03), FIFO (0x04–0x07), LED Pulse Amplitude (0x0C–0x0D), Table 8, Multi-LED Mode Control Registers (0x11–0x12), Table 9, Temperature Data (0x1F–0x21), Table 13, Table 15, Table 16; replaced the *Typical Application Circuit*; removed the *Proximity Function* section and the Proximity Mode Interrupt Threshold (0x30) register	1–5, 8–14, 18 20–24, 26–28, 31

NCP612, NCV612

Voltage Regulator - CMOS, Low Iq, SC70-5

100 mA

The NCP612/NCV612 series of fixed output linear regulators are designed for handheld communication equipment and portable battery powered applications which require low quiescent. The NCP612/NCV612 series features an ultra−low quiescent current of 40 μA. Each device contains a voltage reference unit, an error amplifier, a PMOS power transistor, resistors for setting output voltage, current limit, and temperature limit protection circuits.

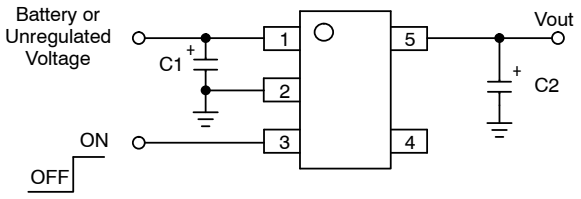
The NCP612/NCV612 has been designed to be used with low cost ceramic capacitors. The device is housed in the micro−miniature SC70−5 surface mount package. Standard voltage versions are 1.5, 1.8, 2.5, 2.7, 2.8, 3.0, 3.1, 3.3, 3.7, and 5.0 V.

Features

- Low Quiescent Current of 40 μA Typical
- Low Dropout Voltage of 230 mV at 100 mA and 3.0 V V_{out}
- Low Output Voltage Option
- Output Voltage Accuracy of 2.0%
- Temperature Range of −40°C to 85°C (NCP612)
  Temperature Range of −40°C to 125°C (NCV612)
- NCV Prefix for Automotive and Other Applications Requiring Unique Site and Control Change Requirements; AEC−Q100 Qualified and PPAP Capable
- These are Pb−Free Devices

Typical Applications

- Cellular Phones
- Battery Powered Consumer Products
- Hand−Held Instruments
- Camcorders and Cameras



This device contains 86 active transistors

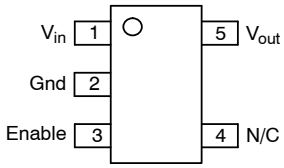
Figure 1. Typical Application Diagram

ON Semiconductor®

www.onsemi.com

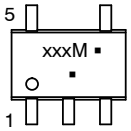
SC70−5
CASE 419A

PIN CONNECTIONS



(Top View)

MARKING DIAGRAM



xxx = Specific Device Code
M = Date Code*
▪ = Pb−Free Package

(Note: Microdot may be in either location)

ORDERING INFORMATION

See detailed ordering and shipping information in the package dimensions section on page 8 of this data sheet.

PIN FUNCTION DESCRIPTION

Pin No.	Pin Name	Description
1	Vin	Positive power supply input voltage.
2	Gnd	Power supply ground.
3	Enable	This input is used to place the device into low−power standby. When this input is pulled low, the device is disabled. If this function is not used, Enable should be connected to Vin.
4	N/C	No internal connection.
5	Vout	Regulated output voltage.

MAXIMUM RATINGS

Rating	Symbol	Value	Unit
Input Voltage	V_{in}	0 to 6.0	V
Enable Voltage	Enable	−0.3 to V_{in} +0.3	V
Output Voltage	V_{out}	−0.3 to V_{in} +0.3	V
Power Dissipation and Thermal Characteristics Power Dissipation Thermal Resistance, Junction−to−Ambient	 P_D $R_{\theta JA}$	 Internally Limited 300	 W °C/W
Operating Junction Temperature	T_J	+150	°C
Operating Ambient Temperature	T_A	−40 to +125	°C
Storage Temperature	T_{stg}	−55 to +150	°C

Stresses exceeding those listed in the Maximum Ratings table may damage the device. If any of these limits are exceeded, device functionality should not be assumed, damage may occur and reliability may be affected.

1. This device series contains ESD protection and exceeds the following tests:
   Human Body Model 2000 V per MIL−STD−883, Method 3015
   Machine Model Method 200 V
2. Latch−up capability (85°C) ±200 mA DC with trigger voltage.

ELECTRICAL CHARACTERISTICS

(V_{in} = $V_{out(nom.)}$ + 1.0 V, V_{enable} = V_{in}, C_{in} = 1.0 μF, C_{out} = 1.0 μF, T_J = 25°C, unless otherwise noted.)

Characteristic	Symbol	Min	Typ	Max	Unit
Output Voltage (T_A = 25°C, I_{out} = 10 mA)	V_{out}				V
1.5 V		1.455	1.5	1.545	
1.8 V		1.746	1.8	1.854	
2.5 V		2.425	2.5	2.575	
2.7 V		2.646	2.7	2.754	
2.8 V		2.744	2.8	2.856	
3.0 V		2.940	3.0	3.060	
3.1 V		3.038	3.1	3.162	
3.3 V		3.234	3.3	3.366	
3.7 V		3.626	3.7	3.774	
5.0 V		4.900	5.0	5.100	
Output Voltage (T_A = −40°C to 85°C, I_{out} = 10 mA)	V_{out}				V
1.5 V		1.455	1.5	1.545	
1.8 V		1.746	1.8	1.854	
2.5 V		2.425	2.5	2.575	
2.7 V		2.619	2.7	2.781	
2.8 V		2.716	2.8	2.884	
3.0 V		2.910	3.0	3.090	
3.1 V		3.007	3.1	3.193	
3.3 V		3.201	3.3	3.399	
3.7 V		3.626	3.7	3.774	
5.0 V		4.900	5.0	5.100	

ELECTRICAL CHARACTERISTICS (continued)

(V_{in} = $V_{out(nom.)}$ + 1.0 V, V_{enable} = V_{in}, C_{in} = 1.0 μF, C_{out} = 1.0 μF, T_J = 25°C, unless otherwise noted.)

Characteristic	Symbol	Min	Typ	Max	Unit
Output Voltage (T_A = −40°C to 125°C, I_{out} = 10 mA) NCV612 Only	V_{out}				V
1.5 V		1.440	1.5	1.560	
1.8 V		1.728	1.8	1.872	
2.5 V		2.400	2.5	2.600	
2.7 V		2.592	2.7	2.808	
2.8 V		2.688	2.8	2.912	
3.0 V		2.880	3.0	3.120	
3.1 V		2.976	3.1	3.224	
3.3 V		3.201	3.3	3.399	
5.0 V		4.850	5.0	5.150	
Output Voltage (T_A = −40°C to 85°C, I_{out} = 100 mA)	V_{out}				V
1.5 V		1.440	1.5	1.560	
1.8 V		1.728	1.8	1.872	
2.5 V		2.400	2.5	2.600	
2.7 V		2.592	2.7	2.808	
2.8 V		2.688	2.8	2.912	
3.0 V		2.880	3.0	3.120	
3.1 V		2.976	3.1	3.224	
3.3 V		3.201	3.3	3.399	
3.7 V		3.589	3.7	3.811	
5.0 V		4.850	5.0	5.150	
Line Regulation (I_{out} = 10 mA)	Reg_{line}				mV/V
1.5 V–4.4 V (V_{in} = $V_{out(nom.)}$ + 1.0 V to 6.0 V)		–	1.0	3.0	
4.5 V–5.0 V (V_{in} = 5.5 V to 6.0 V)		–	1.0	3.0	
Load Regulation (I_{out} = 1.0 mA to 100 mA)	Reg_{load}	–	0.3	0.8	mV/mA
Output Current (V_{out} = (V_{out} at I_{out} = 100 mA) −3%)	$I_{o(nom.)}$				mA
1.5 V–3.9 V (V_{in} = $V_{out(nom.)}$ + 2.0 V)		100	200	–	
4.0 V–5.0 V (V_{in} = 6.0 V)		100	200	–	
Dropout Voltage (T_A = −40°C to 85°C, I_{out} = 100 mA, Measured at $V_{out(nom)}$ −3.0%)	V_{in}–V_{out}				mV
1.5 V		–	530	680	
1.8 V		–	420	560	
2.5 V		–	270	380	
2.7 V		–	270	380	
2.8 V		–	250	380	
3.0 V		–	230	380	
3.1 V		–	210	380	
3.3 V		–	200	380	
3.7 V		–	180	380	
5.0 V		–	160	300	
Ground Current (Enable Input = V_{in}, I_{out} = 1.0 mA to $I_{o(nom.)}$)	I_{GND}	–	40	90	μA
Quiescent Current (T_A = −40°C to 85°C)	I_Q				μA
(Enable Input = 0 V)		–	0.03	1.0	
(Enable Input = V_{in}, I_{out} = 1.0 mA to $I_{o(nom.)}$)		–	40	90	
Output Short Circuit Current (V_{out} = 0 V)	$I_{out(max)}$				mA
1.5 V–3.9 V (V_{in} = $V_{out(nom.)}$ + 2.0 V)		150	300	600	
4.0 V–5.0 V (V_{in} = 6.0 V)		150	300	600	
Output Voltage Noise (f = 100 Hz to 100 kHz) I_{out} = 30 mA, C_{out} = 1 μF	V_n	–	100	–	μVrms
Enable Input Threshold Voltage	$V_{th(en)}$				V
(Voltage Increasing, Output Turns On, Logic High)		0.95	–	–	
(Voltage Decreasing, Output Turns Off, Logic Low)		–	–	0.3	
Output Voltage Temperature Coefficient	T_C	–	±100	–	ppm/°C

3. Maximum package power dissipation limits must be observed.

$$PD = \frac{T_{J(max)} - T_A}{R_{\theta JA}}$$

4. Low duty cycle pulse techniques are used during testing to maintain the junction temperature as close to ambient as possible.

TYPICAL CHARACTERISTICS

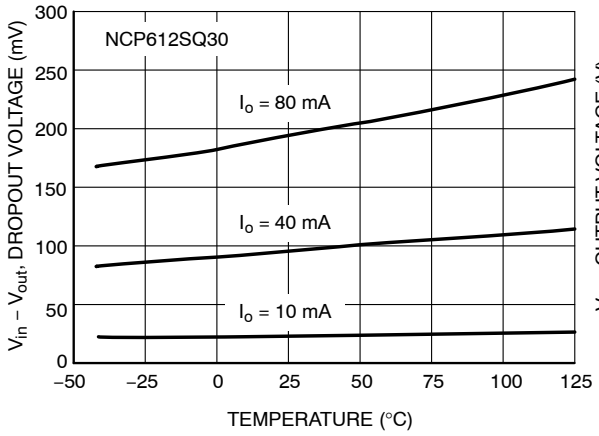


Figure 2. Dropout Voltage vs. Temperature

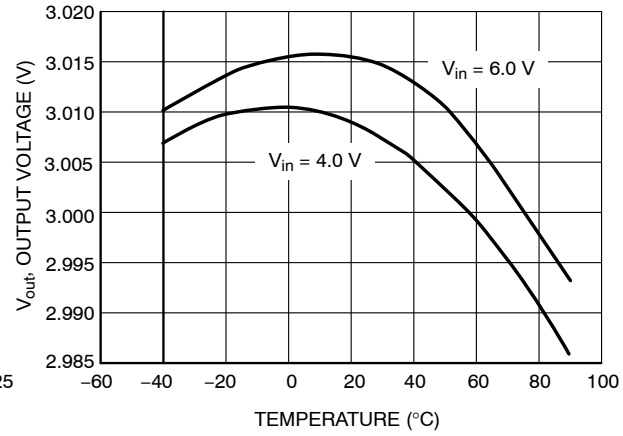


Figure 3. Output Voltage vs. Temperature

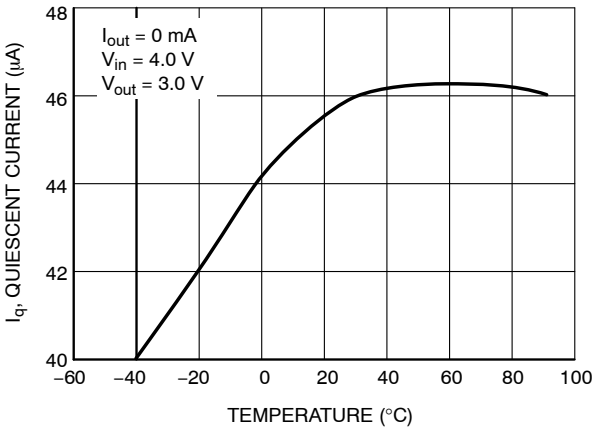


Figure 4. Quiescent Current vs. Temperature

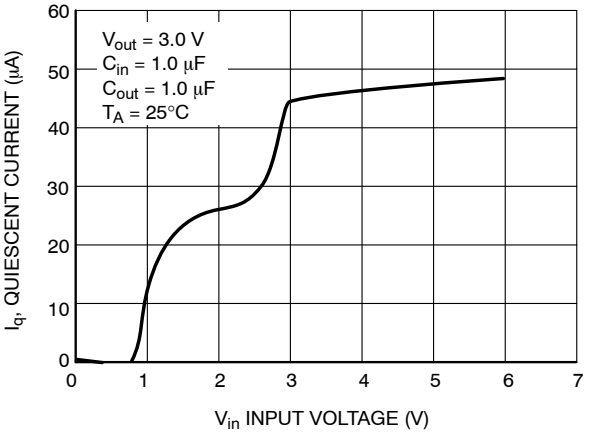


Figure 5. Quiescent Current vs. Input Voltage

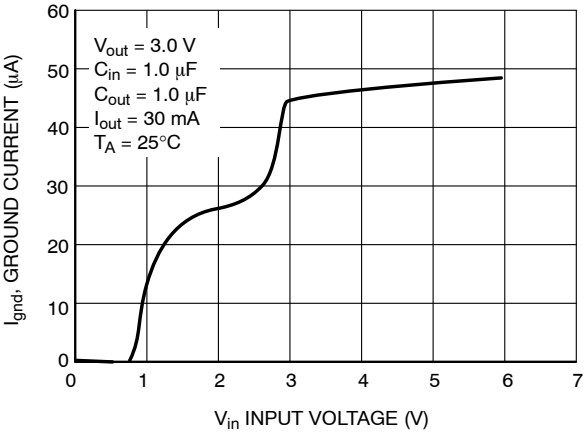


Figure 6. Ground Pin Current vs. Input Voltage

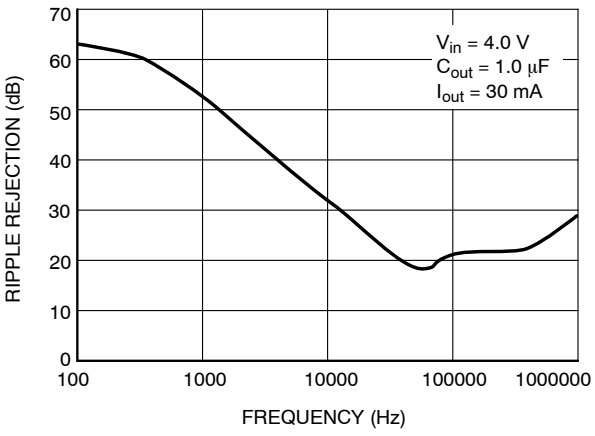


Figure 7. Ripple Rejection vs. Frequency

TYPICAL CHARACTERISTICS

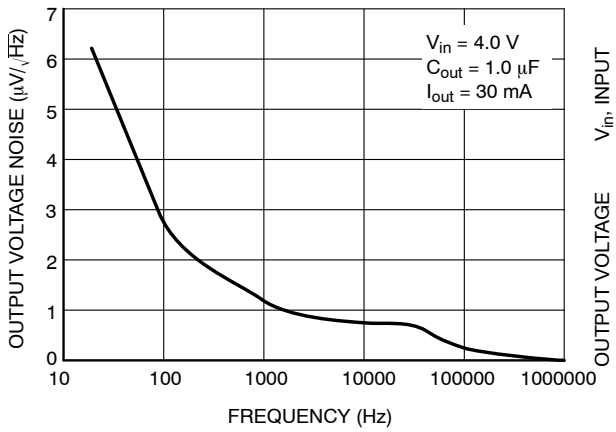


Figure 8. Output Noise Density

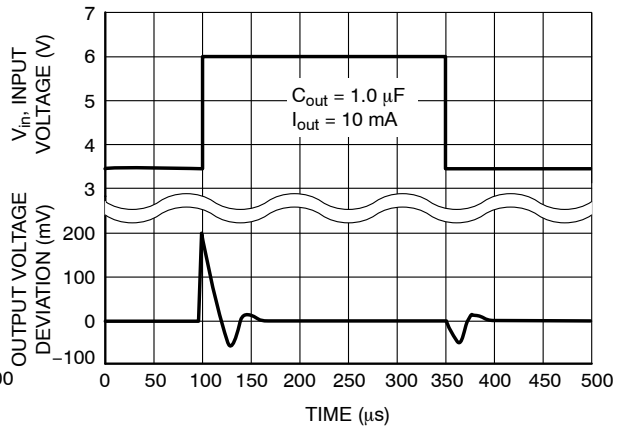


Figure 9. Line Transient Response

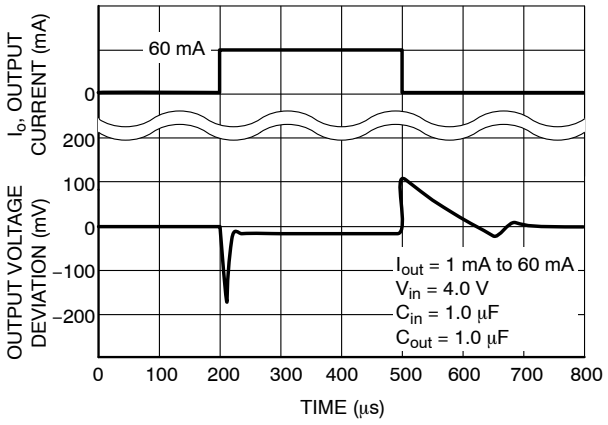


Figure 10. Load Transient Response

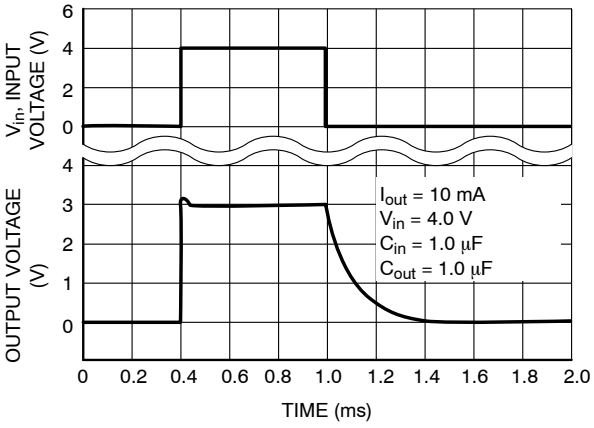


Figure 11. Turn−on Response

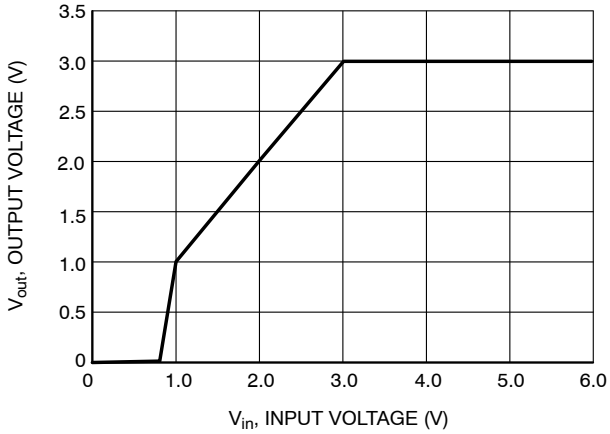


Figure 12. Output Voltage vs. Input Voltage

DEFINITIONS

Load Regulation

The change in output voltage for a change in output current at a constant temperature.

Dropout Voltage

The input/output differential at which the regulator output no longer maintains regulation against further reductions in input voltage. Measured when the output drops 3.0% below its nominal. The junction temperature, load current, and minimum input supply requirements affect the dropout level.

Maximum Power Dissipation

The maximum total dissipation for which the regulator will operate within its specifications.

Quiescent Current

The quiescent current is the current which flows through the ground when the LDO operates without a load on its output: internal IC operation, bias, etc. When the LDO becomes loaded, this term is called the Ground current. It is actually the difference between the input current (measured through the LDO input pin) and the output current.

Line Regulation

The change in output voltage for a change in input voltage. The measurement is made under conditions of low dissipation or by using pulse technique such that the average chip temperature is not significantly affected.

Line Transient Response

Typical over and undershoot response when input voltage is excited with a given slope.

Thermal Protection

Internal thermal shutdown circuitry is provided to protect the integrated circuit in the event that the maximum junction temperature is exceeded. When activated at typically 160°C, the regulator turns off. This feature is provided to prevent failures from accidental overheating.

Maximum Package Power Dissipation

The maximum power package dissipation is the power dissipation level at which the junction temperature reaches its maximum operating value, i.e. 150°C. Depending on the ambient power dissipation and thus the maximum available output current.

APPLICATIONS INFORMATION

A typical application circuit for the NCP612/NCV612 is shown in Figure 1, front page.

Input Decoupling (C1)

A 1.0 μF capacitor either ceramic or tantalum is recommended and should be connected close to the NCP612/NCV612 package. Higher values and lower ESR will improve the overall line transient response.
TDK capacitor: C2012X5R1C105K, or C1608X5R1A105K

Output Decoupling (C2)

The NCP612/NCV612 is a stable regulator and does not require any specific Equivalent Series Resistance (ESR) or a minimum output current. Capacitors exhibiting ESRs ranging from a few mΩ up to 5.0 Ω can thus safely be used. The minimum decoupling value is 1.0 μF and can be augmented to fulfill stringent load transient requirements. The regulator accepts ceramic chip capacitors as well as tantalum capacitors. Larger values improve noise rejection and load regulation transient response.
TDK capacitor: C2012X5R1C105K, C1608X5R1A105K, or C3216X7R1C105K

Enable Operation

The enable pin will turn on the regulator when pulled high and turn off the regulator when pulled low. These limits of threshold are covered in the electrical specification section of this data sheet. If the enable is not used then the pin should be connected to V_{in}.

Hints

Please be sure the Vin and Gnd lines are sufficiently wide. When the impedance of these lines is high, there is a chance to pick up noise or cause the regulator to malfunction.

Set external components, especially the output capacitor, as close as possible to the circuit, and make leads as short as possible.

Thermal

As power across the NCP612/NCV612 increases, it might become necessary to provide some thermal relief. The maximum power dissipation supported by the device is dependent upon board design and layout. Mounting pad configuration on the PCB, the board material and also the ambient temperature effect the rate of temperature rise for the part. This is stating that when the NCP612/NCV612 has good thermal conductivity through the PCB, the junction temperature will be relatively low with high power dissipation applications.

The maximum dissipation the package can handle is given by:

$$PD = \frac{T_{J(max)} - T_A}{R_{\theta JA}}$$

If junction temperature is not allowed above the maximum 125°C, then the NCP612/NCV612 can dissipate up to 330 mW @ 25°C.

The power dissipated by the NCP612/NCV612 can be calculated from the following equation:

$$P_{tot} = \left[V_{in} * I_{gnd}\,(I_{out})\right] + [V_{in} - V_{out}] * I_{out}$$

or

$$V_{inMAX} = \frac{P_{tot} + V_{out} * I_{out}}{I_{gnd} + I_{out}}$$

If an 100 mA output current is needed then the ground current from the data sheet is 40 μA. For an NCP612/NCV612 (3.0 V), the maximum input voltage will then be 6.0 V (Limited by maximum input voltage).

ORDERING INFORMATION

Device	Nominal Output Voltage	Marking	Package	Shipping†
NCP612SQ15T2G	1.5	LHO	SC70−5 (Pb−Free)	3000 Units/Tape & Reel
NCP612SQ18T2G	1.8	LHP		
NCP612SQ25T2G	2.5	LHQ		
NCP612SQ27T2G	2.7	LHR		
NCP612SQ28T2G	2.8	LHS		
NCP612SQ30T2G	3.0	LHT		
NCP612SQ31T2G	3.1	LHU		
NCP612SQ33T2G	3.3	LHV		
NCP612SQ37T2G	3.7	LKH		
NCP612SQ50T2G	5.0	LHW		
NCV612SQ15T2G*	1.5	LHO		
NCV612SQ18T2G*	1.8	LHP		
NCV612SQ25T2G*	2.5	LHQ		
NCV612SQ27T2G*	2.7	LHR		
NCV612SQ28T2G*	2.8	LHS		
NCV612SQ30T2G*	3.0	LHT		
NCV612SQ31T2G*	3.1	LHU		
NCV612SQ33T2G*	3.3	LHV		
NCV612SQ37T2G*	3.7	LKH		
NCV612SQ50T2G*	5.0	LHW		

†For information on tape and reel specifications, including part orientation and tape sizes, please refer to our Tape and Reel Packaging Specification Brochure, BRD8011/D.

*NCV Prefix for Automotive and Other Applications Requiring Unique Site and Control Change Requirements; AEC−Q100 Qualified and PPAP Capable.

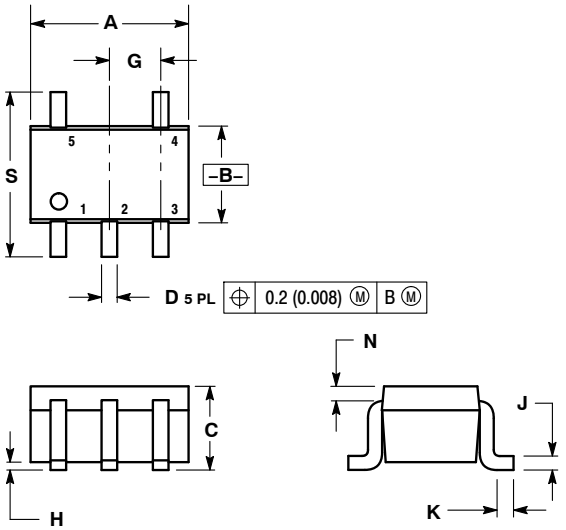
MECHANICAL CASE OUTLINE
PACKAGE DIMENSIONS

ON Semiconductor®

SC−88A (SC−70−5/SOT−353)
CASE 419A−02
ISSUE L

DATE 17 JAN 2013

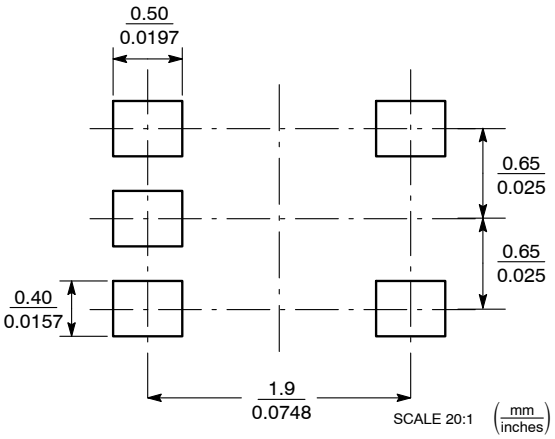
SCALE 2:1



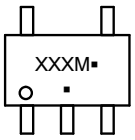
NOTES:
1. DIMENSIONING AND TOLERANCING PER ANSI Y14.5M, 1982.
2. CONTROLLING DIMENSION: INCH.
3. 419A−01 OBSOLETE. NEW STANDARD 419A−02.
4. DIMENSIONS A AND B DO NOT INCLUDE MOLD FLASH, PROTRUSIONS, OR GATE BURRS.

	INCHES		MILLIMETERS	
DIM	MIN	MAX	MIN	MAX
A	0.071	0.087	1.80	2.20
B	0.045	0.053	1.15	1.35
C	0.031	0.043	0.80	1.10
D	0.004	0.012	0.10	0.30
G	0.026 BSC		0.65 BSC	
H	---	0.004	---	0.10
J	0.004	0.010	0.10	0.25
K	0.004	0.012	0.10	0.30
N	0.008 REF		0.20 REF	
S	0.079	0.087	2.00	2.20

SOLDER FOOTPRINT



GENERIC MARKING DIAGRAM*



XXX = Specific Device Code
M = Date Code
▪ = Pb−Free Package

(Note: Microdot may be in either location)

*This information is generic. Please refer to device data sheet for actual part marking. Pb−Free indicator, “G” or microdot “▪”, may or may not be present. Some products may not follow the Generic Marking.

STYLE 1:
PIN 1. BASE
2. EMITTER
3. BASE
4. COLLECTOR
5. COLLECTOR

STYLE 2:
PIN 1. ANODE
2. EMITTER
3. BASE
4. COLLECTOR
5. CATHODE

STYLE 3:
PIN 1. ANODE 1
2. N/C
3. ANODE 2
4. CATHODE 2
5. CATHODE 1

STYLE 4:
PIN 1. SOURCE 1
2. DRAIN 1/2
3. SOURCE 1
4. GATE 1
5. GATE 2

STYLE 5:
PIN 1. CATHODE
2. COMMON ANODE
3. CATHODE 2
4. CATHODE 3
5. CATHODE 4

STYLE 6:
PIN 1. EMITTER 2
2. BASE 2
3. EMITTER 1
4. COLLECTOR
5. COLLECTOR 2/BASE 1

STYLE 7:
PIN 1. BASE
2. EMITTER
3. BASE
4. COLLECTOR
5. COLLECTOR

STYLE 8:
PIN 1. CATHODE
2. COLLECTOR
3. N/C
4. BASE
5. EMITTER

STYLE 9:
PIN 1. ANODE
2. CATHODE
3. ANODE
4. ANODE
5. ANODE

Note: Please refer to datasheet for style callout. If style type is not called out in the datasheet refer to the device datasheet pinout or pin assignment.

DOCUMENT NUMBER:	**98ASB42984B**	Electronic versions are uncontrolled except when accessed directly from the Document Repository. Printed versions are uncontrolled except when stamped “CONTROLLED COPY” in red.	
DESCRIPTION:	**SC−88A (SC−70−5/SOT−353)**		**PAGE 1 OF 1**

PUBLICATION ORDERING INFORMATION

LITERATURE FULFILLMENT:
Email Requests to: orderlit@onsemi.com

ON Semiconductor Website: www.onsemi.com

TECHNICAL SUPPORT
North American Technical Support:
Voice Mail: 1 800−282−9855 Toll Free USA/Canada
Phone: 011 421 33 790 2910

Europe, Middle East and Africa Technical Support:
Phone: 00421 33 790 2910
For additional information, please contact your local Sales Representative

◊

Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

ON Semiconductor:

NCP612SQ15T2G NCP612SQ18T2G NCP612SQ25T2G NCP612SQ27T2G NCP612SQ28T2G NCP612SQ30T2G NCP612SQ31T2G NCP612SQ33T2G NCP612SQ37T2G NCP612SQ50T2G NCV612SQ15T2G NCV612SQ18T2G NCV612SQ25T2G NCV612SQ27T2G NCV612SQ28T2G NCV612SQ30T2G NCV612SQ31T2G NCV612SQ33T2G NCV612SQ50T2G NCV612SQ37T2G

TLV755P 500-mA, Low I_Q, Small Size, Low Dropout Regulator

1 Features

- Input Voltage Range: 1.44 V to 5.5 V
- Available in Fixed-Output Voltages:
  - 0.6 V to 5 V (50-mV Steps)
- Low I_Q: 25 µA (Typical)
- Low Dropout:
  - 220 mV (Maximum) at 500 mA (3.3 V_{OUT})
- Output Accuracy: 1% (Typical)
- Built-In Soft-Start With Monotonic V_{OUT} Rise
- Foldback Current Limit
- Active Output Discharge
- High PSRR: 45 dB at 100 kHz
- Stable With a 1-µF Ceramic Output Capacitor
- Packages:
  - 2.9-mm × 1.6-mm SOT-23-5
  - 1-mm x 1-mm X2SON-4
  - 2 mm × 2 mm WSON-6

2 Applications

- Set-Top Boxes, TV, and Gaming Consoles
- Portable and Battery-Powered Equipment
- Desktop, Notebooks, and Ultrabooks
- Tablets and Remote Controls
- White Goods and Appliances
- Grid Infrastructure and Protection Relays
- Camera Modules and Image Sensors

3 Description

The TLV755P low-dropout regulator (LDO) device is an ultra-small, low quiescent current LDO that sources 500 mA with good line and load transient performance. The TLV755P is optimized for a wide variety of applications by supporting an input voltage range from 1.44 V to 5.5 V. To minimize cost and solution size, the device is offered in fixed output voltages ranging from 0.6 V to 5 V to support the lower core voltages of modern microcontroller (MCUs). Additionally, the TLV755 has a low I_Q with enable functionality to minimize standby power. This device features an internal soft-start to lower inrush current, thus providing a controlled voltage to the load and minimizing the input voltage drop during start up. When shutdown, the device actively pulls down the output to quickly discharge the outputs and ensure a known start-up state.

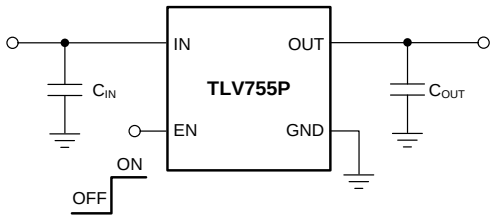
The TLV755P is stable with small ceramic output capacitors allowing for a small overall solution size. A precision band-gap and error amplifier provides a typical accuracy of 1%. All device versions have integrated thermal shutdown, current limit, and undervoltage lockout (UVLO). The TLV755P has an internal foldback current limit that helps reduce the thermal dissipation during short-circuit events.

Device Information(1)

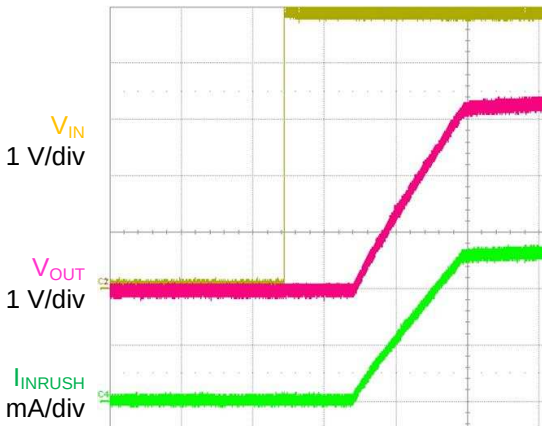
PART NUMBER	PACKAGE	BODY SIZE (NOM)
TLV755P	X2SON (4)	1.00 mm × 1.00 mm
	SOT-23 (5)	2.90 mm × 1.60 mm
	SON (6)	2.00 mm × 2.00 mm

(1) For all available packages, see the orderable addendum at the end of the data sheet.

Typical Application



Startup Waveform



ADVANCE INFORMATION

Table of Contents

1 Features ... 1
2 Applications ... 1
3 Description ... 1
4 Revision History ... 2
5 Pin Configuration and Functions ... 3
6 Specifications ... 4
6.1 Absolute Maximum Ratings ... 4
6.2 ESD Ratings ... 4
6.3 Recommended Operating Conditions ... 4
6.4 Thermal Information ... 4
6.5 Electrical Characteristics ... 5
6.6 Typical Characteristics ... 7
7 Detailed Description ... 8
7.1 Overview ... 8
7.2 Functional Block Diagram ... 8
7.3 Feature Description ... 8
7.4 Device Functional Modes ... 10
8 Application and Implementation ... 11
8.1 Application Information ... 11
8.2 Typical Application ... 14
9 Power Supply Recommendations ... 15
10 Layout ... 15
10.1 Layout Guidelines ... 15
10.2 Layout Examples ... 16
11 Device and Documentation Support ... 17
11.1 Device Support ... 17
11.2 Receiving Notification of Documentation Updates 17
11.3 Community Resources ... 17
11.4 Trademarks ... 17
11.5 Electrostatic Discharge Caution ... 17
11.6 Glossary ... 17
12 Mechanical, Packaging, and Orderable Information ... 17

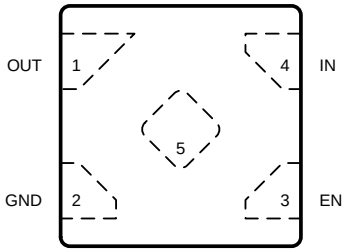
ADVANCE INFORMATION

4 Revision History

DATE	REVISION	NOTES
November 2017	*	Initial release.

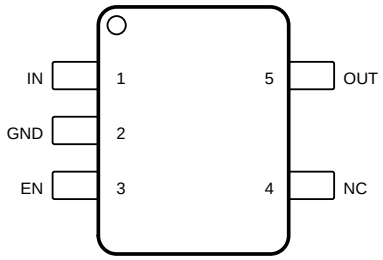
5 Pin Configuration and Functions

DQN Package
6-Pin X2SON
Top View



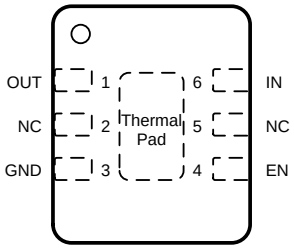
Not to scale

DBV Package
5-Pin SOT-23
Top View



Not to scale

DRV Package
6-Pin WSON With Exposed Thermal Pad
Top View



Not to scale

NC = no internal connection.

Pin Functions

PIN				I/O	DESCRIPTION
NAME	DQN	DBV	DRV		
EN	3	3	4	I	Enable pin. Drive EN greater than V_{HI} to turn on the regulator. Drive EN less than V_{LO} to place the LDO into shutdown mode.
GND	2	2	3	—	Ground pin.
IN	4	1	6	I	Input pin. A capacitor with a value of 1 µF or larger is required from this pin to ground(1). See the *Input and Output Capacitor Selection* section for more information.
NC	—	4	2, 5	—	No internal connection.
OUT	1	5	1	O	Regulated output voltage pin. A capacitor with a value of 1 µF or larger is required from this pin to ground(1). See the *Input and Output Capacitor Selection* section for more information.
Thermal pad	—	—	Pad	—	Connect the thermal pad to a large-area ground plane. The thermal pad is internally connected to GND.

(1) The nominal input and output capacitance must be greater than 0.47 µF; throughout this document the nominal derating on these capacitors is 50%. Take care to ensure that the effective capacitance at the pin is greater than 0.47 µF.

ADVANCE INFORMATION

ADVANCE INFORMATION

6 Specifications

6.1 Absolute Maximum Ratings

over operating free-air temperature range (unless otherwise noted)(1)

		MIN	MAX	UNIT
Voltage	Supply, V_{IN}	–0.3	6	V
	Enable, V_{EN}	–0.3	6	
	Output, V_{OUT}	–0.3	V_{IN} + 0.3(2)	
Temperature	Operating junction, T_J	–40	150	°C
	Storage, T_{stg}	–65	150	

(1) Stresses beyond those listed under *Absolute Maximum Ratings* may cause permanent damage to the device. These are stress ratings only, which do not imply functional operation of the device at these or any other conditions beyond those indicated under *Recommended Operating Conditions*. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.
(2) The absolute maximum rating is V_{IN} + 0.3 V or 6 V, whichever is smaller.

6.2 ESD Ratings

			VALUE	UNIT
$V_{(ESD)}$	Electrostatic discharge	Human-body model (HBM), per ANSI/ESDA/JEDEC JS-001(1)	±1000	V
		Charged-device model (CDM), per JEDEC specification JESD22-C101(2)	±500	

(1) JEDEC document JEP155 states that 500-V HBM allows safe manufacturing with a standard ESD control process. Manufacturing with less than 500-V HBM is possible with the necessary precautions.
(2) JEDEC document JEP157 states that 250-V CDM allows safe manufacturing with a standard ESD control process. Manufacturing with less than 250-V CDM is possible with the necessary precautions.

6.3 Recommended Operating Conditions

over operating free-air temperature range (unless otherwise noted)

		MIN	NOM	MAX	UNIT
C_{IN}	Input capacitor	1			µF
C_{OUT}	Output capacitor	1		200	µF
V_{IN}	Input voltage	1.44		5.5	V
V_{OUT}	Output voltage	0.6		5	V
I_{OUT}	Output current	0		500	mA
V_{EN}	Enable voltage	0		5.5	V
f_{EN}	Enable toggle frequency			10	kHz
T_J	Junction temperature	–40		125	°C

6.4 Thermal Information

THERMAL METRIC(1)		TLV755P			UNIT
		DQN (X2SON)	DBV (SOT-23)	DRV (WSON)	
		4 Pins	5 PINS	6 PINS	
$R_{\theta JA}$	Junction-to-ambient thermal resistance	168.4	231.1	100.2	°C/W
$R_{\theta JC(top)}$	Junction-to-case (top) thermal resistance	139.1	118.4	108.5	°C/W
$R_{\theta JB}$	Junction-to-board thermal resistance	101.4	64.4	64.3	°C/W
ψ_{JT}	Junction-to-top characterization parameter	5.6	28.4	10.4	°C/W
ψ_{JB}	Junction-to-board characterization parameter	101.7	63.8	64.8	°C/W
$R_{\theta JC(bot)}$	Junction-to-case (bottom) thermal resistance	88.4	N/A	34.7	°C/W

(1) For more information about traditional and new thermal metrics, see the Semiconductor and IC Package Thermal Metrics application report.

6.5 Electrical Characteristics

over operating free-air temperature range (T_J = –40°C to +125°C), V_{IN} = V_{OUT} + 0.5 V or 1.44 V (whichever is greater), I_{OUT} = 1 mA, V_{EN} = V_{IN}, and C_{IN} = C_{OUT} = 1 µF (unless otherwise noted); all typical values are at T_J = 25°C

PARAMETER		TEST CONDITIONS		MIN	TYP	MAX	UNIT
V_{IN}	Input voltage			1.44		5.5	V
V_{OUT}	Output voltage			0.6		5	V
	Output accuracy	T_J = 25°C		–1%		1%	
		–40°C ≤ T_J ≤ +85°C, V_{OUT} ≥ 1 V		–1%		1%	
		–40°C ≤ T_J ≤ +85°C, 0.6 V ≤ V_{OUT} < 1 V		–10		10	mV
		V_{OUT} ≥ 1 V		–1.5%		1.5%	
		0.6 V ≤ V_{OUT} < 1 V		–15		15	mV
$(\Delta V_{OUT})_{\Delta VIN}$/ V_{OUT}	Line regulation	V_{OUT} + 0.5 V[1] ≤ V_{IN} ≤ 5.5 V	V_{OUT} > 1.5 V		0.01%		
ΔV_{OUT}/ ΔI_{OUT}	Load regulation	0.1 mA ≤ I_{OUT} ≤ 500 mA, V_{IN} ≥ 2.4 V	DQN package		0.018		V/A
			DBV package		0.030		
			DRV package		0.022		
I_{GND}	Ground current	T_J = 25°C, I_{OUT} = 0 mA		14	25	31	µA
		–40°C ≤ T_J ≤ +85°C, I_{OUT} = 0 mA				33	
		–40°C ≤ T_J ≤ +125°C, I_{OUT} = 0 mA				40	
I_{SHDN}	Shutdown current	V_{EN}= 0 V, 1.44 V ≤ V_{IN} ≤ 5.5 V, –40°C ≤ T_J ≤ +125°C			0.1	1	µA
I_{CL}	Output current limit	V_{IN} = V_{OUT} + $V_{DO(MAX)}$ + 0.25 V	V_{OUT} = V_{OUT} – 0.2 V, V_{OUT} ≤ 1.5 V	600	720	865	mA
			V_{OUT} = 0.9 × V_{OUT}, 1.5 V < V_{OUT} ≤ 4.5 V	600	720	865	
I_{SC}	Short-circuit current limit	V_{OUT} = 0 V			355		mA
V_{DO}	Dropout voltage[2]	I_{OUT} = 500 mA, –40°C ≤ T_J ≤ +85°C	0.6 V ≤ V_{OUT} < 0.8 V		675	700	mV
			0.8 V ≤ V_{OUT} < 1 V		600	650	
			1 V ≤ V_{OUT} < 1.2 V		550	575	
			1.2 V ≤ V_{OUT} < 1.5 V		500	525	
			1.5 V ≤ V_{OUT} < 1.8 V		350	400	
			1.8 V ≤ V_{OUT} < 2.5 V		325	375	
			2.5 V ≤ V_{OUT} < 3.3 V		250	300	
			3.3 V ≤ V_{OUT} ≤ 5.0 V		150	212	
		I_{OUT} = 500 mA, –40°C ≤ T_J ≤ +125°C	0.6 V ≤ V_{OUT} < 0.8 V			725	
			0.8 V ≤ V_{OUT} < 1 V			675	
			1 V ≤ V_{OUT} < 1.2 V			600	
			1.2 V ≤ V_{OUT} < 1.5 V			550	
			1.5 V ≤ V_{OUT} < 1.8 V			425	
			1.8 V ≤ V_{OUT} < 2.5 V			400	
			2.5 V ≤ V_{OUT} < 3.3 V			325	
			3.3 V ≤ V_{OUT} ≤ 5.0 V			238	
PSRR	Power-supply rejection ratio	f = 1 kHz, V_{IN} = V_{OUT} + 1 V, I_{OUT} = 50 mA			52		dB
		f = 100 kHz, V_{IN} = V_{OUT} + 1 V, I_{OUT} = 50 mA			46		
		f = 1 MHz, V_{IN} = V_{OUT} + 1 V, I_{OUT} = 50 mA			52		
V_n	Output noise voltage	BW = 10 Hz to 100 kHz, V_{OUT} = 1.2 V, I_{OUT} = 50 mA			71.5		μV_{RMS}
V_{UVLO}	Undervoltage lockout	V_{IN} rising		1.21	1.3	1.44	V
$V_{UVLO, HYST}$	Undervoltage lockout hysteresis	V_{IN} falling			40		mV

(1) V_{IN} = 1.44 V for V_{OUT} < 0.9 V.
(2) Dropout is measured when V_{OUT} is 5% below $V_{OUT(NOM)}$.

ADVANCE INFORMATION

Electrical Characteristics (continued)

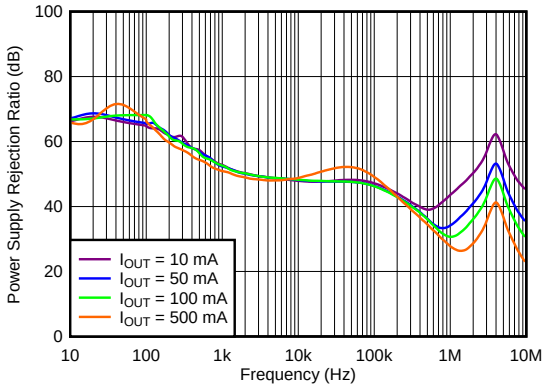
over operating free-air temperature range (T_J = –40°C to +125°C), V_{IN} = V_{OUT} + 0.5 V or 1.44 V (whichever is greater), I_{OUT} = 1 mA, V_{EN} = V_{IN}, and C_{IN} = C_{OUT} = 1 µF (unless otherwise noted); all typical values are at T_J = 25°C

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{STR}	Startup time	Time from EN assertion to 0.95 × V_{OUT}		400		µs
V_{HI}	EN pin high voltage (enabled)		0.9			V
V_{LO}	EN pin low voltage (enabled)				0.4	V
I_{EN}	Enable pin current	EN = 5.5 V, V_{IN} = 5.5 V		10		nA
$R_{PULLDOWN}$	Pulldown resistance	V_{IN} = 3.3 V		120		Ω
T_{SD}	Thermal shutdown	Shutdown, temperature increasing		165		°C
		Reset, temperature decreasing		155		

ADVANCE INFORMATION

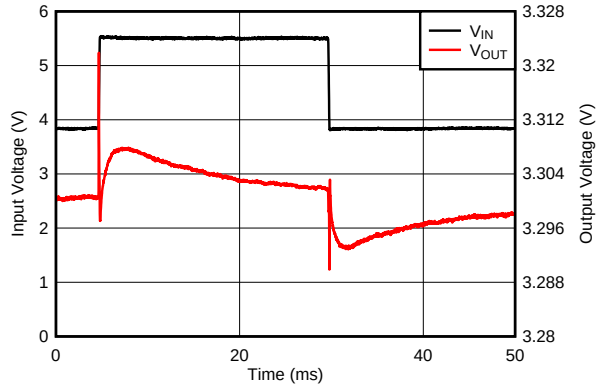
6.6 Typical Characteristics

at operating temperature T_J = 25°C, V_{IN} = $V_{OUT(NOM)}$ + 0.5 V or 1.4 V (whichever is greater), I_{OUT} = 1 mA, V_{EN} = V_{IN}, and C_{IN} = C_{OUT} = 1 µF (unless otherwise noted)



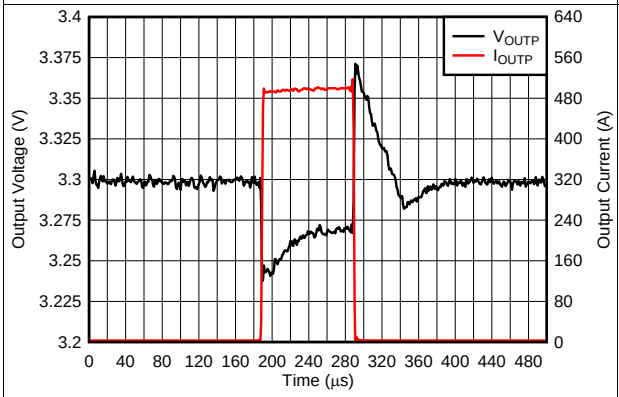
V_{IN} = 4.3 V, V_{OUT} = 3.3 V, C_{OUT} = 1 µF

Figure 1. PSRR vs Frequency and I_{OUT}



V_{OUT} = 3.3 V, C_{OUT} = 1 µF, V_{IN} slew rate = 1 V/µs

Figure 2. Line Transient



V_{IN} = 4.3 V, V_{OUT} = 3.3 V, C_{OUT} = 1 µF, I_{OUT} slew rate = 1 A/µs

Figure 3. 1-mA to 500-mA Load Transient

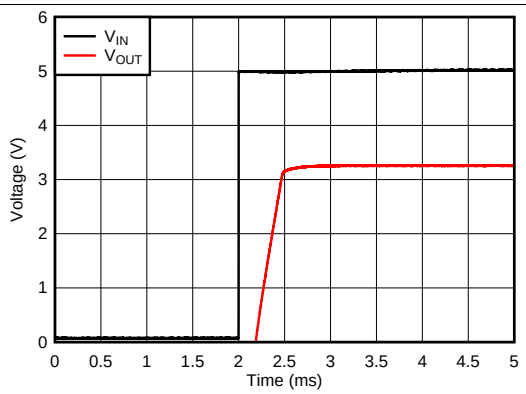


Figure 4. V_{IN} = V_{EN} Power-Up

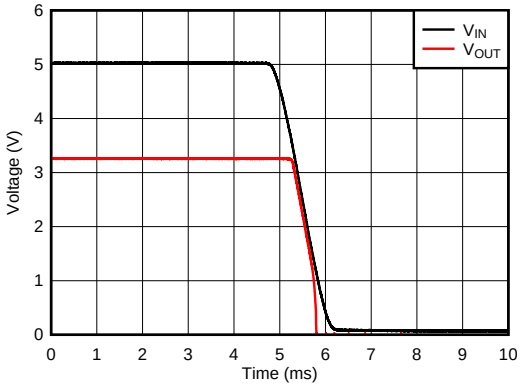


Figure 5. V_{IN} = V_{EN} Shutdown

ADVANCE INFORMATION

ADVANCE INFORMATION

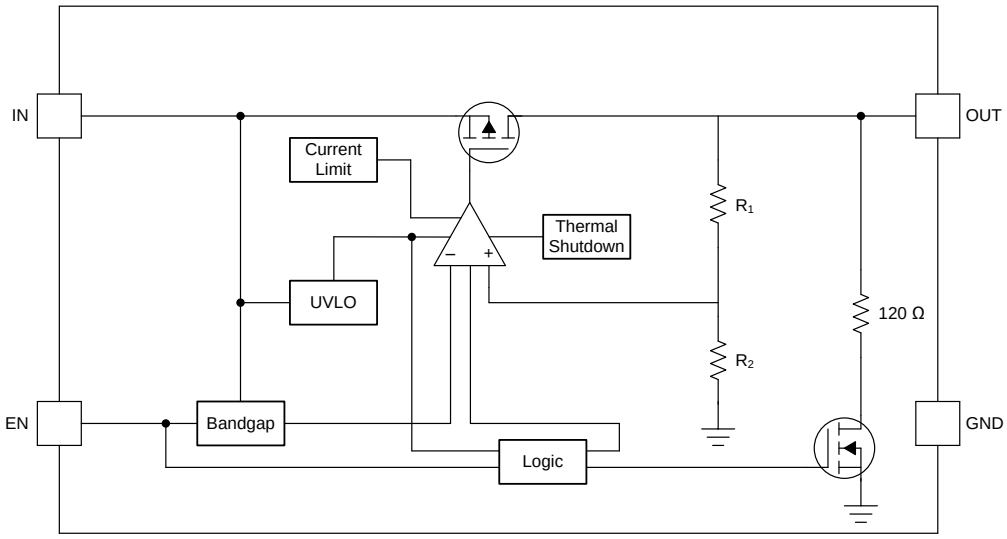
7 Detailed Description

7.1 Overview

The TLV755P belongs to a family of next-generation, low-dropout regulators (LDOs). This device consumes low quiescent current and delivers excellent line and load transient performance. The TLV755P is optimized for a wide variety of applications by supporting an input voltage range from 1.4 V to 5.5 V. To minimize cost and solution size, the device is offered in fixed output voltages ranging from 0.6 V to 5 V to support the lower core voltages of modern MCUs.

This regulator offers foldback current limit, shutdown, and thermal protection. The operating junction temperature is –40°C to +125°C.

7.2 Functional Block Diagram



NOTE: $R_1 + R_2$ = TBD.

7.3 Feature Description

7.3.1 Undervoltage Lockout (UVLO)

An undervoltage lockout (UVLO) circuit disables the output until the input voltage is greater than the rising UVLO voltage (V_{UVLO}). This circuit ensures that the device does not exhibit any unpredictable behavior when the supply voltage is lower than the operational range of the internal circuitry. When V_{IN} is less than V_{UVLO}, the output is connected to ground with a 120-Ω pulldown resistor.

7.3.2 Enable (EN)

The enable pin (EN) is active high. Enable the device by forcing the EN pin to exceed V_{HI}. Turn off the device by forcing the EN pin below V_{LO}. If shutdown capability is not required, connect EN to IN.

The device has an internal pulldown resistor that connects a 120-Ω resistor to ground when the device is disabled. The discharge time after disabling depends on the output capacitance (C_{OUT}) and the load resistance (R_L) in parallel with the 120-Ω pulldown resistor. Equation 1 calculates the time constant τ:

$$\tau = \frac{120 \cdot R_L}{120 + R_L} \cdot C_{OUT} \quad (1)$$

Feature Description (continued)

7.3.3 Internal Foldback Current Limit

The TLV755P has an internal current limit that protects the regulator during fault conditions. The current limit is a hybrid brick-wall scheme until the output voltage is less than 0.4 × $V_{OUT(NOM)}$. When the voltage drops below 0.4 × $V_{OUT(NOM)}$, a foldback current limit is implemented that scales back the current as the output voltage approaches GND. When the output shorts, the LDO supplies a typical current of I_{SC}. The output voltage is not regulated when the device is in current limit. In this condition, the output voltage is the product of the regulated current and the load resistance. When the device output shorts, the PMOS pass transistor dissipates power [(V_{IN} – V_{OUT}) × I_{SC}] until thermal shutdown is triggered and the device turns off. After the device cools down, the internal thermal shutdown circuit turns the device back on. If the fault condition continues, the device cycles between current limit and thermal shutdown.

The foldback current-limit circuit limits the current that is allowed through the device to current levels lower than the minimum current limit at nominal V_{OUT} current limit (I_{CL}) during start up. Figure 6 shows typical current limit values. If the output is loaded by a constant-current load during start up, or if the output voltage is negative when the device is enabled, then the load current demanded by the load may exceed the foldback current limit and the device may not rise to the full output voltage. For constant-current loads, disable the output load until the output rises to the nominal voltage.

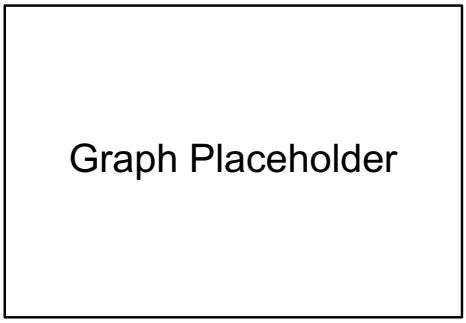


Figure 6. TLV755 Current Limit vs V_{OUT}

7.3.4 Thermal Shutdown

Thermal shutdown protection disables the output when the junction temperature rises to approximately 165°C. Disabling the device eliminates the power dissipated by the device, allowing the device to cool. When the junction temperature cools to approximately 155°C, the output circuitry is enabled again. Depending on power dissipation, thermal resistance, and ambient temperature, the thermal protection circuit may cycle on and off. This cycling limits regulator dissipation and protects the circuit from damage as a result of overheating.

Activating the thermal shutdown feature usually indicates excessive power dissipation as a result of the product of the (V_{IN} – V_{OUT}) voltage and the load current. For reliable operation, limit junction temperature to a maximum of 125°C. To estimate the margin of safety in a complete design, increase the ambient temperature until the thermal protection is triggered; use worst-case loads and signal conditions.

The internal protection circuitry protects against overload conditions but is not intended to be activated in normal operation. Continuously running the device into thermal shutdown degrades device reliability.

ADVANCE INFORMATION

7.4 Device Functional Modes

Table 1 lists a comparison between the normal, dropout, and disabled modes of operation.

Table 1. Device Functional Modes Comparison

OPERATING MODE	PARAMETER			
	V_{IN}	EN	I_{OUT}	T_J
Normal(1)	$V_{IN} > V_{OUT(NOM)} + V_{DO}$	$V_{EN} > V_{HI}$	$I_{OUT} < I_{CL}$	$T_J < T_{SD}$
Dropout(1)	$V_{IN} < V_{OUT(NOM)} + V_{DO}$	$V_{EN} > V_{HI}$	—	$T_J < T_{SD}$
Disabled(2)	$V_{IN} < V_{UVLO}$	$V_{EN} < V_{LO}$	—	$T_J > T_{SD}$

(1) All table conditions must be met.
(2) The device is disabled when any condition is met.

7.4.1 Normal Operation

The device regulates to the nominal output voltage when all following conditions are met:

- The input voltage is greater than the nominal output voltage plus the dropout voltage ($V_{OUT(NOM)} + V_{DO}$)
- The enable voltage has previously exceeded the enable rising threshold voltage and has not decreased below the enable falling threshold
- The output current is less than the current limit ($I_{OUT} < I_{CL}$)
- The device junction temperature is less than the thermal shutdown temperature ($T_J < T_{SD}$)

7.4.2 Dropout Operation

If the input voltage is lower than the nominal output voltage plus the specified dropout voltage, but all other conditions are met for normal operation, the device operates in dropout. In this mode, the output voltage tracks the input voltage. During this mode, the transient performance of the device degrades because the pass device is in a triode state and no longer controls the current through the LDO. Line or load transients in dropout can result in large output-voltage deviations.

When the device is in a steady dropout state (defined as when the device is in dropout, $V_{IN} < V_{OUT(NOM)} + V_{DO}$, right after being in a normal regulation state, but not during startup), the pass-FET is driven as hard as possible when the control loop is out of balance. During the normal time required for the device to regain regulation, $V_{IN} \geq V_{OUT(NOM)} + V_{DO}$, V_{OUT} can overshoot $V_{OUT(NOM)}$ during fast transients.

7.4.3 Disabled

The output is shut down by forcing the enable pin below V_{LO}. When disabled, the pass device is turned off, internal circuits are shut down, and the output voltage is actively discharged to ground by an internal switch from the output to ground. The active pulldown resistor is on when sufficient input voltage is provided.

ADVANCE INFORMATION

8 Application and Implementation

NOTE

Information in the following applications sections is not part of the TI component specification, and TI does not warrant its accuracy or completeness. TI's customers are responsible for determining suitability of components for their purposes. Customers should validate and test their design implementation to confirm system functionality.

8.1 Application Information

8.1.1 Input and Output Capacitor Selection

The TLV755P requires an output capacitance of 0.47 µF or larger for stability. Use X5R- and X7R-type ceramic capacitors because these capacitors have minimal variation in capacitance value and equivalent series resistance (ESR) over temperature. When selecting a capacitor for a specific application, consider the dc bias characteristics for the capacitor. Higher output voltages cause a significant derating of the capacitor. As a general rule, ceramic capacitors must be derated by 50%. For best performance, TI recommends a maximum output capacitance value of 200 µF.

Place a 1-µF capacitor on the input pin of the LDO. Some input supplies have a high impedance, which places the input capacitor on the input supply and reduces the input impedance. This capacitor counteracts reactive input sources and improves transient response and PSRR. If the input supply has a high impedance over a large range of frequencies, several input capacitors are used in parallel to lower the impedance over frequency. Use a higher-value capacitor if large, fast, rise-time load transients are expected, or if the device is located several inches from the input power source.

8.1.2 Dropout Voltage

Use a PMOS pass transistor achieves low dropout. When (V_{IN} – V_{OUT}) is less than the dropout voltage (V_{DO}), the PMOS pass device is in the linear region of operation and the input-to-output resistance is the $R_{DS(ON)}$ of the PMOS pass element. V_{DO} scales linearly with the output current because the PMOS device functions like a resistor in dropout mode. As with any linear regulator, PSRR and transient response degrade as (V_{IN} – V_{OUT}) approaches dropout operation. Figure 7 shows typical dropout values.

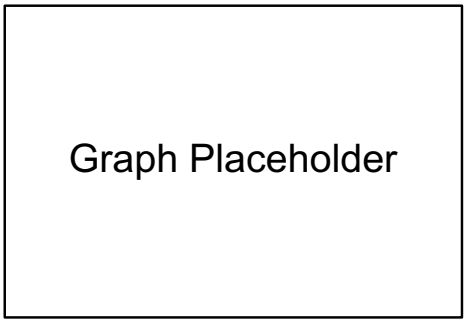


Figure 7. Dropout vs V_{IN}

ADVANCE INFORMATION

Application Information (continued)

8.1.3 Exiting Dropout

Some applications have transients that place the LDO into dropout, such as slower ramps on V_{IN} for start-up. As with other LDOs, the output overshoots on recovery from these conditions. Figure 8 shows that a ramping input supply causes an LDO to overshoot on start-up when the slew rate and voltage levels are in the correct range. Use an enable signal to avoid this condition.

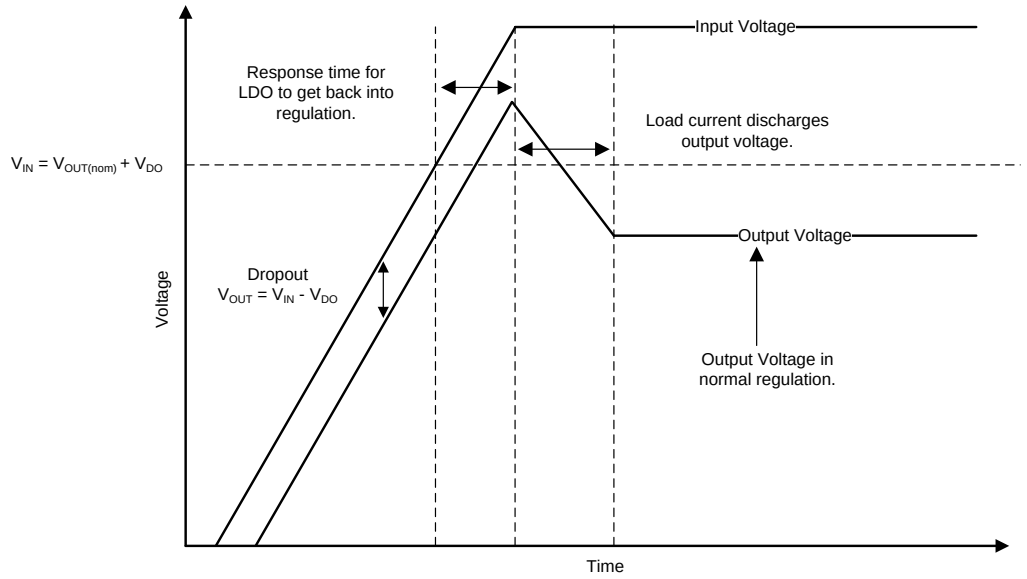


Figure 8. Startup Into Dropout

8.1.4 Reverse Current

As with most LDOs, excessive reverse current can damage this device.

Reverse current flows through the body diode on the pass element instead of the normal conducting channel. At high magnitudes, this current flow degrades the long-term reliability of the device as a result of one of the following conditions:

- Degradation caused by electromigration
- Excessive heat dissipation
- Potential for a latch-up condition

ADVANCE INFORMATION

Application Information (continued)

Conditions where reverse current can occur are outlined in this section, all of which can exceed the absolute maximum rating of $V_{OUT} > V_{IN} + 0.3$ V:

- If the device has a large C_{OUT} and the input supply collapses with little or no load current
- The output is biased when the input supply is not established
- The output is biased above the input supply

If reverse current flow is expected in the application, external protection must be used to protect the device. Figure 9 shows one approach of protecting the device.

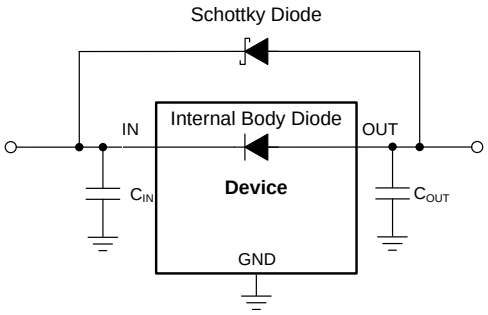


Figure 9. Example Circuit for Reverse Current Protection Using a Schottky Diode

8.1.5 Power Dissipation (P_D)

Circuit reliability demands that proper consideration is given to device power dissipation, location of the circuit on the printed circuit board (PCB), and correct sizing of the thermal plane. The PCB area around the regulator must be as free of other heat-generating devices as possible that cause added thermal stresses.

As a first-order approximation, power dissipation in the regulator depends on the input-to-output voltage difference and load conditions. Use Equation 2 to approximate P_D:

$$P_D = (V_{IN} – V_{OUT}) \times I_{OUT} \quad (2)$$

Power dissipation must be minimized to achieve greater efficiency. This minimizing process is achieved by selecting the correct system voltage rails. Proper selection helps obtain the minimum input-to-output voltage differential. The low dropout of the device allows for maximum efficiency across a wide range of output voltages.

The main heat conduction path for the device is through the thermal pad on the package. As such, the thermal pad must be soldered to a copper pad area under the device. This pad area contains an array of plated vias that conduct heat to inner plane areas or to a bottom-side copper plane.

The maximum power dissipation determines the maximum allowable junction temperature (T_J) for the device. According to Equation 3, power dissipation and junction temperature are most often related by the junction-to-ambient thermal resistance ($R_{\theta JA}$) of the combined PCB, device package, and the temperature of the ambient air (T_A).

$$T_J = T_A + R_{\theta JA} \times P_D \quad (3)$$

Unfortunately, this thermal resistance ($R_{\theta JA}$) is dependent on the heat-spreading capability built into the particular PCB design, and therefore varies according to the total copper area, copper weight, and location of the planes. The $R_{\theta JA}$ value recorded in the table is determined by the JEDEC standard, PCB, and copper-spreading area. The $R_{\theta JA}$ value is only used as a relative measure of package thermal performance. $R_{\theta JA}$ is the sum of the DRV package junction-to-case (bottom) thermal resistance ($R_{\theta JC(bot)}$) plus the thermal resistance contribution by the PCB copper.

ADVANCE INFORMATION

Application Information (continued)

8.1.5.1 Estimating Junction Temperature

The JEDEC standard recommends the use of psi (Ψ) thermal metrics to estimate the junction temperatures of the LDO when in-circuit on a typical PCB board application. These metrics are not thermal resistances, but offer practical and relative means of estimating junction temperatures. These psi metrics are independent of the copper-spreading area. The key thermal metrics (Ψ_{JT} and Ψ_{JB}) are used in accordance with Equation 4 and are given in the table.

$$\Psi_{JT}: T_J = T_T + \Psi_{JT} \times P_D$$
$$\Psi_{JB}: T_J = T_B + \Psi_{JB} \times P_D$$

where:

- P_D is the power dissipated as described in Equation 2
- T_T is the temperature at the center-top of the device package, and
- T_B is the PCB surface temperature measured 1 mm from the device package and centered on the package edge (4)

8.2 Typical Application

This application shows a typical use case for the TLV755, where the device converts from 2.5 V to 1.8 V for a common MCU core rail.

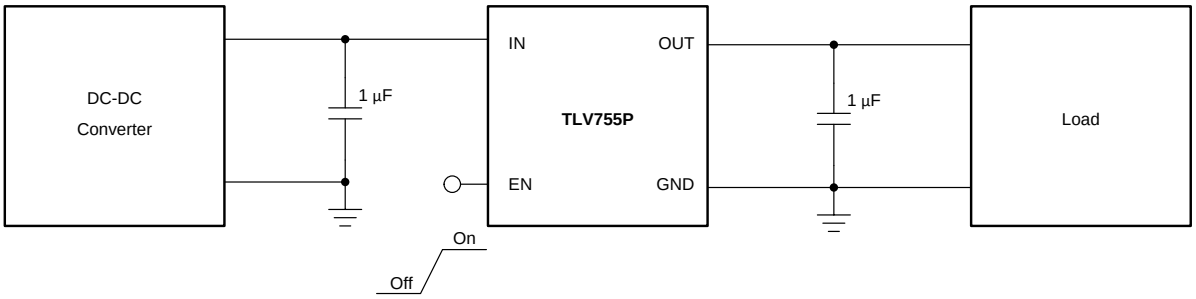


Figure 10. TLV755P Typical Application

8.2.1 Design Requirements

Table 2 lists the design requirements for this application.

Table 2. Design Parameters

PARAMETER	DESIGN REQUIREMENT
Input voltage	2.5 V
Output voltage	1.8 V
Input current	400 mA (maximum)
Output load	300-mA dc
Maximum ambient temperature	85°C

ADVANCE INFORMATION

8.2.2 Detailed Design Procedure

8.2.2.1 Input Current

During normal operation, the input current to the LDO is approximately equal to the output current of the LDO. During startup, the input current is higher as a result of the inrush current charging the output capacitor. Use Equation 5 to calculate the current through the input.

$$I_{OUT(t)} = \left[\frac{C_{OUT} \times dV_{OUT}(t)}{dt}\right] + \left[\frac{V_{OUT}(t)}{R_{LOAD}}\right]$$

where:

- $V_{OUT}(t)$ is the instantaneous output voltage of the turnon ramp
- $dV_{OUT}(t) / dt$ is the slope of the V_{OUT} ramp
- R_{LOAD} is the resistive load impedance (5)

8.2.2.2 Thermal Dissipation

The junction temperature can be determined using the junction-to-ambient thermal resistance ($R_{\theta JA}$) and the total power dissipation (P_D). Use Equation 6 to calculate the power dissipation. As Equation 7 shows, multiply P_D by $R_{\theta JA}$ and add the ambient temperature (T_A) to calculate the junction temperature (T_J).

$$P_D = (I_{GND} + I_{OUT}) \times (V_{IN} - V_{OUT}) \quad (6)$$

$$T_J = R_{\theta JA} \times P_D + T_A \quad (7)$$

If the ($T_{J(MAX)}$) value does not exceed 125°C, use Equation 8 to calculate the maximum ambient temperature. Equation 9 calculates the maximum ambient temperature with a value of 103.39°C.

$$T_{A(MAX)} = T_{J(MAX)} - R_{\theta JA} \times P_D \quad (8)$$

$$T_{A(MAX)} = 125°C - 102.9 \times (2.5\ V - 1.8\ V) \times (0.3\ A) = 103.39°C \quad (9)$$

9 Power Supply Recommendations

Connect a low output impedance power supply directly to the IN pin of the TLV755P. If the input source is reactive, consider using multiple input capacitors in parallel with the 1-µF input capacitor to lower the input supply impedance over frequency.

10 Layout

10.1 Layout Guidelines

- Place input and output capacitors as close as possible to the device
- Use copper planes for device connections to optimize thermal performance
- Place thermal vias around the device to distribute the heat

ADVANCE INFORMATION

10.2 Layout Examples

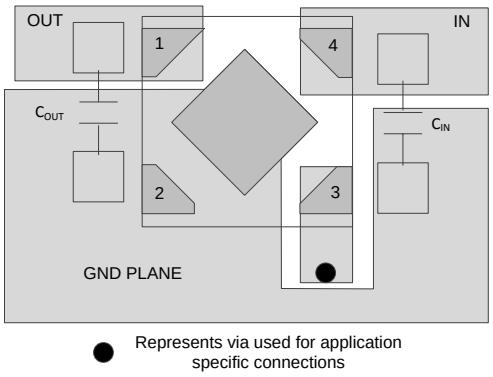


Figure 11. Layout Example: DQN Package

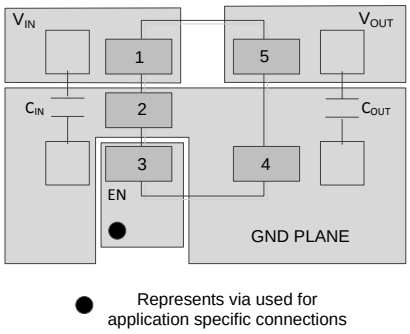


Figure 12. Layout Example: DBV Package

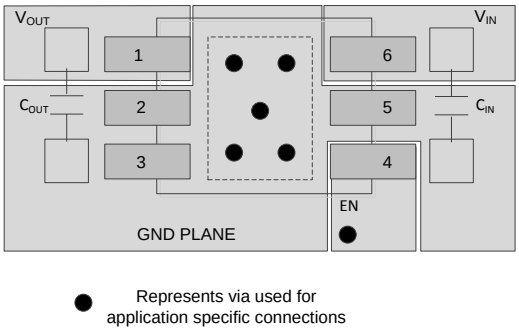


Figure 13. Layout Example: DRV Package

ADVANCE INFORMATION

11 Device and Documentation Support

11.1 Device Support

11.1.1 Device Nomenclature

Table 3. Device Nomenclature(1)(2)

PRODUCT	V_{OUT}
TLV755**xx(x)Pyyyz**	**xx(x)** is the nominal output voltage. For output voltages with a resolution of 100 mV, two digits are used in the ordering number; otherwise, three digits are used (for example, 28 = 2.8 V; 125 = 1.25 V). **P** indicates an active output discharge feature. All members of the TLV755 family will actively discharge the output when the device is disabled. **yyy** is the package designator. **z** is the package quantity. R is for reel (3000 pieces), T is for tape (250 pieces).

(1) For the most current package and ordering information see the Package Option Addendum at the end of this document, or visit the device product folder on www.ti.com.
(2) Output voltages from 0.6 V to 5 V in 50-mV increments are available. Contact the factory for details and availability.

11.2 Receiving Notification of Documentation Updates

To receive notification of documentation updates, navigate to the device product folder on ti.com. In the upper right corner, click on *Alert me* to register and receive a weekly digest of any product information that has changed. For change details, review the revision history included in any revised document.

11.3 Community Resources

The following links connect to TI community resources. Linked contents are provided "AS IS" by the respective contributors. They do not constitute TI specifications and do not necessarily reflect TI's views; see TI's Terms of Use.

TI E2E™ Online Community ***TI's Engineer-to-Engineer (E2E) Community.*** Created to foster collaboration among engineers. At e2e.ti.com, you can ask questions, share knowledge, explore ideas and help solve problems with fellow engineers.

Design Support ***TI's Design Support*** Quickly find helpful E2E forums along with design support tools and contact information for technical support.

11.4 Trademarks

E2E is a trademark of Texas Instruments.
All other trademarks are the property of their respective owners.

11.5 Electrostatic Discharge Caution

This integrated circuit can be damaged by ESD. Texas Instruments recommends that all integrated circuits be handled with appropriate precautions. Failure to observe proper handling and installation procedures can cause damage.

ESD damage can range from subtle performance degradation to complete device failure. Precision integrated circuits may be more susceptible to damage because very small parametric changes could cause the device not to meet its published specifications.

11.6 Glossary

SLYZ022 — *TI Glossary*.

This glossary lists and explains terms, acronyms, and definitions.

12 Mechanical, Packaging, and Orderable Information

The following pages include mechanical, packaging, and orderable information. This information is the most current data available for the designated devices. This data is subject to change without notice and revision of this document. For browser-based versions of this data sheet, refer to the left-hand navigation.

ADVANCE INFORMATION

PACKAGING INFORMATION

Orderable Device	Status (1)	Package Type	Package Drawing	Pins	Package Qty	Eco Plan (2)	Lead/Ball Finish (6)	MSL Peak Temp (3)	Op Temp (°C)	Device Marking (4/5)	Samples
PTLV75507PDQNR	ACTIVE	X2SON	DQN	4	3000	TBD	Call TI	Call TI	-40 to 125		Samples
PTLV75509PDBVR	ACTIVE	SOT-23	DBV	5	3000	TBD	Call TI	Call TI	-40 to 125		Samples
PTLV75510PDBVR	ACTIVE	SOT-23	DBV	5	3000	TBD	Call TI	Call TI	-40 to 125		Samples
PTLV75510PDQNR	ACTIVE	X2SON	DQN	4	3000	TBD	Call TI	Call TI	-40 to 125		Samples
PTLV75512PDBVR	ACTIVE	SOT-23	DBV	5	3000	TBD	Call TI	Call TI	-40 to 125		Samples
PTLV75512PDQNR	ACTIVE	X2SON	DQN	4	3000	TBD	Call TI	Call TI	-40 to 125		Samples
PTLV75515PDBVR	ACTIVE	SOT-23	DBV	5	3000	TBD	Call TI	Call TI	-40 to 125		Samples
PTLV75515PDQNR	ACTIVE	X2SON	DQN	4	3000	TBD	Call TI	Call TI	-40 to 125		Samples
PTLV75518PDBVR	ACTIVE	SOT-23	DBV	5	3000	TBD	Call TI	Call TI	-40 to 125		Samples
PTLV75518PDQNR	ACTIVE	X2SON	DQN	4	3000	TBD	Call TI	Call TI	-40 to 125		Samples
PTLV75519PDBVR	ACTIVE	SOT-23	DBV	5	3000	TBD	Call TI	Call TI	-40 to 125		Samples
PTLV75519PDQNR	ACTIVE	X2SON	DQN	4	3000	TBD	Call TI	Call TI	-40 to 125		Samples
PTLV75525PDBVR	ACTIVE	SOT-23	DBV	5	3000	TBD	Call TI	Call TI	-40 to 125		Samples
PTLV75525PDQNR	ACTIVE	X2SON	DQN	4	3000	TBD	Call TI	Call TI	-40 to 125		Samples
PTLV75528PDBVR	ACTIVE	SOT-23	DBV	5	3000	TBD	Call TI	Call TI	-40 to 125		Samples
PTLV75528PDQNR	ACTIVE	X2SON	DQN	4	3000	TBD	Call TI	Call TI	-40 to 125		Samples
PTLV75529PDBVR	ACTIVE	SOT-23	DBV	5	3000	TBD	Call TI	Call TI	-40 to 125		Samples
PTLV75530PDBVR	ACTIVE	SOT-23	DBV	5	3000	TBD	Call TI	Call TI	-40 to 125		Samples
PTLV75530PDQNR	ACTIVE	X2SON	DQN	4	3000	TBD	Call TI	Call TI	-40 to 125		Samples
PTLV75533PDBVR	ACTIVE	SOT-23	DBV	5	3000	TBD	Call TI	Call TI	-40 to 125		Samples

Orderable Device	Status (1)	Package Type	Package Drawing	Pins	Package Qty	Eco Plan (2)	Lead/Ball Finish (6)	MSL Peak Temp (3)	Op Temp (°C)	Device Marking (4/5)	Samples
PTLV75533PDQNR	ACTIVE	X2SON	DQN	4	3000	TBD	Call TI	Call TI	-40 to 125		Samples
TLV75507PDQNR	PREVIEW	X2SON	DQN	4	3000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	-40 to 125	KD	
TLV75507PDQNT	PREVIEW	X2SON	DQN	4	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	-40 to 125	KD	
TLV75509PDBVR	PREVIEW	SOT-23	DBV	5	3000	Green (RoHS & no Sb/Br)	CU SN	Level-1-260C-UNLIM	-40 to 125		
TLV75509PDQNR	PREVIEW	X2SON	DQN	4	3000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	-40 to 125	AX	
TLV75509PDQNT	PREVIEW	X2SON	DQN	4	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	-40 to 125	AX	
TLV75510PDBVR	PREVIEW	SOT-23	DBV	5	3000	Green (RoHS & no Sb/Br)	CU NIPDAU | CU SN	Level-1-260C-UNLIM	-40 to 125	1FPF	
TLV75510PDQNR	PREVIEW	X2SON	DQN	4	3000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	-40 to 125	KE	
TLV75510PDQNT	PREVIEW	X2SON	DQN	4	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	-40 to 125	KE	
TLV75512PDBVR	PREVIEW	SOT-23	DBV	5	3000	Green (RoHS & no Sb/Br)	CU SN	Level-1-260C-UNLIM	-40 to 125		
TLV75512PDQNR	PREVIEW	X2SON	DQN	4	3000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	-40 to 125	AG	
TLV75512PDQNT	PREVIEW	X2SON	DQN	4	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	-40 to 125	AG	
TLV75515PDBVR	PREVIEW	SOT-23	DBV	5	3000	Green (RoHS & no Sb/Br)	CU SN	Level-1-260C-UNLIM	-40 to 125		
TLV75515PDQNR	PREVIEW	X2SON	DQN	4	3000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	-40 to 125	KF	
TLV75515PDQNT	PREVIEW	X2SON	DQN	4	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	-40 to 125	KF	
TLV75518PDBVR	PREVIEW	SOT-23	DBV	5	3000	Green (RoHS & no Sb/Br)	CU SN	Level-1-260C-UNLIM	-40 to 125		
TLV75518PDQNR	PREVIEW	X2SON	DQN	4	3000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	-40 to 125	AI	
TLV75518PDQNT	PREVIEW	X2SON	DQN	4	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	-40 to 125	AI	

Orderable Device	Status (1)	Package Type	Package Drawing	Pins	Package Qty	Eco Plan (2)	Lead/Ball Finish (6)	MSL Peak Temp (3)	Op Temp (°C)	Device Marking (4/5)	Samples
TLV75519PDBVR	PREVIEW	SOT-23	DBV	5	3000	Green (RoHS & no Sb/Br)	CU SN	Level-1-260C-UNLIM	-40 to 125		
TLV75519PDQNR	PREVIEW	X2SON	DQN	4	3000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	-40 to 125	B5	
TLV75519PDQNT	PREVIEW	X2SON	DQN	4	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	-40 to 125	B5	
TLV75525PDBVR	PREVIEW	SOT-23	DBV	5	3000	Green (RoHS & no Sb/Br)	CU SN	Level-1-260C-UNLIM	-40 to 125		
TLV75525PDQNR	PREVIEW	X2SON	DQN	4	3000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	-40 to 125	AJ	
TLV75525PDQNT	PREVIEW	X2SON	DQN	4	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	-40 to 125	AJ	
TLV75528PDBVR	PREVIEW	SOT-23	DBV	5	3000	Green (RoHS & no Sb/Br)	CU SN	Level-1-260C-UNLIM	-40 to 125		
TLV75528PDQNR	PREVIEW	X2SON	DQN	4	3000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	-40 to 125	KG	
TLV75528PDQNT	PREVIEW	X2SON	DQN	4	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	-40 to 125	KG	
TLV75529PDBVR	PREVIEW	SOT-23	DBV	5	3000	Green (RoHS & no Sb/Br)	CU SN	Level-1-260C-UNLIM	-40 to 125		
TLV75530PDBVR	PREVIEW	SOT-23	DBV	5	3000	Green (RoHS & no Sb/Br)	CU SN	Level-1-260C-UNLIM	-40 to 125		
TLV75530PDQNR	PREVIEW	X2SON	DQN	4	3000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	-40 to 125	KI	
TLV75530PDQNT	PREVIEW	X2SON	DQN	4	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	-40 to 125	KI	
TLV75533PDBVR	PREVIEW	SOT-23	DBV	5	3000	Green (RoHS & no Sb/Br)	CU SN	Level-1-260C-UNLIM	-40 to 125		
TLV75533PDQNR	PREVIEW	X2SON	DQN	4	3000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	-40 to 125	AN	
TLV75533PDQNT	PREVIEW	X2SON	DQN	4	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM	-40 to 125	AN	

(1) The marketing status values are defined as follows:
ACTIVE: Product device recommended for new designs.
LIFEBUY: TI has announced that the device will be discontinued, and a lifetime-buy period is in effect.
NRND: Not recommended for new designs. Device is in production to support existing customers, but TI does not recommend using this part in a new design.

PREVIEW: Device has been announced but is not in production. Samples may or may not be available.
OBSOLETE: TI has discontinued the production of the device.

(2) RoHS: TI defines "RoHS" to mean semiconductor products that are compliant with the current EU RoHS requirements for all 10 RoHS substances, including the requirement that RoHS substance do not exceed 0.1% by weight in homogeneous materials. Where designed to be soldered at high temperatures, "RoHS" products are suitable for use in specified lead-free processes. TI may reference these types of products as "Pb-Free".
RoHS Exempt: TI defines "RoHS Exempt" to mean products that contain lead but are compliant with EU RoHS pursuant to a specific EU RoHS exemption.
Green: TI defines "Green" to mean the content of Chlorine (Cl) and Bromine (Br) based flame retardants meet JS709B low halogen requirements of <=1000ppm threshold. Antimony trioxide based flame retardants must also meet the <=1000ppm threshold requirement.

(3) MSL, Peak Temp. - The Moisture Sensitivity Level rating according to the JEDEC industry standard classifications, and peak solder temperature.

(4) There may be additional marking, which relates to the logo, the lot trace code information, or the environmental category on the device.

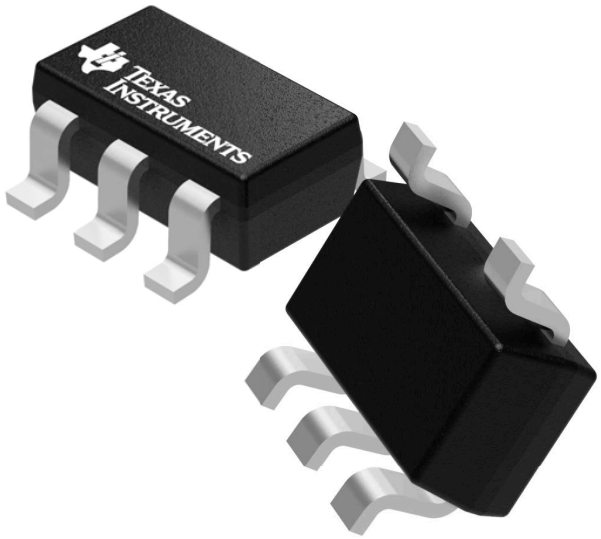
(5) Multiple Device Markings will be inside parentheses. Only one Device Marking contained in parentheses and separated by a "~" will appear on a device. If a line is indented then it is a continuation of the previous line and the two combined represent the entire Device Marking for that device.

(6) Lead/Ball Finish - Orderable Devices may have multiple material finish options. Finish options are separated by a vertical ruled line. Lead/Ball Finish values may wrap to two lines if the finish value exceeds the maximum column width.

GENERIC PACKAGE VIEW

DBV 5 SOT-23 - 1.45 mm max height

SMALL OUTLINE TRANSISTOR



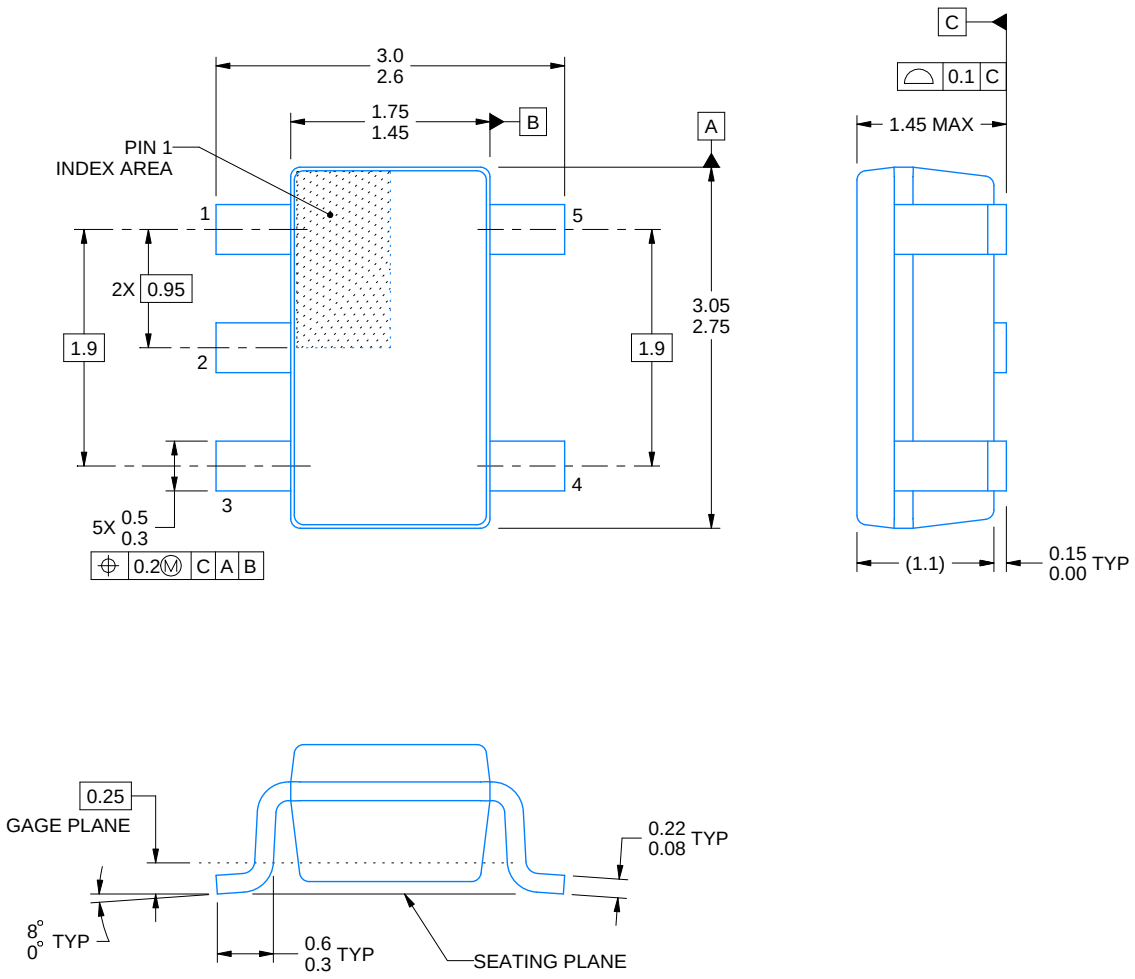
Images above are just a representation of the package family, actual package may vary.
Refer to the product data sheet for package details.

PACKAGE OUTLINE

DBV0005A

SOT-23 - 1.45 mm max height

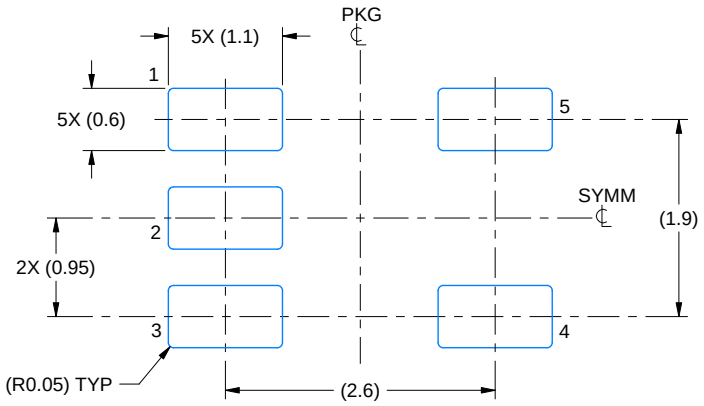
SMALL OUTLINE TRANSISTOR



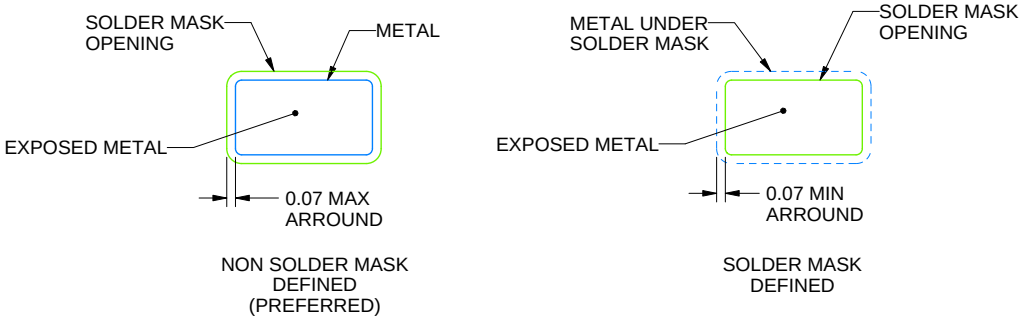
4214839/C 04/2017

NOTES:

1. All linear dimensions are in millimeters. Any dimensions in parenthesis are for reference only. Dimensioning and tolerancing per ASME Y14.5M.
2. This drawing is subject to change without notice.
3. Refernce JEDEC MO-178.



LAND PATTERN EXAMPLE
EXPOSED METAL SHOWN
SCALE:15X

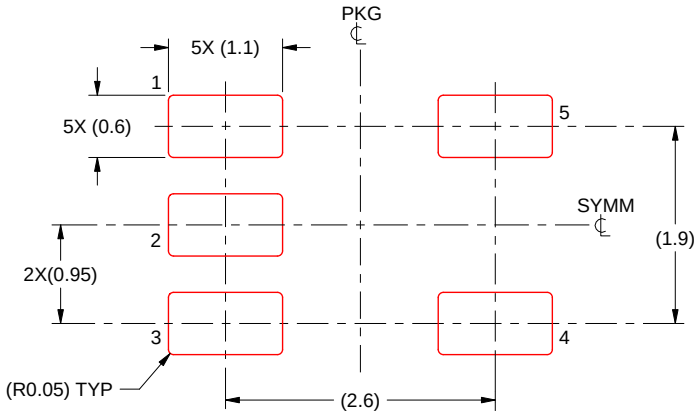


SOLDER MASK DETAILS

4214839/C 04/2017

NOTES: (continued)

4. Publication IPC-7351 may have alternate designs.
5. Solder mask tolerances between and around signal pads can vary based on board fabrication site.



SOLDER PASTE EXAMPLE
BASED ON 0.125 mm THICK STENCIL
SCALE:15X

4214839/C 04/2017

NOTES: (continued)

6. Laser cutting apertures with trapezoidal walls and rounded corners may offer better paste release. IPC-7525 may have alternate design recommendations.
7. Board assembly site may have different recommendations for stencil design.

GENERIC PACKAGE VIEW

DQN 4 X2SON - 0.4 mm max height

PLASTIC SMALL OUTLINE - NO LEAD

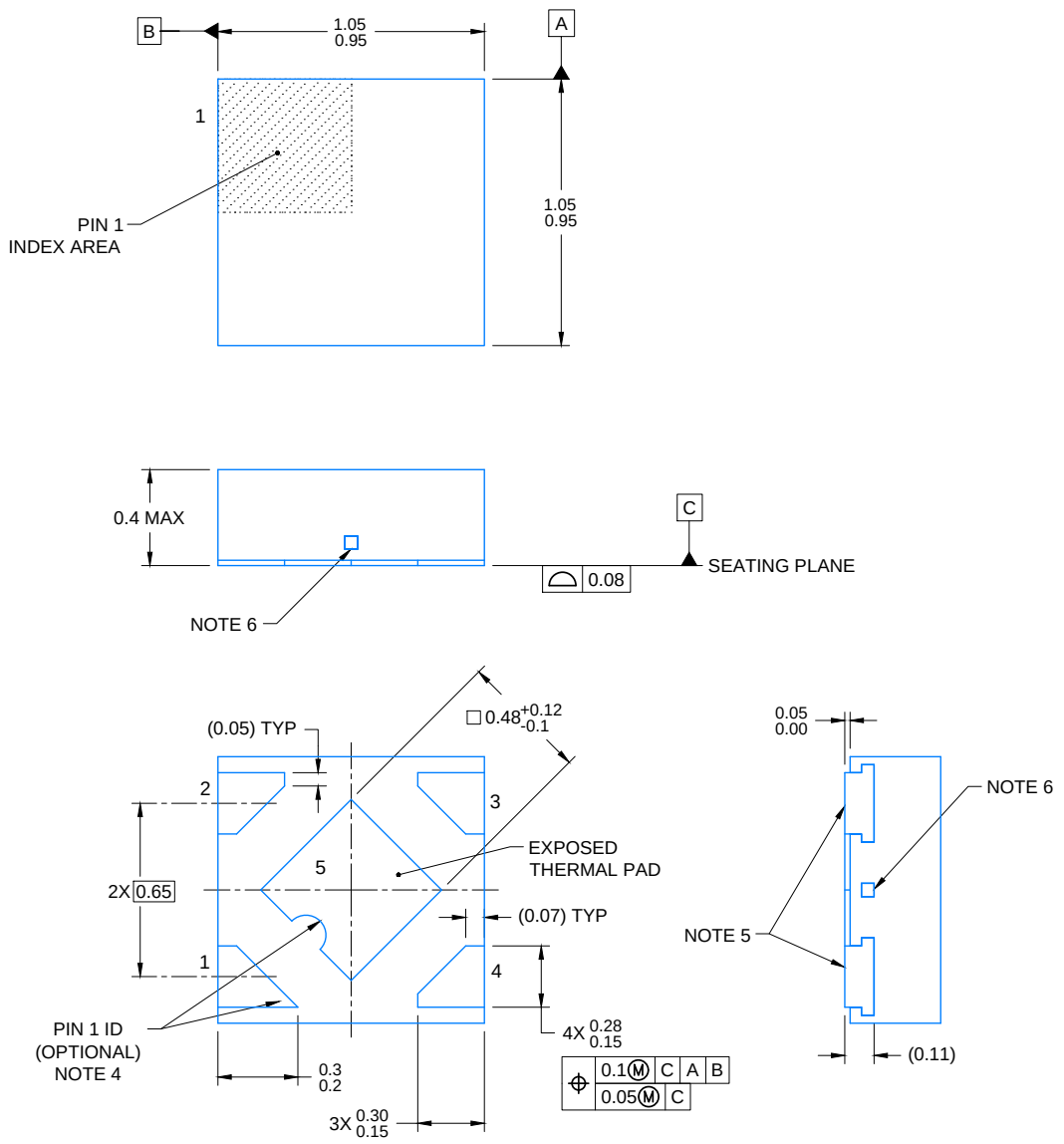
Images above are just a representation of the package family, actual package may vary.
Refer to the product data sheet for package details.

PACKAGE OUTLINE

DQN0004A

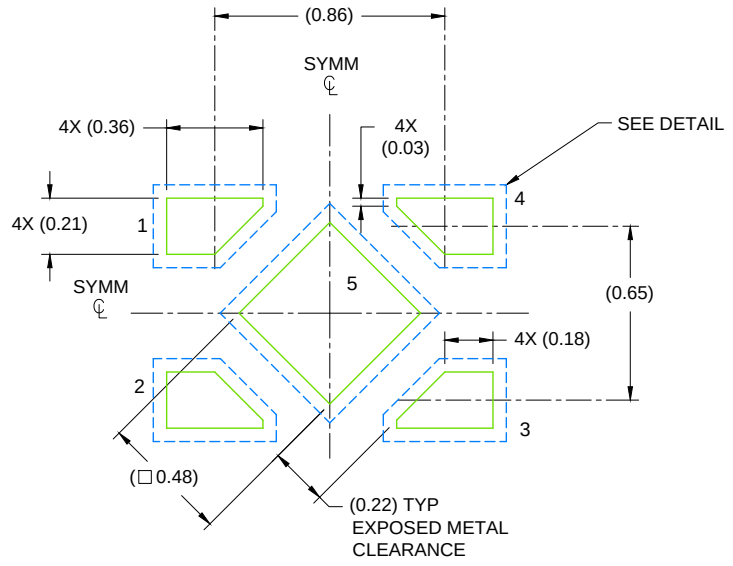
X2SON - 0.4 mm max height

PLASTIC SMALL OUTLINE - NO LEAD



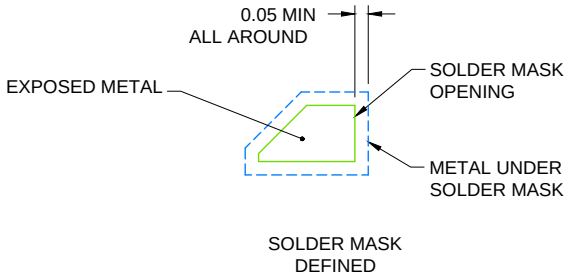
NOTES:

1. All linear dimensions are in millimeters. Any dimensions in parenthesis are for reference only. Dimensioning and tolerancing per ASME Y14.5M.
2. This drawing is subject to change without notice.
3. The package thermal pad must be soldered to the printed circuit board for optimal thermal and mechanical performance.
4. Features may not exist. Recommend use of pin 1 marking on top of package for orientation purposes.
5. Shape of exposed side leads may differ.
6. Number and location of exposed tie bars may vary.



LAND PATTERN EXAMPLE

SCALE: 40X

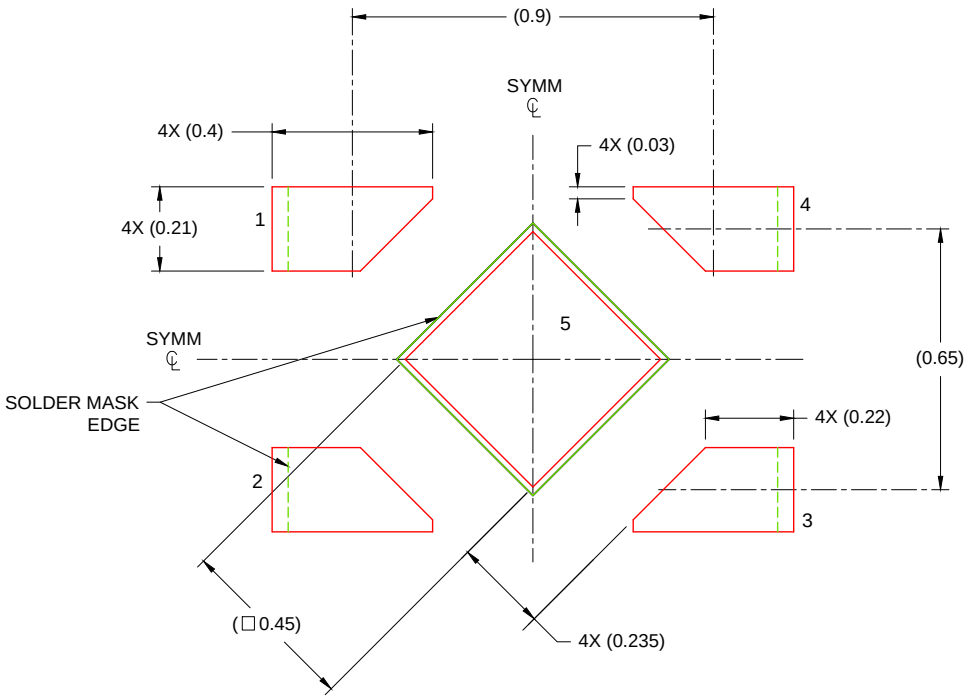


SOLDER MASK DETAIL

4215302/E 12/2016

NOTES: (continued)

7. This package is designed to be soldered to a thermal pad on the board. For more information, see Texas Instruments literature number SLUA271 (www.ti.com/lit/slua271).
8. If any vias are implemented, it is recommended that vias under paste be filled, plugged or tented.



SOLDER PASTE EXAMPLE
BASED ON 0.075 - 0.1mm THICK STENCIL

EXPOSED PAD
88% PRINTED SOLDER COVERAGE BY AREA
SCALE: 60X

4215302/E 12/2016

NOTES: (continued)

9. Laser cutting apertures with trapezoidal walls and rounded corners may offer better paste release. IPC-7525 may have alternate design recommendations.

IMPORTANT NOTICE

Texas Instruments Incorporated (TI) reserves the right to make corrections, enhancements, improvements and other changes to its semiconductor products and services per JESD46, latest issue, and to discontinue any product or service per JESD48, latest issue. Buyers should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

TI's published terms of sale for semiconductor products (http://www.ti.com/sc/docs/stdterms.htm) apply to the sale of packaged integrated circuit products that TI has qualified and released to market. Additional terms may apply to the use or sale of other types of TI products and services.

Reproduction of significant portions of TI information in TI data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. TI is not responsible or liable for such reproduced documentation. Information of third parties may be subject to additional restrictions. Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Buyers and others who are developing systems that incorporate TI products (collectively, "Designers") understand and agree that Designers remain responsible for using their independent analysis, evaluation and judgment in designing their applications and that Designers have full and exclusive responsibility to assure the safety of Designers' applications and compliance of their applications (and of all TI products used in or for Designers' applications) with all applicable regulations, laws and other applicable requirements. Designer represents that, with respect to their applications, Designer has all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. Designer agrees that prior to using or distributing any applications that include TI products, Designer will thoroughly test such applications and the functionality of such TI products as used in such applications.

TI's provision of technical, application or other design advice, quality characterization, reliability data or other services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using TI Resources in any way, Designer (individually or, if Designer is acting on behalf of a company, Designer's company) agrees to use any particular TI Resource solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

Designer is authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS. TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY DESIGNER AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Unless TI has explicitly designated an individual product as meeting the requirements of a particular industry standard (e.g., ISO/TS 16949 and ISO 26262), TI is not responsible for any failure to meet such industry standard requirements.

Where TI specifically promotes products as facilitating functional safety or as compliant with industry functional safety standards, such products are intended to help enable customers to design and create their own applications that meet applicable functional safety standards and requirements. Using products in an application does not by itself establish any safety features in the application. Designers must ensure compliance with safety-related requirements and standards applicable to their applications. Designer may not use any TI products in life-critical medical equipment unless authorized officers of the parties have executed a special contract specifically governing such use. Life-critical medical equipment is medical equipment where failure of such equipment would cause serious bodily injury or death (e.g., life support, pacemakers, defibrillators, heart pumps, neurostimulators, and implantables). Such equipment includes, without limitation, all medical devices identified by the U.S. Food and Drug Administration as Class III devices and equivalent classifications outside the U.S.

TI may expressly designate certain products as completing a particular qualification (e.g., Q100, Military Grade, or Enhanced Product). Designers agree that it has the necessary expertise to select the product with the appropriate qualification designation for their applications and that proper product selection is at Designers' own risk. Designers are solely responsible for compliance with all legal and regulatory requirements in connection with such selection.

Designer will fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of Designer's non-compliance with the terms and provisions of this Notice.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265

TP4056 1A Standalone Linear Li-Ion Battery Charger with Thermal Regulation in SOP-8

DESCRIPTION

The TP4056 is a complete constant-current/constant-voltage linear charger for single cell lithium-ion batteries. Its SOP package and low external component count make the TP4056 ideally suited for portable applications. Furthermore, the TP4056 can work within USB and wall adapter.

No blocking diode is required due to the internal PMOSFET architecture and have prevent to negative Charge Current Circuit. Thermal feedback regulates the charge current to limit the die temperature during high power operation or high ambient temperature. The charge voltage is fixed at 4.2V, and the charge current can be programmed externally with a single resistor. The TP4056 automatically terminates the charge cycle when the charge current drops to 1/10th the programmed value after the final float voltage is reached.

TP4056 Other features include current monitor, under voltage lockout, automatic recharge and two status pin to indicate charge termination and the presence of an input voltage.

FEATURES

- Programmable Charge Current Up to 1000mA
- No MOSFET, Sense Resistor or Blocking Diode Required
- Complete Linear Charger in SOP-8 Package for Single Cell Lithium-Ion Batteries
- Constant-Current/Constant-Voltage
- Charges Single Cell Li-Ion Batteries Directly from USB Port
- Preset 4.2V Charge Voltage with 1.5% Accuracy
- Automatic Recharge
- two Charge Status Output Pins
- C/10 Charge Termination
- 2.9V Trickle Charge Threshold (TP4056)
- Soft-Start Limits Inrush Current
- Available Radiator in 8-Lead SOP Package, the Radiator need connect GND or impending

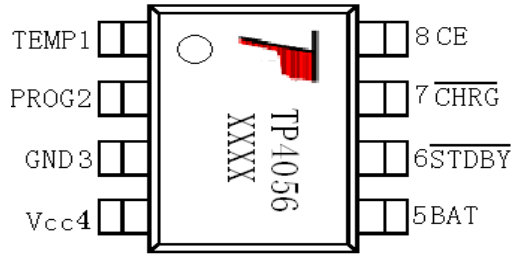
ABSOLUTE MAXIMUM RATINGS

- Input Supply Voltage(V_{CC})：-0.3V～8V
- TEMP：**-0.3V**～10V
- CE：**-0.3V**～10V
- BAT Short-Circuit Duration：Continuous
- BAT Pin Current：1200mA
- PROG Pin Current：1200uA
- Maximum Junction Temperature：145℃
- Operating Ambient Temperature Range：-40℃～85℃
- Lead Temp.(Soldering, 10sec)：260℃

APPLICATIONS

- Cellular Telephones, PDAs, GPS
- Charging Docks and Cradles
- Digital Still Cameras, Portable Devices
- USB Bus-Powered Chargers,Chargers

PACKAGE/ORDER INFORMATION



SOP－8

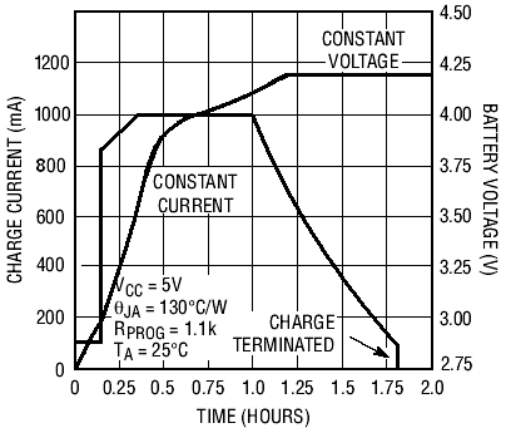
photo

ORDER PART NUMBER

TP4056-42-SOP8-PP

PART MARKING TP4056

Complete Charge Cycle (1000mAh Battery)



TEMP(Pin 1) :Temperature Sense Input Connecting TEMP pin to NTC thermistor's output in Lithium ion battery pack. If TEMP pin's voltage is below 45% or above 80% of supply voltage V_{IN} for more than 0.15S, this means that battery's temperature is too high or too low, charging is suspended. The temperature sense function can be disabled by grounding the TEMP pin.

PROG(Pin 2): Constant Charge Current Setting and Charge Current Monitor Pin charge current is set by connecting a resistor R_{ISET} from this pin to GND. When in precharge mode, the ISET pin's voltage is regulated to 0.2V. When in constant charge current mode, the ISET pin's voltage is regulated to 2V.In all modes during charging, the voltage on ISET pin can be used to measure the charge current as follows:

$$I_{BAT} = \frac{V_{PROG}}{R_{PROG}} \times 1200 \quad (V_{PROG}=1V)$$

GND(Pin3): Ground Terminal

Vcc(Pin 4): Positive Input Supply Voltage V_{IN} is the power supply to the internal circuit. When V_{IN} drops to within 30mv of the BAT pin voltage, TP4056 enters low power sleep mode, dropping BAT pin's current to less than 2uA.

BAT(Pin5): Battery Connection Pin. Connect the positive terminal of the battery to BAT pin. BAT pin draws less than 2uA current in chip disable mode or in sleep mode. BAT pin provides charge current to the battery and provides regulation voltage of 4.2V.

$\overline{STDBY}$(Pin6): Open Drain Charge Status Output When the battery Charge Termination, the $\overline{STDBY}$ pin is pulled low by an internal switch, otherwise $\overline{STDBY}$ pin is in high impedance state.

$\overline{CHRG}$(Pin7): Open Drain Charge Status Output When the battery is being charged, the $\overline{CHRG}$ pin is pulled low by an internal switch, otherwise $\overline{CHRG}$ pin is in high impedance state.

CE(Pin8): Chip Enable Input. A high input will put the device in the normal operating mode. Pulling the CE pin to low level will put the YP4056 into disable mode. The CE pin can be driven by TTL or CMOS logic level.

ELECTRICAL CHARACTERISTICS

The ● denotes specifications which apply over the full operating temperature range, otherwise specifications are at **T_A=25℃， V_{CC}=5V**，unless otherwise noted.

SYMBOL	PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
V_{CC}	Input Supply Voltage		●	4.0	5	8.0	V
I_{CC}	Input Supply Current	Charge Mode, R_{PROG} = 1.2k StandbyMode(Charge Terminated) Shutdown Mode (R_{PROG} Not Connected,V_{CC} < V_{BAT}, or V_{CC} < V_{UV})	● ● ●		150 55 55	500 100 100	μA μA μA
V_{FLOAL}	Regulated Output (Float) Voltage	0℃≤T_A≤85℃， I_{BAT}=40mA		4.137	4.2	4.263	V
I_{BAT}	BAT Pin Current Text condition:VBAT=4.0V	R_{PROG} = 2.4k, Current Mode R_{PROG} = 1.2k, Current Mode Standby Mode, V_{BAT} = 4.2V	● ● ●	450 950 0	500 1000 －2.5	550 1050 －6	mA mA μA
I_{TRIKL}	Trickle Charge Current	V_{BAT}<V_{TRIKL}, R_{PROG}=1.2K	●	120	130	140	mA
V_{TRIKL}	Trickle Charge Threshold Voltage	R_{PROG}=1.2K, V_{BAT} Rising		2.8	2.9	3.0	V
V_{TRHYS}	Trickle Charge Hysteresis Voltage	R_{PROG}=1.2K		60	80	100	mV
T_{LIM}	Junction Temperature in Constant Temperature Mode				145		℃

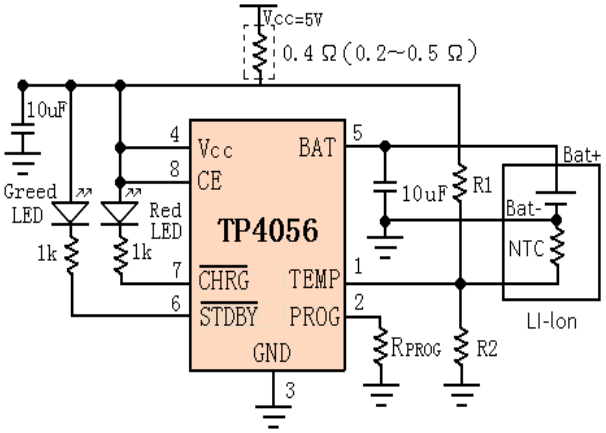
indicator light state

Charge state	Red LED $\overline{CHRG}$	Greed LED $\overline{STDBY}$
charging	bright	extinguish
Charge Termination	extinguish	bright
Vin too low; Temperature of battery too low or too high; no battery	extinguish	extinguish
BAT PIN Connect 10u Capacitance; No battery	Greed LED bright，Red LED Coruscate T=1-4 S	

Rprog Current Setting

R_{PROG} (k)	I_{BAT} (mA)
10	130
5	250
4	300
3	400
2	580
1.66	690
1.5	780
1.33	900
1.2	1000

TYPICAL APPLICATIONS



TP4056 1A Standalone Linear Li-Ion Battery Charger with Thermal Regulation in SOP-8

DESCRIPTION

The TP4056 is a complete constant-current/constant-voltage linear charger for single cell lithium-ion batteries. Its SOP package and low external component count make the TP4056 ideally suited for portable applications. Furthermore, the TP4056 can work within USB and wall adapter.

No blocking diode is required due to the internal PMOSFET architecture and have prevent to negative Charge Current Circuit. Thermal feedback regulates the charge current to limit the die temperature during high power operation or high ambient temperature. The charge voltage is fixed at 4.2V, and the charge current can be programmed externally with a single resistor. The TP4056 automatically terminates the charge cycle when the charge current drops to 1/10th the programmed value after the final float voltage is reached.

TP4056 Other features include current monitor, under voltage lockout, automatic recharge and two status pin to indicate charge termination and the presence of an input voltage.

FEATURES

- Programmable Charge Current Up to 1000mA
- No MOSFET, Sense Resistor or Blocking Diode Required
- Complete Linear Charger in SOP-8 Package for Single Cell Lithium-Ion Batteries
- Constant-Current/Constant-Voltage
- Charges Single Cell Li-Ion Batteries Directly from USB Port
- Preset 4.2V Charge Voltage with 1.5% Accuracy
- Automatic Recharge
- two Charge Status Output Pins
- C/10 Charge Termination
- 2.9V Trickle Charge Threshold (TP4056)
- Soft-Start Limits Inrush Current
- Available Radiator in 8-Lead SOP Package, the Radiator need connect GND or impending

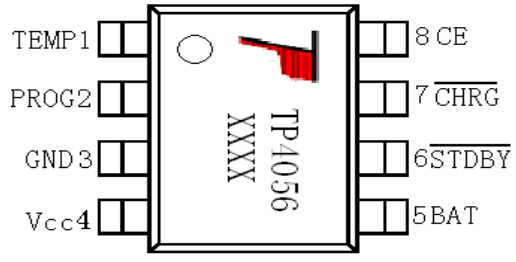
ABSOLUTE MAXIMUM RATINGS

- Input Supply Voltage(V_{CC})：-0.3V～8V
- TEMP：**-0.3V**～10V
- CE：**-0.3V**～10V
- BAT Short-Circuit Duration：Continuous
- BAT Pin Current：1200mA
- PROG Pin Current：1200uA
- Maximum Junction Temperature：145℃
- Operating Ambient Temperature Range：-40℃～85℃
- Lead Temp.(Soldering, 10sec)：260℃

APPLICATIONS

- Cellular Telephones, PDAs, GPS
- Charging Docks and Cradles
- Digital Still Cameras, Portable Devices
- USB Bus-Powered Chargers,Chargers

PACKAGE/ORDER INFORMATION



SOP－8

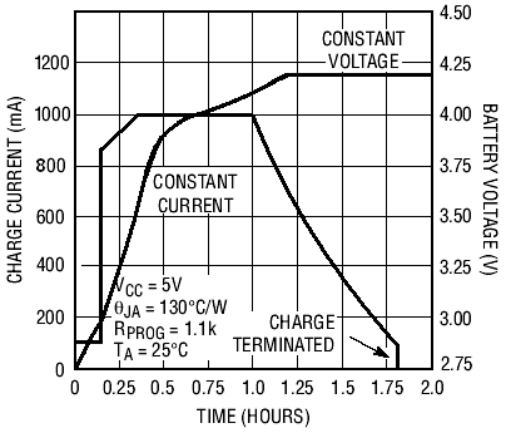
photo

ORDER PART NUMBER

TP4056-42-SOP8-PP

PART MARKING TP4056

Complete Charge Cycle (1000mAh Battery)



TEMP(Pin 1) :Temperature Sense Input Connecting TEMP pin to NTC thermistor's output in Lithium ion battery pack. If TEMP pin's voltage is below 45% or above 80% of supply voltage V_{IN} for more than 0.15S, this means that battery's temperature is too high or too low, charging is suspended. The temperature sense function can be disabled by grounding the TEMP pin.

PROG(Pin 2): Constant Charge Current Setting and Charge Current Monitor Pin charge current is set by connecting a resistor R_{ISET} from this pin to GND. When in precharge mode, the ISET pin's voltage is regulated to 0.2V. When in constant charge current mode, the ISET pin's voltage is regulated to 2V.In all modes during charging, the voltage on ISET pin can be used to measure the charge current as follows:

$$I_{BAT} = \frac{V_{PROG}}{R_{PROG}} \times 1200 \quad (V_{PROG}=1V)$$

GND(Pin3): Ground Terminal

Vcc(Pin 4): Positive Input Supply Voltage V_{IN} is the power supply to the internal circuit. When V_{IN} drops to within 30mv of the BAT pin voltage, TP4056 enters low power sleep mode, dropping BAT pin's current to less than 2uA.

BAT(Pin5): Battery Connection Pin. Connect the positive terminal of the battery to BAT pin. BAT pin draws less than 2uA current in chip disable mode or in sleep mode. BAT pin provides charge current to the battery and provides regulation voltage of 4.2V.

$\overline{STDBY}$(Pin6): Open Drain Charge Status Output When the battery Charge Termination, the $\overline{STDBY}$ pin is pulled low by an internal switch, otherwise $\overline{STDBY}$ pin is in high impedance state.

$\overline{CHRG}$(Pin7): Open Drain Charge Status Output When the battery is being charged, the $\overline{CHRG}$ pin is pulled low by an internal switch, otherwise $\overline{CHRG}$ pin is in high impedance state.

CE(Pin8): Chip Enable Input. A high input will put the device in the normal operating mode. Pulling the CE pin to low level will put the YP4056 into disable mode. The CE pin can be driven by TTL or CMOS logic level.

ELECTRICAL CHARACTERISTICS

The ● denotes specifications which apply over the full operating temperature range, otherwise specifications are at **T_A=25℃， V_{CC}=5V**，unless otherwise noted.

SYMBOL	PARAMETER	CONDITIONS		MIN	TYP	MAX	UNITS
V_{CC}	Input Supply Voltage		●	4.0	5	8.0	V
I_{CC}	Input Supply Current	Charge Mode, R_{PROG} = 1.2k StandbyMode(Charge Terminated) Shutdown Mode (R_{PROG} Not Connected,V_{CC} < V_{BAT}, or V_{CC} < V_{UV})	● ● ●		150 55 55	500 100 100	μA μA μA
V_{FLOAL}	Regulated Output (Float) Voltage	0℃≤T_A≤85℃， I_{BAT}=40mA		4.137	4.2	4.263	V
I_{BAT}	BAT Pin Current Text condition:VBAT=4.0V	R_{PROG} = 2.4k, Current Mode R_{PROG} = 1.2k, Current Mode Standby Mode, V_{BAT} = 4.2V	● ● ●	450 950 0	500 1000 －2.5	550 1050 －6	mA mA μA
I_{TRIKL}	Trickle Charge Current	V_{BAT}<V_{TRIKL}, R_{PROG}=1.2K	●	120	130	140	mA
V_{TRIKL}	Trickle Charge Threshold Voltage	R_{PROG}=1.2K, V_{BAT} Rising		2.8	2.9	3.0	V
V_{TRHYS}	Trickle Charge Hysteresis Voltage	R_{PROG}=1.2K		60	80	100	mV
T_{LIM}	Junction Temperature in Constant Temperature Mode				145		℃

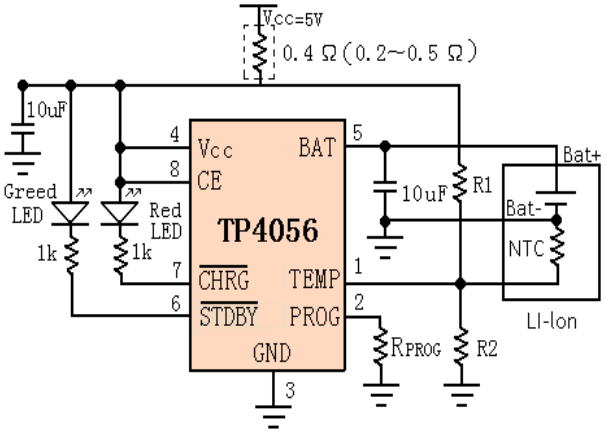
indicator light state

Charge state	Red LED $\overline{CHRG}$	Greed LED $\overline{STDBY}$
charging	bright	extinguish
Charge Termination	extinguish	bright
Vin too low; Temperature of battery too low or too high; no battery	extinguish	extinguish
BAT PIN Connect 10u Capacitance; No battery	Greed LED bright，Red LED Coruscate T=1-4 S	

Rprog Current Setting

R_{PROG} (k)	I_{BAT} (mA)
10	130
5	250
4	300
3	400
2	580
1.66	690
1.5	780
1.33	900
1.2	1000

TYPICAL APPLICATIONS



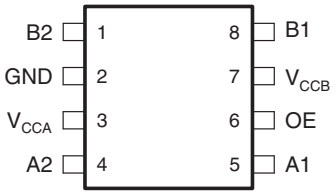
TXS0102
2-BIT BIDIRECTIONAL VOLTAGE-LEVEL TRANSLATOR FOR OPEN-DRAIN APPLICATIONS

SCES640 – JANUARY 2007

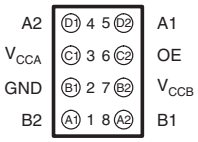
FEATURES

- **Available in the Texas Instruments NanoStar™ and NanoFree™ Packages**
- **1.65 V to 3.6 V on A port and 2.3 V to 5.5 V on B port ($V_{CCA} \leq V_{CCB}$)**
- **V_{CC} Isolation Feature – If Either V_{CC} Input Is at GND, Both Ports Are in the High-Impedance State**
- **I_{off} Supports Partial-Power-Down Mode Operation**
- **Latch-Up Performance Exceeds 100 mA Per JESD 78, Class II**
- **ESD Protection Exceeds JESD 22**
  - **A Port**
    - **2500-V Human-Body Model (A114-B)**
    - **250-V Machine Model (A115-A)**
    - **1500-V Charged-Device Model (C101)**
  - **B Port**
    - **8-kV Human-Body Model (A114-B)**
    - **250-V Machine Model (A115-A)**
    - **1500-V Charged-Device Model (C101)**

DCT OR DCU PACKAGE
(TOP VIEW)



YZP PACKAGE
(BOTTOM VIEW)



DESCRIPTION/ORDERING INFORMATION

This two-bit noninverting translator uses two separate configurable power-supply rails. The A port is designed to track V_{CCA}. V_{CCA} accepts any supply voltage from 1.65 V to 3.6 V. The B port is designed to track V_{CCB}. V_{CCA} must be less than or equal to V_{CCB}. V_{CCB} accepts any supply voltage from 2.3 V to 5.5 V. This allows for low-voltage bidirectional translation between any of the 1.8-V, 2.5-V, 3.3-V, and 5-V voltage nodes.

When the output-enable (OE) input is low, all outputs are placed in the high-impedance state.

To ensure the high-impedance state during power up or power down, OE should be tied to GND through a pulldown resistor; the minimum value of the resistor is determined by the current-sourcing capability of the driver.

ORDERING INFORMATION

T_A	PACKAGE(1)		ORDERABLE PART NUMBER	TOP-SIDE MARKING(2)
–40°C to 85°C	NanoStar™ – WCSP (DSBGA) 0.23-mm Large Bump – YZP	Reel of 3000	TXS0102YZPR	
	SSOP – DCT	Reel of 3000	TXS0102DCTR	NFEZ_ _ _
		Tube of 250	TXS0102DCTT	NFEZ _ _ _
	VSSOP – DCU	Reel of 3000	TXS0102DCUR	_ _ NFE

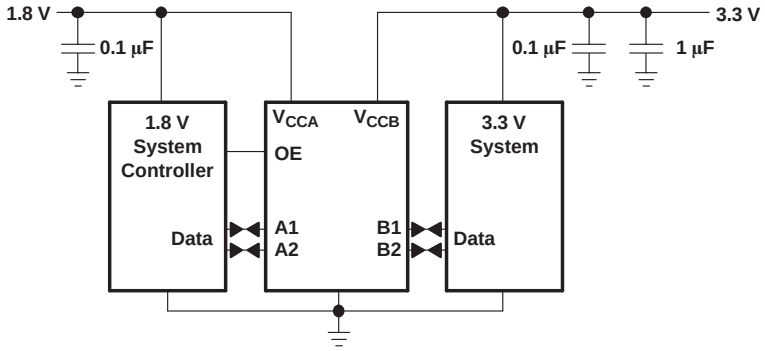
(1) Package drawings, standard packing quantities, thermal data, symbolization, and PCB design guidelines are available at www.ti.com/sc/package.

(2) DCT: The actual top-side marking has three additional characters that designate the year, month, and assembly/test site.
DCU: The actual top-side marking has one additional character that designates the assembly/test site.
YZP: The actual top-side marking has three preceding characters to denote year, month, and sequence code, and one following character to designate the assembly/test site. Pin 1 identifier indicates solder-bump composition (1 = SnPb, • = Pb-free).

PIN DESCRIPTION (DCT AND DCU PACKAGES)

NO.	NAME	FUNCTION
1	B2	Input/output B. Referenced to V_{CCB}.
2	GND	Ground
3	V_{CCA}	A-port supply voltage. 1.65 V ≤ V_{CCA} ≤ 3.6 V and V_{CCA} ≤ V_{CCB}
4	A2	Input/output A. Referenced to V_{CCA}.
5	A1	Input/output A. Referenced to V_{CCA}.
6	OE	3-state output mode enable. Pull OE low to place all outputs in 3-state mode. Referenced to V_{CCA}.
7	V_{CCB}	B-port supply voltage. 2.3 V ≤ V_{CCB} ≤ 5.5 V
8	B1	Input/output B. Referenced to V_{CCB}.

TYPICAL OPERATING CIRCUIT



Absolute Maximum Ratings(1)

over recommended operating free-air temperature range (unless otherwise noted)

			MIN	MAX	UNIT
V_{CCA}	Supply voltage range		–0.5	4.6	V
V_{CCB}	Supply voltage range		–0.5	6.5	V
V_I	Input voltage range(2)	A port	–0.5	4.6	V
		B port	–0.5	6.5	
V_O	Voltage range applied to any output in the high-impedance or power-off state(2)	A port	–0.5	4.6	V
		B port	–0.5	6.5	
V_O	Voltage range applied to any output in the high or low state(2)(3)	A port	–0.5	V_{CCA} + 0.5	V
		B port	–0.5	V_{CCB} + 0.5	
I_{IK}	Input clamp current	V_I < 0		–50	mA
I_{OK}	Output clamp current	V_O < 0		–50	mA
I_O	Continuous output current			±50	mA
	Continuous current through V_{CCA}, V_{CCB}, or GND			±100	mA
θ_{JA}	Package thermal impedance(4)	DCT package		220	°C/W
		DCU package		227	
		YZP package		102	
T_{stg}	Storage temperature range		–65	150	°C

(1) Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under "recommended operating conditions" is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.
(2) The input and output negative-voltage ratings may be exceeded if the input and output current ratings are observed.
(3) The value of V_{CCA} and V_{CCB} are provided in the recommended operating conditions table.
(4) The package thermal impedance is calculated in accordance with JESD 51-7.

Recommended Operating Conditions[1][2]

			V_{CCA}	V_{CCB}	MIN	MAX	UNIT
V_{CCA}	Supply voltage[3]				1.65	3.6	V
V_{CCB}					2.3	5.5	
V_{IH}	High-level input voltage	A-port I/Os	1.65 V to 1.95 V	2.3 V to 5.5 V	V_{CCI} – 0.2	V_{CCI}	V
			2.3 V to 3.6 V		V_{CCI} – 0.4	V_{CCI}	
		B-port I/Os	1.65 V to 3.6 V	2.3 V to 5.5 V	V_{CCI} – 0.4	V_{CCI}	
		OE input			$V_{CCA} \times 0.65$	5.5	
V_{IL}	Low-level input voltage	A-port I/Os	1.65 V to 3.6 V	2.3 V to 5.5 V	0	0.15	V
		B-port I/Os			0	0.15	
		OE input			0	$V_{CCA} \times 0.35$	
$\Delta t/\Delta v$	Input transition rise or fall rate	A-port I/Os, push-pull driving	1.65 V to 3.6 V	2.3 V to 5.5 V		10	ns/V
		B-port I/Os, push-pull driving				10	
		Control input				10	
T_A	Operating free-air temperature				–40	85	°C

(1) V_{CCI} is the supply voltage associated with the input port.
(2) V_{CCO} is the supply voltage associated with the output port.
(3) V_{CCA} must be less than or equal to V_{CCB}, and V_{CCA} must not exceed 3.6 V.

Electrical Characteristics[(1)(2)(3)]

over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER		TEST CONDITIONS	V_{CCA}	V_{CCB}	T_A = 25°C			–40°C to 85°C		UNIT
					MIN	TYP	MAX	MIN	MAX	
V_{OHA}		I_{OH} = –20 µA, $V_{IB} \geq V_{CCB}$ – 0.4 V	1.65 V to 3.6 V	2.3 V to 5.5 V				V_{CCA} × 0.67		V
V_{OLA}		I_{OL} = 1 mA, $V_{IB} \leq$ 0.15 V	1.65 V to 3.6 V	2.3 V to 5.5 V					0.4	V
V_{OHB}		I_{OH} = –20 µA, $V_{IA} \geq V_{CCA}$ – 0.2 V	1.65 V to 3.6 V	2.3 V to 5.5 V				V_{CCB} × 0.67		V
V_{OLB}		I_{OL} = 1 mA, $V_{IA} \leq$ 0.15 V	1.65 V to 3.6 V	2.3 V to 5.5 V					0.4	V
I_I	OE		1.65 V to 5.5 V	1.65 V to 5.5 V			±1		±2	µA
I_{off}	A port		0 V	0 to 5.5 V			±1		±2	µA
	B port		0 to 3.6 V	0 V			±1		±2	µA
I_{OZ}	A or B port		1.65 V to 5.5 V	2.3 V to 5.5 V			±1		±2	µA
I_{CCA}		$V_I = V_O$ = open, I_O = 0	1.65 V to V_{CCB}	2.3 V to 5.5 V					2.4	µA
			3.6 V	0 V					2.2	
			0 V	5.5 V					–1	
I_{CCB}		$V_I = V_O$ = open, I_O = 0	1.65 V to V_{CCB}	2.3 V to 5.5 V					12	µA
			3.6 V	0 V					–1	
			0 V	5.5 V					1	
$I_{CCA} + I_{CCB}$		$V_I = V_{CCI}$ or GND, I_O = 0	1.65 V to V_{CCB}	2.3 V to 5.5 V					14.4	µA
C_I	OE		3.3 V	3.3 V		2.5			3.5	pF
C_{io}	A or B port		3.3 V	3.3 V		10				pF
	A port					5		6		
	B port					6		7.5		

(1) V_{CCI} is the V_{CC} associated with the input port.
(2) V_{CCO} is the V_{CC} associated with the output port.
(3) V_{CCA} must be less than or equal to V_{CCB}, and V_{CCA} must not exceed 3.6 V.

Timing Requirements

over recommended operating free-air temperature range, V_{CCA} = 1.8 V ± 0.15 V (unless otherwise noted)

				V_{CCB} = 2.5 V ± 0.2 V		V_{CC} = 3.3 V ± 0.3 V		V_{CC} = 5 V ± 0.5 V		UNIT
				MIN	MAX	MIN	MAX	MIN	MAX	
	Data rate	Push-pull driving			21		22		24	Mbps
		Open-drain driving			1		1		1	
t_w	Pulse duration	Push-pull driving	Data inputs	47		45		41		ns
		Open-drain driving		500		500		500		

Timing Requirements

over recommended operating free-air temperature range, V_{CCA} = 2.5 V ± 0.2 V (unless otherwise noted)

				V_{CCB} = 2.5 V ± 0.2 V		V_{CC} = 3.3 V ± 0.3 V		V_{CC} = 5 V ± 0.5 V		UNIT
				MIN	MAX	MIN	MAX	MIN	MAX	
	Data rate	Push-pull driving			20		22		24	Mbps
		Open-drain driving			1		1		1	
t_w	Pulse duration	Push-pull driving	Data inputs	50		45		41		ns
		Open-drain driving		500		500		500		

Timing Requirements

over recommended operating free-air temperature range, V_{CCA} = 3 V ± 0.3 V (unless otherwise noted)

				V_{CC} = 3.3 V ± 0.3 V		V_{CC} = 5 V ± 0.5 V		UNIT
				MIN	MAX	MIN	MAX	
Data rate		Push-pull driving			23		24	Mbps
		Open-drain driving			1		1	
t_w	Pulse duration	Push-pull driving	Data inputs	43		41		ns
		Open-drain driving		500		500		

Switching Characteristics

over recommended operating free-air temperature range, V_{CCA} = 1.8 V ± 0.15 V (unless otherwise noted)

PARAMETER	FROM (INPUT)	TO (OUTPUT)	TEST CONDITIONS	V_{CCB} = 2.5 V ± 0.2 V		V_{CCB} = 3.3 V ± 0.3 V		V_{CCB} = 5 V ± 0.5 V		UNIT
				MIN	MAX	MIN	MAX	MIN	MAX	
t_{PHL}	A	B	Push-pull driving		5.3		5.4		6.8	ns
			Open-drain driving	2.3	8.8	2.4	9.6	2.6	10	
t_{PLH}			Push-pull driving		6.8		7.1		7.5	
			Open-drain driving	45	260	36	208	27	198	
t_{PHL}	B	A	Push-pull driving		4.4		4.5		4.7	ns
			Open-drain driving	1.9	5.3	1.1	4.4	1.2	4	
t_{PLH}			Push-pull driving		5.3		4.5		0.5	
			Open-drain driving	45	175	36	140	27	102	
t_{en}	OE	A or B			200		200		200	ns
t_{dis}	OE	A or B			50		40		35	ns
t_{rA}	A-port rise time		Push-pull driving	3.2	9.5	2.3	9.3	2	7.6	ns
			Open-drain driving	38	165	30	132	22	95	
t_{rB}	B-port rise time		Push-pull driving	4	10.8	2.7	9.1	2.7	7.6	ns
			Open-drain driving	34	145	23	106	10	58	
t_{fA}	A-port fall time		Push-pull driving	2	5.9	1.9	6	1.7	13.3	ns
			Open-drain driving	4.4	6.9	4.3	6.4	4.2	6.1	
t_{fB}	B-port fall time		Push-pull driving	2.9	13.8	2.8	16.2	2.8	16.2	
			Open-drain driving	6.9	13.8	7.5	16.2	7	16.2	
$t_{SK(O)}$	Channel-to-channel skew				0.7		0.7		0.7	ns
Max data rate			Push-pull driving	21		22		24		Mbps
			Open-drain driving	1		1		1		

Switching Characteristics

over recommended operating free-air temperature range, V_{CCA} = 2.5 V ± 0.2 V (unless otherwise noted)

PARAMETER	FROM (INPUT)	TO (OUTPUT)	TEST CONDITIONS	V_{CCB} = 2.5 V ± 0.2 V		V_{CCB} = 3.3 V ± 0.3 V		V_{CCB} = 5 V ± 0.5 V		UNIT
				MIN	MAX	MIN	MAX	MIN	MAX	
t_{PHL}	A	B	Push-pull driving		3.2		3.7		3.8	ns
			Open-drain driving	1.7	6.3	2	6	2.1	5.8	
t_{PLH}			Push-pull driving		3.5		4.1		4.4	
			Open-drain driving	43	250	36	206	27	190	
t_{PHL}	B	A	Push-pull driving		3		3.6		4.3	ns
			Open-drain driving	1.8	4.7	2.6	4.2	1.2	4	
t_{PLH}			Push-pull driving		2.5		1.6		1	
			Open-drain driving	44	170	37	140	27	103	
t_{en}	OE	A or B			200		200		200	ns
t_{dis}	OE	A or B			50		40		35	ns
t_{rA}	A-port rise time		Push-pull driving	2.8	7.4	2.6	6.6	1.8	5.6	ns
			Open-drain driving	34	149	28	121	24	89	
t_{rB}	B-port rise time		Push-pull driving	3.2	8.3	2.9	7.2	2.4	6.1	ns
			Open-drain driving	35	151	24	112	12	64	
t_{fA}	A-port fall time		Push-pull driving	1.9	5.7	1.9	5.5	1.8	5.3	ns
			Open-drain driving	4.4	6.9	4.3	6.2	4.2	5.8	
t_{fB}	B-port fall time		Push-pull driving	2.2	7.8	2.4	6.7	2.6	6.6	ns
			Open-drain driving	5.1	8.8	5.4	9.4	5.4	10.4	
$t_{SK(O)}$	Channel-to-channel skew				0.7		0.7		0.7	ns
Max data rate			Push-pull driving	20		22		24		Mbps
			Open-drain driving	1		1		1		

Switching Characteristics

over recommended operating free-air temperature range, V_{CCA} = 3.3 V ± 0.3 V (unless otherwise noted)

PARAMETER	FROM (INPUT)	TO (OUTPUT)	TEST CONDITIONS	V_{CCB} = 3.3 V ± 0.3 V		V_{CCB} = 5 V ± 0.5 V		UNIT
				MIN	MAX	MIN	MAX	
t_{PHL}	A	B	Push-pull driving		2.4		3.1	ns
			Open-drain driving	1.3	4.2	1.4	4.6	
t_{PLH}			Push-pull driving		4.2		4.4	
			Open-drain driving	36	204	28	165	
t_{PHL}	B	A	Push-pull driving		2.5		3.3	ns
			Open-drain driving	1	124	1	97	
t_{PLH}			Push-pull driving		2.5		2.6	
			Open-drain driving	3	139	3	105	
t_{en}	OE	A or B			200		200	ns
t_{dis}	OE	A or B			40		35	ns
t_{rA}	A-port rise time		Push-pull driving	2.3	5.6	1.9	4.8	ns
			Open-drain driving	25	116	19	85	
t_{rB}	B-port rise time		Push-pull driving	2.5	6.4	2.1	7.4	ns
			Open-drain driving	26	116	14	72	
t_{fA}	A-port fall time		Push-pull driving	2	5.4	1.9	5	ns
			Open-drain driving	4.3	6.1	4.2	5.7	
t_{fB}	B-port fall time		Push-pull driving	2.3	7.4	2.4	7.6	ns
			Open-drain driving	5	7.6	4.8	8.3	
$t_{SK(O)}$	Channel-to-channel skew				0.7		0.7	ns
Max data rate			Push-pull driving	23		24		Mbps
			Open-drain driving	1		1		

PRINCIPLES OF OPERATION

Applications

The TXS0102 can be used in level-translation applications for interfacing devices or systems operating at different interface voltages with one another. The TXS0102 is ideal for use in applications where an open-drain driver is connected to the data I/Os. The TXS0102 can also be used in applications where a push-pull driver is connected to the data I/Os, but the TXB0102 might be a better option for such push-pull applications.

Architecture

The TXS0102 architecture (see Figure 1) does not require a direction-control signal to control the direction of data flow from A to B or from B to A.

PRINCIPLES OF OPERATION (continued)

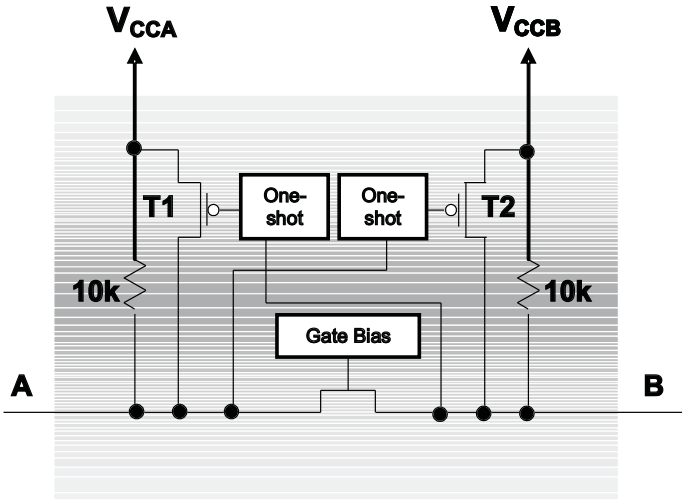


Figure 1. Architecture of a TXS01xx Cell

Each A-port I/O has an internal 10-kΩ pullup resistor to V_{CCA}, and each B-port I/O has an internal 10-kΩ pullup resistor to V_{CCB}. The output one-shots detect rising edges on the A or B ports. During a rising edge, the one-shot turns on the PMOS transistors (T1,T2) for a short duration, which speeds up the low-to-high transition.

Input Driver Requirements

The fall time (t_{fA}, t_{fB}) of a signal depends on the output impedance of the external device driving the data I/Os of the TXS0102. Similarly, the t_{PHL} and max data rates also depend on the output impedance of the external driver. The values for t_{fA}, t_{fB}, t_{PHL}, and maximum data rates in the data sheet assume that the output impedance of the external driver is less than 50 Ω.

Power Up

During operation, ensure that $V_{CCA} \leq V_{CCB}$ at all times. During power-up sequencing, $V_{CCA} \geq V_{CCB}$ does not damage the device, so any power supply can be ramped up first.

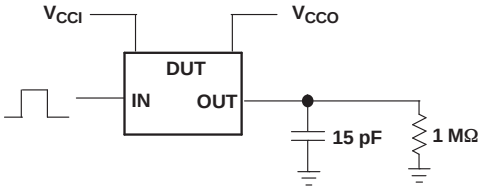
Enable and Disable

The TXS0102 has an OE input that is used to disable the device by setting OE low, which places all I/Os in the Hi-Z state. The disable time (t_{dis}) indicates the delay between the time when OE goes low and when the outputs actually get disabled (Hi-Z). The enable time (t_{en}) indicates the amount of time the user must allow for the one-shot circuitry to become operational after OE is taken high.

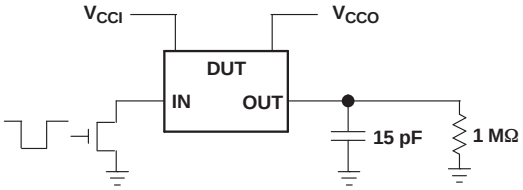
Pullup or Pulldown Resistors on I/O Lines

Each A-port I/O has an internal 10-kΩ pullup resistor to V_{CCA}, and each B-port I/O has an internal 10-kΩ pullup resistor to V_{CCB}. If a smaller value of pullup resistor is required, an external resistor must be added from the I/O to V_{CCA} or V_{CCB} (in parallel with the internal 10-kΩ resistors).

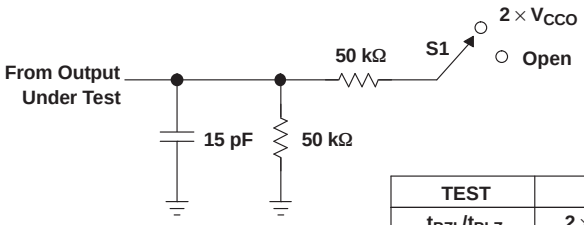
PARAMETER MEASUREMENT INFORMATION



DATA RATE, PULSE DURATION, PROPAGATION DELAY, OUTPUT RISE AND FALL TIME MEASUREMENT USING A PUSH-PULL DRIVER

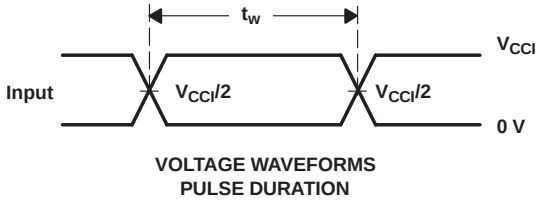


DATA RATE, PULSE DURATION, PROPAGATION DELAY, OUTPUT RISE AND FALL TIME MEASUREMENT USING AN OPEN-DRAIN DRIVER

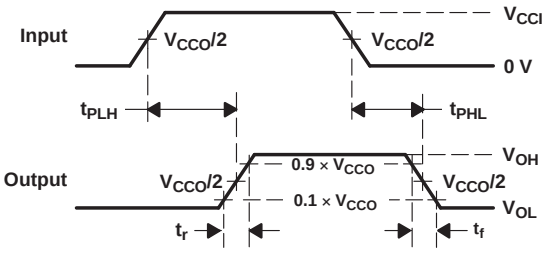


LOAD CIRCUIT FOR ENABLE/DISABLE TIME MEASUREMENT

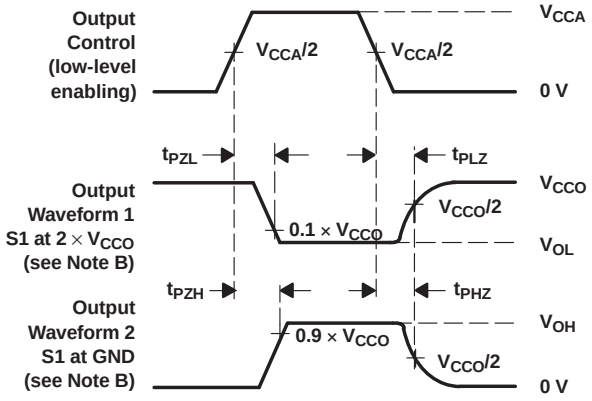
TEST	S1
t_{PZL}/t_{PLZ} t_{PHZ}/t_{PZH}	2 × V_{CCO} Open



VOLTAGE WAVEFORMS PULSE DURATION



VOLTAGE WAVEFORMS PROPAGATION DELAY TIMES



VOLTAGE WAVEFORMS ENABLE AND DISABLE TIMES

NOTES:
A. C_L includes probe and jig capacitance.
B. Waveform 1 is for an output with internal conditions such that the output is low, except when disabled by the output control. Waveform 2 is for an output with internal conditions such that the output is high, except when disabled by the output control.
C. All input pulses are supplied by generators having the following characteristics: PRR ≤ 10 MHz, Z_O = 50 Ω, dv/dt ≥ 1 V/ns.
D. The outputs are measured one at a time, with one transition per measurement.
E. t_{PLZ} and t_{PHZ} are the same as t_{dis}.
F. t_{PZL} and t_{PZH} are the same as t_{en}.
G. t_{PLH} and t_{PHL} are the same as t_{pd}.
H. V_{CCI} is the V_{CC} associated with the input port.
I. V_{CCO} is the V_{CC} associated with the output port.
J. All parameters and waveforms are not applicable to all devices.

Figure 2. Load Circuit and Voltage Waveforms

PACKAGING INFORMATION

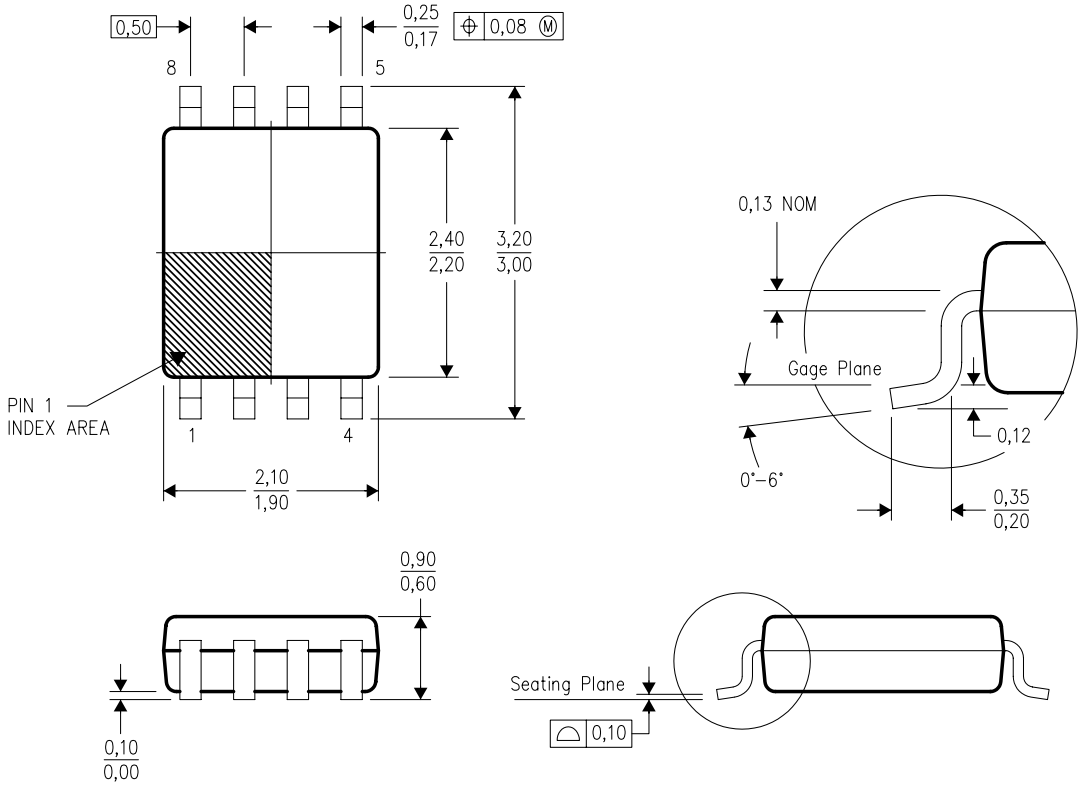
Orderable Device	Status [1]	Package Type	Package Drawing	Pins	Package Qty	Eco Plan [2]	Lead/Ball Finish	MSL Peak Temp [3]
TXS0102DCUR	ACTIVE	US8	DCU	8	3000	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-1-260C-UNLIM

[1] The marketing status values are defined as follows:
ACTIVE: Product device recommended for new designs.
LIFEBUY: TI has announced that the device will be discontinued, and a lifetime-buy period is in effect.
NRND: Not recommended for new designs. Device is in production to support existing customers, but TI does not recommend using this part in a new design.
PREVIEW: Device has been announced but is not in production. Samples may or may not be available.
OBSOLETE: TI has discontinued the production of the device.

[2] Eco Plan - The planned eco-friendly classification: Pb-Free (RoHS), Pb-Free (RoHS Exempt), or Green (RoHS & no Sb/Br) - please check http://www.ti.com/productcontent for the latest availability information and additional product content details.
TBD: The Pb-Free/Green conversion plan has not been defined.
Pb-Free (RoHS): TI's terms "Lead-Free" or "Pb-Free" mean semiconductor products that are compatible with the current RoHS requirements for all 6 substances, including the requirement that lead not exceed 0.1% by weight in homogeneous materials. Where designed to be soldered at high temperatures, TI Pb-Free products are suitable for use in specified lead-free processes.
Pb-Free (RoHS Exempt): This component has a RoHS exemption for either 1) lead-based flip-chip solder bumps used between the die and package, or 2) lead-based die adhesive used between the die and leadframe. The component is otherwise considered Pb-Free (RoHS compatible) as defined above.
Green (RoHS & no Sb/Br): TI defines "Green" to mean Pb-Free (RoHS compatible), and free of Bromine (Br) and Antimony (Sb) based flame retardants (Br or Sb do not exceed 0.1% by weight in homogeneous material)

[3] MSL, Peak Temp. -- The Moisture Sensitivity Level rating according to the JEDEC industry standard classifications, and peak solder temperature.

DCU (R–PDSO–G8) PLASTIC SMALL–OUTLINE PACKAGE (DIE DOWN)



4200503/F 07/05

NOTES:
A. All linear dimensions are in millimeters.
B. This drawing is subject to change without notice.
C. Body dimensions do not include mold flash or protrusion. Mold flash and protrusion shall not exceed 0.15 per side.
D. Falls within JEDEC MO–187 variation CA.

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
Low Power Wireless	www.ti.com/lpw	Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments
Post Office Box 655303 Dallas, Texas 75265