UNIVERSIDAD DE LA LAGUNA

ESCUELA SUPERIOR DE INGENIERÍA Y TECNOLOGÍA

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

TRABAJO FINAL DE GRADO

DISEÑO DE UN SISTEMA PARA LA EXTRACCIÓN DE DATOS EN EL SALTO CON PÉRTIGA

Autor:

ADRIÁN PÉREZ SUÁREZ

Tutor:

OSWALDO BERNABÉ GONZÁLEZ HERNÁNDEZ

Septiembre de 2022

ABSTRACT

Pole vaulting is one of the most complex sports disciplines, with a high technical level due to the use of an elastic device, the pole vault, which catapults the athlete to reach the highest possible height and clear a bar. Coaches' corrections to pole vaulters in the execution of their jumps are almost entirely limited to the coach's observation *in situ* or through the use of audiovisual media, providing technical data and even sensations if the coach had been a pole vaulter and transmitted his own perceptions, and in such a way, making corrections to the vaulter in an empirical way. Therefore, a lot of information about what happens during the jump is ignored.

Analysis of inertial data can provide a greater amount of information about what is happening at each stage of the jump. If enough data is collected, patterns could be obtained that indicate variables for improvement that could lead to an increase in the athlete's performance in terms of personal bests.

With this final degree project, a pole vault data measurement system has been designed, manufactured and tested using a device that incorporates an ESP32 microprocessor, an MPU6050 sensor module, a lithium polymer battery, a general connection board and an encapsulation that protects and attaches the device to the back of the pole vault.

The result has been successful, resulting in a device capable of sending real-time angular and inertial measurement data towards a mobile device or computer via Bluetooth connection.

RESUMEN

El salto con pértiga es una de las disciplinas deportivas más complejas, con un alto nivel técnico debido al uso de un artefacto elástico, la pértiga, que catapulta al atleta para alcanzar la mayor altura posible y franquear un listón. Las correcciones de los entrenadores a los pertiguistas en la ejecución de sus saltos se limitan casi en su totalidad a la observancia del entrenador *in situ* o mediante utilización de medios audiovisuales, aportando datos técnicos e incluso de sensaciones si el entrenador hubiera sido pertiguista y transmitiera sus propias percepciones, y de tal manera, realizar unas correcciones al saltador de forma empírica. Por tanto, se obvia mucha información de lo que sucede durante el salto.

El análisis de datos inerciales puede proporcionar una mayor cantidad de información sobre lo que está sucediendo en cada etapa del salto. Si se recopilase la suficiente cantidad de datos, se podrían llegar a obtener patrones que indiquen variables de mejora que puedan conducir a un aumento en el rendimiento del atleta en términos de marcas personales.

Con este trabajo final de grado se ha diseñado, fabricado y probado un sistema de medida de datos de salto con pértiga mediante un dispositivo que incorpora un microprocesador ESP32, un módulo sensor MPU6050, una batería de polímero de litio, una placa general de conexión y un encapsulado que protege y acopla el dispositivo a la parte trasera del salto con pértiga.

El resultado ha sido exitoso, obteniéndose un dispositivo capaz de enviar datos de medidas angulares e inerciales en tiempo real a un dispositivo móvil u ordenador mediante conexión Bluetooth.

Índice general

1. Introducción 8
1.1. Objeto y enfoque del trabajo 8
1.2. El atletismo 8
1.3. Historia de los récords mundiales 10
1.4. Atletismo e ingeniería 11
1.5. Estructura de la memoria 12
2. Descripción del sistema 13
2.1. Descripción General 13
2.2. Flujo de Ejecución 13
2.3. Normativa aplicable 14
3. Selección del hardware y software 16
3.1. Microcontrolador ESP32 16
3.2. Unidad de medición inercial MPU6050 18
3.3. Batería de polímero de litio 19
3.4. Placa de Circuito Impreso 21
3.5. Impresión 3D 21
3.6. Software utilizado 22
4. Descripción del hardware y software utilizado 25
4.1. Hardware utilizado 25
4.1.1. Placa PCB 25
4.1.2. Placa ESP32 Dev Kit 26
4.1.3. Módulo de sensor MPU6050 27
4.1.3.1. Características del giroscopio 28
4.1.3.2. Características del acelerómetro 29
4.1.3.3. Características adicionales 29
4.2. Descripción del software 30
4.2.1. Arduino 30
4.2.2. KiCAD 31
4.2.3. Fusion360 31
5. Diseño del sistema y resultados obtenidos 33
5.1. Encapsulado 3D 33
5.2. Código del microcontrolador ESP32 35
5.3. Recepción de datos por serial *Bluetooth* 41

5.4. Análisis de los datos en *Python* 43
5.4.1. Discusión de los resultados obtenidos 47
6. Presupuesto 50
6.1. Materiales 50
6.2. Mano de obra 50
6.1. Coste de ejecución material 51
6.1. Gastos generales 51
6.2. Beneficio industrial 51
6.3. Impuestos 51
6.4. Coste final del proyecto 52
7. Conclusión y líneas abiertas 53
8. Bibliografía 55
ANEXOS 57
ANEXO I 58
ANEXO II 66
ANEXO III 69
ANEXO VI 72

Índice de figuras

Figura 1. Ceremonia de apertura de los primeros Juegos Olímpicos Modernos en Atenas 1896 [1]. *8*
Figura 2. Marc Wright during the pole vault competition at the 1912 Summer Olympics [15] *10*
Figura 3. Evolución histórica de las marcas en el salto con pértiga [2]. *11*
Figura 4. Diagrama general del sistema. *13*
Figura 5. Diagrama de flujo de ejecución. *14*
Figura 6. Sensor MPU6050 con 6 ejes de libertad. *18*
Figura 7. Batería de LiPo 250 mAh, 3,7 V. *21*
Figura 8. Escala de dureza D para materiales. *22*
Figura 9. Interfaz de búsqueda de dispositivos BLE (Bluetooth Low Energy). *23*
Figura 10. Terminal de comunicación de la aplicación. *23*
Figura 11. Captura del dispositivo terminado. *25*
Figura 12. Anverso de la placa PCB de conexionado. *25*
Figura 13. Dev. Kit NodeMCU ESP32. ..... *26*
Figura 14. Módulo IMU MPU6050. ..... *28*
Figura 15. Logotipo de la compañía Arduino. *30*
Figura 16. Logotipo del software KiCAD. *31*
Figura 17. Logotipo del Software Autodesk Fusion 360. ..... *32*
Figura 18. Modelo 3D del encapsulado *33*
Figura 19. Vista de planta del diseño inicial del encapsulado. *34*
Figura 20. Diseño del logotipo de la universidad en el encapsulado. *34*
Figura 21. Alzado del modelo 3D y su sección transversal. *35*
Figura 22. Título y autoría del código de ejemplo. *35*
Figura 23. Declaración de copyright *36*
Figura 24. Definición de variables. *37*
Figura 25. Librerías BLE de Arduino. *37*
Figura 26. Información sobre la conexión del Bluetooth. *37*
Figura 27. Definición de la función inicio de Bluetooth. *38*
Figura 28. Código ESP32 - Inicio del reloj y la comunicación serial. *38*
Figura 29. Código ESP32 – Inicio de dispositivo, verificación de conexión y puesta en espera. *39*
Figura 30. Mensaje de inicio de transmisión de datos. *39*
Figura 31. Mensaje de fallo en la transmisión de datos. *39*
Figura 32. Fin declaración función de Setup. *39*
Figura 33. Declaración del bucle de repetición infinita. *40*
Figura 34. Condiciones en función de la conexión. *41*
Figura 35. Emparejamiento del dispositivo con el teléfono móvil. *42*
Figura 36. Terminal Serial Bluetooth, conexión. *42*
Figura 37. Exportar datos a texto plano. *43*
Figura 38. Código de visualizado de datos en Python. *44*
Figura 39. Librerías Python. *44*
Figura 40. Preparación de datos del CSV mediante librería pandas. *45*
Figura 41. Creación de arrays con los datos. *46*
Figura 42. Configuración de la gráfica. *46*
Figura 43. Gráfica de datos de pitch sin ajuste de ángulo. *46*
Figura 44. Ajuste de los datos angulares de pitch. *47*
Figura 45. Gráfica de datos de pitch con ajuste de ángulos. *47*
Figura 46. Fases del salto con pértiga (figura modificada de [12]). *48*
Figura 47. Armand Duplantis en fase de carrera [13]. *49*
Figura 48. Valores de angulos YPR de un salto con pértiga obtenidos con el dispositivo. *67*
Figura 49. Gráfica de valores de angulo pitch de un salto con pértiga medido con el dispositivo. *67*

Figura 50. Gráfica de valores de aceleración en el eje X (transversal al pasillo) de un salto con pértiga medido con el dispositivo. *67*
Figura 51. Gráfica de valores de aceleración en el eje Y (longitudinal al pasillo) de un salto con pértiga medido con el dispositivo. *67*
Figura 52. Gráfica de valores de aceleración en el eje Z (vertical) de un salto con pértiga medido con el dispositivo. *68*

Índice de tablas

Tabla 1. Características principales de las distintas placas de microcontroladores elegibles. *17*
Tabla 2. Coste de materiales. *50*
Tabla 3. Mano de obra *50*
Tabla 4. Coste de ejecución material. *51*
Tabla 5. Gastos generales. *51*
Tabla 6. Beneficio industrial. *51*
Tabla 7. Impuestos IGIC. *52*
Tabla 8. Coste final del proyecto. *52*

Prefacio

Este proyecto final de carrera ha supuesto para mí un nexo entre dos de mis pasiones, el atletismo y la ingeniería. Dos mundos aparentemente muy diferentes, pero que con este proyecto he comprobado que no distan demasiado.

Dos mundos que avanzan de la mano, como podréis comprobar a lo largo del documento. Y es que, en general, el deporte es ingeniería. Buscamos siempre la manera de obtener el máximo rendimiento del cuerpo humano, y esto se consigue analizando y mejorando aquellas flaquezas para transformarlas en fortalezas.

Dos mundos, que, en mi caso, me han aportado la mayoría de los valores que me forman como persona hoy en día. Y es que ambos requieren de mucho sacrificio y paciencia para lograr los objetivos planteados. La ingeniería, por un lado, no permite acomodarse, en cuanto a conocimiento y desarrollo. Se trata de un sector en constante movimiento en el que se lucha cada día por llevar a cabo ideas que nunca antes se consiguieron. Esto implica ser a menudo obligado a salir de tu zona de confort y enfrentarte a proyectos que pueden no salir bien. A su vez, el atletismo requiere de mucho más sacrificio que las recompensas obtenidas en forma de resultados y medallas. Las lesiones muchas veces acaban con carreras deportivas, llegando incluso a generar grandes frustraciones y depresiones.

Ambos caminos son ásperos, pero lo que está claro es que ambas partes aportan a quienes las componen, una satisfacción y realización personal que hace que merezca la pena todo el camino.

1. Introducción

1.1. Objeto y enfoque del trabajo

El objetivo de este proyecto de fin de grado es realizar el diseño de un dispositivo inalámbrico acoplable a la pértiga cuya finalidad sea la de recoger y enviar datos de mediciones inerciales mediante conexión *Bluetooth* a un dispositivo para su posterior procesado.

El salto con pértiga es una de las disciplinas deportivas más complejas técnicamente que existen, por lo que muchas veces llega a ser confuso entre técnicos de la prueba reconocer el modelo de salto idóneo. De hecho, los últimos grandes atletas de la disciplina han provocado, con sus técnicas, que esta idea sobre el modelo del salto perfecto haya sido puesta aún más en duda, pues se alejan bastante del que se creía como el "salto perfecto" desarrollado por los rusos.

El uso de unidades de medida inerciales puede favorecer la búsqueda de patrones que reflejen potenciales variables de mejora del rendimiento del salto, ayudando así a los técnicos y atletas a encontrar las claves para superar sus marcas.

1.2. El atletismo

El atletismo tiene una larga historia, es una de las disciplinas deportivas más antiguas del mundo, e incluye varias actividades como correr, saltar, lanzar y otras pruebas compuestas (pruebas combinadas). Muchos de ellos se han practicado por separado con fines deportivos o prácticos desde los albores de la humanidad.

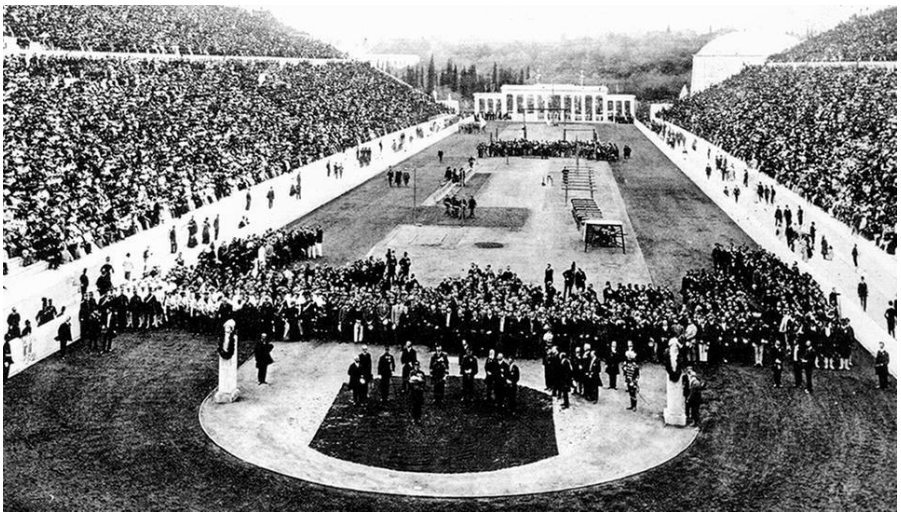


Figura 1. Ceremonia de apertura de los primeros Juegos Olímpicos Modernos en Atenas 1896 [1].

La primera mención del atletismo como deporte se remonta al año 776 a. C., en los juegos desarrollados en Olimpia por parte de los atletas griegos antiguos. En ese momento, estaba limitado a una carrera a pie de 197,27 metros, conocida como *Stadion*, la cual recibe este nombre por el edificio en el que tenía lugar.

La cultura romana también practicaba el ejercicio de una forma similar a la griega, como parte de la herencia etrusca. Introdujeron nuevas prácticas al deporte. Por ejemplo, se agregaron el salto con pértiga, el lanzamiento de martillo y el campo a través.

Muchos de ellos continuaron practicándose a lo largo de la Edad Media, según formas específicas en cada uno de los reinos cristianos de Europa, particularmente Inglaterra y Dinamarca. Algunas disciplinas nuevas tenían un componente militar muy útil, como el tiro con arco.

Una de las carreras más antiguas de la Europa medieval se celebró en Roma a mediados del siglo XV. Allí se recreaban los juegos griegos, y los atletas tenían que competir completamente desnudos, como en la antigüedad.

En 1825 se celebró la primera competición de atletismo moderno cerca de la capital inglesa. Hacia finales de siglo, la visión británica del deporte como elemento social se extendió por toda Europa.

Así comenzó la fusión de las federaciones de atletismo de Francia, Estados Unidos, Bélgica y Alemania. El primer campeonato oficial se celebró en este último país en 1891. Este surgimiento organizado del atletismo en Europa fue clave para la reactivación de los Juegos Olímpicos.

La práctica profesional del deporte se ha confinado casi exclusivamente a Europa Occidental y los Estados Unidos desde principios del siglo XX, pero en la década de 1930 comenzaron a competir los atletas afroamericanos y otros europeos coloniales.

Después de la Segunda Guerra Mundial, los países comunistas del Bloque del Este participaron activamente en la competencia, reivindicando su existencia y poder. Los países del Caribe se incorporaron alrededor de 1970 y los países africanos en 1980. Porque la realidad global del mundo se ha vuelto más patente con este deporte. El atletismo consiste en un ejercicio realizado en condiciones controladas en las que los atletas pueden ser evaluados en una variedad de criterios físicos y mentales. Las pruebas de pista más comunes son:

- Marcha atlética: Fue inventado en Inglaterra y consiste en una modalidad que requiere al menos un pie en contacto constante con el suelo para que no implique correr sino caminar entre 20 y 50 km.
- Saltos: Como sugiere el nombre, los atletas en esta serie de pruebas saltan utilizando una pértiga larga y flexible (salto con pértiga), saltan al final de un *sprint* (salto de longitud) o saltan una barrera (salto de altura).

- Lanzamientos: Implica lanzar un artefacto lo más lejos posible. Existen cuatro tipos de lanzamientos, que son jabalina, martillo, peso y disco.
- Decatlón: En esta categoría, los atletas se someten a diez pruebas diferentes en sucesión. Estas pruebas se realizan de forma continua y sistemática durante dos días. La versión femenina, que consta de siete pruebas en lugar de diez, se conoce como heptatlón.

1.3. Historia de los récords mundiales

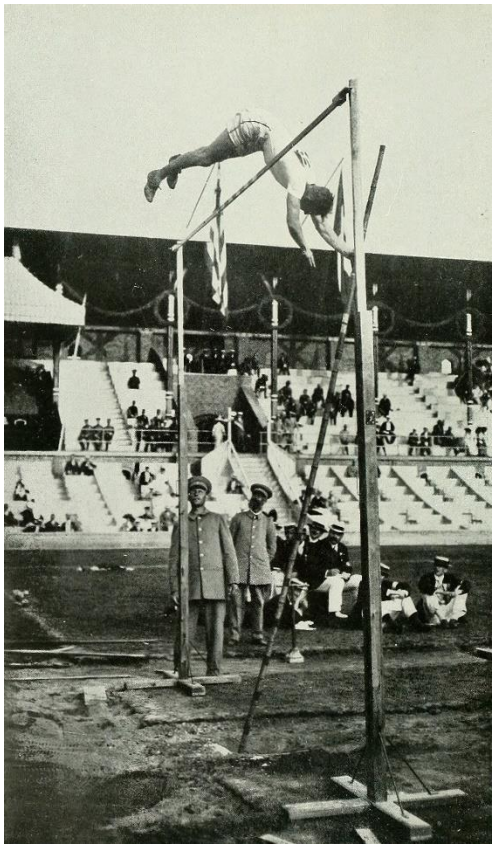


Figura 2.
Marc Wright during the pole vault competition at the 1912 Summer Olympics [15]

El primer récord mundial masculino de salto con pértiga (atletismo) fue reconocido por la IAAF[1] en 1912, y fue del atleta americano Marc Wright con una altura de 4,02 m.

Uno de los saltadores más destacados en toda la historia del atletismo es Serguéi Bubka, atleta ucraniano, quien batió el récord del mundo un total de 35 veces. Es el hombre que más veces ha batido el récord del mundo. En sus años de carrera elevó el récord desde los 5,83 m hasta los 6,15 m, 32 cm de diferencia que implicaron elevar un 5,5% la plusmarca mundial.

Tras la astronómica mejora de Bubka, el récord estuvo inalterado durante más de 20 años, y no fue hasta 2017 que el francés Renaud Lavillenie mejoró por 1 cm esta marca, estableciendo el récord en 6,16 m en la misma reunión y en el mismo estadio.

El hombre que posee actualmente el récord es Armand Duplantis, de nacionalidad sueca, nacido en 1999 en EE. UU., donde ha pasado la mayor parte de su vida. Con un padre ex pertiguista, lleva desde los 3 años en las pistas de atletismo practicando la modalidad. Con apenas 19 años había conseguido el oro en el campeonato de Europa. Con 20 años había conseguido el oro en el campeonato del mundo. En 2020, con apenas 20 años, había superado el récord de Renaud y actualmente ha batido el récord del mundo cinco veces, siendo la última en julio de 2022 con una marca de 6,21 m.

[1]*International Association of Athletics Federations* (Asociación Internacional de Federaciones de Atletismo)

1.4. Atletismo e ingeniería

Para los primeros registros de récords del mundo las pértigas eran muy diferentes, los materiales eran rígidos y no permitían la flexión de esta. Es por ello por lo que los atletas se veían muy limitados. Posteriormente, se empiezan a ver pértigas de bambú que permitían cierta flexión. Pero no fue hasta 1960 que se empiezan a ver las primeras pértigas de fibra de vidrio (material con el que se fabrican actualmente las pértigas junto con la fibra de carbono). A partir de este momento la marca máxima establecida se ve incrementada de forma drástica.

Figura 3. Evolución histórica de las marcas en el salto con pértiga [2].

En el gráfico de la Figura 3 se observa el gradiente de la curva de mejores marcas al aparecer las pértigas de fibra de vidrio. Se observa que en los 20 años anteriores el progreso de la marca se había estancado, mientras que, en un breve intervalo de tiempo, con pértigas de fibra de vidrio, la mejora es notable.

La sustitución de la pértiga rígida por la pértiga flexible implica un cambio en el modelo físico del salto. La nueva pértiga de fibra pone un muelle donde antes había una palanca. Para los atletas, esta mejora provoca un empujón a sus mejores marcas, pero para los técnicos de la prueba, tanto entrenadores como científicos, el cambio supuso una complicación del modelo y, por lo tanto, de la técnica practicada en la modalidad. Se suele reconocer que el salto con pértiga es una de las modalidades deportivas más complejas técnicamente que existen. Y esto no solo afecta al atleta, sino a los técnicos que hay detrás pensando cómo planificar un entrenamiento adecuado para potenciar al máximo las variables que ellos mismos han estipulado como las más influyentes en la mejora del salto.

Aquí radica el porqué de este trabajo final de grado. Hay muchas variables que afectan a dicha técnica, y la incorporación de sensores puede proveer a la comunidad de una herramienta muy útil para obtener datos de los que se puedan sacar patrones y, a partir de ellos, sacar conclusiones para mejorar el rendimiento.

Como conclusión llegamos a la idea de que la tecnología siempre ha propiciado una mejora de rendimiento en el deporte. Y esto es algo general; la investigación y el desarrollo han estado presentes en cualquier modalidad deportiva. Lo vimos con la creación de zapatillas con clavos para traccionar mejor o con la introducción de las pistas de material sintético (tartán) en el atletismo, con el reciente VAR (*Video Assistant Referee*) en el fútbol, que permite obtener imágenes en tiempo real para tomar decisiones rápidas sobre las jugadas ejecutadas por los deportistas o, uno de los ejemplos más notables, la Fórmula 1, en la cual, con frecuencia, parece que se acerca más a una competición de ingeniería que deportiva en sí. Es por esto por lo que se dice que deporte e ingeniería van siempre de la mano.

1.5. Estructura de la memoria

Tras esta introducción, en el **capítulo 2** se realizará una descripción del sistema que engloba flujo de ejecución y normativa aplicable. En el **capítulo 3** se analiza la selección del hardware y software para la realización del diseño planteado para la implementación del sistema.

Tras el análisis de la selección del hardware y el software utilizado, se realiza una descripción más a fonde de cada uno de los elementos del hardware y software en el **capítulo 4**. En el **capítulo 5** se expone el diseño del sistema implementado y los resultados obtenidos a partir del código con que se ha programado el microcontrolador, de la recepción serial del dispositivo en un terminal móvil y un análisis de los datos obtenidos hecho en *Python*. Además, se entra en una discusión sobre los resultados obtenidos. El **capítulo 6** está reservado para el presupuesto. El **capítulo 7** presenta las principales conclusiones de este trabajo, así como las líneas abiertas a seguir, mientras que en el **8** encontramos la bibliografía consultada.

Finalmente se han anexado el código completo del microcontrolador, los planos de la PCB y su esquemático, así como medidas obtenidas por el dispositivo.

2. Descripción del sistema

2.1. Descripción General

El sistema consiste en un conjunto de componentes conectados mediante una placa de conexionado conjunta y a su vez envueltos por un encapsulado. El bloque central es el microcontrolador, al que están conectados directamente todos los restantes componentes. La conexión con la batería sería bidireccional puesto que esta se carga a través de la alimentación de la placa ESP32. La conexión con el módulo MPU6050 se realiza por seis líneas, dos de alimentación, dos para el envío y recepción de datos, la línea de reloj y otro para generar la interrupción. Por último, la conexión con el dispositivo receptor (un ordenador o un *smartphone*) se realiza mediante *Bluetooth*.

En la Figura 4 se muestra una vista general del sistema desarrollado en forma de diagrama de bloques.

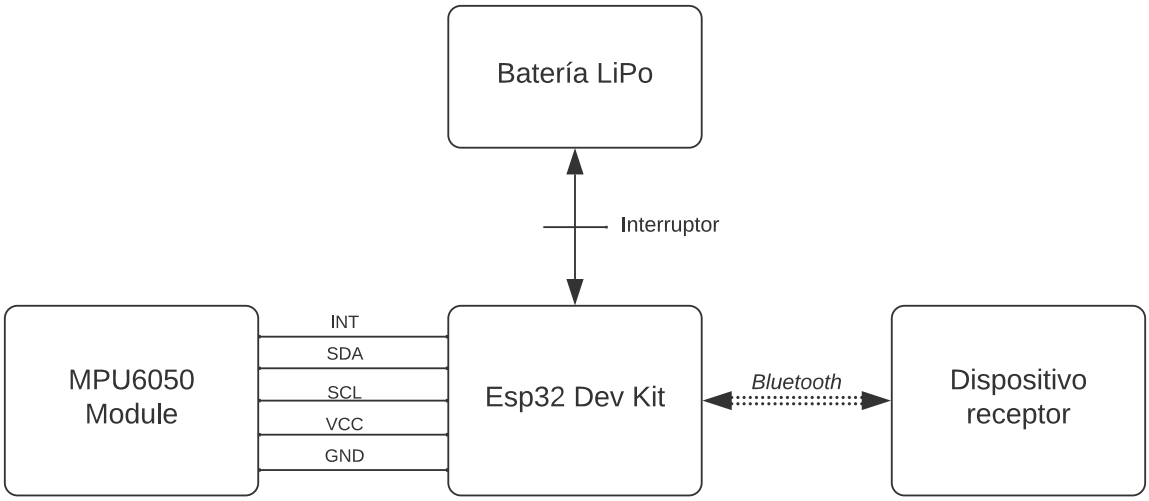


Figura 4. Diagrama general del sistema.

2.2. Flujo de Ejecución

El sistema diseñado consta de pocos pasos para ponerlo en marcha:

1. Instalar el software de recepción y emisión de datos.
2. Cargar la batería de nuestro dispositivo.
3. Encender el dispositivo mediante el interruptor de *on/off*.
4. Activar el *Bluetooth* en nuestro *smartphone* y mantenerlo al alcance del dispositivo (la antena *Bluetooth* tiene unos 30 m de alcance sin paredes de por medio) y buscar el dispositivo "*Pole Vault Tracker*".

5. Enviar un carácter por consola para empezar a recibir valores de aceleración lineal en mm/s^2 (X, Y, Z) y la posición angular en grados (*yaw*, *pitch* y *roll*[2]).

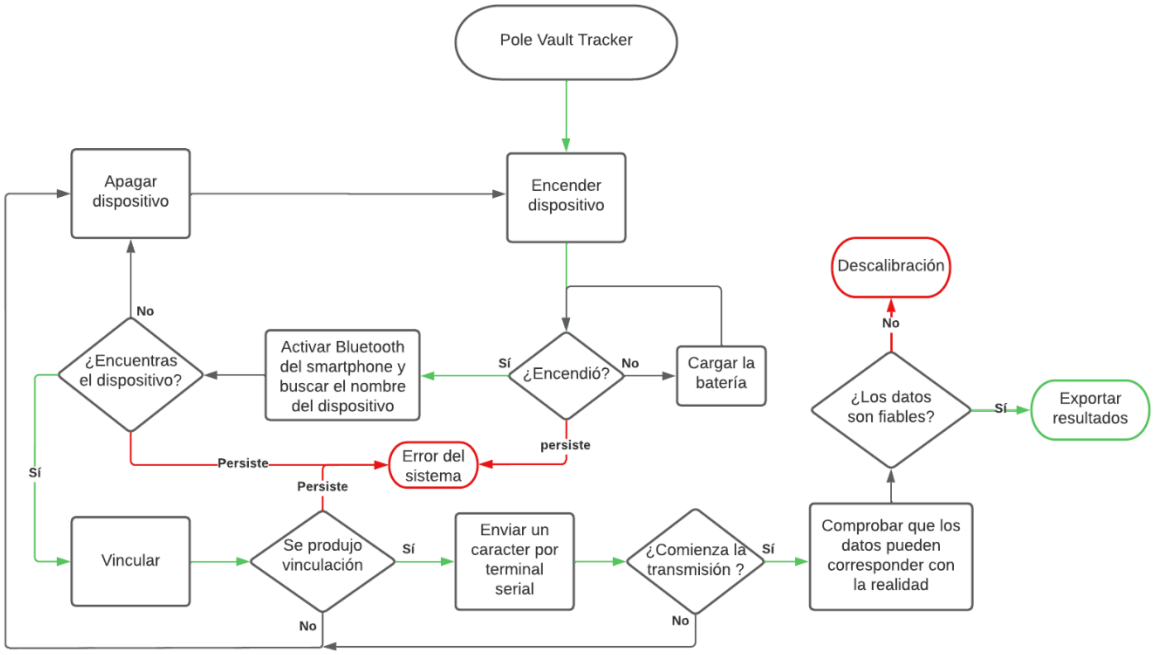


Figura 5. Diagrama de flujo de ejecución.

- Error de sistema: El error de sistema puede deberse a la avería de cualquiera de los componentes del dispositivo (placa ESP32, sensor MPU6050 o batería).
- Descalibración: El código del ESP32 emplea la librería MPU6050.h, creada por Jeff Rowberg [3]. Esta librería requiere de unos parámetros de calibración que se obtienen tras realizar una primera ejecución del código. Si el sensor se descalibra, se debe repetir el proceso de recalibración.

2.3. Normativa aplicable

A este proyecto aplica la siguiente normativa [4]:

- Componentes electrónicos – Símbolos IEC 60617.
- Componentes electrónicos – Valores normalizados IEC 60063 UNE 20531.
- Componentes electrónicos – Marcado IEC 60062 UNE-EN 60062.
- Componentes electrónicos – Marcado Circuitos Integrados.

[2] Rotación respecto al eje vertical o guiñada (*yaw*), rotación respecto al eje transversal o cabeceo (*pitch*) y rotación respecto al eje longitudinal o alabeo (*roll*) [14].

- Componentes electrónicos – Encapsulado JEDEC - JESD30-B.
- Diseño de circuito impreso. IPC 221.
- Fabricación de circuito impreso. IPC 4101.
- Compatibilidad electromagnética, norma IEC 61000 – UNE-EN 61000.
- Sostenibilidad medioambiental -RoHS.
- Sostenibilidad medioambiental -RAEE.

3. Selección del hardware y software

A continuación, se exponen tanto el software como el hardware empleados y su correspondiente justificación de uso.

El dispositivo debe encajar en la mayoría de las pértigas, y además debe haber cierto margen para la incorporación de un encapsulado protector. La mayoría de las pértigas tienen un diámetro interior no superior a 35 mm, mientras que por lo general el espesor de la pared de la pértiga, la cual es hueca, es de unos 2 mm. El espesor deberá tenerse en cuenta para el diseño de encaje del encapsulado en la pértiga. Finalmente, el tamaño seleccionado para la dimensión total del dispositivo encapsulado será de 70 mm de largo y 35 mm de diámetro (en la parte que se introduce en la pértiga).

3.1. Microcontrolador ESP32

El microcontrolador es la parte más importante de nuestro diseño. Un microcontrolador lento puede provocar que la lectura de nuestro sensor sea torpe y perdamos información del salto. Por ese motivo se realizó una tarea de investigación previa a la selección del microcontrolador en cuestión. Para empezar, me informo en foros, revistas digitales, canales de divulgación tecnológica en *YouTube*, páginas web y trabajos compartidos en la web sobre cuáles son los microcontroladores más usados para trabajos de IoT[3], ya que estos proyectos normalmente emplean conexiones *Wireless* como *Bluetooth*, son alimentados por baterías y son compactos, tres características que buscamos en nuestro diseño.

Enseguida observo que la mayoría de los proyectos que se encuentran son realizados con placas *Arduino*. Las más frecuentes son *Arduino UNO*, *Mini*, *Nano*, *Micro* y rara vez la *Arduino Mega*, cuya principal diferencia con el resto es su mayor cantidad de pines para proyectos en los que se requiere una cantidad mayor de entradas y salidas para sensores y actuadores. También posee una memoria EEPROM[4] (*Electrically Erasable Programmable Read-Only Memory*) con mayor capacidad. Esta placa de desarrollo no es óptima para este proyecto por su tamaño y consumo, por lo que es descartada. La *Arduino UNO* es la más frecuente con diferencia, y la más usada para iniciarse en el mundo de los microcontroladores, pero, pese a que tiene unas características aceptables, es descartada por la misma razón que la *Mega*, su tamaño. A continuación, nos encontramos con dos placas de la marca *Arduino* que cuentan con un tamaño bastante reducido y que por consiguiente pueden ser bastante competentes a la hora de postularse como el microcontrolador de nuestro dispositivo. Ambas placas poseen características muy similares, son la *Arduino Micro* y *Nano*. Por sus características de tamaño y precio son interesantes, y pese a que no cuentan con conexión inalámbrica directa, las preselecciono como

[3] *Internet de las cosas*, por sus siglas en inglés de *Internet of Things.*
[4] Es un tipo de memoria ROM que puede ser programada, borrada y reprogramada eléctricamente.

posibles candidatas. Por último, encuentro una placa de la empresa *Espressif*, bastante compacta, con módulo de conexión *Bluetooth* y Wi-Fi incorporados, además de un precio bastante competitivo, lo que la convierte en la mejor candidata hasta el momento. Esta placa es la *ESP32 NodeMCU*. Además, descubro que la empresa tiene una versión anterior más compacta y que por tanto nos puede ser de interés, la *ESP8266 WeMos D1 Mini*.

	ESP32 NodeMCU	**ESP8266 WeMos D1 Mini**	**Arduino Nano RP2040 Connect**	**Arduino Micro**
Microcontrolador	ESP32	ESP8266	ATECC608A Crypto	ATmega32u4
Tensión funcion., V (Mín./Tip./Máx.)	3/3,3/3,6	3,3	3,3	5
Fuente de alimentación (V)	7 - 12	4 - 6	5 - 21	7 - 21
Consumo de corriente (mA)	0,020 - 240	0,015 - 400		19 - 180
Consumo Stand-by (µA)	5	-	-	-
Pines de E/S digitales	36	11	14	20
Pines de E/S digitales con PWM	36	11		7
Pines de entrada analógica	15	1	8	12
SPI/I2C/I2S/UART	4/2/2/2	1/1/1/1	1/3/0/1+3(QSPI)	
Corriente CC por pin de E/S (mA)	20	-	-	40
Memoria FLASH	4 MB	4 MB	16 MB	32 KB
SRAM	520 KB	-	520 KB	2,5 KB
EEPROM	-	-	-	1 KB
Velocidad de reloj (MHz)	80/160	80/160	133	16
Longitud mm	52	34	45	48
Ancho mm	31	26	18	18
WIFI	SÍ	SÍ	SÍ	NO
Bluetooth	SÍ	NO	SÍ	NO
Sensor táctil	SÍ	NO	NO	NO
Interfaz MAC Ethernet	SÍ	NO	NO	NO
CAN	SÍ	NO	NO	NO
Sensor de temperatura	SÍ	NO	SÍ	NO
Conexión USB	SÍ	SÍ	SÍ	SÍ
Sensor giroscopio	NO	NO	SÍ	NO
Sensor acelerómetro	NO	NO	SÍ	NO
Sensor podómetro	NO	NO	SÍ	NO
Precio (€)	**9,79**	**6**	**25,20**	**21,60**

Tabla 1. Características principales de las distintas placas de microcontroladores elegibles.

En la Tabla 1 se especifican las características principales de estas cuatro placas a fin de realizar una comparativa con mayor detalle.

Observamos que la placa que incorpora el chip ESP32 y la *Arduino Nano* con chip de *Raspberry*, destacan bastante del resto en cuanto a especificaciones. Ambas placas cuentan con conectividad *Bluetooth* y Wi-Fi, lo que las hace mejores que las otras dos. Sin embargo, la *Arduino Nano* cuenta con el IMU[5] incorporado, lo que es un gran punto a su favor ya que prácticamente no haría falta ninguna adaptación más que la batería de litio para su funcionamiento. Además, esta placa *Arduino* cuenta con sensor de pasos y características para aplicar algoritmos de *Machine Learning* con *Tensor Flow Lite*.

Finalmente, me decanto por la ESP32, pues esta placa es más que suficiente para realizar la tarea para la que la necesitamos. Emplear una tarjeta como la *Arduino Nano RP2040 Connect* sería poco óptimo pues estaríamos pagando un precio casi tres veces superior por unas características que no vamos a aprovechar. Pese a que tenemos que comprar el sensor e incorporarlo con una placa PCB[6], aun así, sigue saliendo por la mitad de precio.

3.2. Unidad de medición inercial MPU6050

En la búsqueda de un sensor que cumpla con los requisitos no hubo tanta variedad como en la del microcontrolador. De nuevo, se busca información de qué sensores se están usando en proyectos parecidos. Existen multitud de proyectos de robótica y aeronaves como drones o helicópteros, que emplean el MPU6050 como unidad de medición inercial (IMU), e incluso muchos teléfonos móviles incorporan este sensor.

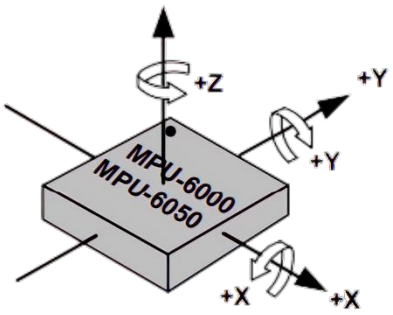


Figura 6. Sensor MPU6050 con 6 ejes de libertad.

Lo principal es que, además, este sensor se comercializa a un precio bastante competente montado en un módulo que cuenta con una librería muy completa para el software *Arduino*, para el cual es compatible el microcontrolador seleccionado.

Otro de los puntos fuertes del módulo es que se comercializa en una versión muy compacta y fácil de conexionar con nuestra placa ESP32 mediante pines con soldadura THD[7].

[5] Unidad de Medición Inercial, por sus siglas en inglés *Inertial Measurement Unit.*
[6] Placa de Circuito Impreso, por sus siglás en inglés *Printed Circuit Board*
[7] *Through-Hole Device*: Dispositivos que se disponen en las placas de circuito impreso mediante huecos perforados en las mismas.

3.3. Batería de polímero de litio

La alimentación del dispositivo debe realizarse mediante una batería puesto que se trata de un elemento portátil que además debe incorporarse a la pértiga. Las condiciones que debe reunir la batería son:

- **Tamaño**: La forma no es estricta, pero sus dimensiones deben ofrecer maniobrabilidad para poder adaptarla al conjunto en el encapsulado. Una pila cilíndrica o muy alta no sería fácil de acoplar.
- **Voltaje**: El voltaje de alimentación de la placa seleccionada es de 3,3 V, pudiendo ir desde 2,55 V hasta 3,6 V. Si no, se requeriría un convertidor *Buck* para reducir su voltaje. Esto no sería óptimo pues los convertidores tienen un rendimiento energético inferior al 100 %, es decir, pierden energía en forma de calor. Elegiremos por tanto una batería con un voltaje de 3,3 V o ligeramente superior en su defecto.
- **Duración/capacidad**: La duración de la batería dependerá directamente del consumo del dispositivo. En la tabla 1 se especifica que el ESP32 tendrá un máximo de corriente de 240 mA, lo que equivaldría aproximadamente a 1 hora de duración con una capacidad de unos 250 mAh; y un mínimo de 5 µA con el modo de sueño profundo (s*tand-by*) activado, lo que equivale a más de 5 años (probablemente la batería se degradaría y dejaría de funcionar antes de agostarse; aun así, esta duración no sería posible con la transmisión *Bluetooth* activada).
- **Variedad**: La variedad de baterías que se encuentra en el mercado hoy en día es abrumadora. Existen multitud de tipos de batería, por lo que iremos uno por uno enumerando pros y contras.
    - **Banco de energía (*Power Bank*)**: el banco de energía es una opción, energéticamente hablando, poco eficiente, pues normalmente se emplean baterías de polímero de litio en su interior, cuyas tensiones son elevadas mediante convertidores para alcanzar el estándar de 5 V al que se cargan los móviles. El ESP32 tendría que reducir internamente estos 5 V para volver a convertirlos en 3,3 V, por lo que hemos desperdiciado bastante energía. Por otro lado, el banco de energía ya trae su propio encapsulado, lo que tampoco sería efectivo a nivel de dimensiones.
    - **Pilas de Ni-MH**[8]: Este tipo de baterías poseen un voltaje nominal de 1,2 V por batería, llegando a 2,4 V con dos unidades. Si añadimos una tercera tendríamos un voltaje máximo de 3,6 V, lo que sería ideal, pero dimensionalmente poco óptimo pues las baterías igualarían en tamaño al dispositivo entero. Por ese motivo quedan descartadas.
    - **Pilas de litio**: Este tipo de batería ofrece un voltaje nominal de 1,5 V por celda que se mantiene bastante estable durante gran parte de la descarga. Con dos celdas obtendríamos un voltaje de 3 V, con lo que estaríamos en el límite de lo recomendado para la alimentación de la placa; por

[8] Una pila o batería de níquel-metalhidruro o de níquel-hidruro metálico es un tipo de pila o batería recargable que utiliza un ánodo de oxihidróxido de níquel, como en la batería de níquel cadmio, pero cuyo cátodo es de una aleación de hidruro metálico.

debajo de esto podría dar fallos de funcionamiento. Dado que no podemos asegurar el correcto funcionamiento tras una cierta descarga, quedan descartadas las baterías de litio.

- **Pilas $LiFePO_4$** [9]: Se trata de una batería con una densidad energética menor que el resto de las baterías, llegando incluso a ofrecer un 70 % menos de energía que una batería de litio del mismo tamaño. Ofrecen una tensión nominal de 3,2 V, lo que hasta ahora es la mejor opción en cuanto a tensión nominal que hemos encontrado. El inconveniente de este tipo de baterías es el tamaño. Por su baja densidad energética, las baterías de LiFePO suelen ser más grandes que el resto de las baterías de sus mismas características.
- **Batería de LiPo**[10]: Las baterías de polímero de litio son una buena opción en cuanto a tamaño pues tienen una morfología distinta al resto que hemos visto, de manera que puede ser fácilmente adaptable para encajar con el resto del dispositivo dentro de un encapsulado. El voltaje máximo indicado por los fabricantes de este tipo de baterías ronda los 4,2 V cuando se encuentran con una carga máxima. Si vamos a alimentar la placa por el puerto de 3,3 V podríamos dañar algo en el funcionamiento de la placa, por lo que tendríamos que alimentarla por el puerto micro USB.

Finalmente escogemos las baterías de LiFePO y LiPo para probarlas y ver qué rendimiento aportan en pruebas reales. La batería de LiPo es más económica y fácil de conseguir con las características especificadas, por lo que buscamos primero una batería de este material para realizar pruebas y ver si existe compatibilidad.

Las baterías de polímero de litio compradas de 250 mAh y 3,7 V (véase la figura 7), tienen las siguientes dimensiones: 32 mm de largo, 20,5 mm de ancho y espesor de 5,3 mm.

En la hoja de datos se especifica que las baterías llegan a 4,2 V en su carga máxima, por lo que pueden conectarse por el pin que atraviesa un reductor de tensión *Buck* que la estabiliza en 3,3 V. En primer lugar, la placa no es capaz de alimentar el dispositivo (ESP32 + MPU6050) y, tras medir la tensión de la batería con carga máxima, se observa que no supera los 3,6 V así que se decide alimentar la placa por el puerto de 3,3 V. Esta vez sí responde y es capaz de alimentar el módulo del sensor MPU6050.

Asignamos la batería como fuente de alimentación del dispositivo. aportándonos energía durante una media de 1 h. Esta duración es suficiente para recoger datos durante una sesión normal de

[9] Las pilas de litio-ferrofosfato son un tipo de batería recargable, concretamente una batería de ion-litio con un cátodo de fosfato de hierro-litio.
[10] Las baterías de polímero de iones de litio son pilas recargables (células de secundaria), compuestas generalmente de varias células secundarias idénticas en paralelo para aumentar la capacidad de la corriente de descarga.

entrenamiento. Si necesitásemos entrenar durante un período más largo se podrían conectar dos baterías en paralelo para prolongar la duración del tiempo de funcionamiento del dispositivo.

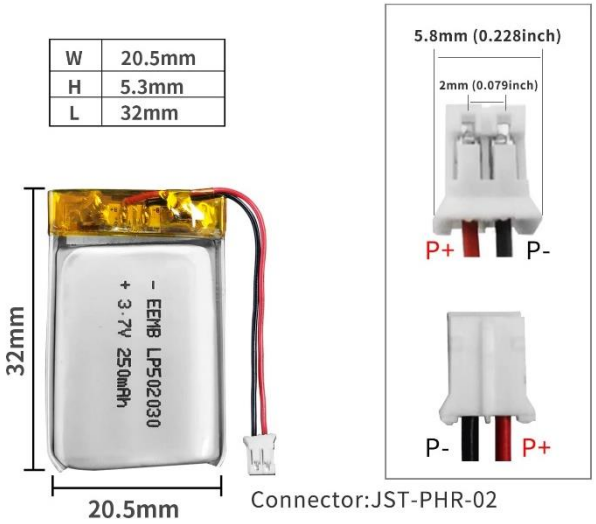


Figura 7. Batería de LiPo 250 mAh, 3,7 V.

La morfología aplanada de la batería permite acoplarla con facilidad al dispositivo sin la necesidad de crear un compartimento individual en el encapsulado.

3.4. Placa de Circuito Impreso

- **Fabricación de la placa de circuito impreso**: La selección del fabricante para imprimir la placa fue hecha en base a precios, tiempos de envío y valoración popular. En otras ocasiones he trabajado con *PCBWay* y *JLCPCB*, que son dos empresas de China y Hong Kong, respectivamente, las cuales ofrecen buenos precios con respecto a su calidad y plazos de entrega, por lo que las elegí nuevamente. Tras comprobar los precios y plazos de envío de ambas, selecciono *JLCPCB*.
- **Interruptor de alimentación**: Se opta por un interruptor unipolar por soldadura THD para controlar el sistema de alimentación de batería, el cual habrá que mantener en estado "*on*" para cargar la batería por el puerto micro USB de la placa ESP32.
- ***Array* de pines THD en L (1x2 pines)**: La conexión de la batería con la placa PCB fue mediante dos pines en L por soldadura THD.

3.5. Impresión 3D

- Para realizar el encapsulado del dispositivo se empleó una impresora de extrusión de filamento termofusible por mapeado 3D o impresora 3D de la marca HP (*Hewlett-Packard*), en concreto el modelo

Ephestos 2. Se trata de una impresora de gran calidad; aun así, el encapsulado se puede imprimir prácticamente con cualquier impresora 3D.

- **Material de impresión 3D**: El material empleado fue TPU [11]amarillo de dureza 60D en la escala de dureza *Shore*[12] (Figura 8). Se trata de un material flexible con mucha resistencia contra impactos.

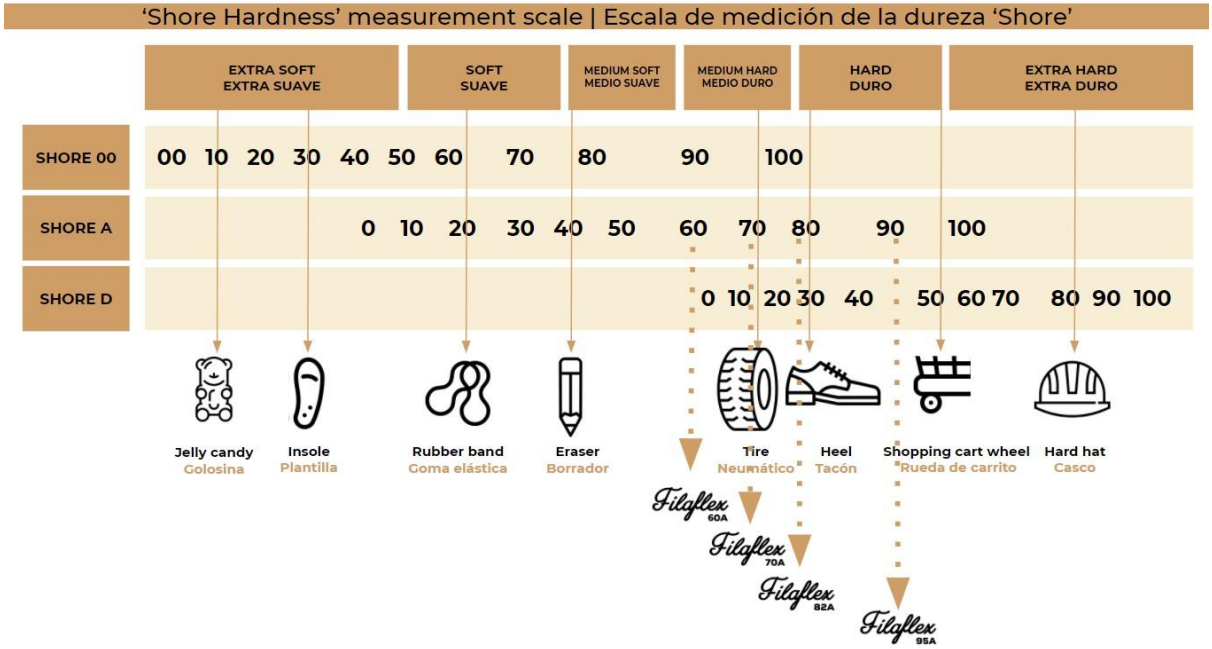


Figura 8. Escala de dureza D para materiales.

3.6. Software utilizado

- **IDE de programación del microcontrolador**: La comunidad de microcontroladores apoya mucho el ***IDE*[13] *de Arduino***, de ahí que cuente con tantas librerías. Para la placa seleccionada, *Espressif* ha desarrollado un software propio que, si vas a programar de forma profesional estos microcontroladores, ofrece posibilidades que el de la marca *Arduino* no te proporciona. Para la programación del código empleado, el IDE de *Arduino* no sólo es suficiente, sino que es más recomendable pues cuenta con librerías concretas para el módulo del sensor seleccionado.
- **Software de diseño de la PCB**: Para realizar el diseño del circuito impreso he utilizado **KiCAD**. La razón es que he realizado otras PCB con este software y es muy cómodo e intuitivo. Es muy fácil

[11] Poliuretano termo plástico. Es un polímero elastómero lineal y, por ende, termoplástico.
[12] La escala de dureza *Shore* es un ensayo no destructivo de dureza de materiales elásticos. Es determinada por la reacción elástica del material. Se deja caer un objeto sobre el material y se mide la altura a la que llega dicho objeto tras rebotar. La altura depende de la energía que se absorbe en el impacto contra el material.

- [13] *Integrated Development Environment* (Entorno Integrado de Desarrollo)

integrar huellas de componentes e incorpora un visor 3D para ver tanto las piezas como el componente final ensamblado.

- **Aplicación móvil de recepción serial por *Bluetooth***: Para recibir los datos que emite el dispositivo por puerto inalámbrico *Bluetooth*, se ha utilizado un software disponible en la *Play Store* de *Google* (*Serial Bluetooth Terminal*, de Kai Morich [5]), el cual ofrece una interfaz sencilla que permite la conexión con el módulo *Bluetooth* que incorpora la placa ESP32, y enviar y recibir datos de esta. Los datos recibidos se muestran en la terminal por pantalla y además te brinda la opción de exportar estos datos en un archivo para su posterior procesado y visualizado.

  Las imágenes de la Figura 9 y la Figura 10 forman parte de la interfaz de recepción de datos por serial *Bluetooth* de la aplicación móvil de Kai Morich. En la primera imagen se observa la selección de los dispositivos BLE disponibles. En la segunda ilustración se observa la terminal serial de la aplicación, donde podemos destacar varios datos recibidos con sus respectivos *timestamp*[14] y una sección para enviar datos al sensor. La aplicación permite personalizar seis botones de acceso directo para enviar mensajes. En este caso he añadido dos botones de "*On*" y "*Off*", que envía un 1 y un 0 respectivamente al dispositivo.

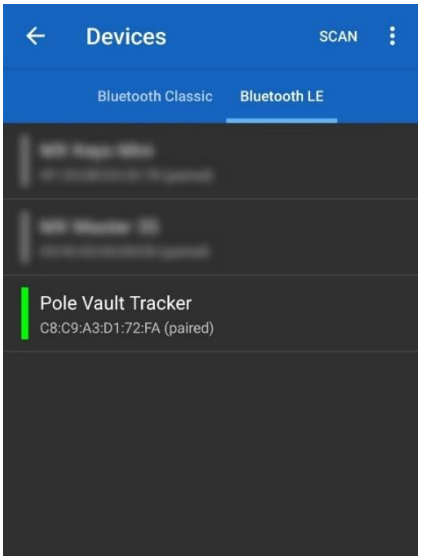


Figura 9. Interfaz de búsqueda de dispositivos BLE (Bluetooth Low Energy).

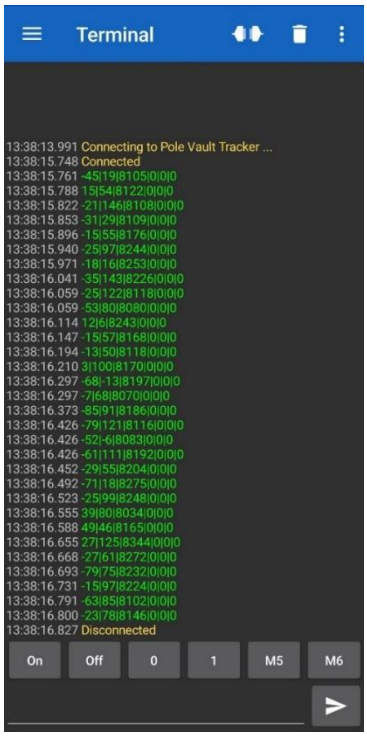


Figura 10. Terminal de comunicación de la aplicación.

- **Software de diseño 3D**: Para el diseño del encapsulado he empleado *Autodesk Fusion 360*. *Autodesk* ofrece licencias gratuitas a los estudiantes de grado que puedan demostrar mediante correo institucional

[14] Marca de tiempo.

que están cursando un grado. Se trata de un software de diseño 3D con una curva de aprendizaje relativamente rápida. Tiene una interfaz intuitiva y sencilla de manejar. Además, posee conexión con la nube donde se guardan los proyectos realizados de forma segura.

- **Ultimaker Cura**: Se trata de un software que convierte el sólido 3D diseñado en una serie de parámetros en código G[15]. *Cura* posee una de la interfaces más modernas y amigables. Es un software que puede utilizarse tanto a nivel principiante como a un nivel muy avanzado. A la hora de configurar nuestra impresión nos ofrece la versión rápida y básica, mientras que a su vez puedes configurar parámetros muy específicos.

[15] Los códigos G se emplean en la industria para la automatización de maquinarias. Funcionan por códigos de grupo que hacen referencia a tareas concretas.

4. Descripción del hardware y software utilizado

4.1. Hardware utilizado

El hardware consiste en un dispositivo SOC[16] que consta de un módulo con sensores giroscopio y acelerómetro ensamblados con un kit de desarrollo con chip ESP32 mediante una placa PCB diseñada mediante software e impresa por un tercero (JLCPCB). La alimentación se realiza con una batería de polímero de litio de 3,7 V, que es posible habilitar o deshabilitar mediante un interruptor incorporado a la placa PCB. Todo el dispositivo se inserta en un encapsulado impreso en material flexible y resistente que realiza la función de proteger y acoplar el sensor en la parte posterior de la pértiga.

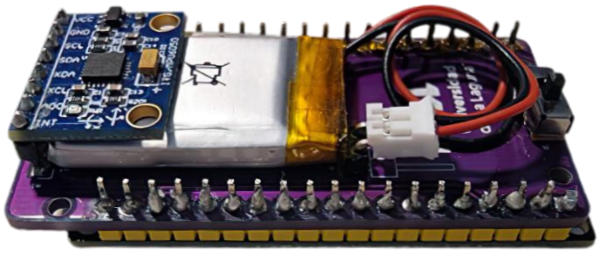


Figura 11. Captura del dispositivo terminado.

4.1.1. Placa PCB

El diseño de la placa PCB (véase la Figura 12) se lleva a cabo con la herramienta KiCAD. Consiste en una placa de conexión para unir de manera fija y sólida los pines del módulo sensor con los pines correspondientes del *ESP32 Dev Kit* mediante pistas de cobre. A esta placa se le incorporan por soldadura THD los siguientes componentes: un pin macho doble en L para la conexión con la batería, un interruptor que controla la alimentación, el módulo de sensores MPU6050 y la placa de desarrollo del módulo ESP32.

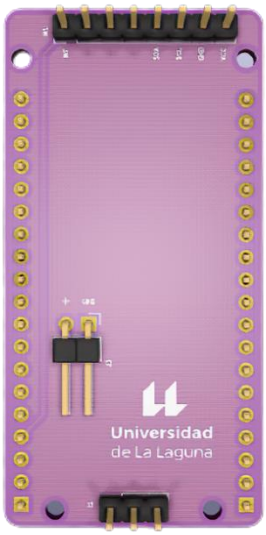


Figura 12. Anverso de la placa PCB de conexionado.

El desarrollo de la placa PCB se divide en cuatro pasos: Realización del esquemático, búsqueda de huellas o *footprints*, realización del archivo Gerber[17] y por último la impresión de la placa. En este caso la impresión se

[16] *System on Chip*
[17] Documentos vectoriales para la producción de circuitos impresos. Componen descripciones de las conexiones eléctricas, pistas, vías y pastillas de una PCB junto con todas las instrucciones para su producción.

encargó a la empresa JLCPCB, que fabrica la placa a partir de los archivos de extensión *.gbr* (*Gerber*) y *.drl* (taladros) que se les suministran.

4.1.2. Placa ESP32 Dev Kit

ESP32 es una familia de microcontroladores fabricados por la marca china *Espressif Systems*. La familia se compone de tres tipos de productos:

- **Chips:** El chip es básicamente el procesador de nuestro ESP32.
- **Módulos:** Incluyen dentro el chip. Pueden incorporar una memoria *Flash* de 4 MB o una memoria PSRAM de 8 MB. En este caso, la memoria es *Flash* de 4 MB. Cuenta con una antena impresa en la placa para las conexiones inalámbricas por Wi-Fi y *Bluetooth*. Además, los módulos facilitan la integración del ESP32 en las placas de desarrollo, pues cuentan con los pines de entrada/salida (E/S) listos para soldadura SMD[18].
- **Placas de desarrollo:** Son las más conocidas popularmente (véase la Figura 13). Estas traen un módulo soldado a una placa que incluye los *arrays* de pines listos para incorporar la placa a una *protoboard* (placa de pruebas), gracias a la separación estándar entre pines de 0,1 pulgadas (2,54 mm). Incluyen un regulador de tensión y un controlador serie micro USB para alimentar y programar la placa desde el mismo conector micro USB. También trae dos pulsadores para reiniciar la placa y para ponerla en modo carga.

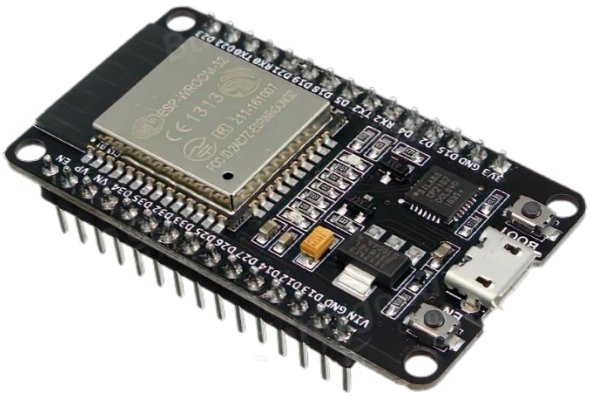


Figura 13. Dev. Kit NodeMCU ESP32.

El punto fuerte del ESP32 es la capacidad para establecer conexiones inalámbricas. Tanto por Wi-Fi como por *Bluetooth*. Esto es una gran ventaja respecto de otros microcontroladores que requieren de módulos extra para realizar estas comunicaciones inalámbricas. Además, cuenta con un modo de *Bluetooth* de bajo consumo. ESP32 aporta un aumento significativo de potencia respecto de su

[18] *Surface-Mount Device* (dispositivo de montaje superficial)

antecesor, el ESP8266, a pesar de que este también contaba ya con este tipo de conexión *Bluetooth*. Esto ocurre debido a que incorpora un procesador de doble núcleo con una frecuencia de hasta 240 MHz. El ESP8266 incorpora un procesador de un solo núcleo que alcanza 160 MHz, lo que se traduce en una potencia un 50 % menor. Además, el ESP32 cuenta con un coprocesador ULP (*Ultra-Low Power*) que permite un ahorro de energía máximo cuando no se necesita el procesador principal. Otra mejora que presenta es el aumento de pines de E/S o *General Purpose Input/Output* (GPIO). También incluye un conversor analógico/digital (ADC, *Analog-to-Digital Converter*) de mayor resolución (12 bits) y funcionalidades como interfaz *Ethernet* MAC (*Media Access Control*) o controlador CAN (*Controller Area Network*).

A continuación, se muestra un resumen de sus características principales:

- Capaz de lograr un consumo de energía ultra bajo
- Chip ESP-WROOM-32 incorporado
- Módulo de *Breadboard Friendly*
- Peso ligero y tamaño reducido
- Sensor *Hall* y de temperatura en el chip
- Utiliza el protocolo inalámbrico 802.11b/g/n (Wi-Fi)
- Capacidades de conectividad inalámbrica incorporadas
- Antena de PCB incorporada en el ESP32-WROOM-32
- Capacidad de PWM, I^2C, SPI, UART, 1 pin analógico
- Utiliza el módulo de interfaz de comunicación serial USB CP2102
- Programable con *ESP-IDF Toolchain*, *LuaNode SDK*, proyecto *Eclipse* (lenguaje *C*) y también permite su programación con el IDE de *Arduino*

4.1.3. Módulo de sensor MPU6050

El MPU6050 (véase la Figura 14) es un dispositivo de seguimiento de movimiento de 6 ejes basado en la tecnología MEMS (*Micro Electro Mechanical Systems*). Además de un sensor de temperatura, tiene un giroscopio, un magnetómetro y un acelerómetro en el chip. El módulo tiene un tamaño muy pequeño, cuenta con bajos requisitos de energía, una precisión alta, y buena resistencia a los golpes, programabilidad de energía específica de la aplicación y un precio más que competitivo. El MPU6050 se puede conectar fácilmente a microcontroladores a través de su interfaz de pines. Fue diseñado para teléfonos inteligentes, tabletas y sensores portátiles de bajo costo y alto rendimiento. Puede procesar

algoritmos de 9 ejes mientras captura simultáneamente el movimiento de los ejes X, Y y Z. El MPU6050 se utiliza en varios proyectos industriales y dispositivos electrónicos para controlar y detectar el movimiento 3D de varios objetos. El módulo MPU6050 consta de los siguientes bloques y funciones: Sensor giroscópico MEMS de 3 ejes con tres ADC de 16 bits y acondicionamiento de señal; un acelerómetro MEMS de 3 ejes con tres ADC de 16 bits y acondicionamiento de señal; motor de procesador de movimiento digital en chip; interfaz de comunicación primaria digital *Inter-Integrated Circuit* (I^2C), interfaz I^2C adicional para comunicarse con sensores externos, reloj interno, un registro de datos para almacenar datos del sensor, memoria FIFO[19] para reducir el consumo de energía, interrupción programable por el usuario y sensor de temperatura con una salida digital.

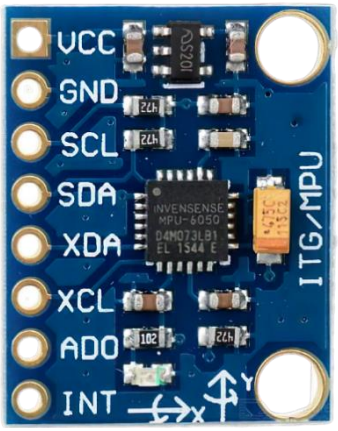


Figura 14. Módulo IMU MPU6050.

4.1.3.1. Características del giroscopio

El giroscopio MEMS de triple eje en el MPU6050 incluye una amplia gama de características [6]:

- Sensores de velocidad angular (giroscopios) de los ejes X, Y y Z de salida digital con un rango de escala completa programable por el usuario de ±250, ±500, ±1000 y ±2000 DPS[20].
- La señal de sincronización externa conectada al pin *FSYNC* admite sincronización de imagen, video y GPS[21].
- Los ADC integrados de 16 bits permiten el muestreo simultáneo de giroscopios.
- La estabilidad mejorada de la temperatura de polarización y sensibilidad reduce la necesidad de calibración por parte del usuario.
- Rendimiento de ruido de baja frecuencia mejorado.
- Filtro de paso bajo programable digitalmente.
- Corriente de funcionamiento del giroscopio: 3,6 mA.
- Corriente de espera: 5 µA.
- Factor de escala de sensibilidad calibrado de fábrica.
- Autoevaluación del usuario.

[19] *First In, First Out* (primero en entrar, primero en salir)
[20] *Degrees per second* (grados por segundo, º/s)
[21] *Global Positioning System* (Sistema de Posicionamiento Global)

4.1.3.2. Características del acelerómetro

El acelerómetro MEMS de triple eje en MPU6050 incluye una amplia gama de características [6]:

- Acelerómetro de triple eje de salida digital con un rango de escala completa programable de ±2g, ±4g, ±8g y ±16g.
- Los ADC de 16 bits integrados permiten el muestreo simultáneo de acelerómetros sin necesidad de multiplexor.
- Corriente de funcionamiento normal del acelerómetro: 500 µA.
- Corriente de modo de acelerómetro de baja potencia: 10 µA a 1,25 Hz, 20 µA a 5 Hz, 60 µA a 20 Hz, 110 µA a 40 Hz.
- Detección de orientación y señalización.
- Detección de toque.
- Interrupciones programables por el usuario.
- Interrupción de alta G.
- Autoevaluación del usuario.

4.1.3.3. Características adicionales

El MPU6050 incluye las siguientes características adicionales [6]:

- *MotionFusion* de 9 ejes por un Procesador de Movimiento Digital (DMP, *Digital Motion Processor*) en chip.
- Bus I^2C maestro auxiliar para leer datos de sensores externos (por ejemplo, magnetómetro).
- Corriente de funcionamiento de 3,9 mA cuando los 6 ejes de detección de movimiento y el DMP están habilitados.
- Rango de voltaje de suministro V_{DD} de 2,375-3,46 V.
- El voltaje de referencia V_{LOGIC} flexible admite múltiples voltajes de interfaz I^2C.
- Encapsulado QFN[22] más pequeño y delgado para dispositivos portátiles: 4×4×0,9 mm^3.
- Sensibilidad mínima del eje cruzado entre los ejes del acelerómetro y del giroscopio.
- El búfer FIFO de 1024 bytes reduce el consumo de energía al permitir que el procesador *host* lea los datos en ráfagas y luego entra en un modo de bajo consumo a medida que la MPU recopila más datos.
- Sensor de temperatura de salida digital.

[22] *Quad-Flat No-leads* es un tipo de encapsulado SMD donde las patillas son planas y se encuentran bajo el encapsulado de plástico en torno al borde del mismo, y por tanto resulta muy eficiente en cuanto a tamaño y uso del espacio, pues no se tienen patillas sobresaliendo por el exterior del encapsulado.

- Filtros digitales programables por el usuario para giroscopio, acelerómetro y sensor de temperatura.
- Tolerante a golpes hasta 10.000g.
- I^2C de modo rápido de 400 kHz para comunicarse con todos los registros.

4.2. Descripción del software

4.2.1. Arduino

Arduino [7] es una empresa que desarrolla hardware y software libre. Esta es mundialmente conocida por sus placas de microcontroladores, las cuales incorporan además memoria y un almacenamiento en el que se carga un código de programación escrito en un lenguaje simplificado de *C*, el cual cuenta con una variedad muy amplia de librerías propias. Estas placas cuentan con una serie de pines de entrada y salida a los cuales se les puede agregar sensores y actuadores digitales y analógicos con los que realizar infinidad de proyectos.

Figura 15. Logotipo de la compañía Arduino.

Para poder llevar a cabo estos proyectos se requiere de un entorno de desarrollo integrado o IDE. *Arduino IDE* es el entorno oficial de esta compañía y, aunque ésta cuenta con un lenguaje propio de programación derivado del *C*, el software también acepta lenguaje escrito en *C*, *C++*, *Wiring* (una plataforma de prototipado electrónico en la cual está basado *Arduino*), así como en *Processing* (un lenguaje de programación basado en *Java*, pero enfocado a placas electrónicas, en el cual a su vez está basado *Wiring*). Entre todos estos lenguajes no existe uno mejor que otro para programar microcontroladores. Por tanto, dependiendo del programador y su experiencia con estos lenguajes, su desempeño a la hora de escribir código será mejor con el lenguaje con que más experiencia tenga. Pero si no se tiene experiencia con ninguno o de ellos, se recomienda utilizar el desarrollado por *Arduino*, pues cuenta con mucho más apoyo de la comunidad.

Su sintaxis, como ya hemos mencionado anteriormente, es muy similar a la de *C++*. Tiene la posibilidad de realizar comentarios (texto que el compilador ignora) en el código para entender mejor lo que vamos programando, tanto nosotros como terceros que vayan a leer nuestro código.

Al igual que en *C++*, cada sentencia escrita debe finalizar con *";"* o de lo contrario el compilador unirá esta sentencia con la siguiente produciendo un error.

Arduino incorpora nuevas funciones o métodos especializadas para microcontroladores como pueden ser *digitalWrite*, la cual nos permite activar o desactivar los pines de nuestra placa.

4.2.2. KiCAD

KiCAD [8] es una herramienta *open source* de diseño de circuitos impresos, la cual es desarrollada por el equipo de desarrollo de *KiCAD* y cuyos principales autores son Jean-Pierre Charras, Dick Hollenbeck y Wayne Stambaugh (líder del proyecto).

Figura 16. Logotipo del software KiCAD.

KiCAD emplea un IDE en el que engloba el diseño del circuito esquemático, el diseño de PCB, y la generación y visualización de archivos *Gees* para *footprints* y símbolos. También dispone de un visualizador 3D para los componentes.

Existen varias herramientas con las que se pueden realizar diseños de circuitos electrónicos, pero *KiCAD* es la más potente e intuitiva dentro de la comunidad *open source*.

4.2.3. Fusion360

Fusion 360 [9] es una plataforma de software de modelado 3D, CAD[23], CAM[24], CAE[25] y PCB basada en la nube destinada al diseño y la fabricación de productos.

[23] *Computer-Aided Design* (Diseño asistido por computadora).
[24] *Computer-Aided Manufacturing* (Fabricación asistida por computadora).
[25] *Computer-Aided Engineering* (Ingeniería asistida por computadora).

Figura 17. Logotipo del Software Autodesk Fusion 360.

Fusion 360 es un software flexible y fácil de usar que permite explorar muchas iteraciones rápidamente; permite la creación de piezas mecanizadas CNC[26] de alta calidad y la fabricación aditiva de las mismas mediante FFF[27] o PBF[28] en el caso de impresión 3D en metal; también ofrece la posibilidad de probar los diseños mediante simulación por software para asegurar que superarán las condiciones reales una vez se fabriquen.

Además, permite trabajar en equipos colaborativos gracias a su herramienta de colaboración en la nube integrada. *Autodesk* destaca por brindar apoyo a la comunidad educativa, ofreciendo licencias gratuitas a estudiantes que puedan justificar su matrícula en un centro de estudios autorizado mediante un correo electrónico institucional.

[26] *Computer Numerical Control* (Control Numérico por Computadora): Sistema automatizado para el control de herramientas de fabricación mediante códigos programados en un dispositivo de almacenamiento.
[27] *Fused Filament Fabrication* (Fabricación con Filamento Fundido): Fabricación aditiva basada en la deposición continuada de material hasta conformar la pieza final.
[28] *Powder Bed Fusion* (fusión por lecho de polvo): Fabricación aditiva basada en la deposición de sucesivas capas de material en polvo, que luego son fundidas.

5. Diseño del sistema y resultados obtenidos

5.1. Encapsulado 3D

Se ha desarrollado un encapsulado (véase la *Figura* 18) que envuelve a la placa que contiene el dispositivo microcontrolador a fin de protegerlo, además de ofrecer la morfología adecuada para encajar en la parte posterior de la pértiga. Se ha diseñado un sistema de cierre del encapsulado por presión para encerrar por completo el dispositivo.

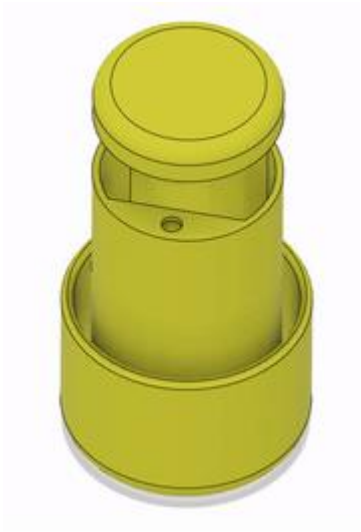


Figura 18. Modelo 3D del encapsulado

El encapsulado fue diseñado mediante la herramienta *Fusion360* con licencia gratuita y *open source*. Aprovechando su morfología simétrica respecto del eje longitudinal, se realiza un diseño 2D y se realiza una operación de generación del sólido por revolución. Posteriormente, se efectúa el diseño de la tapa de la misma forma. Por último, se genera el diseño del sistema de cierre. Este consiste en generar dos esferas salientes en la superficie de contacto de una de las caras, y dos huecos esféricos de radio ligeramente mayor para que encajen a presión.

El primer paso fue tomar medidas de las pértigas con un calibre, para conocer los diámetros y espesores típicos de estas. A continuación, se seleccionó un margen de entre el diámetro máximo y el mínimo válido para el encapsulado, de forma que una pértiga con un diámetro muy estrecho no sería adaptable. Para el caso del diámetro externo no habrá problema ya que se ha adaptado a las pértigas más exigentes.

A continuación, se procede al diseño en *Fusion 360*. El primer paso fue diseñar la vista de la planta a partir de las medidas obtenidas en la medición (véase la Figura 19).

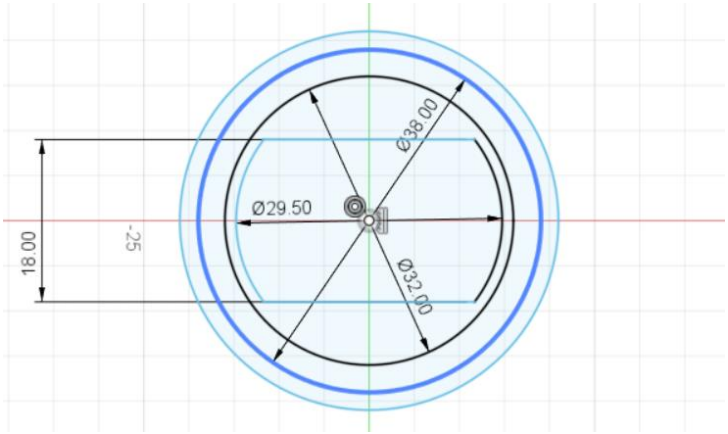


Figura 19. Vista de planta del diseño inicial del encapsulado.

La forma interior ha sido diseñada en base a las medidas del dispositivo terminado para ajustar todo lo posible el espacio entre dispositivo y encapsulado. El espacio podría generar vibraciones,

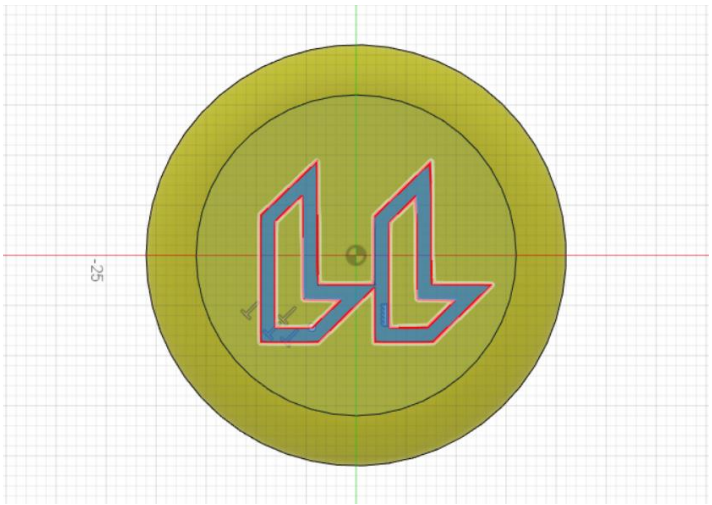


Figura 20. Diseño del logotipo de la universidad en el encapsulado.

El diseño del logotipo de la universidad se ha realizado a partir de una imagen original que ha tenido que ser transformada en formato *.svg*[29] y a partir de él se ha obtenido un boceto. A este boceto se le ha dado grosor y se ha extruido para dar profundidad a las letras.

[29] Gráficos vectoriales escalables

Figura 21. Alzado del modelo 3D y su sección transversal.

En la Figura 21 se observa la estructura exterior e interior del encapsulado. Podemos observar con detalle la forma que toma el encapsulado en su interior a través de una vista de sección.

5.2. Código del microcontrolador ESP32

Se ha empleado como base el código de ejemplo que nos ofrece *Arduino* mediante la librería *I2Cdev.h* y *MPU6050_6axis_MotionApps_V6_12.h* con autoría de Jeff Rowberg [3]:

```
// I2C device class (I2Cdev) demonstration Arduino sketch for MPU6050 class using DMP (MotionApps v2.0)
// 6/21/2012 by Jeff Rowberg <jeff@rowberg.net>
// Updates should (hopefully) always be available at https://github.com/jrowberg/i2cdevlib
//
// Changelog:
//      2019-07-08 - Added Auto Calibration and offset generator
//       - and altered FIFO retrieval sequence to avoid using blocking code
//      2016-04-18 - Eliminated a potential infinite loop
//      2013-05-08 - added seamless Fastwire support
//                 - added note about gyro calibration
//      2012-06-21 - added note about Arduino 1.0.1 + Leonardo compatibility error
//      2012-06-20 - improved FIFO overflow handling and simplified read process
//      2012-06-19 - completely rearranged DMP initialization code and simplification
//      2012-06-13 - pull gyro and accel data from FIFO packet instead of reading directly
//      2012-06-09 - fix broken FIFO read sequence and change interrupt detection to RISING
//      2012-06-05 - add gravity-compensated initial reference frame acceleration output
//                 - add 3D math helper file to DMP6 example sketch
//                 - add Euler output and Yaw/Pitch/Roll output formats
//      2012-06-04 - remove accel offset clearing for better results (thanks Sungon Lee)
//      2012-06-01 - fixed gyro sensitivity to be 2000 deg/sec instead of 250
//      2012-05-30 - basic DMP initialization working
```

Figura 22. Título y autoría del código de ejemplo.

```
I2Cdev device library code is placed under the MIT license
Copyright (c) 2012 Jeff Rowberg

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.
```

Figura 23. Declaración de copyright

El código de ejemplo ofrece la posibilidad de obtener cuaterniones, ángulos de *Euler*, ángulos *yaw*, *pitch* y *roll* (YPR), aceleraciones lineales y aceleraciones respecto del norte magnético. En este caso hemos recopilado los datos de aceleraciones reales respecto del norte magnético y los ángulos YPR.

```
MPU6050 mpu;

// Components with gravity removed and adjusted for the world frame of
// reference (yaw is relative to initial orientation, since no magnetometer
// is present in this case). Could be quite handy in some cases.
#define OUTPUT_READABLE_WORLDACCEL

// pitch/roll angles (in degrees) calculated from the quaternions coming
// from the FIFO. Note this also requires gravity vector calculations.
// Also note that yaw/pitch/roll angles suffer from gimbal lock.

#define OUTPUT_READABLE_YAWPITCHROLL
#define INTERRUPT_PIN 15
#define LED_PIN 2
bool blinkState = false;

// MPU control/status vars
bool dmpReady = false;  // set true if DMP init was successful
uint8_t mpuIntStatus;   // holds actual interrupt status byte from MPU
uint8_t devStatus;      // return status after each device operation (0 = success, !0 = error)
uint16_t packetSize;    // expected DMP packet size (default is 42 bytes)
uint16_t fifoCount;     // count of all bytes currently in FIFO
uint8_t fifoBuffer[64]; // FIFO storage buffer

// orientation/motion vars
Quaternion q;           // [w, x, y, z]         quaternion container
VectorInt16 aa;         // [x, y, z]            accel sensor measurements
VectorInt16 gy;         // [x, y, z]            gyro sensor measurements
VectorInt16 aaReal;     // [x, y, z]            gravity-free accel sensor measurements
VectorInt16 aaWorld;    // [x, y, z]            world-frame accel sensor measurements
VectorFloat gravity;    // [x, y, z]            gravity vector
float euler[3];         // [psi, theta, phi]    Euler angle container
float ypr[3];           // [yaw, pitch, roll]   yaw/pitch/roll container and gravity vector
```

Figura 24. Definición de variables.

Además, hemos adaptado el código para realizar la comunicación serial a través de conexión *Bluetooth*. Se emplearon para ello las librerías de *Bluetooth Low Energy* (BLE) *BLEDevice.h*, *BLEServer.h*, *BLEUtils.h* y *BLE902.h*.

```
//BLE
#include <BLEDevice.h>
#include <BLEServer.h>
#include <BLEUtils.h>
#include <BLE2902.h>
```

Figura 25. Librerías BLE de Arduino.

```
// ================================================================
// ===                      UPDATE BT INFO                      ===
// ================================================================

void updateBTInfo(bool isConnected){
  Serial.print(isConnected ? "BT::Connected!" : "BT::Not Connected");
}
```

Figura 26. Información sobre la conexión del Bluetooth.

```
// ============================================================
// ===                      INIT BLE                        ===
// ============================================================

void initBT(){
  // Create the BLE Device
  BLEDevice::init("Pole Vault Tracker");

  // Create the BLE Server
  pServer = BLEDevice::createServer();
  pServer->setCallbacks(new MyServerCallbacks());

  // Create the BLE Service
  BLEService *pService = pServer->createService(SERVICE_UUID);

  // Create a BLE Characteristic
  pCharacteristic = pService->createCharacteristic(
                      CHARACTERISTIC_UUID,
                      BLECharacteristic::PROPERTY_READ   |
                      BLECharacteristic::PROPERTY_WRITE  |
                      BLECharacteristic::PROPERTY_NOTIFY |
                      BLECharacteristic::PROPERTY_INDICATE
                    );

  // https://www.bluetooth.com/specifications/gatt/viewer?attributeXmlFile=org.bluetooth.descriptor.gatt.client_characteristic_configuration.xml
  // Create a BLE Descriptor
  pCharacteristic->addDescriptor(new BLE2902());

  // Start the service
  pService->start();

  // Start advertising
  BLEAdvertising *pAdvertising = BLEDevice::getAdvertising();
  pAdvertising->addServiceUUID(SERVICE_UUID);
  pAdvertising->setScanResponse(true);
  pAdvertising->setMinPreferred(0x0);  // set value to 0x00 to not advertise this parameter
  BLEDevice::startAdvertising();
  Serial.println("Waiting a client connection to notify...");
}
```

Figura 27. Definición de la función inicio de Bluetooth.

En la función *setup* se inicializan el reloj a 400 kHz y la comunicación serial a 115 200 baudios[30], lo que se puede observar en la Figura 28.

```
// ================================================================
// ===                      INITIAL SETUP                       ===
// ================================================================

void setup() {
  // join I2C bus (I2Cdev library doesn't do this automatically)
  #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
    Wire.begin();
    Wire.setClock(400000); // 400kHz I2C clock. Comment this line if having compilation difficulties
  #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
    Fastwire::setup(400, true);
  #endif

  // initialize serial communication
  Serial.begin(115200);
  Serial.println("Bluetooth Device is Ready to Pair");
```

Figura 28. Código ESP32 - Inicio del reloj y la comunicación serial.

Para poner en marcha la transmisión de datos, el dispositivo se mantiene a la espera de recibir un carácter por *Bluetooth* serial. Una vez lo recibe realiza el proceso de calibración inicial donde establece el vector que indica 0º en todos los ángulos de Euler:

[30] El baudio es una unidad de medida utilizada en telecomunicaciones que representa el número de símbolos por segundo en un medio de transmisión.

```
178  // initialize device
179  Serial.println(F("Initializing I2C devices..."));
180  mpu.initialize();
181  pinMode(INTERRUPT_PIN, INPUT_PULLUP);
182
183  // verify connection
184  Serial.println(F("Testing device connections..."));
185  Serial.println(mpu.testConnection() ? F("MPU6050 connection successful") : F("MPU6050 connection failed"));
186
187    // wait for ready
188  Serial.println(F("\nSend any character to begin DMP programming and demo: "));
189  while (Serial.available() && Serial.read()); // empty buffer
190  while (!Serial.available());                 // wait for data
191  while (Serial.available() && Serial.read()); // empty buffer again
```

Figura 29. Código ESP32 – Inicio de dispositivo, verificación de conexión y puesta en espera.

Se mostrará por pantalla el siguiente mensaje si se ha realizado correctamente la conexión y se ha enviado correctamente la orden de inicio:

```
194  // load and configure the DMP
195  Serial.println(F("Initializing DMP..."));
196  devStatus = mpu.dmpInitialize();
```

Figura 30. Mensaje de inicio de transmisión de datos.

En caso de fallar, aparecerá el siguiente mensaje:

```
227    // 2 = DMP configuration updates failed
228    // (if it's going to break, usually the code will be 1)
229    Serial.print(F("DMP Initialization failed (code "));
230    Serial.print(devStatus);
231    Serial.println(F(")"));
```

Figura 31. Mensaje de fallo en la transmisión de datos.

Se establece el pin por defecto de la ESP32 para notificar la transmisión serial, se envía por puerto serial el encabezado de los datos y se llama la función de inicio de *Bluetooth* vista anteriormente:

```
234  // configure LED for output
235  pinMode(LED_PIN, OUTPUT);
236
237  initBT();
238
239  Serial.println("x,y,z,yaw,pitch,roll");
240 }
```

Figura 32. Fin declaración función de Setup.

A continuación, el proceso entrará en el bucle típico de los microcontroladores que se encarga de ejecutar un código de manera infinita. En él se pueden llamar a interrupciones que generen diferentes estímulos en el sistema. En este bucle se obtienen los ángulos y aceleraciones mencionadas anteriormente mediante el objeto MPU. Se envían por puerto serial y se hace parpadear el led de envío de datos incorporado en nuestra placa ESP32 (ver Figura 33).

```
242 // ================================================================
243 // ===                      MAIN PROGRAM LOOP                     ===
244 // ================================================================
245
246 void loop() {
247   // if programming failed, don't try to do anything
248   if (!dmpReady) return;
249   // read a packet from FIFO
250   if (mpu.dmpGetCurrentFIFOPacket(fifoBuffer)) { // Get the Latest packet
251
252     // display Euler angles in degrees
253     mpu.dmpGetQuaternion(&q, fifoBuffer);
254     mpu.dmpGetGravity(&gravity, &q);
255     mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
256
257     // display initial world-frame acceleration, adjusted to remove gravity
258     // and rotated based on known orientation from quaternion
259     mpu.dmpGetQuaternion(&q, fifoBuffer);
260     mpu.dmpGetAccel(&aa, fifoBuffer);
261     mpu.dmpGetGravity(&gravity, &q);
262     mpu.dmpGetLinearAccel(&aaReal, &aa, &gravity);
263     mpu.dmpGetLinearAccelInWorld(&aaWorld, &aaReal, &q);
264
265     Serial.print(int(aaWorld.x)); Serial.print(",");
266     Serial.print(int(aaWorld.y)); Serial.print(",");
267     Serial.print(int(aaWorld.z)); Serial.print(",");
268     Serial.print(int(ypr[2] * 180 / M_PI)); Serial.print(",");  //Pitch
269     Serial.print(int(ypr[0] * 180 / M_PI)); Serial.print(",");  //Yaw
270     Serial.println(int(ypr[1] * 180 / M_PI));                  //Roll
271
272     // blink LED to indicate activity
273       blinkState = !blinkState;
274       digitalWrite(LED_PIN, blinkState);
```

Figura 33. Declaración del bucle de repetición infinita.

Se establece un *delay* para dar tiempo al microprocesador a que procese bien los datos. Y a continuación se analizan tres posibilidades (Figura 34):

- Si el dispositivo está conectado se actualizan los datos y se notifica.
- Si el dispositivo estaba conectado y se desconecta, se notifica y se detiene la actualización de los datos enviados por *Bluetooth*.
- Si el dispositivo no estaba conectado y se conecta, se vuelve a activar la actualización del envío de datos.

```
276   delay(30);
277
278   // Notify changed value
279   if (deviceConnected) {
280     String str = "";
281     str += int(aaWorld.x);
282     str += ",";
283     str += int(aaWorld.y);
284     str += ",";
285     str += int(aaWorld.z);
286     str += ",";
287     str += int(ypr[2] * 180 / M_PI);
288     str += ",";
289     str += int(ypr[0] * 180 / M_PI);
290     str += ",";
291     str += int(ypr[1] * 180 / M_PI);
292
293     pCharacteristic->setValue((char*)str.c_str());
294     pCharacteristic->notify();
295   }
296   // Disconnecting
297   if (!deviceConnected && oldDeviceConnected) {
298         delay(500); // give the bluetooth stack the chance to get things ready
299         pServer->startAdvertising(); // restart advertising
300         Serial.println("start advertising");
301         oldDeviceConnected = deviceConnected;
302         updateBTInfo(false);
303   }
304   // Connecting
305   if (deviceConnected && !oldDeviceConnected) {
306       // do stuff here on connecting
307       oldDeviceConnected = deviceConnected;
308       updateBTInfo(true);
309   }
310 }
```

Figura 34. Condiciones en función de la conexión.

5.3. Recepción de datos por serial *Bluetooth*

La conexión es realizada mediante emparejamiento *Bluetooth* de dispositivos en el terminal móvil o computadora (véase la Figura 35).

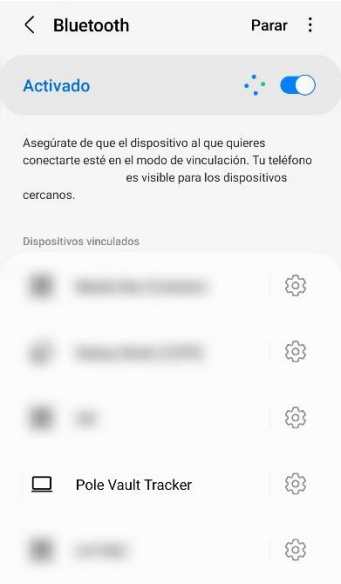


Figura 35. Emparejamiento del dispositivo con el teléfono móvil.

Tras este emparejamiento, realizamos la conexión en el menú de dispositivos o directamente en la terminal a través de la flecha que se muestra en la Figura 36. Una vez nos llegan los mensajes de conexión, se envía un carácter por consola para activar el envío de datos.

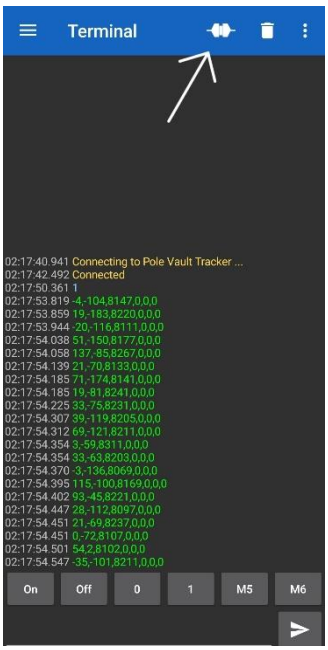


Figura 36. Terminal Serial Bluetooth, conexión.

Para guardar los datos, la *App* nos permite descargarlos como texto plano de la siguiente forma:

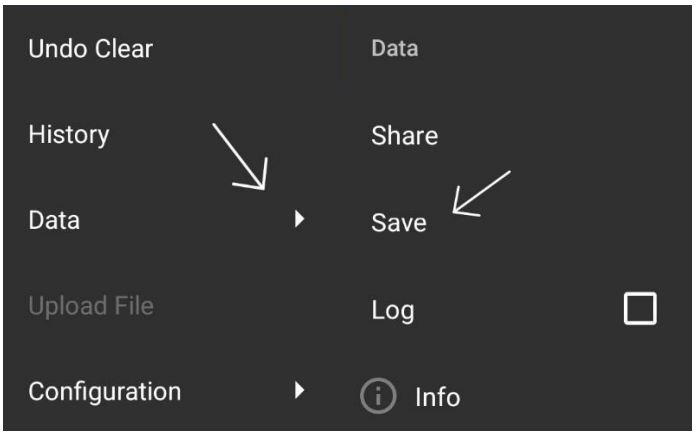


Figura 37. Exportar datos a texto plano.

Se ha de indicar que no ha ocurrido ningún tipo de fallo que haya impedido obtener datos del sensor por conexión *Bluetooth.* La conexión es robusta hasta unos 40 m de distancia. Una vez superada esta, comienza a crear latencia en la transmisión de datos. Además, la latencia provocaría la pérdida de información en las medidas, por lo que no se recomienda situarse a una distancia mayor a 30 m.

5.4. Análisis de los datos en *Python*

Se ha llevado a cabo la realización del siguiente código escrito en lenguaje *Python*, que se encarga de representar gráficamente los resultados de datos angulares de la componente *pitch*. Se le carga como entrada un fichero de datos con extensión *.csv* y se obtiene como salida una gráfica de los valores de *pitch* a lo largo del tiempo.

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import datetime
import time

df = pd.read_csv('SaltoJavi1.csv')
df['timestamp'] = pd.to_datetime(df['timestamp'], format='%H:%M:%S.%f')
df = df.set_index('timestamp')
df.index = pd.to_datetime(df.index)

x = df['x']
y = df['y']
z = df['z']
yaw = df['yaw']
pitch = df['pitch']
roll = df['roll']

dfPitch = pd.DataFrame(df.pitch)

for i in range(183, dfPitch.shape[0]):
    dfPitch.iloc[i] = dfPitch.iloc[i] + 360

for i in range(0,dfPitch.shape[0]):
    dfPitch.iloc[i] = dfPitch.iloc[i] - 90


plt.plot(dfPitch.index, dfPitch.pitch, color = "black", label="Pitch")
plt.xlabel("Tiempo (H/M/S)")
plt.ylabel("Angulo (º)")
plt.legend(loc="upper left")
plt.grid()
plt.show()
```

Figura 38. Código de visualizado de datos en Python.

```python
import pandas as pd
import matplotlib.pyplot as plt
```

Figura 39. Librerías Python.

La primera parte del código son las librerías que se emplean: *matplotlib.pyplot* es una interfaz basada en estado para *matplotlib*; proporciona una forma de representación gráfica implícita, similar a *MATLAB*. También abre figuras en su pantalla y actúa como el administrador de GUI de figuras. Por su parte, *pyplot* está diseñado principalmente para gráficos interactivos y casos simples de generación de gráficos programáticos [10].

La librería *pandas* es necesaria para el tratamiento de datos. Esta librería se caracteriza principalmente por:

- Utilizar un objeto *DataFrame* (tablas en dos dimensiones) rápido y eficiente para la manipulación de datos con indexación integrada
- Disponer de herramientas para leer y escribir datos entre estructuras de datos en memoria y diferentes formatos: CSV y archivos de texto, Microsoft Excel, bases de datos SQL y el formato rápido HDF5
- Permitir la alineación inteligente de datos y el manejo integrado de los mismos
- Proporcionar la transformación y rotación flexibles de conjuntos de datos
- Incorporar la segmentación inteligente basada en etiquetas, indexación elegante y creación de subconjuntos de grandes conjuntos de datos
- Estar altamente optimizado en rendimiento, con rutas de código críticas escritas en *Python* o C.

Además, *Python* con *pandas* se usa en una amplia variedad de dominios académicos y comerciales, que incluyen finanzas, neurociencia, economía, estadísticas, publicidad, análisis web y más [11].

Creamos un marco de datos en el que cargamos los datos. El marco de datos o *data frame* es un formato de la librería *pandas* que estructura los datos en una tabla parecida al formato Excel. Seleccionamos la columna de *timestamp* como índice, lo que facilita la representación gráfica de los datos en base al tiempo.

```
df = pd.read_csv('SaltoJavi1.csv')
df['timestamp'] = pd.to_datetime(df['timestamp'], format='%H:%M:%S.%f')
df = df.set_index('timestamp')
df.index = pd.to_datetime(df.index)
```

Figura 40. Preparación de datos del CSV mediante librería pandas.

Con los datos obtenidos, creamos *arrays* para el posterior tratado de los datos y su representación gráfica.

```
x = df['x']
y = df['y']
z = df['z']
yaw = df['yaw']
pitch = df['pitch']
roll = df['roll']
```

Figura 41. Creación de arrays con los datos.

A continuación, configuramos los distintos apartados que se mostrarán en la gráfica resultante.

```
plt.plot(dfPitch.index, dfPitch.pitch, color = "black", label="Pitch")
plt.xlabel("Tiempo (H/M/S)")
plt.ylabel("Angulo (º)")
plt.legend(loc="upper left")
plt.grid()
plt.show()
```

Figura 42. Configuración de la gráfica.

Finalmente, obtenemos la gráfica de los resultados de *pitch*.

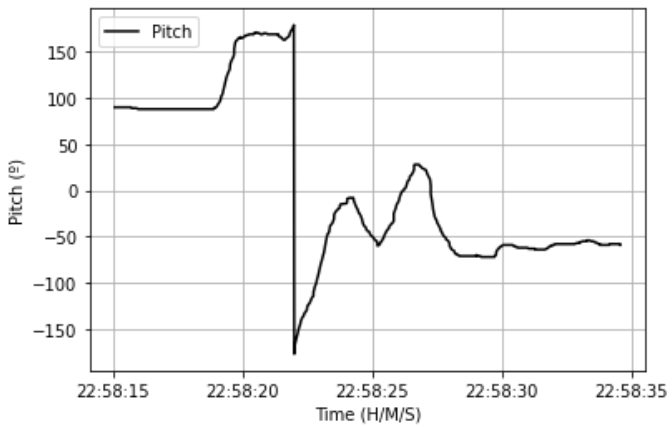


Figura 43. Gráfica de datos de pitch sin ajuste de ángulo.

Debemos sumar 360º a los datos a partir de que se alcanzan 180º, a fin de eliminar el salto brusco que se observa de 180º a -180º (código de la Figura 44). Esto se debe a que el sensor solo muestra datos entre -180º y 180º. Tras realizar este ajuste observamos continuidad en los datos (ver Figura 45).

```
dfPitch = pd.DataFrame(df.pitch)

for i in range(183, dfPitch.shape[0]):
    dfPitch.iloc[i] = dfPitch.iloc[i] + 360

for i in range(0,dfPitch.shape[0]):
    dfPitch.iloc[i] = dfPitch.iloc[i] - 90
```

Figura 44. Ajuste de los datos angulares de pitch.

5.4.1. Discusión de los resultados obtenidos

En la Figura 45 se muestran los datos del *pitch* una vez realizado el ajuste de continuidad descrito al final de la sección anterior. Recuérdese que el *pitch* o cabeceo nos da la rotación respecto al eje transversal y, por tanto, es la información más importante durante el salto con pértiga, pues nos marca el ángulo que tiene la pértiga respecto a la horizontal.

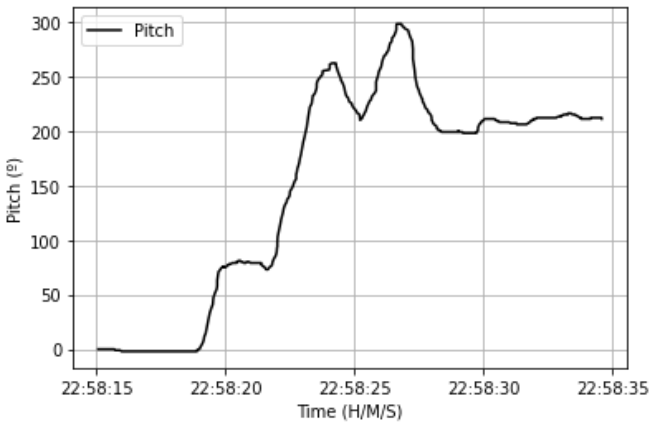


Figura 45. Gráfica de datos de pitch con ajuste de ángulos.

En primer lugar, se observa una continuidad de los datos en el tiempo sin desconexiones, ruido ni picos extraños. Esto indica un buen resultado en cuanto al diseño, fabricación y transmisión de los datos.

Sin embargo, los datos obtenidos son anormales pues en un salto con pértiga el dispositivo jamás llegaría a 300º como se observa en la Figura 45. En un salto con pértiga lo normal son ángulos entre 90º y -90º, pudiendo llegar a -180º en la caída de la pértiga posteriormente al salto.

Esto indica que se requiere ajuste en la matemática de los ángulos en preprocesado o en postprocesado. Esto se debe a que en un salto la pértiga no queda invariante en sus componentes *yaw* y *roll*, y estos giros afectan a la interpretación del grado de *pitch*.

Para una mejor compresión de lo que se quiere decir, pongamos la siguiente situación:

yaw = 0º, *roll* = 0º, *pitch* = 45º

Si girásemos el *roll* y el *yaw* 180º, el *pitch*, que debería permanecer en 45º, ahora valdría -45º. Esto ocurre a menudo en el salto con pértiga pues al pasar de la fase de carrera (Figura 46) a la fase de presentación de la pértiga, el *roll* varía en torno a 180º en el momento en el que el saltador pasa de transportar la pértiga (antebrazo vertical hacia abajo, Figura 47) a elevarla sobre la cabeza (antebrazo vertical hacia arriba). Lo mismo ocurre con el *yaw* entre la fase de presentación y la de franqueo del listón, donde se tiene que el saltador pasa de estar alineado con la dirección del movimiento a darle la espalda, y con él, la pértiga también gira 180º sobre el eje vertical (180º *yaw*).

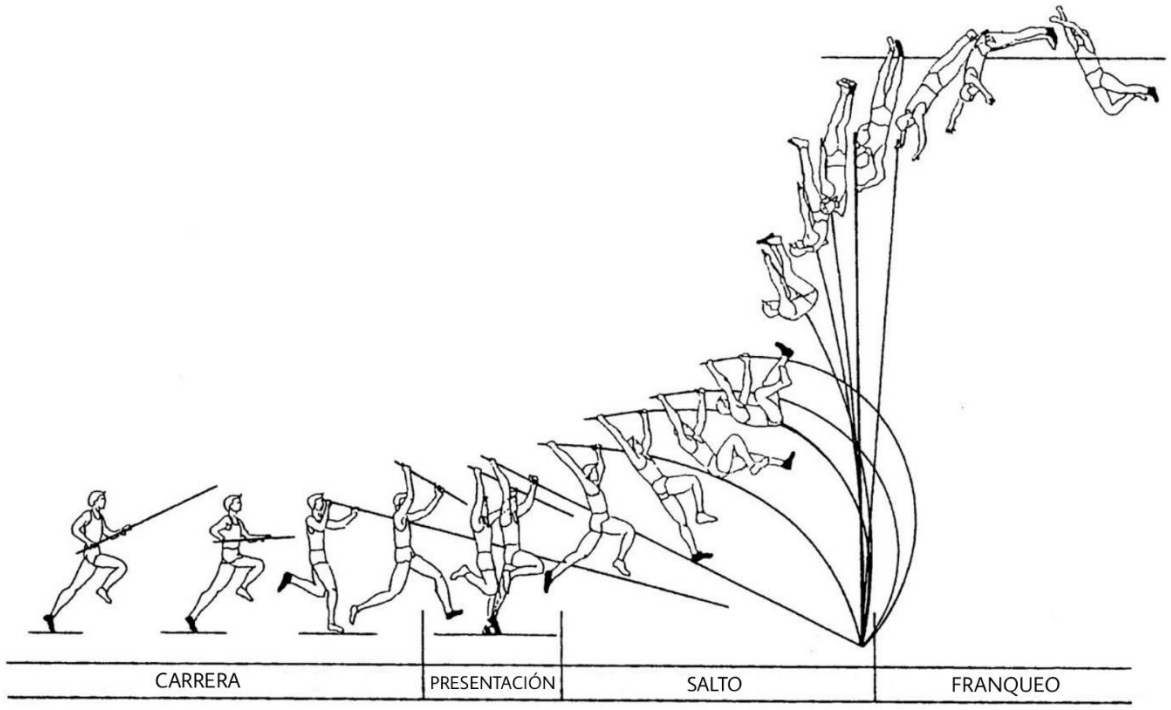


Figura 46. Fases del salto con pértiga (figura modificada de [12]).

Figura 47. Armand Duplantis en fase de carrera [13]

6. Presupuesto

6.1. Materiales

Elementos	P. Unitario (IGIC incluido)	Costes de envío	Precio total (IGIC incluido)
- *Impresión de placa PCB con JLCPCB (x5)*	0,43 €	3,92 €	6,07 €
- *Batería de Polímero de litio (x2)*	7,95	Gratis	15,9 €
- *Placa de desarrollo del módulo ESP32*	8,66 €	Gratis	8,66 €
- *Módulo de sensor MPU6050*	2,53 €	Gratis	2,53 €
- *Total*	19,57 €	3,92 €	**33,16 €**

Tabla 2. Coste de materiales.

6.2. Mano de obra

Elementos	Cantidad (horas)	Precio unitario (€)	Precio total
- *Búsqueda de información*	50	20	1.000 €
- *Diseño del sistema*	100	30	3.000 €
- *Análisis de resultados*	60	25	1.500 €
- *Documentación*	90	25	2.250 €
- *Total*	300	22,26	**7.750 €**

Tabla 3. Mano de obra

6.1. Coste de ejecución material

Elementos	Precio total
- *Materiales*	33,16 €
- *Mano de obra*	7.750 €
- *Total*	**7.783,16 €**

Tabla 4. Coste de ejecución material.

6.1. Gastos generales

Los gastos generales representarán un 13 % sobre el coste de ejecución material.

Elementos	Precio total
- *Gastos generales*	**1.011,81 €**

Tabla 5. Gastos generales.

6.2. Beneficio industrial

A su vez los beneficios industriales representan un 6 % sobre el coste de ejecución material.

Elementos	Precio total
- *Beneficio industrial*	**466,99 €**

Tabla 6. Beneficio industrial.

6.3. Impuestos

Sobre el total después de beneficio industrial se aplican los impuestos pertenecientes a la Comunidad de Canarias, el IGIC (Impuesto General Indirecto Canario) correspondiente a un 7 %.

Elementos	Precio total
- *Impuestos IGIC*	**648,33 €**

Tabla 7. Impuestos IGIC.

6.4. Coste final del proyecto

Elementos	Precio total
- *Materiales*	33,16 €
- *Mano de obra*	7.750 €
- *Gastos generales (13%)*	1.011,81 €
- *Beneficio industrial (6%)*	466,99 €
- *Total, antes de impuestos*	9.261,96 €
- *Impuestos IGIC (7%)*	648,33 €
- *Total*	**9.910,29 €**

Tabla 8. Coste final del proyecto.

7. Conclusión y líneas abiertas

El resultado del proyecto fue exitoso. Finalmente se ha conseguido crear un sistema de extracción de datos en el salto con pértiga con un correcto funcionamiento y una comodidad para el saltador excelente. El dispositivo posee una morfología discreta y compacta que se adapta perfectamente a la pértiga, además se aloja en una zona en la que no genera intrusión con la ejecución de la modalidad. Posee un peso de 54 g, lo que no supondría más de un 0,01 % del peso de un/a atleta, siendo por tanto despreciable. Se han conseguido emitir y recibir los datos recogidos por el dispositivo en un terminal móvil mediante una aplicación de transmisión serial por *Bluetooth*, lo que favorece la recepción de datos en tiempo real sin la necesidad de retirar el dispositivo de la pértiga.

Tras probar el dispositivo en un salto real con un atleta, se observa que los resultados de datos angulares no son 100 % válidos, pues se encuentran desajustes en los datos observables a simple vista, lo que fue desarrollado en el apartado *5.4.1 Discusión de los resultados obtenidos*. Esto se traduce en que se requiere de una reprogramación del microcontrolador en base a los resultados para corregir este desajuste.

En cuanto a los datos inerciales, se observa demasiado ruido, llegando a ser ininterpretables. La solución podría ser implementar un filtro de Kalman que eliminase el ruido blanco[31], pero es posible que el ruido sea demasiado intenso, como para rescatar información. También cabe la posibilidad de que la librería empleada no funcione correctamente. En cuyo caso habría que probar otras librerías para ver si el fallo persiste, lo cual puede ser porque el sensor de inercia lineal esté estropeado.

Mirando al futuro, se me ocurren ideas de mejora para seguir favoreciendo al rendimiento en el salto con pértiga. Y es que este sistema debe ser pensado para su uso por entrenadores y atletas, que no tienen por qué tener conocimientos sobre programación o análisis de datos. Por lo que la implementación de una aplicación móvil que recoja los datos, los procese y devuelva información sencilla de interpretar, puede ser de gran ayuda en un entrenamiento de salto con pértiga.

La combinación de estos datos con imágenes grabadas también puede aportar información al salto, y es que ya existen trabajos que están aplicando algoritmos de visión artificial a diferentes modalidades del atletismo para extraer medidas como velocidades, aceleraciones, etc. En el salto con pértiga, obtener variables como el ángulo que forma la pértiga con la horizontal en la batida o la distancia que separa la

[31] El ruido blanco es una señal de amplitud aleatoria y amplio margen de frecuencias, la cual se caracteriza porque sus sucesivas muestras no guardan correlación estadística en ningún momento.

batida del atleta con el final del cajetín donde clava la pértiga, pueden ser de mucha utilidad y, con estas tecnologías, ya es posible obtenerlas.

8. Bibliografía

[1] D. público, "Ceremonia de apertura de los Juegos Olímpicos de 1896.jpg," 1986. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/b/b6/1896_Olympic_opening_ceremony.jpg.

[2] A. P. Suárez, *Evolución histórica d elas marcas en el salto con pértiga,* Madrid, 2022.

[3] Rowberg, Jeff, "I2C device class (I2Cdev) demonstration Arduino sketch for MPU6050 class using DMP (MotionApps v2.0)".

[4] J. M. Cámara, "Normativa electrónica," [Online]. Available: https://riubu.ubu.es/bitstream/handle/10259/3589/Normativa.pdf?sequence=1&isAllowed=y.

[5] Kai Morich, "Serial Bluetooth Terminal," [Online]. Available: https://play.google.com/store/apps/details?id=de.kai_morich.serial_bluetooth_terminal.

[6] Inven Sense Inc., "MPU-6000/MPU-6050 Product Specification," 19 08 2013. [Online]. Available: https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf.

[7] Arduino, "Arduino," [Online]. Available: https://www.arduino.cc/.

[8] J.-P. Charras, "KiCAD," [Online]. Available: https://www.kicad.org/.

[9] Autodesk, "Fusion 360," [Online]. Available: https://www.autodesk.es/products/fusion-360/overview?term=1-YEAR&tab=subscription&plc=F360#!.

[10] J. Hunter, D. Dale, E. Firing and M. Droettboom, "Matplotlib," The Matplotlib development team, 2012 - 2022. [Online]. Available: https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.html. [Accessed 01 09 22].

[11] W. McKinney, J. Reback, J. Van den Bossche, T. Augspurger and M. Roeschke, "Pandas," 31 08 2022. [Online]. Available: https://pandas.pydata.org/about/team.html.

[12] Á. Sáinz, *Fases del salto con pértiga (Editados los nombres de las fases),* Madrid, 1996.

[13] A. NURKIEWICZ, *Armand Duplantis.*, GETTY IMAGES, 2021.

[14] Wikipedia, "Ejes del avión," [Online]. Available: https://es.wikipedia.org/wiki/Ejes_del_avi%C3%B3n. [Accessed 8 sept 2022].

[15] p. o. IOC, "Marc Wright during the pole vault competition at the 1912 Summer Olympics," 07 1912. [Online]. Available: https://commons.wikimedia.org/wiki/File:1912_Marc_Wright.JPG.

ANEXOS

ANEXO I

CÓDIGO MICROCONTROLADOR

```
// I2C device class (I2Cdev) demonstration Arduino sketch for MPU6050 class using DMP
(MotionApps v2.0)
// 6/21/2012 by Jeff Rowberg <jeff@rowberg.net>
// Updates should (hopefully) always be available at https://github.com/jrowberg/i2cdevlib
//
// Changelog:
//      2019-07-08 - Added Auto Calibration and offset generator
//       - and altered FIFO retrieval sequence to avoid using blocking code
//      2016-04-18 - Eliminated a potential infinite loop
//      2013-05-08 - added seamless Fastwire support
//                 - added note about gyro calibration
//      2012-06-21 - added note about Arduino 1.0.1 + Leonardo compatibility error
//      2012-06-20 - improved FIFO overflow handling and simplified read process
//      2012-06-19 - completely rearranged DMP initialization code and simplification
//      2012-06-13 - pull gyro and accel data from FIFO packet instead of reading directly
//      2012-06-09 - fix broken FIFO read sequence and change interrupt detection to RISING
//      2012-06-05 - add gravity-compensated initial reference frame acceleration output
//                 - add 3D math helper file to DMP6 example sketch
//                 - add Euler output and Yaw/Pitch/Roll output formats
//      2012-06-04 - remove accel offset clearing for better results (thanks Sungon Lee)
//      2012-06-01 - fixed gyro sensitivity to be 2000 deg/sec instead of 250
//      2012-05-30 - basic DMP initialization working

/* ============================================
I2Cdev device library code is placed under the MIT license
Copyright (c) 2012 Jeff Rowberg

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in
all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR
OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
```

```
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS
IN
THE SOFTWARE.
===============================================
*/
//BLE
#include <BLEDevice.h>
#include <BLEServer.h>
#include <BLEUtils.h>
#include <BLE2902.h>

BLEServer* pServer = NULL;
BLECharacteristic* pCharacteristic = NULL;
bool deviceConnected = false;
bool oldDeviceConnected = false;

#define SERVICE_UUID        "4fafc201-1fb5-459e-8fcc-c5c9c331914b"
#define CHARACTERISTIC_UUID "beb5483e-36e1-4688-b7f5-ea07361b26a8"


class MyServerCallbacks: public BLEServerCallbacks {
    void onConnect(BLEServer* pServer) {
      deviceConnected = true;
      BLEDevice::startAdvertising();
    };

    void onDisconnect(BLEServer* pServer) {
      deviceConnected = false;
    }
};

#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps_V6_12.h"

#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
#include "Wire.h"
#endif

MPU6050 mpu;

// Components with gravity removed and adjusted for the world frame of
// reference (yaw is relative to initial orientation, since no magnetometer
// is present in this case). Could be quite handy in some cases.
#define OUTPUT_READABLE_WORLDACCEL

// pitch/roll angles (in degrees) calculated from the quaternions coming
// from the FIFO. Note this also requires gravity vector calculations.
// Also note that yaw/pitch/roll angles suffer from gimbal lock.
```

```
#define OUTPUT_READABLE_YAWPITCHROLL
#define INTERRUPT_PIN 15
#define LED_PIN 2
bool blinkState = false;

// MPU control/status vars
bool dmpReady = false;  // set true if DMP init was successful
uint8_t mpuIntStatus;   // holds actual interrupt status byte from MPU
uint8_t devStatus;      // return status after each device operation (0 = success, !0 = error)
uint16_t packetSize;    // expected DMP packet size (default is 42 bytes)
uint16_t fifoCount;     // count of all bytes currently in FIFO
uint8_t fifoBuffer[64]; // FIFO storage buffer

// orientation/motion vars
Quaternion q;           // [w, x, y, z]         quaternion container
VectorInt16 aa;         // [x, y, z]            accel sensor measurements
VectorInt16 gy;         // [x, y, z]            gyro sensor measurements
VectorInt16 aaReal;     // [x, y, z]            gravity-free accel sensor measurements
VectorInt16 aaWorld;    // [x, y, z]            world-frame accel sensor measurements
VectorFloat gravity;    // [x, y, z]            gravity vector
float euler[3];         // [psi, theta, phi]    Euler angle container
float ypr[3];           // [yaw, pitch, roll]   yaw/pitch/roll container and gravity vector

// ================================================================
// ===                  UPDATE BT INFO                        ===
// ================================================================

void updateBTInfo(bool isConnected){
  Serial.print(isConnected ? "BT::Connected!" : "BT::Not Connected");
}

// ================================================================
// ===                    INIT BLE                            ===
// ================================================================

void initBT(){
  // Create the BLE Device
  BLEDevice::init("Pole Vault Tracker");

  // Create the BLE Server
  pServer = BLEDevice::createServer();
  pServer->setCallbacks(new MyServerCallbacks());

  // Create the BLE Service
  BLEService *pService = pServer->createService(SERVICE_UUID);

  // Create a BLE Characteristic
```

```
  pCharacteristic = pService->createCharacteristic(
                      CHARACTERISTIC_UUID,
                      BLECharacteristic::PROPERTY_READ   |
                      BLECharacteristic::PROPERTY_WRITE  |
                      BLECharacteristic::PROPERTY_NOTIFY |
                      BLECharacteristic::PROPERTY_INDICATE
                    );

  //
https://www.bluetooth.com/specifications/gatt/viewer?attributeXmlFile=org.bluetooth.descriptor.gatt.
client_characteristic_configuration.xml
  // Create a BLE Descriptor
  pCharacteristic->addDescriptor(new BLE2902());

  // Start the service
  pService->start();

  // Start advertising
  BLEAdvertising *pAdvertising = BLEDevice::getAdvertising();
  pAdvertising->addServiceUUID(SERVICE_UUID);
  pAdvertising->setScanResponse(true);
  pAdvertising->setMinPreferred(0x0);  // set value to 0x00 to not advertise this parameter
  BLEDevice::startAdvertising();
  Serial.println("Waiting a client connection to notify...");
}

// ===============================================================
// ===                  INITIAL SETUP                  ===
// ===============================================================

void setup() {
  // join I2C bus (I2Cdev library doesn't do this automatically)
  #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
    Wire.begin();
    Wire.setClock(400000); // 400kHz I2C clock. Comment this line if having compilation difficulties
  #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
    Fastwire::setup(400, true);
  #endif

  // initialize serial communication
  Serial.begin(115200);
  Serial.println("Bluetooth Device is Ready to Pair");

  // initialize device
  Serial.println(F("Initializing I2C devices..."));
  mpu.initialize();
  pinMode(INTERRUPT_PIN, INPUT_PULLUP);
```

```
  // verify connection
  Serial.println(F("Testing device connections..."));
  Serial.println(mpu.testConnection() ? F("MPU6050 connection successful") : F("MPU6050
connection failed"));

    // wait for ready
  Serial.println(F("\nSend any character to begin DMP programming and demo: "));
  while (Serial.available() && Serial.read()); // empty buffer
  while (!Serial.available());                 // wait for data
  while (Serial.available() && Serial.read()); // empty buffer again


  // load and configure the DMP
  Serial.println(F("Initializing DMP..."));
  devStatus = mpu.dmpInitialize();

  // supply your own gyro offsets here, scaled for min sensitivity
  mpu.setXGyroOffset(146);
  mpu.setYGyroOffset(2);
  mpu.setZGyroOffset(24);
  mpu.setXAccelOffset(2104);
  mpu.setYAccelOffset(-320);
  mpu.setZAccelOffset(5012);
  // make sure it worked (returns 0 if so)
  if (devStatus == 0) {
    // Calibration Time: generate offsets and calibrate our MPU6050
    mpu.CalibrateAccel(6);
    mpu.CalibrateGyro(6);
    Serial.println();
    mpu.PrintActiveOffsets();
    // turn on the DMP, now that it's ready
    Serial.println(F("Enabling DMP..."));
    mpu.setDMPEnabled(true);

    // enable Arduino interrupt detection
    mpuIntStatus = mpu.getIntStatus();

    // set our DMP Ready flag so the main loop() function knows it's okay to use it
    dmpReady = true;

    // get expected DMP packet size for later comparison
    packetSize = mpu.dmpGetFIFOPacketSize();
  } else {
    // ERROR!
    // 1 = initial memory load failed
    // 2 = DMP configuration updates failed
    // (if it's going to break, usually the code will be 1)
    Serial.print(F("DMP Initialization failed (code "));
```

```
    Serial.print(devStatus);
    Serial.println(F(")"));
  }

  // configure LED for output
  pinMode(LED_PIN, OUTPUT);

  initBT();

  Serial.println("x,y,z,yaw,pitch,roll");
}

// ================================================================
// ===                    MAIN PROGRAM LOOP                     ===
// ================================================================

void loop() {
  // if programming failed, don't try to do anything
  if (!dmpReady) return;
  // read a packet from FIFO
  if (mpu.dmpGetCurrentFIFOPacket(fifoBuffer)) { // Get the Latest packet

    // display Euler angles in degrees
    mpu.dmpGetQuaternion(&q, fifoBuffer);
    mpu.dmpGetGravity(&gravity, &q);
    mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);

    // display initial world-frame acceleration, adjusted to remove gravity
    // and rotated based on known orientation from quaternion
    mpu.dmpGetQuaternion(&q, fifoBuffer);
    mpu.dmpGetAccel(&aa, fifoBuffer);
    mpu.dmpGetGravity(&gravity, &q);
    mpu.dmpGetLinearAccel(&aaReal, &aa, &gravity);
    mpu.dmpGetLinearAccelInWorld(&aaWorld, &aaReal, &q);

    Serial.print(int(aaWorld.x)); Serial.print(",");
    Serial.print(int(aaWorld.y)); Serial.print(",");
    Serial.print(int(aaWorld.z)); Serial.print(",");
    Serial.print(int(ypr[2] * 180 / M_PI)); Serial.print(",");  //Pitch
    Serial.print(int(ypr[0] * 180 / M_PI)); Serial.print(",");  //Yaw
    Serial.println(int(ypr[1] * 180 / M_PI));                   //Roll

    // blink LED to indicate activity
      blinkState = !blinkState;
      digitalWrite(LED_PIN, blinkState);
  }
  delay(30);
```

```
  // Notify changed value
  if (deviceConnected) {
    String str = "";
    str += int(aaWorld.x);
    str += ",";
    str += int(aaWorld.y);
    str += ",";
    str += int(aaWorld.z);
    str += ",";
    str += int(ypr[2] * 180 / M_PI);
    str += ",";
    str += int(ypr[0] * 180 / M_PI);
    str += ",";
    str += int(ypr[1] * 180 / M_PI);

    pCharacteristic->setValue((char*)str.c_str());
    pCharacteristic->notify();
  }
  // Disconnecting
  if (!deviceConnected && oldDeviceConnected) {
        delay(500); // give the bluetooth stack the chance to get things ready
        pServer->startAdvertising(); // restart advertising
        Serial.println("start advertising");
        oldDeviceConnected = deviceConnected;
        updateBTInfo(false);
  }
  // Connecting
  if (deviceConnected && !oldDeviceConnected) {
      // do stuff here on connecting
      oldDeviceConnected = deviceConnected;
      updateBTInfo(true);
  }
}
```

ANEXO II

MEDIDAS

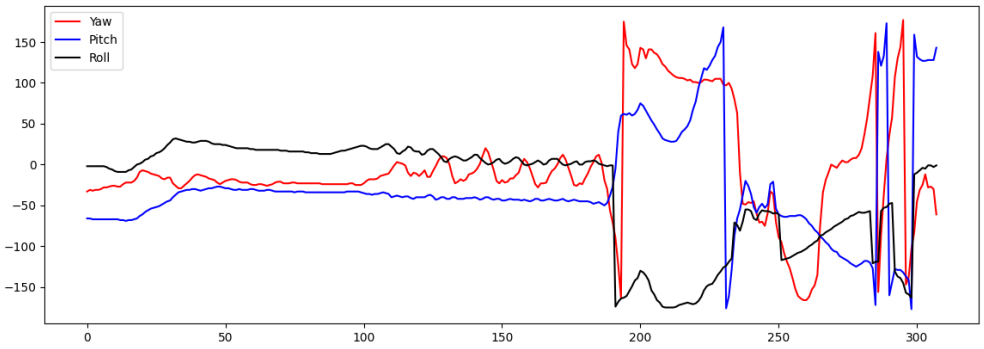


Figura 48. Valores de angulos YPR de un salto con pértiga obtenidos con el dispositivo.

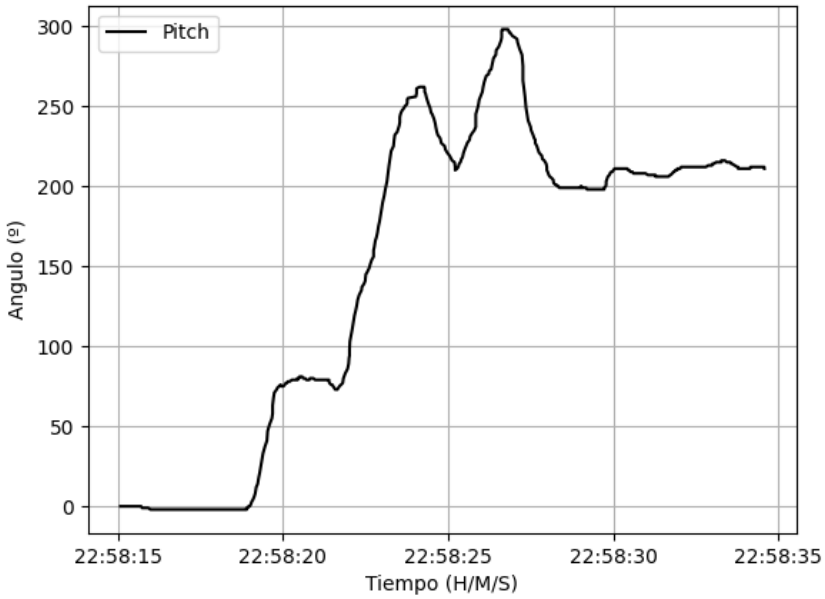


Figura 49. Gráfica de valores de angulo pitch de un salto con pértiga medido con el dispositivo.

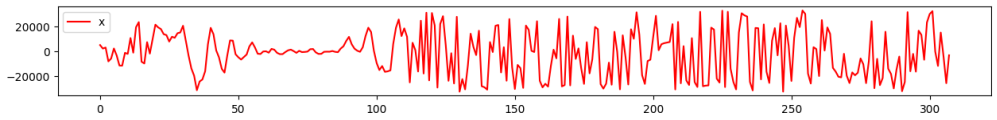


Figura 50. Gráfica de valores de aceleración en el eje X (transversal al pasillo) de un salto con pértiga medido con el dispositivo.

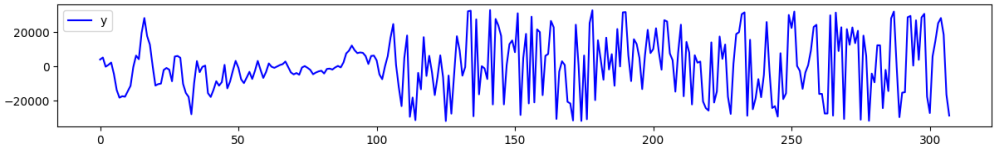


Figura 51. Gráfica de valores de aceleración en el eje Y (longitudinal al pasillo) de un salto con pértiga medido con el dispositivo.

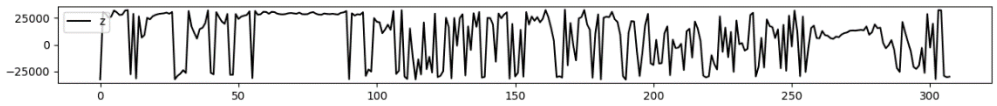
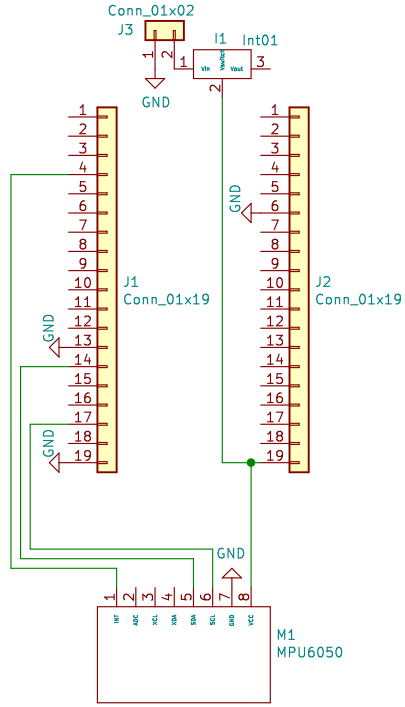


Figura 52. Gráfica de valores de aceleración en el eje Z (vertical) de un salto con pértiga medido con el dispositivo.

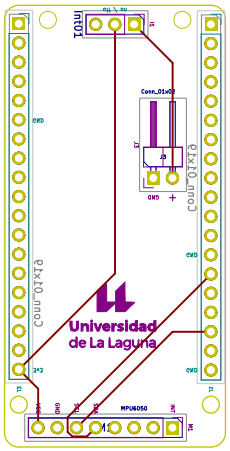
timestamp	x	y	z	yaw	pitch	roll
22:58:15.079	-105	-83	24308	-8	90	-20
22:58:15.108	-116	31	24260	-8	90	-20
22:58:15.108	-132	13	24263	-8	90	-20
22:58:15.114	-296	-55	24293	-8	90	-20
22:58:15.137	-400	88	23939	-8	90	-20
22:58:15.205	-328	296	23852	-8	90	-20
22:58:15.214	-15	97	23926	-8	90	-20
22:58:15.281	129	-108	24257	-8	90	-20
22:58:15.303	187	36	24300	-9	90	-19
22:58:15.311	214	52	24320	-9	90	-19
22:58:15.344	139	36	24301	-8	90	-19
22:58:15.385	-472	-136	24368	-8	90	-19
22:58:15.416	-4	79	24314	-9	90	-19
22:58:15.448	-444	99	24401	-9	90	-19
22:58:15.480	-639	81	24447	-9	90	-19
22:58:15.512	-193	189	24302	-9	90	-19
22:58:15.532	-109	304	23884	-9	90	-19
22:58:15.588	434	36	24607	-9	90	-19
22:58:15.620	-122	254	24254	-9	90	-19
22:58:15.690	113	493	24388	-9	90	-19
22:58:15.703	573	252	24700	-9	90	-19
22:58:15.737	457	306	24450	-9	89	-19
22:58:15.817	884	669	23625	-9	89	-19
22:58:15.860	742	519	24069	-9	89	-19

ANEXO III

PLANOS



Sheet: /		
File: Sensor_Inercial_TFG.sch		
Title: Placa de conexionado		
Size: A4	Date: 06-12-2021	**Rev:**
KiCad E.D.A. kicad (5.1.12)-1		Id: 1/1

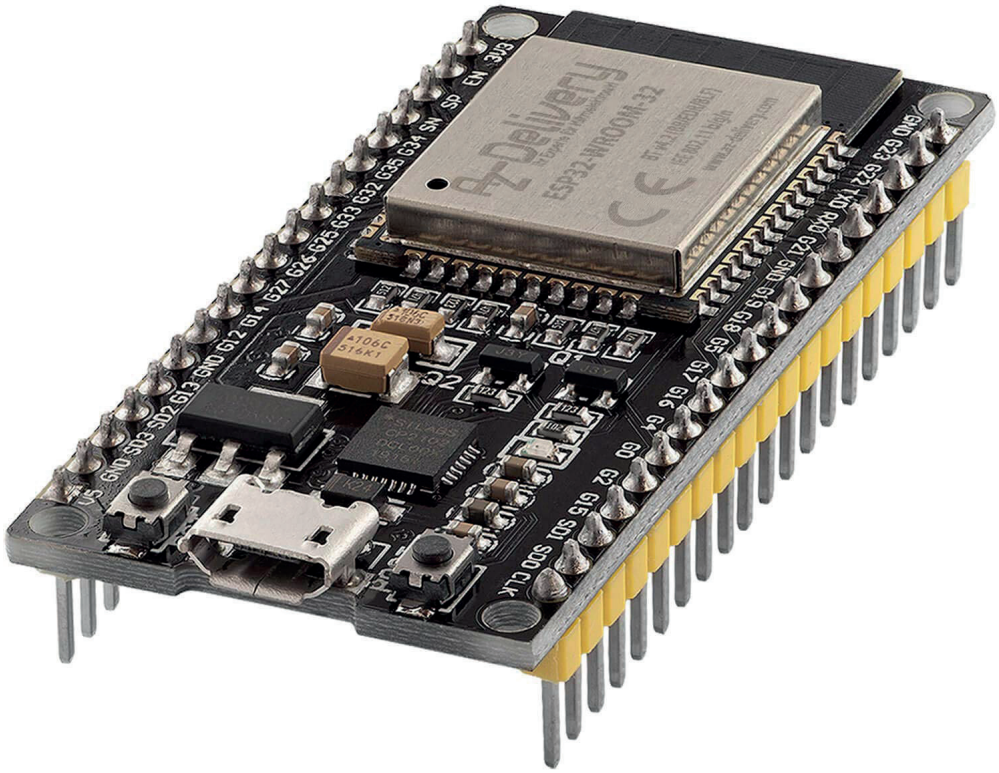

INT01
Conn_01x02
J2
GND
+
Conn_01x13
Conn_01x13
GND
Universidad
de La Laguna
MPU6050
GND
VCC
SCL
SDA
INT

ANEXO VI

DATASHEETS

ESP32 NodeMCU Module WLAN WiFi Development Board mit CP2102

Datenblatt



Content:

1. Features
2. Specifications

1. Features

NodeMCU is an open source IoT platform. ESP32 is a series of low cost, low power system-on-chip (SoC) microcontrollers with integrated Wi-Fi & dual-mode Bluetooth.
The ESP32 series employs a Tensilica Xtensa LX6 microprocessor in both dual-core and single-core variations, with a clock rate of up to 240 MHz. ESP32 is highly integrated with built-in antenna switches, RF balun, power amplifier, low-noise receive amplifier, filters, and power management modules.

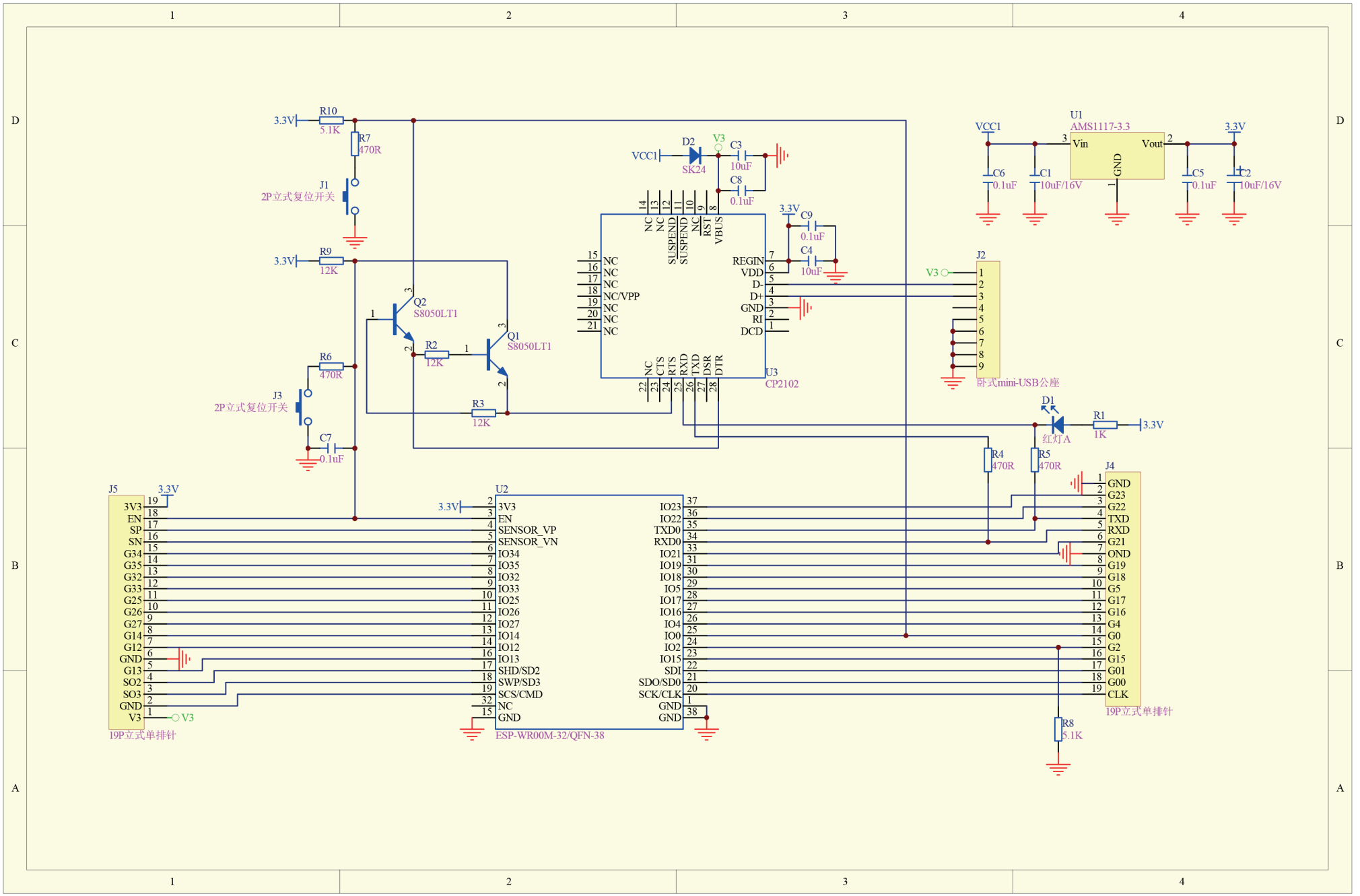
Features:
- Able to achieve ultra-low power consumption.
- Built-in ESP-WROOM-32 chip.
- Breadboard Friendly module.
- Light Weight and small size.
- On-chip Hall and temperature sensor
- Uses wireless protocol 802.11b/g/n.
- Built-in wireless connectivity capabilities.
- Built-in PCB antenna on the ESP32-WROOM-32
- Capable of PWM, I2C, SPI, UART, 1-wire, 1 analog pin.
- Uses CP2102 USB Serial Communication interface module.
- Programmable with ESP-IDF Toolchain, LuaNode SDK supports Eclipse project (C language)

2. Specifications

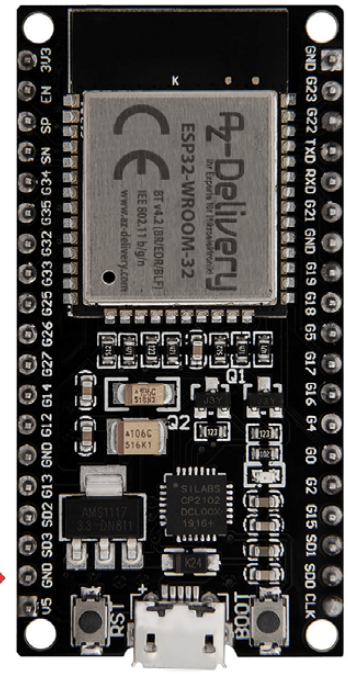
Wireless Standard	FCC/CE/IC/TELEC/KCC/SRRC/NCC
Wireless Protocol	802.11 b/g/n/d/e/l/k/r
Frequency Range	2.4 - 2.5 GHz
Bluetooth Protocol	Bluetooth v4.2 BR/EDR and BLE specification
Bluetooth Specifications	NZIF Receiver with -98dBm sensitiivity Class-1, Class-2 and Class-3 transmitter AFH, CVSD and SBC
Memory	4 MB Flash, 520KB SRAM
Wireless Form	On-board PCB Antenna
IO Capability	UART, I2C, SPI, I2S, PWM, SDIO, GPIO, ADC, DAC
Electrical Characteristic	3.3 V Operated 15 mA output current per GPIO pin 80 mA average working current
Operating Temperature	-40 to +125 ºC
Wireless Network Type	Station / SoftAP / SoftAP + Station / P2P
Security Type	WPA / WPA2 / WPA2-Enterprise / WPS
Encryption Type	AES / RSA / ECC / SHA
Firmware Upgrade	UART Download / OTA / Host
Network Protocol	IPv4, IPv6, SSL, TCP / UDP / FTP / HTTP / MQTT
User Configuration	AT + Order Set, Web Android / iOS, Cloud Server

AZ-Delivery
Ihr Experte für Mikroelektronik!

ESP-32 NodeMCU Developmentboard Schematic



ESP-32 NodeMCU Developmentboard Pinout Diagram



3V3

EN Used by program logic. SW2 grounds via resistor.

GPIO34 ADC1_CH6 RTC_GPIO4

GPIO35 ADC1_CH7 RTC_GPIO5

GPIO32 32K_XP (32.768 kHz osc i/p) ADC1_CH4 TOUCH9 RTC_GPIO9

GPIO33 32K_XN (32.768 kHz osc o/p) ADC1_CH5 TOUCH8 RTC_GPIO8

GPIO25 DAC_1 ADC2_CH8 RTC_GPIO6 EMAC_RXD0

GPIO26 DAC_2 ADC2_CH9 RTC_GPIO7 EMAC_RXD1

GPIO27 ADC2_CH7 TOUCH7 RTC_GPIO17 EMAC_RX_DV

GPIO14 ADC2_CH6 TOUCH6 RTC_GPIO16 MTMS HSPI-CLK HS2_CLK SD_CLK EMAC_TXD2

GPIO12 ADC2_CH5 TOUCH5 RTC_GPIO15 MTDI HSPIQ HS2_DATA2 SD_DATA2 EMAC_TXD3

Gnd

GPIO13 ADC2_CH4 TOUCH4 RTC_GPIO14 MTCK HSPID HS2_DATA3 SD_DATA3 EMAC_RX_ER

GPIO9 SD_DATA2 SPIHD HS1_DATA2 U1RXD

GPIO10 SD_DATA3 SPIWP HS1_DATA3 U1TXD

! CMD

5V

Gnd

GPIO23 VSPID HS1_STROBE

GPIO22 VSPIWP U0RTS EMAC_TXD1

GPIO1 U0TXD CLK_OUT3 EMAC_RXD2

GPIO3 U0RXD CLK_OUT2

GPIO21 VSPIHD EMAC_TX_EN

Gnd

GPIO19 VSPIQ U0CTS EMAC_TXD0

GPIO18 VSPICLK HS1_DATA7

GPIO5 VSPICS0 HS1_DATA6 EMAC_RX_CLK

GPIO17 HS1_DATA5 U2TXD EMAC_CLK_OUT_180

GPIO16 HS1_DATA4 U2RXD EMAC_CLK_OUT

GPIO4 ADC2_CH0 TOUCH0 RTC_GPIO10 HSPIHD HS2_DATA1 SD_DATA1 EMAC_TX_ER

GPIO0 ADC2_CH1 TOUCH1 RTC_GPIO11 CLK_OUT1 EMAC_TX_CLK

GPIO2 ADC2_CH2 TOUCH2 RTC_GPIO12 HSPIWP HS2_DATA0 SD_DATA0 External pulldown

GPIO15 ADC2_CH3 TOUCH3 RTC_GPIO13 MTDO HSPICS0 HS2_CMD SD_CMD EMAC_RXD3

GPIO8 SD_DATA1 SPID HS1_DATA1 U2CTS

GPIO7 SD_DATA0 SPIQ HS1_DATA0 U2RTS

GPIO6 SD_CLK SPICLK HS1_CLK U1CTS

Used for internal flash, not recommended for other use

Input only. No internal pullup or pulldown.

Used by USB/REPL

GPIO0 has a 5KΩ external pullup. SW0 grounds via 470Ω

Used on ESP32-WROVER-KIT etc to access external SPI RAM

ADC: FSD = 4095 = 1.109V (Because 693mV gave 2559. Is the limit 1.0V?)

DAC: FSD = 255 = 3.19V (Vs = 3.3V). 127 gave 1.63V implying 3.3V FS.

Value	Expected	Actual	Error %
10	0.13	0.21	2.4
20	0.26	0.33	2.1
127	1.64	1.63	-0.3
200	2.58	2.53	-1.5
240	3.11	3.01	-3
255	3.3	3.19	-3.3

Remapping peripherals:
uart = machine.UART(1,baudrate=115200,tx=25,rx=26)

ESP32-D2WD is the chip with embedded 2MB flash and the internal flash is connected to different pins (GPIO16, GPIO17, SD_CMD, SD_CLK, SD_DATA_0 and SD_DATA_1)

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		µg/ √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

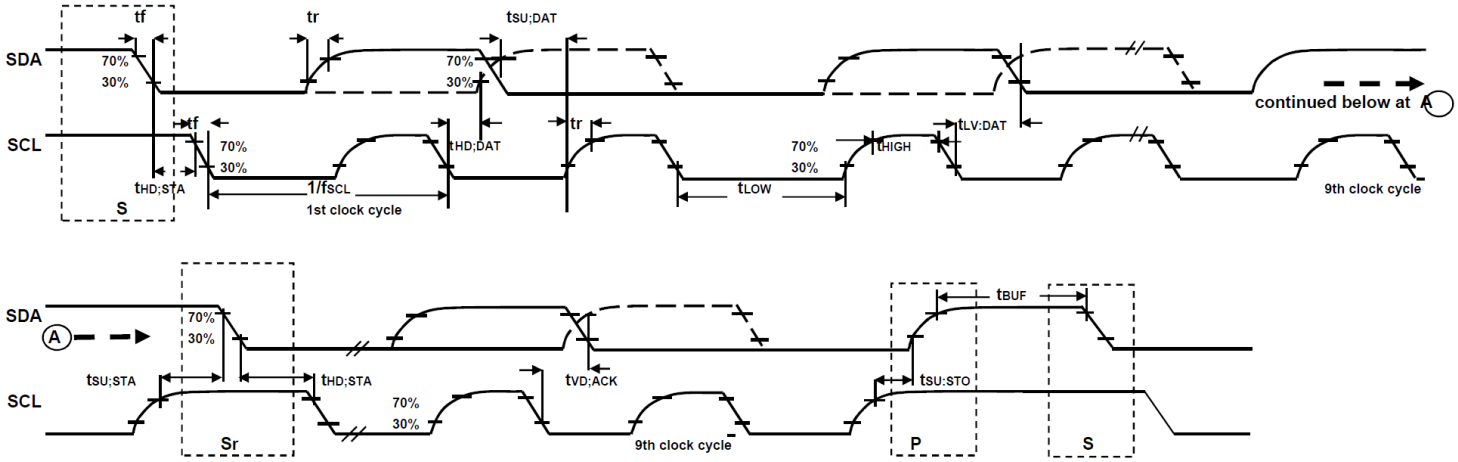
Parameters	Conditions	Min	Typical	Max	Units	Notes
INTERNAL CLOCK SOURCE	**CLK_SEL=0,1,2,3**					
Gyroscope Sample Rate, Fast	DLPFCFG=0 SAMPLERATEDIV = 0		8		kHz	
Gyroscope Sample Rate, Slow	DLPFCFG=1,2,3,4,5, or 6 SAMPLERATEDIV = 0		1		kHz	
Accelerometer Sample Rate			1		kHz	
Clock Frequency Initial 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

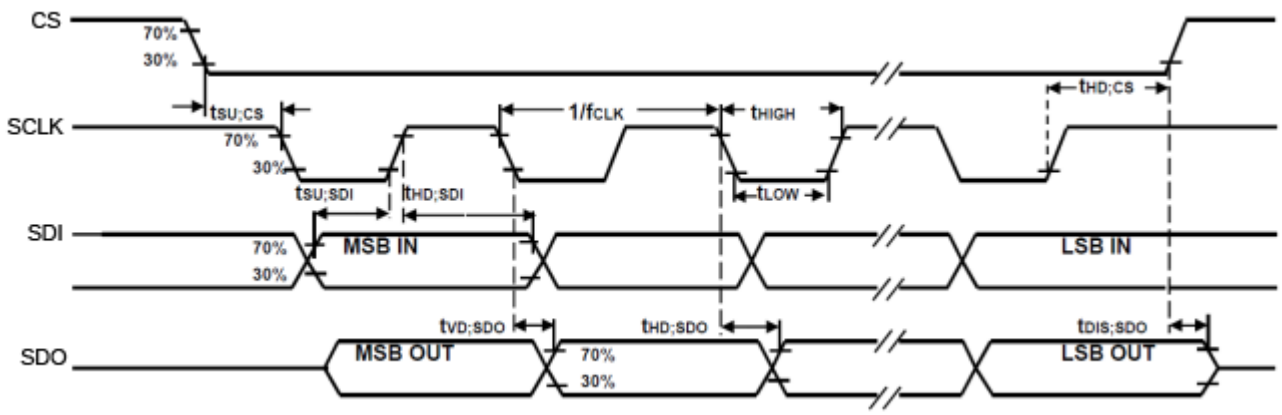


I²C Bus Timing Diagram

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

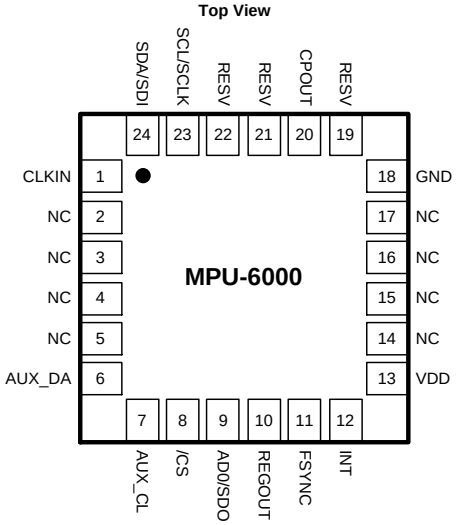
Stress above those listed as “Absolute Maximum Ratings” may cause permanent damage to the device. These are stress ratings only and functional operation of the device at these conditions is not implied. Exposure to the absolute maximum ratings conditions for extended periods may affect device reliability.

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

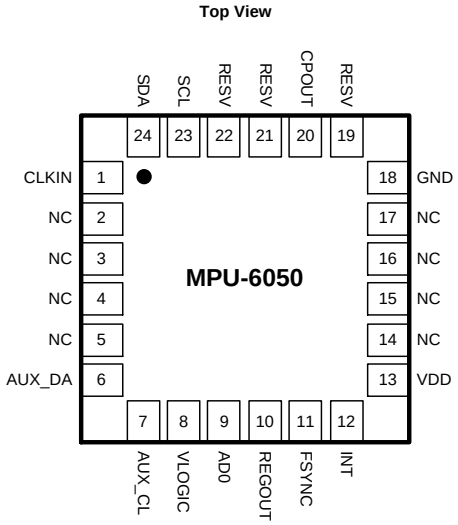
7 Applications Information

7.1 Pin Out and Signal Description

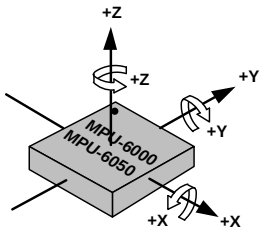
Pin Number	MPU-6000	MPU-6050	Pin Name	Pin Description
1	Y	Y	CLKIN	Optional external reference clock input. Connect to GND if unused.
6	Y	Y	AUX_DA	I^2C master serial data, for connecting to external sensors
7	Y	Y	AUX_CL	I^2C Master serial clock, for connecting to external sensors
8	Y		/CS	SPI chip select (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

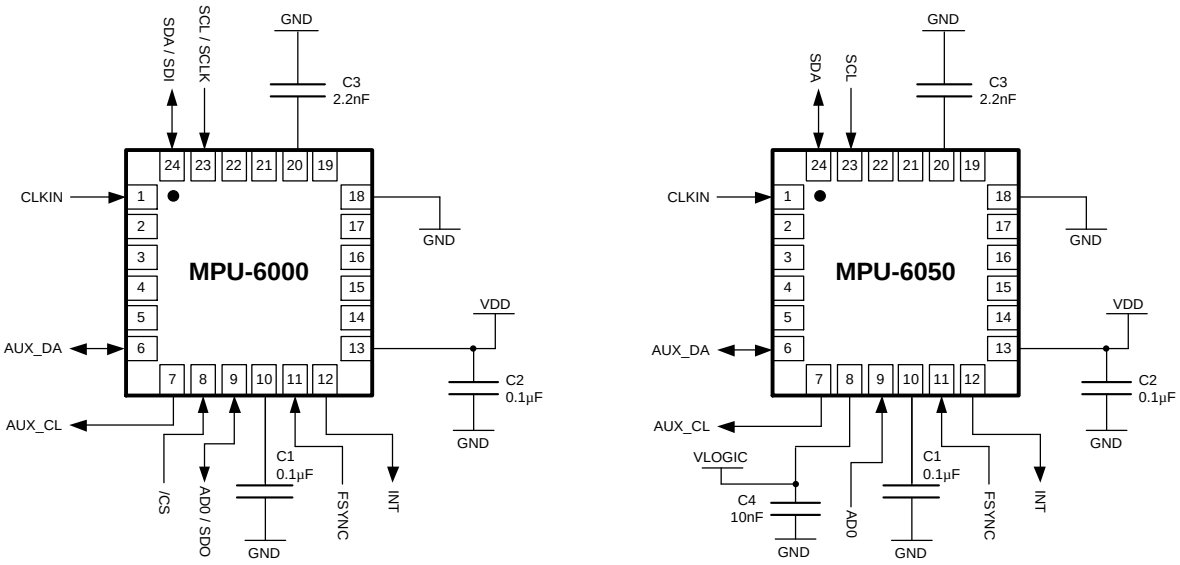


QFN Package
24-pin, 4mm x 4mm x 0.9mm



Orientation of Axes of Sensitivity and Polarity of Rotation

7.2 Typical Operating Circuit



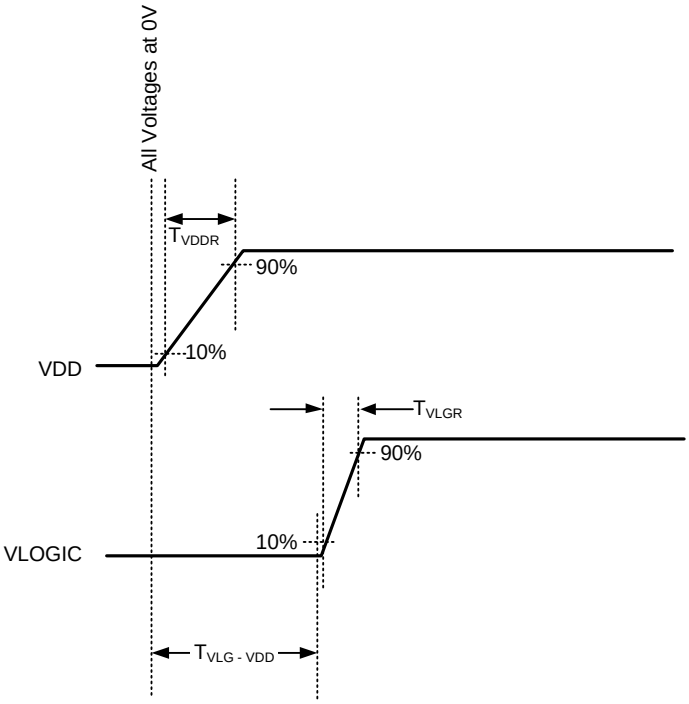
Typical Operating Circuits

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.

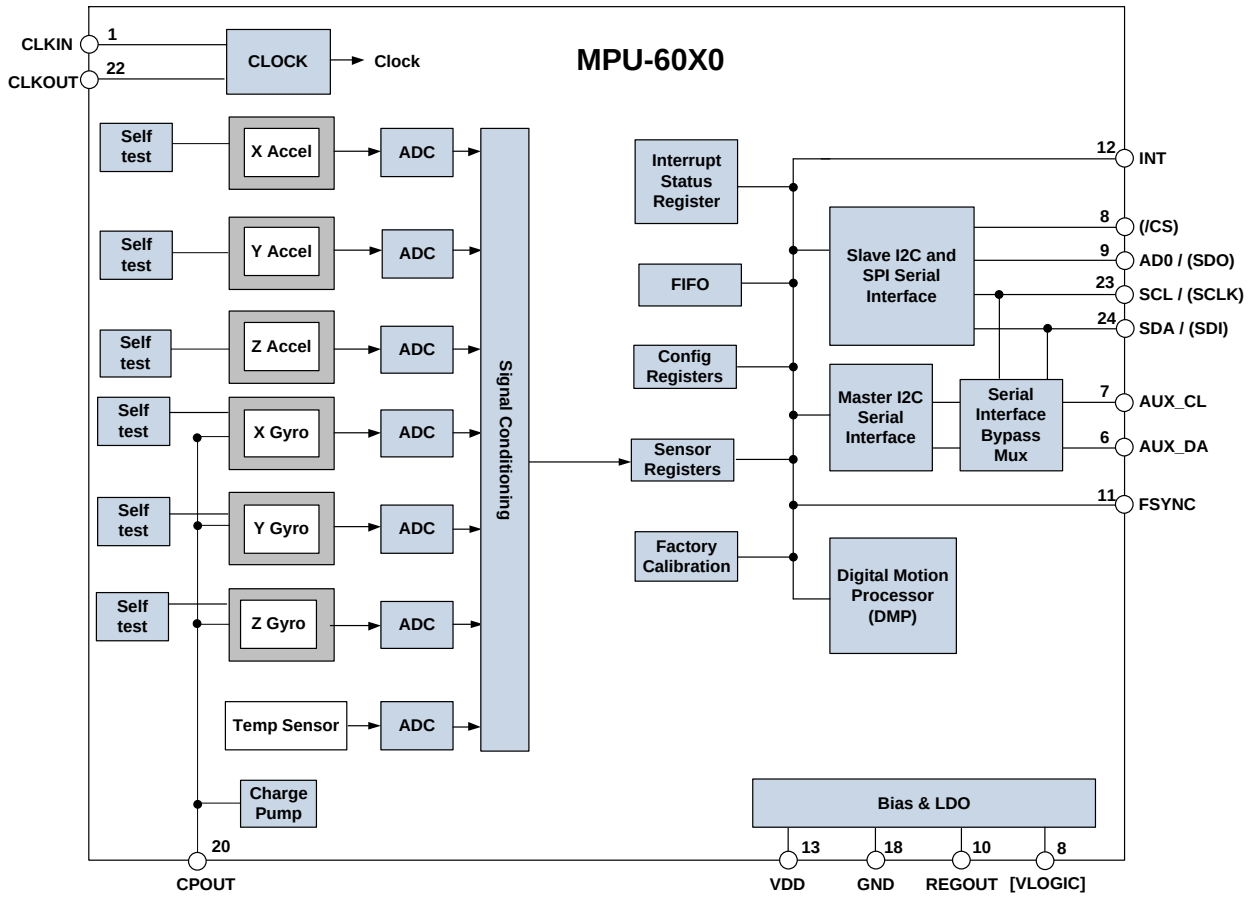
7.4 Recommended Power-on Procedure



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 I^2C Serial Interface

The MPU-60X0 has an auxiliary I^2C bus for communicating to an off-chip 3-Axis digital output magnetometer or other sensors. This bus has two operating modes:

- I^2C Master Mode: The MPU-60X0 acts as a master to any external sensors connected to the auxiliary I^2C bus
- Pass-Through Mode: The MPU-60X0 directly connects the primary and auxiliary I^2C buses together, allowing the system processor to directly communicate with any external sensors.

Auxiliary I^2C Bus Modes of Operation:

- I^2C 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 I^2C 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 I^2C 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 I^2C bus pins (AUX_DA and AUX_CL). In this mode, the auxiliary I^2C bus control logic (3^{rd} party sensor interface block) of the MPU-60X0 is disabled, and the auxiliary I^2C pins AUX_DA and AUX_CL (Pins 6 and 7) are connected to the main I^2C 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 I^2C interface.

Auxiliary I^2C Bus IO Logic Levels

- MPU-6000: The logic level of the auxiliary I^2C bus is VDD
- MPU-6050: The logic level of the auxiliary I^2C 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).

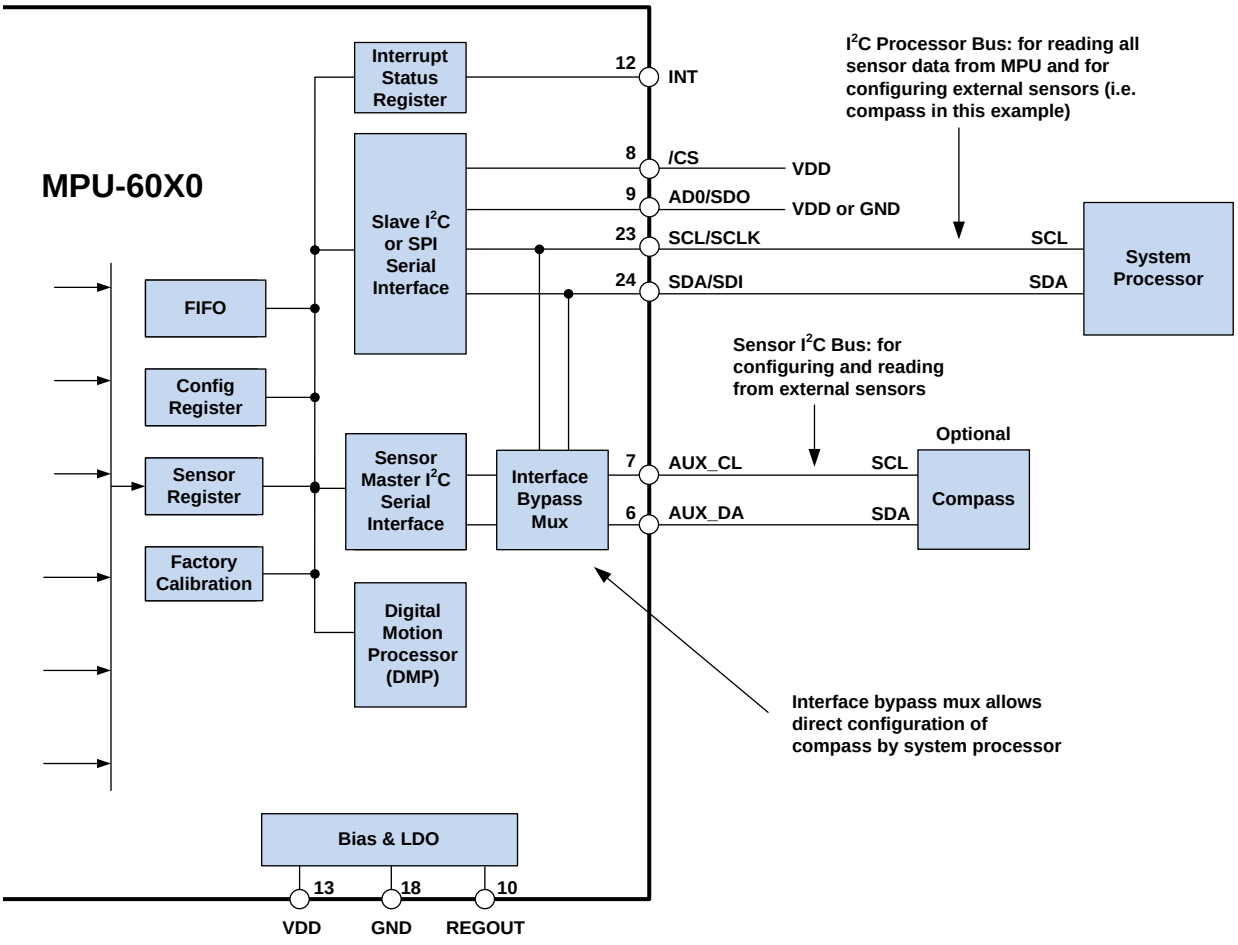
When the value of the self-test response is within the min/max limits of the product specification, the part has passed self test. When the self-test response exceeds the min/max values, the part is deemed to have failed self-test. Code for operating self test code is included within the MotionApps software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

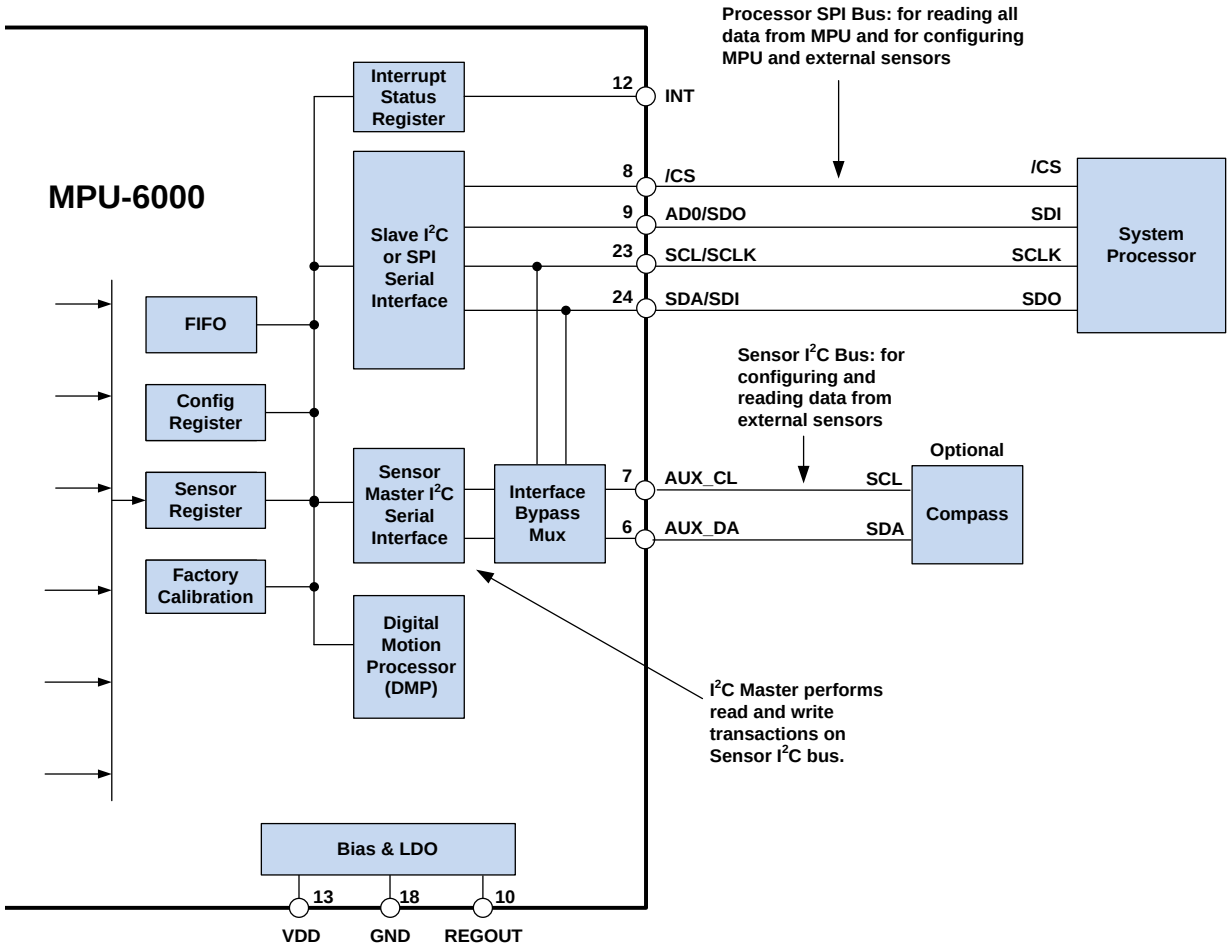
In the figure below, the system processor is an SPI master to the MPU-6000. Pins 8, 9, 23, and 24 are used to support the /CS, SDO, SCLK, and SDI signals for SPI communications. Because these SPI pins are shared with the I^2C slave pins (9, 23 and 24), the system processor cannot access the auxiliary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

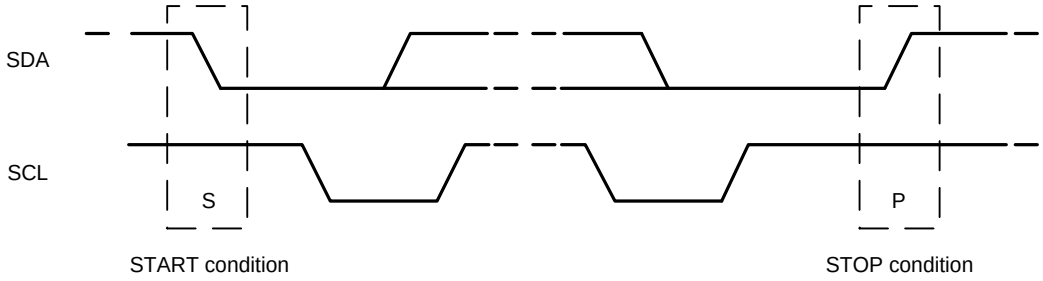
The slave address of the MPU-60X0 is b110100X which is 7 bits long. The LSB bit of the 7 bit address is determined by the logic level on pin AD0. This allows two MPU-60X0s to be connected to the same I²C bus. When used in this configuration, the address of the one of the devices should be b1101000 (pin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

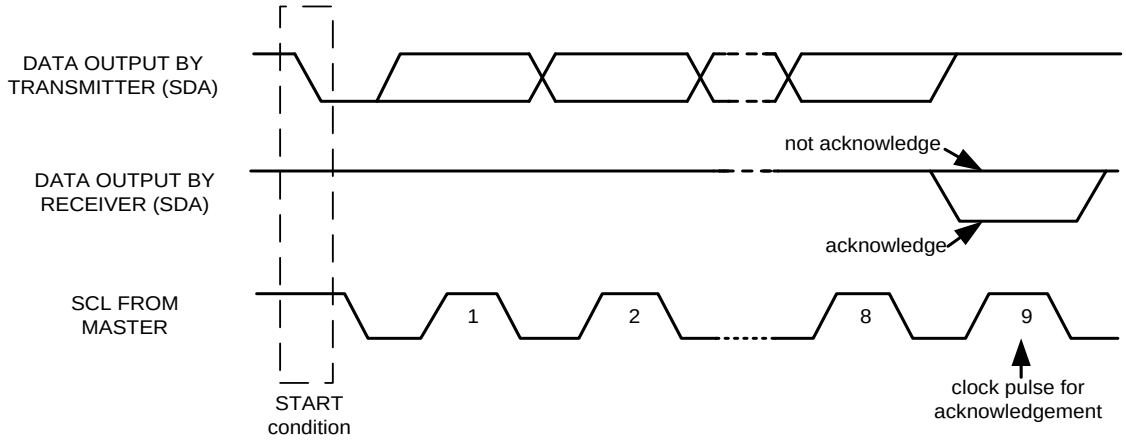


START and STOP Conditions

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.

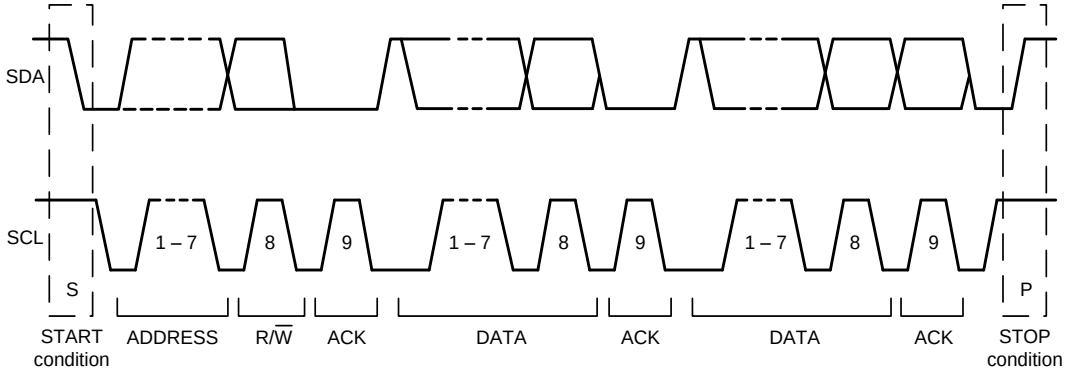
If a slave is busy and cannot transmit or receive another byte of data until some other task has been performed, it can hold SCL LOW, thus forcing the master into a wait state. Normal data transfer resumes when the slave is ready, and releases the clock line (refer to the following figure).



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 9th 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 9th clock cycle
NACK	Not-Acknowledge: SDA line stays high at the 9th 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:

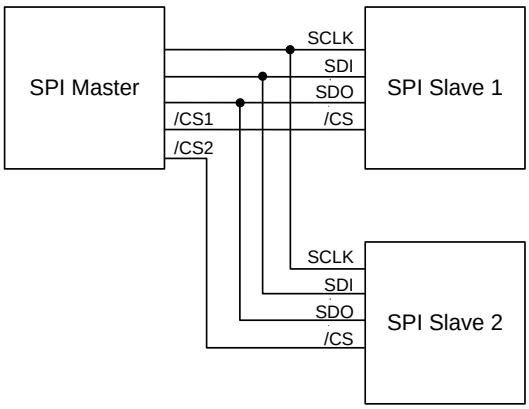
SPI Address format

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*

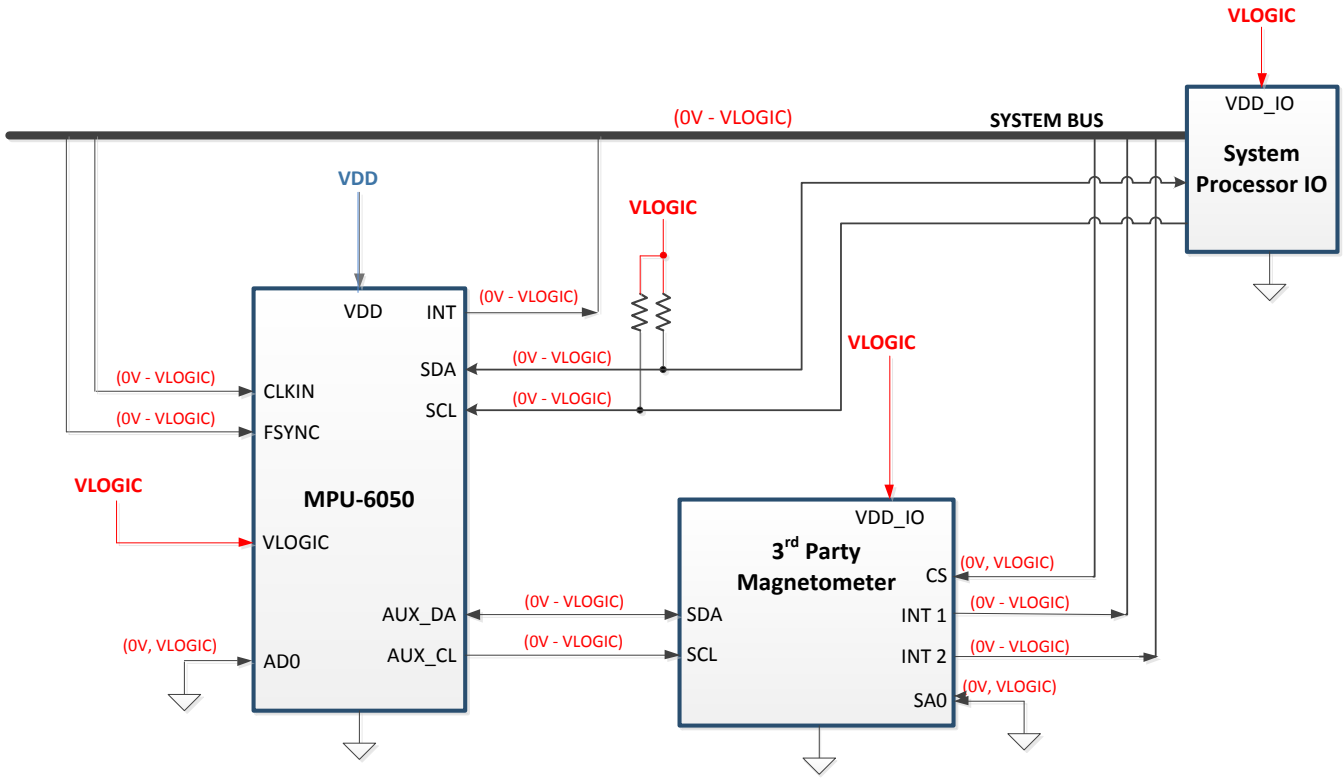
AUX_VDDIO	**MICROPROCESSOR LOGIC LEVELS** (Pins: SDA, SCL, AD0, CLKIN, INT)	**AUXILLARY LOGIC LEVELS** (Pins: AUX_DA, AUX_CL)
0	VLOGIC	VLOGIC

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:

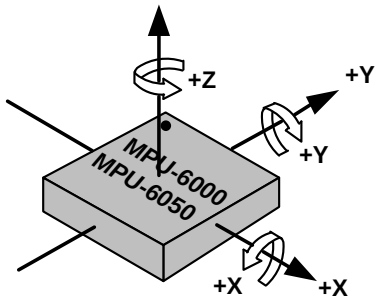
1. AUX_VDDIO determines the IO voltage levels of AUX_DA and AUX_CL (0 = set output levels relative to VLOGIC)
2. All other MPU-6050 logic IOs are referenced to VLOGIC.

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

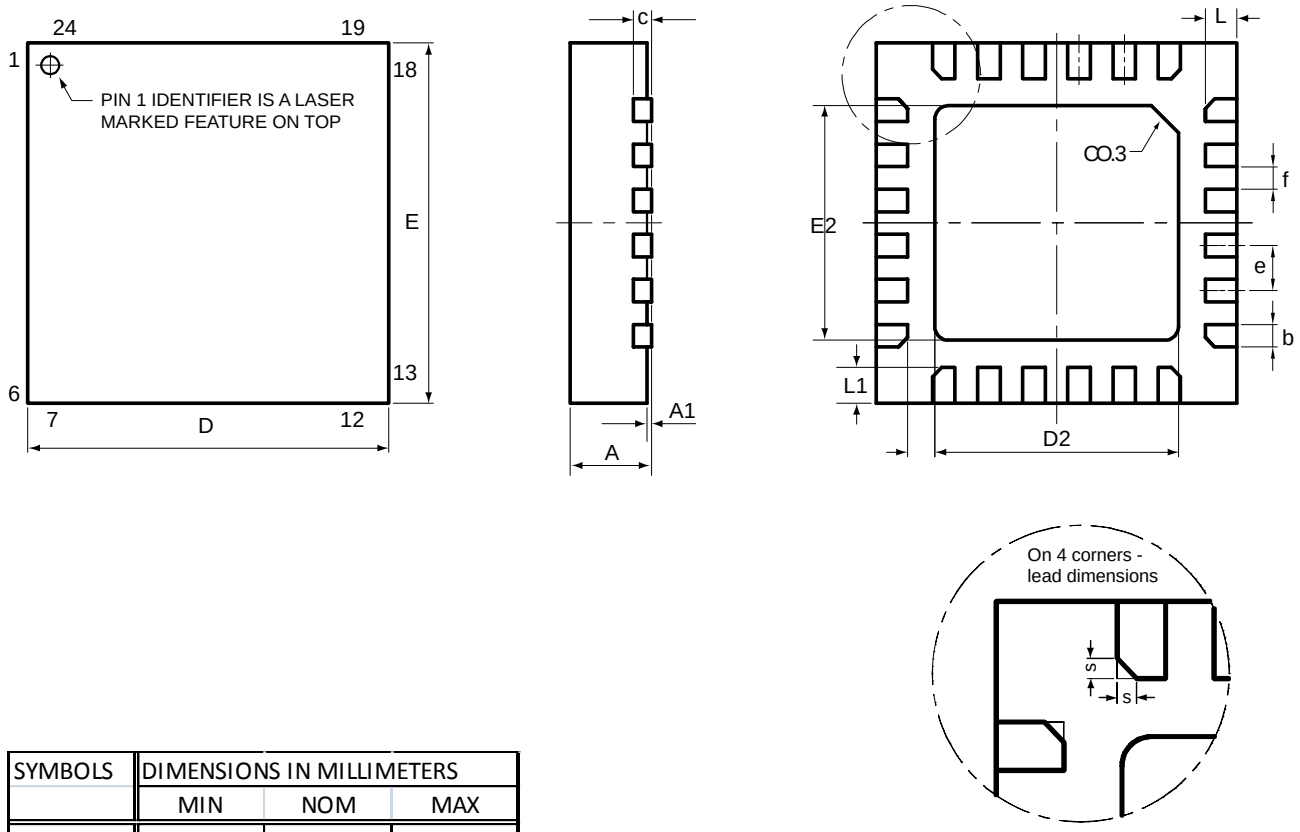
The diagram below shows the orientation of the axes of sensitivity and the polarity of rotation. Note the pin 1 identifier (•) in the figure.



Orientation of Axes of Sensitivity and Polarity of Rotation

11.2 Package Dimensions

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

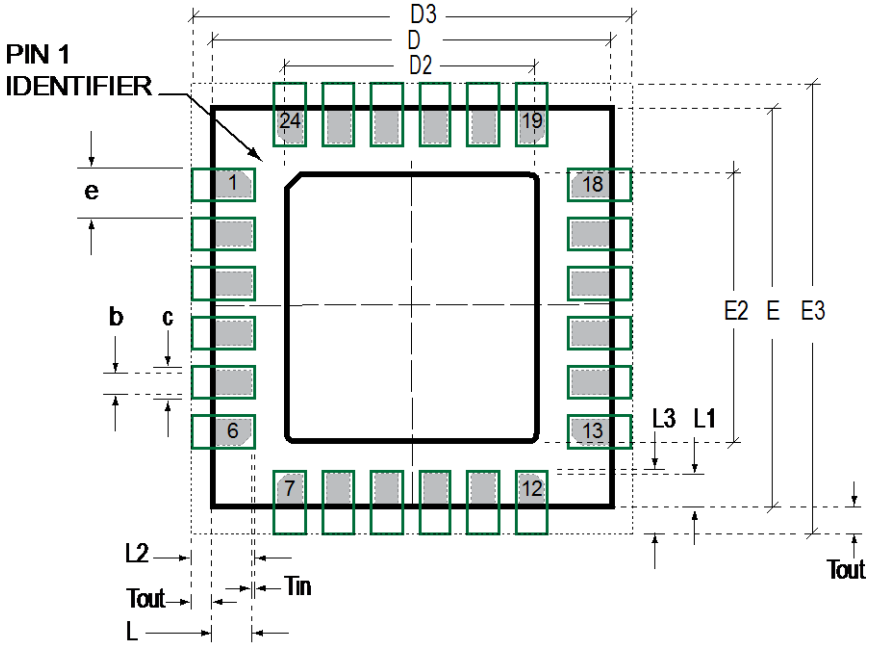


SYMBOLS	DIMENSIONS IN MILLIMETERS		
	MIN	NOM	MAX
A	0.85	0.90	0.95
A1	0.00	0.02	0.05
b	0.18	0.25	0.30
c	---	0.20 REF	---
D	3.90	4.00	4.10
D2	2.65	2.70	2.75
E	3.90	4.00	4.10
E2	2.55	2.60	2.65
e	---	0.50	---
f (e-b)	---	0.25	---
K	0.25	0.30	0.35
L	0.30	0.35	0.40
L1	0.35	0.40	0.45
s	0.05	---	0.15

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

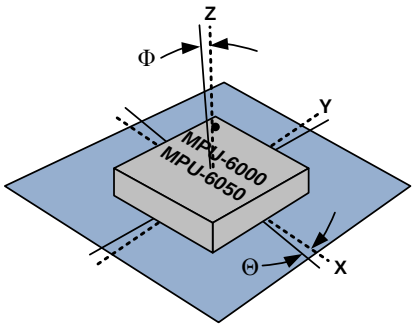
Routing traces or vias under the gyro package such that they run under the exposed die pad is prohibited. Routed active signals may harmonically couple with the gyro MEMS devices, compromising gyro response. These devices are designed with the drive frequencies as follows: X = 33±3Khz, Y = 30±3Khz, and 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.

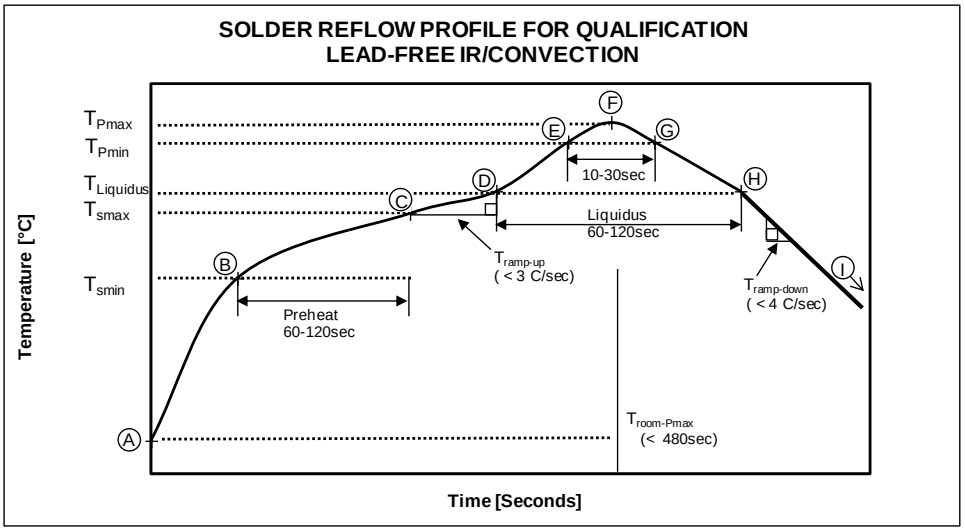
Restrict all device handling to ESD protected work areas that measure less than 200V static charge. Ensure that all workstations and personnel are properly grounded to prevent ESD.

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.

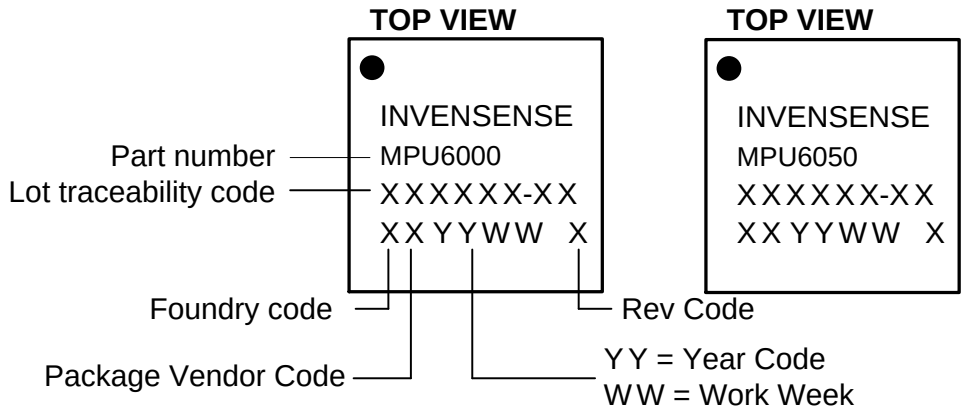
Production Reflow: Check the recommendations of your solder manufacturer. For optimum results, use lead-free solders that have lower specified temperature profiles (Tp_{max} ~ 235°C). Also use lower ramp-up and ramp-down rates than those used in the qualification profile. Never exceed the maximum conditions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

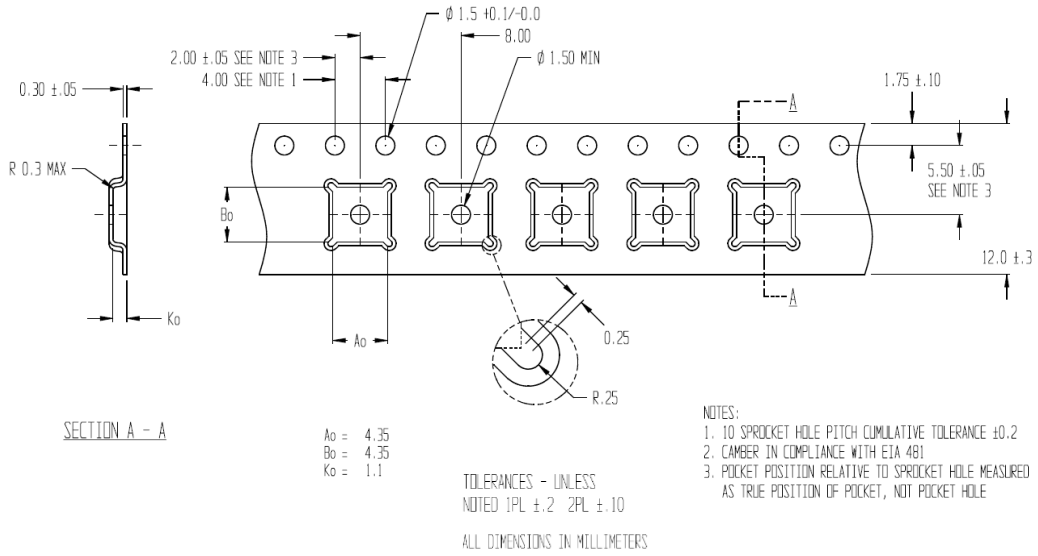
Calculated shelf-life in moisture-sealed bag	12 months -- Storage conditions: <40°C and <90% RH
After opening moisture-sealed bag	168 hours -- Storage conditions: ambient ≤30°C at 60%RH

11.6 Package Marking Specification

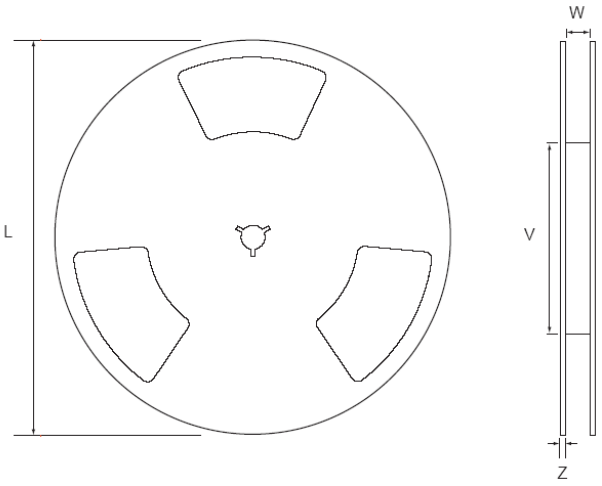


Package Marking Specification

11.7 Tape & Reel Specification



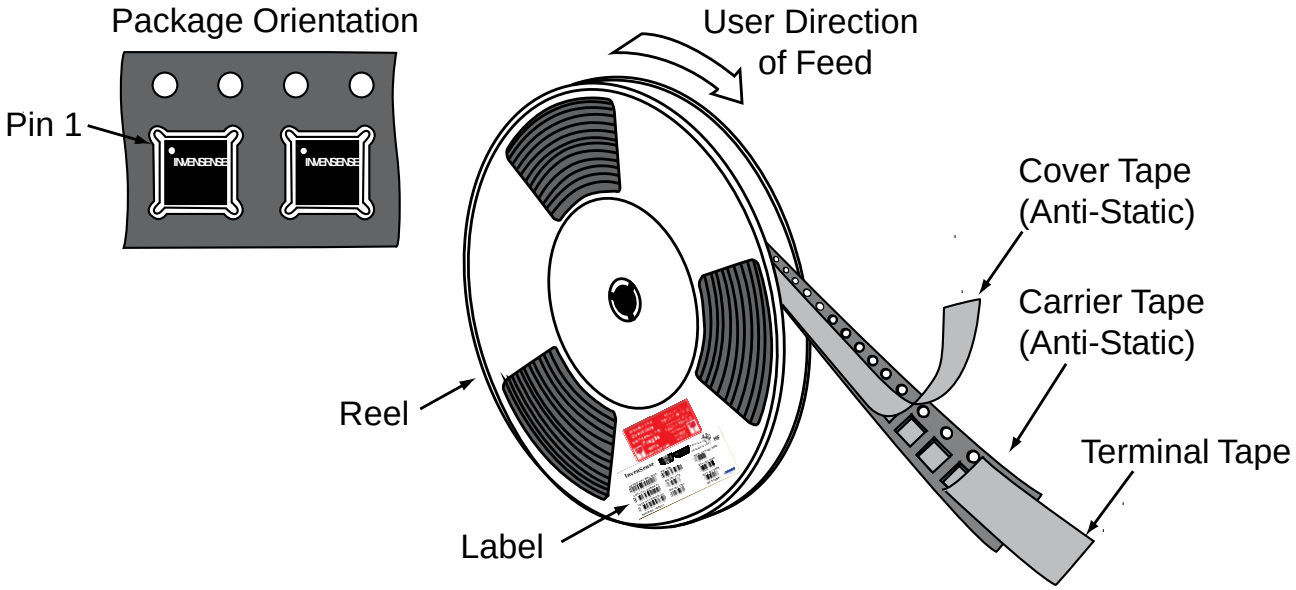
Tape Dimensions



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

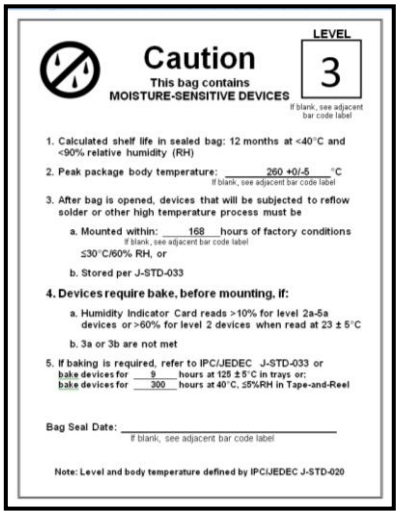
Barcode Label

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

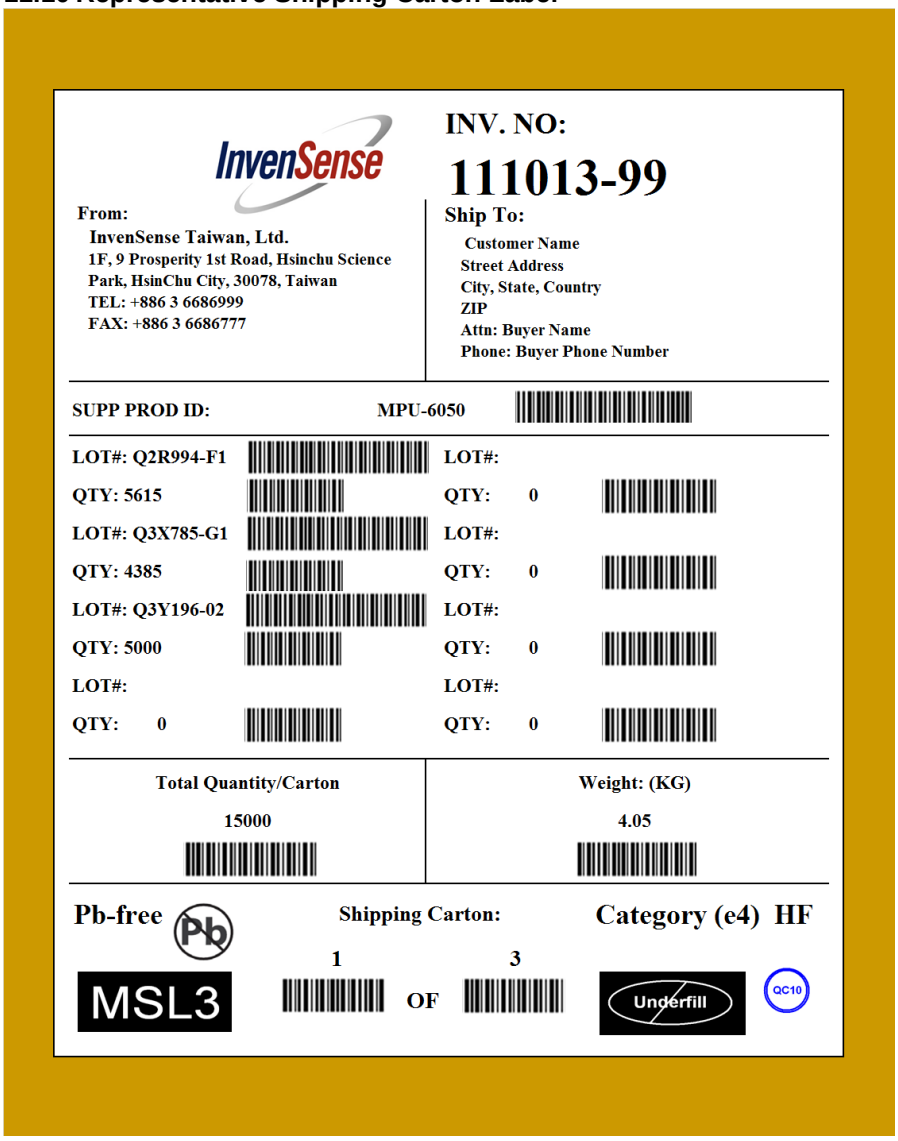
Inner Bubble Wrap

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.