ESCUELA SUPERIOR DE INGENIERÍA Y TECNOLOGÍA

Trabajo de Fin de Grado

Monitorización de parámetros biomédicos y ambientales de un paciente en aislamiento y con movilidad reducida en una habitación

Titulación: Grado en Ingeniería Electrónica Industrial y Automática
Autor: Raúl González González
Tutor: José Ignacio Estévez Damas
Julio, 2021

Me gustaría agradecer a mi tutor José Ignacio Estévez Damas el apoyo demostrado en mi proyecto desde su comienzo y el haberme guiado en el desarrollo del mismo.

Agradecer a mi familia el apoyo que siempre me han demostrado en todas mis decisiones.

Índice:

1.- Introducción

1.1- Motivación del trabajo

La idea de este proyecto surge como consecuencia de las nuevas circunstancias con las que nos hemos visto obligados a convivir, y que nos han cambiado nuestros hábitos y costumbres. Viendo por los medios de comunicación los problemas que han surgido para poder atender a las personas más vulnerables (póngase como ejemplo más drástico el día a día de un servicio de U.V.I. en los hospitales), surge también el problema, de forma más simple pero no menos importante, la atención a una persona que debe quedarse aislada en una habitación por estar infectada y no poder tener contacto con el resto de sus convivientes. Si a esto le añadimos que esta persona por su edad o por problemas de movilidad no se puede valer por sí misma, el caso se complica todavía más.

Como la base de la Ingeniería es darle una solución tecnológica a los problemas, como futuro ingeniero creo que puedo contribuir desarrollando un proyecto que plantee soluciones para ayudar al cuidado y vigilancia de estas personas. Con este objetivo, este trabajo de fin de grado presenta la implementación de un sistema que sea capaz de monitorizar una serie de parámetros, tanto de la habitación donde se encuentre aislado, como biomédicos (temperatura corporal, saturación de oxígeno y pulso), que se puedan leer de forma remota y con la posibilidad de actuar sobre algunos aparatos que se encuentren dentro de la habitación para evitar en lo posible el tener que acceder con las medidas de protección que esta situación requiere. También se ha implementado un sistema de vigilancia con cámara para tener la posibilidad de ver lo que ocurre dentro de la habitación y un botón de ayuda de emergencia del que dispondrá la persona enferma.

Se ha intentado utilizar componentes que se puedan conseguir de forma sencilla en el mercado y que sean económicos para que resulte un sistema accesible para la mayoría de las personas. Los componentes específicos que se utilizan en equipamientos electromédicos resultan demasiado caros para plantear su incorporación al proyecto. Por lo tanto los sensores que se han utilizado para los parámetros biomédicos (que se detallan más adelante), no resultan eficaces para un diagnóstico médico pero sí para tener una ayuda en el cuidado de la persona enferma.

Hoy en día hay mucha información sobre teleasistencia, telemedicina, telesalud, etcétera. En concreto es impresionante el desarrollo que se ha conseguido en este campo con la ayuda de la tecnología denominada Internet of Things (IoT) ya que se ha abierto un gran campo de posibilidades gracias a la capacidad de montar dispositivos inteligentes capaces de aprovechar los servicios de Internet y que monitorizan de forma constante y en cualquier parte a los pacientes que necesitan llevar un seguimiento para el control de distintas patologías. El campo de la telemedicina ha dado un salto importante gracias a este tipo de tecnología.

"*El Internet de las cosas es considerado por muchos como la cuarta revolución industrial, pero a diferencia de las primeras no es una tecnología nueva, es una nueva forma de combinar tecnologías existentes. La forma más sencilla de definir el IoT es pensar en ella como una red de sistemas interconectados. Estos sistemas suelen ser soluciones tecnológicas complejas, diseñadas para abordar una aplicación específica. A estas soluciones se les denomina productos IoT, y cuando interconectan millones de estos productos se obtiene el Internet de las cosas.*" [Daniel Elizalde, *¿Qué es el Internet de las cosas?*: https://danielelizalde.com/what-is-the-internet-of-things/] .

Durante el desarrollo del proyecto se ha observado que el sistema ofrece muchas posibilidades de desarrollo en todos sus elementos, por lo que el resultado final sería la base para futuras ampliaciones tanto a nivel de registro de datos con sensores nuevos, una aplicación más compleja y multiplataforma, otras formas de actuación sobre los elementos ambientales usando el gran abanico de posibilidades que ofrece la Domótica, y la posibilidad de hacer que el sistema sea ampliable a varios enfermos aislados con un sistema de control y monitorización centralizado.

El montaje final consta de dos módulos. El primero será un módulo de sensado con un diseño compacto basado en una caja donde está instalado el microcontrolador y las conexiones necesarias para los sensores. La caja del módulo de sensado irá ubicada cerca de la cabecera del paciente, para intentar hacer los cables lo más cortos posibles. El segundo módulo, que denominaremos módulo de monitorización, irá en la entrada de la habitación, y estará conectado vía Wifi y por cable con comunicación serial (sobre USB), con el módulo de sensado.

1.2- Resumen

La idea inicial de este proyecto, era desarrollar un dispositivo cuyo principal objetivo era ayudar en el cuidado y la atención de personas mayores. Teniendo en cuenta lo vulnerables y aislados que se encuentran, y con las dificultades añadidas en su cuidado debido al confinamiento y la enfermedad, decidí crear un sistema de monitorización remota para poder tener los valores tanto ambientales como biométricos básicos necesarios, y saber si se encuentran bien o necesitan ayuda.

Tratando de buscar una solución a los problemas anteriormente descritos, parece necesario comenzar con el desarrollo de un sistema de sensores que se encargue de la recopilación de datos y el envío fuera de la habitación para su evaluación. Surge la definición de la tecnología IoT como solución a la interconexión de sensores y envío de datos como la mejor opción para desarrollar el proyecto.

Además, será utilizada para actuar sobre algunos dispositivos de la habitación como se explicará más adelante.

Como parte del sistema de monitorización, se ha diseñado una aplicación que lee los datos procedentes de las mediciones realizadas por los sensores, y los muestra en pantalla. Además, los almacena también en una base de datos para su estudio posterior si fuese necesario. Dicha aplicación nos mostrará las alarmas que se generan cuando los valores están fuera de los márgenes establecidos previamente.

Durante el desarrollo del trabajo se ha añadido la posibilidad de configurarlo para varias personas aisladas en diferentes habitaciones.

1.3- Abstract

The initial idea of this project was the development of a device whose main aim was to support the care and attention of the elderly. Taking into account the vulnerability and isolation in which they are involved, in addition to the difficulties associated with their confinement and diseases, I decided to create a remote monitoring system in order to get the required basic environmental and biometric data, as well as, know their mood or if they need help.

Trying to find a solution to the problems described above, it seems necessary to start with the development of a sensor system that takes care of collecting data and sending it out of the room for evaluation.

The definition of IoT technology arises as a solution to the interconnection of sensors and sending data as the best option to develop the project.

In addition, it will be used to act on some devices in the room as will be explained later.

As part of the monitoring system, an application has been designed that reads the data from the measurements made by the sensors, and displays it on the screen. It also stores them in a database for further study if necessary. This application will show us the alarms that are generated when the values are outside of the previously established margins.

During the development of the work, it has been added the possibility of configuring it for several isolated people in different rooms.

1.4- Esquema de funcionamiento

El sistema está compuesto por dos módulos principalmente, con funciones específicas para cada uno de ellos e interconectados entre sí. Cada módulo ha llevado un desarrollo independiente, ya que se ha tenido que estudiar los componentes que se necesitan y la parte de desarrollo del software específico de cada uno de ellos. Se ha utilizado como base para el desarrollo la tecnología de comunicación IoT que nos permitirá el intercambio de datos a través de la nube y la posibilidad de acceder a ellos de forma remota.

El trabajo realizado comenzó con el desarrollo de un módulo de sensado donde se van a tomar las muestras a partir de los sensores que se implementan en el proyecto y que se detallarán más adelante. Para este módulo se ha utilizado una placa microcontroladora que incorpora una conexión WIFI para el envío y recepción tanto de valores de los sensores como de las señales de control. Ademá se ha definido otro módulo independiente físicamente del primero que podría ubicarse fuera de la habitación, dado que el paciente podría requerir aislamiento, para la monitorización in situ, y al que se podrían añadir más dispositivos como pantalla y cámara para video-llamada con el paciente, sistema de sonido con altavoces y micrófono, etcétera.

Este montaje nos obligaría a conectarnos remotamente desde fuera de la habitación. Aquí nos encontramos entonces con dos prestaciones que hay que conjugar: flexibilidad en las comunicaciones y robustez. La flexibilidad nos la puede ofrecer una conexión WiFi para ambos módulos y una infraestructura que les permita conectarse a Internet e intercambiar información. Sin embargo, así dependeremos de la Wifi para las comunicaciones provocando un punto de debilidad en el sistema. Una alternativa a esto que disminuye la flexibilidad pero aumenta la robustez pasaría por instalar el módulo de monitorización fuera de la habitación y hacer una conexión por cable con el módulo de los sensores, para asegurar que siempre haya envío de datos. La solución propuesta en este trabajo conjuga ambas alternativas.

Con estas ideas, se ha elegido para el prototipo del segundo módulo una placa microprocesadora (Raspberry Pi 4) la cual soportará la aplicación que controlará los dispositivos y donde se visualizarán los valores de los sensores así como la webcam para monitorizar el estado del paciente.

El sistema, una vez esté funcionando, debe ser capaz de enviar una alarma a través de la aplicación cuando alguno de los valores que están siendo registrados por los sensores supere los valores prefijados para alguna de las patologías más comunes en este tipo de situaciones (fiebre, taquicardias, bradicardias, u otros). El proyecto es un prototipo y por tanto no están considerados todos los parámetros necesarios para una supervisión completa

del paciente. Se podría añadir en el caso de haber algún dispositivo médico instalado dentro de la habitación, algún sensor más, que nos pueda dar información del correcto funcionamiento de éste (se puede poner como ejemplo un respirador). Una vez que se genere una alarma, tendremos la posibilidad conectándonos a la aplicación, de ver el resto de los valores que se están registrando y actuar si es necesario (bien encendiendo una luz, activando un sistema de ventilación o climatización, visualizando al paciente a través de la cámara, etc.).

El montaje se muestra en la figura 1 donde podemos ver a la izquierda, las distintas opciones de monitorización y acceso remoto, en el centro de la imagen fuera de la habitación encontramos el módulo de monitorización, a la derecha, el módulo de sensado dentro de la habitación con los diferentes accesorios conectados al módulo, y en el centro con la imagen de la nube, la comunicación entre ambos módulos utilizando la tecnología IoT.

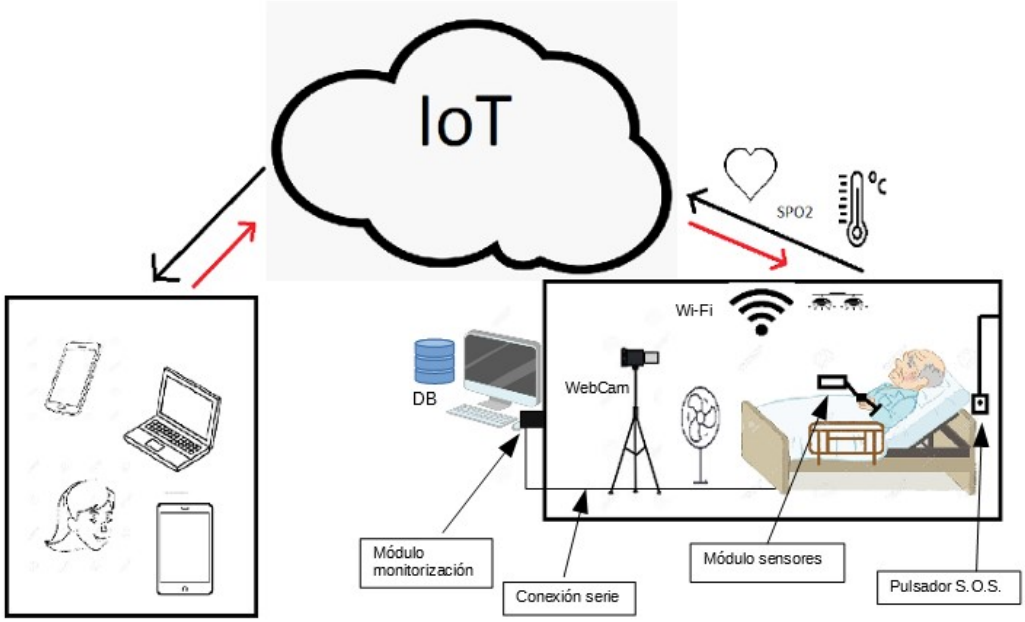


Figura 1 Esquema de funcionamiento

1.5- Antecedentes

Durante la época de la pandemia, he pensado en cómo fabricar un sistema capaz de poder ayudar a las personas que pasan por este tipo de enfermedad, dada la complejidad en la atención de estos pacientes pensé en un montaje que permitiera la asistencia en el hogar de las personas que necesitaban quedarse aisladas y no podían ser atendidas. Tenía que desarrollar un método que fuese casero para su implementación en los hogares. Durante la búsqueda de información localicé varios artículos interesantes que hablaban sobre sistemas similares que se están desarrollando hoy en día y que han tenido un gran empuje debido al COVID-19. Un ejemplo de estos sistemas son el nCapp (Programa Asistente de Diagnóstico y Tratamiento Inteligente del COVID-19) [15].

En este tipo de sistemas es importante el uso de IoT en el seguimiento del estado del paciente una vez se ha dado de alta del centro hospitalario. El prototipo propuesto nos ayudaría a monitorizar los valores necesarios para el control de estos pacientes en sus domicilios, y gracias a la tecnología de IoT que hace posible el envío de datos desde cualquier parte, lo que evitaría en muchos casos un reingreso hospitalario por falta de control y seguimiento.

La justificación del uso de IoT en este proyecto es establecer un modelo orientado a la toma de información a través de sensores "inteligentes" con el soporte de las tecnologías de comunicación para su almacenamiento y posterior uso en el campo de la medicina y biología dando lugar al llamado IoT médico (MIoT).

A diferencia del prototipo propuesto para el proyecto, nCapp es usado para la clasificación de diagnósticos gracias a los datos existentes, los cuestionarios y los resultados de las verificaciones, además de guiar en el tratamiento. Usando como base los últimos datos de casos del mundo real, nCapp mejora la precisión en el tratamiento.

Tomando como idea el sistema en [15], la idea del proyecto surge para crear una fuente de datos de la que se nutriera un sistema similar al nCapp, si se instalaran dispositivos del tipo propuesto en el proyecto, ampliaremos la cobertura del seguimiento de los pacientes con COVID-19 no solo los que estén ingresados en centros de salud u hospitales sino todos los que estén aislados en sus domicilios.

El confinamiento ha sido una de las principales estrategias para evitar la propagación de la pandemia producida por el COVID-19. Esta estrategia encerró a las personas en sus pisos y casas produciendo un cambio de hábito y su adaptación como se describe en la publicación. [20]

Según el artículo [25], la informática de punta (o también llamado computación de borde, consiste en el desarrollo de la programación del punto final para el proceso de datos críticos en lugar de enviarlos a la nube [26]) se necesita en mayor medida para mejorar la potencia de análisis donde se puede mejorar la rapidez de diagnóstico. Por ejemplo en las ambulancias, los hogares particulares, nuestros propios cuerpos, etcétera.

La transformación digital en la atención médica tiene por objetivo principal brindar tratamientos a los pacientes en sus domicilios particulares para aumentar la capacidad de atención y reducir gastos de hospitalización. Para ello se da la idea de la telemedicina, uno de los puntos brillantes surgidos en la pandemia, para manejar de manera segura y eficaz el 75% de todas las visitas médicas y atenciones de urgencia según la Asociación Médica Estadounidense y el Consejo de Bienestar de América. [25]

A medida que aumenta el número de visitas de telesalud, los dispositivos de borde adquieren mayor importancia para la mejora de atención médica por lo que el desarrollo de este tipo de dispositivos mejoraría sustancialmente la capacidad de diagnóstico. Por ejemplo: sensores de movimiento que usan señales de WiFi, rastreadores de sueño, estetoscopios digitales, etcétera. [25]

El auge de todo este desarrollo tecnológico ha sido el motivo de este proyecto.

2.- Estudio y localización de componentes hardware y software

Antes de darle forma al proyecto se investigaron las opciones disponibles en el mercado para conseguir los componentes que necesitaba para hacer pruebas con ellos, y a medida que los iba consiguiendo, verificar que funcionaban correctamente. Algunos de los sensores que se utilizan en el proyecto tuve que pedirlos a través de plataformas como AliExpress o Amazon ya que no los conseguí en las tiendas de electrónica de la zona.

2.1- Microcontrolador para el módulo de sensado

Entre las diferentes placas microcontroladores o también llamadas SoC (System on a Chip) que se pueden conseguir en el mercado para implementar proyectos sobre IoT, destaca de forma clara por sus características la ESP32 que ha sido diseñado por la compañía china Espressif y fabricado por TSMC. Como se busca un controlador con conectividad, potencia, bajo consumo, etc., en este tipo de dispositivos, el SoC (System on a Chip) ESP32 me pareció una opción acertada para empezar a hacer las pruebas. Las

funciones de los pines de este microcontrolador se pueden ver en la figura 2 y sus principales características son:

- CPU: microprocesador de 32-bits Tensilica Xtensa 32-bits LX6.
- Velocidad de 160 MHz (máximo 240 MHz).
- Co-procesador de ultra baja energía.
- Memoria 512 KB SRAM, 448 KB ROM
- Memoria flash externa hasta 16MB
- Conectividad inalámbrica Wi-Fi 802.11 b/g/n 2.4 GHz (soporta WFA/WPA/WPA2/WAPI)
- Bluetooth v4.2 BR/EDR y BLE
- Chip USB-Serial :CP2102
- Antena en PCB
- Voltaje de alimentación (USB) 5 V DC
- Voltaje de Entrada/Salida 3.3 V DC
- Consumo de energía 5 uA en modo suspensión
- Pines digitales GPIO: 24
- Permite la encriptación de la memoria Flash
- Criptografía acelerada por hardware
- Conversor Analógico Digital Dos ADC de 12 bits

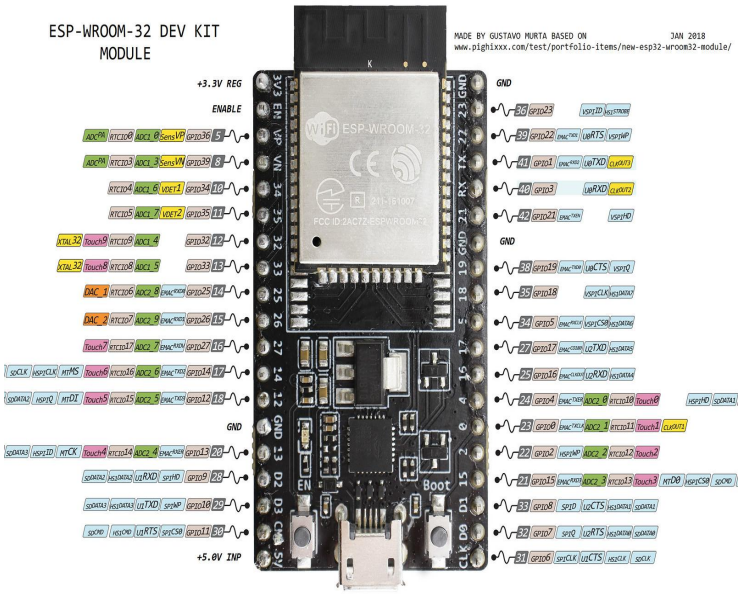


Figura 2 Microcontrolador ESP32 [1]

2.2- Miniordenador para el módulo de monitorización

Para el módulo de monitorización, se optó por favorecer el tamaño reducido para mejorar la portabilidad, así como por un coste económico bajo, pero que tuviera la suficiente capacidad como para permitir la instalación de las aplicaciones y las conexiones que se necesitan, como WiFi para leer los datos desde IoT, instalación de aplicación de control, cámara Web para videovigilancia, etcétera.

El dispositivo más versátil y disponible que se encontró en el mercado fue el computador Raspberry Pi.

Raspberry Pi es un mini-ordenador cuyos primeros modelos fueron lanzados en abril de 2012 y entre sus principales características están sus dimensiones pequeñas, económico

y consumo bajo, lo que le da una gran capacidad para implementarlo en dispositivos portátiles donde podría funcionar incluso con baterías de forma autónoma.

Generalmente este tipo de mini computadores están basados en SO Linux, sin embargo, su hardware no es modificable.

Incorpora dispositivos de E/S como son:

- Pines GPIO (General Purpose Input/Output).
- Sistemas de comunicación:
  - UART (Universal Asynchronous Receiver-Transmitter)
  - SPI(Serial Peripheral Interface)
  - I^2C (Inter-Integrated Circuit).

Fue creado en Reino Unido por la fundación llamada Raspberry Pi, cuyos inicios se remontan a 2008. La idea consistía en el fácil acceso al público ya que resulta un dispositivo de bajo coste e ideal para la mejora de habilidades de programación, electrónica e investigación dado su sencillez de uso.

Existen diferentes tipos de placas microprocesadas que se usarán dependiendo del tipo de aplicaciones y prestaciones como pueden ser [21]:

- Sistemas sencillos y de bajo coste: Raspberry Pi Zero / Zero W / Zero WH
- Sistemas donde se busca potencia de procesador, memoria, conexiones, salidas de vídeo, etcétera: Raspberry Pi 3 en adelante

Modelo	C.P.U.	RAM	Conectividad inalámbrica	Puertos E/S
RASPBERRY PI SPAIN RAS-4-4G	ARM Cortex-A72	4 GB	IEEE 802.11ac Bluetooth 5.0	2x USB 2.0 2x USB 3.0 1x Gigabit Eth. 1x micro HDMI 1x HDMI 1x 3,5mm audio
RASPBERRY PI 4B	1.5 GHz., 4 Core Broadcom BCM 2711 (Cortex-A72)	2, 4 y 8 GB	IEEE 802.11ac Bluetooth 5.0	2x USB 3.0 2x USB 2.0 1x Gigabit Eth. 2x micro HDMI 1x 3,5mm audio
RASPBERRY PI 3B+	1.4 GHz., 4 Core Broadcom BCM 2837B0 (Cortex-A53)	1 GB	IEEE 802.11ac Bluetooth 4.2/ EThernet	4x USB 2.0 1x HDMI 1x 3,5mm audio
RASPBERRY PI ZERO W	1 GHz., 1 Core Broadcom BCM 2835	512 MB	IEEE 802.11n Bluetooth 4.1	1x microUSB 1x mini HDMI
RASPBERRY PI ZERO WH	1 GHz., 1 Core Broadcom BCM 2835	512 MB	IEEE 802.11n Bluetooth 4.1	1x microUSB 1x mini HDMI
RASPBERRY PI ZERO	1 GHz., 1 Core Broadcom BCM 2835	512 MB	-	1x microUSB 1x mini HDMI

En la tabla anterior se muestran las principales características de los diferentes modelos que se pueden encontrar en el mercado actualmente. Datos extraídos de [21] y [22].

La placa utilizada en el proyecto es el modelo Raspberry Pi 4, cuyas características principales son:

- Broadcom BCM 2711 1,50 GHz 64 bits Quad-Core Cortex-A72
- Gráficos VideoCore VI 500 MHz
- Decode 4Kp60 HEVC
- Memoria LPDDR4-3200 4 GB
- WiFi 2,4 Ghz/5 GHz b/g/n/ac
- Bluetooth 5.0 BLE
- Ethernet Gigabit real
- 2x USB 3.0 + 2x USB 2.0
- 2x Micro HDMI con soporte monitor dual
- Conector USB-C para alimentación 5 V 3 A

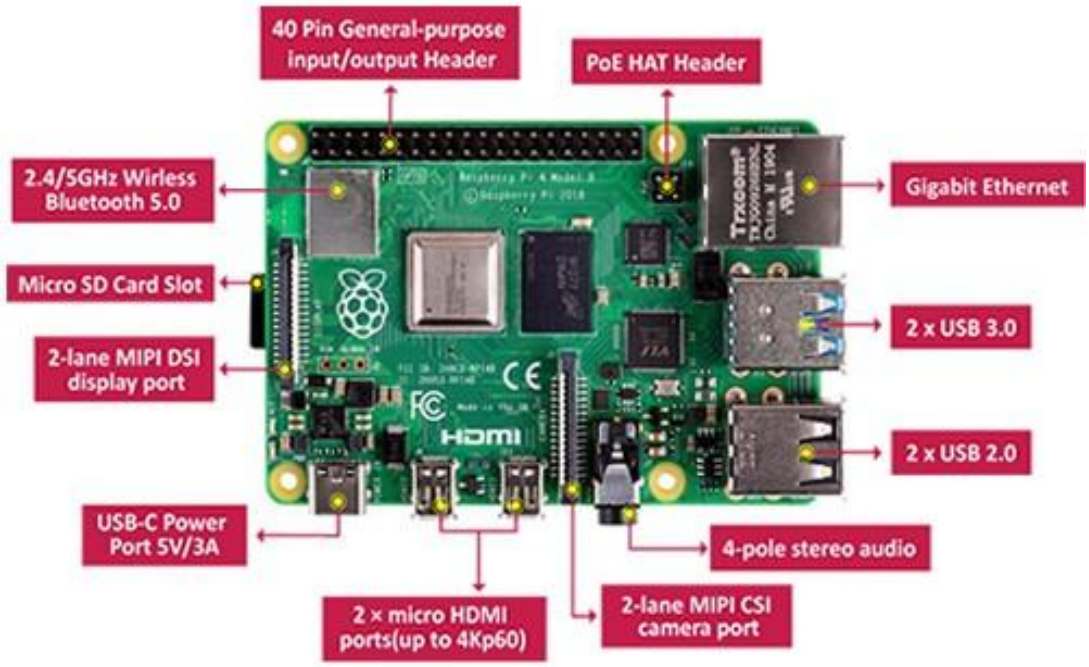


Figura 3 Mini ordenador Raspberry Pi 4 Model B 4 GB [2]

2.3- Sensores ambientales

Para tener monitorizada la habitación donde está aislado el paciente debemos conocer algunos parámetros que nos indiquen la habitabilidad ya que el acceso a la misma es limitado, lo que nos servirá para obtener información y poder actuar en consecuencia. Los sensores que he utilizado miden parámetros como la luz ambiental, la humedad y la temperatura de la habitación. Estos parámetros son especialmente importantes para evitar la transmisión de virus como el COVID-19.

Como explica en las *Noticias ONU* [19], la epidemióloga María Van Kerkhove, líder de la OMS (Organización Mundial de la Salud), la baja humedad atendiendo a los diferentes niveles relativos a este parámetro, puede favorecer la propagación de las "gotitas respiratorias" del coronavirus SARS-CoV-2, causante del COVID-19. De la misma forma, la temperatura ambiental influye en la propagación del mismo.

2.3.1- DHT11

Es un sensor que permite realizar la medición simultánea de temperatura y humedad. Este sensor dispone de un procesador interno que proporciona la medición enviando una

señal digital, por lo que resulta muy sencillo obtener la medición desde un microprocesador como la ESP32. Presenta un encapsulado de plástico similar a la figura 4 con un ejemplo de esquema de conexión.

Las características del DHT11 son las siguientes:

- Medición de temperatura entre 0 y 50 ºC, con una precisión de 2 ºC.
- Medición de humedad entre 20 y 80 %, con una precisión del 5 %.
- Frecuencia de muestreo de 1 muestra por segundo (1 Hz).

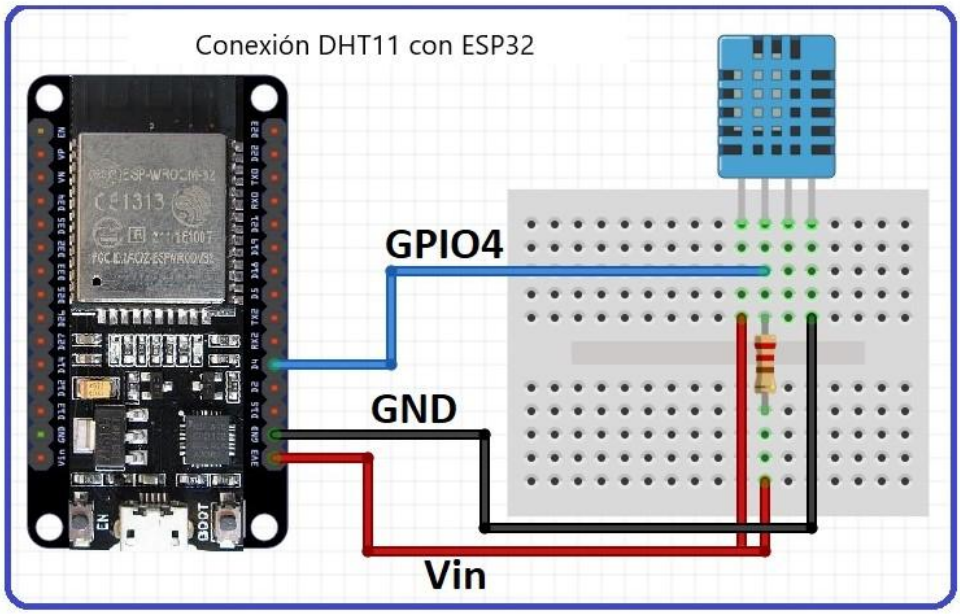


Figura 4 DHT11 [3]

2.3.2- OPT3001

Es un sensor cuya finalidad es proporcionarnos valores de la intensidad luminosa del espectro visible. Las respuestas tanto espectral del sensor como de la fotópica del ojo coinciden estrechamente e incluye un rechazo de infrarrojos significativo. Este sensor viene en una microtarjeta con las conexiones y la circuitería necesaria para su montaje como se muestra en la figura 5.

El fuerte rechazo de infrarrojos también ayuda a mantener una alta precisión cuando el diseño industrial requiere montar el sensor debajo de un vidrio oscuro por motivos de estética.

El OPT3001 está diseñado para sistemas que crean experiencias basadas en la luz para humanos y es un reemplazo preferido ideal para fotodiodos, fotorresistores u otros sensores de luz ambiental con menos coincidencia de ojos humanos y rechazo de IR.

Se pueden realizar mediciones desde 0,01 lux hasta 83 klux sin seleccionar manualmente rangos de escala completa mediante el uso de la función de configuración de escala completa incorporada. Esto permite medir en un rango dinámico de 23 bits de resolución.

La operación digital es flexible para la integración del sistema. Las mediciones pueden ser continuas o de un solo disparo. El sistema de control e interrupción presenta una operación autónoma, lo que permite al procesador dormir mientras el sensor busca los

eventos de activación apropiados para informar a través de la interrupción. La salida digital se informa a través de una interfaz en serie de dos cables compatibles con I^2C y SMBus.

Características:

- Filtrado óptico de precisión para adaptarse al ojo humano:
  - Rechaza > 99 % (típico) de IR
- La función de configuración automática a gran escala simplifica el software y garantiza una configuración adecuada.
- Medidas: 0,01 lux a 83 klux.
- Rango dinámico efectivo de 23 bits con rango de ganancia automático.
- 12 configuraciones de rango de escala completa ponderada binaria:
  - < 0,2 % (típico) Coincidencia entre rangos
- Baja corriente de funcionamiento: 1,8 µA (típico)
- Rango de temperatura de funcionamiento: - 40 ºC a + 85 ºC
- Amplio rango de fuente de alimentación: 1,6 V a 3,6 V
- E/S tolerantes a 5,5 V

Figura 5 OPT3001[4]

2.4- Sensores biométricos

Para la monitorización de los parámetros biométricos este proyecto se centra en aquellos sensores que proporcionan los parámetros básicos de un paciente monitorizado. La temperatura corporal, la saturación de oxígeno en sangre, la presión arterial y el ritmo cardíaco son los valores básicos que nos indican si hay algún tipo de problema cuando no se encuentran estos valores dentro de los márgenes de cada paciente, por su edad, patologías o estado de salud.

Se describen a continuación los sensores utilizados:

2.4.1- *MAX30102*

El MAX30102 es un transductor fabricado por la empresa Maxim Integrated Products que nos proporciona valores de pulsaciones por minuto y porcentaje de oxígeno en sangre, todo esto montado en un integrado para poder ser usado en un microcontrolador ESP32 o similar.

La serie MAX3010x se basa en el comportamiento que tiene la sangre ante la luz en función de su cantidad de oxígeno. Incorpora dos LED, rojo e infrarrojo, y se coloca en el dedo o en la muñeca así la luz reflejada determinará el grado de saturación.

Los glóbulos rojos con mayor cantidad de oxígeno absorben mayor cantidad de luz infrarroja, mientras que los que tienen menos absorben mayor luz roja. Es posible determinar el grado de saturación empleando una diferencia en partes del cuerpo, donde la piel sea muy fina y por la que pasen los vasos sanguíneos.

La comunicación con el MAX30102 se realiza a través del Bus I^2C, así que se montaría en el mismo bus que el utilizado para conectar el OPT3001.

Especificaciones técnicas:

- Voltaje de operación 5 V
- Regulador de voltaje de 3.3 V y 1.8 V en placa
- LED rojo de 660 nm
- LED infrarrojo de 880 nm
- Filtro de luz entre 50 y 60 Hz.
- Protocolo de comunicación I^2C
- Temperatura de trabajo entre -40 ºC hasta 85 ºC
- Conversor ADC delta sigma de hasta 16 bits

El esquema de funcionamiento y la conexión con la ESP32 son las representadas en la figura 6:

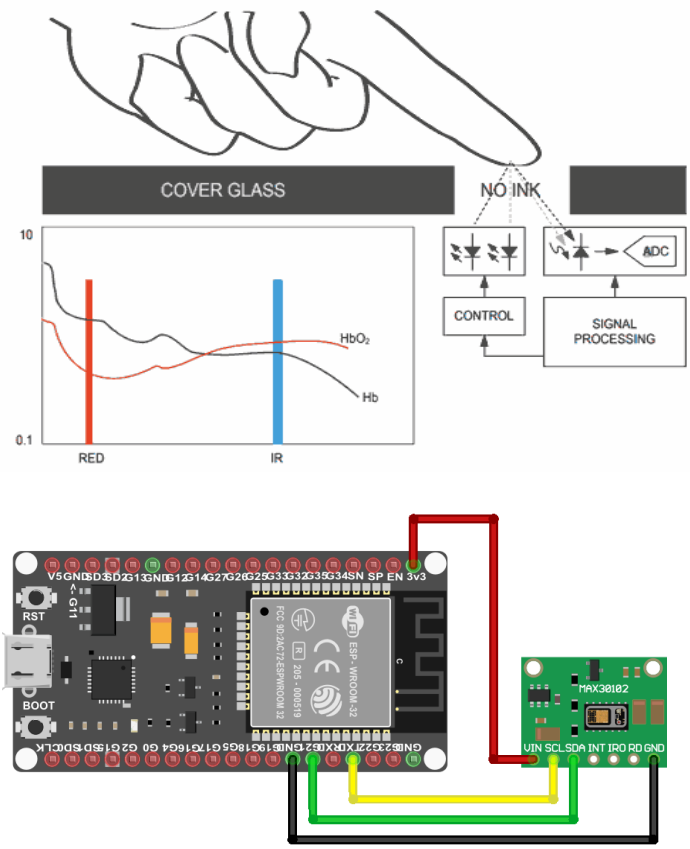


Figura 6 MAX30102 [5], [6]

2.4.2- *MAX30205*

Es un sensor especialmente diseñado para tomar valores de temperatura del cuerpo humano. Mide con una precisión mayor que otros sensores de temperatura similares (tolerancia de 0,1 ºC (37 ºC – 39 ºC)) siendo éste aplicable para uso médico. Utiliza un ADC para convertir la temperatura en señal digital.

La comunicación se realiza a través del bus I^2C así que lo incorporaremos como los sensores anteriores a dicho bus. El esquema de la microtarjeta se representa en la figura 7.

Sus principales características son:

- Voltaje de alimentación entre 2.7 y 3.3 V
- Consumo de 600 µA
- Precisión 0,1 ºC (37-39 ºC)
- Resolución de temperatura de 16 bits
- Pines de entrada/salida 8
- Temperatura de operación 0 – 50 ºC

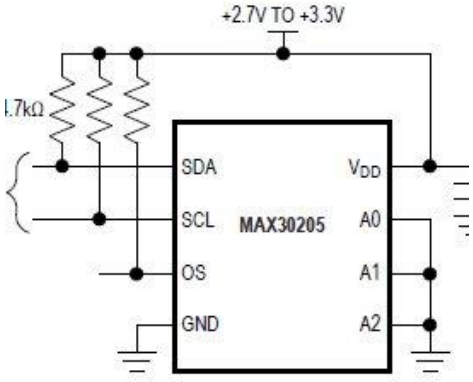


Figura 7 MAX30205 [7]

2.5- Módulos auxiliares de control

Para actuar sobre algunos dispositivos que necesitan mayor voltaje, hemos recurrido a utilizar dos tipos de módulos con los que podemos utilizar diferentes tensiones con las que activaremos varios dispositivos en la habitación del paciente. Dichos módulos se podrían utilizar para varios dispositivos a la vez dependiendo del tipo que usemos. Definimos a continuación los que hemos usado en el proyecto.

2.5.1- *Módulo ULN2003AN*

Es un driver diseñado para controlar motores paso a paso (especialmente el 28BYJ-48 (unipolar)). Puede ser usado para controlar relés, motores DC o cualquier carga que use corriente continua de bajo consumo. La circuitería interna está basada en 7 líneas con pares de transistores NPN en Darlington con diodos protectores para cargas inductivas. Cada uno de ellos es capaz de soportar hasta 500 mA. Si necesitamos que el consumo de corriente entregada sea superior, podemos hacer un montaje utilizando varias salidas y así sumaremos las intensidades para aumentar la capacidad de corriente. En la figura 8, representamos con un ejemplo, el esquema de conexión de un ventilador activado por una de las 7 salidas del módulo.

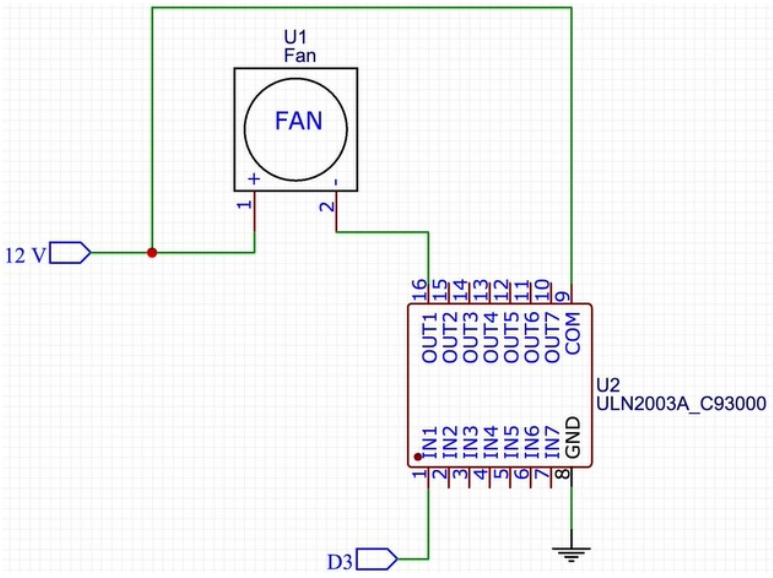


Figura 8 Módulo driver ULN2003 [8]

2.5.2- *Módulo relé SRD-05VDC-SL-C*

Para controlar componentes de alto voltaje o alto amperaje como pueden ser bombillas, motores, solenoides, etcétera, usamos este módulo formado por un relé más sus componentes de protección y control. Esto nos posibilita conectar en los bornes del relé dispositivos que trabajan hasta 220 VAC con lo que nos permite activar con señales pequeñas de control como la de los microcontroladores el funcionamiento de estos componentes de alto voltaje y amperaje. En la figura 9, la línea azul representa la señal de control y los dos cables, rojo y negro, situados en los bornes, activarán el circuito de alta tensión o alto amperaje.

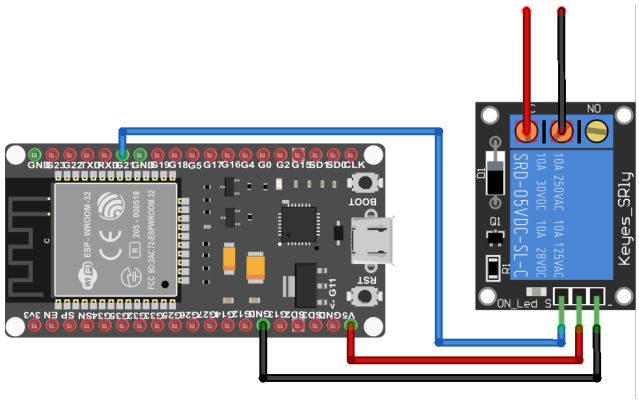


Figura 9 Módulo relé SRD-05VDC-SL-C [9]

2.6- Tecnologías de comunicación

2.6.1- *IoT*

Internet of Things (IoT) es la red de conexiones de objetos físicos provistos de sensores, programación y otras tecnologías, para intercambiar información con otros dispositivos y sistemas utilizando como plataforma de comunicación Internet. El abanico de componentes que forman esta tecnología de comunicaciones, van desde los sensores más diminutos, pasando por electrodomésticos inteligentes, hasta las herramientas industriales más elaboradas.

IoT industrial (IIoT) se refiere al uso de esta tecnología en ambientes industriales o instalaciones similares. De manera especial, se utiliza en la instrumentación, el control de los sensores y dispositivos que participan en este tipo de tecnologías. Últimamente, las industrias han usado la comunicación M2M (máquina a máquina o, en inglés, machine to machine) para lograr de forma inalámbrica su automatización.

En la parte que nos compete de este amplio campo, nos vamos a centrar en la utilización de esta tecnología en el ámbito sanitario. La supervisión de un paciente desde cualquier lugar mediante sensores conectados usando IoT y la posibilidad de interactuar con el medio (véase figura 10), es una de tantas aplicaciones que se le puede dar a esta tecnología, así que se ha implementado en el proyecto para conocer cómo funciona y sus posibilidades.

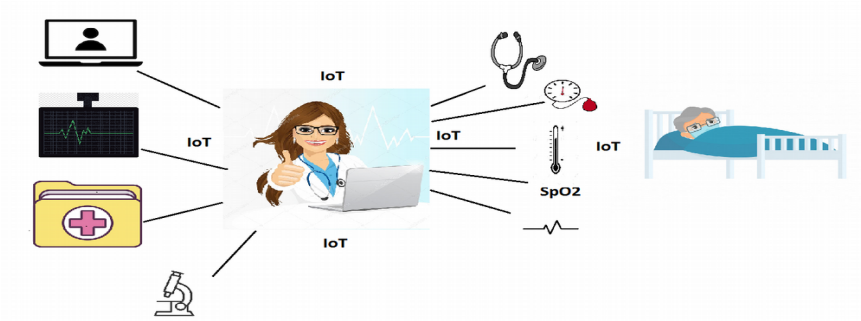


Figura 10 Aplicando IoT en telemedicina

Arquitectura del IoT

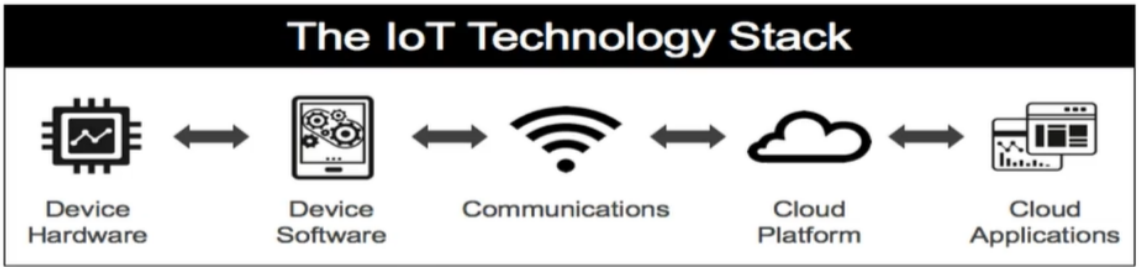


Figura 11 Arquitectura de IoT [10]

Si nos adentramos más en este campo, podemos ver las diferentes etapas que forman la arquitectura del IoT que muestra la figura 11:

- Hardware de dispositivos: Es el conjunto de sensores, actuadores y computadores del tipo que sean, que envían y reciben datos y/o información. Los sensores recogen la información del entorno para enviarlos a equipos o dispositivos capaces de mandar información a Internet, mientras que los actuadores reciben la información de éstos o de las aplicaciones para actuar en el entorno.

Ejemplo de sensores: MAX30205 (sensor de temperatura corporal), DHT11 (sensor de temperatura y humedad ambiente), OPT3001 (sensor de luz ambiente), pulsador, receptor de rayos infrarrojos, etc.

Ejemplo de actuadores: LED, lámpara, motor, ventilador, relé, etc.

- Software del dispositivo: Es una aplicación que transforma un dispositivo hardware en uno inteligente. Puede utilizarse para varias funciones dependiendo del software implementado como puede ser la adquisición de datos a través de sensores, control, análisis y conexión con la nube u otros dispositivos locales [18].

Para el proyecto se han utilizados diferentes tipos de software, desde las librerías usadas para la aplicación del microcontrolador, S.O. de la Raspberry, software de comunicación como VLC, base de datos, etc..., este software se detalla en el apartado 2.7

- Comunicaciones: Es el medio por el cual pueden interactuar entre dos o más dispositivos a la vez, por ejemplo, entre un microcontrolador y un ordenador. La comunicación puede ser mediante una red local, privada, pública, etc. Esta capa incluye tanto a las redes físicas como a los distintos protocolos que se utilizan.

Ejemplo de mecanismos de comunicación: WiFi, ZigBee, LoRa, etc.

- Cloud Platform: Es una plataforma que se sitúa en medio de la interacción entre los dispositivos, podría definirse como la columna vertebral del sistema IoT. Se usa como

sistema de intercambio de información y datos, éstos se alojan en la Nube, lo que da la posibilidad de usarlos remotamente. El protocolo más utilizado en este tipo de tecnologías es MQTT que definiremos más adelante.

Ejemplos de Servidores y Brokers usados para la implementación de este protocolo: CloudMQTT, Ably MQTT Broker, Akiro, Apache ActiveMQ, etcétera. [23]

• Cloud Applications: Son las diferentes aplicaciones desarrolladas para procesar y visualizar los datos de la forma que necesite el usuario final. Las aplicaciones se pueden desarrollar para diferentes plataformas, lenguajes, dispositivos, etc..., todo dependerá de cómo y dónde queramos visualizar los datos, o bien para el tratamiento de los datos masivo (BigData) si nuestro sistema envía gran cantidad de información y ésta necesita ser tratada y evaluada para su comprensión.

2.6.2- *Comunicaciones con sensores*

Actualmente existen multitud de tecnologías de transmisión inalámbrica de datos, analizamos las más utilizadas como ZigBee, Wifi y Bluetooth. Sus principales características son:

- ZigBee

  Es un conjunto de protocolos de comunicación inalámbrica de bajo consumo basado en el estándar IEEE 802.15.4. Es utilizado en gran variedad de instalaciones como son: agricultura, industria alimentaria, control de edificios de viviendas, automatización, seguridad, periféricos de computadoras, monitorización médica, etc.

  Las principales características de este tipo de comunicación son:

  - Optimizar el consumo de baterías.
  - Velocidad de transmisión de datos baja.
  - Nivel de seguridad elevado.

- WIFI

  Del estándar IEEE 802.11n, es el más usado en las comunicaciones inalámbricas de redes locales en edificios y viviendas. Este sistema maneja más volumen de datos y los transfiere con mayores velocidades. Ofrece velocidades de entre 54 y 600 Mbps y las frecuencias de trabajo son de 2,4 y 5 GHz con un alcance de aproximadamente 50 m.

  Usamos este tipo de comunicación porque se usa en la mayoría de los hogares y es una tecnología accesible, ya que el proyecto originalmente estaba pensado para instalarlo en la habitación de algún familiar que tuviésemos aislado en su domicilio.

- Bluetooth

  Del estándar IEEE 802.15.1, Bluetooth fue desarrollado como una comunicación inalámbrica de corto alcance para la sustitución de los cables de conexión entre dispositivos portátiles. A diferencia de ZigBee y WiFi, envía y recibe datos mediante radiofrecuencia. Existen las versiones Bluetooth LE (Low Energy) de bajo consumo y Bluetooth Smart para desarrollo de aplicaciones basadas en IoT.

2.7- Herramientas de software utilizadas

Para el desarrollo del proyecto se han utilizados una serie de herramientas de software que describiremos brevemente a continuación:

- Wiring: Es un lenguaje de programación para placas microcontroladoras tipo Arduino. El equipo que desarrolló la tarjeta Arduino creó un lenguaje de programación de código abierto escrito en C/C++, lo que hace que su sintaxis sea

muy parecida a estos lenguajes. Cuenta con un IDE donde se puede realizar la programación de gran variedad de microcontroladores.

- Protocolo MQTT: Es una comunicación M2M basado en la pila TCP/IP como base para las comunicaciones y es el más usado para IoT por su facilidad y ligereza. Los clientes se enlazan con un servidor central denominado broker. Para organizar los mensajes enviados a los clientes se disponen jerárquicamente mediante el uso de topics (temas). Otro cliente puede suscribirse a un tema y el servidor le hará llegar esos mensajes. La figura 12 muestra cómo la nube actúa de "filtro" entre uno o varios publicadores, que envían diferentes mensajes con distintos temas, y los suscriptores que accederán a los temas específicos que necesiten.

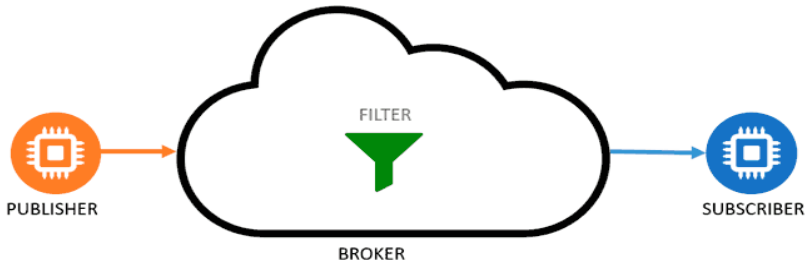


Figura 12 Protocolo MQTT [11]

- Raspbian o Raspberry Pi OS: Es uno de los sistemas operativos más utilizados para las computadoras basadas en Raspberry Pi por la optimización de su hardware. Se basa en Debian, una de las distribuciones de GNU/Linux. Este sistema está diseñado para el aprendizaje e incluye en el menú "raspi-config", permitiendo modificar el S.O. sin editar documentos de configuración de forma manual.

- Tkinter: Es un módulo de interfaz gráfica para trabajar con el lenguaje Python. Viene incluida en el paquete del S.O. Raspbian y nos facilitará la creación de la ventana de visualización de los valores que queremos mostrar de forma continua y nos permitirá también incrustar una ventana para visualizar la imagen que nos envía la webcam.

- MariaDB: Es un gestor de bases de datos desarrollado por Michael Widenius y la fundación MariaDB junto con la colaboración de la comunidad de desarrolladores de software libre. Este gestor proviene de MySQL con licencia GPL (General Public License).[17]

- DBeaver: Es un software de código abierto para el manejo y administración de bases de datos SQL. Es usado por desarrolladores, administradores de bases de datos, analistas y todos los que necesiten utilizar herramientas de este tipo.

- VNC: Es un software libre de estructura cliente-servidor usado para la visualización de las acciones de un ordenador remoto (servidor) a través de otro ordenador diferente (cliente) para su uso y manejo no presencial.

3.- Desarrollo del proyecto

Para el trabajo con la tarjeta microcontroladora he hecho un curso online de Arduino (UDEMY) para aprender a instalar los distintos sensores y controlar varios dispositivos. También hice otro curso online sobre Raspberry Pi para conocer las posibilidades que ofrece este tipo de placas e incorporar como solución para el acceso exterior, instalación de la aplicación y visualización del paciente con cámara web. La experiencia formativa me sirvió para comprender cómo la combinación de sistemas Arduino, con Raspberry Pi y la utilización de la nube permite instalaciones de monitorización de bajo coste y con múltiples posibilidades.

3.1- Comprobación de sensores

La primera fase de la implementación será la comprobación del funcionamiento de los sensores que vamos a utilizar. Para ello se montan sobre una placa Protoboard con una fuente de alimentación (+5V y +12V), y la conexión USB para la comunicación serie con la interfaz IDE de Arduino. Instalaremos en un ordenador portátil el software necesario para programar nuestra placa ESP32.

Una vez configurada la interfaz y comprobado que se comunica a través del IDE, nos descargamos las librerías de cada uno de los sensores. Las pruebas se hacen de forma individual para comprobar que los resultados que nos dan son correctos y están dentro de los valores que necesitamos. Posteriormente iremos incorporando tanto en la placa de ensayo Protoboard como en el código, varios sensores a la vez, hasta que los tengamos todos funcionando como se ve en la figura 13.

También se ha comprobado con un pulsador, un ventilador y una lámpara, que el sistema puede actuar sobre diversos dispositivos. Se han utilizado modelos más pequeños de lo habitual, para que el montaje final del prototipo fuese similar a una maqueta.

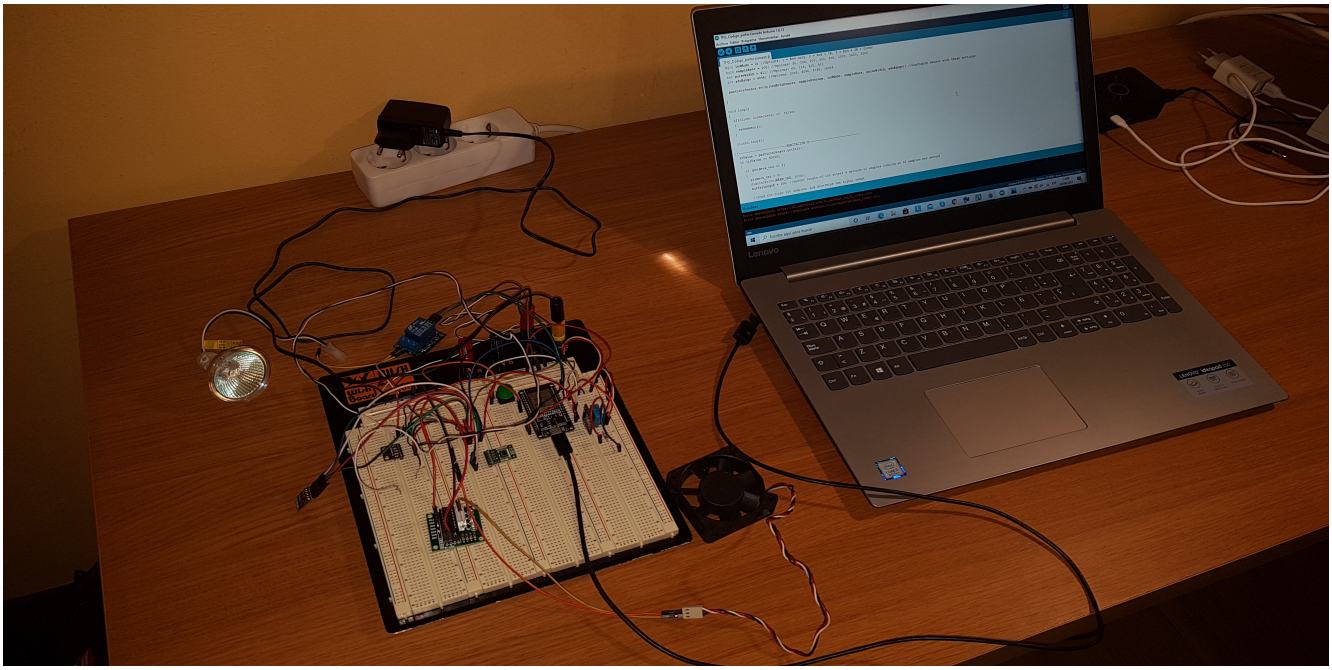


Figura 13 Montaje en Protoboard

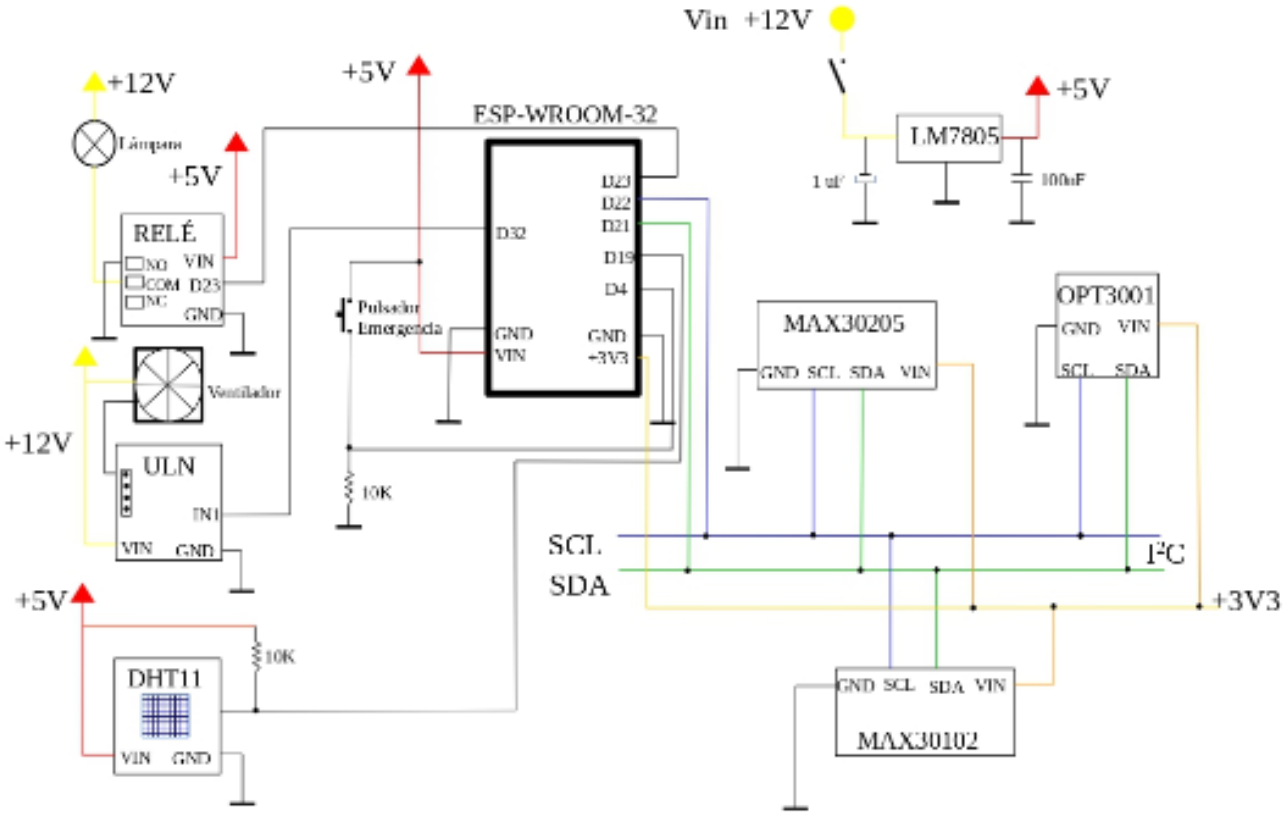


Figura 14 Esquema de conexiones

El circuito de la figura 14 está alimentado con una fuente externa de 12 V, de los cuales sacaremos de un regulador de tensión (LM7805) 5 V para alimentar la ESP32 y los diferentes módulos. El segundo bloque es el de actuadores constituido por un módulo relé (*SRD-05VDC-SL-C*) para activar la lámpara, un módulo de transistores NPN (*ULN2003AN*) para controlar el ventilador, ambos necesitan alimentación a 12 V y actuamos sobre ellos con señales de control y un pulsador de emergencia. El último bloque está compuesto por los sensores, estos son el DHT11 y los tres sensores que utilizan el bus I^2C. El circuito funciona con tres líneas de tensiones diferentes:

- 12 V: Esta línea se usa para alimentar la lámpara y el ventilador.
- 5 V: Usado para alimentar el sensor de temperatura y humedad ambiental, el pulsador de emergencia y la ESP32.
- 3.3 V: Utilizado para alimentar los dispositivos del bus I^2C y generador internamente por la ESP32.

3.2- Comunicaciones e IoT

Una parte importante de la implementación es la configuración de las comunicaciones para el envío y la recepción de los datos. Se configura la comunicación Wifi de la ESP32 para su posterior uso y se activa la comunicación con el Broker MQTT.

Tenemos que configurar ambos módulos para el acceso vía Wifi al Servidor para hacer las pruebas de comunicación. El paso previo es tener un usuario registrado. Faltaba configurar los topics que necesitaba para el proyecto y comprobar que el sistema podía suscribirse y enviar datos. Se define también la estructura necesaria para poder recibir datos

desde diferentes habitaciones, si se diese el caso de montar un sistema para múltiples pacientes. En la imagen se puede apreciar que está definida la HabX como única habitación en el caso de que el sistema fuese para una sola, si se configura para varias, habría que definir cada habitación con su número correspondiente, para que los datos se puedan leer individualmente por habitación.

A continuación se muestra en la figura 15 la pantalla de configuración y los mensajes que van llegando a los distintos topics.

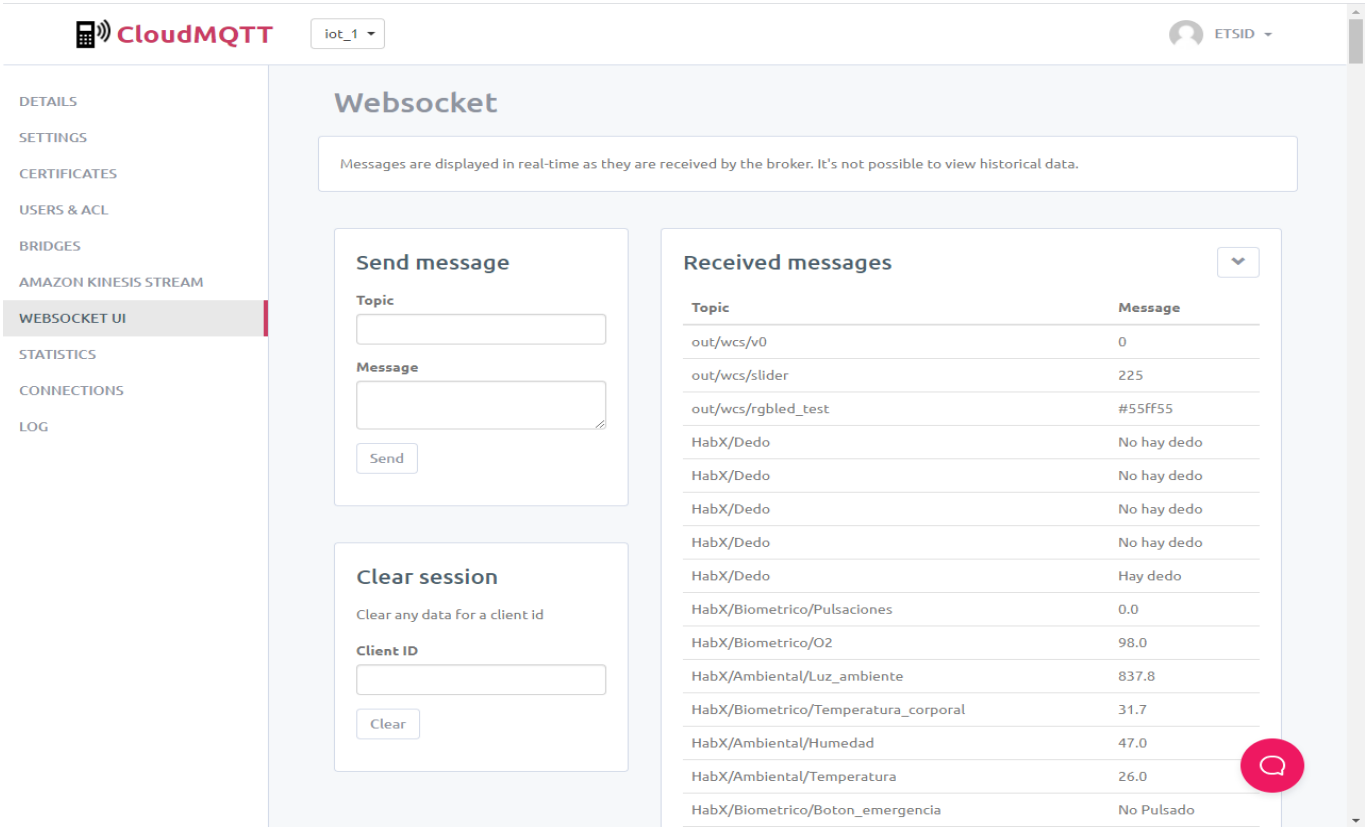


Figura 15 Recepción de mensajes en el CloudMQTT

3.3- Software del módulo de sensado

Se describe a continuación con un esquema de bloques, el funcionamiento del código desarrollado para la ESP32.

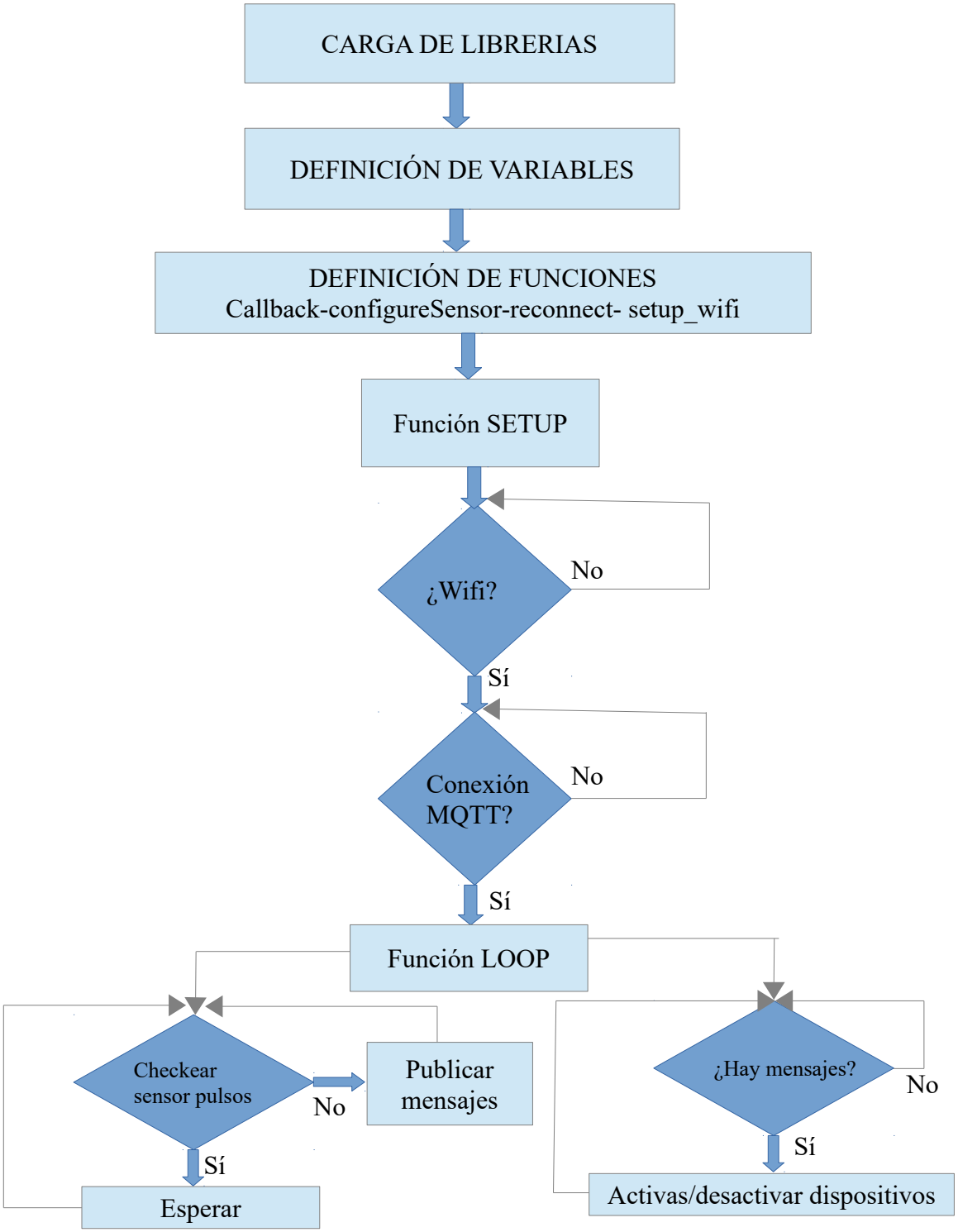


Figura 16 Diagrama de bloques código ESP32

Lo primero que hacemos es cargar las librerías necesarias para usar las diferentes funciones de la WiFi, MQTT y sensores. A continuación, definimos las variables globales del código necesarias y objetos para usar algunas funciones específicas. Luego, creamos funciones para que el programa las llame cuando se necesiten y conseguir una mayor eficiencia a la hora de su ejecución. Ejecutamos la función Setup para comprobar si hay WiFi, luego si tenemos conexión al broker y realizar diferentes funciones dentro de ésta para inicializar los sensores. Finalmente, continuamos con la función Loop en la que estaremos viendo dos partes: la primera, si el sensor MAX30102 ha detectado un pulso del dedo con el que comenzaremos a publicar mensajes, y la segunda parte, en la que comprobaremos si hay mensajes en el servidor de los temas en los que nos suscribimos.

Está basado en ejemplos de los diferentes módulos y dispositivos extraídos de diferentes autores que se han ido probando y adaptando para su funcionamiento en conjunto del proyecto.

Pasamos ahora a explicar algunas partes del código que son relevantes.

Función para conectar con la WiFi

```
void setup_wifi()
{
  WiFi.begin(ssid,password);
  while(WiFi.status() != WL_CONNECTED)
  {
    delay(500);
  }
}
```

Esta función empieza enviando los datos de la WiFi (identificador y contraseña) y comprueba si hay conexión con la red, si no se establece la comunicación, se reinicia el proceso con un retardo de medio segundo. Si no se resuelve, queda en un bucle infinito, por lo que se debe volver a ejecutar el código.

Este código está basado en los ejemplos del curso de Arduino de la página Web de Udemy.

Función para conectar con CloudMQTT

```
void reconnect()
{
  while(!client.connected())
  {
    String clientId = "iot_1_";
    clientId = clientId + String(random(0xffff), HEX);

    if(client.connect(clientId.c_str(),mqtt_user,mqtt_pass))
    {
      client.publish("HabX/MQTT","Conectado a CloudMQTT desde ESP32");
      client.subscribe("HabX/Ambiental/luz_habitacion");
      client.subscribe("HabX/Ambiental/ventilador");
    }
    else
    {
      delay(5000);
    }
  }
}
```

Esta función comienza con la comprobación del estado del cliente, si no se da el caso, introduce un id para el cliente y usa los datos de CloudMQTT para conectarse, si nos conectamos al broker, nos suscribimos a los temas que queremos y publicamos un mensaje al servidor para verificar que la ESP32 está conectada.

La función utilizada para publicar la información de los sensores es la siguiente línea de código:

client.publish(“tema”, “mensaje”)

Al igual que la función de la WiFi, este código proviene del curso de Arduino de la página Web de Udemy.

3.4- Módulo de monitorización

El siguiente paso en el desarrollo del proyecto consiste en el montaje y configuración del módulo de monitorización. Este módulo se basa en la Raspberry Pi 4 como minicomputador para el acceso a los datos y la instalación de la interfaz de usuario donde se muestran los parámetros relevantes y donde también podremos interactuar con los dispositivos ubicados en la habitación.

La Raspberry Pi necesita para su mejor funcionamiento (ya que tiene problemas de calentamiento según comentarios vistos en Internet) una caja con ventilación forzada ya que sería un dispositivo que va a estar funcionando 24-7. Además necesitamos para este módulo una webcam que nos servirá para la visualización del paciente y unos altavoces donde sonará la alarma en caso de emergencia. Se completa el módulo con pantalla, teclado y ratón para su visualización y control. También se contempla la posibilidad de acceder a la Raspberry remotamente a través de tablet o portátil en caso de no estar cerca de la habitación.

Detallamos los pasos para la instalación y puesta en marcha de la Raspberry Pi 4:

1. Conexión de tarjeta microSD a un ordenador para la instalación de la imagen del S.O.

2. Descargamos de la página web oficial de Raspberry la versión para nuestro sistema operativo usando Raspberry Pi Imager.

3. Ejecutamos el archivo descargado siguiendo los pasos por defecto para hacer la instalación del S.O.

4. Elegimos la tarjeta microSD para la instalación.

5. Finalmente pulsamos “WRITE” para escribir sobre la SD.

6. Retiramos la tarjeta microSD e insertamos en la Raspberry.

7. Alimentamos la Raspberry y conectamos el monitor, teclado y ratón.

8. Arrancamos el equipo y terminamos de configurarlo.

9. Configuramos la conexión WIFI y comprobamos que tenemos acceso a Internet y al Cloud MQTT.

10. Instalamos la webcam y comprobamos su funcionamiento.

11. Conectamos los altavoces y probamos sonidos.

Una vez preparada la Raspberry, hay que estudiar cómo desarrollar la interfaz de usuario y de qué herramientas disponemos. Para seguir con la línea de utilizar herramientas de software libre, se utilizará Tkinter [12] que viene incorporado en la distribución linux utilizada. Es una herramienta que nos facilitará la creación de una interfaz gráfica y que está basado en el lenguaje Python.

Durante el desarrollo de la interfaz se debe configurar la conexión con MQTT y la suscripción a los topics donde se están enviando los datos del módulo ESP32.

El diseño realizado es un interfaz de usuario consistente en una ventana donde se irá emitiendo la imagen de la webcam en directo, así como una ventana de incidencias donde se verán los mensajes que se envían cuando se sobrepasa los límites por defecto o los límites que se introduzcan manualmente por el usuario (ver figura 17).

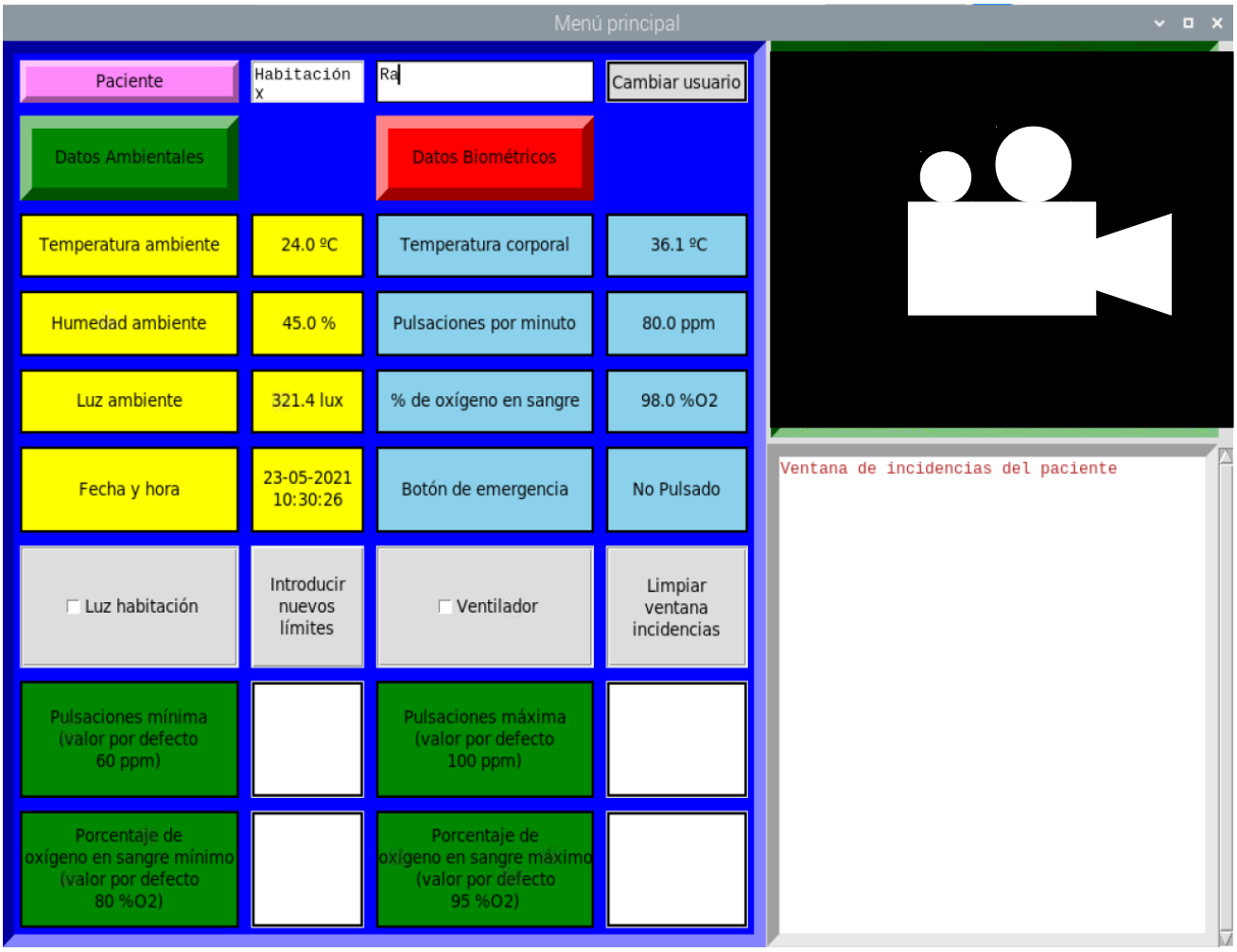


Figura 17 Interfaz de usuario diseñada con Tkinter.

A continuación se muestra en un diagrama de bloques el desarrollo del código en Tkinter.

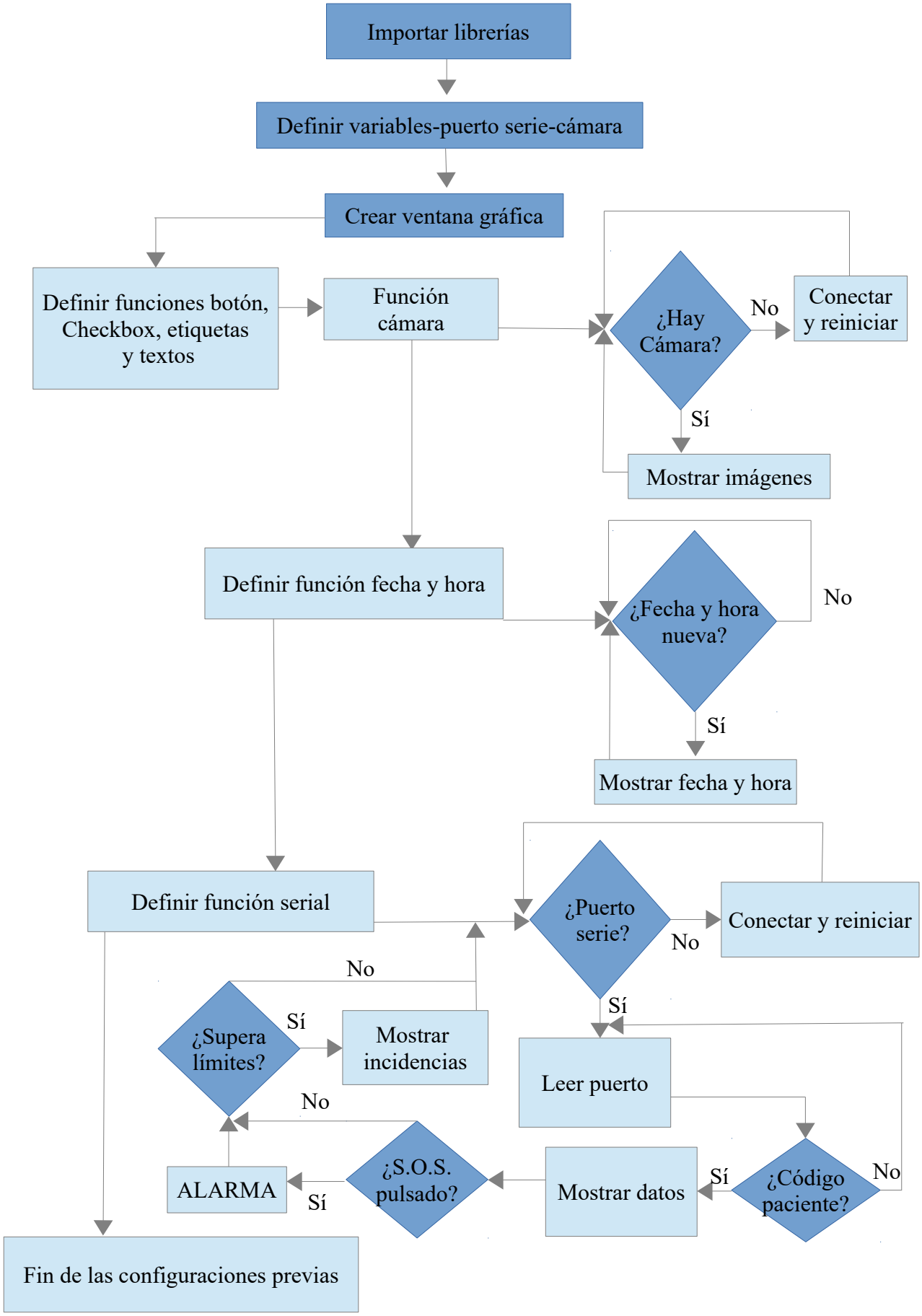


Figura 18.1 Diagrama de bloques código Python.

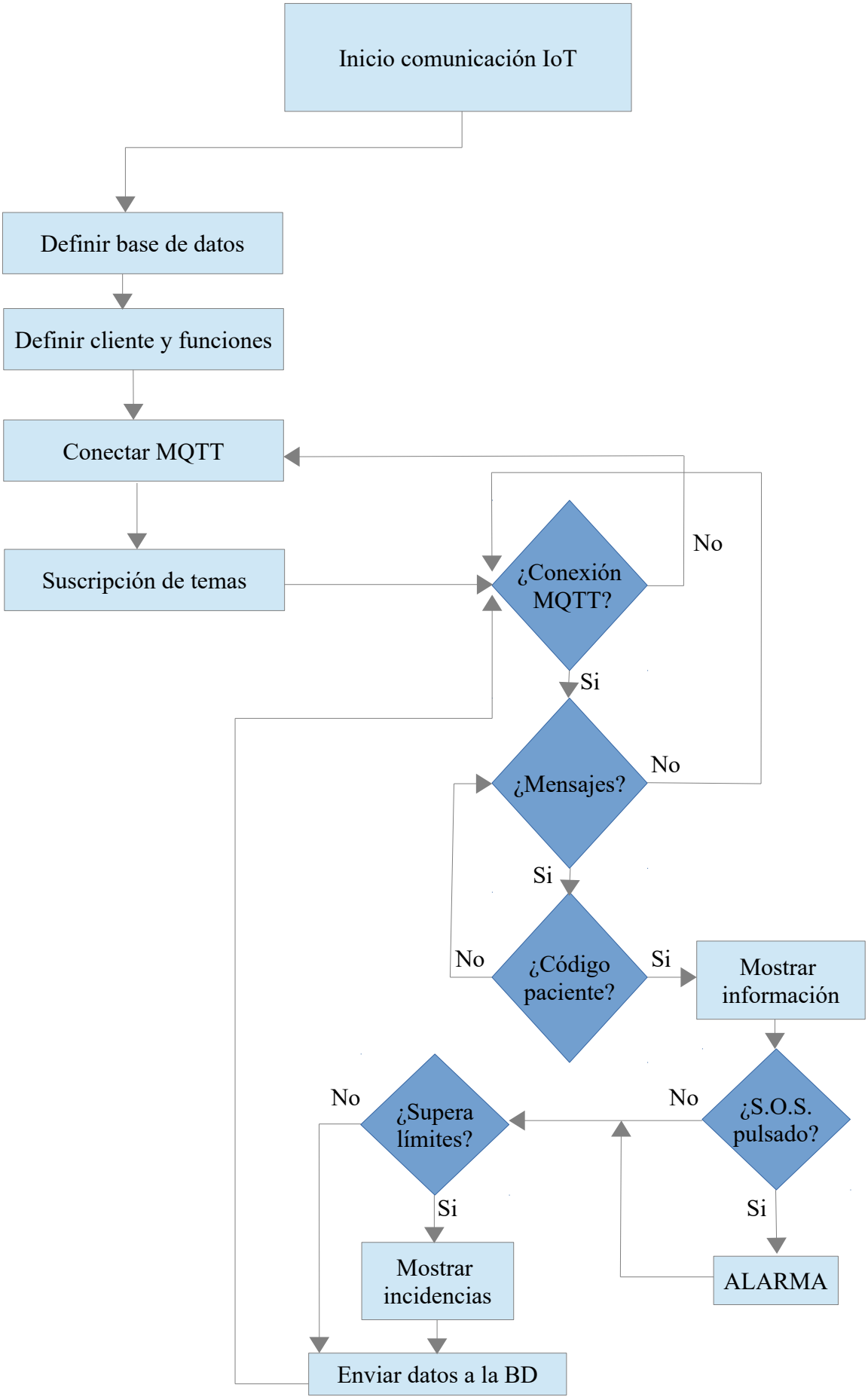


Figura 18.2 Diagrama de bloques código Python.

El programa empieza importando las librerías, creando las variables globales necesarias y la ventana gráfica con los botones, etiquetas, checkbox, cuadros de texto y las funciones. Luego, definimos la función de la cámara y comprobaremos si existe cámara, en caso contrario, debemos conectarla; si funciona creamos la función fecha y hora en la que tendremos en un cuadro estos datos que se irán actualizando; pasamos a la función serial, al igual que la cámara, si no tenemos el puerto serie conectado, el programa no funcionará, esta función lee el puerto cada período de tiempo y muestra los datos en la ventana gráfica y realizará las mismas funciones que si recibiera los datos por WiFi que se explicará más adelante menos el envío a la base de datos. Esta función junto con la cámara son críticas antes de empezar con el núcleo del programa. Si no obtuvimos ningún problema con estas dos, definimos las funciones necesarias para mandar la información a la base de datos y el cliente y sus acciones. Con esto preparado, nos conectamos a CloudMQTT y nos suscribimos a los temas. Antes de saber si hay mensajes en los que estamos suscritos a esos temas, comprobamos por seguridad la conexión a MQTT; si hay mensajes, comprobamos que el cuadro de texto del paciente no esté vacío ya que no mostrará los datos en el caso de que no haya un código o identificador del paciente. Si lo hay, enseñará esa información y pasamos a comprobar si el pulsador SOS ha sido activado para conectar la alarma (sonido enviado a los altavoces), y comprobaremos si se ha superado algún límite para mostrar en el cuadro de incidencias del paciente la fecha, la hora e incidencia que le ocurrió. Finalmente, mandamos esta información a la base de datos.

Las funciones de MQTT son códigos usados en el curso de Raspberry Pi de la página Web de Udemy con modificaciones; la función de fecha y hora es una creación personal basándome en ideas aprendidas de otros códigos; la función serial está extraída del curso de Raspberry Pi y ha sido modificada y adaptada; las funciones de la base de datos y del cliente son del curso con algunas modificaciones; y los botones, las etiquetas, los cuadros de texto y los checkbox son del curso con las configuraciones y ampliaciones necesarias para diseñar la interfaz gráfica.

A continuación comentaremos algunas de las funciones más relevantes.

Función Cámara

```
//El siguiente código es extraído de un ejemplo donde se muestra la webcam en una ventana
//gráfica del Tkinter, incorporándose al código principal [16]
def show_frame():
    _, frame_video = cap.read()
    frame_video = cv2.flip(frame_video, 1)
    cv2image = cv2.cvtColor(frame_video, cv2.COLOR_BGR2RGBA)
    img = Image.fromarray(cv2image)
    imgtk = ImageTk.PhotoImage(image=img)
    etiqueta_camara.imgtk = imgtk
    etiqueta_camara.configure(image=imgtk)
    etiqueta_camara.after(10, show_frame)
```

No solo usaremos la comunicación Wifi para el envío de los datos, también disponemos del puerto serie como respaldo, en caso de algún problema en la lectura de información a través del Broker o caída de la red local. Por ello se crea una función para la lectura del puerto serie y realiza la muestra de datos en la interfaz gráfica del mismo modo que por WiFi.

A continuación se muestra el código realizado para lectura del puerto serie.

Función Serial

```
def serial():
```

```
while True:
        now = datetime.now()
        format = now.strftime('%d-%m-%Y %H:%M:%S')
        global limite_pul_min
        global limite_pul_max
        global limite_O2_min
        global limite_O2_max
        global Avisos
        global paciente
        //Lectura puerto serie
        line = arduino.readline()
        //Filtro de la lectura
        line_2 = str(line.decode("utf-8"))
        //Se introduce los datos separados en un vector
        trozos_line_2 = line_2.split()
        //Si la línea 2 contiene datos, empieza la ejecución de la aplicación
        if line_2 != "":
                //Si hay un código del paciente, se mostrarán los datos recibidos
                if paciente != "":
                        //Muestra de los datos en las casillas correspondientes de la
                        interfaz gráfica
                        label_luz_amb.set(trozos_line_2[0]+" lux")

                        label_temp_corp.set(trozos_line_2[1]+" ºC")
                        global corporal
                        corporal = float(trozos_line_2[1])
                        Avisos = ""
                        //Comprobación de la temperatura corporal
                        if(corporal < 36.0):

                                texto_avisos.insert(tk.INSERT,"\n"+format+"
                                -> Hipotermia")
                                Avisos = "Hipotermia"
                        if(38.0 < corporal):

                                texto_avisos.insert(tk.INSERT,"\n"+format+"
                                -> Fiebre")
                                Avisos = "Fiebre"

                        label_hume_amb.set(trozos_line_2[2]+" %")

                        label_temp_amb.set(trozos_line_2[3]+" ºC")

                        label_boton.set(trozos_line_2[4]+"
                        "+trozos_line_2[5])
                        global Boton
                        Boton = str(trozos_line_2[4]+" "+trozos_line_2[5])
                        //Comprobación del estado del pulsador de emergencia
                        if Boton == "Sí Pulsado":
                                texto_avisos.insert(tk.INSERT,"\n"+format+"
                                -> BOTÓN DE EMERGENCIA")
                                etiqueta_boton_valor.config(bg="red")
                                //Activación de la alarma
```

```
            pygame.mixer.music.load("alarm-clock.mp3")
            pygame.mixer.music.play()
    if Boton == "No Pulsado":
            etiqueta_boton_valor.config(bg="skyblue")

    label_pulso.set(trozos_line_2[6]+" ppm")
    global pulsaciones
    pulsaciones=float(trozos_line_2[6])
    Avisos = ""
    //Comprobación de las pulsaciones por minuto
    if(pulsaciones > int(limite_pul_max)):

            texto_avisos.insert(tk.INSERT,"\n"+format+"
            -> Taquicardia")
            Avisos = "Taquicardia"
    if(pulsaciones < int(limite_pul_min)):

            texto_avisos.insert(tk.INSERT,"\n"+format+"
            -> Bradicardia")
            Avisos = "Bradicardia"

    label_O2.set(trozos_line_2[7]+" %O2")
    global oxigeno
    oxigeno = float(trozos_line_2[7])
    Avisos = ""
    //Comprobación de la saturación de oxígeno en sangre
    if(int(limite_O2_min) <= oxigeno <=  int(limite_O2_max)):

            texto_avisos.insert(tk.INSERT,"\n"+format+"
            -> Hipoxemia")
            Avisos = "Hipoxemia"
    if(oxigeno < int(limite_O2_min)):

            texto_avisos.insert(tk.INSERT,"\n"+format+"
            -> Hipoxemia severa")
            Avisos = "Hipoxemia severa"
```

3.5- Almacenamiento local de datos en el módulo de monitorización

El almacenamiento de datos para su posterior uso lo haremos configurando una base de datos en Raspberry, esta base de datos se llama MariaDB [13].

Para ello, vamos a la ventana de comandos de la Raspberry y configuramos el servidor de la base de datos creando además un usuario y contraseña.

Para realizar la gestión de esta DB usaremos una aplicación externa llamada DBeaver [14], un gestor de bases de datos.

Aquí crearemos una conexión nueva y especificaremos la IP del servidor de la DB, la base de datos usada (en nuestro caso MariaDB), nombre de la conexión (opcional), usuario y contraseña.

Crearemos una nueva base de datos llamada “Habitaciones” y dentro una tabla llamada “Habitacion_X” (esto se ha diseñado así para el caso de que si tenemos varias habitaciones, crearemos tantas tablas como habitaciones hayan, sustituyendo la “X” por su número correspondiente).

Finalmente, crearemos las columnas con los temas y añadimos 3 columnas con los datos del paciente, fecha y hora e incidencias.

Con estos datos se pueden extraer individualmente por paciente, por habitación, etc., dependiendo del tipo de consulta que se quiera realizar.

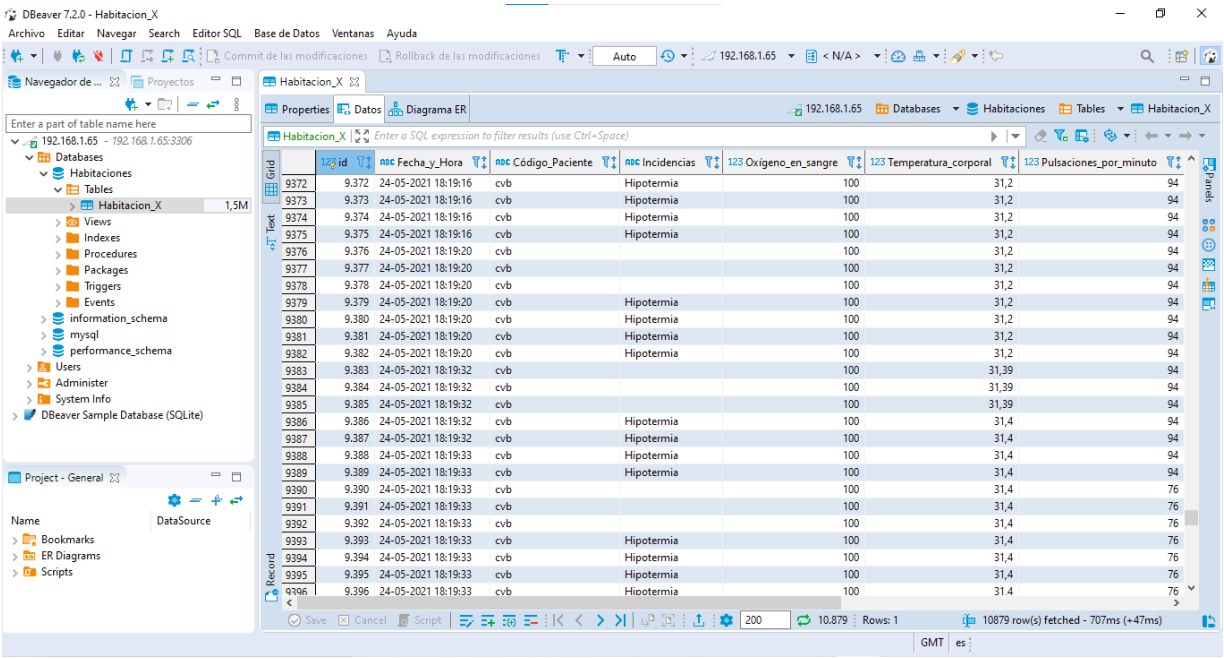


Figura 19 Visualización del gestor DBeaver

3.6- Resultados obtenidos con el prototipo

Una vez el prototipo esté funcionando y se proceda a la instalación del sistema completo, podemos enumerar a continuación los siguientes resultados:

- El primer paso a realizar es la codificación del paciente en el módulo de monitorización para proceder a la apertura de la tabla de la base de datos donde se irán registrando los parámetros del paciente.
- En el caso de ser un sistema con múltiples habitaciones, se repetiría el proceso de forma independiente para cada una de las habitaciones siendo la base de datos y el Servidor MQTT comunes.
- Conexión de la pulsera y dedal al paciente para comenzar el envío de datos de los sensores biométricos. Se modificarían los valores por defecto del módulo de monitorización en caso de que fuese necesario. De esta manera comenzaría el envío de datos y por tanto el sistema estaría monitoreando al paciente apareciendo en la ventana de incidencias la fecha y hora de cada evento ocurrido.
- De forma paralela, se conectan los dispositivos a controlar y se le proporciona al paciente el pulsador de emergencia.
- Una vez conectados los dispositivos y comprobado el funcionamiento correcto de ambos módulos, tendremos como resultado la posibilidad de proporcionar teleasistencia a la persona aislada en la habitación.

Con respecto a los valores obtenidos durante las pruebas con los sensores utilizados, hemos comprobado los siguientes resultados:

- El sensor MAX30102 ha sido el más complejo de ajustar sus parámetros hasta conseguir valores lo suficientemente fiables como para compararlos con valores reales.

Se han probado diferentes programas del mismo sensor e incluso se han hecho pruebas con el MAX30100 que utiliza las mismas funciones de la librería proporcionada para este sensor comparando resultados entre ambos modelos. Se ha podido comprobar que dependiendo de los ajustes de diferentes variables, los resultados son diferentes (frecuencia de muestreo, promedio de la muestra, luminosidad del LED, modo del LED, ancho de pulso y rango del ADC).

Uno de los códigos que se ha utilizado extrae las primeras 100 muestras para la calibración del sensor para a continuación mostrar los valores en referencia a esa calibración.

Finalmente, se ha conseguido que los valores de pulsaciones por minuto estén entre 50 y 130 ppm y los de porcentaje de oxígeno en sangre entre 86 y 100 %O2

- El sensor MAX30205 fue menos complejo a la hora de programarlo pero también hubo que realizar pruebas de adaptación ya que dependía de la zona del cuerpo donde se hiciera la medición.

  Al final se optó por incorporar en una pulsera para mayor comodidad del paciente y se ajustó la diferencia de la temperatura tomada por ese sensor en la muñeca con la temperatura real medida por un termómetro. La diferencia entre las dos temperaturas se corrigió en el código para que el resultado mostrado fuera el valor más cercano al real.

4.- Conclusiones y líneas futuras

4.1- Líneas futuras de desarrollo

Las posibilidades de mejorar y desarrollar nuevos módulos que amplíen las capacidades del sistema son múltiples. Se proponen a continuación varias ideas sobre las que se podría trabajar en un futuro y adaptar el prototipo a las necesidades que cada situación requiera. Se han agrupado las ideas en tres líneas básicas de actuación:

1. *Parámetros y equipamiento médico*:

   Si necesitamos que se amplíe las capacidades de monitorización del estado del paciente, o si éste necesita algún equipo médico funcionando en la habitación, se podría estudiar la posibilidad de incorporar al sistema y por tanto su estudio e implementación, de equipamientos como:

   - **Toma de medidas de presión arterial**: Para ello existen tensiómetros con posibilidad de conexión Bluetooth. El módulo de sensores al incorporar el microcontrolador ESP nos da la posibilidad de este tipo de comunicación y se desarrollaría el código necesario para la lectura de los valores y el tiempo entre cada medida.
   - **Respirador artificial**: Si la situación del paciente requiere de este tipo de equipamiento, podríamos controlar varios parámetros del aparato. Habrá gran variedad de modelos en el mercado desde los más simples que solamente bombean aire hasta los más complejos que permiten su programación y conexión a sistemas externos de control centralizados como los usados en unidades críticas en los hospitales. Como se comenta en el artículo de EFE: SALUD con respecto a los diferentes tipos de respiradores utilizados frente al COVID-19 [24].
   - **Apnea del sueño**: Es un equipo utilizado para el control de las apneas del sueño. Proporciona una presión positiva constante para evitar que se bloqueen las vías respiratorias. En este tipo de aparatos se podría estudiar la posibilidad de monitorizar su funcionamiento a través del módulo de sensores ya que aunque su funcionamiento es muy simple, si sufriera algún tipo de avería o

desconexión, podría provocar al paciente alteraciones o fallos respiratorios graves.

- **Bombas de infusión de jeringa**: Los equipos para el suministro de medicación donde se programan el caudal que debe inyectarle al paciente, son otros de los posibles aparatos que se podría estudiar para incorporar su control al sistema. Podrían incluirse dentro de este aparato otros aparatos como pueden ser las bombas de aire de los colchones antiescara que son muy utilizadas en las personas mayores que llevan mucho tiempo encamadas, o se podría incluso ver la posibilidad del control remoto de las camas articuladas diseñando un interfaz que se conecte al módulo situado en la cabecera para incorporar al paciente o cambiarlo de posición cuando son de movilidad reducida. Habría una solución para controlar y monitorizar cada aparato que se necesite instalar en una habitación, con la idea de que nos permite reducir en gran medida la necesidad de acceder a ella y por lo tanto mejora la seguridad y el aislamiento del paciente.

2. *Condiciones ambientales*:

Para la comodidad del enfermo y para tener acceso y control a las condiciones de la habitación, habría que ver la posibilidad de ampliar la conectividad del módulo de cabecera para el control de los aparatos y dispositivos instalados en ella.

Se podría ampliar con módulos como el instalado en el prototipo con relés si queremos activar aparatos a 240 VAC, o incorporar un módulo de infrarrojo, etc. Otros dispositivos serían:

- **Equipos de aire acondicionado**: Si no existiese instalación general de aire acondicionado se instalaría un equipo portátil y podríamos estudiar la forma de controlar su funcionamiento.
- **Humidificadores de aire**: Para mantener el correcto nivel de humedad ambiental podemos incorporar este tipo de aparato ya que el sistema al monitorizar este valor, podemos establecer los parámetros que necesitamos y activarlo en caso de que sea necesario. Este aparato es fundamental en el cuidado de enfermos de Covid ya que tanto la humedad ambiental como la temperatura, son valores que puede aumentar las posibilidades de propagación del virus.
- En el apartado de **iluminación** se podría ampliar el sistema con la interconexión de iluminación inteligente o implementar módulos de domótica que nos permita controlarla desde la aplicación de monitorización. Podría controlarse tanto la iluminación artificial como la natural incorporando persianas automáticas, cortinas, etc.
- Con la instalación de un **módulo infrarrojos**, podríamos controlar una televisión que instalaremos en la habitación para el entretenimiento del paciente. Si fuese una persona mayor que le resulte engorroso manejar un mando de TV, con la posibilidad de que se le caiga, o que se la podamos apagar en caso de que se quede dormida, como podemos ver al paciente mediante la webcam, podríamos actuar sobre la TV sin entrar en la habitación.

Estas serían algunas de las posibilidades que podríamos estudiar para desarrollar en el futuro y mejorar las condiciones ambientales. Existen hoy en día dispositivos que controlan este tipo de aparatos también por su voz (p.e.: Alexa). Dependerá de la movilidad y capacidades del paciente si decidimos un sistema u otro.

3. *Desarrollo de aplicaciones y monitorización*:

   En la parte software hay también multitud de posibilidades de mejora y desarrollo. Nombramos a continuación algunas de las ideas que se podrían estudiar para su futura implementación.

   - **Monitorización de paciente**: La interfaz gráfica donde se visualiza tanto los datos del paciente como los de la habitación tiene múltiples posibilidades de ampliación y mejoras. Dependiendo del número de dispositivos (como los que se han nombrado anteriormente) con los que se quiera hacer una instalación personalizada, la interfaz debería contemplar la visualización y control de los mismos. En la parte médica se podrían desarrollar unas gráficas con los valores biométricos que se van registrando para hacer más fácil el seguimiento y las posibles incidencias ocurridas.
   - Se podría implementar una **comunicación multimedia** vía monitor-micro-altavoces, tablet o TV que nos permita comunicarnos con el paciente en cualquier momento y poder interactuar con el mismo de forma remota.
   - Como se están enviando los datos a través de IoT, necesitaríamos implementar un **sistema de seguridad** mediante encriptado y acceso a través de VPN's desde cualquier lugar hasta la aplicación de monitorización.
   - Para la ayuda al diagnóstico del personal sanitario, se podría desarrollar con los datos que se están almacenando, algunos programas que les facilite la incorporación de los mismos en el **historial clínico** del paciente para su control "on-line" y añadir posibles alarmas con los valores críticos que se han definido previamente.

4.2- Conclusiones

La idea con la que nació este proyecto, fue la de crear un sistema que ayuda en el cuidado de personas que estuviesen aisladas en una habitación, como consecuencia de sufrir algún tipo de enfermedad contagiosa como el COVID-19 o similar. Durante el desarrollo del mismo se ha podido demostrar que mediante el uso de sensores que se pueden conseguir en el mercado de forma sencilla, y plataformas de desarrollo como Arduino y Raspberry, se puede construir un prototipo que nos permita conocer el estado de salud de esa persona y actuar de forma remota, mejorando sustancialmente la seguridad para el enfermo y sus cuidadores. Se ha intentado buscar los elementos necesarios, que componen el prototipo, más económicos y asequibles, al igual que el software utilizado sea gratuito o libre, consiguiendo con esto que el proyecto no resulte costoso en su conjunto.

Se ha comprobado que las posibilidades de ampliación son muy variadas, el sistema nos posibilita poder crecer en todos los sentidos, tanto en la monitorización del paciente, como en las posibilidades remotas de actuación y desarrollo de aplicaciones que nos permitan el control de distintos dispositivos y el tratamiento de los datos que estamos recibiendo.

En la parte de comunicaciones se ha utilizado la interconexión de los sistemas vía WiFi y cable serie, con lo que nos permite su instalación en casi cualquier parte de forma sencilla. También se ha utilizado en el proyecto la tecnología IoT para el envío y recepción de datos a través de Internet, con lo que nos aporta mayor versatilidad a la hora de controlar remotamente el sistema.

A nivel personal y académico, ha sido un reto enriquecedor, ya que me ha supuesto el tener que estudiar diversas tecnologías (sensores, comunicaciones, plataformas de desarrollo, lenguajes de programación, etc.), para poder desarrollar un sistema que funcione en su conjunto y cuyos resultados han superado las expectativas que tenía puestas en el proyecto.

4.3- Conclusions

This project was born with the idea of creating a system, capable of helping in the caring of people who are isolated in rooms, due to contagious diseases such as COVID-19. During the development of the project, it has been possible to demonstrate that throgh the use of low-cost sensors, and development platforms such as Arduino and Raspberry, it can be built a prototype that allows us to know the state of health from people and act remotely, substantially improving safety for the patient and his caregivers. It has been attempted to find the necessary elements, which integrate the prototype, these are cheaper and affordable, in addition to free software that make the project not very costly.

It has been proven that the possibilities of expansion are very diverse, the system can be expanded in many ways in all senses, both in monitoring patients, as in the remote possibilities of action and development of applications that allow us to control different devices and the processing of the data we are receiving.

In the communications part, it has been used and interconnection between WiFi and serial cable, allowing us to install it in almost any place easily. IoT technology has also been used to send and receive data over the Internet, giving us greater versatility when it comes to controlling the system remotely.

At a personal and academic level, it has been an enriching challenge, since I have had to study several technologies (sensors, communications, development platforms, etc.), in order to be able to develop a system that works as a whole and whose results have exceeded the expectations placed on the project.

5.- Referencias y documentación técnica

5.1- Referencias

[1]: José G. Abreu Murta (Enero 2018) https://www.flickr.com/photos/jgustavoam/40089095211

[2]: I+D Electrónica https://www.didacticaselectronicas.com/index.php/sistemas-de-desarrollo/raspberry/tarjetas-raspberry/tarjeta-raspberry-pi-4-b-4gb-pi4-tarjetas-de-desarrollo-sistemas-de-desarrollo-minipc-mini-computadores-raspberry-pi-4-modelo-b-de-4gb-detail

[3]: *MicrocontrollersLab https://microcontrollerslab.com/esp32-dht11-dht22-web-server/*

[4]: Arduino Learning *(EJEMPLO DE ARDUINO Y SENSOR DE LUZ AMBIENTAL DIGITAL OPT3001)* http://arduinolearning.com/code/opt3001-digital-ambient-light-sensor-and-arduino-example.php

[5]: Luis Llamas (2020) https://www.luisllamas.es/pulsimetro-y-oximetro-con-arduino-y-max30102/

[6]: *Hackster.io (Diciembre 2020) https://www.hackster.io/rswm/diy-blood-oximeter-93529b*

[7]: Journal of Physics: Serie de conferencias (Mayo 2020) https://www.researchgate.net/figure/Application-circuit-for-MAX30205_fig5_342235811

[8]: Curso Udemy *"Master en Arduino 2021 ¡Incluye IoT! Internet of Things"* *https://www.udemy.com/course/master-en-arduino/learn/lecture/11863400#questions*

[9]: *DiyIOt (2021)* https://diyi0t.com/relay-tutorial-for-arduino-and-esp8266/

[10]: Kevin Ashton "*(¿Qué es IoT?)"* https://alfaiot.com/

[11]: Luis Llamas (Mayo 2021) *"¿Qué es MQTT? Su importancia como protocolo IoT"* https://www.luisllamas.es/que-es-mqtt-su-importancia-como-protocolo-iot/

[12]: Python >> 3.9.5 Documentation >> The Python Standard Library >> Graphical User Interfaces with Tk *"Tkinter -Python interface to Tcl/Tk"* https://docs.python.org/3/library/tkinter.html

[13]: *"MariaDB Server: The open source relational database"* https://mariadb.org/

[14]: *"Dbeaver Community"* https://dbeaver.io/

[15]: Bai, L.; Yang, D.; Wang, X.; Tong, L.; Zhu, X.; Zhong, N.; Bai, C.; Powell, C.A.; Chen, R.; Zhou, J.; et al. "*Chinese experts' consensus on the Internet of Things-aided diagnosis and treatment of coronavirus disease 2019 (COVID-19)*". Clin. EHealth 2020, 3, 7–15. https://www.sciencedirect.com/science/article/pii/S2588914120300046

[16]: "Mostrar secuencia de webcam Tkinter" Living-sun.com https://living-sun.com/es/opencv/602709-show-webcam-sequence-tkinter-opencv-tkinter.html

[17]: [Luis Llamas, octubre de 2019, *"Como instalar MariaDB en Raspberry Pi"* https://www.luisllamas.es/como-instalar-mariadb-en-raspberry-pi/]

[18]: *[Daniel Elizalde "Las 5 capas de la tecnología IoT" https://danielelizalde.com/iot-primer/]*

[19]: [*Noticias ONU, (Noviembre 2020)* https://news.un.org/es/story/2020/11/1483412]

[20]: [Stahie, S. COVID-19 Pandemic Increased IoT Adoption, Research Finds. Available online: *https://www.bitdefender.com/box/blog/iot-news/covid-19-pandemic-increased-iot-adoption-research-finds/* (accessed on 30 May 2021).]

[21]: [Eva Rodríguez de Luis (Agosto 2020): *https://www.xataka.com/seleccion/que-modelo-raspberry-pi-comprar-repaso-a-principales-placas-proyectos-habituales-para-dar-mejor*]

[22]: [“Raspberry Pi, breve guía, modelos y características” https://descubrearduino.com/breve-guia-de-la-raspberry-pi/]

[23]: [Harshvardhan Mishra, feb 2021, “MQTT Servers & Brokers List” https://iotbyhvm.ooo/mqtt-servers-brokers-list/]

[24]: [Raúl Casado EFESALUD, abril 2020, “Respiradores frente al COVID-19: Diferentes tipos para cada situación” *https://www.efesalud.com/respiradores-covid-19-tipos-funciones/*]

[25]: [Dan Tynan , marzo 2021, “Cómo la ventaja está remodelando la atención médica” https://www.hpe.com/us/en/insights/articles/how-the-edge-is-reshaping-healthcare-2103.html?jumpid=in_510404507_AroundHPE_NXTedgerehalthcae-042021]

[26]: [Roseli Andrion , agosto 2019, “¿Qué es la computación en el borde?” https://olhardigital.com.br/es/2019/08/13/internet-e-redes-sociais/ainda-nao-sabe-o-que-e-edge-computing-a-gente-conta-para-voce/]

5.2- Documentación técnica

Se adjuntan datasheets de los sensores biométricos MAX30102 y MAX30205.

Absolute Maximum Ratings

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

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

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

Package Information

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

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

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

Electrical Characteristics

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

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

Click here for production status of specific part numbers.

MAX30102

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

General Description

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

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

Applications

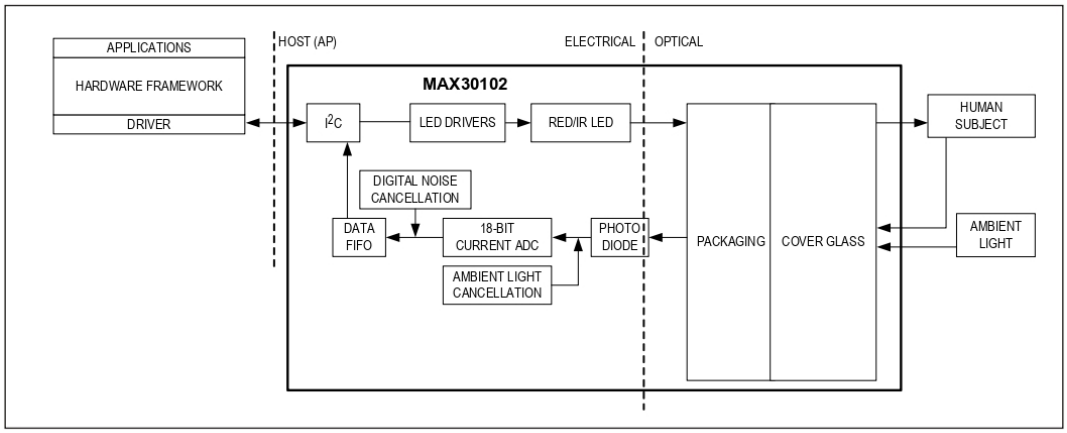
- Wearable Devices
- Fitness Assistant Devices
- Smartphones
- Tablets

Benefits and Features

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

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

System Diagram



19-7740; Rev 1; 10/18

maxim integrated™

Electrical Characteristics (continued)

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

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

Electrical Characteristics (continued)

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

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

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

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

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

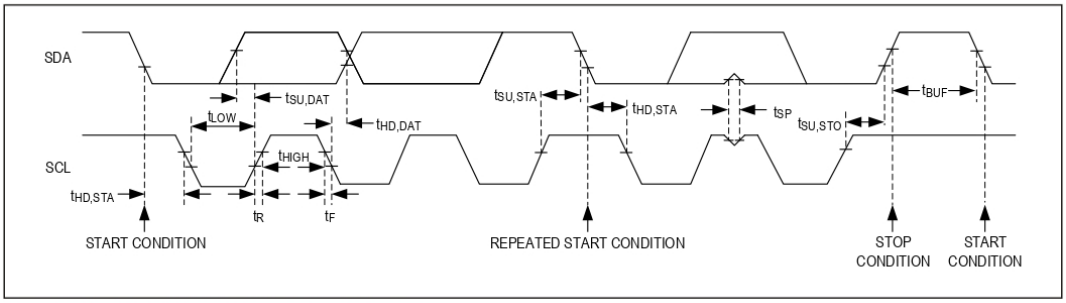
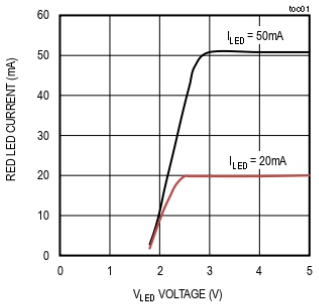


Figure 1. I²C-Compatible Interface Timing Diagram

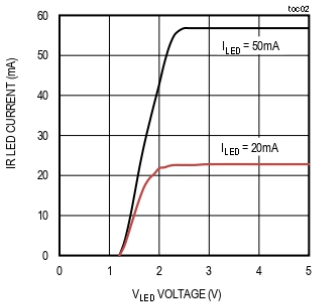
Typical Operating Characteristics

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

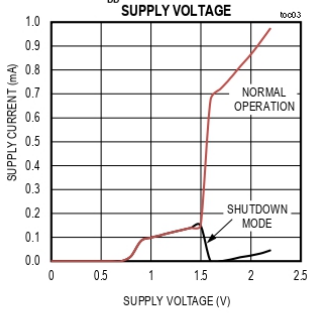
RED LED SUPPLY HEADROOM



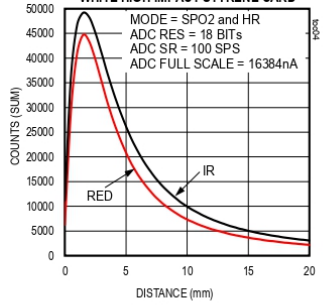
IR LED SUPPLY HEADROOM



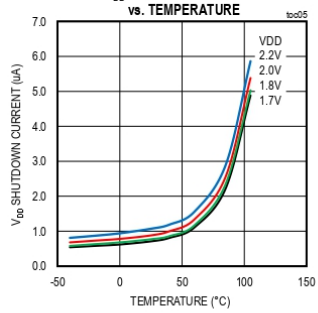
V_{DD} SUPPLY CURRENT vs. SUPPLY VOLTAGE



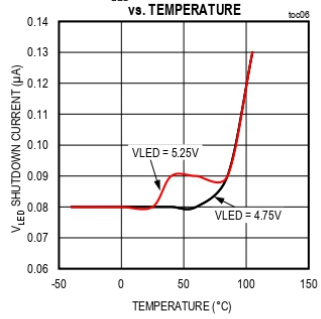
DC COUNTS vs. DISTANCE FOR WHITE HIGH-IMPACT STYRENE CARD



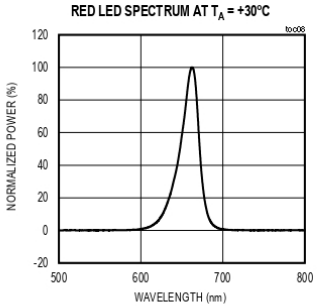
V_{DD} SHUTDOWN CURRENT vs. TEMPERATURE



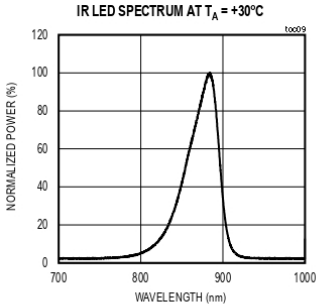
V_{LED} SHUTDOWN CURRENT vs. TEMPERATURE



RED LED SPECTRUM AT T_A = +30°C



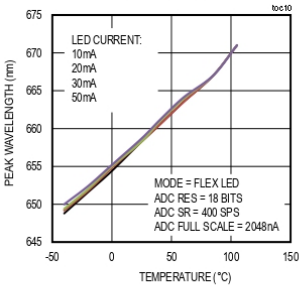
IR LED SPECTRUM AT T_A = +30°C



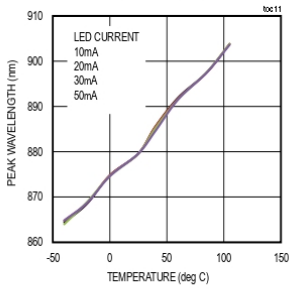
Typical Operating Characteristics (continued)

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

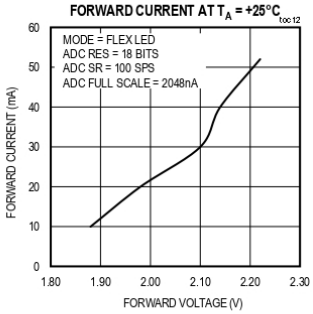
RED LED PEAK WAVELENGTH vs. TEMPERATURE



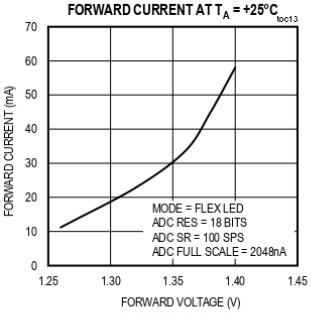
IR LED PEAK WAVELENGTH vs. TEMPERATURE



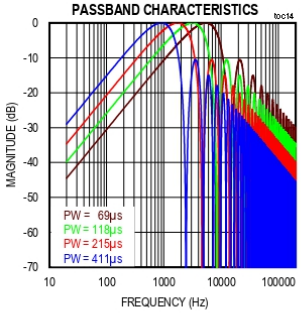
RED LED FORWARD VOLTAGE vs. FORWARD CURRENT AT T_A = +25°C



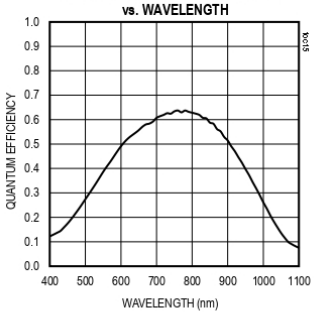
IR LED FORWARD VOLTAGE vs. FORWARD CURRENT AT T_A = +25°C



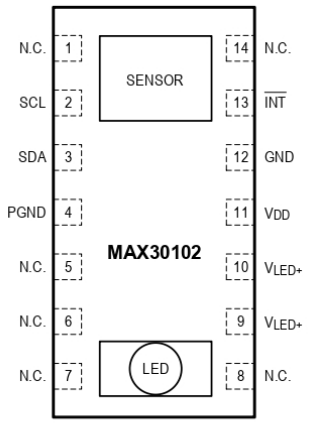
AMBIENT LIGHT CANCELLATION PASSBAND CHARACTERISTICS



PHOTODIODE QUANTUM EFFICIENCY vs. WAVELENGTH



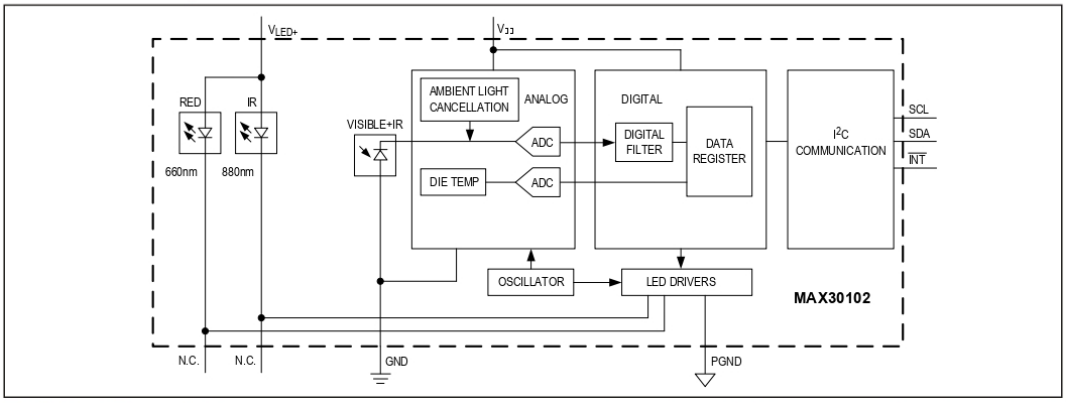
Pin Configuration



Pin Description

PIN	NAME	FUNCTION
1, 5, 6, 7, 8, 14	N.C.	No Connection. Connect to PCB pad for mechanical stability.
2	SCL	I²C Clock Input
3	SDA	I²C Data, Bidirectional (Open-Drain)
4	PGND	Power Ground of the LED Driver Blocks
9	V_{LED+}	LED Power Supply (anode connection). Use a bypass capacitor to PGND for best performance.
10	V_{LED+}	
11	V_{DD}	Analog Power Supply Input. Use a bypass capacitor to GND for best performance.
12	GND	Analog Ground
13	$\overline{INT}$	Active-Low Interrupt (Open-Drain). Connect to an external voltage with a pullup resistor.

Functional Diagram



Detailed Description

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

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

SpO_2 Subsystem

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

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

Temperature Sensor

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

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

LED Driver

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

Register Maps and Descriptions

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

Register Maps and Descriptions (continued)

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

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

Interrupt Status (0x00–0x01)

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

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

A_FULL: FIFO Almost Full Flag

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

PPG_RDY: New FIFO Data Ready

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

ALC_OVF: Ambient Light Cancellation Overflow

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

PWR_RDY: Power Ready Flag

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

DIE_TEMP_RDY: Internal Temperature Ready Flag

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

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

Interrupt Enable (0x02-0x03)

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

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

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

FIFO (0x04–0x07)

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

FIFO Write Pointer

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

FIFO Overflow Counter

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

FIFO Read Pointer

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

FIFO Data Register

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

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

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

Reading from the FIFO

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

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

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

FIFO Data Structure

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

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

Table 1. FIFO Data is Left-Justified

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

FIFO Data Contains 3 Bytes per Channel

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

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

Write/Read Pointers

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

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

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

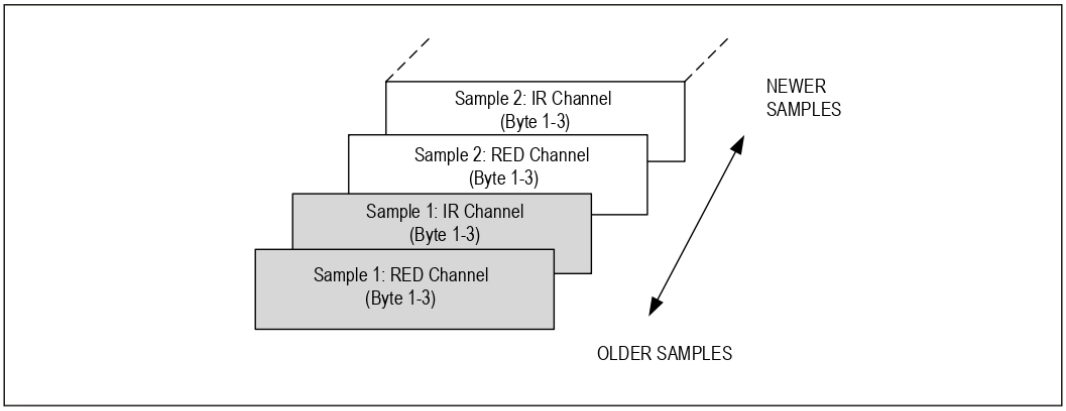


Figure 2. Graphical Representation of the FIFO Data Register. It shows IR and Red in SpO_2 Mode.

Pseudo-Code Example of Reading Data from FIFO

First transaction: Get the FIFO_WR_PTR:

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

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

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

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

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

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

FIFO Configuration (0x08)

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

Bits 7:5: Sample Averaging (SMP_AVE)

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

Table 3. Sample Averaging

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

Bit 4: FIFO Rolls on Full (FIFO_ROLLOVER_EN)

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

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

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

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

Mode Configuration (0x09)

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

Bit 7: Shutdown Control (SHDN)

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

Bit 6: Reset Control (RESET)

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

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

Bits 2:0: Mode Control

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

Table 4. Mode Control

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

SpO_2 Configuration (0x0A)

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

Bits 6:5: SpO_2 ADC Range Control

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

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

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

Bits 4:2: SpO_2 Sample Rate Control

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

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

Table 6. SpO_2 Sample Rate Control

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

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

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

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

Table 7. LED Pulse Width Control

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

LED Pulse Amplitude (0x0C–0x0D)

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

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

Table 8. LED Current Control

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

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

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

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

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

Table 9. Multi-LED Mode Control Registers

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

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

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

Temperature Data (0x1F–0x21)

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

Temperature Integer

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

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

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

Table 10. Temperature Integer

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

Temperature Fraction

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

Temperature Enable (TEMP_EN)

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

Applications Information

Sample Rate and Performance

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

Power Considerations

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

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

Table 11. SpO_2 Mode (Allowed Settings)

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

Table 12. HR Mode (Allowed Settings)

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

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

SpO_2 Temperature Compensation

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

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

Red LED Current Settings vs. LED Temperature Rise

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

Interrupt Pin Functionality

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

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

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

Timing for Measurements and Data Collection

Slot Timing in Multi-LED Modes

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

Table 14. Slot Timing

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

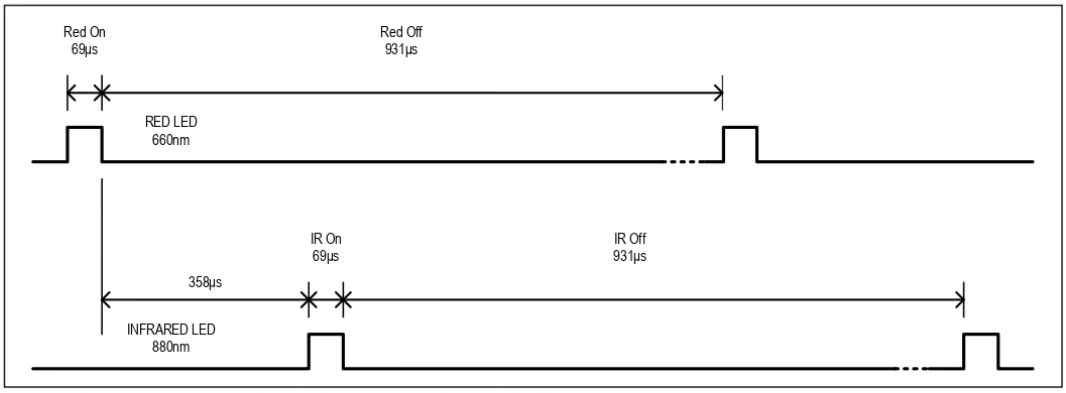


Figure 3. Channel Slot Timing for the SpO_2 Mode with a 1kHz Sample Rate

Timing in SpO_2 Mode

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

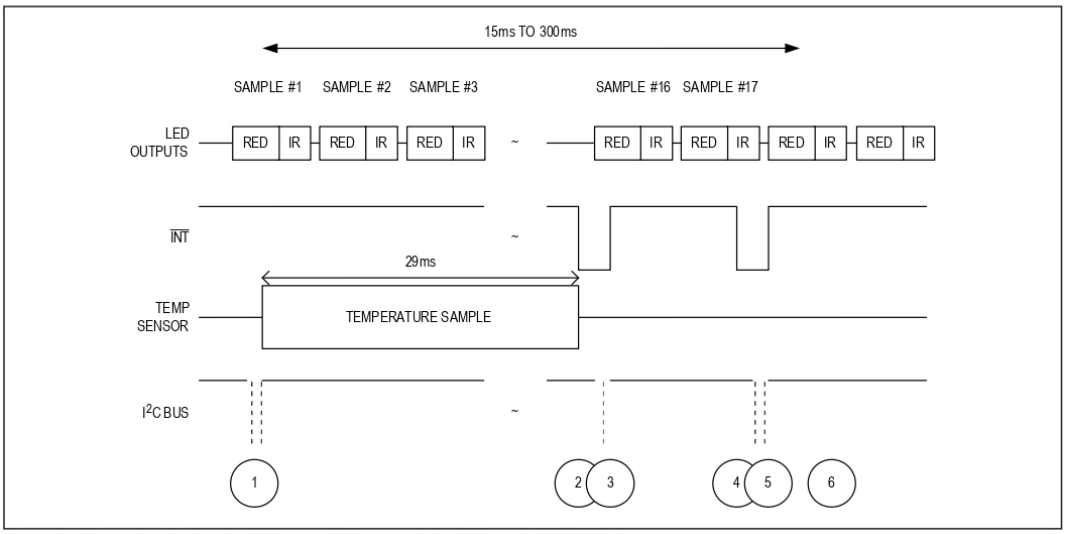


Figure 4. Timing for Data Acquisition and Communication When in SpO_2 Mode

Table 15. Events Sequence for Figure 4 in SpO_2 Mode

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

Timing in HR Mode

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

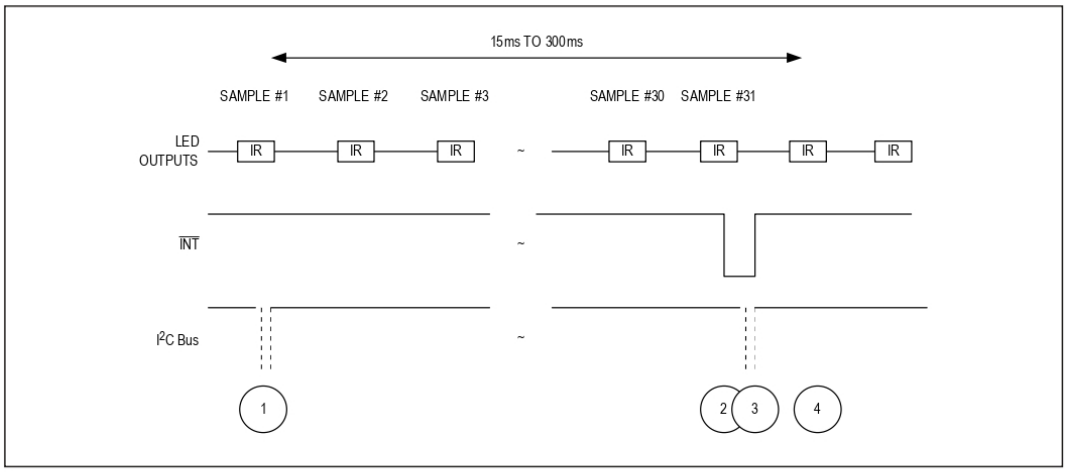


Figure 5. Timing for Data Acquisition and Communication When in HR Mode

Table 16. Events Sequence for Figure 5 in HR Mode

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

Power Sequencing and Requirements

Power-Up Sequencing

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

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

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

Power-Down Sequencing

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

I²C Interface

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

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

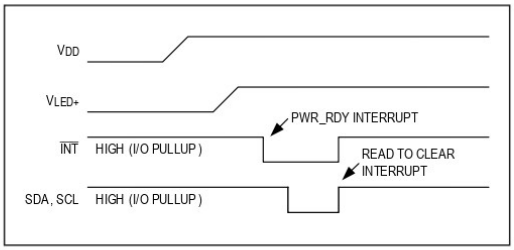


Figure 6. Power-Up Sequence of the Power Supply Rails

Bit Transfer

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

START and STOP Conditions

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

Early STOP Conditions

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

Slave Address

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

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

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

Acknowledge

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

Write Data Format

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

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

Table 17. Slave ID Description

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

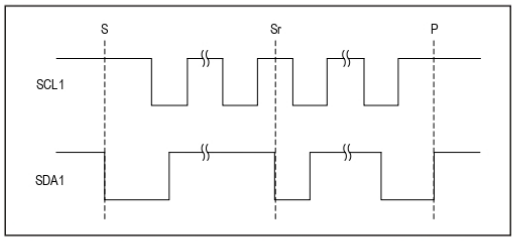


Figure 7. START, STOP, and REPEATED START Conditions

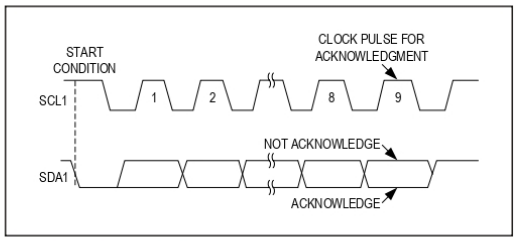


Figure 8. Acknowledge

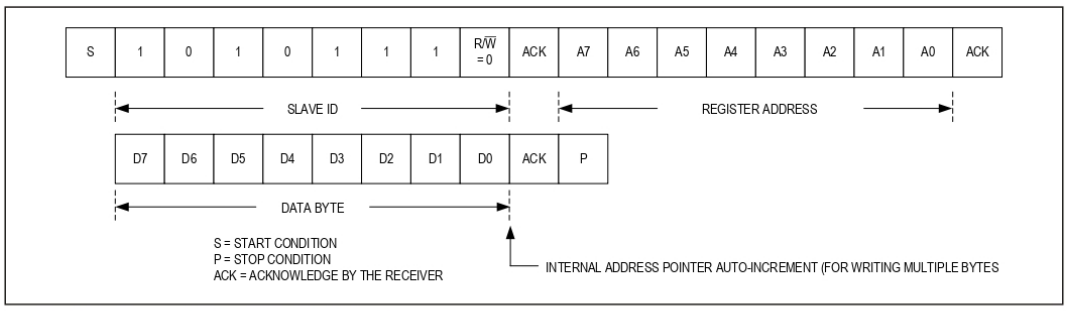


Figure 9. Writing One Data Byte to the MAX30102

Read Data Format

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

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

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

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

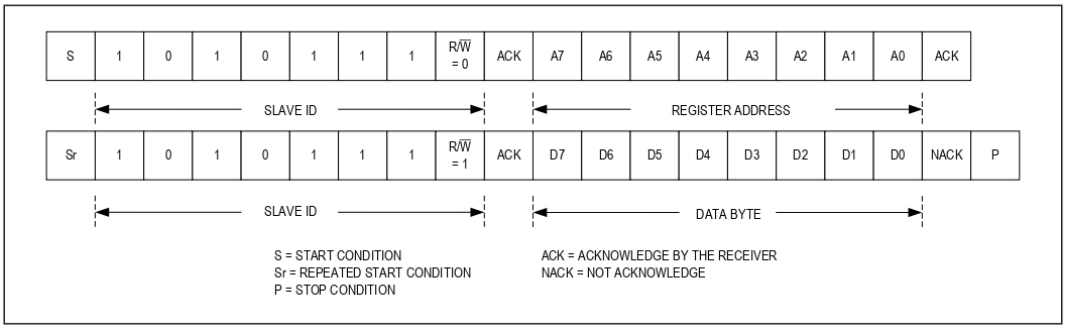


Figure 10. Reading One Byte of Data from MAX30102

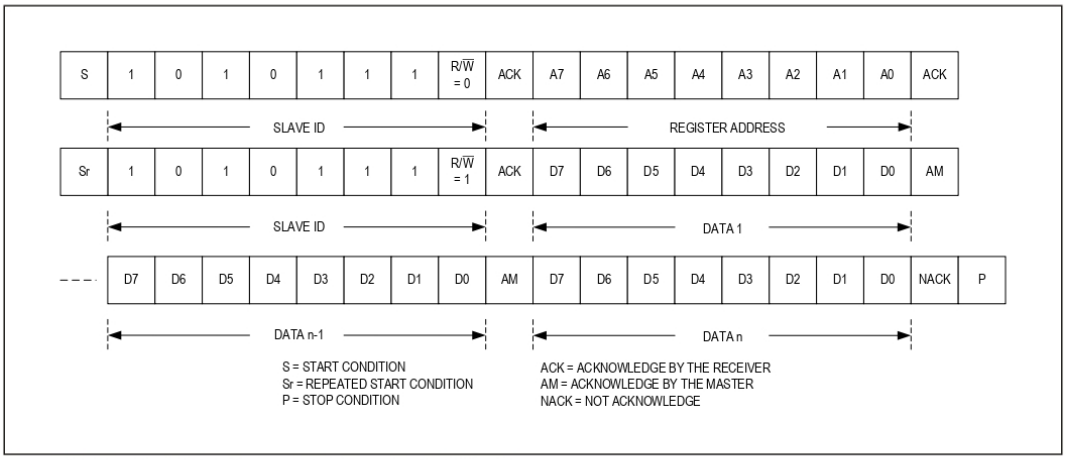
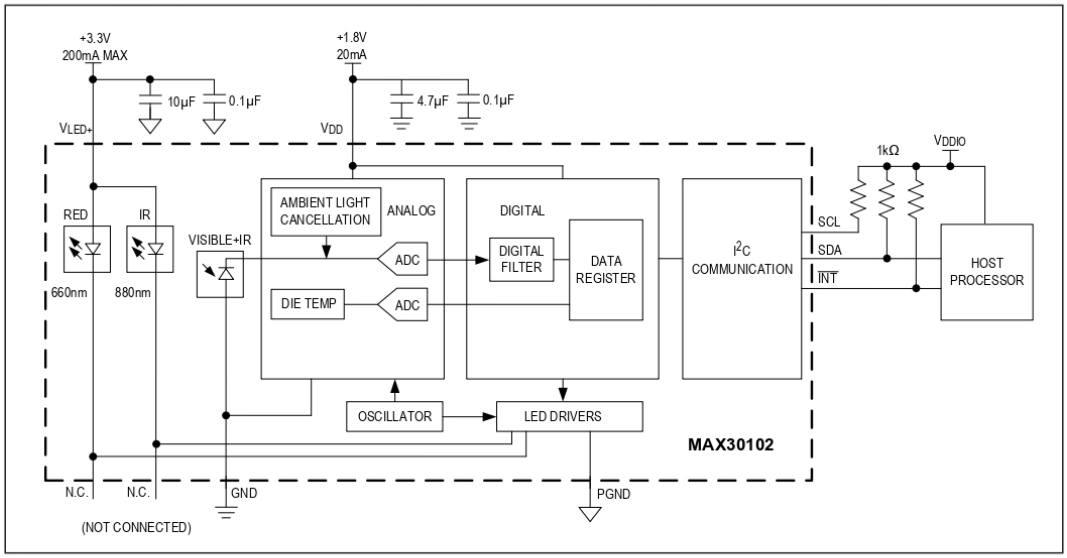


Figure 11. Reading Multiple Bytes of Data from the MAX30102

Typical Application Circuit



Ordering Information

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

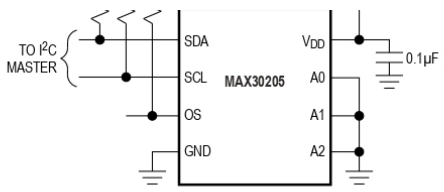
+Denotes lead(Pb)-free/RoHS-compliant package.
T = Tape and reel.

Revision History

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

For pricing, delivery, and ordering information, please visit Maxim Integrated's online storefront at https://www.maximintegrated.com/en/storefront/storefront.html.

Maxim Integrated cannot assume responsibility for use of any circuitry other than circuitry entirely embodied in a Maxim Integrated product. No circuit patent licenses are implied. Maxim Integrated reserves the right to change the circuitry and specifications without notice at any time. The parametric values (min and max limits) shown in the Electrical Characteristics table are guaranteed. Other parametric values quoted in this data sheet are provided for guidance.



Absolute Maximum Ratings

(All voltages relative to GND.)
Voltage Range on V_{DD}, SDA, SCL, A0, A1 .....-0.3V to +4V
Voltage Range on A2, OS ..... -0.3V to (V_{DD} + 0.3V)
Input Current at Any Pin .....+5mA
Package Input Current .....+20mA
Continuous Power Dissipation (T_A = +70°C)
TDFN (derate 24.4mW/°C above +70°C).....1951.2mW

ESD Protection (All Pins, Human Body Model) (Note 1) ... ±4000V
Operating Temperature Range.....0°C to +50°C
Junction Temperature.....+50°C
Storage Temperature Range..... -10°C to +50°C
Lead Temperature (soldering, 10s) .....+300°C
Soldering Temperature (reflow) .....+260°C

Note 1: Human Body Model, 100pF discharged through a 1.5kΩ resistor.

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

Package Thermal Characteristics (Note 2)

TDFN
Junction-to-Ambient Thermal Resistance (θ_{JA}) .....41°C/W
Junction-to-Case Thermal Resistance (θ_{JC}).....8°C/W

Note 2: Package thermal resistances were obtained using the method described in JEDEC specification JESD51-7, using a four-layer board. For detailed information on package thermal considerations, refer to www.maximintegrated.com/thermal-tutorial.

Recommended Operating Conditions

(T_A = 0°C to +50°C, unless otherwise noted.) (Notes 3, 4)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Operating Supply Voltage	V_{DD}		2.7	3.0	3.3	V
Input High Voltage	V_{IH}		V_{DD} x 0.7			V
Input Low Voltage	V_{IL}				V_{DD} x 0.3	V

Electrical Characteristics

(V_{DD} = 2.7V to 3.3V, T_A = 0°C to +50°C, unless otherwise noted. Typical values are V_{DD} = 3.0V, T_A = +25°C.) (Note 3)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Thermometer Error (Note 5)	T_{ERR}	0°C to +15°C	-0.5		+0.5	°C
		+15°C to +35.8°C	-0.3		+0.3	
		+35.8°C to +37°C	-0.2		+0.2	
		+37°C to +39°C	-0.1		+0.1	
		+39°C to +41°C	-0.2		+0.2	
		+41°C to +45°C	-0.3		+0.3	
		+45°C to +50°C	-0.5		+0.5	
ADC Repeatability	T_{repeat}	1 Sigma		0.009		°C
Temperature Data Resolution				16		Bits
Conversion Time				44	50	ms
First Conversion Completed		Data ready after POR			50	ms

Electrical Characteristics (continued)

(V_{DD} = 2.7V to 3.3V, T_A = 0°C to +50°C, unless otherwise noted. Typical values are V_{DD} = 3.0V, T_A = +25°C.) (Note 3)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Quiescent Supply Current	I_{DD}	I^2C inactive, T_A = 0°C to +50°C		600	925	µA
		Shutdown mode, I^2C inactive, T_A = 0°C to +50°C		1.65	3.5	
OS Delay		Depends on fault queue setting	1		6	Conversions
T_{OS} Default Temperature		Factory default setting		80		°C
T_{HYST} Default Temperature		Factory default setting		75		°C
POR Voltage Threshold				2.26		V
POR Hysteresis				130		mV
Input-High Leakage Current	I_{IH}	V_{IN} = 3.3V (all digital inputs)		0.005	1	µA
Input-Low Leakage Current	I_{IL}	V_{IN} = 0V (all digital inputs)		0.005	1	µA
Input Capacitance		All digital inputs		5		pF
Output-High Leakage Current		V_{IN} = 3.3V (SDA and OS)			1	µA
OS Output Saturation Voltage		I_{OUT} = 4.0mA			0.8	V
Output Low Voltage		I_{OL} = 3mA (SDA)			0.4	V

I²C AC Electrical Characteristics

(V_{DD} = 2.7V to 3.3V, T_A = 0°C to +50°C, unless otherwise noted. Typical values are V_{DD} = 3.0V, T_A = +25°C.) (Notes 3, 6) (Figure 1)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Serial Clock Frequency	f_{SCL}	(Note 7)	DC		400	kHz
Bus Free Time Between STOP and START Conditions	t_{BUF}		1.3			µs
START Condition Hold Time	$t_{HD:STA}$		0.6			µs
STOP Condition Setup Time	$t_{SU:STO}$	90% of SCL to 10% of SDA	600			ns
Clock Low Period	t_{LOW}		1.3			µs
Clock High Period	t_{HIGH}		0.6			µs
START Condition Setup Time	$t_{SU:STA}$	90% of SCL to 90% of SDA	100			ns
Data Setup Time	$t_{SU:DAT}$	10% of SDA to 10% of SCL	100			ns
Data Out Hold Time	t_{DH}	(Note 8)	100			ns
Data In Hold Time	$t_{HD:DAT}$	10% of SCL to 10% of SDA (Note 8)	0		0.9	µs
Maximum Receive SCL/SDA Rise Time	t_R	(Note 9)		300		ns
Minimum Receive SCL/SDA Rise Time	t_R	(Note 9)		$20 + 0.1C_B$		ns
Maximum Receive SCL/SDA Fall Time	t_F	(Note 9)		300		ns
Minimum Receive SCL/SDA Fall Time	t_F	(Note 9)		$20 + 0.1C_B$		ns

I²C AC ELECTRICAL CHARACTERISTICS (continued)

(V_{DD} = 2.7V to 3.3V, T_A = 0°C to +50°C, unless otherwise noted. Typical values are V_{DD} = 3.0V, T_A = +25°C.) (Notes 3, 6) (Figure 1)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Transmit SDA Fall Time	t_F	(Note 9)	20 + 0.1C_B		250	ns
Pulse Width of Suppressed Spike	t_{SP}	(Note 10)	0		50	ns
SDA Time Low for Reset of Serial Interface	$t_{TIMEOUT}$	(Note 7)	45	50	55	ms

Note 3: Limits are 100% production tested at T_A = +25°C. Limits over the operating temperature range and relevant supply voltage range are guaranteed by design and characterization. Typical values are not guaranteed.
Note 4: All voltages are referenced to ground. Currents entering the IC are specified positive.
Note 5: These limits represent a 6-sigma distribution of shipped devices and a 3-sigma distribution when these devices are soldered down on the PCB. Sample period > 10s to eliminate self-heating effects.
Note 6: All timing specifications are guaranteed by design.
Note 7: Holding the SDA line low for a time greater than $t_{TIMEOUT}$ causes the devices to reset SDA to the idle state of the serial bus communication (SDA released).
Note 8: A master device must provide a hold time of at least 300ns for the SDA signal to bridge the undefined region of SCL's falling edge.
Note 9: C_B = total capacitance of one bus line in pF. Tested with C_B = 400pF.
Note 10: Input filters on SDA and SCL suppress noise spikes less than 50ns.

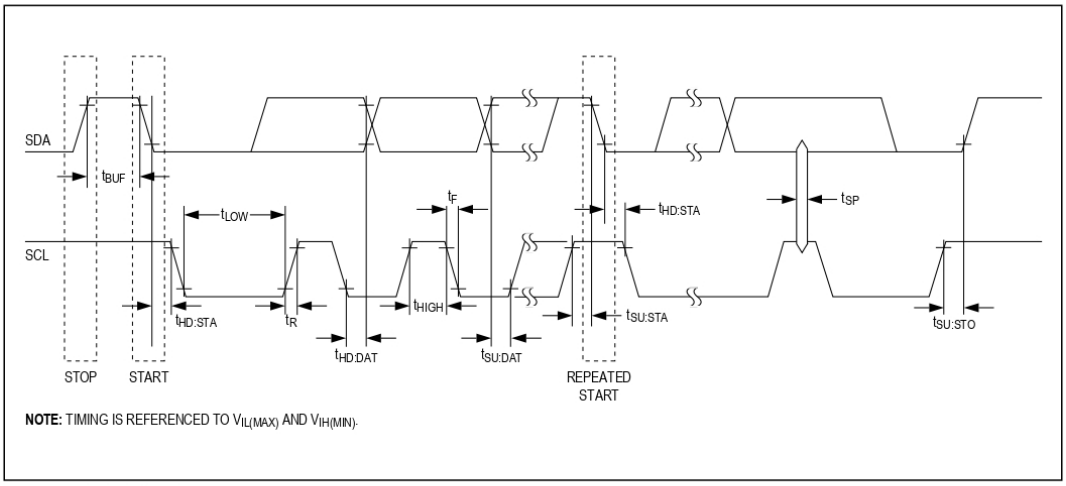
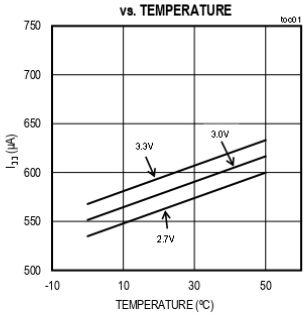


Figure 1. I²C Timing Diagram

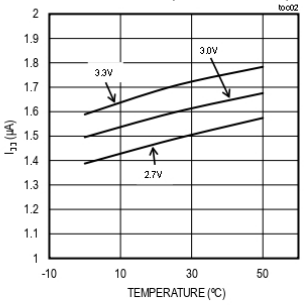
Typical Operating Characteristics

(T_A = +25°C, unless otherwise noted.)

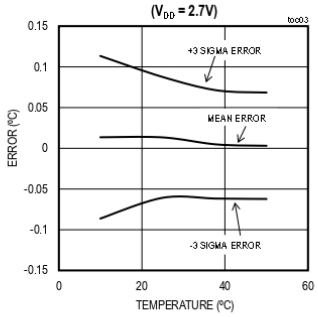
STATIC QUIESCENT SUPPLY CURRENT vs. TEMPERATURE



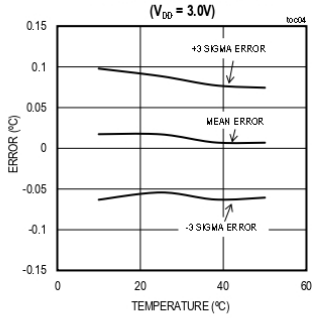
STATIC QUIESCENT SUPPLY CURRENT vs. TEMPERATURE (SHUTDOWN MODE)



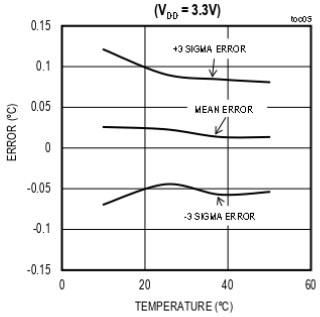
ACCURACY vs. TEMPERATURE (V_{DD} = 2.7V)



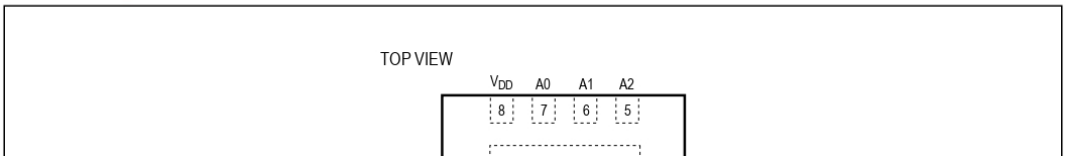
ACCURACY vs. TEMPERATURE (V_{DD} = 3.0V)



ACCURACY vs. TEMPERATURE (V_{DD} = 3.3V)

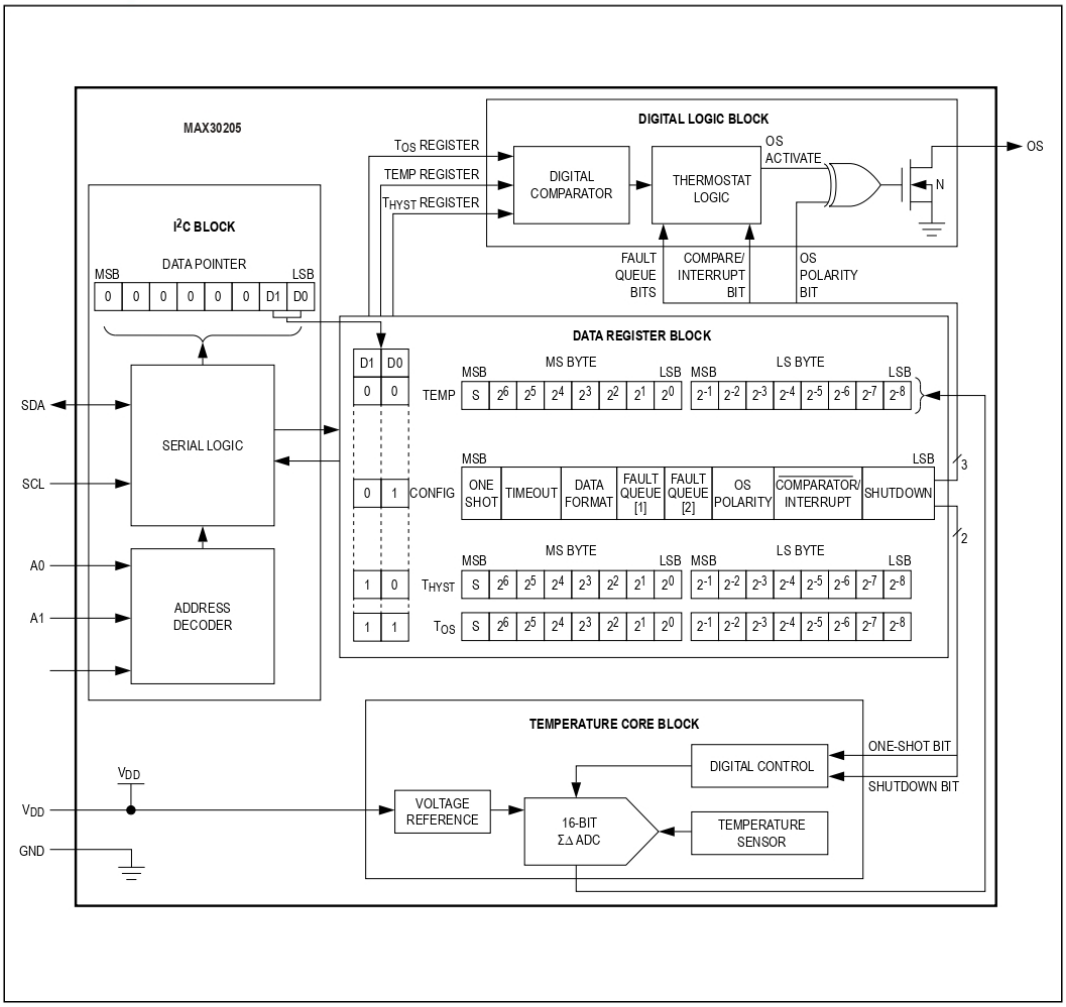


Pin Configuration



MAX30205 Human Body Temperature Sensor

Block Diagram



Detailed Description

The MAX30205 temperature sensor measures temperature and converts the data into digital form. An I²C-compatible two-wire serial interface allows access to conversion results. The device accepts standard I²C commands to read the data, set the overtemperature alarm (OS) trip thresholds, and configure other characteristics. While reading the temperature register, any changes in temperature are ignored until the read is completed. The temperature register is updated for the new temperature measurement upon completion of the read operation.

OS Output, T_{OS} and T_{HYST} Limits

In comparator mode, the OS output behaves like a thermostat (Figure 2). The output asserts when the temperature rises above the limit set in the T_{OS} register. The output deasserts when the temperature falls below the limit set in the T_{HYST} register. In comparator mode, the OS output can be used to turn on a cooling fan, initiate an emergency shutdown signal, or reduce system clock speed.

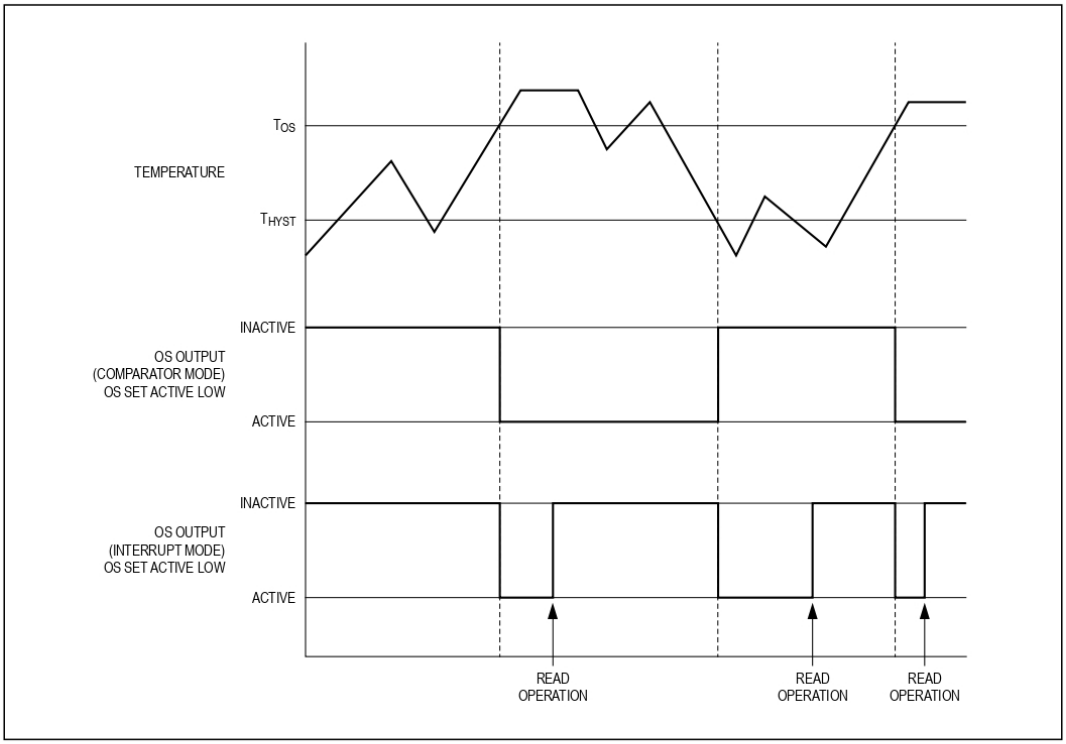


Figure 2. OS Output Temperature Response Diagram

Table 1. MAX30205 Slave Address Selection

A2 CONNECTION	A1 CONNECTION	A0 CONNECTION	SLAVE ADDRESS BYTE (hex)
GND	GND	GND	90h

In interrupt mode, exceeding T_{OS} also asserts OS. OS remains asserted until a read operation is performed on any of the registers. Once OS has asserted due to crossing above T_{OS} and is then reset, it is asserted again only when the temperature drops below T_{HYST}. The output then remains asserted until it is reset by a read. It is then asserted again if the temperature rises above T_{OS}, and so on. Putting the MAX30205 into shutdown mode

MAX30205 Human Body Temperature Sensor

Internal Registers

The device contains four registers, each of which consists of 2 bytes. The configuration register contains only 1 byte of actual data and, when read as a 2-byte register, repeats the same data for the second byte. During a 2-byte write to the configuration register the second byte written takes precedence. The device's pointer register selects between the four data registers shown in Table 2. During reads and writes the pointer register auto increments after every 2 data bytes, but does not wrap from address 03h–00h. The pointer register must be written for each I²C transaction. All registers are read and write, except for the read-only temperature register.

Write to the configuration register by writing the slave address byte, the pointer register byte to value 01h, and a data byte. The T_{OS} and T_{HYST} registers require the slave address byte, pointer register byte, and 2 data bytes. If only 1 data byte is written, it is saved in bits D[15:8] of the respective register. If more than 2 data bytes are written, the pointer register auto increments and if pointing to a valid address, additional data writes to the next address. See Figure 3.

Table 2. Register Functions and POR State

REGISTER NAME	ADDRESS (Hex)	POR STATE		POR STATE (°C)	READ/ WRITE
		Hex	BINARY		
Temperature	00	0000h	0000 0000 0000 0000	0	Read-only
Configuration	01	00h	0000 0000	—	R/W
T_{HYST}	02	4B00h	0100 1011 0000 0000	75	R/W
T_{OS}	03	5000h	0101 0000 0000 0000	80	R/W

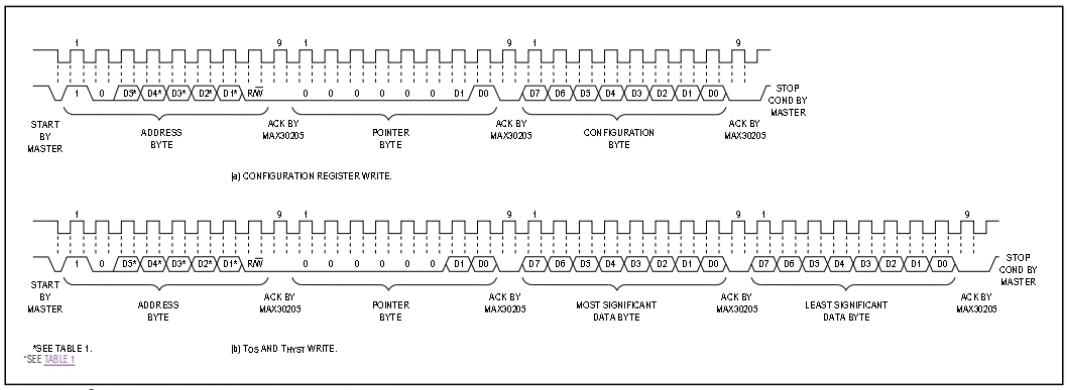


Figure 3. I²C-Compatible Timing Diagram (Write)

Perform a read operation by issuing the slave address byte (write), pointer byte, repeat START, another slave address byte (read), and then reading the data byte. After 2 data bytes the pointer register auto increments and, if pointing to a valid address, additional data can be read. See Figure 4.

Temperature Registers

Temperature data is stored in the temperature, T_{OS} set point, and T_{HYST} set point registers. The temperature data format is 16 bits, two's complement, and the register is read out in 2 bytes: an upper byte and a lower byte. Bits D[15:0] contains the temperature data, with the LSB

MAX30205 Human Body Temperature Sensor

Table 4. Temperature Data Output Format

TEMPERATURE (°C)	NORMAL FORMAT		EXTENDED FORMAT	
	BINARY	Hex	BINARY	Hex
+64	0100 0000 0000 0000	4000h	0000 0000 0000 0000	0000h
+25	0001 1001 0000 0000	1900h	1101 1001 0000 0000	D900h
+0.5	0000 0000 1000 0000	0080h	1100 0000 1000 0000	C080h
0	0000 0000 0000 0000	0000h	1100 0000 0000 0000	C000h

Table 5. Configuration Register Definition

D7	D6	D5	D4	D3	D2	D1	D0
ONE-SHOT	$\overline{\text{TIMEOUT}}$	DATA FORMAT	FAULT QUEUE [1]	FAULT QUEUE [0]	OS POLARITY	$\overline{\text{COMPARATOR}}$/ INTERRUPT	SHUTDOWN

Configuration Register

The configuration register contains 8 bits of data and initiates single conversions (ONE-SHOT), enables bus timeout, controls shutdown, sets the fault queue, sets the data format, selects OS polarity, and determines whether the OS output functions in comparator or interrupt mode. See Table 5.

Shutdown

Set bit D0 to 1 to place the device in shutdown mode and reduce supply current to 3.5µA or less. If bit D0 is set to 1 when a temperature conversion is taking place, the device completes the conversion and then shuts down. In interrupt mode, entering shutdown resets the OS output. While in shutdown, the I^2C interface remains active and all registers remain accessible to the master.

Setting D0 to 0 takes the device out of shutdown and starts a new conversion. The results of this conversion are available to read after the max conversion time.

$\overline{\text{COMPARATOR}}$/INTERRUPT Mode

Set bit D1, the $\overline{\text{COMPARATOR}}$/INTERRUPT bit to 0 to operate OS in comparator mode. In comparator mode, OS is asserted when the temperature rises above the T_{OS} value. OS is deasserted when the temperature drops below the T_{HYST} value. See Figure 2.

Set bit D1 to 1 to operate OS in interrupt mode. In interrupt mode, exceeding T_{OS} also asserts OS. OS remains asserted until a read operation is performed on any of the registers. Once OS has asserted due to crossing above T_{OS} and is then reset, it is asserted again only when the temperature drops below T_{HYST}. The output then remains asserted until it is reset by a read. It is then asserted again if the temperature rises above T_{OS}, and so on. Putting the MAX30205 into shutdown mode also resets OS. Note that if the mode is changed while OS is active, an OS reset may be required before it begins to behave normally.

OS Polarity

Set bit D2, the OS POLARITY bit, to 0 to force the OS output polarity to active low. Set bit D2 to 1 to set the OS output polarity to active high. OS is an open-drain output under all conditions and requires a pullup resistor to output a high voltage. See Figure 2.

Fault Queue

Bits D4 and D3, the fault queue bits, determine the number of faults necessary to trigger an OS condition. See Table 6. The number of faults set in the queue must occur consecutively to trip the OS output. The fault queue prevents OS false tripping in noisy environments.

Table 6. Configuration Register Fault Queue Bits

FAULT QUEUE [1] BIT D4	FAULT QUEUE [0] BIT D3	NUMBER OF FAULTS
0	0	1 (POR state)

Applications Information

The MAX30205 measures the temperature of its own die. The thermal path between the die and the outside world determines the accuracy of temperature measurements. External temperature is conducted to the die primarily through the leads and the exposed pad. Because of this,

MAX30205 Human Body Temperature Sensor

Ordering Information

PART	TEMP RANGE	RESET	TIMEOUT ENABLED AT POR	PIN-PACKAGE
MAX30205MTA+	0°C to +50°C	No	Yes	8 TDFN-EP*

+Denotes a lead(Pb)-free/RoHS-compliant package.
**EP = Exposed pad.*

Package Information

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

PACKAGE TYPE	PACKAGE CODE	OUTLINE NO.	LAND PATTERN NO.
8 TDFN-EP	T833+2	21-0137	90-0059

Revision History

REVISION NUMBER	REVISION DATE	DESCRIPTION	PAGES CHANGED
0	3/16	Initial release	—

For pricing, delivery, and ordering information, please contact Maxim Direct at 1-888-629-4642, or visit Maxim Integrated's website at www.maximintegrated.com.

