industriales
etsii

Escuela Técnica Superior de Ingeniería Industrial

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Escuela Técnica Superior de Ingeniería Industrial

Diseño y programación de un sistema de monitorización de parámetros biomédicos

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Autor: **Mónica Moreno Caballero**
Director: Miguel Almonacid Kroeger
Codirector: Julio José Ibarrola Lacalle

Cartagena, septiembre 2021

Universidad Politécnica de Cartagena

RESUMEN

Este proyecto trata del desarrollo de algunos aspectos de la Industria 4.0 en la medicina.

Se centra en el desarrollo de un monitor de signos vitales controlado por IoT (Internet of Things) y deja las puertas abiertas para una ampliación a la domótica.

Se pretende una visualización de estos signos en cualquier dispositivo, puede ser pantalla fija pero también tablet o smartphone. El objetivo es dar comodidad tanto al personal sanitario como a los pacientes y para eso es importante modernizar el sistema con una minimización del cableado de la instalación y la posibilidad de controlar al paciente desde cualquier parte, lo que puede llevarse a cabo gracias a la comunicación por internet.

DOCUMENTO I: MEMORIA

Mónica Moreno Caballero

ÍNDICE GENERAL

ÍNDICE DE IMÁGENES

1 INTRODUCCIÓN

Este proyecto está orientado al desarrollo de una aplicación en el campo de la medicina. Trata sobre la obtención de datos biomédicos de los pacientes, como pulsaciones, saturación de oxígeno en la sangre, temperatura, etc. Esto se obtendría por medio de los sensores necesarios y, quedaría reflejado en una pantalla móvil a través de una aplicación, que sería la herramienta médica.

Además de visualizar estos datos en tiempo real, almacenaría las constantes detectadas, quedando guardadas a disposición del médico por si creyera necesaria su consulta.

La herramienta para distinguir entre pacientes sería la lectura de un código QR personalizado para cada habitación.

1.1 Motivación

Este proyecto está impulsado por dos ramas de gran importancia en nuestras vidas. La primera de ellas es la ingeniería biomédica, los avances en este sector permiten mejorar la calidad de vida de los humanos. La aplicación de los avances tecnológicos en ambientes médicos es una labor necesaria y muy interesante.

Aplicaciones comunes son los respiradores, escáneres, ecografías, un marcapasos, monitores de signos vitales, etc.

Por otro lado, el internet de las cosas. El desarrollo general de la automatización está en pleno auge y es un campo muy útil y atractivo.

Tiene una gran cantidad de ventajas como es: la facilidad para las conexiones remotas, lo que facilita el control mediante tus dispositivos desde cualquier lugar, la reducción de costes, mejora del servicio al cliente, en este caso la atención sería mucho más directa ya que no es necesario la presencia del sanitario para consultar la salud del paciente.

1.2 Objetivos

El objetivo principal que se quiere conseguir con la realización de este proyecto es la investigación sobre el diferente hardware utilizado en el sector médico y como poder mejorarlo.

Para ello, según el hardware y software escogido tenemos las siguientes tareas:

- Ampliar conocimientos sobre programación y uso de Arduino y Raspberry Pi.
- Crear un HMI médico cómodo y funcional para una monitorización de los pacientes más ágil.
- Desarrollar una comunicación entre el HMI y los sensores de medida colocados en los pacientes.
- Aprender conocimientos de programación de bases de datos para guardar los datos fisiológicos de los pacientes.
- Investigación sobre tipos de sensores que se pueden usar en estas aplicaciones y su posterior comunicación con otro hardware y la base de datos.
- Realizar un proyecto funcional con posibilidades de ampliación en un futuro.

1.3 Desarrollo del proyecto

La finalidad es aplicar la tecnología para conseguir comodidad y optimización de recursos. Con este desarrollo, se podría ahorrar tener tantas pantallas como pacientes que reflejaran constantes de cada uno de ellos y, únicamente cuando el doctor entrara a examinar a su paciente, introduciendo sus datos, podría ver lo que interesara de esa persona.

1.3.1 *Documentación y búsqueda de información*

El primer paso a desarrollar será, estudiar las distintas plataformas hardware y software, decidir el lenguaje utilizado para la programación y el tipo de microcontrolador.

En cuanto al software, hay que diferenciar entre el programa con el que desarrollar el código de programación, un simulador de señales para aquellas constantes que queramos comprobar que sería posible interpretar si contáramos con el tipo de sensor necesario, de esta manera será fácilmente ampliable. Y, por último, el de diseño de aplicaciones desde donde se podrá visualizar y controlar el proyecto.

El diseño de la aplicación deberá contener:

- Una base de datos de los pacientes con sus características, datos y gráficas asociadas.
- Sistema de visualización amigable.
- Identificación del paciente mediante código QR.

Estudiar los monitores de signos vitales que existen en la actualidad, su funcionamiento, los signos que monitorean, etc. y las ampliaciones y/o modificaciones que se pueden hacer con respecto a estos. Una vez hecho esto, decidir los tipos de sensores que se van a emplear para cada medida.

1.3.2 *Experimentación y pruebas*

Una vez conectados los sensores, se realizarán pruebas de calibración, se compararán las señales con otros dispositivos verificados para aprobar su correcta medición y funcionamiento.

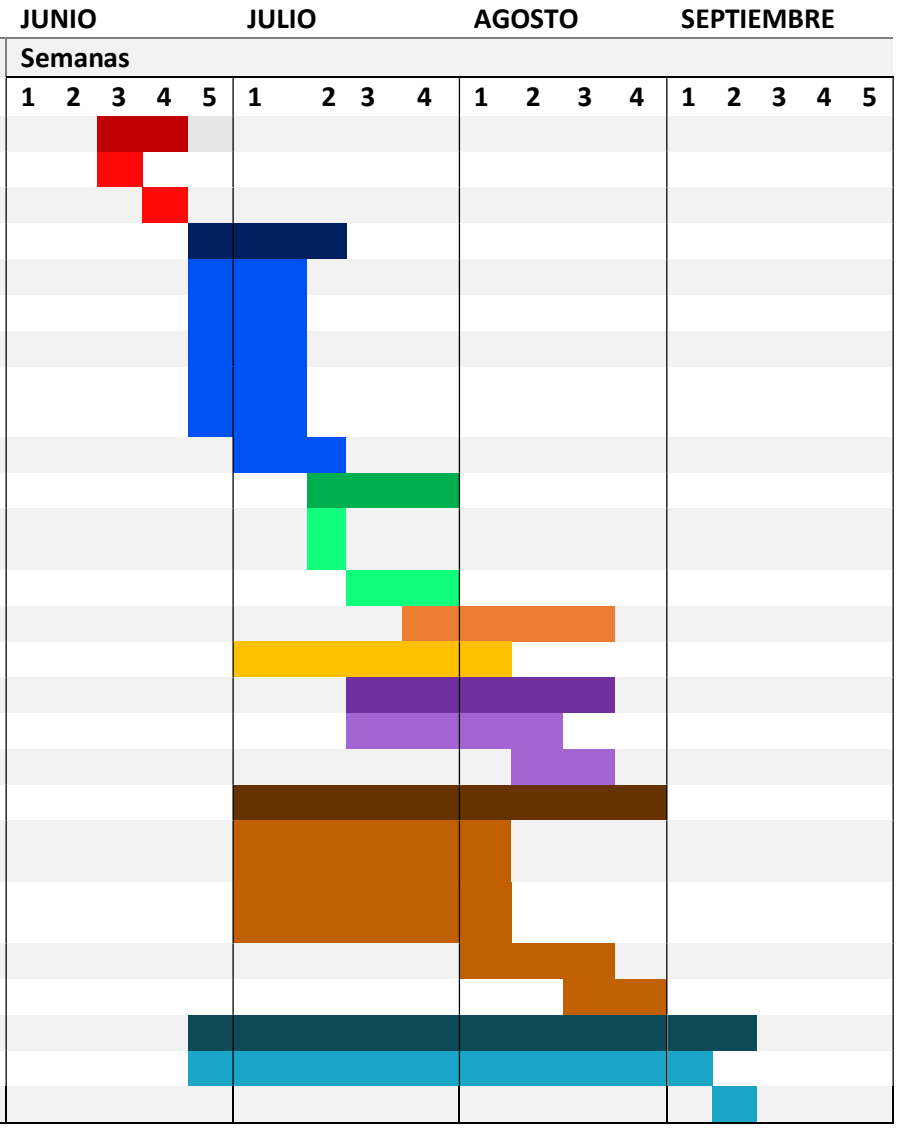
Se comprobará la visualización en el HMI diseñado, comprobando que se reflejan los datos tal y como esperamos.

1.3.3 *Comunicación*

También se diseñará de tal manera que, para la comunicación de los datos con el equipo se implementará la comunicación wifi. Gracias a esto, se podrá contar con gran cantidad de ventajas. Es lo que hace posible el uso de una pantalla móvil, por tanto, ayudará a la reducción de costes, permite la posibilidad de hacer un seguimiento personalizado y en profundidad y ayuda en la identificación el diagnóstico y el tratamiento adecuado [2].

1.4 Cronograma Gantt

NUM	Tarea
1	**Inicio del proyecto**
1.1	Reunión con el tutor del trabajo
1.2	Redactar una propuesta para la aceptación
2	**Investigación**
2.1	Estudiar tipos de plataformas IoT
2.2	Estudiar tipos de equipos de monitoreo
2.3	Estudiar distintos sensores para las medidas
2.4	Estudiar lenguajes de programación compatibles
2.5	Estudiar software compatible
3	**Selección de los componentes**
3.1	Decidir lo que se ajusta más a las características requeridas
3.2	Diseño de la aplicación
4	**Evaluación de riesgos**
5	**Ajuste a las normas según la legislación**
6	**Práctica**
6.1	Programación
6.2	Conexión entre dispositivos
7	**Pruebas**
7.1	Comprobar que la programación se ha implementado correctamente
7.2	Comprobar que la conexión se ha realizado correctamente
7.3	Modificación de errores
7.4	Confirmación de su correcto funcionamiento
8	**Documentación**
8.1	Redactar el proyecto
8.2	Realizar la presentación de la defensa



2 Estado del arte

En este apartado se van a describir los diferentes equipos encontrados en la actualidad para realizar la monitorización como Healthy Pi, monitores de constantes vitales tradicionales, la utilidad de la aplicación de IoT y la domoótica y las características, tipos y aplicación de pulsioxímetros.

2.1 Monitor de constantes vitales

Este dispositivo posibilita de manera ininterrumpida la detección, procesamiento y despliegue de los “parámetros fisiológicos del paciente”. Se compone de un “sistema de alarmas” capaces de alertar cuando se detecta cualquier tipo de situación adversa, así como toda aquella que se encuentre fuera de los límites deseados [36].

Se pueden observar en la pantalla las ondas y/o información numérica de diversos parámetros fisiológicos, a saber: ritmo cardiaco (ECG), presión arterial no invasiva (PANI), pulso, saturación de oxígeno (SPO2), temperatura (TEMP), CO2 y presión arterial invasiva (PAI) [7, 9].

Imagen 1. Monitor de signos vitales

- ***La saturación de oxígeno (SpO2)*** proporciona información acerca de la concentración de oxígeno transportado por la sangre con respecto a su capacidad total. La lectura de referencia en un oxímetro de pulso oscila entre 95 y 100%. [18, 19]

- ***El electrocardiograma (ECG***) es un estudio de rutina que permite observar la actividad eléctrica cardíaca (ciclo cardíaco) a fin de detectar diversas afecciones miocárdicas gracias a electrodos que, al colocarlos en el pecho del paciente, registran las señales eléctricas que originan los latidos cardiacos. Estas señales se muestran como ondas en un monitor, donde serán examinadas [10].

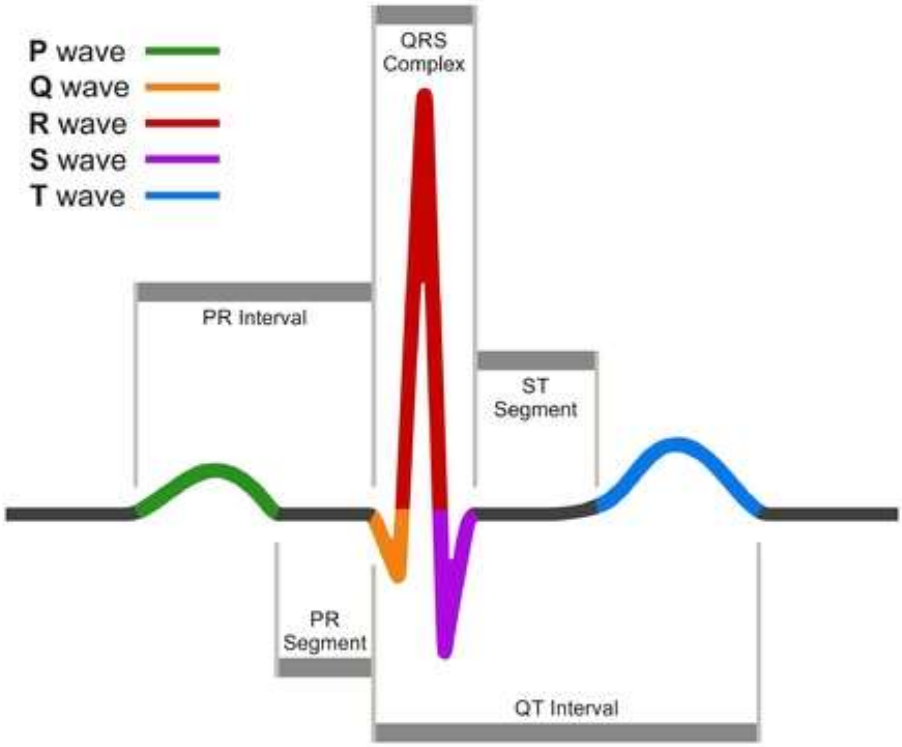


Imagen 2. Electrocardiograma

En la "Imagen 2" se muestra la forma de onda de un electrocardiograma.

- ***Onda P:*** Corresponde con la contracción auricular, esto es, cuando las aurículas se están contrayendo con el fin de transportar la sangre hasta los ventrículos [35].

- ***Segmento PR:*** Durante este intervalo, la cantidad de volumen auricular es prácticamente nula, lo que supone una cierta

"desaceleración en la transmisión de la corriente eléctrica" a lo largo del miocardio, justo antes del inicio de la contracción ventricular [35].

- ***Complejo QRS:*** En este segmento tiene lugar la contracción ventricular, produciéndose el vaciado sanguíneo desde los ventrículos hacia las arterias (pulmonares y aórtica), ergo, la circulación pulmonar y sistémica [35].

- ***Segmento ST:*** Es la fase de repolarización celular, es decir, la recuperación del "potencial de acción de las células cardíacas". Su oscilación se puede considerar un indicador de falta de irrigación miocárdica, más aún si el paciente presenta los síntomas propios de un coinciden con la sintomatología típica asociada a isquemia. Por ello, representa una herramienta diagnóstica insustituible [35].

- ***Onda T:*** Representa la repolarización ventricular, esto es, la relajación del músculo cardíaco. En este período, la sangre ya ha sido enviada al torrente sanguíneo. Esta onda suele tener un valor positivo en comparación con la línea de base. [35].

– La medición de la **presión arterial**, tanto **PAI** como **PANI**, es indudablemente un indicador crucial del estado fisiológico. Se trata de una de las pruebas diagnósticas más frecuentemente usadas, ya que indica con gran precisión la variación en el volumen sanguíneo, la eficiencia de bombeo cardiaco y la resistencia vascular periférica [28].

2.2 HealthyPi

HealthyPi es un complemento para Raspberry. Hace posible la monitorización de constantes vitales de código abierto y tiene la misma funcionalidad que uno tradicional [26].

Al diseñarlo de forma tan compacta y simple de forma que está todo integrado, los resultados que ofrece HealthyPi son altamente fiables y de muy bajo coste

en comparación con los usados en la actualidad. Algunos de los dispositivos compatibles con esta placa son los utilizados en sistemas profesionales de los hospitales [26].

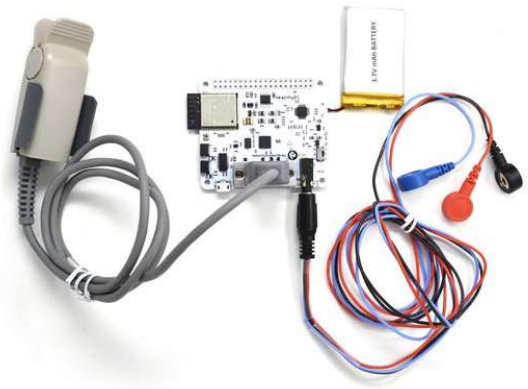


Imagen 3. HealthyPi v4

2.3 IoT (internet de las cosas)

Gracias al "*Internet of things (IoT)"* es posible intercambiar información a través de la red, esto es, establecer una **comunicación** entre dispositivos electrónicos y personas mediante el uso de sensores, módulos wifi, software y otras diferentes tecnologías [24].

En la actualidad hay "más de 10 mil millones de dispositivos IoT conectados y los expertos prevén que este número aumente a **22 mil millones** para 2025" [24].

Uno de los principales objetivos es utilizar dispositivos portátiles a fin de supervisar y controlar analíticamente la salud de la población mundial. Dichos dispositivos IoT permiten a las personas comprender mejor su propia salud y permiten a los sanitarios controlar de manera remota a los pacientes [24].

Imagen 4. Esquema de conexión de dispositivos con IoT

2.4 Domótica

La domótica es la automatización de una sala, una casa, etc. Este tipo de instalación nos da la posibilidad de un control completo del lugar desde un panel o incluso un dispositivo móvil.

Tiene numerosas ventajas como la comodidad, el ahorro energético, instalaciones inalámbricas, seguridad ya que se puede acceder a la información del estado de la sala desde cualquier lugar y comprobar si está todo correcto, bienestar, etc.

Es posible el desarrollo domótico en la Raspberry Pi mediante software como es Home Assistant de código abierto. Este software tiene gran variedad de aplicaciones configurables para esta aplicación y complementos que aumentan la comodidad de su uso como Alexa, Google Assistant, etc.

2.5 Pulsioxímetro

Este instrumento es capaz de medir la saturación de oxígeno en el sistema circulatorio. Gracias a esto es posible estimar la “saturación de oxígeno de la

hemoglobina arterial (SatO2)" indirectamente, es decir, sin requerir la sangre del paciente. Además, permite conocer el número de latidos por minuto (frecuencia cardiaca) y la amplitud del pulso. En clínica es bastante común asociar este objeto a un monitor para que los especialistas sigan la oxigenación del paciente ininterrumpidamente.

Existen varios modelos, entre los que se encuentran los de muñeca, de mesa, de mano y la yema del dedo. Algunos de ellos cuentan con baterías de manera que es posible su uso fuera del hospital [5]. En la "Imagen 5" se muestra un pulsioxímetro de mesa.

Imagen 5. Pulsioxímetro de mesa

Tiene un transductor con dos piezas, un fotodetector y un emisor de luz. Se sirve de la espectrofotometría. Emite luz con dos longitudes de onda, 660 nm, luz roja y 940 nm, infrarroja, características de la oxihemoglobina (HbO2) y la hemoglobina reducida (Hb). La mayor parte de la luz se absorbe constantemente por la piel, hueso y sangre venosa y así se produce un incremento de la absorción en la sangre arterial. Con estas variaciones y el cociente normalizado de luz transmitida (luz roja/infrarroja), el monitor calcula automáticamente un valor de SpO2 [25].

En la situación actual de COVID se han usado estos dispositivos para determinar la gravedad del paciente. Teniendo una medida de oxígeno en la sangre por debajo de 92% es necesario tener vigilancia en un hospital.

Con un simple pulsioxímetro de dedo puede determinarse el cuidado de una persona asintomática o con síntomas de falta de aire, mareos o náuseas y siendo tan pequeño y económico se puede tener en casa.

2.6 Medida de temperatura por infrarrojos

Estos aparatos no solo miden la temperatura de los cuerpos en movimiento si no que, además, funcionan a distancia, evitando así el contacto directo con el objeto que se desea medir, lo que supone una indudable ventaja. Es esta la razón por la que también son conocidos como **termómetros sin contacto**.

Este tipo de instrumentos poseen una mayor **precisión** y **sensibilidad**. De esta manera, si un elemento se encuentra en una posición intermedia entre dicho termómetro y el objeto a medir, este será considerado como un obstáculo ya que puede modificar la lectura de la temperatura, creando así una posibilidad de error.

El fundamento de esta tecnología consiste en que una señal llega a una lentilla situada en el sensor, la cual se amplifica y se transforma dependiendo de la potencia, hasta obtener la temperatura del objeto medido [6].

3 Protocolos de comunicación

Un protocolo es un conjunto de normas que se establecen para que siguiendo ese formato de mensaje sea posible el intercambio de información.

3.1 TCP/IP

El protocolo TCP/IP se utiliza para la comunicación por internet desde principios de los años 80.

Internet es una red de conmutación de paquete. Esto quiere decir que divide la información en partes y cada una se envía de forma individual a la red.

El protocolo de control de transmisión TCP y el protocolo de internet IP hacen que se realice una transmisión sin errores. En la “Imagen 6” se muestra la estructura de este tipo de comunicación.

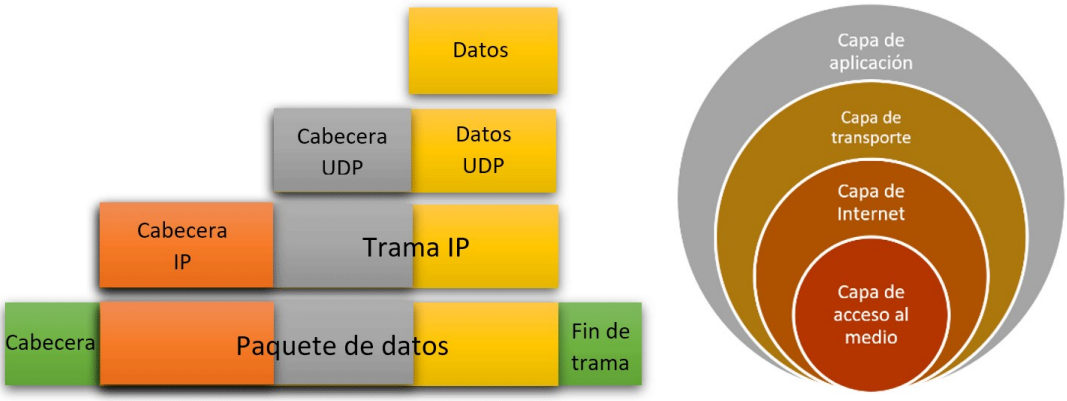


Imagen 6. Estructura TCP/IP

En primer lugar, entra TCP en funcionamiento, divide en fragmentos el documento añadiendo una cabecera a cada uno de ellos. Cada paquete se envía al IP y este añade la información del destinatario. El IP envía cada paquete por la red de forma individual y el receptor los coloca en orden para recomponer el mensaje original. Por último, antes de entregarlo se realiza una suma de chequeo

o checksum para asegurar que los datos que se van a entregar son los correctos y evitar errores [31].

3.2 MQTT

MQTT es un protocolo de mensajería estándar de OASIS usado en la comunicación M2M (machine - to - machine) para internet de las cosas. Está diseñado como un transporte de mensajería de publicación / suscripción extremadamente ligero ideal para conectar dispositivos remotos con un ancho de banda de red limitado. Hoy en día, MQTT se utiliza en una amplia variedad de industrias como la fabricación, las telecomunicaciones, etc. [17].

En sus características tenemos [18]:

- Protocolo de base TCP/IP.
- La comunicación tiene la posibilidad de hacerse de uno a uno o de uno a muchos.
- Usa 14 tipos de mensajes, aunque los más usados son “CONNECT”, “PUBLISH”, “SUBSCRIBE” Y “UNSUBSCRIBE”
- Tiene una cabecera fija y una variable opcional.
- Tres niveles de fiabilidad QoS. Estos son “QoS 0”, no garantiza la llegada del mensaje, “QoS 1”, confirma la llegada con un mensaje y “QoS 2”, con doble confirmación de la llegada.
- Es fácil de implementar, pero tiene complejidad para añadir extensiones.
- Es muy útil para conexiones en localizaciones remotas.
- Usa estándares de seguridad TLS/SSL para el transporte.

3.2.1 *Seguridad MQTT*

La seguridad en MQTT se divide en capas para prevenir ataques.

La división de los niveles es:

- Nivel de red: para conseguir una conexión fiable es necesario una red segura o VPN para la comunicación entre agentes y clientes. Para esto necesitamos un gateway como enlace entre nuestra red y la VPN.

- Nivel de transporte: SSL/TLS se utiliza para la encriptación del transporte para dar confidencialidad a la información. Proporciona una autentificación de certificado de cliente para verificar las identidades.

- Nivel de aplicación: el protocolo proporciona una identificación de cliente, un identificador de cliente y un usuario y una contraseña para autentificar los dispositivos en este nivel [3].

3.3 Bus I2C

El protocolo I2C (Inter-Integrated Circuit) fue desarrollado por Philips en 1982 para la comunicación interna de dispositivos electrónicos en sus artículos. Tuvo mucha aceptación y fue creciendo su aplicación para ser un estándar del mercado.

El bus I2C lleva la información por medio de sólo dos cables. Uno para el envío de datos (SDA) y otro para la señal de reloj (CLK) como se muestra en la "Imagen 7", en el caso del módulo ESP32 se encuentran en los pines 21 y 22 respectivamente.

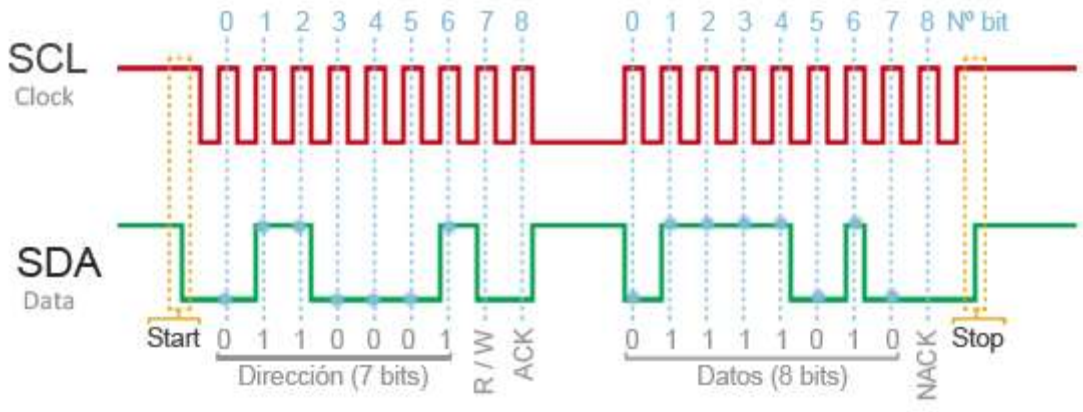


Imagen 7. Pulsos de señal de reloj y envío de datos

Cada dispositivo conectado al bus tiene asignado una dirección de memoria. Esta es la forma por la que se accede a cada uno de ellos.

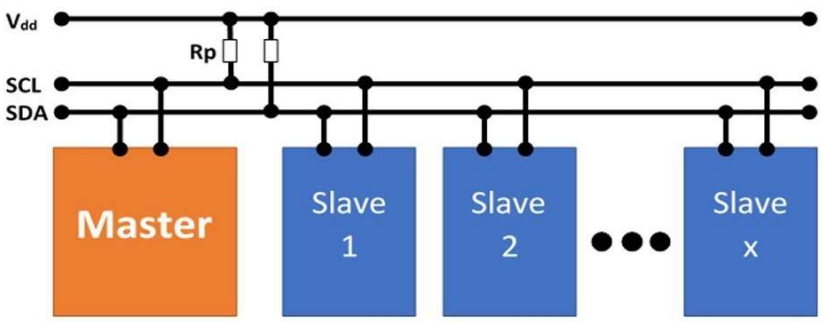


Imagen 8. Protocolo I2C

El maestro proporciona una señal de reloj, que mantiene sincronizados a todos los dispositivos del bus. De esta forma, no es necesario que cada dispositivo, esclavo, tenga esta señal, así se evita la necesidad de sincronizar la transmisión.

Características con las que cuenta la comunicación:

- 7 bits a la dirección del dispositivo esclavo con el que queremos comunicar.
- El bit de sobra indica si la información es enviada o recibida.
- Uno de los bits envía confirmación.
- Uno o más bytes son los datos, ya sean enviados o recibidos, del esclavo.
- Un bit de validación.

Gracias a contar con estos 7 bits se hace posible la conexión y acceso a 112 dispositivos en el mismo bus [16].

Las resistencias pull-up entre la alimentación y el bus son necesarias. Este tipo de resistencias hacen que cuando no haya nada conectado se mantenga prácticamente el voltaje de la fuente en la entrada del ESP32 [21].

Un ejemplo gráfico del uso se ve en la “Imagen 9”.

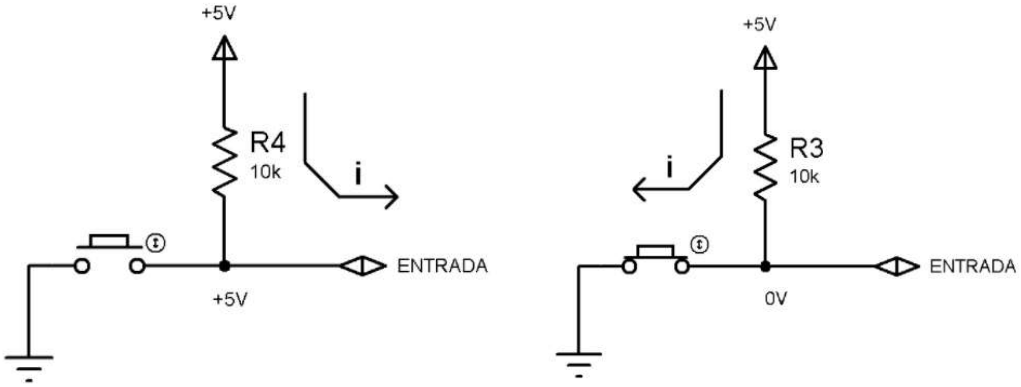


Imagen 9. Acción de la resistencia pull-up

4 Hardware

A continuación, se va a detallar el hardware utilizado. Esto se compone del material físico, dispositivos electrónicos como el módulo wifi ESP32, los sensores utilizados para medir las constantes y Raspberry Pi que ha sido utilizado como miniordenador.

4.1 ESP32

El módulo ESP32 es la evolución de ESP2866, un microcontrolador como los que se encuentran en las placas de Arduino, contando con una gran cantidad de mejoras y adiciones como la conexión wifi/bluetooth y más que se comentan a continuación.

Los módulos facilitan la integración de ESP en otras placas gracias a sus pines de entrada/salida.

Las placas incluyen un regulador de tensión y un controlador serie USB para la alimentación y programación desde el mismo conector. Incluye botones para resetear y conectar.

Una ventaja importante es la posibilidad de programarlo con Arduino IDE debido a que cuenta con la compatibilidad de la tarjeta FireBeetle de DFRobot, un microprocesador de bajo consumo.

Características [1]:

- Procesador de doble núcleo, Tensilica Xtensa LX6, con hasta 240 MHz de frecuencia y un coprocesador ULP de ultra bajo consumo.
- Controlador CAN para la comunicación en sistemas distribuidos.
- 36 pines GPIO, convertidor analógico digital de 12 bits.
- Conexión Wi-Fi con un rango de frecuencia entre 2,4 y 2,5 GHz.
- Conexión bluetooth v4.2.
- La tensión con la que opera está entre 2,7 y 3,6 V.
- Módulo de interfaces que contiene tarjeta SD, UART, SPI, SDIO, I2C, LED PWM, Motor PWM, I2S, IR, contador de pulsos, GPIO, sensor táctil capacitivo, ADC, y DAC.

- 4 MB de memoria flash SPI lo que permite leer y escribir posiciones de memoria en una misma operación. No pierde información, aunque esté desconectada. Son rápidas y económicas.
- Frecuencia cristal de 40 MHz.

Modelo ESP32-WROOM-32 con la descripción de sus pines en la “Imagen 10”.

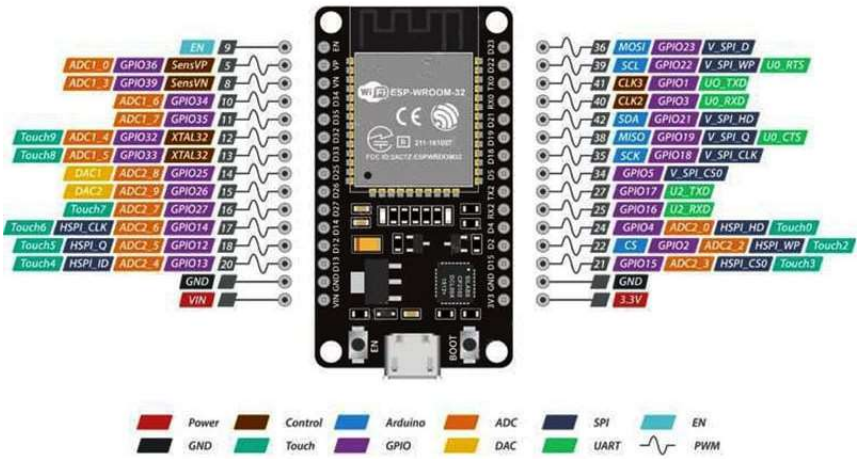


Imagen 10. ESP32- WROOM-32

4.2 Sensores

4.2.1 *Sensor de pulso*

4.2.1.1 Elección previa

Pulse Sensor: La elección de Pulse Sensor se da por su compatibilidad con Arduino y su gran aceptación en este tipo de proyectos en el que se emplean sensores para aplicaciones inalámbricas, además de tener un precio económico. Es el sensor prestado por la universidad al comienzo del proyecto.
El sensor opera en un rango de tensión de entrada entre 3.3 y 5 V.

Está compuesto por un sensor óptico de latidos amplificado con un circuito para crear un filtro y así conseguir cancelación de ruido lo que permite recoger datos

de manera rápida y fiable. Además, tiene un consumo de corriente es bajo, lo que lo hace un buen candidato para aplicaciones móviles [33].

Incluye accesorios que mejoran la calidad de lectura como se ve en la “Imagen 11”:

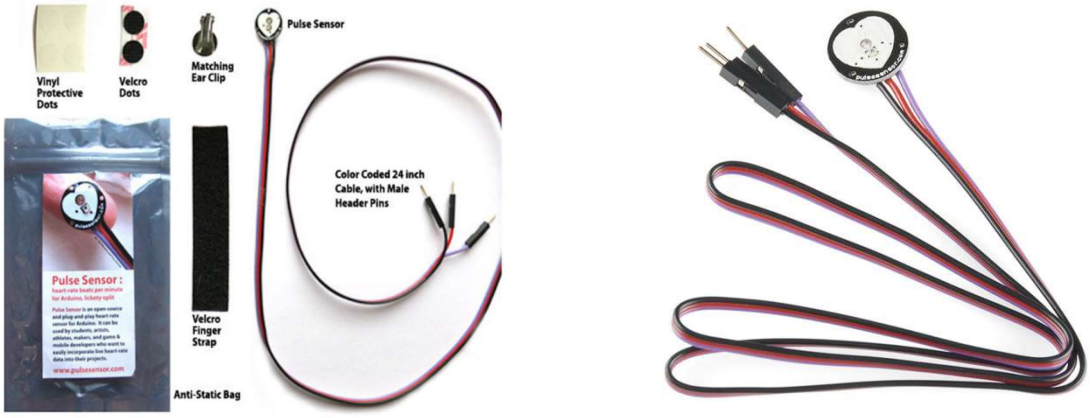


Imagen 11. Kit Pulse Sensor

Cuenta con tres cables que son de alimentación en el caso del color rojo, negro para GND y la señal será comunicada a través del cable de color lila.

4.2.1.2 Elección final

MAX30102 es un sensor de frecuencia cardiaca y saturación de oxígeno. Tiene dos leds, uno de ellos es un led rojo con una longitud de onda de 660 nm y un led infrarrojo con 920 nm. Un fotodetector, óptica especializada, filtro de luz ambiental entre 50 y 60Hz, y un conversor ADC delta sigma de 18 bits y de hasta 1000 muestras por segundo.

La comunicación se lleva a cabo por medio del protocolo I2C.

También, con el fín de compensar los efectos de la temperatura en la medida tiene un sensor interno de temperatura, ampliación de señal, cancelación de ruido y luz ambiental y control de flujos FIFO. Se muestra el circuito interno en “Imagen 12”.

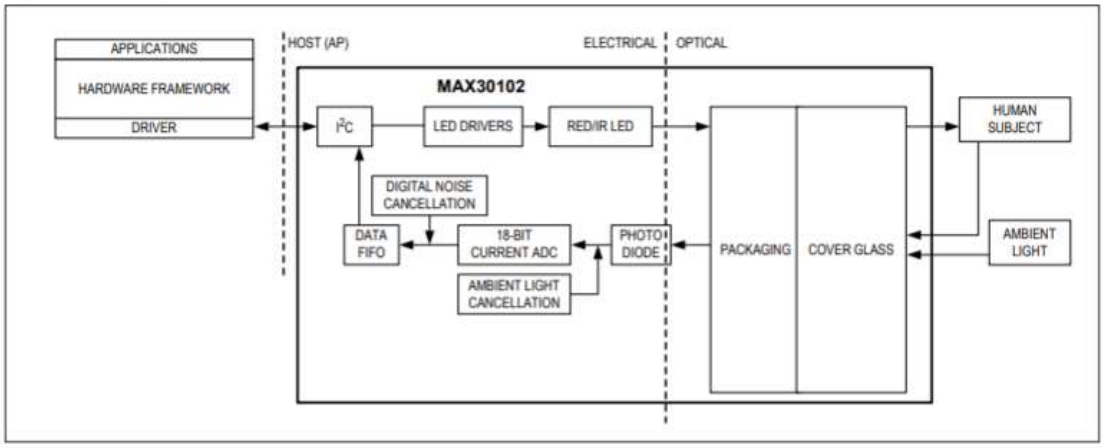


Imagen 12. Diagrama MAX30102

El módulo se alimenta con dos voltajes diferentes, 1,8 V para el circuito y un rango entre 3,3 y 5 V para los leds. Incluye reguladores de voltaje en la placa así que solo necesita 5 V como alimentación. Esto hace que sea buen candidato para aplicaciones portátiles, así como equipos de monitoreo médico, smartwatches, etc. [22].

El motivo de elección ha sido:

- La posibilidad de medir SpO2 además de frecuencia cardiaca como Pulse Sensor.
- Precisión de 97% en el caso de la frecuencia cardiaca y 98% para la saturación de oxígeno en la sangre.
- Bajo coste, alrededor de 8 euros.
- La compatibilidad con Arduino ya que cuenta con una librería llamada SparkFun MAX301x Particle Sensor Library con gran variedad de códigos de ayuda para la recogida de datos.
- Su reducido tamaño que permite medir en cualquier parte del cuerpo y las ranuras que tiene a los lados dan la posibilidad de adjuntar una banda para más comodidad.

Imagen 13. MAX30102

4.2.2 Sensor de temperatura por Infrarrojos - MLX90614ESF

Este es un sensor de temperatura sin contacto. Lleva integrado tanto el sensor termopila como un acondicionamiento de señal, un amplificador de bajo ruido y un convertidor A/D de 17 bits.

Tiene una resolución de 0,02 ºC y precisión de 0, 5 ºC. Si el usuario lo desea puede configurar la salida para que sea PMW, así podremos medir temperatura continuamente en un rango de -20 y 120 ºC con una resolución de 0,014 ºC [32].

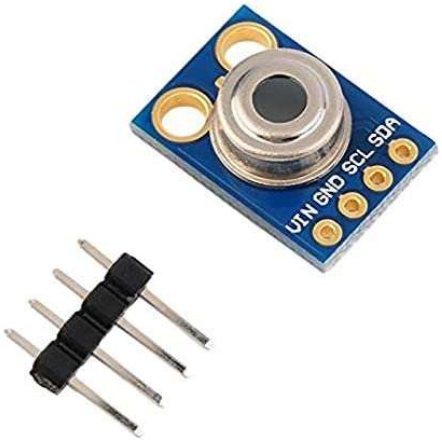


Imagen 14. Imagen del sensor de temperatura por infrarrojos.

4.3 Raspberry Pi

Una Raspberry Pi es un ordenador de placa reducida, su sistema operativo está basado en Linux. El sistema oficialmente compatible se llama Raspberry Pi OS, aunque tiene compatibilidad con muchos otros.

Raspberry Pi OS es una distribución Linux basada en debian. Está optimizado para trabajar con procesadores ARM (Advanced RISC (Reduced Instruction Set Computer) Machine), utilizan un método de procesamiento simplificado y un consumo de energía reducido. En este caso se ha utilizado el de 32 bits.
La instalación de este se puede realizar por medio de Raspberry Pi Imager donde seleccionamos la imagen que queramos descargar y la micro SD donde vamos a cargarla.

En cuanto a la conectividad, Raspberry integra WiFi, Bluetooth, conectividad Ethernet instalada de forma predeterminada, ofrecen velocidad y potencia de procesamiento muy buena con la que se puede manejar datos de audio y video [8].

Es un dispositivo de bajo consumo (limitado a 5 V) alimentados mediante microUSB o encabezados de E/S [8].

Los últimos modelos de Rapsberry tienen hasta 8GB de SDRAM (memoria de acceso aleatorio síncrona y dinámica). Suficiente para ejecutar múltiples funcionalidades para un sistema IoT más complejo [8].

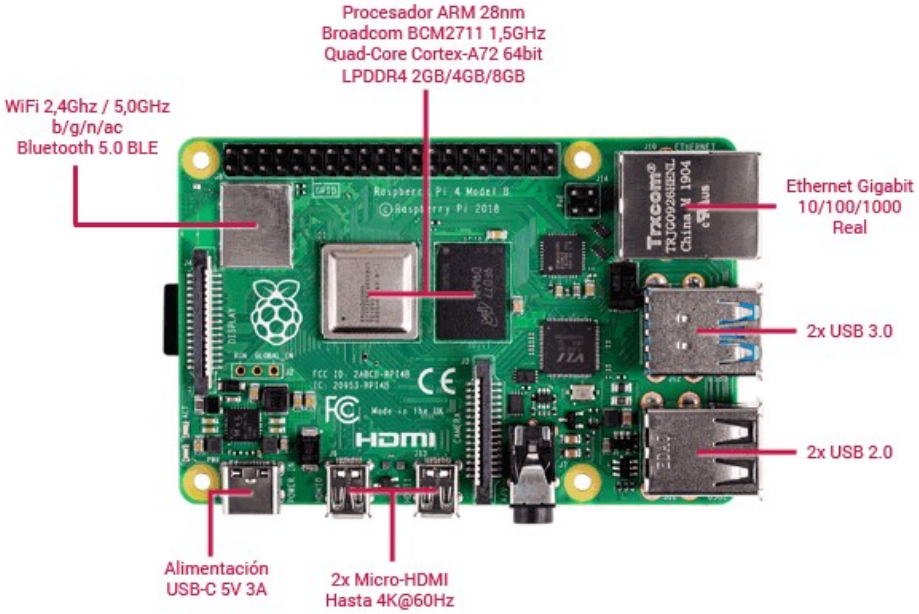


Imagen 15. Raspberry Pi 4 - Model B- 4GB

La aplicación de este ordenador en el proyecto es como bróker MQTT para la comunicación entre suscriptor y publicador. El procedimiento es: un cliente publica un mensaje con un topic, a este topic podrán suscribirse otros clientes y así recibir la información contenida en el mensaje.

Como se pretende que la Raspberry esté encendida bastante tiempo, es aconsejable como complemento Raspberry Pi 4 Case Fan. Es una funda con disipador de calor pegada al procesador. Se ha visto reducida 10 ºC la temperatura, lo que es muy útil para una aplicación de uso diario.

Imagen 16. Raspberry Pi 4 Case Fan

Existen algunas alternativas a Raspberry. Estos SBC han sido creados después de la aparición de nuestro mini ordenador en el mercado.

Existen algunos con características mejoradas y otras iguales, o peores. La elección de compra será dependiendo del uso que le vamos a dar.

Hemos recopilado información de algunos modelos mejorados.

	Raspberry Pi 4B	**Rock Pi 4**	**Orange Pi 4B**	**Odroid N2**
Procesador	Procesador de cuatro núcleos a 1,5 GHz con brazo Cortex-A72	RK3399 de Rockchip con 6 núcleos, 2 a 2GHz y 4 a 1.5GHz	RK3399 de Rockchip con 6 núcleos, 2 a 2GHz y 4 a 1.5GHz	Amilogic S922X con 4 núcleos a 1.8GHz y 2 núcleos a 1.9GHz
Tarjeta gráfica	Broadcom VideoCore VI (Pi 4)	Mali-T860	PCIe	Mali G52
Conexión inalámbrica	Wifi y bluetooth	Wifi y bluetooth en los modelos B, C y pro.	Wifi y bluetooth	No
Memoria flash	Puerto microSD	Puerto eMMC	eMMC 16GB	Puerto eMMC
RAM	1/2/4GB	1/2/4 GB	4 GB	2/4 GB

Tabla 1. Alternativas Raspberry Pi

En la “Tabla 1” se comparan tres modelos con el nuestro. El procesador con el que cuenta Rock Pi 4 y Orange Pi 4B tiene un rendimiento 15% superior al de Raspberry Pi 4. Además, la tarjeta gráfica Mali-T860, nos da el doble de rendimiento.

También Odroid N2 tiene un procesador con 20% más de rendimiento que Raspberry Pi 4 y su tarjeta gráfica 10% más que la de Rock Pi [27].

5 Software

El software utilizado para el desarrollo de proyecto es Arduino IDE donde se ha programado el módulo ESP32. En este escribiremos el código que nos permitirá recoger información de los sensores por las entradas y escribir información en las salidas. Esto se hará posible gracias al servidor MQTT instalado en Raspberry Pi.

En este servidor se desarrollará una comunicación entre el módulo ESP32 y nuestra aplicación controlada por Node-RED.

Node-RED nos permite la programación mediante nodos para la publicación de datos recogidos por los sensores y el control de actuadores mediante una interfaz gráfica por su conjunto de nodos Dashboard, pudiendo controlar estos desde nuestro PC o dispositivo móvil y el almacenamiento de datos en MySQL.

5.1 Arduino IDE

“Para programar Arduino es necesario descargar una IDE (Integrated Development Environment). El IDE es un conjunto de herramientas de software que permiten a los programadores desarrollar y grabar todo el código necesario para hacer que nuestro Arduino funcione como queramos. El IDE de Arduino nos permite escribir, depurar, editar y grabar nuestro programa de una manera sumamente sencilla, en gran parte a esto se debe el éxito de Arduino, a su accesibilidad.” [14].

Gracias a su gran número de librerías y ejemplos muy amplio que facilita mucho el trabajo para programar, en este caso sensores específicos, debido a que tiene funciones preparadas para tratar los datos como especifica la hoja del fabricante.

5.1.1 *Programación en Arduino IDE*

- Programación del sensor MAX30102 y mlx90614 con sus respectivas librerías

- Programación MQTT: Es necesario incluir las librerías WiFi.h y PubSubClient.

5.2 Node-RED

"Node-RED es una herramienta de programación visual." [15].

"Node-RED es un editor de flujo basado en el navegador donde se puede añadir o eliminar nodos y conectarlos entre sí con el fin de hacer que se comuniquen entre ellos." [15].

Permite contar con estructuras de programación sin escribir ni una sola palabra.Gracias a sus nodos y uniones el usuario puede ver la programación de una forma muy sencilla e intuitiva. Además de su potencia, la rapidez en la comunicación y amplia variedad de servicios, puede fomentar el aprendizaje de gente no especializada.

Para el desarrollo, los nodos empleados han sido los siguientes:

- ***MQTT (Message Queue Telemetry Transport):*** Este nodo va a permitir que las entradas de nuestro ESP32 publiquen la información en Node-RED y este a su vez controle las salidas del módulo.

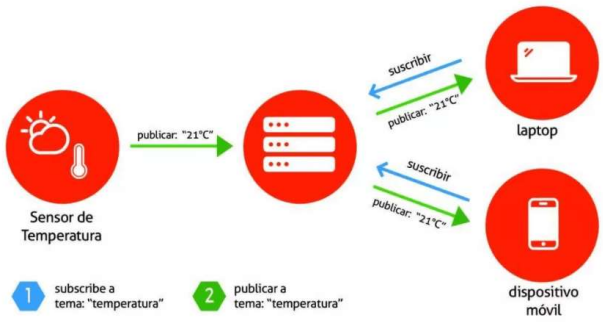


Imagen 17. Paradigma publish/suscribe

- ***Dashboard:*** Este nodo permite crear un gráfico que contenga indicadores, graficas, interruptores, reguladores, texto, datos numéricos, etc. En el caso de

los nodos de entrada, es posible interactuar y una vez establecida la comunicación con el nodo anterior controlar los actuadores conectados a las salidas.
El tablero puede tener varias ventanas, así que en la configuración de estos nodos es necesario especificar a que ventana pertenece cada uno de ellos.

- ***MySQL:*** Es un sistema de gestión de base de datos relacional basado en SQL (Structured Query Language). Es usado a nivel mundial por ser compatible con gran variedad de lenguajes. La aplicación se utiliza para una amplia gama de propósitos, incluidas aplicaciones de almacenamiento de datos, comercio electrónico y registro [34].

  A partir del momento en que Oracle compró MySQL se creó MariaDB, una bifurcación del anterior, pero consiguiendo que siguiera siendo de código abierto y con algunas mejoras en cuanto al rendimiento.

  Este cambio no afecta a nada porque es compatible en su totalidad con lo que antes usaba MySQL. Los códigos en la instalación son igual y la configuración de nodos en Node-RED también como veremos a continuación.

  Los principales procesos que tienen lugar en un entorno MySQL son:

  - "MySQL crea una base de datos para almacenar y manipular datos, definiendo la relación de cada tabla." [4].
  - "Los clientes pueden realizar solicitudes escribiendo declaraciones SQL específicas en MySQL." [4].
  - "La aplicación del servidor responderá con la información solicitada y aparecerá en el lado de los clientes." [4].

5.3 PhpMyAdmin

PhpMyAdmin es una herramienta que permite visualizar y gestionar una base de datos MySQL. Está escrito en el lenguaje PHP.

Permite crear, editar y borrar bases de datos. Su uso es muy sencillo y esto permite la práctica a cualquier usuario.

Entre sus características tenemos:

- Es muy ligero, se puede instalar en una versión en inglés ocupando 6 MB y con la opción de 55 idiomas, 10 MB.
- Es una herramienta de código abierto.
- Existen datos que confirman que es la herramienta más utilizada para gestión de bases de datos.

5.4 Fritzing

Este programa permite hacer diseño de conexiones en protoboard, diseño de PCB, esquemáticos y permite escribir código para probar el funcionamiento del esquemático con la posibilidad de cargarlo en Arduino.

Este software es de código abierto y cuenta con una gran cantidad de librerías para importar los componentes utilizados.

En este caso, ha sido utilizado para el diseño de las conexiones entre los sensores, leds y el módulo ESP32.

5.5 SOLIDWORKS

Es un software para diseño asistido por computadora en 3D. Permite crear piezas en 3D y planos en 2D.

El uso de este programa ha sido posible gracias a la licencia de estudiante por haber cursado la asignatura DAO (diseño asistido por ordenador) durante este curso.

Las funciones que tiene son:

- Modelado de piezas y ensamblajes como modelos de sólido en 3D.
- Dibujos en 2D con la creación automática de vistas de dibujo y acotaciones manuales y automáticas.

- Reutilización y automatización del diseño con la función de búsqueda y DriveWorksXpress para el diseño automático.
- Animación y visualización, movimiento de ensamblajes, videos de CAD.
- Comprobación de interferencias.
- Importar archivos CAD y realizará la conversión y reconocimiento.
- Herramientas básicas de análisis. Análisis del impacto medioambiental de las piezas entre otros.
- Herramientas de productividad potentes.

Dentro de cada punto, aunque solo hemos añadido algunos ejemplos, hay una cantidad mucho más grande de funcionalidades [29].

6 Planos

6.1 Diseño de conexiones

Se muestra el esquema de conexiones con los dos sensores y dos leds, tal y como en el montaje que se ha realizado para este proyecto.

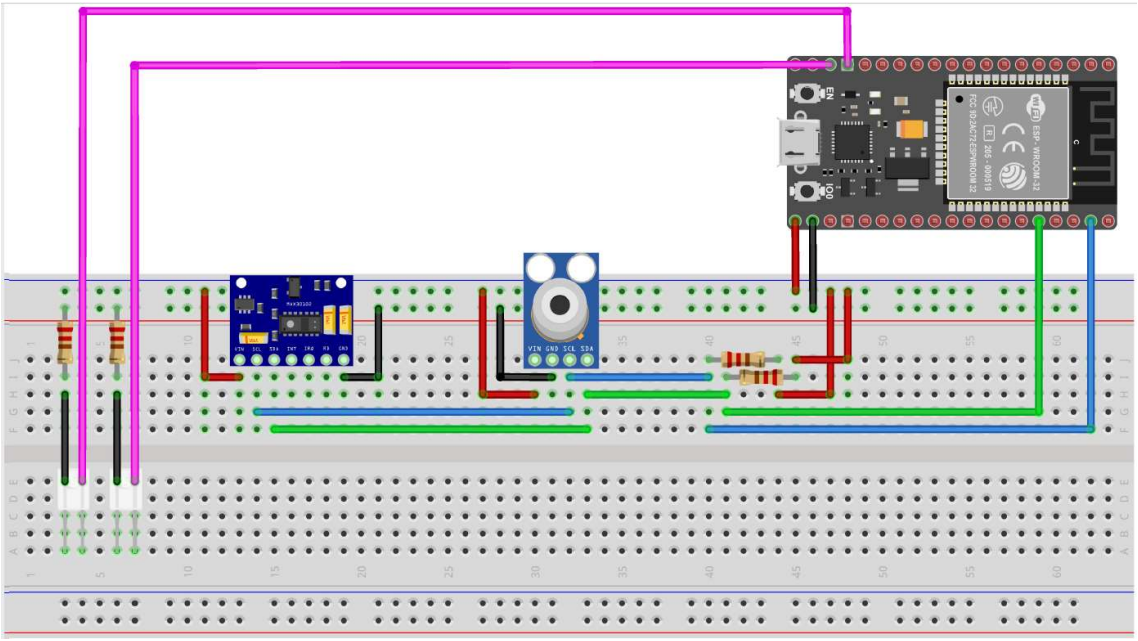


Imagen 18. Esquema de conexiones del proyecto

Los dos sensores están conectados a los pines SDA y SCL por los cables de color verde y azul respectivamente. El rojo indica la alimentación de 3,3 V y el negro, el pin GND.

En cuanto a los leds, tienen el cable rosa conectado a los pines GPIO12 y GPIO13 y el negro los conecta a masa por medio de una resistencia.

6.2 Diseño de la banda

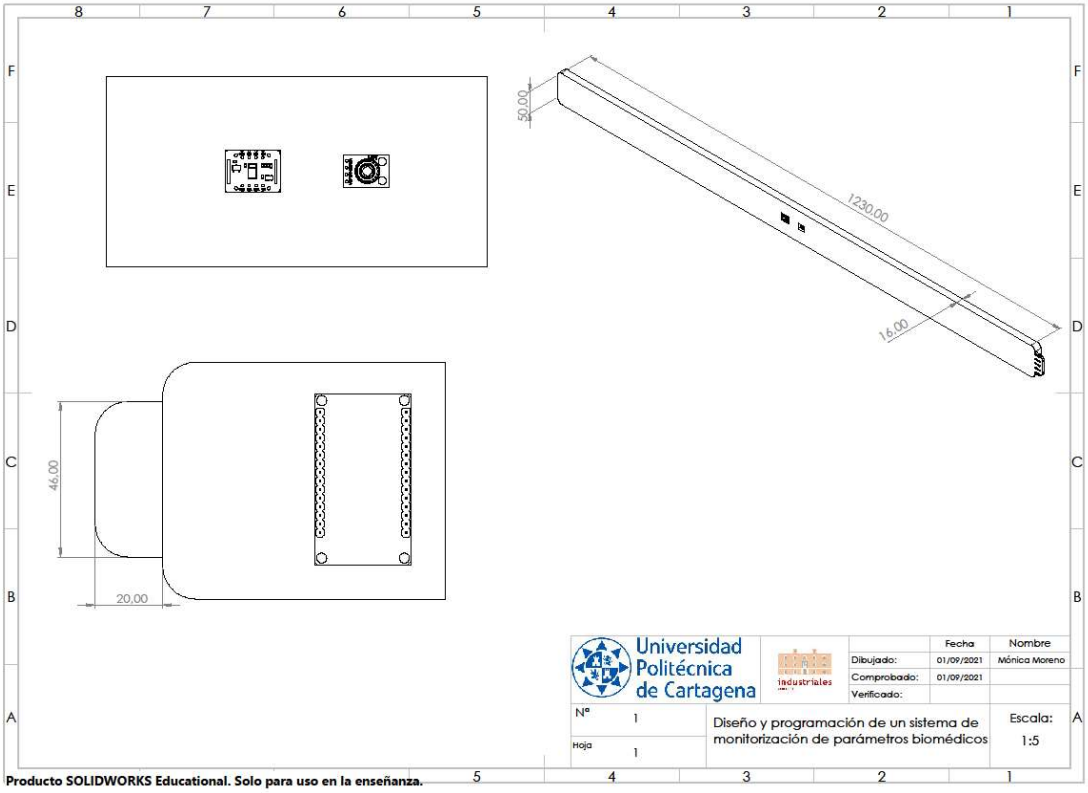


Imagen 19. Diseño banda

Se ha realizado un prototipo 3D con SolidWorks de una banda de tela totalmente regulable gracias a su elasticidad y a un velcro que incorpora en uno de sus extremos, que llevaría incorporada los sensores de temperatura y frecuencia cardiaca en la parte central para un ajuste sencillo, y en la parte posterior de uno de sus extremos encontraríamos el módulo ESP32 de Arduino.

En el plano de la “Imagen 19” se han representado todos los módulos de manera descubierta para una mejor representación del prototipo, pero en el modelo final se encontrarían totalmente cubiertos a excepción de dos ranuras de pequeño tamaño en cada uno de los sensores para su lectura gracias a sus leds infrarrojos. Todo el cableado se realizaría gracias a pequeñas ranuras distribuidas por la banda para poder conectar los sensores con el ESP32, este cableado en el modelo final también se encontraría totalmente cubierto para la comodidad del usuario.

7 Desarrollo del proyecto

En este apartado se va a explicar cómo se ha instalado y programado el software explicado teóricamente en apartados anteriores. Está compuesto por Arduino IDE, donde hemos programado los sensores y la comunicación MQTT y, Node-RED, donde gracias a MQTT se ha podido configurar los datos de los sensores para crear una interfaz gráfica y una base de datos.

También se explica el software instalado en raspberry ya que hace de servidor para la base de datos y Node-RED y de bróker para MQTT.

7.1 Configurar Arduino IDE

En primer lugar, descargamos el software para Windows 10 en [6].

Arduino IDE es compatible con ESP32, pero hay que añadir una URL para hacerlo posible. En la pestaña de Archivo > Preferencias > Ajustes > *"Gestor de URLs Adicionales de Tarjetas, añadimos la dirección". "*https://dl.espressif.com/dl/package_esp32_index.json". Así estará disponible la familia ESP32 para seleccionar cual es la utilizada. En este caso será ESP32 Dev Module.

7.2 Programar Arduino

El código se divide en tres partes importantes. El tratamiento de datos con el sensor de pulso, con el sensor de temperatura y, la MQTT para la comunicación de ESP32 con Node-RED.

Las referencias de los códigos son:

- El sensor de temperatura con [7].
- El sensor de pulso sale a partir del ejemplo 5 de la librería *"SparkFun MAX301x Particle Sensor Library"*.
- MQTT gracias al canal de youtube de techiesms.

Analizándolos por orden de extensión, empezamos por el sensor de temperatura.

El código se encuentra en el documento de Anexos.

7.2.1 *Código para MLX90614*

Este código es muy sencillo gracias a las librerías del sensor disponibles para instalar en Arduino.

Lo primero es la declaración de las librerías. Es necesario tener *"<Wire.h>"*, por el bus I2C y *"<Adafruit_MLX90614.h>"*, por el propio sensor.

Lo único que se hace es inicializarlo en la función *"setup ()"* con *"mlx.begin ()"* y después se escribe en bucle la información que nos importa, la temperatura en grado Celsius, gracias a la función ya programada *"readObjectTempC ()".*

7.2.2 *Código para MAX30102*

Para el sensor de pulso incluimos *"MAX30105.h", "spo2_algorithm.h"* y comparte con el sensor anterior *"<Wire.h>".*

Las variables que se declaran antes de empezar el programa son: el tamaño del vector donde se guardan los valores de infrarrojos y del led que incorpora de color rojo como se comentó en la teoría, el vector donde se almacenan estos datos, un contador para las posiciones del vector y, otro para calcular la media de la frecuencia cardiaca.

También se define lo que queremos presentar en la pantalla que es, las pulsaciones por minuto, el pulso medio y la saturación de oxígeno.

Entrando en las funciones tenemos que, en la función *"setup ()"* tenemos la condición de probar infinitamente hasta que empiece a leer. En la función *"begin ()"* especificamos la dirección de memoria de este sensor, al compartir bus con el otro, ahorramos confusiones. Después se definen modos del sensor según indica la hoja del fabricante sobre el rango del convertidor analógico/digital, el

ancho de pulso, el modo del led, etc. Y posteriormente, con una función de la forma *"particleSensor.setup ()"* se introducen los valores configurados.

En la función bucle *"loop ()"* se calculan los valores de pulso y spo2 de 100 muestras. Después se desechan las 25 primeras medidas de infrarrojos y led rojo y se guardan las restantes como medidas a mostrar, y así con cada muestra. La función *"particleSensor.nextSample()"* permite pasar a analizar la siguiente muestra.

Para calcular el pulso medio se suman todos los datos de *"heartRate"* y se divide por el número de muestras que se están sumando con las variables definidas al principio del programa.

Para terminar, se imprimen los resultados y, con la función *"maxim_heart_rate_and_oxygen_saturation ()"* se recalculan los valores de pulso y saturación de oxígeno.

7.2.3 *Código para MQTT*

Para programar MQTT es necesario conectar ESP32 a una red wifi. Las librerías necesarias para esto son: *"<WiFi.h>"* y *"<PubSubClient>".*

Se han definido variables de entrada, string para guardar los datos de los sensores, y salida, leds. Así es posible comprobar que funciona tanto la publicación como la suscripción de mensajes.

Además, para conectar la red wifi es necesario la ssid del router y la contraseña. También para conectarnos por MQTT tenemos que poner la IP de nuestro bróker, en este caso la Raspberry Pi y el puerto que ocupa, el 1883.

En Arduino se crea un usuario y contraseña mqtt que será el mismo que pondremos en la configuración MQTT de Node-RED.

En este código tenemos varias funciones. La primera que se encuentra es para conectar el wifi. Entra en bucle hasta que se haya conectado, igual que hacíamos con el sensor de pulso.

La función *"callback()"* trata los mensajes, establece la condición de encender o apagar un led si el mensaje recibido es 1 o 0.

La función *"reconnect()"* soluciona un problema de desconexión con el bróker, si esto ocurriera, genera identidades aleatorias para ese usuario y contraseña, igual que la configuración de Node-RED hasta que consiga conectarse. Si se consigue conectar, nos suscribe a los *"topic"* y si no vuelve a intentarlo en 5 segundos.

En *"setup()"* configuramos que los pines 1 y 2 van a ser salidas y se establece el servidor con nuestra ip y el puerto definido anteriormente y la función *"callback"* dentro de *"setCallback()"* de la librería *"PubSubClient".*

Por último, la publicación de mensajes. Se utiliza la función "dtostrf ()" para convertir números con decimales a *"string"* y así publicarlo como texto. Esta función admite cuatro parámetros: el valor que queremos convertir, tamaño mínimo que se ocupa en el *"string"* de salida, números después del punto decimal y variable donde se guarda.

7.2.4 *Código simulado*

Al querer mostrar más constantes que cantidad de sensores, se han simulado algunas señales.

Para mostrar la presión arterial se ha creado un string donde se introduce un valor con el fin de mostrar este dato en la pantalla, en este caso hemos separado entre presión diastólica y sistólica y, los valores introducidos son constantes ya que no es una variable que esté cambiando cada segundo.

7.3 Manual de configuración e instalación de software en Raspberry Pi

En primer lugar, descargamos una imagen. En este caso la imagen seleccionada ha sido *"Raspberry Pi OS"* por recomendación de la página oficial.

Una vez descargada, la cargamos en una tarjeta micro SD con *"Raspberry Pi Imager".*

Imagen 20. Raspberry Pi Imager

Después debemos descargar software para encontrar la IP de nuestra Raspberry y poder acceder a su terminal. Esto lo hemos hecho con *"Advanced IP Scanner"* y *"PuTTy"* respectivamente.

"Putty" pedirá un usuario, pi y una contraseña, raspberry. Después de esto se pueden escribir los comandos necesarios. Los utilizados han sido: *"sudo apt-get update", "sudo apt-get upgrade" y "raspi-config".*

Dentro de *"raspi-config"*, para poder visualizar el escritorio de Raspberry Pi tendremos que habilitar SSH, VNC y cambiar la resolución como se puede ver a continuación en "Imagen 21", "Imagen 22", "Imagen 23", "Imagen 24":

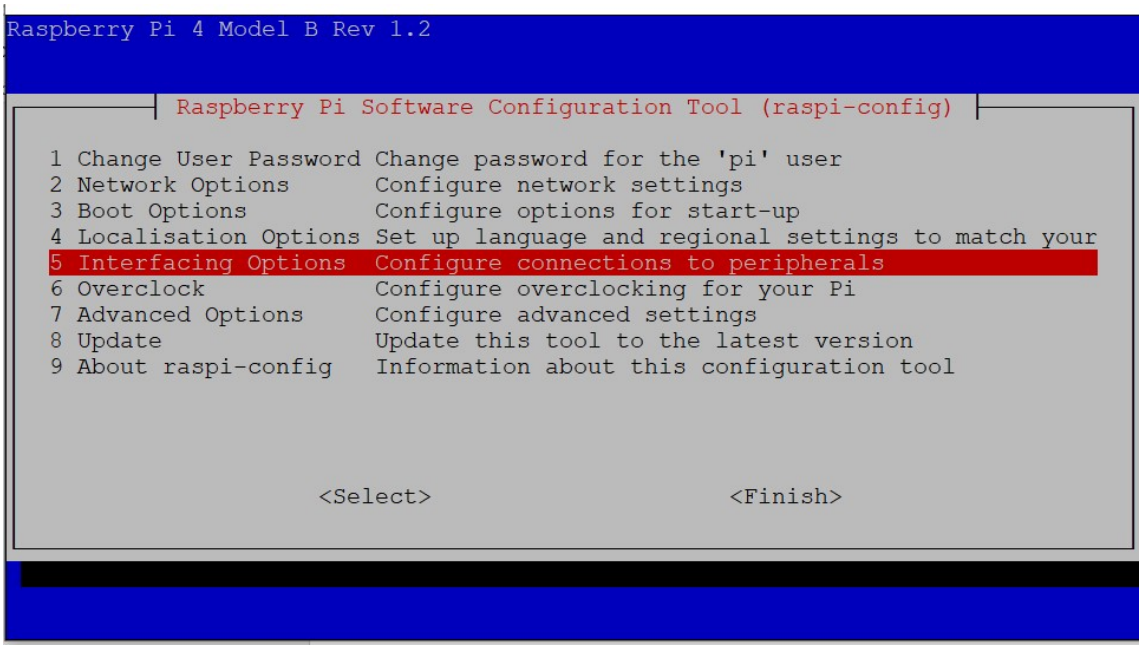


Imagen 21. Pantalla principal "sudo raspi-config"

Habilitamos la opción de SSH y VNC.

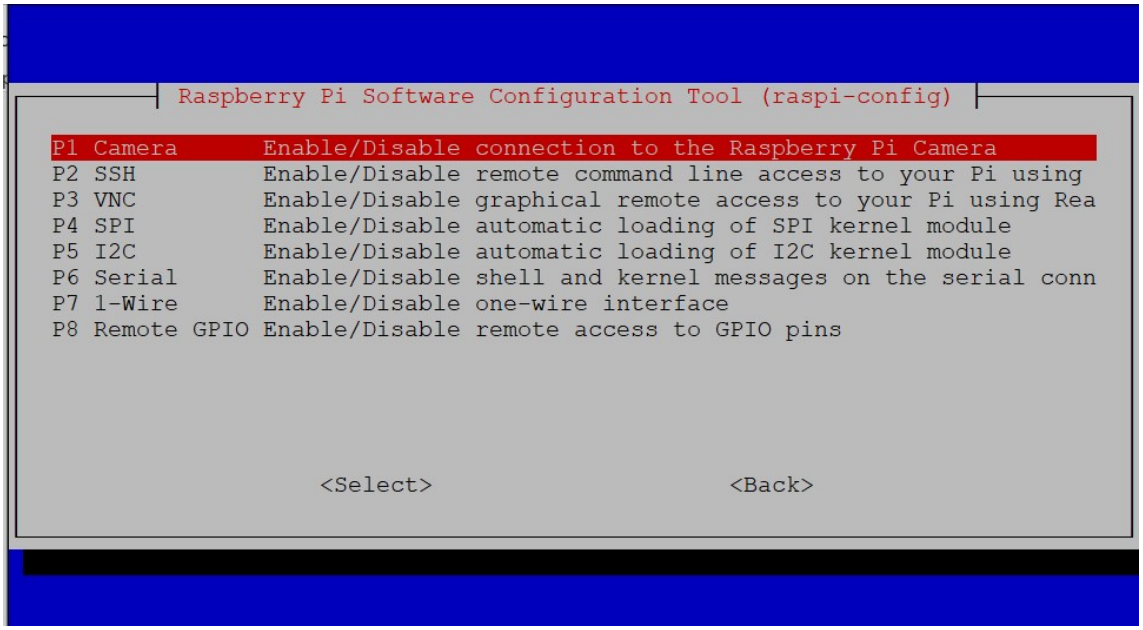


Imagen 22.Opción 5 “Interfacing options”

Y posteriormente, cambiamos la resolución como se muestra:

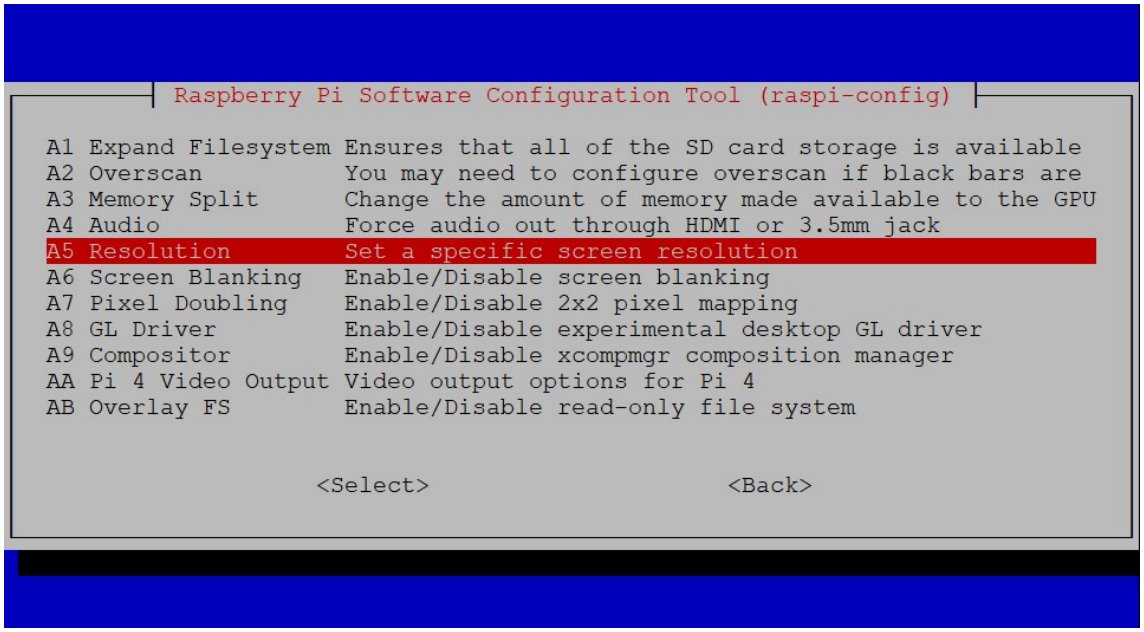


Imagen 23. Opción 7 "Advanced Options"

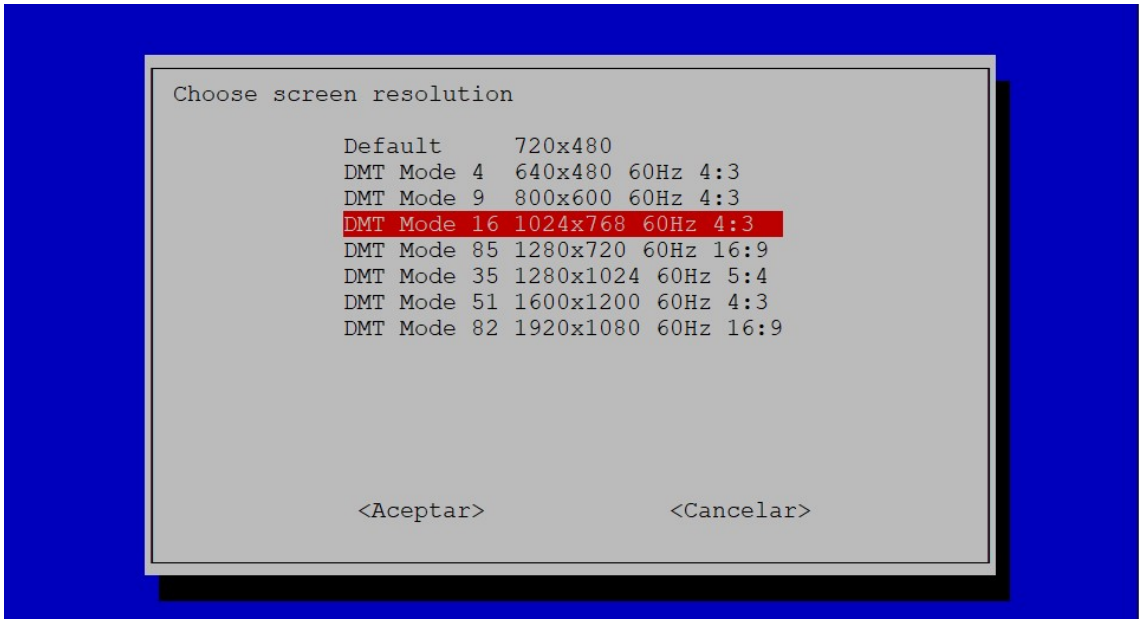


Imagen 24. Resolución elegida

Al terminar se reinicia Raspberry Pi y todo queda establecido.

Una vez configurado, necesitamos un programa para un acceso remoto a la pantalla desde nuestro ordenador, para ello hemos descargado VNC Viewer.

Cuando iniciemos el programa, debemos introducir la dirección IP de Raspberry Pi y el usuario y contraseña, que es el mismo que hemos puesto anteriormente en el terminal.

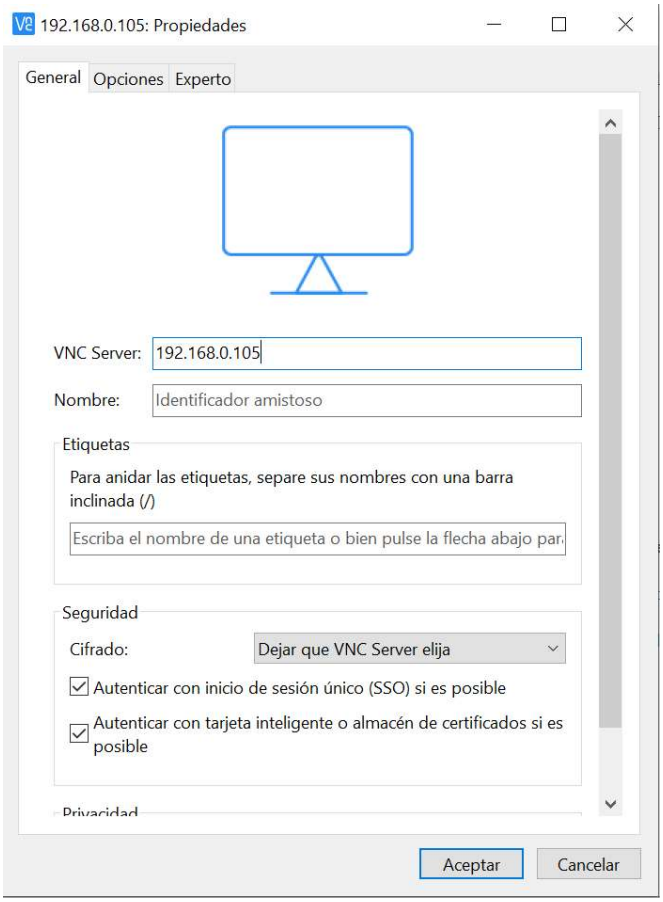


Imagen 25. Conexión al escritorio por IP

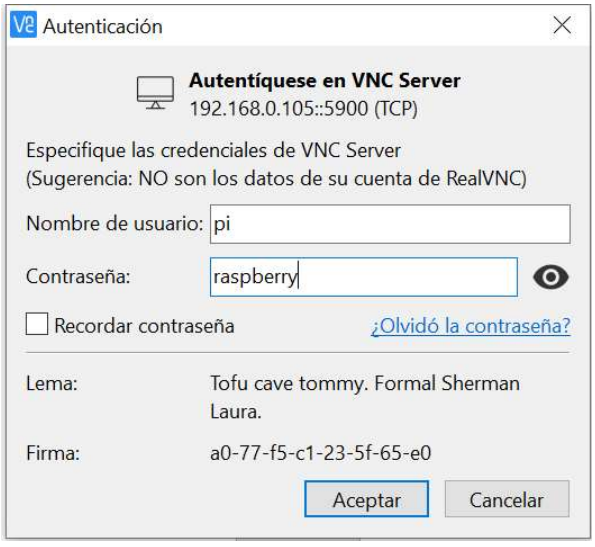


Imagen 26. Inicio de sesión en VNC Viewer

Al iniciar la conexión remota, por seguridad, aparece la opción de cambiar la contraseña que ha sido sustituida por “raspberrypi”.

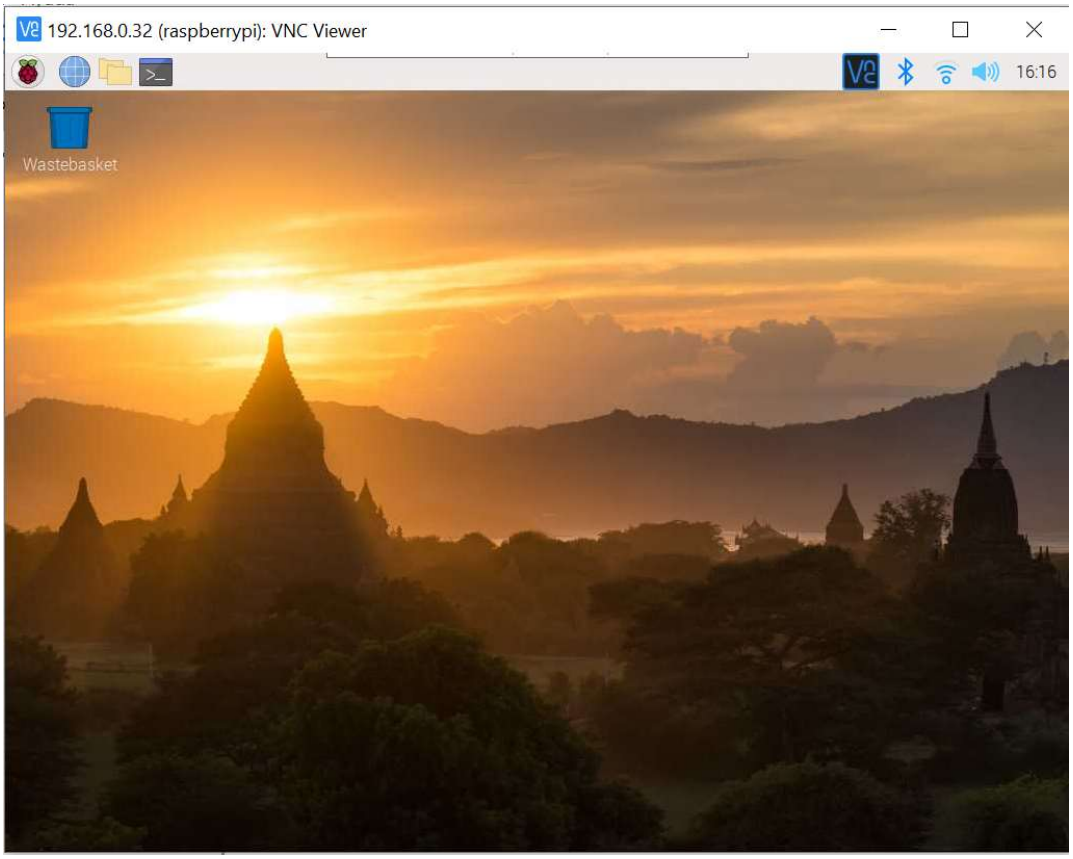


Imagen 27. Escritorio Raspberry Pi

Finalmente, para conectarnos a la red wifi, haciendo click en el icono de las flechas, podemos elegir el nombre de la red e introducir la contraseña.

Imagen 28. Conexión vía WiFi

Una opción interesante es poner la IP estática. Gracias a trabajar en una dirección fija, siempre que encendamos la Raspberry será de la misma manera. Además, este proyecto va enfocado a la instalación de gran cantidad de ellas así que es más interesante aún saber que IP está relacionada con cada una.

Para ello tenemos que editar el fichero *"/etc/dhcpcd.conf"* con el comando *"sudo nano /etc/dhcpcd.conf".* Al abrirlo vemos un texto comentado que debemos modificar según la dirección que queramos establecer, en mi caso ha sido 192.168.0.32 como se ve y, para terminar, debemos reiniciar el dispositivo.

```
# Example static IP configuration:
interface wlan0
fe80::99ca:fb5d:c39d:2aadstatic ip_address=192.168.0.32/24
static ip6_address=fe80::99ca:fb5d:c39d:2aad::ff/64
static routers=192.168.0.1
static domain_name_servers=192.168.0.1
```

Imagen 29. IP estática

7.3.1 *Instalar Node-RED*

El primer paso es escribir el comando de actualizar "sudo apt-get update", se hace para cada instalación, solo se indica en este apartado, pero se supone para los demás.

Después instalamos el programa por comandos en el orden indicado a continuación:

1. *"bash <(curl -sL https://raw.githubusercontent.com/node-red/linux-installers/master/deb/update-nodejs-and-nodered)".* Aparecerán preguntas a las que se responde con y + intro. Una vez hecho esto se cargarán los paquetes necesarios para la instalación como se muestra en la imagen.

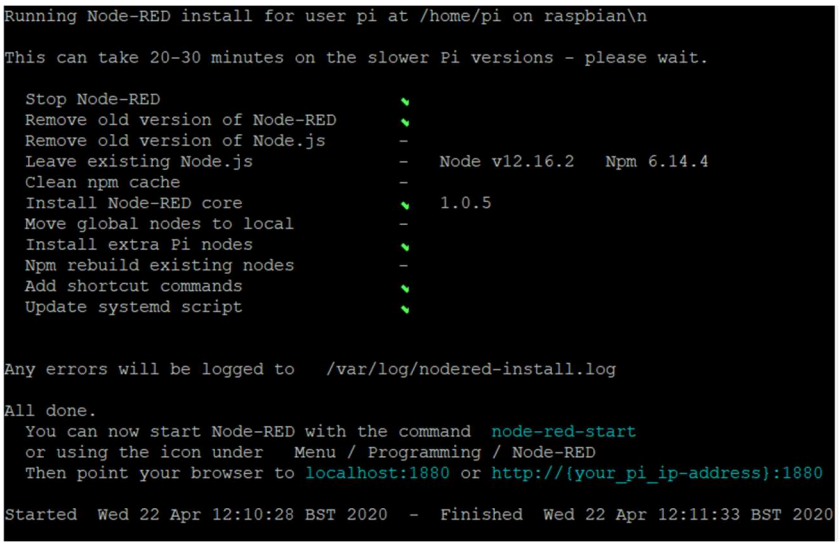


Imagen 30. Instalar node-RED

2. *"node-red-start"* para iniciar el programa, de esta manera con la IP de Raspberry Pi y el puerto 1880 (192.168.0.32:1880) en el buscador se puede acceder a node-red y empezar a programar.
3. Si no queremos que sea necesario escribir el comando anterior cada vez que vayamos a iniciar node-red, se puede ejecutar sudo *"systemctl enable nodered.service"*.

7.3.2 Instalar MQTT

El primer paso, una vez todo actualizado, escribimos los comandos sudo *"apt-get install mosquitto"* y *"sudo apt-get install mosquitto-clients".* Así ya está instalado y tiene el servicio en el puerto 1883 o 8883 con TLS.

Si queremos iniciar, parar o reiniciar debemos ejecutar respectivamente*: "sudo systemctl start mosquitto.service"*, *"sudo systemctl stop mosquitto.service"* y *"sudo systemctl restart mosquitto.service".*

Para probar la comunicación es necesario abrir dos terminales. En uno de ellos será el suscriptor de la manera *"mosquitto_sub -h localhost topic -d"*, y el otro publicador como *"mosquitto_pub -h localhost -t topic -m mensaje".*

Si aparece la palabra "mensaje" en el terminal del suscriptor, la comunicación es correcta.

7.3.3 *Instalar MariaDB*

La instalación de MariaDB utiliza el comando *"sudo apt install mariadb-server".* Para que el servicio sea seguro ejecutamos *"sudo mysql_secure_installation"* en la que pregunta por una contraseña, en este caso se ha escrito *"root".*

Para confirmar la correcta ejecución escribimos *"systemctl status mariadb.service".*

```
pi@raspberrypi:~ $ systemctl status mariadb.service
● mariadb.service - MariaDB 10.3.29 database server
   Loaded: loaded (/lib/systemd/system/mariadb.service; enabled; vendor preset:
   Active: active (running) since Mon 2021-08-30 19:02:46 BST; 17h ago
     Docs: man:mysqld(8)
           https://mariadb.com/kb/en/library/systemd/
  Process: 523 ExecStartPre=/usr/bin/install -m 755 -o mysql -g root -d /var/run
  Process: 537 ExecStartPre=/bin/sh -c systemctl unset-environment _WSREP_START_
  Process: 547 ExecStartPre=/bin/sh -c [ ! -e /usr/bin/galera_recovery ] && VAR=
  Process: 748 ExecStartPost=/bin/sh -c systemctl unset-environment _WSREP_START
  Process: 756 ExecStartPost=/etc/mysql/debian-start (code=exited, status=0/SUCC
 Main PID: 606 (mysqld)
   Status: "Taking your SQL requests now..."
    Tasks: 30 (limit: 4915)
   CGroup: /system.slice/mariadb.service
           └─606 /usr/sbin/mysqld
```

Imagen 31. MariaDB activado

7.3.4 *Instalar phpMyAdmin*

Para tener este software, es necesario instalar algunas cosas antes.

Lo primero necesario es un servidor web como es Apache. El comando para esto es *"sudo apt-get install apache2 -y"*, -y hace que se instale aceptando las preguntas, de forma que se instala por defecto. Una vez ejecutado se puede comprobar si ha ido según lo esperado escribiendo en el buscado "*http://IP".* Debería aparecer la siguiente imagen.

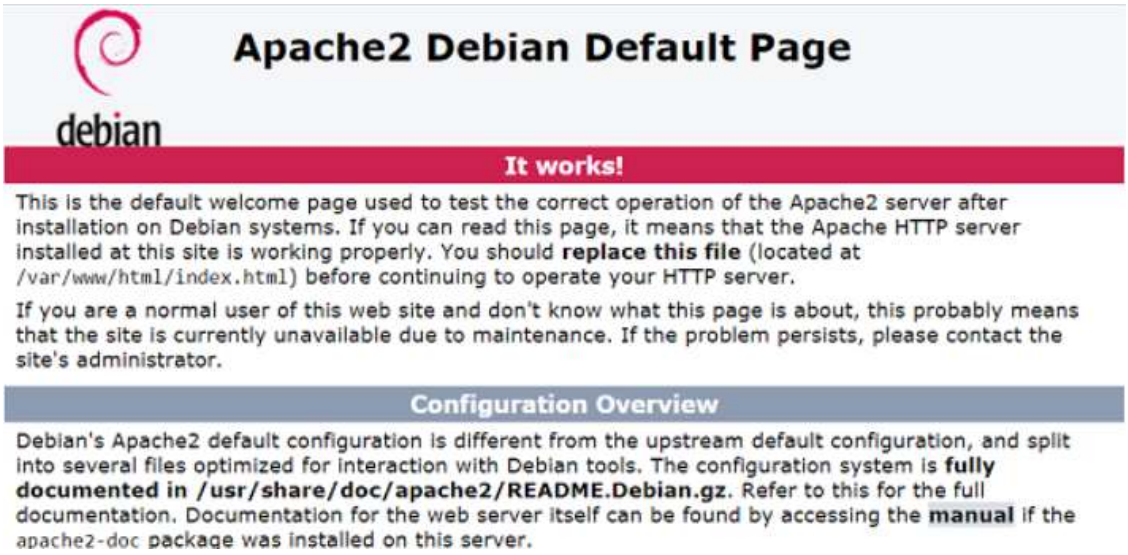


Imagen 32. Prueba de que Apache está correctamente instalado

Para hacer que el propietario sea el usuario pi vamos a ejecutar *"sudo chown pi: /var/www/html/index.html"* y para finalizar, reiniciamos Apache *con "sudo service apache2 restart".*

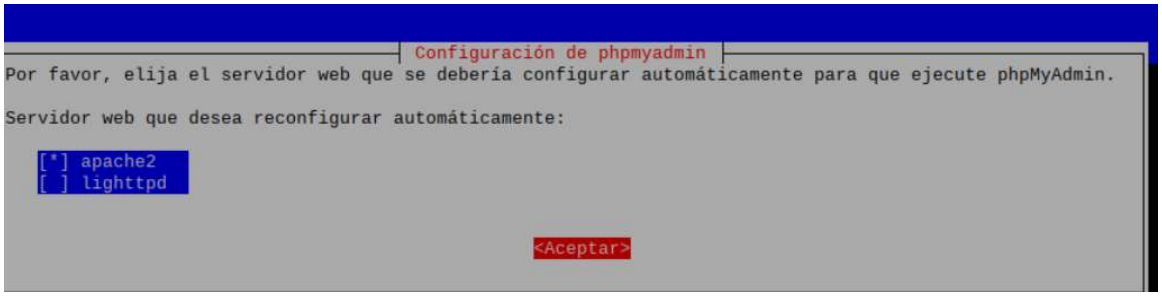
A continuación, vamos a instalar PHP, como se ha comentado es un lenguaje de programación, en el que está escrito phpMyAdmin y hace posible la comunicación entre el servidor instalado y la interfaz que queremos llegar a instalar.

Para ello vamos a ejecutar *"sudo apt-get install php7.1 php7.1-cli php7.1-common libapache2-mod-php7.1 php7.1-mysql php7.1-fpm php7.1-curl php7.1-gd php7.1-bz2 php7.1-mcrypt php7.1-json php7.1-tidy php7.1-mbstring php-redis php-memcached -y".*

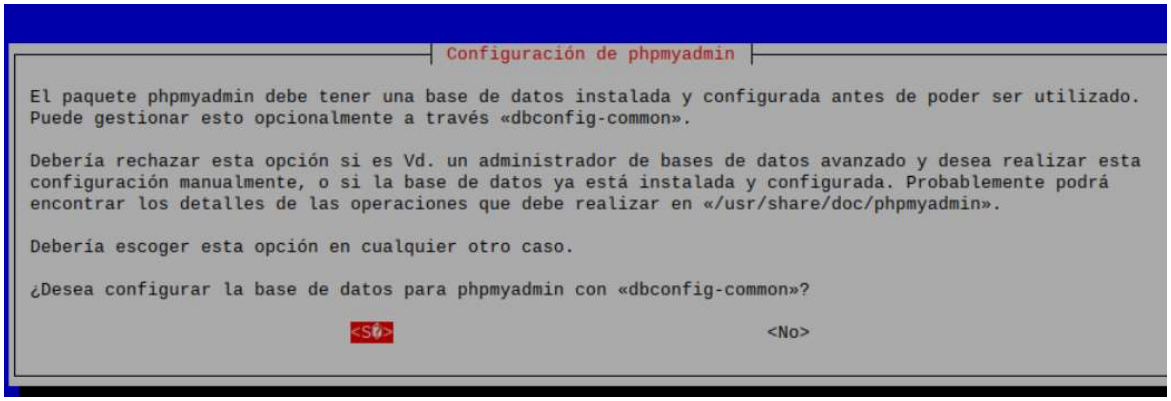
Ya está instalado así que creamos la página *"sudo nano /var/www/html/index.php y, escribimos <?php phpinfo(); ?>"*. Esto significa que si todo va bien, escribiendo en el buscador nuestra *"ip/index.php"* aparecerá la información que lleva la función *"php.info ();".*

Con estos dos complementos, el siguiente paso ya es instalar phpMyAdmin. Las instrucciones para esto son ejecutar *"sudo apt-get install phpmyadmin -y".*

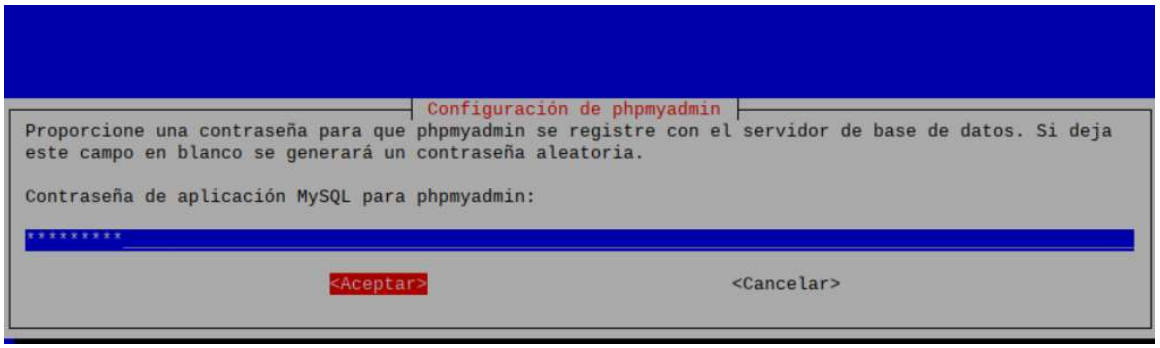
En la instalación aparecen algunas ventanas como:



Se selecciona apache2 con el botón de espacio y con el tabulador nos desplazamos a la opción de aceptar.



Aceptamos.



Escribimos la contraseña en esta ventana y al aceptar aparecerá otra para confirmar la contraseña elegida.

Con estos pasos está la instalación terminada así que escribiendo nuestra *ip/phpmyadmin* tendremos acceso a la aplicación.

7.4 Manual para crear una base de datos

En este apartado se va a explicar cómo construir una base de datos para aplicaciones con Raspberry Pi.

Para seguir este manual es necesario que tener instalado previamente MariaDB y phpMyAdmin como se explica en el apartado anterior.

Los pasos a seguir para crear una base de datos en Raspberry Pi con MariaDB son:

1. Escribimos *"sudo mysql -u root -p – h localhost"*. Nos pide una contraseña, se escribe la configurada anteriormente.
2. Se crea la base de datos haciendo *"create database nombre_de_base_de_datos;"*
3. Para acceder a esta escribimos *"use nombre_de_la_base_de_datos;"*
4. Creamos un usuario asociado a una contraseña: *"create 'nombre_del_usuario'@'localhost' identified by 'contraseña';"*
5. Se otorgan los privilegios: *"grant all privileges on nombre_de_la_base_de_datos.* to 'nombre_del_usuario'@'localhost';"*
6. Para cargar los privilegios sin necesidad de reiniciar: *"flush privileges;"*

```
pi@raspberrypi:~ $ sudo mysql -u root -p -h localhost
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 39
Server version: 10.3.29-MariaDB-0+deb10u1 Raspbian 10

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> create database basedatos;
Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]> use basedatos;
Database changed
MariaDB [basedatos]> create user 'nombredeusuario'@'localhost' identified by 'cont
rasena';
Query OK, 0 rows affected (0.001 sec)

MariaDB [basedatos]>  grant all privileges on basedatos.* to 'nombredeusuario'@'lo
calhost';
Query OK, 0 rows affected (0.001 sec)

MariaDB [basedatos]> flush privileges;
Query OK, 0 rows affected (0.001 sec)
```

Imagen 33. Crear base de datos MariaDB

7. Para crear una tabla donde introducir los datos: *"CREATE TABLE nombre_de_la_tabla (nombre_del_dato1 tipo_de_dato1, nombre_del_dato2 tipo_de_dato2…)."*

```
MariaDB [basedatos]>  CREATE TABLE tabladatos1 (id BIGINT UNSIGNED AUTO_INCREMEN
T PRIMARY KEY, temperatura double, frecuencia_cardiaca int, oxigeno_sat int,pres
ion_diastolica int, presion_sistolica int, fecha date default current_timestamp)
;
Query OK, 0 rows affected (2.465 sec)
```

Imagen 34. Crear tabla en la base de datos

El tipo de dato de la primera fila de la tabla se trata de una cuenta de los datos, cada vez que se añada una nuevo se incrementará. Para la temperatura hemos definido un tipo de dato decimal y para la frecuencia cardiaca y la saturación de oxígeno, números enteros ya que así son las medidas reales. Con las condiciones que hemos impuesto en la fecha, publicará el valor de cada momento que se introducen los datos.

Con el comando *"DESCRIBE nombre_de_la_tabla;"* visualizamos la tabla.

```
MariaDB [basedatos]> DESCRIBE tabladatos;
+--------------------+---------------------+------+-----+---------------------+
----------------+
| Field              | Type                | Null | Key | Default             |
 Extra          |
+--------------------+---------------------+------+-----+---------------------+
----------------+
| id                 | bigint(20) unsigned | NO   | PRI | NULL                |
 auto_increment |
| temperatura        | double              | YES  |     | NULL                |
                |
| frecuencia_cardiaca | int(11)            | YES  |     | NULL                |
                |
| oxigeno_sat        | int(11)             | YES  |     | NULL                |
                |
| fecha              | date                | YES  |     | current_timestamp() |
                |
+--------------------+---------------------+------+-----+---------------------+
----------------+
5 rows in set (0.007 sec)
```

Imagen 35. Tabla de datos

Por último, con el comando *"select * from nombre_de_la_tabla;"* podemos ver los valores que se están almacenando

```
MariaDB [basedatos]> select * from tabladatos;
+----+-------------+---------------------+-------------+------------+
| id | temperatura | frecuencia_cardiaca | oxigeno_sat | fecha      |
+----+-------------+---------------------+-------------+------------+
|  1 |        36.5 |                  80 |          95 | 2021-08-31 |
|  2 |        36.5 |                  78 |          95 | 2021-08-31 |
|  3 |        36.3 |                  79 |          95 | 2021-08-31 |
|  4 |        36.3 |                  78 |          95 | 2021-08-31 |
|  5 |        36.3 |                  78 |          95 | 2021-08-31 |
|  6 |        36.4 |                  76 |          96 | 2021-08-31 |
+----+-------------+---------------------+-------------+------------+
6 rows in set (0.001 sec)
```

Imagen 36. Datos almacenados

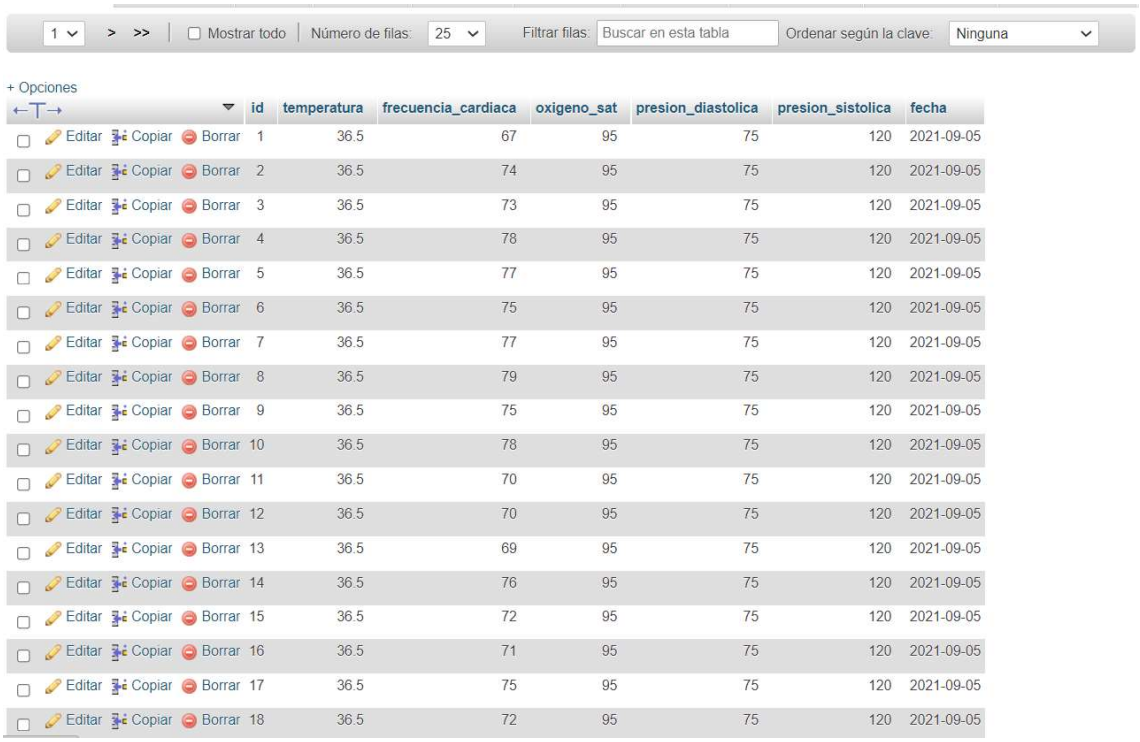
Para una práctica más cómoda es conveniente tener la posibilidad de visualizar la tabla fuera del terminal y no tener que acceder a través de comandos.

Para ello vamos a utilizar phpMyAdmin, una herramienta que nos permite manejar y visualizar las bases de datos de forma sencilla.

Se ha realizado previamente la instalación de esta aplicación en la Raspberry Pi, así que solo escribiendo en el buscado *IP/phpmyadmin* llegamos a esta ventana.

Imagen 37. Página de acceso a phpMyAdmin

Cuando hayamos programado Node-RED para que guarde la información en la base de datos, introduciendo usuario y contraseña y consultando la base de datos donde hemos especificado que se guarden los datos podremos ver esto:



1 > >> | Mostrar todo | Número de filas: 25 | Filtrar filas: Buscar en esta tabla | Ordenar según la clave: Ninguna

+ Opciones

	id	temperatura	frecuencia_cardiaca	oxigeno_sat	presion_diastolica	presion_sistolica	fecha
Editar Copiar Borrar	1	36.5	67	95	75	120	2021-09-05
Editar Copiar Borrar	2	36.5	74	95	75	120	2021-09-05
Editar Copiar Borrar	3	36.5	73	95	75	120	2021-09-05
Editar Copiar Borrar	4	36.5	78	95	75	120	2021-09-05
Editar Copiar Borrar	5	36.5	77	95	75	120	2021-09-05
Editar Copiar Borrar	6	36.5	75	95	75	120	2021-09-05
Editar Copiar Borrar	7	36.5	77	95	75	120	2021-09-05
Editar Copiar Borrar	8	36.5	79	95	75	120	2021-09-05
Editar Copiar Borrar	9	36.5	75	95	75	120	2021-09-05
Editar Copiar Borrar	10	36.5	78	95	75	120	2021-09-05
Editar Copiar Borrar	11	36.5	70	95	75	120	2021-09-05
Editar Copiar Borrar	12	36.5	70	95	75	120	2021-09-05
Editar Copiar Borrar	13	36.5	69	95	75	120	2021-09-05
Editar Copiar Borrar	14	36.5	76	95	75	120	2021-09-05
Editar Copiar Borrar	15	36.5	72	95	75	120	2021-09-05
Editar Copiar Borrar	16	36.5	71	95	75	120	2021-09-05
Editar Copiar Borrar	17	36.5	75	95	75	120	2021-09-05
Editar Copiar Borrar	18	36.5	72	95	75	120	2021-09-05

Imagen 38. Base de datos en phpMyAdmin

En la barra de la parte superior de la imagen tenemos la posibilidad de filtrar por datos, ya sea valor de signos, fecha o lo que se desee.

Dentro de la base de datos tenemos la posibilidad de crear varias tablas, puede ser desde el terminal como en el caso anterior o, para más comodidad desde la aplicación phpMyAdmin.

Se muestran los pasos:

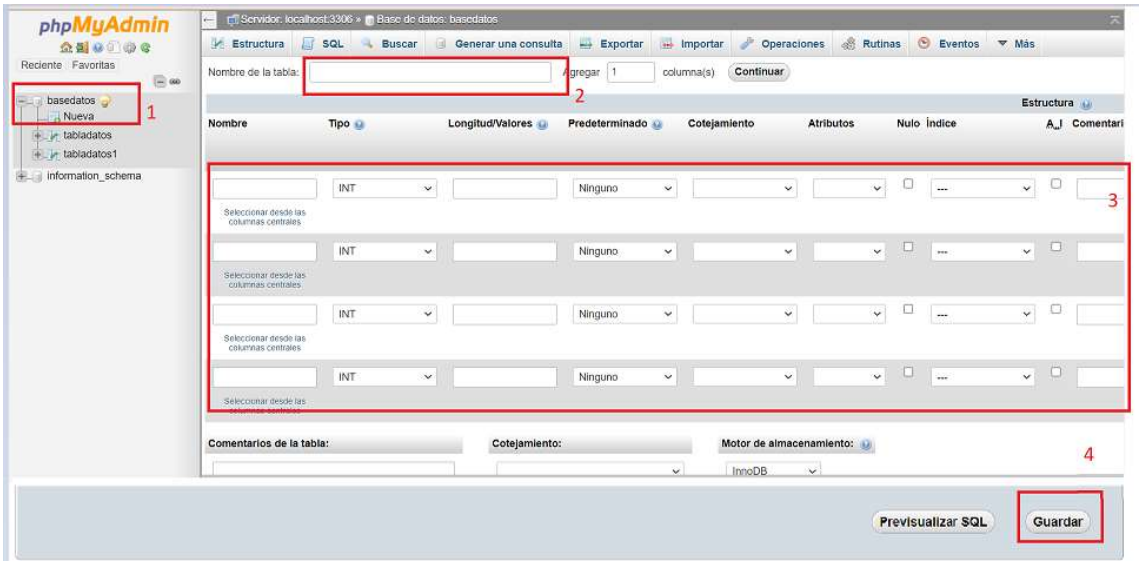


Imagen 39. Crear tabla desde phpMyAdmin

Pulsando en 1 llegamos a la pantalla de la parte derecha de la “Imagen 39”. Escribimos el nombre que queramos darle a la tabla en 2, podemos añadir las columnas que sean necesarias en el apartado “Agregar” justo a continuación del nombre y marcando “Continuar”.

Después indicamos el nombre de las variables y las características que queramos que estas cumplan. Es el mismo procedimiento que cuando hacíamos “CREATE TABLE” así que se pueden seleccionar las mismas opciones que en la tabla anterior.

Hecho esto, en el paso 4, “Guardar”, terminaríamos la construcción de la tabla y la tendríamos disponible para empezar a introducir datos.

7.5 Programar Node-RED

Vamos a analizar cada nodo del flujo de programación realizado en Node-RED. Tiene la forma de la imagen que se muestra.

Algunos nodos vienen instalados en la aplicación pero es posible instalar aquellos que sean necesarios. Para esto hay que ir a la parte superior derecha donde se despliegan varias opciones. Vamos a “manage palette” > “install” y filtramos por nombre.

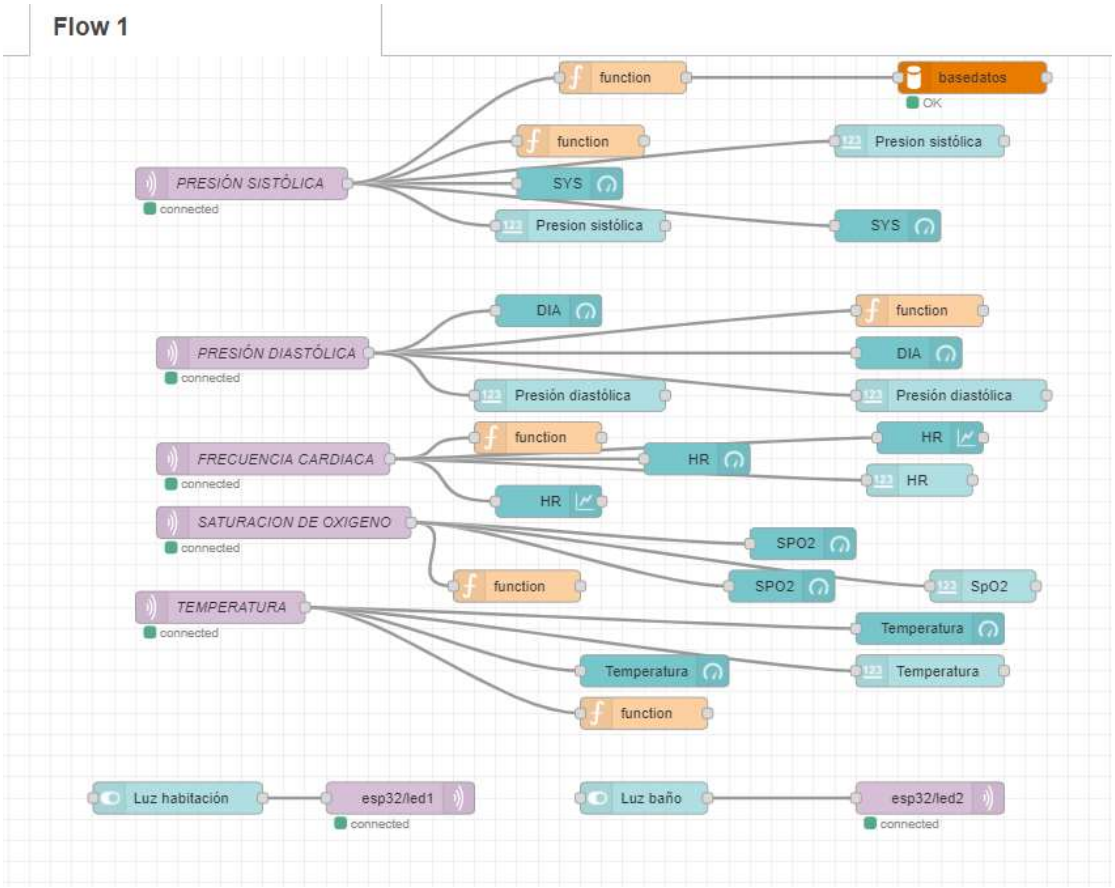


Imagen 40. Programa de nodos en Node-RED

En esta programación los primeros nodos de color lila son mqtt, en las imágenes 40, 41 y 42 se ve lo necesario para que funcionen correctamente. Hace posible la comunicación con Arduino y así podemos ver los datos en el gráfico, en el caso de los signos, con mqtt in y mandar órdenes como es el encendido del led con “mqtt out”.

Los nodos de color azul pertenecen a la familia que compone el gráfico, tenemos indicadores, gráficas, números e interruptores.

Las funciones, en naranja claro, permiten que entre la información a la base de datos de MariaDB, en las salidas de mqtt in asignamos un nombre a cada variable y, en la función que envía la información al nodo naranja oscuro, indicamos en que variable van los valores recogidos en las funciones anteriores.

La función de la base de datos está conectado al primer nodo mqtt in para que pueda activarse por los pulsos que le llegan cada vez que mqtt publica un dato.

El código está adjunto en el documento de Anexos. Copiando y pegando en "Import" conseguiríamos la misma programación de los nodos desde cualquier ordenador.

Para configurar los nodos MQTT hay que especificar la IP de Raspberry Pi donde tenemos el broker instalado, también será necesario especificar el "topic" como indicador del mensaje que tiene que recoger y el puerto reservado, en este caso 1883.

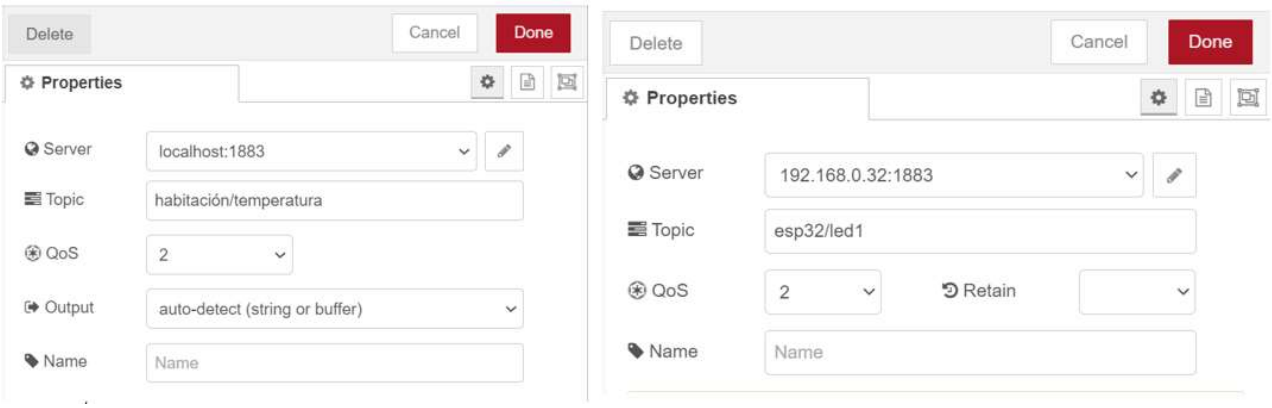


Imagen 41. Configuración de los nodos mqtt in y mqtt out

Además, desde Arduino IDE hemos generado un usuario y contraseña que se introduce en la siguiente imagen:

Imagen 42. Usuario y contraseña MQTT

En el nodo MySQL se indica la dirección localhost, el puerto donde se encuentra MySQL, 3306 y el nombre de la base de datos.

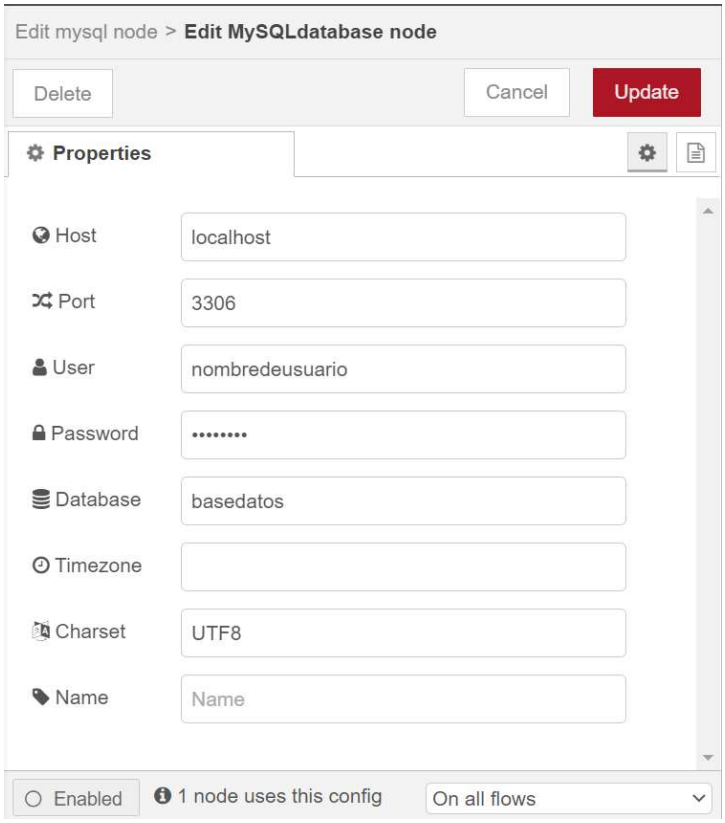


Imagen 43. Configuración del nodo MySQL

El nodo de la base de datos acepta un mensaje tipo topic, de ahí la declaración. Es necesario especificar el nombre de la tabla y las columnas donde introducimos los datos como se ve en la línea 1.

En los nodos “function” tenemos una función para añadir las entradas de mqtt a la base de datos y otras para darle nombre al mensaje “payload” que entra de mqtt y poder configurar la función de la base de datos.

En la imagen del flujo se ve que la primera citada es la que va conectada a la base de datos y las otras tres recogen la información de cada una de las variables.

El código con el que se han programado es el siguiente:

1. Se indica el nombre de la tabla en la que se van a insertar y el valor de cada variable.

```
msg.topic = "INSERT INTO tabladatos1 (temperatura, frecuencia_cardiaca, oxigeno_sat, presion_diastolica,
presion_sistolica) VALUES ('"+global.get("temp")+"','"+global.get("hr")+"','"+global.get("spo2")+"',
'"+global.get("dia")+"','"+global.get("sys")+"')";
return msg;
```

Imagen 44. Función de entrada de datos a la base de datos

2. Se crea una variable para el ritmo cardiaco que hemos llamado “hr” y se iguala al “payload”. Con las funciones set y get conseguimos que ese mensaje llamado “payload” este disponible como variable global con el nombre “hr”. Para el resto de variables se realiza el mismo procedimiento, pero asignando el nombre correspondiente.

```
var hr = msg.payload;
global.set("hr", hr);
msg.payload = global.get("hr");
return msg;
```

Imagen 45. Código para los datos mqtt in de frecuencia cardiaca

```
var spo2 = msg.payload;
global.set("spo2", spo2);
msg.payload = global.get("spo2");
return msg;
```

Imagen 46. Código para los datos mqtt in de saturación de oxígeno

```
var temp = msg.payload;
global.set("temp", temp);
msg.payload = global.get("temp");
return msg;
```

Imagen 47. Código para los datos mqtt in de temperatura corporal

```
1  var sys = msg.payload;
2  global.set("sys", sys);
3  msg.payload = global.get("sys");
4  return msg;
```

Imagen 48. Código para los datos mqtt in de presión sistólica

```
1  var dia = msg.payload;
2  global.set("dia", dia);
3  msg.payload = global.get("dia");
4  return msg;
```

Imagen 49. Código para los datos mqtt in de presión diastólica

Del conjunto “dashboard” se han usado: indicadores, gráficas e interruptores.

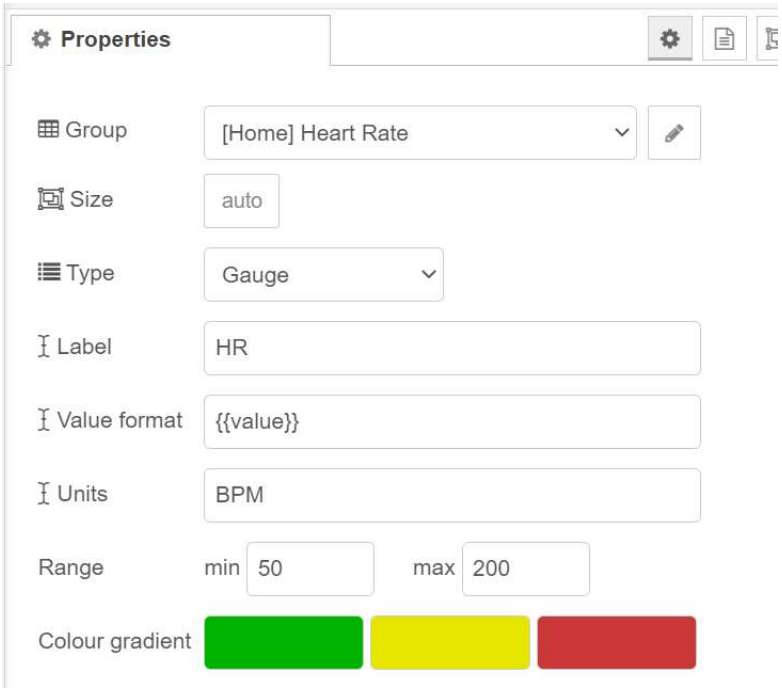


Imagen 50. Configuración de los indicadores

Para los indicadores hay que especificar en que ventana del gráfico la queremos, el nombre, las unidades en que se mide y el rango de valores que es posible mostrar.

También de manera voluntaria se pueden cambiar los colores que cambia según vamos de mínimo a máximo del rango asignado.

Para la gráfica, al igual que antes hay que especificar la ventana y el nombre. Además, existe la opción de elegir una gráfica lineal como es el caso, gráfico de barras o gráfico circular.

Hay que elegir en el eje x cuanto tiempo se quiere graficar, se puede elegir entre minutos, horas, días, etc., también el rango como en el caso del indicador y por último, interpolación, lineal en nuestro caso.

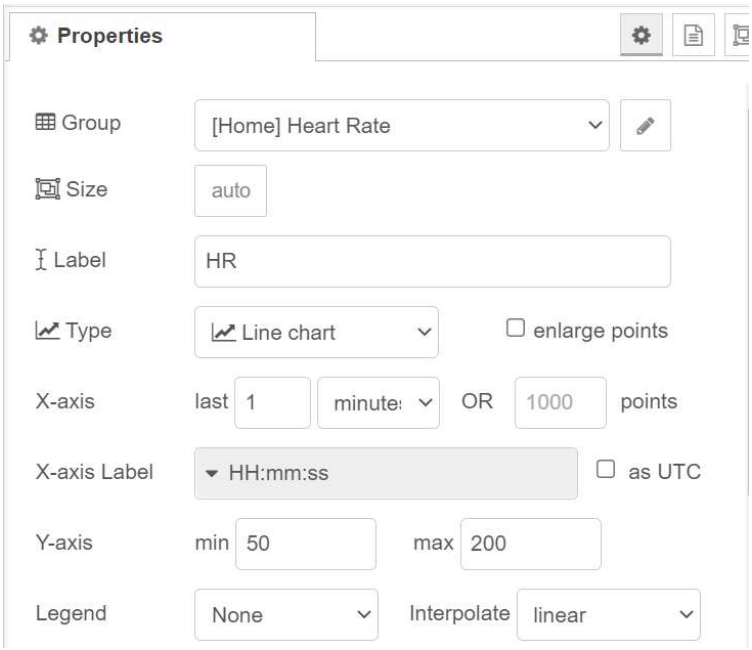


Imagen 51. Configuración de los gráficos

El nodo numérico permite poner los números como se ve en la “Imagen 19” en la parte derecha, esta configuración es la más sencillo, solo hay que elegir el rango de números que va a mostrar.

Para terminar con “dashboard” tenemos los interruptores.

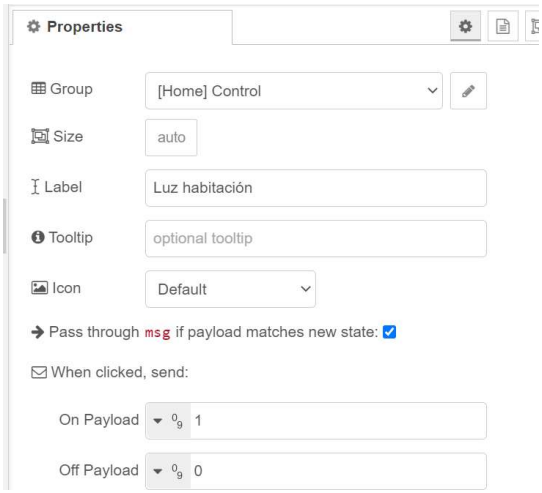


Imagen 52. Configuración de los interruptores

Lo importante en este caso es especificar qué mensaje se va a recibir para encender y apagar. Tenemos 1 para encendido y 0 apagado. Podría ser cualquier cosa como, por ejemplo, true o false, pero es importante especificarlo para que el programa pueda leerlo y funcione correctamente.

Se adjuntan imágenes de las ventanas del gráfico final en la “Imagen 53” e “Imagen 54”.

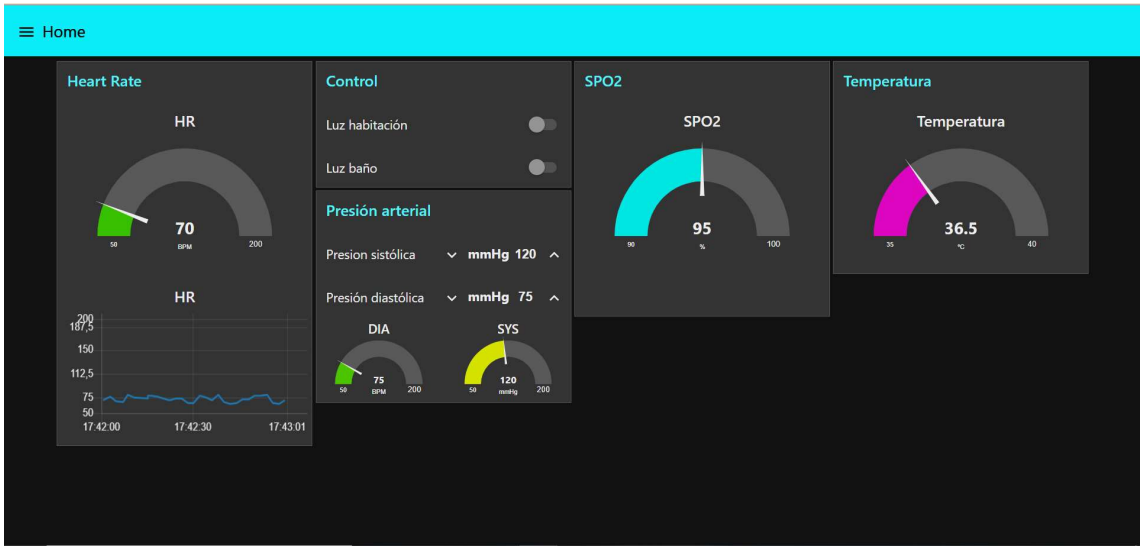


Imagen 53. Interfaz gráfica pestaña general

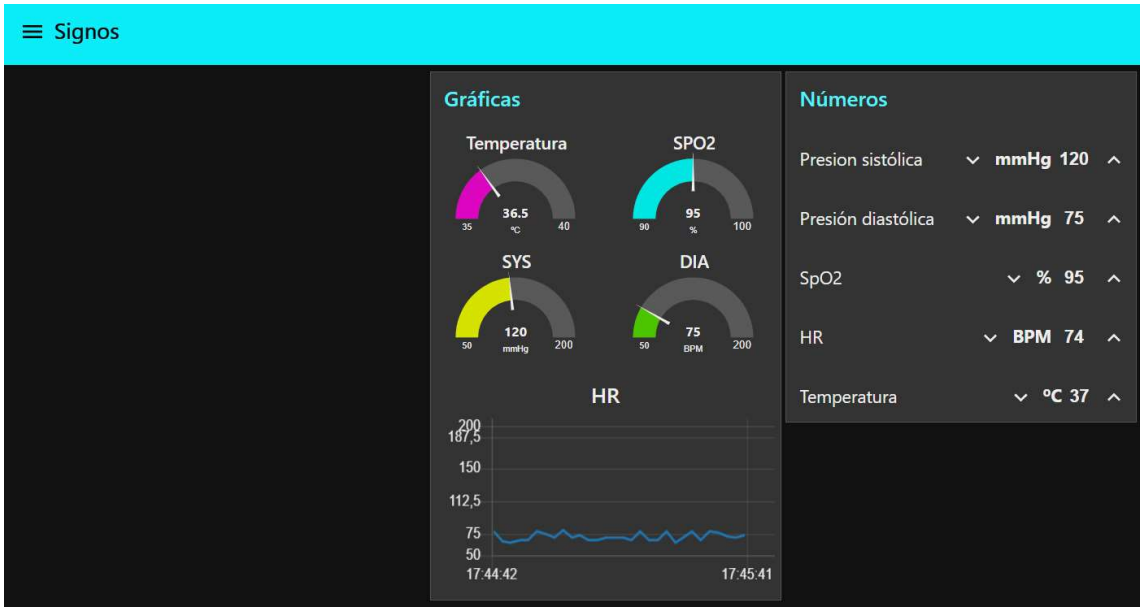


Imagen 54. Interfaz gráfica con la información médica.

7.6 Código QR

Se ha elaborado un código QR con la información de la interfaz gráfica y otro para la base de datos. En la aplicación práctica se crearía uno por habitación de tal manera que se escanee el entrar y recoja la información necesaria.

Se ha hecho con la página referenciada en [23]. Esta página tiene en su pantalla principal un apartado que pone “pega tu enlace aquí”.

Nuestro gráfico se encuentra en la URL de la dirección IP con puerto 1880 de la forma: *192.168.0.32:1880/ui*, se genera un código QR automáticamente que lleva a esta dirección web.

Imagen 55. Código QR dashboard

Para la base de datos se ha creado con la dirección referenciada en [38] y el resultado es como se muestra en la “Imagen 56”.

Imagen 56. Código QR base de datos

8 Conclusiones

Las conclusiones a las que se han llegado tras el desarrollo del proyecto se pueden analizar diferenciando entre hardware y software.

En el primer grupo entra Raspberry Pi. Este miniordenador ha dado muy buenos resultados en esta aplicación. Ha permitido una fluida comunicación a la vez que sin errores entre ESP32 y Node-RED vía MQTT.

También ha hecho posible la instalación de phpMyAdmin y con ello la visualización de la base de datos de manera cómoda y de fácil interpretación.

El módulo wifi ESP32 ha sido una buena elección por la comodidad de programarlo en Arduino IDE y toda la información sobre este software y códigos para él que se encuentran en internet. Además, cuenta con gran variedad de pines y esto ha permitido que no nos encontráramos con ningún problema a la hora de conectar los sensores.

Sin embargo, la conexión wifi es débil aunque estable, es decir, no ha presentado problemas de desconexión pero si necesidad de cercanía para conectarse a la red.

Para terminar con el hardware, los sensores en ocasiones han presentado medidas inestables o desconexión en un ligero movimiento.

Node-RED ha permitido una programación muy cómoda. Al tener tan variedad de nodos y facilidad para comunicarlos, ha hecho sencilla la forma de entender la comunicación entre ellos para poder desarrollar la aplicación esperada.

PhpMyadmin conectado con el nodo MySQL ha mostrado una captura perfecta de los datos enviados desde el entorno de Node-RED. En cada pulso que se ha marcado para la actualización de los valores ha sido capaz de recogerlos y mostrarlos tal y como se indicaba.

Por último, la herramienta más valorada ha sido MQTT, sin la comunicación entre todo el hardware y software usado no se podría haber llevado a cabo el proyecto y esto ha sido gracias a este protocolo de comunicación.

9 Futuras líneas

La idea de la ampliación es que además de tener un monitor de signos vitales visualizable desde un dispositivo móvil, sería interesante que desde este mismo dispositivo se pueda controlar el resto de la habitación.

Esto quiere decir, el control de luces, tanto fijas como regulables para la posibilidad de escenas, subida y bajada de la cama para el ajuste a gusto del paciente, sensores de incendio e inundación, un pulsador de emergencia útil tanto para el paciente como para el médico y un simple timbre de llamada para cualquier necesidad y la climatización de la habitación, conexión con el aparato de aire acondicionado y calefacción.

En cuanto a la aplicación meramente médica, se podría programar unas señales de aviso con el fin de controlar los dispositivos encargados del suministro del medicamento al paciente. Estas señales de aviso también sería interesante programarlas para el control en caso de que las medidas de las constantes salgan del rango establecido como no perjudicial para la salud.

10 Bibliografía general

1. alldatasheet.com. (s. f.). *ESP32-WROOM-32 pdf, ESP32-WROOM-32 description, ESP32-WROOM-32 datasheets, ESP32-WROOM-32 view ::: ALLDATASHEET :::* All Datasheet. Recuperado 8 de septiembre de 2021, de https://pdf1.alldatasheet.com/datasheet-pdf/view/1148026/ESPRESSIF/ESP32-WROOM-32.html

2. Baena, M. R. (2020, 18 marzo). *IoT en el sector salud: aplicaciones y beneficios*. App&Web. https://www.appandweb.es/blog/iot-sector-salud/

3. Boot & Work Corp. S.L., Berta Canet. (2021, 8 septiembre). *Fundamentos de seguridad del MQTT en la automatización industrial*. Boot & Work Corp. S.L. https://www.industrialshields.com/es_ES/blog/blog-industrial-open-source-1/post/fundamentos-de-seguridad-del-mqtt-en-la-automatizacion-industrial-235

4. Boyett, R. (2021, 19 agosto). *What is MySQL: MySQL Explained For Beginners*. Hostinger Tutorials. https://www.hostinger.com/tutorials/what-is-mysql

5. Colaboradores de Wikipedia. (2021, 25 agosto). *Oxímetro de pulso (pulsioxímetro)*. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Ox%C3%ADmetro_de_pulso_(pulsiox%C3%ADmetro)

6. de Girodmedical, M. (2021, 8 abril). *Termómetro infrarrojos: cómo funciona y mejores modelos*. El blog de Girodmedical. https://www.girodmedical.es/blog_es/como-funciona-un-termometro-infrarrojo/

7. designthemes. (2019, 3 octubre). *Midiendo temperatura sin contacto*. Tienda y Tutoriales Arduino. https://www.prometec.net/midiendo-temperatura-sin-contacto/

8. *Difference Between Raspberry Pi and Arduino: Which Is Better for IoT Project*. (2021, 10 agosto). Digiteum. https://www.digiteum.com/comparing-arduino-raspberry-pi-iot/

9. *Donate to*. (s. f.). Arduino. Recuperado 8 de septiembre de 2021, de https://www.arduino.cc/en/donate/

10. *Electrocardiograma (ECG) - Mayo Clinic*. (2021, 29 julio). Mayo Clinic. https://www.mayoclinic.org/es-es/tests-procedures/ekg/about/pac-20384983#:%7E:text=Un%20electrocardiograma%20(ECG%20o%20EKG,computadora%20o%20una%20impresora%20adjuntos.

11. Fernández, Y. (2020, 3 agosto). *Qué es Arduino, cómo funciona y qué puedes hacer con uno*. Xataka. https://www.xataka.com/basics/que-arduino-como-funciona-que-puedes-hacer-uno

12. Holland, K. (2019, 28 septiembre). *¿Es normal mi nivel de oxígeno en la sangre?* Healthline. https://www.healthline.com/health/es/saturacion-de-oxigeno-normal

13. I. (2018, 2 octubre). *Controle la Hipertensión Arterial con los equipos médicos de Biomedic Import*. Biomedic. https://biomedicimport.cl/como-leer-un-monitor-de-signos-vitales-equipos-medicos/

14. I. (2019, 24 enero). *Software de Arduino*. Arduino.cl. https://arduino.cl/programacion/#:%7E:text=El%20IDE%20de%20Arduino%20nos,de%20Arduino%2C%20a%20su%20accesibilidad.

15. J. (2020a, marzo 5). *Qué es Node-RED*. Aprendiendo Arduino. https://aprendiendoarduino.wordpress.com/2020/03/05/que-es-node-red/

16. L. (2020b, mayo 2). *El bus I2C en Arduino*. Luis Llamas. https://www.luisllamas.es/arduino-i2c/

17. *MQTT - The Standard for IoT Messaging*. (s. f.). MQTT. Recuperado 8 de septiembre de 2021, de https://mqtt.org/

18. *MQTT vs CoAP, the battle to become the best IoT protocol*. (s. f.). Pickdata. Recuperado 8 de septiembre de 2021, de https://www.pickdata.net/es/noticias/mqtt-vs-coap-mejor-protocolo-iot

19. *Node-RED*. (s. f.). Node-RED. Recuperado 8 de septiembre de 2021, de https://nodered.org/

20. *node-red-node-serialport*. (s. f.). Node-RED. Recuperado 8 de septiembre de 2021, de https://flows.nodered.org/node/node-red-node-serialport

21. Perú, M. N.-. (s. f.). *Resistencias Pull-Up y Pull-Down*. Naylamp Mechatronics. Recuperado 8 de septiembre de 2021, de https://naylampmechatronics.com/blog/39_resistencias-pull-up-y-pull-down.html

22. *Pulsioxímetro MAX30102*. (s. f.). Naylamp Mechatronics. Recuperado 8 de septiembre de 2021, de https://naylampmechatronics.com/biomedico/444-pulsioximetro-max30102.html

23. *QR code generator - free | Make QR code for link, image or PDF file - ME-QR*. (s. f.). Me QR. Recuperado 8 de septiembre de 2021, de https://me-qr.com/

24. *¿Qué es Internet of Things (IoT)?* (s. f.). oracle. Recuperado 8 de septiembre de 2021, de https://www.oracle.com/es/internet-of-things/what-is-iot/#link0

25. *¿Qué son los Pulsioxímetros y cómo funcionan?* (s. f.). Grupo R. Queraltó. Recuperado 8 de septiembre de 2021, de https://www.queralto.com/blog/que-son-los-pulsioximetros-y-como-funcionan-n241#:%7E:text=El%20pulsiox%C3%ADmetro%20mide%20la%20saturaci%C3%B3n,frecuencia%20card%C3%ADaca%20y%20curva%20de

26. R. (2020c, julio 29). *HealthyPi, monitor de signos vitales de código abierto para Raspberry Pi*. Descubrearduino.com. https://descubrearduino.com/healthypi/

27. S. (2020d, abril 21). *7 alternativas a la Raspberry Pi muy interesantes »*. Raspberry para novatos. https://raspberryparanovatos.com/articulos/alternativas-a-la-raspberry-pi/

28. *Sistemas de monitoreo fisiológico para cuidados agudos*. (s. f.). El Hospital. Recuperado 8 de septiembre de 2021, de https://www.elhospital.com/temas/Sistemas-de-monitoreo-fisiologico-de-electrocardiografia-en-cuidados-agudos-y-neonatales+8083904

29. SolidBi. (2021, 7 junio). *Funciones de SOLIDWORKS CAD 3D*. https://solid-bi.es/funciones-solidworks/

30. *Sp02 - VitalScan*. (s. f.). VitalScan. Recuperado 8 de septiembre de 2021, de http://www.vitalscan.es/dtr_pwv_spo2_sp.htm

31. *TCP/IP | | UPV*. (2011, 21 septiembre). YouTube. https://www.youtube.com/watch?v=tmbSIv-Uwqc

32. *Termómetro infrarrojo MLX90614 - Melexis | DigiKey - Google zoeken*. (s. f.). Google. Recuperado 8 de septiembre de 2021, de https://www.google.com/search?q=Term%C3%B3metro+infrarrojo+MLX90614+-+Melexis+%7C+DigiKey&oq=Term%C3%B3metro+infrarrojo+MLX90614+-+Melexis+%7C+DigiKey&aqs=chrome.0.69i59.702j0j7&sourceid=chrome&ie=UTF-8

33. Ultra-lab. (2019a, diciembre 12). *Pulse Sensor de SparkFun - SEN-11574 - Comprar en España*. http://ultra-lab.net/producto/pulse-sensor-sparkfun-sen-11574/

34. *What is MySQL and how do I use it?* (s. f.). 123 Reg Support. Recuperado 8 de septiembre de 2021, de https://www.123-reg.co.uk/support/servers/what-is-mysql-and-why-do-i-need-it/

35. https://www.fbbva.es/microsites/salud_cardio/mult/fbbva_libroCorazon_cap4.pdf

36. http://www.cenetec.salud.gob.mx/descargas/biomedica/guias_tecnologicas/13gt_monitores.pdf

DOCUMENTO II: ANEXOS

Monica Moreno Caballero

1. Código de Arduino IDE

```
#include <WiFi.h>
#include <PubSubClient.h>
#include <Wire.h>
#include "MAX30105.h"
#include "spo2_algorithm.h"
#include <Adafruit_MLX90614.h>

#define pin1          13
#define pin2          12

#define MAX_BRIGHTNESS 255

WiFiClient espClient;
PubSubClient client(espClient);
MAX30105 particleSensor;
Adafruit_MLX90614 mlx = Adafruit_MLX90614();

//variables para spo2
#if defined(__AVR_ATmega328P__) || defined(__AVR_ATmega168__)
uint16_t irBuffer[100]; //infrared LED sensor data
uint16_t redBuffer[100];  //red LED sensor data
#else
uint32_t irBuffer[100]; //infrared LED sensor data
uint32_t redBuffer[100];  //red LED sensor data
#endif

//variables spo2
int32_t tam;
int32_t spo2;
int8_t bool_spo2;
```

```
int32_t heartRate;
int8_t bool_hr;

//variables para MQTT
char str_hr[10];
char str_spo2[10];
char str_temp[10];
char str_sys[10];
char str_dia[10];
char str_ecg[10];

//datos router
const char* ssid = "vodafoneCDD1";
const char* clave = "EK9rnG4sPmP4dEPr";

//datos MQTT
const char* mqtt_servidor = "192.168.0.32";
int mqtt_puerto = 1883;
const char* mqtt_usuario = "monica";
const char* mqtt_clave = "monica";

unsigned long ultimoMsg = 0;
char msg[50];
int valor = 0;

int k=0, j=0;
float heartRate_suma=0, heartRate_media = 0;

void setup_wifi() {

  delay(10);
  // We start by connecting to a WiFi network
```

```
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, clave);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  randomSeed(micros());

  Serial.println("");
  Serial.println("WiFi conectado");
  Serial.println("Direccion IP: ");
  Serial.println(WiFi.localIP());
}

void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Mensaje recibido :");
  Serial.print(topic);

  if (strstr(topic,"esp32/led1"))
  {
    for (int i = 0; i < length; i++) {
      Serial.print((char)payload[i]);
    }
    Serial.println();
    if ((char)payload[0] == '1') {
      digitalWrite(pin1, HIGH);
```

```
    } else {
      digitalWrite(pin1, LOW);  // Turn the LED off by making the voltage HIGH
    }
  }

  else if ( strstr(topic, "esp32/led2"))
  {
    for (int i = 0; i < length; i++) {
      Serial.print((char)payload[i]);
    }
    Serial.println();
    if ((char)payload[0] == '1') {
      digitalWrite(pin2, HIGH);
    }
    else {
      digitalWrite(pin2, LOW);
    }
  }
  else
  {
    Serial.println("No se ha suscrito a ningún topic.");
  }
}

void reconnect() {
  while (!client.connected()) {
    Serial.print("Conectando MQTT...");
    String clientId = "ESP8266Client-";
    clientId += String(random(0xffff), HEX);

    if (client.connect(clientId.c_str(), mqtt_usuario, mqtt_clave) ) {
      Serial.println("conexion OK");
```

```
      client.subscribe("esp32/led1");
      client.subscribe("esp32/led2");
    } else {
      Serial.print("No se ha podido conectar, rc=");
      Serial.print(client.state());
      Serial.println("Reintentar en 5 segundos");
      delay(5000);
    }
  }
}

void setup() {

  pinMode(pin1, OUTPUT);
  pinMode(pin2, OUTPUT);

  Serial.begin(115200);

  Serial.println("Initializing...");

  mlx.begin();

  if (!particleSensor.begin(Wire, 0x57)) //Use default I2C port, 400kHz speed
  {
    Serial.println("MAX30105 was not found. Please check wiring/power. ");
    while (1);
  }
  Serial.println("Place your index finger on the sensor with steady pressure.");
  Serial.read();

  //indicado según la hoja del fabricante
  byte ledBrightness = 60; //Opciones: 0=Off to 255=50mA
```

```
  byte sampleAverage = 4; //Opciones: 1, 2, 4, 8, 16, 32
  byte ledMode = 2; //Opciones: 1 = Red only, 2 = Red + IR, 3 = Red + IR + Green
  byte sampleRate = 100; //Opciones: 50, 100, 200, 400, 800, 1000, 1600, 3200
  int pulseWidth = 411; //Opciones: 69, 118, 215, 411
  int adcRange = 4096; //Opciones: 2048, 4096, 8192, 16384

  particleSensor.setup();
  particleSensor.setPulseAmplitudeRed(0x0A);
  particleSensor.setPulseAmplitudeGreen(0);
  particleSensor.setup(ledBrightness, sampleAverage, ledMode, sampleRate, pulseWidth, adcRange);

  setup_wifi();

  client.setServer(mqtt_servidor, mqtt_puerto);
  client.setCallback(callback);

}

void loop() {

  if (!client.connected()) {
    reconnect();
  }
  client.loop();

   tam = 100;

  for (byte i = 0 ; i < tam ; i++)
  {
    while (particleSensor.available() == false)
      particleSensor.check();
```

```
    redBuffer[i] = particleSensor.getRed();
    irBuffer[i] = particleSensor.getIR();
    particleSensor.nextSample();

    Serial.print(F("red="));
    Serial.print(redBuffer[i], DEC);
    Serial.print(F(", ir="));
    Serial.println(irBuffer[i], DEC);
  }

  maxim_heart_rate_and_oxygen_saturation(irBuffer, tam, redBuffer, &spo2,
&bool_spo2, &heartRate, &bool_hr);

  while (1)
  {
    for (byte i = 25; i < 100; i++)
    {
      redBuffer[i - 25] = redBuffer[i];
      irBuffer[i - 25] = irBuffer[i];
    }

    for (byte i = 75; i < 100; i++)
    {
      while (particleSensor.available() == false)
        particleSensor.check();

      redBuffer[i] = particleSensor.getRed();
      irBuffer[i] = particleSensor.getIR();
      particleSensor.nextSample();

      Serial.print(F("red="));
```

```
    Serial.print(redBuffer[i], DEC);
    Serial.print(F(", ir="));
    Serial.print(irBuffer[i], DEC);

    Serial.print(F(", SPO2="));
    Serial.print(spo2);

    Serial.print(F(", HR="));
    Serial.print(heartRate);

    heartRate_suma+=heartRate;
    k++;
    heartRate_media = heartRate_suma/k;

    Serial.print ("MEDIA: ");
    Serial.print (heartRate_media);
    Serial.println ();

  maxim_heart_rate_and_oxygen_saturation(irBuffer, tam, redBuffer, &spo2,
&bool_spo2, &heartRate, &bool_hr);


  unsigned long now = millis();
  if (now - ultimoMsg > 2000) {

  double temp = mlx.readObjectTempC();
  double sys = 120;
  double dia = 70;

  dtostrf(heartRate_media, 4, 2, str_hr);
  dtostrf(temp, 4, 2, str_temp);
  dtostrf(spo2, 4, 2, str_spo2);
```

```
    dtostrf(sys, 4, 2, str_sys);
    dtostrf(dia, 4, 2, str_dia);

    ultimoMsg = now;

    Serial.print("Publish message: ");
    Serial.print("Temperature - "); Serial.println(str_temp);
    client.publish("esp32/temp", str_temp);
    Serial.print("HR - "); Serial.println(str_hr);
    client.publish("esp32/hr", str_hr);
    Serial.print("SPO2 - "); Serial.println(str_spo2);
    client.publish("esp32/spo2", str_spo2);
    Serial.print("Presion sistólica - "); Serial.println(str_sys);
    client.publish("esp32/sys", str_sys);
    Serial.print("Presión diastólica - "); Serial.println(str_dia);
    client.publish("esp32/dia", str_dia);
    delay (1000);
      }
    }
  }
}
```

2. Código Node-RED


```
[
    {
        "id": "4dd134790857fcaa",
        "type": "tab",
        "label": "Flow 1",
        "disabled": false,
        "info": ""
    },
    {
        "id": "02bfbe9e8c0a3ae8",
        "type": "mqtt in",
        "z": "4dd134790857fcaa",
        "name": "FRECUENCIA CARDIACA",
        "topic": "esp32/hr",
        "qos": "2",
        "datatype": "auto",
        "broker": "3d96af0795295548",
        "nl": false,
        "rap": true,
        "rh": 0,
        "x": 270,
        "y": 380,
        "wires": [
            [
                "f5aac720cdbd61ec",
                "4b31852b39bbef0d",
                "033db1cfcd151c66",
                "db54a1cf79da0066",
                "b8af81ec9c247b7d"
            ]
```

```
    ]
},
{
    "id": "1ecbcb26ef544e2d",
    "type": "mqtt in",
    "z": "4dd134790857fcaa",
    "name": "SATURACION DE OXIGENO",
    "topic": "esp32/spo2",
    "qos": "2",
    "datatype": "auto",
    "broker": "3d96af0795295548",
    "nl": false,
    "rap": true,
    "rh": 0,
    "x": 280,
    "y": 440,
    "wires": [
        [
            "9b2fd031a7a2312a",
            "adda92b61aa96fd5",
            "00ef03623790a9d1",
            "7a6f38d5f21736f8"
        ]
    ]
},
{
    "id": "1f219816922bdf9f",
    "type": "mqtt out",
    "z": "4dd134790857fcaa",
    "name": "",
```

"topic": "esp32/led1",
"qos": "2",
"retain": "",
"respTopic": "",
"contentType": "",
"userProps": "",
"correl": "",
"expiry": "",
"broker": "3d96af0795295548",
"x": 390,
"y": 700,
"wires": []
},
{
"id": "fb244c2b2be4383c",
"type": "mqtt out",
"z": "4dd134790857fcaa",
"name": "",
"topic": "esp32/led2",
"qos": "",
"retain": "",
"respTopic": "",
"contentType": "",
"userProps": "",
"correl": "",
"expiry": "",
"broker": "3d96af0795295548",
"x": 890,
"y": 700,
"wires": []

```
},
{
    "id": "9dd7a65316f6382a",
    "type": "ui_switch",
    "z": "4dd134790857fcaa",
    "name": "",
    "label": "Luz habitación",
    "tooltip": "",
    "group": "746a362c9103fa5c",
    "order": 0,
    "width": 0,
    "height": 0,
    "passthru": true,
    "decouple": "false",
    "topic": "topic",
    "topicType": "msg",
    "style": "",
    "onvalue": "1",
    "onvalueType": "num",
    "onicon": "",
    "oncolor": "",
    "offvalue": "0",
    "offvalueType": "num",
    "officon": "",
    "offcolor": "",
    "animate": false,
    "x": 180,
    "y": 700,
    "wires": [
        [
```

```
            "1f219816922bdf9f"
        ]
    ]
},
{
    "id": "9b2fd031a7a2312a",
    "type": "ui_gauge",
    "z": "4dd134790857fcaa",
    "name": "",
    "group": "31d732c6059e557a",
    "order": 2,
    "width": 0,
    "height": 0,
    "gtype": "gage",
    "title": "SPO2",
    "label": "%",
    "format": "{{value}}",
    "min": "80",
    "max": "100",
    "colors": [
        "#c5e811",
        "#00e6e2",
        "#3acb6c"
    ],
    "seg1": "",
    "seg2": "",
    "x": 750,
    "y": 500,
    "wires": []
},
```

```
{
    "id": "176b34c024100fe1",
    "type": "ui_switch",
    "z": "4dd134790857fcaa",
    "name": "",
    "label": "Luz baño",
    "tooltip": "",
    "group": "746a362c9103fa5c",
    "order": 0,
    "width": 0,
    "height": 0,
    "passthru": true,
    "decouple": "false",
    "topic": "topic",
    "topicType": "msg",
    "style": "",
    "onvalue": "1",
    "onvalueType": "num",
    "onicon": "",
    "oncolor": "",
    "offvalue": "0",
    "offvalueType": "num",
    "officon": "",
    "offcolor": "",
    "animate": false,
    "x": 620,
    "y": 700,
    "wires": [
        [
            "fb244c2b2be4383c"
```

```
            ]
        ]
    },
    {
        "id": "f5aac720cdbd61ec",
        "type": "ui_chart",
        "z": "4dd134790857fcaa",
        "name": "",
        "group": "e1ed393d076e8195",
        "order": 4,
        "width": "0",
        "height": "0",
        "label": "HR",
        "chartType": "line",
        "legend": "false",
        "xformat": "HH:mm:ss",
        "interpolate": "linear",
        "nodata": "",
        "dot": false,
        "ymin": "50",
        "ymax": "200",
        "removeOlder": 1,
        "removeOlderPoints": "",
        "removeOlderUnit": "60",
        "cutout": 0,
        "useOneColor": false,
        "useUTC": false,
        "colors": [
            "#1f77b4",
            "#aec7e8",
```

```
            "#ff7f0e",
            "#2ca02c",
            "#98df8a",
            "#d62728",
            "#ff9896",
            "#9467bd",
            "#c5b0d5"
        ],
        "outputs": 1,
        "useDifferentColor": false,
        "x": 530,
        "y": 420,
        "wires": [
            []
        ]
    },
    {
        "id": "a72a979d8d3adcb3",
        "type": "mqtt in",
        "z": "4dd134790857fcaa",
        "name": "TEMPERATURA",
        "topic": "esp32/temp",
        "qos": "2",
        "datatype": "auto",
        "broker": "3d96af0795295548",
        "nl": false,
        "rap": true,
        "rh": 0,
        "x": 240,
        "y": 600,
```

```
    "wires": [
        [
            "6a43349f15fd6012",
            "73134923ccd5e4b5",
            "ce656cf6f38ec209",
            "d5e76ad8fd97cc10"
        ]
    ]
},
{
    "id": "6a43349f15fd6012",
    "type": "ui_gauge",
    "z": "4dd134790857fcaa",
    "name": "",
    "group": "af528003b9d19ab0",
    "order": 2,
    "width": 0,
    "height": 0,
    "gtype": "gage",
    "title": "Temperatura",
    "label": "ºC",
    "format": "{{value}}",
    "min": "35",
    "max": "40",
    "colors": [
        "#cf11e8",
        "#e600a8",
        "#ef0606"
    ],
    "seg1": "",
```

"seg2": "",

"x": 630,

"y": 600,

"wires": []

},

{

"id": "4b31852b39bbef0d",

"type": "function",

"z": "4dd134790857fcaa",

"name": "",

"func": "var hr = msg.payload;\nglobal.set(\"hr\", hr);\nmsg.payload = global.get(\"hr\");\nreturn msg;",

"outputs": 1,

"noerr": 0,

"initialize": "",

"finalize": "",

"libs": [],

"x": 520,

"y": 360,

"wires": [

[]

]

},

{

"id": "adda92b61aa96fd5",

"type": "function",

"z": "4dd134790857fcaa",

"name": "",

"func": "var spo2 = msg.payload;\nglobal.set(\"spo2\", spo2);\nmsg.payload = global.get(\"spo2\");\nreturn msg;",

"outputs": 1,

```
        "noerr": 0,
        "initialize": "",
        "finalize": "",
        "libs": [],
        "x": 500,
        "y": 500,
        "wires": [
            []
        ]
    },
    {
        "id": "73134923ccd5e4b5",
        "type": "function",
        "z": "4dd134790857fcaa",
        "name": "",
        "func": "var temp = msg.payload;\nglobal.set(\"temp\",
temp);\nmsg.payload = global.get(\"temp\");\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "initialize": "",
        "finalize": "",
        "libs": [],
        "x": 620,
        "y": 640,
        "wires": [
            []
        ]
    },
    {
        "id": "9903346f737547d2",
        "type": "function",
```

"z": "4dd134790857fcaa",
"name": "",
"func": "msg.topic = \"INSERT INTO tabladatos1 (temperatura, frecuencia_cardiaca, oxigeno_sat, presion_diastolica, presion_sistolica) VALUES ('\"+global.get(\"temp\")+\"','\"+global.get(\"hr\")+\"','\"+global.get(\"spo2\")+\"','\"+global.get(\"dia\")+\"','\"+global.get(\"sys\")+\"')\";\nreturn msg;",
"outputs": 1,
"noerr": 0,
"initialize": "",
"finalize": "",
"libs": [],
"x": 600,
"y": 20,
"wires": [
[
"3cd8b4366a9554e3"
]
]
},
{
"id": "7fe38d4d8b9a956f",
"type": "mqtt in",
"z": "4dd134790857fcaa",
"name": "PRESIÓN DIASTÓLICA",
"topic": "esp32/dia",
"qos": "2",
"datatype": "auto",
"broker": "3d96af0795295548",
"nl": false,
"rap": true,
"rh": 0,

```
    "x": 260,
    "y": 280,
    "wires": [
      [
        "2ab17855730ac71f",
        "0bc21479bfe4613b",
        "41600327f599998f",
        "61a39904967ffb84",
        "d490f7a349994d45"
      ]
    ]
  },
  {
    "id": "93e59ca81368c6d3",
    "type": "mqtt in",
    "z": "4dd134790857fcaa",
    "name": "PRESIÓN SISTÓLICA",
    "topic": "esp32/sys",
    "qos": "2",
    "datatype": "auto",
    "broker": "3d96af0795295548",
    "nl": false,
    "rap": true,
    "rh": 0,
    "x": 240,
    "y": 120,
    "wires": [
      [
        "1fd2d0f89b4b1ad7",
        "05bfd2b75de8d6e4",
```

```
            "205edb5be58b921e",
            "f627933e8976e29c",
            "e2ad3e80581b07fb",
            "9903346f737547d2"
        ]
    ]
},
{
    "id": "0bc21479bfe4613b",
    "type": "ui_gauge",
    "z": "4dd134790857fcaa",
    "name": "",
    "group": "18bae98e7738c922",
    "order": 3,
    "width": "3",
    "height": "2",
    "gtype": "gage",
    "title": "DIA",
    "label": "BPM",
    "format": "{{value}}",
    "min": "50",
    "max": "200",
    "colors": [
        "#00b500",
        "#e6e600",
        "#ca3838"
    ],
    "seg1": "",
    "seg2": "",
    "x": 870,
```

"y": 280,
"wires": []
},
{
"id": "2ab17855730ac71f",
"type": "function",
"z": "4dd134790857fcaa",
"name": "",
"func": "var dia = msg.payload;\nglobal.set(\"dia\", dia);\nmsg.payload = global.get(\"dia\");\nreturn msg;",
"outputs": 1,
"noerr": 0,
"initialize": "",
"finalize": "",
"libs": [],
"x": 880,
"y": 240,
"wires": [
[]
]
},
{
"id": "1fd2d0f89b4b1ad7",
"type": "ui_gauge",
"z": "4dd134790857fcaa",
"name": "",
"group": "18bae98e7738c922",
"order": 3,
"width": "3",
"height": "2",
"gtype": "gage",

"title": "SYS",
"label": "mmHg",
"format": "{{value}}",
"min": "50",
"max": "200",
"colors": [
"#00b500",
"#e6e600",
"#ca3838"
],
"seg1": "",
"seg2": "",
"x": 550,
"y": 120,
"wires": []
},
{
"id": "05bfd2b75de8d6e4",
"type": "function",
"z": "4dd134790857fcaa",
"name": "",
"func": "var sys = msg.payload;\nglobal.set(\"sys\", sys);\nmsg.payload = global.get(\"sys\");\nreturn msg;",
"outputs": 1,
"noerr": 0,
"initialize": "",
"finalize": "",
"libs": [],
"x": 560,
"y": 80,
"wires": [

```
            []
        ]
    },
    {
        "id": "41600327f599998f",
        "type": "ui_numeric",
        "z": "4dd134790857fcaa",
        "name": "",
        "label": "Presión diastólica",
        "tooltip": "",
        "group": "18bae98e7738c922",
        "order": 2,
        "width": "0",
        "height": "0",
        "wrap": false,
        "passthru": true,
        "topic": "topic",
        "topicType": "msg",
        "format": "mmHg",
        "min": "60",
        "max": "120",
        "step": 1,
        "x": 910,
        "y": 320,
        "wires": [
            []
        ]
    },
    {
        "id": "205edb5be58b921e",
```

```
    "type": "ui_numeric",
    "z": "4dd134790857fcaa",
    "name": "",
    "label": "Presion sistólica",
    "tooltip": "",
    "group": "7b57f931b680826c",
    "order": 1,
    "width": 0,
    "height": 0,
    "wrap": false,
    "passthru": true,
    "topic": "topic",
    "topicType": "msg",
    "format": "mmHg",
    "min": "100",
    "max": "150",
    "step": 1,
    "x": 880,
    "y": 80,
    "wires": [
        []
    ]
},
{
    "id": "f627933e8976e29c",
    "type": "ui_gauge",
    "z": "4dd134790857fcaa",
    "name": "",
    "group": "5495366eb67a0127",
    "order": 3,
```

```
        "width": "3",
        "height": "2",
        "gtype": "gage",
        "title": "SYS",
        "label": "mmHg",
        "format": "{{value}}",
        "min": "50",
        "max": "200",
        "colors": [
            "#00b500",
            "#e6e600",
            "#ca3838"
        ],
        "seg1": "",
        "seg2": "",
        "x": 850,
        "y": 160,
        "wires": []
    },
    {
        "id": "e2ad3e80581b07fb",
        "type": "ui_numeric",
        "z": "4dd134790857fcaa",
        "name": "",
        "label": "Presion sistólica",
        "tooltip": "",
        "group": "18bae98e7738c922",
        "order": 1,
        "width": 0,
        "height": 0,
```

```
    "wrap": false,
    "passthru": true,
    "topic": "topic",
    "topicType": "msg",
    "format": "mmHg",
    "min": "100",
    "max": "150",
    "step": 1,
    "x": 560,
    "y": 160,
    "wires": [
        []
    ]
},
{
    "id": "61a39904967ffb84",
    "type": "ui_gauge",
    "z": "4dd134790857fcaa",
    "name": "",
    "group": "5495366eb67a0127",
    "order": 4,
    "width": "3",
    "height": "2",
    "gtype": "gage",
    "title": "DIA",
    "label": "BPM",
    "format": "{{value}}",
    "min": "50",
    "max": "200",
    "colors": [
```

```
            "#00b500",
            "#e6e600",
            "#ca3838"
        ],
        "seg1": "",
        "seg2": "",
        "x": 530,
        "y": 240,
        "wires": []
    },
    {
        "id": "d490f7a349994d45",
        "type": "ui_numeric",
        "z": "4dd134790857fcaa",
        "name": "",
        "label": "Presión diastólica",
        "tooltip": "",
        "group": "7b57f931b680826c",
        "order": 2,
        "width": "0",
        "height": "0",
        "wrap": false,
        "passthru": true,
        "topic": "topic",
        "topicType": "msg",
        "format": "mmHg",
        "min": "60",
        "max": "120",
        "step": 1,
        "x": 550,
```

```
    "y": 320,
    "wires": [
        []
    ]
},
{
    "id": "ce656cf6f38ec209",
    "type": "ui_gauge",
    "z": "4dd134790857fcaa",
    "name": "",
    "group": "5495366eb67a0127",
    "order": 1,
    "width": "3",
    "height": "2",
    "gtype": "gage",
    "title": "Temperatura",
    "label": "ºC",
    "format": "{{value}}",
    "min": "35",
    "max": "40",
    "colors": [
        "#cf11e8",
        "#e600a8",
        "#ef0606"
    ],
    "seg1": "",
    "seg2": "",
    "x": 890,
    "y": 540,
    "wires": []
```

```
},
{
    "id": "00ef03623790a9d1",
    "type": "ui_gauge",
    "z": "4dd134790857fcaa",
    "name": "",
    "group": "5495366eb67a0127",
    "order": 2,
    "width": "3",
    "height": "2",
    "gtype": "gage",
    "title": "SPO2",
    "label": "%",
    "format": "{{value}}",
    "min": "80",
    "max": "100",
    "colors": [
        "#c5e811",
        "#00e6e2",
        "#3acb6c"
    ],
    "seg1": "",
    "seg2": "",
    "x": 770,
    "y": 460,
    "wires": []
},
{
    "id": "d5e76ad8fd97cc10",
    "type": "ui_numeric",
```

```
    "z": "4dd134790857fcaa",
    "name": "",
    "label": "Temperatura",
    "tooltip": "",
    "group": "7b57f931b680826c",
    "order": 6,
    "width": 0,
    "height": 0,
    "wrap": false,
    "passthru": true,
    "topic": "topic",
    "topicType": "msg",
    "format": "ºC",
    "min": "34",
    "max": "42",
    "step": 1,
    "x": 890,
    "y": 580,
    "wires": [
        []
    ]
},
{
    "id": "033db1cfcd151c66",
    "type": "ui_gauge",
    "z": "4dd134790857fcaa",
    "name": "",
    "group": "e1ed393d076e8195",
    "order": 3,
    "width": "0",
```

```
    "height": "0",
    "gtype": "gage",
    "title": "HR",
    "label": "BPM",
    "format": "{{value}}",
    "min": "50",
    "max": "200",
    "colors": [
      "#00b500",
      "#e6e600",
      "#ca3838"
    ],
    "seg1": "",
    "seg2": "",
    "x": 670,
    "y": 380,
    "wires": []
  },
  {
    "id": "db54a1cf79da0066",
    "type": "ui_numeric",
    "z": "4dd134790857fcaa",
    "name": "",
    "label": "HR",
    "tooltip": "",
    "group": "7b57f931b680826c",
    "order": 5,
    "width": 0,
    "height": 0,
    "wrap": false,
```

```
    "passthru": true,
    "topic": "topic",
    "topicType": "msg",
    "format": "BPM",
    "min": "50",
    "max": "200",
    "step": 1,
    "x": 880,
    "y": 400,
    "wires": [
        []
    ]
},
{
    "id": "7a6f38d5f21736f8",
    "type": "ui_numeric",
    "z": "4dd134790857fcaa",
    "name": "",
    "label": "SpO2",
    "tooltip": "",
    "group": "7b57f931b680826c",
    "order": 4,
    "width": 0,
    "height": 0,
    "wrap": false,
    "passthru": true,
    "topic": "topic",
    "topicType": "msg",
    "format": "%",
    "min": "90",
```

```
        "max": "100",
        "step": 1,
        "x": 940,
        "y": 500,
        "wires": [
            []
        ]
    },
    {
        "id": "b8af81ec9c247b7d",
        "type": "ui_chart",
        "z": "4dd134790857fcaa",
        "name": "",
        "group": "019379c595b99e65",
        "order": 6,
        "width": "0",
        "height": "0",
        "label": "HR",
        "chartType": "line",
        "legend": "false",
        "xformat": "HH:mm:ss",
        "interpolate": "linear",
        "nodata": "",
        "dot": false,
        "ymin": "50",
        "ymax": "200",
        "removeOlder": 1,
        "removeOlderPoints": "",
        "removeOlderUnit": "60",
        "cutout": 0,
```

```
    "useOneColor": false,
    "useUTC": false,
    "colors": [
        "#1f77b4",
        "#aec7e8",
        "#ff7f0e",
        "#2ca02c",
        "#98df8a",
        "#d62728",
        "#ff9896",
        "#9467bd",
        "#c5b0d5"
    ],
    "outputs": 1,
    "useDifferentColor": false,
    "x": 890,
    "y": 360,
    "wires": [
        []
    ]
},
{
    "id": "3cd8b4366a9554e3",
    "type": "mysql",
    "z": "4dd134790857fcaa",
    "mydb": "f3168f5ea887d8f2",
    "name": "",
    "x": 930,
    "y": 20,
    "wires": [
```

```
            []
        ]
    },
    {
        "id": "a4c5e35a37eb6490",
        "type": "ui_spacer",
        "z": "4dd134790857fcaa",
        "name": "spacer",
        "group": "31d732c6059e557a",
        "order": 3,
        "width": 1,
        "height": 1
    },
    {
        "id": "3d96af0795295548",
        "type": "mqtt-broker",
        "name": "",
        "broker": "192.168.0.32",
        "port": "1883",
        "clientid": "",
        "usetls": false,
        "protocolVersion": "4",
        "keepalive": "60",
        "cleansession": true,
        "birthTopic": "",
        "birthQos": "0",
        "birthPayload": "",
        "birthMsg": {},
        "closeTopic": "",
        "closeQos": "0",
```

```
    "closePayload": "",
    "closeMsg": {},
    "willTopic": "",
    "willQos": "0",
    "willPayload": "",
    "willMsg": {},
    "sessionExpiry": ""
},
{
    "id": "746a362c9103fa5c",
    "type": "ui_group",
    "name": "Control",
    "tab": "fe37047bce15b013",
    "order": 6,
    "disp": true,
    "width": "6",
    "collapse": false
},
{
    "id": "31d732c6059e557a",
    "type": "ui_group",
    "name": "SPO2",
    "tab": "fe37047bce15b013",
    "order": 5,
    "disp": true,
    "width": "6",
    "collapse": false
},
{
    "id": "e1ed393d076e8195",
```

"type": "ui_group",
"name": "Heart Rate",
"tab": "fe37047bce15b013",
"order": 2,
"disp": true,
"width": "6",
"collapse": false
},
{
"id": "af528003b9d19ab0",
"type": "ui_group",
"name": "Temperatura",
"tab": "fe37047bce15b013",
"order": 4,
"disp": true,
"width": "6",
"collapse": false
},
{
"id": "18bae98e7738c922",
"type": "ui_group",
"name": "Presión arterial",
"tab": "fe37047bce15b013",
"order": 3,
"disp": true,
"width": "6",
"collapse": false
},
{
"id": "7b57f931b680826c",

"type": "ui_group",
"name": "Números",
"tab": "b133b2c2de07ba71",
"order": 2,
"disp": true,
"width": "6",
"collapse": false
},
{
"id": "5495366eb67a0127",
"type": "ui_group",
"name": "Indicadores",
"tab": "b133b2c2de07ba71",
"order": 3,
"disp": true,
"width": "6",
"collapse": false
},
{
"id": "019379c595b99e65",
"type": "ui_group",
"name": "Gráficas",
"tab": "b133b2c2de07ba71",
"order": 1,
"disp": true,
"width": "6",
"collapse": false
},
{
"id": "f3168f5ea887d8f2",

```
        "type": "MySQLdatabase",
        "name": "",
        "host": "localhost",
        "port": "3306",
        "db": "basedatos",
        "tz": "",
        "charset": "UTF8"
    },
    {
        "id": "fe37047bce15b013",
        "type": "ui_tab",
        "name": "Home",
        "icon": "dashboard",
        "disabled": false,
        "hidden": false
    },
    {
        "id": "b133b2c2de07ba71",
        "type": "ui_tab",
        "name": "Signos",
        "icon": "dashboard",
        "order": 2,
        "disabled": false,
        "hidden": false
    }
]
```