PROYECTO FINAL DE INGENIERÍA

SISTEMA DE MONITOREO CONSTANTE MEDIANTE WIFI DE VALORES CARDIACOS ESENCIALES DE BAJO COSTO EN PACIENTES HOSPITALARIOS-AMBULATORIOS

Sabbadin, Federico – LU1093735
Ingeniería electromecánica
Eiras, Nicolás Angel – LU1086838
Ingeniería Telecomunicaciones

Tutor:
Carrazan, Luis Alejandro Silvestre, UADE

2021

UNIVERSIDAD ARGENTINA DE LA EMPRESA
FACULTAD DE INGENIERÍA Y CIENCIAS EXACTAS

1. Agradecimientos

Este proyecto tiene mucho significado para nosotros porque lo plantamos y desarrollamos como una muestra de todo lo aprendido en los años de carrera donde buscamos generar algo con nuestra impronta.

Esto no hubiese sido posible sin el apoyo incondicional de nuestras familias que siempre nos inculcaron el trabajo duro y la dedicación para las distintas etapas de nuestras vidas.

No queremos dejar de agradecer a nuestros amigos por sus colaboraciones técnicas y emocionales durante este recorrido que nos fueron guiando para obtener un desarrollo optimo.

También queremos agradecer a la médica clínica Julieta Arduin por su tiempo y aporte, quien fue nuestra guía durante todo el desarrollo para cumplir con ciertas necesidades médicas que hoy en día no estaban siendo abarcadas.

A nuestro tutor, Luis Carrazan, con quien estamos completamente agradecidos por todo el tiempo que nos brindó a la hora del desarrollo, también por sus ideas de mejora y experiencia propia, no solo técnica sino también organizacional.

2. Resumen

La necesidad de control y monitoreo permanente de los signos vitales de las personas surge como tema en creciente importancia en nuestro país y alrededor del mundo, y más debido a la pandemia de COVID-19 iniciada en el año 2020. En este momento el avance progresivo de las tecnologías y redes móviles habilita nuevas herramientas para dar seguimiento de estos sujetos. Es así como el uso de dispositivos móviles añade nuevas dimensiones como la movilidad y personalización, así como también el uso de IOT (de sus siglas en ingles de Internet of Things). Esto nos permitirá estar conectados constantemente, acceder a los datos del dispositivo rápidamente y desde cualquier parte del mundo.

El presente trabajo explica el diseño e implementación de una pulsera capas de sensar pulsaciones por minutos, saturación de oxígeno en sangre y temperatura de una persona y que, mediante una conexión WIFI, esta se comunique con una base de datos. La cual es leída por código WEB y es mostrada de forma organizada al usuario.

3. Abstract

Constant WIFI monitoring system of low-cost essential cardiac values in hospital-ambulatory patients

The requirement for permanent control and monitoring of people's vital signs emerges as an issue of increasing importance in our country and around the world, and more due to the COVID-19 pandemic that began in 2020. At this moment, the progress of mobile technologies and networks enable new tools to follow up on these subjects. That is why the use of mobile devices adds new dimensions such as mobility and personalization, as well as the use of IOT (Internet Of Things). This allow us to be constantly connected and access the device data quickly from anywhere in the world.

This work explains the design and implementation of a bracelet capable of sensing beats per minute, blood oxygen saturation and temperature of a person, that will be communicated to a database through WIFI. In the other side, this information will be read by a website where will be organized.

4. Índice

5. Introducción

A raíz de la pandemia de COVID-19 que genero grandes inconvenientes en la salud, vimos cómo había un gran sector de personas que no tenían acceso a equipos médicos debido al colapso por la demanda de uso, donde hubo casos que se tuvo que priorizar a algunos pacientes, dejando a otros sin equipos.

El desarrollo en este proyecto busco en todo momento la obtención de un aparato electrónico e inteligente que sea fácil de producir, con un valor altamente competitivo al mercado médico.

Esta pulsera es capaz de obtener valores básicos altamente necesarios para conocer la situación actual de un paciente sin la necesidad de estar presente de forma física en un lugar, pudiendo obtener la información desde cualquier lado que se disponga de una conexión a internet. El objetivo sería tener la información mostrada continuamente en un monitor que se encuentra en algún sector común de los encargados, además de poder ingresar mediante una conexión a internet con algún dispositivo, en el caso que se desee.

La persona cuenta con un botón en esta pulsera que le permitirá accionarlo en caso de una emergencia, donde el médico o encargado podrá conocer el inconveniente en cuestión de segundos y acudir a la misma.

La página donde se mostrará la información del usuario fue diseñada con amplios estándares de visualización para que sea amigable y sencilla al encargado. También, cuenta con estándares de seguridad de autenticación y divide la aplicación web en dos para que, en caso de conocer el código de ingreso, este no tenga conocimiento de lo demás usuarios.

Uno de los puntos más fuertes que buscamos preservar fue su costo, por eso optamos por la utilización de componentes de un precio accesible y servicios que sean altamente competitivos en relación precio-calidad.

En este informe se busca detallar inicialmente los conceptos utilizados para luego poder entrar en la parte técnica del desarrollo, donde será dividido en dos sectores muy fuertes, en el primero se encuentra la pulsera física, sus componentes y su formato de sensado, y por el otro lado, todo lo relacionado a la aplicación web, desde la página para el usuario de la pulsera hasta la página para el dueño del servicio. Como conexión a estos dos mundos, está el servicio denominado Firebase, que es provisto por Google. Este servicio nos aportó del hosteo de la página, las autenticaciones de usuarios y la base de datos para almacenar toda la

información. Por último, se realiza un análisis comercial, con un enfoque financiero y económico, donde utilizamos herramientas de comercialización y marketing para ubicar el producto con el máximo alcance y margen posible.

6. Antecedentes

6.1. Planteamiento del problema

Este proyecto final está inspirado en la necesidad de que, en los hospitales o centros de salud, el control y monitoreo de pacientes es solo para aquellos que se encuentran en terapia intensiva dejando sin control permanente a los internados en una sala común, que solo son revisados esporádicamente por enfermeros.

Al existir esta problemática, una solución para poder controlar y monitorear a estos pacientes de forma continua, que solo pueden generar alertas de malestar a través de un timbre, es la de colocarles una pulsera que contenga los respectivos sensores, un botón antipánico, una batería y un chip WIFI.

6.2. Justificación

En principio querer hacer un monitoreo constante en todos los pacientes de un hospital podría ser muy costoso, ya que habría que comprar equipamiento específico para tal motivo y pagar su instalación. Esto valdría la pena solo para pacientes que deben ser controlados minuciosamente y sin posibilidad de falla en los dispositivos.

Gracias a estas pulseras esto no sería así, ya que su fabricación y adquisición de todos los componentes es muy económica. Obviamente que estas pulseras no reemplazarían los equipos instalados en las terapias intensivas por lo dicho anteriormente. Pero si, pudieran usarse perfectamente en terapias intermedias, internados en habitaciones comunes, pacientes ambulatorios, etc. Los mismos no requieren la misma precisión que un paciente en estado crítico.

6.3. Objetivos

El objetivo general del proyecto es el desarrollo completo de una pulsera WIFI capas de sensar las pulsaciones por minuto, oxígeno en sangre, y temperatura de una persona, junto con un botón para emergencias. Recopilar toda esta información, subirla a la nube y poder visualizarla desde cualquier dispositivo con acceso a internet. A su vez esta interfaz será capaz de mostrar alertas según el estado de cada individuo y según el estado actual comparado con los límites de riesgo que se hayan configurado.

De esta forma el hospital o personal médico podrá controlar constantemente el estado de todas aquellas personas que dispongan del dispositivo, para así anticiparse o reducir los tiempos de atención en el caso de que, por ejemplo, el paciente entre en paro cardiaco.

Para este trabajo se generará tanto el diseño electrónico como visual y estético de la pulsera, la programación y conexión a internet de esta. Y a su vez su emparejamiento con la interfaz donde se podrán visualizar los datos y el diseño de la misma.

6.4. Metodología

Desarrollaremos una pulsera WIFI (capas de sensar pulsaciones por minuto, oxígeno en sangre de una persona y temperatura corporal) y una página web (donde se podrán visualizar todos los datos enviados por la pulsera). Para ello investigaremos cuales son las mejores alternativas a cuento componentes electrónicos a usar, forma final de la pulsera, metodología de sensado, base de datos, hosteo de página web, etc.

Una vez hecho esto se hará un prototipo de la pulsera y se creará una página web 100% funcional.

7. Marco teórico

7.1. IoT - Internet Of Things

La definición de IoT podría ser la agrupación e interconexión de dispositivos y objetos a través de una red (bien sea privada o Internet), dónde todos ellos podrían ser visibles e interaccionar. Respecto al tipo de objetos o dispositivos podrían ser cualquiera, desde sensores y dispositivos mecánicos hasta objetos cotidianos como pueden ser la heladera, el calzado o la ropa. Cualquier cosa que se pueda imaginar podría ser conectada a internet e interaccionar sin necesidad de la intervención humana, el objetivo por tanto es una interacción de máquina a máquina, o lo que se conoce como una interacción M2M (Machine To Machine) o dispositivos M2M (María Gracia, Deloitte, 2021).

Este proceso de desarrollo IoT está en continua evolución, y es algo bastante nuevo, por lo que hay muchos protocolos de comunicación entre dispositivos. Uno que se está tratando de estandarizar es el propuesto por IBM, denominado MQTT (Message Queuing Telemetry Transport), este protocolo es de tipo abierto, de esta forma permite la participación de muchos fabricantes y facilita la comunicación entre dispositivos.

Lo que siempre se busca en los dispositivos IoT es un tamaño reducido y un bajo consumo, es por eso que los fabricantes de microcontroladores cumplen un rol muy importante, tratando de fabricar SoCs (System on Chip) lo más completos posibles, que no solo procesen señales y datos, sino que también contenga integrados de conexión, como wifi o redes GSM. Algunos fabricantes son Intel, MediaTek, Samsung, o alguna alternativa más asequible son los producidos por Arduino.

Una parte muy importante en IoT es la tecnología usada para la comunicación entre dispositivos. La más común es a través de una red wifi, aunque no es la mejor, ya que tiene grandes consumos y bajo alcance. Otro método de comunicación mejor es a través de redes móviles como 3G o 4G.

Como se ha comentado el termino IoT es muy extenso, con infinitas aplicaciones. Y no es algo nuevo, si no que se le busco ponerle un nombre a aquellos dispositivos con un algún grado de inteligencia que se conectan entre sí. La idea de estos dispositivos es que estén conectados a internet, en lo posible de forma directa, y que sean capaces de procesar y almacenar información.

7.2. Microcontroladores

Un microcontrolador (mCU) es un circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria. Está compuesto de varios bloques como un microprocesador, convertidores analógicos a digital, memorias RAM y ROM, oscilador, entre otros. Cada uno de estos cumplen una función en específico. Otra parte fundamental de los MCU son las entradas y salidas físicas, estas pueden ser de alimentación, teniendo señales analógicas o digitales a la entrada como a la salida. Esta es la forma de comunicar nuestros microcontroladores con los periféricos, como por ejemplo sensores.

La variedad de MCU es muy amplia, va desde algunos muy simples, con velocidades muy bajas de procesamiento y con muy pocas entradas y salidas, hasta algunos muy potentes, con grandes consumos energéticos, muchas entradas y salidas, capaces de procesar hasta varias entradas y salidas analógicas; estos últimos reciben el nombre de DSP (Procesador digital de señales).

Cuando el microcontrolador sale de fábrica, no contiene datos en la memoria ROM. Para que pueda cumplir alguna función es necesario generar o crear y luego grabar en la EEPROM, o equivalente del microcontrolador, algún programa que puede ser escrito en lenguaje ensamblador u otro lenguaje de alto nivel para microcontroladores; sin embargo, para que el programa pueda ser grabado en la memoria del microcontrolador, debe ser compilado en el sistema numérico hexadecimal que es el lenguaje que entiende el MCU.

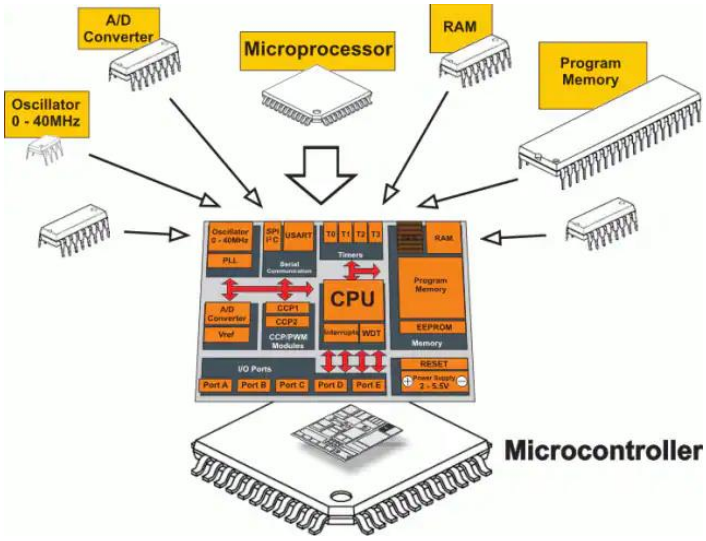


Ilustración 1 : Partes de un microcontrolador
Fuente: https://aprendiendoarduino.wordpress.com/2016/06/26/microcontroladores-2/

7.2.1. Compilación

Todo código escrito en un lenguaje de alto nivel (como java, C, C++, etc.) debe ser transformado o traducido en código máquina o hexadecimal. Los programas que escribimos los entendemos nosotros, pero no así el microcontrolador.

Un software de computadora, llamado compilador, traduce y transforma nuestro programa en código máquina, que es lo que realmente puede leer e interpretar el microcontrolador.

Una vez compilado en código maquina el programa, es momento de conectar el microcontrolador y transferir el código hacia la memoria interna del mismo, usualmente hacia la ROM. Para esta tarea se utiliza un programador físico o circuitos integrados en el dispositivo, que cumplen la función de intermediarios entre el ordenador y la memoria interna del hardware.

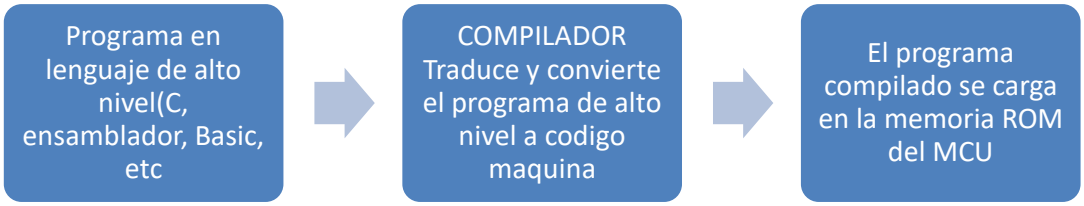


Ilustración 2: Orden de compilación

7.2.2. Historia microcontroladores

En 1971, se lanzó el primer procesador pequeño llamado Intel 4004 de 4 bits, donde lo siguió el Intel 8008. Ambos procesadores necesitan de circuitos extras que permitan implementar el sistema de trabajo, generando un costo del sistema mayor.

Desde el Instituto Smithsoniano se cree que los Ingenieros de la firma Texas Instruments, Gary Boone y Michael Cochran, crearon el primero microcontrolador denominado TMS 1000, en 1971, comercializado en 1974. Este combinaba memoria RAM, ROM, microprocesador y reloj en un chip, destinada a sistemas embebidos.

Gracias a la existencia del TMS 1000, una empresa llamada Intel, desarrollo un sistema de ordenador en un chip optimizado utilizado en aplicaciones de control, conocido como Intel 8048, comercializado en 1977. Este chip combina la memoria RAM y ROM en el mismo dispositivo. Tuvo una gran utilización en teclado IBM PC, que se comercializaron más de mil millones. Luke J. Valenter, presidente de Intel de ese momento, consideraba que el

microcontrolador era uno de los productos más trascendentes y exitosos en la historia de la compañía, dándolo un margen mayor de inversiones.

Los microcontroladores en estos momentos eran regidos por dos variantes. Los que tenían una memoria EPROM podían ser reprogramados, teniendo un valor más caro a los que una vez programados no variaban, llamados PROM. El borrado de la EPROM era mediante la exposición de la tapa de cuarzo transparente a luz ultravioleta. Un detalle muy importante, era que el valor de los chips que eran todo opaco tenían un coste menor.

Cuando se lanzó el EEPROM en los microcontroladores, en 1993, permitió avanzar grandes rasgos. Este permitía borrar esta memoria de forma eléctrica rápidamente sin la necesidad de la utilización de un paquete externo como se necesita en EPROM, permitiendo crear prototipos rápidos y programarlos en el sistema. En ese año se lanza el primer microcontrolador, por la empresa Atmel, que utiliza una memoria flash. Rápidamente otras compañías optaron por seguir este camino de los dos tipos de memoria disponible.

Pasado el tiempo su costo se redujo hasta valores ínfimos, donde un microcontrolador de 8 bits está disponible por menos de 25 centavos de dólar y los mismos de 32 bits se pueden encontrar a 1 dólar. Estos valores permiten un gran sector de aficionados que crece día a día gracias a las comunidades encontradas por internet.

Las MRAM puede ser un gran punto para considerar en los microcontroladores porque tiene una resistencia infinita y el valor del material semiconductor es muy bajo.

7.3. Arduino IDE

El software Arduino IDE es una aplicación multiplataforma (para Windows, macOS, Linux) que está escrita en el lenguaje de programación Java. Es un editor de código desarrollado por Arduino en el cual vamos a poder escribir nuestro código en lenguaje C y C++ con algunas reglas especiales del mismo software, y este se encargará de compilar y cargar nuestro programa a las diferentes placas, tanto propias como de terceros.

Arduino IDE es muy fácil de usar para los principiantes, pero suficiente flexible para aquellos usuarios avanzados. Es una herramienta de código abierto con mucha variedad de librerías tanto propias como de terceros, accesibles directamente desde el software, siendo esto de gran utilidad.

El código está dividido en dos grandes funciones básicas, setup(), donde se inicializan las variables y todas las librerías, y loop() donde se va a repetir consecutivamente todo lo que escribamos en ella.

El IDE de Arduino emplea el programa avrdude para convertir el código ejecutable en un archivo de texto en codificación hexadecimal que es el que finalmente se carga en la placa Arduino.

7.4. Actualizaciones vía OTA

OTA, hace referencia a Over The Air, es decir, 'sobre el aire'. Este concepto, en informática, se refiere a un método de actualización de un nuevo software.

Los antiguos dispositivos salían de fabrica con un software o firmware precargado y no era posible modificarlo. Al pasar los años esto varió, algunos aparatos eran capaces de actualizarse a través de un ordenador mediante la utilización de una conexión alámbrica, es decir, por cable. Pero con la llegada de los primeros teléfonos inteligentes esto paso a ser a través de las redes, llegando así las actualizaciones OTA u Over The Air, de manera inalámbrica y aprovechando la conexión de datos móviles o, sobre todo, las conexiones a Internet por WiFi.

La tecnología OTA requiere que sean compatibles con esta función tanto el software como el hardware. Porque no solo se recibe el software en cuestión, correspondiente a una actualización, sino que también se instala aprovechando la red inalámbrica del proveedor. Todo este proceso se inicia con una consulta al sistema por parte del proveedor, o de otros servidores, para que se lleve a cabo de forma automática.

Las actualizaciones de software por OTA, evidentemente, es una de las grandes evoluciones de los dispositivos inteligentes. Porque es otro aspecto más que ha dotado a los dispositivos IOT de independencia completa del ordenador. Antes, cuando era necesaria la conexión por cable, las actualizaciones no eran tan sencillas de aplicar, ni se podían introducir el dispositivo en cualquier momento.

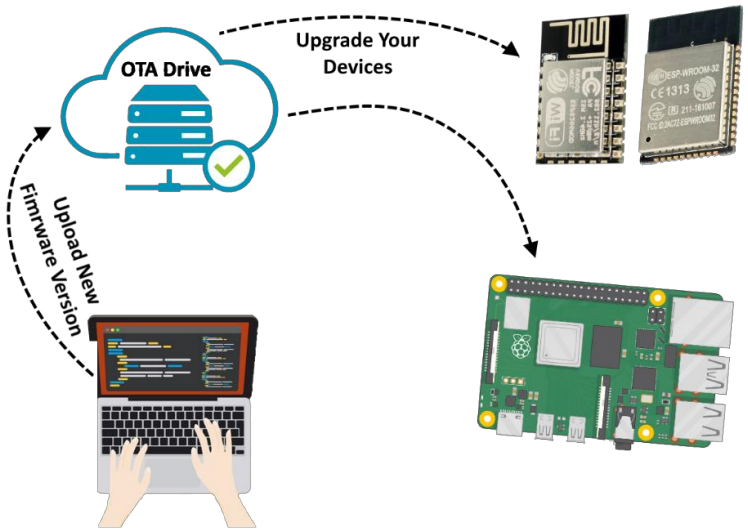


Ilustración 3: Transferencia vía OTA
Fuente: https://www.otadrive.com/

7.5. Firebase

A la hora de desarrollar aplicaciones web o móviles se puede utilizar la plataforma denominada Firebase que fue lanzada en 2011 y, luego de 3 años, comprada por Google.

La ubicación de esta plataforma es en la nube, gracias al uso de las redes que dispone actualmente. Tiene integración con Google Cloud Plataform, que usa herramientas para poder generar y sincronizar proyectos de alta calidad, permitiendo la escalabilidad de usuarios y así generar el máximo beneficio posible.

7.5.1. Servicios

A continuación, vamos a ver los servicios de Firebase utilizados en este proyecto, que no son todos los que ofrece Google a través de esta plataforma.

7.5.1.1. Realtime database

Este servicio en tiempo real es uno de los más importantes y esenciales de Firebase. Están alojadas en la nube y son NoSQL, donde los datos son almacenados en formato de JSON. La información alojada es guardada en tiempo real y se mantiene actualizada durante el transcurso del tiempo sin que el usuario modifique alguno.

En el caso de que los datos cambien, Firebase envía automáticamente eventos a las aplicaciones y los guarda en su disco. Uno de los puntos fuerte de este servicio, es su obtención de datos sin que hubiera conexión por parte de un usuario, es decir que su

información y los cambios serian sincronizados una vez que el usuario sea capaz de reestablecer la conexión.

7.5.1.2. Autenticación de usuarios

Las aplicaciones disponen de información que no se desea mostrar para todos los usuarios, por eso se nace este servicio. Donde se conocerá mediante un autenticado de persona, quien es el sujeto que dice que es.

Esta plataforma ofrece un servicio que permite mediante un mail y contraseña, autenticarse. Se puede utilizar plataformas externas como Facebook, Twitter o Google para generar el registro en sí, siendo así más fácil a la hora de generar usuarios.

Así, este tipo de tareas se ven simplificadas, considerando también que desde aquí se gestionan los accesos y se consigue una mayor seguridad y protección de los datos. Un aspecto muy importante es que Firebase almacena los datos de inicio de sesión en la nube con todos los estándares de seguridad necesario, haciendo que cualquier persona tenga que autenticarse para ser capaz de ingresar a la sesión.

7.5.1.3. Almacenamiento en la nube

El almacenamiento de Firebase es gracias a la utilización de su nube, donde los creadores de la aplicación pueden guardar la información de esta, que se vinculan con referencias a un árbol de ficheros para mejorar el rendimiento. Su uso es adaptado a cada caso, donde se deberá corregir las reglas para la debida situación.

Para los usuarios de la aplicación es de gran utilidad porque permite cargar y acceder datos más rápido y fácil que en otros servicios. La descarga de esta información está pensada para ser rápida y segura.

7.5.1.4. Hosting

Firebase también ofrece un servidor para alojar las aplicaciones de manera rápida y sencilla, esto es, un hosting estático y seguro. Proporciona certificados de seguridad SSL y HTTP2 de forma automática y gratuita para cada dominio, reafirmando la seguridad en la navegación. Funciona situándolas en el CDN (Content Delivery Network) de Firebase, una red que recibe los archivos subidos y permite entregar el contenido.

7.6. HTML

También conocido en inglés como HyperText Markup Language, es la herramienta inicial a la hora de desarrollar un sitio web. Se suele utilizar en conjunto con herramientas como CSS, para darle apariencia a esta, y JavaScript, para manipular su funcionalidad y comportamiento.

Este lenguaje no es de programación, sino que es un lenguaje marcado capaz de definir la estructura del contenido. Consiste en disponer elementos que se usaran para ubicar el contenido, que será mostrado de una determinada forma.

Su forma de organización es mediante el uso de etiquetas, que se usaran para insertar desde imágenes, vínculos a otros sitios, tablas o textos. Se podrá asignar el tamaño del texto como también el formato de esta.

A la hora de desarrollar la disposición se divide en dos elementos clave, primero el encabezado o también llamado head, este elemento contiene todo lo que se necesite incluir en la página HTML que no es visible para el usuario, como la descripción necesaria para aparecer en buscadores web o código en formato CSS para darle aspectos al mismo. Por otro lado, está la etiqueta body, encerrando todo el contenido que se muestra por pantalla a las personas, como ser imágenes, videos, sonidos, textos y demás.

Al pasar de los años surgieron necesidades a la hora de desarrollar páginas, es por esto que nacieron distintos estándares con etiquetas necesarias para el lenguaje. Hoy en día, la más utilizada es HTML5, donde permite adoptar elementos de navegación más sencillo y hace referencias más fáciles que las versiones más antiguas.

7.7. CSS

El lenguaje Cascading Styles Sheets (Hojas de Estilo en Cascada), es el utilizado para dar estilos que diagraman la presentación de documentos HTML o XML. Se encarga de describir cada elemento a la hora de renderizarlo por pantalla, papel u otros medios.

Al igual que HTML, no es conocido por ser un lenguaje de programación, sino que es un lenguaje de hojas de estilo. Permite aplicar estilos y formatos de forma selectiva a elementos del documento HTML.

Para su uso es necesario declarar el archivo CSS en el documento inicial de HTML, para que sea capaz de conocer los formatos y diseños que deberá de mostrar por pantalla. En este archivo de diseño se seleccionan los elementos a los cuales se le modifica su aspecto.

Tiene una estandarización especifica, donde han nacido versiones denominas CSS1, CSS2 y CSS3. A partir de la versión 3, gracias al mejorado del desarrollo de los módulos, se hizo más efectivo y masivo su uso.

7.8. JavaScript

JavaScript es el lenguaje de programación encargado de dotar de mayor interactividad y dinamismo a las páginas web. Cuando se ejecuta en el navegador, no necesita de un compilador. El navegador lee directamente el código, sin necesidad de terceros. Por lo tanto, se le reconoce como uno de los tres lenguajes nativos de la web junto a HTML (contenido y su estructura) y a CSS (diseño del contenido y su estructura).

Con este lenguaje de programación del lado del cliente (no en el servidor) podemos crear efectos y animaciones sin ninguna interacción, o respondiendo a eventos causados por el propio usuario tales como botones pulsados y modificaciones del DOM (Document Object Model). Nada tiene que ver con el lenguaje de programación Java, ya que su principal función es ayudar a crear páginas webs dinámicas.

El código de programación de JavaScript se ejecuta en los navegadores, ya sean de escritorio o móviles, como Android o IPhone. Sirve para exactamente lo mismo, da igual en el tipo de dispositivo que se ejecute el navegador.

JavaScript es capaz de detectar errores en formularios, de crear sliders que se adapten a cualquier pantalla, de hacer cálculos matemáticos de forma eficiente, de modificar elementos de una página web de forma sencilla, etc. Pero también JS es el encargado de que existan herramientas como Google Analytics, Google Tag Manager, Facebook Pixel y tantas otras, que son claros ejemplos de este.

Existe una tecnología llamada AJAX que permite intercambiar información con el servidor sin tener que recargar la página. Es decir, sólo cargamos de la página lo necesario. Esta tecnología desarrollada en JavaScript ha supuesto uno de los principales avances en el desarrollo web. Aunque no la sepamos reconocer, es la encargada de que podamos conseguir más mensajes, tweets, emails…sólo pulsando un botón, sin tener que recargar la página.

7.9. Visual Studio Code

A partir del desarrollo de un editor de código fuente para Windows, Linux y macOS, por Microsoft, es que nace Visual Studio Code. Este editor tiene soporte de depuración, integración con GIT, resaltado de sintaxis y otras herramientas que facilitan el desarrollo.

Al ser compatible con múltiples lenguajes de programación y su amplia gama de características, muchos desarrolladores optan por usarlo. Su uso va desde lenguajes como HTML, CSS, JavaScript hasta C, C++, Python, etc.

Una de las características más importantes a la hora de desarrollar webs, es su facilidad para mantener distintos archivos abiertos en segundo plano, y así tener una interacción de documentos mucho más dinámica.

7.10. Impresión 3D

Una impresora 3D es una máquina capaz de realizar objetos 3D, diseñados previamente en algún software de un ordenador. Comúnmente se utiliza en la fabricación de piezas o componentes, en sectores como la arquitectura y el diseño industrial. Es muy útil a la hora de crear prototipos o muestras de piezas que serán producidas en serie. En la actualidad, debido a su fácil acceso, variedad de materiales, gran calidad y terminación, se está extendiendo su uso en la fabricación de todo tipo de objetos, moldes, piezas complicadas, alimentos, prótesis médicas y odontológicas. Esto último es gracias a los scanner 3D, capaz de generar un modelo 3D de por ejemplo nuestra muñeca y hacer un producto personalizado que se adapte perfectamente a nuestro cuerpo.

Las impresoras con mayor presencia en el mercado son aquellas que inyectan polímeros (filamentos de PLA, ABS, PETG). Estas funden el plástico construyendo con él capas muy finas sobrepuestas para crear el objeto. Al ser de muy bajo costo y fáciles de utilizar muchas personas cuentan con una de ellas en su casa.

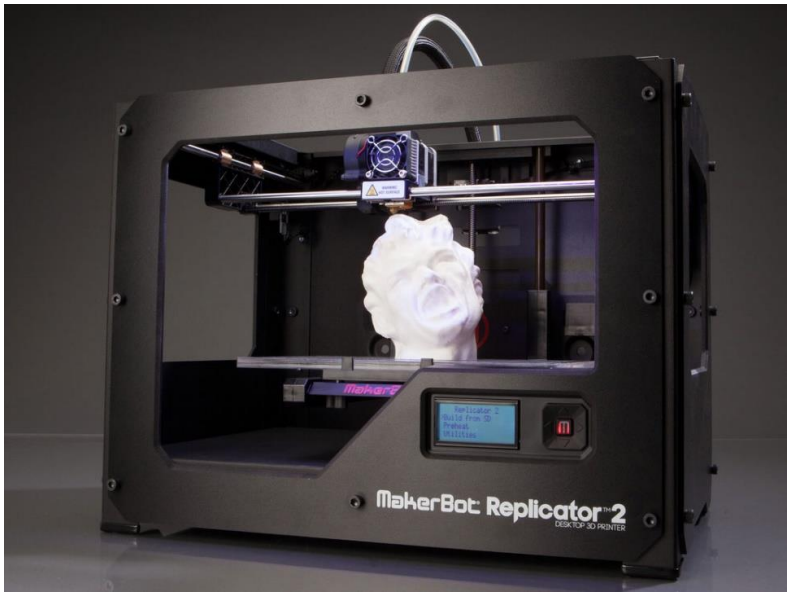


Ilustración 4: Impresora 3D por FDM
Fuente: https://www.enter.co/especiales/hogar-digital/hogar-del-futuro-con-impresoras-3d/

8. Descripción

Este proyecto está dividido en dos partes, por un lado, está la pulsera, que es la encargada del sensado de las pulsaciones por minutos, el oxígeno en sangre y la temperatura del paciente, y a su vez sube todos estos valores a la base de datos a través de una conexión WiFi. Por otra parte, está la página web, donde serán procesados y mostrados todos los datos administrados por la pulsera. Esta se encarga de leer la base de datos, y según quien inicie sesión es que datos finalmente muestra y analiza.

El recorrido de la información inicia en lo tomado por la pulsera, esta lo carga en la base de datos de Firebase para luego ser mostrada en la página web, que está hosteada en el mismo servicio (ver Ilustración 5).

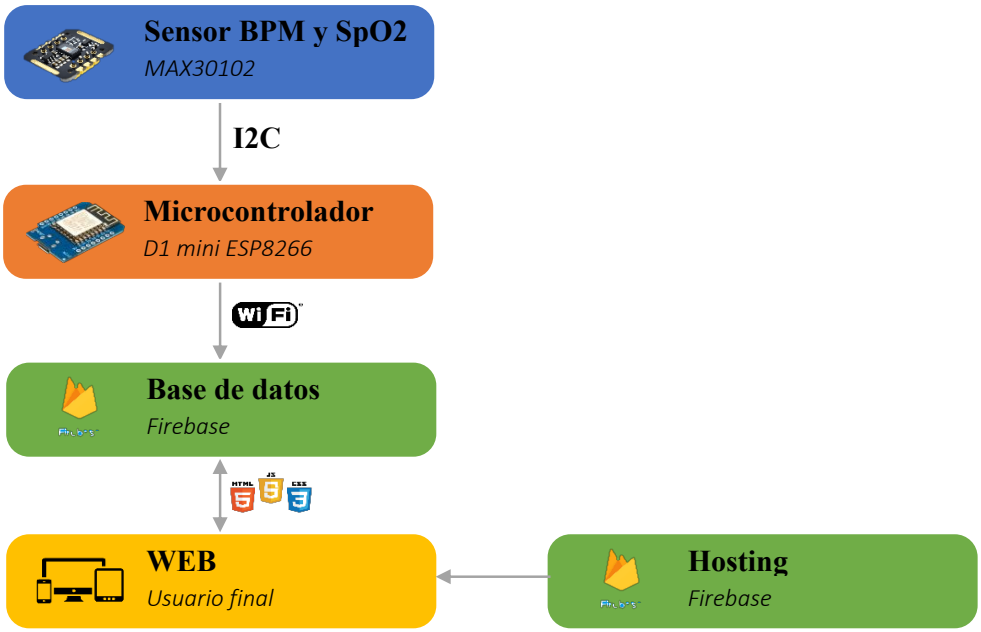


Ilustración 5: Comportamiento de la pulsera y aplicación web
Fuente: elaboración propia

8.1. Pulsera

Primero vamos a hablar de la pulsera, la cual posee un sensor que es el encargado de medir todos los parámetros, pulsaciones por minutos, oxígeno en sangre y temperatura de la persona; posee un microcontrolador, el cual es el intermediario entre la base de datos y los valores que le entrega el sensor, este se encarga de procesar la señal otorgada por el MAX30102, procesarla y subir estos valores a la base de datos. La pulsera también cuenta con

una batería, y su respectivo cargador, botones, resistencias y un capacitor, del cual vamos a hablar posteriormente.

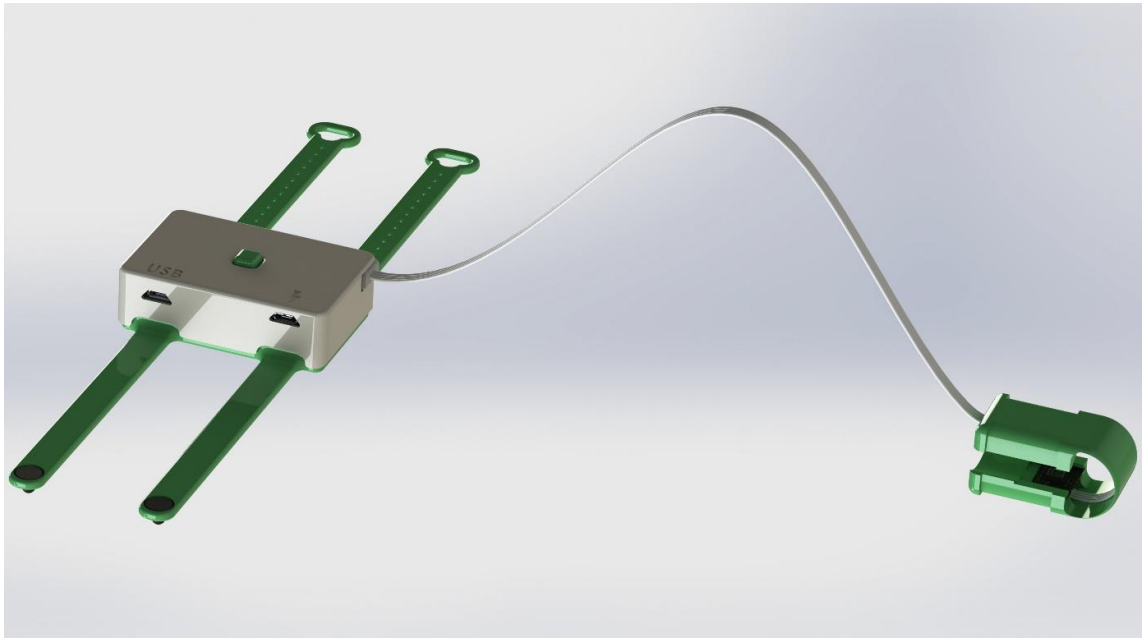


Ilustración 6: Modelado digital de pulsera

8.1.1. Componentes electrónicos

Al momento de pensar en una pulsera WiFi capas sensar oxígeno en sangre, pulsaciones por minutos y temperatura corporal es necesario contemplar cuales son los componentes que van a cumplir con todos nuestros objetivos, es decir, poder mantener un tamaño reducido y un bajo costo de producción. A pesar de analizar cuáles eran aquellos mejores componentes que nos ayudarían a cumplir con el objetivo no fue necesario para llegar a nuestro producto final, si no que tuvimos que probar nosotros mismos estos componentes y finalmente llegar a este conjunto final que conformaran nuestra pulsera.

8.1.2. Sensor de oxígeno en sangre y pulsaciones por minutos

El sensor utilizado es el MAX30102 debido a su fácil adquisición, fiabilidad y su tamaño reducido.

Ilustración 7: Frente sensor MAX30102
Fuente: https://store.fut-electronics.com/products/pulse-oximeter-spo2-heart-rate-sensor-max30100

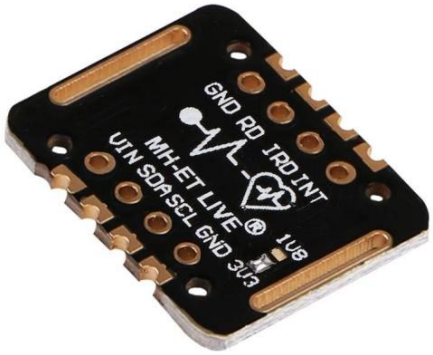


Ilustración 8: Dorso sensor MAX30102
Fuente: https://store.fut-electronics.com/products/pulse-oximeter-spo2-heart-rate-sensor-max30100

La pulsioximetría es un método no invasivo, que permite medir el porcentaje de saturación de oxígeno de la hemoglobina (SaO2) en sangre de un paciente utilizando un

circuito fotoeléctrico (sensores sensibles a la luz). Para esto se usa un pulsioxímetro, que es un dispositivo que integra leds emisores de luz y el sensor que mide la cantidad de luz reflejada por el dedo del paciente. La luz detectada por el sensor varía de acuerdo con las pulsaciones y también de acuerdo a la concentración de oxígeno en la sangre, la sangre oxigenada absorbe mayor cantidad de luz infrarroja, mientras que la sangre poco oxigenada absorbe mayor luz roja.

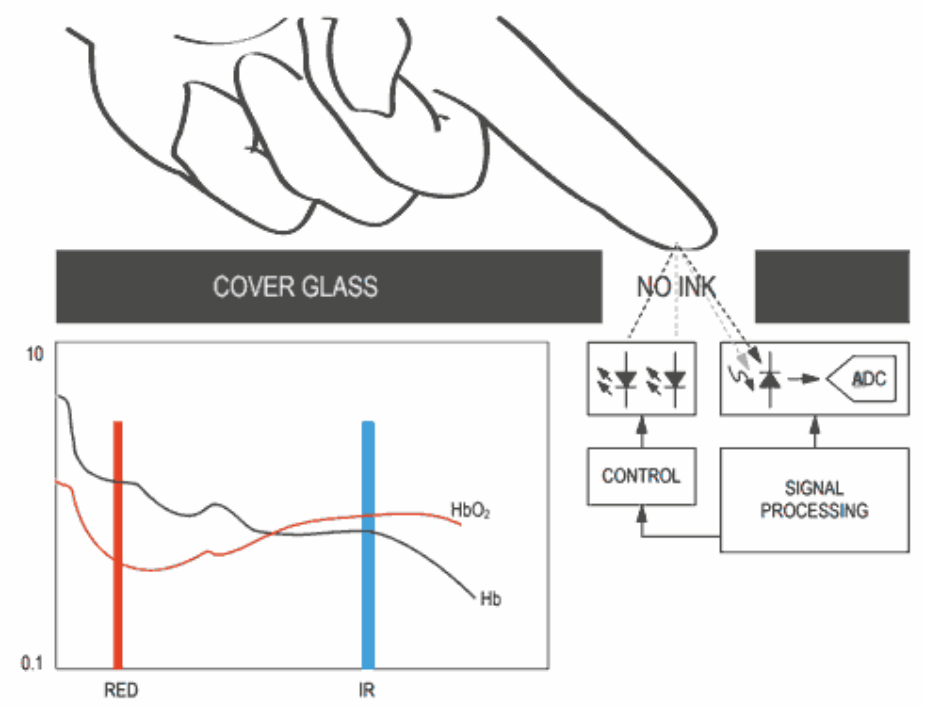


Ilustración 9: Comportamiento del sensor MAX30102
Fuente: https://www.luisllamas.es/pulsimetro-y-oximetro-con-arduino-y-max30102/

El MAX30102 es un dispositivo que integra un pulsioxímetro y un monitor de frecuencia cardiaca, es la evolución del sensor MAX30100 fabricado por Maxim Integrated. Posee dos Leds: un led rojo (660nm) y un led infrarrojo (920nm), un fotorreceptor, óptica especializada, filtro de luz ambiental entre 50 y 60Hz, y un conversor ADC delta sigma de 16 bits y da hasta 1000 muestras por segundo. Además, posee un sensor de temperatura interno para compensar los efectos de la temperatura en la medición.

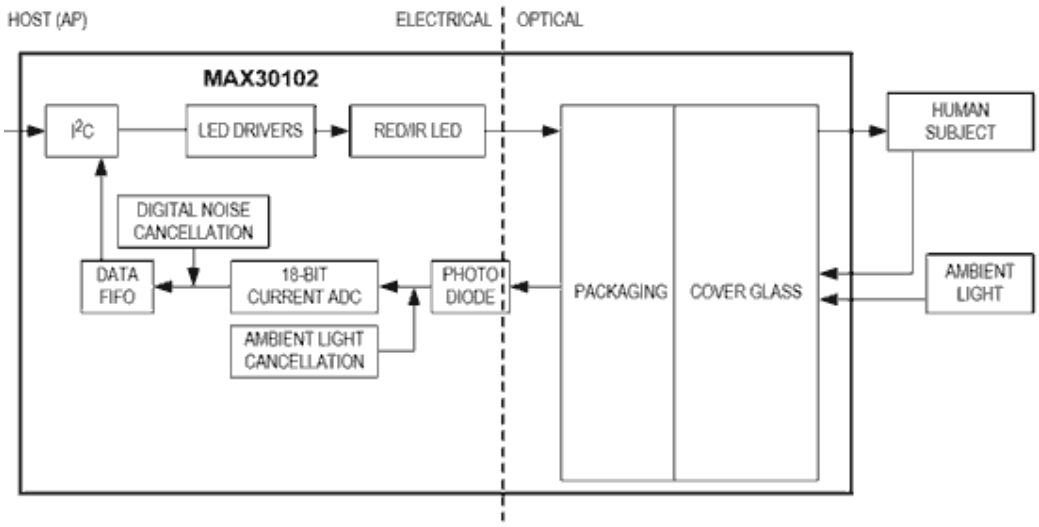


Ilustración 10: Diagrama de bloques del sensor MAX30102
Fuente: https://www.luisllamas.es/pulsimetro-y-oximetro-con-arduino-y-max30102/

El MAX30102 necesita de dos voltajes para funcionar: 1.8V para alimentar el controlador y un voltaje entre 3.3v y 5v para los leds rojo e infrarrojo. Este módulo incluye ambos reguladores de voltaje en placa por lo que solo se necesita una fuente de 5v o 3.3Vv para la alimentación. Su consumo de energía es mínimo, por lo que es ideal para aplicaciones portátiles. Puede ser utilizado en equipos de monitoreo médico, asistentes de estado físico y smartwatch en general.

8.1.2.1. Especificaciones Técnicas

Voltaje de Operación: 5V DC
Regulador de voltaje de 3.3V y 1.8V en placa
Protocolo de comunicación: I2C (compatible con 5v y 3.3v)
Sensor: MAX30102 (Maxim Integrated)
Led rojo de 660nm
Led infrarrojo de 880nm
Filtro de luz entre 50 y 60Hz
ADC delta sigma de hasta 16 bits
Temperatura de trabajo: -40°C hasta +85°C
Dimensiones: 14mm x 17mm

8.1.2.2. Conexión

VIN: 5V DC
GND: 0V
SCL: I2C CLOCK
SDA: I2C DATA

Un oxímetro de pulso es básicamente un dispositivo que puede medir el pulso y la saturación de oxígeno en la sangre. Por lo general, este sensor consta de dos LED que emiten luz: uno en espectro rojo (650nm) y el otro en infrarrojo (950nm). Este sensor se coloca comúnmente en el dedo o en cualquier lugar donde la piel no sea demasiado gruesa para que ambas frecuencias de luz puedan penetrar fácilmente en la piel (otros ejemplos de lugares donde puede colocarse son en el lóbulo de la oreja, muñeca, o en los bebes se suele colocar en el pie). Una vez que ambos leds brillan a través de su dedo, por ejemplo, la absorción se mide con un fotorreceptor. Y dependiendo de la cantidad de oxígeno que tenga la sangre, la proporción entre la luz roja absorbida y el led IR será diferente. A partir de esta proporción es posible calcular "fácilmente" el nivel de oxígeno en la sangre.

El MAX30102 ya posee varias funciones incorporadas, para poder obtener mediciones correctas los dos leds deben prenderse y apagarse alternativamente, esto ya está automatizado por el sensor, al igual que el regulador de brillo de los leds a través de un PWM, además posee un filtro de ruido de 50 y 60 Hz entre otras funciones. El resto se hace a través de una librería de Arduino llamada “SparkFun MAX3010x Pulse and Proximity Sensor Library”.

Desde la librería podremos controlar el ancho de pulso de encendido y apagado de cada led, la frecuencia de muestreo, el brillo del led, cuantos datos tener en cuenta para el promedio, entre otros valores. Al momento de sensar, la librería es la encargada de generar un filtro de corriente continua, y de esta forma se pueden contabilizar los cruces por cero para calcular la frecuencia cardiaca. No es el único filtro que aplica, si no que aplica varios más para poder generar una señal sin tanto ruido.

Ilustración 11: Lectura del sensor MAX30102 en el tiempo
Fuente: https://morf.lv/implementing-pulse-oximeter-using-max30100

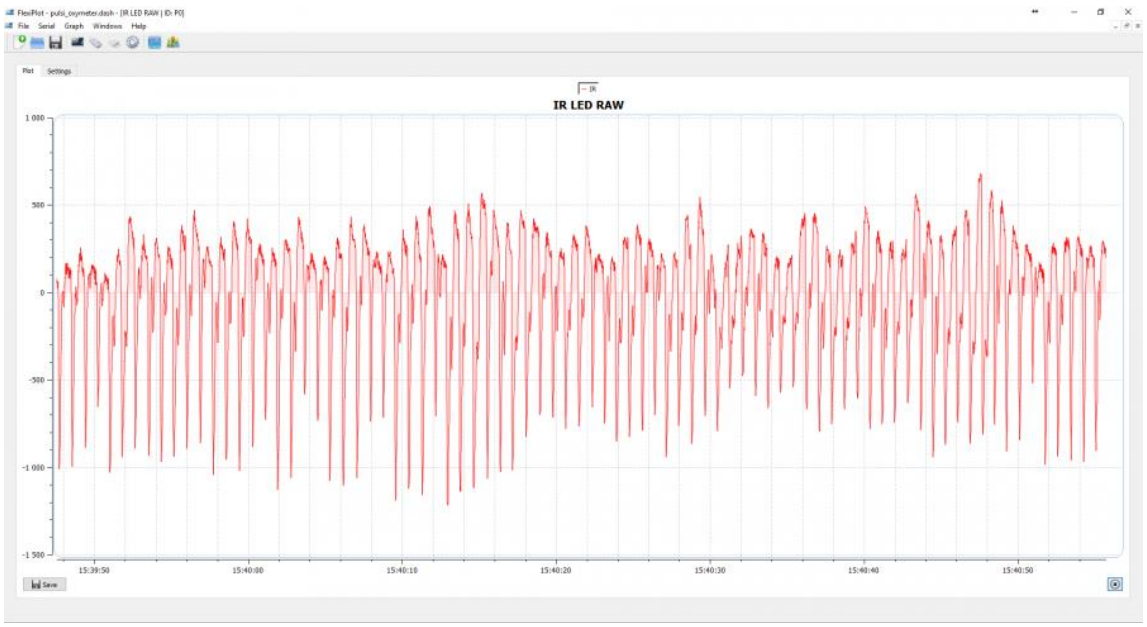


Ilustración 12: Lectura con filtro DC del sensor MAX30102 en el tiempo
Fuente: https://morf.lv/implementing-pulse-oximeter-using-max30100

Ilustración 13:Lectura filtrada del sensor MAX30102 en el tiempo
Fuente: https://morf.lv/implementing-pulse-oximeter-using-max30100

Y para calcular el oxígeno en sangre, que se mide calculando la relación entre la luz absorbida del LED infrarrojo y el LED rojo. El sensor se encarga de apagar y prender estos leds alternativamente y dependiendo de la luz absorbida de cada led es que calcula la oxigenación de la sangre.

8.1.3. Microcontrolador

El microcontrolador utilizado es el Wemos D1 Mini que posee el chip ESP8266, el cual es el encargado de conectarse con el sensor MAX30102, procesar todo lo sensado, establecer una conexión WIFI y enviar estos valores a la base de datos.

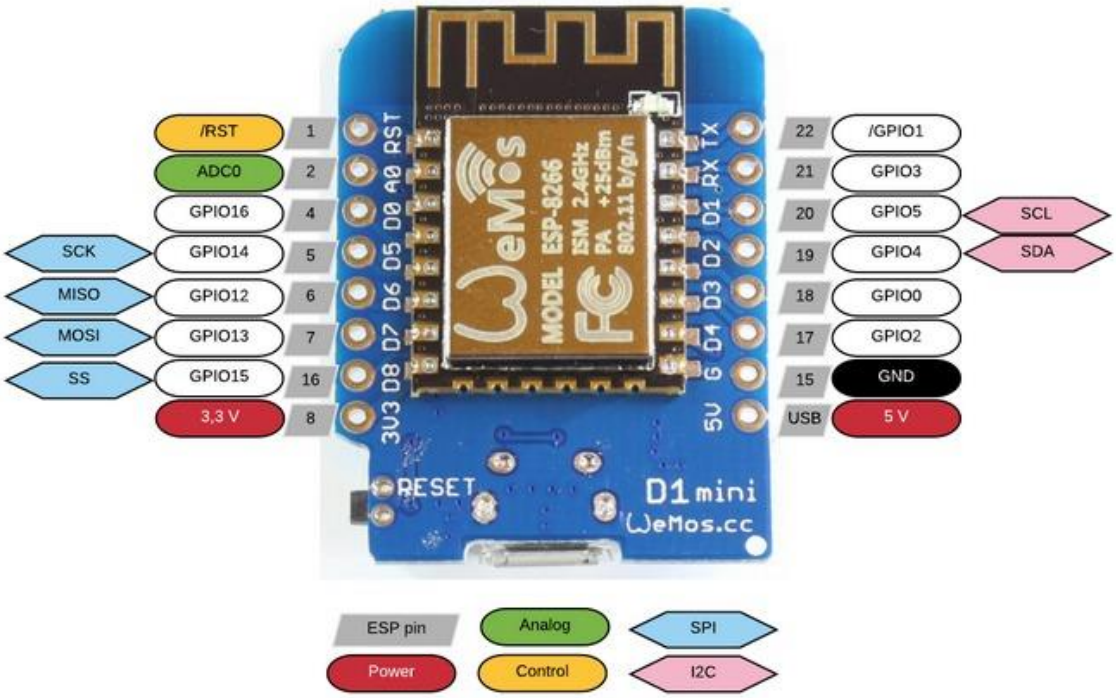


Ilustración 14: Wemos D1 mini ESP8266 conexiones
Fuente: https://www.researchgate.net/figure/Figura-2-Datasheet-WeMos-D1-Mini_fig1_332353123

Wemos D1 mini ESP8266 (ver Ilustración 14) es una plataforma de desarrollo similar a Arduino especialmente orientada al Internet de las cosas (IoT). La placa Wemos D1 Mini ESP8266 es un microcontrolador, es decir, posee su propio procesador, memoria RAM, memoria rom, oscilador y su propio SoC Wifi. De esta forma podríamos usar el ESP8266 de forma independiente, sin necesidad de otros dispositivos o placas microcontroladoras, aunque este chip también puede ser usado en conjunto con otros aparatos siendo el intermediario para una conexión Wifi.

El Wemos D1 mini posee un regulador de voltaje de 3.3V en placa, que permite alimentarla directamente del puerto micro-USB o por los pines 5V y GND. Los pines de entradas/salidas (GPIO) trabajan a 3.3V por lo que para conectar sistema de 5V es necesario un reductor de tensión DC/DC.

El microcontrolador ESP8266 de Espressif Systems es un chip especialmente diseñado para las necesidades de un mundo IOT, integra un potente microcontrolador con arquitectura de 32 bits y conectividad WiFi.

La plataforma ESP8266 permite el desarrollo de aplicaciones en diferentes lenguajes como: Arduino, Lua, MicroPython, C/C++, Scratch. Al trabajar dentro del entorno Arduino

IDE podremos utilizar un lenguaje de programación conocido y hacer uso de un IDE sencillo, además de la amplia cantidad de librerías disponibles en internet y en su propia base de datos. La integración y programación de esta placa a través del software de Arduino es muy reciente, pero desde ese entonces la placa se volvió mucho más atractiva, debido a la gran comunidad detrás de este software.

8.1.3.1. Especificaciones Técnicas

Voltaje de Alimentación: 5V DC
Voltaje de Entradas/Salidas: 3.3V DC (No usar 5V)
Placa: WeMos D1 mini
Chip conversor USB-serial: CH340G
SoM: ESP-12E (Ai-Thinker)
SoC: ESP8266 (Espressif)
CPU: Tensilica Xtensa LX3 (32 bit)
Frecuencia de Reloj: 80MHz/160MHz
Instruction RAM: 32KB
Data RAM: 96KB
Memoria Flash Externa: 4MB
Pines Digitales GPIO: 11 (3.3V)
Pin Analógico ADC: 1 (0-1V)
Puerto serial UART: 1 (3.3V)
Certificación FCC
Antena en PCB
Corriente Standby: 40uA
Corriente Pico: 400mA
Consumo corriente promedio: 70mA
Consumo de potencia Standby < 1.0mW (DTIM3)
Dimensiones: 35*26*12 mm
Peso: 6 gramos

8.1.3.2. Conectividad

802.11 b/g/n
Wi-Fi Direct (P2P), soft-AP
Stack de Protocolo TCP/IP integrado
Procesador MAC/Baseband integrado
Módulos WEP, TKIP, AES y WAPI integrados
PLLs, reguladores, DCXO y manejo de poder integrados
Potencia de salida de +19.5dBm en modo 802.11b
STBC, 1×1 MIMO, 2×1 MIMO
SDIO 2.0, SPI, UART

8.1.4. Cargador LiPo

Modulo cargador de baterías de LiPo. Usa el circuito integrado TP4056 configurado en una corriente de carga de 1A.

La entrada es por medio de un conector USB, ya sea microusb, miniusb, tipo-C, etc. Alguno de estos módulos cuenta con un circuito de protección de batería contra sobre descargas (tensión de la batería menor a 3V), cortocircuito, o contra sobre tensiones (tensión de la batería mayor a 4.2v).

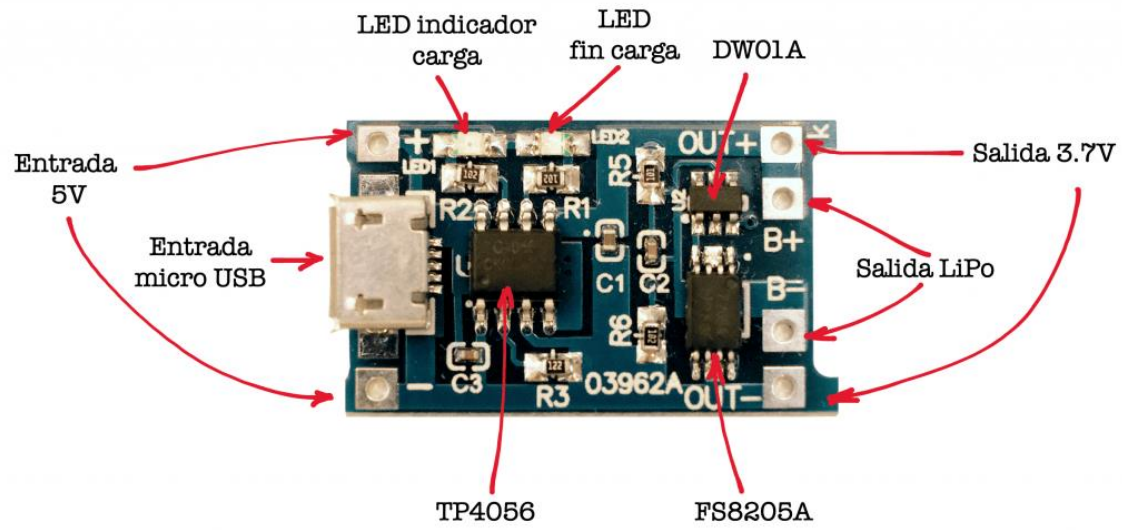


Ilustración 15: Cargador de baterías
Fuente: https://bikepixels.com/product/modulo-de-carga-lipo-tp4056-micro-usb/?lang=es

Este chip es uno de los más usados a la hora de cargar baterías LiPo debido a su reducido tamaño, bajo costo y excepcional método de carga. Comienza su carga a corriente constante y la finaliza a tensión constante, de esta forma a medida que se acerca a la carga completa reduce la corriente entregada a la batería y esto hace que se alargue la vida útil de la misma.

8.1.4.1. Características

Chip: TP4056
Método de carga: Lineal
Corriente de carga: 1A
Precisión de carga: 1.5%
Tensión de entrada: 4.5V-5.5V
Tensión de plena carga: 4.2V
Indicador de carga: Rojo cargando - Verde carga completa

Interface de entrada: micro USB
Inversión de polaridad: NO PERMITIDA
Dimensiones: 25 * 19 * 10mm

8.1.5. Batería

Para este proyecto, desde un primer momento se buscaron baterías de celulares ya que poseen una gran capacidad, un tamaño reducido, fácil acceso y si son de celulares viejos poseen un precio muy bajo (como es nuestro caso). Una posible desventaja es que no son capaces de entregar una gran corriente instantánea, pero esta no era nuestra situación, así que nos venían perfecto.

Después de varias pruebas de consumo y tamaño se llegó a la batería final. La elegida es de NOKIA, exactamente el modelo BL-5J, con una capacidad de 1430mAh, y cada pulsera lleva dos, dándonos una capacidad total de 2860mAh.

Estas baterías fueron usadas en los siguientes dispositivos: NOKIA 5228, NOKIA 5230, NOKIA 5800, NOKIA C3, NOKIA N900, NOKIA X6, NOKIA Lumia 520, NOKIA Lumia 530, NOKIA ASHA 302, entre otros.

Ilustración 16: Batería BL-5J
Fuente: https://es.aliexpress.com/item/32994285531.html

8.1.6. Carcasa

Para el armado de la correa, la carcasa y el soporte del sensor de la pulsera se utilizó una impresora 3D del tipo FDM (por deposición de material). Se realizaron diversas pruebas

hasta llegar al modelo final, ya sea por comodidad de uso o porque fueron cambiando los componentes electrónicos.

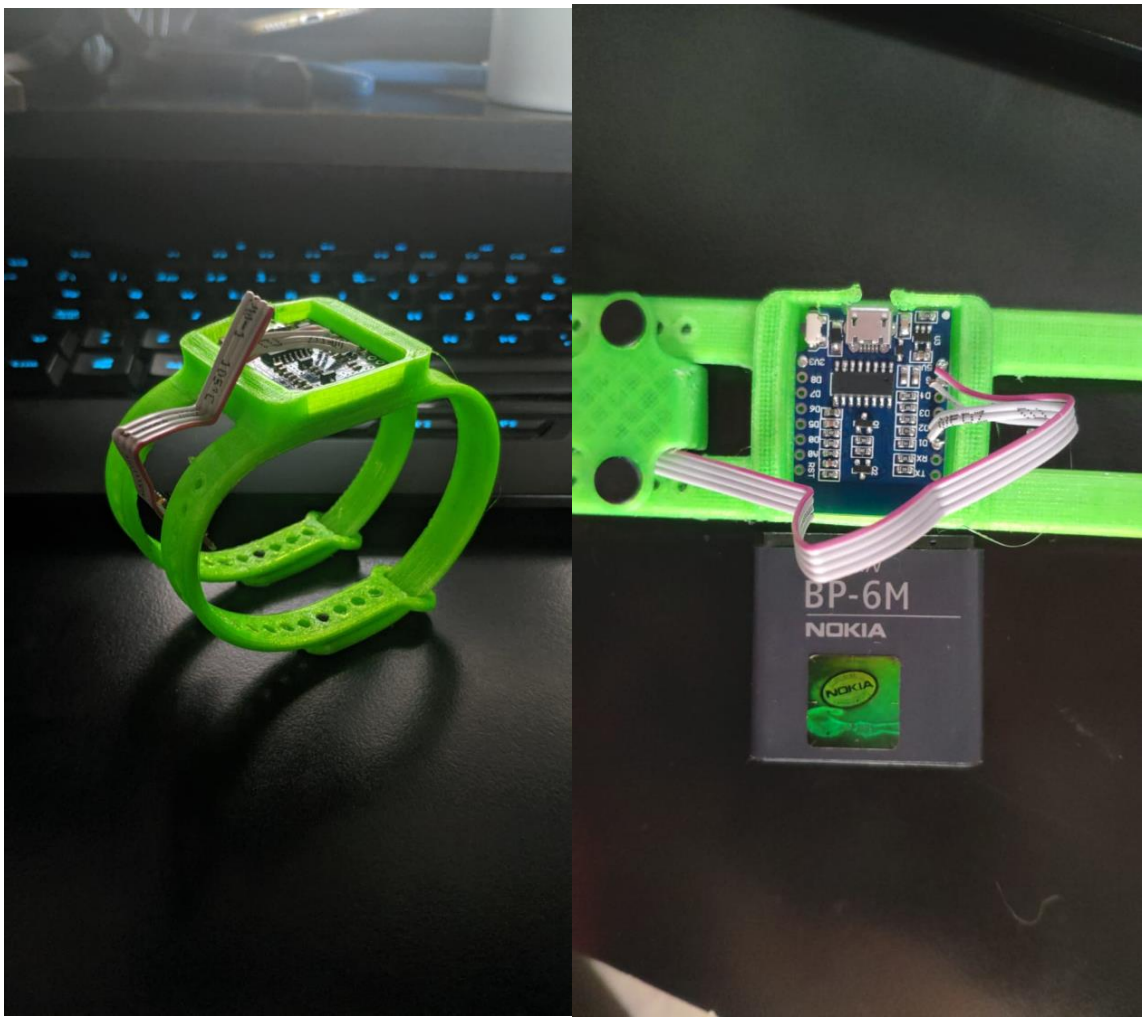


Ilustración 17: Primeras pruebas de la pulsera

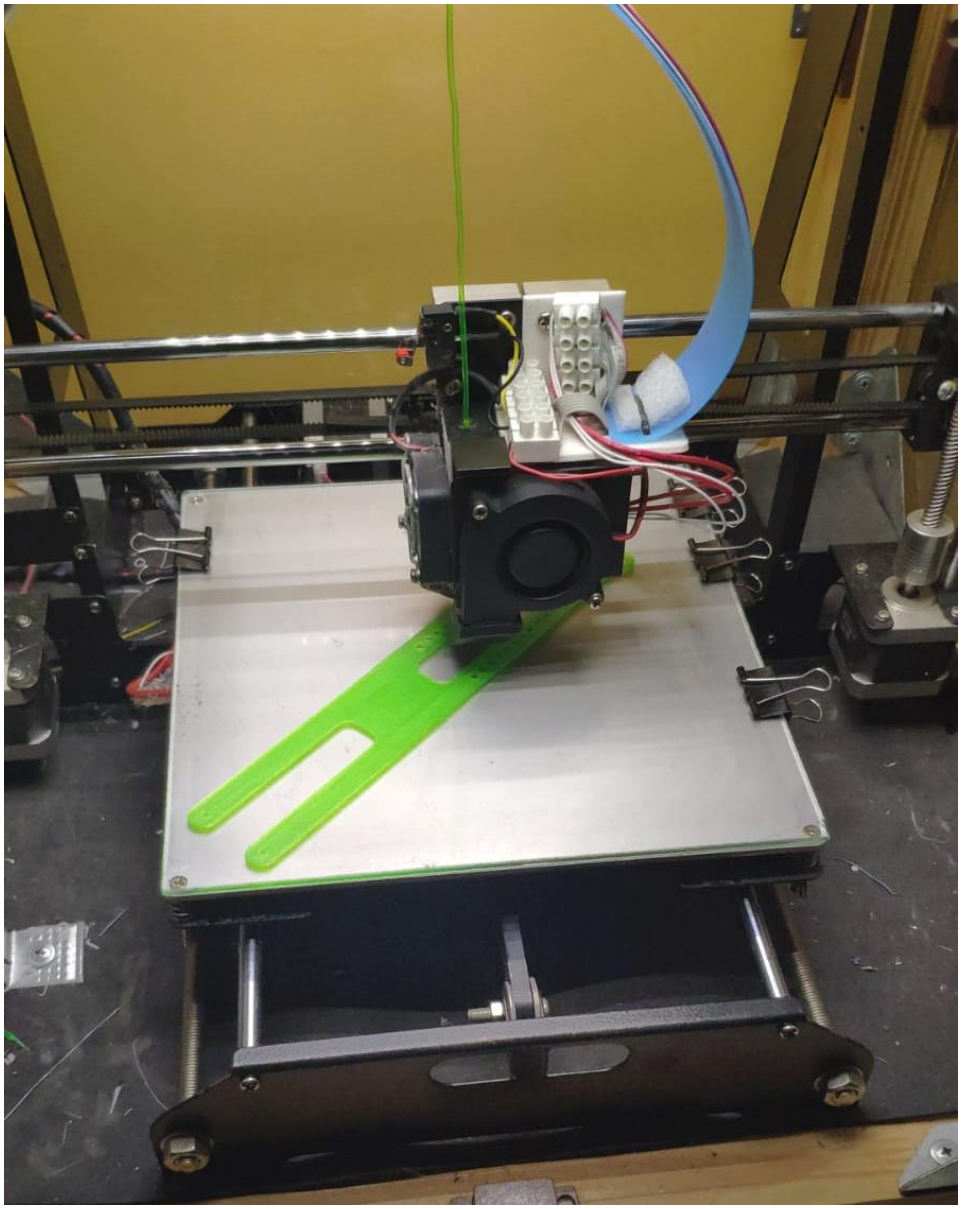


Ilustración 18: Impresión de la correa de la pulsera

Para la fabricación de la correa y del soporte para el sensor MAX30102 se eligió el material flexible TPU (poliuretano termoplástico), utilizado en diversos mercados donde nunca se le asociaron alergias. En total se realizaron más 15 correas de prueba utilizando aproximadamente 500gr de este material.

Para la carcasa donde se alojan los componentes electrónicos se utilizó un filamento de PLA aditivado (ácido poliláctico, fabricado a base de recursos renovables como el almidón de maíz), pudiendo así poder hacer paredes muy finas sin perder rigidez ni dureza en la

pulsera. En este caso se realizaron más de 10 pruebas utilizando alrededor de 300gr de este material.

La impresión de cada correa lleva un tiempo aproximado 4 horas, y cada carcasa 2 horas. Podríamos decir que tendríamos una pulsera cada 6 horas, con un peso total de 51gr contando todas sus partes.

8.1.7. Esquema de montaje

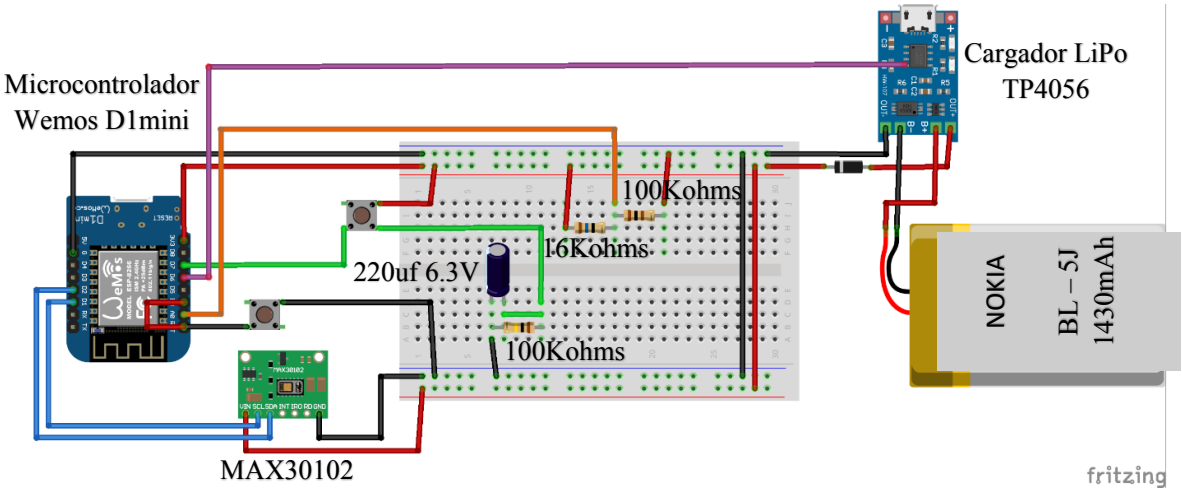


Ilustración 19: Montaje de componentes
Fuente: elaboración propia

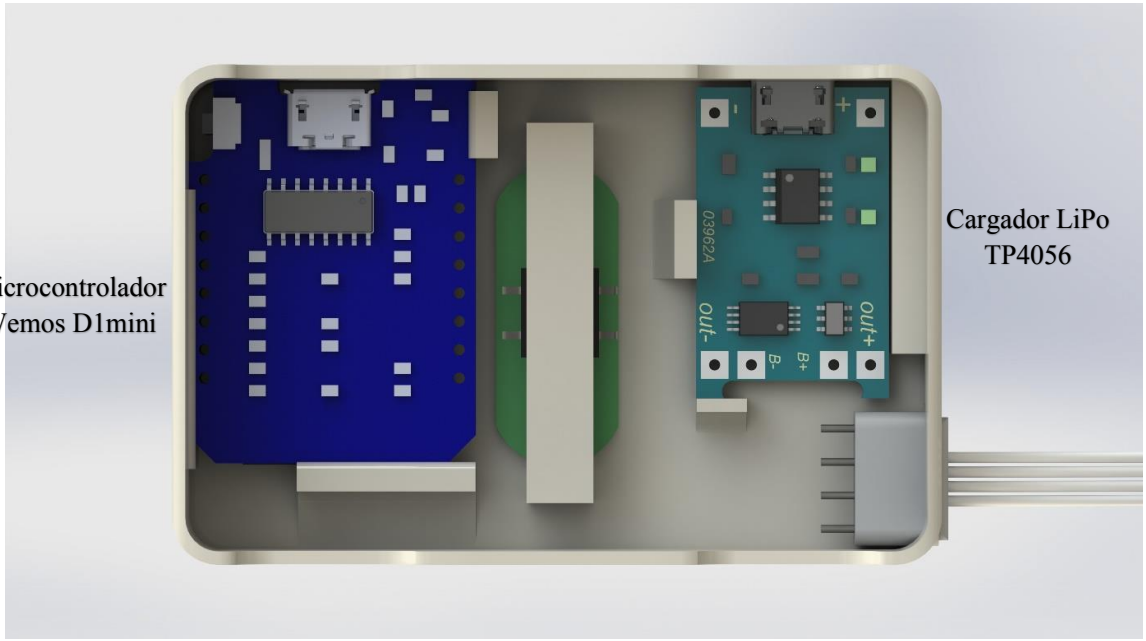


Ilustración 20: Diseño digital para ubicación de componentes
Fuente: elaboración propia

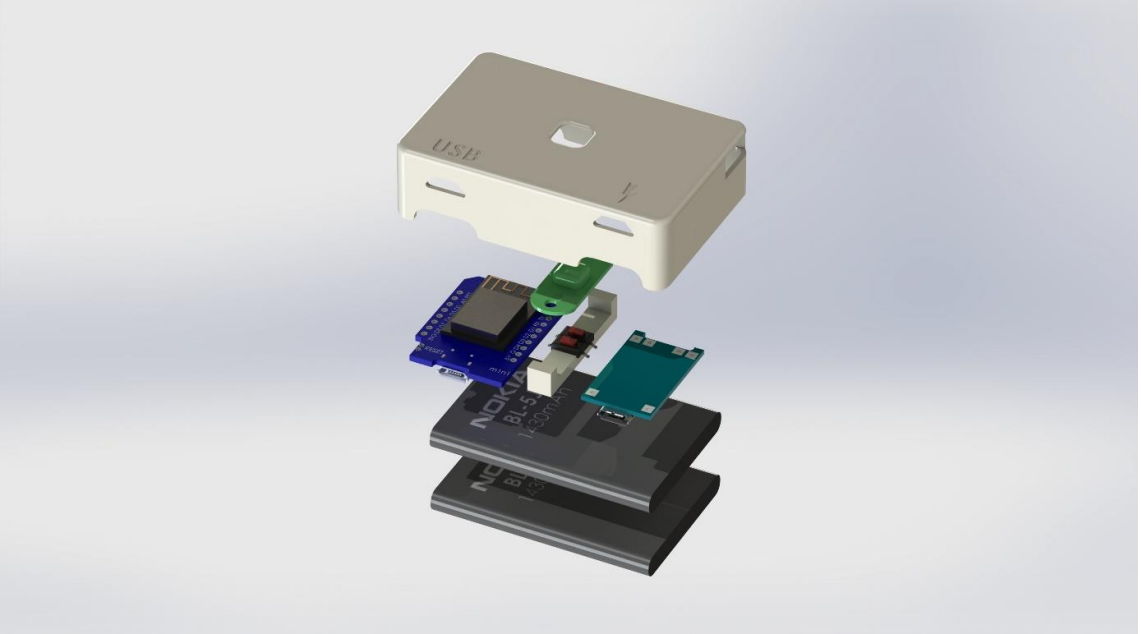


Ilustración 21: Diseño digital con la superposición de componentes
Fuente: elaboración propia

8.1.8. Software de la pulsera

La pulsera no se encuentra prendida todo el tiempo, si no que hace ciclos. Simplificando muchos pasos se podría dividir ese ciclo en 4 pasos:

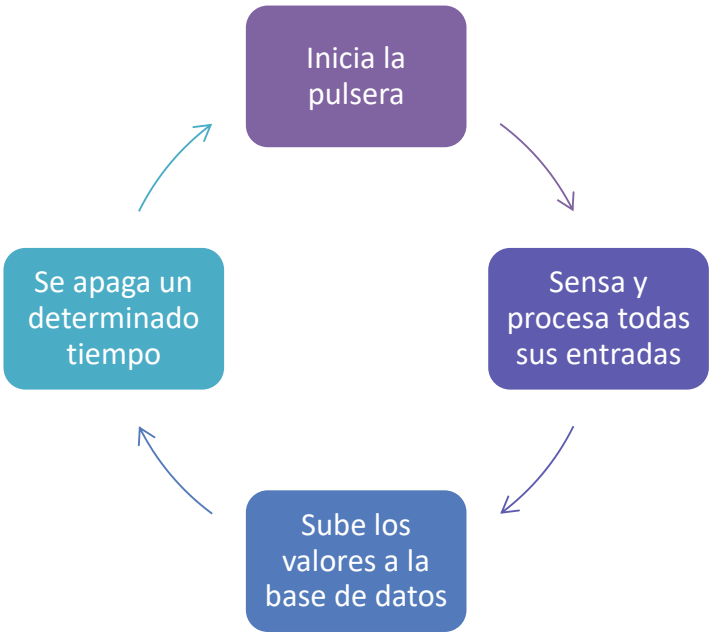


Ilustración 22: Comportamiento del software de la pulsera
Fuente: elaboración propia

Donde se podría decir que empieza prendiéndose, inicia todas las variables y librerías, establece los valores de la base de datos, del WiFi, del timestamp y finalmente establece una conexión I2C con el sensor y hace sus respectivos seteos. En este momento es cuando la pulsera chequea si se presionó el botón de pánico una vez, tres veces o no se presionó.

Ahora es cuando vamos a hablar del porqué del capacitor. La pulsera cuenta con un único botón, que en verdad internamente estamos presionando dos pulsadores del tipo tact switch.

Nosotros queremos que una vez que se aprete el botón de pánico, este lo informe lo más rápido posible a la base de datos, que a su vez se visualizara en la página web. Pero como no sabemos en qué estado se encuentra pulsera, si sensando, iniciándose, subiendo datos, apagada, etc., no importa cuál sea este, interrumpa todo lo que está haciendo y suba a la base de datos que se apretó el pulsador. Para ello la forma que se nos ocurrió es forzando el reinicio de la pulsera a través del botón, pero no podemos tener este último como botón de reseteo y a su vez como entrada del microcontrolador informando que se apretó el botón. Además, que la pulsera se reinicie forzosamente o complete su ciclo iniciándose por sí sola para el procesador es lo mismo. Por la tanto debíamos tener una memoria que nos indique que se presionó el pulsador de pánico. Para eso es que esta el capacitor y los dos tact switch. Uno se encarga de poner a masa el pin de reset, y el otro botón se encarga de cargar el capacitor. Por lo tanto, una vez que se presione el único botón visible de la pulsera, este resetea el microcontrolador y carga el capacitor que tiene su pata de positivo en una de las entradas del ESP8266. En el instante que se suelta el pulsador, la pulsera se inicia y ahí es cuando pregunta si la entrada de botón de pánico está en “HIGH” (que en verdad pregunta si el capacitor está cargado o no). De esta forma nos aseguramos que sin importar en qué estado se encuentra el microcontrolador, si se aprieta el botón de pánico lo primero que va a hacer es informar a la base de datos de que este último fue presionado. Este circuito se puede ver carillas más arriba en el esquema de montaje (ver Ilustración 19).

Una vez explicado esto podemos continuar con el ciclo de la pulsera. Como dijimos anteriormente, dependiendo cuantas veces se presione el botón de pánico o directamente no se presione son los distintos caminos que puede tomar el software.

Primer caso, donde no se presionó el botón de pánico, ni la pulsera se está cargando. En esta situación, el software se va a iniciar, cargará todas las librerías, establecerá conexión con el sensor, y empezara a leer y procesar los datos del MAX30102 por 20 segundos, este es

el tiempo que tarda en estabilizarse la medición de pulsaciones por minuto y dar un valor correcto, lo mismo sucede con el porcentaje de oxígeno en sangre. Una vez transcurrido este tiempo la pulsera se conecta a la red WiFi que tenga configurada en su memoria y sube los valores de pulsaciones por minuto, oxígeno en sangre y temperatura a la base de datos. A su vez también carga en qué estado se encuentra la pulsera.

Posee siete posibles estados, el primero es el mencionado anteriormente, donde está todo correcto, no se presionó el botón de pánico ni se está cargando y el sensor se encuentra en el dedo. Es el único estado en el cual se van a subir datos de pulsaciones por minuto, oxígeno en sangre y temperatura corporal. El segundo es igual al primero, pero este informa que la pulsera tiene batería baja. El tercero es parecido al anterior, pero en este caso la pulsera detecto que el sensor no estaba en el dedo, por lo tanto, no sube ningún valor de PPM (Pulsaciones por minutos), oxígeno en sangre y temperatura, si no que carga que se encuentra en el estado número tres. El cuarto estado es cuando la pulsera entra en modo StandBy, en este caso el microcontrolador se inicia y pregunta si el botón de pánico se apretó tres veces, si fue así, se carga en la base de datos que la pulsera se pondrá en modo StandBy y esta se apaga indefinidamente. Quinto estado es cuando la pulsera se está cargando, sexto es cuando tiene la carga completa. Por último, el séptimo estado es cuando se presiona una sola vez el botón de pánico, de esta forma el software lo único que hace es cargar en la base de datos que se presionó el botón de emergencia.

Por ahora solo hablamos del primer caso, donde se podría decir que la pulsera funciona con normalidad, donde se cumple el ciclo de prenderse, sensar, subir los valores a la base de datos y apagarse. En este estado el microcontrolador se apaga por 40 segundos y repite el ciclo.

En el segundo caso, se cumple lo mismo que en el primero, pero a la hora de subir en qué estado se encuentra, informa que la pulsera tiene batería baja, y se apaga por 40 segundos.

Tercer caso, el software se inicia, carga las librerías, establece conexión con el sensor, pero este nunca sensa la presencia del dedo, por lo tanto, después de 20 segundos, se conecta a la red WiFi, y sube a la base de datos el estado en el que se encuentra, que es el número tres, significa que el sensor esta fuera del dedo, y la pulsera se apaga por 60 segundos.

Cuarto caso, la pulsera se inicia, carga las librerías, y antes de establecer conexión con el sensor, detecta que el botón de pánico se presionó 3 veces, por lo tanto, no pierde tiempo en sensar, se conecta a la red WiFi, carga el estado número cuatro y se apaga indefinidamente.

Quinto caso, la pulsera se inicia, carga las librerías, y detecta que esta está conectada al cargador, por lo tanto, no sensa, se conecta a la red WiFi, y carga en la base de datos el estado número cinco, en esta situación el microcontrolador se apaga por 180 segundos.

Sexto caso, igual al quinto, pero en este caso carga en la base de datos el estado número 6, significa que la pulsera tiene la carga completa.

Séptimo caso, la pulsera se inicia, carga las librerías, y detecta que el botón de pánico fue presionado una sola vez, por lo tanto, no pierde tiempo en sensar, se conecta a la red WiFi, carga en la base de datos el estado número siete y se apaga por 40 segundos.

Podemos observar que la pulsera hace ciclos, donde pasa cierto tiempo apagada, de esta forma hacemos que se gaste lo menos posible la batería. Lo mismo hacemos con la conexión a la red WiFi, siempre lo hacemos a lo último, de esta forma el consumo se reduce drásticamente.

8.1.8.1. Actualizaciones vía OTA

Todas las pulseras tienen la posibilidad de ser actualizadas vía web, es decir, que no importa en que parte del mundo nos encontremos, ni la cantidad de pulseras fabricadas en ese momento, cuando estas se conecten a un cargador, controlaran que versión de firmware poseen y cuál es la última en la base de datos. Si hay una nueva versión, o distinta a la que poseen, se actualizarán automáticamente. Ya sea para una versión más estable, o mejor, con más funciones, etc. Todo esto sucede gracias a la página web OTAdrive.com donde se pueden visualizar todas las pulseras, cuando fue la última vez que se fijaron si hay una nueva versión, cuantas veces lo hicieron, en que versión se encuentra cada microcontrolador, entre otras funciones. Esta página es muy segura, solo se podrá acceder a estas pulseras quienes conozcan el nombre de usuario y la contraseña de la cuenta asociada.

Este servicio es gratuito hasta cierta cantidad de pulseras, es por esto que dependiendo el sitio a cubrir variara su costo total.

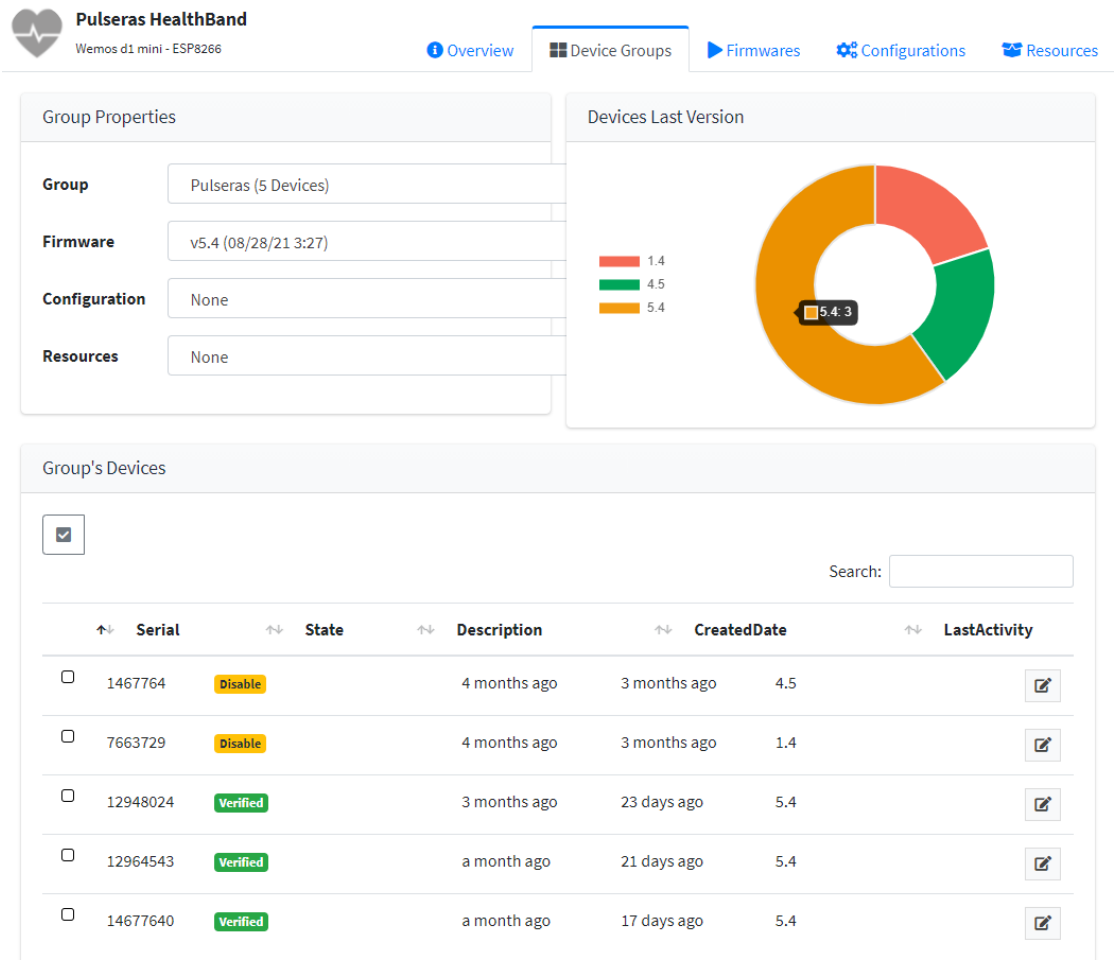


Ilustración 23: Tablero OTAdrive.com

8.1.8.2. Consumos de batería

Después de varias pruebas, más de 50 versiones de software distintas, se llegó a la mejor desde nuestro punto de vista, donde se consiguen valores correctos de pulsaciones por minuto, oxígeno en sangre, temperatura y una duración de batería razonable.

Gracias a los ciclos de prendido y apagado, al control de encendido del sensor dependiendo de la situación, el escaso tiempo en que se conecta a la red WiFi se llegó a un aproximado de 3 a 4 días de duración de batería.

Se usaron dos baterías de 1430mha y la pulsera pasa por 3 estados de consumos diferentes, donde funcionando sin conexión WiFi da un consumo de 30mha; funcionando con conexión WiFi, el máximo consumo es de 85mha y el ultimo estado es cuando se apaga consumiendo 18mha.

Todos estos consumos son aproximados, ya que puede varían unos 5mha de los valores dados

8.1.8.3. Conexión WiFi

La conexión del WiFi no está fijada por firmware, si no que esta puede ser configurada a través de una notebook o celular. Si la pulsera aún no tiene configurado el WIFI de donde se encuentra, esta generará una red WIFI, donde deberemos acceder, y desde ahí la configuraremos, eligiendo la red del lugar, escribiendo la contraseña y finalizará la configuración. Estos valores quedaran guardados en la EPROM del microcontrolador. De esta forma no importa cuantas veces se apague o se prenda la pulsera, siempre se conectará a esa red WiFi, o a las que tenga guardadas, ya que, si llegamos a ir a otro lugar con una conexión distinta de WiFi, tendremos que repetir la configuración. La pulsera creara una red WiFi, y tendremos que repetir los pasos mencionados anteriormente.

8.1.8.4. Dualidad de pulseras

Debido al método de carga y a la duración de la batería, las pulseras trabajan de a pares. Mientras una se está cargando la otra se está usando y viceversa. De esta forma el usuario nunca dejara de ser monitoreado. En la web estas pulseras están sincronizadas, donde se puede ver el histórico de las dos pulseras en uno solo, a su vez aparece que pulsera está funcionando y al momento de configurar una, la otra copia esta configuración. Esta dualidad está pensada para que mientras una se usa, la otra se cargue, y esta no debería desconectarse del cargador hasta que no vaya a ser usada, más que nada para prevenir la autodescarga de la batería.

8.1.8.5. Batería baja

El método de informar al usuario que la pulsera tiene batería baja es prendiendo un led azul. Cuando este está prendido indica que la batería tiene solo un 20%, por consiguiente, en las siguientes horas debería poner a cargar la pulsera que tiene y empezar a usar su pareja, es decir, la pulsera que se estaba cargando.

El sensado de la batería se hace a través de un divisor resistivo conectado a la entrada analógica del microcontrolador, de esta forma es que se puede ver cuál es el porcentaje de carga de la misma.

8.1.8.6. Identificación de cada pulsera

Cada pulsera posee un número de identificación único rotulado en su parte frontal.

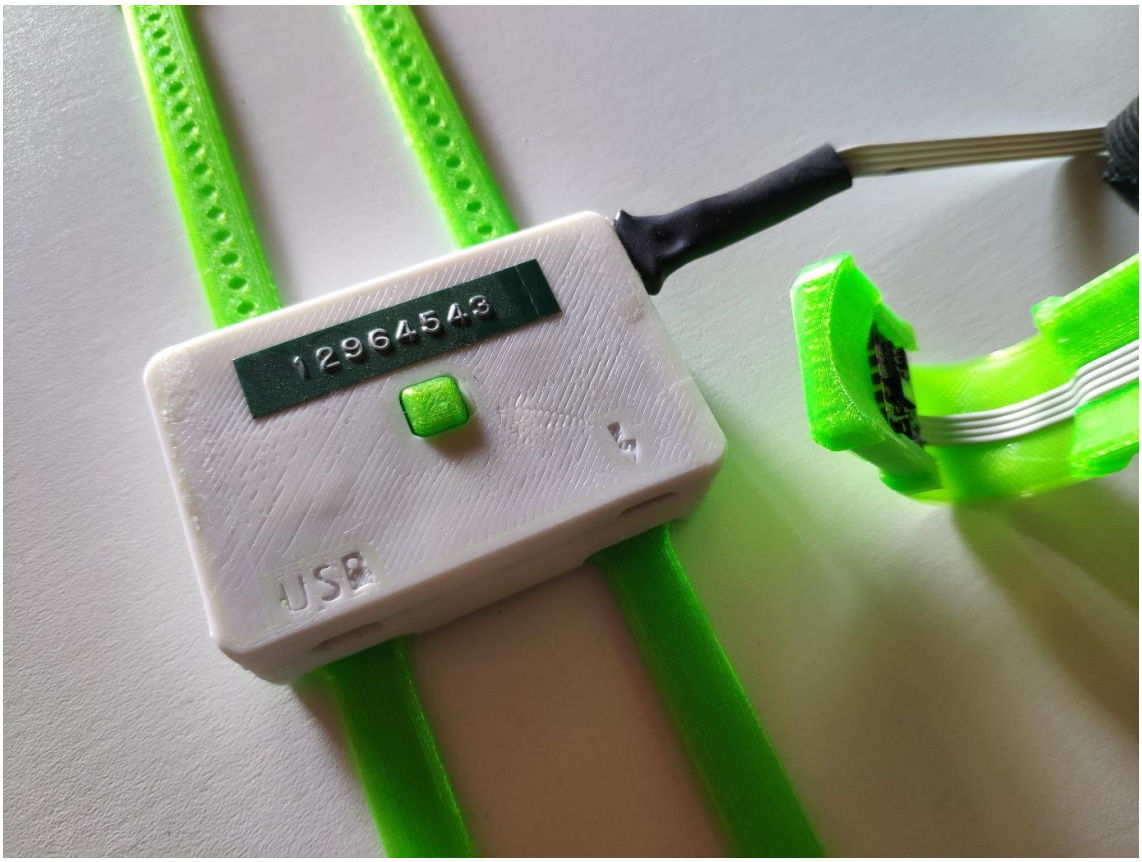
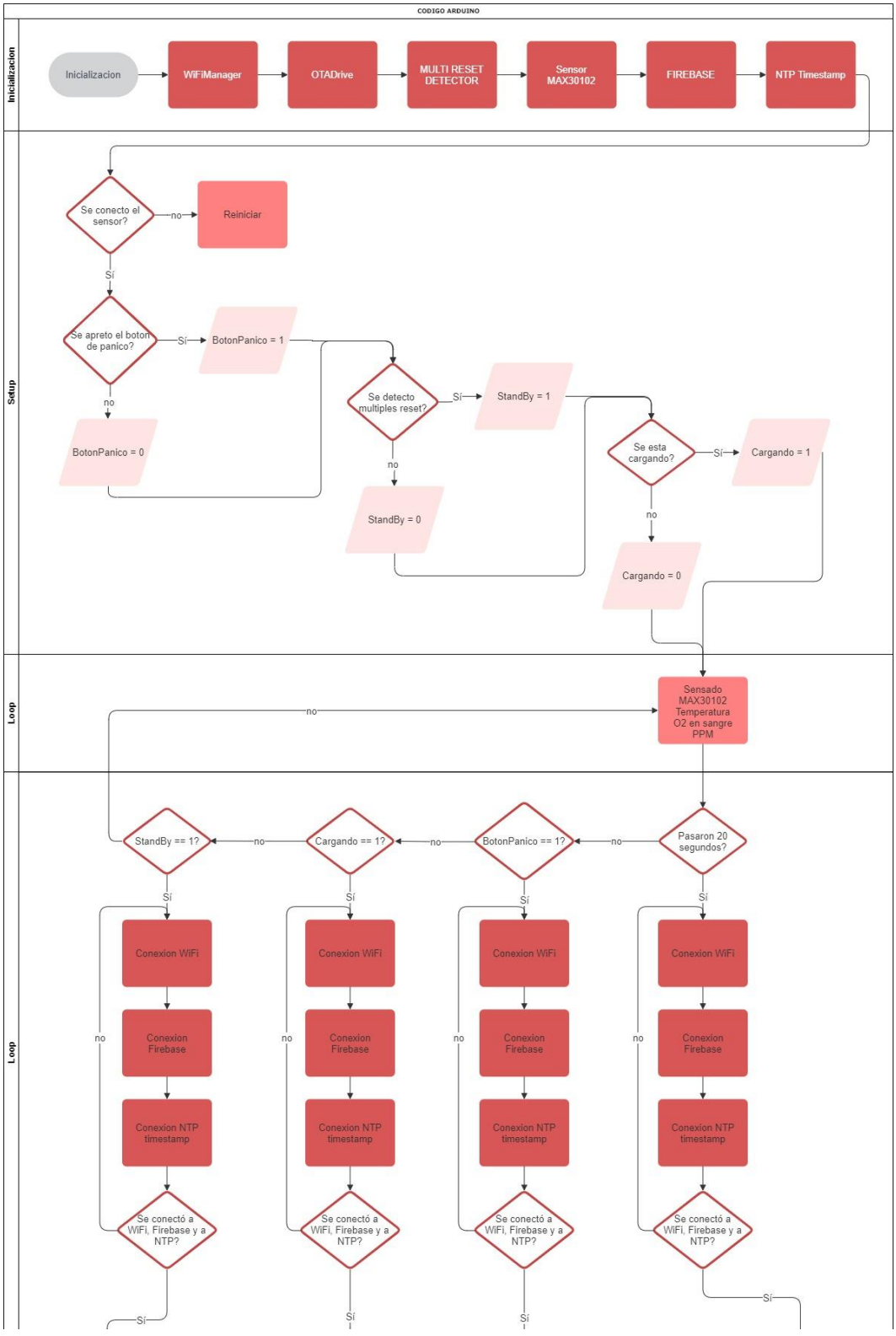


Ilustración 24: Numero de identificación de la pulsera

Este número es proporcionado por el mismo microcontrolador a través de una función por la cual se consulta este código, de esta forma es que todas las pulseras cuentan con el mismo código de programación. Esto nos permite poder actualizar todas las pulseras muy fácilmente sin tener que hacer un software distinto para cada una.

A través de este número es que diferenciamos cada pulsera en la base de datos y a su vez es por el cual podemos tener dualidad de pulseras, ya que sabemos que códigos conforman cada par de pulseras en la web.

8.1.8.7. Diagrama de flujos del código de Arduino



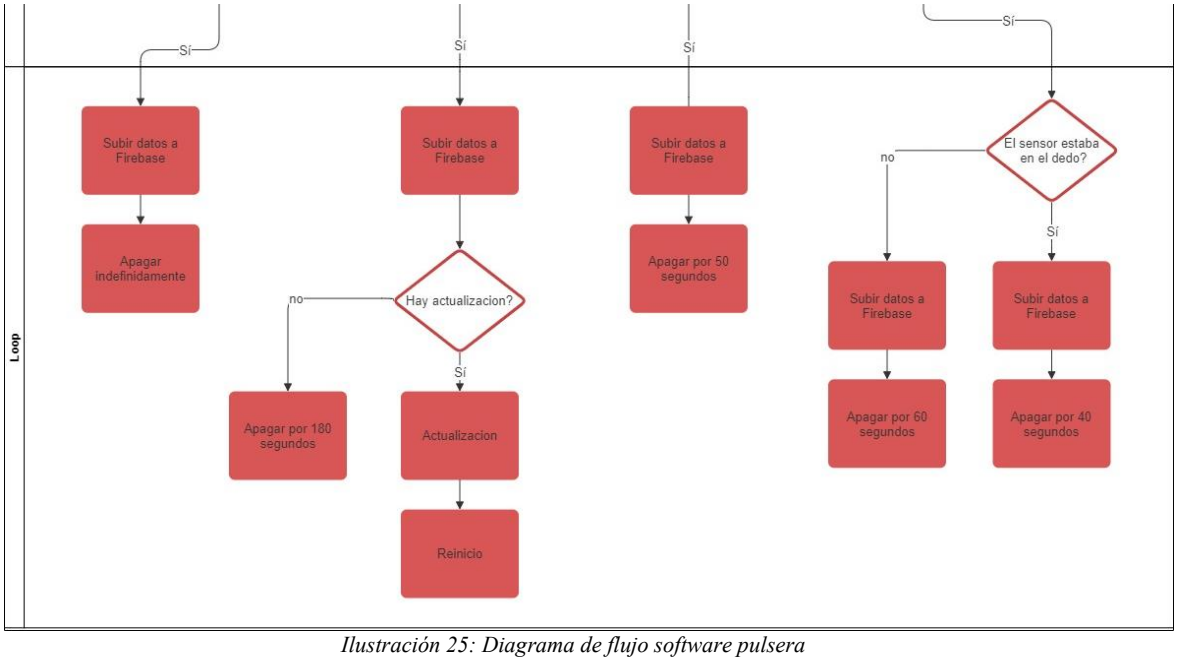


Ilustración 25: Diagrama de flujo software pulsera
Fuente: elaboración propia

8.2. Firebase

Firebase es uno de los servicios corporativos para desarrollo de aplicaciones, capaz de proveer distintas herramientas de procesamiento y almacenamiento de datos. Desde funciones que corren por detrás de la misma hasta hosteo de aplicaciones webs.

Hoy en día es uno de los servicios de mayor confiabilidad y conocimiento debido a su valor de precio y calidad, presentándose como unos de los competidores de Amazon Web Servicies (AWS).

También presenta una utilidad muy practica que permite hacer un desarrollo de prueba, brindando una amplia gama de utilidades a valor cero. En caso de tener mayor escalabilidad se puede incorporar utilidades según el consumo diario y tráfico.

Gracias a este, pudimos realizar una aplicación que se encuentra automatizada según la cantidad de pulseras que haya.

Una vez ingresado a la sesión de esta, tiene una descripción general del proyecto que sirve para visualizar a grandes rasgos el funcionamiento de la aplicación. (ver Ilustración 26).

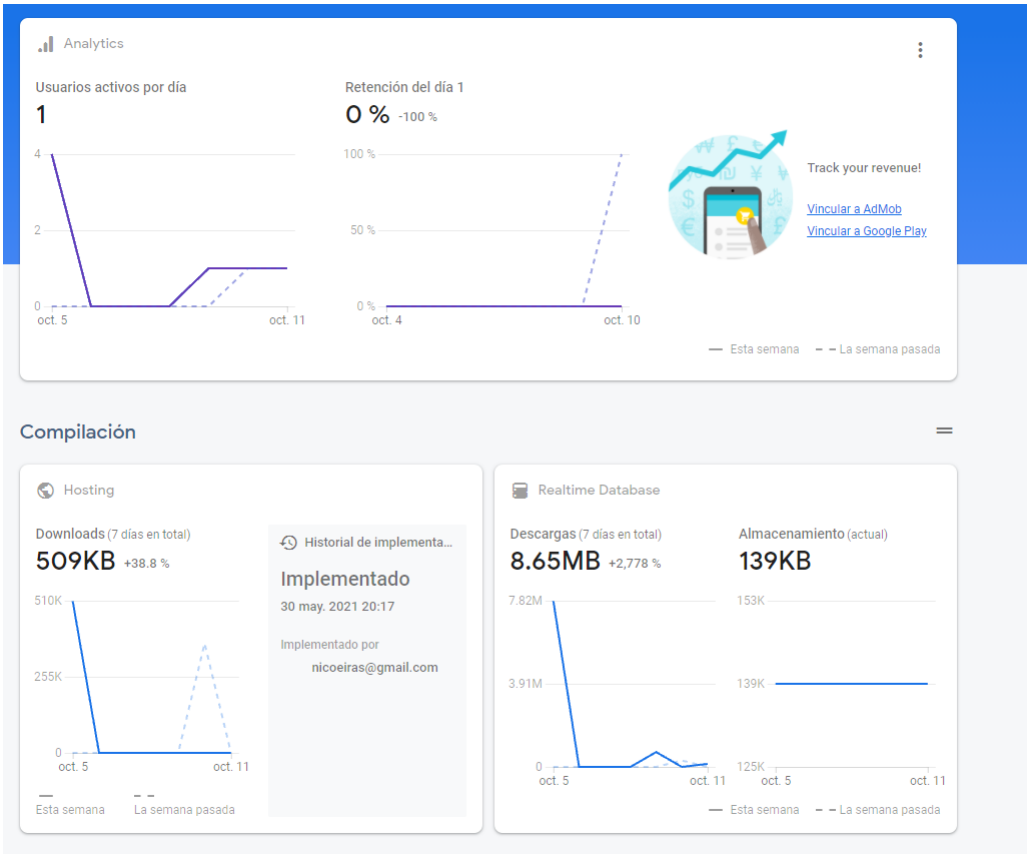


Ilustración 26: Tablero Firebase

8.2.1. Authentication

Para este desarrollo es necesario identificar los usuarios, conociendo su identidad para poder guardar datos de forma segura en la nube y así brindar una solución personalizada según la persona y sus dispositivos.

Firebase Authentication cumple función de un servicio backend con SDK y bibliotecas fácil de implementar, previamente elaboradas para poder autenticar los usuarios de la aplicación. Tiene distintas formas de realizar este proceso, puede ser por contraseñas, números de teléfono, proveedores de identidades, como Facebook o Gmail.

Consideramos que este servicio es de suma utilidad ya que la información manejada es muy sensible porque son datos personales de cada usuario de pulsera. Por eso nace la necesidad de implementar alguna protección ubicada por detrás de la pulsera y la página.

La forma en la que hacemos el autenticado para nuestro desarrollo es mediante la utilización de un correo electrónico que será provisto y guardado por nosotros en Firebase, para así evitar el contacto del usuario con la información ubicada por detrás.

Por una parte, la pulsera en su código (ver sección 14.2), dispone de las líneas necesarias para entablar una conversación con Firebase y así guardar los valores de las personas.

Cuando se desea acceder a la página web, deberá de logearse mediante este mismo correo y contraseña para que el sistema sea capaz de conocer su identificación y así mostrar sus datos. Al ingresar al tablero podrá saber quién es la persona conectada mediante un mensaje ubicado en la esquina superior derecha que mostrará el correo del usuario. Además, la sesión permanecerá abierta hasta que el usuario decida deslogearse de la misma, donde la próxima vez será dirigido al Login para brindar las credenciales necesarias.

La limitación de cuentas brindadas es uno de los aspectos más importantes para este desarrollo porque consideramos que a mayor contacto de cuentas, más probabilidades habría de hacer visible información a personas no autorizadas. Es por esto que a la hora de crear la página para la persona que utiliza la pulsera, optamos por brindarle un código para así no tener contacto con el servicio que se encuentra por detrás.

En caso de que el autenticado no sea posible, ya sea por una falta de ortografía en el correo o contraseña, el proceso no podrá ser llevado a cabo y volverá a pedir credenciales hasta que se produzca el enlace.

El servicio es gratuito por la cantidad y tipo de autenticados diarios que se producen, consideramos que la pulsera se autenticará cada vez que sense y la pagina se autenticará una sola vez porque estará corriendo constantemente en una computadora en el sitio de control.

Tendremos un apartado dentro de Firebase Authentication que se encargara de mostrarnos los usuarios activos, que previamente fueron cargados dentro del mismo servicio. También, nos dejara cargar nuevos usuarios, considerando las restricciones previamente dichas, buscamos limitar al máximo la cantidad de cuentas provistas para así mantener un estándar de seguridad alto (ver Ilustración 27).

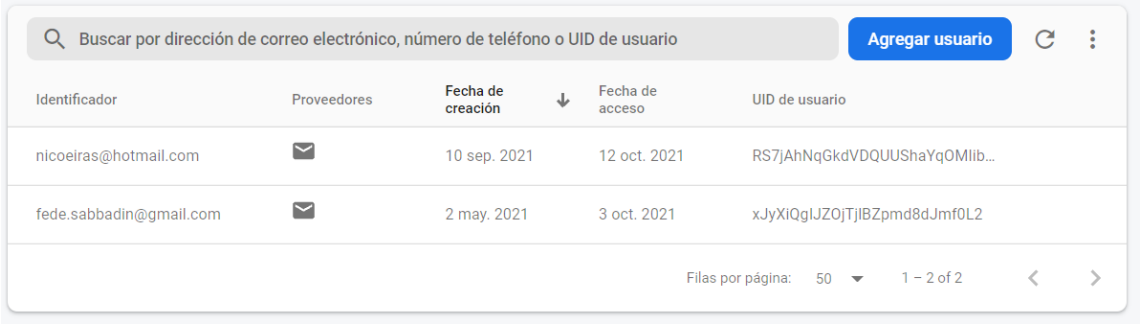


Ilustración 27: Firebase Authentication usuarios cargados

8.2.2. Realtime Database

Otro de los servicios utilizados es la base de datos en tiempo real que es capaz de almacenarlos y sincronizarlos en tiempo real en la nube. Los datos son sincronizados en todos los usuarios que previamente se autenticaron, y se mantendrán disponibles, aunque la aplicación no tenga conexión a internet. Tiene una estructura de base de datos denominada JSON y se despliega mediante la utilización de un SDK en el código de la aplicación que se busque desarrollar.

Las pulseras a la hora de sensar los datos en su código, establecen su conexión con Firebase y guardaran los datos en variables destinadas a su función. Se crea una pulsera en este servicio que dentro tendrá todas las mediciones e información del usuario, donde automáticamente se ira guardando con sus dependencias correspondientes.

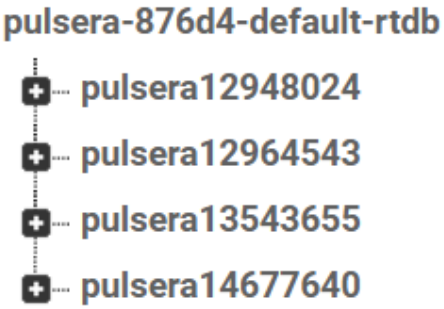


Ilustración 28: Realtime Database con las pulseras

El código de la misma está diseñado para automatizar la base de datos, es decir que, al momento de registrar una nueva pulsera, se cargara en Firebase una nueva sección con la información de esta, e ira sensando y cargando datos acorde pase el tiempo.

En el otro lado, la aplicación leerá estos datos y mediante lógica computacional será capaz de informar a la persona que visualice la página cual es el estado del usuario de la

pulsera. La página no es capaz de ver esta base de datos directamente, sino que solo se le será mostrado los aspectos necesarios y correspondientes como las mediciones y la información de la persona. Un aspecto muy importante es que el único usuario capaz de setear la pulsera a fabrica es el dueño de la página.

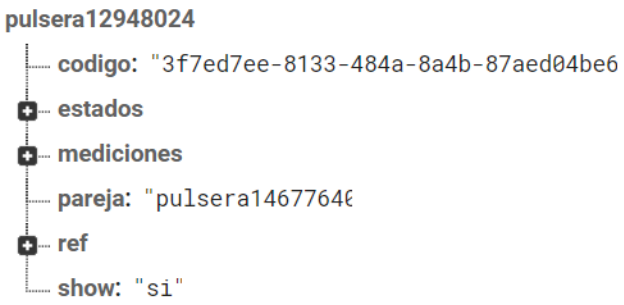


Ilustración 29: Realtime Database con información de cada pulsera

Dentro de cada pulsera se visualiza su código único, que será el mismo que tendrá la otra pulsera pareja, los estados que informan cómo se encuentra la pulsera (ver sección 8.1.8), las mediciones para cada fecha y hora, el nombre de la pulsera que es su pareja, los valores de referencia y una dependencia llamada “show”, que servirá para mostrar las pulseras en el tablero principal.

Optamos por tener separadas las mediciones y los estados para así reducir el almacenamiento de Firebase y que este sea lo más optimo posible, ya que para el estado solo almacena una variable y en cada medición hay 4 valores.

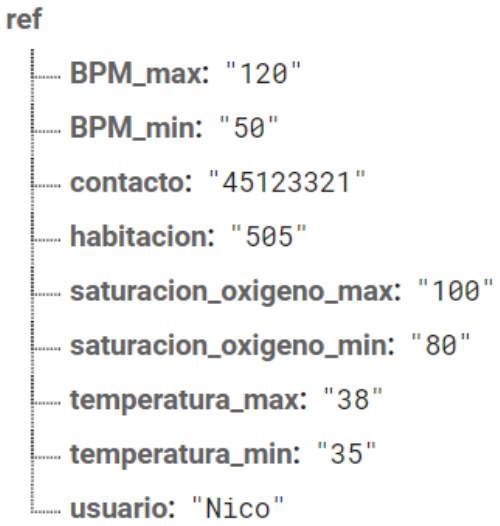


Ilustración 30: Realtime database con la referencia de la pulsera

La pulsera tiene su apartado llamado “ref” donde se encuentra toda la información del paciente, no solo los valores médicos sino la de contacto.

La base de datos permite definir ciertas reglas de acceso y edición. Para este proyecto fueron definidas como de libre lectura, conociendo una serie de datos para realizar la relación entre página y Firebase como pulsera y Firebase; y la escritura de los datos está limitada a usuarios que fueron autenticados sí o sí.

Por defecto Realtime Database tiene asignado 1 Gigabytes de almacenamiento, actualmente tenemos 139.4 Kilobytes, que representa aproximadamente 5 días por 4 pulseras de información. Haciendo un análisis con este número de días podemos obtener que el desarrollo funcionara de forma gratuita por 38868 días para 4 pulseras, si se pasara de este almacenamiento el valor sería de 5 U$S por Gigabyte.

8.2.3. Hosting

Firebase Hosting nos ayudó para hostear la aplicación web y su contenido. El cual lo instalamos mediante el uso de comandos por consola y cargamos la capeta correspondiente a la página. Es una opción muy practica porque permite probar la página de forma local, sin la necesidad de correr riesgos en desplegarla a nivel global y tener vulnerabilidades.

Configuramos el servicio de forma que, al ingresar a la dirección web, destine directamente al Login para luego ir pasando por las páginas correspondientes.

Gratuitamente Firebase brinda 10 Gigabytes de almacenamiento de la página y 360 Megabytes de transferencia de datos por día. Hoy en día la página web utiliza un almacenamiento de 1,8 Kilobytes, lo cual nos deja un amplio margen de crecimiento según avance la necesidad de aspectos varios. Un cuello de botella seria las descargas diarias que estaríamos limitados para mantener el servicio gratuito, llegado al caso tiene un valor de 0,15 U$S por Gigabyte.

8.3. Página WEB

El muestreo de la información capturada por la pulsera será mediante una página web que se encuentra hosteada en Firebase, un servicio de Google. Este servicio no solo tiene la información de la página, sino que también tiene los datos de la pulsera, siendo un intermediario entre ambos.

La página web se divide en dos grandes sectores, el del usuario y el del dueño del servicio. Donde a grandes rasgos el usuario será capaz de ver sus valores actuales para tener una alimentación del servicio; y el dueño tendrá distintas funcionalidades que producirán un servicio eficiente.

Para acceder a la misma se puede hacer desde la siguiente dirección: https://healthband.web.app.

8.3.1. Página para el dueño

Esta parte del desarrollo web es la más importante y la que más información permite conocer, debido a que está pensada para ser usada por personal con capacitación médica, que en caso de emergencia sea capaz de tomar los recaudos correspondientes.

8.3.1.1. Login

Una vez ingresado a la dirección podrá registrarse mediante su mail y su contraseña, estos deberán estar cargados en Firebase ya que autenticaran y autorizaran al usuario a poder ingresar a la web.

En el caso de que el usuario o la contraseña no sea correcta, ni que este autenticado, saldrá un mensaje informando de la situación.

El desarrollo web está pensado para que, una vez iniciada la sesión, permanecerá abierta hasta que el usuario decida salir, donde deberá cerrar la sesión sin importar en que sección se encuentre.

Por debajo tiene una opción donde podrá ingresar mediante el código del par de pulseras. Al presionarlo lo llevara a la página de Login del usuario (ver sección 8.3.2.1)

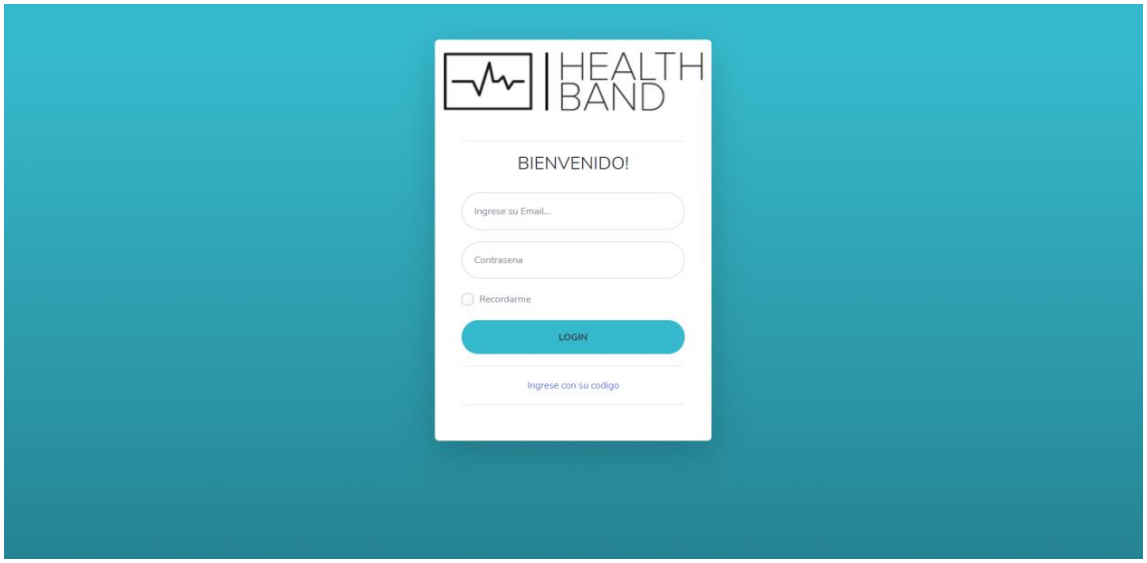


Ilustración 31: Página de logeo

Gracias al autenticado de esta página, tendremos en cada una de las secciones de la aplicación, la información del usuario con un mensaje de bienvenida (ver Ilustración 32), ubicado en el margen superior derecho al costado de la fecha y hora actual.

Ilustración 32: Usuario en sesión

8.3.1.2. Tablero

Este apartado es uno de los más importantes porque es el que brinda la primer alerta al visualizador. En la parte superior se obtiene un primer contacto con la situación de los pacientes actuales y en la parte inferior se puede ver los valores de cada paciente actuales y sus pulseras.

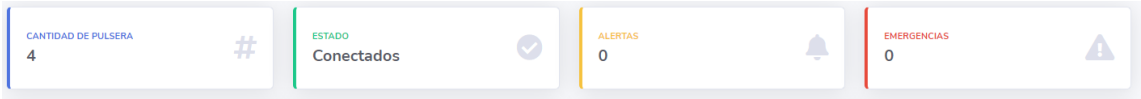


Ilustración 33: Parte superior de página tablero

Como se ve en la imagen de arriba (ver Ilustración 33), primero se muestra cuantas pulseras actualmente hay en el sistema, las cuales pueden estar en uso o en carga. Luego, se muestra el estado de las pulseras siendo conectados o desconectados, que se producirá cuando una de las pulseras tenga una diferencia de 5 minutos entre el último estado y la hora actual. Los dos últimos recuadros informan la situación actual de las pulseras, donde irán aumentando, dependiendo de las situaciones. Las alertas suman 1 si una de las mediciones

actuales está por debajo o por encima de las que la persona tiene como referencia, también suma 1 si la batería se encuentra por debajo del 5%, para ambos casos el recuadro cambiara de color blanco a rojo para generar una alerta visual. En cambio, las emergencias informan situaciones críticas donde, gracias al suceso de variables nos indican lo necesario para responder en un tiempo acotado, sumara 1 si una de las mediciones actuales y su anterior sea menor o mayor a las que debería de tener, cambiando a rojo el color del recuadro y emitiendo una alerta sonora.

En el caso de que la persona presione el botón de emergencia en su pulsera, la página está diseñada para generar la mayor alerta, primero se cambiará el estado actual a PANICO, se coloreará el recuadro a rojo y por sobre encima de las variables se escribirá un texto informando lo sucedido, todo esto mientras se emite un pitido (ver Ilustración 37).

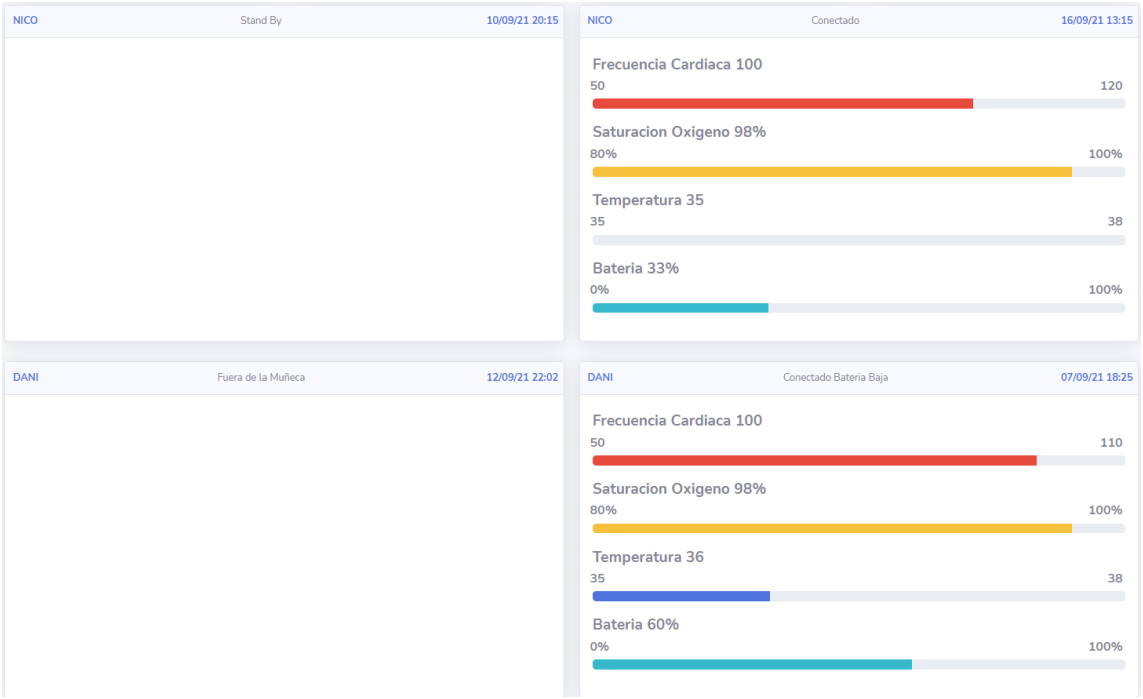


Ilustración 34: Parte inferior de página tablero

Los recuadros de las pulseras brindan mucha información. En la parte superior cuenta con el nombre de la persona, el estado actual (ver sección 8.1.8) y el horario del último estado.

Debajo está la información más importante, la frecuencia cardiaca, la saturación de oxígeno, la temperatura y la batería de la pulsera. Cada una tiene a la derecha de su nombre el valor actual y debajo tiene una barra que informa la referencia mínima y máxima del paciente.

Ilustración 35: Recuadro de una pulsera

Ilustración 36: Recuadro de una pulsera con alerta

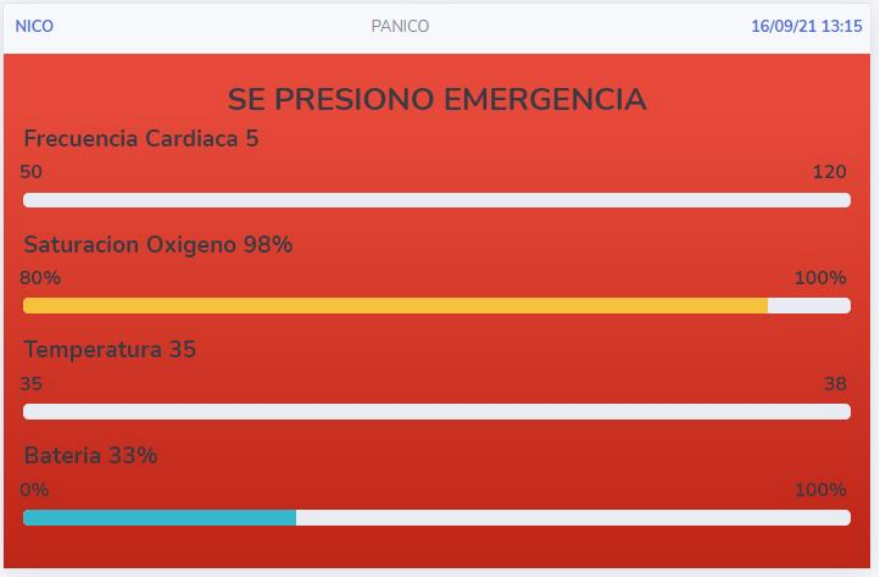


Ilustración 37: Recuadro de una pulsera presionando emergencia

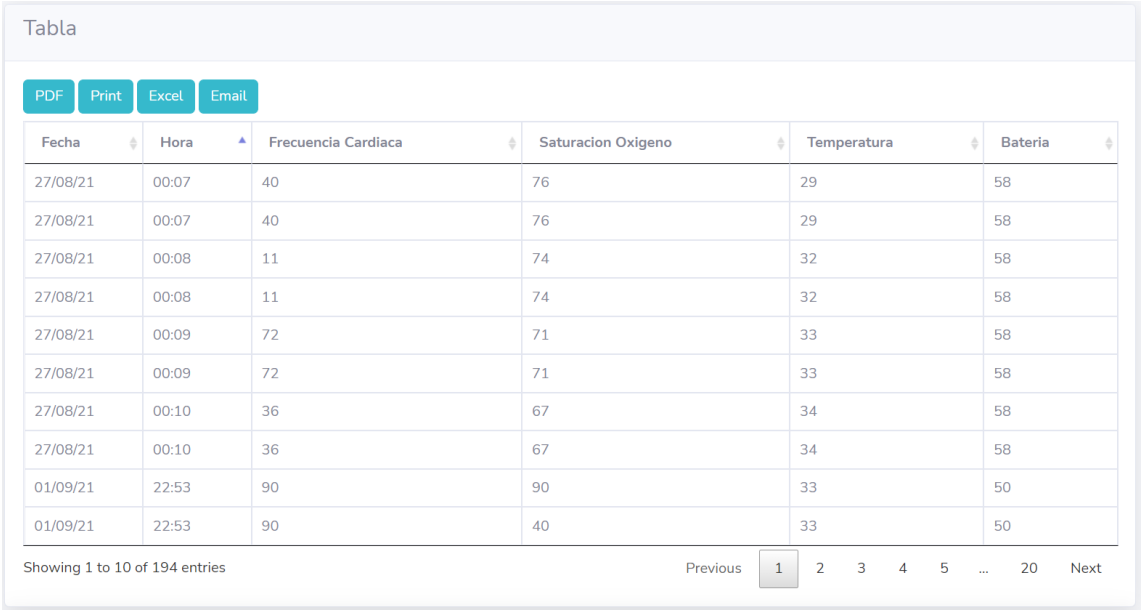
Esta página actualiza su información en cuestión de segundos, para acceder al paciente lo más rápido posible ante un inconveniente.

8.3.1.3. Histórico

El histórico se divide en dos grandes secciones, una tabla con todos los valores del usuario y el grafico de los mismos valores.

Primero, se deberá de seleccionar el par de pulseras que se desea analizar. Donde se mostrará que par se seleccionó en el medio de la página. Mediante esta selección se permite conocer la ubicación en la base de datos de la información necesaria a mostrar debajo, donde se traerá todo lo que esté por debajo de mediciones en cada una de las pulseras.

Al cargase la tabla, se hace uso de la librería de JavaScript llamada Datatables.net, que permite ordenar y visualizar la información de forma amigable al usuario. Cuenta de importantes beneficios como la paginación de los datos, obteniendo páginas de 10 filas de datos. Se disponen de botones en la parte superior para exportar los datos y realizar un análisis particular, mediante la descarga en formato de PDF o Excel, como la impresión de la tabla para tenerla en formato físico; y si se desea se la puede enviar mediante mail. Para el último caso, se le solicitara un correo electrónico, donde una vez cargado y enviado se le mostrara un cartel informando que la tabla fue enviada con éxito (ver sección 8.3.1.8).



Tabla

PDF Print Excel Email

Fecha	Hora	Frecuencia Cardiaca	Saturacion Oxigeno	Temperatura	Bateria
27/08/21	00:07	40	76	29	58
27/08/21	00:07	40	76	29	58
27/08/21	00:08	11	74	32	58
27/08/21	00:08	11	74	32	58
27/08/21	00:09	72	71	33	58
27/08/21	00:09	72	71	33	58
27/08/21	00:10	36	67	34	58
27/08/21	00:10	36	67	34	58
01/09/21	22:53	90	90	33	50
01/09/21	22:53	90	40	33	50

Showing 1 to 10 of 194 entries Previous 1 2 3 4 5 ... 20 Next

Ilustración 38: Tabla con la información histórica de las pulseras

La sección del grafico permite observar las variaciones de forma directa y en un sentido macro. Para obtenerlo es necesario usar la librería de JavaScript denominada Chart.js, con el fin de generar un objeto diseñado para el usuario final que muestre las variables con distintos colores y formatos.

Inicialmente se tiene la frecuencia cardiaca, la saturación de oxígeno, la temperatura y la batería en función del tiempo. Pero es posible ocultar variables que no se desee, para tener el grafico de una sola variable y ver su comportamiento en el tiempo sin obstrucciones en el mismo.

Al ubicarse en los puntos de la línea se muestra la fecha y el valor de esa variable.

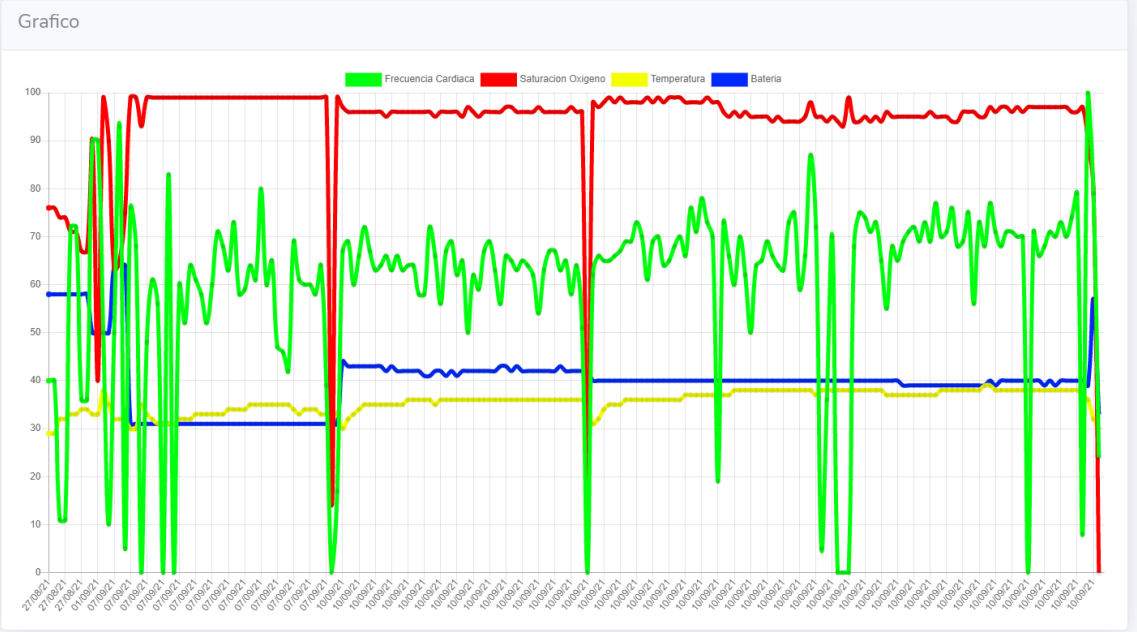


Ilustración 39: Grafico con la información histórica de las pulseras

La página cuenta con un botón, ubicado en la esquina superior izquierda que permite borrar todos los datos de la pulsera, ya sea las mediciones, estados, código, etc.; y cargarla como de fabrica con los valores predeterminados (ver sección 8.3.1.7).

8.3.1.4. Administrar

En esta sección de la página se puede conocer las parejas de cada pulsera, con su usuario actual y su código de ingreso a la página del usuario que será generado cada vez que se ponga de fabrica la pulsera (ver sección 8.3.1.7).

Por último, se tiene una columna con actividades varias, el botón verde (ver Ilustración 40) se utiliza para cargar el mail de la persona que quiere conocer los valores actuales de la pulsera, que mediante un correo automatizado podrá utilizar la página de usuario.

El botón medio, azul, se utiliza para agregar el par de pulseras al tablero y poder visualizar su información, y por último el botón rojo hace lo opuesto a lo anterior, donde al seleccionarlo y confírmalo, se puede eliminar el par de pulseras del tablero. Estos dos botones no tienen efecto en la información guardada, ya que solo se utilizan para su muestreo en el tablero.

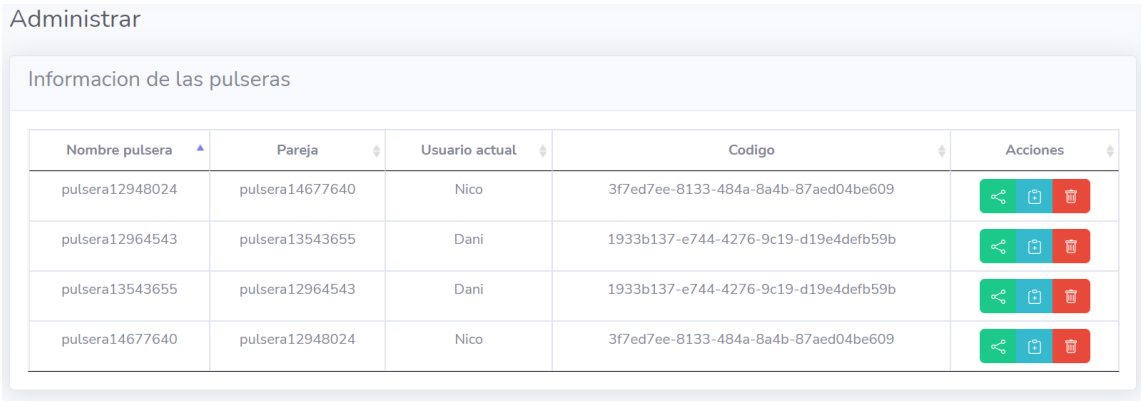


Ilustración 40: Tabla con la información de las pulseras

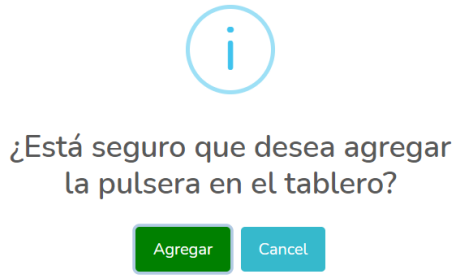


Ilustración 41: Pop up confirmativo de agregación

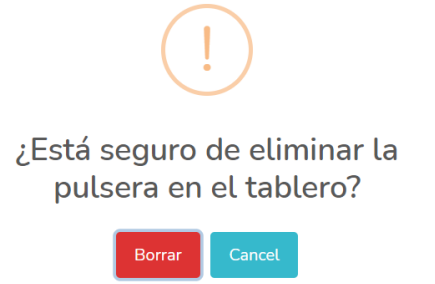


Ilustración 42: Pop up confirmativo de eliminación

Por detrás de esta página, se utiliza Firebase Realtime Database para poder leer las variables e información del usuario de cada pulsera, y así poder mostrarlos. En este caso el código esta guardado debajo de cada pulsera, y se tiene una variable llamada show, que será modificada si se desea mostrar o no la pulsera en el tablero, intercambiando entre sí o no en esta variable.

8.3.1.5. Configuración

Las personas al ser distintas entre sí, tienen valores vitales distintos a los estándares, es por esto que se optó en generar una página que pueda dar solución a este punto.

Al seleccionar una pulsera, se mostrará la configuración del paciente que este cargada a ese momento, pudiendo ver los valores de fábrica. En la parte inferior se verá distintos rectángulos que permiten el ingreso de valores para así modificar la información personal que se divide en dos sectores, nombre, habitación y numero de contacto; y por otro lado la frecuencia cardiaca máxima y mínima, la saturación de oxígeno máxima y mínima, y la temperatura máxima y mínima que esa persona pueda tener distinta a la seteada en los valores de fábrica. Estos datos seteados de forma predeterminada fueron consultados a una médica clínica que nos indicó valores normales, y valores para los cuales generar una alerta, aunque siempre nos resaltó que estos valores pueden variar según el paciente.

Toda esta información se mostrará en los recuadros respectivamente a su información.

El botón ubicado en la esquina superior izquierda será utilizado en el caso de querer cambiar la pulsera de usuario, poniendo la misma en modo de fabrica (ver sección 8.3.1.7).

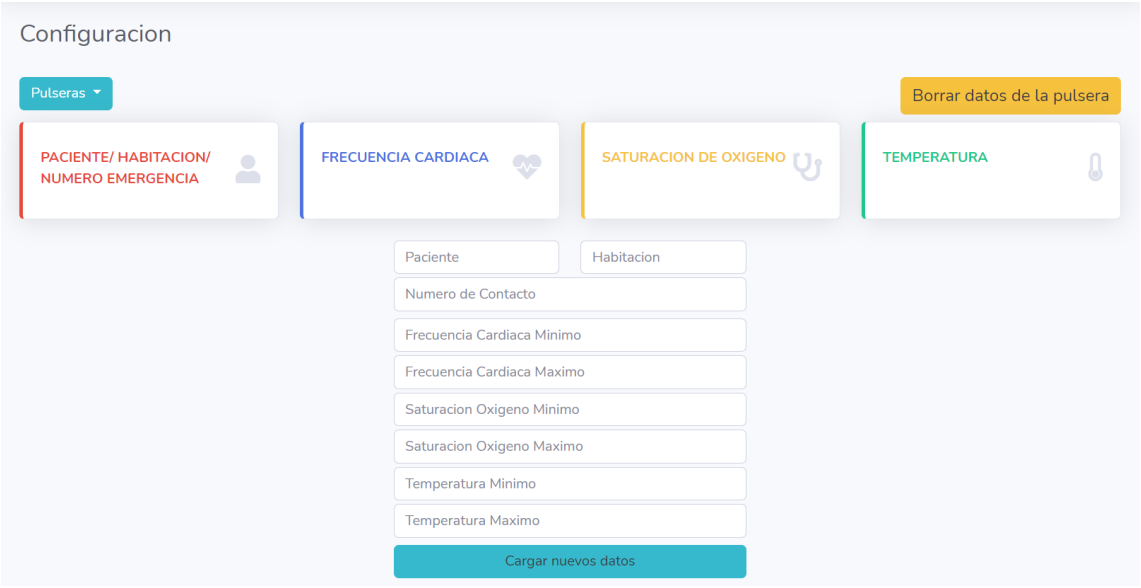


Ilustración 43: Página de configuración

Para mostrar esta información, es necesario guardarla en la sección de Firebase denominada Realtime Database, que gracias a su formato de descendientes permite relacionar estos datos a cada pulsera. Este servicio nos permite actualizar la información en momento real y mostrarlo en este sector.

8.3.1.6. Ayuda

A la hora de utilizar la página web es necesario poner ciertas normas y estándares como punto de partida, para esto se utiliza esta sección.

En la primera parte se muestra cómo utilizar cada una de las secciones. La parte posterior muestra, de forma ilustrativa, los números de contacto dentro de la institución que serán usados en caso de emergencia, y llegado al caso se dispone de números de contacto y correo de la empresa proveedora del servicio.

8.3.1.7. Borrado de la pulsera

Este botón borrara la información del usuario y sus mediciones, seteando la pulsera a valores de fábrica.

Los datos cargados inicialmente son los siguientes:

- Pulsaciones máximas – 120
- Pulsaciones mínimas – 50
- Saturación de oxígeno máxima – 100%
- Saturación de oxígeno mínima – 89%
- Temperatura máxima – 38°
- Temperatura mínima – 34°

Se borrará el nombre del usuario, la habitación y su número de contacto; y el código del par de las pulseras se restablecerá a uno nuevo único, descartando el anterior para que no se puede utilizar más, con el siguiente formato “xxxxxxxx-xxxx-4xxx-yxxx-xxxxxxxxxxxx” donde las x e y serán reemplazadas por un carácter alfanumérico aleatorio.

Este botón debe ser presionado en caso de cambiar el usuario de las pulseras, donde luego se carga en función de sus datos personales los valores de referencia.

8.3.1.8. Envío de correos

La aplicación web cuenta de dos envíos de mails, el primero es para informar el código de ingreso al usuario para ver sus datos y el segundo es para enviar el histórico al mail cargado.

Para enviar estos correos se utiliza una librería de JavaScript denominada stmp.js que usa un servidor STMP, donde gracias a un protocolo de transmisión simple se permite el

intercambio de mensajes entre dispositivos. Estos mails serán enviados mediate un correo electrónico creado con el nombre de healthbandaso@gmail.com.

Ambos mails están seteados para cargar la información correspondiente según sea el caso, para el correo que envía el código se utiliza un template que carga el asunto y el texto del cuerpo dejando un sector para escribir el código (ver Ilustración 44Ilustración 44). Por otro lado, el mail con los datos históricos cargará su template y escribirá en el cuerpo la tabla designada (ver Ilustración 45).

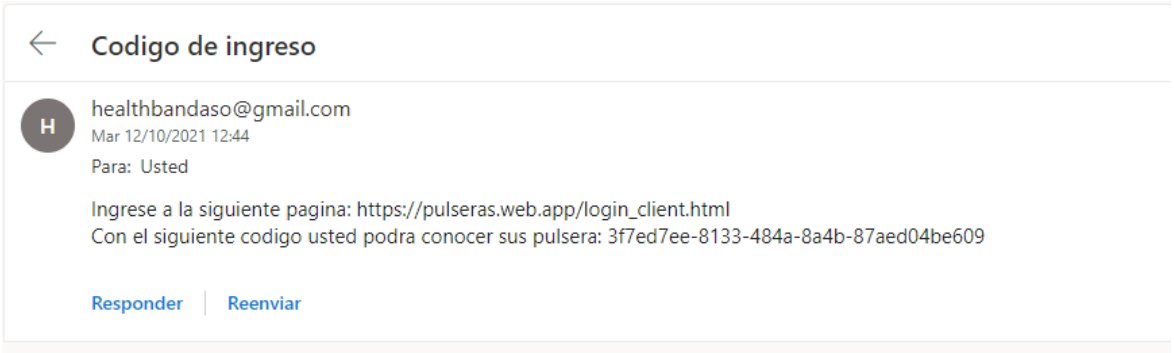


Ilustración 44: Mail enviado al usuario

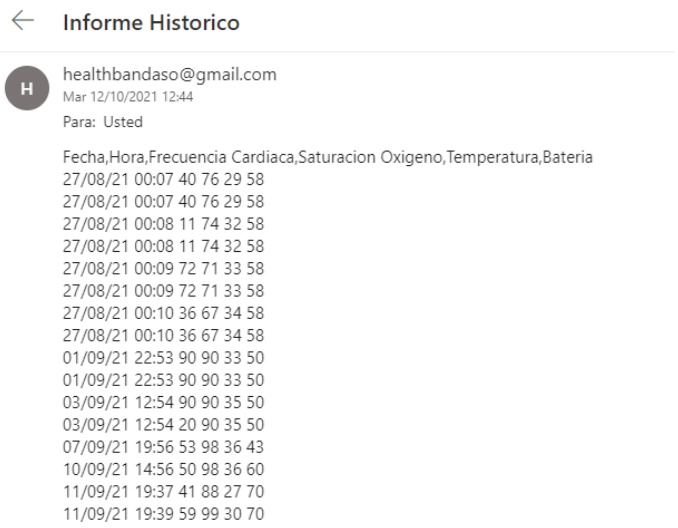


Ilustración 45: Mail con información histórica

8.3.1.9. Sección común a todas las paginas

Toda la página web cuenta con una barra lateral ubicada a la izquierda de la pantalla que sirve para poder moverse a las distintas secciones de la aplicación (ver Ilustración 46), también podrá ser reducida (ver Ilustración 47) para casos donde se quiera tener el tablero o la información en mayor sector de la pantalla.

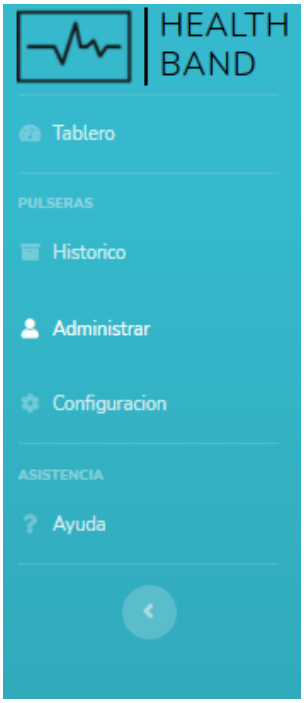


Ilustración 46: Barra lateral común

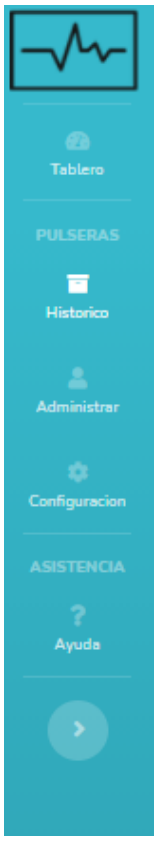


Ilustración 47: Barra lateral común comprimida

8.3.2. Página del usuario

Esta página está pensada para separar al dueño de la página y al usuario, donde la persona que tiene la pulsera tiene un acceso limitado a los datos debido a que no necesita conocer valores médicos de otras personas, ni sus referencias.

Se separa en dos secciones, el logeo mediante el código que le brindo el dueño y el tablero donde están sus datos.

8.3.2.1. Login

Esta página está diseñada para que el cliente sea capaz de ingresar a sus datos actuales mediante el código enviado a su casilla de mail. Esta página no cuenta con autenticado mediante Firebase, es por esto que utiliza el código de las pulseras.

Una vez ingresado el código deberá de presionar el botón “Submit” para así proceder a la otra página.

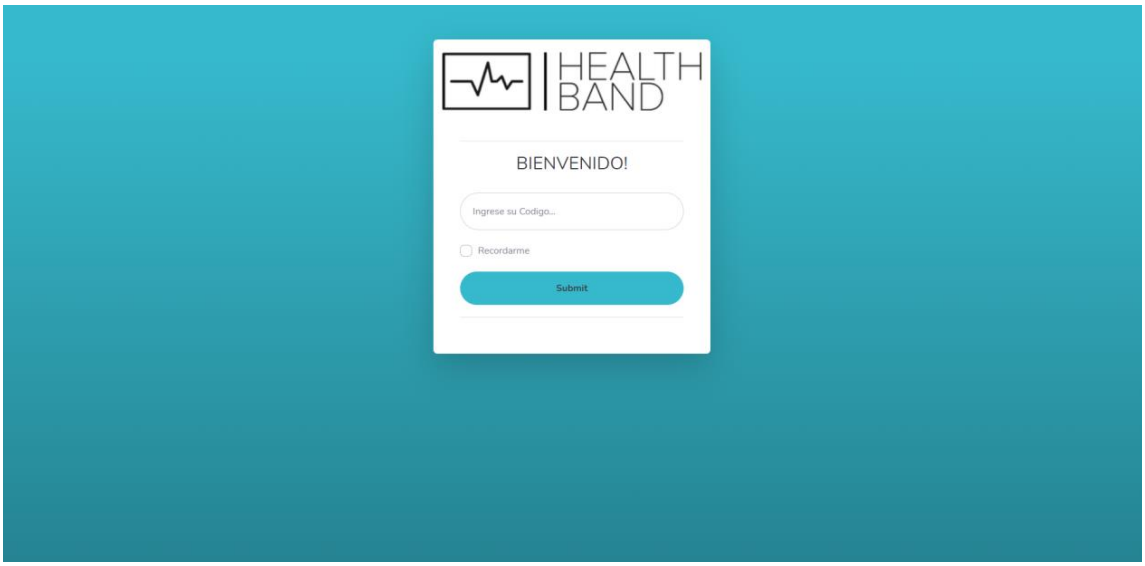


Ilustración 48: Login del usuario

8.3.2.2. Tablero

Una vez ingresado al tablero, podrá tener una vista similar al tablero del dueño, pero solo será capaz de ver los datos de sus pulseras, como así la cantidad, los estados, las alertas y las emergencias. Las alertas y emergencias responderán de la misma forma que para el tablero del dueño (ver sección 8.3.1.2).

Gracias a la utilización del código enviado por mail, la página es capaz de encontrar los valores correspondientes en la base de datos. Donde cargara la referencia del usuario, la última medición, el estado de cada pulsera y el nombre.

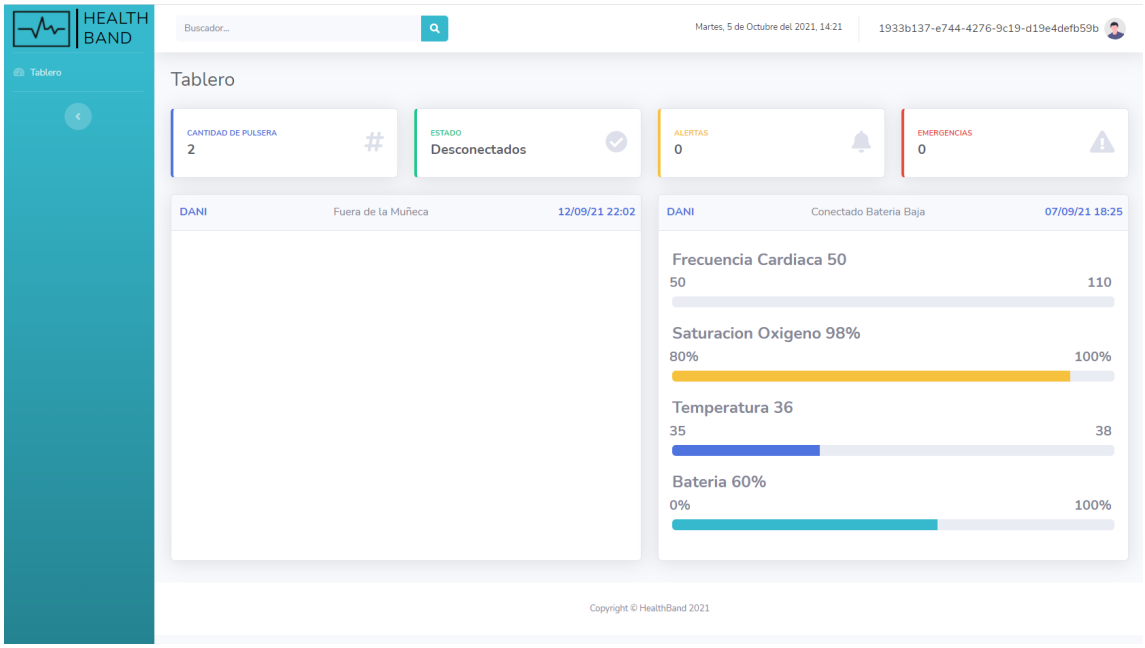


Ilustración 49: Tablero del usuario

9. Análisis económico

Nuestro proyecto busca crear una empresa denominada HealthBand ubicada en Capital Federal, Argentina, para dar soporte a centros médicos como hospitales o geriátricos donde haya personas que no son monitoreadas debido a la cantidad de equipos médicos. Esta cantidad es reciproca al precio de los mismos, por eso nace nuestra iniciativa de crear un producto de bajo costo y competitivo para introducirse en este sector sumamente limitado.

Nuestra misión es brindar un producto altamente competitivo debido a su fácil composición y características. La visión es poder abarcar el sector médico y reducir la cantidad de personas que no son controladas mediante equipos.

Por parte de los objetivos, buscamos cumplirlos mediante la utilización de objetivos SMART. Apuntamos a cubrir el 20% del mercado hospitalario y geriátrico de la capital, donde en cada uno se destine un dispositivo a personas sin control alguno, en un plazo de 5 años.

La empresa está pensada en que armará el producto una vez que consigue el contrato con las instituciones médicas, para tener el beneficio de no disponer de productos sin sentido que al pasar el tiempo pierde valor debido a la desactualización de componentes. Una vez cerrado el contrato se le cobrara un porcentaje del monto final para poder comenzar la fabricación.

Cabe aclarar que los montos utilizados fueron obtenidos para septiembre de 2021.

9.1. Precio unitario pulsera

En este cálculo tomamos como referencia todos los componentes electrónicos necesarios para obtener la pulsera en su totalidad, también la materia prima como filamentos para imprimirla y la energía necesaria para armarla.

Costo de produccion	Cantidad	Unidades	Precio unitario (ARS)	Costos (ARS)
Filamento PLA y TPU	102	gr	$ 1,80	$ 183,60
Microcontrolador Wemos	2	un	$ 710,00	$ 1.420,00
Sensor MAX30102	2	un	$ 1.000,00	$ 2.000,00
Cables Wire Wrap	1	metros	$ 150,00	$ 150,00
Cargador tp4056	2	un	$ 450,00	$ 900,00
Baterias BL 5J NOKIA	4	un	$ 400,00	$ 1.600,00
Pulsadores	4	un	$ 70,00	$ 280,00
Resistencias	10	un	$ 5,00	$ 50,00
Capacitores	2	un	$ 10,00	$ 20,00
Energia electrica	4	kw	$ 12,00	$ 48,00
Cargador 5v	1	un	$ 600,00	$ 600,00
Elastico sensor	0,5	metros	$ 100,00	$ 50,00
Mano de obra	0		Propio	$ -
			TOTAL	$ 7.301,60

Tabla 1: costo unitario por 2 pulseras

Estas pulseras son vendidas por 2 unidades, ya que el desarrollo fue pensado para nunca perder registro de las personas habiendo otra unidad que pueda reemplazarla en caso de batería baja

9.2. Inversión inicial

Para poder armar esta empresa es necesario contar con activos tangibles o también conocidas como herramientas, que nos permitirían fabricar las pulseras.

Activos tangibles	Precio (ARS)
Impresora 3D (ender 3-pro)	$ 50.000,00
Soldador de estaño	$ 1.000,00
Computadoras X2	$ 160.000,00
TOTAL	$ 211.000,00

Tabla 2: activos tangibles

Por otro lado, tenemos los activos intangibles que para nosotros son nulos ya que todo parte de un desarrollo propio.

Activos intangibles	Precio (ARS)
Diseño 3D de la pulsera	$ -
Firmware de la pulsera	$ -
Desarrollo web	$ -
TOTAL	$ -

Tabla 3: activos intangibles

El desembolso inicial que necesitamos para poder realizar las tareas y así la fabricación es de $211000 pesos que solo serán de los activos tangibles. La inversión será

financiada por fondos propios sin la necesidad de acceder a préstamos de dinero, los cuales incrementarían nuestro monto final.

Es necesario disponer de este dinero porque nos permitirá comenzar con el funcionamiento de la empresa y así poder comercializar el producto final.

9.3. Lean CANVAS

Esta tabla se utiliza generalmente en empresas chicas o también llamadas startup, donde es necesario conocer obtener una visualización del problema y la solución que se brindara.

<table>
<tr><th>Problema</th><th>Solución</th><th>Proposición de valor única</th><th>Ventaja especial</th><th>Segmentos de clientes</th></tr>
<tr><td rowspan="3">• Alto precio de máquinas médicas.
• Pacientes sin control.
• Contacto con pacientes.</td><td>• Pulsera económica.
• Interfaz intuitiva.
• Alertas de pacientes.</td><td rowspan="3">Pulsera de bajo costo con interfaz de control.</td><td>• Competitividad del precio de la pulsera.
• Centralización de datos en la nube.
• Visualización de datos en la web.</td><td rowspan="3">Centros médicos como hospitales o geriátricos que buscan controlar pacientes sin control.</td></tr>
<tr><td>Métricas clave</td><td>Canales</td></tr>
<tr><td>• Pulseras vendidas.
• Tiempo de fabricación.</td><td>Uso de visitadores médicos que recorran los centros médicos.</td></tr>
<tr><th colspan="3">Estructura de costes</th><th colspan="2">Flujo de ingresos</th></tr>
<tr><td colspan="3">Costos mínimos, inversión inicial relativamente baja.</td><td colspan="2">Beneficio en margen calculado en las pulseras.</td></tr>
</table>

Tabla 4: análisis Lean Canvas

9.4. PESTEL

Utilizamos este análisis para poder conocer un poco más sobre el entorno donde ser vería afligido esta empresa, aspectos como políticos, económicos, sociales, tecnológicos, ecológicos y legales.

Político	Económico	Social	Tecnológico	Ecológico	Legal
Elecciones de medio termino como presidenciales	Inflación Comportamiento del dólar Importaciones	Contacto con el COVID-19	Avance ultra rápido de componentes	Cambio de pensamiento de la sociedad con las baterías	Regulación sobre maquinaria medica

Tabla 5: análisis PESTEL

9.5. FODA

En este apartado queremos obtener un análisis FODA para poder conocer la situación de la empresa, viendo cuales son nuestras fuerzas internas y externas, como también cuales son los puntos en contra que tenemos dentro de la institución y fuera.

Fortalezas	Debilidades
• Fácil obtención de recursos materiales y producto final • Automatización al agregar nuevas pulseras • Bajo costo de fabricación	• Tiempo de producción de pulseras • Dependencia de Firebase • Limitación del tiempo de la batería • Dependencia de WIFI
Oportunidades	**Amenazas**
• Maquinas medicas de control demasiado caras • Compatibilidad de sistema móvil y web • Seguridad de los datos del usuario	• Competencia de pulseras de origen chino • Aumento de precios por dependencia al dólar • Leyes de herramientas medicas

Tabla 6: análisis FODA

9.6. Análisis 4P

Este análisis nos sirve para conocer cómo se establecerá el producto en el mercado y así su precio.

9.6.1. Producto

Nuestro producto es un par de pulseras que busca satisfacer el mercado medico en los pacientes que no tiene control debido al costo de los aparatos de primeras marcas. Buscamos conocer el estado de los pacientes mediante un sitio web que brinde alertas en el caso ser necesario.

Este producto dispone de una alerta simultanea donde el paciente puede accionarla y la persona a cargo sabrá de esto en cuestión de segundos. Por parte del personal, podrá saber cómo se comportaron los valores del paciente mediante la página.

9.6.2. Precio

Buscamos reducir el precio al máximo, para así cumplir con uno de los principios y ser competitivos en el mercado. Es por esto que haremos un servicio personalizado donde primero buscaremos aprovechar el servicio gratuito brindado por Firebase.

En cambio, el precio de las pulseras fue optimizado al máximo y al tener una dependencia de componentes importados estamos limitados en su variación.

9.6.3. Punto de venta

El punto de venta de este producto es a pedido donde previamente se acordó la cantidad para el sitio. Es por eso que se reduce los costes de almacenamiento e inversión. En cambio, los costes de envío serán considerados a la hora de hacer la compra por parte de la empresa.

Los canales de distribución serán mediante venta directa donde previamente el visitador medico habrá presentado el producto.

9.6.4. Promoción

Uno de los papeles más importantes lo tiene el visitador médico, que se encargará de recorrer los sitios para presentar el producto y comentar de las soluciones que brinda a un costo muy competitivo para este mercado.

9.7. Estigmatización de la demanda

Si nos ponemos como objetivo abastecer un 20% de los geriátricos y hospitales de Capital Federal en los próximos 5 años, donde en total hay unos 700 establecimientos aproximadamente, buscaremos abarcar a 140 sitios. Suponiendo una demanda aproximada de 20 pares de pulseras (ya que 80% de los 140 establecimientos son geriátricos con un promedio de 15 personas cada uno) por cada establecimiento, dando un total de 2800 pares de pulseras, es decir 5600 pulseras en total. Dándonos una producción aproximada de 4 pulseras por día hábil, siendo la impresora nuestro cuello de botella.

Establecimientos en CABA	Cantidad
Hospitales Publicos	65
Hospitales Privados	35
Geriatricos	600
TOTAL	700

Tabla 7: establecimientos en CABA

9.8. Caso de análisis

Para este análisis tomamos como referencia el Hospital Municipal de Pigüé, que cuenta con 50 personas en terapia sin control médico mediante el uso de dispositivos clínicos, donde le brindaremos 100 pulseras en total. Como ya mostramos anteriormente el precio del producto, que equivale a 2 pulseras, es de $7301.60. Si multiplicamos por 50 obtenemos un costo total de materias primas de $365080. A este valor solo nos quedaría sumarle el precio de las actualizaciones vía OTA.

El servicio de OTADrive para 100 dispositivos tiene un valor de 100usd por año, suponiendo que la pulsera debido a su batería va a tener una vida útil aproximada de 4 años, vamos a considerar el costo de pagar 4 años el servicio de actualizaciones. Al día de hoy el valor del dólar unitario más impuestos es de $173.25 (13/10/2021), por lo tanto, 100usd por 4 años por $173.25, nos da un valor de $69300 que se le agregarían al costo final. Dándonos así el resultado final de $434380 solo de costos para poder abastecer a este hospital.

Si dividimos este valor por 50 obtenemos que cada par de pulseras tiene un costo de $8687.60, tomando un valor aproximado de packaging de $500, este número se eleva a $9187.60. Ahora bien, si suponemos un precio total de venta por el par de $16000, obtendríamos una ganancia de $6812.40.

Podríamos decir que el costo del par de pulseras para un hospital es relativamente bajo, teniendo en cuenta que solo el hospital de Pigüé gastó más de $300000 en hisopados de COVID-19 en lo que va del año (13/10/2021). Podríamos analizar que el costo de cada par de pulseras equivale aproximadamente a 16 hisopados, y cuando hablamos de la pulsera estamos hablando de una inversión de 4 años.

10.Metodología de desarrollo

Para llevar a cabo este proyecto, lo primero que se hizo fue hablar con un médico clínico, donde nos ayudó a establecer un claro objetivo, sobre que producto hacer y a que público apuntar, que funciones debía y no debía tener tanto la pulsera como la página web.

Una vez fijado el objetivo lo primero que se hizo fue elegir los componentes electrónicos, donde lo único que fue cambiando, fue la batería hasta encontrar la del tamaño y almacenaje correcto, y que mejor se adapte al proyecto. En cuanto al sensor, el modelo siempre fue el mismo, lo que se tuvo que cambiar fue la posición, nuestra idea principal fue colocar el sensor sobre la muñeca, lo cual después de varios intentos y pruebas tuvimos que desistir colocando el sensor en el dedo. A medida que terminábamos con la pulsera, se comenzó con la página web, primero elegimos donde íbamos a hostear la página y que base de datos usar. Luego sincronizamos esta última con la pulsera. A partir de este momento solo quedaba elegir como y que íbamos a mostrar en la web. Como íbamos a manejar las distintas pulseras y usuarios. Como mostrar los datos y como configurar las pulseras.

Finalmente obtuvimos una pulsera confiable con 3 o 4 días de duración de batería y una página web donde se pueden visualizar en tiempo real los valores actuales de cada pulsera, un histórico de cada paciente, los límites de pulsaciones por minutos, oxígeno en sangre y temperatura corporal con sus respectivas alarmas, una pestaña con ayudas, y hasta la posibilidad de tener una página independiente para el usuario donde podrá visualizar solamente los datos de su pulsera en tiempo real.

11. Pruebas realizadas

Debido al tiempo disponible para la realización del proyecto final y a las condiciones dadas en 2021 en consecuencia a la pandemia mundial COVID-19 las pulseras y la página web fueron probadas por nosotros mismos. Ambas se probaron minuciosamente durante todo el desarrollo, y una vez finalizada la pulsera y la página web se probaron en conjunto, generando todas las posibles alertas y situaciones. Todo se encuentra 100% funcional, todas las funciones y opciones de la página web se encuentran activas y realizando lo que dicen hacer, y en cuanto a la pulsera se compararon sus mediciones con un oxímetro y pulsímetro aprobado por la ANMAT y estas eran similares o iguales. Lamentablemente no conseguimos una estabilidad robusta en cuanto a la posición del sensor en el dedo o a los movimientos de muñeca o mano. Si el paciente mueve mucho la mano o hace esfuerzos las mediciones empiezan a no ser 100% fiables. Pero a pesar de ello se realizó una prueba de 8 horas donde se obtuvieron resultados muy buenos para la pulsera, y en cuanto a la página nada para objetar, no dio ninguna falla, y se pudieron visualizar los resultados correctamente en todo momento. Entre medio se cambió la pulsera por la pareja y se obtuvo un histórico correcto sumando ambas mediciones.

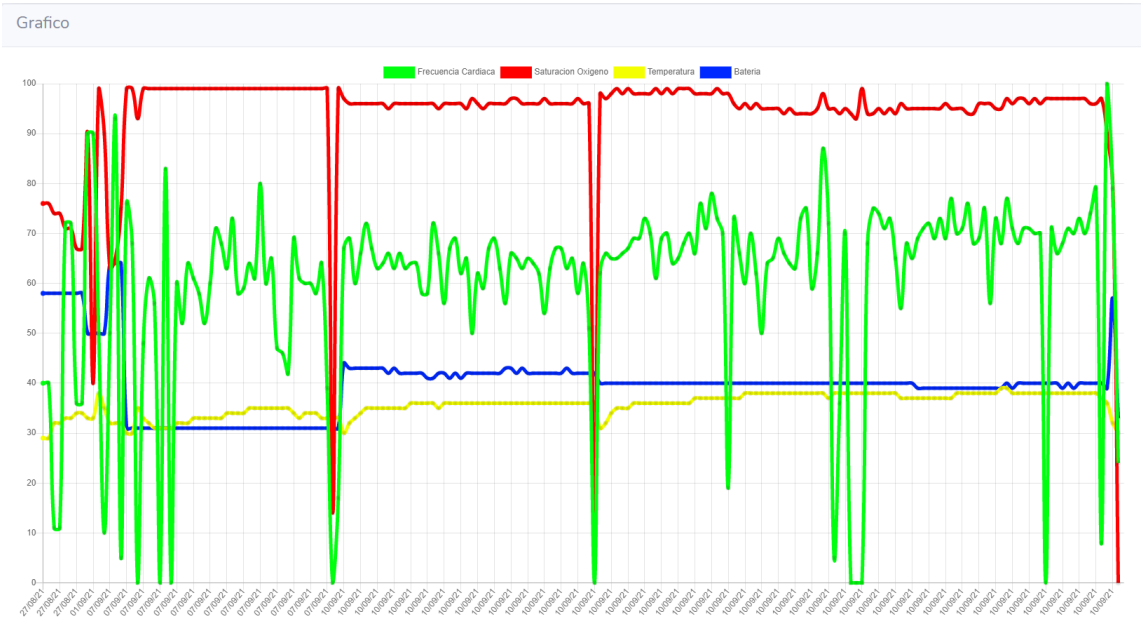


Ilustración 50: Grafico comportamiento histórico

El grafico obtenido en la página web muestra cómo se comportó en esas 8 horas el par de pulseras, la parte de la izquierda fueron algo inestables debido a unas pruebas realizadas, pero una vez colocada la pulsera en su posición correcta se lograron muy buenos resultados,

solo dando algunos picos hacia abajo en algunos casos en los que se movía la mano o se corría de posición el sensor.

12.Conclusión

Al pasar de los días, fuimos corrigiendo lo que en principio fue nuestra idea a llevar a cabo, porque considerábamos que había aspectos que hoy en día no eran considerados con el mérito necesario.

Uno de los aspectos más importantes que buscamos preservar fue el precio del producto y su fácil elaboración porque vimos como las herramientas medicas durante la pandemia tenían un fuerte requerimiento y un difícil acceso.

Como resultado podemos decir que obtuvimos todos los objetivos que planteamos inicialmente. Desarrollamos un dispositivo que nos permite conocer las pulsaciones por minuto, la saturación de oxígeno en sangre y la temperatura corporal de cada persona. Luego, sincronizamos esta información con una base de datos ubicada en la nube con un costo relativamente nulo. A su vez, diseñamos y trabajamos en una aplicación web que es capaz de obtener esta información y mostrarla mediante lógica computacional para que el usuario obtenga una retroalimentación correcta. Esta página está pensada para ser lo más ordenada a la visión posible, considerando situaciones de emergencias y alertas que el paciente pudiera llegar a tener.

Mediante la utilización de la pulsera durante días fuimos capaces de corregir los errores de diseño y optimizar las mediciones, como la ubicación del sensor, la cantidad de baterías y su tamaño. Optamos por usar servicios tercerizados, como OTAdrive, para poder actualizar las pulseras de forma remota, o diseñar el firmware para que los dispositivos sean capaces de generar su propia red WIFI y así poder configurarlos. Un punto muy importante fue la adaptación de los estados porque nos evitó el consumo innecesario de almacenamiento, transferencia de datos y batería.

El servicio de la nube, Firebase, también fue cambiando acorde nuestros avances y descubrimientos, donde mejoramos la disposición de la información para que la página y la pulsera se complementen entre sí, lo mejor posible. La autenticación también varió desde el inicio porque optamos por limitar las cuentas solo a las personas autorizadas, que serían las mínimas posibles.

Por último, nuestra página web, vario fuertemente, donde al inicio teníamos una página sin control, ni seguridad. La fuimos mejorando para obtener una página que sea capaz de proteger la información del usuario, como así sus mediciones. Por otra parte, notamos la

necesidad de crear una sección en paralelo que permita al paciente obtener sus valores actuales, sin necesidad de acceder a toda información de Firebase. Además, el histórico de la página tiene mucha importancia porque nos retroalimenta con el comportamiento de las variables de las personas durante el tiempo, gracias al historial y al gráfico.

Nuestro caso de análisis nos dio mucha información de la situación de los hospitales durante la pandemia COVID-19, vimos cómo había variables que eran muy importante y era necesaria su medición.

En síntesis, se logró un producto IOT muy útil para los nosocomios; económico y con grandes prestaciones, con un gran desarrollo web capas de visualizar el estado actual de varias pulseras al mismo tiempo y obtener un histórico de las mismas.

13.Bibliografía

HAVERBEKE, Marijn. Eloquent javascript: a modern introduction to programming. 3a ed. 2019.

OUYANG, Erwin. Hands-On ESP8266: mastering basic peripherals. HandsOnEmbedded, 2018.

SERPANOS, Dimitrios y Marilyn WOLF. Internet-of-Things (iot) systems. Springer, 2018.

WILLIAM, Cristancho Gómez. Oxígeno: Fisiología, terapéutica, toxicidad. Bogotá: Manual Moderno, 2019.

DOCUMENTACION ARDUINO. [en línea]. [consulta agosto 2021]<https://es.wikipedia.org/wiki/Arduino_IDE>

DOCUMENTACION CSS. [en línea]. [consulta jul. 2021]<https://developer.mozilla.org/es/docs/Web/CSS>

DOCUMENTACION ESP8266. [en línea]. [consulta jun 2021]< https://naylampmechatronics.com/blog/56_usando-esp8266-con-el-ide-de-arduino.html>

DOCUMENTACION FIREBASE. [en línea]. [consulta ago. 2021]<https://www.digital55.com/desarrollo-tecnologia/que-es-firebase-funcionalidades-ventajas-conclusiones/ >

DOCUMENTACION FIREBASE. [en línea]. [consulta jul. 2021]<https://firebase.google.com/docs>

DOCUMENTACION HTML. [en línea]. [consulta jul. 2021]<https://developer.mozilla.org/es/docs/Web/HTML>

DOCUMENTACION HTML5. [en línea]. [consulta jul. 2021]<https://developer.mozilla.org/es/docs/Glossary/HTML5>

DOCUMENTACION IMPRESORAS 3D. [en línea]. [consulta ago. 2021]<https://es.wikipedia.org/wiki/Impresora_3D>

DOCUMENTACION IOT. [en línea]. [consulta agosto 2021]<https://www2.deloitte.com/es/es/pages/technology/articles/IoT-internet-of-things.html>

DOCUMENTACION MAX30102. [en línea]. [consulta ago. 2021]<https://www.nubbeo.com.ar/productos/sensor-de-pulso-cardiaco-y-oxigeno-max30102-arduino-nubbeo/?gclid=Cj0KCQjw1dGJBhD4ARIsANb6OdlPjsuL6EKxeLTxgHEGh3_RMYNlsZFKVM9xDGqdshTfd2Iikk8qlCQaAtkEEALw_wcB>

DOCUMENTACION MICROCONTROLADORES. [en línea]. [consulta agosto 2021]<https://www.arrow.com/en/research-and-events/articles/engineering-basics-what-is-a-microcontroller>

DOCUMENTACION MICROCONTROLADORES. [en línea]. [consulta agosto 2021]<https://es.wikipedia.org/wiki/Microcontrolador>

DOCUMENTACION OTA. [en línea]. [consulta ago. 2021]<https://androidayuda.com/android/que-es/ota/ >

DOCUMENTACION VISUAL STUDIO CODE. [en línea]. [consulta jul. 2021]<https://code.visualstudio.com/>

14. Anexos

14.1. Fotos de la pulsera

Ilustración 51: Pulsera en funcionamiento

Ilustración 52: Pulsera en funcionamiento

Ilustración 53: Pulsera en funcionamiento

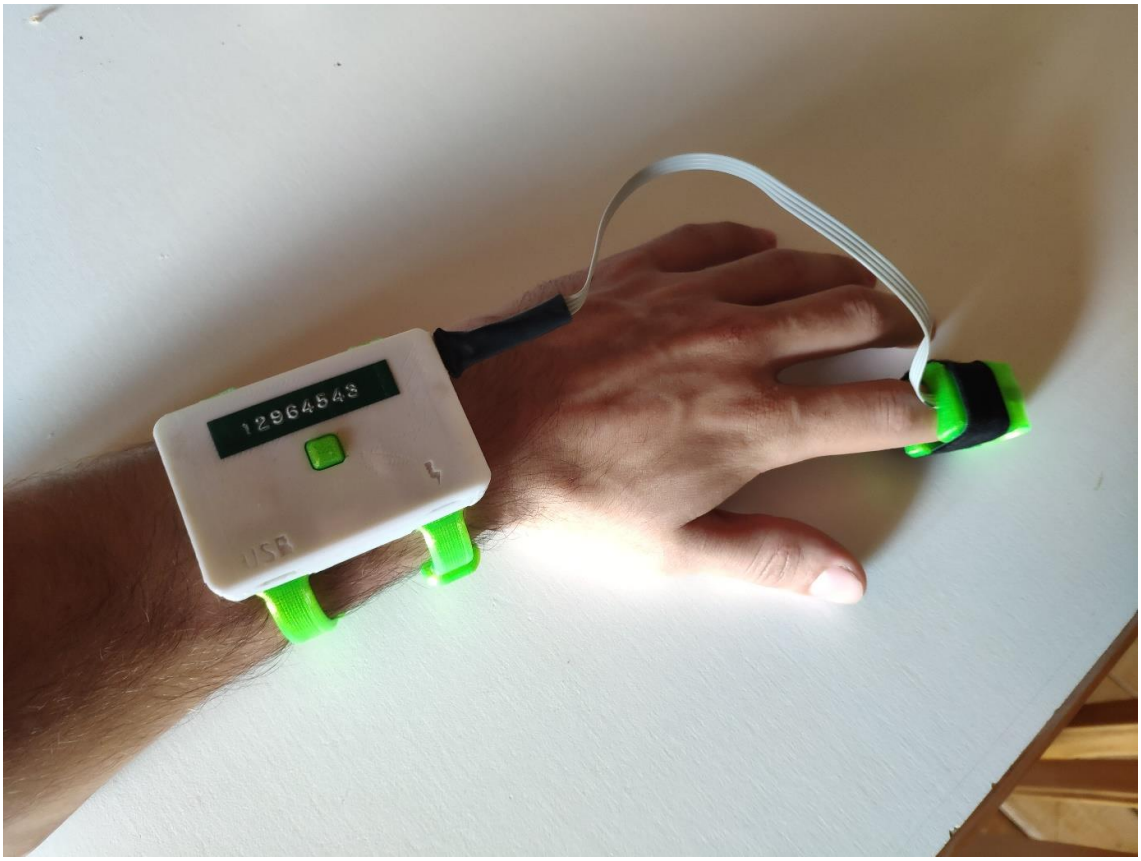


Ilustración 54: Pulsera en funcionamiento

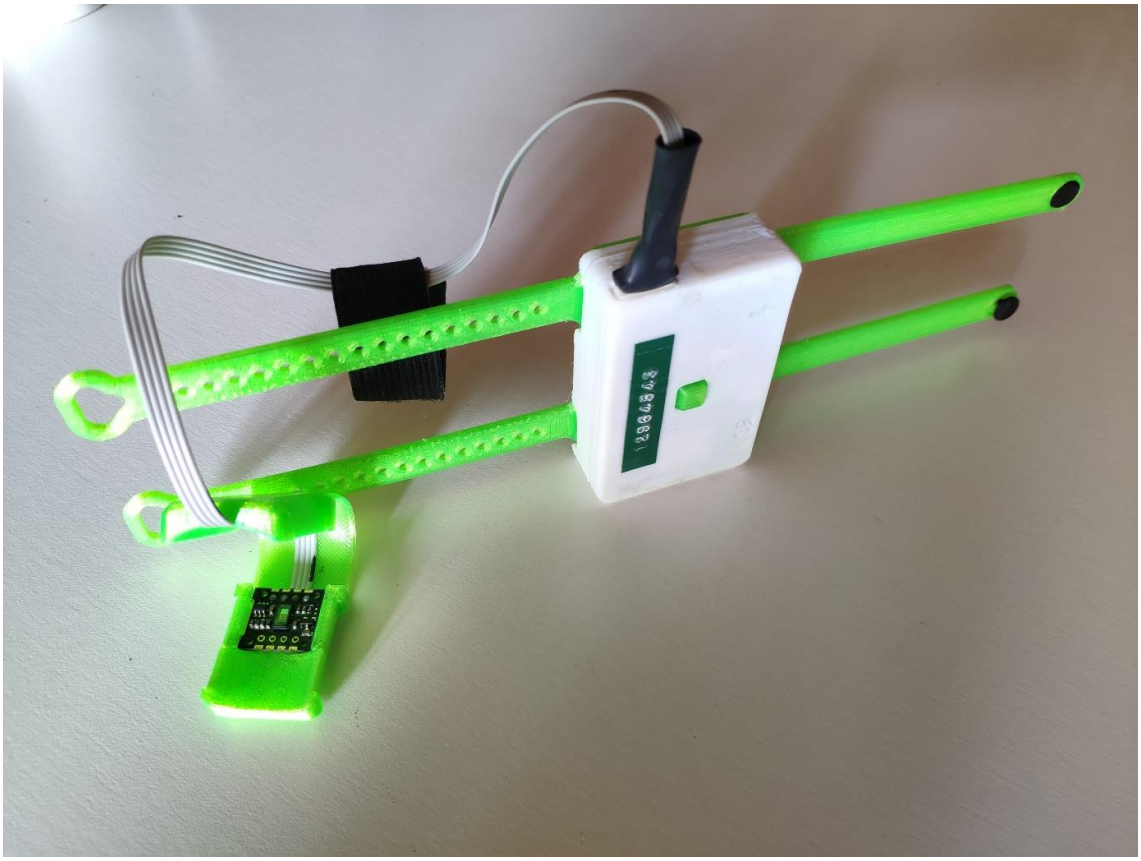


Ilustración 55: Pulsera y sensor

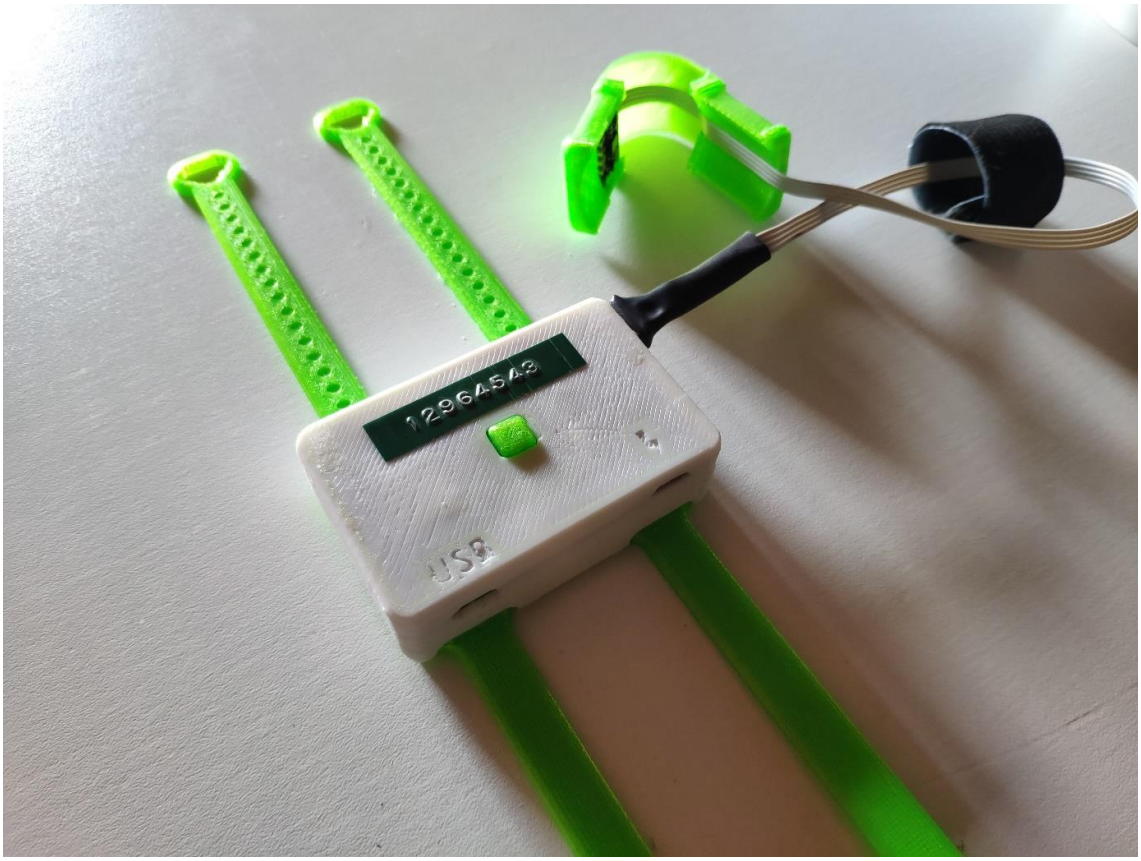


Ilustración 56: Pulsera y sensor

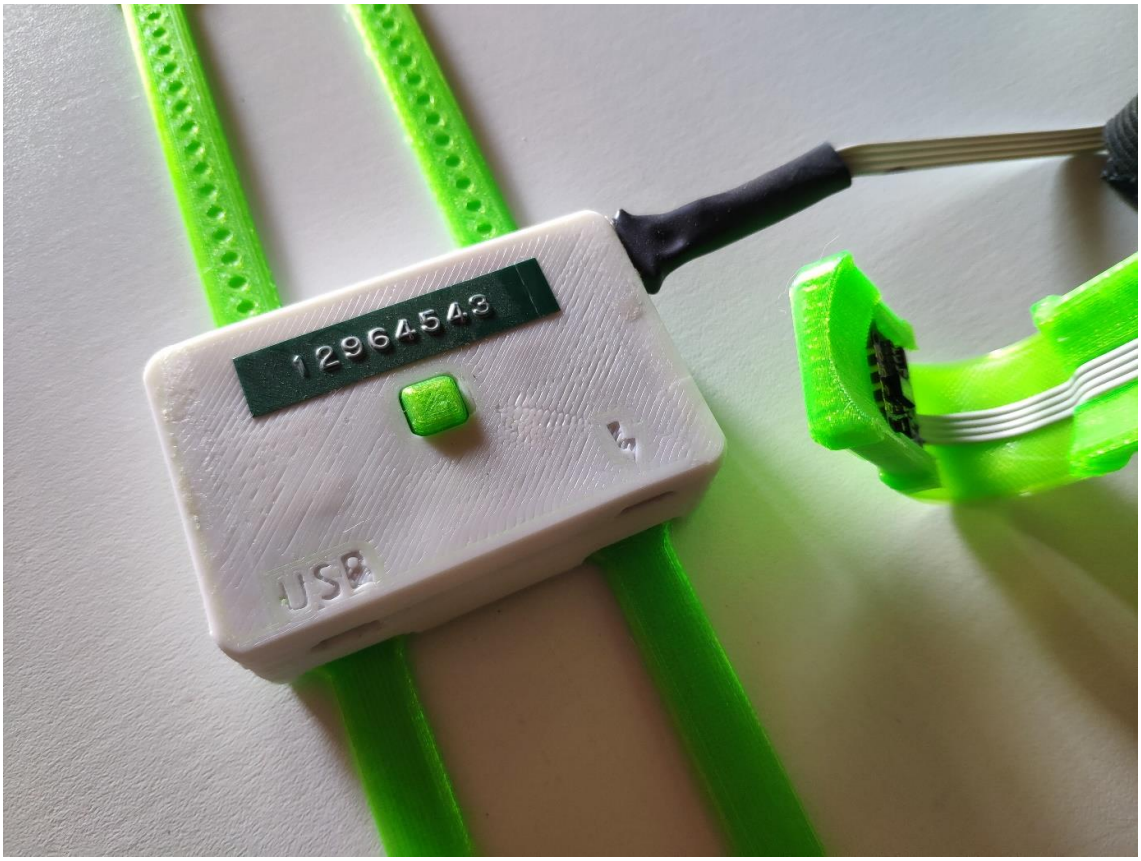


Ilustración 57: Pulsera y sensor

Ilustración 58: Pulsera y sensor

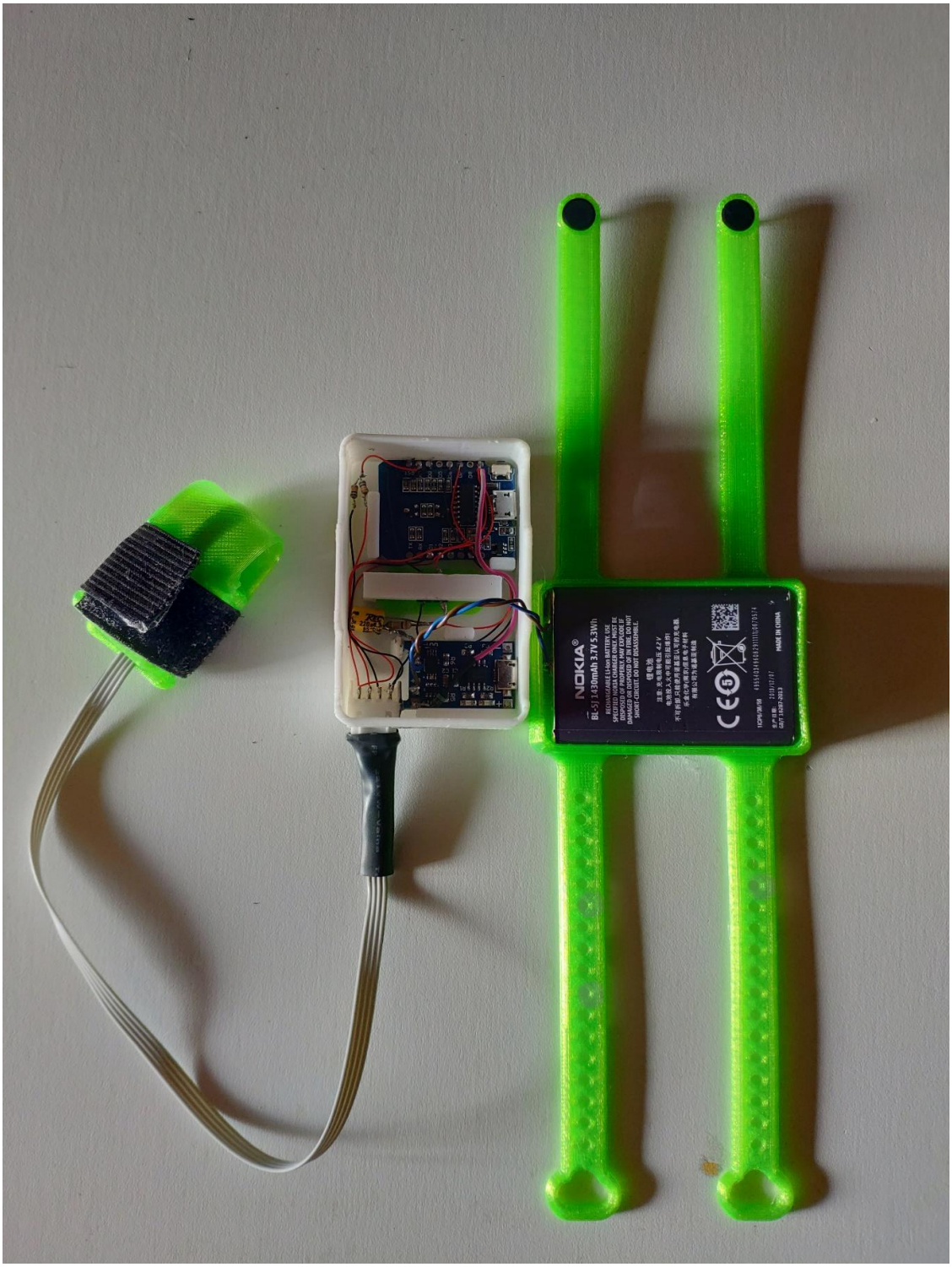


Ilustración 59: Interior de la pulsera

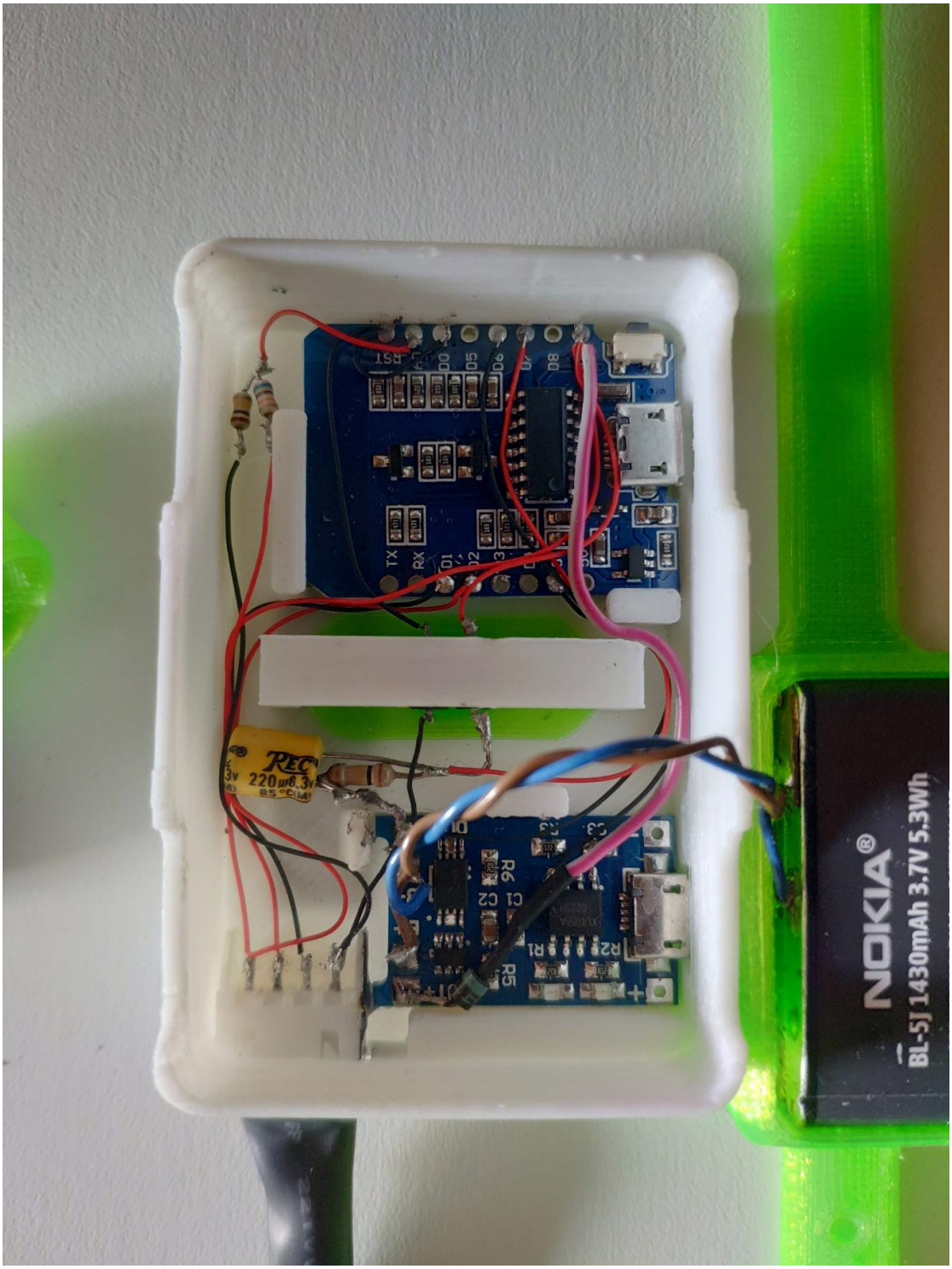


Ilustración 60: Componentes electrónicos de la pulsera

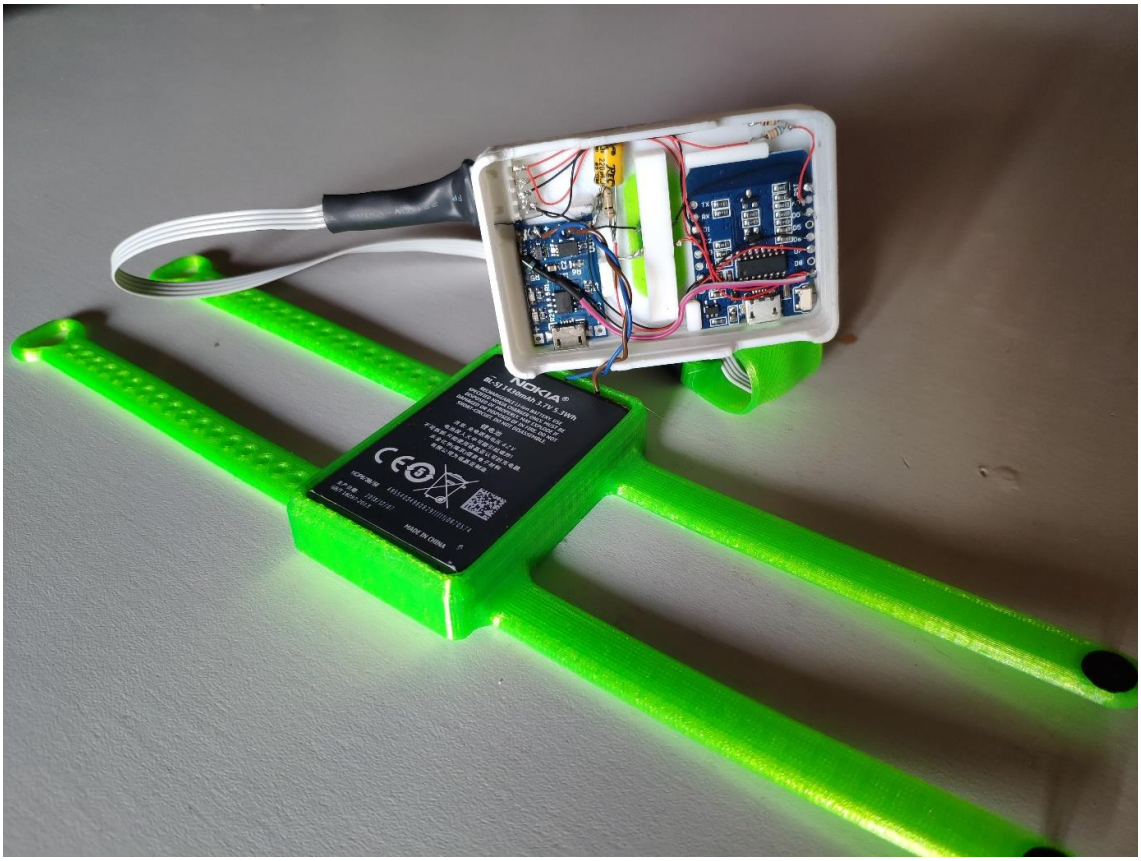


Ilustración 61: Interior de la pulsera

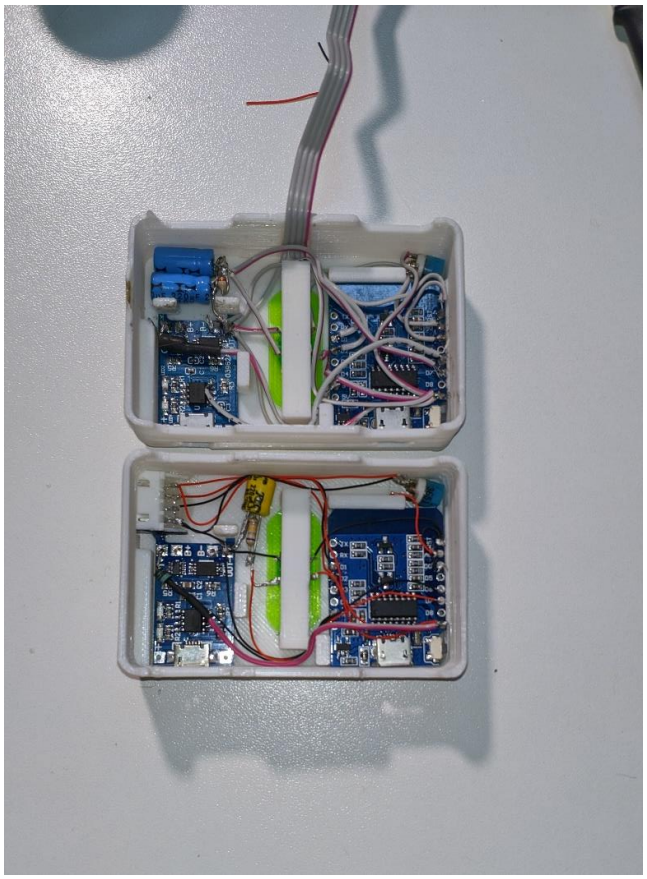


Ilustración 62: Comparación de electrónica (arriba primera versión, abajo segunda versión)

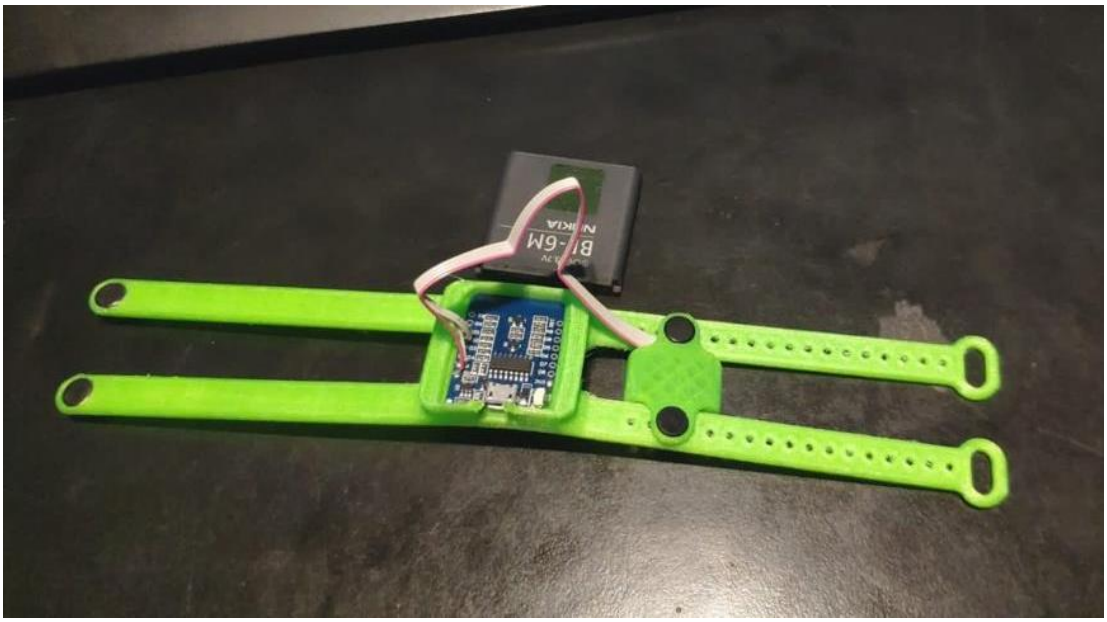


Ilustración 63: Primer modelo de pulsera

14.2. Código Pulsera Arduino

```
/*--------------DEFINE WIFI----------------*/
#include <WiFiManager.h>
WiFiManager wm; // global wm instance
WiFiManagerParameter custom_field; // global param ( for non blocking w params )

/*--------------INCLUDE Actualiacion OTADrive----------------*/
//#include <Arduino.h>
#include <ESP8266httpUpdate.h>
String version = "5.4";

/*--------------INCLUDE Detector de doble reset----------------*/
#ifdef ESP8266
#define ESP8266_MRD_USE_RTC     false   //true
#endif

#define ESP_MRD_USE_LITTLEFS    true
#define ESP_MRD_USE_SPIFFS      false
#define ESP_MRD_USE_EEPROM      false

#define MULTIRESETDETECTOR_DEBUG       true  //false

// These definitions must be placed before #include <ESP_MultiResetDetector.h> to be used
// Otherwise, default values (MRD_TIMES = 3, MRD_TIMEOUT = 10 seconds and
MRD_ADDRESS = 0) will be used
// Number of subsequent resets during MRD_TIMEOUT to activate
#define MRD_TIMES             3

// Number of seconds after reset during which a
// subsequent reset will be considered a multi reset.
#define MRD_TIMEOUT           2

// RTC/EEPROM Memory Address for the MultiResetDetector to use
#define MRD_ADDRESS           0

#include <ESP_MultiResetDetector.h>      //https://github.com/khoih-
prog/ESP_MultiResetDetector

MultiResetDetector* mrd;

/*--------------INCLUDE SENSOR----------------*/
#include <Wire.h>
#include "MAX30105.h"
#include "heartRate.h"
#define EMERG_PIN    D7
#define CARGADOR_PIN    D6
```

```
/*--------------DEFINES FIREBASE----------------*/
#include <FirebaseESP8266.h>
//Provide the token generation process info.
#include "addons/TokenHelper.h"
//Provide the RTDB payload printing info and other helper functions.
#include "addons/RTDBHelper.h"
#include <FirebaseESP8266.h>

/** 2. Define the API key */
#define API_KEY "AIzaSyBWyYTrP5mkeuHHLkVVujDjltqQQlqfkKk"

/* 3. Define the user Email and password that already registerd or added in your project */
#define USER_EMAIL "fede.sabbadin@gmail.com"
#define USER_PASSWORD "123456789"

/* 4. If work with RTDB, define the RTDB URL */
#define DATABASE_URL "https://pulsera-876d4-default-rtdb.firebaseio.com/"
//<databaseName>.firebaseio.com or <databaseName>.<region>.firebasedatabase.app

/** 5. Define the database secret (optional)*/
#define DATABASE_SECRET "B6ELG1y86pbJLyuH7hmv5u7xlDx0uPyAUEG4iIQG"

/* 6. Define the Firebase Data object */
FirebaseData fbdo;

/* 7. Define the FirebaseAuth data for authentication data */
FirebaseAuth auth;

/* 8. Define the FirebaseConfig data for config data */
FirebaseConfig config;

String path = "";
int count = 0;

/*--------------DEFINES SENSOR----------------*/

MAX30105 particleSensor;


const byte RATE_SIZE = 4; //Increase this for more averaging. 4 is good.
byte rates[RATE_SIZE]; //Array of heart rates
byte rateSpot = 0;
long lastBeat = 0; //Time at which the last beat occurred

float beatsPerMinute;
int beatAvg;
```

```
long samplesTaken = 0; //Counter for calculating the Hz or read rate
long unblockedValue; //Average IR at power up
long startTime; //Used to calculate measurement rate

int perCent;
int degOffset = 0.5; //calibrated Farenheit degrees
int irOffset = 1800;
int count1;
int noFinger;
//auto calibrate
int avgIr;
int avgTemp;
int brillo;
int tempmedia;
int urgencia;
int pase = 1;
int standby = 0;
int cargando = 0;
int conectado = 0;
int fueradelamuñeca = 0;
int bateria = ((0.526315789) * analogRead(A0) - (247.368421)); //los valores de A0 varian
entre 0 y 1024(bateria en 4,2V - 660 y en 3V 470)
int chip;


/*--------------DEFINE NTP----------------*/
#include <NTPClient_Generic.h>
#include <WiFiUdp.h>
WiFiUDP ntpUDP;
char timeServer[] = "0.ca.pool.ntp.org";
#define TIME_ZONE_OFFSET_HRS            (-3)
#define NTP_UPDATE_INTERVAL_MS          60000L
NTPClient timeClient(ntpUDP, timeServer, (3600 * TIME_ZONE_OFFSET_HRS),
NTP_UPDATE_INTERVAL_MS);
unsigned long i;

void setup()
{

  Serial.begin(115200);


  /*--------------VOID SENSOR----------------*/
  // Initialize sensor
  if (!particleSensor.begin(Wire, I2C_SPEED_FAST)) //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.");

 //The LEDs are very low power and won't affect the temp reading much but
 //you may want to turn off the LEDs to avoid any local heating
 particleSensor.setup(0); //Configure sensor. Turn off LEDs
 particleSensor.enableDIETEMPRDY(); //Enable the temp ready interrupt. This is required.

 //Setup to sense up to 18 inches, max LED brightness
 byte ledBrightness = 24; //Options: 0=Off to 255=50mA=0xFF hexadecimal. 100=0x64;
50=0x32 25=0x19
 byte sampleAverage = 4; //Options: 1, 2, 4, 8, 16, 32
 byte ledMode = 3; //Options: 1 = Red only, 2 = Red + IR, 3 = Red + IR + Green
 int sampleRate = 400; //Options: 50, 100, 200, 400, 800, 1000, 1600, 3200
 int pulseWidth = 411; //Options: 69, 118, 215, 411
 int adcRange = 2048; //Options: 2048, 4096, 8192, 16384

 particleSensor.setup(ledBrightness, sampleAverage, ledMode, sampleRate, pulseWidth,
adcRange); //Configure sensor with these settings

 //particleSensor.setPulseAmplitudeRed(0x0A); //Turn Red LED to low to indicate sensor is
running
 particleSensor.setPulseAmplitudeGreen(150); //Turn off Green LED
 particleSensor.enableDIETEMPRDY(); //Enable the temp ready interrupt. This is required.

 /*--------------BOTONES----------------*/
 pinMode(EMERG_PIN, INPUT); //Boton emergencia
 pinMode(CARGADOR_PIN, INPUT); //Sensor de cargador
 pinMode(LED_BUILTIN, OUTPUT);// Led integrado

 if (digitalRead(EMERG_PIN) == LOW) {
  //    Serial.println();
  //    Serial.print("OFF");
  //    Serial.println();
  urgencia = 0;
 }
 else {
  //    Serial.println();
  //    Serial.print("ON");
  Serial.println("BOTON PANICO");
  urgencia = 1;
 }

#if ESP_MRD_USE_LITTLEFS
 Serial.println(F(" using LittleFS"));
#elif ESP_MRD_USE_SPIFFS
```

```
  Serial.println(F(" using SPIFFS"));
#else
  Serial.println(F(" using EEPROM"));
#endif

  Serial.println(ESP_MULTI_RESET_DETECTOR_VERSION);

  mrd = new MultiResetDetector(MRD_TIMEOUT, MRD_ADDRESS);

  if (mrd->detectMultiReset())
  {
   Serial.println("Multi Reset Detected");
   digitalWrite(LED_BUILTIN, LOW);
   standby = 1;
  }
  else
  {
   Serial.println("No Multi Reset Detected");
   digitalWrite(LED_BUILTIN, HIGH);
  }


  if (bateria > 100) {
   bateria = 100;
  }
  if (bateria < 0) {
   bateria = 0;
  }
  if (bateria < 20) { //bateria baja
   digitalWrite(LED_BUILTIN, LOW);
  }

}

void loop()
{
  mrd->loop();
  if (millis() < 20000 && urgencia == 0) {

    /*--------------ENTRADAS----------------*/
    if (digitalRead(CARGADOR_PIN) == LOW) { //cardador desconectado
      digitalWrite(LED_BUILTIN, HIGH);
      Serial.println("CARGADOR DESCONECTADO");
    }
    else {
      digitalWrite(LED_BUILTIN, LOW); //Cargador conectado
      brillo = 0;
```

```
  particleSensor.setup(brillo);
  cargando = 1;
  Serial.println("CARGADOR CONECTADO");
 }


 /*--------------SENSOR----------------*/
 long irValue = particleSensor.getIR();

 if (checkForBeat(irValue) == true)
 {
  //We sensed a beat!
  long delta = millis() - lastBeat;
  lastBeat = millis();

  beatsPerMinute = 60 / (delta / 1000.0);

  if (beatsPerMinute < 255 && beatsPerMinute > 20)
  {
   rates[rateSpot++] = (byte)beatsPerMinute; //Store this reading in the array
   rateSpot %= RATE_SIZE; //Wrap variable

   //Take average of readings
   beatAvg = 0;
   for (byte x = 0 ; x < RATE_SIZE ; x++)
    beatAvg += rates[x];
   beatAvg /= RATE_SIZE;
  }
 }

 perCent = irValue / irOffset;
 if (perCent >= 100) {
  perCent = 99;
 }

 float temperatureF = particleSensor.readTemperatureF(); //Because I am a bad global
citizen
 temperatureF = temperatureF + degOffset;

 Serial.print("IR=");
 Serial.println(irValue);
 Serial.print(irValue);
 Serial.print(", BPM=");
 Serial.print(beatsPerMinute);
 Serial.print(", Avg BPM=");
 Serial.print(beatAvg);
 Serial.print(" ");
```

```
  Serial.print("Oxygen=");
  Serial.print(perCent);
  Serial.print("%");
  Serial.print(" Temp(F)=");
  Serial.print(((temperatureF - 32) / 1.8 + 2.5), 2);
  Serial.print("°");
  Serial.print(" IR=");
  Serial.print(irValue);

  if (irValue < 50000) {
    //    Serial.print(" No finger?");
    noFinger = noFinger + 1;
  } else {
    //only count and grab the reading if there's something there.
    count = count + 1;
    avgIr = avgIr + irValue;
    avgTemp = avgTemp + ((temperatureF - 32) / 1.8 + 2.5);
  }

  //Get an average IR value over 100 loops
  if (count == 100) {
    avgIr = avgIr / count;
    avgTemp = avgTemp / count;
    tempmedia = avgTemp;
    //    Serial.print(" avgO=");
    //    Serial.print(avgIr);
    //    Serial.print(" avgF=");
    //    Serial.print(avgTemp);
    //
    //    Serial.print(" count=");
    //    Serial.print(count);
    //reset for the next 100
    count = 0;
    avgIr = 0;
    avgTemp = 0;
    conectado = 1;
  }

  //turn off the LED if there's no finger
  //this doesn't work yet.
  if (noFinger == 500) {
    fueradelamuñeca = 1;
  }
}

if ((millis() > 20000 || urgencia == 1 || standby == 1 || cargando == 1)  && pase == 1) {
  pase = 0;
```

```
  brillo = 0;
  particleSensor.setup(brillo);
  /*--------------INICIO WIFI----------------*/
  WiFi.mode(WIFI_STA); // explicitly set mode, esp defaults to STA+AP
  Serial.setDebugOutput(true);
  Serial.println("\n Starting");
  //wm.resetSettings(); // wipe settings
  int customFieldLength = 40;
  const char* custom_radio_str = "<br/><label for='customfieldid'>Custom Field
Label</label><input type='radio' name='customfieldid' value='1' checked> One<br><input
type='radio' name='customfieldid' value='2'> Two<br><input type='radio'
name='customfieldid' value='3'> Three";
  new (&custom_field) WiFiManagerParameter(custom_radio_str); // custom html input
  wm.addParameter(&custom_field);
  //wm.setSaveParamsCallback(saveParamCallback);
  std::vector<const char *> menu = {"wifi", "info", "param", "sep", "restart", "exit"};
  wm.setMenu(menu);
  wm.setConfigPortalTimeout(60); // auto close configportal after n seconds
  bool res;
  res = wm.autoConnect("HealthBand"); // password protected ap
  if (!res) {
    Serial.println("Failed to connect or hit timeout");
    // ESP.restart();
  }
  else {
    //if you get here you have connected to the WiFi
    Serial.println("connected");
  }

  /*--------------INICIO NTP----------------*/
  timeClient.begin();

  /*--------------INICIO FIREBASE----------------*/
  /* Assign the api key (required) */
  config.api_key = API_KEY;

  /* Assign the user sign in credentials */
  auth.user.email = USER_EMAIL;
  auth.user.password = USER_PASSWORD;

  /* Assign the RTDB URL */
  config.database_url = DATABASE_URL;

  Firebase.reconnectWiFi(true);
  fbdo.setResponseSize(4096);

  //String base_path = "/pulsera1/";
```

```
  /* Assign the callback function for the long running token generation task */
  config.token_status_callback = tokenStatusCallback; //see addons/TokenHelper.h

  /** Assign the maximum retry of token generation */
  config.max_token_generation_retry = 5;

  /* Initialize the library with the Firebase authen and config */
  Firebase.begin(&config, &auth);
  mrd->loop();
  while (timeClient.updated() == 0) {

    Serial.print("Firebase = ");
    Serial.print(Firebase.ready());
    Serial.print("  ");
    Serial.print("NTP = ");
    Serial.print(timeClient.updated());
    Serial.print("  ");
    Serial.print("Bateria = ");
    Serial.print(bateria);
    Serial.println();
    Serial.print("Bateria analoga= ");
    Serial.print(analogRead(A0));
    Serial.println();
    timeClient.update();
    mrd->loop();
    if (millis() > 50000) {
      brillo = 0;
      particleSensor.setup(brillo);
      ESP.deepSleep(60e6);
    }
  }
}


 if (((millis() > 20100 && Firebase.ready()) || urgencia == 1 || standby == 1 || cargando == 1)
&& timeClient.updated())
 {
  brillo = 0;
  particleSensor.setup(brillo);
  if (ESP.getChipId() < 10000000){
    chip = ESP.getChipId()*10;
  }
  else{
    chip = ESP.getChipId();
  }
```

```
    i = (timeClient.getYear() % 100) * 100000000 + timeClient.getMonth() * 1000000 +
timeClient.getDay() * 10000 + timeClient.getHours() * 100 + timeClient.getMinutes();
    Serial.println(i);
    String direccion = "/pulsera" + String(chip) + "/mediciones/" + String(i);
    String estados = "/pulsera" + String(chip) + "/estados/" + String(i);

    String url = "http://otadrive.com/deviceapi/update?";
    url += "k=32fc44a2-a368-401a-a1d3-d49353d6bf11";
    url += "&v=" + version;
    url += "&s=" + String(chip);

    WiFiClient client;
    ESPhttpUpdate.update(client, url, version);

    if (standby == 1) {
      Firebase.setInt(fbdo, estados, 5);
      Serial.println("STANDBY");
      for (int i = 0; i <= 10; i++) {
        digitalWrite(LED_BUILTIN, HIGH);
        delay(200);
        digitalWrite(LED_BUILTIN, LOW);
        delay(200);
      }
      ESP.deepSleep(0);
    }
    else if (cargando == 1 && bateria > 98) { //carga completa
      Firebase.setInt(fbdo, estados, 3);
      Serial.println("CARGA COMPLETA");
      ESP.deepSleep(180e6);
    }
    else if (cargando == 1) { //
      Firebase.setInt(fbdo, estados, 4);
      Serial.println("CARGANDO");
      ESP.deepSleep(180e6);
    }
    else if (urgencia == 1) { //
      Firebase.setInt(fbdo, estados, 7);
      Serial.println("BOTON PANICO");
      ESP.deepSleep(50e6);
    }
    else if (conectado == 1) {
      if ((Firebase.setInt(fbdo, direccion + "/BPM", beatAvg)) && (Firebase.setInt(fbdo,
direccion + "/saturacion_oxigeno", perCent)) && (Firebase.setFloat(fbdo, direccion +
"/temperatura", tempmedia)) && (Firebase.setInt(fbdo, direccion +  "/bateria", bateria)))
      {
        Serial.println("CONECTADO");
        if (bateria < 20) { //bateria baja
```

```
        Firebase.setInt(fbdo, estados, 1);
        Serial.println("CONECTADO CON BATERIA BAJA");
       }
       else {
        Firebase.setInt(fbdo, estados, 2);
       }
       ESP.deepSleep(40e6);
     }
    }
    else if (fueradelamuñeca == 1) {
     Firebase.setInt(fbdo, estados, 6);
     Serial.println("FUERA DE LA MUÑECA");
     ESP.deepSleep(60e6);
    }
  }

  if (millis() > 50000) {
   brillo = 0;
   particleSensor.setup(brillo);
   ESP.deepSleep(60e6);
  }

}
```