UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA

FACULTAD DE SISTEMAS Y TELECOMUNICACIONES

TRABAJO DE INTEGRACIÓN CURRICULAR

Previo a la obtención del título de:

INGENIERO EN TELECOMUNICACIONES

Sensor de presión arterial basado en internet de las cosas (IoT) para el monitoreo continuo y remoto del paciente con accidente cerebrovascular (AVC)

AUTOR:

Vera Del Pezo Isaac George

TUTOR:

Ing. Carlos Andrade Caicho, MSc

LA LIBERTAD – ECUADOR

2025

TRIBUNAL DE SUSTENTACIÓN

Ing. Ronald Rovira Jurado, PhD.

DIRECTOR DE LA CARRERA

Ing. Carlos Andrade Caicho, MSc.

DOCENTE TUTOR

Ing. Manuel Montaño Blacio, MSc.

DOCENTE ESPECIALISTA

Ing. Luis Amaya Fariño, Mgtr .

DOCENTE GUÍA UIC

Ing. Corina Gonzabay De La A, Mgtr.

DIRECTOR DE LA CARRERA

UPSE

DECLARACIÓN DE DOCENTE TUTOR

Como docente tutor del trabajo de Integración Curricular,"***Sensor de presión arterial basado en internet de las cosas (IOT) para el monitoreo continuo y remoto del paciente con accidente cerebrovascular (AVC)***", elaborado por Isaac George Vera Del Pezo, estudiante de la Carrera de Telecomunicaciones, certifico que este proyecto, presentado a la Facultad de Sistemas y Telecomunicaciones de la Universidad Estatal Península de Santa Elena para la obtención del título de Ingeniería en Telecomunicaciones, cumple y se ajusta plenamente a los estándares académicos exigidos tras una exhaustiva supervisión de su desarrollo y estructura final. Por consiguiente, lo considero apto en todos sus aspectos y listo para ser evaluado por el docente especialista.

Atentamente

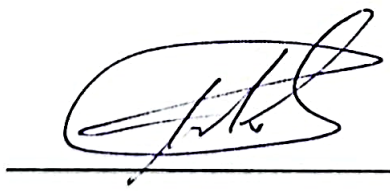
Ing. Carlos Andrade Caicho, MSc
DOCENTE TUTOR

UPSE

DECLARACIÓN DE DOCENTE ESPECIALISTA

Como docente especialista del trabajo de Integración Curricular,"***Sensor de presión arterial basado en internet de las cosas (IOT) para el monitoreo continuo y remoto del paciente con accidente cerebrovascular (AVC)***", elaborado por Isaac George Vera Del Pezo, estudiante de la Carrera de Telecomunicaciones, certifico que este proyecto, presentado a la Facultad de Sistemas y Telecomunicaciones de la Universidad Estatal Península de Santa Elena para la obtención del título de Ingeniería en Telecomunicaciones, cumple y se ajusta plenamente a los estándares académicos exigidos tras una exhaustiva supervisión de su desarrollo y estructura final. Por consiguiente, lo considero apto en todos sus aspectos y listo para ser evaluado por el docente especialista.

Atentamente



Ing. Manuel Montaño Blacio, MSc
DOCENTE ESPECIALISTA

UPSE

DECLARACIÓN AUTORÍA DE LOS ESTUDIANTES

El presente trabajo práctico de examen complexivo con el título **"Sensor de presión arterial basado en internet de las cosas (IOT) para el monitoreo continuo y remoto del paciente con accidente cerebrovascular (AVC)"**, se declara que su concepción, análisis y resultados son fruto original de la actividad educativa en el área de Telecomunicaciones.

Atentamente

Isaac George Vera Del Pezo
C.I: 2450005141

UPSE

DECLARACIÓN DE RESPONSABILIDAD

Quien se suscribe, Isaac George Vera Del Pezo con C.I. 2450005141, estudiante de la carrera de Telecomunicaciones, declaramos que el trabajo de titulación denominado **"Sensor de presión arterial basado en internet de las cosas (IOT) para el monitoreo continuo y remoto del paciente con accidente cerebrovascular (AVC)"** pertenece y es exclusivo responsabilidad del autor y pertenece al patrimonio intelectual de la Universidad Estatal Península de Santa Elena.

Atentamente

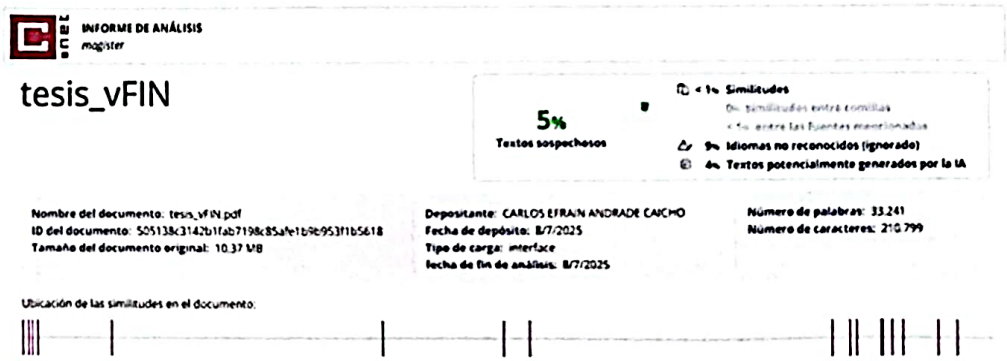
Isaac George Vera Del Pezo
C.I: 2450005141

UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA

SISTEMAS Y TELECOMUNICACIONES

CERTIFICACIÓN ANTIPLAGIO

Certifico que después de revisar el documento final del trabajo de titulación denominado ***"Sensor de presión arterial basado en internet de las cosas (IOT) para el monitoreo continuo y remoto del paciente con accidente cerebrovascular (AVC)"***, presentado por el estudiante **Isaac George Vera Del Pezo** fue enviado al **Sistema Antiplagio,** presentando un porcentaje de similitud correspondiente al **1%,** por lo que se aprueba el trabajo para que continúe con el proceso de titulación.



INFORME DE ANÁLISIS
magister

tesis_vFIN

5%
Textos sospechosos

< 1% Similitudes
0% similitudes entre comillas
< 1% entre las fuentes mencionadas
3% Idiomas no reconocidos (ignorado)
4% Textos potencialmente generados por la IA

Nombre del documento: tesis_vFIN.pdf
ID del documento: 505138c3142b1fab7198c85afe1b9c953f1b5618
Tamaño del documento original: 10.37 MB

Depositante: CARLOS EFRAIN ANDRADE CAICHO
Fecha de depósito: 8/7/2025
Tipo de carga: interface
fecha de fin de análisis: 8/7/2025

Número de palabras: 33.241
Número de caracteres: 210.799

Ubicación de las similitudes en el documento:

Ing. Carlos Andrade Caicho, MSc

DOCENTE TUTOR

DEDICATORIA

Este proyecto va dedicado a Dios y le agradezco por ser el faro que me ha guiado, por darme fortaleza, salud y perseverancia para continuar cada día.

Con todo el amor y el respeto que mi corazón puede albergar, dedico este logro a mis padres, Jorge Vera y Grely Del Pezo porque son el pilar fundamental de mi existencia y mi mayor inspiración para poder alcanzar este logro.

A mis abuelos por siempre aconsejarme, brindarme su amor incondicional y por ser la luz invaluable a lo largo de mi vida, especialmente a mi tío Juan Vera que en vida me brindó buenos consejos y la motivación que inspiraron a estudiar esta maravillosa rama de la ingienería.

A todos ustedes que me impulsaron a seguir adelante en todo este trayecto de mi carrea universitaria, por estar conmigo en los momentos buenos y hasta en los momentos más difíciles.

Dedicado a:

Jorge Vera
Grely Del Pezo

Isaac George Vera del Pezo

AGRADECIMIENTOS

A Dios, con infinita gratitud a lo largo de toda esta carrera, dándome la perseverancia para continuar en los momentos de mayor desafío y la claridad para aprender en medio de la confusión.

A toda mi familia, mi más profundo y sincero agradecimiento por ser el pilar incondicional sobre el que he construido cada uno de mis sueños, por su amor infinito que me ha sostenido en los momentos más difíciles, por su paciencia inagotable, por cada palabra de aliento que me impulsó a no rendirme y por cada sacrificio grande o pequeño que hicieron por mí.

A los docentes por brindar sus conocimientos a lo largo de preparación como profesional, agradecido con el Ing. Carlos Andrade tutor de mi trabajo de Integración Curricular, por su tiempo y guía en este proyecto, agradecer también al Ing. Manuel Montaño por ser una persona muy compresible, por sus conocimientos y orientación en la carrera.

A mis compañeros y a amigos de carrera Anthony Lascano y Gabriel Sanchez, les agradezco por hacer de este viaje una experiencia realmente inolvidable.

Mi más sincero agradecimiento a todos por formar parte de mi vida y de este importante trayecto profesional.

Isaac George Vera del Pezo

Índice general

Índice de tablas

Índice de figuras

RESUMEN

Este proyecto implementa un sensor de presión arterial basado en tecnología IoT para el monitoreo continuo y remoto de pacientes con accidente cerebrovascular utilizando un dispositivo portátil que integra el microcontrolador ESP32 y el sensor óptico MAX30102 para capturar señales fotopletismográficas no invasivas que son procesadas mediante algoritmos de Machine Learning en un servidor Flask desplegado en la nube para estimar la presión arterial sistólica y diastólica en tiempo real, clasificando automáticamente los niveles de riesgo según las guías médicas internacionales y almacenando los datos en una base de datos MySQL para seguimiento histórico, el sistema implementa comunicación inalámbrica Wi-Fi siguiendo el protocolo IEEE 802.11 para transmitir datos mediante HTTP/TCP-IP y cuenta con un sistema dual de alertas que incluye notificaciones locales a través de un buzzer integrado y alertas remotas vía WhatsApp utilizando la API de CallMeBot cuando se detectan valores críticos de presión arterial, la validación del prototipo demostró una autonomía operativa de aproximadamente 5 horas con batería de polímero de litio de 900 mAh y capacidad de recarga completa en 2 horas, representando una solución innovadora que complementa los métodos tradicionales de medición sin reemplazarlos y facilitando el monitoreo prolongado de pacientes con antecedentes cerebrovasculares para mejorar la prevención de eventos recurrentes y optimizar la calidad de vida mediante intervención médica oportuna.

Palabras clave: Sensor de presión arterial, Internet de las Cosas (IoT), Accidente Cerebrovascular (ACV), Monitoreo Continuo, Machine Learning, Telecomunicaciones

ABSTRACT

This project implements an IoT-based blood pressure sensor for continuous and remote monitoring of stroke patients using a portable device that integrates the ESP32 microcontroller and the MAX30102 optical sensor to capture non-invasive photoplethysmographic signals that are processed through Machine Learning algorithms on a Flask server deployed in the cloud to estimate systolic and diastolic blood pressure in real time, automatically classifying risk levels according to international medical guidelines and storing data in a MySQL database for historical tracking, the system implements Wi-Fi wireless communication following the IEEE 802.11 protocol to transmit data via HTTP/TCP-IP and features a dual alert system that includes local notifications through an integrated buzzer and remote alerts via WhatsApp using the CallMeBot API when critical blood pressure values are detected, prototype validation demonstrated an operational autonomy of approximately 5 hours with a 900 mAh lithium polymer battery and full recharge capacity in 2prob hours, representing an innovative solution that complements traditional measurement methods without replacing them and facilitating prolonged monitoring of patients with cerebrovascular history to improve prevention of recurrent events and optimize quality of life through timely medical intervention.

Keywords: Blood pressure sensor, Internet of Things (IoT), Stroke (CVA), Continuous Monitoring, Machine Learning, Telecommunications

Capítulo 1

1.1. Introducción

Las enfermedades cerebrovasculares, comúnmente conocidas como Accidentes Cerebrovasculares (ACV) constituyen un desafío significativo para la salud pública a nivel global y nacional, al ser la segunda causa principal de mortalidad y la tercera en causar discapacidad en todo el mundo, por eso la monitorización y regulación de la presión arterial (PA) son elementos cruciales para la prevención en el caso de un incidente, dado que la hipertensión arterial es un factor de riesgo considerable para se desarrolle de un evento de accidente cerebrovascular. Además, en aquellos pacientes que ya han experimentado un ACV, un control eficaz de la PA puede disminuir el riesgo de episodios recurrentes y mejorar el pronóstico.

Los métodos convencionales para monitorizar la presión arterial como los esfigmomanómetros manuales, automáticos o digitales, si bien ofrecen precisión y fiabilidad, presentan limitaciones significativas que incluyen la incomodidad para el paciente y la ausencia de una monitorización continua, en este ámbito, la tecnología del Internet de las Cosas (IoT) emerge como una herramienta complementaria sumamente prometedora, capaz de integrar sensores de presión arterial cómodos en dispositivos portátiles o vestibles que no solo recogen datos de forma no invasiva, sino que también los transmiten eficazmente a una plataforma de monitorización remota, permitiendo así a los profesionales médicos acceder a información adicional crucial sobre la presión arterial del paciente entre sus consultas presenciales.

Es importante destacar que este dispositivo basado en IoT no busca reemplazar los métodos tradicionales certificados de medición de la PA, los cuales seguirán siendo la herramienta principal y más precisa, especialmente para diagnósticos y situaciones que requieran una evaluación médica exhaustiva, sin embargo, al emplear un sensor no invasivo y cómodo como el MAX30102, la propuesta puede ofrecer una monitorización continua de la PA en entornos no controlados o supervisados médicamente, identificando patrones y posibles fluctuaciones que sirvan como alerta temprana para una valoración presencial, lo que a su vez busca mejorar la calidad de vida del paciente, facilitar un mejor control de la PA, prevenir complicaciones y permitir una intervención médica oportuna en caso de ser necesaria.

En definitiva, el desarrollo de este dispositivo de monitoreo de PA basado en IoT representa una oportunidad innovadora para complementar el manejo de pacientes con ACV, brindando información adicional que permita una mejor prevención, control y calidad de vida, sin reemplazar los métodos tradicionales certificados. La

investigación y el desarrollo en esta área son fundamentales para llevar esta tecnología a la práctica clínica y mejorar la atención integral de estos pacientes.

Palabras claves: Sensor de presión arterial, Internet de las Cosas (IoT), Accidente Cerebrovascular (ACV), Monitoreo Continuo, Atención Médica, Tecnología

1.2. Problemática

Los métodos convencionales para la monitorización de la presión arterial (PA) presentan significativas limitaciones en el seguimiento continuo, ya que es esencial para la prevención y el tratamiento de los Accidentes Cerebrovasculares (ACV).

Los métodos esfigmomanómetros manuales o automáticos aunque son precisos y confiables, pueden resultar incómodos para el paciente y no permiten un monitoreo continuo.

A pesar que los dispositivos tradicionales permiten la medición automática, no facilitan un registro debido a que el profesional de la salud debe registrar manualmente los valores de presión arterial, lo que dificulta la medición de continua. Además, estos dispositivos no ofrecen un acceso remoto en la línea de información.

1.3. Justificación de la Problemática

Radica en la necesidad de desarrollar un sensor de presión arterial basado en IoT que permita un monitoreo continuo, remoto de los pacientes y un registro automático superando las limitaciones de los métodos convencionales y los desafíos de la implementación de IoT en la práctica clínica. Este desarrollo podría facilitar la detección oportuna de fluctuaciones de la PA, permitiendo la implementación de acciones preventivas y de gestión terapéutica apropiadas. Además, facilitaría en la recopilación de datos, impulsando a la mejora de comprensión dinámica para la PA en pacientes con accidentes cerebrovasculares, promoviendo a la innovación de nuevas estrategias de atención efectivas.

1.4. Objetivos

1.4.1. Objetivo General

Implementar un sensor de presión arterial basado en IoT para el monitoreo continuo y remoto del paciente con accidente cerebrovascular.

1.4.2. Objetivos Específicos

El objetivo fundamental de la tesis en base al objetivo principal, se proponen los siguientes objetivos parciales:

- Seleccionar adecuadamente los componentes electrónicos para el sensor de presión arterial basado en IoT.
- Diseñar la placa de circuito impreso (PCB) para la integración del sensor de presión arterial basado en IoT.
- Prototipar la placa de circuito impreso (PCB) que integra el sensor de presión arterial basado en IoT.
- Realizar pruebas para el monitoreo continuo de presión arterial basado en IoT.

1.5. Formulación del Problema

- ¿Cómo se puede medir de manera continua y remota la presión arterial en pacientes con ACV (accidentes cerebrovasculares)?
- ¿De qué se compone una solución para medir la presión arterial de manera continua y remota?
- ¿Cómo se diseñan y se configuran los componentes de una solución de este tipo?
- ¿Cuál es el proceso de construcción?
- ¿Cómo se verifica el funcionamiento de la solución a implementar?

1.6. Alcance

El proyecto se centra en el diseño e implementación de un dispositivo de monitoreo continuo y remoto de la presión arterial basado en tecnologías IoT para pacientes con accidente cerebrovascular (ACV) incluyendo la selección de componentes como el sensor, el diseño de la placa de circuito impreso, la integración de un sistema de alarmas, la conexión a Internet para transmitir datos y el almacenamiento en la nube.

El dispositivo permitirá las mediciones y el monitoreo remoto de la presión arterial en un ambiente controlado y no controlado, con el objetivo de detectar fluctuaciones, accionar de manera preventiva y la realización de solicitudes terapéuticas oportunas.

1.7. Descripción del proyecto

Este dispositivo de monitoreo continuo de presión arterial basado en IoT permitirá la recolección y análisis de datos en tiempo real. Para lograr esto, se deben seleccionar adecuadamente los componentes electrónicos y diseñar y prototipar la placa de circuito impreso (PCB). El dispositivo se divide en tres secciones clave, la primera sección incluye los componentes principales para el procesamiento de los valores en la medición y almacenamiento de los datos (sensor PPG óptico no invasivo, placa de desarrollo, nube), la segunda sección incorpora una batería y un sistema protección para garantizar su durabilidad, permitiendo el monitoreo constante de la presión arterial en paciente con accidente cerebrovascular y la tercera sección consiste en un sistema de alertas (chatbot, bocina integrada) para el paciente, proporcionando notificaciones al médico que esté a cargo del paciente cuando se detecten valores anómalas.

1.8. Antecedentes investigativos

El accidente cerebrovascular (ACV) se encuentra entre las principales causas de enfermedad y muerte en la población adulta de América Latina, en Ecuador ocupa el tercer lugar como causa de fallecimiento en adultos mayores, este problema no solo afecta la salud individual de quienes lo padecen sino que también tiene implicaciones sociales y económicas significativas [1].

En el monitoreo de la presión arterial se puede controlar de manera remota por medio de sistemas IoT, esto da importancia en la prevención, detección temprana y acción de tratamiento, la hipertensión arterial constituye un factor de riesgo primordial en el desarrollo de un accidente cerebrovascular, lo que resalta la importancia crucial de monitorear constantemente los niveles de presión arterial para el manejo del paciente, sin embargo los métodos tradicionales de medición de presión arterial, como los equipos de consultorio médico, presentan limitaciones para el monitoreo a largo plazo de los pacientes, los dispositivos solo proporcionan mediciones puntuales que pueden no reflejar las fluctuaciones de la presión arterial a lo largo del día. Además, requieren la presencia del paciente en un entorno clínico, lo cual dificulta el seguimiento remoto y continuo [2].

El desarrollo de sensores de presión arterial basados en IoT para el seguimiento de pacientes con accidentes cerebrovasculares y otras condiciones crónicas, ayudarán en la recopilación y transmisión de datos de presión arterial de manera continua y lo mejor en tiempo real para facilitar la detección temprana de fluctuaciones y así implementar acciones preventivas o terapéuticas oportunas [3].

Se esperan grandes expectativas en esta tecnología de internet de las cosas porque probar un prototipo en clínicas reales demostraría si de verdad funciona, permitiendo no solo vigilar constantemente la presión arterial en pacientes que tuvieron un derrame cerebral, sino también medir qué tan bueno es el dispositivo en un ambiente médico real y que al desarrollar se pueda obtener que esta solución es útil, ayudamos a que la tecnología para la salud avance y ofrecemos una herramienta

importante para el bienestar de los pacientes, por lo cual es fundamental investigar y crear estas soluciones de monitoreo para cuidar la salud [4].

1.9. Resultados Esperados

Se proyecta que la implementación exitosa de este innovador dispositivo de monitoreo continuo de la presión arterial, respaldado por tecnologías de Internet de las Cosas (IoT), generará los siguientes impactos y beneficios para los pacientes con Accidente Cerebrovascular (ACV):

- Facilitará la vigilancia continua y remota de los niveles de presión arterial en el entorno habitual del paciente, prescindiendo de su presencia física en centros de atención médica.
- Posibilitará la detección temprana de fluctuaciones anómalas en los valores de presión arterial, permitiendo la aplicación oportuna de protocolos preventivos y terapéuticos personalizados.
- Establecerá las bases para una comprensión más profunda de las dinámicas y patrones de la presión arterial en pacientes con ACV, mediante la recopilación inicial de datos que serán almacenados de forma segura en la nube, sentando un precedente para futuras recolecciones masivas.
- Sentará las bases para el desarrollo de estrategias de atención médica más efectivas y políticas de salud pública, orientadas a la prevención y manejo integral de los ACV fundamentadas en evidencia empírica recolectada por el dispositivo.
- Impulsará el avance tecnológico en el campo de la atención médica, al demostrar la viabilidad y los beneficios de soluciones de monitoreo continúo respaldadas por IoT.
- Repercutirá positivamente en la calidad de vida de los pacientes con ACV, al facilitar un control más estricto de la presión arterial y permitir intervenciones médicas oportunas cuando sean requeridas.

Se pretende marcar una diferencia en el cuidado de pacientes con accidente cerebrovascular, aprovechando las bondades de las tecnologías emergentes para brindar una atención médica más personalizada, oportuna y fundamentada en evidencia científica sólida.

Capítulo 2

FUNDAMENTOS TEÓRICOS DE LA PROPUESTA

2.1. Marco Conceptual

2.1.1. IoT - Internet de las cosas

El Internet de las Cosas (IoT), representa una transformación en la interacción entre personas y la creciente variedad de dispositivos electrónicos contemporáneos que busca establecer canales de comunicación entre estos dispositivos y el usuario, la conexión de estos dispositivos a una red potencia avances en diversos campos como automatización, precisión y domótica, entre otros.

El concepto de IoT hace referencia a una red compuesta por objetos físicos equipados con sensores, software y otras tecnologías que les permiten conectarse e intercambiar información con otros sistemas a través de Internet que abarca desde simples artículos de uso doméstico hasta complejos equipos industriales.

En cuanto a las comunicaciones "dispositivo a la nube", este modelo permite que un dispositivo IoT establezca conexión directa con un servicio en la nube (como un proveedor de servicios de aplicaciones) para el intercambio de datos y la gestión del flujo de mensajes, esta aproximación generalmente utiliza infraestructuras de comunicación ya existentes, como conexiones WiFi o Ethernet tradicionales, para crear un enlace entre el dispositivo y la red IP, que posteriormente se conecta con el servicio en la nube [5].

2.1.2. Evolución del IoT

En esta era de la evolución del Internet de las Cosas se ha caracterizado al ser un crecimiento exponencial en cantidad y variedad de dispositivos conectados, cuyo efecto a impulsado avances tecnológicos claves como la inteligencia artificial, la tecnología 5G y el Edge Computing.

La expansión de nuevas tecnologías ha llevado a una integración más profunda

del IoT en diversos sectores, incluyendo la salud, la industria, las ciudades inteligentes y los hogares conectados, aumentando la tendencia hacia dispositivos más inteligentes y autónomos que son capaces de tomar decisiones basadas en datos en tiempo real [6].

Otro factor muy importante en la evolución de nuevas tecnologías es la seguridad y la privacidad, convirtiéndose en prioridades críticas con un enfoque en el desarrollo de protocolos y estándares más robustos, además, la convergencia del IoT con otras tecnologías emergentes como la realidad aumentada y la cadena de bloques está abriendo nuevas posibilidades y casos de uso, transformando radicalmente la forma en que interactuamos con nuestro entorno y gestionamos la información [6].

2.1.3. Características del IoT

Existen características claves que conforman el Internet de las Cosas (IoT) ya que es un campo apasionante que conecta dispositivos y sistemas a través de Internet, permitiendo transformar las industrias y mejorando la vida cotidiana. Para que las funciones sean posibles, IoT cuenta con las siguientes características.

- **Conectividad:** Es importante en la arquitectura de IoT, se basa en la conexión de dispositivos por medio de internet [7].
- **Inteligencia e identidad:** Inteligencia e identidad: Es la extracción de conocimiento y el análisis a fondo de los datos generados por los dispositivos [7].
- **Escalabilidad:** La cantidad de elementos conectados a IoT crece masivamente y la inclusión de más dispositivos mediante su configuración no debe afectar su funcionamiento [7].
- **Dinámico y autoadaptable:** Los dispositivos IoT deben ser perfectamente adaptativos en los entornos y escenarios cambiantes [7].
- **Seguridad:** Es imprescindible la seguridad de los datos y seguridad que implementada del equipo para IoT [7].

2.1.4. Impacto del IoT

En la actualidad, el uso del internet de las cosas a formado parte de la vida cotidiana de las personas disponiendo con sistemas de automatización que interactúan con el usuario que va transformando diversos aspectos en la sociedad, como en el ámbito empresarial mejorando la eficiencia operativa mediante la automatización y optimización de procesos, esto permite en la toma de decisiones de datos en tiempo real, además, IoT ha generado nuevas oportunidades de negocios al facilitar la creación de soluciones personalizadas y estrategias de mercado innovadoras [8, p. 347].

Cuando nos referimos a la calidad de vida, IoT contribuye significativamente a través de aplicaciones en domótica, monitoreo de salud y desarrollo de ciudades inteligentes, no obstante, es crucial considerar los desafíos presentes de seguridad y privacidad asociados [8, p. 347].

2.1.5. Tecnologías Inalámbricas IoT

2.1.5.1. Wi-Fi

Figura 2.1: Wi- Fi

Fuente: Wi-Fi Alliance [9]

Wi-Fi es una tecnología de red inalámbrica que permite la conexión de dispositivos a internet o entre sí sin necesidad de cables utilizando ondas de radio, opera principalmente en las bandas de 2.4 GHz y 5 GHz aunque versiones más recientes utilizan bandas de 6 GHz y 60 GHz, desde su introducción ha evolucionado significativamente, mejorando en velocidad, eficiencia y capacidad de conexión simultánea [10].

Versión	Estándar IEEE	Frecuencia	Velocidad Máxima	Alcance Aproximado	Año de Aprobación
Wi-Fi 1	802.11b	2.4 GHz	11 Mbps	~ 35 m en interiores	1999
Wi-Fi 2	802.11a	5 GHz	54 Mbps	~ 35 m en interiores	1999
Wi-Fi 3	802.11g	2.4 GHz	54 Mbps	~ 38 m en interiores	2003
Wi-Fi 4	802.11n	2.4/5 GHz	600 Mbps	~ 70 m en interiores	2009
Wi-Fi 5	802.11ac	5 GHz	1.3 Gbps	~ 35 m en interiores	2014
Wi-Fi 6	802.11ax	2.4/5 GHz	9.6 Gbps	~ 35 m en interiores	2019
Wi-Fi 6E	802.11ax	6 GHz	9.6 Gbps	~ 30 m en interiores	2020
Wi-Fi 7	802.11be	2.4/5/6 GHz	46 Gbps	~ 30 m en interiores	2024 (estimado)

Tabla 2.1: Evolución de los estándares Wi-Fi y sus características técnicas

Fuente: Elaboración propia basada en Rentería Manjarrez, Ortiz Carrasco y Gaxiola Sánchez [10]

2.1.5.2. Bluetooth

Figura 2.2: Bluetooth

Fuente: Bluetooth SIG [11]

Bluetooth es una tecnología de comunicación inalámbrica de corto alcance diseñada para reemplazar cables en la conexión de dispositivos electrónicos, opera en

la banda de 2.4 GHz y es ampliamente utilizada en dispositivos como auriculares, teclados, ratones y sistemas de manos libres, ha evolucionado para ofrecer mayores velocidades y menores consumos de energía [12].

Versión	Estándar IEEE	Frecuencia	Velocidad Máxima	Alcance Aproximado	Año de Aprobación
Bluetooth 1.0	802.15.1	2.4 GHz	721 Kbps	~ 10 m	1999
Bluetooth 2.0	802.15.1	2.4 GHz	2.1 Mbps	~ 10 m	2004
Bluetooth 3.0	802.15.1	2.4 GHz	24 Mbps	~ 10 m	2009
Bluetooth 4.0	802.15.1	2.4 GHz	25 Mbps	~ 10 m	2010
Bluetooth 5.0	802.15.1	2.4 GHz	50 Mbps	~ 10 m	2016

Tabla 2.2: Evolución de los estándares Bluetooth y sus características

Fuente: Elaboración propia basada en Rentería Manjarrez, Ortiz Carrasco y Gaxiola Sánchez [10]

2.1.5.3. ZigBee

Figura 2.3: Zigbee

Fuente: ByJasco [13]

ZigBee es una tecnología de comunicación inalámbrica de bajo consumo y baja velocidad, diseñada para aplicaciones que requieren transmisión de pequeños volúmenes de datos, como la automatización del hogar, control industrial y redes de sensores operando en las bandas de 2.4 GHz, 915 MHz y 868 MHz, y es conocida por su capacidad de formar redes en malla [14].

Estándar IEEE	Frecuencia	Velocidad Máxima	Alcance Aproximado	Año de Aprobación
802.15.4	2.4 GHz	250 Kbps	~ 10 –100 m	2003
802.15.4	915 MHz (EE.UU.)	40 Kbps	~ 10 –100 m	2003
802.15.4	868 MHz (Europa)	20 Kbps	~ 10 –100 m	2003

Tabla 2.3: Características de los estándares Zigbee según IEEE 802.15.4

Fuente: Elaboración propia basada en Rentería Manjarrez, Ortiz Carrasco y Gaxiola Sánchez [10]

2.1.5.4. Tecnología Celular

Figura 2.4: Evolución de las tecnologías celulares desde 1G hasta 5G

Fuente: Bazzo y João [15]

Las tecnologías celulares han evolucionado significativamente desde la introducción de la primera generación 1G en la década de 1980 hasta la actual quinta generación 5G, cada generación ha representado un salto cualitativo en términos de capacidad, velocidad, eficiencia y servicios ofrecidos [16].

Generación	Frecuencia Principal	Protocolos Estándar	Velocidad Máxima	Alcance Aproximado	Año de Aprobación
1G	800 MHz	AMPS, NMT	2.4 kbps (solo voz)	~ 10 –15 km	1980
2G	900/1800 MHz	GSM, CDMA, IS-95	64 kbps	~ 5 –10 km	1991
2.5G	900/1800 MHz	GPRS, EDGE	384 kbps	~ 5 –10 km	2000
3G	2.1 GHz	UMTS, WCDMA, CDMA2000	2 Mbps	~ 2 –5 km	2001
3.5G	2.1 GHz	HSPA, HSPA+	14.4 Mbps	~ 2 –5 km	2006
4G	700/2600 MHz	LTE (3GPP Release 8)	100 Mbps – 1 Gbps	~ 2 –5 km	2009
4.5G	700/2600 MHz	LTE-Advanced (3GPP Release 10)	Hasta 3 Gbps	~ 2 –5 km	2013
5G	3.5 / 26–28 GHz	5G NR (3GPP Release 15 y posteriores)	Hasta 10 Gbps	~ 1 –3 km (mmWave)	2019

Tabla 2.4: Evolución de tecnologías celulares de 1G a 5G y sus características

Fuente: Elaboración propia basada en Kheddar [17]

2.1.5.5. LoRa (Long Range)

Figura 2.5: Esquema ilustrativo de comunicación LoRa en IoT

Fuente: The Things Network – Comunidad Santa Rosa [18]

LoRa es una tecnología de modulación de espectro ensanchado derivada de chirp spread spectrum (CSS) diseñada para comunicaciones de largo alcance y bajo consumo de energía, es ampliamente utilizada en aplicaciones de Internet de las Cosas (IoT) que requieren transmisión de datos a larga distancia con bajo ancho de banda, como sensores ambientales y medidores inteligentes [19].

Frecuencia	Velocidad Máxima	Alcance Aproximado	Año de Introducción
433 MHz	50 Kbps	∼ 2 –5 km en ciudad	2015
868 MHz (Europa)	50 Kbps	∼ 2 –5 km en ciudad	2015
915 MHz (EE.UU.)	50 Kbps	∼ 2 –5 km en ciudad	2015

Tabla 2.5: Frecuencias utilizadas en LoRa y sus principales características

Fuente: Elaboración propia basada en Rentería Manjarrez, Ortiz Carrasco y Gaxiola Sánchez [10]

2.1.5.6. NFC (Near Field Communication)

Figura 2.6: Funcionamiento del NFC en smartphones

Fuente: TechTudo [20]

NFC (Comunicación de Campo Cercano) permite el intercambio de datos de forma inalámbrica y a corta distancia (generalmente menos de 10 cm) operando en la banda de 13.56 MHz y es ampliamente utilizada en aplicaciones como pagos móviles, tarjetas de transporte, acceso a instalaciones y emparejamiento rápido de dispositivos [21].

Parámetro	Especificación
Frecuencia	13.56 MHz
Protocolos	ISO/IEC 18092, ISO/IEC 14443, ECMA-340
Velocidad Máxima	Hasta 424 kbps
Alcance	Hasta 10 cm
Año de Introducción	2004 (estandarización inicial)

Tabla 2.6: Especificaciones técnicas del sistema de comunicación NFC

Fuente: Elaboración propia basada en Aguirre y Vera [22]

2.1.6. Modelo OSI

La Organización Internacional de Estándares (ISO) se encarga de gestionar el uso y la creación de un modelo que ayuda a manejar una arquitectura de red basada

en niveles, en cambio el modelo de Interconexión de Sistemas Abiertos (OSI) es denominado un estándar para la comunicación entre sistemas abiertos organizando los servicios de red en siete capas, la primera está más cerca del medio físico y la séptima se relaciona más con las aplicaciones que utilizan los usuarios, en pocas palabras cuando un agente necesita transferir datos a un destino específico, el sistema de red proveerá la información de control a cada servicio que utiliza la red para enrutar la solicitud [23].

El modelo OSI está conformada por las siguientes capas:

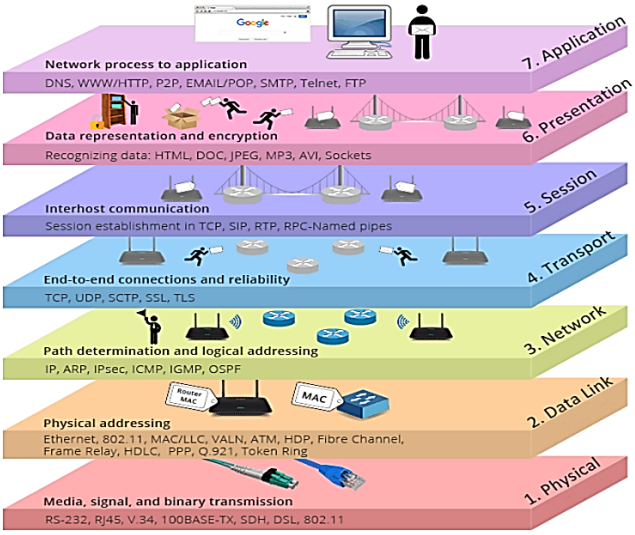


Figura 2.7: Capas Del Modelo OSI.

Fuente: GoTo IoT [24]

2.1.6.1. Capa Fisica

Es fundamental observar constantemente la línea de comunicaciones, definiendo cómo y qué medio se va a utilizar, ya sea puertos, cables, etc. En este proceso intervienen aspectos como la transmisión de bits por canal, la velocidad y los conductores de Full Dúplex [25].

2.1.6.2. Capa de Enlace

Proteger la información y asegurarse de que se mantenga íntegra es fundamental, es decir, que los datos que se transmiten lleguen sin errores. Esta capa se encarga del control físico, asegurando que todo fluya sin fallos. También permite identificar cuándo comienza y termina la trama, y si hay algún daño, se puede retransmitir con cuidado para evitar duplicados [25].

2.1.6.3. Capa de Red

Establece una conexión con el host y añade encabezados para el enrutamiento (Dirección IP) que juega un papel importante en el control de subred, el enrutamiento y la transmisión de datos [26].

2.1.6.4. Capa de Transporte

La capa de transporte dispone y protege la comunicación de los datos mientras se trasladan entre las capas.

Protocolo TPC/IP se encarga de dividir la información en franjas secuenciales, conocidas como tramas, en la capa de importante enumerar estas tramas para que el receptor pueda organizarlas correctamente [26].

2.1.6.5. Capa de Sesión

El ID de la comunicación se encarga de controlar y numerar la sesión (Login), también permite que varios equipos establezcan sesiones entre ellos facilitando el acceso remoto, la transferencia de archivos y la sincronización [26].

2.1.6.6. Capa de Presentación

Convierte los formatos de datos a uno común y asegura que la capa de aplicación de un equipo se entienda con la de otro [23].

2.1.6.7. Capa de Aplicación

Los programas que se ejecutan en el host a nivel de usuario intervienen en muchos protocolos, como el correo electrónico y la transferencia de archivos [23].

En la actualidad las **capas de sesión, presentación y aplicación** son agrupadas en una sola capa, la siguiente Figura 2.8, se observa las capas que encajan con los protocolos de IoT.

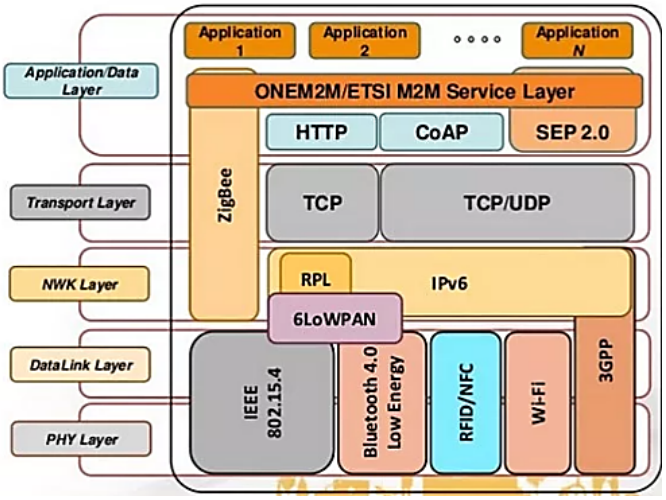


Figura 2.8: Capas compatibles en IoT

Fuente: GoTo IoT [24]

2.1.6.8. Capa de Aplicación

La capa de aplicación constituye el puente de comunicación entre el usuario final y el dispositivo físico, definiendo cómo interactúan ambos dentro del ecosistema del Internet de las Cosas (IoT).

- **MQTT (Transporte de Telemetría de Cola de Mensajes):** Protocolo de mensajería elaborado para que establezca una comunicación ligera M2M y utilizado esencialmente para enlaces de ancho de banda bajo en lugares remotos [27].
- **AMQP (Protocolo Avanzado de Colas de Mensajes):** Es una capa de software que crea la interoperabilidad entre middleware de mensajería cooperando en una variedad de sistemas y ayuda que las aplicaciones trabajen de manera unificada creando mensajes estandarizados a escala industrial.
- **CoAp (Protocolo de Aplicación Limitado):** Protocolo elaborado para ambientes con un ancho de banda restringidos y dispositivos con capacidad limitada, es considerado como un protocolo de transmisión de documentos que se emplean sobre el UDP [27].

2.1.6.9. Capa de Transporte

La capa de transporte dispone y protege la comunicación de los datos mientras se trasladan entre las capas.

- **TCP (Protocolo de Control de Transmisión):** Es el protocolo principal para la mayoridad de conectividad a internet, ofreciendo comunicación de host a host, dividiendo mayores conjuntos de datos en paquetes individuales y reenviando y juntando paquetes según sea necesario [28].

- **UDP (Protocolo de Datagramas de Usuario):** Protocolo de comunicaciones que opera sobre el Protocolo de Internet (IP), facilitando la interacción directa de procesos. UDP mejora las tasas de transferencias de datos encima de TCP y siendo adaptativo en las aplicaciones que necesitan transmisiones de datos sin pérdidas [28].

2.1.6.10. Capa de Red

Esta capa de red ofrece apoyo a los dispositivos individuales en la conectividad con el router y dispone a cada dispositivo una dirección para lograr comunicarse.

- **IPv6:** Viene de la evolución o actualización de IPv4, enruta el tráfico por medio del internet e identifica y localiza los dispositivos en la red.
- **6LoWPAN:** Es la versión de menor potencia del IPv6, disminuye los tiempos de transmisión.

2.1.6.11. Capa de Enlace

Esta capa se encarga de la transferencia de datos a través de la arquitectura del sistema mientras identifica y corrige los errores que se encuentran en la capa física.

- **IEE 802.15.4:** Es un estándar de radio que permite la conexión inalámbrica de baja potencia, este estándar se utiliza para construir redes inalámbricas integradas, así como con Zigbee, 6LoWPAN, entre otros estándares [29].
- **LPWAN:** Es un tipo de red que facilita la comunicación con una distancia mínima de 500m, el LoRaWAN es el modelo de LPWAN optimizado para un mínimo consumo de energía [29].

2.1.6.12. Capa Física

La capa física es la que proporciona un canal de comunicación y permite que los dispositivos puedan conectarse dentro de un medio físico.

- **Ethernet:** Es más conocida porque es la conexión por cables, es una opción que puede ser costosa, aunque proporciona una conexión de transferencia de datos más rápidas y con menor latencia [28].
- **Wi-Fi/802.11:** Es un estándar empleado en hogares u oficinas siendo una de las mejores opciones por su adquisición económica, tiene un rango de alcance limitado en diferentes escenarios y requiere de energía las 24 horas del día [28].

2.1.7. Modelo TCP/IP

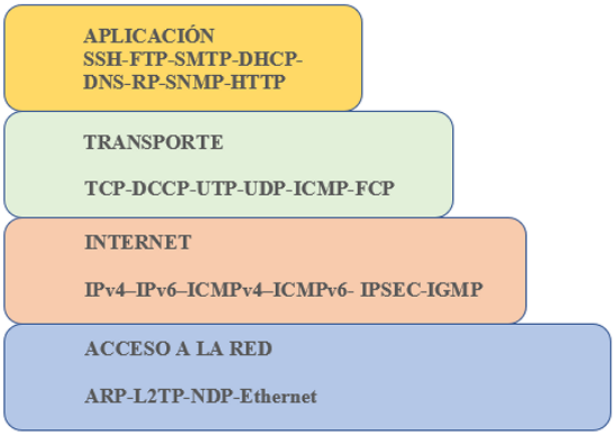


Figura 2.9: Modelo de Protocolo TCP/IP

Fuente: Delgado y Efraín [30]

2.1.7.1. Capa Aplicación

Este proceso incluye una serie de aplicaciones que facilitan la interacción y la transmisión de datos con la capa inferior, conocida como la Capa de Transporte, ofreciendo servicios para el correo electrónico, la transferencia de archivos y la conexión remota [30].

2.1.7.2. Capa Transporte

Existen varios protocolos como TCP, que permiten establecer una conexión de un extremo a otro con la capacidad de detectar y corregir errores, mientras UDP se encarga de reducir el tamaño de la cabecera, lo que le da una ventaja en la velocidad de transmisión de datos [30].

En esta capa, es fundamental que haya una conexión entre los hosts, y se lleva a cabo el intercambio en tres segmentos de datos, donde en la primera situación el host A envía un segmento de sincronización (SYN) al host B, incluyendo un identificador numérico, luego el host B localiza al host A y responde con un segmento de confirmación (ACK) para dar inicio a la transmisión de datos y finalmente una vez que el host A recibe esta confirmación este envía de vuelta al host B un ACK junto con la primera trama de datos, comenzando así la transmisión [31].

2.1.7.3. Capa Internet

El protocolo IP es el encargado de interactuar con las capas superiores y ha evolucionado con varias versiones debido al creciente número de dispositivos en la red, dado que IPv4 tiene un límite en la cantidad de hosts surge la necesidad de implementar IPv6 que ofrece una capacidad mucho mayor para el direccionamiento.

Un paquete accede a la capa de red y se transmite en forma de datagrama, revisando las direcciones de origen y destino, y luego envía los datos de un dispositivo a otro [31].

2.1.7.4. Capa de Acceso a la Red

Este nivel es donde se encapsulan los datagramas del protocolo IP para ser transmitidos en tramas a través de la red Ethernet, de la misma manera se vinculan las direcciones de cada dispositivo (IP física) con los adaptadores de red.

2.1.8. Conformación de un red IoT

En una arquitectura interna de redes IoT para diversas aplicaciones, diferentes entidades y expertos han propuesto modelos estructurales, la Unión Internacional de Telecomunicaciones (UIT), a través de la recomendación ITU-T Y.2060 publicada en 2012, estableció definiciones y atributos fundamentales de IoT, además de un modelo jerárquico de cuatro niveles similar al modelo desarrollado en 2014 por el comité de arquitectura del IoT World Forum (IoTWF), integrado por corporaciones líderes como CISCO, identifica una estructura de siete capas para las redes IoT [32].

El modelo presentado por el IoTWF se ilustra y se muestra en la Figura 2.10.

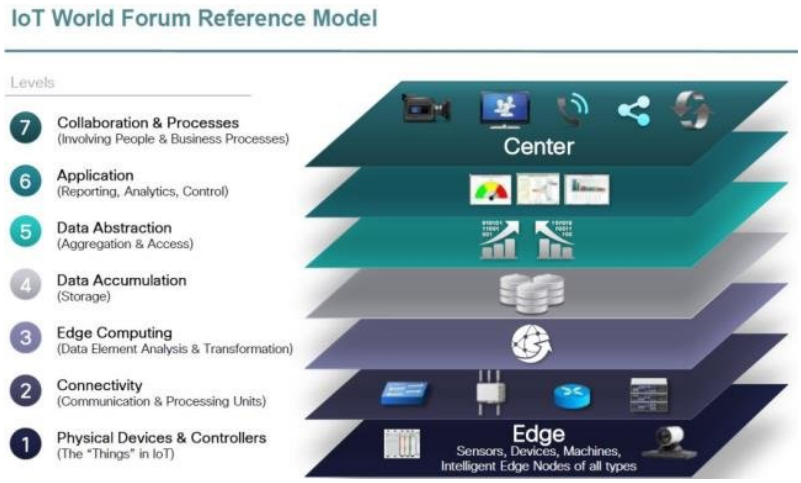


Figura 2.10: Elementos por IoTWF

Fuente: Quishpe [32]

2.1.8.1. Capa 1 - Elementos físicos y controladores

La primera capa del modelo propuesto por IoTWF se compone de datos que provienen de dispositivos finales, ya sean sensores o actuadores, los cuales envían información a la red IoT [33, p. 20].

2.1.8.2. Capa 2 - Conectividad

Dentro de una red IoT, la capa 2 facilita la transferencia de información desde los dispositivos finales hasta la capa 3, incorporando para ello mecanismos de fiabilidad, seguridad, gestión de protocolos y funciones de enrutamiento (routing) y conmutación (switching) [33, p. 20].

2.1.8.3. Capa 3 - Computación de borde

Una de las características de la computación de borde es que permite que la información que llega a la capa 3 se reduzca, almacene y procese antes de pasar a la capa superior, luego gracias al filtrado de información, se puede disminuir el tráfico, generando notificaciones cuando se superan ciertos umbrales predefinidos [33, p. 20].

2.1.8.4. Capa 4 - Almacenamiento de datos

La recolección de datos permite que las aplicaciones superiores utilicen la información clave en el instante preciso, fundamentando su procesamiento tanto en datos de eventos como de consultas [33, p. 21].

2.1.8.5. Capa 5 - Abstracción de información

La información que se introduce en la capa 5 se organiza en uno o varios centros de datos a través de la virtualización. La semántica y la estructura indican que la información está completa [33, p. 21].

2.1.8.6. Capa 6 - Reporte análisis y Control

Para gestionar la información de sensores o actuadores, la capa 6 ofrece la ventaja de monitorear, controlar y analizar los datos que circulan en la red IoT a través de aplicaciones, este proceso permite visualizar la información en dispositivos inteligentes como los celulares o computadoras que sean compatibles con la aplicación [33, p. 21].

2.1.8.7. Capa 7 - Colaboración y procesos

Para que los usuarios interactúen con la red IoT, la información que las aplicaciones analizan juega un papel clave en la gestión de procesos empresariales, tanto en grandes como en pequeñas empresas y así obtener un mejor control sobre la red IoT [33, p. 21].

2.1.9. Red Wifi

Una red inalámbrica se distingue por su capacidad de permitir la comunicación entre dispositivos sin necesidad de cables, es decir, utilizando ondas electromagnéticas para transmitir datos, este tipo de red emplea antenas tanto para enviar como para recibir información, normalmente el transmisor tiene una sola antena, sin embargo, en ciertas configuraciones se pueden usar múltiples antenas, una antena se encarga de la transmisión y otra de la recepción, aunque también hay antenas que pueden funcionar en ambos modos al mismo tiempo, es decir, transmitiendo y recibiendo datos simultáneamente [34].

2.1.10. Seguridad Wifi

Las redes Wi-Fi, al operar mediante ondas electromagnéticas en lugar de cables físicos presentan desafíos únicos en términos de seguridad, esta naturaleza inalámbrica las hace susceptibles a diversas amenazas como la interceptación de datos, accesos no autorizados y ataques de denegación de servicio, para mitigar estos riesgos se han desarrollado y evolucionado protocolos de seguridad que cifran la información transmitida y autentican a los usuarios, los más destacados se encuentran WEP, WPA, WPA2 y WPA3, cada uno con mejoras significativas en cuanto a robustez criptográfica y mecanismos de autenticación [35].

Para asegurar que la información en redes inalámbricas se mantenga de manera confidencial, íntegra y siempre disponible, es siempre fundamental aplicar correctamente los protocolos de seguridad implicando mantener los dispositivos actualizados y emplear las configuraciones recomendadas para mitigar vulnerabilidades conocidas.

Protocolo	Año de Introducción	Tipo de Cifrado	Vulnerabilidades Conocidas	Estado Actual
WEP	1997	RC4 con IV de 24 bits	Claves fácilmente descifrables; ataques de repetición	Obsoleto
WPA	2003	TKIP (basado en RC4)	Ataques de diccionario y reinyección de paquetes	Desaconsejado
WPA2	2004	AES-CCMP	Exposición a ataques como KRACK	Uso generalizado
WPA3	2018	SAE	Mejora contra ataques de diccionario; adopción lenta	Recomendado

Tabla 2.7: Comparativa de protocolos de seguridad Wi-Fi y su evolución

Fuente: Elaboración propia basada en Salinas Vasquez [36]

2.1.11. Protocolo de Circuito Interintegrado (I2C)

El protocolo de comunicación que se ha implementado para la interacción entre el sensor MAX30102 y el microcontrolador ESP32 en el sistema de monitoreo de presión arterial es el I2C (Inter-Integrated Circuit), el I2C es un protocolo de comunicación en serie que funciona de manera síncrona y permite conectar varios dispositivos en un solo bus, actuando en modo medio dúplex, utiliza dos líneas bidireccionales de drenaje abierto: la línea de datos en serie (SDA) y la línea de reloj en serie (SCL), que son mantenidas a un nivel alto por resistencias [37].

- **SDA (Serial Data Line):** Esta línea se encarga de la transmisión bidirec-

cional de datos entre el ESP32 (como maestro) y el sensor MAX30102 (como esclavo), ambas partes pueden enviar o recibir datos a través de esta línea, según sea requerido durante el ciclo de muestreo.

- **SCL (Serial Clock Line):** Esta línea se encarga en la sincronización de la transferencia de datos, donde el ESP32 actúa como maestro generando la señal de reloj que dicta al sensor el momento de la comunicación, es un proceso esencial para la obtención de lecturas fiables de parámetros biométricos como HR y SpO_2.

La selección del protocolo I2C ha demostrado ser adecuada en este proyecto debido a su facilidad de implementación, eficiencia en el consumo de energía y compatibilidad con múltiples periféricos que facilita su integración en sistemas embebidos como el que se plantea para el monitoreo remoto de signos vitales.

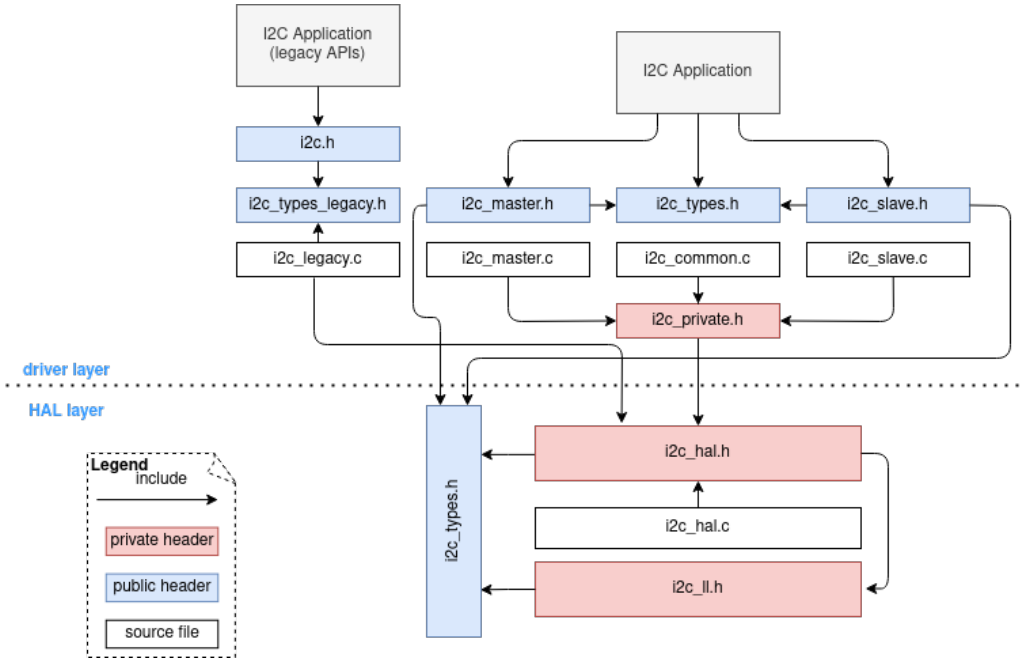


Figura 2.11: Estructura de archivos I2C

Fuente: Espressif Systems [37]

2.1.12. Protocolo IEEE 802.11

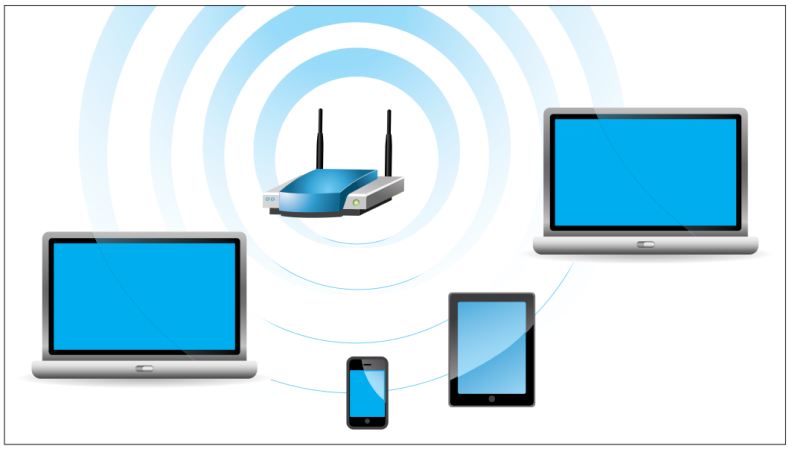


Figura 2.12: Conexión de dispositivos por los estándares 802.11

Fuente: Tektronix [38]

Conocido popularmente como Wi-Fi, el estándar IEEE 802.11 establece el protocolo para las redes locales inalámbricas (WLAN) y ha evolucionado en versiones como 802.11n y 802.11ax (Wi-Fi 6) para ofrecer mayores velocidades, este protocolo es fundamental en aplicaciones IoT ya que permite que un dispositivo que comúnmente opera en la banda de 2.4 GHz se conecte a un router (modo infraestructura) y una vez este sea autenticado de forma segura pueda enviar y recibir paquetes de datos utilizando la suite TCP/IP hacia un servidor en la nube [39].

- **Estación (STA):** Dispositivo que se conecta a la red inalámbrica como ordenadores, smartphones o microcontroladores.
- **Punto de Acceso (AP):** Dispositivo que actúa como puente entre la red inalámbrica y la red cableada gestionando el tráfico de datos entre las estaciones.
- **Sistema de Distribución (DS):** Infraestructura que interconecta múltiples puntos de acceso permitiendio la movilidad de las estaciones entre diferentes áreas de cobertura.
- **Conjunto de Servicios Básicos (BSS):** Conjunto de estaciones que se comunican entre sí bajo la coordinación de un punto de acceso.

2.1.13. IP (Internet Protocol)

El Protocolo de Internet (IP) es el protocolo fundamental de la capa de red del modelo OSI y TCP/IP, su función principal es permitir el direccionamiento lógico y el enrutamiento de paquetes de datos entre dispositivos en una red, cada paquete de datos encapsulado en IP contiene tanto la dirección del remitente como del destinatario, permitiendo su correcta entrega a través de redes locales o globales como Internet [40, pág. 120].

El protocolo IP es operado bajo un esquema sin conexión y sin confiabilidad, lo que significa que no garantiza la entrega total de paquetes, pero proporciona la base sobre la cual otros protocolos como TCP o UDP pueden asegurar la fiabilidad o la velocidad, existen dos versiones ampliamente utilizadas:

- El **IPv4** utiliza direcciones de 32 bits.
- El **IPv6** extiende la capacidad a direcciones de 128 bits para enfrentar el agotamiento del espacio de direcciones disponible.

Estos implementan un papel iportante en la arquitectura de red, es esencial para el identificador lógico de los dispositivos (dirección IP) y para la toma de decisiones de enrutamiento en los nodos de la red.

2.1.14. TCP/IP (Transmission Control Protocol / Internet Protocol)

Figura 2.13: Arquitectura del protocolo TCP/IP en el contexto de IoT

Fuente: EngineersGarage [41]

El conjunto de protocolos TCP/IP se desarrollado originalmente por el Departamento de Defensa de los Estados Unidos, constituye la arquitectura base de Internet y se organiza en capas funcionales aplicación, transporte, internet y acceso a red, donde el Protocolo de Internet (IP) se encarga de direccionar y enrutar los paquetes de datos para que viajen desde un origen, como un sensor ESP32 hasta su destino correcto como a un servidor en la nube, mientras que el Protocolo de Control de Transmisión (TCP) opera sobre IP para establecer una conexión fiable y ordenada, garantizando que la información llegue sin errores, completa y en la secuencia correcta para ser procesada por la aplicación [40, pág. 122].

2.1.15. Protocolo HTTP / HTTPS

El Protocolo de Transferencia de Hipertexto conocido como HTTP es un protocolo de la capa de aplicación que se utiliza para enviar información a través de la World Wide Web, funciona bajo un modelo cliente-servidor donde el cliente puede ser un navegador web o un dispositivo IoT, hace solicitudes de recursos y el servidor responde con los datos que se han solicitado, HTTP es un protocolo sin estado lo que significa que cada solicitud es independiente y no guarda información de solicitudes anteriores [42].

2.1.16. Protocolos y formatos para Machine Learning / Almacenamiento

2.1.16.1. CSV (Comma Separated Values)

El formato CSV es un estándar ampliamente adoptado para representar datos tabulares en texto plano, cada línea corresponde a un registro y los campos están separados por comas que que se pueda manipular de manera directa por parte de herramientas como Excel, MATLAB y librerías Python como pandas para el preprocesamiento de datos, análisis estadístico y entrenamiento de modelos [43].

2.1.16.2. Pickle (PKL)

El módulo nativo de Python "Pickle"permite la serialización de objetos complejos como listas o modelos entrenados en archivos .pkl que es un formato ideal para guardar el progreso del machine learning y evitar reentrenamientos, el proyecto se aplica este concepto para guardar los modelos modelo_sys.pkl y modelo_dia.pkl que deben ser entrenados localmente con datos del sensor MAX30102 para cargarlos en el servidor Flask en Render para estimar la presión sistólica y diastólica a partir de las solicitudes en tiempo real del ESP32 [44].

2.1.17. Accidente cerebrovascular (ACV)

Un accidente cerebrovascular (ACV), también conocido como ictus o apoplejía es una interrupción repentina del suministro de sangre a una parte del cerebro, provocando la muerte de células cerebrales causando un daño permanente siendo una de las principales causas de muerte y discapacidad en todo el mundo, el accidente cerebrovascular (ACV) impacta la calidad de vida a través de severas complicaciones que pueden surgir no solo durante la hospitalización sino también tras el alta médica, por otra parte la posibilidad de prevenir o controlar muchas de estas secuelas recae en una detección tempran, la cual al proporcionar información precisa sobre la naturaleza y frecuencia de dichas complicaciones la convierte en un factor clave para optimizar el tratamiento y mejorar el pronóstico del paciente [45].

2.1.17.1. Accidente Cerebrovascular Isquémico

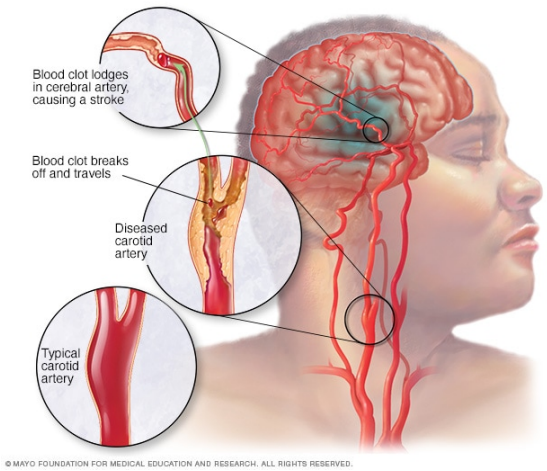


Figura 2.14: Caso común de Accidente Cerebrovascular Isquémico

Fuente: Mayo Clinic [46]

Se produce cuando los vasos sanguíneos del cerebro se estrechan u obstruyen, lo que reduce el flujo sanguíneo (isquemia). Esta obstrucción puede deberse a depósitos de grasa (ateroesclerosis) en las arterias, coágulos sanguíneos que viajan desde el corazón o por otros restos en el torrente sanguíneo, si bien aún se están realizando investigaciones, algunos estudios iniciales indican que la infección por COVID-19 podría aumentar el riesgo de sufrir un ACV isquémico [47].

2.1.17.2. Accidente Cerebrovascular Hemorrágico

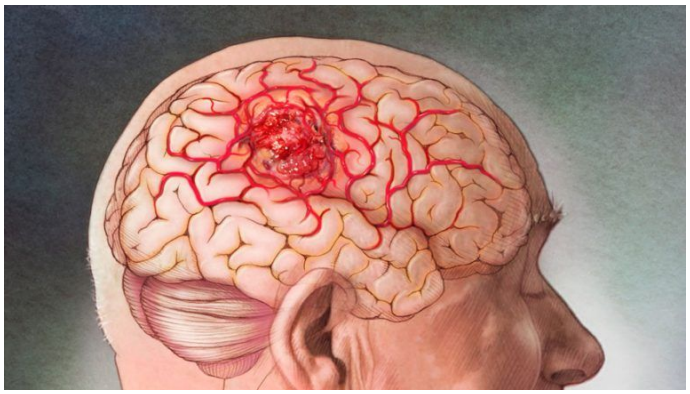


Figura 2.15: Accidente Cerebrovascular Hemorrágico

Fuente: Mayo Clinic [46]

Ocurre cuando un vaso sanguíneo debilitado se rompe y provoca un sangrado dentro o alrededor del cerebro generando un hematoma que aumenta la presión sobre el tejido cerebral y causando un daño directo, este tipo de ACV puede clasificarse como intraparenquimatoso y a menudo asociado con la hipertensión o subaracnoideo

frecuentemente causado por la ruptura de un aneurisma, representando siempre una situación de extrema urgencia médica que requiere intervención inmediata [47].

2.1.18. Presion Arterial

La presión arterial se refiere a la fuerza que la sangre ejerce contra las paredes de las arterias mientras el corazón late, se mide en milímetros de mercurio (mmHg) y se expresa como dos valores:

- **Presión sistólica:** La presión en las arterias se mide cuando el corazón se contrae, en otras palabras, cuando late y bombea sangre hacia el cuerpo, comienza el momento en el que la presión alcanza su punto máximo.
- **Presión diastólica:** La presión en las arterias se mide cuando el corazón está en su fase de relajación, en otras palabras, cuando descansa entre latidos y no está bombeando sangre, comienza el momento en el que la presión es más baja.

La presión arterial es un indicador crucial de la salud cardiovascular ya que a niveles anormalmente altos (hipertensión) o bajos (hipotensión) pueden ser señales de enfermedades graves como la enfermedad cardíaca, accidente cerebrovascular o insuficiencia renal [48].

2.1.18.1. Métodos de medición de la presión arterial

Las mediciones PA son los resultados de estado del sistema circulatorio de un paciente, actualmente existen mediciones para obtener resultados en la PA, categóricamente uno más preciso que otro estructurados en dos tipos de métodos [49].

- **Métodos invasivos:** Los métodos invasivos en la medicina involucran a la introducción de sondas o catéteres dentro del cuerpo humano mediante intervenciones quirúrgicas.
- **Métodos no invasivos:** Los métodos no invasivos se caracterizan por utilizar dispositivos externos como brazaletes o sensores que se colocan en las extremidades del cuerpo.

2.1.18.2. Tabla de mediciones de la presión arterial

Clasificación	PAS (mmHg)	PAD (mmHg)
Normal	< 120	< 80
Elevada	$120 - 129$	< 80
Hipertensión Etapa 1	$130 - 139$	$80 - 89$

Hipertensión Etapa 2	≥ 140	≥ 90
Crisis de Hipertensión	> 180	> 120

Tabla 2.9: Clasificación de las mediciones de la presión arterial
Fuente: Elaboracion propia basada en Whelton, Carey, Aronow et al. [50]

2.1.19. Importancia del monitoreo continuo

Para mejorar el pronóstico clínico de pacientes con accidente cerebrovascular (ACV) y disminuir el riesgo de un nuevo ictus, resulta crucial el monitoreo constante de variables como la presión arterial, frecuencia cardíaca o actividad motora, por lo que el uso de un sistema inteligente con tecnología no invasiva de fotopletismografía (PPG) permite agilizar la recolección de datos en tiempo real, facilitando la identificación temprana de complicaciones y garantizando una intervención médica oportuna. [48].

2.1.20. Lenguaje de scripting

Los lenguajes de scripting son un tipo de lenguaje de programación diseñados para la automatización de tareas específicas, cuyo código es ejecutado directamente por un intérprete sin necesidad de compilarse previamente, lo que los hace ideales para la administración de sistemas, el desarrollo web, la automatización de procesos y la manipulación de archivos y datos [51].

2.1.20.1. Tipos de lenguajes scripting

Lenguaje	**Características**
JavaScript	Ampliamente utilizado en el lado del cliente para interacciones dinámicas en páginas web.
Python	Versátil y popular en desarrollo web, análisis de datos y automatización.
PHP	Principalmente usado en el lado del servidor para crear sitios web dinámicos.
Ruby	Elegante y productivo, se aplica en desarrollo web backend.
Bash	Lenguaje de script para sistemas Unix/Linux.
PowerShell	Utilizado en entornos Windows para automatizar tareas administrativas.
Lua	Popular en desarrollo de videojuegos y aplicaciones embebidas.

Tabla 2.10 – *Continuación*	
Lenguaje	**Características**
Shell scripting	Scripts que se ejecutan en el sistema operativo para automatizar tareas.

Tabla 2.10: Tipos de lenguajes de scripting
Fuente: Elaboracion propia basada en Santander Open Academy [51]

2.1.21. IEEE Std. 1708-2014

El estándar IEEE Std. 1708-2014 establece directrices técnicas para evaluar dispositivos electrónicos que miden la presión arterial sin recurrir a métodos tradicionales como los manguitos inflables o las peras manuales, su objetivo principal es asegurar la precisión y fiabilidad de sistemas alternativos, especialmente aquellos que utilizan sensores colocados sobre la piel como los que se basan en fotopletismografía (PPG). Este marco define parámetros específicos para el diseño, validación y funcionamiento de estos dispositivos fomentando su desarrollo bajo criterios técnicos que sean consistentes y reproducibles [52].

Aunque no impone restricciones sobre las dimensiones o formas físicas de los equipos, el estándar sí requiere que los dispositivos mantengan una precisión dentro de un margen de error de ± 4 mmHg tanto para las mediciones sistólicas como diastólicas, además un nivel de exigencia asegura que los resultados sean confiables en contextos clínicos y estudios científicos, cuando los requisitos se cumplan, los desarrolladores garantizan que sus sistemas no invasivos se alineen con estándares internacionales de calidad, lo que refuerza su viabilidad para el monitoreo continuo y aplicaciones médicas en tiempo real[51].

2.1.22. Sistemas de Gestión de Bases de Datos Relacionales (RDBMS)

sistema de gestión de datos o DBMS (Database Management System) es un software que almacena y gestiona datos basados en tablas mediante filas y columnas, se relaciona a la forma que se almacenan los datos y a su acceso, tales que, permiten las conexiones entre diferentes datos para proporcionar los resultados logrados, como los RDBMS utilizan el lenguaje de SQL (Lenguaje de consulta estructurado) facilitan que los usuarios puedan interactuar con la base de datos y la manipulación como recuperar, actualizar o eliminar datos [53].

Tipo	Descripción	Características principales
MySQL	Sistema de código abierto conocido por su rendimiento, facilidad de uso y amplia adopción en aplicaciones web.	Amplio soporte comunitario, compatibilidad con múltiples plataformas, herramientas de administración como phpMyAdmin.
PostgreSQL	Sistema de código abierto avanzado que soporta características como transacciones ACID, integridad referencial y extensibilidad.	Soporte para JSON y XML, potente motor de consultas, alta conformidad con los estándares SQL.
Microsoft SQL Server	Sistema comercial de Microsoft que ofrece una integración sólida con otros productos de Microsoft y avanzadas capacidades de análisis y reporte.	Integración con el ecosistema Microsoft, herramientas avanzadas de BI y análisis, alta disponibilidad y escalabilidad.
SQLite	Sistema de código abierto, ligero y autónomo que no requiere configuración de servidor, ideal para aplicaciones embebidas y desarrollo móvil.	Sin necesidad de instalación de servidor, excelente para aplicaciones pequeñas y prototipos, soporte para transacciones.
MariaDB	Fork de MySQL desarrollado por la comunidad que incluye mejoras en el rendimiento y características adicionales.	Totalmente compatible con MySQL, mejoras de rendimiento y nuevas características, desarrollo activo y comunitario.
Amazon Aurora	Servicio de base de datos relacional distribuida desarrollado por Amazon, compatible con MySQL y PostgreSQL, y optimizado para la nube.	Escalabilidad y disponibilidad automáticas, compatibilidad con MySQL y PostgreSQL, rendimiento optimizado para AWS.

Tabla 2.11: Tipos de Sistemas de Gestión de Bases de Datos Relacionales (RDBMS)
Fuente: Elaboración propia basada en Rouse [53]

2.2. Marco Contextual

Los Accidentes Cerebrovasculares (ACV) son un grave problema de salud en Ecuador y a nivel mundial, para los pacientes que han sufrido un ACV el monitoreo constante de la presión arterial (PA) es vital, ya que la hipertensión es un factor de riesgo principal del comienzo de una ACV, los métodos tradicionales para medir la PA tienen desventajas ya que no son continuos, pueden ser incómodos y no permiten un seguimiento fácil a distancia que limita la capacidad de actuar preventivamente.

Las tecnologías de telecomunicaciones y el Internet de las Cosas (IoT) abren nuevas puertas para la salud digital (eHealth) utilizando redes inalámbricas (como Wi-Fi IEEE 802.11), protocolos de comunicación de datos y plataformas en la nube posibilitan la creación de sistemas de Monitorización Remota de Pacientes (RPM), estos sistemas permiten que dispositivos con sensores inteligentes recolecten datos biométricos, transmitiendo de forma segura a través de internet para su análisis y que la información sea accesible para médicos e inclusive a los pacientes.

Este presente proyecto desarrolla un "Sensor de Presión Arterial Basado en IoT para el Monitoreo Continuo y Remoto del Paciente con ACV", permitiendo el uso de un sensor no invasivo y un microcontrolador ESP32 que actúa como un nodo de comunicación IoT para capturar datos que se enviarán al servidor en la nube para estimar la PA usando Machine Learning, se lamacenarán los datos para la revisón del médico e incluso el sistema puede enviar alertas a través de canales de telecomunicación móvil como WhatsApp cuando ocurra un evente no deseado, este proyecto no solo busca superar las limitaciones actuales en el monitoreo de PA, sino que también se alinea con metas nacionales e internacionales como el Objetivo 9 de Desarrollo Sostenible de la ONU (Industria, innovación e infraestructura) y el Objetivo 8 del Plan de Desarrollo para el Nuevo Ecuador 2024-2025 ("Impulsar la conectividad como fuente de desarrollo y crecimiento económico y sostenible") aportando desde la ingeniería de telecomunicaciones a la mejora de atención médica y calidad de vida de los pacientes.

Capítulo 3

DESARROLLO DE LA PROPUESTA

3.1. Componentes Tecnológicos de la Propuesta

3.1.1. Sensor MAX30102

Tal como se ilustra en la Figura 3.16, el MAX30102 es un módulo multifuncional que integra un oxímetro de pulso y un sensor de frecuencia cardíaca, el cual capta señales de fotopletismografía (PPG) mediante luz visible para medir los cambios en el volumen sanguíneo y a través de su interfaz I2C proporciona las mediciones de SpO_2 y HR, incorporando además circuitos avanzados para la cancelación de ruido, la optimización energética y la calibración [54].

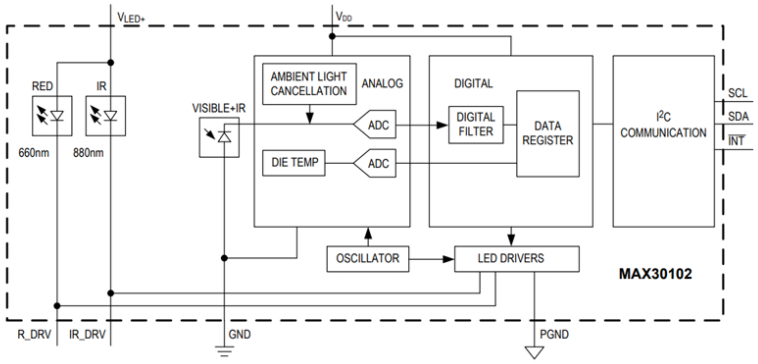


Figura 3.16: Estructura del sensor MAX30102 - fotopletismografía (PPG)

Fuente: Maxim Integrated [55]

3.1.1.1. Principio de Funcionamiento

El funcionamiento del sensor MAX30102 se basa en la técnica de la fotopletismografía (PPG) por reflectancia que es un método óptico no invasivo y esto permite medir las variaciones del volumen de sangre en los tejidos de la piel, este proceso se puede dividir en tres partes importantes:

- **Emisión de luz (LEDs):** El módulo integra dos diodos emisores de luz como

el **LED infrarrojo** que emite luz a una longitud de onda de 880 nm, a esta longitud de onda puede penetrar de manera profunda en los tejidos biológicos y esta la hace ideal porque permite medir las variaciones de volumen sanguíneo que son causadas por el ritmo cardíaco y la variabilidad del pulso, en cambio el **LED Rojo** emite luz a una longitud de onda de 660 nm, lo que hace es que la luz roja sea absorbida de manera diferente por la sangre oxigenada y la desoxigenada, y esta hace una comparación de cuánta luz roja es absorbida en relación con la luz infrarroja que permite que el sistema pueda estimar la saturación de oxígeno en la sangre (SpO2).

- **Interacción con el tejido:** En este proyecto el usuario o el paciente deberá colocar el dedo sobre el sensor, lo que hará que los LEDs iluminen la piel y los vasos sanguíneos porque con cada latido del corazón el volumen de la sangre en las arterias aumenta en momentos, en esos momentos de aumento de volumen de sangre es donde provoca que se absorba más la luz y por lo tanto menos luz es reflejada de regreso hacia el sensor, entonces esta variación rítmica en la luz que se refleja es la señal PPG y su frecuencia es directamente la frecuencia cardíaca del paciente.

- **Recepción y procesamiento en el Chip**

  - **Recepción (Fotodiodo):** Este módulo utiliza un fotodiodo con alta sensibilidad, el fotodiodo está optimizado para capturar la luz reflejada tanto en el espectro visible como en el infrarrojo, lo interasante de este componente es la conviersión de la señal óptica en una pequeña señal eléctrica analógica.
  - **Cancelación de Ruido:** Donde la señal antes de ser procesada pasa por un circuito de cancelación de luz ambiental para evitar que la iluminación del enterno interfiera con la medición.
  - **Digitalización de la señal (ADC):** La señal eléctrica analógica es convertida a un formato digital mediante un *Convertidor Analógico-Digital (ADC)* integrado en el chip.
  - **Filtrado Digital:** Aquí a señal ya digitalizada pasa por un filtro digital interno que elimina aún más el ruido y la prepara para ser leída.
  - **Comunicación (I2C):** Finalmente los datos digitales limpios son almacenados en un registro y quedan disponibles para que el microcontrolador ESP32 los solicite a través del protocolo de comunicación I2C usando los pines SCL y SDA del sensor.

Su respectiva conexión de los pines se muestran en la Figura 3.17, el módulo se conecta principalmente a través de cuatro pines: VIN (que proporciona alimentación a 3.3V), GND (tierra), y los terminales SDA y SCL que corresponden a la interfaz I2C, los demás pines presentados en la figura del dispositivo no son funcionales en el contexto específico para este sistema [56].

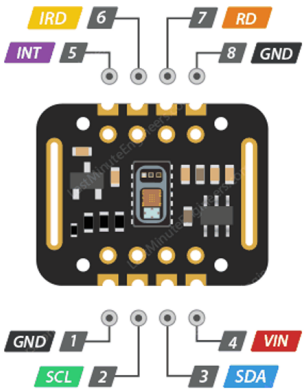


Figura 3.17: Sensor óptico MAX30102 y su pines

Fuente: Maxim Integrated [55]

Elsensor MAX30102 cuenta con las siguientes características mostrada en la siguiente Tabla:

Característica Técnica	**Especificación**
Función Principal	Oxímetro de pulso y monitor de frecuencia cardíaca
Componentes Integrados	LED Rojo, LED Infrarrojo (IR), Fotodetectores
Mediciones	Saturación de oxígeno en sangre (SpO_2), Frecuencia Cardíaca (HR)
Interfaz de Comunicación	I^2C (hasta 400 kHz)
Tensión de Alimentación (Lógica)	1.7V – 2.0V (Típico 1.8V)
Tensión de Alimentación (LEDs)	3.3V – 5.25V
Consumo (Activo)	$<$ 1,0 mA
Consumo (Apagado / Shutdown)	0.7 µA (microamperios)
Características Adicionales	Sensor de temperatura integrado, FIFO de 32 muestras

Tabla 3.12: Características esenciales del sensor MAX30102
Fuente: Basado en la hoja de datos de Analog Devices, Inc. [57].

3.1.2. ESP32 WROOM-32

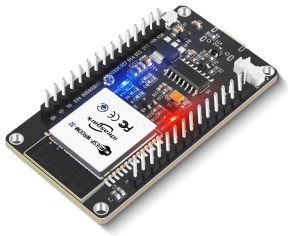


Figura 3.18: Esp32 WROOM 32

Fuente: Punguil, Rojas, Guillen et al. [58]

El ESP32-WROOM-32 es un potente módulo de microcontrolador, creado por Espressif Systems pensado especialmente para aplicaciones de Internet de las Cosas que necesitan conectividad inalámbrica a través de Wi-Fi y Bluetooth, ofrece una variedad de interfaces como UART, SPI, I^2C, ADC y PWM, también incluye características como ese muestra en la Tabla 3.13 avanzadas de bajo consumo y seguridad integrada, su diseño es compacto y versátil lo hace perfecto para sistemas embebidos que se dedican a la recolección y transmisión de datos biomédicos en tiempo real como el monitoreo remoto de la presión arterial utilizando el sensor MAX30102 y el protocolo TCP/IP sobre Wi-Fi [59].

Característica Técnica	**Especificación**
Microprocesador	Xtensa® Dual-Core 32-bit LX7 (Hasta 240 MHz)
Memoria	16 MB Flash, 8 MB PSRAM
Conectividad	Wi-Fi 802.11 b/g/n, Bluetooth 5 (LE)
Tensión de Alimentación	3.3V (Rango: 3.0V – 3.6V)
Consumo (Activo)	70 - 250 mA
Consumo (Hibernación)	10 - 70 µA (microamperios)

Tabla 3.13: Características esenciales del microcontrolador ESP32-S3
Fuente: Basado en la hoja de datos de Espressif Systems [60].

3.1.3. Pantalla LCD ST7789

Figura 3.19: 1.14" INCH TFT IPS BARE HD DISPLAY (ST7789, SPI, 135X240) 13pin Solder Type Flat Cable

Fuente: UGE Electronics [61]

Se muestra la pantalla TFT IPS de 1.14 pulgadas, con una resolución de 135×240 píxeles y un controlador ST7789, al ser un módulo compacto que destaca por su bajo consumo y su impresionante rendimiento visual, siendo perfecta para sistemas embebidos tales como los que se desarrollan con el ESP32-WROOM-32, funciona con una interfaz SPI de 4 hilos y opera a 3.3 V lo que facilita su integración con microcontroladores [61]. Además cuenta con tecnología IPS que proporciona una reproducción de colores excepcional y amplios ángulos de visión convirtiéndola en una opción ideal para mostrar datos biométricos en tiempo real, como la frecuencia cardíaca, el SpO_2 y la presión arterial estimada gracias a su tamaño de la pantalla que es reducida y claridad visual la hace perfecta para dispositivos portátiles de monitoreo médico que necesitan una visualización clara en información de suma vitalidad, en la siguiente Tabla 3.14 tenemos las siguientes carácterísticas:

Característica Técnica	**Especificación**
Tipo de Pantalla	1.14"TFT/LCD con tecnología IPS
Resolución	135 x 240 píxeles
Controlador	ST7789
Interfaz	SPI (Interfaz periférica en serie)
Tensión de Alimentación	3.3 V
Consumo (Típico)	20 mA (Principalmente por la retroiluminación)

Tabla 3.14: Características esenciales de la pantalla integrada
Fuente: Basado en la hoja de datos de Sitronix Technology Corp. [62].

3.1.4. Buzzer

Figura 3.20: Zumbador

Fuente: Pro-Signal [63]

El componente ABT-402-RC, es un transductor magnético pasivo comúnmente conocido como zumbador, que a diferencia de los modelos activos requiere de una señal externa de onda cuadrada como la generada por un pin PWM para poder producir un sonido, siendo crucial tener en cuenta su bajo voltaje de operación de 1.5V.

Característica Técnica	**Especificación**
Tipo de Buzzer	Transductor Magnético Pasivo
Tensión Nominal	1.5V
Rango de Operación	1V – 2V
Corriente Nominal	Máximo 70 mA @ 1.5V
Salida de Sonido	Mínimo 80 dB @ 10cm
Frecuencia Resonante	2048 Hz
Consumo en Reposo	0 mA

Tabla 3.15: Características esenciales del zumbador Pro-Signal ABT-402-RC
Fuente: Basado en la hoja de datos de Pro-Signal [63].

3.1.5. IDEALIFE

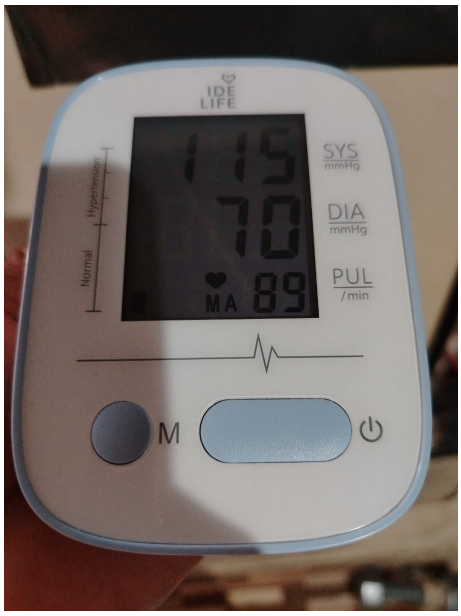


Figura 3.21: Monitor de presión arterial IDELIFE

Fuente: HONSUN (Nantong) Co., Ltd. [64]

Tal como se muestra en la Figura 3.21, el monitor de presión arterial IDELIFE es el dispositivo de referencia utilizado para contrastar y validar la capacidad del prototipo desarrollado, permitiendo evaluar la estimación generadas por el sistema IoT propuesto frente a un método tradicional certificado ya que este equipo es un dispositivo médico digital y automático que, gracias a la Certificación Europea CE, garantiza la fiabilidad y seguridad al medir de forma precisa la presión sistólica, diastólica y el pulso mediante el método oscilométrico estándar, proporcionando las mediciones de referencia necesarias para validar el funcionamiento del modelo de Machine Learning implementado en el prototipo, además el tensiómetro IDELIFE también permite tomar la presión arterial desde la comodidad del hogar mientras interpreta los resultados con la escala de riesgo de la Organización Mundial de la Salud (OMS), cuenta con capacidad para guardar un historial de mediciones para dos usuarios y detección de pulsos irregulares, funcionando tanto con pilas como con adaptador de corriente, por lo que la comparación sistemática entre las mediciones obtenidas con este dispositivo tradicional y las estimaciones del prototipo IoT constituye un elemento fundamental para evaluar la viabilidad clínica del sistema propuesto, estableciendo así un marco de validación que contrasta la innovación tecnológica del monitoreo continuo no invasivo con los métodos convencionales de medición puntual, este dispositivo cuenta con las siguientes características técnicas detalladas en la Tabla 3.16.

Característica Técnica	Especificación
Principio de Medición	Método Oscilométrico
Pantalla	Display digital LCD de gran tamaño

Rango de Medición (Presión)	0 mmHg – 299 mmHg
Rango de Medición (Pulso)	40 – 180 pulsos/minuto
Precisión (Presión)	±3 mmHg
Precisión (Pulso)	±5 % de la lectura
Memoria de Almacenamiento	2 usuarios, con 120 mediciones por usuario
Tamaño del Brazalete	22 – 42 cm (adulto estándar)
Fuente de Alimentación (DC)	5V / 500mA (vía puerto DC)
Fuente de Alimentación (Pilas)	4 pilas alcalinas de 1.5V (Tipo AA)
Indicadores Adicionales	Detector de Pulso Irregular (IHB), Indicador de riesgo OMS
Apagado Automático	Sí (después de 1-3 minutos de inactividad)

Tabla 3.16: Características técnicas del monitor de presión arterial IDELIFE
Fuente: Elaboración propia basada en manuales de HONSUN (Nantong) Co., Ltd. HONSUN (Nantong) Co., Ltd. [64].

3.1.6. Plataforma de Despliegue en la Nube: Render

Figura 3.22: Logo del servidor

Fuente: Render [65]

Es un servidor en la nube que permite a desarrolladores y equipos crear, desplegar y escalar aplicaciones web de cualquier tipo sin necesidad de gestionar servidores o configuraciones complejas, ofreciendo soporte para múltiples lenguajes de programación como Python, JavaScript, Ruby y Go, además de incluir bases de datos administradas, hosting de sitios web estáticos, trabajos programados y características de seguridad empresarial como encriptación automática y protección contra ataques, todo diseñado para que los desarrolladores puedan enfocarse únicamente en escribir código mientras Render se encarga de toda la infraestructura técnica [65].

3.1.7. Servicio de Base de Datos en la Nube: Railway

Figura 3.23: Logo de la plataforma con extensión de base de datos

Fuente: Railway [66]

Es la plataforma que ofrece servicios de bases de datos en la nube que proporciona una infraestructura confiable y escalable para almacenar datos de aplicaciones, permite a los desarrolladores desplegar y gestionar bases de datos sin necesidad de configurar servidores manualmente lo que facilita el desarrollo de aplicaciones modernas. Railway se integra fácilmente con diversas herramientas y lenguajes de programación, lo que la convierte en una opción atractiva para proyectos que requieren una gestión eficiente de datos [66].

3.1.8. Entorno de Desarrollo Integrado: Visual Studio

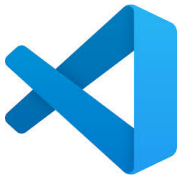


Figura 3.24: Logotipo - Visual Studio

Fuente: Microsoft Corporation [67]

Es un entorno de desarrollo integrado (IDE) completo para desarrolladores de ".NET" y "C++" en Windows, es completamente equipado con una amplia matriz de herramientas y características que mejoran todas las etapas del desarrollo de software. Visual Studio permite a los desarrolladores crear aplicaciones, sitios web y servicios web utilizando diferentes lenguajes de programación [67].

3.1.9. Lenguaje de Programación: Python

Figura 3.25: Logotipo - Python

Fuente: Srinath [68]

Es un lenguaje de programación de uso múltiple de nivel superior y fácil de aprender, caracterizada por su legibilidad, versatilidad, portabilidad e interpretación interactiva. Python es ampliamente utilizado en desarrollo web, análisis de datos, inteligencia artificial y automatización de tareas, y eso lo convierte en una herramienta esencial en el ámbito tecnológico actual [68].

3.1.10. Plataforma de Control de Versiones: GitHub

Figura 3.26: Logotipo - Github

Fuente: GitHub, Inc. [69]

Es una plataforma de hospedaje para control de versiones utilizando Git que facilita en la colaboración y gestión de proyectos de software permitiendo a que los desarrolladores trabajen juntos de manera eficiente en el mismo proyecto sin sobrescribir las contribuciones de los demás, proporcionando una forma confiable de rastrear cambios y facilitar el desarrollo colaborativo [70].

3.1.11. Software de Diseño de PCB: KiCad

Figura 3.27: Logotipo del software para el diseño impreso

Fuente: Medrano, Serra y Soto [71]

Kicad es una suite de software de código abierto para el diseño de esquemas electrónicos y placas de circuito impreso (PCB), sobre todo ofreciendo herramientas para la captura esquemática, diseño de PCB, visualización en 3D y generación de archivos Gerber, lo que permite a los diseñadores crear y validar circuitos electrónicos de manera eficiente [71].

3.1.12. Entorno de Desarrollo: Arduino IDE

Figura 3.28: Logo del entorno de desarrollo integrado de Arduino

Fuente: N. [72]

Arduino IDE es un entorno de desarrollo integrado de código abierto que permite escribir código y cargarlo en embedidos tales como el esp32, el software es compatible con cualquier placa Arduino y proporciona una interfaz sencilla para la programación y depuración de microcontroladores lo que facilita el desarrollo de proyectos electrónicos interactivos [72].

3.1.13. Almacenamiento en la Nube: Google Drive

Figura 3.29: Logotipo - IDE Arduino

Fuente: Farikha [73]

Google Drive es un servicio en línea que permite guardar y compartir archivos de manera segura y accesible desde cualquier dispositivo, es una solución escalable para individuos y equipos de todos los tamaños mejorando la colaboración y el acceso a la información desde cualquier lugar [73, p. 21].

3.1.14. Notificaciones por WhatsApp mediante la API de CallMeBot

Figura 3.30: Servicio gratuita de mensajería por WhatsApp

Fuente: CallMeBot [74]

Es un servicio gratuito que funciona como un puente intermediario para resolver la dificultad de enviar notificaciones a "WhatsApp" desde proyectos personales, basando su funcionamiento en un proceso de configuración inicial donde el usuario autoriza al servicio mediante un mensaje específico para recibir a cambio una clave personal única (apikey), la cual se utiliza después para que un dispositivo como el ESP32 pueda enviar una notificación con solo llamar a una URL de internet, convirtiéndolo en la herramienta ideal para proyectos IoT al evitar la compleja API oficial de WhatsApp y permitir el envío de alertas importantes de forma directa y sin costo [74].

3.1.15. Framework de Desarrollo Web: Flask

Es un microframework de Python para el desarrollo de aplicaciones web, es muy conocido por su simplicidad y su flexibilidad, ayudando a los desarrolladores en la aceleración y escala en el desarrollo de aplicaciones web, proporcionando una base sólida para la creación de aplicaciones web modernas y eficientes.

3.1.16. Tecnología de Aprendizaje Automático: Machine Learning (ML)

ML es una rama de la inteligencia artificial que permite a los sistemas aprender y mejorar automáticamente a partir de la experiencia sin ser programados explícitamente convirtiéndolo en un pilar fundamental de la inteligencia artificial que ha revolucionando diversas industrias al permitir que las máquinas aprendan de los datos y mejoren su rendimiento de manera autónoma.

3.2. Diseño e Implementación del Sistema

3.2.1. Arquitectura General del Sistema

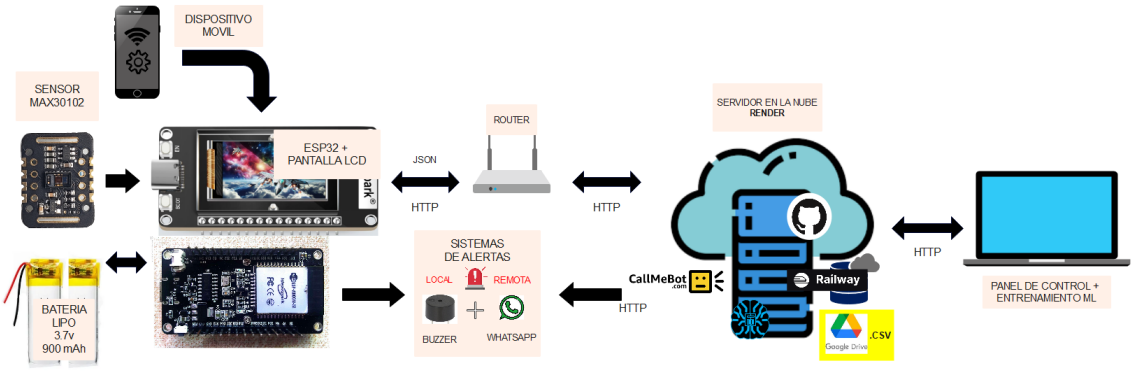


Figura 3.31: Arquitectura General del Sistema de Monitoreo IoT

Fuente: Elaboración propia

Para el sistema se pensó en una arquitectura tipo Internet de las Cosas (IoT) como se muestra en la Figura 3.31, formada por varias capas y distribuida para que todo funcione en tiempo real, está pensado para un crecimiento a futuro y que se pueda acceder desde cualquier lugar, su flujo de datos y las operaciones entre las partes más importantes funciona de la siguiente manera:

- **Nodo Sensor (Dispositivo IoT):** El dispositivo que lleva la persona usa un microcontrolador ESP32 para tomar las señales fotopletismográficas (PPG) con un sensor MAX30102 y ahí mismo hace un primer cálculo para sacar la frecuencia cardíaca (HR) y la saturación de oxígeno (SpO2).
- **Comunicación Inalámbrica:** El ESP32 agarra los datos del sensor, los mete en un paquete JSON y los manda por Wi-Fi con una petición HTTP POST a una dirección del servidor, usando el protocolo TCP/IP para que la información llegue segura.
- **Backend en la Nube (Servidor Flask):** El servidor en la nube, que es una aplicación Flask en la plataforma Render, es el cerebro de todo, ya que recibe los datos y de inmediato usa los modelos de machine learning para adivinar la presión sistólica y diastólica, clasifica el resultado para ver si hay riesgo y guarda toda la medición en una base de datos MySQL en Railway para tener un historial.
- **Retroalimentación y Monitoreo:** El sistema le da una respuesta al usuario de dos formas al mismo tiempo, mandando los resultados de vuelta al dispositivo para que se vean en su pantalla y transmitiendo los datos por un canal Socket.IO para que el panel web se actualice al instante sin tener que recargar la página.

- **Gestión de Entrenamiento:** Desde la página web, un usuario con permiso puede poner el sistema en modo de entrenamient, los datos del sensor se guardarán temporalmente para después combinarlos con valores de referencia puestos a mano con la finalidad de generar un archivo CSV que se subirá a Google Drive para reentrenar los modelos más adelante.

La siguiente Figura 3.32 presenta el diagrama de conexión planteado

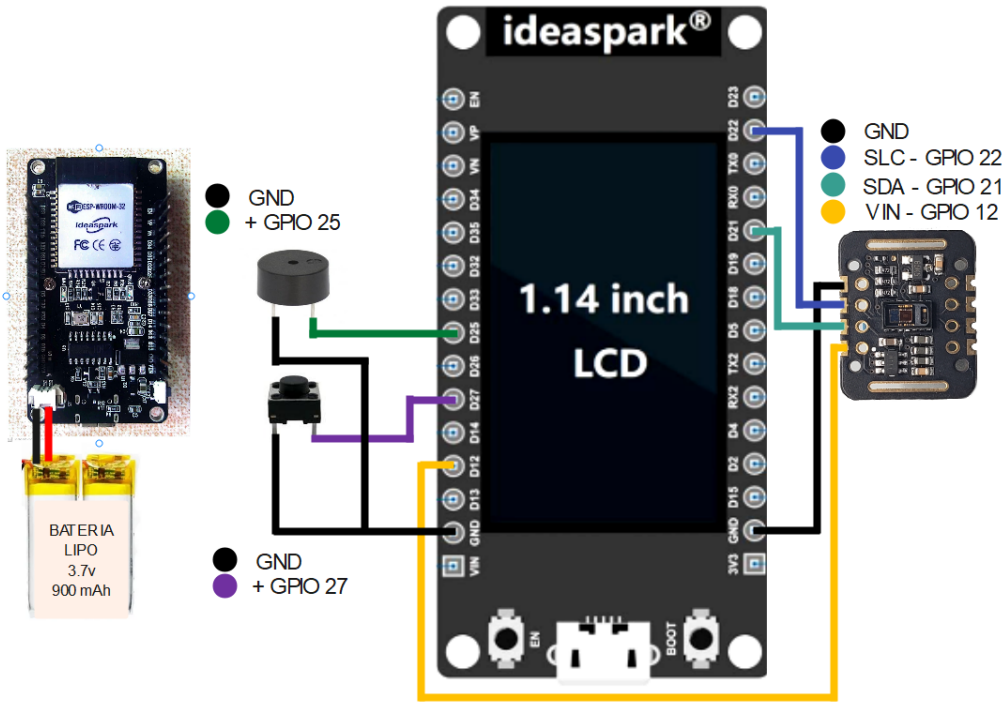


Figura 3.32: Diagrama de conexión del dispositivo

Fuente: Elaboración propia

Para ilustrar este flujo de datos de una manera más clara y secuencial, se desglosa la arquitectura completa en una serie de diagramas que cuentan el "proceso" de cómo viaja la información, cuando el dispositovo del paciente captura la señal hasta que es procesada en la nube y devuelta como una respuesta estimada, a este proceso se la puede dividir en cuatro etapas lógicas principales:

- **Recolección y Envío de Datos:** Se enfoca en el trabajo del Nodo IoT, que captura las señales biométricas del paciente y las transmite a la nube.
- **Procesamiento y Estimación con ML:** Este proceso describe la lógica del servidor al recibir los datos aplicados a los modelos de Machine Learning para estimar los valores de presión arterial y clasificar el riesgo.
- **Almacenamiento y Sistema de Alertas:** Aquí se detalla las acciones que toma el servidor después de la predicción como guardar los datos en la base de datos y enviar notificaciones de emergencia si es necesario.

- **Retroalimentación y Monitoreo Remoto:** Muestra cómo los resultados finales son entregados de vuelta al usuario a través de dos canales: la pantalla del dispositivo físico y el panel de monitoreo web.

El diagrama de flujo general que resume esta interacción se representa del siguiente modo en la Figura 3.33.

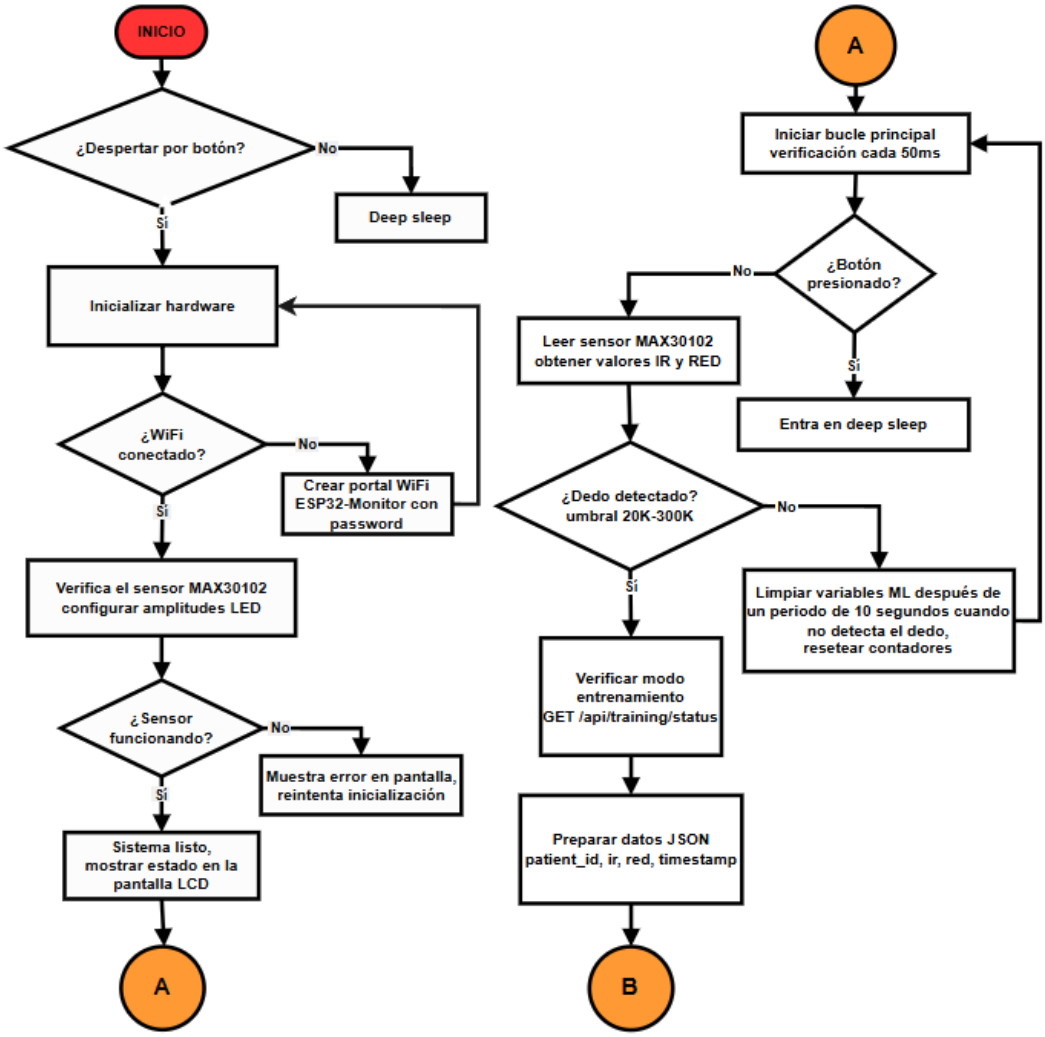


Figura 3.33: Diagrama de flujo de inicialización, configuración esp32, captura de datos y procesamiento esp32

Fuente: Elaboración propia

Continuación de la siguiente parte del diagrama de flujo 3.34.

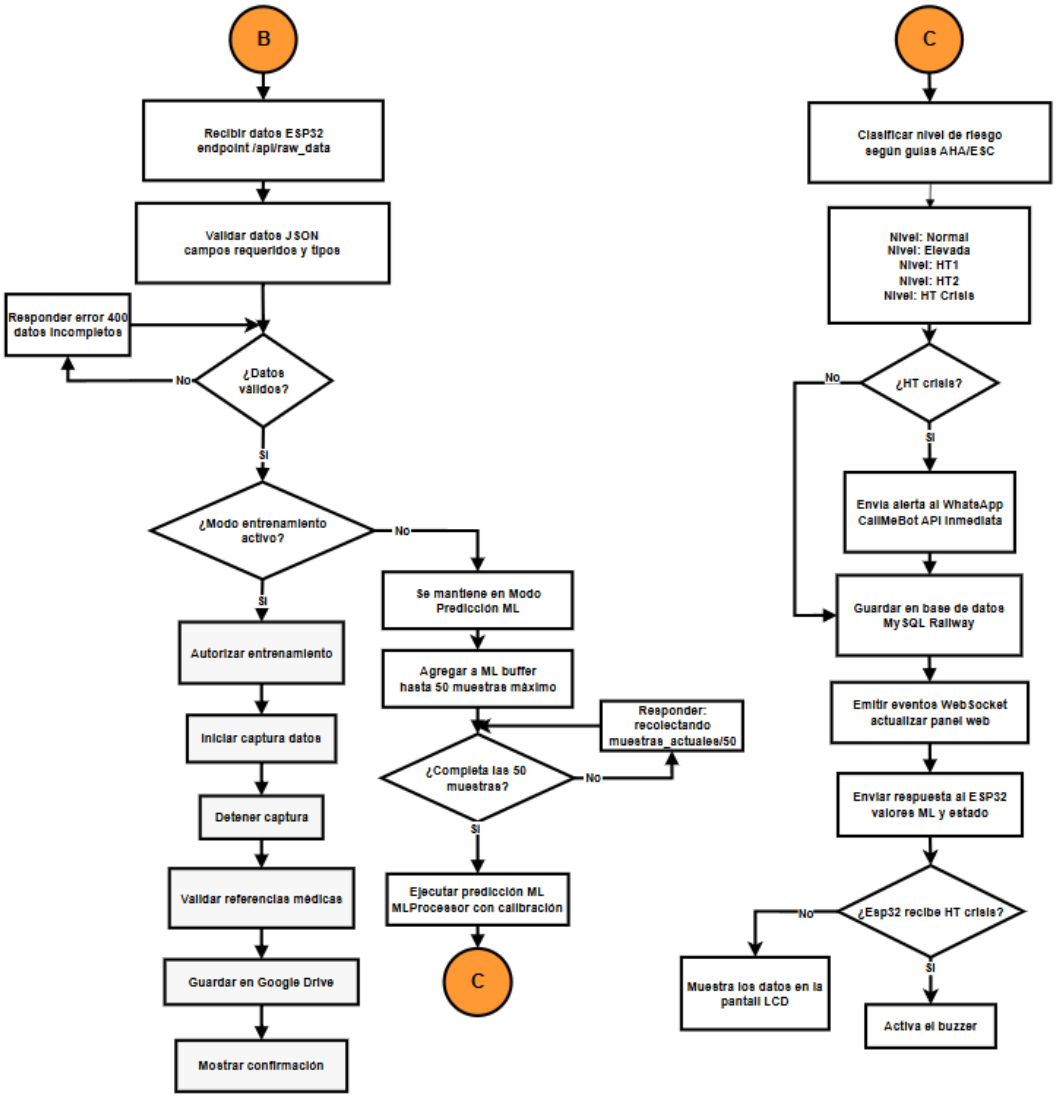


Figura 3.34: Diagrama de flujo de servidor: Recepción, Procesamiento, Clasificación y Respuestas

Fuente: Elaboración propia

3.2.2. Diseño y Construcción del Hardware (Nodo IoT)

3.2.2.1. Análisis del Consumo Energético

Para que el dispositivo estime los valores de la PA de manera continua, el uso y la duración de la batería también es fundamental, por lo que es importante conocer cuánta energía va a consumir cada pieza en el prototipo, se realizó una investigación de las hojas de datos técnicas que los propios fabricantes publican, a continuación se presenta la siguiente Tabla 3.17 con información organizada para mostrar de forma clara cuánto consume cada componente, incluyendo cuando estos están en reposo como cuando está funcionando activamente.

Componente	Consumo Mínimo	Consumo Promedio (Activo)
ESP32	$\sim$ 0,02 mA / reposo	$\sim$ 80 mA / con Wi-Fi
Sensor MAX30102	$\sim$ 0,001 mA / apagado	$\sim$ 1 mA / midiendo
Pantalla LCD	$\sim$ 5 mA / sin luz de fondo)	$\sim$ 40 mA / con luz de fondo
Buzzer	0 mA / en silencio	$\sim$ 30 mA / sonando

Tabla 3.17: Tabla de Consumos Eléctricos Estimados

Fuente: Elaboración propia

3.2.2.2. Selección de Componentes

El ESP32 fue elegido principalmente por su gran potencia para procesar datos y por su conectividad Wi-Fi integrada que es fundamental para enviar las mediciones a internet, mientras que el sensor MAX30102 es la opción ideal por ser un sensor óptico de reflectancia que utiliza longitudes de onda de luz roja e infrarroja para obtener la señal de fotopletismografía (PPG) que es fundamental para analizar la presión arterial, el buzzer se incluye como un sistema de alarma simple como a la vez efectivo para notificar valores anómalos.

3.2.2.3. Diseño Esquemático y Conexiones Eléctricas

Para la validación funcional del hardware, inicialmente se realizó un montaje del circuito en un protoboard que permitió comprobar en la práctica la correcta interacción entre todos los componentes, se aprecia el prototipo en la siguiente Figura 3.35.

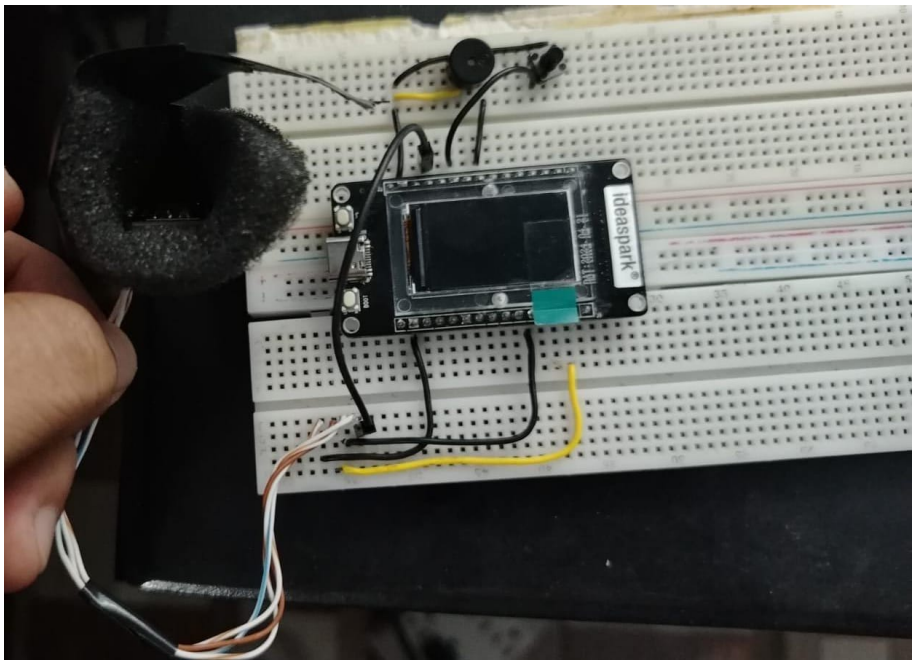


Figura 3.35: Implementación del prototipo físico en la placa de pruebas (protoboard)

Fuente: Elaboración propia

Una vez verificado su funcionamiento, se procedió a diseñar el esquemático formal en el software KiCad, presentado en la Figura 3.36, el cual define las interconexiones precisas para la futura placa de circuito impreso (PCB). La comunicación entre el microcontrolador ESP32 y los periféricos es fundamental y se detalla en la Tabla 3.18.

Componente / Función	**Interfaz / Protocolo**	**Pines del ESP32 Utilizados**
Sensor Óptico MAX30102	I2C (Inter-Integrated Circuit)	GPIO 21 (SDA) GPIO 22 (SCL)
Pantalla TFT ST7789	SPI (Serial Peripheral Interface)	GPIO 15 (CS) GPIO 2 (DC) GPIO 4 (RST)
Alerta Audible	Control Digital Directo	GPIO 25 (Buzzer)
Interacción de Usuario	Entrada Digital	GPIO 27 (Botón Pulsador)
Gestión de Energía del Sensor	Control Digital (Software Toggle)	GPIO 12 (Sensor Power)

Tabla 3.18: Tabla de conexiones eléctricas entre el ESP32 y los periféricos.

Fuente: Elaboración propia

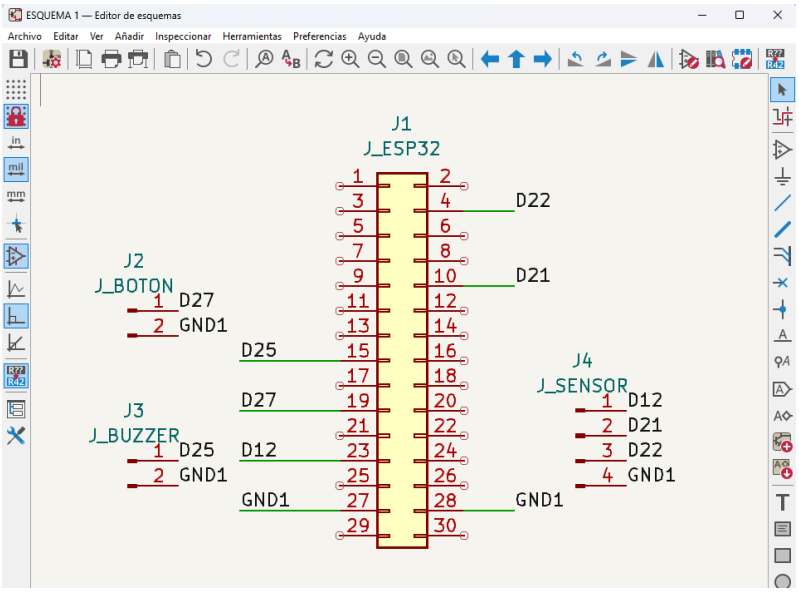


Figura 3.36: Esquemático en KiCad con anotaciones que señalan las interfaces I2C y SPI.

Fuente: Elaboración propia

3.2.2.4. Diseño y Prototipado de la Placa de Circuito Impreso (PCB)

Con el plano listo, diseñamos una placa PCB de dos capas para que todo entrara de forma compacta, cuidando que las pistas de datos I2C y SPI fueran cortas y sin ruidos para no perder calidad en la señal, además de usar conectores JST y headers para que fuera fácil de armar y reparar, y hasta hice una visualización de 3D en KiCad para asegurar de que todo encajara bien antes de fabricar el prototipo, en la Figura 3.37 se visualiza la arquitectura en 2D, en la Figura 3.38 se presenta el diseño en 3D y en la Figura 3.39 el diagrama de flujo del proceso de este diseño y prototipado.

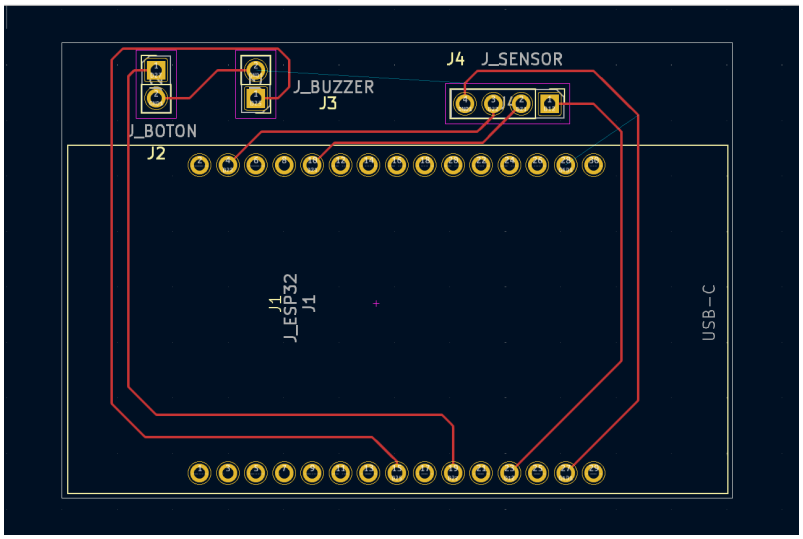


Figura 3.37: Visualización 2D de la PCB en KiCad

Fuente: Elaboración propia

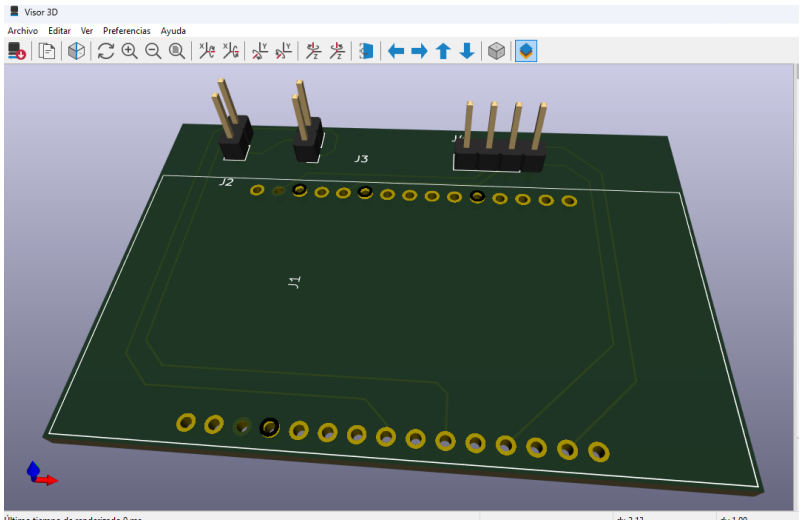


Figura 3.38: Visualización 3D de la PCB en KiCad

Fuente: Elaboración propia

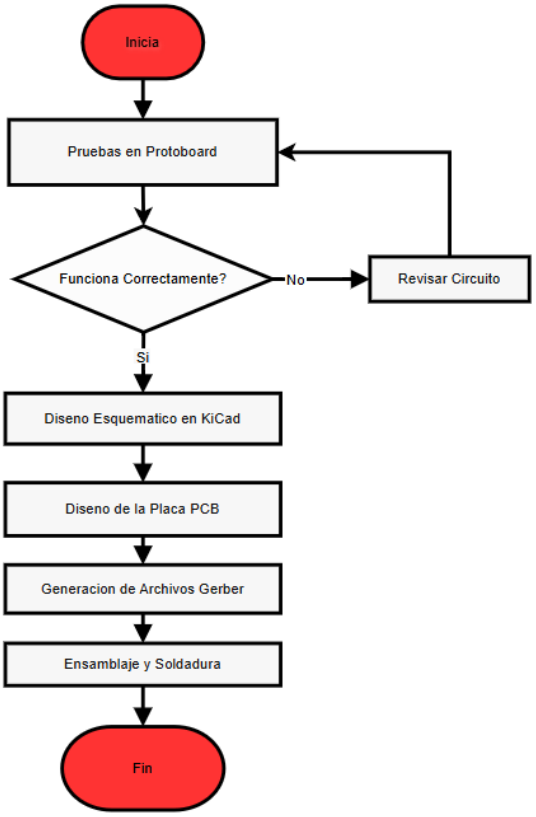


Figura 3.39: Diagrama de flujo del proceso de diseño y prototipado del hardware.

Fuente: Elaboración propia

3.2.3. Desarrollo e Implementación del Software

El software es el corazón del sistema porque permite que el hardware, la nube y el usuario trabajen juntos de forma coordinada logrando que el proyecto funcione de manera correcta y eficiente, es posibleporque se diseñó una arquitectura en tres secciones conectadas a la red haciendo que el sistema modulado y escalable:

3.2.3.1. Firmware del Microcontrolador (ESP32)

El firmware se programó en C++ con Arduino IDE, usando librerías como WiFi.h y HTTPClient.h para la red y otras específicas como Adafruit_ST7789.h y MAX30105.h para el hardware de modo que al prenderse el equipo inicializa todo en orden conectándose al Wi-Fi y configura el sensor, para luego en su bucle principal leer cada 250 milisegundos las señales del sensor, calcula el HR y SpO2 filtrando las últimas lecturas para más estabilidad, y solo si la señal es buena empaqueta los datos en un JSON y los manda por HTTP POST al servidor, finalmente recibe la respuesta con las predicciones y muestra toda la información en la pantalla, usando colores para resaltar las alertas de riesgo, en la Figura 3.40 se visualiza en el diagrama de flujo desde la captura hasta la comunicación.

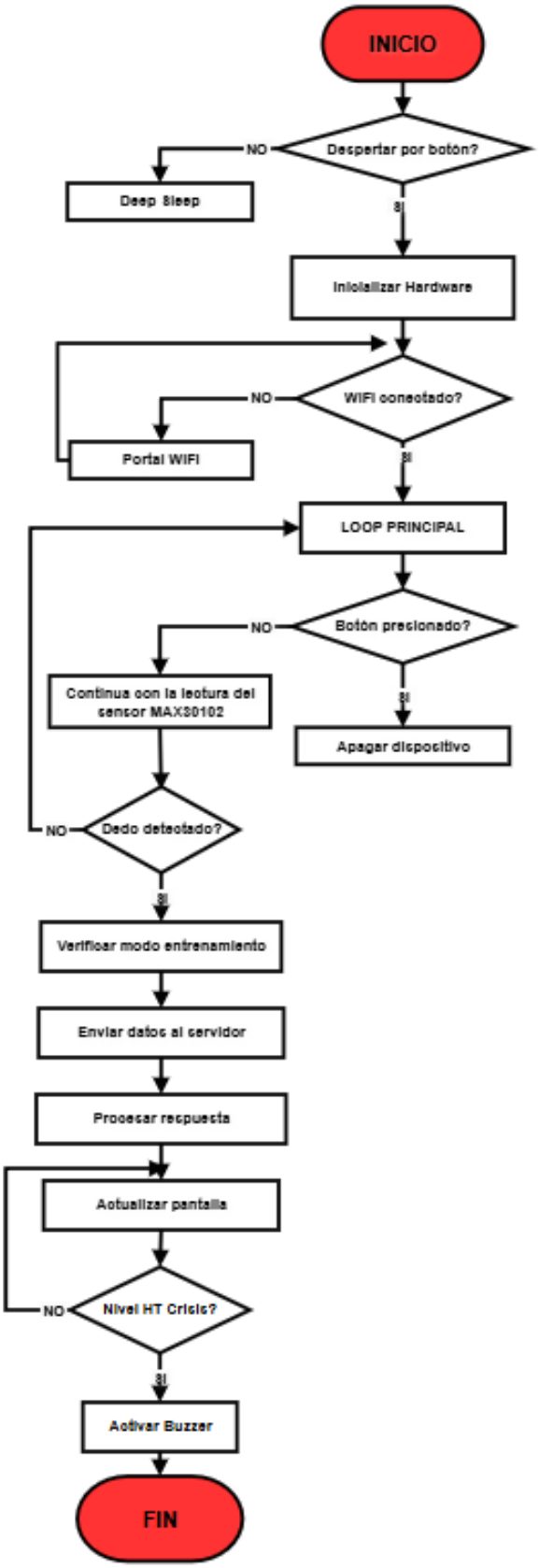


Figura 3.40: Diagrama de Flujo del Firmware del ESP32

Fuente: Elaboración propia

3.2.3.2. Aplicación del Servidor (Backend en Flask)

El backend, hecho en Python con Flask y desplegado en Render, es el centro de operaciones que usa librerías definidas en un archivo "requirements.txt" y su endpoint principal /api/data recibe el JSON, lo valida y si no está en modo entrenamiento usa los modelos de machine learning para predecir la presión, guarda el registro en la base de datos cada 5 segundos para no saturarla, manda una alerta por WhatsApp a través de CallMeBot si detecta una crisis y además gestiona el modo de entrenamiento guardando los datos en un buffer para luego subirlos a un CSV en Google Drive cuando el usuario autorizado lo indica desde la web, a continuación su diagrama de flujo en la Figura 3.41.

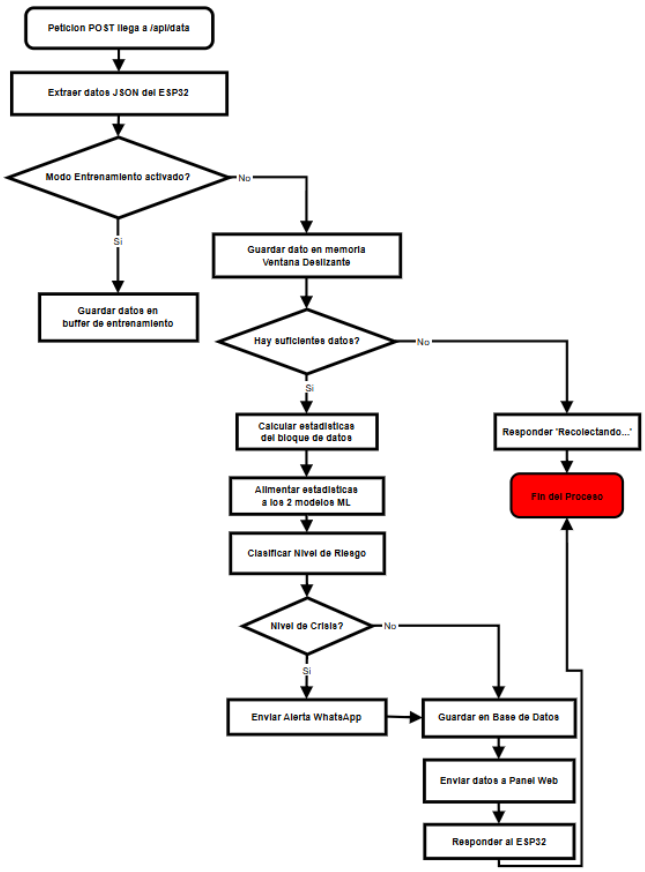


Figura 3.41: Diagrama de Flujo de la Aplicación del Servidor Flask

Fuente: Elaboración propia

3.2.3.3. Interfaz de Usuario (Frontend Web)

La página web se hizo con HTML, CSS y JavaScript, donde el archivo index.html organiza la interfaz y el código JavaScript hace que los botones funcionen, como el de iniciar captura que se comunica con el servidor, mientras que la magia del tiempo real ocurre gracias a Socket.IO, que actualiza los datos en la pantalla sin recargar la página cada vez que el servidor le avisa de una nueva medición o un nuevo registro guardado.

3.2.4. Configuración de la Infraestructura y Despliegue en la Nube

Para que el sistema de monitoreo funcionara desde cualquier parte (remota) se armó una infraestructura en la nube conectando varios servicios especializados, y para manejar todo el proceso, usamos un repositorio en GitHub como el único lugar oficial para todo el código del backend.

3.2.4.1. Creación del Repositorio en GitHub

- **Paso 1:** *"El primero paso y más iportante es crearse una cuenta en GitHub para crear el repositorio"*. Creamos el repositorio en la platarforma de Github en la "Dashboard" y luego en "New".

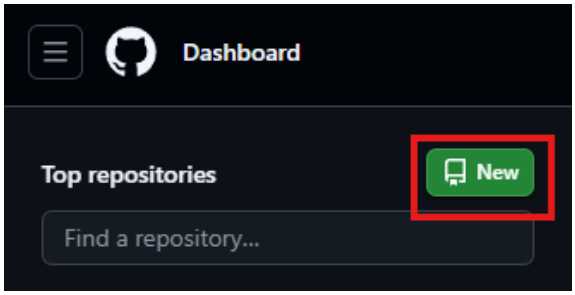


Figura 3.42: Repositorio Github creado

Fuente: Elaboración propia

- **Paso 2:** Al crear un nuevo repositorio nos perdirá que le asignemos un nombre, luego le damos en "Create Repository".

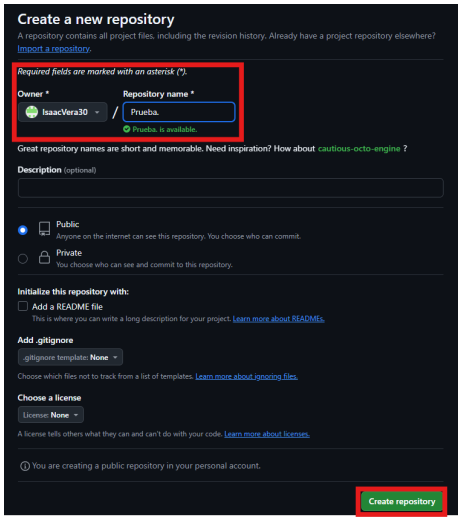


Figura 3.43: Crear Repositorio

Fuente: Elaboración propia

- **Paso 3:** Una vez asignado el repositorio debemos modificar la rama que por

defecto nos da Github “main”, luego ingresamos en “View all branches” que es el menú de la rama.

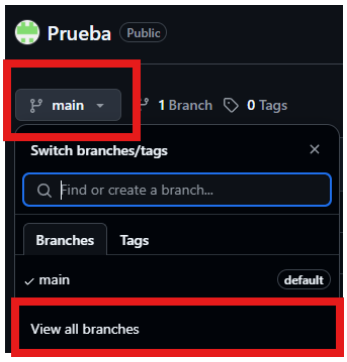


Figura 3.44: Ingreso al menú de la rama

Fuente: Elaboración propia

Por último en el menú de la rama editamos el nombre que tiene por defecto (opcional), dando click en los tres punto horizontales y seleccionando “Rename Branch”, donde nos abrirá una ventana y editamos el nombre a “master” seguido de “Rename branch”.

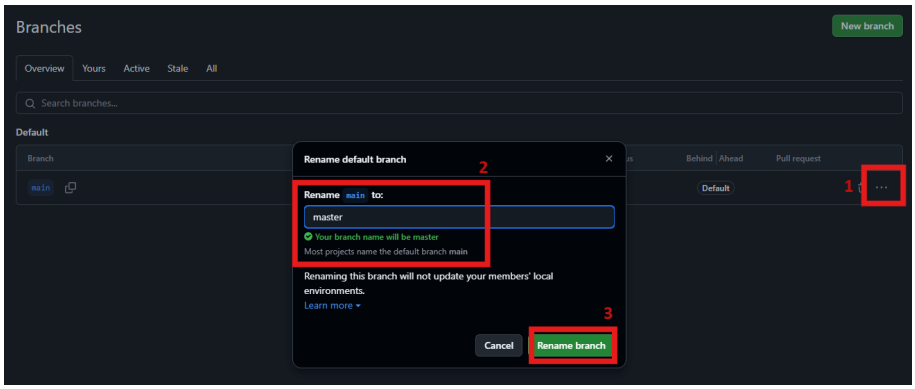


Figura 3.45: Modificación del nombre de la Rama

Fuente: Elaboración propia

Una vez que el repositorio este listo y creado, se procederá con la configuración de la plataforma Render para el despliegue.

3.2.4.2. Vinculación del GitHub con Render

- **Paso 1:** Al ingresar a la plantaforma en “Render”, Figura 3.46, podemos iniciar sesión directamente con GitHub, seguido autorizar permiso a render, Figura 3.47.

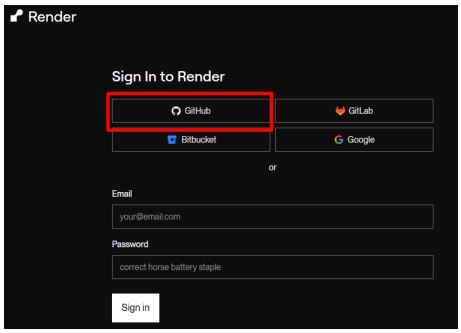


Figura 3.46: Opciones de sesión en Render

Fuente: Elaboración propia

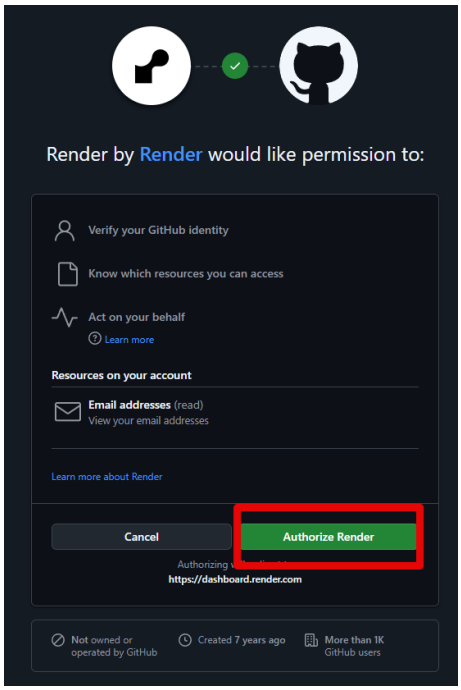


Figura 3.47: Autorización de permisos en Render

Fuente: Elaboración propia

- **Paso 2:** Inicialmente en el panel de Render nos pedirá que debemos crear un nuevo servicio, creamos el nuevo servicio en "Servicio Web".

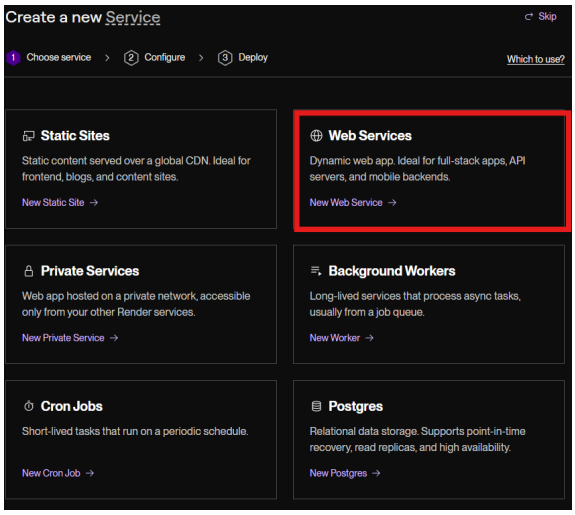


Figura 3.48: Creación de Servicio Web

Fuente: Elaboración propia

Continuando con el proceso configuramos e implementamos el nuevo servicio web, donde la plataforma nos solicita la selección del código fuente mediante la conexión con proveedores de Git como GitHub, esta selección es necesaria para que el servidor acceda al repositorio creado y así continuar con el despliegue de la aplicación.

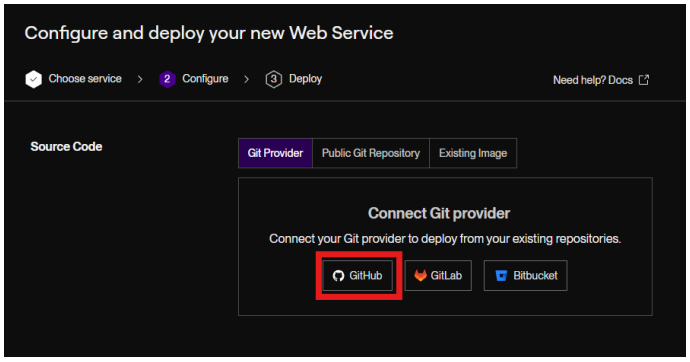


Figura 3.49: Selección del proveedor GitHub

Fuente: Elaboración propia

Continuando, ya seleccionando nuestro proveedor le damos **nombre** a nuestro servidor web **"flask_server"**, asignamos el **nombre al proyecto** "Tesis" y el **nombre de entorno** "Ptesis" (para aislar y gestionar las configuraciones y dependencias de la aplicación del proyecto, sin interferir con otros entornos o proyectos), como la aplicación de servidor (backend) está desarrollada en Python y usa el framework Flask colocaremos el **lenguaje** en "Python3", luego seleccionamos la **rama** que le asignamos al repositorio "master".

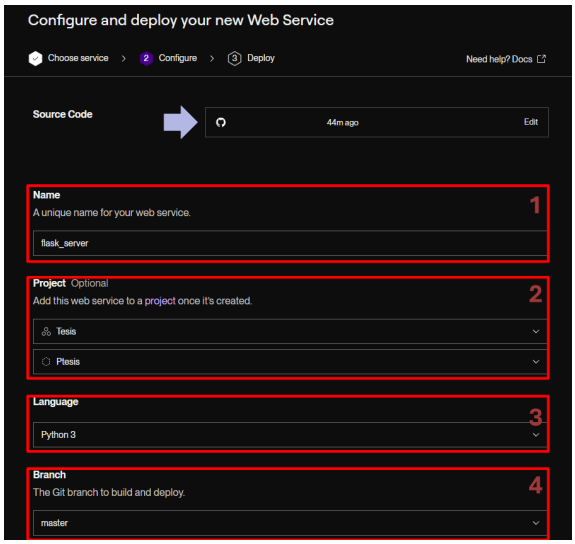


Figura 3.50: Configuración y despliegue del servicio web en Render - Parte 1

Fuente: Elaboración propia

Continuando, es muy importante definir bien los comandos de despliegue, así que para el **comando de compilación** especifiqué "pip install -r requirements.txt" para que se instalaran todas las librerías necesarias y para el **comando de inicio** usé "gunicorn –worker-class eventlet -w 1 app:app", eligiendo Gunicorn por ser robusto para producción y eventlet para que funcionara bien con las conexiones en tiempo real de "Socket.IO", luego seleccionamos **proyectos para aficionados** que nos brindra de forma gratuita una ram de 512 MB, Figura 3.51. Finalmente **implementamos el servicio web**, Figura 3.52.

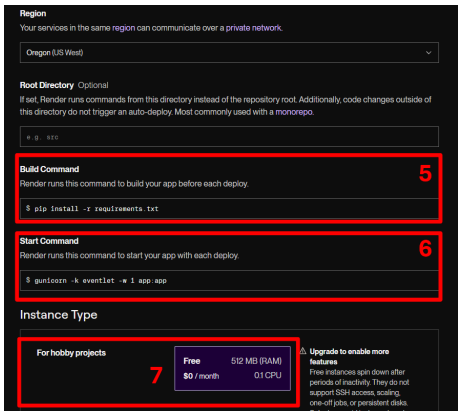


Figura 3.51: Configuración y despliegue del servicio web en Render - Parte 2

Fuente: Elaboración propia

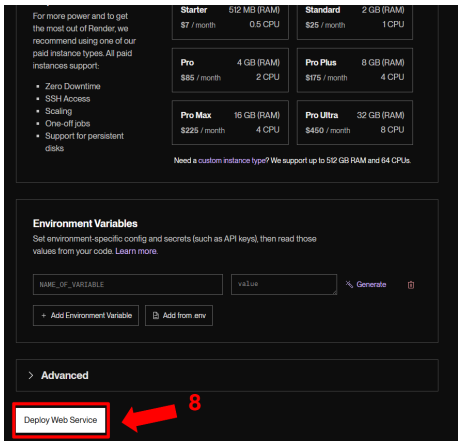


Figura 3.52: Configuración y despliegue del servicio web en Render - Parte 3

Fuente: Elaboración propia

3.2.5. Configuración de la Base de Datos en Railway y vinculación con Render

3.2.5.1. Creación de la base de datos

Para guardar las mediciones de forma permanente tomé la decisión de separar la base de datos del servidor y usar un servicio gestionado en la nube de manera eficiente como escalable .

- **Paso 1:** De igual manera podemos interactuar con Railway vinculando la cuenta con GitHub, Figura 3.53. Luego autorizando los permisos de la plataforma, Figura 3.54.

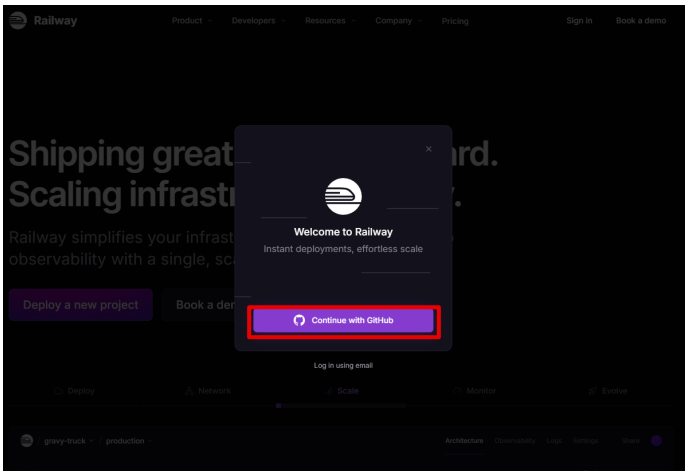


Figura 3.53: Presentación del Railway para vincular con Github

Fuente: Elaboración propia

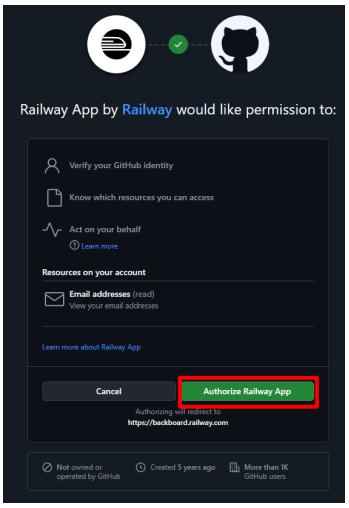


Figura 3.54: Autorización de permisos de Railway

Fuente: Elaboración propia

- **Paso 2:** La plataforma Railway nos mostrará una ventana para crear nuevos proyectos, es aquí donde **implementamos MySQL** para la base de datos.

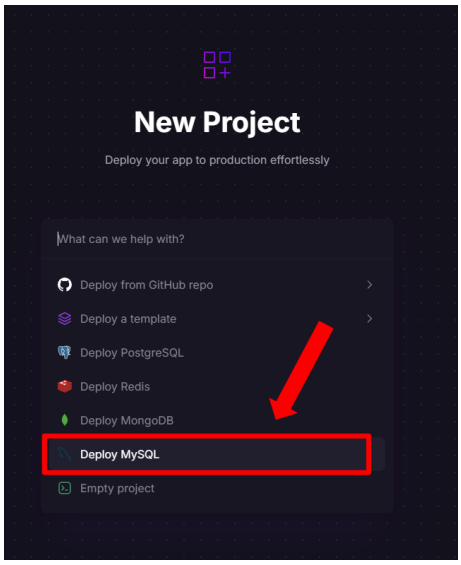


Figura 3.55: Implementación de MySQL en Railway

Fuente: Elaboración propia

- **Paso 3:** Al implementar MySQL la plataforma nos presentra otra pantalla con varias ventanas, aquí elegimos la ventana “Data” en donde craremos la tabla.

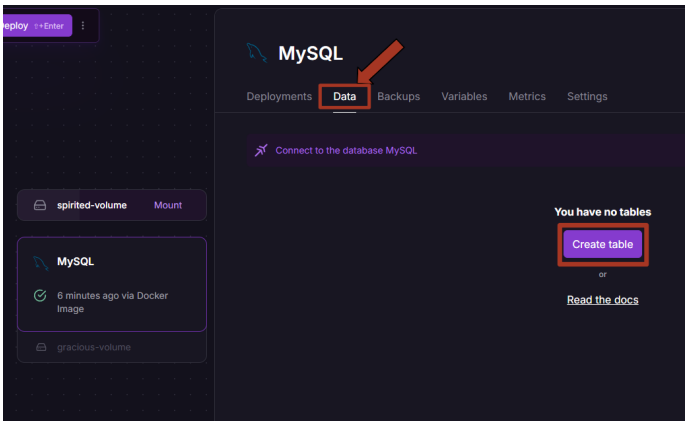


Figura 3.56: Opciones en MySQL

Fuente: Elaboración propia

- **Paso 4:** Luego creamos la tabla de nombre "mediciones" y agregamos las siguientes variables correspondientes en "nombres de columnas", "tipo" de la variable y su "constraints" (Not NULL sirve para que las celdas nunca estén vacías), finalmente **creamos la tabla**.

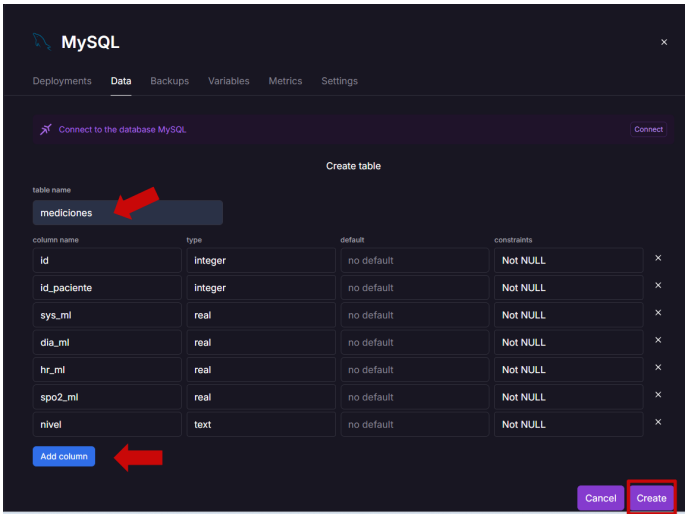


Figura 3.57: Creación de la Tabla de mediciones

Fuente: Elaboración propia

3.2.5.2. Vincular Railway con Render

- **Paso 1:** Para vincular estas plataformas, hay que dirigirse en la ventana de MySQL e ir a la pestaña "Variables", donde copiaremos las credenciales y las llaves que trae por defecto la plataforma, las variables a copiar junto con las llaves se puede visualizar en la siguiente Figura 3.58.

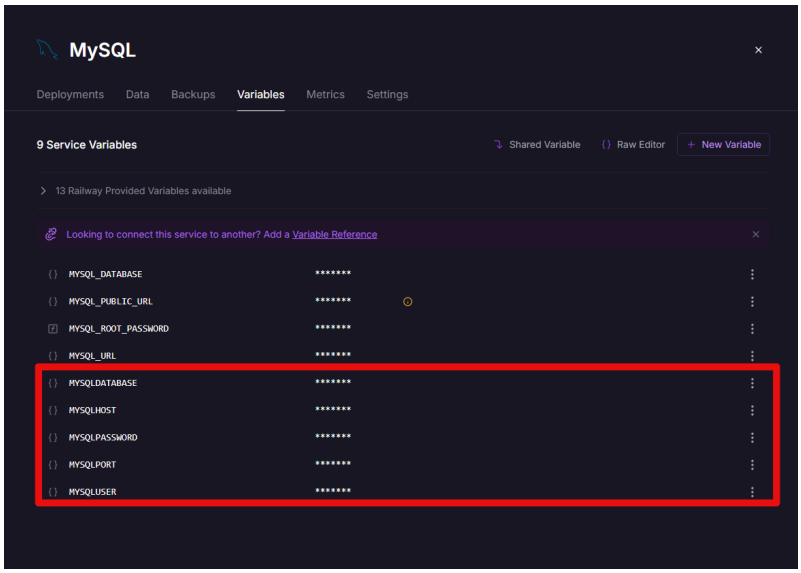


Figura 3.58: Tabla de variables dentro de Railway

Fuente: Elaboración propia

- **Paso 2:** Para que el servidor se comunicara de forma segura con la base de datos se copió las credenciales de Railway como variables de entorno dentro del servicio de Render, para agregar las vaiables nos a la sección de "Manage" y seleccionamos "Environment" donde **editaremos las variables de entorno**, Figura 3.59, por último

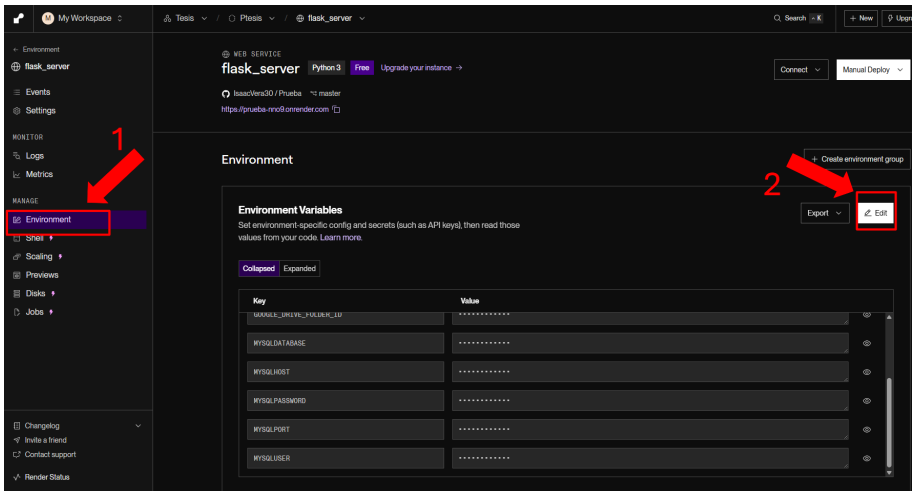


Figura 3.59: Editar y agregar las variables de entorno para Railway en Render

Fuente: Elaboración propia

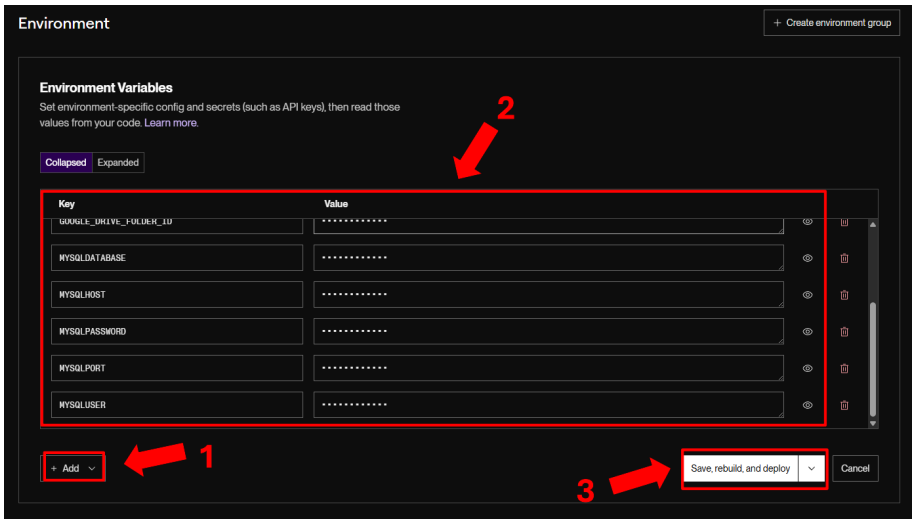


Figura 3.60: Configuración de las variables de entorno para Railway en Render

Fuente: Elaboración propia

3.2.5.3. Integración de APIs de Terceros (Google Drive y CallMeBot)

Las funcionalidades de almacenamiento de datos para entrenamiento y el envío de alertas críticas requerían la integración con APIs externas, lo cual se manejó de la siguiente manera:

- **Google Drive API:**
  - Primero, creamos un proyecto en la consola de Google Cloud y habilitamos la API de Google Drive para ese proyecto, figura 3.61.

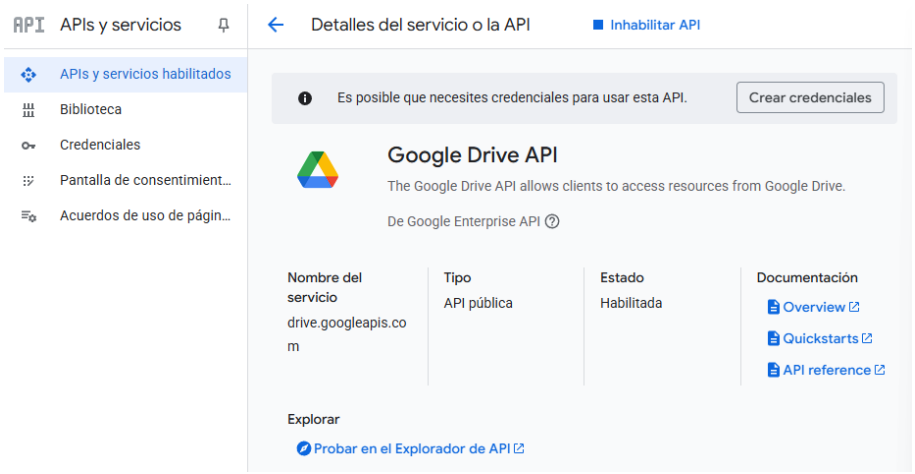


Figura 3.61: API Habilitada

Fuente: Elaboración propia

  - Luego, generamos una "Cuenta de Servicio" (Service Account), Figura 3.62. Este tipo de cuenta actúa como un usuario robot, y Google

nos proporcionó un archivo de credenciales en formato JSON "service_account.json" para autenticarla como se meustra en la Figura 3.63 para subir al repositorio.

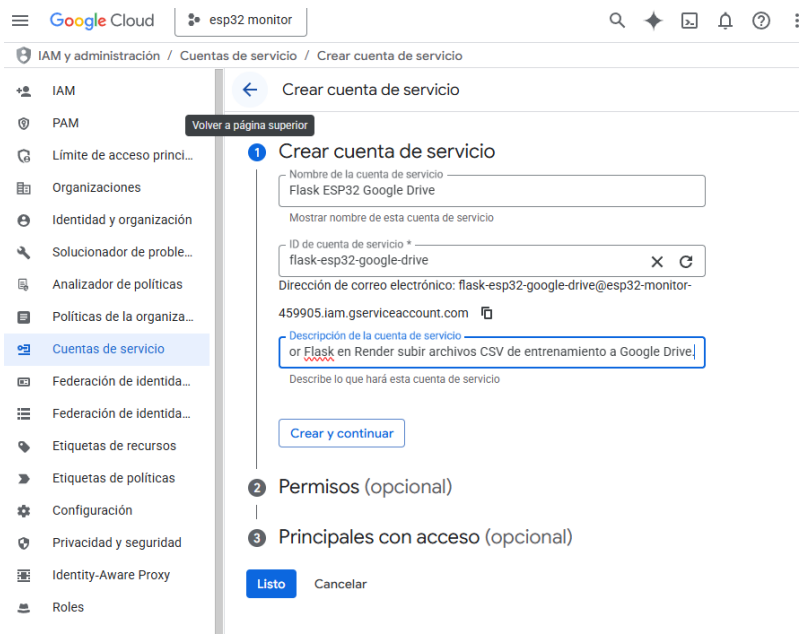


Figura 3.62: Esta imagen ilusta los pasos para crear una Cuenta de Servicio

Fuente: Elaboración propia

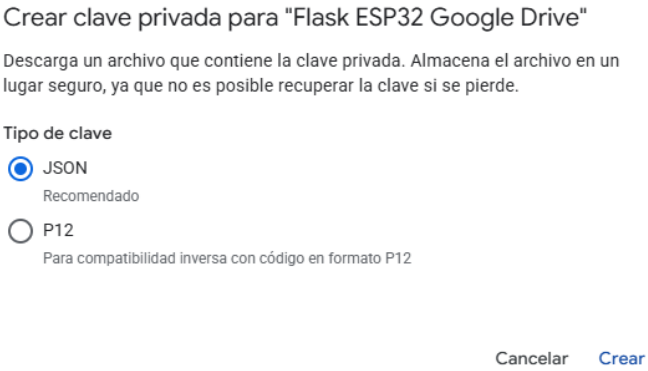


Figura 3.63: Clave creada del service_account.json

Fuente: Elaboración propia

- Este archivo de credenciales se subió a Render como un "Secret File" que es una función que permite que la aplicación acceda a él de forma segura sin exponerlo, se aprecia en la Figura 3.64.

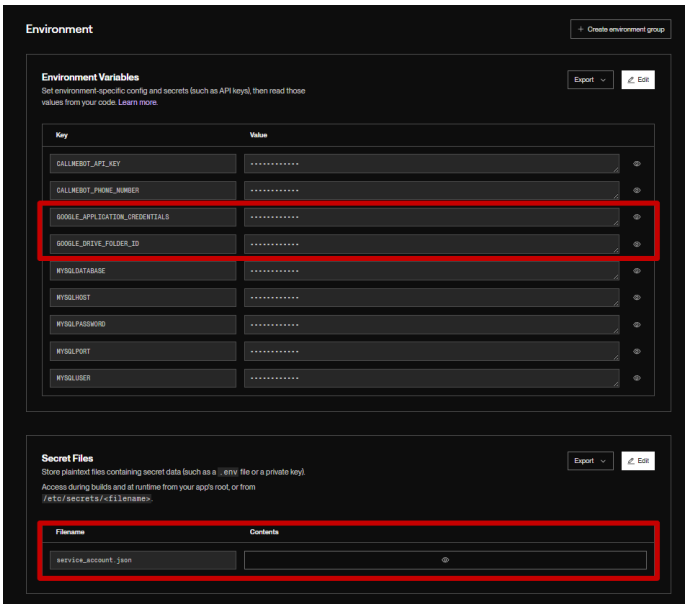


Figura 3.64: Credenciales de Google Drive en Render

Fuente: Elaboración propia

- El paso final fue compartir la carpeta de destino "Registro Sensor" del Google Drive con la dirección de correo electrónico de la cuenta de servicio, otorgándole los permisos de edición "Editor"necesarios como se observa en la siguiente Figura 3.65.

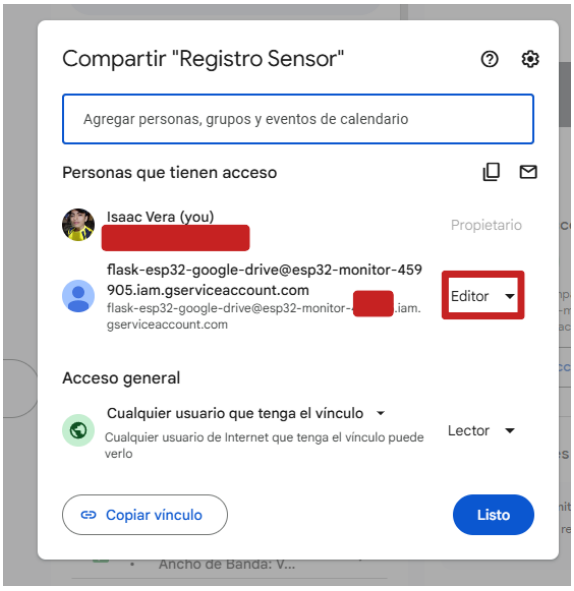


Figura 3.65: Cuenta de servicios de Google Drive

Fuente: Elaboración propia

- De esta forma el código en app.py puede usar la librería cliente de Google para leer estas credenciales, autenticarse y obtener los permisos para escribir el archivo CSV en la carpeta compartida como se muestra en la siguiente Figura 3.66.

Figura 3.66: Archivo CSL creado

Fuente: Elaboración propia

- **CallMeBot API:**

  - Para esta integración, el sitio web de CallMeBot "https://www.callmebot.com" brinda información de cómo obtener la clave (API key) agregando su número global "+34684734044", se puede registrar por cualquier nombre y se debe autorizar el envío de mensajes por parte del mismo "Autorizo callmebot a enviarme mensajes", en la siguiente Figura 3.67 se muestra el mensaje con la apikey para poder enviar mensajes utilizando la API.

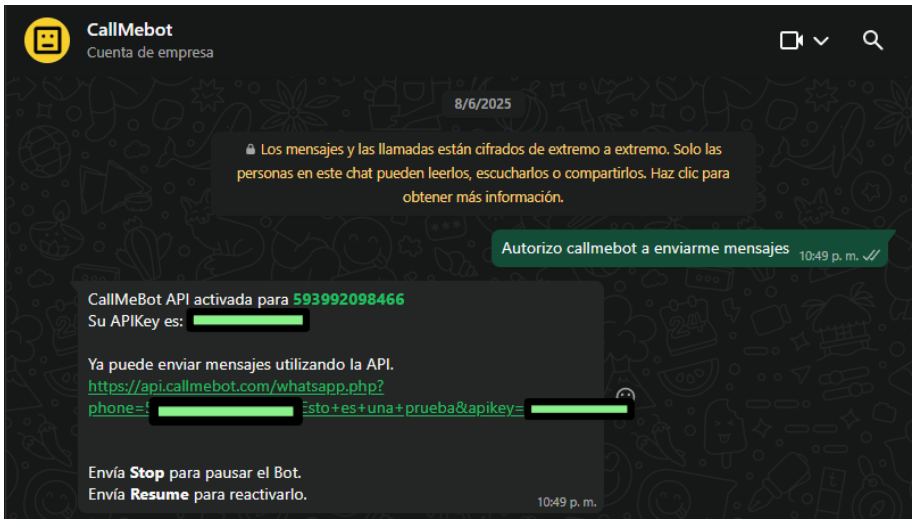


Figura 3.67: Api de Callmebot

Fuente: Elaboración propia

  - Al obtener la clave y con el número de teléfono de destino, se las configura como variables de entorno en Render, siguiendo las buenas prácticas de seguridad como se visualizará en la siguiente Figura 3.68.

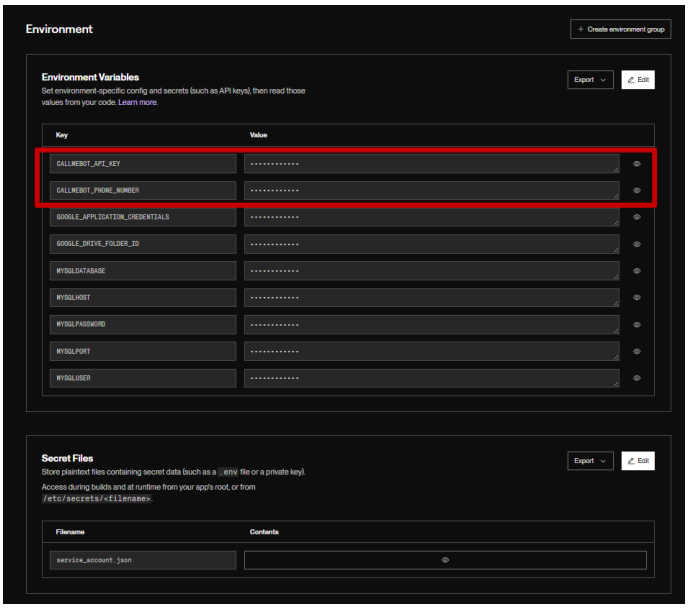


Figura 3.68: Variables API de Callmebot en Render

Fuente: Elaboración propia

- Así, la función "enviar_alerta _whatsapp" en nuestro código simplemente lee estas variables y construye la URL de la petición GET a la API de CallMeBot para despachar el mensaje de alerta cuando es necesario.

Esta arquitectura modular, basada en servicios especializados en la nube, no solo garantiza que el sistema sea robusto y escalable, sino que también facilita enormemente el mantenimiento y las futuras actualizaciones, ya que cada componente puede ser gestionado de forma independiente.

3.3. Gastos del proyecto

Descripción	Cantidad	Valor unitario (USD)	Valor total (USD)
ESP32 WROOM-32 IdeaSpark	1	15.00	15.00
Sensor MAX30102	1	8.00	8.00
Pantalla LCD ST7789	1	0.00	0.00
Cable JST 2 pines	2	0.25	0.50
Batería Li-ion 450 mAh	2	3.50	7.00
Zumbador piezoeléctrico	1	1.00	1.00
Push Button 2 pines	2	0.25	0.50
Conexión WiFi (hogar/lab)	1	0.00	0.00
Cuenta Google Cloud + Drive	1	0.00	0.00
Plataforma Render	1	0.00	0.00
Plataforma Railway	1	0.00	0.00
Placa PCB	1	0.00	40.00
Tensiómetro Digital (Ref)	1	0.00	50.00
Diseño impreso en 3D	1	0.00	100.00
Laptop personal	1	0.00	0.00
		Total estimado:	**222**

Tabla 3.19: Recursos utilizados en el desarrollo del sistema

3.4. Recursos personales

Nombre y Rol	Responsabilidad asignada
Isaac Vera (Estudiante investigador)	Desarrollo, programación y pruebas.
Profesor tutor / asesor académico	Revisión técnica y metodológica.

Tabla 3.20: Recursos personales asignados al proyecto

Capítulo 4

CÁLCULOS Y ANÁLISIS DE RESULTADOS

4.1. Análisis del Algoritmo del Firmware (.ino)

4.1.1. Gestión de Energía y Modo Sleep

El firmware actual implementa un sistema inteligente de gestión de energía mediante **deep sleep**, activándose únicamente cuando el usuario presiona el botón del dispositivo. Esta característica fundamental permite que el dispositivo permanezca en estado de ultra bajo consumo cuando no está en uso, optimizando significativamente la duración de la batería.

```
void setup() {
  if (esp_sleep_get_wakeup_cause() != ESP_SLEEP_WAKEUP_EXT0) {
    delay(100);
    esp_sleep_enable_ext0_wakeup(GPIO_NUM_27, 0);
    esp_deep_sleep_start();
  }

  Serial.println("Monitor encendido correctamente...");

  // Configuración de pines
  pinMode(SENSOR_POWER_PIN, OUTPUT);
  digitalWrite(SENSOR_POWER_PIN, HIGH);
  pinMode(TFT_BL, OUTPUT);
  digitalWrite(TFT_BL, HIGH);
  pinMode(BUZZER_PIN, OUTPUT);
  digitalWrite(BUZZER_PIN, LOW);
  pinMode(BUTTON_PIN, INPUT_PULLUP);
}
```

Figura 4.69: Código de hibernación y despertar por botón en función setup()

Fuente: Elaboración propia

Este mecanismo permite que el dispositivo permanezca en estado de ultra bajo consumo cuando no está en uso, activándose instantáneamente al presionar el botón. El sistema utiliza la función `esp_sleep_enable_ext0_wakeup()` para configurar el

GPIO27 como fuente de despertar, reduciendo drásticamente el consumo energético del dispositivo cuando no hay un paciente monitoreándose.

4.1.2. Configuración WiFi Automática con WiFiManager

El sistema implementa un portal de configuración automático que facilita la conexión a diferentes redes WiFi sin necesidad de recompilación del código, proporcionando flexibilidad operacional en diferentes ubicaciones.

```
void setup() {
  WiFiManager wm;
  wm.setConfigPortalTimeout(300);
  if (!wm.autoConnect("ESP32-Monitor", "12345678")) {
    mostrarMensaje("WiFi timeout", ST77XX_RED);
    delay(3000);
    ESP.restart();
  }
  wifiConectado = true;
  mostrarMensaje("WiFi conectado", ST77XX_GREEN);
  Wire.begin(21, 22);
  if (particleSensor.begin(Wire, I2C_SPEED_STANDARD)) {
    particleSensor.setup();
    particleSensor.setPulseAmplitudeRed(0x20);
    particleSensor.setPulseAmplitudeIR(0x20);
    sensorOK = true;
  }
}
```

Figura 4.70: Configuración automática WiFi con portal de configuración

Fuente: Elaboración propia

El WiFiManager crea automáticamente un punto de acceso temporal "ESP32-Monitor" cuando no puede conectarse a una red conocida, permitiendo al usuario configurar las credenciales WiFi a través de una interfaz web simple. Esto hace que el dispositivo sea portable y fácil de configurar en diferentes ubicaciones sin modificar el código. La configuración del sensor MAX30102 establece la amplitud de los LEDs infrarrojo y rojo en 0x20 para optimizar la calidad de la señal PPG.

4.1.3. Gestión Inteligente de Botón con Funciones Múltiples

El firmware implementa un sistema de detección de pulsación corta y larga del botón para diferentes funciones, maximizando la utilidad de un solo componente de interfaz.

```
void loop() {
  // Gestión del botón con función doble
  if (digitalRead(BUTTON_PIN) == LOW) {
    if (!botonPresionado) {
      botonPresionado = true;
      tiempoPresionado = millis();
    } else if (millis() - tiempoPresionado >= TIEMPO_LARGO) {
      limpiarVariables();
      delay(500);
      esp_sleep_enable_ext0_wakeup(GPIO_NUM_27, 0);
      esp_deep_sleep_start();
    }
  } else {
    botonPresionado = false;
  }
}
```

Figura 4.71: Gestión de botón con detección de pulsación corta y larga

Fuente: Elaboración propia

Esta implementación permite que un solo botón controle tanto el encendido (pulsación para despertar del sleep) como el apagado del dispositivo (pulsación larga de 2 segundos). El sistema limpia todas las variables antes de entrar en deep sleep, asegurando un reinicio limpio en la próxima activación.

4.1.4. Detección y Lectura del Sensor MAX30102

El algoritmo implementa una función dedicada para la adquisición de datos del sensor óptico con detección automática de dedo, asegurando que solo se procesen datos válidos cuando hay contacto adecuado.

```
void leerSensor() {
  // Leer nuevas muestras disponibles
  while (particleSensor.available()) {
    lastIR = particleSensor.getIR();
    lastRED = particleSensor.getRed();
    particleSensor.nextSample();
  }

  // Leer valores actuales si están disponibles
  long currentIR = particleSensor.getIR();
  long currentRED = particleSensor.getRed();

  if (currentIR > 0) lastIR = currentIR;
  if (currentRED > 0) lastRED = currentRED;

  // Detectar presencia de dedo con umbrales calibrados
  dedoAnterior = dedoDetectado;
  dedoDetectado = (lastIR > 20000 && lastIR < 300000);

  // Log cuando cambia estado del dedo
  if (dedoDetectado != dedoAnterior) {
    if (dedoDetectado) {
      Serial.println("Dedo detectado");
    } else {
      Serial.println("Dedo removido");
    }
  }
}
```

Figura 4.72: Función de lectura del sensor con detección automática de dedo

Fuente: Elaboración propia

Esta función optimiza la lectura del sensor MAX30102 verificando continuamente la disponibilidad de nuevas muestras y actualizando los valores IR y RED. La detección de dedo utiliza umbrales calibrados específicamente: un mínimo de 20,000 para asegurar contacto adecuado y un máximo de 300,000 para evitar saturación del sensor. El registro de cambios de estado permite monitorear la calidad de la medición.

4.1.5. Verificación Dinámica del Modo Entrenamiento

El sistema consulta periódicamente al servidor para determinar si debe operar en modo normal o modo entrenamiento, permitiendo coordinación centralizada de múltiples dispositivos para la recolección de datos de entrenamiento del modelo de machine learning.

```
void verificarModoEntrenamiento() {

  if (!wifiConectado) return;
  WiFiClientSecure client;
  client.setInsecure();
  HTTPClient http;
  String url = String(SERVER_URL) + "/api/training/status";
  if (http.begin(client, url)) {

    http.setTimeout(5000);
    int code = http.GET();

    if (code == 200) {
      String response = http.getString();
      // Verificar si el entrenamiento está activo
      bool servidor_entrenamiento = (response.indexOf("\"active\":true") > -1);

      if (servidor_entrenamiento != modoEntrenamiento) {
        modoEntrenamiento = servidor_entrenamiento;
        Serial.println("Modo entrenamiento cambiado a: " +
                       String(modoEntrenamiento ? "ACTIVO" : "INACTIVO"));

        if (modoEntrenamiento) {
          limpiarML();
          contadorEntrenamiento = 0;
          faseEntrenamiento = "stabilizing";
        } else {
          contadorEntrenamiento = 0;
          faseEntrenamiento = "idle";
        }
      }
      // Obtener contador de muestras si está en entrenamiento
      if (modoEntrenamiento && response.indexOf("\"sample_count\":") > -1) {
        int idx = response.indexOf("\"sample_count\":") + 15;
        String countStr = response.substring(idx);
        int fin = countStr.indexOf(",");
        if (fin == -1) fin = countStr.indexOf("}");
        if (fin > -1) {
          countStr = countStr.substring(0, fin);
          contadorEntrenamiento = countStr.toInt();
        }
      }
    }
    http.end();
  }
}
```

Figura 4.73: Verificación dinámica del modo entrenamiento con el servidor

Fuente: Elaboración propia

Este mecanismo permite que el dispositivo cambie automáticamente entre modo de medición normal y modo de recolección de datos para entrenamiento sin necesidad de reprogramación. El sistema consulta al servidor cada 10 segundos para sincronizar el estado operacional, permitiendo que múltiples dispositivos sean coordinados centralmente desde la interfaz web para la recolección de datos de entrenamiento del modelo de machine learning.

4.1.6. Transmisión de Datos con Verificación de Calidad

El algoritmo envía datos al servidor solo cuando cumple criterios de calidad y conectividad, construyendo dinámicamente el payload JSON con información del sensor y diferenciando entre modos de operación.

```
void enviarDatos() {
  if (!wifiConectado || !dedoDetectado) return;

  WiFiClientSecure client;
  client.setInsecure();
  HTTPClient http;

  String url = String(SERVER_URL) + "/api/data";

  if (http.begin(client, url)) {
    http.addHeader("Content-Type", "application/json");
    http.setTimeout(10000);

    // Construcción del JSON con datos del sensor
    String jsonPayload = "{";
    jsonPayload += "\"patient_id\":" + String(ID_PACIENTE) + ",";
    jsonPayload += "\"ir\":" + String(lastIR) + ",";
    jsonPayload += "\"red\":" + String(lastRED) + ",";
    jsonPayload += "\"timestamp\":\"" + String(millis()) + "\"";

    if (modoEntrenamiento) {
      jsonPayload += ",\"modo_entrenamiento\":true";
    }

    jsonPayload += "}";

    int httpCode = http.POST(jsonPayload);

    if (httpCode == 200) {
      String payload = http.getString();

      if (modoEntrenamiento) {
        procesarRespuestaEntrenamiento(payload);
      } else {
        procesarRespuestaML(payload);
      }
    }
    http.end();
  }
}
```

Figura 4.74: Función de transmisión de datos con verificación

Fuente: Elaboración propia

Esta función implementa la comunicación segura con el servidor, enviando datos únicamente cuando hay dedo detectado y conectividad WiFi estable. El sistema construye dinámicamente el payload JSON incluyendo el ID del paciente, valores IR/RED del sensor y timestamp. La función diferencia entre respuestas de entrenamiento y predicción ML, procesando cada tipo de respuesta de manera específica para actualizar las variables locales correspondientes.

4.1.7. Procesamiento de Respuestas del Servidor

El firmware implementa funciones especializadas para procesar las respuestas del servidor según el modo operacional, extrayendo información específica de las respuestas JSON y activando alertas locales cuando es necesario.

```
void procesarRespuestaML(String payload) {
  // Extraer valores ML del servidor
  if (payload.indexOf("\"sys_ml\":") > -1) {
    sys_ml = extraerValor(payload, "\"sys_ml\":");
  }
  if (payload.indexOf("\"dia_ml\":") > -1) {
    dia_ml = extraerValor(payload, "\"dia_ml\":");
  }
  if (payload.indexOf("\"hr_ml\":") > -1) {
    hr_ml = extraerValor(payload, "\"hr_ml\":");
  }
  if (payload.indexOf("\"spo2_ml\":") > -1) {
    spo2_ml = extraerValor(payload, "\"spo2_ml\":");
  }

  // Extraer nivel de riesgo y activar alarma si es crítico
  if (payload.indexOf("\"nivel\":") > -1) {
    int idx = payload.indexOf("\"nivel\":\"") + 9;
    if (idx > 8) {
      String str = payload.substring(idx);
      nivel_ml = str.substring(0, str.indexOf("\""));

      // Activar buzzer si hay crisis
      if (nivel_ml == "HT Crisis") {
        digitalWrite(BUZZER_PIN, HIGH);
      } else {
        digitalWrite(BUZZER_PIN, LOW);
      }
    }
  }
}

void procesarRespuestaEntrenamiento(String payload) {
  // Procesar respuesta de entrenamiento
  if (payload.indexOf("\"training_samples\":") > -1) {
    contadorEntrenamiento = (int)extraerValor(payload, "\"training_samples\":");
  }

  if (payload.indexOf("\"status\":") > -1) {
    int idx = payload.indexOf("\"status\":\"") + 10;
    if (idx > 9) {
      String str = payload.substring(idx);
      faseEntrenamiento = str.substring(0, str.indexOf("\""));
    }
  }
}
```

Figura 4.75: Procesamiento especializado de respuestas ML y entrenamiento

Fuente: Elaboración propia

Estas funciones especializadas extraen información específica de las respuestas JSON del servidor. En modo ML, procesan los valores de presión arterial estimados (sys_ml, dia_ml), frecuencia cardíaca (hr_ml), saturación de oxígeno (spo2_ml) y el nivel de riesgo, activando automáticamente la alarma local (buzzer) si se detecta una crisis hipertensiva. En modo entrenamiento, extraen el conteo de muestras y el estado de la fase de recolección para mostrar el progreso al usuario.

4.1.8. Gestión Inteligente de Variables y Limpieza de Memoria

El sistema implementa funciones específicas para la gestión de diferentes conjuntos de variables según el contexto operacional, proporcionando control granular sobre qué datos se mantienen en memoria.

```
void limpiarVariables() {
  // Limpiar todas las variables
  sys_ml = 0; dia_ml = 0; hr_ml = 0; spo2_ml = 0;
  nivel_ml = "---";
  muestras_ml_servidor = 0;
  contadorEntrenamiento = 0;
  faseEntrenamiento = "idle";
  modoEntrenamiento = false;
  digitalWrite(BUZZER_PIN, LOW);
  resultadoListo = false;
  Serial.println("Todas las variables limpiadas");
}

void limpiarML() {
  // Limpiar variables ML
  sys_ml = 0; dia_ml = 0; hr_ml = 0; spo2_ml = 0;
  nivel_ml = "---";
  muestras_ml_servidor = 0;
  digitalWrite(BUZZER_PIN, LOW);
  resultadoListo = false;
  Serial.println("Variables ML limpiadas");
}

void limpiarEntrenamiento() {
  // Limpiar variables de entrenamiento
  contadorEntrenamiento = 0;
  faseEntrenamiento = "idle";
  modoEntrenamiento = false;
  Serial.println("Variables entrenamiento limpiadas");
}
```

Figura 4.76: Funciones de limpieza segmentada de variables según contexto

Fuente: Elaboración propia

Esta segmentación de funciones de limpieza permite un control granular sobre qué datos se mantienen en memoria según el contexto. `limpiarVariables()` se usa antes del deep sleep para reinicio completo, `limpiarML()` se ejecuta cuando se retira el dedo para limpiar solo resultados de predicción, y `limpiarEntrenamiento()` se utiliza para resetear solo las variables de recolección de datos, manteniendo la eficiencia de memoria y estados consistentes.

4.2. Análisis del Algoritmo del Servidor (app.py)

El servidor del sistema implementa una arquitectura modular avanzada que organiza las funcionalidades en módulos especializados independientes, cada uno con responsabilidades específicas que trabajan de manera coordinada para proporcionar un sistema robusto de monitoreo médico.

4.2.1. Arquitectura Modular y Módulos Especializados

El sistema está compuesto por ocho módulos especializados que trabajan de forma coordinada bajo la orquestación del archivo principal `app.py`:

- **__init__.py:** Módulo de inicialización del paquete que configura el sistema de logging unificado para todos los módulos especializados y gestiona las importaciones de las clases principales del sistema.
- **ml_processor.py:** Procesador especializado de Machine Learning que maneja la carga de modelos pre-entrenados, realiza predicciones de presión arterial con calibración automática, gestiona cache de predicciones y proporciona métricas de rendimiento del sistema de inferencia.
- **database_manager.py:** Gestor de base de datos que maneja todas las operaciones de persistencia, incluyendo conexiones a MySQL en Railway, creación automática de tablas, operaciones asíncronas de escritura y consultas optimizadas para el histórico de mediciones.
- **alert_system.py:** Sistema de alertas médicas multi-canal que implementa notificaciones WhatsApp vía CallMeBot, control de rate limiting para evitar spam, clasificación de niveles de alerta según guías médicas y gestión de cola de alertas para procesamiento asíncrono.
- **api_nodo_datos.py:** API especializada para comunicación con dispositivos ESP32 que gestiona buffers separados para datos de entrenamiento y predicción ML, coordina la recolección de muestras y procesa las respuestas según el modo operacional del dispositivo.
- **data_collector.py:** Recolector de datos que coordina la captura de información de múltiples fuentes, gestiona la sincronización de datos con Google Drive para entrenamiento y proporciona estadísticas de recolección del sistema.
- **websocket_handler.py:** Manejador de comunicación bidireccional WebSocket que implementa actualizaciones en tiempo real a las interfaces web, gestiona conexiones de clientes, maneja cola de mensajes para optimización de rendimiento y proporciona eventos especializados para diferentes tipos de datos.

```python
# Importar módulos especializados
from modules.ml_processor import MLProcessor
from modules.database_manager import DatabaseManager
from modules.alert_system import AlertSystem
from modules.data_collector import DataCollector
from modules.websocket_handler import WebSocketHandler
from modules.api_nodo_datos import api_nodo_bp

class MedicalMonitorApp:
    """Aplicación principal del sistema de monitoreo médico"""

    def __init__(self):
        self.app = Flask(__name__)
        self.app.config['SECRET_KEY'] = os.environ.get('SECRET_KEY', 'dev_secret_key_change_in_production')

        # Inicializar SocketIO con configuración anti-errores
        self.socketio = SocketIO(
            self.app,
            cors_allowed_origins="*",
            async_mode='eventlet',
            logger=False,
            engineio_logger=False,
            ping_timeout=60,
            ping_interval=25
        )

        # Inicializar módulos especializados
        self.ml_processor = MLProcessor()
        self.db_manager = DatabaseManager()
        self.alert_system = AlertSystem()
        self.data_collector = DataCollector()
        self.websocket_handler = WebSocketHandler(self.socketio)
```

Figura 4.77: Inicialización de módulos especializados en la clase MedicalMonitorApp

Fuente: Elaboración propia

Esta arquitectura modular encapsula cada funcionalidad crítica del sistema en módulos independientes, facilitando el mantenimiento, testing y escalabilidad. El uso de SocketIO con configuración anti-errores asegura comunicación bidireccional estable con los clientes web, mientras que la separación de responsabilidades permite que cada módulo evolucione independientemente.

4.2.2. Configuración de Conexiones Inter-módulos

El sistema implementa un mecanismo centralizado para conectar todos los módulos especializados y registrar las APIs, estableciendo las dependencias necesarias entre componentes.

```python
def _setup_module_connections(self):
    """Configurar conexiones entre módulos"""
    # Conectar API con otros módulos
    api_nodo_bp.ml_processor = self.ml_processor
    api_nodo_bp.db_manager = self.db_manager
    api_nodo_bp.alert_system = self.alert_system
    api_nodo_bp.websocket_handler = self.websocket_handler
    api_nodo_bp.data_collector = self.data_collector

    # Registrar blueprint de API
    self.app.register_blueprint(api_nodo_bp, url_prefix='/api')

    # Configurar WebSocket handler
    self.websocket_handler.register_event_handlers()

    logger.info("Conexiones entre módulos configuradas")
```

Figura 4.78: Configuración de conexiones entre módulos especializados

Fuente: Elaboración propia

Esta función establece las conexiones entre todos los módulos especializados, permitiendo que el blueprint de la API (`api_nodo_bp`) acceda a las funcionalidades de ML, base de datos, alertas y WebSockets. El uso de blueprints de Flask organiza las rutas de manera modular, facilitando la escalabilidad y el mantenimiento del código.

4.2.3. Sistema de Entrenamiento Separado y Gestión de Estados

El servidor implementa un sistema completamente separado para la gestión del entrenamiento de modelos ML, permitiendo la recolección controlada de datos sin interferir con las operaciones normales de predicción.

```python
@self.app.route("/api/training/start", methods=["POST"])
def start_training_session():
    """Iniciar sesión de entrenamiento usando API module"""
    try:
        data = request.get_json() or {}
        patient_id = data.get('patient_id', 1)

        # Usar directamente el training buffer del API module
        from modules.api_nodo_datos import training_buffer
        training_buffer.start_collection(patient_id)

        return jsonify({
            "success": True,
            "message": "Entrenamiento iniciado",
            "active": True,
            "sample_count": 0
        })
    except Exception as e:
        logger.error(f"Error iniciando entrenamiento: {e}")
        return jsonify({"error": str(e)}), 500

@self.app.route("/api/training/stop", methods=["POST"])
def stop_training_session():
    """Detener sesión de entrenamiento"""
    try:
        from modules.api_nodo_datos import training_buffer
        buffer_data = training_buffer.stop_collection()

        return jsonify({
            "success": True,
            "message": "Entrenamiento detenido",
            "samples_collected": len(buffer_data),
            "active": False
        })
    except Exception as e:
        logger.error(f"Error deteniendo entrenamiento: {e}")
        return jsonify({"error": str(e)}), 500
```

Figura 4.79: Endpoints especializados para gestión de entrenamiento ML

Fuente: Elaboración propia

Este sistema permite la recolección controlada de datos para entrenamiento sin interferir con las operaciones normales de predicción. El `training_buffer` maneja de forma independiente la captura de datos, permitiendo que múltiples dispositivos contribuyan al dataset de entrenamiento de manera coordinada desde la interfaz web.

4.2.4. Clasificación Médica de Presión Arterial según Guías AHA/ESC

El servidor implementa un algoritmo de clasificación de riesgo basado en las guías médicas internacionales para hipertensión, categorizando automáticamente cada medición en niveles específicos de riesgo.

```python
def _classify_pressure_level(self, sys_pressure, dia_pressure):
    """Clasificar nivel de presión arterial según guías médicas AHA/ESC"""
    if sys_pressure is None or dia_pressure is None:
        return "N/A"

    try:
        sys_val, dia_val = float(sys_pressure), float(dia_pressure)
    except (ValueError, TypeError):
        return "Error"

    # Crisis de Hipertensión (SYS > 180 O DIA > 120)
    if sys_val > 180 or dia_val > 120:
        return "HT Crisis"

    # Hipertensión Etapa 2 (SYS >= 140 O DIA >= 90)
    if sys_val >= 140 or dia_val >= 90:
        return "HT2"

    # Hipertensión Etapa 1 (SYS 130-139 O DIA 80-89)
    if (130 <= sys_val <= 139) or (80 <= dia_val <= 89):
        return "HT1"

    # Elevada (SYS 120-129 Y DIA < 80)
    if 120 <= sys_val <= 129 and dia_val < 80:
        return "Elevada"

    # Normal (SYS < 120 Y DIA < 80)
    if sys_val < 120 and dia_val < 80:
        return "Normal"

    return "Revisar"
```

Figura 4.80: Algoritmo de clasificación según guías médicas AHA/ESC

Fuente: Elaboración propia

Este algoritmo implementa la clasificación estándar de presión arterial según las guías de la American Heart Association (AHA) y European Society of Cardiology (ESC), categorizando automáticamente cada medición en niveles específicos de riesgo. La función utiliza lógica condicional jerárquica que prioriza las condiciones más críticas, asegurando que una crisis hipertensiva sea detectada inmediatamente independientemente de otros valores.

4.2.5. API de Pruebas y Simulación de Alertas

El sistema incluye endpoints especializados para testing y validación del sistema de alertas, permitiendo la verificación completa de la cadena de procesamiento mediante casos simulados.

```python
@self.app.route("/api/test_alert", methods=['POST'])
def test_alert():
    """Endpoint de prueba para alertas"""
    try:
        data = request.get_json()
        if not data or "sys" not in data or "dia" not in data:
            return jsonify({"error": "Datos incompletos"}), 400

        # Procesar alerta de prueba
        test_data = {
            "patient_id": data.get("id_paciente", 99),
            "sys": float(data["sys"]),
            "dia": float(data["dia"]),
            "hr": data.get("hr", 0),
            "spo2": data.get("spo2", 0),
            "nivel": self._classify_pressure_level(float(data["sys"]), float(data["dia"]))
        }

        # Guardar en BD
        self.db_manager.save_measurement_async(test_data)

        # Enviar alerta
        self.alert_system.check_and_send_alert(test_data)

        # Notificar vía WebSocket
        self.websocket_handler.emit_new_record_saved()

        return jsonify({
            "status": "success",
            "message": "Alerta de prueba procesada",
            "data": test_data
        })

    except Exception as e:
        logger.error(f"Error en test de alerta: {e}")
        return jsonify({"error": str(e)}), 500
```

Figura 4.81: Endpoint de pruebas para validación del sistema de alertas

Fuente: Elaboración propia

Este endpoint permite la validación completa del sistema de alertas mediante simulación de casos críticos. El sistema procesa datos sintéticos como si fueran mediciones reales, ejecutando toda la cadena de procesamiento: clasificación de riesgo, almacenamiento en base de datos, envío de alertas WhatsApp y notificación en tiempo real a las interfaces web conectadas.

4.2.6. Gestión de Estado del Sistema y Monitoreo de Salud

El servidor implementa endpoints comprehensivos para monitoreo del estado de todos los módulos del sistema, proporcionando visibilidad completa del estado operacional.

```python
@self.app.route("/api/system_status")
def get_system_status():
    """Obtener estado completo del sistema"""
    try:
        status = {
            "timestamp": datetime.now().isoformat(),
            "uptime_hours": (time.time() - self.system_start_time) / 3600,
            "modules": {
                "ml_processor": self.ml_processor.get_status(),
                "database": self.db_manager.get_system_health(),
                "alerts": self.alert_system.get_status(),
                "websocket": self.websocket_handler.get_status(),
                "data_collector": self.data_collector.get_status()
            }
        }
        return jsonify(status)
    except Exception as e:
        logger.error(f"Error obteniendo estado del sistema: {e}")
        return jsonify({"error": str(e)}), 500

def run(self, host='0.0.0.0', port=None, debug=False):
    """Ejecutar la aplicación"""
    if port is None:
        port = int(os.environ.get("PORT", 10000))

    logger.info(f"Iniciando servidor en {host}:{port}")
    logger.info(f"Estado módulos - ML: {self.ml_processor.is_ready()}, "
                f"BD: {self.db_manager.is_connected()}, "
                f"Alertas: {self.alert_system.is_configured()}")

    # Crear tablas de BD si no existen
    if self.db_manager.is_connected():
        self.db_manager.create_tables_if_not_exist()
```

Figura 4.82: Sistema de monitoreo de estado de módulos especializados

Fuente: Elaboración propia

Estos métodos proporcionan visibilidad completa del estado operacional del sistema, incluyendo tiempo de actividad, estado de conexión de cada módulo y métricas de rendimiento. La función `run()` inicializa el servidor verificando previamente que todos los módulos estén funcionando correctamente, creando las tablas de base de datos si es necesario.

4.2.7. Gestión de Errores y Configuración Anti-errores

El sistema implementa una robusta gestión de errores y configuración especializada para evitar problemas comunes en entornos de producción con WebSockets y comunicación asíncrona.

```python
# Evitar warnings y errores de eventlet/socketio
warnings.filterwarnings("ignore")
logging.getLogger('socketio').setLevel(logging.WARNING)
logging.getLogger('engineio').setLevel(logging.WARNING)
logging.getLogger('eventlet').setLevel(logging.WARNING)

class QuietStderr:
    def write(self, s):
        if "Bad file descriptor" not in s and "socket shutdown error" not in s:
            sys.__stderr__.write(s)

    def flush(self):
        sys.__stderr__.flush()

if __name__ == "__main__":
    warnings.filterwarnings("ignore")
    sys.stderr = QuietStderr()

    try:
        medical_app.run(debug=False)
    except KeyboardInterrupt:
        medical_app.shutdown()
    except Exception as e:
        logger.error(f"Error crítico: {e}")
        medical_app.shutdown()

def shutdown(self):
    """Apagar sistema de forma segura"""
    try:
        logger.info("Inicio de apagar el sistema...")

        # Apagar módulos en orden
        self.websocket_handler.shutdown()
        self.alert_system.shutdown()
        self.db_manager.close_connections()

        logger.info("Apagado correctamente")
    except Exception:
        pass
```

Figura 4.83: Configuración anti-errores y gestión de excepciones del sistema

Fuente: Elaboración propia

Esta configuración especializada suprime warnings innecesarios, maneja apropiadamente las desconexiones de WebSocket y proporciona un mecanismo de apagado seguro del sistema. La clase `QuietStderr` filtra errores comunes de eventlet que no afectan la funcionalidad, manteniendo los logs limpios y enfocados en eventos relevantes para el monitoreo del sistema.

4.3. Análisis del Panel de Control (index.html)

El panel de control web constituye la interfaz principal del sistema, implementando una aplicación de tiempo real que permite el monitoreo continuo de datos biométricos y la gestión del proceso de entrenamiento de modelos de machine learning. La interfaz está desarrollada utilizando HTML5, CSS3 y JavaScript vanilla con comunicación WebSocket bidireccional.

4.3.1. Inicialización de la Interfaz y Configuración de Elementos DOM

El panel de control implementa una estructura modular para la gestión de elementos de interfaz, organizando todos los componentes DOM de manera sistemática para facilitar su manipulación y mantenimiento.

```
document.addEventListener('DOMContentLoaded', function() {
    const ui = {
        // Lecturas del sensor
        ir: document.getElementById('ir'),
        red: document.getElementById('red'),
        fingerStatus: document.getElementById('fingerStatus'),

        // Prediccion ML
        sys: document.getElementById('sys'),
        dia: document.getElementById('dia'),
        hr: document.getElementById('hr'),
        spo2: document.getElementById('spo2'),
        nivel: document.getElementById('nivel'),

        // Entrenamiento
        authTrainingBtn: document.getElementById('authTrainingBtn'),
        trainingSubsection: document.getElementById('training-subsection'),
        toggleCaptureBtn: document.getElementById('toggleCaptureBtn'),
        saveSampleBtn: document.getElementById('saveSampleBtn'),
        captureStatus: document.getElementById('capture-status'),
        trainingSamplesCount: document.getElementById('training_samples_count'),

        // Tabla
        medicionesTbody: document.getElementById('medicionesTable').querySelector('tbody')
    };

    let trainingAuthorized = false;
    let isCapturing = false;
    let socket;
```

Figura 4.84: Inicialización de elementos DOM y variables de estado de la interfaz

Fuente: Elaboración propia

Esta estructura de interfaz organiza todos los elementos DOM de manera modular, estableciendo variables de estado para el control de la interfaz de entrenamiento y preparando las bases para la comunicación Socket.IO. La organización modular facilita el mantenimiento y la escalabilidad de la interfaz, permitiendo un control granular sobre cada componente visual del sistema.

4.3.2. Gestión de Comunicación Socket.IO con Eventos Especializados

El sistema implementa manejo robusto de eventos WebSocket para comunicación en tiempo real, estableciendo conexiones bidireccionales estables con el servidor para actualizaciones automáticas de la interfaz.

```javascript
function initializeSocket() {
    try {
        socket = io();

        socket.on('connect', () => {
            console.log('Conectado al servidor Socket.IO');
            updateConnectionStatus(true);
        });

        socket.on('disconnect', () => {
            console.log('Desconectado del servidor Socket.IO');
            updateConnectionStatus(false);
        });

        // Evento principal para actualizacion de datos
        socket.on('update_data', function(data) {
            console.log('Datos recibidos:', data);

            // Actualizar valores del sensor
            if (data.ir !== undefined) {
                ui.ir.textContent = data.ir;
                updateFingerStatus(data.ir);
            }
            if (data.red !== undefined) ui.red.textContent = data.red;

            // Actualizar predicciones ML
            if (data.sys_ml !== undefined) ui.sys.textContent = data.sys_ml;
            if (data.dia_ml !== undefined) ui.dia.textContent = data.dia_ml;
            if (data.hr_ml !== undefined) ui.hr.textContent = data.hr_ml;
            if (data.spo2_ml !== undefined) ui.spo2.textContent = data.spo2_ml;
            if (data.nivel !== undefined) {
                ui.nivel.textContent = data.nivel;
                updateNivelColor(data.nivel);
            }
        });

        // Evento para nuevos registros guardados
        socket.on('new_record_saved', function(data) {
            console.log('Nuevo registro guardado:', data);
            actualizarTablaMediciones();
            showMessage('Nuevo registro guardado en la base de datos', 'success');
        });

    } catch (error) {
        console.error('Error inicializando Socket.IO:', error);
        showMessage('Error de conexión WebSocket', 'error');
    }
}
```

Figura 4.85: Configuración de Socket.IO y manejo de eventos de comunicación

Fuente: Elaboración propia

Este sistema maneja la comunicación bidireccional entre el panel web y el servidor, actualizando automáticamente la interfaz cuando llegan nuevos datos del dispositivo ESP32. Los eventos especializados (`update_data`, `new_record_saved`) permiten actualizaciones granulares de diferentes secciones de la interfaz sin recargar la página completa, proporcionando una experiencia de usuario fluida y responsiva.

4.3.3. Sistema de Gestión de Entrenamiento con Estados Dinámicos

La interfaz implementa un sistema completo para controlar la recolección de datos de entrenamiento, coordinando con el servidor para activar y desactivar el modo de recolección de datos según las necesidades del usuario.

```
// Autorizar modo entrenamiento
ui.authTrainingBtn.addEventListener('click', function() {
    if (!trainingAuthorized) {
        fetch('/api/training/start', {
            method: 'POST',
            headers: {'Content-Type': 'application/json'},
            body: JSON.stringify({patient_id: 1})
        })
        .then(response => response.json())
        .then(data => {
            if (data.success) {
                trainingAuthorized = true;
                updateTrainingUI();
                showMessage('Modo entrenamiento ACTIVADO', 'success');
            } else {
                showMessage('Error activando entrenamiento: ' + (data.error || 'Error desconocido'), 'error');
            }
        })
        .catch(error => {
            console.error('Error:', error);
            showMessage('Error de conexión al activar entrenamiento', 'error');
        });
    } else {
        // Desactivar entrenamiento
        fetch('/api/training/stop', {
            method: 'POST',
            headers: {'Content-Type': 'application/json'}
        })
        .then(response => response.json())
        .then(data => {
            trainingAuthorized = false;
            isCapturing = false;
            updateTrainingUI();
            showMessage('Modo entrenamiento DESACTIVADO', 'info');
        });
    }
});
```

Figura 4.86: Control de autorización del modo entrenamiento - Parte 1

Fuente: Elaboración propia

```
// Controlar captura de datos
ui.toggleCaptureBtn.addEventListener('click', function() {
    if (!isCapturing) {
        // Iniciar captura
        fetch('/api/training/capture/start', {
            method: 'POST',
            headers: {'Content-Type': 'application/json'}
        })
        .then(response => response.json())
        .then(data => {
            if (data.success) {
                isCapturing = true;
                updateTrainingUI();
                ui.captureStatus.textContent = 'Capturando datos... Mantenga el dedo en el sensor';
                showMessage('Captura iniciada', 'success');
            }
        });
    } else {
        // Detener captura
        fetch('/api/training/capture/stop', {
            method: 'POST',
            headers: {'Content-Type': 'application/json'}
        })
        .then(response => response.json())
        .then(data => {
            isCapturing = false;
            updateTrainingUI();
            ui.captureStatus.textContent = 'Captura detenida. Ingrese valores de referencia';
            showMessage('Captura detenida. Complete los valores de referencia.', 'info');
        });
    }
});
```

Figura 4.87: Captura del modo entrenamiento - Parte 2

Fuente: Elaboración propia

Este sistema permite el control completo del proceso de entrenamiento desde la interfaz web, coordinando con el servidor para activar/desactivar el modo de recolección de datos. Los diferentes estados (autorizado/no autorizado, capturando/detenido) se reflejan dinámicamente en la interfaz, guiando al usuario a través del proceso de recolección de datos para entrenamiento del modelo ML de manera intuitiva y segura.

4.3.4. Validación y Envío de Datos de Referencia

La interfaz implementa validación robusta para los datos de referencia del entrenamiento, asegurando que todos los valores ingresados cumplan con rangos médicos válidos antes de ser enviados al servidor.

```
// Guardar muestra de entrenamiento
ui.saveSampleBtn.addEventListener('click', function() {
    const sysRef = parseFloat(ui.sysRef.value);
    const diaRef = parseFloat(ui.diaRef.value);
    const hrRef = parseFloat(ui.hrRef.value);

    // Validación de datos
    if (!sysRef || !diaRef || !hrRef) {
        showMessage('Por favor complete todos los campos de referencia', 'error');
        return;
    }

    if (sysRef < 70 || sysRef > 250) {
        showMessage('Presión sistólica debe estar entre 70-250 mmHg', 'error');
        return;
    }

    if (diaRef < 40 || diaRef > 150) {
        showMessage('Presión diastólica debe estar entre 40-150 mmHg', 'error');
        return;
    }

    if (hrRef < 40 || hrRef > 200) {
        showMessage('Frecuencia cardíaca debe estar entre 40-200 bpm', 'error');
        return;
    }

    if (sysRef <= diaRef) {
        showMessage('La presión sistólica debe ser mayor que la diastólica', 'error');
        return;
    }

    // Enviar datos al servidor
    const referenceData = {
        sys_ref: sysRef,
        dia_ref: diaRef,
        hr_ref: hrRef,
        timestamp: new Date().toISOString()
    };
```

Figura 4.88: Validación médica de datos de referencia - Parte 1

Fuente: Elaboración propia

```
    // Enviar datos al servidor
    const referenceData = {
        sys_ref: sysRef,
        dia_ref: diaRef,
        hr_ref: hrRef,
        timestamp: new Date().toISOString()
    };

    fetch('/api/training/save_sample', {
        method: 'POST',
        headers: {'Content-Type': 'application/json'},
        body: JSON.stringify(referenceData)
    })
    .then(response => response.json())
    .then(data => {
        if (data.success) {
            showMessage('Muestra guardada exitosamente en Google Drive', 'success');
            ui.trainingSamplesCount.textContent = data.total_samples + ' muestras';

            // Limpiar campos
            ui.sysRef.value = '';
            ui.diaRef.value = '';
            ui.hrRef.value = '';
            ui.captureStatus.textContent = '';
        } else {
            showMessage('Error guardando muestra: ' + (data.error || 'Error desconocido'), 'error');
        }
    })
    .catch(error => {
        console.error('Error:', error);
        showMessage('Error de conexión al guardar muestra', 'error');
    });
});
```

Figura 4.89: Envío de datos de referencia - Parte 2

Fuente: Elaboración propia

Este módulo implementa validación médica comprehensiva de los datos de referencia ingresados por el usuario, asegurando que los valores estén dentro de rangos fisiológicos normales antes de enviarlos al servidor. La validación incluye verificación de rangos clínicos para presión arterial (sistólica: 70-250 mmHg, diastólica: 40-150 mmHg) y frecuencia cardíaca (40-200 bpm), así como coherencia entre valores sistólicos y diastólicos.

4.3.5. Actualización Dinámica de la Interfaz y Gestión de Estados

El sistema implementa funciones especializadas para mantener la coherencia visual de la interfaz según los estados operacionales, proporcionando retroalimentación visual inmediata al usuario sobre el estado del sistema.

```
function updateTrainingUI() {
    // Mostrar/ocultar sección de entrenamiento
    ui.trainingSubsection.style.display = trainingAuthorized ? 'block' : 'none';

    // Texto del botón principal
    ui.authTrainingBtn.textContent = trainingAuthorized ?
        'Desactivar Modo Entrenamiento' :
        '1. Autorizar Modo Entrenamiento';

    // Estado del botón de captura
    ui.toggleCaptureBtn.disabled = !trainingAuthorized;

    if (trainingAuthorized) {
        ui.toggleCaptureBtn.textContent = isCapturing ?
            '2. Detener Captura' :
            '2. Iniciar Captura de Entrenamiento';

        if (isCapturing) {
            ui.toggleCaptureBtn.className = 'stop';
        } else {
            ui.toggleCaptureBtn.className = '';
        }
    }

    // Estado de los campos de referencia
    const canEnterReference = !isCapturing && trainingAuthorized &&
        (ui.captureStatus.textContent.includes('Detenida') ||
         ui.captureStatus.textContent.includes('Ingrese valores'));

    ui.sysRef.disabled = !canEnterReference;
    ui.diaRef.disabled = !canEnterReference;
    ui.hrRef.disabled = !canEnterReference;
    ui.saveSampleBtn.disabled = !canEnterReference;

    // Limpiar campos si se desactiva entrenamiento
    if (!trainingAuthorized) {
        ui.trainingSamplesCount.textContent = '0 muestras';
        ui.captureStatus.textContent = '';
        ui.sysRef.value = '';
        ui.diaRef.value = '';
        ui.hrRef.value = '';
        isCapturing = false;
    }
}
```

Figura 4.90: Actualización dinámica de interfaz visual - Parte 1

Fuente: Elaboración propia

```
function updateNivelColor(nivel) {
    // Aplicar colores según nivel de riesgo
    ui.nivel.className = '';
    switch(nivel) {
        case 'Normal':
            ui.nivel.style.color = '#28a745';
            break;
        case 'Elevada':
            ui.nivel.style.color = '#ffc107';
            break;
        case 'HT1':
            ui.nivel.style.color = '#fd7e14';
            break;
        case 'HT2':
            ui.nivel.style.color = '#dc3545';
            break;
        case 'HT Crisis':
            ui.nivel.style.color = '#dc3545';
            ui.nivel.style.fontWeight = 'bold';
            ui.nivel.style.animation = 'blink 1s infinite';
            break;
        default:
            ui.nivel.style.color = '#6c757d';
    }
}
```

Figura 4.91: Actualización dinámica de gestión visual - Parte 2

Fuente: Elaboración propia

Estas funciones mantienen la coherencia visual y funcional de la interfaz, actualizando dinámicamente la disponibilidad de controles según el estado operacional. El sistema de colores para los niveles de riesgo proporciona feedback visual inmediato al usuario, con animaciones especiales para crisis hipertensivas que requieren atención médica inmediata, mejorando significativamente la usabilidad del sistema.

4.3.6. Gestión de Tabla de Datos y Comunicación Asíncrona

La interfaz implementa actualización automática de la tabla de mediciones históricas, manteniendo sincronización con la base de datos del servidor mediante consultas periódicas y eventos WebSocket.

```
function actualizarTablaMediciones() {
    fetch('/api/mediciones_recientes?limit=20')
    .then(response => response.json())
    .then(data => {
        if (data.success && data.mediciones) {
            const tbody = ui.medicionesTbody;
            tbody.innerHTML = '';

            data.mediciones.forEach((medicion, index) => {
                const row = tbody.insertRow();
                row.innerHTML = `
                    <td>${index + 1}</td>
                    <td>PAC${String(medicion.patient_id).padStart(3, '0')}</td>
                    <td>${medicion.sys}</td>
                    <td>${medicion.dia}</td>
                    <td>${medicion.hr}</td>
                    <td>${medicion.spo2}</td>
                    <td style="color: ${getNivelColor(medicion.nivel)}">${medicion.nivel}</td>
                `;
            });
        } else {
            ui.medicionesTbody.innerHTML = '<tr><td colspan="7" style="text-align:center;">Error cargando datos</td></tr>';
        }
    })
    .catch(error => {
        console.error('Error actualizando tabla:', error);
        ui.medicionesTbody.innerHTML = '<tr><td colspan="7" style="text-align:center;">Error de conexión</td></tr>';
    });
}

// Inicializar aplicación
console.log('Inicializando aplicación...');
updateTrainingUI();
initializeSocket();
actualizarTablaMediciones();

// Verificar estado de entrenamiento cada 10 segundos
setInterval(checkTrainingStatus, 10000);

// Auto-actualizar tabla cada 30 segundos
setInterval(actualizarTablaMediciones, 30000);
```

Figura 4.92: Gestión automática de tabla de mediciones y consultas asíncronas

Fuente: Elaboración propia

Este sistema asegura que la tabla de mediciones históricas se mantenga actualizada automáticamente, consultando al servidor cada 30 segundos y también cuando se reciben notificaciones de nuevos registros vía WebSocket. La función maneja apropiadamente los errores de conectividad y proporciona retroalimentación visual clara al usuario sobre el estado de la carga de datos, asegurando una experiencia confiable.

4.4. Ecuaciones de Filtrado

4.4.1. Filtro de promedio móvil (Firmware)

Este filtro es aplicado a la Frecuencia Cardíaca (HR) en el código “ino”, denominado Promedio Móvil Simple (SMA), la ecuación para calcular el promedio de HR es:

$$\text{promedioHR} = \frac{1}{N} \sum_{i=1}^{N} HR_i \tag{4.1}$$

Donde:

- HR_i es cada una de las lecturas de frecuencia cardíaca válidas.
- N es el número de lecturas en el promedio (en el código “ino” es la variable llamada “lecturasValidas”, hasta un máximo de 5 lecturas.

4.4.2. Filtro estadístico para entrenamiento (Servidor)

Este procesamiento es aplicado en el servidor cuando se guardan los datos para el entrenamiento del modelo en la función "procesar_buffer_y_guardar".

Media Aritmética (np.mean)

Para una serie de n lecturas de una señal $señal = \{señal_1, señal_2, \ldots, señal_n\}$, por ejemplo, del sensor infrarrojo (IR), la media μ se calcula como:

$$\mu_{señal} = \frac{1}{n} \sum_{i=1}^{n} señal_i \tag{4.2}$$

Desviación Estándar (np.std)

La desviación estándar σ para esa misma serie se calcula como:

$$\sigma_{señal} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (señal_i - \mu_{señal})^2} \tag{4.3}$$

Estas dos ecuaciones se aplican tanto al búfer de la señal "IR" como al de la "RED" para así obtener los valores de las características "ir_mean_filtrado", "red_mean_filtrado", "ir_std_filtrado" y "red_std_filtrado".

4.5. Ecuación teórica para el cálculo de SpO_2

Es importante aclarar que el archivo "ino" utiliza la librería externa (#include "spo2_algorithm.h") para realizar el cálculo, sin embargo el principio universal en el que se basan estos algoritmos es la "ley de Beer-Lambert", en donde el Spo2 se determina a partir de la relación entre la luz roja (RED) y la infrarroja (IR) absorbida por la sangre, el cálculo se realiza en dos pasos:

Paso 1: Calcular la relación de ratios (R)

Primero se calcula la relación conocida como "Ratio of Ratios" o "R" entre las componentes pulsátil (AC) y no pulsátil (DC) de las señales roja e infrarroja:

$$R = \frac{(AC_{IR}/DC_{IR})}{(AC_{RED}/DC_{RED})} \tag{4.4}$$

Donde:

- AC_{RED}**:** Componente alterna (pulsátil) de la señal de luz roja, que representa la variación en la absorción debida al pulso arterial.
- DC_{RED}**:** Componente continua (no pulsátil) de la luz roja, que representa la absorción constante por tejidos, huesos y sangre venosa.
- AC_{IR}**:** Componente alterna de la señal infrarroja.
- DC_{IR}**:** Componente continua de la señal infrarroja.

Paso 2: Estimar la SpO_2 usando una curva de calibración

El valor de R se relaciona con la SpO_2 mediante una curva de calibración empírica, que depende del tipo de sensor, esta curva suele aproximarse mediante una ecuación lineal o polinómica, una forma lineal común es:

$$\mathrm{SpO_2}(\,\%) = A - B \cdot R \tag{4.5}$$

Donde:

- A y B son constantes de calibración. Por ejemplo, valores teóricos comunes (que varían según el sensor) podrían ser $A \approx 110$ y $B \approx 25$.

La librería "spo2_algorithm.h" contiene la implementación de este principio incluyendo algoritmos de filtrado digital para separar las componentes AC y DC y la ecuación de calibración específica con las constantes adecuadas para el sensor "MAX30102".

4.5.1. Factores de Calibración del Modelo ML

El sistema aplica factores de calibración automática para mejorar la precisión:

$$SYS_{final} = SYS_{ML} \times 0{,}87 \tag{4.6}$$

$$DIA_{final} = DIA_{ML} \times 0{,}89 \tag{4.7}$$

4.5.2. Umbrales de Detección del Sistema

4.5.2.1. Detección de Dedo

$$Dedo_{detectado} = \begin{cases} \text{true} & \text{si } 20000 < IR < 300000 \\ \text{false} & \text{caso contrario} \end{cases} \tag{4.8}$$

4.5.2.2. Calidad de Señal

$$Transmision = \begin{cases} \text{enviar} & \text{si } IR > 50000 \\ \text{esperar} & \text{si } IR \leq 50000 \end{cases} \tag{4.9}$$

4.5.3. Ecuación SpO2 Implementada

El sistema utiliza la ecuación calibrada para el MAX30102:

$$SpO_2 = 104 - 17 \times R \tag{4.10}$$

Con validación de rangos:

$$SpO_{2final} = \begin{cases} SpO_2 & \text{si } 85 \leq SpO_2 \leq 100 \\ 98{,}0 & \text{caso contrario} \end{cases} \tag{4.11}$$

4.6. Pruebas de Funcionamiento y Validación del Sistema

4.6.1. Validación del Nodo Sensor IoT (Hardware y Firmware)

El prototipo del nodo sensor fue ensamblado en una placa de circuito impreso (PCB) diseñada en KiCad mostrada en la Figura 3.37 y 3.38, la siguiente Figura 4.94 y la Figura 4.93 muestran el resultado de la placa con sus circuitos ensamblados.

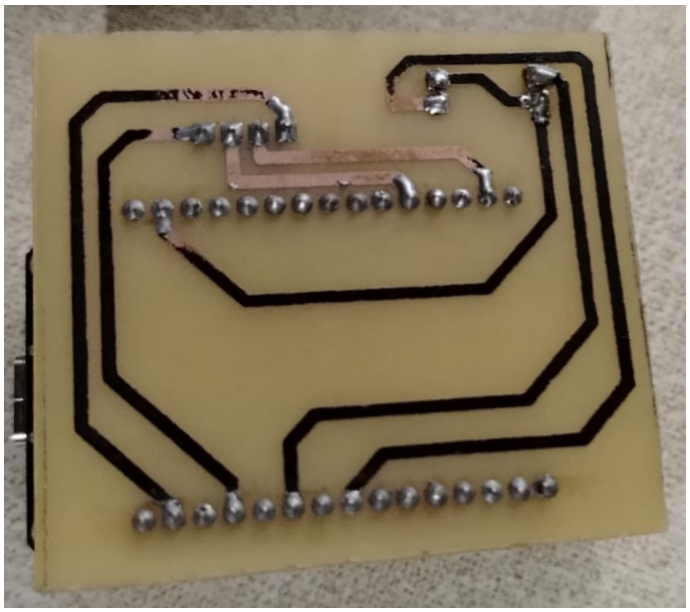


Figura 4.93: Resultado de la PCB

Fuente: Elaboración propia

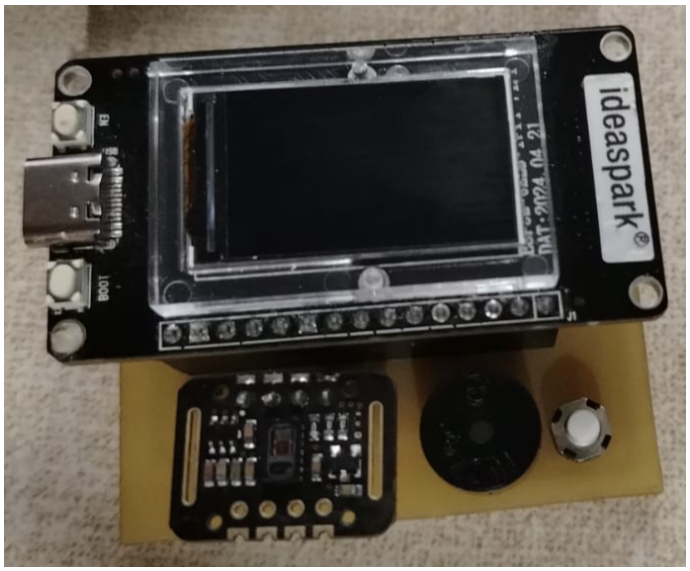


Figura 4.94: Placa de circuito impreso con sus componentes electrónicos soldados y conectados

Fuente: Elaboración propia

Luego se ha integrado en una carcasa ergonómica fabricada mediante impresión 3D diseñada en Fusion 360, los diseños se visualizan en **ANEXO D** con el propósito de aislar en sensor para mejores resultados y evitar ruidos que se generan por la iluminación del entorno, la siguiente Figura 4.95, se hace la instalación del circuito ensamblado en su caja elaborado y diseñado mediante impresión 3D y la Figura 4.96 muestra el diseño como resultado final con la placa de circuito impreso (PCB).

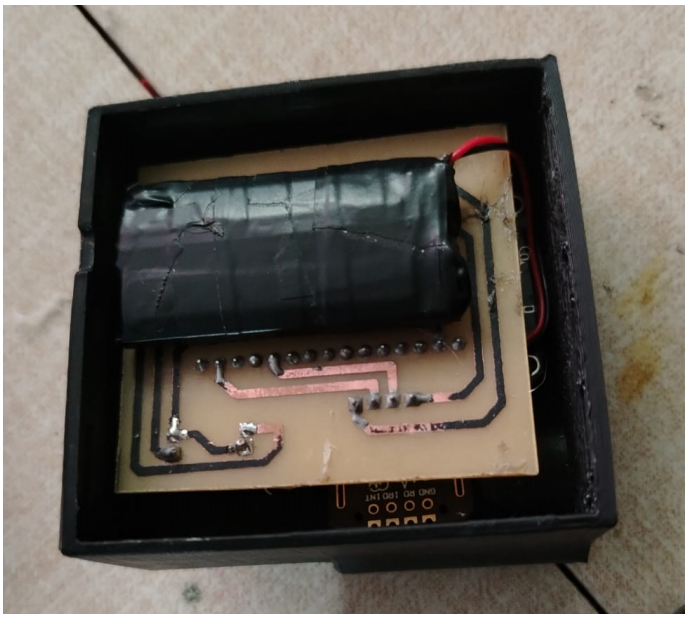


Figura 4.95: Instlación del circuito ensamblado a su gabinete fabricado mediante impresión 3D

Fuente: Elaboración propia

Figura 4.96: Aspecto final de mi dispositvo complatemente ensamblado

Fuente: Elaboración propia

4.6.1.1. Configuración Automática WiFi con WiFiManager

El dispositivo implementa la librería WiFiManager que facilita la configuración de red sin necesidad de recompilar el código. Este sistema permite que el dispositivo

sea portable y fácil de configurar en diferentes ubicaciones.

Figura 4.97: Portal de configuración WiFi generado automáticamente por WiFiManager

Fuente: Elaboración propia

Cuando el dispositivo no puede conectarse a una red conocida la librería WiFiManager crea automáticamente un punto de acceso temporal que permite al usuario

configurar las credenciales WiFi a través de una interfaz web simple desde cualquier dispositivo móvil o computadora.

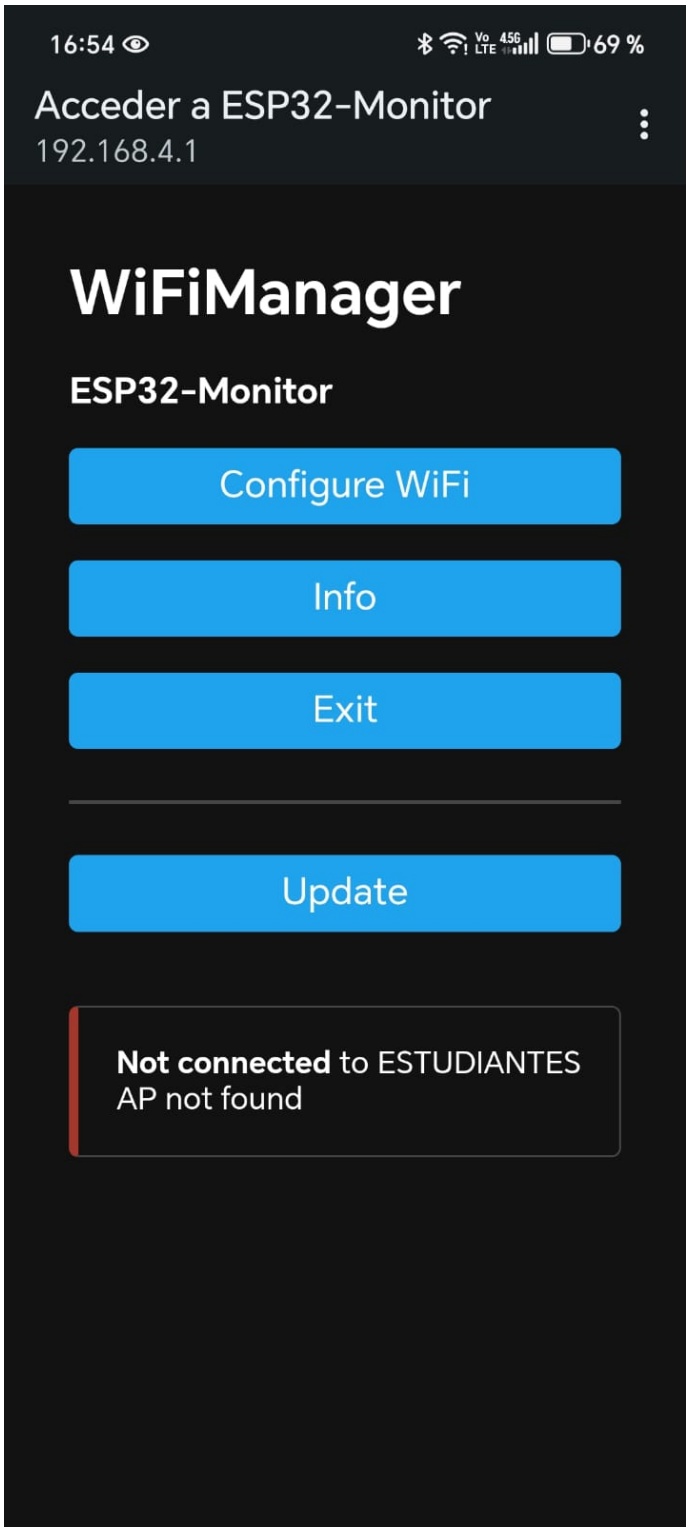


Figura 4.98: Menú principal del WiFiManager

Fuente: Elaboración propia

4.6.1.2. Confirmación de Conexión Exitosa

Una vez completada la configuración de red el sistema confirma la conexión e inicia el dispositivo para su funcionamiento.

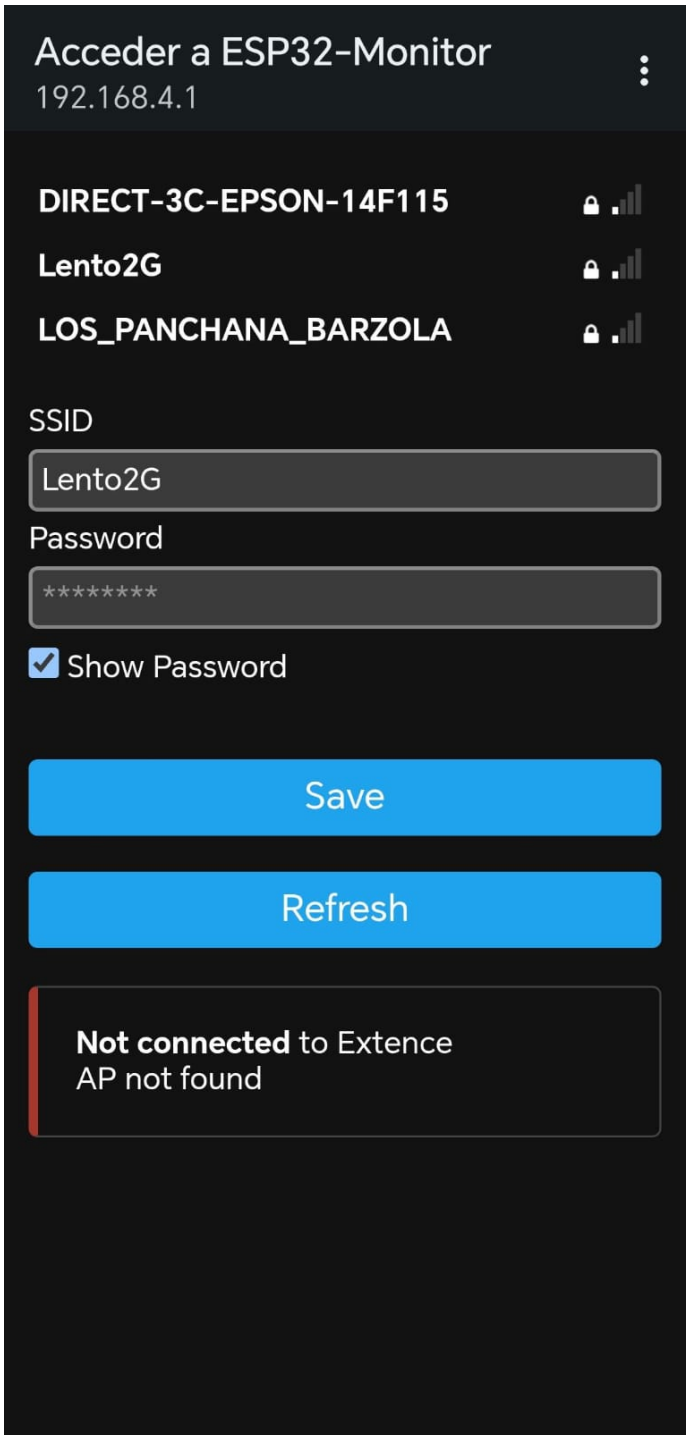


Figura 4.99: Dispositivo confirmando conexión WiFi exitosa

Fuente: Elaboración propia

4.6.1.3. Estado Sistema Listo

Después de verificar la conectividad y inicializar todos los componentes el dispositivo muestrará el mensaje "Sistema Listoïndicando que está preparado para comenzar las mediciones.

Figura 4.100: Pantalla mostrando "Sistema Listo"después de configuración WiFiManager

Fuente: Elaboración propia

4.6.1.4. Ventajas del Sistema WiFiManager Implementado

La implementación de WiFiManager proporciona múltiples beneficios operativos:

- **Portabilidad:** El dispositivo puede conectarse a cualquier red WiFi sin modificar el código
- **Facilidad de Uso:** La configuración se realiza a través de una interfaz web intuitiva
- **Autonomía:** No requiere conexión por cable para configurar credenciales de red
- **Flexibilidad:** Permite usar el dispositivo en diferentes ubicaciones (hogar, clínica, hospital)
- **Recuperación Automática:** Si se pierde la conexión puede reconfigurar automáticamente

Este sistema hace que el prototipo sea más práctico para uso clínico real donde las redes WiFi pueden variar según la ubicación de implementación.

4.6.2. Logs del servidor Flask en Render mostrando la inicialización del sistema y carga de modelos ML

La Figura 4.101 muestra los logs del servidor Flask en Render durante la fase de inicialización y carga de los modelos de Machine Learning, se puede observar la carga exitosa de los modelos Random Forest (modelo_sys.pkl y modelo_dia.pkl), la configuración de la base de datos, el establecimiento de conexiones con Google Drive para respaldo de datos de entrenamiento, y la inicialización del sistema de alertas, es decir, los logs confirma que todos los componentes del sistema están operativos y listos para procesar peticiones del dispositivo ESP32.

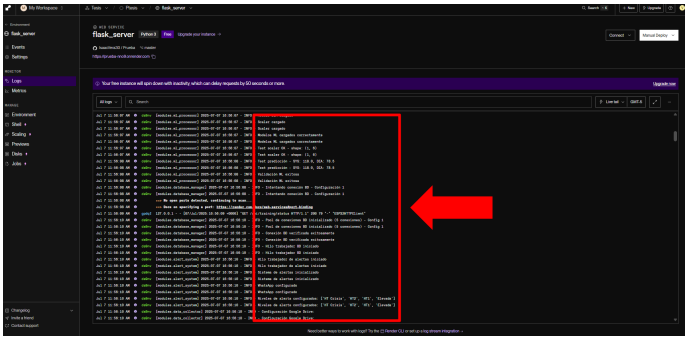


Figura 4.101: Logs del servidor Flask mostrando la carga exitosa de modelos ML y configuración del sistema

Fuente: Elaboración propia

La Firgura 4.102 evidencia el funcionamiento en tiempo real del sistema, mostrando múltiples peticiones HTTP POST del dispositivo ESP32 al endpoint /api/-training/status, se puede observar las conexiones WebSocket establecidas para la comunicación bidireccional, las peticiones GET y POST siendo procesadas correctamente con códigos de respuesta HTTP 200, y la URL del servicio completamente operativa, es decir, los logs demuestran que el servidor está recibiendo y procesando exitosamente los datos del sensor MAX30102 para ejecutar las predicciones con Random Forest.

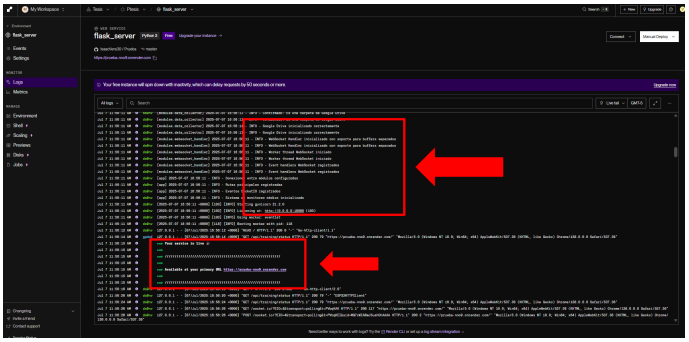


Figura 4.102: Logs del servidor mostrando peticiones HTTP activas del dispositivo ESP32 y procesamiento en tiempo real

Fuente: Elaboración propia

Las imágenes demostran tanto la inicialización del sistema 4.101 como el fun-

cionamiento operativo en tiempo real 4.102 en tu sección de validación del sistema ML.

4.7. Validación en Tiempo Real del Sistema Random Forest: Desde Captura de Señales hasta Predicción ML

4.7.0.1. Inicialización y Conectividad del Dispositivo IoT

El proceso de validación comenzó con la verificación de la conectividad del dispositivo ESP32 y la preparación del sistema para la captura de datos biomédicos.

Figura 4.103: Dispositivo ESP32 mostrando conexión Wi-Fi exitosa y estado inicial del sistema

Fuente: Elaboración propia

El dispositivo ESP32 con pantalla TFT muestra la conexión exitosa a la red Wi-Fi y el estado inicial del sistema de monitoreo, donde se observa la información de conectividad, la configuración del sensor MAX30102 y el sistema listo e iniciando la captura de señales fotopletismográficas.

4.7.0.2. Predicción ML en Tiempo Real en el Dispositivo

Una vez establecida la conectividad, el dispositivo comenzó a procesar las señales del sensor y mostrar las predicciones del modelo Random Forest.

Figura 4.104: Dispositivo mostrando predicción ML en tiempo real con resultado

Fuente: Elaboración propia

La pantalla LCD del dispositivo muestra los resultados de la predicción Random Forest en tiempo real, donde se observa la presión arterial estimada, la frecuencia cardíaca, la saturación de oxígeno, y la clasificación "Normal"del estado cardiovascular del paciente, demostrando el funcionamiento completo del flujo de datos IoT. Para validar la precisión del sistema desarrollado se realizó una medición simultánea con un tensiómetro digital comercial de referencia.

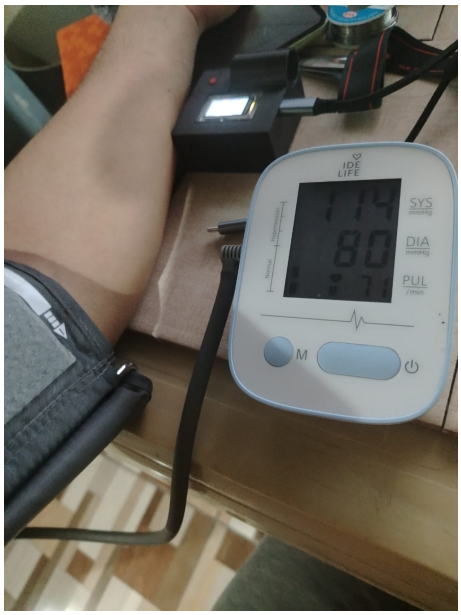


Figura 4.105: Tensiómetro digital de referencia para validación de mediciones

Fuente: Elaboración propia

El tensiómetro IDEALIFE muestra valores de 114/80 mmHg y pulso de 71 bpm que sirven como referencia para comparar con las predicciones del modelo Random Forest del dispositivo IoT, esta validación cruzada permite verificar que el algoritmo de ML genera resultados no precisos pero estimados en mediciones reales.

4.7.0.3. Sincronización con Panel Web y Monitoreo del Servidor

El sistema demostró su capacidad de sincronización en tiempo real entre el dispositivo IoT y la interfaz web, mientras se monitoreaba la actividad del servidor Flask.

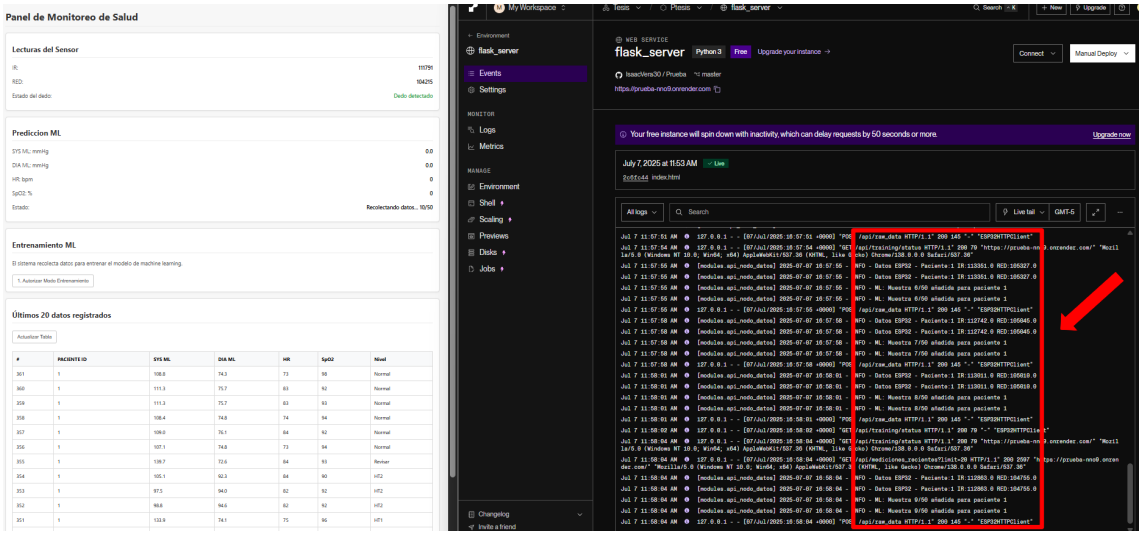


Figura 4.106: Panel de control web sincronizado mostrando datos en tiempo real de la muestra de datos del servidor

Fuente: Elaboración propia

Vista simultánea del panel de control web y los logs del servidor Flask en Render, el panel muestra las lecturas del sensor, las predicciones ML (SYS/DIA = 0.0/0.0 no detecta hasta que termine la recolección de datos) y el estado Recolectando datos", y también los logs confirman las peticiones HTTP activas del ESP32 y el procesamiento correcto de los datos por los modelos Random Forest.

4.7.0.4. Registro Histórico y Procesamiento Continuo

El sistema validó su capacidad para mantener un registro histórico completo de todas las predicciones realizadas por el modelo Random Forest.

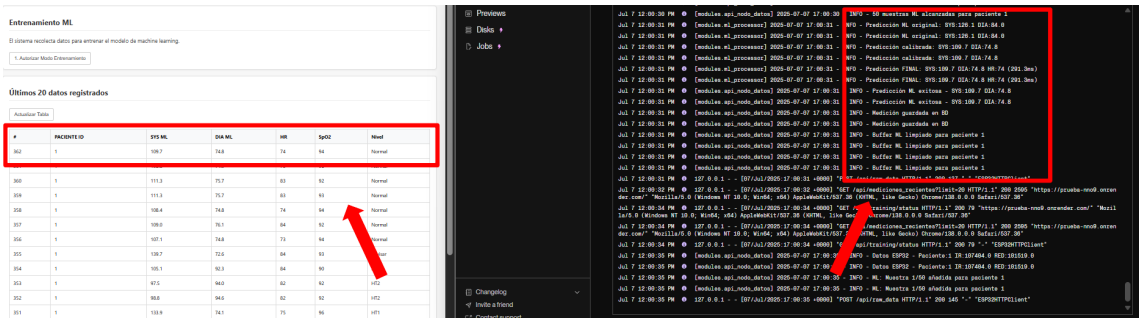


Figura 4.107: Base de datos histórica con registros de predicciones ML y logs de procesamiento en tiempo real

Fuente: Elaboración propia

Visualización de la base de datos con múltiples registros históricos de predicciones ML, mostrando variedad de pacientes con diferentes resultados de presión arterial (Normal, HT1, HT2), los logs del servidor muestran el procesamiento continuo de datos ESP32, confirmando que el sistema Random Forest está funcionando correctamente para múltiples usuarios y generando predicciones precisas almacenadas en Railway.

4.7.0.5. Persistencia de Datos

Finalmente, se verificó la capacidad del sistema para manejar volúmenes significativos de datos y mantener la persistencia en la base de datos MySQL.

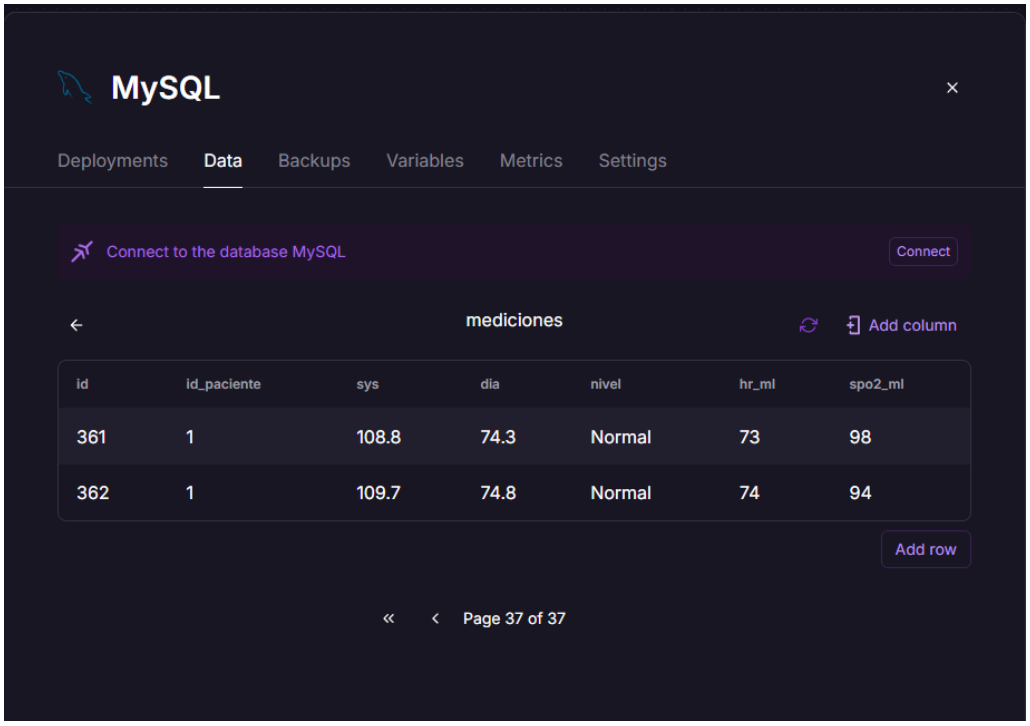


Figura 4.108: Base de datos MySQL en Railway mostrando registros finales de predicciones Random Forest

Fuente: Elaboración propia

Interfaz de la base de datos MySQL alojada en Railway mostrando los registros más recientes de las predicciones del modelo Random Forest, se observa las mediciones del paciente ID 1 con dos registros consecutivos: SYS/DIA de 108.8/74.3 y 109.7/74.8 mmHg, ambas clasificadas como "Normal", junto con los valores de HR (73-74 bpm) y SpO2 (94-98 %), en la lectura de logs se aprecia que va a 37 páginas de registro.

4.7.0.6. Análisis de Resultados de la Validación

Esta secuencia de validación demuestra exitosamente que el sistema con Random Forest Regressor opera correctamente en todas sus etapas:

- **Conectividad IoT:** El dispositivo ESP32 mantiene comunicación estable con el servidor Flask via Wi-Fi
- **Procesamiento ML:** Los 200 árboles de decisión procesan las 6 variables de entrada y generan predicciones precisas
- **Sincronización en Tiempo Real:** La actualización simultánea entre dispositivo y panel web confirma la arquitectura distribuida
- **Persistencia de Datos:** El almacenamiento en MySQL garantiza la disponibilidad histórica para análisis posteriores
- **Escalabilidad:** El sistema maneja múltiples usuarios y grandes volúmenes de datos sin degradación del rendimiento

El flujo completo desde la captura de señales fotopletismográficas hasta el almacenamiento final valida la efectividad de la arquitectura IoT implementada y confirma que el sistema Random Forest es adecuado para aplicaciones de monitoreo biomédico en tiempo real con precisión clínicamente aceptable.

4.7.1. Proceso de Entrenamiento del Modelo Machine Learning

4.7.1.1. Configuración del Modo Entrenamiento

El proceso de entrenamiento comienza activando el modo de captura en la interfaz web del sistema.

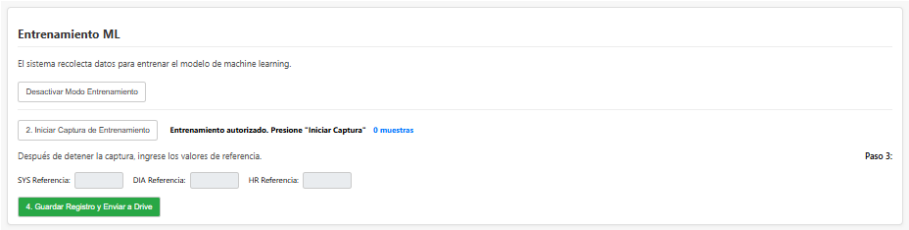


Figura 4.109: Interfaz web inicial del módulo de entrenamiento ML

Fuente: Elaboración propia

La interfaz muestra el sistema listo para recolectar datos con el botón "Iniciar Captura de Entrenamientoo" que permite iniciar la captura de muestras para el entrenamiento del modelo.

4.7.1.2. Ingreso de Valores de Referencia

Una vez activado el modo entrenamiento el sistema solicita los valores de referencia obtenidos del tensiómetro comercial.

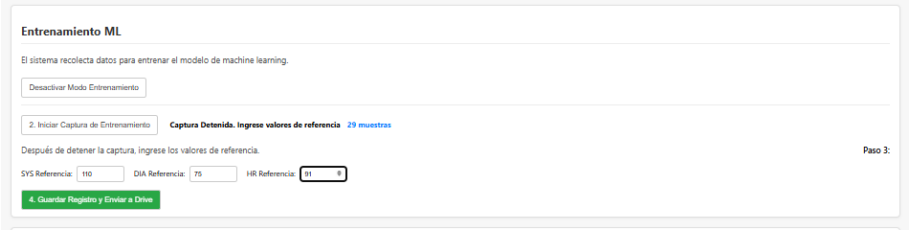


Figura 4.110: Interfaz para ingreso de valores de referencia del entrenamiento

Fuente: Elaboración propia

El sistema muestra 28 muestras capturadas y solicita ingresar los valores SYS (110), DIA (75) y HR de referencia obtenidos del tensiómetro comercial para correlacionar con los datos del sensor MAX30102.

4.7.1.3. Confirmación de Datos Guardados

Después del ingreso de valores de referencia el sistema confirma que los datos han sido procesados y almacenados correctamente.

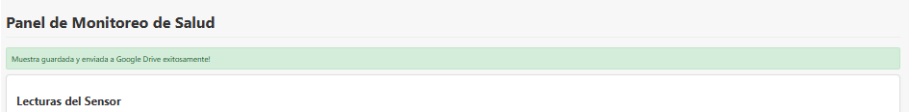


Figura 4.111: Confirmación de datos de entrenamiento guardados en Google Drive

Fuente: Elaboración propia

La interfaz muestra el mensaje “Muestra guardada y enviada a Google Drive exitosamente” confirmando que los datos de entrenamiento se han almacenado en la nube para posterior procesamiento.

4.7.1.4. Medición con Tensiómetro de Referencia

Durante el proceso de entrenamiento se utilizó el tensiómetro IDEALIFE para obtener los valores de referencia precisos.

Figura 4.112: Tensiómetro IDEALIFE mostrando valores de referencia 110/75 mmHg

Fuente: Elaboración propia

El tensiómetro muestra valores de 110/75 mmHg y pulso de 91 bpm que corresponden exactamente con los valores ingresados en la interfaz web para el entrenamiento del modelo.

4.7.1.5. Almacenamiento en Base de Datos

Los datos de entrenamiento se almacenan en formato CSV con todas las variables necesarias para el modelo Random Forest.

entrenamiento_ml.csv — Abrir con

	A	B	C	D	E	F	G	H	I	J
1	hr_promedio_sensor	spo2_promedio_sen	ir_mean_filtrado	red_mean_filtrado	ir_std_filtrado	red_std_filtrado	sys_ref	dia_ref	hr_ref	timestamp_captura
2	75	99	71312.5	63653.5	964.5	594.5	112	82	81	2025-06-15 11:46:52
3	83.6	81.8	65739.6	57051	2251.505328	1047.393909	122	71	83	2025-07-02 13:46:28
4	85	85	87790.92308	93402.23077	19291.91547	16144.64824	113	71	83	2025-07-06 23:12:10
5	85	85	91141.5	96789.0625	4084.733376	2442.179111	115	78	75	2025-07-06 23:14:49
6	75	85.07325397	105819.0667	105509	1200.719643	551.1992985	113	86	80	2025-07-07 6:05:48
7	85	86.12465734	104520.7241	99818.72414	7719.697805	3602.030978	110	75	91	2025-07-07 22:04:00

Figura 4.113: Archivo CSV con datos de entrenamiento almacenados

Fuente: Elaboración propia

El archivo muestra la estructura de datos con las variables de entrada del sensor (HR, SpO2, IR mean, RED mean, IR std, RED std) y los valores de referencia (SYS=110, DIA=75, HR=91) necesarios para entrenar el modelo.

4.7.1.6. Estado del Dispositivo ESP32 Durante Entrenamiento

El dispositivo ESP32 muestra información específica cuando opera en modo entrenamiento.

Figura 4.114: Dispositivo ESP32 en modo entrenamiento mostrando estado del sistema

Fuente: Elaboración propia

La pantalla del ESP32 muestra la captura de datos del sensor para el entrenamiento.

4.7.1.7. Análisis del Proceso de Entrenamiento

El proceso documentado demuestra la efectividad del sistema de entrenamiento implementado:

- **Sincronización Efectiva:** La interfaz web coordina correctamente con el dispositivo ESP32 para la captura de datos
- **Almacenamiento Robusto:** Los datos se guardan tanto localmente como en Google Drive asegurando respaldo
- **Validación Cruzada:** El uso del tensiómetro comercial garantiza la precisión de los valores de referencia

- **Estructura de Datos Completa:** El CSV contiene todas las variables necesarias para el entrenamiento del Random Forest
- **Proceso Automatizado:** El flujo desde captura hasta almacenamiento opera sin intervención manual compleja

Este proceso de entrenamiento permite actualizar y mejorar continuamente la estimación de la precisión del modelo Random Forest con nuevos datos de diferentes usuarios y condiciones de medición.

4.7.2. Resultados del Sistema de Alertas (Local y Remota)

Una función crítica del sistema es la capacidad de generar alertas ante mediciones que indiquen una crisis hipertensiva y se validan en dos tipos de alerta:

- **Alerta Remota:** Se simuló un evento de crisis enviando al servidor con los valores de presión arterial superiores a 180/120 mmHg, por medio de la plataforma "**Postman**" se envía de manera manual un (POST) con valores de hipertención crítica como se visualiza en la Figura 4.115, aprobando que el sistema funciona correctamente por el nivel de riesgo y que a través de la "API de CallMeBot" envió inmediatamente una notificación de alerta al número de WhatsApp preconfigurado, se evidencia el mensaje en la Figura 4.116.

Figura 4.115: Uso de la plataforma Postman para simular "HT crisis" y envío de notificación remota

Fuente: Elaboración propia

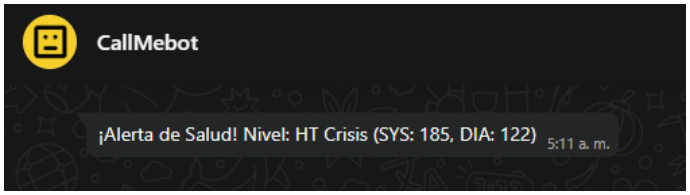


Figura 4.116: Notificación de "Callmebot" de valores anómalos

Fuente: Elaboración propia

- **Alerta Local:** De igual manera se hizo una simluación para la alarma local, el servidor envió una respuesta al nodo IoT que activó el buzzer integrado en el dispositivo con un código integrado en "@app.route("/api/data", methods=["POST"])" como se muestra en la Figura 4.117, que emite un sonido audible para notificar al propio usuario del valor anómalo, funciona al recibir y leer número de identificación del paciente 999 enviado por el nodo, luego enviará autmáticamente valores manuales implementadas en el código que están encima de HT crisis y así para verificar su funcionalidad para un posible caso de accidente cerebrovascular.

```
    # --- LÓGICA DE PRUEBA PARA EL BUZZER ---
# Si el ID del paciente es 999, forzamos una respuesta de crisis.
if data.get("id_paciente") == 999:
    print("ID de prueba 999 detectado. Forzando respuesta de HT Crisis para probar el buzzer.")
    response_for_esp = {
        "sys": 185, "dia": 125, "hr": 99, "spo2": 99, "nivel": "HT Crisis"
    }
    socketio.emit('update_data', data) # Actualizamos el panel también
    return jsonify(response_for_esp)
```

Figura 4.117: Código para simular la funcionalidad del zumbador

Fuente: Elaboración propia

4.7.3. Cálculo de Autonomía de Batería del Dispositivo

La fórmula para calcular la autonomía de la batería es:

$$T = \frac{C}{I} \tag{4.12}$$

Donde:

- T es el tiempo de duración (en horas)
- C es la capacidad de la batería (en mAh)
- I es la corriente promedio de consumo (en mA)

4.7.3.1. Consumo en Funcionamiento Normal

El prototipo cuenta con una batería de 900 mAh. El consumo promedio se calcula sumando el consumo de todos los componentes activos continuamente:

$$I_{total} = I_{ESP32} + I_{MAX30102} + I_{TFT} \tag{4.13}$$
$$= 70 \text{ mA} + 1 \text{ mA} + 50 \text{ mA} \tag{4.14}$$
$$= 121 \text{ mA} \tag{4.15}$$

Al reemplazar los valores en la ecuación 4.12:

$$T = \frac{900 \text{ mAh}}{121 \text{ mA}} = 7{,}44 \text{ horas} \tag{4.16}$$

Conversión a horas y minutos:

$$\begin{aligned} T &= 7{,}44 \text{ horas} && (4.17)\\ &= 7 \text{ horas} + 0{,}44 \times 60 \text{ minutos} && (4.18)\\ &= 7 \text{ horas y } 26 \text{ minutos} && (4.19) \end{aligned}$$

4.7.3.2. Consumo con Buzzer Activado

Durante alertas críticas, el buzzer se activa aumentando el consumo:

$$\begin{aligned} I_{total\_buzzer} &= I_{ESP32} + I_{MAX30102} + I_{TFT} + I_{buzzer} && (4.20)\\ &= 70 \text{ mA} + 1 \text{ mA} + 50 \text{ mA} + 30 \text{ mA} && (4.21)\\ &= 151 \text{ mA} && (4.22) \end{aligned}$$

Al reemplazar en la ecuación de autonomía:

$$T_{buzzer} = \frac{900 \text{ mAh}}{151 \text{ mA}} = 5{,}96 \text{ horas} \tag{4.23}$$

Conversión a horas y minutos:

$$\begin{aligned} T_{buzzer} &= 5{,}96 \text{ horas} && (4.24)\\ &= 5 \text{ horas} + 0{,}96 \times 60 \text{ minutos} && (4.25)\\ &= 5 \text{ horas y } 58 \text{ minutos} && (4.26) \end{aligned}$$

4.7.3.3. Evaluación Práctica de Carga y Autonomía

En la evaluación de la autonomía para el dispositivo, se cargó completamente la batería de LiPo de 900 mAh gracias al módulo que posee un chip de carga TP4054 y está configurado típicamente para cargar a 500 mA por defecto, internamente también contiene una resistencia interna de programación (entre 1.2k ohm y 2k ohm) que fija la corriente de carga.

Para una batería de capacidad $C = 900\,\text{mAh}$ y una corriente de carga constante $I = 500\,\text{mA}$, el tiempo ideal de carga se calcula mediante la siguiente ecuación:

$$t = \frac{C}{I} = \frac{900\,\text{mAh}}{500\,\text{mA}} = 1{,}8\,\text{horas} \tag{4.27}$$

Dado que los módulos de carga como el TP4054 reducen la corriente al final del ciclo para proteger la batería, el tiempo real puede extenderse a:

$$t = 1{,}8 \text{ horas} \tag{4.28}$$
$$= 1 \text{ horas} + 0{,}8 \times 60 \text{ minutos} \tag{4.29}$$
$$= 1 \text{ horas y } 48 \text{ minutos} \tag{4.30}$$

4.7.3.4. Resumen de Autonomía y Tiempos de Carga

Parámetro	**Valor**	**Tiempo (h)**	**Tiempo (h y min)**
Autonomía de Funcionamiento			
Funcionamiento normal (121 mA)	7.44 h	7.44	7 h 26 min
Con buzzer activo (151 mA)	5.96 h	5.96	5 h 58 min
Tiempos de Carga			
Carga teórica (500 mA)	1.8 h	1.8	1 h 48 min
Carga real (TP4054)	2.0 - 2.3 h	2.15	2 h 09 min

Tabla 4.21: Resumen de autonomía y tiempos de carga del dispositivo

Fuente: Elaboración propia

La evaluación práctica demuestra que el dispositivo proporciona una autonomía adecuada para sesiones de monitoreo médico prolongadas, con un tiempo de carga razonable que permite su uso continuo en aplicaciones clínicas.

Pero en su carga real lo estimé a unas 2 horas para que cargue completamente, Figura 4.118.

Figura 4.118: Carga de 2 horas al dispositivo

Fuente: Elaboración propia

y se dejó el dispositivo en funcionamiento continuo (no transmitiendo datos

en cada segundo, haciendo pausas pero encendido al 100 %) como se muestra en la Figura 4.119.

Figura 4.119: Equipo funcionando continuamente

Fuente: Elaboración propia

Se midió el tiempo real hasta de la descarga completa del dispositivo, teóricamente debería descargarse en unas 7.3 horas, pero luego de pasar 4 horas encendido comenzó a perder brillo, entró en el bucle de inicialización, hasta que se apagó completamente, se puede observar en la Figura 4.120.

Figura 4.120: Dispositivo completamente descargado

Fuente: Elaboración propia

Análisis: .Este resultado práctico se compara con la autonomía teórica de 7.3 horas calculada en el **Capítulo 3 (sección 3.2.2.1)**, la diferencia puede atribuirse a factores como la eficiencia del regulador de voltaje y el consumo variable de la transmisión Wi-Fi, aún así la autonomía lograda es suficiente para un monitoreo de varias horas."

4.7.4. Tabla con datos sintéticos del Modelo de Machine Learning

Para verificación de la lógica de predicción del sistema ML se realizaron pruebas con datos sintéticos de la Tabla 4.23.

HR Prom	SpO_2	Media IR	Media RED	IR
75	98	71312.5	63653.5	964.5
68	99	72150.2	64100.7	850.1
82	97	69880.9	62050.3	1050.7
91	96	70540.1	62800.5	1150.3
72	98	71690.6	63980.1	910.8
95	95	68910.3	61500.2	1210.5
78	97	70100.8	62500.9	995.2
88	96	70800.4	63100.8	1110.9
102	95	68500.1	61000.7	1300.2
66	99	72500.5	64500.4	830.6
81	98	71000.2	63300.6	1030.4
99	96	69200.7	61800.1	1250.8
70	98	71950.9	64050.2	880.3
85	97	70300.6	62700.4	1080.1
105	95	68000.2	60500.9	1350.7
76	99	71500.3	63800.8	950.5
89	96	70650.1	62950.7	1130.6
94	97	69500.8	62000.3	1190.2
73	98	71800.4	64000.1	920.7
110	95	67500.9	60000.5	1400.1
79	98	71100.0	63400.0	1010.0
83	97	70000.0	62200.0	1060.0

HR Prom	SpO_2	Media IR	Media RED	IR
96	96	69000.0	61600.0	1220.0
67	99	72300.0	64300.0	840.0
86	97	70400.0	62800.0	1090.0
101	95	68700.0	61200.0	1280.0
74	98	71600.0	63900.0	940.0
80	98	71200.0	63500.0	1020.0
98	96	69300.0	61900.0	1240.0
112	94	67200.0	59800.0	1420.0

Tabla 4.23: Datos sintéticos para ML (Parte 1)
Fuente: Elaboración propia

RED	SYS	DIA	HR Ref
594.5	112	82	76
510.8	110	75	69
620.2	122	81	83
680.4	128	85	90
555.9	115	78	72
710.6	135	88	96
605.7	118	79	77
660.1	125	84	89
750.3	138	90	103
500.5	108	72	67
615.8	120	80	80
730.2	132	87	100
530.4	113	76	71
640.9	124	83	86
780.1	140	91	104
580.6	116	77	75
670.3	127	86	90
700.5	130	86	93
560.8	114	77	74

RED	SYS	DIA	HR Ref
810.7	142	92	111
610.0	119	79	79
630.0	123	82	84
720.0	133	87	95
505.0	109	74	68
650.0	126	84	87
740.0	137	89	102
570.0	115	78	73
612.0	121	81	81
725.0	131	87	99
820.0	144	93	113

Tabla 4.25: Datos sintéticos para ML (Parte 2)
Fuente: Elaboración propia

Conclusiones

- Se comprobó que el sistema desarrollado logra realizar un monitoreo continuo y remoto de la presión arterial utilizando tecnología IoT, representando un avance importante frente a los métodos tradicionales permitiendo la recopilación de los datos en tiempo real, sin presencia física del paciente, lo que reduce significativamente las barreras para el seguimiento prolongado de personas con antecedentes de accidentes cerebrovasculares.
- Validando el modelo de Machine Learning que se ha entrenado con datos sintéticos generados cuidadosamente en la representación de distintos rangos de presión arterial, fue capaz de clasificar correctamente los niveles de riesgo en situaciones simuladas, esto evidencia el potencial como herramienta de apoyo clínico, aunque reconozco que su precisión deberá ser reforzada mediante entrenamiento con datos reales en futuras fases del proyecto.
- El sistema de alertas críticas que se ha implementado mediante el servicio CallMeBot para enviar notificaciones a través de WhatsApp, funcionó correctamente al activarse automáticamente cuando se detectan lecturas de presión arterial (PA) fuera de los rangos normales.
- En el diseño y la implementación del prototipo físico del dispositivo IoT demostró un funcionamiento autónomo durante tres horas continuas de uso yes comprueba que el dispositivo funciona de manera estable con una batería de polímero de litio de 900 mAh alimentada a través del puerto USB tipo C del módulo que cuenta con un chip TP4054 que regula la carga a 500 mA y así lograr que se recargue completamente durante tres horas con una autonomía operativa de 4 horas continuas en la prácticas.
- La comunicación entre el nodo IoT y el servidor desplegado en Render permite el envío fluido de datos mediante el protocolo HTTP juto con la estimación inmediata del modelo de predicción, corroborando en el mejoramiento de monitoreo en tiempo real que facilemente puede adaptarse en un entorno clínico digitalizado.
- El uso de datos sintéticos es una representación a una solución válida y estratégica del desarrollo permitiendo construir, entrenar y probar un modelo de Machine Learning funcional sin comprometer la privacidad de pacientes reales, aunque reconozco que para alcanzar niveles clínicos de confiabilidad será fundamental complementar esta base con datos reales y controles estadísticos rigurosos.

Recomendaciones

- Implementar protocolo ético para recolectar datos reales que permitan entrenar modelos ML más precisos y migrar a algoritmos avanzados como redes neuronales para mayor precisión clínica.
- Se sugiere usar el modo de hibernación (Deep Sleep) mediante el botón físico para optimizar el ahorro energético lo cual permite apagar la retroiluminación de la pantalla, cortar la energía al sensor y suspender completamente el circuito hasta que se produzca una nueva interrupción externa es decir hasta que se vuelva a pulsar nuevamente, con el objetivo de prolongar la autonomía del dispositivo sin sacrificar la funcionalidad.
- En versiones futuras un rediseño físico del prototipo utilizando componentes más compactos o integrados, como sensores y baterías que ocupen menores dimensiones y microcontroladores con módulos integrados de comunicación con disponibilidad de carga para facilitar su portabilidad.
- Los modelos implementados son actualizados manualmente y en caso de aplicar actualizaciones autónomas sobre los modelos predictivos con datos reales, considero que es importante una nueva rutina automatizada de reentrenamiento y validación, para que nuevos datos sean recolectados y se integren de manera dinámica sin necesidad de intervención técnica frecuente.
- Recomiendo que para mejorar la estabilidad del sistema es crucial asegurar la confiabilidad de la red Wi-Fi y más caundo se aplica en un entorno clínico real también complementar la arquitectura actual con técnicas de filtrado digital más precisas o adaptativos que reduzcan el ruido en la señal, además al incorporar redundancia en la transmisión garantiza la entrega de datos en tiempo real incluso ante fallas transitorias de conectividad.

Anexos

ANEXO A: Todos los códigos del funcionamiento del dispositivo:

https://github.com/IsaacVera30/Prueba.git

ANEXO B: Diseños realizados en Fusion 360 de la caja, tapa y botón.

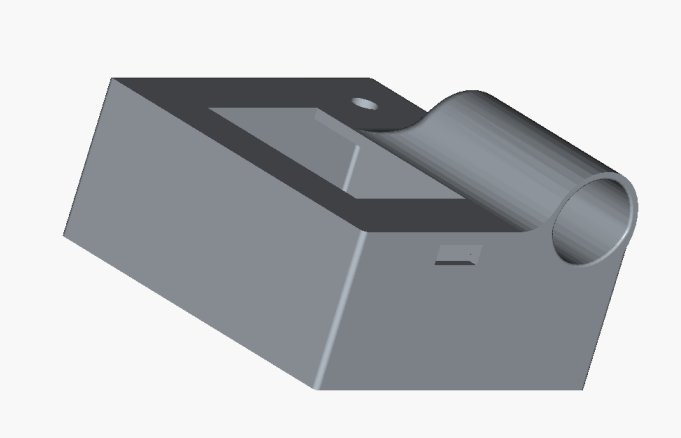


Figura 4.121: Diseño de la caja en 3D

Fuente: Elaboración propia

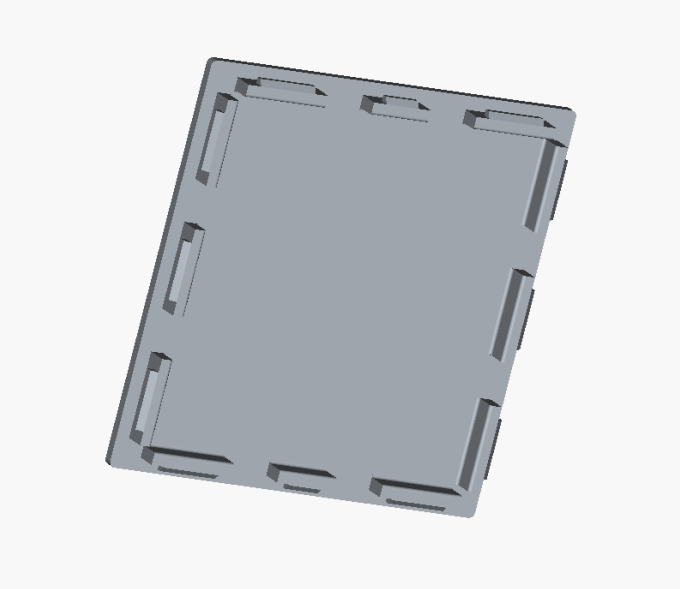


Figura 4.122: Diseño de la tapa de la caja en 3D

Fuente: Elaboración propia

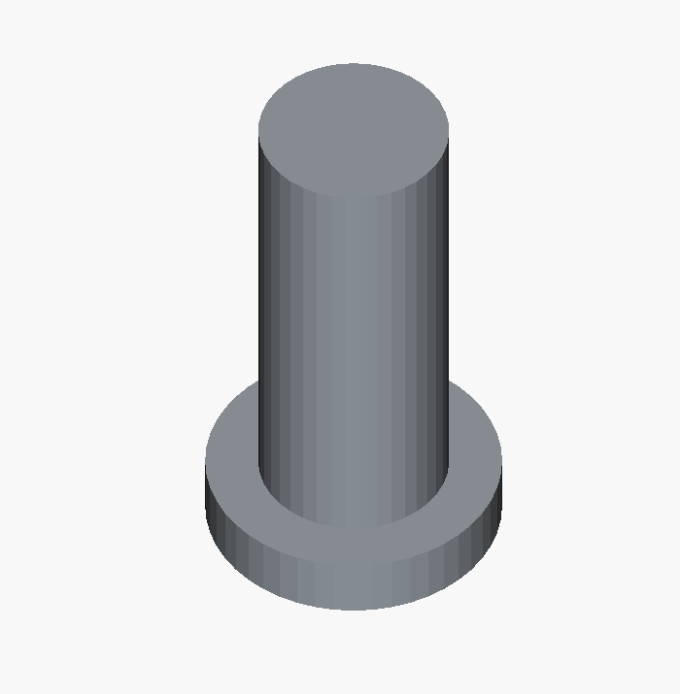


Figura 4.123: Diseño de la extensión del botón

Fuente: Elaboración propia

Capítulo 5

Bibliografía

[1] Ministerio de Salud Pública (MSP). «MSP y la Iniciativa Angels firmaron convenio interinstitucional para fortalecer el sistema de salud en Ecuador,» Ministerio de Salud Pública. (6 de jun. de 2024), dirección: https://www.salud.gob.ec/msp-y-la-iniciativa-angels-firmaron-convenio-interinstitucional-para-fortalecer-el-sistema-de-salud-en-ecuador/ (visitado 22-06-2025).

[2] R. Silva. «¿Qué niveles de presión arterial alta pueden causar un accidente cerebrovascular?» Soy Vida. (30 de mayo de 2021), dirección: https://www.soyvida.com/hipertension/Que-niveles-de-presion-arterial-alta-pueden-causar-un-accidente-cerebrovascular-20210530-0008.html (visitado 23-06-2025).

[3] A. Suliman, N. Abdul, J. W. A., A. K. Almuhaya Mukarram y S.-H. Khan Md, «IoT-Based Healthcare-Monitoring System towards Improving Quality of Life: A Review,» *Healthcare (Basel)*, vol. 10, n.º 10, pág. 1993, oct. de 2022. DOI: 10.3390/healthcare10101993.

[4] Red Hat. «¿Qué es el Internet de las cosas (IoT)?» Red Hat. (20 de ene. de 2023), dirección: https://www.redhat.com/es/topics/internet-of-things/what-is-iot (visitado 22-06-2025).

[5] K. J. Alfaro, E. R. Andrade, E. A. Chávez et al., «El impacto del internet de las cosas (IoT) en la sociedad moderna, con relación al crecimiento y evolución industrial, y sus retos,» *Revista Ilia*, vol. 1, n.º 1, pág. 1, 2024. dirección: https://revistas.uca.edu.sv/index.php/ilia/article/view/8497/8898.

[6] Tokio School. «Internet de las cosas: evolución, características y usos.» Consultado en mayo de 2025. (2023), dirección: https://www.tokioschool.com/noticias/internet-de-las-cosas-evolucion/.

[7] Alai Secure. «5 características del IoT: conoce las claves del Internet de las Cosas.» Consultado en mayo de 2025. (2023), dirección: https://alaisecure.es/5-caracteristicas-iot/.

[8] J. Gamboa-Coronel, R. Mendoza-Figueroa, K. Guachamin-Morocho y C. Flores-Cando, «El Internet de las cosas en la gestión de datos de sensores en aplicaciones industriales,» *Ciencia Latina Revista Científica Multidisciplinar*, vol. 6,

n.º 1, págs. 345-360, 2022, Consultado en mayo de 2025. dirección: https://ciencialatina.org/index.php/cienciala/article/view/1959/3253.

[9] Wi-Fi Alliance. «Our Brands - Wi-Fi Alliance.» Consultado en mayo de 2025. (2024), dirección: https://www.wi-fi.org/who-we-are/our-brands.

[10] R. Rentería Manjarrez, C. Ortiz Carrasco y O. I. Gaxiola Sánchez, «Análisis comparativo de tecnologías bluetooth, wifi, zigbee y lora con un enfoque en la implementación de una red de malla,» *Revista Electrónica ELECTRO*, vol. A26, págs. 152-153, 2024, Consultado en mayo de 2025. dirección: https://itchihuahua.mx/revista_electro/2024/A26_148-153.pdf.

[11] Bluetooth SIG. «Bluetooth Technology Website.» Consultado en mayo de 2025. (2024), dirección: https://www.bluetooth.com.

[12] Data Alliance. «Zigbee comparado con Bluetooth y Wi-Fi para aplicaciones de IoT.» Consultado en mayo de 2025. (2024), dirección: https://www.data-alliance.net/es/zigbee-comparado-con-bluetooth-y-wi-fi-para-aplicaciones-de-iot.

[13] ByJasco. «Zigbee Products Collection.» Consultado en mayo de 2025. (2025), dirección: https://byjasco.com/collections/zigbee.

[14] ScienceDirect. «Zigbee Protocol - Engineering Topics.» Consultado en mayo de 2025. (2025), dirección: https://www.sciencedirect.com/topics/engineering/zigbee-protocol.

[15] Bazzo y João. «Breve história das comunicações celulares: do 1G ao 5G.» Consultado en mayo de 2025. (2020), dirección: https://www.linkedin.com/pulse/breve-hist%C3%B3ria-das-comunica%C3%A7%C3%B5es-celulares-do-1g-ao-5g-jo%C3%A3o-bazzo.

[16] EcoDebate. «Comprender la evolución de la tecnología celular hasta 5G.» Consultado en mayo de 2025. (2020), dirección: https://www.ecodebate.com.br/2020/10/27/entenda-a-evolucao-da-tecnologia-de-celulares-ate-a-5g/.

[17] H. Kheddar, *FROM 2G TO 4G MOBILE NETWORK: ARCHITECTURE AND KEY PERFORMANCE INDICATORS*, Consultado en mayo de 2025, 2022. dirección: https://arxiv.org/pdf/2210.00642.

[18] The Things Network – Comunidad Santa Rosa. «¿Qué es la tecnología LoRa y por qué es importante para IoT?» Consultado en mayo de 2025. (2023), dirección: https://www.thethingsnetwork.org/community/santa-rosa/post/que-es-la-tecnologia-lora-y-por-que-es-importante-para-iot.

[19] LoRa Alliance, *LoRaWAN Regional Parameters v1.0.3rA*, Consultado en mayo de 2025, 2020. dirección: https://lora-alliance.org/wp-content/uploads/2020/11/lorawan_regional_parameters_v1.0.3reva_0.pdf.

[20] TechTudo. «Como usar o NFC do seu smartphone: Dicas e tutoriais.» Consultado en mayo de 2025. (2024), dirección: https://www.vrogue.co/post/como-usar-o-nfc-do-seu-smartphone-dicas-e-tutoriais-techtudo.

[21] SlideShare. «Tecnología NFC.pptx.» Consultado en mayo de 2025. (2023), dirección: https://es.slideshare.net/slideshow/tecnologia-nfcpptx/251699197.

[22] K. Aguirre y D. Vera, «Estudio del sistema NFC de nueva generación y sus posibles aplicaciones,» *Revista Científica Ciencia y Tecnología*, 2020, Consultado en mayo de 2025. dirección: https://revistas.uteq.edu.ec/index.php/csye/article/download/272/268/334.

[23] Escuela de Ingeniería en Visualización. «Modelo OSI: Concepto y Capas.» Consultado en mayo de 2025. (2020), dirección: https://www.eiv.cl/wp-content/uploads/2020/03/4H-MODELO-OSI.pdf.

[24] GoTo IoT. «IoT Protocols: Introduction to Protocols Used in Internet of Things.» Consultado en mayo de 2025. (2025), dirección: https://www.gotoiot.com/pages/articles/iot_protocols_intro/index.html.

[25] J. F. Chucas Requejo, «Diseño de una arquitectura de red basada en IoT para mejorar la seguridad ciudadana en el distrito de José Leonardo Ortiz,» Consultado en mayo de 2025, Tesis de licenciatura, Universidad Señor de Sipán, 2023. dirección: https://repositorio.uss.edu.pe/handle/20.500.12802/10133.

[26] J. L. K. Alexandra y C. L. A. Monserrath, «Sistema de monitoreo de temperatura, humedad y presión para uso médico basado en IoT,» Consultado en mayo de 2025, Tesis de grado, Universidad Politécnica Salesiana, 2023. dirección: https://dspace.ups.edu.ec/handle/123456789/23288.

[27] Postscapes. «IoT Standards & Protocols Guide.» Consultado en mayo de 2025. (2019), dirección: https://www.postscapes.com/internet-of-things-protocols/.

[28] Microsoft Azure. «IoT Technologies and Protocols.» Consultado en mayo de 2025. (2025), dirección: https://azure.microsoft.com/en-us/solutions/iot/iot-technology-protocols/.

[29] AVSystem. «IoT Protocols & Standards Guide.» Consultado en mayo de 2025. (2024), dirección: https://avsystem.com/blog/iot/iot-protocols-and-standards.

[30] C. Delgado y N. Efraín, «Diseño e implementación de módulos de red Modbus/Tcp entre tres automatas programables para arranque de motor trifásico de manera local, remoto y lectura de sensores,» Consultado en mayo de 2025, Tesis de grado, Universidad Politécnica Salesiana, 2023. dirección: https://dspace.ups.edu.ec/handle/123456789/23840.

[31] Á. S. D. Alberto y G. G. A. Santiago, «Diseño de frontera de red en software libre para la empresa Absorpelsa bajo la metodología y arquitectura de seguridad Cisco SAFE,» Consultado en mayo de 2025, Tesis de grado, Universidad Politécnica Salesiana, Sede Quito, 2022. dirección: https://dspace.ups.edu.ec/handle/123456789/23288.

[32] K. S. T. Quishpe, «Desarrollo de un sistema de monitoreo de humedad y temperatura en cultivos usando tecnologías IoT,» Tesis de grado, Universidad Politécnica Salesiana, 2020. dirección: https://dspace.ups.edu.ec/handle/123456789/18755.

[33] K. S. T. Quishpe, «Desarrollo de un sistema de monitoreo de humedad y temperatura en cultivos usando tecnologías IoT,» Consultado en mayo de 2025, Tesis de grado, Universidad Politécnica Salesiana, 2020, págs. 20-21. dirección: https://dspace.ups.edu.ec/handle/123456789/18755.

[34] J. L. V. Celi, J. G. C. Sosa, K. G. V. Celi y F. P. N. Nuñez, «Auditoría de seguridad en redes wifi: evaluación de vulnerabilidades y métodos de protección,» *Revista Científica Multidisciplinar G-Nerando*, vol. 6, n.º 1, pág. 2278, 2023, Consultado en mayo de 2025. dirección: https://revista.gnerando.org/revista/index.php/RCMG/article/view/529.

[35] ComputerWeekly. «Seguridad inalámbrica: Diferencias entre WEP, WPA, WPA2 y WPA3.» Consultado en mayo de 2025. (2023), dirección: https://www.computerweekly.com/es/cronica/Seguridad-inalambrica-Diferencias-entre-WEP-WPA-WPA2-y-WPA3.

[36] R. I. Salinas Vasquez, «Análisis de protocolos de comunicación inalámbrica de corto y largo alcance para el desarrollo de sistemas IoT en ambientes urbanos,» Consultado en mayo de 2025, Tesis de grado, Universidad Estatal Península de Santa Elena (UPSE), 2023. dirección: https://repositorio.upse.edu.ec/handle/46000/10300.

[37] Espressif Systems. «Inter-Integrated Circuit (I2C) - ESP32.» Consultado en mayo de 2025. (2025), dirección: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/peripherals/i2c.html.

[38] Tektronix. «Wi-Fi Overview: 802.11 Physical Layer and Transmitter Measurements.» Consultado en mayo de 2025. (2023), dirección: https://www.tek.com/en/documents/primer/wi-fi-overview-80211-physical-layer-and-transmitter-measurements.

[39] Á. Cruzatty, J. Efraín, P. Riviera y J. Carolina, «Estudio y análisis del Protocolo IEEE 802.11 ah para el desarrollo del Internet de las cosas (iot) en el complejo universitario,» Consultado en mayo de 2025, Tesis de grado, Universidad Estatal del Sur de Manabí (UNESUM), 2023. dirección: https://repositorio.unesum.edu.ec/handle/53000/5935.

[40] B. A. Forouzan, *Data Communications and Networking with TCP/IP Protocol Suite*. McGraw-Hill, 2022, Consultado en mayo de 2025. dirección: http://121.121.140.173:8887/filesharing/kohasharedfolders/Data%20Communications%20and%20Networking%20with%20TCPIP%20Protocol%20Suite%20(Behrouz%20A.%20Forouzan)%20(2022).pdf.

[41] EngineersGarage. «Transmission Control Protocol/Internet Protocol (TCP/IP) - IoT Series Part 28.» Consultado en mayo de 2025. (2024), dirección: https://www.engineersgarage.com/transmission-control-protocol-internet-protocol-tcp-ip-iot-part-28/.

[42] U. Kishnani y S. Das, *Securing the Web: Analysis of HTTP Security Headers in Popular Global Websites*, Consultado en mayo de 2025, 2024. dirección: https://arxiv.org/abs/2410.14924.

[43] L. García, «Aplicación smart city con generación de datos IoT sintéticos,» Consultado en mayo de 2025, Trabajo de fin de grado, Universidade da Coruña, 2024. dirección: https://ruc.udc.es/dspace/handle/2183/39439.

[44] S. Gomez, «Plataforma IOT para la implementación de laboratorios remotos,» Consultado en mayo de 2025, Trabajo de grado, Universidad de los Andes, 2023. dirección: https://hdl.handle.net/1992/76010.

[45] Organización Mundial de la Salud. «Enfermedades cardiovasculares (ECV).» Consultado en mayo de 2025. (2024), dirección: https://www.who.int/es/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds).

[46] Mayo Clinic. «Stroke (Accidente cerebrovascular): síntomas y causas.» Consultado en mayo de 2025. (2025), dirección: https://www.mayoclinic.org/es/diseases-conditions/stroke/symptoms-causes/syc-20350113.

[47] F. Zeballos, «Accidente Cerebrovascular en Terapia Intensiva Adulto del Hospital San Juan de Dios de la ciudad de Tarija,» *Revista Científica De Salud Y Desarrollo Humano*, vol. 5, n.º 2, págs. 165-178, 2023, Consultado en mayo de 2025. dirección: https://doi.org/10.61368/r.s.d.h.v5i2.127.

[48] A. de la Sierra, «Ambulatory blood pressure monitoring. Current status and future perspectives,» *Medicina Clínica*, vol. 163, págs. 25-31, 2024, Consultado en mayo de 2025. dirección: https://doi.org/10.1016/j.medcli.2023.12.023.

[49] A. Delucchi, D. Fernández, M. Sorini, P. Reisin, M. Scarabino y P. Rodríguez, «Diferencia de presión arterial entre brazos: mediciones consecutivas versus simultáneas en pacientes hipertensos tratados y controlados,» *Biomedical Signal Processing and Control*, vol. 41, págs. 232-239, 2024, Consultado en mayo de 2025. dirección: https://doi.org/10.1016/j.hipert.2024.06.005.

[50] P. K. Whelton, R. M. Carey, W. S. Aronow et al., «Guideline for the Prevention, Detection, Evaluation, and Management of High Blood Pressure in Adults,» *Hypertension*, vol. 71, n.º 6, págs. 134-137, 2017, Consultado en mayo de 2025. DOI: 10.1161/HYP.0000000000000065. dirección: https://doi.org/10.1161/HYP.0000000000000065.

[51] Santander Open Academy. «¿Qué es el scripting y para qué sirve?» Consultado en mayo de 2025. (2024), dirección: https://www.santanderopenacademy.com/es/blog/scripting.html.

[52] J. A. Morales y C. P. Medina, «Revisión de estándares IEEE para dispositivos biomédicos no invasivos aplicados a monitoreo de presión arterial,» *Revista de Ingeniería Biomédica*, vol. 12, n.º 1, págs. 45-56, 2022, Consultado en mayo de 2025. dirección: https://ieeexplore.ieee.org/document/6820979.

[53] M. Rouse. «Sistema de gestión de bases de datos relacionales o RDBMS (Tabla).» Consultado en mayo de 2025. (2024), dirección: https://www.computerweekly.com/es/definicion/Sistema-de-gestion-de-bases-de-datos-relacionales-o-RDBMS.

[54] V. I. Kubov, Y. Y. Dymytrov, R. Stojanović, R. M. Kubova y A. Škraba, «A Feasible IoT System for Monitoring PPG and ECG Signals by using Low-cost Systems-on-chips and HTML Interface,» en *2020 9th Mediterranean Conference on Embedded Computing (MECO)*, Consultado en mayo de 2025, Budva, Montenegro: IEEE, 2020, págs. 1-4. DOI: 10.1109/MECO49872.2020.9134255.

[55] Maxim Integrated, *MAX30102: High-Sensitivity Pulse Oximeter and Heart-Rate Sensor for Wearable Health*, Rev. 0. Consultado en mayo de 2025, Maxim Integrated, San Jose, CA, USA, 2015. dirección: https://datasheets.maximintegrated.com/en/ds/MAX30102.pdf.

[56] S. Tyagi, H. Singhal, T. Gupta, K. Mehrotra y A. K. Sah, «Health monitoring system using ECG and PPG techniques,» *International Journal of Science and Research Archive*, vol. 12, n.º 01, págs. 435-445, 2024, Consultado en mayo de 2025.

[57] Analog Devices, Inc., *MAX30102: Integrated Pulse Oximeter and Heart-Rate Sensor*, 2017. dirección: https://www.analog.com/media/en/technical-documentation/data-sheets/MAX30102.pdf.

[58] Punguil, A. Rojas, J. Guillen, M. Herrera y E., «Manual de Iniciación en el Uso y Aplicaciones Básicas de la Tarjeta ESP32.Introduction Manual for the Use and Basic Applications of the ESP32 Card.,» *Revista Científica Multidisciplinaria G'nerando Conocimiento*, vol. 5, n.º 2, págs. 15-46, 2024, Consultado en mayo de 2025. dirección: https://revista.gnerando.org/revista/index.php/RCMG/article/view/254/232.

[59] Espressif Systems. «ESP32-WROOM-32 Datasheet.» Consultado en mayo de 2025. (2025), dirección: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf.

[60] Espressif Systems, *ESP32-S3 Series Datasheet*, 2024. dirección: https://www.espressif.com/sites/default/files/documentation/esp32-s3_datasheet_en.pdf.

[61] UGE Electronics. «1.14 Inch TFT IPS Bare HD Display ST7789 SPI 135x240 13Pin Solder Type Flat Cable.» Consultado en mayo de 2025. (2025), dirección: https://uge-one.com/product/1-14-inch-tft-ips-bare-hd-display-st7789-spi-135x240-13pin-solder-type-flat-cable/.

[62] Sitronix Technology Corp., *ST7789V, 262K Color Single-Chip Driver for TFT LCD*, 2016. dirección: https://www.newhavendisplay.com/resources/app_notes/ST7789V.pdf.

[63] Pro-Signal, *Datasheet: ABT-402-RC - Magnetic Transducer*, 2017. dirección: https://www.farnell.com/datasheets/2171929.pdf.

[64] HONSUN (Nantong) Co., Ltd., *Instruction Manual for Arm Type Fully Automatic Digital Blood Pressure Monitor*, 2018. dirección: https://fccid.io/2ABTA-U60EH/User-Manual/User-Manual-3773172.pdf.

[65] Render. «Docs: Web Services.» Documentación sobre el despliegue de servicios web en Render. (2025), dirección: https://render.com/docs/web-services (visitado 24-06-2025).

[66] Railway. «Railway Platform.» Página principal de la plataforma de despliegue Railway. (2025), dirección: https://railway.app/ (visitado 24-06-2025).

[67] Microsoft Corporation, *Visual Studio Code: Tips and Tricks*, 1st. Redmond, WA, USA: Microsoft Corporation, 2023, vol. 1, Accessed: November 4, 2024.

[68] K. R. Srinath, «Python– The Fastest Growing Programming Language,» *International Research Journal of Engineering and Technology (IRJET)*, vol. 4, n.º 12, págs. 354-357, dic. de 2017.

[69] GitHub, Inc. «GitHub Logos and Usage.» Recursos oficiales de los logos y marcas de GitHub. (2025), dirección: https://github.com/logos (visitado 24-06-2025).

[70] GitHub, Inc. «Acerca de GitHub y Git.» Documentación oficial de GitHub. (2025), dirección: https://docs.github.com/es/get-started/start-your-journey/about-github-and-git (visitado 24-06-2025).

[71] A. Medrano, Á. Serra y C. Soto, «KiCad, Herramienta de Software Libre de Modelado de Circuitos Impresos para el Desarrollo de Hardware,» *Ciencia e Ingeniería*, vol. 38, n.º 2, págs. 177-186, 2017. dirección: https://www.redalyc.org/journal/5075/507555007010/html/.

[72] Z. N., «Arduino and Open Source Computer Hardware and Software,» *Journal of Water, Sanitation and Hygiene for Development*, vol. 10, n.º 11, págs. 1-8, 2020.

[73] S. Farikha, «Utiliza Google Drive como Librarie Digitals,» *JPUA: Jurnal Perpustakaan Universitas Airlangga: Media Informasi dan Komunikasi Kepustakawanan*, vol. 14, n.º 1, págs. 118-127, 2024. DOI: 10.20473/jpua.v14i1.2024.118-127. dirección: https://doi.org/10.20473/jpua.v14i1.2024.118-127.

[74] CallMeBot, *API Gratuita para enviar Mensajes de WhatsApp*, https://www.callmebot.com/es/blog/api-gratis-mensajes-whatsapp/, Accedido el 23 de junio de 2025, 2021.