UNIVERSIDAD TÉCNICA DE AMBATO

FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E INDUSTRIAL

CARRERA DE INGEN IERÍA EN ELECTRÓNICA Y COMUNICACIONES

Tema:

SISTEMA ELECTRÓNICO DE NOTIFICACIÓN DE EMERGENCIAS BASADO EN IoT PARA LA ASISTENCIA MÉDICA A PERSONAS DE LA TERCERA EDAD.

Trabajo de Titulación Modalidad: Proyecto de Investigación, presentado previo a la obtención del título de Ingeniero en Electrónica y Comunicaciones.

ÁREA: Electrónica y Comunicaciones
LÍNEA DE INVESTIGACIÓN: Tecnología de la información y Sistemas de Control
AUTOR: Jefferson Ismael Guamán Egas
TUTORA: Ing. Ana Pamela Castro Martin, Mg.

AMBATO - ECUADOR

marzo - 2023

APROBACIÓN DEL TUTOR

En calidad de tutor del Trabajo de Titulación, con el tema: SISTEMA ELECTRÓNICO DE NOTIFICACIÓN DE EMERGENCIAS BASADO EN IoT PARA LA ASISTENCIA MÉDICA A PERSONAS DE LA TERCERA EDAD, desarrollado bajo la modalidad Proyecto de Investigación por el Señor Jefferson Ismael Guamán Egas, estudiante de la Carrera de Ingeniería en Electrónica y Comunicaciones, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, de la Universidad Técnica de Ambato, me permito indicar que el estudiante ha sido tutorado durante todo el desarrollo del trabajo hasta su conclusión, de acuerdo a lo dispuesto en el Artículo 15 del Reglamento para obtener el Título de Tercer Nivel, de Grado de la Universidad Técnica de Ambato, y el numeral 7.4 del respectivo instructivo.

Ambato, marzo 2023

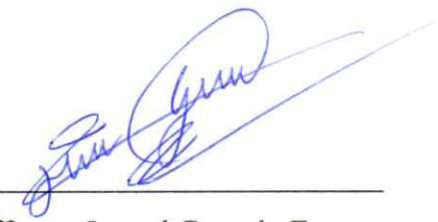
Ing. Ana Pamela Castro Martin, Mg

TUTOR

AUTORÍA

El presente Proyecto de Investigación titulado: SISTEMA ELECTRÓNICO DE NOTIFICACIÓN DE EMERGENCIAS BASADO EN IoT PARA LA ASISTENCIA MÉDICA A PERSONAS DE LA TERCERA EDAD es absolutamente original, autentico y personal. En tal virtud, el contenido, efectos legales y académicos que se desprenden del mismo son de exclusiva responsabilidad del autor.

Ambato, marzo 2023



Jefferson Ismael Guamán Egas

C.C. 1850578178

AUTOR

DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato para que haga uso de este Trabajo de Titulación como un documento disponible para la lectura, consulta y procesos de investigación.

Cedo los derechos de mi Trabajo de Titulación en favor de la Universidad Técnica de Ambato, con fines de difusión pública. Además, autorizo su reproducción total o parcial dentro de las regulaciones de la institución.

Ambato, marzo 2023

Jefferson Ismael Guamán Egas

C.C. 1850578178

AUTOR

APROBACIÓN DEL TRIBUNAL DE GRADO

En calidad de par calificador del Informe Final del Trabajo de Titulación presentado por el señor Jefferson Ismael Guamán Egas, estudiante de la Carrera de Ingeniería en Electrónica y Comunicaciones, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, bajo la Modalidad Proyecto de Investigación, titulado SISTEMA ELECTRÓNICO DE NOTIFICACIÓN DE EMERGENCIAS BASADO EN IoT PARA LA ASISTENCIA MÉDICA A PERSONAS DE LA TERCERA EDAD, nos permitimos informar que el trabajo ha sido revisado y calificado de acuerdo al Artículo 17 del Reglamento para obtener el Título de Tercer Nivel, de Grado de la Universidad Técnica de Ambato, y al numeral 7.6 del respectivo instructivo. Para cuya constancia suscribimos, conjuntamente con la señora Presidenta del Tribunal.

Ambato, marzo 2023

Firmado electrónicamente por:
ELSA PILAR URRUTIA URRUTIA

Ing. Pilar Urrutia, Mg.
PRESIDENTE DEL TRIBUNAL

Firmado electrónicamente por:
FREDDY GEOVANNY BENALCAZAR PALACIOS

Dr. Freddy Benalcázar Palacios, Mg.
PROFESOR CALIFICADOR

Firmado electrónicamente por:
MARIO GEOVANNY GARCIA CARRILLO

Ing. Mario García Carrillo, Mg.
PROFESOR CALIFICADOR

DEDICATORIA

El presente trabajo de titulación está dedicado a Dios, quien con su infinito amor me dio salud y vida para cumplir mis objetivos.

A mis padres María y Rodrigo por brindarme siempre su apoyo incondicional, por sus buenos consejos y enseñanzas. Por inculcarme siempre buenos valores que me han ayudado a lo largo de este camino.

A mi novia Nathaly por apoyarme en todos y cada uno de mis pasos que doy en mi vida, por guiar y sostener en los momentos difíciles.

A mis hermanos, por estar siempre presentes y por su apoyo moral. A mi hermano Alex, aunque ya no se encuentre conmigo peros siempre es y será mi gran inspiración.

Jefferson Ismael Guamán Egas

AGRADECIMIENTOS

Agradezco a Dios por la vida de mis padres y por permitirme adquirir nuevos conocimientos durante el transcurso de la universidad para cumplir todas mis metas.

Gracias a mis padres por ser los principales promotores para cumplir mis sueños, gracias a toda mi familia por apoyarme en cada decisión y proyecto.

Un agradecimiento especial a mi tutora Ing. Pamela Castro por guiarme en este trabajo de investigación, por su comprensión y apoyo con sus conocimientos.

Finalmente, a mis amigos por el apoyo moral y por siempre ofrecerme la mano en los momentos difíciles para juntos llegar hacer profesionales, gracias.

Jefferson Ismael Guamán Egas

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN EJECUTIVO

En el presente trabajo se implementa un sistema con arquitectura IoT que permite detectar y alertar caídas de adultos mayores, siendo estos eventos de gran incidencia en este grupo de personas; además, se realiza el monitoreo de signos vitales.

Se diseñó un dispositivo tipo pulsera con la tarjeta de desarrollo ESP32, con el sensor MPU6050 para la detección de caídas, mientras que, con los sensores MAX30100 y MLX90614 monitorea la frecuencia cardiaca, saturación de oxígeno y temperatura corporal. Para la detección de caídas se crea modelos basados en "Machine Learning" usando la plataforma "open source" Edge Impulse. Para crear el modelo se utilizó 10 participantes: 5 hombres y 5 mujeres. Cada participante realizó 5 tipos de caídas, cada caída fue repetida por 10 veces obteniendo 50 caídas por cada participante, de forma similar se implementó actividades de la vida diaria que realiza una persona de la tercera edad.

Para la construcción de una aplicación móvil se utilizó Flutter compatible con Android e IOS. Se programó distintos widgets para el desarrollo de la interfaz gráfica a fin de presentar los datos de los sensores, información del usuario, crear alertas sobre el estado de los signos vitales en un intervalo de tiempo y para presentar el historial de alertas. Se empleo dependencias como "provider" para gestionar el estado de la aplicación permitiendo controlar la conectividad y los mensajes entrantes por medio del protocolo MQTT. Para alertar al cuidador del adulto mayor se utiliza notificaciones push basadas en la arquitectura de "Firebase Cloud Messaging" propias de Google.

Se obtuvo un dispositivo IoT en base a un microcontrolador con capacidad de: conectar a la red WiFi para enviar datos a un servidor; ejecutar el modelo de "Machine Learning" para detección de caídas; monitorear la frecuencia cardiaca, saturación de oxígeno y temperatura corporal a través de la muñeca; y generar alertas automáticas en el celular por medio de la aplicación móvil.

Palabras clave: Machine Learning, Edge Impulse, sistema IoT, detección de caídas, signos vitales.

ABSTRACT

In the present work, a system with IoT architecture is implemented that allows detecting and alerting falls in older adults, these events being of great incidence in this group of people; In addition, vital signs are monitored.

A bracelet type device was designed with the ESP32 development board with the MPU6050 sensor for fall detection and the MAX30100 and MLX90614 sensors for monitoring heart rate, oxygen saturation and body temperature. For fall detection, models based on "Machine Learning" are created using the "open source" platform Edge Impulse. To create the model, 10 participants were drawn: 5 men and 5 women. Each participant made 5 types of falls, each fall was repeated 10 times, obtaining 50 for each participant, in a similar way activities of daily living carried out by an elderly person were implemented.

For the mobile application, Flutter compatible with Android and IOS was brought, different widgets were programmed for the development of the graphical interface to build sensor data, user information, create alerts on the status of vital signs at an interval of time and to present the history of alerts. Dependencies such as "provider" were used to manage the state of the application, allowing control of connectivity and incoming messages through the MQTT protocol. To alert the caregiver of the elderly, push notifications based on Google's own "Firebase Cloud Messaging" architecture are used.

An IoT device was obtained based on a microcontroller with the ability to: connect to the WiFi network to send data to a server; run the "Machine Learning" model for fall detection; monitor heart rate, oxygen saturation and body temperature through the wrist; and generate automatic alerts on the cell phone through the application.

Keywords: Machine Learning, Edge Impulse, IoT system, fall detection, vital signs.

CAPÍTULO 1

MARCO TEÓRICO

1.1 Antecedentes investigativos

Para el desarrollo del proyecto se cuenta con investigaciones realizadas en los últimos años con relación al tema, indexadas en repositorios de Universidades y artículos científicos que sustentarán la implementación del proyecto.

En el año 2018, Sai Sathis, Sai Vamsi, Arjun V y Bhavana V, de la Universidad Amrita, realizan un artículo titulado "Implementation of Compact Wearable Fall Detector for the Elderly" publicado en India. El proyecto utiliza como unidad de procesamiento principal el microcontrolador At Mega 328-PU para recolectar por medio de un acelerómetro, la aceleración del usuario además procesa estos datos y determina mediante un algoritmo si el usuario ha sufrido una caída, si es así, se levanta rápidamente una bandera en el software. Los datos adquiridos son transmitidos a un teléfono inteligente mediante el módulo bluetooth para notificar por SMS al cuidador del anciano. El prototipo está ubicado en el cinturón debido a que está cerca del centro de masa. Al implementar el sistema, puede reconocer cuando el usuario está realizando actividades diarias o ha caído, de acuerdo con el umbral calculado, teniendo una confiabilidad del 88%. [1]

Luis Ruiz, en Ambato, en el año 2018 el cual presenta el trabajo de titulación "Sistema de telemedicina para monitoreo continuo de constantes vitales en lactantes menores para evitar el síndrome de muerte súbita" consta de tres nodos: adquisición, procesamiento y visualización móvil. Este sistema es de uso doméstico, con capacidad de medir el pulso cardiaco y SPO2 del bebe, estos datos son enviados al microcontrolador por medio del protocolo de comunicación I2C; se utiliza un módulo bluetooth para transmitir información del lactante a la interfaz del móvil desarrollada en Android Studio, permitiendo monitorear las variables tomadas del láctate. Para medir la factibilidad del sistema, se utiliza la teoría de cálculo de errores obteniendo 99,2% con error absoluto extremadamente bajo, además es capaz de generar notificaciones vibratorias y auditivas al detectar alteraciones de los signos vitales. [2]

En el 2019, Ronny Montero en Sangolquí presenta el trabajo de titulación "Diseño e implementación de un sistema wireless sensors network personal para el monitoreo constante de signos vitales y visualización en tiempo real en aplicación web". El diseño tiene varios sensores físicos en distintas partes del cuerpo humano: en el tórax para adquirir señales electrocardiogramas, en la muñeca para monitorear fotopletismografía y finalmente en el brazo para adquirir la temperatura corporal. Todos estos sensores tienen comunicación inalámbrica bajo el protocolo 802.11 para conectar a un Gateway concentrador de datos y tener acceso a internet para almacenar y gestionar los datos en el servidor. Para repetir el proceso de toma de datos del cuerpo humano hasta enviar a la base de datos utiliza una frecuencia de muestreo de 100Hz suficiente para recolección de información. Para definir la factibilidad del sistema utiliza las ondas que se obtiene de los sensores y se compara con ondas referenciales para verificar si cumplen con las características, analizando las señales, todas son aceptables, pero no son iguales debido a la existencia de ruido, con un error promedio de 2 %. [3]

En Indonesia, en el año 2019 Tirza Bernadus, Luki Subekti y Yoanes Bandung, publicó su artículo titulado "IoT-Based fall detection and heart rate monitoring system for elderly care". En este proyecto desarrollan un prototipo portátil con capacidad para conectar a una red Wi-Fi utilizando una placa NodeMCU. Utiliza el protocolo de comunicación I2C para la transmisión de datos de los sensores. Los datos son muestreados alternativamente cada 3ms por el microcontrolador y enviados a Google Firebase que está en formato JSON. La estructura de datos consta de: 2 nodos, uno para almacenar datos y el otro para identificar al usuario. Tiene una aplicación móvil como subsistema que sirve para presentar los datos y notificar. Al realizar pruebas de funcionamiento con 70 jóvenes haciendo diversas tareas, al 90% el sistema pudo detectar caídas sin enviar ninguna notificación falsa a la aplicación, por otra parte, se comprobó que la fuente de alimentación dura alrededor de 6.5 horas. [4]

En el trabajo de Estefanía Terán titulado "Sistema de monitoreo remoto y visualización para dispositivos de análisis de signos vitales orientados a E-HEALTH" publicando en el año 2019, en Sangolquí. Este proyecto de investigación consta de partes principales de hardware como un dispositivo e-Health ubicado en el tórax, está compuesto por módulos de sensorización, conectados mediante el protocolo de comunicación I2C a la placa NodeMCU que trabaja con el estándar 802.11b/g/n. La

arquitectura está basada en IoT para trabajar mediante el protocolo MQTT mientras que para almacenamiento de los datos usa MySQL de código abierto. Toda la información es presentada a través de una página web en gráficas y tablas que solo puede ser visualizada por usuarios registrados previamente en la base de datos. Para evaluar el sistema fue indispensable distintas pruebas de funcionamiento durante un periodo de 10 min por 10 veces. Para determinar el error implementaron dispositivos médicos apropiados para cada signo vital, en base a esto el sistema obtuvo un error promedio del 3,1%. [5]

En el 2021, en Ambato Roberto Garcés realiza el trabajo de titulación "Sistema de telemedicina con monitoreo de signos vitales en IoT en un ambiente smart TV" basado en cuatro subsistemas: sensorización, conexión, administración e Interfaz para el usuario. La frecuencia cardiaca y la saturación de oxígeno son recogidos por el microcontrolador para enviar por medio de bluetooth al sistema Android TV, estos datos se transmiten por medio del protocolo TCP/IP para administrar mediante Odoo. La información almacenada es presentada a través de tablas o gráficos para tomar acciones oportunas ante anomalías detectadas del usuario. Los resultados que presentó el sistema son positivos, alcanzando respuestas exitosas del 95,73% con latencia de 3360 ms. El autor utiliza 60 segundos para medir las variables del usuario, dentro de este periodo existe latencia en la transmisión inalámbrica de 5,37 milisegundos. [6]

En Ambato, Andrés Carrillo en el año 2022 publica el trabajo de titulación "Sistema de telemedicina basado en IoT para monitoreo de pacientes con enfermedades respiratorias". Este sistema se desarrolló mediante la capa de sensorización que permite capturar constantes vitales y enviar al microcontrolador ATMEGA 328p para procesar los datos, la capa de intercambio de datos está formado por un módulo NodeMCU para tener conectividad con la "plataforma ThingSpeak" mediante el protocolo HTTP, finalmente la capa de aplicación IoT está basada en Open Source llamada ThingSpeak que ofrece una API para almacenar los datos de la capa inferior y presentar al usuario en una interfaz con parámetros establecidos. Al enviar 100 veces los datos a la nube en un intervalo de 15 segundos presenta errores relativos menores al 1%. [7]

Los antecedentes investigativos sirven como base para implementar el sistema de notificación de emergencias para que las personas encargadas del cuidado del adulto mayor sean alertadas y puedan asistir en tiempo oportuno. Las investigaciones ayudan a determinar la ubicación del dispositivo IoT, la mayor parte de investigaciones utilizan como brazalete en la muñeca ya que en esta parte se puede recopilar la frecuencia cardiaca por medio de la arteria radial, de forma similar ocurre para determinar la temperatura corporal del paciente, todo esto usando sensores no invasivos. Los accidentes que monitorean son las caídas ya que este evento tiene mayor ocurrencia con personas de la tercera edad, además puede ser monitoreada usando sensores que miden la aceleración del individuo.

1.2 Contextualización del problema

Las estadísticas del INEC revelan que en el año 2020 el 7,4% de la población total eran personas mayores de 65 años, para el año 2054 se estima que representen el 18% de la población con esperanza de vida mayor a 82,5 años [8]. En esta etapa de la vida existen mayores limitaciones al acceso de los recursos de supervivencia y aumenta la necesidad de atención de cuidado. En la actualidad el 14,6% de hogares en Ecuador se componen de un adulto mayor viviendo solo [9], aumentando la vulnerabilidad de sufrir accidentes.

Los adultos mayores tienen una disminución de las distintas funciones físicas, provocando un mayor riesgo para experimentar problemas de salud y ser dependientes de otros. La OMS calcula que alrededor del mundo anualmente se producen 684000 caídas mortales [10], mientras que, a nivel nacional el doctor Pablo Álvarez médico especialista en Geriatría-Gerontología y asesor del programa nacional del adulto mayor, afirma que es la quinta causa de muerte de las personas adultas mayores [11]. Además, un 47% de ancianos mayores de 75 años que cayeron no pudieron levantarse por sí solos, si la asistencia toma un tiempo de espera prolongado incrementa las probabilidades de hospitalización o incluso la muerte [12].

En la etapa del envejecimiento existen más vulnerabilidades a diversas enfermedades requiriendo un control constante de su estado de salud. La mayor parte de ancianos tienen enfermedades que provocan alteraciones al ritmo cardiaco, la presión arterial,

la frecuencia respiratoria y cambio de temperatura, muchas de estas deben ser monitoreadas constantemente para evitar y controlar alteraciones del adulto mayor. Para manejar este fenómeno los familiares están obligados, a enviar hacia asilos generando gastos dentro de la canasta familiar; a nivel nacional el costo de este servicio está por los 300 dólares, pero cuando el asilado no puede valerse por sus propios medios o necesita un acompañante constante el costo sube a 500 dólares [13], otra alternativa sería contratar una persona que vele por la salud del anciano en el hogar.

La forma habitual de cuidar a los ancianos no es suficiente, debido a que existen problemas como la falta de notificación de caídas, asistencia médica en tiempos prolongados y supervisión del estado de salud del adulto mayor, siendo los principales motivos para causar hospitalizaciones o incluso la muerte. En Ecuador existen fundaciones y asilos que garantizan el cuidado y bienestar de los adultos mayores, mas no cuentan con una herramienta o dispositivo tecnológico que alerte el estado y las condiciones de salud, en caso de que el anciano se encuentre en peligro.

Con el objetivo de alertar accidentes que sufre un anciano y examinar el estado de los signos vitales, es necesario crear una herramienta tecnológica que funcione en caso de que la integridad del adulto mayor se encuentre en riesgo. Por lo tanto, el sistema de notificaciones de emergencias busca cubrir estas necesidades mediante una infraestructura IoT, ya que los datos de los sensores serán enviados a un teléfono inteligente de la persona o familiar responsable del cuidado del adulto mayor para que lo monitoree y asista en el tiempo oportuno. Los principales beneficiarios serán los adultos mayores y las personas encargadas del cuidado de este sector vulnerable de la sociedad, gracias a los recursos que ofrece en tiempo real la tecnología.

La presente investigación brindará tranquilidad a los cuidadores, satisfaciendo la acción de alertar el estado de los signos vitales, mediante innovaciones tecnológicas, contando con módulos para implementar este dispositivo con conectividad a la red para interactuar desde cualquier parte. Se puede reducir gastos económicos usando solamente sensores necesarios para la adquisición de signos vitales y la estabilidad del usuario. La construcción e implementación del software es factible utilizando plataformas Open Source, además para ejecutar este proyecto se tiene amplias fuentes bibliográficas y conocimientos necesario del investigador para poder cubrir los requerimientos en el desarrollo de un dispositivo basado en IoT.

1.3 Fundamentación teórica

1.3.1 Envejecimiento

La organización mundial de la salud define el envejecimiento como la acumulación de distintos daños celulares y moleculares de acuerdo como avanza la edad del cuerpo humano, disminuyendo progresivamente las reservas fisiológicas y aumentando la vulnerabilidad para contraer enfermedades, convirtiendo al adulto mayor ser dependiente en sus actividades cotidianas. Además, la tercera edad no tiene solamente cambios en las perdidas biológicas sino también en las funciones y posiciones sociales, como consecuencia tratan de tener menos metas y actividades sociales, dejando de practicar las capacidades o funciones que adquirió en el trascurso de su vida. [14]

1.3.2 Población adultos mayores

En las últimas décadas ha incrementado la esperanza de vida en Ecuador esto se debe a distintos factores relacionados a los avances de salud e higiene sanitaria. Según las estadísticas del INEC revelan que aproximadamente existen 1 millón de mayores de 65 años, es decir 7% de habitantes. Además, existe cinco puntos de diferencia entre hombre y mujeres, esto es 47% varones y 53% mujeres, [15] teniendo siempre el número mayor de mujeres en los grupos etarios como se ilustra en la figura 1.

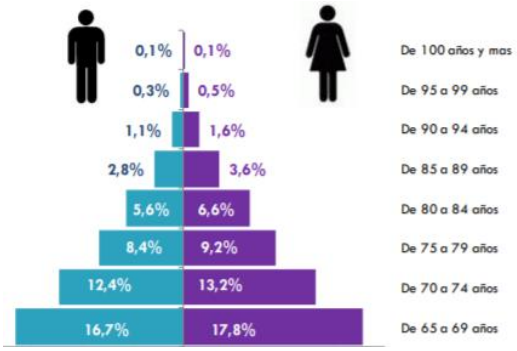


Figura 1.- Adultos mayores por grupos etarios [15]

Para el año 2032 se estima que la población joven logrará llegar a su tope y empezará el índice a decrecer hasta finales del siglo. Algo similar sucederá para los habitantes de edades en el rango de 20 a 39 y 40 a 59 años, pero esto se estima en el periodo 2052 a 2073, entonces el grupo que sigue incrementando es el de personas adultas mayores de 60 años, este grupo tiene el mayor ritmo de crecimiento y seguirá de igual forma por el resto del siglo. [16]

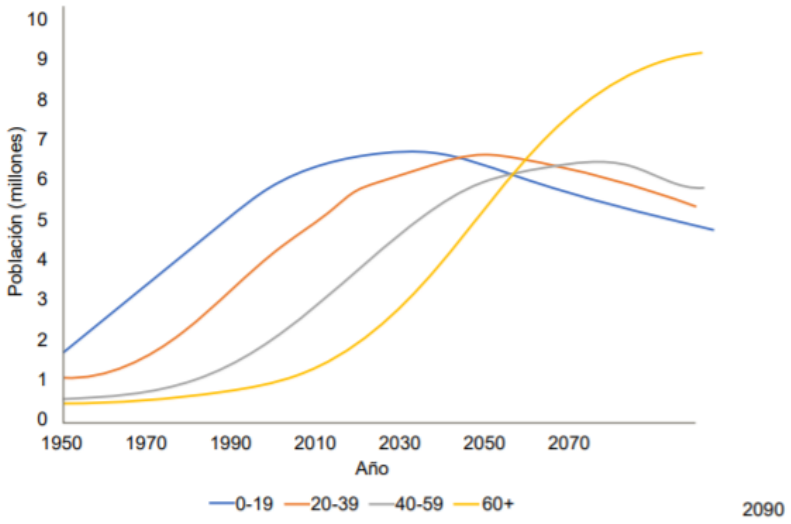


Figura 2.- Estimaciones de la población ecuatoriana en base a Naciones Unidas [16]

1.3.3 Accidentes en la tercera edad

Las causas y efectos que tiene los tipos de accidentes que sufre un individuo está relacionado estrechamente con la edad y actividades que realiza. Tiene un alto porcentaje de accidentes con mayor frecuencia dentro del domicilio, las personas mayores a 65 años por baja capacidad funcional. El tipo de accidente más frecuente en personas de la tercera edad son las caídas en el mismo nivel, este problema incrementa con la edad avanzada. [17] En un estudio de tipo descriptivo, observacional y transversal se determina los principales accidentes en esta población, con una muestra de 345 personas, de los cuales 55 participantes sufrieron algún tipo de accidente, todos los datos fueron recopilados en el periodo septiembre 2016 y febrero 2017. [18]

Tabla 1.- Tipos de accidentes en personas de la tercera edad. *[18]*

Tipos de accidentes	**Masculino**		**Femenino**		**Total**	
	No.	%	No.	%	No.	%
Caídas	12	57,1	15	44,1	27	49,1
Fracturas	3	14,3	6	17,6	8	14,5
Quemaduras	2	9,5	5	14,7	7	12,7
Electrocuciones	0	0	2	5,9	2	3,6
Heridas	8	38	4	11,8	12	21,8
Tránsito	1	4,8	0	0	1	1,8
Ingestión Accidental de medicamentos o tóxicos	2	9,5	4	11,8	6	10,9

Caídas de adulto mayor

La organización mundial de la salud define a las caídas como sucesos involuntarios que hace perder la estabilidad del cuerpo y terminar en el suelo o en alguna otra superficie que lo detenga, causando lesiones mortales, aunque la mayor parte de ellas no lo son, pero si una gran parte producen hospitalizaciones. Las caídas a nivel mundial son la segunda causa de muerte debido al traumatismo involuntario que produce, pero los que más sufren estos sucesos son los mayores de 65 años, se estima que anualmente se generan 37.3 millones de caídas, un alto índice requiere atención médica urgente. [12]

Las caídas de los adultos mayores se dan por distintas variables del mismo cuerpo humano o variables que rodean el entorno. Para identificar de mejor manera se clasifican en intrínsecas y extrínsecas. [19]

Variables intrínsecas de caídas

Las causas principales están relacionadas con el proceso de envejecimiento: [20]

- Índice de masa corporal
- Alto índice de discapacidad visual
- Avería cognitiva
- Enfermedades crónicas

Variables extrínsecas de caídas

Las causas extrínsecas están relacionadas con el entorno: [20]

- Pisos irregulares y resbaladizos
- Muebles extremadamente grandes
- Ausencia de barras de apoyo
- Zapatos inadecuados para el piso

1.3.4 Signos vitales

Los signos vitales son usados para medir las funciones básicas del cuerpo humano, que representan valores para estimar la temperatura corporal, frecuencia cardiaca, frecuencia respiratoria y presión arterial, todos estos datos permiten evaluar las condiciones de salud del paciente, detectar rápidamente modificaciones o cambios que representan algún tipo de alteración en el estado de salud. Hay que destacar que los

rangos normales de los signos vitales varían con el peso, el sexo y aun mas con la edad, [21] como se indica en la tabla 2.

Tabla 2.- Rangos normales de signos vitales [22]

Edad	**Tensión arterial (mmHg)**	**Frecuencia cardiaca**	**Frecuencia respiratoria**	**Temperatura Corporal**
3 años		90 a 100 BPM	25 RPM	36,6 a 37,8 °C
4 - 8 años		86 a 90 BPM	20 - 25 RPM	36,5 a 37 °C
8 – 15 años	10 a 14 años Sistólica: años x 2+100 Diastólica: ½ sistólica +10	80 a 86 BPM	18 – 20 RPM	36,5 a 37 °C
Adultos	120/80 +/- 10	60 a 80 BPM	16 – 20 RPM	36,5 °C – 37 °C
Tercera edad	121/91	60 a 100 BPM	14 – 16 RPM	<35 °C – 37,2°C

Elaborado por: El investigador

En la actualidad también es considerado como signo vital a la oximetría de pulso o la pulsioximetría, encargada de medir la cantidad de oxígeno en la sangre, aunque también ayuda a evaluar:

- Respiración de un individuo o el nivel de oxígeno.
- A medir la eficacia para tratamientos de enfermedades pulmonares.
- Capacidad que posee la persona para aumentar la actividad.

Tabla 3.- Rangos de saturación de oxígeno. *[23]*

Índice	**Rango**	**Estado**
SpO2	95 – 100 %	Normal
SpO2	91– 94%	Moderado
SpO2	Menor que 92%	Hipoxemia (Grave)

Elaborado por: El investigador

Se puede estimar las condiciones de salud en base al estado de las constantes vitales como se detalla en la tabla 4.

Tabla 4.- Alteración de signos vitales *[24]*

Padecimiento	Temperatura corporal	Frecuencia cardiaca	Saturación de oxigeno
Hipotermia	< 35 °C		
Fiebre	> 37,1 °C		
Taquicardia		> 100 BPM	
Broquicardia		< 60 BPM	
Hipoxemia			< 92 %

La hipotermia tiene distintas clasificaciones:

- Hipotermia accidental primaria: esta es causada por exposición directa al frío.
- Hipotermia accidenta secundaria: es causada por enfermedades graves, la tasa de mortalidad aquí es alta.
- Hipotermia terapéutica: esta se produce cuando la persona es practicada algún tipo de cirugía, sirviendo como anestésico, es voluntaria.

Para valorar el estado de un paciente de la tercera edad se utiliza de distintas técnicas, como la observación, el examen físico y también medir datos objetivos que están directamente relacionados con los signos vitales. En la tabla 5 se especifica la alteración de los signos vitales frente a una enfermedad infecciosa, pero además de presentar esas condiciones debe presentar otros síntomas como: diarrea, tos, perdida del gusto o del olfato, cansancio, ojos rojos, dolor de cabeza, entre otros. [24]

Tabla 5.- Alteración de signos vitales frente a covid-19. [*[23], [25]*]

Enfermedad	Temperatura	Frecuencia cardiaca	Saturación de oxigeno	Frecuencia respiratoria	Conducta
Covid-19	> 37,5 °C	≤ 90 BPM	95% ó >	≤ 20 RPM	Monitoreo remoto
	> 37,5 °C	91 - 130 BPM	93% - 94%	21 – 24 RPM	Hospitalización
	> 37,5 °C	≥ 131 BPM	92% ó <	≥ 25 RPM	Hospitalización

1.3.5 IoT

El termino internet de las cosas es una tecnología que conecta a los distintos objetos cotidianos a Internet para intercambiar, agregar y procesar los datos adquiridos sobre el entorno físico y presentar los valores en una interfaz. Estos sistemas también pueden reaccionar o actuar de forma adecuada y autónoma, entonces representa una nueva dimensión digital debido a que una gran cantidad de dispositivos tienen mayor capacidad de cómputo para procesar y ser independientes. [26]

Este concepto ha hecho posible el monitoreo de las condiciones de salud del paciente brindando seguridad y fácil acceso a los médicos al historial clínico, reduciendo significativamente los costes de atención médica. En la actualidad existen dispositivos portátiles conectados por alguna tecnología inalámbrica a la red que funcionan como recordatorios o seguimiento de rutinas de ejercicio físico. [27]

Arquitectura IoT

Un sistema inteligente IoT básico, que tenga mínimas aplicaciones debe tener el siguiente flujo de trabajo.

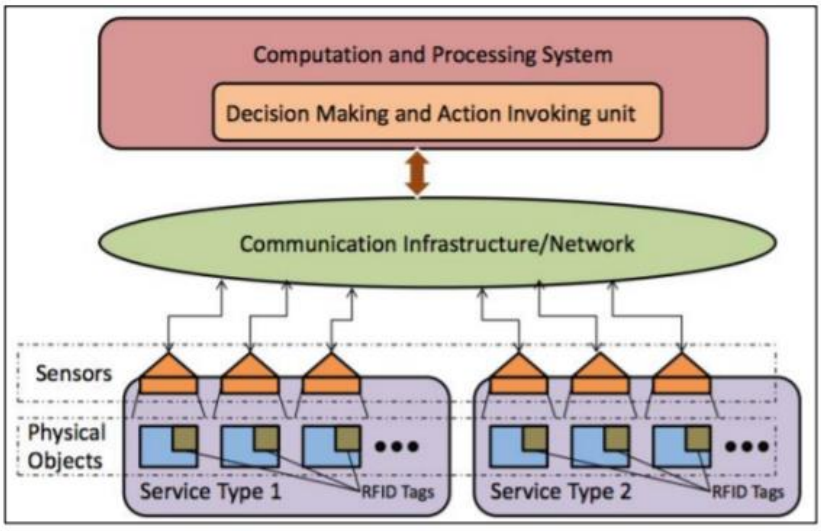


Figura 3.- Flujo de trabajo IoT [28]

El dispositivo IoT que realiza acciones, ofrece servicios inteligentes, además incluye mecanismos con el fin de facilitar datos al administrador sobre el estado actual o acciones ejecutadas, esto es capaz a la arquitectura que tiene.

Tabla 6.- Arquitectura de cuatro capas [29]

Capa de sensorización	Está compuesta por sensores, variables físicas y la adquisición de datos.
Capa de transmisión	Esta encargada de transmitir los datos por medio de la tecnología de comunicación que se utilice en el sistema.
Capa de integración de datos	Tiene el objetivo de procesar la información presentada por la tecnología de comunicación, filtrar los datos que no se desean y también integrar información para los usuarios finales y servicios.
Capa de servicio de aplicación	Presenta el contenido por medio de interfaces al usuario final.

Protocolos de IoT

HTTP

HTTP es un protocolo de comunicación para aplicación, tiene el objetivo de transferir información a la web. Consiste en el modelo Cliente-Servidor, donde el cliente debe hacer peticiones a un determinado servidor que sirve peticiones a los clientes. Todas las peticiones se hacen por medio de mensajes HTTP para enviar por texto plano, motivo por el cual los mensajes son largos, también existen otras características que el uso no sea optimo en redes de bajo consumo. [30]

Coap

Es un protocolo de aplicación, pensado para usar en sensores de baja potencia, el diseño está basado en el modelo HTTP, pero mejorado, es decir tiene requisitos como bajo overhead y multicast. Este protocolo implementa el REST permitiendo a los clientes y servidores publicar y consumir diferentes servicios, también tiene los métodos HTTP como: get, put, post y delete. [31]

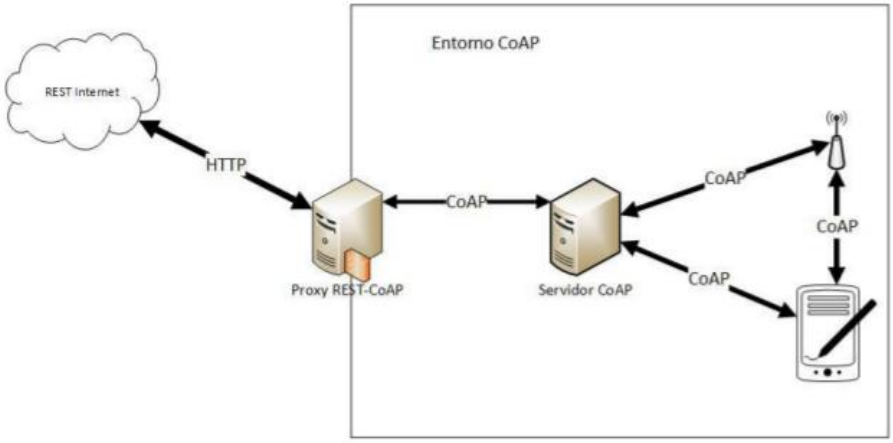


Figura 4.- Entorno Coap [30]

MQTT

Es un protocolo para conexión de dispositivos empotrados, para la parte de conexión es usado un mecanismo de routing, convirtiéndole ideal para implementar en dispositivos que requieren IoT o M2M. Usa publicación y suscripción para transmitir de forma flexible e implementación sencilla, se adapta fácilmente a equipos que usan una banda ancha baja. Fue construida en base a TCP permitiendo entregar la información por medio de tres niveles de calidad de servicio (QoS). La arquitectura de MQTT se compone de suscriptor, publicador y bróker. [31]

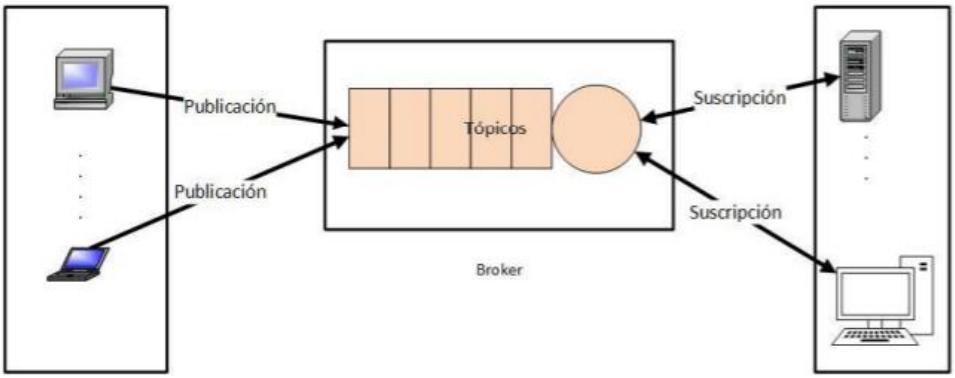


Figura 5.- Arquitectura MQTT [30]

Tabla 7.- Comparación entre los protocolos de comunicación IoT

Concepto	HTTP	MQTT	COAP
Arquitectura	Cliente/Servidor	Cliente/Broker	Cliente/Broker Cliente/Servidor
Tamaño de la cabecera	Sin límite	2 bytes	4 bytes
Métodos	-Post -Get -Put -Patch -Options -Delete -Connect -Head	-Connect -Disconnect -Publish -Suscribe -Unsubscribe -Close	-Post -Get -Put -Delete
Calidad de servicio (QoS)	Mensaje confirmable – o no confirmable	-QoS 0 -QoS 1 -QoS 2	Limitado TCP
Seguridad	TLS/SSL	TLS/SSL	DTLS, IPSec
Puerto disponible	80 (TLS)/443 (SSL)	1883 (TLS) / 8883 (SSL)	5683 (UDP) / 5684 (DLTS)
Licencia	Free	Open Source	Open Source
Formato codificación	Texto	Binario	Binario
Protocolo de transporte	TCP	-TCP -MQTT-SN puede usar UDP	-UDP -SCTP
Estándares	-IETF -W3C	-OASIS -Eclipse fundations	-IETF -Eclipse fundations
Implementación	Fácil de implementar	Fácil de implementar	Complejo, pocas librerías y soporte

Elaborado por: El investigador

1.3.6 Tecnologías Inalámbricas

La tecnología avanza a pasos agigantados gracias a ello, sigue el incremento de tecnologías inalámbricas, estas son usadas a nivel mundial en diversas aplicaciones, aunque las más utilizadas son Wi-Fi, Zigbee y Zwave, pero en los últimos años se introdujo Lora teniendo una gran aceptación en el mercado por el alcance y la capacidad de transmisión que posee. Para aplicar adecuadamente el tipo de tecnología se debe tomar en cuenta la seguridad, velocidad de transmisión, alcance máximo, entre otros factores. [32]

Tabla 8.- Comparación entre varias tecnologías inalámbricas para IoT [33]

Funciones	**Bluetooth**	**Wi-Fi**	**Z-WAVE**	**ZIGBEE**	**LORA**
Topología lógica	Malla	Malla	Malla	Malla	Malla
Topología física	Bus AC HOC	Bus AC HOC	Bus	Bus	BUS
Forma de transmisión	Inalámbrica	Inalámbrica	Inalámbrica	Inalámbrica	Inalámbrica
Estándar	IEEE 802.15.1	IEEE 802.11	Z-Wave Alliance ZAD12837	IEEE 802.15.4	LoRaWAN
Rango de transmisión	10 metros	30 a 50 metros	30 metros	10 a 100 metros	1 a 2 kilómetros
Frecuencia	2,4GHz (ISM)	2,4GHz y 5GHz	900MHz (Banda ISM)	2.4 GHz	Varias
Número máximo de dispositivos	8	250	232	25	PtP
Interferencia WLAN	Si	No	No	Si	Baja

1.3.7 Espressif

Es una empresa mundial creada en el año 2008, líder en chips y módulos de comunicación bluetooth y WiFi, diseñado específicamente para Internet de las cosas (IoT). Los chips esp32 y esp8266 son los más usados por las empresas pioneras en IoT, en la tabla 9 se detalla las características más relevantes.

Tabla 9.- Características de esp32 y esp8266. [*[34]*, *[35]*]

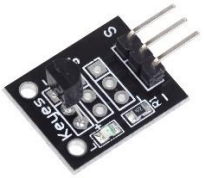
Ítems	Esp32 devkit	Esp8266 devkit
Numero de núcleos	2	1
SRAM	520 kB	50kB
ROM	448 kB	No posee
SPI flash	16 MB	16 MB
Rango de frecuencia Wifi	2.4GHz – 2.5GHz	2.4GHz – 2.5GHz
Temperatura de operación	-40°C a 105°C	-40°C a 85°C
Protocolos de red	MQTT, COAP, IPv4, TCP/UDP/HTTP/FTP	MQTT, COAP, IPv4, TCP/UDP/HTTP/FTP
Bluetooth	V4.2 BR/EDR y LE x	No posee
Interfaces	HSPI, PWM, IR, ADC, I2C, UART, I2S	HSPI, PWM, IR, ADC, I2C, UART, I2S
Precio	15$	7$

1.3.8 Sensorización

Sensor de temperatura corporal

En el mercado existen diferentes tipos de sensores de temperatura corporal como los que necesitan estar en contacto y los infrarrojos (IR), estos son capaces de determinar la temperatura sin contacto con el cuerpo humano, hechos para aplicaciones médicas. Son usados para medir la temperatura de la frente, temperatura de la piel o temperatura del oído. En el mercado existen diferentes sensores de temperatura, como se especifica en la tabla 10.

Tabla 10.- Sensores de temperatura

	MLX90614	D6T-1A-02	DS18B20
Imagen			
Infrarrojo	Si	Si	No
Rango de temperatura	-40 °C a 125 °C	-40 °C a 80 °C	-55 °C a +125 °C
Aplicación médica	Si	Si	No
Alimentación	2.4 V a 3.6 V	4.5 V a 5.5 V	3.0 V a 5 V
Corriente de suministro	1mA – 1.5mA	18.5mA	5mA
Interfaces	-I2C -SPI	-I2C -SPI	-Analógica
Contacto	Tiene infrarrojo	Dispone de infrarrojo	Debe estar en contacto con el objeto
Precisión	± 0.02 °C	± 0.4 °C	± 1 °C
Precio	$15	$20	$5

Elaborado por: El investigador

Sensor de frecuencia cardiaca y oximetría

Los sensores de frecuencia cardiaca operan a bajo voltaje entre 3v a 5v, la mayoría de los sensores trabajan por medio de longitudes de onda de un led y un infrarrojo para medir la absorción de la sangre pulsante por medio de un fotodetector. En la tabla 11 se especifica algunas características entre sensores.

Tabla 11.- Sensores de frecuencia cardiaca y oximetría

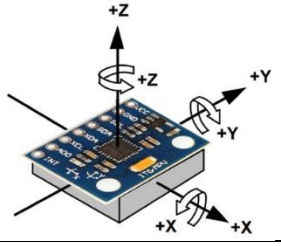
	SEN-11574	**MAX30100**	**MAX30101**
Imagen			
Alimentación	3V a 5.5V	3V a 5V	3V a 5V
Corriente	2.5mA	600µA – 1200µA	600µA – 1100µA
Rango de temperatura de operación	-25 °C a 70 °C	-40 °C a 85 °C	-40 °C a 85 °C
Incorpora oxímetro	No	Si	Si
Longitud de onda	No dispone	Red: 660 nm IR: 880 nm	Red: 660 nm IR: 880 nm Green: 527 nm
Partes del cuerpo para incorporar	-Dedo	-Dedo -Muñeca -Otras	-Dedo -Muñeca
Disponible en el mercado	Si	Si	Solo en países desarrollados
Precio	$20	$9	$25

Elaborado por: El investigador

Acelerómetro

Es un dispositivo electrónico capas de medir las fuerzas de aceleración o vibración que actúan sobre un objeto. Existen diferentes tipos de acelerómetros con diferente funcionalidad como son los piezoeléctricos, piezorresistivos y capacitivos.

Tabla 12.- Comparación entre acelerómetros

	MPU6050	MMA7361	ADXL335
Imagen			
Tipo de acelerómetro	Capacitivo	Capacitivo	Capacitivo
Sensibilidad	±2g, ±4g, ±8g, ±16g	±3g, ±6g,	±3
Interfaces	I2C, SPI	Analógica	Analógica
Librearías	-Adafruit_MPU -MPU6050	No disponible	No disponible
Alimentación	3.3V	3.3V	3.3V
Aceleración máxima	10000g		
Consumo de corriente	500µA	700µA	650µA
Dimensiones	4x4x0.9 mm	3x5x1 mm	4x4x1.45 mm
Precio	$5	$10	$8

Elaborado por: El investigador

1.3.9 Inteligencia Artificial

Se refiere a cualquier inteligencia similar a la humana exhibida por una computadora, robot u otra máquina. En el uso popular, la Inteligencia Artificial es la capacidad de una computadora o máquina para imitar la mente humana. Lo que diferencia a la Inteligencia Artificial de otros programas de ordenador es que no hay que programarla específicamente para cada escenario. Podemos enseñarle cosas (Machine Learning), pero también puede aprender por sí mismo (Deep Learning). [36]

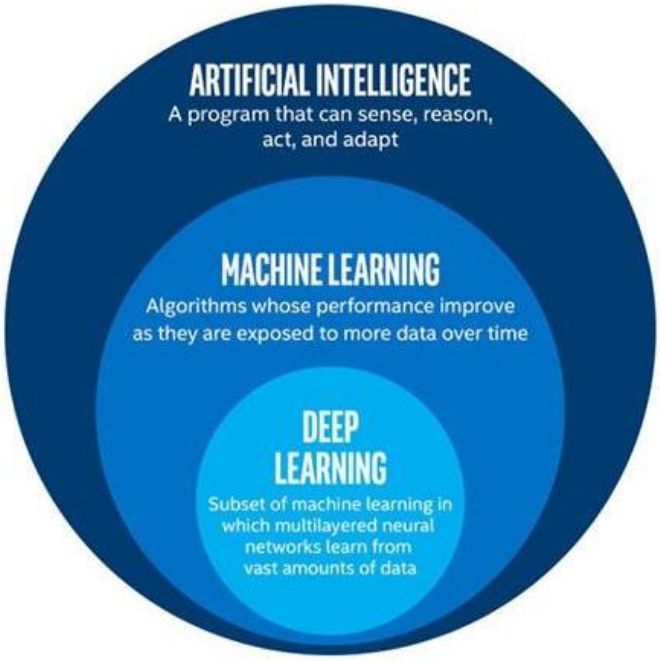


Figura 6.- Inteligencia Artificial y sus escenarios *[36]*

1.3.10 Machine Learning

El informático Arthur Samuel define machine learning como un subcampo de la informática, que se centra en la resolución de un determinado problema práctico en base a la recopilación de un gran conjunto de datos para construir un modelo estadístico y detectar patrones, sin tener la necesidad de crear un código (commands) en un lenguaje de programación. Los modelos no requieren comandos de entrada directamente para realizar una tarea especifica, solamente necesitan ser alimentados por datos de aprendizaje a una determinada frecuencia. [37]

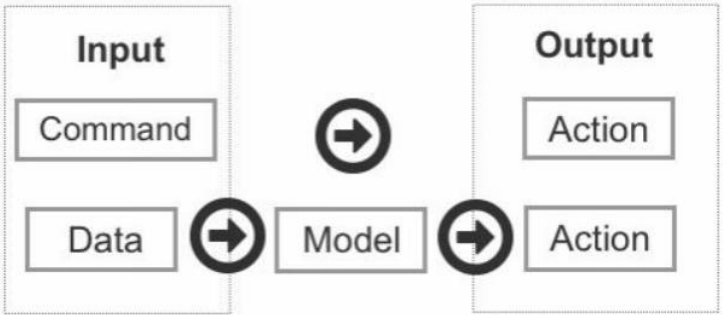


Figura 7.- Input command vs. input data *[37]*

Tipos de Machine Learning

Aprendizaje Supervisado

Este tipo de aprendizaje se aplica cuando la muestra o un conjunto de datos de entrada están asociados con una etiqueta por ejemplo (a1.bpm, “perro”), (a2.bpm, “pájaro”), (a3.bpm, “gato”), entonces a partir de este conjunto de datos el aprendizaje supervisado puede ser de: [36]

- Clasificación
- Regresión

Aprendizaje No Supervisado

El aprendizaje no supervisado usa datos no etiquetados, las aplicaciones que usa este aprendizaje son:

- Clustering (agrupamiento)
- Reducción de dimensiones

Aprendizaje Semi-Supervisado

El aprendizaje semi supervisado trabaja con dos conjuntos de datos donde un conjunto tiene datos con etiqueta y el otro conjunto no contiene etiqueta en los datos.

Aprendizaje por Refuerzo

Es un área innovadora debido a que está inspirada en mecanismos naturales, el algoritmo de aprendizaje recepta datos de un entorno simulado o de un entorno real. Si el modelo recibe una acción es penalizada o recompensada como ocurre con los seres vivos, es decir aprende siguiendo los principios de la evolución natural. [36]

1.3.11 Machine Learning Integrado (TinyML)

El avance de diseño de dispositivos con ello, la arquitectura de los microcontroladores a hecho posible ejecutar machine learning en los microcontroladores incluso en los más pequeños con capacidades muy reducidas, a este concepto se le conoce como TinyML, para lograr esto se utiliza herramientas como TensorFlow Lite creando modelos inteligentes y ligeros con alta capacidad de reaccionar a una acción determinada en el modelo. [38]

Tabla 13.- Aspectos positivos y negativos de TinyML

Ventajas	Desventajas
Los modelos pueden ser inherentemente más confiables en comparación con dispositivos que dependen de una conexión.	Los modelos pueden admitir el procesamiento por falta de recursos del microcontrolador
Al ejecutar datos en el propio microcontrolador, se evitará gastos de la transmisión de los datos y el procesamiento en la nube.	Al implementar un sistema controlado por modelos para el procesamiento de imágenes la memoria del microcontrolador es pequeña.
La latencia no es un problema, los modelos en el dispositivo responden en tiempo real a las entradas de los datos.	El modelo pude producir aproximaciones o estimaciones, no respuestas exactas.

Elaborado por: El investigador

Entre las principales plataformas de uso esta Edge Impulse y NanoEdge como se detalla en la tabla 14, sin embargo, la compatibilidad con la mayoría de firmware tiene Edge Impulse ya que está disponible para C, C++, WebAssembly o incluso permite crear un firmware personalizado, siendo esta una de las principales desventajas de NanoEdge, además Edge Impulse también permite: [39]

- Fácil recopilación de los datos de sensores en tiempo real
- Procesamiento de señales
- Gestor de redes neuronales
- Pruebas con el sensor conectado a la plataforma
- Implementación en cualquier dispositivo

Tabla 14.- Comparación entre plataformas de TinyML

	Edge Impulse	NanoEdge
Logo	EDGE IMPULSE	NANOEDGE AI STUDIO
Interfaz gráfica	Si	Si
Documentación	Alta	Media
Compatible con Arduino IDE	Si	Si
Compatibilidad	Con la mayoría de firmware	Arduino
Uso de RAM	1,8 KB	1 KB
Uso de memoria FLASH	14,6 KB	20 KB
Dificultad de uso	Fácil	Compleja
Necesita acceso a internet	No	No
Procesamiento de señal	Posee bloques de procesamiento para cada aplicación.	Tiene procesamiento de señales específicas.

Elaborado por: El investigador

1.3.12 Edge Impulse

Es una plataforma para el desarrollo de machine learning integrado (TinyML), brinda una mejor experiencia para el desarrollo de ML en dispositivos integrados con el uso de sensores de diferentes magnitudes, visión artificial y audio a gran escala. Esta plataforma permite realizar aplicaciones interactivas en MCU de bajo consumo, con tan solo unos pocos minutos de datos de entrenamiento se podría detectar o clasificar patrones que se establecieron en el sistema. [40]

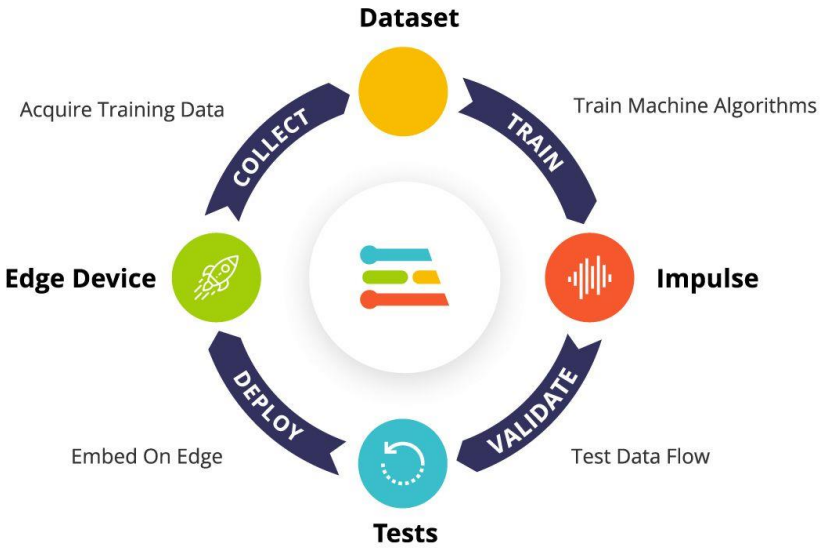


Figura 8.- Proceso Edge Impulse *[40]*

Bloques de procesamiento

Los bloques de procesamiento son los encargados de extraer todas las características significativas de los datos entrantes, es decir son esenciales para construir sistemas de Machine Learning confiables y muy pequeños. [41] La plataforma Edge Impulse posee una amplia serie de bloques de procesamiento de datos como se detalla en la tabla 15.

Tabla 15.- Comparación de bloques de procesamiento de Edge Impulse

Bloque de procesamiento	Descripción	Parámetros	Aplicaciones
Flatten	Hace un análisis basado en estadística a cada ventana, calculando entre 1 y 7 características para cada eje, en base a los métodos seleccionados.	Método de cada ventana: -Promedio -Mínimo -Máximo -Raíz cuadrada media -Desviación estándar -Sesgo -Curtosis	Excelente para promedios de datos con frecuencia de variaciones baja, como la temperatura.

Spectral Analysis	Este tipo de bloque extrae características de potencia y frecuencia de una señal, además permite aplicar filtros como paso alto o bajo para eliminar frecuencias no deseadas.	Crea dos tipos de características por cada eje/canal: -Estadísticas: RMS, oblicuidad y curtosis. -Espectrales: Valores máximos de fotogramas FFT en cada contenedor que no se filtró.	Es potente analizando patrones repetitivos de la señal, como vibraciones o movimientos continuos de un acelerómetro.
Spectrogram	Este bloque extrae características como la frecuencia y el tiempo de una señal. Fragmenta una ventana en varios marcos superpuestos, pero dependen de la longitud del frame y paso de frame.	Normalización: -Ruido (dB): aplica un filtro para eliminar la señal por debajo.	Trabaja muy bien con datos de audio pero que no use reconocimiento de voz, o datos de sensores, pero con frecuencias continuas.
IMU (Syntiant)	El bloque IMU Syntiant hace un escalamiento de datos sin procesar a valores de 8 bits con el fin de cumplir con los requisitos de los procesadores de reconocimiento de voz.	Escala datos de un tamaño de 16 bits a 8 bits en rangos establecidos de –1 a 1.	Útil para escalar bits de datos sin procesar.
Raw data	Forma ventanas a partir de muestras de datos sin procesar, se utiliza en señales procesadas con el fin	Ejes de escala: se usa para normalizar los datos entre 0 y 1	Implementada en aprendizaje profundo para aprender funciones.

	de ingresar datos en una red neuronal.		
Imagen	Este bloque trabaja específicamente con imágenes, ya que realiza la normalización para convertir el canal de cada píxel de una imagen a valores flotantes en el rango de 0 a 1.	Profundidad de color: trabaja en RGB o escala de grises.	Útil para aplicaciones de visión artificial
Audio (MFCC)	El bloque de procesamiento de audio MFCC extrae coeficientes de una señal auditiva, por lo que usa una escala no lineal Mel-Scale, adecuada para reconocimiento de voz humana o robótica.	Coeficientes ceptral de frecuencia Mel: -Número de coeficientes. -Duración de frame -Paso de fotograma -Número de filtro -Longitud de FFT -Tamaño de la ventana.	Trabaja con señales auditivas que necesiten reconocimiento de voz.

Elaborado por: El investigador

Bloques de aprendizaje

Cuando se ha extraído características significativas de una señal mediante bloques de procesamiento, es necesario implementar bloques de aprendizaje para entrenar el modelo en base a los datos de entrenamiento con algoritmos de aprendizaje que sean compatibles con los datos. [42]

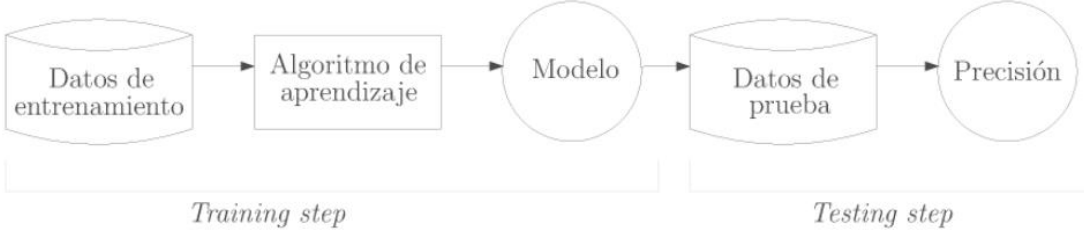


Figura 9.- Proceso de creación y validación del modelo *[42]*

Keras

Keras es una API de alto nivel de machine learning o deep learning, creada por Google para facilitar la implementación en redes neuronales. Es de código abierto y trabaja sobre bibliotecas como TensorFlow, CNTK, teano, tartanML entre otras. Con más frecuencia se encuentra en TensorFlow, biblioteca con matemática simbólica que es usada para modelos de aprendizaje profundo o crear redes neuronales. [43]

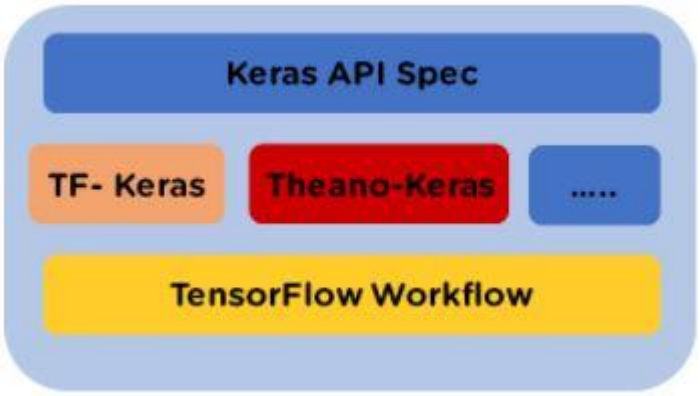


Figura 10.- Backed de Keras. *[43]*

Las aplicaciones más comunes de Keras se describen a continuación:

- Se utiliza en teléfonos inteligentes para crear nuevos modelos de aprendizaje profundo.
- En entrenamiento distribuido de los diferentes modelos de aprendizaje profundo.
- Ideal para redes de convolución y recurrentes

Edge Impulse tiene a su disposición diferentes bloques de aprendizaje como se detalla a continuación:

Clasificación Keras

La clasificación se basa en una red neuronal que toma los datos de la entrada y crea una puntuación de probabilidad, con el fin de indicar la probabilidad de que pertenecen a una de las clases especificadas. La función de la clasificación puede escribirse como:

$$c: X \rightarrow C$$

Donde c es la función de clasificación, X el conjunto de atributos que conforman una instancia y finalmente C es la etiqueta de una clase de dicha instancia. [42]

La figura 11 es un claro ejemplo de clasificación, debido a que los círculos y cruces representan a elementos de dos grandes clases, se trata de dividir el espaciado tal que separe a la mayoría de los elementos posibles.

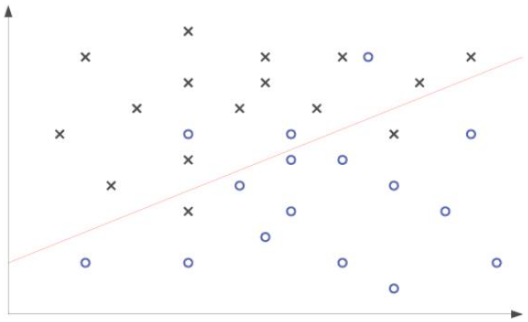


Figura 11.- Ejemplo de clasificación *[42]*

A propósito, la red neuronal consta de diferentes capas, cada una de ellas está conformada por diversas neuronas, que están interconectadas entre sí, es decir las neuronas de la primera capa se conectan con las neuronas de la segunda capa y estas se conectarán con la tercera capa y así sucesivamente con el número de capas que posee. La arquitectura de una red neuronal toma como entradas a las características extraídas de los bloques de procesamiento para pasar las características a cada capa de la arquitectura. [44]

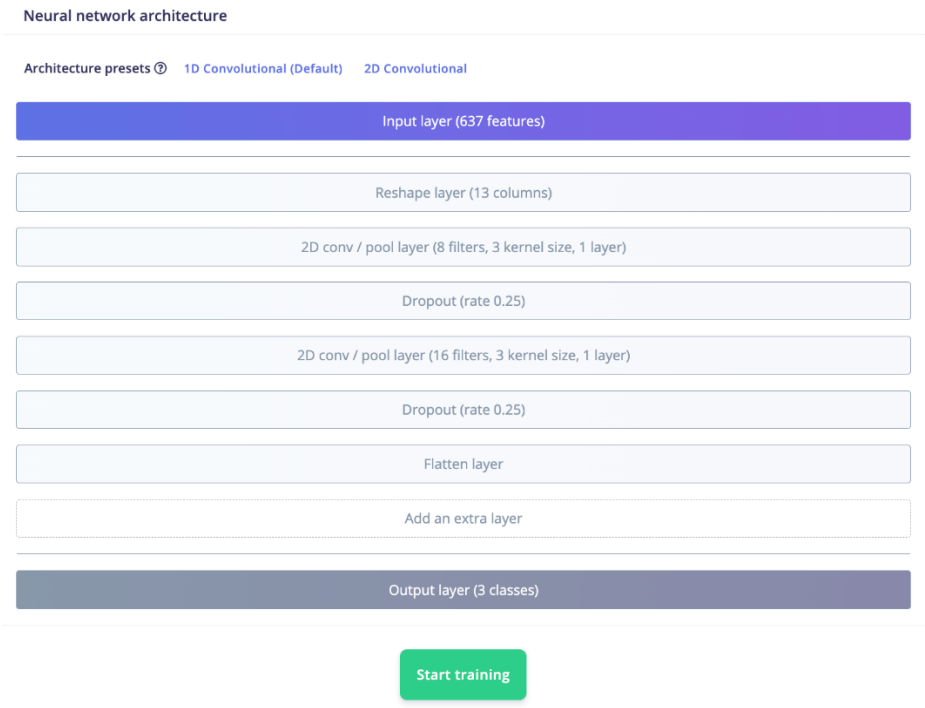


Figura 12.- Arquitectura de la red neuronal *[44]*

Para la configuración de una red neuronal presenta diferentes parámetros, como se detalla en la figura 13.

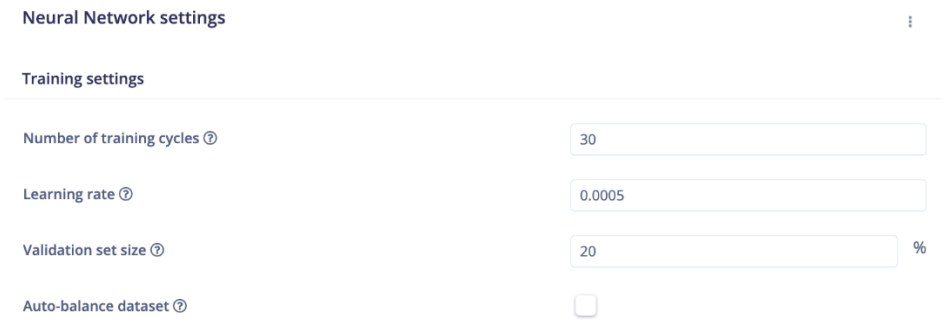


Figura 13.- Parámetros de configuración de la red neuronal *[44]*

A continuación, se realiza una descripción de cada parámetro:

- Number of training: es el número de veces que el algoritmo de entrenamiento hace una pasada completa por todos los datos de entrenamiento (épocas).
- Learning rate: la tasa de aprendizaje refleja que tan rápido aprende la red neuronal, en caso de que la red se adapte rápidamente se reduce este parámetro.
- Validation set size: es el porcentaje del conjunto de entrenamiento reservado para validación del modelo.
- Auto-balance dataset: al seleccionar mezcla los datos que no son muy comunes, esto ayuda a que el modelo sea más robusto en caso de tener pocos datos en una clase.

Cantidad de entrenamiento (épocas)

Para entrenar una red es necesario que los datos pasen por un número determinado de épocas, pero no exagerar en épocas muy elevadas ya que produce el sobre entrenamiento a la red, esto es un problema importante en las redes neuronales, para entrenar de manera eficiente se necesita una forma para determinar cuándo se está sobre entrenando. Afortunadamente existen algunas técnicas para reducir el sobre entrenamiento, más conocidas como técnicas de regulación. En la tabla 16 se detalla algunas técnicas de las más empleadas que ofrecen mejores resultados.

Tabla 16.- Técnicas para determinar el sobre entrenamiento.

Técnica	Descripción
Regularización L2	Esta técnica tiene como objetivo modificar los pesos en la red de forma gradual, es decir puede ser vista como una forma de equilibrio entre minimizar la función de coste y encontrar pesos pequeños.
Early Stopping	Consiste en parar el proceso de entrenamiento justo cuando el error en el conjunto de validación posee un valor mínimo, se puede ver como una técnica de prueba y error ya que en el proceso de entrenamiento se debe ir testeando al modelo con datos de validación en un determinado tiempo.
Dropout	Ofrece resultados muy destacados, esta técnica consiste en cada etapa de entramiento asignar una probabilidad (p) cada neurona, cuyo resultado será que en cada etapa de entrenamiento es modificada la arquitectura de la red. El valor de p (dropout rate) normalmente usado es p=0.5

Elaborado por: El investigador

Tasa de aprendizaje

La velocidad de aprendizaje (η) es un factor elemental para el proceso de aprendizaje de las redes neuronales. Según Bosch Rué y colaboradores recomiendan la experimentación para determinar este parámetro, también mencionan que valores muy altos pude convertirse en una red inestable y valores mínimos puede mejorar la red, [42] esto se demuestra en la figura 14.

Tabla 17.- Experimentación de la velocidad de aprendizaje para determinar las más optima.

Valor de η	Descripción
$\eta = 10$	El erro produce considerables oscilaciones desde que inicia el proceso de entrenamiento y se mantiene hasta el final
$\eta = 1$	Reduce el error en las primeras etapas, pero luego se mantiene con picos altos y bajos
$\eta = 0.1$	El error reduce de forma progresiva y suave durante todo el periodo de entrenamiento

Elaborado por: El investigador

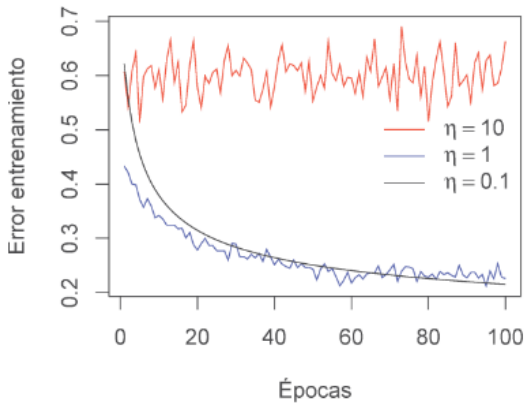


Figura 14.- Error de entrenamiento con diferentes valores η *[42]*

Regresión Keras

Hace referencia a una técnica estadística usada para predecir o estimar una variable cuantitativa en función de otra variable cuantitativa, es decir el resultado esperado será un valor numérico. La regresión tiene la tarea de asignar valores numéricos a instancias de un dominio dado, definidos por un conjunto de valores continuos como en la figura 15.

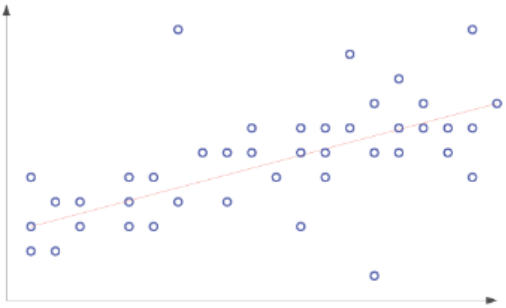


Figura 15.- regresión lineal *[42]*

Para entrenar el boque de regresión en Edge Impulse se utiliza de algún tipo de bloque de procesamiento. Es necesario configurar todos los parámetros del bloque de regresión, aunque estos son idénticos a los de clasificación como se detalló en la figura 13.

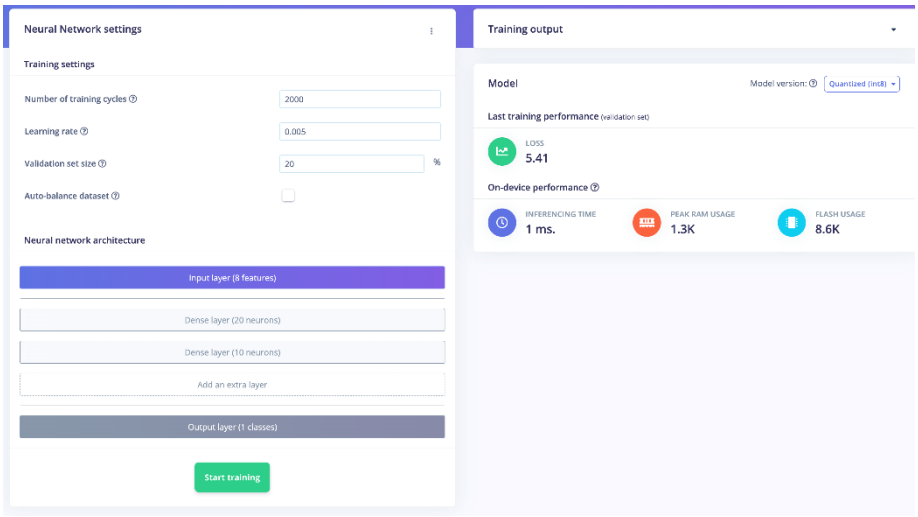


Figura 16.- Bloque de regresión

Transferencia de aprendizaje

Este bloque de aprendizaje se basa en el proceso de recopilar características aprendidas de un determinado problema y utilizar en un problema nuevo pero relacionado. Se usa con mayor frecuencia en conjunto de datos con una gran escala de objetos comunes convirtiendo en más ágil y preciso de ajustar y adaptar a tareas nuevas. El bloque de transferencia de aprendizaje se puede encontrar para:

- Imágenes
- Detección de palabras claves

Edge impulse posee otros bloques de aprendizaje, pero para datos más específicos como para reconocimiento de objetos, en la tabla 18 se detallan las características más relevantes de los bloques de aprendizaje.

Tabla 18.- Tabla comparativa de bloques de aprendizaje

Bloque de aprendizaje	**Enfoque**	**Bloques de procesamiento compatible**	**Aplicaciones**
Clasificación Keras	Permite clasificar en grandes grupos, basado en técnicas de estadística, por ejemplo ¿Mañana va a llover? [Si, No] la respuesta dependerá de los datos de entrada.	-Raw Data -Flatten -Spectral analysis -Spectrogram -Audio -Imagen	-Reconocimiento de movimiento continuo -Fusión de sensores -Reconocer sonidos de audio -Responder a la voz humana.
Regresión Keras	Permite estimar un valor numérico en base a los datos de entrada. Por ejemplo ¿Cuál será el salario de un empleado en función de la edad? El salario dependerá de cuanta experiencia tenga.	-Raw Data -Flatten -Spectral analysis -Spectrogram -Audio -Imagen	-Útil para predecir el futuro en base a modelos. -Estima valores numéricos
Detección de anomalías	Permite manejar datos que nunca se entrenaron a la red neuronal, como un gesto nuevo.	Trabaja bien en cualquier bloque de procesamiento.	-Detecta anomalías en base a k-medias
Transferencia de aprendizaje de imágenes	Resuelve el problema de clasificación de imágenes cuando se trabaja con conjunto de datos pequeños.	-Imagen	-Clasificación de imágenes

Transferencia de aprendizaje de detección de palabras claves	Esta función aprende de un gran conjunto de datos de audio, permite ajustar un modelo de detección de palabras claves.	-Audio	-Detección de palabras claves en audio.
Detección de objetos (imágenes)	La detección de objetos toma una imagen y crea información relevante sobre la clase, la cantidad de objetos y la posición de la imagen.	-Imagen	-Ideal para detección de objetos en diferentes posiciones.

Elaborado por: El investigador

1.3.13 Docker

Docker es un software de contenerización abierto que permite desarrollar y ejecutar aplicaciones, con el fin de separar las aplicaciones de la infraestructura. Es posible administrar la infraestructura de la forma similar que son administrada las aplicaciones. Trabaja en entornos estandarizados, en base a contenedores locales que contienen a las aplicaciones y/o servicios. [45]

Arquitectura

La arquitectura que usa Docker es cliente -servidor, donde el cliente realiza una comunicación con el Daemon de Docker, encargado de realizar el trabajo duro de crear, ejecutar y distribuir los diferentes contenedores mientras que el cliente Docker y el Daemon hacen una comunicación a través de una API REST, por medio de interfaz de red o sockets UNIX. [45]

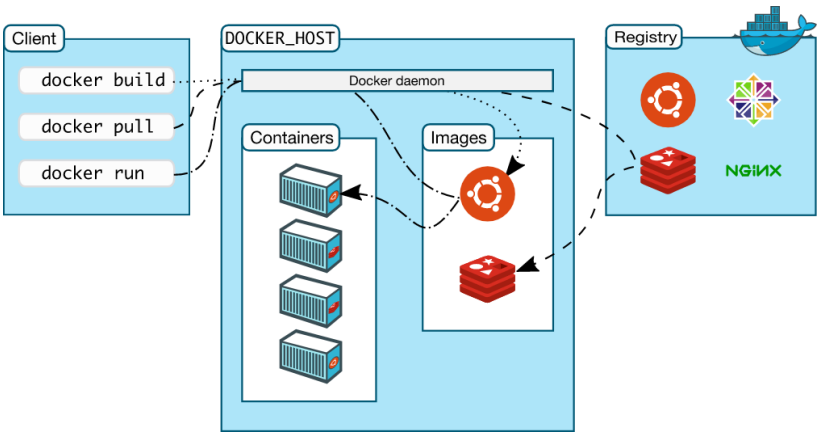


Figura 17.- arquitectura Docker *[45]*

EMQX

EMQX es un bróker open-source MQTT. Tiene un motor de procesamiento en tiempo real de mensaje que trabaja a un rendimiento alto, ayuda a la transmisión de datos para dispositivos IoT a cualquier escala. La principal función que usa el bróker EMQX es reenviar y publicar mensajes a los diferentes suscriptores de cada nodo, como se muestra en la figura 18.

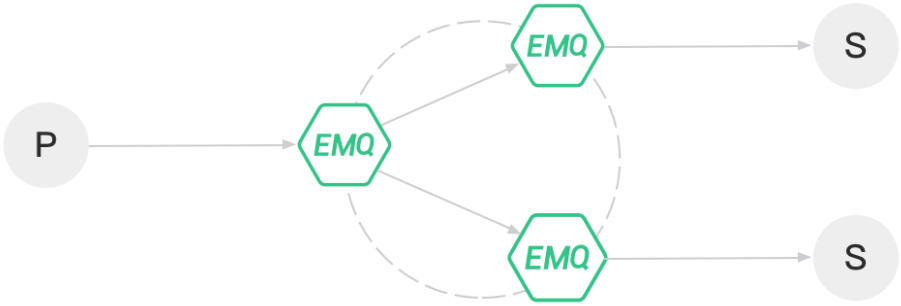


Figura 18.- Diseño del clúster distribuido

El bróker EMQX tiene beneficios entre los cuales se puede destacar:

Tabla 19.- Ventajas del bróker EMQX *[46]*

Ventaja	Descripción
Alto rendimiento	Procesa y mueve millones de mensajes por cada segundo en un único bróker.
Alta disponibilidad	Se puede logra alta escalabilidad y disponibilidad por medio de una arquitectura sin maestro.
Escala masiva	Es posible escalar a 100 millones de conexiones MQTT usando solamente un solo clúster
Baja latencia	Solamente usa sub-milisegundos en la entrega de los datos
MQTT 5.0	Tiene una compatibilidad del 100 % con el estándar de MQTT 5.0, 3.1 y 3.1.1 con el fin de mejorar confiabilidad, escalabilidad y seguridad.
Nativo de la nube y k8	Tiene la ventaja de instalar en nubes públicas que usan operator, terraform o kubernetes.

MongoDB

MongoDB es una base de datos de documentos, está diseñada para proporcionar escalabilidad y desarrollo de aplicaciones. Trabaja por medio de registros, que se basa en una estructura compuesta de datos por pares de campo y valor. Cada documento de MongoDB es idéntico a los objetos del formato JSON, los valores de cada parámetro pueden adicionar otros documentos y matrices. Algunas características claves son:

- Alto rendimiento: este tipo de base de datos ofrece persistencia de datos de rendimiento alto.
- Alta disponibilidad: dispone de un conjunto de réplicas que proporciona redundancia de datos y conmutación por error automática.
- API de consulta: permite operaciones de escritura y de lectura (CRUD)
- Escalabilidad horizontal: ofrece escalabilidad horizontal para trabajar por medio de sharding y clave de fragmentos.
- Compatible con varios motores de almacenamiento: almacenamiento en memoria y almacenamiento WiredTiger.

Tabla 20.- Comparación con diferentes bases de datos *[47]*

Concepto	Firebase	MongoDB	MySQL
Modelo principal	Basado en documentos	Basado en documentos	Basado en SGBD relacional
Modelo secundario		Spatial DBMS Motor de búsqueda serie	Spatial DBMS Almacén de documentos
Desarrollador	Google	Inc y MongoDB	Oracle
Licencia	Comercial	Open source	Open source
Disponibilidad en la nube	Si	No	No
Lenguajes de programación	- Objective-C -Java -JavaScript	-Java -JavaScript -C, C# y C++ -Dart -PHP -Python -Swift -Scala -Rust -Smalltalk -Otros	- Objective-C -Java -JavaScript -C, C# y C++ -Dart -PHP -Python -Ruby -Otros
Triggers	Callback se activa cuando cambia los datos	Si	Si
Esquema de datos	No posee	No posee	Si
Sistema operativo de servidor	Alojado	-Windows -Linux -Solaris -OS X	-Windows -Linux -Solaris -OS X -FreeBSC
Compatible con Docker	No	Si	Si

Node.js

Es un entorno de servidor de código abierto, que ejecuta del lado del servidor, está basada en el motor de JavaScript de Google Chrome V8. Node.js usa un modelo de E/S sin bloqueo alguno y basado en eventos convirtiendo en eficiente y liviano, adecuado para ejecutar aplicaciones en tiempo real. [48]

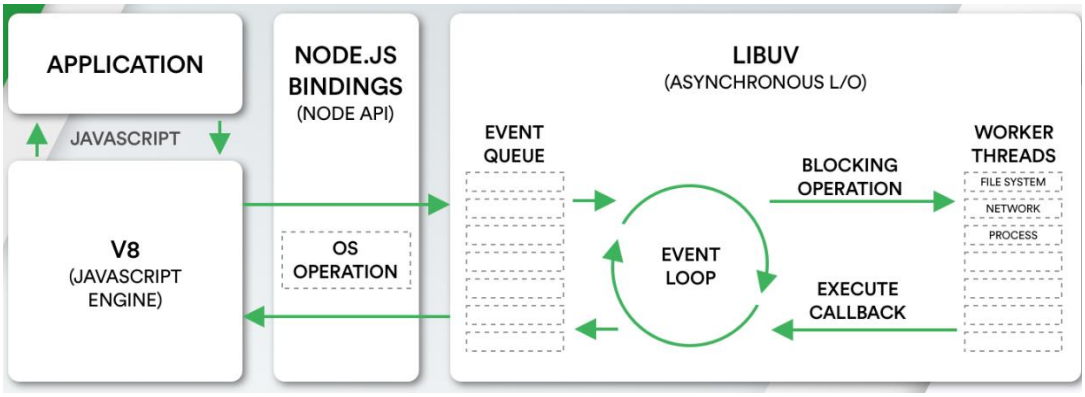


Figura 19.- Arquitectura de Node.js

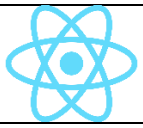
Algunas características relevantes de Node.js son:

- Es asíncrono: todas las bibliotecas son asíncronas, básicamente no usan un bloqueo en las API. Nunca espera a que una API regrese datos, sino que pasa a la otra API notificando a un servidor de eventos en Node.js
- Velocidad: la ejecución del código es muy rápida, debido a que está trabajando sobre el motor de JavaScript V8 de Google Chrome.
- No tiene almacenamiento en búfer: las aplicaciones no guardan los datos en el búfer simplemente generan todos los datos en fragmentos.
- Escalabilidad: al disponer de un entorno de eventos ayuda a ejecutar el código sin bloque convirtiéndole altamente escalable.
- Licencia: trabaja bajo la licencia MIT.

1.3.14 Desarrollo de aplicaciones multiplataformas

En la actualidad, los smartphones se han convertido de uso esencial. En nuestro entorno existe millones de personas que posee un teléfono inteligente con capacidades enormes, provocando un aumento a gran escala al mercado de desarrollo de aplicaciones. Cada entorno de programación mejora las características para convertir en un desarrollador de aplicaciones multiplataformas, es decir que permita ejecutar para Android e IOS la misma aplicación, [49] a continuación, se detalla los desarrolladores de aplicaciones más comunes en la actualidad:

Tabla 21.- Comparación entre frameworks para desarrollo de aplicaciones móviles.

Parámetro	Flutter	React Native	Klotin
Logo			
Creador	Google	Facebook	JetBrains
Open Source	Si	Si	Si
Lenguaje de programación	Dart	JavaScript	-Java -JavaScript -Native
Hot reload	Soporta	Soporta	No soporta
Aplicaciones que usan	-Google Ads -Philips Hue -Postmuse	-Facebook -Instagram -Skype	-Pinterest -Coursera -Evernote
Plataformas soportadas	-Android -iOS -Aplicaciones web	-Android -iOS	-JVM -Android -Navegador
Documentación	Información bien organizada	Fácil de usar, pero no está organizada	Está bien estructurada, pero las bibliotecas están en cambio continuo.
Popularidad	146 000 en GitHub en octubre 2022	105 000 en GitHub en octubre 2022	43 000 en GitHub en octubre 2022

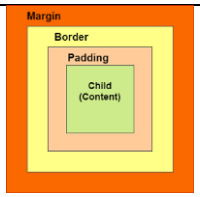
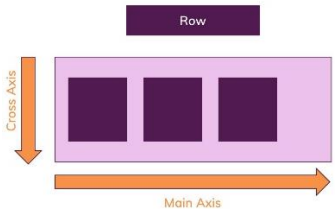
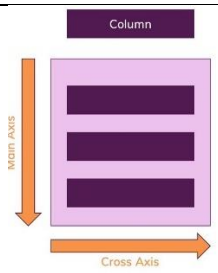
Elaborado por: El investigador

Haciendo un análisis del mejor framework para el desarrollo de la aplicación es flutter, debido a que es multiplataforma, además cuenta con documentación bien organizada y librerías libres en su página principal. Flutter crea interfaces por medio de widgets con estado (statefulwidget) y sin estado (statelesswidget), cada parte de la aplicación contiene un widget.

Figura 20.- widgets dentro de una aplicación
Elaborado por: El investigador

A continuación, se describen algunos widgets básicos para el desarrollo de aplicaciones en flutter.

Tabla 22.- Widgets básicos de Flutter

Widget	Descripción	Gráfica
AppBar	Es una barra de aplicaciones que se coloca en la parte superior de la aplicación	
Container	Primero rodea al widget secundario (interno) con relleno y luego aplica condiciones o restricciones	
Row	Es un widget que tiene la capacidad de mostrar a los widgets internos en forma horizontal	
Column	Muestra a todos los elementos internos en una matriz vertical	
Text	Tiene la capacidad de mostrar una cadena de texto con un estilo, maneja datos strings.	
Icon	Permite introducir diferentes iconos, dando un estilo apropiado	

Elaborado por: El investigador

1.4 Objetivos

1.4.1 Objetivo general

Desarrollar un sistema electrónico de notificación de emergencias basado en IoT para la asistencia médica a personas de la tercera edad.

1.4.2 Objetivos específicos

- Determinar los signos vitales y accidentes más frecuentes que sufren los adultos mayores.
- Seleccionar el software y hardware según los requerimientos tecnológicos de un sistema electrónico basado en IoT.
- Implementar un sistema inteligente capaz de notificar accidentes y el estado de salud de adultos mayores dentro del hogar.

CAPÍTULO II

METODOLOGÍA

2.1 Materiales

Para la elaboración del dispositivo IoT de alerta de emergencias para la asistencia médica a apersonas de la tercera edad se utilizó diferentes materiales y herramientas, tanto hardware como software. Para la parte de hardware del dispositivo se utilizó el módulo ESP32 como microcontrolador, MAX30100 para adquirir los parámetros de la frecuencia cardiaca, MPU6050 para medir la aceleración del usuario, MLX90614 para adquirir la temperatura corporal yTP4056 para recargar la batería de litio. En software se utilizó visual estudio apoyando en el desarrollo de código, flutter para crear la aplicación móvil, Docker para contenerización de los servicios de Node.js, EMQX, portainer y MongoDB.

2.2 Métodos

2.2.1 Modalidad de la investigación

En el presente proyecto de investigación se aplicó las siguientes modalidades de investigación:

Investigación Aplicada porque se empleó los conocimientos obtenidos en el transcurso de la carrera en desarrollar un sistema inteligente para monitorear y alertar las condiciones de salud de un adulto mayor en base a la arquitectura IoT.

Se utilizó la Investigación Bibliográfica para recopilar información relevante de trabajos similares realizados, utilizando los distintos medios que ofrece la universidad como libros, artículos, revistas o base de datos disponibles para estudiantes universitarios. Todos los recursos mencionados están disponibles en forma digital o física.

Al construir el sistema inteligente de alerta de emergencias y de monitoreo de signos vitales se aplicó la Investigación Experimental para realizar pruebas de funcionamiento y corregir errores que tenga el sistema, con el fin de tener un dispositivo funcional y de alta confiabilidad.

2.2.2 Recolección de la información

En la recolección de información se utilizó distintas bases de datos científicas para acceder a libros, artículos y revistas, también se utilizan los diferentes repositorios de las universidades ecuatorianas para obtener trabajos similares de titulación, de preferencia aquellos que han sido publicados en los últimos cinco años.

2.2.3 Procesamiento y análisis de datos

Al terminar la recolección de información se llevó a cabo un análisis en los siguientes puntos:

- Revisión de la información.
- Clasificación de información obtenida
- Análisis de la información para establecer soluciones al problema
- Selección de la propuesta de solución óptima
- Planteamiento de la propuesta de solución

2.2.4 Desarrollo del proyecto

Para realizar el proyecto se debe cumplir con las siguientes actividades que permitan implementar el sistema electrónico de notificación de emergencias:

- Recolección de información sobre los principales accidentes que sufren las personas de la tercera edad dentro del hogar.
- Análisis de las causas de alteraciones de los signos vitales en adultos mayores.
- Determinación de los accidentes más frecuentes y signos vitales que deben ser monitoreados.
- Identificación de la arquitectura IoT a implementar en el sistema de notificación de emergencias de adultos mayores.
- Selección de los módulos para la adquisición de signos vitales y detección de accidentes.
- Determinación de la tecnología inalámbrica a utilizar y la base de datos que mejor se acople al sistema.
- Diseño del sistema electrónico del dispositivo de notificación y monitoreo.
- Construcción de una estructura para incorporar todos los elementos del dispositivo IoT.
- Programación de las placas electrónicas.

- Desarrollo de la interfaz para la visualización de datos en el móvil.
- Pruebas del funcionamiento del dispositivo IoT.
- Corrección de errores.
- Elaboración del informe

CAPÍTULO III

RESULTADOS Y DISCUSIÓN

3.1 Análisis y discusión de los resultados

La implementación de un sistema de notificaciones de emergencias para adultos mayores permitió dar seguimiento en tiempo real a las constantes vitales como la temperatura corporal, frecuencia cardiaca y saturación de oxígeno, cuando alguno de estos parámetros han sobre pasado los rangos establecidos para este grupo de personas se enviará una notificación push al celular inteligente del cuidador del adulto mayor, además la interfaz permite crear alertas para notificar el estado de los signos vitales en intervalos de tiempo que el usuario elija. El sistema también es capaz de detectar caídas del adulto mayor por medio de "Machine Learning Integrado" en el microcontrolador, que posee un modelo inteligente basado en Machine Learning con alta capacidad para reaccionar ante un evento de caída. Todos los datos de los sensores son enviados por medio del bróker MQTT a una aplicación móvil para ser presentados al usuario.

3.2 Desarrollo de la propuesta

El desarrollo del sistema de notificación de emergencias se basó en los accidentes que sufren los adultos mayores en la vida diaria analizados en la tabla 1, el accidente de mayor ocurrencia son las caídas en el mismo nivel provocando lesiones de huesos o hospitalizaciones, además, existen factores de riesgo como la propia edad avanzada o el entorno que le rodea. En caso de producir un evento de caída esta puede ser monitoreado por medio de dispositivos electrónicos para notificar al cuidador del adulto mayor.

Por otra parte, existen otras necesidades de atención a la salud de los adultos mayores, analizados en la tabla 2, siendo los cuatro indicadores principales para determinar el estado de salud del paciente. Los indicadores de salud son temperatura corporal, frecuencia cardiaca, frecuencia respiratoria y presión arterial, pero no todos se puede monitorear usando algún tipo de sensor electrónico. A continuación, en la tabla 23 se hace un análisis de cada indicador.

Tabla 23.- Análisis de indicadores de salud [*[3], [4]*]

Indicador	Descripción
Temperatura corporal	Existen dos formas diferentes de medir la temperatura, de forma analógica y digital. Se puede realizar colocando un termómetro en diferentes partes del cuerpo como en el cuello, frente, axila u oreja. Medir en otras partes del cuerpo también es posible gracias al avance tecnológico, por lo tanto, la tecnología hace posible tomar la temperatura corporal.
Frecuencia respiratoria	Para medir la frecuencia respiratoria se hace contando el movimiento del pecho o el abdomen durante un minuto. También se utiliza sensores adhesivos acústicos, pero estos sensores no se pueden implementar en un dispositivo portátil debido a que añade una señal de ruido al sensor, por lo tanto, la tecnología no apoya para implementar este indicador en el dispositivo.
Presión arterial	Al intentar tomar la presión arterial, se utiliza una herramienta para ejercer presión en el brazo del anciano, convirtiendo en un método incomodo de uso continuo, entonces debido a la tecnología que necesita este indicador, no es posible medir la presión arterial en el sistema electrónico IoT.
Frecuencia cardiaca	La frecuencia cardiaca es un signo vital muy importante, debido a que representa el estado de vida. Si existe alteraciones en la frecuencia cardiaca entonces muestra un trastorno de salud de los adultos mayores, por lo tanto, debe manejarse lo antes posible. En el mercado existen algunos sensores de tamaño muy reducidos que ayudan a determinar la frecuencia cardiaca junto con la saturación de oxígeno de forma no invasiva. Por lo tanto, la importancia médica y la tecnología son motivos para la implementación de este indicador en el sistema.

Teniendo en cuenta los requerimientos descritos anteriormente, el dispositivo electrónico puede ser de tipo brazalete, para cargar en la muñeca, ya que en esta parte del cuerpo existe la arteria radial para tomar el pulso como muestra la figura 21, la temperatura se puede tomar en esta área, pero para mejorar la precisión, el sensor se coloca cerca de una arteria, además la muñeca ofrece comodidad para llevar puesto el dispositivo de forma continua.

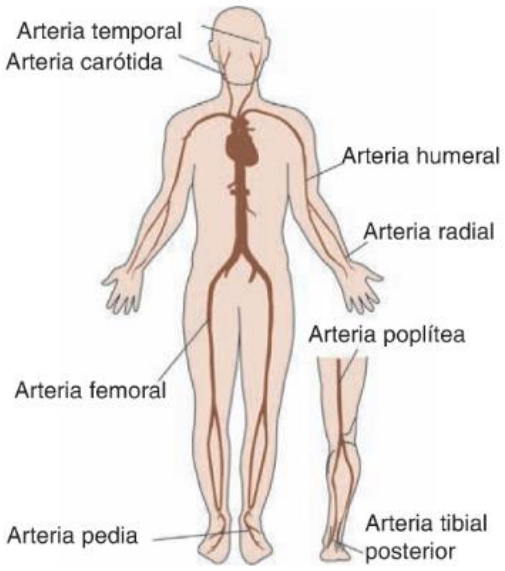


Figura 21.- Arterias para tomar el pulso *[24]*

3.2.1 Selección de los sensores

Para la adquisición de datos se tomó en cuenta sensores electrónicos que tengan un tamaño apropiado para implementar en un dispositivo IoT tipo brazalete y que se adapten perfectamente en la muñeca del usuario.

Sensor de temperatura

Según las características de los sensores de temperatura analizados en tabla 10 el óptimo es el sensor MLX90614, debido a que está diseñado para aplicaciones médicas, además el sensor no debe estar en contacto directo con el paciente ya que está diseñado para ser sensible a la radiación infrarroja emitida por un objeto a distancia. Otra ventaja es que a nivel de software es fácil de programar ya que dispone de librerías para Arduino IDE, mientras que, para hardware, solo necesita alimentación de 3 voltios y comunicación por el puerto I2C para la lectura de los datos.

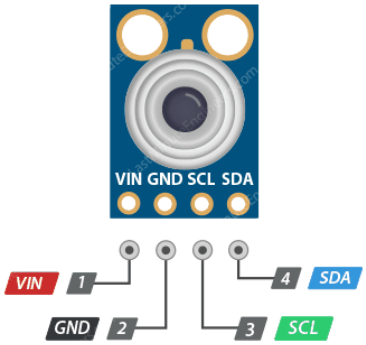


Figura 22.- Sensor MLX90614 *[50]*

Sensor de frecuencia cardiaca

Las características analizadas en la tabla 11 permiten establecer que el sensor MAX30100 presenta grandes beneficios como la incorporación de oxímetro en el mismo dispositivo para adquirir la saturación de oxígeno, además es adecuado para implementar en la muñeca del usuario para que recopile datos a través de la arteria radial, también es fácil de adquirir en el mercado. En base al software, dispone de librerías para Arduino IDE facilitando la programación y para hardware se necesita alimentación de 3 voltios, mientras que para recopilar los datos lo hace a través del puerto I2C.

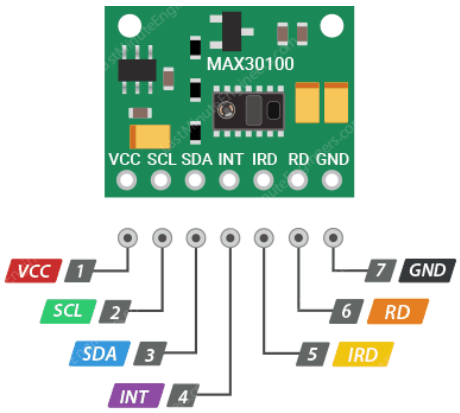


Figura 23.- Pinout del sensor MAX30100 [51]

Detector de caídas

En base a las características analizadas en la tabla 12 el acelerómetro que mejor se ajusta al sistema es el MPU6050, ya que dispone de un rango más amplio de sensibilidad, permitiendo detectar caídas en un intervalo de tiempo de milisegundos. Para programar este sensor es relativamente fácil ya que dispone de librerías compatibles con Arduino IDE y para recopilar los datos se necesita del puerto I2C del microcontrolador.

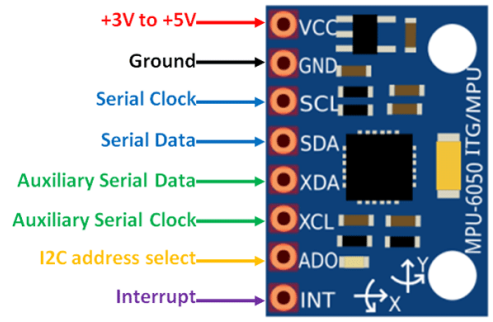


Figura 24.- Pinout del sensor MPU6050 *[52]*

3.2.2 Selección del microcontrolador

Al desarrollar un sistema IoT lo primordial que debe tener el microcontrolador es acceso a internet para interactuar con los protocolos de comunicación IoT, para lo cual se hace un análisis de las características más relevantes en la tabla 9. La tarjeta más optima es la ESP32 ya que dispone de doble núcleo de procesamiento, esto es de gran ayuda debido a que el sensor de caídas debe adquirir datos de forma continua sin perder intervalos de milisegundos porque el evento de caída dura alrededor de 200ms. Con respecto a la comunicación, la ESP32 dispone del puerto I2C para recibir datos de los sensores, se puede encontrar información más detallada en el anexo 1.

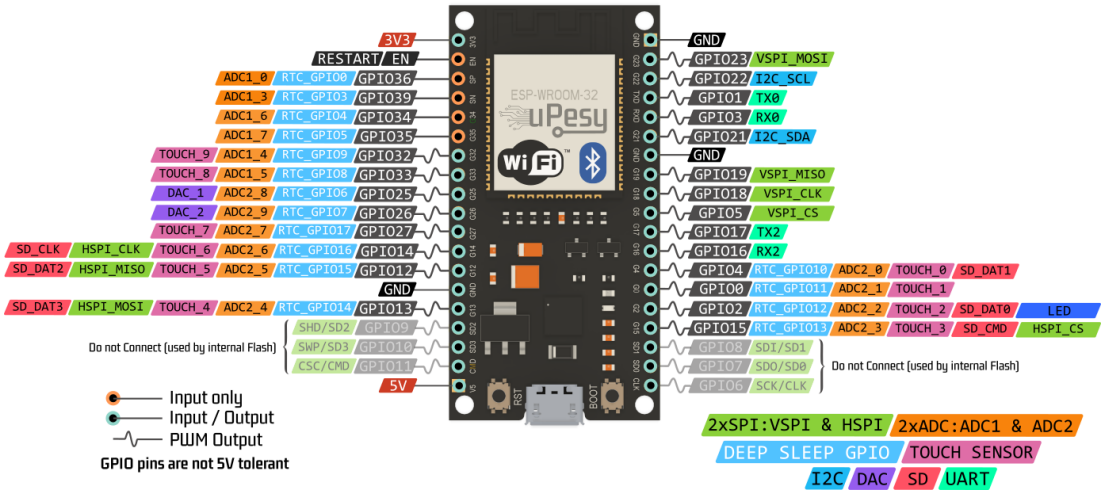


Figura 25.- Pinout de ESP32 *[53]*

3.2.3 Selección de los servicios web

Para la persistencia de datos, gestión de usuarios y transmisión de datos IoT es necesario implementar tecnología de virtualización de aplicaciones de software para lo cual se utilizó de Docker, ya que es de código abierto y dispone de documentación muy amplia en la web, como herramienta para administrar los contenedores se usó portainer.

Una vez, analizada las características de los diferentes brokers en la tabla 19, se determina que EMQX es apropiado para la trasmisión de datos ya que permite procesar y trasmitir millones de mensajes en cada segundo. El bróker seleccionado trabaja con protocolos MQTT y COAP convirtiendo en un bróker altamente escalable. Por otra parte, analizando las características de la tabla 20, se determina que MongoDB es la mejor opción para persistencia de datos ya que es de código abierto y es compatible con la mayoría de los lenguajes de programación, siendo el más usado JavaScript.

Con el fin de gestionar a los usuarios es necesario implementar un servidor, para lo cual es de suma importancia usar el servicio de Node.js, debido a que permite ejecutar código escrito en JavaScript y así comunicar con los servicios de EMQX y MongoDB.

Analizando la tabla 21, el framework óptimo para aplicación del sistema de emergencias es Flutter ya que es multiplataforma, es decir, ejecutable en IOS y Android, además contiene documentación bien organizada y permite emular en tiempo real.

3.2.4 Arquitectura del sistema IoT

La arquitectura del sistema IoT está formado por cuatro capas, empezando por la capa de adquisición siendo la encargada de tomar los datos del usuario que lleva puesto el brazalete en la muñeca, para posteriormente enviar a la capa de procesamiento cuyo trabajo lo realiza una ESP32 haciendo el trabajo de analizar y procesar las señales captadas por los sensores, para luego transmitir por medio de un protocolo IoT a la capa de presentación, aquí el usuario final puede interactuar con los datos del adulto mayor y tomar acciones oportunas, la arquitectura se aprecia de mejor manera en la figura 26.

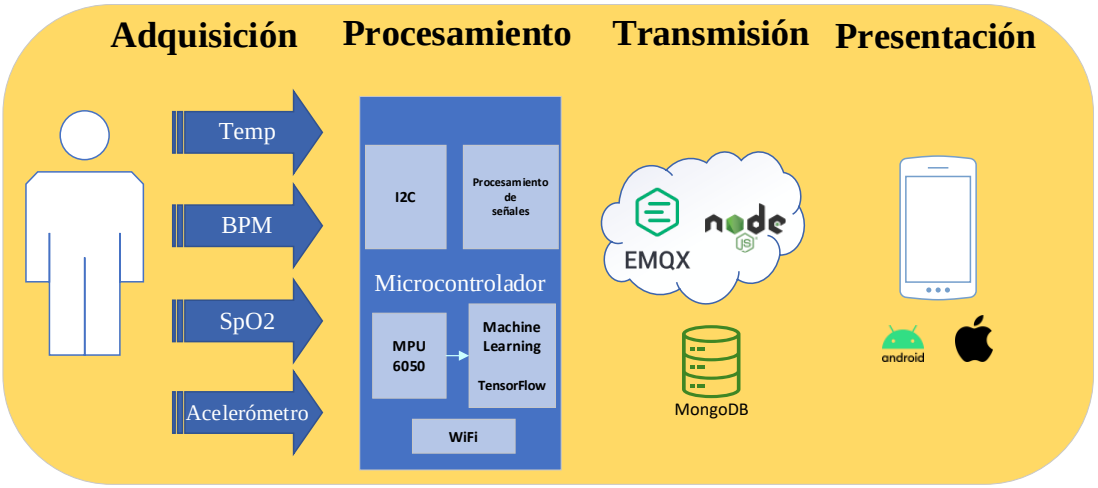


Figura 26.- Arquitectura del sistema

Elaborado por: El investigador

3.2.5 Capa de adquisición y procesamiento de datos

Para la adquisición de datos del usuario, se conectó todos los sensores al puerto de comunicación serial (I2C) ya que este puerto es capaz de soportar 127 dispositivos como esclavos, están configurados con una velocidad de trasmisión de 100kbits/s, además los sensores están alimentados con un voltaje propio de la ESP32 de 3.3 V, como se muestra en la figura 27.

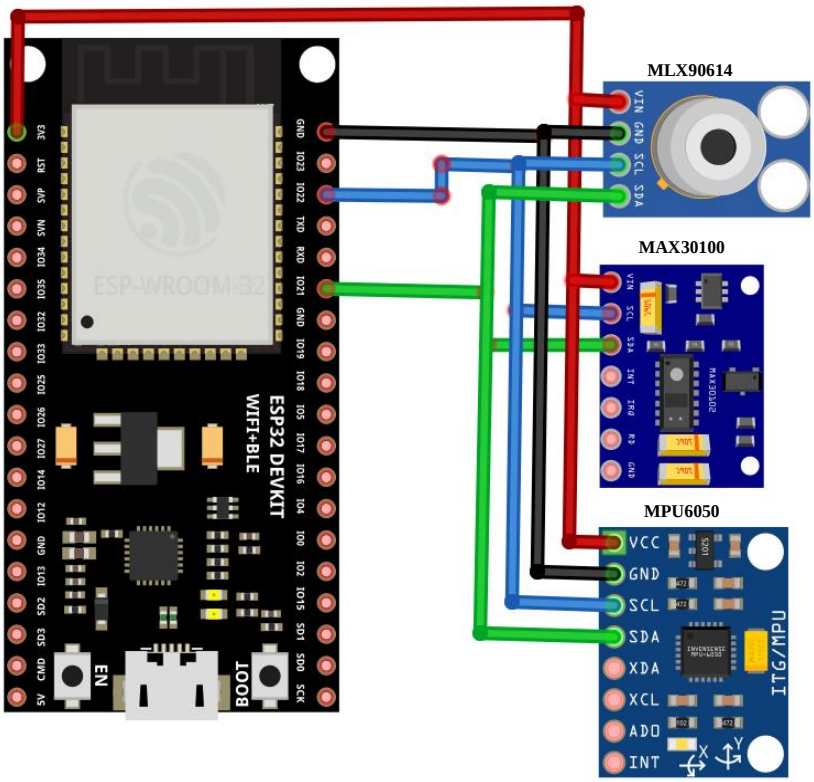


Figura 27.- Diagrama de conexión con los sensores
Elaborado por: El investigador

Para programar la ESP32 se utilizó el software "Visual Studio Code" mediante la extensión de platformIO ya que es un IDE abierto para C y C++ basado en Python. Para iniciar se instaló librerías mediante "PIO Home" extensión librerías, todas estas están referenciadas en un archivo llamado platformio.ini el mismo que está encargado de gestionar la configuración propia del microcontrolador.

Temperatura corporal: Para captar la radiación infrarroja del cuerpo humano se usó el sensor MLX90614, que dispone de librerías fáciles de implementar, como la Adafruit_MLX90614 que llama a la función readObjectTempC() para detectar la temperatura del objeto.

Frecuencia cardiaca: El sensor MAX30100 es el encargado de capturar datos de la frecuencia cardiaca y saturación de oxígeno, contiene un circuito fotoeléctrico, con un led rojo de longitud de onda de 660 nm y un led infrarrojo de longitud de onda de 880 nm, como se muestra en el anexo 2. Para la funcionalidad del sensor es necesario implementar la librería MAX30100_PulseOximeter ya que dispone de una función callback cuyo funcionamiento es detectar la mayor cantidad de luz IR absorbida por la sangre, de esta forma se determina la frecuencia cardiaca.

Detección de caídas: Para detectar caídas se utiliza el sensor MPU6050 ya que permite obtener la aceleración en los ejes x, y, z. Al implementar un dispositivo IoT tipo brazalete el sistema puede confundir una caída con un simple movimiento de la muñeca o al hacer movimientos rápidos, para que el dispositivo presente confiabilidad se utilizó un campo de la Inteligencia Artificial conocido como Machine Learning, para que sea capaz de sacar sus propias conclusiones a partir de los datos que le proporciona el acelerómetro.

Analizando los beneficios de la tabla 13, se puede implementar Machine Learning integrado (TinyML) en el dispositivo IoT ya que es posible ejecutar modelos inteligentes dentro de la ESP32, evitando gastos en la transmisión y en el procesamiento de los datos en la nube. Además, la ESP32 es compatible con la biblioteca móvil TensorFlow Lite, ayudando en el desarrollo de modelos inteligentes y ligeros con alta capacidad para reaccionar ante un evento de caída.

Para el desarrollo de Machine Learning Integrado es necesario el uso de aplicaciones que trabajen con bloques de procesamiento de señales y bloques de aprendizaje autónomo, para lo cual analizando la tabla 14 se selecciona la plataforma EDGE IMPULSE que es apropiada para crear este tipo de modelos compatibles con los microcontroladores de Espressif (ESP32).

Para lograr una detección de caídas los datos recogidos por el acelerómetro deben pasar por un proceso de aprendizaje, mientras más datos de entrenamiento tenga el modelo más preciso serán las detecciones de caídas. Para crear un modelo robusto es necesario tomar los datos del acelerómetro y enviar hacia la plataforma Edge Impulse. Como primer paso es necesario crear una cuenta en https://studio. edgeimpulse.com/signup y un proyecto para finalmente instalar el CLI (Interfaz de línea de comandos) de Edge Impulse en este caso se instala para Windows 10 como se muestra en la tabla 24.

Tabla 24.- Instalación de CLI de Edge Impulse.

Herramientas	URL de descarga o comando para instalar
Python 3	https://www.python.org/
Node.js V14 o superior	https://nodejs.org/en/
Instalar CLI por consola	npm install -g edge-impulse-cli --force

Elaborado por: El investigador

Al terminar la instalación se puede ejecutar por consola diferentes herramientas de CLI de Edge Impulse, descritas en el anexo 3. En el envió de datos a la ESP32 se usó la herramienta "data forwarder" que permite recopilar los datos del puerto serial (COM) y enviar a la plataforma de Edge Impulse. La recopilación de datos del acelerómetro se realiza conectando el sensor MPU6050 a la ESP32 por medio de los pines SCL y SDA (I2C) y se alimenta a 3.3 V, la palca de desarrollo debe estar conectado a la computadora por medio de un cable USB para que pueda detectar el CLI al puerto serial que llegan los datos del sensor, como se muestra en la Figura 28.

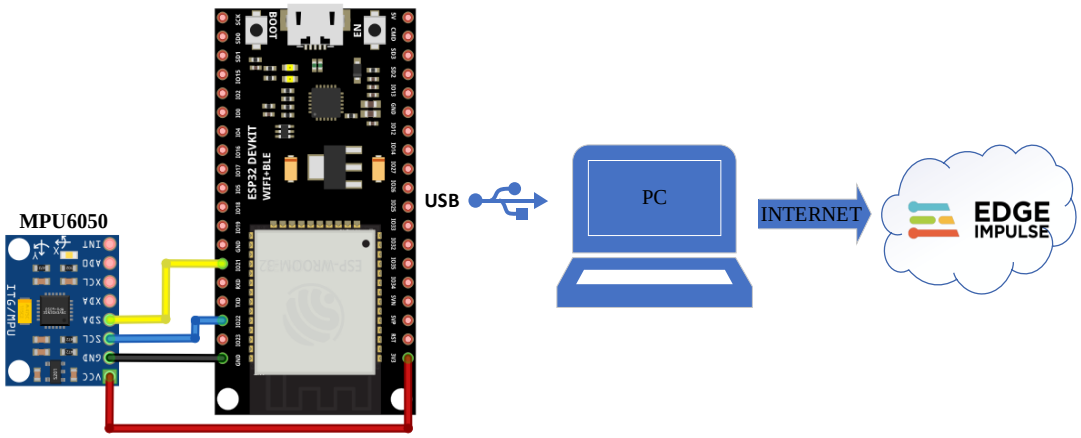


Figura 28.- Diagrama de conexión para enviar datos a Edge Impulse
Elaborado por: El investigador

Para recopilar los datos del sensor se programa la ESP32 con la librería Adafruit_MPU6050 para configurar la sensibilidad a ±2g convirtiendo al sensor lo más sensible posible, además es necesario especificar el puerto serial al que llegarán los datos en este caso se usó el COM8. Para que Edge Impulse pueda detectar los datos se debe enviar a una frecuencia relativamente baja y el acelerómetro debe enviar los datos separados por una coma (,) como se muestra en la figura 29, el código de programación de la ESP32 se presenta en el anexo 4.

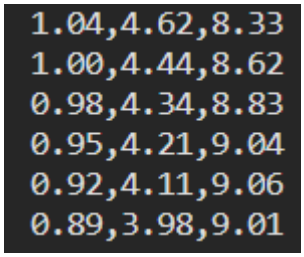


Figura 29.- Formato de datos para enviar a través de CLI
Elaborado por: El investigador

La herramienta “data forwarder” de Edge Impulse, permite leer los datos de puerto serial a una tasa de 115200 baudios por lo que es posible trabajar con frecuencia de muestreo inferior a 400Hz. Después de varias pruebas, se define la frecuencia de muestreo de181 Hz, ya que es la frecuencia más alta que permite conectar “data forwarder” con Edge Impulse. Para hacer este proceso se ejecuta en el símbolo del sistema (CMD) el comando:

```
edge-impulse-data-forwarder
```

El “data forwarder” le pedirá que inicie sesión con las credenciales registradas, también debe elegir al proyecto que va a registrar los datos y especificar el nombre de cada variable como se muestra en la figura 30.

```
:\Users\USER>edge-impulse-data-forwarder --clean
dge Impulse data forwarder v1.15.1
 What is your user name or e-mail address (edgeimpulse.com)? jguaman8178@uta.edu.ec
 What is your password? [hidden]
ndpoints:
   Websocket: wss://remote-mgmt.edgeimpulse.com
   API:       https://studio.edgeimpulse.com/v1
   Ingestion: https://ingestion.edgeimpulse.com

SER] Connecting to COM8
SER] Serial is connected (00:01)
WS ] Connecting to wss://remote-mgmt.edgeimpulse.com
WS ] Connected to wss://remote-mgmt.edgeimpulse.com

 To which project do you want to connect this device? Jefferson Guaman / fall_proyecto
SER] Detecting data frequency...
SER] Detected data frequency: 181Hz
 3 sensor axes detected (example values: [-2.02,1.36,9.08]). What do you want to call them? Separate
the names with ',': acX,acY,acZ
WS ] Device "MPU_Fall" is now connected to project "fall_proyecto"
WS ] Go to https://studio.edgeimpulse.com/studio/138671/acquisition/training to build your machine l
arning model!
```

Figura 30.- Parámetros al ejecutar “data forwarder”
Elaborado por: El investigador

Para verificar que el dispositivo está conectado a Edge Impulse se debe ingresar a la página principal de Edge Impulse, seleccionar el proyecto registrado en el CLI (figura

30) y finalmente al apartado “devices” aquí se muestra los detalles del sensor conectado y el estado de este.

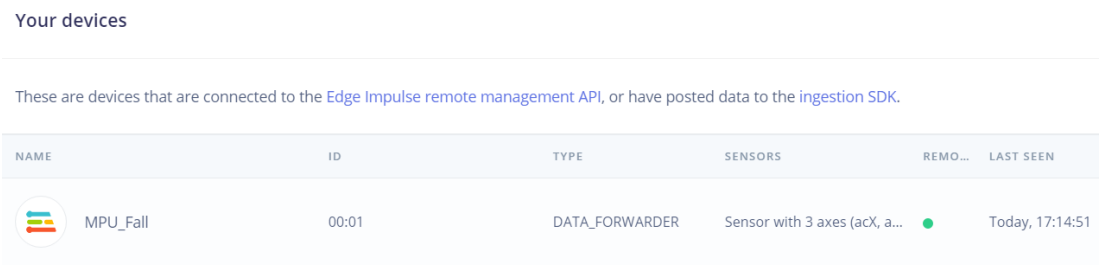


Figura 31.- Verificar conexión del dispositivo en Edge Impulse
Elaborado por: El investigador

Mediante el apartado de “*data adquisition”* se recopiló datos del sensor MPU6050, especificando las etiquetas que serán usadas para clasificar los datos más adelante. También se debe definir la longitud de la muestra en milisegundos y las frecuencias que ha detectado el CLI, también es posible observar la información recolectada y editar ya sea la etiqueta o dividir la muestra tomada, como se puede observar en la figura 32.

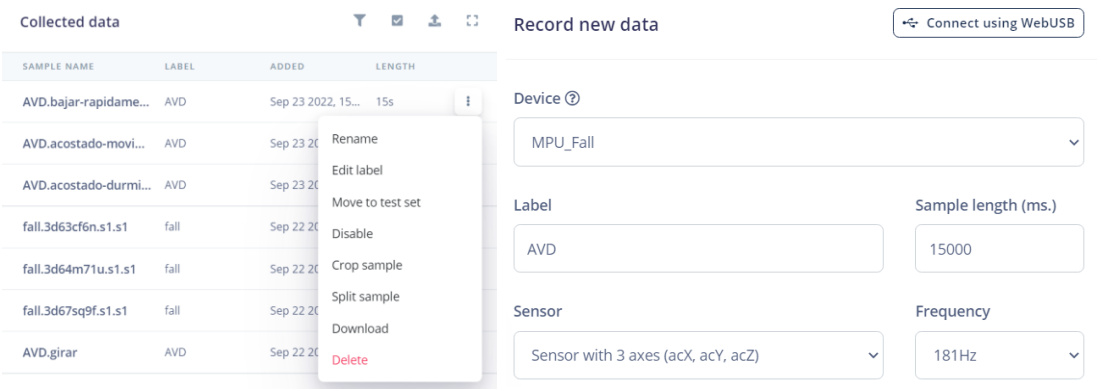


Figura 32.- Registrar nuevos datos
Elaborado por: El investigador

Para iniciar con la recopilación de datos hay que definir dos grupos de actividades, entre ellas están las actividades de la vida diaria (AVD) y caídas (fall). Para el registro de datos de entrenamiento es necesario tomar datos de aceleración cuando ocurren caídas y las actividades diarias.

La investigación “SisFall” publicada en el 2017 realiza un amplio estudio con 704 personas de la tercera edad, donde determinan que los adultos mayores no tienen la capacidad de reaccionar ante un evento de caída es decir llegan al piso en la misma posición que estuvieron de pie. Además, mediante una encuesta determinaron que las

caídas fueron causadas por tropiezos, resbalones y perdidas de equilibrio; donde el 60 % de caídas fueron hacia adelante. [54]

Al recopilar datos de caídas con personas de la tercera edad se corre el riesgo de tener lesiones graves o fracturas de huesos por tal motivo se opta recopilar datos de caídas con un grupo de adultos jóvenes, pero basando en la investigación de SisFall [54]. Entonces, se realizó con un grupo de 10 personas, 5 mujeres y 5 hombres en un rango de edad de 35 años hasta 50 años (anexo 5). Cada participante realizó 5 tipos de caídas: caer hacia adelante, caer vertical, caer hacia atrás, caer hacia atrás mientras está sentado y caer lateral mientras está sentado, cada caída fue repetida por 10 veces obteniendo 50 caídas de cada participante, de forma similar se hizo para recolectar actividades de la vida diaria, como se muestra en la tabla 25.

Tabla 25.- Tipos de actividades para entrenar al modelo

Caídas		
Tipo caída	**Cantidad**	**Tiempo (seg)**
Caída hacia delante al caminar a causa de un tropiezo.	10	4
Caída vertical al caminar a causa de resbalón o tropiezo	10	4
Caída hacia atrás al caminar causada por un resbalón o desmayo	10	4
Caerse hacia atrás mientras está sentado	10	4
Caída lateral mientras está sentado	10	4
AVD (actividades de la vida diaria)		
Caminando despacio	3	15
Caminando rápido	3	15
Trotando rápido	1	100
Sentar y levantar de una silla	3	15
Movimientos en la cama (acostado)	1	100
Subir y bajar mano	1	100
Movimiento horizontal de la mano	1	100
Atrás y adelante la mano	1	100
Tocar el rostro y cabeza	1	100
Rascar	1	100
Diferentes movimientos de la mano	1	60
Girar la mano	1	20

Elaborado por: El Investigador

Las actividades y caídas de la tabla 25 para entrenar al modelo fueron elegidas en base a la investigación “SisFall” [54].

La recopilación de caídas tiene una muestra de 4 segundos, tiempo suficiente para grabar un evento de caída, mientras que para las actividades se hace en diferentes intervalos de muestras según la actividad, esto con el fin de mejorar la clasificación del modelo.

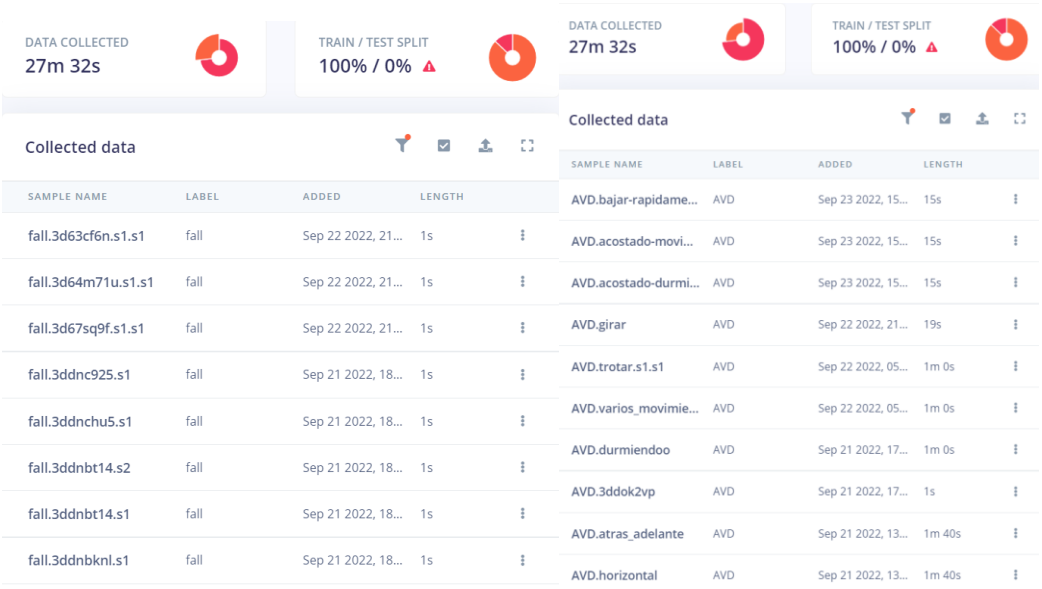


Figura 33.- Recolección de diferentes actividades en Edge Impulse
Elaborado por: El investigador

Al finalizar la adquisición de datos, Edge Impulse permite procesar la señal por medio de bloques de procesamientos de señal para extraer características esenciales de la señal, esto se hace en el apartado “create impulse”. A partir de la información presentada en la tabla 15, el bloque de procesamiento óptimo para datos de un acelerómetro es el “spectral analysis" ya que permite analizar patrones repetitivos y movimientos continuos, además extrae características como la frecuencia y potencia de las señales de acelerómetro.

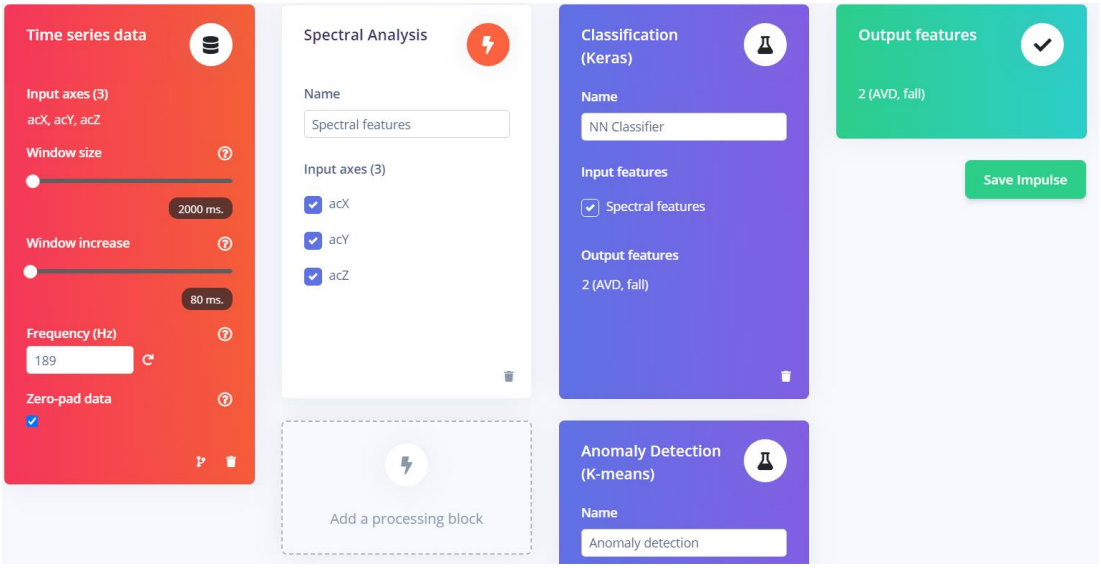


Figura 34.- Insertando bloques de procesamiento y aprendizaje
Elaborado por: El investigador

El bloque de procesamiento “spectral analysis" tiene la capacidad de filtrar frecuencias no deseadas mediate el filtro pasa bajo o pasa alto, ayudando a suavizar la señal, en este caso no se utiliza ningún tipo de filtro. Entonces, la señal pasa a la sección de “potencia espectral” donde calcula la FFT (Transformada Rápida de Fourier) con el fin de obtener las características espectrales, como indica la figura 35.

Los parámetros necesarios para filtrar y FFT son:

- Escala: multiplica todos los valores de entrada sin procesar por el número 1, es decir no se aplica scaling a la señal.
- Tipo: es el tipo de filtro que se utiliza, en este caso ningún tipo de filtro ya que, los mejores resultados se obtienen al no aplicar un filtro, esto se puede apreciar en el aparto de resultados en la tabla 38.
- Longitud FFT: es el tamaño de FFT. Ayuda a determinar la cantidad de contenedores FFT. Un número bajo significa que más señales se promediarán juntas dentro del mismo contenedor FFT, además reduce la cantidad de funciones y el tamaño del modelo. Un número más alto separará mas señales en contenedores separados, pero genera un modelo más grande. Este apartado acepta números de potencia de 2, como el modelo se implementa en la ESP32 es decir, se requiere un modelo pequeño que no consuma demasiada memoria, para lo cual se utiliza el número 32, con este valor consume los recurso indicados en la figura 36.

Figura 35.- Parámetros de filtrado y FFT
Elaborado por: El investigador

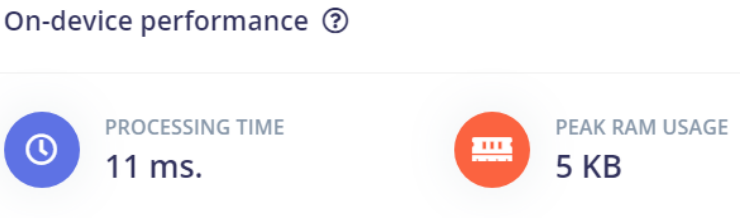


Figura 36.- recursos que consume el modelo
Elaborado por: El investigador

Una vez que la señal ha pasado por “spectral análisis” las características recopiladas se enviaron a bloques de aprendizaje descritos en la tabla 18, como el objetivo es clasificar en dos grandes grupos los datos se utilizó la clasificación keras y el detector de anomalías para datos que el modelo no ha sido entrenado. Para determinar el número de ciclos de entrenamiento se emplea la técnica “Early Stopping” descrita en la tabla 16, de forma similar se determinar la tasa de aprendizaje siguiendo las recomendaciones de Bosch Rué y colaboradores que determinan la tasa de aprendizaje por medio de la experimentación, [42] esto se puede apreciar en la tabla 17.

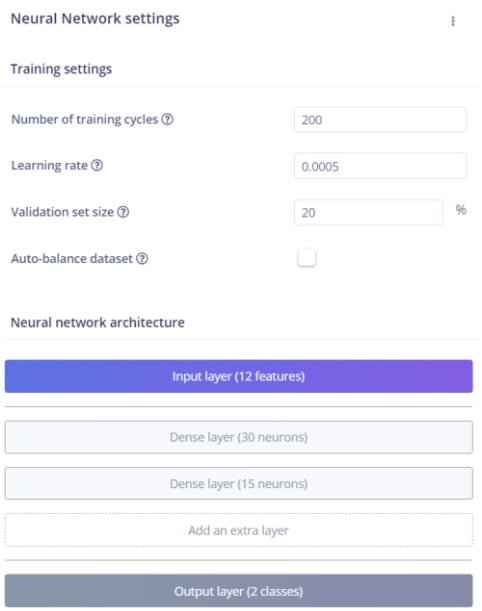


Figura 37.- Entrenamiento de la red
Elaborado por: El investigador

Para configurar el bloque de detección de anomalías se eligen los ejes que proporciona la señal es decir x,y,z con el fin de calcular el grupo mas cercano para un nuevo punto de datos y mostrar la distancia desde el borde del grupo, si esta dentro de un clúster (sin anomalías) se obtendrá un valor inferior a 0. Por otra parte, se debe crear un número de clústeres, aunque este valor puede ser ideal en algunos casos ya que la serie de datos en el tiempo siempre está cambiando, por lo que un valor de "cluster" puede ser ideal en diferentes momentos.

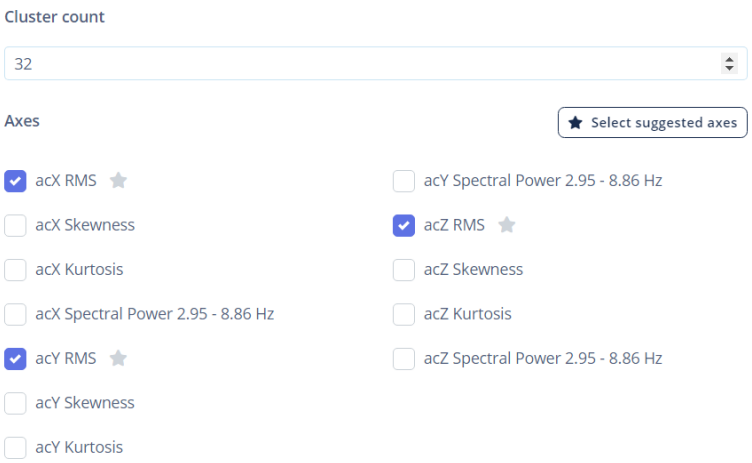


Figura 38.- Configuración para detección de anomalías
Elaborado por: El investigador

A continuación, se despliega el modelo para implementar dentro del código de la tarjeta ESP32, desde la plataforma Edge Impulse se procede a descargar un archivo en formato .zip para usar en el código como una librería de Arduino.

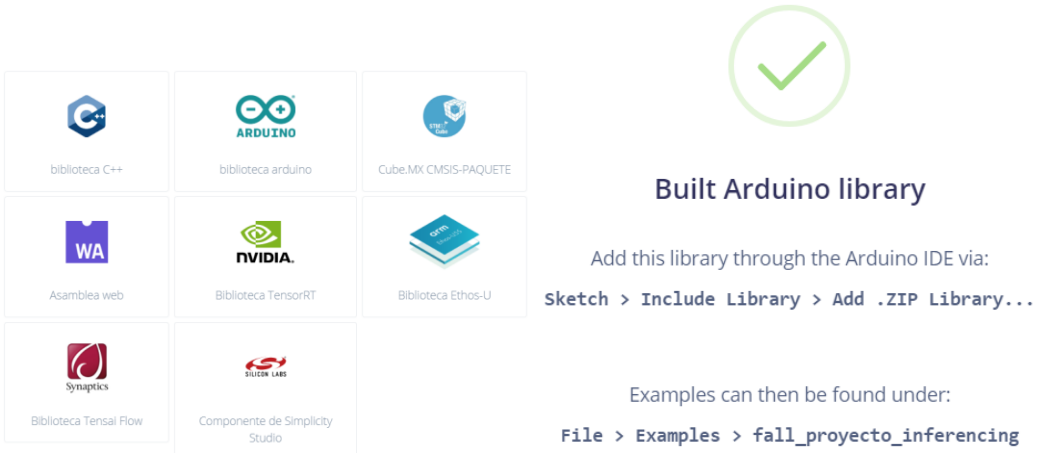


Figura 39.- Construir el modelo en biblioteca de Arduino
Elaborado por: El investigador

Programación del dispositivo IoT

Para implementar la librería "ei_fall_proyecto_arduino" (creada anteriormente) en la programación de la placa ESP32 se usó platformIO. En la figura 40 muestra el archivo raíz "platformio.ini" aquí se declaró la librería descargada de Edge impulse junto con las demás librerías, pero también es necesario colocar la librería en formato .zip dentro de la carpeta lib.

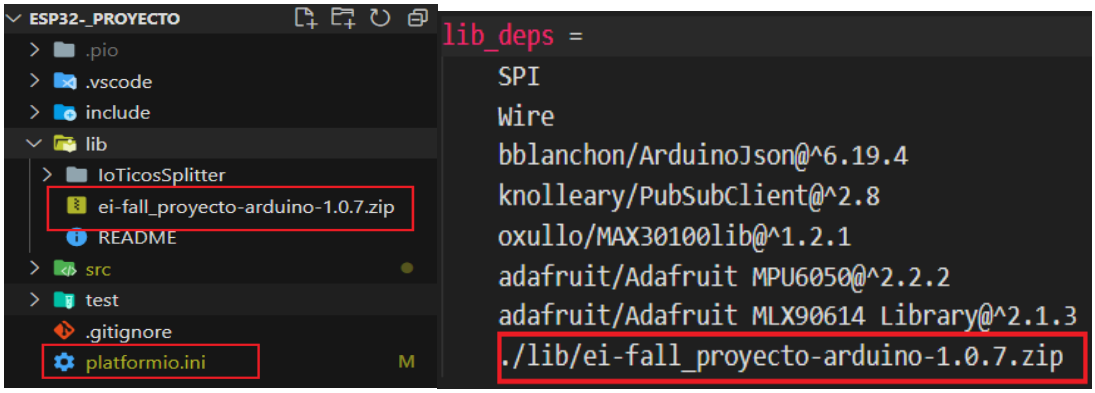


Figura 40.- Insertar librería en platformIO
Elaborado por: El investigador

En el archivo main.ccp se incluye a la librería de Edge Impulse, esto se puede apreciar en el anexo 6, aunque la forma de incluir se va a ver reflejado por el nombre del proyecto que se coloca en la plataforma de Edge Impulse, en este caso se llama a la librería como:

```
#include <fall_proyecto_inferencing.h>
```

Para la lectura de datos de los sensores y detección automática de caídas se usó el núcleo 0 de la ESP32, mientras que la comunicación inalámbrica al servidor y envío de datos al bróker se usó el núcleo 1 de la tarjeta, esto se hace con el fin de mejorar la detección de caídas.

La clasificación de actividades “AVD” y “fall” se determinan a través de probabilidades con valor numérico 1 según los datos del acelerómetro analizados. Al conectar la salida del puerto serial arroja los datos del sensor que están ingresando (features) y el valor de probabilidad de la clasificación como se aprecia en la figura 41, lo que quiere decir que el modelo ha clasificado como “AVD” ya que el valor de probabilidad es 1.

```
Features (63 ms.): 0.053264 -2.356182 4.074813 -1.566201 2.501677 2.401073
Running neural network...
Predictions (time: 0 ms.):
AVD:  1.000000
fall: 0.000000
Anomaly score (time: 0 ms.): 0.352483
(DSP: 63 ms., Classification: 0 ms., Anomaly: 0 ms.):
```

Figura 41.- Salida del puerto serial al implementar el modelo
Elaborado por: El investigador

Otros requerimientos técnicos es que el microcontrolador debe permanecer conectado a la red Wifi para que pueda conectar al servidor y enviar los datos por el protocolo IoT, además debe autenticar como un usuario, esto por medio de peticiones HTTP, por lo tanto, se realiza la codificación para que la ESP32 se conecte a wifi y envié los datos por medio de MQTT, estos parámetros siempre van a ser interpretados por la codificación.

Finalmente, se incluye la librería del sensor MLX90614, para la codificación se hizo la lectura de la radiación emitida por el objeto y se determina en °C, por otra parte, se incluye la librería del sensor MAX30100 para que trabaje con la función “callback”,

esta función se activa cuando detecta mayor cantidad de luz IR absorbida por la sangre de esta forma se determina la frecuencia cardiaca y saturación de oxígeno.

En la figura 42 se puede apreciar de mejor manera el diagrama de flujo que sigue el dispositivo IoT.

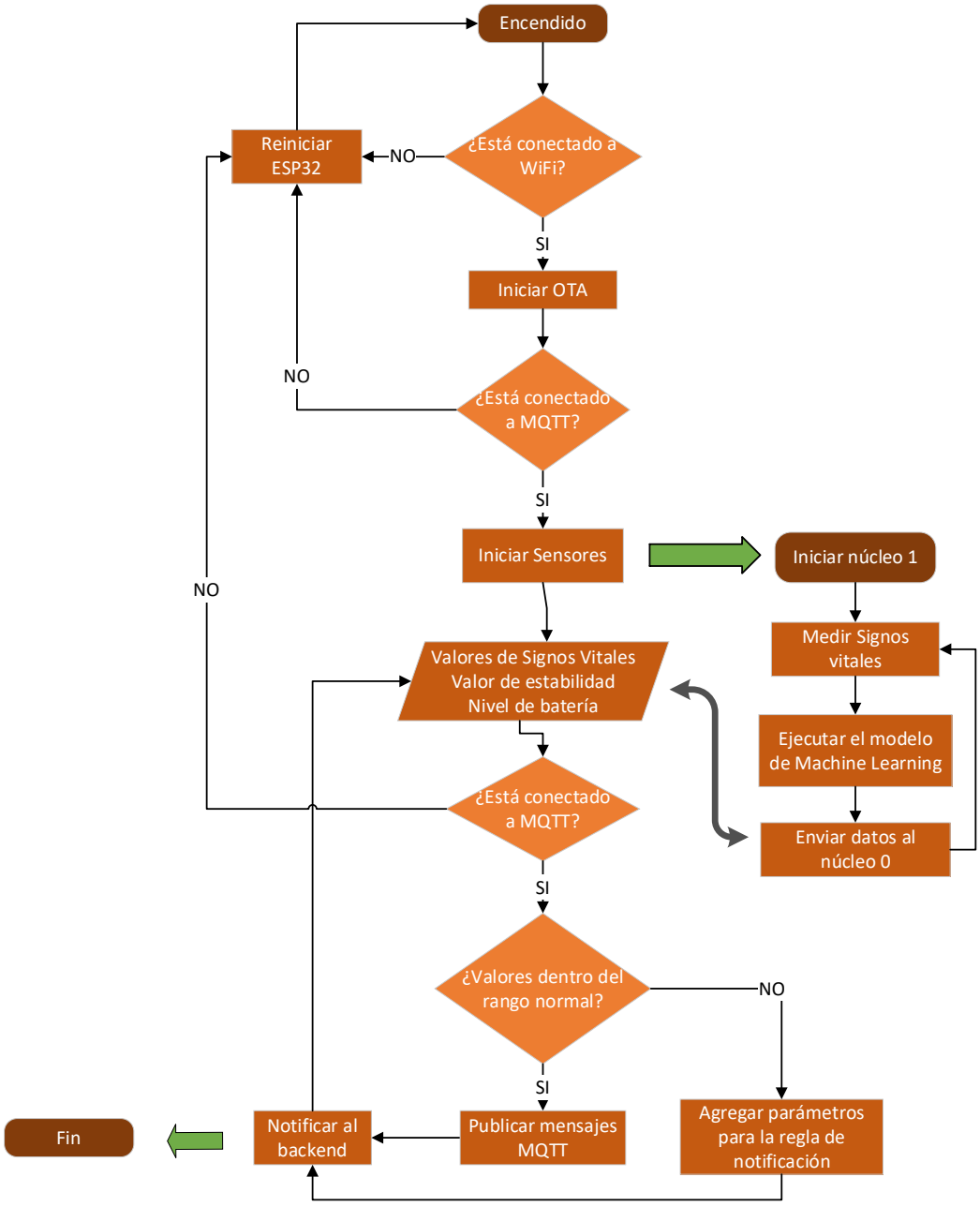


Figura 42.- Diagrama de flujo que sigue la ESP32
Elaborado por: El investigador

Suministro de energía

Para el suministro de energía se hace un análisis del consumo energético que tienen los sensores del brazalete, como muestra la tabla 26.

Tabla 26.- Consumo energético de los sensores

Componente	Corriente	Voltaje
MAX30100	600µA – 1200µA	3V a 5V
MPU6050	500µA	3.3V
MLX90614	1mA – 1.5mA	2.4 V a 3.6 V

Elaborado por: El investigador

Sumando el máximo consumo de corriente de los sensores se tiene:

$$I_{sensores} = I_{MAX} + I_{MPU} + I_{MLX}$$

$$I_{sensores} = 1.2mA + 0.5mA + 1.5\text{mA}$$

$$I_{sensores} = 3.2mA$$

Por otro lado, el consumo de la ESP32 varía de acuerdo que tecnología inalámbrica y los protocolos de comunicación que se utiliza. Para el sistema se utiliza la red Wifi y Dual Core, este consumo se ve reflejado en la tabla 27.

Tabla 27.- Consumo de corriente para la ESP32

Concepto	Consumo de corriente
Uso de dual-core	30mA – 68mA
Wifi	95mA - 180mA

Elaborado por: El investigador

Entonces para determinar el consumo máximo de corriente por la ESP32 se hace los cálculos en base a los valores máximos.

$$I_{ESP32} = I_{core} + I_{wifi}$$

$$I_{ESP32} = 68mA + 180mA$$

$$I_{ESP32} = 248mA$$

Por lo tanto, la ESP32 puede llegar a consumir 248mA siendo el valor máximo de consumo, pero si la ESP se mantiene en temperatura de alrededor de 25 °C el consumo se mantiene en valores mínimos consumiendo alrededor de 125mA.

Entonces la corriente máxima de consumo del dispositivo IoT sería:

$$I_{Max} = I_{sensores} + I_{ESP32}$$

$$I_{Max} = 3.2mA + 248mA$$

$$I_{Max} = 251.2mA$$

El suministro de energía para el brazalete debe ser capaz de alimentar a tres sensores y la ESP32. Como los sensores trabajan con un voltaje de alimentación de 3.3V al igual que la placa de desarrollo, con una corriente máxima de consumo de 251.2mA, por lo que se selecciona una batería de Litio recargable de 3.7V con capacidad de 1000mAh. Para recargar la batería se utilizó del módulo TP4056 ya que permite proteger la batería ante cargas y descargas por debajo de 2.4V ayudando a optimizar la vida útil de la batería.

Con el fin de alertar el nivel de la batería se usa el pin analógico GPIO36, cuando la batería está cargada el voltaje alcanza los 4V, mientras que cuando esta descargada llega hasta los 3.2V; con estos datos se hace un mapeo para determinar el nivel de la batería. Para leer el voltaje de la batería es necesario implementar un divisor de voltaje cuya salida debe alcanzar un máximo de 3.3V ya que los pines analógicos de la ESP32 permiten leer voltajes hasta este rango, en caso de sobrepasar este valor puede dañar el microcontrolador. El diagrama de conexión para el suministro de energía se muestra en la figura 43.

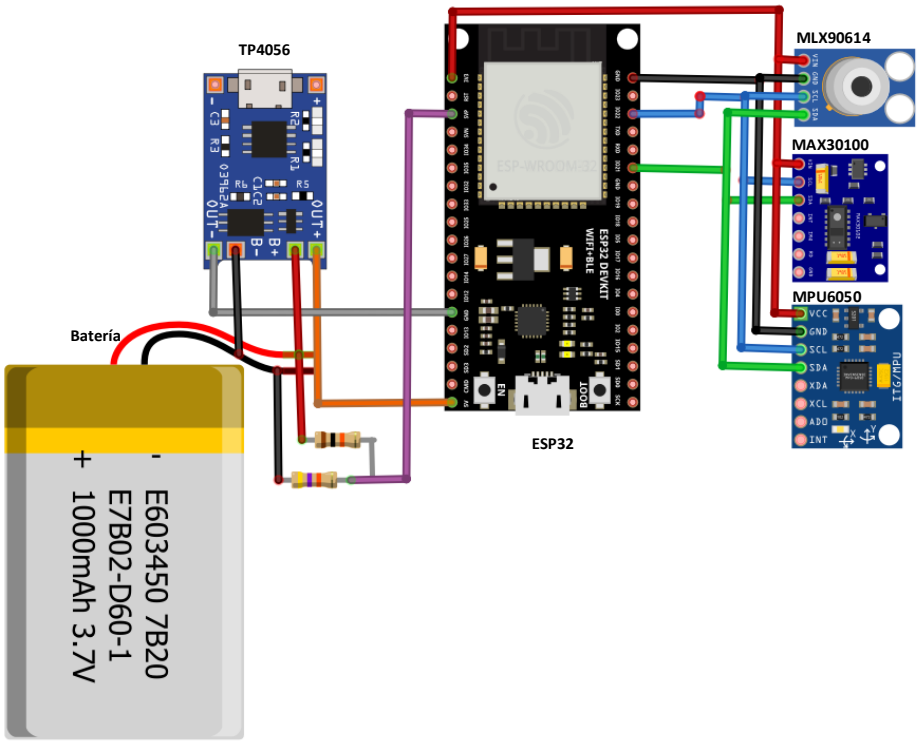


Figura 43.- Suministro de energía para el brazalete
Elaborado por: El investigador

3.2.6 Capa de transmisión

En la capa de transmisión es necesario implementar los servicios de EMQX, MongoDB y Nodejs, cada uno tiene una tarea específica dentro del sistema IoT, para esto se utilizó Docker para ejecutar las imágenes de forma separada. La gestión de los contenedores se realizó con portainer, para instalar se ejecutó el comando "*docker pull portainer/portainer"* a través del puerto 9000, todos los servicios están corriendo dentro de un sistema operativo Ubuntu.

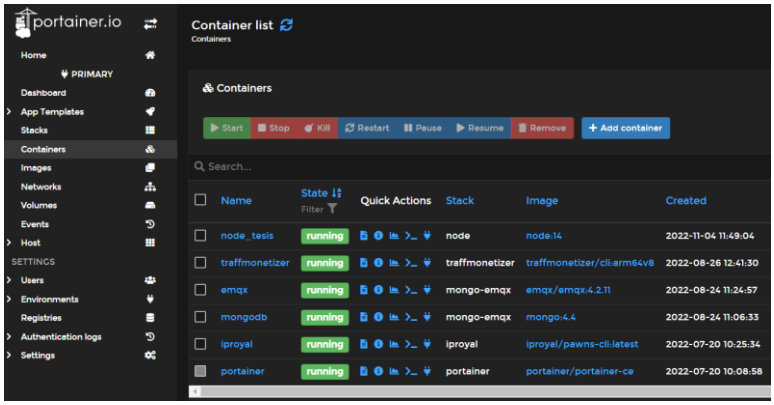


Figura 44.- Interfaz de portainerio
Elaborado por: El investigador

Para almacenar los datos de los usuarios se implementó “mongoDB” como base de datos, para lo cual, es necesario crear el contenedor por medio de “Docker Compose” o lo que es lo mismo se puede definir por los stacks (colección de servicios) de “portainer”, donde se define el nombre puerto 27017 y credenciales para conectar con los servicios de “Nodejs” y “EMQX” (anexo 7). Para gestionar la base de datos es necesario implementar el software “mongoDBCompas” y acceder por medio de la dirección *192.168.100.160:27017* y proporcionar todas las credenciales.

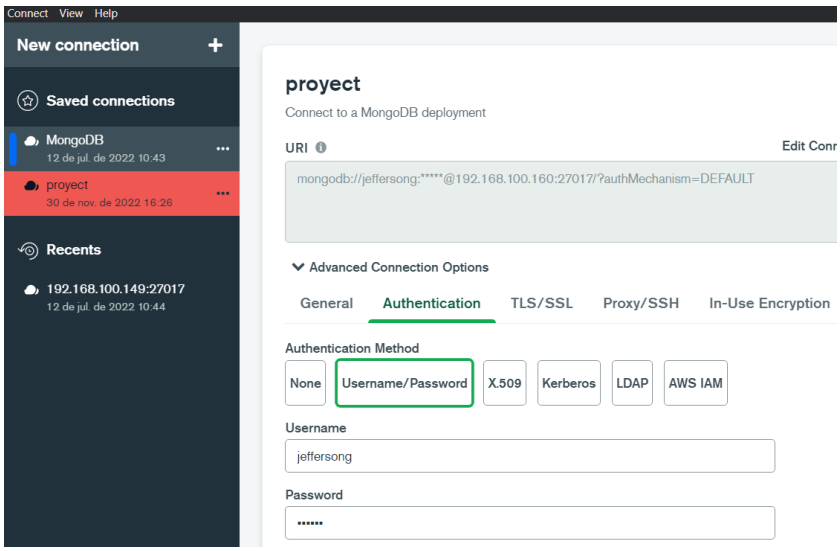


Figura 45.- Interfaz de MongoDBCompas
Elaborado por: El investigador

EMQX funciona como bróker para correr el protocolo MQTT, en el presente proyecto se utilizaron los siguientes puertos:

- 18083 para gestionar el bróker por medio de “dashboard”
- 1883 puerto para el protocolo MQTT
- 8083 puerto para webSocket y MQTT
- 8080 puerto para utilizar la API por medio del protocolo HTTP

Además, se especifica la colección de la base de datos que contiene credenciales para conectar a MQTT, todo el código se detalla en el anexo 8. Ingresando por el puerto 18083 se aprecia la interfaz de EMQX donde se puede gestionar a todos los “clients” conectados, tópicos entre otros, como se aprecia en la figura 46.

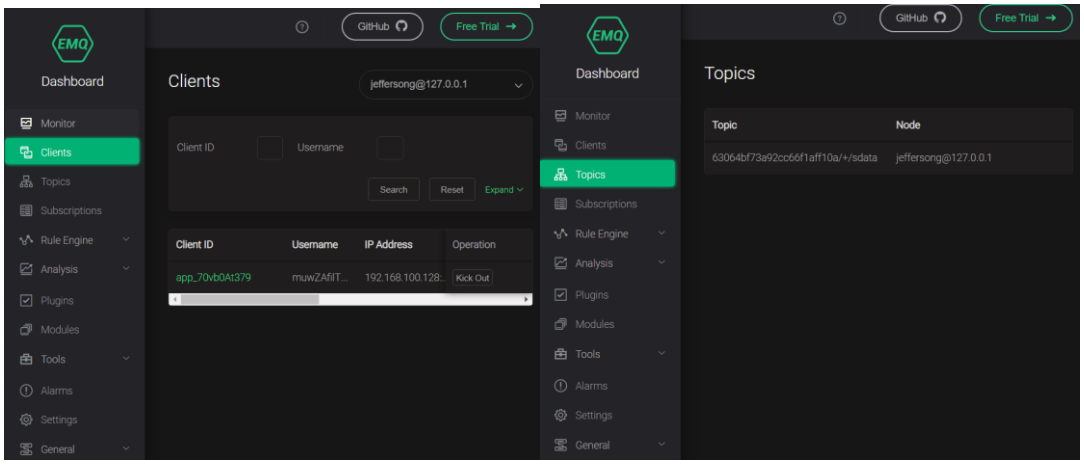


Figura 46.- Dashboard de contenedor EMQX
Elaborado por: El investigador

Finalmente se creó el contenedor de Nodejs mediante los "stacks" de portainer, donde se define la dirección para ejecutar los archivos de JavaScript con el puerto 3000 para escuchar las peticiones (anexo 9), la principal función es que sirva como un backend para trabajar con servicios REST. Para empezar a crear el backend a través de Nodejs es necesario instalar los módulos por medio del comando "*npm install"*, e instalar las dependencias de la tabla 28.

Tabla 28.- Dependencias necesarias para crear el backend

Dependencia	**Descripción**
Nodemon	Reinicia automáticamente la aplicación cuando detecta un cambio en el código
Cors	Otorga permisos para acceder a la API desde cualquier origen
Express	Es el framework de Nodejs
Dotenv	Carga variable de entorno desde el archivo .env
bcryptjs	Para encriptar contraseñas
Express-validator	Usado para validar las credenciales del usuario, en caso de no ser correctas devuelve un error
jsonwebtoken	Para generar tokens para la autenticación del usuario
Mongoose	Sirve para conectar el backend con la base de datos (MongoDB)
Firebase admin	Permite acceder con todos los privilegios de usuario a los servicios de firebase como servidores o a la nube.
Axios	Para hacer peticiones HTTP desde Nodejs

Elaborado por: El investigador

Al terminar la instalación de las dependencias se puede apreciar todas en el archivo raíz “packeg.json” con su respectiva versión, como se muestra en la figura 47.

Figura 47.- dependencias instaladas en Nodejs
Elaborado por: El investigador

Las principales funciones que se ejecutan dentro del “backend” para ofrecer utilidad en el “frontend” (aplicación móvil) son:

- Autenticación de usuarios
- Conectar al bróker MQTT
- Crear reglas en EMQX por usuario
- Enviar notificaciones push a la aplicación móvil
- Guardar los datos adquiridos por los sensores en MongoDB

A continuación, se detalla cada función del backend

Autenticación de usuarios

Esta función permite que solo usuarios registrados puedan acceder a la aplicación móvil controlando el acceso por medio de un correo electrónico, contraseña y nombre de usuario, esto es posible gracias a la dependencia de “express-validator” encargada de validar cada uno de los campos requeridos, para crear un usuario e iniciar sesión cada uno tiene “APIs” diferentes.

Tabla 29.- APIs para crear usuario e iniciar sesión

Concepto	API
Nuevo usuario	192.168.100.160:3000/api/login/new
Iniciar sesión	192.168.100.160:3000/api/login/

Elaborado por: El investigador

El proceso general para registrar nuevos usuarios e iniciar sesión se detalla mediante el diagrama de flujo de la figura 48. En el proceso de registrar un usuario, el correo y

el nombre de usuario son almacenados de forma plana mientras que la contraseña es encriptada por medio de la dependencia de bcryptjs y así almacenada de forma segura. Como respuesta al hacer la petición POST a cualquier API viene el token, el id y los datos del usuario en formato json.

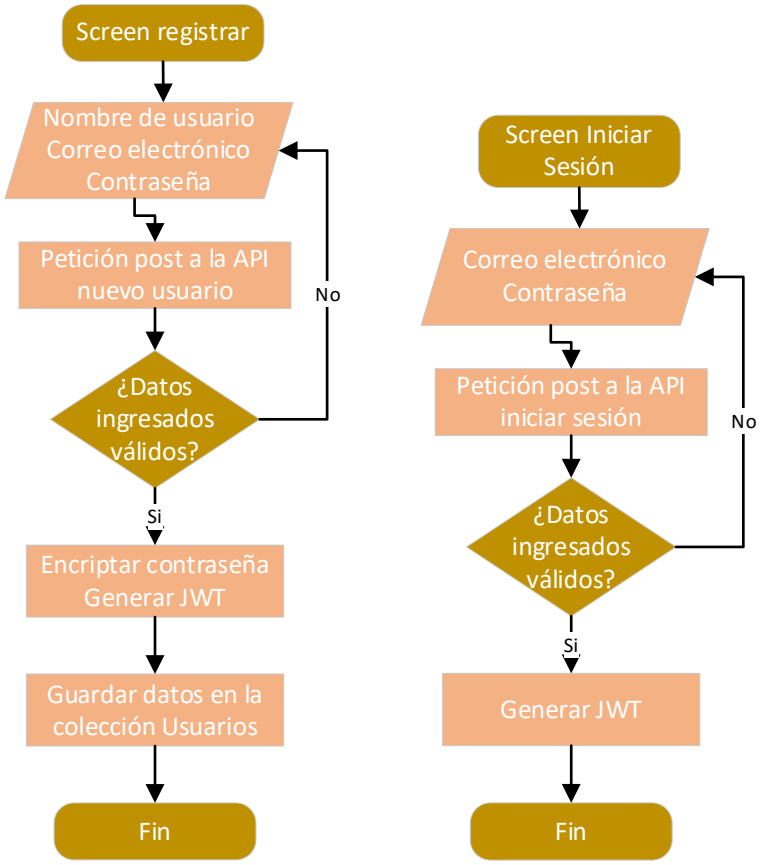


Figura 48.- Diagrama de flujo para registrar usuarios e iniciar sesión
Elaborado por: El investigador

Conectar al bróker MQTT

Para brindar seguridad al bróker MQTT es necesario autenticar por medio de usuario y contraseña, para esto la ESP32 o el smartphone deben iniciar sesión para acceder a la API *192.168.100.160:3000/api/login/find* y requerir las credenciales (usuario y contraseña MQTT) y el topic para conectar al bróker MQTT como muestra la figura 49.

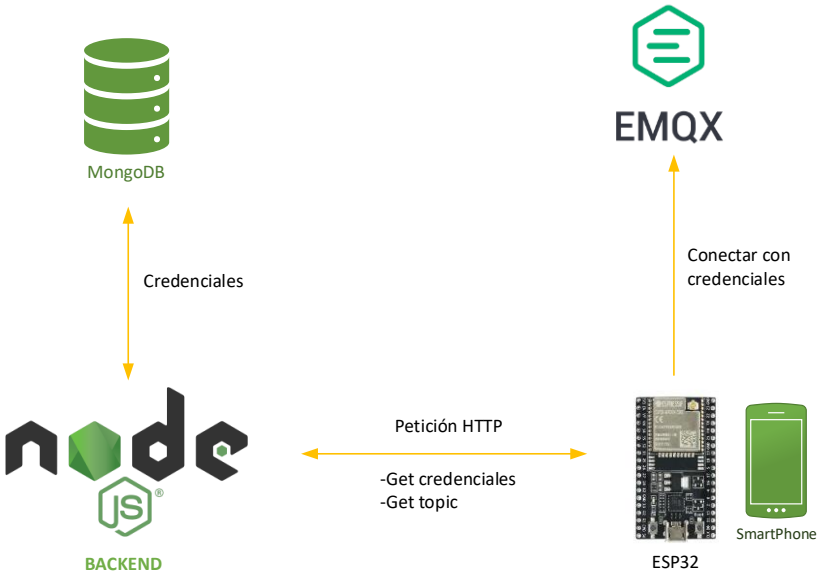


Figura 49.- Proceso para gestionar credenciales del bróker MQTT
Elaborado por: El investigador

La petición POST a la API *192.168.100.160:3000/api/login/find* se detalla en el diagrama de flujo de la figura 50. Las credenciales para MQTT que proporciona la petición son creadas de forma aleatoria, justo en el momento de registrar un nuevo usuario, las credenciales son almacenadas en la base de datos en la colección proyect/emqxauthrules, de esta forma el usuario puede conectar al bróker solamente estando autenticado.

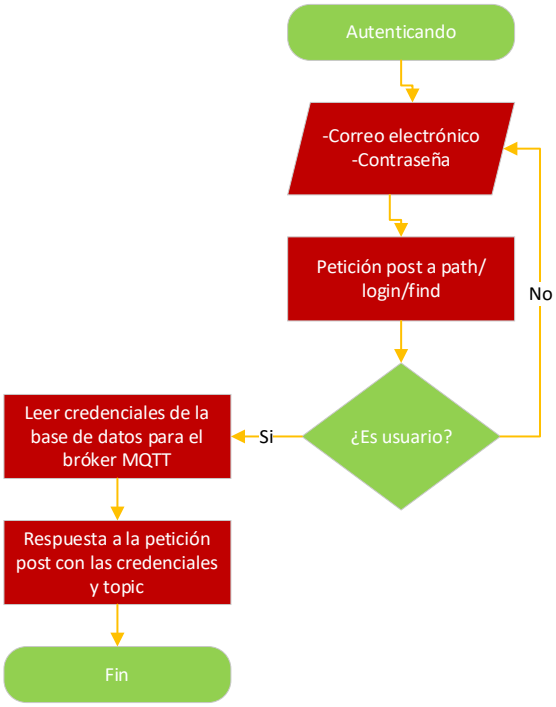


Figura 50.- Diagrama de flujo para requerir credenciales de EMQX al backend
Elaborado por: El investigador

Crear reglas en EMQX por usuario

EMQX dispone de un motor de reglas utilizado para gestionar el flujo de mensajes; es decir, cuando un mensaje cumpla con la condición escrita por una regla, esta ejecutará sus propias declaraciones en SQL para activar, filtrar y procesar la información de los mensajes y realizar un evento en el backend por medio de la API declarada.

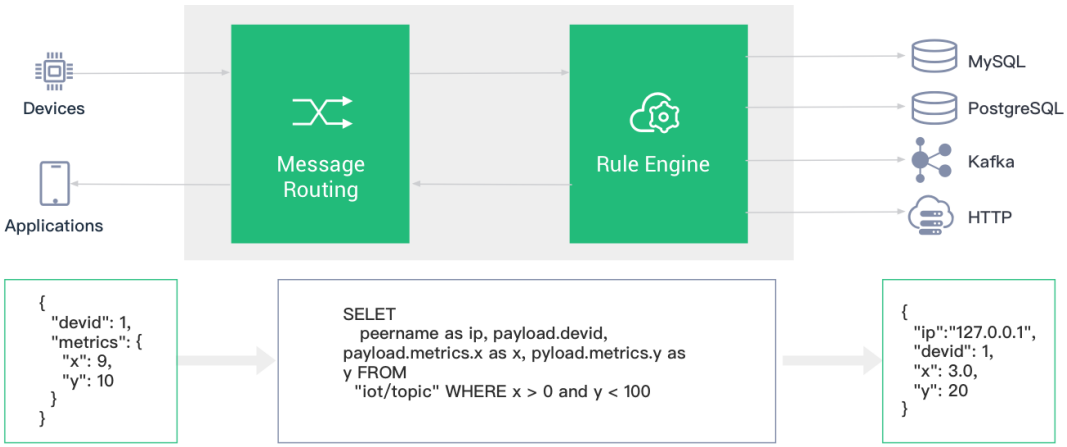


Figura 51.- Motor de reglas de EMQX *[55]*

Al coincidir los mensajes con una regla de EMQX, esta se comunica con el backend, para esta comunicación es necesario crear un resource de tipo webhook para declarar una API y parámetros que debe contener, como se aprecia en la figura 52.

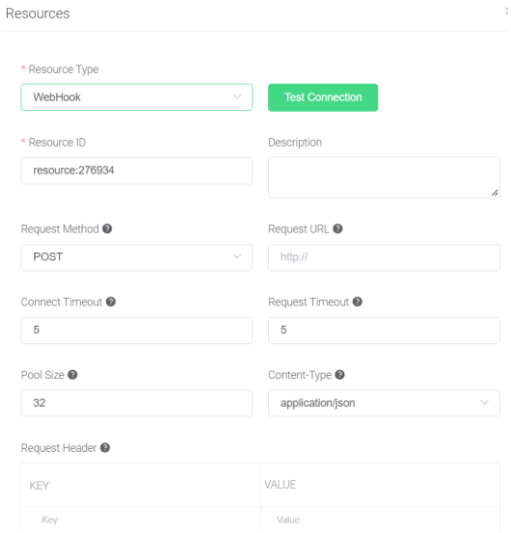


Figura 52.- Parámetros de resource para declarar API de comunicación con el backend
Elaborado por: El investigador

Para crear un "resource" en la regla se utilizó de la API integrada de EMQX mediante el método POST por *http:localhost/api/v4/resources.* Se escribió código en un documento de JavaScript con todos los parámetros de la figura 52 en formato json, de esta forma el resource se crea por medio del backend, además como método de seguridad se declara un token como se detalla a continuación:

```
  const url = "http://"+ process.env.EMQX_API_RULE
1 +"/api/v4/resources"
2 const data1 = {
3   "type": "web_hook",
4   "description": "alarm-webhook",
5   "config": {
6      url: "http://"+ process.env.LOCALHOST + ":" +
7 process.env.PORT +"/api/login/alarm-webhook",
8      method: "POST",
9      headers: {
10      token: "A4xW0$1c56gR3T!kllI09ZX#31"
11            },
12        }
13        }//data1
14 const res1 = await axios.post (url, data1, auth);
15 if (res1.status === 200){
16 console.log('!!!alarm creado!!!');
  }
```

Se utilizó el código anterior para crear 2 resources de esta forma la regla podrá comunicar con el banckend por medio de un endpoint. Cada "resource" tiene diferentes endpoint donde alarm-webhook está encargada de alertar o enviar notificaciones a la

aplicación móvil mientras que saver-webhook almacena los datos de los sensores en la base de datos.

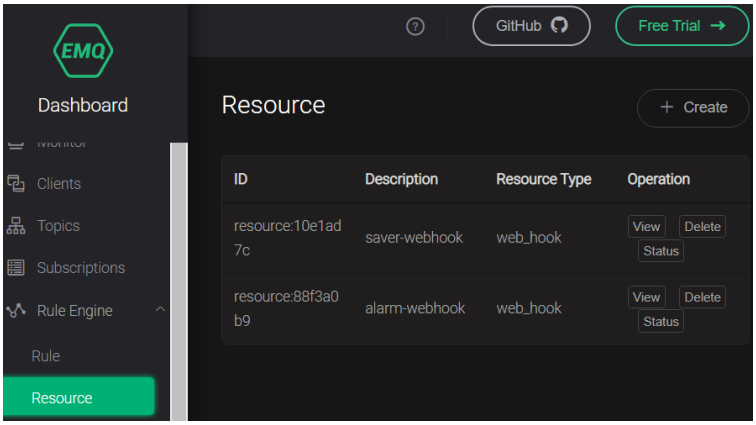


Figura 53.- Resources creada por medio de la API de EMQX
Elaborado por: El investigador

De forma similar se crea una regla usando la API integrada de EMQX por medio del método POST a través de *http:localhost/api/v4/rules*. Donde se especifica el topic y la condición mediante el "payload" escrito en SQL como se especifica a continuación:

```
const url = "http://"+ process.env.EMQX_API_RULE
+"/api/v4/rules";

const topic = userId + "/+/sdata";
const rawsql ="SELECT topic, payload FROM \"" + topic + "\" WHERE
payload.alarm = 1";

 var newRule = {
     rawsql: rawsql,
     actions: [
          {
           name: "data_to_webserver",
           params: {
                $resource: global.alarmResource.id,
                payload_tmpl: '{"userId": "' + userId +
'","payload":${payload}, "topic":"${topic}"  }'
             }
        }
    ],
    description: "SAVER-RULE",
    enabled: status
    };
const res = await axios.post(url, newRule, auth);
```

Los "topics" de cada usuario están escritos de la forma: "*userId/+/sdata"*, donde userId es el Id de cada usuario registrado, mientras que el símbolo + en el "topic" permitirá crear diferentes "topics" en base al principal, de esta forma se convierte en

un sistema dinámico, además tiene la capacidad de crear reglas para cada usuario de forma automática. Además, se creó una regla que detecta mensajes con la variable $saver = 1$ para proceder almacenar los datos en la base de datos, y otra regla que detecta la variable $alarm = 1$ para notificar a la aplicación móvil, como muestra la figura 54.

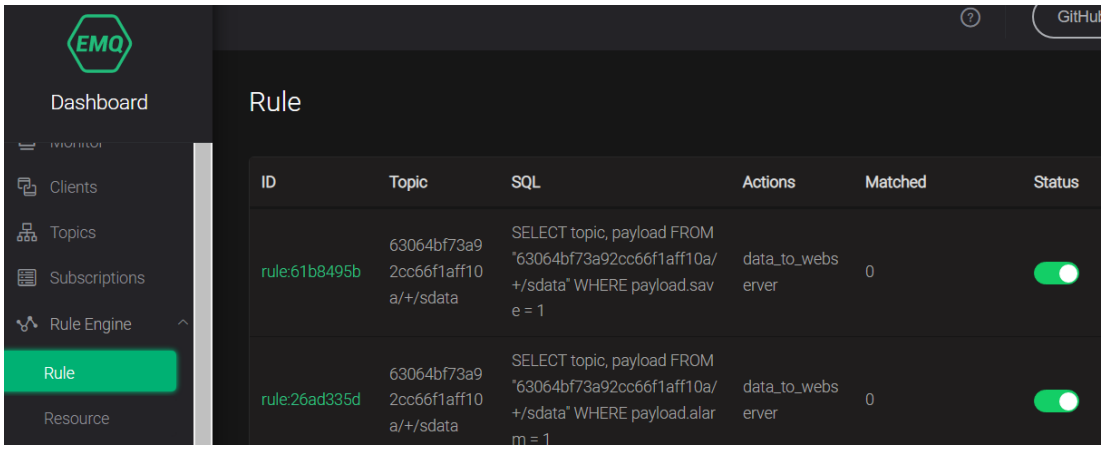


Figura 54.- Regla creada por la API integrada de EMQX
Elaborado por: El investigador

Enviar notificaciones push a la aplicación móvil

Para que el sistema tenga la capacidad de enviar notificaciones a la aplicación móvil debe existir una regla en EMQX con características de alerta, esto fue descrito anteriormente. El funcionamiento de la regla para enviar notificaciones push se basa en detectar un mensaje en el "payload" con la variable $alarm = 1$, EMQX rápidamente envía el contenido del mensaje por medio de la API registrada en "resource", en este caso por medio de *http:/localhost/ api/login/alarm-webhook*, en el cuerpo del mensaje llega el "topic" y el valor de la variable como se indica en la figura 55.

```
userId: '63064bf73a92cc66f1aff10a',
payload: { value: 36, alarm: 1 },
topic: '63064bf73a92cc66f1aff10a/temp/sdata'
```

Figura 55.- Cuerpo del mensaje de EMQX
Elaborado por: El investigador

Una vez que llega los datos al "backend" este procede a determinar que variable va a alertar a la aplicación móvil y a que usuario se debe enviar la notificación, para esto se

guardó en la base de datos el token que proporciona el dispositivo móvil al instalar la aplicación, este token será único para cada smartphone, entonces, con el fin de alertar a todos los usuarios que instalen la aplicación y se autentifiquen bajos las misma credenciales del dispositivo IoT podrán ser notificadas por este dispositivo, de esta forma se pretende alertar a todos los cuidadores del adulto mayor. El funcionamiento de envío de notificaciones se hace como el diagrama de flujo de la figura 56.

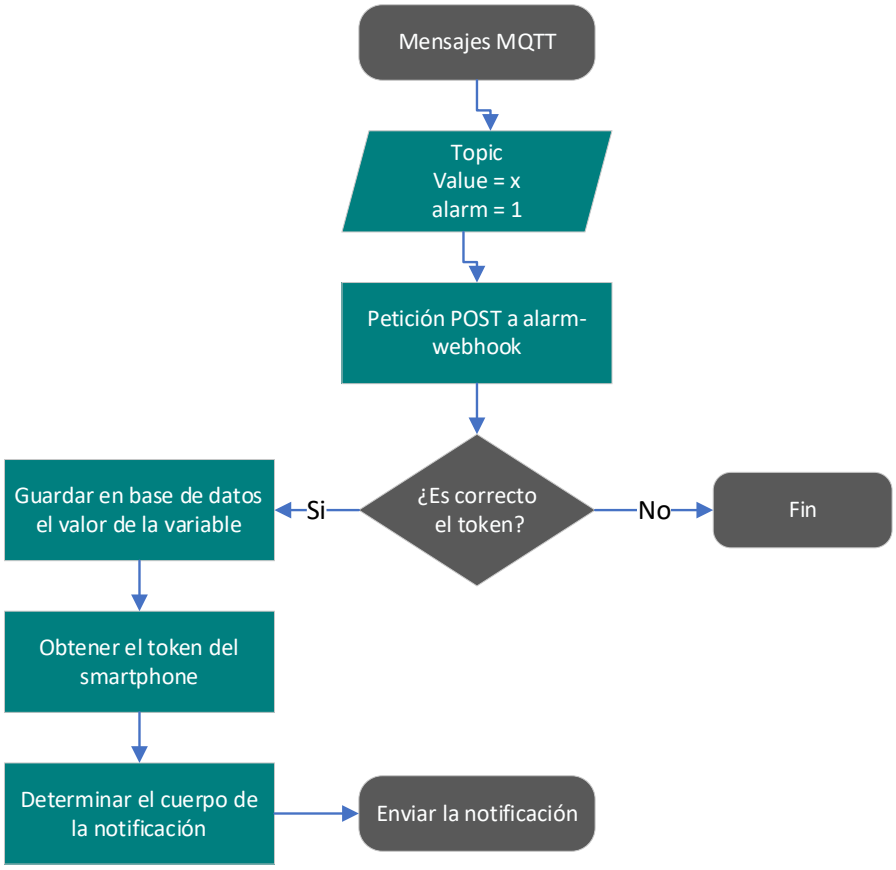


Figura 56.- Diagrama de flujo para enviar notificaciones al smartphone
Elaborado por: El investigador

Una vez que el valor de la variable se guarda en la base de datos el "backend" busca en la colección *proyect/deviceapps* los tokens almacenados al instalar la aplicación en el smartphone con el fin de enviar a todos los usuarios registrados una notificación push, el cuerpo de la notificación dependerá de que variable está sobrepasando los rangos normales como se detalla en la tabla 30.

Tabla 30.- Cuerpo de la notificación según el tipo de variable alterada

Variable	Contenido del cuerpo de la notificación
status	Se detectó una caída
Temp	Temperatura corporal elevada
Heart	Frecuencia cardiaca alterada
spo2	Saturación de oxígeno inestable
Battery	Batería baja

Elaborado por: El investigador

Para enviar notificaciones se usó la infraestructura de Firebase mediante FCM (Firebase Cloud Messaging) esta arquitectura es gratuita, se puede enviar notificaciones ilimitadas sin costo. Para lo cual se crea un proyecto en *https://console.firebase.google.com* y se registra la aplicación llenando todos los parámetros que contiene la figura 57.

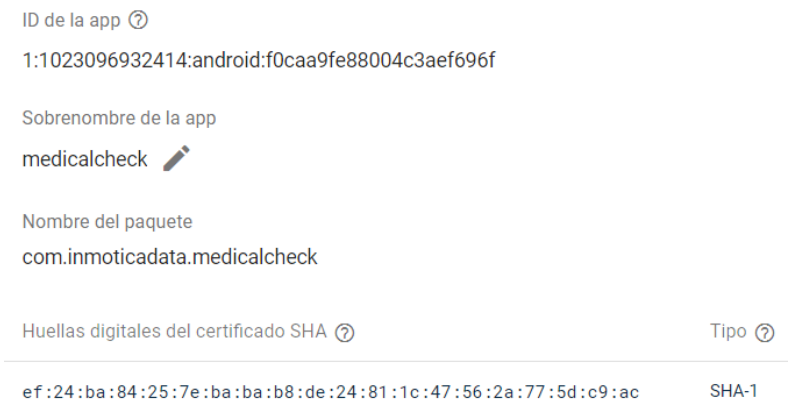


Figura 57.- Registrar aplicación en Firebase
Elaborado por: El investigador

Todos los parámetros son propios de la aplicación móvil, en este caso el sobrenombre de la aplicación debe ser único; es decir, no debe coincidir con nombres de otras aplicaciones, el nombre del paquete se obtiene de flutter en la carpeta raíz mediante la ruta *android/app/build.gradel*. mientras que, para determinar el certificado SHA-1 se ejecuta por consola el comando *cd android && ./gradiew signingReport*, la salida se aprecia en la figura 58.

```
Variant: profileUnitTest
Config: debug
Store: /Users/strider/.android/debug.keystore
Alias: AndroidDebugKey
MD5: A0:AF:1B:88:EB:B5:1D:89:EB:41:2A:C1:0E:66:B6:85
SHA1: 1D:4F:38:94:62:06:9F:C6:75:A7:73:BD:E4:B0:DA:27:80:B1:9D:F0
SHA-256: C2:17:27:53:07:3D:0E:58:47:45:AE:19:C3:B6:40:24:CB:0F:72:6E:E0:E9:4E:58:D9:E0:56:E0:6B:22
:8C:39
Valid until: Tuesday, December 7, 2049
```

Figura 58.- Salida al ejecutar el comando para crear el SHA-1
Elaborado por: El investigador

El introducir todos los parámetros, Firebase genera un archivo.josn para autenticar los dispositivos a FCM, este archivo se descarga y se coloca en la carpeta raíz del proyecto de Nodejs. Los parámetros del archivo .json son generados por la misma aplicación de Firebase con credenciales únicas.

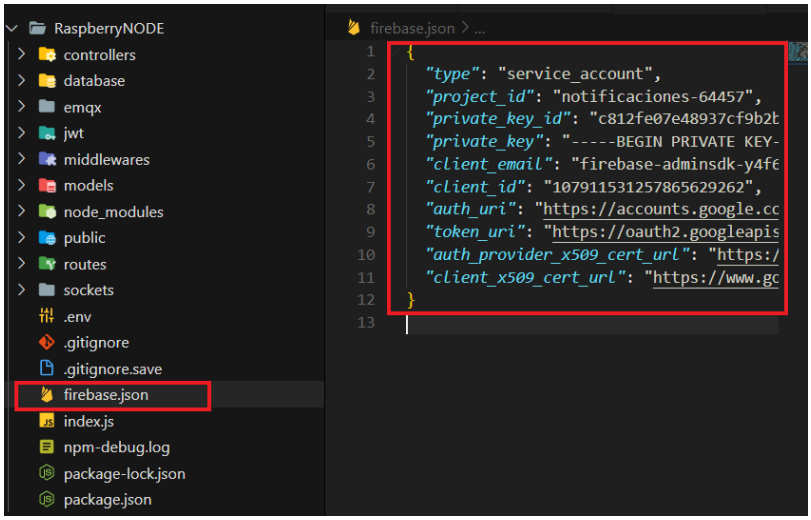


Figura 59.- Parámetros del archivo de autenticación
Elaborado por: El investigador

Guardar los datos adquiridos por los sensores en MongoDB

Todos los datos adquiridos por la ESP32 son almacenados en MongoDB, esto es posible por la regla que se creó con anterioridad en EMQX, cuando la variable $saver = 1$ EMQX hace una petición http al backend por el método POST mediante *http:localhost/api/login/saver-webhook*, el cuerpo de la petición lleva el estado de la variable, el Id del usuario y el topic, como muestra la figura 60.

```
{
  userId: '63064bf73a92cc66f1aff10a',
  payload: { save: 1, value: 36 },
  topic: '63064bf73a92cc66f1aff10a/temp/sdata'
}
```

Figura 60.- Body de la petición http con el método POST al backend
Elaborado por: El investigador

Para almacenar los datos de los sensores se tiene diferentes colecciones en la base de datos, de esta forma se obtiene los datos clasificados, esto se detalla en la tabla 31.

Tabla 31.- Diferentes colecciones para almacenar los datos

Colección	Datos almacenados
datahearts	Almacena datos de frecuencia cardiaca
dataspo2	Almacena datos de saturación de oxígeno
datatemps	Almacena datos de temperatura corporal

Elaborado por: El investigador

El proceso para almacenar los mensajes en MongoDB se detalla por medio del diagrama de flujo de la figura 61.

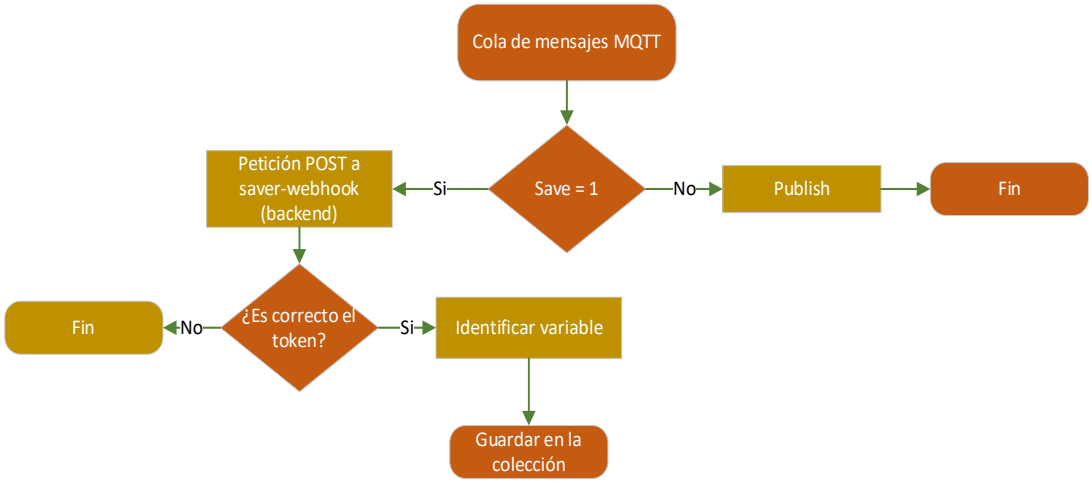


Figura 61.- Diagrama de flujo para almacenar los datos de los sensores en MongoDB
Elaborado por: El investigador

Entonces, el backend es el encargado de proporcionar APIs para almacenar los datos de EMQX y gestionar la aplicación móvil. Como método de seguridad se implementó un token proporcionado por la dependencia de JesonWebToken; es decir, cuando alguien desee hacer una petición esta pasa por un proceso de validación en el backend.

3.2.7 Capa de presentación

Esta capa es la encargada de mostrar en una interfaz gráfica todos los datos recopilados por los sensores, además recibe alertas para notificar al usuario por medio de sonidos o "screens". Para el desarrollo de la interfaz se utilizó de Flutter, los requisitos previos para instalar este "framework" en Windows son:

- Windows 10 o posterior de 64 bits
- Espacio en disco duro de 10 GB
- Android Studio
- Visual Studio Code
- Github

Al cumplir con los requisitos se descarga el paquete de instalación de Flutter en *https://docs.flutter.dev/get-started/install/Windows,* en la versión actual. Al finalizar la descarga se crea una carpeta con nombre "flutter" y se extrae el archivo .zip dentro de esta; para finalizar, en una interfaz de consola se ejecuta el comando para instalación:

```
C:\flutter> git clone https://github.com/flutter/flutter.git -b stable
```

Para verificar si la instalación se hizo correctamente se ejecuta el comando "*flutter doctor"* cuya salida debe ser similar a la figura 62.

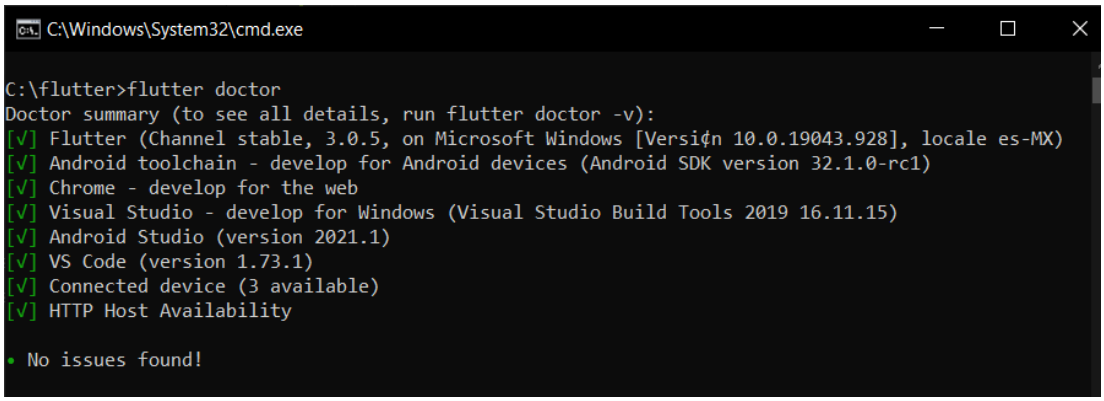


Figura 62.- Verificar instalación de Flutter
Elaborado por: El investigador

Para crear un emulador para la aplicación móvil se hace mediante el software Android Studio, en el aparatado de "device manager" creando un nuevo dispositivo y configurando con los parámetros de la figura 63.

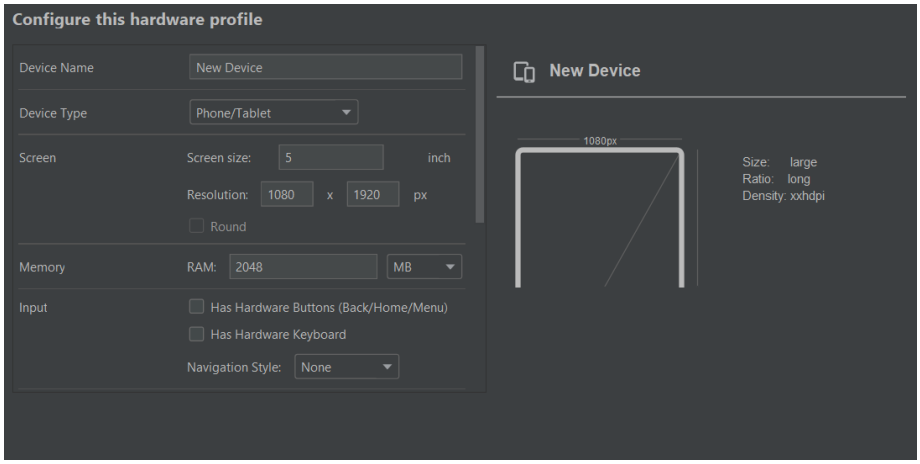


Figura 63.- Parámetros para crear un nuevo dispositivo
Elaborado por: El investigador

El desarrollo de la aplicación comienza con la creación de un nuevo proyecto de flutter en Visual Estudio Code, como resultado se crea un paquete de carpetas y archivos, la carpeta lib contiene todo el código para ejecutar la aplicación, mientras que el archivo "pubspec.yaml" lleva todas las dependencias y versiones utilizadas, como muestra la figura 64.

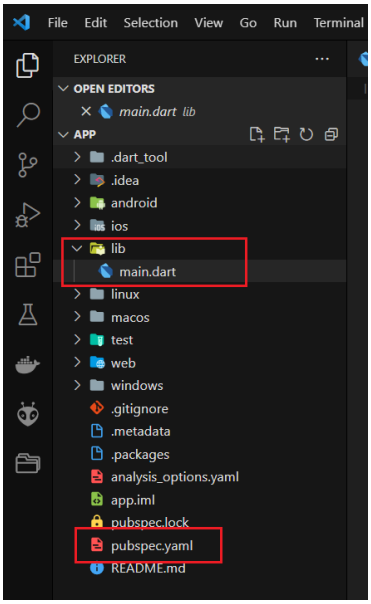


Figura 64.- Paquetes para iniciar el desarrollo de la aplicación
Elaborado por: El investigador

Las dependencias utilizadas para el desarrollo óptimo de la aplicación se detallan en la tabla 32, que se descargan mediante el comando *flutter pub add "dependencia"*, y de forma automática son almacenadas dentro del archivo "pubspec.yaml"

Tabla 32.- Dependencias para el desarrollo de la aplicación

Dependencia	Concepto
Cupertino icons	Este es un repositorio de archivos de iconos, utilizados por los widgets.
Flutter secure storage	Es un complemento para almacenar los datos en un almacenamiento seguro, contiene cifrado AES
http	Es una biblioteca para facilitar el consumo de algunos recursos HTTP.
Provider	Es un gestor de estados que permite simplificar recursos en el desarrollo
Mqtt client	Es un cliente mqtt que permite hacer suscripción y publicación en todos los niveles de Qos.
Sync function chart	Es un paquete que permite hacer graficas de datos
Timezone	Es un recurso que proporciona la fecha y hora desde internet.
Firebase core	Este complemento permite conectar con las diferentes APIs de Firebase
Firebase messanging	Permite conectar con la API de Firebase Cloud Messaging (FCM), para recibir notificaciones.

Elaborado por: El investigador

Las principales funciones de la aplicación móvil son:

- Registro y login de usuarios
- Presentación de los datos
- Crear alertas de los signos vitales
- Información del usuario
- Recibir notificaciones

A continuación, se detalla cada una de las funciones de la aplicación

Registro y login de usuarios

Iniciando por el registro de los usuarios es necesario implementar el widget textfiel, esta es una caja para introducir texto y ser almacenado en un controlador de texto,

además se incorpora avisos para identificar cada espacio en blanco y un icono de reconocimiento, la interfaz gráfica se puede apreciar en la figura 65.

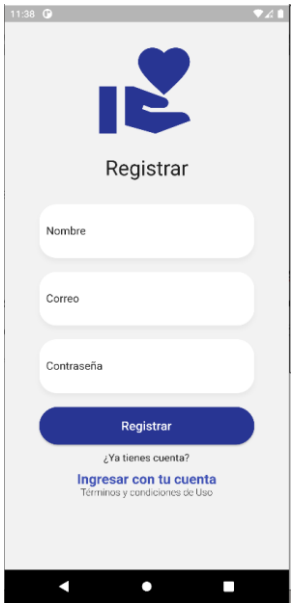


Figura 65.- Screen para registrar nuevos usuarios
Elaborado por: El investigador

En la parte inferior se incorpora un botón (Registrar) para recopilar los datos del controlador de texto y hacer una petición por el método POST al "backend". La interfaz registra un nuevo usuario solamente si se llena todos los campos y el formato del correo electrónico es válido, caso contrario la interfaz presenta una alerta para que el usuario tenga en cuenta todos los parámetros de registro.

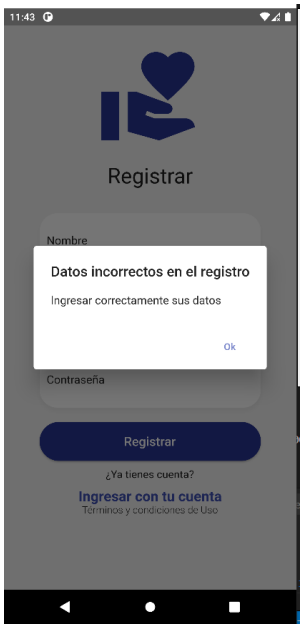


Figura 66.- Alerta para los datos de registro
Elaborado por: El investigador

La interfaz gráfica login es similar al registro, aquí el usuario debe ingresar con los datos registrados, de igual manera tiene incorporado un botón (Ingresar) para hacer una solicitud al backed, al ingresar el correo y la contraseña la aplicación hace una solicitud al backend, esta responde con los datos encontrados en la base de datos, en caso de no encontrar coincidencia con las credenciales la aplicación lanza rápidamente una alerta, manifestando que las credenciales ingresadas fueron incorrectas, como muestra la figura 67.

Figura 67.- Screen de login y alerta
Elaborado por: El investigador

Cuando el interesado ha registrado correctamente un usuario o a ingresado con los datos validados por el “backed”, la aplicación automáticamente pasa a la pantalla principal (home screen), para mostrar los datos de los sensores.

Presentación de los datos

Por medio de tarjetas se hace la presentación del valor actual de temperatura corporal, saturación de oxígeno y frecuencia cardiaca, para obtener estos parámetros la aplicación hace una solicitud al “backend” pidiendo las credenciales para conectar al bróker MQTT, cuando esta suscrita al “topic”, por medio de “provider” (gestor de estados) los datos nuevos son notificados a cada tarjeta para actualizar los datos.

En la pantalla se grafican los datos de forma de historial; es decir, cada punto está determinado por el valor y la fecha que fueron ingresados, esto se hizo por medio de la dependencia “Sync function chart”. En la figura 68 se presentan las tarjetas para mostrar los datos del brazalete donde:

- La primera tarjeta sirve para mostrar notificaciones entrantes a la aplicación.
- Las tarjetas estabilidad, temperatura, frecuencia cardiaca y pulsioxímetro sirven para mostrar los valores de los sensores respectivamente.
- Los gráficos dibujan una línea de tiempo, el punto esta definido por el valor de cada signo vital vs (tiempo) la hora que fueron grabados.

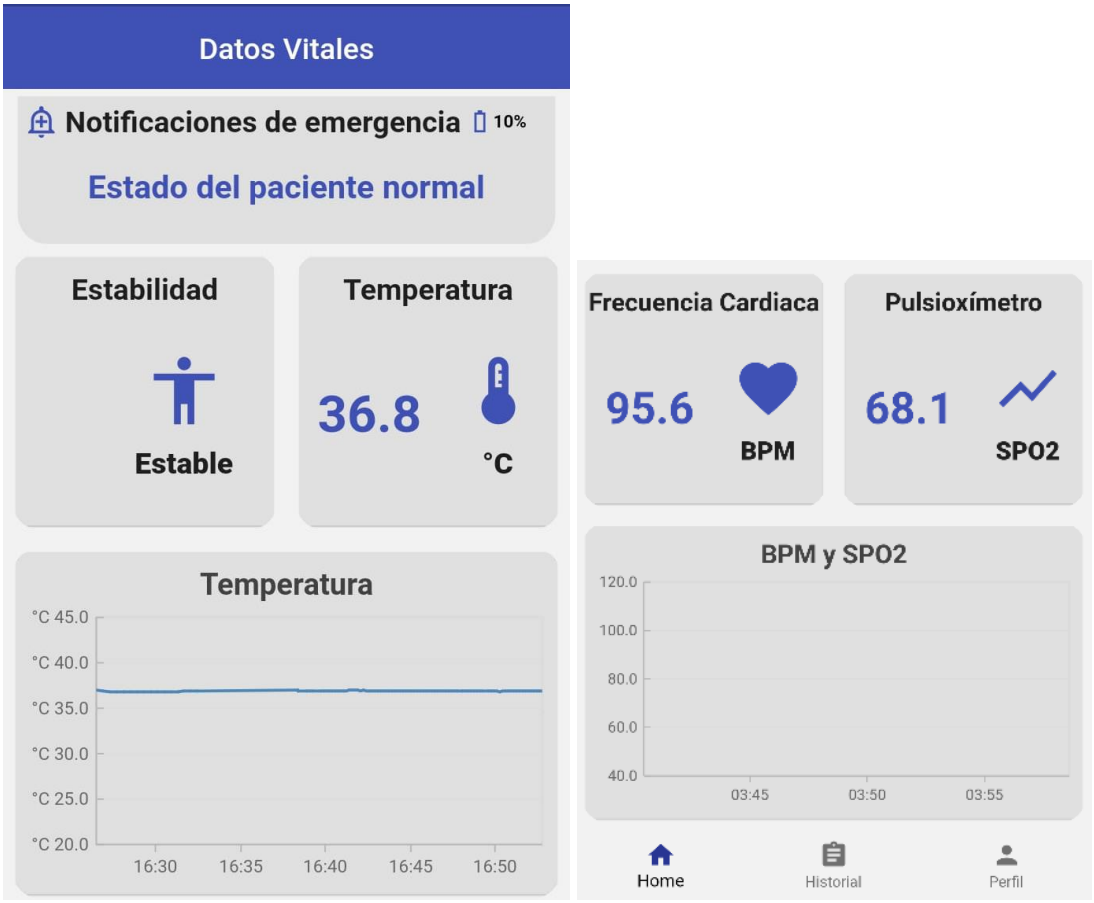


Figura 68.- Screen para presentar los datos de los sensores
Elaborado por: El investigador

En la parte inferior de home screen se implementa el widget “bottom navigation bar”, para manejar tres pantallas:

- Home: es la pantalla principal de la aplicación,
- Historial: presenta el estado de los signos vitales en forma de un historial clínico.
- Perfil: lleva la información del usuario.

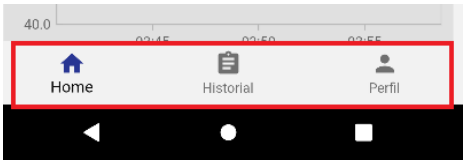


Figura 69.- Bottom navigation bar con diferentes menús
Elaborado por: El investigador

Al usar un “bottom navigation bar” es necesario manejar el estado de la aplicación para lo cual se utilizó provider, el mismo que debe ser declarado en el archivo principal

en este caso es main.dart (figura 64) ya que la aplicación empieza ejecutando este archivo. A continuación, se detalla las funciones principales de provider:

- AuthService: para realizar la conexión con el backend
- BarProvider: para controlar el estado de "bottom navigation bar"
- AuthMqtt: para gestionar los mensajes de MQTT y presentar en el screen principal
- DataChartService: se utiliza para gestionar las gráficas de los datos
- MedicalHistoryServices: para conectar con la base de datos
- NewAlertServices: para crear alertas nuevas por medio del screen nueva alerta.

```
@override
Widget build(BuildContext context) {
 return MultiProvider(
   providers: [
     ChangeNotifierProvider(create: (_) => AuthService()),
     ChangeNotifierProvider(create: (_) => BarProvider()),
     ChangeNotifierProvider(create: (_) => AuthMqtt()),
     ChangeNotifierProvider(create: (_) => DataChartService()),
     ChangeNotifierProvider(create: (_) =>
MedicalHistoryServices()),
     ChangeNotifierProvider(create: (_) => NewAlertService()),
      ],
```

En este punto la aplicación es capaz de suscribir al topic *userId/+/sdata* donde userId es el Id de cada usuario, mientras que el símbolo + permite registrar diferentes variables, en este caso la aplicación es capaz de suscribir a los topics de la tabla 33.

Tabla 33.- Topics para la aplicación móvil

Topic	Datos de recopilación
userId/temp/sdata	Lee los datos de temperatura corporal del paciente
userId/heart/sdata	Posee los valores de frecuencia cardiaca
userId/spo2/sdata	Obtiene los valores de saturación de oxígeno
userId/status/sdata	Determina la estabilidad del usuario
userId/battery /sdata	Determina el estado de la batería del dispositivo IoT

Elaborado por: El investigador

El diagrama de flujo de la pantalla principal se muestra en la figura 70.

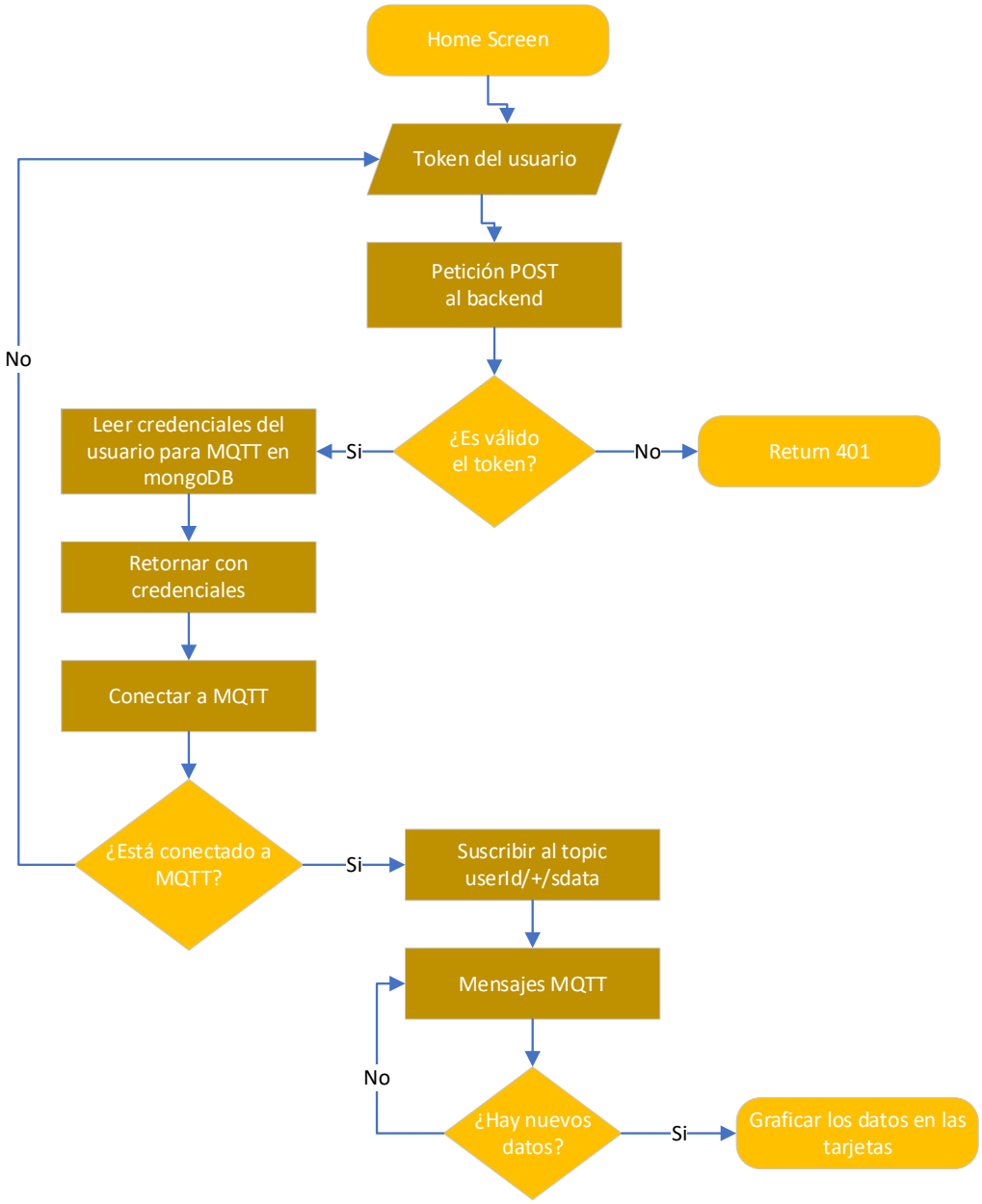


Figura 70.- Diagrama de flujo para conectar a MQTT y graficar en las tarjetas
Elaborado por: El investigador

Crear alerta de los signos vitales

Esta pantalla permite mostrar el historial de alertas creadas y recibidas por el usuario de la aplicación, esto ayuda a crear un tipo de historial médico dentro de la aplicación siendo de gran ayuda para registrar las enfermedades que tenía el adulto mayor y los valores de los signos vitales que tenían en ese momento.

El historial muestra todas las alertas recibidas, el orden se determina por la fecha que fueron creadas. La pantalla de la figura 71 contiene tarjetas para mostrar el estado de los signos vitales, esta información es obtenida desde el backend por medio de una

solicitud HTTP con el método GET, además posee un botón (Nueva Alerta) para crear alertas de acuerdo con la necesidad del usuario.

Figura 71.- Screen para detallar el historial del adulto mayor
Elaborado por: El investigador

La figura 72 muestra el diagrama de flujo para obtener el listado de alertas desde el backend y graficar en las tarjetas del historial.

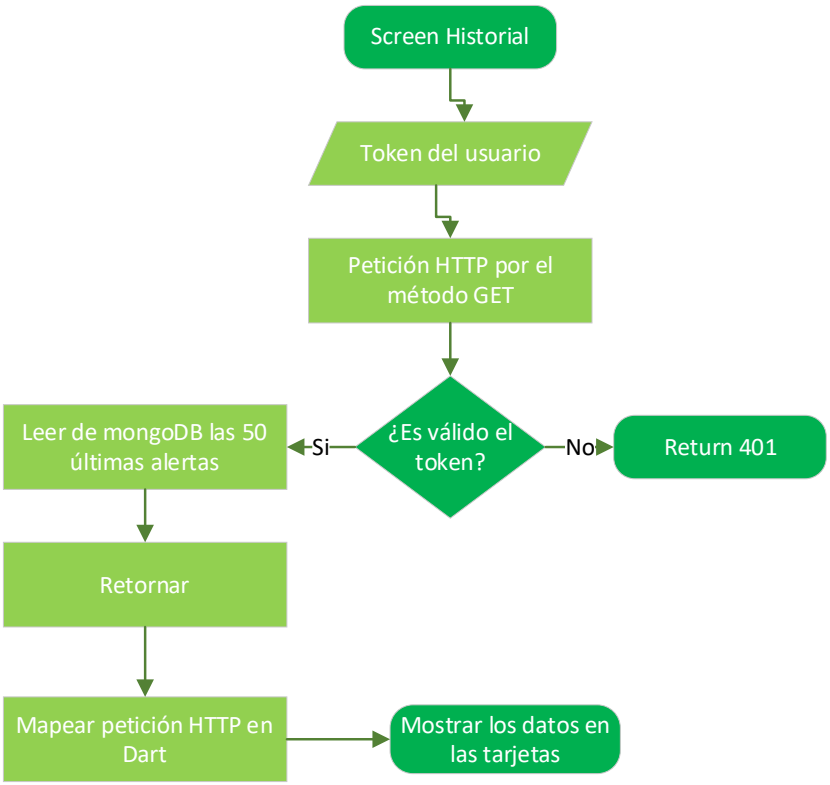


Figura 72.- Diagrama de flujo para obtener el historial de alertas
Elaborado por: El investigador

Al presionar el botón “Nueva Alerta” se crea la interfaz gráfica de la figura 73, esta permite crear alertas o eliminar en caso de existir. Para crear una nueva alerta el usuario debe activar el switch y seleccionar el intervalo de tiempo (cada 1, 4 u 8 horas) para recibir notificaciones, además es posible describir algún tipo de enfermedad que contiene el adulto mayor y el medicamente recetado por el doctor, todos estos datos serán presentados en la pantalla del historial (figura 71).

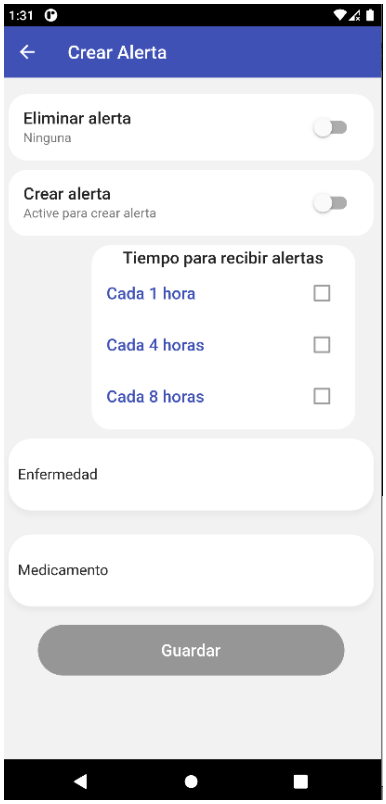


Figura 73.- Screen para crear nuevas alertas
Elaborado por: El investigador

Al guardar la alerta la aplicación realiza una petición POST al backend llevando toda la información insertada en la figura 73. El diagrama de flujos de nuevas alertas se describe en la figura 74.

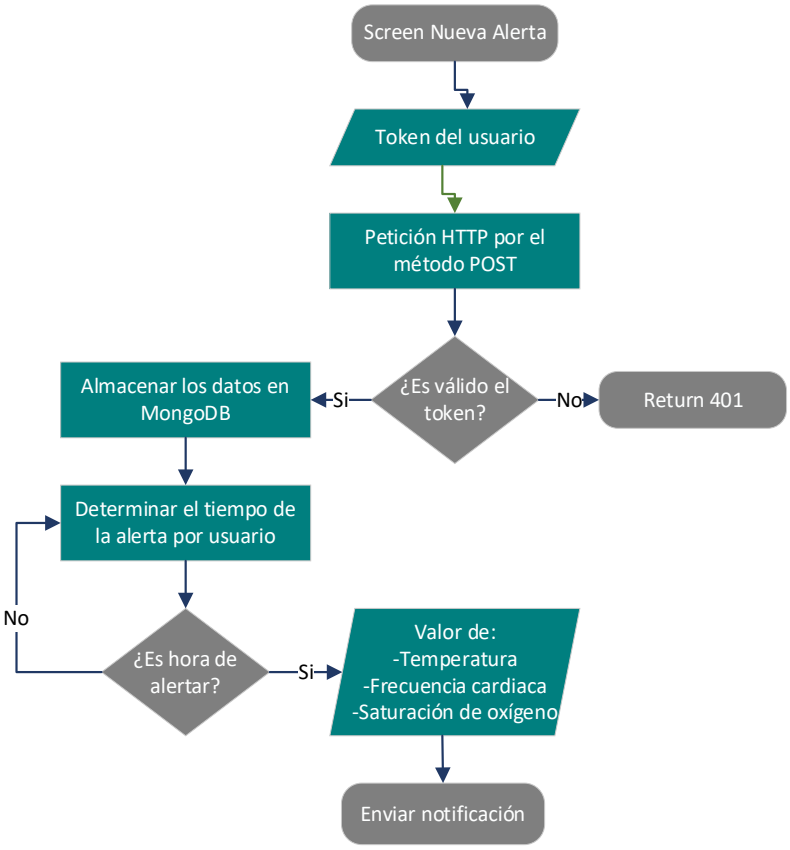


Figura 74.- Diagrama de flujo para crear una alerta y notificar el estado de los SV
Elaborado por: El investigador

Información del usuario

Cuando se inicia sesión en la aplicación, la respuesta de la petición HTTP se mapea mediante una clase para almacenar los datos en variables, de esta forma facilita el acceso a cada parámetro del usuario (anexo 10), en este caso se utilizó la variable email para almacenar el correo y la variable usuario para almacenar el nombre. Además, se implementó un botón el mismo que permite cerrar sesión en la aplicación, cuyo proceso es eliminar el token de “secure storage”, cuando la aplicación detecta este cambio automáticamente es dirigida a la pantalla de iniciar sesión.

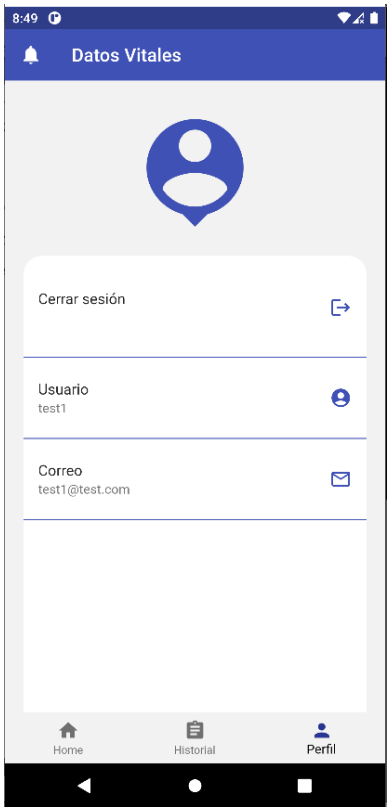


Figura 75.- Screen para mostrar información del usuario registrado
Elaborado por: El investigador

Recibir notificaciones

Al igual que el "backend" la aplicación móvil debe autenticar a Firebase Cloud Messaging (FCM), para esto se utilizó el archivo en formato json descargado desde FCM (usado en el backend) y se coloca dentro de la carpeta raíz *android/app*, como muestra la figura 76.

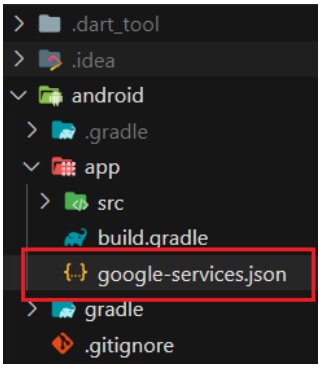


Figura 76.- Ubicación del archivo para autenticar a FCM
Elaborado por: El investigador

Las notificaciones son las encargadas de avisar al cuidador del adulto mayor sobre el estado de los signos vitales o la estabilidad, por este motivo mientras el cuidador reciba

todas las alertas posibles sin importar el estado de la aplicación será más confiable el sistema, entonces la aplicación puede recibir notificaciones en tres tipos de estados, esto se detalla en la tabla 34.

Tabla 34.- Estados de la aplicación para recibir notificaciones

Estado de la aplicación	Descripción
Terminada	Es posible recibir la notificación cuando la aplicación está cerrada por completo o el teléfono está bloqueado.
Fondo	Cuando la aplicación está ejecutando en segundo plano o esta minimizada.
Primer plano	Cuando la aplicación está abierta; es decir, cuando está siendo usada.

Elaborado por: El investigador

En la figura 77 se muestra los estados de la aplicación descritos anteriormente, cuando está en primer plano notifica mediante un widget snackbar para dibujar un recuadro en la parte inferior de la aplicación.

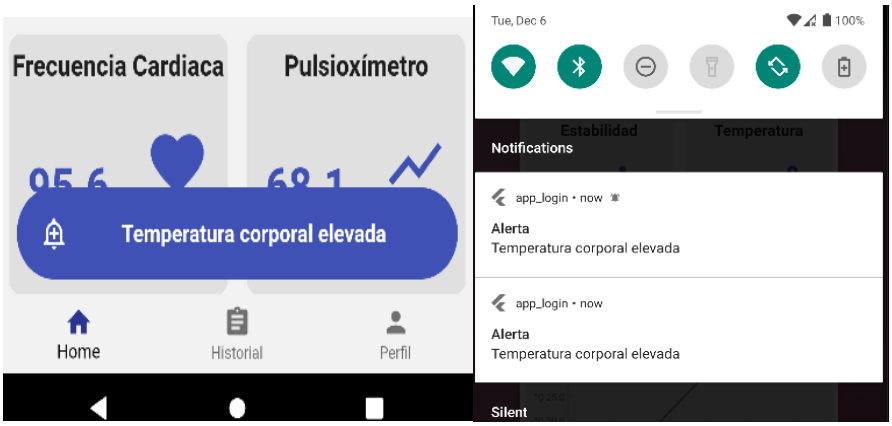


Figura 77.- Estado de la aplicación en primer plano y fondo
Elaborado por: El investigador

Al instalar la aplicación en un celular, Firebase Messaging crea un token único para cada dispositivo, este token es enviado al backend y almacenado en mongoDB en la colección *proyect/deviceapps*, de esta forma cuando se rompa la regla de EMQX se notifica al propietario del token.

3.2.8 Case para incorporar todos los elementos del dispositivo IoT

Para empezar con el diseño de una estructura para incorporar todos los elementos es necesario hacer un análisis de la ubicación de los sensores, para tener un mejor rendimiento del dispositivo, para la frecuencia cardiaca y saturación de oxígeno se ubica al sensor en la arteria radial mientras que el sensor de temperatura se coloca en la parte superior de la muñeca por la arteria interósea, con estos datos se procede a tomar medidas de los sensores, detallado en la tabla 35.

Tabla 35.- Tamaño de los componentes electrónicos incorporados en el brazalete

Componente electrónico	Tamaño (mm)
ESP32	26x18x3.2
MAX30100	5.6x2.8x1.2
MPU6050	20x16x2.5
MLX90614	16x10x4.1
TP4056	25x13x2.1
Batería	50x35x3.5

Elaborado por: El investigador

El primer paso fue la distribución de los sensores sobre la batería ya que esta es la que posee un tamaño superior a los demás elementos. El sensor MLX90614 debe tener contacto con la muñeca para detenerminar la temperatura corporal, se colocó el sensor de temperatura, microcontrolador, acelerómetro y módulo de carga de bajo de la batería, para optimizar espacio y tener contacto directo con la muñeca, como muestra la figura 78.

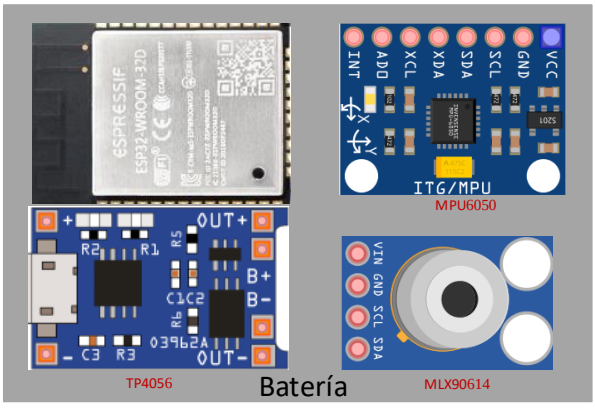


Figura 78.- Distribución de los elementos
Elaborado por: El investigador

El sensor MAX30100 debe ir fuera de la estructura para obtener la frecuencia cardiaca mediante la arteria radial, entonces lo óptimo fue colocar dentro de la correa que asegura al case en la muñeca. Teniendo en cuenta todos los criterios se procede a la construcción del case en el software CAD 3D Inventor ya que brinda herramientas potentes basadas en reglas para el diseño de estructuras de tres dimensiones, la versión que se utilizó fue 2019.

Siguiendo las dimensiones de la tabla 35 se diseñó una estructura en tres dimensiones, dando formas redondeadas para mejorar la presentación, además se realizó los siguientes criterios en el diseño:

- Corte para el periférico de carga
- Corte para el sensor MLX90614
- Pasador para la correa

La figura 79 muestra la estructura final del case, con todos los criterios descritos.

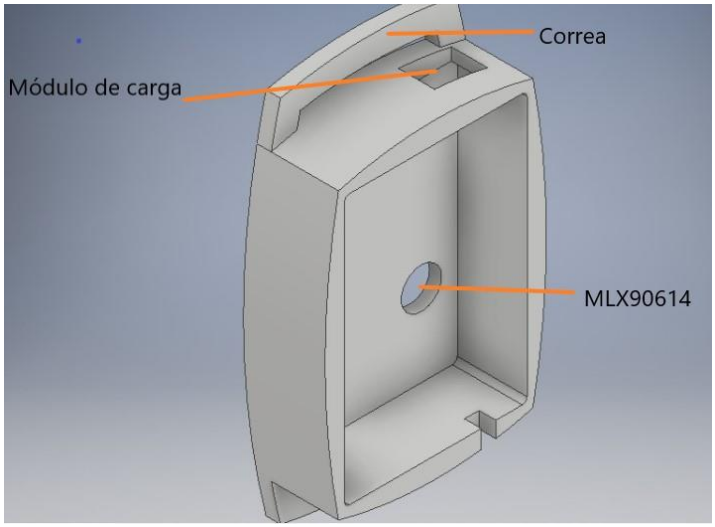


Figura 79.- Diseño del case en 3D
Elaborado por: El investigador

Para terminar con el diseño de la estructura se realiza una tapa para la parte superior del case que permita proteger todos los elementos que están incorporados dentro de la estructura. La tapa cuenta con un borde que va introducido 2mm dentro del case, esto para que permita asegurar la tapa de mejor manera.

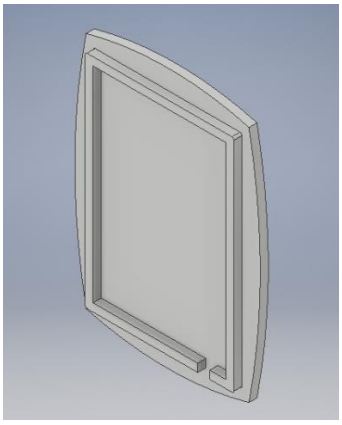


Figura 80.- Diseño de la tapa del case
Elaborado por: El investigador

Al concluir el diseño en “inventor” se generó un archivo en formato de estereolitografía, conocido por las siglas STL, con el fin de hacer una impresión en tres dimensiones. El material utilizado para la impresión fue filamento PLA conocido como ácido poliláctico debido a su resistencia en estructuras 3D, terminado el proceso de impresión se obtiene la estructura como muestra la figura 81.

Figura 81.- Case impreso en 3D
Elaborado por: El investigador

3.2.9 Pruebas de funcionamiento

Pruebas de autenticación

Las pruebas de funcionamiento se inician por la autenticación del usuario, para esto se crea una cuenta en la aplicación móvil y se verifica que las credenciales en la base de datos sean correctas, como muestra en la figura 82.

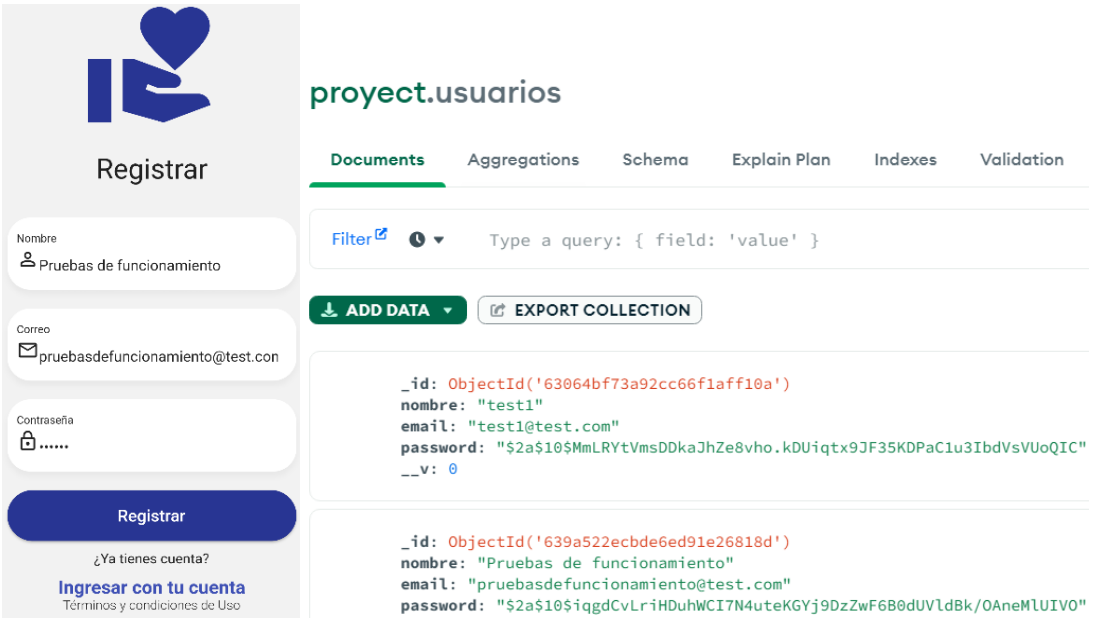


Figura 82.- Datos registrados en mongoDB
Elaborado por: El investigador

Entonces el sistema almacena correctamente las credenciales del usuario, y encripta la contraseña para que no sea visible por el gestor de la base de datos. Cuando el usuario es autenticado el backend automáticamente crea la regla en EMQX para almacenar los datos en mongoDB y enviar notificaciones al smartphone como muestra en la figura 83.

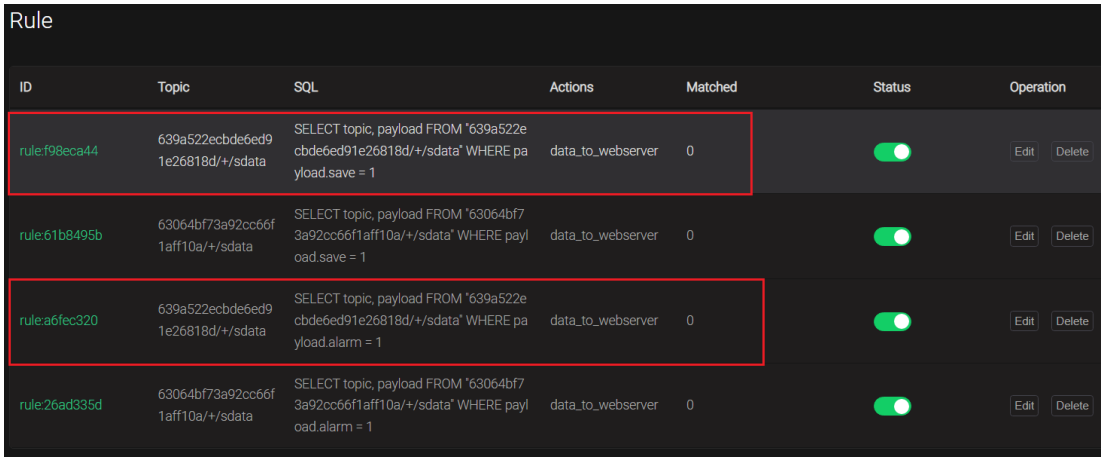


Figura 83.- Reglas creadas para el usuario registrado
Elaborado por: El investigador

Además, el bróker mantiene conexión constante con la aplicación móvil y con el brazalete, esto se puede apreciar en la figura 84 que son los clientes conectados a EMQX.

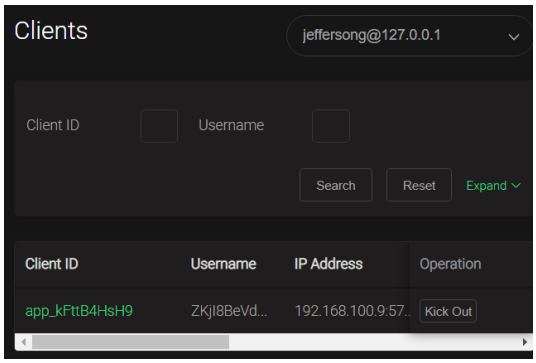


Figura 84.- Clientes conectados a EMQX (MQTT)
Elaborado por: El investigador

Pruebas del brazalete

Para mejorar la conectividad a wifi se controla en el firmware que la ESP32 esté conectado a WiFi siempre, en caso de no estar conectado, se reinicia hasta establecer una conexión estable a WiFi, con el fin de que la ESP32 pueda enviar los datos por MQTT; además, es necesario que el microcontrolador mantenga una conexión constante a WiFi para no perder comunicación con el servidor.

El valor de la temperatura que recopila el sensor se relacionó con la temperatura de la piel y la temperatura corporal, con el fin de mejorar la precisión de este signo vital, como se sabe cada parte del cuerpo humano tiene diferente temperatura, que suele ir decreciendo a los extremos, lo óptimo es determinar la temperatura central, para esto

fue necesario utilizar la ecuación que fue determinada en la investigación [56] usando el límite central de la estadística.

$$Temp = 0.109 * x + 33.07$$

Incorporando esta ecuación en la codificación del microcontrolador (ESP32) se obtiene mejores resultados de la temperatura corporal, además, coincide con la temperatura central del cuerpo humano de una persona de la tercera edad.

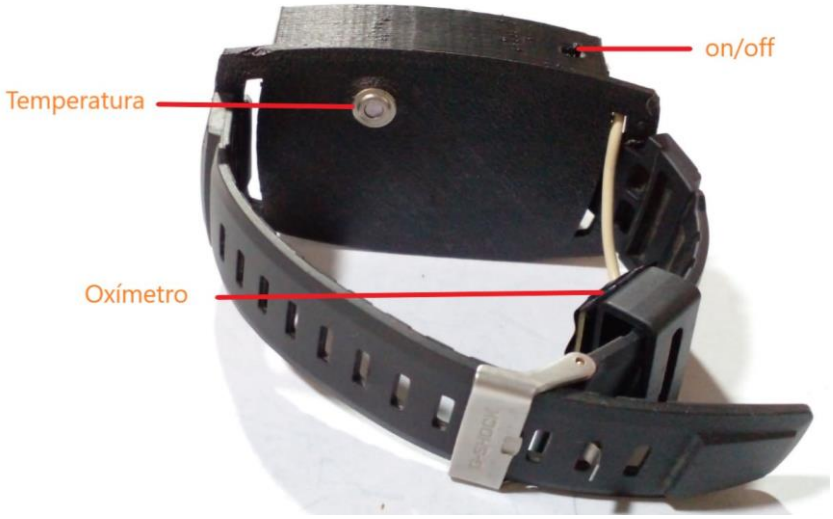


Figura 85.- Brazalete con sus partes principales
Elaborado por: El investigador

En la figura 85 muestra el brazalete con todos los instrumentos de medición de signos vitales incorporados, por lo tanto, el oxímetro tiene la capacidad de mover hasta ajustar a la arteria radial, de esta forma se ofrece mayor precisión al captar la frecuencia cardiaca y saturación de oxígeno.

Figura 86.- Brazalete incorporado en la muñeca.
Elaborado por: El investigador

Pruebas de la interfaz gráfica

Para probar el funcionamiento de la interfaz gráfica se instala la aplicación en un Samsung A02 que dispone de Android 10. La instalación se realiza con normalidad, proporcionando todos los permisos necesarios para acceder al almacenamiento y otros recursos. Al abrir la aplicación la primera interfaz que muestra es la autenticación, al ingresar las credenciales se puede acceder a la pantalla principal de la aplicación sin ningún problema, esta pantalla es la encargada de mostrar los valores de los sensores y las gráficas. Los valores de frecuencia cardiaca, saturación de oxígeno son valores enteros mientras que la temperatura corporal se utiliza valores decimales para lo cual se realiza la codificación en la aplicación móvil para presentar en las tarjetas con un solo decimal.

En la figura 87 muestra las tarjetas para presentar los valores de temperatura corporal, frecuencia cardiaca y saturación de oxígeno.

Figura 87.- Screen principal para mostrar los datos
Elaborado por: El investigador

Cuando el backend envía una notificación a la aplicación móvil, la tarjeta de notificaciones de emergencias muestra el aviso con texto de color rojo y fondo amarillo, para que tenga más atención del usuario; además, se implementa en la parte inferior un mensaje emergente, para lo cual se utiliza el widget snackbar, este aparece solamente por un periodo de 30 segundos, de esta forma muestra el contexto de la notificación cuando la aplicación está abierta.

Figura 88.- Tarjeta de notificaciones
Elaborado por: El investigador

En la gráfica histórica muestra los valores de temperatura corporal, frecuencia cardiaca y saturación de oxígeno, al tocar sobre un punto muestra el valor del sensor con su respectiva fecha de registro siendo de gran ayuda para los especialistas de la salud.

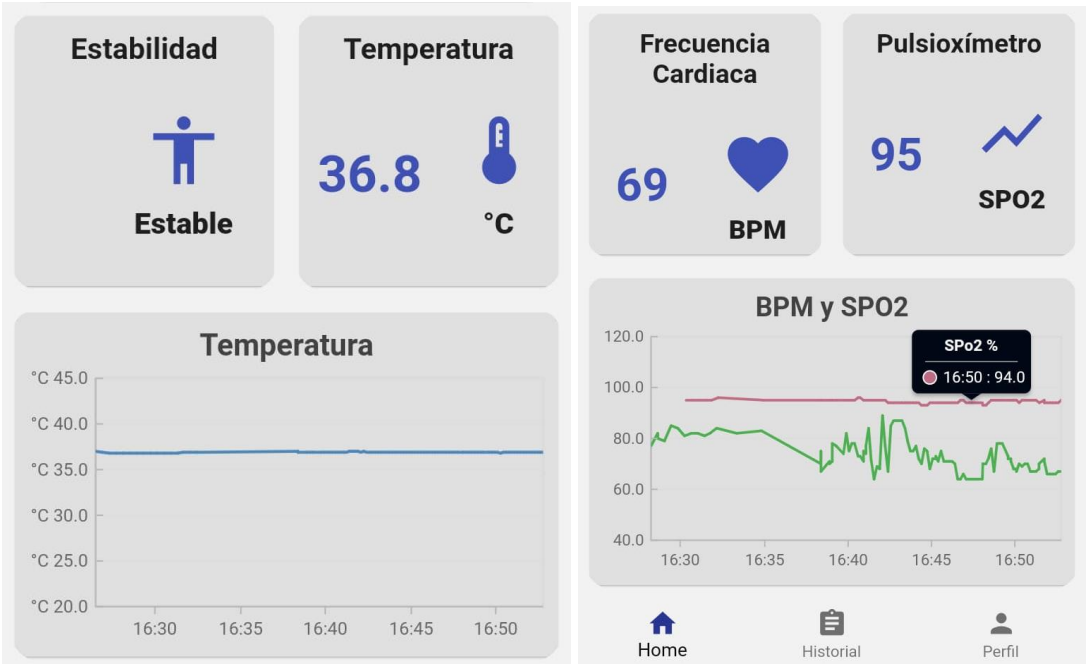


Figura 89.- Gráfico del histórico de datos
Elaborado por: El investigador

Para mostrar correctamente el contenido de las alertas se coloca condiciones a la respuesta de la petición http, esto ayuda que las tarjetas de presentación de datos no se dibujen en el screen cuando la respuesta de la petición HTTP este vacía. La figura 90 muestra las alertas recibidas con la enfermedad registrada y con los valores de signos vitales que presentaban en ese momento, además cada tarjeta tiene en la parte inferior registrada la fecha en que recibió la notificación.

Figura 90.- Historial de alertas recibidas
Elaborado por: El investigador

Además, en la barra principal de notificaciones del teléfono inteligente muestra los detalles de la alerta recibida (notificación) con los respectivos valores de los signos vitales que posee el adulto mayor en ese momento, como muestra la figura 91.

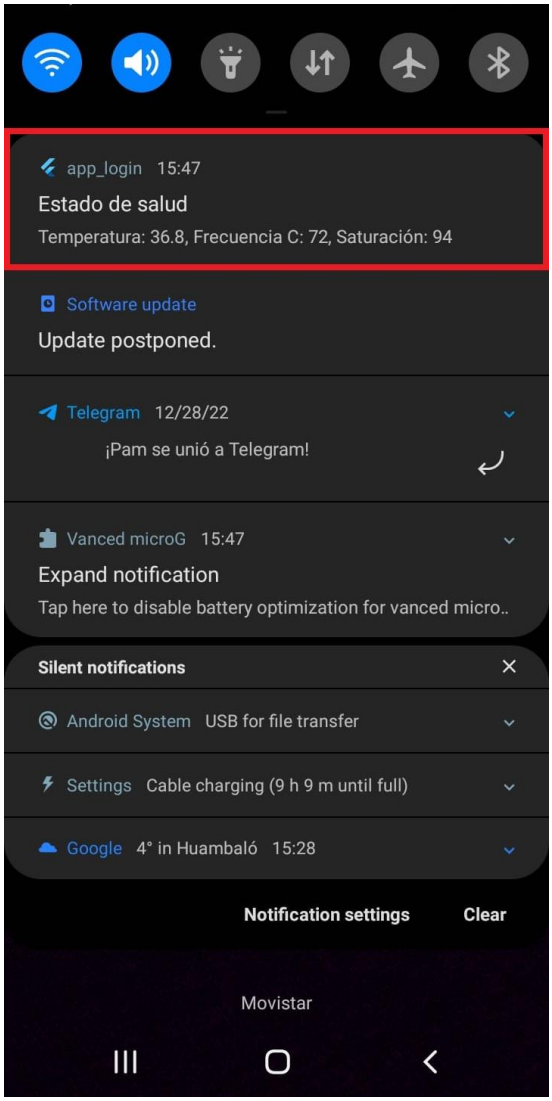


Figura 91.- Alerta en la barra de notificaciones del teléfono inteligente
Elaborado por: El investigador

En el menú crear alertas fue necesario controlar el botón “Guardar”, esto consiste en que el usuario debe activar el switch “Crear Alerta” para que tenga algún efecto al presionar el botón, en la figura 92 muestra que el botón esta desactivado hasta que se habilite el switch de crear alerta, de la misma forma con los “checklist” estos permiten seleccionar solamente uno de esta forma se controla que el usuario elija un límite de tiempo específico.

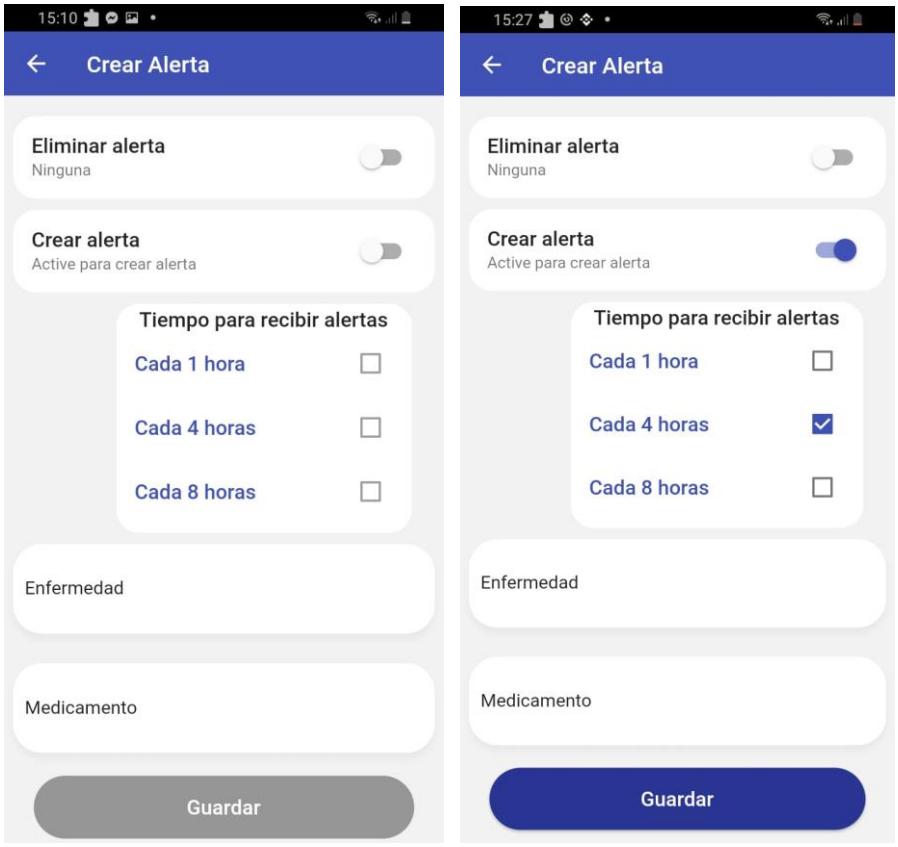


Figura 92.- Screen para crear alertas, botón desactivado
Elaborado por: El investigador

Cuando el usuario no completa todos los campos requeridos la aplicación muestra una alerta ante este evento, de esta forma el usuario conoce los requerimientos necesarios de la aplicación.

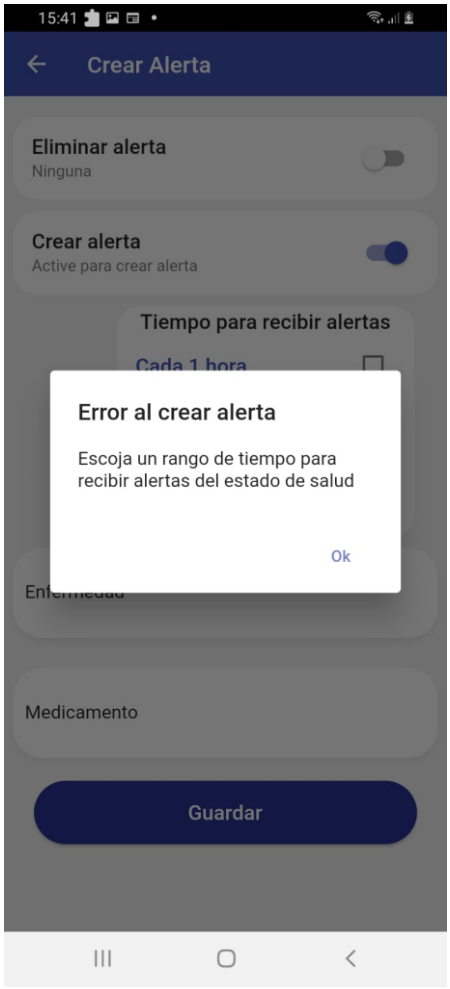


Figura 93.- Alerta de error al no cumplir con todos los parámetros
Elaborado por: El investigador

De la misma forma, cuando la alerta es creada correctamente muestra un mensaje satisfactorio o cuando el usuario elimina la alerta creada muestra un mensaje de éxito, como muestra la figura 94.

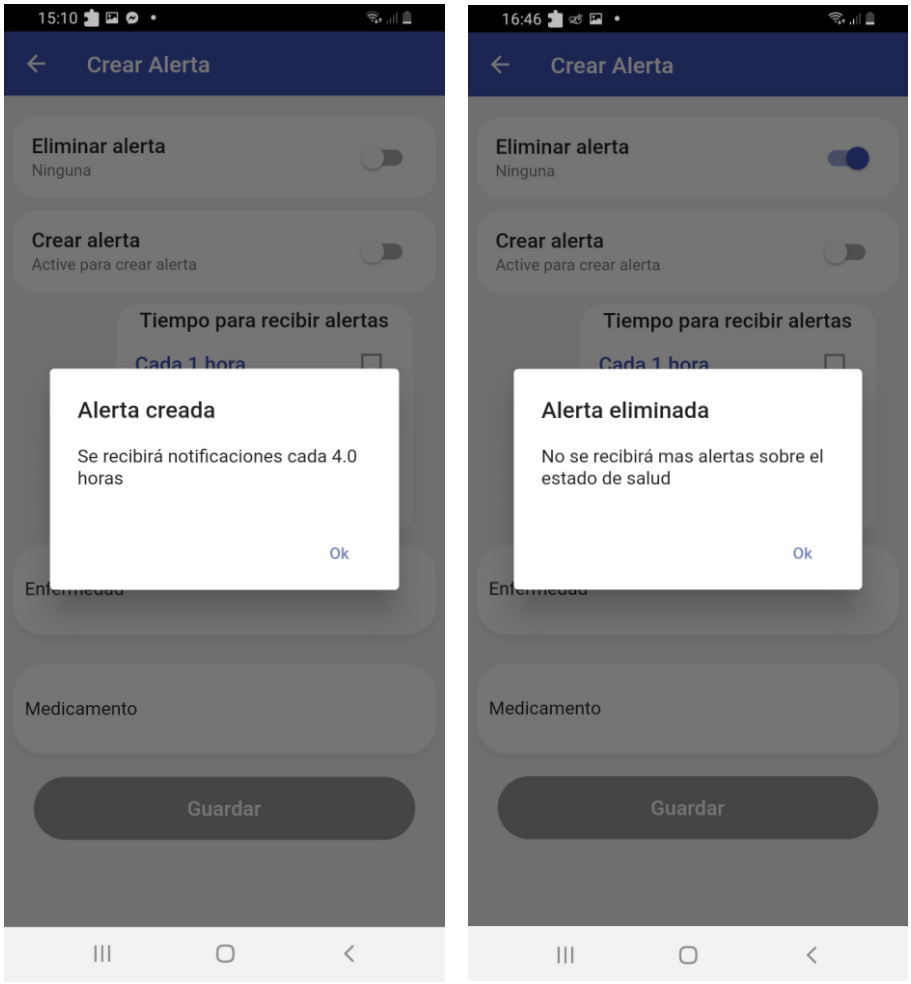


Figura 94.- Mensaje al crear o eliminar una alerta
Elaborado por: El investigador

El menú perfil almacena correctamente el usuario y correo electrónico con el que está registrado el usuario, de esta forma se recuerda la información personal antes de cerrar sesión en la aplicación.

Pruebas de notificaciones

Las notificaciones cumplen con su propósito en los tres estados de la aplicación:

- Terminado
- Fondo
- Primer plano

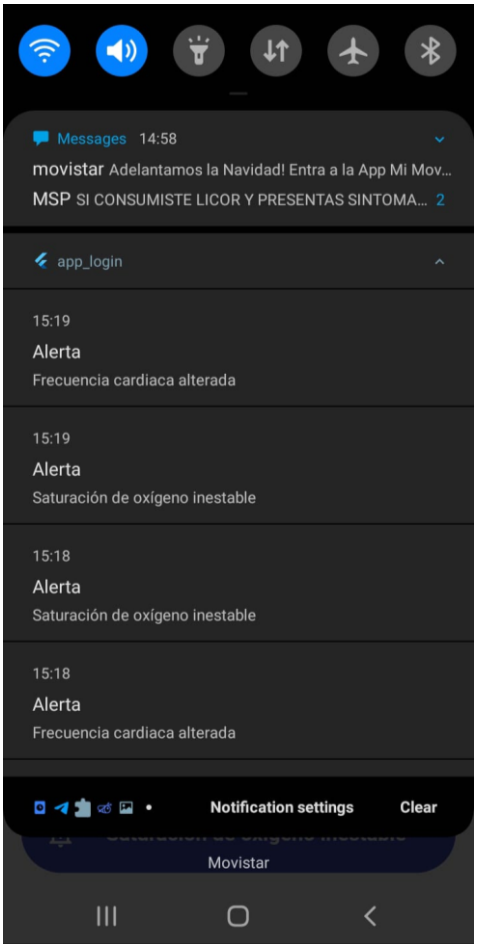


Figura 95.- Notificaciones cuando la aplicación esta minimizada
Elaborado por: El investigador

Pruebas de conectividad

Todas las peticiones HTTP al backend se realiza con éxito retornando el código 200, para mejorar la seguridad se implementa en los “headers” el token del usuario, de esta forma solamente pueden acceder a los recursos los usuarios con este token único, para esto se implementó la dependencia de “Jeson Web Token” en el backend, como se aprecia en la figura 96 todas las rutas disponen del validador JWT.

```
const {Router} = require('express');
const { getMedicalHistory } = require('../controllers/medical_history');
const { newAlert, updateAlert } = require('../controllers/new_alert');
const { validarJWT } = require('../middlewares/validar-jwt');

const router = Router();

//historial clínico
router.get('/medical_history', validarJWT,getMedicalHistory);

//new alert
router.post('/new_alert', validarJWT,newAlert);

//actualizar valor de status
router.post('/update_alert', validarJWT,updateAlert);
```

Figura 96.- Validador JWT
Elaborado por: El investigador

Tiempo de autonomía del dispositivo IoT

Se hizo un análisis profundo sobre la autonomía del dispositivo IoT estando activo todos los sensores y microcontrolador con conexión a wifi, teniendo en cuenta los recursos de consumo energético de cada componente que conforma el sistema, para lo cual fue necesario utilizar un multímetro DT858L (anexo 13). En la tabla 27 se determinó que la corriente máxima de consumo alcanzaba 248 mA por ende se utilizó el multímetro en la escala de amperaje de 10 A.

La pulsera IoT consume alrededor de 160mA, la batería que se utiliza tiene una capacidad de 1000mAh, por lo tanto, el dispositivo puede permanecer encendido 6 horas sin interrupciones.

$$160mA\ x\ 6horas = 960mAh$$

La tabla 36 muestra la autonomía del brazalete, para lo cual se hizo pruebas en tres días diferentes, obteniendo que el dispositivo puede permanecer encendido alrededor de 6 horas.

Tabla 36.- Tiempo de autonomía del brazalete

Fecha	Inicia	Apaga	Duración
12/12/2022	9:00 am	15:07 pm	6 horas y 7 minutos
13/12/2022	9:00 am	14:52 pm	5 horas y 52 minutos
14/12/2022	9:00 am	15:18 pm	6 horas y 18 minutos

Elaborado por: El investigador

Recursos del servidor

Los recursos que consume el servidor están reflejando en función de los clientes conectados, para pruebas de consumo se conecta 3 clientes, el consumo del CPU llega hasta el 8%, la RAM consumida es de 0.54GB mientras que el ancho de banda para subida es de 1.70Mbps y de bajada 3.26Mbps como se muestra en la figura 97.

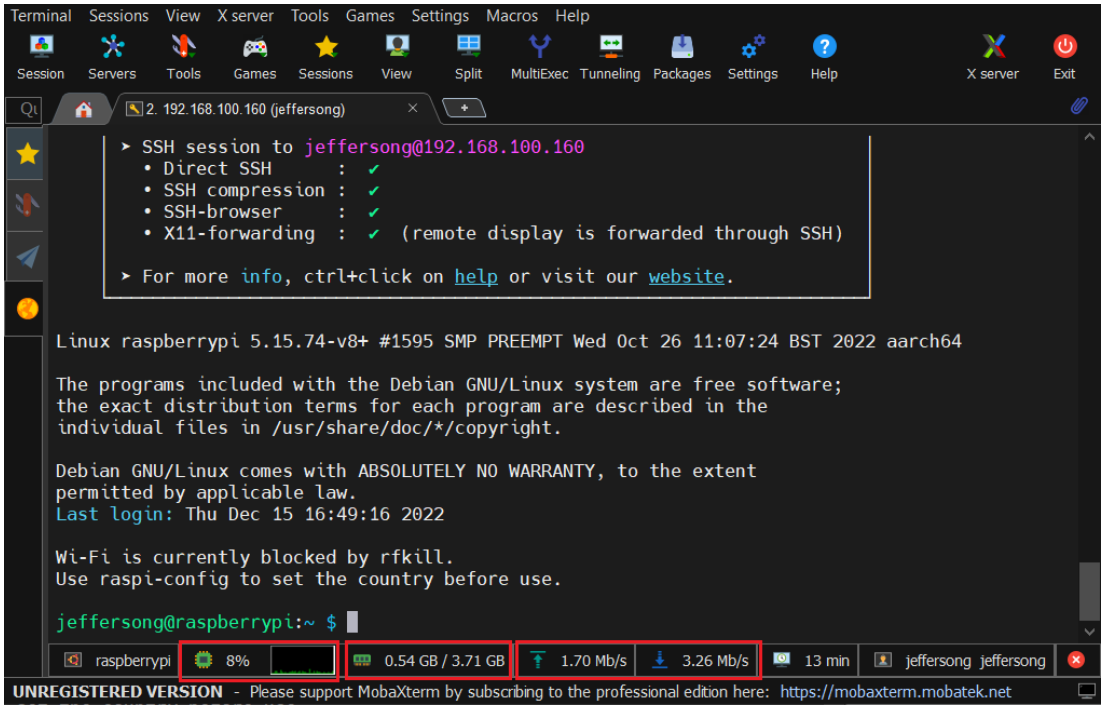


Figura 97.- Recursos consumidos por el servidor
Elaborado por: El investigador

3.2.10 Resultados

Para definir los resultados obtenidos del dispositivo IoT, es necesario dividir en dos partes, la primera se verifica que el modelo de "Machine Learning" creado en Edge Impulse sea capaz de reaccionar ante un evento de caída, la segunda parte es verificar que las condiciones de los signos vitales tomadas por los sensores del dispositivo IoT del adulto mayor sean correctas.

En la implementación del modelo de "Machine Learning" se prueba el efecto de filtrado (bloque de procesamiento) para determinar la mejor precisión en la clasificación de actividades, para esto se usó todos los tipos de caídas y actividades de la vida diaria de la tabla 25. Siguiendo las recomendaciones de Bosch Rué y colaboradores se aplicó la experimentación para definir el número de ciclos (épocas) y la tasa de aprendizaje.

Iniciando con la experimentación, los primeros resultados tienen una precisión muy baja; además, la matriz de confusión presenta puntajes muy bajos al clasificar los datos, esto era consecuencia de subir la muestra de caídas en un intervalo de 4 segundos, pero este fenómeno dura alrededor de 200 ms a 500 ms, entonces la muestra de 4 segundos contenía datos de actividades de la vida diría (AVD) y caídas (fall). Para mejorar la precisión de clasificación se divide la muestra de 4 segundos a 1 segundo dejando solamente los datos que producen al ejecutar un evento de caída como muestra el anexo 12, con esto se pretende mejorar la clasificación de datos.

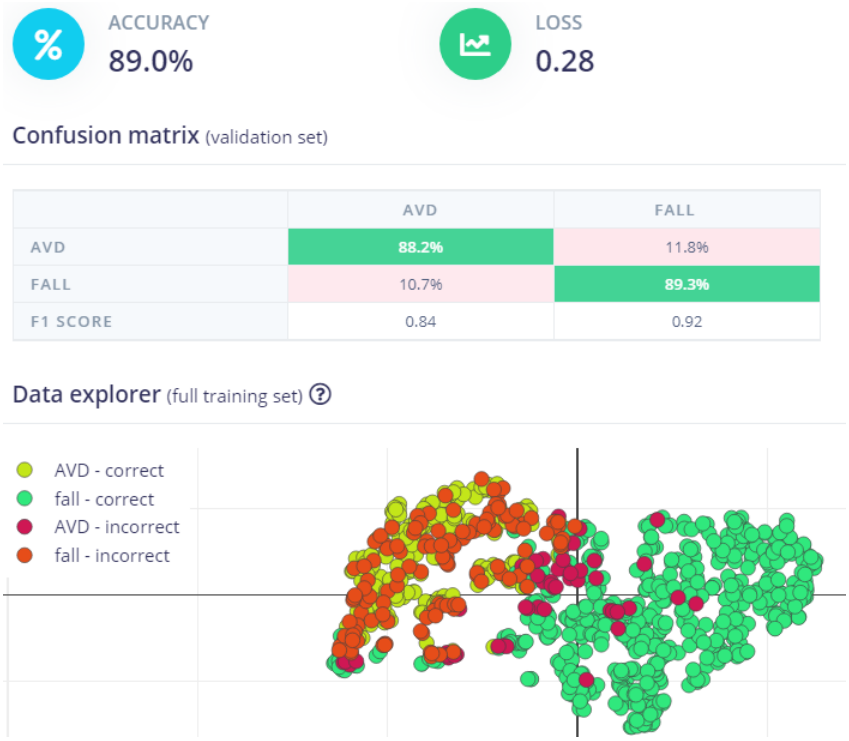


Figura 98.- Precisión del modelo con muestra de 4 segundos
Elaborado por: El investigador

La tabla 37 muestra los resultados obtenidos en la experimentación al implementar el filtro pasa bajo de orden 6, donde al variar el número de épocas y la tasa de aprendizaje se mantiene los mismos resultados desde el inicio incluso entrenando a la red neuronal con 240 épocas, siendo este valor muy elevado para el aprendizaje de la red.

Tabla 37.- Experimentación aplicando filtro pasa bajo de orden 6

Aplicando filtro de orden 6				
Épocas	Tasa de aprendizaje	Precisión	Matriz de confusión	
			AVD	Fall
10	0.05	99.2%	99.6%	80.8%
10	0.005	99.3%	100.0%	75.3%
25	0.05	99.2%	99.6%	80.8%
25	0.005	99.3%	100.0%	75.3%
60	0.05	99.2%	99.6%	80.8%
60	0.005	99.5%	99.9%	83.8%
120	0.05	99.2%	99.6%	80.8%
120	0.005	99.6%	100.0%	82.2%
240	0.05	99.2%	99.6%	80.8%
240	0.005	99.6%	100.0%	82.2%

Elaborado por: El investigador

La figura 99 muestra la precisión del 99.6% utilizando 120 épocas con una tasa de aprendizaje de 0.005 usando un filtro pasa bajo de orden 6. La matriz de confusión muestra que las AVD no tienen confusión con las caídas, mientras que en un 15.1% las caídas pueden ser interpretadas como una actividad (AVD), siendo un porcentaje muy elevado de confusión en caídas.

6

Figura 99.- Clasificación al aplicar filtro pasa bajo de orden 6
Elaborado por: El investigador

La tabla 38 muestra los resultados obtenidos en la experimentación sin utilizar un filtro, la tasa de aprendizaje de 0.005 presenta mejores resultados desde el inicio, mientras que el número de épocas se mantiene con los mismos resultados al principio y mejora con 25 épocas, entonces este número se utilizó para evitar el sobre entrenamiento ya que con valores más elevados se mantiene los mismos resultados.

Tabla 38.- Experimentación sin filtrado

Sin filtro				
Épocas	Tasa de aprendizaje	Precisión	Matriz de confusión	
			AVD	Fall
10	0.05	99.5%	99.7%	90.4%
10	0.005	99.9%	99.9%	97.3%
18	0.05	99.6%	99.7%	95.9%
18	0.005	99.9%	99.9%	97.3%
25	0.05	99.6%	99.7%	94.5%
25	0.005	99.9%	100.0%	97.3%
40	0.05	99.6%	99.9%	86.3%
40	0.005	99.9%	100.0%	97.3%
70	0.05	99.6%	99.9%	86.3%
70	0.005	99.9%	100.0%	97.3%

Elaborado por: El investigador

La figura 100 tiene mejores resultados que la figura 99, al usar los datos sin filtrar (bloque de procesamiento) para la clasificación, teniendo una precisión del 99.9% utilizando 25 épocas y una tasa de aprendizaje de 0.005. La matriz de confusión también es mejor debido a que las caídas pueden ser interpretadas como AVD en un porcentaje del 2.7%, siendo este el modelo óptimo para incorporar en el dispositivo IoT.

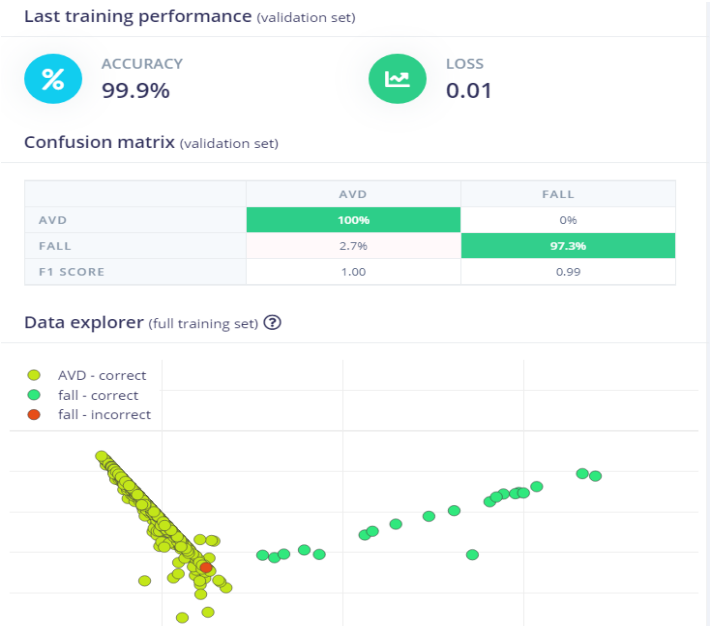


Figura 100.- Clasificación al usar datos sin filtrar con 45 épocas
Elaborado por: El investigador

Para verificar la precisión del modelo se utilizó a otro grupo diferente de personas con edades que aproximen a la tercera edad, esto para ser mas realista ante un evento de caída de una persona de la tercera edad. Para esto se se utilizó a 10 personas de las cuales 7 tienen entre 40 y 55 años mientras que las 3 personas restantes tienen entre 58 y 65 años (anexo 14).

Cada participante realizó 5 tipos de caídas: caer hacia adelante, caer vertical, caer hacia atrás, caer hacia atrás mientras está sentado y caer lateral mientras está sentado, cada tipo de caída fue repetida por 2 veces, teniendo un total de 100 caídas de todos los participantes, los resultados obtenidos se tabulan en la tabla 39. El tipo de caída que menos aciertos tiene es al caer hacia atrás, esto se debe por la forma de caer o los movimientos que produce la muñeca al producir este tipo de caída, pero el sistema es capaz de detectar el 91 % de caídas.

Tabla 39.- Resultados obtenidos al producir un evento de caída

Participante	Caer hacia adelante	Caer lateral	Caer hacia atrás	Sentado	
				Caer hacia atrás	Caer lateral
Participante 1	2	1	2	2	2
Participante 2	2	2	1	2	2
Participante 3	2	2	2	1	2
Participante 4	2	1	2	2	2
Participante 5	2	2	1	2	1
Participante 6	2	2	2	2	2
Participante 7	1	2	2	2	2
Participante 8	2	2	2	2	2
Participante 9	2	2	2	1	2
Participante 10	2	2	1	2	2
sub Total	**19**	**18**	**17**	**18**	**19**
TOTAL	**91**				

Elaborado por: El investigador

En la figura 101 muestra de mejor manera los resultados obtenidos del total de caídas, donde el 91% de caídas fueron detectadas, mientras que apenas el 9% de caídas el dispositivo IoT no pudo detectar.

.

Figura 101.- Caídas detectadas por el modelo de Machine Learning
Elaborado por: El investigador

En la segunda parte, utilizando instrumentos de medición para los signos vitales y el dispositivo IoT, se determina el error que representa el dispositivo calculando el error absoluto y el error relativo por medio de las ecuaciones:

$$E_{absoluto} = medición\ del\ instrumento - medición\ del\ dispositivo\ IoT$$

$$E_{relativo} = \frac{E_{absoluto}}{medición\ del\ instrumento}\ x\ 100\%$$

Para la toma de temperatura, frecuencia cardiaca y saturación de oxígeno se utilizó a dos personas donde el primer participante tiene 92 años (edad avanzada) mientras que el segundo participante tiene 82 años, para determinar la eficacia del dispositivo IoT se realiza en dos partes, cuando la mano está en reposos (sin movimiento) y cuando se encuentra haciendo una actividad (en movimiento continuo).

El primer signo vital que se analizó es la temperatura corporal tabulada en la tabla 40, donde se muestra datos adquiridos por el brazalete (dispositivo IoT) y el termómetro médico (anexo 15). La temperatura se mide en la axila, en promedio el termómetro médico alcanzó los 36.2°C, pero esta no es la temperatura central, entonces los profesionales de la salud recomiendan sumar 0.5°C al valor arrojado de temperatura de la axila, con el fin de estimar la temperatura central, haciendo esta operación la temperatura central estimada alcanzaría los 36.7°C, es necesario realizar esta acción porque el brazalete determina la temperatura central. El anexo 16 muestra los resultados del termómetro digital (médico).

Tabla 40.- Resultados de la temperatura corporal cuando la mano está en movimiento y en reposo.

TEMPERATURA								
No	**Participante 1**				**Participante 2**			
	Brazalete °C	**Instrumento °C**	**Error Absoluto**	**Error Relativo**	**Brazalete °C**	**Instrumento °C**	**Error Absoluto**	**Error Relativo**
Muñeca sin movimiento								
1	36.7	36.6	0.1	0.27%	36.4	36.5	0.1	0.27%
2	36.7	36.7	0	0.00%	36.4	36.5	0.1	0.27%
3	36.7	36.6	0.1	0.27%	36.5	36.4	0.1	0.27%
4	36.8	36.7	0.1	0.27%	36.4	36.6	0.2	0.55%
5	36.9	36.7	0.2	0.54%	36.5	36.5	0	0.00%
6	36.8	36.7	0.1	0.27%	36.5	36.6	0.1	0.27%
7	36.7	36.7	0	0.00%	36.6	36.4	0.2	0.55%
8	36.8	36.7	0.1	0.27%	36.6	36.5	0.1	0.27%
9	36.9	36.7	0.2	0.54%	36.6	36.6	0	0.00%
10	36.8	36.7	0.1	0.27%	36.6	36.5	0.1	0.27%
Muñeca en movimiento								
1	36.6	36.7	0.1	0.27%	36.6	36.6	0	0.00%
2	36.8	36.6	0.2	0.55%	36.5	36.6	0.1	0.27%
3	36.8	36.6	0.2	0.55%	36.5	36.5	0	0.00%
4	36.7	36.7	0	0.00%	36.6	36.4	0.2	0.55%
5	36.9	36.7	0.2	0.54%	36.4	36.5	0.1	0.27%
6	36.8	36.6	0.2	0.55%	36.6	36.4	0.2	0.55%
7	36.8	36.7	0.1	0.27%	36.5	36.6	0.1	0.27%
8	36.9	36.7	0.2	0.54%	36.5	36.6	0.1	0.27%
9	36.8	36.6	0.2	0.55%	36.6	36.5	0.1	0.27%
10	36.9	36.8	0.1	0.27%	36.6	36.4	0.2	0.55%

Elaborado por: El investigador

La temperatura corporal no presenta inconvenientes al estar la muñeca en movimiento constante, el error máximo en ambos casos es de 0.2°C. La figura 102 y la figura 103 muestra de mejor manera la variación de la temperatura recopilada por el brazalete en las diez mediciones, además los valores de temperatura están dentro del rango normal.

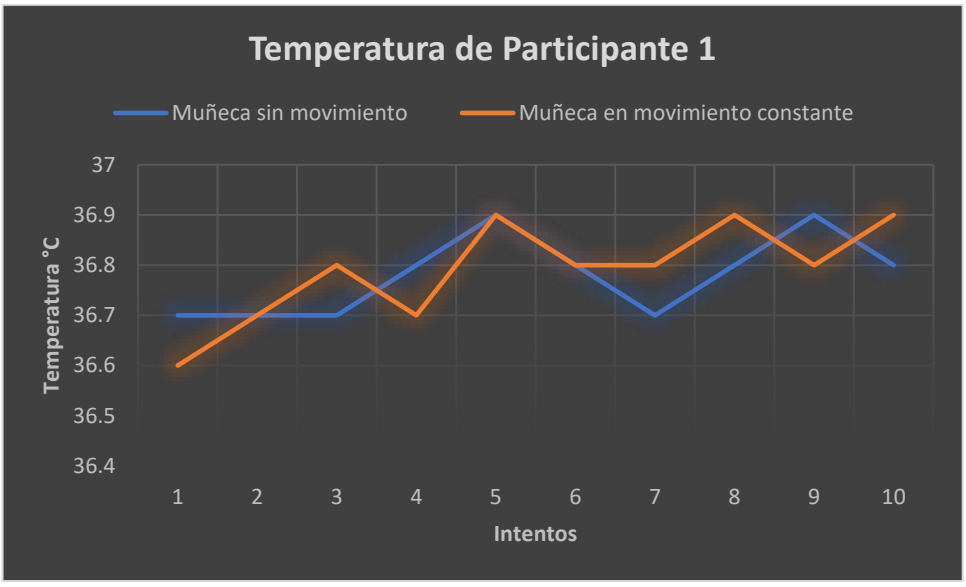


Figura 102.- Variación de temperatura corporal del primer participante
Elaborado por: El investigador

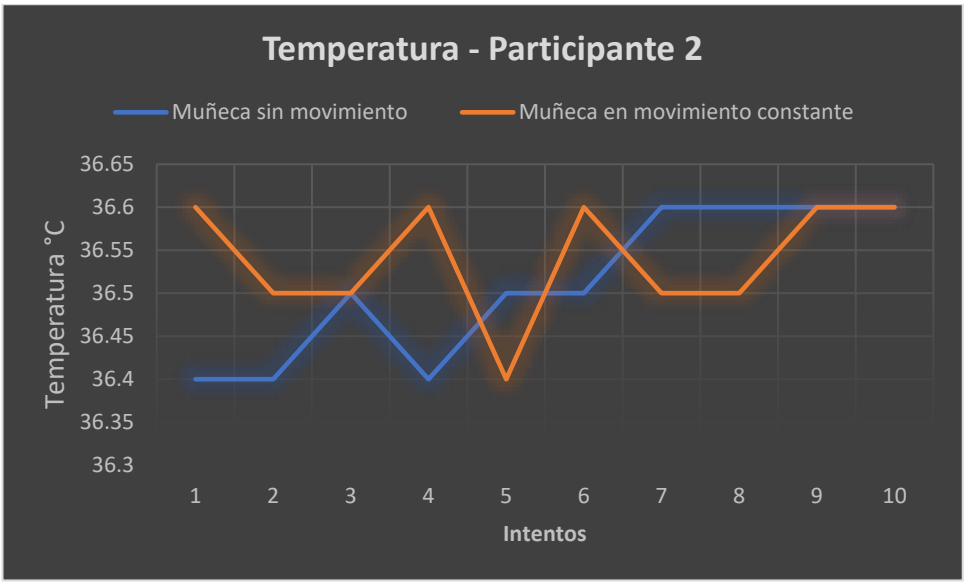


Figura 103.- Variación de temperatura corporal del participante 2
Elaborado por: El investigador

Para recopilar la frecuencia cardiaca se hizo en un intervalo de 5 minutos cada medida, para tener diferentes valores, se utilizó el oxímetro Yuwell del anexo 17 para comparar con el brazalete. Para comprobar la confiabilidad del brazalete se realiza en dos campos, cuando la mano está en movimiento continuo y cuando está en reposo, aunque los profesionales de la salud recomiendan medir este parámetro cuando el paciente está tranquilo y en reposo.

Tabla 41.- Resultados de la frecuencia cardiaca, cuando la muñeca está en movimiento continuo y en reposo.

Frecuencia Cardiaca								
No	Participante 1				Participante 2			
	Brazale-te (BPM)	Instrumen-to (BPM)	Error Absolu-to	Error Relati-vo	Brazalete (BPM)	Instrumento (BPM)	Error Absolu-to	Error Relati-vo
Muñeca sin movimiento								
1	70	72	2	2.78%	82	84	2	2.38%
2	67	70	3	4.29%	84	83	1	1.20%
3	69	68	1	1.47%	80	81	1	1.23%
4	71	69	2	2.90%	79	82	3	3.66%
5	68	70	2	2.86%	81	79	2	2.53%
6	70	67	3	4.48%	83	80	3	3.75%
7	67	67	0	0.00%	81	81	0	0.00%
8	68	70	2	2.86%	80	78	2	2.56%
9	71	71	0	0.00%	78	78	0	0.00%
10	67	68	1	1.47%	78	77	1	1.30%
Muñeca en movimiento constante								
1	62	69	7	10.14%	79	73	6	8.22%
2	69	75	6	8.00%	87	81	6	7.41%
3	76	70	6	8.57%	80	75	5	6.67%
4	65	68	3	4.41%	75	82	7	8.54%
5	73	71	2	2.82%	73	79	6	7.59%
6	80	75	5	6.67%	83	81	2	2.47%
7	69	69	0	0.00%	74	78	4	5.13%
8	73	77	4	5.19%	81	76	5	6.58%
9	63	68	5	7.35%	83	79	4	5.06%
10	78	71	7	9.86%	77	80	3	3.75%

Elaborado por: El investigador

Los mejores resultados se obtienen cuando la mano del participante está en reposo (sin movimiento) alcanzado un error absoluto de 3 BPM. Cuando la muñeca está en continuo movimiento alcanza un error más alto, esto porque ingresa ruido del movimiento o por la luz ambiente, aunque un motivo de mayor afectación puede ser que el brazalete esta sujetado con demasiada ligereza a la muñeca. Además, el sensor utiliza un solo led infrarrojo (IR) para determinar esta constante vital, midiendo el fotodetector la absorción de la luz IR, esto lo convierte en un sensor sensible, ya que, si existe un pequeño movimiento del sensor con respecto a la muñeca, la lectura de frecuencia cardiaca se verá afectada en el dispositivo IoT.

La figura 104 muestra la variación de frecuencia cardiaca tomada por el brazalete cuando la muñeca está en continuo movimiento y sin movimiento del participante 1.

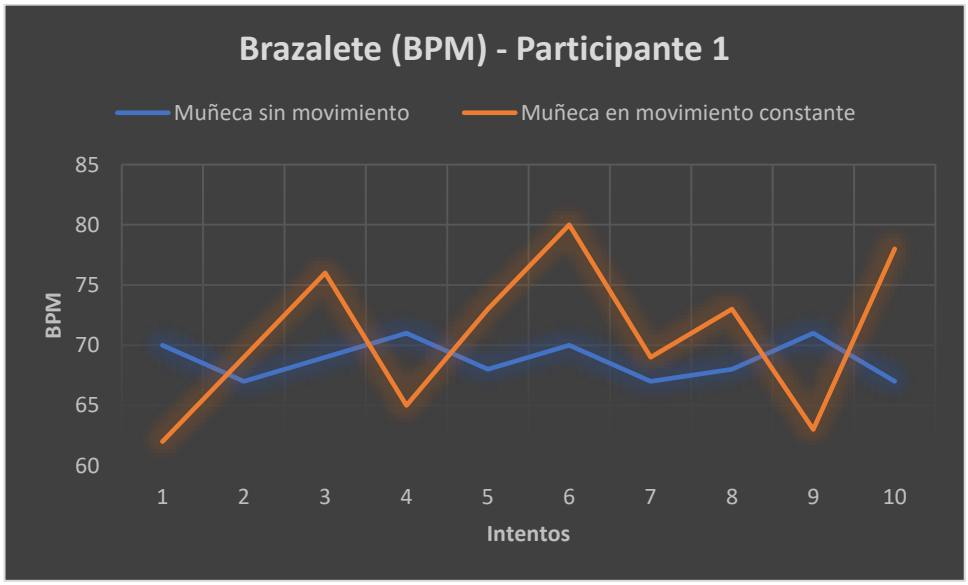


Figura 104.- Variación de frecuencia cardiaca del participante 1
Elaborado por: El investigador

La figura 105 muestra la variación de las mediciones del brazalete con respecto al participante 2 cuando la muñeca está en reposo y en movimiento.

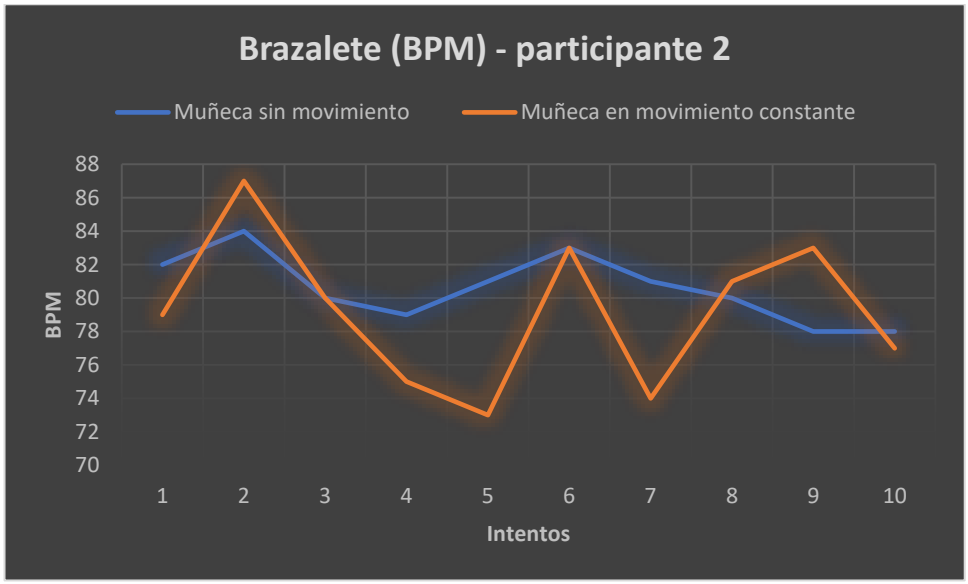


Figura 105.- Variación de frecuencia cardiaca del participante 2
Elaborado por: El investigador

La figura 106 y figura 107 muestra la variación de la frecuencia cardiaca con respecto al oxímetro médico, este instrumento médico también se ve afectado cuando el paciente realiza constantes movimientos cuando está siendo tomado la frecuencia

cardiaca, mientras que cuando está en reposo la frecuencia cardiaca se mantiene con una variación muy pequeña.

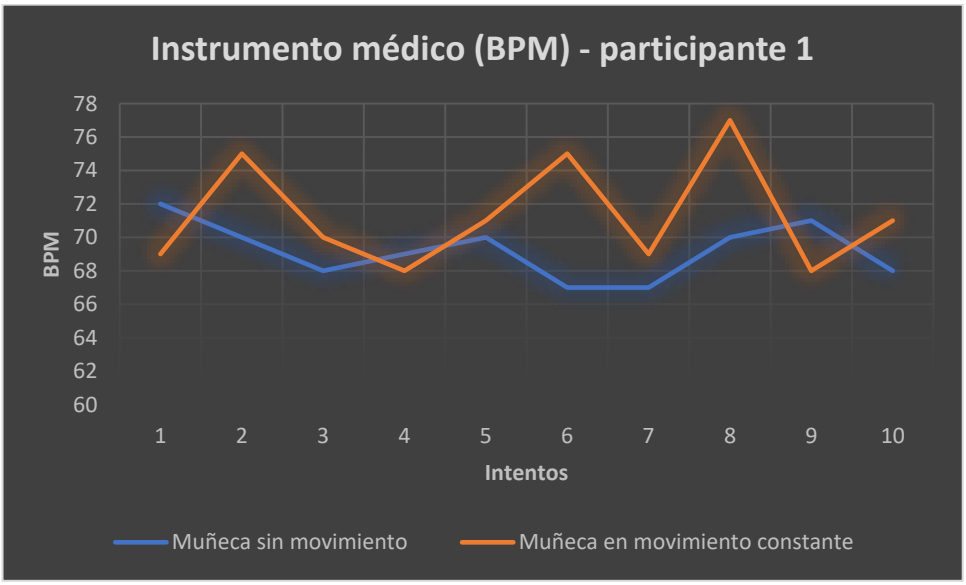


Figura 106.- Variación de la frecuencia cardiaca del oxímetro médico con respecto al participante 1
Elaborado por: El investigador

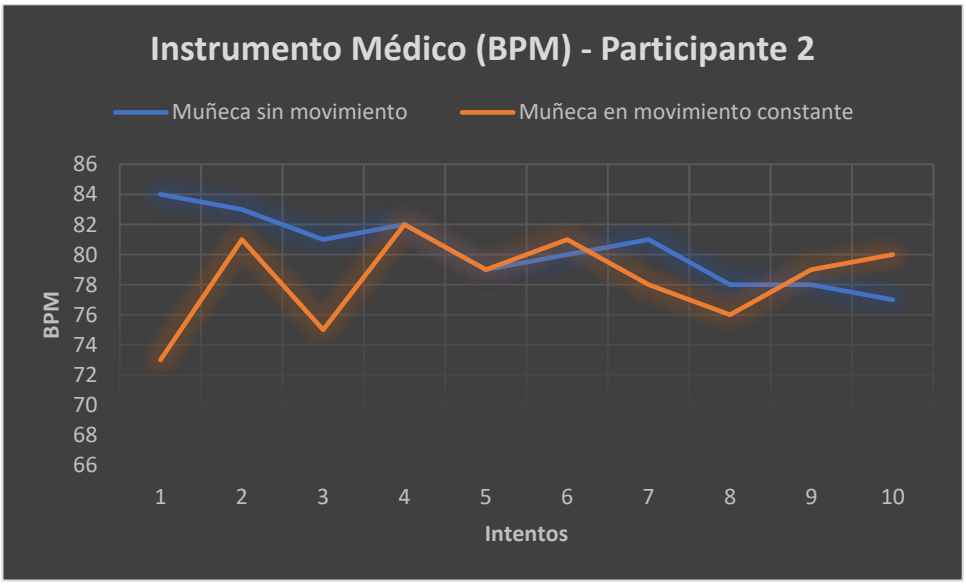


Figura 107.- Variación de frecuencia cardiaca del oxímetro médico con respecto al participante 2
Elaborado por: El investigador

Al igual que la frecuencia cardiaca la saturación de oxígeno de los participantes se tomó en un intervalo de 5 minutos, para comparar los resultados se utiliza el oxímetro

Yuwell descrito en el anexo 17. Para verificar la confiabilidad del dispositivo se realiza la toma de esta constante vital cuando el paciente está en movimiento y en reposo.

Tabla 42.- Niveles de saturación de oxígeno cuando el participante está en reposo y en continuo movimiento.

Saturación de oxígeno								
No	**Participante 1**				**Participante 2**			
	Brazale-te (%)	**Instrum ento (%)**	**Error Absolut-o**	**Error Relativo**	**Braza-lete (%)**	**Instrum-ento (%)**	**Error Absolut-o**	**Error Relativo**
Muñeca sin movimiento								
1	93	93	0	0.00%	92	94	2	2.13%
2	94	93	1	1.08%	91	92	1	1.09%
3	93	95	2	2.11%	93	91	2	2.20%
4	94	94	0	0.00%	92	92	0	0.00%
5	94	95	1	1.05%	91	92	1	1.09%
6	93	94	1	1.06%	94	95	1	1.05%
7	94	95	1	1.05%	92	94	2	2.13%
8	92	94	2	2.13%	93	93	0	0.00%
9	92	93	1	1.08%	92	94	2	2.13%
10	93	93	0	0.00%	91	94	3	3.19%
Muñeca en movimiento constante								
1	92	94	2	2.13%	92	93	1	1.08%
2	94	93	1	1.08%	94	91	3	3.30%
3	92	95	3	3.16%	91	93	2	2.15%
4	93	93	0	0.00%	93	94	1	1.06%
5	94	92	2	2.17%	92	95	3	3.16%
6	91	90	1	1.11%	95	94	1	1.06%
7	92	94	2	2.13%	93	92	1	1.09%
8	92	92	0	0.00%	91	91	0	0.00%
9	94	95	1	1.05%	94	92	2	2.17%
10	93	95	2	2.11%	92	95	3	3.16%

Elaborado por: El investigador

La saturación de oxígeno alcanza un error absoluto de 3 % (SpO2) en el caso que la muñeca se encuentre en constante movimiento; mientras que cuando está en reposo los valores son más precisos. A diferencia de la frecuencia cardiaca la saturación de oxígeno presenta mejores resultados en ambos casos, esto se debe a que esta constante vital es determinada por un led rojo y un led infrarrojo (IR), configurados en 660nm y 880nm respectivamente, encargados de determinar la hemoglobina oxigenada (HbO2) y hemoglobina desoxigenada (Hb).

La figura 108 y la figura 109 muestra la variación de los valores tomados por el brazalete para los dos participantes, cuando la mano está en continuo movimiento y en reposo (sin movimiento).

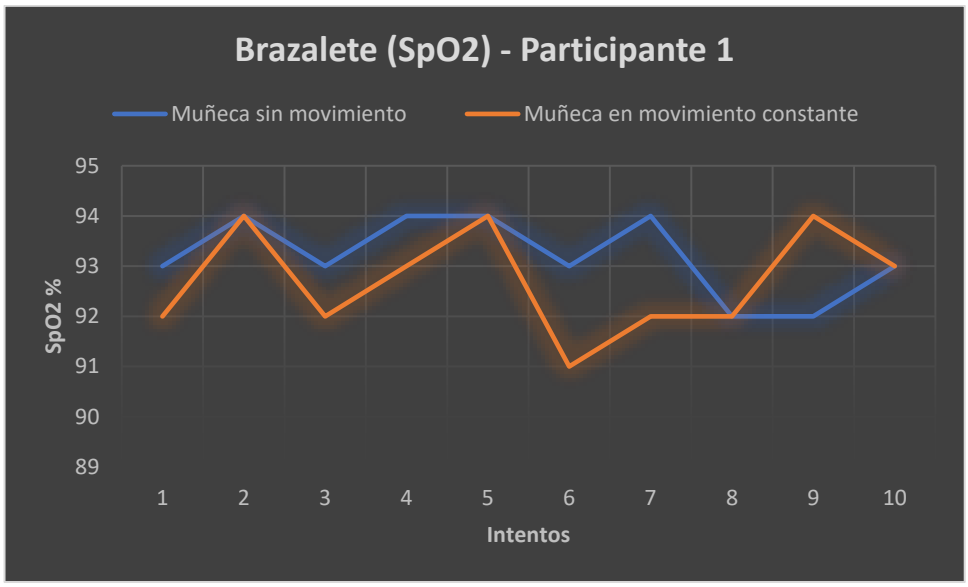


Figura 108.- Variación de la saturación de oxígeno en el participante 1
Elaborado por: El investigador

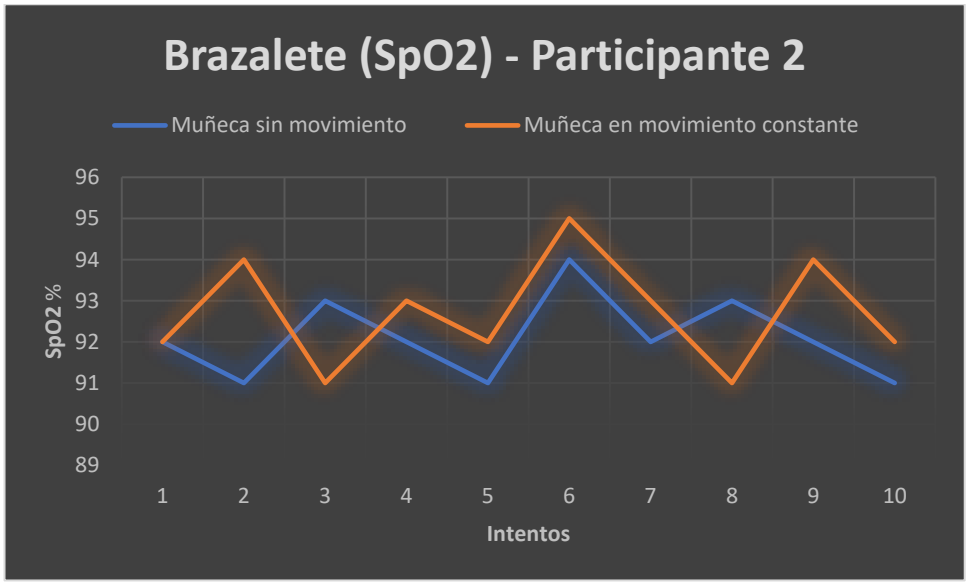


Figura 109.- Variación de la saturación de oxígeno en el participante 2
Elaborado por: El investigador

La figura 110 y la figura 111 muestran la variación de saturación de oxígeno medido por el instrumento médico (oxímetro).

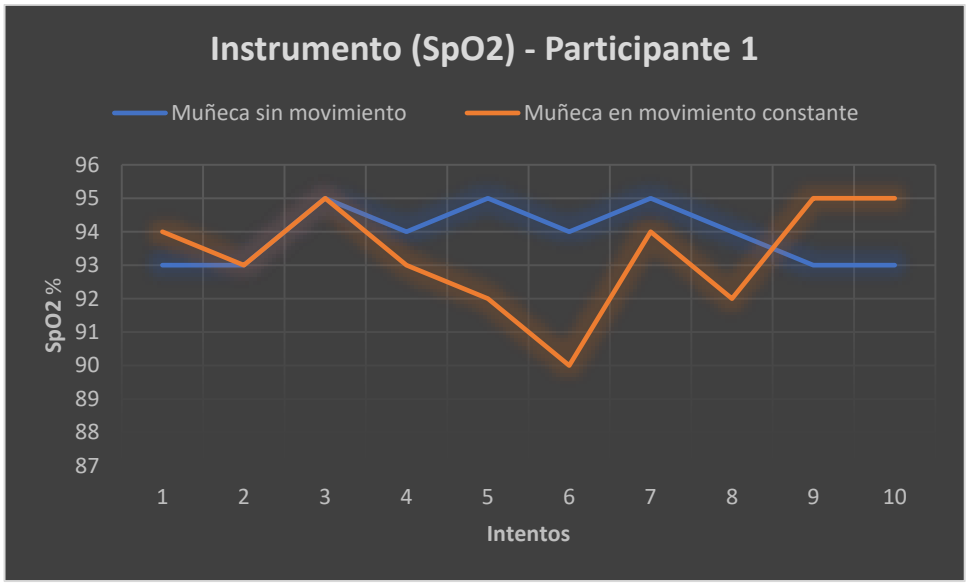


Figura 110.- Variación de la saturación de oxígeno con respecto al instrumento médico en el participante 1
Elaborado por: El investigador

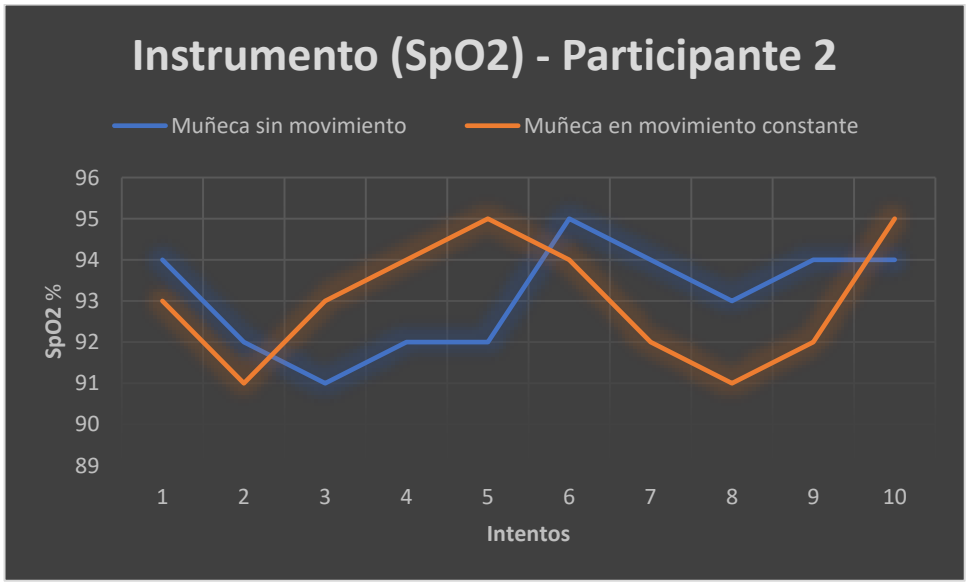


Figura 111.- Variación de la saturación de oxígeno con respecto al instrumento médico en el participante 2
Elaborado por: El investigador

Por otra parte, se mide la capacidad que tiene el dispositivo IoT de reconectar a WiFi y al bróker MQTT de forma automática. Para medir estos factores es necesario apagar el modem y luego encender, esto se hizo por 10 veces, la tabla 43 muestra los resultados obtenidos, donde el brazalete tiene la capacidad suficiente para reconectar las veces necesarias a la red y al bróker de forma automática.

Tabla 43.- Capacidad para reconectar a Wifi y al bróker MQTT

Intentos	Conecta	
	WiFi	**MQTT**
1	Si	Si
2	Si	Si
3	Si	Si
4	Si	Si
5	Si	Si
6	Si	Si
7	Si	Si
8	Si	Si
9	Si	Si
10	Si	Si

Elaborado por: El investigador

3.2.11 Presupuesto

El presupuesto del proyecto se divide en dos partes, la primera es el presupuesto para la implementación del dispositivo IoT, mientras que la segunda parte es la inversión necesaria para el servidor.

La tabla 44 detalla el presupuesto necesario para construir el dispositivo IoT con capacidad para detectar caídas y tomar signos vitales.

Tabla 44.- Presupuesto del dispositivo IoT

Concepto	Cantidad	Precio Unitario	Precio Total
ESP32	1	$15.00	$15.00
MAX30100	1	$9.00	$9.00
MLX90614	1	$23.00	$23.00
MPU6050	1	$5.00	$5.00
Resistencias	2	$0.05	$0.10
Cables de conexión	1	$0.25	$0.25
Batería	1	$22.00	$22.00
Capacitores	2	$0.40	$0.80
Regulador 3.3V	1	$2.00	$2.00
TP4056	1	$3.50	$3.50
Case	1	$15.00	$15.00
Correa para sujetar	1	$5.00	$5.00
Subtotal			**$100.65**

Elaborado por: El investigador

La tabla 45 muestra la inversión necesaria para el servidor, con capacidad para ejecutar el sistema operativo Ubuntu.

Tabla 45.- Presupuesto del servidor

Concepto	Cantidad	Precio Unitario	Precio Total
Raspberry pi 4	1	$230.00	$230.00
Subtotal			**$230.00**

Elaborado por: El investigador

La tabla 46 muestra la inversión necesaria para el diseño y ensamblado

Tabla 46.- Presupuesto de diseño y ensamblado

Descripción	**Precio**
Diseño	30
Ensamblado	10
Subtotal	**40**

Elaborado por: El Investigador

En la tabla 47 se detalla los costos de implementación del dispositivo IoT y el costo que cubre para el servidor, entonces el costo total para el proyecto alcanza los $370. 65, pero a esto se aumenta imprevistos con el 10% llegando a valer el proyecto $407.72.

Tabla 47.- Presupuesto final para el sistema IoT

Concepto	**Precio**
Dispositivo IoT	$100.65
Diseño y ensamblado	$40.00
Servidor	$230.00
Subtotal	**$370.65**
Imprevisto (10%)	$37.07
Total	**$407.72**

Elaborado por: El investigador

CPAÍTULO IV

CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

Al terminar el presente trabajo de titulación se establece las siguientes conclusiones:

- Al implementar un brazalete para la muñeca los signos vitales que se pueden medir en esta área es la temperatura corporal usando la ecuación que relaciona la temperatura de la piel con la temperatura corporal, la frecuencia cardiaca y saturación de oxígeno por medio de la arteria radial.

- Los accidentes más frecuentes en adultos mayores son las caídas provocando fracturas o hospitalizaciones, además este accidente puede ser monitoreado por acelerómetros. Al implementar un brazalete en la muñeca el dispositivo puede confundir una actividad diaria por una caída fácilmente, por este motivo se implementa "Machine Learning Integrado" entrenando con la mayoría de las actividades que realiza una persona de la tercera edad y diferentes caídas, para esto se utiliza Edge Impulse ya que facilita el entrenamiento para la clasificación de actividades, además, permite crear modelos basados en librerías compatibles con Arduino. El modelo tiene la capacidad de detectar 91% de caídas, siendo un porcentaje aceptable.

- Mediante la investigación se determinó que el protocolo MQTT ofrece calidad del servicio permitiendo asegurar que el mensaje llegue a su destino, además tiene la capacidad de publicar nuevos mensajes sin la necesidad que el dispositivo final esté conectado y recibirá estos mensajes cuando este activo.

- En la selección de hardware se eligen a los dispositivos que poseen el protocolo de comunicación I2C configurados a una tasa de trasferencia de 100kbps, además tienen bajo consumo de energía con el fin de mejorar el tiempo de autonomía, alcanzando las 6 horas sin interrupciones, este tiempo depende de la capacidad que posee la batería.

- Se diseño una aplicación móvil mediante Flutter que permite ejecutar en Android e IOS, con capacidad para alertar ante un evento de caída o signos vitales que estén fuera del rango establecido para adultos mayores, mediante notificaciones push emitidas por el backend a todos los usuarios que están registrados bajo el mismo dispositivo IoT. Además, permite crear alertas sobre el estado de los signos vitales en un intervalo de tiempo, para que el cuidador del adulto mayor este al tanto de estos parámetros en caso de ser necesario.

4.2 Recomendaciones

Al concluir el trabajo de investigación se propone algunas recomendaciones:

- Para investigaciones futuras se recomienda mejorar el modelo de Machine Learning, esto con mayor cantidad de datos sobre caídas y datos de actividades diarias que realizan los adultos mayores.

- La frecuencia cardiaca es un parámetro vital que se toma por medio de la arteria radial con el uso del sensor no invasivo, por ende, está expuesto al ruido del movimiento, para mejorar este resultado se recomienda medir cuando la mano del paciente esta sin movimiento como recomiendan los profesionales de la salud o tomar este dato solo por la noche cuando el paciente está en cama.

- El tamaño de la batería que tiene el brazalete es extremadamente grande ya que en el mercado nacional no se encuentra la adecuada, para reducir el tamaño del brazalete se recomienda utilizar una batería de menor tamaño, adquiriendo mediante tiendas virtuales como es Amazon u otra tienda.

- Para determinar el rango normal de temperatura corporal en una persona de la tercera edad en específico se recomienda realizar un examen de signos vitales por un profesional de la salud ya que existen personas adultas que puede llegar la temperatura corporal normal por debajo de los 35 °C esto presentará confusión al dispositivo IoT.

BIBLIOGRAFÍA

[1] S. Sai, V. Sai y B. V, «Implementation of Compact Wearable Fall Detector for the Elderly,» *IEEE,* pp. 1-6, 2018.

[2] Luis Ruiz, «Sistema de telemedicina para monitoreo continuo de constantes vitales en lactantes menores para evitar el síndrome de muerte súbita,» Ambato, 2018.

[3] Ronny Montero, «Diseño e implementación de un sistema wireless sensors network personal para el monitoreo constante de signos vitales y visualización en tiempo real en aplicación web,» Sangolquí, 2019.

[4] B. Tirza, S. Luki y Bandung Yoanes, «IoT-Based fall detection and heart rate monitoring system for elderly care,» *IEEE,* pp. 1-6, 2019.

[5] E. Teran, «Sistema de monitoreo remoto y visualización para dispositivos de análisis de signos vitales orientados a E-HEALTH,» Sangolquí, 2019.

[6] Roberto Garcés, «Sistema de telemedicina con monitoreo de signos vitales en IoT en un ambiente smart TV,» Ambato, 2021.

[7] Andrés Carrillo, «Sistema de telemedicina basado en IoT para monitoreo de pacientes con enfermedades respiratorias,» Ambato, 2022.

[8] INEC, «Dirección población adulta mayor,» Quito, 2020.

[9] Ministerio de Inclusión Económica y Social, «Caracterización de Adultos Mayores,» Quito, 2018.

[10] OMS, «Caídas adultos mayores,» 2021.

[11] Pablo Álvarez, «Grandes síndromes geriátricos,» Quito, 2016.

[12] Organizacion mundial de la salud, «Caídas,» 26 Abril 2021. [En línea]. Available: https://www.who.int/news-room/fact-sheets/detail/falls.

[13] Corazón de Jesus, «Pensionado,» Guayaquil, 2020.

[14] OMS, «Informe mundial sobre el envejecimiento y la salud,» Luxemburgo, 2015.

[15] Paula Valdivia, «Envejecimiento y atención a la dependencia en Ecuador,» 2020.

[16] M. Tim y Mejía Iván, «El envejecimiento de la población en Ecuador: la revolución silenciosa.,» 2020.

[17] Salud Madrid, «Madrid Salud,» 2016. [En línea]. Available: https://madridsalud.es/los-accidentes-en-las-personas-mayores-i-los-accidentes-como-problema-de-salud/.

[18] L. Moreno, M. Díaz y Y. Artega, «Accidentes en el adulto mayor de un consultorio médico,» pp. 2-7, Marzo 2019.

[19] R. Criselda, G. María, G. Diana, G. Liliana, A. Marcela y Mino Dolores, «Factores intrínsecos y extrínsecos asociados con caídas en adultos mayores: estudio de casos y controles en México,» *Scielo,* pp. 134-135, 2020.

[20] Fhon Silva, «Causas y factores asociados a las caídas del adulto mayor,» *scielo,* vol. XVI, nº 1, p. 33, 2018.

[21] Alicia Durán, «Signos vitales,» Tandil, 2017.

[22] V. Juliana, V. Oscar y Villegas Valentina, Semiología de los signos vitales: Una mirada novedosa a un problema vigente, Caldas, 2012.

[23] Organización panamericana de la salud, «Aspectos técnicos y regulatorios sobre el uso de oxímetros de pulso en el monitoreo de pacientes con COVID-19,» Washington DC, 2020.

[24] Envagelina Pérez de la Plaza, Técnicas básicas de enfermería, Madrid, 2013.

[25] T. yoshikawa y D. Norman, «Fever in the Elderly,» 2018.

[26] Rodolfo Rodríguez, «Internet de las cosas: Futuro y desafío para la epidemiología y la salud pública,» *Scielo,* p. 254, 2019.

[27] Wipro, «Digital operations,» 2019. [En línea]. Available: https://www.wipro.com/business-process/what-can-iot-do-for-healthcare.

[28] Antonio Domínguez, «Diseño e implementación de una arquitectura IoT basada en tecnologías Open Source.,» Sevilla, 2016.

[29] S. Jordi y Silvestre Santiago, «INTERNET DE LAS COSAS,» 2019.

[30] L. José y Vilajosana Xavi, «IoT: Dispositivos, tecnologias de trasporte y aplicaciones,» 2017.

[31] Oscar Kowalewski, «Evaluación de protocolos limitados de nivel de aplicación para Internet de las Cosas,» Madrid, 2018.

[32] Jordi Salazar, «Redes Inalámbricas,» 2015.

[33] César Valencia, «Evaluación de tecnologías inalámbricas en redes de área doméstica para obtener la curva característica de carga en edificios inteligentes,» Quito, 2019.

[34] Espressif , «Espressif,» Julio 2022. [En línea]. Available: https://www.espressif.com/sites/default/files/documentation/esp-wroom-02u_esp-wroom-02d_datasheet_en.pdf.

[35] Espressif, «Espressif,» 2022. [En línea]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32e_esp32-wroom-32ue_datasheet_en.pdf.

[36] J. Bobadilla, Machine learning y deep learning usando python, scikit y keras, RA-MA Editorial, 2020.

[37] T. Oliver, Machine Learning for absolute beginners, 2017.

[38] P. Warden y D. Situnayake, Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers, 2019.

[39] I. Edge, «Hackster,» 2019. [En línea]. Available: https://www.hackster.io/news/tinyml-for-all-developers-with-edge-impulse-2cfbbcc14b90.

[40] E. Rust, «Edge Impulse,» 20 Mayo 2020. [En línea]. Available: https://www.edgeimpulse.com/blog/getting-started-with-edge-impulse.

[41] I. Edge, «Edge Impulse,» 2020. [En línea]. Available: https://docs.edgeimpulse.com/docs/edge-impulse-studio/processing-blocks.

[42] A. Bosch Rué, J. Casas Roma y T. Lozano Bagén, Deep learning: principios y fundamentos, UOC, 2019.

[43] Simplilearn, 18 Septiembre 2021. [En línea]. Available: https://www.simplilearn.com/tutorials/deep-learning-tutorial/what-is-keras.

[44] Edgeimpulse, 2019. [En línea]. Available: https://docs.edgeimpulse.com/docs/edge-impulse-studio/learning-blocks/classification.

[45] Docker, «Docker,» 2021. [En línea]. Available: https://docs.docker.com/get-started/overview/.

[46] EMQX, «EMQX,» 2021. [En línea]. Available: https://www.emqx.io/docs/en/v5.0/#benefits.

[47] Db-engines, 2020. [En línea]. Available: https://db-engines.com/en/system/Firebase+Realtime+Database%3BMongoDB%3BMySQL.

[48] Node.js, «Node.js,» 2021. [En línea]. Available: https://nodejs.org/en/about/.

[49] Bluewhaleapps, «bluewhaleapps,» 2020. [En línea]. Available: https://bluewhaleapps.com/blog/comparing-react-native-vs-flutter-vs-ionic-vs-kotlin-for-mobile-app-development.

[50] A. PIetri, 2018. [En línea]. Available: https://lastminuteengineers.com/mlx90614-ir-temperature-sensor-arduino-tutorial.

[51] Lastminute, 2019. [En línea]. Available: https://lastminuteengineers.com/max30100-pulse-oximeter-heart-rate-sensor-arduino-tutorial/.

[52] Componen, 2020. [En línea]. Available: https://components101.com/sensors/mpu6050-module.

[53] Espressif, 2021. [En línea]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf.

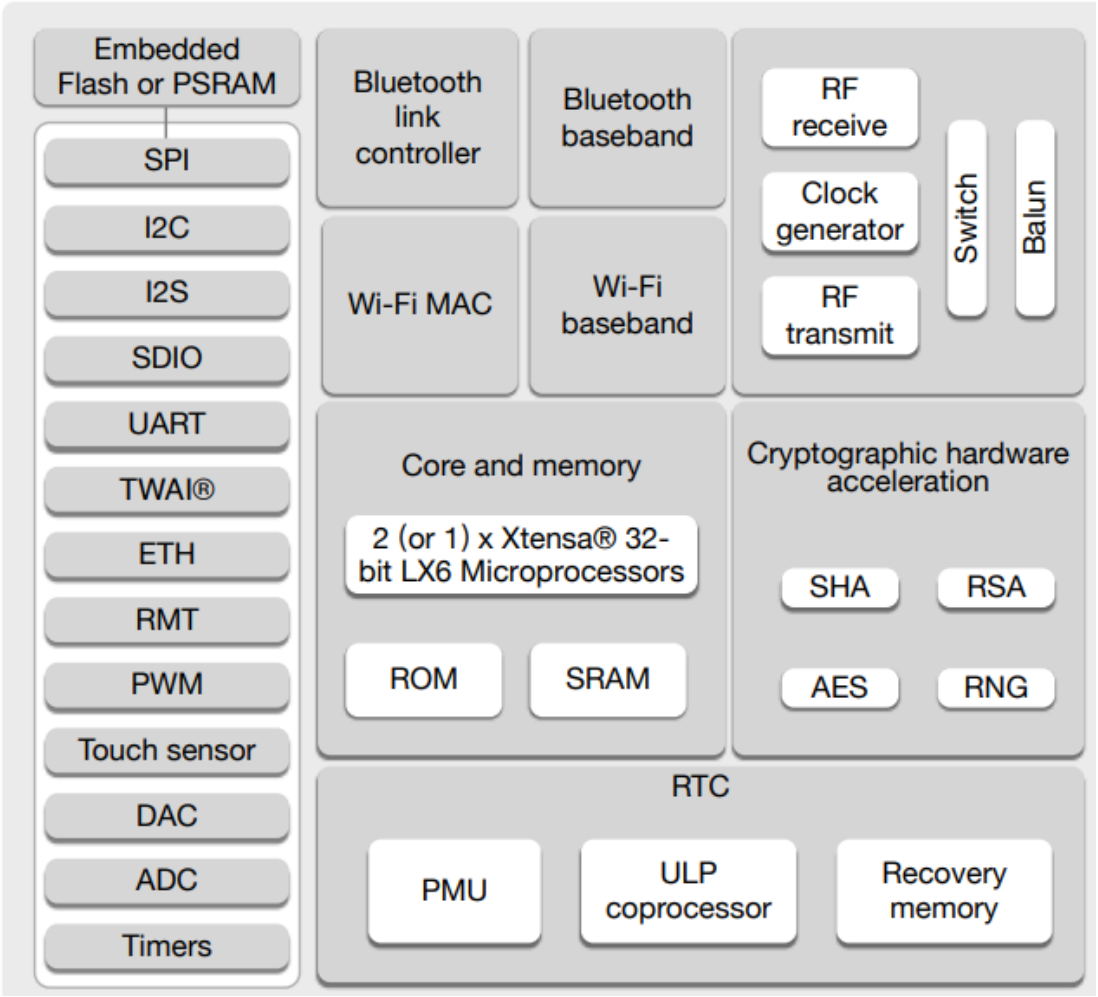
[54] A. Sucerquia, J. López y J. Vargas, «SisFall: A Fall and Movement Dataset,» MDPI, 2017.

[55] EMQX, «Rule Engine,» 2021.

[56] Y.-T. Kwak, Y. Lee y Y. You, «Statistical Estimation of Body Temperature from Skin Temperature for Smart Band,» 2019.

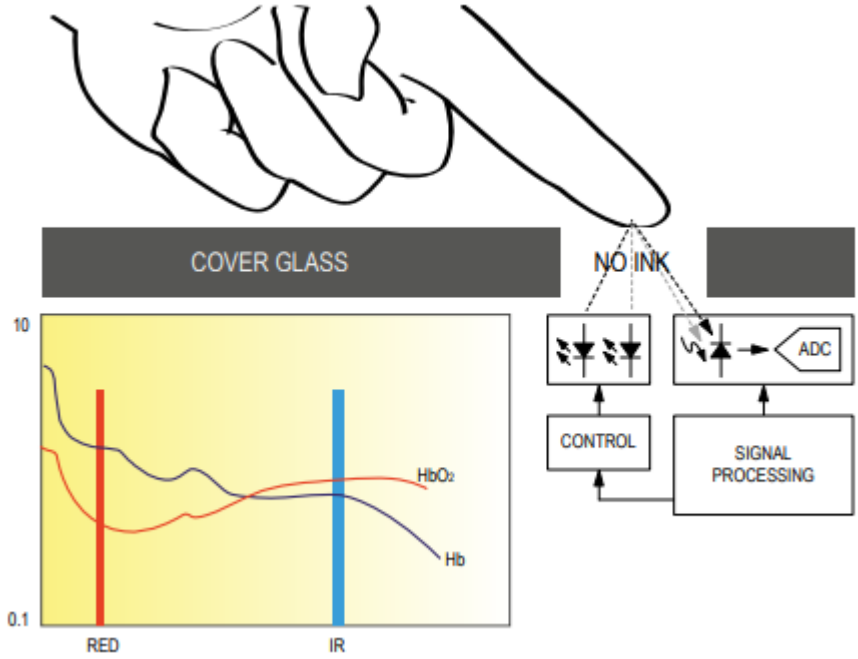
Anexos

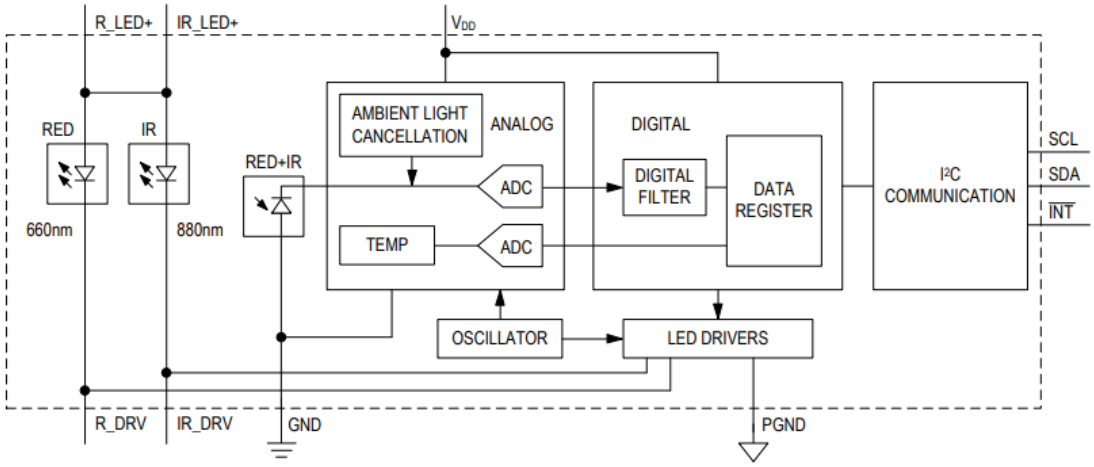
Anexo 1.- Diagrama de bloques de funcionalidad de la ESP32



Anexo 2.- Sensor MAX30100 y diagrama de funcionalidad

El sensor consta de longitudes de ondas específicas ya que con estas longitudes de ondas la hemoglobina oxigenada y desoxigenada poseen propiedades de absorción muy diferentes. Como se muestra en la siguiente figura, donde existe una gran diferencia entre Hb oxigenada (HbO2) y Hb desoxigenada (Hb)





Anexo 3.- Descripción de las herramientas de CLI de Edge Impulse

Todas las herramientas pueden ser ejecutadas a través de la consola de Windows como PowerShell o símbolo del sistema (CMD). Los path son rutas específicas donde se encuentra el archivo guardado para subir a Edge Impulse.

Herramienta	Descripción
edge-impulse-daemon	Usada para incorporar nuevos dispositivos a la plataforma, además es capaz de configurar los ajustes de carga, pero también puede actuar como proxy para dispositivos que no poseen conexión con una IP.
edge-impulse-uploader path/to/file/*.wav	Permite firmar archivos locales para luego cargar en el servicio de ingestión, es muy útil para migrar datos entre instancias de Edge Impulse o cargar conjunto de datos existentes.
edge-impulse-data-forwarder	El data forwarder se utiliza para transmitir datos del puerto serial de la computadora hacia Edge Impulse, es decir esta herramienta permite recopilar los datos de la conexión serial (COM), los firma y finalmente los envía a la plataforma Edge Impulse.
edge-impulse-run-impulse	Esta herramienta muestra los resultados de la ejecución de Edge Impulse en la placa de desarrollo. Tomará una pequeña muestra de datos de los sensores, clasifica los datos ingresados y finalmente imprime el resultado obtenido.
edge-impulse-blocks init	Permite crear diferentes tipos de bloques, utilizadas en funciones organizativas. Se puede crear bloques de transformación, bloques DSP personalizados, modelos de aprendizaje automático personalizados o bloques de implementación.
himax-flash-tool -f path/to/a/firmware.img	Esta herramienta permite cargar nuevos binarios a Himax WE-I Plus mediante una conexión serial.

Anexo 4.- Sketch para enviar datos del acelerómetro a Edge Impulse

En la parte del Setup se coloca las configuraciones necesarias para el sensor mientras que en el loop se obtiene todos los valores de acelerómetro en los ejes x, y, z y son enviados al puerto serial separados por una coma (,).

```
#include <Arduino.h>
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>
#include <Wire.h>

#define FREQUENCY_HZ 200
#define INTERVAL_MS  (1000 / (FREQUENCY_HZ + 1))
static unsigned long last_interval_ms = 0;
Adafruit_MPU6050 mpu;

void setup(void) {
  Serial.begin(115200);
  Serial.println(mpu.begin() ? F("IMU iniciado correctamente") :
F("Error al iniciar IMU"));
  mpu.setAccelerometerRange(MPU6050_RANGE_2_G);
  mpu.setGyroRange(MPU6050_RANGE_250_DEG);
  mpu.setFilterBandwidth(MPU6050_BAND_10_HZ);
  Serial.println("");
  delay(100);
}

void loop() {
  sensors_event_t a, g, temp;
  mpu.getEvent(&a, &g, &temp);
  if (millis() > last_interval_ms + INTERVAL_MS) {
    last_interval_ms = millis();
    Serial.print(a.acceleration.x);
    Serial.print(",");
    Serial.print(a.acceleration.y);
    Serial.print(",");
    Serial.println(a.acceleration.z);
  }
}
```

Anexo 5.- Grupo de participantes para recolección de actividades

Participantes utilizados para la fase de recolección de datos de caídas. Se utilizó a un grupo de adultos jóvenes para evitar lesiones o fracturas de huesos en personas de la tercera edad.

Participante	Peso (kg)	Altura (m)	Edad (años)
Participante 1	64	1.58	43
Participante 2	65	1.67	38
Participante 3	58	1.64	40
Participante 4	57	1.72	35
Participante 5	51	1.55	36
Participante 6	59	1.63	39
Participante 7	53	1.5	45
Participante 8	68	1.8	39
Participante 9	61	1.58	50
Participante 10	60	1.6	48

Anexo 6.- Código para ejecutar el modelo de Edge Impulse

```
1  void loop2(void *parameter){
 2   for (;;){
 3   sensors_event_t a, g, temp;
 4   mpu.getEvent(&a, &g, &temp);
 5   features[feature_ix++] = a.acceleration.x;
 6   features[feature_ix++] = a.acceleration.y;
 7   features[feature_ix++] = a.acceleration.z;
 8   if(feature_ix == EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE){
 9     signal_t signal;
10     ei_impulse_result_t result;
11     int err = numpy::signal_from_buffer(features,
12 EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE, &signal);
13   if (err != 0) {
14         ei_printf("Failed to create signal from buffer (%d)\n",
15 err);
16         return; }
17   EI_IMPULSE_ERROR res = run_classifier(&signal, &result, true);
18   if(res != 0) return;
19     for (size_t ix = 0; ix < EI_CLASSIFIER_LABEL_COUNT; ix++) {
20         if(result.classification[ix].value > 0.8){
21           if(result.classification[ix].label == "fall"){
22             ready_send_data_MPU = 1;
23           }
24           if(result.classification[ix].label == "AVD"){
25           }
26         }
27     }
28 #if EI_CLASSIFIER_HAS_ANOMALY == 1
29 #endif
30     feature_ix = 0;
31   }
32   max30100.update();
33   getHeart = max30100.getHeartRate();
34   getSpo2 = max30100.getSpO2();
35   readTemperature();
36 }} //for  //loop2
```

Anexo 7.- Docker compose (stack) para crear el contenedor de MongoDB

```
version: '3.8'
# volumes:
#   vol-emqx-data:
#     name: foo-emqx-data
#   vol-emqx-etc:
#     name: foo-emqx-etc
#   vol-emqx-log:
#     name: foo-emqx-log
services:
  mongo:
    container_name: mongodb
    image: mongo:4.4
    restart: always
    ports:
      - 27017:27017
    # links:
    #   - emqx
    volumes:
      - /etc/localtime:/etc/localtime:ro
    #   - /home/jeffersong/mongodata:/data/db
    environment:
      TZ: "America/Guayaquil"
      MONGO_INITDB_ROOT_USERNAME: "jeffersong"
      MONGO_INITDB_ROOT_PASSWORD: "jg0411"
```

Anexo 8.- Docker compose (stack) para crear el contenedor de EMQX

```
version: '3.8'
  emqx:
    container_name: emqx
    image: emqx/emqx:4.2.11
    restart: always
    ports:
      - 18083:18083
      - 1883:1883
      - 8883:8883
      - 8083:8083
      - 8085:8081
    volumes:
      - /etc/localtime:/etc/localtime:ro
    #   - vol-emqx-data:/opt/emqx/data
    #   - vol-emqx-etc:/opt/emqx/etc
    #   - vol-emqx-log:/opt/emqx/log
    links:
      - mongo
    environment:
      TZ: "America/Guayaquil"
      EMQX_NAME: jeffersong
      EMQX_HOST: 127.0.0.1
      # password dashboard
     EMQX_DASHBOARD__DEFAULT_USER__PASSWORD: "jg0411"
      #password API EMQX_MANAGEMENT__DEFAULT_APPLICATION__SECRET:
"jg0411"
      #topico
      EMQX_ALLOW_ANONYMOUS: "false"
      EMQX_NOMATCH: "deny"
      #?COnectar MOngo
      EMQX_AUTH__MONGO__TYPE: single
      EMQX_AUTH__MONGO__SERVER: "mongo:27017"
      EMQX_AUTH__MONGO__POOL: 8
      #*credenciales para ingresar a mongo
      EMQX_AUTH__MONGO__LOGIN: "jeffersong"
      EMQX_AUTH__MONGO__PASSWORD: "jg0411"
      EMQX_AUTH__MONGO__AUTH_SOURCE: admin
      #*Base de datos y colección
      EMQX_AUTH__MONGO__DATABASE: "proyect"
      EMQX_AUTH__MONGO__AUTH_QUERY__COLLECTION: "emqxauthrules"
      #*super usiario apagado
      EMQX_AUTH__MONGO__SUPER_QUERY__COLLECTION: "emqxauthrules"
      EMQX_AUTH__MONGO__SUPER_QUERY__SUPER_FIELD: "is_superuser"
      EMQX_AUTH__MONGO__SUPER_QUERY__SELECTOR: "username=%u"
      EMQX_AUTH__MONGO__SUPER_QUERY: "off"
      #*Cntraseña de usuario
      EMQX_AUTH__MONGO__AUTH_QUERY__PASSWORD_HASH: plain
      EMQX_AUTH__MONGO__AUTH_QUERY__PASSWORD_FIELD: "password"
      EMQX_AUTH__MONGO__AUTH_QUERY__SELECTOR: "username=%u"
      #*
      EMQX_AUTH__MONGO__ACL_QUERY: "on"
      EMQX_AUTH__MONGO__ACL_QUERY__COLLECTION: "emqxauthrules"
      EMQX_AUTH__MONGO__ACL_QUERY__SELECTOR: "username=%u"
```

Anexo 9.- Docker compose (stack) para crear el contenedor de Nodejs

```
version: "3.8"
services:
  node:
    container_name: node_tesis
    image: "node:14"
    restart: always
    environment:
      TZ: "America/Guayaquil"
    working_dir: /home/jeffersong/proyecto/server_node
    volumes:
      - /home/jeffersong/proyecto/server_node:
/home/jeffersong/Proyecto/server_node
    ports:
      - 3000:3000
    command: sh -c "npm run start:dev"
```

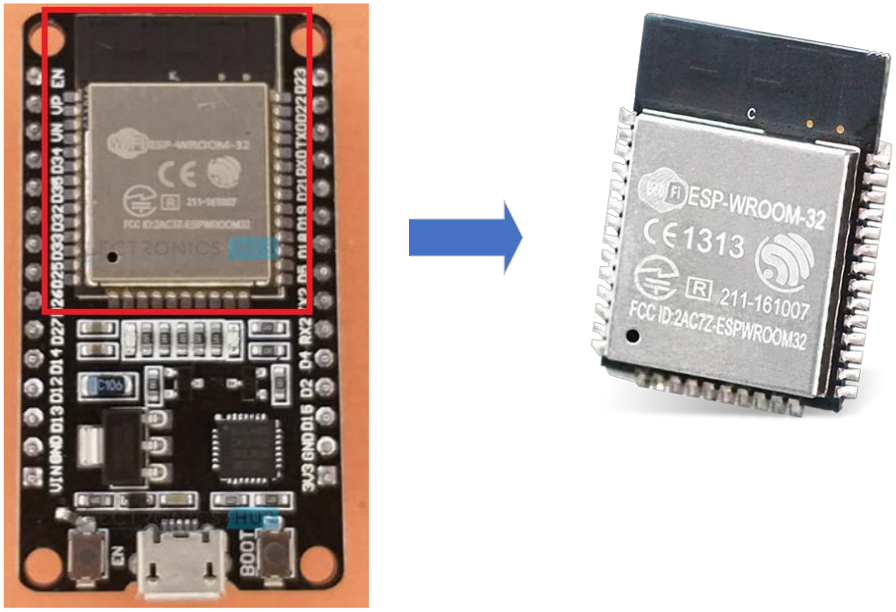
Anexo 10.- Clase para mapear la petición HTTP

Cuando un individuo registra un nuevo usuario o solamente inicia sesión, la respuesta de la petición HTTP es almacenada en esta clase, teniendo como variable al nombre, email y userId.

```
  import 'dart:convert';
Usuario usuarioFromJson(String str) =>
Usuario.fromJson(json.decode(str));
String usuarioToJson(Usuario data) => json.encode(data.toJson());
class Usuario {
    Usuario({
      required this.nombre,
      required this.email,
      required this.uid,
    });
    String nombre;
    String email;
    String uid;
    factory Usuario.fromJson(Map<String, dynamic> json) =>
Usuario(
        nombre: json["nombre"],
        email: json["email"],
        uid: json["uid"],
    );
    Map<String, dynamic> toJson() => {
        "nombre": nombre,
        "email": email,
        "uid": uid,
    };
  }
```

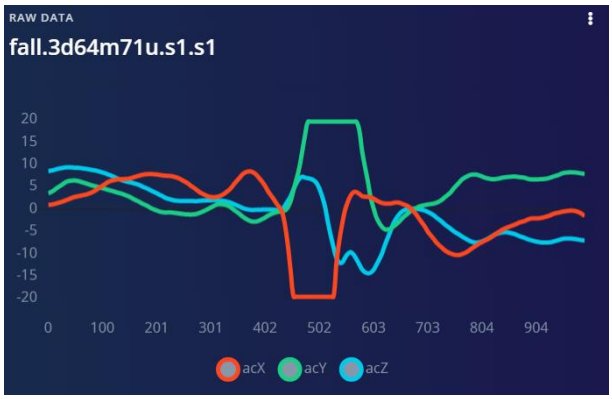
Anexo 11.- Desoldar ESP WROOM 32 de la placa de desarrollo

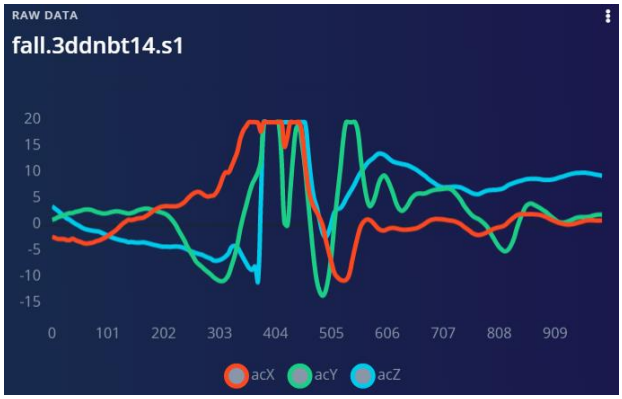
Para reducir el tamaño se procede a desoldar con una estación de calor el microcontrolador de la placa de desarrollo como muestra la figura. El pinout del microcontrolador está escrito en la parte posterior, esto viene de fábrica.



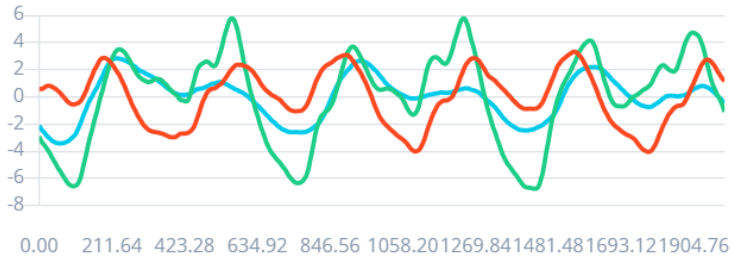
Anexo 12.- Respuesta del acelerómetro ante diversas actividades

Las gráficas presentan los valores de la aceleración ante un evento de caída, donde la aceleración en algunos ejes se mantiene constante en 20 m/s^2 por algunos milisegundos.

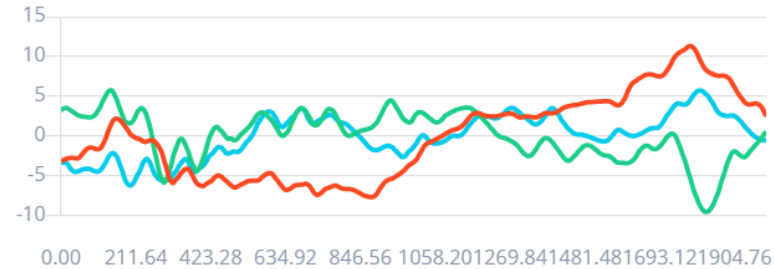




Gráfica con el participante trotando.



Gráfica de varios movimientos de la mano del participante.



Anexo 13.- Multímetro DT858L para medir corriente de consumo

Precisión del multímetro en el rango que se utilice, como en la tabla 27 la ESP32 llega a consumir una corriente máxima de 248mA se utiliza el multímetro en el rango de 10A con precisión del ±2%.

Rango	Resolución	Precisión
200µA	100nA	±1.8% lectura + 2 dígitos
2000µA	1µA	
20mA	10µA	
200mA	100µA	±2% lectura + 2 dígitos
10A	10mA	±2% lectura + 10 dígitos

Anexo 14.- Participantes para la fase pruebas

Los 10 participantes se utilizaron para la fase de pruebas, es decir para comprobar la precisión del modelo basado en “Machine Learning” creado en Edge Impulse.

Participantes	Peso (kg)	Altura (m)	Edad (años)
Participante 1	60	1.55	40
Participante 2	59	1.63	49
Participante 3	52	1.54	54
Participante 4	65	1.62	53
Participante 5	58	1.59	65
Participante 6	59	1.63	55
Participante 7	61	1.58	50
Participante 8	68	1.8	63
Participante 9	61	1.58	58
Participante 10	60	1.6	48

Anexo 15.- Termómetro digital mt-118

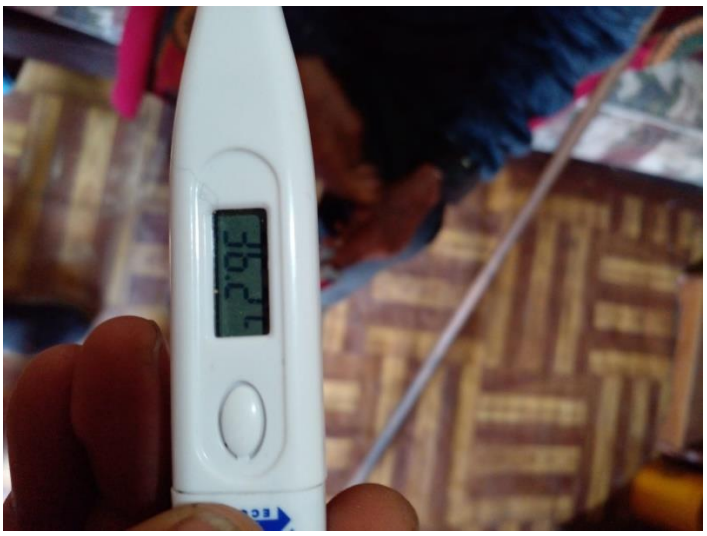
Se utilizó este instrumento para medir la temperatura corporal, dentro del rango de 96.4 °F a 97.9 °F con una precisión de ± 0.3°F.

Range	: 90.0~109.9°F (32.0~43.9°C) Below 90.0°F (32.0°C) displays L°F(°C) Above 109.9°F (43.9°C) displays H°F(°C)		
Resolution	: 0.1 °F (°C)		
Accuracy	: ± 0.2°F (98.0°F – 102.0°F)	± 0.1°C	(37.0°C - 39.0°C)
	± 0.3°F (96.4°F - 97.9°F)	± 0.2°C	(35.8°C - 36.9°C)
	± 0.3°F (102.1°F – 106.0°F)	± 0.2°C	(39.1°C - 41.0°C)
	± 0.5°F less than 96.4°F	± 0.3°C	less than 35.8°C
	± 0.5°F greater than 106.0°F	± 0.3°C	greater than 41.0°C
Display	: Liquid crystal display 3 1/ 2 digits		
Battery (included)	: Micro Alkaline battery 192, LR41 1.55V		
Power Consumption	: 0.15 milliwatts in measurement mode		
Battery Life	: More than 200 hours of continuous operation		
Dimensions	: 130mm x 18.5mm x 9.5mm		
Weight	: Approx. 11 grams including battery		
Beeper	: Approx. 8 sec. sound signal when peak temperature reached		
Working Conditions	: Temperature: 50 ~ 104°F (10 ~ 40°C)		
Relative Humidity	: 15 ~ 95% non condensing		
Storage Conditions	: Temperature: -13 ~ 131°F (-25~ 55°C)		
Relative Humidity	: 15 ~ 95% non condensing		
Guarantee of Quality	: Cert. ISO 13485; Complies w/ ASTM-E1112, EN12470-3, EN60601-1		
Product Classification	: Type BF equipment		

Anexo 16.- Resultados del termómetro digital.

La temperatura se toma en la axila de cada participante.

Participante 1



Participante 2

Anexo 17.- Oxímetro utilizado para comparar la frecuencia cardiaca y saturación de oxígeno.

Especificaciones técnicas

- Modo de visualización: OLED
- Visualización de la saturación de oxígeno: 70 ~ 100%, ± 2%
- Pantalla de potencia de pulso: 25 - 250 BPM, ± 1% o ± 1 BPM, y tome el más grande
- Fuente de alimentación: 2 pilas alcalinas AAA de 1,5 V; Rango de voltaje adecuado: 2.6 ~ 3.6V
- Consumo de energía: menos de 30 mA
- Precisión de la medida: la saturación de oxígeno entre 70% -99%, la precisión es ± 2%; y menos del 70% la precisión no tiene definición; la potencia del pulso es ± 1% o ± 1 BPM, y tome la más grande.
- Capacidad de medición en situación de perfusión débil: puede mostrar correctamente la saturación de oxígeno y la potencia del pulso cuando el grado de plenitud del pulso es del 0,6%.
- En comparación con el valor de medición en el cuarto oscuro, la saturación de oxígeno en la luz natural y el faro iluminado, la deformación es inferior a ± 1%.
- Tiene interruptor de función, y se detendrá automáticamente después de 8 segundos sin insertar el dedo.

Anexo 18.- Resultados de frecuencia cardiaca y saturación de oxígeno

Participante 1

Participante 2