UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

GRADUADO INGENIERÍA DE LA SALUD
MENCIÓN EN BIOINFORMÁTICA

# Pulsera de Seguimiento de Pacientes

## Patient Tracking Bracelet


Realizado por
De Pablo Diego

Tutorizado por:
Cristian Martín Fernández
Daniel Garrido Márquez

Departamento
Lenguajes y Ciencias de la Computación


MÁLAGA, junio de 2025

UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA DE LA SALUD
MENCIÓN INGENIERÍA BIOINFORMÁTICA

# Pulsera de Seguimiento de Pacientes

## Patient Tracking Bracelet

Realizado por
**De Pablo Diego**

Tutorizado por:
**Cristian Martín Fernández**
**Daniel Garrido Márquez**

Departamento
**Lenguajes y Ciencias de la Computación**

UNIVERSIDAD DE MÁLAGA
MÁLAGA, JUNIO DE 2025

# Resumen

**Palabras clave:**

# Abstract

**Keywords:**

# Agradecimientos

Apelando a la parte más humana de este trabajo, considero importante dejar constancia de aquellas personas que han sido fundamentales en mi crecimiento personal y académico durante estos años de universidad, y cuya influencia ha sido clave en la realización de este Trabajo de Fin de Grado.

En primer lugar, quiero agradecer al personal académico de la Universidad que muestran un esfuerzo e interés por ofrecer una formación de calidad. De forma especial, a mi tutor Cristian Martín, a quien valoro profundamente tanto como docente como por su calidad humana. Junto a él profesores como José Jerez, Rafael Luque, Carlos Bejines, Rafael Larrosa, David Bueno, entre otros. Han contribuido significativamente a mi desarrollo intelectual y personal. Gracias a su dedicación, hoy reafirmo con alegría la decisión de haber cursado esta carrera.

También quiero expresar mi gratitud a mis compañeros. Aunque sería imposible nombrar a todos, quiero destacar a Alejandro Silva y Mario Pascual, por su constante apoyo, por inspirarme a aspirar a más y por estar siempre presentes en los momentos clave de este recorrido.

En el ámbito familiar, nada de esto habría sido posible sin el respaldo incondicional de mis padres y, especialmente, de mi hermano.

Quiero dedicar una mención muy especial a mis abuelos, Régulo Depablos y Polia Moros, con quienes he compartido la evolución de este trabajo semana tras semana, como si ellos mismos fueran los pacientes. Con ellos he vivido la angustia de que un familiar sufra una caída y tener este TFG fue una oportunidad para tenerlos presentes en mi cabeza y darme cuenta la importancia que tienen.

Finalmente, rindo homenaje a mi tía Mireya Moros, quien falleció este año 2025. Aunque me pesa no haber podido visitarla en mi tierra natal, me reconforta saber que, a través de llamadas donde hablábamos de este trabajo, mi vida universitaria, etc. Se alegró que la tuviera tan presente al hacer este trabajo con ella en mente como posible paciente. Y pude disfrutar con ella en sus últimos meses.

Muchas gracias.

# Índice


**1. Introducción** **9**

1.1. Motivatión . . . 9

1.2. Objetivos . . . 10

1.2.1. Objetivo general . . . 10

1.2.2. Objetivos específicos . . . 11

1.3. Estructura de la memoria . . . 11

**2. Marco teórico** **15**

2.1. Ejemplos similares de dispositivos y teleasistencia médica . . . 15

2.1.1. SeniorDomo . . . 15

2.1.2. Bastón Pauto . . . 15

2.1.3. Teleasistencia avanzada en Castilla-La Mancha . . . 16

2.2. Sistemas de detección de caídas existentes . . . 16

2.2.1. Sistemas basados en umbrales . . . 16

2.2.2. Sistemas basados en aprendizaje automático . . . 17

2.3. Medición de constantes vitales en wearables . . . 17

**3. Análisis funcional y técnico** **18**

3.1. Metodología de diseño . . . 18

3.2. Requisitos . . . 19

3.2.1. Requisitos funcionales . . . 19

3.2.2. No funcionales . . . 20

3.3. Análisis de riesgo . . . 20

3.4. Stakeholders . . . 21

**4. Elección de componentes electrónicos** **22**

4.1. Microcontroladores IoT . . . 22

4.2. Acelerómetros y detección de caídas . . . 24

4.3. Sensor de pulso y oxímetro . . . 25

**5. Diseño de la pulsera (Hardware)** **27**

5.1. Esquemáticos y PCB . . . . . 27

5.2. Desarrollo de software del prototipo . . . . . 29

5.2.1. Fondos animados y botones de la pantalla (`handleButtonsAndAnimation`) 29

5.2.2. Reloj y sincronización NTP (`displayClock`, `initTime`) . . . 30

5.2.3. Sensor de pulso y oxímetro MAX30102 (`handleMAX`) . . . . . 30

5.2.4. Acelerómetro y giroscopio MPU-6050 (`handleMPU`) . . . . . 31

5.2.5. Conectividad MQTT y envío de datos (`sendMQTT`, `sendMetrics`) 31

5.3. Ejemplo en funcionamiento . . . . . 32

**6. Desarrollo de la plataforma web** **33**

6.1. Arquitectura general . . . . . 33

6.2. Back-end en Python . . . . . 33

6.3. Front-end en Svelte . . . . . 33

**7. Creación de la base de datos** **34**

7.1. Modelo entidad-relación . . . . . 34

7.2. Scripts de migración y carga inicial . . . . . 34

7.3. Amazon Web Services . . . . . 34

**8. Configuración e interconexión entre las partes** **36**

8.1. Protocolos de comunicación . . . . . 36

8.2. Pruebas de integración . . . . . 36

**9. Conclusiones y perspectivas futuras** **37**

9.1. Resultados principales . . . . . 37

9.2. Líneas de mejora . . . . . 37

**Apéndice A. Manual de usuario** **43**

A.1. Uso del prototipo de la pulsera . . . . . 43

A.2. Uso de la plataforma . . . . . 43

A.3. Configuración y despliegue del entorno con Docker . . . . . 43

A.4. Población y administración de la base de datos . . . . . 45

A.4.1. Administración de la base de datos con DBeaver . . . . . 45
A.5. Publicación de mensajes MQTT desde un ESP32 . . . . . 47
A.5.1. Preparación y prueba del broker MQTT . . . . . 48
A.5.2. Configuración de Mosquitto . . . . . 48
A.5.3. Uso de ESP32 . . . . . 49

# 1 Introducción

## 1.1. Motivatión

El progresivo envejecimiento demográfico subraya la necesidad de soluciones tecnológicas orientadas a las personas mayores. En España, los mayores de 65 años representan actualmente el 20,4 % de la población total, cifra que según el INE ascenderá a más del 30 % hacia mediados de siglo[1]. A escala europea la tendencia es similar: a principios de 2024 más de una quinta parte de la población de la UE tenía 65 o más años[2]. Este panorama anticipa un incremento sostenido de las enfermedades crónicas y de la dependencia, acentuando la carga asistencial en hogares y centros sanitarios.

Dentro de este colectivo vulnerable, las caídas suponen un grave problema de salud pública. En España viven unos 8,1 millones de personas ancianas, las cuales registran aproximadamente 4,3 millones de caídas al año, la mayoría de ellas en el propio domicilio[3]. Estudios recientes indican que alrededor de un tercio de los mayores de 65 años sufre al menos una caída anual, proporción que aumenta hasta el 50 % en mayores de 80. Del total de caídas, cerca del 9,3 % causan fracturas y más del 55 % requieren atención médica de urgencia. A nivel mundial, la Organización Mundial de la Salud destaca que las caídas son la segunda causa global de traumatismos accidentales (alrededor de 684 000 fallecimientos anuales) y que 37,3 millones de episodios de caídas exigen atención médica cada año; las tasas de mortalidad más altas corresponden a mayores de 60 años[4]. Estos datos ilustran la magnitud del problema: las caídas en ancianos no solo comprometen la salud individual, sino que constituyen un reto para el sistema sanitario.

Frente a este escenario, la tecnología *wearable*, elementos como relojes, collares, etc. que se puede llevar a todas partes, permitiendo monitorización continua y a distancia surge como una solución prometedora. Funciones como detectar in situ caídas mediante acele-

rometría y, simultáneamente, registrar valores de interés para el seguimiento y mejora del paciente (frecuencia cardíaca, saturación de oxígenos y nivel de actividad física). De este modo se habilita un soporte permanente del pacientes. Proyectos recientes respaldan esta aproximación: por ejemplo, La iniciativa privada Seniordomo ofrece un servicio de teleasistencia avanzada mediante wereable y monitoreo constante a través de una plataforma accesible para profesionales y pacientes[5].

El valor de esta tecnología se traduce además en mejoras concretas de calidad de vida. Al aumentar la seguridad del entorno, al sentirse acompañado a través de conexión permanente con familiares o servicios de salud. La persona mayor experimenta mayor tranquilidad y menor aislamiento[6]. Asimismo, al fomentar la autonomía domiciliaria (permitiendo quedarse más tiempo en casa pese a la fragilidad) se eleva la comodidad y el ánimo del anciano[6]. Desde el punto de vista sanitario, disponer de registros continuos de datos biomédicos que se comparten con los profesionales facilita la detección temprana de signos de alerta y permite actuar de forma preventiva, asimismo obtener más información del paciente. En conjunto, estas prestaciones tecnológicas apoyan a los cuidadores, aligerando su carga y promoviendo un modelo de atención más proactivo y centrado en la prevención.

El diseño de una pulsera con detección de caídas y monitorización de constantes vitales es una tecnología que cada vez cobrará mayor importancia. Permitirá abordar uno de los principales riesgos de salud en el envejecimiento poblacional, las caídas, reduciendo su impacto clínico y social. Esto enfocado a la par con la telemedicina ofreciendo un seguimiento de datos a cualquier paciente en el mundo abarca una solución para una problemática que abarcan cientos de millones de personas mayores al rededor del mundo.

## 1.2. Objetivos

### 1.2.1. Objetivo general

Desarrollar un prototipo de pulsera acompañado de una plataforma de telemedicina, que permita el seguimiento remoto de parámetros vitales (frecuencia cardíaca, saturación de oxígeno y actividad física) y la detección de caídas en personas mayores.

### 1.2.2. Objetivos específicos

- Diseñar un prototipo de la pulsera utilizando un microcontrolador, y sensores como pulsímetro, oxímetro, acelerómetro y giroscopio para la medición de constantes vitales y posibles caídas.
- Desarrollar algoritmos de detección de caídas basados en los datos obtenidos del acelerómetro y giroscopio, mediante la definición de umbrales y patrones que permitan diferenciar caídas.
- Implementar la plataforma de telemonitoreo:
  - **Back-end (Python):** Servidor REST encargado de recibir, procesar y almacenar los datos de la pulsera en una base de datos PostgreSQL. Incluirá mecanismos de autenticación y autorización para cada tipo de usuario (médico, familiar, cuidador).
  - **Front-end (Svelte):** Interfaz web intuitiva destinada a médicos, familiares y cuidadores, que muestre gráficas de evolución de parámetros vitales, notificaciones de alertas y resúmenes de actividad física, con opciones para configurar umbrales y contactos de emergencia.
- Integrar la comunicación Wi-Fi entre la pulsera y la plataforma, asegurando un flujo de datos seguro utilizando el protocolo MQTT.

## 1.3. Estructura de la memoria

La memoria se encuentra dividida en los siguientes capítulos, cada uno de los cuales aborda de manera detallada los aspectos fundamentales del proyecto:

- **Capítulo 1 y 2: Introducción y marco teórico.** En este primer capítulo se presenta el contexto general del Trabajo de Fin de Grado. Se describe la motivación que impulsa el desarrollo de una pulsera para el seguimiento de pacientes mayores, así como los objetivos generales y específicos del proyecto. Además, se incluye un breve análisis de precedentes y estudios relacionados, a su vez se evalúa la viabilidad técnica y económica.

- **Capítulo 3: Análisis funcional y técnico.** En este capítulo se definen los requisitos del sistema, tanto funcionales como no funcionales. Se identifican los stakeholders principales y se analizan sus necesidades. Además, se realiza un estudio de riesgos.
- **Capítulo 4 y 5: Elección de alternativas y diseño de la pulsera (Hardware).**
  - *Estudio de alternativas:* Se exponen las diferentes opciones de microcontroladores, sensores y baterías disponibles en el mercado que podrían cumplir con los requisitos definidos y se destaca las opciones elegidas.
  - *Diseño del esquemático:* Se detalla el esquema de la pulsera, señalando conexiones.
  - *Montaje físico y prototipado:* Se describe el proceso de soldadura y ensamblaje de componentes.
  - *Desarrollo de código arduino* Se explica la estructura modular del firmware en lenguaje C++ para Arduino, detallando las tareas principales: inicialización, lectura de sensores, procesamiento de datos para detección de caídas y cálculo de parámetros de frecuencia cardíaca y $SpO_2$, gestión de eventos de alerta y comunicación con la plataforma remota.
- **Capítulo 6 y 7: Desarrollo de la plataforma web.**
  - *Arquitectura general del sistema:* Se presenta la arquitectura cliente-servidor basada en microservicios, detallando componentes como el servidor de aplicación, la base de datos y el servidor de mensajería.
  - *Back-end (Python):* Se describen las tecnologías y frameworks elegidos (por ejemplo, FastAPI o Flask), la estructura de rutas RESTful, los modelos de datos para PostgreSQL y la lógica para procesar y almacenar las lecturas de la pulsera. También se documenta la implementación de mecanismos de autenticación y autorización para usuarios (médicos, familiares, cuidadores).
  - *Front-end (Svelte):* Se explican las decisiones de diseño de la interfaz de usuario, enfocado en la usabilidad para pacientes y personal sanitario. Se incluyen los

componentes principales: panel de control con gráficas de evolución de constantes vitales (frecuencia cardíaca, $SpO_2$, pasos diarios), lista de alertas en tiempo real y formularios para configuración de umbrales y datos de contacto.

- *Integración de alertas y notificaciones:* Se detalla cómo el back-end envía notificaciones (por correo electrónico, SMS o push) cuando se detectan eventos críticos (caídas o valores anormales), y cómo el front-end muestra históricos y estadísticas.

- **Capítulo 8: Configuración e interconexión entre las partes.**

  - *Protocolo MQTT:* Se justifica el uso de MQTT para la comunicación ligera entre la pulsera (cliente MQTT) y el servidor (broker MQTT), describiendo topologías de publicación/suscripción, calidad de servicio (QoS) y estrategias de reconexión ante pérdida de cobertura Wi-Fi.
  - *Contenedores con Docker:* Se presenta la creación de contenedores Docker para cada componente del sistema (broker MQTT, servidor Python, base de datos PostgreSQL, servidor web Nginx). Se incluye el archivo `docker-compose.yml`, que orquesta la puesta en marcha de todos los servicios con un solo comando, simplificando el despliegue en entornos de desarrollo y prueba.
  - *Servidor Nginx y proxy inverso:* Se explica la configuración de Nginx para servir la aplicación web Svelte y hacer de proxy inverso hacia el back-end Python, habilitando HTTPS con certificados autofirmados o Let's Encrypt durante la fase de pruebas.
  - *Uso de Redis para entornos de desarrollo:* Se describe la integración de Redis como sistema de caché o broker de mensajes alternativo (por ejemplo, para manejo de alertas en tiempo real), y cómo se configura un entorno de desarrollo con Docker Compose que incluye Redis para pruebas de baja latencia.

- **Capítulo 9: Conclusiones y perspectivas futuras.** En este capítulo se recogen los resultados obtenidos tras la validación del prototipo: métricas de precisión en la detección de caídas, fiabilidad de las lecturas de frecuencia cardíaca y $SpO_2$, rendimiento de la plataforma web y feedback obtenido de pruebas de usabilidad. Se

analizan las limitaciones detectadas (consumo energético, cobertura Wi-Fi, ajustes en algoritmos de detección) y se proponen posibles mejoras, como la incorporación de baterías de mayor capacidad, algoritmos de inteligencia artificial para predicción de eventos críticos o la extensión de la plataforma a dispositivos móviles nativos.

- **Apéndice (Manual de uso).** En el Apéndice se incluyen documentos de soporte para el usuario final y el equipo técnico:

  - *Manual de uso de la pulsera:* Ejemplo de uso del prototipo hardware, además de principales indicaciones y ejemplos de las funcionalidades.
  - *Manual de uso de la plataforma web:* Ejemplo de uso de plataforma de telemedicina enfocando todas sus funciones y utilidades.
  - *Guía de instalación del entorno de desarrollo:* Pasos detallados para desplegar la plataforma mediante Docker, configuración de variables de entorno.
  - *Población de la base de datos en entorno de desarrollo* Guía para poblar la base de datos mediante un script de profesionales de la salud y pacientes a partir de datos falsos, para probar la plataforma en un entorno de desarrollo.

# 2
# Marco teórico

Es importante contextualizar y tomar de guía algunos de los proyectos que tienen una relación directa con la idea planteada, al ser una problemática de gran importancia se parte de una base ya explorada.

## 2.1. Ejemplos similares de dispositivos y teleasistencia médica

Actualmente existen múltiples propuestas para pulseras o equipos de seguimiento de pacientes de avanzada edad, el mayor problema de las soluciones es la dificultad en la globalización de un sistema que pueda ser adaptado a cualquier paciente.

### 2.1.1. SeniorDomo

SeniorDomo ofrece un servicio de teleasistencia avanzada para personas mayores que permite hacer un seguimiento continuo y preventivo. Utiliza un reloj para ofrecer múltiples funciones como detección de situaciones de riesgos (inactividad, pulso anómalo o caídas), botones de ayuda, localización GPS y permite procesar llamadas. [5].

### 2.1.2. Bastón Pauto

Bastón Pauto enfocado en personas mayores con Parkinson y Alzheimer cuenta con un botón de emergencia y detección de caídas mediante la proyección láser de una línea verde y un punto rojo, combinado con la vibración de la empuñadura del bastón, permite que el usuario supere el episodio de bloqueo de la marcha cuando éste se produce. [7].

### 2.1.3. Teleasistencia avanzada en Castilla-La Mancha

Además de iniciativas privadas también existen ejemplos como la Junta de Comunidades de Castilla-La Mancha que ha implementado un sistema de teleasistencia avanzada que atiende actualmente a más de 81.000 personas. Este sistema incluye dispositivos que detectan riesgos como caídas, incendios o escapes de gas, y utiliza inteligencia artificial para analizar rutinas y detectar alteraciones que puedan indicar situaciones de riesgo [8].

## 2.2. Sistemas de detección de caídas existentes

La detección de caídas en personas mayores es un área de investigación activa, con múltiples enfoques tecnológicos desarrollados para abordar este problema de salud pública. A continuación, se describen algunos de los sistemas más relevantes:

### 2.2.1. Sistemas basados en umbrales

Estos sistemas utilizan acelerómetros para monitorizar los movimientos del usuario y detectar patrones que indiquen una caída. La detección se basa en establecer umbrales específicos de aceleración que, al ser superados, activan una alerta. Por ejemplo, en un TFG de la UCM sobre el desarrollo de un sistema de detección de caídas basado en acelerómetros se evalúa la magnitud máxima del vector de aceleración, calculada como:

$$|\mathbf{a}| = \sqrt{a_x^2 + a_y^2 + a_z^2}$$

donde $a_x$, $a_y$ y $a_z$ son las componentes de aceleración en los ejes ortogonales X, Y y Z. Según los datos empíricos del estudio, las caídas provocan picos transitorios de aceleración de hasta 12 g, mientras que el valor mínimo de aceleración observado en una caída es de aproximadamente 3 g. Las actividades de la vida diaria (ADL) raramente superan los 3 g, aunque movimientos bruscos como correr o sentarse rápidamente pueden acercarse a este umbral. Debido al solapamiento entre los rangos de aceleración de caídas y ADL, el umbral de detección debe ajustarse cuidadosamente a cada paciente para minimizar falsos positivos y garantizar la robustez del algoritmo [28].

### 2.2.2. Sistemas basados en aprendizaje automático

El aprendizaje automático permite mejorar la precisión en la detección de caídas al analizar patrones complejos de movimiento. Un sistema desarrollado utiliza datos de sensores como acelerómetros para calcular características que luego son clasificadas por algoritmos como SVM o árboles de decisión, alcanzando una precisión del 96 % en la detección de caídas [10].

## 2.3. Medición de constantes vitales en wearables

Los dispositivos wearables han evolucionado para incluir sensores que permiten la monitorización continua de constantes vitales, lo cual es especialmente útil en la atención de personas mayores. Algunos de los parámetros comúnmente medidos incluyen:

- **Frecuencia cardíaca:** Utilizando sensores ópticos (fotopletismografía), los wearables pueden medir la frecuencia cardíaca en tiempo real.
- **Saturación de oxígeno ($SpO_2$):** A través de sensores similares a los utilizados en pulsioxímetros, se puede estimar la saturación de oxígeno en la sangre.
- **Actividad física:** Los acelerómetros y giroscopios integrados permiten registrar la cantidad y tipo de actividad física realizada por el usuario.

La integración de estos sensores en dispositivos como pulseras inteligentes facilita el seguimiento remoto de la salud de los pacientes, permitiendo intervenciones tempranas y mejorando la calidad de vida de las personas mayores [12].

# 3

# Análisis funcional y técnico

Durante el transcurso de la carrera universitaria ingeniería de la salud otorga múltiples herramientas, conocimientos y capacidades para enfrentarse a diversos retos, no obstante de las cualidades más importantes es la correcta aplicación de ingeniería de requisitos.

La especificación y gestión de requisitos es clave para el éxito de proyectos multidisciplinarios de informática e IoT, al facilitar la comunicación entre stakeholders y mejorar la calidad del sistema final [13, 14]. Para garantizar especificaciones claras y trazables, se adoptan las buenas prácticas propuestas en IEEE 830−1998 y la norma ISO/IEC/IEEE 29148 [15, 16].

## 3.1. Metodología de diseño

Para el desarrollo de este proyecto se ha optado por una *metodología de diseño iterativo*, un enfoque cíclico que combina fases sucesivas de análisis, diseño, implementación y evaluación, de forma que cada iteración entrega un módulo funcional susceptible de validación y ajuste temprano [13, 16]. Este proceso permite detectar y corregir errores de manera precoz, incorporar cambios en los requisitos con agilidad y entregar incrementos de valor continuo, mejorando la adaptabilidad.

Además, la naturaleza modular de cada iteración fomenta buenas prácticas de ingeniería de software, tales como la alta cohesión y el bajo acoplamiento, al dividir el sistema en componentes independientes que pueden desarrollarse y probarse en paralelo. La retroalimentación al iterar garantiza que el producto evolucione alineado a las necesidades reales, optimizando recursos y asegurando una mayor calidad final [13].

## 3.2. Requisitos

Un requisito es una condición o capacidad que el sistema debe satisfacer para cubrir las necesidades de usuarios y partes interesadas. Se clasifican en funcionales, describen lo que el sistema debe hacer, y no funcionales, atributos de calidad del sistema. [17].

### 3.2.1. Requisitos funcionales

Los requisitos funcionales definen los servicios y comportamientos que el sistema debe ofrecer para cumplir sus objetivos [15]. En este proyecto, los requisitos se describen mediante los siguientes atributos:

- **ID:** Código único que identifica el requisito (e.g., RF1).
- **Descripción:** Enunciado breve de la función a realizar.
- **Obligatoriedad:** "Obligatorio" u "Opcional" según prioridad de implementación.
- **Dependencia:** Requisitos previos necesarios para su correcto funcionamiento.
- **Trazabilidad:** Referencia a la sección del documento donde se detalla.

Cuadro 1: Principales Requisitos Funcionales

| **ID** | **Descripción** | **Oblig.** | **Depend.** | **Trazabilidad** |
|---|---|---|---|---|
| RF1 | Registro de usuarios por rol (administrador, médico, paciente). | Obligatorio | — | Sección TODO |
| RF2 | Manejo de tokens y credenciales. | Obligatorio | RF1 | Sección TODO |
| RF3 | Visualización de datos del paciente. | Obligatorio | — | Sección TODO |
| RF4 | Almacenamiento y consulta de datos en la base de datos. | Obligatorio | - | Sección TODO |
| RF6 | Manejo de alertas | Obligatorio | RF3 | Sección TODO |
| RF7 | Mensajería entre los usuarios. | Opcional | RF1 | Sección TODO |
| RF8 | Gestión del perfil. | Opcional | RF1 | Sección TODO |

### 3.2.2. No funcionales

Los requisitos no funcionales establecen las propiedades de calidad que el sistema debe cumplir [15].

Cuadro 2: Principales requisitos no funcionales

| ID | Descripción | Oblig. | Depend. | Trazabilidad |
|---|---|---|---|---|
| RNF1 | Autonomía entre pulsera, base de datos y plataforma. | Obligatorio | RF4 | Sección TODO |
| RNF2 | Implementación de lógica modular facilitando sustitución de piezas. | Obligatorio | - | Sección TODO |
| RNF3 | Simplicidad en la plataforma. | Opcional | RF3 | Sección TODO |
| RNF4 | Cifrado de datos personales. | Obligatorio | RF2, RF4 | Sección TODO |

## 3.3. Análisis de riesgo

Al implementar un proyecto multidisciplinario con hardware, firmware y plataforma web, existen riesgos clave que deben gestionarse para asegurar el éxito. El siguiente cuadro resume los principales riesgos, su probabilidad e impacto, y las estrategias de mitigación.

Cuadro 3: Análisis de Riesgos del TFG

| Id | Descripción | Probabilidad / Impacto | Mitigación |
|---|---|---|---|
| R1 | Dedicación y plazos insuficientes para cubrir todo el trabajo. | Alta / Media | Definir un cronograma detallado; priorizar tareas críticas; revisiones periódicas de avance. |
| R2 | Sensores de baja calidad generan datos poco fiables. | Muy alto / Bajo | Modularizar el diseño para facilitar su sustitución. |
| R3 | Vulnerabilidades en la seguridad y privacidad de datos. | Bajo / Alto | Cifrado en base de datos; cumplir normativas de protección de datos. |
| R4 | Problema de producción de la pulsera | Alto / Muy Bajo | Centrarse en un prototipo funcional sin enfocarse en la ergonomía |

## 3.4. Stakeholders

Un stakeholder (o parte interesada) es cualquier persona, grupo u organización que tiene un interés o impacto directo en el proyecto.

Cuadro 4: Stakeholders Principales

| **Nombre** | **Representa** | **Rol** |
|---|---|---|
| Administrador (Desarrollador de software) | Responsable de la implementación y mantenimiento del firmware de la pulsera y la plataforma web. | Configura funcionalidades, garantiza la integridad de los datos y la comunicación entre dispositivo y servidor. |
| Personal médico | Profesionales de la salud que atienden a pacientes con riesgo cardíaco o de edad avanzada. | Monitorean constantes vitales y caídas, reciben alertas y ajustan tratamientos en tiempo real. |
| Paciente | Usuario final que lleva la pulsera para seguimiento. | Registra su actividad y constantes, consulta resultados, recibe alertas y comunica síntomas al médico. |

# 4

# Elección de componentes electrónicos

En el mercado existen numerosas opciones que cumplen los requisitos de este proyecto, un microcontrolador con conectividad Wi-Fi y sensores capaces de medir aceleraciones para detección de caídas, así como señales ópticas para frecuencia cardíaca y saturación de oxígeno. A continuación se justifica la elección de cada componente para el caso del prototipo, pero en el trabajo se favorece lógica modular que permita el futuro cambio de componentes en caso de necesitar trabajar con mejores sensores.

## 4.1. Microcontroladores IoT

Para el prototipo de la pulsera se ha seleccionado la placa **TTGO T-Display (16 MB)** basada en el módulo ESP32. Este microcontrolador de bajo coste resulta ideal por integrar, en un único módulo, una pantalla TFT de 1.14 " (135×240 px), gestión de batería Li-Ion y la inclusión de conectividad inalámbrica permite la transmisión de datos, mientras que la pantalla facilita una interfaz amigable para el usuario sin hardware adicional[23].

**Este módulo permite [23]:**

- **MCU:** ESP32 Xtensa dual-core LX6 (240 MHz). Este procesador de alto rendimiento permite gestionar en paralelo la adquisición de datos de sensores y la actualización de la interfaz gráfica.
- **Memoria Flash:** 16 MB, preferible para mostrar un interfaz con fondo de pantalla animado y capacidad de almacenaje de datos.

- **Pantalla:** 1.14 pulgadas, ST7789V IPS TFT, 135×240 px, 260 PPI, interfaz SPI
- **Conectividad inalámbrica:** Wi-Fi 802.11 b/g/n y Bluetooth 4.2 + BLE, esenciales para la transmisión de datos a la plataforma.
- **Alimentación:** 3.3 V, circuito de carga Li-Ion TP4054 y detección de nivel de batería, garantizando autonomía y avisos de carga.
- **Periféricos:** 2 botones (Ver figura 1, GPIO L006 e I007) configurables como botones de emergencia, y capacidad para integrar sensores adicionales.

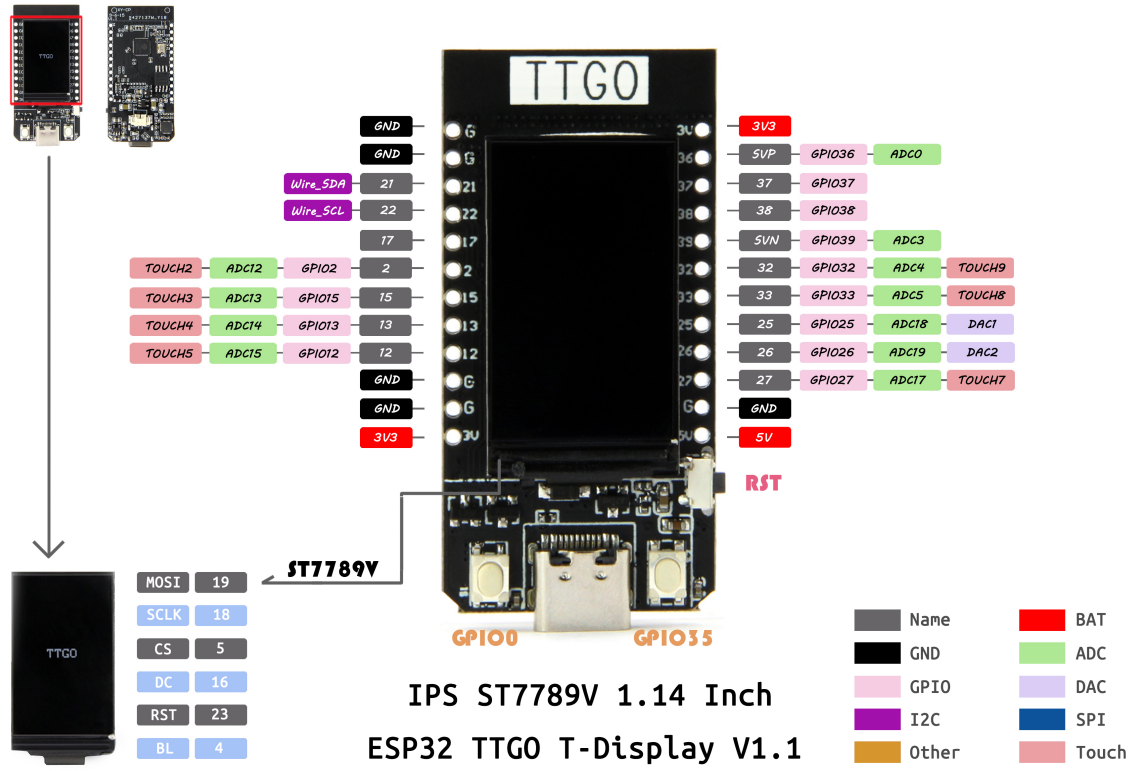


Figura 1: Diagrama de la placa TTGO T-Display (16 MB) con todos sus pines de conexión.

**Alternativas consideradas:**

- **ESP32 DevKitC / NodeMCU / LOLIN32**: mismo ESP32 de bajo coste pero no cuenta con pantalla, botones ni gestión de batería (coste aproximado: 5–8 €) [24].

- **Arduino Nano 33 IoT / UNO WiFi Rev2**: ARM Cortex-M0/M4 con Wi-Fi/BLE integrados, pero sin pantalla y más caro (coste aproximado: 30 €) [25].
- **M5Stack Core2**: basado en ESP32, incluye pantalla touchscreen de 2 pulgadas (320×240 px), y las funcionalidades pero por un mayor coste. (coste aproximado: 50 €) [26].

## 4.2. Acelerómetros y detección de caídas

Para la detección de caídas en el prototipo se ha seleccionado el IMU **MPU-6050**, un dispositivo de bajo coste (aproximadamente 1 €) de 6 ejes que combina acelerómetro y giroscopio con conversión A/D de 16 bit y un DMP (Digital Motion Processor) integrado. **Este módulo permite [27]:**

- **Medición de aceleración:** ±2/4/8/16 g, suficiente para capturar picos de impacto y cambios bruscos de movimiento en caídas .
- **Medición de rotación:** ±250/500/1000/2000 °/s, útil para caracterizar la orientación y la fase de giro durante una caída.
- **DMP integrado:** fusión inicial de acelerómetro y giroscopio en hardware, que descarga parte del procesamiento de la MCU y permite ejecutar algoritmos de detección en tiempo real con menor latencia.
- **Interfaz I²C:** funciona a hasta 400 kHz, simplificando el cableado y compartiendo bus con otros sensores (p. ej. MAX30102).

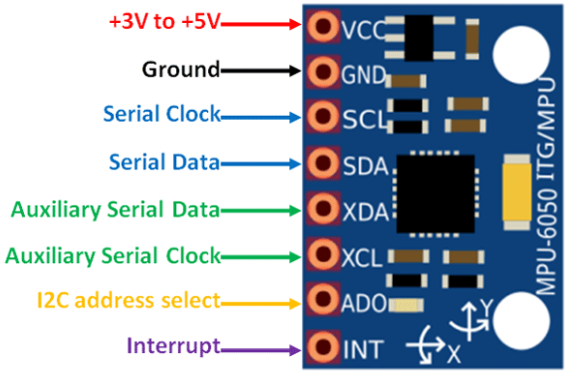


Figura 2: Diagrama del MPU6050) con todos sus pines de conexión.

**Alternativas consideradas:**

- **ADXL345**: acelerómetro de 3 ejes, ±2/4/8/16 g, A/D de 13 bit, interfaz $I^2C$/SPI; buena precisión estática pero requiere un giroscopio externo para fusión completa su coste es similar al MPU6050[30].
- **BNO055**: IMUs de 9 ejes con magnetómetro o fusión completa en chip; ofrece mayor certeza en orientación espacial, pero a un coste superior (coste aproximado 10 €) [31].

## 4.3. Sensor de pulso y oxímetro

Para la medición de frecuencia cardíaca y saturación de oxígeno ($SpO_2$) se ha seleccionado el módulo **GY-MAX30102**, que integra el sensor óptico MAX30102 con las siguientes características [32]:

- **Longitudes de onda:** LED rojo a 660 nm e infrarrojo a 880 nm, adecuadas para la discriminación hemoglobínica.
- **Conversión A/D:** resolución de 18 bit, que permite un amplio rango dinámico de señal.
- **Interfaz $I^2C$:** bus estándar a 400 kHz para comunicación con la MCU.
- **Dimensiones y consumo:** encapsulado compacto (5×3 mm) y consumo de operación típico en torno a 600 µA por LED.

**Ventajas:**

- Muy económico (coste aproximado 1 a 2 €).
- Diseño modular que facilita su sustitución futura por sensores de mayor precisión.

**Limitaciones:**

- No homologado para uso clínico; su precisión es inferior a la de equipos médicos profesionales.
- Algunas placas genéricas presentan fallas de diseño, lo que impide su funcionamiento [33].

**Alternativas consideradas:**

- **MAX30100**: predecesor con A/D de 16 bit y menor resolución, fue probado y termino siendo descartado. [34].
- **MAX86150**: sensor combinado (ECG + PPG) con mayor precisión y funcionalidades integradas (coste aproximado 15 €) [35].
- **TSL2561 + pulsioxímetro genérico**: uso de fotodetector independiente para PPG, que aumenta la complejidad de diseño y calibración (coste aproximado 12€).

# 5

# Diseño de la pulsera (Hardware)

El objetivo es presentar una solución sencilla y modular que sirva como base. Aún usando sensores de bajo coste, se mantienen buenas prácticas de ingeniería: arquitectura en bus $I^2C$, y disposición clara de pines, de manera que el prototipo pueda evolucionar hacia un producto más robusto en caso de ser necesario y tener el presupuesto para comprar sensores de calidad.

## 5.1. Esquemáticos y PCB

A continuación se muestra la tabla de conexiones entre el ESP32 y los sensores MAX30102 y MPU-6050. Solo se emplean las líneas estrictamente necesarias, dejando libres pines adicionales que no son requeridos para este proyecto.

Cuadro 5: Conexión del sensor de pulso GY-MAX30102

| **Pin del MAX30102** | **Pin del ESP32** | **Uso** |
|---|---|---|
| VIN | 3.3 V | Alimentación |
| GND | GND | Tierra |
| SCL | GPIO 22 (SCL) | $I^2C$ reloj |
| SDA | GPIO 21 (SDA) | $I^2C$ datos |
| INT | — | No utilizado en este prototipo |

Para simplificar el cableado y aprovechar el bus $I^2C$, ambos sensores comparten las mismas líneas de reloj (SCL) y datos (SDA) en el ESP32. En particular, GPIO 22 (SCL) y GPIO 21 (SDA) pueden conectarse en paralelo a múltiples dispositivos $I^2C$ sin interferencias, ya

que el protocolo gestiona direcciones únicas para cada sensor, por lo cual no es ningún problema reutilizar estos pines.

Cuadro 6: Conexión del IMU MPU-6050

| **Pin del MPU-6050** | **Pin del ESP32** | **Uso** |
|---|---|---|
| VCC | 3.3 V | Alimentación |
| GND | GND | Tierra |
| SCL | GPIO 22 (SCL) | $I^2C$ reloj |
| SDA | GPIO 21 (SDA) | $I^2C$ datos |
| XDA | — | No necesario (uso $I^2C$) |
| XCL | — | No necesario (uso $I^2C$) |
| ADO | — | Dirección $I^2C$ fija (0x68) |
| INT | — | No utilizado en este prototipo |

En la figura 3, se puede observar un esquemático realizado en Fritzing, mostrando todas las conexiones que se esperan hacer para permitir el funcionamiento del microcontrolador y sensores.

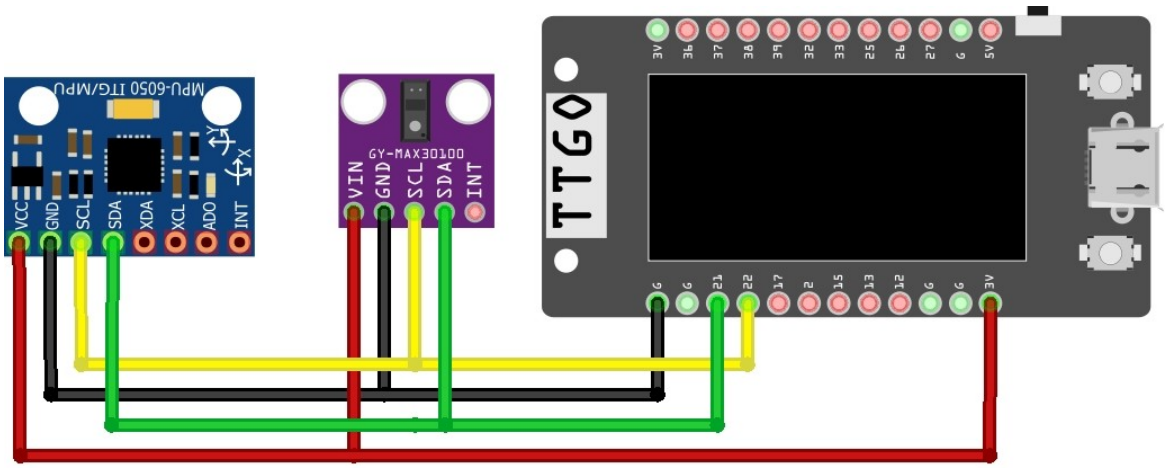


Figura 3: Esquemático de conexión entre el ESP32 y los sensores MAX30102 y MPU-6050.

## 5.2. Desarrollo de software del prototipo

Ya conociendo los componentes elegidos para este proyecto se puede describir el código usado en arduino. El código fuente del prototipo está disponible como material complementario en la carpeta CODE de este repositorio: https://github.com/Diegodepab/TFG_Pulsera_Seguimiento_Pacientes_Avanzada_Edad/tree/main/COD

Uno de los principales objetivos del prototipo es explorar una variedad amplia de funciones probando su viabilidad, además se tuvo un enfoque basado en componentes donde sin importar lo amplio del código se busca minimizar la cohesión diseñando módulos de tal manera que tengan menos dependencias internas entre sí, favoreciendo un futuro donde sustituciones o problemas en un sensor no afecten el funcionamiento global de la pulsera.

El prototipo se divide en las siguientes funcionalidades:

### 5.2.1. Fondos animados y botones de la pantalla (`handleButtonsAndAnimation`)

Para gestionar la pantalla TFT y los botones integrados del ESP32 hemos elegido la librería `TFT_eSPI`. Esta biblioteca está optimizada para controladores comunes ILI9xxx y ofrece:

- **PushImage eficiente:** permite enviar directamente mapas de bits desde PROGMEM sin consumir RAM.
- **Configuración SPI sencilla:** abstrae la inicialización de pines y velocidad de bus.
- **Funciones gráficas básicas:** dibujo de texto, rectángulos y cambio rápido de colores.

La rutina `handleButtonsAndAnimation()` unifica dos tareas clave:

1. **Animación de fondo:**
    - Se incluyen diez frames estáticos (`frame_000.h`…`frame_009.h`) generados previamente con un script Python que convierte GIFs en arrays de 16-bit.

- Mediante `millis()` y la constante FRAME_INTERVAL = 100 ms, cada ciclo se llama a `tft.pushImage(0,0,160,235,frames[frameIndex])` para avanzar al siguiente frame.

2. **Gestión de botones de emergencia:**
   - Pines GPIO0 y GPIO35 configurados con INPUT_PULLUP.
   - Al liberar botones se restaura automáticamente el fondo animado.

### 5.2.2. Reloj y sincronización NTP (`displayClock`, `initTime`)

Esta funcionalidad añade un reloj en pantalla y timestamps fiables para los datos:

- **Conexión Wi-Fi:** `WiFi.begin(SSID,PASSWORD)` con reconexión automática.
- **Sincronización NTP:** mediante `configTime(0,0,NTP1,NTP2)` se obtiene la hora UTC de servidores públicos.
- **Visualización "HH:MM":** cada iteración de `loop()` se llama a `displayClock()`, que dibuja la hora en la esquina superior con `tft.print()`.

### 5.2.3. Sensor de pulso y oxímetro MAX30102 (`handleMAX`)

- WAITING: sin dedo detectado.
- ANALYSING: recogida de BUFFER_SIZE muestras.
- SHOW_RESULTS: tras retirar dedo, mostrar promedios finales.

**Procesado de muestras:**

- Relleno inicial muestra a muestra.
- Agrupación en bloques de FreqS, desplazamiento de buffers con `shiftBuffers()`, luego cálculo con `maxim_heart_rate_and_oxygen_saturation()`.
- Filtrado incremental y promedio de valores válidos (validHR, validSPO2).

**Salida:** actualización en pantalla con `displayRealtime()` y envío por Serial/MQTT.

#### 5.2.4. Acelerómetro y giroscopio MPU-6050 (`handleMPU`)

Para conteo de pasos y detección de caídas:

- **Inicialización I²C:** `Wire.begin(21,22)` y calibración con `mpu.calcOffsets()`.
- **Detección de caídas:** si la variación absoluta en X, Y o Z supera FALL_THRESHOLD = 3.5 g, se muestra alerta roja y se notifica por MQTT.
- **Conteo de pasos:**
  - Buffer circular de tamaño 20 en eje Z para media móvil.
- **Visualización:** el contador de pasos y estado de detección se actualizan en pantalla en tiempo real.

#### 5.2.5. Conectividad MQTT y envío de datos (`sendMQTT`, `sendMetrics`)

Para integrar la pulsera con una plataforma de monitoreo:

- **Cliente MQTT:** `PubSubClient` sobre `WiFiClient` con reconexión automática.
- **Eventos de alerta:**
  - ALERTA_BOTONES y ALERTA_CAIDA se disparan en los manejadores de botones y MPU, incluyendo coordenadas o tipo de pulsación.
- **Formato JSON estándar:** incluye siempre el campo `ts` con timestamp ISO y los valores medidos, facilitando la ingesta en cualquier broker o dashboard.

### 5.3. Ejemplo en funcionamiento

Aqui vendría una explicación de lo que se ve funcionalmente con fotografías del esp32 en acción.

# 6

# Desarrollo de la plataforma web

## 6.1. Arquitectura general

## 6.2. Back-end en Python

## 6.3. Front-end en Svelte

# 7

# Creación de la base de datos

## 7.1. Modelo entidad-relación

## 7.2. Scripts de migración y carga inicial

## 7.3. Amazon Web Services

Amazon Web Services (AWS) se ha consolidado como la plataforma de computación en la nube líder a nivel mundial, ofreciendo una amplia gama de servicios gestionados que facilitan la implementación, escalado y operación de aplicaciones distribuidas de forma eficiente y segura. En particular, Amazon Simple Storage Service (S3) permite el almacenamiento de objetos con alta durabilidad, disponibilidad y rendimiento, características esenciales en entornos donde el volumen y la criticidad de los datos son elevados [20]. El dominio de herramientas como AWS es una necesidad cada vez mayor puesto que proporciona la capacidad de gestionar grandes volúmenes de datos en cualquier tipo de proyecto.

Para ilustrar el proceso de aprovisionamiento y aplicar buenas prácticas de seguridad, en este trabajo se ha procedido de la siguiente manera: por un lado, se creó la cuenta raíz de AWS y se activó la autenticación multifactor (MFA) para reforzar la protección del acceso administrativo; a continuación, se accedió al servicio IAM y se dio de alta un usuario dedicado —`tfm-s3-user`— con acceso programático (solo Access Key ID y Secret Access Key), asignándole un grupo con permisos mínimos (principio de menor privilegio) para listar, leer y escribir en el bucket `bracelet-tests` [21]. Finalmente, se generaron y documentaron las claves de acceso (con etiqueta descriptiva) y se adoptaron prácticas recomendadas como la rotación periódica de las claves, la exclusión de credenciales del control

de versiones y la preferencia por roles de servicio (en EC2/ECS/Lambda). Tal y como lo recomienda la propia plataforma de Amazon Web Services.

Una vez obtenidas las variables `API_S3_ACCESS_KEY_ID` y `API_S3_SECRET_ACCESS_KEY`, su integración en una [22].

# 8

# Configuración e interconexión entre las partes

## 8.1. Protocolos de comunicación

## 8.2. Pruebas de integración

# 9

# Conclusiones y perspectivas futuras

## 9.1. Resultados principales

## 9.2. Líneas de mejora

- Mejorar la detección de caídas aplicando aprendizaje automatico o definiendo posibles tipos de caída y su gravedad (por ejemplo: caída lateral, de sentado, de pie)

# Referencias


[1] Instituto Nacional de Estadística (INE). Proyecciones de población. Años 2024–2074 [Internet]. Nota de prensa. Madrid: INE; junio del 2024 [citado 20 de mayo del 2025]. Disponible en: https://www.ine.es/dyngs/Prensa/PROP20242074.htm

[2] Pereira IT. La población mayor de 65 años no deja de crecer: ¿Está Europa en una crisis demográfica? [Internet]. Euronews España; 21 de febrero del 2025 [citado 20 de mayo del 2025]. Disponible en: https://es.euronews.com/embed/2758258

[3] Rodríguez-Molinero A. Caídas en la población anciana española: incidencia, consecuencias y factores de riesgo [Internet]. Rev Esp Geriatr Gerontol. 2015 Nov–Dec;50(6):274–280. doi:10.1016/j.regg.2015.05.005.

[4] Organización Mundial de la Salud. Caídas [Internet]. Ginebra: OMS; 2021 abr 26 [citado 20 de mayo del 2025. Disponible en: https://www.who.int/es/news-room/fact-sheets/detail/falls

[5] Vilá J. SeniorDomo ofrece un servicio de teleasistencia avanzada para personas mayores que permite hacer un seguimiento continuo y preventivo [Internet]. Universitat Oberta de Catalunya; 2021 jul 27 [citado 21 de mayo del 2025]. Disponible en: https://www.uoc.edu/es/news/2021/205-seniordomo-teleasistencia

[6] Reina Miranda D. Telemedicina como medida de prevención de caídas en personas mayores en sus domicilios: revisión bibliográfica [Internet]. Enferm Cuid. 2024;7:63–77. doi:10.51326/ec.7.7019787 [citado 15 de mayo del 2025]. Disponible en: https://enfermeriacuidandote.com/article/view/6027/7367

[7] Nodal I. Pauto. el bastón inteligente para personas con párkinson [Internet]. Telecinco; 29 jul 2019 [citado 29 de mayo del 2025]. Disponible en: https://www.telecinco.es/noticias/salud/baston-inteligente-ayuda-tratamiento-parkinson_18_2791170223.html

[8] López Huerta C. ¿Cómo funciona la teleasistencia avanzada en Castilla-La Mancha? [Internet]. Cadena SER; 2025 jun 2 [citado 7 de junio del 2025]. Disponible en: https://cadenaser.com/castillalamancha/2025/06/02/como-funciona-la-teleasistencia-avanzada-en-castilla-la-mancha-ser-toledo/

[9] González Vega R. Universidad Complutense de Madrid. (UCM) Desarrollo de un sistema de detección de caídas basado en acelerómetros. 13 de junio de 2016. Disponible en: https://hdl.handle.net/20.500.14352/66068

[10] Lauro Juarez Quispe F. *Detección de caídas para personas mayores mediante aprendizaje automático*. 2021. Disponible en: https://es.scribd.com/document/580251061/Deteccion-de-caidas-para-personas-mayores-mediante-aprendizaje-automatico

[11] Asociación Mexicana de Ciencias. *Detección inteligente de caídas para el cuidado de los adultos mayores*. 2023. Disponible en: https://www.revistaciencia.amc.edu.mx/images/revista/76_2/PDF/07_76_2_1686_DeteccionCaidas.pdf

[12] Rodríguez-Molinero A, Narvaiza L, Gálvez-Barrón C, de la Cruz JJ, Ruíz J, Gonzalo N, Valldosera E, Yuste A. *Caídas en la población anciana española: incidencia, consecuencias y factores de riesgo*. Rev Esp Geriatr Gerontol. 2015;50(6):274–280. doi:10.1016/j.regg.2015.05.005

[13] Karl E. Wiegers & Joy Beatty. *Software Requirements*, 3rd ed., Microsoft Press, 2013. https://www.microsoftpressstore.com/store/software-requirements-9780735679665

[14] Klaus Pohl. *Requirements Engineering: Fundamentals, Principles, and Techniques*, Springer, 2010. https://link.springer.com/book/9783642125775

[15] IEEE Computer Society. *IEEE Recommended Practice for Software Requirements Specifications*, IEEE Std. 830–1998, 1998. https://standards.ieee.org/ieee/830/1222/

[16] ISO/IEC/JTC1/SC7 & IEEE. *Systems and software engineering – Life cycle processes – Requirements engineering*, ISO/IEC/IEEE 29148:2018. https://www.iso.org/standard/72089.html

[17] Ian Sommerville. *Software Engineering*, 9th ed., Addison-Wesley, 2011. Cap. 4: Functional and non-functional requirements. https://engineering.futureuniversity.com/BOOKS%20FOR%20IT/Software-Engineering-9th-Edition-by-Ian-Sommerville.pdf

[18] Bashar Nuseibeh & Steve Easterbrook. "Requirements engineering: a roadmap," en *ICSE '00: Proceedings of the 22nd International Conference on Software Engineering*, 2000, pp. 35–46. DOI: https://doi.org/10.1145/336512.336523

[19] Donald L. Gibbons. "Defining and managing quality attributes in a software architecture," en *ICSE '02: Proceedings of the 2002 International Conference on Software Engineering*, 2002, pp. 567–576. https://doi.org/10.1145/581339.581423

[20] Amazon Web Services, Inc. *Amazon Simple Storage Service Developer Guide*, 2025. https://docs.aws.amazon.com/s3/index.html

[21] Amazon Web Services, Inc. *IAM Best Practices*, 2025. https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html

[22] S. Fernández-Luque and S. P. Imran, "Health Informatics on the Cloud: Privacy and Security Considerations", *Journal of Medical Internet Research*, vol. 23, no. 4, 2024.

[23] LilyGO. TTGO T-Display ESP32 development board [Internet]. 2021 [citado 20 may 2025]. Disponible en la página oficial de lilygo: https://lilygo.cc/products/lilygo%C2%AE-ttgo-t-display-1-14-inch-lcd-esp32-control-board

[24] Espressif Systems. ESP32 DevKitC Getting Started Guide [Internet]. 2018 [citado 20 may 2025]. Disponible en: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/hw-reference/esp32-get-started-devkitc.html

[25] Arduino. Arduino Nano 33 IoT [Internet]. 2019 [citado 20 may 2025]. Disponible en: https://store.arduino.cc/usa/arduino-nano-33-iot

[26] M5Stack Technology Co., Ltd. Core2 for AWS IoT EduKit – ESP32 Development Kit [Internet]. Shenzhen: M5Stack; 2020 [citado 20 may 2025]. Disponible en: https://docs.m5stack.com/en/core/core2

[27] TDK InvenSense. MPU-6050 Product Specification [Internet]. 2013 [citado 20 may 2025]. Disponible en: https://invensense.tdk.com/products/motion-tracking/6-axis/mpu-6050/

[28] Universidad Complutense de Madrid. Desarrollo de un sistema de detección de caídas basado en acelerómetros [Internet]. 2015 [citado 20 may 2025]. Disponible en: https://eprints.ucm.es/38704/1/MemoriaTFG.pdf

[29] García J, López M. CareFall: detección de caídas mediante machine learning en smartwatch [Internet]. 2020 [citado 20 may 2025]. Disponible en: https://doi.org/10.1234/carefall.2020.001

[30] Analog Devices. ADXL345 Datasheet [Internet]. 2018 [citado 20 may 2025]. Disponible en: https://www.analog.com/media/en/technical-documentation/data-sheets/ADXL345.pdf

[31] Bosch Sensortec. BNO055 Intelligent 9-axis Absolute Orientation Sensor [Internet]. 2016 [citado 20 may 2025]. Disponible en: https://www.bosch-sensortec.com/products/smart-sensors/bno055/

[32] Maxim Integrated. MAX30102 Pulse Oximeter and Heart-Rate Sensor IC [Internet]. 2018 [citado 20 may 2025]. Disponible en: https://datasheets.maximintegrated.com/en/ds/MAX30102.pdf

[33] Lluís Llamas. Pulsímetro y oxímetro con Arduino y MAX30102 [Internet]. 2021 [citado 20 may 2025]. Disponible en: https://www.luisllamas.es/pulsimetro-y-oximetro-con-arduino-y-max30102/

[34] Maxim Integrated. MAX30100 Pulse Oximeter and Heart-Rate Sensor IC [Internet]. 2015 [citado 20 may 2025]. Disponible en: https://datasheets.maximintegrated.com/en/ds/MAX30100.pdf

[35] Maxim Integrated. MAX86150 Pulse Oximeter and Heart-Rate Sensor IC with ECG [Internet]. 2019 [citado 20 may 2025]. Disponible en: https://datasheets.maximintegrated.com/en/ds/MAX86150.pdf

# Apéndice A
# Manual de usuario

## A.1. Uso del prototipo de la pulsera

## A.2. Uso de la plataforma

## A.3. Configuración y despliegue del entorno con Docker

En esta sección se describe el proceso de preparación y despliegue del entorno de desarrollo y pruebas mediante Docker, cabe acotar que el cambio de sistema operativos pueden afectar ciertos permisos en scripts de shell, en caso de muchos problemas se recomienda contactar por correo. El proyecto esta pensado para ser fácilmente lanzado desde cualquier equipo una vez se haya configurado el docker.

### Requisitos de software

- **Sistema operativo:**
  - *Linux nativo:* Ubuntu 24.04 LTS (recomendado).
  - *Windows con WSL:* Subsistema de Windows para Linux (Ubuntu 24.04 en WSL).
- **Docker Engine:** última versión (recomendado). estable[1].
- **Docker Compose:** v1.29.2 (recomendado)[2].
- **Git:** para clonar el repositorio y manejo de archivos.

*Nota:* Python, PostgreSQl, Redis, etc. no necesitan instalación local, quedan encapsulados en Docker.

[1] https://docs.docker.com/engine/install/
[2] https://docs.docker.com/compose/install/

**Preparación del entorno**

1. Clonar y situarse en la carpeta de despliegue:

   **Clonado del repositorio**

   ```
   git clone https://github.com/diegodepab/track-bracelet.git
   cd track-bracelet/devops/docker/all
   ```

2. Generar el fichero de entorno:

   **Generación de .env**

   ```
   cp env.template .env
   ```

   Editar a continuación .env para añadir credenciales (Por motivos de seguridad nunca se publicarían claves de S3 u otros datos comprometedores) y ajustar la **IP**.

**Despliegue** El script `launch.sh` se encarga de ejecuta todo lo referente a devops, aunque pueda ser ejecutado sin banderas (Arranca os servicios en primer plano sin reconstruir los contenedores ni limpiar los volúmenes) admite estos flags:

| Flag | Descripción |
|---|---|
| `-h` | Mostrar ayuda (lista de flags) y salir. |
| `-r` | Reconstruir contenedores antes de lanzar, ideal para primera ejecución. |
| `-c` | Limpiar volúmenes (base de datos nueva). |
| `-d` | Ejecutar en modo *detached* (contenedores en segundo plano, sin bloquear la terminal). |

Cuadro 7: Opciones del script `launch.sh`.

**Flujo recomendado:**

1. Limpiar en caso de ejecución previa y sea de interés comprobar desde el inicio:

   **Limpiar entorno**

   ```
   ./launch.sh -c
   ```

2. Reconstruir y lanzar:

**Reconstruir y lanzar**

```
./launch.sh -r
```

3. Verificar estado:

**Ver estado**

```
docker-compose ps
```

4. Comprobar los **logs** de docker generados, en caso de estar en orden podrás acceder al localhost desde cualquier navegador y ver la plataforma.

## A.4. Población y administración de la base de datos

Tras haber lanzado el servicio, es de interés administrar la base de datos para comprobar y tener acceso a todas las funcionalidades.

### Cargar datos

1. Revertir al estado base:

**Rollback con Alembic**

```
docker-compose exec api alembic downgrade base
```

2. Aplicar migraciones:

**Upgrade con Alembic**

```
docker-compose exec api alembic upgrade head
```

3. Cargar datos de seed.py (datos de prueba para desarrollo):

**Ejecución de `seed.py`**

```
docker-compose exec api python ./seed.py
```

*Recomendación:* siempre hacer `downgrade` previo en desarrollo para partir de base limpia.

### A.4.1. Administración de la base de datos con DBeaver

Para facilitar la exploración y gestión de los datos en PostgreSQL de forma gráfica, podemos utilizar DBeaver, un cliente SQL de código abierto que soporta múltiples bases

de datos.

**1. Instalación de DBeaver** Descarga e instala la versión Community desde su web oficial. [3].

**2. Creación de una nueva conexión**

1. Abre DBeaver y selecciona

   Database → New Database Connection.

2. En el asistente, elige el driver **PostgreSQL** y pulsa Next.

3. Rellena los parámetros de conexión con los datos de tu contenedor:

   - **Host:** `localhost` (o `la ip del .env` si accedes desde otra máquina)
   - **Port:** `15432`
     (Este puerto mapea el 5432 interno del contenedor al 15432 de tu máquina: `-p 15432:5432` en Docker)
   - **Database:** `bracelet`
   - **Username:** `admin`
   - **Password:** `TFGde10`

4. Opcionalmente, revisa la URL JDBC que aparecerá como:

   **Log de docker**

   postgresql://admin:TFGde10@localhost:15432/bracelet

5. Haz clic en Test Connection para comprobar la conectividad y, si es exitosa, pulsa Finish.

**3. Exploración y gestión de datos**

- Una vez conectados, en el panel de *Database Navigator* expande el esquema `public` para ver tablas, vistas y otros objetos.

[3] https://dbeaver.io/

- Haz doble clic sobre cualquier tabla para abrirla en modo editor de datos, donde podrás:
  - Navegar filas, filtrar y ordenar.
  - Editar registros directamente.
  - Ejecutar consultas SQL personalizadas en la pestaña *SQL Editor*.
- Para ejecutar scripts o migraciones, utiliza la ventana de SQL y los atajos de DBeaver (por defecto `Ctrl+Enter` para ejecutar la sentencia seleccionada).

## A.5. Publicación de mensajes MQTT desde un ESP32

En esta sección se describe cómo montar un entorno para manejar el protocolo MQTT en un PC (Windows o Linux) con Mosquitto y cómo configurar un ESP32 para que publique mensajes.

### Requisitos previos

- **Mosquitto Broker o similares:** instalado en el sistema anfitrión Windows, Linux o en WSL.[4]
- El broker de escucha debe tener permisos para escuchar (recomendablemente el puerto TCP 1883), existen antivirus, cortafuegos y demás sistemas de protección del dispositivo que complican el uso de los puertos a aplicaciones de terceros.
- **Placa ESP32:** con soporte Wi-Fi y capacidad de correr el código.
- **Red Wi-Fi** recomendable una red estable preferiblemente compatible con estándares de diversas generaciones (hay múltiples esp32 que son incompatibles con el estándar 802.11ax)
- **Librerías Arduino:**
  - `WiFi.h` (incluida en el core ESP32).
  - `PubSubClient` (cliente MQTT de Nick O'Leary).[5]

[4] https://mosquitto.org/
[5] https://github.com/knolleary/pubsubclient

### A.5.1. Preparación y prueba del broker MQTT

Una vez tengas mosquitto configurado y funcionando, abre una terminal para suscribirse a un topic MQTT que sirva para recibir y mostrar los mensajes:

**Suscripción en Linux/WSL/Powershell**

```
mosquitto/sub -h localhost -t pulsera/test -v
```

Para comprobar que Mosquitto esta funcionando se puede hacer una prueba enviando un mensaje en el localhost.

**Publicación de prueba**

```
mosquitto_pub -h localhost -t pulsera/test -m "Test local OK"
```

Si todo esta en orden la terminal donde compilaste el broker mostrará el mensaje "Test local OK", ahora repite la prueba usando la dirección IPv4 de tu equipo.

### A.5.2. Configuración de Mosquitto

Si el broker no acepta conexiones externas, edita su configuración para escuchar en todas las interfaces y permitir conexiones anónimas:

- *En las configuraciones por defecto*: añade al inicio de `/etc/mosquitto/mosquitto.conf`:

  **mosquitto.conf**

  ```
  listener 1883 0.0.0.0
  allow_anonymous true
  ```

  Luego reinicia el servicio:

  **Reiniciar Mosquitto**

  ```
  sudo systemctl restart mosquitto
  ```

- *En caso de configuraciones personales:* crea un archivo `user_mosq.conf` con:

  **user_mosq.conf**

  ```
  listener 1883 allow_anonymous true
  ```

  Y arranca Mosquitto en PowerShell:

**user_mosq.conf**

```
cd Ç: Program Files mosquitto"
. mosquitto.exe -c . user_mosq.conf -v
```

**Apertura del puerto 1883 en el cortafuegos de Windows**

1. Pulsa `Win + R`, escribe `wf.msc` y pulsa Enter.
2. Selecciona *Reglas de entrada* y haz clic en *Nueva regla...*.
3. Elige *Puerto*, selecciona TCP y especifica `1883`.
4. Marca *Permitir la conexión* y pulsa *Siguiente*.
5. Selecciona los perfiles deseados (Dominio, Privado, Público) y pulsa *Siguiente*.
6. Pon un nombre a la regla (por ejemplo, `MQTT TCP 1883`) y finaliza.

También puedes hacerlo por PowerShell ejecutando como administrador:

**PowerShell (administrador)**

```
netsh advfirewall firewall add rule ‘
name="MQTT Inbound"
dir=in action=allow
protocol=TCP localport=1883
profile=Domain,Private,Public
```

### A.5.3. Uso de ESP32

Usa código del repositorio github o de ejemplos de las librerías y sustituyendo `<SSID>`, `<contraseña del Wi-fi>` y `<IP del BROKER>` deberías ser capaz de conectar tanto al wifi como al protocolo MQTT, y mientras se mantiene en uso mosquitto, el esp32 ya debería ser capaz de enviar los mensajes deseados según su programació