UNIVERSIDAD DE MÁLAGA

ESCUELA DE INGENIERÍAS INDUSTRIALES

ESCUELA DE INGENIERÍAS INDUSTRIALES

Departamento de Ingeniería de Sistemas y Automática.

Ingeniería de Sistemas y Automática.

TRABAJO FIN DE GRADO

SISTEMA DE BAJO CONSUMO PARA MONITORIZACIÓN REMOTA DE CAMINADORES

Doble Grado en

Ingeniería Mecánica y Eléctrica

Autor: PABLO AGUILAR ORELLANA

Tutor: JESÚS MANUEL GÓMEZ DE GABRIEL

Cotutor JUAN MANUEL GANDARIAS PALACIOS

MÁLAGA, 24 de Agosto de 2020.

UNIVERSIDAD DE MÁLAGA

ESCUELA DE INGENIERÍAS INDUSTRIALES

DECLARACIÓN DE ORIGINALIDAD DEL PROYECTO/TRABAJO FIN DE GRADO

D./ Dña.: *Pablo Aguilar Orellana.*

DNI/Pasaporte: *79037776R* Correo electrónico: *ing.pablouma@gmail.com*

Titulación: *Doble Grado de Ingeniería Mecánica y Eléctrica.*

Título del Proyecto/Trabajo: *Sistema de bajo consumo para monitorización remota de caminadores.*

DECLARA BAJO SU RESPONSABILIDAD

Ser autor/a del texto entregado y que no ha sido presentado con anterioridad, ni total ni parcialmente, para superar materias previamente cursadas en esta u otras titulaciones de la Universidad de Málaga o cualquier otra institución de educación superior u otro tipo de fin.

Así mismo, declara no haber trasgredido ninguna norma universitaria con respecto al plagio ni a las leyes establecidas que protegen la propiedad intelectual, así como que las fuentes utilizadas han sido citadas adecuadamente.

En Málaga, a 24. de Agosto de 2020.

Fdo.

Escuela de Ingenierías Industriales, Universidad de Málaga

SISTEMA DE BAJO CONSUMO PARA MONITORIZACIÓN REMOTA DE CAMINADORES

Resumen

Walkit es un kit modular, de código abierto y de bajo coste que permite dotar a cualquier andador convencional de sensores inteligentes conectados a la red. Esto permite monitorizar diversos parámetros del paso del usuario, para extraer información relevante de su estado de salud y su respuesta al tratamiento de rehabilitación a la que está sometido.

Este proyecto pretende desarrollar una plataforma basada en el Internet de las Cosas que conecte el kit a la red y permita la monitorización a distancia de los datos capturados mediante una plataforma que los represente gráficamente, permitiendo la toma de decisiones preventivas basadas en la evolución de las lecturas. Además, habilita la posibilidad de que en un futuro cercano pueda implementarse gracias a ella, un sistema de inteligencia artificial que en función del analisis en tiempo real de dichos datos, prevenga caidas y compense dificultades motoras. Para ello se cambian ciertos componentes físicos y electrónicos del diseño original con el fin de mejorar el rendimiento, el consumo y reducir el espacio ocupado, gestionando posteriormente como se transmiten los datos del kit a la red, su procesamiento y almacenamiento - entre la gran multitud de opciones posibles - para finalmente, poder acceder a ellos de forma elegante y visual en cualquier momento.

Palabras clave

Internet de las Cosas, Plataforma, Monitorización, Asistencia robótica, Bajo coste, Código abierto, Análisis del paso, Rehabilitación, Bajo consumo.

Escuela de Ingenierías Industriales, Universidad de Málaga

LOW POWER ON GAIT MONITORING SYSTEM FOR ROLLATORS

Abstract

Walkit is an open source, low cost, modular kit developed in order to provide intelligence to any conventional rollator. It allows users on gait data analysis with the goal of obtaining valuable information on their health and their rehabilitation progress.

This project aims to develop an Internet of Things system that connects the kit to the network and allows the captured data to be monitored remotely through its display on an online platform. It enhances preventitative measures according to the data series evolution. It also allows for the implementation of an artificial intelligence based system in a near future that acts in response of the captured data so it can help to prevent potential falls and compensate for motor disabilities. To achieve this, some hardware and electronics from the original design has been changed in order to improve power consumption, space and overall performance for an IoT system. Thereafter managing how data is transmitted from the kit to the platform as well as its processing and storage- from all different options available, with the result of finally being able to access the data through an elegant and visual dashboard display at any time.

Keywords

Internet of Things, Platform, Monitoring, Robotic assistance, Low cost, Open source, On Gait Analysis, Healthcare, Low power, Modular.

A mis padres y hermano,
quienes me han apoyado, entendido,
a pesar de mi gran ausencia durante estos 5 años.
¡Os quiero!

Pablo

Índice

Índice de figuras

Índice de Tablas

Parte I

Introducción

Capítulo 1

Introducción y visión general

Contenido

1.1. Motivación

El andador inteligente Walkit tiene como objeto monitorizar el estado de recuperación de un paciente en rehabilitación por medio de la captación de información relativa al apoyo sobre los mangos o de la velocidad de desplazamiento y actividad del paciente a través de una serie de sensores.

Actualmente los test pueden ser realizados en el hospital, bajo la supervisión de personal y en un entorno delimitado y controlado, lo que puede conllevar a una actuación del paciente alejada a su comportamiento cotidiano habitual. Por otro lado, nuevas versiones permiten trasladar el andador al domicilio, eliminando dicho problema, pero impide al personal conocer la evolución experimentada hasta un periodo posterior, cuando el paciente sea citado al centro, lo que entorpece el proceso de monitorización y elimina la posibilidad de tomar acciones preventivas basadas en el estudio diario de su evolución. Además, el diseño actual posee una autonomía más reducida de lo deseable, en torno a 3 horas, requiriendo la recarga del dispositivo varias veces al día.

Bajo este pretexto, la motivación de este proyecto reside en la implantación de una plataforma IoT que permita la monitorización de los datos captados. Estos se actualizanç al menos, una vez al día, permitiendo a los pacientes usar el dispositivo en cualquier localización y quedando los datos siempre disponibles para su análisis en el hospital. Así, los problemas asociados a las versiones anteriores quedan eliminados. Por otro lado se pretende realizar una renovación de algunos de los elementos de hardware de la unidad del andador por otros que dispongan de un menor consumo y por ende, aumenten su autonomía.

1.2. Objetivos

La realización de este proyecto persigue los siguientes objetivos:

- Adquirir una visión global del Internet de las cosas: sus aplicaciones y las diversas técnicas para llevarlas a cabo
- Estudiar un amplio número de componentes que pueden formar parte de una plataforma: sensores, microcontroladores y componentes de software.
- Estudiar las diversas alternativas existentes para alojar una plataforma en la nube y estudiar sus ventajas, desventajas y sus aspectos económicos.
- Proponer diversos modelos de funcionamiento para la transmisión de datos entre el andador y la plataforma y seleccionar el más adecuado.

- Implantar una plataforma de monitorización de datos elegante y funcional, de la que se pueda obtener información útil y en el caso de que se desee, permita la descarga de los datos.

1.3. Metodología

El internet de las cosas y todo aquellos métodos, protocolos y componentes físicos que la hacen posible, componen un campo de conocimiento muy extenso. La metodología seguida para alcanzar los objetivos expuestos consiste en:

- Realizar un extensa revisión bibliográfica, comenzando por los aspectos más generales y avanzando poco a poco a cada elemento particular.
- Analizar la información recogida para poder alcanzar conclusiones justificadas de los elementos seleccionados
- Implementación adaptativa. Una vez tomadas las decisiones, se trabaja en cada sección de la plataforma de forma simultanea (base de datos, plataforma de gestión, y plataforma de monitorización) permitiendo realizar correcciones de errores en una parte al observar incompatibilidades con la otra, evitando tener que deshacer todo el trabajo hecho.

1.4. Directrices seguidas

La extensión del IoT es tal, que hay que aceptar que no existe una solución única para llevar a cabo la plataforma de la mejor forma posible. Por tanto, para su creación se han seguido las siguientes directrices:

- Intentar conservar el mayor número de elementos del diseño previo.
- Todo cambio de diseño que suponga una mejora de la autonomía es aceptado.
- Hacer uso de elementos que permitan en la medida de lo posible los requisitos expuestos en el Apartado 1.5.
- Primará aquella opción que, cumpliendo el apartado anterior, sea más económica.
- La plataforma será usada de forma real, por lo que todas las decisiones en la medida de lo posible deben crear una buena experiencia al usuario final.

1.5. Requisitos de la plataforma

En la medida de lo posible, a lo largo de este proyecto se debe tratar de considerar el cumplimiento de los siguientes aspectos:

- *Escalabilidad:* La plataforma debe poder soportar más de un andador para que las futuras unidades puedan aprovecharse de la infraestructura creada en este proyecto.
- *Autonomía:* Los elementos seleccionados deben contribuir a una reducción del consumo energético.
- *Alta frecuencia:* Para poder realizar un filtrado a posteriori de los datos captados sin perder resolución. Una frecuencia entre 50-100Hz es deseable.
- *Simplicidad y usabilidad:* Los pacientes pueden ser personas mayores con bajo conocimiento técnico, por lo que la unidad de andador y el modelo en el cual los datos son transmitidos no deben suponer una molestia o la causa de malfuncionamiento del andador.
- *Economía:* Los servicios adoptados deben suponer un bajo coste para facilitar la implementación de mayor número de unidades.

1.6. Software

Las herramientas de software usadas en este proyecto se resumen en el siguiente listado:

- **Arduino IDE,** para la programación de los microcontroladores.
- **Mosquitto MQTT,** como broker de los mensajes.
- **MariaDB,** como gestor de la base de datos.
- **Apache2,** como servidor web.
- **PHPMyAdmin,** como interfaz para interaccionar con la base de datos.
- **Node-Red,** para gestionar el flujo de datos a la plataforma.
- **Grafana,** para la implementación de la plataforma de monitorización de datos.

Por otro lado, las herramientas enfocadas a la creación del documento son:

- **LaTeX,** para la confección del documento.
- **PowerPoint,** para la confección imagenes, diagramas y flujos.

1.7. Estructura del documento

El documento se estructura en partes y capítulos para discernir correctamente el objetivo de los mismos.

La primera parte supone, en su *Capítulo 1*, la introducción al proyecto, las metodologías y objetivos.

La segunda parte supone una revisión del estado del arte y la técnica. El *Capítulo 2*, introduce el andador, los trabajos previos relacionados y el punto de partida sobre el que se comienza a trabajar, describiendo los componentes, el montaje y las especificaciones que la plataforma debería cumplir. El *Capítulo 3*, enfoca dicha revisión al Internet de las cosas, sus aplicaciones y modelos de arquitectura, mientras que el *Capítulo 4*, se centra en introducir los protocolos de comunicación entre dispositivos y de transporte de datos más relevantes.

La tercera parte compone el cuerpo del proyecto. El *Capítulo 5*, sirve de puente entre el estado de la técnica y la implementación de la plataforma, plasmando la revisión bibliográfica acerca de los posibles componentes que pueden formar parte de este proyecto en particular, para acto seguido, analizar y seleccionar en el *Capítulo 6*, dichos componentes en base a lo expuesto. El *Capítulo 7* compone la implementación de todo lo expuesto con anterioridad. Comienza resumiendo brevemente las decisiones tomadas y procede a explicar las decisiones de diseño en la implementación, así como a exponer el desarrollo de la ejecución e implementación de la misma. Por último, el *Capítulo 8* compone las pruebas y experimentos realizados a la plataforma.

Finalmente, se exponen las **conclusiones y las líneas de trabajo futuras. El Anexo A** redirige al lector al repositorio donde se encuentran todos los códigos relativos al andador.

Parte II

Estado del arte y de la técnica

Capítulo 2

WalKit. El andador inteligente

Contenido

2.1. El andador

El proposito de equipar el dispositivo de medición en el andador y no en otros equipos reside en [5]:

- Debe ser un elemento de uso cotidiano.
- No debe ser incomodo para el usuario.
- No requiere expertos para su calibración cada vez que se pone en marcha.
- Su apariencia debe ser normal y no llamar la atención, para evitar el posible estigma social que puedan sufrir los usuarios [1].

El sistema está diseñado para poder ser equipado en cualquier andador convencional, aunque actualmente se está empleando un modelo $MEYRA^{®}$, muy común entre los usuarios de andadores, representado en la Figura 2.1.

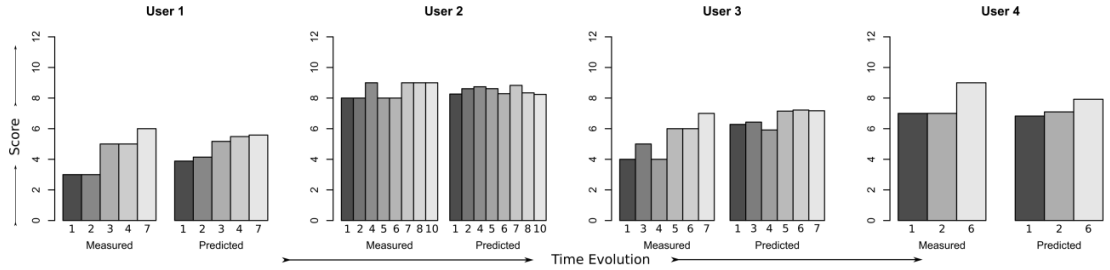
Figura 2.1: Modelo del andador utilizado en el proyecto.

2.2. Trabajos previos

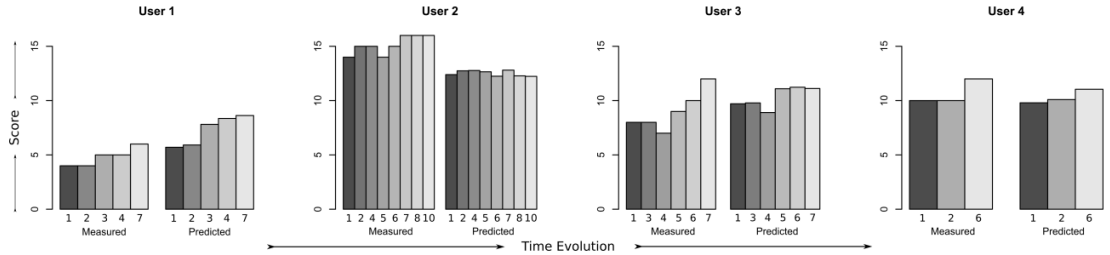
Los métodos de análisis de la forma de caminar de pacientes con movilidad reducida en rehabilitación lleva en desarrollo durante varios años por numerosos grupos

de investigación. En el caso concreto del grupo que lleva el desarrollo del andador que este proyecto concierne, en Noviembre de 2016 se publica un articulo estimando el error que supone medir la asistencia que requiere el paciente para caminar por medio de dispositivos como el Walkit, que no requieren personal especializado para su calibración, ni de elementos portables incomodos para el usuario tales como sensores que pueden estar equipados sobre el cuerpo [5], en lugar de la instrumentación clásica, más precisa pero con más complicaciones.

La escala de movilidad clínica de Tinetti es un ensayo que permite evaluar a partir de ciertos parámetros el grado de movilidad del paciente. Tradicionalmente requiere de la intervención del paciente para rellenar una serie de cuestiones, así como del analisis del terapéuta. En 2017, *se valida un método de evaluación de dicho parámetro empleando la plataforma Walkit,* donde se observa que los resultados predichos se asemejan mucho y siguen la misma tendencia que los reales empleando los métodos tradicionales. Los ensayos fueron realizados en el Hospital Civil de Málaga y la Fondazione Santa Lucia de Roma [1]. La Figura 2.2 ilustra dichos resultados.



(a) Tinetti gait rehabilitation progress



(b) Tinetti balance rehabilitation progress

Figura 2.2: Resultados del Ensayo Tinneti realizado en 2017 [1].

Además del andador Walkit, se han desarrollado otros dispositivos para monitorizar la condición del paciente. En concreto, en 2019 se publica un bastón con sensores de fuerza integrados *(Smart Cane),* con el objetivo de recoger un gran volumen de datos que permitan detectar anomalias que pongan en aviso a posibles asistencias médicas, siendo usado de forma diaria por los pacientes sin que ello sea un problema al ser un objeto de uso cotidiano ni suponga un problema de ergono-

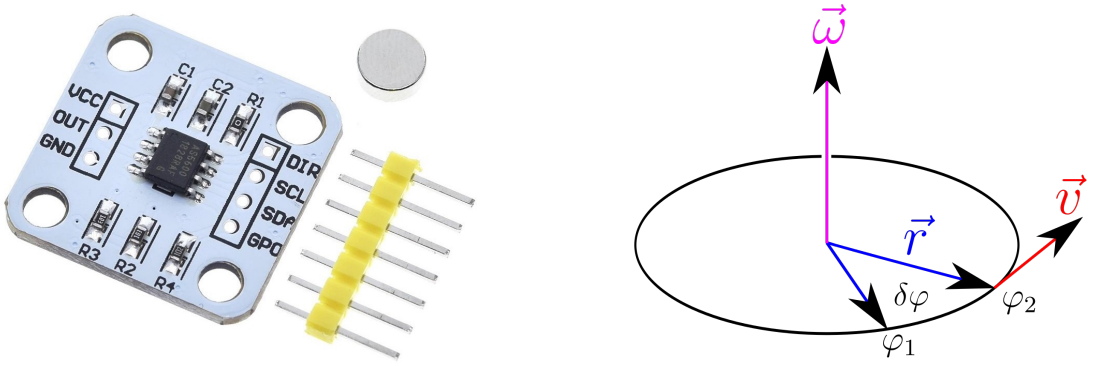
mia y con bajo consumo de batería [6]. Meses más adelante, en [7] se profundiza en la arquitectura de la aplicación IoT que emplea el bastón (Bluetooh low energy y dispositivo móvil como gateway) y en como la autoadaptación de los dispositivos inteligentes les permite gestionar por si solos su modo de funcionamiento para poder optimizar el consumo de batería, aplicado al bastón anteriormente mencionado. Dos de los métodos planteados son el uso de umbrales o redes neuronales, para que el sistema pueda aprender por sí mismo la mejor gestión energética, aunque queda descartado por su alto consumo de memoria.

2.3. Punto de partida

Los sensores y actuadores empleados [8] para las mediciones de los parámetros requeridos son:

- 2 Encoders.
- 2 Amplificadores de célula de carga.
- 2 Servos situados en la maneta de freno.

El **Encoder AS5600 1828RAF C** es un potenciómetro de alta resolución (12 bits) con salida analógica o PWM, para medir ángulos diametrales haciendo uso del efecto Hall [9]. Permite medir de forma indirecta la velocidad a la que se desplaza el andador. Se dispone uno en cada rueda.



(a) AS5600

(b) Relación entre diferencia de ángulo y velocidad.

Figura 2.3: Encoder y fundamentación física para medición indirecta de la odometría.

La placa PCB 1828RAF C - Figura 2.3a - incluye las resistencias de pull up y condensadores externos que requiere el encoder para funcionar. Las mediciones deseadas - Figura 2.3b - se obtienen mediante el cálculo mostrado por la siguiente ecuación:

$$v = \frac{\varphi_2 - \varphi_1}{t_{21}} * r \tag{2.1}$$

El **Amplificador HX711** es un convertidor ADC (*Analog to Digital Converter en inglés*) de 24 bits de precisión [10] que permite leer las lecturas recogidas por las galgas extensiométricas situadas en los mangos del andador, midiendo la fuerza con la que el usuario se apoya sobre ellos.

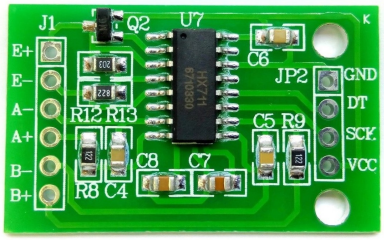


Figura 2.4: Amplificador HX711.

Una galga extensiométrica es un transductor en forma de hilo conductor muy delgado con gran parte de su longitud en una direccion, doblada de forma que constituye una serie de hilos paralelos [11]. La resistencia de un hilo es dependiente de la sección y al experimentar la galga una deformación, esta varía. La expresión fundamental de la extensiometría es por tanto:

$$\frac{\triangle R}{R} = K\frac{\triangle l}{l} = K\varepsilon \tag{2.2}$$

La teoría fotoelástica no es objeto de este trabajo de fin de grado y por tanto no se desarrollan las ecuaciones intermedias que permiten alcanzar la siguiente expresión:

$$\triangle v = \frac{VK}{4} * (\varepsilon_1 - \varepsilon_2 + \varepsilon_3 - \varepsilon_4) \tag{2.3}$$

Donde:

- V es el voltaje de alimentación.
- K es el factor de galga.
- ε son las deformaciones leidas por cada una de las galgas.

- $\triangle$v es la variación de tensión en bornas del circuito debido al incremento de resistencia generado.

El HX711 - Figura 2.4 - se encarga de controlar dicho factor de galga para generar un valor de mayor magnitud para mejorar la precisión de la lectura, ya que el incremento de voltaje generado es muy pequeño.

El mango del andador puede modelarse como un voladizo empotrado en uno de sus extremos con una fuerza aplicada en el extremo contrario. Modelos más precisos pueden considerar la rigidez de la unión del extremo del mango, permitiendo cierto grado de giro en el mismo, y una fuerza superficial en lugar de puntual, pero para ilustrar el fundamento teórico no es necesario. Como las galgas están pegadas en la superficie, el cortante es nulo y se obtiene el estado tensional de forma directa en direcciones principales. En la situación descrita, el estado tensional del voladizo es el mostrado en la Figura 2.5:

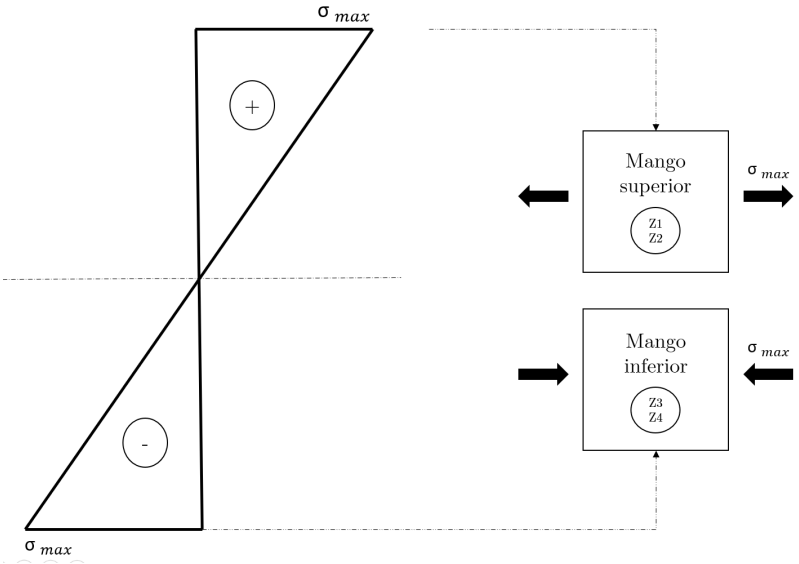


Figura 2.5: Estado de tensión en los puntos de medición de las galgas situadas en los mangos del andador.

Para obtener las lecturas, se hace uso de un puente de Wheatstone, con un montaje de puente completo con cada par de galgas opuestas en una cara diferente, tal y como se muestra en la Figura 2.6.

Esto es así debido a que el estado tensional es simétrico respecto a la fibra neutra pero de signo contrario, es decir, la cara superior está en tracción y la inferior a compresión. En la Ecuación 2.3, las deformaciones correspondientes poseen signos contrarios, por lo que estos terminos acaban sumandose y como hay dos en cada cara,

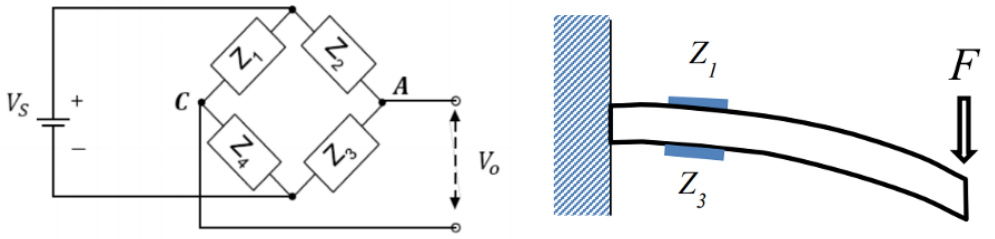


Figura 2.6: (I) Montaje del puente de Wheatstone. (D) Voladizo y montaje de galgas.

la lectura final tiene cuatro veces más precisión. Además, los efectos de incremento de temperatura que podrían afectar a la medida quedan compensados por este mismo motivo.

El **servo SC182029-V1R0** actúa sobre los cables de freno, entre un ángulo de 0-90º y, consecuentemente, modificando la posición de las manetas de freno, con el proposito de poder ser controlado a distancia automáticamente en caso de que el sistema detecte una anomalía, frenando el andador para prevenir posibles caidas, por ejemplo.

(a) Maneta de freno. (b) Servo.

Figura 2.7: Sistema de frenado del andador.

Se hace uso de una señal de servo estandar, con dos cables para la alimentación DC y otro para el control de los pulsos PWM (*Pulse Width Modulation*).

2.3.1. Componentes y esquema del circuito.

Se puede dividir el circuito en dos partes bien diferencias: circuito de alimentación y circuito de datos.

Circuito de alimentación

Las baterias se encuentran emparejadas dos a dos, de forma que se cuentan con dos baterías 2S con un voltaje en conjunto de 14.8V. Los sensores se alimentan a 5V, por lo que se dispone de un regulador de tensión para que en ella la tensión sea la adecuada. La Raspberry Pi es alimentada de forma separada a los sensores, por medio de una power bank común para la carga portatil de telefonos móviles.

Por otro lado, cada par de baterías se encuentra protegido por un *protector de baterías* de forma que el voltaje no pueda caer por debajo de un mínimo peligroso[1] o cargarse en exceso. Cada una de ellas se hayan conectadas tal y como muestra la Figura 2.8.

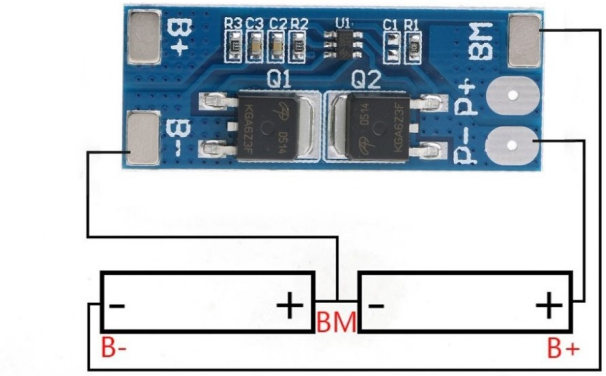


Figura 2.8: Esquema de conexión del protector de baterías HX-2S-D01.

Finalmente, se dispone de un *switch* para encender o apagar el dispositivo interrumpiendo la alimentación, un *puerto hembra* para cargar las baterías sin tener que extraerlas y un *fusible* como elemento de protección ante sobreintensidades, evitando cualquier potencial daño en el circuito.

Circuito de datos

Está compuesto por las conexiones necesarias en los sensores para extraer correctamente los datos del mismo. Cada sensor se conecta directamente a un *Arduino*, para poder realizar la conversión del protocolo de comunicación a CAN Bus (*Controller Area Network bus*), por medio del *conversor CAN Bus* . Dicha conexión se puede observar en la Figura 2.9. El protocolo de comunicación empleado debe salvar las siguientes complicaciones:

[1]Las baterías poseen un rango de voltajes de funcionamiento y estar fuera del mismo causa daños en la batería, reduciendo su vida útil y rendimiento.

- Se requiere un protocolo fiable para no perder información importante en las comunicaciones.
- La distancia entre sensores y Raspberry Pi es elevada, y la frecuencia de muestreo alta, por lo que se requiere un protocolo rápido.

El protocolo I2C (*Inter Integrated Circuits*) alcanza velocidades de 100 kbps, lejos del 1Mbps que alcanza el CAN bus y debido a su funcionamiento síncrono, la gran longitud de los cables no es adecuada. SPI (*Serial Peripheral Interface*) por su parte requiere 4 cables por sensor, añadiendo un montaje más complejo. No obstante su velocidad es muy superior a los otros dos, en torno a los 10 Mbps.

El protocolo CAN cumple con ambos requisitos, al ser rápido y fiable. Además permite recoger los datos de todos los sensores en solo 2 cables, lo que permite compactar el diseño y su consumo de batería es reducido [12].

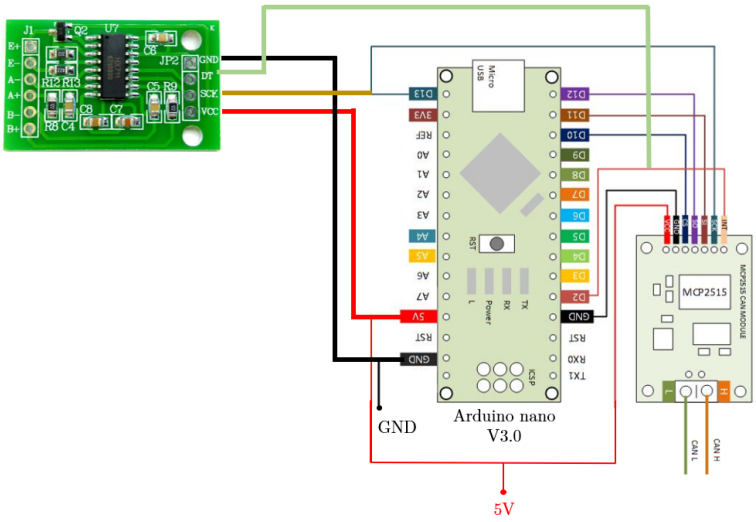


Figura 2.9: Esquema de sensor a Arduino para conversión CAN Bus.

Para comunicar los sensores con el núcleo donde se encuentra el resto de electrónica, se emplean *conectores RJ45.* Una vez estos acceden a la caja de conexiones, se dispone una placa a modo de *nodo eléctrico* en el que la alimentación, tensión y tierra, de cada uno de los sensores, son soldados en un solo cable, es decir, quedando todos los sensores en paralelo. Por otro lado, en esta, los cables de datos provenientes de cada sensor se sueldan para reducir el volumen de cables que pasan posteriormente al interior de la caja. Estos dos cables se conectan a la *Raspberry Pi 3,* por medio del *conversor CAN* para reconvertir la señal a SPI.

El circuito resultante puede apreciarse en la Figura 2.10.

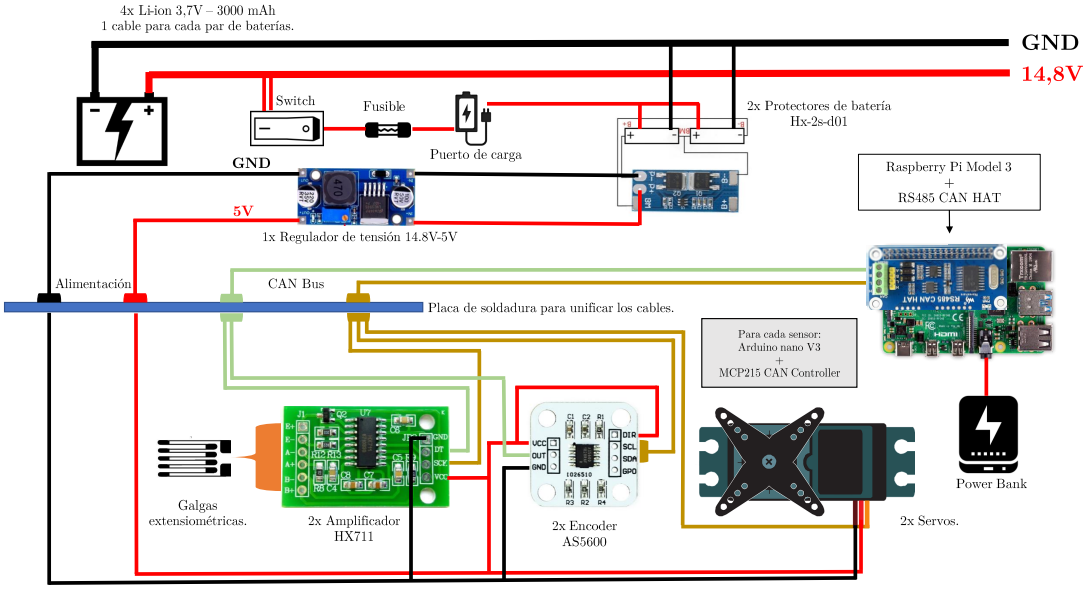


Figura 2.10: Esquema del circuito original.

Los elementos mencionados en el texto anterior se recogen en la Tabla 2.1, especificando el modelo y la cantidad a disponer en el circuito.

CIRCUITO	*COMPONENTE*	*MODELO*	*CANTIDAD*
Alimentación	Baterías	Li-Ion 3.7V - 3000 mAh	4
	Regulador de tensión		1
	Power bank		1
	Protector de baterías	HX-2S-D01	2
	Botón On/Off	-	1
	Fusible		1
Datos	Sensores/Actuadores	Descritos en Apartado 2.3	6
	Microcontrolador	Arduino Nano V3	6
	Conversor CAN bus	MCP2515 con interfaz SPI	6
	Cables	RJ45	6
	Placa de soldadura	-	1
	Adaptador	RS485 CAN HAT	1
	Ordenador	Raspberry Pi	1

Tabla 2.1: Elementos que componen el circuito original previo a la realización del proyecto.

Capítulo 3

Internet de las cosas

Contenido

3.1. Introducción

El término Internet de las cosas, más conocido por su acrónimo inglés, IoT (proveniente de "Internet of Things") ha sido definido de numerosas formas por varios autores. Vermesan [13] lo define como "una simple interacción entre el mundo físico y el digital, por medio de una variedad de sensores y actuadores". Por otro lado, Peña-Lopez en [14] lo define como "un paradigma en el cual las capacidades de computación y de red están incrustadas en cualquier objeto concebible. Usamos estas capacidades para consultar el estado de los mismos y si es posible, cambiarlo".

En general, el IoT conforma un paradigma complejo que no puede definirse como una tecnología única, sino como un conjunto de tecnologías que aúnan sensores, actuadores, procesos de comunicación, computación... Esto supone una revolución en el papel que juegan los dispositivos que nos rodean en la vida cotidiana y en la industria. La comunicación e interacción entre diversos sensores y dispositivos, la recogida de datos y el análisis de los mismos a tiempo real o a posteriori permite coordinar decisiones sin necesidad de conexión física entre los mismos ni de intervención humana.

3.1.1. Breve reseña histórica

Aunque pueda parecer que el Internet de las Cosas es un concepto nuevo, este da comienzo a principios del siglo XIX[1], con los primeros experimentos de telemetría realizados con pequeñas estaciones meteorológicas situadas en el Mont-Blanc, en 1874. El propio Nikola Tesla mostraba cuan adelantado a su tiempo estaba comentando una realidad que hoy día es posible, para una entrevista para la revista Colliers, en 1926: *"Cuando lo inalámbrico esté perfectamente desarrollado, el planeta entero se convertirá en un gran cerebro, que de hecho ya lo es, con todas las cosas siendo partículas de un todo real y rítmico... y los instrumentos que usaremos para ellos serán increíblemente sencillos comparados con nuestros teléfonos actuales. Un hombre podrá llevar uno en su bolsillo"*. Alan Turing, en 1950, también discutía acerca de la inteligencia y capacidad de comunicación de los sensores: *"...también se puede sostener que es mejor proporcionar la máquina con los mejores órganos sensores que el dinero pueda comprar, y después enseñarla a entender y hablar inglés. Este proceso seguirá el proceso normal de aprendizaje de un niño"*.

[1] Ante la ausencia de articulos públicos referente a la evolución histórica, para realizarla se ha extraido información de:

- http://www.bcendon.com/el-origen-del-iot

(Acceso Online el 22 de Agosto de 2020)

No obstante, debido a la inmadurez tecnológica de la época, no fue hasta la década de los 60-70, cuando se empezaron a desarrollar los primeros protocolos de comunicación , de uso exclusivamente militar. Posteriormente con la aparición del protocolo TCP/IP, lo que en su día fue una red de comunicaciones militar, *arpanet*, derivó en lo que hoy día es *Internet*. El primer objeto conectado a internet mostrado al público surgió en 1990, cuando John Romkey controló una tostadora por medio de un protocolo denominado SNMP (*Simple Network Management Protocol*). Pero por aquel entonces, las conexiones eran cableadas. La verdadera revolución nació con el surgimiento de la conectividad inalámbrica (celular y WiFi), acelerando el desarrollo de las comunicaciones entre dispositivos.

El término IoT como tal se acuñó finalmente en 2009, de la mano del profesor del MIT, Kevin Ashton, en el RFID Journal (*Radio Frequency Identification*), a pesar de que la investigación sobre el mismo llevara ya bastantes años en marcha, contando desde principios de siglo con infinidad de textos científicos sobre aplicaciones del mismo, y a partir de ahí el desarrollo del mismo ha sido exponencial, con la estandarización de nuevos conceptos como el M2M (*Machine To Machine*) o el WSN (*Wireless Sensor Networks*) y de protocolos como el MQTT (*Message Queue Telemetry Transport*), hasta alcanzar el día de hoy, donde deja de ser un concepto a ser prácticamente una realidad, aunque aún con mucho potencial por otorgar.

3.2. Aplicaciones

El inmenso potencial del IoT para mejorar la calidad de vida o la eficiencia de los procesos industriales es inmenso. De esta forma, se abre una vía de posibles aplicaciones prácticamente ilimitada, en cualquier tipo de sector, para dispositivos individuales o para un conjunto de ellos, con inteligencias de mayor o menor complejidad. Cualquier campo imaginable puede ser precedido por el término Smart y dar lugar a un nuevo mundo de posibilidades. Algunas de las más populares son [2]:

- **Smart home:** Permiten al usuario controlar los diversos elementos de la vivienda, como luces, persianas, aire acondicionado... con la voz (Amazon Alexa) o que estas se regulen por si solas dependiendo de diversos ambientales o la presencia humana. Otras aplicaciones más complejas, como por ejemplo, el proyecto MavHome [15], analiza las rutinas de los usuarios, almacenando y analizando estos datos para otorgar cierto grado de automatización de las mismas de forma adaptativa.

  La Figura 3.1 muestra las numerosas aplicaciones disponibles.

- **Smart Cities:** Entre otras aplicaciones, permiten la detección de parkings

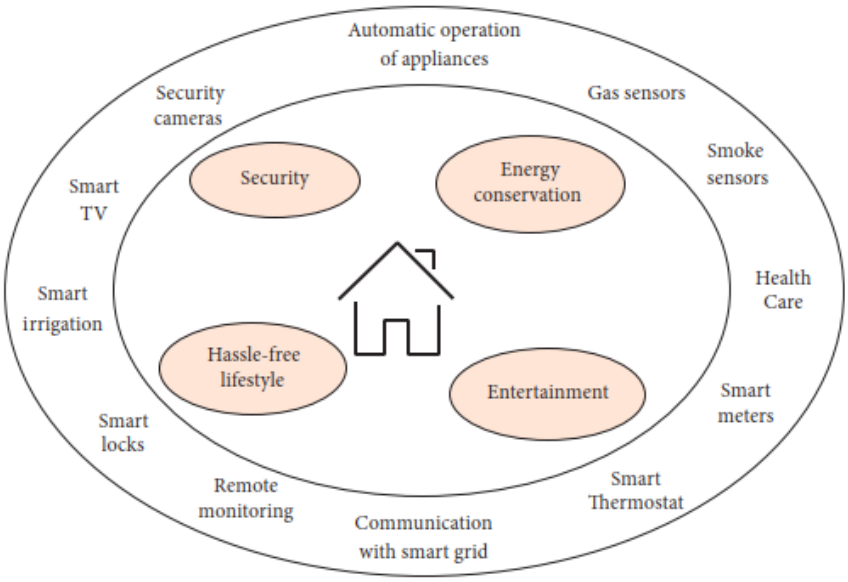


Figura 3.1: Potenciales aplicaciones del servicio Smart Home [2].

libres cercanos (Figura 3.2), predicción de duración de trayectos según el estado actual del tráfico o una gestión automática del mismo. Para ello, hace uso de sensores que monitorizan la densidad de vehículos en un área para así determinar cuál es el tiempo de espera óptimo en los semáforos y minimizar la congestión.

- **Smart Health and Fitness:** Permite parametrizar el estado de salud de pacientes en tiempo real y mandar avisos si algún parámetro sale fuera de lo normal, poseer una base de datos más completa del estado y en ciertos casos, preescribir medidas para paliar ciertos problemas. Por otro lado, los relojes deportivos recogen la actividad del usuario, el número de pasos, la energía gastada o el ritmo cardiaco y son capaces de medir el estado de forma del usuario y recoger un historial de sus actividades en la nube.

- **Smart Grids:** Permiten mejorar el proceso de generación, transmisión y distribución de la energía, añadiendo inteligencia en cada una de de las etapas, de forma que la energía puede fluir en dos sentidos (del suministrador al consumidor y viceversa), haciendo uso de micro redes [16].

En la industria también existen infinidad de aplicaciones, como por ejemplo, gestión de flotas de vehículos, mantenimiento de maquinaria, gestión logística, agricultura, ganadería, etc. Por ejemplo, se puede contar con una localización en tiempo

real de suministros en la cadena logística por medio de sensores RFID y NFC (*Near Field Communication*), ayudando a analizar además la eficiencia del proceso e incluso a predecir la futura demanda [17]

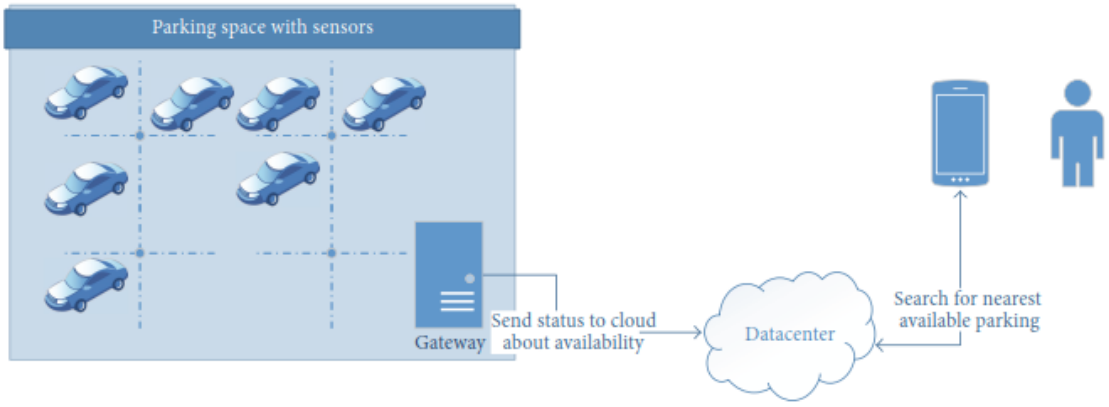


Figura 3.2: Sistema de parking inteligente en ciudades [2].

Sin duda, uno de los ejemplos más claros que indican la actualidad y relevancia del IoT tiene origen en el *Covid-19 [18]*, donde multitud de aplicaciones de *turismo inteligente* están en pleno desarrollo. Así, se permite monitorizar la ocupación de las playas, el flujo de personas por las calles o el estado de salud de las personas que han accedido a un establecimiento en una franja horaria para poder facilitar el seguimiento y confinamiento de nuevos casos tratando de dañar lo menor posible al sector turistico. Así mismo, el IoT abre la puerta a nuevos modelos de mercado gracias al uso de técnicas de almacenamiento y análisis basados en Big Data.

No obstante, el gran potencial de poseer infinidad de datos y dispositivos en la nube también se convierte en uno de los grandes desafíos del sistema, ya que toda esta información debe de estar altamente securizada para garantizar la privacidad y evitar que usuarios maliciosos puedan controlar los dispositivos. Así mismo, en aplicaciones donde el volumen de datos es demasiado voluminoso, el almacenamiento, análisis e interpretación de los mismos requiere de técnicas no convencionales, las cuales se aúnan en el término Big Data. El potencial de recogida de datos a tiempo real del IoT unido la eficiente gestión de los datos del big data, abre la puerta a nuevos modelos de mercado y toma de decisiones en grandes empresas, convirtiéndose ambos en dos de los grandes pilares de la industria 4.0 [3].

3.3. Capacidades de un sistema IoT

La Figura 3.3 de [3] describe las 6 capacidades fundamentales que todo sistema IoT debe tener:

Figura 3.3: Capacidades de un sistema IoT [3].

1. Identificación.

Es una capacidad crucial nombrar cada elemento con una ID única dentro de la red, por ejemplo, T1, temperatura del sensor uno. Algunos métodos empleados son el uCode o el EPC (Electronic Product Code). Por otro lado, es vital emparejar univocamente los servicios con su demanda, de lo contrario, el sistema no funcionaría. Para ello se debe considerar una dirección para cada uno de los objetos dentro de la red, usando protocolos como el IPv6, IPv4 o 6LoWPAN para redes de baja potencia.

2. Captación.

El sistema debe ser capaz de recopilar información de los objetos que componen la red, por medio de sensores o actuadores, conectados normalmente a placas de bajo presupuesto como Arduino, Raspberry Pi o ESP32, para permitir así su conexión a la red.

3. Comunicación.

Permite la conexión inalámbrica de los diversos nodos que conforman el sistema, uniendo los dispositivos con los diversos servicios, para proveer una aplicación inteligente. Existen numerosas tecnologías, las cuales serán descritas en el Capítulo 4. Las más conocidas son: Conexión celular(3GPP, 4G o la nueva 5G), WiFi, BLE (bluetooh low energy) o LoRa/LoRaWAN.

4. Computación.

Tras la transmisión de datos, y con una red interconectada, la computación hace referencia al almacenamiento y procesamiento de datos. Puede realizarse tanto de forma privada, como por medio de plataformas en la nube (Cloud computing).

5. Servicios.

Los datos almacenados se gestionan para poder llevar a cabo una aplicación determinada, es decir, proveer de un servicio al usuario. Existen varios tipos:

- *Servicios de identidad vinculada:* Es el más básico e importante. Identifica los dispositivos.
- *Servicios de identidad agregada:* Recoge y asocia los datos de los sensores que serán procesados posteriormente en la aplicación.
- *Servicios de cooperación:* Hace uso de los servicios anteriores para tomar decisiones por medio de dicha información.
- *Servicios ubicuos:* Pueden categorizarse como servicios de cooperación disponibles en cualquier momento. Los servicios aspiran a alcanzar este estado, aunque es un fin complicado.

6. Semántica.

El término se refiere a la habilidad de extraer conocimiento de diferentes máquinas de forma inteligente para proveer el servicio requerido. En definitiva, puede entenderse como el cerebro de la aplicación, que es capaz de reconocer y analizar datos, intepretarlos y finamente mandar recursos al sitio adecuado.

3.4. Arquitecturas de un sistema IoT

En la actualidad, no existe consenso acerca de como definir la arquitectura de un sistema IoT. Ning and Wang, en [19] se inspira en el ser humano, definiendo tres capas: La primera capa, el cerebro, que procesa y maneja los datos y es capaz de otorgar una respuesta. La segunda, la columna vertebral, equivalente a la red, y finalmente, los nervios, equivalente a los sensores y actuadores. No obstante, en [2] se definen tres tipos de arquitecturas más conocidas entre los investigadores:

3.4.1. Arquitectura de 3 capas

Es la arquitectura más básica de todas, y define la idea principal que persigue el internet de las cosas, no obstante, este modelo de arquitectura no es adecuada para definir por completo todos los aspectos que componen el sistema. Puede emplearse para aplicaciones sencillas. Las capas de las que consta se muestran en la Figura 3.4 y son:

- *Capa de percepción:* Es la capa física donde los sensores recogen la información requerida para cada tipo de aplicación, tales como los parámetros físicos o la identidad de otros objetos inteligentes del entorno. Así mismo, también está formado por los actuadores que interaccionan con el mundo físico ante la señal de la aplicación.
- *Capa de red:* Consiste en la conexión entre los objetos inteligentes o de estos con los servidores y los dispositivos red. También se encarga de transmitir y procesar los datos recogidos.
- *Capa de aplicación:* Capa de aplicación: Es la responsable de proporcionar los servicios de acuerdo a cada aplicación para los usuarios: Smart homes, Smart cities, smart water...

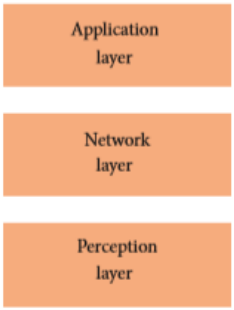


Figura 3.4: Arquitectura de 3 capas [2].

3.4.2. Arquitectura de 4 capas

Como se puede observar, la capa de red de la arquitectura de 3 capas engloba demasiadas tareas heterogéneas que requieren de una distinción para que la arquitectura sea efectiva. La nueva capa que se añade, la de apoyo a servicios, sirve de intermediario entre la aplicación y la red para gestionar los datos recogidos y transmitidos.

3.4.3. Arquitectura de 5 capas

Es la arquitectura más aplicada a las aplicaciones en la actualidad, ya que satisface un modelo bastante completo de las tecnologías empleadas de forma sencilla. El significado de cada capa queda ligeramente modificado, quedando la nueva estructura de la siguiente forma:

- *Capa de percepción:* Equivalente a la de 3 capas.
- *Capa de red:*: Se encarga de transferir los datos recogidos desde la capa de perecepción a la de procesado y viceversa, a través de la red de comunicación elegida (WiFi, 3G, NFC, Bluetooth, por ejemplo).
- *Capa de procesado:* También conocida en la bibliografía inglesa como middleware layer, almacena y analiza los datos transportados en la red. Permite la gestión de los mismos para proporcionar diferentes servicios a capas inferiores. Puede estar compuesta por bases de datos, sistemas de computación en la nube o modulos de procesamiento específicos para big data.
- *Capa de aplicación:* Es la capa a cargo de proveer los servicios a los usuarios, al igual que en el modelo de tres capas.
- *Capa de negocio:* Gestiona la totalidad del sistema teniendo en consideración modelos de negocio, ganancias y asuntos de privacidad. Esto permite la posibilidad de llevar a cabo optimizaciones y mejoras del sistema. Debido a esto, esta arquitectura es la más empleada por los fabricantes.

En la Figura 3.5 se muestran las capas y se resumen las diversas tecnologías que pueden formar parte de cada una.

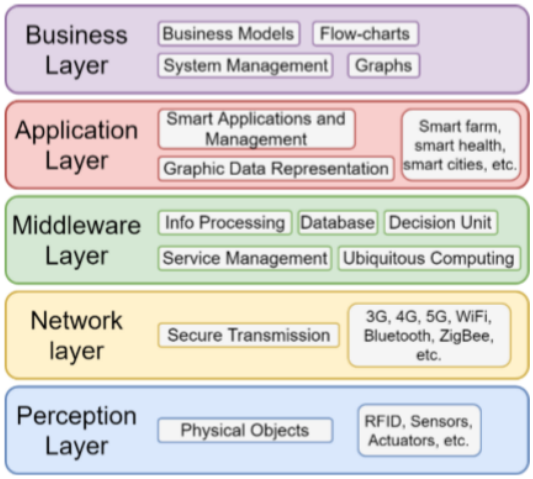


Figura 3.5: Arquitectura de 5 capas.

3.5. Cloud Computing

Las arquitecturas comentadas anteriormente están basadas en protocolos mientras que la computación en la nube basa su arquitectura en los propios sistemas. Realmente la línea que separa a ambas es difusa ya que la computación en la nube sigue manteniendo en esencia aspectos fundamentales de las arquitecturas de 4 y 5 capas, pues en definitiva, se requiere de captación, comunicación, procesamiento, etc. No obstante, merecen una distinción al brindar servicios tales como infraestructuras, plataformas, herramientas de software, inteligencia artificial o herramientas de visualización de datos, provistos de terceras partes a través de la nube, permitiendo a los desarrolladores de aplicaciones IoT un gran potencial de posibilidades a bajo coste tanto computacional (al ejecutarse los servicios por terceros) como económico (relativo al potencial que brinda). La computación en la nube se hace esencial para manejar los recursos en análisis big data a tiempo real de forma eficiente.

En [3] se exponen algunos de los retos que presenta:

- Sincronización por parte de diferentes proveedores en la nube para asegurar servicios en tiempo real.
- Estandarización de servicios.
- Equilibrio entre los requerimientos del sistema IoT y la nube.
- Seguridad.
- Gestión de los recursos y componentes, debido a su disparidad.

No obstante, el uso de arquitecturas en la nube posee numerosas ventajas:[2]

- Autoservicio bajo demanda: Un consumidor puede proveerse unilateralmente de un tiempo de servidor y almacenamiento de red sin requerir interacción humana.
- Acceso ubicuo.
- Distribución de recursos dinámico.
- Elasticidad: Sus funcionalidades pueden ser proporcionadas de forma rápida y flexible en cualquier lugar y momento.

[2] https://it.toolbox.com/tech-101/what-is-cloud-computing

(Acceso online el 22 de Agosto de 2020).

3.5.1. Arquitecturas en la nube

La arquitectura de la computación en la nube puede dividirse en dos secciones conectadas a través de la red, tal y como se muestra en la Figura 3.6:

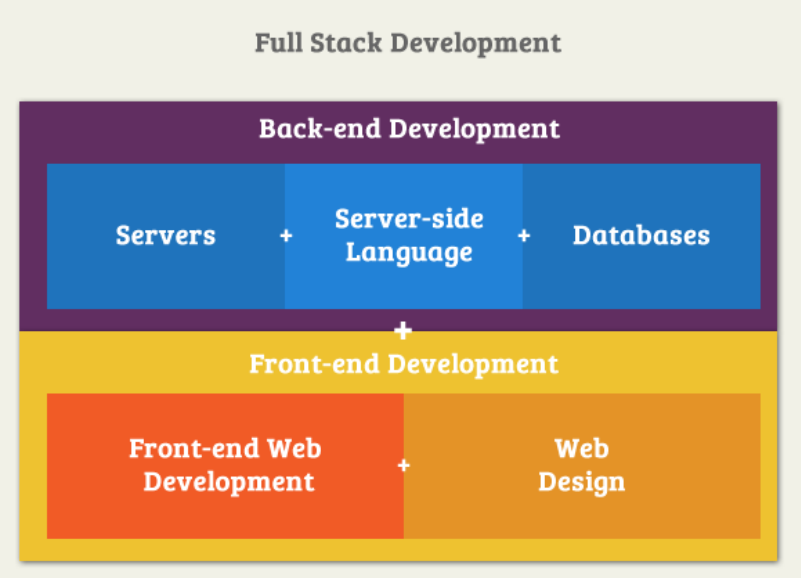


Figura 3.6: Componentes del front-end y back-end.

Front-end: Orientada al usuario final. Engloba todo aquello con lo que el usuario interacciona. Incluye:

- *Software e interfaz de usuario:* El sofware generalmente viene provisto por parte del usuario cliente, con el propio navegador. El sofware generalmente viene provisto por parte del usuario cliente, con el propio navegador web, mientras que la interfaz es la parte visible para el usuario, donde interactúa con la aplicación (por ejemplo, Gmail)
- *Dispositivo cliente:* No requiere potencia computacional por parte del usuario ya que el procesamiento se realiza en la nube.

Back-end: Sección de la arquitectura que impulsa el servicio que se muestra en el front-end. Es responsabilidad del proveedor del servicio en la nube. Formada por:

- *La aplicación que se pone a disposición del usuario final.* Aquí se coordinan las necesidades del cliente con los recursos que se disponen para proporcionar la respuesta adecuada.

- *Servicios.* Es uno de los componentes principales, pues es donde se desempeñan las tareas de computación en la nube. Algunos ejemplos de servicios pueden ser: almacenamiento, servicios web o entornos de desarrollo de aplicaciones.
- *Almacenamiento* de los datos necesarios para la ejecución del software en la nube.
- *Servicios de administración de los recursos en la nube* para conseguir un funcionamiento del sistema fluido y eficiente.
- *Seguridad:* Para proteger al servidor de pérdidas de datos o ataques. También se realizan copias de seguridad del almacenamiento para garantizar el correcto funcionamiento del sistema en caso de error interno.

El NIST (*National Institute of Standards and Technology*) clasifica [20] los modelos de computación en la nube en dos grandes categorías:

- *Modelo de implementación:* Localización y administración de la infraestructura.
- *Modelo de servicio:* Servicios específicos que ofrecen las plataformas de computación en la nube

3.5.2. Modelos de implementación

Los modelos de implementación de la computación en la nube pueden agruparse principalmente en cuatro grupos [20]-[21].

- *Nube pública:* Es una infraestructura abierta para cualquier usuario. Las hay gratuitas y de pago, ofreciendo estas últimas mejores características en capacidad, seguridad, etc. Enfocada para pequeños proyectos personales o académicos, donde la seguridad de los datos no es extremadamente relevante, aunque existen opciones como AWS (*Amazon Web Service*) o Microsoft Azure enfocadas a organizaciones.
- *Nube privada:* Se proporciona para el uso exclusivo de una organización y por lo tanto los servicios no se ofrecen al público general, solo a partes autorizadas. Puede pertenecer a dicha organización y ser administrada por ella, por un tercero o una combinación de ambos. Admite mayor grado de customización, seguridad y control, ideal para empresas, no obstante, aumenta los costes en mantenimiento de la infraestructura y en la educación del personal.

- *Nube híbrida:* Combinación de dos o más nubes individuales de cualquier tipo de las anteriormente mencionadas. Permite la portilidad de los datos entre ambas al poseer los mismos estandares y tecnologías, permitiendo un hibrido entre almacenamiento de información importante de forma segura con información menos relevante
- *Nube comunitaria:* La infraestructura es compartida por múltiples organizaciones para perseguir un objetivo común, manteniendo criterios comunes en materia de metas, políticas de privacidad, seguridad, etc, permitiendo así reducir los costes. Al igual que la nube privada, puede ser gestionada internamente o por un tercero.

3.5.3. Modelos de servicio

Una vez revisadas las diversas implementaciones de la nube, es necesario destacar las diferentes posibilidades que los proveedores ofrecen a los usuarios [20] [21], como se muestra en la Figura 3.7:

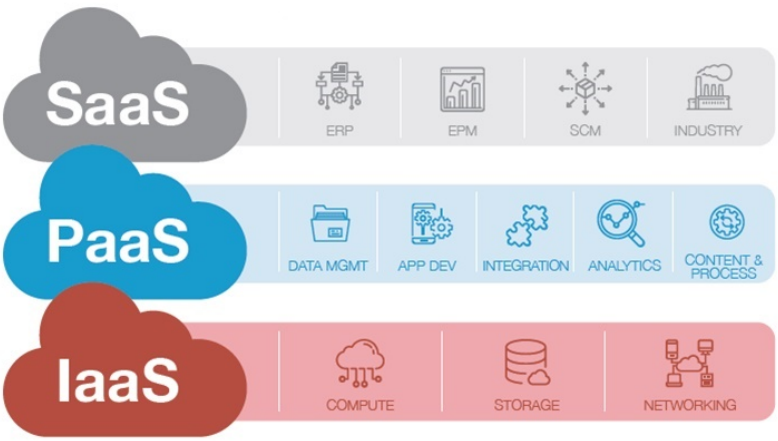


Figura 3.7: Modelos de servicios en la nube y características más relevantes.

- *SaaS (Software as a Service):* El usuario recibe la capacidad de hacer uso de los recursos del proveedor, siendo estos aplicaciones o softwares que son ejecutados en una infraestructura en la nube. El usuario no dispone de ningún tipo de control sobre la misma a excepción de ciertos parámetros de configuración y personalización. Algunos ejemplos son: Gmail, Dropbox, Google Docs.
- *PaaS (Platform as a Service):* El usuario hace uso de la plataforma del proveedor para poder desplegar sus propias aplicaciones. Generalmente disponen

de herramientas de programación para permitir el desarrollo de las mismas. No obstante, no permiten la gestión de la infraestructura de la plataforma. Algunos ejemplos son Google App Engine o Cloud Foundry.

- *IaaS (Infraestructures as a Service):* El proveedor pone a disposición recursos de comunicación, procesamiento y almacenamiento, permitiendo al usuario total libertad para ejecutar por medio de ellos, cualquier tipo de software, desde sistemas operativos hasta aplicaciones.

3.6. Fog computing y Edge computing

Existen ciertas aplicaciones donde los requerimientos de latencia o de eficiencia hacen que la comunicación de datos a un nodo servidor centralizado no sea lo más óptimo. Para poder optimizar esta situación, nace un tipo de arquitectura en la nube similar a la anteriormente explicada, pero con la diferencia de que los recursos quedan descentralizados[3], es decir, permite realizar parte del procesamiento y análisis de los datos en una serie de puertas de acceso a la red (gateways) situados entre la nube y los dispositivos. De esta forma, no solo la latencia disminuye al reducir la distancia de trasmisión de los datos hasta ser procesados, si no que se reduce la carga de datos que debe soportar la nube, por lo que se puede reducir el ancho de banda de las comunicaciones. La Figura 3.8 esquematiza esta estructura.

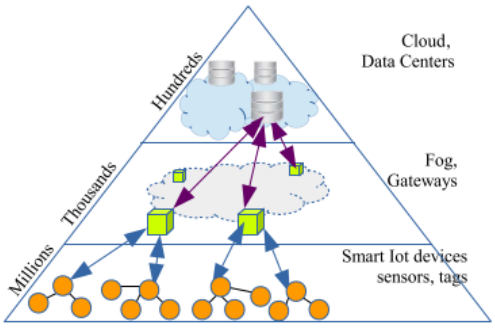


Figura 3.8: Jerarquía seguida en el Fog Computing.

Este tipo de arquitectura es empleada cuando la tasa de generación de datos es mayor que la capacidad de transporte de la red, como puede ser el caso de ciudades

[3] https://www.arsys.es/blog/fogcomputing/

(Acceso Online el 22 de Agosto de 2020)

donde la congestión de la red sea alta o en lugares donde las tecnologías de comunicación existentes no sean suficientemente rápidas. Al acercar la inteligencia a los dispositivos, se puede filtrar que datos se transmiten a la nube por su relevancia, y cuales no, reduciendo el tráfico de datos. Evidentemente, el tamaño del sistema de computación en la niebla es significativamente más reducido que el de computación en la nube pero a cambio, además de las ventajas ya expuestas, es interesante por [3]:

- Distribución de puertas de acceso más barata que un servidor centralizado.
- Sistema escalable con el número de dispositivos de manera sencilla.
- Conforma una nube móvil debido a que la localización de las puertas de acceso también es distribuida.

No debe confundirse con una arquitectura similar, denominada Edge computing, cuya diferencia fundamental reside en que el procesamiento de los datos se realiza en los propios dispositivos, y no en las puertas de acceso, acentuando todas las ventajas ya expuestas. Es común encontrar una combinación de ambos, como se muestra en la Figura 3.9.

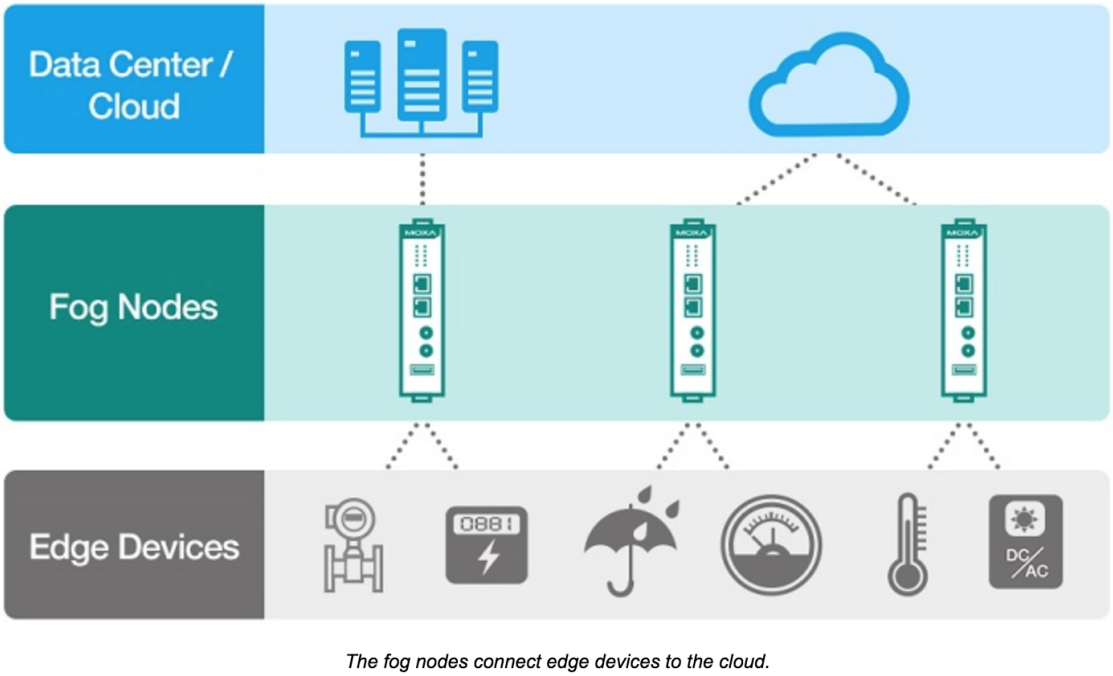


Figura 3.9: Estructura de arquitecturas Fog Computing y Edge Computing.

Capítulo 4

Comunicaciones y protocolos

Contenido

4.1. Protocolos de transporte

Los protocolos de transporte más importantes se describen a continuación [22].

4.1.1. Ethernet

La conexión vía cable ethernet es uno de los protocolos de comunicación más antiguos que existen, estando totalmente estandarizado a nivel global y abierto para cualquier usuario. Existen puertos de conexión ethernet en prácticamente cualquier localización y proporciona una alta velocidad de comunicación, pudiendo alcanzar desde 10Mb/s hasta incluso 1GB/s. Los conectores empleados generalmente son del tipo RJ45, como se muestra en la Figura 4.1.

Figura 4.1: Conectores RJ45.

Otras de sus características son:

- No sufre interferencias con otros protocolos, lo que lo hace muy robusto.
- Es plug & play, por lo que no requiere emparejamiento ni configuración SSID (*Service Set Identifier*).
- Muy barato.

No obstante, posee una serie de desventajas:

- Requiere conexión por cable permanente y el rango queda limitado por la longitud del mismo.
- Consumo alto de corriente, entorno a 200-300 mA constante.

4.1.2. WiFi

Esta tecnología de comunicación es una de las más estandarizadas a día de hoy para acceder a internet, pues es usada prácticamente por cualquier usuario de a pie en el día a día, sea para uso personal o corporativo. Debido a ello, es uno de los protocolos a tener en cuenta para desarrollar cualquier aplicación de IoT. Su principal característica frente al anterior, es la ausencia de cables para realizar la comunicación, no obstante, dependiendo del espectro de frecuencia en el cual trabaje la señal, se obtienen diferentes características. Los protocolos más conocidos y sus características (Tabla 4.1) son:

Protocolo	**Frecuencia**	**Ratio [Mb/s]**	**Alcance aprox. (interior) [m]**	**Alcance aprox (exterior) [m]**
802.11b	2.4 GHz	1, 2, 5.5 o 11	35	140
802.11g	2.4 GHz	6, 9, 12, 18, 24, 36, 48, 54	38	140
802.11n	2.4 GHz	Hasta 288.8	70	250
802.11n	5 GHz	Hasta 600	70	240

Tabla 4.1: Protocolos WiFi y sus características.

Además, dichos espectros pueden ser usados por cualquier otro dispositivo, por lo que dependiendo de la densidad de dispositivos usando dicho canal de frecuencia, es muy posible que los rangos o la calidad de la señal varíen, pudiendo haber interferencias o perdida de datos.

Ventajas:

- No requiere cables.
- Rango de transmisión relativamente bueno.
- Universal y barato de implementar en microcontroladores, incluso habiendo chips con wifi ya incluido.
- Buena seguridad/encriptación de datos ya incluida.

Desventajas:

- Requiere autentificación cada vez que el dispositivo quiera conectarse.
- Puede consumir entorno a 120mA, lo cual puede ser un problema en ciertos proyectos y requiere de desarrollo de "Deep sleep" en los controladores para salvar energía.
- Requiere de certificación (CE en Europa).

4.1.3. Bluetooth y BLE

Bluetooth es un protocolo de comunicación más joven que los dos anteriores, pero su uso es muy extendido en pequeños periféricos como auriculares, teclados, ratones o reproductores de música. A continuación, se describen sus características:

- Rango pequeño, de entorno 10-20 metros (aunque técnicamente alcanza 100 metros).
- Opera a la misma frecuencia que el WiFi, a 2.4GHz.
- Transmisión de entre 1-2Mb/s.
- Consumo de energía muy bajo, demandando corrientes de entre 20-30 mA.
- Compatible con móvil, tableta, portátil o PC moderno.
- No viene encriptado por defecto como el WiFi, por lo que requiere añadir seguridad de forma manual.
- No conectado a internet de forma directa, por lo que requiere de una puerta de acceso.

En consecuencia a lo expuesto, este protocolo puede ser usado para enviar pequeños paquetes de datos a corta distancia en aplicaciones donde se requiere una durabilidad elevada de la batería del dispositivo, dado su bajo consumo. El emparejamiento de dispositivos bluetooth con smartphones para poder acceder a internet es muy extendido, por ejemplo, en relojes inteligentes para subir a la nube la información recogida durante una actividad física.

Las siglas **BLE** hacen referencia a **"Bluetooth Low Energy"**, siendo la cuarta generación de esta tecnología. Su rango de transmisión permanece intacto, pero la cantidad de datos a transmitir es más reducida, a cambio de consumos de energía mucho más bajos, pudiendo funcionar durante semanas, meses o incluso años con la misma batería, dependiendo de la aplicación y de la frecuencia de transmisión de datos.

4.1.4. Celular

Las tecnologías anteriormente mencionadas requieren de un router o un puerto cercano, requieran o no de cables. La transmisión de datos por satélite hace uso de de las torres satélite locales para mandar la información, por lo que proporcionan

un rango extremadamente superior a los anteriores, del orden de kilómetros, sin importar donde se encuentre el dispositivo, siempre que tenga conexión a esta red.

Exiten numerosas generaciones de celular. Los primeros modulos eran llamados 2G o GPRS (*General Packet Radio Service*). Estos permiten transmitir alrededor de 200kb/s de datos y ya no está disponible en muchos países. Actualmente el paradigma del celular consta de modulos 3G y de los denominados LTE (*Long Term Evolution*), en los que se incluyen las generaciones 4G y 5G.

Las ventajas de usar celular son:

- Disponible prácticamente en cualquier localización.
- No hay problema de alcance.
- Posee seguridad incluida. La información está bien encriptada.
- Es fiable.

No obstante:

- Los módulos puede ser caros y se debe pagar mensualmente por cada megabyte usado.
- Consumo de potencia alto.

4.1.5. Low Power Wide Area Network

Hasta ahora se han cubierto protocolos de comunicación de corta distancia. Las soluciones basadas en celular [4] proveen gran rango, pero elevan el coste del proyecto y consumen excesiva energía del dispositivo. Por la necesidad de encontrar una tecnología de menor consumo emergen las denominadas "Low Power Wide Area Networks" (LPWAN). Su rango varía dependiendo de la localización, obteniendo entre 10-40km en zonas rurales y hasta 5km en zonas urbanas, con un coste de menos de 2€para el radio chip y un coste operativo de 1€por dispositivo y año. Estas características hacen que sea altamente interesante para aplicaciones de IoT que requieran transmitir pequeñas cantidades de datos a gran distancia. Existen diversas soluciones en el mercado, las cuales se describen a continuación:

LoRa y LoRaWAN

Las siglas LoRa hacen referencia a largo alcance (en la bibliografía inglesa a "LOng RAnge") y es un protocolo de gestión de frecuencia, que modula la señal en

las bandas de frecuencia ISM (*Industrial, Scientific and Medical bands*) sin licencia, por debajo de los GHz [4], siendo estas diferentes según la región en la que el dispositivo se encuentre: 868 MHz en Europa, 915 MHz en Norte America o 433 MHz en Asia. LoRaWAN es un protocolo de comunicación basado en la anterior, que fue estandarizado en 2015, formando la denominada LoRa-Alliance. De esta forma, un dispositivo manda un dato y este es registrado de forma redundante por todas las estaciones en su rango, mejorando el ratio de mensajes entregados de forma exitosa, como se observa en la Figura 4.2.

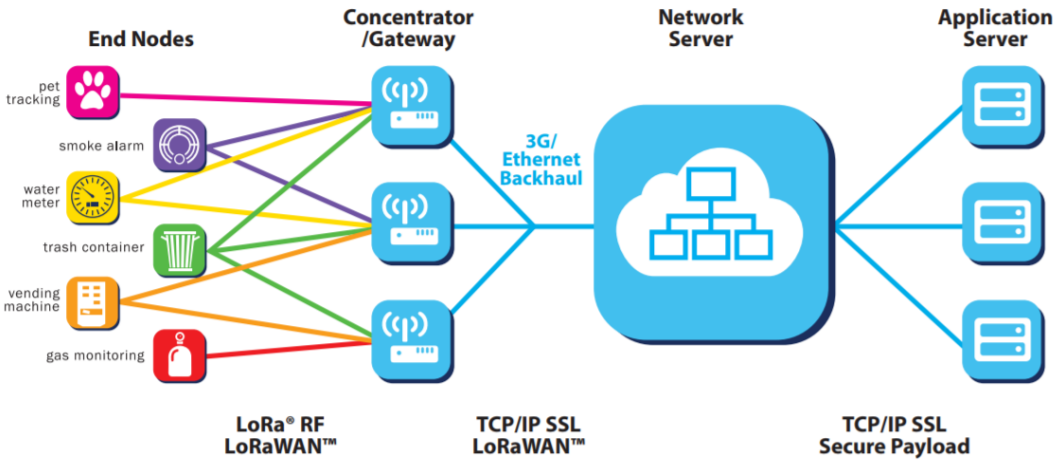


Figura 4.2: Estructura basada en LoRaWAN / LoRa.

The Things Network [22] compone una red social de puertas de acceso a la red de LoRaWAN, de forma que diversos usuarios pueden implementar dichos "gateways" en sus comunidades para así poder tejer junto con otros usuarios, una red donde poder implementar sus aplicaciones haciendo uso de este protocolo (Figura 4.3).

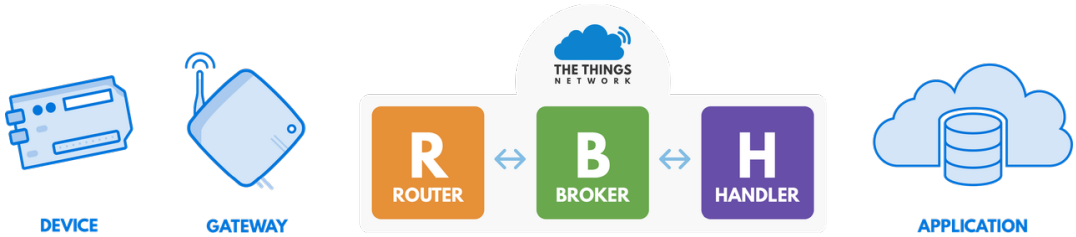


Figura 4.3: Funciones de The Things Network.

SigFox

Es una tecnología patentada que funciona de forma similar a la anterior, con la diferencia de que es la propia compañía la que ya brinda una red ya configurada por la misma. Solo es necesario configurar los dispositivos y disponer de una red disponible en las inmediaciones del mismo [22] como se observa en la Figura 4.4.

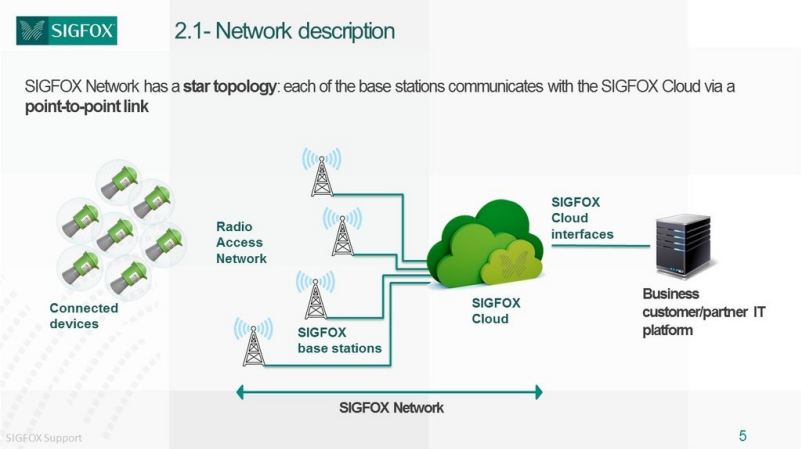


Figura 4.4: Topología de SigFox.

Nb-IoT

Sus siglas hacen referencia a Narrow Band IoT, y coexiste con las redes GSM (*Global System for Mobile communications*, 2G, 3G) y las LTE (4G, 5G) bajo frecuencias con licencia de hecho, está basado en este último.

Comentarios finales

Las características técnicas de estas tres tecnologías mencionadas se muestran en la Tabla 4.2.

Se observa que SigFox ofrece rangos de transmisión mayores que las otras dos alternativas. Sin embargo, la capacidad de transmisión de datos es muy reducida, con una baja tasa de datos, del orden de bits/s, y con limite de mensajes diarios. LoRaWAN y NB-IoT brindan mayores prestaciones en este aspecto y por tanto la dimensión de los mensajes así con la tasa de envío de los mismos puede ser mayor, aunque hay que tener en cuenta que sus anchos de banda siguen siendo pequeños y no permitirán satisfacer las necesidades de proyectos con alta tasa de envío de datos.

	Sigfox	*LoRaWAN*	*NB-IoT*
Modulación	BPSK	CSS	QPSK
Frecuencia	Bandas ISM sin Licencia (868 MHZ en Europa, 915 MHZ en Norte América y 433 MHz en Asia)	Bandas ISM sin Licencia (868 MHZ en Europa, 915 MHZ en Norte América y 433 MHz en Asia)	Bandas licenciadas LTE
Ancho de banda	100 Hz	250 kHz y 125 kHz	200 kHz
Tasa de datos máxima	100 bps	50 kbps	200 kbps
Bidireccionalidad	Limitado	Si	Si
Mensajes/día máximos	140	Ilimitados	Ilimitados
Alcance	10 km (urbano), 40 km (rural)	5 km (urbano), 20 km (rural)	1 km (urbano), 10 km (rural)
Inmunidad ante interferencias	Muy alto	Muy alta	Baja
Encriptación	No	Si (AES 128b)	Si (Encriptación LTE)
Adaptación de tasa de datos	No	Si	No
Entregas	Los dispositivos entregan a varias estaciones.	Los dispositivos entregan a varias estaciones	Los dispositivos entregan a una sola estación
Localización	Si (RSSI)	Si (TDOA)	No
Permite redes privadas	No	Si	No
Estandarización	Colabora con ETSI en la estandarización de su red.	LoRa Alliance	3GPP

Tabla 4.2: Comparativa técnica de las tecnologías LPWAN [adaptado de 3.2]

Finalmente, en la Tabla 4.3 se muestra un resumen de lo expuesto hasta el momento.

Tecnología	***Datos aprox. [kbps]***	***Alcance aprox. [m]***	***Soporta movilidad***
NFC	424	0.1	Si
ZigBee	250	100	Si
Bluetooth	1000	100	Si
WiFi	54000	150	Si
LTE	1000	11000	Si
LoRaWAN	0.3	14000	Si
Sigfox	0.1	17000	Si
NB-IoT	200	20000	No

Tabla 4.3: Resumen de las características de los protocolos de transporte.

4.2. Protocolos de comunicación

Los datos enviados por los medios anteriormente expuestos no están estructurados. Para que los dispositivos sean capaces de interpretar la información y por tanto, ser capaces de comunicarse, hace falta establecer un lenguaje común entre los mismos y una serie de reglas a seguir. A esto se le denomina protocolo [23].

Aunque en teoría, el medio de transporte y el lenguaje empleado puedan parecer conceptos independientes, la realidad demuestra qué debido a las características de cada uno, existen ciertas incompatibilidades, por lo que esto es un factor a tener en cuenta al elegir que protocolo usar. En el Internet de las Cosas, el uso del protocolo

MQTT está más que estandarizado para casi cualquier tipo de circunstancia por los motivos que se expondrán en el Apartado 4.2.3. No obstante, se exponen brevemente otros protocolos.

4.2.1. HTTP / REST

HTTP (Hyper Text Transfer Protocol) [24] es un protocolo sin estado, altamente estandarizado en las páginas web, con una estructura de petición/respuesta unidireccional, en el que el cliente manda una petición y el servidor devuelve la respuesta a la misma (por ejemplo, la temperatura de un sensor). Está optimizado para manejar grandes cantidades de datos y debe conectarse y desconectarse del dispositivo cada vez que desee realizar una acción (escribir o leer datos), no siendo estas peticiones muy ligeras ni relativamente rápidas, lo cual supone un problema [23].

REST (Representational State Transfer) no es exactamente un protocolo y está íntimamente ligado a HTTP. Permite la transmisión de datos con las ordenes get/-put/post/delete. La ventaja de usar esta combinación HTTP/REST es que al ser tan estandarizado, cualquier dispositivo con conexión a internet ya cuenta con un módulo del mismo integrado. Debido a que muchas aplicaciones de IoT se basan en el estado de un dispositivo únicamente, el proceso se simplifica bastante [23].

El problema de usar este protocolo es que cada vez que se necesita consultar el dispositivo o realizar un cambio, se debe realizar una petición y esperar a una respuesta, lo que conduce a una transmisión excesiva de datos, con respuestas lentas y alto consumo de batería. Por lo que los requisitos de la aplicación en términos de autonomía y rapidez de respuesta jugarán un papel clave a la hora de seleccionar o no este tipo de protocolo [22].

4.2.2. CoAP

CoAP (*Constrained Application Protocol*) es un protocolo [24] dedicado a redes con ciertas limitaciones (poca potencia o propensa a perdidas). Sigue el mismo modelo de petición / respuesta usando los conceptos get/put/post/delete que posee REST, pero estructurando los paquetes de datos en una estructura mínima y muy ligera, representado la mayor parte de datos de forma binaria para ello.

4.2.3. MQTT

Los protocolos anteriores pueden ser usados en aplicaciones de IoT sin problemas, pero son demasiado pesados MQTT es un protocolo de manejo de mensajes

extremadamente ligero y sencillo, basado en un sistema de publicación/suscripción, especialmente idóneo para intercambiar datos en la nube en tiempo real, desarrollado por IBM (*International Business Machines*) a finales de los 90 y estandarizado en 2013 [24]. Posee una arquitectura en estrella (Figura 4.5) y está formada por tres participantes [25] :

- *Broker:* Se encarga de gestionar el intercambio de mensajes entre los participantes. Todos estos se encuentran conectados a él, y únicamente a él, es decir, no existe conexión entre dispositivos de forma directa.
- *Publicadores:* Son aquellos dispositivos que envían datos al broker para que este lo contenga hasta que otros dispositivos los demanden.
- *Suscriptores:* Son los dispositivos que recogen la información enviada al broker por los publicadores.

Figura 4.5: Topología en estrella del protocolo MQTT.

El broker es iniciado una sola vez y permanece en constante funcionamiento, permitiendo un intercambio de datos bidireccional. Para que los subscriptores puedan acceder a los datos que realmente están buscando, estos son publicados en los denominados "topics", etiquetas de texto similares a las URLs en las cuales se categoriza los datos de forma que a cada etiqueta que se añade hace al topic más específico [23]. De esta forma, permite manejar los mensajes de forma asíncrona ya que un dispositivo puede publicar un dato en un topic determinado, siendo este almacenado por el broker hasta que es demandado en otro momento por el subscriptor.

Un sistema de iluminación en una casa inteligente por ejemplo, puede estar categorizada de forma que las bombillas de cada planta y de cada habitación se pueden controlar de forma independiente. La Tabla 4.4 ejemplifica el proceso.

Topic	Función
Bombillas/p1/b1	Permite controlar con un solo mensaje On/Off la bombilla designada como uno en la cocina.
Bombillas/p1/cocina	Permite controlar con un solo mensaje On/Off las bombillas de la cocina de la primera planta
Bombillas/p2/#	El hashtag permite controlar todas las bombillas de cualquier habitación de la segunda planta.

Tabla 4.4: Ejemplo de llamadas a los topics.

Algunas características [25]

Calidad del servicio. QoS (*Quality of Service*), permite gestionar la robustez del envio de mensajes al cliente ante posibles fallos. Pueden definirse 3 calidades de servicio diferentes:

- *QoS 0:* El mensaje es transmitido como máximo una única vez sin esperar confirmación de de que se ha recibido. Es el nivel usado por defecto.
- *QoS 1:* El cliente continua enviando el mensaje hasta que el broker confirma su envio a la red, por lo que se asegura que el mensaje se entrega al menos una vez.
- *QoS 2:* Asegura que el mensaje es recibido exactamente una vez. Una mejor calidad de servicio aumenta la robustez del sistema, pero también supone una mayor latencia y la necesidad de mayor ancho de banda.

Una mejor calidad de servicio aumenta la robustez del sistema, pero también supone una mayor latencia y la necesidad de mayor ancho de banda.

Retención de mensajes. Permite al broker si se desea, almacenar una copia de los mensajes publicados, para que cuando nuevos suscriptores se conencten a posteriori, también puedan recibirlos.

Última voluntad. Cuando un cliente se desconecta de forma inesperada, permite lanzar un mensaje que indique de ello. Es útil cuando otros clientes están suscritos al cliente desconectado, y la ausencia de mensajes puede causar problemas de seguridad o básicamente un funcionamiento inadecuado

Parte III

Desarrollo del proyecto

Capítulo 5

Estudio de potenciales componentes del proyecto

Contenido

5.1. Capa de percepción

5.1.1. ESP-2866

Es un chip desarrollado por la compañía Espressif caracterizado por ser uno de los primeros en acercar las posibilidades del IoT a los usuarios con un bajo coste (alrededor de 3 euros), al contar con modulo WiFi integrado, al igual que el Arduino MKR1000 en 2016, aunque teniendo este un coste muy superior, entorno a 40 euros.[1] La Figura 5.1 muestra la compacidad y simplicidad del chip.

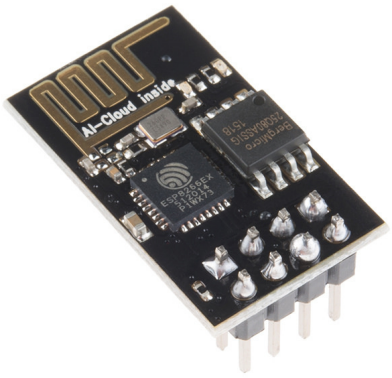


Figura 5.1: Chip ESP2866.

Las características que hacen a este microcontrolador ideal para aplicaciones IoT son [26]:

- *Bajo consumo:* Operación entre 80 mA y 170 mA cuando se está transmitiendo al máximo de su capacidad.[2]
- *Gestión energética eficaz:* Puede entrar en modo sueño, manteniendo la sincronización estando alerta de eventos que le hagan volver a funcionar a plena carga, con un consumo de 0.6mA-1mA o modo sueño profundo, con un consumo de alrededor de 20 μA. Esto supone un gran avance para dispositivos portables.
- *Compacto:* Lo que mejora su portabilidad.

[1] https://store.arduino.cc/arduino-mkr1000-wifi
[2] https://www.espressif.com/en/products/socs/esp8266/overview

(Acceso online el 22 de Agosto de 2020)

5.1.2. ESP-32

El ESP32 es la evolución final de una serie de chips con nacimiento posterior al ESP2866. La propia empresa lo define como una solución para microcontroladores que no disponen de conectividad, de forma que puede emplearse como puente de acceso a la red y soluciones IoT [26]. Además, es capaz de ejecutar internamente aplicaciones en tiempo real.

La Tabla 5.1 muestra una comparativa técnica entre ambos chips en profundidad:

CARACTERÍSTICAS	***ESP 2866***	***ESP 32***
Procesador	Tensilica LX106 32 bit 80-160 MHz	Tensilica Xtensa LX6 32 bit Dual core 160-240 MHz
Memoria RAM	80 KB (40 disponibles)	520 KB
Memoria flash	Hasta 4 MB	Hasta 16 MB
ROM	No	448 KB
Alimentación	3.0 V a 3.6 V	2.2 V a 3.6 V
Rango de temperaturas	-40ºC a 125ºC	-40ºC a 125ºC
Consumo de corriente	80 mA (promedio) - 225 mA (máximo)	80 mA (promedio) - 225 mA (máximo)
Consumo en deep-sleep	20 uA	2.5 uA
Coprocesador	No	Si
WiFi	802.11 b/g/n WEP, WPA	802.11 b/g/n WEP, WPA
Bluetooth	No	v4.2 BR/EDR y BLE
UART	2 puertos	3 puertos
I2C	1 interfaz	2 interfaces
SPI	2 interfaces	4 interfaces
GPIO	32 pines	11 pines
PWM	8 canales	16 canales
ADC	1 (10 bit)	2 (8 bit)
CAN 2.0	No	1 bus
Ethernet	No	10/100 Mbps
Sensor de temperatura	No	Si
Sensor hall	No	Si
Infrarojos	No	Si
Timers	3	4
Encriptación	No (TLS)	Si (AES, SHA, RSA, ECC)
Secure boot	No	Si

Tabla 5.1: ESP2866 vs ESP32. Características.

Se observa que son similares, pero el ESP32 conforma una versión con caracterísiticas técnicas superiores. De todas las características mostradas en la Tabla 5.1, las más destacadas con respecto a su antecesor son:

- Doble nucleo.
- Coprocesador de bajo consumo.
- Posee tecnología Bluetooth.

- Mayor número de puertos de comunicación I2C.
- Cuenta con sensores internos de temperatura y hall.

5.1.3. Raspberry Pi

La Raspberry Pi [27] es la placa de un ordenador sencillo, compuesto por un SoC (*Software on Chip*), CPU, Memoria, RAM, puertos de entrada, ranura SD, etc. En definitiva, un ordenador de bajo coste y de tamaño muy reducido, que llega a caber en la palma de la mano, con el que se puede llegar a hacer infinidad de proyectos. Los sistemas operativos sobre los que puede funcionar son múltiples, entre los que destacan Ubuntu, Raspbian (Debian) o Pidora (Fedora).

A lo largo del tiempo han surgido nuevas versiones, cuyas características básicas pueden observarse en la Tabla 5.2:[3]

MODELO	*SOC*	*FRECUENCIA DEL RELOJ [MHz]*	*RAM*	*PUERTOS USB*	*ETHERNET*	*WIFI / BLUETOOH*
Raspberry Pi Model A+	BCM 2835	700	512 MB	1	No	No
Raspberry Pi Model B+	BCM 2835	700	512 MB	4	Si	No
Raspberry Pi 2 Model B	BCM 2836 / 2837	900	1 GB	4	Si	No
Raspberry Pi 3 Model B	BCM 2837	1200	1 GB	4	Si	Si
Raspberry Pi 3 Model B+	BCM 2837B0	1500	1 GB	4	Gigabit ethernet sobre USB 2.0	Si
Raspberry Pi Zero	BCM 2835	1000	512 MB	1	No	No
Raspberry Pi Zero W	BCM 2835	1000	512 MB	1	No	Si

Tabla 5.2: Modelos de Raspberry Pi y sus características.

Para proyectos IoT es necesario conexión a internet, por lo que solo la Raspberry 3 y la Zero W pueden ser utilizadas con conexión WiFi (también disponen de Bluetooth), no obstante, es posible dotar a las demás de conexión inalambrica comprando modulos Wifi para las demás por un precio adicional.

No obstante, debido a su mayor potencia, requieren de una fuente de alimentación de 12V y 2.5A, por lo que su consumo energético es claramente mayor a los

[3]Tabla adaptada de la siguiente web. Acceso online el 20 de Agosto de 2020.

https://www.xataka.com/makers/cero-maker-todo-necesario-para-empezar-raspberry-pi

anteriormente descritos, lo que puede suponer un problema en portabilidad (tamaño de baterías) y autonomía (alto consumo).

5.1.4. M5-Stack

No se trata de un microcontrolador en sí, sino de una placa de desarrollo basada en ESP32, pero diseñado de tal forma que es portable, open source y sobretodo, modular. Posee alrededor de 30 modulos apilables magnéticamente y de 40 unidades de extensión, de forma que partiendo de un nucleo base, se puede configurar una placa que cumpla con practicamente cualquier necesidad de forma muy compacta, como si de un Lego se tratase (de hecho la propia marca usa dicho termino para referirse al montaje). Algunos de estos modulos dotan a la placa de capacidad de comunicarse con diversos protocolos, como LoRa o Nb-IoT de forma sencilla [28]. Puede ser programado en gran variedad de plataformas, como Arduino IDE o UIFlow, que permite programación por bloques; y lenguajes, como C, C++, Micropython, etc.

Su gran versatilidad lo convierte en una herramienta increiblemente potente. Algunas de las características técnicas del módulo basic (Tabla 5.3) son:

RECURSOS	***PARÁMETROS***
ESP32	240 MHz (dual core), 600 DMIPS, 520KB SRAM, WiFi, dual mode Bluetooth
Memoria flash	16 MB
Alimentación	5V @ 500 mA
Puertos	USB Tipo C x1 , GROOVE (I2C + I/O + UART) x1
Pantalla IPS	2 pulgadas
Botones	Si. 3 botones personalizables
Altavoz	1W - 0928
Batería	110 mAh @ 3.7 V
Antena	2.4 GHz tridimensional
Temperatura de operación	0ºC a 40ºC
Peso neto	47.2 g
Tamaño	54 x 54 x 18 mm
Material de carcasa	Plastico (PC)

Tabla 5.3: Características técnicas del M5Stack Core Basic.

Otros núcleos como el Gray o el Fire suponen pequeñas evoluciones del basic aumentando la capacidad de la batería de serie o añadiendo una IMU (Inertial Measurement Unit). Los núcleos Stick por su parte, reducen el tamaño del dispositivo para hacerlo todavía más portable. Todos cuentan con una pantalla LCD.

5.2. Capa de procesado.

El procesamiento de los datos captados puede constar de varias capas, según la aplicación. Pero en lineas generales, primero es necesario captar los datos en un servidor y posteriormente almacenarlos en una base de datos para poder analizarlos posteriormente.

5.2.1. Brokers

Cuando se hace uso de protocolos como MQTT, se debe proveer al sistema de un broker que gestione el recibo y entrega de mensajes a los publicadores y subscriptores. En el caso de hacer uso de plataformas de computación en la nube, la gran mayoría ya cuentan con uno en su infraestructura de forma que solo hay que establecer comunicación entre el dispositivo y la plataforma. En los apartados posteriores se ahonda más en el tema. No obstante, si se opta por no usarlas, existen varias opciones:

Broker privado: Existen numerosos brokers MQTT de código abierto disponibles, cada uno con diversas características que lo hacen más o menos adecuado para cada aplicación. Algunos de ellos[4] son:

- *Aedes:* Servidor diseñado para node.js.
- *EMQTT:* Diseñado para aplicaciones con grandes exigencias de escalabilidad.
- *Mosquitto:* Es el más conocido del sector. Es liviano y adecuado para servidores de baja potencia.

Los brokers anteriormente mencionados pueden ser instalados en una máquina virtual con el sistema operativo que necesiten para funcionar, en un PC, o bien en una placa de bajo coste, como la Raspberry Pi.

Broker público: Si no se desea configurar un broker propio, en la red existen multitud de brokers de acceso público, tanto gratuitos como de pago. Generalmente son empleados para hacer pruebas y prototipos por cuestiones de seguridad. Algunos de ellos se muestran a continuación en la Tabla 5.4.

[4] https://www.luisllamas.es/principales-broker-mqtt-open-source-para-proyectos-iot/

(Acceso online el 20 de Agosto de 2020)

BROKER	*TIPO*	*CARACTERÍSTICAS*
test.mosquitto.org	Gratuito	Sin garantías de disponibilidad permanente. Indicado para prototipado.
mqtt.dioty.co	Gratuito	Incluye aplicación móvil.
mqtt.flespi.io	Gratuito	Velocidad de transmisión muy rápida
CloudMQTT [29]	De pago	En la versión más económica, admite 25 clientes a una velocidad de 20 kb/s.

Tabla 5.4: Algunos brokers públicos y sus características.

Sea cual sea la opción elegida, una cualidad muy importante de estos es que puedan funcionar correctamente sin necesidad de estar conectados en la misma red de los dispositivos.

5.2.2. Bases de datos

Debido a que el proposito del proyecto es realizar un analisis de los datos recogidos, es necesario almacenarlos en una base de datos. No obstante, existen varias categorías, aunque básicamente se pueden dividir en bases de datos basadas en SQL (*Structured Query Language*) y NOSQL. Es importante describirlas brevemente para poder seleccionar la más adecuada:

Bases de datos relacionales SQL

Este tipo de bases de datos está formada por un conjunto de una o más tablas estructuradas por *registros (lineas)* y *campos (columnas)*, las cuales están relacionadas por un campo en común denominado *identificador o clave*, de forma que quedan relacionadas entre sí por este[5] y se evitan los datos duplicados.

El lenguaje SQL, es considerado el estandar para acceder y manipular datos en las bases de datos [28] de tipo relacional, tanto que se puede usar base de datos SQL o relacional de forma arbitraria. Destaca por la facilidad de navegación y edición de las tablas por parte de los usuarios, al emplear un lenguaje de muy alto nivel.

Existen diversos tipos de bases relacionales, cada una enfocada a diversas funciones, como se representa en la Tabla 5.5.

[5] https://styde.net/que-es-y-para-que-sirve-sql/

(Acceso online el 20 de Agosto de 2020)

SGDB	***CARACTERÍSTICAS***
mySQL	Es una de las bases de datos más populares en desarrollo de entornos web.
MariaDB	Es un sistema deribado de MySQL, mejorado y más optimizado.
Influx DB	Enfocada al almacenamiento rápido y optimizado de grandes series de tiempo para realizar análisis en tiempo real.
PostgreSQL	*Orientada a objetos. Buena elección para sistemas con datos complejos.*

Tabla 5.5: Algunas bases de datos relacionales y sus características.

Bases de datos NOSQL

Estas bases de datos han comenzado a ser usadas en sistemas en los que se deben manejar ingentes cantidades de datos, que a diferencia de las SQL, están desestructurados, pudiendo estar enfocados a representaciones gráficas, documentos, geolocalizaciones, etc. En general, a cualquiera de los campos en los que las bases relacionales no funcionan especialmente bien. Su uso se hace imprescindible para aplicaciones de Big Data, ya que las SQL no actuan de forma eficiente ante esta situación. Las grandes compañias, como Google, Amazon o Facebook hacen uso de ellas [30].

Infraestructuras

Una base de datos está formada por un conjunto de componentes y no solo por el SGDB (*Sistema Gestor de la Base de Datos*). A dicho conjunto se le denomina infraestructura. Existen numerosos tipos, las cuales se destacan en la Tabla 5.6.

A pesar de que la única diferencia parece ser el sistema operativo sobre el que son implantados, hay que tener en cuenta que LAMP es una infraestructura más seguro que XAMPP, por lo que si se planea que la base de datos no trabaje en local, sino que esté abierto a la red, será preferible.

5.2.3. Node-Red

Node-RED, originalmente desarrollada por IBM, es una herramienta de programación basada en flujos [31] que permite establecer las relaciones entre los diversos nodos del sistema de forma sencilla y robusta, lo cual es ideal para sistemas IoT ya que simplifica la interconexión de los diversos dispositivos que componenen el

INFRAESTRUCTURA	***COMPONENTES***	
XAMPP	X	Linux, Windows o Mac OS X.
	A	Apache, como servidor web.
	M	MySQL, como SGDB.
	P	PHP, lenguaje de programación en el lado del servidor.
	P	Perl, lenguaje de programación para administrar el sistema.
LAMP	L	Linux
	A	Apache, como servidor web.
	M	MySQL, como SGDB.
	P	PHP, lenguaje de programación en el lado del servidor.
WAMP		Igual que LAMP pero únicamente para Windows.
MAMP		Idem, pero para MAC OS.

Tabla 5.6: Tipos de infraestructuras de bases de datos.

sistema, ahorrando gran cantidad de lineas de código. Actualmente se encuentra integrado en gran cantidad de dispositivos IoT en el mercado y su apariencia se muestra en la Figura 5.2.

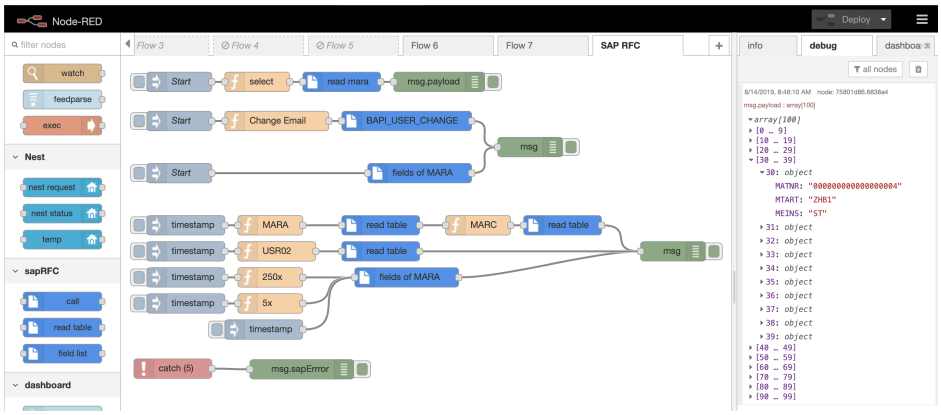


Figura 5.2: Ejemplo de flujos en Node-Red.

Cuenta con multitud de nodos disponibles basados en Node.js, por lo que la programación en JavaScript queda en el lado del servidor[6] y permite crear nuevos como una función programada en Javascript por parte del usuario. Debido a su bajo consumo de recursos, puede ser implementado en equipos locales o en placas como la Raspberry Pi.

[6] https://www.techedgegroup.com/es/blog/fundamenos-node-red

(Acceso online el 20 de Agosto de 2020)

5.3. Capa de aplicación

5.3.1. Node-Red UI

Dentro del servicio de Node-Red se incluye un paquete de nodos que permite generar una interfaz gráfica. Con los nodos de funciones y bases de datos, se pueden realizar consultas que pueden ser representadas en forma de gráficas, indicadores, etc. Permite un alto grado de personalización con los conocimientos necesarios.

5.3.2. Grafana

Grafana es un servicio PaaS que permite representar datos de varias bases de datos en un mismo panel, con una interfaz muy elegante y profesional. Permite realizar cualquier tipo de consulta escrita directamente en lenguaje SQL y representar los datos de diversas maneras. También permite la creación de usuarios y roles dentro del dashboard, limitando lo que puede usar y ver cada uno [32].

5.4. Plataformas en la nube open source

Como ya se ha mencionado anteriormente, la computación en la nube permite aunar las diversas capas de procesamiento y aplicación de forma sencilla mediante una sola o una combinación de varias plataformas, permitiendo una gestión ágil y eficiente de grandes volúmenes de datos. Aquellas que poseen código abierto al público brindan gran capacidad de personalización y existe amplio número de opciones, tanto gratuitas como de pago. Las cantidad de plataformas en la nube open source que pueden encontrarse en la actualidad son prácticamente ilimitadas, contando la mayoría de ellas con características similares. Es por ello que a continuación se describen las más interesantes.

5.4.1. Kaa IoT

Compone una tecnología destinada al desarrollo de IoT a nivel empresarial a cualquier escala y para multitud de ámbitos, como comercio, deporte, salud, agricultura, etc.

Algunas de sus características son [33]:

- Cuenta con un modelo de microservicios, permitiendo un alto grado de customización.
- Es independiente de la tecnología, permitiendo a los desarrolladores emplear cualquier lenguaje de programación.
- Flexible en el modelo de servicios que se desee emplear: nube pública, privada, hibrida, etc.
- Usa protocolos abiertos como MQTT y CoAP.
- Es escalable y auto-reparable, adaptándose al número de clientes que se requiera y restaurándose ante posibles fallos.

5.4.2. Thingspeak

Thingspeak es la plataforma más potente de las mostradas en este capítulo, ya que además de permitir la recogida y almacenamiento y visualización a posteriori en la nube de los datos captados por los dispositivos, su característica diferenciadora es que enfoca su potencial al análisis de dichos datos [34]. Además, hace posible la creación de prototipos y sistemas IoT sin tener que configurar servidores o desarrollo de software web.

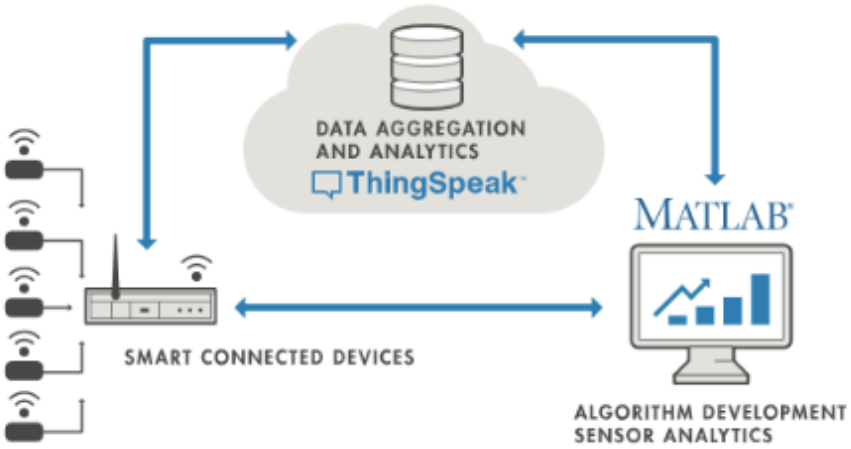


Figura 5.3: Arquitectura de Thingspeak.

Sus capacidades más importantes pueden condensarse en los siguientes puntos:

- Configuración sencilla de los dispositivos, haciendo uso de protocolos populares. Brinda la opción de almacenar los mismos en nubes privadas por defecto, pero también habilita la posibilidad de hacer uso de canales públicos.
- Proporciona acceso a una librería que interconecta MATLAB con la plataforma, permitiendo hacer uso del potencial matemático del software para realizar el análisis de datos en tiempo real aplicando complejos algoritmos, reconocimiento de padrones, etc. Permitiendo aplicaciones de desarrollo de modelos predictivos y de "machine learning".
- Permite configurar respuestas automáticas tras el análisis de dichos datos, pudiendo crear alertas a través de otros servicios de terceros, por ejemplo, Twitter.

5.4.3. Adafruit IO

Adafruit es una compañía tecnológica originalmente fundada como una comunidad online para aprender electrónica, que opera como desarrolladora de soluciones de hardware para aplicaciones específicas demandadas por la comunidad. De esta forma ofrece multitud de placas enfocadas a IoT y, actualmente, ofrece una plataforma en la nube de tipo PaaS [35].

Sus características más importantes son similares a las anteriormente descritas, aunque posee otras cualidades interesantes [36]:

- Configuración de alarmas (Triggers en inglés) que son enviadas a otras plataformas, como al correo, cuando un dispositivo alcanza un estado determinado.
- Integración de IFTT (If this then that).
- Gran soporte técnico y comunitario. Elevado número de productos y librerias ya creadas para los mismos, facilitando la integración.

Posee una versión gratuita y una de pago.

5.5. Plataformas en la nube no open source

La capacidad de almacenamiento y la potencia computacional de estas plataformas es muy superior a las anteriormente descritas, ofreciendo una mayor cantidad

de servicios. Son suministradas por grandes compañías como Google o Microsft y están enfocadas a aplicaciones a gran escala, creación de modelos predictivos y sobretodo para aquellos usuarios interesados en contar con una capa de negocio en su aplicación IoT. No obstante, son plataformas modulares en las cuales puede pagarse solo por aquellos servicios que se usen, permitiendo a pequeñas empresas y usuarios su uso. Algunas de estas plataformas son:

Amazon Web Services. AWS IoT es un servicio muy completo que proporciona una comunicación bidireccional segura entre los dispositivos conectados a internet, tales como sensores o acturadores, y la nube. Esto permite recopilar datos de telemetría, y su posterior almacenamiento y análisis. Finalmente permite crear aplicaciones que permitan a los usuarios controlar dichos dispositivos desde sus dispositivos móviles [37].

Microsoft Azure IoT. Creada por Microsoft, Azure es una plataforma que brinda a sus usuarios numerosas tecnologías para crear soluciones IoT a medida, lo que supone un producto con alta flexibilidad. Permite gestionar tanto *IoT devices* como *IoT Edge devices*. Además, los módulos *Azure Stream Analytics* o *Azure Databricks* permiten el análisis y procesado de grandes flujos de datos, permitiendo el aprendizaje automático tras el mismo para generar algoritmos predictivos si además, se hace uso de *Azure Machine Learning [38].*

Google Cloud IoT. Al igual que las anteriores, hace posible el procesado, almacenamiento y análisis de los datos. Permite el uso de gran variedad de sistemas operativos, aunque destaca su compatibilidad con Debian Linux [39].

Capítulo 6

Analisis de soluciones

Contenido

6.1. Selección de arquitectura

Sin duda es la primera decisión que se ha de tomar ya que determina el transcurso de la implementación de la plataforma, pues, decidiendo usar una de las arquitecturas en la nube descritas en el Capítulo 5 se ahorra tiempo en la implementación de los elementos que la componen, para ceñirse estrictamente al procesado y gestión de los datos para el servicio final (Tabla 6.1).

VENTAJAS	***INCONVENIENTES***
Multitud de widgets	Hay que pagar.
Sencillez	Pueden poseer limitaciones o restricciones.
Para aplicaciones muy grandes.	No se posee control total.

Tabla 6.1: Ventajas e inconvenientes de implementación en la nube.

Por otro lado, implementar los componentes de forma autónoma requiere la instalación y configuración de brokers, bases de datos, UI (*User Interface*), etc. Lo cual dificulta la implementación (Tabla 6.2), pero se posee control absoluto sobre el tráfico de datos y la gestión de los mismos y resulta una opción más económica en la mayoría de los casos.

VENTAJAS	***INCONVENIENTES***
Control total.	Complejidad de configuración.
Económico.	Sin servicio técnico.
Didáctico academicamente	Requiere más softwares.

Tabla 6.2: Ventajas e inconvenientes de implementación autónoma.

6.1.1. Valoración técnica y económica

Se analizan los aspectos más relevantes de cada opción, tanto las cualidades técnicas como el aspecto económico, para llegar finalmente a una conclusión.

Plataformas en la nube

Cada plataforma posee diversos planes de pago, incluso planes gratuitos, por lo que el analisis técnico no puede ser separado del económico. Se considera mejor plataforma aquella que con el plan más económico permita implementar este proyecto respetando las consideraciones expuestas en el Apartado 1.4 y en el Apartado 1.5.

Kaa Iot [33] ofrece la posibilidad de emplear 5 dispositivos de forma gratuita en su modelo KAA Cloud, lo cual no es viable por que el número de sensores es 6 por andador, por lo que habría que recurrir a modelos de pago. La siguiente opción más económica, de 12.74 €/mes permite hasta 15 dispositivos, por lo que podría alojar 2 andadores.

La Tabla 6.3 resume las características y los aspectos económicos más relevantes de cada uno de los plantes ofrecidos.

KAA IoT		
Planes	**Características**	**Precios [eur/mes]**
KAA Cloud	Proporciona un servicio PaaS	5 dispositivos gratis
	5-1000 dispositivos	
	Sin límite de tasa de datos.	15 dispositivos - 12
	1 GB por dispositivo	
KAA Hosted	Proporciona una máquina virtual (4 nucleos, 16GB RAM)	Opción más económica - 250 dispositivos - 499
	250-2500 dispositivos	
	Sin límite de tasa de datos.	
KAA Self-Hosted	Permite alojar los recursos en una infraestructura propia (nube hibrida)	Opción más económica - 25 dispositivos - 99
	25- dispositivos ilimitados	

Tabla 6.3: Planes de Kaa IoT y sus características.

La gran ventaja de Kaa IoT es que no ofrece un límite de tasa de datos, a diferencia del resto, por lo que simplifica el proceso ya que no es necesario crear paquetes de datos.

Thingspeak [34] ofrece diversos planes según el ámbito de aplicación de la plataforma, ya sea uso comercial, personal, académico o de estudiante, ofreciendo facilidades económicas para los ambitos más académicos pero con limitaciones de rendimiento. Dentro del plan académico se ofrecen las opciones mostradas en la Tabla 6.4.

Se observa que en el plan gratuito, solo se permiten 3 subscripciones MQTT lo cual obliga a tener que usar el plan de pago, que permite 50,habilitando espacio para alojar 8 andadores en la misma plataforma. No obstante, el intervalo entre mensajes solo puede ser de 1 segundo, muy superior a los 50-100Hz deseados, por lo que es necesario condensar los datos capturados en 1 segundo en un gran paquete de datos, para que estos puedan ser enviados en un solo mensaje.

THINGSPEAK Planes académicos disponibles	**Gratuito**	**Pago**
Intervalo entre mensajes [s]	15	1
Canales	4	250
Subscripciones MQTT máximas	3	50
Tiempo máximo de cálculo permitido [s]	20	60
Precio [eur/año]	0	212.4

Tabla 6.4: Características de plan académico de Thingspeak.

Adafruit IO [35] ofrece dos planes únicos, cuyas caracterísiticas fundamentales se observan en la Tabla 6.5

ADAFRUIT IO	**Gratis**	**Pago**
Mensajes / minuto	30	60
Almacenamiento [días]	30	60
Topics permitidos	5	Ilimitado
Precio [eur/año]	0	85

Tabla 6.5: Características de planes de Adafruit IO.

La opción gratuita no puede ser empleada en el proyecto, pues limita los topics a solo 5, mientras que se necesitan 6 como mínimo. La opción de pago brinda un número ilimitado de topics, por lo que este problema queda resuelto, sin embargo, el número de mensajes/minuto supone una complicación ya que al igual que Thingspeak, implica agrupar los datos en paquetes de datos para que todos puedan ser recibidos por la aplicación. Para el caso de la frecuencia mínima de funcionamiento 50Hz:

- $50\ \text{Hz} = 50\ \frac{mensajes}{segundo} \longrightarrow 3000\ \frac{datos}{min\ sensor}$
- $6\ \text{sensores} \longrightarrow 18000\ \frac{datos}{min\ andador}$

Otro problema adicional es el almacenamiento máximo de los datos en la aplicación, por lo que si estos se desean tener en la nube por un tiempo superior a 60 días, se requiere alojarlos en una base de datos propia, lo cual añade una capa indeseada

más a la aplicación, lo cual no es ideal ya que se recurre a estas plataformas para simplificar el proceso al máximo.

El uso de las **plataformas no open source** descritas en el Capítulo 5, como AWS, Azure IoT o Google cloud ofrecen una potencia computacional y capacidad de almacenamiento muy superiores a todas la plataformas anteriormente descritas. Esto se traduce en un precio muy superior también, por lo que se descartan ya que su ámbito de aplicación está enfocado a aplicaciones de dimensiones mayores que requieran analisis en tiempo real, no necesarias en este proyecto.

Arquitecturas implementadas de forma autónoma

En el Capítulo 3 se describen los diversos tipos de arquitecturas que pueden ser implementadas. Es importante definir cual usar para poder seleccionar los elementos que compondrán la aplicación.No obstante, para esta aplicación la decisión es bastante sencilla, ya que una arquitectura de 3 capas es demasiado sencilla y una de 5 carece de interés ya que no se requiere de una capa de negocio para este producto. Por lo tanto, usar una arquitectura de 4 capas es ideal.

La *capa de percepción y red* es idéntica tanto si se usan plataformas en la nube o se implementa una arquitectura autónoma; los sensores recogen la información y son gestionados por un microcontrolador, que los envia al broker por medio de un protocolo de transporte y comunicación dado. Pero las capas de *procesado y apoyo a los servicios* no son gestionadas por un tercero sino por el propio usuario. Por lo general se puede recurrir a la instalación de paquetes que proporcionen estos servicios en una Raspberry Pi, de forma gratuita, por lo que los costes asociados a esta arquitectura son únicamente los de dicha placa. La capacidad computacional es suficiente para manejar la frecuencia de los datos y la capacidad de almacenamiento se adapta a las necesidades del usuario, por lo que no existe una limitación apreciable para un número reducido de andadadores.

6.1.2. Justificación de la selección

A continuación se muestra un resumen de las características más destacadas de cada una de las arquitecturas mencionadas:

PLATAFORMA	*UNIDADES*	*FRECUENCIA*	*PRECIO [euros/año]*
KAA IoT Cloud	15 sensores - 2 andadores	No hay límite	144
Thingspeak	50 subscripciones - 8 andadores	1 segundo	212.2
Open IoT	No especificado	No hay límite	Gratis
Aafruit IO	Sujeto al tamaño máximo de los paquetes	60 mensajes / min	85
No open source	Sujeto al precio, pero muy superior a lo requerido	Idem	369 - 10.000
Raspberry Pi	Sin limitación apreciable	Idem	40 euros, pago único

Tabla 6.6: Resumen de características técnicas y económicas más destacadas.

A priori, Adafruit la plataforma más barata pero su limitación de mensajes por minuto supone una complicación, por lo que KAA IoT es una mejor opción aún siendo algo más caro. No obstante, usar una Raspberry Pi o similar para implementar el middleware solo supone un pago único, por lo que económicamente es una gran ventaja. Además no existe una limitación apreciable para el número de andadores que se implementarán, ya que posee potencia de sobra para manejar unos pocos andadores y no presenta problemas de capacidad de almacenamiento en la base de datos ya que se puede conectar a discos duros de gran capacidad si la tarjeta SD no es suficiente.

Por tanto, **se selecciona una arquitectura de 4 capas,** implementando los componentes en una Raspberry o similar, dado a su bajo precio y flexibilidad para albergar más o menos unidades de andadores, de forma sencilla.

6.2. Protocolos de comunicación

En cuanto a la comunicación entre los sensores con el microcontrolador, el Capítulo 2 establece que las conexiones entre los sensores y la Raspberry Pi se realizan actualmente a través de CAN Bus y explica los motivos por los cuales es la opción más idónea, frente a otras como SPI o I2C. Por lo tanto, dado que los condicionantes no han cambiado, **se mantiene la comunicación por CAN bus.**

En cuando a los protocolos de comunicación posibles entre el microcontrolador y la plataforma:

- **HTTP** es un protocolo que puede ser útil para ciertas aplicaciones, pero es necesaria bidireccionalidad de datos para poder actuar sobre el servo. Además, para el registro de datos es un protocolo muy pesado y lento para la frecuencia de escritura que se desea.
- **CoAP y MQTT** son opciones mejores debido a su ligereza, no obstante, el primero está más enfocado a redes con mayores limitaciones y la estandarización de MQTT es muy superior. Además, su estructura es muy intuitiva e ideal para el posterior registro de los datos.

Por tanto, se hace uso de **MQTT como protocolo de comunicación** de los datos.

6.3. Selección de Hardware

A continuación se procede a seleccionar los componentes físicos con los que se procede a la captación de los datos recogidos por los sensores, así como todos los periféricos necesarios para su correcto funcionamiento.

6.3.1. Microcontrolador

En el Apartado 5.1 del Capítulo 5 se describían diversas opciones de microcontroladores.

La **Raspberry Pi** es la placa empleada por el andador en su versión actual, sin plataforma IoT.

- La captación y la mayor capacidad computacional de la Raspberry Pi permite un procesamiento de datos mejor, en caso de realizar filtrado antes del envio de los mismos.
- Permite aunar en un solo dispositivo, el broker[1] y el resto de capas de procesamiento como la base de datos o la representación gráfica de los datos almacenados. Pero sin embargo, lo anteriormente descrito solo es útil para una plataforma enfocada a un dispositivo, pues el objetivo del broker es manejar las publicaciones y subscripciones de varios clientes, por lo que se debe contar con un broker en una red independiente.

[1] En el caso de que la solución no haga uso de computación en la nube

Además, uno de los objetivos de la plataforma es permitir la portabilidad y la autonomía del dispositivo. El uso de la Raspberry Pi en referente a estos dos puntos cuenta con dos inconvenientes:

- Requiere de una fuente de alimentación de 2.5A, 12V, por lo que el tamaño y número de baterías a emplear es elevado.
- Además es necesario alimentar a los sensores por lo que el consumo anterior aumenta.
- No cuenta con modo deep sleep[2], lo cual supone un gran problema en la autonomía.

No obstante, dado que no se va a hacer uso de una plataforma de computación en la nube, sino que se va a crear una de forma particular, es necesario hacer uso de diferentes componentes de software. Por lo general, la Raspberry Pi suele ser la placa a elegir para este fin gracias a sus características, además de contar con gran cantidad de programas desarrollados para cada uno de los componentes de la aplicación final.

El **chip ESP2866** no se considera entre las opciones, pues el **ESP32** es un chip mucho más desarrollado, con el doble de potencia y más funcionalidades enfocadas al IoT. No obstante, a pesar de funcionar sobre este chip, la gran versatilidad y cantidad de módulos del **M5Stack**, unido a la sencillez para el prototipado y su compacidad lo hacen un dispositivo más atractivo para el desarrollo de este proyecto.

Por tanto, se selecciona un M5Stack por las siguientes razones:

- Consumo de 150 mA - 3.5V a plena carga, muy inferior a la Raspberry.
- Posee varios modos de sueño, permitiendo mejor gestión energética y autonomía.
- Compacidad y portabilidad.
- Modularidad.

[2] https://www.raspberrypi.org/forums/viewtopic.php?t=243421

(Acceso online el 24 de Agosto de 2020)

6.3.2. Stacks

M5Stack proporciona un numeroso abanico de modulos (stacks, en inglés) [28] que permiten adaptar el núcleo a las necesidades del proyecto. Este proyecto requiere en particular:

- **Un modulo que acepte conexiones CAN bus.** Es necesario recoger la señal CAN Bus y transformarla en una que sea capaz de leer el microcontrolador, ya que el núcleo original no es compatible (Figura 6.1a).
- **Un módulo de carga.** El diseño actual (vease Figura 6.1b), cuenta con diversos elementos enfocados a la carga del dispositivo que añaden gran cantidad de cables y ocupan espacio adicional.

(a) Modulo COMMU CAN M5Stack (b) M5GO charger.

Figura 6.1: Stacks necesarios para la implementación del nuevo montaje.

Gracias a estos módulos, el número de cables y el espacio ocupado se ve reducido, al eliminar elementos adicionales y condensarlo todo en un área de 5x5 cm.

6.3.3. Conectores

Se mantienen los conectores RJ45 que portan los cables del CAN bus para transportar los datos de los sensores a la placa de soldadura, en la que se unen los cables de cada sensor. Para transportar los cuatro cables resultantes (dos de datos, dos de alimentación) al M5Stack, se hace uso del cable mostrado en la Figura 6.2, proporcionado por la misma compañía ya que el puerto es de tipo connext [28].

Figura 6.2: Cable Groove M5Stack con puerto Connext.

6.3.4. Baterías

Se mantienen el mismo número y tipo de baterías del modelo original.

6.4. Selección de Sofware

6.4.1. Broker

En el Apartado 5.2.1 se describen los diversos tipos de brokers que pueden emplearse.

Para el desarrollo de este proyecto, lo ideal es hacer uso de un broker privado, ya que el servicio que se pretende proveer requiere de fiabilidad y capacidad de personalización para adecuarse a las necesidades del mismo. Los broker públicos no aseguran ninguna garantía y es mejor que sean destinados a pruebas y prototipados. Además, el coste económico del mismo reside en el coste de la placa Raspberry usada, por lo tanto, no supone ningún tipo de inconveniente.

Entre los diversos softwares a disposición, aunque cualquiera puede funcionar, **se elige MQTT Mosquitto,** al ser el más estandar y ligero de los que se describieron.

6.4.2. Base de datos

En el Apartado 5.2.2 se describen dos tipos de bases de datos. Debido a la estructura de los topics[3], el uso de bases de tipo SQL es ideal, ya que en el propio

[3]Etiquetas donde se encuentra cierto mensaje.

topic es fácil incluir un identificador, facilitando su almacenamiento. Además, la cantidad de datos o los propositos de la aplicación no justifican el uso de bases de datos NOSQL.

Dentro de las opciones de la Tabla 5.5 del Capítulo 5, las mejores opciones son MariaDB frente a MySQL, al ser una versión mejorada y más actual del mismo; e InfluxDB, por sus ventajas a la hora de analizar series temporales. Si bien es cierto que Influx puede parecer una mejor opción para aplicaciones de IoT y análisis en tiempo real, **se opta por usar MariaDB** ya que la monitorización no se realizará en tiempo real (vease Apartado 6.6) por lo que no se necesita esa optimización extra que aporta InfluxDB. No obstante, se plantea estudiar el cambio a este SGDB en trabajos posteriores a este proyecto en el caso de que no se esté satisfecho con el rendimiento de MariaDB.

6.4.3. Gestión de los datos

Una vez seleccionados los softwares que gestionan y almacenan la información, aún queda tomar una decisión: su gestión. Como se van a manejar y como van a fluir desde el dispositivo hasta el servicio final.

Si bien esto puede realizarse por medio de scripts, **Node-Red** permite realizar la gestión de los flujos de datos de forma visual. Se decide hacer uso del mismo debido a:

- *Simplicidad.*
- *Actualidad.* Es un estandar en gran parte de las aplicacioens IoT.
- *Escalabilidad.* Gracias a los flujos, permite escalar el proyecto de forma más sencilla y rápida.
- *Posee UI integrada.*

6.4.4. Capa de aplicación

La interfaz gráfica de Node-Red permite mostrar cualquier tipo de información y abre grandes posibilidades ya que es posible combinar funciones de Javascript con HTML y estilos de CSS para generar interfaces muy vistosas.

Grafana por su parte ya posee una interfaz muy profesional y elegante, únicamente conectando la base de datos con el servicio, por lo que hacer uso de esta última simplifica mucho el proceso y además los datos quedan presentados de forma

más vistosa. Por otro lado, dado que la base de datos se controla desde Node-Red, se decide emplear su UI únicamente para ofrecer a la/las personas encargadas en el hospital una plataforma mediante la cual interactuar con la base de datos sin necesidad de entrar en el gestor, evitando la posibilidad de que puedan modificar parámetros indeseados. En resumen, la capa de aplicación está compuesta por:

- *Grafana,* como plataforma de monitorización de datos.
- Node-Red UI, como interfaz para interactuar con la base de datos.

6.5. Modelos de funcionamiento de la aplicación

Una vez determinado el tipo de arquitectura a emplear y los componentes de Hardware y Software que componen la aplicación final, es necesario establecer el modelo de funcionamiento que va a seguir, es decir, como se van a transmitir los datos desde el andador a la plataforma.

En el Capítulo 4 se describen los diversos tipos de protocolos de comunicación en mayor detalle, pero la Figura 6.3 resume las dos características más importantes a tener en cuenta:

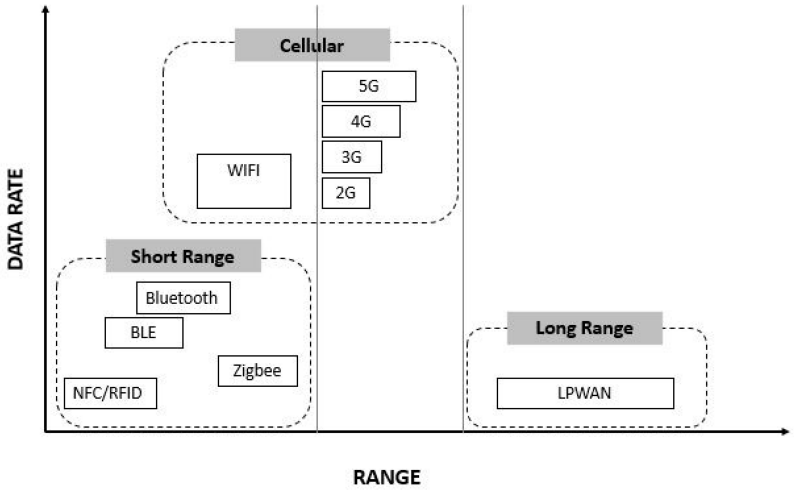


Figura 6.3: Rango vs tasa de envio de datos en protocolos de transporte [4]

En base a lo expuesto, se procede a realizar varias propuestas que la plataforma podría adaptar. Las Figuras 6.4, 6.5, 6.6 y 6.7 ilustran el flujo de datos y composición de cada modelo.

6.5.1. Modelo 1: LPWAN + WiFi

Sin duda, dejando al margen las posibles limitaciones técnicas, que se comentarán más adelante, hacer uso de comunicaciones a largo alcance parece ideal. Estas proporcionan un rango de transmisión de datos suficientemente grande como para que el usuario pueda desplazarse alrededor de la ciudad sin perder conexión, mientras que el dispositivo funciona con WiFi cuando el usuario se encuentre dentro de su vivienda. De esta forma, se puede contar con información en tiempo real prácticamente de forma constante.

No obstante, la tasa de datos que permiten las LPWAN es muy pequeña, tal y como se aprecia en la Figura 6.3, de 100 bits/s en el caso de SigFox y de 50 kbits/s-200 kbits/s para LoRaWAN y Nb-IoT respectivamente, tal y como muestra la Tabla 4.2. Además, todos ellos cuentan con un ancho de banda muy pequeño. Debido a ello, esta solución se hace incompatible con las demandas del proyecto.

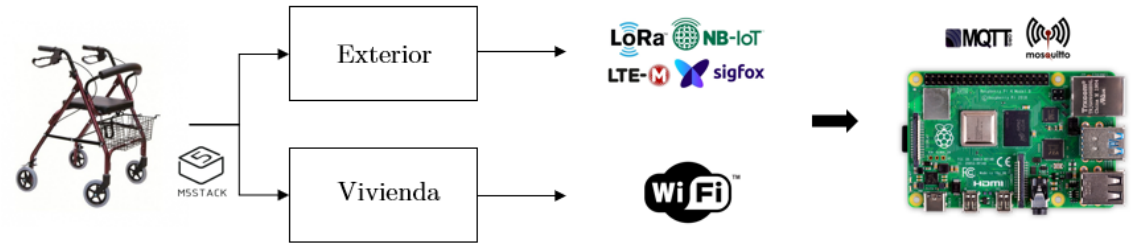


Figura 6.4: Esquema del modelo de implementación 1.

6.5.2. Modelo 2: Celular + WiFi

Teniendo en cuenta que los datos recogidos por el andador cuando el usuario se encuentra en la calle cuentan con un gran valor ya que permite observar las diferencias de los datos respecto a entornos controlados, el uso de celular parece una opción interesante ya que:

- Habilita conexión permanente entre el dispositivo y la plataforma prácticamente en cualquier localización.
- La tasa de datos es suficiente para la demanda del proyecto.
- En la vivienda, el dispositivo transmite por WiFi para evitar pagar por datos transmitidos innecesariamente por celular y ahorrar batería.

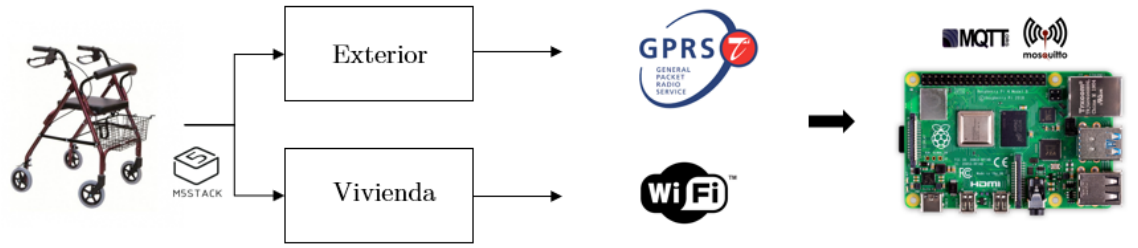


Figura 6.5: Esquema del modelo de implementación 2.

A pesar de sus ventajas, el uso de celular acarrea diversas desventajas que han de ser recalcadas:

- Requiere pagar tarifas para usar las bandas de celular.
- Consume mucha energía, por lo que la autonomía se reduce drásticamente.

6.5.3. Modelo 3: Bluetooth + Telefono móvil + WiFi

Este modelo permite la transmisión de los datos a la plataforma siempre que el usuario cuente de un telefono móvil con conexión a internet, de igual forma que lo hace un reloj deportivo actual

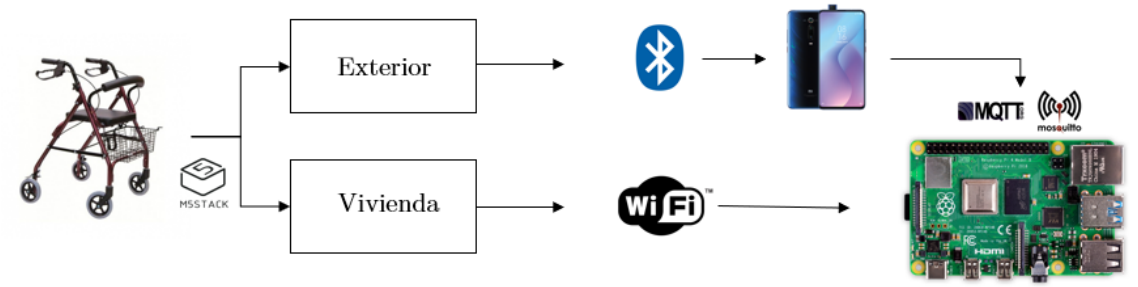


Figura 6.6: Esquema del modelo de implementación 3.

. De esta forma, se cuenta con las ventajas del modelo 2 pero se eliminan los inconvenientes, ya que no hay que pagar tarifa adicional para transmitir los datos y este protocolo de comunicación consume poca energía en el dispositivo, lo que mejora la autonomía del andador. Al igual que en los anteriores, se cambia a transmisión por WiFi una vez el dispositivo se encuentre en la vivienda.

No obstante, el uso de esta solución, a pesar de sus ventajas, añade nuevas desventajas a tener en consideración:

- Aumenta el consumo de batería del dispositivo móvil del usuario.
- Idem con el consumo de la tarifa de datos.
- Sujeto a que el usuario disponga de telefono móvil con Bluetooth 4.0, lo cual puede ser un problema en personas mayores y ancianos.

6.5.4. Modelo 4: Almacenamiento en SD + WiFi

Esta solución almacena los datos captados por los sensores en una tarjeta SD en el interior del microcontrolador cuando el andador se encuentra fuera de casa. Estos quedan almacenandos junto con la hora de registro en formato timestamp. Así, cuando el andador regresa a la vivienda, los datos se suben a la plataforma haciendo uso del WiFi, de forma ordenada de acuerdo a la hora de recogida de los datos.

Este modelo no permite realizar un procesado en tiempo real de los datos pero, sin embargo, cuenta con numerosas ventajas:

- Al no transmitir diariamente de forma constante, la autonomía del dispositivo aumenta considerablemente, contando con un pico de consumo cuando estos datos se descargan.
- Los datos siempre son recogidos independientemente de si en la localización del dispositivo cuenta con conexión a internet.

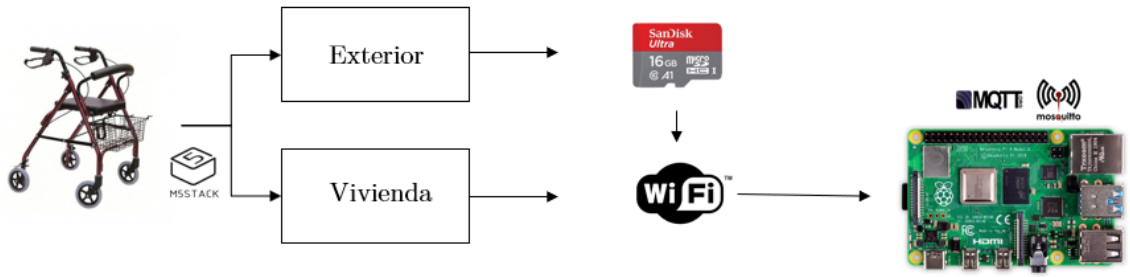


Figura 6.7: Esquema del modelo de implementación 4.

6.6. Justificación del modelo escogido

A excepción del primer modelo, el cual no es posible implementar por el bajo ancho de banda y tasa de envio de datos que posee en relación a las necesidades del

proyecto, todos los modelos son buenas opciones. Cada una cuenta con una serie de pros y contras que la hacen destacar más en unos aspectos y menos en otros. La conexión con celular permite transmisión rápida de datos en cualquier lugar a costa de un coste adicional, mientras que el uso de Bluetooth está sujeto al tipo de usuario que haga uso del andador.

Para seleccionar el modelo se han tenido en cuenta dos criterios: economía y autonomía.

- *Desde el punto de vista económico,* se ha de descartar el modelo 2 por las tarifas adicionales que acarrea, pues si la plataforma pretende funcionar para un número elevado de usuarios, el desembolso económico comienza a aumentar.
- *Desde el punto de vista de la autonomía,* los modelos 3 y 4 suponen un consumo energético aceptable, siendo quizás mejor en el 4 ya que no se transmiten datos mientras el andador se encuentra en el exterior.

Sin embargo, hay un factor clave a la hora de decidir entre el modelo 3 y 4: *El servicio que se pretende proveer no demanda un analisis y procesado de los datos en tiempo real,* pues estos datos cobran valor con un conjunto de los mismos a lo largo de los días y las semanas. Luego, con poder contar con los datos al menos, una vez al día, es más que suficiente.

Por lo tanto, **el modelo 4 se considera el más idóneo** para este proyecto por los motivos que se exponen a continuación:

- Es el más económico.
- No requiere periféricos.
- *Es más amigable para el usuario,* ya que no se debe realizar emparejamiento entre dispositivos ni consume batería / datos de su dispositivo móvil.
- No limita la ubicación del dispositivo para recoger los datos.
- Satisface las demandas del servicio.

Capítulo 7

Implementación de la plataforma IoT

Contenido

7.1. Introducción

Los capítulos anteriores han valorado las alternativas y seleccionado de forma justificada la arquitectura y los componentes de la plataforma. Para afianzar más lo estudiado, o bien para aquel lector que no esté interesado en leer los capítulos anteriores[1], la Tabla 7.1 enumera y resume la función principal de cada elemento:

CAPA	***ELEMENTO***	***CANTIDAD***	***FUNCIÓN***
PERCEPCIÓN	Encoder AS6500	2. Uno en cada rueda.	Mide el ángulo girado por la rueda en un intervalo de tiempo, permitiendo conocer la distancia recorrida o la velocidad.
	Amlificador HX711	2. Uno en cada mango.	Captura las lecturas de las galgas extensiométricas en mV y las amplifica para obtener mayor resolución. Tras cálculo mecánico de un voladizo con una fuerza aplicada a una distancia L, ofrece la fuerza aplicada sobre el mango.
	Servo	2. Uno en cada freno.	Es controlado por el microcontrolador para frenar el andador en caso de peligro.
	Arduino nano V3	6. Uno por cada sensor/actuador	Dos funciones.Recoge la señal de cada sensor, la procesa para que sea una lectura útil y permite conectar el conversor a CAN bus.
	CAN Bus MCP2515	6. Uno por cada Arduino.	Permite transmitir los datos por CAN Bus, permitiendo usar la interfaz SPI del Arduino para ello. Transmisión rápida, fiable y unificada en solo cuatro cables.
	Placa de soldadura	1	Unifica cada cable del CAN bus por tipo: 5V, GND, datos L, datos H. De esta forma tras la placa, solo se requiere un cable para transmitir los datos.
	M5Stack	1	Microcontrolador encargado de recibir los datos, prepararlos para su envio y gestionar su almacenamiento, la conexión con la red y su posterior envio al broker. También es el encargado de gestionar las señales que mandan todos los sensores en modo ahorro de energía, para optimizar batería (no implementado).
RED	Tarjeta SD		Recoge los datos cuando no hay acceso a la red, para su posterior envio.
	WiFi		Protocolo de comunicación seleccionado para la transferencia de datos, cuando el andador se encuentra en la casa. Envio de datos una vez por día.
PROCESADO	Raspberry Pi	1. Situada en el hospital.	Encargada de albergar todos los servicios que conforman el procesado y aplicación de la plataforma.
	Mosquitto MQTT	1. Instalado en la Raspberry Pi.	Broker. Recibe los mensajes de los andadores (publicadores) y los distribuye a Node-Red (suscriptor).
	LAMP	1. Idem.	Infraestructura que habilita la base de datos en la cual se almacenan los datos.
	PHPMyAdmin	1. Idem.	Interfaz gráfica que permite gestionar la base de datos fuera de la consola.
	Node-Red	1. Idem.	Entorno de programación por flujos destinado a recibir los mensajes del broker y almacenarlos en la base de datos.
APLICACIÓN	Node-Red UI	1. Idem.	Interacciona con la base de datos realizando consultas que permiten ofrecer al usuario una interfaz gráfica por la cual esta puede ser gestionada, o bien, descargar los datos.
	Grafana	1. Idem.	Plataforma de monitorización por la cual los datos son representados gráficamente para permitir la monitorización a distancia del andador. Objetivo final de la aplicación.

Tabla 7.1: Resumen de los elementos seleccionados y sus funciones en el proyecto.

[1]Por ejemplo, aquellos que usen la plataforma en el hospital o personas interesadas en continuar este trabajo.

La Figura 7.1 por su parte, resume visualmente los contenidos más importantes de la tabla anterior:

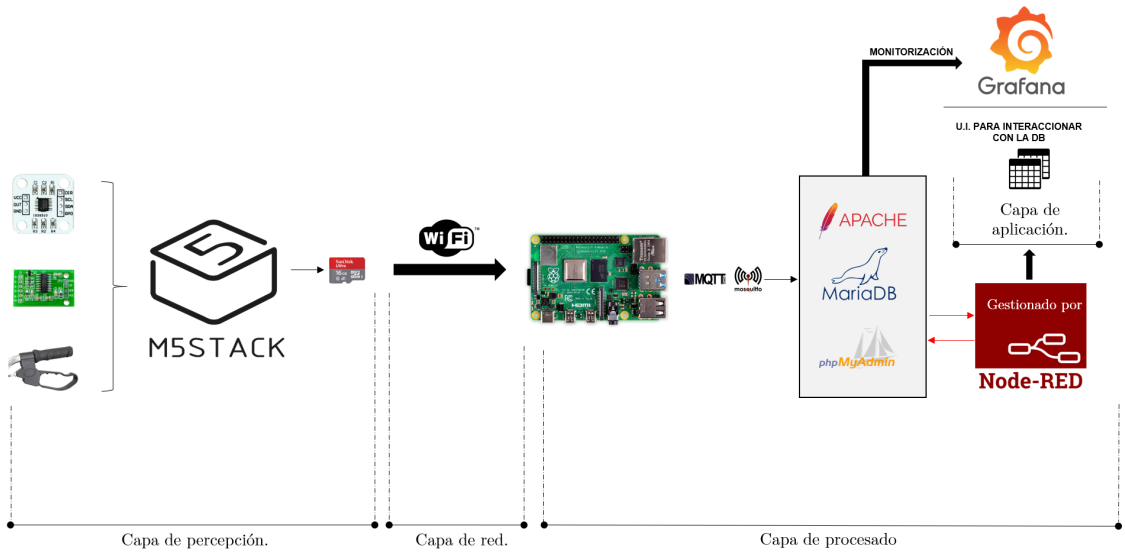


Figura 7.1: Arquitectura y componentes de cada capa de la plataforma.

A continuación se procede a implementar un nuevo montaje físico de los componentes para adaptar la plataforma a los nuevos elementos, y se resume el proceso para crear la plataforma IoT, desde los paquetes empleados hasta los criterios de diseño y finalmente, la ejecución.

7.2. Montaje

La Figura 7.2 muestra un diagrama de flujo del montaje a realizar para cada sensor:

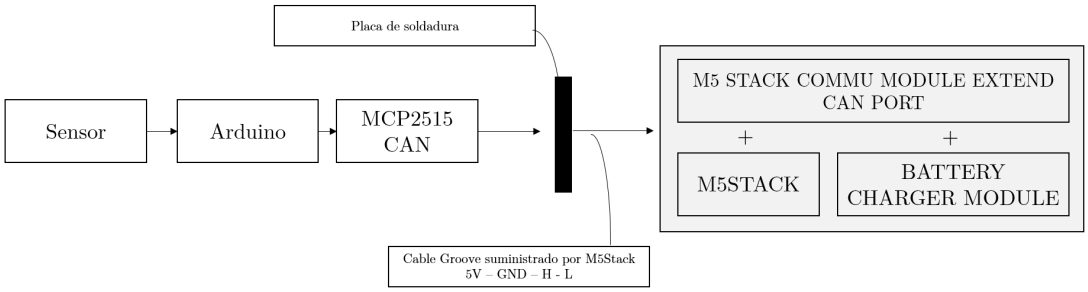


Figura 7.2: Diagrama de flujo del montaje del nuevo sistema.

El nuevo montaje es idéntico al anterior desde los sensores hasta la placa de soldadura. No obstante, una vez unidos los cables de cada sensor en los dos cables de alimentación y los dos de datos resultantes tras la placa, se hace uso del conector groove de 0.5 metros de largo para conectarse al M5Stack por medio del módulo adaptador CAN.

La Figura 7.3 muestra un diagrama completo del nuevo montaje, equivalente al de la Figura 2.8 del Capítulo 2. Se observa que se eliminan gran cantidad de elmentos destinados a la carga de la batería, gracias al modulo de carga. No obstante, es necesario mantener el regulador de tensión para adaptar a 5V la tensión de entrada al M5Stack.

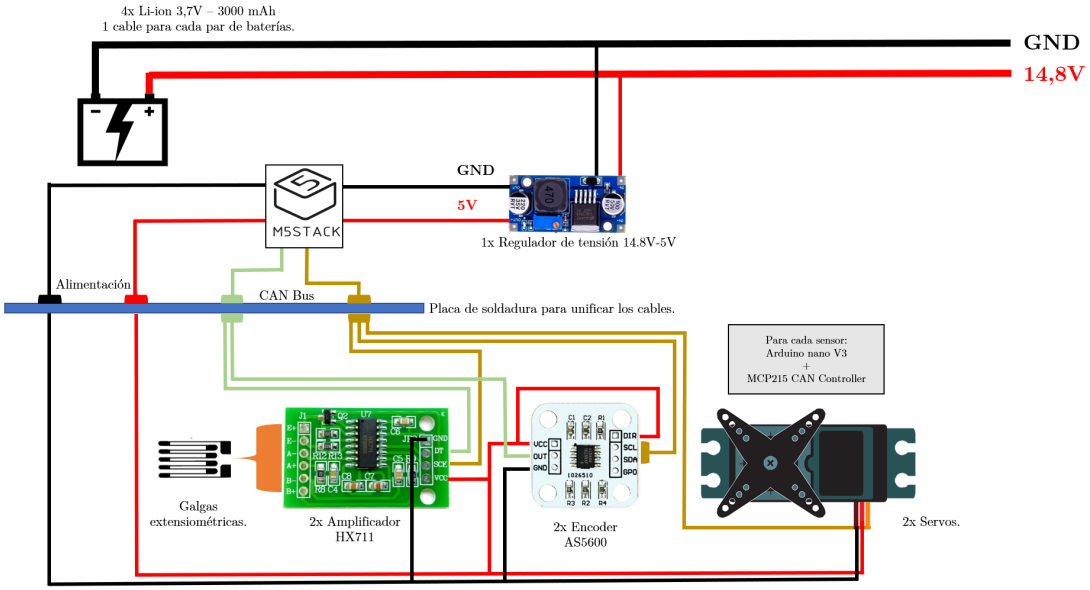


Figura 7.3: Esquema de del nuevo montaje.

Es necesario tener en cuenta una serie de factores asociados a la ubicación del M5Stack en el andador. Se discuten a continuación;

- **Situación exterior.** Ya que solo acceden dos cables para alimentarlo, y el cable groove para conectar los sensores, se puede aprovechar la pantalla LCD para mostrar el estado de la batería y así alertar al usuario de la necesidad de recarga. Además, el propio microcontrolador ya dispone de botón de encendido y el puerto de carga queda más cercano al usuario. Sin embargo, puede que no sea deseable que el usuario tenga proximidad a cables y al propio microcontrolador.

- **Situación interior.** Mejor del lado de la seguridad, al no disponer a la vista cualquier elemento electrico. No obstante, requiere extender el botón de encendido y el puerto de carga a un lugar accesible al usuario.

Se recomienda disponerlo en el interior de la caja, pero se ha descrito la situación exterior como una posibilidad que puede ser estudiada por las personas encargadas del andador en trabajos posteriores a este proyecto, debido a no poder trabajar físicamente con el andador y observar su vialibilidad.

7.3. Puesta a punto

Para instalar los paquetes en la Raspberry Pi, primero es necesario poner a punto la misma. Para ello se instala el sistema operativo *Raspberry Pi OS (32-bit) Lite*[2] flasheandolo mediante el programa *Balena Etcher* [3].

Una vez realizado esto, es necesario establecer un entorno de trabajo. Puede emplearse periféricos para poder controlar la placa como un ordenador, sin embargo, se opta por controlarlo de forma remota, accediendo a través de la red WiFi empleando una conexión SSH[4]. Para ello se siguen los siguientes pasos:

- En la raiz del sistema operativo, crear un archivo sin extensión de nombre "ssh". Esto permite hacer uso del mismo, ya que está desactivado por defecto.
- En la misma raiz, editar el archivo *wpa_supplicant.conf* añadiendo la SSID y la contraseña de la red que se va a usar.
- Buscar dirección IP estática, mediante un analizador de IPs y anotar la del terminal. En este caso, **192.168.1.67**.
- Conectarse a dicha IP a través del cliente Putty[5]

Hay que tener en cuenta que para el producto final hay que realizar una configuración específica según el modelo de router que se disponga en el centro de rehabilitación para que los andadores puedan conectarse desde otra localización. Esta configuración solo permite conexión en la red local.

[2] https://www.raspberrypi.org/downloads/raspberry-pi-os/
[3] https://www.balena.io/etcher/
[4] Secure Shell, permite conexiones remotas seguras haciendo uso de un modelo cliente-servidor.
[5] https://www.putty.org/

(Acceso online el 20 de Agosto de 2020).

Para poder comenzar la instalación de los paquetes de software necesario es necesario actualizar la Raspberry Pi.

Código 7.1: Actualización de los paquetes

```
1 sudo apt-get update
2 sudo apt-get upgrade
```

La instalación de los paquetes puede encontrarse resumida en el Anexo A, y contenida en el repositorio.

7.3.1. Identificación y rutas de acceso

La Tabla 7.2 resume las rutas de acceso, los usuarios y las contraseñas establecidas en cada servicio instalado, para poder acceder a ellas en un momento posterior a este proyecto por parte de los beneficiarios de la plataforma, para poder asegurar los servicios.

SERVICIO	***ACCESO***	***USUARIO***	***CONTRASEÑA***
Raspberry Pi	192.168.1.67	pi	tfg
Mosquitto MQTT	192.168.1.67:1883	pablo	tfg
Node-Red	192.168.1.67:1880	-	-
Maria DB	192.168.1.67:3306	Pablo	tfg
PHPMyAdmin	192.168.1.67/phpmyadmin	Pablo	tfg
Grafana	192.168.1.67:3000	pablotfg	pablotfg

Tabla 7.2: Rutas de acceso a los servicios utilizados, usuarios y contraseñas.

7.4. Topics

La plataforma debe soportar varios andadores, por lo que es necesario un campo llave que sirva de identificador dentro de los topics para poder acceder a los datos de un andador particular.

Además, es interesante conocer el estado del andador: ¿Cúando se conecta?, ¿cúando se apaga?, y ¿cúando sufre una desconexión imprevista?. En el Apartado 4.2.3 del Capítulo 4 se describen los conceptos de ultima voluntad y de mensaje de nacimiento. Se configuran estos campos de forma que cuando el andador se enciende, automáticamente el broker publica un mensaje con valor 1. Cuando se apaga, el broker no detecta al andador y publica un 0. Finalmente, si hay una desconexión

imprevista, por ejemplo un corte de luz en la planta del hospital del broker, o un fallo en el andador cuando se están transmitiendo datos, se publica un 2. De esta forma, es el propio broker el que registra el estado de los andadores y se ahorra batería en el dispositivo al no tener este que estar enviando esta información.

El gran poder de esta configuración, es que debido al modelo de implementación escogido, es posible conocer el escenario en el cual el paciente está haciendo uso del andador. Cuando se desconecta del broker, es por que el paciente esta usandolo en la calle, y cuando está conectado, es por que está en su vivienda. Así se dispone de información adicional al analisis, por que se puede comparar el comportamiento del paciente en dos tipos de escenarios distintos y observar si existen diferencias.

El flujo de los topics se representa en la Figura 7.4:

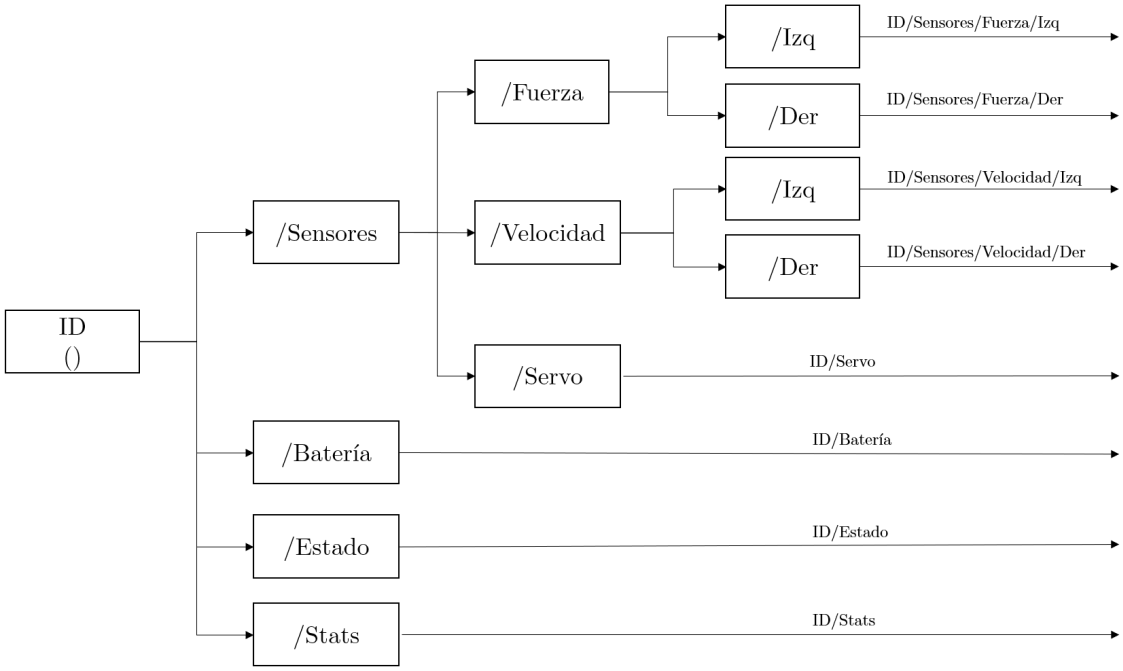


Figura 7.4: Estructura de los topics de la plataforma.

Los datos de relevancia y la frecuencia con la que deben ser enviados se muestran en la Tabla 7.3 y quedan estructurados de acuerdo a la estructura mostrada en la figura anterior.

DATOS DE INTERÉS	*FRECUENCIA*
ID del andador	-
Lectura de sensores de fuerza	50-100Hz
Lectura de los encoders	50-100Hz
Estado del servo	Suscrito a la plataforma. Solo un mensaje cada vez que se cambie de estado.
Fecha y hora con precisión de milisegundos	50-100Hz. Tiene que acompañar la frecuencia de las lecturas.
Estado de la batería: Voltaje y porcentaje	1 mensaje por minuto.
Distancia recorrida en un día.	1 mensaje por día, a una hora determinada al final del día.
Valor medio de la asistencia del andador en cada mango	Se calcula desde la base de datos 1 vez al día. No requiere un topic.

Tabla 7.3: Datos de interés y la frecuencia de publicación de sus topics.

7.5. Tablas

La estructura de las tablas de la base de datos, los campos, su longitud y las relaciones entre ellas se pueden observar en la Figura 7.5.

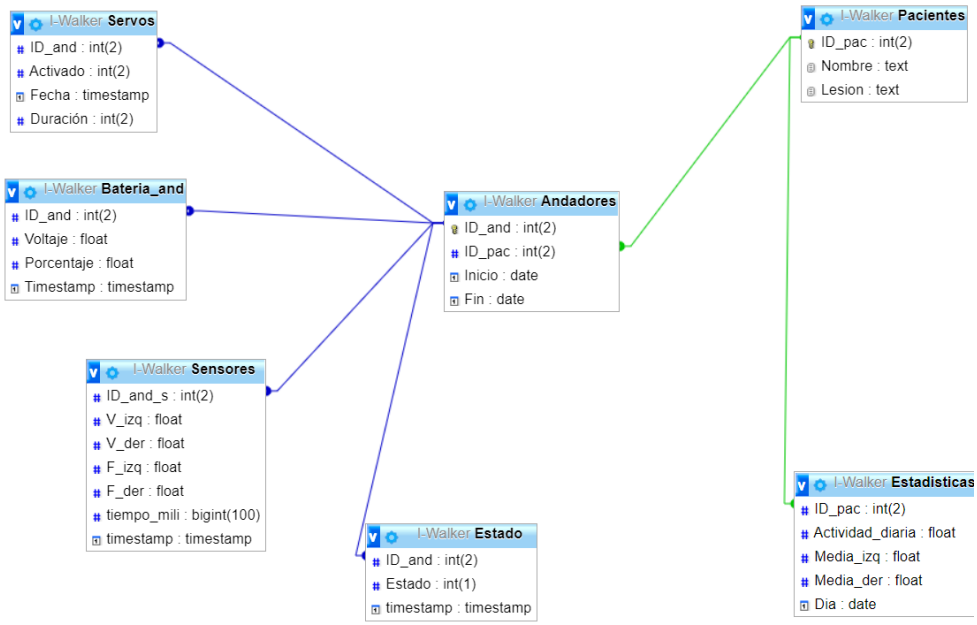


Figura 7.5: Tablas, campos y relaciones que componen la base de datos.

Se procede a comentar los aspectos más destacados de la estructura de la base de datos:

- **Tabla pacientes:** Esta tabla registra la información personal de cada paciente. *IDpac* es la llave primaria de esta tabla, y es única para cada paciente, por lo que se autoincrementa cuando se añade uno nuevo. Por el momento se ha registrado solo nombre y lesión, pero puede añadirse cualquier campo que se estime oportuno por parte del hospital.
- **Tabla sensores:** En esta tabla se recogen los valores captados por los sensores. Almacena la fecha y hora de registro, y el tiempo en milisegundos para permitir el analisis de los datos.
- **Tabla principal: andadores.** Esta tabla ejerce de registro temporal de los pacientes que han usado cada andador y sirve de nexo de unión entre las dos anteriores, permitiendo acceder a datos de los sensores realizando consultas a través de datos del paciente. *IDand* es la llave primaria de esta tabla, pero no se autoincrementa, por que el número de andadores es limitado, y varios pacientes pueden usar el mismo andador, siempre y cuando la última fecha de fin de uso de un andador sea más antigua que la de inicio del siguiente paciente que lo utiliza.
- **Tabla servos:** Esta tabla registra cuando se activa el servo y por cuanto tiempo. En cualquier otro caso, no registra nada.
- **Relaciones:** La tabla andadores se relaciona a través de la llave primaria *IDand* con toda la información relativa al mismo. cada tabla cuenta con una llave foranea del mismo nombre. Debido a que la tabla sensores se actualiza a una frecuencia muy alta, la información relativa al estado, los servos y la batería se debe crear en tablas separadas para que se actualicen a su frecuencia, y no haya campos nulos. *IDpac* de la tabla Andadores es una clave foránea, por que es más lógico que la primaria se encuentre en el registro de pacientes. Finalmente las estadisticas se asocian con el paciente en lugar del andador ya que es información relativa a la evolución del paciente.

7.6. Programación de los Arduinos

Los datos alcanzan el arduino en primer lugar,donde son preprocesados en y posteriormente, son convertidos a CAN bus para ser enviados al M5Stack. Se emplea Arduino IDE como entorno de programación y por tanto, C++ como lenguaje.

El objetivo de este apartado no es explicar en detalle todo el código, si no mostrar los aspectos más relevantes del mismo para plasmar una visión general del transporte y procesado de los datos.

7.6.1. Librerias

Las librerias empleadas para todos los arduinos existentes en el andador:

- **Mcp_can.h.** Permite controlar el conversor CAN a través del Arduino, establecer la velocidad de envio de los datos y la frecuencia del reloj, así como asignar una dirección a los datos, de forma que aun viajando por el mismo cable puedan ser recibidos e identificados por el M5Stack.
- **SPI.h.** Los sensores se conectan las conversor anterior a través de los pines SPI, por lo que hay que declararlo en el código y usar dichos pines en la conexión.

Cada par de arduinos posee una de las siguientes librerias para poder leer y tratar los datos de cada sensor:

- **HX711.h.** Permite adquirir las lecturas de fuerza y calibrar el cero.
- **AS5600.h.** Permite detectar el iman y adquirir las lecturas angulares bajo las directrices y opciones que permite la libreria.
- **Servo.h.** Permite controlar diferentes modelos de servos con Arduino.

7.6.2. Aspectos destacados de los códigos

Debido a que la captación mantiene los mismos elementos que en la versión original, los códigos de los arduinos se mantienen. Estos pueden ser encontrados en el repositorio suministrado en el Anexo B.

Encoder

Se deben calibrar los sensores en la inicialización para establecer el valor de 0º en la posición actual, posteriormente, se leen los valores de los ángulos. El encoder puede detectar 4096 posiciones a lo largo de la circunferencia.

El ángulo leido se debe convertir a grados.

Código 7.2: Leer ángulo

```
float readAS5601() {
  pos = Sensor.getAngle();
  return ( 360 * ( float ) pos ) / 4095;
}
```

Por otro lado, cada vez que se inicializa el andador, una variable denominada vueltas se activa en valor nulo. La función Código 7.3 es llamada en el bucle. La función anterior registra la posición actual. Si esta estaba a 3/4 de vuelta y en la nueva lectura ha superado el 0, entonces se añade una vuelta, en caso contrario, se resta. Se registra la lectura de posición de esta iteración en una variable que la almacena como posición antigua, para permitir la comparación.

Código 7.3: Comprobar distancia recorrida.

```
void checkTurn() {
  if ( oldPos > 3071 && pos < 1023 ) {
    turn.value ++ ;
  }
  else if ( oldPos < 1023 && pos > 3071 ) {
    turn.value -- ;
  }
  oldPos = pos;
}
```

Envio de datos

Los datos son enviados por CAN bus, por el mismo cable, por lo que se les debe proporcionar un identificador, tal y como muestra la Tabla 7.4. Estos son codificados en hexadecimal. El paquete enviado por cada sensor posee la siguiente información:

- ID hexadecimal del paquete.
- Información sobre la estructura de bits: 0 o 1 según si es estandar o extendida, respectivamente.
- Longitud del array.
- Contenido del array.

SENSOR	***POSICIÓN***	***ID***
Encoder	Derecha	0x012
	Izquierda	0x022
Célula de carga	Derecha	0x011
	Izquierda	0x021
Servo	Derecha	0x013
	Izquierda	0x023

Tabla 7.4: Identificación de los datos enviados por el CAN bus.

7.7. Programación del M5Stack

El proposito de esta sección, al igual que en el apartado anterior, es mostrar los aspectos más relevantes del código.

Los datos leidos por los sensores acceden al M5Stack a través del adaptador a CAN bus del propio fabricante. Es necesario procesarlos para que adopten su forma adecuada, gestionar su almacenamiento en la tarjeta SD y su posterior extracción, enviandolos en paquetes en lugar de en datos individuales. Por otro lado, se debe gestionar la conexión a la red WiFi y los mensajes MQTT. Se hace uso del mismo entorno de programación y lenguaje que en el apartado anterior.

7.7.1. Librerias

El M5Stack no requiere las librerias de los sensores, ya que estos acceden sencillamente como datos con una dirección determinada a través del CAN bus.

M5Stack.h

En lugar de declarar multiples librerias, es más recomendable inicializar esta libreria propia, ya que habilita todas las librerias asociadas al ESP-32 que permiten el funcionamiento de todos los componentes adicionales que posee el módulo (que de otra forma deberían ser incluidos aparte). Se observa que al inicializarla, se habilitan las librerias necesarias para el almacenamiento en la tarjeta de memoria.

Código 7.4: Librerías incluidas en el paquete M5Stack.

```
1       #include <Arduino.h>
2       #include <Wire.h>
3       #include <SPI.h>
4       #include "FS.h"
5       #include "SD.h"
6
7       #include "M5Display.h"
8       #include "utility/Config.h"
9       #include "utility/Button.h"
10      #include "utility/Speaker.h"
11      #include "utility/Power.h"
12      #include "utility/CommUtil.h"
```

No obstante, se desea que se almacene la hora exacta de la medición, para ello se hace uso de las siguientes librerias:

- **NTPClient.h [40].** Habilita la conexión a un servidor Network Time Protocol (NTP) para adquirir el momento en el que se almacena un dato. Puede obtenerse en cualquier formato.
- **WifiUDP.h [41].** Permite recibir los paquetes UDP de tiempo anteriores.

El resto de librerías empleadas para el correcto funcionamiento de esta capa y su función se describen a continuación:

- **freertos/FreeRTOS.h [42].** Arduino tiene dos partes principales: void.setup(), que se ejecuta una vez y void.loop(). Con esta libreria se permiten crear funciones fuera de estas dos y siempre que no haya tareas pendientes, se ejecuta el bucle, si no, se ejecutan las sentencias de fuera. De esta forma se ahorra batería y gestionar mejor los eventos.
- **freertos/timers.h.**[6] Permite acceder a temporizadores que gestionan la comprobación de las tareas pendientes.
- **WiFI.h [43].** Permite al microcontrolador acceder a la red WiFi si la autentificación SSID y la PSK son correctamente configuradas, así como gestionar su conexión.

[6] https://www.freertos.org/FreeRTOS-Software-Timer-API-Functions.html

(Acceso online el 24 de Agosto de 2020).

- **AsyncMqttClient.h** [44]**.** Habilita el microcontrolador como un cliente MQTT y proporciona las funciones necesarias para gestionar las conexiones al broker y la publicación de mensajes.
- **ArduinoJSON.h** [45]**.** Permite generar un objeto que contenga las lecturas de los sensores en formato JSON, de forma que se pueden enviar a la plataforma todos en un solo mensaje.

7.7.2. Recepción y preparación de los datos

Una vez inicializadas las librerias, en primer lugar se deben recoger los datos y almacenarlos en variables que facilitan el procesado. Para ello se filtra por ID, de la forma que muestra el Código 7.5.

Código 7.5: Lectura de los datos del CAN bus.

```
1  CAN.readMsgBuf(&len, buf);
2
3      if (CAN.getCanId() == 0x012){
4          vder = (buf[3]/periodo)*R;
5          vueltader =
6         }
7      if (CAN.getCanId() == 0x022){
8          vizq = (buf[3]/periodo)*R;
9          vueltaizq = buf[5];
10        }
11     if (CAN.getCanId() == 0x011){
12         fder = buf[3];
13        }
14     if (CAN.getCanId() == 0x021){
15         fizq = buf[3];
16        }
17
18 vuelta = vueltader;
```

Se conoce que la lectura de ángulo girado mostrado en el Código 7.2 es entre la lectura anterior y la actual, por lo que el tiempo es 0.01s si la frecuencia asignada es 100Hz. Haciendo uso de la Ecuación 2.1 a la hora de guardar los datos, se obtiene la velocidad de cada rueda.

Por otro lado, el array de llegada de los encoders envia las vueltas giradas. Dentro del bucle existe una variable llamada "Vueltas". Esta se actualiza como se muestra la última línea del Código 7.5.

Una vez almacenados los datos en variables, debe recogerse la fecha, hora de recogida de dichos datos. Si la frecuencia de toma de datos es de 100 Hz, proba-

blemente el proceso de asignación de datos, y el de busqueda de la hora provoque un retardo que puede ser significativo frente a 0.01s, sin embargo, lo importante es que los datos estén secuencialmente ordenados en intervalos de 0.01s. El retardo se asume aproximadamente constante, por lo que no afecta a la calidad de las lecturas.

Tras las inicializaciones pertinentes, para obtener la fecha y hora:

Código 7.6: Obtención de la fecha y hora.

```
1 timeClient.update();
2
3  timestamp = timeClient.getFormattedTime();
```

Para obtener los milisegundos exactos en el que los datos son capturados, se guarda la siguiente variable justo antes de guardar los datos en variables.

Código 7.7: Obtención del tiempo transcurrido en milisegundos.

```
1 startMillis = millis(); // Tiempo de referencia.
2
3 mili = millis()
```

Una vez obtenida, se crea un objeto JSON, que contendrá los campos de la Figura 7.6.

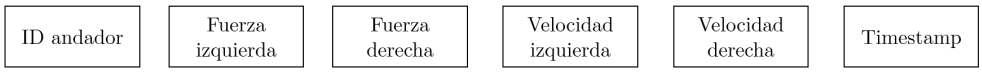


Figura 7.6: Contenido del objeto JSON.

Para ello, se hace uso de la librería JSON. Se crea un objeto llamado lectura y se establece que el tamaño del archivo sea dinámico, para que se adapte a los datos entrantes, y posteriormente se añaden los datos guardados en las variables anteriores, siguiendo la estructura de la Figura 7.4. El formato resultante es:

Código 7.8: Formato JSON expandido del objeto de las lecturas de los sensores.

```
1 lectura = {
2              "ID" : 1,
3              "Fuerza" : [15.23,17.72],
4              "Velocidad" : [5,4.8],
5              "timestamp" : "19-08-2020T16:00:13Z",
6              "milis" : 12563452
7            }
```

Para ello, es necesario hacer uso de las líneas de código empleadas en el Código 7.9.

Código 7.9: Creación del objeto JSON.

```
1  DynamicJsonDocument  lectura(200);
2
3  lectura["ID"] = ID;    //Asigando a cada andador
4  JsonArray data = lectura.createNestedArray("Fuerza")
5       Fuerza.add(fizq);
6       Fuerza.add(fder);
7  JsonArray data = lectura.createNestedArray("Velocidad")
8       Fuerza.add(vizq);
9       Fuerza.add(vder);
10 lectura["timestamp"] = timestamp;
11 lectura["tiempo_mili"] = mili;
12
13 serializeJson(lectura, Serial);
```

La última línea del Código 7.9 estructura el mensaje mostrado en el Código 7.8 en una sola línea, haciendo que el fichero sea más compacto.

7.7.3. Almacenamiento de datos

Si el dispositivo no encuentra red WiFi esto implica que el usuario no se encuentra cerca de su red. Por tanto, los datos leidos son guardados en la SD. Tras inicializar la SD por medio de las funciones de la librería y abrir el fichero, cada vez que se genera el archivo JSON, este se escribe en una nueva linea, como se muestra en el Código 7.11.

Código 7.10: Incluir lecturas en la SD.

```
1  appendFile(SD, "/sensores.txt", lectura.c_str());
```

Como los objetos almacenados poseen información relativa al tiempo de captación y estos son escritos de forma secuencial, no hay que preocuparse de que estos se almacenen erroneamente en la base de datos.

7.7.4. Subida de datos a la plataforma

Cuando la red WiFi es detectada, el microcontrolador se suscribe al topic MQTT.

Código 7.11: Conexión del cliente al broker.

```
1
2  void onMqttConnect(bool sessionPresent) {
3    Serial.println("Connected to MQTT.");
4    Serial.print("Session present: ");
5    Serial.println(sessionPresent);
6
7    uint16_t packetIdSub = mqttClient.subscribe("ID/Sensores", 2);
8    Serial.print("Subscribing at QoS 0, packetId: ");
9    Serial.println(packetIdSub);
10   }
```

El fichero donde los datos son almacenados es abierto y su contenido es enviado. Cuando se recibe confirmación de que el contenido. Cuando el proceso de completa, los datos son eliminados.

Código 7.12: Envio del mensaje.

```
1   mqttClient.publish("ID/Sensores", 2, true, "/sensores.txt");
```

El motivo de crear un archivo JSON es que todos los datos son enviados en un solo topic, y el resto de etiquetas que las distinguen van incluidas en el nombre de cada campo del objeto, de forma que simplifica la gestión de los datos.

7.8. Gestión de los datos en Node-Red

Node Red funciona de forma que la salida de los nodos conforma un objeto denominado *msg*. Este objeto posee una serie de propiedades:

- **Payload:** El contenido del mensaje. Puede ser un número, una cadena de caracteres..
- **Topic:** La parte del objeto en la que el valor del payload esta contenido. Para acceder a dicho payload basta con escribir msg.payload.[nombre del topic].
- **Options:** Contiene las propiedades del objeto. No suele ser necesario configurarlo.

7.8.1. Nodos empleados

Para facilitar la comprensión de los apartados posteriores, se realiza una pequeña revisión del funcionamiento de los nodos empleados.

Los nodos representados en la Figura 7.7 son los nodos principales que permiten recoger los datos, procesarlos en un formato adecuado y almacenarlos en la base de datos.

Figura 7.7: Nodos principales del flujo.

- **MQTT.** Es el encargado de recoger los datos. Para ello, es necesario configurar en que IP y puerto debe estar escuchando el nodo. Además, debe configurarse de que topic recibe los datos y sus características de calidad del servicio.
- **Función.** Permite interaccionar con el mensaje de entrada y devolver el resultado en el de salida. El lenguaje empleado es Javascript.
- **Base de datos.** Conecta la entrada y la salida a la base de datos seleccionada. Para ello es necesario incluir la IP, los puertos y las credenciales de acceso. Combinado con las funciones se pueden realizar consultas o bien introducir datos.

Por otro lado, la Figura 7.8 muestra los nodos necesarios para gestionar la dirección de la información a lo largo del flujo.

Figura 7.8: Nodos secundarios del flujo.

- **Join.** Permite aunar en un único objeto, mensajes procedentes de diversos sitios. Es necesario cuando a una función entran más de dos mensajes, ya que ambos objetos poseen el mismo nombre *msg* y no es psoible discernir su procedencia, dando lugar a errores.
- **Switch.** Ejerce de nodo condicional. Dependiendo del resultado de la comparación, deriva el mensaje en una dirección u otra.
- **Set flow.** Cualquier mensaje que acceda a este nodo, es guardado como variable global, de forma que puede ser recuperado desde cualquier nodo sin necesidad de que esten conectados, mediante las función *flow.get(nombre)*.

7.8.2. Flujos de almacenamiento de datos

Gracias a la preparación de los datos en el microcontrolador, la gestión del flujo de datos en Node-Red es bastante sencilla.

Originalmente, la configuración para recibir los datos de los sensores era como se muestra en la Figura 7.9.

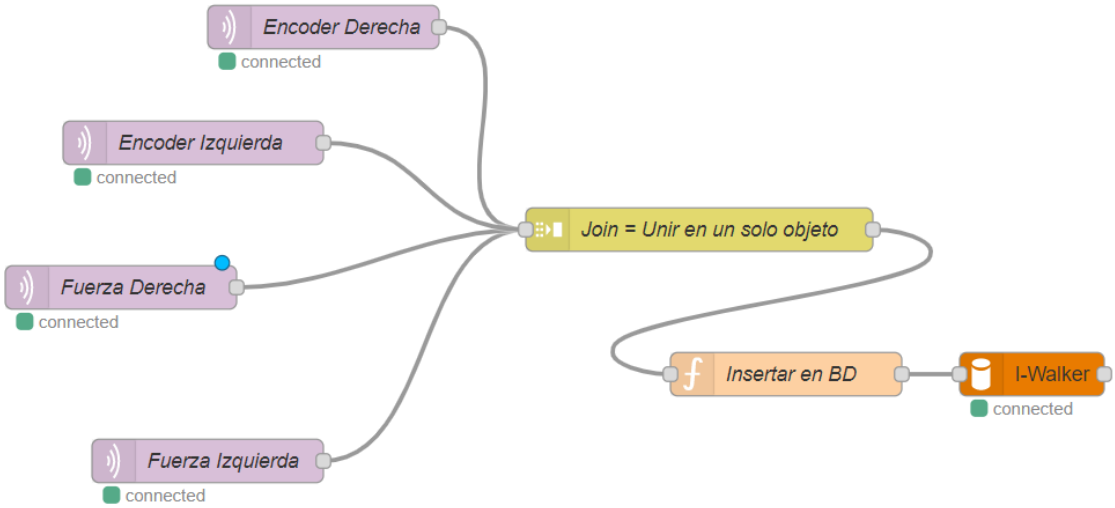


Figura 7.9: Primera versión del flujo de captación de datos de los sensores.

Sin embargo, es más eficiente recibir los datos de los sensores en un solo nodo, como un objeto JSON, permitiendo acceder ellos en el nodo de función fácilmente, eliminando la necesidad de crear un objeto posteriormente con el nodo join.

El nodo MQTT de los sensores debe recibir todos los datos recogidos por la SD solo una vez, para no falsear el analisis, por lo que el QoS se configura en 2. Por otro lado, las lecturas de voltaje no son tan relevantes y si se pierde alguna lectura, no supone un problema, por lo que el QoS se configura en 0. En cuanto a las estadísticas, es necesario recibir al menos una vez el mensaje, pero la redundancia de datos no supone un problema en caso de fallos, por que se representa solo la estadística diaria por lo que se estaría representando siempre el mismo valor. Idem para el estado. Lo expuesto se resume en la Tabla 7.5.

En cuanto al flujo encargado de gestionar las estadísticas, es necesario realizar dos tareas:

- Recibir el mensaje de distancia diaria recogida a través del nodo MQTT.

- Hacer una consulta a la base de datos de las lecturas de los sensores en las últimas 24h y añadir una función agregada para calcular su media.

TOPIC	*QoS*	*CONTENIDO*
Sensores	2	Objeto JSON con todos los sensores.
Voltaje	0	Un número float.
Estadísticas	1	Un solo número float.
Estado	1	Un número de tipo entero.

Tabla 7.5: Configuración de nodos MQTT.

El flujo de nodos resultante se muestra en la Figura 7.10.

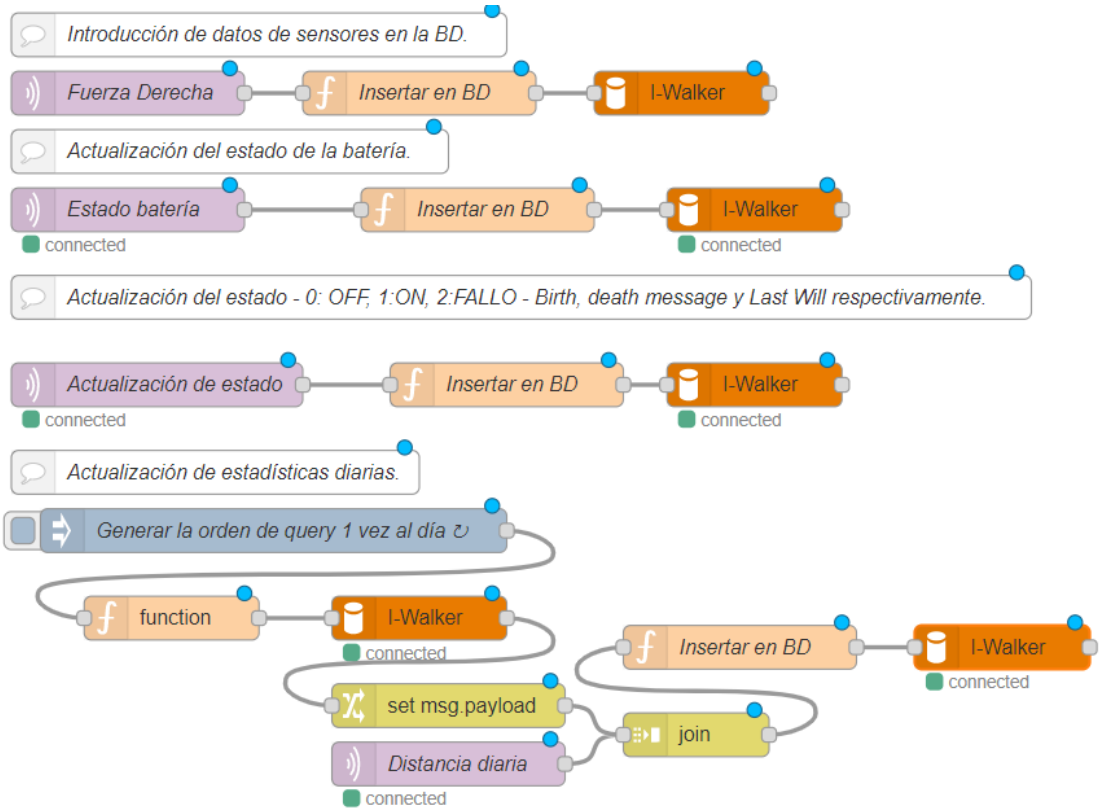


Figura 7.10: Versión final del flujo de captación.

Por tanto, se aplica un nudo Join para generar un objeto único y posteriormente, se añade el producto resultante a la base de datos. El dato procedente del nodo MQTT es recibido pero hasta que no alcance el nodo la media de los sensores, no

puede acceder a la siguiente parte del flujo. Se añade un nodo de inyección que ordena realizar dicha consulta una vez al día a las 23:59 y es a esa hora a la que se añadirán los datos.

7.9. Plataforma de gestión de la BD

En primer lugar se describen las opciones que brinda la interfaz al usuario para generar un visión clara del resultado final. Posteriormente se procede a explicar el proceso seguido para crear la interfaz.

7.9.1. Posibilidades de la interfaz

La interfaz permite a los encargados del hospital la interacción dcon la tabla de los pacientes y los andadores, ya que son tuplas dinámicas. Se resumen las posibilidades que brinda la interfaz al usuario:

- Añadir nuevos pacientes al sistema.
- Asignar un andador disponible dicho paciente.
- Establecer la fecha en la que un paciente que está usando un andador, finaliza su uso. Cuando este empieza, no se puede conocer esta fecha y el valor de este campo es nulo.
- Cambiar algún dato relativo al paciente. Pueden haberse cometido errores al añadirlo.
- Descargar los datos de los sensores relativos a un paciente en un intervalo de tiempo determinado.

Las opciones de interacción entre el usuario y la base de datos son ilimitadas, pero se considera que estas son las necesidades básicas de las que se debe proveer a los usuarios de la interfaz para poder trabajar con ella adecuadamente. En trabajos posteriores, tras un periodo de uso de la plataforma, se pueden crear nuevas opciones basadas en las carencias que hayan encontrado los usuarios de la interfaz.

La Figura 7.11 muestra el conjunto de nodos y flujos que han sido necesarios para generar la interfaz gráfica. Como se puede comprobar, la extensión es muy amplia y por ello, se procede a analizar cada módulo en apartados separados para una mejor comprensión y organización.

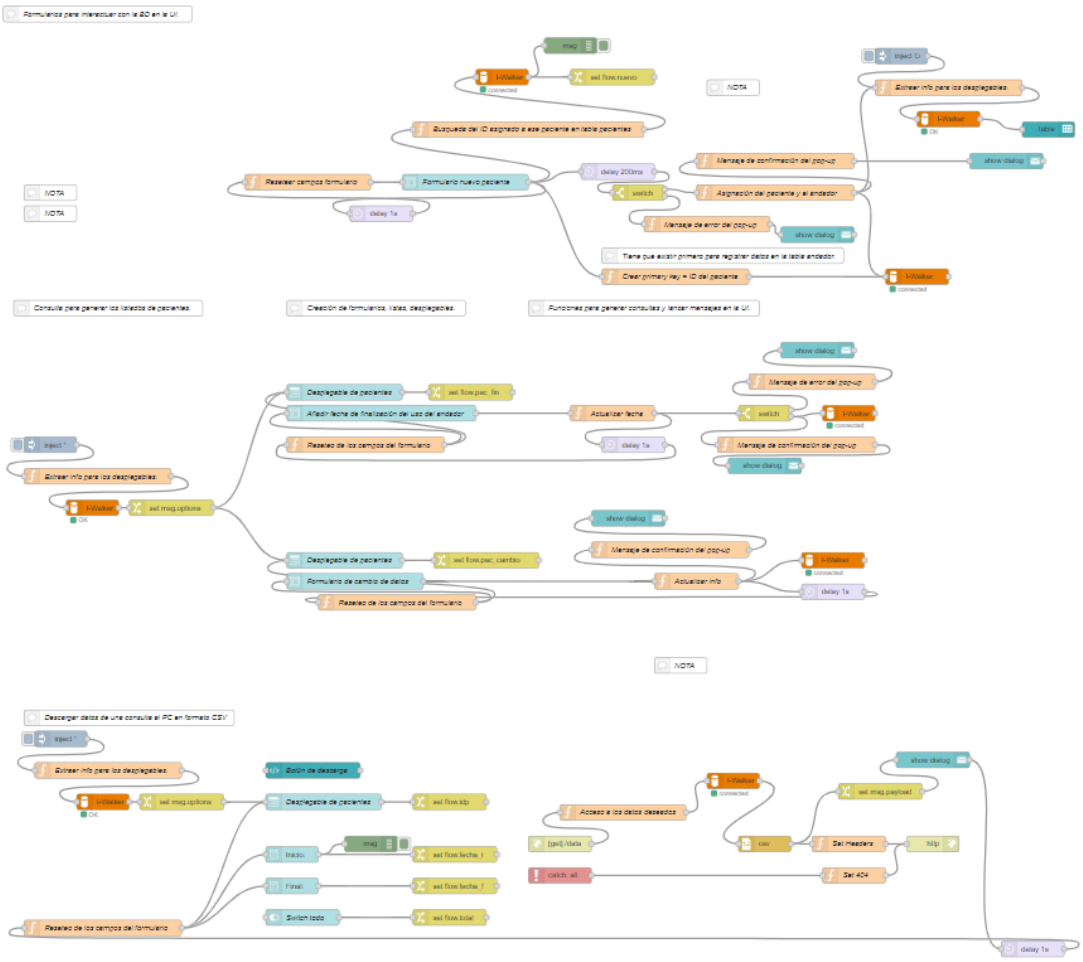


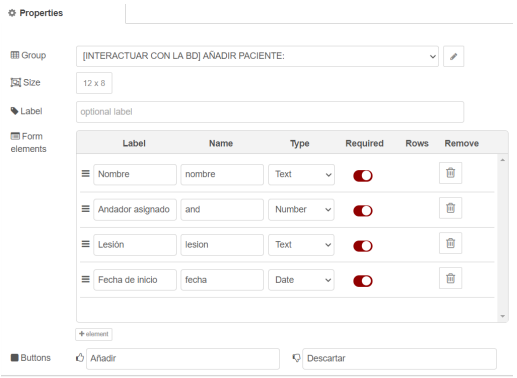
Figura 7.11: Flujo de nodos completo para creación de interfaz gráfica.

7.9.2. Nodos utilizados

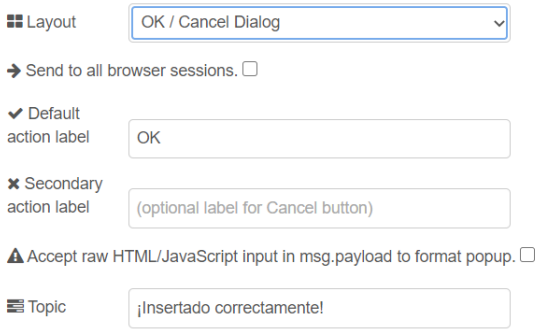
Para facilitar la comprensión de los apartados posteriores, la Figura 7.12 realiza una pequeña revisión del funcionamiento de los nodos empleados especificamente para la interfaz, tal y como se realizó en el Apartado 7.8.1.

Figura 7.12: Nodos usados para la interfaz gráfica.

- **Show dialog.** Es empleado para crear notificaciones emergentes cuando cierto evento ocurre. En este flujo se han empleado para mostrar dialogos de confirmación o de error, para avisar al usuario si está cometiendo errores. El contenido del topic será mostrado en negrita y el payload que accede a el, se mostrará sin negrita (Figura 7.13b).
- **Formulario.** Permite crear un formulario en el que el usuario debe rellenar campos. Puede seleccionarse si son obligatorios o no. Devuelve el valor de las selecciones al presionar el botón de confirmación. La etiqueta será el nombre que verá el usuario, sin embargo su nombre es el que se debe usar para programar las funciones (Figura 7.13a).



(a) Configuración de un formulario.



(b) Configuración de la ventana emergente.

Figura 7.13: Detalles de configuración de nodos de la interfaz gráfica.

- **Calendario.** Es empleado para que el usuario pueda seleccionar fechas. Su salida será la fecha seleccionada.
- **Lista/Tabla.** Muestra una lista de parámetros que recibe por su entrada o bien permite mostrar información estática configurada en su propio menu de propiedades.
- **Desplegable.** Ofrece al usuario seleccionar un valor de los mostrados. Puede configurar los campos en el propio nodo, o ser introducidos en la entrada, para que sea un deplegable que se actualiza con la base de datos.
- **Switch todo.** Botón On/Off.

7.9.3. Módulo 1: Añadir nuevos pacientes

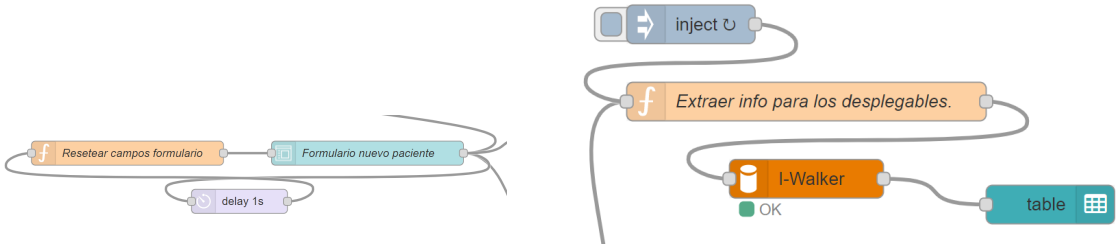
Para añadir un nuevo paciente se deben rellenar los datos clave de la tabla pacientes y asignarle un andador disponible. En primer lugar se deben pedir los datos del paciente y mostrar al usuario que andadores puede asignarle, a través de la interfaz de la Figura 7.17. Para ello se hace uso de dos subflujos.

El subflujo 7.14a esta formado por un formulario en el cual se piden los datos del paciente. Todos los campos son obligatorios. Posee un botón para aceptar los datos y otro para descartarlos, borrando por el formulario. Al pulsar el botón de aceptar, se manda un mensaje de salida. El mensaje continua por dos salidas: la primera continua el flujo, la segunda accede tras un retraso de 1 segundo, a una función que resetea los campos del formulario, usando el código:

Código 7.13: Reseteo de campos del formulario.

```
1 msg={};
2 msg.payload={};
3 msg.payload.nombre='';
4 msg.payload.and='';
5 msg.payload.lesion='';
6 msg.payload.fecha='';
7
8 return msg;
```

El subflujo 7.14b por su parte realiza una consulta a la base de datos para obtener el ID de los andadores cuya fecha de finalización no posee valor *null*. Los resultados son mostrados en la tabla que se observa en la Figura 7.17.



(a) Subflujo de formulario y reseteo (b) Subflujo de andadores disponibles.

Figura 7.14: Sensor y medición indirecta.

La tabla andadores asigna los ID de los andadores y los pacientes. Pero no se puede asignar un andador con un paciente que no ha sido creado todavía ya que se debe tener en cuenta que *IDpac* es un campo autoincremental, y llave primaria de la tabla pacientes. Para solventar este problema, el mensaje toma tres caminos.

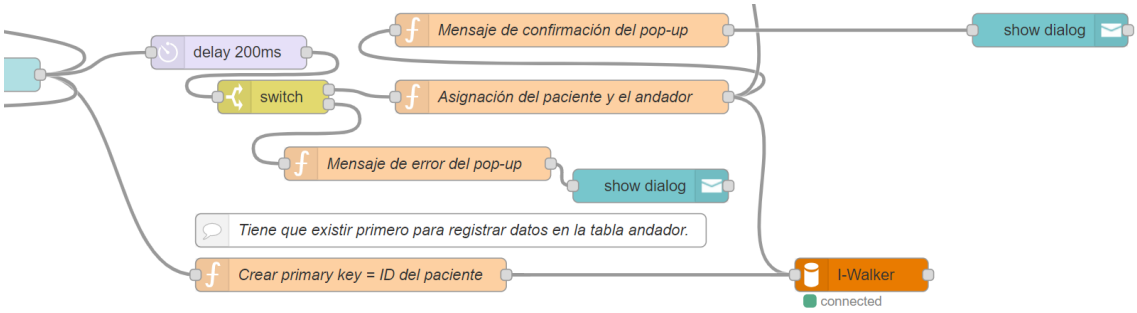


Figura 7.15: Inserción del nuevo paciente.

- El primero crea el paciente con los datos del formulario (Figura 7.15).
- El segundo consulta que ID ha sido asignado a dicho paciente una vez añadido y lo guarda en una variable global (Figura 7.16).
- El tercero, tras un retraso de 200 ms para asegurar que los dos caminos han sido realizados correctamente, genera la asignación paciente-andador en la tabla Andadores (Figura 7.15).

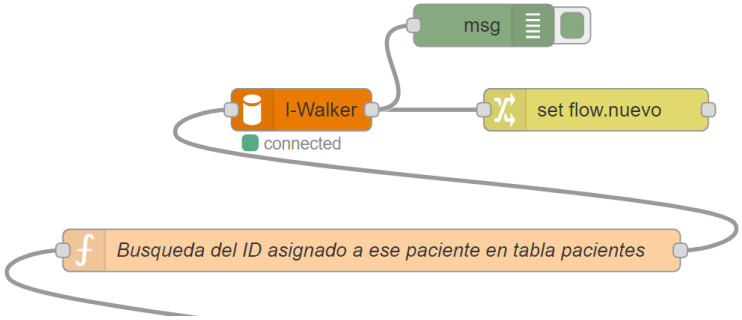


Figura 7.16: Busqueda del ID generado automaticamente para inserción en la tabla de Andadores.

La tabla andadores solo requiere la fecha y no la hora, ya que es configurada como un campo de tipo date. La fecha seleccionada por defecto incluye la hora, así que es necesario formatearla. Se representa la función necesaria para ello y la sentencia de insercción a la base de datos. El proceso en el resto de módulos es idéntico, pero cambiando el nombre de las tablas y campos:

Código 7.14: Formmateo de la fecha introducida para su correcta inserción en la base de datos.

```
var id_and = msg.payload.and;

const dt = new Date(msg.payload.fecha);
const dd = dt.getDay();
const mm = dt.getMonth() + 1;
const yyyy = dt.getUTCFullYear();

var inicio = `${yyyy}-${mm}-${dd}` ;

msg.topic = "INSERT INTO
                Andadores (ID_and,ID_pac,Inicio)" +
                "VALUES
                ("+id_and+","+flow.get("nuevo")+","+inicio+")";
return msg;
```

El nodo Switch de la Figura 7.15 permite mostrar un mensaje de error si el andador que se ha asignado ya está ocupado o si se introduce un número negativo. Por otro lado, una vez realizada la asignación del andador, se manda un mensaje al subflujo que genera la tabla para que se actualice.

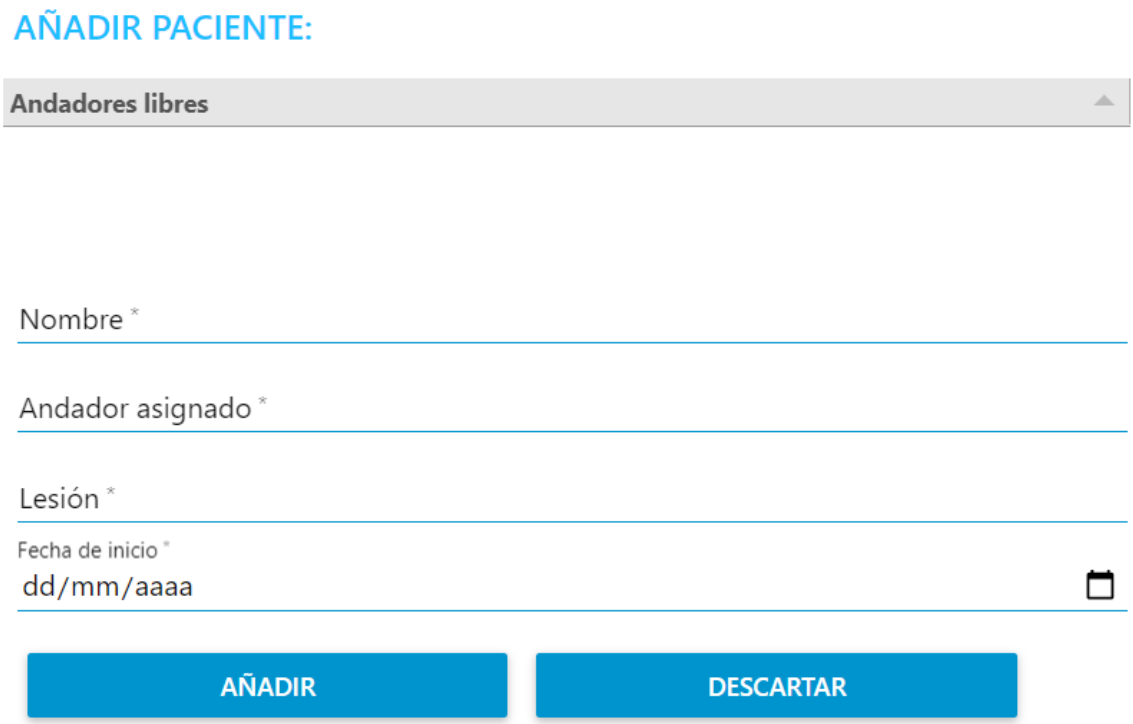
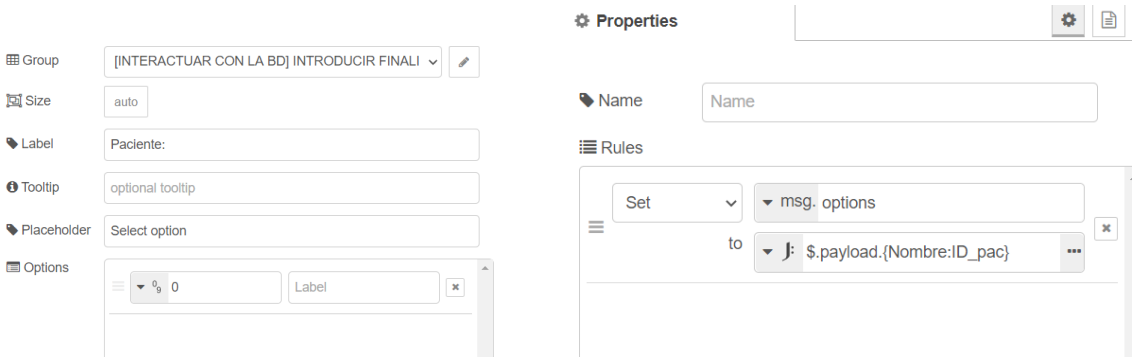


Figura 7.17: Resultado del módulo 1.

7.9.4. Generación de desplegables

Los módulos 2, 3 y 4 hacen uso de un desplegable para seleccionar el paciente al cual se desea aplicar la fecha de fin (módulo 2), corregir un dato (módulo 3) o descargar los datos (módulo 4).



(a) Opciones del desplegable.

(b) Configuración de mensaje de entrada al desplegable.

Figura 7.18: Creación de listados.

Se desea que estos desplegables sean generados de forma dinámica, de forma que cualquier paciente nuevo sea reflejado automáticamente en el mismo. Por ello el menu de propiedades de dicho nodo no puede ser configurado de forma manual y debe quedarse vacío (Figura 7.18a).

Se desea mostrar al usuario el nombre del paciente, pero que el mensaje resultante contenga el ID de dicho paciente. Para poder conseguirlo, se debe realizar una consulta a la base de datos de ambos valores y usar un nodo de tipo "set"para configurar las opciones del mensaje que entrará al listado (Figura 7.18b).

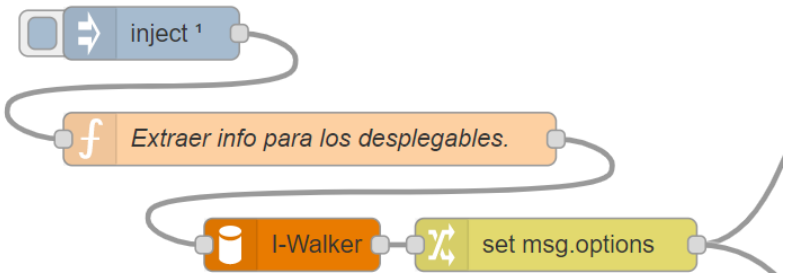


Figura 7.19: Subflujo para generar listados.

El subflujo resultante se muestra en la Figura 7.19 y su salida debe estar conectada a todos los listados de la interfaz gráfica. El nodo de inyección que se observa genera la consulta cada vez que se recarga la página, para actualizar el listado.

7.9.5. Módulo 2: Establecer fecha de finalización

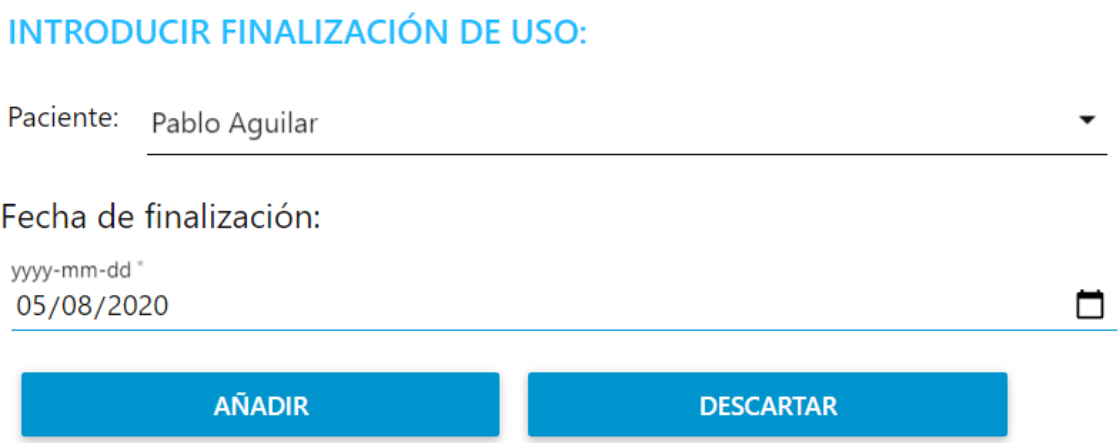


Figura 7.20: Módulo de establecimiento de fecha de fin de uso.

La selección de paciente queda almacenada en una variable global. Por otro lado, la fecha seleccionada por el usuario accede a la función que realiza la inserción del mismo a la base de datos. En dicha función se accede a la variable global mediante flow.get() y se formatea la fecha tal y como se realiza en el Código 7.14.

El flujo seguido por los datos puede observarse en la Figura 7.21.

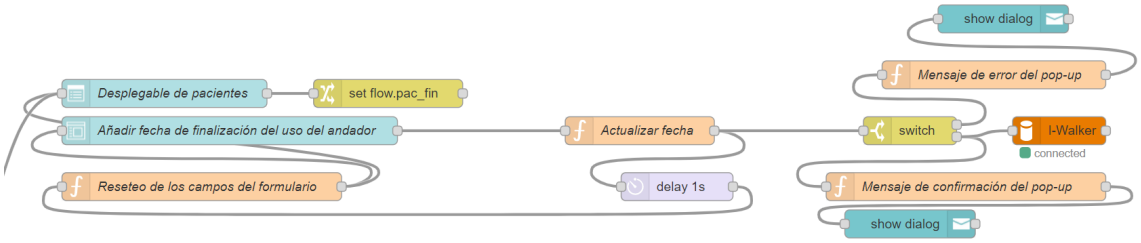


Figura 7.21: Flujo para inserción de final de uso del andador.

La actualización de campos se realiza como se muestra en Código 7.15. Es necesario especificar que se actualice aquel campo que es nulo, para que no se actualicen todas las fechas de fin asociadas a ese andador con antiguedad.

Se configura un nodo condicional para permitir el almacenamiento en la base de datos si la fecha es correcta y superior a la fecha de inicio, de no ser así, o bien si no se ha rellenado un valor, advertirá al usuario con un mensaje de error.

Código 7.15: Actualización de la fecha de fin de uso del andador.

```
msg.topic =
      "UPDATE Andadores SET Fin = '"+fin+"'
           WHERE
           ID_pac = "+flow.get("pac_fin")+"
              AND Fin IS NULL";
```

7.9.6. Módulo 3: Cambiar datos relativos a un paciente

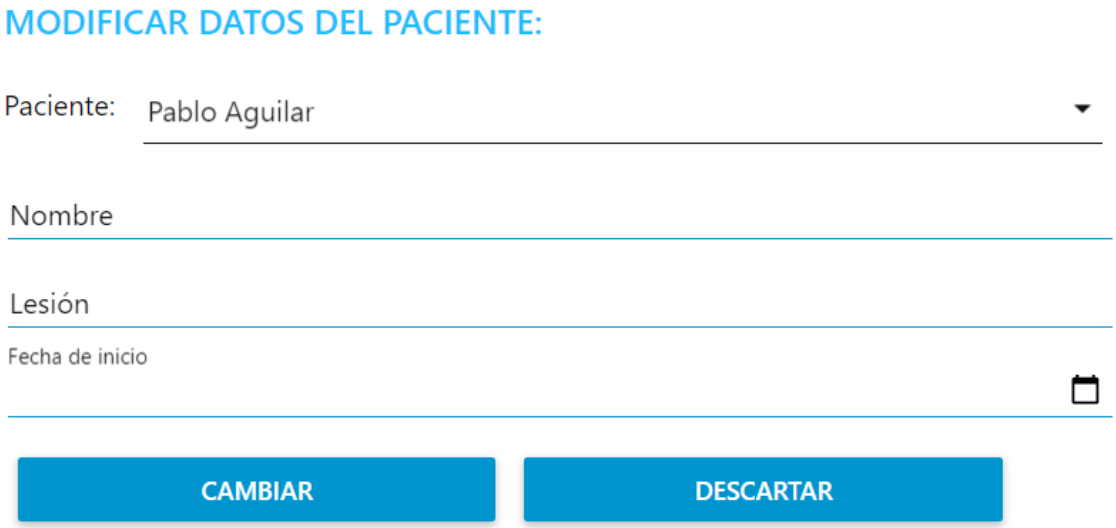


Figura 7.22: Flujo para actualización de campos.

La Figura 7.23 muestra el flujo de nodos empleado.El proceso es igual al descrito en el módulo 2, pero se debe tener en cuenta que si no se desea actualizar un campo y este se deja vacío, se ejecute una sentencia que no actualice dicho campo a nulo.

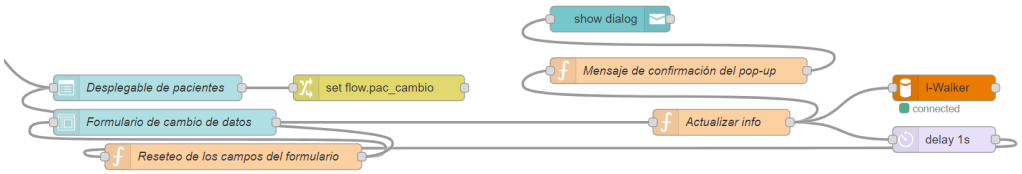


Figura 7.23: Resultado del módulo 3.

7.9.7. Módulo 4: Descarga de los datos a Excel

Este módulo posee tres partes diferenciadas. La primera de ellas es la encargada de generar los listados, tal y como se ha explicado en el Apartado 7.9.4 . La segunda de ellas se encarga de almacenar el paciente y el intervalo de fechas en el cual se desean descargar los datos (Figura 7.24) en variables globales.

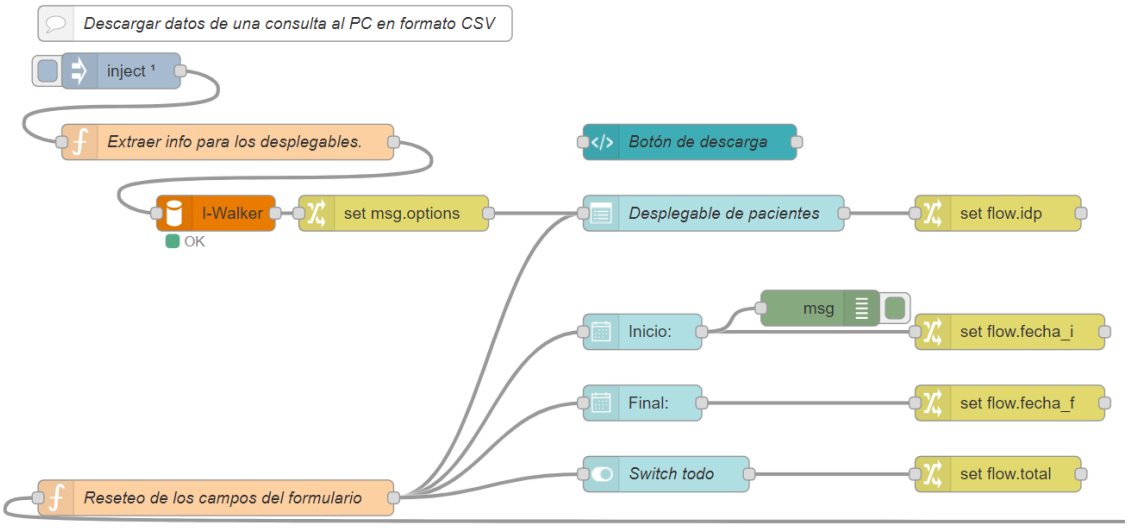


Figura 7.24: Flujo para guardar las selecciones

Finalmente, cuando se presiona el botón de descarga, se inicia una petición de tipo *get* en HTTP, que inicializa la consulta con los parámetros seleccionados. Estos datos son guardados en un archivo csv que permitirá acceder a los datos en excel. Este archivo es guardado en una dirección URL. La Figura 7.25 ilustra el flujo necesario para ello. Para generar lo mencionado a través del botón de descarga, es necesario estilar un enlace como un botón, por medio de HTML y CSS, usando el siguiente código.

Código 7.16: Botón de descarga y estilo CSS del mismo.

```
1 <a href="/Data"
2    type="button"
3    class="btn btn-outline-primary">HAZ CLICK PARA DESCARGAR
4    </a>
```

Es posible descargar todos los datos relativos a dicho paciente, en lugar de hacer uso de un intervalo de tiempo haciendo click en el switch. Cuando este está activado, omite la consulta con filtro de tiempo, por medio de una sentencia if dentro de la función. La apariencia del módulo se puede apreciar en la Figura 7.26.

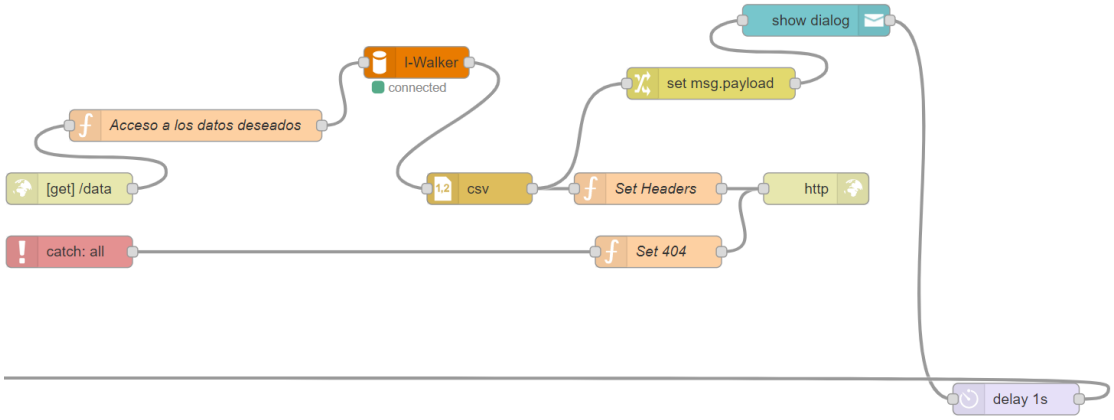


Figura 7.25: Flujo para guardar las selecciones

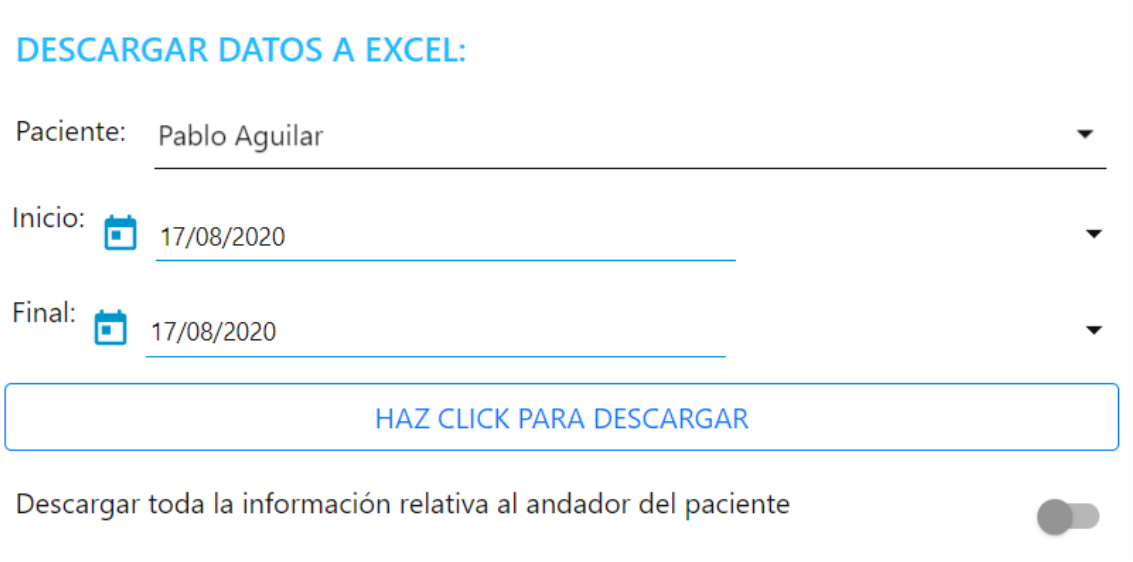


Figura 7.26: Resultado del módulo 4 de descarga de datos.

7.10. Plataforma de monitorización

En primer lugar se introduce la plataforma, mostrando sus elementos y funcionalidades, creando una visión clara del resultado final. Posteriormente se procede a detallar el proceso seguido para alcanzar dichos resultados.

7.10.1. Estructura principal.

Grafana permite crear diversos tipos de paneles, de forma que los datos puedan ser representados de la forma más conveniente. La Figura 7.27 muestra todas las opciones disponibles.

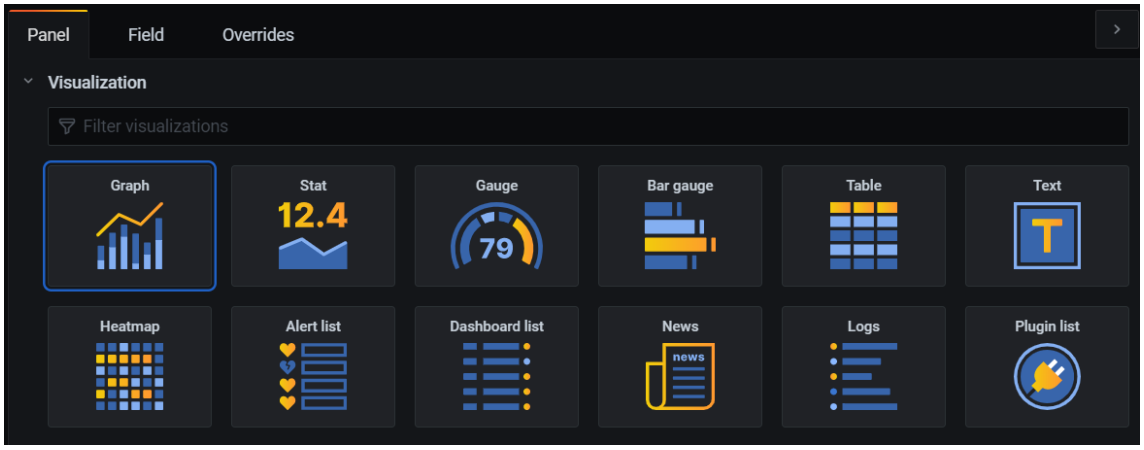


Figura 7.27: Formatos de representación de datos.

Además, una vez creados todos los paneles, se pueden crear módulos desplegables, de forma la información pueda ser extendida o ocultada a gusto del usuario. Se ha creido conveniente implementar esta función para añadir mayor contenido de información a la plataforma. Los módulos creados para este proyecto se muestran en la Tabla 7.6.

MÓDULO	***OBJETIVO***
Inicio	Muestra información general relativa al paciente al que corresponden los datos y muestra el estado del andador y su batería.
Resumen	Representa las lecturas de los sensores en una gráfica cada uno, para proporcionar información rápida y completa a golpe de vista.
Estadísticas	Contiene funciones agregadas de la base de datos para representar información relativa a la actividad del paciente y su mejora a lo largo del tiempo.
Históricos	Representa la misma información que el módulo resumen, pero se emplea una gráfica que ocupa toda la pantalla, para analizar con mayor detalle los datos. Además, solapa las gráficas de fuerza por un lado, y las de velocidad por otro, para poder compararlos.

Tabla 7.6: Módulos implementados en la plataforma y funciones de los mismos.

7.10.2. Primeros pasos.

Haciendo uso de la ruta de acceso y las credenciales suministradas en la Tabla 7.2, se accede a Grafana. Este servicio permite crear multiples dahsboards, cada uno con múltiples paneles. Se procede a crear un nuevo dashboard, como se muestra en la Figura 7.28 y la Figura 7.30.

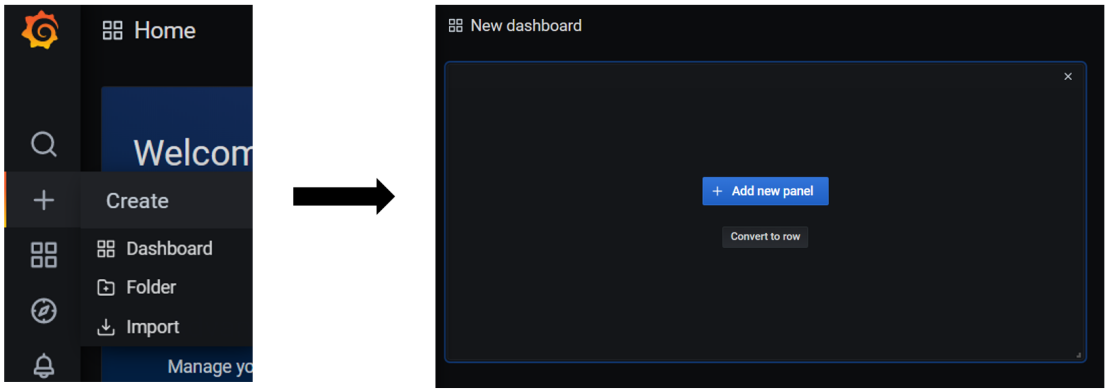


Figura 7.28: Creación de un nuevo dashboard.

Para poder añadir datos, es necesario especificar a Grafana a que tipo de base de datos necesita acceder, la dirección IP, el puerto y las credenciales de la misma.

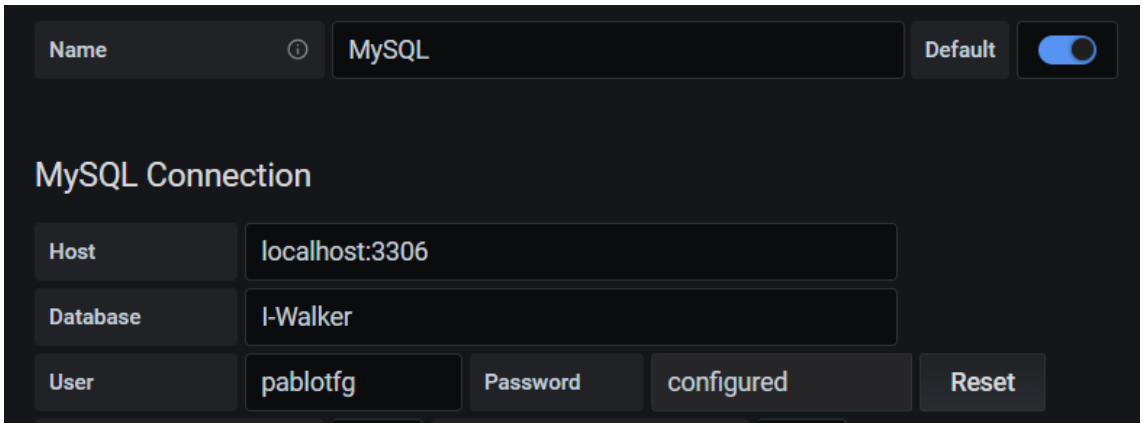


Figura 7.29: Vinculación de la base de datos con la plataforma.

7.10.3. Variables

Como la plataforma va a proveer datos de varios pacientes con varios andadores, es necesario poder representar los datos de cada paciente por separado. Mientras que

una opción podría ser crear un dashboard por paciente, Grafana permite la creación de variables. Esto habilita poder acceder en un mismo dashboard a los datos de varios pacientes, por medio de un desplegable. El sistema se diseña de la siguiente manera:

- Se declara una variable llamada Paciente.
- Grafana ofrece diversos tipos de variables, entre ellas, que el contenido sea obtenido a través de una consulta a la base de datos.
- Mediante el comando del Código 7.17, se establece que el contenido de la variable sea todos los pacientes de la tabla Pacientes.

Código 7.17: Adquisición de los pacientes registrados en la variable.

```
1 SELECT Nombre FROM Pacientes;
```

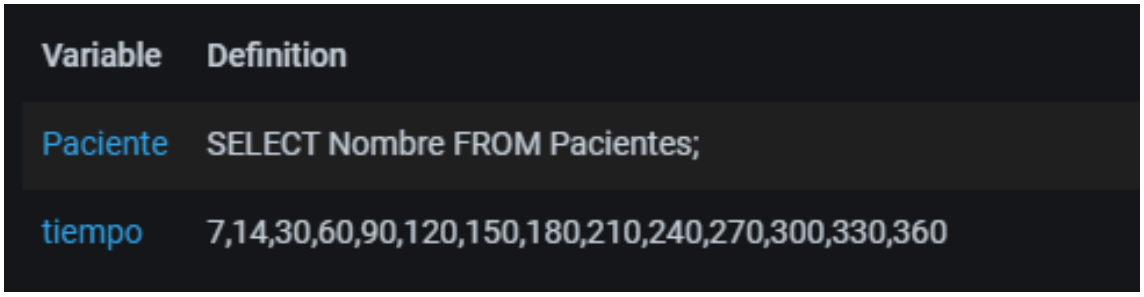


Figura 7.30: Variables creadas.

La tabla de los sensores no cuenta con el nombre del paciente. No obstante, si se recuerda la Figura 7.5, se creó una tabla denominada andadores, como nexo de unión entre ambas tablas. En diseños previos a la plataforma final, esta tabla no existía y fue por este motivo por el que se reestructuró la base de datos. Podría haberse creado la variable con el ID del andador directamente, pero en el desplegable se mostraría dicho ID y no el nombre, por lo que esta solución es más elegante y además cuenta con las ventajas añadidas que ya fueron explicadas en su correspondiente apartado.

Por otro lado, si se usa el desplegable que posee Grafana para la representación temporal, todos los paneles cambian acordemente. Los paneles de estadísticas proporcionan información útil sólo a grandes intervalos de tiempo, por lo que supone una molestia que estos cambien a intervalos de 5 segundos, 10 segundos, etc. Para evitar este inconveniente, se crea una variable que contenga diversas opciones de días a representar, desde 1 semana, hasta 1 año. De esta forma, el eje de abcisas solo varía cuando se cambia la opción de este nuevo desplegable.

7.10.4. Creación de paneles

Para ilustrar el proceso se muestra en detalle la configuración del panel que muestra la información de los sensores de fuerza.

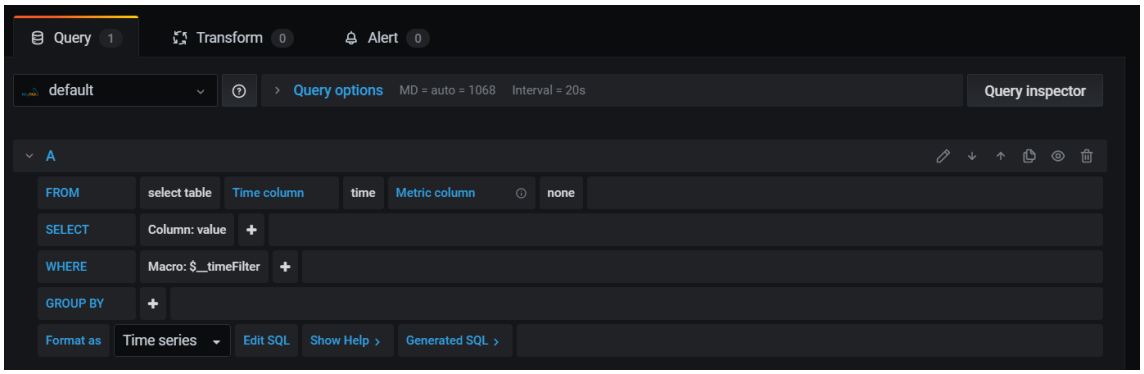


Figura 7.31: Panel de generación de sentencias SQL de Grafana.

En la Figura 7.31 se observa que al acceder al menu de creación de paneles, por defecto existe una interfaz de edición No obstante, se puede acceder al menu de sentencias SQL pulsando el botón -Edit SQL- el cual proporciona mayor flexibilidad.

La siguiente consulta SQL es muy similar a cualquier consulta que se realiza para cualquier tabla, únicamente cambiando los campos a seleccionar:

Código 7.18: Modelo de consulta SQL.

```
1  SELECT
2    $__timeGroupAlias(timestamp,$__interval),
3    F_der AS "Fuerza derecha",
4    F_izq AS "Fuerza izquierda"
5  FROM Sensores
6  WHERE
7    ID_and_s = (SELECT ID_and FROM Andadores
8                  WHERE
9                  ID_pac IN (SELECT ID_pac from Pacientes
10                    WHERE
11                    Nombre = '$Paciente')
12                )
13 ORDER BY $__timeGroup(timestamp,$__interval)
```

Como se mencionaba anteriormente, para acceder al ID del paciente que se ha seleccionado como variable, hay que filtrar la consulta con una serie de subconsultas, estableciendo que se representen los datos asociados al ID del andador que a su vez esta asociado a dicho paciente por medio de la tabla Andadores. Por otro lado, si se compara el campo where del código respecto a la Figura 7.31, se observa que se ha

eliminado la función timefilter. Ésta mostraba los datos en el intervalo de tiempo seleccionado en un desplegable perteneciente al servicio por defecto. Durante el desarrollo de la plataforma se descubrió fallos en la misma, por lo que se sustituye por una sentencia order by que cumple la misma función y funciona correctamente.

Dentro del menú del panel se pueden modificar opciones relativas a los ejes. Para todas las tablas, se configura el valor mínimo del eje de abcisas en 0 y el máximo en automático. Se añaden las etiquetas en los ejes y la leyenda identifica los valores representados y muestra los valores máximos, mínimos y la media de ellos. El resultado final es el siguiente.

7.10.5. Módulo de inicio

Para crear este módulo se hace uso de los siguientes tipos de paneles:

TIPO DE PANEL	***CONTENIDO***
Imágen	Contiene el Logo de Walkit
Textos	Información del paciente y títulos
Número	Indica si el estado del andador: Activo, no activo o fallo.
Indicador	Muestra el porcentaje de la batería.
Gráfico	Muestra la evolución del voltaje de la batería.

Tabla 7.7: Contenido del módulo de inicio.

Los paneles de texto están escritos en HTML, con la sentencia mostrada en el Código. El resto de textos cuenta con el mismo código pero muestran un contenido diferente.

Código 7.19: Textos HTML usados para los títulos.

```
1 <p style="font-size:250%;
2           font-weight:bold;
3           color:#C8D200;>">TEXTO </p>
```

En este panel se hace uso de unas características particulares de Grafana denominadas *Campos* (*Fields*, en la plataforma) y *Límites* (*Thresholds*, en la plataforma) y *Mapeo de valores* (*Value mappings*, en la plataforma) . Estos permiten modificar la forma en la que los datos son representados.

- **Indicador.** Representa el porcentaje de batería. En esta pestaña se configura la unidad como porcentaje 0-100, y se añaden configuran 4 límites, de forma que de 0-25 muestre un indicador rojo, de 25-50, amarillo y 50-100, verde.

- **Estado.** Este panel hace uso de un mapeo de valores. La entrada en la base de datos es numérica y se especifica que en lugar de mostrar el número, muestre un texto alternativo (0: no activo; 1: activo; 2:Error).

No se añade ninguna función agregada a la consulta, de forma que el valor representado es el último valor recogido en la base de datos. El resultado final del módulo se aprecia en la Figura 7.32.

Figura 7.32: Apariencia del módulo de inicio.

7.10.6. Módulo de resumen

Este módulo (Figura 7.33) está destinado a mostrar toda la información en una sola pantalla, para adquirir una visión general del estado de las lecturas en un momento dado.

Los paneles que componen este módulo se recogen en la Tabla 7.8:

TIPO DE PANEL	***CONTENIDO***
Textos	Títulos más elegantes que los que proporcionan los paneles por defecto.
Barras deslizantes	Representa el valor en tiempo real del sensor de fuerza.
Gráficos	Uno por sensor, pudiendo ver todos a golpe de vista.

Tabla 7.8: Paneles del módulo resumen.

No posee ninguna particularidad especial, a salvedad de que las barras deslizantes que muestran el estado de los sensores en tiempo real han sido configurados de la misma forma que el indicador de batería, para mostrar colores más intensos mientras más fuerza se aplique.

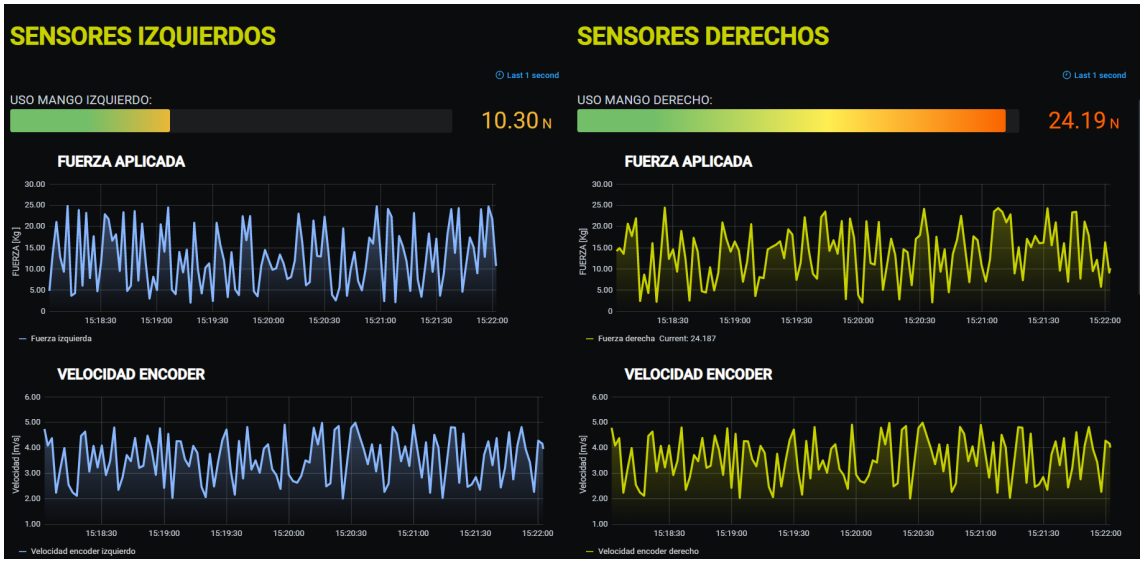


Figura 7.33: Módulo de resumen.

7.10.7. Módulo de estadísticas

Este módulo cuenta con dos gráficos y dos textos formateados en HTML y cuenta con la apariencia de la Figura 7.34

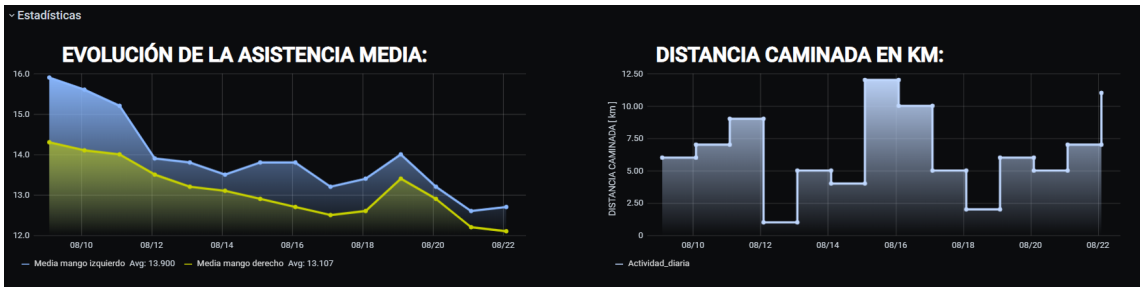


Figura 7.34: Módulo de estadísticas.

7.10.8. Módulo de históricos

Este módulo realmente está compuesto por dos módulos idénticos, cada uno dedicado cada par de sensores (Figuras 7.35 y 7.36). En el módulo de resumen se podía observar toda la información de un sólo vistazo, pero para un análisis más en profundidad, es deseable observar un gráfico que ocupe toda la pantalla y solape

ambos sensores en el mismo. Así mismo, si se desea estudiar la evolución de las lecturas desde un periodo determinado, también permite dicha posibilidad.

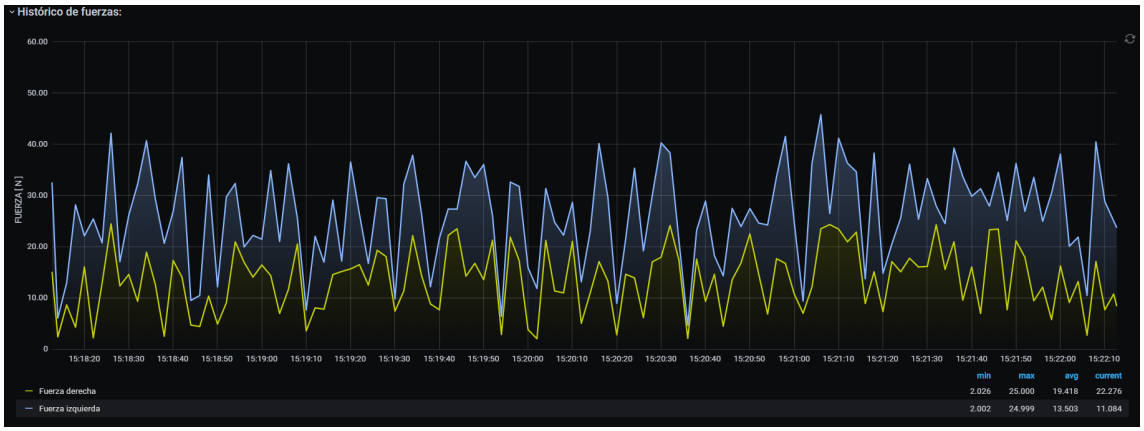


Figura 7.35: Módulo de histórico de fuerza.

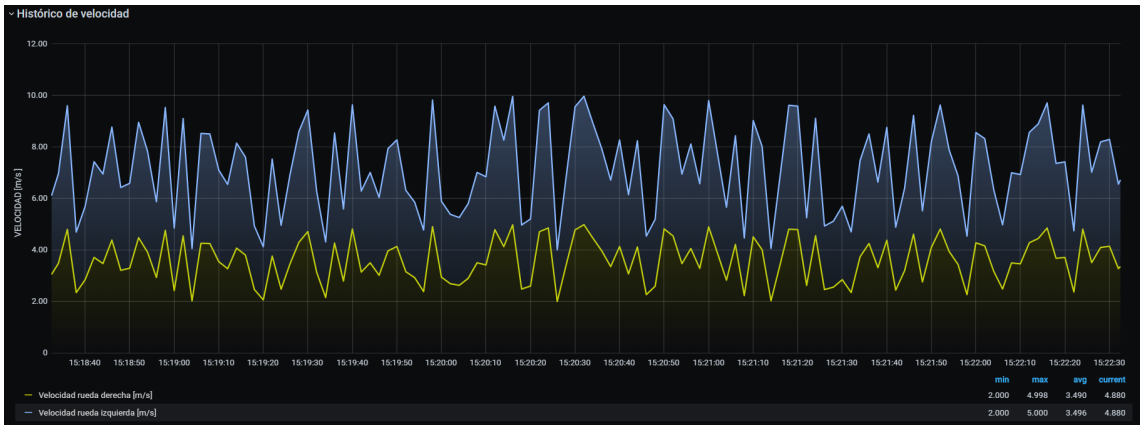


Figura 7.36: Módulo de histórico de velocidad.

7.10.9. Consideraciones finales

En cada gráfica que representa datos de más de una procedencia, es posible ocultar cualquiera de estos para estudiar solo uno con mayor precisión. Para ello solo es necesario hacer click en la leyenda de aquél que se dese ver.

Los gráficos históricos se han configurado en formato modular de forma que sus líneas no se solapen y el análisis de los mismos sea más sencillos. Esto implica que la gráfica superior no responde al eje de ordenadas, si no que su valor real sólo se puede observar colocando el ratón en la posición deseada.

Capítulo 8

Pruebas de la plataforma

Contenido

8.1. Envio de datos a la plataforma

Se desea verificar si los mensajes almacenados en el fichero de la tarjeta SD en el microcontrolador alcanza la plataforma correctamente. Como no se dispone de sensores durante la realización del proyecto, se genera un fichero que contenga 1000 objetos JSON, para simular como responde la plataforma ante la tasa de 100Hz de envio, la máxima que se podría implementar, durante 10 segundos, para observar si se genera pérdida de datos

Para verificarlo, se vacía la base de datos y se genera un paciente. A este se le asigna un andador. De esta forma ya pueden enviarse datos asociados a estos ID. Tras el envio de datos, deberían haberse generado 1000 filas, de lo contrario, ha habido perdida de información ya que no pueden existir campos duplicados debido al QoS 2.

El objeto enviado es el mismo que fué representado en el Código 7.8.

Se comprueba que no ha habido ningún tipo de perdida de de información y por lo tanto, la transmisión de datos se ha realizado con exito.

8.2. Prueba de la interfaz de monitorización

Para simular la apariencia real de la interfaz, se inyecta una serie de datos aleatorios con la misma frecuencia de envio que en un caso real. La apariencia de cada módulo ya se ha mostrado en el Apartado 7.10, no obstante se muestra una visión de la plataforma en conjunto en la Figura 8.1.

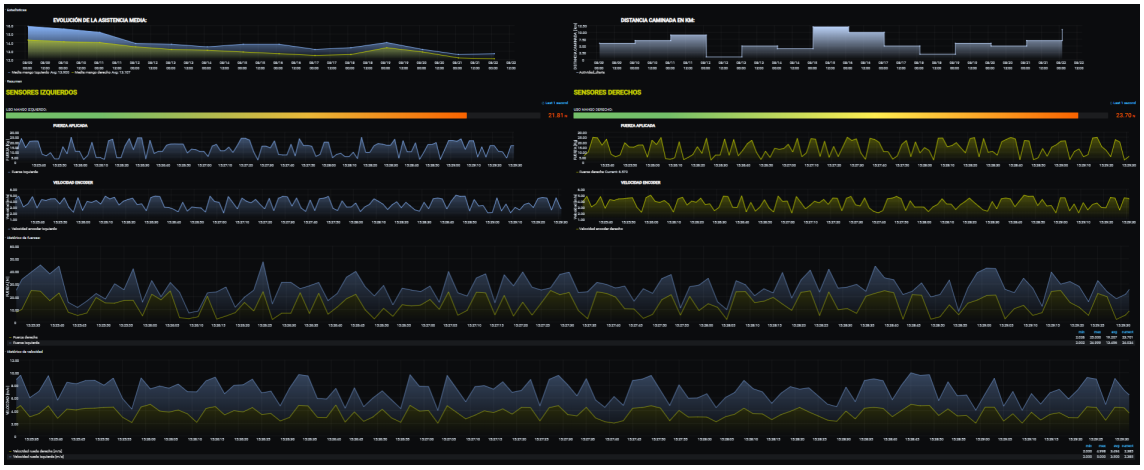


Figura 8.1: Plataforma de monitorización.

Una vez implementado, se ha experimentado con la plataforma. La metodología empleada se ha basado en hacer un uso intensivo de la misma como un usuario, con el fin de detectar posibles errores. Aunque en líneas generales, funciona correctamente, se destacan aquellos que han sido encontrados.

Primer error. El refresco automático de la página está configurado a 5 segundos, no obstante, al salir y entrar, se desconfigura solo y es necesario volver a configurarlo. Parece un error del servicio.

Segundo error. La lista de pacientes no se actualiza automáticamente, a pesar de estar configurada así. Es necesario acceder al menú de variables y actualizarlo manualmente.

Tercer error. En la memoria se ha mencionado que las estadísticas están sujetas a una parametrización del intervalo de tiempo representado diferente al resto de paneles. Se comprueba que al cambiar dicho intervalo responde adecuadamente, pero si en lugar de seleccionarlo en el panel, se hace una selección manual en el propio panel, los datos desaparecen del panel de estadísticas, tal y como se representa en la Figura 8.2.

Figura 8.2: Datos fuera de rango.

Como se puede apreciar, los errores detectados no son significativos y no entorpecen el correcto funcionamiento de la plataforma, pero deben ser estudiados en el futuro.

8.3. Prueba de la interfaz de la BD

En esta prueba se pretende en primer lugar plasmar el correcto funcionamiento de cada uno de los módulos y posteriormente simular posibles errors del usuario a la hora de introducir campos en la interfaz, para observar que no se generan datos incongruentes y los dialogos de error aparecen cuando el usuario añade un campo incorrecto.

Se parte de la base de datos completamente vacía, como se muestra en la Figura

8.3.

	Tabla	Acción	Filas	Tipo	Cotejamiento	Tamaño	Residuo a depurar
☐	Andadores	Examinar Estructura Buscar Insertar Vaciar Eliminar	0	InnoDB	utf8mb4_general_ci	32 KB	-
☐	Bateria_and	Examinar Estructura Buscar Insertar Vaciar Eliminar	0	InnoDB	utf8mb4_general_ci	32 KB	-
☐	Estadisticas	Examinar Estructura Buscar Insertar Vaciar Eliminar	0	InnoDB	utf8mb4_general_ci	32 KB	-
☐	Estado	Examinar Estructura Buscar Insertar Vaciar Eliminar	0	InnoDB	utf8mb4_general_ci	32 KB	-
☐	Pacientes	Examinar Estructura Buscar Insertar Vaciar Eliminar	0	InnoDB	utf8mb4_general_ci	16 KB	-
☐	Sensores	Examinar Estructura Buscar Insertar Vaciar Eliminar	0	InnoDB	utf8mb4_general_ci	16 KB	-
☐	Servos	Examinar Estructura Buscar Insertar Vaciar Eliminar	0	InnoDB	utf8mb4_general_ci	32 KB	-
	7 tablas	Número de filas	0	InnoDB	utf8mb4_general_ci	192 KB	0 B

Figura 8.3: Estado inicial totalmente vacío de la base de datos al comienzo de la prueba.

8.3.1. Módulo 1

Se introducen los datos mostrados en la Figura 8.4.

Figura 8.4: Introducción de los datos de prueba.

Como se observa en las Figuras 8.5a y 8.5b, los datos han sido introducidos correctamente.

ID_pac	Nombre	Lesion	DNI
1	Pablo Aguilar	Cadera	79037776R

(a) Datos incluidos en tabla Pacientes.

ID_and	ID_pac	Inicio	Fin
1	1	2020-08-00	*NULL*

(b) Datos incluidos en tabla Andadores.

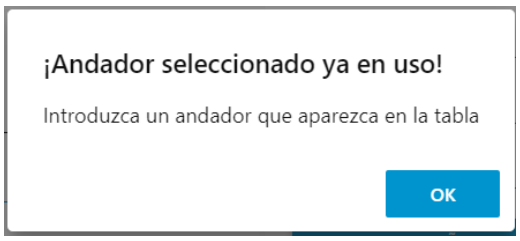
Figura 8.5: Comprobación del módulo 1. Los datos han sido introducidos en ambas tablas.

Por otro lado, se ha testado a cometer los siguientes posibles errores:

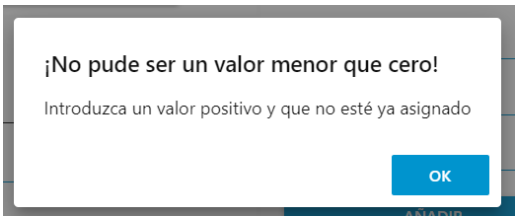
- Que algún campo se quede sin rellenar.
- Que el andador especificado sea un número negativo.
- Que el andador seleccionado ya esté en uso.

El primer error queda eliminado por parte del servidor, ya que se ha indicado que los campos sean obligatorios. Los dos siguientes problemas quedan teóricamente anulados con un nodo switch que desvia el flujo a un diálogo de error en lugar de hacia la consulta, tal y como muestra la Figura8.6a. Al intentar simular el error del ID negativo, el comportamiento es el esperado.

Para experimentar el comportamiento al seleccionar un andador seleccionado, se añaden un nuevo usuario y se establece su fecha de fin de uso, para que el sistema entienda que está libre. Al realizar el experimento, no funciona, por lo que se ha modificado ligeramente el flujo. La información plasmada en la tabla que contiene los andadores libres se almacena en una variable global. Se añade una nueva salida al nodo switch anterior en el que se compara el valor seleccionado con el contenido de la tabla. Una vez aplicado, el sistema sí funciona, mostrandose la ventana emergente de la Figura 8.6b.



(a) Ventana de Números negativos.



(b) Ventana de andador usado.

Figura 8.6: Ventanas emergentes de error.

Y la modificación pertinente tal y como muestra la Figura 8.7

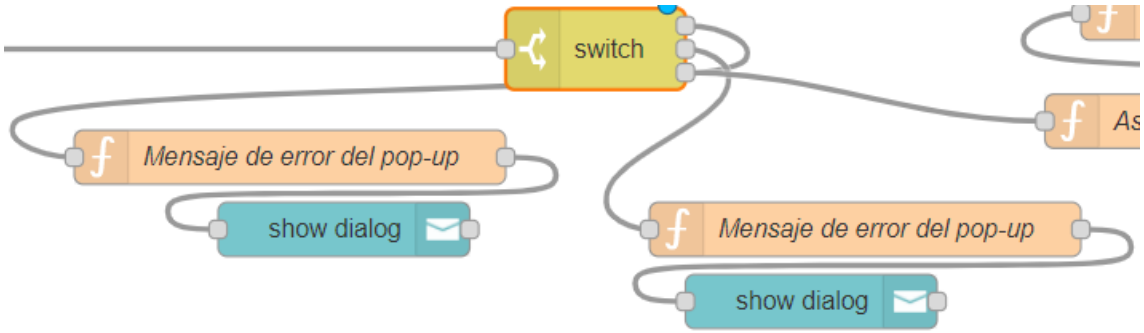


Figura 8.7: Modificación del flujo tras la prueba del módulo 1 .

8.3.2. Módulo 2

Una vez introducido el paciente anterior, se introduce su fecha de finalización. Si se contrasta la Figura 8.8 con la Figura 8.5b, se observa que el campo de fecha final se ha actualizado, por lo que el módulo funciona.

ID_and	ID_pac	Inicio	Fin
1	1	2020-08-05	*NULL*
2	2	2020-08-02	2020-08-26

Figura 8.8: Fecha de finalización añadida

Los posibles errors por parte del usuario que pueden darse son:

- Que no se rellene algún campo.
- Que se introduzca una fecha de final de uso más antigua que la de inicio, lo cual no sería lógico.

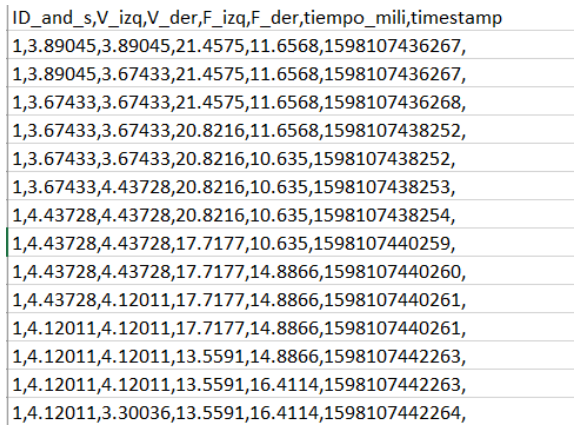
Se prueban ambos errors potenciales y se comprueba la eficacia del módulo ante el primer error. No obstante, no se ha encontrado forma sencilla de poder comparar fechas para aplicar la segunda por parte del usuario, por lo que no ha quedado implementado.

8.3.3. Módulo 3

En este módulo deben solo actualizar aquellos campos que hayan sido introducidos. Para ello se ha empleado un sencillo switch, case, de forma que se estipule una acción con cada caso posible. El funcionamiento es acorde a lo esperado.

8.3.4. Módulo 4

Se introduce en la base de datos una serie de datos aleatorios. A continuación se selecciona el paciente asociado a los mismos y se marca la opción que permite la descarga de todos los datos. Se observa como se genera un archivo csv que incluye todos los datos contenidos, como muestra la Figura 8.9.



ID_and_s,V_izq,V_der,F_izq,F_der,tiempo_mili,timestamp
1,3.89045,3.89045,21.4575,11.6568,1598107436267,
1,3.89045,3.67433,21.4575,11.6568,1598107436267,
1,3.67433,3.67433,21.4575,11.6568,1598107436268,
1,3.67433,3.67433,20.8216,11.6568,1598107438252,
1,3.67433,3.67433,20.8216,10.635,1598107438252,
1,3.67433,4.43728,20.8216,10.635,1598107438253,
1,4.43728,4.43728,20.8216,10.635,1598107438254,
1,4.43728,4.43728,17.7177,10.635,1598107440259,
1,4.43728,4.43728,17.7177,14.8866,1598107440260,
1,4.43728,4.12011,17.7177,14.8866,1598107440261,
1,4.12011,4.12011,17.7177,14.8866,1598107440261,
1,4.12011,4.12011,13.5591,14.8866,1598107442263,
1,4.12011,4.12011,13.5591,16.4114,1598107442263,
1,4.12011,3.30036,13.5591,16.4114,1598107442264,

Figura 8.9: Archivo csv generado tras la descarga.

El potencial error por parte del usuario en este módulo es que se seleccione un intervalo de fecha y el botón de descargar todo simultaneamente. Para probarlo, se añaden dos filas más a la tabla de sensores, uno con fecha del 22 de Agosto y otro con fecha del 23 de Agosto, mientras que el resto de datos están fechados en el 21 de Agosto. Se selecciona dicho intervalo en el módulo, de forma que se esperan solo dos resultados y no todos. El flujo cuenta con un switch que en caso de estar activado el botón deslizante, dirige el flujo hacia la consulta de todos los datos, por lo que debería funcionar. Tras descargar los datos, se observa que esto es lo que sucede.

Sin embargo, se ha observado que cuando el peso del archivo a descargar se vuelve considerable, el servidor puede fallar y desconectarse, por lo que en futuras versiones se debe trabajar más este aspecto.

8.4. Cometarios finales sobre las pruebas realizadas

Se puede concluir que todos los módulos funcionan acorde a lo esperado, tanto en su funcionalidad como en la prevención de posibles erratas por parte del usuario.

Conclusiones y líneas futuras

Durante el transcurso de la memoria se ha podido observar que la creación de una plataforma IoT es una tarea que implica poseer un conocimiento extenso sobre una gran variedad de áreas aparentemente inconexas entre sí. Su interconexión permite formar un conjunto único que permita obtener el resultado deseado. En el caso de este proyecto, se ha designado cada área como captación, red, procesado y aplicación.

Puede concluirse que, tras la realización del proyecto, las áreas de procesado y aplicación, es decir, la recogida de los datos, su gestión y presentación al usuario final, han sido aplicadas íntegramente y con éxito. Esto supone un gran avance en el sistema WalKit, al permitir disponer de toda la información relativa a los sensores del andador de cada uno de los pacientes de forma remota. Las capacidades de dicha plataforma se resumen a continuación.

- Puede alojar un número ilimitado de pacientes, sujeto a añadir mayor o menor capacidad de almacenamiento en la Raspberry Pi.
- Permite habilitar usuarios dentro de la plataforma, de forma que varios usuarios pueden hacer uso de la misma, con las restricciones pertinentes, y un administrador que lo modere.
- Los datos captados por los sensores de todos los pacientes pueden visualizarse en una plataforma de monitorización única. Esto proporcionará gran comodidad a las personas encargadas del análisis de los mismo al poder estudiar varios pacientes usando el desplegable proporcionado.
- Al contar con una interfaz para interactuar con la base de datos, el gestor solo es accedido por los técnicos. Los médicos solo deben rellenar formularios, evitando potenciales problemas que puedan surgir al dar acceso a personal no formado al gestor. Esto supone un grado más de usabilidad a la plataforma.

Por otro lado, la plataforma permite realizar dos tipos de análisis: evolución temporal de la recuperación del paciente ante la rehabilitación, mediante el módulo de

estadísticas, en primer lugar, y por otro lado, un análisis más en detalle del paso, en el momento deseado. No obstante, información con mayor alto grado de procesamiento, como muestran las referencias [1][5] (por ejemplo, la escala de movilidad Tinetti mencionada en el Apartado 2.3), no son soportados por la plataforma. Sin embargo, mediante la descarga de los datos en formato csv se solventa el problema. Los terapeutas pueden realizar dicho análisis de forma remota descargándolos periódicamente, aunque se propone como una línea de trabajo futura, permitir que el cálculo sea mostrado en la plataforma.

Respecto a la estética y comodidad de uso del andador por parte de los propios pacientes, no existen cambios significativos: se mantiene el número de cables y conexiones hasta llegar a la caja donde residen las baterías y el resto de electrónica.

Por otro lado, los datos mostrados a lo largo de la memoria han indicado que se produce una mejora del consumo del andador. El consumo de los sensores sin gestión de modo deep sleep se mantiene ya que no han cambiado respecto a la versión original, pero el M5Stack demanda sustancialmente menos corriente que la Raspberry Pi y además, permite gestionar dicho modo deep sleep. Debido a ello, el consumo de los sensores y del conjunto en general puede ser administrado de forma más eficaz respecto al modelo original. En consecuencia, a falta de realizar ensayos de consumo con instrumentación, se puede afirmar que la autonomía mejorará. Esto se traduce en una mejor usabilidad. Al poseer una mayor autonomía el paciente no se tiene que preocupar de cargar el dispositivo varias veces al día.

En cuanto a las capas de captación y red, estas no han sido implementadas totalmente. Esto se ha debido a la falta de información sobre el andador original, propiciado por la pandemia actual, impidiendo trabajar in situ sobre él. Los ensayos de consumo tampoco han sido realizados por el mismo motivo. Esto no quiere decir que no se haya trabajado sobre estos campos, si no que se proponen las partes restantes como trabajos futuros. Durante la memoria se han querido plasmar todas las decisiones de diseño e implementación, así como las librerías que se deben usar y los fragmentos de código más relevantes. Esto servirá de guía a aquella persona encargada de finalizar dichas secciones, tal y como se estipuló en el Apartado 1.3 del Capítulo 1.

Por otro lado, si se recuerdan los objetivos mencionados en el Apartado 1.2, todos han sido cumplidos: la revisión bibliográfica ha sido plasmada en gran detalle y todas las consideraciones de diseño han tenido en cuenta la funcionalidad real que tendrá el andador en un futuro. La plataforma resultante es elegante y funcional, a falta de ejecutar experimentos con pacientes y datos reales.

Además, el nuevo montaje del sistema WalKit es económico, residiendo el coste de cada kit, en la electrónica. El coste asociado a la plataforma de monitorización basada en el Internet de las Cosas es nulo, al ser todos los servicios elegidos, gratuitos,

y estar instalados en la propia Raspberry Pi - considerando esta como parte de la electrónica y no parte de la plataforma. No obstante, para cumplir algunas de las líneas de trabajo futuras, como el procesamiento de alto nivel, será necesario adquirir nuevos servicios que pueden conllevar un cierto coste adicional. La siguiente tabla refleja el coste de los elementos del modelo original respecto sus sustitutos tras el proyecto, dentro del andador, observando la mejora económica que ha supuesto.

MODELO	***COMPONENTE***	***CANTIDAD***	**PRECIO UNITARIO (Eur)**
Original	Raspberry Pi 3B+	1	40
	RS485 CAN Hat	1	15,95
	Power Bank 20000 mAh	1	30
	Protectores de batería HX-2S D01	2	0,56
		TOTAL	87,07
Nuevo	M5Stack Core Basic	1	23,5
	M5 Charge base	1	3,33
	M5 Commu Module	1	9,61
	M5 Cable Groove	1	3,60
		TOTAL	40,04

AHORRO ECONÓMICO DE LA NUEVA VERSIÓN
CUARENTA Y SEIS EUROS CON SESENTA Y SIETE CÉNTIMOS (46.67 €)

Tabla 8.1: Diferencia económica entre la versión original y la nueva propuesta de sistema de andador tras el proyecto.

Gracias a los cambios realizados, se observa un ahorro del 53 %. en el coste del sistema. Hay que destacar que evidentemente, para implementar la plataforma se debe incluir una Raspberry Pi en el hospital, incrementando el coste mostrado en 40 €, pero, al incrementarse en ambos, el ahorro permanece intacto. Si se compara el coste entre el modelo original, sin plataforma, respecto al coste asociado a la implementación de la misma, esta opción sigue siendo más rentable. La diferencia entre ambos es de 7,03 €, suponiendo un ahorro del 8,07 %.

Las posibilidades y el potencial que puede adquirir la plataforma en la actualidad es elevado y por ello se plantean, además de las ya mencionadas, las siguientes **líneas de trabajo futuras:**

- En primer lugar, acabar de depurar el código, para poder implantar el andador de forma real con pacientes reales.
- Los sistemas de gestión de ahorro de energía (*deep sleep*) no han sido implantados, ya que si se desea hacerlo de forma eficiente, supone un área de conocimiento que podría reflejarse en otro proyecto. Se propone que el M5Stack

entre en dicho modo cuando las lecturas de los sensores de velocidad y fuerza sean nulas, simultaneamente, pues un andador quieto no implica que el usuario no esté apoyado sobre él, y cabe la posibilidad de que haya objetos apoyados sobre los mangos (chaquetas o ropa) cuando se está en casa, manteniendo el micrcontrolador activo cuando este no debería estarlo.

- La monitorización del voltaje de la batería no es soportada por M5Stack. Requiere de la implementación de algunos componentes de hardware adicionales que lo permitan. Se recomienda estudiar esta linea de trabajo, pues es una información bastante importante para seguir mejorando la gestión de energía de los andadores. No obstante, la plataforma contiene la infraetsructura para poder representar dichos datos.

- Tras realizar pruebas reales, se propone estudiar cuando se deben enviar los datos. Por ejemplo, que el dispositivo deba estar cargando para la transmisión, de forma que no pueda quedarse sin batería durante el envio. O bien que solo pueda realizarlo con una batería superior a un umbral dado.

- La funcionalidad del servo no estaba implementada en el proyecto original, no obstante se ha creado la infraestructura en la plataforma para permitir su futura implementación.

- Se propone finalmente la implementación de sistemas de seguridad que aseguren la confidencialidad de los datos.

A título personal, este proyecto me ha otorgado una gran multitud de competencias transversales a mi formación, muy alejadas del campo de conocimiento de mi titulación, pero de una gran relevancia en la actualidad. Empecé el proyecto sin conocer el funcionamiento del IoT y tras la finalización del mismo estoy en proceso de crear una start-up de productos de smart water que utiliza las competencias adquiridas como uno de los elementos diferenciadores del producto.

Por todo lo mencionado, considero que se ha cumplido el objetivo más importante de un trabajo de fin de grado:

Mejorar la formación de un ingeniero.

Bibliografía

[1] Joaquin Ballesteros, Cristina Urdiales, Antonio B Martinez, and Marina Tirado. Automatic assessment of a rollator-user's condition during rehabilitation using the i-walker platform. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 25(11):2009–2017, 2017.

[2] Pallavi Sethi and Smruti R Sarangi. Internet of things: architectures, protocols, and applications. *Journal of Electrical and Computer Engineering*, 2017, 2017.

[3] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash. Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE communications surveys & tutorials*, 17(4):2347–2376, 2015.

[4] Kais Mekki, Eddy Bajic, Frederic Chaxel, and Fernand Meyer. A comparative study of lpwan technologies for large-scale iot deployment. *ICT express*, 5(1):1–7, 2019.

[5] Joaquin Ballesteros, Cristina Urdiales, Antonio B Martinez, and Jaap H Van Dieën. On gait analysis estimation errors using force sensors on a smart rollator. *Sensors*, 16(11):1896, 2016.

[6] Joaquin Ballesteros, Alberto Tudela, Juan Rafael Caro-Romero, and Cristina Urdiales. A cane-based low cost sensor to implement attention mechanisms in telecare robots. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 1473–1478. IEEE, 2019.

[7] Inmaculada Ayala, Joaquin Ballesteros, Juan Rafael Caro-Romero, Mercedes Amor, and Lidia Fuentes. Self-adaptation of mhealth devices: The case of the smart cane platform. In *Multidisciplinary Digital Publishing Institute Proceedings*, volume 31, page 23, 2019.

[8] J.Ballesteros y J.Manuél Gómez. Taislab/walkit. (último acceso el 26.08.2020). [Online]. Disponible en: https://github.com/TaISLab/WalKit.

[9] AMS sensors. As5600 datasheet. (último acceso el 26.08.2020). [Online]. Disponible en: https://ams.com/documents/20143/36005/AS5600_DS000365_5-00.pdf.

[10] Avia semiconductors. Hx711 datasheet. (último acceso el 26.08.2020). [Online]. Disponible en: https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/hx711_english.pdf.

[11] James W Dally and William F Riley. Theory of photo-elasticity. *Experimental Stress Analysis*, pages 406–446, 1978.

[12] J. Li, R. Jia, K. Zhang, Z. Yang, H. Liu, X. Zhu, and X. Li. Research on construction of crude set model of critical fault information for bus based on can-bus data. *IEEE Access*, 8:14875–14892, 2020.

[13] Ovidiu Vermesan, Peter Friess, Patrick Guillemin, Sergio Gusmeroli, Harald Sundmaeker, Alessandro Bassi, Ignacio Soler Jubert, Margaretha Mazura, Mark Harrison, Markus Eisenhauer, et al. Internet of things strategic research roadmap. *Internet of things-global technological and societal trends*, 1(2011):9–52, 2011.

[14] Ismael Peña-López et al. Itu internet report 2005: the internet of things. 2005.

[15] Diane J Cook, Michael Youngblood, Edwin O Heierman, Karthik Gopalratnam, Sira Rao, Andrey Litvin, and Farhan Khawaja. Mavhome: An agent-based smart home. In *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, 2003.(PerCom 2003).*, pages 521–524. IEEE, 2003.

[16] Stamatis Karnouskos. The cooperative internet of things enabled smart grid. In *Proceedings of the 14th IEEE international symposium on consumer electronics (ISCE2010), June*, pages 07–10, 2010.

[17] Bo Yan and Guangwen Huang. Supply chain information transmission based on rfid and internet of things. In *2009 ISECS International Colloquium on Computing, Communication, Control, and Management*, volume 4, pages 166–169. IEEE, 2009.

[18] Vinay Chamola, Vikas Hassija, Vatsal Gupta, and Mohsen Guizani. A comprehensive review of the covid-19 pandemic and the role of iot, drones, ai, blockchain, and 5g in managing its impact. *IEEE Access*, 8:90225–90265, 2020.

[19] Huansheng Ning and Ziou Wang. Future internet of things architecture: like mankind neural system or social organization framework? *IEEE Communications Letters*, 15(4):461–463, 2011.

[20] Peter Mell, Tim Grance, et al. The nist definition of cloud computing. 2011.

[21] Luis Joyanes Aguilar. Computación en la nube: Notas para una estrategia española en cloud computing. *Revista del Instituto Español de Estudios Estratégicos*, 1(1), 2013.

[22] Lady Ada. All the internet of things - episode one: Transports.

[23] Lady Ada. Adafruit Learning System. All the internet of things - episode two: Protocols. (último acceso el 26.08.2020). [Online]. Disponible en: https://cdn-learn.adafruit.com/downloads/pdf/alltheiot-protocols.pdf.

[24] Eyhab Al-Masri, Karan Raj Kalyanam, John Batts, Jonathan Kim, Sharanjit Singh, Tammy Vo, and Charlotte Yan. Investigating messaging protocols for the internet of things (iot). *IEEE Access*, 2020.

[25] Eduardo Buetas Sanjuan, Ismael Abad Cardiel, Jose A Cerrada, and Carlos Cerrada. Message queuing telemetry transport (mqtt) security: A cryptographic smart card approach. *IEEE Access*, 8:115051–115062, 2020.

[26] Espressif. Esp 32 datasheet. (último acceso el 26.08.2020). [Online]. Disponible en: https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf.

[27] Raspberry Pi. What is a raspberry pi. (último acceso el 26.08.2020). [Online]. Disponible en: https://www.raspberrypi.org/help/.

[28] M5Stack. M5docs core basic. (último acceso el 26.08.2020). [Online]. Disponible en: https://docs.m5stack.com/#/en/core/basic.

[29] Cloud MQTT. Cloud mqtt docs. (último acceso el 26.08.2020). [Online]. Disponible en: https://www.cloudmqtt.com/docs/index.html.

[30] WK Hauger and Martin S Olivier. Nosql databases: forensic attribution implications. *SAIEE Africa Research Journal*, 109(2):119–132, 2018.

[31] Node-Red. About. (último acceso el 26.08.2020). [Online]. Disponible en: https://nodered.org/about/.

[32] Grafana. Documentation. (último acceso el 26.08.2020). [Online]. Disponible en: https://grafana.com/grafana/.

[33] KAA IoT. Architecture overview. (último acceso el 26.08.2020). [Online]. Disponible en: https://docs.kaaiot.io/KAA/docs/current/Architecture-overview/.

[34] Thingspeak. Documentation. (último acceso el 26.08.2020). [Online]. Disponible en: https://thingspeak.com/pages/learn_more.

[35] Adafruit. About. (último acceso el 26.08.2020). [Online]. Disponible en: https://www.adafruit.com/about.

[36] Adafruit. Http iot api documentation. (último acceso el 26.08.2020). [Online]. Disponible en: https://io.adafruit.com/api/docs/#adafruit-io-http-api.

[37] Adafruit. Http iot api documentation. (último acceso el 26.08.2020). [Online]. Disponible en: https://io.adafruit.com/api/docs/#adafruit-io-http-api.

[38] AWS Amazon. Aws iot guía del desarrollador.

[39] Microsoft Azure. Reference architectures. (último acceso el 26.08.2020). [Online]. Disponible en: https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/iot.

[40] Arduino. Ntpclient library. (último acceso el 26.08.2020). [Online]. Disponible en: https://github.com/taranais/NTPClient.

[41] Arduino. Wifi udplibrary. (último acceso el 26.08.2020). [Online]. Disponible en: https://www.arduino.cc/en/Tutorial/WiFiSendReceiveUDPString.

[42] Richard Barry. Freertos library. (último acceso el 26.08.2020). [Online]. Disponible en: https://github.com/feilipu/Arduino_FreeRTOS_Library.

[43] Arduino. Wifi library. (último acceso el 26.08.2020). [Online]. Disponible en: https://www.arduino.cc/en/Reference/WiFi.

[44] Marvin Roger. Async mqtt client library. (último acceso el 26.08.2020). [Online]. Disponible en: https://github.com/marvinroger/async-mqtt-client.

[45] Benoît Blanchon. Arduinojson library. (último acceso el 26.08.2020). [Online]. Disponible en: https://arduinojson.org/.

[46] Justin J. Novack. Public brokers. (último acceso el 26.08.2020). [Online]. Disponible en: https://github.com/mqtt/mqtt.github.io/wiki/public_brokers.

[47] Qi Li, Weishi Li, Junfeng Wang, and Mingyu Cheng. A sql injection detection method based on adaptive deep forest. *IEEE Access*, 7:145385–145394, 2019.

Parte IV

Anexos

Anexo A

Códigos y scripts realizados

A.1. Introducción

Durante el desarrollo de la memoria se ha mencionado en numerosas ocasiones que los fragmentos de código mostrados poseen una función meramente ilustrativa, refiriendose a la existencia de un repositorio alojado en GitHub para aquellos interesados en hacer uso del mismo.

A.2. Contenido del repositorio

El repositorio contenía el siguiente material con anterioridad al desarrollo del proyecto.

- Documentación varia sobre cada sensor. Esquemáticos, diagramas, etc.
- Diseños de CAD de las piezas móviles y otros elementos.
- Scripts de los arduinos de cada sensor

A lo anteriormente mencionado, se incluye:

- Directrices para la instalación manual de los paquetes necesarios.
- Contenido de la tarjeta SD de la Raspberry Pi, para podeer flashear su contenido en cualquier otra Raspberry Pi.

- Modelo JSON del flujo de nodos que gestionan los datos y la interfaz gráfica, para poder replicarlo en cualquier otro dispositivo.
- Modelo JSON de la plataforma de monitorización, para poder replicarlo en otro dispositivo.
- Script del M5Stack.

A.3. Repositorio

Para acceder al repositorio, sólo es necesario acceder a la siguiente dirección [8]:

https://github.com/TaISLab/WalKit