Universitat Autònoma de Barcelona

Dipòsit digital de documents de la UAB

This is the **published version** of the bachelor thesis:

Sanchez Madrid, Raul; Aragonés Ortiz, Raúl, dir. Sistema de monitorización meteorológico y ambiental con componentes de bajo coste y gestión de mensajería MQTT. 2021. (958 Enginyeria Informàtica)

This version is available at https://ddd.uab.cat/record/248534

under the terms of the BY-NC-ND license

Sistema de monitorización meteorológico y ambiental con componentes de bajo coste y gestión de mensajería MQTT

Raúl Sánchez Madrid

Resumen— Este proyecto consiste en la creación de un prototipo de IoT compuesto por un microcontrolador y una serie de sensores, capaz de monitorizar métricas relacionadas con la meteorología y el ambiente. A su vez, se detalla la creación de una arquitectura capaz de ofrecer la transmisión, almacenamiento y visualización de estos datos de dos formas distintas. Se miden un total de siete métricas entre las cuales se encuentran la temperatura, humedad, presión atmosférica, calidad del aire, detección de lluvia, detección de luz y altitud y se proporcionan dos formas de monitorización basadas en dos plataformas distintas como son una aplicación móvil y un software libre. Estos dos sistemas se encuentran basados en dos protocolos de comunicación tales como Bluetooth y Wifi, otorgando al proyecto un amplio abanico de funcionalidades.

Palabras clave— Prototipo, Microcontrolador, Arduino, ESP8266 – ESP-01, Sensores, Métricas, Meteorología, Ambiente, SMART, MQTT, IoT, Broker, Node-RED, Base de Datos de series temporal, Grafana, Dashboard, Query.

Abstract— This project consists of the creation of an IoT prototype composed by a microcontroller and a group of sensors, capable of monitoring metrics related to meteorology and the environment. In turn, the creation of an architecture capable of offering the transmission, storage and visualization of this data in two different ways is detailed. A total of seven metrics are being measured, among which are temperature, humidity, atmospheric pressure, air quality, rain detection, light detection and altitude. Moreover, two forms of monitoring are provided based on two different platforms such as a mobile application and free monitoring dashboard. These two systems are based on two different wireless communication protocols. In particular Bluetooth and Wifi, giving the project a wide range of functionalities.

Index Terms—Prototype, Microcontroller, Arduino, ESP8266 – ESP-01, Sensors, Metrics, Meteorology, Environment, SMART, MQTT, IoT, Broker, Node-RED, Time Series Database, Grafana, Dashboard, Query.

1 Introducción

Este trabajo nace con el objetivo de crear un prototipo formado por un microcontrolador y una serie de sensores capaz de medir de forma precisa y exacta métricas relacionadas con la meteorología y el ambiente, para posteriormente poder monitorizar estos datos a través de dos plataformas distintas.

El principal propósito por el cual se decidió crear este prototipo era el de poder llegar a formar una red de estaciones meteorológicas de bajo coste capaz de generar un conjunto de datos de gran valor para su posterior uso en distintos fines como algoritmos predictivos o Big Data.

En la actualidad vivimos en la sociedad de la información donde existe una generación de datos que abre un mundo de posibilidades al estudio de patrones, predicciones o tendencias. La importancia de los datos es incuestionable y la creación de una red de estaciones meteorológicas de bajo coste nos podría proporcionar la manera de conseguirlos de una forma sencilla y con unos recursos limitados.

La información y los pronósticos climáticos son la base para la toma de decisiones en distintos campos como son la salud publica, agricultura, pesca, transporte, turismo y otros muchos. Para poder elaborar estos pronósticos y estudios se necesita de una base de datos sólida que cuanto mayor calidad tenga mejores resultados se podrán obtener.

Conforme ha avanzado el proyecto, se ha visto que las funcionalidades de este prototipo podrían ser mayores y que aparte de la propia generación de datos, la monitorización también tiene un rango amplio de funcionalidades y le otorga al proyecto un gran valor.

Actualmente, estamos viviendo una pandemia a nivel global que está cambiando la manera de concebir los espacios cerrados y la necesidad de ventilación. Por esta razón, vemos que nuestro prototipo tiene un gran potencial, ya que es capaz de medir las partículas por millón de CO_2 que como diversos estudios han afirmado, nos ayudan a poder controlar si la ventilación de un lugar es correcta y por lo tanto prevenir la expansión del virus y otras enfermedades[1]. No solo quedándonos con esta métrica, podemos medir otra como la temperatura o humedad que también tienen incidencia directa en nuestra salud.

En este documento se detallan los objetivos del proyecto, la metodología utilizada para alcanzarlos y la exposición de los resultados obtenidos llegando así a unas conclusiones sobre los mismos.

- *E-mail de contacto: raul.sanchezmad@e-campus.uab.cat*
- *Mención realizada: Inegeniería de Computadores*
- *Trabajo tutorizado por: Raúl Aragonés*
- *Curso 2020/21*

2 Objetivos

En esta sección se van a describir tanto los objetivos principales como los objetivos específicos que se establecieron para la realización del proyecto.

2.1 Objetivos Principales

El proyecto consta de tres objetivos principales que conjuntamente forman la base del trabajo realizado:

O.P.1. Creación de un prototipo funcional, compuesto por un microcontrolador y un grupo de sensores que nos permita obtener una serie de métricas relacionadas con la meteorología y el ambiente.

O.P.2. Creación de una aplicación móvil para la visualización de los datos basada en el protocolo de comunicación Bluetooth.

O.P.3. Creación de un conjunto de Dashboards en Grafana para la visualización y monitorización de los datos.

2.2 Objetivos Específicos

Complementando los objetivos principales existen una serie de objetivos más específicos que los complementan.

O.E.1. Establecer una metodología de desarrollo y una planificación para la realización del proyecto.

O.E.2. Diseño del prototipo y de su funcionamiento lógico.

O.E.3. Selección del microcontrolador, sensores y módulos de comunicación.

O.E.4. Integración e interconexión de los componentes con el microcontrolador.

O.E.5. Diseño de una aplicación móvil capaz de establecer comunicación con el prototipo a través del protocolo de comunicación Bluetooth.

O.E.6. Diseño de una arquitectura que permita enviar, tratar y almacenar los datos medidos.

O.E.7. Otorgar la capacidad de conectividad Wifi al sistema.

O.E.8. Envío de los datos a través del protocolo MQTT.

O.E.9. Tratamiento de los datos y guardado en una base de datos de series temporales.

O.E.10. Creación de paneles en el software Grafana.

3 Estado del Arte

En la actualidad el IoT es una de las grandes tendencias tecnológicas y está teniendo un gran impacto en los últimos años. El concepto de dotar de Internet a las distintas cosas que nos rodean en el día a día ha hecho que está tendencia se encuentre presente en cualquier ámbito y aspecto de la vida. Desde la agricultura estudiando cultivos u optimizando recursos, pasando por los edificios inteligentes enfocados en mejorar la sostenibilidad con la gestión y el uso eficiente de recursos hasta la sanidad son algunos de los ámbitos que abarca lo que también se denomina la Industria 4.0.

Según el último informe realizado por CISCO, se calcula que en el año 2023 habrá 29.300 millones de dispositivos electrónicos con conexión a internet en el mundo donde la mitad serán objetos del IoT[2].

Si nos centramos en el ámbito meteorológico vemos que ya en la actualidad el uso de sensores es una práctica asentada y el dotarlos de conectividad a Internet nos otorga la capacidad de poder interconectarlos entre ellos y obtener los datos de una forma eficiente.

Actualmente, encontramos infinidad de sistemas compuestos por sensores con el objetivo de monitorizar métricas específicas que nos permitan posteriormente poder estudiar estos datos o actuar en tiempo real. Un ejemplo de ello son los medidores de CO_2 que con la COVID-19 se han instaurado en todo tipo de recintos cerrados para monitorizar y asegurar las correctas medidas de ventilación.

En cuanto a la realización de proyectos de estación meteorológica con componentes de bajo coste encontramos una infinidad de ellos por internet, cada cual con sus variantes y objetivos específicos. Esto es debido a la accesibilidad que se tiene a los distintos componentes que nos permiten poder realizar todo tipo de proyectos sin necesidad de grandes recursos. Actualmente, encontramos kits de desarrollo que nos proporcionan un grupo de sensores y un microcontrolador por un precio razonable que permiten realizar proyectos a medida adentrándonos en el mundo del IoT.

4 Metodología y planificación

Para la realización del proyecto se han establecido metodologías de trabajo distintas para cada una de las tres partes que lo componen.

En primer lugar, se ha trabajado de manera continuada sobre el prototipo, añadiendo sensores de forma gradual a lo largo del proyecto. Para el desarrollo se ha utilizado el IDE llamado PlatformIO y el lenguaje de programación C++.

En el apartado de la aplicación móvil se ha optado por desarrollarla con el framework Flutter[3].

Finalmente, en el apartado de los Dashboards y la monitorización se ha optado por trabajar con el software libre Grafana. Para ello, se ha desplegado sobre una máquina virtual Ubuntu donde también encontramos una base de datos de series temporal basada en InfluxDB.

5 ARQUITECTURA DEL PROYECTO

En este proyecto se ha diseñado una arquitectura con tres partes claramente diferenciadas.

En la primera parte encontramos el prototipo, compuesto por el microcontrolador, un grupo de sensores y dos módulos que será el encargado de ir tomando las mediciones de las distintas métricas.

Por otro lado, se han diseñado dos maneras distintas de leer y monitorizar los datos generados por nuestro prototipo. Por una parte, existe una aplicación móvil conectada a nuestro sistema mediante el protocolo de comunicación Bluetooth. A su vez, encontramos una serie de Dashboards creados en Grafana obteniendo los datos mediante la tecnología de red inalámbrica Wifi.

Con esta arquitectura obtenemos dos formas diferentes basadas en dos protocolos de comunicación distintos de ver y monitorizar los datos. Esto nos permite tener un rango grande de funcionalidades y no depender exclusivamente de un protocolo de comunicación o tecnología para poder visualizarlos. La Figura 1 muestra un esquema general de toda la estructura del proyecto.

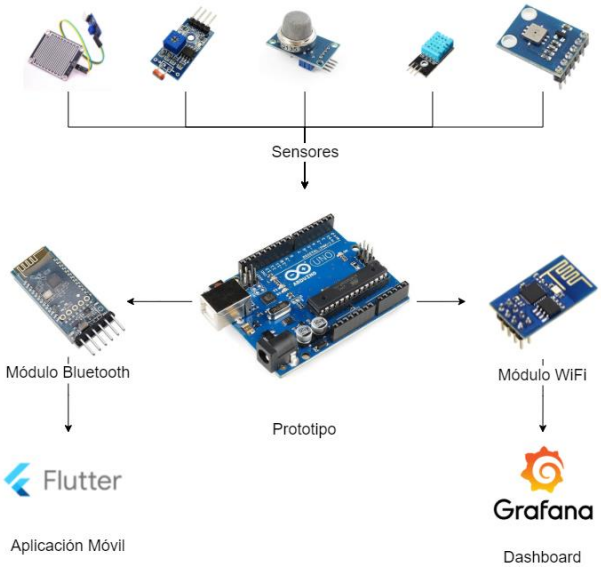

Fig. 1: Esquema general del proyecto.

5.1 Diseño del Sistema

La pieza angular de nuestro proyecto es el prototipo que como se ha comentado con anterioridad consta de un microcontrolador, un grupo de cinco sensores y dos módulos de conectividad. En este apartado vamos a ver más en detalle por que componentes está formado.

5.1.1 Microcontrolador

En el apartado del microcontrolador se ha optado por utilizar Arduino UNO. Es una placa basada en el microcontrolador ATmega328P y está compuesta por 14 pines de I/O digital, 6 entradas analógicas, un cristal de 16Mhz, conexión USB, Jack de alimentación, terminales para ICSP y un botón de reset. Para este proyecto se encuentra alimentada a 5V a través del cable USB. El esquema final de como se encuentran realizadas las conexiones entre el microcontrolador los sensores y módulos de conectividad se encuentra en el anexo A1 en la Figura 22.

Fig. 2: Vista frontal de Arduino UNO

5.1.2 Sensores

En cuanto a sensores, encontramos hasta cinco sensores que nos permiten medir un total de siete métricas distintas.

DHT11: Encargado de la medición de la temperatura y la Humedad. Nos ofrece una salida digital y se encuentra alimentado a 3,3V. Para su funcionamiento se utiliza la librería DHT.h[4].

Fig. 3: Sensor de Humedad DHT11

BMP085: Encargado de la medición de la presión atmosférica y altitud. La comunicación se realiza a través del bus I2C y se encuentra alimentado a 3,3V. Para su funcionamiento se utiliza la librería Adafruit_BMP085_U.h[5] que nos permite obtener además de la presión atmosférica la altitud a la que se encuentra el sensor mediante una conversión indicando la presión atmosférica promedio donde se encuentra situado.

Fig. 4: Sensor de Presión Atmosférica y Altitud BMP085

MQ-135: Encargado de la medición de la calidad del aire, concretamente de las ppm (partículas por millón) de CO_2. Esta medida hace referencia a la cantidad de unidades de una sustancia, en nuestro caso CO_2, que hay por cada millón de unidades del conjunto. Es un sensor muy común en proyecto de detección de gases debido a que es capaz de detectar distintos de ellos. Nos ofrece una salida analógica y se encuentra alimentado a 3,3V. Para su calibración y funcionamiento se ha utilizado la librería MQ135.h[6].

Fig. 5: Sensor de Calidad del Aire MQ-135.

Sensor de Lluvia: Encargado de otorgar a nuestro prototipo la capacidad de detectar lluvia. Esto se consigue mediante una placa que detecta la presencia de lluvia por la variación de la conductividad del sensor al entrar en contacto con el agua. Esta placa se encuentra conectada a un comparador LM393 que nos permite obtener una lectura digital cuando se supere cierto umbral, establecido a través de un potenciómetro. Se encuentra alimentado a 5 voltios y nos otorga una salida digital.

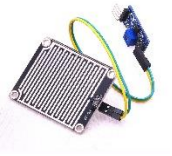

Fig. 6: Sensor de detección de Lluvia.

Sensor de Luz: Encargado de otorgar a nuestro prototipo la capacidad de detectar luz en el ambiente. Esto se consigue con un fotoresistor, que es un dispositivo cuya resistencia varía en función de la luz que recibe. Esta placa está compuesta por un comparador LM393 como el sensor anterior donde marcaremos el umbral de detección a través de un potenciómetro. Se encuentra alimentado a 3,3 voltios y nos otorga una salida digital con un valor de 0 o 1 dependiendo si detecta o no detecta luz.

Fig. 7: Sensor de Detección de Luz

5.1.3 Módulos de conectividad

Una vez vistos todos los sensores que componen nuestro prototipo y las métricas que son capaces de medir, vamos a ver los módulos que van a otorgar al sistema conectividad. Para ello, se ha optado por la elección de dos módulos.

ESP8266 ESP-01: Encargado de dotar a nuestro prototipo conectividad Wifi. Está basado en el SoC (System on Chip) ESP8266 integrando un procesador con una arquitectura de 32 bits. Tiene cuatro pines digitales de los cuales utilizaremos dos para la comunicación con el microcontrolador. Este módulo funciona a través de la comunicación Serial, para ello se utiliza la librería SoftwareSerial[7] que nos permite emular puertos de comunicación Serial sobre puertos de I/O digitales. Este módulo se encuentra alimentado a 3,3V.

Soporta el protocolo 802.11 b/g/n y un soporte de red de 2,4 GHz. Estas capacidades son más que suficientes para poder desempeñar el papel de módulo Wifi y otorgar a nuestro microcontrolador la capacidad de poder tener conexión a internet a través de redes inalámbricas. Para su funcionamiento se utilizan las librerías WiFiEsp.h y WiFiEspClient.h[8].

Fig. 8: Módulo Wifi ESP8266 ESP-01.

HC-31: Encargado de dotar a nuestro prototipo de conectividad Bluetooth. Utiliza el protocolo Bluetooth 2.0 y tiene una frecuencia de 2.4GHz. Tiene un poder de transmisión máxima de 4dBm lo que se traduce en una distancia de referencia de aproximadamente 10 metros. Trabaja alimentado a 5V y funciona mediante comunicación Serial por lo que se encuentra conectado a los dos puertos Serie de nuestro microcontrolador, TX y RX.

Fig. 9: Módulo Bluetooth HC-31

5.2 Aplicación móvil

Una de las dos maneras de monitorización y lectura de los datos que mide nuestro prototipo es a través de una aplicación móvil.

La vía de comunicación es la conexión Bluetooth que nos proporciona el módulo HC-31. Esta aplicación ha sido concebida para poder realizar una lectura de los datos en tiempo real de una forma fácil e intuitiva. No se contempla leer datos históricos, ya que toda la comunicación está basada en Bluetooth y por lo tanto solo recibiremos datos a partir del momento de la conexión con el prototipo.

Se ha creado la aplicación utilizando un plug-in llamado flutter_bluetooth_serial[9] que nos ofrece todo lo referente al manejo del módulo Bluetooth del dispositivo móvil, la conexión con el prototipo y la transmisión de los datos.

Como se ha visto con anterioridad, la transmisión de los datos se realiza a través de los puertos Serial de Arduino. En Bluetooth el envío de datos es en forma de Bytes y la recepción en la aplicación se realiza mediante un Buffer donde asignaremos cada Byte a la métrica correspondiente.

5.2.1 Lógica del Sistema

La lógica del Sistema implementada para la aplicación móvil se basa en el muestreo de las métricas y en la espera de conexiones de clientes para realizar el envío de los datos a través de Bluetooth.

El prototipo toma medidas cada dos minutos y en caso de que tengamos un mensaje de entrada a través del puerto Serie, comprobamos si el mensaje es un “Start” indicando que se desean recibir los datos vía Bluetooth, o si por otra parte es un “Stop” y por lo tanto se desea dejar de recibir los datos.

Si recibimos un "Start" significara que existe un cliente conectado a nuestro prototipo vía Bluetooth y por lo tanto pasaremos a enviar los datos a través del puerto Serie.

El envío de datos se realiza en forma de Bytes a través de la función Serial.write(). Se envía un primer byte con la parte entera de la muestra y un segundo byte con la parte decimal. Para las métricas de la presión atmosférica y la calidad del aire necesitaremos tres bytes, ya que son valores que no se pueden representar con dos.

Finalmente, volveremos a la condición inicial del margen de tiempo entre mediciones, ya que el código se ejecuta en bucle.

En la Figura 10 se puede ver el diagrama de flujo que muestra el funcionamiento descrito anteriormente.

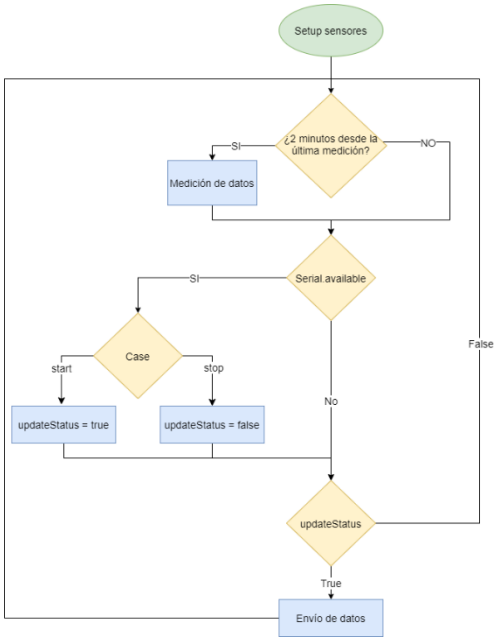

Fig. 10: Diagrama de Flujo de la lógica del sistema para la aplicación móvil.

5.2.2 Interfaz

Inicialmente se han diseñado las diferentes interfaces que componen la aplicación móvil.

En la Figura 11 se muestra la interfaz principal de la aplicación, donde encontramos un menú con dos secciones diferenciadas. Una primera sección dedicada a los ajustes del Bluetooth del dispositivo móvil y una segunda sección para la visualización de los datos.

En la opción Bluetooth de la primera sección podemos encontrar los ajustes que nos permite realizar la aplicación en el ámbito del Bluetooth de nuestro dispositivo móvil y la opción de Conexión con SMART METEO nos permite ver los dispositivos con Bluetooth activado disponibles para establecer conexión. En nuestro caso el módulo HC-31 recibe el nombre de BT UART.

Fig. 11: Interfaz principal.

Junto con la interfaz principal se observa una opción referente al listado de componentes donde se puede ver un listado de todos los componentes que componen el prototipo y dentro de cada uno una pequeña descripción de este. Estas interfaces se pueden ver en las figuras Figura 23 y Figura 24 situadas en el Anexo A2.

El resto de las interfaces referentes a la monitorización de los datos se podrán observar en la sección de resultados.

5.3 Dashboard

En esta segunda parte del proyecto tenemos el objetivo de crear una serie de Dashboards en Grafana, que nos permitan monitorizar las métricas medidas por nuestro prototipo. Para ello, se ha creado una arquitectura donde los datos siguen un flujo por diferentes tecnologías hasta llegar a estar presentes en los paneles.

Este apartado del proyecto se basará en la conectividad Wifi que nos ofrece el módulo ESP8266 ESP-01. A través de él, el prototipo publicará los datos medidos en un bróker a través del protocolo MQTT.

Un cliente, en nuestro caso node-RED será el encargado de obtener estos datos y almacenarlos en una base de datos de series temporales. Finalmente, Grafana atacará a esta base de datos mediante querys con las que podrá obtener los valores que serán mostrados en los distintos paneles.

En la Figura 12 se puede observar de una forma más clara la arquitectura para esta parte del proyecto y cual es el flujo de datos que se sigue desde que el prototipo toma las mediciones hasta que finalmente son mostradas en Grafana.

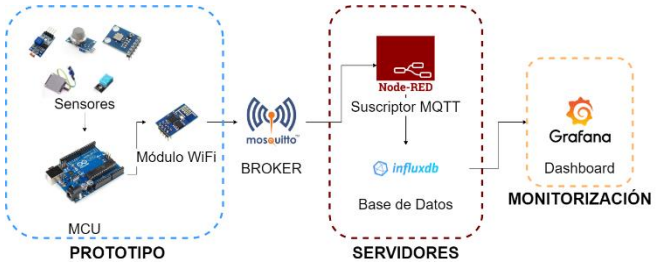

Fig. 12: Esquema General del apartado del Dashboard

5.3.1 MQTT

MQTT o Message Queue Telemetry Transport es un protocolo de comunicación Machine to Machine (M2M) de tipo messague queue empleado en el ámbito de IoT. Este protocolo está orientado a la comunicación de sensores gracias a su bajo consumo de ancho de banda y a que puede ser utilizado por la mayoría de los dispositivos empleados en proyectos de IoT que disponen de pocos recursos.

Funciona sobre TCP/IP como base para la comunicación, cada conexión se mantiene abierta y se reutiliza en cada comunicación.

El funcionamiento es el de un servicio de mensajería push con un patrón de publicador/suscriptor. En este tipo de infraestructuras los clientes se conectan a un servidor central llamado bróker que será el encargado de recepcionar los mensajes enviados por los clientes y distribuirlos entre sí mediante el sistema publicación/suscripción.

MQTT sigue una topología de estrella, con un nodo central que sería el bróker, lo que hace este protocolo altamente escalable permitiendo la conexión simultánea de diferentes dispositivos, ya que los clientes no dependen unos de los otros.

La estructura de un mensaje MQTT consta de tres partes, encontramos un encabezado fijo con un tamaño de 2 bytes, un encabezado variable con un tamaño de 4 bits que no es obligatorio y por último el mensaje también conocido como playload.

Cabecera Fija	Cabecera Variable	Mensaje(playload) 0-256Mbs
Código de Control 1 Byte Longitud del mensaje 1-4 Bytes		

Fig. 13: Estructura de un mensaje MQTT.

Uno de los puntos fuertes de este protocolo es la fiabilidad, algo sumamente importante en proyectos como el nuestro donde tenemos un envío constante de datos y no queremos que exista perdida de ellos. Para garantizar esta ventaja MQTT utiliza un servicio de calidad o QoS que determina como se entrega el mensaje a los suscriptores.

Existen tres grados de calidad:

- QoS 0: El mensaje se entrega como máximo una vez o no se entrega. Es la modalidad más rápida, ya que no se efectúa acuse de recibo y los mensajes no se almacenan.
- QoS 1: El mensaje se entrega como mínimo una vez, si el emisor no recibe el acuso de recibo el mensaje se enviará de nuevo con el distintivo DUP establecido hasta que se reciba el acuse de recibo. Esto hace que un mismo mensaje pueda ser enviado diversas veces y por lo tanto procesado diversas veces.
- QoS 2: El mensaje se entrega exactamente una vez y debe almacenarse localmente tanto en el emisor como en el receptor hasta que se procese. Es la modalidad más segura, pero también la más lenta, ya que antes de que el mensaje pueda suprimirse de la parte del emisor se deben realizar dos pares de transmisiones entre emisor y receptor.

La elección del nivel de QoS depende en las necesidades de fiabilidad que existan en el sistema, en nuestro caso queremos una fiabilidad alta, pero tenemos un tiempo entre envío de datos de dos minutos, lo que nos permite poder asumir un QoS de grado 2 sin afectaciones negativas en el rendimiento.

5.3.2 Broker

Para el proyecto se ha optado por utilizar Mosquitto[10] como bróker instalado en una máquina Windows.

La comunicación esta basada en unos temas (topics) que el cliente publica y los nodos que deseen recibirlo deben suscribirse a ellos. Los topics tienen su propia sintaxis y están representados mediante una cadena conformando una estructura jerárquica, separando cada jerarquía con una "/".

Los clientes inician una conexión TCP/IP sobre el bróker, el cual mantendrá un registro de los clientes conectados.

En la Figura 14 podemos ver como se han estructurado los distintos topics del proyecto, optando por un primer nivel jerárquico llamado sensores y un segundo nivel individual para cada uno de ellos. El cliente encargado de suscribirse al bróker para obtener los datos publicados por el prototipo será node-RED.

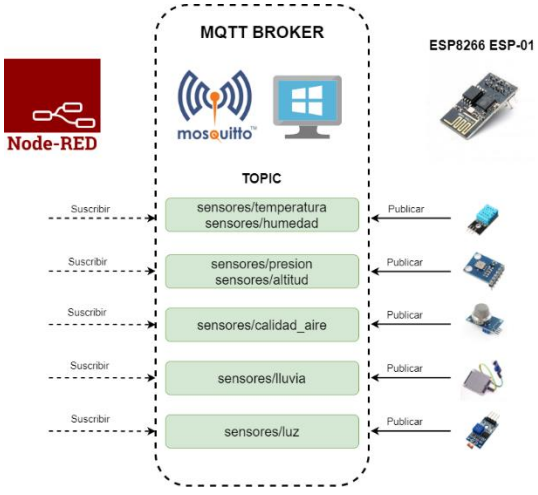

Fig. 14: Esquema topics MQTT.

5.3.3 Node-RED

El papel de suscriptor en este proyecto lo va a realizar node-RED[11]. Es una herramienta de programación visual y trata de dar solución a la complejidad que surge cuando queremos integrar hardware con otros servicios.

Está creado a partir de NodeJS que le proporciona la potencia necesaria para que sea una herramienta fiable y escalable y la librería JavaScript 3DS que es el encargado de proporcionar la interfaz web.

La estructura mínima en node-RED son pequeños módulos llamados nodos que realizan una función especifica. Estos nodos se organizan en flows cuando los conectamos entre ellos.

El cometido de node-RED en el proyecto es el de hacer de suscriptor de los distintos tópicos vistos anteriormente recogiendo los datos publicados por el prototipo, tratando estos datos y finalmente almacenándolos en una base de datos de series temporal.

Para ello, se ha creado el Flow que podemos observar en la Figura 15, compuesto por tres nodos.

El primero de ellos es el encargado de conectarse al bróker mediante el protocolo MQTT, suscribirse a los tópicos y recibir así los datos.

El segundo nodo está enfocado al tratamiento de los datos obtenidos por el primero, cambiando el tipo de dato para convertirlos en números, puesto que en el mensaje se han recibido como caracteres y este formato no nos interesa.

Por último, encontramos el nodo encargado de conectarse a nuestra base de datos de series temporales e insertar las diferentes mediciones en sus respectivas tablas.

Con este sencillo flujo de tres nodos conseguimos recibir los datos publicados por nuestro prototipo, hacer el tratamiento necesario para poder trabajar con ellos posteriormente y almacenarlos en la base de datos.

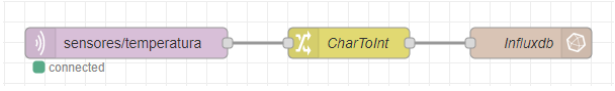

Fig. 15: Flujo de Temperatura en node-RED.

5.3.4 InfluxDB

En la sección anterior hemos visto como obteníamos los datos publicados por nuestro prototipo y lo guardábamos en una base de datos.

La base de datos escogida para este proyecto es InfluxDB[12]. Esta base de datos se encuentra alojada en una máquina Virtual Linux.

InfluxDB está concebida para bases de datos de time series o series temporales. Esta característica hace que se adapte perfectamente a nuestro proyecto, ya que uno de los requisitos para poder llegar a nuestros objetivos es guardar las distintas mediciones con sus correspondientes marcas temporales.

Otra ventaja importante es la velocidad, ya que las bases de datos de series temporales son más rápidas que las bases de datos relacionales a la hora de almacenar y procesar datos, lo que es de gran importancia para nuestro sistema que se encuentra constantemente midiendo métricas.

El funcionamiento es algo diferente a las bases de datos convencionales, se crean buckets que sería lo equivalente a tablas para cada una de las métricas, encontrando un total de siete buckets.

Los buckets son simples y están compuestas por cuatro columnas:

- _time: Tiempo en el cual se ha tomado el valor.
- _value: Valor de la muestra.
- _field: Cadena que representa el nombre del campo.
- _measurament: Nombre de la muestra.

En la Figura 16 se puede observar el bucket referente a la Calidad del Aire con las columnas y valores que lo componen.

_time	_value	_field	_measurement
2021-06-10 19:57:30 GMT...	586.74	value	ppm_co2
2021-06-10 19:59:30 GMT...	536.99	value	ppm_co2
2021-06-10 20:01:30 GMT...	595.31	value	ppm_co2
2021-06-10 20:03:30 GMT...	536.99	value	ppm_co2
2021-06-10 20:05:30 GMT...	578.24	value	ppm_co2

Fig. 16 Bucket de la medición de Calidad del Aire en InfluxDB.

5.3.5 Grafana

Todo este flujo de datos finaliza en Grafana[14]. Grafana es un software libre que tiene el objetivo de visualizar datos de series temporales. Nos permite crear paneles con los datos recolectados por nuestro sistema siendo capaces de monitorizarlos y disponer de paneles dinámicos para el análisis de las distintas métricas.

La forma de obtención de los datos que vamos a utilizar para su posterior visualización es mediante unas querys que atacan a la base de datos de series temporales alojada en InfluxDB. Estas querys están hechas en un lenguaje llamado Flux.

Flux es un lenguaje de scripts y consultas para bases de datos de series temporales que ha sido optimizado para el proceso ETL (Extracción, Transformación y Carga). La sintaxis de Flux está basada en el lenguaje JavaScript y una característica principal es la compatibilidad que tiene con diferentes fuentes de datos.

La sintaxis de las querys generalmente es la misma, primero de todo deberemos escoger de que bucket queremos extraer los datos. Una vez indicado esto, describiremos el rango temporal sobre el cual queremos los datos y con un filtro seleccionaremos la medida que deseamos. Una vez tenemos esto ya podremos realizar otras operaciones como hacer medias, mapear valores o seleccionar el primero, último, mayor, menor, etc.

En el Anexo A3 podremos encontrar un ejemplo de Query y el resultado que nos ofrece.

5.3.6 Lógica del Sistema

Una vez vistas las diferentes teconologías por las que pasan los datos en este ámbito del proyecto vamos a ver cual es la lógica del sistema en cuanto a microcontrolador.

La lógica del sistema en la parte del Dashboard se basa en la medición y posterior publicación de los datos a través del protocolo MQTT.

Al iniciar el sistema nos conectamos a la red Wifi configurada en el microcontrolador para tener conexión a internet.

Si han pasado dos minutos desde la última medición, procederemos a comprobar que la conexión a la red Wifi establecida al inicio es correcta. Si no es así, utilizaremos la función de reconexión para volver a establecer-la. Una vez comprobada la conectividad, haríamos lo mismo con el servidor Broker.

Por último, comprobada la correcta conexión tanto a la red como al Broker, pasaríamos a la medición de los datos y a su publicación en los distintos tópicos vía protocolo MQTT, volviendo una vez finalizado a la condición del intervalo entre medición y medición.

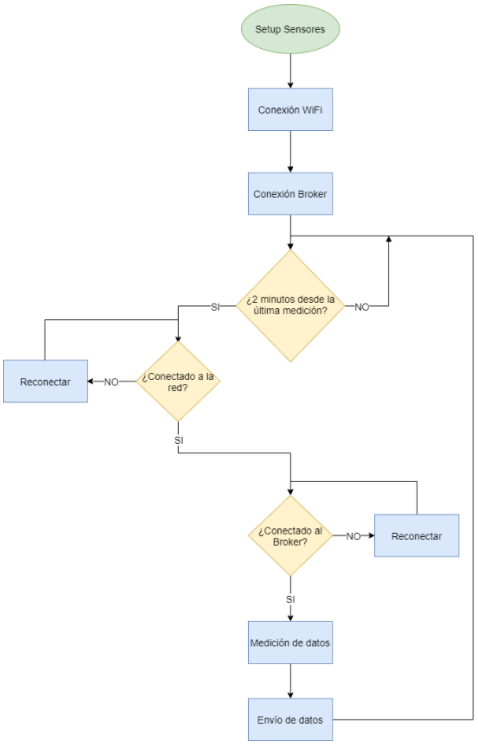

Fig. 17: Diagrama de Flujo de la lógica del sistema para los Dashboards.

6 RESULTADOS

En este apartado se describen los resultados obtenidos en los diferentes ámbitos del proyecto.

6.1 Aplicación Móvil

Para la aplicación móvil se han diseñado las interfaces que nos permiten monitorizar las distintas métricas.

En la Figura 18 encontramos la visualización de la temperatura y humedad en forma de gráficos radiales.

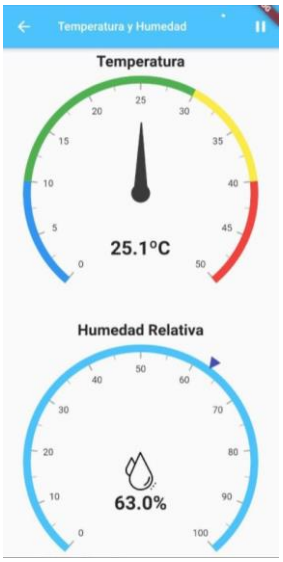

Fig. 18: Interfaz Temperatura y Humedad.

En la Figura 19 podemos observar una gráfica que indica la calidad del aire en ppm de CO_2, encontrando también una lista de con los distintos niveles establecidos en la normativa NTP 742: Ventilación general de edificios[15] para poder interpretar el valor mostrado.

Fig. 19: Interfaz Calidad del Aire.

Finalmente, en la Figura 20 observamos la interfaz donde podemos visualizar un resumen con todas las métricas.

Fig. 20: Interfaz de Todos los Datos.

En el anexo A2 Figura 24 podemos encontrar una interfaz que nos permite ver gráficas de como fluctúan los datos mientras estamos conectados al sistema.

6.2 Grafana

Para la creación de los paneles en Grafana se ha seguido una política de directorios diferenciados por el rango de tiempo que monitorizan. Encontramos un total de tres directorios:

- Diario: Rango de las últimas 24 horas.
- Semanal: Rango de los últimos 7 días.
- Mensual: Rango de los últimos 30 días.

6.2.1 Paneles Diarios

Dentro de esta primera sección encontramos paneles que monitorizan los datos en un rango de las últimas 24 horas. Existen un total de cinco paneles creados.

El primer panel es el principal, en él podemos encontrar una recopilación de todas las métricas que nuestro prototipo monitoriza en tiempo real. Su utilidad principal es poder tener recogidas todas las métricas en un mismo panel.

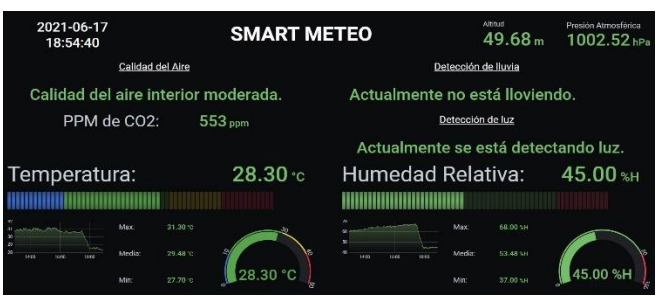

Fig. 21: Panel principal.

Si analizamos un poco el panel, vemos arriba a la izquierda el día y la hora, esto nos indica el momento en el que se tomaron las medidas que se muestran en el panel, siendo muy útil para ver la actualidad de los datos.

Encontramos la calidad del aire en ppm de CO_2 y el rango en el que se encuentra siguiendo el mismo estándar comentado con anterioridad para la aplicación móvil.

Se observa también la presión atmosférica, altitud, detección de luz y la detección de lluvia.

Finalmente, encontramos tanto la temperatura como la humedad relativa, viendo unos pequeños gráficos de las últimas seis horas y la máxima, mínima y media registrada en las últimas 24 horas.

Se puede observar el panel principal de forma completa en el enlace de la referencia[16].

Complementando este panel principal encontramos paneles específicos para cada métrica, permitiendo así la monitorización individual de cada una de ellas. El diseño se encuentra unificado para todos ellos. En este panel podemos encontrar una gráfica que muestra la variación de la métrica en las últimas 24 horas. Encontramos también el valor actual, la máxima, mínima y media registrada.

Finalmente, vemos dos desplegables, el primero de ellos muestra una gráfica comparativa entre el día actual y el día anterior. El segundo desplegable muestra gráficas donde se compara la métrica del panel con el resto de ellas, permitiendo ver si existen correlaciones entre ellas.

Se puede observar el panel diario de calidad del aire de forma completa en el enlace de la referencia[17].

6.2.2 Paneles Semanales

Dentro de esta sección encontramos paneles que monitorizan los datos en un rango de los últimos siete días. Existen un total de cinco paneles creados. Todos los paneles son individuales para cada métrica y siguen el estilo de los paneles diarios descritos en la sección anterior.

Se puede observar la medición actual, media semanal, la máxima y mínima semanal y el día y hora en el que se registraron. Siguen apareciendo dos desplegables con la comparación de la última semana con la anterior y de la métrica del panel con el resto de las métricas.

Se puede observar el panel semanal de temperatura de forma completa en el enlace de la referencia[18].

Para este rango temporal se añade un nuevo panel referente a la detección de lluvia y luz. En este podremos ver de forma gráfica el tiempo que se han detectado estas métricas en los últimos siete días y el valor total. Este panel nos sirve para poder monitorizar las horas de luz y lluvia que ha habido durante los últimos siete días, siendo capaces de analizar los días individualmente.

Se puede observar el panel semanal referente a la detección de lluvia y luz de forma completa en el Anexo A4 Figura 27.

6.2.3 Paneles Mensuales

Dentro de esta sección encontramos paneles que monitorizan los datos en un rango de los últimos 30 días. Existen un total de cinco paneles creados. Se sigue la misma política que en el rango semanal y los diseños son iguales.

Se puede observar el panel mensual de la humedad relativa de forma completa en el enlace de la referencia[19].

7 CONCLUSIONES

Este proyecto implementa la creación de un sistema capaz de monitorizar una serie de métricas relacionadas con la meteorología y el ambiente.

En el apartado del microcontrolador se ha trabajado con Arduino UNO utilizando cinco sensores que nos proporcionaban las lecturas de hasta siete métricas como son temperatura, humedad, presión atmosférica, calidad del aire en ppm de CO_2, detección de lluvia, detección de luz y altitud y dos módulos de para otorgar conectividad al sistema.

Se han creado dos formas distintas de monitorizar los datos basadas en dos plataformas como son una aplicación móvil y un sistema de paneles en un software libre. Cada una de estas variantes utiliza un protocolo de conexión distinto lo que nos amplía el rango de funcionalidades para el cual se puede utilizar este sistema.

Para la aplicación móvil se ha trabajado principalmente el front-end recibiendo los datos y mostrándolos en forma de gráficas.

Todos los datos medidos se han almacenado en una base de datos de series temporales que nos ha permitido poder generar un histórico con el que posteriormente realizar los paneles. Para el envío de estos datos se ha trabajado con el protocolo MQTT viendo sus ventajas y el potencial que tiene en el ámbito del IoT. De esta base de datos es de la cual se ha alimentado Grafana para la creación de distintos paneles con rangos temporales diversos permitiendo la monitorización y el análisis de las distintas métricas.

Se han alcanzado los objetivos establecidos al inicio del proyecto obteniendo unos resultados satisfactorios.

En lo personal, el proyecto me ha permitido adentrarme en el mundo del IoT tanto en lo referente a hardware como en el ámbito de protocolos y conexiones. La heterogeneidad de tecnologías utilizadas como son Flutter, node-RED, bases de datos de series temporales o Grafana lo han hecho un proyecto muy enriquecedor a nivel de conocimientos.

LÍNEAS FUTURAS

Uno de los objetivos de futuro sería la implementación de un sistema de alimentación a través de una batería de litio alimentada por una placa solar. De esta forma, se conseguiría un sistema completamente autónomo lo que ampliaría en gran medida su rango de funcionalidades. Junto a este sistema de alimentación, se podría desarrollar que el propio Arduino fuera capaz de ver el estado de la batería y cambiar así su tiempo entre mediciones adaptando el consumo a la energía restante.

Otro objetivo futuro seria alojar el servidor bróker en el Cloud así como la base de datos de series temporal, ya que actualmente se encuentran en un servidor local.

Añadir otros sensores como un anemómetro capaz de medir la velocidad del viento serian avances interesantes para enriquecer más el histórico de datos recogidos y añadir valor al sistema.

AGRADECIMIENTOS

En primer lugar, agradecer a mi tutor Raúl Aragonés por darme la oportunidad de realizar este trabajo y los consejos y el soporte mostrado a lo largo de todo el proyecto.

A nivel personal, también quiero agradecer a mi familia por ofrecerme un lugar donde poder trabajar y tener siempre el sistema funcionando, así como su apoyo mostrado.

BIBLIOGRAFÍA

[1] S. Garcia (2021, Febrero 27). ¿Por qué la concentración de CO2 en un lugar puede ser indicador de riesgo de contagio de COVID-19? [Online]. Disponible: https://www.aa.com.tr/es/mundo/-por-qu%C3%A9-la-concentraci%C3%B3n-de-co2-en-un-lugar-puede-ser-indicador-de-riesgo-de-contagio-de-covid-19/2158723

[2] CISCO, Cisco Annual Internet Report (2018–2023) White Paper [Online]. Disponible: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[3] Documentación Flutter. https://flutter.dev/docs

[4] "DHT-sensor-library". Librería para la serie de sensores DHT de Temperatura y Humedad. Disponible en Github. https://github.com/adafruit/DHT-sensor-library

[5] "Adafruit_BMP085_Unified". Librería para la serie de sensores BMP085/BMP180 de Presión Atmosférica. Disponible en Github. https://github.com/adafruit/Adafruit_BMP085_Unified

[6] "MQ135". Librería para el sensor MQ135 disponible en Github. https://github.com/GeorgK/MQ135

[7] Librería SoftwareSerial disponible en la web de Arduino. https://www.arduino.cc/en/Reference/SoftwareSerial

[8] "WifiEsp". Librería para el módulo ESP01 para proporcionar capacidades de red a placas Arduino disponible en Github. https://github.com/bportaluri/WiFiEsp

[9] Plugin flutter_bluetooth_serial para la implementación básica del Bluetooth disponible en: https://pub.dev/packages/flutter_bluetooth_serial

[10] Documentación Mosquitto. https://mosquitto.org/documentation/

[11] Documentación node-RED. https://nodered.org/docs/

[12] Documentación InfluxDB. https://docs.influxdata.com/influxdb/v2.0/

[13] Documentación Flux. https://docs.influxdata.com/influxdb/v2.0/query-data/get-started/

[14] Documentación Grafana. https://grafana.com/docs/

[15] NTP 742: Ventilación general de edificios. https://app.mapfre.com/documentacion/publico/en/catalogo_imagenes/grupo.do?path=1033667

[16] Enlace al Snapshot del panel Principal Diario. https://snapshot.raintank.io/dashboard/snapshot/1hmuxY7Q82bELb5aOYTiotPZG7Xma81p?orgId=2

[17] Enlace al Snapshot del panel Diario de Calidad del Aire. https://snapshot.raintank.io/dashboard/snapshot/03Dw634ErOsC4v35BIhRpYMtPWodO2UM?orgId=2

[18] Enlace al Snapshot del panel Semanal de Temperatura. https://snapshot.raintank.io/dashboard/snapshot/1g6X7kA9DtUuW5ciwN5Fq4lxa7SSftr5?orgId=2

[19] Enlace al Snapshot del panel Mensual de Humedad. https://snapshot.raintank.io/dashboard/snapshot/OzJhbvMPyeSF4X3RtkzyZ0uIGrqm0Er7?orgId=2

APÉNDICE

A1. ESQUEMA INTERCONEXIONES ARDUINO

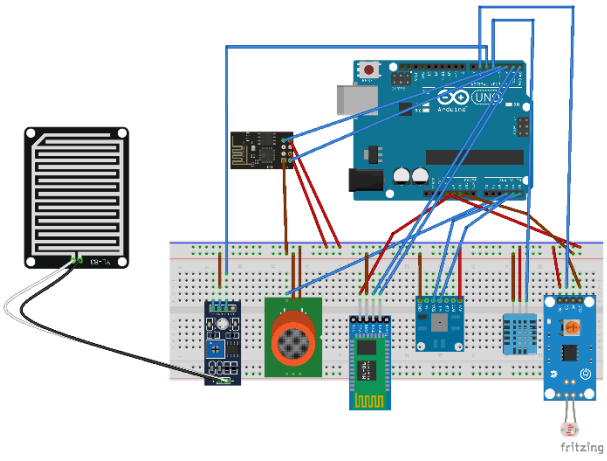

Fig. 22: Esquema general de conexiones.

A2. INTERFACES DE LA APLICACIÓN MÓVIL

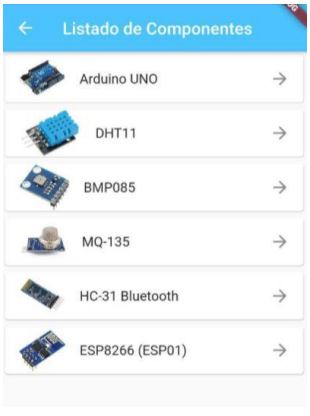

Fig. 23: Interfaz del Listado de Componentes.

Fig. 24: Interfaz detalles de componente.

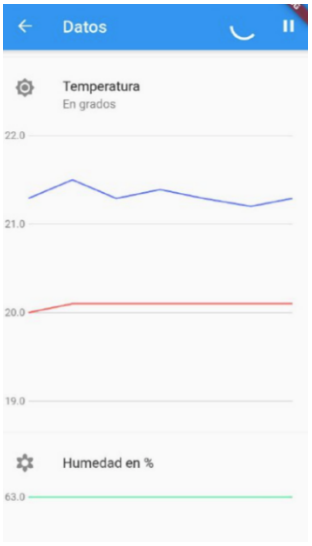

Fig. 25: Interfaz gráficas en tiempo real.

A3. QUERY EN FLUX

En la Figura 26 podemos observar un ejemplo de una query lanzada sobre la base de datos de series temporal para extraer datos posteriormente mostrados en Grafana.

El objetivo de esta Query es el de obtener todas las mediciones de las ppm de CO2 en los últimos 30 días. Como medimos cada dos minutos tenemos que utilizar la función aggregateWindow que nos permite quedarnos con la media de ventanas de 10 minutos. Esto lo realizamos debido a que si obtenemos todas las mediciones tenemos problemas de memoria, ya que Grafana no es capaz de graficar tantos puntos. Finalmente mapeamos los valores con el nombre "PPM de CO2" para que no tenga el nombre "value".

```
from(bucket:"Ppm_CO2")
  |> range(start:-30d)
  |> filter(fn:(r) =>
    r._measurement == "ppm_co2"
  )
  |> aggregateWindow(every: 10m, fn:mean)
  |> map(fn: (r) => ({
    r with name:
      if r._measurement == "ppm_co2" then "PPM de CO2"
    }))
```

Fig. 26: Query Calidad del Aire últimos 30 dias.

A4. Panel Semanal Detección Luz/Lluvia

Fig. 27: Query Calidad del Aire últimos 30 días.

A5. Elección del Microcontrolador

Inicialmente, para la realización del proyecto el microcontrolador escogido era Seeeduino XIAO. Este microcontrolador es compatible con el IDE de Arduino y está basado en el CPU-ARM Cortex -M0+(SAMD21G18) de 32 bits. Dispone de 14 pines de entrada y salida donde 11 son digitales y 11 pueden funcionar como entrada o salida analógica, 10 pines PWM, 1 salida DAC, 1 interfaz I2C, 1 interfaz UART y 1 interfaz SPI. Todos los pines de E/S son de 3,3V y se puede alimentar mediante USB de Tipo-C. Estas características lo hacían a priori un microcontrolador perfecto para este proyecto.

Se realizaron los primeros avances con Seeeduino XIAO, pero en la integración de los módulos de conectividad se encontraron ciertas limitaciones. La limitación en concreto hace referencia a la necesidad de comunicación Serial por parte de los dos módulos de conectividad del sistema. Seeeduino XIAO solo dispone de una interfaz UART lo que nos obligaba a utilizar una librería llamada SoftwareSerial.h comentada en el apartado de los componentes que permitía utilizar pines de I/O Digitales en puertos Serie. Esta librería no funcionaba correctamente con Seeeduino XIAO, lo que imposibilitaba tener los dos módulos de conectividad funcionando de manera simultánea. Tras una búsqueda exhaustiva por foros y documentación se llegó a la conclusión de que la librería trabajaba con un baud rate demasiado bajo lo que impedía la correcta comunicación.

Otra limitación importante estaba relacionada con la alimentación, ya que los pines de entrada no podían exceder los 3,3V mientras que en nuestro proyecto encontramos hasta dos componentes que se encuentran alimentados a 5 voltios y por lo tanto no se podrían conectar de forma adecuada al microcontrolador.

Por estos dos motivos, finalmente nos decantamos por utilizar Arduino UNO que era el microcontrolador utilizado en un principio para la realización de pruebas. Con Arduino UNO manteníamos una cantidad de pines suficientes tanto digitales como analógicos, y en el tema de los puertos Serie, aunque solo dispone de una interfaz UART como Seeeduino XIAO la librería SoftwareSerial.h sí que funcionaba de manera correcta y nos permitía emular puertos Serie en I/O Digitales. En cuanto a la alimentación también resolvía nuestro problema, ya que dispone de alimentación a 3,3V y a 5V permitiendo también la conexión de las salidas tanto a 3,3 como a 5V. Por estas dos razones finalmente se escogió como microcontrolador Arduino UNO.