UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE

ESCUELA POLITÉCNICA SUPERIOR DE ELCHE

GRADO EN INGENIERÍA ELECTRÓNICA Y AUTOMÁTICA INDUSTRIAL

DATALOGGER BASADO EN INTERNET OF THINGS DE BAJO COSTE

TRABAJO DE FIN DE GRADO

Septiembre - 2021

AUTOR: Santiago López Antón

DIRECTOR/ES: Javier Molina González

RESUMEN

Vivimos en un mundo conectado donde cada vez damos más importancia a los datos. De esta necesidad nace este proyecto motivado a dar una solución sencilla y barata para monitorizar y operar instalaciones térmicas desde cualquier lugar del planeta.

El crecimiento de los sistemas embebidos y las plataformas DIY (Do It Yourself) facilitan el desarrollo de multitud de aplicaciones que requieren una cierta capacidad computacional y de almacenamiento de datos a un coste reducido.

Evaluadas diferentes opciones de implementación en este proyecto, se pretende desarrollar un dispositivo que actúe como registrador de instalaciones térmicas, que envíe esta información a un servidor de base de datos que nos permita luego visualizar y monitorizar estas instalaciones desde cualquier lugar por medio de una aplicación web. También es necesaria la capacidad de actuación sobre la instalación, que por medio de un relé de activación puede conectarse y desconectarse remotamente.

ABSTRACT

We live in a connected world where each time data has more impact. From this need borns this project, motivated to give an easy and cheap solution to monitorize and operate thermal installations from each part of the world.

The growth of embedded systems and DIY (Do It Yourself) platforms ease the development of many applications that require a certain computational power and data storage at a low price.

Evaluating different options of implementation in this project, it is intended to develop a device that acts as thermal installation registrator, that sends the registered information to a database server to visualize and monitorize these installations from each part of the world using a web application afterwards. Moreover, there is the need to operate the installation using an activation relay it is possible to switch on or off remotely.

RESUM

Vivim en un món connectat on cada vegada donem més importància a les dades. D'esta necessitat naix aquest projecte motivat a donar una solució senzilla i barata per a monitoritzar i operar instal·lacions tèrmiques des de qualsevol lloc del planeta.

El creixement dels sistemes embeguts i les plataformes DIY (Do It Yourself) faciliten el desenvolupament de multitud d'aplicacions que requerixen una certa capacitat computacional i d'emmagatzemament de dades a un cost reduït.

Avaluades diferents opcions d'implementació en este projecte, es pretén desenvolupar un dispositiu que actue com a registrador d'instal·lacions tèrmiques, que envie aquesta informació a un servidor de base de dades que ens permeta després visualitzar i monitoritzar estes instal·lacions des de qualsevol lloc per mitjà d'una aplicació web. També és necessària la capacitat d'actuació sobre la instal·lació, que per mitjà d'un relé d'activació pot connectar-se i desconnectar-se remotament.

AGRADECIMIENTOS

Para mi familia, amigos y para mi chica, que tanto me han apoyado.

Para los que ya no están y a los que tanto debo agradecer.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

1.1. DESCRIPCIÓN Y JUSTIFICACIÓN DEL PROYECTO

Un registrador de datos se denomina técnicamente “Datalogger” y es un sistema que almacena datos recogidos a través de un sensor en un dispositivo de almacenamiento para su posterior gestión o procesado.

El propósito de este trabajo de fin de grado es la implementación de un datalogger con capacidad de registrar datos de magnitudes físicas, en nuestro caso de temperaturas, y su posterior gestión de forma remota para evitar la necesidad de operar con el dispositivo físicamente.

Los motivos que han llevado a realizar este proyecto son poner en práctica conocimientos aprendidos a lo largo de la carrera, y además adquirir conocimientos sobre desarrollo web.

1.2. OBJETIVOS Y REQUISITOS DEL PROYECTO

Como hemos descrito en el apartado anterior, se quería implementar un dispositivo capaz de registrar instalaciones de temperatura; por un lado, deberá adquirir datos de temperatura y enviarlos por medio de una conexión inalámbrica para su almacenamiento en un servidor de base de datos. A su vez implementar una aplicación web para visualizar y exportar estos datos de una manera síncrona o asíncrona. Basándonos en esto, hemos establecido los siguientes requisitos a cumplir por el dispositivo:

- **Almacenamiento de datos:** Se deben almacenar los datos obtenidos mediante el sensor de forma organizada y automatizada en un servidor accesible a la red para poder obtenerlos bajo demanda.
- **Configuración frecuencia:** El usuario debe de poder configurar cada cuanto tiempo se recoge una muestra.

- **Exportación de datos:** Los datos recogidos deben de poder ser almacenados en ficheros con formato ".csv", que son versátiles para el manejo de la información. Estos ficheros tendrán una estructura específica donde se indique la temperatura (ºC), el tiempo y fecha de registro, etc.
- **Representación de datos almacenados u importados:** El usuario debe poder visualizar en una gráfica los datos que se han almacenado y además importar los datos que desee visualizar desde un fichero con formato ".csv".
- **Posibilidad de gestión de forma remota:** Se debe proporcionar al usuario una página web desde la cual se pueda gestionar el dispositivo; se debe poder configurar el dispositivo registrador de la aplicación y, además, poder activar y desactivar la instalación cuando sea necesario.
- **Funcionamiento a largo plazo:** El dispositivo está pensado para que se quede en un lugar durante un periodo de tiempo prolongado, por lo que debemos poder conectarlo a una fuente de alimentación de forma continua.
- **Conexión a Internet:** Para poder gestionar de forma remota el dispositivo y almacenar los datos, el dispositivo debe de estar conectado a un router cercano por medio de conexión inalámbrica.

Teniendo en cuenta estos requisitos se propone una solución basada en tres sistemas independientes conectados entre sí: **dispositivo electrónico registrador, servidor de base de datos** y **aplicación web.**

Figura 1. Esquema aplicación (Fuente: propia)

Estos tres componentes integran la solución propuesta en nuestro proyecto "Datalogger basado en Internet of Things de bajo coste".

1.3. ESTRUCTURA DEL PROYECTO

Tomando como base los requisitos descritos en el apartado anterior para el funcionamiento del dispositivo, hemos decidido que el trabajo se va a componer de 7 capítulos, siendo el primero el de introducción. En el segundo se realizará un análisis entre las diferentes alternativas disponibles de hardware para la implementación.

En el tercer punto, hablaremos en detalle del hardware específico que utilizaremos, escogido según las conclusiones obtenidas en el análisis previo realizado en el segundo apartado.

El cuarto apartado tratará del software empleado, incluyendo las librerías utilizadas para cada pieza hardware y la programación necesaria para su funcionamiento. Explicaremos detalladamente la programación de las diferentes funciones realizadas para la adquisición y almacenamiento de datos.

Explicaremos todo el desarrollo de la aplicación web en el quinto apartado, así como las tecnologías utilizadas y en que nos hemos basado para su elección. Así como el desarrollo de todas las funcionalidades de representación de datos y de gestión.

En el siguiente apartado, el sexto, realizaremos una comparación con los dataloggers que existen actualmente en el mercado y un presupuesto.

Por último, en el séptimo punto, planteamos las conclusiones obtenidas en cuanto al resultado final, coste de la aplicación, eficiencia y de programación a los que hemos llegado.

2. ANÁLISIS DE ALTERNATIVAS PARA LA IMPLEMENTACIÓN

El objetivo del proyecto es conseguir un dispositivo datalogger que sustituya a los utilizados actualmente en la universidad para monitorizar instalaciones térmicas continuamente, por lo que hay dos puntos que son muy importantes a la hora de valorar el hardware a utilizar: consumo energético y coste. Tampoco queremos que sea un hardware sobredimensionado porque, dejando de lado que es una mala práctica, necesitamos un número reducido de entradas y salidas.

Uno de los requisitos indispensables del proyecto es que el dispositivo datalogger tenga conexión inalámbrica para poder enviar la información al servidor de base de datos que después conectará con la aplicación web que se usará de visualizador.

Otro requisito, aunque no indispensable, es que nuestro hardware sea compatible con una comunidad open source grande (Arduino, Raspberry Pi...) De esta manera se facilita al máximo el desarrollo obteniendo acceso a librerías mantenidas por estas comunidades.

Teniendo en cuenta estos requisitos se valoraron diferentes opciones que detallamos en los siguientes puntos.

2.1. ARDUINO

2.1.1. Plataforma Open Source Arduino

Arduino es una plataforma de prototipos electrónicos de código abierto basada en hardware y software flexibles y fáciles de utilizar. Esta compañía se dedica a la realización de placas de desarrollo de hardware y software compuestas por un microcontrolador y un entorno de desarrollo (IDE). Puede ser utilizado para desarrollar una ingente cantidad de proyectos interactivos los cuales pueden ser ejecutados desde la tarjeta Arduino o un entorno en un ordenador.

Figura 2. Logo Arduino (Fuente: Arduino)

Está basado en el uso de un microcontrolador, los cuales se caracterizan por el control y adquisición de datos a través de entradas y salidas analógicas/digitales.. Existen diferentes clases de microcontroladores de los cuales hablaremos posteriormente, que varían dependiendo de las necesidades del proyecto, además hay una gran variedad de fabricantes y versiones disponibles.

Los sistemas basados en microcontrolador tienen una gran cantidad de entradas y salidas donde podemos conectar, por ejemplo, sensores para el control de magnitudes físicas. También cuentan con áreas de prototipado para conexiones de otros dispositivos y la posibilidad de conectar otros módulos (shields) que amplían las características de funcionamiento de la placa, como por ejemplo el de almacenamiento en tarjetas SD, conexión Ethernet, Wi-Fi, etc.

El funcionamiento de Arduino se puede resumir en tres partes:

- **Interfaz de entrada:** Se encuentra directamente unida a los periféricos o conectada a ellos a través de puertos. Principalmente se encarga de llevar la información al microcontrolador.
- **Procesado de datos:** El microcontrolador se encarga de procesar los datos que se van adquiriendo.
- **Interfaz de salida:** Se encarga de llevar la información procesada a los periféricos encargados de hacer el uso final de dichos datos, ya sea a otra tarjeta Arduino o en la misma enviando la información a un dispositivo de salida como un motor, un led, un relé, etc.

La tarjeta Arduino está compuesta por diferentes puertos de entrada/salida, en las que cada una de ellas desarrolla una función específica:

- **Entradas:** Se trata de aquellos pines que se utilizan para captar las lecturas, pueden ser analógicas o digitales.
- **Salidas:** Son aquellos pines que se encargan de transmitir las señales que envía el dispositivo a los periféricos de salida.

- **Pines adicionales:** Se trata de los pines necesarios para la alimentación (Vin), los de conexión a tierra (GND), el voltaje que deseamos dar a nuestra placa (5V/3,3V), los necesarios para la conexión SPI (ICSP), los necesarios para la conexión I2C/TWI (SDA, SCL).

Para comunicarse con los dispositivos de E/S, las tarjetas electrónicas Arduino llevan incorporados pines que actúan como interfaces con el exterior bajo diferentes protocolos de comunicación serie. Con ella puede dejar de ser un chip aislado permitiéndole interactuar con cualquier dispositivo que también soporte una comunicación serie: ordenadores, sensores con conexión serie, equipos controlados por MIDI, smartphones, servidores u otros microcontroladores.

Dentro de las comunicaciones series podemos encontrar los siguientes protocolos:

- **UART (Universal Asynchronous Receiver-Transmitter):** Se trata de uno de los protocolos serie más utilizados. Utiliza una línea de datos para transmitir y otra para recibir los datos.
- **SPI (Serial Peripheral Interface):** Un maestro envía la señal de reloj, y tras cada pulso de reloj se envía un bit al esclavo y se recibe un bit de éste. Los nombres de las señales son SCK para el reloj, MOSI (Master Out Slave In) y MISO (Master In Slave Out). Para controlar más de un esclavo es preciso utilizar SS (Slave Select) que no es más que otra línea para seleccionar el esclavo.
- **I2C (Inter-Integrated Circuit):** Protocolo síncrono que utiliza solamente dos cables, uno para el reloj (SCL) y otro para el dato (SDA). Maestro y esclavo envían información por el mismo cable, el cual es controlado por el maestro, que crea la señal de reloj. I2C no utiliza SS sino direccionamiento.

2.1.2. Hardware oficial

Existen multitud de tarjetas electrónicas oficiales de Arduino. En esta sección se van a analizar brevemente las que por características se ajustan más a nuestro proyecto:

➔ **Arduino Nano 33 IoT**

La tarjeta de desarrollo Arduino Nano 33 IoT es la mejor y más barata forma de iniciarse en Arduino para aplicaciones IoT. El procesador principal de la tarjeta es un Arm® Cortex®-M0 32-bit SAMD21 de bajo consumo. La conectividad WiFi y Bluetooth la proporciona el módulo NINA-W10 de u-blox que es un chipset de bajo consumo que opera en la franja de 2.4 GHz.

Figura 3. Placa Arduino Nano 33 IoT (Fuente: Arduino)

Como principales características destacamos:

- Microcontrolador: SAMD21 Cortex®-M0+ 32bit low power ARM MCU.
- Módulo de conexiones inalámbricas: u-blox NINA-W102.
- Voltaje de circuito de operaciones: 3.3 VDC.
- Voltaje límite de alimentación: 21V.
- Corriente continua por pin: 7 mA.
- Velocidad de reloj: 48 MHz.
- Memoria flash interna de la CPU :256 KB.
- SRAM: 32KB.
- EEPROM: No.
- Número de entradas y salidas digitales: 8

- Número de pines PWM: 11.
- Puertos UART x1, SPI x1, I2C x1.
- Número de entradas analógicas: 8.
- Número de salidas analógicas: 1.
- Sensor inercial (IMU): LSM6DS3
- Tamaño: 45 x 18 mm.
- Peso: 5 g.

➔ **Arduino MKR WiFi 1010**

La MKR WiFi 1010 es la hermana mayor de la Nano 33 IoT, tienen los mismos chips principales y de comunicaciones. Su diferencia radica en un mayor número de puertos de E/S, no traer una IMU de serie y la posibilidad de cargar una batería Li-Po cuando está conectada por USB.

Esto último es muy interesante para muchos proyectos IoT ya que la propia placa gestiona cuando debe ser alimentada por la batería y cuando por el USB para cargar la batería. La gestión de la alimentación se hace totalmente automática. Aunque esta última característica es muy interesante el precio del dispositivo se encarece bastante.

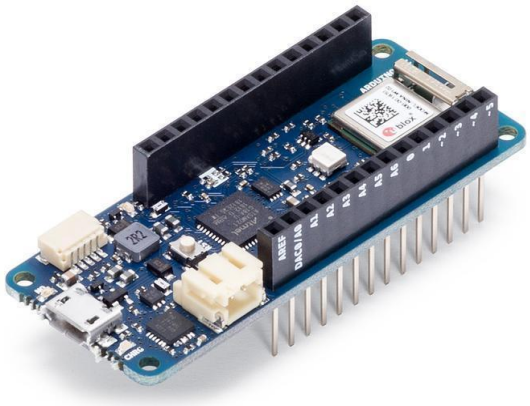

Figura 4. Placa Arduino MKR WiFi 1010 (Fuente: Arduino)

Entre sus principales características encontramos:

- Microcontrolador: SAMD21 Cortex®-M0+ 32bit low power ARM MCU.
- Módulo de conexiones inalámbricas: u-blox NINA-W102.
- Alimentación del sistema: 5 VDC.
- Batería soportada: Li-Po de una celda, 3.7 VDC, 1024 mAh de corriente mínima.
- Voltaje de circuito de operaciones: 3.3 VDC
- Número de entradas y salidas digitales: 8
- Número de pines PWM: 13.
- Puertos UART x1, SPI x1, I2C x1.
- Número de entradas analógicas: 7.
- Número de salidas analógicas: 1.
- Corriente continua por pin: 7 mA.
- Memoria flash interna de la CPU :256 KB.
- SRAM: 32 KB.
- EEPROM: No.
- Velocidad de reloj: 32.768 kHz (RTC), 48 MHz.
- Tamaño: 61.5 x 25 mm.
- Peso: 32 g.

Una de las pegas de las placas oficiales de Arduino es que tienen un coste superior a otras que son compatibles con el proyecto open source: en el caso de la Arduino Nano 33 IoT 16 € y en el caso de la Arduino MKR Wifi 1010 27,9 €. Esto puede deberse a que mantienen todo un ecosistema open source de microcontroladores que también incluye un IDE y a la calidad de los componentes.

Otra de las pegas de estas placas es que normalmente incluyen sensores inerciales (IMUs) que encarecen el dispositivo. Estos sensores son muy útiles para otros tipos de aplicaciones, pero no nos sirven para nada en nuestro proyecto por lo que sería ideal encontrar un dispositivo que carezca de este sensor.

Pese a que son una muy buena opción, buscamos minimizar costes y no sobredimensionar el hardware por lo que finalmente nos decantamos por otra opción.

2.1.3. Hardware compatible con Arduino

La comunidad de usuarios de Arduino es una de las principales bazas a la hora de decantarnos por un hardware compatible con Arduino. Esta comunidad mantiene multitud de librerías que facilitan el desarrollo de cualquier sistema basado en microcontrolador. Una de estas plataformas compatibles con Arduino es la basada en el microcontrolador ESP8266.

La plataforma ESP8266 permite el desarrollo de aplicaciones en diferentes lenguajes como: Arduino, Lua, MicroPython, C/C++, Scratch. Al trabajar dentro del entorno Arduino podremos utilizar un lenguaje de programación conocido y hacer uso de un IDE sencillo de utilizar, además de hacer uso de toda la información sobre proyectos y librerías disponibles en internet.

El SoC (System On a Chip) **ESP8266** de Espressif Systems es un chip de bajo coste especialmente diseñado para las necesidades de un mundo conectado, integra un potente microcontrolador con arquitectura de 32 bits (más potente que el Arduino Due) y conectividad Wi-Fi. El SoM (System on Module) ESP-12E fabricado por Ai-Thinker integra en un módulo el SoC ESP8266, memoria FLASH, cristal oscilador y antena WiFi en PCB.

Las características más importantes de este microcontrolador son las siguientes:

- CPU RISC de 32-bit: Tensilica Xtensa LX106 a un reloj de 80 MHz.
- RAM de instrucción de 64 KB, RAM de datos de 96 KB.
- Capacidad de memoria externa flash QSPI - 512 KB a 4 MB* (puede soportar hasta 16 MB).
- IEEE 802.11 b/g/n Wi-Fi.
    - Tiene integrados: TR switch, balun, LNA, amplificador de potencia de RF y una red de adaptación de impedancias.
    - Soporte de autenticación WEP y WPA/WPA2.
- 16 pines GPIO (Entradas/Salidas de propósito general).
- SPI, I²C.
- Interfaz I²S con DMA (comparte pines con GPIO).

- Pines dedicados a UART, más una UART únicamente para transmisión que puede habilitarse a través del pin GPIO2.
- 1 conversor ADC de 10-bit.

Esta opción es por la que nos decantamos ya que nos proporciona el hardware más ajustado a nuestro proyecto.

2.2. RASPBERRY PI

Desde sus primeras versiones, este mini PC de bajo coste se ha convertido en uno de los principales actores del mundo de los sistemas embebidos. Es un sistema basado en microprocesador con un sistema operativo linux preinstalado.

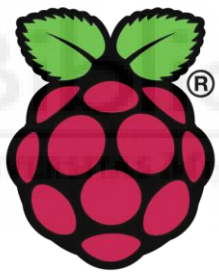

Figura 5. Logo Raspberry Pi (Fuente: Raspberry Pi)

Su última versión la Raspberry Pi 4 model B tiene las siguientes caracteristicas:

- **SoC:** Broadcom BCM2711.
- **CPU:** Procesador de cuatro núcleos a 1,5 GHz Cortex-A72.
- **GPU:** VideoCore VI.
- **Memoria:** 1/2/4GB LPDDR4 RAM.
- **Conectividad:** 802.11ac Wi-Fi / Bluetooth 5.0 / Gigabit Ethernet.

- **Vídeo y sonido:** 2 x puertos micro-HDMI que admiten pantallas de 4K@60Hz a través de HDMI 2.0, puerto de pantalla MIPI DSI, puerto de cámara MIPI CSI, salida estéreo de 4 polos y puerto de vídeo compuesto.
- **Puertos:** 2 x USB 3.0, 2 x USB 2.0.
- **Alimentación:** 5V/3A vía USB-C, 5V vía cabezal GPIO.
- **Expansión:** Cabezal GPIO de 40 pines.

Pese a que podría ser útil para nuestro proyecto es una opción más cara (el precio oscila desde unos 45 € de la versión con 2 GB de RAM hasta los 85 € de la versión con 8GB de RAM) que la que finalmente elegimos.

Pese a ello, podría ser una buena opción si en algún momento se quieren integrar servidor de base de datos, datalogger y visualizador en un sólo dispositivo. Sin embargo, si el hardware sólo se va a utilizar para registrar la temperatura esta opción no resulta tan interesante puesto que sus especificaciones (número de puertos de comunicación, pines…) son excesivas para los requisitos que afrontamos este proyecto.

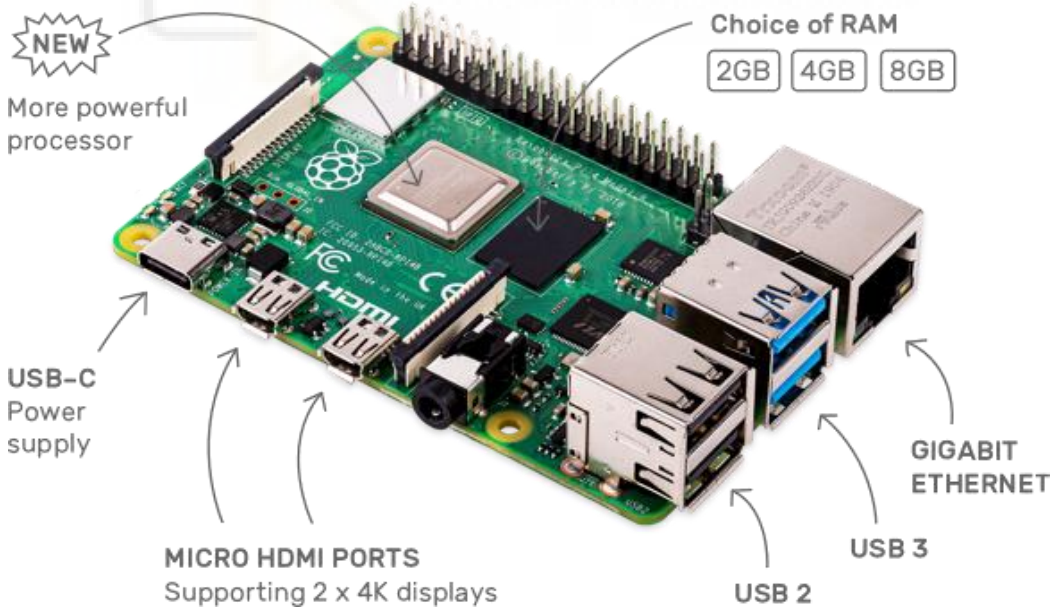

Figura 6. Placa Raspberry Pi (Fuente: Rapsberry Pi)

3. HARDWARE DEL PROYECTO

Tras analizar las diferentes opciones decidimos optar por hardware compatible con la plataforma ESP8266 por diferentes motivos:

- **Coste:** el coste de estos dispositivos es sensiblemente inferior al del hardware oficial Arduino.
- **Compatibilidad total con Arduino:** la plataforma ESP8266 tiene una compatibilidad total con Arduino.
- **Comunidad:** la comunidad ESP8266 es una de las más grandes dentro del mundo de los microcontroladores y hay infinidad de librerías mantenidas por esta comunidad que facilitan los desarrollos.
- **Adecuación del hardware al proyecto:** el hardware es el que más se adecua al proyecto por el número de E/S y consumo energético.

3.1. NodeMCU

NodeMCU ESP8266 es una plataforma de desarrollo similar a Arduino especialmente orientada al Internet de las cosas (IoT). La placa NodeMcu v2 ESP8266 tiene como núcleo al SoM ESP-12E que a su vez está basado en el SoC Wi-Fi ESP8266, integra además el conversor USB-Serial TTL CP2102 y conector micro-USB necesario para la programación y comunicación a PC.

NodeMcu v2 ESP8266 está diseñado especialmente para trabajar montado en protoboard o soldado sobre una placa. Posee un regulador de voltaje de 3.3V en placa, esto permite alimentar la placa directamente del puerto micro-USB o por los pines 5V y GND. Los pines de E/S (GPIO) trabajan a 3.3V.

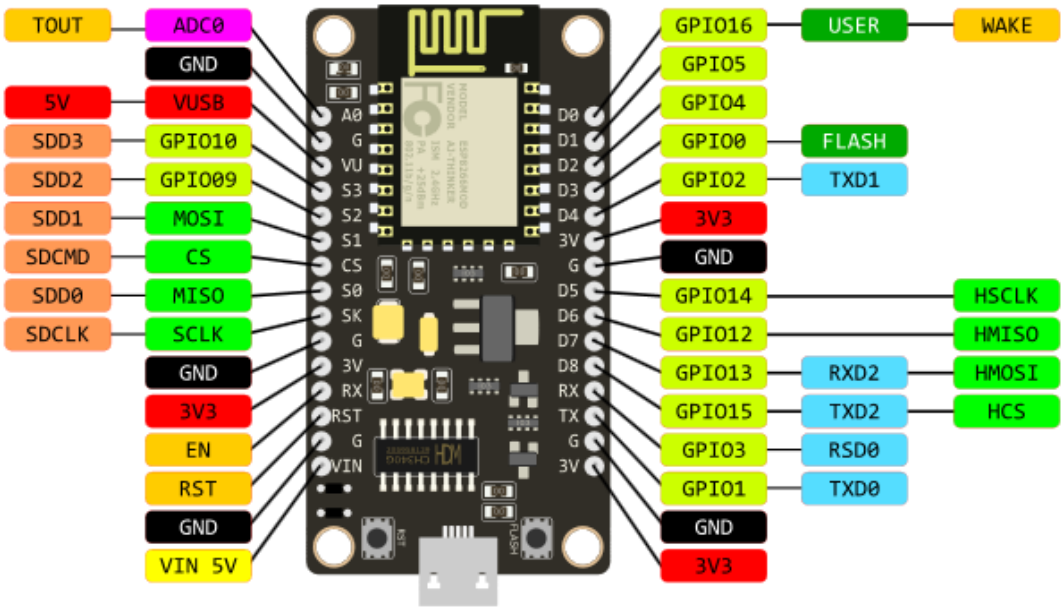

Figura 7. Placa NodeMCU

NodeMCU viene con un firmware preinstalado el cual nos permite trabajar con el lenguaje interpretado LUA, enviando comandos mediante el puerto serial (CP2102). Las tarjetas NodeMCU y Wemos D1 mini son las plataformas más usadas en proyectos de Internet de las cosas (IoT). No compite con Arduino, pues cubre objetivos distintos.

Las principales características de esta placa son:

- Voltaje de Alimentación: 5 VDC.
- Voltaje de Entradas/Salidas: 3.3 VDC.
- Chip conversor USB-serial: CP2102.
- SoM: ESP-12E (Ai-Thinker).
- SoC: ESP8266 (Espressif).
- CPU: Tensilica Xtensa LX3 (32 bit).
- Frecuencia de Reloj: 80MHz/160MHz.
- Instruction RAM: 32KB.
- Data RAM: 96KB.
- Memoria Flash Externa: 4MB.
- Pines Digitales GPIO: 17 (4 pueden configurarse como PWM a 3.3V).

- Pin Analógico ADC: 1 (0-1V).
- Puerto Serie UART: 2.
- Wireless 802.11 b/g/n.
- Stack de Protocolo TCP/IP integrado
- Consumo de potencia Standby < 1.0mW.
- Dimensiones: 49*26*12 mm
- Peso: 9 gramos

3.2. WeMos D1 Mini

WeMos D1 mini ESP8266 es una plataforma de desarrollo similar a Arduino especialmente orientada al Internet de las cosas (IoT). La placa Wemos D1 Mini ESP8266 tiene como núcleo al SoM ESP-12E que a su vez está basado en el SoC Wi-Fi ESP8266, integra además el conversor USB-Serial TTL CH340G y conector micro-USB necesario para la programación y comunicación a PC.

Wemos D1 mini está diseñado especialmente para trabajar montado en protoboard o soldado sobre una placa. Posee un regulador de voltaje de 3.3V en placa, esto permite alimentar la placa directamente del puerto micro-USB o por los pines 5V y GND. Los pines de E/S (GPIO) trabajan a 3.3V.

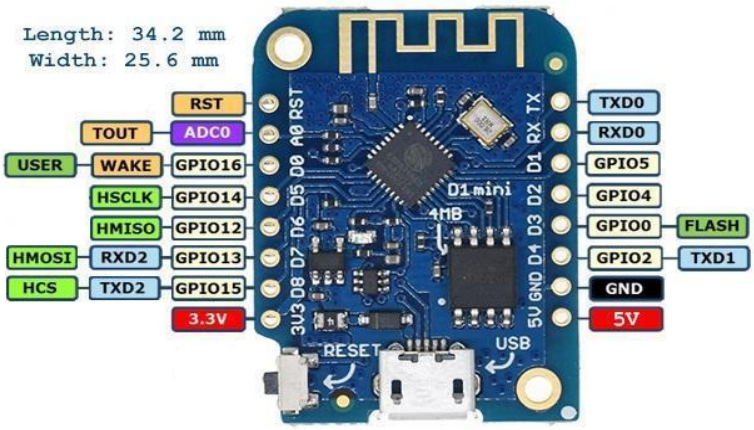

Figura 8. Placa Wemos D1 Mini

- Voltaje de Alimentación: 5 VDC.
- Voltaje de Entradas/Salidas: 3.3 VDC.
- Chip conversor USB-serial: CH340G
- SoM: ESP-12E (Ai-Thinker).
- SoC: ESP8266 (Espressif).
- CPU: Tensilica Xtensa LX3 (32 bit).
- Frecuencia de Reloj: 80MHz/160MHz.
- Instruction RAM: 32KB.
- Data RAM: 96KB.
- Memoria Flash Externa: 4MB.
- Pines Digitales GPIO: 11 (3.3V).
- Pin Analógico ADC: 1 (0-1V).
- Puerto serial UART: 1 (3.3V).
- Corriente Standby: 40uA.
- Corriente Pico: 400mA.
- Consumo corriente promedio: 70mA.
- Consumo de potencia Standby < 1.0mW.
- Dimensiones: 35*26*12 mm.
- Peso: 6 gramos.

Como podemos ver tanto la WeMos D1 Mino como la NodeMCU son placas de desarrollo muy parecidas. La única diferencia notable es el número de puertos del que dispone cada placa lo que hace a la WeMos una placa más pequeña y con menores posibilidades de comunicación al sólo disponer de un puerto UART. Existe también una diferencia de precio provocada por las diferencias en el número de E/S.

En el proyecto hemos usado ambos dispositivos indiferentemente puesto que disponíamos de ambos en el departamento. Lo único que hay que adecuar en la programación son las entradas y salidas del dispositivo para su correcta comunicación tanto con el conversor MAX6675 como con el relé.

3.3. Termopar y conversor

3.3.1. Teoría de funcionamiento del termopar

Un termopar es un dispositivo formado por la unión de dos metales distintos que produce un voltaje (efecto Seebeck) en función de la diferencia de temperatura entre uno de los extremos denominado “punto caliente” o unión caliente o de medida y el otro denominado “punto frío” o unión fría o de referencia.

Este tipo de sensores son ampliamente utilizados en aplicaciones de instrumentación industrial debido principalmente a su bajo costo y su amplio rango de temperaturas. La principal desventaja de los termopares es su exactitud, ya que rara vez se consiguen errores inferiores a 1 grado celsius.

3.3.2. Conversor digital de termopar tipo K “MAX6675”

Para el registro de temperaturas necesitamos convertir la señal analógica que nos facilita el termopar a una señal digital que nuestro microcontrolador pueda entender. Para esta tarea hemos elegido el conversor digital MAX6675.

El conversor es capaz de darnos la temperatura con una resolución de 0,25 ºC y permite medir temperaturas entre 0 y 1024 ºC.

Figura 9. Conversor termopar

Las características del conversor son las siguientes:

- Conversión digital directa de termopar tipo K.
- Compensación de unión fría.
- Interfaz serial simple compatible con SPI.
- Resolución de 12 bits, resolución de temperatura 0,25 ° C.
- Detección de termopar abierto.
- Entradas diferenciales de alta impedancia.
- Voltaje de operación 3-5 V. Corriente de operación 50 mA.
- Temperatura de operación: -20 º C - 85 º C.
- Tamaño: 15mm x 25 mm.

Antes de convertir los voltajes termoeléctricos en valores de temperatura, es necesario compensar la diferencia entre la parte de la unión fría del termopar (temperatura ambiente del MAX6675) y una referencia virtual de 0 ºC. Para un termopar de tipo K, el voltaje varía en 41µV/°C, lo que nos lleva a aproximar las características del termopar con la siguiente ecuación lineal:

Vo = (41µV / °C) * (Tr - Tamb)

Donde Vo es el voltaje de salida del termopar en µV, Tr es la temperatura de la unión del termopar ºC y Tamb es la temperatura ambiente en ºC.

3.4. Tarjeta electrónica de relés “2PH63083A”

Para poder activar la instalación remotamente necesitamos un relé que nos permita actuar sobre la alimentación de la instalación térmica. Para esto disponemos de la placa 2PH63083A.

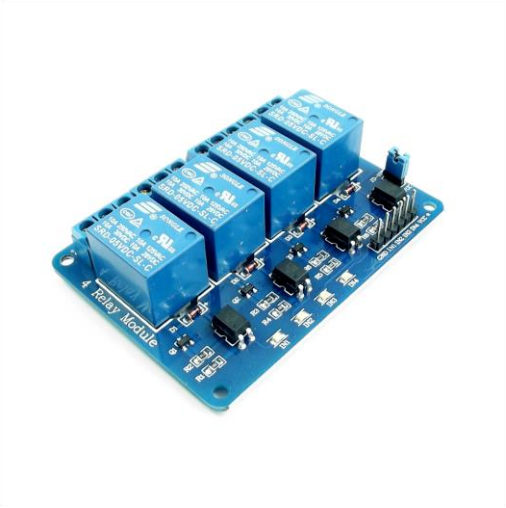

Figura 10. Tarjeta electrónica de relés

Las características de esta placa son las siguientes:

- Voltaje de Operación: 5V DC.
- Señal de Control: TTL (3.3V o 5V).
- N.º de Relés: 4 Canales.
- Modelo Relé: SRD-05VDC-SL-C.
- Capacidad máxima: 10A/250VAC, 10A/30VDC.
- Corriente máxima: 10A (NO), 5A (NC).
- Tiempo de acción: 10 ms / 5 ms.
- Para activar salida NO: 0 Voltios.
- Entradas Optoacopladas.
- Indicadores LED de activación.

Este tipo de relé requiere una señal de nivel bajo para activar el circuito de potencia, es decir, necesita 0 voltios en el circuito de control para poder activar la instalación. Esto se conoce como activación a nivel bajo de tensión.

Aunque la placa dispone de 4 relés, no necesitaremos todos para nuestro proyecto ya que con sólo uno de ellos podremos activar el circuito de potencia de la instalación.

4. DESARROLLO DEL PROGRAMA DE REGISTRO DE TEMPERATURA

4.1. CONFIGURACIÓN DEL ENTORNO ARDUINO IDE

Para poder desarrollar el proyecto con el hardware elegido se debe configurar el entorno de programación para poder programar nuestras placas electrónicas.

Por un lado deberemos instalar en el gestor de tarjetas el módulo correspondiente al ESP8266 y por otro lado deberemos instalar las librerías que vayamos a utilizar: en este caso la del conversor, otra para poder almacenar los valores en la base de datos y otra que nos permita obtener la fecha y hora de un servidor NTP para poder programar la activación del relé.

4.1.1. Gestor de tarjetas

Como no utilizamos un hardware oficial Arduino hay que añadirlo manualmente a nuestro IDE para que sea posible programar nuestro dispositivo.

Para esto hay que añadir en Archivo => Preferencias => Gestor de URLs Adicionales de Tarjetas el siguiente enlace:

http://arduino.esp8266.com/stable/package_esp8266com_index.json

Con esto conseguimos que en Herramientas => Placa => Gestor de tarjetas aparezca el paquete ESP8266 que nos habilita para programar multitud de dispositivos que incluyen este SoC.

Una vez añadido seleccionaremos la tarjeta a programar en Herramientas => Placa => ESP8266 boards. Con esto tendremos Arduino configurado para nuestra placa de desarrollo.

4.1.2. Instalación de librerías

Con el objetivo de poder acceder a diferentes funciones que nos habilitan por ejemplo a leer el sensor de temperatura, o a insertar y/o consultar una entrada en la base de datos, es necesaria la instalación de ciertas librerías que nos permitan realizar estas funciones.

- **Librería MAX6675:** esta librería nos permite leer nuestro termopar comunicando directamente con el conversor que realiza una conversión directa de los voltajes leídos a valores de temperatura.
- **Librería MySQL:** esta librería aparte de permitirnos inserciones o consultas en la base de datos, también nos permite crear tablas.
- **Librería NTPClient:** este servidor de hora nos permite consultar la fecha y hora a un servidor. Será usada para la activación programada de la instalación por medio del relé.

Para la instalación de estas librerías debemos ir a Herramientas => Administrar Bibliotecas... en Arduino, buscarlas e instalarlas.

Con esto finalizamos la configuración de Arduino IDE para la realización del proyecto y ya se puede proceder con la programación del dispositivo.

4.2. CONEXIÓN A RED VÍA WIFI

La conexión a la red se realiza con la ayuda de la librería <ESP8266WiFi.h>. Utilizando esta librería usamos la función *WiFi.begin(ssid,password)*, que nos permite conectar con cualquier red inalámbrica facilitando el nombre de la misma y la contraseña asociada a esta.

De esta manera conseguimos que nuestro dispositivo esté conectado a la red y pueda facilitar datos a nuestra base de datos en tiempo real que serán luego leídos por la aplicación web para su correcta representación.

```
void SetupWifi()

{
```

```
    WiFi.mode(WIFI_STA);
    WiFi.disconnect();
    delay(100);
    Serial.println("Setup done");
    WiFi.begin(wifiSSID, wifiPassword);
     while (WiFi.status() != WL_CONNECTED)
      {
            delay(500);
            Serial.print(".");
    }
    Serial.println("\nConnected");
    delay(2000);
}
```

Código 1. Conexión al punto de acceso inalámbrico.

4.3. REGISTRO DE TEMPERATURA Y ALMACENAMIENTO EN BASE DE DATOS

Para poder registrar la temperatura de la instalación utilizando el termopar y el conversor necesitamos obtener comunicación con la base de datos, y si no están creadas las tablas, crearlas.

Lo primero se realiza con la función *connect(server_ip, 3306, user, password)* del objeto MySQL_Connection de la librería MySQL. A esta función se le pasa la IP del servidor/equipo en el que se encuentra instalada nuestra base de datos, el puerto de la conexión, el usuario de la base de datos y la contraseña. La IP del servidor/equipo se puede obtener de un servidor DNS con la función hostByName().

```
void SetupDatabase()
{
        WiFi.hostByName(hostname,server_ip);
        Serial.println(server_ip);
        Serial.println("Connecting...");
         if (conn.connect(server_ip, 3306, user, password))
                delay(1000);
        else
                Serial.println("Connection failed.");
        cur_mem = new MySQL_Cursor(&conn);
        cur_mem->execute(CREATE_TEMPERATURE_TABLE);
         cur_mem->execute(CREATE_CONFIGURATION_TABLE);
}
```

Acto seguido se procede a la creación de las tablas necesarias para el registro de temperatura y para la configuración. Se realiza con la función *execute("query")* del puntero al objeto MySQL_Cursor. Esta función simplemente ejecuta una *query* que se encarga de la creación de las tablas de la base de datos. Las dos tablas que se procede a crear son: la **tabla de registro de temperatura** y la **tabla de configuración**.

- La **tabla de temperatura** sirve para las consultas de inserción entre el dispositivo electrónico y la base de datos, de tal manera que el microcontrolador insertará un registro cada vez que obtenga la temperatura. Esta tabla también servirá para las consultas de lectura por la aplicación web para la obtención de una gráfica temporal de registro de temperaturas. La tabla tiene los siguientes datos:

	Temperatura	**Fecha**	**Sensor**

Tipo de dato	Decimal	Timestamp	Entero
Explicación	Dato de temperatura de la instalación.	Fecha asociada al muestreo de la temperatura.	Identificador de sensor que ha registrado esta entrada.

Tabla 1. Estructura tabla temperatura

- La **tabla de configuración** sirve para las consultas de actualización entre la aplicación web y la base de datos, es decir, cada vez que el usuario cambie algo de la configuración se actualizará en la base de datos. Esta tabla también servirá para las consultas de lectura que el microcontrolador realizará para la activación de la instalación programada o no programada y para obtener la velocidad de muestreo de la temperatura.

	Tiempo de muestreo	**Relé activado**	**Fecha de inicio**	**Fecha de fin**
Tipo de dato	Entero	Entero	Timestamp	Timestamp
Explicación	Velocidad de muestreo de temperatura.	Indica si la instalación está activa.	Sirve para la activación programada de la instalación. Indica cuándo será activada.	Sirve para la activación programada de la instalación. Indica cuándo será desactivada.

Tabla 2. Estructura tabla configuración

De esta manera la base de datos servirá de operador intermedio entre nuestro dispositivo registrador de temperatura y la aplicación web que nos ofrecerá la información de manera gráfica.

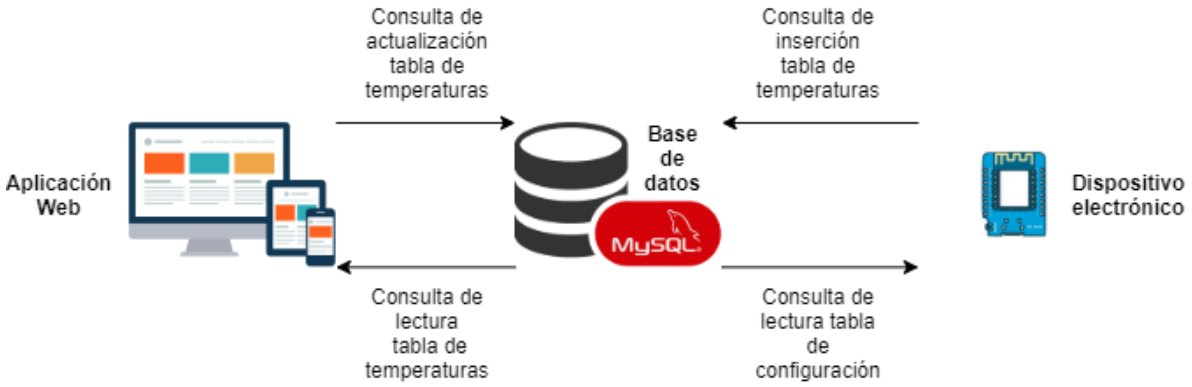

Figura 11. Esquema estructura de nuestro sistema (Fuente: propia)

Una vez creadas las tablas, se chequea la tabla de configuración para comprobar el tiempo de muestreo y si la instalación ha de estar activa o no. Si la instalación está activada se procede a leer el conversor del termopar por medio de la librería MAX6675 con la función *readCelsius* del objeto MAX6675.

```
if (activeInstallation)
{
    ReadTermoCouple();
    sprintf(query, INSERT_SQL, temperature, 1);
    cur_mem->execute(query);
}
```

Una vez obtenida la temperatura se procede a realizar un INSERT ejecutando una query con la función *execute* antes mencionada. De esta manera se tendría un registro en la base de datos que nos servirá más adelante para representar las temperaturas con respecto al tiempo en la aplicación web.

4.4. ACTIVACIÓN DE LA INSTALACIÓN REMOTAMENTE

En este apartado abordaremos cómo resolvimos el problema de actuar (activación y desactivación de la instalación) sobre la instalación de manera remota. Se pretendía que fuera posible la actuación sobre la instalación de dos maneras diferentes: directamente o programando su activación.

4.4.1. Activación por operación directa

Como ya hemos comentado la aplicación web y el dispositivo se comunican a través de la base de datos. En la tabla de configuración disponemos de una entrada que indica al dispositivo si debe activar o desactivar el relé que habilita la activación de la instalación. El dispositivo consulta si debe activar la instalación, y si encuentra un 1 en la base de datos "activa" la salida correspondiente que activa la instalación.

Esto permite la operación directa de la instalación desde cualquier lugar del mundo simplemente activando un botón en la aplicación web. El dispositivo en su bucle de operación lee esta entrada para activar o desactivar el relé de activación de la instalación. De esta manera conseguimos una actuación directa sobre la instalación.

4.4.2. Activación programada

Para la activación programada de la instalación es necesario la *"puesta en hora del dispositivo electrónico".* Esto se realiza conectando el dispositivo con un servidor NTP que nos facilita la hora en el formato que queramos.

Desde la aplicación web es posible programar la activación y desactivación de la instalación de tal manera que nuestro dispositivo lee esta entrada y la traduce para poder saber si el relé debe estar apagado o encendido.

Una vez leída la hora se guarda en una variable que se consulta en cada ejecución del bucle de operación de nuestro microcontrolador y si estamos dentro de la franja horaria

seleccionada por el usuario de la aplicación web, se activa el relé. El relé continúa activado hasta que llega la hora de desactivación, que es guardada en otra variable para su consulta en el bucle de operación.

Esta opción permite la automatización horaria de la instalación de una manera remota y totalmente monitorizada.

5. DESARROLLO DE APLICACIÓN WEB

5.1. ANÁLISIS DE LENGUAJES DE PROGRAMACIÓN WEB

Para el desarrollo de la aplicación web queríamos trabajar con el lenguaje de programación JavaScript, por los que analizamos diferentes frameworks para ver cuál se adapta mejor a nuestras necesidades.

Un framework ofrece a los desarrolladores la base necesaria para construir aplicaciones. Esto ahorra a los desarrolladores el esfuerzo de comenzar desde cero utilizando una base funcional para poner las cosas en marcha. Para JavaScript, esta base incluye una colección de librerías que generan funcionalidades específicas. En resumen, el framework definirá toda la estructura de la aplicación. Ahora vamos a analizar los cinco frameworks de JavaScript más populares.

5.1.1. Node.js

Figura 12. Logo Node.js (Fuente: Node.js)

Node.js no es exactamente un framework, es un entorno en tiempo de ejecución JavaScript del lado del servidor, que es de código abierto y funciona en plataformas cruzadas. Aunque JavaScript normalmente opera en el lado de cliente, front-end, los scripts del lado del servidor proporcionan mejores tiempos de carga ya que no se requiere la tecnología del navegador, y muestra propiedades similares de JAVA como subprocesamiento, empaquetado o formación de bucles. Características:

- **Velocidad:** La librería de node.js es rápida en cuanto a ejecución de código ya que se basa en el motor JavaScript V8 de Google Chrome.
- **E/S asíncrona y controlada por eventos:** Todas las API son asíncronas, esto quiere decir que el servidor no espera a que la API devuelva datos. Es decir, el servidor llama a las API una a una y va pasando a la siguiente mientras utiliza un mecanismo de notificación de eventos para generar una respuesta a la llamada de la API previa.
- **Un solo hilo:** Node.js, junto con el bucle de eventos sigue un modelo de un solo hilo.
- **Escalable:** Al seguir un mecanismo de eventos esto hace posible que el servidor responda de forma no bloqueante, lo que lo hace escalable.
- **Sin búfer:** Al cargar archivos de audio o video, Node.js reduce significativamente el tiempo de procesamiento. No almacena ningún dato en el búfer y la aplicación extrae los datos en trozos.
- **Código abierto:** Como es de código abierto, la comunidad de Node.js ha creado varios modelos que se pueden utilizar para añadir mejores capacidades a sus aplicaciones.

5.1.2. Vue.js

Figura 13. Logo Vue.js (Fuente: Vue.js)

Vue se llama a sí mismo el framework de JavaScript "progresivo". Esto se debe a su filosofía de adopción incremental. En Vue, la librería principal se centra solo en la capa de vista, por lo que cualquier funcionalidad adicional debe añadirse en incrementos.

El framework utiliza una arquitectura "*model-view-viewmodel*" (MVVM). Este patrón separa la interfaz gráfica de la aplicación (UI), que es la vista, de la lógica empresarial, el modelo. La capa de modelo vista "*viewmodel*" es un medio de conversión que sincroniza datos.

Su modo de integración dual es una de las características más atractivas para crear aplicaciones de una sola página (SPA). Es una plataforma confiable para desarrollar multiplataforma. Algunas de sus características:

- **DOM virtual:** El DOM virtual es un clon del DOM principal, y el virtual absorbe todos los cambios previstos para el DOM que se presentan en forma de estructuras de datos JavaScript, que se comparan con la estructura de datos original.
- Los usuarios ven los cambios finales que se reflejan en el DOM real. Es un método creativo y rentable y los cambios se realizan rápidamente.
- **Enlace de datos:** Esta función facilita manipular o asignar valores a los atributos HTML, cambiar el estilo y asignar clases con v-bind, que es una directiva de enlace.
- **Transiciones o animaciones CSS:** Esta función proporciona métodos para aplicar transiciones a elementos HTML, cuando se añaden, actualizan o eliminan del DOM. El funcionamiento consiste en un componente incorporado que envuelve el elemento encargado de devolver el efecto de transición.
- **Plantilla:** Proporciona plantillas basadas en HTML que vinculan el DOM con los datos de la instancia de Vue.js. Las plantillas se compilan en funciones de renderizado del DOM virtual.
- **Métodos:** Usamos métodos cuando se produce un evento que no está relacionado con la mutación de los datos de la instancia o con el deseo de cambiar el estado de un componente. Los métodos no mantienen registros de ninguna dependencia, pero pueden aceptar argumentos.
- **Complejidad:** Vue es simple en términos de API y diseño.

5.1.3. AngularJS

Figura 14. Logo AngularJS (Fuente: AngularJS)

Angular es mantenido por Google y aborda las complicaciones comunes en la creación de aplicaciones de una sola página (SPA). Este framework funciona aprovechando el vocabulario HTML en páginas web dinámicas. En el pasado, HTML únicamente se podía utilizar para contenido estático.

Las SPA (Single Page Applications) funcionan cargando dinámicamente contenido desde el servidor web en lugar de desde el navegador web. Por tanto, funcionan de manera similar a las aplicaciones móviles y no necesitan recargarse. Propiedades:

- **Multiplataforma:** Se puede utilizar para desarrollar aplicaciones web progresivas, ofrece una experiencia similar a la de una aplicación de alto rendimiento, sin conexión y con una instalación sin pasos. Para desarrollar aplicaciones nativas utilizando Cordova, Ionic o NativeScript. Y también para desarrollar aplicaciones de escritorio, estas se desarrollan utilizando los mismos métodos de Angular que las aplicaciones web, añadiendo la capacidad de acceder a las API nativas del sistema operativo.
- **Velocidad y rendimiento:** Angular convierte las plantillas en código optimizado para máquinas virtuales JavaScript, lo que ofrece beneficios en el código escrito a mano. Es universal, permite renderizar de forma instantánea Node.js, PHP, .NET y otros servidores en HTML y CSS. También ofrece división de código para poder cargar únicamente aquello que los usuarios necesitan.
- **Productividad:** Permite crear interfaces gráficas con una sintaxis sencilla y potente. Además ofrece herramientas de línea de comandos que permiten compilar rápidamente, agregar componentes, hacer pruebas e implementar código

instantáneamente. En los editores de código (IDEs) ofrece corrección de errores y autocompletado.

- **Pruebas:** Permite ejecutar pruebas de forma rápida y estable.
- **Animación:** Se pueden crear coreografías complejas de alto rendimiento y líneas de tiempo de animación con muy pocas líneas de código gracias a la API de Angular.
- **Accesibilidad:** Favorece la creación de aplicaciones accesibles con componentes habilitados para ARIA, guías para desarrolladores e infraestructura de prueba incorporada.

5.1.4. Ember.js

Figura 15. Logo Ember.js (Fuente: Ember.js)

Ember es un framework de JavaScript que emplea un patrón de servicio de componentes. A diferencia de la arquitectura tradicional modelo-vista-controlador (MVC), los componentes en Ember son fundamentales para el framework. Casi todo se puede clasificar como servicio o componente. Los componentes son transitorios y modifican el lenguaje de marcado y los estilos de la interfaz de usuario de una aplicación. Los servicios son objetos que viven mientras dure la aplicación. Pueden estar disponibles para diferentes partes de las aplicaciones y se utilizan mejor para estados persistentes. Admite enlace de datos de forma bidireccional, por lo que son una plataforma confiable para interfaces de usuario complicadas. Netflix, LinkedIn, Nordstrom usan Ember para sus sitios web. Características:

- **Usabilidad:** Creación de aplicaciones web usables y cómodas para mantener.
- Ofrece HTML y CSS como modelo de desarrollo del núcleo.
- Proporciona inicializadores de instancias.

- **Rutas:** Ofrece rutas que son las características principales que se utilizan para administrar la URL.
- **Depuración:** Proporciona la herramienta Ember Inspector para depurar aplicaciones Ember.
- **Plantillas:** Utiliza plantillas que ayudan a actualizar automáticamente el modelo si se cambia el contenido de las aplicaciones.

5.1.5. React

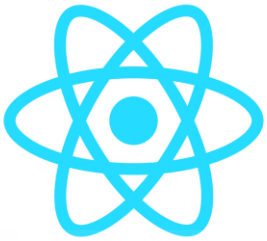

Figura 16. Logo React (Fuente: React)

React es un framework desarrollado por Facebook que simplifica el proceso de creación de interfaces de usuario interactivas. Es la base de React Native, un framework adyacente para crear aplicaciones móviles. Ambos tienen un flujo de datos unidireccional, que se considera más intuitivo que el enlace de datos bidireccional. La "recarga en caliente" (hot reload), es otra característica popular de los framework de React que permite a los desarrolladores ver inmediatamente los cambios a medida que se aplican. Se utiliza para desarrollar y operar interfaces de usuario dinámicas con alto tráfico entrante. Hace uso de un DOM virtual, por lo que la integración con cualquier aplicación es más sencilla. Algunas de sus principales características son:

- **Declarativo:** Crea una interfaz de usuario interactiva y dinámica para sitios webs y aplicaciones móviles. React se actualiza de forma eficiente y renderiza los componentes afectados cuando cambian los datos. Las vistas declarativas hacen que el código sea legible y fácil de depurar.
- **DOM virtual:** Para cada objeto DOM, hay un objeto DOM virtual correspondiente. Se crea una copia virtual del DOM original y es una representación de un objeto DOM.

- **Manejo de eventos:** React ha creado su propio sistema de eventos modelo de objetos W3C totalmente compatible. También proporciona una interfaz de navegador cruzado para un evento nativo, lo que significa que no hay que preocuparse por nombres y campos de eventos incompatibles. React reduce la capacidad de memoria a medida que el sistema de eventos se implementa a través de la delegación de eventos y tiene un grupo de objetos de eventos.
- **JSX:** JSX es una sintaxis de marcado que se parece mucho a HTML. JSX facilita la escritura de componentes de React al hacer que la sintaxis sea casi idéntica al HTML inyectado en la página web.
- **Rendimiento:** React utiliza un enlace de datos unidireccional con una arquitectura de aplicación llamada controles Flux. ReactJS ayuda a actualizar la vista para el usuario y Flux controla el flujo de trabajo de la aplicación. El DOM Virtual añade ventajas ya que compara los nuevos datos con el DOM original y actualiza la vista automáticamente.
- **React Native:** Es un renderizador personalizado para React; utiliza componentes nativos como bloques de construcción, en lugar de componentes web como hace React. También brinda acceso a las funciones de estas plataformas, además de transformar el código de React para que funcione en iOS y Android.
- **Basado en componentes:** En React, todos es un componente de la web dividido en pequeños componentes para crear una vista (o interfaz de usuario). Cada parte visual de la aplicación está envuelta dentro de un módulo autónomo conocido como componente. Los componentes de React se utilizan para definir los elementos visuales y las interacciones en las aplicaciones.

5.1.6. Conclusiones

Tras analizar algunos de los frameworks más utilizados de javascript, hemos decido que para la parte del servidor, back-end, de nuestra aplicación, el cual va a contener una api para gestionar todo el almacenamiento y consulta de datos de la base de datos, vamos a utilizar el framework de Node.js, ya que como hemos visto, es el mejor para realizar estas funciones y

ofrece mejores tiempos. Para el lado del cliente, front-end, la decisión ha sido complicada, ya que podríamos haber escogido cualquiera de los mencionados, pero al final nos hemos decantado por React, ya que nos ha parecido bastante cómodo gracias a su "hot reload", y también que sea un framework que se basa en componentes.

5.2. CREACIÓN SERVIDOR WEB Y API REST

En primer lugar, para el desarrollo de nuestra aplicación web, vamos a comenzar desarrollando la parte del servidor en el que se encontrará nuestra API.

Una API es un conjunto de definiciones y protocolos que se utilizan para desarrollar e integrar el software de las aplicaciones. Las siglas de API vienen de Interfaz de Programación de Aplicaciones (*Application Programming Interface*).

Nuestra API permitirá abstraer al cliente web de todas las operaciones CRUD (*Create, Read, Update, Delete*) que se realicen sobre la base de datos. De esta forma desde el cliente simplemente se realizará una llamada a un método de nuestra API, y nuestro servidor será el encargado de consultar, actualizar, almacenar o borrar los datos solicitados, mejorando el rendimiento de nuestra aplicación y haciendo que sea más segura.

5.2.1. JSON vs XML

A la hora de definir nuestra API, nos surge como duda que lenguaje es mejor utilizar para su definición si XML o JSON. Para ello, hemos realizado una tabla comparativa entre ambos lenguajes:

JSON	**XML**
El tipo de formato es intercambio de datos	El tipo de formato es lenguaje de marcado
OpenSource	OpenSource
Los objetos creados tienen tipo	Los objetos creados no tienen tipo

Los tipos de datos soportados son: string, number, boolean, null y array	Los datos están en formato string
No tiene capacidad de mostrar contenido	Al ser un lenguaje de marcado, tiene la capacidad de mostrar contenido
No tiene etiquetas	Los datos se representan con etiquetas
Es más rápido para leer y escribir	Cuesta más de leer, ya que la curva de aprendizaje es mayor
Se puede leer como una función de JavaScript	Se tiene que leer y transformar al lenguaje deseado

Tabla 3. Diferencias entre XML y JSON

Tras analizar algunas de las características más importantes de cada lenguaje, ha sido fácil llegar a la conclusión de que el mejor lenguaje para nuestro caso es JSON. Esto se debe a que se puede leer como una función JavaScript de forma sencilla, y permite que los datos enviados estén tipados. Por lo que vamos a desarrollar una API JSON.

5.2.2. Definición API REST JSON

Como hemos comentado previamente, vamos a crear una API REST que utiliza el lenguaje JSON. Los métodos que va a implementar nuestra API son todos los necesarios para la gestión de consultas CRUD con la base de datos.

5.2.2.1. getTemperature

Este método permite obtener las temperaturas que el dispositivo electrónico va registrando en la base de datos para una instalación concreta.

Petición GET	~/api/getTemperature

Cuerpo	Vacío

Respuesta	{
	"data": [
	{
	"Temperature": 27.75,
	"Date": "2021-04-25T12:14:10.000Z",
	"Sensor": 1
	},
	…
	]
	}

Tabla 4. Definición método GetTemperature

Este será el método que utilicemos para representar los datos en una gráfica en tiempo real, para ello lo llamaremos de forma recursiva después de un número determinado de segundos.

5.2.2.2. getConfiguration

Este método nos devuelve la configuración actual que tiene la instalación.

Petición GET	~/api/getConfiguration
Cuerpo	Vacío
Respuesta	"data": [{ "SamplingRate": 69, "ActivateRelay": 0, "StartDate": "2021-05-05T10:40:00.000Z", "EndDate": "2021-06-

	11T15:40:00.000Z" }]

Tabla 5. Definición método GetConfiguration

5.2.2.3. getActivateRelay

Este método nos sirve para consultar si el relé está activo o no.

Petición GET	~/api/getActivateRelay
Cuerpo	Vacío

Respuesta	{
	"data": [
	{
	"ActivateRelay": 0
	}
	]
	}

Tabla 6. Método GetActivateRelay

5.2.2.4. setConfiguration

Este método nos permite cambiar la configuración de la instalación y guardarla en bbdd.

Petición POST	~/api/setConfiguration
Cuerpo	{ "samplingRate": number, "startDate": string, "endDate": string

	}

Respuesta	{
	"data": {
	"result": "OK"
	}
	}

Respuesta NO OK	{
	"error": "Invalid data"
	}

Tabla 7. Método SetConfiguration

Los parámetros de fecha del cuerpo, deberán tener el siguiente formato “YYYY-MM-DD HH:mm”.

5.2.2.5. activateRelay

Permite activar o desactivar el relé.

Petición POST	~/api/activateRelay

Cuerpo	{
	"active": **boolean**
	}

Respuesta	Misma que en el punto anterior, tanto cuando da error como cuando funciona correctamente.

Tabla 8, Método ActivateRelay

5.2.2.6. changeInstallation

Permite cambiar entre las distintas instalaciones que definidas en el sistema.

Petición POST	~/api/changeInstallation

Cuerpo	{
	"installation": string
	}

Respuesta	Misma que en los puntos anteriores, tanto cuando da error como cuando funciona correctamente.

Tabla 9. Método ChangeInstallation

5.2.3. Implementación servidor

Para implementar el servidor, el primer paso que debemos hacer es descargar e instalar node, el cual incluye npm que es un sistema de gestión de paquetes, el cual nos servirá para instalar todos los paquetes que necesitemos para implementar nuestra aplicación.

El servidor será la única parte de la aplicación que se conecte a la base datos, por lo que tendremos que implementar la conexión a la misma. Como el sistema de gestión de base de datos que utilizamos es MySQL, necesitamos usar su paquete de npm, que es un driver de node.js para MySQL.

Para instalar cualquier paquete utilizando npm, lo único que tenemos que hacer es ejecutar en nuestra consola de comandos "*npm install nombre_paquete*". En la página de npm https://www.npmjs.com/ encontramos toda la documentación de todos los paquetes existentes y de cómo se utilizan.

Para establecer la conexión con el servidor de base de datos, hemos creado un fichero de configuración en el que podemos configurar la conexión al servidor. Y para establecer la conexión a partir de lo que se haya configurado hacemos lo siguiente:

```
/** Función que crea una conexión con la base de datos */
var con = mysql.createConnection({
    host: configData.dbconnection.host,
```

```
    user: configData.dbconnection.user,
     password: configData.dbconnection.password
});
```

Código 2. Código para establecer conexión con base de datos

Una vez establecida la conexión, procedemos a desarrollar los distintos métodos de la API. Para ello necesitamos instalar el paquete express de npm que incorpora muchas herramientas para desarrollar nuestra API REST.

```
/** Método para obtener las temperaturas registradas */
app.get('/api/getTemperature', (req, res) => {
   con.query('SELECT * FROM '+ installation +'.Temperature', function (err, result) {
      if (err) throw err;
      res.send({ data: result });
  });
});
```

Código 3. Función para obtener las temperaturas

Cuando se realice una petición HTTP **GET** para obtener las temperaturas, el servidor realizará una consulta de lectura para obtener todos los registros de temperatura que se hayan registrado, y los devolverá como un objeto JSON con la estructura mencionada en el punto 5.2.2.1.

Los métodos *getConfiguration* y *getActivateRelay*, se implementan de la misma forma que el anterior, realizando una consulta de lectura para obtener toda la configuración en el primer caso, o para obtener el valor de configuración de ActivateRelay, y devolver los resultados obtenidos como se indica en los puntos 5.2.2.2 y 5.2.2.3.

Por otro lado, tenemos las peticiones de tipo **POST**, que son aquellas en las que se envían datos para que se escriban en la base de datos.

```
/** Método que permite guardar la configuración */
app.post('/api/setConfiguration', (req, res) => {
   if(req.body && req.body.samplingRate && req.body.startDate && req.body.endDate){
     con.query('SELECT COUNT(*) as count FROM '+ installation +'.Configuration', function (err, result) {
        if (err) throw err;
        if(result[0] && result[0].count == 0){
          //Insertamos la fila con valores por defecto
          con.query('INSERT INTO '+ installation +'.configuration VALUES ('+ req.body.samplingRate +', 0, "' + req.body.startDate  +'", "'+ req.body.endDate + '")', function (err, result) {
              if (err) {
                 throw err;
                  res.send({ data: { result: “ERROR”} });
              }
              res.send({ data: { result: “OK” } });
           });
        }
        else{
          //Modificamos la fila con los valores
          con.query('UPDATE '+ installation +'.configuration SET SamplingRate = ' + req.body.samplingRate + ', StartDate = "'  + req.body.startDate + '", EndDate = "' + req.body.endDate + '"', function (err, result) {
              if (err) {
                 throw err;
                  res.send({ data: { result: “ERROR”} });
              }
              res.send({ data: { result: “OK” } });
           });
        }
```

```
    });
  }
else res.send({ error: "Invalid data"  });
});
```

Código 4. Código para guardar la configuración en bbdd

En este método lo que hacemos es comprobar si existe alguna configuración guardada para la instalación, si existe, entonces actualiza la que había para guardar la nueva. En caso de que no exista, lo que hace es crear un nuevo registro. Ya que en la tabla de la configuración debe de haber un único registro. Este método devuelve como resultado "OK", cuando se registró correctamente y "ERROR", en caso contrario. Cuando faltan datos en la consulta, lo que se hace es devolver "Invalid data".

Con el método *activateRelay* hacemos lo mismo que con el que acabamos de mencionar, pero modificando únicamente el valor que indica si el relé está activo o no.

Por último, tenemos el método *changeInstallation* el cual nos permite cambiar la instalación de la cual estamos obteniendo los datos. Las distintas instalaciones existentes en el sistema las tenemos definidas en el fichero de configuración.

5.3. CREACIÓN CLIENTE WEB

5.3.1. Diseño y mockups

El primer paso para implementar la parte del cliente de nuestra aplicación es realizar el diseño y mockups de cómo queremos que sea la aplicación. Para esto, hemos utilizado una herramienta llamada Sketch.

La idea es estructurar la aplicación de la siguiente manera. En la parte superior encontraremos el menú de la aplicación, desde el que podremos cambiar entre las diferentes vistas de la aplicación.

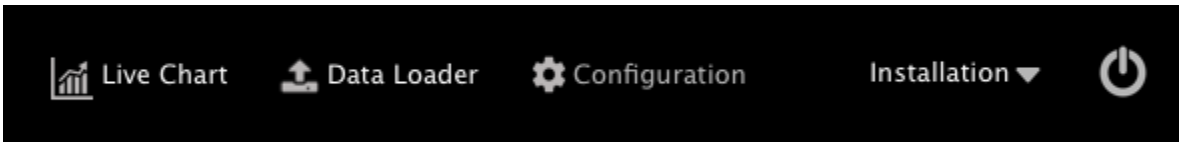

Figura 17. Mockup menú aplicación (Fuente: propia)

En la parte izquierda del menú, como se puede observar en la figura 17, encontraremos los nombres de las distintas vistas de las que se compone nuestra aplicación. En la parte derecha, encontramos los siguientes elementos, un botón para encender/apagar el relé y un desplegable para cambiar de instalación. El botón de encendido aparecerá coloreado de azul verdoso cuando la instalación esté activada, y rojo cuando se encuentre desactivada. En cuanto al desplegable, en lugar de "installation" aparecerá el nombre de la instalación, que coincide con el nombre de la base de datos, seleccionada en ese momento.

Además, cada vez que se seleccione una vista aparecerá resaltada de color azul verdoso en el menú para que al usuario le quede claro que vista se está visualizando.

Una de las vistas de la aplicación será la que muestre una gráfica actualizando los datos en tiempo real, a la que llamaremos "Live Chart". Desde esta página además podremos exportar los datos de la sesión activa a un fichero con formato CSV, por lo que tendremos que añadir un botón desde el que se podrá realizar dicha acción, como podemos observar en la imagen que se encuentra a continuación.

En la parte central de esta vista tendremos la gráfica temporal de los datos de la instalación en la que el tiempo estará representado en el eje de abscisas y la temperatura en el eje de ordenadas.

Figura 18. Mockup vista Live Chart (Fuente: propia)

En segundo lugar, tendremos otra página que también será una gráfica, pero en este caso no se mostrarán los datos en tiempo real, sino que en su lugar la gráfica estará vacía hasta que se carguen los datos de un fichero ".csv". Estos datos se cargarán desde el botón "Load files".

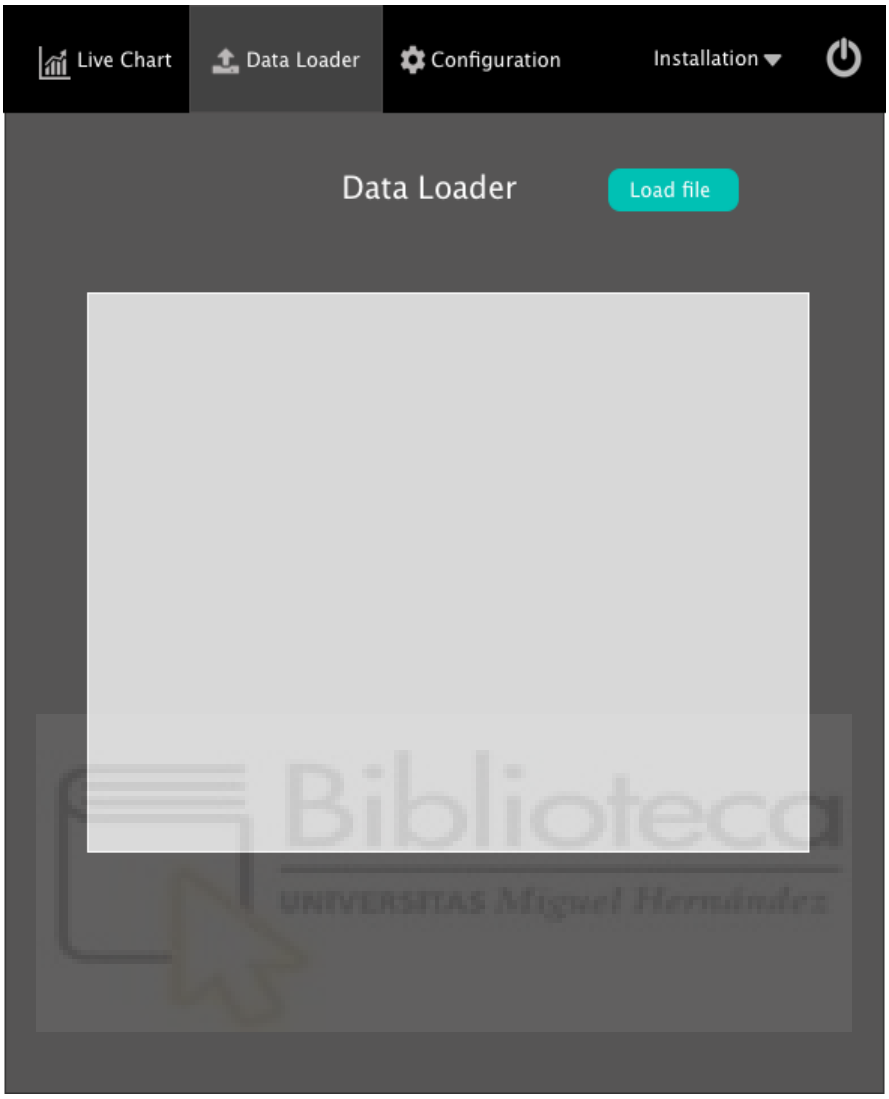

Figura 19. Mockup vistq Data Loader (Fuente: propia)

Por último, nos encontramos con la vista de configuración en la que tendremos un formulario con los valores que se pueden configurar y un botón para guardarlos. Estos serán la velocidad del muestreo de datos y la fecha y hora de activación/desactivación de la instalación a la que estamos apuntando.

Figura 20: Mockup vista Configuration (Fuente: propia)

5.3.2. Preparar entorno de desarrollo con React

Como ya comentamos en el punto 5.1, el framework que hemos escogido para implementar la parte front-end de nuestra aplicación es react. Para poder trabajar con este framework lo

primero que tenemos que hacer es abrir un nuevo terminal en el ide con el que estemos trabajando y ejecutar lo siguiente:

```
npx create-react-app my-app
cd my-app
npm start
```

Figura 21. Código para crear aplicación react (Fuente: React)

El primer comando lo podemos ejecutar ya que tenemos instalado node como comentamos en punto 5.2, y lo que hace es crearnos una nueva aplicación React con el nombre que la hayamos indicado, para ello tenemos que sustituir "my-app" por el nombre de nuestra app, en nuestro caso la hemos llamado "datalogger".

El segundo lo único que hace es situarnos en la carpeta que se ha creado con nuestra aplicación, y el último arrancarla. Toda la documentación la encontramos en la página de react:

https://es.reactjs.org/docs/create-a-new-react-app.html

Nosotros además vamos a utilizar Semantic UI que ofrece componentes para el estilo de la aplicación por lo que también lo tenemos que instalar. Una vez hecho esto ya podemos comenzar a implementar nuestra aplicación.

5.3.3. Implementación cliente web

En primer lugar, para tener organizado nuestro proyecto, hemos creado una carpeta llamada "client" en la que hemos metido las carpetas "public" y "src". En public guardaremos todos los recursos que necesitemos, ya sean hojas de estilo, imágenes, etc. Y en src, guardaremos todo el código de la aplicación como se puede ver en la imagen a continuación.

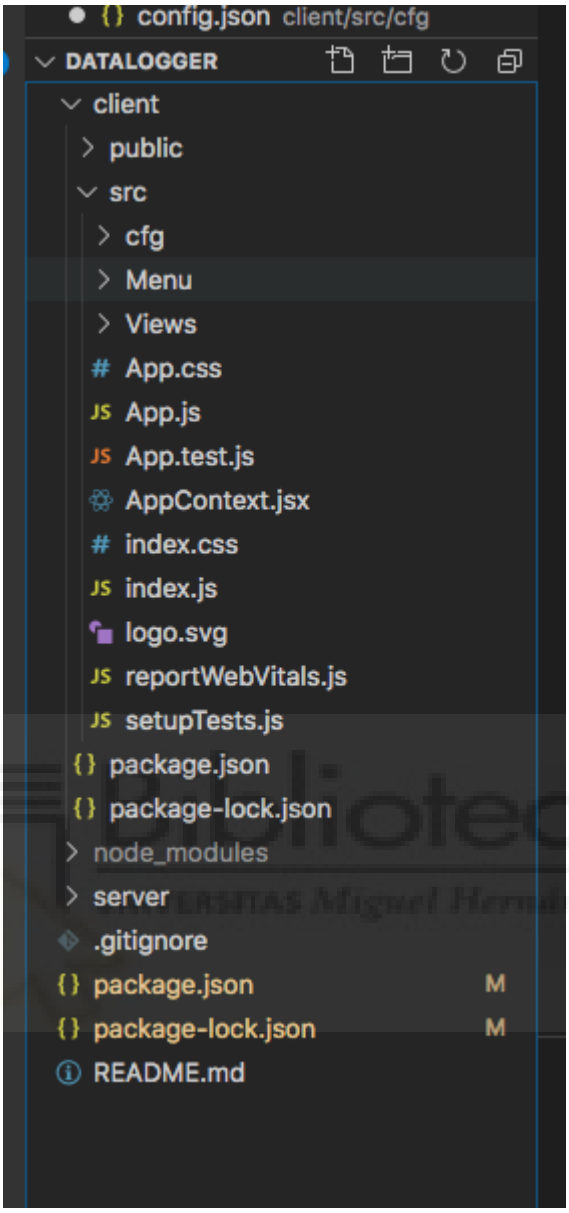

Figura 22. Estructura código aplicación web (Fuente: propia)

Hemos empezado creando el menú, para ello lo que hemos hecho ha sido crear un nuevo componente de react al que hemos llamado MenuBasic, que tiene dos propiedas una para saber el elemento que se está visualizando y otra para saber el estado del relé para saber si el botón de encendido debe de visualizarse rojo o azul verdoso. Hemo utilizado el componente Menu que ofrece Semantic UI y hemos añadido las distintas vistas con las que va a contar nuestra aplicación.

Además, hemos implementado varias funciones:

- getStatus() -> que realiza una petición HTTP GET, a nuestra API para saber si el relé está activo o no.
- changeStatus() -> realiza una petición HTTP POST a la API para cambiar el valor de configuración y activar o desactivar el relé.
- changeInstallation() -> realiza una petición HTTP POST a la API indicando al servidor que instalación ha sido seleccionada para que vuelva a enviar los datos al cliente de la nueva instalación y se muestren los datos correspondientes.

El código es el siguiente:

```
export default class MenuBasic extends Component {
  constructor(props) {
    super(props);
    this.state = {
      activeItem: '',
      power: false      };
  }
  static contextType = AppContext;
  handleItemClick = (e, { name }) => {this.setState({ activeItem: name });}
  setStatus = () => { this.setState({power: !this.state.power}); this.changeStatus(!this.state.power); }
  getStatus = async () => {
    const response = await fetch('http://localhost:5000/api/getActivateRelay');
    const data = await response.json();
    if (this.state.power != data.data[0].ActivateRelay) this.setState({power: data.data[0].ActivateRelay})
  }


  //Función que llama a la api para cambiar el vlor de la configuración y activar o desactivar el relé
```

```
  changeStatus = (status) => {
    const requestOptions = {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({ active: status })
    };
    fetch('http://localhost:5000/api/activateRelay', requestOptions)
      .then(response => response.json())
  }
  changeInstallation = (data) => {
    const requestOptions = {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify({ installation: data.value })
    };
    fetch('http://localhost:5000/api/changeInstallation', requestOptions)
      .then(response => response.json())
  }
  componentDidMount() {
    this.getStatus();
  }
  componentDidUpdate(){
    this.getStatus();
  }
  render() {
    const { activeItem, power } = this.state
    const databaseOptions = configData.databases.map(db => ({ key: db, text: db, value: db}));
    return (
```

```
<Menu inverted size="large">
  <Menu.Item
    className='menu-item'
  ></Menu.Item>
  <NavLink id='liveChart' to="/liveChart">
    <Menu.Item as='div'
      className='menu-item'
      name='liveChart'
      active={activeItem === 'liveChart'}
      onClick={this.handleItemClick}
    >
      <Icon name='chart line' size='large'/>
    Live Chart
    </Menu.Item>
  </NavLink>

  <NavLink id='dataLoader' to="/dataLoader">
    <Menu.Item as='div'
      className='menu-item'
      name='dataLoader'
      active={activeItem === 'dataLoader'}
      onClick={this.handleItemClick}
    >
      <Icon name='upload' size='large'/>
      Data loader
    </Menu.Item>
  </NavLink>
```

```
<NavLink id='configuration' to="/configuration">
  <Menu.Item as='div'
    className='menu-item'
    name='configuration'
    active={activeItem === 'configuration'}
    onClick={this.handleItemClick}
  >
    <Icon name='cog' size='large'/>
  Configuration
  </Menu.Item>
</NavLink>
<Menu.Item className='menu-item' id='installationDropdown'
  position='right'
>
  <Icon name='database' />
  <Dropdown
    inline
    header='Installations'
    options={databaseOptions}
    defaultValue={databaseOptions[0].value}
    onChange={(_e, data) => { this.changeInstallation(data);
this.context.setInstallation(data.value); }}
  />
</Menu.Item>
<Menu.Item className='menu-item'
  icon={power ? powerIcon : powerOffIcon }
  position='right'
  onClick={this.setStatus}
></Menu.Item>
```

```
        </Menu >
    )
  }
}
```

Código 5. Código componente React del menú de la aplicación

El desplegable que permite cambiar de instalación muestra las instalaciones que tengamos configuradas en nuestro fichero de configuración y al cambiar la instalación, llama al método changeInstallation() que hemos mencionado arriba y se refrescan los datos de la aplicación.

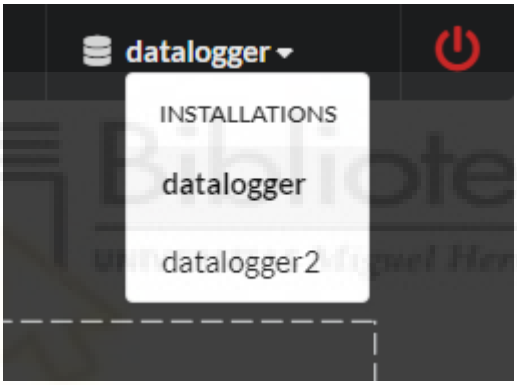

Figura 23. Selector para cambiar de instalación (Fuente: propia)

A continuación, debemos implementar las diferentes vistas de la aplicación. Vamos a comenzar por la vista “Live Chart”. Para esta vista, hemos creado un componente llamado chart, usando el paquete de npm llamado VictoryChart que sirve para representar datos en gráficas.

A este componente se le pasan como propiedad los datos que debe representar y dibuja una gráfica que tiene el tiempo en el eje de abscisas y la temperatura en el eje de ordenadas. Este componente lo podremos utilizar también para la vista “Load View” pasándole los datos que hayamos obtenido del fichero que se haya seleccionado.

Entonces en la vista de “Live Chart”, lo que hacemos es añadir nuestro componente que representa los datos en una gráfica, y para obtener los datos lo que hacemos es realizar una petición HTTP GET al método getTemperature de nuestra API cada 30s. Por lo que la información de la gráfica se va actualizando. El resultado lo podemos observar en la siguiente imagen.

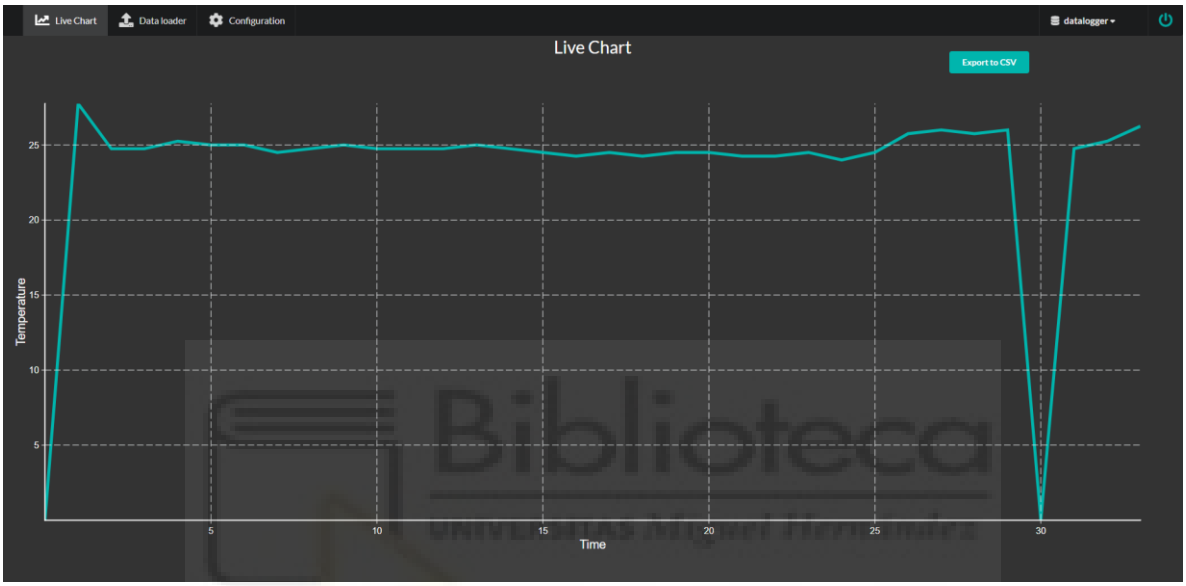

Figura 24. Vista gráfica Live Chart (Fuente: propia)

Para el botón de exportar a CSV, hemos creado el componente ExportButton, en el que generamos un CSV de dos columnas (tiempo y temperatura) con los datos obtenidos hasta el momento. El nombre del CSV es la fecha en la que se han importados los datos.

La vista de “Load View” es igual que la anterior, solo que por defecto no tiene datos que representar hasta que se carga un fichero desde el botón de “Load file”. Esta vista nos sirve para ver sesiones pasadas del datalogger.

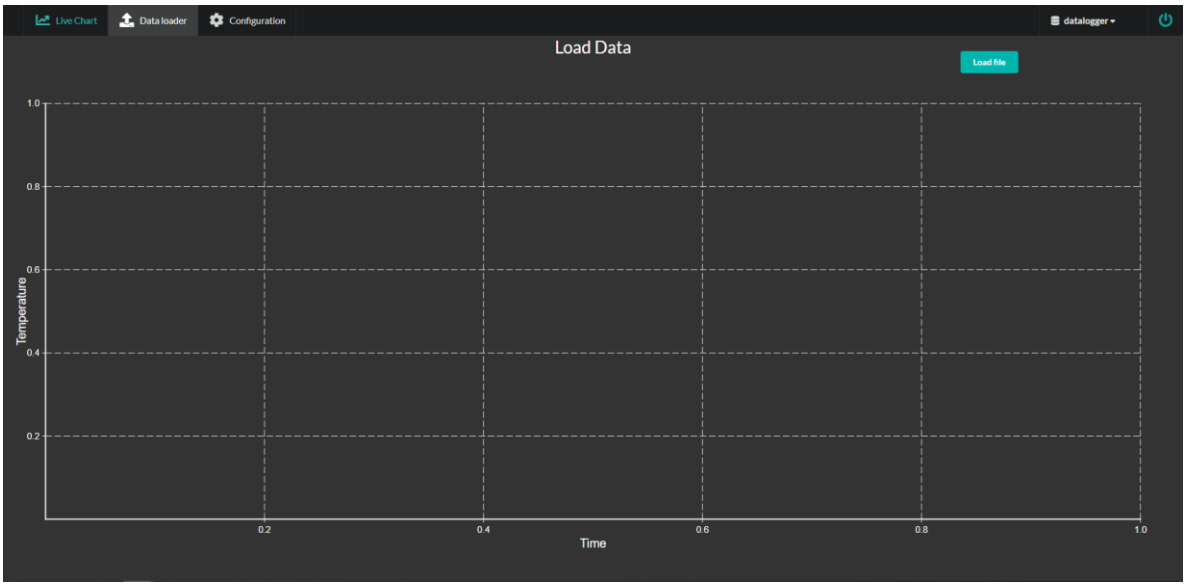

Figura 25. Vista Load Data (Fuente: propia)

Para este botón hemos generado un nuevo componente que abre un diálogo para poder seleccionar el archivo desde el que se desea que se carguen los datos para representarlos.

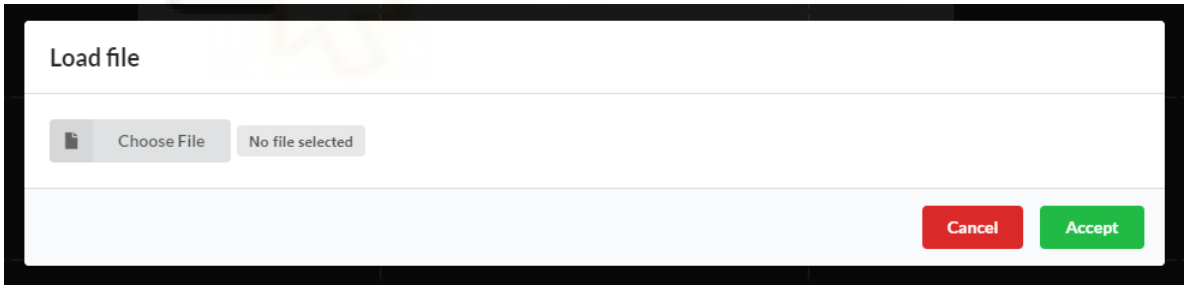

Figura 26. Modal para cargar archivos CSV (Fuente: propia)

El mensaje de “No file selected” se cambia por el nombre del archivo seleccionado, y “Choose file” abre el sistema de archivos para poder escoger el archivo seleccionado. En caso de seleccionar un archivo que no sea de tipo CSV, mostramos un mensaje como se muestra a continuación.

6. PRESUPUESTO Y COMPARATIVA CON DATALOGGERS COMERCIALES

6.1. DATALOGGERS COMERCIALES

Las características del modelo usado en el laboratorio de la universidad “34970A Data Acquisition / Datalogger Switch Unit son las siguientes:

- Precisión 0.003% DCV.
- 9 swictches, RF y control de módulos plugin, incluyendo 4 canales simultáneos.
- Digitalizador de muestras.
- Hasta 450 canales/s de frecuencia de muestreo.
- Hasta 120 canales por sistema.
- Hasta un millón de puntos almacenados en memoria.
- Mide y convierte hasta 14 señales de entrada diferentes: temperatura con termopares, RTDs y termistores, voltaje AC/DC, resistencias de 2 y 4 cables, frecuencia y periodo, corriente y capacitancia AC/DC, voltaje directo y en puente.
- Gran pantalla a color de 4,3” para facilidad de configuración y visionado de datos.
- Conectividad LAN y USB con PC.
- Memoria USB flash para copiar datos en una aplicación externa.

Figura 27. Datalogger comercial disponible en el laboratorio

El coste de este dispositivo parte de aproximadamente 1800 € la unidad. Se puede observar que estos dispositivos, a pesar de tener múltiples opciones y capacidades, son notablemente caros.

6.2. DATALOGGER INTERNET OF THINGS BAJO COSTE

Pasamos a detallar los costes de nuestro datalogger basado en internet of things de bajo coste:

Hardware	**Coste aproximado**
NodeMCU	8,5 €
WeMos D1 mini	5 €
Conversor MAX6675 y termopar	2 €
Placa de relés “2PH63083A”	2 €
Total	**12,5 € NodeMCU** **9 € WeMos D1 mini**

Tabla 10. Presupuesto

Se puede observar en la tabla que el coste de hardware de nuestro datalogger es inferior a 15 € utilizando cualquiera de las dos placas de desarrollo. Esto comparado con los dataloggers comerciales que cuestan cientos e incluso miles de euros dependiendo del modelo hacen de esta solución una buenísima opción para reducir el coste de equipos en laboratorios.

Hay que tener en cuenta que sería necesaria la contratación de un servidor MySQL que serviría de enlace entre nuestro microcontrolador y nuestra aplicación web y también para otorgarle persistencia a los datos de las instalaciones.

6.3. COMPARATIVA DE PRECIO

A la hora de evaluar qué opción es más interesante deberemos hacernos unas preguntas:

- ¿Dispongo de servidor de base de datos?
- ¿Cuántas unidades de datalogger necesito?
- ¿Necesito actuar remotamente sobre la instalación?

Estas preguntas son clave a la hora de decantarnos por una opción, pero si disponemos de un servidor de base de datos como puede tener una institución como la Universidad, y el número de unidades a comprar es alto parece evidente que la opción más económica pasa por el datalogger de bajo coste basado en internet of things. Además, nos permite programar la activación de la instalación y su operarla directamente.

7. CONCLUSIONES Y MEJORAS

7.1. DESARROLLOS FUTUROS

Un desarrollo interesante futuro sería la utilización del deep sleep del microcontrolador ESP8266 para ahorrar energía mientras la instalación está desconectada. Para la realización de esta tarea tendríamos que conectar la salida WAKE a la entrada RST y calcular el tiempo entre desconexión y conexión para generar un pulso cuando se deba despertar la tarjeta. La pega de este desarrollo es que la instalación solo funcionaria con la manera programada de activación. Si queremos que funcionen ambos modos de operación sobre la instalación habría que reiniciarla físicamente porque es la única manera de sacar el hardware del deep sleep.

También puede ser interesante soportar varios termopares en una instalación, para esto se ha incluido el ID del sensor cuando se registra la temperatura. Para esto lo único que habría que modificar es la gráfica de representación en la aplicación web para que acepte datos de diferentes identificadores de sensor y que los represente en diferentes colores. Otra modificación para registrar varios sensores a la vez sería representar cada sensor en diferentes canales, de manera que, tendríamos tantos canales como sensores tenga una instalación.

Además, se podría con relativamente pocos cambios en la aplicación web registrar cualquier tipo de magnitud física con respecto al tiempo. Esto ampliaría las posibilidades de la aplicación web para cualquier tipo de sensor. Para esta opción además de adecuar la

aplicación web, también habría que programar nuestro microcontrolador para adecuar la magnitud física medida.

Por último y a mi modo de ver, menos importante, también se podría usar la aplicación web para poner en hora nuestro microcontrolador y así obviar la parte del servidor NTP.

7.2. CONCLUSIONES

En mi opinión se ha conseguido un sistema robusto, flexible y barato de monitorización de instalaciones térmicas que era el principal objetivo. Además, la aplicación web, como hemos comentado, puede tener un amplio abanico de posibilidades y no ha de verse solamente como una representación de temperatura si no que puede también utilizarse para representar cualquier magnitud física con respecto al tiempo.

En cuanto a desarrollo personal, he conseguido aprender más sobre aplicaciones web que es algo muy útil hoy en día en caso de querer enfocar mi futuro profesional al sector de internet of things y, además, también he adquirido conocimientos sobre base de datos que pueden serme útiles en un futuro para campos como big data.

8. BIBLIOGRAFÍA

Arduino página oficial. https://www.arduino.cc/

Raspberry Pi página oficial. https://www.raspberrypi.org/

ESP8266. https://programarfacil.com/podcast/esp8266-wifi-coste-arduino/

WeMos y MAX6675. http://www.esp8266learning.com/wemos-max6675-example.php

Ntp. https://randomnerdtutorials.com/esp8266-nodemcu-date-time-ntp-client-server-arduino/

MySQL Arduino library. https://github.com/ChuckBell/MySQL_Connector_Arduino

Frameworks Javascript. https://trio.dev/blog/javascript-framework

Mejores frameworks Javascript. https://hackr.io/blog/best-javascript-frameworks

React. https://es.reactjs.org/

Node. https://nodejs.org/es/

SemanticUI. https://react.semantic-ui.com/

Express. https://expressjs.com/es/guide/database-integration.html