Proyecto Fin de Máster
Máster de Ingeniería Electrónica, Robótica y Automática

Red Inalámbrica de sensores para el control de climatización basado en datos

Autor: Richard Mark Haes Ellis

Tutor: Ignacio Alvarado

Dpto. Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2014

Departamento de Teoría de la Señal y Comunicaciones

Proyecto Fin de Máster
Máster de Ingeniería Electrónica, Robótica y Automática

Red Inalámbrica de sensores para el control de climatización basado en datos

Autor:
Richard Mark Haes Ellis

Tutor:
Ignacio Alvarado
Profesor Titular

Dpto. Sistemas y Automática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2014

Proyecto Fin de Máster: Red Inalámbrica de sensores para el control de climatización basado en datos

Autor: Richard Mark Haes Ellis
Tutor: Ignacio Alvarado

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Resumen

En esta tesis se presenta un proyecto en el que se despliega una red de sensores para monitorizar ciertas magnitudes de un sistema de climatización en el edificio de los laboratorios de la escuela técnica superior de ingeniería de la Universidad de Sevilla. Para ello se desarrolla unas placas electrónicas, unas conectadas a la corriente y otras que operan a batería. Se presenta entonces una solución para sensorizar un sistema que en un principio no tiene ningun sistema de monitorización. Estas medidas se usan para montar un modelo basado en datos recogidos de dicho sistema de sensorización en la que ya se puede aplicar una ley de control.

Abstract

This document presents a project in which a wireless sensor network is developed and used to monitor certain magnitudes of a HVAC system in the building of the laboratories of the engineering school in the University of Seville. In order to deploy such system a pair of electronics boards were developed; one that is continuously connected to the grid for power and another that operates with battery power. We therefore present a solution to gather data on a system that in principle does not have any type of feedbacl control because there are no measuremntes. With this data a model can be idetified based on data and with this model a control law can be designd to control the system.

Índice

1 Introducción

Durante la última decada, ha habido un auge tecnológico en el avance de las comunicaciones de redes de sensores inálambricas del tal modo que se ha inventado los WSN (del inglés Wireless Sensor Networks). Estos tipos de redes pueden proporcionar una diversidad de funcionalidades tales como la monitorización y tracking de procesos, la creacion de gemelos digitales para el mantenimeinto preventivo y control. Por ejemplo, los sensores pueden interconectarse para monitorizar y controlar condiciones medioambientales en los bosques u océanos. También se pueden ver aplicaciones de WSNs en el sistema de salud en hospitales o en maquinarias de procesos industriales.

Uno de los aspectos mas importantes de las WSNs es el consumo y operación a batería. Dependiendo del consumo de los nodos sensores, un WSNs, puede mantenerse operativo durante dias o incluso años, aunque considerando las dificultades de acceso para los nodos, el repuesto de baterías puede no ser práctico.

La seguridad juega un papel muy importante en las WSNs los datos intercambiados por los nodos pueden ser sensibles y el proceso de encriptación es una operación que consume bastantes recursos por lo que agotan prematuramente la bateria. En general, los algoritmos de encriptación y autetificación proporcionan servicios de confidencialidad y de integridad de datos. Este aspecto de la seguridad no se tratará ya que queda fuera del alcance del proyecto.

La recolección de datos tiene un interes particular en el modelado de sistemas inciertos. Con el auge del Machine learing, se ha llegado a poder modelar sistemas complejos basandose exclusivamente en datos. Esto es muy importante sobre todo cuando el modelado del sistema es muy complejo o cuando el sistema en si contiene muchos componentes que introducen incertidumbres.

1.1 Planteamiento del problema

En este proyecto se presenta un proyecto de redes inalámbricas de sensores para la monitorización del sistema de climatización de un edificio. En la figura 1.1 podemos ver un esquema del sistema de climatización del edificio de los laboratiorios de la escuela técnica superior de ingeniería.

Como se ve en la figura 1.1, la parte de interés del proyecto se ve remarcada en un recuadro discontinuo en rojo. Esta zona representa un despacho del edificio, esta formado por un fancoil secundario y otro primario. Un fancoil no es mas que un intercambiador de calor en el que circula agua caliente o fría por un radiador que contiene un ventilador para hacer pasar un flujo de aire. Cada despacho tiene su propio fancoil secundario en el que se absorbe aire de la misma sala y se vuelve a impulsar a dentro mas frío, o mas caliente. Luego hay otro fancoil primario que se reparte entre todos los despachos de la planta, el aire que entra en este fancoil primario viene del exterior y viene tratada. En esta sala nos interesa medir las siguientes magnitudes:

- Termperatura, presión y humedad en diferentes zonas de una sala, sobre todo en las salidas de los fancoils.
- Cantidad de personas en una sala.
- Apertura y cierre de puertas.
- Temperatura de agua de entrada y salida de los fancoils.
- Potencia consumida del fancoil secundario.
- Temperatura y humedad de la entrada y salida de los intercambiadores de calor.

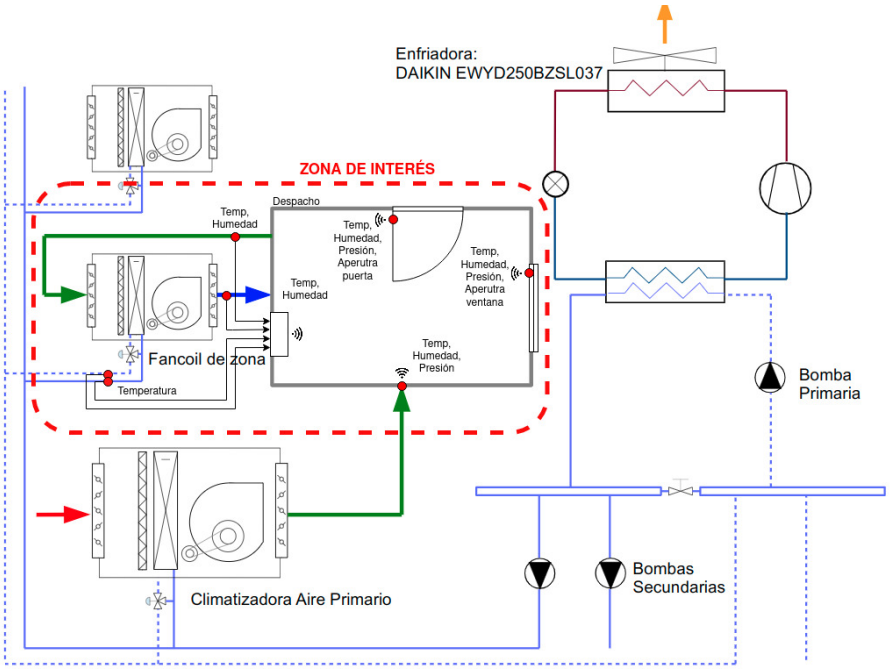

Figura 1.1 Esquema de climatización.

Otro requisito del proyecto es poder realizar las mediciones de forma no invasiva, ya que modificar la instalación requiere no solo trámites burocráticos interminables sino que también se tendría que parar la operación de la maquinária entera para su instalación.

Existen ciertas métricas que también pueden resultar difíciles de medir como la presencia y cantidad de personas que hay en una sala. Esta métrica la podemos medir indirectamente midiendo la cantidad de CO2 y partículas orgánicas (VOC), esto nos da una medida de la cual podemos estimar la ocupación de una sala.La apertura y cierre de puertas es importante ya que infuye sustancialemnte en la dinámica de climatización de la sala. Finalmente nos interesa en que proporción estan abiertos las puertas y ventanas, ya que una puerta inflye notablemente sobre la dinámica del climatización del edificio.

1.2 Soluciones comerciales

Existen diversas soluciones de monitorización para sistemas de cimatización en el mercado, sin embargo, muchas de ellas vienen restringidas de una forma u otra y lo hacen incompatible o inconveniente para este proyecto en particular. A continuación veremos algunas de estas soluciones:

Figura 1.2 Eupry.

En la figura 1.4 tenemos los sensores de Eupry. Estos permiten monitorizar temperatura, humedad, y CO2.

Los sensores se conectan mediante WiFi a una página web en la que se pueden visualizar los datos. Los aparatos funcionan bajo suscripción y el software es propietario. Los precios rondan los 400€ anuales.En este [https://eupry.com/] enlace se encuentra la página del fabricante.

Figura 1.3 Wizzard.

Wizzard es un WSN que permite desplegar cientos de nodos tanto cableados como inalámbricos y además permiten conexión mediante NODE-RED y MQTT. Es una solución mas flexible que la anterior ya que se puede interconectar con cualquier sistema que admita NODE-RED y MQTT. El inconveniente es que tienen un coste más elevado, eso si, con una calidad nivel industrial. En este [https://advantech-bb.com/product-technology/iot-and-network-edge-platforms/industrial-wireless-sensing-solutions/wzzard/] enlace se encuentra la página del fabricante.

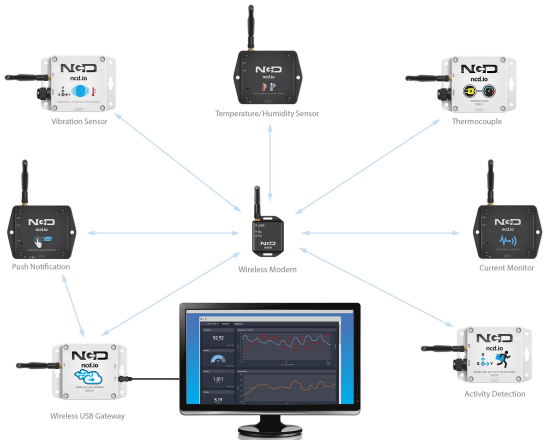

Figura 1.4 NCD.io .

Por último tenemos los sensores de NCD.io. Esta marca tiene una diversidad de sensores disponibles en su catalogo y permiten comunicarse por NODE-RED (Y por tanto MQTT y otros protocolos),se pueden conectar con dispositivos ESP32 y raspberry. Es una solución más atractiva ya que permite conectar otros micros custom, pero tiene el inconveniente de que los sensores vienen a un coste elevado. (Ronda los 250 euros por sensor). En este[https://store.ncd.io/] enlace se encuentra la página del fabricante.

Como podemos ver, existen soluciones en el mercado que puedan en un principio cubrir las necesidades de este proyecto pero vienen con un alto coste. Cuanto más flexible es el sistema, más caro es. Además, teniendo en cuenta que hay alrededor de 40 salas y en cada una hay unas 10 medidas a tomar, el precio se dispara rápidamente. Por esto se ha optado por diseñar y montar nuestro propio sistema IoT basado en ESP32, ya que contamos con una amplia comunidad de soporte, tenemos acceso a un sin fin de sensores de todo tipo a coste sustancialmente menor y podemos decidir que tipo de interfaz de comunicación queremos para el sistema completo.

1.3 Planteamiento de la solución

Tomando como referencia la figura 1.1, tenemos una serie de medidas que tomar cerca del fancoil secundario de la sala junto con medidas ambientales de temperatura humedad y presión en otros lados lejos del fancoil, por tanto, se ha decidido realizar la red de sensores en dos partes.

La primera esta compuesta por una centralita de sensores que se encargará de medir la temperatura de entrada y salida del agua del fancoil, la temperatura y humedad ambiental a la entrada y salida del fancoil, el nivel de CO2 y particulas orgánicas en el airle dentro de la sala y por último la corriente consumida por el fancoil. Además se encargara de registrar la fecha y hora de las medidas mediante un RTC. Estas medidas junto con su *Timestamp* se almacenarán en una tarjeta microSD. Por último, servirá como router en el que se conectarán nodos con diferentes capacidades de sensorización.

La segunda parte compuesta por nodos se encargara de medir la temperatura, humedad y presión en las diferentes partes de la sala, y además podrá medir el porcentaje de apertura de una puerta o ventana. Esto lo realiza mediante un encoder de cuadratura y un mecanismo que se detallará mas adelante. Un esquema del planteamiento se puede visualizar en la figura 1.5.

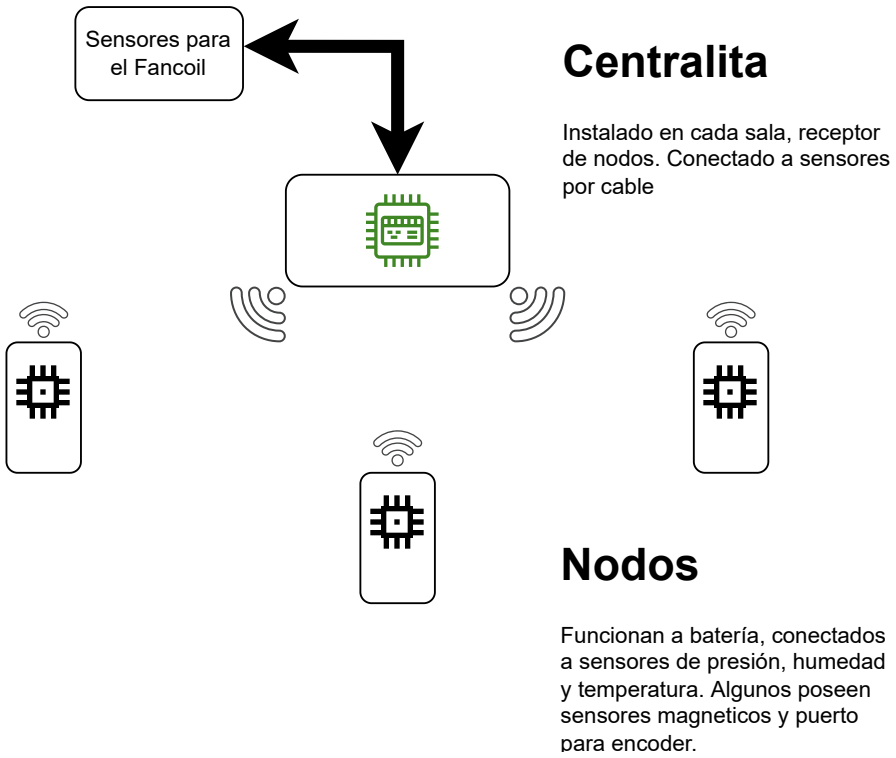

Figura 1.5 Esquema del planteamiento.

2 Hardware del sistema

En este capítulo se hablará sobre el harware del sistema de sensorización. Tal y como se ha visto en el capítulo anterior, el sistema se compone de dos partes, la centralita de datos y los nodos. La centralita de datos estará conectada continuamente a la red de alimentación mientras que los nodos estarán conectados a baterías. Esto permite usar en la centralita los sensores de mayor consumo y que poseen un precalentamiento (Como el sensor de CO2) y en los nodos aquellos que no requieren ningún precalentamiento y que tienen modos de bajo consumo.

2.1 Microcontrolador

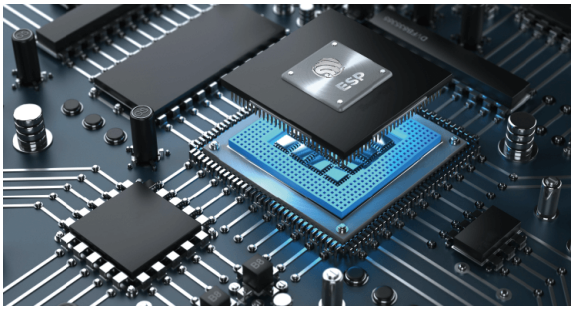

Figura 2.1 Plataforma ESP.

Para el microcontrolador se ha estudiado la posibilidad de usar un Arduino por su gran soporte comunitario y por su bajo coste. Pero no se ha optado por esta soución ya que al optar por una placa con cualquier tipo de comunicación inalámbrica se disparaba en precio y además no aportaban suficientes entradas y salidas para los sensores elegidos.

Se ha optado entonces por usar microcontroladores de la marca Espressif ya que aportan varias ventajas frente a los Arduinos, entre ellas, las diferencias mas importantes son:

- Menor coste
- Mayor capacidad computacional (40Mhz mínimo, hasta 240Mhz!!)
- Compatible con librerías Arduino
- Modos de bajo consumo.
- WIFI/Bluetooth integrado.
- Doble núcleo.
- Mayor memoria flash.
- Soporte RTOS.

2.1.1 Centralita general

Para la centralita general de datos se ha optado por usar una placa de desarrollo Sparkfun ESP-32 thing. En este caso no se necesita ninguna funcionalidad de bajo consumo ya que en este caso la placa irá conectado a la red de alimentación y estaŕa continuamente encendido.

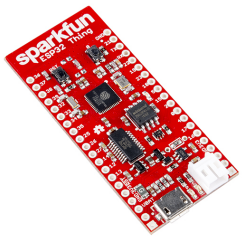

Figura 2.2 Placa de desarrollo Sparkfun ESP32 Thing.

Las características más importantes de esta placa son las siguentes:

- Procesador doble nucleo de Tensilica LX6
- Velocidad CPU hasta 240MHz
- 520kB SRAM
- Wifi integrado 802.11 BGN
- 28 GPIO
- 4MB Memoria flash

Esta placa irá conectada a una PCB diseñada para albergar los diferentes conectores y componentes tales como el RTC, tarjeta microSD, sensor CO2 y puertos de conexion.

2.1.2 Nodos

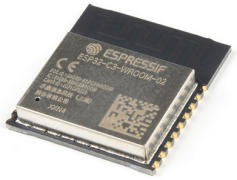

Figura 2.3 Modulo ESP32-C3-Wroom-N2.

Para los nodos se ha optado por usar la serie ESP32-C3 de Espressif, ya que cuenta con similares características que el micro anterior pero dispone de variantes en un formato compacto con pines suficientes para nuestra aplicación. Además veremos unos modulos en formato SMD (Surface Mounted Device) que incluyen ya la circuitería de la antenna de radio bien dimensionado junto con una pantalla RF que aisla la circuitería antes interferencias externas.

En nuestra aplicación usaremos el ESP32-C3-WROMM-N2-H4 el cual se puede ver en la figura 2.3. El ESP32-C3-WROOM-N2-H4 es un SoC (System on Chip) de un único nucleo con WiFi y Bluetooth 5 (LE). Está basado en la architectura open-source de RISC-V.

En la siguiente lista se muestra las principales características del SoC.

- CPU 32 bits de un núcleo hasta 160Mhz.
- Protocolos IEEE 802.11b/g/n
- Bajo consumo (5uA hasta 130uA)
- 4MB memoria flash SPI.
- 15 GPIO

Este SoC se usara conjuntamente en una placa PCB con circuitería adicional para la alimentación a batería, sensor de temperatura, presión y humedad, y los canales de entrada para los sensores de puertas y ventanas.

2.2 Sensores

2.2.1 Temperatura y humedad: DHT-22

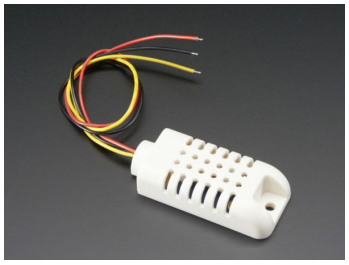

Figura 2.4 Sensor DHT22.

Este sensor dispone de un procesador interno que realiza el proceso de medición, la medida se puede leer mediante una señal digital, por lo que resulta muy sencillo obtener la información desde un microcontrolador. El protocolo que se utiliza para leer dicho sensor se llama *1-Wire*.

2.2.2 Temperatura: DS18B20

Figura 2.5 Sensor DS18B20.

El DS18B20 es un termómetro con una resolución de 9 a 12 bits que incluye una función de alarma de temperatura inferior y superior programable mediante su memoria no volátil. El DS18B20 se comunica mediante el bus 1-wire (1 cable) que por definición solo requiere de 1 cable y tierra para funcionar. El D18B20 puede derivar su alimentación mediante el cable de datos (Alimentación parasítica) eliminando la necesidad de un cable adicional de alimentación. El D18B20 dispone de un código de 64 bits único para cada sensor que permite conectar múltiples sensores al mismo bus *1-Wire*. Las aplicaciónes que pueden benifciarse de estas funcionalidades son principlalemnte los sistemas HVAC para el control de climatización.

2.2.3 CO2 y Particulas Orgánicas(VOC): CCS811

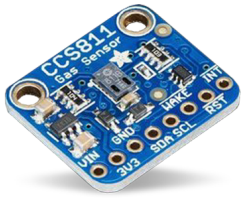

Figura 2.6 Sensor CCS811.

El CCS811 es un sensor de gas de bajo consumo que integra un sensor de óxido de metal (MOX) para detectar una variedad de partículas orgánicas (VOCs). Esto es útil para la monitorización de la claidad del aire en interiores.

El sensor esta basado en la tecnologia de AMS que habilita alta fiabilidad para sensores de gas, altos ciclos de vida y reduccion de consumo de energía.

El CCS811 lleva integrado un microcontrolador que maneja todas las funcionalidades del sensor y procesa las medidas. Despues de procesar estas medidas, las proporciona a través del puerto I2C para su lectura. Este método simplifica la integración de dicho sensor en las aplicaciones y disminiye el tiempo que se tarda para sacar el producto a mercado.

El sensor soporta diversos algoritmos que procesan las medidas raw para dar como resultados las medidas de TVOC, esta medida es un equivalente de CO2 (eCO2), y los humanos son la principal cuasa de los TVOSs (del inglés, Total Volatile Organic Compounds) lo cual nos da una medida de la ocupación de las salas.

2.2.4 Presión, Temperatura y Humedad: BME280

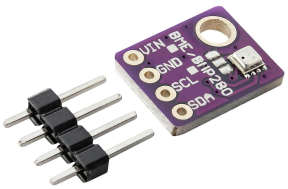

Figura 2.7 Sensor BME280.

El BME280 es un sensor combinado que incluye humedad, presión y temperatura. Esta basado en principios de medida probados en la industria luego proporciona una alta fiabilidad y alto ciclo de vida. El sensor esta encapsulado en un paquete LGA el cual es muy compacto y se puede integrar facilmente en cualquier sitio. Su pequeño formato permite un consumo muy pequeño. Esto lo hace adecuado para aplicaciones que funcionan bajo batería.

El sensor de humedad proporciona una respuesta rápida ante cambios externos y posee gran exactitud sobre un rango amplio de temperaturas.

El sensor de presión mide la presión barometrica absoluta con una alta exactitud y resolución con muy poca varianza de ruido.

El sensor de temperatura ha sido optimizado para minimizar el efecto de ruido con una resulución alta. Dicha medida se usa internamente para compensar el sesgo en los dos sensores anteriores y además puede usarse como estimación de la temperatura ambiental.

El dispositivo propociona dichas medidas mendiante las interfaces SPI e I2C. Se puede alimentar desde 1.71 a 3.6V. Cuando no se esta midiendo el sensor consume alrededor de 0.1uA lo cual es ideal para aplicaciones a batería.

2.2.5 Interruptor magnético

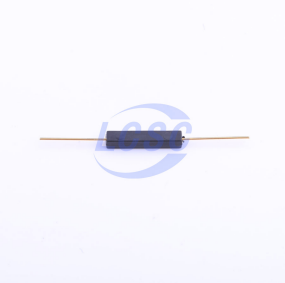

Figura 2.8 Interruptor magnético.

El interruptor magnetico actua como un interruptor normal solo que se acciona mediante la presencia de un imán. Esto lo usaremos en los nodos operados a bateria para detectar la apertura y cierre de las puertas.

2.2.6 Sensor de corriente

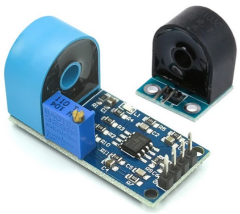

Figura 2.9 Sensor de corriente.

El sensor de corriente es el ZMCT103 y permite medir corrientes alternas hasta 5 amperios. La salidna nos la da en tension y es proporcional a la corriente detectada en el toroide. Con el potenciometro podemos ajustar la sensibiidad del sensor. Este sensor se usara para medir la potencia consumida por el fancoil y por tanto habra que hacer un procesamiento para calcular dicha potencia a partir de la señal sinusolidal.

2.2.7 RTC: Reloj tiempo real

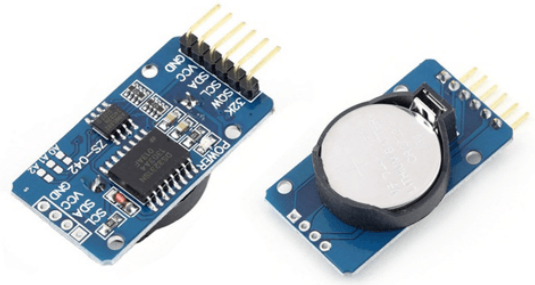

Figura 2.10 RTC.

Un RTC es un dispositivo electrónico que permite obtener medidas del tiempo. Estan formados por un cristal resonador integrado en la electrónca que contabiliza de forma correcta el paso del tiempo. Además los

RTC tienen en cuenta las peculiaridades del sistema de tiempo que tenemos, como es el sistema sexsagecimal, los meses, años bisiestos, etc..

Nosotros usaremos el integrado DS3231 que incorpora medición y compensación de la temperatura. Esto nos garantiza una precision de 2ppm lo que equivale a un desfase de 172ms/día.

Este integrado permite además la incorporación de una batería para mantener la contbilización del tiempo incluso durante un apagón. Las medidas se proporcionan a través del puerto I2C. Cabe destacar que no usaremos un módulo hecho sino implementaremos la circuiteria en la placa PCB directamente para ahorrar espacio en la placa.

Nota: Es importante destacar que el microcontrolador ya posee un reloj RTC que tiene alrededor de la misma precisión que el DS3231, pero con la diferencia de que no permite conectar una batería externa por si se corta la corriente. Esto es algo que ocurre amenudo en la zona donde se instalará el sistema por lo que a primera vista resulta imprescindible usar un RTC externo.

2.3 Memoria

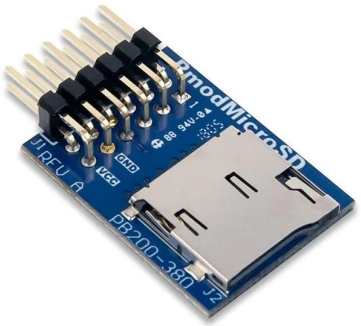

Figura 2.11 Modulo tarjeta microSD.

Para poder almacenar los datos de los sensores se ha optado por usar una tarjeta microSD en la centralita de datos. La centralita recogerá información de los nodos e irá leyendo los sensores en los puertos de conexion y registrando los datos en un fichero de texto. En la figura se muestra un módulo de tarjeta microSD pero en nuestro caso implementaremos directaemtne en la placa PCB la circuiteria para aceptar la tarjeta.

3 Diseño de PCB

Figura 3.1 Editor PCB.

Para incorporar todos los elementos electrónicos diseñaremos dos placas PCB. Una de ellas será para la centralita de datos y otra para los nodos sensores. Para el diseño de estas placas se ha hecho uso del programa EasyEDA. Este programa nos permite diseñar un esquemático e incorporar los componentes seleccionados desde la tienda LCSC Electronics. La ventaja de esta herramientas es que estan ligadas los símbolos, footprints y hoja de datos a cada elemento del esquemático, facilitando el diseño.

3.1 Diseño Centralita

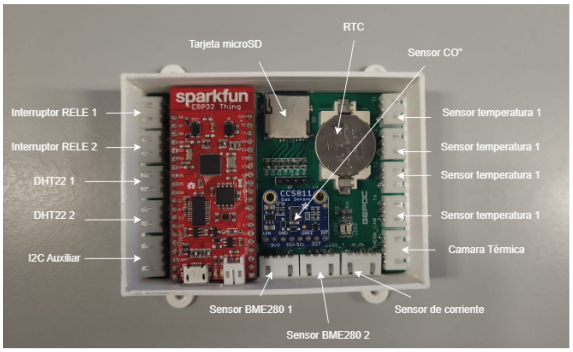

Figura 3.2 Esquema de componentes de la PCB de la centralita.

El diseño de la PCB se ve en la figura 3.2 y muestra los diferentes puertos de conexión que posee la placa junto los los componentes internos. Como se puede ver, se ha introducido algunos puertos adicionales como

el de I2C auxiliar, conector para una cámara térmica y puertos para sensores BME280, estas conexiones son para futuras ampliaciones.

- Placa de desarrollo ESP32-Thing de Sparkfun
- Sensor CCS811 de Sparkfun
- Reloj timepo real (RTC)
- 2 Leds indicadores
- Ranura para tarjeta microSD
- Resitencias pullup para I2C y One-Wire
- Conectores de 3 y 4 pines para los sensores.

El esquematico de la placa centralita se ve en la figura 3.3.

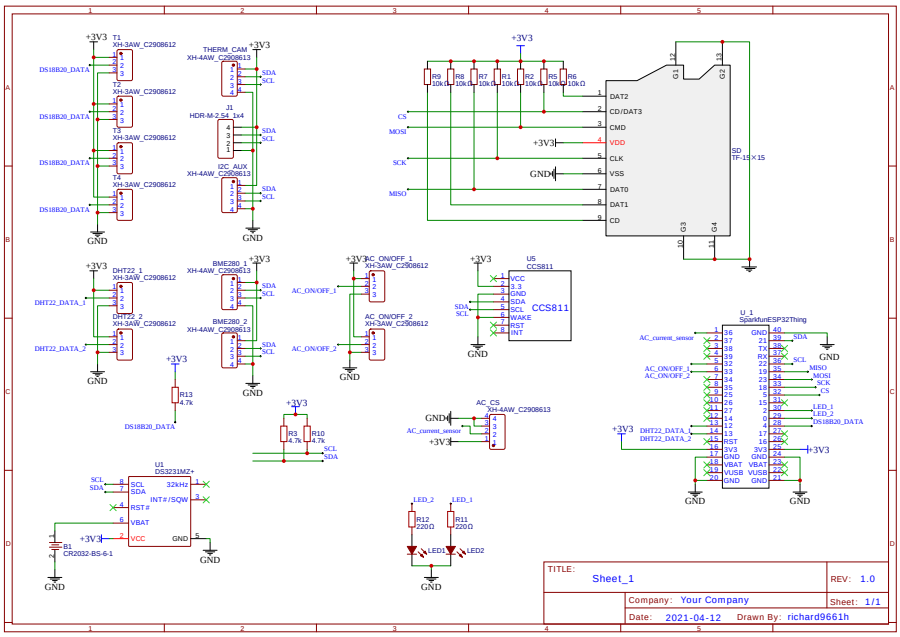

Figura 3.3 Esquematico centralita.

En la figura 3.4 podemos ver el conexionado del sensor de CO2. En este caso el sensor se comunica con el micro ESP32-Thing mediante I2C. Este bus I2C requiere de dos resitencias pullup ya que el módulo no las incluye. Lo mismo ocurre con el integrado del reloj a timepo real. El conexionado del RTC lo podemos ver detallado en la figura 3.5.

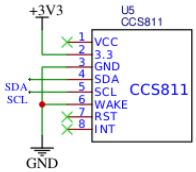

Figura 3.4 Esquematico sensor CO2.

Para medir el tiempo en el que se realizan se usar un DS3231, el circuito para este integrado se muestra en la figura 3.5.

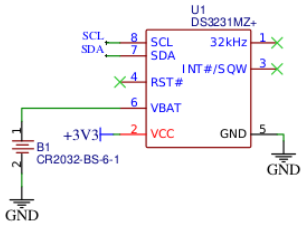

Figura 3.5 Esquematico RTC.

Como se comentó anteriormente, se ha usado una tarjeta microSD para almacenar los datos recogidos de los sensores. Las tarjetas microSD soportan dos protocolos de comuinicación: SD y modo SPI bus. El host, en este caso el ESP32 puede elegir cualesqueira de esto dos modos. En nuestro caso se ha optado por usar SPI ya que el micro ESP32 dispone de un puerto SPI nativo. El esquema de conexionado se ve detallado en la figura 3.6. Una conexión típica requiere de resistencias pullup en las líneas de transmisión y de seleción (Chip select CS). Cabe notar que no estamos haciendo uso del pin de detección de tarjeta *CD* (Card Detect).

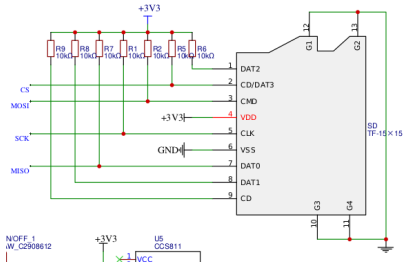

Figura 3.6 Esquematico tarjeta microSD.

Por último queda conectar el resto de los componentes al micro, en la siguiente figura se ve un detalle de dichas conexiones.

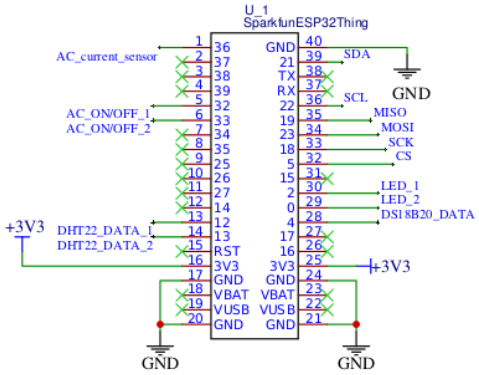

Figura 3.7 Esquematico ESP32-Thing.

3.2 Diseño Nodos

En la siguente figura podemos ver un esquema de las diferentes partes que tiene un sensor nodo.

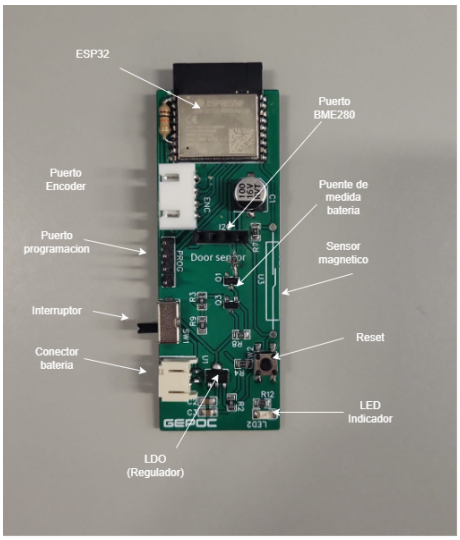

Figura 3.8 Esquema de componentes PCB de los nodos sensores.

En el esquematico de la figura 3.9 tenemos el conexionado para la placa de los nodos. En ella se incluyen los siguentes componenetes:

- Módulo ESP32-C3-Wroom-N2
- Conector de bateria 2P
- Conector para sensor BME280
- Interruptor ON/OFF
- Regulador LDO de bajo consumo 3.3V
- Circuito de medición de tensión de bateria
- Interruptor magnético
- Puerto de programación
- Boton de reset
- LED indicador

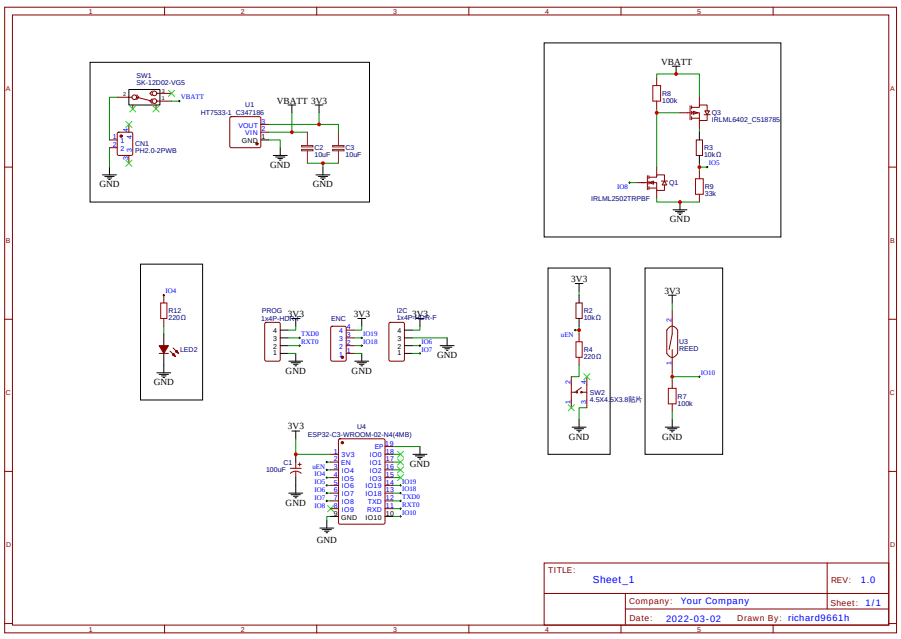

Figura 3.9 Esquematico nodos.

En los nodos remotos operados a batería puede ser interesante medir la tensión la batería para estimar la vida útil que pueda tener el nodo y tomar desiciones de mantenimiento preventivo. Para ello se ha diseñado el siguente circuito que mide la tension de la batería. Esta compuesto por un divisor de tensión activo por mosfet. Esto nos permite ahorrar batería cuando no se esté midiendo. Se ha tenido en cuenta el uso de componentes con baja corriente inactiva (Quiescent), por ello se ha hecho uso de mosfets.

El mosfet Q3 de canal P (Activo a nivel bajo) corta la circuiteria encima del divisor de tensión (High-side switch), esto es asi, ya que si desconectamos abajo en la tierra, la tensión de la bateria viajará al pin del procesador, lo cual podría dañarlo. Al ser un mosfet High-Side tenemos, para desactivarlo tenemos que usar una tension equivalente a la tension del colector del mosfet. Luego colocamos otro mosfet Q1 de canal N (Activo a nivel alto) que, al activarse, conecta la puerta del mosfet Q3 a tierra, en caso de que este apagado Q1, en la puerta del mosfet Q3 aparecerá la tensión de la batería, desconectando el circuito del divisor de tensión.

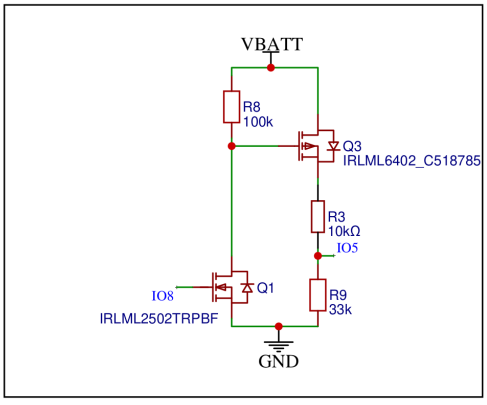

Figura 3.10 Esquematico medida de nivel de batería.

Para detectar la apertura y cierre de las puertas y ventanas se usa un interruptor magnético. Este interruptor se conecta con un pullup tal y como se muestra en la figura 3.11. Nota: El pin 10 del micro no tiene funcionalidad de interrupción cuando el microcontrolador esta en modo bajo consumo. Para que funcione desde bajo

consumo tiene que conectarse a algun pin habilitado con RTC. [https://docs.espressif.com/projects/esp-idf/en/latest/esp32c3/api-reference/peripherals/gpio.html]

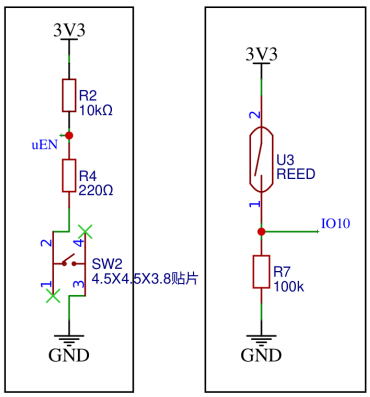

Figura 3.11 Sensor interruptor magnético.

Para poner el micro del nodo en modo (Modo programación) y modo normal para que arranque el programa tenemos que conectarle una serie de componentes pasivos. Hay que destacar que por errores de diseño no se han incluido ciertos componentes necesarios en el diseño original y por tanto el esquematico actualizado no corresponde completamente a la foto de la figura 3.8. Las diferencias entre el ciruito antiguo y el nuevo se ven la la figura siguente:

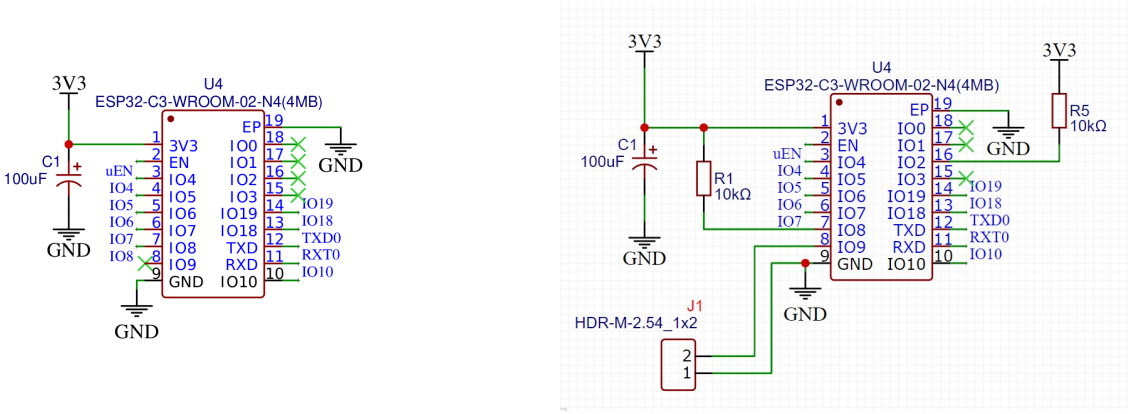

Figura 3.13 Esquematico antiguo (izquierda) vs nuevo (derecha).

En este caso se ha añadido un puente para meter el micro en modo bootloader para cargar código y luego se le ha añadido dos resistencias que hacen faltan para que arranque el programa desde flash.

4 Programación

4.1 Toolchain: Vscode con ESP-IDF

4.1.1 VSCODE

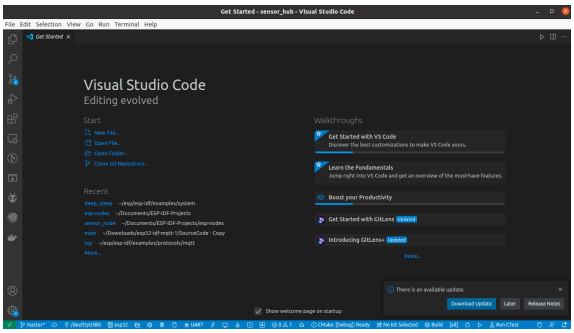

Figura 4.1 Visual Studio Code.

Visual Studio Code, tambien conocido como VS Code, es un editor de código general hecho por Microsoft. Las principales características de este editor es que permiten depurar, permite detectar errores de sintaxis mientras se escribe código, tiene funcionalidades auto-completado de código, macros o snippets de código, refactorizado de código, y control de versiones entre muchas otras cosas. Es un editor muy personalizable y además permite instalar extensiones para ampliar la funcionalidad del editor.

Puede usarse con una multitud de lenguajes de programación, incluyendo Java, JavaScript, Go, Node.js, Python, C++, C, Rust y Fortran. En nuestro caso nos interesa C/C++.

Una de las funcionalidades más importantes para abarcar un proyecto de esta magnitud es el control de versiones con Git. Esta funcionalidad viene por defecto en el editor con el añadido de que se tiene una interfaz más intuitiva para manejar tu código. Con Git podemos tener una versión del código funcional y desarrollar encima de esta versión sin el miedo de perder dicha versión estable.

4.1.2 ESP-IDF

Como hemos comentado anteriormente, uno de los puntos fuertes del VS Code es la posibilidad de instalar extensiones para ampliar la funcionalidad del editor. ESP-IDF es un framework para desarrollar código en los dispositivos de Espressiff, en ese sentido es similar a lo que te proporciona el IDE de Arduino, por lo que instalaremos la extension de ESP-IDF en vscode. Esta extensión incluye todo lo que necesitamos para poder compilar, depurar y ejecutar código. Además incluye todas las librerías enstandar junto con ejemplos para cada periférico del micro lo cual lo hace muy útil para el desarrollo de cualquier proyecto.

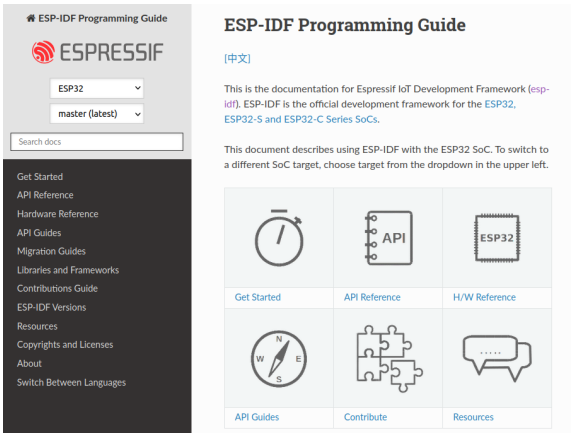

Figura 4.2 Plataforma de desarrollo ESP-IDF para micros de Espressiff.

Para instalarlo solo hay que seguir las instrucciones de la pagina de ESP-IDF para el editor VS Code.

4.1.3 ESP-IDF-LIB

ESP-IDF-LIB es un repositorio de librerías para los sensores mas comunes del mercado, esta activamente en desarrollo por lo cual podemos encontrar ejemplos para incorporar a nuestro proyecto.

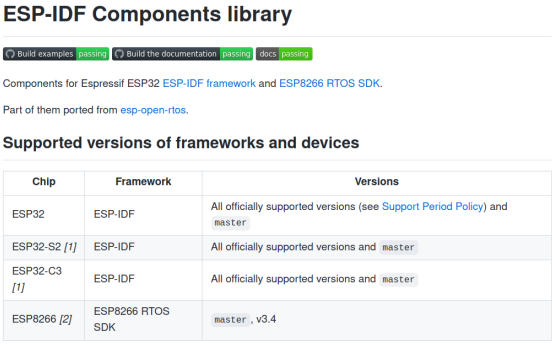

Figura 4.3 Librería de sensores para ESP-IDF.

4.2 FreeRTOS

Figura 4.4 FreeRTOS.

FreeRTOS es un sistema operativo de tiempo real para microcontroladores y pequeños procesadores. Esta distribuido gratuitamente bajo la licencia de código abierto del MIT. FreeRTOS incluye un kernel y una creciente lista de librerias para IoT adecuadas para el uso comercial. Esta construido con emfasis en robustez y facilidad de uso.

Haremos uso de freeRTOS ya que el microcontrolador de Espressif lleva por defecto código de FreeRTOS que maneja los radios de WiFi y Bluetooth. En algunos modelos de doble nucleo incluso se puede configurar hilos para que se ejecuten en la cpu secundaria. También cabe destacar que los ejemplos que vienen en las librerías de Espressiff vienen escritas en hilos, lo cual lo hace muy portable y fácil de implementar en cualquier proyecto.

4.3 ESP-NOW

Figura 4.5 Esquema de comunicación ESP32-NOW.

ESP-NOW es un protocolo desarrollador por Espressif, que habilita la comunicacion entre multples dispositivos sin la necesisdad de usar Wi-Fi. El protocolo es similar a la comunicación inhalambrica de bajo consumo a 2.4GHz que se suele desplegar en ratones para ordenador. El emparejamiento es necesario antes de iniciar una comunicaciones entre dispositivos. Despues del emparejamiento, la conexion es segura y peer-to-peer, sin necesidad de un mensaje ACK (Acknoledge, reconocimiento) o handshake. Esta es la cominicación que usaremos entre los nodos y la centralita de datos, tendremos varios nodos que inicializaran una cominciación con la centralita y se registran, a partir de entonces los nodos mandan información cada cierto tiempo sin necesidad de volver a establecer la conexión.

4.4 Esquema de software

En este apartado se trata de explicar el esquema del firmware del microcontrolador principal y del nodo. En la figura 4.6 podemos ver dicho esquema. Como se puede ver, el firmware esta compuesto por varios hilos ejecutandose de forma concurrente, esto nos permite operar los sensores a diferentes frecuencias en función de lo lento o rápido que cambia las magnitudes que miden. Además esta estrategia de hilos permite organizar y compartimentar código con su funcionalidad, de esta forma el firmware es mas mantenible y escalable frente a una estrategia de programación mono-hilo.

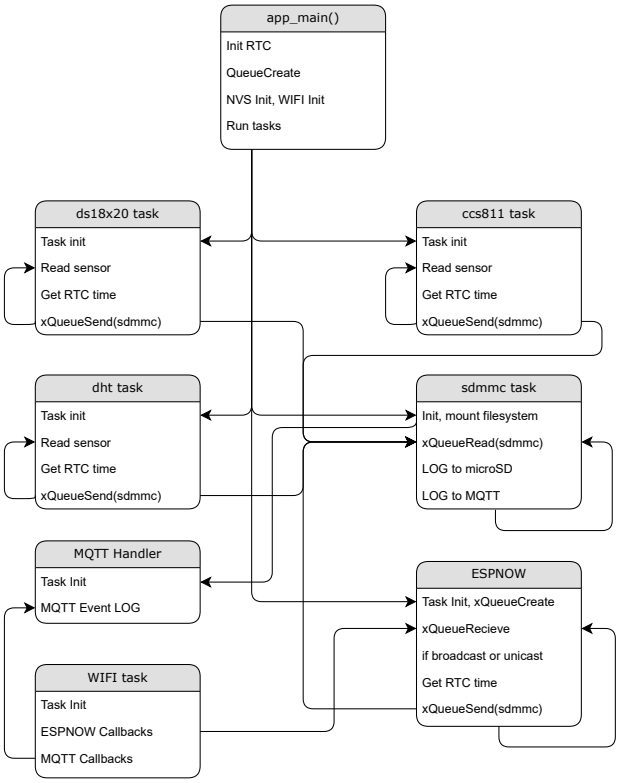

Figura 4.6 Esquema de software.

Tal y como podemos ver en la figura 4.6 tenemos un hilo principal llamado app main. En este hilo inicializamos algunas funciones que son requeridas de forma global como el reloj RTC, el flash NVS y una cola de datos. Posteriormente se van lanzando los demás hilos y acaba terminando el hilo principal. De este modo solo queda operativo los hilos hijos del hilo principal.

Los hilos asociados a los sensores siguen una estrucutra prácticamente similiar, primeramente se inicializa el periférico y protocolo para comunicar con el sensor, se realizan comprobaciones para verificar que se haya inicializado bien y luego entran en bucles infinitos en los que se lee los datos del sensor o sensores, se lee luego el tiempo actual con el RTC y por último se escribe dicha información a una cola de datos dirigida al hilo de sdmmc.

En el hilo sdmmc se centra en recibir datos de una cola y escribirlos a la tarjeta microSD. Primeramente se inicializa el puerto SPI y se verifica que haya una tarjeta microSD conectada. Después se monta el sistema de archivos y finalmente entra en un bucle infinito en el que se espera a que le lleguen datos por la cola de datos. Una vez leido un dato, abre un fichero en el sistema de archivos y escirbe una nueva entrada en formato JSON para su posterior procesamiento en MATLAB. Además de guardar los datos en la tarjeta microSD, se manda también por el protocolo MQTT para su visualizacion a tiempo real en un PC remoto.

Los datos que llegan al hilo de sdmmc se almacenan en un fichero de la tarjeta microSD codificados como un elemento JSON. Éste elemento JSON tiene la siguente forma:

Código 4.1 Formato JSON.

```
{"id ":8," Topic ":"/ DHT−1/Humidity","value":57.900002,"ts":"2022−10−16T16:14:17+02:00"}
{"id ":9," Topic ":"/ DHT−2/Temperature","value":25.700001,"ts":"2022−10−16T16:14:17+02:00"}
{"id ":10," Topic ":"/ DHT−2/Humidity","value":52.700001,"ts":"2022−10−16T16:14:17+02:00"}
{"id ":3," Topic ":"/ DS18B20−0/Temperature","value":25.500000,"ts":"2022−10−16T16:14:17+02:00"}
{"id ":4," Topic ":"/ DS18B20−1/Temperature","value":25.500000,"ts":"2022−10−16T16:14:17+02:00"}
{"id ":5," Topic ":"/ DS18B20−2/Temperature","value":25.500000,"ts":"2022−10−16T16:14:17+02:00"}
{"id ":1," Topic ":"/ CCS811/CO2","value":400.000000,"ts":"2022−10−16T16:14:22+02:00"}
                                                    .                                   .
                                                    .
```

Este formato JSON tiene 4 campos, el primero es el ID, esto nos permite clasificar las medidas según su origen. El siguente es el tópico, esto no es mas que una descripción de la medida y coincide con el nombre del tópico al cual se publica dicha medida en MQTT, luego tenemos el valor de la muestra y finalmente el tiempo en el que se realizó dicha medida recogida del RTC.

5 MQTT

Figura 5.1 MQTT.

MQTT viene de las siglas del inglés Message Queue Telemetry Transport. Es un protocolo de comunicación machine-to-machine que hace uso de colas de mensajes.

Es un protocolo que se basa en el stack TCP/IP. Las conexiones se mantienen abiertas entre máquinas

Está basado en la pila TCP/IP como base para la comunicación. En el caso de MQTT cada conexión se mantiene abierta y se "reutiliza" en cada comunicación. Es una diferencia, por ejemplo, a una petición HTTP 1.0 donde cada transmisión se realiza a través de conexión.

MQTT fue creado por el Dr. Andy Stanford-Clark de IBM y Arlen Nipper de Arcom (ahora Eurotech) en 1999 como un mecanismo para conectar dispositivos empleados en la industria petrolera.

Aunque inicialmente era un formato propietario, en 2010 fue liberado y pasó a ser un estándar en 2014 según la OASIS (Organization for the Advancement of Structured Information Standards).

5.0.1 Funcionamiento

MQTT es un servicio de mensajes de tipo push con estructura publicador y suscriptor (pub-sub). En este tipo de comunicaciones el cliente se conecta con un servidor centralizado que se denomina broker. Para organizar los mensajes, cada clinete dispone sus mensajes en topicos que se organizan jerárquicamente. El cliente puede publicar mensajes en cualquier topico y otros clientes pueden suscribirse a cualquier topico. El broker se encargara de que los mensajes de los publicadores lleguen a sus respectivos suscriptores.

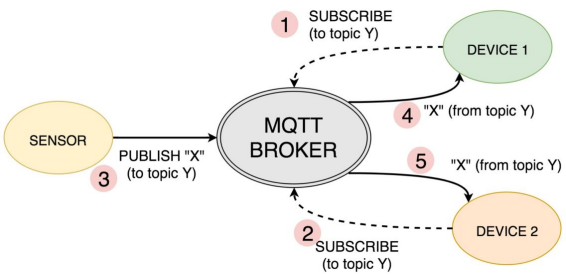

Figura 5.2 Esquema de funcionamiento de MQTT.

Son los clientes que inician una nueva conexion con el broker a través de una conexión TCP/IP, el broker mantiene una lista de clientes con comunicaciones inicializadas y se mantiene dicha conexión abierta hasta que el clinete decida finalizarla. Por defecto MQTT utiliza el puerto 8883 cuando se usa una conexión TLS.

Para una conexión TLS el cliente manda un mensaje tipo CONNEXT que tiene informacion del nombre de usuario, contraseñá de la cominicación y el id del cliente entre otros. El broker entonces contesta con otro mensaje de tipo CONNACK para hacerle llegar al cliente que su conexioón ha sido acceptada (O rechazada en su caso).

Es habitual que el broker de MQTT esté alojado remotamente en la nube proporcionado por algún servicio web. En nuestro caso instalaremos nuestro propio broker en un ordenador. Para ello hemos usado mosquitto.

Mosquitto es un broker de MQTT de licencia libre desarrollado por la fundacion Eclipse. Nos permite instalar un broker en nuestro sistema para probar las comunicaciones antes de desplegarlo en el mundo real. Además es muy liguero (de ahi el nombre mosquito), lo cual lo hace ideal para instalarlo en sistemas de bajo consumo como es una rapsberry pi.

5.1 MQTT Explorer

Para poder visualizar los datos en un ordenador resulta conveniente disponer de un tipo de cliente que se suscriba a todo y que nos muestre por pantalla todos los topicos disponibles, de esta forma podemos verificar el correcto funcionamiento de la red MQTT que tenemos montada. En la actualidad existen muchos clientes genéricos que permiten hacer eso pero uno de los mejores programas se llama MQTT Explorer.

Es un programa que emula un cliente generico y que fue desarrollado bajo licencias Open-Source por Thoamas Nordquist. MQTT Explorer esta caracterizado por su interfaz de usuario amigable que resulta sencillo e intuitivo de usar. Los mensajes y topicos se muestra de forma jerárquica en arbol y es posible deplegar y plotear los valores publicados por cada topic en gráficas. Además se puede registrar dicha información en un fichero para su posterior analisis.

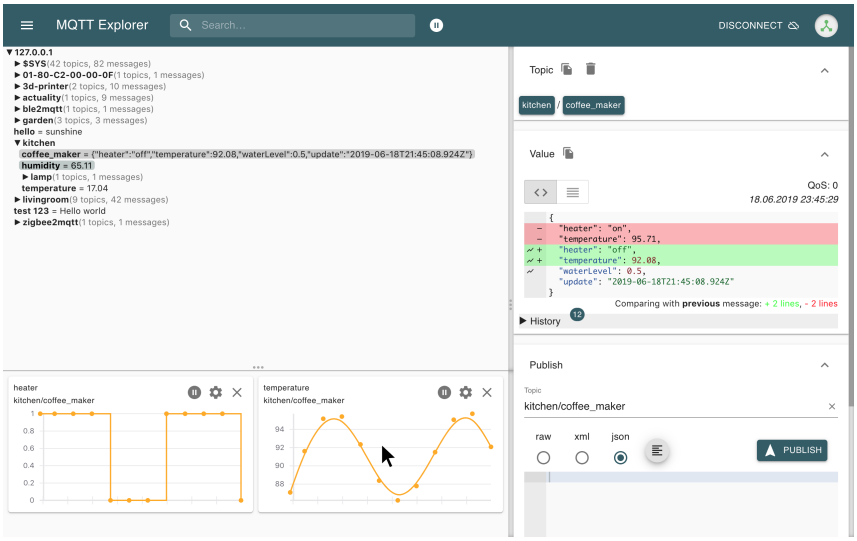

Figura 5.3 MQTT Explorer.

Tambien cabe destacar que este programa esta altamente optimizado para la gestion de cientos de miles de mensajes por minuto. Lo cual lo hace suficiente para volcar un gran numero de datos de cientos de sensores a la vez.

En las figuras 5.4 y 5.5 podemos ver resumido un esquema final de como esta montado el sistema y una foto del mismo. Se puede ver que tenemos una red wifi en la que se puede comunicar la centralita con el ordenador a través de mensajes MQTT y de los nodos a la centralita los mensajes usan el protocolo ESP-NOW. Asimismo a la centralita esta conectada diversos sensores conectados al fancoil para monitorizar y caracterizar el dispositivo.

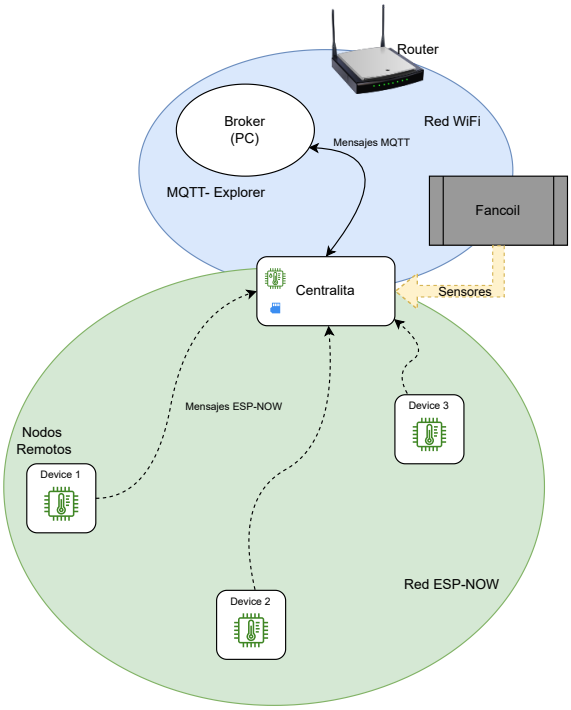

Figura 5.4 Diagrama del montaje.

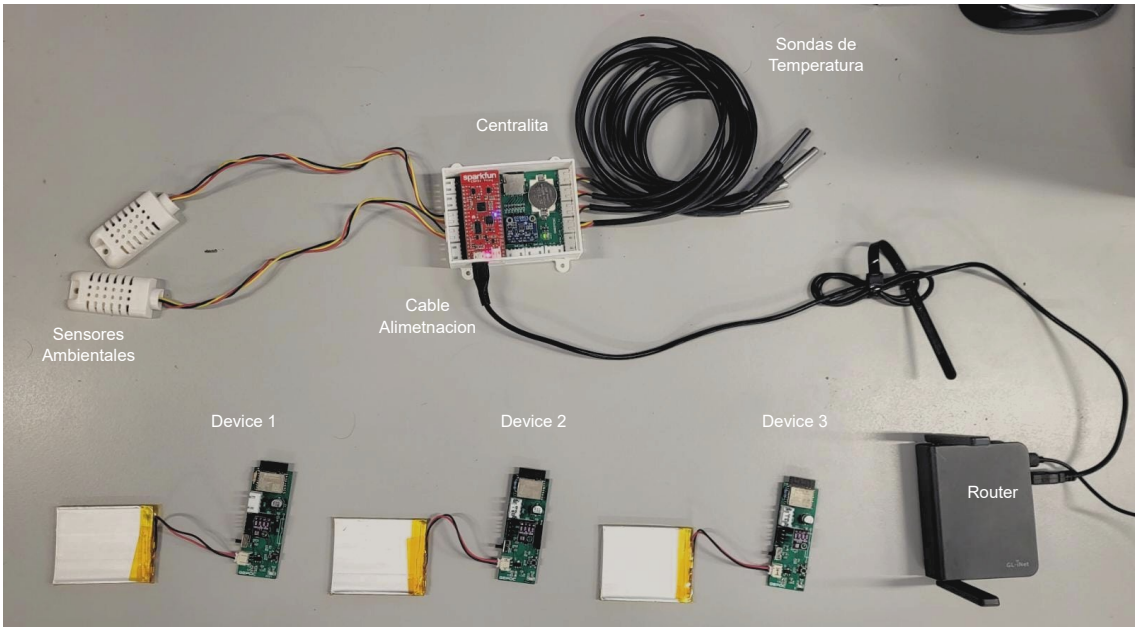

Figura 5.5 foto del montaje.

6 Resultados

Para valorar el funcionamiento del sistema se ha puesto a medir en el despacho F2 de los laboratiorios L1. Se ha dejado los sensores midiendo al aire libre durante 1 semana. Para recoger los datos, se ha extraido la información de la tarjeta microSD de la placa centralita. De ahí se ha usado un código de matlab para procesar el archivo y converirlo en formato CSV, de ahi se ordena los datos según el tipo de sensor y a su vez se ordena según el tiempo de medida. Por último se grafica los datos, los cuales se pueden visualizar desde la figura 6.1 hasta la figura 6.16.

Matlab Code

Código 6.1 Data processor.

```
clear all;
close all;

fid = fopen('LOG.TXT');
fdCSV = fopen('LOGcsv.csv','w');

fprintf(fdCSV,"id,topic,value,time\n");

tline = fgetl(fid);
while true

    tline = fgetl(fid);
    if ~ischar(tline); break; end %end of file
    % disp(tline)
    str = char(tline);
    if isempty(str)
        return
    end
    data = jsondecode(str);

    fprintf(fdCSV,"%d,""%s"",%8.2f,""%s""\n",data.id,data.Topic,data.value,
        data.ts);


end

fclose(fid);
fclose(fdCSV);
```

Código 6.2 Data plotter.

```
clear all;
close all;

LOGcsv = importfile("LOGcsv.csv",[2 inf]);

LOGcsv = sortrows(LOGcsv,'id','ascend');

[ci,ia1,ic1] = unique(LOGcsv.id);
LOGcsv.time = datetime(string(LOGcsv.time),'TimeZone','UTC','Format','yyyy-MM-
    dd''T''HH:mm:ssXXX')

for i=1:ic1(end)-1
    figure;

    plot(LOGcsv.time(ia1(i):ia1(i+1)-1),LOGcsv.value(ia1(i):ia1(i+1)-1));
    title(LOGcsv.topic(ia1(i)))
end

LOGcsv = sortrows(LOGcsv(ia1(end):end,:),'topic')

[ci2,ia2,ic2] = unique(LOGcsv.topic);


for i=1:ic2(end)-1
    figure;

    plot(LOGcsv.time(ia2(i):ia2(i+1)-1),LOGcsv.value(ia2(i):ia2(i+1)-1));
    title(LOGcsv.topic(ia2(i)));
end
i = ic2(end);
figure;
plot(LOGcsv.time(ia2(i):end),LOGcsv.value(ia2(i):end));
title(LOGcsv.topic(ia2(i)));
```

El código que procesa los datos y ordena el archivo CSV resultante se encuentra en los códigos 6.1 y 6.2

6.1 Datos obtenidos

6.1.1 Datos de la centralita

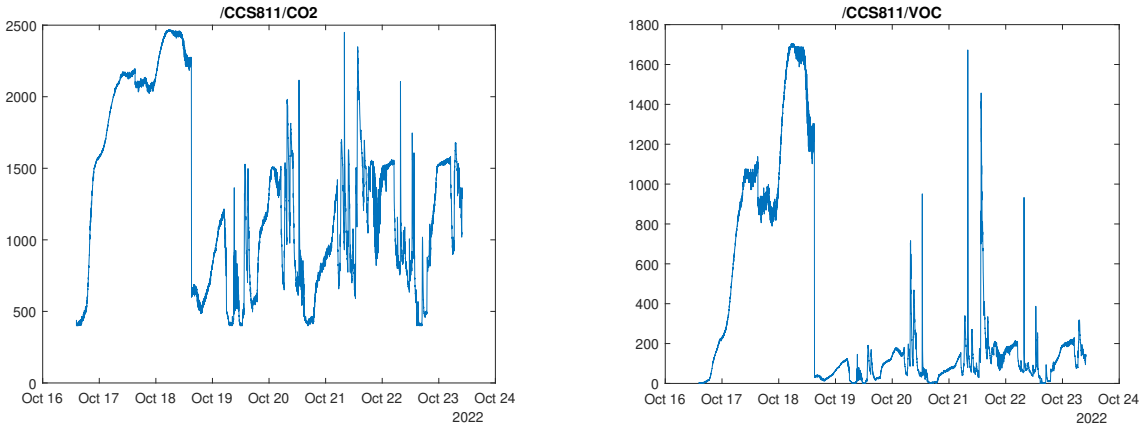

Figura 6.2 Sensor CO2 CCS811 - ppm.

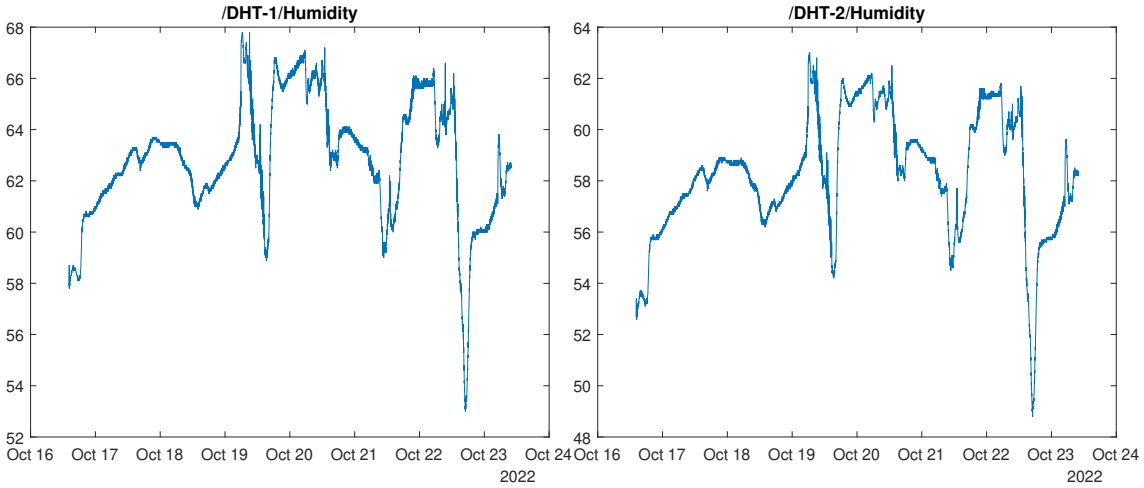

Figura 6.4 Sensor DHT22 1 y 2 - Humedad relativa.

Como se puede ver, las medidas comparadas entre si mismo dan lugar a los mismos resultados, esto nos indica que están funcionando correctamente. Faltaría comprobar con sensores calibrado para ver lo preciso y fiable que son los resultados y si requieren de calibración.

Cabe destacar que en los sensores DS18B20 se ven muy escalonados las medidas por la resolución del sensor, actualmente esta puesto a una resolución de 9 bits pero se puede cambiar hasta 12 bits. Además resulta que en esa semana, dentro de los despachos la variación de temperatura es tan pequeña que no hace variar mucho la medida.

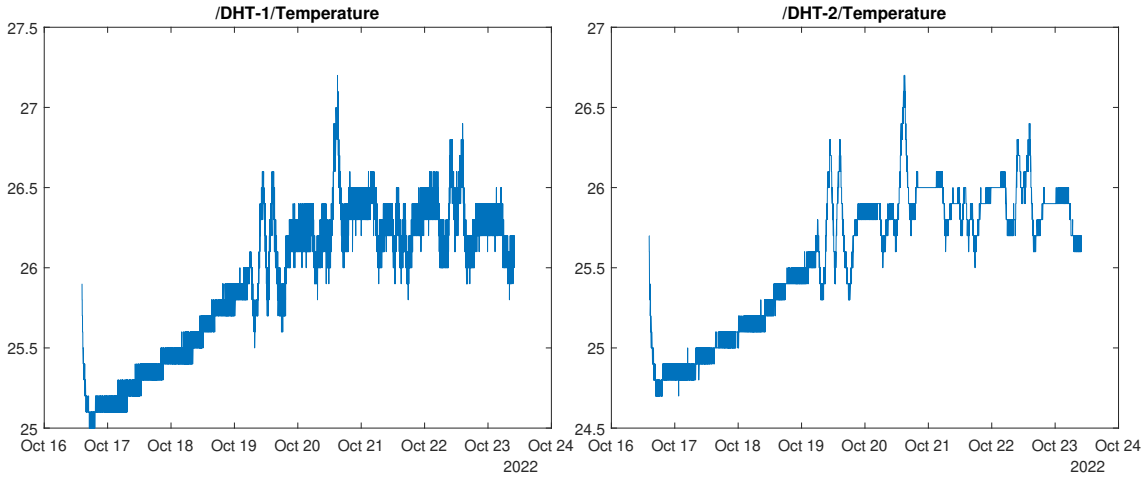

Figura 6.6 Sensor DHT22 2 - Temperatura ambiental.

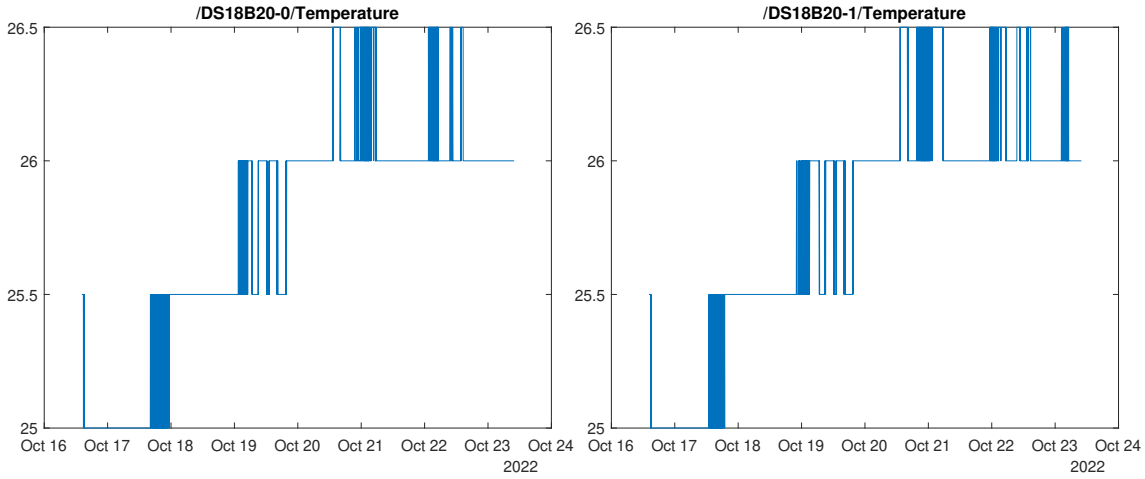

Figura 6.8 Sensor DS18B20 1 y 2 - Temperatura superficial.

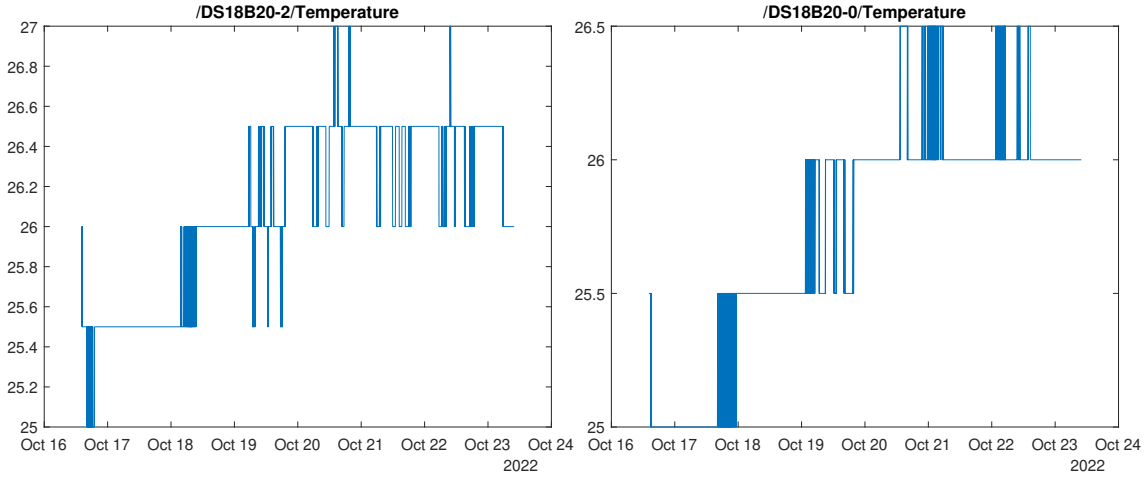

Figura 6.10 Sensor DS18B20 3 y 4 - Temperatura superficial.

6.1.2 Datos de los nodos remotos

En cuanto a los sensores remotos, cabe mencionar que han estado funcionando a batería y que recogen medidas cada 5 minutos. Comparado con los sensores de la centralita, se ven aparentemente con menos ruido

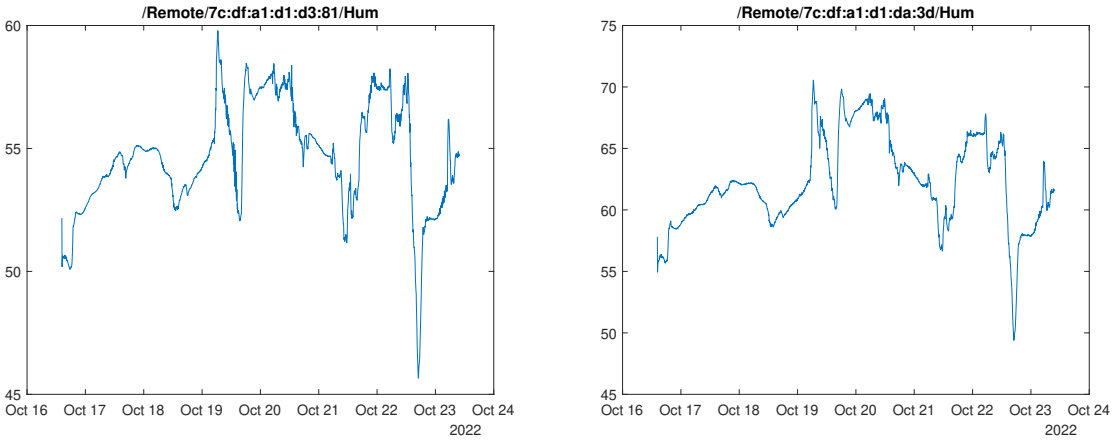

Figura 6.12 Sensor Remoto 1 y 2 - Humedad relativa.

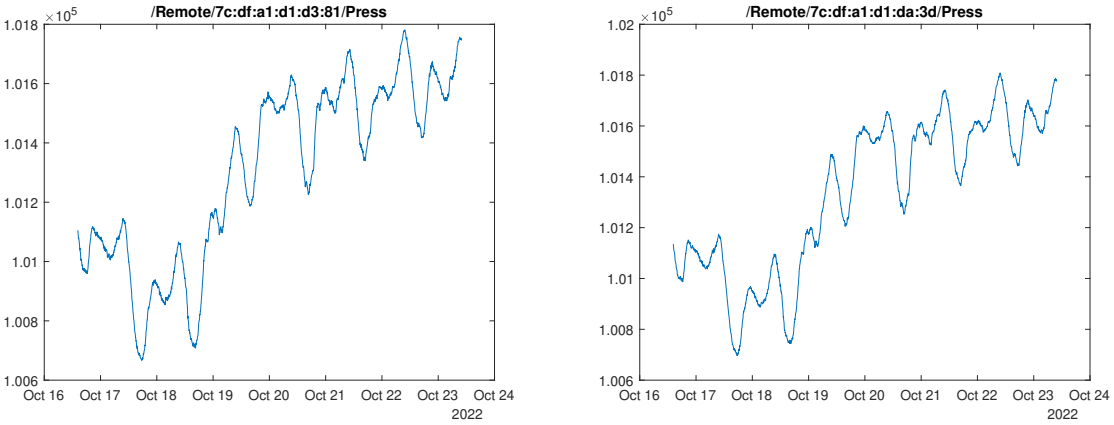

Figura 6.14 Sensor Remoto 1 y 2 - Presión.

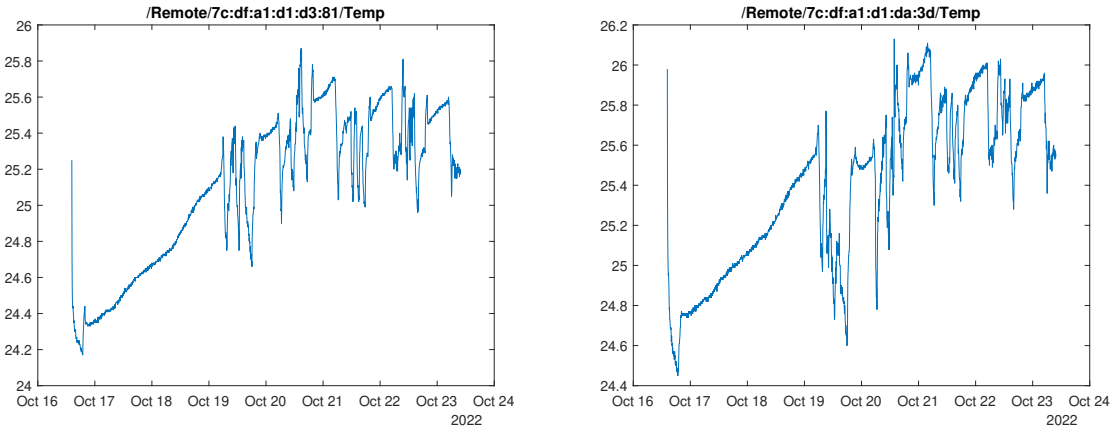

Figura 6.16 Sensor Remoto 1 y 2 - Temperatura.

las medidas de temperatura y humedad, esto puede ser por el hecho de usar un sensor de BOSCH que es una marca reconocida mientras que el DHT22 es uno de bajo coste. El hecho que arrojen los dos nodos los mismos resultados nos da confianza de que los sensores estan funcionando correctamente pero habra que comparar con un sensor externo calibrado para ver si éstas requieren calibración.

6.2 Mejoras futuras

En cuanto a las mejoras futuras se procedera a realizar una nueva versión de la placa de los nodos en las que se incluye la medida de las baterías, la detección de cierre de puertas en modo bajo consumo y medición del porcenaje de apertura de puertas con el encoder. Para realizar estas medidas se tiene que cambiar los puertos/pines a otros que incluyen la funcionalidad de RTC. Los pines RTC son capaces de despertar el micro aún cuando este esta en modo bajo consumo extremo en la que se apaga incluso la memoria del microcontrolador.

Existe la posibilidad de eliminar la circuitería del RTC ya que el microcontrolador ESP32 lleva uno integrado pero sin batería. El problema esta en que cuando se corta la luz hay que volver a sincronizar el tiempo de este RTC interno para que las medidas tengan sentido. Hay dos formas en el que se puede resolver este problema:

- Conectarse a un servidor y solicitar el tiempo actual.
- Mediante MQTT solicitar el tiempo mediante la publicación de un topico (Requiere de otro programa en el PC que de esta respuesta)

Otro aspecto a mejorar es incluir un procesamiento para la medida de la corriente alterna del fancoil, ya que de momento solo se ha conseguido medir la señal sinusoidal en las que se almacena dichas medidas en la memoria microSD para luego ser procesada posteriormente en matlab, el problema con este método es que la memoria se llena muy rápidamente por lo que resulta mejor hacer el procesamiento y cálculo de la potencia en el mismo microcontrolador y reportar el consumo a una frequencia inferior a la de muestreo de la medida para no llenar la memoria.

6.3 Conclusión

Durante este proyecto se ha conseguido abarcar la mayoría de los requisitos del sistema, se intentó alejarse un poco mas de las plataformas Arduino que se usan mas para educación y prototipos básicos y acercase mas a un entorno más profesional como lo es Espressiff y su entonrno ESP-IDF.

En el proyecto se ha conseguido disñear dos placas que funcionan en su mayoría para las aplicaciones descritas en el capítulo 1. Un aspecto que ha resultado especialmente dificultoso es la falta de microcontroladores y de chips en general durante el desarrollo de estas placas, como es sabido, el mundo en el momento de escribir este TFM ha pasado por una crisis de semiconductores a nivel mundial, al punto en que se terminaba de diseñar un circuito y ya era imposible de fabricar por la falta de chips, teniendo que rediseñar partes en función de la disponibilidad de los componentes. Aún asi, se ha conseguido complir con los objetivos propuestos del proyecto y se han coseguido resultados aceptables para su posterior utilización en el modelado de climatización de edificios basado en los datos obtenidos.

7 Anexo

El código fuente se encuentra en el repositorio de github: https://github.com/Richard-Haes-Ellis/esp-nodes.

Índice de Figuras

Índice de Códigos

Bibliografía

Índice alfabético