

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES
UNIVERSIDAD POLITÉCNICA DE MADRID

José Gutiérrez Abascal, 2. 28006 Madrid
Tel.: 91 336 3060
info.industriales@upm.es

www.industriales.upm.es



POLITÉCNICA

INDUSTRIALES

05 TRABAJO FIN DE MASTER

José Bes Ayarzagüena

TRABAJO FIN DE MASTER

RESIDENTIAL HOME EFFICIENCY IMPROVEMENT MONITORING SYSTEM WITH IOT DEVICES

SEPTIEMBRE 2019

José Bes Ayarzagüena

DIRECTOR DEL TRABAJO FIN DE MASTER:
Antonio Barrientos
Guido Lombardi

AGRADECIMIENTOS

En primer lugar, me gustaría agradecer a mis tutores de proyecto en Italia, Dr. Guido Lombardi y Dr. Eros Gian Alessandro Pasero, por ofrecerme la posibilidad de realizar mi Proyecto fin de grado durante mi intercambio Erasmus. Gracias a su colaboración y experiencia el proyecto pudo ser llevado a cabo.

Dar las gracias al Dr. Antonio Barrientos por su atención y consejos sobre cómo enfocar y dirigir el proyecto.

También agradecer a todos los colaboradores y personal del laboratorio de Neurónica del Departamento de Electrónica y Telecomunicaciones (DET) de la Universidad Politecnico di Torino en Turín, Italia, por facilitarme y permitirme el uso de sus equipos para el desarrollo del proyecto. En especial al estudiante Ángel Castro por su ayuda y colaboración en el diseño de los circuitos y PCBs.

Finalmente me gustaría agradecer a mi familia, compañeros y a la casa Botero por su ayuda y apoyo incondicional a lo largo del proyecto, con mención especial a Ana Romero.

RESUMEN

El mundo tiende a estar cada vez más interconectado, existiendo millones de dispositivos compartiendo información entre ellos y con Internet. De esto surge el concepto de Internet de las cosas, en inglés “**Internet of Things**” (IoT). Se trata de redes de sensores que son capaces de monitorizar el entorno y reaccionar ante cambios de este. Esto genera una gran cantidad de información que por sí sola no vale nada, por lo que es necesario analizarla y clasificarla. De esta idea surge la computación en la nube o “**Cloud computing**” que permite almacenar y monitorizar todos los datos desde cualquier parte del mundo y con recursos “ilimitados”. De la fusión de ambos términos nace “**CloudIoT**”.

Sobre este concepto se apoya este trabajo fin de máster. En él se desarrolla una **red de sensores**, en inglés “Wireless Sensor Network” (WSN) capaces de tomar medidas del entorno relacionadas con la **eficiencia energética** en el hogar. Esta red debe de ser capaz de comunicarse a través de Internet, mediante un protocolo de bajo peso como es **MQTT**. Para alojar el **servidor** se utiliza una **Raspberry Pi 3** en la que se busca almacenar y **monitorizar los datos**. La arquitectura del servidor se desarrolla desde la base adaptándose a las necesidades del proyecto. En la Ilustración 1 se puede observar un diagrama de los componentes del proyecto.

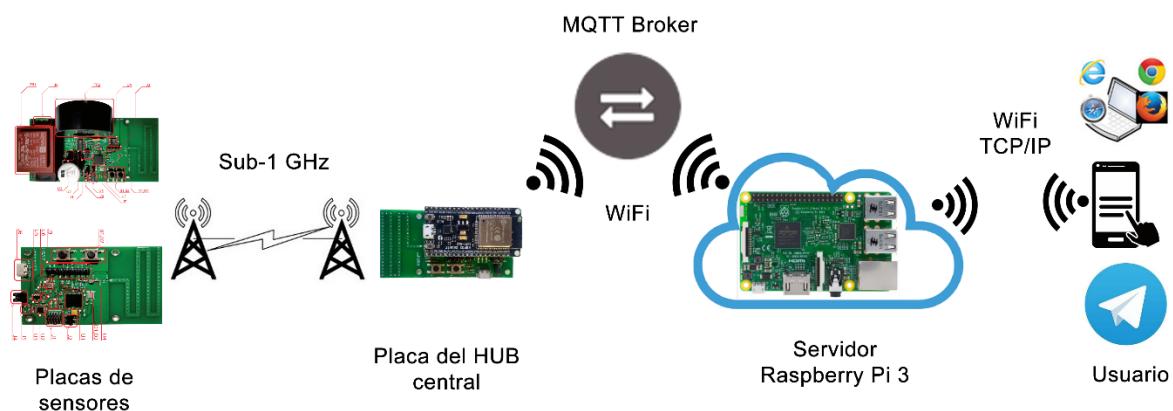


Ilustración 1 - Diagrama general de la plataforma

Describiendo la plataforma desde izquierda, donde se encuentra la parte física, a derecha, donde se encuentra la parte del servidor y el usuario, se tiene:

- **Placas de sensores:** Se trata de dos placas capaces de medir variables del entorno. La placa superior mide el consumo de potencia eléctrica. La inferior mide la temperatura, presión, humedad y calidad del aire.
- **Sub-1 GHz:** Representa el tipo de comunicación por radio frecuencias que existe entre las placas de sensores y la placa del HUB central. Esta comunicación es a través de Sub-1GHz.
- **Placa del HUB central:** Se trata del dispositivo encargado de recibir la información de los sensores y transferirla a través de MQTT al servidor.
- **MQTT Broker:** Se trata del agente encargado de gestionar las comunicaciones MQTT entre la red de sensores y el servidor.

- Servidor: Se trata de la parte del servidor. Este se encarga de almacenar, gestionar y monitorizar la información de la red de sensores. El servidor está alojado físicamente en una Raspberry Pi 3.
- Usuario: Este bloque representa al usuario o encargado de monitorizar los datos. Este puede configurar la plataforma, así como visualizar y descargar los datos de la red de sensores.

Se puede observar cómo existen dos partes diferenciadas dentro del proyecto. Por un lado, se tiene el diseño y desarrollo de la red de sensores, a lo que se le ha denominado **parte Hardware**. Por otro lado, se tiene el desarrollo del servidor y las comunicaciones con este, a lo que se denomina **parte Software**.

Como se observa en la Ilustración 1, la parte Hardware se compone de dos grupos de placas, que forman en total 3 placas diferentes. Esta WSN tiene una topología de **red tipo estrella**, en la que existe un nodo central, la placa del HUB central, al que se conecta el resto de los nodos, las placas de sensores. Entre el nodo central y el resto de los nodos se comunican a través de Sub-1 GHz, lo que permite comunicaciones a largas distancias con bajos consumos.

Para las 3 placas ha sido necesario diseñar los PCBs, ensamblarlas y desarrollar su propio firmware. Para ello, se ha hecho uso de **microcontroladores CC1310 de Texas Instruments** que permiten la comunicación Sub-1 GHz con bajos consumos. Además, disponen de todas las funcionalidades necesarias para cumplir los objetivos del proyecto.

En cuanto a las placas de sensores, se dispone de la **placa de sensores ambientales** y de la **placa de consumo de potencia eléctrica**. La primera, que se observa en la Ilustración 2, permite la obtención de la temperatura, humedad, presión y calidad del aire gracias a su sensor **BME 680**. Este sensor está diseñado para realizar las mediciones tanto en el interior como exterior ya que cuenta con alimentación por cable y por batería. Además, cuenta con una entrada I²C y otra analógico/digital en caso de que se le quiera añadir otro sensor, lo que aporta mayor flexibilidad a la placa.

La segunda placa, que se observa en la Ilustración 3, permite obtener el consumo de potencia eléctrica en la vivienda. Se hace uso de un transformador de tensión y otro de corriente para acondicionar la tensión y corriente de la red y calcular la potencia eléctrica. Este sensor soporta hasta 250 V RMS y 60 A RMS. Dispone de un modo calibración para corregir el desfase ocasionado por los propios transformadores. Esta placa no dispone de baterías ya que dispone de su propio circuito de alimentación a través de la red eléctrica.

Estas dos placas de sensores transmiten los datos al HUB central el cual se muestra en la Ilustración 4, que los envía al servidor. Para ello, el HUB central dispone de dos microcontroladores. Uno se encarga de la comunicación con la red de sensores, el CC1310. El otro se encarga de adaptar el mensaje y enviarlo a través de MQTT al servidor. De este segundo paso se encarga el **ESP 32**. Para comunicarse entre ellos se utiliza la comunicación UART. Gracias a que la placa se comunica de forma inalámbrica puede ser colocada en cualquier lado de la vivienda.

El hecho de tener dos microcontroladores permite dividir y separar los códigos, lo que facilita el trabajo. Además, permite utilizar lenguajes de programación distintos, utilizando **C/C++** para los CC1310 y **MicroPython** para el ESP 32.

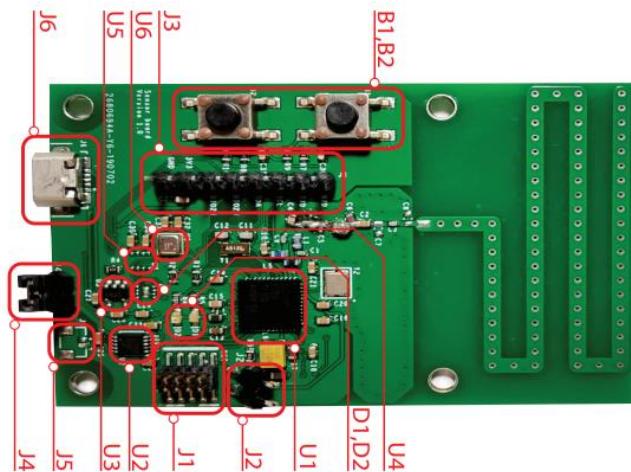


Ilustración 2 - Placa de sensores ambientales

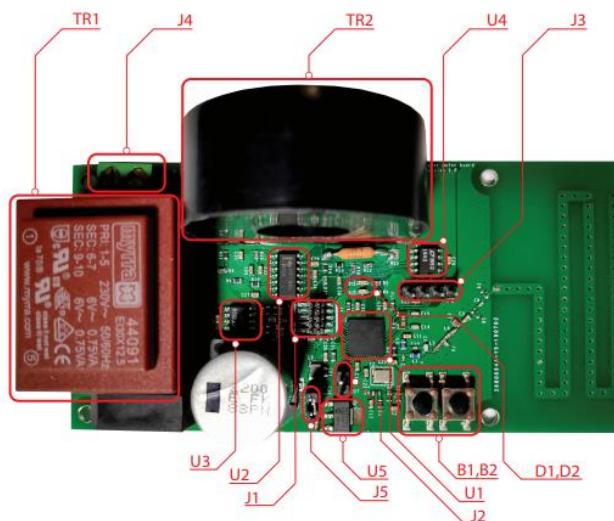


Ilustración 3 - Placa de consumo de potencia eléctrica

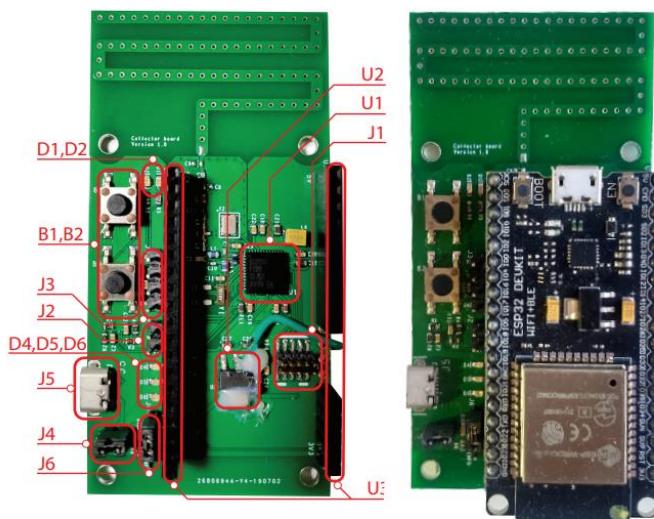


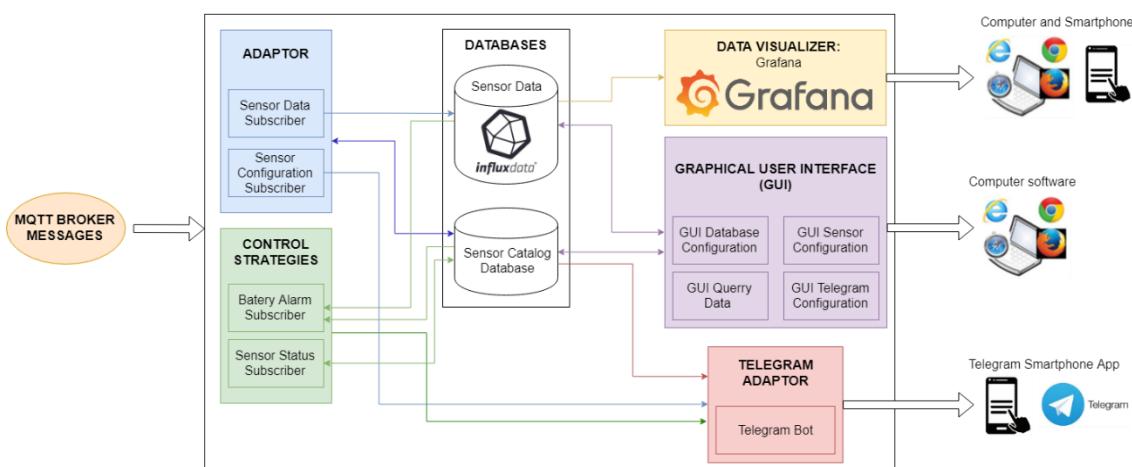
Ilustración 4 - placa del hub central. Izquierda sin ESP 32, derecha con ESP 32

En este punto empieza la parte Software, en la que se desarrolla el servidor y las comunicaciones por completo.

Para la transmisión del mensaje a través de Internet se utiliza el protocolo MQTT. Este permite una comunicación ligera y asíncrona, ideal para redes con anchos de banda reducidos como es el caso. Este protocolo necesita de un Broker que media entre el agente que envía los datos (publisher) y el agente que los recibe (subscriber). Se ha escogido un Broker gratuito y de software libre como es **Mosquitto** que pertenece a Eclipse.

Una vez llega el mensaje al servidor, es necesario que se transforme para procesar la información. En la Ilustración 5 se puede observar la arquitectura del servidor. En esta aparecen todos los bloques que lo componen junto con las comunicaciones que llevan a cabo entre ellos. Dentro de estos bloques se sitúan los **microservicios** que ejecutan pequeñas tareas independientes del servidor.

El lenguaje de programación utilizado para todo el servidor es **Python** ya que es un lenguaje gratuito de software libre con infinidad de módulos que permite una correcta y sencilla gestión de la información.



Entre los bloques que se observan en la Ilustración 5 se tiene:

- Adaptor: Se encarga de transformar los mensajes que llegan por MQTT y almacenar la información relevante en las bases de datos.
- Databases: Almacenan la información de la red de sensores. Se dispone de dos bases de datos. La primera se basa en **InfluxDB**. Se trata de una base de datos especializada en series temporales con un lenguaje parecido a SQL. La segunda base de datos se denomina catálogo. Es un archivo de texto plano en formato JSON. Esta almacena la información de los sensores y configuración básica de la plataforma.
- Data visualizer: Se utiliza el servicio **Grafana**. Se trata de un visualizador de datos que dispone de una amplia galería de gráficas. El servicio está adaptado para la base de datos InfluxDB y para series temporales.
- Control strategies: Se encarga de alertar al usuario ante posibles cambios y errores en la red de sensores. Por un lado, en caso de batería baja se alerta al usuario mediante **Telegram**. Por otro lado, actualiza los sensores del catálogo y avisa por Telegram en caso de que alguno se comporte de forma inesperada.

- Graphical user interface: Interfaz gráfica diseñada para que un usuario profesional pueda configurar la plataforma, así como extraer información de la base de datos sobre las medidas de los sensores.
- Telegram Adaptor: Se encarga de transmitir los mensajes de alerta del servidor a Telegram para avisar al usuario.

El usuario puede visualizar los datos desde un ordenador o dispositivo móvil, aunque no se encuentre en la misma red que el servidor permitiendo disponer de la información en cualquier lado. Además, se cuenta con avisos a través de la aplicación de Telegram en caso de que ocurra algún cambio en la red de sensores.

Para comprobar que las funcionalidades de plataforma funcionan correctamente, tal como se establece en los objetivos, se han realizado una serie de pruebas. Estas pruebas van desde la parte de ensamblado de las PCBs y los componentes hasta la visualización de los datos y alerta al usuario. El resultado final de todas ellas ha sido positivo, lo que permite concluir que se han cumplido con todos los objetivos del proyecto.

Sin embargo, todavía es necesario seguir desarrollando e implementando nuevas funcionalidades y sensores a la plataforma para que esta pueda realizar una mejor monitorización de los parámetros de la eficiencia energética en una vivienda. Este proyecto sienta las bases tanto en la parte Hardware, ya que se tiene como referencia la red y las placas de sensores desarrolladas, como en la parte Software, ya que se tiene un servidor funcional al cual se le pueden añadir nuevos bloques de manera sencilla gracias a la arquitectura de microservicios.

Como se puede concluir, el proyecto abarca una gran cantidad de conocimientos y temas relacionados con la especialidad electrónica y automática de Industriales. Desde conocimientos electrónicos para el diseño de los circuitos y ensamblado de las PCBs; Conocimientos de programación de microcontroladores para poder obtener las medidas y establecer la red de sensores. Así como conocimientos de programación de servicios web para el correcto almacenado y monitorización de los datos. Es por esto, que se considera que el proyecto cumple con las exigencias técnicas y organizativas que se exigen para un trabajo fin de máster de 30 créditos ECTS.

Palabras clave:

Internet of Things, eficiencia energética, Wireless Sensor Network, Sub-1 GHz, CC1310 TI, ESP 32, MQTT, servidor, servicios web, microservicios, Raspberry Pi, InfluxDB, Grafana, Telegram.

Códigos UNESCO:

120325 Diseño de sistemas de sensores

330703 Diseño de circuitos

220302 Elementos de circuitos

ÍNDICE

AGRADECIMIENTOS	2
RESUMEN	3
ÍNDICE	8
ABREVIATURAS Y ACRÓNIMOS	13
ILUSTRACIONES	15
ECUACIONES	18
TABLAS	19
CÓDIGOS	20
1. INTRODUCCIÓN	21
1.1 MOTIVACIÓN, HISTORIA Y APLICACIONES	21
1.1.1 Internet de las cosas	21
1.1.2 Computación en la nube	24
1.1.3 Motivación	26
1.2 PRINCIPALES DESAFIOS	26
1.3 OBJETIVOS DEL PROYECTO	29
1.4 ORGANIZACIÓN DE LA MEMORIA	29
2. ESTADO DEL ARTE	31
2.1 TECNOLOGÍAS DE CONECTIVIDAD	31
2.1.1 6LoWPAN	31
2.1.2 Bluetooth	32
2.1.3 Ethernet	32
2.1.4 LoRa	32
2.1.5 NFC	32
2.1.6 RFID	32
2.1.7 Sigfox	33
2.1.8 Wi-Fi	33
2.1.9 ZigBee	33
2.2 PROTOCOLOS DE COMUNICACIÓN	33
2.2.1 AMQP	34
2.2.2 CoAP	34
2.2.3 MQTT	34
2.2.4 REST (HTTP)	34
2.2.5 TCP/IP	34
2.3 PLATAFORMAS HARDWARE	35
2.3.1 Raspberry Pi	35
2.3.2 Arduino	35

2.3.3	ESP 8266/32	36
2.3.4	Texas Instruments SimpleLink MCU SDK.....	36
2.4	SERVICIOS DE “CLOUD COMPUTING”	37
2.4.1	Thingspeak.....	37
2.4.2	Altair SmartWorks (Carriots)	37
2.4.3	Microsoft Azure.....	38
2.4.4	IBM Cloud.....	38
2.4.5	Google Cloud Platform (GCP)	38
2.4.6	Amazon Web Services (AWS)	39
3.	OBJETIVOS	40
3.1	OBJETIVO GENERAL.....	40
3.2	OBJETIVOS ESPECIFICOS.....	41
3.2.1	Objetivos: Plataforma hardware.....	41
3.2.2	Objetivos: Plataforma software	42
4.	MÉTODOS Y EQUIPO	44
4.1	ORGANIZACIÓN DEL PROYECTO	45
4.2	FUNDAMENTOS TEÓRICOS DEL PROYECTO.....	47
4.2.1	Principios básicos generales	47
4.2.1.1	Redes inalámbricas de sensores y actuadores.....	47
4.2.1.2	Teoría y componentes para la transmisión de la información	50
4.2.1.3	Servicio NTP	52
4.2.1.4	Arquitectura monolítica vs microservicios	53
4.2.1.5	Modelos de servicio	54
4.2.2	Principios básicos específicos	56
4.2.2.1	Tecnología de comunicación: Sub-1 GHz y Wi-Fi.....	56
4.2.2.2	Protocolos de comunicación: MQTT y REST	60
4.2.2.3	Sistemas operativos en tiempo real	64
4.2.2.4	Parámetros asociados a la eficiencia energética	66
4.2.2.5	Protocolos de comunicación entre sensores y MCUs	74
4.3	EQUIPO	77
4.3.1	Hardware.....	77
4.3.1.1	CC1310	77
4.3.1.2	ESP 32	78
4.3.1.3	Raspberry Pi 3 Model B.....	80
4.3.1.4	BME 680.....	81
4.3.1.5	Transformadores	81
4.3.1.6	Otros componentes	83
4.3.2	Software	84
4.3.2.1	Code Composer Studio (CCS).....	84

4.3.2.2	OrCAD.....	84
4.3.2.3	Python y Micropython	84
4.3.2.4	Raspbian Stretch Lite	86
4.3.2.5	InfluxDB.....	86
4.3.2.6	Grafana	87
4.3.2.7	Telegram	87
4.3.2.8	Otros softwares	87
5.	DESARROLLO DEL SISTEMA DE MONITOREO DE LA EFICIENCIA ENERGETICA	88
5.1	PARTE HARDWARE	88
5.1.1	Aspectos comunes de las placas.....	89
5.1.1.1	Hardware.....	89
5.1.1.2	Firmware	89
5.1.1.3	Configuración de la comunicación	93
5.1.1.4	Estados de la comunicación	94
5.1.2	Placa de sensores ambientales	95
5.1.2.1	Introducción.....	95
5.1.2.2	Diagrama de bloques.....	96
5.1.2.3	Esquemáticos.....	96
5.1.2.4	PCB.....	103
5.1.2.5	Cambios en la lista de materiales	103
5.1.2.6	Firmware y funcionalidades	104
5.1.3	Placa de consumo de potencia eléctrica.....	105
5.1.3.1	Introducción.....	105
5.1.3.2	Diagrama de bloques.....	106
5.1.3.3	Esquemáticos.....	106
5.1.3.4	PCB.....	110
5.1.3.5	Cambios en la lista de materiales	111
5.1.3.6	Firmware y funcionalidades	111
5.1.4	Placa hub central.....	114
5.1.4.1	Introducción.....	114
5.1.4.2	Diagrama de bloques.....	114
5.1.4.3	Esquemáticos.....	115
5.1.4.4	PCB.....	118
5.1.4.5	Cambios en la lista de materiales	119
5.1.4.6	Firmware y funcionalidades	119
5.2	PARTE SOFTWARE	119
5.2.1	Comunicación con el servidor: ESP 32	119
5.2.2	Servidor	125
5.2.3	Servidor: almacenamiento	130

5.2.4	Servidor: adaptador	137
5.2.5	Servidor: visualizador	142
5.2.6	Servidor: GUI	145
5.2.7	Servidor: estrategias de control	156
5.2.8	Servidor: Adaptador de Telegram	162
6.	EXPERIMENTOS	165
6.1	INTRODUCCIÓN	165
6.2	PRUEBAS	165
6.2.1	Test de continuidad de PCBs	165
6.2.2	Test de carga de firmware	166
6.2.3	Test de perdida de red del hub central.....	166
6.2.4	Test de comunicación entre placas de sensores y hub central	167
6.2.5	Test de envío y recepción de mensajes al servidor por parte del hub central	167
6.2.6	Test de almacenamiento y visualización del mensaje.....	168
6.2.7	Test de alarma de batería.....	169
6.2.8	Test de estado del sensor.....	169
6.2.9	Test de extracción de información	170
7.	RESULTADOS Y ANÁLISIS	171
7.1	INTRODUCCIÓN	171
7.2	PRUEBAS	171
7.2.1	Test de continuidad de PCBs	171
7.2.2	Test de carga de firmware	172
7.2.3	Test de perdida de red del hub central.....	172
7.2.4	Test de comunicación entre placas de sensores y hub central	172
7.2.5	Test de envío y recepción de mensajes al servidor	173
7.2.6	Test de almacenamiento y visualización del mensaje.....	174
7.2.7	Test de alarma de batería.....	175
7.2.8	Test de estado del sensor.....	176
7.2.9	Test de extracción de información	177
8.	CONCLUSIONES	178
9.	LINEAS FUTURAS	180
10.	PLANIFICACIÓN TEMPORAL Y PRESUPUESTO	184
10.1	PLANIFICACIÓN TEMPORAL	184
10.2	PRESUPUESTO	190
11.	BIBLIOGRAFÍA	193
ANEXO I:	COMPONENTES UTILIZADOS	198
A1.	PLACA DE SENsoRES AMBIENTALES	198
A2.	PLACA DE CONSUMO DE POTENCIA ELÉCTRICA	199
A1.	PLACA DEL HUB CENTRAL	201

ABREVIATURAS Y ACRÓNIMOS

Advance Message Queuing Protocol	AMQP
Application Programming Interface	API
Additive White Gaussian Noise	AWGN
Amazon Web Services	AWS
Bluetooth Low Energy	BLE
Code Composer Studio	CCS
Constrained Application Protocol	CoAP
United States Defense Advanced Research Projects Agency	DARPA
Google Cloud Platform	GCP
Graphical User Interface	GUI
Hypertext Transfer Protocol	HTTP
Centro de operaciones, concentrador	HUB
Inter Integrated Circuits	I2C
Infrastructure as a Service	IaaS
Interruptor de Control de Potencia	ICP
Integrated Development Enviroment	IDE
Institute of Electrical and Electronics Engineers	IEEE
European Research Cluster on the Internet of Things	IERC
Internet of Things	IoT
Industrial, scientific and medical	ISM
JavaScript Object Notation	JSON
Quality of System	QoS
Local Area Networks	LAN
Machine to Machine	M2M
Microcontroller	MCU
Micro Electro-Mechanical System	MEMS
Master Input, Slave Output	MISO
Master Output, Slave Input	MOSI
Message Queuing Telemetry Transport	MQTT
Near Field Communication	NFC
National Institute of Standards and Technology	NIST
Network Time Protocol	NTP
Objetivos de desarrollo sostenible	ODS
Platform as a Service	PaaS
Printed Circuit Board	PCB
Physical layer	PHY
Power Line Control	PLC
Portable Operating System Interface	POSIX
Pequeña Y Media Empresa	PYME
REpresentational State Transfer	REST
Radio Frequency Identification	RFID
Root-Mean_square	RMS
Real Time Operating Systems	RTOS
Software as a Service	SaaS
Serial Clock	SCL/ SCLK
Serial Data	SDA
Software Development Kit	SDK
Structured Query Language	SQL

Abreviaturas y acrónimos

Surface-Mont Device	SMD
Signal Noise Ratio	SNR
System on a Chip	SoC
Serial Peripheral Interface	SPI
Short-Range Devices	SRD
Slave Selector	SS
Secure Sockets Layer	SSL
Transfer Control Protocol/ Internet Protocol	TCP/IP
Texas Instruments Real Time Operating Systems	TI RTOS
Transport Layer Security	TLS
Universal Asynchronous Receiver-Transmitter	UART
User Datagram Protocol	UDP
Universal Resource Identifier	URI
Uniform Resource Locator	URL
Universal Serial Bus	USB
World Wide Web Consortium	W3C
Wireless Sensor Network	WSN
X as a Service	XaaS
eXtensible Markup Language	XML

Tabla 1: Abreviaturas y acrónimos

ILUSTRACIONES

Ilustración 1 - Diagrama general de la plataforma	3
Ilustración 2 - Placa de sensores ambientales	5
Ilustración 3 - Placa de consumo de potencia eléctrica	5
Ilustración 4 - placa del hub central. Izquierda sin ESP 32, derecha con ESP 32	5
Ilustración 5 - Arquitectura del servidor y usuario.....	6
Ilustración 6 - Global number of Connected Devices	21
Ilustración 7- Most IoT projects in Smart City	23
Ilustración 8 - Porcentage of electronic waste.....	28
Ilustración 9 - The OSI model, a Wi-Fi® stack example and the 6LoWPAN stack.....	31
Ilustración 10 - Cloud, Fog and Edge layers of IoT platforms architecture.....	44
Ilustración 11 - Edge and Fog layer: Hardware diagram.	45
Ilustración 12- Cloud layer: Software diagram.....	46
Ilustración 13 - Wireless Sensor Actuator Network	48
Ilustración 14 - Common WSAN topologies	49
Ilustración 15 - Chain WSAN topology	49
Ilustración 16 - Transmission chain.....	50
Ilustración 17 - NTP service structure	52
Ilustración 18 - Monolithic vs Microservices architecture	53
Ilustración 19 - On-Premises, IaaS, PaaS, SaaS services	55
Ilustración 20 – Technologies range comparision	57
Ilustración 21 - TI 15.4-Stack layers.....	59
Ilustración 22 - General architecture of MQTT communication.....	61
Ilustración 23 - MQTT communication of the platform	63
Ilustración 24 - TI RTOS modules	65
Ilustración 25 - Task states diagram	66
Ilustración 26 – Power triangle.....	70
Ilustración 27 - Types of phase shifting depending on the load	71
Ilustración 28 - Voltage and current shift phase representation. Current lags the voltage	71
Ilustración 29 - Voltage and current shift phase representation. Current leads the voltage	71
Ilustración 30 - Power measurement block diagram.....	72
Ilustración 31 - voltage grid waveform representation	73
Ilustración 32 - voltage waveform representation in the MCU	73
Ilustración 33 - voltage waveform representation in the grid (upper) and in the MCU (bottom)	73
Ilustración 34 - UART communication.....	75
Ilustración 35 - UART communication diagram	75
Ilustración 36 - I2C communication diagram	76
Ilustración 37 - I2C Communication	76
Ilustración 38 SimpleLink™ Sub-1 GHz CC1310 wireless MCU LaunchPad™ development kit (SDK)	77
Ilustración 39 - CC13XX block diagram	78
Ilustración 40 - ESP 32 Development board	79
Ilustración 41 - ESP 32 block diagram	79
Ilustración 42 - Raspberry pi 3 specifications	80
Ilustración 43 - BME 680 sensor	81
Ilustración 44 - Current transformer	82
Ilustración 45 - Voltage transformer	82
Ilustración 46 - Current transformer ilustration	82
Ilustración 47 – Simple platform diagram	88

Ilustración 48 - Environmental sensor block diagram	96
Ilustración 49 - Environmental board. CC1310 schematics	98
Ilustración 50 - Environmental board. Antenna schematics.....	99
Ilustración 51 - Environmental board. Bypass schematics	99
Ilustración 52 - Environmental board. LEDs and buttons schematics	100
Ilustración 53 - Environmental board. External ports schematics	100
Ilustración 54 - Environmental board. BME 680 and 260 sensors schematics	101
Ilustración 55 - Environmental board. Batery charger schematics	101
Ilustración 56 - Environmental board. Buck/boost converter schematics	102
Ilustración 57 - Environmental board. Batery fuel gauge schematics	102
Ilustración 58 - Environmental sensor PCB.....	103
Ilustración 59 - Electric power consumption sensor block diagram.....	106
Ilustración 60 - Power meter board. CC1310 schematics.....	107
Ilustración 61 - Power meter board. External ports schematics.....	108
Ilustración 62 - Power meter board. Power supply schematic	108
Ilustración 63 - Power meter board. Voltage conditioning schematic.....	109
Ilustración 64 - Power meter board. Current conditioning schematic	109
Ilustración 65 - Power meter board. Signal offset schematic	110
Ilustración 66 - Power meter board PCB	111
Ilustración 67 - Power meter board. Calibration configuration	113
Ilustración 68 - Central hub block diagram	114
Ilustración 69 - Central hub board. CC1310 schematics	115
Ilustración 70 - Central hub board. ESP 32 schematics	116
Ilustración 71 - Central hub board. Configuration jumper schematics.....	117
Ilustración 72 - Central hub board. LED indicators schematics	117
Ilustración 73 - Central hub board. Power supply schematics	118
Ilustración 74 - Central hub board PCB	119
Ilustración 75 - ESP32 firmware and files.....	121
Ilustración 76 - ESP 32 firmware flowchart.....	122
Ilustración 77 - ESP 32 timer interrupt flowchart	123
Ilustración 78 - ESP 32 configuration mode flowchart	125
Ilustración 79 - Hardware - Software MQTT communication	126
Ilustración 80 - Server architecture diagram.....	127
Ilustración 81 - InfluxDB data type	131
Ilustración 82 - Sensor Catalog Database files.....	136
Ilustración 83 - Adaptor block files	138
Ilustración 84 - Sensor Data Subscriber flowchart.....	139
Ilustración 85 - Sensor Configuration Subscriber flowchart	141
Ilustración 86 - Add Data source menu of Grafana	144
Ilustración 87 - Grafana dashboard example	145
Ilustración 88 - GUI block files	146
Ilustración 89 - GUI main Window	147
Ilustración 90 - InfluxDB login GUI	147
Ilustración 91 - Influxdb login error GUI	147
Ilustración 92 - GUI Database Configuration window	148
Ilustración 93 - New Database and user GUI windows.....	148
Ilustración 94 - Modify database and user GUI windows	149
Ilustración 95 - GUI Sensor Configuration window	150
Ilustración 96 - Add new sensor GUI window	150
Ilustración 97 - Modify sensor GUI window	151
Ilustración 98 - GUI Query Data window	152
Ilustración 99 - Search funtionality example GUI Query	154
Ilustración 100 - Search funtionality graph example GUI Query	154
Ilustración 101 - Modify Tags GUI window	155
Ilustración 102 - GUI Telegram Configuration window	156

Ilustración 103 - Control strategies files	157
Ilustración 104 - Batery Alarm Subscriber flowchart.....	158
Ilustración 105 - Sensor Status Subscriber flowchart.....	161
Ilustración 106 - Sensor Config Subs. Add new sensor Telegram message	163
Ilustración 107 - Sensor Config Subs. modify sensor Telegram message.....	164
Ilustración 108 - Batery Alarm Subs. low batery first advise Telegram msg	164
Ilustración 109 - Batery Alarm Subs. low batery second advise Telegram msg.....	164
Ilustración 110 - Sensor Status warning active status Telegram msg.....	164
Ilustración 111 - Sensor Status warning inactive status Telegram msg.....	164
Ilustración 112 - Test de carga de firmware. Placas de sensores	172
Ilustración 113 - Test de carga de firmware. Placa del hub central	172
Ilustración 114 - Test de envío y recepción de mensajes al servidor. ESP 32.....	173
Ilustración 115 - Test de envío y recepción de mensajes al servidor. Sensor Data Subscriber	173
Ilustración 116 - Test de almacenamiento y visualización. Grafana en ordenador.	174
Ilustración 117 - Test de almacenamiento y visualización. Grafana en móvil.	174
Ilustración 118 – Test de almacenamiento y visualización. Sensor Configuration Subscriber	175
Ilustración 119 - Test de almacenamiento y visualización. Telegram	175
Ilustración 120 - Test de alarma de batería. Batery Alarm Subscriber.....	175
Ilustración 121 - Test de alarma de batería. Telegram	176
Ilustración 122 - Test de alarma de batería. Grafana	176
Ilustración 123 - Test de estado del sensor. GUI Sensor Configuration inactive sensor	176
Ilustración 124 - Test de estado del sensor. Sensor Status Subscriber.....	176
Ilustración 125 - Test de estado del sensor. GUI Sensor Configuration active sensor	176
Ilustración 126 - Test de estado del sensor. GUI Sensor Configuration inactive sensor. Telegram	177
Ilustración 127 - Test de extracción de información. GUI Query Data.....	177
Ilustración 128 - Test de extracción de información. Archivo .csv	177
Ilustración 129 - Sustainable Development Goals.....	179
Ilustración 130 - Gantt diagram part 1.....	187
Ilustración 131 - Gantt diagram. Part 2	188
Ilustración 132 - EDP	189
Ilustración 133 - Budget evolution	192

ECUACIONES

Ecuación 1 - Shanon equation	50
Ecuación 2 - Friis equation.....	51
Ecuación 3 - Noise power model as AWGN from Johnson Nyquist model	51
Ecuación 4 - Developed Shanon equation	51
Ecuación 5 - Discrete equation of electrical power and electrical energy	69
Ecuación 6 - Continuous equation of electrical power and electrical energy	69
Ecuación 7 – Active power, reactive power and apparent power	70
Ecuación 8 - Shift phase angle equation	72
Ecuación 9 - Nyquist Theorem.....	72
Ecuación 10 - Transformation relation	82
Ecuación 11 - Phase difference calibration	112
Ecuación 12 - Phase difference calculation.....	113
Ecuación 13 - linear amortización project equation	190

TABLAS

Tabla 1: Abreviaturas y acrónimos.....	14
Tabla 2 - Aplicaciones según la regulación en las telecomunicaciones europeas	59
Tabla 3 - Limits of non-specific SRD	60
Tabla 4 - Duty cycle time limits	60
Tabla 5 - HTTP/REST methods	63
Tabla 6 - HTTP status codes	63
Tabla 7 - Temperature, humidity and pressure required values	69
Tabla 8 - BME 680 specifications.....	81
Tabla 9 - Sensor board states.....	94
Tabla 10 - Environmental sensor. Changed components.....	104
Tabla 11 - Power meter board. measurement signals.....	107
Tabla 12 - InfluxDB data type project example.....	131
Tabla 13 - Catalog GET requests	136
Tabla 14 - Catalog POST request.....	136
Tabla 15 - Catalog PUT requests.....	137
Tabla 16 - Catalog DELETE request.....	137
Tabla 17 – Project stages	186
Tabla 18 - Hardware budget	191
Tabla 19 - Total project budget	191
Tabla 20 - Environmental sensor board components	199
Tabla 21 - Power sensor board components.....	201
Tabla 22 - Central HUB board components	202

CÓDIGOS

Código 1 - Firmware variable modifications	90
Código 2 - Application/Sensor.c firmware. MAC address	91
Código 3 - Application/sensor.c firmware. Message transformations	91
Código 4 - Application/sensor.c firmware. Sensor states	92
Código 5 - Firmware communication configuration	93
Código 6 – Firmware PHY configuration	93
Código 7 - Environmental sensor. BME680 firmware functions.....	104
Código 8 - Environmental sensor. bme680.c functions	105
Código 9 - ESP 32 configuration file	121
Código 10 - ESP 32 entry message	123
Código 11 - ESP 32 exit message	124
Código 12 - MQTT publish topic	124
Código 13 - MQTT subscribe topic.....	126
Código 14 - Python 3 installation Linux command.....	129
Código 15 - Python 3 modules installation Linux command	129
Código 16 - Influxdb client Python instance	132
Código 17 - Insert Influxdb sata series command	132
Código 18 – InfluxDB insert data message format	133
Código 19 - Query data from Influxdb command.....	133
Código 20 - SELECT SQL language instance.....	133
Código 21 - Influxdb Python module installation Linux command.....	133
Código 22 - Initial catalog file	134
Código 23 - Sensor register in the catalog	135
Código 24 - Cherrypy Python module installation Linux command.....	135
Código 25 - Catalog request example	136
Código 26 - Requests Python module installation Linux command	138
Código 27 - Constants in ADAPTOR Sensor Data Subscriber file.....	139
Código 28 - Write message to Influxdb in JSON format example	140
Código 29 - Grafana installation Linux commands	143
Código 30 - Configuration and execution of Grafana Linux commands	143
Código 31 - Basic Grafana Linux commands	143
Código 32 - Grafana URL	144
Código 33 - WxPython, Pandas and Numpy Python module installation Linux commands	146
Código 34 - csv file format example	154
Código 35 - Sensor Status Subscriber constants	160
Código 36 - Telebot Python module installation Linux command	162
Código 37 - Telegram chat id URL request.....	163
Código 38 - Telegram chat id JSON message	163

1. INTRODUCCIÓN

1.1 MOTIVACIÓN, HISTORIA Y APLICACIONES

Durante los últimos años, la automatización y la electrónica han introducido muchas mejoras que han hecho cambiar la vida de las personas. Estas mejoras en muchos casos son ventajas, pero también tienen sus inconvenientes, y estos crecen cada vez que el mundo se vuelve más interconectado. Es por ello necesario que el desarrollo se haga de una manera sostenible, en el que el ser humano se lucre de la tecnología actual sin comprometer a las generaciones venideras y haciendo frente a los grandes problemas con los que se encuentra.

1.1.1 Internet de las cosas

Realizando un pequeño recorrido a lo largo de los años, se puede observar como la tendencia de la interconectividad y del uso de dispositivos inalámbricos se ha incrementado de forma exponencial, como se observa en la Ilustración 6 (Knud, 2018). Desde que el sistema por cable empezó a ser sustituido por el inalámbrico, este último ha sufrido un crecimiento sin precedentes. Esto ha permitido sistemas más flexibles y con costes reducidos. La mejora en el estado del arte de la tecnología junto con el desarrollo de protocolos de comunicación ha permitido el desarrollo de dispositivos de bajo coste y consumo, que pueden transmitir datos a largas distancias. Gracias a todo esto aparece lo que se denomina “Internet of Things” (IoT) y las redes de sensores inalámbricos o “Wireless Sensor Network” (WSN)

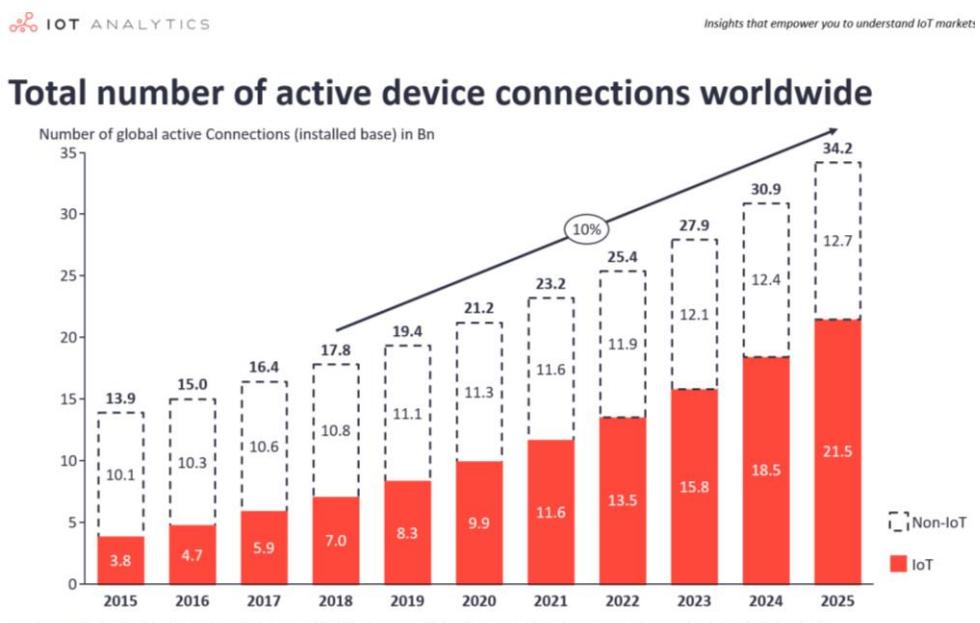


Ilustración 6 - Global number of Connected Devices¹

Las redes de sensores inalámbricos datan de 1980. El “United States Defense Advanced Research Projects Agency”, o (DARPA) aplica este concepto al ámbito militar,

¹ (Knud, 2018)

en el que redes de diferentes nodos (sensores y actuadores) de bajo coste intercambian información entre sí (IEC, 2018). Posteriormente se extiende al ámbito comercial, dando lugar a lo que se conoce como Internet de las Cosas. Esto fue posible gracias al desarrollo de nuevas tecnologías y protocolos como los Microsistemas Electrónico-Mecánicos, en inglés “Micro Electro-Mechanical System” o (MEMS) en los que se combinan partes mecánicas y electrónicas en el mismo dispositivo.

Existen muchas definiciones de Internet de las Cosas, pero muchas de ellas no son capaces de cubrir todos los aspectos. Observando la definición que aporta el European Research Cluster on the Internet of Things (IERC) en su “Strategic Research Agenda” de 2009, se define como una infraestructura global, dinámica y auto configurable, basada en protocolos de comunicación donde las “cosas” están perfectamente integradas con la red de información, siendo una parte de “Future Internet”. Estas “cosas” formarían parte del entorno comunicándose con este y entre sí, de forma que puedan compartir información y reaccionar ante cambios. (Vermesan, Friess, Guillemin, & Gusmeroli, 2009)

Esta definición puede ser bastante completa en términos tecnológicos, pero también se debe tener en cuenta otras características en la red, como la calidad de esta, con variables como robustez, escalabilidad, y sobre todo algo fundamental, el usuario, ya que este se verá condicionado por esta tecnología, siendo parte activa en el mismo.

El Internet de las Cosas aparece en torno al 2000, en un intento de mejora de las cadenas de suministro por la empresa Auto-ID Center (Ashton, 2009), de forma que el seguimiento del producto se hiciera de manera automática. Esto permitiría que se pudiera monitorizar el estado de los diferentes componentes de la cadena, recolectando datos y procesándolos para mejorar la eficiencia. Por lo tanto, en el ámbito de la logística supuso un gran avance, pero a su vez un reto, ya que habría que crear redes complejas e interconectadas de dispositivos.

Para poder realizar la transición con éxito de la tecnología inteligente actual como ordenadores y móviles al nuevo panorama en el que cualquier objeto es “inteligente” es necesario una serie de características:

- Usuarios con capacidad y confianza en la transmisión de datos y “cosas” interconectadas compartiendo información
- Establecer una infraestructura de comunicación y desarrollar arquitecturas y estándares que permitan transmitir y procesar la ingente cantidad de información para hacerla accesible
- Desarrollo de algoritmos capaces procesar la información de manera eficiente y automática.

Con estas características tanto en la sociedad como en la tecnología, la implantación de redes inteligentes podrá ser llevada a cabo. (Gubbi, Buyya, Marusic, & Palaniswami, 2012)

Uno de los grandes problemas y a su vez una ventaja, es la posibilidad de elección entre proveedores de servicios de IoT. Por una parte, existe gran competitividad entre empresas, que hace que los precios sean cada vez más reducidos, pero por otra parte existe una gran heterogeneidad de servicios, ya que no existen estándares de tecnologías y arquitecturas. Esto hace que haya infinidad de plataformas de IoT en el mercado, con gran inestabilidad, ya que muchas de ellas desaparecen con gran velocidad. (Uckelmann, Harrison, & Michahelles, 2011)

El número de aplicaciones de las plataformas IoT también se encuentra en crecimiento, abriendose paso en numerosas áreas. Desde las ciudades inteligentes, o “Smart cities”, pasando por la sanidad, agricultura o en la industria, con el concepto de Industria 4.0, en la que las maquinas operan de manera interconectada, recibiendo y

enviando datos de las variables fundamentales en el desempeño de su trabajo. Como se puede observar, el concepto de Internet de las Cosas es multidisciplinario, por lo que el término “cosa” se refiere a diferentes tipos de objetos en función de que ámbito se refiere.

Hoy en día, los mayores campos en los que las plataformas IoT están teniendo gran auge son la industria, en el estudio ambiental y en la sociedad. En cuanto a la industria, aparece la Industria 4.0, como ya se ha explicado, lo que permite tener mayor control en el ciclo de vida del producto. En el estudio ambiental, permite analizar variables ambientales, para el estudio de la calidad del aire, pero también en la mejora de eficiencia en ganadería y cultivos. En lo referente a la sociedad, se pueden estudiar el uso de determinados espacios públicos para mejorar la eficiencia energética y mejorar la calidad de servicio al ciudadano. (Vermesan, Friess, Guillemin, & Gusmeroli, 2009)

En la Ilustración 7 se observa los resultados de un estudio en 2018 sobre las aplicaciones en IoT que mayor uso y desarrollo están experimentando, después de observar 1600 proyectos.

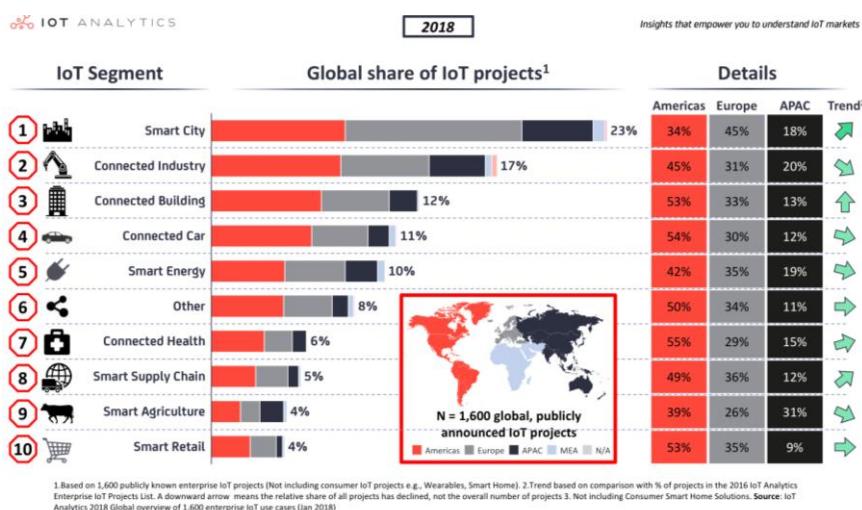


Ilustración 7- Most IoT projects in Smart City²

Además de las aplicaciones ya descritas, se encuentran en muchos otros campos como el militar, en el que permite monitorizar la cantidad de suministros, de munición, posiciones de las tropas, etc.

Todas estas aplicaciones se pueden englobar en cuatro grupos más amplios que son (1) Empresa, (2) Servicios públicos, (3) personal y hogar y (4) Transporte. Sin embargo, estos cuatro grupos no son independientes entre sí. Por ejemplo, la información obtenida por un individuo en su hogar, como puede ser la medida del consumo eléctrico y su historial de consumo, pueden ser compartidas con la empresa eléctrica para conocer la demanda eléctrica con mayor precisión y así ofrecer un servicio de mayor calidad y sostenible.

² (Scully, 2018)

1.1.2 Computación en la nube

Como es lógico, la información que está generando la inmensa cantidad de dispositivos debe ser almacenada en algún sitio. Antiguamente, aunque hoy en día sigue siendo la práctica más popularizada, cada compañía tenía que implementar su propia infraestructura para poder llevar a cabo esta tarea. Sin embargo, con la aparición de la computación en la nube o “Cloud computing” este paradigma da un cambio radical. Este término aparece en 1961 por John McCarthy, que ya sienta las bases de la tecnología “Time-Sharing”, en el que los ordenadores comparten recursos, de ahí el término tiempo compartido, para llevar a cabo operaciones computacionales y aplicaciones para poder venderlas como servicio. (Boxbyte, 2012)

Sin embargo, esta tecnología no empieza a tener popularidad hasta 2006 cuando Eric Schmidt, CEO de Google la vuelve a introducir, debido a las mejoras en ancho de banda y la aparición de grandes infraestructuras de servidores alojados en diferentes partes del mundo que hicieron la tecnología viable (Schmidt, 2006). De modo que en los últimos años se ha podido observar un aumento de estos servicios, como Salesforce, posteriormente Amazon Web Services, siendo este gran parte del negocio de Amazon y sumándose a esta tecnología Google con aplicaciones como Google Docs o Dropbox.

Con esta pincelada de historia, se puede pasar a su definición. No existe una única definición, pero basándose en la que ofrece el “National Institute of Standards and Technology” (NIST) de “U.S. Department of Commerce” se tiene: (la definición ha sido traducida al castellano)

“Cloud computing se trata de un modelo que permite acceso ubicuo y bajo demanda a la red con un conjunto de recursos informáticos configurables (como puede ser redes, servidores, almacenamiento, aplicaciones y servicios) que pueden ser suministrados y publicados de una manera rápida y con un mínimo esfuerzo de gestión o interacción del proveedor de servicios.” (National Institute of Standards and Technology, 2011)

Gracias a este modelo, usuarios sin conocimientos tecnológicos en el desarrollo y establecimiento de un servidor pueden acceder a los servicios de la nube que ofrece una compañía. Entre las ventajas que ofrece esta tecnología se encuentran:

(1) Ahorro de costes al no tener que invertir en infraestructura. (2) Aprovisionamiento bajo demanda, ya que el coste inicial es muy bajo al no tener que establecer la infraestructura y solo pagar en función del servicio requerido. (3) Multitenencia, ofreciendo los proveedores servicios parecidos a sus clientes, haciendo que comparten la misma arquitectura. (4) Flexibilidad, ajustándose a lo que el cliente desee. (5) Calidad y fiabilidad, al ser un servicio dedicado específicamente a computación en la nube, el servicio es especializado. (6) Seguridad en la nube, al igual que el punto anterior, al estar especializados en servicios en la nube tendrán mayores conocimientos que una empresa no especializada. (7) Menos mantenimiento, más construcción, lo que permite al cliente centrarse en su producto y no en mantener la plataforma.

Como se ha podido observar, se tienen dos tecnologías que están experimentando un crecimiento sin precedentes y de forma independiente. Ambas tienen sus fortalezas en su campo, pero también sus debilidades. La tecnología IoT tiene capacidades de almacenamiento, accesibilidad y computación limitadas, sin embargo, puede establecerse en cualquier lugar del mundo. Por el contrario, la tecnología “Cloud”, permite accesibilidad ubicua con recursos ilimitados, pero no se puede distribuir en todos los lugares. Estas dos tecnologías son complementarias, es de ahí de donde surge el concepto “CloudIoT”.

“CloudIoT” se beneficia de ambas tecnologías para crear un nuevo paradigma con recursos ilimitados. Permite una integración reduciendo los costes ya que unifica la

heterogeneidad de protocolos de IoT facilitando el despliegue de plataformas. Con esto se logra que el almacenamiento, la computación, la comunicación y el alcance de ambas tecnologías sea ilimitado. (IEEE, Fox, Kamburugamuve, & Hartman, 2012)

Esta fusión de tecnologías tiene infinitud de aplicaciones, algunas de ellas ya descritas en el apartado de “Internet de las Cosas”, pero desarrollando algunas de las más importantes se tiene:

- Mejora del sistema sanitario:

La posibilidad de una sanidad más personalizada, en el que la asistencia en el hogar gana protagonismo al no tener que ir tanto al médico. Esto supone un ahorro de costes y reducción de tiempos en sanidad, así como una mejora en el servicio.

- Ciudades inteligentes o “Smart cities”

Mejora de las ciudades a nivel ambiental, social y político ya que los ciudadanos dispondrán de un sistema de información en tiempo real, lo que aumenta la conciencia colectiva en ámbitos como la sostenibilidad.

- Domótica

La monitorización de parámetros en el hogar es uno de los campos en los que el IoT tiene mayor desarrollo. En gran parte por la propia voluntad de la gente a desarrollar dispositivos para automatizar tareas rutinarias en casa y poder mejorar la eficiencia energética para reducir el consumo y por ende el gasto económico en calefacción o luz. Además, gracias a la implementación de plataformas IoT en el “Cloud” se pueden monitorizar y visualizar los datos de forma que el usuario puede conocer en tiempo real la información y tener un consumo responsable.

- Movilidad inteligente

Gracias a la obtención de datos en tiempo real, junto con plataformas para su procesado, se puede aumentar la seguridad vial, reducir los atascos, mejorar el servicio de estacionamiento y mantenimiento de carreteras, lo que se traduce en mejoras ambientales y coste humanitario.

También cabe destacar la conducción autónoma gracias a la red de sensores interconectados que son capaces de comunicarse entre sí y reaccionar ante peligros. Hoy en día ya existen empresas como Tesla, que implementan conducción semi autónoma en una gran cantidad de carreteras, posicionándose como uno de los coches eléctricos con mayor seguridad en carretera.

- Energía inteligente o “Smart grid”

Gracias a la monitorización de los parámetros energéticos, las distribuidoras pueden ofrecer un servicio eficiente de distribución al conocer el consumo. Esto genera una red más segura que se traduce en una mayor calidad para el usuario y menores perdidas energéticas.

- Logística

Mejora de la eficiencia al tener un seguimiento del producto en todas las etapas del producto, pudiendo monitorizar los tiempos en los que los productos pasan más tiempo para optimizar la cadena de suministro. Además, se pueden controlar variables ambientales como temperatura y humedad muy importante en algunos sectores como el alimentario. También un aumento de seguridad en el usuario final ya que se pueden detectar defectos y productos en mal estado de forma temprana.

- Mejora ambiental

El despliegue de sensores por grandes áreas que permitan la obtención de datos ambientales del agua, concentración de gases, vibraciones, posibles incendios en áreas rurales, hace que la acción temprana sea eficiente y se puedan desarrollar políticas sostenibles al disponer de la información en tiempo real en cualquier parte del mundo.

- Video vigilancia

Mejora de la seguridad y reducción de tiempos al poder procesar los videos y descartar aquellos tramos que sean prescindibles. Además, gracias al reconocimiento de imágenes y objetos se pueden crear algoritmos que permitan mejorar la eficiencia en ciertas tareas.

1.1.3 Motivación

Después de conocer y estudiar los dos conceptos anteriores, se puede apreciar su gran importancia en el panorama tecnológico actual. Así pues, en este Trabajo Fin de Máster se busca desarrollar una plataforma IoT que permita obtener ciertos parámetros a estudiar y que estos se puedan almacenar, visualizar y procesar en cualquier otro lugar. En el caso a estudiar se busca realizar la monitorización de parámetros relevantes para la eficiencia energética en un bloque de viviendas a reformar. De esta forma, se puede conocer el grado de eficiencia obtenido en la reforma y si cumple con los estándares.

Además de mejorar la eficiencia energética por parte de la empresa, aspecto que debería ser fundamental para cualquier compañía, se logra crear en el usuario una concienciación acerca de la eficiencia energética al poder conocer en tiempo real los consumos y variables ambientales de su hogar. Esto hace que se busque un consumo eficiente, aunque solo sea por reducir las facturas.

Hoy en día existen varios servicios IoT en el mercado, como se analiza en el estado del arte, que permiten establecer una plataforma IoT de forma sencilla reduciendo el “Time to market”, es decir el tiempo que se necesita desde que el producto se crea, hasta que se pone a la venta. Se podría haber escogido uno de estos servicios y haber simplificado el proyecto, pero se buscaba poder obtener una plataforma flexible y personalizada para que fuera a medida del usuario final, sin tener restricciones ni gastos variables según la utilización. Esto hace que el proyecto sea más complicado a cambio de ganar más flexibilidad y personalización.

Por lo tanto, se busca desarrollar una WSN que permita obtener parámetros relacionados con la eficiencia energética en una vivienda y transmitirlos a un servidor para su monitorización. De este modo, se estarían aplicando todos los conceptos que se han explicado en la introducción.

1.2 PRINCIPALES DESAFIOS

Debido a la irrupción de estas nuevas tecnologías en el día a día de las compañías y personas, aparecen nuevos desafíos que el conjunto de la sociedad y empresas deben afrontar. Entre los desafíos no solo aparecen desafíos técnicos, sino que es un problema multidisciplinar, involucrando ámbitos y sectores energéticos, medio ambientales, éticos y económicos. En este apartado se realiza un análisis general de ciertos aspectos que se han considerado relevantes para que el lector, en caso de que no la tenga, tome conciencia de que no solo se limita a un aspecto tecnológico.

Este nuevo paradigma en el que conviven miles de dispositivos conectados, transmitiendo centenares de mensajes entre ellos necesita de nuevas tecnologías de conectividad y de protocolos de comunicación. En los últimos años se han desarrollado protocolos en los que se buscaba optimizar la cantidad de información a enviar, sin embargo, con los dispositivos IoT todo esto cambia.

En los dispositivos IoT, se busca optimizar el consumo de batería, el número de dispositivos conectados con diferente identificación o la distancia de transferencia. Además, el formato de los mensajes cambia. Son mensajes de tamaño pequeño que se realizan cada cierto tiempo, en vez de ser mensajes de gran capacidad. Es por ello por lo que se necesitan de protocolos que minimicen el consumo de batería y se adecuen al formato del mensaje. Algunos de estos nuevos protocolos se muestran en el capítulo 2.

En este sentido existe un problema actualmente. Las compañías y desarrolladores se centran en el desarrollo de protocolos en ciertas aplicaciones o tecnologías, sin existir una predisposición a crear estándares comunes. Puede que hoy en día no sea un problema muy grande, pero si se buscan redes gigantes de dispositivos interconectados de bajo coste, será necesario desarrollar estándares universales de código abierto o el desarrollo de la tecnología IoT puede verse ralentizado.

Debido al número de sensores conectados se generan enormes cantidades de datos que tendrán que procesarse y almacenarse. Esto supone una capacidad computacional descomunal, y por tanto mayor consumo de recursos, tanto minerales como energéticos. Para lidiar con esto, se tendrán que desarrollar algoritmos eficientes en el procesamiento de datos, así como técnicas de mejora energética (Gubbi, Buyya, Marusic, & Palaniswami, 2012).

Para hacer mayor énfasis en ello, según estudios realizados en 2013, se calcula que el 90% de los datos producidos hasta entonces habían sido creados en torno a los dos últimos años, produciendo cada día aproximadamente entre 2 y 3 millones de Yottabyte ($2-3 \times 10^{30}$). (Jacobson, 2013)

Estas redes de sensores, en muchos casos, se encontrarán en ambientes hostiles con vibraciones, suciedad, variaciones ambientales, lluvia, etc. Esto es muy diferente a lo que hasta ahora se tenía en los grandes centros de datos, en los que las variables están controladas. Es por ello por lo que los ingenieros tendrán que desarrollar nuevos modelos y diseños para poder hacer frente a las adversidades. (IERC, 2010)

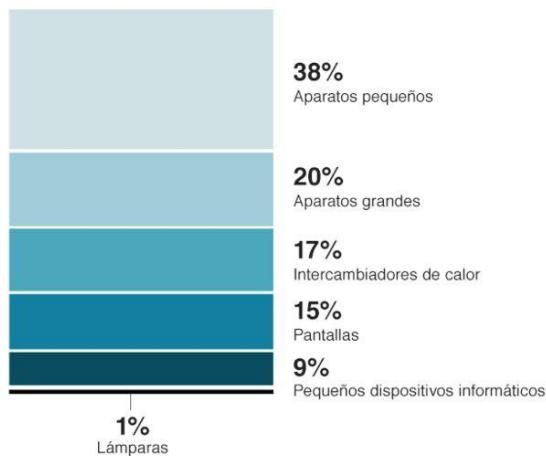
En el desarrollo de este tipo de tecnologías no es posible conocer el número de aplicaciones finales en las que podrá beneficiar, como en su día no fue posible concebir el número de aplicaciones que tendría un ordenador o un teléfono móvil. Por lo tanto, ante la incertidumbre de qué deparará el futuro a largo plazo, la mejor posición es la de asumir que cualquier cosa puede suceder, intentando prevenir las consecuencias y minimizando las perdidas.

El aumento de dispositivos electrónicos a las cifras que se pueden observar en la Ilustración 6 supondrá un incremento del consumo energético como en su día lo supuso Internet. Por lo tanto, será necesario un desarrollo paralelo de las energías renovables, una mejora en las baterías y una concienciación real y global que permita tomar medidas políticas para tener un desarrollo sostenible tanto medioambiental como social.

Si se centra la vista en la generación de residuos electrónicos, observando los números de basura generada en la actualidad, es lógico pensar que el problema al desarrollar millones de dispositivos se va a agravar. Hoy en día menos del 20% de los dispositivos desecharados se recicla en Estados Unidos, de los cuales el mayor porcentaje proviene de aparatos pequeños como se muestra en Ilustración 8 (BBC News, 2019). En consecuencia, si no se toma un camino en el que el diseño del dispositivo sea

sostenible y busque reducir los impactos ambientales, el problema aumentará de forma exponencial.

Composición de los residuos electrónicos



Fuente: Global E-waste Monitor, 2017

BBC

Ilustración 8 - Porcentage of electronic waste.³

La generación de basura electrónica supone otro problema social, ya que gran parte de los residuos generados son depositados en países en vías de desarrollo o subdesarrollados. Además, al problema social provocado por la basura, se le suma el aumento de la desigualdad entre países desarrollados y en vías de desarrollo. Esto se debe a que en los últimos no existe una infraestructura que permita conexiones rápidas y en muchos casos son inexistentes para el uso de Internet. En ese aspecto existen soluciones que se están llevando a cabo como establecer Internet mediante satélites, en el que una red global de satélites pueda ofrecer Internet en todas las partes del mundo a un coste reducido. (Boyle, 2019)

En cuanto a cloud computing, los principales retos que presentan son de seguridad, teniendo tres vertientes:

La primera, la seguridad de los datos, ya que no se puede saber con certeza donde están alojados ni que tratamiento está haciendo la empresa proveedora con ellos. En un futuro no muy lejano, las compañías recolectarán datos del día a día de las personas, ya que los datos son oro. Con esta información se podrá crear patrones de comportamiento, que en muchos casos contradice la propia moral. Por ello es necesario que se tomen medidas legales acerca de cuál es el trato que se realiza con los datos.

La segunda, la seguridad ante caída del servicio, en el que no se pueda disponer de la información ante la caída de los servidores, o que incluso se pierda debido a un fallo de seguridad. Las compañías proveedoras de servicios muestran que es un servicio seguro. Sin embargo, el miedo en la sociedad ante este tipo de sucesos es lo que no permite que este tipo de servicio sea completamente extendido entre usuarios y empresas.

La tercera, la seguridad de los usuarios ante ataques de terceros. Son numerosos los casos registrados de hackers que logran acceder a los dispositivos de los usuarios y robarle los datos personales o incluso mediante llamadas masivas a un servidor, hacer que sus servicios caigan (González, 2016). Frente a este panorama, las compañías deberán desarrollar dispositivos más seguros frente a reducción de costes, así como los

³ (BBC News, 2019)

usuarios tendrán que tomar conciencia de los peligros que entraña este tipo de tecnología.

1.3 OBJETIVOS DEL PROYECTO

En este apartado se pretende presentar en una primera aproximación cuales son los objetivos por cumplir con el proyecto. Posteriormente, en el capítulo 3 se detallan objetivos más específicos, el alcance y los requisitos del proyecto.

El objetivo principal del Trabajo Fin de Máster es diseñar una plataforma IoT que permita monitorizar la mejora de la eficiencia energética en un edificio residencial. Para ello, es necesario desarrollar una serie de sensores que permitan realizar mediciones relacionadas con parámetros energéticos. Estos sensores deben ser capaces de transmitir los datos a una placa central, llamada hub central que a su vez se comunica a través de Internet con un servidor donde se alojan los datos. El servidor debe permitir almacenar, visualizar y procesar los datos transmitidos por los sensores.

Se trata de un proyecto amplio que abarca desde el diseño de sistemas electrónicos a la creación de algoritmos para la nube. Por lo tanto, a continuación se presenta enumerados una serie de objetivos más específicos:

- Diseño del esquema electrónico, del PCB y ensamblado de los sensores y hub central.
- Desarrollo de los firmwares que controlan los sensores y hub central.
- Desarrollo de la comunicación entre la red de sensores y el servidor.
- Desarrollo de los algoritmos que permitan almacenar, visualizar y procesar los datos de los sensores.
- Desarrollo y evaluación de una serie de experimentos para comprobar la fiabilidad y robustez del sistema

1.4 ORGANIZACIÓN DE LA MEMORIA

El desarrollo de la documentación se va a dividir en 11 capítulos. El capítulo 1 – Introducción, como su nombre indica, consiste en una introducción a las diferentes tecnologías que durante los años han aparecido en el campo del “Internet de las Cosas” junto con sus aplicaciones y desafíos. Además, se puntuiza en que consiste el proyecto a desarrollar.

En el capítulo 2 – Estado del arte, se estudia el estado actual de las tecnologías y protocolos relacionados con la aplicación del proyecto. El capítulo 3 – Objetivos, se detallan los objetivos, el alcance y los requisitos a conseguir.

En el capítulo 4 – Métodos y equipo, Se divide en tres partes. Primero se explica cómo se organiza el proyecto, para tener una visión general de la estructura de este y de los componentes básicos. Segundo, se desarrolla de manera general los conocimientos básicos para entender el proyecto para después explicar más detalladamente cada una de las tecnologías utilizadas. Tercero, se detallan todos los componentes del proyecto dividiéndolo en hardware y software.

El capítulo 5 – Desarrollo del proyecto, versa sobre todo el trabajo que se ha desarrollado para conseguir que todos los componentes del proyecto funcionen correctamente, desde el diseño de las placas de circuito impreso, en inglés “Printed Circuit Board” (PCB), hasta los algoritmos para la visualización y procesado de los datos.

El capítulo 6 – Experimentos, se detallan la serie de pruebas que se han llevado a cabo para validar el sistema y como se han ejecutado. En el capítulo 7 - Resultados y análisis, se concluyen cuáles han sido los resultados del capítulo anterior. En el capítulo 8 – Conclusiones, se profundiza sobre los resultados globales obtenidos, reflexionando sobre las características que se han logrado en la plataforma. Además, se evalúa que aspectos sociales, ambientales, legales o éticos están relacionados con el proyecto. En el capítulo 9 – Líneas futuras, se detallan posibles mejoras de la plataforma y siguientes pasos a tomar.

Para finalizar, en el capítulo 10 – Planificación temporal y Presupuesto, se detallan los aspectos de carácter organizativo, tanto temporal como de objetivos del proyecto, junto con un análisis económico del mismo. Como últimos capítulos del proyecto se tienen los anexos, donde se amplía la información del proyecto que, a pesar de ser muy relevante en el mismo, harían una lectura pesada de la documentación.

2. ESTADO DEL ARTE

En este capítulo se va a realizar un estudio de las diferentes tecnologías que se pueden encontrar hoy en día desde el punto de vista de conectividad, así como diferente hardware que hay disponible para desarrollar una plataforma IoT y los servicios que se pueden encontrar de los diferentes proveedores.

Al ser un campo que está en pleno auge y desarrollo, las tecnologías que se exponen en el documento puede que hayan variado, algunas hayan desaparecido o que existan nuevas que las dejen obsoletas. En cualquier caso, se expondrán las más relevantes relacionadas con el proyecto.

2.1 TECNOLOGÍAS DE CONECTIVIDAD

Los dispositivos IoT necesitan de redes de comunicación para la transferencia de información, y estas fundamentalmente son inalámbricas debido a su versatilidad. Existen numerosas tecnologías de comunicación con diferentes características, entre las cuales cabe destacar para este tipo de proyectos: velocidad de transmisión, ancho de banda, alcance, consumo de energía, seguridad, duración de batería, precio de infraestructura o precio de licencias para operar. En función de la aplicación a desarrollar se tendrá en cuenta unas características frente a otras. A continuación, se presentan una serie de tecnologías populares en el ámbito de estudio del proyecto:

2.1.1 6LoWPAN

El nombre deriva de “IPv6 over Low-Power Wireless Personal Area Networks”. Se trata de una capa de adaptación que permite enviar paquetes de IPv6 según se define en IEEE 802.15.4. Se trata de un estándar abierto definido en RFC 6282 por el IETF. En sus comienzos se desarrolló para trabajar con redes de 2.4 GHz de bajo consumo, sin embargo, se adaptó para otras redes como Sub-1 GHz de baja potencia, RF, Bluetooth Low Energy (BLE), control de líneas de potencia o “Power Line Control” (PLC) y Wi-Fi.

Gracias a esta tecnología se simplifica el proceso de adaptación entre la pila de IPv6 y los enlaces de radio IEEE 802.15.4 como se muestra en la Ilustración 9, en la que se muestra además un ejemplo de Wi-Fi y el modelo “Open Systems Interconnect” (OSI), que es el modelo más común de los sistemas de comunicación, que se muestra simplificado. (Olsson & Texas Instruments, 2014)

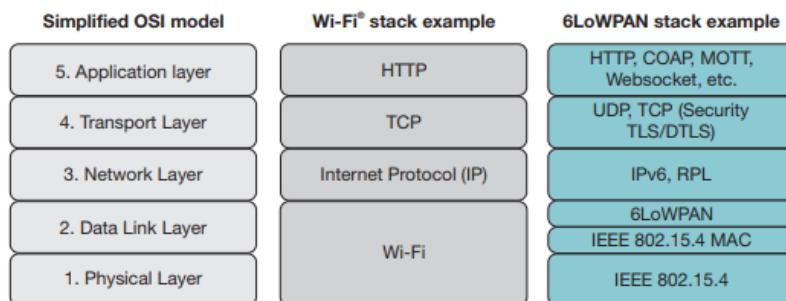


Ilustración 9 - The OSI model, a Wi-Fi® stack example and the 6LoWPAN stack.⁴

⁴ (Olsson & Texas Instruments, 2014)

2.1.2 Bluetooth

Se trata de una tecnología de corto alcance y bajo consumo que mayor popularidad posee. Permite conectar dispositivos inalámbricos mediante radio frecuencias con consumos de batería pequeños, lo que lo hace ideal para dispositivos que funcionan con batería. Las últimas versiones como Bluetooth 5 permiten mayores alcances y mayor velocidad con consumos de energía menores. Trabaja con frecuencias de 2.4 GHz con niveles limitados de seguridad en la transmisión de información. (Bluetooth & Woolley, 2019)

2.1.3 Ethernet

Es la tecnología de comunicación más popular para conectar redes de área local (LAN). Debido a que no es inalámbrica, es la que posee mayor resistencia ante interferencias o interrupciones, a la vez que posee mayor velocidad que las inalámbricas. El mayor problema que presenta es la necesidad de medio de transmisión físico, como es un cable, para la transmisión de información.

2.1.4 LoRa

Su nombre proviene de “Long Range” o largo alcance, ya que permite la comunicación de sistemas a cientos de kilómetros. Esto lo consigue mediante una modulación especial en la capa física que le permite mantener consumos bajos y largos alcances. Esto hace que la tecnología sea ideal para conectar cientos de dispositivos en Smart cities. El problema es que la velocidad de transmisión de información y el ancho de banda es bajo en comparación a las tecnologías estudiadas. (Semtech, 2019)

2.1.5 NFC

Su nombre es “Near Field Communication”, y como este indica se trata de una conexión inalámbrica de corto alcance. Al ser de corto alcance se convierte en una tecnología muy útil para dispositivos móviles y tarjetas a la hora de efectuar transacciones o intercambios de información de forma segura. Se basan en el mecanismo de inducción de campo magnético al acercarse un dispositivo al otro. (NFC Forum, 2019)

2.1.6 RFID

En esta tecnología aparecen varios agentes que permiten la lectura de la información. Por un lado, está la etiqueta RFID o transpondedor. Esta contiene una identidad digital única, la información a enviar, que se almacena en la memoria, además de una antena y microchip. Por otro lado, está el lector RFID, o transceptor que envía señales de radio frecuencia que interaccionan con la etiqueta de forma que esta envíe su identificación.

Existen diferentes tipos de etiqueta RFID. Estas pueden ser activas o semi pasivas, las cuales necesitan de una batería o pila para funcionar, alcanzando mayores rangos. El otro tipo son pasivas, las cuales se activan al recibir radiofrecuencias e inducir una corriente. (Rouse, IoT Agenda, 2019)

2.1.7 Sigfox

Se trata de una tecnología que transmite en “Ultra Narrow Band” (UNB) o banda de radio frecuencias muy estrecha, en la que no es necesario disponer de licencias. Se convierte en una tecnología interesante cuando se busca una red de cientos de dispositivos, con velocidades de transferencia bajas y alcances intermedios. Así pues, es ideal para Smart cities al igual que LoRa.

Permite tener cientos de dispositivos que transmiten datos sin que estos mantengan la conexión a red. Esto hace que la duración de la batería aumente considerablemente. La red se gestiona desde la Nube, lo que permite una reducción de costes y consumos.

Todas estas características hacen que esta tecnología haya sido implantada en numerosas ciudades europeas ya que permite comunicaciones a decenas de kilómetros en ciudades, con bajos consumos y de forma eficiente. (Sigfox, 2019)

2.1.8 Wi-Fi

Es la tecnología de comunicación inalámbrica más extendida, sobre todo en el entorno doméstico. Esto se debe a que, al igual que con Ethernet, existe una red establecida en todo el mundo que permite la comunicación mediante WiFi.

Su gran popularidad se debe a su elevada tasa de transferencia de datos, que se ve incrementada con los nuevos estándares que se desarrollan. Sin embargo, en el mundo IoT presenta grandes desventajas frente a otras tecnologías. Por lo general en IoT no es fundamental grandes velocidades de transferencia de información, priorizando el consumo de batería y el alcance. Es aquí donde esta tecnología es débil, ya que los consumos son elevados y el alcance es reducido en comparación a otras tecnologías actuales. (Liao, Saleh, & Haque, 2010)

2.1.9 ZigBee

Se trata de una nueva tecnología que se usa en entornos industriales, pero también está ganando mucha relevancia en el ámbito de la domótica. Se trata de una tecnología de alcance medio, con bajas tasas de transferencia de datos. Esto hace que los consumos de energía sean bajos, más que por bluetooth, convirtiéndose en una tecnología muy interesante para el hogar, en el que varios nodos se conectan. Además, se trata de una tecnología de bajo coste con alta duración de batería. (Elahi & Gshwender, 2009)

2.2 PROTOCOLOS DE COMUNICACIÓN

Una vez se conoce la tecnología de comunicación a utilizar, es necesario conocer qué protocolo de comunicación es el adecuado. Esta elección se basa en los requisitos del proyecto, así como en la tecnología escogida anteriormente. El protocolo no es más que el conjunto de reglas que se establecen en la comunicación de forma que los actores se entiendan entre ellos sin errores. Se van a identificar los protocolos de comunicación más relevantes para la tecnología IoT y para el proyecto que se está desarrollando:

2.2.1 AMQP

“Advance Message Queuing Protocol” se trata de un protocolo de mensajería estructurado en capas, que permite la transmisión de mensajes entre dos actores de forma segura. Se trata de mensajes de pequeño tamaño que van dirigidos a un tema, un filtro o directamente a la dirección de destino. Dispone de numerosas características de mensajería y tiene cierta similitud al protocolo MQTT que se verá a continuación. (International Organization for Standardization, 2014)

2.2.2 CoAP

“Constrained Application Protocol” está especializado para la transferencia de información entre nodos de dispositivos IoT. Como HTTP, hace uso del protocolo REST, en el que mediante peticiones GET, POST, PUT, DELETE a una URL se puede acceder a los recursos que deben estar alojados en un servidor. Como está optimizado para IoT en comunicaciones “Machine to Machine” (M2M), su tráfico es más reducido que HTTP, con encabezados menores y ligeros, empleando “User Datagram Protocol” (UDP) en vez de TCP/IP. Además, funciona en microprocesadores con pocos recursos sin prescindir de la importancia de la seguridad. (CoAP technology, 2019)

2.2.3 MQTT

“Message Queuing Telemetry Transport” está destinado a la comunicación de diferentes nodos que poseen recursos limitados ya que este protocolo es muy ligero. Hace uso de un sistema de suscripción/publicación en el que un bróker gestiona las comunicaciones. Los clientes crean una comunicación TCP/IP y publican la información en un tema determinado al que otros subscriptores se pueden subscribir. Dispone de dos puertos estándar para la comunicación, 1883 si es no cifrada, o 8883 si es cifrada mediante SSL/TLS. (Rouse, IoT Agenda, 2018)

2.2.4 REST (HTTP)

“REpresentational State Transfer” se trata de un tipo de arquitectura basada en el protocolo HTTP para el desarrollo de “Webservices” o servicios web. Su implementación es sencilla debido a que se basa en HTTP que es un protocolo existente. Se basa en peticiones y respuestas por parte del servidor mediante peticiones del tipo GET, POST, PUT y DELETE. Consumo mayor ancho de banda que los protocolos vistos hasta ahora, ya que lo que se transmite es texto plano, pero es más fácil de entender. (Rouse, REST (REpresentational State Transfer), 2017)

2.2.5 TCP/IP

Este protocolo se compone de un conjunto de protocolos, entre los que destacan los que componen su nombre, el TCP y el IP. Si se estudia según las capas del modelo de referencia OSI que aparece en Ilustración 9, permite por un lado una comunicación segura a nivel de la capa de transporte. Por otro lado, el protocolo IP permite la identificación en la capa de red entre las máquinas. (O'Reilly, 2010)

2.3 PLATAFORMAS HARDWARE

Para que las tecnologías de comunicación y protocolos presentados funcionen es necesario hacer uso de dispositivos hardware que permitan generar los mensajes. Por ello en este apartado se van a estudiar soluciones, algunas de las cuales son utilizadas en el proyecto, que debido a su popularidad y características son relevantes. En función del propósito del proyecto se buscará unas características determinadas, pero en general para proyectos IoT, se buscan soluciones que minimicen el consumo de energía, permitan una comunicación estable y fiable, con precios moderados o bajos y que permitan la escalabilidad del proyecto.

2.3.1 Raspberry Pi

Se trata de una plataforma hardware versátil, con una capacidad de computo moderada, ideal para el desarrollo de proyectos IoT. Permite un desarrollo rápido de la plataforma adaptándose correctamente a las necesidades y aporta una gran escalabilidad. Esto se debe a la gran documentación existente y a la comunidad que desarrolla soluciones IoT. (Raspberry Pi ORG, 2019)

Existen diferentes versiones de Raspberry, desde su versión inicial con menor memoria y sin conectividad, la Raspberry Pi 1 modelo A+, hasta su última versión, la Raspberry Pi 4 que alcanza niveles comparables a los de un ordenador de baja gama. También existen versiones en miniatura de la Raspberry como la versión ZERO, ideales para proyectos en los que el tamaño es un factor limitante.

Gracias al uso de distribuciones open source GNU/Linux, como sistema operativo hace que su precio sea menor y que desarrolladores diseñen sus propias aplicaciones sin tener que pagar licencias. Sin embargo, también puede trabajar con UNIX o Windows, como Microsoft Windows IoT Core, distribución de Windows optimizada para dispositivos IoT.

2.3.2 Arduino

Al igual que la plataforma hardware anterior, Arduino es una plataforma open source muy popular y extendida entre desarrolladores debido a su sencillez y su carácter educativo. Existen varias versiones oficiales basadas por lo general en un microprocesador Atmel AVR. Se trata de una placa sencilla y de iniciación con múltiples entradas y salidas analógicas que permiten la comunicación e interacción con el entorno. Dispone de placas de expansión que le permiten extender sus funcionalidades. (Arduino, 2019)

Dispone de un software propio, Arduino IDE, para el desarrollo de proyectos que puede ser ejecutado en los sistemas operativos más extendidos del mercado. Así pues, su carácter multiplataforma, versatilidad y su sencillez la convierten en uno de los hardware preferidos a la hora de desarrollar nuevos proyectos. Dispone de menos capacidad de procesamiento que Raspberry Pi, pero se trata de una placa más enfocada al desarrollo de hardware que la anterior.

Además, al ser una plataforma open source con una gran comunidad de desarrolladores, a partir del concepto que supuso Arduino en el mercado, se han desarrollado nuevas versiones con otros microprocesadores. Esto hace que existan infinidad de plataformas hardware especializadas en un campo en concreto, como se verá a continuación.

2.3.3 ESP 8266/32

Se trata de dos microprocesadores de bajo coste y bajo consumo que integran conexión WiFi con el protocolo TCP/IP, lo que permite la conexión a internet del dispositivo. El ESP 32 es el sucesor de ESP 8266, incorporando además conexión Bluetooth, mayor capacidad de memoria y procesamiento, un segundo procesador y modos de menor consumo. Además, cuenta con otros tipos de comunicaciones como I²C, Serial, SPI, así como entradas analógicas y digitales como un Arduino. (Espressif, 2019)

Todo esto hace que este microprocesador sea ideal para el desarrollo de plataformas hardware ya que implementa todo lo necesario para interaccionar con el entorno, transmitir la información y mantener consumos de energía bajos. A partir de estos microprocesadores básicos, las empresas han desarrollado sus propias “development boards” o placas de desarrollo, creando versiones más enfocadas a ciertos ámbitos del IoT.

Otra característica muy interesante de esta plataforma es que no posee un único entorno de programación. Esto hace que se puedan crear programas en diferentes lenguajes y con distintos programas. Entre ellos destacan: Arduino IDE por ser muy popular entre los desarrolladores debido a Arduino, y MicroPython, ya que el lenguaje de programación Python ha sufrido un gran auge en los últimos años debido a su sencillez y a la gran cantidad de módulos para el tratamiento de datos.

2.3.4 Texas Instruments SimpleLink MCU SDK

A pesar de que la empresa Texas Instruments es uno de los líderes en el sector de semiconductores, sus placas de desarrollo o “Software Development Kit” son las menos conocidas en comparación a las que se han mostrado. Sin embargo, estas presentan, en algunos casos, características superiores frente a las anteriores. (Texas Instruments, Texas Instruments SimpeLink Solutions, 2019)

Las placas de desarrollo de Texas Instruments disponen de todas las características que se han visto en los dos últimos apartados. Pero además tienen compatibilidad con muchos de los protocolos y tecnologías que se han expuesto a lo largo del estado del arte, lo que las convierte en unas placas muy versátiles a la hora de desarrollar redes inalámbricas de sensores y actuadores. Además de soportar la tecnología WiFi, lo que permite crear toda una plataforma IoT. También tiene optimizada la parte de consumos de energía, lo que permite desarrollar aplicaciones basadas en baterías.

Cabe destacar la cantidad de documentación y ejemplos que la propia compañía aporta para comenzar a desarrollar aplicaciones con su plataforma y la ayuda personalizada que ofrecen en caso de que surja algún problema.

Su mayor problema se encuentra en que al ser una plataforma profesional, hace que los precios se encarezcan frente a las soluciones anteriores. Además, al disponer de más características que las anteriores, hace que sean más complicadas de entender a la hora de desarrollar nuevas aplicaciones, lo que supone un aumento del tiempo de desarrollo. Sin embargo, una vez se conoce la plataforma se consiguen aplicaciones con mayores capacidades y optimizadas.

2.4 SERVICIOS DE “CLOUD COMPUTING”

Después de haber estudiado las plataformas hardware, sus protocolos y tecnologías de comunicación más relevantes, se tiene que hacer un análisis de las plataformas software o servicios de Cloud computing. Estas serán las que reciban la información generada por el hardware u otros medios.

Como ya se ha mencionado anteriormente, se trata de un mercado con mucha incertidumbre ya que está en pleno auge y cambio. Es un mercado de gran valor económico al que muchas empresas se quieren sumar debido a la utilidad y valor de los datos hoy en día. Es por esto por lo que las plataformas que se muestran a continuación, o alguna de sus características, puede que hayan cambiado, sirviendo este apartado al lector como punto de partida para la búsqueda de la plataforma adecuada.

A la hora de escoger la plataforma Cloud se tiene que atender a las características que presenta, entre las que cabe destacar: Alcance de servicios, precio de la plataforma, capacidad de gestión de dispositivos, protocolos y servicios utilizados para integrar aplicaciones con el almacenamiento y el procesamiento de datos, seguridad, protocolos de comunicación soportados por la plataforma para la transmisión de datos, servicios de visualización de los datos disponibles, servicios de procesamiento de los datos en tiempo real, y por último la documentación aportada por la plataforma para su correcto funcionamiento y como de robusta es la compañía. En este apartado no se va a realizar un análisis de las características de cada plataforma, simplemente se comentarán las más relevantes de estas.

2.4.1 Thingspeak

Se trata de una plataforma de código libre que pertenece a Mathworks, por lo que implementa Matlab como motor de procesado de datos. Esto la convierte en una plataforma interesante ya que Matlab es uno de los entornos de programación más populares que permite el computo numérico y dispone de multitud de paquetes que amplían sus capacidades. Dispone de diferentes tipos de licencias, entre ellas para estudiantes, que permite el almacenamiento, visualización y procesado de la información gratis, pero con limitaciones. (Mathworks, 2019)

Dispone de una API para la comunicación con los dispositivos, disponiendo de documentación para llevarla a cabo. En la documentación se tratan plataformas hardware ya mencionadas como Raspberry Pi o Arduino. La plataforma acepta protocolos como MQTT o REST, lo que permite acceder desde la mayoría de los dispositivos.

Además de utilizar Matlab como motor de computo, dispone de otras utilidades para actuar en caso de que alguna condición ocurra, enviando comandos, mensajes, avisando mediante Twitter y otras redes sociales. Debido a todas estas características, y su sencillez y facilidad de uso, hacen de Thingspeak una plataforma ideal para el comienzo en el mundo de IoT, aunque en proyectos grandes puede que se quede pequeña.

2.4.2 Altair SmartWorks (Carriots)

Se trata de una solución más enfocada al mundo profesional frente a la anterior. También es una plataforma de pago, aunque permite disponer de una cuenta gratis con ciertas limitaciones. Esta plataforma dispone de una API REST para la integración de

datos, que se realizan bajo el formato JSON o XML. Permite la visualización de los datos y un análisis de estos en tiempo real. (Altair SmartWorks, 2019)

Además de estas características parecidas al punto anterior, también ofrece un servicio de nube privada para alojar contenido. Esto aporta escalabilidad a la plataforma lo que la convierte en una plataforma interesante para dispositivos IoT en el mundo profesional.

2.4.3 Microsoft Azure

Plataforma desarrollada por Microsoft y alojada en sus servidores, lo que al ser uno de los gigantes tecnológicos, asegura que su nivel de servicio sea prácticamente del 100%. Esto confiere una gran seguridad a sus clientes ya que sus aplicaciones están en funcionamiento siempre. Al igual que las anteriores, permite el almacenamiento, así como el procesado de la información, pero de manera más profesional. Permite instalar cualquier gestor de contenido y framework en multitud de plataformas, ya sean móviles o web. Además, permite la implementación de Machine Learning y análisis de Big Data. (Microsoft Azure, 2019)

Además, esta plataforma dispone de servicios IaaS y PaaS, de los que se entrará más en detalle en el capítulo 4 del documento. Básicamente el cliente puede elegir si gestionar completamente la aplicación disponiendo de una máquina virtual Windows Server o Linux o que la plataforma se encargue de gestionar los recursos y desplegar la aplicación, mientras que el cliente solo se encarga de desarrollar la aplicación.

Todo esto hace de Microsoft Azure una plataforma más completa que las anteriores debido a su versatilidad, seguridad de servicio, escalabilidad de aplicaciones y flexibilidad. Esta plataforma junto con las siguientes, conforman las llamadas “Big Four” debido a su importancia en el mercado IoT. (Bocchio, 2014)

2.4.4 IBM Cloud

Como su nombre indica, es una plataforma desarrollada por la compañía IBM, enfocada en el mundo profesional. Como se veía en Microsoft Azure disponen de un servicio enfocado al cliente en el que este puede elegir el modelo de nube en función de sus necesidades. Esto se puede lograr gracias a la escalabilidad que aporta la plataforma entre los diferentes modelos.

IBM Cloud además de implementar los modelos IaaS y PaaS implementa el modelo SaaS. En este modelo el proveedor ofrece aplicaciones ya desarrolladas y optimizadas al departamento en el que el cliente quiere implementar el servicio Cloud. Gracias a esto, el cliente no se tiene que preocupar de nada, solo se limita a comprar una aplicación ya desarrollada por IBM Cloud y personalizarla a su gusto. (IBM Cloud, 2019)

2.4.5 Google Cloud Platform (GCP)

Plataforma Cloud computing desarrollada por Google. Su gran ventaja frente al resto de plataformas es la infraestructura con la que cuenta el gigante Google, tanto en redes de comunicación como en servidores por todo el mundo. Además, a esto hay que sumarle que la plataforma reúne los servicios que ofrece Google por separado. Estas características, junto con la posibilidad de elección del modelo de nube que más se adecue a las necesidades del cliente, la convierten en una de las plataformas más interesantes entre los desarrolladores. (Google Cloud, 2019)

Posee los tipos de modelos que se han visto en Microsoft Azure con Google Compute Engine (IaaS) y Google App Engine (PaaS), así como diversas herramientas de monitorización, diferentes tipos de bases de datos y herramientas de traducción de Google.

De esta forma, los desarrolladores no se tienen que preocupar por mantener una infraestructura, dedicándose al desarrollo de su aplicación, escogiendo y pagando por el modelo que se adapte a sus necesidades. En el caso de necesitar mayor capacidad, gracias a la escalabilidad que ofrece Google, no se tienen que preocupar por las compatibilidades.

2.4.6 Amazon Web Services (AWS)

Plataforma desarrollada por Amazon que alberga aplicaciones tan conocidas como Dropbox. Se trata del mayor proveedor de servicios de clouding. Amazon Web Services dispone de numerosos servicios y soporta diferentes lenguajes de programación. También ofrece máquinas virtuales en los que alojar el servicio deseado, así como servicios de almacenamiento de bases de datos SQL o no SQL. Además, permite el análisis de Big Data con Hadoop.

Permite conectar un gran número de dispositivos soportando grandes cantidades de mensajes. Puede funcionar como enrutador intercambiando los mensajes entre diferentes dispositivos del sistema, además de hacer un seguimiento en tiempo real de los mismos.

3. OBJETIVOS

3.1 OBJETIVO GENERAL

El alcance final del Trabajo Fin de Máster es la obtención de una plataforma IoT funcional que permita la obtención de ciertos parámetros relacionados con la eficiencia energética en el ámbito doméstico. Esto se lleva a cabo mediante el diseño y desarrollo de sensores y la red de comunicaciones. Además, esta red de sensores inalámbrica debe ser capaz de transmitir la información a un servidor donde será almacenada y estará disponible para el usuario final. Este podrá visualizar, analizar, extraer la información de la red, así como gestionar por completo toda la plataforma IoT.

Se trata de un proyecto que surge de la colaboración entre la universidad “Politecnico di Torino”, donde se desarrolla el proyecto, y la empresa de reformas ENEA. Esta última busca un sistema capaz de medir variables relacionadas con la eficiencia energética. De esta forma se busca tomar medidas antes de reformar un bloque de viviendas y comparar esos valores con los obtenidos después de realizar la reforma. De este modo, la compañía es capaz de estimar el grado de mejora de la eficiencia energética en el bloque. El objetivo es poder obtener todos esos datos de forma automática una vez se haya desplegado la red de sensores.

En el alcance del proyecto queda fuera el despliegue y la toma de medidas en las viviendas. Esto se debe a la ventana temporal que se dispone para desarrollar el proyecto y debido a que se trata de un proyecto en fase temprana de desarrollo, en el que se tiene que desarrollar toda la base. Esto supone tener que definir las tecnologías y protocolos que mejor se adaptan al proyecto, que sensores permiten obtener los parámetros adecuados, que servicios del servidor son necesarios para la recolección de información y desarrollar los algoritmos necesarios para que toda la plataforma funcione correctamente.

De esta forma, este proyecto sienta las bases creando una plataforma IoT completa y desarrollando una serie de sensores para demostrar que es funcional y escalable. A partir de esta documentación se podrá seguir desarrollando el resto de los sensores necesarios para obtener todos los parámetros relacionados con la eficiencia energética. En el capítulo de líneas futuras se explica con mayor detalle cuales son los siguientes pasos que tomar.

Como se puede observar, se trata de un proyecto complejo, ya que consiste en el desarrollo de una plataforma completa, desde la plataforma hardware con el desarrollo de sensores, a la plataforma software con el desarrollo de la Nube al completo. Por ello, es necesario precisar unos objetivos más específicos que muestren con mayor detalle las partes del proyecto.

3.2 OBJETIVOS ESPECÍFICOS

Una vez se conoce el objetivo y alcance global del proyecto, es preciso definir con mayor grado de detalle cada una de las fases que se deben realizar. A continuación, se enumera cada una de las fases, algunas con mayor detalle que otras. Su enumeración no corresponde a su desarrollo real en el tiempo. Para conocer la distribución real de las tareas en el tiempo es necesario ir al capítulo 10, apartado Planificación temporal.

A la hora de desarrollar los objetivos específicos, se dividen en dos grupos: Objetivos para el desarrollo de la plataforma hardware, es decir, los objetivos relacionados con los sensores y establecer la red inalámbrica de comunicaciones entre sensores. Y los objetivos para el desarrollo de la plataforma software, que son los relacionados con el desarrollo de los servicios del servidor.

3.2.1 Objetivos: Plataforma hardware

1. Búsqueda, simulación, diseño y fabricación de las diferentes placas de sensores para obtener los parámetros necesarios. Como ya se ha mencionado anteriormente, debido a la magnitud del proyecto y la limitación temporal, no es posible desarrollar todos los sensores necesarios. Por lo tanto, se han escogido aquellos que permiten obtener los parámetros más importantes. Estos son: temperatura, humedad, presión, calidad del aire y consumo eléctrico. Deben cumplir una serie de características:
 - Usar un MCU de Texas Instruments.
 - Usar una tecnología de comunicación que permita transmitir información con un alcance de al menos dos pisos de una vivienda común.
 - Uso de baterías recargables como fuente de energía, excepto para el sensor de consumo que se tomará de la misma red eléctrica.
 - Los sensores deben cumplir unos límites de error, precisión y rango en sus medidas. Los límites se verán en el apartado oportuno de cada sensor.
 - Diseño de la PCB lo más compacto posible de forma que el impacto visual sea el mínimo posible. Esto se logra mediante un buen diseño y el uso de componentes SMD
2. Desarrollo del firmware que permita la obtención de los valores de cada sensor.
 - Deben estar optimizados de forma que el consumo de batería sea el mínimo posible para alargar la vida de estas.
3. Implementación de las comunicaciones entre las placas de sensores y el HUB central.
4. Simulación, diseño y fabricación del HUB central.
 - Esta PCB debe usar la misma tecnología de comunicación que la red de sensores para extraer los datos al mismo tiempo que usa una tecnología de comunicación compatible con Internet para transmitir información al servidor.
 - No es necesario que se alimente mediante baterías.
 - Al igual que el punto 1, debe ser lo más compacta posible.

3. Objetivos

5. Desarrollo del firmware para gestionar las comunicaciones y datos de los diferentes sensores por parte del HUB central.
6. Implementación de la commutación entre la tecnología de comunicación con la red de sensores e Internet en el HUB central.
 - Como ya se ha explicado, el HUB central constará de dos MCU que se comunicaran entre sí mediante UART, por lo que no es necesario commutar. Sin embargo, existen MCU que poseen las dos tecnologías de comunicación y en ese caso, sería necesario commutar entre ellas.
7. Desarrollo y evaluación de experimentos que permitan analizar la fiabilidad y correcto funcionamiento de las placas de sensores y hub central.
 - Con estas pruebas se pretende conocer si las placas han sido diseñadas, ensambladas y programadas según los objetivos del proyecto. Se trata de unas pruebas básicas que permiten conocer si el sistema funciona en situaciones de funcionamiento normales.

3.2.2 Objetivos: Plataforma software

Ambas partes de la plataforma IoT deben ser flexibles, robustas y escalables. Sin embargo, la parte software de la plataforma debe cumplir con la arquitectura de servicios web o “web services”. Este tipo de arquitectura se explicará en el siguiente capítulo con mayor detalle. No se enumera como objetivo, pero se ha tenido como principio básico en el desarrollo de la plataforma software.

8. Implementación de la comunicación entre el HUB central con el servidor mediante uno de los protocolos analizados.
 - El mensaje que se transmite debe incluir toda la información relevante del sensor, no solamente la medida tomada.
9. Desarrollo de algoritmos que permitan adaptar los mensajes entrantes al servidor con los diferentes bloques o servicios del servidor.
 - El desarrollo de la plataforma debe cumplir el concepto de escalabilidad. Por lo tanto, se deben poder introducir nuevos bloques en el servidor sin necesidad de modificar los algoritmos existentes.
 - El método para adaptar los mensajes de una plataforma a otra no debe depender de aplicaciones ya desarrolladas por compañías, ya que se busca la máxima flexibilidad y personalización de la plataforma IoT.
10. Implementación de una base de datos en el servidor para almacenar los datos obtenidos por los sensores.
 - Esta debe estar optimizada para almacenar grandes cantidades de series temporales de datos. Además, debe permitir incluir etiquetas a los datos de forma que posteriormente se puedan distinguir e identificar y filtrar.

11. Implementación de un visualizador de los datos almacenados en la base de datos en el servidor.
 - Al menos debe ser capaz de visualizar los datos que se encuentran en la base de datos según su registro temporal y poder comparar valores según las etiquetas. Es deseable que posea características adicionales como: diferentes tipos de gráficos para mostrar los valores, permita personalizar las visualizaciones según las necesidades, o se puedan activar alarmas si se alcanza algún valor deseado.
12. Implementación de una base de datos para almacenar información de la configuración de los sensores.
 - Deberá almacenar todos los sensores que se conecten a la plataforma con información básica como su identificador, localización, última conexión, etc.
13. Desarrollo de algoritmos que permitan la manipulación de las bases de datos.
 - En el caso de la base de datos de las medidas de los sensores, debe poderse crear, eliminar, modificar las bases de datos y crear, eliminar, modificar, buscar y extraer los datos de estas.
14. Desarrollo de algoritmos que permitan alertar al usuario de los cambios que se producen en la configuración de los sensores.
 - Deben cumplir también con el concepto de escalabilidad, de forma que se puedan añadir nuevas alertas sin tener que modificar la plataforma.
15. Desarrollo de una Interfaz gráfica para el usuario o “Graphical User Interface” (GUI) para configurar las diferentes bases de datos o la plataforma y extraer datos de las medidas de los sensores.
16. Desarrollo y evaluación de experimentos para comprobar la robustez y fiabilidad del servidor junto con la parte Hardware.
 - Al igual que en el apartado anterior se busca conocer si la plataforma funciona correctamente con todas las partes operativas bajo situaciones de funcionamiento normal.

4. MÉTODOS Y EQUIPO

En este capítulo se comienza a desarrollar con mayor detalle la estructura y componentes de la plataforma del Internet de las Cosas que se busca obtener con el proyecto. Para ello, es necesario explicar ciertos fundamentos teóricos tanto de la parte de hardware como de la parte software, para que en el siguiente capítulo se entiendan todos los conceptos. Antes de comenzar a detallar la estructura y organización del proyecto se va a exponer las partes básicas de la arquitectura típica de las plataformas IoT.

Por lo general, la arquitectura se divide en tres partes, en algunos casos no tan diferenciadas. En la Ilustración 10 se puede observar las partes que se describen a continuación: (García Cortiñas, 2018)

- Edge layer: Se trata de la capa física en la que se encuentran los diferentes dispositivos que adquieren los datos de su entorno. Es la capa en la que mayor número de dispositivos se encuentran, pero estos tienen capacidades de procesamiento limitadas, por lo que requieren de otras capas para el análisis de los datos.
- Cloud layer: En esta capa se encuentran los servidores, que permiten el almacenado, visualización y procesado de todos los datos generados por las capas inferiores. Gracias a esta capa se dispone de todos los datos en tiempo real y desde cualquier lugar, en la mayoría de los casos.
- Fog layer: Capa intermedia entre Edge y Cloud. Se trata de una capa difusa, como su nombre indica, que actúa como puente entre ambas capas. Puede dividirse en subcapas, en función de la lejanía e infraestructura que tenga respecto a las otras dos capas. Si solo es una capa suele posicionarse cerca de la Edge layer, de forma que actúa como puerta de enlace o “Gateway”, debido a que posee las tecnologías y protocolos necesarios para comunicarse con ambas capas.

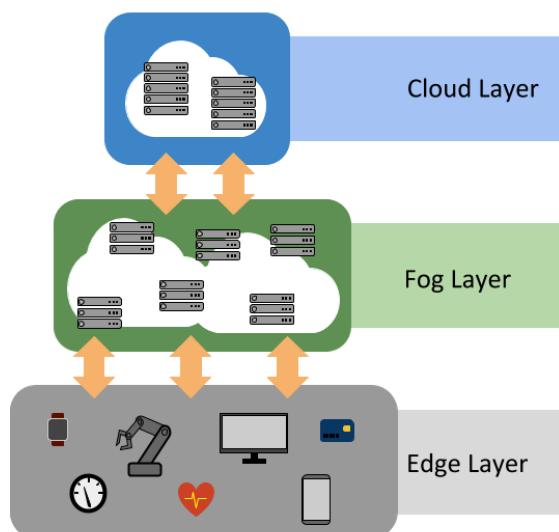


Ilustración 10 - Cloud, Fog and Edge layers of IoT platforms architecture.⁵

⁵ (García Cortiñas, 2018)

4.1 ORGANIZACIÓN DEL PROYECTO

Una vez se conoce la arquitectura típica de las plataformas IoT, se puede comenzar a detallar la estructura utilizada en el proyecto. También cuenta con las tres capas, las cuales están bien diferenciadas.

En la Edge layer se encuentran todas las placas destinadas a la adquisición de datos, llamadas placas de sensores. Estas se distribuyen por un mismo piso del bloque de viviendas para la adquisición de los parámetros de eficiencia energética. En el mismo piso se encuentra la Fog layer, que se trata de una placa con dos MCU. Uno se comunica con las diferentes placas de sensores y el otro permite la conexión a Internet. Ambos MCU se comunican mediante UART. En la Ilustración 11 se puede observar una representación poco detallada de esta parte de la plataforma.

En la Ilustración 11 se pueden observar las diferentes placas de sensores representadas mediante la placa de desarrollo de Texas Instruments. Estás se comunican mediante Sub-1GHz con el HUB central, también llamado "Gateway" o colector, que está compuesto por una placa de desarrollo y un microcontrolador ESP32. El colector se comunica mediante WiFi a la red personal de la vivienda para transmitir los datos a la Cloud layer.

Esta parte de la plataforma está más ligada a la búsqueda de componentes, diseño, fabricación, ensamblado y testeo de las placas. Es por ello, que se va a referir a toda esta parte del proyecto como Parte Hardware, frente a la que se detalla a continuación.

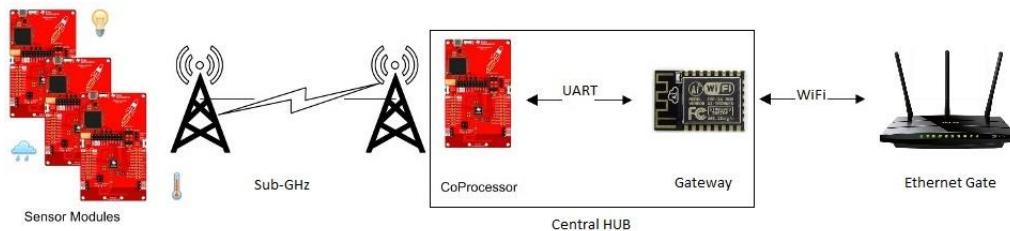


Ilustración 11 - Edge and Fog layer: Hardware diagram.⁶

Para la siguiente capa, la Cloud layer, se va a referir como Parte Software. Esto se debe a que corresponde con el almacenado, visualización y procesamiento de los datos que se realiza en un "servidor". Se compone solo de un elemento físico, que de no ser porque es un proyecto en fase inicial, incluso se podría prescindir de él. Para reducir costes, no se ha contratado un servicio de Cloud como los analizados en el capítulo 2. Para simular el servicio Cloud, de forma que la escalabilidad y la migración de plataformas sea lo más sencilla posible, se ha utilizado una Raspberry Pi como servidor.

En la Ilustración 12 se puede apreciar una representación de cómo es la arquitectura global del servidor. Se ha representado también el MCU del Gateway para entender de donde se reciben los mensajes, pero no pertenece al servidor.

Los mensajes llegan a través de MQTT al servidor. Este los procesa y los almacena en la base de datos. Se dispone de dos bases de datos, una es para los datos transmitidos por los sensores, para la que se usa Influx Database como base de datos. La segunda almacena información de las placas de sensores y de la plataforma. Para visualizar los datos se utiliza el servicio Grafana que permite una visualización óptima. El servidor interacciona con el usuario a través de un navegador o dispositivo móvil para poder monitorizar los datos. Para modificar o extraer datos de la plataforma solo se

⁶ Elaboración propia

permite mediante navegador. Además, cuenta con un sistema de alarmas a través de Telegram.

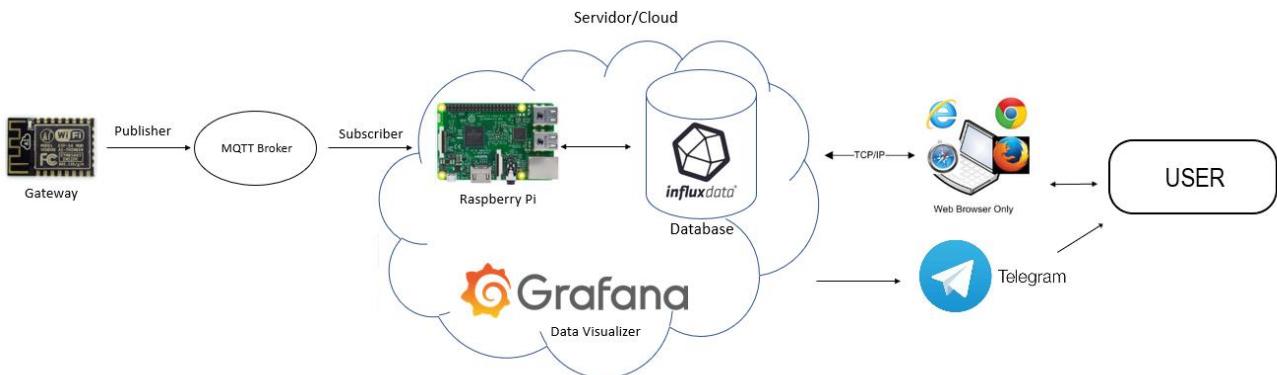


Ilustración 12- Cloud layer: Software diagram⁷

Después de definir las partes en las que se divide el proyecto, se puede entender la complejidad del mismo. Se trata del desarrollo completo de una plataforma desde la base, donde se realizan las medidas, hasta la gestión de la información en la Nube. Esto requiere del conocimiento de múltiples tecnologías, protocolos, lenguajes de programación, así como de que hardware se adapta correctamente a las necesidades del proyecto.

Para la organización del proyecto existen varias etapas en las que las tareas que se van a describir no coinciden necesariamente con el desarrollo temporal de la plataforma. En la primera etapa se organiza e investiga acerca de los elementos y tecnologías necesarias para el desarrollo de la plataforma. Uno de los requisitos era la utilización de un MCU de Texas Instrument. Por lo tanto, se aprende el funcionamiento de los dispositivos de Texas, con su software específico, Code Composer Studio (CCS). También comienza a familiarizarse con softwares de diseño y prototipado de PCBs, como es el caso de OrCAD.

Para la Parte de Software ya se disponían de conocimientos previos, lo que simplificó el proceso. Sin embargo, se tuvo que investigar acerca de cuál sería la mejor base de datos y la forma de visualizarlos de la que no se tenían conocimientos.

En la siguiente fase se van alternando el desarrollo de las dos partes de la plataforma. Por la Parte de Hardware, se comienza con el diseño electrónico de la adquisición de datos, seguido del prototipado de las placas de sensores. Además de las placas de sensores, se tiene que diseñar la placa del colector. Para la realización de pruebas iniciales se usan las placas de desarrollo de Texas Instruments y posteriormente se diseñan las PCBs específicas. Para realizar las pruebas iniciales se montan los circuitos en placas de modelado y se crean los firmwares que se introducirán en los MCU.

Por la Parte de Software, se desarrollan los algoritmos en MicroPython para transmitir la información desde el colector al servidor. Se establece la comunicación MQTT y se desarrollan algoritmos en Python de la parte del servidor para poder almacenar y gestionar la información que se recibe del exterior. Para almacenar esa información se desarrollan dos bases de datos. Una, encargada de almacenar los datos de los sensores, para la que se utiliza InfluxDB como sistema de almacenamiento. Y una segunda que almacena la información de la plataforma y de los sensores, en la que se usa un archivo al que se accede mediante comunicación REST gracias a Python.

⁷ Elaboración propia

Una vez se tiene almacenada toda la información, son necesarias aplicaciones para monitorear y gestionar dicha información. Para monitorear y visualizar los datos, es necesario implantar la aplicación Grafana en el servidor y configurar el panel principal para visualizar los datos. Además, se desarrolla una interfaz gráfica de forma que el usuario pueda gestionar los datos de las bases de datos, añadiendo, modificando, eliminando y extrayendo, teniendo control completo de la plataforma. También se implementa un sistema de alarmas en caso de que se añada un nuevo sensor, se apague o en caso de batería baja. Esto se realiza mediante la implantación de un Bot en Telegram que se comunica con la plataforma.

Una vez se completa lo anterior, se entra en la siguiente fase. En esta se centra exclusivamente en el ensamblado de las PCBs con los distintos componentes. Se trata de una fase de precisión en la que se tiene que realizar el trabajo con prudencia ya que si se comete errores en el ensamblado de los componentes las placas de sensores no funcionaran correctamente. Además, se trata de una fase que transcurre en la última parte del proyecto, por lo que no se dispone de mucho tiempo y por lo tanto de oportunidades de fallo.

Finalmente, se lleva a cabo una serie de pruebas que buscan comprobar si la plataforma y las diferentes placas de sensores funcionan correctamente.

4.2 FUNDAMENTOS TEÓRICOS DEL PROYECTO

En este apartado se explican las características básicas de una plataforma IoT y posteriormente se detallan características específicas del proyecto a desarrollar. De esta forma, se busca que el lector comprenda cómo funcionan este tipo de plataformas y el motivo de haber escogido las tecnologías adecuadas.

4.2.1 Principios básicos generales

Como ya se ha mencionado, en este apartado se realiza un repaso de las características fundamentales de las plataformas IoT.

4.2.1.1 Redes inalámbricas de sensores y actuadores

Gracias a la reducción de costes, de consumo energético y miniaturización de los componentes electrónicos, que se han detallado en el capítulo 1, se ha hecho posible el incremento de los sensores. Los datos obtenidos por un sensor no son útiles por sí solos, es necesario extraer esos datos y procesarlos. En la búsqueda de mejorar la eficiencia de este proceso, sobre todo cuando se tiene más de un sensor, es cuando aparecen las diferentes topologías de redes inalámbricas de sensores y actuadores o WSAN.

Estas redes se componen de diferentes nodos que pueden ser sensores o actuadores inalámbricos distribuidos y usualmente interconectados. Estos nodos interaccionan con los diferentes elementos de su entorno tomando información de este, o realizando acciones para poder controlar las variables deseadas. En la Ilustración 13 se puede observar un tipo de topología de red en el que aparecen numerosos nodos y otros elementos que se describen a continuación. (Salarian, Chin, & Naghdy, 2012)

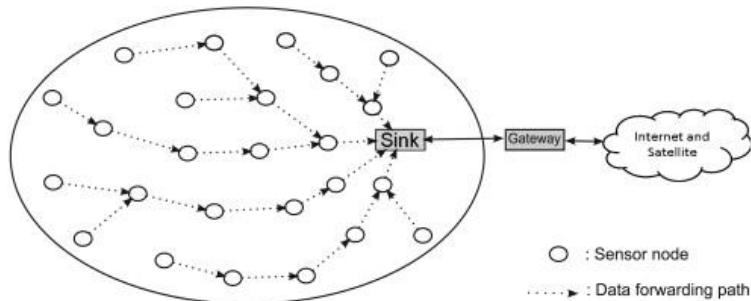


Ilustración 13 - Wireless Sensor Actuator Network⁸

- Nodos de sensores:

Son los encargados de realizar las medidas en el entorno. Normalmente la red se compone de muchos nodos de sensores, aunque el número depende de la aplicación de la red. A pesar de que los componentes de los nodos dependerán también de la aplicación, suelen tener partes comunes: Un MCU que gestiona toda la información del nodo, desde las comunicaciones, los datos que se adquieren o el consumo de batería. Disponen de un sistema de comunicación inalámbrico para comunicarse con otros nodos o con el Gateway. Disponen de un circuito de alimentación de energía y de uno o varios sensores que realizan las medidas.

- Nodos de actuadores:

Disponen de los mismos elementos que los nodos de sensores, pero estos en vez de tomar medidas, realizan acciones que modifican el entorno. Suelen activarse o desactivarse en función de los valores adquiridos con los nodos de sensores, de forma que se obtenga un escenario deseado.

- HUB central, Gateway o colector:

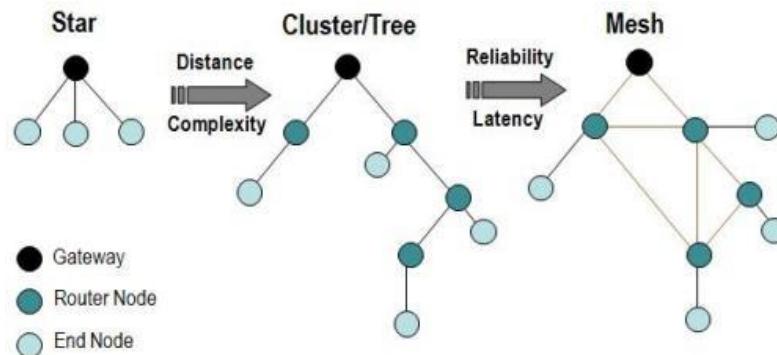
Se trata del nodo que controla toda la red de sensores y actuadores. Dispone de tecnología para comunicarse con la red y a la vez con el exterior, ya sea directamente con el usuario final, con otro colector o con la Nube. Toda la información pasa por este punto lo que permite configurar la topología de la red.

- Servidor o servicio Cloud:

Permite almacenar y disponer de la información de la red en cualquier lugar. Recibe la información de uno de los Gateway. Gracias a este servicio el usuario puede monitorizar los datos obtenidos y actuar en consecuencia, logrando una gestión completa de la plataforma.

Una vez se conocen los diferentes elementos de la red, es muy fácil relacionarlos con la arquitectura básica de una plataforma IoT. La Edge layer se correspondería con los nodos, la Fog layer sería el Gateway y la Cloud layer sería el servidor. Como se puede observar, es común que la Edge y Fog layer se encuentren relativamente cerca. Así pues, en función de la aplicación deseada la red compuesta por estas dos capas dispone de una topología u otra. En la Ilustración 14 se puede observar las topologías más comunes que se describen a continuación:

⁸ (Salarian, Chin, & Naghdy, 2012)

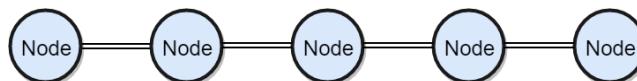
Ilustración 14 - Common WSAN topologies⁹

- Topología estrella:

Se trata de la topología más simple. Todos los nodos se conectan al HUB central por el que pasa toda la información. Sus ventajas son su sencillez, escalabilidad, ya que permite añadir nuevos nodos sin modificar la red, y el bajo consumo de batería, ya que los sensores transmiten su información al HUB central y pueden apagarse. Sus mayores inconvenientes son que no permite transmitir información a tan largas distancias como el resto de las topologías al ser la comunicación nodo – HUB central. El otro inconveniente es que el HUB central tiene que gestionar todas las comunicaciones, lo que en redes con un gran número de nodos puede suponer retrasos en las señales.

- Topología de cadena:

También se trata de una tipología sencilla, en el que los nodos se disponen en serie. Transmiten la información de uno a otro, por lo que estos necesitan disponer de receptor y transmisor. La ventaja de esta topología es que permite transmitir información a largas distancias. Además, se pueden disponer los nodos en forma de anillo cerrando la red y permitiendo la transmisión en ambos sentidos. Los mayores inconvenientes son el coste ya que se necesita disponer de transmisor y receptor y la complejidad al añadir nuevos nodos a la red, ya que hay que abrir la red. Es por todo esto que no suelen ser muy utilizadas. En la Ilustración 15 se puede observar un ejemplo de la topología.

Ilustración 15 - Chain WSAN topology¹⁰

- Topología de árbol:

En esta red los nodos se comunican mediante saltos entre nodos. Se trata de la evolución de la topología de estrella, en la que varias estrellas se unen mediante la topología de cadena. De esta forma se logra obtener las ventajas de ambas topologías. Tienen una estructura jerárquica, por lo que la transmisión es de los superiores a los inferiores.

⁹ (Muthukarpagam, Niveditta, & Neduncheliyan, 2010)

¹⁰ Elaboración propia

- Topología de malla:

Se trata de la unión de todas las topologías anteriores, lo que la convierte en la más compleja de todas. Permite la comunicación jerárquica de la topología de árbol, pero además permite comunicar los nodos del mismo nivel. Esto es una gran ventaja en redes inalámbricas, ya que, si un nodo deja de funcionar debido a la batería o interferencias, la información puede ser transmitida a través de otro nodo. Sin embargo, esto hace que la red sea mucho más cara y compleja.

4.2.1.2 Teoría y componentes para la transmisión de la información

Este apartado se centra en el estudio de las propiedades físicas y componentes necesarios para la transmisión de la información entre los nodos y el gateway. Debido a la complejidad de este tema, simplemente se analizará de forma global las variables relacionadas en el caso más simple posible.

Este caso es un escenario ideal, en el que se tiene un emisor, un receptor y el mensaje que se envía y recibe sin obstáculos, en la línea de visión o “Line Of Sight” (LOS), sin ningún rebote y en un único camino. También se supondrá que la respuesta frecuencial del canal es plana, sin interferencias de otras ondas. Una vez aclaradas las hipótesis, se describen los componentes físicos relacionados en la transmisión de la información.

Para transmitir un mensaje de un dispositivo a otro es necesario la utilización de antenas. Desde la parte del emisor, las antenas traducen una señal eléctrica en una onda electromagnética que viajará por el canal hasta el receptor. Por lo general, y sobre todo en el mundo del IoT, se trabaja con dispositivos digitales. Por lo tanto, se necesita de un elemento en el transmisor que traduzca los bits en la señal eléctrica a transmitir. Estos componentes existen de la misma forma en el receptor, pero realizando el trabajo inverso. Existen más componentes en esta cadena, pero este es el caso más simple. En la Ilustración 16 se puede apreciar los componentes.

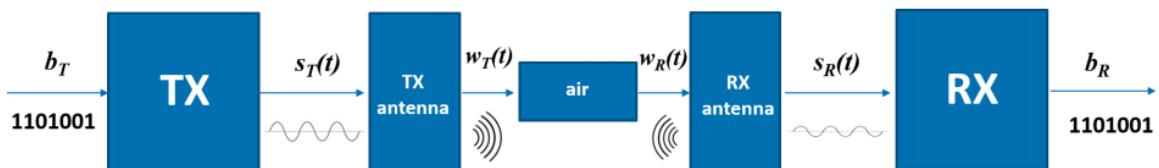


Ilustración 16 - Transmission chain¹¹

Debido a esta cadena existe un retraso entre la señal b_T y b_R , en el que cuantos más componentes existan mayor será el retraso. De igual modo, a mayor distancia, mayor retraso. También se debe tener en cuenta que los componentes y el medio introducen ruido a la señal.

Para analizar las propiedades físicas más relevantes, se hace uso del Teorema de Shannon. Este relaciona el máximo de datos que se pueden transmitir por el canal, en función del ancho de banda y de las interferencias por ruido. Su ecuación es:

$$Rb \leq B \log_2(1 + \frac{P_{RX}}{P_N})$$

Ecuación 1 - Shannon equation

¹¹ Elaboración propia

Siendo:

Rb: Bitrate o capacidad del canal [bit/s]

B: Ancho de banda [Hz]

P_{RX} : Potencia de la señal en el receptor [W, mW]

P_N : Potencia del ruido en el canal [mW, μ W]

La $\frac{P_{RX}}{P_N}$ es la relación señal/ruido o “Signal Noise Ratio” (SNR). Desarrollando este término para obtener las propiedades del medio y del transmisor, se utiliza la ecuación de Friis y la ecuación de la potencia del ruido en el canal según el modelo de Johnson Nyquist de ruido blanco gaussiano aditivo o “Additive White Gaussian Noise” (AWGN):

$$P_{RX} = P_{TX} \frac{G_{TX} G_{RX}}{\left(\frac{4\pi df}{c}\right)^2}$$

Ecuación 2 - Friis equation

$$P_N = BKT$$

Ecuación 3 - Noise power model as AWGN from Johnson Nyquist model

Siendo de la Ecuación 2:

P_{TX} : Potencia de la señal en el transmisor [mW, μ W]

G_{TX} y G_{RX} : Ganancias de la antena del transmisor y receptor

d: distancia recorrida por la onda entre transmisor y receptor [m, km]

f: frecuencia de la onda [Hz, MHz]

c: velocidad de la luz [m/s]

$\left(\frac{4\pi df}{c}\right)^2$: Atenuación del medio: A(f)

Y de la Ecuación 3:

K: Constante de Boltzmann [J/K]

T: Temperatura del medio [K]

F: Factor de ruido; La cantidad que se reduce de la señal por el ruido.

Aplicando la Ecuación 2 y Ecuación 3 en la Ecuación 1, se obtiene la siguiente ecuación que relaciona la capacidad del canal con las diferentes variables que se buscan estudiar:

$$Rb \leq B \log_2 \left(1 + \frac{P_{TX} G_{TX} G_{RX}}{BKT \left(\frac{4\pi df}{c} \right)^2} \right)$$

Ecuación 4 - Developed Shannon equation

Un R_b alto, supone una gran capacidad del canal y por tanto se tendrán más datos en menor tiempo, esto es menor latencia. Esta característica es imprescindible en aplicaciones como dispositivos IoT, descargas clouding, visionado de video de alta calidad, realidad virtual, realidad aumentada, aplicaciones en tiempo real como coches autónomos, drones, cirugía por control remoto, etc.

Para incrementar R_b , es necesario aumentar B , pero el espectro a frecuencias medias se encuentra ocupado por las aplicaciones actuales, por lo que es necesario moverse a frecuencias altas. Esto es un problema ya que la atenuación aumenta con el cuadrado de la frecuencia como se observa en la Ecuación 2. Por lo tanto, se tiene que reducir la distancia de transmisión para que la atenuación no aumente en gran medida, lo que causa una densificación de la red.

Otra opción para aumentar R_b , que es la que se utiliza en el proyecto, sería buscar grandes anchos de banda (B), pero en vez de aumentar frecuencias, disminuirlas. De esta forma se evita los problemas derivados del aumento de la atenuación. Sin embargo, una reducción de la frecuencia implica un aumento de la longitud de onda, que se relaciona de manera proporcional con el tamaño de las antenas. Esto hace que los dispositivos sean más grandes y costosos por la antena. A cambio, se obtienen tasas de transmisión considerables a largas distancias, característica imprescindible para aplicaciones IoT.

También se podría aumentar P_{TX} para tener mayor R_b , pero esta última aumenta de forma logarítmica con la potencia. Además, supondría un aumento en el coste de antenas, menor tiempo de vida de las baterías y problemas para la salud.

4.2.1.3 Servicio NTP

“Network Time Protocol” se trata de un protocolo desarrollado por Dave Mills para sincronizar los relojes de los diferentes dispositivos conectados a la red de servidores horarios o “network time server”. (David L., James J., Jack L., & William T., 2010)

Se compone de una estructura jerárquica piramidal de servidores que se encuentran sincronizados con los niveles superiores. En la cúspide se encuentran relojes atómicos que permiten obtener la hora a todos los servidores y clientes con una precisión de 200 μs y un error menor a 10 ms en el mejor de los casos. En la base de la pirámide se encuentran los clientes que pueden sincronizarse con los servidores y actualizar la hora de sus dispositivos. En la Ilustración 17 se puede observar la estructura del servicio NTP.

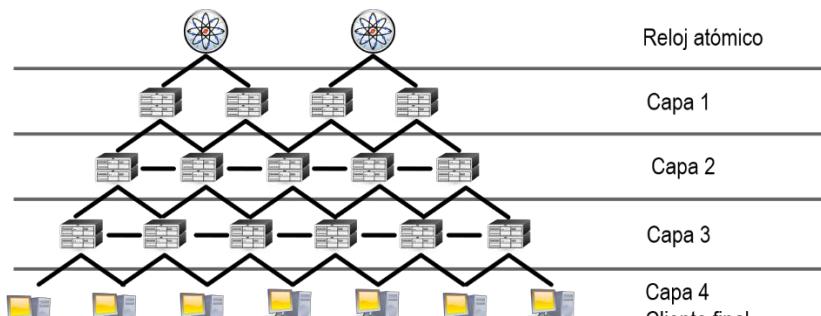


Ilustración 17 - NTP service structure¹²

Esta estructura permite calcular la latencia de los paquetes enviados entre los diferentes puntos, de forma que no se cometen errores y permite independencia de la

¹² (Wikipedia, 2007) y elaboración propia

zona horaria. Estos paquetes se envían mediante el protocolo UDP/IP debido a que posee mayor seguridad y rapidez.

Los dispositivos electrónicos disponen de un reloj interno que por lo general está fabricado con cristal de cuarzo. Este reloj permite llevar un registro de la hora en el dispositivo, lo cual es fundamental para MCU. Sin embargo, estos cristales disponen de impurezas, lo que hace que a la larga cometan errores horarios. Para aplicaciones de IoT estos errores no se pueden permitir, ya que suponen grandes errores en la medida.

Gracias a este protocolo se logra corregir los errores horarios siempre y cuando el dispositivo pueda conectarse a la red de servidores. De esta forma, configurando el servicio y realizando una solicitud cada cierto tiempo al servidor el dispositivo puede actualizar su hora.

4.2.1.4 Arquitectura monolítica vs microservicios

Se trata de dos tipos de arquitecturas de desarrollo de software que desembocan en la misma acción final, pero la realizan de formas muy diferentes. Por ello, surge el paradigma a la hora de desarrollar una aplicación o un servicio web. En la Ilustración 18 se puede ver la comparativa entre ambas arquitecturas que se explicarán a continuación: (Loureiro & Chiles, 2016)

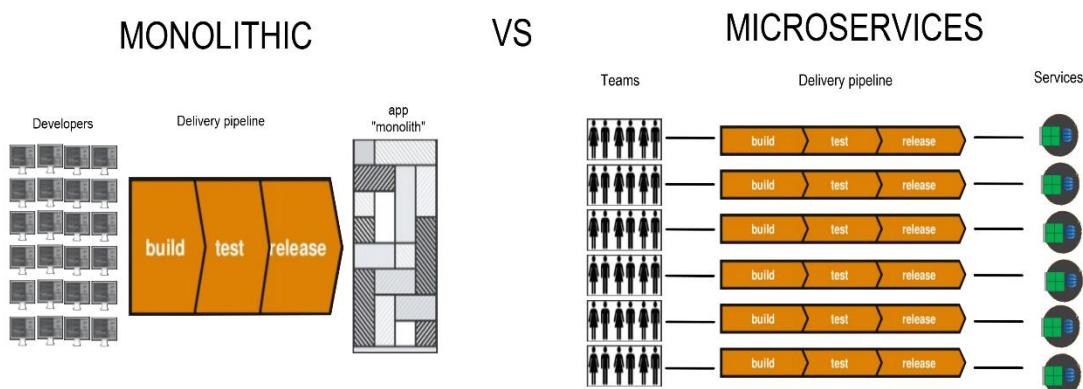


Ilustración 18 - Monolithic vs Microservices architecture¹³

Comenzando con la arquitectura monolítica, esta se basa en el desarrollo de una aplicación o servicio único que permite realizar una o varias tareas. En el caso de que trabajaran varios desarrolladores, todos ellos trabajarían en la misma tarea hasta que se complete toda la aplicación. Esta estructura hace que el código base sea largo, que no esté bien definido a qué desarrollador pertenece cada parte de código y por tanto, largos ciclos en la fase de desarrollo. Sin embargo, la fase de desarrollo, testeo y despliegue de la aplicación es más sencillo. Además, como todo el equipo se encarga de toda la aplicación está muy centralizado.

Con el paso de los años, los códigos y lenguajes de programación han aumentado en cantidad y longitud. Esto hace que una aplicación pueda disponer de varios lenguajes de programación y que el código base pueda estar fragmentado en códigos más pequeños. Estas novedades hacen que las arquitecturas monolíticas fallen, ya que para actualizarlas o añadir nuevos componentes es necesario modificar toda la estructura de la aplicación. Esto desemboca en una falta de agilidad, de innovación y en clientes insatisfechos debido a la escasa escalabilidad de la aplicación.

¹³ (Loureiro & Chiles, 2016) y elaboración propia

De estos problemas surge la arquitectura de microservicios. Se basa en el desarrollo de aplicaciones formadas por pequeños servicios bien definidos, que pueden funcionar independientes y se comunican entre ellos mediante protocolos ligeros. Por lo tanto, se trata de servicios pequeños, independientes que desarrollan su propia tarea.

Esto hace que cada servicio se pueda desarrollar en paralelo, haciendo que equipos independientes trabajen a la vez reduciendo el tiempo de la fase de desarrollo. También hace que la fase de testeo sea más sencilla, y toda la plataforma se entienda con facilidad al poder observar cada módulo por separado. Esto resuelve los problemas de la arquitectura monolítica, ya que se dispone de códigos fragmentados, que pueden funcionar con lenguajes de programación diferentes, lo que los vuelve más potentes a su vez.

Como se puede observar, se aporta mucha flexibilidad y escalabilidad, ya que se pueden incorporar o modificar servicios sin alterar el resto de la plataforma, haciendo el despliegue más continuo. Además, los desarrolladores no necesitan saber cómo funcionan el resto de los servicios, siendo estos más especializados en un servicio concreto. En mantenimiento también se encuentran ventajas; si aparece un problema, este será fácilmente identificable y por lo tanto solucionado sin que toda la aplicación se bloquee al funcionar cada servicio independientemente. Por lo tanto, esta arquitectura aporta una solución escalable y flexible, lo que hace que los clientes dispongan de un buen servicio.

Sin embargo, también existen cosas negativas. Es necesario un registro exhaustivo de todas las modificaciones, de las APIs y protocolos que dispone cada microservicio de forma que no existan errores ni llamadas incorrectas entre servicios.

Este paradigma es similar al que se puede apreciar hoy en día en numerosas compañías internacionales. Antiguamente estas disponían de una estructura organizativa jerárquica, en la que las tareas se ordenaban desde arriba para abajo como si de una pirámide se tratara. Sin embargo, hoy en día ese paradigma cambia y se puede apreciar como las estructuras organizativas tienden a ser más flexibles. Disponen de diferentes departamentos que trabajan en su área específica y colaboran con el resto para desarrollar el trabajo de la compañía.

4.2.1.5 Modelos de servicio

Cuando se habla de modelos de servicio en el ámbito de “Cloud computing”, se refiere a la metodología que se sigue a la hora de trabajar en la nube. Como se analizó en el apartado 1.1.2, la aparición de la computación en la nube permite un modelo de negocio más dinámico ya que se adquieren los servicios en función de las necesidades del cliente. Existen varios tipos de modelos de servicio, en los que se ofrece “algo” como servicio, en inglés X as a Service (XaaS).

Entre estos tipos se diferencian tres: Infraestructura como servicio (IaaS), Plataforma como Servicio (PaaS) y Software como Servicio (SaaS). En la Ilustración 19 se puede apreciar una comparativa entre ambas que aclara visualmente las diferencias. “On-premises” corresponde al caso en el que el cliente gestiona todo el servicio en la nube con sus propios servidores (Greiner, 2014). Analizando cada tipo, empezando desde el alojamiento de la aplicación entera, hasta el más básico se tiene:

Separation of Responsibilities

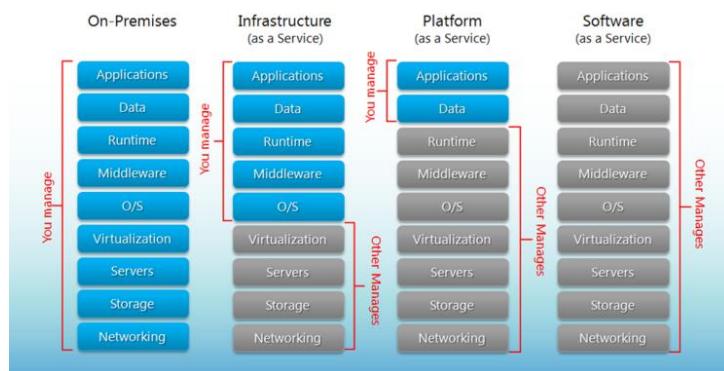


Ilustración 19 - On-Premises, IaaS, PaaS, SaaS services¹⁴

a) Software como servicio (SaaS)

Es el tipo de servicio en el que la plataforma de “Cloud computing” se encarga de toda la gestión de la plataforma. Se convierte en una alternativa muy interesante para empresas en las que su actividad económica no está relacionada con el mundo de la tecnología como actividad primaria. Este servicio le permite a este tipo de compañías ahorrar costes ya que solo necesitan alquilar el servicio de la compañía proveedora.

De esta forma, el cliente que adquiere el producto no necesita comprar el producto, ni una infraestructura donde instalarlo, ni mantenerlo o configurarlo, el servicio en la nube se encarga de ello. En un mundo en el que las empresas deben ser flexibles ante la demanda de los consumidores, esto les permite poder cambiar de servicio si la tendencia cambia, sin necesidad de modificar todo el sistema, lo que supone un ahorro de costes. Además, el proveedor generalmente provee de un grado de personalización en la aplicación que hace que las empresas que lo adquieran se puedan diferenciar.

Se trata de un servicio óptimo para PYMES o empresas que no están especializadas en ese sector, ya que ahorran costes de mantenimiento, gestión, desarrollo y seguridad.

b) plataforma como servicio (PaaS)

Es el siguiente eslabón en la cadena. En este caso, el proveedor proporciona y gestiona la plataforma al completo, pero el cliente tiene el control sobre la aplicación que se aloja en la infraestructura del proveedor. De esta manera, el cliente puede ahorrar costes de infraestructura, mantenimiento y configuración de la plataforma, que pueden suponer una gran inversión inicial. En este tipo de servicio se paga por el uso que se haga de la infraestructura de la plataforma.

Es por esto, que lo convierte en una opción ideal para desarrolladores de aplicaciones, ya que pueden centrarse en el desarrollo de su aplicación, dejando el mantenimiento y gestión al proveedor. Esto hace que se reduzcan enormemente los tiempos de las fases de implementación y despliegue de una aplicación. Además, al confiar en plataformas líderes del sector, se tiene la certeza de que se dispondrá de la mejor tecnología y con garantía de seguridad.

¹⁴ (Greiner, 2014)

c) Infraestructura como servicio (IaaS)

De los modelos de servicio, este es el que permite mayor flexibilidad al usuario que lo adquiere, pero por lo tanto este deberá tener más conocimientos en la gestión de plataformas. El proveedor se encarga de mantener y disponer de la infraestructura que el cliente necesita y por lo tanto alquila para su uso. Este último se tiene que encargar por lo tanto del alojamiento y gestión de la plataforma completa.

Al no tener que comprar la infraestructura, el usuario ahorra costes iniciales de compra y de mantenimiento. Sin embargo, a diferencia de las anteriores, el usuario se encarga de la escalabilidad de la plataforma teniendo que modificar sus aplicaciones según las necesidades que surjan.

El usuario solo paga por la infraestructura propiamente dicha, como capacidad de computo, memoria, gestión de las comunicaciones y su mantenimiento, es decir, por lo que usa. Por lo tanto, se convierte en un servicio valioso para empresas del sector tecnológico que no buscan realizar un desembolso en infraestructura ni encargarse del mantenimiento o seguridad por causas económicas o por desconocimiento.

4.2.2 Principios básicos específicos

Una vez analizadas las características que podrían poseer las plataformas IoT, se va a dar paso a desarrollar las características fundamentales que se aplican al proyecto. Sin embargo, todavía no se va a explicar cada bloque, que aparece en el capítulo 5 del proyecto.

4.2.2.1 Tecnología de comunicación: Sub-1 GHz y Wi-Fi

- *Elección de la banda adecuada*

Como se ha observado a lo largo de la documentación, es necesario disponer de una tecnología y protocolo de comunicación para que la red inalámbrica de nodos funcione correctamente. Para descubrir cuál es la que mejor se adapta al proyecto, es necesario analizar las tecnologías disponibles, como ya se ha tratado en el capítulo 2, y analizar las características que se buscan satisfacer en el proyecto. Estas características se pueden sintetizar en: (1) rango de conexión efectiva, (2) consumo de potencia, (3) La velocidad de transmisión y (4) el tamaño del dispositivo. Además de esta serie de características, existen otras que también se deben tener en cuenta como la normativa, robustez ante interferencias o coste. (Silicon Laboratories Inc., 2010)

Si se observan las bandas más utilizadas para aplicaciones industriales, científicas y médicas, o aplicaciones en el mundo de IoT, aparecen dos opciones: 2.4 GHz y Sub 1-GHz. En la primera se encuentran las principales tecnologías como Bluetooth Low Energy (BLE), Wi-Fi o ZigBee. La segunda se trata de una banda menos ocupada, con tecnologías menos conocidas, pero muy útil para aplicaciones de bajas tasas de datos.

Por lo tanto, se va a analizar cuál de las dos bandas es mejor para el proyecto a desarrollar según las características del proyecto.

a) Rango de conexión

En cuanto al rango de conexión se tiene que Sub-1GHz es muy superior a 2.4 GHz. Esto se debe a dos motivos principales, a su atenuación debido a la frecuencia y a los obstáculos. En la Ilustración 6 se puede observar una comparativa entre las tecnologías que se están analizando. (Youkhana, 2016)

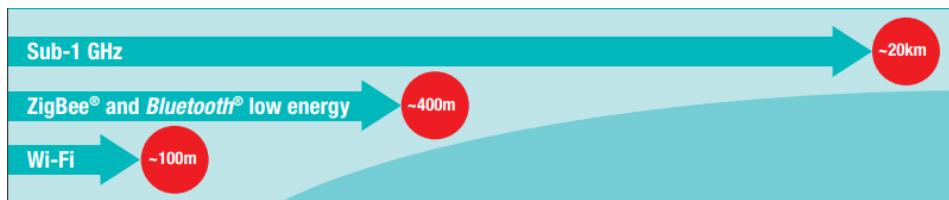


Ilustración 20 – Technologies range comparision¹⁵

Atendiendo a la atenuación debido a la frecuencia. Si se observa la parte de atenuación de la Ecuación 1 en el apartado 4.2.1.2, al aumentar la frecuencia, la atenuación se incrementa de forma cuadrada. Esto hace que, para obtener el mismo rango efectivo en el caso de 2.4 GHz se necesiten dispositivos intermedios, que hacen más complicada y costosa la red. Además, Sub-1 GHz al poder transmitir a mayores distancias, hace que la desventaja de la tipología de red estrella desaparezca. Por lo tanto, se simplifica aún más la red y el coste. (Copley, 2015)

Si se analiza la atenuación debida a los obstáculos, cuando una onda atraviesa un obstáculo, como puede ser una pared, la señal se debilita debido a la absorción y reflexión. Los obstáculos absorben mejor las ondas de mayor frecuencia que las de menor frecuencia. Además, en cuanto a la reflexión, de la misma forma las de mayor frecuencia tienden a reflejarse más y por tanto se disipan rápidamente. Por todo esto, en ambientes con muchos dispositivos, los que funcionan a 2.4 GHz se ven, muy afectados en calidad e interferencias de la señal. (Hellan, 2016)

También debido a los obstáculos aparece otro fenómeno. Cuando la onda choca con una esquina esta se curva. Las ondas de menor frecuencia poseen un ángulo de difracción mayor, lo que las permite rodear más el obstáculo y por lo tanto no atenuarse debido a absorción o reflexión.

b) Consumo de potencia

En cuanto al consumo de energía, existen dos motivos por los cuales Sub-1 GHz tiene ventaja frente a su rival. Por un lado, se posee un ciclo de trabajo menor y por otro lado se tienen menos interferencias.

El primer motivo responde a que, en el caso de no existir ningún paquete en el canal, el MCU se puede poner en modo dormir o "sleep". De esta forma, cada cierto tiempo solo el módulo de radio analiza el canal en busca de paquetes para el dispositivo. En caso de no existir ninguno valido, el módulo de radio se duerme sin interrumpir al MCU y por lo tanto reduce enormemente los consumos. (Copley, 2015)

El segundo motivo se debe a las interferencias. Al existir menor número de dispositivos en la banda de Sub-1 GHz, la probabilidad de interferencia se ve disminuida. Además, como ya se ha analizado en el punto (a) de rangos de la señal, se reduce la atenuación ante obstáculos y distancias. Todo esto hace que sean necesarios menos intentos para lograr que un paquete sea obtenido en el receptor y por lo tanto menor consumo de energía. (Silicon Laboratories Inc., 2010)

También existe otro motivo que hace que Sub-1 GHz consuma menos energía. El receptor de la señal, así como el transmisor, necesita de una serie de componentes para traducir la señal desde su forma de onda, hasta los bits que el MCU entiende. Entre estos circuitos se encuentran amplificadores, que consumirán mayor energía a mayor frecuencia y filtros, que al tener que filtrar mayor ancho de banda, introducirán más ruido en la señal, lo que se traduce en mayor consumo de energía por perdida de paquetes.

¹⁵ Recorte de (Youkhana, 2016)

c) Tasa de transferencia de datos

Como ya se apuntó en el punto 3.2.1.2, existen varios factores que limitan la tasa de transferencia de datos. Como ya se comentó, aplicaciones en las que se necesite que sea elevada, es mejor trabajar a altas frecuencias. Es por esto por lo que la banda 2.4 GHz se convierte en la mejor candidata para la función.

Sin embargo, si se tiene un sistema WSAN compuesto por numerosos nodos que transmiten pequeños paquetes de información en períodos de tiempo largos, se puede prescindir en cuanto a tasa de transferencia frente a otras características como rango o consumo de potencia.

d) Tamaño

Poco a poco el tamaño de los dispositivos inalámbricos ha ido disminuyendo considerablemente. Esto se debe en gran parte a la miniaturización de los componentes y a la mejora de la maquinaria destinada a ensamblar las placas. Sin embargo, cuando se refiere a antenas, a pesar de los avances en los materiales y en tecnología, existen propiedades físicas que no permiten miniaturizar la antena sin que los costes se disparen y en algunos casos es físicamente imposible.

De nuevo, en el punto 3.2.1.2, ya se trató el tema de las antenas debido a la frecuencia de la onda a emitir o recibir. Concretando para el caso que se estudia, en Sub-1 GHz, el tamaño de las antenas es superior respecto a 2.4 GHz. Cuanto menor es la frecuencia, mayor es el tamaño de las antenas.

Además de las ventajas que ofrece la banda Sub-1 GHz frente a 2.4GHz en WSAN, dispone de otra característica adicional llamada salto de frecuencia, del inglés “frequency hopping”. Esto permite a los nodos realizar saltos del canal en el que transmiten a lo largo del tiempo, según su disponibilidad. Por lo tanto, la probabilidad de interferencia disminuya y por tanto la necesidad de mandar de nuevo el mensaje, que consumirá energía. (Vijayasankar & Movva, 2016)

Se acaban de analizar las características básicas de las dos bandas que mejor se adaptan a las características del proyecto. De estas se puede concluir que la banda Sub-1 GHz es óptima para el uso entre la comunicación de las diferentes placas de sensores y el HUB central de la plataforma. Se obtendrán grandes rangos de señal y consumos de potencia bajos, necesarios ya que las placas se alimentan con baterías. Como la tasa de transferencia de datos no es un factor limitante y en cuanto al tamaño se ha intentado miniaturizar al máximo, se ha escogido como tecnología a implantar. Junto con esta banda, se implanta la tipología de red estrella ya que, gracias a las características ya nombradas, es la más sencilla y barata.

Sin embargo, esto no quiere decir que no se vaya a usar la banda de 2.4 GHz. Se hace uso de esta banda con la tecnología Wi-Fi para la comunicación del HUB central o Gateway y el router que permiten el acceso de la plataforma a internet. Esta placa está conectada a la red eléctrica, ya que la tecnología Wi-Fi consume mucha potencia.

- *Tecnología de comunicación*

En cuanto a la tecnología de comunicación usada, no se va a entrar en mucho detalle, ya que se usa un software proporcionado por Texas Instruments. La elección de este viene dada por el objetivo de desarrollar una plataforma con un MCU de Texas Instruments, por lo que el software usado tiene que ser TI 15.4-Stack. Se trata de un software para el desarrollo de redes de bajo consumo, largo alcance, robustas y

seguras, especializado para tipologías de red estrella y bandas de Sub-1 GHz y 2.4GHz. Es por esto que se trata de una solución perfecta para el proyecto.

Se basa en el estándar IEEE 802.15.4 disponiendo de capa de aplicación, MAC y PHY como se puede ver en Ilustración 21, que se puede comparar con la Ilustración 9. Permite modificar numerosos parámetros de comunicación reduciendo así el tiempo de desarrollo de la aplicación. (Texas Instruments, Out-of-box star-network solution: TI 15.4-Stack, 2019)

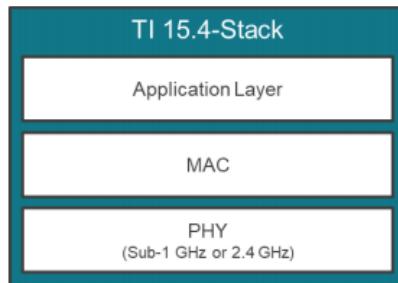


Ilustración 21 - TI 15.4-Stack layers¹⁶

- *Normativa*

Con los puntos anteriores se habría estudiado todos los aspectos tecnológicos de la comunicación de la parte Hardware de la plataforma. Sin embargo, todavía queda por analizar un aspecto fundamental para el correcto despliegue de la plataforma, la normativa de telecomunicaciones. Debido a que en la tecnología Wi-Fi no se puede configurar nada a nivel físico, solamente se analiza la que aplica a Sub-1 GHz.

La normativa impone unos límites que los diseñadores de dispositivos deben de cumplir. Esta es diferente en función del continente en la que se comercialice el producto, como puede ser la europea (EU) o la estadounidense (US). Los límites suelen aplicarse a frecuencias, tipo de modulación, potencia emitida, etc.

Centrando el estudio a la normativa europea, esta clasifica los dispositivos inalámbricos, llamados dispositivos de corto alcance, del inglés "Short-Range Devices" (SRD), según su aplicación. Existen ciertas bandas que están reservadas, las llamadas bandas industrial, científica y médica, del inglés "Industrial, scientific and medical band" (ISM), en las que se restringe a los desarrolladores transmitir. En la Tabla 2 se puede observar la clasificación según las aplicaciones. (Loy, 2005)

Aplicaciones	
1	Non-specific Short Range Devices
2	Equipment for Detecting Avalanche Victims
3	Local Area Networks, RLANS and HIPERLANs
4	Automatic Vehicle Identification for Railways
5	Road transport and traffic Telematics
6	Equipment for detecting Movement and Equipments for Alert
7	Alarms
8	Model Control
9	Inductive Applications
10	Radio Microphones
11	RF Identification Systems
12	Ultra-Low Power Medical Implants
13	Wireless Audio Applications

Tabla 2 - Aplicaciones según la regulación en las telecomunicaciones europeas

¹⁶ (Texas Instruments, Out-of-box star-network solution: TI 15.4-Stack, 2019)

Por lo tanto, los desarrolladores de dispositivos se tienen que centrar en la banda específica a “Non-specific SRD” o dispositivos de corta distancia no específicos. El problema es que estas bandas estarán más ocupadas que el resto. Dentro de las bandas de los SRD no específicos se limitan la banda de frecuencias, el ciclo de trabajo, el ancho de banda o la potencia efectiva radiada, del inglés “Effective Radiated Power” (ERP). En la Tabla 3 se puede observar las limitaciones.

Frequency Band	ERP	Duty Cycle	Ch. Bandwidth	Remarks
433.05 – 434.79 MHz	+10 dBm	< 10%	No limits	No Audio and Voice
	0 dBm	No limits	No limits	≤ -13 dBm/10 kHz, no Audio and Voice
433.05 – 434.79 MHz	+10 dBm	No limits	< 25kHz	No Audio and Voice
868.00 – 868.60 MHz	+14 dBm	< 1%	No limits	
868.70 – 869.20 MHz	+14 dBm	< 0.1%	No limits	
869.30 – 869.40 MHz	+10 dBm	No limits	< 25kHz	Appropriate Access Protocol Required
869.40 – 869.65 MHz	+27 dBm	< 10%	< 25kHz	Channels may be combined to one high speed channel
869.70 – 870.00 MHz	+7 dBm	No limits	No limits	
2400.0 – 2483.5 MHz	+7.85 dBm	No limits	No limits	Transmit power limit is 10-dBm ERP

Tabla 3 - Limits of non-specific SRD

En este Proyecto las bandas de frecuencia que interesan son las que se encuentran en el rango de 868 – 870 MHz para la comunicación Sub-1 GHz y las que van de 2.4 GHz – 2.4835 GHz para Wi-Fi.

Además de estas limitaciones que se deben a la banda de frecuencias, existen otras específicas al tiempo del ciclo de trabajo. Estas hacen referencia al tiempo máximo que se puede transmitir en una hora en función del ciclo de trabajo y de la cantidad de transmisiones. En la se puede observar estas limitaciones.

Duty-Cycle Limit	Total On Time Within One Hour	Maximum On Time of One Transmission	Minimum Off Time of Two Transmission
< 0.1%	3.6 seconds	0.72 seconds	0.72 seconds
< 1%	36 seconds	3.6 seconds	1.8 seconds
< 10%	360 seconds	36 seconds	3.6 seconds

Tabla 4 - Duty cycle time limits

Por lo tanto, a la hora de desarrollar la parte Hardware de la plataforma se han tenido en cuenta todas las restricciones que impone la normativa, para que la plataforma sea viable en un futuro.

4.2.2.2 Protocolos de comunicación: MQTT y REST

En el punto anterior se han analizado las tecnologías de comunicación inalámbricas necesarias para el proyecto y sus propiedades. Por lo tanto, en este punto se analizan los protocolos que utilizan esas tecnologías para la comunicación.

En el proyecto se utilizan más protocolos de los que se van a tratar en este punto. Sin embargo, entre todos ellos destacan dos, el protocolo MQTT y REST. Destacan

debido a que son los protocolos para los cuales se ha programado y modificado parámetros. Existen otros protocolos de comunicación, como el propio de Texas Instruments, EasyLink, a través del cual se comunican los nodos de la WSAN o TCP/IP o UDP/IP que se han utilizado en el proyecto.

Los protocolos que se describen a continuación son utilizados en la parte Software de la plataforma. El primero para transmitir los datos entre las dos partes de la plataforma y el segundo para la comunicación interna entre los bloques del servidor. En el capítulo 2, apartado 2.2.2 se encuentra la definición de ambos protocolos.

- **MQTT**

Se trata de un protocolo de comunicación asíncrono, simple y ligero. Se basa en una red tipo estrella, en el que los nodos publican (publishers) y se suscriben (subscribers) a un tema determinado. Para mediar las comunicaciones existe un actor intermedio, llamado bróker, que conecta los publishers con los subscribers. Es un protocolo ideal para redes en las que el ancho de banda es reducido, la latencia no es relevante o redes que no son estables. Es por esto por lo que se convierte en un protocolo perfecto para aplicaciones IoT en las que no se necesite actividad en tiempo real.

Por lo tanto, en una red MQTT se tiene que disponer de al menos tres actores. Un Publisher que publica el mensaje en un tema. El bróker, que gestiona las comunicaciones entre los publishers y los subscribers, pudiendo almacenar información de ellos. Y un subscriber que se suscribe a un tema por el que recibe mensajes. Esta sería la red más sencilla, pero se puede tener múltiples clientes tanto publishers como subscribers que publican o se suscriben a varios temas. Además, un cliente puede funcionar como publisher y como subscriber a la vez. Esto permite crear una red muy flexible, en la que se pueden añadir fácilmente nuevos clientes. (IEEE, Hunkeler, Truong, & Stanford-Clark, 2008)

En la Ilustración 22 se puede apreciar una representación de la estructura generalizada de la comunicación MQTT.

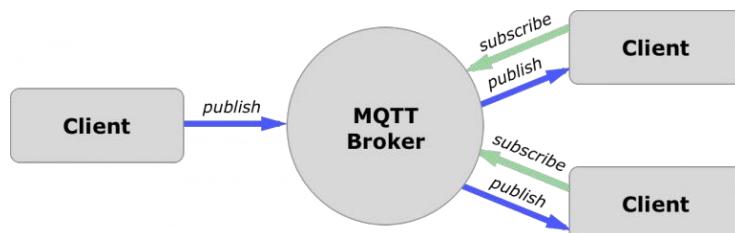


Ilustración 22 - General architecture of MQTT communication¹⁷

En lo referente al tema, no es necesario configurarlo, simplemente basta con publicar o suscribirse en él. Los temas se componen de separadores (/) que generan una jerarquía entre las palabras, parecido a lo que ocurre con las carpetas en un ordenador. A causa de esta sencillez y flexibilidad hace que este protocolo sea muy extendido utilizándose en plataformas de renombre como Facebook Messenger. A continuación, se muestra un ejemplo de un posible tema, en el que los valores entre "< >", podrían modificarse para enviar a un subscriptor u a otro:

"Home_efficiency_project/room/<name_room>/sensor_type/<sensor_type>"

En cuanto al mensaje, este se compone de tres partes. Un encabezado fijo, un encabezado variable y el mensaje en sí. De esta forma, se obtiene un protocolo de baja carga, con un consumo mínimo y anchos de banda bajos. En estos encabezados se

¹⁷ (Shriram, Deekshit, Roopa, & Jaya Kathuria, 2019)

guarda información importante acerca de la comunicación y conexión con el bróker. Entre ellas se encuentra el tipo de mensaje, si la conexión es temporal o durable o el sistema de calidad, en inglés “Quality of System” (QoS). Estas dos últimas son configurables y se explican a continuación. (Rouse, IoT Agenda, 2019)

Para crear un cliente se deben seguir una serie de pasos: (1) crear una instancia u objeto de un cliente MQTT. (2) Esta instancia debe conectarse al bróker. (3) La conexión debe mantenerse viva para que exista flujo de mensajes. (4) El cliente se subscribe o publica a un tema. (5) Cuando se quiere terminar el proceso, si es un subscriber, primero debe de desuscribirse del tema. (6) Terminar la conexión con el bróker.

Cuando el cliente está conectado y suscrito a un tema, esta conexión puede ser temporal o durable. Si es temporal, cuando el cliente se desconecte, el bróker no almacenará información de este. Si es durable, a pesar de desconectarse, el bróker almacenará por un tiempo su información. Esto hace que la próxima vez que se conecte el cliente, no tenga que volver a mandar toda la información, lo que se traduce en menores tiempos y consumos en la reconexión. Estos ahorros se verán incrementados sobre todo en casos en los que la red sea inestable y tenga que reconnectarse muchas veces.

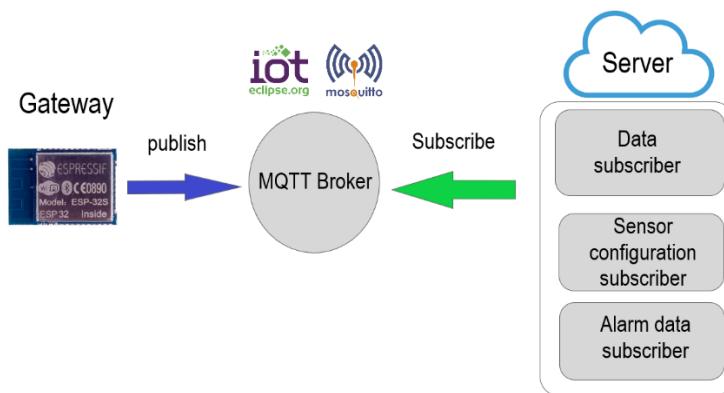
Si se habla de QoS, se refiere a la seguridad con la que el mensaje es enviado o recibido por el cliente. Existen tres niveles de calidad:

- QoS 0: Menor grado de calidad, en el que el mensaje es enviado solo una vez sin confirmación por parte de los actores.
- QoS 1: El mensaje se entrega al menos una vez, por lo que se asegura que llegará. El receptor devuelve una señal cuando lo recibe.
- QoS 2: El mensaje se entrega una vez, y existe doble confirmación por parte del cliente y bróker de que ha ocurrido. Los actores se preguntan inicialmente si están preparados para el envío y recepción del mensaje. En caso afirmativo se realiza la comunicación.

Según sea más alto el grado de calidad, más certeza se tiene de que el mensaje llega a su destino, sin embargo, se consumen más tiempo y por lo tanto batería. Por lo tanto, es decisión del desarrollador elegir el QoS en función de la aplicación deseada.

Cuando se habla de comunicaciones a través de Internet, una característica muy importante es la seguridad. MQTT permite el cifrado TLS, que asegura un correcto funcionamiento. El problema es que, al aumentar la complejidad, aumenta la carga por parte del MCU y la red. Es otra decisión del desarrollador en la que debe elegir entre seguridad y sencillez con bajos consumos.

En la Ilustración 23 se observa la estructura de la comunicación MQTT del proyecto que se está realizando. En ella se aprecia el Gateway que actúa como publisher, publicando la información de los sensores. En el medio se encuentra el bróker, que se trata del bróker suministrado por iot.eclipse.org de acceso público. Existen muchos tipos de brókers, este funciona bajo el software Mosquitto que se verá en capítulos posteriores. Por último, está el servidor que se subscribe a los temas creados por el publisher. El servidor dispone de tres subscriptores diferentes para realizar las tareas necesarias. Uno procesa los datos de los sensores, otro la configuración de estos y el tercero, comprueba si ciertas condiciones se cumplen para informar al usuario.

Ilustración 23 - MQTT communication of the platform¹⁸

- REST (HTTP)

Como ya se apuntaba en el capítulo 2, apartado 2.2.4, este protocolo está diseñado para la creación de servicios web o “Web services”. Según W3C, un web service es un software diseñado para soportar interacciones machine-to-machine (M2M) en la red. Esto es, dos dispositivos electrónicos que se comunican a través de Internet. Por lo tanto, se usan tecnologías como HTTP para realizar la comunicación.

La arquitectura REST se basa en HTTP para llevar a cabo su cometido. De esta forma, la comunicación se realiza mediante peticiones entre cliente-servidor o M2M, a través de URLs y un puerto determinado, en la que se pueden enviar parámetros. Estas peticiones de acceso o manipulación de datos entre cliente y servidor se realizan usando una serie de operaciones sin estado predefinidas. Estas operaciones predefinidas o métodos se basan en los métodos de HTTP, usando los métodos CRUD (Create, Read, Update, Delete) como se ve en la Tabla 5.

HTTP	REST - CRUD
POST	Crear
GET	Leer
PUT	Actualizar/Modificar/Reemplazar
DELETE	Borrar

Tabla 5 - HTTP/REST methods

Al igual que en HTTP, en REST se dispone de códigos de estado cada vez que se realiza una transmisión del mensaje. Esto permite conocer cual ha sido el resultado de la comunicación. En la Tabla 6 se muestran los grupos en los que se dividen estos códigos de estado.

HTTP códigos de estado	Clase
1xx	Información
2xx	Éxito
3xx	Redirección
4xx	Error de cliente
5xx	Error de servidor

Tabla 6 - HTTP status codes

Se trata de un protocolo orientado a los recursos. Un recurso representaría cualquier tipo de información, ya sea un documento, texto, imágenes, etc. Cuando un cliente realiza una petición al servidor, lo que obtiene es una representación de ese recurso

¹⁸ Elaboración propia

que se encuentra almacenado en el servidor. En el caso en el que se busca modificar un recurso, lo que hace es enviar uno nuevo el cliente. Para identificar a los recursos se dispone de identificadores universales de recursos, en inglés “Universal Resource Identifier” (URI). Un recurso debe tener al menos un URI que funcione como su dirección para acceder a él.

A continuación, se muestra un ejemplo. Se trata de una URL, que accede a la dirección “localhost”, en el puerto “8080”. El URI sería cualquier cosa que vaya detrás de la barra (/), que en este caso es “hello_world”. Las URI funcionan de forma jerárquica, al igual que el tema en MQTT, mediante barras (/). Ante esta petición, el servidor responderá con el recurso que se encuentre en esa URI.

“http://localhost:8080/hello_world”

Al igual que MQTT, como REST se basa en HTTP, se dispone de encriptación TLS y SSL. Esto hace que sea un protocolo seguro y sencillo, ya que son estándares familiares para los desarrolladores.

En el proyecto el uso de REST está limitado a la comunicación entre los diferentes bloques que componen el servidor. Este se encuentra alojado en un mismo dispositivo, que es una Raspberry Pi. Es por ello, por lo que no sería necesario que los bloques realizaran peticiones a través de Internet para acceder a los recursos internos, ya toma más tiempo. Sin embargo, el proyecto se desarrolla buscando optimizar la flexibilidad y escalabilidad, siguiendo la arquitectura de microservicios que se analizó en el capítulo 4, apartado 4.2.1.4.

De esta forma, se puede identificar correctamente los diferentes web services dentro del servidor, facilitando la gestión, mantenimiento e identificación de errores. Además, esto permite que los servicios de la aplicación se puedan dividir entre varios servidores, aportando mayor seguridad ante caídas de la red a la plataforma. También aporta mayor seguridad ante bloqueos de la plataforma. En caso de bloquearse un web service, simplemente fallará una parte de la plataforma dejando el resto funcionando.

4.2.2.3 Sistemas operativos en tiempo real

Como su nombre indica, se trata de sistemas operativos que se diseñan para el desempeño de tareas en tiempo real. De esta manera es capaz de gestionar varias tareas y recursos a la vez en las que el tiempo y la sucesión entre tareas es relevante.

Son más complejos que los sistemas operativos que solo permiten realizar una tarea sin interrupciones, pero a cambio poseen características interesantes. Entre ellas destacan: (Texas Instruments, Texas Instruments Resource Explorer - RTOS concepts, 2019)

- Permiten planificar una serie de tareas según su importancia.
- Rapidez en la ejecución de tareas.
- Permiten la interrupción de procesos.
- No demandan grandes cantidades de recursos computacionales o de memoria.
- Disponen de menos servicios que los sistemas operativos convencionales.

Uno de los objetivos principales del proyecto es el uso de un MCU de Texas Instruments, como se puede ver en el capítulo 3. Por lo tanto, el sistema operativo viene marcado por el que proporciona el fabricante: Texas Instruments Real Time Operating Systems kernel (TI RTOS).

Este sistema operativo se engloba dentro de un sistema que denominan TI RTOS que controla otras funciones como la energía, las comunicaciones, sistemas de archivos, drivers, etc. TI RTOS se divide en diferentes módulos que se comunican a

través de APIs. Por encima de TI RTOS existe otro nivel llamado Portable Operating System Interface o POSIX, que no es un RTOS exactamente, pero facilita el uso a usuarios menos avanzados. En la Ilustración 24 se puede apreciar tanto POSIX, como los diferentes módulos de TI RTOS. (Texas Instruments, Texas Instruments Resource Explorer - TI-RTOS basics, 2019)

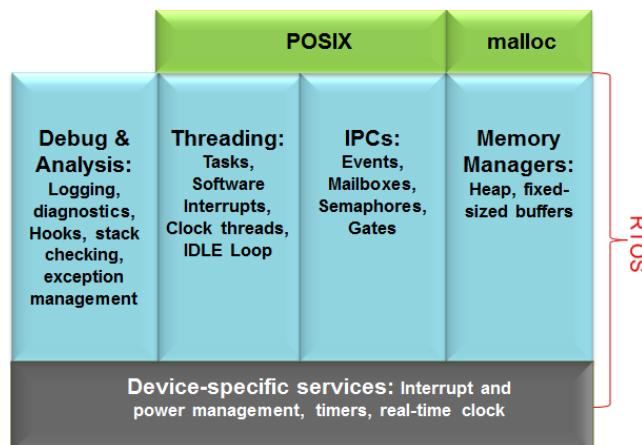


Ilustración 24 - TI RTOS modules¹⁹

Una de las tareas principales de TI RTOS kernel es el planificador de tareas, o en inglés “Scheduler”. Este se encarga de ejecutar de manera precisa las tareas en función de su prioridad. TI RTOS dispone de cuatro tareas que se diferencian por su prioridad:

- Interrupción hardware (Hwi):

Son las de mayor prioridad, se ejecutan hasta completarse sin que nada las bloquee. Son las interrupciones provocadas por dispositivos hardware. Tienen latencia cero, lo que implica que se ejecutan inmediatamente. El problema es que no disponen de APIs como el resto.

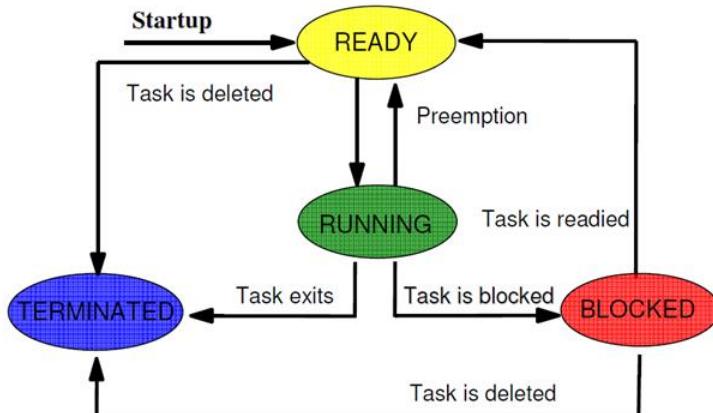
- Interrupción software (Swi):

Son idénticas a las anteriores, lo único que se inicializan mediante software. Comparten el mismo bloque de memoria que las interrupciones hardware.

- Tareas (Tasks):

Son las tareas más comunes. Disponen de un bloque de memoria propio, por lo que se pueden bloquear. No hay un número máximo de tareas. Pueden tener varios estados: “Ready”: Está esperando una condición a empezar; “Running”: Se está ejecutando; “Block”: se encuentra esperando a que un recurso esté disponible, no consume recursos de la CPU. “Terminated”: La tarea ha sido eliminada o ha terminado. En la Ilustración 25 se puede observar un diagrama de los estados de las tareas:

¹⁹ (Texas Instruments, Texas Instruments Resource Explorer - TI-RTOS basics, 2019)

Ilustración 25 - Task states diagram²⁰

- Inactivo (Idle):

Se trata de una tarea con la prioridad más baja. Se encarga de tareas de supervisión de los sistemas del MCU. Para reducir el consumo de los MCU, se pueden establecer estados de bajo consumo cuando el MCU se encuentre en la tarea Idle.

Para la comunicación entre tareas, TI RTOS dispone de numerosos mecanismos como: semáforos para controlar recursos compartidos, buzones para compartir información, colas para crear listas de datos, puertas para proteger el acceso a áreas críticas o eventos que permite la sincronización de procesos. Además, dispone de módulos de gestión de la memoria y sincronización.

Para el caso que se trata en el proyecto, el TI RTOS funciona perfectamente, ya que permite ejecutar varias tareas a la vez de forma precisa. De esta forma, las placas de sensores pueden ejecutar varias tareas como pueden ser: la adquisición de la medida que controlan, el envío de la información al HUB central y el modo Idle para mantener el MCU en bajo consumo cuando no tenga que realizar ninguna tarea. De esta forma, el MCU se encontrará en modo Idle y cuando una interrupción de hardware o software, como puede ser un cronómetro, despertará al microprocesador que realizará el resto de las tareas.

4.2.2.4 Parámetros asociados a la eficiencia energética

En la parte Hardware es necesario disponer de diferentes placas de sensores para la obtención de los parámetros relacionados con la eficiencia energética. Hoy en día, es fácil obtener los sensores para medir cualquier propiedad del entorno a precios asequibles. Sin embargo, en el caso de que se busquen tolerancias, rangos o precisiones más allá de los casos generales, estos sensores aumentan mucho de precio.

Por lo tanto, en este proyecto se ha decidido desarrollar las placas de sensores enteras, de forma que en la misma placa se disponga del sensor, el circuito para acondicionar la señal, el MCU de Texas Instruments junto con su sistema de alimentación y antena. De esta forma, se consiguen placas personalizadas, que se desarrollan para trabajar de forma óptima en la plataforma.

Una vez se conoce el motivo por el cual se desarrollan las placas enteras, se tiene que definir qué parámetros están relacionados con la eficiencia energética. Según la

²⁰ (Texas Instruments, Texas Instruments Resource Explorer - RTOS concepts, 2019)

compañía de restauración con la cual colabora la universidad Politecnico di Torino, los parámetros más relevantes son: Temperatura interior y exterior de la vivienda, humedad interior y exterior, presión, consumo de energía térmica, consumo de energía eléctrica, calidad del aire, radiación solar exterior, ventilación interior (velocidad del viento), sensores de alerta ante apertura de ventanas o puertas.

Debido a la envergadura del proyecto, no es posible desarrollar todas las placas de sensores necesarias para medir los parámetros que se han mencionado. Se han escogido los que se han considerado más relevantes para la monitorización y cálculo de la eficiencia energética. Además, se han escogido teniendo en cuenta que este proyecto no acaba aquí, de forma que estas placas sirvan como ayuda a futuros desarrolladores de la plataforma. Gracias a que en el desarrollo del proyecto se ha buscado maximizar la flexibilidad y escalabilidad de la plataforma, no debería suponer un gran reto el desarrollo de las placas de sensores restantes.

Los parámetros escogidos han sido temperatura, humedad, presión y consumo energético. Las tres primeras se deben a que son variables ambientales fundamentales muy ligadas entre ellas, lo que permite encontrar sensores que midan las tres a la vez. Además, son variables conocidas por la sociedad por lo que son muy representativas a la hora de hablar de confort. La última se relaciona con el gasto energético que el usuario realiza y que se traduce en un gasto económico. Hoy en día la forma de conocer el consumo es mirando la factura a final de mes. Esto provoca que el usuario no sea consciente de cuánto está gastando tanto energéticamente como económicaamente. Según muestran los estudios, si el usuario conoce cuánto está gastando, este tenderá a reducir ese consumo, lo que se traduce en mejoras en la eficiencia. (Vega, Santamaría, & Rivas, 2015)

- *Variables ambientales*

Se refiere con variables ambientales a las variables de temperatura, humedad y presión. A pesar de ser variables muy conocidas, se va a explicar en qué consiste cada una.

- Temperatura:

Se trata de la dimensión física que mide la transmisión de calor entre dos sistemas. Esta transmisión se realiza del cuerpo más caliente al más frío, buscando el equilibrio térmico. Las unidades más conocidas para medir la temperatura son: el grado Celsius ($^{\circ}\text{C}$), el Kelvin (K) y el grado Fahrenheit ($^{\circ}\text{F}$). Las dos primeras son las más extendidas en Europa, siendo el Kelvin la medida del Sistema Internacional.

- Humedad:

Esta unidad cuantifica la cantidad de vapor de agua contenida en el aire. Cuando se habla de humedad, existen tres formas de definirla:

- a) Humedad absoluta: Se define como la masa de vapor de agua que contiene un volumen determinado de aire. Se trata de una densidad medida en kg/m^3 , siendo aproximadamente en la atmósfera entre 0 y 30 g/m^3 . Debido a que depende de la temperatura y presión, no es una medida muy utilizada.
- b) Humedad relativa: Es la cantidad de vapor de agua que hay en un volumen de aire frente a la cantidad máxima de vapor de agua que ese volumen de aire puede contener en condiciones de saturación. Por lo tanto, se trata de un porcentaje y equivale a la presión parcial del vapor frente a la presión de saturación.

c) Humedad específica: Es la relación entre la masa de vapor de agua y la masa total de aire en un volumen determinado. Al igual que la primera, su uso es menos habitual, restringiéndose a ciertas áreas como la termodinámica.

- Presión:

Mide la fuerza que ejerce un sólido, líquido o gas por unidad de superficie. En cuanto a sus unidades, depende del campo de aplicación tienen más relevancia unas que otras. En el Sistema Internacional, la unidad es el Pascal (Pa). Sin embargo, en meteorología está más extendido el uso del Bar o mbar, siendo 1 bar = 10^5 Pa. También están las atmosferas (atm) 1 atm = 1.01325×10^5 Pa, siendo la presión a nivel del mar. Otra unidad relevante es el milímetro de mercurio (mmHg), siendo 1 mmHg = 133.3 Pa, usada en el campo de la medicina.

La obtención de estos tres valores no es un trabajo arduo, hoy en día existen infinidad de dispositivos, electrónicos, y no electrónicos que permiten conocerlos. Sin embargo, cuando se habla de eficiencia energética la mera obtención de los parámetros no resuelve el problema. Esto se debe a dos factores. El primero es que en muchos casos estas tres variables están relacionadas entre sí. Y el segundo, que se relaciona con el primero, es que no existe una temperatura ideal para todo el mundo. Es por esto que se habla de zonas o rangos de confort. Además, también influye la estación o lugar en el que se realiza las mediciones, dificultando todo el proceso. (Actitud Ecológica, Temperatura de confort: ¿cuál es la temperatura ideal para una casa?, 2016)

Para conocer ese estado de confort, será necesario analizar varios factores. Los principales que se deben tener en cuenta son la temperatura exterior y la humedad relativa. El primero no se refiere al valor en sí, sino al hecho que obligará a las personas a ir más abrigadas o menos en función de la estación o del lugar donde se reside. El segundo se debe a su estrecha relación con la sensación térmica.

La relación entre la humedad relativa, la temperatura y la velocidad del viento constituyen la sensación térmica. A pesar de que la temperatura como valor solo puede ser una, la relación con la humedad hace que aparezcan los conceptos de "temperatura seca" y "temperatura húmeda", dando la temperatura equivalente y por consiguiente la sensación térmica.

La temperatura seca se refiere a la temperatura que se obtiene de un termómetro. La temperatura húmeda tiene en cuenta la humedad y la velocidad del aire. Cuanta mayor corriente y menor humedad se tenga en el ambiente, la temperatura húmeda será menor que la seca. La temperatura efectiva o equivalente, relaciona la temperatura seca y húmeda. Existen diferentes fórmulas matemáticas que permiten obtener esta temperatura teniendo las otras dos. (Actitud Ecológica, Temperatura seca, temperatura húmeda y temperatura efectiva, 2017)

Una vez se conoce la temperatura efectiva, será necesario conocer cuál es la temperatura ideal o de confort de la vivienda. Para ello de nuevo existen diferentes programas y estándares que permiten calcularlo. Por ejemplo, el ASHRAE permite obtener la temperatura ideal, conociendo los datos de temperatura seca, humedad relativa y velocidad del viento. También existen normas que nos permiten determinar esta temperatura. En el caso de España, se aplica la norma UNE-EN ISO 7730:2006.

Para el cálculo exacto de la temperatura ideal se tienen todas estas herramientas. Sin embargo, existen valores que se pueden tomar por regla general válidos si no se dispone de toda esa información. Se estipula que 22°C y 25°C son valores ideales para invierno y verano. En cuanto a la humedad relativa, se establece entre un 30 y un 60%, siendo óptimo entre 45-50%

Ahora que se conocen las variables a estudiar y sus valores característicos, se pueden escoger y diseñar los sensores que permitan medir estas variables y cumplan una serie de requisitos establecidos por las necesidades de la empresa de reformas y que cumplan lo explicado hasta ahora. En la Tabla 7 se detallan estos requisitos.

	Rango	Precisión	Unidades
Temperatura interior	0 – 50 / 32 – 112	± 0.3/ ± 0.54	°C / °F
Temperatura exterior	- 40 – 65 / - 40 – 150	± 0.3/ ± 0.54	°C/ °F
Humedad relativa	0 – 100	± 3	%
Presión	260 – 1260 / 7.7 – 32.2	± 1 / ± 0.03	mbar / inHg

Tabla 7 - Temperature, humidity and pressure required values

- *Consumo eléctrico*

El consumo eléctrico se calcula como la potencia eléctrica consumida por unidad de tiempo. Para calcular el consumo eléctrico en viviendas la unidad más extendida es el kilovatio por hora (kWh), que se trata de una unidad de energía. Para calcular esta energía es necesario determinar qué consumo de potencia eléctrica se está realizando en la vivienda en cada momento. Por lo tanto, será preciso obtener las variables que permitan determinar la potencia eléctrica, medida en Watios (W). Estas variables son la tensión (V) y la corriente (I).

En la Ecuación 5 se puede ver esta relación entre las variables descritas. Sin embargo, estas ecuaciones solo son válidas en un periodo de tiempo (t) discreto, en el que ni la tensión ni la corriente varían con el tiempo. Se trataría de un caso ideal, que como aproximación es válida. Simplemente se necesita conocer la tensión y la intensidad y considerarlas constantes en un periodo t . Sumando la energía obtenido en cada periodo se obtendría el consumo eléctrico.

$$P = VI; E = Pt$$

Ecuación 5 - Discrete equation of electrical power and electrical energy

Para obtener un valor más realista tanto de la potencia como de la energía eléctrica, habrá que considerar que tanto la tensión, como la intensidad varían con el tiempo. Estas dos variables están relacionadas entre sí mediante la resistencia por la ley de Ohm. Sin embargo, conocer la carga de una vivienda resulta complicado, ya que está puede variar en el tiempo. Es por esto, que no se va a tratar de sustituir la carga en la ecuación. En la Ecuación 6 se puede observar la fórmula para calcular la potencia y la energía eléctrica de forma continua entre los instantes t_1 y t_2 . Esto se podría extender a intervalos de tiempos infinitos.

$$P(t) = V(t)I(t); E = \int_{t_1}^{t_2} P(t)dt$$

Ecuación 6 - Continuous equation of electrical power and electrical energy

A pesar de que en el caso anterior se obtendría el consumo eléctrico exacto, el cálculo de la potencia continuamente sería demandante para el MCU. Por ello, el método para obtener los parámetros fundamentales será tomar la tensión y corriente cada periodo de tiempo y estimarlo como una recta entre los dos puntos.

Una vez se conoce como se relaciona la potencia y la energía eléctrica con el resto de las variables, es preciso concretar para el caso específico que se estudia en este proyecto. El cálculo de la potencia eléctrica en una instalación de corriente alterna, que es la que se encuentra en todas las viviendas.

El problema de tratar con corriente alterna, es que la tensión y la corriente pueden estar desfasadas entre sí, por lo que aparece el término ángulo de desfase, que no es más que el ángulo de retraso entre la tensión y la intensidad sinusoidal. Por todo esto, en vez de tener solo una potencia, aparecen tres potencias diferentes: La potencia activa (P_a) que se mide en watos, la potencia reactiva (Q) medida en voltio amperios reactivos (VAR) y la potencia aparente (S) medida en voltio amperios (VA). En la Ilustración 26 se puede observar el conocido triángulo de potencias, que forman las tres potencias mencionadas. En la Ecuación 7 se puede observar cómo se calculan cada una de ellas. (Area Tecnologica, 2019)

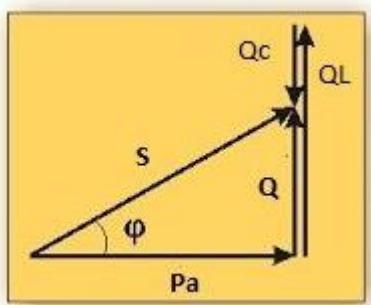


Ilustración 26 – Power triangle²¹

$$P_a = VI \cos\varphi; Q = VI \sin\varphi; S = VI$$

Ecuación 7 – Active power, reactive power and apparent power

La potencia que interesa calcular es la potencia activa, ya que es la que genera potencia útil y por lo tanto trabajo. Por lo tanto, en las ecuaciones para el cálculo del consumo eléctrico se estaba introduciendo un error correspondiente al coseno del ángulo de desfase, conocido como factor de potencia.

La potencia reactiva se considera como energía perdida. Esta potencia depende de las cargas que se conectan a la red. Las cargas pueden ser resistivas, inductivas o capacitivas, que más adelante se explicaran con mayor detalle. Por lo tanto, si la carga es inductiva, aparece la potencia reactiva inductiva (Q_I) que la consumen las bobinas y si es capacitiva, aparece la potencia reactiva capacitativa (Q_C) que la consumen los condensadores.

La potencia aparente es la suma vectorial de la potencia activa y reactiva. Como se puede observar, es preferible que el ángulo de desfase sea el mínimo posible, de forma que toda la potencia se convierta en potencia útil. Para ello es preferible disponer de una red con cargas capacitativas que reduzca la potencia reactiva.

Por lo tanto, para conocer el consumo eléctrico en una vivienda es necesario conocer tres valores, la tensión, la corriente y el factor de potencia. Para saber cómo obtener el último es preciso conocer de donde surge este factor.

Como ya se adelantaba, el ángulo de desfase es el ángulo de retraso entre la tensión y la corriente. Esto tiene sentido solo si se habla de corrientes alternas. Este retraso se debe al tipo de carga que se conecte a la red, que pueden ser resistivas, inductivas y capacitivas. El valor del ángulo de desfase vendrá marcado por el tipo de carga. En la Ilustración 27 se puede observar que valor toma el ángulo en función del circuito.

²¹ (Area Tecnologica, 2019)

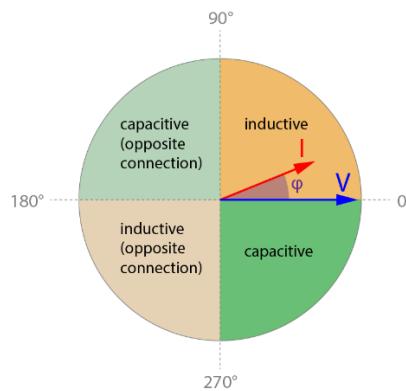


Ilustración 27 - Types of phase shifting depending on the load²²

Se dice que un circuito es puramente inductivo, si el ángulo de desfase forma 90° . Este sería el caso de un circuito formado solamente por una bobina. Por el contrario, si forma -90° o se encuentra en 270° , se habla de capacitivo puro, que podría ser un circuito formado por un solo condensador. En el caso de que el ángulo sea 0° , se trata de un circuito resistivo puro, como puede ser el formado por una resistencia. En el último caso, la potencia aparente coincide con la que se veía en la Ecuación 5 y en la Ecuación 6, que coincide con la de corriente continua.

Observando la representación sinusoidal de la tensión y la corriente en la Ilustración 28 y la Ilustración 29, se puede entender mejor el concepto de ángulo de fase y como obtenerlo. En la Ilustración 28 se puede observar como la corriente está atrasada frente a la tensión, esto se debe a cargas inductivas, y por tanto genera ángulos de entre 0° y 90° . Por el contrario, en la Ilustración 29 se observa como la corriente está adelantada a la tensión debido a cargas capacitivas, generando un ángulo de desfase entre 270° y 360° .

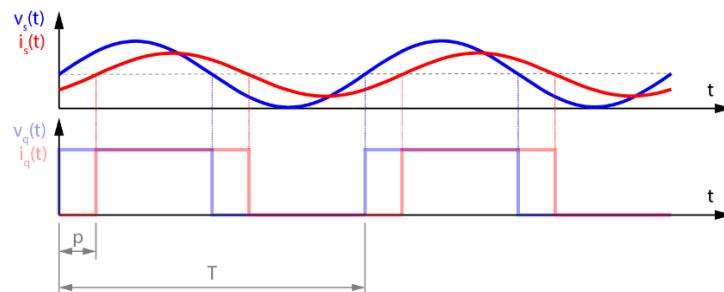


Ilustración 28 - Voltage and current shift phase representation. Current lags the voltage²³

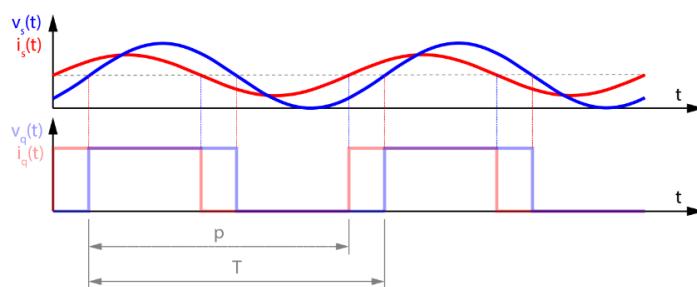


Ilustración 29 - Voltage and current shift phase representation. Current leads the voltage²⁴

²² Elaboración propia

²³ Elaboración propia

²⁴ Elaboración propia

Por lo tanto, para calcular el ángulo de desfase, es necesario conocer el periodo de ambas ondas y cuánto tiempo transcurre dentro de un periodo hasta que ambas ondas se encuentran en la misma fase. Estos valores son los que aparecen en las ilustraciones como T y p. Con la obtención de estos parámetros, es posible obtener el ángulo de desfase a través de la Ecuación 8.

$$\varphi = \frac{p}{T} 360^\circ$$

Ecuación 8 - Shift phase angle equation

Pero para poder obtener estos valores de forma automática y precisa, es necesario acondicionar la señal de entrada de la red, a una que el MCU sea capaz de entender y soportar. Para ello, es necesario una serie de bloques que se explicarán a continuación. Sin embargo, existe un requisito a nivel teórico y de hardware que es necesario cumplir. Se trata de la transformación de la onda sinusoidal analógica, a sus correspondientes valores digitales. Según Nyquist, la frecuencia de muestreo o de adquisición de valores (f_s) debe ser superior a dos veces la frecuencia de la onda a muestrear (f_m). En la Ecuación 9 se puede observar dicha afirmación. (Rouse, Tech Target - Nyquist theorem, 2005)

$$f_s > 2f_m$$

Ecuación 9 - Nyquist Theorem

Por lo tanto, si la red eléctrica en Europa funciona a una frecuencia de 50 Hz, al menos la frecuencia de muestreo tendrá que ser superior a 100 Hz. A pesar de esto, para la aplicación que se busca en el proyecto, será necesario más de dos puntos para poder obtener los parámetros de potencia buscados.

Ya se han analizado todos los aspectos teóricos necesarios para la obtención de los parámetros energéticos. Ahora solo falta conocer cuál será el proceso para acondicionar la señal desde la tensión y corriente de la red eléctrica a un MCU que tome y analice los valores estudiados. En este apartado se definen los diferentes bloques, pero no se entra en detalle sobre los componentes, ya que se analizan en el capítulo 5. En la Ilustración 30 se puede observar la cadena de bloques que componen el sistema, con sus entradas y salidas. Este diagrama está duplicado, uno para medir la tensión de la vivienda, y otro para medir la corriente. Sin embargo, ambos circuitos tienen el mismo diagrama de bloques.

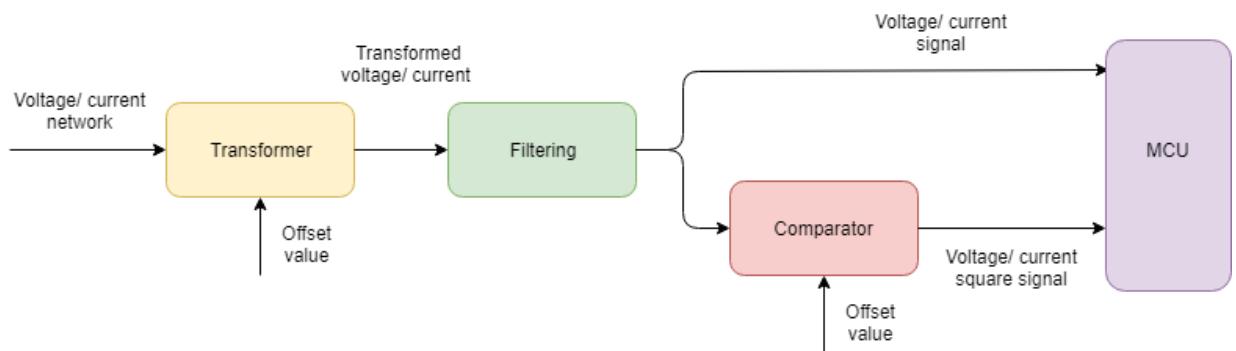


Ilustración 30 - Power measurement block diagram²⁵

Se va a explicar el diagrama de la tensión, pero aplica de igual forma al de corriente. El problema a resolver es que la tensión de la red eléctrica oscila entre los valores $-250\sqrt{2}$ a $250\sqrt{2}$ V como se muestra en la Ilustración 31. Sin embargo, la tensión que

²⁵ Elaboración propia

puede leer el MCU está entre 0 y 3.3 V. Por lo tanto, es necesario reescalar la señal en amplitud, y trasladarla a valores positivos, de forma que se obtenga una señal como la señal superior que se muestra en la Ilustración 32.

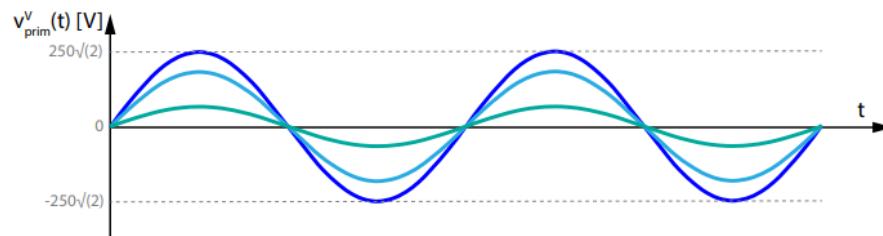


Ilustración 31 - voltage grid waveform representation²⁶

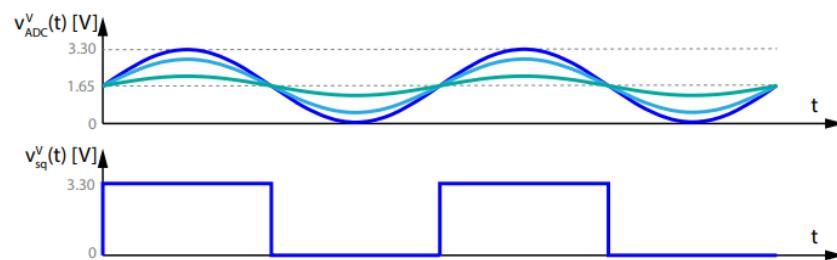


Ilustración 32 - voltage waveform representation in the MCU²⁷

Para ello, primero se comienza con el proceso de reescalado, reduciendo la tensión con el primer bloque de la Ilustración 30. Este transforma la tensión de red a los 3.3 V de amplitud máxima del MCU. Consta de un transformador y divisor de tensión que reducen la tensión a 3.3 V pico a pico. Pero esta tensión todavía necesita que sea positiva. En ese sentido se le suma al valor anterior un valor de offset. Como el rango va de 0 a 3.3 V, este valor de offset debe ser la mitad, es decir, 1.65 V aproximadamente.

Para el caso de la corriente ocurre exactamente lo mismo, a diferencia de que como el MCU no es capaz de leer valores de corrientes, es necesario de un transformador de corriente a tensión. En la Ilustración 33 se puede observar gráficamente el comportamiento de la corriente en la red con la ilustración superior, y de la misma corriente transformada a tensión en la ilustración inferior.

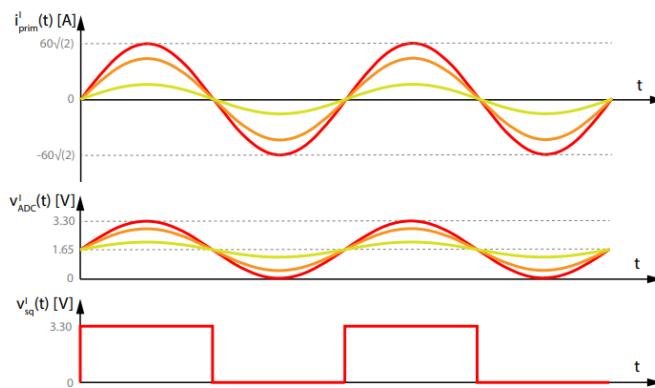


Ilustración 33 - voltage waveform representation in the grid (upper) and in the MCU (bottom)²⁸

²⁶ Elaboración propia

²⁷ Elaboración propia

²⁸ Elaboración propia

En el segundo bloque se aplica un filtro de paso bajo con el objetivo de filtrar los armónicos. Este filtro dispondrá de una frecuencia de corte entorno a los 500 Hz, para que no aparezcan muchos armónicos, pero tampoco se pierda demasiada información de la señal inicial. De este bloque se obtiene una señal de tensión que el MCU es capaz de leer para obtener la potencia.

A pesar de que ya se tiene la señal acondicionada, aparece un tercer bloque que se encarga de comparar la señal de tensión acondicionada con el valor de offset que se introdujo en el primer bloque. Del comparador se obtiene una señal cuadrada, de amplitud 3.3V que se introduce al MCU. Esta señal alterna entre los estados alto (3.3V) y bajo (0V) como se puede ver en la Ilustración 32. Se encuentra en estado alto cuando el valor de la señal de tensión es superior al offset y estado bajo en el caso contrario. Gracias a esta señal cuadrada, se puede conocer el ángulo de desfase que hay entre la señal de tensión y corriente como ya se ha explicado en este punto.

Por lo tanto, se obtendrán cuatro señales finales, dos serán las correspondientes a las señales de tensión y corrientes ya acondicionadas y otras dos serán las señales cuadradas de tensión y corriente. Con todos estos datos ya se puede proceder al cálculo de la potencia como se ha explicado con anterioridad.

Por último, falta conocer cuáles van a ser los valores que el circuito tiene que soportar en su uso normal. Como ya se ha visto, el MCU solo soporta tensiones positivas y que no sean superiores a 3.3V. Por ello, se tiene que diseñar el circuito con márgenes ante posibles casos de picos de tensión y corriente.

Centrándose en los límites de tensión y corriente de entrada de la red, la tensión en una vivienda doméstica oscila en torno a los 230V. Por lo tanto, con disponer de un transformador de tensión que soporte picos de hasta 250V es suficiente. Sin embargo, para la corriente, este valor cambia en función de los dispositivos electrónicos que se conecten dentro de la vivienda.

Para conocer los límites de corriente se utiliza la potencia contratada en la vivienda. La potencia contratada es la potencia máxima que se puede consumir en la vivienda. En caso de que se supere, el Interruptor de Control de Potencia (ICP) cortará la electricidad en la instalación. Este límite de potencia se contrata con la compañía, si se quiere un límite superior, se debe pagar más en la factura. Hoy en día con los contadores inteligentes el ICP queda obsoleto disponiendo de un contador de tele gestión que funciona de forma similar. En una vivienda de tamaño medio, la potencia contratada habitual varía entre 3.45 kW, 4.6 kW o 5.75 kW, en función de la cantidad y uso de electrodomésticos, siendo más común 3.45 kW o 4.6 kW.

Por lo tanto, la corriente máxima que debe soportar el transformador en amperios (A) estará entre 15 A y 20 A. Así pues, los transformadores deben soportar 230V y 20A, sobredimensionando el circuito ante posibles fluctuaciones y picos en la red.

4.2.2.5 Protocolos de comunicación entre sensores y MCUs

Para la comunicación entre MCU y los sensores o entre MCUs es necesario disponer de protocolos para que esta sea efectiva. En este punto se van a tratar los protocolos que se utilizan o pueden ser implementados para el intercambio de información entre los actores. Estos protocolos son:

- **UART**

Se trata de una comunicación serial, que responde a protocolo Universal Asíncrono Receptor-Transmisor, en inglés “Universal Asynchronous Receiver-Transmitter”. Se basa en el envío secuencial de bits entre dos dispositivos, los cuales deben configurarse para que el mensaje enviado pueda ser interpretado por el receptor.

Para ello es necesario definir una serie de bits en el propio mensaje como el bit de comienzo (start bit) el cual marca el inicio de la transmisión pasando de 1 a 0. Posteriormente se envía el mensaje el cual puede ser de 7 a 9 bits de datos, siendo el primer bit el menos significativo. Para finalizar la transmisión se envía un bit de parada (stop bit) aunque normalmente cuando se trabaja con 8 bits de mensaje, se añade un bit de paridad (parity bit) para detectar si ha habido errores en el mensaje. Por último, se define baudrate, es decir, la tasa de transmisión de bits. En la se puede observar un ejemplo de la topología del mensaje.

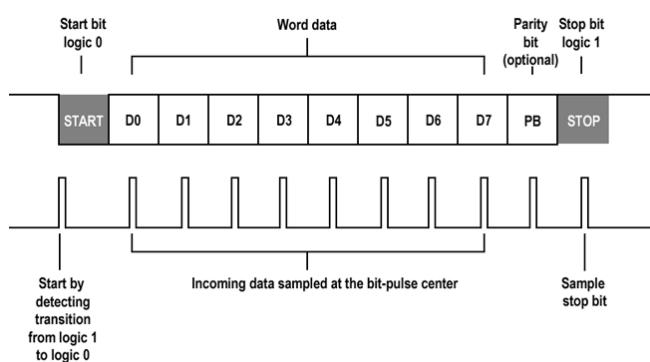


Ilustración 34 - UART communication²⁹

En el proyecto este tipo de comunicación se utiliza para el intercambio de información entre los MCU en la placa del HUB central. Por un lado, se tiene un MCU que funciona de colector recibiendo la información de las placas de sensores por Sub-1 GHz. Por otro lado, se tiene otro MCU que funciona como Gateway transmitiendo esa información a la nube a través de Wi-Fi. Para que ambos MCU se comuniquen se establece el protocolo UART, ya que es fácil y rápido de implementar. En la se puede observar el diagrama típico de conexión para la comunicación UART, en la que se tiene un emisor y un receptor, transmitiendo por la línea TxD y recibiendo por RxD.

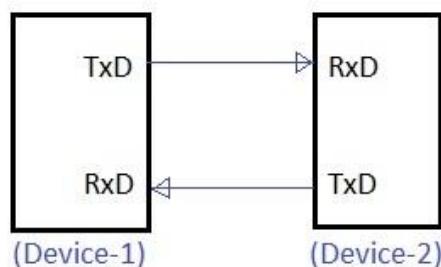


Ilustración 35 - UART communication diagram³⁰

²⁹ (Electric imp DevCenter, s.f.)

³⁰ (RF Wireless World, 2019)

- *I_C*

Se trata de nuevo de una comunicación serial, que responde a protocolo Inter Circuitos Integrados, en inglés “Inter-Integrated Circuits”. Se trata de un protocolo destinado al intercambio de información entre diferentes dispositivos. Existen dos tipos de actores, los maestros que regulan la comunicación y los esclavos que obedecen a los dispositivos maestros. En la Ilustración 36 se puede observar la estructura habitual de comunicación. (Keim, All about circuits, 2015)

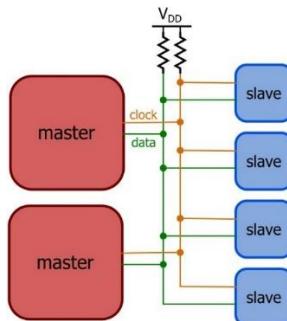


Ilustración 36 - I_C communication diagram³¹

En la ilustración se puede observar los diferentes actores y las líneas necesarias para llevar a cabo la comunicación. Estas líneas son la línea de datos o “Serial Data” (SDA) y la línea del reloj o “Serial Clock” (SCL). Para el intercambio de información el maestro debe conocer previamente la dirección de los esclavos. Debido a que dispone de una señal de reloj común para todos los actores, se trata de una comunicación síncrona a diferencia de UART. Este protocolo tiene como ventaja la comunicación de muchos elementos, sin embargo, a cambio aumenta la complejidad y las distancias de comunicación se limitan a una PCB.

Para iniciar la comunicación primero se debe empezar con el start bit, esto es, la línea SDA debe flanco de bajada y la SCL bit lógico alto. La siguiente información es por parte del maestro, que transmite 1 byte con la dirección del esclavo a la que se refiere (7 bits), seguido de un bit de lectura (alto) o escritura (bajo). El siguiente byte se trata de los datos a enviar, empezando por el bit más significativo. Después se envía un bit de acuse de recibo ((N)ACK) en el que la información que aporta depende de la configuración del firmware. Por último, se dispone del stop bit, que debe ser la línea SDA en flanco de subida y SCL en bit lógico alto. (Keim, All about circuits, 2015)

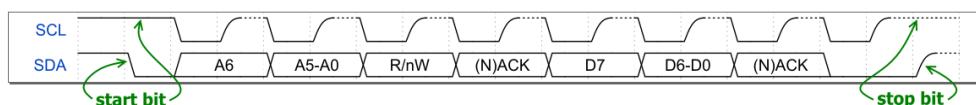


Ilustración 37 - I_C Communication³²

En el proyecto este protocolo de comunicación se ha implementado en una de las placas de sensores que mide las variables ambientales. De esta forma, con la misma placa con la que se podría tomar las medidas ambientales vistas en este capítulo, en el apartado 4.2.2.4, también se puede disponer de otro sensor para realizar medidas de otra magnitud física.

³¹ (Keim, All about circuits, 2015)

³² (Keim, All about circuits, 2015)

4.3 EQUIPO

En este apartado se detallan cuáles son los elementos que se han utilizado para llevar a cabo el proyecto. Se divide en dos subapartados, por un lado, los componentes físicos más importantes, la parte hardware y, por otro lado, los programas y lenguajes de programación utilizados para desarrollar la plataforma, la parte software. No se debe confundir esta clasificación con la realizada con anterioridad en el presente documento. Esta hace referencia a la naturaleza física de los elementos, la otra simplemente se usa para distinguir las dos partes del proyecto.

A la hora de escoger los elementos hardware y software se ha tenido en cuenta ciertos criterios. Una parte de los elementos eran de exigido uso en el proyecto por parte de la universidad Polito, ya que el laboratorio donde se ha desarrollado el proyecto estaba familiarizado con su uso. Sin embargo, otros elementos se han escogido buscando maximizar los criterios de bajo coste, flexibilidad, escalabilidad, modularidad y descentralización de los servicios.

4.3.1 Hardware

En este apartado se van a tratar los elementos hardware de la plataforma. No obstante, no se van a tener en cuenta ciertos componentes, como los componentes pasivos necesarios para ensamblar las placas, ya que su uso se detallará con posterioridad. Se detallan componentes que se consideran relevantes para la comprensión del lector.

4.3.1.1 CC1310

Se trata de uno de los MCU principales de la plataforma. Es un producto desarrollado por Texas Instruments de la familia de dispositivos CC13XX. En la Ilustración 38 se puede apreciar la placa de desarrollo creada por Texas Instruments que permite a los desarrolladores diseñar sus aplicaciones. En esta, se puede observar el MCU. Se trata del microchip que se encuentra a la derecha de pequeñas dimensiones. (Texas Instruments, Texas Instruments, 2019)

En el proyecto este MCU se encuentra en las placas de sensores y en el HUB central. Primero se ha hecho uso de las placas de desarrollo para diseñar y testear los firmwares, y posteriormente se han diseñado los PCBs específicos para la plataforma con el MCU.

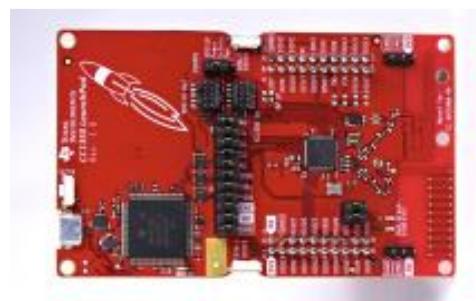


Ilustración 38 SimpleLink™ Sub-1 GHz CC1310 wireless MCU LaunchPad™ development kit (SDK)³³

Centrándose en las características, se trata de un MCU de bajo coste y bajo consumo que permite la comunicación por radio frecuencias que soporta el protocolo Sub-1 GHz.

³³ (Texas Instruments, Texas Instruments, 2019)

Esto lo convierte en un MCU muy interesante para el proyecto, ya que es justo las características que se buscan explotar.

Respecto a los módulos que lo componen, en la Ilustración 39 se puede ver el diagrama de bloques. En este se aprecia un procesador de 48 MHz ARM Cortex-M3, un transceptor de baja frecuencia y un controlador de radio dedicado con un procesador ARM Cortex-M0. De esta forma, el procesador principal puede entrar en modo de bajo consumo y ser despertado por el procesador de radio cuando sea necesario, reduciendo enormemente los consumos. También se puede apreciar que dispone de todos los protocolos de comunicación que se han visto en el capítulo 4, apartado 4.2.2.5 para la comunicación con los sensores.

Como se puede observar, este MCU cumple con todas las necesidades de la plataforma, ya que permite la comunicación Sub-1 GHz con bajos consumos, existiendo mucha documentación por parte de Texas Instruments sobre como implementar dicha tecnología. Esto hace que el desarrollo de la aplicación sea más rápido.

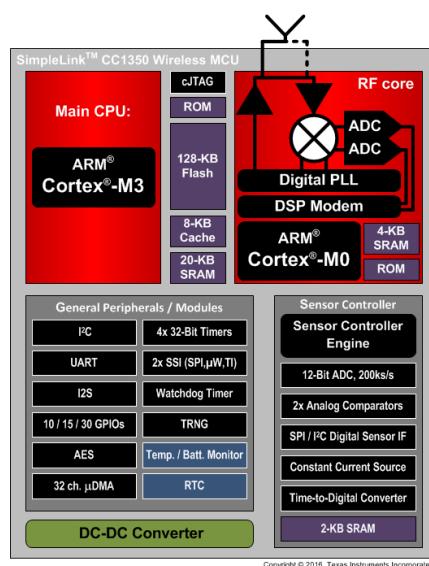


Ilustración 39 - CC13XX block diagram³⁴

Como ya se ha comentado, este MCU se encuentra en las placas de sensores y en el HUB central. En las placas de sensores se encarga de gestionar la adquisición de medidas por parte de los sensores y gestionar la comunicación Sub-1 GHz con el HUB central. En el HUB central también se dispone del MCU que funciona como colector recibiendo la información de las placas de sensores en una red tipo estrella. Desde este MCU se envía mediante UART los datos de las placas de sensores a el MCU que se detallará a continuación, que los envía a través de Internet al servidor.

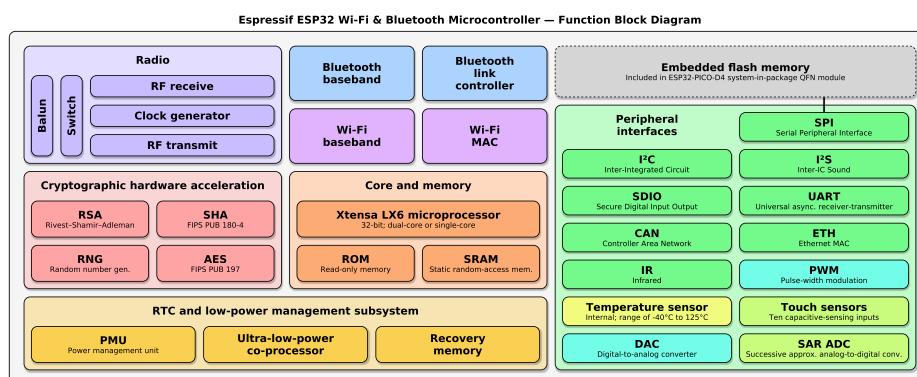
4.3.1.2 ESP 32

Como el anterior, se trata de un sistema en un chip, en inglés “System on a Chip” (SoC) MCU de bajo coste y bajo consumo. El desarrollador de este MCU es Espressif Systems, aunque existen numerosas placas de desarrollo que emplean este MCU fabricadas por otros proveedores. Se trata del sucesor del conocido ESP 8266. En la Ilustración 40 se puede observar una placa de desarrollo, estando el MCU dentro de la caja metálica. (Espressif Systems, 2019)

³⁴ (Texas Instruments, Texas Instruments, 2019)

Ilustración 40 - ESP 32 Development board³⁵

Si se atiende a las características que presenta, además de ser de bajo coste y bajo consumo, dispone de un módulo dedicado de comunicación de doble modo, que permite los protocolos de comunicación Wi-Fi y Bluetooth. Además, dispone de modos de bajo consumo con un coprocesador de ultra bajo consumo. En la Ilustración 41 se puede observar los módulos que lo componen.

Ilustración 41 - ESP 32 block diagram³⁶

Cuenta con un procesador Xtensa dual-core de 32 bits LX6, que trabaja a 160 o 240 MHz. Dispone de un módulo dedicado a la comunicación de radio frecuencias Wi-Fi y Bluetooth con un bloque dedicado a la criptografía que aumenta la seguridad del módulo. Además, dispone de los protocolos de comunicación necesarios para comunicarse con el MCU CC1310 y dispone de numerosas entradas y salidas, en caso de que se quiere aumentar las funcionalidades de la placa. (Prometec, 2019)

Otra característica relevante de este MCU es que soporta numerosos lenguajes de programación. Esto hace que el desarrollador pueda elegir el lenguaje más adecuado en función del uso requerido. Entre los lenguajes más relevantes se encuentran C, con el entorno de programación Arduino IDE y Micropython.

Con estas características, se trata de un MCU perfecto para integrar a la plataforma. Se encuentra en el HUB central junto al CC1310 que actúa como colector. De esta forma, el CC1310 le envía la información mediante UART, y el ESP 32 actúa como gateway, transfiriendo mediante Wi-Fi la información al servidor. La elección de este MCU se debe a su bajo coste, gran documentación, facilidad para conectarse a Internet y la posibilidad de programación en MicroPython para el manejo de la información.

Cabe destacar que este MCU se integra en la placa del HUB central en su forma de placa de desarrollo en vez de utilizar el MCU en sí. Esto se debe a que además de ser más sencillo, es necesario disponer de una entrada USB para la comunicación con un ordenador. Esto se debe a que se necesita configurar ciertos parámetros como puede ser la red Wi-Fi a la que conectarse.

³⁵ (Espressif Systems, 2019)

³⁶ (Espressif Systems, 2019)

4.3.1.3 Raspberry Pi 3 Model B

Se trata de un ordenador de placa única de bajo coste, creada por la compañía Raspberry Pi Foundation. Dispone de licencia de código libre, lo que permite a los desarrolladores desarrollar sistemas operativos y aplicaciones específicas para sus necesidades. El sistema operativo oficial es Raspbian, una versión de Debian, GNU/Linux. Existen diferentes modelos en los que sucesivamente se han ido ampliando las capacidades de la placa manteniendo el bajo coste. También existen modelos reducidos para pequeños proyectos.

En cuanto a las características de la placa, en la Ilustración 42 se puede observar un resumen de ellas. Como características a destacar para el proyecto, cuenta con conector Ethernet y comunicación Wi-Fi que permite recibir datos de Internet, un procesador Quad Core 1.2 GHz Broadcom con 1 GB de RAM que permite el procesado de información requerida para el proyecto. Además, cuenta con entradas USB, que permiten aumentar la capacidad de almacenamiento externa. (Raspberry Pi Foundation, 2019)

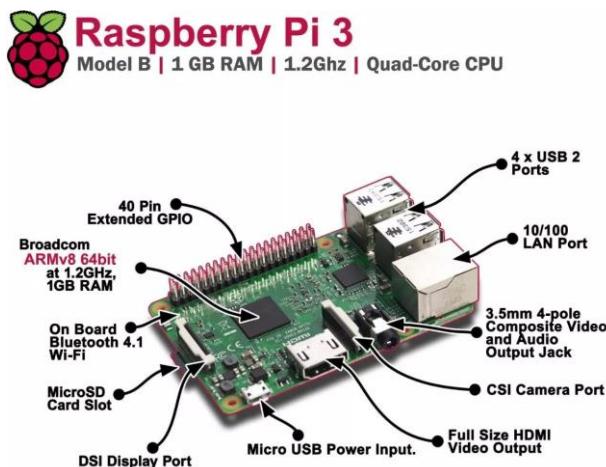


Ilustración 42 - Raspberry pi 3 specifications³⁷

El resto de las entradas y salidas no son relevantes para el proyecto, ya que el sistema operativo que se le instala no hace uso de entorno gráfico, lo que permite destinar toda la capacidad del procesador a las tareas principales del proyecto.

En cuanto al software que soporta este ordenador, soporta numerosas versiones de Linux, lo que abarata los costes por este punto, además de existir documentación por parte de la comunidad de como instalar los programas necesarios. El lenguaje de programación optimizado para la Raspberry es Python, que es ideal para el tipo de aplicación del proyecto. Es por esto, que se convierte en un ordenador ideal para el proyecto ya que, al usar softwares de código libre el coste de la plataforma se reduce. Además, permite que la plataforma se pueda migrar a un servidor de una compañía especializada en el sector.

Dadas las características descritas, este elemento se ha utilizado en el proyecto como servidor, para almacenar, visualizar y procesar la información de los sensores. Raspberry permite desarrollar la plataforma de una forma sencilla y barata, dando la posibilidad de en un futuro migrar la plataforma a un servidor especializado sin necesidad de grandes cambios.

³⁷ (Raspberry Pi Foundation, 2019)

4.3.1.4 BME 680

Se trata del sensor digital encargado de medir las variables ambientales de temperatura, humedad y presión. Además, incorpora una cuarta medida de calidad del aire, que mide la concentración de componentes orgánicos volátiles en el aire. Es un sensor desarrollado por la compañía Bosch para aplicaciones como pueden ser IoT y domótica. En la Ilustración 43 se puede observar el sensor con sus medidas.



Ilustración 43 - BME 680 sensor³⁸

Se trata de un módulo de dimensiones reducidas y de bajo consumo, por lo que es óptimo para aplicaciones con batería. Permite la comunicación mediante I²C y SPI. En la Tabla 8 se detallan las especificaciones relevantes de los parámetros a medir. (Bosch, 2017)

	Rango	Precisión	Resolución	Unidades
Temperatura	- 40 – 85	± 0.5 – 1	0.01	°C
Humedad relativa	0 – 100	± 3	0.008	%
Presión	300 - 1100	± 0.6	0.0018	hPa (mbar)
Calidad del aire	0 – 500	± 3	1	Índice IAQ

Tabla 8 - BME 680 specifications

Comparando la Tabla 8 con la Tabla 7 en la que se muestran las especificaciones de las variables ambientales para el proyecto, a excepción de algunos límites, el sensor cumple con las necesidades del proyecto. Además, es de tamaño reducido, de bajo coste, permite bajos consumos y está desarrollado por una compañía relevante y conocida, que aporta seguridad. Es por todo esto por lo que se ha escogido para el proyecto.

4.3.1.5 Transformadores

Los transformadores se encuentran en la placa de sensores encargada de la medida del consumo eléctrico. En el capítulo 4, apartado 4.2.2.4, en la Ilustración 30 aparecen integrados en el bloque encargado de transformar la señal. Como ya se explicó en ese apartado, es necesario disponer de dos transformadores que se encarguen de reducir la tensión y la corriente a valores que el MCU pueda soportar. En la Ilustración 45 y la Ilustración 44 se pueden observar ambos, el de la izquierda es el transformador de tensión y el de la izquierda el de corriente.

³⁸ (Bosch, 2017)



Ilustración 45 - Voltage transformer



Ilustración 44 - Current transformer

40

Los transformadores funcionan por el principio de inducción eléctrica. Estos transforman la tensión de entrada, aumentándola o disminuyéndola, en función de la proporción de número de vueltas que tengan las bobinas. Se componen de un bobinado primario y uno secundario. Si se prescinde de las perdidas, se obtienen las siguientes fórmulas que permiten determinar la tensión y corriente de salida (V_s e I_s) en función de la de entrada (V_p e I_p) y del número de vueltas del bobinado (N_p y N_s).

$$\frac{N_p}{N_s} = \frac{V_p}{V_s} = \frac{I_s}{I_p}$$

Ecuación 10 - Transformation relation

Para el transformador de tensión, se componen de un bobinado primario y uno secundario. Estos se deben ajustar a de forma que se obtenga la tensión de salida deseada. Si se quiere aumentar la potencia que soportan los transformadores no basta con aumentar la relación de transformación de forma que aumente la potencia. Habrá que aumentar también el grosor de los cables para que estos no se quemen. Esto hace que, a mayor potencia, más grande sean los transformadores.

En el caso del transformador de corriente, el bobinado primario lo forma el cable de la red eléctrica que atraviesa el agujero del transformador. Por lo tanto, el primer bobinado será de una vuelta. El secundario se trata de un bobinado en un toroide al que se le coloca una resistencia que genera una caída de tensión. En la Ilustración 46 se puede observar el funcionamiento interno del circuito. Utilizando la relación de transformación, al saber el número de vueltas de ambos bobinados y obteniendo la caída de tensión en la resistencia, se puede determinar la intensidad que circula por la red eléctrica. (Talema Group, Digikey, 2016)

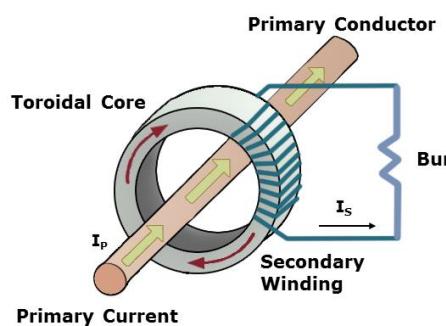


Ilustración 46 - Current transformer illustration⁴¹

³⁹ (Myrra, 2018)

⁴⁰ (Talema Group, Current Transformer Quick Reference Guide, 2018)

⁴¹ (Talema Group, Digikey, 2016)

Para el transformador de corriente se ha escogido el transformador de la compañía Talema. Se trata de transformadores de bajo coste de para ondas alternas de 50/60 Hz. La familia de transformadores se identifica por AC1XXX, siendo las X la corriente primaria que pueden soportar. Para cumplir con las condiciones que se vieron en el apartado 4.2.2.4 este transformador deberá ser superior a el número de referencia AC1020 que soporta 20 A de corriente. Para que no existan problemas, se ha sobredimensionado el transformador escogiendo el número AC1060 que soporta 60A de corriente. (Talema Group, Current Transformer Quick Reference Guide, 2018)

En cuanto al transformador de tensión, se ha tomado el transformador de la compañía Myrra. Se trata de un transformador encapsulado, que al igual que el transformador de corriente se dividen en dos series 44000 y 45000. Se busca un transformador que reduzca la tensión de 230V a valores cercanos a 5V. Además, no es necesario que soporte mucha potencia, ya que el circuito no demanda mucha corriente porque solo se busca el valor de la tensión. Con estas características se ha escogido el transformador con número de referencia 44091. (Myrra, 2018)

Este transformador dispone de dos bobinados secundarios de 6V que soporta una potencia de 1.5 VA. Un bobinado se utilizará para realizar la medida de la tensión de la red eléctrica, reduciendo los 6V a 3.3V de máxima mediante un divisor de tensión. El otro bobinado se rectifica y se transforma en corriente continua para alimentar toda la placa.

4.3.1.6 Otros componentes

En este apartado se enumeran y detallan una serie de componentes que no actúan de forma activa en el funcionamiento de la plataforma, pero que sin ellos sería imposible que se hubiera llevado a cabo. Entre ellos se encuentran:

- a) Ordenador: Para la elaboración de los algoritmos y llevar a cabo la programación de la plataforma, así como el diseño y elaboración de las PCBs.
- b) Estación soldadora de 40W: Se trata de una estación soldadora de la marca Weller T0053292699 con la que se han soldado componentes a las PCBs o elaborar circuitos en la fase de prueba.
- c) eC-Stencil-mate: stencil printer de Euro Circuits que permite alinear la PCB con el Stencil para aplicar la pasta térmica.
- d) eC-placer: Banco de trabajo para colocar componentes SMD mediante succión con base magnética.
- e) Mantis Compact microscope: Microscopio que permite colocar componentes SMD con precisión.
- f) eC-reflow-oven: Horno con función reflujo para soldar los componentes SMD a la PCB.
- g) Otros componentes electrónicos: Se trata de componentes como resistencias, condensadores, bobinas y circuitos integrados necesarios para que las PCBs funcionen correctamente.

4.3.2 Software

En este apartado se detallan los principales programas y lenguajes de programación utilizados en el proyecto. De nuevo, existen softwares que se han utilizado para instalar ciertos programas, pero en este punto solo se detallan los que han sido relevantes.

4.3.2.1 Code Composer Studio (CCS)

Se trata del IDE desarrollado por Texas Instruments para sus MCU. Este software incluye un conjunto de herramientas para el desarrollo y depurado de aplicaciones basadas en sus MCU. Está basado en el entorno Eclipse, incluyendo una optimización del lenguaje C/C++. Al estar optimizado para sus MCU, permite visualizar al completo en tiempo real las acciones que ejecuta el TI-RTOS, así como los registros y en qué consume tiempo el microprocesador. (Texas Instruments, Texas Instrument CCStudio, 2019)

Texas Instruments dispone de gran cantidad de documentación para desarrollar una aplicación usando su software. Entre la documentación se encuentran muchos ejemplos que agilizan el proceso de desarrollo y depurado de las aplicaciones.

También está disponible el CCS en versión Cloud, lo que permite editar y cargar programas desde la nube sin la necesidad de tener el programa instalado.

Este programa y derivados de la familia de Texas Instruments como Smart RF Studio se utilizan en el proyecto para el desarrollo de los firmwares de las placas de sensores y para el hub central. Para ello, se ha hecho uso de los ejemplos que Texas Instruments proporciona para crear redes tipo estrella basadas en Sub-1 GHz y modificarlos para las necesidades de este proyecto.

4.3.2.2 OrCAD

Se trata de un software dedicado a la simulación, diseño y elaboración de circuitos electrónicos y esquemas de circuitos impresos para elaborar PCBs. Está desarrollado por Cadence Design Systems y se trata de un programa de pago. OrCAD incluye varios programas, entre los que destacan OrCAD Capture, destinado al diseño de los esquemas de los circuitos y OrCAD PCB Editor que permite diseñar la disposición del PCB resultante del circuito. También dispone de otra herramienta para simular circuitos. (OrCAD, 2019)

Como se dispone de los dos programas integrados en el mismo software, el diseño del esquemático y el diseño de la PCB se vuelve más sencillo y ágil. Además, se puede disponer de visualización en 3D de la PCB resultante. También dispone de una gran base de datos de componentes, por lo que se puede encontrar el componente adecuado.

Gracias a este software se han podido diseñar y elaborar los esquemas y PCBs de las placas de sensores y hub central. La elección de este software viene dada por el laboratorio donde se desarrollar el proyecto, ya que disponían de licencias de pago, además de experiencia con el programa.

4.3.2.3 Python y Micropython

Python se trata de un lenguaje de programación interpretado, es decir no necesita ser compilado ya que se interpreta en tiempo real maximizando la eficiencia y permitiendo ser ejecutado en diferentes sistemas operativos. Además, es un lenguaje

de alto nivel y orientado a objetos que busca la legibilidad y lo práctico. (Python Software Foundation, 2019)

Se encuentra entre uno de los lenguajes de programación más usados del mundo. Esto se debe en parte a su sencillez, a ser de código libre y a tener numerosos módulos que amplían las capacidades del lenguaje. Entre ellos caben destacar las destinadas al cómputo numérico, a la gestión de datos, a la generación de gráficas y con gran auge las destinadas al Machine Learning.

En cuanto a MicroPython, se trata de una implementación eficiente de Python 3 escrita en C. Esta incluye menos librerías que el lenguaje Python y además ha sido optimizada para funcionar en MCU y en sistemas embebidos. Dispone de módulos para controlar el hardware del MCU en bajo nivel. (Yegulalp, 2014)

Gracias a las características que se han mencionado, se ha escogido estos lenguajes de programación para desarrollar los algoritmos del servidor y para gestionar la parte de Gateway en el hub central. Más concretamente, se ha utilizado MicroPython para programar el MCU ESP 32 del hub central, de manera que reciba los datos por UART del MCU CC1310 y los envíe por MQTT al servidor. En la parte del servidor, se ha utilizado Python para desarrollar los algoritmos que gestionan los datos de forma que se puedan almacenar y procesar correctamente.

Se podrían haber utilizado otros lenguajes o herramientas para estas tareas. Sin embargo, debido a la cantidad de documentación, coste nulo de implementación, facilidad del lenguaje para la gestión de datos y conocimientos previos sobre el lenguaje se convierten en los elegidos.

Entre los principales módulos utilizados para el desarrollo de la plataforma se encuentran:

- Numpy: permite el computo numérico de un número elevado de datos. Se trata del equivalente gratuito de Matlab.
- Time/datetime: librería destinada al control de fechas y tiempos.
- Matplotlib: permite generar gráficos para visualizar los datos.
- Paho-mqtt: implementa el protocolo MQTT para los clientes de la librería Eclipse Paho MQTT. De esta forma permite crear clientes que se conectan a un bróker.
- Mosquitto: implementa un bróker MQTT de código libre. También pertenece a Eclipse Foundation.
- Json/ujson: gestor de mensajes en formato JSON. Ujson es la implementación para MicroPython.
- WxPython: permite generar una interfaz gráfica plataforma – cliente.
- Requests: librería que gestiona las comunicaciones HTTP
- Cherrypy: permite crear aplicaciones web.
- Esptool: permite comunicarse y flashear código en el MCU ESP 32.

- Telebot: Permite la comunicación con bots de Telegram
- Influxdb: módulo que implementa las comunicaciones con la base de datos InfluxDB
- Pandas: módulo para el manejo de estructuras de datos, series temporales y estadísticas.

4.3.2.4 Raspbian Stretch Lite

Raspbian se trata del sistema operativo oficial de Raspberry Pi. Por lo tanto, es una distribución del sistema operativo GNU/Linux, ya que es de código libre, que está basada en Debian. Aproximadamente cada dos años Raspberry Pi Foundation saca nuevas versiones de este sistema operativo. Raspbian Stretch es una de estas versiones, siendo la más actual Raspbian Buster. Lo único que implementa nuevo son optimizaciones de código. La palabra “Lite” responde a que es la versión del sistema operativo Raspbian Stretch más ligera, en la que no incluye interfaz gráfica ni programas recomendados. (Long, Raspbian Stretch has arrived for Raspberry Pi, 2017)

Como se trata de la versión que proporciona el desarrollador, se encuentra optimizado para el uso en la Raspberry Pi 3 del proyecto. Es por esto por lo que se ha elegido Raspbian. Escoger la versión Strech se debe a que cuando se comenzó el proyecto, era la versión más reciente. Sin embargo, la nueva versión no incluye grandes cambios que tengan relevancia con las características del proyecto. Los cambios se enfocan más en cambios estéticos y para añadir funciones del nuevo modelo de Raspberry Pi. La elección de la versión Lite se debe a que, para las necesidades de servidor del proyecto, no es necesario disponer de una GUI. De esta forma se ahorra en memoria, consumo y eficiencia. (Long, Buster – the new version of Raspbian, 2019)

4.3.2.5 InfluxDB

Se trata de una base de datos orientada al almacenamiento de datos estructurados por series temporales. Es una base de datos de código abierto desarrollada por InfluxData y escrita en Go. Esta optimizada para el almacenamiento y extracción de series temporales de forma rápida y segura. Por lo tanto, se convierte en una base de datos óptima para aplicaciones como IoT, monitoreo de parámetros, y analíticas en tiempo real. (InfluxData, InfluxDB documentation, 2019)

InfluxDB proporciona un lenguaje parecido a SQL, en el que se realizan peticiones mediante HTTP, TCP y UDP para almacenar y extraer datos. Además, la compañía proporciona un paquete llamado TICK stack, que alberga cuatro softwares diferentes. Con ellos se puede gestionar los datos desde que se generan en el sensor, se almacenan, se visualizan y analizan. El uso de estas herramientas es sencillo, ya que cuentan con una interfaz gráfica intuitiva. Esto permite minimizar el tiempo de desarrollo de aplicaciones al tener integrado todo el proceso. (InfluxData, TICK stack: InfluxDB, 2019)

Como en el presente proyecto se trabaja con datos en los que la hora en la que se generan es crucial, esta base de datos encaja perfectamente a las necesidades. Sin embargo, no se hace uso del paquete TICK stack. El motivo es que se busca desarrollar una plataforma que se adapte perfectamente a las necesidades y para ello debe ser flexible, escalable y que funcione de forma automática. Por lo tanto, se ha decidido desarrollar los algoritmos que gestionan los datos hasta que se almacenan y su posterior extracción mediante Python.

4.3.2.6 Grafana

Se trata de un software de código abierto dedicado a la visualización de datos. Permite generar diferentes tipos de gráficos, sobre todo gráficos de series temporales, estando adaptado para soportar InfluxDB de manera sencilla. También se encuentra programada en lenguaje Go y permite realizar peticiones mediante HTTP. Entre sus funciones principales permite visualizar y entender los datos gracias a la cantidad de gráficos que dispone, permite crear alertas si los datos sobrepasan algún valor, soporta las bases de datos más relevantes y es de código libre, lo que permite instalarlo y desarrollar plugins a los usuarios. (Grafana Labs, 2019)

Gracias a esta serie de características, Grafana se convierte en la elección para la visualización y monitorización de los datos obtenidos por los sensores.

4.3.2.7 Telegram

Es una aplicación gratuita de mensajería instantánea y VOIP, que permite el envío de archivos y mensajes entre sus usuarios. Se trata de una aplicación multiplataforma que funciona en los sistemas operativos más relevantes del mercado. No es una aplicación de código libre completamente. La parte del cliente es de código abierto, lo que permite a desarrolladores crear clientes externos que interactúen con la aplicación.

La aplicación dispone de muchas funcionalidades, pero la más relevante para el proyecto es la posibilidad de desarrollar chatbots. Esto es, permite crear servicios automáticos que obedecen comandos. Estos comandos pueden ser enviados desde el exterior a través de peticiones HTTP. Gracias a esto, en el proyecto se pueden crear notificaciones ante ciertos eventos provocados por los datos de los sensores. Así pues, se puede notificar al usuario si algo ha ocurrido en la plataforma a través de la aplicación móvil.

4.3.2.8 Otros softwares

Además de los softwares y lenguajes de programación ya descritos, para el desarrollo del proyecto han sido necesarios otros programas. Sin embargo, no se consideran tan importantes como los anteriores, ya que existen otras alternativas que los desarrolladores pueden utilizar. A continuación, se presentan algunos de ellos:

- a) Jupyter notebook: Es una aplicación web de código abierto que permite el procesamiento de datos en tiempo real en diferentes lenguajes de programación. En este caso se ha utilizado para desarrollar los algoritmos en Python.
- b) Notepad++: Editor de texto para Windows de código libre que soporta varios lenguajes de programación, entre ellos Python y MicroPython. Se ha utilizado junto con el anterior para desarrollar los algoritmos del servidor y del Gateway.
- c) Win32 disk instaler: Herramienta para Windows que permite escribir imágenes en memorias USB o tarjetas SD. Se ha utilizado para quemar la imagen de Raspbian Stretch Lite en la tarjeta SD.
- d) ESPlorer: IDE para desarrollar y subir los algoritmos de MicroPython al MCU ESP 32.
- e) uPyCraft: Herramienta que tiene el mismo uso que la anterior.

5. DESARROLLO DEL SISTEMA DE MONITOREO DE LA EFICIENCIA ENERGETICA

Como ya se adelantaba en el capítulo 4, apartado 4.1, el proyecto consta de dos partes muy diferenciadas. Por un lado, se tiene el diseño, desarrollo y ensamblado de las placas de sensores y hub central y sus correspondientes firmwares. Por el otro lado, se tiene el desarrollo y elaboración de los algoritmos necesarios para la transmisión, almacenado, visualización y procesado de los datos generados por las placas de sensores.

La primera parte se centra más en el desarrollo de elementos físicos, la parte Hardware, mientras que la segunda se centra más en el desarrollo de la nube, la parte Software. Si se clasifican según a que parte de la arquitectura de una plataforma IoT pertenecen, siguiendo el modelo visto en la Ilustración 10, la parte Hardware pertenece a la Edge layer y parte de la Fog layer, mientras que la parte Software pertenece a la Cloud layer y la otra parte de la Fog layer. La Fog layer esta compartida entre las dos partes, ya que la primera se encarga del diseño y desarrollo de la PCB, mientras que la segunda del desarrollo de los algoritmos necesarios para enviar el mensaje a la nube.

Por lo tanto, este capítulo se va a dividir en las dos partes que conforman el proyecto. Se comienza con la parte en la que se generan los datos, la parte Hardware, hasta llegar a su almacenamiento en el servidor, la parte Software. Se ha escogido este procedimiento para así comprender el flujo real de información que ocurre en la plataforma. Sin embargo, a la hora de desarrollarla no es la secuencia de actividades que se ha llevado a cabo para completar el proyecto.

Cabe recalcar que todas las ilustraciones de este capítulo son de elaboración propia. Se ha decidido no utilizar el pie de página para ahorrar espacio.

5.1 PARTE HARDWARE

En este apartado se van a detallar los componentes y el proceso necesario para obtener los diferentes elementos del sistema. Para ello, se va a dividir el apartado en los elementos que lo componen. En la Ilustración 47 se puede observar estos elementos. Se trata de dos nodos conectados a un colector central siguiendo una topología de red tipo estrella.

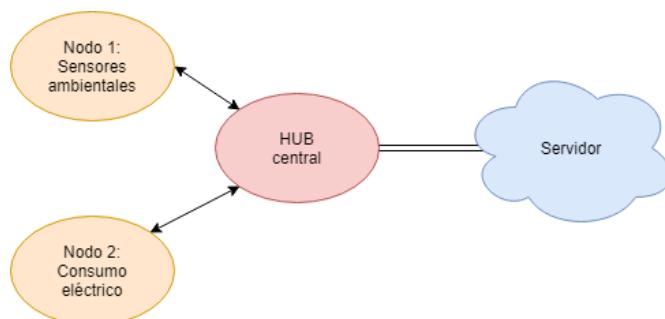


Ilustración 47 – Simple platform diagram

Estos nodos son sensores que toman mediciones del entorno. El primero es un sensor ambiental que realiza medidas de temperatura, presión, humedad y calidad del aire. El segundo es un sensor que realiza medidas de tensión y corriente para calcular la potencia consumida. El nodo central o hub central, se encarga de recibir los datos de los nodos y transmitir esa información al servidor donde se pueden monitorizar.

5.1.1 Aspectos comunes de las placas

5.1.1.1 Hardware

Las tres placas a desarrollar en el proyecto están compuestas del MCU CC1310 de Texas Instruments. Además, como las tres se comunican a través de Sub-1 GHz necesitan la misma antena. Para su diseño se ha seguido la hoja de diseño del fabricante. (Wallace & Texas Instruments, 2013)

En cuanto al diseño de las PCBs y del firmware, se ha partido también de los diseños y ejemplos ofrecidos por el fabricante para que los desarrolladores puedan llevar a cabo sus aplicaciones. Por ello, se han incluido parte de los elementos que se encuentran en las placas de desarrollo LaunchPad de Texas, como son los botones y los LEDs, además de utilizar su esquemático como referencia. (Texas Instruments)

Para la comunicación física con otros dispositivos se ha incluido un puerto JTAG para la programación, un puerto UART con TX y RX para depurar el código y conectores para suministrar energía a las placas.

Además de los conectores anteriores, se dispone de dos pines adicionales. Uno se utiliza para resetear la placa, es decir, cuando los pines se cortocircuitan la placa se reinicia. El otro se sitúa en la parte de alimentación de la placa para poder conocer cuál es el consumo de esta. Debido a limitaciones temporales, no se ha llevado a cabo pruebas en este aspecto como se refleja en el capítulo de Líneas futuras.

5.1.1.2 Firmware

Para el desarrollo del firmware se ha partido de los ejemplos proporcionados por Texas Instruments. En concreto de los referentes a “TI 15.4-Stack”, el ejemplo “sensor” y “collector”. Para acceder al firmware original, se puede acceder a través de CCS o del Explorador de recursos (“Resource Explorer”) de Texas Instrument en Internet. (Texas Instruments, s.f.)

Dentro del Explorador de Recursos se accede al firmware a través de:

“Software/SimpleLink CC113x0 SDK – 3.80.00.23/Examples/Development Tools/CC1310 LaunchPad/TI 15.4-Stack”.

Frente al firmware original se han realizado una serie de modificaciones que se van a analizar a continuación:

- *Variables del mensaje*

Para la aplicación a desarrollar es necesario modificar el número de variables que se transmiten entre las placas ya que las del firmware original se quedan cortas. Para ello, es necesario añadir nuevas variables de tipo “`uint16_t`” que son las siguientes:

- “`Id0`”, “`id1`” ..., “`id3`”: Para identificar cada placa es necesario el uso de un identificador único. Se usa el identificador MAC que se compone de 64 bits, por lo que se requiere de 4 variables de 16 bits para almacenarlo.

- “type”: Almacena el identificador del tipo de sensor que se define en el firmware como “SENSOR_TYPE”
- “battery”: Almacena el nivel de batería de la placa si es que posee una batería.
- “f0”, “f1”, ..., “f9”: Se trata de las variables que almacenan las mediciones de los sensores y el resultado de procesar esos datos. Se trata de los valores propiamente dichos por lo que a partir de ahora se referirá a ellos como campos. En caso de que una variable a medir sea mayor de 16 bits y por lo tanto no entre en un campo, se divide la variable en dos campos y en la placa del colector se vuelve a juntar. Esto da lugar a un nuevo firmware que se incorpora en las placas y se llama “new_sensor_template”.

Para que la variable “type” pueda ser enviada, es necesario definir en todos los archivos “Application/subg/config.h” la definición de “SENSOR_TYPE”. Esto permite conocer al colector de quien pertenece el mensaje y por tanto como ha de tratar los datos que se mandan en los campos.

```
/*! Power meter sensor uses (1), environmental sensor uses (2) */
#define DEF_SENSOR_TYPE (1)
```

Código 1 - Firmware variable modifications

- *Firmware de las placas de sensores*

Para las dos placas de sensores es necesario realizar una serie de modificaciones en el código. La mayoría de estas están relacionadas con las variables a enviar y con la funcionalidad de los botones en la placa. A continuación, se muestra y explica cómo funcionan los códigos en los archivos principales.

- Application/main.c

El firmware se compone de dos tareas principales “appTask” y “appSensorTask”. La primera se encarga de las comunicaciones y posee mayor prioridad. La segunda se encarga de realizar las mediciones de los sensores y es invocada por la primera.

Para ello, se dispone de un semáforo “semHandle” que bloquea la segunda tarea hasta que la primera lo desbloquea. En caso de que ambas estén activas, la segunda tendrá que esperar ya que tiene menor prioridad.

- Application/sensor.c

Este archivo contiene diferentes funciones. En lo referente a la función “Sensor_init()”, se puede apreciar al final de esta como se asigna a las variables “id0”, “id1”, ..., “id3” el valor de la dirección MAC de la placa. Además, también se le asigna el valor de la variable “type” para reconocer que tipo de sensor es. Las siguientes líneas de código muestran su funcionamiento.

```
// MAC address is read from the registers using the following
functions
uint32_t macAddrlsb_32 = HWREG(FCFG1_BASE + FCFG1_O_MAC_15_4_0);
uint32_t macAddrmsb_32 = HWREG(FCFG1_BASE + FCFG1_O_MAC_15_4_0);

//a temporary array to store the splitted MAC address is created
uint16_t macAddr_16[4];

//the address is splitted into 4 uint16_t variables
macAddr_16[0] = (macAddrlsb_32 & 0x0000ffff); //lowest part
```

```

macAddr_16[1] = (macAddrlsb_32 & 0xffff0000) >> 16;
macAddr_16[2] = (macAddrmsb_32 & 0x0000ffff);
macAddr_16[3] = (macAddrmsb_32 & 0xffff0000) >> 16; //highest part

//finally the address is copied to the message variables
id0Sensor.id0RawData = macAddr_16[0];
id1Sensor.id1RawData = macAddr_16[1];
id2Sensor.id2RawData = macAddr_16[2];
id3Sensor.id3RawData = macAddr_16[3];

// type
typeSensor.typeRawData = DEF_SENSOR_TYPE;

```

Código 2 - Application/Sensor.c firmware. MAC address

La siguiente función es “readSensors()” que se encarga de realizar la medición de los sensores. Para ello dispone de un semáforo “Semaphore_post(semHandle)” que desbloquea la tarea “sensorTask” como ya se vio en el firmware anterior.

Una vez se desbloquea el semáforo se realiza la medición de los valores y se transforman a un formato que permita enviarse en el mensaje. Estas transformaciones son necesarias en caso de que los valores sean de tipo “float” o que sean valores negativos.

En caso de ser valores negativos, es necesario añadir un “offset” a la variable para transformarla en positiva. En caso de ser decimales, se multiplica por 10^n el valor de la variable en función del número de decimales a conseguir. Luego, en el colector estas transformaciones serán revertidas para obtener los valores iniciales.

Como ya se ve en el punto de “Variables del mensaje”, el tipo de las variables es “uint16_t”. Sin embargo, para algunas mediciones es necesario utilizar “uint32_t”. Para solucionar el problema se divide el valor en dos campos de tipo “uint16_t” que posteriormente serán unidos de nuevo en el colector obteniendo el valor original. A continuación, se puede observar un fragmento del firmware de la placa de sensores ambientales en la que se muestra lo explicado anteriormente.

```

// NOTE: all the variables inside databme are "double"

//temperature
//2 decimal places, offset = 40
f0Sensor.f0RawData = (databme.temperature + 40)*100;

//pressure (2 fields used)
//2 decimal, offset = 0
uint32_t pressure_32 = (uint32_t)(databme.pressure*100)
f1Sensor.f1RawData = (pressure_32 & 0x0000ffff); //lowest
part
f2Sensor.f2RawData = (pressure_32 & 0xffff0000) >> 16; //highest
part

//humidity
//3 decimal, offset = 0
f3Sensor.f3RawData = (databme.humidity)*1000;

//gas(2 fields used)
//0 decimal, offset = 0
uint32_t gas_32 = (uint32_t)databme.gas_resistance
f4Sensor.f4RawData = (gas_32 & 0x0000ffff); //lowest part
f5Sensor.f5RawData = (gas_32 & 0xffff0000) >> 16; //highest part

```

Código 3 - Application/sensor.c firmware. Message transformations

Para terminar, existe una función de retorno “Jd1lcStateChangeCb()” que se ejecuta cada vez que las placas de sensores cambian de estado. Su función es activar o desactivar los LEDs de las placas para que el usuario sepa en qué estado se encuentra. El siguiente código muestra cómo funciona esta función.

```

switch(state){
    case Jd1lc_states_orphan: // state 5
    case Jd1lc_states_accessDenied: // state 6
        Board_Led_control(board_led_type_LED2, board_led_state_ON); //red led
        on
        Board_Led_control(board_led_type_LED1, board_led_state_OFF); //green
        led off
        break;

    case Jd1lc_states_joined: // state 3
    case Jd1lc_states_rejoined: // state 4
        Board_Led_control(board_led_type_LED2, board_led_state_OFF); //red led
        off
        Board_Led_control(board_led_type_LED1, board_led_state_ON); //green led
        on
        break;

    case Jd1lc_states_initWaiting: // state 0
    case Jd1lc_states_joining: //state 1
    case Jd1lc_states_initRestoring: //state 2
    default:
        Board_Led_control(board_led_type_LED2, board_led_state_OFF); //red led
        off
        Board_Led_control(board_led_type_LED1, board_led_state_OFF); //green
        led off
        break;
}

```

Código 4 - Application/sensor.c firmware. Sensor states

- Application/csf.c

Este archivo se encarga del control de los botones en la placa. Ambos botones realizan la misma tarea relacionada con la conexión y desconexión con el colector. Cuando se pulsa durante 2 o 3 segundos algún botón, la conexión con el colector se olvida y se procede a buscar otro colector. Acerca de los estados del sensor, se verán con mayor detalle en el apartado 5.1.1.4 de este capítulo.

- *Firmware del colector*

Al igual que con las placas de sensores, en el colector también hay que realizar una serie de cambios al firmware para transformar los datos y para que los botones funcionen correctamente.

- Application/cllc.c

En este archivo las modificaciones se realizan en la función “assocIndCb()”. Estas modificaciones permiten que los sensores se reconecten a la red en caso de que pierdan momentáneamente la conexión con el colector. Para ello, se siguió los pasos que se aportan en el foro de discusiones de Texas Instruments. (Texas Instruments, 2019)

- Application/csf.c

Este archivo controla el funcionamiento de los botones y de las transformaciones de los mensajes.

La función “Csf_processEvents()” se modifica para que al pulsar cualquiera de los botones, la red se alterne entre abierta o cerrada.

La función “Csf_deviceSensorDataUpdate()” se ejecuta cuando llega un nuevo mensaje al colector. Lo primero comprueba el identificador del sensor “type-Sensor” para saber que transformaciones se han de realizar. En caso de que el mensaje no sea nulo, se procesa y muestra por UART.

5.1.1.3 Configuración de la comunicación

Para que las placas se puedan comunicar entre ellas, es necesario establecer la misma configuración de la red de Sub-1 GHz. A continuación, se muestra que parámetros se deben establecer para que se lleve a cabo. Entre ellos destaca deshabilitar la característica de “frequency hopping” debido a problemas de inestabilidad en las conexiones. Esto se lleva a cabo en el archivo “Application/subg/features.h”.

```
/*! If defined, builds the image with all the modes of operation
   (frequency hopping, beacon mode and non beacon mode) */
#undef FEATURE_ALL_MODES
/*! If defined, builds the image with the frequency mode of operation
*/
#define FEATURE_FREQ_HOP_MODE
/*! If defined, builds the image with beacon mode of operation */
#define FEATURE_BEACON_MODE
/*! If defined, builds the image with non beacon mode of operation */
#undef FEATURE_NON_BEACON_MODE
/*! Builds the image with the full function device */
#define FEATURE_FULL_FUNCTION_DEVICE
/*! If defined, builds the image with the mac layer security turned on
*/
#define FEATURE_MAC_SECURITY
```

Código 5 - Firmware communication configuration

Además, en el archivo “Application/subg/config.h” es necesario modificar el “PHY” para que sea el de la banda europea como se ve en la Tabla 3. También existe un modo de largo alcance que se habilita modificando el “PHY”, sin embargo, no se ha llevado a cabo pruebas debido a que la tasa de transmisión de datos es muy baja, lo que hace que el tiempo entre mensajes aumente, incrementando los tiempos de pruebas. Por último, se ha aumentado la potencia de transmisión de la señal para tener mayores alcances.

```
/*! Setting for Phy ID */
#define CONFIG_PHY_ID          (APIMAC_STD_ETSI_863_PHY_3)
...
*/
Value for Transmit Power in dBm
For US and ETSI band, Default value is 10, allowed values are
-10, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 and 14dBm.
*/
#define CONFIG_TRANSMIT_POWER    14
```

Código 6 – Firmware PHY configuration

5.1.1.4 Estados de la comunicación

En el momento que se les suministra energía a las placas, estas intentan establecer conexión entre ellas. En ese proceso pasan por diferentes estados que pueden ser conocidos por el usuario en caso de que esté conectado por UART a la placa. Sin embargo, para agilizar el proceso de desarrollo y prueba y por razones prácticas se dispone de dos LEDs que muestran al usuario el estado en el que se encuentran las placas. A continuación, se detallan los posibles estados para las placas de sensores, que son comunes, y el colector.

- *Placa del colector*

La primera vez que se enciende el colector crea automáticamente una nueva red y enciende el LED verde sin parpadear. En caso de que no sea su primera vez, vuelve a cargar su red antigua, dejando que los sensores que estaban conectados en su momento se vuelvan a reconectar.

Los botones del colector permiten a abrir y cerrar la red. Por defecto, cuando se enciende el colector la red está cerrada. Cuando el colector tiene la red abierta, los sensores que se encuentren en el estado de escaneo de redes pueden establecer la conexión con el colector. Cuando el colector está abierto, el LED verde parpadea.

El LED rojo se encenderá y apagará cada vez que se reciba un mensaje o se envíe a través de UART al ESP 32.

- *Placas de sensores*

A diferencia del colector, las placas de sensores disponen de más estados, ya que se debe reflejar si estas están conectadas con el colector. En la Tabla 9 se muestran los posibles estados junto con el estado de los LEDs que lo reflejan.

Estado	Descripción	LED verde	LED rojo	
0	initWaiting	Encendido, esperando a conectarse a la red por el usuario	APAGADO	APAGADO
1	Joining	Buscando redes abiertas para conectarse	APAGADO	APAGADO
2	initRestoring	Encendido y restaurando la red previa	APAGADO	APAGADO
3	Joined	Dispositivo conectado a la red	ENCENDIDO	APAGADO
4	Rejoined	Dispositivo se ha reconnectado a la red	ENCENDIDO	APAGADO
5	Orphan	Dispositivo huerto	APAGADO	ENCENDIDO
6	accessDenied	Conexión denegada	APAGADO	ENCENDIDO

Tabla 9 - Sensor board states

El estado “0” no es de uso para el usuario, ya que el dispositivo automáticamente se ve forzado al estado “1” o “2” nada más se enciende el dispositivo. Por lo tanto, el dispositivo se encuentra en el estado “1” nada más encenderse en el caso de que no tenga almacenado en su memoria una conexión antigua. Esto es, la primera vez que se conecta o si se ha forzado al dispositivo a que olvide la última red. El estado “2” se ejecuta en caso de que sí que exista en la memoria una red previa, por lo que trata de reconectarse.

Una vez el dispositivo encuentra una red, ya sea nueva (estado “1”) o antigua (estado “2”), este intenta conectarse. Cuando lo consigue pasa al estado “3” si procedía del estado “1” o al estado “4” en caso de conocer ya la red y proceder del estado “2”. En caso de que no logre conectarse a la red, este pasará a el estado “5” o “6”.

El estado “5” se logra cuando el dispositivo pierde la conexión con el colector. Esto puede ocurrir cuando la señal de la red es muy débil o si el colector ha sido desconectado de la red. El estado “6” es un caso especial del estado anterior que se da cuando el dispositivo no puede conectarse con el colector al haber alcanzado este el número máximo de dispositivos conectados. Este número es 50 dispositivos conectados, aunque se puede modificar en el archivo “Application/subg/config.h”, modificando el valor de la constante “CONFIG_MAX_DEVICES”.

En cuanto a los botones que disponen las placas, estos permiten que el dispositivo olvide la red anterior a la que estaban conectados e intenten conectarse a otra que haya en ese momento abierta. Esto se consigue al mantener pulsado durante unos segundos alguno de los botones, lo que le hace pasar al estado “1”.

5.1.2 Placa de sensores ambientales

5.1.2.1 Introducción

Esta placa se encarga de la obtención de los parámetros relacionados con el ambiente en lo que se refiere a climatología y calidad del aire. Para ello dispone del sensor BME680 el cual se analiza en el capítulo 4, apartado 4.3.1.4. Gracias a este sensor, se pueden realizar medidas de la temperatura interior y exterior, de la humedad relativa, de la presión y de la calidad del aire. Esto permite conocer parámetros relevantes a la hora de determinar la eficiencia energética en una vivienda como se apuntaba en el capítulo 4, apartado 4.2.2.4.

Como el resto de las placas desarrolladas, el MCU es el CC1310 el cual se comunica a través de I₂C con el sensor BME680. Una vez obtiene las medidas de los parámetros ambientales, los envía a través de Sub-1 GHz al colector.

Además de la conexión con el sensor BME680, se dispone de dos entradas analógicas/digitales y un puerto I²C extra en caso de que se quiera añadir más sensores a la placa o no se disponga del sensor BME680. Esto se ha realizado con el objetivo de aportar mayor flexibilidad y modularidad a la placa. También permite aprovechar placas que ya se hayan producido y no sea necesario más sensores ambientales, lo que ahora costes.

Para alimentar la placa se dispone de una entrada USB a 5V y de un adaptador para conectar una batería de litio de una celda. De esta manera el sensor se puede posicionar en cualquier lado de la vivienda o en el exterior.

5.1.2.2 Diagrama de bloques

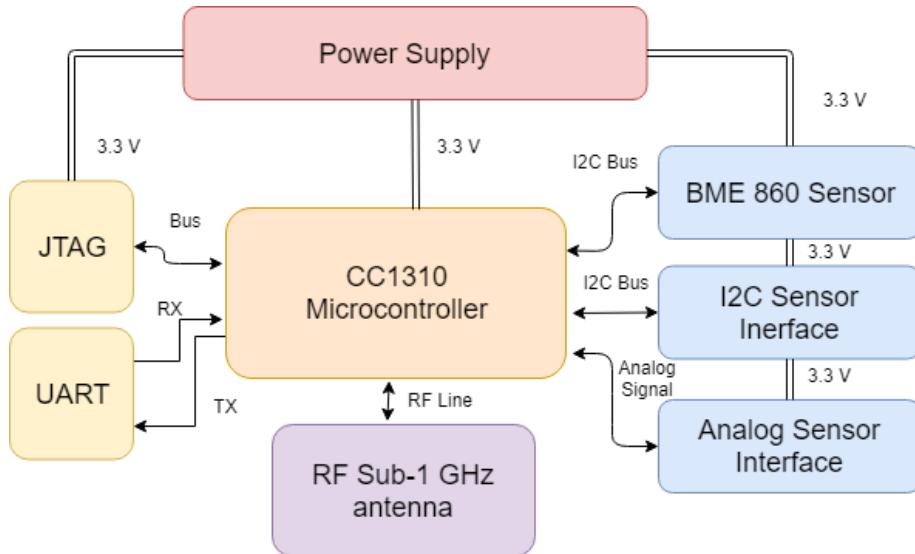


Ilustración 48 - Environmental sensor block diagram

En la Ilustración 48 se muestra el diagrama de bloques que forma la placa de sensores ambientales. A continuación, se aporta una pequeña descripción de cada uno de ellos.

- Power supply: Proporciona la energía a toda la placa. Puede ser una conexión por USB de 5V o una batería de litio. La salida de este será una tensión regulada de 3.3V
- RF Sub-1 GHz antenna: Se trata de la antena de Sub-1GHz que permite la conexión con el colector.
- JTAG: Se trata del puerto utilizado para programar el firmware en el MCU. También sirve para depurar código.
- UART: Puerto utilizado para depurar código y mostrar en otro ordenador el estado del MCU.
- BME 860 Sensor: Representa la interfaz del sensor BME 860 utilizado para medir los parámetros ambientales.
- I2C Sensor Interface: Representa el puerto I²C adicional para comunicarse con otros sensores en caso de que se incluyan.
- Analog Sensor Interface: Se trata de los puertos adicionales analógicos/digitales en caso de que se quiera incorporar nuevos sensores.

5.1.2.3 Esquemáticos

Los esquemas electrónicos se dividen en varias secciones que se van a mostrar a continuación. Todos se han basado en los esquemáticos ofrecidos por Texas Instruments sobre la placa de desarrollo LaunchPad CC1310 (Texas Instruments).

- *Microcontrolador CC1310*

Se trata del núcleo central de la placa. Se trata del MCU CC1310 de Texas Instruments que soporta la comunicación por radiofrecuencias en la banda de Sub-1 GHz. Dispone de dos tensiones con las marcas “ V_{DDS} ” y “ V_{DDR} ”. El primero se suministra desde la fuente de alimentación con una tensión de 3.3V. El segundo lo proporciona el propio MCU a través de un conversor DC/DC interno a la tensión de 1.68V. Este último se utiliza para módulos internos del MCU como el módulo de radiofrecuencias y no puede ser utilizado para otros propósitos. El regulador DC/DC proporciona su tensión por el pin “DCDC_SW”.

Se dispone del condensador “ C_{13} ” para desacoplar el núcleo del CC1310 al que se le suministra una tensión regulada de 1.28V desde “ V_{DDR} ”.

Para el reloj del sistema se dispone de dos osciladores de cristal “ Y_1 ” y “ Y_2 ”. “ Y_2 ” proporciona la referencia de reloj para el núcleo. “ Y_1 ” proporciona la señal de reloj para ciertos módulos internos.

Los pines 1, 2 y 3 están reservados para la comunicación con la antena que se detalla en el apartado Antena.

Para la comunicación JTAG se utilizan los pines con las etiquetas 24, 25, 26, 27. Para realizar un reseteo del circuito se utiliza el pin “CC1310_RESET”. Para la comunicación UART se reservan los pines “DIO2_RXD” y “DIO3_TXD”. Todos ellos se explican con mayor detalle en el apartado de Puertos.

Los pines “DIO7_GLED” y “DIO8_RLED” se trata de los pines reservados a los LEDs que muestran el estado del MCU. Los pines “DIO13_BTN1” y “DIO14_BTN2” se encargan de los botones.

En cuanto a los puertos utilizados para comunicarse con los sensores, se dispone de “DIO4_SCL” y “DIO5_SDA” para la comunicación I₂C y los pines “DIO23_A0” y “DIO24_A1” para la comunicación analógico/digital. En el apartado Puertos se muestran con mayor detalle.

Por último, se dispone de “DIO10_CHG_BAT” y “DIO11_LOW_BAT” que se utilizan para la indicar si se está cargando la batería y el nivel bajo de la misma. Se muestra con mayor detalle en el apartado de Fuente de alimentación.

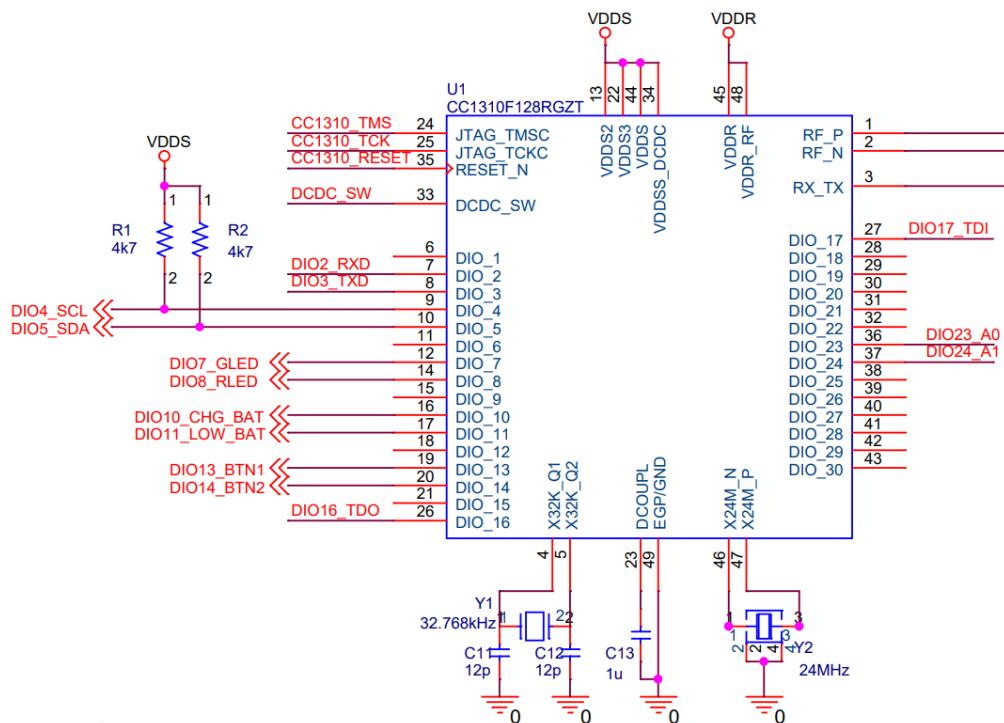


Ilustración 49 - Environmental board. CC1310 schematics

- **Antena RF**

La antena está diseñada siguiendo el manual de referencia proporcionado por Texas Instruments. Se trata de una antena impresa en la propia PCB por ambas caras de la misma. El circuito se trata de un filtro con la antena en el PCB. (Wallace & Texas Instruments, 2013)

Los condensadores "C₇" y "C₈" se montan para hacer que la frecuencia en la que emite la antena sea la correcta. Estos condensadores no se montaron ya que, tras probar la comunicación de las placas, esta era correcta. Además, debido a problemas con el pedido de los componentes, las inductancias "L₃" y "L₄" no se montaron. Sin embargo, al ser valores muy pequeños, se prescindió de ellas, en todas las placas y los resultados fueron los esperados.

En cuanto a las líneas que provienen del MCU, el pin "RF_P" está conectado a los componentes "C₁", "L₁" y "L₂", el pin "RF_N" a los componentes "L₂", "L₅" y "C₉" y el pin "RX/TX" a los componentes "L₅" y "C₁₀".

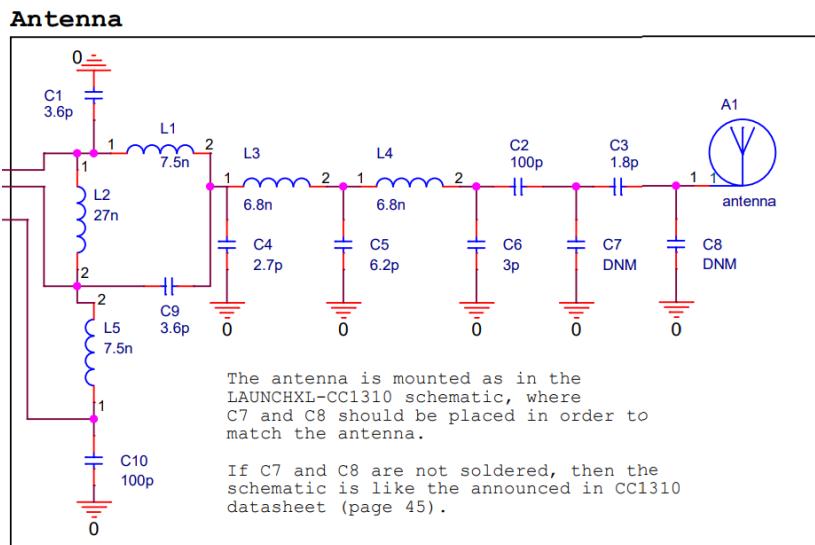


Ilustración 50 - Environmental board. Antenna schematics

- *Bypass*

Se trata del apartado que se encarga de filtrar las tensiones de la fuente de alimentación y los condensadores de bypass del MCU para tener un buen funcionamiento.

El circuito de la izquierda es el filtro de la fuente de alimentación de 3.3V a “ V_{DDS} ”. Se trata de una serie de condensadores que deben de colocarse cerca del pin que aparece y de una inductancia de ferrita que elimina los ruidos a altas frecuencias.

El circuito de la derecha es el filtro del conversor DC/DC interno del MCU que proporciona los 1.65V. De nuevo se trata de condensadores e inductancias que deben de posicionarse cerca de sus pines.

Microcontroller supply

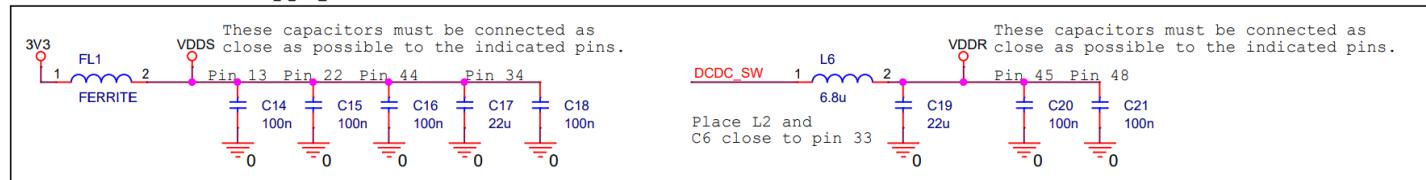
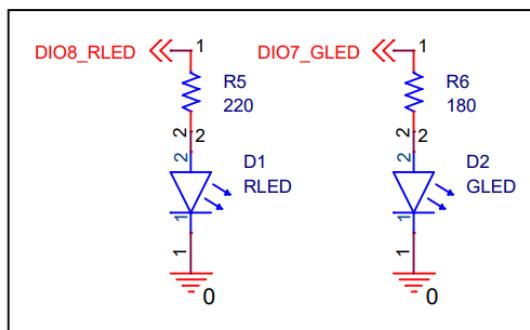


Ilustración 51 - Environmental board. Bypass schematics

- *LEDs RF y botones*

Se trata de circuitos simples para controlar los LEDs y los botones. Estos sirven para mostrar y modificar el estado en el que se encuentra el MCU como ya se explica en el capítulo 5, apartado 5.1.1.4.

RF LED indicators



Buttons

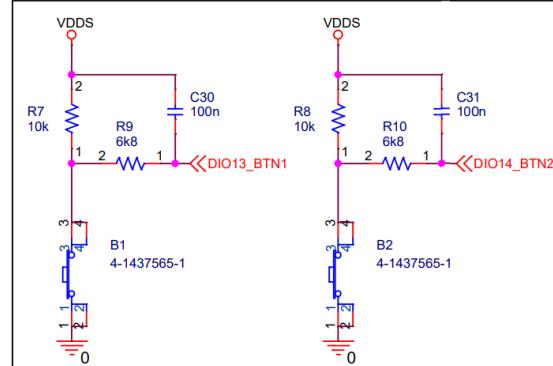


Ilustración 52 - Environmental board. LEDs and buttons schematics

- *Puertos adicionales*

En este apartado se muestran los diferentes puertos adicionales de la placa ambiental. A la izquierda se dispone del conector JTAG, así como del pin reservado para reset. El pin de reset se encuentra en estado lógico alto por defecto. Si se cortocircuita los pines “J₂” entonces el MCU se reiniciará. En cuanto a JTAG, permite la programación del firmware. se dispone de los siguientes puertos:

- CC1310_SWO: Salida de escritura Serial
- DIO16_TDO: Test de salida de datos
- DIO17_TDI: Test de entrada de datos
- CC1310_TCK: Test de señal de reloj
- CC1310_TMS: Test de modo de selección
- VDDS: fuente de alimentación externa o de la placa.

A la derecha se tienen los pines destinados a la comunicación UART para la depuración de código, los destinados a la comunicación I²C y las entradas analógico digital para la incorporación de sensores externos. Los nombres son los mismos que se muestran en el apartado Microcontrolador CC1310.

Se dispone de condensadores de bypass para asegurar una correcta tensión.

External ports

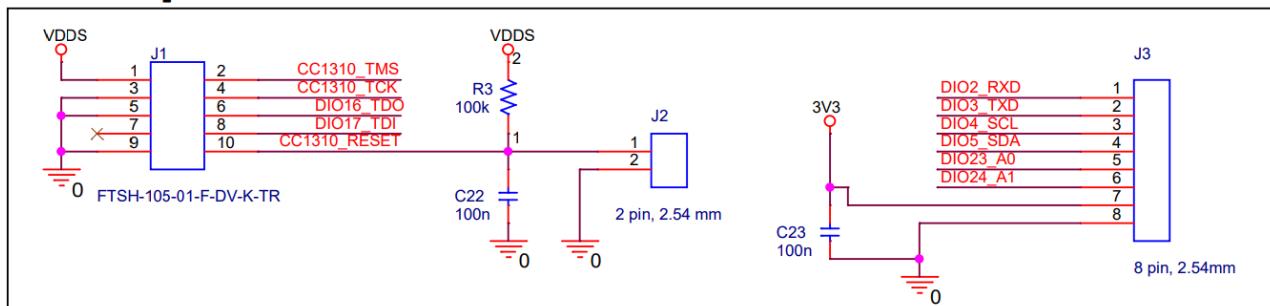


Ilustración 53 - Environmental board. External ports schematics

- *BME 680 y 280*

Se trata de los sensores que permiten tomar medidas sobre los parámetros atmosféricos. En la PCB se ha diseñado el esquemático para que se pueda incorporar tanto el sensor BME 680 como el 280. Ambos sensores pertenecen al mismo fabricante, lo único que el sensor BME 280 dispone de menor precisión y menor rango de medidas que su hermano mayor. Sin embargo, este tiene un precio menor, por lo que se decidió disponer de ambas pistas en caso de que la aplicación no exija tanta precisión, aportando mayor flexibilidad a la placa.

Para su diseño, se ha seguido el modelo de referencia aportado por su fabricante (Bosch, 2017). Se comunica mediante I²C con el MCU a través de las líneas que se explican en el apartado de Microcontrolador CC1310. En este también se dispone de condensadores de bypass para tener una tensión adecuada.

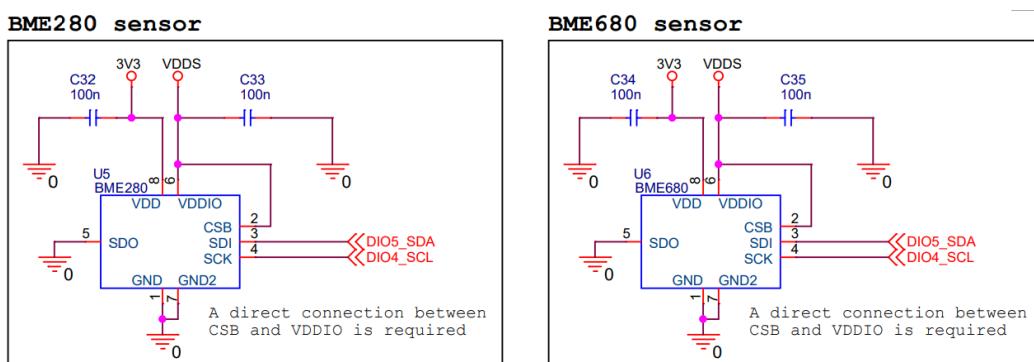


Ilustración 54 - Environmental board. BME 680 and 280 sensors schematics

- *Fuente de alimentación*

Para la alimentación de la placa se dispone de dos posibles métodos. Se puede alimentar mediante un puerto USB o a través de una batería de litio. Ambos tienen su propio circuito que se muestra a continuación.

El circuito que se muestra en la Ilustración 55 se trata de la alimentación mediante USB ("J₆"). Además de alimentar la placa, también sirve para cargar la batería en caso de que su nivel sea bajo. Para ello, se encarga el "U₃" que administra la carga de la batería. En caso de que se encuentre cargando, genera una señal por el pin "DIO10_CHG_BAT".

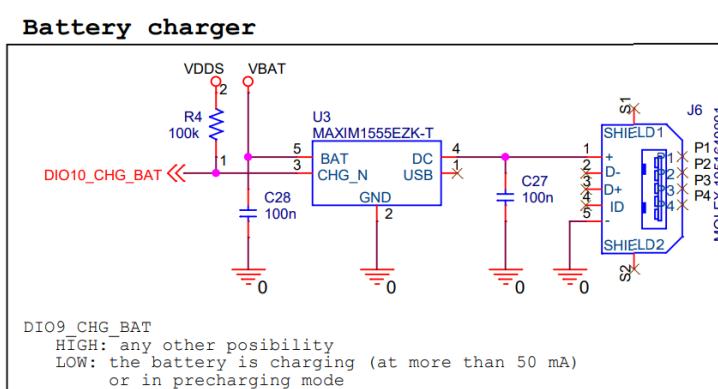


Ilustración 55 - Environmental board. Batery charger schematics

En la Ilustración 56 se muestra el circuito de alimentación en el caso de que la fuente de alimentación sea una batería. Esta se conecta en el conector “J₅”. Se dispone de dos pines “J₄” que permiten medir el consumo de corriente si se coloca un amperímetro en serie. En el funcionamiento normal, ambos se encuentran cortocircuitados con un jumper.

Debido a que la tensión proporcionada por la batería no es constante y puede ser menor a 3.3V es necesario de un conversor que resuelva el problema. Para ello, se dispone de “U₂” que puede proporcionar hasta 100 mA. Se ha escogido este circuito integrado que pueda proporcionar esa corriente ya que la corriente máxima del MCU es 25 mA, la del sensor 12 mA y cada LED 10 mA, lo que supone 77mA. Es por esto por lo que con un conversor de 100 mA es suficiente.

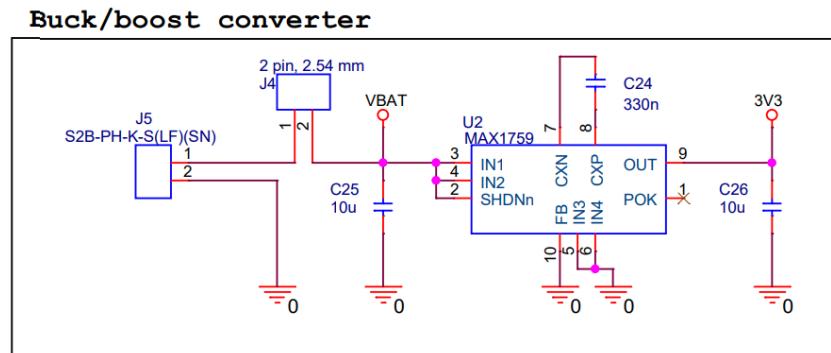


Ilustración 56 - Environmental board. Buck/boost converter schematics

En caso de que la placa tenga poca batería es necesario disponer de un sistema que avise de esta situación ya que la vida de las baterías se ve reducida si la tensión de esta se encuentra por debajo de cierto valor. Para ello se dispone del circuito de la Ilustración 57. En función del nivel de batería alerta al MCU del estado de esta. En el momento que supere cierto valor que puede ser configurado mediante I₂C, el valor de la señal “DIO11_LOW_BAT” se establece como nivel lógico bajo. Esto creará una interrupción en el MCU que hará que el sistema se apague.

A pesar de haber diseñado el circuito e incluido las pistas en la PCB, este componente no se encuentra soldado en esta. Esto se debe a que por problemas con el envío de componentes no puso ser instalado el sistema. Por lo tanto, queda como líneas futuras del proyecto.

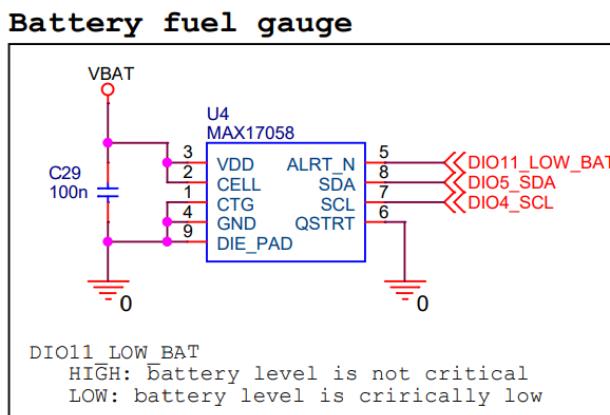


Ilustración 57 - Environmental board. Batery fuel gauge schematics

5.1.2.4 PCB

En la Ilustración 58 se puede observar la PCB real de la placa con todos los componentes soldados excepto los que por problemas con el envío de los componentes no pudieron ser instalados. Estos son, el conector “J₅” para la conexión de la batería de litio. El circuito integrado “U₄” encargado de avisar ante situaciones de batería baja. El sensor BME 280 que corresponde a “U₅”, ya que se instaló el sensor BME 680 (“U₆”).

Además, existen ciertos errores derivados de un diseño erróneo del PCB. Debido a la falta de experiencia en el diseño de PCB se colocó el conector de 2 pines “J₂” muy cerca del conector de JTAG “J₁” lo que impide que se pueda conectar este último. Para solucionar el problema, se dobló el conector “J₂”, solventando de esta forma el problema.

Uno de los objetivos principales era mantener el tamaño del PCB lo más pequeño posible y que fuera versátil. Estos objetivos se ven cumplidos ya que el PCB puede posicionarse en cualquier lado gracias a su sistema de batería, puede incorporar sensores externos con mínimas modificaciones en el hardware y firmware y su tamaño es de 7cm x 4,5cm, siendo la antena casi la mitad del largo de la placa, módulo que viene impuesto por la hoja de referencia de Texas Instruments.

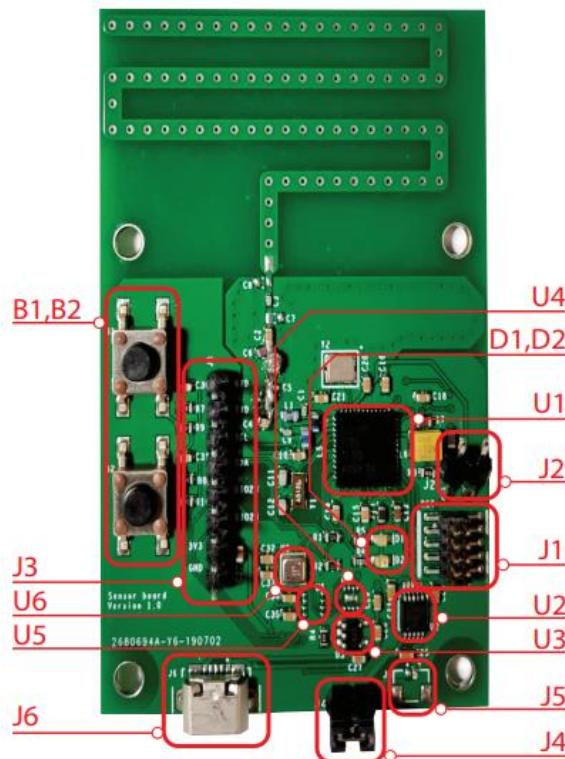


Ilustración 58 - Environmental sensor PCB

5.1.2.5 Cambios en la lista de materiales

La lista de materiales completa se encuentra en el anexo I. Sin embargo, debido a problemas con el envío de los componentes ha sido necesario realizar algunos cambios para poder tener la placa operativa. En la Tabla 10 se muestran estos cambios. Los condensadores e inductancias pertenecen a la antena como ya se ha explicado en su correspondiente apartado y las resistencias se cambiaron a su valor más cercano del que se tenía disponible en el laboratorio.

Ref	Valor original	Nuevo valor
R8	39k	56k
R12	120k	180k
C8		Abierto
C9		Abierto
L3		Cortocircuito
L4		Cortocircuito

Tabla 10 - Environmental sensor. Changed components

5.1.2.6 Firmware y funcionalidades

Para el desarrollo del firmware se ha tomado como referencia los ejemplos que aporta Texas Instruments de “TI 15.4 Stack sensor” que ya se ha visto en este capítulo, en el apartado 5.1.1.2. Respecto al sensor, también se ha tomado como referencia las librerías que proporciona el fabricante (Bosch, 2016) junto con el ejemplo de Texas Instruments sobre el sensor BME 280 para establecer la comunicación. (Texas Instruments, 2019)

- *Archivos*

Del fabricante del sensor se tienen las siguientes librerías:

- Bme680_defs.h: Contiene la declaración de macros, constantes y tipos de datos necesarios.
- Bme680.h: Contiene la declaración de la API del sensor.
- Bme680.c: Contiene la definición de la API del sensor.

Para la implementación de las APIs en la placa se utilizan dos archivos: “bme680_user_implementation.h” y “bme680_user_implementation.c”

- *Application/ bme680_user_implementation.c*

Este sensor implementa dos funciones principales “BME680_init()” y “BME680_measure()”. La primera inicializa el sensor y establece la configuración para la comunicación además de que medidas se van a tomar. La segunda se utiliza para tomar las medidas previamente establecidas en el primer archivo.

Si el usuario quisiera modificar alguna propiedad del sensor, deberá acceder a la primera función y cambiar los parámetros según se explica en la documentación del fabricante. Para realizar la toma de las medidas, será necesario llamar a la segunda función en el código y esta nos devolverá el resultado.

- *Application/ bme680_user_implementation.h*

Se trata del encabezado del archivo anterior y dispone de la declaración de las funciones anteriores:

```
void BME680_init();
void BME680_measure(struct bme680_field_data *data);
```

Código 7 - Environmental sensor. BME680 firmware functions

- *bme680.c*

También se ha realizado modificaciones al resto de archivos que aparecen en el apartado Archivos, pero las del archivo “bme680.c” son las más relevantes. Entre ellas, se encuentra la definición de funciones que no estaban completas y la adaptación de los ejemplos de Texas Instruments sobre el sensor BME 280 para que sean compatibles con el sensor BME 680. Entre las funciones modificadas, se encuentran las referentes a la lectura y escritura en el bus I²C para la comunicación con el MCU. Las funciones modificadas son las que se citan a continuación.

```
//delay function
void user_delay_ms(uint32_t period);
//reads the I2C bus
int8_t user_i2c_read( uint8_t dev_id ,uint8_t reg_addr , uint8_t
*reg_data , uint16_t len );
//write the I2C bus
    int8_t user_i2c_write( uint8_t dev_id ,uint8_t reg_addr , uint8_t
*reg_data , uint16_t len );
```

Código 8 - Environmental sensor. bme680.c functions

5.1.3 Placa de consumo de potencia eléctrica

5.1.3.1 Introducción

Esta placa permite el cálculo de la potencia eléctrica consumida en una vivienda doméstica. Para ello, dispone de un circuito capaz de obtener la tensión, corriente y desfase entre ambas, lo que permite calcular la potencia tal como se mostró en el capítulo 4, apartado 4.2.2.4.

Dispone de un transformador de tensión y de corriente que permite tomar medidas de ambas variables de una manera segura ya que aísla la parte de baja tensión además de permitir medir altas tensiones y corrientes. Los transformadores utilizados son los que se mostraron en el capítulo 4, apartado 4.3.1.5. Estos permiten obtener mediciones de como máximo 250 V RMS y hasta 60 A RMS, más que suficiente en una vivienda convencional.

Como los componentes no son precisos, existiendo tolerancias frente a los valores establecidos por el fabricante, la placa dispone de un modo calibración a través de UART para ajustar los errores y que la medida del consumo de la potencia eléctrica sea más precisa.

La placa cuenta con el MCU CC1310 como núcleo del sistema, el cual procesa los cálculos para la obtención de la potencia eléctrica. Una vez se obtiene este valor, es enviado al colector a través de Sub-1 GHz.

Como la placa tiene que estar conectada a la red eléctrica, se ha escogido prescindir de batería, y diseñar el circuito de alimentación para la propia placa a partir de la tensión de salida del transformador.

5.1.3.2 Diagrama de bloques

En la Ilustración 59 se muestra el diagrama de bloques que forma la placa de consumo de potencia eléctrica. Como se puede observar dispone de bloques similares a la anterior placa descrita en la Ilustración 48. Por ello, solo se van a describir los bloques que tengan alguna diferencia con la anterior.

- Power supply: Proporciona la energía a toda la placa. Esta es proporcionada a través del transformador de tensión de la red eléctrica. Para ello se dispone de un circuito específico que se muestra en el apartado de esquemáticos. De este sale la tensión para el MCU, para JTAG y 1.65 V de referencia para la obtención de los parámetros de consumo de potencia.
- UART: Puerto utilizado para depurar código, mostrar en otro ordenador el estado del MCU y calibrar la placa.
- Power consumption sensor: Se trata del bloque encargado en transformar los valores de tensión y corriente en valores que sean válidos para el MCU y así obtener el parámetro requerido. La comunicación con el MCU se realiza a través de entradas analógico/digitales.

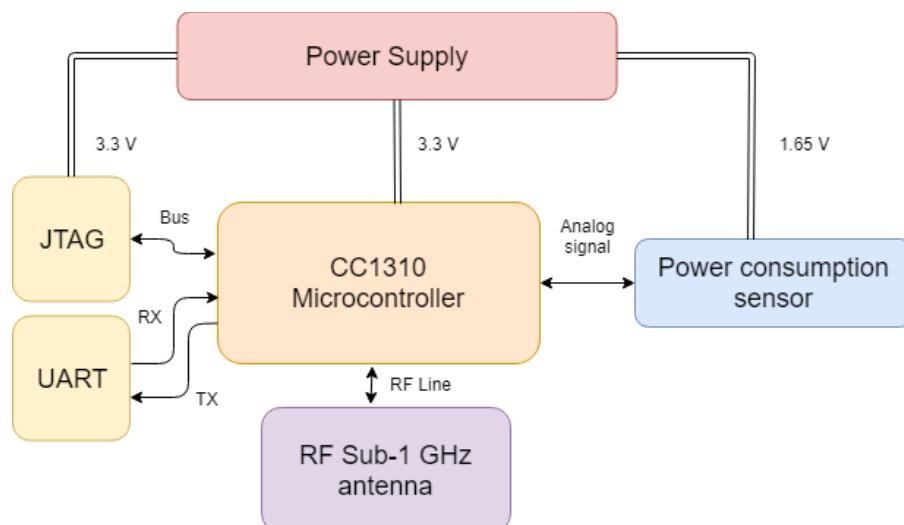


Ilustración 59 - Electric power consumption sensor block diagram

5.1.3.3 Esquemáticos

Como ocurría en la placa anterior, los esquemas se dividen en varias secciones que se muestran a continuación. Debido a que existen secciones que son iguales a las que se explican en el mismo apartado de la placa anterior, se va a omitir su explicación por redundancia. Sin embargo, sí que se van a citar para saber cuáles son.

- *Microcontrolador CC1310*

A pesar de que el MCU es el mismo que en la placa anterior, algunas de sus entradas y salidas no son las mismas. De nuevo se trata del núcleo central de la placa que procesa todos los datos.

En lo que se refiere a su alimentación se trata de las mismas alimentaciones que se vieron en la placa anterior. Su diferencia es en la fuente de alimentación que se explica en el apartado de fuente de alimentación.

Ocurre lo mismo con el condensador de desacople y los cristales osciladores, coinciden exactamente con la placa anterior. También se tiene que las entradas para la comunicación JTAG, UART, el pin de reset, los pines de los LEDs y de los botones coinciden con la anterior.

Lo único diferente frente a la placa anterior se encuentra en la parte de adquisición de datos y en la eliminación de los pines de la batería. Esta placa elimina la comunicación por I₂C y toma los datos a través de las entradas analógicas/digitales.

Estos pines son el 31, 32, 36, 37, 38 que corresponden a las medidas que se observan en la Tabla 11. En el capítulo 4, apartado 4.2.2.4, se explica con mayor detalle a qué señales se hace referencia.

Señales del esquema	Medidas
V_SQ_WAVE	Señal cuadrada de tensión
I_SQ_WAVE	Señal cuadrada de corriente
V_SIGNAL	Señal de tensión
I_SIGNAL	Señal de corriente
OFFSET_MEAS	Sin uso

Tabla 11 - Power meter board. measurement signals

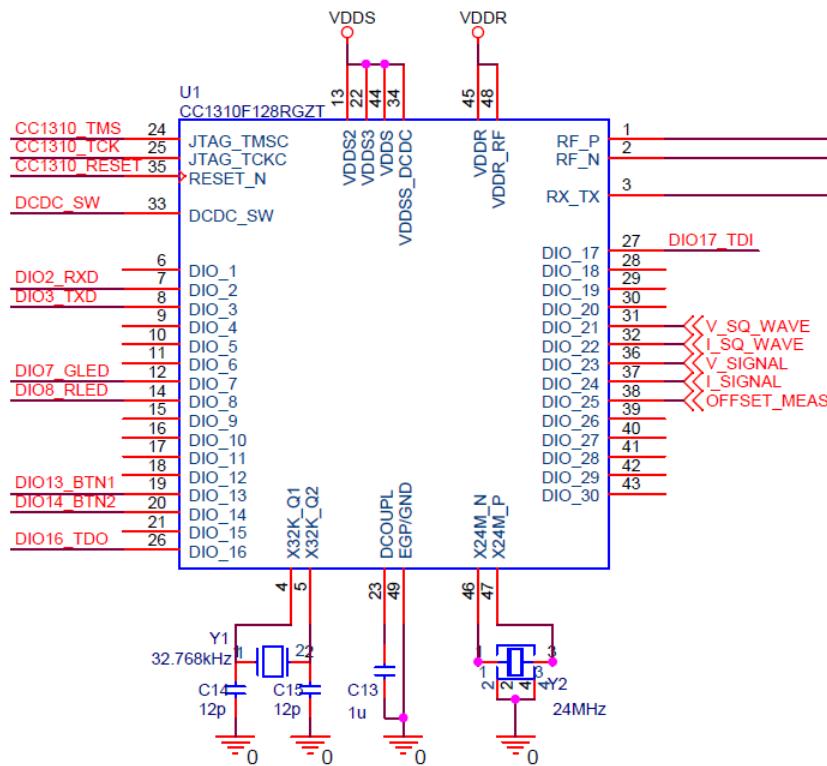


Ilustración 60 - Power meter board. CC1310 schematics

- Puertos adicionales

Esta sección también tiene mucho en común con la placa anterior. La parte de la izquierda es similar, por lo que se prescinde de su explicación. En cuanto a la parte derecha la diferencia radica en que se eliminan los pines de comunicación I₂C y las entradas analógico/digitales, conservando solo la comunicación UART.

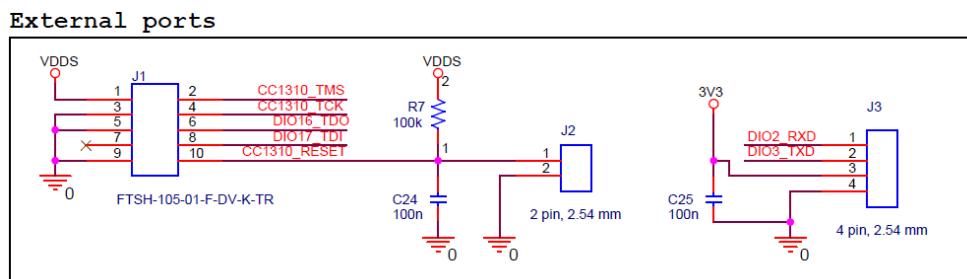


Ilustración 61 - Power meter board. External ports schematics

- *Fuente de alimentación*

Esta placa se aprovecha de su conexión a la red eléctrica para obtener la alimentación de esta. El transformador utilizado dispone de dos bobinados de salida, por lo que uno se utiliza para la fuente de alimentación y el otro para el cálculo de la potencia eléctrica.

A la salida del transformador, la onda es alterna, por lo que es necesario rectificarla. Para ello, la señal a traviesa el puente de diodos “D₃” y mediante el condensador “C₃₃” se obtiene la señal rectificada. Sin embargo, para obtener una tensión de salida constante y de valor 3.3 V, se utiliza el regulador de tensión “U₅”. Para reducir el ruido de altas frecuencias se añade una inductancia de ferrita “FL₂” que las elimina. El LED “D₄” se ilumina cuando la placa se conecta a la corriente para indicar al usuario que hay energía en el sistema. Para terminar, se dispone de la resistencia “R21” de valor 0 ohmios que une las dos tierras.

El transformador puede suministrar como máximo 125 mA por cada bobinado secundario. Este valor es suficiente para alimentar los LEDs de 20 mA cada uno y el MCU que consume como máximo 25 mA, siendo el total 65 mA. En cuanto al regulador de tensión, puede suministrar de máxima 250 mA. El motivo de escoger este valor tan alto viene impuesto por la placa del colector, ya que esta consume más energía y con el objetivo de minimizar costes, se escogió dos reguladores idénticos.

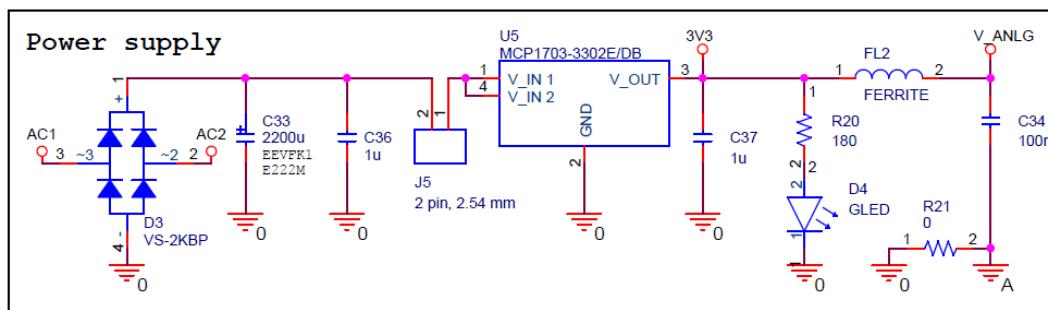


Ilustración 62 - Power meter board. Power supply schematic

- *Circuito de acondicionamiento de la tensión*

Esta sección del esquema es la encargada de condicionar la tensión de la red eléctrica a una a la que el MCU pueda procesar. En el capítulo 4, apartado 4.2.2.4 se explica el porqué de utilizar este método con los bloques que aquí se muestran como elementos electrónicos. De esta sección se obtienen la señal “V_SIGNAL” y “V_SQ_WAVE” que se han visto en el apartado del MCU para calcular la potencia eléctrica.

Para obtener ambas señales, la señal de entrada se debe condicionar. Mediante el transformador “TR₁” y el divisor de tensión “R₈” y “R₁₁” se logra obtener una tensión de 3.3 V pico a pico. A esta tensión se le suma una tensión de offset de 1.65 V para que la señal esté acotada entre 0 y 3.3 V. Esta tensión de offset se verá a continuación en la sección de señal de offset de esta placa. El resultado de esta operación es el que se muestra en la ilustración superior de la Ilustración 32.

Posteriormente se aplica un filtro paso bajo de segundo orden “U_{2B}” para eliminar el ruido y los armónicos. Su frecuencia de corte es de 1 kHz, ya que no se busca que sea tampoco reducida porque se podría perder información de la señal. De este se obtiene ya la señal “V_SIGNAL” que entra directamente al MCU.

Además, se dispone de un comparador “U₃” que compara la señal “V_SIGNAL” con el valor de tensión de offset. De esta manera se obtiene una señal cuadrada “V_SQ_WAVE” similar a la ilustración inferior de la Ilustración 32 con valores lógicos altos (3.3 V) cuando la señal “V_SIGNAL” es mayor que el offset. De esta manera se puede calcular el desfase entre la tensión y la corriente al disponer de otra sección similar a esta para la corriente.

Para que todo esto funcione, es necesario conectar en paralelo a la red eléctrica este bloque a través del conector “J₄”

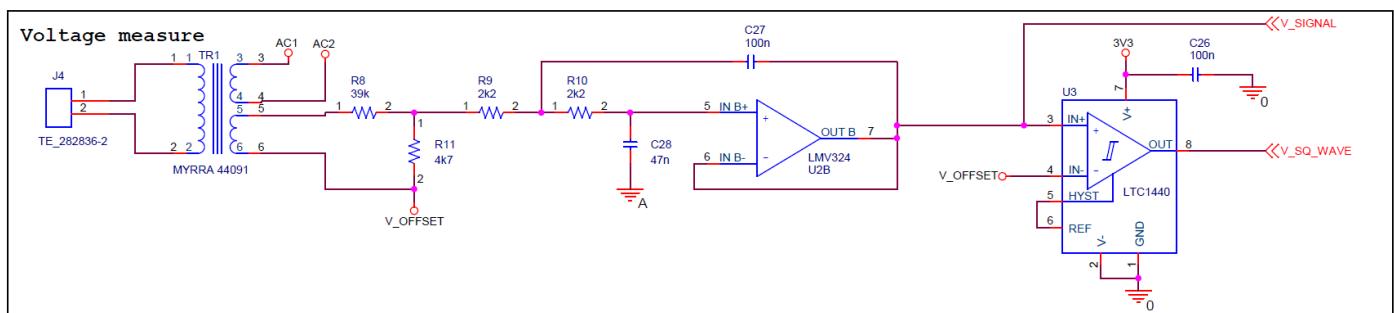


Ilustración 63 - Power meter board. Voltage conditioning schematic

- *Circuito de acondicionamiento de la corriente*

Esta sección se comporta exactamente igual que la anterior, a diferencia de que esta dispone de un transformador de corriente a tensión. Al igual que la anterior, se obtienen dos señales, “I_SIGNAL” y “I_SQ_WAVE” similares a las que se observan en la Ilustración 33 para poder procesar la corriente.

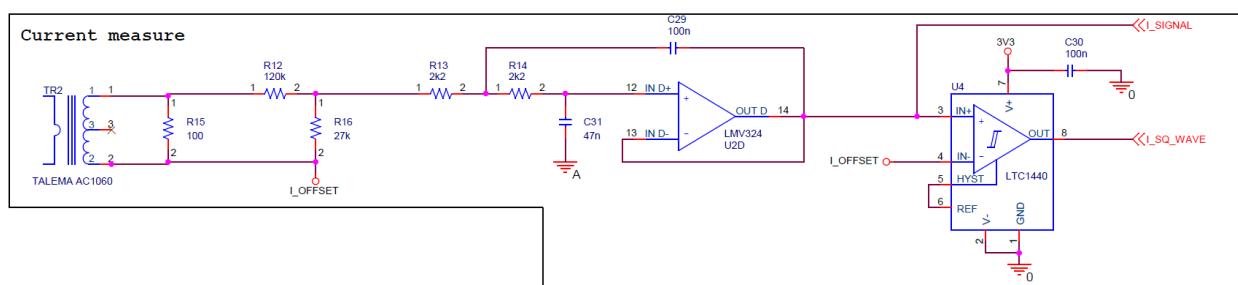


Ilustración 64 - Power meter board. Current conditioning schematic

- *Señal de offset*

Esta sección se encarga de producir la tensión de offset necesaria para poder tener valores positivos que le MCU pueda leer. Su comportamiento es muy sencillo, se trata de un divisor de tensión “R17” y “R18” que reducen los 3.3 V de tensión de la fuente de alimentación a los 1.65 V necesarios. Posteriormente se dispone de dos seguidores de tensión que “U_{2A}” y “U_{2C}” que hacen que la tensión de offset de salida sea independiente al circuito de la fuente de alimentación. Además, se incorporan condensadores a la salida con el objetivo de que la tensión de offset sea lo más estable posible.

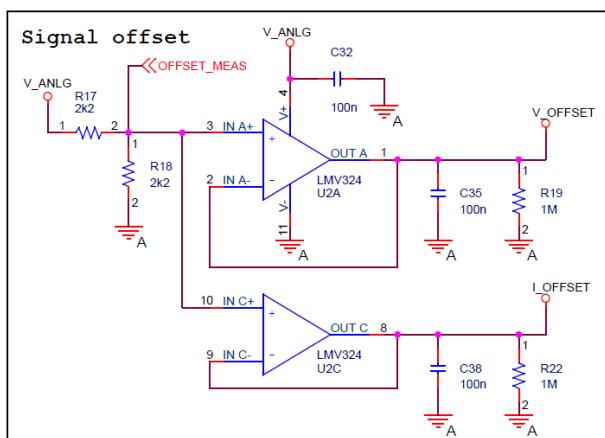


Ilustración 65 - Power meter board. Signal offset schematic

- *Antena RF, Bypass, LEDs RF y Botones*

Estas secciones son idénticas a las que se analizan en la placa anterior, por lo que se prescinde de su explicación.

5.1.3.4 PCB

La Ilustración 66 muestra la PCB de la placa de consumo de potencia eléctrica totalmente ensamblada, excepto los que por problemas de envío no fue posible su ensamblaje. Estos se detallan en el siguiente apartado.

Existen ciertos errores que al igual que con la placa anterior, derivan de la falta de experiencia en el diseño de PCBs. Este se trata del fallo al diseñar las pistas para el regulador de tensión MCP1703 que aparece en la Ilustración 66 o en el esquemático de la Ilustración 62 como “U₅”. El fallo proviene por una confusión en su encapsulado lo que provoca que el “footprint” sea incorrecto. Sin embargo, este problema se soluciona al dejar un pin del regulador sin soldar.

Como en la placa anterior se buscaba minimizar el tamaño lo máximo posible siendo de 5.5 cm x 11 cm. Además, se buscaba que los cables de la red se pudieran conectar sin dificultad a la placa haciendo pasar el cable por el transformador de corriente de forma fácil y segura.

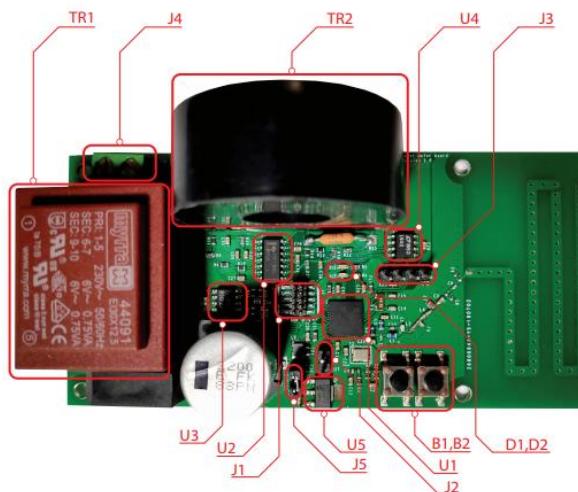


Ilustración 66 - Power meter board PCB

5.1.3.5 Cambios en la lista de materiales

La lista de materiales completa se encuentra en el anexo I. Sin embargo, como ocurría en la placa anterior debido a problemas con el envío de los componentes ha sido necesario realizar algunos cambios para poder tener la placa operativa. Los componentes que se han sustituido son los mismos que en la placa anterior y se muestran en la Tabla 10.

5.1.3.6 Firmware y funcionalidades

Como se adelantaba en el apartado 5.1.1.2 de este capítulo, el firmware desarrollado para esta placa se basa en el ejemplo aportado por Texas Instruments de “TI 15.4 Stack sensor”. Se van a comentar los archivos y el procedimiento que sigue el MCU para obtener a partir de las señales de entrada, la potencia eléctrica consumida.

- *Archivos*

Para el cálculo de la potencia eléctrica necesitan los siguientes archivos que se encuentran dentro de “Application/PowerMeter”.

- power_meter_config.h: Contiene parámetros de configuración a establecer por el usuario.
- power_meter.c: Contiene el código propiamente dicho para el cálculo de la potencia y la calibración.
- power_meter.h: Se trata de la declaración de las funciones y funciones del archivo anterior

- *Algoritmo para la obtención de la potencia eléctrica*

En el capítulo 4, apartado 4.2.2.4 se explica cuál es el proceso que se debe seguir para la obtención de la potencia eléctrica. Sin embargo, en este apartado se va a entrar en mayor detalle sobre cómo se obtiene el valor de salida. En este proceso entran en juego las señales vistas en el apartado anterior de “V_SIGNAL”, “I_SIGNAL”, “V_SQ_WAVE” y “I_SQ_WAVE”. Los siguientes puntos explican el algoritmo para la tensión, pero ocurre lo mismo para la corriente.

1. Se comienza con la detección de un flanco de subida de “V_SQ_WAVE” que activa una interrupción y permite la adquisición ADC de “n” períodos de “V_SIGNAL”. De esta operación se obtiene un array que se denomina “V_{ADC}”.
2. Para conocer cuál ha sido el valor de tensión de offset utilizado para poder restárselo a la señal y obtener la onda de origen, se calcula el valor medio de “V_{ADC}”, obteniendo “V_{offset}”.
3. El siguiente paso es el cálculo del valor de tensión RMS. Primero se le resta a “V_{ADC}” la tensión de offset “V_{offset}” y se calcula la tensión RMS, dando “V_{RMS}”.
4. Con el valor “V_{RMS}” ya se puede obtener el valor original a la entrada del transformador (“V_{PRIM}”). Para ello, es necesario revertir las transformaciones realizadas en el esquemático. Esto es, multiplicar el divisor de tensión y la relación de transformación del transformador.

Con el algoritmo que se ha descrito ya se puede conocer la tensión y corriente de la red eléctrica, pero aún falta conocer la frecuencia de la red y el desfase que existe entre ambas señales.

Para calcular la frecuencia, como ya se analizaba en el capítulo anterior, se mide el tiempo entre dos flancos de la onda cuadrada. En la Ilustración 28 se puede observar lo que se entiende como flancos y periodo. Se calcula para la señal de tensión, pero el cálculo de la señal de corriente proporciona el mismo valor. Por lo tanto, para obtener la frecuencia:

1. Cuando se detecta un flanco de subida en “V_SQ_WAVE” se produce una interrupción que inicializa un “timer”.
2. Para evitar errores de precisión debidos al tiempo que tarda en inicializarse el “timer”, se espera a que ocurra el flanco de bajada. Cuando este ocurre, se activa el “timer” y mide el tiempo de “n” flancos de bajada.
3. De esta forma se puede realizar la media de tiempo que se ha obtenido dividiendo el tiempo total entre los “n” flancos medidos. Con esto se obtiene el periodo, y realizando la inversa la frecuencia.

Por último, queda calcular el valor del desfase entre ambas señales. Debido al uso de los transformadores, estos introducen también un desfase que deberá ser corregido a través de la calibración. Esta se debe realizar con una carga resistiva pura, en el que el desfase debería ser 0, pero debido a los transformadores será diferente. De esta forma, si el desfase en funcionamiento normal sin calibrar es “p”, en la calibración se puede obtener “p_e” y así obtener el desfase real de la red “p_r”, mediante la Ecuación 11. A partir de esta ecuación y la Ecuación 8, se obtiene el mismo resultado, pero para el ángulo de desfase que es el que se busca para el cálculo de la potencia eléctrica.

$$p_r = p - p_e$$

Ecuación 11 - Phase difference calibration

Para la obtención de “p” es necesario realizar una serie de pasos. En la Ilustración 28 se puede observar a qué se hace referencia para entender mejor el algoritmo que se describe a continuación. Para evitar errores en caso de que se conecten cargas resitivas puras que provoquen que el desfase sea cercano a 0, se toman “n” períodos para tener más precisión.

1. Cada vez se detecta un flanco de subida en “V_SQ_WAVE” se produce una interrupción que activa un “timer”.
2. Simultáneamente, cada vez que se detecta un flanco de subida en “I_SQ_WAVE” se produce otra interrupción. En este caso, esta interrupción lo que hace es comprobar si el número de periodos “n” de “V_SQ_WAVE” ha sido alcanzado.
3. El “timer” sigue contando hasta que se produce una interrupción en “I_SQ_WAVE” por flanco de subida y confirma que el número de periodos “n” ha sido alcanzado, lo que detiene el “timer”. De esta forma se ha conseguido contar el tiempo desde el primer flanco de subida de “V_SQ_WAVE” hasta el “n” flanco de subida de “I_SQ_WAVE”, lo que se denomina “ p_n ”
4. Al conocer el tiempo entre los flancos de subida de ambas señales y conocer el número de periodos realizados “n” y el periodo de la señal “T”, se puede obtener el desfase entre ambas señales con la ecuación:

$$p = p_n - nT$$

Ecuación 12 - Phase difference calculation

5. Con “p” ya se puede calcular el ángulo de desfase “ φ ” con la Ecuación 8.

Así con estos 3 algoritmos se pueden calcular los parámetros necesarios para obtener la potencia eléctrica que son: “ V_{PRIM} ”, “ I_{PRIM} ” y “ φ ”, aplicando la Ecuación 7. Existe un caso en el que, en vez de aplicar la ecuación anterior, se aplica la Ecuación 5. Esto ocurre cuando la corriente es demasiado pequeña que provoca errores en el cálculo del ángulo de desfase, por lo que se aproxima el ángulo a 0° .

La potencia calculada a través de los tres parámetros vistos en el párrafo anterior es correcta. Sin embargo, la placa dispone de un modo calibración que permite eliminar los errores introducidos por los transformadores, obteniendo en vez de “ φ ”, “ φ_r ”. Para ello, es necesario acceder al modo calibración pulsando durante unos segundos los botones de la placa y mediante UART introducir los valores que se necesitan para su calibración.

Los valores que se necesitan para la calibración se obtienen de enchufar la placa a la red eléctrica “G₁”, colocar dos multímetros en la configuración que se observa en la Ilustración 67 junto con una carga resistiva pura “R_L”.

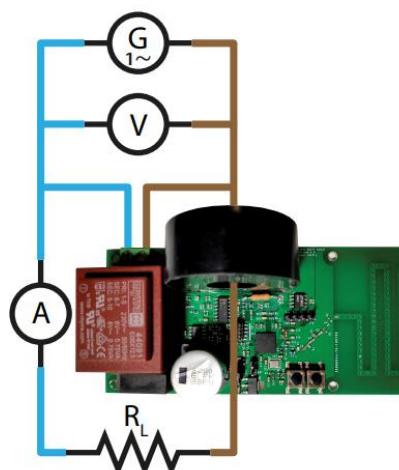


Ilustración 67 - Power meter board. Calibration configuration

5.1.4 Placa hub central

5.1.4.1 Introducción

Esta placa se denomina HUB central, pero también se la puede denominar colector, ya que recibe mediante Sub-1 GHz la información de todos los sensores que están conectados a su red y por otro lado Gateway, ya que transmite esta información a través de WiFi a un servidor para procesarla.

A diferencia del resto de placas, esta placa se compone de dos MCU. Por un lado, dispone del MCU CC1310 ya que permite la comunicación con los sensores por Sub-1 GHz. Por otro lado, se dispone del MCU ESP 32 que recibe esa información a través de UART y la transmite al servidor por WiFi. Se escogió el uso de dos MCU ya que no se encontró uno que cumpliera con todas estas necesidades.

Esto además facilita el desarrollo del firmware, ya que por un lado se desarrolla el firmware del CC1310 para la adquisición de datos y UART, y por otro el firmware de la recepción de los datos y transmisión mediante MQTT. Esto permite crear firmwares más pequeños que facilitan la detección de errores. De esta forma, como ya se adelantaba al principio de este capítulo, en este apartado se va a analizar la parte relativa al MCU CC1310 y al hardware de la placa y en el apartado 5.2, se analiza la parte software del ESP 32.

Para la alimentación de la placa se dispone de una entrada USB a 5 V sin posibilidad de batería, ya que la transmisión de datos a través de WiFi consume mucha energía. Sin embargo, como el rango de Sub-1 GHz es tan elevado y la conexión a Internet es inalámbrica, la placa se puede colocar en cualquier lugar de la vivienda siempre y cuando disponga de una toma de corriente eléctrica.

5.1.4.2 Diagrama de bloques

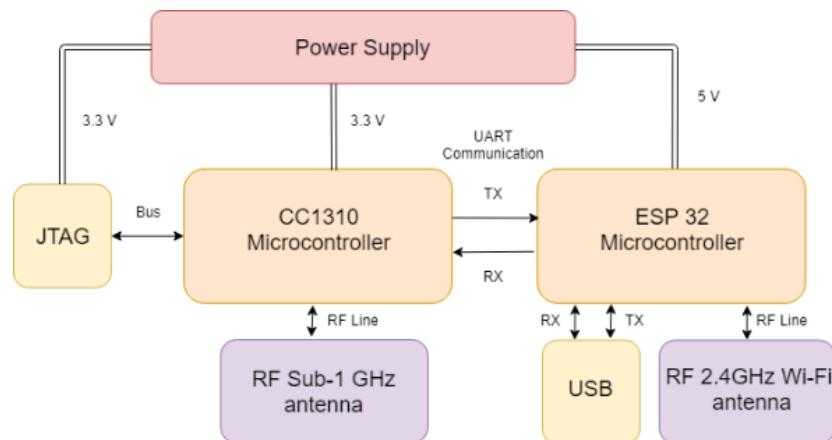


Ilustración 68 - Central hub block diagram

En la Ilustración 68 se muestra el diagrama de bloques de la placa HUB central. Se va a describir los bloques excepto aquellos que son iguales a los vistos en las placas anteriores.

- Power supply: Proporciona la energía a toda la placa. Se trata de una conexión por USB de 5V. Dispone de un regulador de tensión para alimentar los 3.3 V necesarios del MCU CC1310 y los 5 V del MCU ESP 32 se toman directamente de la alimentación USB.

- ESP 32 MCU: Se trata del MCU que se muestra en el capítulo 4, apartado 4.3.1.2. Recibe la información de los sensores a través de UART.
- USB: Puerto USB que permite la comunicación Serial entre el MCU ESP 32 y un ordenador. Este puerto se utiliza para programar el firmware del ESP 32, depurarlo y configurar los parámetros de la red.
- RF 2.4GHz Wi-Fi antenna: Antena de 2.4 GHz que permite la conexión mediante WiFi a Internet.

5.1.4.3 Esquemáticos

Este apartado se organiza de la misma forma que la placa anterior. Se omitirán las secciones del esquema que sean similares a las anteriores.

- *Microcontrolador CC1310*

Al igual que ocurría en la placa anterior, la mayoría de los elementos que aparecen en la Ilustración 69 coinciden. Estos son: las entradas de alimentación, el condensador de desacople, los cristales osciladores, la comunicación JTAG, el pin de reset, las salidas para los LEDs y las entradas de los botones.

Incluso en este esquema coinciden los pines para la comunicación UART. Sin embargo, estos se utilizan para la comunicación con el MCU ESP 32, aunque también se dispone de un conector de dos pines en la placa para leer mediante otro dispositivo que se está transmitiendo.

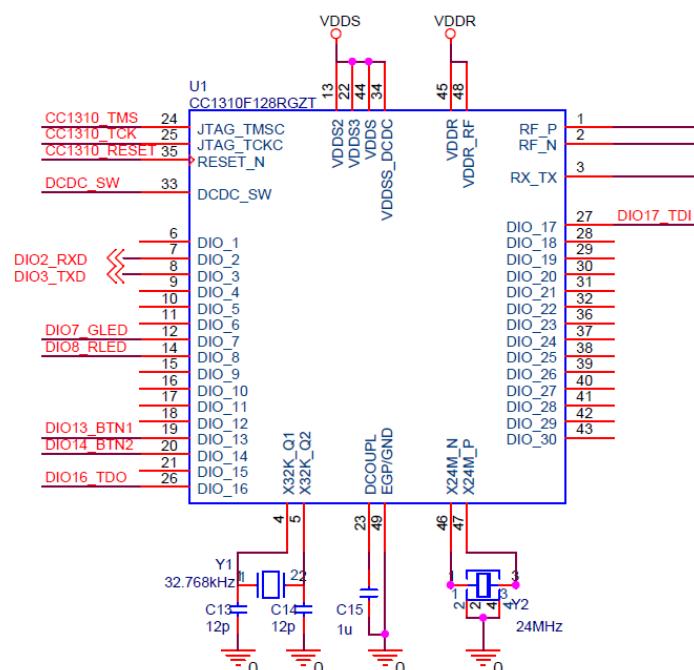


Ilustración 69 - Central hub board. CC1310 schematics

- **ESP 32**

Se trata del MCU ESP 32 que se utiliza para la transmisión de la información a través del protocolo MQTT mediante WiFi al servidor donde se procesa la información. Para alimentar el MCU se utiliza los 5 V procedentes de la fuente de alimentación USB. Al tratarse de una placa de desarrollo en vez del MCU solo, internamente dispone de un regulador de tensión que proporciona 3.3 V. Sin embargo, estos 3.3 V no se utilizan para alimentar el otro MCU.

En la Ilustración 70 solo se muestra las conexiones que se han utilizado para este proyecto. Como es una placa de desarrollo, dentro de “U₃” dispone de más circuitos eléctricos. No obstante, no son relevantes para el desarrollo del proyecto en cuestión.

Para la comunicación UART con el MCU CC1310 utiliza las entradas y salidas 16 y 17 teniendo las señales “DIO3_TXD” y “DIO2_RXD”. A pesar de ser lectura y escritura, el firmware solo está diseñado para que le ESP 32 lea información del CC1310. El MCU dispone de 3 puertos para la comunicación UART. Los pines 16 y 17 corresponden al segundo puerto ya que solo se puede utilizar ese. Esto se debe a que el primero está reservado para la comunicación entre el MCU y el ordenador mediante el cual se programa el firmware, se depura el código y se configura el dispositivo.

Dispone de dos salidas digitales “WIFI_LED”, “MSG_LED” y “CFG_LED” que se utilizan para encender y apagar los LEDs de indicación de estado. Estos se verán en más adelante en su sección.

La señal de entrada “CFG_SEL” sirve para acceder al modo configuración del ESP 32. De nuevo, se verá en su sección.

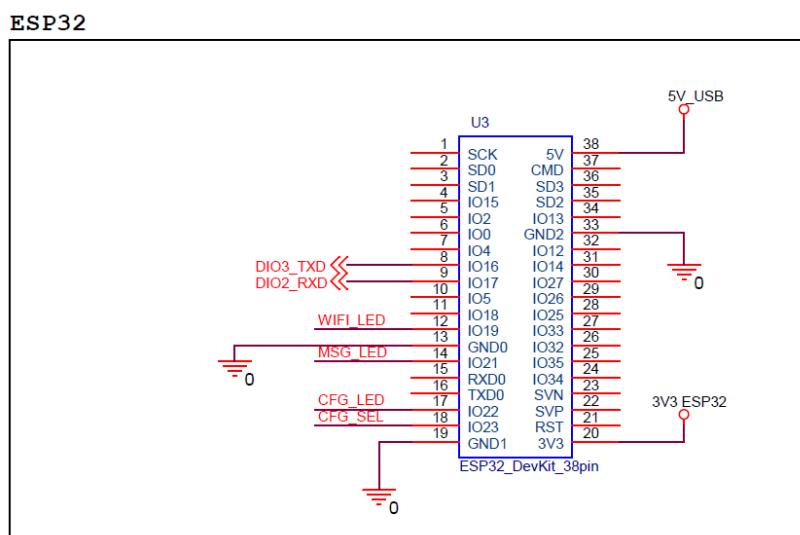


Ilustración 70 - Central hub board. ESP 32 schematics

- **Jumper de configuración y ESP 32 LEDs**

Para acceder al modo configuración se dispone del esquema de la Ilustración 71. Se dispone de la alimentación de 3.3 V del ESP 32 y de un conector de 3 pines. De este modo, cuando el jumper se posiciona en los pines 1 y 2, el MCU funciona en modo, ya que la señal “CFG_SEL” está en valor lógico alto. Sin embargo, si el jumper se posiciona en los pines 2 y 3, “CFG_SEL” toma valor lógico bajo y el MCU está preparado para entrar en modo configuración. El modo configuración solo se comprueba cada vez que se reinicia el MCU, por lo que será necesario pulsar el botón reset.

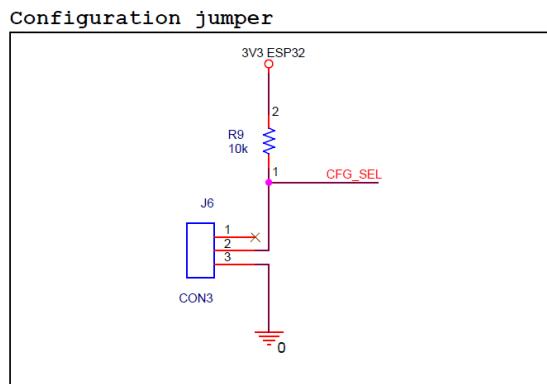


Ilustración 71 - Central hub board. Configuration jumper schematics

En la Ilustración 72 se puede ver los indicadores de estado LED para indicar en qué estado se encuentra el ESP 32, tal y como ocurría con el CC1310. En este caso se tiene la señal “WIFI_LED” que estará activa siempre y cuando exista conexión con la red WiFi. La señal “MSG_LED” se activa cuando se recibe y envía un mensaje. Por último, la señal “CFG_LED” se activa si el programa se encuentra en modo configuración.

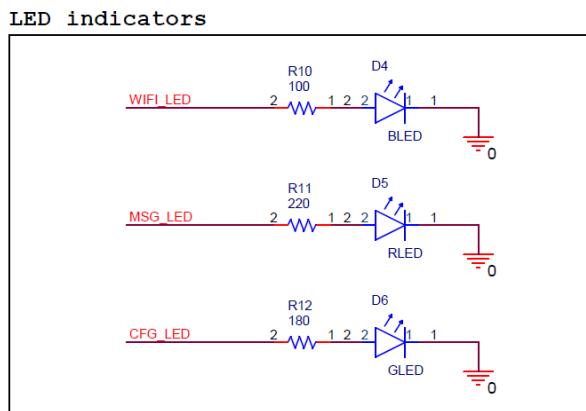


Ilustración 72 - Central hub board. LED indicators schematics

- *Fuente de alimentación*

En la fuente de alimentación se cuenta con una entrada micro USB que alimenta todo el sistema. Se dispone de un conector de dos pines “J₄”, el cual en condiciones normales tiene un jumper conectado. El objetivo de este conector es medir la corriente que consume la placa al conectar un amperímetro en los pines, con fines de mejora de la placa.

Directamente de la alimentación USB salen los 5 V para el ESP 32. Después se encuentra un regulador de tensión que suministra los 3.3 V para el MCU CC1310. También se dispone de un LED verde “D₃” que indica que la placa está conectada a la corriente.

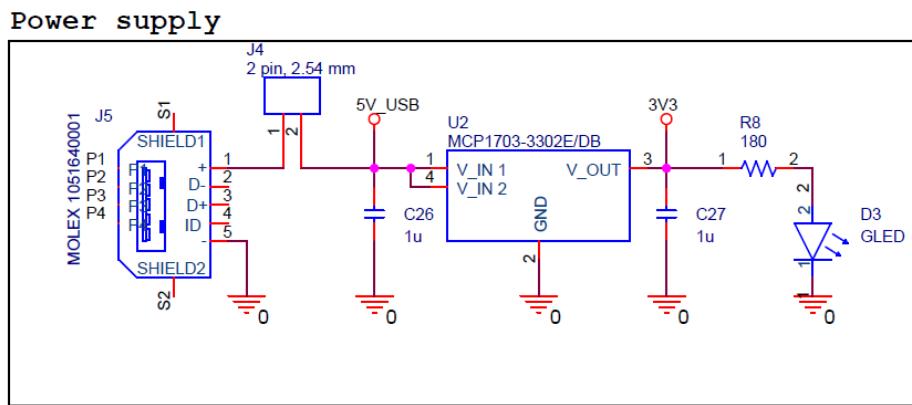


Ilustración 73 - Central hub board. Power supply schematics

- Antena RF, Bypass, LEDs RF, Botones, Puertos adicionales

Estas secciones son idénticas a las que se analizan en la placa anterior, por lo que se prescinde de su explicación.

5.1.4.4 PCB

En la Ilustración 74 se muestra el PCB completo de la placa HUB central. En la ilustración de la derecha se muestra sin el ESP 32 conectado, mientras que en el de la derecha está completa.

En esta placa también se encuentran errores en su diseño. El primero y más visible se encuentra en "U2", que es el regulador de tensión. En este caso, al ser el mismo que el de la placa de potencia, se cometió el mismo error en el diseño. A la hora de solucionar el problema, primero se intentó en esta conectando uno de los pines del regulador a la tensión de entrada mediante un cable. Posteriormente se observó que con dejar un pin del regulador sin conectar bastaba, por lo que esa solución se aplicó a la placa de consumo de potencia eléctrica.

El segundo error es parecido al que se encontraba en la placa de sensores ambientales. Al diseñar el conector JTAG no se tuvo en cuenta las dimensiones del cable que se debe conectar. Una vez soldados los componentes se observó que los pines hembra de la derecha del ESP 32 no permitían introducir el cable de la conexión JTAG. Para solucionar este problema, se cortó los conectores hembra que colisionaban con el conector JTAG ya que ninguno de ellos tenía una función en el proyecto.

Como se puede observar, se posicionó el ESP 32 sobre el propio circuito de forma que se redujera al máximo el tamaño del PCB, siendo este de 8.5 cm x 4 cm. Además, se diseñó de forma que ambas antenas se encontraran enfrentadas, encontrándose la antena de Sub-1 GHz en la parte superior y la antena WiFi en la parte inferior, para evitar al máximo las interferencias.

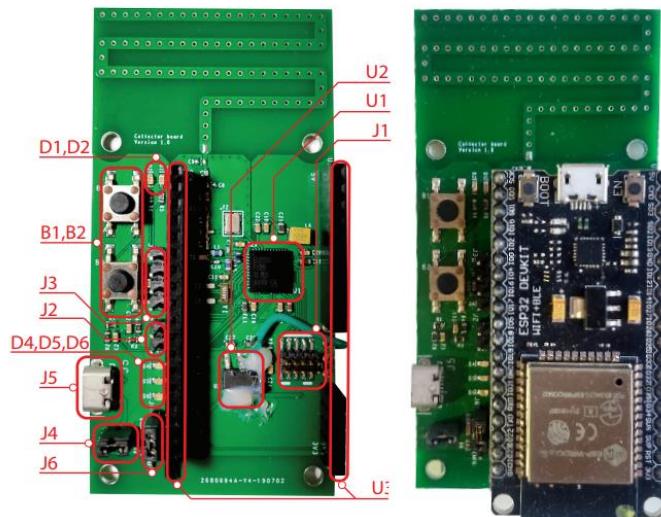


Ilustración 74 - Central hub board PCB

5.1.4.5 Cambios en la lista de materiales

En el anexo I se encuentra la lista completa de los materiales utilizados para esta placa. Los materiales que se han modificado son los mismos que aparecían en las placas anteriores. En la Tabla 10 se encuentran estos elementos.

5.1.4.6 Firmware y funcionalidades

En cuanto a este apartado, en lo que se refiere a la parte del MCU CC1310, sus archivos y funciones principales se han comentado en el apartado 5.1.1.2, en la parte de “Firmware del colector”. Por otro lado, se tiene el firmware del MCU ESP 32, el cual se comenta en el siguiente apartado 5.2.1.

5.2 PARTE SOFTWARE

Después de ver en el apartado anterior la parte física de la plataforma, este apartado se va a centrar más en la parte virtual de la misma. Si se vuelve a la Ilustración 47, esta parte comprendería el firmware del MCU ESP32 y el desarrollo del servidor.

Para conocer la arquitectura de esta parte, se va a dividir por un lado el ESP 32 y por otro cada bloque que compone el servidor. Además, se detallan las APIs que utiliza cada bloque y el tipo de mensaje utilizado para que la comunicación entre bloques sea posible.

5.2.1 Comunicación con el servidor: ESP 32

A diferencia del apartado 5.1.4, este se centra en el firmware que lleva incorporado el MCU ESP 32 que permite la transmisión de los datos de los sensores al servidor a través de Internet. Esto se puede llevar a cabo gracias a los protocolos que lleva incluido el MCU.

- *Instalación y gestión de software*

Como ya se ha visto anteriormente, el MCU se programa en MicroPython. Esto tiene ventajas e inconvenientes. Por un lado, la gestión de los datos con MicroPython resulta más cómoda al ser un lenguaje de alto nivel y disponer de tipos de datos y módulos específicos para ello. Sin embargo, no dispone de un entorno de programación oficial que permita subir el firmware al MCU con la misma sencillez que se encuentra en C/C++. A pesar de esto, se decide programar en MicroPython por los motivos que se han visto a lo largo del proyecto.

Para la comunicación con el MCU se hace uso de tres softwares diferentes compatibles con Windows. Por un lado está “PuTTY”, que permite la comunicación serial con el MCU para poder instalar MicroPython, subir el firmware y observar en qué estado se encuentra del programa. Junto con este software, se usan las herramientas de Python:

- El módulo “esptool” permite instalar (flashear) MicroPython en el MCU (Hymel, How to load MicroPython on a Microcontroller Board, 2019).
- Una vez se tiene flasheado el MCU, se utiliza el módulo “ampy” para interaccionar con el MCU. Este permite manipular los archivos a través de la línea de comandos (tdicola, dhalbert, & dezxpy, 2016).
- También se utiliza “REPL” (Read-Eval-Print-Loop) que es la forma más sencilla de interaccionar con la placa, enviando comandos a través de la comunicación serial. (Hymel, MicroPython Programming Tutorial: Getting Started with the ESP32 Thing, 2019)

En la bibliografía utilizada en los puntos anteriores se detalla el uso de cada herramienta en caso de que el lector quiera llevar a cabo el proceso. Además de la bibliografía anterior, se añade otro documento más de la empresa Adafruit para intentar facilitar al lector su puesta en práctica. (DiCola, 2019)

Por otro lado, están los programas que funcionan como entornos de programación que permiten subir el firmware y gestionar los archivos. Se ha hecho uso de “uPyCraft” y de “ESPlorer”. Ambos softwares pertenecen a desarrolladores independientes, por lo que no cuentan con la misma robustez y fiabilidad que entornos de programación profesionales como ocurre en C/C++. Ambos cuentan con una IDE parecida que permite editar los archivos existentes en el MCU, subir el firmware y obtener la respuesta por pantalla. Para su instalación y funcionamiento se recomienda la siguiente bibliografía:

- uPyCraft: (DFRobot Community, 2019), (ouki-wang & mengbishi, 2018).
- ESPlorer: (4refr0nt & nulrik, 2014).

Se aconseja el uso de uPyCraft para subir el firmware y archivos al MCU y posteriormente hacer uso de PuTTY o ESPlorer para el depurado del código.

- *Firmware y archivos*

Para que el MCU funcione correctamente se tiene que incorporar el firmware y una serie de archivos que funcionan como memoria y como módulos adicionales a los que posee MicroPython. En la Ilustración 75 se puede observar los archivos que se almacenan en la memoria del ESP 32, que responde a la carpeta “device” en la ilustración.

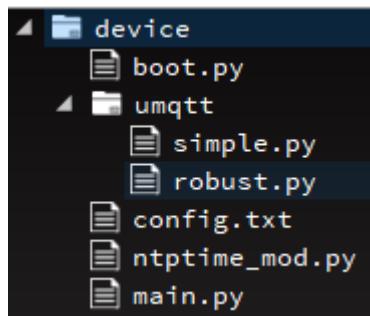


Ilustración 75 - ESP32 firmware and files

Existen dos archivos principales, el “boot.py” y el “main.py”. El primero se ejecuta al comienzo de cada reinicio incluso cuando el MCU se “despierta” de estados de baja energía. El segundo es el firmware principal que se ejecuta inmediatamente después del primero. En el proyecto no se hace uso de “boot.py”, pero es necesario que exista.

Los archivos “ntptime_mod.py” y los que aparecen en la carpeta “umqtt”; “simple.py” y “robust.py” son módulos que no incorpora MicroPython y son necesarios para la gestión de los datos. El primero es necesario para añadir a los datos de los sensores la hora real en la que se realiza la medición. Esta hora es tomada del servicio NTP por los motivos que se analizan en el capítulo 4, apartado 4.2.1.3. Este archivo está modificado respecto al original (dpgeorge, 2018) ya que es necesario corregir la hora de referencia.

Los otros dos archivos son los módulos para gestionar la comunicación MQTT con el servidor. El archivo “simple.py” contiene todo lo necesario para llevar a cabo la comunicación, pero el archivo “robust.py” realiza la reconexión automática con el bróker en caso de errores en la red. Es por esto por lo que se escoge “robust.py”, a pesar de que consume más recursos. Cabe señalar que este módulo no soporta nivel de servicio 2. (pfalcon, dxxb, & dpgeorge, 2014)

El archivo “config.txt” almacena la información básica necesaria para establecer la comunicación con el bróker y con la red Wi-Fi, así como información de donde se encuentra el hub central. A continuación, se observa un ejemplo de la información que contiene el archivo:

```
{
    "BROKER_IP": "mqtt.eclipse.org",
    "BROKER_PORT": 1883,
    "SSID": "SSID_Wi-Fi",
    "PASSWORD": "Contraseña_Wi-Fi",
    "BUILDING": "Edificio_hub_central",
    "FLOOR": "Planta_hub_central"
}
```

Código 9 - ESP 32 configuration file

Como se observa es una variable de tipo diccionario, en el que aparece la IP y puerto del bróker, la SSID y contraseña de la red Wi-Fi a la que necesita conectarse y el edificio y piso en el que se encuentra el hub central. Esta última información es utilizada en la parte de monitoreo del servidor. Permite conocer al usuario a qué hub central están conectados los sensores y donde se encuentran ubicados de manera sencilla y amistosa.

En el caso de que sea la primera vez que se utiliza el hub central, o en caso de error de conexión con la red Wi-Fi o el bróker, el firmware pide al usuario que se acceda al modo configuración para configurar los parámetros. La forma de entrar en modo configuración es sacar el jumper que aparece en la Ilustración 71 y forzar el reinicio del programa mediante el botón “BOOT” de la placa. A partir de ahí, se accede a un menú mediante la terminal que permite de forma sencilla y guiada configurar el hub central.

- *Funcionamiento del firmware principal*

El firmware principal se encuentra en el archivo “main.py”. Este gestiona la comunicación UART con el CC1310, el modo configuración, la comunicación MQTT a través de Wi-Fi y la conexión con el servicio NTP. Para entender el funcionamiento se hace uso de la Ilustración 76 que es un diagrama de flujo del programa.

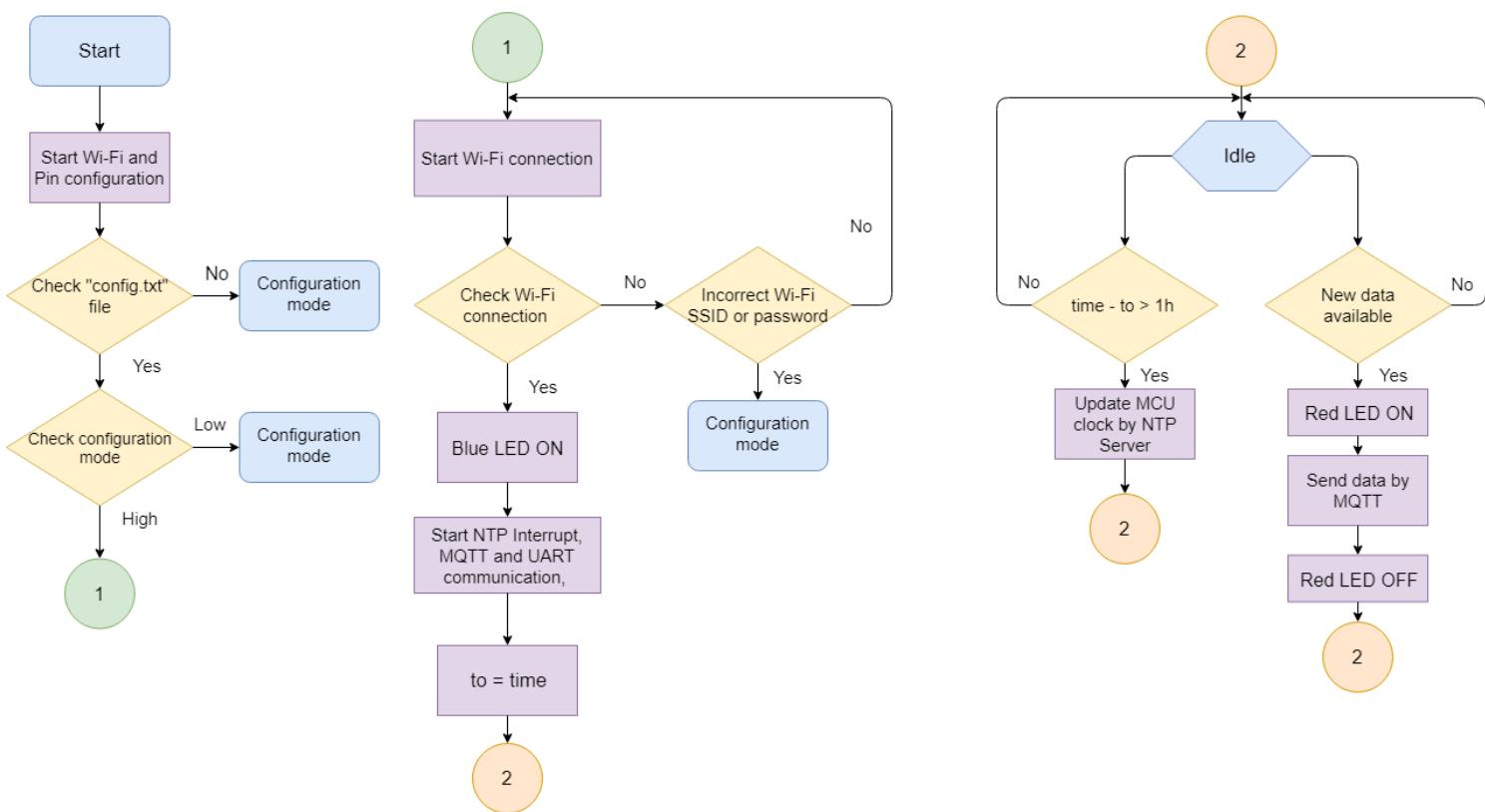


Ilustración 76 - ESP 32 firmware flowchart

En el diagrama se ven cada una de las acciones y fases por las que pasa el programa. En cuanto la placa del hub central se conecta a la corriente, el ESP 32 activa el módulo Wi-Fi y configura los pines de entrada para el modo configuración y los de salida para los LED que marcan el estado del programa.

Después de estas instrucciones iniciales comprueba si existe y es correcto el archivo de configuración “config.txt”. En caso negativo lo crea y pasa a modo configuración para que el usuario introduzca los valores. En caso afirmativo, pasa al siguiente estado en el que comprueba si el modo configuración está activado. Para ello comprueba el valor en el pin que posee el jumper de configuración. Si esta puesto el jumper, corresponde al estado lógico “High” y por tanto pasa al siguiente estado. En caso de ser estado “Low” entra en modo configuración.

Esta primera parte sirve para configurar el hub central. En la siguiente parte se inicializan los servicios. Se comienza intentando conectarse a la red Wi-Fi proporcionada. En el caso de que no consiga conectarse debido a problemas en la red, vuelve a intentarlo hasta conectarse. En caso de que sea debido a que no existe la red WiFi o que la contraseña incorrecta, sugiere entrar en modo configuración.

Si consigue conectarse a la red, se enciende el LED azul dando paso a la inicialización del cliente MQTT, la comunicación UART con el MCU CC1310 y activa un contador o timer de una hora de duración que funciona de forma periódica. Este timer

activa una interrupción cada hora que cambia el valor de la variable “interrupt_flag” a valor “True” respecto al valor por defecto que se le da en el siguiente bloque “False”. También se guarda en la variable “to” el valor actual del tiempo que ha transcurrido desde que se ha encendido el MCU.

Una vez se tienen todos los servicios inicializados y configurados correctamente, el firmware entra en estado de espera o “idle”. En este estado se queda el MCU hasta que uno de los dos posibles eventos ocurra.

El primero está relacionado con el timer. Cuando el timer lleva una hora funcionando produce una interrupción que cambia el valor de la variable “interrupt_flag” como ya se ha descrito. En la Ilustración 77 se puede observar el diagrama de flujo de la interrupción del timer.

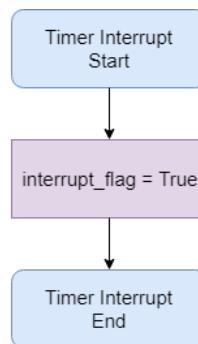


Ilustración 77 - ESP 32 timer interrupt flowchart

Al tener la variable el estado lógico “True” pasa a la siguiente fase en la que asigna el valor lógico “False” a la variable para que solo se cumpla la condición cuando el timer vuelva a producir la interrupción. Después actualiza el reloj interno del MCU al realizar una petición de la hora al servidor NTP y por último vuelve al estado “Idle”.

El otro evento está relacionado con la llegada de datos de las placas de sensores a través de la conexión UART con el MCU CC1310. Cuando un nuevo dato llega, el LED de color rojo se enciende simbolizando que se está procesando el mensaje. El mensaje que llega a través de UART es diferente en función de la placa de la que proceda. En el caso de ser de la placa de sensores ambientales será el primer mensaje que aparece en el Código 10. En el caso de ser de la placa de consumo de potencia será el segundo mensaje. Ambas son cadenas de caracteres. Los valores numéricos del Código 10 no corresponden a valores reales

```

'{"t":25,"p":1013,"h":60,"bat": 50,"id":"001"}'
'{"w":14061,"v":238.8099,"i":58.9229,"f":50,"a":2.1699,
 "id":"001"}'
  
```

Código 10 - ESP 32 entry message

Estos mensajes están escritos en formato de texto “JSON”. Se trata de un formato de texto muy extendido que se aplica en el intercambio de datos, usualmente entre cliente y servidor. Gracias a su popularidad, existen herramientas para el manejo de este formato de texto en los lenguajes de programación principales. Por lo tanto, Python, y en este caso MicroPython, permite generar a partir de la cadena de caracteres un objeto de tipo diccionario que facilita la gestión de los datos. De esta forma, se pueden añadir y eliminar elementos a la cadena simplemente conociendo la clave del diccionario.

En los mensajes aparecen diferentes claves que representan las variables medidas por las placas. Estas claves significan:

- “t”: Temperatura (interior o exterior)
- “p”: Presión ambiental.
- “h”: Humedad relativa.
- “bat”: Porcentaje de batería restante de la placa.
- “id”: Número de identificación de la placa que corresponde con su número MAC.
- “w”: Potencia eléctrica calculada.
- “v”: Tensión de la red.
- “i”: Corriente del circuito.
- “f”: Frecuencia de la red.
- “a”: Ángulo de desfase entre tensión y corriente.

A ambos mensajes es necesario añadirles el tiempo en el que se ha realizado la medición que se incorporará como clave “time” en el mensaje. Sin embargo, este tiempo en verdad no es cuando se ha realizado la medición, si no que corresponde al momento en el que se recibe dicho mensaje. A pesar de ello, los mensajes suelen ser instantáneos, por lo que el error es imperceptible. Este valor de tiempo está representado en Unix Timestamp, es decir, el número de segundos que han transcurrido desde la fecha 1/1/1970 a las 0:00:00. Se trata de una representación muy extendida en muchos sistemas computacionales.

También es necesario eliminar en ambos mensajes una de las claves antes de que sean enviados a través de MQTT. Este valor es el que se representa por “id”, que es enviado en el tema del mensaje de MQTT para distinguir una placa de otra. Por lo tanto, los mensajes que se envían por MQTT son de la forma:

```
'{"t":25,"p":1013,"h":60,"bat": 50,"time":1564504091}'  
'{"w":14061,"v":238.8099,"i":58.9229,"f":50,"a":2.1699,  
"time":1564504091}'
```

Código 11 - ESP 32 exit message

Para enviar los mensajes, el cliente MQTT los publica bajo un tema que tiene que ser conocido por el servidor. En el tema se incorpora información acerca de la ubicación del hub central, es decir, el edificio y planta en el que se sitúa, y el número de identificación de la placa de sensores (“id”):

“Home_efficiency_project/gateway/<building>/<floor>/<id>”

Código 12 - MQTT publish topic

Una vez visto el funcionamiento del firmware principal, solo queda analizar el funcionamiento del modo configuración. A este modo se puede acceder de varias formas. En caso de que no exista archivo de configuración o sus valores sean incorrectos, en caso de que el jumper de configuración este quitado o en caso de que la contraseña de la red Wi-Fi sea incorrecta. De nuevo se va a hacer uso de un diagrama de flujo en la Ilustración 78 para mostrar el funcionamiento de esta parte.

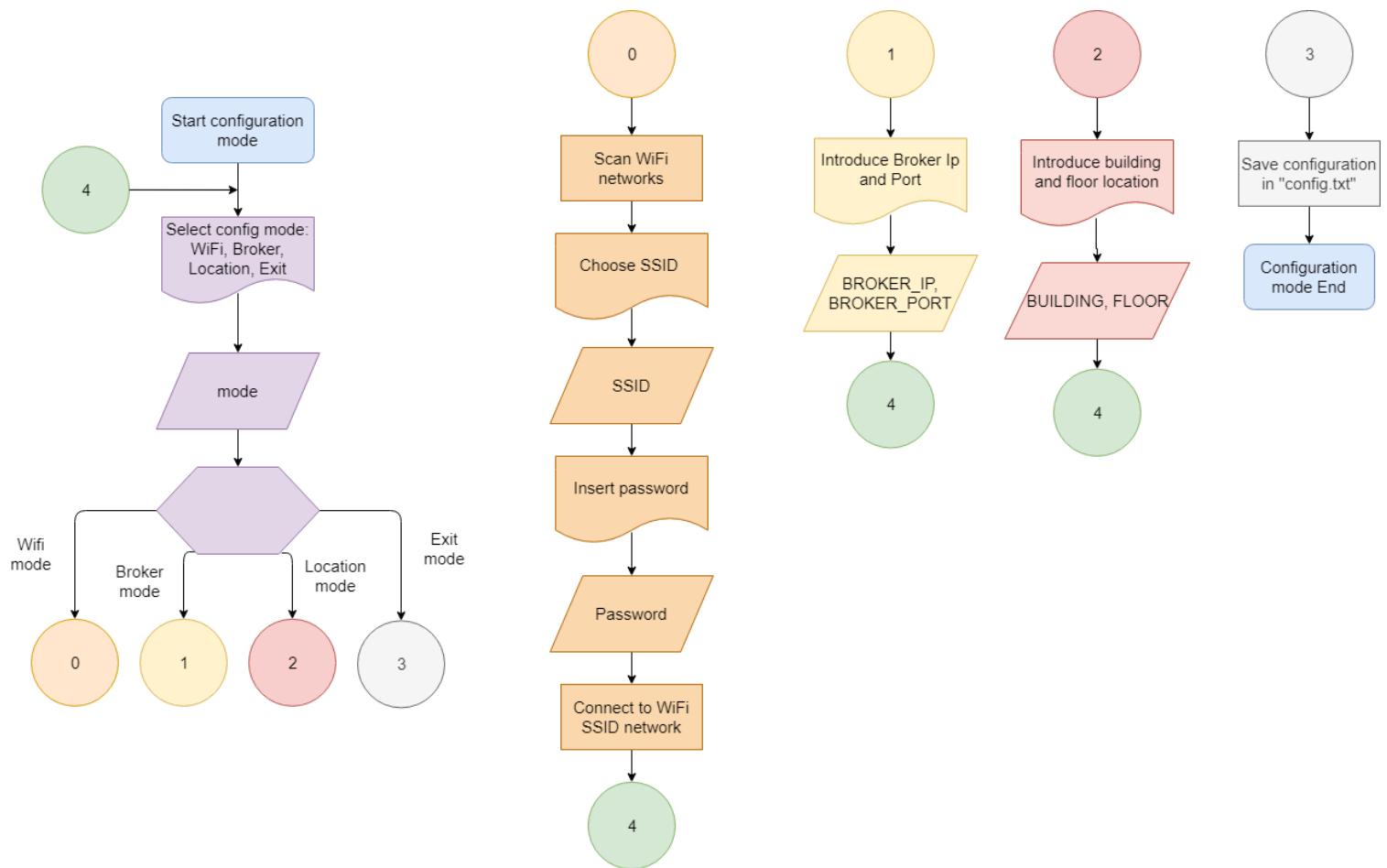


Ilustración 78 - ESP 32 configuration mode flowchart

El modo de configuración permite modificar la información almacenada en el archivo "config.txt" que se ha visto en este apartado. De esta forma en función del modo que se introduzca permite modificar unos parámetros u otros. En el caso de configurar la red WiFi, primero escanea las redes WiFi que se encuentran en el lugar. Después pide al usuario que introduzca la red a utilizar y su contraseña. Con esto, intenta conectarse a la red, mostrando por pantalla el resultado de la operación y vuelve al menú principal.

El modo "Broker" y "Location" funcionan de forma similar. El primero permite introducir la IP y el puerto del bróker MQTT con el que se establece conexión. El segundo permite introducir el edificio y piso en el que se encuentra el hub central. El último modo es el de salida, por lo que se limita a guardar en el archivo de configuración los cambios y reiniciar el dispositivo.

Las interacciones referentes a configuración se llevan a cabo a través de comunicación serial entre un ordenador y el MCU ESP 32 por USB, para lo que se puede utilizar el software PuTTY o ESPlorer.

5.2.2 Servidor

Una vez el mensaje es publicado a través de MQTT, se pasa a la siguiente capa en la arquitectura de las plataformas IoT. Se deja atrás la capa Fog para entrar en la capa Cloud, donde se encuentra el servidor y la información se puede monitorear y procesar.

- *Comunicación Gateway - Servidor*

Para llevar a cabo la comunicación MQTT entre el Gateway y el servidor, es necesario utilizar un Broker. En la Ilustración 79 se puede observar un diagrama con las partes que conforman esta comunicación.

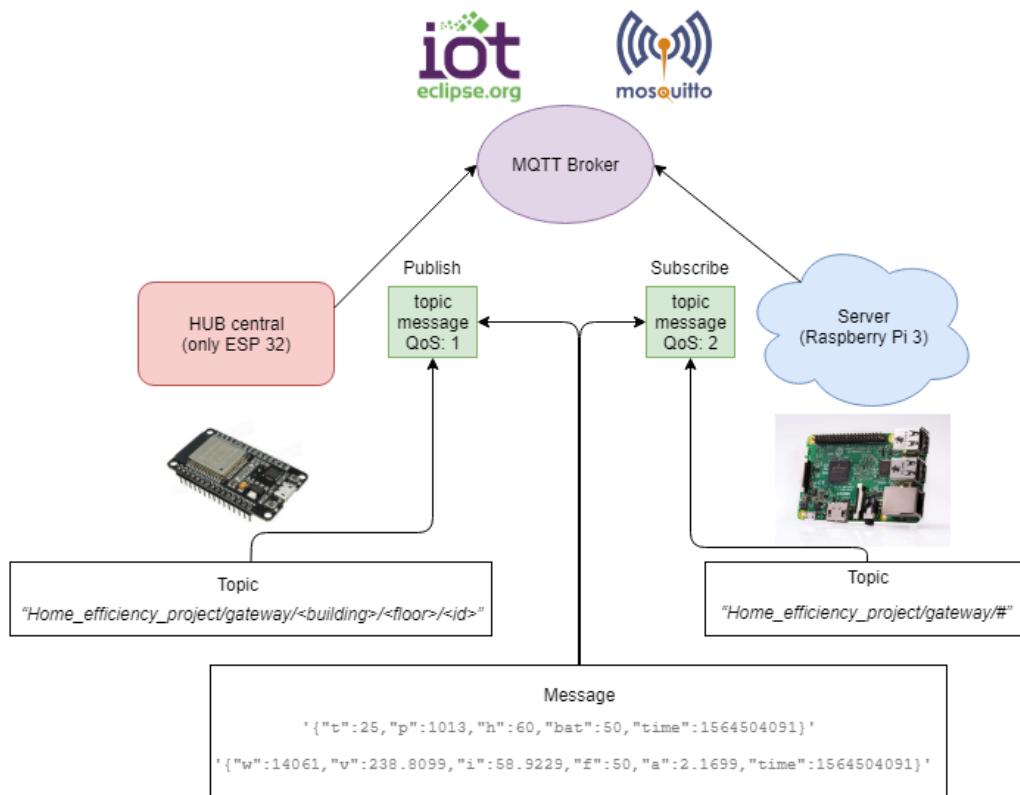


Ilustración 79 - Hardware - Software MQTT communication

Como se observa, el mensaje que se vio en el apartado anterior es publicado por el ESP 32 bajo el tema que también se vio en el apartado anterior. Debido a que el módulo de MicroPython no soporta QoS de nivel 2, se tiene que publicar con nivel 1.

Como Broker se utiliza el proporcionado de manera gratuita por iot.eclipse.org, que se basa en Mosquitto. Sin embargo, debido a ser gratuito y libre, no dispone de todas las características de Mosquitto. La más relevante es que no permite conexiones durables, esto es, una vez se desconecta el publisher o el subscriber, el Broker no guarda información de ellos. Esto es un inconveniente ya que supone mayores tiempos de conexión y reconexión y, por lo tanto, mayor consumo en el proceso. Otra desventaja es que a pesar de que la comunicación puede cifrarse, los temas son públicos. Esto hace que cualquier persona pueda acceder a ellos.

La solución a estos problemas pasa por instalar un Broker propio en un servidor o utilizar uno de pago. Debido a que en el proyecto se busca minimizar costes y mostrar una plataforma funcional, se considera esta función fuera del alcance y por tanto en líneas futuras.

Por el otro lado tenemos el servidor, para el que se utiliza una Raspberry Pi 3 como ya se había adelantado. Esta funciona como subscriptor a un tema genérico para así recibir los mensajes del hub central. Este tema se puede ver a continuación:

"Home_efficiency_project/gateway/#"

Código 13 - MQTT subscribe topic

La diferencia con el tema del publisher es que se eliminan los campos variables y a cambio se añade "#". Este símbolo, junto con "+", en los temas representan en MQTT una carta comodín o wild card. En el caso de "#" significa que el subscriptor se subscribe a cualquier tema que tenga los niveles superiores iguales desde el "#". Por lo tanto, aunque cambie el id o la localización del hub central, el servidor recibe el mensaje.

El mensaje es el mismo que se ha publicado en el tema, pero en este caso varia el QoS. El subscriptor tiene nivel 2, mientras que el publisher tiene nivel 1. Esto hace que el mensaje sea entregado al subscriptor con nivel 1. Sin embargo, se ha escogido el nivel 2 debido a posibles mejoras futuras del proyecto en el que se pueda subir el nivel del publisher, o incluso añadir nuevos publishers que sí permitan el nivel 2.

- *Partes del servidor*

El servidor es la capa Cloud de la arquitectura de las plataformas IoT. En ella se lleva a cabo una serie de operaciones con la información recibida del exterior con el objetivo de monitorizar dicha información. Es por ello por lo que el servidor se compone de diferentes partes bien diferenciadas.

Como ya se ha explicado en el capítulo 4, apartado 4.2.1.4 y posteriormente con mayor detalle en el apartado 4.2.2.2, cada una de las partes puede trabajar independientemente de las otras, es decir el fallo de una de ellas no provoca el colapso de las restantes. Entre ellas simplemente deben de conocer las APIs que permiten acceder a sus recursos. Esto se logra mediante una comunicación HTTP bajo la arquitectura REST entre las partes.

Para entender cómo funciona el servidor se va a hacer uso del diagrama de la Ilustración 80, en el que se puede visualizar cada una de sus partes que se explican a continuación.

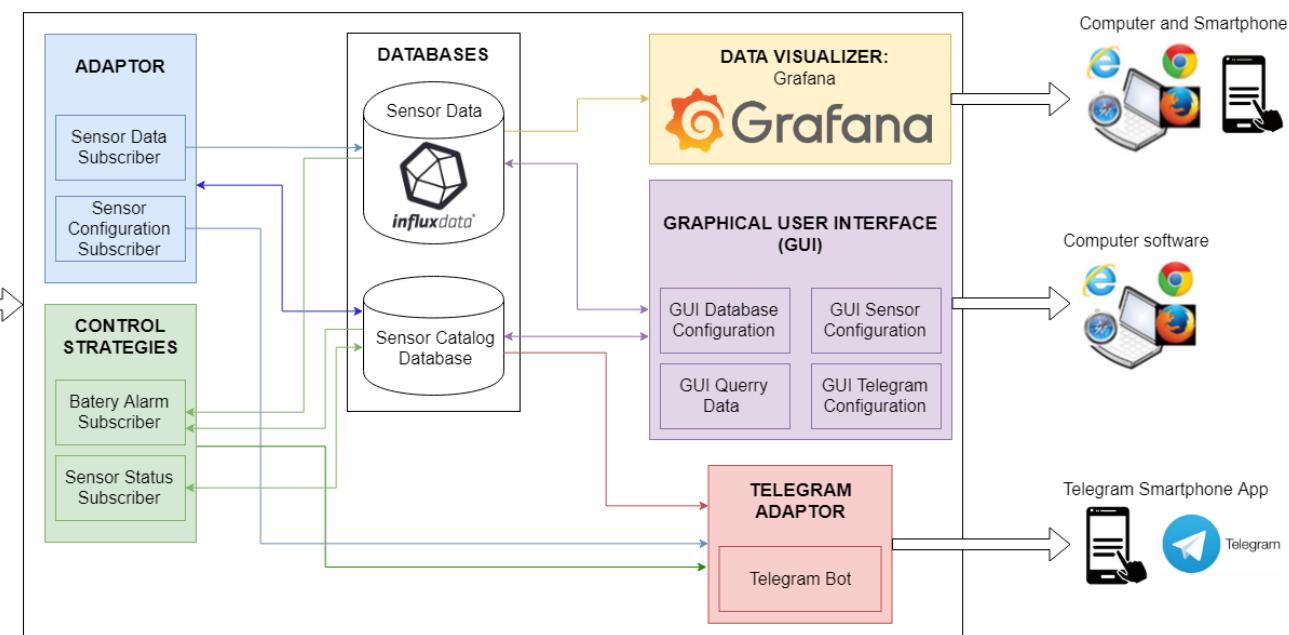


Ilustración 80 - Server architecture diagram

Como se puede observar, se dispone de 6 partes diferenciadas que cumplen funciones diferentes. Fuera del servidor se tiene a dos actores:

El primero sería el Broker MQTT por el que se reciben los mensajes del hub central tras subscribirse el servidor al tema como se ha visto en el punto anterior.

El segundo sería el usuario que dispone de varias herramientas para comunicarse con el servidor. Para visualizar y monitorizar los datos puede hacer uso de un navegador web desde un ordenador o dispositivo inalámbrico como teléfono móvil o Tablet. Para configurar los parámetros del servidor y extraer los datos almacenados es necesario disponer del software adecuado, por lo que tiene que realizarse desde un ordenador. Por último, para recibir notificaciones acerca del estado de los sensores puede hacer uso de la aplicación móvil y para ordenadores Telegram.

Si se comienza a describir las partes del servidor desde que se recibe el mensaje hasta que lo obtiene el usuario se tiene:

- Adaptador: Este bloque se encarga de transformar el mensaje que llega por MQTT al lenguaje que el servidor pueda entender. Este será almacenado en las bases de datos. Por un lado, las medidas obtenidas por los sensores y por otro lado la configuración básica del sensor. Además, se comunica con el Bot de Telegram para comunicar sobre nuevos sensores en la plataforma.
- Bases de datos: En ellas se almacena la información de los sensores. Se dispone de dos bases de datos. La primera se usa Influx Database para almacenar las medidas obtenidas por los sensores. La segunda es un archivo de texto plano con formato JSON que almacena la configuración básica de los sensores. Es la parte principal del servidor, ya que todos los bloques se comunican con ellas.
- Visualizador de datos: Permite la visualización gráfica de los datos. Se ha hecho uso de Grafana como servicio web para llevar a cabo esta tarea. Permite al usuario monitorizar el valor actual e históricos de las diferentes variables medidas.
- Interfaz gráfica de la plataforma: Permite visualizar de forma gráfica el estado de la plataforma. A diferencia de la anterior, esta muestra información sobre el estado y configuración de los sensores, de la base de datos y del Bot de Telegram. Además, permite extraer de forma sencilla y amistosa los datos obtenidos por los sensores para posterior análisis.
- Estrategias de control: Este bloque se encarga de avisar al usuario a través del Bot de Telegram del estado de los sensores y del nivel de batería de estos. Estos avisos se realizan a través del Bot de Telegram y permiten al usuario tener de manera temprana el estado de los sensores.
- Adaptador de Telegram: Se encarga de transformar los mensajes del servidor a un lenguaje que el Bot de Telegram comprenda para poder recibir avisos del estado de los sensores.

En los próximos apartados se entra en detalle acerca del funcionamiento de cada uno de estos bloques y las interacciones que tienen entre ellos.

En la Ilustración 80 se pueden ver estas interacciones entre los bloques. Cada una de las flechas responden a peticiones en formato REST entre los bloques. Las peticiones se realizan desde los diferentes bloques a las bases de datos, por lo que se toma ese bloque como referencia. De esta forma, si son entrantes al bloque base de datos representan una petición que modifica la base de datos, es decir, POST, PUT o DELETE. Si por el contrario salen de la base de datos representa una petición de información, es decir, GET.

Solo existen dos bloques que no se comunican directamente con la base de datos, y es debido a que es una petición directa al Bot de Telegram. Estas se usan para alertar al usuario de que algo ha sucedido con los sensores que no se esperaba o alertar de batería baja. Por lo tanto, no es necesario almacenar el aviso en la base de datos.

- *Software requerido*

En este punto se van a explicar que programas son necesarios para el funcionamiento del servidor. Todos los programas ya se han explicado en el capítulo 4, apartado 4.3.2, en el que se analizaban los softwares utilizados.

El servidor se aloja en una Raspberry Pi 3 por lo tanto lo primero que se necesita es instalar el sistema operativo. Para ello se utiliza el sistema operativo recomendado por el fabricante, Raspbian Stretch Lite. Se ha escogido la versión Lite, es decir, sin interfaz gráfica para reducir el consumo de recursos. Sin embargo, no hay ningún problema en utilizar la versión con interfaz gráfica. Para llevar a cabo la instalación se recomienda seguir los pasos que se detallan en la página del fabricante (Raspberry Pi Foundation, 2019), o en páginas de aficionados como la siguiente: (Mackenzie, 2017).

Es necesario habilitar la comunicación SSH para el acceso remoto desde otro ordenador y se recomienda el uso de programas como FileZilla que permiten la comunicación cliente servidor mediante el protocolo FTP para el intercambio de archivos.

Con el sistema operativo viene instalado por defecto Python en su versión 3. Sin embargo, algunos de los módulos o paquetes de Python que se utilizan en el proyecto no vienen instalados. Para ello, se puede usar el gestor de instalación de paquetes de Python que se instala con el siguiente comando:

```
$ sudo apt-get install python3-pip -y
```

Código 14 - Python 3 installation Linux command

Una vez instalado el gestor de instalación, se pueden instalar los paquetes con el siguiente comando:

```
$ sudo pip3 install <nombre_del_paquete>
```

Código 15 - Python 3 modules installation Linux command

Cuando se analicen los bloques del servidor, se verán los paquetes que son necesarios instalar.

Para el almacenamiento de los datos de los sensores y la visualización será necesario instalar la base de datos Influxdb y el visualizador Grafana. Esto se explicará con mayor detalle en cada apartado.

Durante el proceso de desarrollo del servidor, en vez de utilizar la Raspberry Pi se utiliza un ordenador convencional con el sistema operativo Windows 10. Esto se debe a que se está más familiarizado con el sistema operativo Windows, además de agilizarse el proceso de desarrollo de la plataforma al disponer de interfaz gráfica y mayor capacidad computacional. Por lo tanto, posteriormente se tiene que exportar todos los códigos y programas al sistema operativo Raspbian.

5.2.3 Servidor: almacenamiento

Se trata del núcleo central del servidor, donde se almacenan los datos de la plataforma. Por lo tanto, todos los bloques necesitan de las bases de datos para funcionar. Es por esto por lo que a pesar de no ser el primer bloque en el flujo natural de los mensajes se va a analizar con antelación.

Como se observa en la Ilustración 80, el almacenamiento de los datos se realiza en dos bases de datos diferentes. Esto se debe a que cada una de ellas gestiona diferentes tipos de datos.

Por un lado, se tiene la base de datos de Influx database, que está especializada en la gestión de datos de series temporales. Esto es, datos que tienen ligada una referencia de cuando se realizaron. Por lo tanto, es ideal para el almacenamiento de las medidas obtenidas por los sensores.

Por otro lado, es necesario almacenar información acerca de los sensores para tener un control de que dispositivos están conectados y como están funcionando. Esta base de datos se basa en un archivo de texto plano, en la que los datos se almacenan en formato JSON. Existen alternativas más elaboradas, pero esta permite de una forma sencilla y rápida disponer de un control de los dispositivos de la plataforma.

A continuación, se explican las dos bases de datos con mayor detalle:

- *InfluxDB*

Como ya se ha explicado, InfluxDB está especializada en almacenar datos de series temporales. Esto permite hacer búsquedas rápidas mediante etiquetas de tiempo. Además, permite añadir etiquetas a cada dato que se guarda, de modo que permite búsquedas avanzadas.

Para almacenar información en InfluxDB es necesario conocer cómo interpreta los datos y en qué formato se tienen que presentar para poder introducirlos o extraerlos. InfluxDB denota a los datos que se quieren introducir como valores de campo o “field values” que tienen asociado un instante de tiempo o “timestamp” y una clave de campo o “field key” que son únicos. Al par field value y field key se considera conjunto de campo o “field set”. La field key del valor no es indexable, esto es, no está optimizada para búsquedas, por lo que es preferible no utilizar palabras clave. (InfluxDB Corp., 2019)

Junto con el field value pueden ir asociadas etiquetas o “tags” que permiten añadir más información al valor. De nuevo las tags cuentan con un “tag key” y un “tag value” que conforman un “tag set”. A diferencia del field key, las tags sí son indexables, lo que permite realizar búsquedas más rápidas. Es por esto por lo que se aconseja utilizar tags como información adicional.

Por encima de todo lo anterior están las medidas o “measurements”. Estas funcionan como contenedor de todos los datos, campos y etiquetas anteriores. Las measurements describen el tipo de datos que han sido almacenados en los correspondientes fields.

Por lo tanto, InfluxDB considera punto o “point” a un único dato que posee measurement, tag set, field set y timestamp. Un punto es único en la base de datos. En la Ilustración 81 se puede observar un ejemplo de lo que se acaba de describir para entender los conceptos. El conjunto de datos que aparece en la ilustración es inventado y no guarda relación con el proyecto.

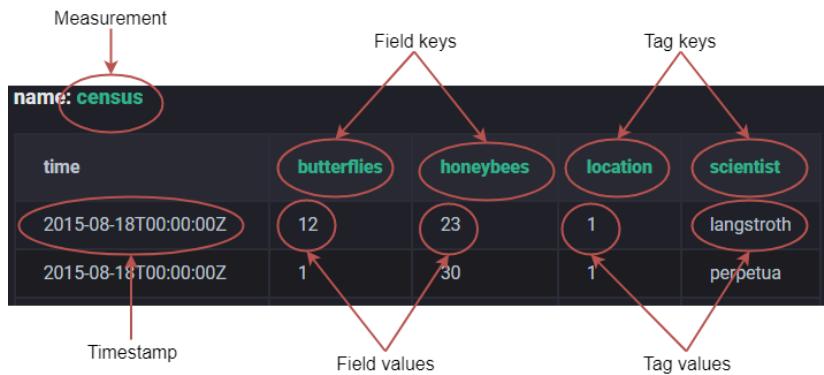


Ilustración 81 - InfluxDB data type

Todos estos tipos de datos están contenidos a su vez dentro de una base de datos. Esto quiere decir que dentro de InfluxDB se puede disponer de diferentes bases de datos. Serían como contenedores de diferentes series temporales, pero todos siguen la arquitectura de InfluxDB. Esto permite tener copias de una misma base de datos en caso de que ocurra algún problema, o poder realizar diferentes proyectos con el mismo servicio de InfluxDB.

Para acceder al servicio de almacenamiento de InfluxDB y a las bases de datos que este pueda tener, se crean usuarios. Por defecto el servicio de InfluxDB está en modo público. Esto es, que cualquier petición que se realice al servicio será aceptada. Para evitar peticiones indeseadas, se debe poner en modo privado modificando el archivo de configuración. Además, se debe crear un usuario al que se le otorgan privilegios para acceder. Existen cuatro tipos de privilegios:

- “Admin”: Es el privilegio de administrador. Permite tanto la escritura como lectura de los datos de las diferentes bases de datos, además de poder configurar y gestionar el servicio de InfluxDB. Puede crear, modificar y eliminar bases de datos, usuarios, datos o configuraciones.
- “Write”: Privilegio de escritura en una base de datos. Permite al usuario añadir datos en una base de datos específica.
- “Read”: Privilegio de lectura. Permite al usuario leer datos de una base de datos específica.
- “All”: Privilegio de escritura y lectura. Se trata de la suma de los dos anteriores.

No se debe confundir “Admin” con “All”. El usuario “Admin” además de poder escribir y leer de todas las bases de datos del servicio InfluxDB puede modificar el servicio. Sin embargo, el usuario con “All” está restringido a la escritura y lectura de datos en la base de datos en el que se le haya concedido ese privilegio. Además, no puede modificar el servicio de InfluxDB.

En la Tabla 12 se puede observar un ejemplo real de un point del proyecto, que sigue la misma organización que los datos de la ilustración anterior. En este caso dispone solo de un field key que es “value” y el resto son tag keys.

Temperature

time	value	ID	Building	Floor	Room
2019-07-28T00:00:00Z	25	0343GR34	Serra_28	6	dormitory

Tabla 12 - InfluxDB data type project example

En el proyecto se ha escogido estructurar los datos de la forma que aparece en la Tabla 12. Dentro de un “measurement” solo se dispone de una “field key” que es la que

aparece con el nombre de “value”. Como este campo no está optimizado para búsquedas, se ha escogido un nombre. Se dispone de cuatro etiquetas que son: “ID”, “Building”, “Floor” y “Room”. Estas permiten filtrar valores en función del identificador del sensor y de la ubicación del mismo. Todos los “measurements” disponen de los mismos “fields key” y “tags key”.

Existen varios “measurements”, tantos como variables de interés que los sensores puedan medir. Por lo tanto, se tiene: “Temperature”, “Humidity”, “Pressure”, “VOC”, “Batery” y “Electric power”.

De esta forma es sencillo obtener rápidamente los valores de varias variables en un periodo de tiempo concreto y filtrando según la estancia que se quiera analizar o en función de determinados sensores.

Para poder introducir, extraer y gestionar la base de datos es necesario realizar peticiones a través de HTTP. También se puede realizar a través de la línea de comandos. Sin embargo, la primera forma hace que el proceso sea más complicado al ser peticiones de bajo nivel y la segunda no permite automatizar el proceso. Es por eso por lo que existe una API especial para Python que permite realizar todas las operaciones.

Esta librería de Python dispone de numerosos comandos que permiten la gestión completa de la base de datos. Sin embargo, existen tres comandos que son de vital importancia y por lo tanto se van a describir con más profundidad. Estos son:

- Objeto que permite la conexión cliente-servidor con InfluxDB:

```
>>influxdb.InfluxDBClient(host=u'localhost',port=8086,username=u'root',  
    password=u'root',database=None,ssl=False,verify_ssl=False,timeout=None,  
    retries=3,use_udp=False,udp_port=4444proxies=None,pool_size=10,path  
    =u'')
```

Código 16 - Influxdb client Python instance

Esta instrucción permite generar un objeto en Python que mantiene la conexión con InfluxDB. Gracias a esto se puede interaccionar con la base de datos para añadir, extraer, o modificar datos, usuarios o bases de datos. Los parámetros más relevantes que se han utilizado en el proyecto son:

- host: dirección IP donde se aloja el servicio InfluxDB.
- port: puerto donde se aloja el servicio InfluxDB.
- username: nombre de usuario para acceder a InfluxDB.
- password: contraseña del usuario.
- database: nombre de la base de datos a la que acceder.

Además de estos parámetros existen otros que permiten ajustar el cifrado ssl si se desea aumentar la seguridad de la conexión, así como utilizar el protocolo UDP en vez del TCP.

- Insertar nuevos datos:

```
>>write_points(points,time_precision=None,database=None,retention_policy=None,tags=None,batch_size=None,protocol=u'json')
```

Código 17 - Insert Influxdb data series command

Con este comando se insertan múltiples series de datos. Para ello es preciso que el parámetro “points”, que es el único relevante, tenga un formato adecuado. Para ello se hace uso del formato JSON para enviar el mensaje. A continuación, se muestra un ejemplo de cómo debe ser el formato:

```
[{
  "measurement": <name_measurement>,
  "tags": {
    <tag_key>: <tag_value>,
    <tag_key>: <tag_value>
  },
  "time": "2009-11-10T23:00:00Z",
  "fields": {
    <field_key>: <field_value>,
    <field_key>: <field_value>
  },
  {
    ...
  }
}]
```

Código 18 – InfluxDB insert data message format

Como se puede observar se tiene que disponer siempre de un “measurement”, del “time” y de al menos un “field”. El resto de los campos se añaden según se deseen. Además, se pueden añadir más puntos en el mismo mensaje.

- Extraer datos:

```
>>query(query,params=None,epoch=None,expected_response_code=200,database=None,raise_errors=True,chunked=False,chunk_size=0,method=u'GET')
```

Código 19 - Query data from Influxdb command

Con este comando se pueden extraer puntos de la base de datos que se esté gestionando. Para ello es necesario introducir en el parámetro “query” una variable de tipo String que cumpla con un formato de lenguaje SQL. Aquí se van a exponer los casos que se han tratado en el proyecto que son: extracción de datos según un periodo de tiempo, etiquetas, campo y medida y eliminación de valores según los mismos atributos anteriores. A continuación, se expone la cadena de comando para extraer los datos. Para eliminarlos es la misma estructura, pero se cambia “SELECT” por “DELETE”.

```
SELECT <field_key>[,<field_key>,<tag_key>] FROM
<measurement_name>[,<measurement_name>] WHERE time >= <start_time> AND
time <= <end_time> AND <tag_key> = '<tag_value>' [ OR <tag_key> =
'<tag_value>']
```

Código 20 - SELECT SQL language instance

Los parametros referentes al tiempo de comienzo y finalización deben tener el siguiente formato: %Y%m%dT%H%M%SZ

Por último, se va a especificar que hace falta para poder ejecutar la base de datos en el servidor. Para ello es necesario descargar e instalar el servicio de InfluxDB desde la página del desarrollador en la cual se explica al detalle el procedimiento a seguir. (InfluxDB Corp., 2019). También es necesario instalar la API de InfluxDB para Python (aviau, areski, & georgijd, 2013). Para ello simplemente hay que ejecutar el siguiente comando:

```
$ pip3 install influxdb
```

Código 21 - Influxdb Python module installation Linux command

- Base de datos de la configuración de los sensores

En esta otra base de datos se guarda información básica de los sensores y de la plataforma. Al no ser información que lleve asociada una referencia temporal no es posible usar InfluxDB para almacenarla.

En esta base de datos hay dos tipos de datos. Por un lado, está la configuración básica de la plataforma, es decir, direcciones, puertos e identificadores que necesitan los diferentes bloques del servidor para establecer conexión. Por otro lado, se almacena información de los sensores que se han conectado a la plataforma.

Gracias a esta base de datos se puede implementar una arquitectura de microservicios, en el que cada microservicio o bloque solo tiene que acceder a esta base de datos para saber con quién debe comunicarse y además permite tener un seguimiento de los sensores. Es por esto por lo que se refiere a esta base de datos como catálogo.

El catálogo no es más que un archivo de texto plano en formato JSON. A continuación, se muestra el catálogo utilizado en el proyecto en el estado inicial de la plataforma, es decir, cuando no se ha añadido ningún sensor:

```
{
    "telegram": {
        "BOT_token": "846839523:AAFxvvhqwM1U6wB06vcEQgAhCRBLInE9fDI",
        "chat_id": "-398477790"
    },
    "database": {
        "IP": "localhost",
        "port": 8086
    },
    "broker": {
        "IP": "iot.eclipse.org",
        "port": 1883,
        "sub_topic": "Home_efficiency_project/gateway"
    },
    "devices": [],
    "grafana": {
        "IP": "localhost",
        "port": 8080
    }
}
```

Código 22 - Initial catalog file

En este fragmento se puede observar la configuración de varios bloques de la plataforma. Primero se tiene la identificación y el token del Bot de Telegram. Segundo la dirección IP y puerto de la base de datos InfluxDB. Tercero la dirección IP, puerto del bróker y la raíz del tema al que deben suscribirse para recibir mensajes. Cuarto, la información acerca de los sensores, que al no haber ninguno en la plataforma está vacía la lista. El último elemento es la dirección IP y puerto de Grafana.

El motivo por el cual la raíz del tema se encuentra almacenada en el catálogo, es que, si se modifica el tema en la parte Hardware, solamente habrá que modificar el tema en este archivo, y no en todos los subscriptores, ya que es una información compartida por todos ellos.

En el caso de que se añada algún sensor a la plataforma, se añadiría a la lista de la clave “devices”. Aquí se muestra un ejemplo en el cual se ha añadido un sensor ambiental a la plataforma. Se ha omitido el resto del catálogo debido a que esa parte se mantiene estática.

```

    "devices": [
        {
            "status": true,
            "available_resources": {
                "Pressure": "Pa",
                "Humidity": "%",
                "Temperature": "C",
                "Batery": "%"
            },
            "frequency_measure": 0.0,
            "timestamp": 1563815712,
            "location": {
                "building": "None",
                "room": "Nan",
                "floor": "None"
            },
            "device_ID": "00124b000e017202"
        }
    ]

```

Código 23 - Sensor register in the catalog

Como se puede observar, se almacena información acerca del sensor entre la que se encuentra:

- “status”: simboliza el estado del sensor en la plataforma. “True” si está activo, es decir, enviando datos, “False” si está desconectado.
- “available resources”: Se trata de las medidas que proporciona el sensor, junto con sus unidades.
- “frequency_measure”: Periodo con el que realiza medidas el sensor. Por defecto es 0. El usuario debe introducir manualmente este valor mediante la GUI.
- “timestamp”: Ultima vez que el sensor envió información al servidor en formato Unix timestamp.
- “location”: Se trata de la ubicación del sensor. Tiene tres campos, “building” y “floor” que vienen definidos por la ubicación del hub central y “room”. Este último valor lo debe introducir el usuario mediante la GUI.
- “device_ID”: Se trata del identificador del sensor que va ligado a su MAC, por lo que es único.

De esta forma se irían añadiendo automáticamente los sensores al catálogo para disponer de una trazabilidad de los sensores.

Para que este archivo esté accesible por el resto de los bloques a través de peticiones HTTP, es necesario desarrollar una aplicación web que manipule el archivo. De este modo, se ha hecho uso de un paquete de Python llamado Cherrypy, que permite del desarrollo de aplicaciones web basadas en el formato REST de manera rápida y sencilla. Para ello, es necesario instalar el paquete con el siguiente comando:

```
$ pip3 install Cherrypy
```

Código 24 - Cherrypy Python module installation Linux command

Por lo tanto, este punto se compone de dos archivos. En la Ilustración 82 se puede observar ambos archivos. El “catalog.txt” es el catálogo que almacena la información de la plataforma y de los sensores como ya se ha visto. El “Sensor Catalog Database.py”

se trata del archivo que crea una aplicación web por la cual se puede acceder mediante peticiones REST al catálogo.

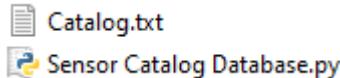


Ilustración 82 - Sensor Catalog Database files

Como ya se ha mencionado, para acceder a los datos del catálogo, es necesario hacer peticiones REST a la aplicación web. Por lo tanto, es necesario implementar en el archivo de Python las llamadas que se permiten hacer desde los otros bloques para acceder a la base de datos, como si de una API se tratara. Se han implementado diferentes funciones para acceder a los datos. En las siguientes tablas se muestran las más relevantes y utilizadas por los bloques del servidor.

Para acceder al catálogo es necesario conocer su dirección y puerto. Se ha escogido como dirección la propia Raspberry por defecto y el puerto por defecto que es:

`"http://localhost:80" + URI + parameters`

Código 25 - Catalog request example

GET	URI	parameters
1	/broker	No
2	/database	No
3	/grafana	No
4	/all	No
5	/telegram	No
6	/location	{"id":<sensor_id>}

Tabla 13 - Catalog GET requests

En la Tabla 13 se muestran las peticiones GET. En función de la URI que se introduzca se obtiene:

- 1: dirección IP, puerto del bróker y la raíz del tema para los subscriptores.
- 2: dirección IP y puerto de la base de datos InfluxDB.
- 3: dirección IP y puerto de Grafana.
- 4: devuelve una lista con todos los sensores que almacena el catálogo.
- 5: identificación del chat y token de Telegram.
- 6: edificio, planta y habitación del sensor.

POST	URI	Message body
1	/add	{...}

Tabla 14 - Catalog POST request

En la Tabla 14 se muestran la petición POST. Esta petición añade un nuevo sensor al catálogo. Para ello, se tiene que mandar en el cuerpo de la petición un diccionario con todos los parámetros necesarios del sensor. Debe de ser similar al que se observar en el texto introducido sobre el catálogo de la plataforma con clave "devices".

PUT	URI	parameters
1	/location	{"id":<sensor_id>,"building": <building>,"room": <room>,"floor": <floor>}
2	/timestamp	{"id":<sensor_id>,"timestamp":<sensor_timestamp>}
3	/status	{"id": <sensor_id>,"status":<True/False>}
4	/frequency	{"id":<sensor_id>,"frequency":<sensor_measure_period>}
5	/telegram	{"token":<telegram_token>,"chat_id": <chat_id>}
6	/resources	{"device_ID":<sensor_id>,"available_resources": [...]}

Tabla 15 - Catalog PUT requests

En la Tabla 15 se muestran las peticiones PUT. En función de la URI especificada, se actualiza la siguiente información en el catálogo:

- 1: el edificio, planta y habitación del sensor especificado.
- 2: la última vez que se conectó el sensor especificado.
- 3: el estado del sensor especificado.
- 4: el periodo con el que realiza medidas el sensor especificado.
- 5: la identificación del chat y token de Telegram.
- 6: los variables que mide el sensor especificado. La lista debe tener el mismo formato que se observa en el texto introducido sobre el catálogo de la plataforma con la clave “devices”, apartado “available_resources”.

DELETE	URI	parameters
1	/delete/one	{"id":<sensor_id>}

Tabla 16 - Catalog DELETE request

En la Tabla 16 se muestra la petición de DELETE. Esta permite eliminar el sensor especificado del catálogo.

Sin embargo, para realizar estas peticiones al catálogo es necesario que los bloques del servidor conozcan la dirección IP y puerto de este previamente. Para ello, es necesario incluir en el código de cada bloque esta información. Esto trae problemas ya que en caso de que la dirección IP o el puerto del catálogo cambien, será necesario modificar manualmente en todos los archivos estos datos. En las líneas futuras se trata este problema y se plantea como resolverlo.

5.2.4 Servidor: adaptador

Hasta ahora se ha analizado como recibe el mensaje el servidor y como lo almacena. Sin embargo, es necesario de un bloque intermedio que transforme el formato de entrada de los mensajes a uno con el que las bases de datos puedan trabajar. Este es el trabajo del bloque adaptador, que actúa de intermediario entre ambos.

Como se dispone de dos bases de datos diferentes se ha dividido el bloque en dos microservicios. Ambos son subscriptores del mismo tema, que reciben los mensajes a través de MQTT. No obstante, El servicio “Sensor Data Subscriber” de la Ilustración 80 se encarga de almacenar las medidas obtenidas por los sensores en InfluxDB, mientras que “Sensor Configuration Subscriber” almacena la información sobre los sensores en el Catálogo. Además, este último servicio se comunica con el bloque Adaptador de Telegram para informar al usuario si un nuevo sensor se conecta a la plataforma.

En la Ilustración 83 se puede observar los archivos que componen este bloque del servidor. Los dos primeros se corresponden a los microservicios descritos. El archivo

“MyMQTT.py” contiene un objeto de Python que permite y gestiona la comunicación MQTT de forma sencilla. Este objeto se utiliza para crear los subscriptores, por lo que cualquier bloque que reciba mensajes a través de MQTT lo necesita. El último archivo “TelegramBot.py” pertenece al bloque Adaptador de Telegram. De nuevo es un objeto de Python que crea y gestiona la comunicación con la aplicación de Telegram. Los bloques que interactúen con el Adaptador de Telegram necesitan de este archivo para poder comunicarse.

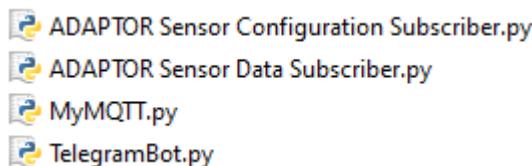


Ilustración 83 - Adaptor block files

Para que este bloque funcione es necesario instalar una serie de paquetes de Python. El que permite la comunicación con influxdb que ya se ha explicado, el que permite la comunicación con Telegram, que se verá en su apartado y el que permite realizar peticiones HTTP. Este paquete es “requests” y se instala de la siguiente forma:

```
$ pip3 install requests
```

Código 26 - Requests Python module installation Linux command

A continuación, se presentan con mayor detalle cada uno de los servicios del bloque Adaptador.

- *Subscriptor de los datos de los sensores*

Como ya se ha mencionado anteriormente, este servicio se encarga de transformar el mensaje entrante por MQTT a un formato que la base de datos InfluxDB pueda comprender. Para ello es necesario que se subscriba al tema en el que se envían los mensajes, se conecte con la base de datos y extraiga los datos de las medidas obtenidas por los sensores. Además de comunicarse con estos dos actores, es necesario que se comunique con el catálogo para obtener información de la plataforma y del sensor que ha recibido el dato.

Si se observa la Tabla 13, este servicio realiza las peticiones GET 1 y 2 para obtener la dirección IP, el puerto y el tema base tanto del bróker MQTT como de InfluxDB. Además, realiza la petición 6 para conocer la localización del sensor del que se ha recibido el mensaje. Esto es necesario, ya que cuando se añade el punto a InfluxDB se le incluyen las etiquetas de ubicación.

De la Tabla 15, realiza las peticiones PUT 1 y 2. La primera la realiza para actualizar la localización en el catálogo, en caso de que la localización del sensor del que se recibe el mensaje haya cambiado. La segunda actualiza la fecha y hora de la última conexión del sensor a la plataforma.

La última petición es la que realiza a la base de datos InfluxDB para almacenar el nuevo dato si todo es correcto. Para ello es necesario enviar un mensaje en formato JSON similar al que se detalló en el apartado anterior, en el Código 18.

Antes de comenzar el servicio es necesario configurar unas constantes en el código para que funcione correctamente. Estas son:

```

address_catalog = 'http://<url_catalog>:<port>'

CLIENT_ID = "<nombre del subscriptor_MQTT>"
QoS_msg = <nivel de servicio>
DB_USERNAME = '<nombre de usuario>'
DB_PASSWORD = '<contraseña de usuario>'
NAME_DB = "<nombre de la base de datos>"
```

Código 27 - Constants in ADAPTOR Sensor Data Subscriber file

El primero es la dirección del catálogo para obtener la información de la plataforma. Los dos siguientes son necesarios para el subscriptor MQTT. El primero es el nombre e identificador del subscriptor y el segundo el nivel de servicio. Los tres últimos tienen que ver con InfluxDB. Son el nombre de usuario y contraseña para acceder a InfluxDB y el nombre de la base de datos en el que se quiere almacenar los datos de los sensores.

Una vez se ha configurado correctamente, el programa se ejecutaría automáticamente introduciendo los nuevos valores a la base de datos. En la Ilustración 84 se muestra un diagrama de flujo que representa de forma resumida el funcionamiento interno del programa.

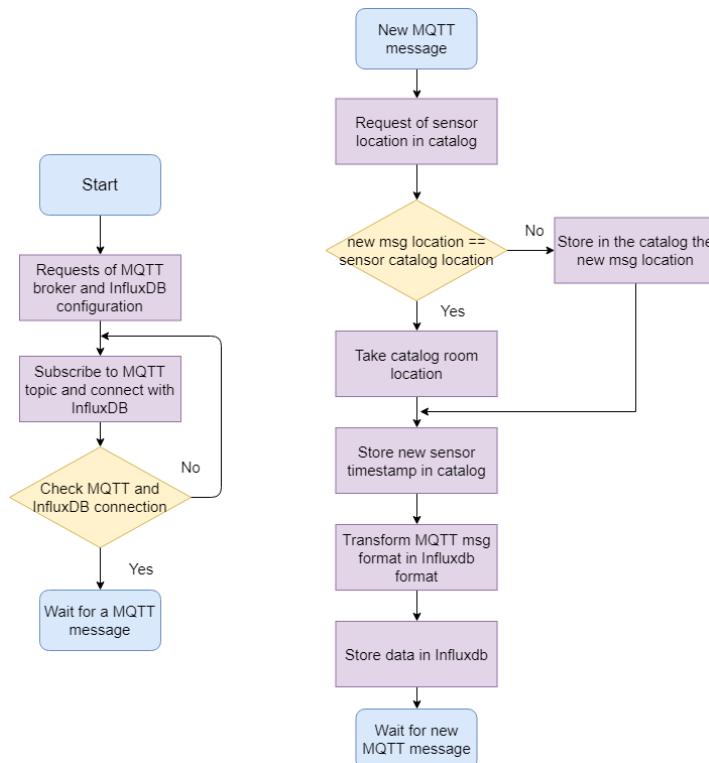


Ilustración 84 - Sensor Data Subscriber flowchart

El programa comienza con el diagrama de flujo de la izquierda. Primero realiza las peticiones al catálogo sobre el MQTT bróker, el tema de subscripción y la base de datos InfluxDB. Posteriormente, establece la comunicación MQTT suscribiéndose al tema y con InfluxDB. En caso de que no sea posible conectarse con alguno de los actores, se queda bloqueado en ese punto, ya que no tiene sentido continuar. Si el nombre de la base de datos no existe en InfluxDB, automáticamente se creará una nueva. Si todo es correcto, el programa se queda a la espera de que llegue algún mensaje a través del tema MQTT.

Cuando se recibe un mensaje se ejecuta el diagrama de la derecha. Como ya se vio en la Ilustración 79, en el tema viene la identificación del sensor y parte de la localización del mismo. En el apartado de almacenamiento se ha observado que la localización se compone de los atributos “building”, “floor” y “room”. Por lo tanto, falta conocer la habitación en la que se encuentra el sensor.

Por ello, se realiza una petición al catálogo para conocer cuál era la última ubicación del sensor almacenada. En caso de que los atributos de “building” y “floor” del nuevo mensaje sean iguales a los almacenados en el catálogo, se toma el atributo “room” del catálogo. En caso de que sean diferentes, se actualizará el catálogo con la nueva localización y se asignará al atributo “room” el valor nulo o “Nan”.

Este último caso hace que no sea posible conocer la ubicación exacta del sensor, lo cual es muy útil a la hora de monitorizar los datos. Para solucionarlo, el usuario deberá modificar manualmente mediante la GUI el atributo “room” del sensor. Una vez lo realice, el programa guardará esta información para el resto de los mensajes automáticamente. Esto se analiza con mayor detalle en el apartado destinado a la GUI.

Una vez se conocen todas las etiquetas y que valores mide el sensor, se procede a su almacenamiento. Previo a esto, se actualiza la fecha y hora de la última conexión del sensor en el catálogo.

Para poder almacenar los datos en InfluxDB es necesario cumplir con el formato requerido. Primero se transforma el tiempo de formato Unix timestamp al formato RFC 3339: %Y-%m-%dT%H:%M:%S, siendo de izquierda a derecha los valores de año, mes, día, hora, minuto y segundo de la medición. Posteriormente se estructura la información en formato JSON como el que se observa en Código 18. Y por último se realiza la petición de escritura de datos a InfluxDB con el Código 17. El Código 28 muestra un ejemplo real del proyecto de una medición de temperatura. En este código solo se muestra un punto, pero se puede almacenar más de un punto en una sola petición si se añaden más puntos a la lista.

```
[{  
    "measurement": "Temperature",  
    "tags": {  
        "ID": "00124b000e017202",  
        "building": "Narciso_28",  
        "floor": "3",  
        "room": "Nan"  
    },  
    "time": "2009-11-10T23:00:00Z",  
    "fields": {"value": 25.0}  
}]
```

Código 28 - Write message to Influxdb in JSON format example

En caso de que el almacenamiento de los datos falle, el programa intentará volver a añadir esa información hasta cinco veces. Pasado ese número de intentos, los datos se perderán. Si el proceso ha sido exitoso, entonces el programa espera a que le llegue otro mensaje para volver a realizar el diagrama de flujo.

- *Subscriptor de la configuración de los sensores*

Este servicio se encarga de almacenar la información de los sensores en la base de datos del catálogo. Al igual que el anterior, este se subscribe al mismo tema MQTT para poder recibir mensajes de los sensores. Cuando llega un mensaje nuevo, el servicio analiza si el sensor es nuevo en la plataforma, o en caso de ya estar registrado, si ha sufrido alguna modificación. En cualquiera de los dos casos anteriores, se registra el

suceso en el catálogo y se avisa al usuario a través de la aplicación de Telegram para que tenga constancia de los hechos.

De esta forma, se puede tener un registro de los sensores que están activos en la plataforma, a la vez que se tiene una alerta temprana ante posibles cambios en la plataforma.

Si se analizan las peticiones que realiza este servicio con el catálogo se tiene. De la Tabla 13 se realiza las peticiones GET 1 y 4. Con la primera se obtiene la información del bróker MQTT como en el servicio anterior y con la segunda la información de todos los sensores registrados en el catálogo. De la Tabla 14 se realizan la petición POST 1, para añadir el sensor a la plataforma en caso de que no estuviera registrado. Y de la Tabla 15, se realiza la petición PUT 6, para actualizar las variables que el sensor está midiendo en caso de que se modifiquen. Por último, en caso de que el sensor sea nuevo o se haya modificado alguno se sus atributos, se realiza una petición al Bot de Telegram para que envíe un mensaje al usuario.

Como ocurría en el caso anterior, en este archivo también es necesario configurar previamente una serie de constantes. Sin embargo, como este servicio no interacciona con la base de datos InfluxDB, solo es necesario introducir las tres primeras constantes del Código 27 relacionadas con la dirección del catálogo y la información del bróker MQTT.

A continuación, en la Ilustración 85 se muestra el diagrama de flujo resumido del servicio una vez se han configurado las constantes. Este diagrama se ejecuta automáticamente una vez se inicia el servicio.

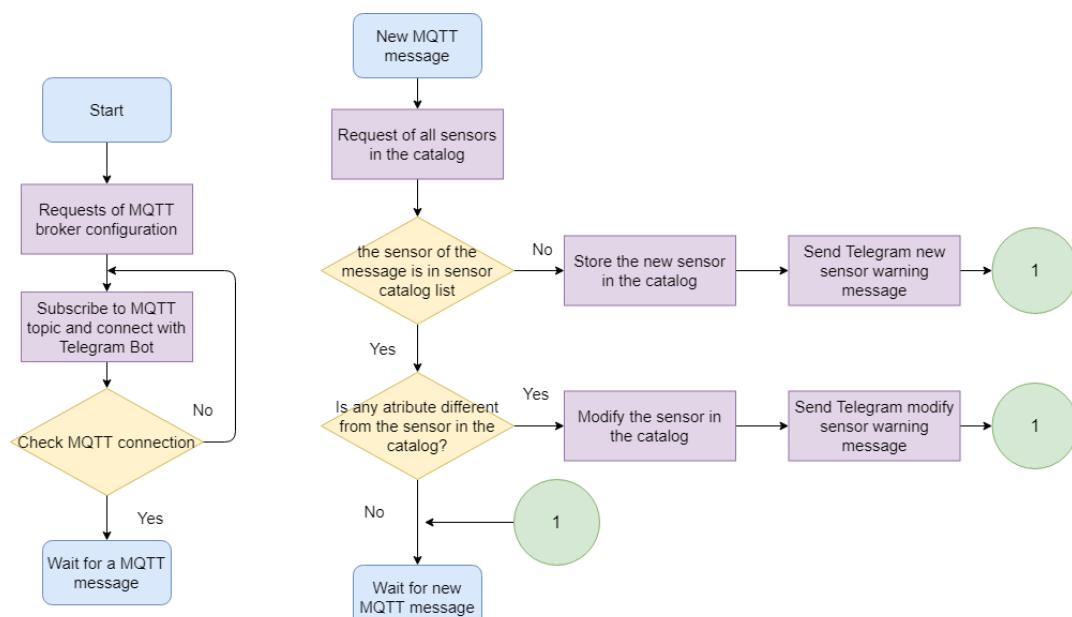


Ilustración 85 - Sensor Configuration Subscriber flowchart

Como en el diagrama de flujo del servicio anterior, primero comienza el diagrama de la izquierda. En el se realizan las configuraciones iniciales y se establece conexión con el bróker MQTT y con el Bot de Telegram. En caso de que no sea posible la conexión con el bróker o la subscripción al tema, el programa se queda bloqueado hasta que se establezca la conexión. Una vez se ha establecido conexión con los dos medios el programa espera a que llegue un nuevo mensaje a través del tema.

Así pues, al llegar un nuevo mensaje se ejecuta el diagrama de la derecha. Con el mensaje se puede saber que variables está midiendo el sensor y con el tema se conoce la ubicación e identificación del sensor. Por lo tanto, para conocer si el sensor está registrado en la plataforma, se hace una petición al catálogo de todos los sensores registrados.

Después se busca el sensor del mensaje en la lista de sensores del catálogo. En caso de que no aparezca, entonces el sensor es nuevo en la plataforma y se tiene que registrar. Para ello, se realiza la petición POST al catálogo en el que se envía un mensaje con formato JSON en el cuerpo de la petición. El formato del mensaje es similar al que se puede observar en el Código 23. Sin embargo, para los atributos “building” y “floor” se almacena la ubicación que se obtiene del tema del mensaje. A “room” se le asigna el valor “Nan”. Los atributos de “available_resources” cambian en función de las variables que mida el sensor. Una vez la petición es correcta se realiza una petición al Bot de Telegram para que avise al usuario de que un nuevo sensor se ha añadido a la plataforma y se vuelve al estado de espera de mensajes.

Si el sensor ya existía en la lista, se comprueba si alguno de sus atributos es diferente frente al existente en el catálogo. Para ello se comprueba que los atributos de la variable “available_resources” sean los mismos. Esto se debe a que un mismo sensor que antes media tres variables como temperatura, presión y humedad, se haya reconfigurado para que solo mida temperatura para ahorrar batería. Solo se comprueba esta variable ya que el identificador no puede cambiar y el tiempo y la ubicación se modifican en el anterior servicio.

En caso de que exista alguna modificación respecto al registrado, se realiza una petición PUT al catálogo en la cual se modifica únicamente los “available_resources” del sensor en cuestión. Como parámetros de la petición se envían los que se observan en la Tabla 15 y dentro del parámetro “available_resources” aparecen las variables que mide el sensor junto con sus unidades. Este parámetro es similar al que se observa en el Código 23 con su mismo nombre. Posteriormente se realiza una petición al Bot de Telegram para que envíe un mensaje al usuario alertando de que un sensor se ha visto modificado en la plataforma y se vuelve al estado de espera de un nuevo mensaje.

Si ocurre que el sensor ya existía en la lista y que es igual al que existía en el catálogo, no se realiza ninguna operación y se espera a que llegue otro mensaje nuevo.

En este servicio ocurre lo mismo que en el anterior al almacenar la localización del sensor. Como no se dispone de la habitación en la que se encuentra el sensor se le asigna a ese atributo el valor nulo. Será trabajo del usuario modificar esta etiqueta a la deseada para que la plataforma automáticamente lo utilice.

5.2.5 Servidor: visualizador

Con todo lo visto hasta ahora, ya se dispone de una plataforma que permite almacenar y realizar un seguimiento de los sensores que se conectan. Sin embargo, para poder llevar a cabo un monitoreo de las variables relacionadas con la eficiencia energética en el hogar es necesario poder visualizar correctamente estos datos y que el acceso a los mismos se pueda hacer de manera sencilla y desde varios dispositivos.

Es por ello, que se escogió Grafana como servicio para la visualización de los datos. Grafana es un software de código abierto que dispone de un gran catálogo de gráficos para poder representar los datos de una manera clara y a la vez dispone de una interfaz sencilla y optimizada para el uso de bases de datos como InfluxDB (Grafana Labs, 2019). Como está orientada al uso de esta base de datos permite realizar operaciones con los datos almacenados e incluso establecer alarmas en caso de que una variable supere cierto límite.

Además, cuenta con una gran comunidad de usuarios que amplían las funcionalidades de Grafana. Se pueden encontrar diferentes plugins y dashboards ya implementados para la funcionalidad que se le quiera dar a la plataforma (Grafana Labs, 2019).

Para la visualización de los datos, grafana se organiza en dashboards. Dentro de este se organiza en paneles, que son los bloques básicos en los que se puede visualizar datos. Cada panel se puede configurar independientemente para que extraiga unos determinados datos de cualquier base de datos que se haya configurado previamente. Se le puede asignar el estilo de gráfico deseado a cada panel para visualizar los datos de la forma que más convenga. El máximo de paneles por fila dentro de un dashboard son 12. Los dashboards permiten albergar infinidad de paneles, hacerlos accesibles a diferentes usuarios y facilitar su búsqueda. Se recomienda leer la guía proporcionada por Grafana en el que aparecen diferentes tutoriales en los que se detallan todos los conceptos. (Grafana Labs, 2019)

Para su instalación es recomendable seguir los pasos que proporciona la documentación de Grafana. Debido a su popularidad, también existen recursos de aficionados que explican como instalarlo. Es compatible con los sistemas operativos más utilizados en la actualidad. Aquí se van a mostrar los comandos necesarios para su instalación en una Raspberry Pi 3 con sistema operativo Raspbian Stretch Lite.

Primero se descarga la última versión disponible para sistemas ARMv6 según indica el propio desarrollador (Grafana Labs, 2019). En el Código 29 se muestra la versión actual, pero es recomendable descargar la última versión existente.

```
$ wget https://dl.grafana.com/oss/release/grafana_6.2.5_armhf.deb  
$ sudo dpkg -i Grafana-rpi_6.2.5_armhf.deb
```

Código 29 - Grafana installation Linux commands

Una vez instalado se deben actualizar los paquetes, solucionar posibles errores, configurar y habilitar el servicio y ejecutarlo para que arranque automáticamente cada vez que inicie la Raspberry Pi. Esto se lleva a cabo con los siguientes comandos:

```
$ sudo apt-get upgrade #Actualizar repositorio  
$ sudo apt --fix-broken install #Arreglar paquetes dañados  
$ sudo /bin/systemctl daemon-reload #Recargar configuración  
$ sudo /bin/systemctl enable grafana-server #Arranque automático  
$ sudo /bin/systemctl start grafana-server #Empezar el servicio
```

Código 30 - Configuration and execution of Grafana Linux commands

Los comandos básicos para controlar Grafana desde la terminal son:

```
$ sudo /bin/systemctl start grafana-server  
$ sudo /bin/systemctl status grafana-server #Estado del servicio  
$ sudo /bin/systemctl stop grafana-server #Parar el servicio
```

Código 31 - Basic Grafana Linux commands

Si se desea modificar la configuración por defecto de Grafana se aconseja seguir la documentación (Grafana Labs, 2019). Para el proyecto se utiliza el servicio con la configuración por defecto.

Una vez se tiene instalado, configurado e iniciado el servicio de Grafana ya se puede acceder a este desde un navegador con su dirección IP y puerto. En el proyecto se utiliza por defecto:

`"http://localhost:3000"`

Código 32 - Grafana URL

Tras añadir el usuario y contraseña, que por defecto es admin/admin, ya se puede utilizar Grafana. El primer paso es introducir la base de datos de InfluxDB. Para ello, desde el panel izquierdo se accede a “Data Sources” y desde ahí a “Add data source”. Entre las bases de datos se encuentra InfluxDB. El siguiente paso es configurar el panel con la configuración de la base de datos del proyecto. En la Ilustración 86 se muestran los campos a llenar. El primero es el nombre con el que Grafana va a identificar esta fuente de datos. El segundo es la dirección de la base de datos en el servidor. Los últimos son el nombre que posee la base de datos donde se almacenan las medidas de los sensores y el usuario y contraseña de InfluxDB para acceder a los datos.

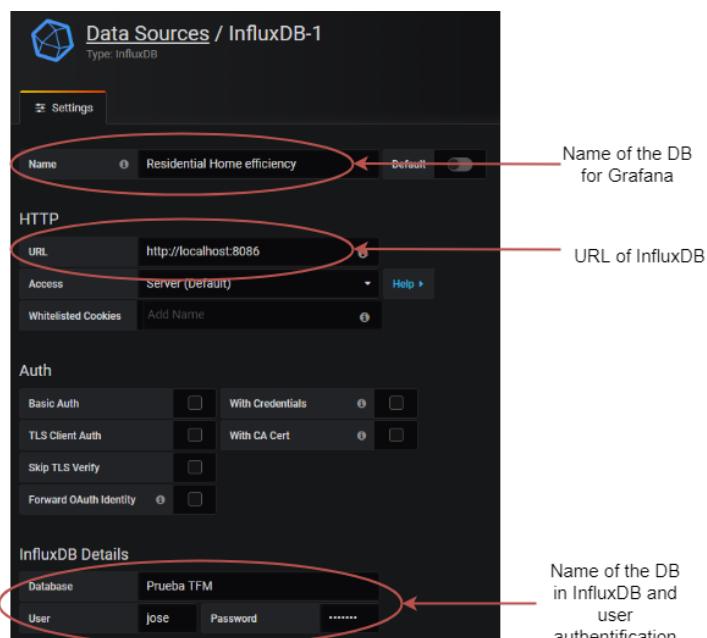


Ilustración 86 - Add Data source menu of Grafana

Una vez Grafana tiene acceso a la base de datos, se debe crear un dashboard para albergar los gráficos. En el menú de la izquierda en “Create dashboard” y automáticamente se genera. Dentro de este se generan los paneles para visualizar las variables de interés de los sensores. La manera de crear los paneles es muy intuitiva y para seleccionar los datos a mostrar, sigue un lenguaje parecido al SQL, pero más visual y sencillo. En la Ilustración 87 se muestra la plantilla base utilizada en el proyecto. Para mostrar los datos.



Ilustración 87 - Grafana dashboard example

El dashboard mostrado es muy simple en comparación a las posibilidades que Grafana ofrece. Sin embargo, debido a la complejidad del proyecto, se prefirió invertir más tiempo en desarrollar otras partes del servidor y dejar Grafana como prueba de concepto. En el capítulo de líneas futuras se desarrolla con mayor profundidad este tema.

5.2.6 Servidor: GUI

Con todo lo analizado en los apartados anteriores, ya sería posible realizar la monitorización de las variables para la mejora en la eficiencia energética en el hogar. Sin embargo, en una plataforma IoT no solo son importantes las medidas. También se debe llevar un control de los sensores, así como poder modificar ciertos parámetros de la plataforma en caso de que el usuario quiera modificarlos. Además, falta por analizar una de las funciones principales de un sistema de monitoreo, la extracción de los datos en el formato adecuado.

Es posible realizar todas las tareas descritas en el párrafo anterior mediante líneas de comando, accediendo directamente a la base de datos. Sin embargo, no es muy funcional y queda restringido a usuarios con conocimiento técnico en los lenguajes de programación usados en el proyecto y con conocimiento sobre esta plataforma en concreto.

Para solucionar este problema, se decide crear una interfaz gráfica que permita a un usuario con cierto conocimiento en la plataforma realizar las tareas de una forma sencilla intuitiva y gráfica. Cabe señalar que el usuario que accede a este bloque debe tener conocimientos más avanzados que el usuario que accede a la visualización de los datos. Se podría decir que, en un escenario real, a Grafana podrían acceder tanto el propietario de la vivienda para conocer los valores actuales e históricos de las variables, como el técnico que monitoriza los valores con un carácter profesional. Mientras que a esta interfaz gráfica solo deberá acceder el técnico para modificar parámetros o extraer datos.

Este bloque se divide a su vez en cuatro microservicios independientes como se muestra en la Ilustración 80. El “GUI Database Configuration” está relacionado con la gestión de InfluxDB. Permite modificar parámetros de las bases de datos y los usuarios.

El “GUI Sensor Configuration” está relacionado con la base de datos del catálogo. De nuevo permite modificar y visualizar el estado de los sensores. “GUI Query Data” permite al usuario realizar búsquedas de datos en InfluxDB y descargarlos. Por último, está “GUI Telegram Configuration” que permite modificar los parámetros del Bot de Telegram.

Para poder acceder a estos servicios es necesario disponer de un ordenador, siendo compatible con los sistemas operativos Windows, Mac y Linux. Además, es necesario disponer de una serie de archivos que permiten ejecutar el servicio y la comunicación. En la Ilustración 88 se muestran los archivos necesarios. De los cinco archivos que se muestran, cuatro coinciden con el nombre de los microservicios que representan. El archivo restante es “GUI_Main.py”. Este es el archivo raíz, es decir, es el archivo que se ejecuta y sirve como menú para acceder a los demás.

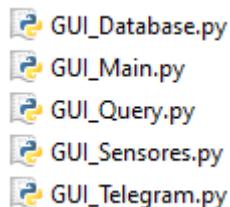


Ilustración 88 - GUI block files

Como se puede observar, se trata de archivos con extensión “.py”. Por lo tanto, el usuario que quiera ejecutar este bloque necesita instalar previamente Python. En el Código 14 se muestra cómo realizarlo. Además, es necesario que se instalen ciertos módulos de Python para que funcione. Algunos ya se han visto a lo largo de la documentación como: requests (Código 26) e influxdb (Código 21). Pero para este bloque es necesario instalar además los módulos wxPython, pandas y numpy. A continuación, se muestran sus respectivos comandos de instalación:

```
$ pip3 install wxPython  
$ pip3 install pandas  
$ pip3 install numpy
```

Código 33 - WxPython, Pandas and Numpy Python module installation Linux commands

A nivel de usuario no es el método más sencillo, ya que hay que requiere varios archivos e instalar varios programas y módulos. Como ocurría anteriormente con Grafana, se escogió desarrollar otras funcionalidades de la plataforma que centrarse en desarrollar un ejecutable único con Python. Por lo tanto, se realizó el bloque GUI ya que era necesario a modo de prueba de concepto, dejando el ejecutable como líneas futuras.

Una vez se tienen los archivos y se han instalado los módulos de Python necesarios, se puede ejecutar el archivo principal “GUI_Main.py”. Como resultado aparece una ventana del menú principal que permite al usuario navegar y elegir a cuál de los cuatro microservicios acceder. En la Ilustración 89 se puede observar dicha ventana con sus cuatro botones.

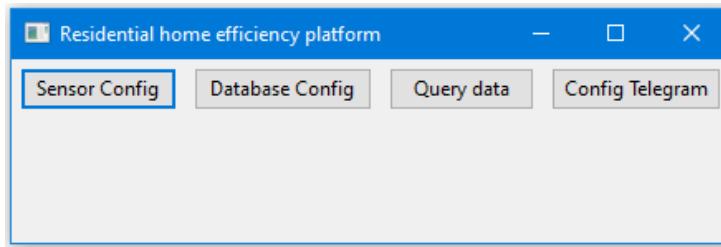


Ilustración 89 - GUI main Window

En los siguientes puntos se va a analizar cada uno de los microservicios que aparecen en la ilustración.

- *GUI configuración de la base de datos*

Este servicio permite al usuario realizar ciertas tareas relacionadas con la configuración de las bases de datos de InfluxDB. Se accede a este desde su botón en el menú principal. Sin embargo, es necesario registrarse previamente con el usuario administrador de InfluxDB. Para ello, aparece una ventana como la que se muestra en la Ilustración 90. En ella se tiene que introducir la dirección IP y puerto de InfluxDB, así como el nombre de usuario del administrador y su contraseña. Solo se puede acceder desde usuarios con privilegios de administrador.

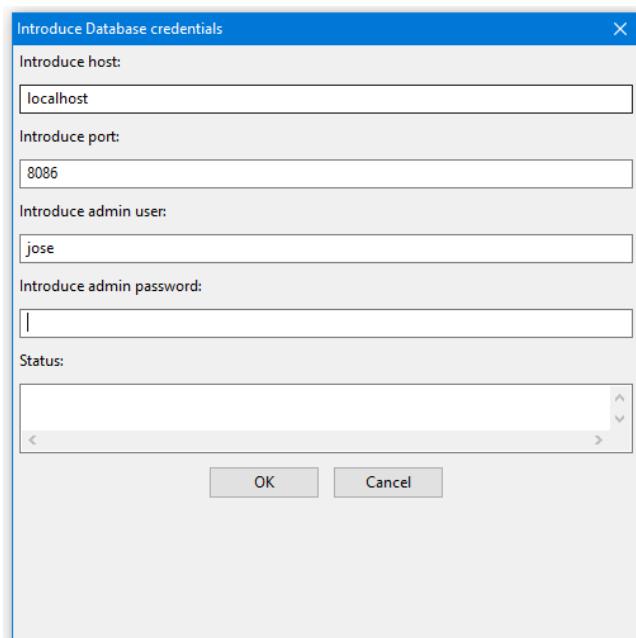


Ilustración 90 - InfluxDB login GUI

Si los parámetros introducidos no son correctos, en el cuadro de "Status" se mostrará el error correspondiente como se muestra en la Ilustración 91.

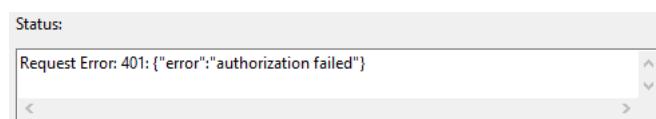


Ilustración 91 - Influxdb login error GUI

En caso de ser correctos, se establece conexión con InfluxDB y aparece una ventana como la que se muestra en la Ilustración 92. En esta se puede observar dos listas. La primera muestra las diferentes bases de datos que existen dentro de InfluxDB junto con las medidas que almacenan cada una. La segunda muestra los usuarios que existen junto con información acerca de sus privilegios. Estos van a ser las dos variables que se pueden gestionar desde este microservicio. Las opciones posibles son: crear, modificar, eliminar y actualizar los datos en caso de que algo haya cambiado.

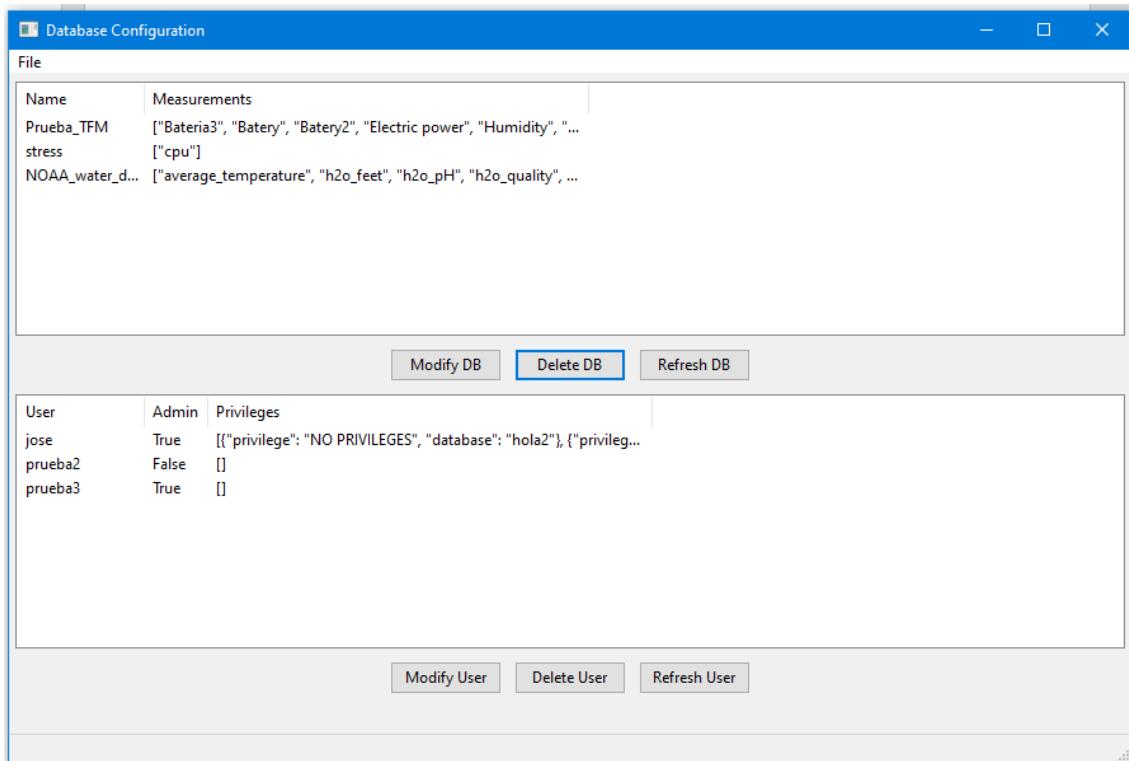


Ilustración 92 - GUI Database Configuration window

Para la opción de crear, se debe pinchar en el menú superior “File” y elegir entre base de datos o usuario. Según la opción elegida se mostrará una de las dos ventanas de la Ilustración 93. En ellas se puede introducir el nombre de la base de datos y usuario, además de en la segunda añadir contraseña y privilegios de administrador. Una vez se pulsa “OK”, se realiza una petición de escritura a InfluxDB, quedando añadidas y, por lo tanto, aparecen en las listas de la Ilustración 92.

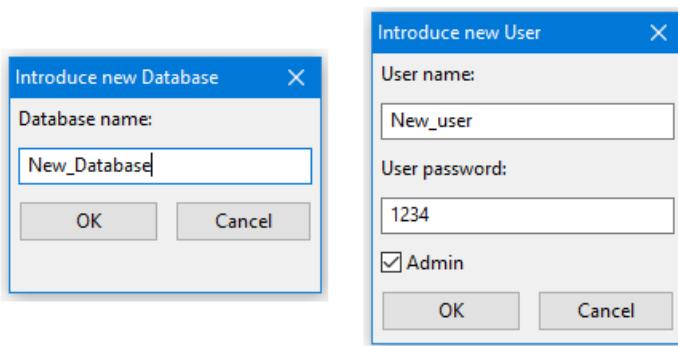


Ilustración 93 - New Database and user GUI windows

La siguiente opción es la de modificar. Para ello, es necesario pinchar primero en el elemento de la lista, ya sea una base de datos o un usuario, que se quiere modificar y luego pinchar en el botón de “Modify”. Las ventanas que aparecen son las que se

muestran en la Ilustración 94. Como base de datos se ha escogido la que aparece en la Ilustración 92 con nombre “Prueba_TFM” y de usuario “jose”.

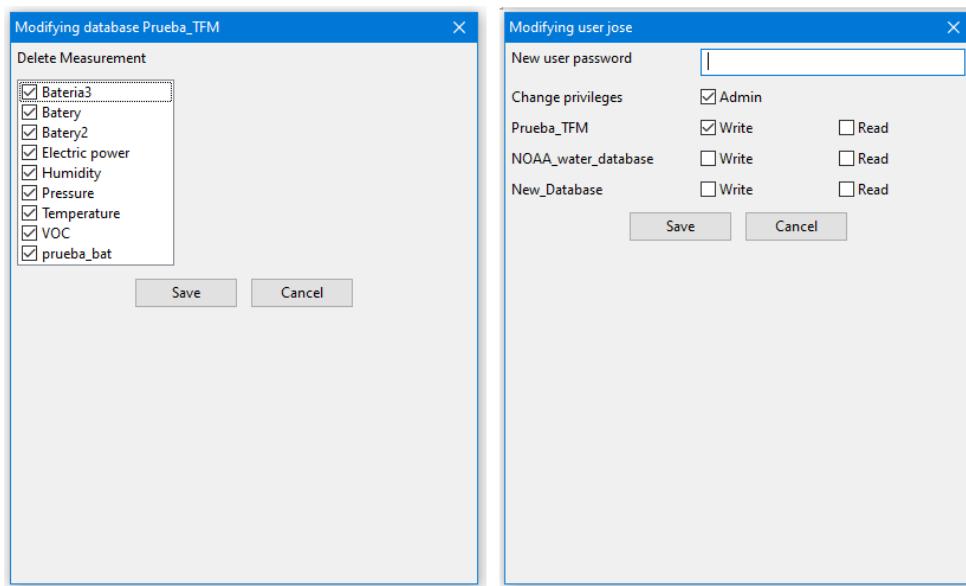


Ilustración 94 - Modify database and user GUI windows

En el caso de la opción modificar de la base de datos su funcionalidad es muy simple, ya que solamente permite eliminar la medida que se desee. Sin embargo, en el microservicio “GUI Query Data” aparecen más opciones relacionadas con los datos. En el caso de modificar el usuario, se permite cambiar su contraseña, cambiar los privilegios de administrador y conceder o quitar los privilegios de escritura o lectura a cada una de las bases de datos. Cuando se pulse el botón “Save”, se realiza la petición a InfluxDB y los cambios se quedan guardados.

Para la opción de borrar, de nuevo es necesario seleccionar previamente un elemento de la lista y pulsar el botón de “Delete”. Tras pulsar el botón, aparece una ventana emergente que avisa si realmente se quiere eliminar esa base de datos o usuario en caso de que se haya pulsado el botón por error. Si se procede con la eliminación, de nuevo se realiza una petición a InfluxDB y se elimina el parámetro seleccionado.

Por último, se encuentra el botón de “Refresh” que se encarga de actualizar los valores de la lista en caso de que el usuario quiera comprobar si se han modificado alguno de los parámetros.

- *GUI configuración de la base de datos de sensores*

Este microservicio se encarga de mostrar al usuario la información que se encuentra en el catálogo referente a los sensores. Además de mostrarla, también permite modificar ciertos atributos de los sensores para poder configurarlos. Para acceder al servicio se debe pinchar en el botón “Sensor Config” del menú principal en la Ilustración 89. En este no se necesita autentificación para acceder, sin embargo, es necesario que el archivo “GUI_Sensores.py” de la Ilustración 88 contenga la URL del catálogo para acceder a la información. Por lo tanto, si la URL del catálogo cambia, es necesario cambiarla en este archivo también. Cuando se pincha en el botón aparece una ventana como la que se muestra en la Ilustración 95.

5. Desarrollo del sistema de monitoreo de la eficiencia energética del hogar

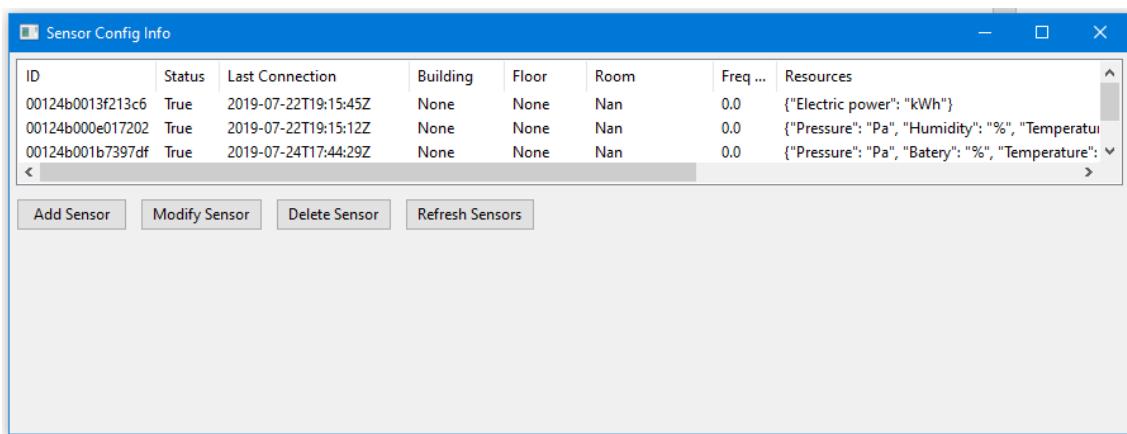


Ilustración 95 - GUI Sensor Configuration window

En la ilustración se muestra la ventana principal en la que aparece una lista con todos los sensores del catálogo y sus propiedades. Además, se cuenta con una serie de botones que permiten: añadir, modificar y eliminar cualquier sensor y un último botón que actualiza los valores de la lista. Para que aparezca la información de los sensores es necesario que el microservicio realice una petición GET al catálogo que corresponde con la número 4 en la Tabla 13.

La lista se divide en 7 columnas que hacen referencia a las propiedades de los sensores. En el apartado 5.2.3, punto “Base de datos de la configuración de los sensores” se explica qué significa cada propiedad.

Para añadir un nuevo sensor a la plataforma se tiene que pulsar el botón “Add Sensor”. Esto hará que aparezca una nueva ventana como la que se observa en la Ilustración 96. En esta se debe introducir el identificador o “ID” del sensor, la ubicación en la que se encuentra y el periodo entre dos medidas consecutivas del sensor en segundos. Una vez se pulse el botón “Save”, se realiza la petición POST 1 de la Tabla 14 que almacena la información del sensor.

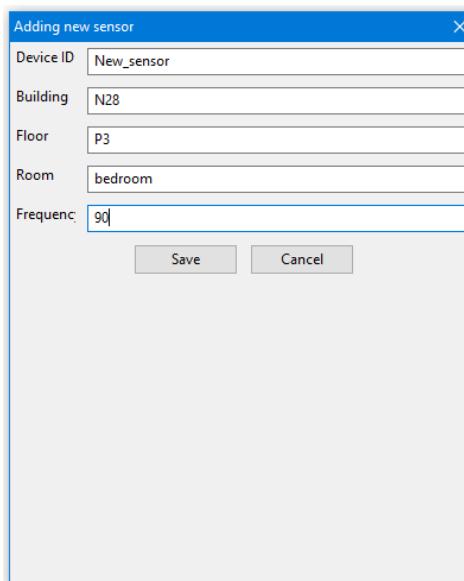


Ilustración 96 - Add new sensor GUI window

Si se observa la Ilustración 95 y la Ilustración 96 se puede ver que al añadir un nuevo sensor faltan tres parámetros que no son establecidos por el usuario. Estos son: “Status”, “Last connection” y “Resources”. Esto se debe a que como nunca se ha

conectado el sensor a la plataforma estos parámetros son nulos. Por lo tanto, toman valores de “False”, “1970-01-01T01:00:00Z” que corresponde a la fecha de valor 0 en formato Unix timestamp y “[]” respectivamente.

Esta funcionalidad puede parecer inútil a priori, ya que cuando un sensor envía datos al servidor automáticamente es almacenado en el catálogo. Por lo tanto, no debería ser necesario añadirlo antes a la base de datos. Sin embargo, esta forma de añadir nuevos sensores permite configurar previamente dos parámetros muy relevantes en el funcionamiento óptimo de la plataforma.

Por un lado, permite configurar la ubicación del sensor, en lo que a la habitación o “room” se refiere. Por lo tanto, cuando el sensor envíe los datos, estos serán almacenados con la ubicación completa, lo que facilita el filtrado de datos. Por otro lado, se configura la frecuencia de medida o tiempo entre medidas. Esto permite al microservicio “Sensor Status Subscriber” del bloque “Control Strategies” de la Ilustración 80 comprobar si el sensor está activo o no. Esta segunda funcionalidad se explica con mayor detalle en el apartado de estrategias de control.

Por consiguiente, se recomienda primero introducir manualmente la información de los sensores al catálogo y posteriormente conectarlos. Sin embargo, aunque no se realice esta operación la plataforma funciona de forma correcta.

Para acceder a la función de modificar un sensor se debe seleccionar primero el sensor que se busca modificar en la lista y luego pulsar “Modify Sensor”. Aparecerá una ventana similar a la Ilustración 97. Esto permite modificar las propiedades “Status”, “room” y “frequency” del sensor. De esta forma, se pueden dar valor a estas propiedades en caso de que no se hubiera añadido el sensor al catálogo de forma manual antes de que se conectara al servidor. Como se ha visto en la funcionalidad de añadir un nuevo sensor, estas propiedades son muy importantes para el funcionamiento óptimo de la plataforma y se recomienda completar sus campos.

De nuevo, al pulsar “Save” los campos que hayan variado respecto a los valores anteriores serán almacenados en el catálogo. Para ello, se realizan las peticiones PUT 1, 3 y 4 de la Tabla 15.

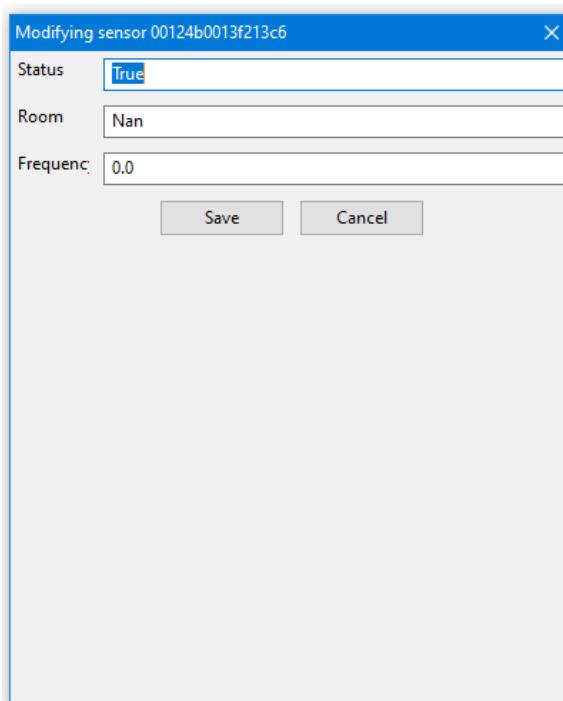


Ilustración 97 - Modify sensor GUI window

5. Desarrollo del sistema de monitoreo de la eficiencia energética del hogar

La función de eliminar sensores funciona igual que la vista en el microservicio de “GUI Database Configuration”. Al seleccionar un sensor y pulsar el botón “Delete Sensor”, el programa preguntará si eliminarlo del catálogo. En caso afirmativo se realiza la petición DELETE 1 de la Tabla 16 que lo elimina del registro.

El último botón de “Refresh Sensors” actualiza la lista de sensores en caso de que alguno de los valores se haya modificado.

- *GUI para la extacción de datos*

Este microservicio permite gestionar la información almacenada en las bases de datos de InfluxDB. Se trata de una funcionalidad muy importante en la plataforma ya que, gracias a esta se pueden modificar etiquetas de los sensores y extraer los valores de los sensores mediante un sistema de filtros para su posterior procesado en aplicaciones externas a la base de datos.

Para acceder al microservicio es necesario pinchar en el botón “Query Data” de la Ilustración 89. Como este microservicio se conecta con InfluxDB es necesario que el usuario que desea acceder se identifique. Por lo tanto, se tendrá que seguir el proceso descrito en este apartado, en el punto “GUI para la configuración de la base de datos” con la Ilustración 90 y la Ilustración 91. Tras este paso, se mostrará una ventana como la que aparece en la Ilustración 98.

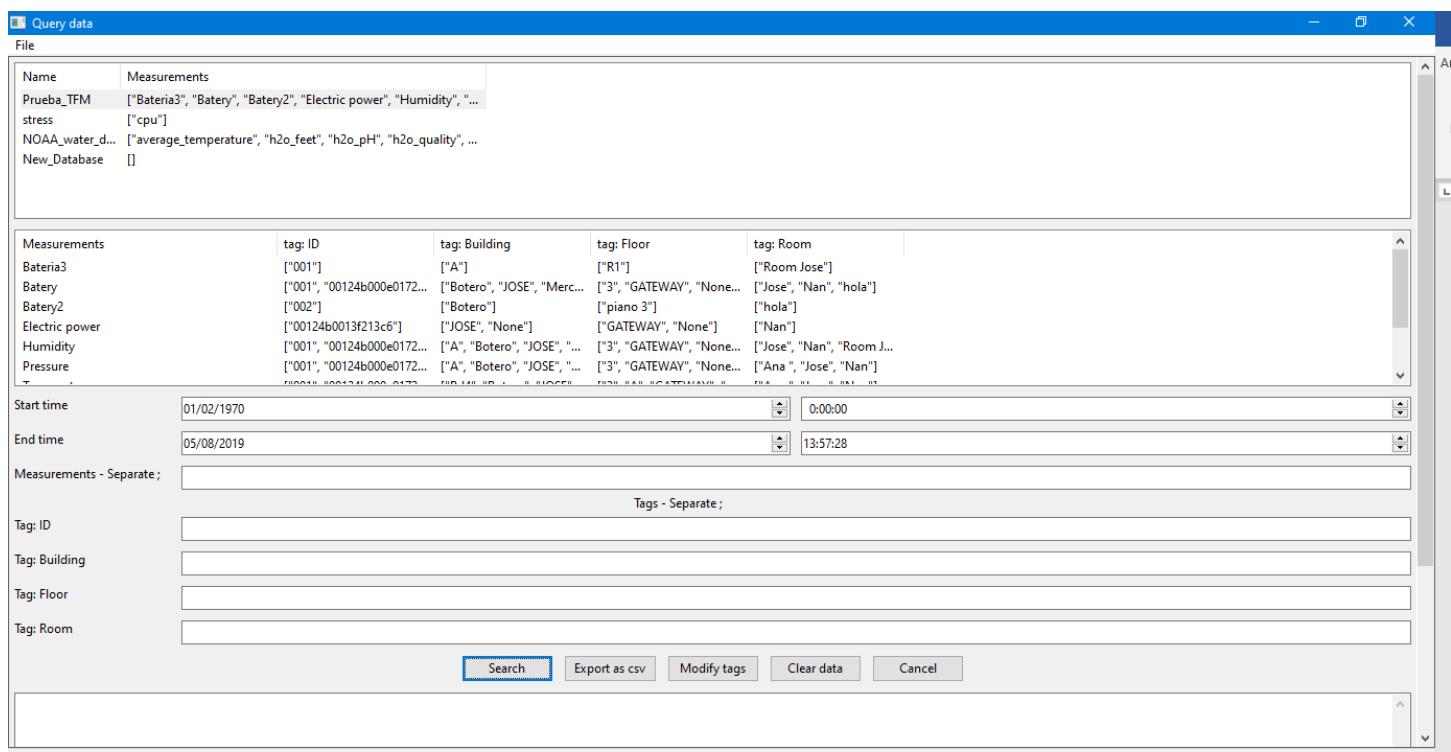


Ilustración 98 - GUI Query Data window

En la ilustración aparece una ventana compleja. Se tiene un menú superior con el nombre de “File” con el que se puede elegir en que carpeta y con qué nombre se quieren guardar los datos extraídos de la base de datos. Después se dispone de dos listas. La primera muestra las bases de datos dentro de InfluxDB con las medidas que almacena cada una. Cuando se pincha en una de las bases de datos, en la segunda lista aparecen cada una de las medidas junto con las etiquetas. Se tiene una columna por cada etiqueta diferente. Dentro de una columna de etiquetas aparecen los valores de las etiquetas

para cada una de las medidas. Esto es muy útil para filtrar los datos ya que el usuario puede conocer que etiquetas puede buscar para encontrar valores.

A continuación, se tiene una serie de campos para realizar la búsqueda de datos. Las dos primeras permiten introducir la fecha y hora entre las que se quiere realizar la búsqueda. El siguiente campo permite realizar búsquedas por el nombre de la medida. Para realizar búsquedas este campo debe ser rellenado con al menos un valor. Se pueden introducir varias medidas pero estas deben ir separadas por “;”, es decir, si se quisieran buscar dos medidas de la base de datos “Prueba_TFM” se introducirían como: “Temperature;Humidity”. Los siguientes campos corresponden con las etiquetas. Funcionan de la misma forma que el campo de medidas.

Siguiendo con el ejemplo puesto para las medidas, si se introdujera en las etiquetas de “ID” y “building” los valores: “001” y “A;Botero”, daría como resultado los valores de las medidas “Temperature” y “Humidity” con el valor de la etiqueta “ID” igual a “001” y a la vez los datos que cumplan con el valor de “building” igual a “A” o que tengan el valor “Botero”. Esto quiere decir que dentro del mismo tipo de etiqueta se busca según el operador lógico “OR” y entre diferentes tipos de etiquetas según el operador lógico “AND”.

Como se dispone de la lista de medidas y etiquetas se puede saber previamente si la búsqueda dará resultado. En caso de que no haya resultado y en la lista se muestre el valor de las etiquetas o medidas, puede deberse bien a que no existen valores en el rango temporal introducido, o bien no existen datos que cumplan los diferentes tipos de etiquetas a la vez.

Los botones que aparecen corresponden a las funcionalidades disponibles con este microservicio. Entre ellos se encuentra, buscar datos en la base de datos seleccionada, extraer datos de la base de datos, modificar etiquetas de los datos almacenados en la base de datos y eliminar datos. Por último, se dispone de un cuadro en el que se muestran los resultados de la búsqueda realizada.

Por lo tanto, la forma de utilizar el servicio es la siguiente. Primero se debe seleccionar la ubicación y nombre del fichero donde se quiere guardar los datos. Esto se realiza pinchando en el menú superior. Este primer paso no es obligatorio, ya que el programa volverá a preguntar en caso de no haberse introducido. Luego se selecciona la base de datos con la que se quiere trabajar. Para poder utilizar las funcionalidades del servicio es necesario introducir al menos el nombre de una medida en el campo “measurements”. Tras estos pasos, ya se puede hacer uso de las funcionalidades que ofrece el servicio.

La funcionalidad de búsqueda de datos permite al usuario encontrar los valores de la base de datos que deseé según los filtros introducidos. Al pulsar el botón de “Search” aparecerá en el cuadro inferior los resultados deseados como se muestra en la Ilustración 99. Además, se muestra una gráfica con los valores buscados como en la Ilustración 100, que pertenecen a la búsqueda de la medida “Temperature”. Esta funcionalidad permite al usuario visualizar los datos antes de trabajar con ellos.

Para llevar a cabo esta acción, el programa realiza una petición de extracción de datos a InfluxDB como la que se muestra en el Código 19, y con un mensaje como el del Código 20.

5. Desarrollo del sistema de monitoreo de la eficiencia energética del hogar

Query request: SELECT * FROM Temperature WHERE time >= '2019-07-01 00:00:00' AND time <= '2019-08-14 16:55:02'				
Query:	Temperature :	ID building	floor	room
2019-07-17 18:22:42+00:00	None	JOSE GATEWAY	Nan	0
2019-07-17 18:24:12+00:00	None	JOSE GATEWAY	Nan	0
2019-07-17 18:25:42+00:00	None	JOSE GATEWAY	Nan	0
2019-07-17 18:27:12+00:00	None	JOSE GATEWAY	Nan	0
2019-07-17 18:28:42+00:00	None	JOSE GATEWAY	Nan	0
2019-07-18 17:58:24+00:00	001 Botero	3	Nan	25
2019-07-18 17:58:31+00:00	001 Botero	3	Nan	25
2019-07-18 17:58:38+00:00	001 Botero	3	Nan	25
2019-07-18 17:58:45+00:00	001 Botero	3	Nan	25
2019-07-18 17:58:52+00:00	001 Botero	3	Nan	25
2019-07-18 17:58:59+00:00	001 Botero	3	Nan	25
2019-07-18 17:59:06+00:00	001 Botero	3	Nan	25
2019-07-18 17:59:13+00:00	001 Botero	3	Nan	25

Ilustración 99 - Search functionality example GUI Query

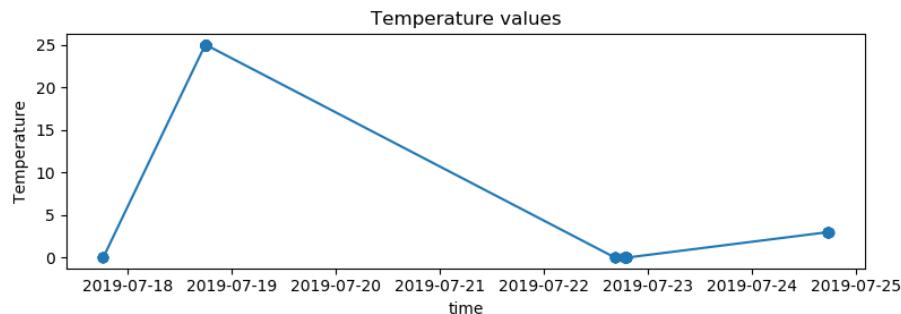


Ilustración 100 - Search functionality graph example GUI Query

Para utilizar el resto de las funcionalidades, es obligatorio realizar previamente la búsqueda de los datos como se acaba de explicar. El objetivo es aumentar la protección de los datos ante posibles errores del usuario, ya que la operación que se realice posteriormente es irreversible.

La funcionalidad de extracción de datos permite al usuario obtener un archivo con extensión “.csv” de los datos que se han buscado previamente. Al pulsar el botón “Extract as csv” se generan ficheros en la ubicación y con el nombre seleccionados. Se generarán tantos ficheros como nombres de medidas introducidos. Esto quiere decir que si se busca sobre las medidas “Temperature;Humidity” se obtendrán dos ficheros csv con los valores que cumplen los filtros introducidos.

En cuanto al formato con el que se almacena la información, se trata de una primera fila con el nombre de los tipos de etiquetas y la medida buscada, seguido de filas con los valores temporales, de las etiquetas y el valor de la medida. Toda esta información se encuentra separada por comas. En el Código 34 se muestra un ejemplo de las dos primeras filas del archivo csv generado al realizar la búsqueda de la medida “Temperature”.

```
1. ,ID,building,floor,room,temperature,time
2. 2019-07-17 18:22:42+00:00,,JOSE,GATEWAY,Nan,0,2019-07-17 18:22:42+00:00
```

Código 34 - csv file format example

A esta información extraída de los sensores se pueden aplicar algoritmos de Machine Learning para predecir comportamientos domésticos y mejorar la eficiencia energética en el hogar mediante avisos o control automático de diferentes dispositivos. Sin embargo, esto queda fuera del proyecto y se analizará con mayor detalle en las líneas futuras.

La siguiente funcionalidad de este microservicio es modificar las etiquetas de los datos de los que se ha realizado la búsqueda previamente. Como ya se ha visto anteriormente, en caso de que el sensor no sea añadido al catálogo antes de conectarlo,

la etiqueta de “room” aparecerá con el valor “Nan”. Esta funcionalidad permite modificar el valor de esta etiqueta en los datos que ya han sido almacenados en la base de datos. También permite modificar el valor del resto de etiquetas en caso de que haya un error en su valor, o que el usuario prefiera cambiarlas para búsquedas futuras.

En cualquier caso, tras pinchar en el botón “Modify tags” se abre una ventana como la que se muestra en la Ilustración 101, que permite modificar el valor de las etiquetas de los datos ya buscados. Se modificarán las etiquetas para las que se introduzca algún valor en su respectivo campo.

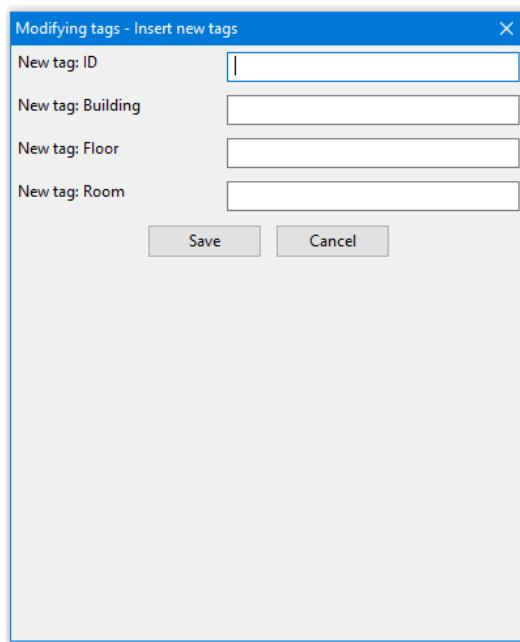


Ilustración 101 - Modify Tags GUI window

Sin embargo, a diferencia de las funcionalidades vistas hasta ahora, al pulsar el botón “Save” no se realiza una única petición a InfluxDB, ya que no existe una petición de modificación de datos. Para ello, es necesario llevar a cabo una serie de pasos:

5. Guardar en una variable los datos que cumplen los filtros y se quieren modificar.
6. Borrar los datos que se quieren modificar de la base de datos.
7. Modificar las etiquetas de los datos almacenados en la variable con los nuevos valores.
8. Escribir los datos modificados en la base de datos.

Como se puede ver, no se puede sobrescribir los datos que ya existen en la base de datos, es necesario eliminarlos primero y volverlos a almacenar. Con esta funcionalidad se lleva a cabo las operaciones de extracción, eliminación y escritura de datos que ya se han explicado en el Código 17, Código 19 y Código 20.

La última funcionalidad es el borrado de datos de la base de datos. Para ello simplemente hay que pulsar el botón “Clear data” tras haber realizado la búsqueda de los datos. Al pulsarlo, todos los datos que cumplieran los filtros serán eliminados de la base de datos. Esta operación es irreversible, por lo que se advierte al usuario de que solo utilice esta función si de verdad está seguro de qué datos se van a eliminar. El funcionamiento de esta funcionalidad es sencillo, simplemente utiliza el Código 19 y Código 20 con la llamada “DELETE” en vez de “SELECT”.

- *GUI para la configuración del Bot de Telegram*

Este último microservicio del bloque GUI permite al usuario modificar la configuración del Bot de Telegram. Esta información se encuentra almacenada en el catálogo, por lo que solo es necesario comunicarse con este y por tanto, la URL del catálogo está en el archivo y debe ser modificada en caso de que varie. Para acceder a este microservicio se tiene que pulsar el botón “Config Telegram” de la Ilustración 89. Esto realiza la petición GET número 5 de la Tabla 13 de la que se obtiene la información del Bot y muestra una ventana como la que se observa en la Ilustración 102.

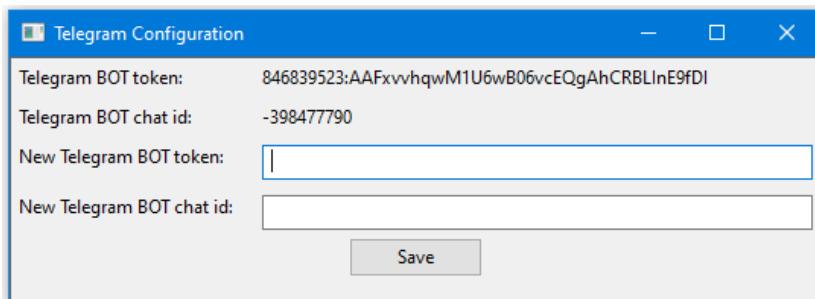


Ilustración 102 - GUI Telegram Configuration window

Los dos primeros campos muestran el token del Bot y el id del chat necesarios para llevar a cabo la comunicación entre el servidor y el Bot. En el apartado 5.2.8 de este capítulo se entrará en más detalle sobre los campos. Los dos siguientes permiten introducir el nuevo token o id del chat. Si se introduce algún valor en los campos y se pulsa “Save”, la información es almacenada en el catálogo a través de la petición PUT número 5 de la Tabla 15.

5.2.7 Servidor: estrategias de control

Los anteriores apartados permiten al usuario almacenar, monitorear y configurar la plataforma al completo. Sin embargo, es necesario desarrollar bloques que se encarguen de monitorear el comportamiento de ciertas variables que se almacenan en la base de datos. Por lo tanto, se desarrolla este bloque con la idea de poder crear una serie de alarmas y avisos para el usuario en caso de que los valores de ciertas variables superen unos límites establecidos o se comporten de manera inesperada.

Sería posible llevar a cabo esta tarea de manera manual, en la que el usuario a través de Grafana y de la GUI monitorea el estado de las medidas y de los sensores en tiempo real. No obstante, debido a la cantidad de datos y a la imposibilidad e inefficiencia de tener a una persona controlando el sistema constantemente, es necesario desarrollar un bloque que analice y notifique de los cambios que ocurren automáticamente.

A pesar de que este bloque podría disponer de numerosos microservicios que analizaran cada variable, debido a la complejidad del proyecto y problemas temporales, se han desarrollado únicamente dos microservicios que se encargan de monitorear el estado de los sensores. Esto es fundamental, ya que un mal funcionamiento de los sensores implica la pérdida de datos o incluso la pérdida del sensor, con su coste económico añadido. Sin embargo, monitorear los valores que envían los sensores se puede llevar a cabo desde otros bloques del servidor y no es tan crítico como lo anterior. Es por ello por lo que se decide desarrollar dos microservicios, uno que monitoree el estado de la batería de los sensores y otro que alerte ante cambios en el estado de los sensores.

Si se observa la Ilustración 80, el bloque “Control strategies” se divide en los dos microservicios citados. Por un lado, el “Batery alarm System” que se encarga de avisar mediante Telegram al usuario si la batería de los sensores está en niveles bajos. Por otro lado, el “Sensor Status Subscriber” que comprueba el estado de los sensores en la plataforma y en caso de que ocurra un cambio en el estado, avisa al usuario a través de Telegram.

Para que este bloque funcione es necesario ejecutar en el servidor dos archivos de Python que se muestran en la Ilustración 103. Para el primer microservicio descrito en el párrafo anterior es necesario ejecutar el archivo “Batery_warning.py” y para el segundo el archivo “Sensor_status_warning.py”. Los archivos “MyMQTT.py” y “TelegramBot.py” son los mismos archivos que se han visto en la Ilustración 83 y funcionan de la misma manera.

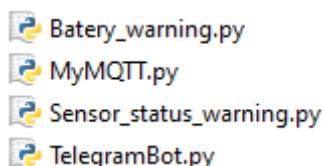


Ilustración 103 - Control strategies files

Para que los archivos se puedan ejecutar, es necesario que el servidor tenga instalado ciertos módulos de Python, aunque ya se han visto en los apartados anteriores. Estos son: influxdb (Código 21), requests (Código 26). Para el archivo “TelegramBot.py” es necesario otro módulo más, pero se explicará en el apartado oportuno.

A continuación, se va a explicar con mayor detalle el funcionamiento de los microservicios de este bloque:

- *Subscriptor de alarma de batería*

Este microservicio se encarga de avisar al usuario en caso de que los niveles de batería de los sensores se encuentren bajos. Para ello, el servicio se subscribe al tema al que se envían los mensajes MQTT. Cuando llega un mensaje nuevo, extrae la información relativa al nivel de batería y en caso de estar en los umbrales establecidos manda un mensaje a través de Telegram para que el usuario compruebe el estado del sensor y lo recargue en caso de que sea necesario.

Gracias a esto el usuario es capaz de conocer previamente a que se apague un sensor si es necesario cargarlo. Además, el servicio aporta una estimación de la duración de la batería para que el usuario sepa cuando se apagará. Con esto también se puede conocer si la vida útil de la batería ha disminuido al tener muchos avisos del mismo sensor sin necesidad de comprobar Grafana.

Analizando la comunicación de este microservicio con otros bloques del servidor, se realizan las peticiones GET 1, 2 y 3 de la Tabla 13 al catálogo. Con la primera obtiene información del bróker MQTT, con la segunda la dirección y puerto de InfluxDB y con la última la dirección y el puerto de Grafana. También se comunica con InfluxDB para extraer el histórico de batería del sensor con la que estima la duración de esta. En caso de que los niveles de batería sean menores que cierto nivel realiza una petición al Bot de Telegram para avisar al usuario.

Para que este microservicio funcione correctamente se tienen que definir en el código una serie de constantes. Como se ha visto en el párrafo anterior, este se comunica con

el catálogo y con InfluxDB, por lo que, como ocurría con el microservicio “Sensor Data Subscriber”, se necesita definir las constantes que se observan en el Código 27.

Para comprender con mayor detalle el funcionamiento de este microservicio, en la Ilustración 104 se muestra el diagrama de flujo que se ejecuta automáticamente al comienzo del servicio.

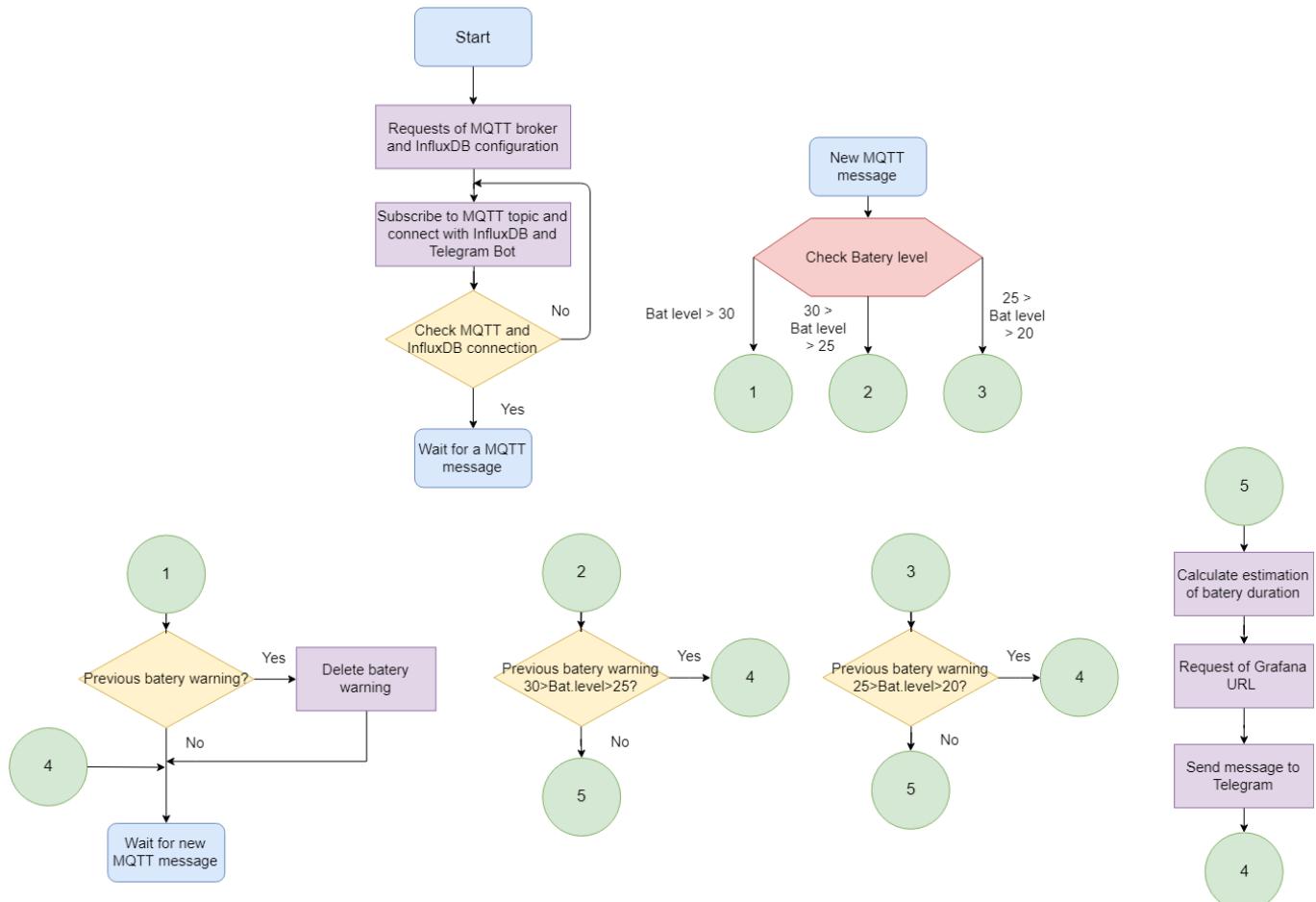


Ilustración 104 - Batery Alarm Subscriber flowchart

Todo comienza con el diagrama de flujo superior izquierda, en el que se llevan las tareas de configuración. Primero se hace la petición al catálogo sobre la información del bróker y de InfluxDB. Luego se conecta con el bróker MQTT y se subscribe al tema, al igual que se conecta con InfluxDB y con el Bot de Telegram. En caso de que no sea posible la conexión con los dos primeros, se queda bloqueado en un bucle hasta que logra conectarse. Una vez que todo lo anterior es correcto el programa espera a que se reciba algún mensaje a través de MQTT.

Cuando llega un mensaje, el servicio extrae la información del nivel de batería, la identificación del sensor y su localización para poder realizar más tarde la estimación de la duración de la batería. El valor del nivel de la batería se compara con tres rangos. El primero si el valor es superior a 30, el segundo si está entre 30 y 25 y el tercero si se encuentra entre 25 y 20. El valor de 20 corresponde al nivel de batería en el cual se recomienda apagar el sensor para evitar la descarga completa de la batería y por tanto que reduzca su vida útil.

Un sensor por lo general envía información al servidor más de una vez entre los rangos que van de 30 a 20. Sin embargo, no es de utilidad que el usuario reciba un mensaje de batería baja cada vez que el sensor envía un dato en ese rango. Es por ello

que se necesita una variable que almacene información acerca de si el usuario ha recibido un mensaje en cada uno de los rangos. En caso de que lo haya recibido, no se le mandará otro mensaje hasta que sobrepase el siguiente nivel inferior.

Por lo tanto, como el rango de nivel de batería superior a 30 se considera nivel de batería alto no es necesario informar al usuario. Pero si que es necesario comprobar en la variable si existe algún aviso de batería baja previo de ese sensor. En caso de existir, se elimina el registro en la variable y se vuelve al estado de espera de un mensaje nuevo a través de MQTT. El caso en el que el sensor tiene un aviso previo corresponde, por ejemplo, al caso en el que el sensor entró en el rango de batería baja y el usuario lo desconectó de la red para cargar la batería al completo. Cuando vuelve a conectarlo a la red y el sensor envía un mensaje al servidor, este tiene en la variable un registro de que se avisó al usuario de batería baja. Para que el programa vuelva a avisar de este estado es necesario borrar el registro.

En el caso de que el nivel de batería se encuentre inferior a 30%, se dice que el sensor tiene la batería baja y por lo tanto se tiene que avisar al usuario. Una vez que entra en este rango se pueden recibir dos mensajes de batería baja. Uno cuando se encuentra en el rango de 30%-25% y otro entre 25%-20%.

En cualquiera de los dos casos, el servicio comprueba si existen avisos previos del sensor sobre batería baja en alguno de esos dos rangos. En caso de existir, el servicio vuelve a su estado de espera de mensaje nuevo, ya que ya se avisó previamente sobre ese nivel de batería bajo. En caso de que no exista registro previo de aviso, lo primero se crea el registro y se procede a estimar la duración de la batería y a avisar al usuario.

Cualquiera de los dos rangos de batería baja llevan a cabo el mismo proceso que corresponde al diagrama inferior derecha de la Ilustración 104. Primero se estima la duración de la batería hasta que llegue al nivel del 20%. Para ello se hace una petición a InfluxDB para extraer el histórico del nivel de batería de ese sensor que cumpla con la información de localización que venía en el mensaje. Se toman datos desde el momento que llega el mensaje hasta 10 semanas atrás en el tiempo. Se ha configurado 10 semanas porque se considera un horizonte estable en lo que a consumo de batería se refiere. La petición es similar a la que se observa en el Código 19 con el mensaje del Código 20. Con esa información se busca la última vez que se cargó el sensor, de lo que se obtiene su valor de batería desde la última carga y el momento en el que se realizó. Con estos puntos y la fecha y valor de la batería que se recibe en el mensaje se puede hacer una recta, que en el corte con el valor de 20%, que corresponde a no tener batería, se obtiene una aproximación lineal de la duración de la batería.

La hipótesis de que la batería se descarga de forma lineal no es correcta, pero si es válida para aportar al usuario una aproximación, ya que lo importante es que el usuario sea alertado de que la batería está baja.

Una vez se tiene la estimación de la duración de la batería, se hace una petición al catálogo sobre la URL de Grafana y se procede a enviar el mensaje a través de Telegram. La información de Grafana será incluida en el mensaje para que el usuario, al ver el aviso, pueda acceder directamente desde el móvil a los gráficos de Grafana para comprobar que está ocurriendo con el sensor. Además, en el mensaje de Telegram se incluye información sobre el nivel de batería actual del sensor y la estimación de la duración en minutos. Tras esto, el servicio vuelve al estado de reposo en el que espera que un nuevo mensaje llegue a través de MQTT para repetir el proceso.

- *Subscriptor de alarma por estado de sensores*

Este otro microservicio se encarga de supervisar el estado de los sensores en la plataforma y avisar al usuario en caso de que se produzca algún cambio inesperado. Analiza los dos posibles cambios de estado de los sensores, de activo a inactivo y de inactivo a activo y avisa en caso de que se produzca alguno de ellos.

Para ello, lleva a cabo dos tareas simultaneas. Por un lado, se subscribe al tema al que se envían los mensajes MQTT. Cuando entra un nuevo mensaje comprueba el estado del sensor del que se recibe el mensaje. En caso de estar inactivo se cambia a activo y se envía un mensaje al usuario. Por otro lado, dispone de un temporizador que cuando pasa una cantidad de tiempo establecida por el usuario, comprueba si todos los sensores que hay en el catálogo siguen estando activos. En caso de que no lo estén, los cambia a su estado correcto y envía un mensaje al usuario.

Con este microservicio se obtiene un catálogo de los sensores preciso y actualizado de forma automática. Además, el usuario es alertado acerca de estos cambios, por lo que puede conocer en todo momento si un sensor se conecta al servidor durante su instalación o si un sensor deja de funcionar inesperadamente debido a un mal funcionamiento. Por lo tanto, con este microservicio se agilizan las tareas de instalación y mantenimiento ya que también se dispone de su ubicación exacta.

Si se observa las comunicaciones del microservicio con los diferentes bloques, este realiza las peticiones GET 1 y 4 de la Tabla 13. De nuevo la primera es para obtener la información del bróker MQTT y la segunda para obtener información actualizada del estado de los sensores en el catálogo. También realiza la petición PUT 3 de la Tabla 15 con la que actualiza el estado de los sensores en el catálogo.

Existen una serie de constantes que el usuario puede modificar accediendo al código. Como solo se comunica con el catálogo, si se observa el Código 27 solo necesita de las tres primeras constantes para funcionar correctamente. Sin embargo, existen dos constantes que el usuario también puede definir que son:

```
STATUS_CHECK_INTERVAL = <intervalo de tiempo en segundos>
STATUS_CHECK_TIMES= <número de periodos>
```

Código 35 - Sensor Status Subscriber constants

La primera constante es el temporizador, es decir, el tiempo que tiene que transcurrir para que el programa compruebe si algún sensor en el catálogo está activo, pero debería estar inactivo. Esta constante está definida en segundos, por lo tanto, si su valor es 30x60, comprobará cada media hora si un sensor debiera cambiar de estado.

La segunda está relacionada con el valor del parámetro “frequency_measure”. Este parámetro aparece en el Código 23 y el usuario puede asignarle el valor en el microservicio “GUI Sensor Configuration” como se muestra en la Ilustración 96 o la Ilustración 97. La constante determina el número de periodos entre medidas, es decir, de “frequency_measure” que deben transcurrir para considerar que el sensor debería pasar del estado activo al inactivo. Por ejemplo, si la constante es 10 quiere decir que debe transcurrir 10 veces el periodo entre mediciones del sensor para que se considere inactivo. Si se aumenta el número, el microservicio permite a los sensores estar más tiempo inactivos antes de enviar una alerta. Si se reduce, ocurrirá lo contrario.

A continuación, en la Ilustración 105 se muestra el diagrama de flujo del microservicio para conocer su funcionamiento:

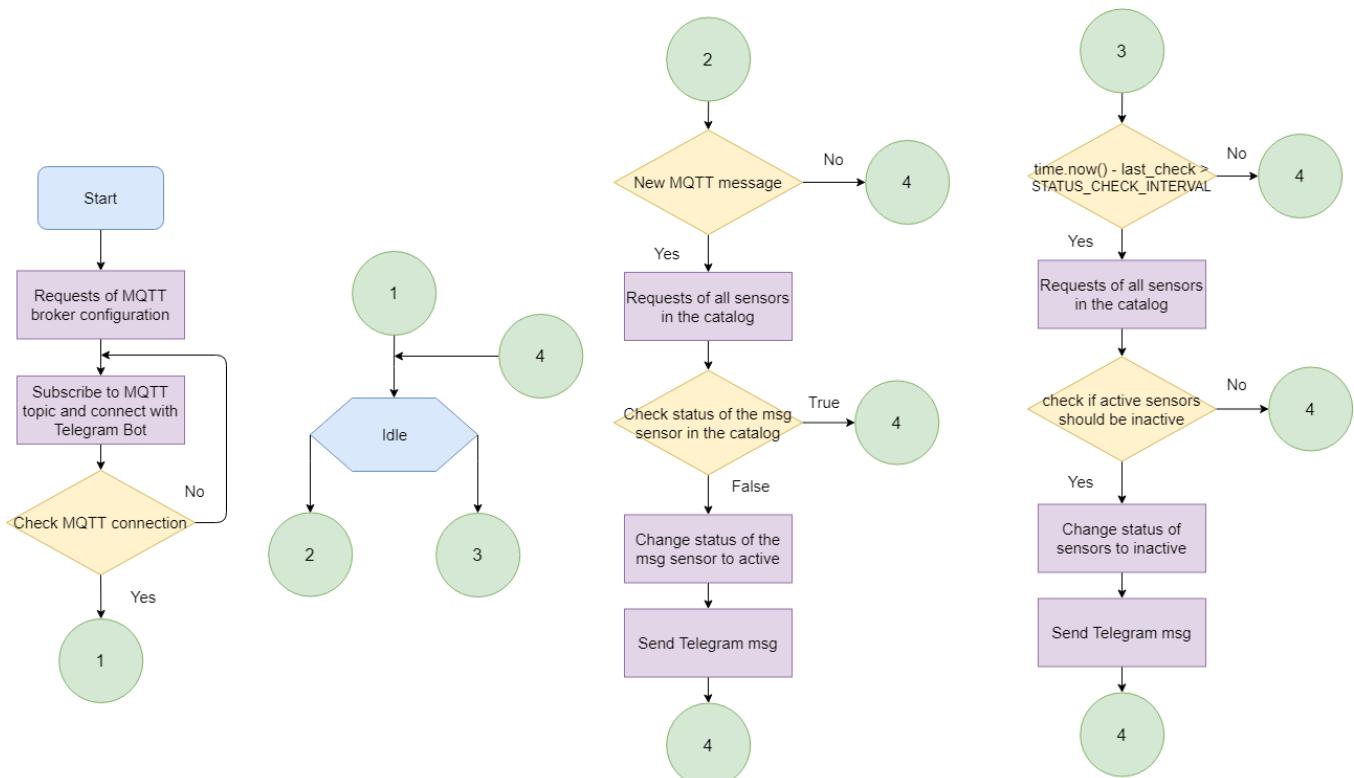


Ilustración 105 - Sensor Status Subscriber flowchart

Este diagrama comienza de nuevo con la configuración inicial de las comunicaciones que corresponde al diagrama de la izquierda. El primer flujo es idéntico al primero que se analiza en la Ilustración 85 a diferencia de la parte final en la que, en vez de esperar a un nuevo mensaje entra en modo de espera ya que se llevan a cabo dos tareas simultáneas. El segundo diagrama se muestra este estado. Las dos tareas que este servicio lleva a cabo corresponden a comparar el estado actual de los sensores con el que está registrado en el catálogo. De esta forma el tercer diagrama analiza si el sensor que en el catálogo está inactivo debería estar activo y el cuarto analiza lo opuesto.

Analizando primero el tercer diagrama, el servicio espera a que llegue un mensaje nuevo a través de MQTT. Cuando este llega, se realiza una petición al catálogo de toda la información almacenada de los sensores. Después se comprueba el estado del sensor que envía el mensaje. En caso de ser activo se vuelve al estado de espera, pero si es inactivo se lleva a cabo la modificación. Para ello, se hace una petición al catálogo para asignar al estado del sensor el valor 'True' o activo. Despues se envía un mensaje a través de Telegram al usuario especificando que sensor se ha activado. Una vez se ha llevado a cabo este proceso, se vuelve al estado de espera.

Por otro lado, si se analiza la segunda tarea que corresponde al cuarto diagrama, se trata de un temporizador que activa la tarea cuando pasa un tiempo establecido por el usuario en la constante 'STATUS_CHECK_INTERVAL'. Para ello, el servicio compara el tiempo establecido por el usuario en la constante con el tiempo actual menos la última vez que se llevó a cabo la tarea. En caso de que haya transcurrido el tiempo asignado en la constante, o lo que es lo mismo, que el segundo término que se acaba de definir sea mayor que la constante, la tarea se lleva a cabo. En caso de que no sea así, el servicio se queda en estado de reposo.

Cuando se cumple la condición, lo primero se toma toda la información de los sensores en el catálogo. Como esta tarea comprueba el estado de los sensores que en el catálogo están activos, pero deberían estar inactivos, solo es necesario comprobar los sensores que están activos, reduciendo los tiempos de búsqueda. Además, para que

esta tarea tenga sentido ejecutarla, es necesario que el usuario haya establecido el valor de la propiedad “frequency_measure” del sensor en el catálogo como se muestra en la Ilustración 96 o la Ilustración 97. También su última conexión con el servidor debe ser mayor que 0 según el formato Unix timestamp, ya que si no significa que el usuario introdujo manualmente el sensor, pero nunca se ha conectado a la plataforma, por lo que no tiene sentido analizarlo. Si se cumplen todas las condiciones descritas hasta ahora, entonces se comprueba si el sensor debiera estar inactivo.

Para llevar a cabo esta comprobación, se compara la hora actual con la última conexión del sensor al servidor más un margen de tiempo que se le concede al sensor en caso de que por problemas de red no haya podido enviar mensajes. Este margen de tiempo viene definido por la variable “frequency_measure” multiplicada por la constante ‘STATUS_CHECK_TIMES’ f. La “frequency_measure” representa el tiempo que pasa entre que el sensor envía mensajes y la constante representa el número de mensajes que el sensor debería haber enviado. Por lo tanto, juntas representan el tiempo que lleva al sensor enviar ese número de mensajes y por consiguiente es el margen de tiempo que se le concede al sensor.

En caso de que la hora actual sea superior al segundo término, el sensor debería estar en estado inactivo. Por ello, se realiza una petición al catálogo para cambiar su estado y se envía un mensaje a través de Telegram al usuario especificando que sensor ha pasado a inactivo y cuánto tiempo lleva inactivo en minutos. Después de esto, la tarea termina y vuelve al estado de reposo esperando que se cumpla alguna de las condiciones iniciales para empezar una de las tareas.

5.2.8 Servidor: Adaptador de Telegram

Se trata del último bloque del servidor el cual es utilizado por el resto de los bloques para enviar avisos al usuario a través de la aplicación móvil o web Telegram. Para transmitir el mensaje desde los otros bloques hasta el móvil, es necesario de un adaptador que sea capaz de comunicarse con ambos actores. Para ello, Telegram dispone de una API para Python que permite comunicarse y manipular lo que se conoce como Bot.

El Bot funciona como una interfaz entre servidor y aplicación. Para comunicarse con el Bot desde la parte del servidor es necesario realizar peticiones HTTPS y conocer su token e identificador. Desde la aplicación simplemente es necesario entrar en el grupo en el que esté el Bot, de forma que varios usuarios puedan recibir avisos sobre la plataforma.

En el servidor hay tres microservicios que utilizan este bloque para enviar avisos. Estos son: “Sensor Configuration Subscriber” del bloque Adaptador y los dos microservicios del bloque de Estrategias de control. Para hacer uso del microservicio “Telegram Bot” cada uno de los bloques debe incorporar el archivo “TelegramBot.py” que se puede observar en la Ilustración 83 y la Ilustración 103. Este archivo contiene lo que se conoce en Python como un objeto que permite la comunicación con el Bot de Telegram. Este archivo necesita que se instalen previamente unos módulos de Python para funcionar. Uno de ellos ya se ha explicado con anterioridad, el módulo requests (Código 26) y el otro es telebot. Para instalarlo es necesario introducir el Código 36 en la terminal del servidor.

```
$ pip3 install pyTelegramBotAPI
```

Código 36 - Telebot Python module installation Linux command

Cada vez que un microservicio crea una instancia del objeto de “Telegram Bot” es necesario que se inicialice pasándole como argumento la URL del catálogo. Se debe a

que el objeto de “Telegram Bot” necesita realizar la petición GET 5 de la Tabla 13 al catálogo para obtener la identificación y token del Bot que permiten establecer la comunicación desde el servidor. Se ha escogido que la dirección del catálogo se pase como argumento para no tener que escribir en el código del archivo esta dirección, reduciendo así el lugar donde debería modificarse dicha dirección en caso de que se vea modificada.

Para crear el Bot de Telegram es necesario de disponer de la aplicación móvil o web. Desde esta se tiene que buscar y empezar una conversación con el Bot llamado “BotFather”. Enviando el comando “/help” se mostrarán todos los comandos que acepta el Bot, apareciendo entre ellos el de “/newbot” que permite crear un Bot nuevo. Si se siguen los pasos que se muestran se obtiene el token del Bot que debe ser añadida al servidor para que funcione.

El siguiente paso es añadir el Bot al grupo en el que se quiere que se den los avisos y obtener la identificación del chat. Para ello, se debe enviar un comando al grupo mencionando al Bot como puede ser: “/my_id @<nombre_Bot>”. Tras esto, se debe acceder a la dirección del Código 37 desde un navegador sustituyendo “<Bot_token>” por el token del Bot que se ha creado.

`https://api.telegram.org/bot<Bot_Token>/getUpdates`

Código 37 - Telegram chat id URL request

Esto mostrará un mensaje en formato JSON del cual se tiene que tomar el parámetro “id” que está dentro de la clave de diccionario “chat”. En el Código 38 se muestra un fragmento del mensaje JSON que puede mostrarse y cuál es el valor que se debe tomar en rojo.

```
"chat": {
    "id": -259326713,
    "title": "Sensor Warning group",
    "type": "group",
    "all_members_are_administrators": true
}
```

Código 38 - Telegram chat id JSON message

Una vez ya se dispone del token y del identificador del chat, se puede hacer uso del microservicio “GUI Telegram Configuration” del bloque GUI para almacenar en el catálogo los valores del Bot de Telegram necesarios para que el servidor funcione correctamente.

A continuación, se muestran como ejemplo los mensajes que se pueden recibir de cada uno de los microservicios que se han analizado a lo largo del apartado del servidor. En este caso el Bot se llama “Sensor warning” y el identificador del sensor es “00124b001b7397df”.

- Sensor Configuration Subscriber

En este caso existen dos tipos de mensajes, el primero cuando un nuevo sensor se añade a la plataforma y el segundo cuando alguna configuración del sensor se modifica.

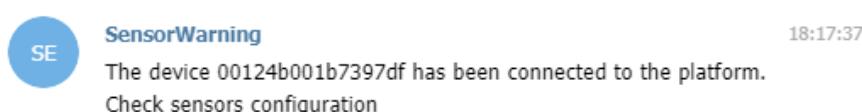


Ilustración 106 - Sensor Config Subs. Add new sensor Telegram message

The device 00124b001b7397df has been modify. Check sensors configuration 18:22:00

Ilustración 107 - Sensor Config Subs. modify sensor Telegram message

- Batery Alarm Subscriber

En este microservicio también hay dos tipos de mensajes. El primero se realiza cuando el nivel de batería se encuentra entre el 30% y 25%. El segundo se realiza cuando el nivel de batería está entre 25% y 20%. En ambos mensajes se muestra la identificación del sensor, el nivel de batería actual del sensor, la estimación de la duración de la batería y al pulsar sobre el botón “DATA”, Telegram te redirige a Grafana desde la que se puede ver las gráficas de los sensores.

The device 00124b001b7397df has lower batery: 28 %. It will go out at 2019-07-17 18:22:42. Charge the device and check Grafana 19:02:16

DATA

Ilustración 108 - Batery Alarm Subs. low batery first advise Telegram msg

The device 00124b001b7397df has lower batery: 22 %. It will go out at 2019-07-17 18:22:42. Charge the device and check Grafana 18:56:04

DATA

Ilustración 109 - Batery Alarm Subs. low batery second advise Telegram msg

- Sensor Status Subscriber

Este también dispone de dos tipos de mensaje. El primero se manda en caso de que un sensor que estaba inactivo pasa a estado activo. Este caso ocurre cuando un sensor se apaga por batería baja o se desconecta de la red y posteriormente se vuelve a conectar. El segundo es cuando el sensor pasa de activo a inactivo. Ocurre cuando el sensor lleva cierto tiempo sin enviar un mensaje al servidor. Se puede ver cómo además del identificador se muestra cuánto tiempo lleva inactivo.

The device 00124b001b7397df has been activated. Check sensors configuration 18:28:37

Ilustración 110 - Sensor Status warning active status Telegram msg

The device 00124b001b7397df has been more than 30 min inactive, It will be consider inactive. Check sensors configuration

Ilustración 111 - Sensor Status warning inactive status Telegram msg

6. EXPERIMENTOS

6.1 INTRODUCCIÓN

Los experimentos que se llevan a cabo son para comprobar que tanto la plataforma completa, así como sus funcionalidades principales funcionan correctamente. Por lo tanto, las pruebas se definen según lo visto en el capítulo anterior.

Se comienza con pruebas de los elementos físicos, para ir subiendo dentro de las capas de la plataforma hasta llegar al servidor, comprobando que la información se almacena correctamente. Estas pruebas se centran en comprobar el correcto funcionamiento de todos los elementos del proyecto. Es por esto, que quedan fuera de ellas todas las pruebas que intenten sacar al sistema de su funcionamiento ordinario, como podría ser: perdida de conexión, caída del servidor, y derivados. Esto se contemplará en el capítulo de líneas futuras.

Las pruebas se van a dividir en dos capítulos. En este capítulo se desarrolla su objetivo, en el cual se explica el motivo de la prueba, su implementación, en la que se explica cómo se va a llevar a cabo y su resultado esperado. En el siguiente capítulo de resultados, se analizará el resultado obtenido de cada prueba, y en caso de que sea negativo, se propone su solución.

6.2 PRUEBAS

A continuación, se describen las pruebas que se han llevado a cabo para validar la plataforma.

6.2.1 Test de continuidad de PCBs

- [Objetivo](#)

El objetivo de esta prueba es comprobar que no existe ningún cortocircuito ni circuito abierto inesperado en las PCBs después de haber soldado los componentes de las 3 placas del proyecto.

- [Implementación](#)

Para esta prueba, primero se realiza una inspección ocular bajo microscopio de las pistas de las 3 PCBs sin los componentes soldados. Se comprueba que las pistas corresponden con el esquemático del circuito.

Segundo, tras soldar los componentes, se lleva a cabo una segunda inspección ocular que permita detectar posibles fallos en las soldaduras, prestando especial atención a los circuitos integrados por su pequeña separación entre los pines.

Tercero, se realiza una prueba de continuidad con el multímetro para comprobar que no existe ningún cortocircuito entre las pistas y los componentes.

Cuarto, con una fuente de alimentación del laboratorio se alimentan las placas, limitando la fuente en corriente. De esta forma, si no se había detectado algún cortocircuito, ningún componente se verá dañado.

- **Resultado esperado**

Tras realizar los pasos que se describen en el punto anterior, no debería existir ningún circuito abierto ni cortocircuito en las pistas antes de montar los componentes. Posteriormente, la prueba de continuidad debería ser positiva, sin que la pasta de soldar haya cortocircuitado ningún componente. De esta forma, al alimentar la placa con la fuente de alimentación limitada en corriente, la placa debería encenderse sin problemas, y se podría aumentar la corriente hasta quitar la limitación.

6.2.2 Test de carga de firmware

- **Objetivo**

En esta prueba se busca que los firmwares de las placas funcionen correctamente. Para ello, se lleva a cabo pruebas de las diferentes funcionalidades de cada placa para comprobar que el firmware funciona como se espera.

- **Implementación**

Primero se carga un firmware de prueba que sirva para comprobar que las diferentes entradas y salidas básicas funcionan. Esto es un firmware que tras pulsar alguno de los botones, encienda y apague los LEDs. De esta forma, se pueden descartar problemas tanto de hardware, como que se hayan podido estropear en el proceso de soldadura, que la polaridad no sea la correcta o que el MCU se caliente demasiado y por otro lado de software, al no corresponder las salidas con el firmware programado.

Segundo, se carga el firmware específico de cada placa y se comprueba mediante la terminal que su funcionamiento es el esperado.

- **Resultado esperado**

Tras subir el primer firmware a las placas, se espera que los LEDs funcionen acorde a las pulsaciones de los botones y que la temperatura del MCU sea baja para esa carga. Con el segundo firmware, se espera que los LEDs y botones funcionen según lo esperado en el firmware, tal como se ha visto en el capítulo anterior. Además, por la terminal se debería poder observar el estado en el que se encuentra cada placa.

6.2.3 Test de perdida de red del hub central

- **Objetivo**

El objetivo de esta prueba es comprobar si la red de sensores es capaz de reconectarse al hub central automáticamente si este se reinicia o si tras reiniciar un sensor, este vuelve a su red anterior. Esta prueba es importante ya que, en el caso de tener una red con un gran número de sensores, es inviable tener que reconectar manualmente uno por uno todos los sensores en caso de pérdida de red.

- **Implementación**

Primero se tiene que conectar las tres placas a la red eléctrica, abrir la red del colector, que los sensores se conecten a ella y cerrar la red del colector.

Tras esto, el hub central se reiniciará y se esperará unos segundos para que los sensores se vuelvan a reconectar.

Por último, se reiniciará los dos sensores y se esperará unos segundos a que los sensores vuelvan a conectarse, a pesar de que la red del colector está cerrada.

- [Resultado esperado](#)

El resultado esperado es que los tres pasos se completen con éxito. Para comprobar que la prueba es correcta, se debe observar el estado de los LEDs de las 3 placas. Estos deben comportarse tal y como se describen en el capítulo 5, apartado 5.1.1.4.

6.2.4 Test de comunicación entre placas de sensores y hub central

- [Objetivo](#)

El objetivo es comprobar que cada 10 segundos, la información de los sensores es enviada al hub central para comprobar que tanto el firmware como la comunicación a través de Sub-1 GHz es correcta.

- [Implementación](#)

Para ello, tras conectar las placas de sensores a la red del hub central, se debe conectar mediante JTAG el hub central y una placa de desarrollo LaunchPad CC1310 de Texas Instruments. Estas disponen del depurador y de la conversión entre UART y Serial del ordenador. Después conectar mediante micro USB la placa de desarrollo a un ordenador y abrir una terminal para observar los mensajes.

- [Resultado esperado](#)

Tras realizar todos los pasos para montar la prueba, el resultado esperado que sale por pantalla deberían ser unos mensajes como los que se pueden observar en el Código 10. Los mensajes que se muestran en ese código pertenecen al ESP 32 que son los que recibe del MCU CC1310

6.2.5 Test de envío y recepción de mensajes al servidor por parte del hub central

- [Objetivo](#)

En esta prueba se pretende comprobar que el envío de mensajes entre las partes Hardware y Software del proyecto funciona correctamente. Si la prueba es correcta, se demuestra que la comunicación UART entre los MCU es correcta, que el firmware del ESP 32 funciona y que ambos clientes de la comunicación MQTT son capaces de enviar y recibir el mensaje correctamente.

- [Implementación](#)

Para esta, y el resto de las pruebas a partir de ahora, se considera que los sensores se encuentran conectados a la red del hub central, y se va a utilizar siempre el mismo mensaje de ejemplo.

El primer paso es conectar a través del puerto micro USB el MCU ESP 32 a un ordenador y abrir una terminal para poder observar los mensajes que le llegan a través de UART del CC1310 y que el ESP 32 modifica para enviar a través de MQTT.

El segundo paso es encender el servidor y abrir la terminal del microservicio “Sensor Data Subscriber” que se muestra en la Ilustración 80, para poder observar el mensaje una vez llega a la parte del servidor.

- **Resultado esperado**

De esta prueba se espera obtener en la terminal del primer paso el mismo mensaje que se observa en el apartado anterior. Junto a este también debe aparecer el mensaje que se envía al servidor que difiere del anterior en el valor “id” y en la incorporación de la hora “time”, tal y como se muestra en el Código 11.

Del segundo paso, se obtendrá un mensaje como el que envía el ESP 32 junto con el tema en el que se envía El Código 11 y Código 12 debería ser el resultado esperado.

6.2.6 Test de almacenamiento y visualización del mensaje

- **Objetivo**

Con esta prueba se demuestra que el mensaje que llega a la parte del servidor es capaz de almacenarse en las bases de datos y este está accesible para su visualización a cualquier dispositivo. Además, se muestra como el microservicio “Sensor Configuration Subscriber” alerta correctamente ante la entrada de datos de un nuevo sensor.

- **Implementación**

Para ello, con la terminal del microservicio “Sensor Data Subscriber”, se obtiene la primera confirmación de que el mensaje se ha almacenado correctamente en la base de datos de InfluxDB.

Para observar el nuevo sensor que se ha añadido al catálogo, se ejecuta el microservicio “GUI Sensor Configuration” desde el que se puede observar los sensores que existen en la plataforma y sus propiedades.

En cuanto a la visualización, se debe abrir en un navegador la dirección en la que se ejecuta Grafana y abrir el “dashboard” donde se representa la información. Para esta prueba se va a visualizar en dos dispositivos. Un ordenador conectado a la misma red que el servidor y un dispositivo móvil conectado a su propia red de datos, independiente de la del servidor. Esto quiere decir que el servidor es multiplataforma, además de poder visualizarse en cualquier lugar.

Por último, se ejecutará la aplicación de Telegram en la que se tenga contacto con el Bot del proyecto, para poder visualizar los mensajes de alerta del servidor.

- **Resultado esperado**

Siguiendo el orden del punto anterior, primero se observará en la terminal del microservicio un mensaje que muestre el mensaje en el formato que se envía a InfluxDB junto con un mensaje de confirmación al haberlo insertado correctamente.

En cuanto a la GUI, en la lista que se muestra en la Ilustración 95, se mostrará el sensor que acaba de enviar el mensaje al servidor, en el que los parámetros “Status” y “Last Connection” deberán reflejar “True” y la fecha en la que se envió el mensaje respectivamente. Además, deberá mostrarse el “Building” y “Floor” que tenga configurado el ESP 32.

En cuanto a Telegram, un nuevo mensaje deberá llegar al chat en el que muestre que el sensor correspondiente es nuevo y acaba de enviar información al servidor.

6.2.7 Test de alarma de batería

- **Objetivo**

Esta prueba muestra el correcto funcionamiento del sistema de alarma a través de Telegram debido al estado de batería baja de los sensores. Esta prueba involucra al microservicio “Batery Alarm Subscriber”.

- **Implementación**

Debido a que ninguno de los sensores dispone actualmente de una batería operativa, para esta prueba se va a crear unos datos falsos en los que simulen el escenario a describir.

Para ello, se simularán 10 valores de batería. Estos serán: [100, 60, 50, 40, 28, 27, 24, 23, 19, 0]. Para observar que estos valores se han recibido se utiliza un navegador con la dirección de Grafana. Además, se debe abrir la terminal del microservicio en cuestión en la que se mostrará los mensajes.

Se debe abrir el Bot de Telegram para poder observar las situaciones de alerta.

- **Resultado esperado**

Según lo descrito, tras mandar al servidor los valores de batería en Grafana aparecerá una gráfica con esos valores.

En cuanto a la terminal se observará el mensaje que corresponde al estado de batería. Si es mayor de 30, la batería será correcta, entre 30 y 25 aparecerá aviso de batería baja y entre 25 y 20 de batería muy baja. En el valor 0 se mostrará que el sensor no dispone de mediciones de batería.

En Telegram se deberá obtener un mensaje de alerta cuando se envíe el mensaje con valor 28 pero no con 27. De igual forma se espera un mensaje con 24, pero no con 23 ni 19.

6.2.8 Test de estado del sensor

- **Objetivo**

El objetivo de esta prueba es comprobar el correcto funcionamiento del microservicio “Sensor Status Subscriber”. Este microservicio alerta al usuario ante cambios de sensores de activo a inactivo y viceversa, esencial para conocer si los sensores están enviando datos cuando deben.

- **Implementación**

Para realizar esta prueba es necesario utilizar uno de los sensores que ya haya enviado información al servidor. Por lo tanto, se tomará la placa de sensores ambientales que se ha utilizado en pruebas anteriores. Si no se ha realizado ninguna prueba anteriormente, solo es necesario enviar algún mensaje desde alguna de ellas.

El siguiente paso es ejecutar el microservicio “GUI Sensor Configuration”, mostrándose una ventana como la de la Ilustración 95. Se escoge de la lista el sensor

de la prueba y se le modifica el parámetro “frequency_measure”, atribuyéndole el periodo en segundos con el que envía información, en este caso 10 segundos y el parámetro “Status” que se le asigna “False”.

Se debe abrir al mismo tiempo la terminal del microservicio “Sensor Status Subscriber” para observar el mensaje y la conversación con el Bot de Telegram.

A continuación, se debe esperar al menos el periodo de tiempo asignado a la variable “frequency_measure” para que el sensor envíe otro dato y comprobar el paso del sensor de inactivo a activo.

Para comprobar el caso de activo a inactivo, se debe apagar el sensor en cuestión y esperar el intervalo de tiempo asignado en el Código 35. Se recomienda para la prueba asignar intervalos de tiempo cortos para que no demore mucho tiempo.

- **Resultado esperado**

En el primer caso, tras esperar el tiempo que se haya asignado, deberá aparecer por la terminal un mensaje mostrando el cambio de un estado a otro. También en la ventana de la GUI deberá cambiar el parámetro “Status” de “False” a “True” y “Last connection” debe tener la hora de conexión. Por último, se debe recibir un mensaje en Telegram alertando de que el sensor se acaba de conectar.

En el segundo caso, se espera exactamente lo mismo excepto por que el parámetro “Status” cambiará a “False” y los mensajes deberán ser acerca de que el sensor se ha desconectado de la red.

6.2.9 Test de extracción de información

- **Objetivo**

Esta prueba pretende comprobar el correcto funcionamiento de la extracción de información de la base de datos InfluxDB. Para ello, se utiliza el microservicio “GUI Query Data” que permite extraer la información.

- **Implementación**

Para comprobar que la extracción de datos es correcta, se debe simular unos datos de los cuales se conozcan previamente sus valores. En este caso, se tomarán los valores de batería que se insertaron a la base de datos en la prueba del apartado 6.2.7. Si no se ha realizado, cualquier otro dato que se conozca su valor sería válido.

Una vez se tienen los valores insertados en la base de datos se ejecuta el microservicio “GUI Query Data”. Tras iniciar sesión con el usuario y contraseña de la base de datos aparecerá una ventana como la que se muestra en la Ilustración 98. En ella se debe introducir en el campo “Measurements” la palabra “Batery” y en el capo “ID” el identificador del sensor con el que se insertaron los datos. Después se debe pulsar en “Search”. Posteriormente se debe pinchar en la pestaña superior de “File” y seleccionar el directorio donde se desea almacenar la información. Tras esto se debe pulsar en “Export as csv”.

- **Resultado esperado**

El resultado final deberá ser un archivo .csv que contendrá los valores de batería introducidos. Además, durante el proceso se puede observar esos mismos valores, ya que deberán aparecer en la lista inferior de la ventana, así como aparecerá una gráfica que los muestre antes de exportarlos.

7. RESULTADOS Y ANÁLISIS

7.1 INTRODUCCIÓN

En este capítulo se analizan los resultados obtenidos de las pruebas realizadas del proyecto descritas en el capítulo anterior.

Para ello, se exponen los resultados obtenidos de llevar a cabo cada una de las pruebas. En caso de existir fallos, se expone cual es la solución al problema y su resultado final de aplicar dicha solución.

Cabe recalcar que todas las ilustraciones de este capítulo son de elaboración propia. Se ha decidido no utilizar el pie de página para ahorrar espacio.

7.2 PRUEBAS

7.2.1 Test de continuidad de PCBs

- **Resultado**

En el segundo paso tras soldar los componentes, existen fallos de soldadura al aplicar la pasta térmica. Estos errores se encuentran en los pines del MCU CC1310 de la placa de consumo de potencia eléctrica y en integrado “U₄” que se observa en la Ilustración 66.

Además, se encuentran fallos en la placa de sensores ambientales entre los conectores “J₁” y “J₂” de la Ilustración 58 y en la placa de hub central entre los conectores de la Ilustración 74 “J₁” y los conectores del MCU situados a la derecha, ya que colisionan entre ellos. Esto se debe a que no se tuvo en cuenta las dimensiones del cable para la conexión JTAG.

En el tercer paso, se observa un cortocircuito derivado del regulador de tensión que se encuentra en la placa de consumo de potencia, “U₅” en la Ilustración 66, y en la placa hub central, “U₂” en la Ilustración 74, debido a un mal diseño de las pistas. Esto se debe a la confusión en su tipo de encapsulado.

Es necesario de una corrección de los fallos para que la plataforma funcione correctamente.

- **Corrección**

En cuanto al primer problema, se debe repasar con un soldador de punta fina los pines de ambos componentes para eliminar el cortocircuito entre los pines.

Para el segundo problema, en el primer caso, es necesario doblar el conector “J₂” para permitir la conexión del cable. En el segundo caso, es necesario cortar los pines del conector del MCU para poder conectar el cable. Los pines cortados no hacen que el sistema pierda ninguna funcionalidad.

Para el tercer problema solo se necesita desoldar uno de los pines del regulador el cual forma el cortocircuito. Sin embargo, para la placa hub central, se trató de solucionar cortando la pista cortocircuitada, por lo que hubo que soldar un cable, siendo esta solución peor.

Con estos cambios, el resultado final es correcto, disponiendo de las 3 PCBs para su correcto funcionamiento.

7.2.2 Test de carga de firmware

- Resultado

Tras probar el primer firmware, las tres placas responden correctamente ante las entradas de los botones, encendiendo los correspondientes LEDs.

Tras probar el segundo firmware también se obtiene los resultados esperados, obteniendo como salida de las terminales la Ilustración 112 para la placa de sensores ambientales a la izquierda y para la para la placa de consumo de potencia a la derecha. En cuanto a la placa del hub central, el mensaje que se obtiene es el que se observa en la Ilustración 113 que muestra un mensaje de la placa de sensores ambientales.

```

TI Sensor
State Changed: 2
Restarted: 0x3
Channel: 3
State Changed: 4
-----
id:00124b001b7397df
sensor type:2 (BME680 sensor)
battery = 0 [%]
    temperature = 29.3639 [degrees Celsius]
    pressure = 94580.8906 [Pa]
    humidity = 20.0791 [%]
    gas = 23400.0683 [ohms]
-----
TI Sensor
State Changed: 2
Restarted: 0x4
Channel: 3
State Changed: 4
-----
id:00124b001b739a00
sensor type:1 (power meter sensor)
    power = 9 [W]
    voltage = 229.8371 [Vrms]
    current = 0.0401 [Arms]
    frequency = 49.9881 [Hz]
    angle M1 = 0.0000 [deg]
    angle M2 = 0.0000 [deg]

```

Ilustración 112 - Test de carga de firmware. Placas de sensores

```

TI Collector
Restarted
Channel: 3
{"t":27.9099,"p":94576.1599,"h":20.6969,"g":162596.0000,"bat":0,"id":"00124b001b7397df"}
ConfigRsp: 0x3
ConfigRsp: 0x4
{"w":9,"v":226.5199,"i":0.0409,"f":49.9799,"a":0.0000,"id":"00124b001b739a00"}

```

Ilustración 113 - Test de carga de firmware. Placa del hub central

Por lo tanto, el firmware de las 3 placas funciona correctamente, superando la prueba.

7.2.3 Test de perdida de red del hub central

- Resultado

Tras realizar ambas pruebas se obtiene que tanto los sensores como el hub central son capaces de reconectarse a la red sin que el usuario realice ninguna acción. Por lo tanto, la prueba se considera superada. Sin embargo, en ocasiones la placa de consumo de potencia pierde la conexión con el hub central durante unos pocos segundos, aunque a pesar de ello, vuelve a establecer conexión con el hub central sin problema.

7.2.4 Test de comunicación entre placas de sensores y hub central

- Resultado

De esta prueba se obtiene el mismo resultado que se observa en la Ilustración 113. En esta aparecen reflejados los mensajes de cada placa de sensores, siendo el primero el de la placa de sensores ambientales y el segundo el de la placa de consumo de potencia. Los mensajes que aparecen en medio corresponden a mensajes del sistema indicando que un sensor se ha conectado a la red del hub central. Este mensaje es el que se transmite por UART al MCU ESP 32.

A pesar de que esta prueba pudiera parecer redundante, ya que se obtiene el mismo resultado que la anterior, podría darse el caso en el que el resultado fuera distinto. Este sería el caso de una red con más sensores, en el que todos los sensores fueran capaces de obtener la medida de su sensor, pero algunos no la transmitieran al hub central debido a problemas con esa parte del firmware. La prueba anterior mostraría que los sensores funcionan correctamente y al mostrarse la terminal del hub central, daría a entender que también, ya que este es capaz de recibir información. Por lo tanto, esta prueba hace más hincapié en el hecho de que todas las placas de sensores sean capaces de transmitir información. Por ello, esta prueba también se considera superada.

7.2.5 Test de envío y recepción de mensajes al servidor

- Resultado

En la Ilustración 114 se puede observar la terminal del ESP 32. Se observan 6 líneas, las 3 primeras corresponden a un mensaje de la placa de consumo de potencia y las 3 últimas a la placa de sensores ambientales. Se va a analizar las 3 primeras, ya que ambas partes funcionan igual. La primera línea es el mensaje que se recibe del MCU CC1310, que si se observa la Ilustración 113, coincide con el mismo formato, ya que el mensaje enviado es otro. La segunda línea es el tema MQTT en el que se publica el mensaje. Se puede observar cómo se ha incluido el identificador del sensor. La tercera línea corresponde al mensaje que se publica por MQTT. Difiere del de la primera línea en que se le ha incorporado el tiempo "time" y se le ha eliminado el identificador "id".

```
b'{"w":8,"v":233.5999,"i":0.0380,"f":50.0000,"a":0.0000,"id":"00124b001b739a00"}\r\n'
TFM/PRUEBA00/JOSE/GATEWAY/Narciso/5/00124b001b739a00
{'v': 233.5999, 'f': 50.0, 'a': 0.0, 'time': 1567764670, 'i': 0.038, 'w': 8}
b'{"t":26.4599,"p":94847.7999,"h":24.4680,"g":5536.0000,"bat":0,"id":"00124b001b7397df"}\r\n'
TFM/PRUEBA00/JOSE/GATEWAY/Narciso/5/00124b001b7397df
{'g': 5536.0, 'p': 94847.8, 'time': 1567764672, 't': 26.4599, 'bat': 0, 'h': 24.468}
```

Ilustración 114 - Test de envío y recepción de mensajes al servidor. ESP 32

En la Ilustración 115 se puede observar esos mismos mensajes que aparecen en la ilustración anterior. En la primera línea aparecen igual, y posteriormente, entre las líneas 9 y 10 se puede observar como el microservicio ha transformado el mensaje para que InfluxDB pueda almacenarlo. La línea 11 confirma que los datos han sido insertados correctamente. Por lo tanto, se puede concluir que el sistema supera la prueba.

```
received '{"v": 233.5999, "f": 50.0, "a": 0.0, "time": 1567764670, "i": 0.038, "w": 8}' under topic 'TFM/PRUEBA00/JOS
E/GATEWAY/Narciso/5/00124b001b739a00'
{"location": {"building": "Narciso", "room": "NaN", "floor": "5"}}
{'u'building': u'Narciso', 'u'room': u'NaN', 'u'floor': u'5'}
Error: Incorrect sensor type
Error: Incorrect sensor type
Error: Incorrect sensor type
Error: Incorrect sensor type
[{'fields': {'value': 8.0}, 'tags': {'building': 'Narciso', 'ID': '00124b001b739a00', 'room': 'NaN', 'floor': '5'},
'time': '2019-09-06T12:11:10Z', 'measurement': 'Electric power'}]
Data insert correctly
received '{"g": 5536.0, "p": 94847.8, "time": 1567764672, "t": 26.4599, "bat": 0, "h": 24.468}' under topic 'TFM/PRUE
BA00/JOSE/GATEWAY/Narciso/5/00124b001b7397df'
{"location": {"building": "Narciso", "room": "NaN", "floor": "5"}}
{'u'building': u'Narciso', 'u'room': u'NaN', 'u'floor': u'5'}
[{'fields': {'value': 0.0}, 'tags': {'building': 'Narciso', 'ID': '00124b001b7397df', 'room': 'NaN', 'floor': '5'},
'time': '2019-09-06T12:11:12Z', 'measurement': 'Battery'}, {'fields': {'value': 5536.0}, 'tags': {'building': 'Narciso', 'ID': '00124b001b7397df', 'room': 'NaN', 'floor': '5'}, 'time': '2019-09-06T12:11:12Z', 'measurement': 'VOC'}, {'fields': {'value': 24.468}, 'tags': {'building': 'Narciso', 'ID': '00124b001b7397df', 'room': 'NaN', 'floor': '5'}, 'time': '2019-09-06T12:11:12Z', 'measurement': 'Humidity'}, {'fields': {'value': 94847.8}, 'tags': {'building': 'Narciso', 'ID': '00124b001b7397df', 'room': 'NaN', 'floor': '5'}, 'time': '2019-09-06T12:11:12Z', 'measurement': 'Temperature'}]
Data insert correctly
```

Ilustración 115 - Test de envío y recepción de mensajes al servidor. Sensor Data Subscriber

7.2.6 Test de almacenamiento y visualización del mensaje

- Resultado

Como se observa en la Ilustración 115, ambos mensajes de las placas de sensores son insertados correctamente en la base de datos InfluxDB. En cuanto a la visualización de los datos en Grafana se pueden observar varios datos en la Ilustración 116 los cuales proceden de la prueba llevada a cabo. Se enviaron más de uno, ya que la red de sensores está diseñada para enviar datos automáticamente, dificultando el proceso de obtención de un único valor. Sin embargo, se puede observar, que los datos mostrados en las ilustraciones anteriores corresponden a los que se observan en Grafana.

En la Ilustración 116 se puede observar el mismo dashboard pero desde un dispositivo móvil que no se encuentra conectado a la misma red que el servidor. Esto demuestra que la plataforma es multiplataforma y ubicua.



Ilustración 116 - Test de almacenamiento y visualización. Grafana en ordenador.

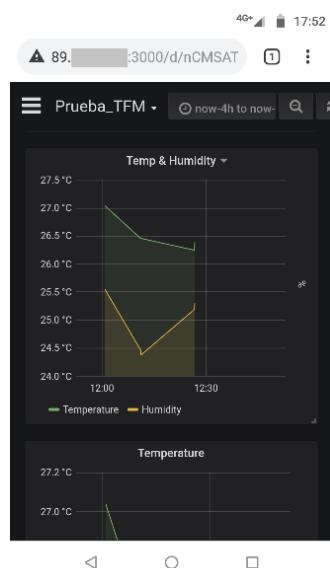


Ilustración 117 - Test de almacenamiento y visualización. Grafana en móvil.

En la Ilustración 118 se muestra los sensores que almacena el catálogo. Como se puede observar, dispone de su identificador, su estado activo y su última conexión.

ID	Status	Last Connection	Building	Floor	Room	Freq ...
00124b001b739a00	True	2019-09-06T12:11:10Z	Narciso	5	Nan	0.0
00124b001b7397df	True	2019-09-06T12:11:22Z	Narciso	5	Nan	0.0

Ilustración 118 – Test de almacenamiento y visualización. Sensor Configuration Subscriber

Por último, se tiene el mensaje que se recibe en Telegram que se observa en la Ilustración 119 en la que se muestra cómo se alerta al usuario ante la conexión por primera vez de los sensores a la plataforma.

```
The device 00124b001b739a00 has been connected to the platform. 12:10:54
Check sensors configuration

The device 00124b001b7397df has been connected to the platform. 12:11:04
Check sensors configuration
```

Ilustración 119 - Test de almacenamiento y visualización. Telegram

Como se ha comprobado, la plataforma supera correctamente la prueba establecida.

7.2.7 Test de alarma de batería

- Resultado

En la Ilustración 120 se observa la terminal del microservicio que se involucra en la prueba. En este se muestran los mensajes generados de manera manual que llegan al servidor. Se ha recortado la terminal a partir del nivel de batería 40, para reducir el espacio de la imagen, ya que los mensajes posteriores a este generan la misma salida. Cuando se alcanza el valor de batería 28, el sistema alerta por la terminal de que el sensor se encuentra con batería baja. Además, se envía un mensaje por Telegram para avisar del estado, como se muestra en la Ilustración 121. Sin embargo, en el valor 27 no se envía dicho mensaje, ya que con una alerta al usuario mediante Telegram es suficiente.

Cuando se supera el siguiente límite con el valor 24, el sistema de nuevo alerta por terminal y por Telegram, obviando el valor de 23 e inferiores. Cuando llega al valor de 0, por terminal se indica que el sensor en cuestión no tiene batería, ya que el 0 y los valores negativos se reservan para indicar que el sensor no dispone de ese recurso.

```
received '{"bat": 40, "g": 42671.0, "h": 23.3497, "p": 97030.41, "t": 23.15, "time": 1567767256.102}' under topic 'TFM/PRUEBA00/JOSE/GATEWAY/BD1/4/001'
Batery level correct
received '{"bat": 28, "g": 115097.0, "h": 21.4844, "p": 97082.7, "t": 21.92, "time": 1567767266.109}' under topic 'TFM/PRUEBA00/JOSE/GATEWAY/BD1/4/001'
Batery level low (30-25%)
2019-09-06T12:54:26Z
received '{"bat": 28, "g": 18394.0, "h": 24.1994, "p": 96765.86, "t": 24.0, "time": 1567767276.115}' under topic 'TFM/PRUEBA00/JOSE/GATEWAY/BD1/4/001'
Batery level low (30-25%) but warned
received '{"bat": 24, "g": 61259.0, "h": 20.2167, "p": 97128.95, "t": 21.29, "time": 1567767286.121}' under topic 'TFM/PRUEBA00/JOSE/GATEWAY/BD1/4/001'
Batery level low (25-0%)
2019-09-06T12:54:40Z
received '{"bat": 23, "g": 120198.0, "h": 28.6062, "p": 97264.43, "t": 29.9, "time": 1567767296.127}' under topic 'TFM/PRUEBA00/JOSE/GATEWAY/BD1/4/001'
Batery level low (20-0%) but warned
received '{"bat": 19, "g": 120610.0, "h": 28.6679, "p": 95843.39, "t": 24.2, "time": 1567767306.134}' under topic 'TFM/PRUEBA00/JOSE/GATEWAY/BD1/4/001'
Batery level low (25-0%) but warned
received '{"bat": 0, "g": 87195.0, "h": 24.7624, "p": 95402.69, "t": 29.73, "time": 1567767316.142}' under topic 'TFM/PRUEBA00/JOSE/GATEWAY/BD1/4/001'
This sensor does not have batery
```

Ilustración 120 - Test de alarma de batería. Batery Alarm Subscriber

```
The device 001 has lower batery: 28 %. It will go out at 2019-09-06T12:54:26Z. Charge the device and check Grafana 12:54:31
```

DATA

```
The device 001 has lower batery: 24 %. It will go out at 2019-09-06T12:54:40Z. Charge the device and check Grafana 12:54:50
```

DATA

Ilustración 121 - Test de alarma de batería. Telegram

Como última muestra de que la prueba es correcta, en la Ilustración 122 se muestran los valores en Grafana.

*Ilustración 122 - Test de alarma de batería. Grafana*

7.2.8 Test de estado del sensor

- Resultado

Con el sensor transmitiendo información al servidor se debe modificar los parámetros tal y como se muestra en la Ilustración 123. Esto es, establecer la placa de sensores ambientales en el catálogo como que está inactiva.

ID	Status	Last Connection	Building	Floor	Room	Freq ...
00124b001b739a00	True	2019-09-06T12:11:10Z	Narciso	5	Nan	0.0
00124b001b7397df	False	2019-09-06T12:26:44Z	Narciso	5	Nan	10.0

Ilustración 123 - Test de estado del sensor. GUI Sensor Configuration inactive sensor

Dicha placa realmente está transmitiendo información continua al servidor. Por lo que, tras esperar el tiempo que tarda la placa en enviar nueva información, el microservicio que se muestra en la Ilustración 124 indica que el sensor ha cambiado de inactivo a activo, que es como realmente debe estar. Esto se puede ver reflejado en la Ilustración 125, en la que el sensor vuelve a estar activo y con su última conexión.

Para comprobar el caso contrario, se debe esperar el tiempo establecido para la prueba. Como se ve, tras pasar dicho tiempo, en la última línea se muestra como la plataforma cambia el estado al sensor al comprobar que ya no está conectado a la plataforma.

```
running Subscriber Server_SSW
subscribing to TFM/PRUEBA00/JOSE/GATEWAY#
received '[{"g": 131245.0, "p": 94825.39, "time": 1567765598, "t": 26.25, "bat": 0, "h": 25.1929}' under topic 'TFM/PRUEBA00/JOSE/GATEWAY/Narciso/5/00124b001b7397df'
Checking sensors change from inactive to active
Sensor 00124b001b7397df has change from inactive to ACTIVE
received '[{"g": 7652.0, "p": 94822.78, "time": 1567765604, "t": 26.39, "bat": 0, "h": 25.2999}' under topic 'TFM/PRUEBA00/JOSE/GATEWAY/Narciso/5/00124b001b7397df'
Checking sensors change from inactive to active
Checking sensors change from active to inactive
Checking sensors change from active to inactive
Checking sensors change from active to inactive
Sensor 00124b001b7397df has change from active to INACTIVE
```

Ilustración 124 - Test de estado del sensor. Sensor Status Subscriber

ID	Status	Last Connection	Building	Floor	Room	Freq ...
00124b001b739a00	True	2019-09-06T12:11:10Z	Narciso	5	Nan	0.0
00124b001b7397df	True	2019-09-06T12:26:44Z	Narciso	5	Nan	10.0

Ilustración 125 - Test de estado del sensor. GUI Sensor Configuration active sensor

Junto con lo descrito anteriormente, cada vez que la plataforma detecta que el sensor cambia de estado, el microservicio emite un mensaje por Telegram, tal y como se puede observar en la Ilustración 126.

SensorWarning 12:26:41
The device 00124b001b7397df has been activated. Check sensors configuration

The device 00124b001b7397df has been more than 1.666666666667 min inactive, It will be consider inactive. Check sensors configuration 12:29:21

Ilustración 126 - Test de estado del sensor. GUI Sensor Configuration inactive sensor. Telegram

7.2.9 Test de extracción de información

- Resultado

En la Ilustración 127 se muestra el resultado de llevar a cabo los pasos de la prueba en el microservicio indicado. Se puede observar que este avisa mediante la lista y mediante un gráfico cuales son los valores que el usuario pretende extraer. Se trata de una función muy útil para ahorrar tiempo al usuario a la hora de evitar errores al extraer datos.

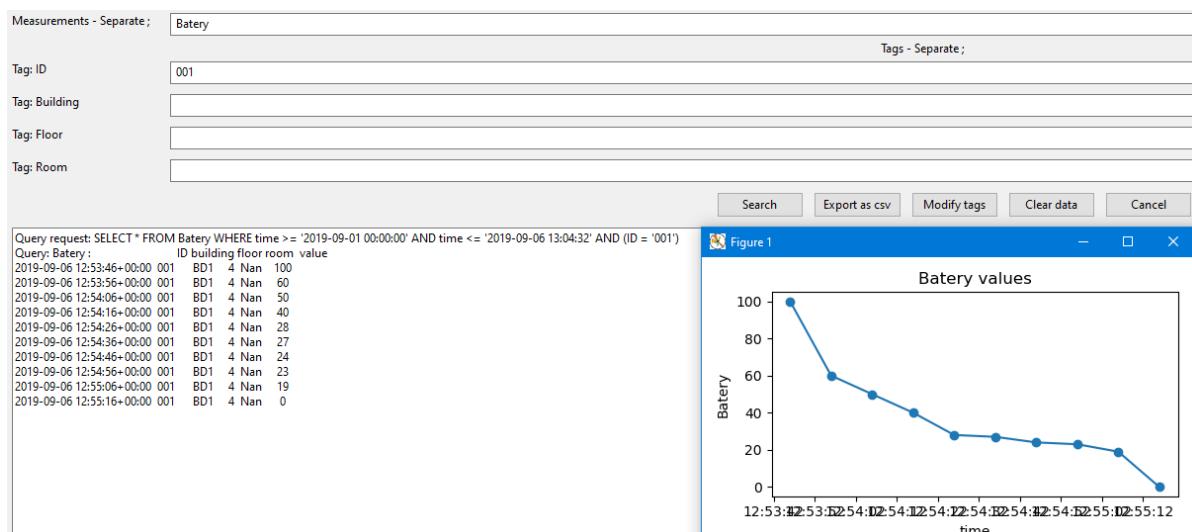


Ilustración 127 - Test de extracción de información. GUI Query Data.

En la Ilustración 128 se muestra el resultado final que se obtiene. Este es un archivo .csv con todos los valores que previamente se habían seleccionado.

	A	B	C	D	E	F
1	,ID,building,floor,room,Batery,time					
2	2019-09-06 12:53:46+00:00,001, BD1, 4, Nan, 100, 2019-09-06 12:53:46+00:00					
3	2019-09-06 12:53:56+00:00,001, BD1, 4, Nan, 60, 2019-09-06 12:53:56+00:00					
4	2019-09-06 12:54:06+00:00,001, BD1, 4, Nan, 50, 2019-09-06 12:54:06+00:00					
5	2019-09-06 12:54:16+00:00,001, BD1, 4, Nan, 40, 2019-09-06 12:54:16+00:00					
6	2019-09-06 12:54:26+00:00,001, BD1, 4, Nan, 28, 2019-09-06 12:54:26+00:00					
7	2019-09-06 12:54:36+00:00,001, BD1, 4, Nan, 27, 2019-09-06 12:54:36+00:00					
8	2019-09-06 12:54:46+00:00,001, BD1, 4, Nan, 24, 2019-09-06 12:54:46+00:00					
9	2019-09-06 12:54:56+00:00,001, BD1, 4, Nan, 23, 2019-09-06 12:54:56+00:00					
10	2019-09-06 12:55:06+00:00,001, BD1, 4, Nan, 19, 2019-09-06 12:55:06+00:00					
11	2019-09-06 12:55:16+00:00,001, BD1, 4, Nan, 0, 2019-09-06 12:55:16+00:00					

Ilustración 128 - Test de extracción de información. Archivo .csv.

8. CONCLUSIONES

Si se observan los objetivos fijados en el capítulo 3, tanto el objetivo general como los específicos han sido cumplidos obteniendo una plataforma IoT completa que cumple con las propiedades de flexibilidad, modularidad, escalabilidad, multiplataforma, ubicuidad y robustez, que son las propiedades fundamentales de una plataforma IoT. No obstante, existen ciertas funcionalidades y mejoras que sería recomendable implementar en la plataforma que no entran dentro de los objetivos establecidos. Estas se desarrollan en el capítulo siguiente de líneas futuras.

Se ha logrado desarrollar dos placas de sensores que permiten obtener parámetros medioambientales y de consumo eléctrico que son capaces de funcionar correctamente en un amplio rango de situaciones. Gracias a la placa del hub central, se puede transmitir toda esta información a un servidor donde se puede gestionar y monitorizar. El desarrollo de una placa central permite agilizar el futuro de desarrollo de más placas de sensores, al dividir el trabajo en dos partes. Con esta parte del trabajo se ha logrado conocer en detalle el funcionamiento de los MCU y sus comunicaciones tanto físicas como inalámbricas. Además, se han adquirido conocimientos de diseño y ensamblaje de PCBs, fundamentales en la especialidad en la que se desarrolla el Máster.

En cuanto a la parte del servidor, se ha desarrollado un servidor desde la base, estableciendo la arquitectura que más se adapte a las necesidades del proyecto. Todo esto se ha realizado con softwares gratuitos y en muchos casos de software libre, intentando reducir costes y apostando por esa filosofía. De esto, se ha obtenido un servidor capaz de almacenar y visualizar toda la información de la red de sensores, aspectos fundamentales para la monitorización y mejora de la eficiencia energética en el hogar. Gracias a esto se ha aprendido sobre las estructuras de web services y microservicios, sector que está en auge en la actualidad debido a su importancia en la gestión de datos.

Observando las pruebas realizadas, se puede concluir que la plataforma entera cumple con las especificaciones establecidas. Solo hay una prueba en la que el proyecto presenta errores, pero estos han sido solucionados con éxito. Esto deja una plataforma totalmente funcional que cumple con los objetivos establecidos.

Además de la aplicación del proyecto, este mismo se puede utilizar en hospitales para la monitorización continua de los espacios en los que se debe tener unos umbrales estrictos de confort en las habitaciones. Otra aplicación es en invernaderos, almacenes alimenticios o de objetos que deban cumplir ciertas condiciones ambientales, en el que es necesario conocer en todo momento el estado de los parámetros ambientales o si existen apagones en la red eléctrica.

No se puede olvidar las implicaciones económicas, sociales y medioambientales del proyecto. Gracias a la obtención de los parámetros medioambientales y de consumo, el usuario dispone de la información actual, pasada, y en algunos casos, futura, al estimar cual será la evolución de los datos. Esto permite tanto al usuario que reside en la vivienda como a las empresas constructoras que hagan uso del sistema, establecer patrones para mejorar la eficiencia energética y por tanto contribuir positivamente en los impactos medioambientales.

A nivel de usuario, este puede establecer patrones de consumo que le permitan ahorrar en calefacción y electricidad. También se puede establecer un sistema de alarma en caso de pérdida de electricidad para no perder alimentos por desconocimiento de la situación. A nivel de empresa constructora, le permite conocer el

grado de mejora en eficiencia al realizar una reforma. Esto se puede traducir en una menor utilización de materiales para obtener los mismos beneficios energéticos.

Ligado a las mejoras en el impacto medioambiental están las mejoras económicas al utilizar los recursos de una manera más eficiente, ayudando a familias a ahorrar en las facturas y a las empresas a utilizar menos recursos.

Sin embargo, este tipo de tecnología también conlleva unos impactos negativos tanto medioambientales como sociales. En cuanto a los medioambientales, el fabricar una red de cientos de sensores conlleva un gasto de recursos materiales y energéticos que, como se veía en la introducción del proyecto, se va a ir multiplicando con los años. En cuanto al impacto social, estas redes de sensores darán una sociedad totalmente conectada y dependiente de la tecnología y la información, con sus aspectos negativos como la violación de la intimidad y privacidad.

Para minimizar estos impactos, es trabajo de los gobiernos y de la sociedad contribuir a un desarrollo sostenible. Para ello se debe educar y establecer leyes en materia de utilización de recursos y su posterior gestión de residuos, así como establecer mecanismos de manera transparente para que la sociedad conozca como se están utilizando sus datos.

En esta línea ya se han realizado iniciativas por parte de las Naciones Unidas, desarrollando unos Objetivos de Desarrollo Sostenible (ODS). Estos objetivos globales tratan de erradicar problemas de carácter global y proponen medidas para que el planeta prospere de manera sostenible para asegurar el futuro a las generaciones venideras. Se busca obtener estos objetivos para 2030. En la Ilustración 129 se pueden observar cuales son estos objetivos.



Ilustración 129 - Sustainable Development Goals⁴²

Con este Proyecto se busca colaborar de manera activa con los ODS. De esta forma se considera que el proyecto colabora principalmente con los objetivos 11, 12 y 13 de la Ilustración 129 ya que mejora y conciencia con un consumo responsable, ayuda a la creación de ciudades sostenibles y como ya se ha visto en este apartado, tiene impactos positivos en el clima.

⁴² (Naciones Unidas, 2019)

9. LINEAS FUTURAS

Como se ha detallado en el capítulo anterior, los objetivos establecidos del proyecto han sido satisfechos. Sin embargo, existen ciertos elementos y funcionalidades que se pueden desarrollar para mejorar la monitorización de la eficiencia energética en la vivienda.

Los puntos que se van a desarrollar a continuación no se han podido completar en el marco de este proyecto debido a la falta de tiempo, ya que los objetivos del proyecto eran muy amplios, y debido a que algunos surgen durante el desarrollo del proyecto y no se dispone de los medios para solucionarlos. Además, para el desarrollo del proyecto no se partía de ningún proyecto anterior realizado en el laboratorio. Esto quiere decir que este proyecto sienta las bases para que futuros estudiantes sigan implementando funcionalidades y elementos para completar la plataforma.

Por lo tanto, se trata una prueba conceptual para que estudiantes más especializados como puede ser en el desarrollo de circuitos o en servidores partan de una base a la hora de realizar su tesis. Así pues, es necesario que existan unas líneas futuras para posibles mejoras.

Como se ha realizado a lo largo del proyecto, se va a dividir las líneas futuras en la parte Hardware y la parte Software, comenzando con la primera. Primero se van a contemplar las mejoras a las placas ya desarrolladas, para luego aportar nuevas ideas.

A pesar de no ser un problema crítico ya que con las soluciones aportadas se pueden corregir, se debería rediseñar las PCBs solucionando los problemas de colisión entre los pines y de diseño del regulador de tensión. Esta mejora es más bien estética que funcional ya que las placas funcionan correctamente.

En cuanto a la placa de sensores ambientales, se debería soldar el componente que administra el nivel de batería y probar su funcionamiento con una batería. De esta forma esta placa dispondrá de todas sus funcionalidades. En esta línea, se deberá realizar pruebas acerca de la duración de la batería a través del conector que está implementado en la placa para conocer su consumo. Esta prueba también es recomendable realizarla con el resto de las placas para conocer sus consumos. En caso de que sean elevados se deberá rediseñar el firmware para introducir modos de bajo consumo.

En cuanto a la placa del HUB central, se le podría añadir más funcionalidades. Entre ellas están: la adición de una pantalla que muestre información básica de la plataforma que permita conocer al usuario el estado de la red y de los sensores. Implementar una batería a la placa de forma que en caso de pérdida de conexión a Internet pueda almacenar la información, o que en caso de pérdida de energía en la casa avise al usuario de tal evento. También se recomienda utilizar modos de bajo consumo en el ESP 32.

Además de implementar esas funcionalidades, se plantean cambios que transformarían por completo al HUB central. El primero sería eliminar la placa de desarrollo ESP 32 y utilizar solamente el MCU ESP 32. Esto ahorraría consumos y dinero, pero hace del proceso de soldadura más complicado. El segundo sería eliminar por completo el MCU ESP 32 y sustituirlo por un módulo Ethernet como puede ser el módulo ENC28J60-II de Molinex que conecta al MCU CC1310 a Internet. Esto ahorraría costes y tiempo en el diseño y compra del ESP 32 pero toda la carga de trabajo la tendría el ESP 32, además de que el HUB central debe colocarse en un sitio que disponga de toma por cable Ethernet.

Para poder obtener todos los parámetros relevantes para una completa monitorización de la eficiencia energética es necesario desarrollar nuevas placas con diferentes sensores. Para el desarrollo de estas placas se recomienda utilizar los esquemáticos y diseños del proyecto ya que, si se utiliza el MCU CC1310 la mayoría de las secciones serán iguales. De esta forma se recomienda desarrollar sensores que permitan obtener:

- Temperatura, humedad y presión: Además del sensor empleado, también se pueden utilizar otros sensores como el SHT31, el HDC2010 de temperatura y humedad, o como ya se ha visto el BME 280 de menor precisión y rango.
- Radiación solar: Al conocer la radiación solar incidente se podrán conocer cuánto se pueden calentar ciertas partes de la casa. Además, es un parámetro muy relevante si se busca utilizar placas solares.
- Consumo de agua: Conocer el consumo de agua en una vivienda puede concienciar acerca del uso que el usuario está realizando y de esta forma intentar que este reduzca su consumo.
- Consumo de energía térmica: Mediante sensores de temperatura colocados en los radiadores se puede conocer la energía térmica consumida. Esta información es muy relevante, ya que se puede conocer cuánto se consumía antes y después de realizar una reforma en la vivienda.
- Velocidad del viento: Conocer la velocidad del viento interna es relevante en el caso de disponer de sistemas de aire acondicionado o de recirculación del aire. Para ello se recomienda utilizar anemómetros o medidas indirectas a través de los gases o de la presión del aire a través de ultrasonidos. En cuanto a la velocidad del viento exterior permite conocer la sensación térmica exterior, así como realizar mejores predicciones climatológicas.
- Ventana o puerta abierta: Permite tener monitorizadas las ventanas y puertas por motivos de seguridad y para conocer si existe ventilación en ciertas salas de la vivienda. Esto puede ser útil en colaboración de otros sensores de viento, calidad del aire y actuadores.
- Detectores de presencia: Mediante el conocimiento de la presencia en ciertas habitaciones se pueden automatizar procesos para calentar o ventilar ciertas partes de la vivienda más que otras al realizar patrones de conducta de los habitantes.
- Calidad del aire: Se debería obtener mediciones de CO₂ y de partículas ya que se consideran elementos contaminantes en ciertas cantidades. Para ello, se recomienda los sensores: CCS811 para la obtención del CO₂ y el Sharp GP2Y1010AU0F para las partículas.
- Luminosidad: Para la obtención de la cantidad de luz se recomienda el sensor TSL2561. Si se busca obtener también la cantidad de rayos ultravioleta se recomienda el sensor VEML6075.
- Actuadores: Esto supondría modificar gran parte de la plataforma ya que el servidor se convertiría en emisor de órdenes a la plataforma. Sin embargo, esta implementación permitiría tener control sobre elementos de calefacción y de esta forma reducir consumos.

En cuanto a las líneas futuras que corresponden a la parte Software, de nuevo se va a plantear mejoras en funcionalidades o pruebas a realizar para luego aportar nuevas líneas de mejora.

Se recomienda realizar más pruebas acerca la conexión y reconexión tanto del ESP 32 y del bloque Adaptador del servidor en caso de pérdida del Broker MQTT o de la red WiFi. En caso de que esto ocurra, el ESP 32 debería almacenar los datos de los sensores en algún archivo temporal y en cuanto al Adaptador, debería notificar al usuario del evento. En esta línea se aconseja incorporar al servidor el módulo de Python logger que permite generar un registro de los eventos que ocurren en el servidor y de la red de sensores de forma que se facilite la corrección de errores.

También se recomienda modificar el bloque del Adaptador en el servidor. A pesar de que se generaría más archivos .py y se repetiría partes del código, se aconseja dividir los microservicios actuales en microservicios aún más pequeños, de forma que un microservicio se encargue de almacenar cada una de las medidas de los sensores. Esto hace que se consuma más tiempo en el desarrollo de la aplicación, pero ahorra tiempo en la identificación y corrección de errores. Además, a la hora de incorporar otra medida, solo es necesario copiar y pegar un microservicio modificándolo para la nueva medida.

Grafana es un servicio de visualización de datos que permite instalar una gran cantidad de plugins desarrollados por la propia comunidad. El uso de estos plugins puede aumentar las funcionalidades a la hora de representar datos o incluso añadir nuevos datos como puede ser conocer la temperatura que aparece en internet de la zona, hora, mapa de precipitaciones, localización de la plataforma, etc.

Se debería incluir más usuarios a la base de datos InfluxDB. De esta forma asignar un usuario con unos privilegios determinados a cada microservicio para aumentar la seguridad. Por ejemplo, asignar un usuario de solo lectura a Grafana puede evitar problemas de seguridad ya que Grafana solo necesita leer los datos de la base de datos.

En la misma línea de aumentar la seguridad de la plataforma, se debería incorporar cifrados a las comunicaciones. Entre estas comunicaciones se encuentran los diferentes bloques con InfluxDB y la transmisión de datos por MQTT. Ambos servicios permiten la habilitación de certificados SSL y MQTT además incorpora cifrado TSL.

Para poder implementar el cifrado en la comunicación MQTT es necesario utilizar un Broker privado. Para ello, se necesita disponer de un servicio de hosting que aloje el Broker privado. Además, el disponer de un hosting o servidor privado, permite alojar el servidor en vez de en la Raspberry Pi 3 en la nube real. Esto ahorra costes y tiempos en configurar el servidor. También añade seguridad al saber que el servidor siempre estará disponible. En el proyecto se han comentado varios servicios de servidores que podrían ser válidos para este cometido.

Ya que la visualización de los datos en Grafana a través de un teléfono móvil es poco práctica se recomiendan dos soluciones. La primera, crear un dashboard paralelo al original que solo muestre el último valor de las variables medidas. Esta solución es más sencilla y fácilmente aplicable. La segunda, desarrollar una aplicación móvil para poder disponer en cualquier dispositivo móvil de la información de la plataforma descargándose una App. Esta segunda solución es más complicada ya que se necesitan conocimientos de programación de aplicaciones, pero se obtendrá una solución más personalizada.

Para la mejora de la monitorización de los datos ya almacenados en InfluxDB se proponen dos ideas. La primera es la creación de un nuevo microservicio que permita insertar nuevos datos en la base de datos a través de un archivo .csv generado por otra base de datos. Esta funcionalidad es muy importante ya que permite importar y exportar entre servidores las bases de datos. La segunda es la incorporación al servidor de

mecanismos de Machine Learning para realizar predicciones y formar patrones de conducta de los residentes de la vivienda de forma que se puedan dar consejos a los usuarios para mejorar la eficiencia energética. Para este cometido se recomienda el uso de herramientas de desarrollo de inteligencia artificial como son Tensor Flow, Pytorch o Keras.

Con esto se concluiría las líneas futuras relacionadas con el desarrollo de la plataforma IoT del proyecto. Sin embargo, existen ciertos aspectos que también se pueden mejorar a nivel de gestión de proyecto.

Entre ellos se encuentra la realización de un análisis del impacto ambiental más detallado. Para ello se recomienda utilizar como elemento unidad la placa de sensores ambientales ya que probablemente sea la que mayor impacto genere debido a su batería. Si además se realiza a una placa que no contenga batería, se podría conocer con gran detalle cual es el impacto ambiental que generan los componentes del proyecto. Hoy en día es esencial que cualquier proyecto cuente con este estudio, sin embargo, debido a la falta de tiempo no ha sido posible realizarlo.

Como se puede observar se han detallado bastantes medidas, sobre todo relacionadas con la elaboración de nuevas placas de sensores. Gracias al desarrollo de las 3 placas en el proyecto, el diseño de nuevas placas será más sencillo al tener varios ejemplos.

En cuanto a las aplicaciones futuras de este proyecto, el campo del IoT está en pleno auge surgiendo cada año nuevas aplicaciones y tecnologías que introducen cada vez más este concepto en el consumidor. Por lo tanto, el desarrollo de proyectos como el presente permiten aumentar la técnica y que se pueda controlar y mejorar la eficiencia energética en hogares, pero también en industrias, hospitales, almacenes e invernaderos.

10. PLANIFICACIÓN TEMPORAL Y PRESUPUESTO

10.1 PLANIFICACIÓN TEMPORAL

Para el desarrollo de cualquier proyecto se tiene que establecer una planificación temporal. Una mala planificación temporal supone retrasos en las tareas y en proyectos de mayor calibre esto significa mayores costes e incluso la claudicación del proyecto.

Para la planificación de este proyecto se ha hecho uso de dos herramientas. Por un lado, se dispone de un diagrama de Gantt que es una representación gráfica de las diferentes fases que se suceden en el proyecto, las cuales se encuentran en la Tabla 17. Este diagrama que se puede observar dividido en la Ilustración 130 y en la Ilustración 131 permite saber en todo momento de manera visual que tareas se deberían estar realizando, cuales deben estar concluidas y cuales se han de comenzar.

Por otro lado, se ha añadido una Estructura de Descomposición del Proyecto (EDP), en la que se desglosa en un esquema de tipo árbol que se observa en la Ilustración 132 todas las tareas independientes que se deben realizar para completar el proyecto.

El proyecto se ha llevado a cabo en el marco del programa de intercambio Erasmus+, desarrollándose en el Laboratorio de Neuronica del Departamento de Electrónica y Telecomunicaciones (DET) del Politecnico di Torino en Turín, Italia. La carga de trabajo corresponde a 30 créditos ECTS, lo que corresponde al trabajo del segundo periodo del último año del Máster. Es de interés remarcar que el segundo periodo en Turín corresponde al periodo comprendido entre el 2 de marzo de 2019 y 26 de junio de 2019

No obstante, es necesario realizar tareas previas y posteriores a las fechas indicadas. Las tareas previas corresponden a la búsqueda de tema del proyecto y búsqueda de tutores. Esta tarea no se incluirá en la planificación temporal ya que se llevó a cabo en torno a diciembre y enero del año académico en cuestión.

Se considera que el proyecto da comienzo el 14 de marzo de 2019 después de sucesivas reuniones para concretar el tema de este y enseñar las instalaciones. El 25 de junio de 2019 se dejan las instalaciones en Turín para volver a Madrid. A partir de ese momento solo se puede trabajar en la documentación y corregir errores de software. El proyecto concluye el 9 de septiembre de 2019. Por lo tanto, el proyecto se lleva a cabo durante 6 meses. Más concretamente son 179 días, que se quedan en 127 si solo se contabilizan los días laborales. Si se aproximan las horas trabajadas en el proyecto a 6 horas diarias, se obtiene un total de 762 horas empleadas. Esto equivale a 25,4 horas/ 1 crédito ECTS.

	Fase	Fecha de comienzo	Fecha de finalización	Duración (días)
1	PROYECTO TOTALMENTE COMPLETADO	14/03/19	09/09/19	179
1.1	DISEÑO Y ESTUDIO DEL PROYECTO			71
1.2	Estudio y viabilidad del proyecto	14/03/19	01/04/19	19
1.3	Familiarización con CCS	18/03/19	05/04/19	21
1.4	Aprendizaje CC1310 SDK	25/03/19	14/04/19	21
1.5	Aprendizaje ESP 32	15/04/19	28/04/19	14
1.6	Aprendizaje de MicroPython	29/04/19	03/05/19	5
1.7	Estudio de módulos necesarios de Python y MicroPython	04/05/19	06/05/19	3
1.8	Aprendizaje de InfluxDB	17/04/19	23/04/19	7
1.9	Aprendizaje de Grafana	24/04/19	30/04/19	7
1.10	Aprendizaje de OrCAD	31/05/19	13/06/19	14
1.10	Aprendizaje banco de trabajo para soldadura SMD	22/07/19	23/07/19	3
2	DESARROLLO DEL PROYECTO			129
2.1	Pruebas y primeros códigos con CC1310 SDK	18/03/19	05/04/19	21
2.2	Desarrollo de códigos de Sensor Data Subs. Adaptor	25/03/19	16/04/19	23
2.3	Instalación y configuración de InfluxDB	17/04/19	23/04/19	7
2.4	Instalación y configuración de Grafana	24/04/19	27/04/19	4
2.5	Instalación y desarrollo de firmware del ESP 32	29/04/19	12/05/19	14
2.6	Desarrollo de códigos de Sensor Catalog DB	13/05/19	31/05/19	19
2.7	Desarrollo de códigos de Sensor Config. Subs. Adaptor	30/05/19	14/06/19	16
2.8	Desarrollo del firmware básico de las placas	15/04/19	05/05/19	21
2.9	Desarrollo del circuito de la placa de sensores ambientales	06/05/19	13/05/19	8
2.10	Desarrollo del circuito del hub central	13/05/19	17/05/19	5
2.11	Desarrollo del circuito de la placa de consumo de potencia	17/05/19	24/05/19	8
2.12	Creación del BOM	24/05/19	30/05/19	7
2.13	Elaboración de los circuitos con OrCAD	31/05/19	20/06/19	21
2.14	Envío de los circuitos a empresa fabricante de PCBs	21/06/19	12/07/19	22
2.15	Compra de los componentes	21/06/19	22/07/19	30
2.16	Desarrollo de GUI Sensor Config.	21/06/19	27/06/19	7
2.17	Desarrollo de GUI DB Config.	26/06/19	30/06/19	5
2.18	Desarrollo de GUI Query Data	29/06/19	5/07/19	7
2.19	Desarrollo de Telegram Bot Adaptor	6/07/19	09/07/19	4
2.20	Desarrollo de GUI Telegram Config.	09/07/19	09/07/19	1
2.21	Desarrollo de Sensor Status Subs. Control strategies	10/07/19	16/07/19	7
2.22	Desarrollo de Batery Alarm Subs. Control Strategies	15/07/19	21/07/19	7

2.23	Montar las PCBs con los componentes	22/07/19	24/07/19	3
2.24	Instalar firmwarea las placas de sensores y hub central	24/07/19	25/07/19	1
3	EXPERIMENTOS			70
3.1	Envío de mensaje de las placas SDK de sensores a hub central SDK	1/04/19	07/04/19	7
3.2	Prueba de apertura, cierre y reconexión de la red de sensores	08/04/19	12/04/19	5
3.3	Envío de mensajes entre el hub central SDK y servidor	12/04/19	19/04/19	8
3.4	Almacenado de datos en InfluxDB y en el catálogo	20/04/19	26/04/19	7
3.5	Visualización correcta de los datos	25/04/19	26/04/19	2
3.6	Aviso por Telegram ante cambios en la plataforma	09/07/19	19/07/19	11
3.7	Comprobar la correcta elaboración de las PCBs	22/07/19	24/07/19	3
3.8	Comprobación de cortocircuito tras montar las PCBs	22/07/19	24/07/19	3
3.9	Primera prueba de placas con código de ejemplo	24/07/19	24/07/19	1
3.10	Comprobar correcto funcionamiento firmware placa de sensores	29/07/19	02/08/19	5
3.11	Comprobar correcto funcionamiento firmware de hub central	05/08/19	09/08/19	5
3.12	Comprobar correcto funcionamiento firmware placa de consumo de potencia	12/09/19	16/09/19	5
3.13	Comprobar correcto funcionamiento de toda la plataforma	19/09/19	2/09/19	15
	Corrección de errores	26/08/19	30/08/19	5
4	DOCUMENTACIÓN DEL PROYECTO			54
4.1	Escritura del proyecto	25/06/19	09/09/19	54

Tabla 17 – Project stages

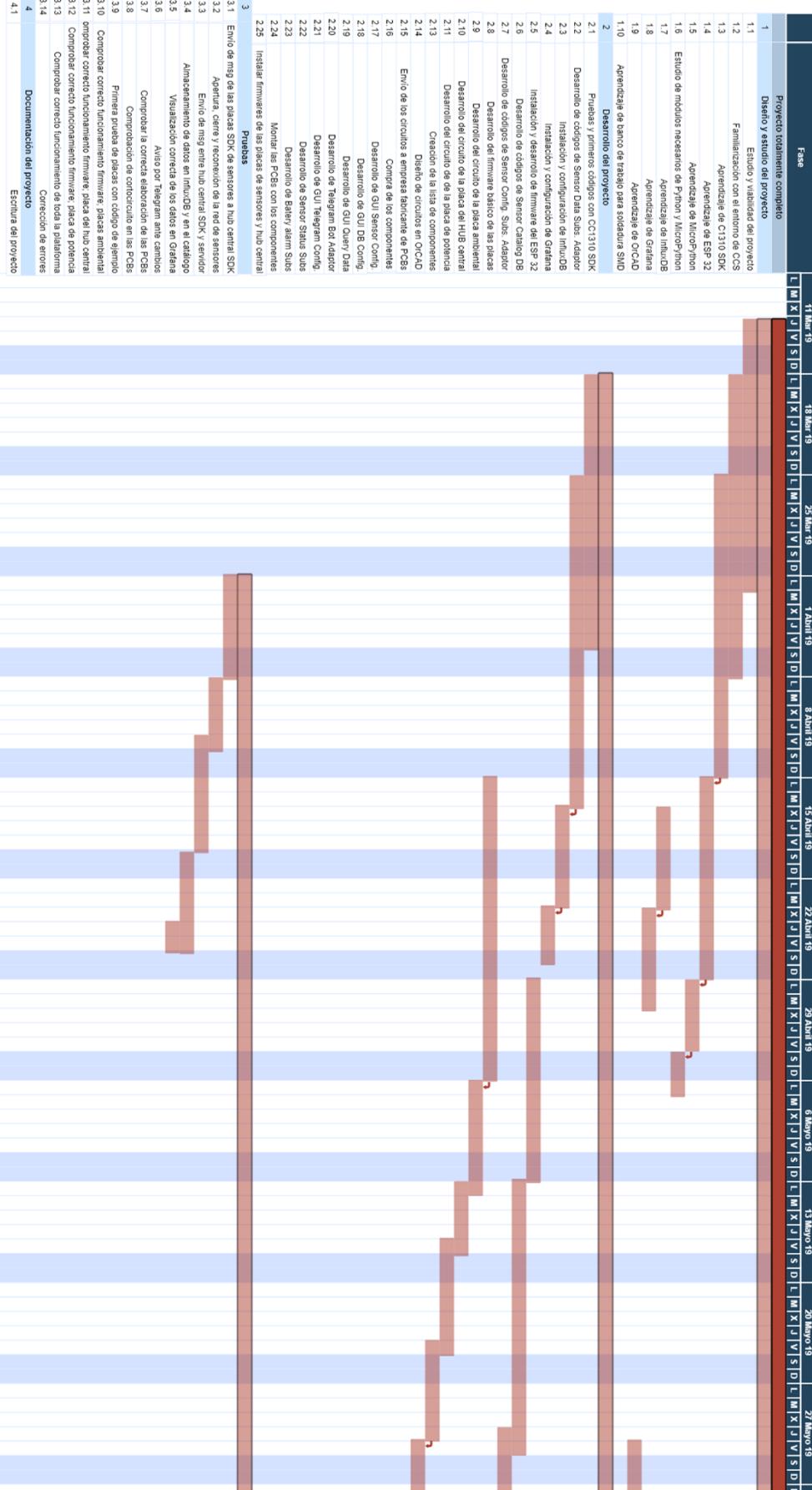


Ilustración 130 - Gantt diagram part 1

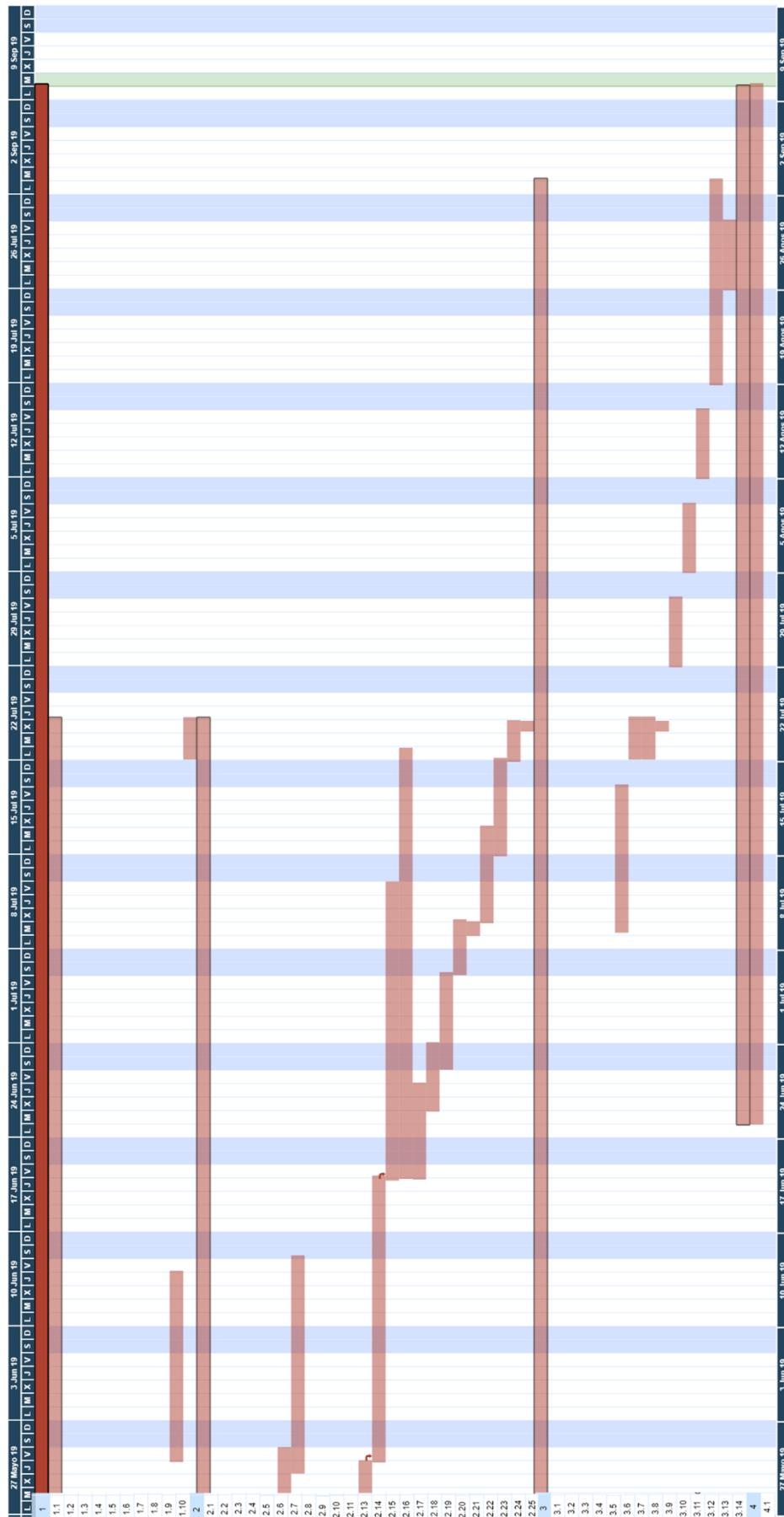


Ilustración 131 - Gantt diagram. Part 2

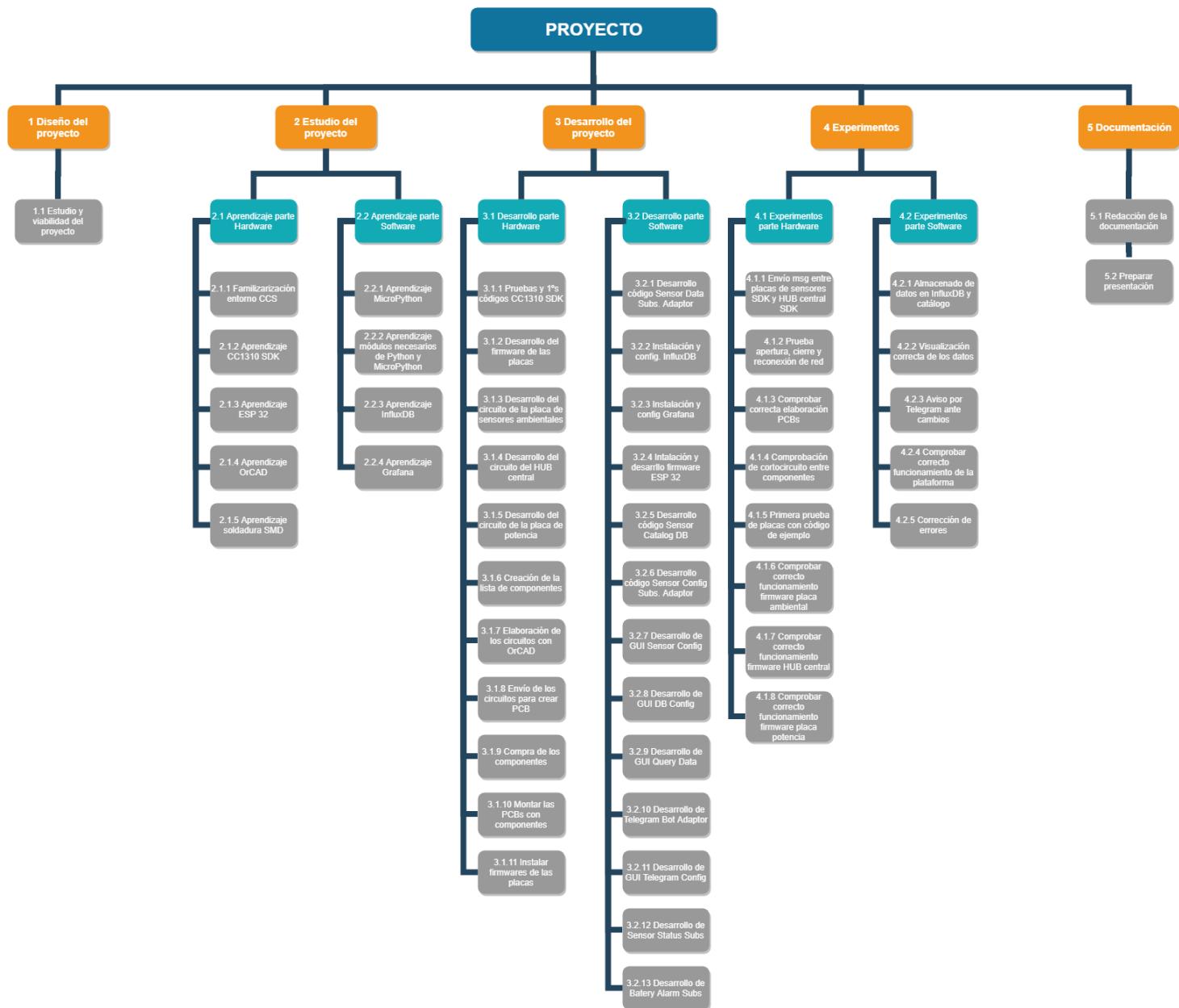


Ilustración 132 - EDP

10.2 PRESUPUESTO

En el presupuesto del proyecto se engloban 3 gastos diferentes. Por un lado, se tiene el gasto en componentes y equipos necesarios para desarrollar el proyecto. Por otro lado, el coste del software para crear los firmwares de las placas y del servidor. Por último, el salario estimado de la persona que desarrollar el proyecto.

En cuanto al hardware necesario, los equipos de mayor coste no han sido comprados en exclusiva para este proyecto, el laboratorio donde se desarrolla cuenta con ellos y son utilizados por varios estudiantes. Por lo tanto, se tiene en cuenta la depreciación de los equipos. Para simplificar, se va a suponer una depreciación lineal, en la que el coste del equipo se divide entre su vida útil estimada. Además, se debe tener en cuenta la duración del proyecto, que se va a considerar 6 meses por simplificar. Esto permite conocer la amortización de un equipo con la siguiente ecuación:

$$\text{Amortización (\%)} = \frac{0.5 \text{ (años)}}{\text{Vida útil (años)}} \times 100$$

Ecuación 13 - linear amortización project equation

En la Tabla 18 se muestran los componentes utilizados junto a su coste y amortización. A continuación, se va a detallar cada componente.

En cuanto a las placas de desarrollo CC1310 de Texas Instruments, son necesarias dos placas para poder realizar pruebas de conexión y envío de mensajes entre ellas. El precio de una placa es 26.4 €, por lo que el precio de 2 más los gastos de envío e impuestos el coste asciende a un valor aproximado de 80€. En el caso de placas de desarrollo, la vida útil depende de la cantidad de uso. Debido a que en un laboratorio se las utiliza más de la media, se va a considerar 5 años.

Para los componentes se va a considerar el coste de todos ellos juntos en vez de separarlos en cada placa. En la compra de componentes se sobredimensionó el pedido ya que al ser componentes SMD era muy sencillo que se perdieran o se dañaran al soldarlos. Además, para el proyecto solo se ensambló una PCB para cada placa, pero en el caso real se necesitaría más de una. Con esto, el precio total asciende a 346.62 €. La vida útil de estos componentes va ligada a la vida útil de 10 años. Sin embargo, como los componentes se compraron para este proyecto, se va a considerar que se amortizan completamente.

En cuanto a las PCBs, el fabricante tiene un límite mínimo de 5 PCBs por cada placa. El precio de 5 PCBs es muy barato, pero a este se le tiene que sumar el coste de envío, impuestos y stencil para extender la pasta térmica. Esto hace que ascienda a 98.22 €. En cuanto a la vida útil, se va a considerar la misma suposición que en los componentes

La placa de desarrollo ESP 32 se puede encontrar en Amazon por menos de 10€, aunque debido a los gastos de envío se aproxima a 10€. En cuanto a su vida útil se considera igual que las placas de desarrollo SDK de 5 años.

El precio de una Raspberry Pi 3 se encuentra actualmente en Amazon a 39€ con gastos de envío. Sin embargo, a esto se tiene que añadir el coste de la fuente de alimentación, tarjeta SD y posible carcasa. Comprando uno de los Kits que aparecen en Amazon, el precio asciende a 62€. En cuanto a la vida útil se va a considerar 10 años al igual que las placas de sensores.

Para el ordenador portátil se va a considerar un ordenador de gama baja-media. Su uso principal en el proyecto es la programación de firmwares y servidor, ofimática. Por lo tanto, se le va a asignar un precio de 400€, con una vida útil de 7 años.

Para los equipos de soldadura y montaje de PCBs, se encuentran los equipos del capítulo 4, apartado 4.3.1.6. Más concretamente los equipos: estación de soldadura de 400€, eC-Stencil-mate de 2000€, eC-Placer 2600 €, Mantis Compact microscope de 1640€, eC-reflow-oven de 3800€, que asciende a un total de 10440€. Estos componentes disponen de una vida útil más elevada que ronda los 30 años.

En los equipos de análisis y medición se encuentra: la fuente de alimentación Rigol DP832A de 950 €, el osciloscopio Tektronix MD03104 de 18000€ y 2 multímetros UNI-T UT61E de 150€ los dos. El coste asciende a 19100€. Que al igual que los equipos de soldadura se consideran 30 años de uso.

En otros componentes se engloban cables de conexión, placas de pruebas, material de soldadura y cualquier componente desecharable o consumible. La vida útil de estos componentes son el propio proyecto y se le va a atribuir un coste de 20 € en total

Equipo	Precio (€)	% Amortización	Total (€)
<i>Placas CC1310 SDK</i>	80	10	8
Componentes	346.62	100	346.62
PCBs	98.22	100	98.22
ESP 32	10	10	1
Raspberry Pi 3	62	5	3.1
Ordenador	400	7	28.57
<i>Equipos de soldadura y montaje de PCBs</i>	10440	1.7	174
<i>Equipos para el análisis y medición de circuitos</i>	19100	1.7	318.3
Otros	20	100	20
TOTAL			997.81

Tabla 18 - Hardware budget

En cuanto a los programas y softwares utilizados, la mayoría son gratuitos. Solo existe un programa que no es gratuito, aunque se contaba con la licencia de estudiante, por lo que no se incurre como gasto. Para conocer el precio de este programa es necesario contactar con el proveedor. También están el coste del sistema operativo Windows 10 y de Microsoft Office para la creación del documento. Pero de nuevo, al utilizar la licencia de estudiante que la universidad habilita, su coste es nulo. En caso de no poseer dicha licencia, la licencia de Windows 10 son 145€ y de Microsoft Office 100€ al año o 150€ la licencia completa. Por lo tanto, en total el precio asciende a 245€.

Por último, queda contabilizar el salario que se le atribuye a la persona por desarrollar el proyecto. Para conocer el salario se obtienen los datos del estudio realizado por la consultora PageGroup sobre remuneración en el mercado laboral de 2019 para ingenieros y técnicos. De este estudio se puede concluir que, para el área de IT y digital el puesto con menor remuneración, es decir, de programador web ronda entre los 20000 € y 26000€ anuales. Escogiendo el valor más bajo y quitándole impuestos se queda en 16000€ anuales. Como el proyecto dura medio año, el coste de salarios es de 8000€. (Page Personnel, 2019)

Por lo tanto, sumando los costes de equipos y de salarios el coste total del proyecto asciende a 8575.17€ tal y como se observa en la Tabla 19.

Concepto	Total (€)
Coste del equipo	997.81
Salarios	8000
Total	8997.81

Tabla 19 - Total project budget

En la Ilustración 133 se puede observar la evolución de los gastos a lo largo de la duración del proyecto. En cuanto al coste de equipos, el gasto inicial corresponde a los elementos que disponía el laboratorio desde el principio. El siguiente gasto se realiza en el mes de junio de 2019 en el que se realiza la compra de los componentes y PCBs.

Sin embargo, el coste de equipos en comparación al de los salarios es casi despreciable a lo largo del proyecto. Esto quiere decir que en proyectos de este tipo en el que el mayor gasto de equipos es inicial, es conveniente desarrollarlo en el menor tiempo posible.

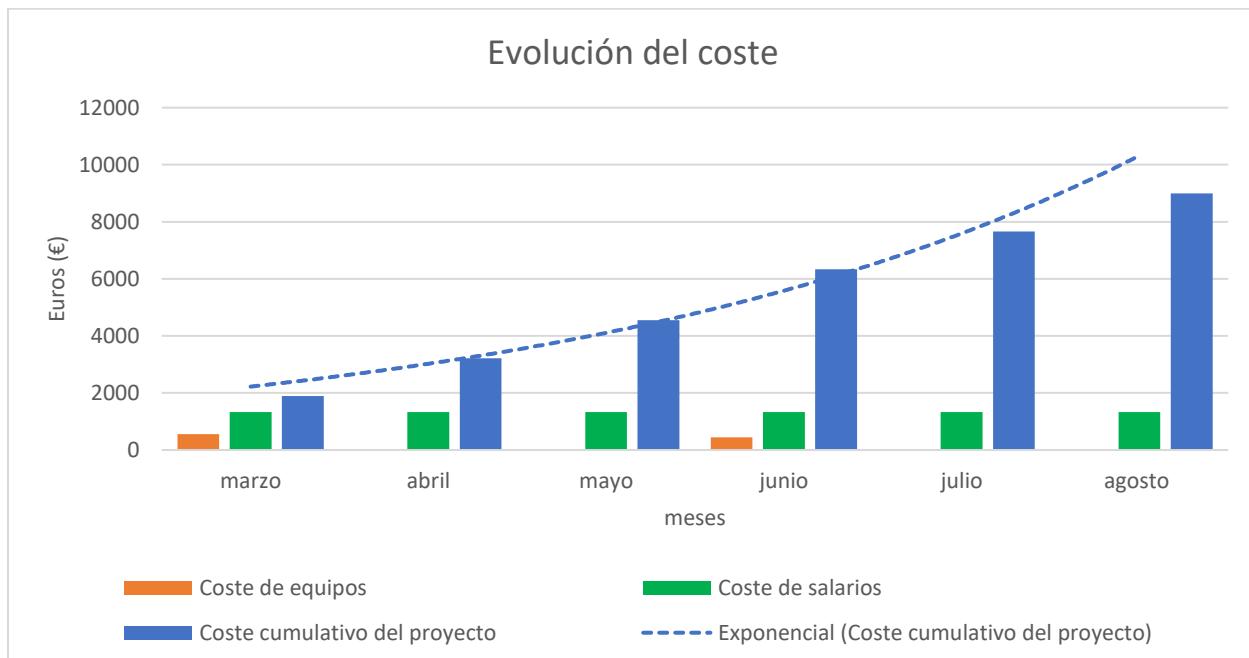


Ilustración 133 - Budget evolution

11. BIBLIOGRAFÍA

- 4refr0nt, & nulrik. (28 de 12 de 2014). GitHub: *ESPlorer*. Obtenido de GitHub: <https://github.com/4refr0nt/ESPlorer>
- Actitud Ecológica. (10 de 2016). *Temperatura de confort: ¿cuál es la temperatura ideal para una casa?* Obtenido de Actitud Ecológica: <https://actitudecologica.com/temperatura-de-confort-cual-es-temperatura-ideal-casa/>
- Actitud Ecológica. (6 de 2017). *Temperatura seca, temperatura húmeda y temperatura efectiva.* Obtenido de Actitud Ecológica : <https://actitudecologica.com/temperatura-seca-temperatura-humeda-y-temperatura-efectiva/>
- Altair SmartWorks. (7 de 2019). Altair SmartWorks. Obtenido de <https://www.altairsmartworks.com/>
- Arduino. (7 de 2019). Arduino. Obtenido de <https://www.arduino.cc/>
- Area Tecnologica. (7 de 2019). Area Tecnologica. Obtenido de <https://www.areatecnologia.com/electricidad/potencia-electrica.html>
- Ashton, K. (2009). That 'Internet of Things' Thing. *RFID Journal*.
- Bluetooth, & Woolley, M. (2019). *Bluetooth 5: Go Faster, Go Further.*
- Bocchio, F. (2014). *Estudio Comparativo de Plataformas Cloud Computing para Arquitecturas SOA*. ResearchGate.
- Bosch. (7 de 2017). *BME680 – Datasheet*. Obtenido de Bosch: <https://cdn-shop.adafruit.com/product-files/3660/BME680.pdf>
- Boxbyte. (2012). El origen de: El Cómputo en la Nube. *FayerWayer*.
- Boyle, A. (28 de 6 de 2019). Geek wire. Obtenido de <https://www.geekwire.com/2019/spacex-reports-milestone-starlink-satellite-links-sparks-debate/>
- CoAP technology. (6 de 2019). *CoAP - Constrained Application Protocol*. Obtenido de <https://coap.technology/>
- Copley, J. (2015). *Diversifying the IoT with Sub-1 GHz technology*. Texas Instruments. Obtenido de <http://www.ti.com/lit/wp/swry017/swry017.pdf>
- David L., M., James J., M., Jack L., B., & William T., K. (2010). *Network Time Protocol Version 4: Protocol and Algorithms Specification*. RFC 2010.
- DFRobot Community. (2019). Documentation uPyCraft. Obtenido de DFRobot: <http://docs.dfrobot.com/upycraft/>
- DiCola, T. (21 de 1 de 2019). *MicroPython Basics: Load Files & Run Code*. Obtenido de Adafruit: <https://cdn-learn.adafruit.com/downloads/pdf/micropython-basics-load-files-and-run-code.pdf>

- dpgeorge. (5 de 6 de 2018). *GitHub:micropython/ports/esp8266/modules/ntptime.py*. Obtenido de GitHub: <https://github.com/micropython/micropython/blob/master/ports/esp8266/modules/ntptime.py>
- Elahi, A., & Gshwender, A. (2009). *Zigbee Wireless sensor and control network*.
- Espressif. (7 de 2019). *Espressif*. Obtenido de <https://www.espressif.com/en/products/hardware>
- Espressif Systems. (2019). *Espressif Systems - ESP 32*. Obtenido de Hardware ESP 32 overview: <https://www.espressif.com/en/products/hardware/esp32/overview>
- García Cortiñas, L. (12 de 4 de 2018). *Gradiant*. Obtenido de <https://www.gradiant.org/blog/edge-fog-computing-cloud/>
- González, M. (22 de 10 de 2016). *Xataka*. Obtenido de <https://www.xataka.com/servicios/los-responsables-del-ddos-a-dyn-usaron-camaras-ip-y-dvrs-para-tumbar-medio-internet>
- Google Cloud. (7 de 2019). *Google Cloud*. Obtenido de <https://cloud.google.com/docs/>
- Grafana Labs. (2019). *What is Grafana?* Obtenido de Grafana: <https://grafana.com/grafana>
- Greiner, R. (02 de 3 de 2014). *Windows Azure IaaS vs. PaaS vs. SaaS*. Obtenido de <http://robertgreiner.com: http://robertgreiner.com/2014/03/windows-azure-iaas-paas-saas-overview/>
- Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2012). *Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions*. ResearchGate.
- Hellan, S. (5 de 2016). *Texas Instruments Training*. Obtenido de Why Sub-1GHz?: <https://training.ti.com/why-sub-1ghz>
- Hughes, M. (2 de 2017). *All about circuits*. Obtenido de SPI Serial Peripheral Interface: <https://www.allaboutcircuits.com/technical-articles/spi-serial-peripheral-interface/>
- Hymel, S. (7 de 2019). *How to load MicroPython on a Microcontroller Board*. Obtenido de Sparkfun: <https://learn.sparkfun.com/tutorials/how-to-load-micropython-on-a-microcontroller-board/esp32-thing>
- Hymel, S. (7 de 2019). *MicroPython Programming Tutorial: Getting Started with the ESP32 Thing*. Obtenido de Sparkfun: <https://learn.sparkfun.com/tutorials/micropython-programming-tutorial-getting-started-with-the-esp32-thing/all>
- IBM Cloud. (7 de 2019). *IBM Cloud*. Obtenido de <https://www.ibm.com/cloud>
- IEC, I. E. (2018). *Internet of Things: Wireless Sensor Networks*.
- IEEE, Fox, G. C., Kamburugamuve, S., & Hartman, R. D. (2012). *Architecture and measured characteristics of a cloud based internet of things*. *2012 International Conference on Collaboration Technologies and Systems (CTS)*. Denver.
- IEEE, Hunkeler, U., Truong, H. L., & Stanford-Clark, A. (2008). *MQTT-S — A publish/subscribe protocol for Wireless Sensor Networks*.
- IERC, C. o. (2010). *Vision and Challenges for Realising the Internet of Things*. CERP-IoT.

- InfluxData. (2019). *InfluxDB documentation*. Obtenido de <https://docs.influxdata.com/influxdb/v1.7/>
- InfluxData. (2019). *TICK stack: InfluxDB*. Obtenido de InfluxDB: <https://www.influxdata.com/time-series-platform/>
- International Organization for Standardization. (2014). *Information technology -- Advanced Message Queuing Protocol (AMQP) v1.0 specification*.
- Jacobson, R. (4 de 2013). *IBM*. Obtenido de <https://www.ibm.com/blogs/insights-on-business/consumer-products/2-5-quintillion-bytes-of-data-created-every-day-how-does-cpg-retail-manage-it/>
- Keim, R. (12 de 2015). *All about circuits*. Obtenido de Introduction to the I2C bus: <https://www.allaboutcircuits.com/technical-articles/introduction-to-the-i2c-bus/>
- Keim, R. (12 de 2015). *All about circuits*. Obtenido de The I2C Bus: Firmware Implementation Details: <https://www.allaboutcircuits.com/technical-articles/the-i2c-bus-firmware-implementation-details/>
- Knud, L. L. (8 de 2018). *IoT ANALYTICS*. Obtenido de <https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/>
- La basura electrónica en 4 gráficos: cómo el mundo desperdicia US\$62.500 millones cada año. (29 de 1 de 2019). BBC News.
- Liao, G., Saleh, A., & Haque, A. (2010). *Characteristics of WiFi*. ENSC 427 Spring 2010.
- Long, S. (8 de 2017). *Raspbian Stretch has arrived for Raspberry Pi*. Obtenido de Raspberry Pi Foundation: <https://www.raspberrypi.org/blog/raspbian-stretch/>
- Long, S. (6 de 2019). *Buster – the new version of Raspbian*. Obtenido de Raspberry Pi Foundations: <https://www.raspberrypi.org/blog/buster-the-new-version-of-raspbian/>
- Loy, M. (2005). *ISM-Band and Short Range Device Regulatory Compliance Overview*. Semantic scholar.
- Mackenzie, J. (2 de 1 de 2017). *Headless Raspberry Pi Setup*. Obtenido de Hackernoon: <https://hackernoon.com/raspberry-pi-headless-install-462ccabd75d0>
- Mathworks. (7 de 2019). *Thingspeak*, Mathworks. Obtenido de <https://es.mathworks.com/products/thingspeak.html>
- Microsoft Azure. (7 de 2019). *Microsoft Azure*. Obtenido de <https://azure.microsoft.com/es-es/overview/what-is-cloud-computing/>
- Myrra. (2018). *Catalogue Myrra*. Obtenido de <https://www.tme.eu/Document/cb1d7fdbbe3570188ff28030ca3a4f1cf/Catalogue%20Myrra%202018%20compress%C3%A9.pdf>
- National Institute of Standards and Technology. (2011). *The NIST Definition of Cloud*.
- NFC Forum. (6 de 2019). *NFC Forum*. Obtenido de <https://nfc-forum.org/nfc-and-the-internet-of-things/>
- Olsson, J., & Texas Instruments. (2014). *Texas Instruments*.
- OrCAD. (2019). *Products: OrCAD Cadence PCB Solutions*. Obtenido de OrCAD Cadence PCB Solutions: <https://www.orcad.com/products/orcad-overview/orcad-capture-pspice-simulation-pcb-layout-design-routing-software>

- O'Reilly, M. (2010). *TCP/IP Network Administration, 3rd Edition.*
- ouki-wang, & mengbishi. (7 de 1 de 2018). *uPyCraft GitHub.* Obtenido de GitHub: https://github.com/DFRobot/uPyCraft_src
- pfalcon, dxxb, & dpgeorge. (30 de 3 de 2014). *GitHub: umqtt.robust.* Obtenido de GitHub: <https://github.com/micropython/micropython-lib/tree/master/umqtt.robust>
- Prometec. (2019). *Prometec.* Obtenido de Instalando ESP 32: <https://www.prometec.net/installando-esp32/>
- Python Software Foundation. (2019). *Wiki Python.* Obtenido de <https://wiki.python.org/moin/BeginnersGuide/Overview>
- Raspberry Pi Foundation. (2019). *Raspberry Pi 3 Model B.* Obtenido de <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>
- Raspberry Pi Foundation. (2019). *Raspbian: Raspberry Pi org.* Obtenido de Raspberry Pi org: <https://www.raspberrypi.org/downloads/raspbian/>
- Raspberry Pi ORG. (7 de 2019). *Raspberry Pi.* Obtenido de <https://www.raspberrypi.org/>
- Rouse, M. (2005). *Tech Target - Nyquist theorem.* Obtenido de <https://whatis.techtarget.com/definition/Nyquist-Theorem>
- Rouse, M. (12 de 2017). *REST (REpresentational State Transfer).* Recuperado el 6 de 2019, de Search Microservices: <https://searchmicroservices.techtarget.com/definition/REST-representational-state-transfer>
- Rouse, M. (2 de 2018). *IoT Agenda.* Obtenido de <https://internetofthingsagenda.techtarget.com/definition/MQTT-MQ-Telemetry-Transport>
- Rouse, M. (6 de 2019). *IoT Agenda.* Obtenido de RFID (radio frequency identification): <https://internetofthingsagenda.techtarget.com/definition/RFID-radio-frequency-identification>
- Salarian, H., Chin, K.-W., & Naghd, F. (2012). *Coordination in wireless sensor–actuator networks: A survey.* Wollongong: ScienceDirect.
- Schmidt, E. (8 de 2006). *Search Engine Round Table.* Obtenido de <https://www.seroundtable.com/archives/004343.html>
- Semtech. (6 de 2019). *Semtech.* Obtenido de <https://www.semtech.com/lora/what-is-lora>
- Sigfox. (6 de 2019). *Sigfox Technology Overview.* Obtenido de Sigfox Technology Overview: <https://www.sigfox.com/en/sigfox-iot-technology-overview>
- Silicon Laboratories Inc. (2010). *Key Priorities for Sub-GHz Wireless Deployment.* Austin. Obtenido de <https://www.silabs.com/documents/public/white-papers/Key-Priorities-for-Sub-GHz-Wireless-Deployments.pdf>
- Talema Group. (2 de 2016). *Digikey.* Obtenido de Understanding and using current transformers: https://www.digikey.com/en/ptm/t/talema/understanding-and-using-current-transformers?pn_sku=1295-1099-ND&part_id=3881399
- Talema Group. (7 de 2018). *Current Transformer Quick Reference Guide.* Obtenido de Talema Group: <https://talema.com/uploads/documents/product-datasheets/AC%20Series.pdf>

- tdicola, dhalbert, & dezxpy. (31 de 7 de 2016). Aampy. Obtenido de GitHub: <https://github.com/pycampers/ampy>
- Texas Instruments. (2019). Out-of-box star-network solution: TI 15.4-Stack. Obtenido de Texas Instruments: <http://www.ti.com/lit/ml/swat015/swat015.pdf>
- Texas Instruments. (2019). Texas Instrument CCStudio. Obtenido de <http://www.ti.com/tool/CCSTUDIO>
- Texas Instruments. (2019). Texas Instruments. Obtenido de CC1350 SimpleLink Ultra-Low Power Dual Band Wireless Microcontroller: <http://www.ti.com/product/CC1350>
- Texas Instruments. (7 de 2019). Texas Instruments Resource Explorer - RTOS concepts. Obtenido de http://dev.ti.com/tirex/explore/node?node=AIHYsnK2VMe6vGRWXhb3uQ_eCfARaV_LATEST
- Texas Instruments. (7 de 2019). Texas Instruments Resource Explorer - TI-RTOS basics. Obtenido de http://dev.ti.com/tirex/explore/node?node=AP0xu9WGzmfM9bchOnXboA_eCfARaV_LATEST
- Texas Instruments. (7 de 2019). Texas Instruments SimpleLink Solutions. Obtenido de <http://www.ti.com/wireless-connectivity/simplelink-solutions/overview/overview.html>
- Uckelmann, D., Harrison, M., & Michahelles, F. (2011). *Architecting the Internet of Things*. Berlin: Springer.
- Vega, A., Santamaria, F., & Rivas, E. (2015). *Modeling for home electric energy management: A review*. Bogotá: Science Direct.
- Vermesan, O., Friess, P., Guillemin, P., & Gusmeroli, S. (2009). *Internet of Things Strategic Research Roadmap*. ResearchGate.
- Vijayasankar, K., & Movva, P. (2016). TI 15.4-Stack Frequency Hopping Mode FCC Compliance. Texas Instruments. Obtenido de <http://www.ti.com/lit/an/swra529a/swra529a.pdf>
- Yegulalp, S. (6 de 2014). Micro Python's tiny circuits: Python variant targets microcontrollers. Obtenido de InfoWorld: <https://www.infoworld.com/article/2608012/micro-python-s-tiny-circuits--python-variant-targets-microcontrollers.html>
- Youkhana, D. (2016). Why use Sub-1 GHz in your IoT application. Obtenido de E2E TI Forum: <http://e2e.ti.com/blogs/b/process/archive/2016/07/27/why-use-sub-1-ghz-in-your-iot-application>

ANEXO I: COMPONENTES UTILIZADOS

A1. PLACA DE SENsoRES AMBIENTALES

Circuit references	short desc.	Qt	Manuf. Code	Manuf.	Description
B1,B2	Button rs	2	4-1437565-1	TE	Black Button Tactile Switch, Single Pole Single Throw (SPST) 50 mA @ 12 V ac 1.4mm
C1,C9	3.6 pF rs	2	GRM1553C1H3R6 BA01D	Murata	Murata 0402 (1005M) 3.6pF MLCC 50V dc ±0.1pF SMD GRM1553C1H3R6BA01D
C2,C10	100 pF rs	2	885012206028	Wurth Elektronik	Wurth Elektronik 0603 (1608M) 100pF MLCC 16V dc ±10% SMD 885012206028
C3	1.8 pF rs	1	GRM1555C1H1R8 CA01D	Murata	Murata 0402 (1005M) 1.8pF MLCC 50V dc ±0.25pF SMD GRM1555C1H1R8CA01D
C4	2.7 pF rs	1	C0603C279C1GAC TU	KEMET	KEMET 0603 (1608M) 2.7pF MLCC 100V dc ±0.25pF SMD C0603C279C1GACTU
C5	6.2 pF rs	1	GRM1555C1H6R2 DA01D	Murata	Murata 0402 (1005M) 6.2pF MLCC 50V dc ±0.5pF SMD GRM1555C1H6R2DA01D
C6	3 pF rs	1	06035J3R0BBSTR	AVX	AVX 0603 (1608M) 3pF MLCC 50V dc ±0.1pF SMD 06035J3R0BBSTR
C7,C8	DNM	2	DNM	DNM	DNM
C11,C12	12 pF rs	2	GRM1882C2A120J A01D	Murata	Murata 0603 (1608M) 12pF MLCC 100V dc ±5% SMD GRM1882C2A120JA01D
C13	1 uF rs	1	CGA3E1X5R1C105 K080AC	TDK	MLCC TDK CGA 1μF, ±10%, 16V cc, SMD
C14,C15,C16,C18,C20,C21,C22,C23,C27,C28,C29,C30,C31,C32,C33,C34,C35	100 nF rs	17	8,85012E+11	Wurth Elektronik	Wurth Elektronik 0603 (1608M) 100nF MLCC 10V dc ±10% SMD 885012206020
C17,C19	22 uF rs	2	C1608X5R1A226M 080AC	TDK	TDK 0603 (1608M) 22μF MLCC 10V dc ±20% SMD C1608X5R1A226M080AC
C24	330 nF rs	1	885012206074	Wurth Elektronik	Wurth Elektronik 0603 (1608M) 330nF MLCC 25V dc ±10% SMD 885012206074
C25,C26	10 uF rs	2	GRM188B31A106 ME69D	Murata	MLCC Murata GRM 10μF, ±20%, 10V cc, SMD
D1	RLED rs	1	LS Q976	Osram	2.5 V Red LED 1608 (0603) SMD, Osram Opto CHIPLED 0603 LS Q976
D2	GLED rs	1	KP-1608SGC	Kingbright	2.5 V Green LED 1608 (0603) SMD, Kingbright KP-1608SGC
FL1	FERRITE rs	1	BLM18HE152SN1 D	Murata	Murata Ferrite Bead (Chip Ferrite Bead), 1.6 x 0.8 x 0.8mm (0603 (1608M)), 1500Ω impedance at 100 MHz
J1	FTSH 10 pin rs	1	FTSH-105-01-F-DV	Samtec	Samtec FTSH, 1.27mm Pitch, 10 Way, 2 Row, Straight Pin Header, Surface Mount
J2,J4	2 pin	2	5-146278-2	TE	TE Connectivity AMPMODU Mod II, 2.54mm Pitch, 2 Way, 1 Row, Straight Pin Header, Through Hole
J3	8 pin	1	77311-101-08LF	Amphenol FCI	Amphenol FCI BergStik, 2.54mm Pitch, 8 Way, 1 Row, Straight Pin Header, Through Hole
J5	BAT connector rs	1	S2B-PH-K-S(LF)(SN)	JST	JST PH S2B, 2mm Pitch, 2 Way, 1 Row, Right Angle PCB Header, Through Hole
J6	MicroUSB rs	1	105164-0001	Molex	USB Connector, Micro USB Type B, USB 2.0, Receptacle, 5 Ways, Surface Mount, Right Angle
L1,L5	7.5 nH rs	2	744761075A	Wurth Elektronik	Wurth WE-KI Series Type 0603A Shielded Wire-wound SMD Inductor with a Ceramic Core, 7.5 nH Wire-Wound 700mA Idc Q:28

L2,L3	6.8 nH rs	2	744761068A	Wurth Elektronik	Wurth WE-KI Series Type 0603 Wire-wound SMD Inductor with a Ceramic Core, 6.8 nH ±5% Wire-Wound 700mA Idc Q:30
L4	27 nH rs	1	744761127C	Wurth Elektronik	Wurth WE-KI Series Type 0603C Shielded Wire-wound SMD Inductor with a Ceramic Core, 27 nH Wire-Wound 600mA Idc Q:35
L6	6.8 uH rs	1	LQH2HPZ6R8MGR L	Murata	Murata LQH Series Type 2520 Shielded Wire-wound SMD Inductor with a Ferrite Core, 6.8 µH Wire-Wound 860mA Idc
R1,R2	4K7 rs	2	MCR03EZPFX4701	ROHM	4.7kΩ 0603 Thick Film SMD Resistor ±1% 0.1W - MCR03EZPFX4701
R3,R4	100K rs	2	CR0603-FX-1003ELF	Bourns	100kΩ 0603 Thick Film Surface Mount Fixed Resistor ±1% 0.1W - CR0603-FX-1003ELF
R5	220 rs	1	CRCW0603220RFK EAHP	Vishay	220Ω 0603 Thick Film Power Resistor ±1% 0.25W - CRCW0603220RFKEAHP
R6	180 rs	1	CRCW0603180RJN EAIF	Vishay	180Ω 0603 Thick Film SMD Resistor ±5% 0.1W - CRCW0603180RJNEAIF
R7,R8	10K rs	2	CR0603-FX-1002ELF	Bourns	10kΩ 0603 Thick Film Surface Mount Fixed Resistor ±1% 0.1W - CR0603-FX-1002ELF
R9,R10	6K8 rs	2	MCR03EZPFX6801	ROHM	6.8kΩ 0603 Thick Film SMD Resistor ±1% 0.1W - MCR03EZPFX6801
U1	CC1310 rs	1	CC1310F128RGZT	TI	RF Microcontrollers - MCU SimpleLink Sub-1 GHz Ultra-Low Power Wireless Microcontroller 48-VQFN -40 to 85
U2	MAX175 9 rs	1	MAX1759EUB+	Maxim	Maxim MAX1759EUB+, Charge Pump Step Down, Step Up 100mA 1800 kHz, 2.5 → 5.5 V 10-Pin, μMAX
U3	MAX155 5 rs	1	MAX1555EZK+T	Maxim	Maxim MAX1555EZK+T, Lithium-Ion, Lithium-Polymer, Battery Charge Controller, 280mA 5-Pin, SOT-23
U4	MAX170 58 rs	1	MAX17058	Maxim	MAX17058
U6	BME680 rs	1	BME680	Bosch Sensortec	Bosch Sensortec BME680, Temperature & Humidity Sensor
Y1	32K7 Crystal rs	1	Q13FC135000041 1	Epson	Epson 32.768kHz Crystal ±20ppm SMD 2-Pin 3.2 x 1.5 x 0.8mm
Y2	24M Crystal rs	1	X1E000021012211	Epson	Epson 24MHz Crystal ±10ppm SMD 4-Pin 3.2 x 2.5 x 0.6mm

Tabla 20 - Environmental sensor board components

A2. PLACA DE CONSUMO DE POTENCIA ELÉCTRICA

Circuit references	short desc.	Q t	Manuf. Code	Manuf.	Description
B1,B2	Button rs	2	4-1437565-1	TE	Black Button Tactile Switch, Single Pole Single Throw (SPST) 50 mA @ 12 V ac 1.4mm
C1,C10	3.6 pF rs	2	GRM1553C1H3R6BA01 D	Murata	Murata 0402 (1005M) 3.6pF MLCC 50V dc ±0.1pF SMD GRM1553C1H3R6BA01D
C2,C12,C16,C17,C18,C20,C22,C23,C24,C25,C26,C27,C29,C30,C32,C34,C35,C38	100 nF rs	1 8	8,85012E+11	Wurth Elektronik	Wurth Elektronik 0603 (1608M) 100nF MLCC 10V dc ±10% SMD 885012206020
C3,C11	100 pF rs	2	885012206028	Wurth Elektronik	Wurth Elektronik 0603 (1608M) 100pF MLCC 16V dc ±10% SMD 885012206028
C4	1.8 pF rs	1	GRM1555C1H1R8CA01 D	Murata	Murata 0402 (1005M) 1.8pF MLCC 50V dc ±0.25pF SMD GRM1555C1H1R8CA01D
C5	2.7 pF rs	1	C0603C279C1GACTU	KEMET	KEMET 0603 (1608M) 2.7pF MLCC 100V dc ±0.25pF SMD C0603C279C1GACTU
C6	6.2 pF rs	1	GRM1555C1H6R2DA01 D	Murata	Murata 0402 (1005M) 6.2pF MLCC 50V dc ±0.5pF SMD GRM1555C1H6R2DA01D
C7	3 pF rs	1	06035J3R0BBSTR	AVX	AVX 0603 (1608M) 3pF MLCC 50V dc ±0.1pF SMD 06035J3R0BBSTR

Anexo I: Componentes utilizados

	DNM	2	DNM	DNM	DNM
C8,C9	1 uF rs	3	CGA3E1X5R1C105K080 AC	TDK	MLCC TDK CGA 1µF, ±10%, 16V cc, SMD
C13,C36,C37	12 pF rs	2	GRM1882C2A120JA01D	Murata	Murata 0603 (1608M) 12pF MLCC 100V dc ±5% SMD GRM1882C2A120JA01D
C14,C15	22 uF rs	2	C1608X5R1A226M080AC	TDK	TDK 0603 (1608M) 22µF MLCC 10V dc ±20% SMD C1608X5R1A226M080AC
C19,C21	47 nF rs	2	VJ0603Y473KXXAC	Vishay	Vishay 0603 (1608M) 47nF MLCC 25V dc ±10% SMD VJ0603Y473KXXAC
C28,C31	2200 uF rs	1	EEVFK1E222M	Panasonic	Panasonic 2200µF 25V dc Aluminium Electrolytic Capacitor, Surface Mount 16 (Dia.) x 16.5mm +105°C 16mm
C33	RLED rs	1	LS Q976	Osram	2.5 V Red LED 1608 (0603) SMD, Osram Opto CHIPLED 0603 LS Q976
D1	GLED rs	2	KP-1608SGC	Kingbright	2.5 V Green LED 1608 (0603) SMD, Kingbright KP-1608SGC
D2,D4	Full bridge rectifier rs	1	VS-2KBP005	Vishay	Vishay VS-2KBP005, Bridge Rectifier, 2A 50V, 4-Pin D 44
FL1,FL2	FERRITE rs	2	BLM18HE152SN1D	Murata	Murata Ferrite Bead (Chip Ferrite Bead), 1.6 x 0.8 x 0.8mm (0603 (1608M)), 1500Ω impedance at 100 MHz
J1	FTSH 10 pin rs	1	FTSH-105-01-F-DV	Samtec	Samtec FTSH, 1.27mm Pitch, 10 Way, 2 Row, Straight Pin Header, Surface Mount
J2,J5	2 pin	2	5-146278-2	TE	TE Connectivity AMPMODU Mod II, 2.54mm Pitch, 2 Way, 1 Row, Straight Pin Header, Through Hole
J3	4 pin	1	M20-9990446	HARWIN	HARWIN M20, 2.54mm Pitch, 4 Way, 1 Row, Straight Pin Header, Through Hole
J4	AC terminals rs	1	282836-2	TE	TE Connectivity Buchanan, 2 Way 5mm Pitch PCB Terminal Strip
L1,L5	7.5 nH rs	2	744761075A	Wurth Elektronik	Wurth WE-KI Series Type 0603A Shielded Wire-wound SMD Inductor with a Ceramic Core, 7.5 nH Wire-Wound 700mA Idc Q:28
L2	27 nH rs	1	744761127C	Wurth Elektronik	Wurth WE-KI Series Type 0603C Shielded Wire-wound SMD Inductor with a Ceramic Core, 27 nH Wire-Wound 600mA Idc Q:35
L3,L4	6.8 nH rs	2	744761068A	Wurth Elektronik	Wurth WE-KI Series Type 0603 Wire-wound SMD Inductor with a Ceramic Core, 6.8 nH ±5% Wire-Wound 700mA Idc Q:30
L6	6.8 uH rs	1	LQH2HPZ6R8MGRL	Murata	Murata LQH Series Type 2520 Shielded Wire-wound SMD Inductor with a Ferrite Core, 6.8 µH Wire-Wound 860mA Idc
R1,R3	10K rs	2	CR0603-FX-1002ELF	Bourns	10kΩ 0603 Thick Film Surface Mount Fixed Resistor ±1% 0.1W - CR0603-FX-1002ELF
R2,R4	6K8 rs	2	MCR03EZPFX6801	ROHM	6.8kΩ 0603 Thick Film SMD Resistor ±1% 0.1W - MCR03EZPFX6801
R5,R20	180 rs	2	CRCW0603180RJNEAIF	Vishay	180Ω 0603 Thick Film SMD Resistor ±5% 0.1W - CRCW0603180RJNEAIF
R6	220 rs	1	CRCW0603220RFKEAHP	Vishay	220Ω 0603 Thick Film Power Resistor ±1% 0.25W - CRCW0603220RFKEAHP
R7	100K rs	1	CR0603-FX-1003ELF	Bourns	100kΩ 0603 Thick Film Surface Mount Fixed Resistor ±1% 0.1W - CR0603-FX-1003ELF
R8	39K rs	1	CRGH0603J39K	TE	39kΩ 0603 Thick Film Surface Mount Fixed Resistor ±5% 0.2W - CRGH0603J39K
R9,R10,R13,R14,R17,R18	2K2 rs	6	MCR03EZPFX2201	ROHM	2.2kΩ 0603 Thick Film SMD Resistor ±1% 0.1W - MCR03EZPFX2201
R11	4K7 rs	1	MCR03EZPFX4701	ROHM	4.7kΩ 0603 Thick Film SMD Resistor ±1% 0.1W - MCR03EZPFX4701
R12	120K rs	1	CPF0603B120KE	TE	120kΩ 0603 Thin Film Precision Thin Film Surface Mount Fixed Resistor ±0.1% 0.063W - CPF0603B120KE
R15	Current resistor rs	1	CPF1100R00FKE14	Vishay	Vishay CPF Series Axial Metal Film Fixed Resistor 100Ω ±1% 1W ±100ppm/°C
R16	27K rs	1	ERA3AEB273V	Panasonic	27kΩ 0603 Thin Film SMD Resistor ±0.1% 0.1W - ERA3AEB273V

R19,R22	1M rs	2	CRG0603F1M0	TE	1MΩ 0603 Thick Film SMD Resistor ±1% 0.1W - CRG0603F1M0
TR1	Myrra 44091 rs	1	44091	Myrra	6V ac 2 Output Through Hole PCB Transformer, 1.5VA
TR2	Talema AC1060 rs	1	AC-1060	Nuvotem Talema	Nuvotem Talema Circuit Transformer, , 60A Input,
U1	CC1310 rs	1	CC1310F128RGZT	TI	RF Microcontrollers - MCU SimpleLink Sub-1 GHz Ultra-Low Power Wireless Microcontroller 48-VQFN -40 to 85
U2	LMV324 rs	1	LMV324IDR	TI	LMV324IDR Texas Instruments, Op Amp, RRO, 1MHz, 3 V, 5 V, 14-Pin SOIC
U3,U4	LTC1440 rs	2	LTC1440CS8#PBF	Linear Technology	LTC1440CS8#PBF Linear Technology, Comparator, 3 → 9 V 8-Pin SOIC
U5	MCP1703 rs	1	MCP1703-3302E/DB	Microchip	Microchip MCP1703-3302E/DB, LDO Regulator, 250mA, 3.3 V, ±2% 3+Tab-Pin, SOT-223
Y1	32K7 Crystal rs	1	Q13FC1350000411	Epson	Epson 32.768kHz Crystal ±20ppm SMD 2-Pin 3.2 x 1.5 x 0.8mm
Y2	24M Crystal rs	1	X1E000021012211	Epson	Epson 24MHz Crystal ±10ppm SMD 4-Pin 3.2 x 2.5 x 0.6mm

Tabla 21 - Power sensor board components

A1. PLACA DEL HUB CENTRAL

Circuit references	short desc.	Q t	Manuf. Code	Manuf.	Description
B1,B2	Button rs	2	4-1437565-1	TE	Black Button Tactile Switch, Single Pole Single Throw (SPST) 50 mA @ 12 V ac 1.4mm
C1,C10	3.6 pF rs	2	GRM1553C1H3R6BA01 D	Murata	Murata 0402 (1005M) 3.6pF MLCC 50V dc ±0.1pF SMD GRM1553C1H3R6BA01D
"C2,C12,C16,C17,C18,C20,					
C22,C23,C24,C25"	100 nF rs	1 0	8,85012E+11	Wurth Elektronik	Wurth Elektronik 0603 (1608M) 100nF MLCC 10V dc ±10% SMD 885012206020
C3,C11	100 pF rs	2	885012206028	Wurth Elektronik	Wurth Elektronik 0603 (1608M) 100pF MLCC 16V dc ±10% SMD 885012206028
C4	1.8 pF rs	1	GRM1555C1H1R8CA01 D	Murata	Murata 0402 (1005M) 1.8pF MLCC 50V dc ±0.25pF SMD GRM1555C1H1R8CA01D
C5	2.7 pF rs	1	C0603C279C1GACTU	KEMET	KEMET 0603 (1608M) 2.7pF MLCC 100V dc ±0.25pF SMD C0603C279C1GACTU
C6	6.2 pF rs	1	GRM1555C1H6R2DA01 D	Murata	Murata 0402 (1005M) 6.2pF MLCC 50V dc ±0.5pF SMD GRM1555C1H6R2DA01D
C7	3 pF rs	1	06035J3R0BBSTR	AVX	AVX 0603 (1608M) 3pF MLCC 50V dc ±0.1pF SMD 06035J3R0BBSTR
C8,C9	DNM	2	DNM	DNM	DNM
C13,C14	12 pF rs	2	GRM1882C2A120JA01D	Murata	Murata 0603 (1608M) 12pF MLCC 100V dc ±5% SMD GRM1882C2A120JA01D
C15,C26,C27	1 uF rs	3	CGA3E1X5R1C105K080 AC	TDK	MLCC TDK CGA 1μF, ±10%, 16V cc, SMD
C19,C21	22 uF rs	2	C1608X5R1A226M080A C	TDK	TDK 0603 (1608M) 22μF MLCC 10V dc ±20% SMD C1608X5R1A226M080AC
D1,D5	RLED rs	2	LS Q976	Osram	2.5 V Red LED 1608 (0603) SMD,Osram Opto CHIPLED 0603 LS Q976
D2,D3,D6	GLED rs	3	KP-1608SGC	Kingbright	2.5 V Green LED 1608 (0603) SMD, Kingbright KP-1608SGC

Anexo I: Componentes utilizados

D4	BLED rs	1	LB Q39G-N10O-35-1	Osram	2.85 V Blue LED 1608 (0603) SMD, Osram Opto CHIPLED 0603 LB Q39G-N10O-35-1
FL1	FERRITE rs	1	BLM18HE152SN1D	Murata	Murata Ferrite Bead (Chip Ferrite Bead), 1.6 x 0.8 x 0.8mm (0603 (1608M)), 1500Ω impedance at 100 MHz
J1	FTSH 10 pin rs	1	FTSH-105-01-F-DV	Samtec	Samtec FTSH, 1.27mm Pitch, 10 Way, 2 Row, Straight Pin Header, Surface Mount
J2,J4	2 pin	2	5-146278-2	TE	TE Connectivity AMPMODU Mod II, 2.54mm Pitch, 2 Way, 1 Row, Straight Pin Header, Through Hole
J3	4 pin	1	M20-9990446	HARWIN	HARWIN M20, 2.54mm Pitch, 4 Way, 1 Row, Straight Pin Header, Through Hole
J5	MicroUSB rs	1	105164-0001	Molex	USB Connector, Micro USB Type B, USB 2.0, Receptacle, 5 Ways, Surface Mount, Right Angle
J6	3 pin	1	77311-118-03LF	Amphenol FCI	Amphenol FCI BergStik 77311, 2.54mm Pitch, 3 Way, 1 Row, Vertical PCB Header, Through Hole
L1,L5	7.5 nH rs	2	744761075A	Wurth Elektronik	Wurth WE-KI Series Type 0603A Shielded Wire-wound SMD Inductor with a Ceramic Core, 7.5 nH Wire-Wound 700mA Idc Q:28
L2	27 nH rs	1	744761127C	Wurth Elektronik	Wurth WE-KI Series Type 0603C Shielded Wire-wound SMD Inductor with a Ceramic Core, 27 nH Wire-Wound 600mA Idc Q:35
L3,L4	6.8 nH rs	2	744761068A	Wurth Elektronik	Wurth WE-KI Series Type 0603 Wire-wound SMD Inductor with a Ceramic Core, 6.8 nH ±5% Wire-Wound 700mA Idc Q:30
L6	6.8 uH rs	1	LQH2HPZ6R8MGRL	Murata	Murata LQH Series Type 2520 Shielded Wire-wound SMD Inductor with a Ferrite Core, 6.8 µH Wire-Wound 860mA Idc
R1,R3,R9	10K rs	3	CR0603-FX-1002ELF	Bourns	10kΩ 0603 Thick Film Surface Mount Fixed Resistor ±1% 0.1W - CR0603-FX-1002ELF
R2,R4	6K8 rs	2	MCR03EZPFX6801	ROHM	6.8kΩ 0603 Thick Film SMD Resistor ±1% 0.1W - MCR03EZPFX6801
R5,R11	220 rs	2	CRCW0603220RFKEAHP	Vishay	220Ω 0603 Thick Film Power Resistor ±1% 0.25W - CRCW0603220RFKEAHP
R6,R8,R12	180 rs	3	CRCW0603180RJNEAIF	Vishay	180Ω 0603 Thick Film SMD Resistor ±5% 0.1W - CRCW0603180RJNEAIF
R7	100K rs	1	CR0603-FX-1003ELF	Bourns	100kΩ 0603 Thick Film Surface Mount Fixed Resistor ±1% 0.1W - CR0603-FX-1003ELF
R10	100 rs	1	MCT0603MD1000BP100	Vishay	100Ω 0603 Thin Film Precision Thin Film Surface Mount Fixed Resistor ±0.1% 0.1W - MCT0603MD1000BP100
U1	CC1310 rs	1	CC1310F128RGZT	TI	RF Microcontrollers - MCU SimpleLink Sub-1 GHz Ultra-Low Power Wireless Microcontroller 48-VQFN -40 to 85
U2	MCP1703 rs	1	MCP1703-3302E/DB	Microchip	Microchip MCP1703-3302E/DB, LDO Regulator, 250mA, 3.3 V, ±2% 3+Tab-Pin, SOT-223
U3	ESP32 rs	1	ESP32_DevKit_38pin	Espressif	ESP32_DevKit_38pin
Y1	32K7 Crystal rs	1	Q13FC1350000411	Epson	Epson 32.768kHz Crystal ±20ppm SMD 2-Pin 3.2 x 1.5 x 0.8mm
Y2	24M Crystal rs	1	X1E000021012211	Epson	Epson 24MHz Crystal ±10ppm SMD 4-Pin 3.2 x 2.5 x 0.6mm

Tabla 22 - Central HUB board components

