Universidad de Oviedo

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

GRADO EN INGENIERÍA INFORMÁTICA EN TECNOLOGÍAS DE LA INFORMACIÓN

ÁREA DE ARQUITECTURA Y TECNOLOGÍA DE COMPUTADORES

DESPLIEGUE Y EVALUACIÓN DE UNA APLICACIÓN IOT CLOUD BASADA EN FOG COMPUTING

MEMORIA

D. MARTÍNEZ JIMENEZ Nicolás
TUTOR: D. SUÁREZ ALONSO Francisco José

Junio 2020

Índice de contenidos

1 Introducción 5
2 Alcance y objetivos 8
3 Tecnologías empleadas 9
3.1 SENSORES: STEVAL-MKSBOX1V1 9
3.2 SISTEMAS OPERATIVOS PARA DISPOSITIVOS EN EL BORDE 11
3.3 GATEWAY 14
3.4 ST LINK V2 15
3.5 STM32 CUBE IDE 16
3.6 STM32CUBE PROGRAMMER 17
3.7 AMAZON WEB SERVICES 18
3.8 THINGSBOARD 20
3.9 GRAFANA 21
3.10 NGINX 30
4 Metodología de trabajo 33
5 Desarrollo de la investigación 34
5.1 CONFIGURACIÓN INICIAL DEL MÓDULO 34
5.2 CONFIGURACIÓN DEL GATEWAY 37
5.3 DESARROLLO DEL FIRMWARE 39
5.4 ACTUALIZACIÓN DEL FIRMWARE 50
5.5 EJEMPLO DE USO DE NUBE PÚBLICA 58
5.6 EJEMPLO DE USO DE NUBE PRIVADA 71
6 Prototipo de monitorización de temperatura y humedad en un edificio 89
6.1 RESULTADO OBTENIDOS 94
7 Anexos 101
7.1 PLANIFICACIÓN TEMPORAL 101
8 Conclusiones y trabajo futuro 102
9 Bibliografía 104

Índice de Figuras

Ilustración 1: Gráfico de dispositivos conectados 5
Ilustración 2: Esquema de red IoT 7
Ilustración 3: STEVAL-MKSBOX1V1 10
Ilustración 4: Esquema STEVAL-MKSBOX1V1 10
Ilustración 5: Esquema AWS FreeRTOS 12
Ilustración 6: Placa Raspberry Pi 3 Model B+ 14
Ilustración 7: Raspberry Pi 3 Model B+ con carcasa 15
Ilustración 8: STLINK-V2 y cables de conexión 15
Ilustración 9: STLINK-V2 16
Ilustración 10: STM32CubeIDE 17
Ilustración 11 :STM32CubeProgrammer 18
Ilustración 12: Esquema básico AWS 19
Ilustración 13: Cadena de reglas por defecto 21
Ilustración 14: Demo Grafana 22
Ilustración 15: Panel loggin Grafana 23
Ilustración 16: Tareas básicas Grafana 23
Ilustración 17: Fuentes de datos Grafana 24
Ilustración 18: Configuración fuente de datos 25
Ilustración 19: Crear nuevo panel Grafana 26
Ilustración 20: Configuración panel Grafana 26
Ilustración 21: Ejemplo consulta Grafana 27
Ilustración 22: Gráfico resultante 27
Ilustración 23: Opción share panel Grafana 28
Ilustración 24: Opciones importar gráfico Grafana 28
Ilustración 25: Archivo grafana.ini 29
Ilustración 26: Parámetro allow_embedding 29
Ilustración 27: Permitir modo anónimo 29
Ilustración 28: Salida comando systemctl status nginx 30
Ilustración 29: Página web por defecto 31
Ilustración 30: Contenido index.html 31
Ilustración 31: Contenido temperatura_aulas.com 32
Ilustración 32: Resultado configuración de ejemplo 32
Ilustración 33: ST BLE Sensor APP Example Apps 34
Ilustración 34: ST BLE Sensor APP Barometer 35
Ilustración 35: ST BLE Sensor APP Search Devices 36
Ilustración 36: ST BLE Sensor APP Barometer Display 36
Ilustración 37: Salida comando hcitool lescan 37
Ilustración 38: Salida script getUUID.py 38
Ilustración 39: Salida script temperature.py 38
Ilustración 40: Salida script tempearature_sem.py 39
Ilustración 41: STM32 Cube IDE-opción Import 41
Ilustración 42: STM32 Cube IDE-opción Existing projects into workspace 42

Ilustración 43: STM32 Cube IDE-ventana para importar proyectos 43
Ilustración 44: Timers definidos 44
Ilustración 45: Timers definidos después de la modificación 44
Ilustración 46: Función SendEnviromentalData sin modificar 45
Ilustración 47: Función SendEnviromentalData modificada 45
Ilustración 48: Función ReadTemperatureData 46
Ilustración 49: Función Enviromental_Update 47
Ilustración 50: Función Temperature_Update 48
Ilustración 51: Función HAL_Delay 48
Ilustración 52: Uso de la función HAL_Delay en ProcessThread 49
Ilustración 53: Opción Build all 49
Ilustración 54: Abrir consola 50
Ilustración 55: Salida consola comando DFU 51
Ilustración 56: STM32CubeProgrammer I 51
Ilustración 57: STM32CubeProgrammer II 52
Ilustración 58: Cuadro de diálogo para proceso de borrado correcto 52
Ilustración 59: STM32CubeProgrammer III 53
Ilustración 60: STM32CubeProgrammer IV 54
Ilustración 61: Cuadro de diálogo actualización de firmware correcta 54
Ilustración 62: Montaje para actualización firmware 55
Ilustración 63: STM32CubeProgrammer Botón Firmware Update 56
Ilustración 64: Ventana para ST LINK firmware update botón “Open in update mode” 56
Ilustración 65: Ventana para ST LINK firmware update botón “Upgrade” 57
Ilustración 66: Ventana para ST LINK firmware update mensaje "upgrade successful" 57
Ilustración 67: STM32CubeProgrammer-opción ST-LINK 58
Ilustración 68: Estructura archivo JSON 59
Ilustración 69: Consola AWS 59
Ilustración 70: Consola AWS-Greengrass groups 60
Ilustración 71: Consola AWS-Greengrass cores 61
Ilustración 72: Certificados y claves almacenados 61
Ilustración 73: Salida del comando para lanzar AWS Greengrass 62
Ilustración 74: Opción Crear Tabla consola DynamoDB 62
Ilustración 75: Configuración de la tabla 63
Ilustración 76: Consola AWS-opción Actuar 64
Ilustración 77: Configuración de la regla I 65
Ilustración 78: Configuración de la regla II 65
Ilustración 79: Configuración de la regla III 66
Ilustración 80: Configuración de la regla IV 67
Ilustración 81: Configuración de un nuevo rol 67
Ilustración 82: Salida script temperature_AWS.py 68
Ilustración 83: Contenido de la tabla Temperaturas 68
Ilustración 84: Consola AWS-Pruebas 69
Ilustración 85: Consola AWS-Crear suscripción 70
Ilustración 86: Cliente MQTT consola AWS-mensajes recibidos 70

Ilustración 87: Configuración base de datos Thingsboard 72
Ilustración 88: Comandos para lanzar Thingsboard 72
Ilustración 89: Formulario login Thingsboard 73
Ilustración 90: Consola administrador Thingsboard 74
Ilustración 91: Gestión de organizaciones 74
Ilustración 92: Crear organización 75
Ilustración 93: Botón gestionar administradores de la organización 76
Ilustración 94: Gestión usuarios organización 76
Ilustración 95: Crear nuevo usuario 77
Ilustración 96: Diálogo para activar cuenta nuevo usuario 77
Ilustración 97: Formulario para cambio de contraseña nuevo usuario 78
Ilustración 98: Consola administrador de la organización 78
Ilustración 99: Botón para agregar nuevo activo 79
Ilustración 100: Agregar nuevo activo 80
Ilustración 101: Creación nuevo dispositivo 81
Ilustración 102: Botón agregar relación 82
Ilustración 103: Agregar relación 83
Ilustración 104: Estructura archivo JSON 84
Ilustración 105: Salida script temperature_TB.py 85
Ilustración 106: Última telemetría sensor 85
Ilustración 107: Agregar panel 85
Ilustración 108: Widgets del panel 86
Ilustración 109: Botón Alias de las entidades 86
Ilustración 110: Ventana para crear nuevo alias 87
Ilustración 111: Ventana para agregar datos al widget 87
Ilustración 112: Gráfico temperatura sensor 88
Ilustración 113: Fichero de direcciones y ubicaciones 90
Ilustración 114: Script m_temp_DI.py 90
Ilustración 115: archivo temperatura_aulas.com 91
Ilustración 116: Salida script m_temp_DI.py 92
Ilustración 117: Última telemetría I 92
Ilustración 118: Última telemetría II 93
Ilustración 119: Última telemetría III 93
Ilustración 120: Últimos valores página web 94
Ilustración 121: Ubicación RPi 95
Ilustración 122: Ubicación en Sala de reuniones 96
Ilustración 123: Ubicación en Sala de postgrados 96
Ilustración 124: Ubicación en Sala del café 97
Ilustración 125: Plano de la primera planta del edificio departamental Este 98
Ilustración 126: Temperatura y humedad media 99
Ilustración 127: Temperatura últimas 24 horas de la Sala de café 100
Ilustración 128: Humedad últimas 24 horas de la Sala de café 100
Ilustración 129: Planificación Temporal 101
Ilustración 130: Diagrama de GANTT 101

1 Introducción

El IoT o "Internet of Things" es una tecnología relativamente nueva y en crecimiento. Se puede definir como "la red de objetos físicos que llevan sensores integrados, software y otras tecnologías con el fin de conectar e intercambiar datos con otros dispositivos y sistemas a través de Internet"[1].

La idea de interconectar dispositivos ya existía años atrás, pero se considera que el nacimiento del IoT fue en 1999, cuando el tecnólogo inglés Kevin Ashton utilizó este término por primera vez[2].

Hoy en día IoT ya no es un concepto, es una realidad. Se estima que para 2030 habrá unos 125 millones de dispositivos IoT. La aparición del 5G, y el interés de las empresas, hacen creer que alcanzar este número de dispositivos no sea tan descabellado. Se calcula que las empresas invertirán 1.1 billones de dólares en IoT para 2023.[3]

Ilustración 1: Gráfico de dispositivos conectados

El IoT es una tecnología relativamente nueva y con mucho camino por recorrer, pero a la vista de estos datos, esta tecnología es una apuesta para el futuro.

[1] (¿Qué es IoT?)
[2] (Birth of IoT)
[3] (IoT statistics 2020), (The IoT rundown for 2020)

La obtención de la información es realizada por los dispositivos en el borde o *edge devices*. Los dispositivos en el borde son aquellos dispositivos, como sensores o actuadores, encargados de la captación o generación de datos. Pero los datos no aportan valor por sí solos, es necesario traducirlos de alguna forma en información útil. Esta labor la realiza la computación en la nube o *Cloud Computing*. Puede definirse como la prestación de servicios informáticos, como almacenamiento, análisis, bases de datos… a través de Internet. En otras palabras, los dispositivos se encargan de recopilar los datos para enviarlos a través de internet a la nube, donde serán tratados, almacenados y analizados.

Existen tres tipos de nube[4]:

- **Nube pública:** propiedad de un proveedor externo de servicios, que proporciona servicios informáticos a través de internet. Accesibles para cualquier usuario.
- **Nube privada:** propiedad de una empresa u organización exclusivamente. Solo accesible para algunos usuarios.
- **Nube híbrida:** combina nube pública y nube privada.

Una red IoT puede estar formada por un número ilimitado de dispositivos. Estos dispositivos pueden captar grandes cantidades de información o datos, lo que puede causar problemas de latencia a la hora de enviarlos a la nube. En ciertos casos, como por ejemplo en sistemas de control de seguridad de maquinaria industrial, la demora en la comunicación de un dato puede suponer un alto coste humano o económico. Para lidiar con este problema surge la computación en la niebla o *Fog Comuting*.

El *Fog Computing* consiste en agregar dispositivos inteligentes entre los dispositivos en el borde y la nube. Así se consigue añadir una capa intermedia que realice parte de las funcionalidades de la nube, y acercar la inteligencia a los dispositivos en el borde.[5] En la siguiente ilustración se presenta un esquema que resume los elementos que, a gran escala, forman una red IoT que implemente *Fog Computing.*

[4] (Types of cloud computing)
[5] (Fog computing)

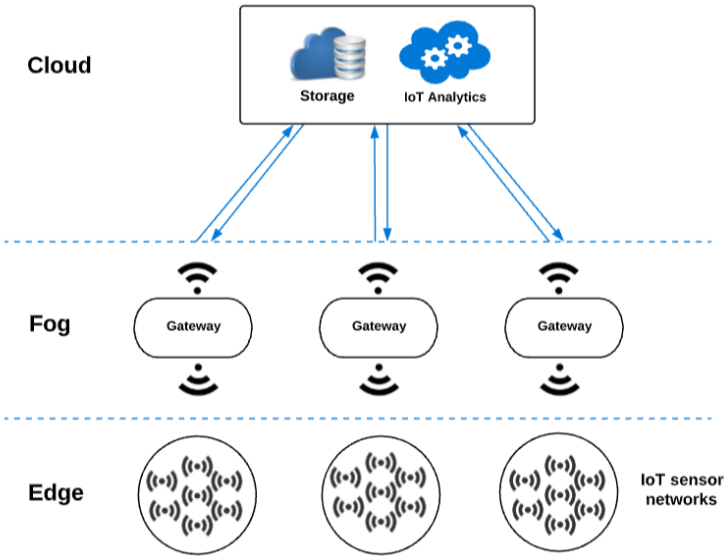

Ilustración 2: Esquema de red IoT

Realizar parte de la computación en la capa intermedia también permite reducir costes. Los datos pueden ser tratados y filtrados para enviar solo aquellos que proporcionen información útil. Esto permite reducir costes debido a que la cantidad de información enviada, procesada y almacenada en la nube es menor.

Por otro lado, también cabe la posibilidad de delegar en los dispositivos en el borde parte de la computación. A esto se le conoce como *Edge Computing*[6]. Al igual que el *Fog Computing*, permite reducir la latencia, ya que no es necesario comunicarse con la nube para realizar el procesamiento de los datos, estos se procesan in situ. Asimismo, permite reducir costes, ya que los datos pueden ser filtrados previamente para solo enviar aquellos que aporten valor.

Por tanto, es posible distribuir la inteligencia entre los diferentes elementos que forman la red IoT: la nube, la capa intermedia o los dispositivos en el borde. Todo dependerá de las necesidades, de la cantidad de datos que se estén procesando y de las características de los dispositivos.

[6] (Edge Computing)

2 Alcance y objetivos

En este proyecto se busca ganar experiencia y adquirir conocimiento en el mundo del IoT. Más en concreto, se analizan distintas tecnologías, protocolos y herramientas relacionados con dispositivos en el borde, transferencia de datos por la red, administración de dispositivos y almacenamiento y visualización de los datos, con el fin de poder crear aplicaciones IoT basadas en *Fog Computing*.

En último lugar, se desarrolla un pequeño prototipo para plasmar los conocimientos obtenidos. Este prototipo permite monitorizar la temperatura y la humedad de ciertos espacios del edificio donde tiene su sede en Gijón el Departamento de Informática de la Universidad.

En la investigación se abordan los siguientes temas:

- Configuración de dispositivos en el borde: para la obtención de datos se utiliza el sensor de temperatura equipado en un módulo multisensor. Este dispositivo cuenta con una antena bluetooth que permite la transmisión de los datos a otros dispositivos que trabajen con esta tecnología. Pero también se valora usar otras tecnologías que amplíen la funcionalidad de este módulo sensor.

- Comunicación entre dispositivos: todos los datos recogidos por los dispositivos en el borde deben de viajar a través de la red hasta llegar a la nube. Esta tarea puede ser realizada por los sensores, o por dispositivos intermediarios que actúen como gateway si se quiere seguir el paradigma del *Fog Computing*. En este proyecto se utiliza una Raspberry Pi Model B+ como intermediario entre los dispositivos en el borde y la nube.

- Gestión y almacenamiento de los datos: el objetivo final de la aplicación IoT a desarrollar es mostrar gráficos con los datos obtenidos. Para ello, es necesario almacenarlos en algún lugar donde sean accesibles, en la nube. Se usan ejemplos tanto de nube pública como privada para llevar a cabo esta tarea.

- Visualización de los datos: para que los usuarios puedan ver los valores de temperatura y humedad de los diferentes espacios, es necesario poner a su disposición gráficos donde se muestren los datos. Se han buscado herramientas que posibiliten esto.

3 Tecnologías empleadas

Para el desarrollo del proyecto se utilizan distintos dispositivos y herramientas software, los cuales se describen a continuación.

3.1 SENSORES: STEVAL-MKSBOX1V1

Para la recogida de datos de temperatura y humedad se emplean varios nodos multisensor IoT, que son capaces de medir la temperatura y la humedad del lugar y transmitirla a otro dispositivo que actúa como gateway (puerta de enlace). En un principio, comenzamos con 2 nodos multisensor "Steval-MKSBOX1V1"[7]. Estos son kits multisensor, equipados con sensores ambientales y de movimiento y un microcontrolador ultra-low-power ARM Cortex-M4 STM32L4R9 basado en el núcleo Arm® Cortex®-M4 32-bit RISC.

Como este proyecto se centra en la medición de la temperatura y de humedad, solo se usa el sensor de temperatura y humedad HTS221. El HTS221 es un sensor ultra compacto que permite medir humedad relativa y temperatura. Realiza mediciones de humedad relativa y temperatura con una precisión de ± 3.5% y de ± 0.5 °C respectivamente.

Cuenta con un modo stand-by. En este modo, el sensor realiza una única lectura de la temperatura y se apaga durante un intervalo de tiempo, hasta realizar la siguiente lectura. De esta forma se pretende incrementar el tiempo de funcionamiento del sensor, al reducir el consumo de batería.

La placa se encuentra dentro de una caja azul, y está equipada con una batería recargable. La placa y la caja contenedora se muestran en la siguiente ilustración.

[7] (Steval-MKSBOX1V1)

Ilustración 3: STEVAL-MKSBOX1V1

Tiene conectividad con móviles u otros dispositivos a través de protocolo Bluetooth Low Energy (BLE). Este es un subconjunto del estándar Bluetooth que permite comunicaciones más eficientes y con un menos consumo energético.[8]

Además del sensor nombrado anteriormente, el STEVAL-MKSBOX1V1 está compuesto por otros sensores, los cuales se muestran en la Ilustración 4.

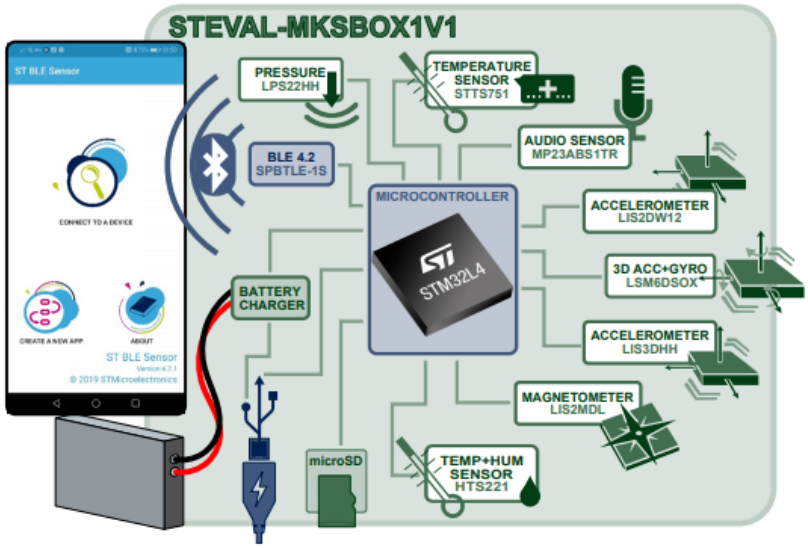

Ilustración 4: Esquema STEVAL-MKSBOX1V1

[8] (BLE)

3.2 SISTEMAS OPERATIVOS PARA DISPOSITIVOS EN EL BORDE

Debido al gran desarrollo del IoT, han surgido un gran número de sistemas operativos para dispositivos en el borde. Estos sistemas operativos buscan llevar la funcionalidad de un sistema operativo tradicional a los dispositivos en el borde (sensores o actuadores), teniendo en cuenta las limitaciones hardware de estos debido a su pequeño tamaño. Se ha investigado en profundidad sobre dos de los sistemas operativos más relevantes, FreeRTOS[9] y Contiki-NG[10].

FreeRTOS es un sistema operativo de tiempo real para microcontroladores y pequeños microprocesadores gratuito, soportado en más de 40 arquitecturas. La lista oficial de dispositivos a los que se puede portar FreeRTOS puede encontrarse en la web[11].

Fue desarrollado por diversas compañías del sector. Proporciona bibliotecas que facilitan la programación, la implementación, la protección, la conexión y la administración de los dispositivos en el borde. Las bibliotecas proporcionadas se dividen en 3 categorías:

- **FreeRTOS+:** proporciona conectividad y funciones adecuadas para construir dispositivos inteligentes basados en microcontroladores y conectar dispositivos IoT a la nube.
    - FreeRTOS+TCP: pila TCP/IP ligera
    - FreeRTOS+Cli: para añadir intérprete de línea de comandos
    - FreeRTOS+IO: para añadir funciones tipo Linux/POSIX como open(), read(), write(), ioctl() a las bibliotecas de drivers de periféricos
    - FreeRTOS+UDP: pila UDP/IP ligera

- **IoT Libraries:** proporciona conectividad y funcionalidades de seguridad para construir dispositivos inteligentes basados en microcontroladores y conectarlos a la nube.
    - MQTT: para añadir protocolo MQTT
    - HTTPS: para añadir protocolo HTTPS
    - OTA: para controlar, descargar y verificar actualizaciones de firmware
    - PKCS#11: API layer que abstrae el almacenamiento de claves y otras operaciones relacionadas con la seguridad
    - AWS IoT Jobs: conjunto de operaciones remotas enviadas o ejecutadas en 1 o más dispositivos conectados a AWS IoT
    - AWS IoT Device Shadow: representación virtual y persistente de un dispositivo conectado a AWS IoT

[9] (FreeRTOS)
[10] (Contiki-NG)
[11] (FreeRTOS ports)

- **FreeRTOSLabs:** son completamente funcionales, pero están siendo optimizadas o refactorizadas para mejorar el uso de memoria, modularidad, documentación, usabilidad de demostración o cobertura de prueba.
    - MQTT Lightweight Demo: cliente MQTT que no requiere asignación dinámica de memoria
    - POSIX: contenedor de subprocesos POSIX para el kernel FreeRTOS
    - FreeRTOS+FAT: sistema de archivos FAT compatible con subprocesos

AWS tiene su propia implementación, la cual permite conectar dispositivos fácilmente con los servicios en la nube de AWS. En la siguiente ilustración se muestra un esquema de una red IoT donde los nodos, que utilizan la implementación de AWS de FreeRTOS, se comunican con los servicios en la nube de AWS.

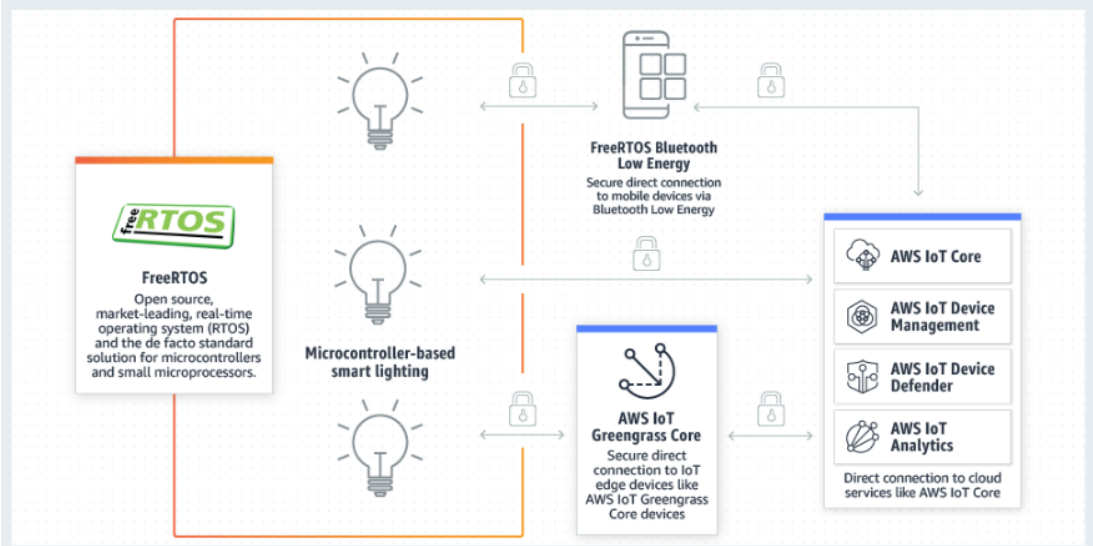

Ilustración 5: Esquema AWS FreeRTOS

Esta implementación es soportada solo por algunas plataformas. En el caso de STMicroelectronics, está disponible únicamente para STM32L4 Discovery Kit IoT Node,B-L475E-IOT01A[12].

Por otro lado, se encuentra Contiki-Ng, que es un sistema operativo de código abierto para dispositivos IoT. Se centra en la comunicación segura y de bajo consumo y protocolos estándar como IPv6/6LoWPAN, 6TiSCH, RPL, and CoAP.

Contiki-NG está soportado en las siguientes plataformas:

- cc2538dk: TI cc2538 development kit
- cooja: Cooja native motes platform
- jn516x: NXP jn516x series
- native: Contiki-NG as a native process
- nrf52dk: Nordic Semiconductor nRF52 development kit

[12] (STM32L4 Discovery Kit Node)

- openmote-cc2538: OpenMote cc2538
- simplelink: TI SimpleLink MCU Platform
- sky: Tmote Sky / TelosB
- cc26x0-cc13x0 / srf06-cc26xx: TI cc26x0 and cc13x0 platforms
- zoul: Zolertia Zoul platforms: Firefly, RE-mote and Orion

Pero si no se dispone de una de las plataformas anteriores, se puede portar Contiki-NG a un nuevo dispositivo hardware siempre que el dispositivo cuente con:

- Un microcontrolador(ej. Arm Cortex)
- Una antena (ej. antena BLE)
- Varios dispositivos off-chip (ej. sensores)

Existe una guía en el repositorio de GitHub donde se explican los pasos a realizar para portar Contiki-NG a una nueva plataforma hardware[13].

El Steval-MKSBOX1V1 cumple los requisitos para poder portar en él Contiki-NG, ya que cuenta con un microcontrolador ARM CortexM4, una antena bluetooth y sensores ambientales y de movimiento. Pero debido a que solo cuenta con una antena bluetooth no es posible utilizar la mayoría de los protocolos que Contiki-NG soporta. Únicamente se puede utilizar como protocolo alternativo BLEach[14].

BLEach es una pila IPV6 sobre BLE de código abierto que permite intercambiar paquetes IPv6 a través de conexiones BLE de acuerdo con el estándar RFC 766[15], donde se describe como realizar comunicaciones IPv6 a través de conexiones BLE.

Utilizando BLEach se consigue ampliar la conectividad de los dispositivos, de modo que puedan comunicarse con otros que soporten comunicación IPv6 sin necesidad de utilizar un gateway intermedio. Admite cualquier capa de transporte sobre IPv6. Por lo tanto, BLEach puede reutilizar cualquier implementación de IPv6 para dispositivos en el borde.

En un principio, las comunicaciones inalámbricas de una red de sensores de bajo consumo que quisieran comunicarse con otros dispositivos estaban limitadas a usar IPv6 sobre IEEE 802.15.4 (6LowPAN). Utilizando BLEach se consigue mejorar las comunicaciones en comparación con 6LowPAN. En concreto, consigue un menor consumo de batería y una reducción de los tiempos de procesamiento de datos[16].

Portar Contiki-NG proporcionaría diversas ventajas como se ha descrito anteriormente. Pero esta tarea no es algo trivial, y escapa del alcance de este proyecto. Por lo tanto, se opta por utilizar BLE como protocolo de comunicación, y se emplea un gateway capaz de comunicarse con los sensores a través de BLE.

[13] (Guía para portar Contiki-NG)
[14] (BLEach)
[15] (RFC 7668)
[16] (BLEach: Exploiting the Full Potential of IPv6 over BLE in Constrained Embedded IoT Devices)

3.3 GATEWAY

Se utiliza un ordenador de placa reducida Raspberry Pi 3 Model B+[17]. Se encarga de comunicarse con los sensores, recopilar las temperaturas leídas por los sensores, tratarlas si es necesario y enviarlas al almacenamiento en la nube. En el presente documento se emplea la abreviación RPi para referirse a ella.

Características generales:

- CPU + GPU: Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
- RAM: 1GB LPDDR2 SDRAM
- Wi-Fi + Bluetooth: 2.4GHz y 5GHz IEEE 802.11.b/g/n/ac, Bluetooth 4.2, BLE
- Ethernet: Gigabit Ethernet sobre USB 2.0 (300 Mbps)
- 4 puertos USB 2.0
- Micro-SD

En las siguientes ilustraciones se muestra la RPi. En la primera de ellas la parte superior de la placa. En la segunda colocada dentro de la carcasa de plástico donde irá colocada en su ubicación definitiva.

Ilustración 6: Placa Raspberry Pi 3 Model B+

[17] (Raspberry Pi 3 Model B+)

Ilustración 7: Raspberry Pi 3 Model B+ con carcasa

3.4 ST LINK V2

El ST LINK V2 18Es un "debugger" y "programmer" (depurador y programador) para las familias de microcontroladores de STM32 y STM8. De entre todos los elementos y cables de conexión, se utilizan los indicados a continuación:

Ilustración 8: STLINK-V2 y cables de conexión

A. ST LINK V2 depurador y programador
B. Cable estandar USB a mini-usb
C. Adaptador JTAG 20-10 pines
D. Cable de cinta con conector de 20 pines
E. Cable de cinta de 14-10 pines

[18] (ST LINK V2)

El debugger tiene las siguientes interfaces y elementos:

Ilustración 9: STLINK-V2

A. Interfaz para conector STM32 JTAG y SWD
B. Interfaz para conector STM8 SWIM
C. LED de actividad
D. Interfaz para cable mini USB

3.5 STM32 CUBE IDE

STM32CubeIDE[19] es una herramienta de desarrollo gratuita, parte del ecosistema de software STM32Cube. STM32Cube es una iniciativa de la compañía que busca mejorar la productividad de los diseñadores reduciendo el esfuerzo y los costes a la hora de desarrollar código.

STM32CubeIDE integra todos los servicios de la herramienta STM32CubeMX en una sola herramienta. De modo que permite la configuración de periféricos, generación y compilación de código y depuración de características para microcontroladores y microprocesadores de la compañía STM. Está basado en Eclipse, por lo que la forma de trabajar será similar. En la Ilustración 10 se puede ver una captura del IDE.

[19] (STM32CubeIDE)

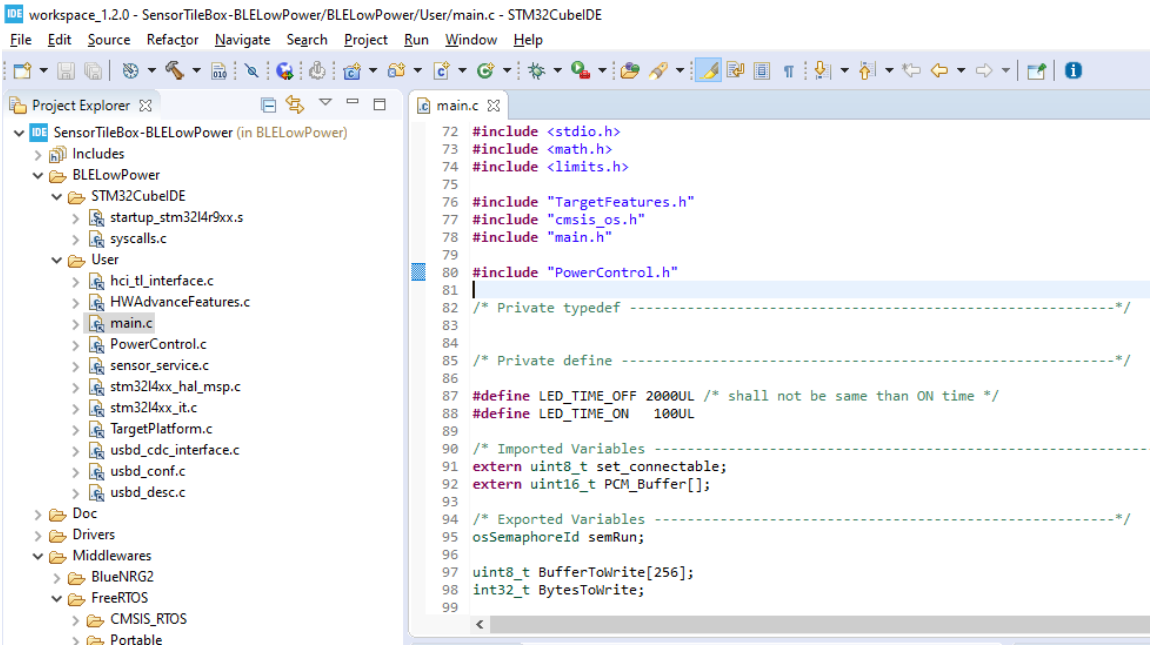

Ilustración 10: STM32CubeIDE

3.6 STM32CUBE PROGRAMMER

STM32CubeProgrammer[20] es una herramienta gratuita diseñada por la compañía STMicroelectronics. Forma parte del ecosistema de software STM32Cube. STM32Cube es una iniciativa de la compañía que busca mejorar la productividad de los diseñadores reduciendo el esfuerzo y los costes a la hora de desarrollar código.

STM32CubeProgrammer cuenta con una interfaz a través de línea de comandos, y una interface gráfica, que es la usada en este proyecto. Proporciona un entorno para leer, escribir y verificar la memoria de dispositivos. En el proyecto se usa para cargar los binarios desarrollados. En la siguiente ilustración se muestra la interfaz gráfica del STM32CubeProgrammer.

[20] (STM32CubeProgrammer)

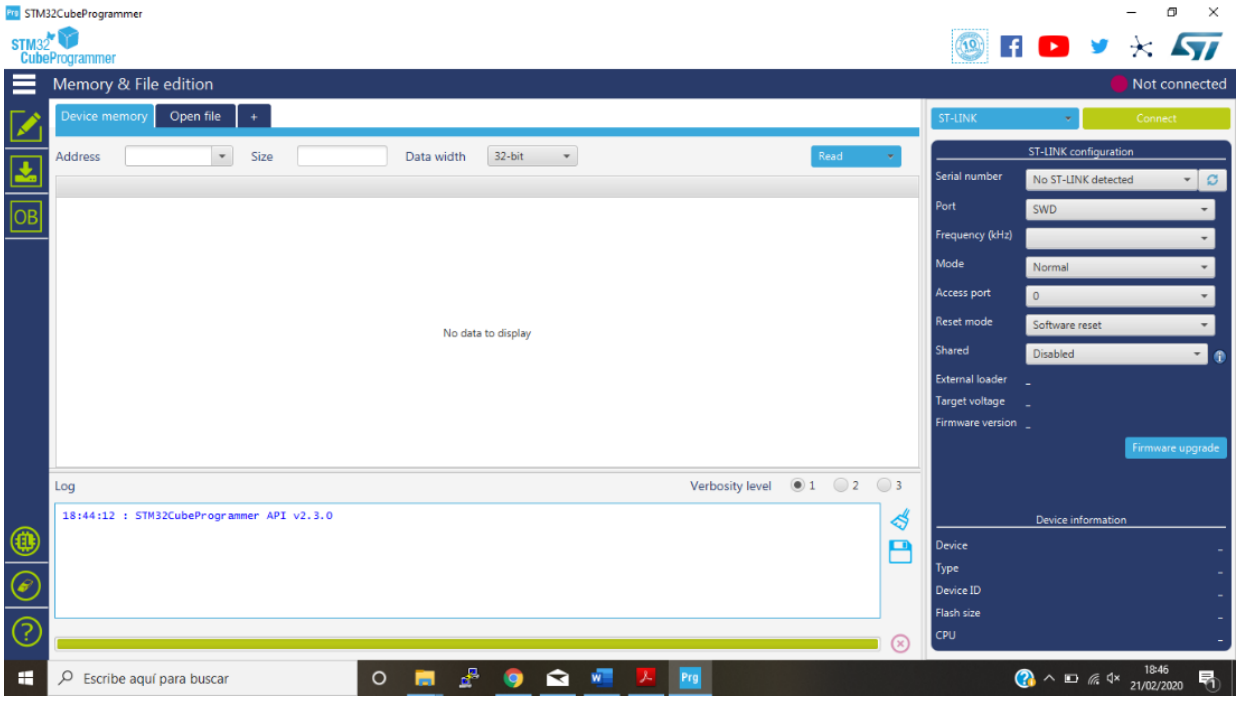

Ilustración 11 :STM32CubeProgrammer

3.7 AMAZON WEB SERVICES

Amazon proporciona un conjunto de servicios para almacenamiento, procesamiento de datos, machine learning, iot … dentro de la plataforma AWS (Amazon Web Services)[21]. Es la plataforma en la nube más utilizada a nivel mundial, por eso es de interés comenzar a trabajar con ella. Su mayor inconveniente es que es una plataforma de pago. Para nuestro proyecto se hace uso de los servicios AWS IoT Greengrass[22] y AWS DynamoDB[23].

Greengrass posibilita interconectar entre sí y utilizar los servicios de AWS, a los dispositivos encargados de recopilar y enviar los datos (edge devices). La forma de trabajar básica de Greengrass se ilustra en la Ilustración 12:

[21] (Amazon Web Services)
[22] (AWS IoT Greengrass)
[23] (DynamoDB)

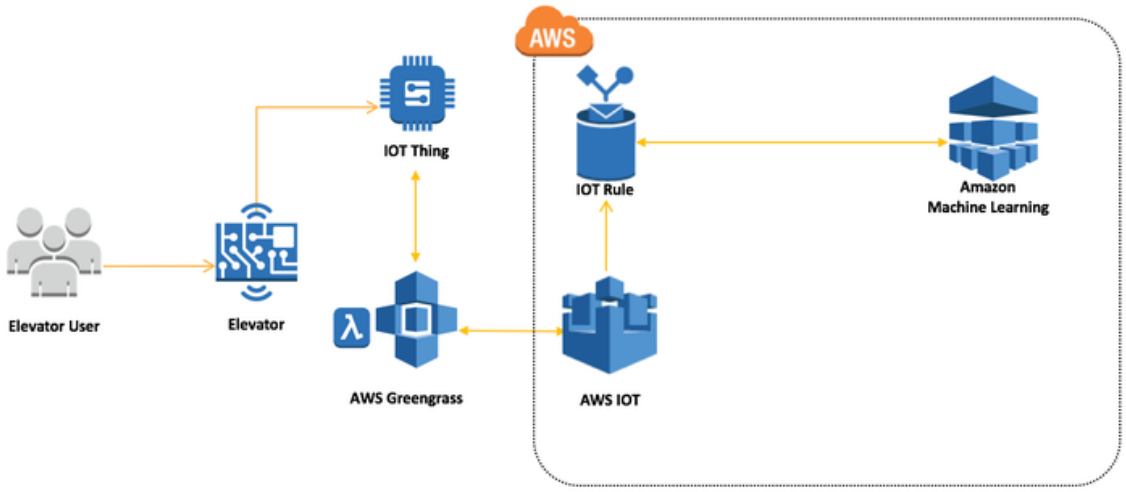

Ilustración 12: Esquema básico AWS

Greengrass trabaja con grupos. Cada grupo es un conjunto de opciones de configuración y dispositivos. Por ejemplo, los sensores y el gateway de una sala. De este modo se agrupan todos los elementos que van a interaccionar entre sí. Todos los grupos deben de tener obligatoriamente un "Greengrass core" que realiza la función de intermediario entre los dispositivos de la red local y la nube. En él se ejecuta la lógica de aplicación.

Además del Greengrass core, los grupos pueden estar formados por otros elementos:

- **Funciones Lambda:** lista de funciones Lambda que se ejecutan en el núcleo. Estas funciones se definen en el servicio AWS Lambda que permite ejecutar funciones sin aprovisionar ni administrar servidores.

- **Suscripciones:** una suscripción está formada por un origen del mensaje, un destino y un "topic" o tópico, que permite filtrar mensajes. Los mensajes se envían a través del protocolo MQTT. Los tópicos son los filtros que usa el broker MQTT para filtrar los mensajes publicados y discriminar a que clientes suscritos es entregado. Las suscripciones permiten a los dispositivos comunicarse con funciones lambda o con conectores.

- **Conectores:** son módulos precompilados que permiten agilizar el desarrollo. Se ejecutan de forma local en el core.

- **Dispositivos:** todos aquellos dispositivos conectados al core. Por ejemplo, sensores.

- **Sombras:** la sombra de un dispositivo es un documento JSON que almacena el estado actual de un dispositivo. La sombra se puede utilizar para obtener información sobre el estado de un dispositivo, aunque este no tenga conexión a internet.

- **Recursos:** conjunto de recursos locales almacenados en el core. De esta forma las funciones lambda y conectores pueden acceder a datos, modelos de predicción o secretos (contraseñas, claves API …) almacenadas localmente.

- **Reglas:** en un grupo pueden definirse una o varias reglas en las que se especifique una acción a realizar cuando se produzca un evento en concreto. Los eventos se filtran usando los tópicos de los mensajes MQTT.

3.8 THINGSBOARD

Thingsboard[24] es una plataforma IoT de código abierto que permite la recolección de datos, el procesamiento, la visualización y la gestión de dispositivos. Con este software se puede gestionar fácilmente las diferentes redes IoT desplegadas, los elementos que las forman y los datos recopilados.

Soporta "multitenancy" o multitenencia, es decir, que una única instancia de Thingsboard puede ejecutar múltiples instancias de aplicación. Para acceder a cada instancia de aplicación es necesario un usuario. Existen 3 roles diferentes:

Administrador del sistema: es responsable de la configuración general y de seguridad del sistema. Tiene la capacidad de crear "Organizaciones" y nuevos usuarios. Cada "Organización" representa un individuo o empresa que es propietario de dispositivos.

- **Administrador de la organización:** tiene la capacidad de crear y gestionar dispositivos, activos, clientes, gráficos y configurar el "rule engine".

- **Cliente:** solo puede acceder a los dispositivos o gráficos que el administrador de la organización le haya asignado.

Los principales elementos que forman la estructura de una red IoT administrada en Thingsboard son los siguientes:

- **Activos:** Thingsboard permite jerarquizar entidades a través de relaciones entre activos. Por ejemplo, una organización que gestione la domótica de las casas de un barrio tendrá un activo por cada casa, y un activo barrio que contendrá todas las casas.

- **Dispositivos:** todos aquellos dispositivos que estén situados en el borde y que recopilen información o realicen alguna acción. Por ejemplo, sensores. Pueden comunicarse con Thingsboard a través del protocolo MQTT, CoAP o HTTP.

[24] (Thingsboard)

- **Rule engine:** Thingsboard incluye un motor de reglas que permite establecer cadenas de reglas donde se describen acciones a realizar cuando ocurren eventos. Por ejemplo, se puede definir una regla que establezca que cada vez que un dispositivo publique una lectura, esta se almacene en la base de datos. Por defecto, existe una cadena de reglas que define las acciones más básicas. En la Ilustración 13 se puede ver la cadena de reglas que por defecto viene creada. En ella se define que hacer cuando un cliente actualiza sus atributos, cuando actualiza los datos (telemetría), cuando envía una llamada RPC (Remote Procedure Call) u otras acciones.

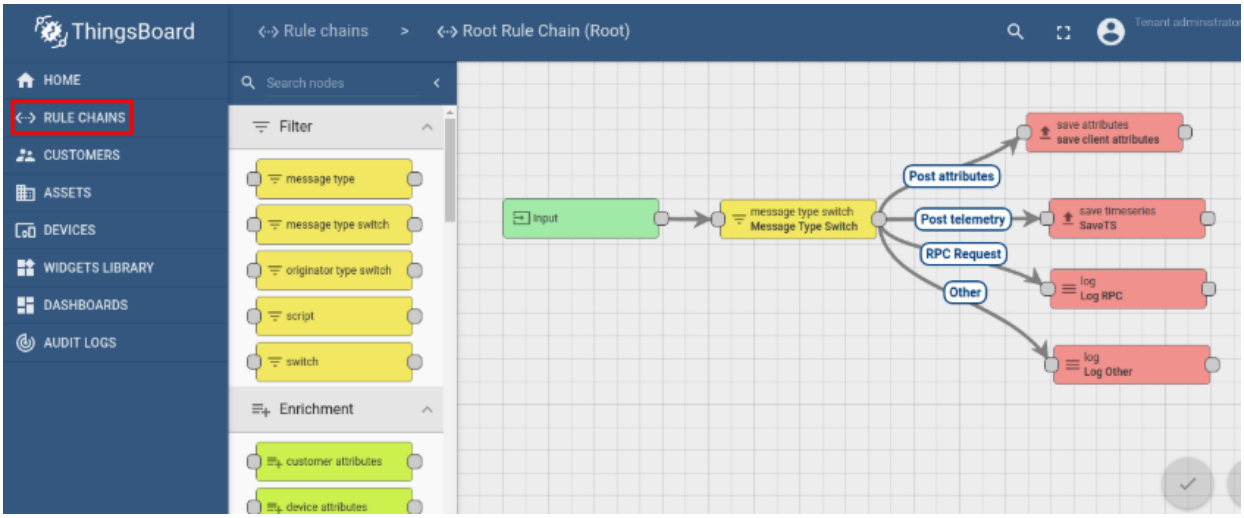

Ilustración 13: Cadena de reglas por defecto

- **Alarmas:** Thingsboard permite establecer alarmas que se activan cuando ocurre un suceso que se quiere registrar. Por ejemplo, cuando un sensor falla al intentar actualizar la telemetría.

3.9 GRAFANA

Grafana[25] es un software de código abierto para la visualización y análisis de datos de series temporales. Cuenta con más 400.000 instalaciones activas. Algunas de las compañías que hacen uso de este software son Ebay, Paypal o Digital Ocean…

Esta herramienta permite transformar fácilmente los datos en gráficos y sin importar donde se encuentren almacenados. Admite datos de más de 30 fuentes de datos distintas. Otra de sus ventajas es la diversidad de gráficos existentes, que se encuentra en aumento constante debido a la contribución los de usuarios.

[25] (Grafana)

Tanto por sus ventajas, porque permite la exportación de sus gráficos de forma sencilla, y por ser una herramienta útil para posibles proyectos futuros, se ha apostado por su uso en este proyecto.

Ilustración 14: Demo Grafana

Ha sido necesario instalar y configurar Grafana. Se siguieron estos pasos para realizarlo:

Se descarga el software y se instala:

```
sudo apt-get install -y adduser libfontconfig1
```

```
wget https://dl.grafana.com/oss/release/grafana_6.7.3_amd64.deb
```

```
sudo dpkg -i Grafana grafana_6.7.3_amd64.deb
```

A continuación, se lanza:

```
sudo service grafana-server start
```

Una vez arrancado, es posible acceder a la página de inicio de sesión, con la IP de la máquina virtual y el puerto 3000, para comenzar a configurar Grafana. Aparece la siguiente página, donde introducimos las credenciales para el usuario “admin” creado por defecto.

Ilustración 15: Panel loggin Grafana

Tras realizar el login con éxito, se muestra el dashboard de inicio. En él, se listan las tareas básicas a realizar para configurar Grafana con éxito. Se realizan cada una de ellas. Como se puede apreciar en la Ilustración 15, la primera de ellas, "Instalar Grafana", ya ha sido completada. Ahora se agrega una base de datos.

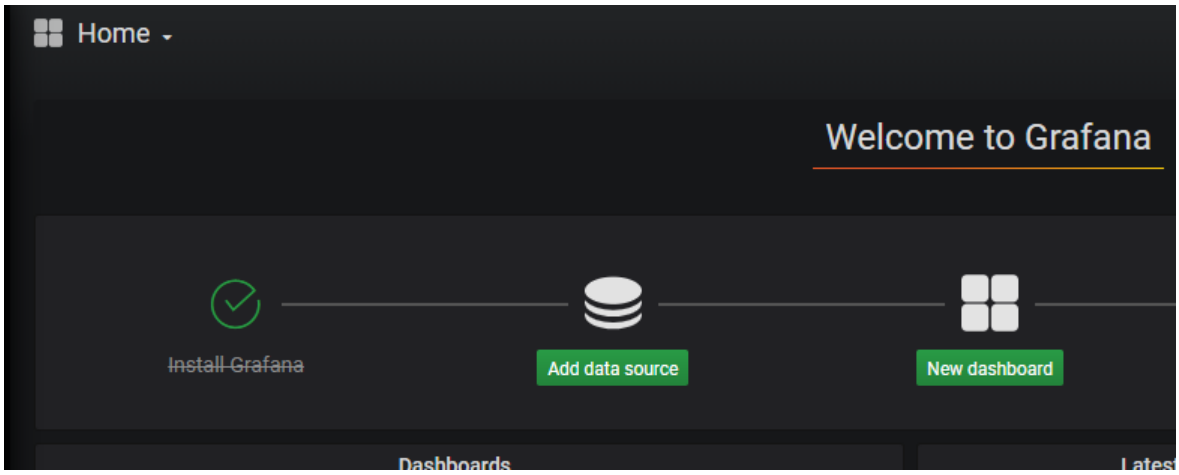

Ilustración 16: Tareas básicas Grafana

Al pulsar sobre el botón "Add data source", se nos muestra la siguiente página, donde se debe escoger y configurar la base de la que se obtienen los datos a visualizar. En nuestro caso, se selecciona Postgres SQL, para utilizar la base de datos de Thingsboard.

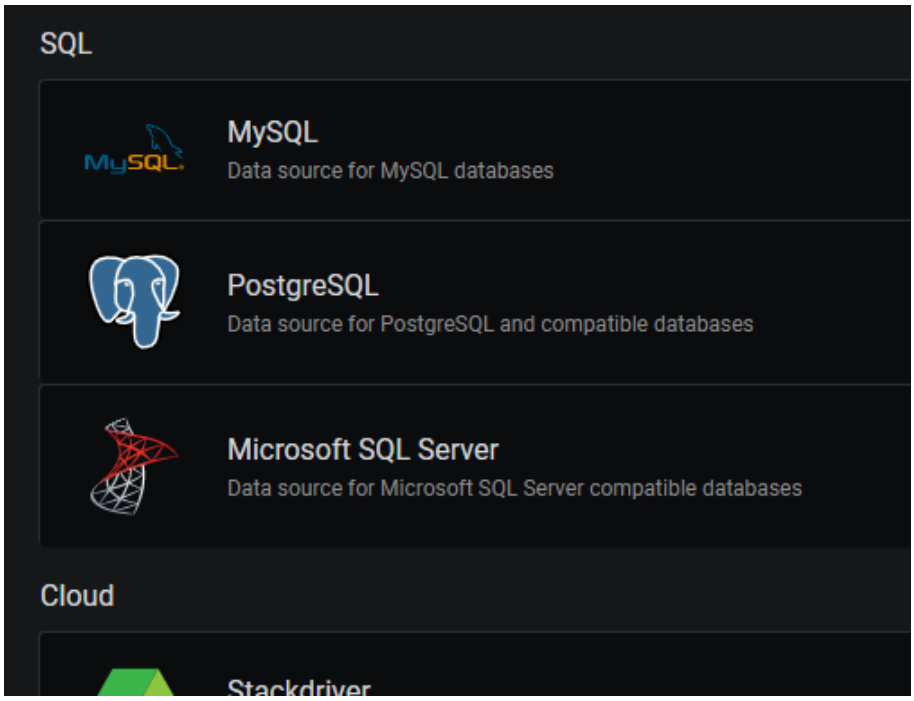

Ilustración 17: Fuentes de datos Grafana

El siguiente paso es configurar la conexión con la base de datos de Thingsboard.

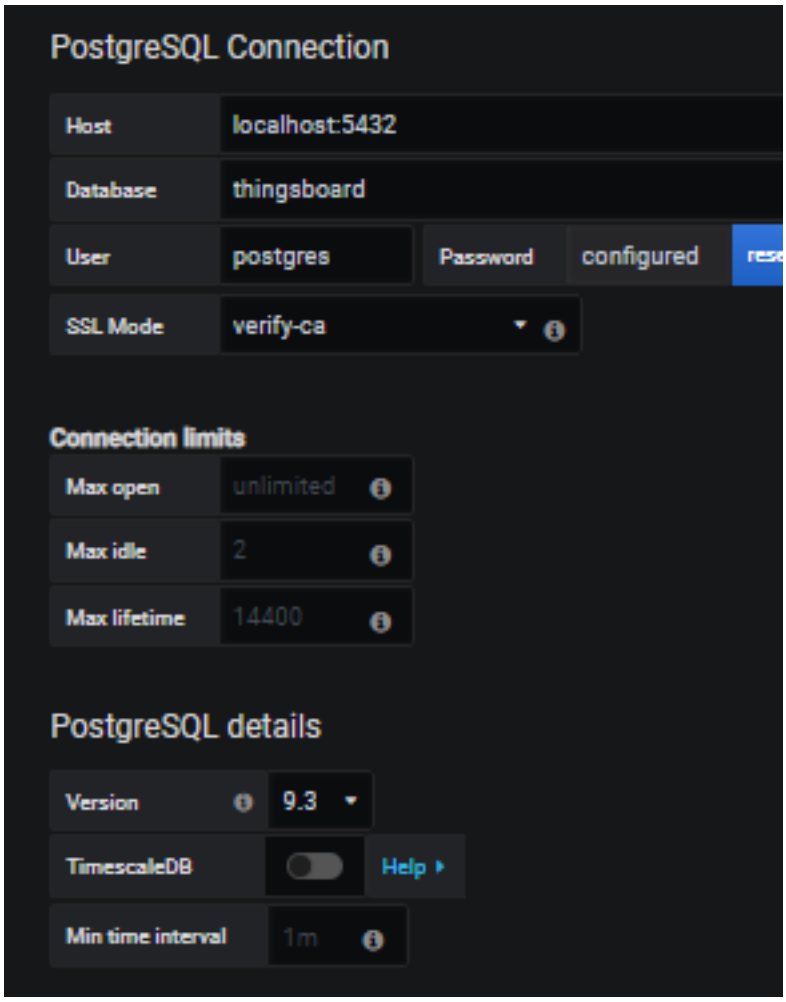

Ilustración 18: Configuración fuente de datos

Para visualizar los datos es necesario crear un dashboard. Los dashboards es la forma en la que Grafana organiza los paneles (gráficos). Cada dashboard está formado por uno o varios paneles.

En nuestro caso, se crea el dashboard “Temperatura Aulas” que alberga todos los paneles donde se muestran las lecturas de temperaturas recogidas por los sensores. Para realizar esta acción, seleccionamos la opción “New dashboard” del dashboard de inicio. Esto conduce a un nuevo dashboard donde se pueden añadir gráficos.

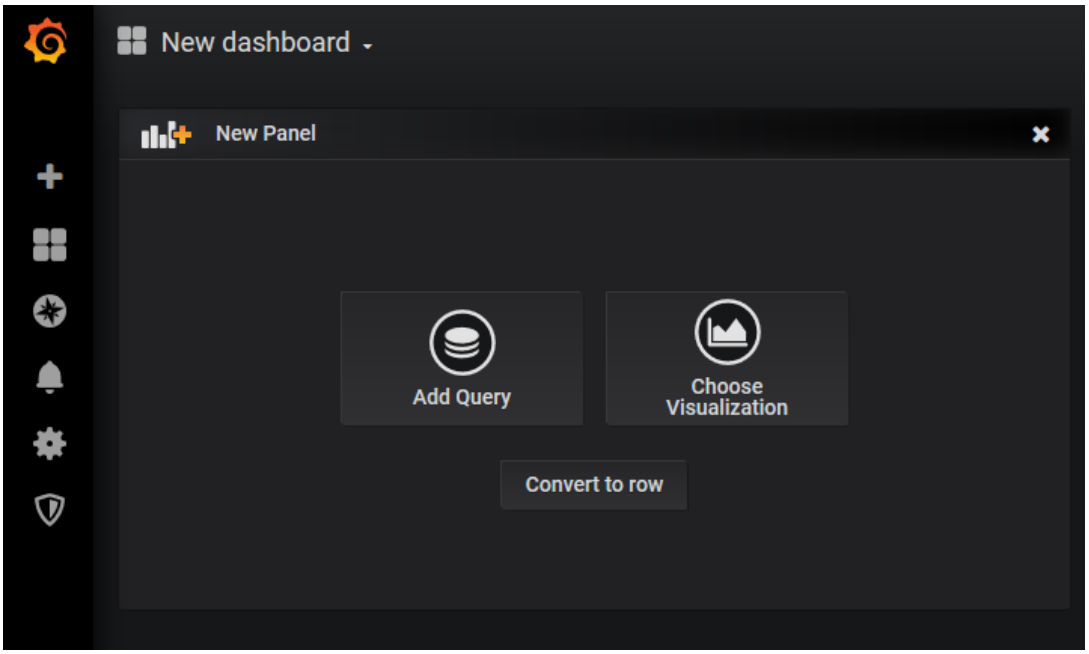

Ilustración 19: Crear nuevo panel Grafana

Se comienza eligiendo los datos que se quieren mostrar. Para ello se selecciona "Add Query". Esto lleva a la ventana de configuración del gráfico. Desde aquí, se puede indicar que datos mostrar a través del editor de consultas, o realizar cambios en la configuración del gráfico (título, valores de los ejes, alertas …). En función del elemento que se quiera configurar, se debe seleccionar una de las pestañas señaladas en la Ilustración 19:

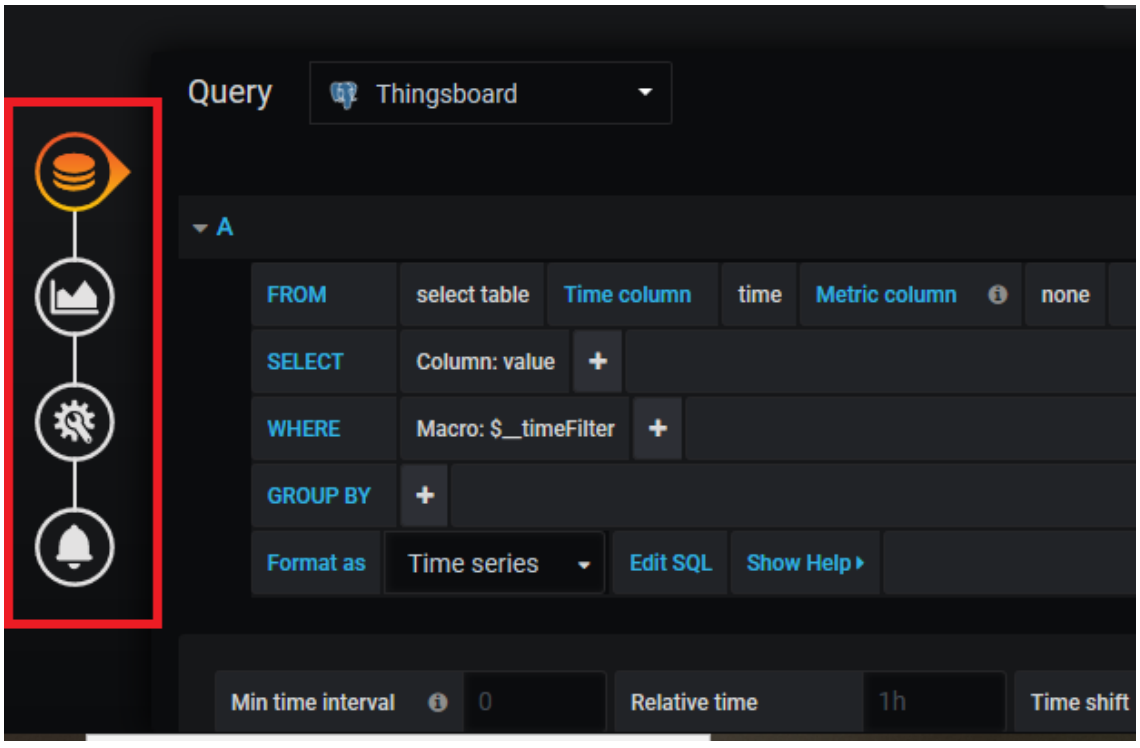

Ilustración 20: Configuración panel Grafana

Para crear un gráfico donde se muestren las temperaturas leídas por un sensor se debe realizar la siguiente consulta:

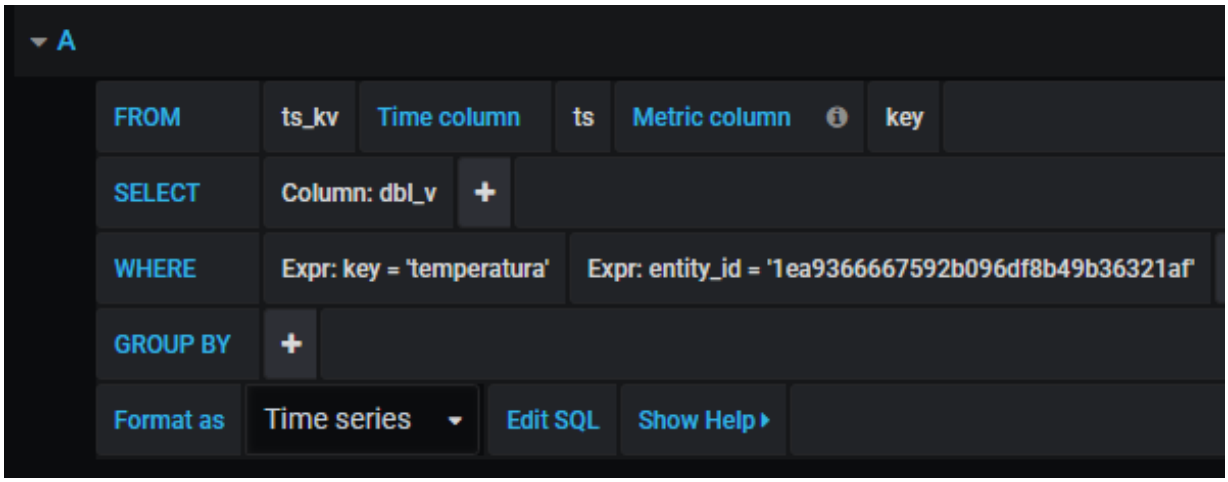

Ilustración 21: Ejemplo consulta Grafana

De esta forma se consigue que se seleccionen todos los datos de temperatura almacenados en la base de datos para el sensor con id=1ea9366667592b096df8b49b36321af. Al indicar que "ts" es la "Time column", Grafana usa el timestamp proporcionado junto a la lectura de temperatura para establecer la hora a la que se realizó esa lectura. El resultado final es el siguiente:

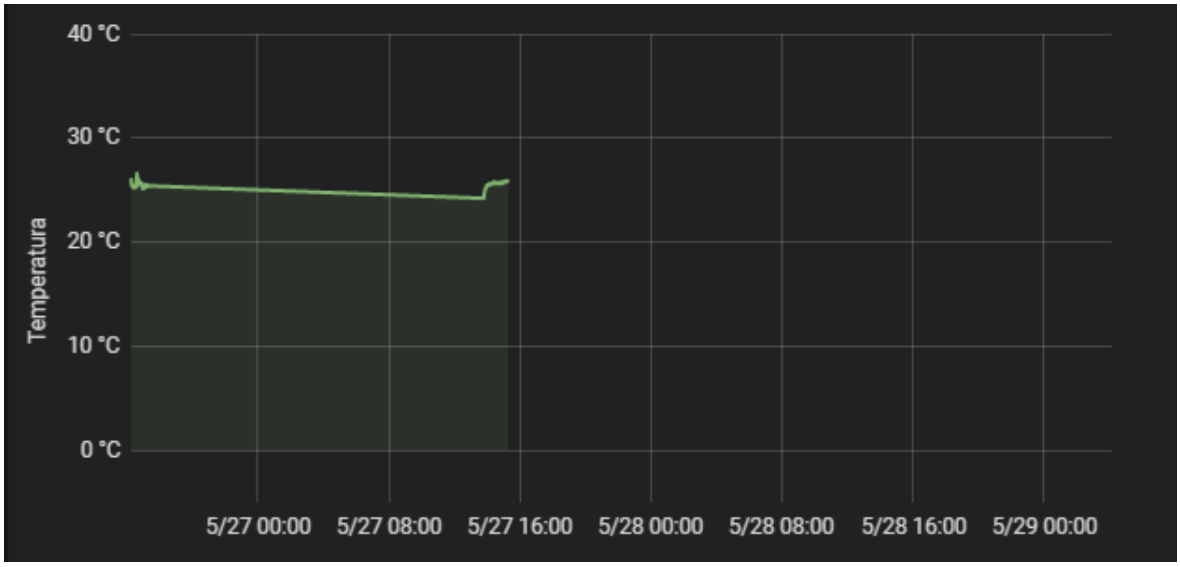

Ilustración 22: Gráfico resultante

Exportar uno de los gráficos generados con Grafana a otras webs es sencillo. Para realizarlo, se elige la opción "Share" del gráfico.

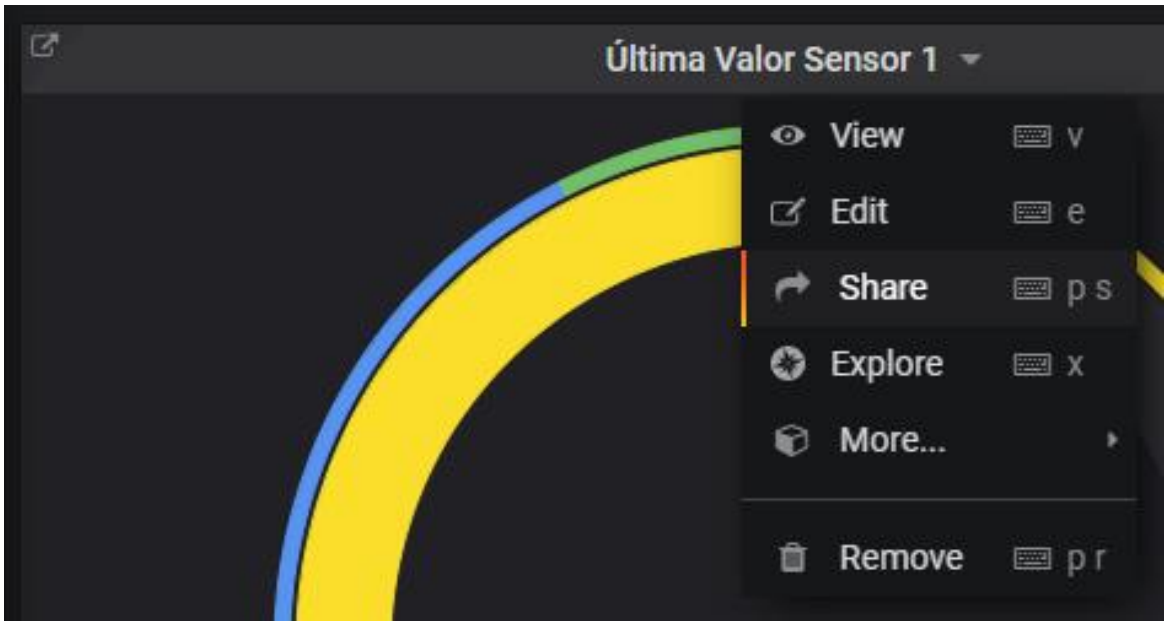

Ilustración 23: Opción share panel Grafana

Existen 3 posibilidades, “Link”, “Snapshot” y “Embeded”. Para importar un gráfico a otra página web, se debe seleccionar la última opción “Embeded”, la cual proporciona el código html que se debe incluir para importar el gráfico.

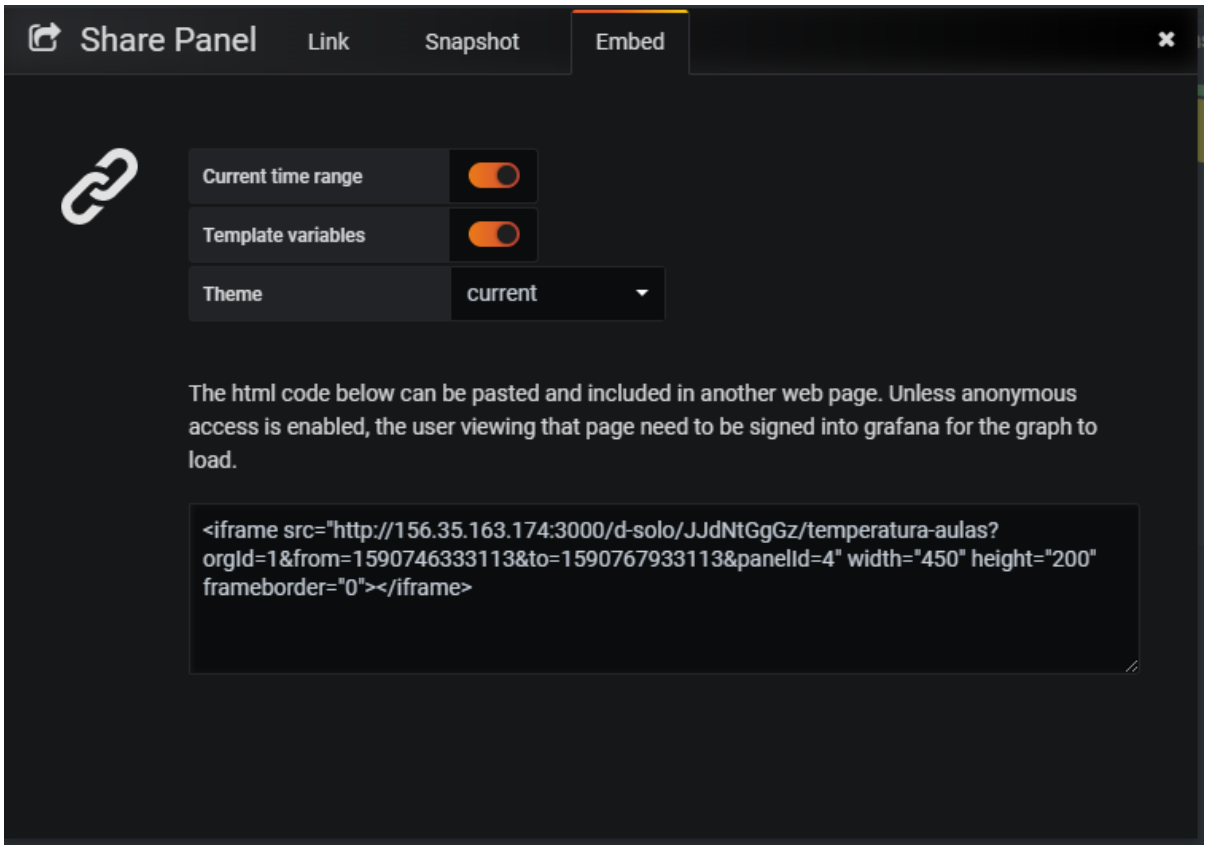

Ilustración 24: Opciones importar gráfico Grafana

Por defecto, Grafana requiere autenticación. Por tanto, si se intenta visualizar un gráfico, ya sea accediendo directamente a Grafana o a través de otra web, es necesario introducir las credenciales de un usuario que esté autorizado para ver esos gráficos. Sim embargo, es posible configurar Grafana para que puedan mostrarse gráficos sin necesidad de introducir credenciales. Para realizar esto se accede al archivo de configuración "grafana.ini". En la siguiente ilustración se muestra una parte del fichero:

```
#################### Grafana Configuration Example #####################
#
# Everything has defaults so you only need to uncomment things you want to
# change

# possible values : production, development
;app_mode = production

# instance name, defaults to HOSTNAME environment variable value or hostname if HOSTNAME var is empty
;instance_name = ${HOSTNAME}

################################### Paths ####################################
[paths]
# Path to where grafana can store temp files, sessions, and the sqlite3 db (if that is used)
;data = /var/lib/grafana

# Temporary files in `data` directory older than given duration will be removed
;temp_data_lifetime = 24h

# Directory where grafana can store logs
;logs = /var/log/grafana

# Directory where grafana will automatically scan and look for plugins
;plugins = /var/lib/grafana/plugins

# folder that contains provisioning config files that grafana will apply on startup and while running.
;provisioning = conf/provisioning

################################### Server ####################################
```

Ilustración 25: Archivo grafana.ini

En concreto, es necesario modificar la variable "allow_embedding" (Ilustración 26) y permitir la autorización anónima (Ilustración 27).

```
# set to true if you want to
allow_embedding = true
```

Ilustración 26: Parámetro allow_embedding

```
#################################
[auth.anonymous]
# enable anonymous access
enabled = true
```

Ilustración 27: Permitir modo anónimo

3.10 NGINX

"Nginx es un servidor HTTP y de proxy inverso, un servidor proxy de correo y un servidor proxy TCP / UDP genérico."[26] Nginx apoyó el 25.54% de los sitios más ocupados en abril de 2020 . Algunas de las compañías de éxito que utilizan Nginx son Dropbox , Netflix o Wordpress.com.

En este proyecto se utiliza con un servidor web HTTP. Se crea una página web a donde se exportan los gráficos creados en Grafana.

Para su instalación se llevaron a cabo los siguientes pasos:

Paso 1: Instalación Nginx

Descarga e instalación de Nginx.

```
sudo apt install nginx
```

Paso 2: Configuración Firewall

Permitir tráfico por el puerto 80

```
sudo ufw allow 'Nginx HTTP'
```

Paso 3: Comprobar funcionamiento

Verificar que el servicio se está ejecutando con el siguiente comando:

```
systemctl status nginx
```

La salida de este comando es la siguiente:

```
nico@monitortemp:~$ systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2020-05-29 17:23:18 CEST; 9min ago
     Docs: man:nginx(8)
  Process: 103143 ExecStop=/sbin/start-stop-daemon --quiet --stop --retry QUIT/5 --pidfile /run/nginx.pid (c
  Process: 103158 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
  Process: 103144 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0
 Main PID: 103160 (nginx)
    Tasks: 3 (limit: 4623)
   CGroup: /system.slice/nginx.service
           ├─103160 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
           ├─103164 nginx: worker process
           └─103165 nginx: worker process

may 29 17:23:17 monitortemp.edv.uniovi.es systemd[1]: Starting A high performance web server and a reverse p
may 29 17:23:18 monitortemp.edv.uniovi.es systemd[1]: nginx.service: Failed to parse PID from file /run/ngin
may 29 17:23:18 monitortemp.edv.uniovi.es systemd[1]: Started A high performance web server and a reverse pr
nico@monitortemp:~$
```

Ilustración 28: Salida comando systemctl status nginx

[26] (Nginx)

Comprobar que es accesible desde un navegador web:

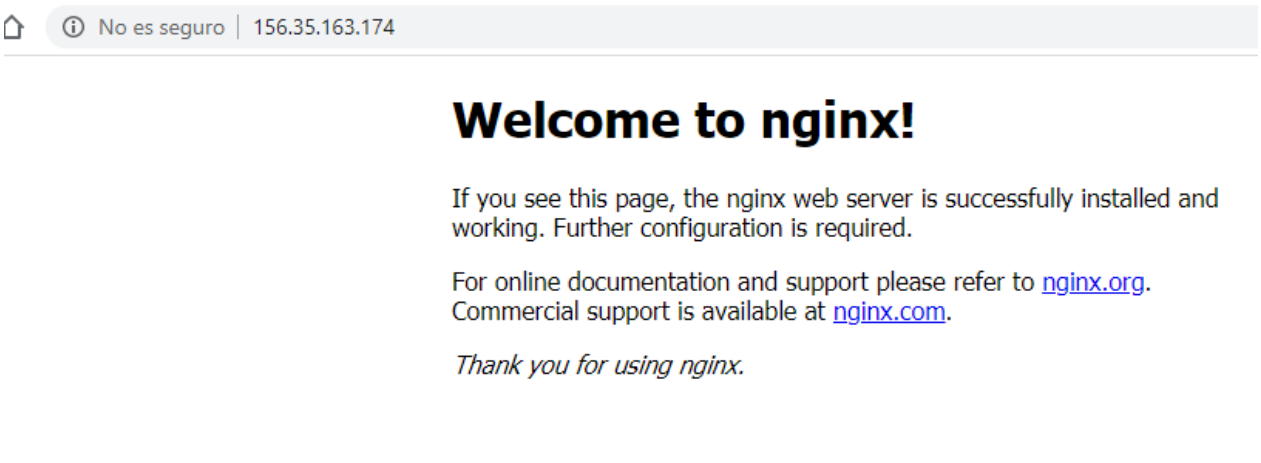

Ilustración 29: Página web por defecto

Paso 5: Configurar bloques del servidor

Crear el directorio "temperatura_aulas.com" como se indica a continuación, usando el indicador -p para crear cualquier directorio matriz que pueda requerirse:

```
sudo mkdir -p /var/www/temperatura_aulas.com/html
```

Posteriormente, asignar la titularidad del directorio con la variable de entorno $USER:

```
sudo chown -R $USER:$USER /var/www/temperatura_aulas.com /html
```

Luego crear una página index.html como ejemplo:

```
nano /var/www/temperatura_aulas.com/html/index.html
```

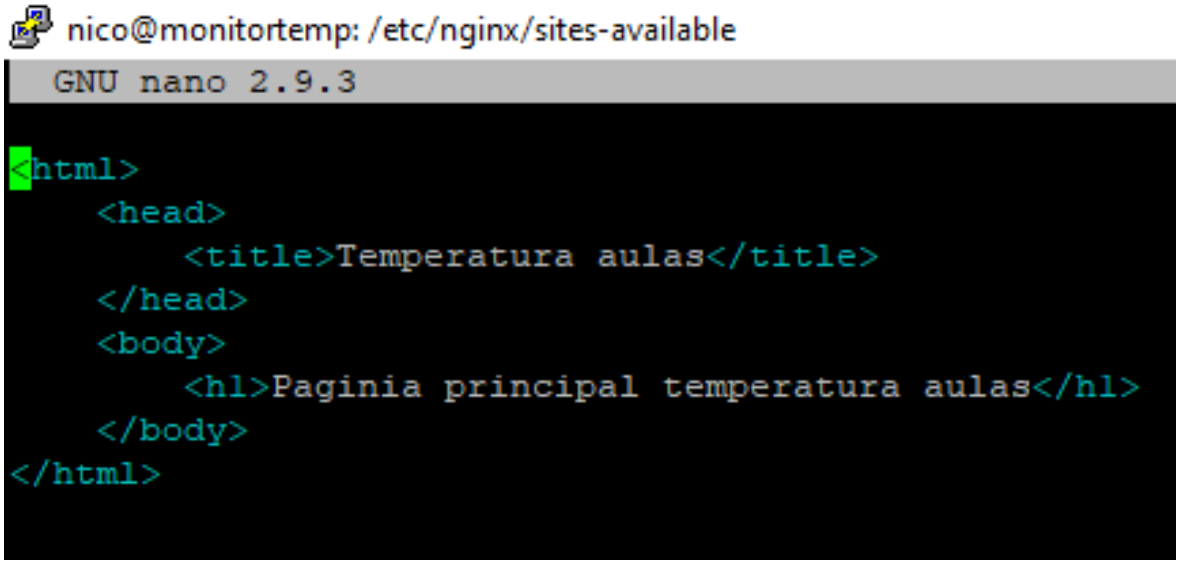

Ilustración 30: Contenido index.html

En lugar de modificar el archivo de configuración por defecto, "/etc/nginx/sites-available/default", se crea uno nuevo con el comando:

`sudo nano /etc/nginx/sites-available/temperatura_aulas.com`

Se realiza una configuración básica, utilizando las siguientes directivas:

listen: se indica que el servidor debe escuchar a peticiones que se realicen a la IP 156.35.163.174 y en el puerto 80.

root: se especifica la ruta en la que el servidor debe buscar los archivos que va a servir

index: se indica el nombre del archivo que queremos que sirva como índice.

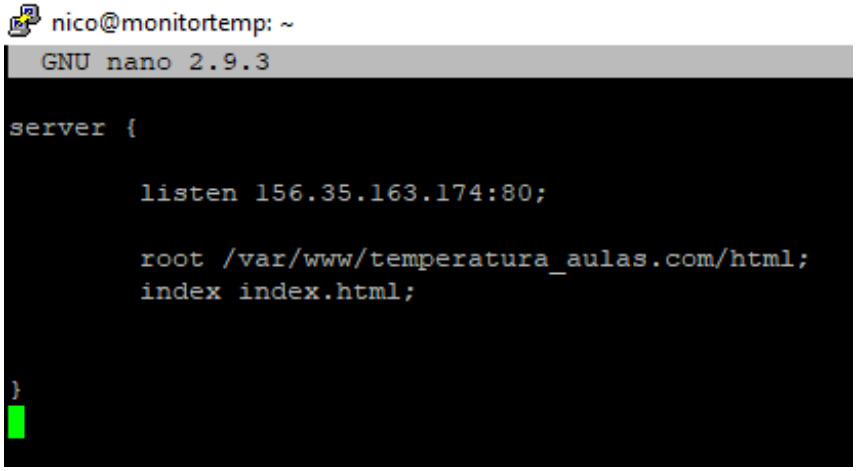

Ilustración 31: Contenido temperatura_aulas.com

Después, se habilita el archivo creando un enlace desde el mismo al directorio "sites-enabled" el cual Nginx usa para leer durante el inicio:

`sudo ln -s /etc/nginx/sites-available/temperature_aulas.com /etc/nginx/sites-enabled/`

Se reinicia Nginx para que se actualice la configuración con el siguiente comando:

`sudo systemctl restart nginx`

Prabar si es accesible desde navegador y que sirve el archivo index.html

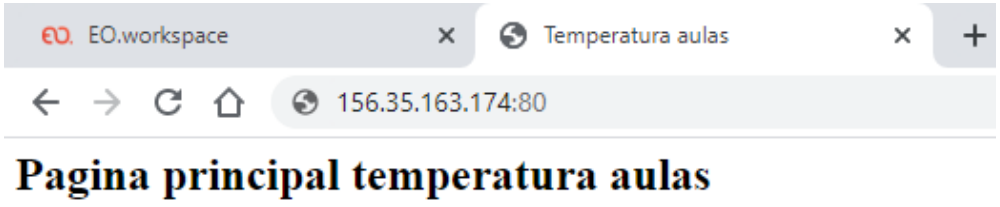

Ilustración 32: Resultado configuración de ejemplo

4 Metodología de trabajo

Con el fin de satisfacer los objetivos descritos previamente, el proyecto ha ido pasando por distintas fases. Estas fases pueden agruparse en tres partes diferentes que corresponden a:

- Configuración módulo multisensor: modificaciones en el firmware para conseguir los datos necesarios reduciendo el consumo de batería del sensor.
- Comunicación entre dispositivos en el borde y gateway: desarrollo de scripts que permiten obtener los datos leídos por el sensor de temperatura para realizar con ellos las acciones pertinentes.
- Configuración de la nube: en esta parte se llevan a cabo todos los pasos necesarios para tener operativa la nube, de tal forma que puedan enviarse datos desde el gateway, y estos sean gestionados y almacenados.

Las diferentes fases que se ha ido siguiendo durante el desarrollo del proyecto son las siguientes:

1. Configuración inicial del módulo
2. Configuración del gateway
3. Desarrollo del firmware
4. Actualización del firmware
5. Ejemplo de uso de nube pública
6. Ejemplo de uso de nube privada

Todas estas fases se describen con mayor destalle en el apartado **Desarrollo de la investigación**. También se agrega la planificación temporal, donde se indica el tiempo dedicado a cada tarea.

5 Desarrollo de la investigación

En este apartado se describen cada una de las fases mencionadas en el apartado anterior **Metodología de trabajo**, y que se han ido desarrollando a lo largo del proyecto.

5.1 CONFIGURACIÓN INICIAL DEL MÓDULO

En primer lugar, se procede a comprobar el correcto funcionamiento de los sensores. Para ello se hace uso de la app móvil ST BLE Sensor App[27], proporcionada por el fabricante. La app móvil permite elegir que sensor o sensores se quieren utilizar para recopilar datos, que función aplicar a los datos (media, máximo, mínimo …) y cómo transmitirlos (vía Bluetooth, almacenarlos en la memoria del propio sensor …). También incluye unos ejemplos precargados. En nuestro caso se utiliza el ejemplo precargado "Barometer" que se puede ver en la Ilustración 33. Este mide temperatura, humedad y presión y lo transmite por bluetooth.

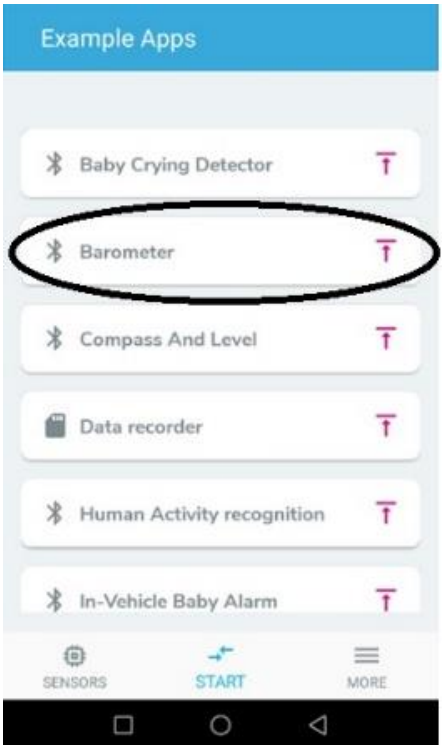

Ilustración 33: ST BLE Sensor APP Example Apps

[27] (ST BLE Sensor App)

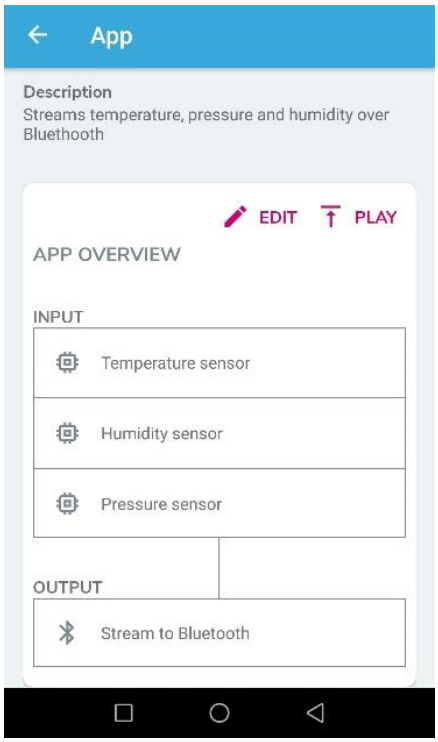

Ilustración 34: ST BLE Sensor APP Barometer

Una vez cargada en el sensor, se establece conexión con el dispositivo, eligiéndolo en la lista de dispositivos. Cuando ambos se encuentran conectados se puede visualizar temperatura, humedad y presión, como se muestra en las siguientes ilustraciones.

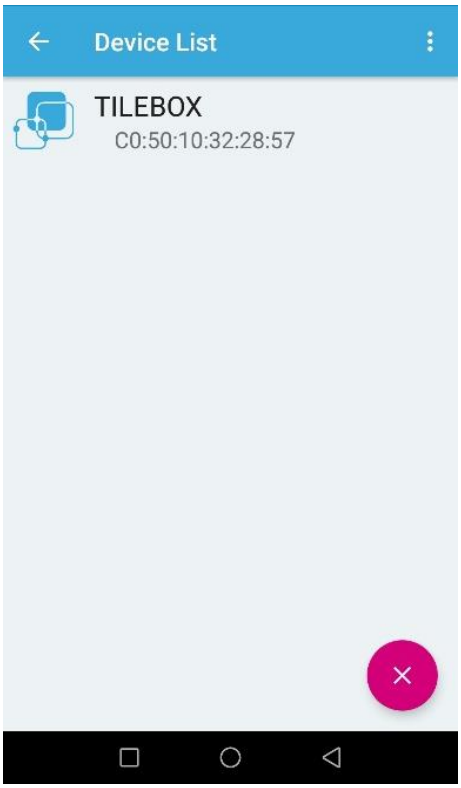

Ilustración 35: ST BLE Sensor APP Search Devices

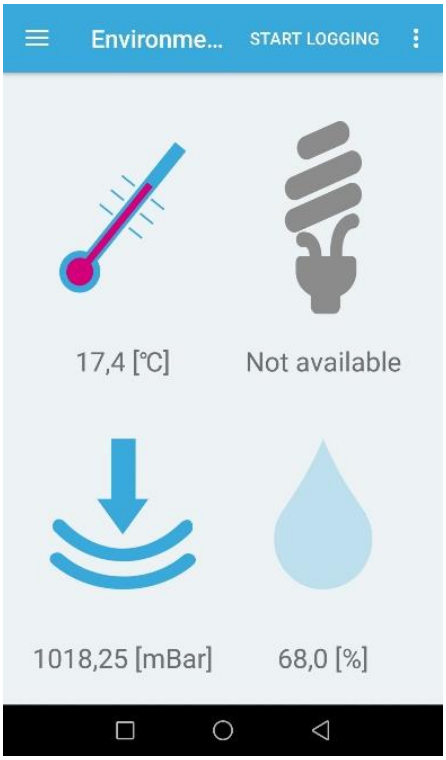

Ilustración 36: ST BLE Sensor APP Barometer Display

5.2 CONFIGURACIÓN DEL GATEWAY

Tras realizar la comprobación del apartado anterior en ambos sensores, se realiza la configuración del dispositivo que actúa como gateway. El primer paso es instalar un sistema operativo. El escogido es el sistema operativo Raspbian. Una vez instalado, se activa el acceso a través de SSH para poder conectarse remotamente desde nuestro PC. También es necesario instalar Bluepy[28], un módulo de Python que permite conectarse a dispositivos BLE a través de la pila de software Bluetooth oficial para Linux, Bluez[29].

A continuación, se realizan 2 comprobaciones. En primer lugar, que la RPi detecta el sensor utilizando el comando "sudo hcitool lescan", que muestra los dispositivos bluetooth detectados. La salida obtenida es la siguiente:

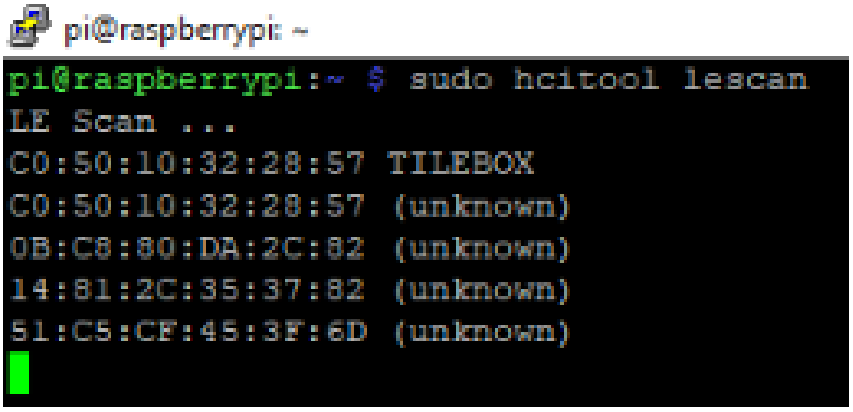

Ilustración 37: Salida comando hcitool lescan

Entre los dispositivos disponibles se encuentra el sensor. Aparece con el nombre "TILEBOX", y vemos que la dirección coincide con la que aparece en la aplicación.

Ahora que se ha comprobado que el sensor es visible para la RPi, se procede a realizar la conexión. Para ello se desarrolla un sencillo script que se conecte al sensor y muestre los UUIDs de los servicios proporcionados por el sensor. La salida al ejecutar este script se muestra en la Ilustración 38.

[28] (Bluepy)
[29] (Bluez)

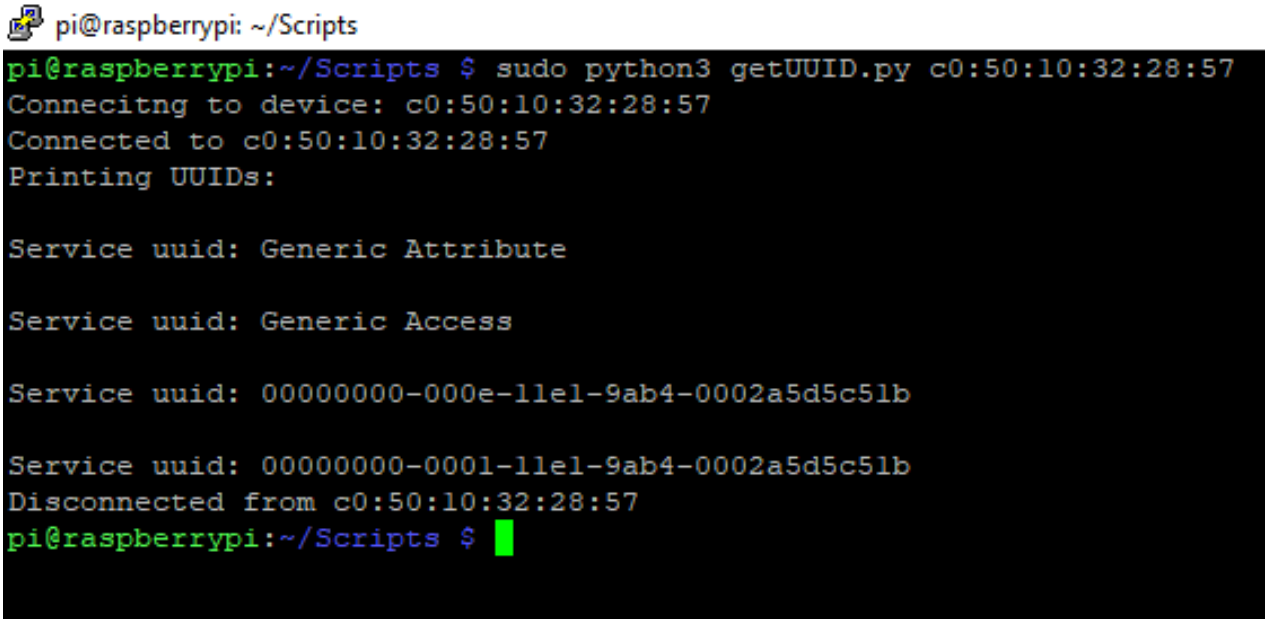

Ilustración 38: Salida script getUUID.py

El sensor es visible y accesible para el gateway, por lo tanto, se puede proceder a obtener la temperatura. Para ello se elabora un script más complejo que debe realizar las siguientes acciones:

1. Conectarse al sensor
2. Activar las notificaciones: escribir
3. Esperar a que llegue una notificación
4. Tratar los datos recibidos.

Todo esto se realiza en el script "temperature.py". El resultado de ejecutar el script es el siguiente:

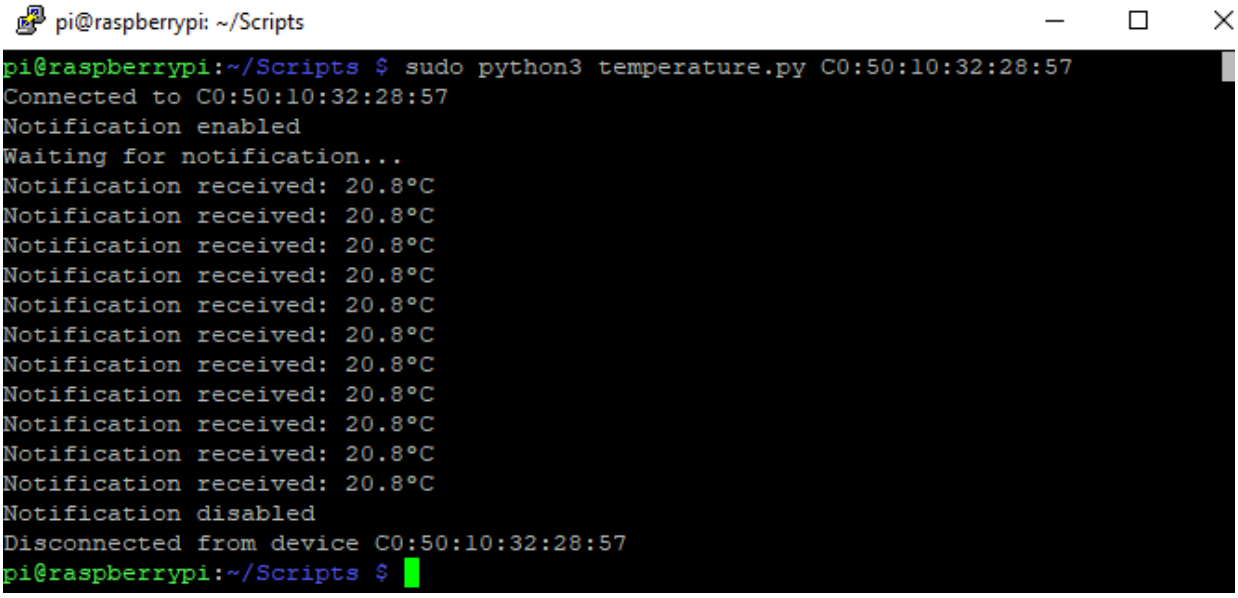

Ilustración 39: Salida script temperature.py

Como en un futuro se trabajará con varios sensores, es necesario que el script sea capaz de conectarse a varios dispositivos de forma simultánea. Por tanto, se realiza una nueva versión, en la que se lanza un hilo por cada nodo sensor. Esto se realiza en el script "temperature_sem.py" y el resultado es el mostrado a continuación:

```
pi@raspberrypi:~/Scripts $ sudo python3 temperature_sem.py C0:50:24:32:0E:57 C0:50:10:32:28:57
Connected to C0:50:24:32:0E:57
Connected to C0:50:10:32:28:57
Notification enabled
Waiting for notification...
Notification enabled
Waiting for notification...
Notification received from C0:50:24:32:0E:57: 19.6°C
Notification received from C0:50:24:32:0E:57: 19.6°C
Notification received from C0:50:10:32:28:57: 20.7°C
Notification received from C0:50:10:32:28:57: 20.7°C
Notification received from C0:50:10:32:28:57: 20.7°C
Notification received from C0:50:24:32:0E:57: 19.6°C
Notification received from C0:50:24:32:0E:57: 19.6°C
Notification received from C0:50:24:32:0E:57: 19.6°C
Notification received from C0:50:10:32:28:57: 20.7°C
Notification received from C0:50:10:32:28:57: 20.7°C
Notification received from C0:50:10:32:28:57: 20.7°C
Notification received from C0:50:24:32:0E:57: 19.6°C
Notification received from C0:50:24:32:0E:57: 19.6°C
Notification received from C0:50:10:32:28:57: 20.7°C
Notification received from C0:50:10:32:28:57: 20.7°C
Notification received from C0:50:10:32:28:57: 20.7°C
Notification received from C0:50:24:32:0E:57: 19.6°C
Notification received from C0:50:24:32:0E:57: 19.6°C
Notification received from C0:50:24:32:0E:57: 19.6°C
Notification disabled
Disconnected from device C0:50:10:32:28:57
Notification received from C0:50:10:32:28:57: 20.8°C
Notification disabled
Disconnected from device C0:50:10:32:28:57
pi@raspberrypi:~/Scripts $
```

Ilustración 40: Salida script tempearature_sem.py

Se ha comprobado que la RPi es capaz de comunicarse con varios sensores simultáneamente e ir mostrando los datos que estos capturan.

5.3 DESARROLLO DEL FIRMWARE

Para este proyecto, es necesario desarrollar un firmware específico para los nodos. Se necesita que estos midan únicamente la temperatura y humedad. Con el firmware que viene instalado por defecto, se están usando todos los sensores, es decir sensores ambientales y sensores de movimiento. Esto provoca un consumo innecesario de energía, que se traduce en un mayor consumo de batería. Además de utilizar todos los sensores, también realiza un número de lecturas muy elevado, una lectura por segundo. Como los sensores irán ubicados en espacios donde la temperatura y la humedad no sufrirán grandes cambios, ni serán muy repentinos, puede aumentarse el tiempo entre cada lectura. Con objeto de reducir aún más el

consumo de energía, el nodo sensor permanecerá en un estado de ahorro de batería mientras no sea necesario hacer una lectura de temperatura.

En lugar de desarrollar un firmware desde 0, se parte de un paquete de funciones proporcionado por STMelectronics llamado FP-SNS-STBOX1[30]sobre el que se realiza alguna modificación.

FP-SNS-STBOX1 es un paquete de funciones que dota a los nodos de conectividad BLE para intercambiar datos ambientales y de sensores de movimiento en tiempo real con otros dispositivos. Tiene implementadas funciones para un consumo de batería ultra bajo.

Dentro del paquete de funciones se incluyen los siguientes proyectos de ejemplo:

- **BLEFOTA:** ejemplo para realizar actualización FOTA(Firmware Over The Air) a través de BLE segura.
- **BLEDualProgram:** ejemplo para realizar actualizaciones a través de BLE seguras usando funciones "dual bank flash features", que permite ejecutar código en un banco de memoria mientras que otro se está programando o borrando su contenido.
- **BLELowPowerRToS:** ejemplo para realizar transmisiones de datos de bajo consumo vía BLE.
- **BLEMLC:** ejemplo para programar el core de machine learning.
- **BLESensors:** ejemplo simple para realizar transmisiones de datos vía BLE.
- **SDDataLogRToS:** ejemplo para almacenar datos en la tarjeta SD.
- **BootLoader:** BootLoader para activar la actualización de Código desde memoria flash o desde la tarjeta SD.
- **DataLogExtended:** ejemplo para enviar datos vía USB.

De los ejemplos anteriores, se ha escogido "BLELowPowerRTOS" como base para desarrollar un firmware personalizado.

En primer lugar, se importa el proyecto al workspace. Cada ejemplo contiene 3 proyectos para 3 IDEs diferentes. Por lo tanto, se debe elegir el que corresponda con el IDE que se esté utilizando, STM32 Cube IDE en este caso. Para realizar la importación se debe realizar la siguiente sucesión de acciones:

En el menú "File", elegir la opción "Import".

[30] FP-SNS-STBOX1

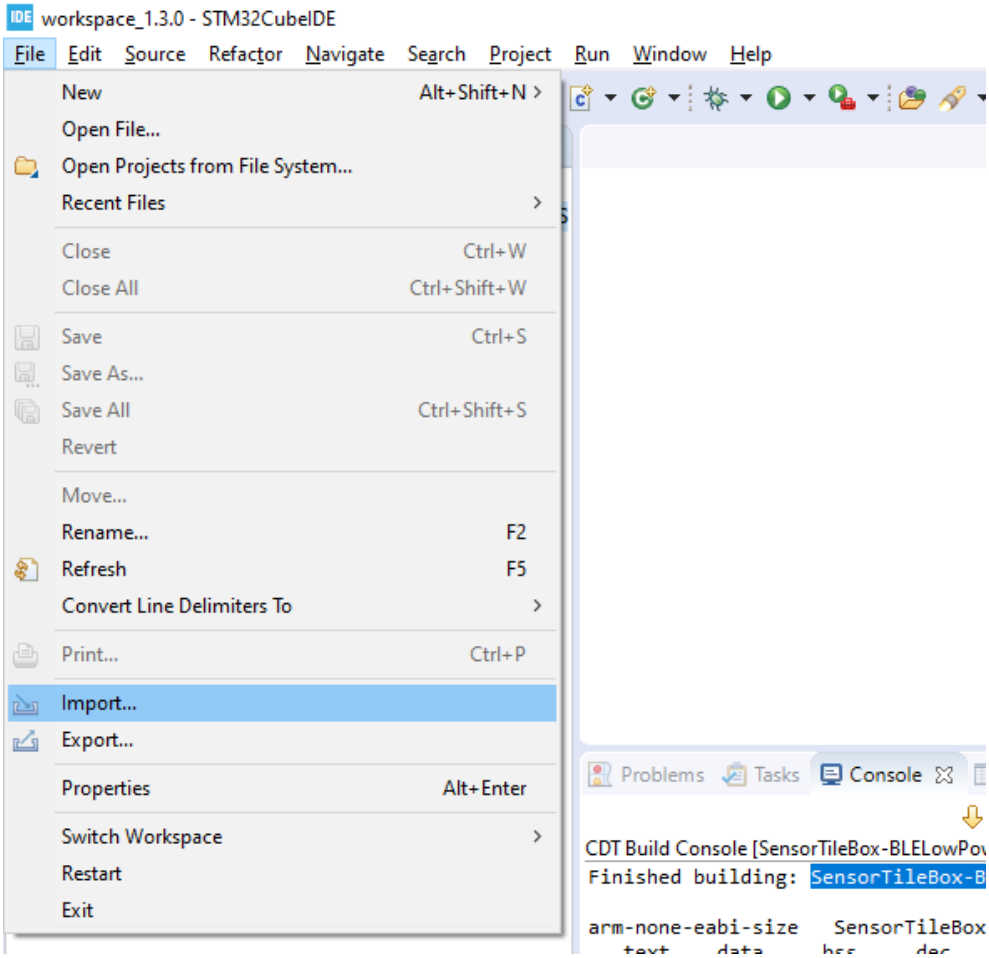

Ilustración 41: STM32 Cube IDE-opción Import

En la ventana emergente, desplegar el menú “General” y seleccionar la opción ”Existing projects into workspace”.

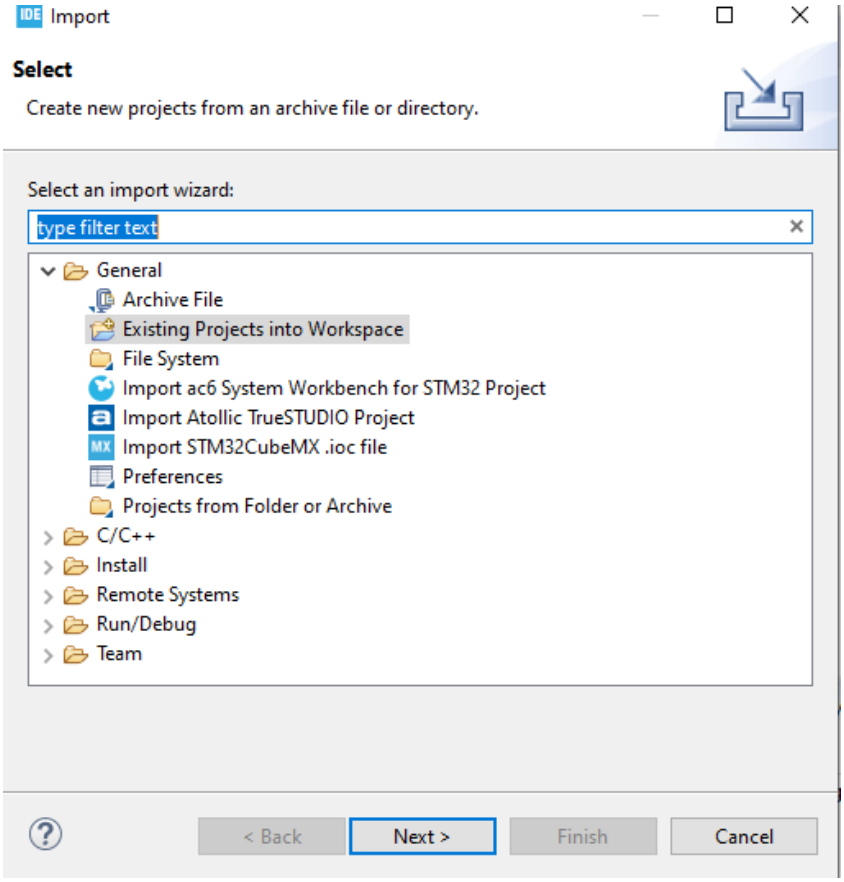

Ilustración 42: STM32 Cube IDE-opción Existing projects into workspace

En la nueva ventana seleccionar la opción “Select root directory” e indicar la siguiente ruta (suponiendo que hemos descomprimido en el escritorio el archivo descargado de la web de STMicroelectronics):

C:\ Desktop\Firmwares\en.FP-SNS-STBOX1_firmware\STM32CubeFunctionPack_STBOX1_V1.3.0\Projects\STM32L4R9ZI-SensorTile.box\Applications\BLELowPowerRToS\STM32CubeIDE\BLELowPowerRToS

Por último, pulsar el botón “Finish”.

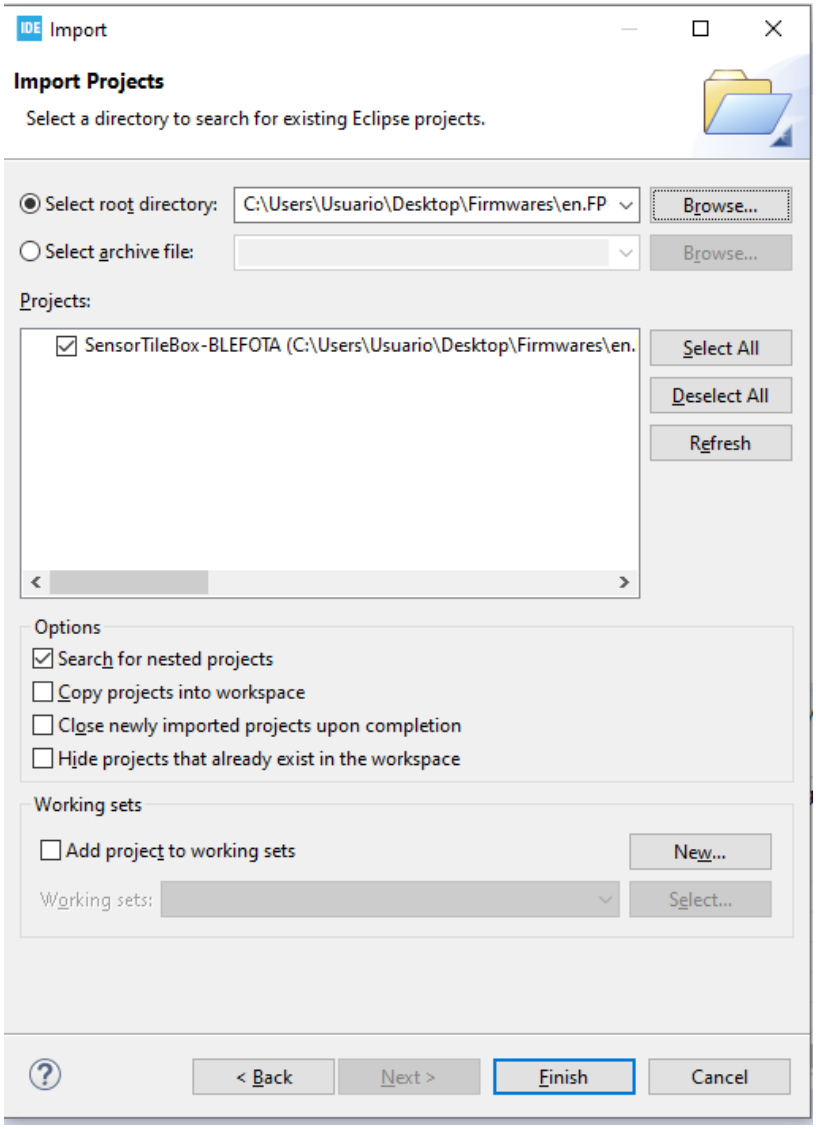

Ilustración 43: STM32 Cube IDE-ventana para importar proyectos

Una vez importado al workspace, es necesario realizar modificaciones en el archivo "main.c". Este archivo contiene la función principal que realiza las inicializaciones necesarias (variables, timers …) y crea y lanza 2 hilos. Uno de ellos ejecuta la función HostThread y el otro la función ProcessThread.

La función ProcessThread es la encargada de comunicar a la función HostThread cuando se produce una nueva lectura, para que actualice el valor de la característica que expone los datos recogidos por los sensores.

Cada tipo de sensor tiene asociado un timer. Cuando un timer se agota, lanza una interrupción, que es capturada por el hilo que ejecuta la función ProcessThread. En función del tipo de interrupción, la función ProcessThread llama al manejador correspondiente. El manejador, se encarga de obtener el dato, y comunicarlo al hilo HostThread.

Es necesario realizar los siguientes cambios para conseguir que el nodo sensor tenga un menor consumo de energía:

1. Eliminar los timers que produzcan interrupciones para realizar medidas con sensores de movimiento.
2. Modificar las funciones de lectura de temperatura y humedad. Los datos de temperatura y humedad se recogen junto a los de presión. Por lo tanto, es necesario modificar las funciones que realizan esta tarea para que únicamente se lea la temperatura y la humedad.
3. Aumentar el tiempo entre cada lectura.

En la Ilustración 44 se muestran todos los timers que se definen.

```
164 /* Timers */
165 osTimerId timLedId,timEnvId,timMotionId;
166 osTimerId timAudioLevId;
167 osTimerId timBatId;
168
169 osTimerDef (TimerLedHandle , LedBlinkCb);
170 osTimerDef (TimerEnvHandle , OsTimerCallback);
171 osTimerDef (TimerBatHandle , OsTimerCallback);
172 osTimerDef (TimerMotionHandle, OsTimerCallback);
173 osTimerDef (TimerAudioLevHandle, OsTimerCallback);
174
```

Ilustración 44: Timers definidos

Como solo son necesarias lecturas del sensor ambiental, se eliminan los timers innecesarios. En concreto, solo serán necesarios:

- **timLedId:** timer para el control de la luz led del sensor.
- **timEnvId:** timer para el control de sensores ambientales.
- **timBatId:** timer para el control de datos de batería.

Una vez eliminados, se tienen definidos los siguientes timers:

```
164 /* Timers */
165 osTimerId timLedId,timEnvId;
166 osTimerId timBatId;
167
168 osTimerDef (TimerLedHandle , LedBlinkCb);
169 osTimerDef (TimerEnvHandle , OsTimerCallback);
170 osTimerDef (TimerBatHandle , OsTimerCallback);
171
```

Ilustración 45: Timers definidos después de la modificación

Por otro lado, es necesario modificar las funciones que son llamadas cuando el timer timEnvId produce una interrupción. En lugar de leer presión, humedad y temperatura, solo se debe obtener la temperatura y la humedad. Para ello, se modifica la función SendEviromentalData(), la cual se muestra en la siguiente ilustración:

```
822 /**
823   * @brief  Send Environmental Data (Temperature/Pressure/Humidity) to BLE
824   * @param  None
825   * @retval None
826   */
827 static void SendEnvironmentalData(void)
828 {
829   msgData_t msg;
830
831   /* Pressure,Humidity, and Temperatures*/
832   if(W2ST_CHECK_CONNECTION(W2ST_CONNECT_ENV)) {
833     int32_t PressToSend;
834     uint16_t HumToSend;
835     int16_t TempToSend;
836
837     /* Read all the Environmental Sensors */
838     ReadEnvironmentalData(&PressToSend,&HumToSend, &TempToSend);
839
840     msg.type      = ENV;
841     msg.env.press = PressToSend;
842     msg.env.hum   = HumToSend;
843     msg.env.temp  = TempToSend;
844     SendMsgToHost(&msg);
845   }
846 }
847
```

Ilustración 46: Función SendEnviromentalData sin modificar

Se sustituye la función ReadEnviromentalData(&PressToSend, &HumToSend, &TempToSend) por la función ReadTemperatureData(&HumToSend, &TempToSend). Como no hay datos de presión y humedad, se eliminan las variables donde se almacenan estos datos. La función resultante es la siguiente:

```
828 /**
829   * @brief  Send Environmental Data (Temperature/Humidity) to BLE
830   * @param  None
831   * @retval None
832   */
833 static void SendEnvironmentalData(void)
834 {
835   msgData_t msg;
836
837   /* Pressure,Humidity, and Temperatures*/
838   if(W2ST_CHECK_CONNECTION(W2ST_CONNECT_ENV)) {
839
840     int16_t TempToSend;
841     uint16_t HumToSend;
842
843     /* Read the Environmental Sensors */
844     ReadTemperatureData(&HumToSend,&TempToSend);
845
846     msg.type      = ENV;
847     msg.env.temp  = TempToSend;
848     msg.env.hum   = HumToSend;
849     SendMsgToHost(&msg);
850   }
851 }
```

Ilustración 47: Función SendEnviromentalData modificada

```
/**
  * @brief  Read The Environmental Data (Temperature/Humidity)
  * @param  uint16_t *HumToSend  pointer to Humidity Value
  * @param  int16_t *TempToSend pointer to Temperature1 Value
  * @retval None
  */
void ReadTemperatureData(uint16_t *HumToSend,int16_t *TempToSend)
{
  float SensorValue;
  int32_t decPart, intPart;

  *TempToSend=0;
  *HumToSend=0;

  if(TargetBoardFeatures.HandleHumSensor != STBOX1_SNS_NOT_VALID) {
    /*Read Humidity and Temperature */
    BSP_ENV_SENSOR_GetValue(TargetBoardFeatures.HandleHumSensor,ENV_HUMIDITY,&SensorValue);
    MCR_BLUEMS_F2I_1D(SensorValue, intPart, decPart);
    *HumToSend = intPart*10+decPart;

#ifdef ONE_SHOT
    BSP_ENV_SENSOR_Set_One_Shot(TargetBoardFeatures.HandleHumSensor);
#endif
  }

  if(TargetBoardFeatures.HandleTempSensor != STBOX1_SNS_NOT_VALID) {
      BSP_ENV_SENSOR_GetValue(TargetBoardFeatures.HandleTempSensor,ENV_TEMPERATURE,&SensorValue);
      MCR_BLUEMS_F2I_1D(SensorValue, intPart, decPart);
      *TempToSend = intPart*10+decPart;

#ifdef ONE_SHOT
    BSP_ENV_SENSOR_Set_One_Shot(TargetBoardFeatures.HandleTempSensor);
#endif
  }
}

```

Ilustración 48: Función ReadTemperatureData

Al modificar esta función, también es necesario modificar la función Enviromental_Update(int32_t Press,uint16_t Hum,int16_t Temp), que se muestra en la Ilustración 49, que es llamada en el hilo HostThread cuando se quiere actualizar el valor de la característica que expone el valor de la temperatura y la humedad.

```
573 tBleStatus Environmental_Update(int32_t Press,uint16_t Hum,int16_t Temp)
574 {
575   tBleStatus ret=BLE_STATUS_SUCCESS;
576   uint8_t BuffPos;
577
578   uint8_t buff[2+4/*Press*/+2/*Hum*/+2/*Temp*/];
579
580   STORE_LE_16(buff  ,(HAL_GetTick()>>3));
581   BuffPos=2;
582
583   if (TargetBoardFeatures.HandlePressSensor != STBOX1_SNS_NOT_VALID) {
584     STORE_LE_32(buff+BuffPos,Press);
585     BuffPos+=4;
586   }
587
588   if (TargetBoardFeatures.HandleHumSensor != STBOX1_SNS_NOT_VALID) {
589     STORE_LE_16(buff+BuffPos,Hum);
590     BuffPos+=2;
591   }
592
593   if (TargetBoardFeatures.HandleTempSensor != STBOX1_SNS_NOT_VALID) {
594     STORE_LE_16(buff+BuffPos,Temp);
595     BuffPos+=2;
596   }
597
598   ret = ACI_GATT_UPDATE_CHAR_VALUE(HWServW2STHandle, EnvironmentalCharHandle, 0, EnvironmentalCharSize,buff);
599
600   if (ret != BLE_STATUS_SUCCESS){
601     if(W2ST_CHECK_CONNECTION(W2ST_CONNECT_STD_ERR)){
602       BytesToWrite =sprintf((char *)BufferToWrite, "Error Updating Environmental Char\n");
603       Stderr_Update(BufferToWrite,BytesToWrite);
604     } else {
605       STBOX1_PRINTF("Error Updating Environmental Char\r\n");
606     }
607   }
608   return ret;
```

Ilustración 49: Función Enviromental_Update

En lugar de modificar esta función, se crea una nueva función Temperatura_Update(uint16_t Hum, int16_t Temp). El código de esta nueva función es el siguiente:

```
617 tBleStatus Temperature_Update(uint16_t Hum, int16_t Temp)
618 {
619     tBleStatus ret=BLE_STATUS_SUCCESS;
620       uint8_t BuffPos;
621 
622       uint8_t buff[2+4+2/*Hum*/+2/*Temp*/];
623 
624       STORE_LE_16(buff  ,(HAL_GetTick()>>3));
625       BuffPos=2;
626 
627       if (TargetBoardFeatures.HandleHumSensor != STBOX1_SNS_NOT_VALID) {
628         STORE_LE_16(buff+BuffPos,Hum);
629         BuffPos+=2;
630       }
631 
632       if (TargetBoardFeatures.HandleTempSensor != STBOX1_SNS_NOT_VALID) {
633         STORE_LE_16(buff+BuffPos,Temp);
634         BuffPos+=2;
635       }
636 
637       ret = ACI_GATT_UPDATE_CHAR_VALUE(HWServW2STHandle, EnvironmentalCharHandle, 0, EnvironmentalCharSize,buff);
638 
639       if (ret != BLE_STATUS_SUCCESS){
640         if(W2ST_CHECK_CONNECTION(W2ST_CONNECT_STD_ERR)){
641           BytesToWrite =sprintf((char *)BufferToWrite, "Error Updating Environmental Char\n");
642           Stderr_Update(BufferToWrite,BytesToWrite);
643         } else {
644           STBOX1_PRINTF("Error Updating Environmental Char\r\n");
645         }
646       }
647       return ret;
648     }
```

Ilustración 50: Función Temperature_Update

La siguiente tarea es modificar el tiempo entre cada lectura. Para ello se emplea la función HAL_Delay. Esta función hace uso de la instrucción WFI(), que suspende la ejecución y provoca la entrada en el modo de ultra bajo consumo “sleep-mode”. El código de esta función se muestra en la siguiente ilustración:

```
1206 /**
1207   * @brief This function provides accurate delay (in milliseconds) based
1208   *        on variable incremented.
1209   * @note This is a user implementation using WFI state
1210   * @param Delay: specifies the delay time length, in milliseconds.
1211   * @retval None
1212   */
1213 void HAL_Delay(__IO uint32_t Delay)
1214 {
1215   uint32_t tickstart = 0;
1216   tickstart = HAL_GetTick();
1217   while((HAL_GetTick() - tickstart) < Delay){
1218     __WFI();
1219   }
1220 }
```

Ilustración 51: Función HAL_Delay

Se puede controlar el tiempo entre cada lectura si se invoca la función HAL_Delay() cada vez que se realiza una lectura de datos. También se consigue que el nodo multisensor consuma la menor cantidad de batería posible.

Se realizan lecturas de temperatura cada 6 minutos, por tanto, se llama a la función HAL_Delay() y se le pasan como parámetro esos 6 minutos, pero expresados en milisegundos, que es la unidad de tiempo con la que trabaja la función. Esta llamada se realiza en el hilo ProccesThread, cada vez el timer se agota y se llama a la función SendEnviromentalData() que desata el proceso de lectura del valor del sensor de temperatura y humedad. Es decir, cuando el timer encargado de controlar las lecturas de temperatura y humedad produce una interrupción, se procede a leer y actualizar el dato de la temperatura y humedad. Una vez hecho esto, el sensor se duerme durante 6 minutos, para luego volver a esperar por el timer, tal como se muestra en la Ilustración 52.

```
508         /* Environmental Data */
509         if(SendEnv) {
510           SendEnv=0;
511           SendEnvironmentalData();
512           //Delay
513           HAL_Delay(360000);
```

Ilustración 52: Uso de la función HAL_Delay en ProcessThread

Por último, se construye el firmware. Para ello, se selecciona la opción “Build all” de la barra de herramientas, señalada con un círculo rojo en la Ilustración 53.

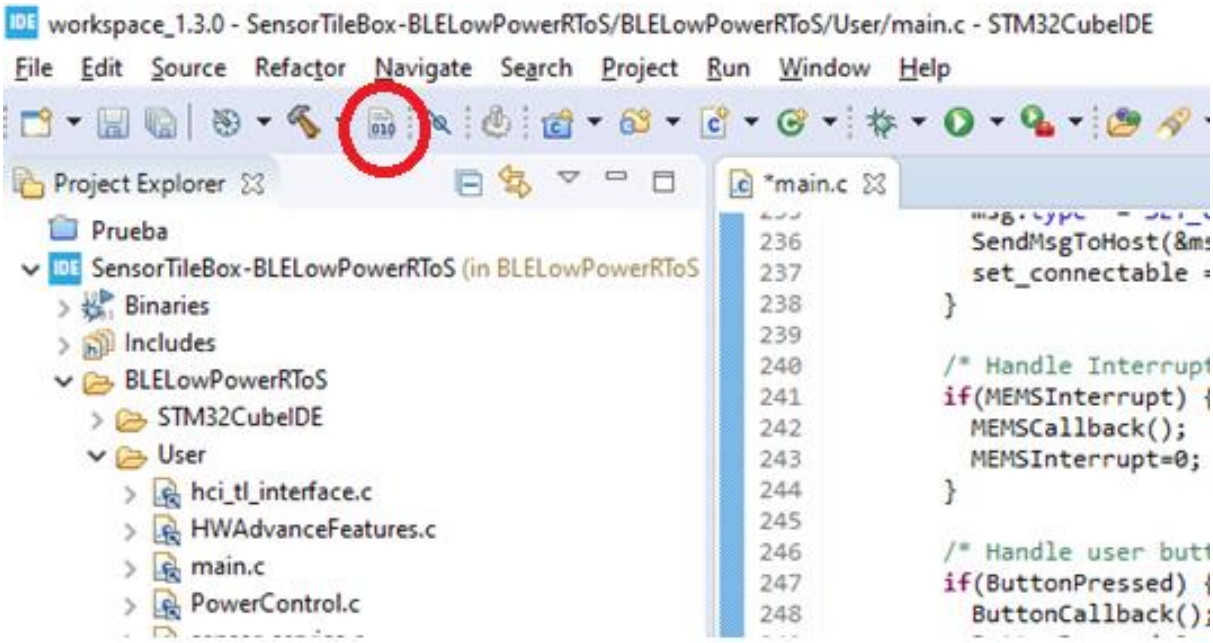

Ilustración 53: Opción Build all

Si la operación se completa con éxito, se genera el archivo *\BLELowPowerRToS\Debug\SensorTileBox-BLELowPowerRToS.bin* que se debe cargar en el nodo.

Este firmware es utilizado para llevar acabo la implantación final. Para el resto de las pruebas y ejemplos desarrollados se trabaja con el firmware instalado por defecto, de modo que puedan obtenerse varias lecturas de datos en pocos segundos, agilizando el desarrollo.

Para cualquier consulta más profunda, se puede importar el proyecto "BLELowPowerRToS" utilizando el IDE STM32 Cube IDE.

5.4 ACTUALIZACIÓN DEL FIRMWARE

Una vez desarrollado el firmware, es necesario reemplazar el firmware que viene instalado por defecto en el sensor. El STEVAL-MKSBOX1V1 permite hacerlo de 2 formas. O bien activando el modo DFU (Direct Firmware Update) y conectándolo al ordenador con un cable USB, o utilizando un debugger.

Para comprobar que ambas opciones son válidas, y para dejar constancia del procedimiento a realizar para actualizar el firmware, se hace de ambas formas.

Para realizar la actualización al nuevo firmware con el modo DFU, es necesario activarlo DFU en el nodo sensor. Para realizar esto, se debe conectar un teléfono móvil con el nodo sensor, como ya se hizo anteriormente. Dentro de la app, se selecciona el sensor sobre el que se quiere realizar la actualización de firmware. Cuando se encuentren conectados, se selecciona la opción de "Debug Console" como se muestra a continuación:

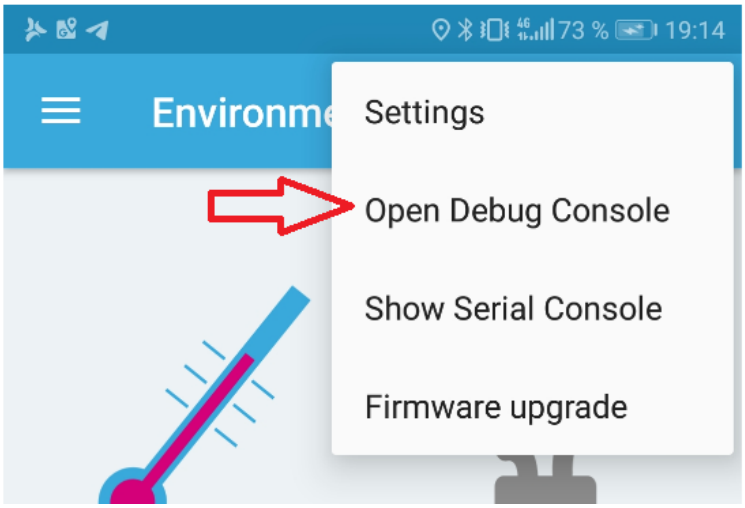

Ilustración 54: Abrir consola

Antes de ejecutar ningún comando es necesario conectar el nodo a nuestro ordenador a través de un cable USB-miniUSB. Cuando ya se encuentren conectados, se ejecuta el comando DFU, como se muestra en la Ilustración 55.

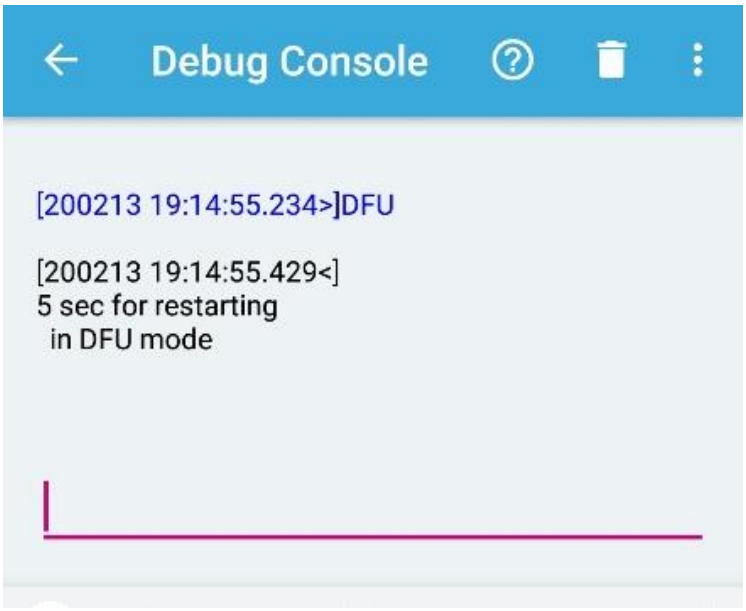

Ilustración 55: Salida consola comando DFU

Otra vía para acceder al modo DFU es presionar los botones PWR y ROOT al mismo tiempo.

A continuación, se abre en un ordenador la herramienta STM32 CubeProgrammer. El primer paso dentro de la herramienta es conectarse al dispositivo. Para ello se selecciona la opción "USB" y en el apartado "Port" la opción "USB1". Por último, seleccionar "Connect". La ubicación de estas opciones en la interfaz gráfica del programa se muestra en la Ilustración 56.

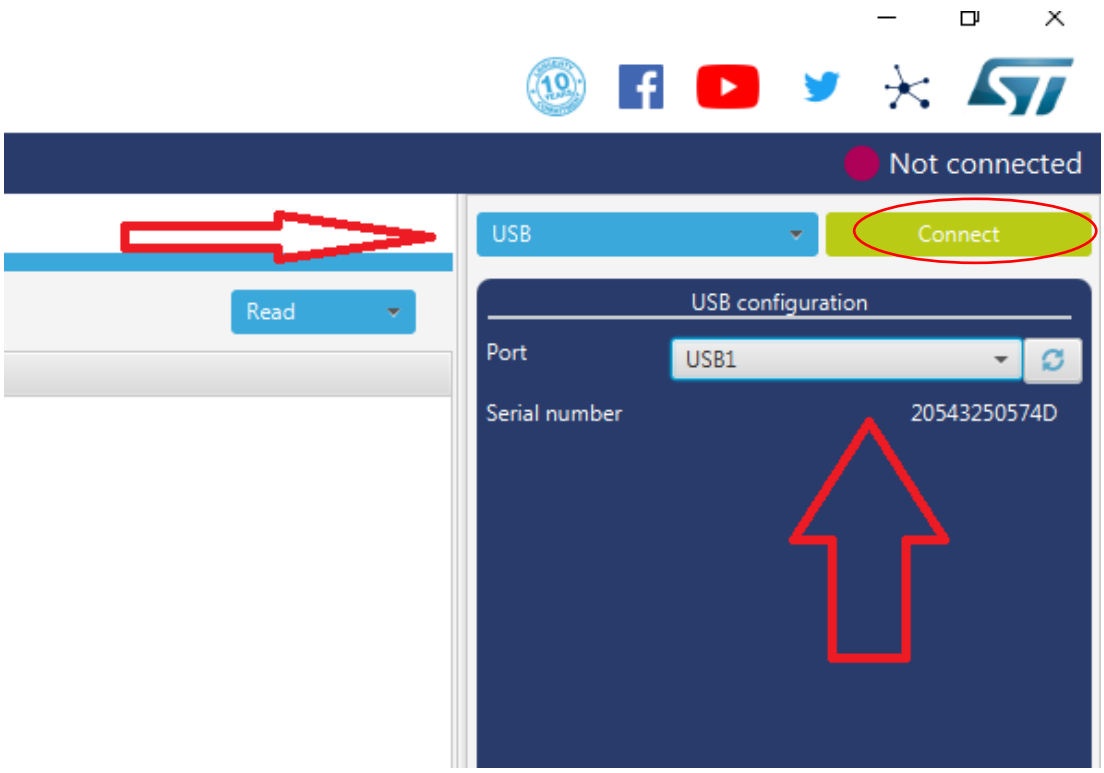

Ilustración 56: STM32CubeProgrammer I

Si se ha conectado correctamente se muestra lo siguiente:

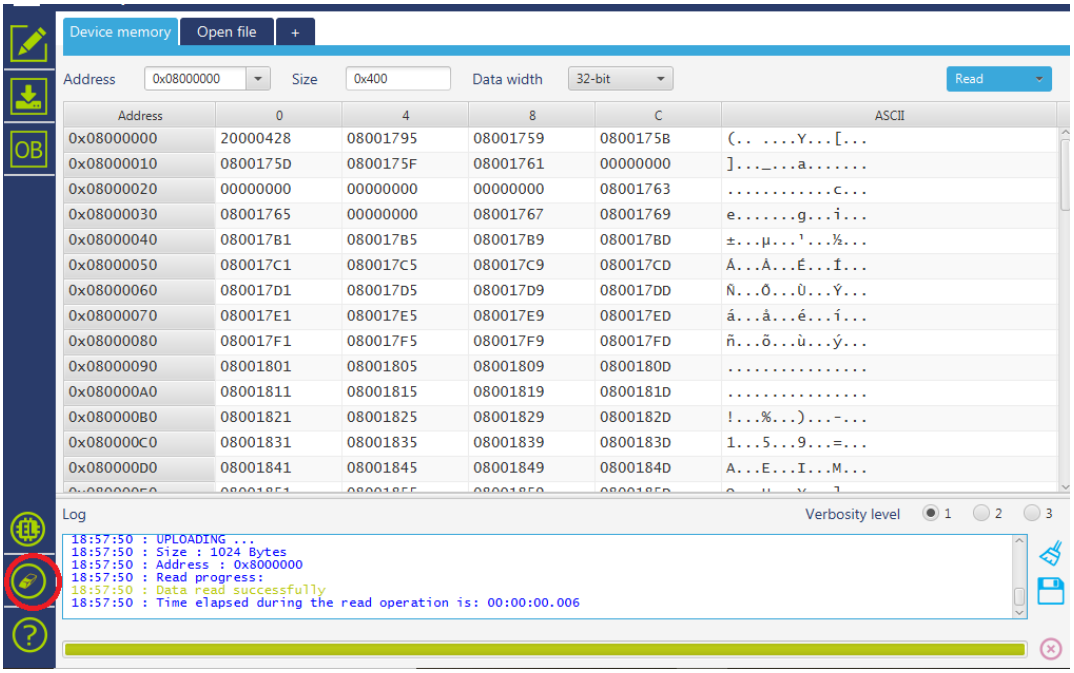

Ilustración 57: STM32CubeProgrammer II

Para realizar la actualización de firmware se realizarán 2 acciones. Primero, eliminar de la placa el firmware actual. Para ello seleccionamos el icono de la goma de borrar (señalado con un círculo rojo en la Ilustración 57). Si el borrado se realizó con éxito aparecerá un cuadro de diálogo indicándolo. Además, habrán cambiado todos los valores de los registros a 0xFFFFFFFF.

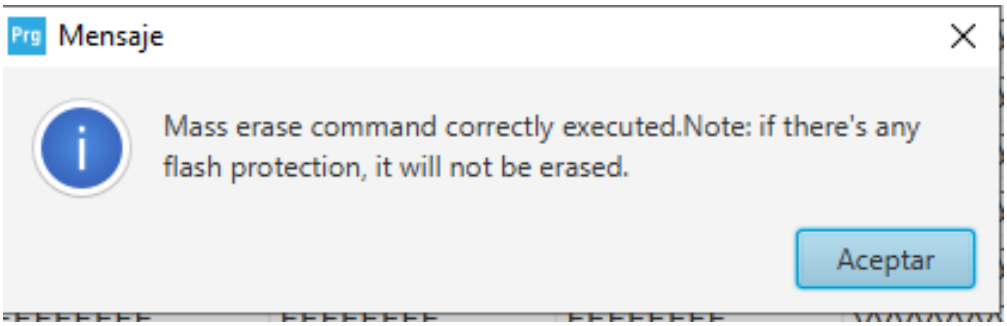

Ilustración 58: Cuadro de diálogo para proceso de borrado correcto

En segundo lugar, se procede a instalar el firmware. Para ello se selecciona la pestaña de carga (marcada con un círculo rojo en la ilistración siguiente).

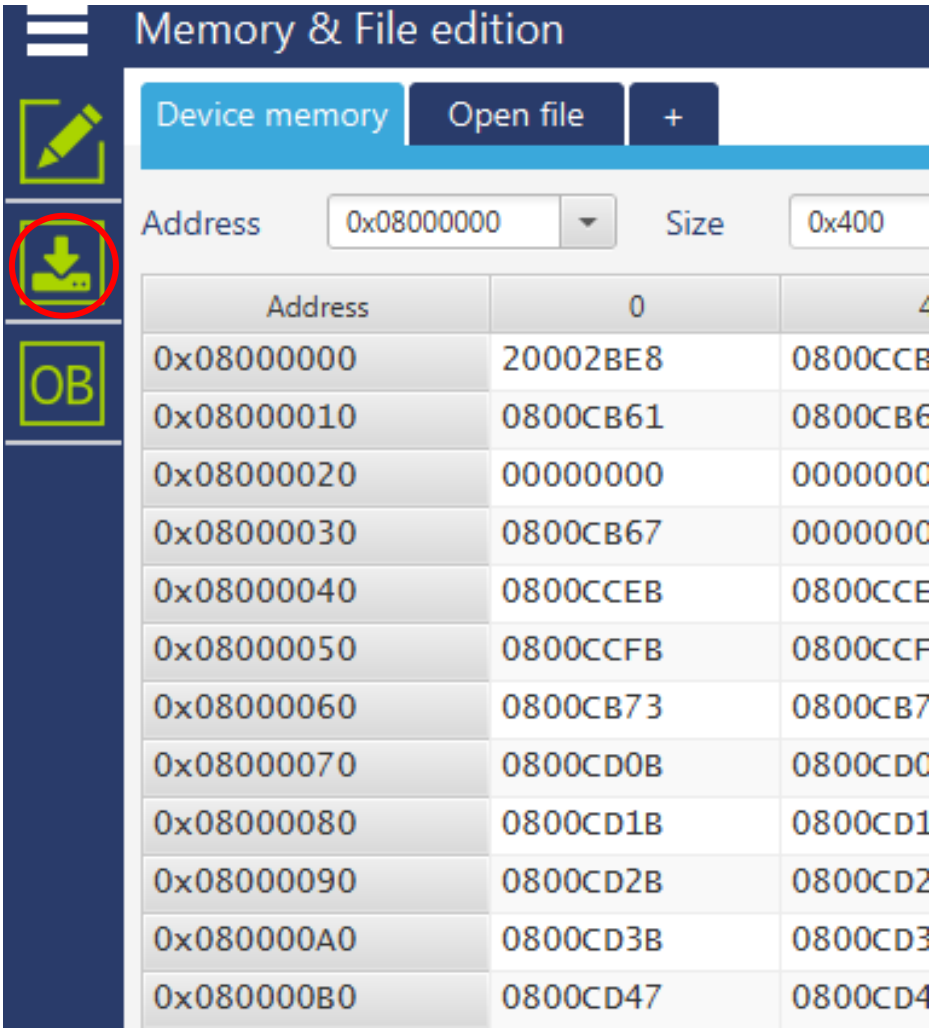

Ilustración 59: STM32CubeProgrammer III

Al seleccionar la pestaña de carga, aparecerá la siguiente ventana donde es posible seleccionar la ruta del firmware a cargar y realizar la carga del nuevo fimrware:

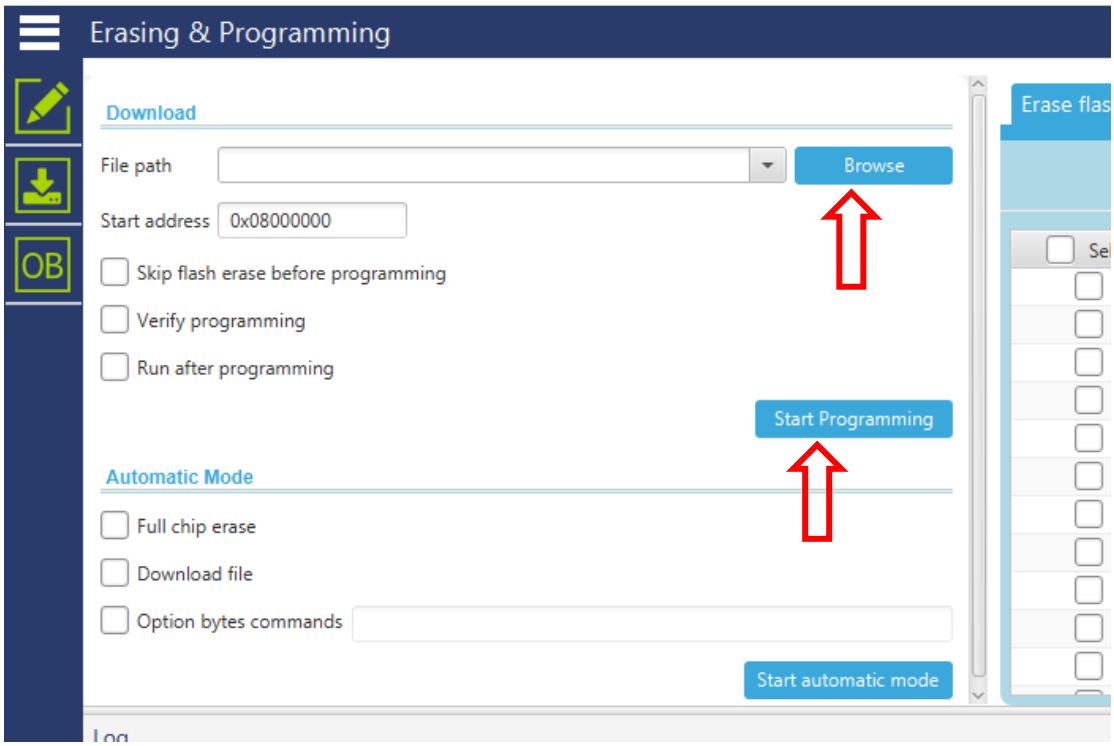

Ilustración 60: STM32CubeProgrammer IV

Ahora es necesario selecionar el binario que se desea cargar utilizando el botón "Browse" o escribiendo directamente la ruta del archivo. Por último, se selecciona "Start Programming". Si todo ha ido bien aparecerá el siguiente diálogo:

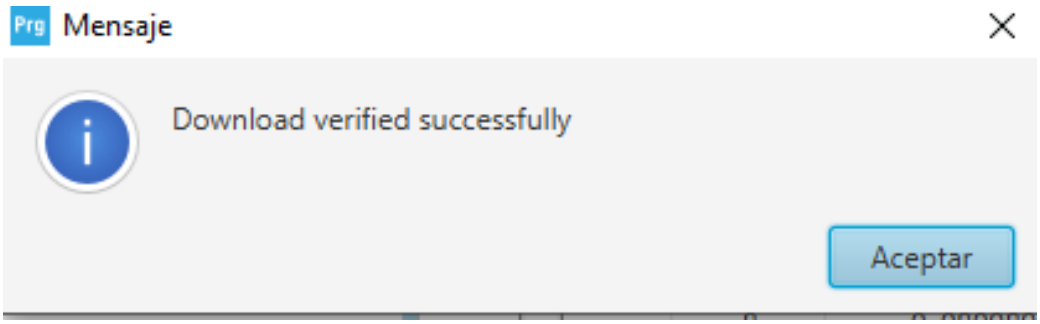

Ilustración 61: Cuadro de diálogo actualización de firmware correcta

La otra opción para realizar la actualización del firmware es utilizar el debugger ST-LINK V2. Para ello, se conecta el sensor al debugger, que a su vez se encuentra conectado al ordenador por un cable USB. El montaje al completo se describe en la Ilustración 62:

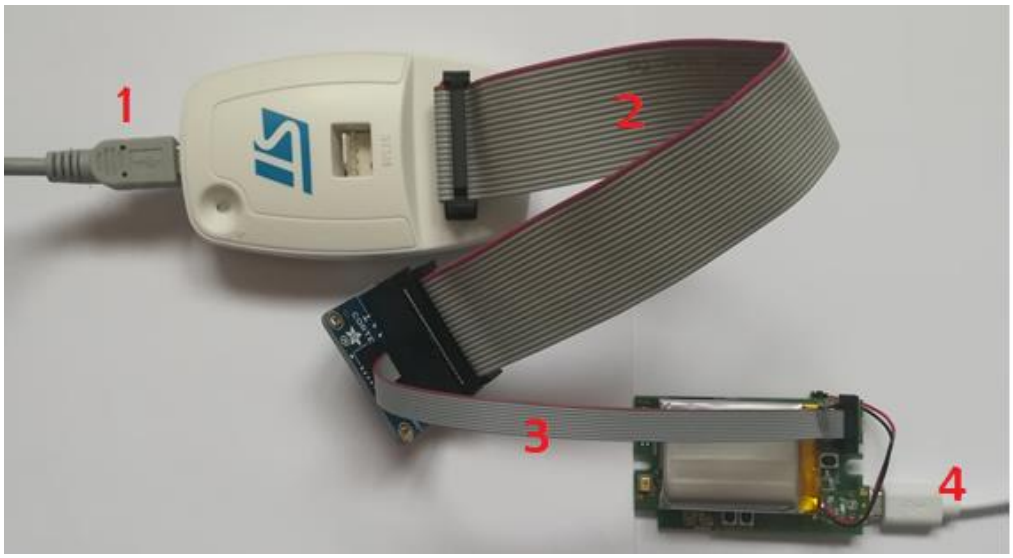

Ilustración 62: Montaje para actualización firmware

1. Conexión ordenador y debugger por medio de cable USB-miniUSB.
2. Conexión debugger y adaptador por medio de cable JTAG 20 de pines.
3. Conexión adaptador y placa por medio del cable SWD 10-14 pines.
4. Conexión placa y ordenador (únicamente para alimentación) por medio de cable USB-micro usb

Es importante realizar el montaje de forma correcta, conectando los cables de modo que la línea morada coincida con el pin 1 de la interfaz. Por seguridad, se recomienda conectar en último lugar los cables 1 y 4.

Cuando se haya realizado el montaje, se abre de nuevo el programa STMCubeProgramer. Antes de realizar la actualización del firmware, hay que asegurarse de que se encuentra instalada la última versión del firmware del ST LINK V2. Para ello, debemos conectarlo a un ordenador vía USB y abrir el programa STMCubeProgrammer. A continuación, seleccionar “Firmaware Update” como se muestra en la Ilustración 62.

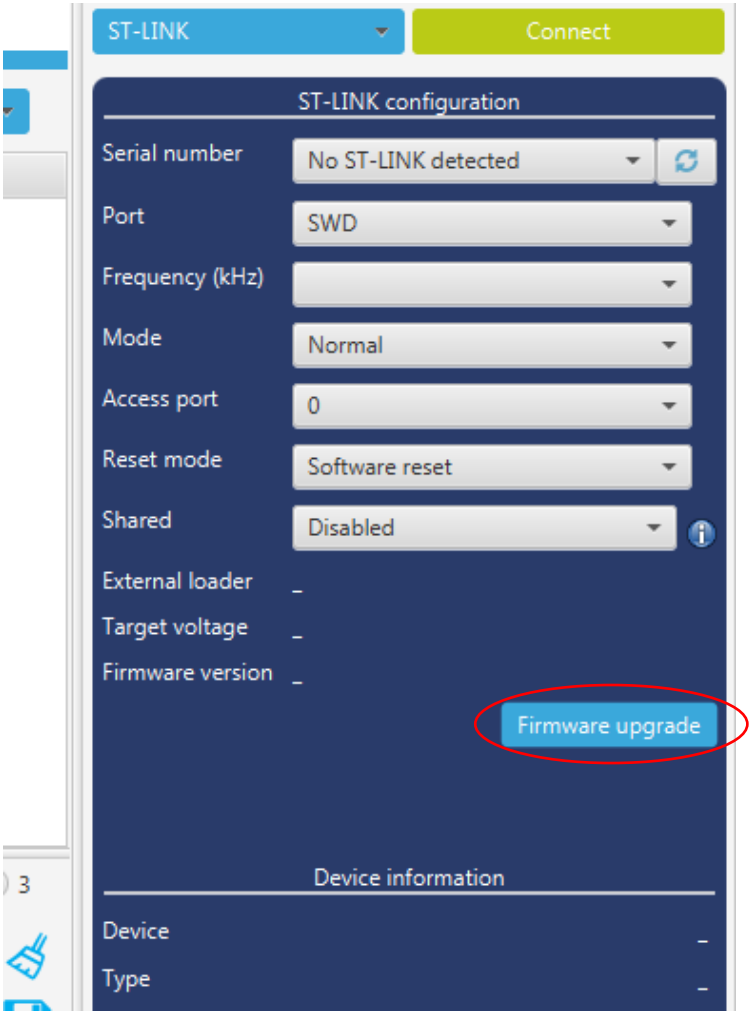

Ilustración 63: STM32CubeProgrammer Botón Firmware Update

Esto lleva a una nueva ventana. Seleccionar la opción “Open in update mode”.

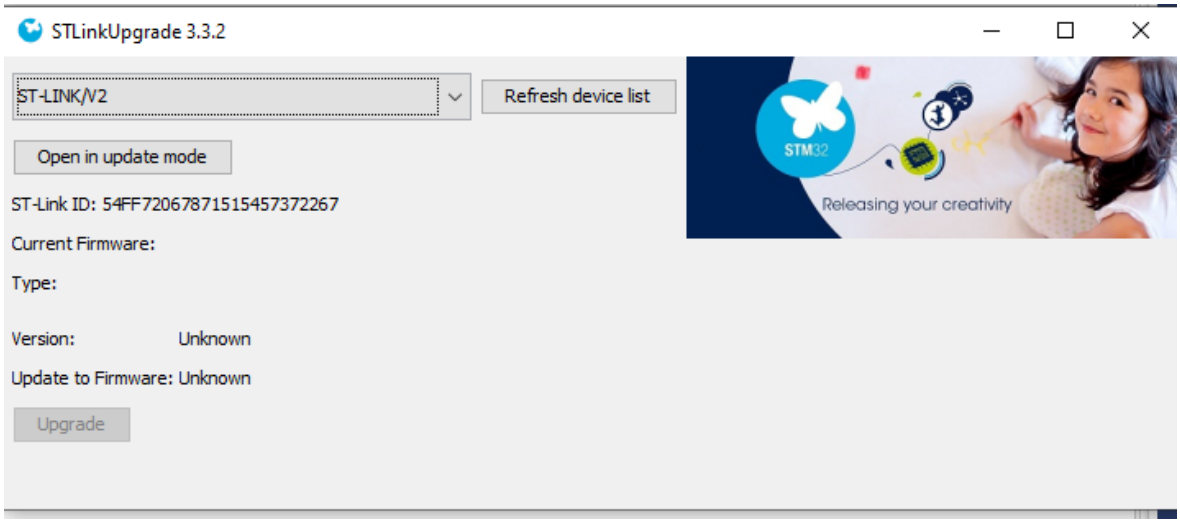

Ilustración 64: Ventana para ST LINK firmware update botón “Open in update mode”

Se muestra la versión del firmware instalada. Para actualizarlo a la última versión, seleccionar la opción “Upgrade”.

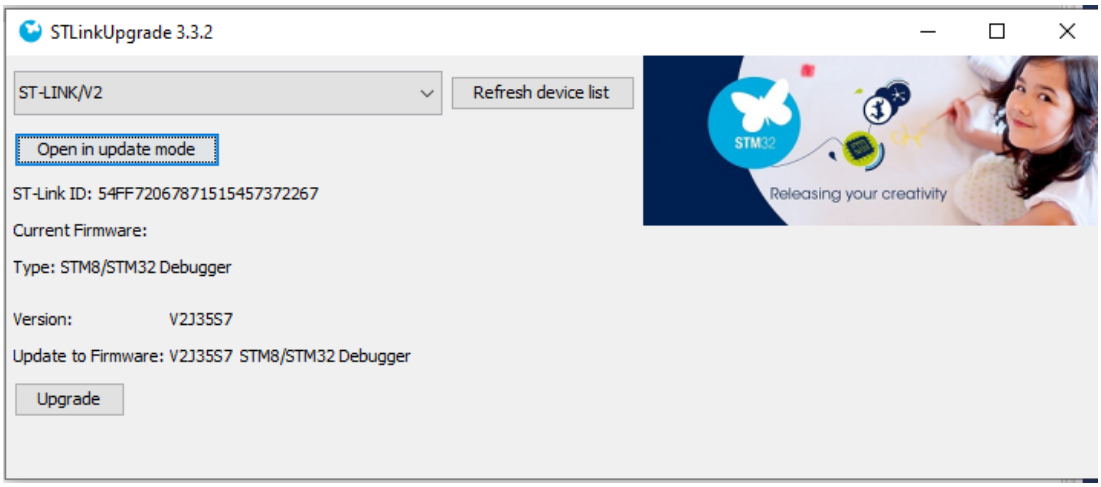

Ilustración 65: Ventana para ST LINK firmware update botón “Upgrade”

Cuando el proceso finalice, nos mostrará abajo del todo un mensaje confirmando que se ha realizado la operación con éxito como se muestra en la siguiente ilustración:

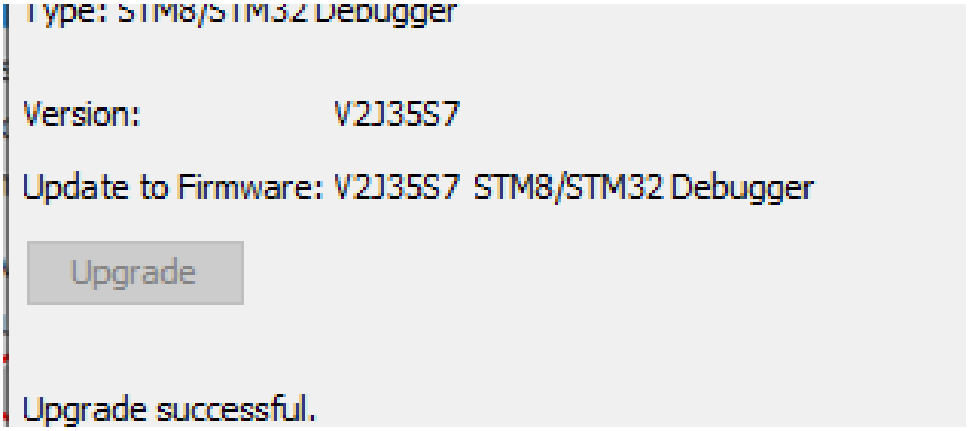

Ilustración 66: Ventana para ST LINK firmware update mensaje "upgrade successful"

Una vez realizada la actualización del firmware del debugger, es posible realizar la actualización del firmware del nodo multisensor con este segundo método. Se realiza el montaje de la Ilustración 62 y se abre el programa STCube Programmer. En este caso, se elige la opción ST-LINK como se muestra en la Ilustración 67.

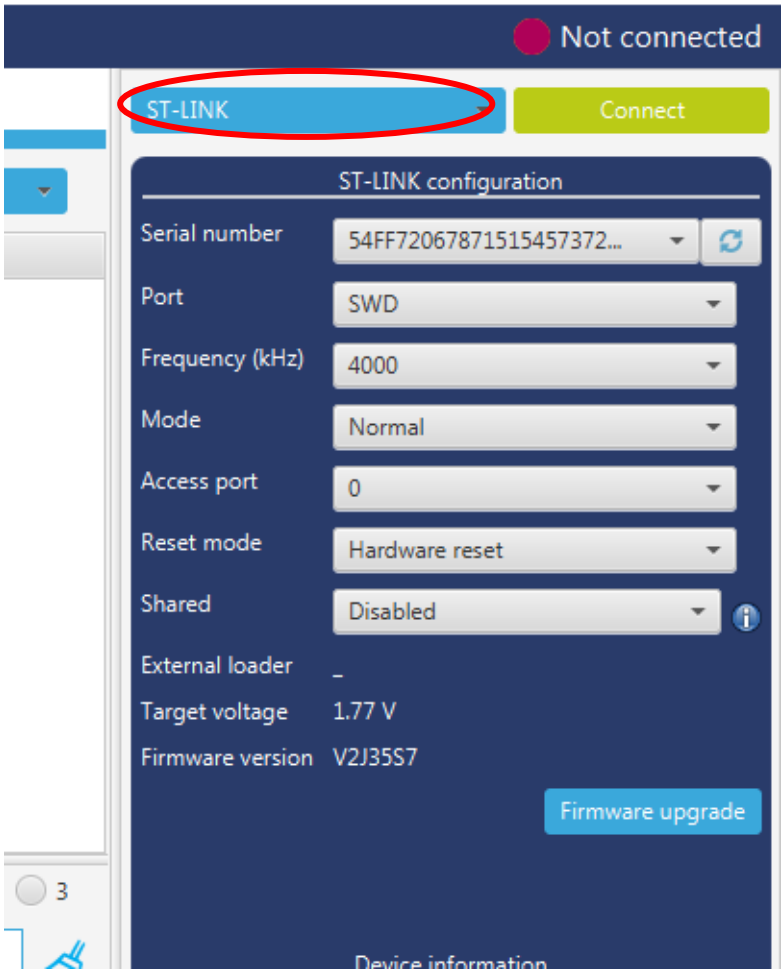

Ilustración 67: STM32CubeProgrammer-opción ST-LINK

Por último, se pulsa el botón "Connect" y se llevan a cabo los pasos realizados anteriormente.

5.5 EJEMPLO DE USO DE NUBE PÚBLICA

Una de las posibilidades para el desarrollo de una aplicación IoT es el uso de una nube pública. En este apartado se desarrolla una pequeña demo utilizando los servicios de AWS Greengrass.

Este primer ejemplo está formado por un nodo sensor y un gateway. En este ejemplo el sensor lee únicamente la temperatura de una sala. Estos datos son recogidos por el gateway, la RPi. Una vez tratados, se envían a AWS IoT en formato JSON, y usando el tópico "temperatura/sensor1". El JSON tiene la siguiente estructura:

- **timestap:** marca de tiempo generada en el momento en el que se realice la lectura.

- **sala**: nombre de la sala en la que está ubicado el sensor. Para esta pequeña prueba se usará la dirección del sensor.
- **temperatura**: temperatura (ºC) obtenida por el sensor.

```
1  {
2     "timestamp":1234,
3     "sala":sensor_addr,
4     "temperatura":t
5  }
```

Ilustración 68: Estructura archivo JSON

En primer lugar, se accede a la consola de AWS, y se escoge el servicio AWS IoT. Desde aquí se puede configurar y administrar los distintos elementos de AWS IoT.

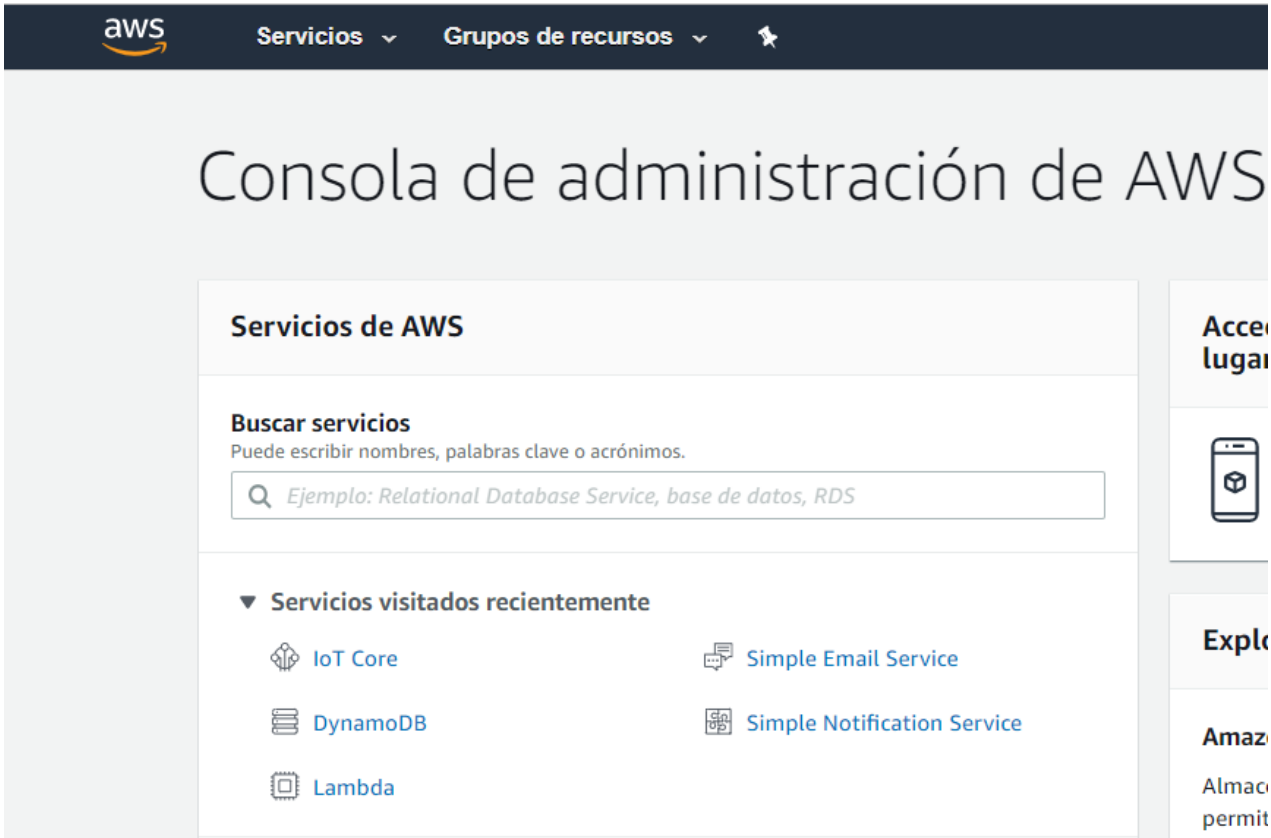

Ilustración 69: Consola AWS

La primera acción es la creación de un grupo. El grupo utilizado se llamará *"TempAulas"*. Este grupo consta de un core, que se ejecutará en la RPi. No es necesario definir ningún dispositivo más dentro del grupo ya que el cliente MQTT utilizado para enviar la información se ejecuta en la misma RPi que actúa como core. Como podemos ver en la Ilustración 70 y en la Ilustración 71, el grupo "TempAulas" y el core se han creado con éxito.

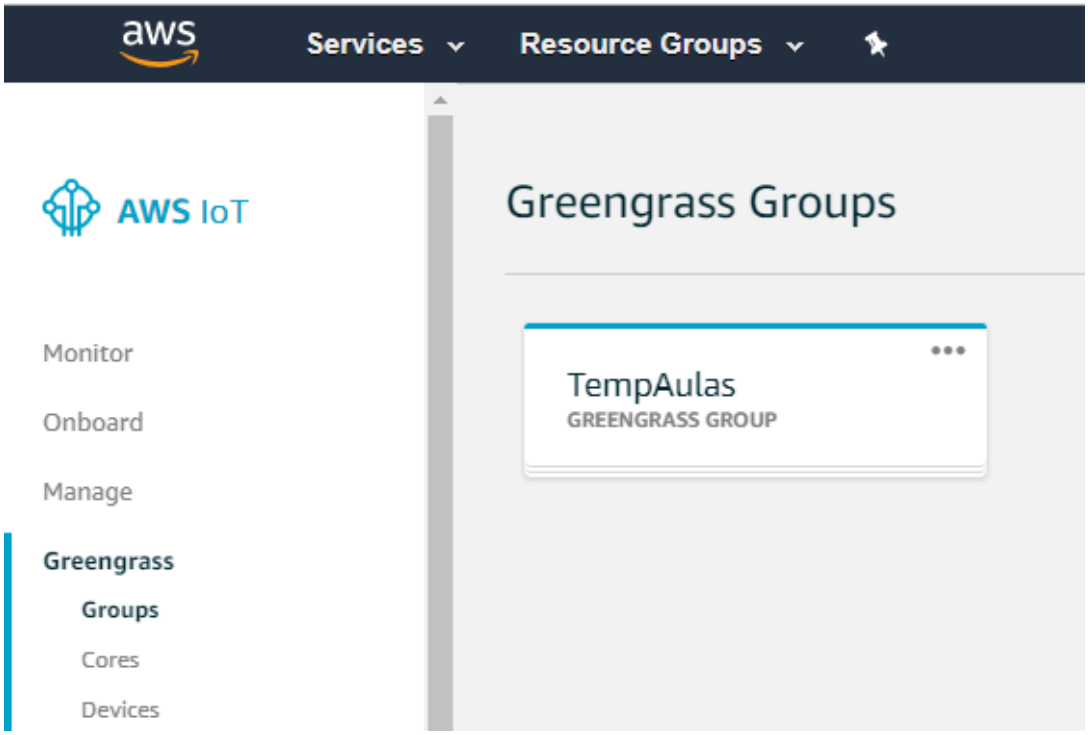

Ilustración 70: Consola AWS-Greengrass groups

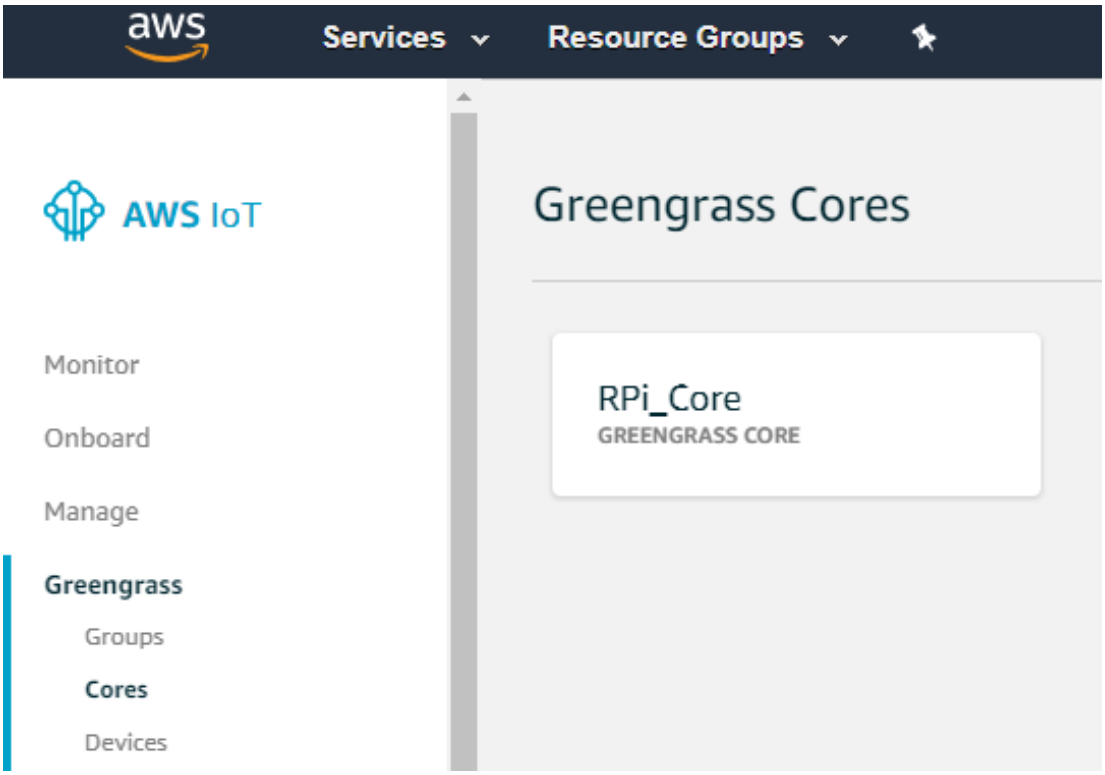

Ilustración 71: Consola AWS-Greengrass cores

El siguiente paso es instalar el software AWS GreenGrass core en la RPi. También es necesario descargar los certificados generados durante la creación del core en la consola AWS, y el certificado raíz.

Deben ser almacenados en el directorio /greengrass/certs, donde deben ubicarse todos los certificados y claves. En la siguiente ilustración se muestra el contenido del directorio, donde se han almacenado los certificados y claves generados.

```
pi@raspberrypi:~/Downloads/pi_core $ sudo tar -xzvf picore-setup.tar.gz -C /greengrass/
certs/48a024a24e.cert.pem
certs/48a024a24e.private.key
certs/48a024a24e.public.key
config/config.json
pi@raspberrypi:~/Downloads/pi_core $
```

Ilustración 72: Certificados y claves almacenados

Una vez realizado este proceso se prueba a lanzar el AWS IoT Greengrass en la RPi. El resultado de lanzarlo es el siguiente:

```
pi@raspberrypi:/greengrass/certs $ cd /greengrass/ggc/core/
pi@raspberrypi:/greengrass/ggc/core $ sudo ./greengrassd start
Process with pid 1693 does not exist already
Setting up greengrass daemon
Validating hardlink/softlink protection
Waiting for up to 1m10s for Daemon to start

Greengrass successfully started with PID: 2465
pi@raspberrypi:/greengrass/ggc/core $
```

Ilustración 73: Salida del comando para lanzar AWS Greengrass

Ahora que el core está lanzado, se pueden utilizar las funcionalidades de AWS, para los dispositivos de nuestro grupo. El siguiente paso es crear una regla, donde se establece que cada vez que se publique una temperatura debe almacenarse en una tabla de Amazon DynamoDB

Amazon DynamoBD es una base de datos NoSQL que admite modelos de datos de clave-valor y de documentos. En este servicio es donde almacenaremos los datos recopilados por el sensor.

Primero, es necesario crear una tabla. Para ello, se accede al servicio DynamoDB en la consola de AWS. Se selecciona la opción "Crear Tabla", como se muestra en la Ilustración 74.

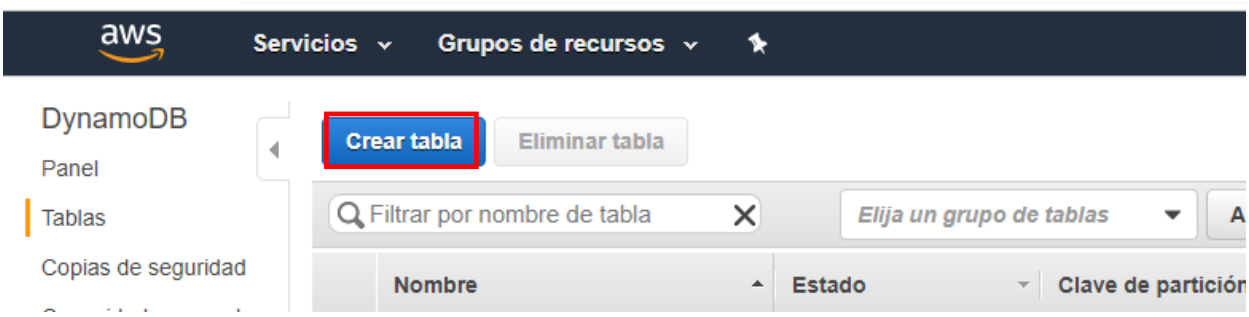

Ilustración 74: Opción Crear Tabla consola DynamoDB

Se crea la tabla "*Temperaturas*", donde la clave primaria será el timestamp. Se añade la sala como clave de ordenación tal y como se indica a continuación:

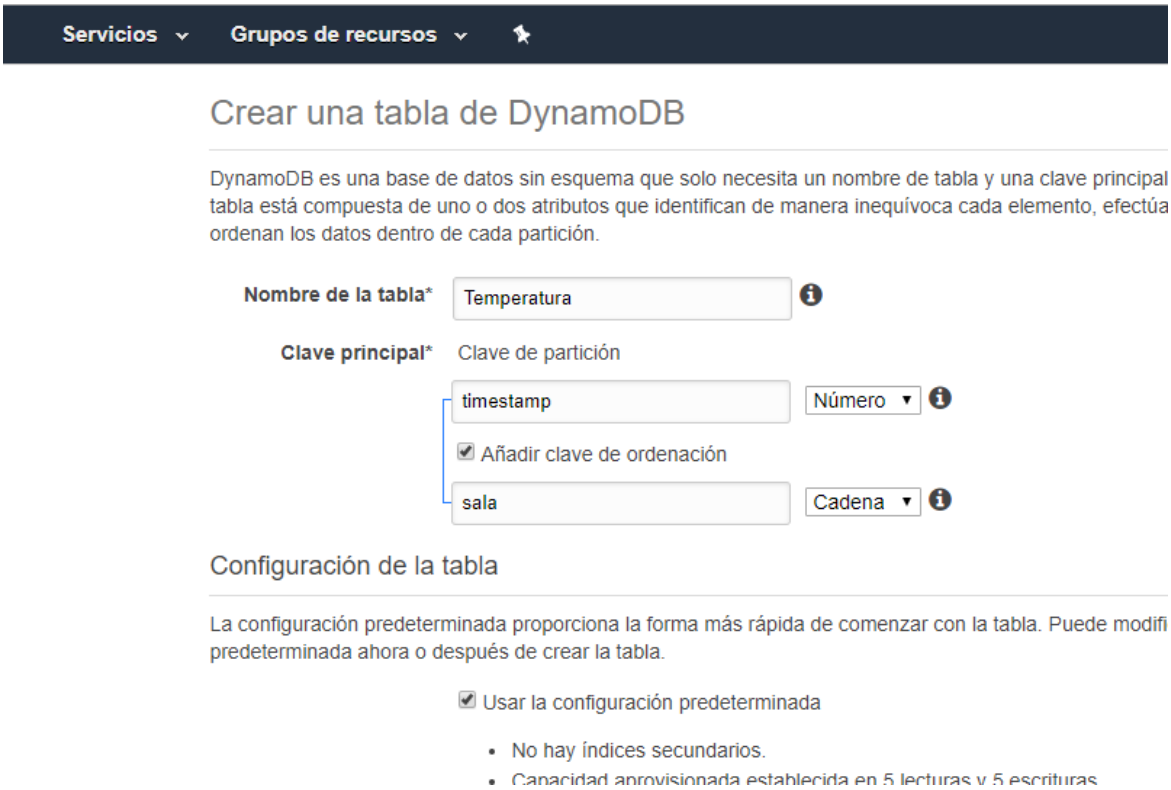

Ilustración 75: Configuración de la tabla

A continuación, hay que establecer la regla encargada de almacenar los datos en la tabla creada en el apartado anterior. Para ello se vuelve de nuevo a la consola de IoT Core para crear la regla, y se crea la regla "*updateTemp*". Para ello, accedemos al apartado "Reglas" de la consola:

Ilustración 76: Consola AWS-opción Actuar

Configuramos la regla como se realiza en la Ilustración 77 e Ilustración 78. Además de poner un nombre y una descripción, es necesario definir cuando se quiere que se aplique esta regla y que acción debe realizar. Para indicar cuando debe aplicarse, se crea la "instrucción de consulta de regla". Esta es una consulta SQL donde se selecciona que información se quiere rescatar de entre toda la publicada en un tópico. En este caso seleccionamos todo lo publicado para el tópico "*temperatura/sensor1*".

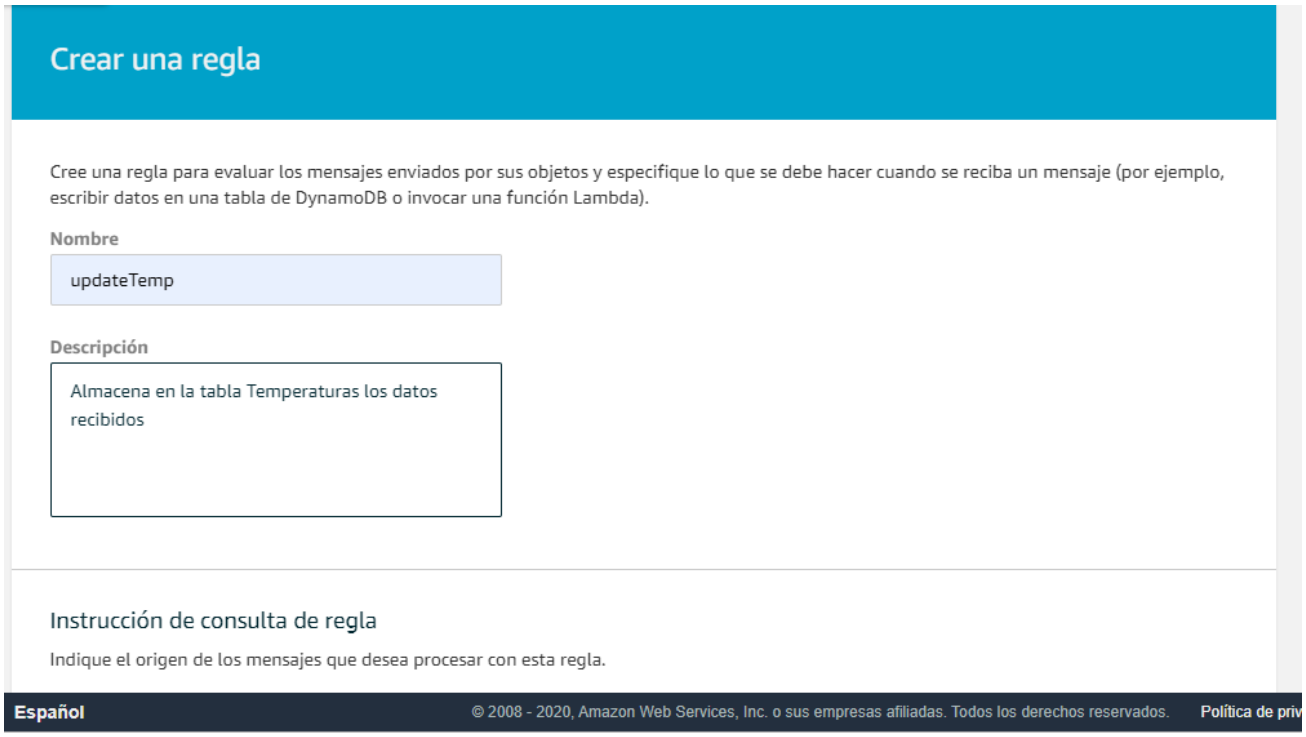

Ilustración 77: Configuración de la regla I

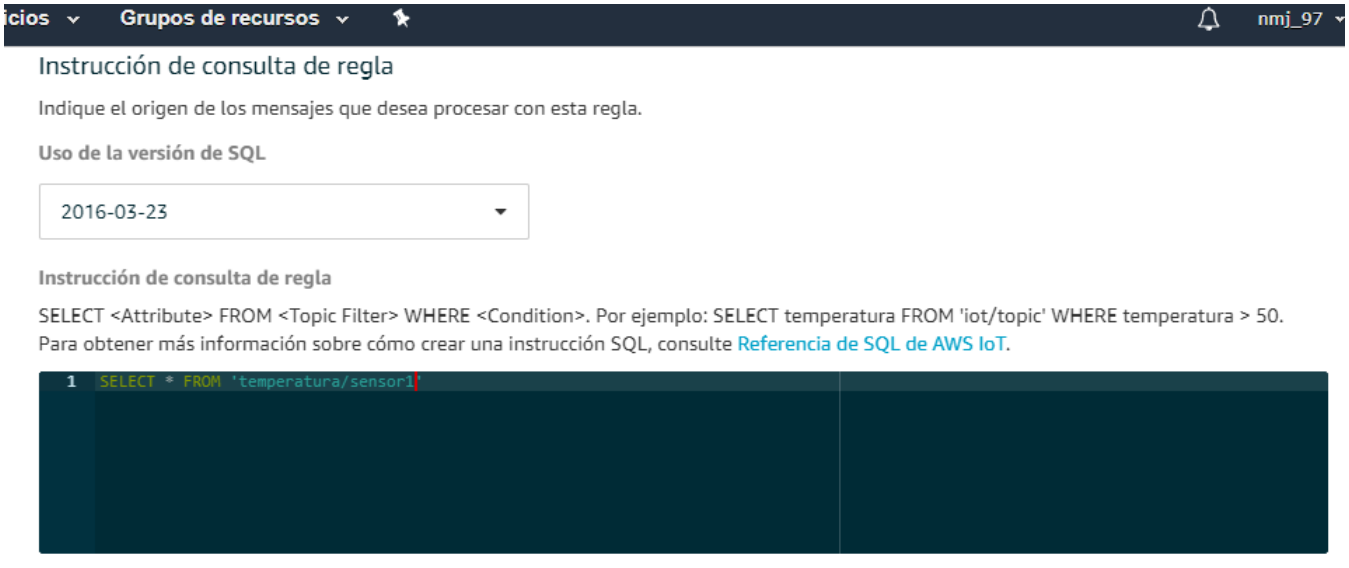

Ilustración 78: Configuración de la regla II

Tras establecer la instrucción de consulta de regla, hay que indicar que acción se quiere realizar. De entre las diferentes acciones posibles, seleccionamos la acción “Insertar un mensaje en una tabla de DynamoDB”. Esta opción se muestra en la siguiente ilustración:

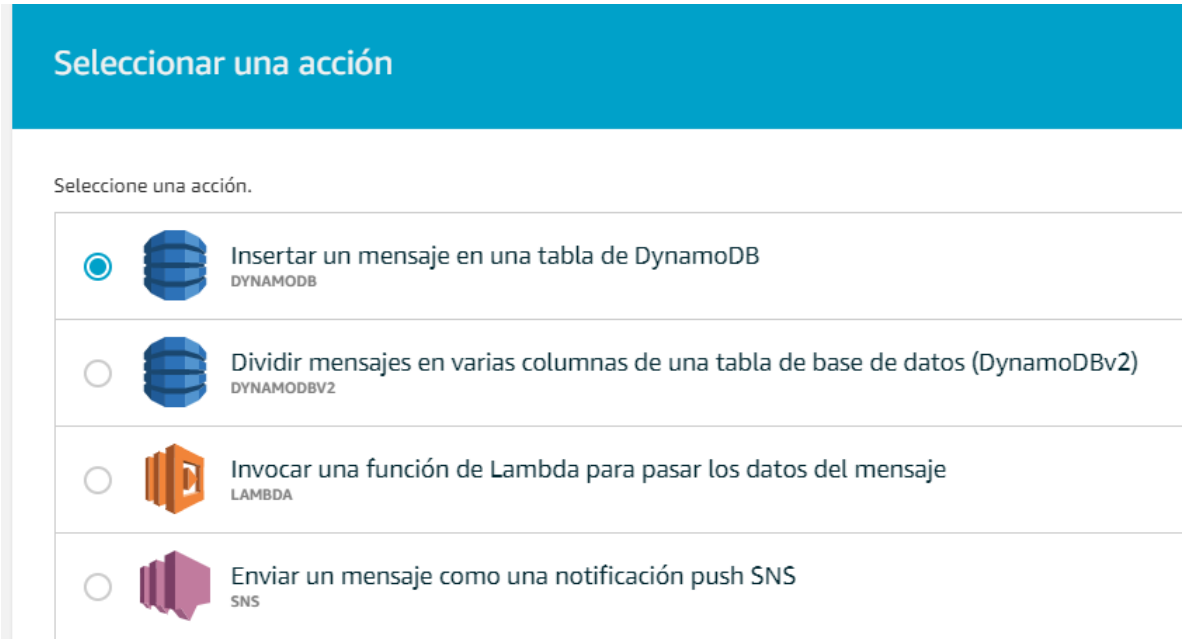

Ilustración 79: Configuración de la regla III

Se realiza la configuración indicada en la Ilustración 80, eligiendo como recurso la tabla creada anteriormente. Se indica que valores deben seleccionarse para la clave primaria y la clave de ordenación. Para indicar que queremos coger el valor del atributo “timestamp” de la consulta se utiliza el formato ${timestamp}. La operación a realizar es INSERT.

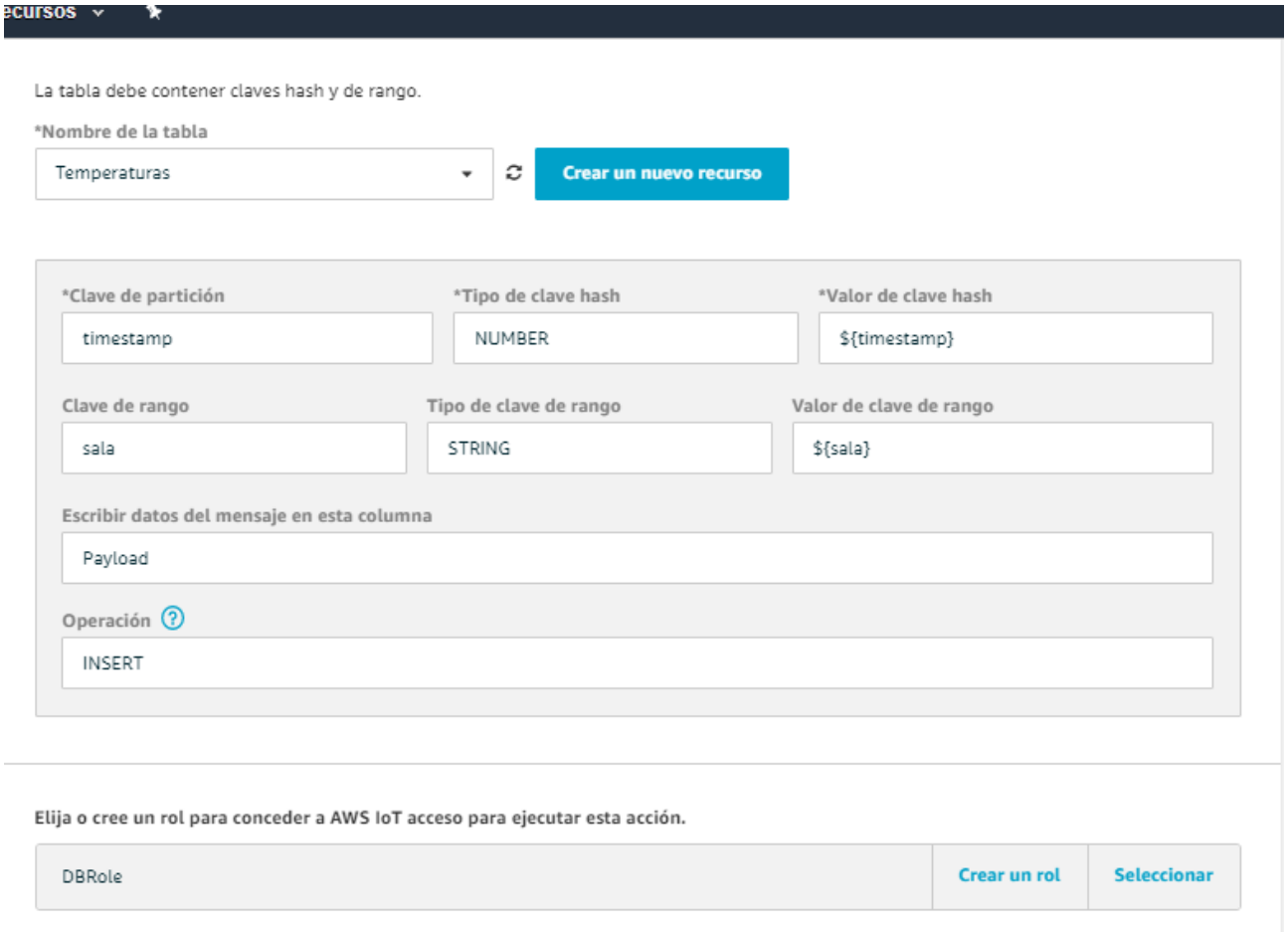

Ilustración 80: Configuración de la regla IV

Por último, se crea un rol, seleccionando el botón "*Crear Rol*", como se expone en la siguiente iustración:

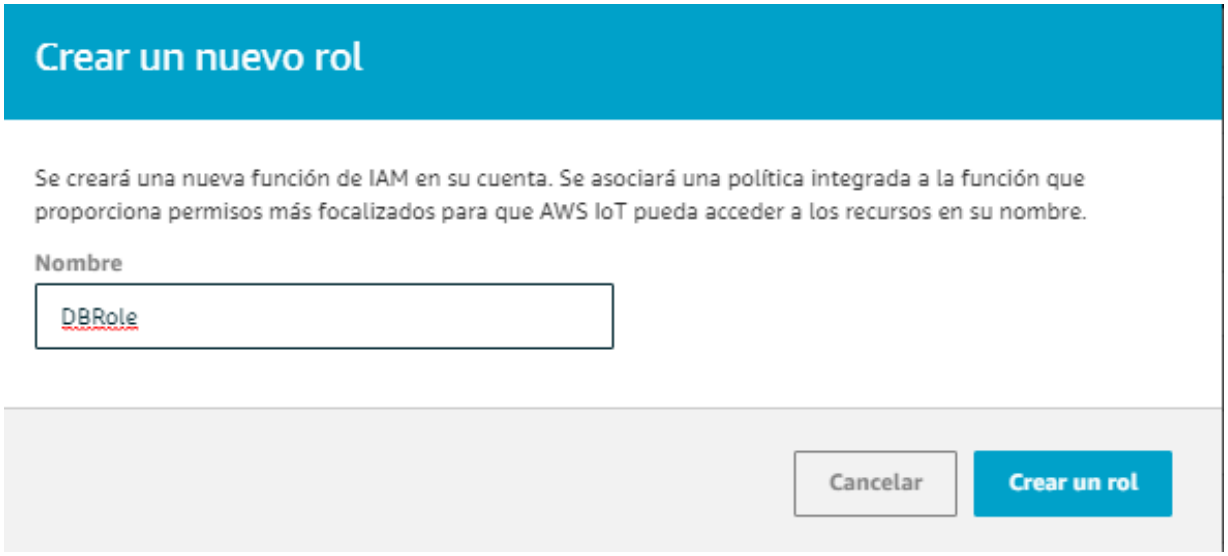

Ilustración 81: Configuración de un nuevo rol

El último paso es crear el cliente MQTT que publique las lecturas de temperatura. Para ello es necesario instalar el SDK aws-iot-device-sdk[31] en la RPi. Este es la nueva generación para crear clientes AWS IoT para Python.

Para crear el cliente MQTT se utiliza el script "temperature_AWS". Para su desarrollo se realizan modificaciones en el script "temperature_sem.py" usado anteriormente. Además de mostrar la temperatura, se crea un cliente MQTT que publique en el tópico "temperatura/sensor1" la temperatura con el formato indicado anteriormente.

Para comprobar que todo se encuentra configurado correctamente, se ejecuta el script "temperature_AWS.py" para que se conecte con el sensor y suba a la nube las lecturas de temperatura. La salida de ejecutar este script es la siguiente:

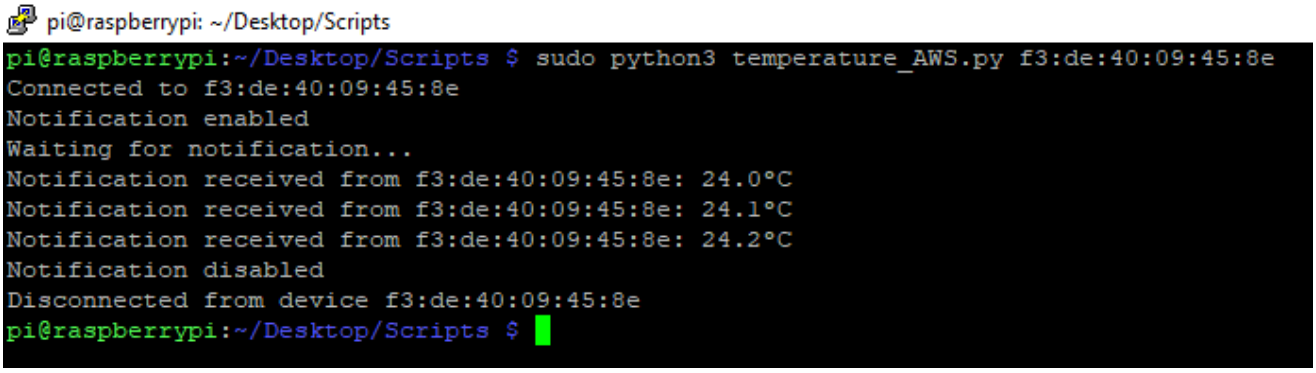

Ilustración 82: Salida script temperature_AWS.py

Como observar en la Ilustración 83, la tabla "*Temperaturas*" se ha actualizado y tiene una entrada para cada lectura de temperatura.

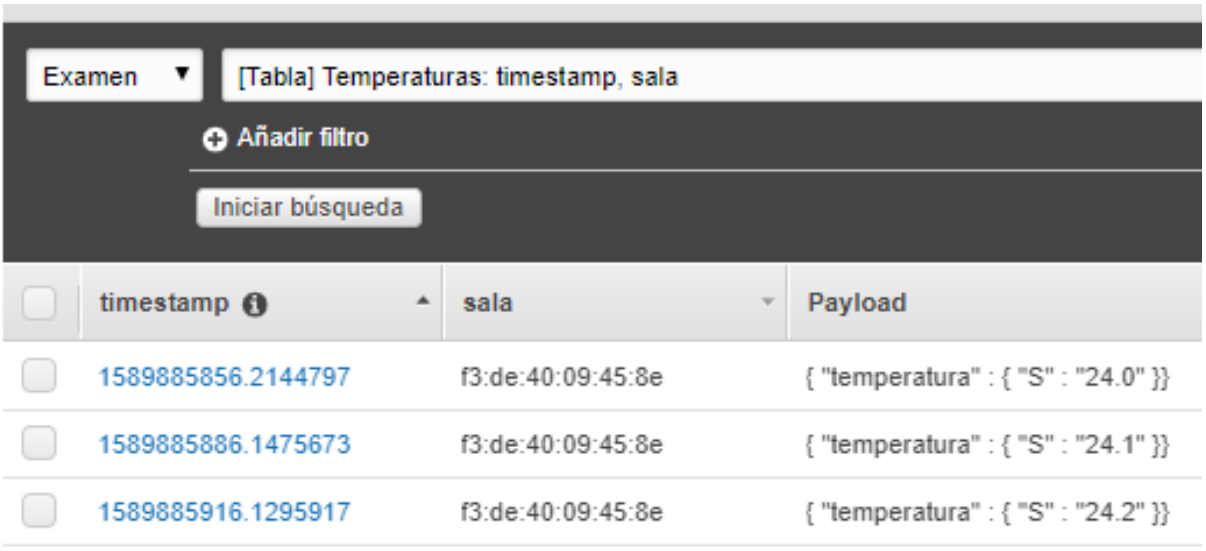

Ilustración 83: Contenido de la tabla Temperaturas

[31] (aws-iot-device-sdk)

AWS IoT permite usar un cliente MQTT desde la consola para realizar pruebas. Desde él se pueden crear suscripciones a un tópico o publicar mensajes en un tópico. Para acceder a este cliente, se debe elegir el apartado "Pruebas" de la consola señalado en la siguiente ilustración:

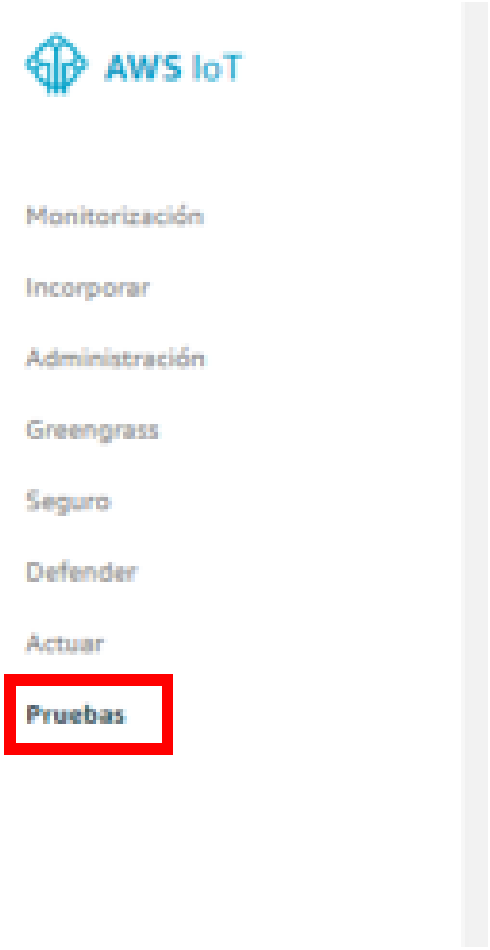

Ilustración 84: Consola AWS-Pruebas

Si se crea una suscripción al tópico “temperatura/sensor1”, es posible ver que los mensajes se están publicando correctamente como se muestra en las siguientes ilustraciones:

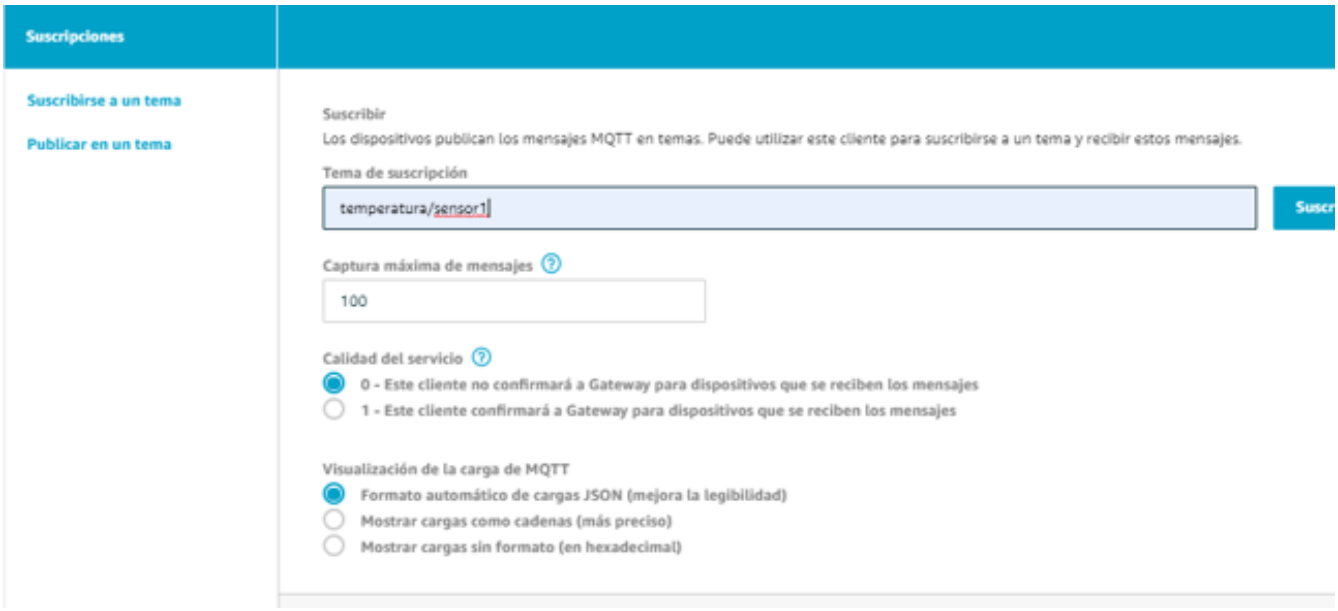

Ilustración 85: Consola AWS-Crear suscripción

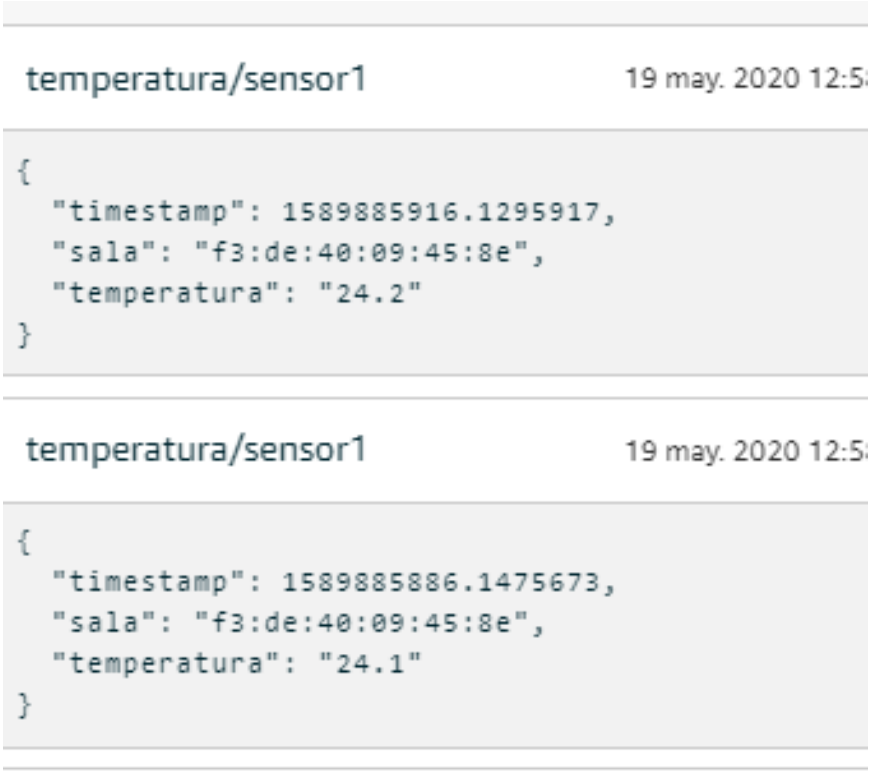

Ilustración 86: Cliente MQTT consola AWS-mensajes recibidos

5.6 EJEMPLO DE USO DE NUBE PRIVADA

Al igual que con AWS Greengrass, se desarrolla una primera demo de aplicación IoT formada por un sensor y un gateway, pero en este caso usando Thingsboard.

Existen diversas opciones de instalación, tanto en la nube como "on-premise". Se elige la opción "on-premise", y se realiza en la RPi. La RPi además de actuar como gateway, ejecuta el software de Thingsboard y alberga toda la información. Es decir, emulará la nube. En una implantación real Thingsboard estará albergado en otro dispositivo. Por ejemplo, una máquina virtual en un servidor de la universidad.

Para su instalación se siguen los pasos del manual de instalación[32]. Este manual recoge todos los pasos a realizar para poder instalar y lanzar una instancia de Thingsboard correctamente en una RPi. Se divide en 6 pasos:

- **Paso 1. Installar Java:** Thingsboard se ejecuta en Java 8. Por tanto, es necesario OpenJDK. Para ello se ejecuta el comando:
  `sudo apt install openjdk8-jdk`

- **Paso 2. Descargar Thingsboard e instalarlo:** Descargar el paquete e instalarlo como un servicio. Se utilizan los siguientes comandos:
  Para descargar
  `wget https://github.com/thingsboard/release/download/v2.4.3/thingsboard-2.4.3.deb`
  Para instalar
  `sudo dpkg -i Thingsboard-2.4.3.deb`

- **Paso 3. Configurar la base de datos de Thingsboard:** Es necesario instalar un gestor de bases de datos. En el manual se opta por PostgresSQL. En primer lugar, se instala y se lanza
  `sudo apt-get install postgresql postgresql-contrib`
  `sudo service postgresql start`
  Una vez que el servicio está en ejecución, se realiza la conexión y se crea la base de datos Thingsboard
  `psql -U postgres -d postgres -h 127.0.0.1 -W`
  `CREATE DATABASE thingsboard;`
  `\q`
  También es necesario editar el archivo de configuración "/etc/thingsboard/conf/thingsboard.conf" y añadir las siguientes líneas:

[32] (Manual instalación Thingsboard)

```
# DB Configuration
export DATABASE_ENTITIES_TYPE=sql
export DATABASE_TS_TYPE=sql
export SPRING_JPA_DATABASE_PLATFORM=org.hibernate.dialect.PostgreSQLDialect
export SPRING_DRIVER_CLASS_NAME=org.postgresql.Driver
export SPRING_DATASOURCE_URL=jdbc:postgresql://localhost:5432/thingsboard
export SPRING_DATASOURCE_USERNAME=postgres
export SPRING_DATASOURCE_PASSWORD=PUT_YOUR_POSTGRESQL_PASSWORD_HERE
```

Ilustración 87: Configuración base de datos Thingsboard

- **Paso 4. Actualización de memoria para máquinas lentas:** en el mismo archivo de configuración editado en el paso 3, se inserta la siguiente línea para restringir el uso de memoria RAM:
  export JAVA_OPTS="$JAVA_OPTS -Xms256M -Xmx256M"

- **Paso 5. Script de instalación:** se ejecuta el archivo de instalación para finalizar la instalación por completo. Esto se hace ejecutando el siguiente comando:
  sudo /usr/share/thingsboard/bin/install/install.sh --loadDemo
  Es posible incluir la opción --loadDemo para que cargue una base de datos de ejemplo, o no incluirla para utilizar una base de datos vacía.

- **Paso 6. Inicio servicio Thingsboard:** el último paso es lanzar el servicio para poder empezar a trabajar, con el siguiente comando:
  sudo service thingsboard start

Cada vez que la RPi se apague, o se detengan los servicios, es necesario volver a lanzarlos. Para ello se deben ejecutar los siguientes comandos:

```
pi@raspberrypi:~ $ sudo service postgresql start
pi@raspberrypi:~ $ sudo service thingsboard start
```

Ilustración 88: Comandos para lanzar Thingsboard

Una vez instalado, se puede hacer uso del servicio. Para ello, se accede a través del navegador, usando la IP de la RPi. En este caso se accede a la dirección http://192.168.1.150:8080/login. Es necesario introducir un usuario y contraseña. En el primer acceso se utilizan las credenciales del usuario administrador creado por defecto durante la instalación.

Usuario : sysadmin@thingsboard.org

Contraseña: sysadmin

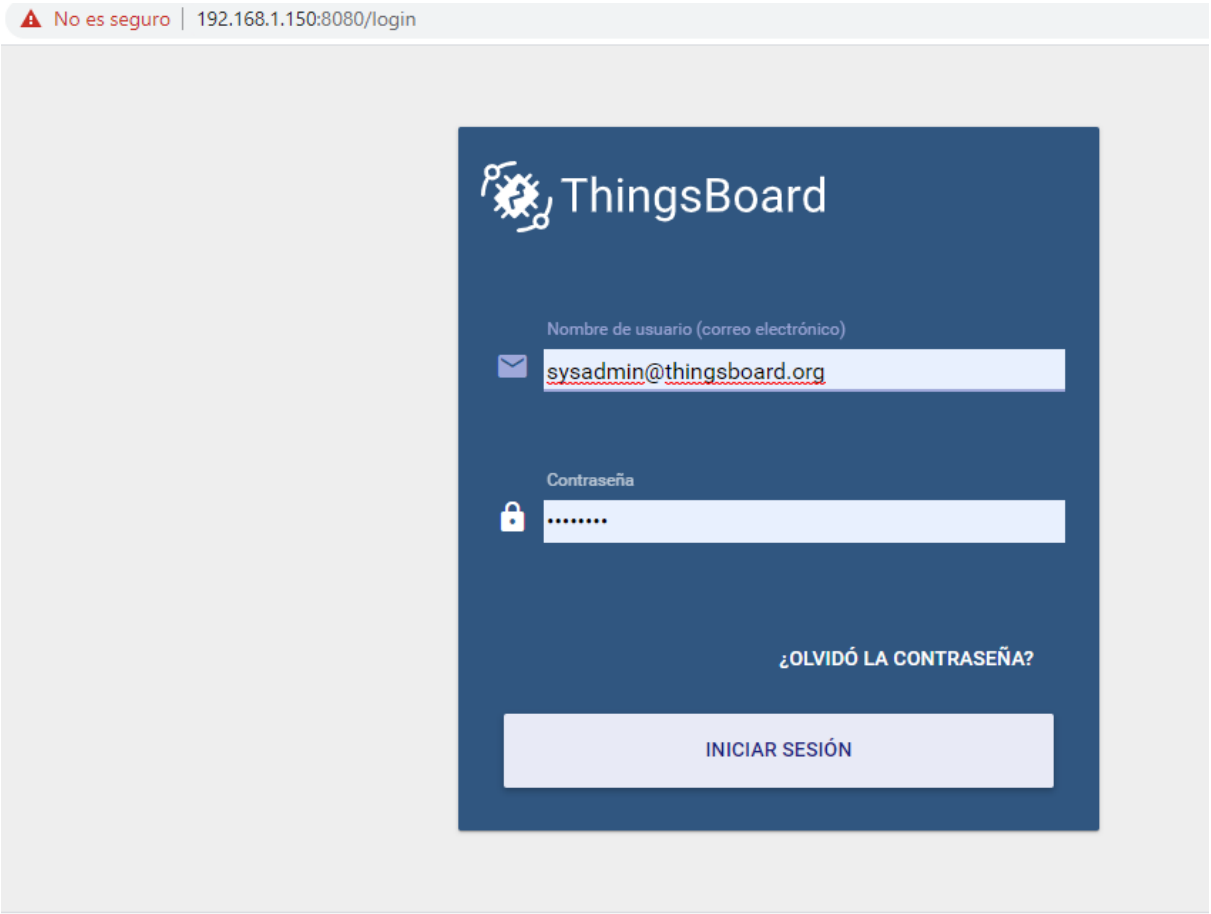

Ilustración 89: Formulario login Thingsboard

Una vez logueados, se dispone de acceso a la consola de administrador. Desde esta consola es posible crear organizaciones y clientes, además de realizar otras configuraciones. Tiene la siguiente apariencia:

Ilustración 90: Consola administrador Thingsboard

Para este ejemplo, se crea la organización “Departamento de Informática”. Este perfil será el encargado de administrar los diferente recursos y dispositivos. Para crear este perfil se realizan las siguientes acciones:

Se crea una nueva organización

Ilustración 91: Gestión de organizaciones

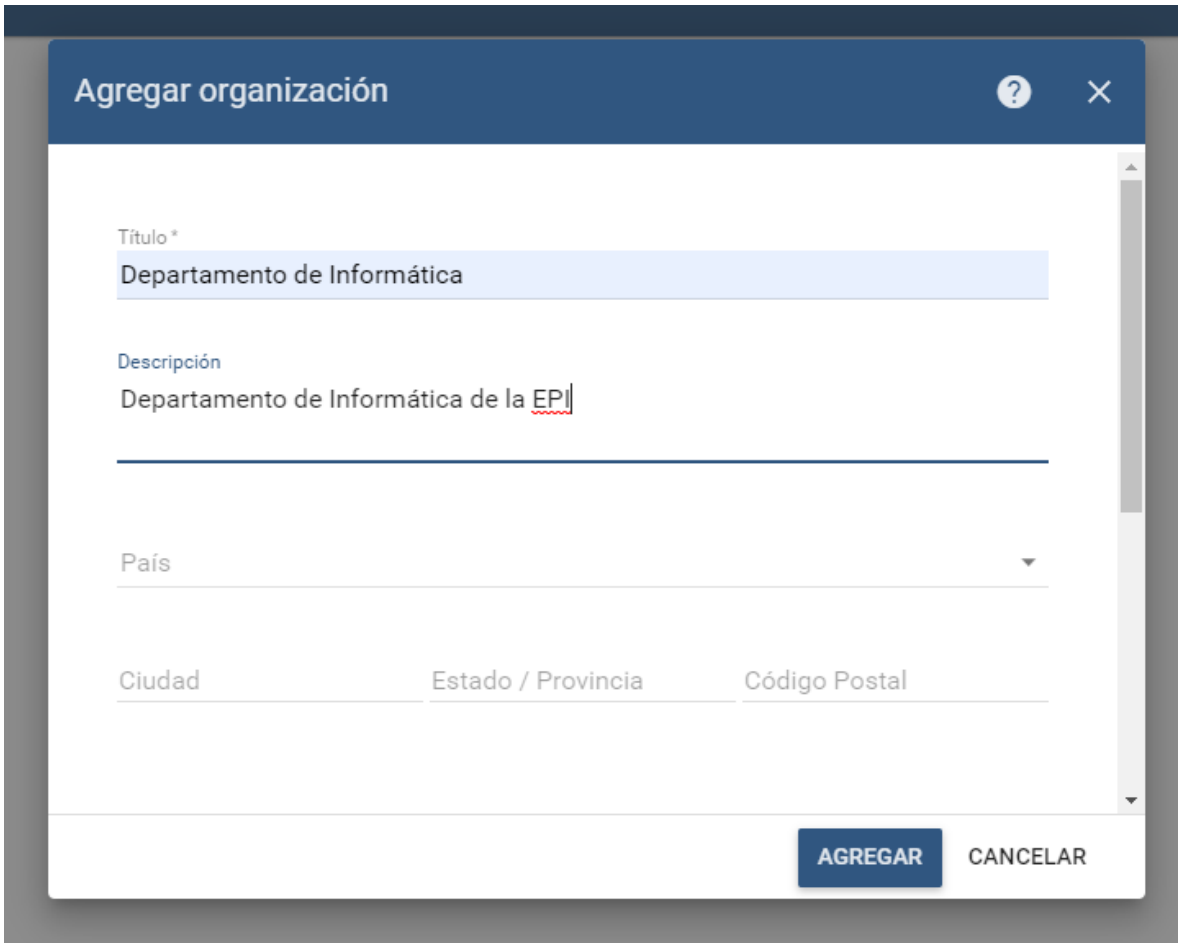

Ilustración 92: Crear organización

Se le asigna un usuario para que pueda loguearse. Para ello, se accede a través del botón "Gestionar administradores de la organización"

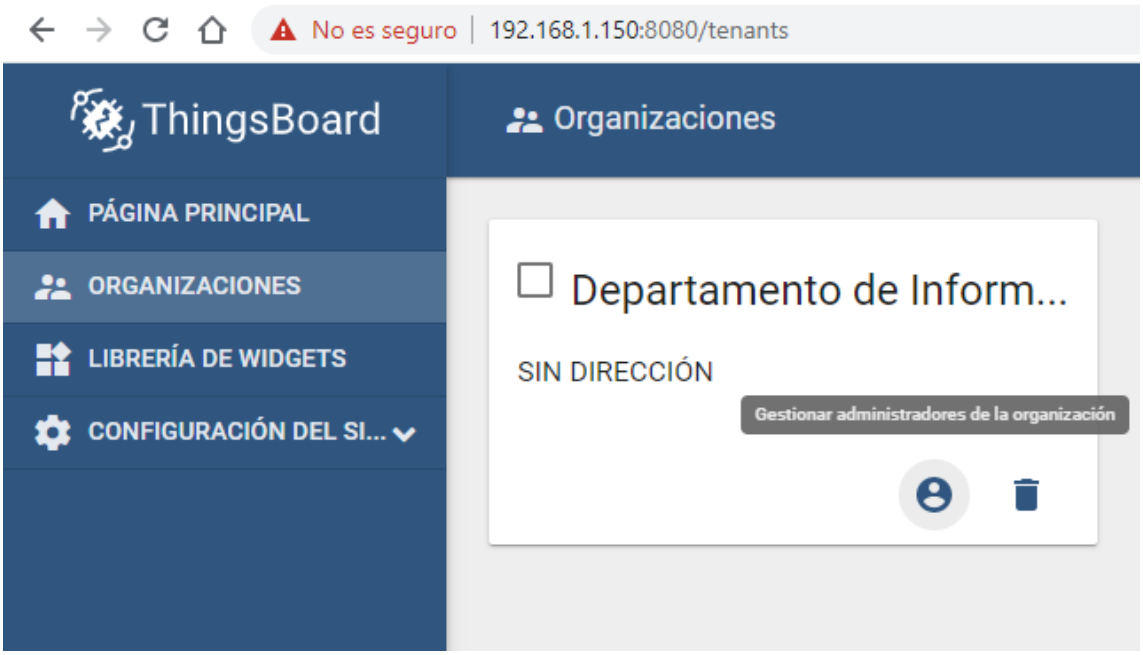

Ilustración 93: Botón gestionar administradores de la organización

Como se puede apreciar en la siguiente ilustración, no existen usuarios creados, por tanto, se crea uno, tal y como se hizo con la organización.

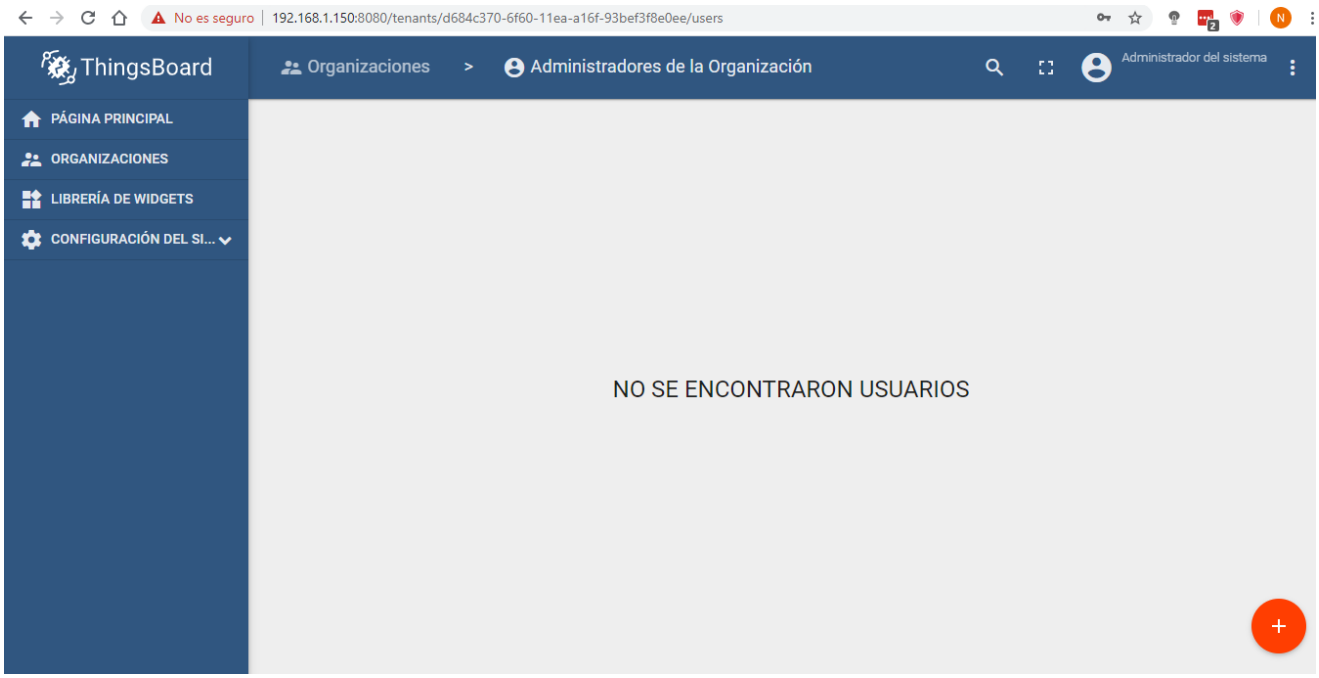

Ilustración 94: Gestión usuarios organización

El usuario accederá usando el siguiente correo: departamento_informatica@correo.com

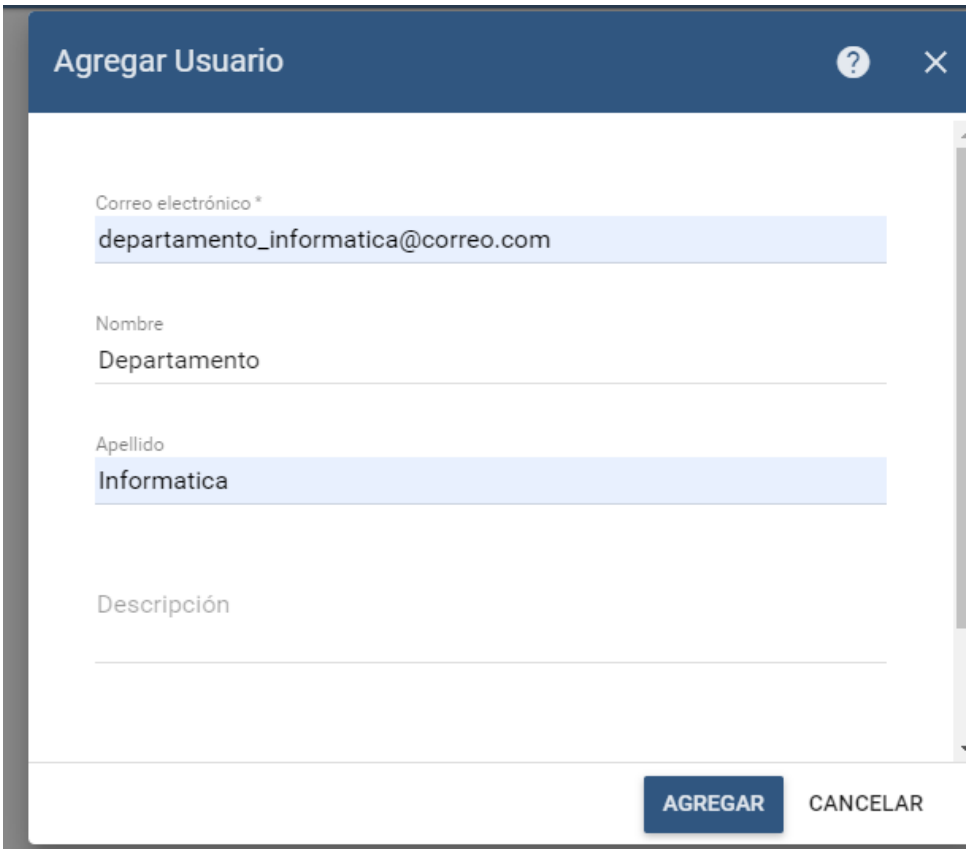

Ilustración 95: Crear nuevo usuario

Al agregar el usuario, aparece un diálogo con un enlace para activar la cuenta.

Ilustración 96: Diálogo para activar cuenta nuevo usuario

Si se accede al enlace, mostrará un formulario para introducir la contraseña que se quiere asignar al usuario.

Ilustración 97: Formulario para cambio de contraseña nuevo usuario

Al crear la contraseña, redirige directamente a la consola del usuario departamento_informatica. Desde esta consola se puede crear activos, dispositivos, reglas … y administrarlos.

Ilustración 98: Consola administrador de la organización

Ya es posible comenzar a dar de alta en Thingsboard los activos y dispositivos que forman el prototipo. En primer lugar, se selecciona la opción “Activos” para crear un nuevo activo, como se indica en la siguiente ilustración.

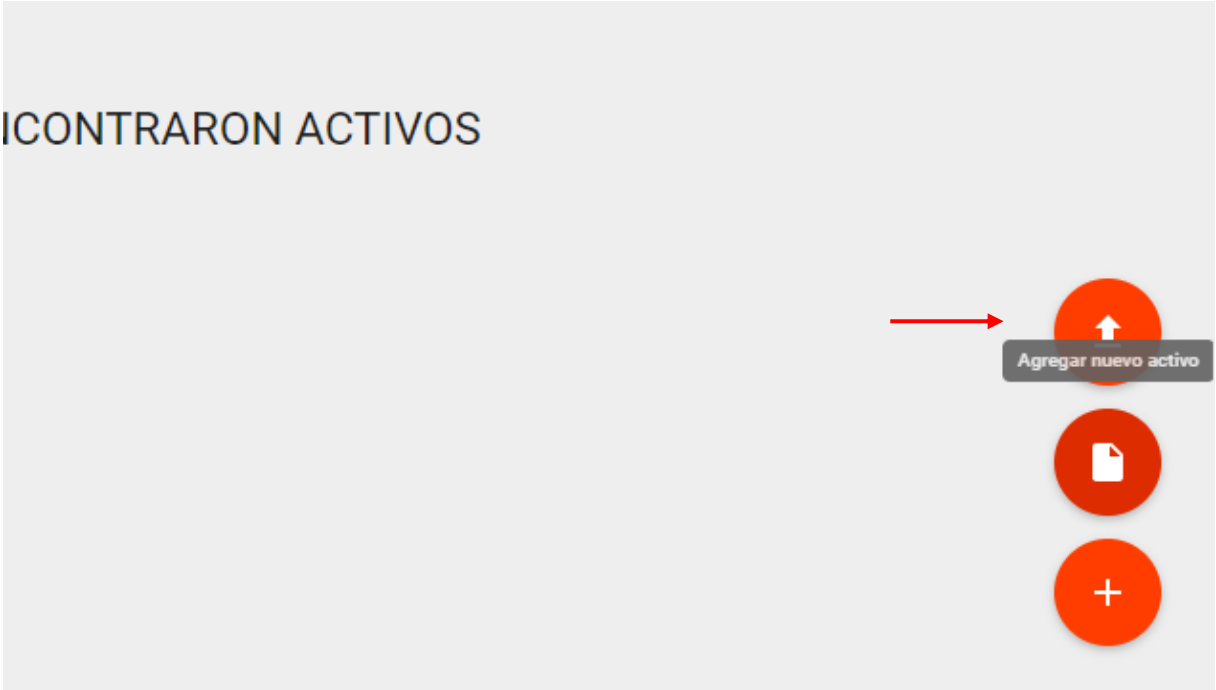

Ilustración 99: Botón para agregar nuevo activo

Se crea el activo “Edificio 1” representa el edificio donde se ubica el sensor, rellenando los campos indicados en la Ilustración 100.

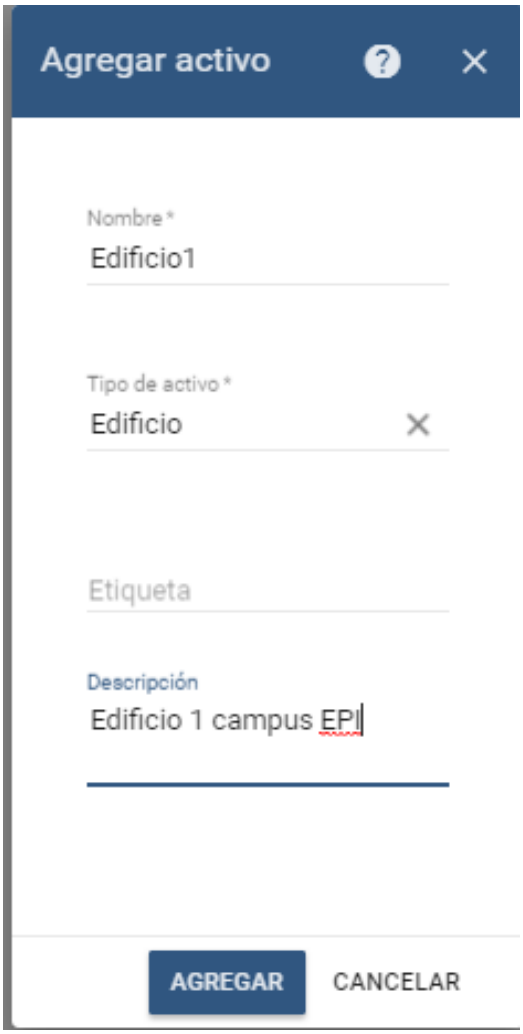

Ilustración 100: Agregar nuevo activo

En segundo lugar, se accede a la pestaña de "Dispositivos", para crear el dispositivo "SensorTemperatura1" que represente al sensor. El procedimiento es el mismo que para crear un activo.

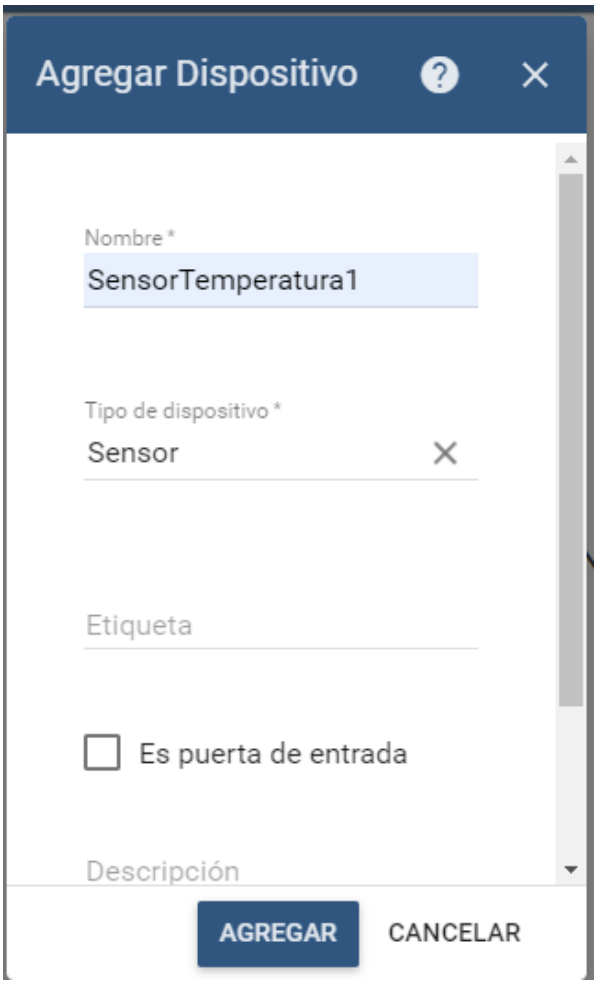

Ilustración 101: Creación nuevo dispositivo

Una vez creado, se le asigna una relación para indicar que este sensor está ubicado en el Edificio 1. De esta forma, cuando en un futuro se cree un usuario cliente, solo será necesario asignarle el activo "Edificio 1" y no todos los dispositivos contenidos en él. Para crear la relación, se accede a los detalles del activo, a la pestaña "Relaciones", y se selecciona agregar, como se describe en la Ilustración 102.

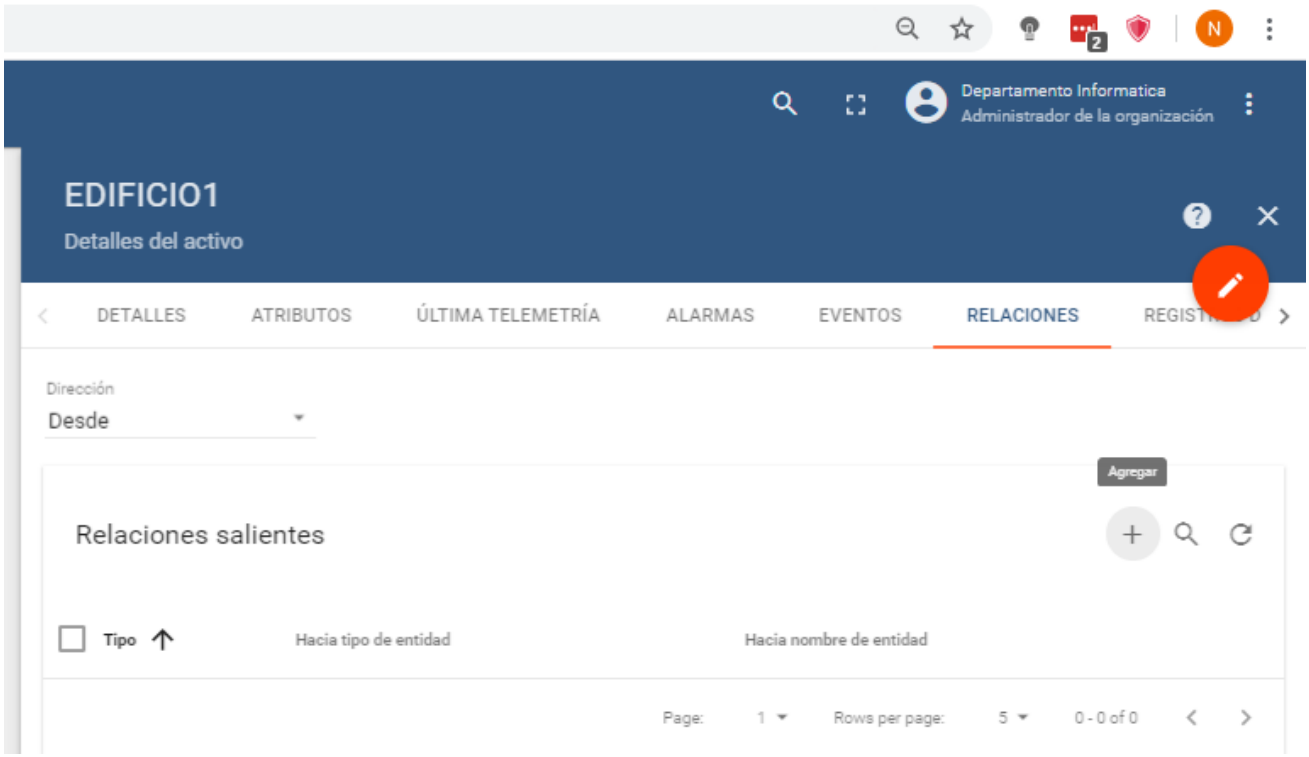

Ilustración 102: Botón agregar relación

Al pulsar sobre el botón “Agragar relación” aparece una nueva ventana, donde se configura el tipo de relación y con que elemento se relaciona. Esta ventana se muestra en la siguiente ilustración:

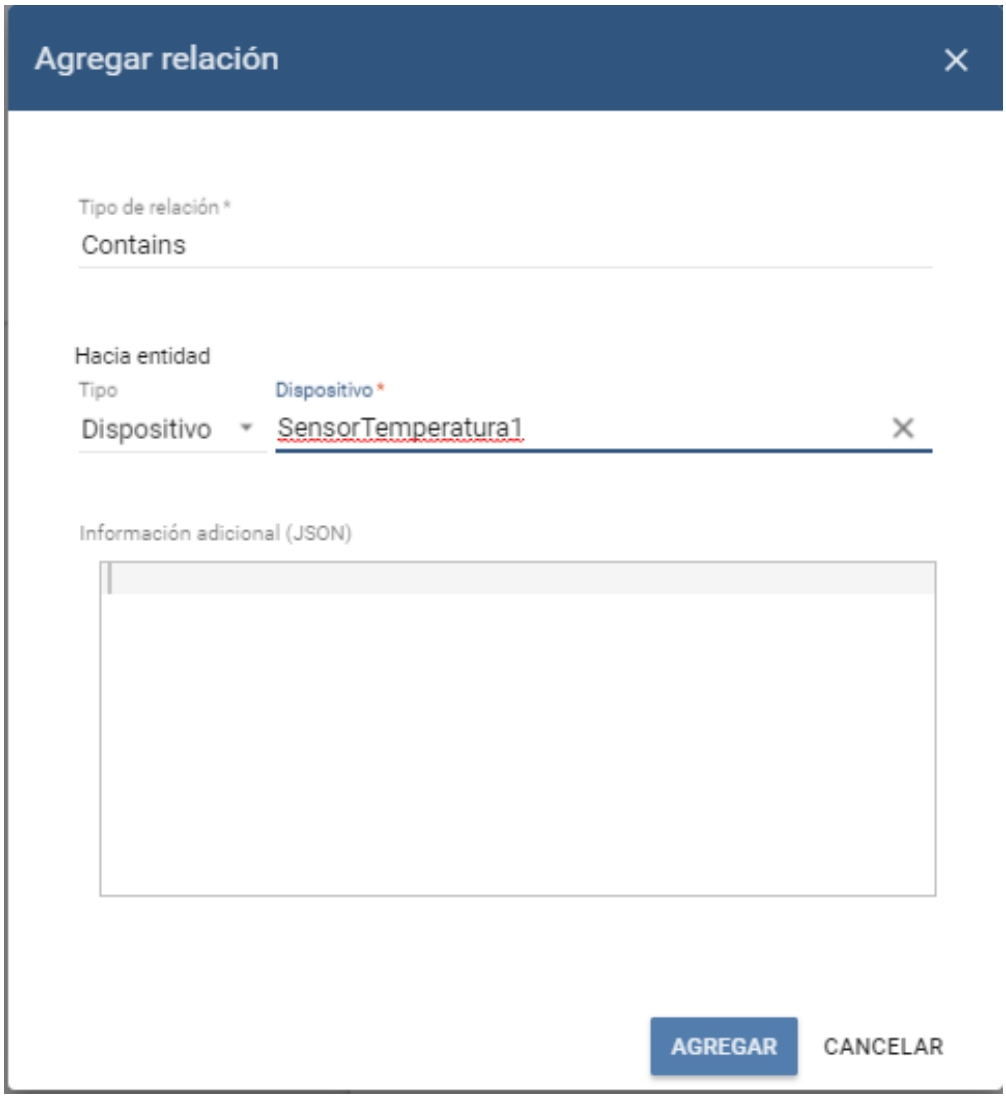

Ilustración 103: Agregar relación

El siguiente paso es actualizar los datos del dispositivo. Existe varias formas de hacerlo:

- Crear un cliente MQTT que envíe un mensaje de tipo PUBLICAR al tópico “v1/device/me/telemetry”.
- Crear un cliente CoAP que haga una petición POST a la siguiente url: *coap://host/api/v1/$ACCESS_TOKEN/telemetry*
- Crear un cliente HTTP que haga una petición POST a la siguiente URL: *http(s)://host:port/api/v1/$ACCESS_TOKEN/telemetry*

En este caso se ha decantado por la opción del cliente MQTT. Para ello, se crea el script "cliente_MQTT_TB.sh". Este script hace uso de Mosquitto[33], un bróker de mensajes de código abierto que implementa el protocolo MQTT. De esta forma es posible publicar la lectura del sensor, que recibe como parámetro en formato JSON con la siguiente estructura:

- **timestamp:** marca de tiempo generada en el momento en el que se realice la lectura.
- **sala**: nombre de la sala en la que está ubicado el sensor. Para esta pequeña prueba se usará la dirección del sensor.
- **temperatura**: temperatura (ºC) obtenida por el sensor.

```
1  {
2     "timestamp":1234,
3     "sala":sensor_addr,
4     "temperatura":t
5  }
```

Ilustración 104: Estructura archivo JSON

Además del JSON con los datos, es necesario indicar la dirección del host donde se encuentra la instancia de Thingsboard y el AccessToken que Thingsboard tiene asignado a ese dispositivo.

El script "cliente_MQTT_TB.sh" no será lanzado directamente, sino que se lanzará cada vez que se reciba una lectura. Por esta razón, es necesario modificar el script "temperature_sem.py", para que cada vez que se reciba una notificación del sensor con una nueva lectura de temperatura, se publique utilizando el script "cliente_MQTT_TB". Esto se implementa en el script "temperature_TB.py".

Se prueba a ejecutarlo y se puede comprobar que se actualiza la información del sensor. Esta información se encuentra en la pestaña "ÚLTIMA TELEMETRÍA". Se puede comprobar que los datos mostrados al ejecutar el script (Ilustración 105) coinciden con los mostrados en Thingsboard (Ilustración 106).

[33] Mosquitto

```
pi@raspberrypi:~/Desktop/Scripts/thingsboard $ sudo python3 temperature_TB.py fb:6a:9b:f2:3f:15
Connected to fb:6a:9b:f2:3f:15
Notification enabled
Waiting for notification...
Notification received from fb:6a:9b:f2:3f:15: 24.9°C
Client mosqpub|2118-raspberryp sending CONNECT
Client mosqpub|2118-raspberryp received CONNACK (0)
Client mosqpub|2118-raspberryp sending PUBLISH (d0, q0, r0, m1, 'v1/devices/me/telemetry', ... (78 bytes))
Client mosqpub|2118-raspberryp sending DISCONNECT
Notification disabled
Disconnected from device fb:6a:9b:f2:3f:15
```

Ilustración 105: Salida script temperature_TB.py

☐	2020-06-07 19:02:50	sala	fb:6a:9b:f2:3f:15
☐	2020-06-07 19:02:50	temperatura	24.9
☐	2020-06-07 19:02:50	ts	1.5915493699892251E9

Ilustración 106: Última telemetría sensor

Para poder visualizar datos recogidos, es necesario crear un widget donde se muestre esta información. En este caso, será una tabla con las temperaturas. Para crearlo, se accede desde la consola del administrador a "Paneles", y se crea un panel para el departamento de informática como se muestra en la Ilustración 107. Este panel contendrá todos los gráficos que quieran visualizarse para el sensor que creado anteriormente.

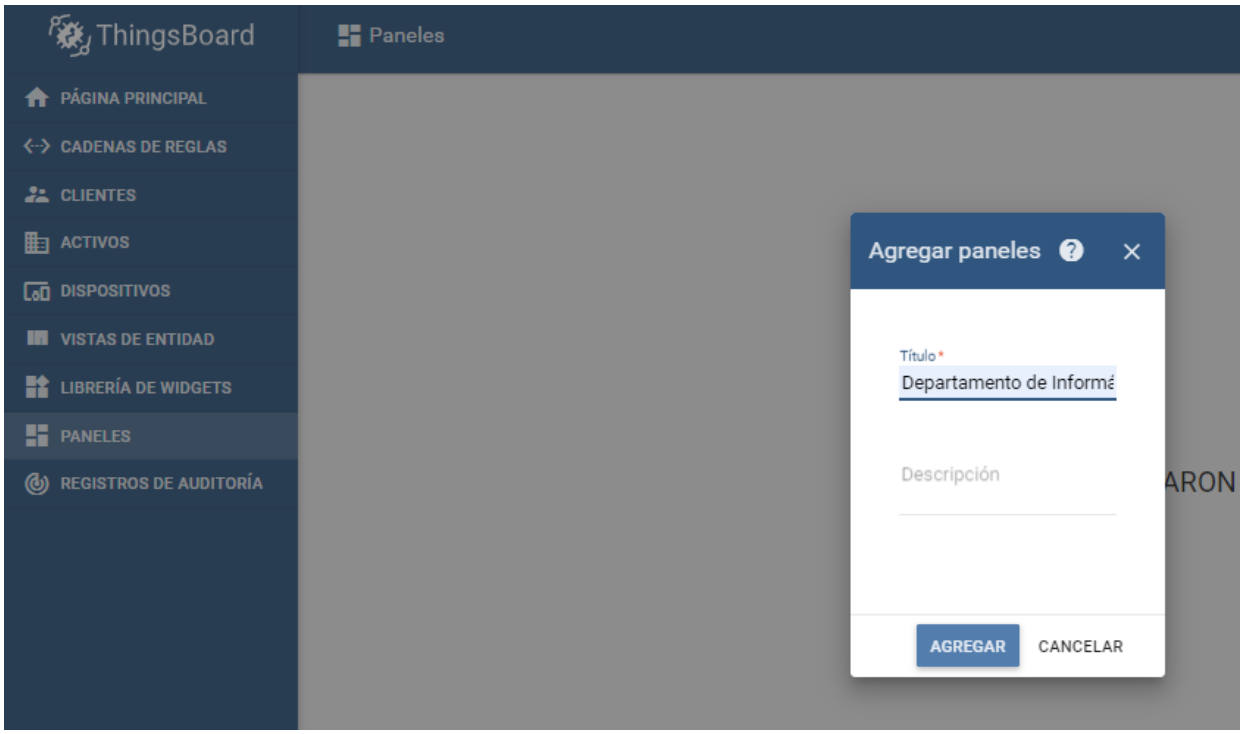

Ilustración 107: Agregar panel

Se puede ver que todavía no existen widgets que monitoricen la actividad. Por tanto, se crea uno donde se muestre las temperaturas medidas por el sensor.

Ilustración 108: Widgets del panel

Antes de añadir un widget, hay que crear un alias para la entidad “SensorTemperatura1”, para indicarle al widget que se elija más adelante de que entidad se desea mostrar los datos. Para ello se utiliza el botón “Alias de las entidades”

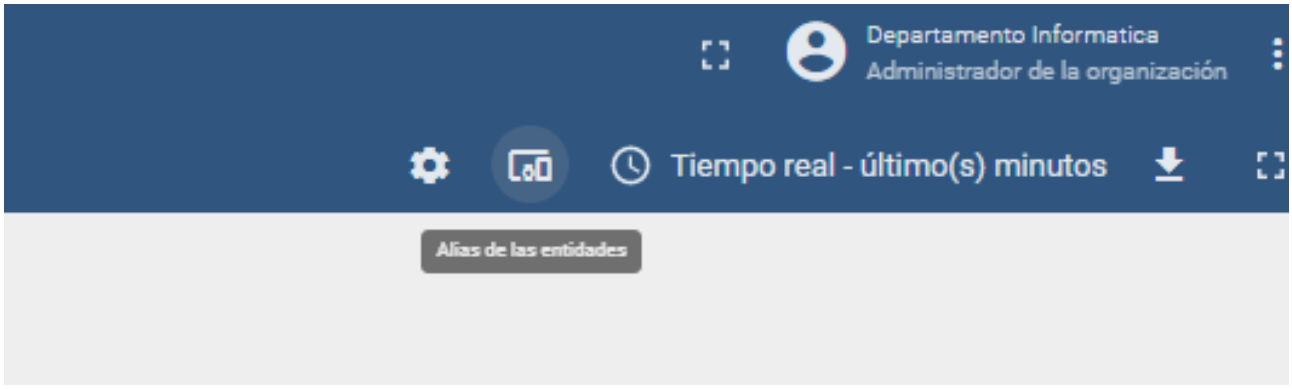

Ilustración 109: Botón Alias de las entidades

Aparece una ventana como la siguiente, donde se debe especificar el nombre del alias y la entidad a la que hace referencia. En este caso el sensor de temperatura.

Ilustración 110: Ventana para crear nuevo alias

Tras definir el alias, se crea un nuevo widget y se agrega en "origen de datos" la entidad "SesnsorTemperatura1". De entre las 3 opciones de datos a mostrar (timestamp, sala, temperatura), se elige "temperatura", que es la que contiene el valor de la temperatura medido en grados centígrados.

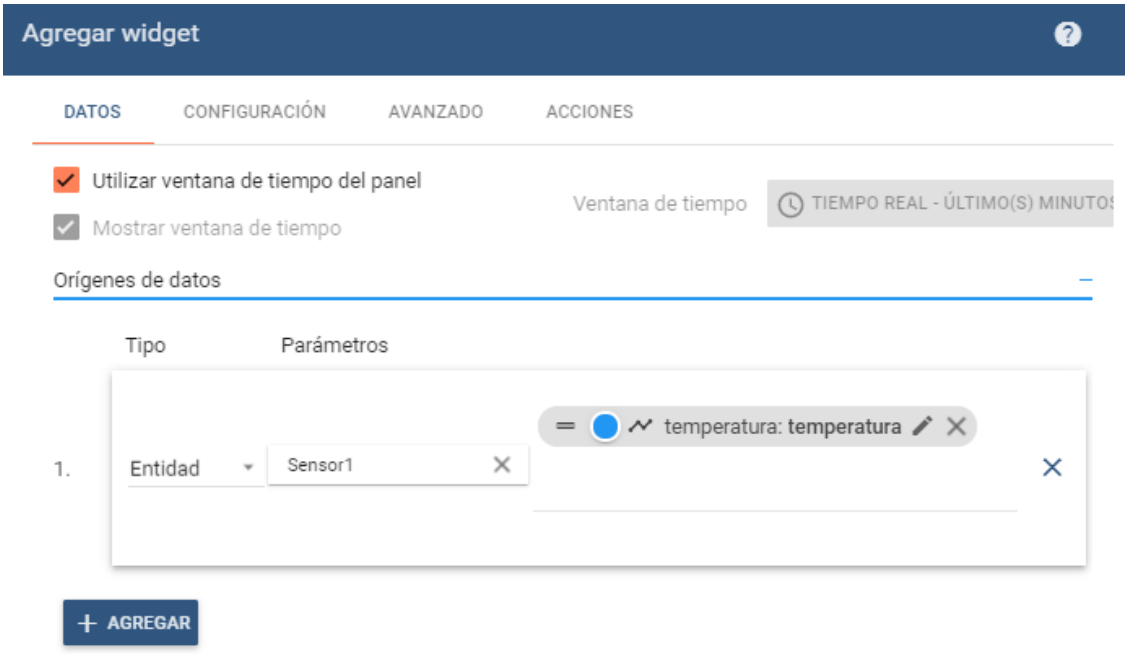

Ilustración 111: Ventana para agregar datos al widget

Se selecciona el botón “Agregar”, y es posible visualizar el gráfico, el cual se muestra a continuación:

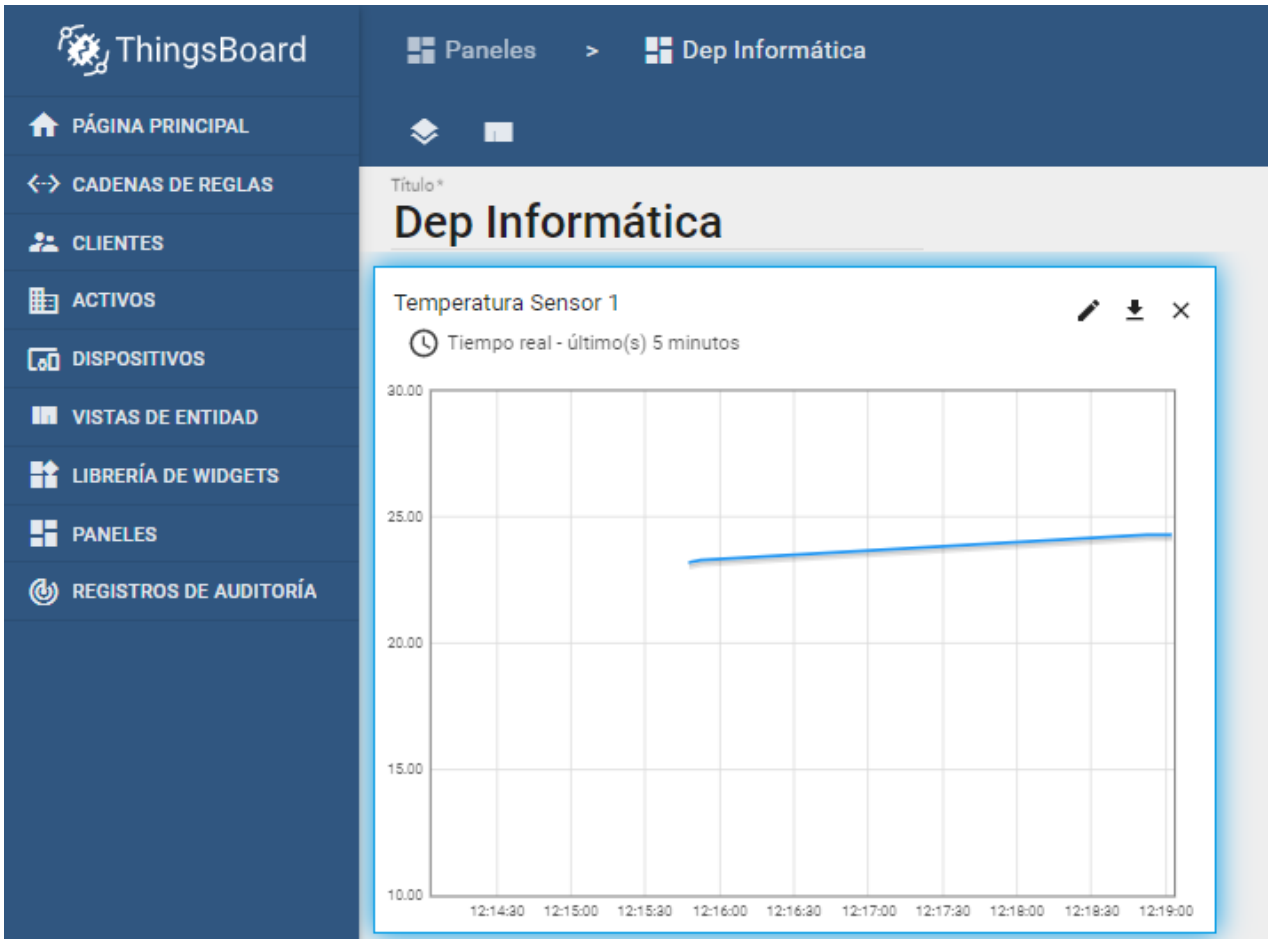

Ilustración 112: Gráfico temperatura sensor

6 Prototipo de monitorización de temperatura y humedad en un edificio

Con el fin de aplicar los conocimientos adquiridos durante el desarrollo del proyecto, se construye el prototipo de aplicación IoT basada en fog computing "Prototipo de monitorización de temperatura y humedad en un edificio", que permite monitorizar la temperatura y la humedad de varios espacios de la sede que tiene en Gijón el Departamento de Informática.

La recogida de datos es llevada a cabo por los nodos multisensor Steval- MKSBOX1V1 que forman parte de la red IoT. Como ya se ha mencionado en el apartado **DESARROLLO DEL FIRMWARE**, se realizan lecturas de temperatura y humedad cada 6 minutos.

Cada vez que un nodo realice una lectura, esta es enviada a la nube. La opción elegida ha sido usar una nube privada. Para ello, se crea una máquina virtual en los servidores de la Universidad donde se ejecuta una instancia de Thingsboard.

En Thingsboard se realiza una configuración idéntica a la del subapartado **Ejemplo nube privada** , pero añadiendo dos sensores más. En concreto, se ha dado de alta un edificio, "Edificio Departamental", y tres sensores, "Sensor1", "Sensor2" y "Sensor3". El edificio y los sensores están relacionados con una relación donde se indica que el "Edificio Departamental" contiene a los tres sensores.

Para la comunicación entre dispositivos en el borde y la nube se utiliza una RPi que se comunica con el sensor a través de bluetooth. Para ello se ha modificado el script "temperature_TB.py" utilizado en el subapartado **Ejemplo nube privada**. Este script llamado "m_temp_DI.py", recibe como parámetro un fichero donde se indican las direcciones de los sensores a los que debe conectarse junto a la sala donde se ubican.

En caso de querer sustituir un nodo, sería necesario detener la ejecución del script "m_temp_DI.py" y modificar este fichero sustituyendo la dirección del nodo reemplazado por la del nuevo. Si lo que se desea es añadir un nuevo nodo, además de introducir su dirección en este nuevo fichero, sería necesario darlo de alta en Thingsboard y añadir el token de acceso al fichero "tokens.txt"

```
cc:8d:73:24:f7:6f,SalaPostgrados
c6:08:41:22:75:39,SalaCafe
ed:0c:99:a8:48:db,SalaReuniones
```

Ilustración 113: Fichero de direcciones y ubicaciones

Cuando se lanza el script, se establece una conexión con cada dispositivo y se obtienen los datos de temperatura y humedad. A continuación, se envían los datos recibidos a la nube, haciendo uso del script "cliente_MQTT_MTDI.sh", el cual crea un cliente MQTT que actualiza la telemetría en Thingsboard para cada sensor. Seguidamente, finaliza la conexión con el sensor. Antes de volver a conectar con el sensor, ejecuta la función sleep(), que detiene la ejecución durante un tiempo. De este modo, la RPi espera hasta que el nodo multisensor vuelve a estar disponible. Esto se realiza en la función "conn", la cual se muestra en la siguiente ilustración:

```
 94
 95   #Metodo que se encarga de realizar la conexion con el dispositivo via BLE
 96   #Una vez establecida llama a la funcion getTemperature
 97   def conn(addr):
 98           #Conexion con el dispositivo
 99       p=btle.Peripheral(addr,btle.ADDR_TYPE_RANDOM)
100       print("\nConectado a  " + p.addr)
101
102           #Indicamos que objeto Delegate queremos que sea llamado cuando se
              notificacion
103       p.setDelegate(MyDelegate(p.addr))
104       while(1):
105               #Obtenemos la temperatura y humedad
106           p.setDelegate(MyDelegate(p.addr))
107           getTemperature(p)
108           #Desconexion con el dispocitivo
109           p.disconnect()
110           #Esperamos hasta que el sensor se encienda
111           time.sleep(360)
112           #Conectamos con el sensor
113           p.connect(p.addr,btle.ADDR_TYPE_RANDOM)
```

Ilustración 114: Script m_temp_DI.py

Para que la aplicación sea de utilidad para los usuarios que deseen conocer la temperatura o humedad de los espacios donde se encuentren ubicados los sensores, es necesario transformar los datos en gráficos. Para esto, se hace uso de Grafana.

Se complementa el uso de Grafana con la creación de una página web que facilite el acceso de los usuarios a los datos. La web está albergada en la máquina virtual mencionada anteriormente, donde también tendremos en ejecución Nginx, realizando la labor de servidor web HTTP. Esta web permite a los usuarios ver un listado de los espacios de los que se está monitorizando la temperatura y la humedad, mostrando el último valor de temperatura y de

humedad capturado. También se les proporciona la opción de acceder al histórico de cada uno de los espacios. Más en concreto, pueden visualizar 10 gráficos:

- Último valor de temperatura
- Último valor de humedad
- Temperatura media de las últimas 24 horas
- Humedad media de las últimas 24 horas
- Temperatura de las últimas 24 horas
- Humedad de las últimas 24 horas
- Temperatura del último mes
- Humedad del último mes
- Temperatura del último año
- Humedad del último año

Se utiliza la configuración empleada en el apartado **3.10 NGINX,** sobre la cual se han realizado las ampliaciones necesarias. En concreto, se modifica el archivo "index.html", se crean tres nuevos archivos detalle1.html, detalle2.html y detalle3.html para mostrar los datos de los 3 espacios en los que se va a monitorizar la temperatura y la humedad y se modifica el archivo de configuración "temperatura_aulas.com". Su contenido es el siguiente:

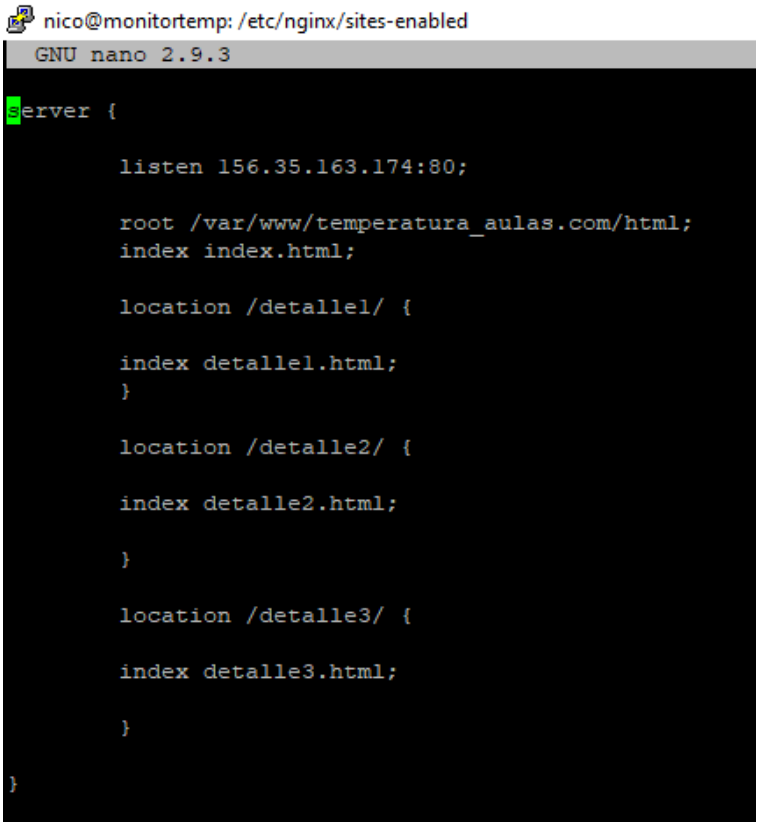

Ilustración 115: archivo temperatura_aulas.com

Una vez desarrollados todos los scripts y configuraciones necesarias, se procede a verificar el funcionamiento del prototipo. Para ello se realiza una pequeña prueba, en la que se recojan

varias medidas de temperatura y humedad utilizando 3 nodos multisensor y se comprueba que es posible visualizar los datos en la página web.

Una vez que los sensores se encuentran encendidos, se lanza el "script m_temp_DI.py" en la RPi, el cual produce la siguiente salida:

```
pi@raspberrypi: ~/Desktop/Scripts/SolFinal
pi@raspberrypi:~/Desktop/Scripts/SolFinal $ sudo python3 m_temp_DI.py fichero.txt

Conectado a  CC:8D:73:24:F7:6F

Conectado a  C6:08:41:22:75:39

Conectado a  ED:0C:99:A8:48:DB

Notificacion recibida de CC:8D:73:24:F7:6F: 26.8°C   70.3%

Notificacion recibida de C6:08:41:22:75:39: 27.1°C   61.8%

Notificacion recibida de ED:0C:99:A8:48:DB: 27.3°C   60.2%
```

Ilustración 116: Salida script m_temp_DI.py

Inmediatamente, se establece conexión con los nodos. Para comprobar que todo el proceso se realiza sin errores, se debe verificar que las lecturas mostradas en la Ilustración 116 son idénticas a las que se pueden visualizar en Thingsboard (Ilustración 117, Ilustración 118 e Ilustración 119).

Key ↑	Value
humedad	61.8
sala	SalaCafe
temperatura	27.1
ts	1.592838002634494 5E9

Ilustración 117: Última telemetría I

Key	Value
humedad	70.3
sala	SalaPostgrados
temperatura	26.8
ts	1.592838002338053 5E9

Ilustración 118: Última telemetría II

Key	Value
humedad	60.2
sala	SalaReuniones
temperatura	27.3
ts	1.592838003309491E9

Ilustración 119: Última telemetría III

Como se acaba de demostrar la temperatura leída por los sensores es almacenada en Thingsboard correctamente. Ahora se verifica que los últimos datos leídos que se presentan en la web encajen con la última telemetría almacenada en Thingsboard. En la siguiente ilustración se puede apreciar que los datos presentados en los gráficos de la página web muestran los mismos valores que la salida del script y la última telemetría almacenada en Thingsboard.

Ilustración 120: Últimos valores página web

Tras comprobar el correcto funcionamiento del prototipo, es posible hacer el despliegue en su ubicación final y comenzar a monitorizar la temperatura y la humedad del Departamento de Informática. Este despliegue y los resultados obtenidos se describen en el subapartado **Resultados obtenidos**.

6.1 RESULTADO OBTENIDOS

En este apartado se detalla cómo se ha llevado a cabo el despliegue de la aplicación "Prototipo de monitorización de temperatura y humedad en un edificio" en la sede que el Departamento de Informática tiene en el Campus de Gijón y se visualizan los datos obtenidos.

Los espacios en los que se decide ubicar los nodos multisensor, y por tanto de los que se monitoriza la temperatura y la humedad son la Sala de postgrado, la Sala de reuniones y la sala donde se encuentra la máquina del café o Sala de café.

Teóricamente, BLE permite establecer conexiones entre dispositivos separados hasta 100 metros, pero se ha comprobado que esto solo se produce en condiciones ideales, ya que cuando el gateway y un nodo multisensor se encuentran a más de 10 metros de distancia no es posible establecer una conexión BLE entre ambos. Para lidiar con esta restricción, ha sido necesario buscar la distribución ideal de modo que los nodos pudieran comunicarse con la RPi.

Es necesario colocar la RPi en un espacio cerrado para prevenir posibles robos y que disponga de toma de corriente. Antes de encontrar la ubicación definitiva fue necesario realizar pruebas de conexión en distintos puntos. Incluso se barajó la posibilidad de situarla alejada de una toma de corriente, alimentada con una batería portátil recargable. Finalmente se ubicó en la Sala de reprografía. Este espacio es el más indicado, ya que es posible la comunicación con los 3 nodos, existe una toma de corriente y es una sala poco transitada y normalmente cerrada con llave.

Ilustración 121: Ubicación RPi

Las ubicaciones de los sensores en las salas nombradas anteriormente son las siguientes:

- **Sala de reuniones**

Ilustración 122: Ubicación en Sala de reuniones

- **Sala de postgrado**

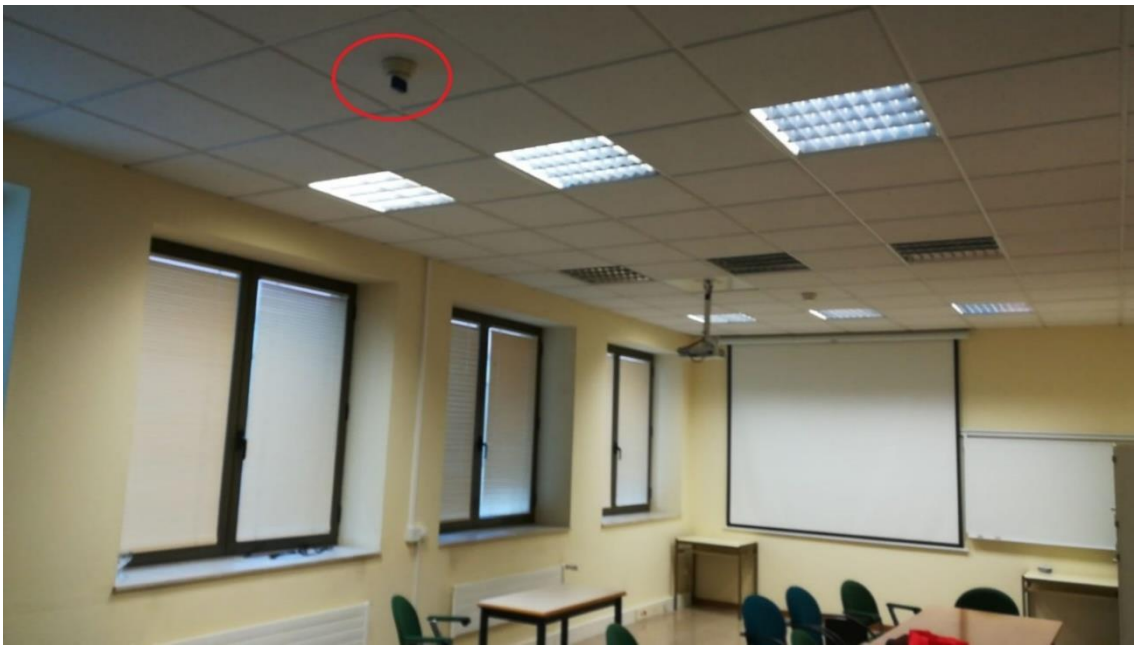

Ilustración 123: Ubicación en Sala de postgrados

- **Sala del café**

Ilustración 124: Ubicación en Sala del café

La ubicación de todos los espacios mencionados anteriormente puede verse en la siguiente ilustración siendo:

- Sala de reprografía: despacho 1.1.02
- Sala de café: vestíbulo
- Sala de Postgrado: Sala de Postgrado (1.1.25)
- Sala de reuniones: sala ubicada en la planta 2 encima de los despachos 1.1.23 y 1.1.24

Ilustración 125: Plano de la primera planta del edificio departamental Este

Una vez que han sido colocados, se procede a la recopilación de datos para poder visualizar y comparar los valores de temperatura y humedad de los 3 espacios.

Analizando los datos recogidos hasta la fecha se puede comprobar que la temperatura y la humedad tienen unos valores similares en los tres espacios monitorizados. En la siguiente ilustración se puede comprobar que la temperatura en esta época del año suele rondar los 26 ºC, mientras que la humedad se encuentra entre un 60% y un 70% de humedad relativa.

Ilustración 126: Temperatura y humedad media

En la siguiente ilustración se pueden ver los valores de temperatura que se recogieron en la Sala de café durante 24 horas. Se puede apreciar una caída de temperatura, producida por un cambio en la climatología, que produjo a su vez una bajada de temperaturas. El resto del tiempo, la temperatura se mantiene constante.

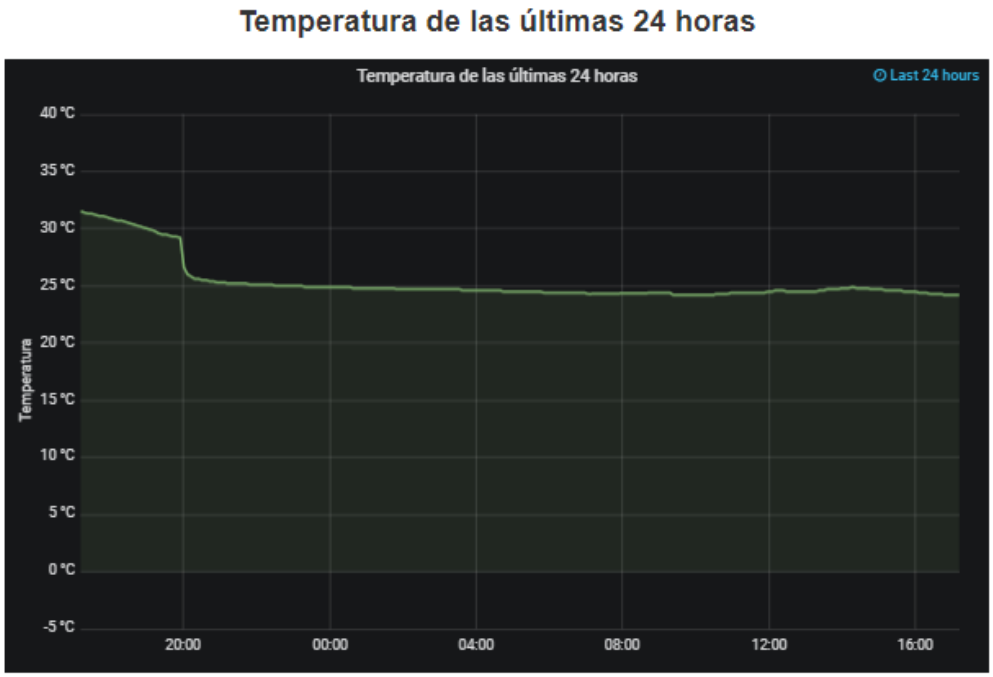

Ilustración 127: Temperatura últimas 24 horas de la Sala de café

A su vez, vemos que este cambio en la climatología produjo un aumento del porcentaje de humedad. Al igual que en el gráfico de temperatura, el resto del tiempo la humedad se mantiene constante, en torno a un 70%.

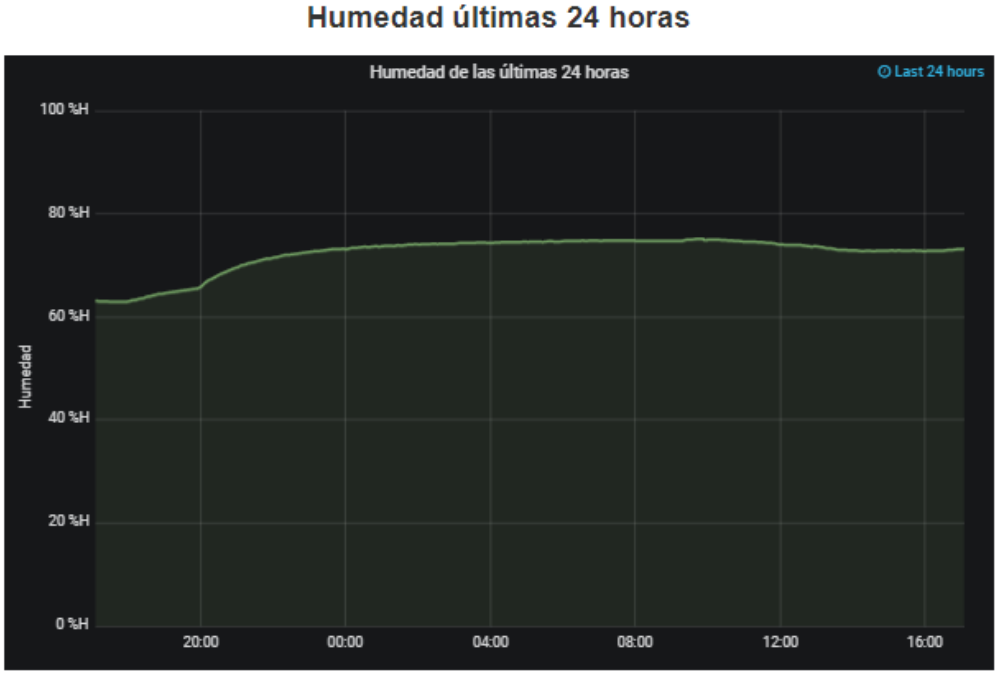

Ilustración 128: Humedad últimas 24 horas de la Sala de café

7 Anexos

7.1 PLANIFICACIÓN TEMPORAL

En la siguiente ilustración se puede apreciar la planificación temporal realizada, donde se detalla la duración estimada de cada una de las tareas en las que se ha dividido el proyecto. La previsión se ha realizado teniendo en cuenta que se dedica a la realización del proyecto una media de 3 horas diarias y que el TFG tiene un peso de 12 créditos ECTS que equivale a 300-360 horas de formación.

	Nombre	Duración	Inicio	Terminado
1	Estudio de la documentación	20 days	15/01/20 8:00	11/02/20 17:00
2	Configuración inicial del módulo	3 days	12/02/20 8:00	14/02/20 17:00
3	Configuración del gateway	15 days	17/02/20 8:00	6/03/20 17:00
4	Desarrollo del firmware	20 days	9/03/20 8:00	3/04/20 17:00
5	Actualización del firmware	5 days	6/04/20 8:00	10/04/20 17:00
6	Ejemplo de uso de nube pública	10 days	13/04/20 8:00	24/04/20 17:00
7	Ejemplo de uso de nube privada	10 days	27/04/20 8:00	8/05/20 17:00
8	Desarrollo del prototipo	15 days	11/05/20 8:00	29/05/20 17:00
9	Implantación	5 days	1/06/20 8:00	5/06/20 17:00
10	Documentación	109 days	15/01/20 8:00	15/06/20 17:00

Ilustración 129: Planificación Temporal

El diagrama de GANTT de la planificación anterior es el siguiente:

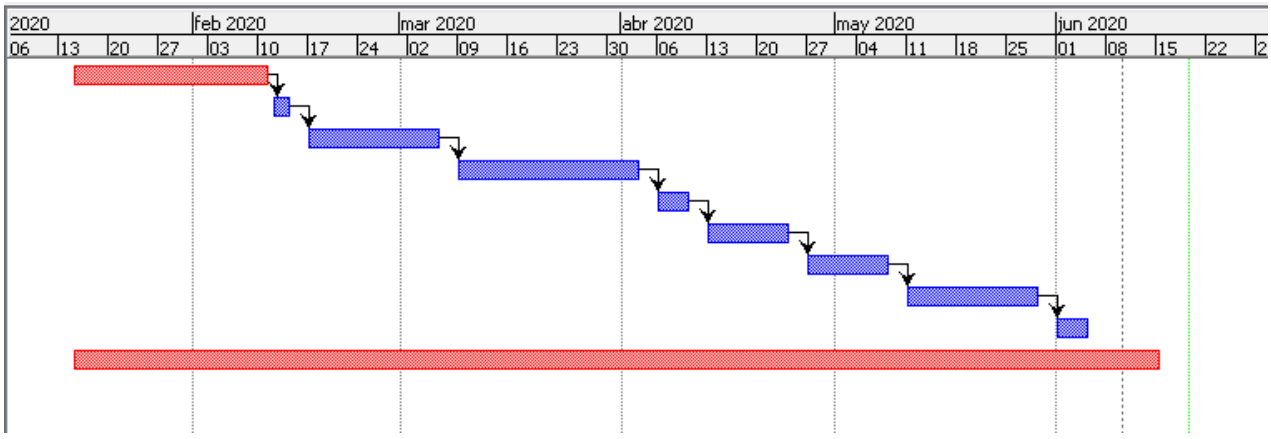

Ilustración 130: Diagrama de GANTT

8 Conclusiones y trabajo futuro

A la vista de los objetivos planteados al inicio del proyecto y los resultados obtenidos, puede concluirse que estos se han cumplido.

En lo referido a dispositivos en el borde, se ha adquirido conocimiento sobre algunas de las tecnologías existentes, que puede servir como base para otros proyectos. Se ha realizado el desarrollo de un firmware que se adaptase a las necesidades del proyecto y se ha configurado una RPi como gateway. Además, se han realizado con éxito comunicaciones entre sensores y gateway.

En lo relativo a la nube, se han desarrollado aplicaciones de ejemplo tanto en nube pública como en nube privada. En ambos casos se ha logrado transmitir los datos leídos por los sensores y almacenarlos en la nube, independientemente del tipo de nube, pública o privada. Esto puede servir como base para otros proyectos que usen Amazon AWS o Thingsboard, y también para proyectos que se decanten por utilizar otros proveedores de *Cloud Computing*, ya que la configuración que estos requieren es muy similar.

El último objetivo era el de aplicar todos los conocimientos adquiridos durante el desarrollo del proyecto para crear un prototipo de red IoT basada en *Fog Computing*. Como se puede comprobar en el subapartado **RESULTADOS OBTENIDOS**, se ha realizado con éxito. El prototipo "Prototipo de monitorización de temperatura y humedad en un edificio" recoge las temperaturas y la humedad de ciertos espacios del departamento, los cuales pueden ser visualizados con facilidad y claridad desde la web. Este prototipo también permite añadir más sensores o gateways para captar datos de otros espacios en el mismo edificio o en otro.

En cuanto a mi interés personal, he podido iniciarme en el mundo del IoT, que creo que proporcionará muchas oportunidades laborales en un futuro cercano. Todo el conocimiento adquirido me proporciona una cierta ventaja sobre otros compañeros de promoción, ya que la gran mayoría de los aspectos con los que hay que lidiar como la programación de microcontroladores, el uso de APIs específicas, la configuración de una RPi como gateway u otros no son temas que se aprendan durante el Grado.

Este proyecto puede ser continuado en un futuro. Las posibles tareas o mejoras para realizar en un futuro pueden dividirse en dos grupos: a corto y a largo plazo

A corto plazo:

- Utilizar otros protocolos de comunicación entre dispositivos en el borde para amplificar el rango de comunicación entre ellos. Ya sea portando a los sensores utilizados actualmente otros sistemas operativos como Contiki-NG, o utilizando nodos que cuenten con la capacidad de comunicarse a través de otros protocolos.
- Aumentar el número de sensores que forman la red IoT para analizar cómo influye el crecimiento de nodos en la latencia de las comunicaciones.

- Implementar una nueva funcionalidad que envíe mensajes de alerta cuando la temperatura de una sala sea demasiado baja o elevada, de modo que el personal de la universidad pueda realizar las acciones pertinentes para regular la temperatura.
- Añadir una funcionalidad que muestre el porcentaje de batería actual de cada sensor, de modo que se pueda recargar la batería o sustituir el nodo por otro totalmente cargado antes de que se agote. Así se evitará o se reducirá el número de lecturas no realizadas.

A medio plazo:

- Debido a los cambios de temperatura que pueden producirse en las distintas estaciones del año, sería interesante disponer de una base de datos completa, donde se recojan datos de todo un año. De esta manera podría analizarse si la temperatura y la humedad de estos espacios es la adecuada durante todo el año o no. Principalmente, para ver si en aquellos meses del año en los que la temperatura es más baja se está realizando un uso adecuado de la calefacción y tratar así de reducir ese importante coste.
- Integrar un nuevo módulo que permita el control de los dispositivos que se encargan de regular la temperatura como radiadores o equipos de aire acondicionado. Este módulo debería permitir tanto el control manual desde la web por usuarios autorizados como el control automático en función de las alertas generadas por temperatura elevada o temperatura baja.

9 Bibliografía

(Enero 2020). Obtenido de ¿Que es IoT? Página web de Oracle donde se describe qué es el IoT y otros temas relacionados.

https://www.oracle.com/es/internet-of-things/what-is-iot.html

(Enero 2020). Obtenido de Birth of IoT. Página web donde se hace un breve repaso del IoT desde su nacimiento.

https://iot-analytics.com/internet-of-things-definition/

(Mayo 2020). Obtenido de IoT statistics. Página web de la empresa vXchnge donde se proporcionan datos del número de dispositivos IoT actuales y en un futuro.

https://www.vxchnge.com/blog/iot-statistics

(Mayo 2020). Obtenido de the IoT rundown for 2020. Artículo de la web Security Today donde se hace un resumen del IoT en 2020.

https://securitytoday.com/Articles/2020/01/13/The-IoT-Rundown-for-2020.aspx?Page=2

(Mayo 2020). Obtenido de Types of cloud computing. Página web de Microsoft Azure donde se describe el cloud computing.

https://azure.microsoft.com/en-us/overview/what-is-cloud-computing/

(Junio 2020). Obtenido de Edge Computing. Documento donde se describe qué es el edge computing, su evolución y sus ventajas.

https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8789742

(Mayo 2020). Obtenido de fog computing. Artículo de la web IT Professional Resource Center donde se describe el fog computing.

https://www.itprc.com/fog-computing/

(Enero 2020). Obtenido de Steval-MKSBOX1V1. Página web de STMicroelectronics donde se describe el Steval-MKSBOX1V1.

https://www.st.com/content/st_com/en/products/evaluation-tools/product-evaluation-tools/mems-motion-sensor-eval-boards/steval-mksbox1v1.html

(Enero 2020). Obtenido de BLE. Web del grupo empresarial ELT donde se describe BLE.

https://www.elt.es/ble-bluetooth-low-energy

(Enero 2020). Obtenido de Raspberry Pi 3 Model B+. Página oficial de Raspberry donde puede encontrarse información de los diferentes productos ofertados.

https://www.raspberrypi.org/

(Enero 2020). Obtenido de ST Link V2. Página web de STMicroelectronics donde se describe el ST Link V2.

https://www.st.com/en/development-tools/st-link-v2.html

(Enero 2020). Obtenido de STM32CubeIDE. Página web de STMicroelectronics donde se describe el STM32CubeIDE.

https://www.st.com/content/st_com/en/products/development-tools/software-development-tools/stm32-software-development-tools/stm32-ides/stm32cubeide.html

(Enero 2020). Obtenido de STM32CubeProgrammer. Página web de STMicroelectronics donde se describe el STM32CubeProgrammer.

https://www.st.com/content/st_com/en/products/development-tools/software-development-tools/stm32-software-development-tools/stm32-programmers/stm32cubeprog.html

(Marzo 2020). Obtenido de Amazon Web Services. Página web de AWS donde se describen sus diferentes productos.

https://aws.amazon.com/es/

(Marzo 2020). Obtenido de AWS IoT Greengrass. Página web de AWS IoT GreenGrass donde se describe su funcionamiento, características, beneficios …

https://aws.amazon.com/es/greengrass/

(Marzo 2020). Obtenido de DynamoDB. Página web de DynamoDB donde se describe su funcionamiento, características, beneficios …

https://aws.amazon.com/es/dynamodb/?nc2=type_a

(Abril 2020). Obtenido de Thingsboard. Página web de Thingsboard donde se describe el producto y sus características y beneficios.

https://thingsboard.io/

(Mayo 2020). Obtenido de Grafana. Página web de Grafana donde se describe el producto y sus características y beneficios.

https://grafana.com/

(Mayo 2020). Obtenido de Nginx. Página web de Nginx donde se describe el producto y sus características y beneficios.

https://docs.nginx.com/

(Enero 2020). Obtenido de ST BLE Sensor App. Ubicación de la app móvil en el Play Store.

https://play.google.com/store/apps/details?id=com.st.bluems

(Enero 2020). Obtenido de Bluepy. Ubicación del repositrio de Github donde se encuentra Bluepy.

https://github.com/IanHarvey/bluepy

(Enero 2020). Obtenido de Bluez. Página web oficial de la pila del protocolo Bluetooth para Linux.

http://www.bluez.org/

(Abril 2020).Obtenido de FreeRTOS. Página web de FreeRTOS donde se describe el producto, características, beneficios …

https://www.freertos.org/

(Abril 2020).Obtenido de Contiki-NG. Ubicación del repositrio de Github donde se encuentra Contiki-NG.

https://github.com/contiki-ng/contiki-ng

(Abril 2020). Obtenido de FreeRTOS ports. Página web de FreeRTOS donde se describen los dispositivos a los que se puede portar FreeRTOS.

https://www.freertos.org/RTOS_ports.html

(Abril 2020). Obtenido de STM32L4 Discovery Kit Node. Página web de STMicroelectronics donde se describe el STM32L4 Discovery Kit Node.

https://www.st.com/content/st_com/en/products/evaluation-tools/product-evaluation-tools/mcu-mpu-eval-tools/stm32-mcu-mpu-eval-tools/stm32-discovery-kits/b-l475e-iot01a.html

(Abril 2020). Obtenido de RFC7668. Documento del IETF donde se describe el protocolo IPV6 sobre BLE.

https://tools.ietf.org/html/rfc7668

(Febrero 2020). Obtenido de Guía para portar Contiki-NG. Guía para portar Contiki-NG a otras plataformas.

https://github.com/contiki-ng/contiki-ng/wiki/Porting-Contiki%E2%80%90NG-to-new-platforms#platform-code

(Febrero 2020). Obtenido de BLEach. Página web donde se describe el protocolo BLEach.

http://spoerk.github.io/contiki/

(Mayo 2020). Obtenido de BLEach: Exploiting the Full Potential of IPv6 over BLE in Constrained Embedded IoT Devices. Documento donde se analiza el consumo de energía y la latencia de conexiones realizadas con el protocolo BLEach.

https://wwwpub.zih.tu-dresden.de/~mzimmerl/pubs/spoerk17bleach.pdf

(Marzo 2020). Obtenido de FP-SNS-STBOX1. Página web de STMicroelectronics donde se describe el paquete de funciones FP-SNS-STBOX1.

https://my.st.com/content/my_st_com/en/products/embedded-software/mcu-mpu-embedded-software/stm32-embedded-software/stm32-ode-function-pack-sw/fp-sns-stbox1.html

(Marzo 2020).Obtenido de aws-iot-device-sdk. Ubicación del repositrio de Github donde se encuentra aws-iot-device-sdk.

https://github.com/aws/aws-iot-device-sdk-python-v2

(Marzo 2020). Obtenido de manual instalación Thingsboard. Manual de instalación de Thngsboard en una Raspberry Pi 3 Model B.

https://thingsboard.io/docs/user-guide/install/rpi/

(Abril 2020). Obtenido de Mosquitto. Página web donde se describe el funcionamiento y las características del broker MQTT Mosquitto.

https://mosquitto.org/