SISTEMA DE MONITORIZACIÓN MEDIOAMBIENTAL CON LORA PARA EL BOSQUE METROPOLITANO DE MADRID

ENVIRONMENTAL MONITORING SYSTEM WITH LORA FOR THE METROPOLITAN FOREST OF MADRID

TRABAJO FIN DE MÁSTER
CURSO 2021-2022

AUTOR
CARLOS REJÓN GÓMEZ

DIRECTOR
JOSE IGNACIO GÓMEZ PÉREZ

MÁSTER EN INTERNET DE LAS COSAS
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

SISTEMA DE MONITORIZACIÓN MEDIOAMBIENTAL CON LORA PARA EL BOSQUE METROPOLITANO DE MADRID
ENVIRONMENTAL MONITORING SYSTEM WITH LORA FOR THE METROPOLITAN FOREST OF MADRID

TRABAJO FIN DE MÁSTER EN INTERNET DE LAS COSAS
DEPARTAMENTO DE ARQUITECTURA DE COMPUTADORES Y AUTOMÁTICA

AUTOR
CARLOS REJÓN GÓMEZ

DIRECTOR
JOSE IGNACIO GÓMEZ PÉREZ

CONVOCATORIA: SEPTIEMBRE 2022
CALIFICACIÓN: 10

MÁSTER EN INTERNET DE LAS COSAS
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

23 DE SEPTIEMBRE DE 2022

Dedicatoria

A la memoria de mi padre, Carlos Rejón Fernández, que sin su obstinado empeño, nunca hubiese continuado mis estudios universitarios.

"Allá donde estés, siempre te llevaré conmigo."

Resumen

SISTEMA DE MONITORIZACIÓN MEDIOAMBIENTAL CON LORA PARA EL BOSQUE METROPOLITANO DE MADRID

En la actualidad, los gobiernos y organizaciones están cada vez más concienciados con el cambio climático y sus terribles consecuencias para la sociedad, por lo que se están llevando a cabo proyectos de muy diversa índole para revertir o paliar estos cambios. Uno de ellos es el Bosque Metropolitano de Madrid, que contempla la creación de un anillo forestal alrededor de la ciudad, plantando cerca de medio millón de árboles, con la intención de mejorar la calidad del aire y reducir el efecto "isla de calor" que se produce en las grandes urbes como consecuencia de los edificios y el asfalto.

La finalidad de este proyecto es la creación de un nodo prototipo para la medición de parámetros ambientales en las zonas de actuación del Bosque Metropolitano, de manera que se puedan verificar los propósitos de su creación, demostrando cómo la reforestación de hábitats degradados beneficia a las ciudades y poblaciones cercanas. Al tratarse de ubicaciones en zonas aisladas, la alimentación del nodo se realiza de forma autónoma mediante una placa solar, en conjunto con una batería, y la comunicación inalámbrica se lleva a cabo mediante LoRaWAN. Los mensajes salientes con la información de los sensores son enviados a la plataforma AWS IoT, donde se decodifican y almacenan para su posterior análisis.

Palabras clave: Internet de las Cosas (IoT), monitorización ambiental, LoPy4, LoRaWAN, AWS IoT, Amazon Timestream, Grafana

Abstract

ENVIRONMENTAL MONITORING SYSTEM WITH LORA FOR THE METROPOLITAN FOREST OF MADRID

Nowadays, governments and organisations are increasingly aware of climate change and its terrible consequences for society, which is why a wide range of projects are being carried out to reverse or alleviate these changes. One of them is the Metropolitan Forest of Madrid, which envisages the creation of a forest ring around the city, planting almost half a million trees, with the intention of improving air quality and reducing the "heat island" effect that occurs in large cities as a result of buildings and asphalt.

The purpose of this project is the creation of a prototype node for measuring environmental parameters in the areas of action of the Metropolitan Forest, so that the purposes of its creation can be verified, demonstrating how the reforestation of degraded habitats benefits the cities and nearby populations. As the locations are in isolated areas, the node is powered autonomously by a solar panel, in conjunction with a battery, and wireless communication is carried out via LoRaWAN. Outgoing messages with sensor information are sent to the AWS IoT platform, where they are decoded and stored for further analysis.

Keywords: Internet of Things (IoT), environmental monitoring, LoPy4, LoRaWAN, AWS IoT, Amazon Timestream, Grafana

Índice de contenidos

Índice de figuras

1. Introducción

En este primer capítulo introductorio se expone la motivación del presente proyecto, los objetivos propuestos que deben ser cumplidos y la organización presentada en la memoria.

1.1 Motivación

Las ciudades mediterráneas son especialmente vulnerables al cambio climático según informes recientes de organismos internacionales que demuestran que el incremento de la temperatura en el área mediterránea está un 20% por encima de la media del planeta con riesgo de una subida de 3,5ºC para 2080 con implicaciones serias en materia de desertificación y pérdida de ecosistemas.

El ayuntamiento de Madrid actualmente está llevando a cabo un proyecto para la creación de un cinturón forestal que rodea la ciudad llamado Bosque Metropolitano. Supone un nuevo despliegue de infraestructura verde consistente en la plantación de especies forestales autóctonas en suelo municipal disponible, contribuyendo tanto a la mejora ambiental, como a la restauración ecológica y paisajística de zonas degradadas. Otras ciudades como Londres o Copenhague marcaron el camino en las décadas de 1930 y 1940. En España, tuvimos que esperar hasta 1993 a que Vitoria iniciase su Anillo Verde [1].

Esta corona forestal abarca 14.200 hectáreas y prevé la plantación de hasta 450.000 árboles de especies autóctonas, durante los próximos 12 años, lo que incorporará a la ciudad hasta 5.900 hectáreas netas más de arbolado (el equivalente a 51 veces El Retiro). Conectará las áreas naturales protegidas de El Pardo, al norte, con los cursos bajos del Manzanares y el Jarama, al sureste.

El Bosque se divide en cinco secciones diferenciadas por continuidad geográfica y planes de actuación, Figura 1.1. La primera en llevarse a cabo, y en la que se centra este trabajo, es la Corona Noroeste, donde la prioridad es conectar El Pardo con el monte de Valdelatas y con la Casa de Campo a través del río Manzanares [2].

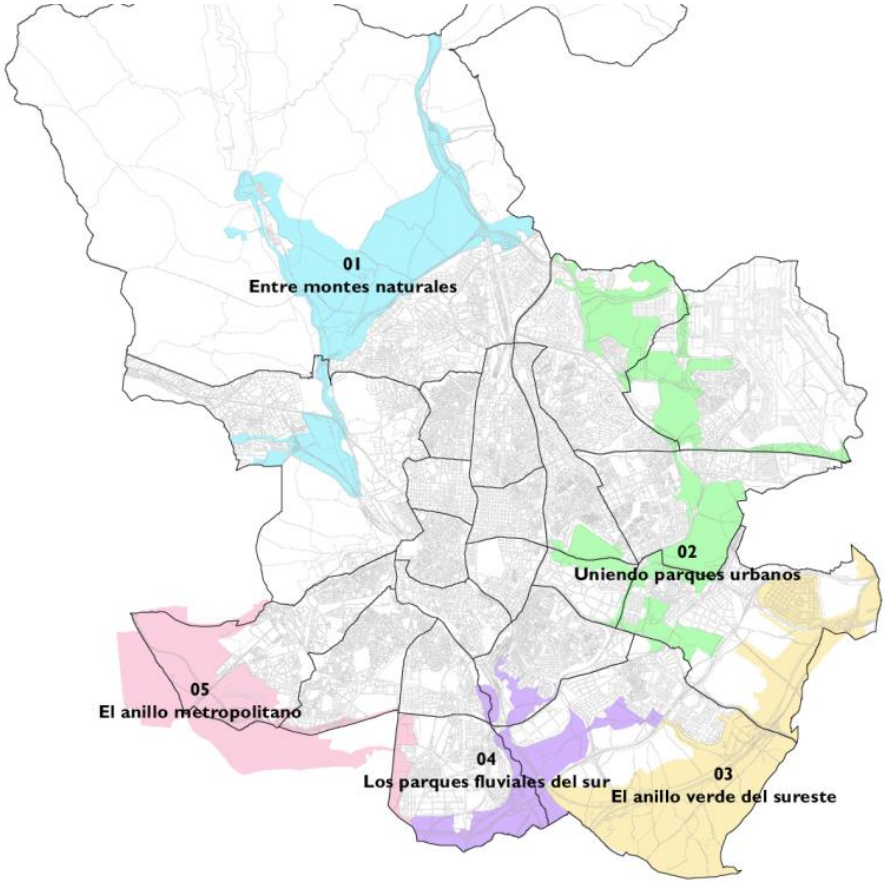


Figura 1.1 – Secciones del Bosque Metropolitano

Aunque el proyecto contempla la creación de senderos y áreas de ocio, su papel es secundario, ya que no se trata de un parque urbano. Su misión es mejorar la calidad del aire y disminuir el efecto "isla de calor" que se produce en Madrid durante los meses de verano debido a los edificios, el asfalto y las diferentes actividades humanas.

El desarrollo del Bosque Metropolitano aportará, entre otros, los siguientes beneficios ambientales y sociales:

- Mejora de la calidad del aire
- Mejora de la biodiversidad urbana
- Contribución a la mitigación y adaptación al cambio climático en la ciudad
- Fomento de la salud y bienestar de la ciudadanía mediante la promoción del ocio y las actividades deportivas en el medio natural
- Mejora del paisaje y puesta en valor de los espacios próximos consolidados y planificados

1.2 Objetivos

En el contexto actual de cambio climático, que lamentablemente se nos hace más visible año tras año, poder corroborar de forma plausible como la reforestación de zonas degradadas mejora el hábitat, y por tanto la calidad de vida de todas las personas que coexistan con ellas, puede incentivar a que se planteen políticas que vayan encaminadas a que más municipios adopten este tipo de medidas.

El objetivo principal del presente proyecto es el diseño, montaje y prueba funcional de un nodo autónomo capaz de verificar las metas medioambientales buscadas con la ejecución del Bosque Metropolitano de Madrid. De entre ellas podemos resaltar la disminución de la temperatura en épocas de calor, el aumento de la humedad relativa y la reducción de las partículas microscópicas en suspensión.

El sistema IoT que se pretende desarrollar deberá cumplir con las siguientes especificaciones:

- Medición de parámetros ambientales y calidad del aire
- Alimentación autónoma de los nodos
- Comunicación a larga distancia con baja tasa de transmisión
- Gestión y almacenamiento de datos

Para verificar el correcto funcionamiento, el nodo se situará en una ubicación exterior durante un periodo de un 1 mes y se estudiarán los datos obtenidos mediante gráficas temporales.

1.3 Organización de la memoria

El documento se compone de los apartados que se describen a continuación.

Este primer capítulo que sirve a modo introductorio, describiendo la motivación del proyecto y los objetivos a alcanzar.

En el segundo capítulo se dará una visión de las tecnologías y dispositivos que permiten desarrollar el sistema.

En el tercer capítulo nos centraremos en la creación de los nodos y la configuración de la plataforma cloud.

En el cuarto capítulo se exponen los resultados obtenidos mediante la creación de gráficas con los valores captados por los sensores.

En el quinto capítulo se presentarán las conclusiones del proyecto y los posibles trabajos futuros que se pueden llevar a cabo.

Por último, encontramos las referencias bibliográficas utilizadas.

2. Estado de la cuestión

En los siguientes apartados de este capítulo se hace una descripción de las tecnologías y dispositivos en los que se basa el actual proyecto.

2.1 Tecnología LoRa

LoRa es una tecnología inalámbrica que emplea un tipo de modulación en radiofrecuencia patentado por Semtech, una importante empresa fabricante de chips de radio. La tecnología de modulación se denomina *Chirp Spread Spectrum* (CSS) y se emplea en comunicaciones militares y espaciales desde hace décadas. Funciona en bandas ISM que no requieren licencia y están reservadas internacionalmente para fines industriales, científicos y médicos [3].

La transmisión modulada LoRa es resistente a las perturbaciones y se puede recibir a través de grandes distancias. Es ideal para aplicaciones que transmiten pequeños fragmentos de datos con tasas de bits bajas. Estas características hacen que sea ideal para dispositivos IoT que funcionan en modo de bajo consumo.

Sus características son:

- Alta tolerancia a las interferencias
- Alta sensibilidad para recibir datos (-168dB)
- Basado en modulación *Chirp*
- Bajo consumo (hasta 10 años con una batería)
- Largo alcance 10 a 20 km
- Baja transferencia de datos (hasta 255 bytes)
- Conexión punto a punto
- Frecuencias de trabajo: 868 Mhz en Europa, 915 Mhz en América, y 433 Mhz en Asia

LoRaWAN es un protocolo de capa de control de acceso a medios (MAC) para redes LPWAN (*Low Power Wide Area Network*) construido sobre la modulación LoRa. Es una capa de software que define cómo los dispositivos usan el hardware LoRa, por ejemplo, cuándo transmiten y el formato de los mensajes. Es desarrollado y mantenido por LoRa Alliance, asociación abierta sin fines de lucro establecida en 2015, que garantiza la interoperabilidad de todos los productos y tecnologías asociadas [4].

Una red LoRaWAN típica, Figura 2.1, consta de dispositivos finales (nodos), gateways (puertas de enlace), un servidor de red (llamado **LoRaWAN Network Server** o **LNS**) y servidores de aplicaciones donde se procesan los datos. Los dispositivos finales se comunican con puertas de enlace cercanas y cada puerta de enlace está conectada al servidor de red.

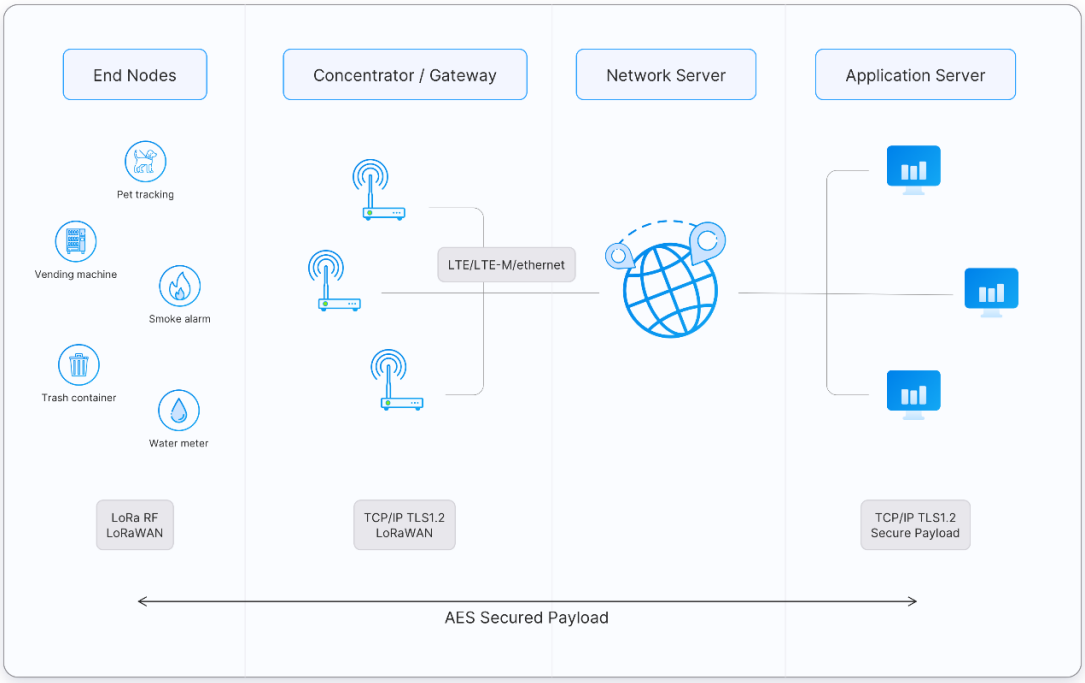


Figura 2.1 – Arquitectura LoRaWAN

Se utiliza un protocolo basado en ALOHA, por lo que los dispositivos finales no necesitan emparejarse con puertas de enlace específicas y los mensajes enviados viajan a través de todas las puertas de enlace dentro del alcance. Si el servidor de red ha recibido varias copias del mismo mensaje, conserva una sola copia del mensaje y descarta las demás. Esto se conoce como deduplicación de mensajes.

2.1.1 OTAA y ABP

Cada dispositivo final debe estar registrado en una red antes de enviar y recibir mensajes. Este procedimiento se conoce como activación y hay dos métodos disponibles:

- **Over-The-Air-Activation (OTAA)**: el método de activación más seguro y recomendado para dispositivos finales. Los dispositivos realizan un procedimiento de unión con la red, durante el cual se asigna una dirección de dispositivo dinámica y se negocian claves de seguridad con el dispositivo.

- **Activación por personalización (ABP)**: requiere codificar la dirección del dispositivo, así como las claves de seguridad en el dispositivo. ABP es menos seguro que OTAA y también tiene la desventaja de que los dispositivos no pueden cambiar de proveedor de red sin cambiar manualmente las claves en el dispositivo.

2.1.2 LoRa Basics Station

LoRa Basics Station [5] es una implementación de software para gestionar los paquetes LoRA, denominado ***packet forwarder*** e instalado en la puerta de enlace LoRaWAN, que proporciona funcionalidad central en términos de manejo del flujo de paquetes, administración del acceso al espectro y conectividad de la red de retorno (*backhaul*) LNS, entre otros aspectos. Para llevar a cabo estas tareas, la especificación de Basics Station define dos protocolos back-end:

- **Protocolo LNS**: es el plano de datos principal, que proporciona un canal de comunicación bidireccional de baja latencia a través de WebSockets seguros. Este protocolo incorpora aspectos de equilibrio de carga y gestión de la configuración centralizada.
- **Protocolo CUPS (*Configuration and Update Service*)**: proporciona una gestión de credenciales y una interfaz de actualización del firmware mediante transacciones HTTPS autentificadas.

2.2 Nodo para monitorización ambiental

En el mercado encontramos empresas que ofrecen soluciones IoT para monitorización ambiental basadas en comunicación LoRa. Un ejemplo es la empresa *Libelium*, que ofrece en su gama *Plug & Sense* el dispositivo ***Smart Enviroment***, capaz de captar parámetros ambientales (temperatura, humedad, presión atmosférica, luminosidad…), partículas en suspensión y hasta 10 tipos de gases distintos. También tiene la posibilidad de conectarse directamente a una placa fotovoltaica para obtener la alimentación necesaria [6].

Figura 2.2 – Smart Enviroment de Libelium

2.2.1 LoPy 4

Una placa de desarrollo que permite diseñar dispositivos similares al anterior es LoPy 4 de Pycom [7], basada en el chip SoC ESP32 de Espressif. Dispone de cuádruple conexión inalámbrica mediante LoRa, Sigfox, WiFi y Bluetooth.

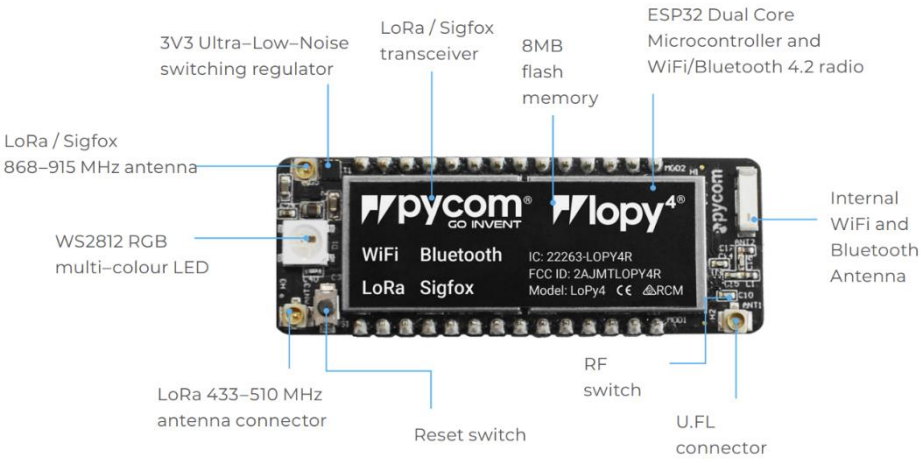


Figura 2.3 – LoPy 4

Al utilizar el chipset ESP32 podría programarse con FreeRTOS o incluso Arduino, pero viene configurada para ser utilizada con MicroPython [8], una implementación del lenguaje de programación Python 3, escrita en C, que incluye un pequeño subconjunto de la biblioteca

estándar y está optimizada para ejecutarse en microcontroladores y sistemas embebidos. Al ser un lenguaje interpretado, y no compilado como los mencionados al comienzo de este párrafo, su ejecución es más lenta, pero de momento solo se dispone en este lenguaje la librería necesaria para usar la conectividad LoRa.

2.3 Plataforma cloud IoT

Existen diversas compañías que ofrecen plataformas en la nube para la gestión de dispositivos IoT, de entre las que podemos destacar Google, Azure, Bosch, IBM Watson y **AWS**. Se ha elegido esta última por disponer de un servidor de red LoRaWAN (LNS) instalado por defecto en su solución IoT, además de permitir un año de suscripción gratuita para estudiantes universitarios, tiempo más que suficiente para realizar las pruebas necesarias en este proyecto.

2.3.1 AWS IoT Core

AWS IoT Core [9] permite conectar los dispositivos a servicios de AWS y a otros dispositivos, proteger datos e interacciones, procesar y actuar sobre los datos de dispositivos y habilitar las aplicaciones para que interactúen con dispositivos, incluso aunque no estén conectados.

En la figura 2.4 se puede ver el funcionamiento cuando se utiliza LoRaWAN y como las pasarelas se comunican con AWS IoT Core mediante el protocolo *LoRa Basic Station* sobre *WSS*, que es utilizado por el servidor LNS.

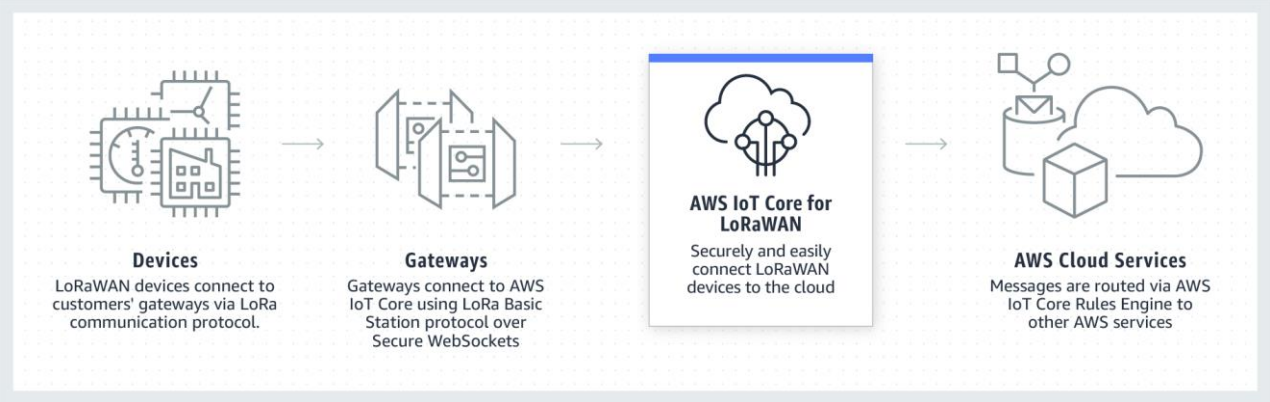


Figura 2.4 – AWS IoT Core

2.3.2 AWS TimeStream

De entre todas las opciones de almacenamiento disponibles en AWS, se ha optado por AWS Timestream [10] ya que se trata de una base de datos de series temporales, siendo éstas las que mejor se ajustan a los datos recogidos por sensores en los sistemas IoT.

Una BBDD de series temporales es un sistema de gestión de base de datos que está optimizado para el proceso de escritura, almacenamiento y consulta de las características de los datos de series de tiempo, siendo el índice la marca temporal de la medición.

2.3.3 Amazon Managed Grafana

Grafana es una plataforma de análisis open source que permite la visualización y el análisis de datos métricos. Permite crear cuadros de mando y gráficos a partir de múltiples fuentes y aplicaciones, incluidas bases de datos de series temporales.

AWS dispone de esta herramienta mediante el servicio Amazon Managed Grafana [11], que una vez configurada, permite la conexión con AWS Timestream.

3. Desarrollo

3.1 Implementación de los nodos

El nodo de medición ambiental diseñado se compone de los siguientes elementos de hardware que podrían separarse en cuatro grupos:

- **Elementos base**
  - LoPy 4
  - Expansion board
  - Batería LiPo
- **Alimentación fotovoltaica**
  - Placa solar
  - Cargador solar
- **Monitorización**
  - Sensor de luz IR, UV y visible
  - Sensor de temperatura y humedad relativa
  - Sensor de partículas en suspensión
  - Monitor de batería
- **Elementos auxiliares**
  - Convertidor elevador dc/dc
  - Reloj en tiempo real

En el siguiente apartado se realiza una descripción de los dispositivos utilizados y sus principales características técnicas.

3.1.1 Componentes

Lopy 4

Como se ha comentado con anterioridad, esta placa de desarrollo está basada en el chipset ESP32 al que se le ha añadido conectividad LoRa y Sigfox. Dispone de 24 GPIOs y 8 conversores analógicos, así como interfaces UART, I2C, I2S y SPI. Los distintos pines y sus funciones asignadas pueden verse en la figura 3.1.

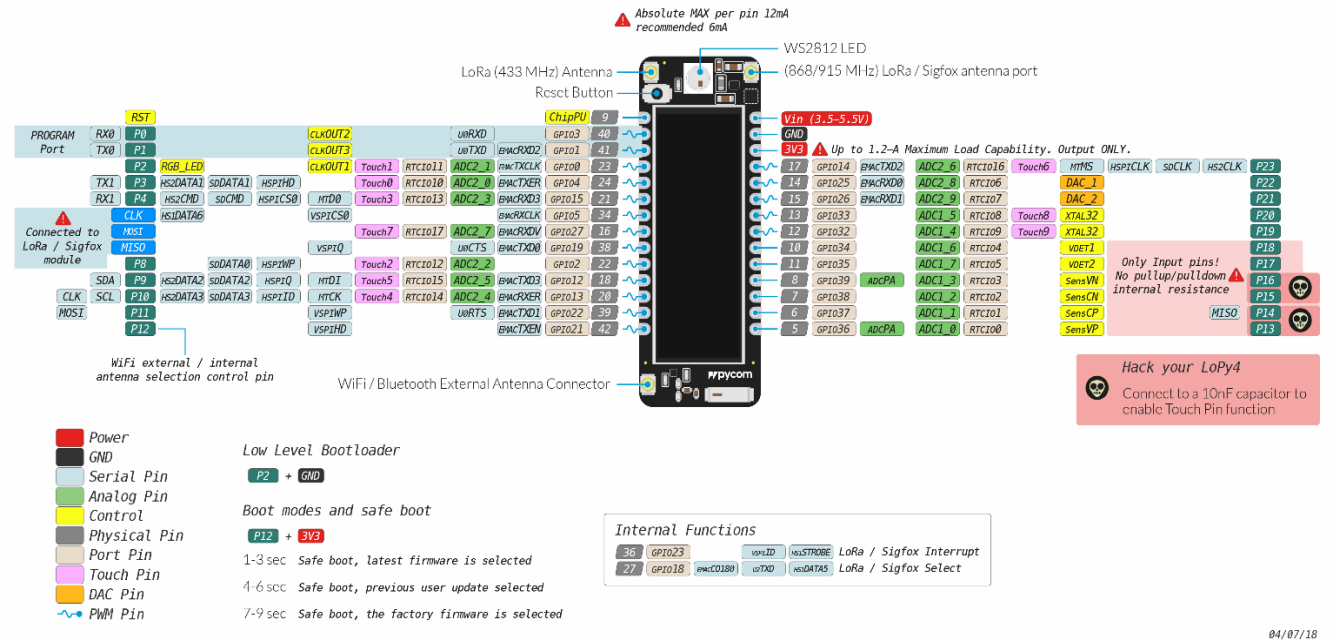


Figura 3.1 – LoPy 4 pinout [12]

Especificaciones técnicas:

- Microprocesador Xtensa dual–core 32–bit LX6
- RAM: 520KB + 4MB
- Memoria flash externa: 8MB
- Conectividad LoRa, Sigfox, WiFi y Bluetooth
- Alimentación: 3.5 ~ 5.5 V

Expansion Board [13]

Para facilitar la conexión mediante USB a un ordenador, el acceso a los GPIOs y la conexión de la batería se añade a la placa LoPy 4 un módulo de expansión, Figura 3.2, que además permitiría añadir una tarjeta de memoria MicroSD en el caso de que fuera necesaria.

Especificaciones técnicas:

- Alimentado por USB y batería LiPo
- Ranura para tarjeta MicroSD
- Conector de batería estilo JST
- Cargador de batería BQ24040
- Distribuidor de alimentación TPS2115A con protección contra voltaje inverso

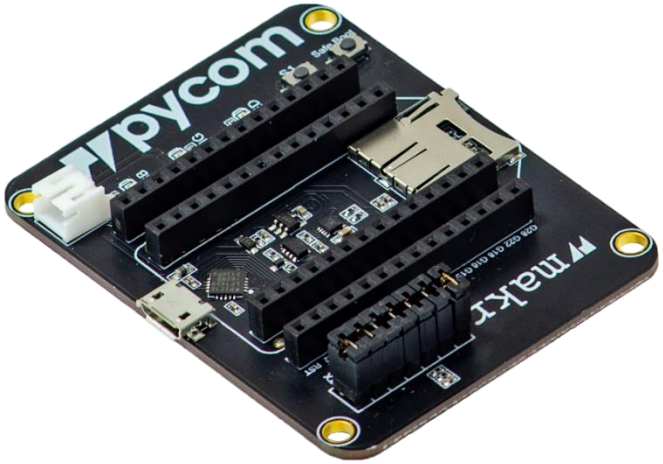


Figura 3.2 – Expansion Board

Batería Lipo [14]

Se utiliza una batería de Polímero de Litio porque el rango de tensión de salida de este tipo de baterías se adapta a la tensión de entrada soportada por la placa LoPy 4. El valor de la capacidad se ha escogido en exceso con el fin de comprobar su comportamiento de carga junto con la placa solar disponible.

Figura 3.3 – Batería Lipo

Especificaciones técnicas:

- Tensión nominal: 3.7 V
- Capacidad: 3000 mAh
- Conector JST
- Corte de carga de 4.2 V y corte de descarga de emergencia de 2.4 V
- Protección contra cortocircuitos

Placa solar [15]

Placa solar distribuida por el fabricate Adafruit capaz de producir 6 V a 330 mA a través de un conector jack de corriente continua de 3.5 mm x 1.1 mm.

Figura 3.4 – Placa solar

Especificaciones técnicas:

- Tipo de célula: Monocristalino
- Eficiencia celular: 19%
- Potencia nominal: 2 W
- Potencia máxima: 2,27 W

Cargador solar bq24074 [16]

Cargador universal para baterías de iones o polímero de litio que permite la alimentación a través de corriente continua, USB o placa solar.

Especificaciones técnicas:

- Voltaje de entrada: 5 ~ 10 V
- Tasa de carga: hasta 1.5 A
- Voltaje máximo regulado: 4.4 V
- Puerto USB tipo C

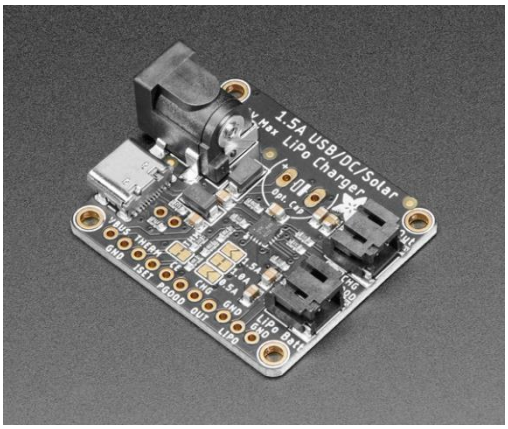


Figura 3.5 – Cargador solar bq24074

Sensor de luz IR, UV y visible SI1145 [17]

Con la finalidad de poder observar como la incidencia solar afecta a la carga de la batería, se ha añadido este sensor de luz. No contiene un elemento de detección UV real, sino que lo aproxima en función de la luz visible e IR del sol.

La librería MicroPython utilizada este creada por Nelio Gonçalves [18].

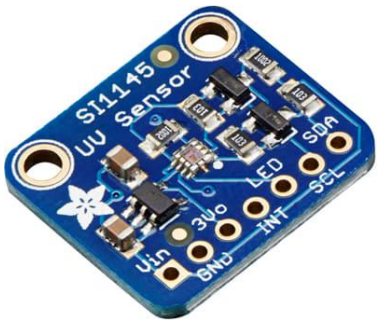


Figura 3.6 – Sensor SI1145

Especificaciones técnicas:

- Voltaje de entrada: 3 ~ 5 V
- Comunicación: I2C (dirección 0x60)
- Espectro luz visible: 400 – 800 nm (centrado en 530)
- Espectro IR: 550 – 1000 nm (centrado en 800)
- Temperatura de funcionamiento: -40 ~ 85 °C

Sensor de temperatura y humedad relativa Si7021 [19]

Sensor fabricado por Adafruit para la medición de la humedad y la temperatura ambiental. La librería utilizada ha sido creada por Chris Balmer [20] bajo licencia MIT.

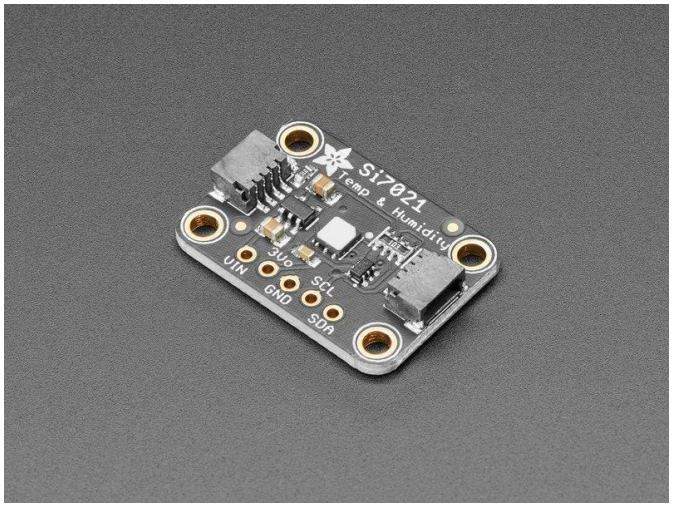


Figura 3.7 – Sensor Adafruit Si7021

Especificaciones técnicas:

- Voltaje de entrada: 3.3 ~ 5 V
- Comunicación: I2C (dirección 0x40)
- Rango temperatura: -10 ~ 85 °C
- Precisión temperatura: ± 0.4 °C
- Rango humedad relativa: 0 ~ 80 %
- Precisión humedad relativa: ± 3 %

Sensor de partículas en suspensión PMS5003 [21]

Creado por la empresa Pimoroni, este sensor detecta el tamaño y cantidad de partículas en suspensión, que están estrechamente relacionadas con la contaminación que hay en el aire. Las partículas más representativas son las PM2,5 y las PM10, cuyo tamaño es inferior o igual a 2,5 y 10 micrómetros, respectivamente. La misma compañía proporciona una librería Python [22] para utilizar el sensor.

El sensor dispone de un pin para reset y otro para que se active, momento en el que se enciende el ventilador y la resistencia de medición. Aunque la alimentación tiene que ser cercana a los 5 V, todos los pines de control y comunicación funcionan con lógica de 3,3 V.

Figura 3.8 – Sensor PMS5003

Especificaciones técnicas:

- Voltaje de entrada: 4.5 ~ 5.5 V
- Detecta partículas PM1, PM2.5, PM10
- Resolución: 1 $\mu g/m^3$
- Comunicación: UART
- Nivel lógico comunicación: 3.3 V

Para facilitar la conexión a los pines de este sensor, se utiliza el módulo de conexión [23] que puede verse en la Figura 3.9.

Figura 3.9 – Breakout para PMS5003

Monitor de batería LC709203F [24]

Permite obtener el voltaje de la batería LiPo, lo que da una indicación del nivel de carga en el que se encuentra. Su instalación debe hacerse entre la batería y el cargador solar. Su

fabricante es Adafruit, que proporciona una librería Python para su uso, pero se ha decido utilizar la creada por Scopel Emanuele [25] con licencia GNU General Public License v3.0.

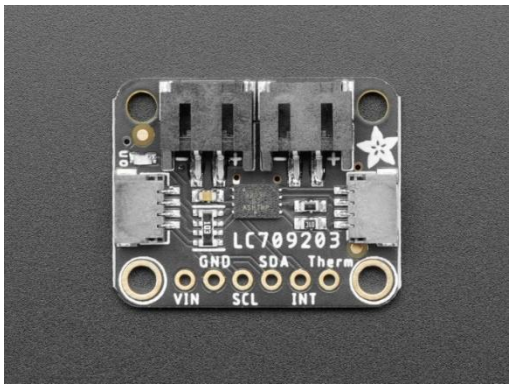


Figura 3.10 – Monitor de batería LC709203F

Especificaciones técnicas:

- Voltaje de entrada: 3.3 ~ 5 V
- Comunicación: I2C (dirección 0x0B)
- Nivel lógico comunicación: 3.3 - 5 V
- Pines de conexión tipo JST

Convertidor elevador dc/dc MT3608 [26]

Para alimentar el sensor PMS5003 mediante una batería LiPo, es necesario un elevador de tensión para obtener los 5 V requeridos. El voltaje de salida se regula mediante un potenciómetro multivuelta hasta obtener el deseado.

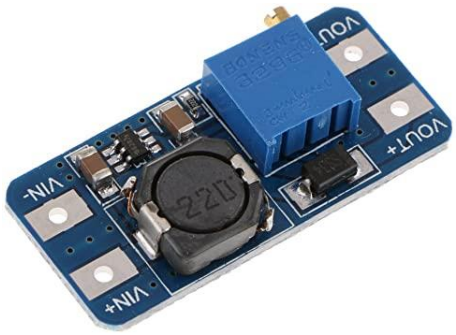


Figura 3.11 – Convertidor MT3608

Especificaciones técnicas:

- Voltaje de entrada: 2 ~ 24 V
- Voltaje de entrada: 2 ~ 28 V

- Corriente máx. salida: 2 A
- Eficiencia máx. conversión: 93 %

Reloj en tiempo real DS3231

Su finalidad es dotar al nodo de la fecha y hora correctas mediante un RTC alimentado por una batería propia (componente amarillo en la Figura 2.12). La librería utilizada ha sido creada por MicroPython Chinese Community [27].

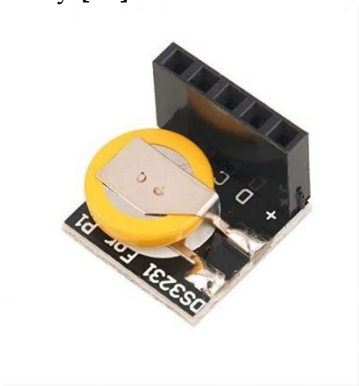


Figura 3.12 – RTC DS3231

Especificaciones técnicas:

- Voltaje de entrada: 2.3 ~ 5.5 V
- Comunicación: I2C (dirección 0x68)
- Nivel lógico máx. comunicación: Vcc + 0.3 V
- Batería recargable tipo botón

3.1.2 Montaje

El esquema eléctrico de conexión de todos los componentes anteriores puede verse en la Figura 2.13.

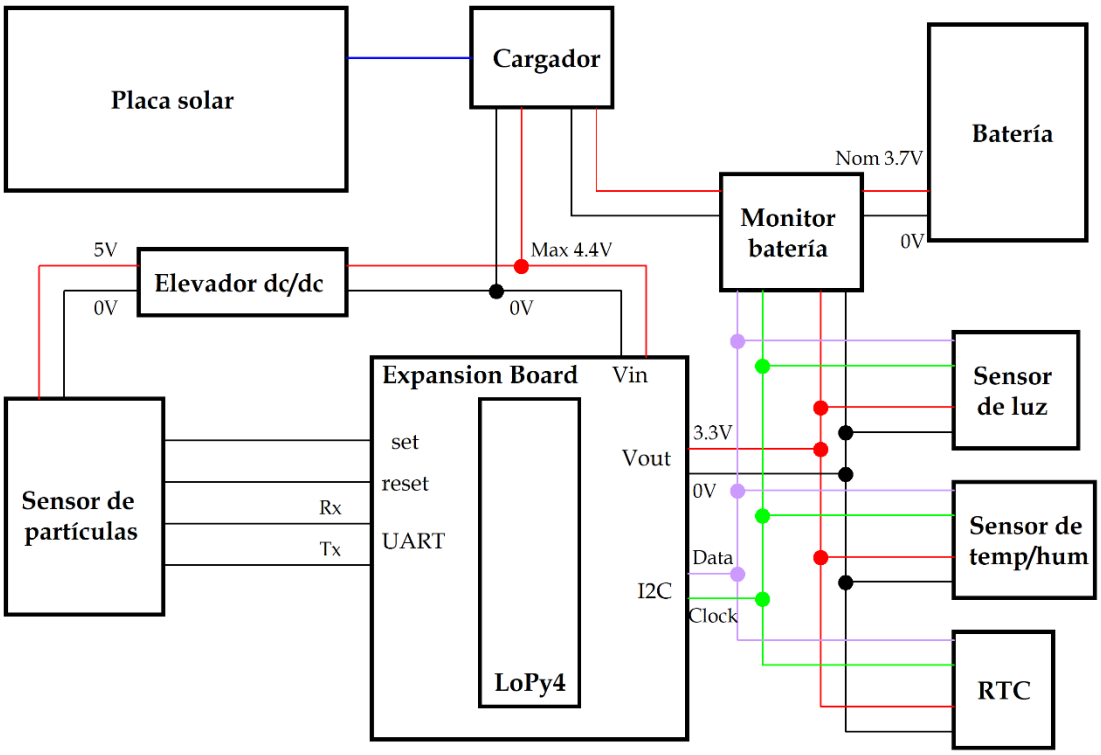


Figura 3.13 – Esquema eléctrico

La carcasa del dispositivo ha sido creada con una caja de conexiones para exterior a la que se le ha añadido una estructura, Figura 3.14, fabricada mediante impresión 3D. El plástico utilizado es PLA (ácido poliláctico) por presentar características que facilitan su impresión, aunque para aplicaciones en el exterior, como es el caso, hubiese sido mejor usar materiales de impresión como el ABS (acrilonitrilo butadieno estireno).

El programa de diseño utilizado ha sido Fusion 360 de Autodesk, en su versión gratuita para estudiantes, y la impresora una Ultimaker S3 junto con su aplicación de impresión, Ultimaker Cura.

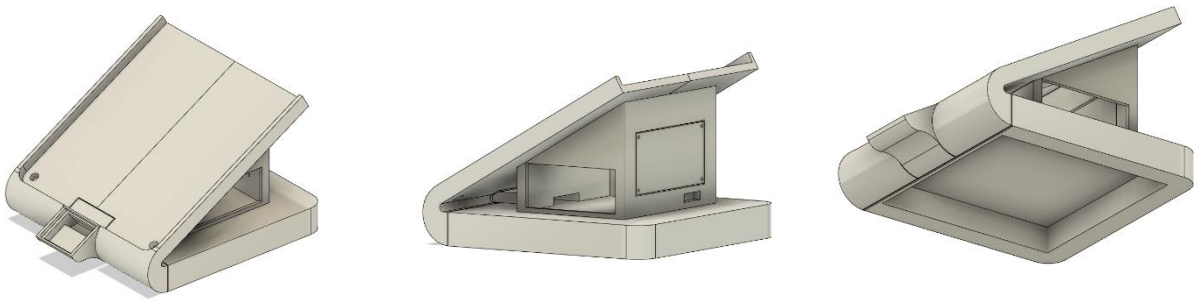


Figura 3.14 – Diseño 3D del soporte

Se ha escogido una inclinación del panel solar cercana a los 40º, que es la recomendada para paneles estáticos a la latitud en la que se encuentra Madrid. El montaje de los componentes en ambas carcasas puede verse en la Figura 2.15.

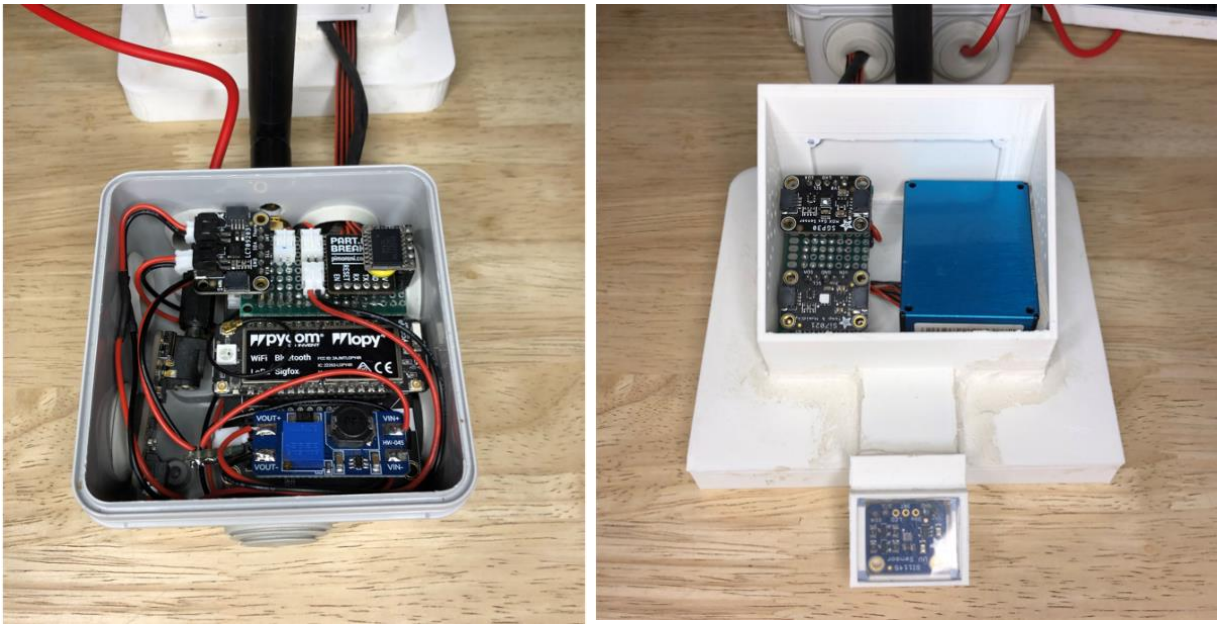


Figura 3.15 – Montaje de componentes

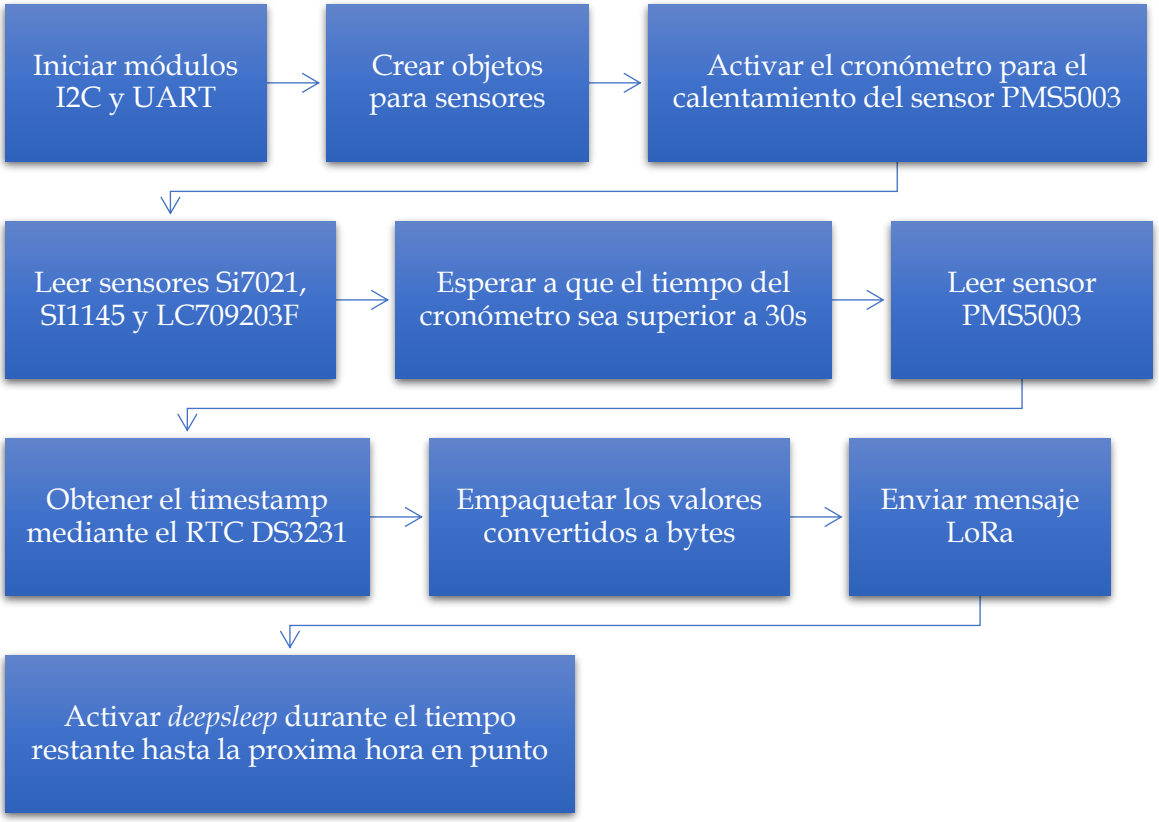
Para las pruebas funcionales, el nodo se ha instalado con orientación Sur en un balcón ubicado en el barrio de Argüelles, a unos 4 km de la primera zona de actuación.

Figura 3.16 – Ubicación para pruebas

3.1.3 Funcionalidad

El código desarrollado se encuentra alojado en Github [28], dentro de la carpeta *final*. En la carpeta *test* se encuentra el código de pruebas para verificar, por separado, el funcionamiento de cada uno de los componentes.

El nodo se activa cada hora en punto y trabaja de forma secuencial, leyendo y transformando los valores leídos de los sensores para empaquetarlos en un array de bytes y enviarlos mediante un mensaje LoRa. Además de estos valores, también se incluye el timestamp, obtenido del RTC externo, en UTC (Universal Time Coordinated) +0 para evitar problemas con los cambios de hora en invierno y verano. Por último, se calcula el tiempo restante hasta la próxima hora y se configura la placa en modo *deepsleep* hasta ese momento. A continuación, se representa un diagrama secuencial del funcionamiento.



El mensaje de salida se compone de los 9 valores leídos y tiene un tamaño total de 20 bytes, 8 enteros de 2 bytes y un entero (el timestamp) de 4 bytes, codificados en *big-endian*. El orden de los valores, y la modificación realizada para el envío en algunos de ellos, se enumeran a continuación.

1. Temperatura: se redondea a 2 decimales y se multiplica por 100 para convertirla en un entero.

2. Humedad relativa: se redondea a un entero sin decimales, ya que la precisión del sensor es del ± 3%.
3. PM2,5: la lectura directa desde el sensor son valores enteros con un rango de 0 a 1000 (µg/m³), por lo que no es necesario realizar ninguna conversión.
4. PM10: se procede igual que en el caso anterior.
5. Luz UV: se redondea a 2 decimales y se multiplica por 100.
6. Luz IR: no se modifica.
7. Luz Visible: no se modifica.
8. Voltaje batería: se redondea a 3 decimales y se multiplica por 1000.
9. Timestamp: no se modifica.

Este orden y las conversiones realizadas deben tenerse en cuenta en el momento de descodificar el mensaje cuando se recibe en AWS.

3.1.4 Estudio de ubicaciones

Teniendo en consideración que el radio de alcance mínimo de la puerta de enlace son 10 km en campo abierto, la ubicación aproximada de esta pasarela deberá estar donde se indica en la Figura 3.17, con la finalidad de dar cobertura a toda la zona de actuación de la primera fase.

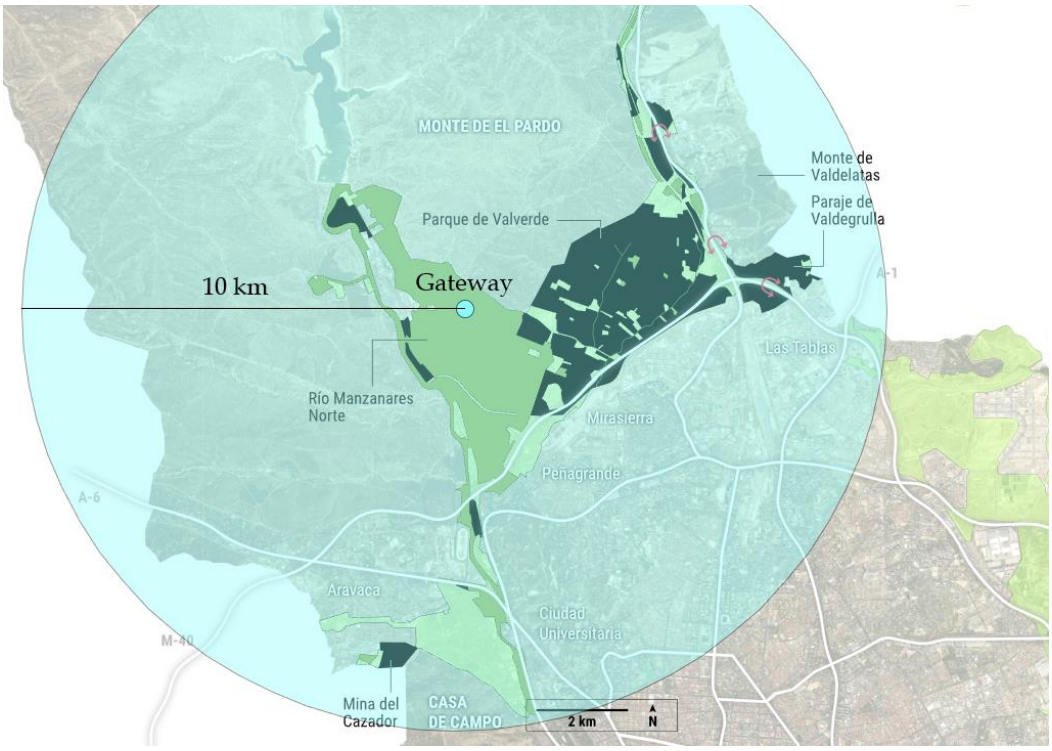


Figura 3.17 – Ubicación gateway

El punto más elevado cercano a esa zona es el pico Tambor, con unos 720 metros de altura, siendo éste el mejor emplazamiento donde posicionarla.

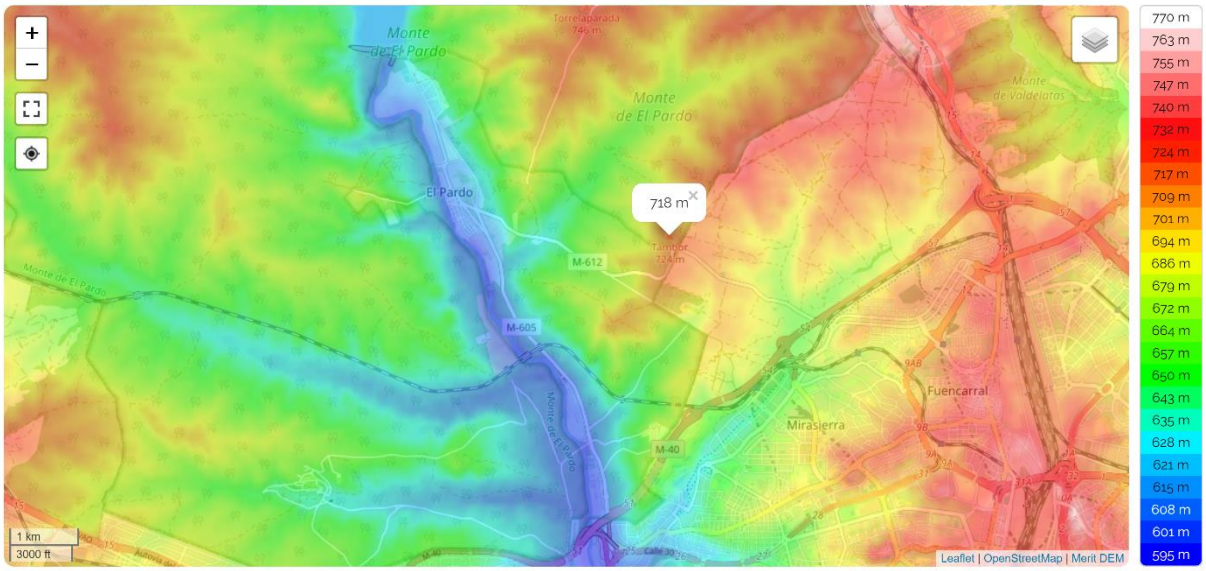


Figura 3.18 – Mapa topográfico [29]

Para ubicar los nodos sobre el plano, se puede tomar como referencia la red de estaciones fijas de control de calidad del aire del ayuntamiento de Madrid, Figura 3.19, donde la distancia aproximada entre ellas es de unos 3 km.

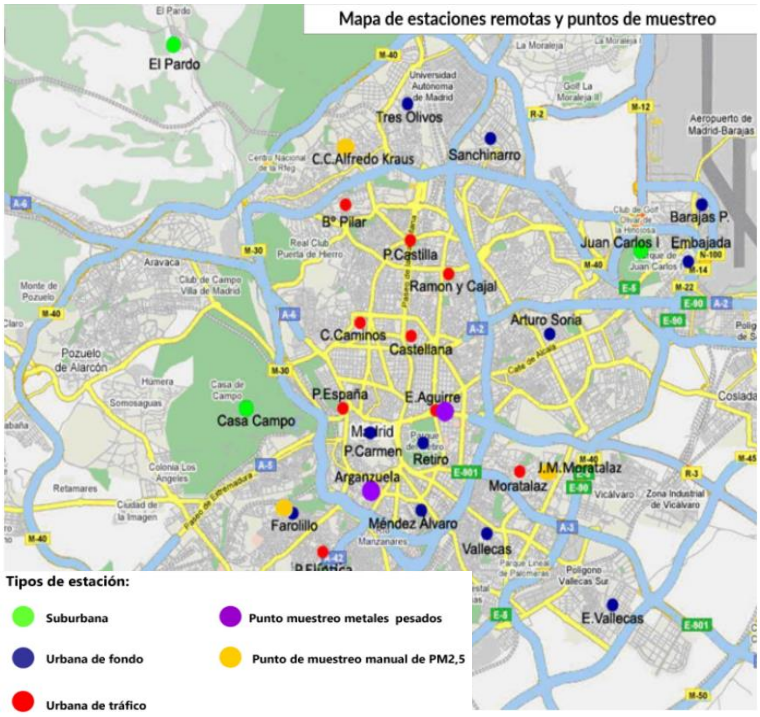


Figura 3.19 – Estaciones de control de calidad del aire (Madrid) [30]

Por lo tanto, una posible configuración espacial de los nodos es mostrada en la Figura 2.19, representando las 10 ubicaciones con un círculo amarillo.

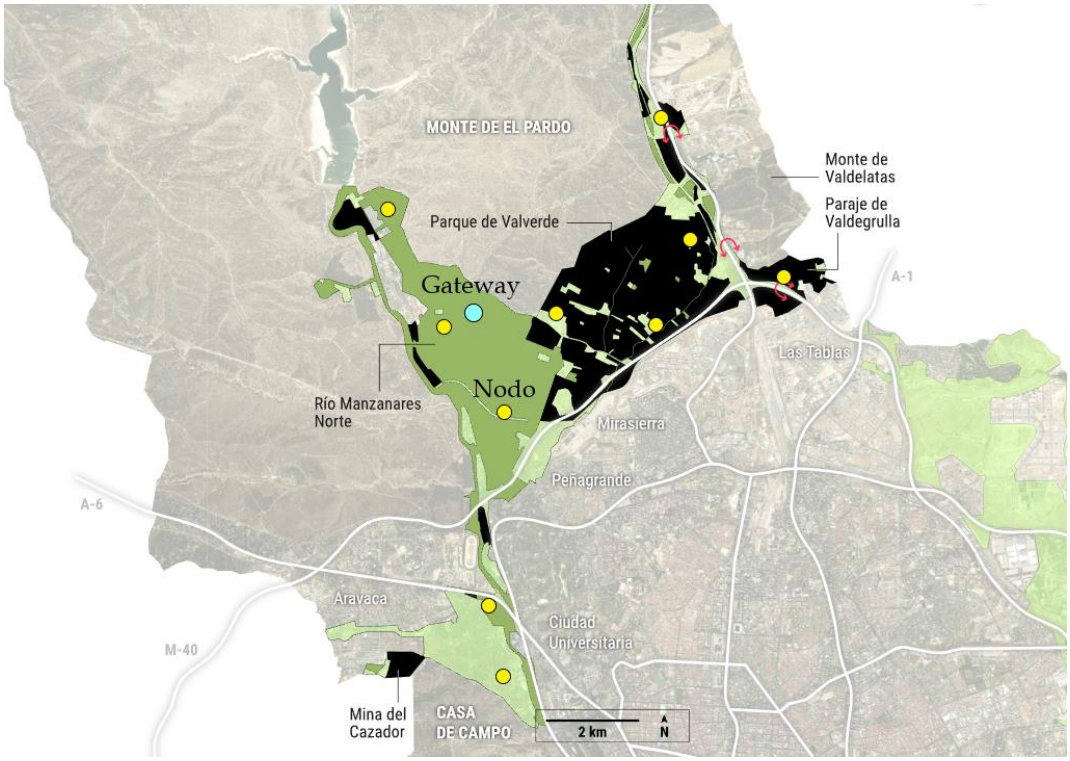


Figura 3.20 – Ubicación de los nodos

Debido a que no se ha utilizado un dispositivo GPS, sería necesario apuntar la correspondencia entre la ubicación en la que se instala un nodo y su identificador LoRa único, DevEUI.

3.2 Conectividad LoRaWAN

En este apartado se describen los pasos dados para la creación de la red LoRaWAN, que incluye la configuración de AWS IoT Core y de la puerta de enlace o gateway.

3.2.1 Gateway

El router utilizado como puerta de enlace es el modelo **Sentrius RG186** [31]de la compañía Laird Connectivity (Figura 3.21). Está basado en el chip Lora SX1301/SX1257 de Semtech, e incluye además interfaces WiFi, Bluetooth y Ethernet. Expone un servidor web HTTP, con un DNS que se utiliza para crear una dirección única, de modo que se puede acceder a su página de configuración dentro de la red local usando la URL *https://RG1xxXXXXXX.local* desde un navegador, donde *XXXXXX* son los tres últimos bytes de la dirección MAC del dispositivo.

Figura 3.21 – Gateway Sentrius RG186 [30]

Utiliza un Linux embebido como sistema operativo y se requiere una versión mínima de firmware de 93.8.5.25 para asegurar que la versión de Semtech Basics Station cumple con el mínimo requerido (v2.0.5) por AWS IoT Core.

Antes de realizar la configuración de la gateway, es necesario realizar un par de procedimientos previos en AWS. El primero de ellos es añadir un Rol de IAM que permita al servidor de configuración y actualización (CUPS) gestionar las credenciales de la pasarela inalámbrica.

AWS **Identity and Access Management (IAM)** es un servicio web que ayuda a controlar de forma segura el acceso a los recursos de AWS. Se utiliza para controlar quién está autentificado (ha iniciado sesión) y autorizado (tiene permisos) para utilizar recursos.

Un **Rol** es una identidad IAM que tiene permisos específicos. Un rol IAM tiene algunas similitudes con un usuario IAM. Tanto los roles como los usuarios son identidades de AWS con políticas de permisos que determinan lo que la identidad puede y no puede hacer. Sin embargo, en lugar de estar asociado de forma exclusiva a una persona, un rol puede ser asumido por cualquiera que lo necesite.

Accediendo al panel de control de IAM, en el apartado *Roles*, podemos crear el Rol necesario. Lo primero que nos pide es seleccionar la entidad de confianza, que en este caso es la propia cuenta de AWS (Figura 3.22).

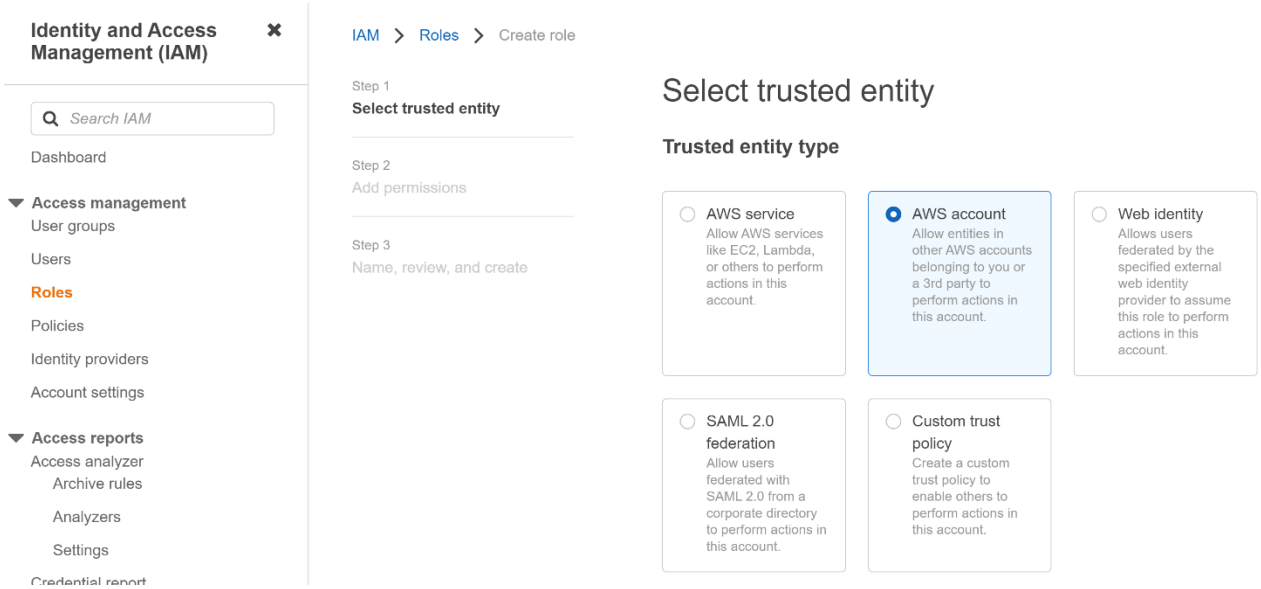


Figura 3.22 – Creación de un Rol IAM

Lo siguiente es añadir la política de permisos, que se puede encontrar entre las que ya nos ofrece por defecto AWS. En el buscador de políticas (Figura 3.23), ingresamos el nombre *AWSIoTWirelessGatewayCertManager*, que permite el acceso de la identidad asociada a crear, listar y describir los Certificados IoT.

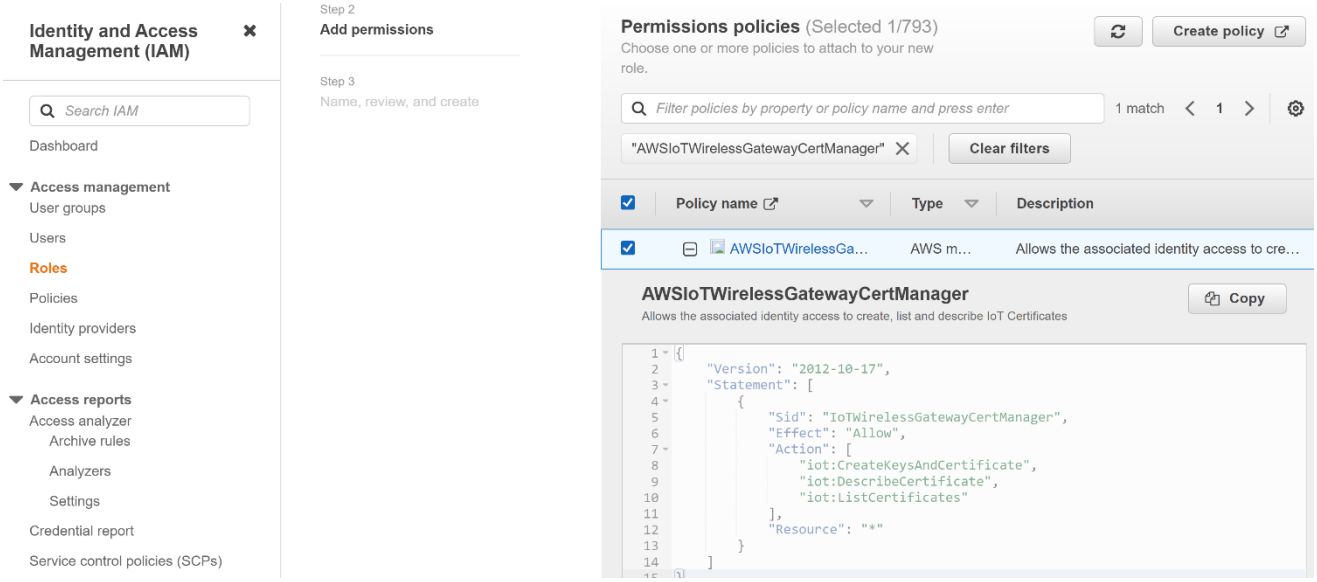


Figura 3.23 – Política de gestión de certificados

Por último, le damos un nombre al Rol (*IotWirelessGatewayCertManagerRole*) y lo creamos. Debemos acceder a él y cambiar las relaciones de confianza, desde la pestaña asociada, para que incluya el objeto JSON que se muestra en la figura 3.24.

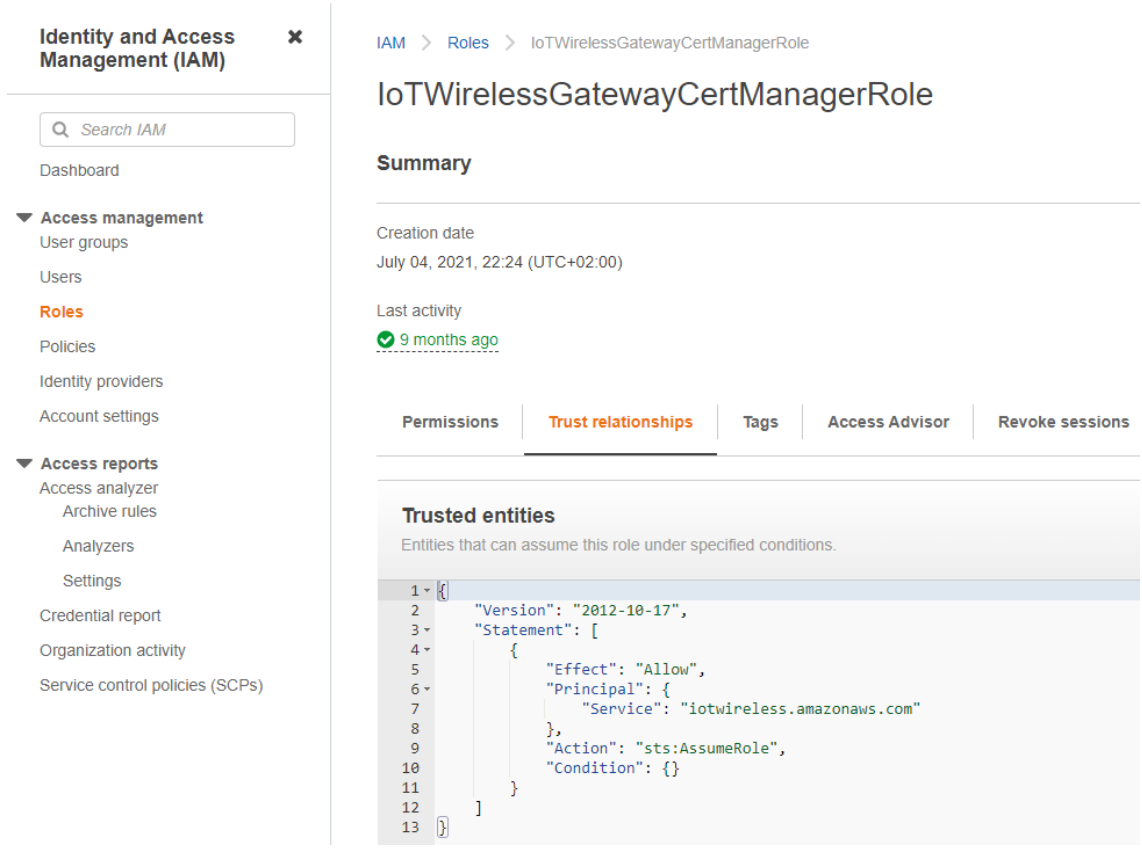


Figura 3.24 – Modificación de las relaciones de confianza

El siguiente procedimiento es añadir la puerta de enlace a AWS IoT Core, por lo que necesitaremos el identificador **GatewayEUI** que se encuentra apuntado en la etiqueta posterior del dispositivo. Dentro de la consola de *AWS IoT*, en el apartado *Wireless connectivity*, seleccionamos *Gateways* y añadimos una nueva (Figura 3.25).

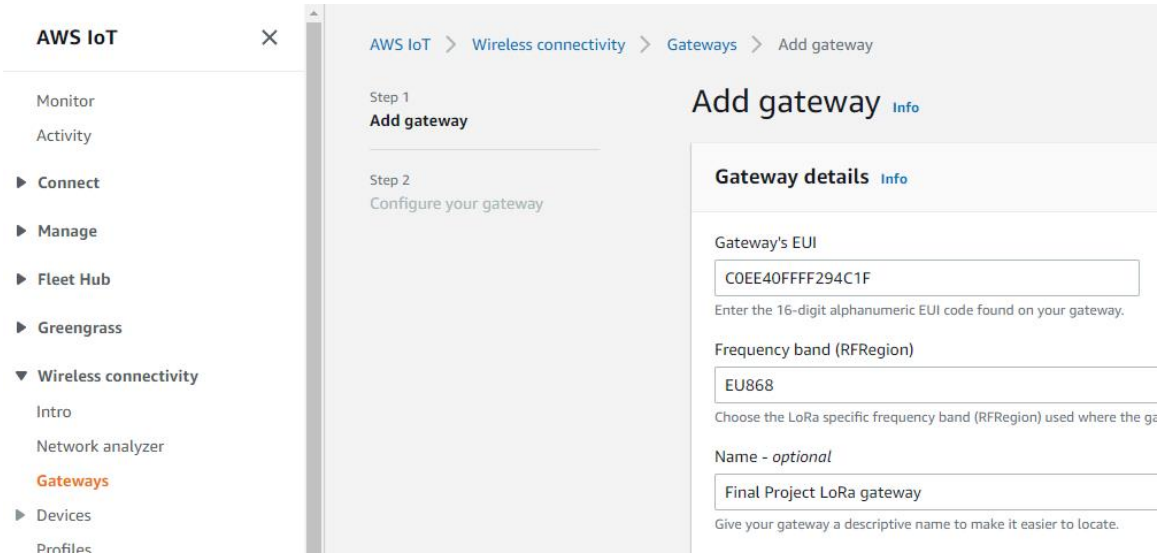


Figura 3.25 – Integración de la puerta de enlace en AWS IoT

En el primer paso, se identifica la puerta de enlace con su GatewayEUI, se indica la banda de frecuencia en la que funciona, en nuestro caso EU868, y se le da un nombre característico.

En el segundo paso, Figura 3.26, debemos hacer clic en el botón *Create certificate* y una vez que aparezca el mensaje de verificación, hacemos clic en *Download certificate files* para descargar el certificado (**xxxx.cert.pem**) y la clave privada (**xxxx.private.key**). En la sección *Provisioning credentials*, hacemos clic en *Download server trust certificates* para descargar los certificados de confianza de los servidores CUPS (**cups.trust**) y LNS (**lns.trust**). Finalmente copiamos los **endpoints** del CUPS y del LNS y guardamos toda esta información para configurar la pasarela.

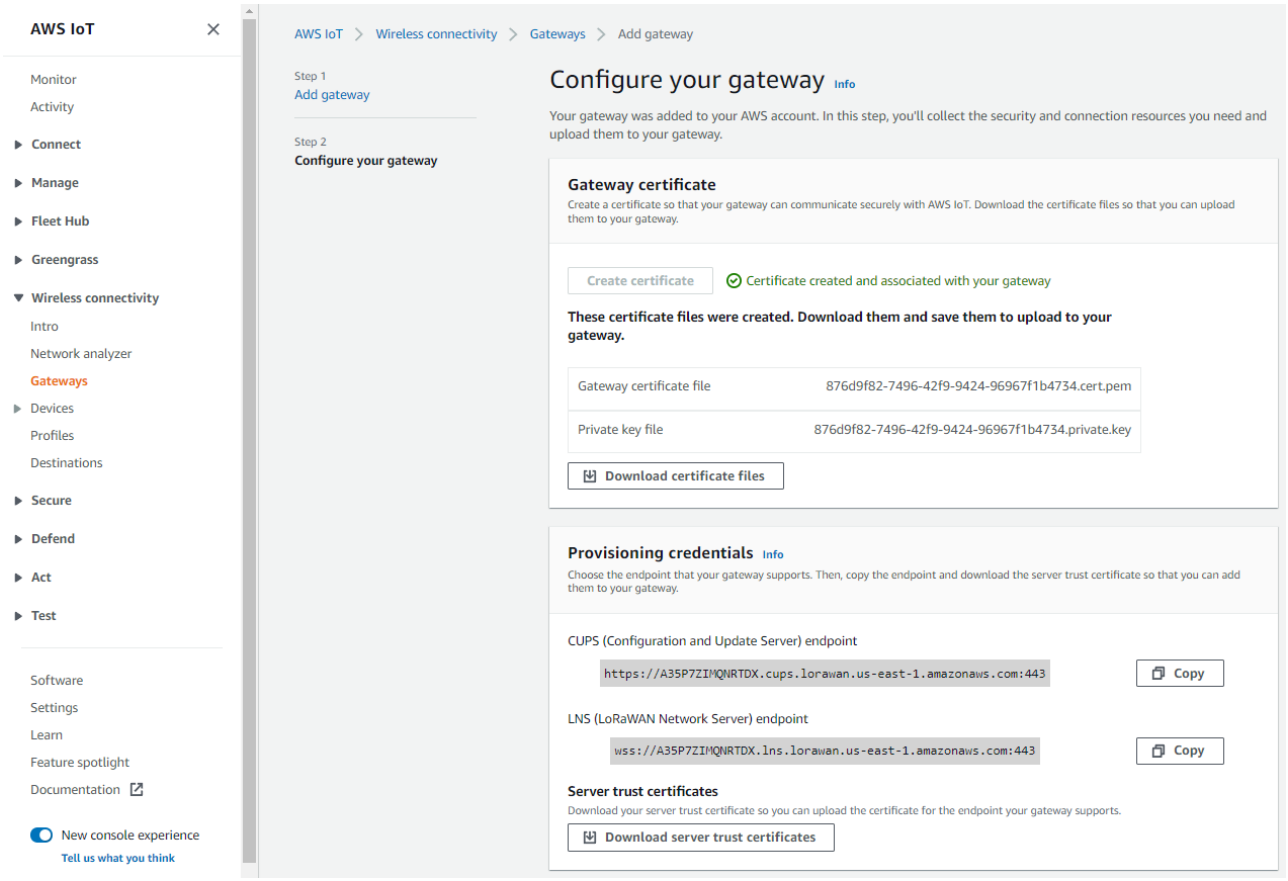


Figura 3.26 – Creación de certificados

Una vez se ha creado la puerta de enlace en la consola AWS IoT, accedemos a su página interna de configuración, mediante la URL mencionada anteriormente, y clicamos en el apartado de *LoRa* (Figura 3.27). Seleccionamos el modo como *Semtech Basics Station* y rellenamos la información pedida con los archivos y endpoints obtenidos de AWS. El servidor CUPS Boot es el mismo que el servidor CUPS, por lo que la información a rellenar es la misma en ambos (Figura 2.28).

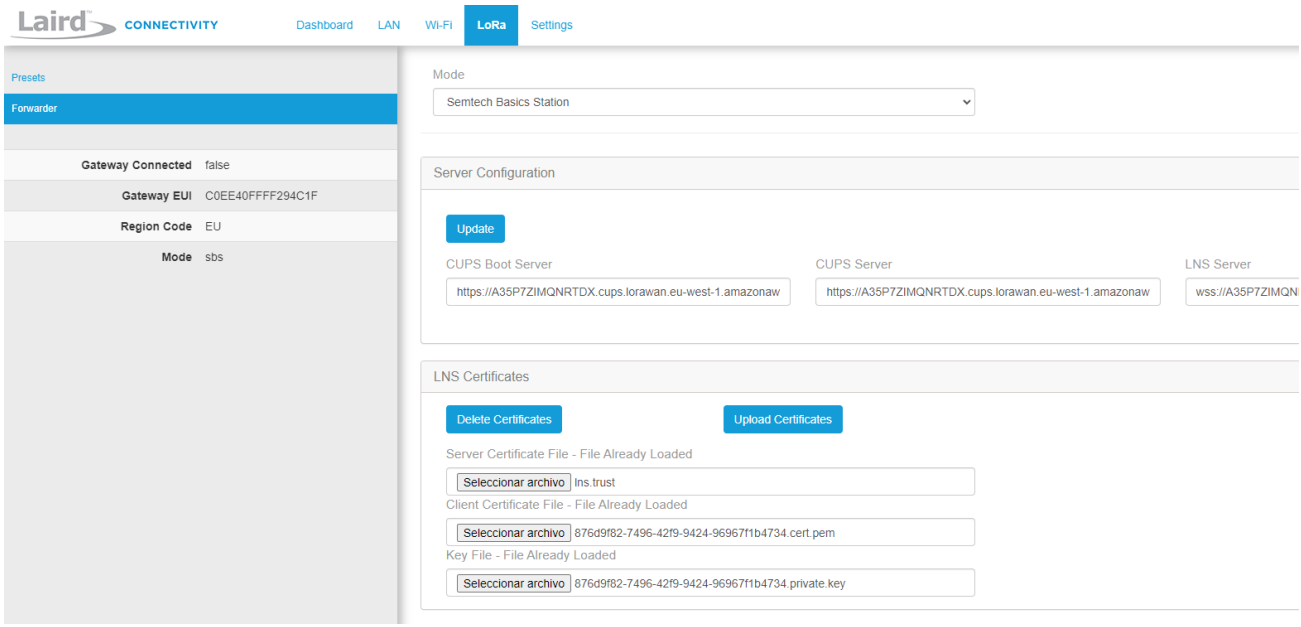


Figura 3.27 – Endpoints y certificados para el LNS

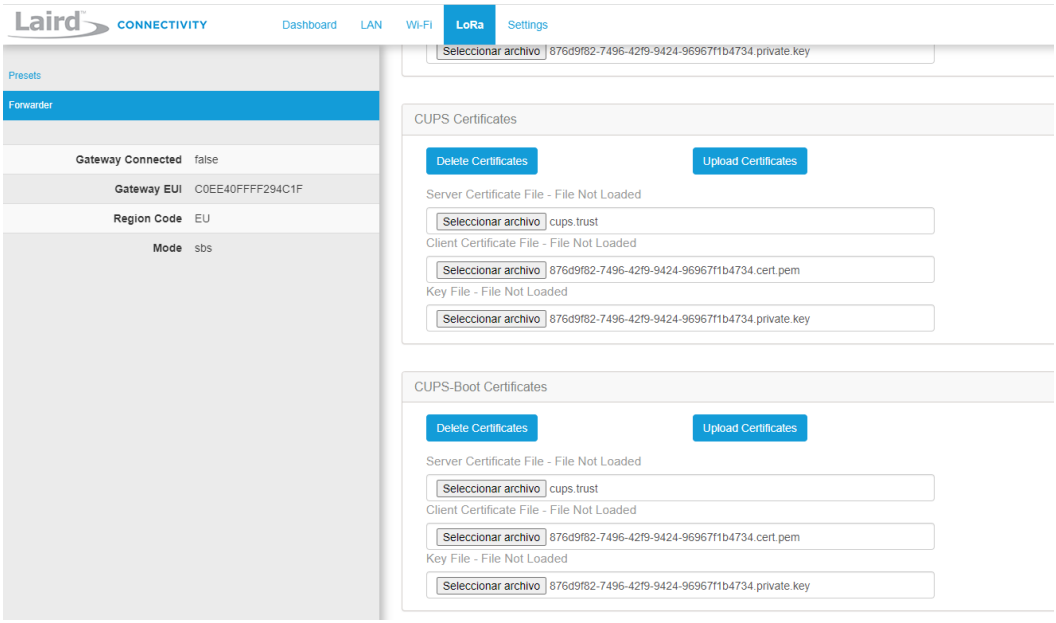


Figura 3.28 – Certificados para el servidor CUPS y CUPS-Boot

Podemos verificar que la conexión con AWS se ha realizado correctamente mediante el apartado *Dashboard*, en la parte correspondiente a LoRa (Figura3.29).

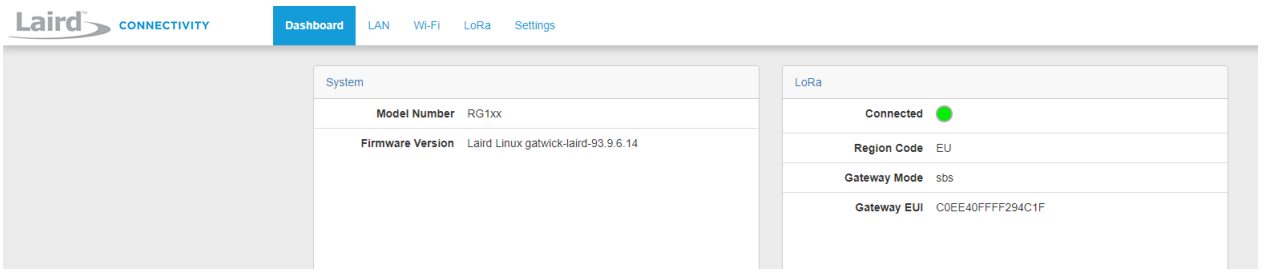


Figura 3.29 – Verificación de conexión

3.2.2 Nodo

Antes de configurar AWS IoT, hay que identificar las tres claves raíz (*root keys*) para LoRaWAN de nuestro dispositivo.

- **DevEUI** (16 dígitos hexadecimales): identificador único asignado por el fabricante. En la placa LoPy puede obtenerse con la función *lora.mac()* importando previamente la librería *lora*.
- **AppKey** (32 dígitos hexadecimales): Secreto compartido entre el dispositivo y la aplicación. Se ha elegido al azar y se ha almacenado como una variable dentro del código MicroPython desarrollado.
- **AppEUI** (16 dígitos hexadecimales): identificador único para el servidor de unión cuando el dispositivo acepta OTAA. Para su creación se ha procedido del mismo modo que con la AppKey.

Una vez que disponemos de estas claves, ingresamos en la consola de AWS IoT, dentro del apartado *Wireless Connectivity*, y seleccionamos Perfiles (*Profiles*). Vemos que existen de dos tipos, Perfiles de dispositivo y Perfiles de servicio, y la forma en la que pueden crearse (Figura 3.30).

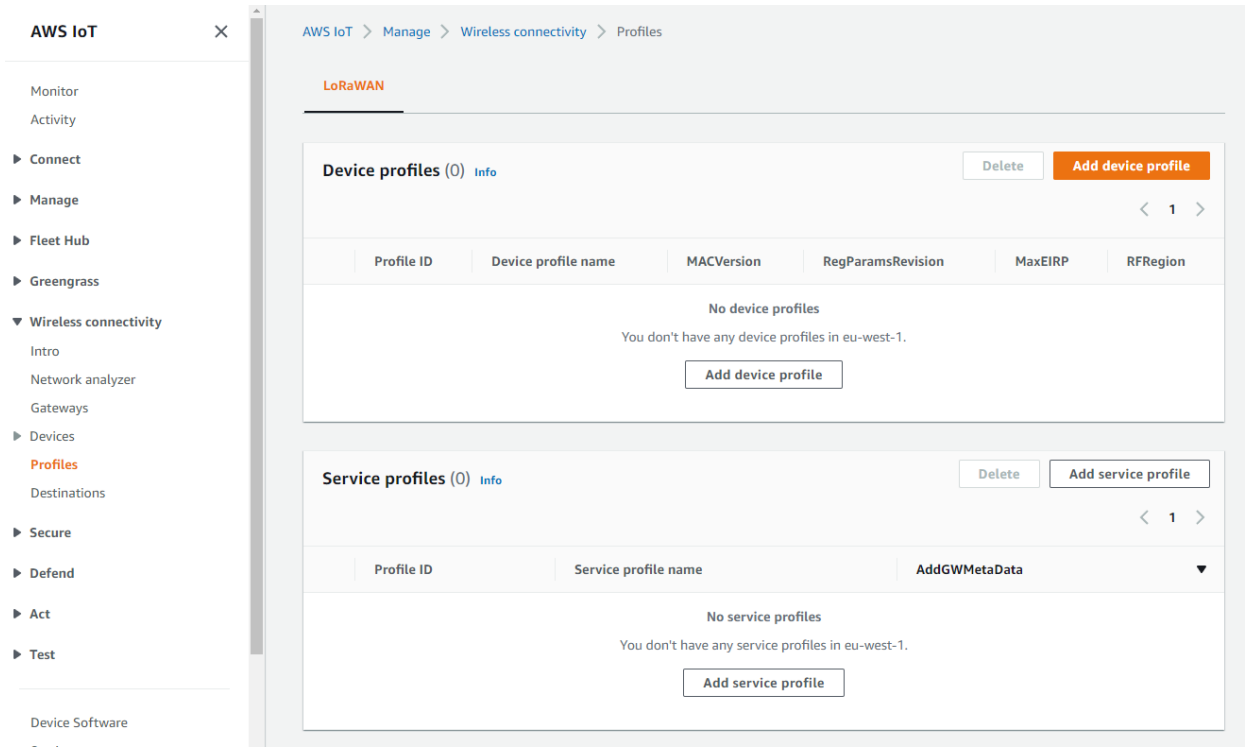


Figura 3.30 – Perfiles LoRaWAN

A continuación, se describen los 5 pasos a realizar para la configuración de un dispositivo LoRaWAN en AWS IoT Core.

1. Creación del Perfil de dispositivo (*Device profile*)

Los Perfiles de dispositivo son ajustes preconfigurados para los dispositivos que vayamos a utilizar. En el momento de su creación, elegimos el perfil por defecto para un dispositivo de clase A que trabaja en la frecuencia europea, llamado **EU868 – A**, y dejamos sin modificar la configuración obtenida (Figura 3.31).

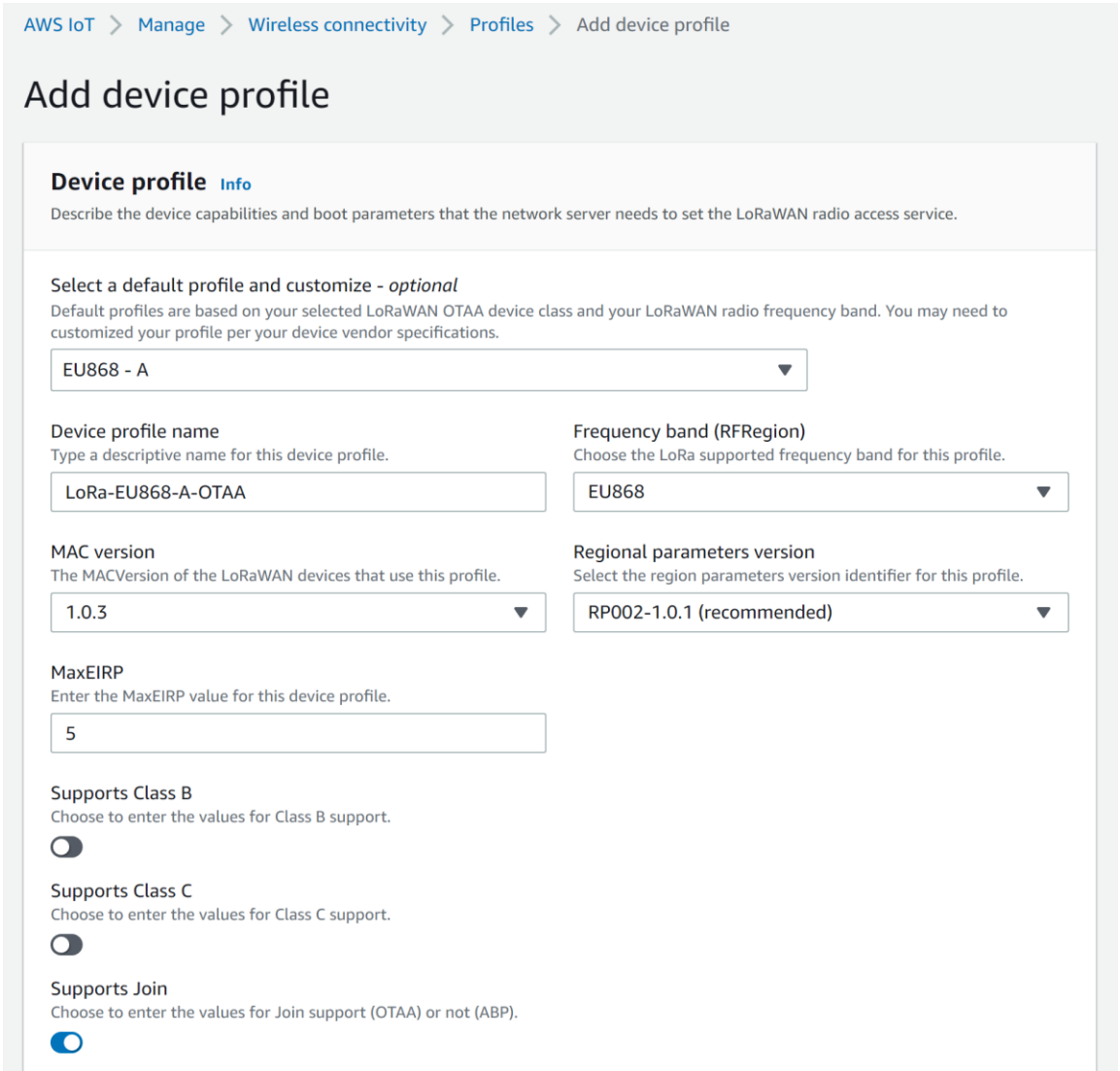


Figura 3.31 – Perfil de dispositivo

2. Creación de Perfil de servicio (*Service profile*)

Un Perfil de servicio describe las características que se activan para el usuario y la tasa de mensajes que se pueden enviar a través de la red. Cuando creamos este Perfil (Figura 3.32), elegimos un nombre adecuado y activamos el parámetro **AddGWMetaData** para recibir

metadatos adicionales con cada carga útil del dispositivo, por ejemplo, RSSI y SNR para la transmisión de datos. Estos metadatos podrían utilizarse para optimizar la cobertura de la red.

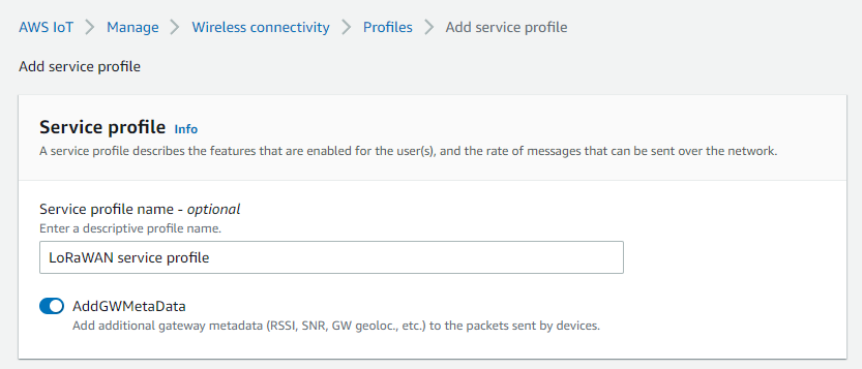


Figura 3.32 – Perfil de servicio

3. Creación de un Rol IAM para el Destino (*Destination*)

Un Destino describe la regla de AWS IoT que procesa los datos de un dispositivo inalámbrico para que los servicios de AWS IoT puedan utilizar esos datos. Para que sea funcional, antes es necesario crear un Rol IAM junto con la política de permisos adecuada.

Primero, creamos una política (*IoTWirelessDestinationPolicy*) que otorga al Rol permisos para describir el endpoint de IoT y publicar mensajes en AWS IoT, ya que esta no se encuentra disponible por defecto. La configuración mediante formato JSON debe ser la mostrada en la Figura 3.33.

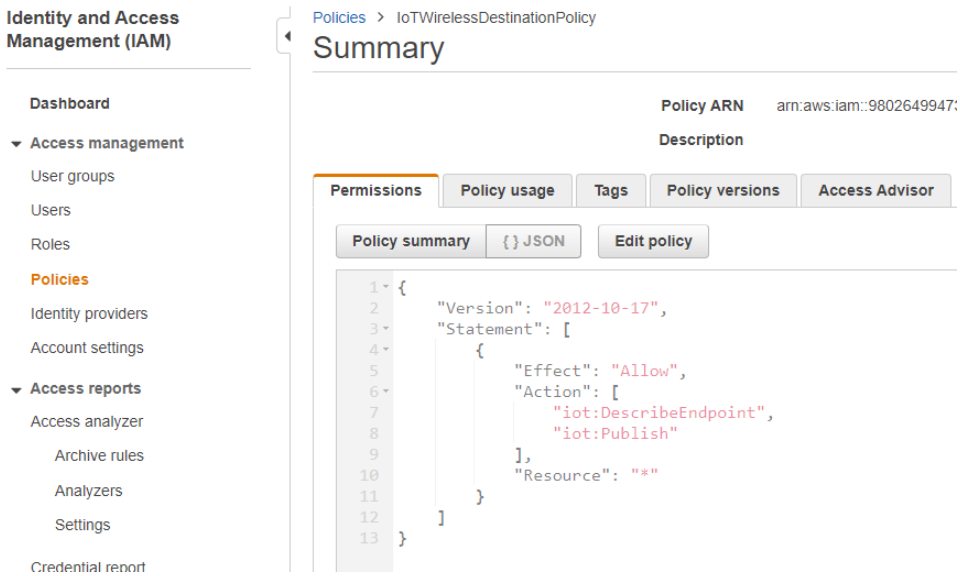


Figura 3.33 – Política de permisos para el Destino

A continuación, se crea un Rol (*IoTWirelessDestinationRole*) y se le añade la anterior política de permisos (Figura 3.34).

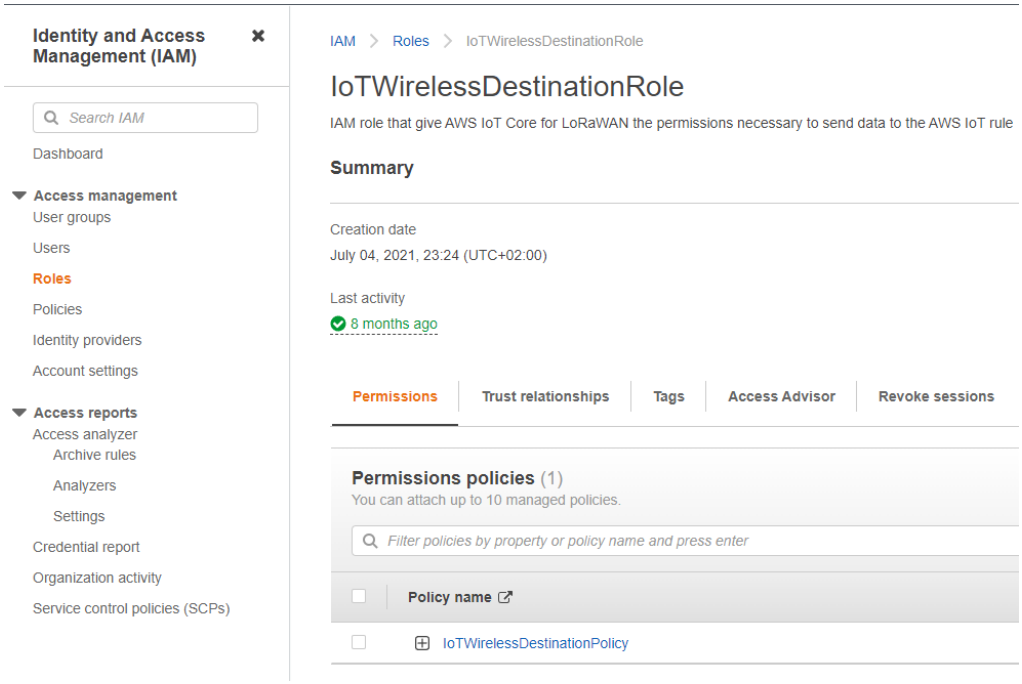


Figura 3.34 – Creación de un Rol para el Destino

Debemos cambiar las relaciones de confianza del mismo modo que con el Rol creado para la gestión de certificados (Figura 3.35).

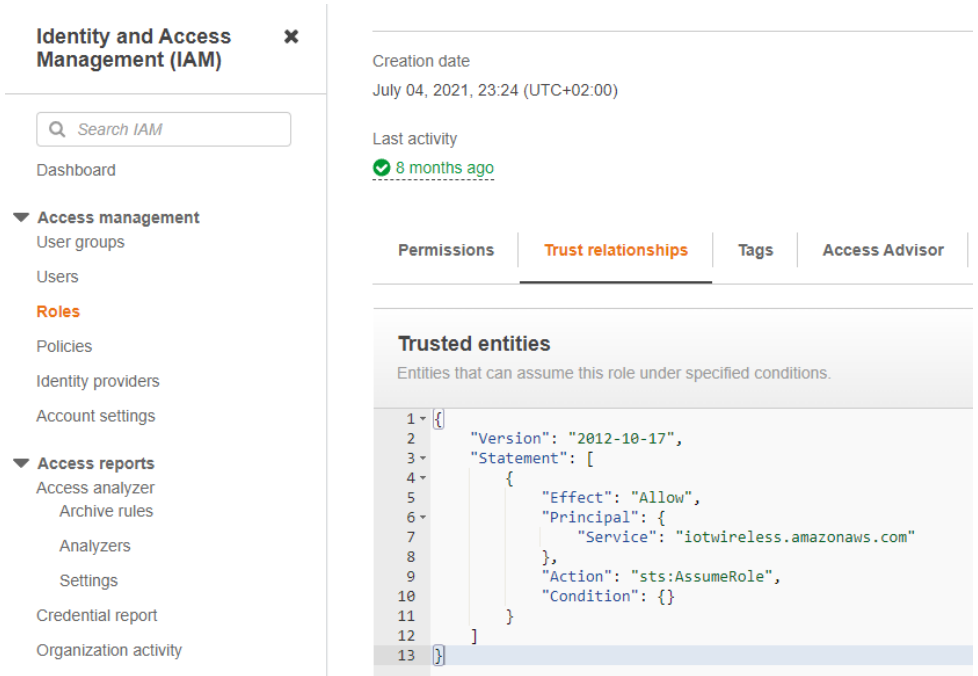


Figura 3.35 – Modificación de la relación de confianza

4. Creación del Destino (*Destination*)

Un Destino describe la regla de AWS IoT que procesa los datos de un dispositivo inalámbrico para que los servicios de AWS IoT puedan utilizar los datos. Un destino puede ser compartido por muchos dispositivos y permite realizar un filtro para redirigir los mensajes en función de la finalidad del dispositivo.

En el momento de añadir un Destino, dentro de la consola de AWS IoT, introducimos un nombre identificativo (*LoRaWAN_destination*) y el nombre de una Regla (*LoRaWAN_messages_rule*) que será creada con el mismo nombre más adelante (Figura 3.36).

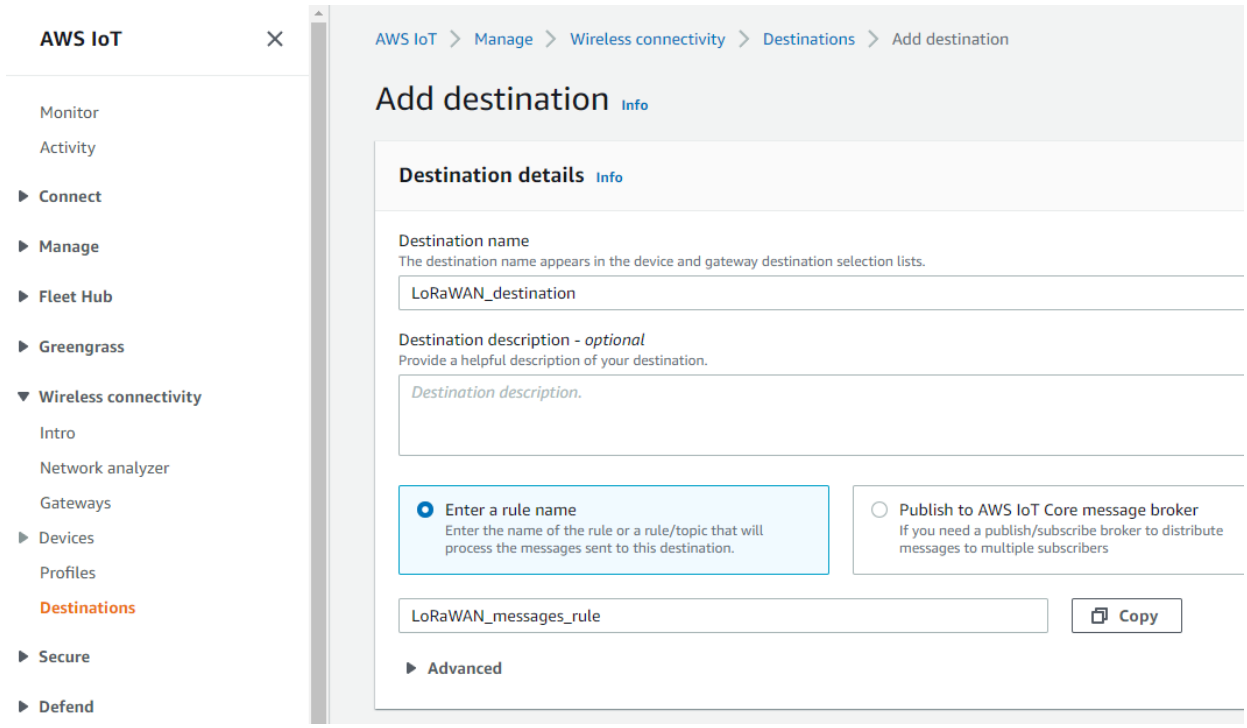


Figura 3.36 – Creación de un Destino

También encontramos una sección para introducir los permisos necesarios, donde debemos seleccionar el Rol creado en el paso anterior (Figura 3.37).

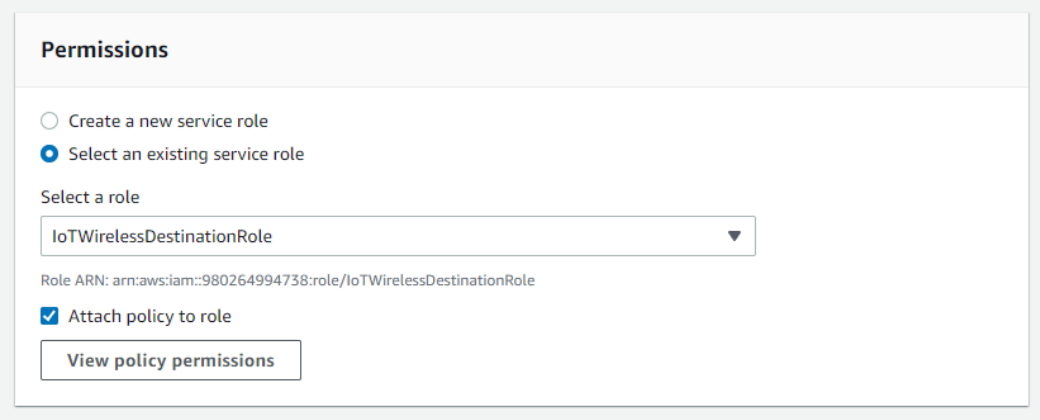


Figura 3.37 – Selección del Rol para el Destino

5. Creación de un Dispositivo (Device)

En la consola de AWS IoT, dentro del apartado *Wireless Connectivity*, seleccionamos *Devices* y desde ahí añadimos uno nuevo. Escogemos como especificación **OTAA v1.0.x** e introducimos los valores para DevEUI, AppKey y AppEUI (Figura 3.38).

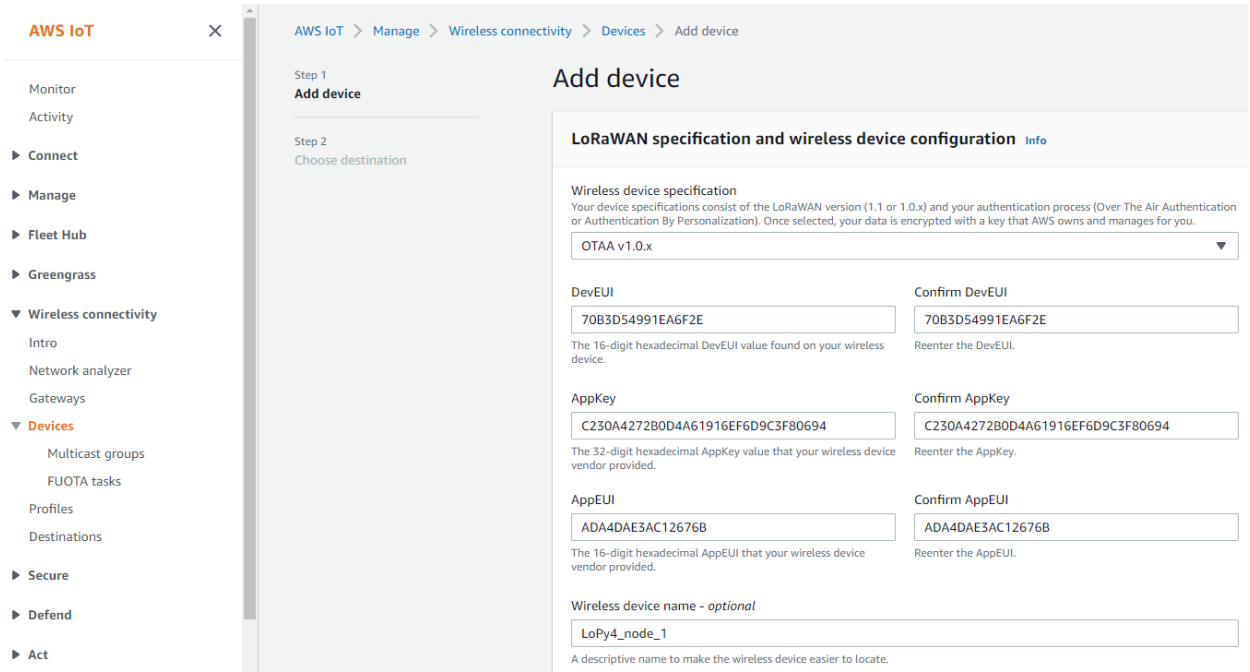


Figura 3.38 – Creación de un Dispositivo

Más abajo en esa página, encontramos la sección de Perfiles, donde debemos seleccionar los creados para el dispositivo y para el servicio (Figura 3.39).

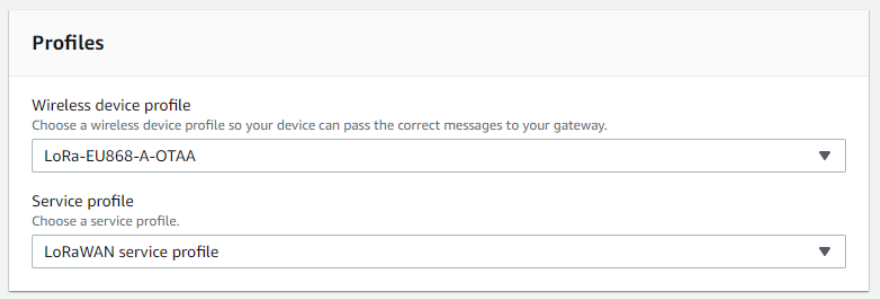


Figura 3.39 – Selección de los Perfiles para el Dispositivo

En el siguiente paso, se nos pide elegir un Destino, donde seleccionaremos en el desplegable el creado en el punto anterior (Figura 3.40).

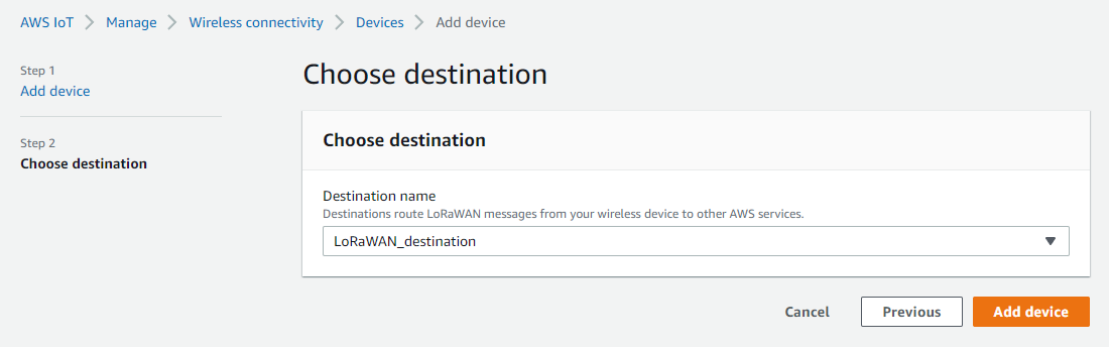


Figura 3.40 – Selección del Destino para el Dispositivo

3.3 Gestión de los mensajes

En este apartado se hará una explicación de cómo se decodifica el mensaje LoRa binario recibido en AWS IoT y su almacenamiento en una base de datos de series temporales.

3.1.1 Decodificación binaria

Para comunicarse con el resto de los servicios dentro de AWS IoT, el servidor LNS utiliza mensajes en formato JSON que son enviados a un broker MQTT instalado por defecto en la plataforma, en el que pueden crearse todos los *topics* que sean necesarios para intercomunicar servicios entre sí.

En el momento en el que se recibe un mensaje Lora en el servidor LNS, éste crea un objeto JSON en el que incluye los datos binarios recibidos, representados en base 64, como valor de la

propiedad **PayloadData**. También incluye las propiedades **WirelessDeviceId**, con el identificador único del dispositivo dado por AWS IoT, y **WirelessMetadata**, que incluye información relevante sobre la comunicación LoRaWAN. A continuación, se muestra la estructura y contenido de uno de los mensajes salientes del servidor LNS, en el que se han eliminado algunos de los metadatos para clarificar.


```
{
  "PayloadData": "CboAJgAHAAcAAgD+AQQOkGKZNYo=",
  "WirelessDeviceId": "e735d508-cf9a-45cd-a7bb-8b42013d0442",
  "WirelessMetadata": {
      "LoRaWAN": {
          "DataRate": "5",
          "DevEui": "70b3d54991ea6f2e",
          "Frequency": "867500000",
          "Gateways": [
              {
                  "GatewayEui": "c0ee40ffff294c1f",
                  "Rssi": -53,
                  "Snr": 7
              }
          ],
          "Timestamp": "2022-06-02T22:12:08Z"
      }
  }
}
```


Para decodificar el payload en base 64, y convertirlo en los valores leídos por los sensores, es necesario utilizar **AWS Lambda** [32], un servicio sin servidor que ejecuta código como respuesta a eventos y administra automáticamente los recursos informáticos subyacentes. Permite desarrollar el código en múltiples lenguajes, por lo que se ha decidido utilizar Python. La creación y configuración del servicio Lambda se expone en los siguientes pasos.

1. Crear la función Lambda

Accediendo a la consola de AWS Lambda, en el apartado *Functions*, podemos crear la función que se ejecutará cada vez que se recibida un mensaje del servidor LNS. En nuestro caso

seleccionaremos *Author from scratch* de entre las opciones disponibles que pueden verse en la Figura 3.41.

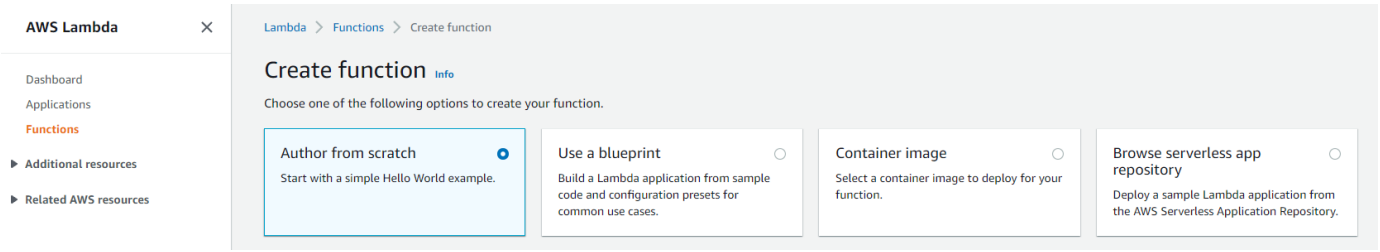


Figura 3.41 – Creación de la función AWS Lambda

En la información básica que nos pide en esa misma página (Figura 3.42), introducimos el nombre de la función, el lenguaje de programación y la arquitectura donde se queremos que se ejecute. También nos posibilita crear automáticamente un Rol con los permisos básicos necesarios para la ejecución.

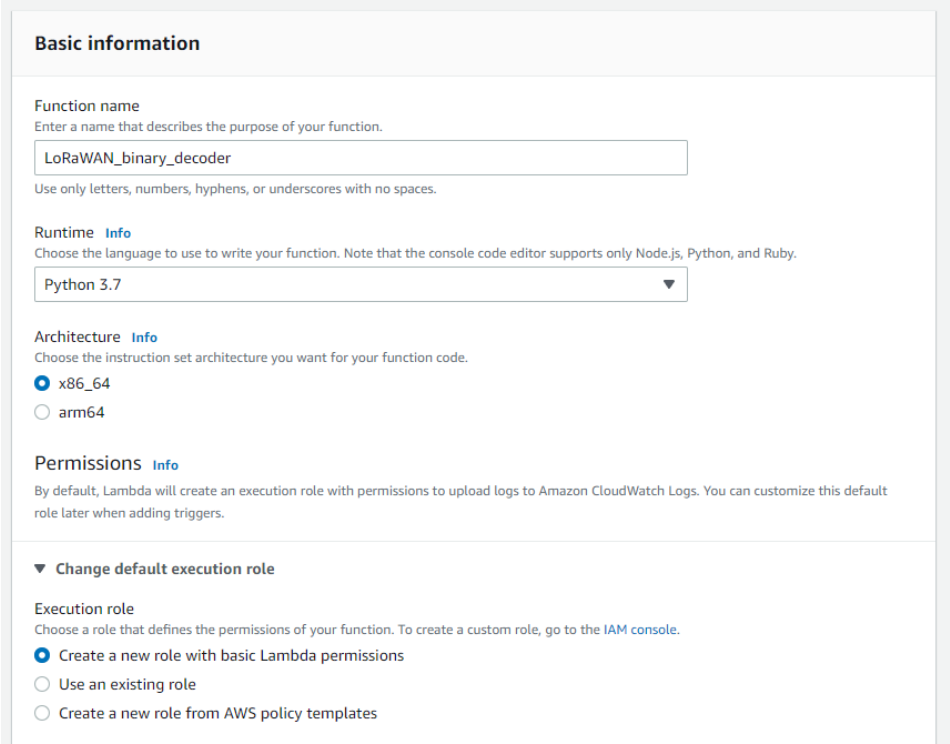


Figura 3.42 – Configuración de la función AWS Lambda

Seguidamente nos abrirá una página con un editor de código donde podemos crear nuestra función. El código desarrollado se encuentra en el repositorio Github de este proyecto, dentro de la carpeta AWS y con el nombre **lambda_fun.py**. Está basado en el código de ejemplo dado en la aplicación **aws-iot-corelorawan-sampledecoder** del *Serverless Application Repository* que dispone AWS para los desarrolladores. La modificación principal se ha realizado en la función

dict_from_payload(), donde se realiza la decodificación propiamente dicha y se devuelve un diccionario con los valores leídos por los sensores.

```
24
25      decoded = base64.b64decode(base64_input)
26
27      byteArray = list(decoded)
28
29      temp = (byteArray[0] << 8 | byteArray[1] ) / 100
30      rel_hum =  (byteArray[2] << 8 | byteArray[3] )
31      pm25_standard = (byteArray[4] << 8 | byteArray[5] )
32      pm100_standard =  (byteArray[6] << 8 | byteArray[7] )
33      uv = (byteArray[8] << 8 | byteArray[9] ) / 100
34      ir =  (byteArray[10] << 8 | byteArray[11] )
35      view = (byteArray[12] << 8 | byteArray[13] )
36      volt =  (byteArray[14] << 8 | byteArray[15] ) / 1000
37      timestamp =  (byteArray[16] << 8 | byteArray[19] )
38
39      result = {
40          "temperature": temp,
41          "humidity": rel_hum,
42          "pm2.5": pm25_standard,
43          "pm10": pm100_standard,
44          "ultraviolet": uv,
45          "infrared": ir,
46          "visible": view,
47          "voltage": volt,
48          "timestamp": timestamp
49      }
50
51      return result
```

Figura 3.43 – Decodificación del payload

2. Dar permisos a AWS IoT para invocar la función

Si accedemos a la función lambda, podemos cambiar su configuración y añadir permisos de ejecución para que sea posible invocarla desde otros servicios de AWS (Figura 3.44). De entre las opciones disponibles, seleccionamos *AWS service* para garantizar permisos a un servicio y en el desplegable *Service* escogemos AWS IoT, lo que hará que los campos *Principal* y *Source ARN* se rellene automáticamente. Elegimos un nombre como ID único y seleccionamos que queremos invocar la función Lambda en el campo *Action*.

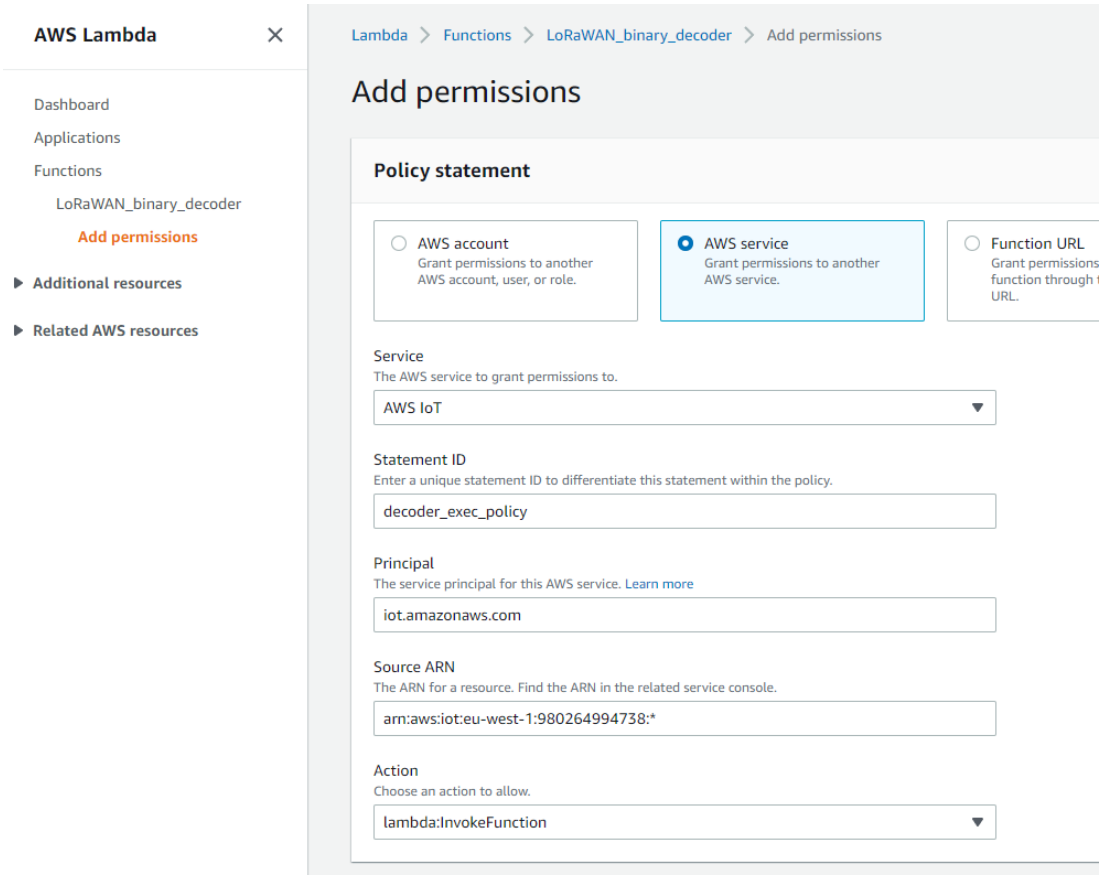


Figura 3.44 – Permisos de la función Lambda

Una vez creada la función copiamos su ARN (*Amazon Resource Name*), para poder referenciarla en el paso siguiente.

3. Crear un Regla en AWS IoT que ejecute la función

Una **Regla** (*Rule*) le dice a AWS IoT qué hacer cuando recibe mensajes de los dispositivos. Pueden extraer datos de los mensajes, evaluar expresiones usando datos de mensajes e invocar una o más acciones cuando se cumplen las condiciones que impone la propia Regla.

Para la creación de ésta, accedemos a la consola de AWS IoT, en el apartado *Act*, y seleccionamos *Rules*. Debe tener el mismo nombre (*LoRaWAN_messages_rule*) que se utilizó en el momento en el que se creó el Destino (Figura 3.45).

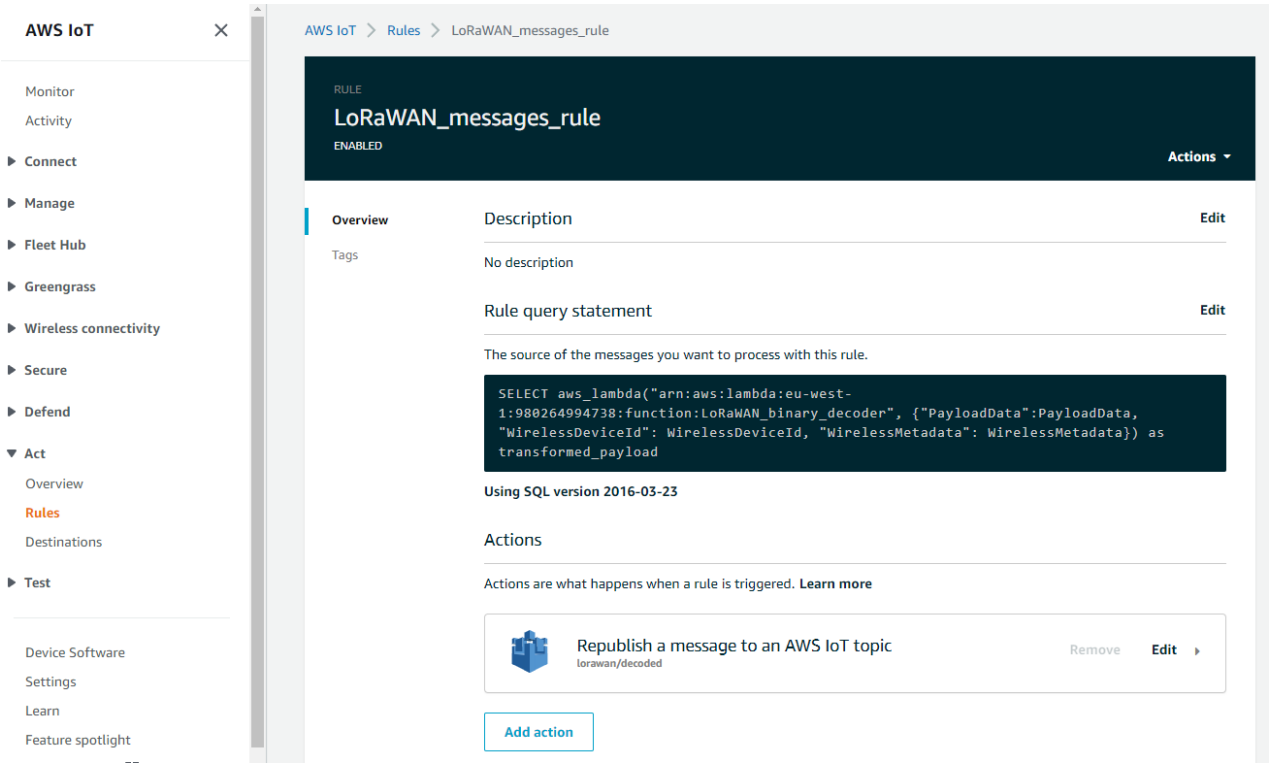


Figura 3.45 – Creación de una Regla (Rule)

Hay dos campos de configuración importantes, el primero de ellos es *Rule query statement*, donde se utiliza el formato SQL para filtrar la información proveniente del broker MQTT y a la vez invocar la función Lambda mediante su ARN. La consulta (*query*) a realizar es de la siguiente forma:

SELECT aws_lambda("arn:aws:lambda:eu-west-1:980264994738:function:LoRaWAN_binary_decoder", {"PayloadData":PayloadData, "WirelessDeviceId": WirelessDeviceId, "WirelessMetadata": WirelessMetadata}) as transformed_payload

El segundo campo de configuración es *Actions*, que nos permite elegir entre una gama preconfigurada de Acciones relacionadas con distintos servicios de AWS. En nuestro caso elegimos publicar el objeto JSON a un topic del broker MQTT, mediante la Acción *Republish a message to an AWS IoT topic*. Durante su configuración (Figura 3.46), se debe elegir el nombre del topic de salida (*lorawan/decoded*) y la calidad del servicio (QoS 1). También debemos elegir el nombre de un Rol que será creado automáticamente con las políticas necesarias.

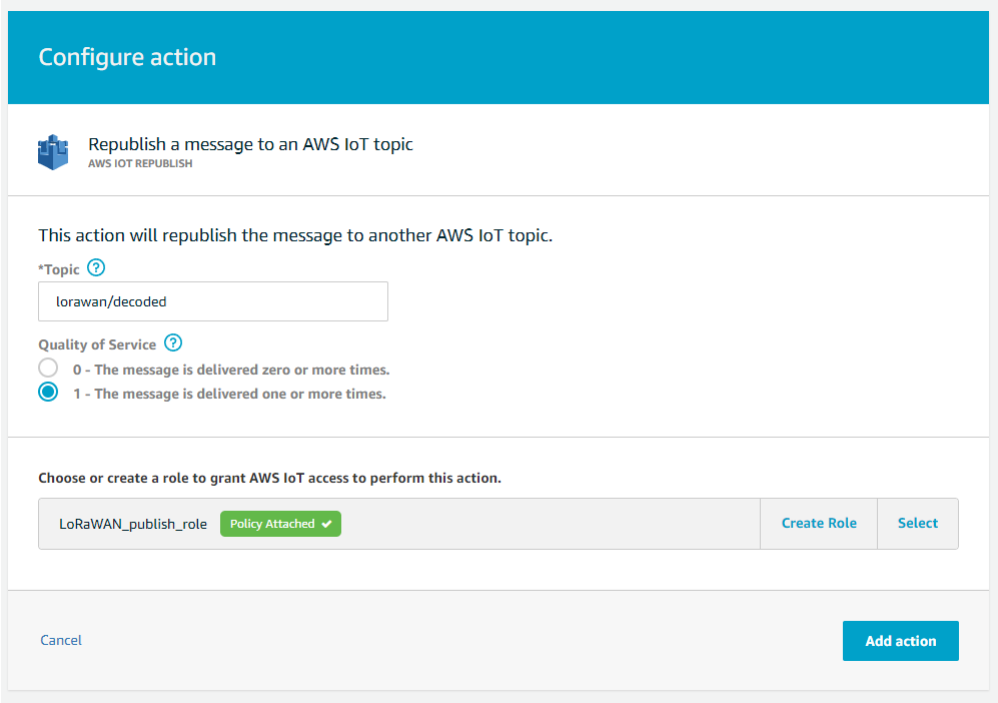


Figura 3.46 – Configuración de la Acción en LoRaWAN_messages_rule

Para comprobar el correcto funcionamiento de las configuraciones realizadas hasta este punto, se puede usar el cliente MQTT de pruebas en la sección *Test* dentro de AWS IoT. Si nos subscribimos al topic creado, recibiremos los mensajes que contienen el payload decodificado (Figura 3.47).

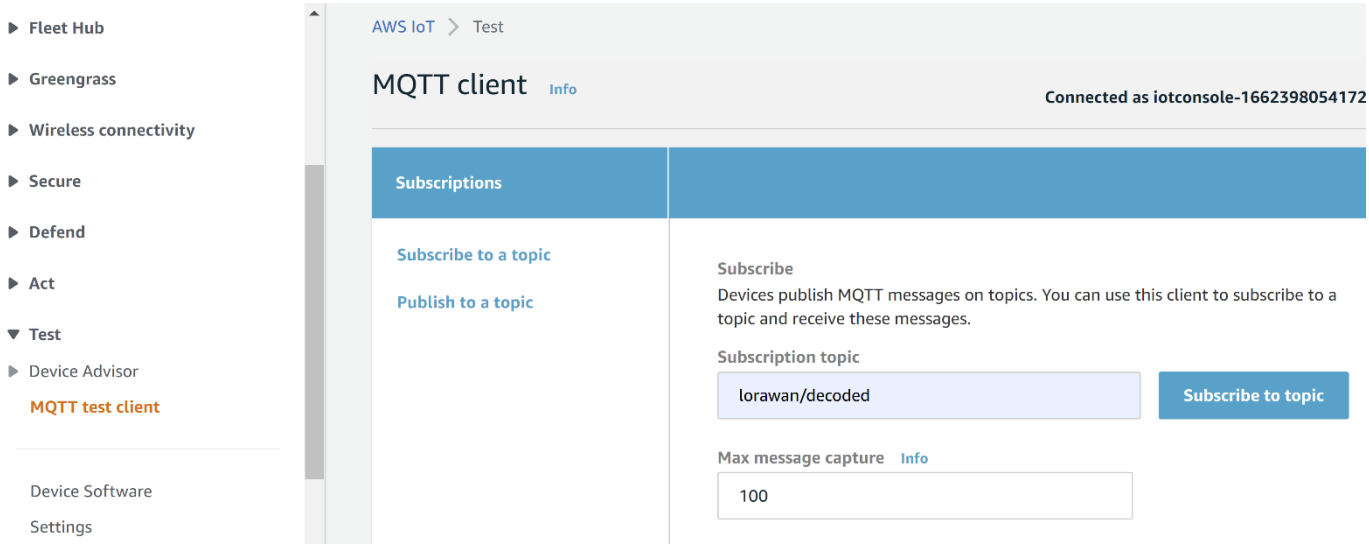


Figura 3.47 – Cliente MQTT para testeo

El formato JSON de los mensajes recibidos en el topic *lorawan/decoded* es el mostrado a continuación.

```
{
 "status": 200,
 "LNSData": {
    "PayloadData": "CboAJgAHAAcAAgD+AQQOkGKZNYo=",
    "WirelessDeviceId": "e735d508-cf9a-45cd-a7bb-8b42013d0442",
    "WirelessMetadata": {
          …
     }
 },
 "TransformedPayloadData": {
       "temperature": 24.9,
       "humidity": 38,
       "pm25": 7,
       "pm100": 7,
       "ultraviolet": 0.02,
       "infrared": 254,
       "visible": 260,
      "voltage": 3.728,
      "timestamp": 1654207882
  }
}
```


3.3.2 Almacenamiento

Los valores decodificados de los sensores se almacenarán en la base de datos de series temporales AWS Timestream. Si entramos en su consola de uso, podemos acceder al apartado *Databases* y crear una nueva base de datos (Figura 3.48). En su configuración seleccionamos *Standard database* y le damos un nombre representativo (*metroForestDB*).

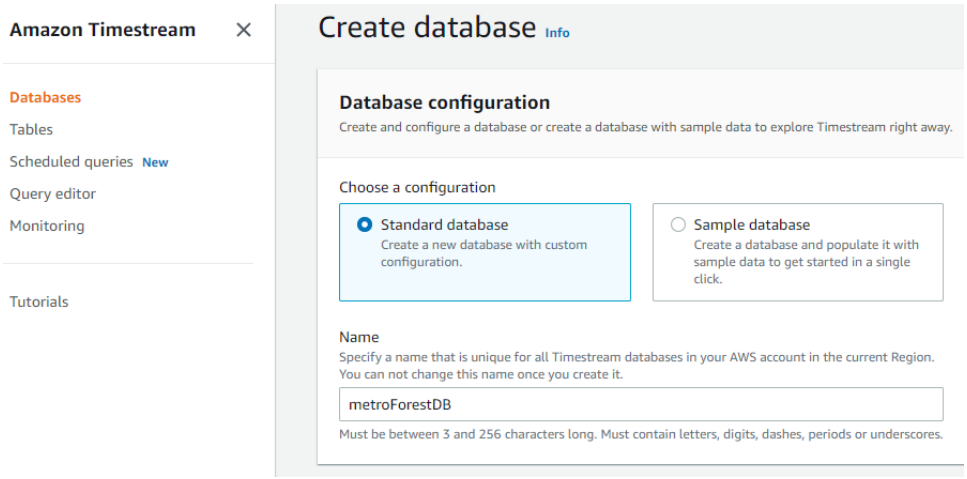


Figura 3.48 – Creación de la base de datos

Para crear una tabla dentro de esta base de datos debemos acceder al apartado *Tables* (Figura3.49). En la configuración de la nueva tabla indicamos que pertenece a *metroForestDB* y le damos un nombre (*nodes*). También podemos seleccionar el tiempo máximo que estarán guardados los datos en memoria y en soporte magnético. La diferencia entre ambos es que la información que esté almacenada en memoria es más rápida de consultar.

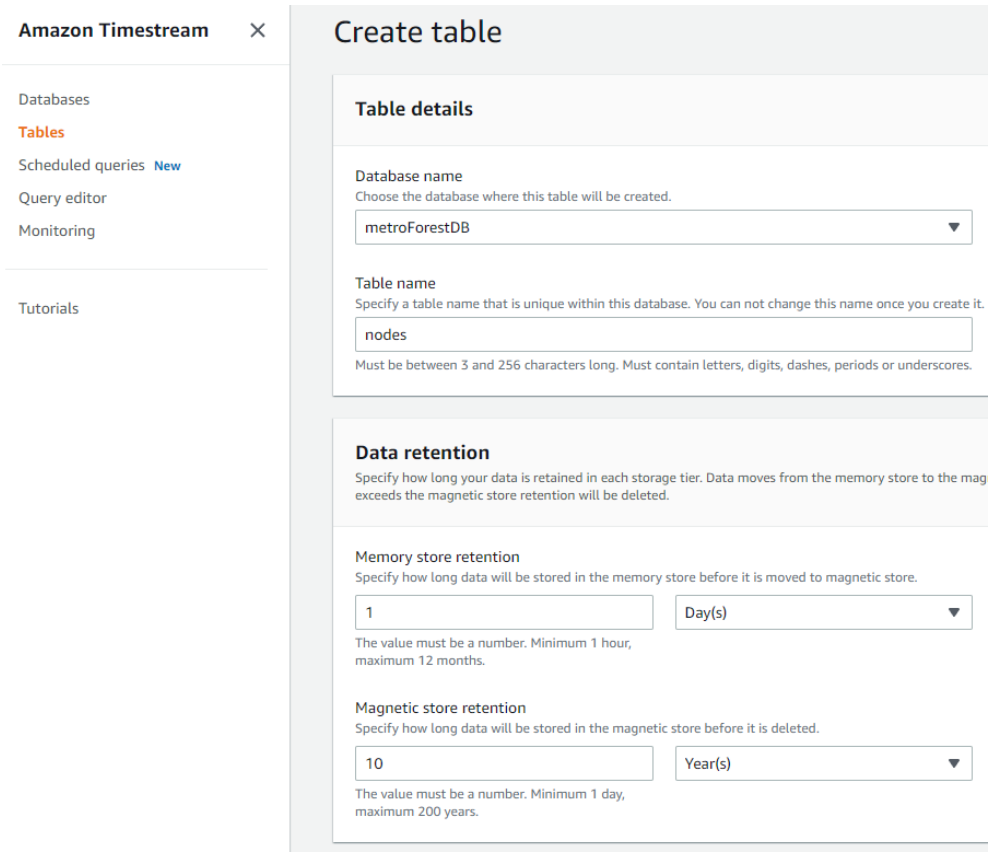


Figura 3.49 – Creación de la tabla en la base de datos

Una vez que se ha configurado Amazon Timestream, accedemos a la consola de AWS IoT para crear un nueva Regla (*LoRaWAN_to_Timestream_rule*), de forma similar a la que se ha creado anteriormente, que guardará los valores de los sensores en la base de datos (Figura 3.50).

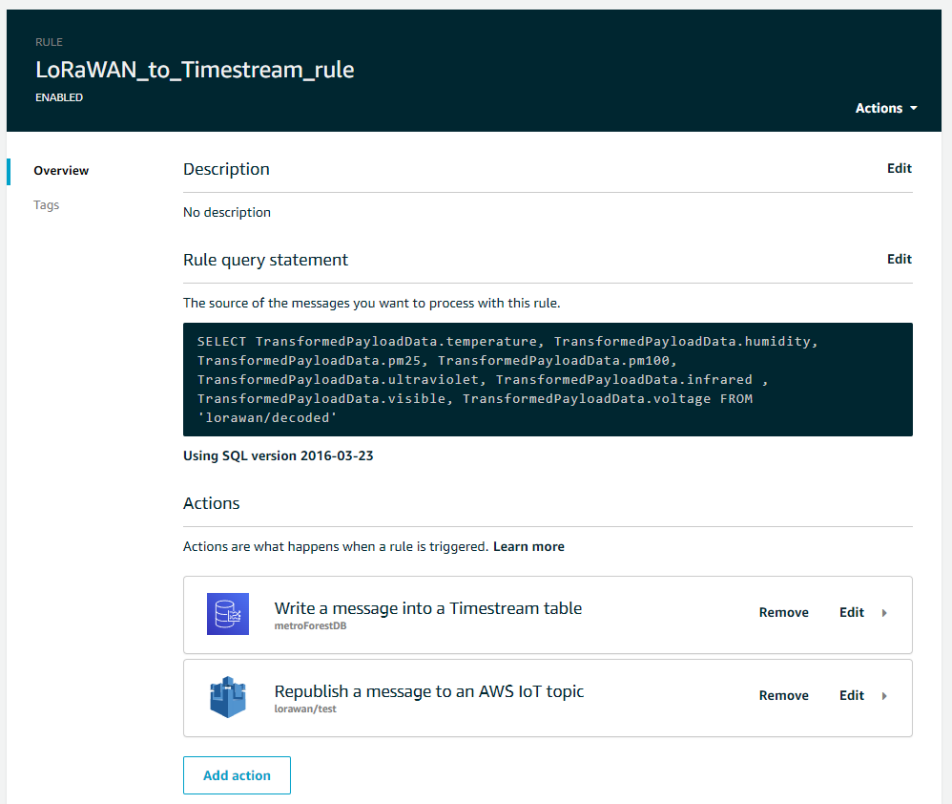


Figura 3.50 – Creación de la Regla LoRaWAN_to_Timestream_rule

La consulta SQL hecha en el campo *Rule query statement* contiene, en esta ocasión, el comando *FROM* para especificar el topic de entrada, ya que es diferente del utilizado por defecto en AWS IoT. La consulta realizada queda de la siguiente forma:

SELECT TransformedPayloadData.temperature, TransformedPayloadData.humidity, TransformedPayloadData.pm25, TransformedPayloadData.pm100, TransformedPayloadData.ultraviolet, TransformedPayloadData.infrared, TransformedPayloadData.visible, TransformedPayloadData.voltage FROM 'lorawan/decoded'

En el campo *Actions*, la Acción preconfigurada que necesitamos en esta ocasión tiene como nombre *Write a message into a Timestream table* (Figura 3.51), donde seleccionamos la base de datos y la tabla donde queremos realizar la escritura. El campo *Dimensions* representa los

metadatos que se asociarán con cada uno de los valores guardados, donde la mejor opción es incluir el parámetro *dev_eui* con el fin de realizar consultas individuales en cada nodo. Se especifica que el campo *Timestamp*, que sirve de índice en la tabla, sea el valor proporcionado por el nodo en el momento de la lectura. Si no se indica, se utilizaría por defecto el momento en el que se guardan los datos, que da lugar a errores temporales por la latencia de la red o la disponibilidad de los servicios cloud. Por último, introducimos el nombre del Rol (*decoder_to_Timestream_role*) que se creará por defecto.

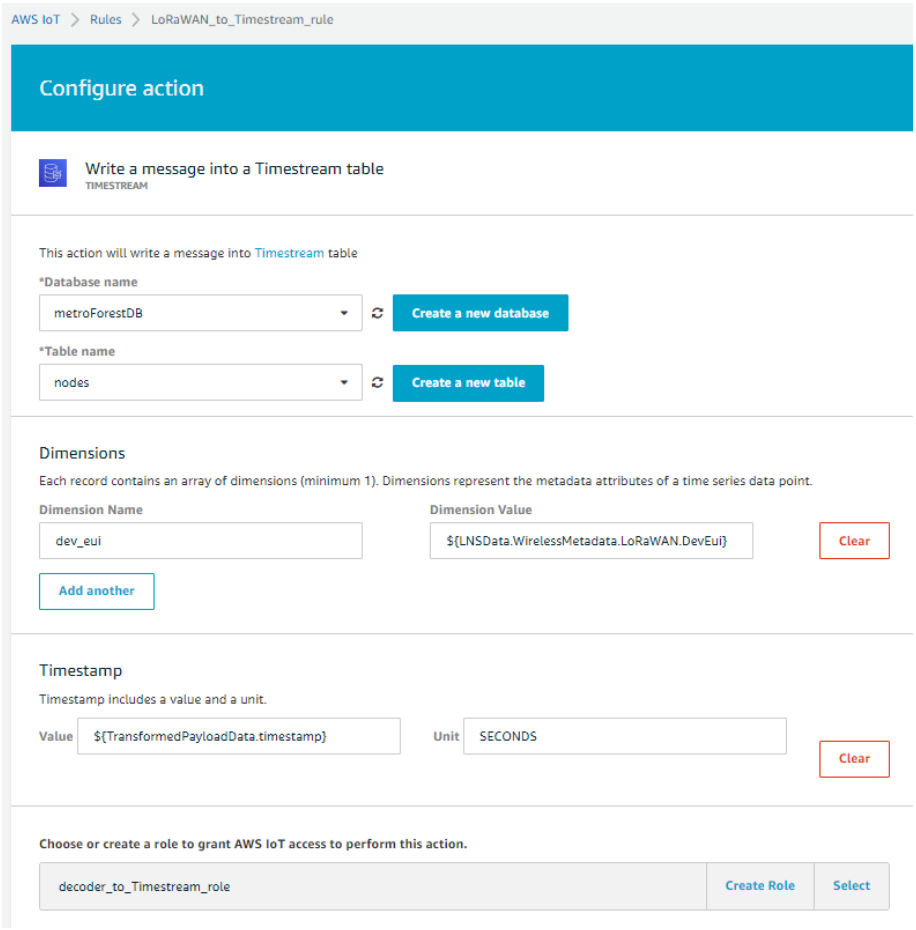


Figura 3.51 – Configuración de la Acción en LoRaWAN_to_Timestream_rule

Aunque podría parecer que esta Acción se debiera incluir en la Regla ya creada (*LoRaWAN_messages_rule*), AWS IoT no permite acceder a los datos internos del resultado dado por una función Lambda desde la misma Regla que lo invoca, por lo que es necesario utilizar un topic intermedio, que en este caso es *lorawan/decoded*.

Para verificar que los datos son almacenados correctamente, podemos usar el cliente MQTT de pruebas y publicar en el anterior topic el mensaje, con formato JSON, recibido en el testeo anterior (Figura 3.52).

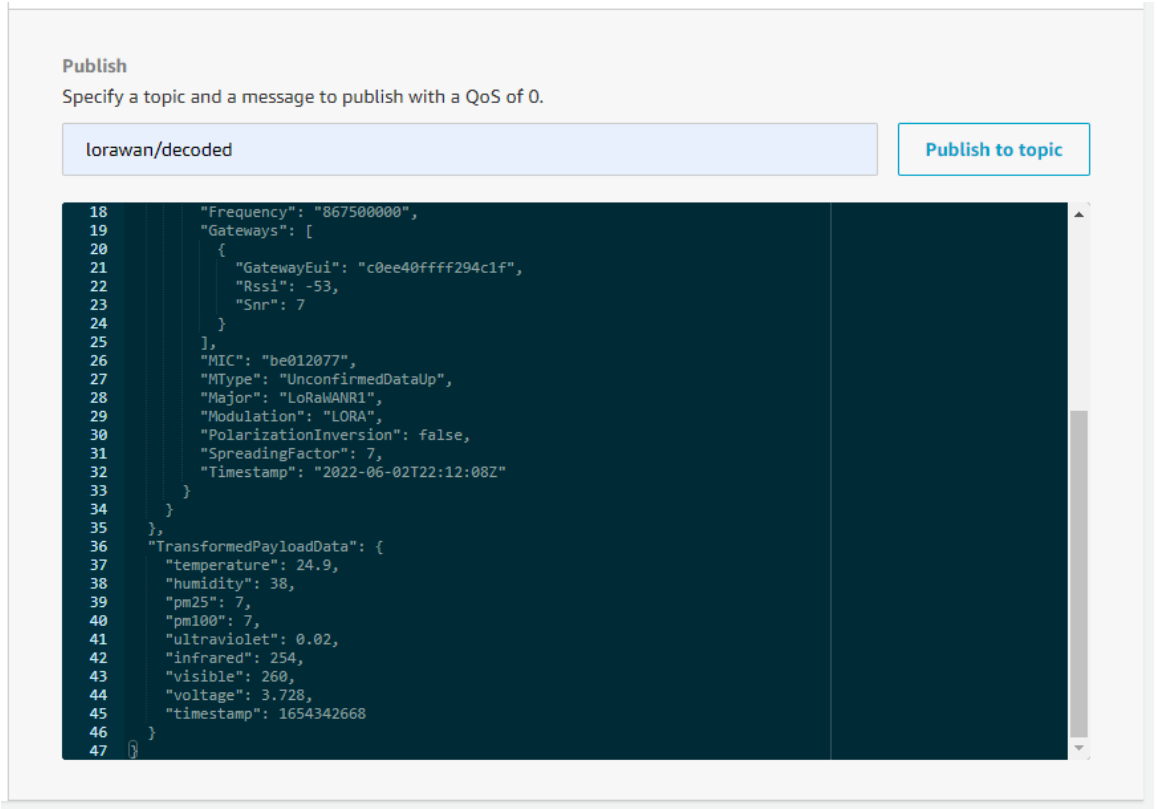


Figura 3.52 – Mensaje MQTT para el test de la base de datos

Si accedemos a la tabla desde Amazon Timestream, podemos verificar que el mensaje anterior se ha guardado correctamente (Figura 3.53).

Table details | Query results | Output

Rows returned (8)

Filter

1

dev_eui	measure_name	time	measure_value::bigint	measure_value::double
70b3d54991ea6f2e	pm25	2022-06-04 11:37:48.000000000	7	-
70b3d54991ea6f2e	infrared	2022-06-04 11:37:48.000000000	254	-
70b3d54991ea6f2e	ultraviolet	2022-06-04 11:37:48.000000000	-	0.02
70b3d54991ea6f2e	voltage	2022-06-04 11:37:48.000000000	-	3.728
70b3d54991ea6f2e	temperature	2022-06-04 11:37:48.000000000	-	24.9
70b3d54991ea6f2e	visible	2022-06-04 11:37:48.000000000	260	-
70b3d54991ea6f2e	pm100	2022-06-04 11:37:48.000000000	7	-
70b3d54991ea6f2e	humidity	2022-06-04 11:37:48.000000000	38	-

Figura 3.53 – Datos almacenados en Amazon Timestream

4. Resultados

Aunque los datos almacenados en Timestream pueden consultarse en ese mismo servicio, obteniéndolos en forma de tabla, es recomendable realizar gráficas con ellos para facilitar de manera visual su análisis. La herramienta utilizada para ello es Grafana, que se encuentra disponible dentro del servicio Amazon Managed Grafana.

Su configuración requiere acceder a la consola del servicio y desde ahí crear un Espacio de Trabajo (*Workspace*). En el primer paso debemos darle un nombre representativo, que en nuestro caso ha sido *Metropolitan_forest* (Figura 4.1).

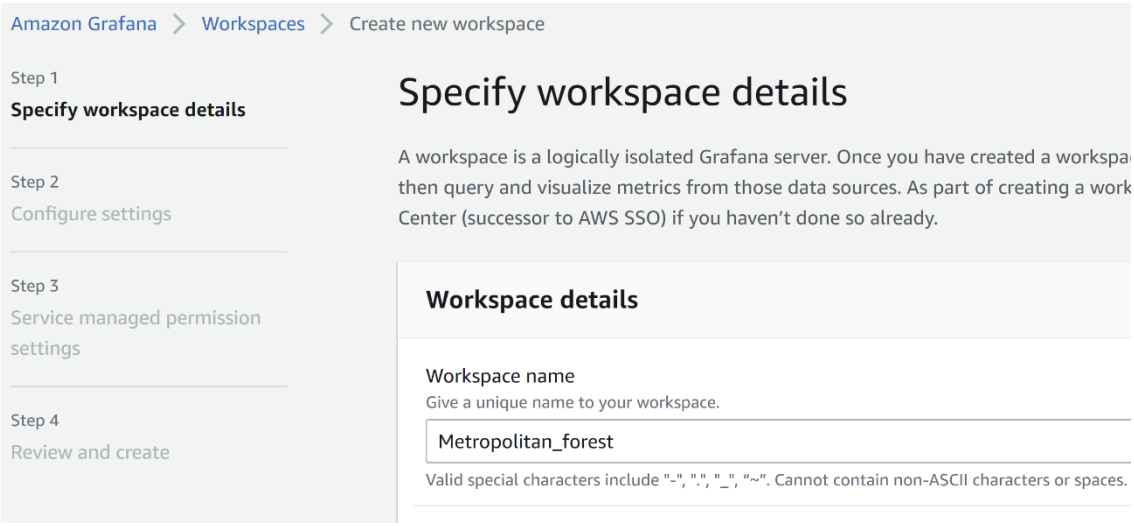


Figura 4. 1 – Creación del Espacio de trabajo para Grafana

En el siguiente paso configuramos los ajustes necesarios (Figura 4.2). En el campo inferior *Permission type* seleccionamos *Service managed*, que añadirá de forma automática los permisos necesarios. En el campo superior *Authentication access* marcamos el método de autentificación como *AWS IAM Identify Center* y nos aparecerá una alerta, de color rojo en la imagen, donde nos indica que el servicio no está habilitado. Se nos da la opción de crear un nuevo usuario para este servicio, lo que automáticamente lo habilitará para que pueda ser utilizado. En el momento de la creación del usuario se nos pedirá su correo electrónico y nombre completo. AWS IAM Identify Center envía entonces un email de activación, en el que se solicita escoger una contraseña, que debe ser verificado antes de acceder al Espacio de Trabajo con este usuario.

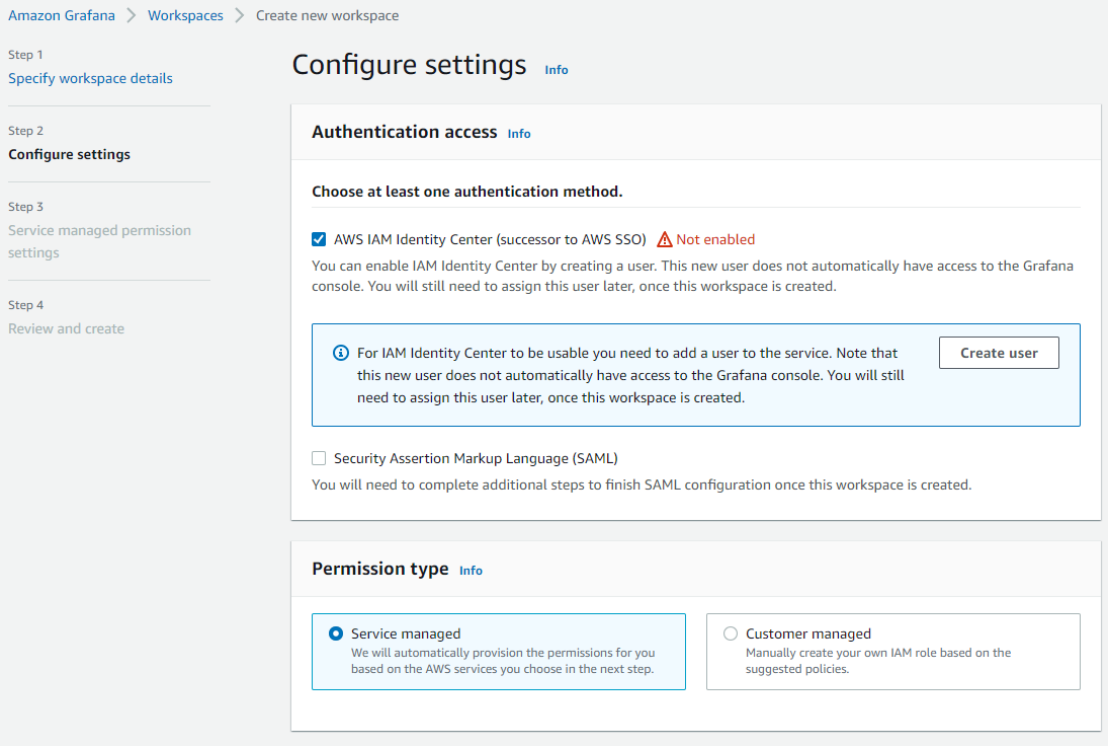


Figura 4.2 – Configuración del Espacio de trabajo

En el último paso de configuración (Figura 4.3), seleccionamos los permisos de acceso IAM mediante la opción *Current account* y la fuente de datos a *Amazon Timestream*.

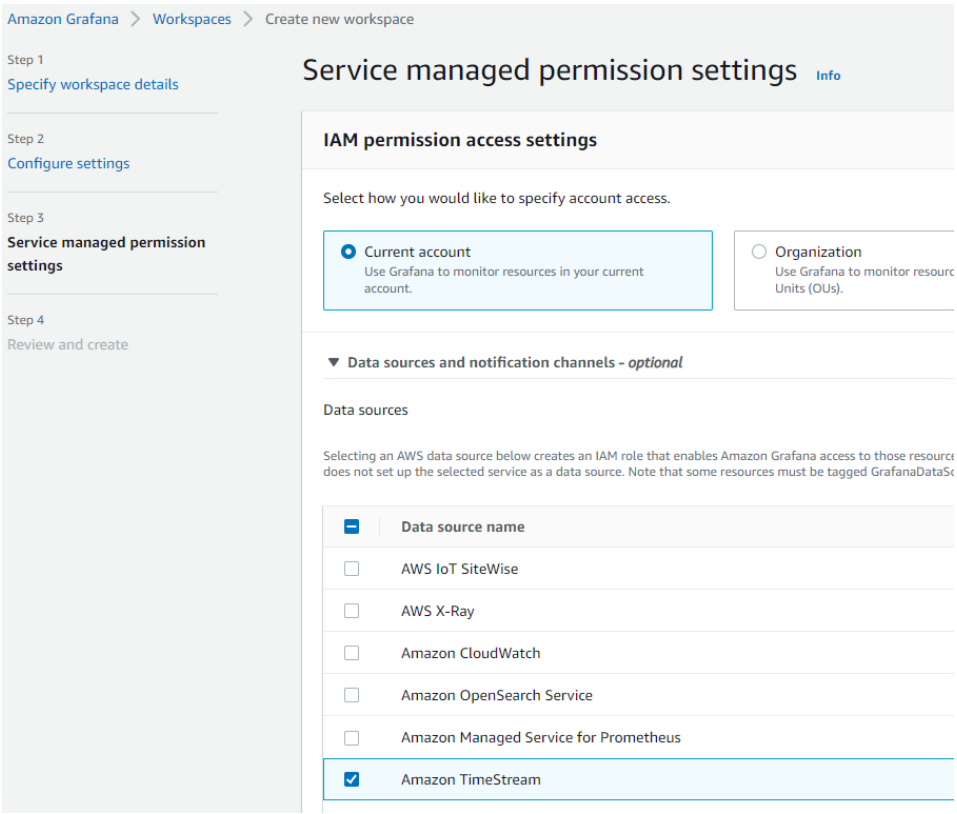


Figura 4.3 – Permisos de acceso a Timestream

Una vez que se ha creado el Espacio de Trabajo accedemos a él y verificamos que su estado es *Active* (Figura 4.4). Accedemos a *Configure users and user groups* para indicar los usuarios de AWS IAM Identify Center que pueden acceder a este Workspace.

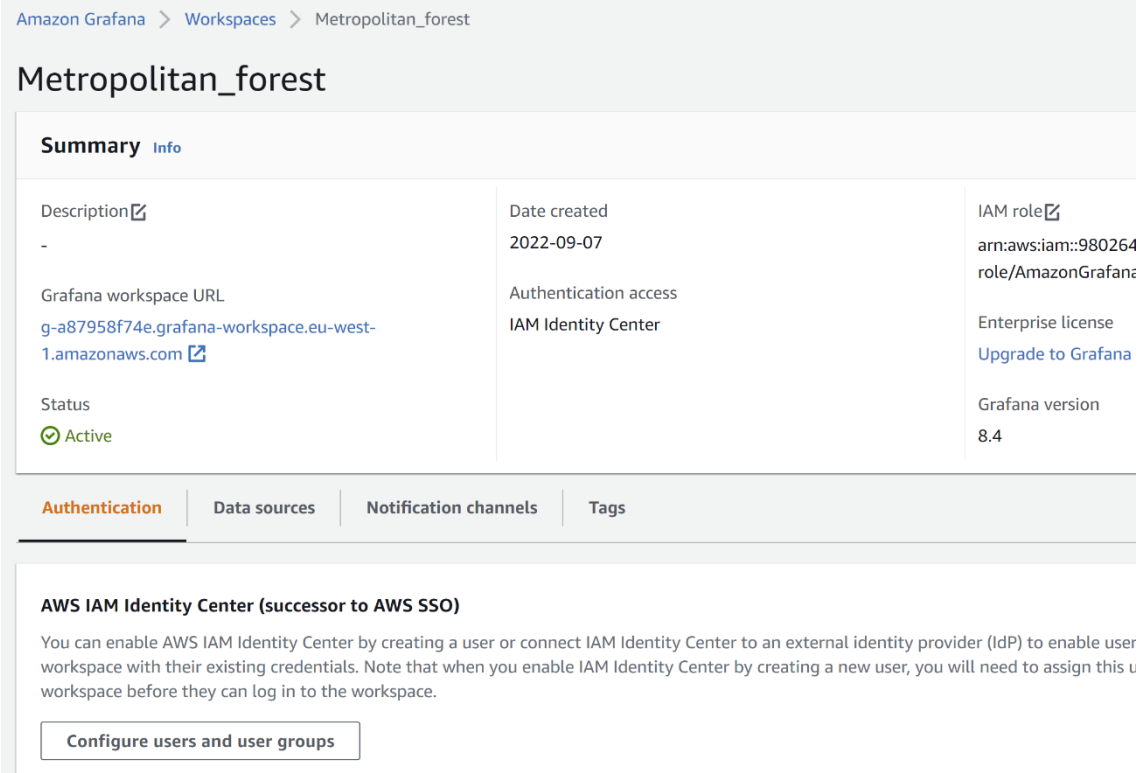


Figura 4.4 – Configuración de los usuarios para el Espacio de trabajo

Añadimos el usuario creado anteriormente y lo seleccionamos (Figura 4.5) para cambiar su tipo de *Viewer* a *Administrator* mediante la acción *Make admin* en el desplegable *Action*.

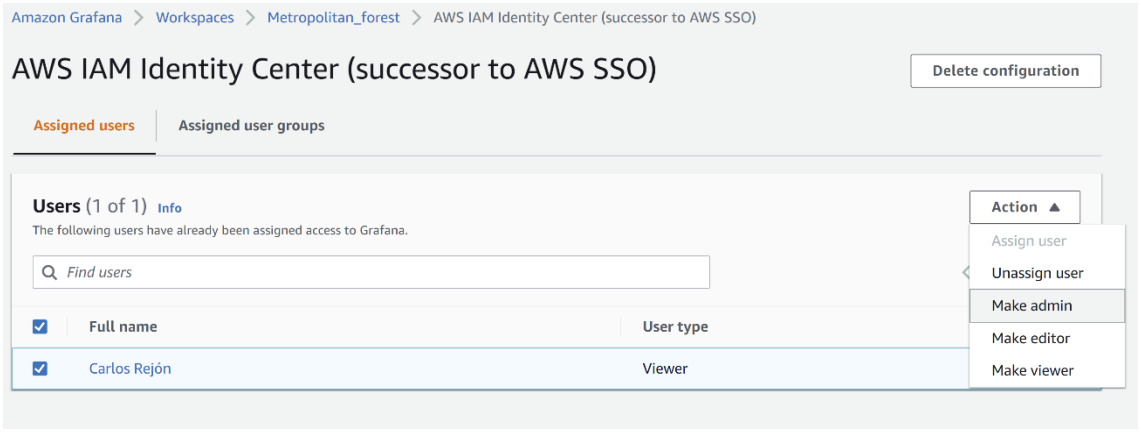


Figura 4.5 – Selección de administrador para el Espacio de Trabajo

Después de realizar esta configuración, ya podemos acceder a Grafana mediante la URL generada para este Espacio de Trabajo (*Grafana workspace URL* en la Figura 4.4), donde se nos pedirá nuestro usuario y contraseña.

Una vez dentro de Grafana, accedemos al menú para la selección de la Fuentes de Datos (*Data Sources*) y configuramos Timestream (Figura 4.6), de esta forma ya podemos realizar las gráficas necesarias.

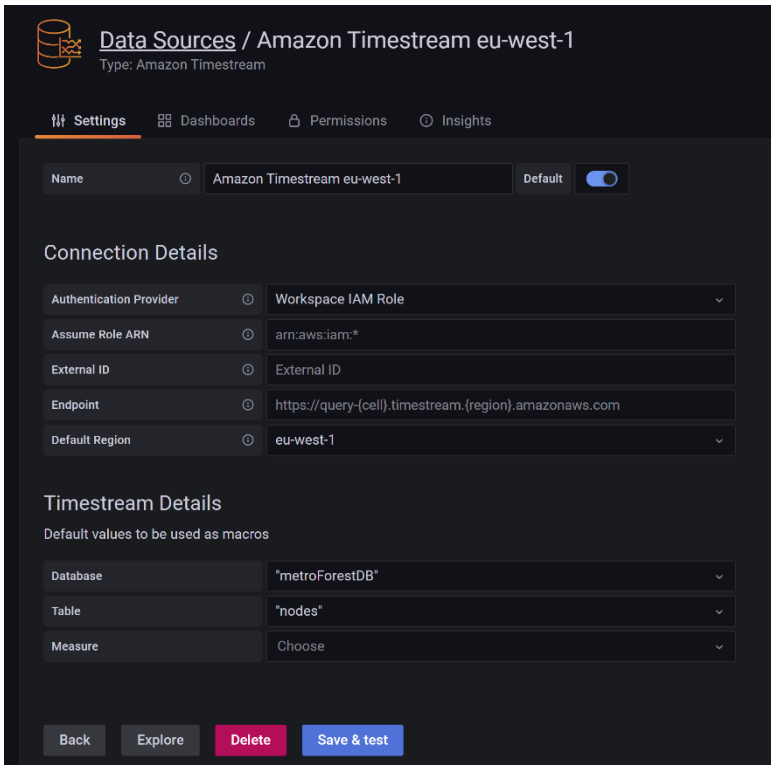


Figura 4.6 – Configuración de la fuente de datos en Grafana

Figura 4.7 – Gráficas de la temperatura y humedad relativa

El rango de tiempo seleccionado para la visualización de los datos en las gráficas es de un mes completo, del 1 de julio al 1 de agosto. En la Figura 4.7 se pueden ver los valores de la temperatura y la humedad relativa proporcionados por el sensor Si7021. De ellos podemos decir que la temperatura marca valores demasiado elevados, ya que supera los 50 ºC en varias ocasiones. Esto es debido a que el sensor se encuentra debajo de la placa solar, por lo que hubiese sido mejor ubicarlo en otra posición.

En la Figura 4.8 se muestran los valores de la luz visible, infrarroja y ultravioleta obtenidos mediante el sensor SI1145, donde los valores son bastantes estables en general, aunque se pueden observar dos mediciones erróneas (*outliers*) en la luz ultravioleta los días 17 y 20.

Figura 4.8 – Gráficas de la luz visible, infrarroja y visible

En la Figura 4.9 se muestran las gráficas creadas con los valores obtenidos del sensor de partículas PMS5003, que normalmente se encuentran por debajo de los 100 µg/m³, aunque en los días 8 y 26 se obtuvieron valores inusualmente altos, por encima de los 300.

Figura 4.9 – Gráficas de las partículas PM2.5 y PM10

En la Figura 4.10 observamos los valores leídos por el monitor de batería, donde el voltaje con carga máxima es de unos 4,19 V y el mínimo nunca baja de los 4,12 V, tensión más que suficiente para alimentar el nodo de forma correcta.

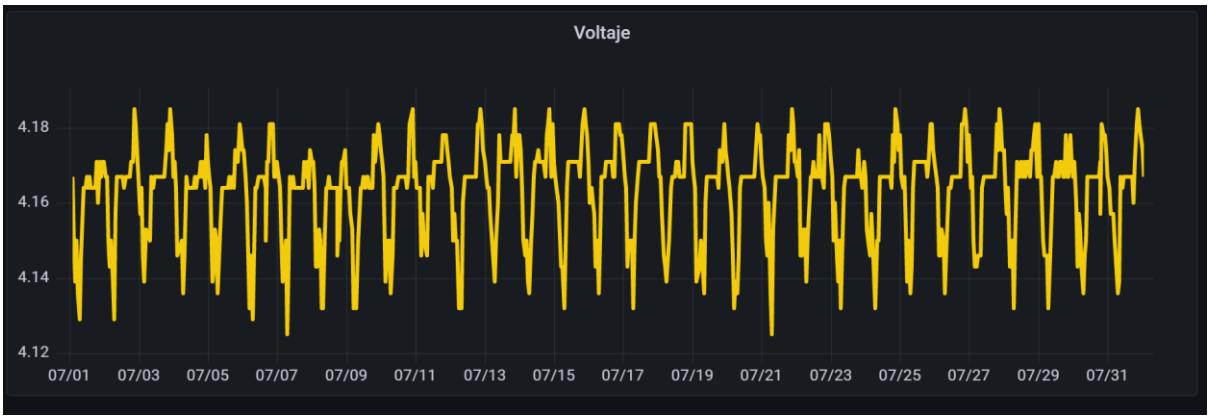


Figura 4.10 – Gráfica del voltaje de la batería

5. Conclusiones y trabajo futuro

En este último capítulo se presentan las principales conclusiones obtenidas después de la elaboración del presente trabajo y las posibles propuestas de trabajos futuros que pueden llevarse a cabo.

5.1 Conclusiones

En este trabajo se ha podido observar la viabilidad de desarrollar un nodo de medición ambiental y calidad del aire, con alimentación autónoma y comunicación a larga distancia, para proyectos de índole medioambiental como es el Bosque Metropolitano de Madrid.

Tanto el panel solar como la batería seleccionada han sido los adecuados para el consumo energético de la placa de desarrollo y todos los sensores utilizados, lo que posibilitaría incluir si se quisiera más sensores sin necesidad de cambios en la alimentación.

La ubicación del sensor de temperatura y humedad no ha sido la más correcta, debido a que se producen picos elevados de temperatura por encontrarse debajo de la placa solar. En un nuevo rediseño del nodo sería necesario reubicar este sensor en otra posición.

Como el sensor de luz produce valores bastante estables, podría utilizarse en conjunto con el voltaje de la batería para detectar posibles fallos en la placa solar, ya que una elevada radiación solar siempre debe implicar una carga alta de la batería.

5.2 Trabajo futuro

A continuación, se detallan las propuestas de mejora y continuación que podría realizarse en este proyecto.

- Inclusión de más sensores ambientales (NO_2, SO_2, O_3, CO …)
- Creación de un servicio de aprovisionamiento para el resto de los nodos
- Testeo de la carga de la batería en épocas de menor radiación solar
- Implementación de un sistema de alerta en el caso de desconexión de algún nodo
- Tratamiento de las series temporales de datos con técnicas de *Deep Learning* para la detección de anomalías o predicción de valores futuros (*forecasting*)

6. Introduction

This first introductory chapter sets out the motivation for this project, the proposed objectives to be achieved and the organisation presented in the report.

6.1 Motivation

Mediterranean cities are particularly vulnerable to climate change according to recent reports from international organisations which show that the temperature increase in the Mediterranean area is 20% above the global average with a risk of a 3.5ºC rise by 2080 with serious implications for desertification and loss of ecosystems.

Madrid City Council is currently implementing a project to create a forest belt around the city called the Metropolitan Forest. It involves a new deployment of green infrastructure consisting of planting native forest species on available municipal land, contributing both to environmental improvement and to the ecological and landscape restoration of degraded areas. Other cities such as London and Copenhagen led the way in the 1930s and 1940s. In Spain, we had to wait until 1993 for Vitoria to initiate its Green Belt.

This forest ring covers 14,200 hectares and foresees the planting of up to 450,000 trees of native species over the next 12 years, which will add up to 5,900 net hectares of trees to the city (the equivalent of 51 times El Retiro). It will connect the protected natural areas of El Pardo to the north with the lower reaches of the Manzanares and Jarama rivers to the southeast.

Although the project contemplates the creation of paths and leisure areas, their role is secondary, as it is not an urban park. Its mission is to improve air quality and reduce the "heat island" effect that occurs in Madrid during the summer months due to buildings, asphalt and different human activities.

The development of the Metropolitan Forest will provide, among others, the following environmental and social benefits:

- Improving air quality
- Improving urban biodiversity
- Contribution to climate change mitigation and adaptation in the city

• Promoting the health and well-being of citizens through the promotion of leisure and sporting activities in the natural environment.

• Improving the landscape and enhancing the value of nearby consolidated and planned areas.

6.2 Objectives

In the current context of climate change, which unfortunately becomes more visible year after year, being able to plausibly corroborate how the reforestation of degraded areas improves the habitat, and therefore the quality of life of all the people who coexist with them, can encourage policies that are aimed at getting more municipalities to adopt this type of measures.

The main objective of this project is the design, assembly and functional testing of an autonomous node capable of verifying the environmental goals sought with the implementation of the Metropolitan Forest of Madrid. Among them we can highlight the reduction of temperature in hot periods, the increase of relative humidity and the reduction of microscopic particles in suspension.

The IoT system to be developed shall comply with the following specifications:

- Measurement of environmental parameters and air quality
- Autonomous power supply of the nodes
- Long distance communication with low transmission rate
- Data management and storage

To verify correct operation, the node shall be placed in an outdoor location for a period of 1 month and the data obtained shall be studied by means of time graphs.

6.3 Organization of the report

The document is made up of the sections described below.

This first chapter serves as an introduction, describing the motivation of the project and the objectives to be achieved.

The second chapter will give an overview of the technologies and devices that allow the system to be developed.

The third chapter will focus on the creation of the nodes and the configuration of the cloud platform.

The fourth chapter presents the results obtained by creating graphs with the values captured by the sensors.

The fifth chapter presents the conclusions of the project and possible future work that can be carried out.

Finally, we find the bibliographical references used.

7. Conclusions and future work

This last chapter presents the main conclusions drawn from this work and possible proposals for future work.

7.1 Conclusions

This work has shown the feasibility of developing an environmental and air quality measurement node, with autonomous power supply and long-distance communication, for environmental projects such as the Metropolitan Forest of Madrid.

Both the solar panel and the battery selected were suitable for the energy consumption of the development board and all the sensors used, which would make it possible to include more sensors if required without the need to change the power supply.

The location of the temperature and humidity sensor has not been the most correct, due to the high temperature peaks produced by being underneath the solar panel. In a new redesign of the node, it would be necessary to relocate this sensor to another position.

As the light sensor produces stable values, it could be used in conjunction with the battery voltage to detect possible faults in the solar panel, as high solar radiation should always mean a high battery charge.

7.2 Future work

The following are the proposals for improvement and continuation that could be made in this project.

- Inclusion of more environmental sensors (NO2, SO2, O3, CO ...)
- Creation of a provisioning service for the rest of the nodes
- Testing the battery charge in a situation of lower solar radiation
- Implementation of an alert system in case of disconnection of any node
- Treatment of time series data with Deep Learning techniques to detect anomalies or predict future values (*forecasting*)

8. Referencias bibliográficas

[1] Alberto Hernández, Pablo R. Roces. El Mundo. "Así será la M-40 verde que rodeará Madrid". URL: https://lab.elmundo.es/bosque-metropolitano-madrid/index.html

[2] Ayuntamiento Madrid. Bases concurso Bosque Metropolitano. URL: https://estrategiaurbana.madrid.es/bases-concurso-bosque-metropolitano/

[3] The Things Network. LoraWAN. URL: https://www.thethingsnetwork.org/docs/lorawan/

[4] Catsensors. Tecnología LoRa y LoRaWAN. URL: https://www.catsensors.com/es/lorawan/tecnologia-lora-y-lorawan

[5] Semtech. LoRa basics for gateways. URL: https://lora-developers.semtech.com/build/software/lora-basics/lora-basics-for-gateways

[6] Libelium. Plug Sense. URL: https://www.libelium.com/iot-products/plug-sense/

[7] Pycom. LoPy 4. URL: https://pycom.io/product/lopy4/

[8] MicroPython. URL: https://micropython.org

[9] AWS IoT Core. URL: https://aws.amazon.com/es/iot-core/

[10] AWS Timestream. URL: https://aws.amazon.com/es/timestream/

[11] Amazon Manage Grafana. URL: https://aws.amazon.com/es/grafana/

[12] Docs Pycom. LoPy 4 datasheet. URL: https://docs.pycom.io/datasheets/development/lopy4/

[13] Docs Pycom. Expansion boards datasheet. URL: https://docs.pycom.io/datasheets/expansionboards/expansion3/

[14] Pimoroni. Batería LiPo. URL: https://shop.pimoroni.com/products/lipo-battery-pack?variant=20429082247

[15] Adafruit. Panel solar. URL: https://www.adafruit.com/product/200

[16] Adafruit. Cargador bq24074. URL:https://learn.adafruit.com/adafruit-bq24074-universal-usb-dc-solar-charger-breakout

[17] Adafruit. Sensor SI1145. URL: https://learn.adafruit.com/adafruit-si1145-breakout-board-uv-ir-visible-sensor

[18] Librería Sensor SI1145. URL: https://github.com/neliogodoi/MicroPython-SI1145

[19] Adafruit. Sensor Si7021. URL: https://learn.adafruit.com/adafruit-si7021-temperature-plus-humidity-sensor

[20] Librería Sensor Si702. URL: https://github.com/chrisbalmer/micropython-si7021

[21] Pimoroni. Sensor PMS5003. URL: https://shop.pimoroni.com/products/pms5003-particulate-matter-sensor-with-cable?variant=29075640352851

[22] Librería Sensor PMS5003. URL: https://github.com/pimoroni/pms5003-python

[23] Pimoroni. Breakout para sensor PMS5003. URL: https://shop.pimoroni.com/products/particulate-matter-sensor-breakout?variant=29493578301523

[24] Adafruit. Monitor de batería LC709203F. URL: https://learn.adafruit.com/adafruit-lc709203f-lipo-lipoly-battery-monitor

[25] Librería Monitor de batería LC709203F. URL: https://github.com/scopelemanuele/Micropython-LC709203F

[26] Elevador de tensión dc/dc MT3608. URL: https://www.electrocomponentes.es/m%C3%B3dulos-reguladores/802-mt3608-modulo-elevador-de-tension-ajustable-step-up-boost-2a-dc-dc.html

[27] Librería para el RTC DS3231 URL: https://github.com/micropython-Chinese-Community/mpy-lib/tree/master/misc/DS3231

[28] Código del proyecto. URL: https://github.com/cRejon/MDF_Project

[29] Mapa topográfico de la Comunidad de Madrid. URL: https://es-es.topographic-map.com/maps/6och/Comunidad-de-Madrid/

[30] Red de estaciones fijas de control de calidad del aire. Ayuntamiento de Madrid. URL: https://www.madrid.es/portales/munimadrid/es/Inicio/Medio-ambiente/Direcciones-y-telefonos/Red-de-estaciones-fijas-de-control-de-calidad-del-aire/?vgnextfmt=default&vgnextoid=aeca16d591bbf710VgnVCM2000001f4a900aRCRD&vgnextchannel=864f79ed268fe410VgnVCM1000000b205a0aRCRD

[31] Puerta de enlace Laird. Sentrius RG186. URL: https://www.lairdconnect.com/wireless-modules/lorawan-solutions/sentrius-rg1xx-lorawan-gateway-wi-fi-ethernet-optional-lte-us-only

[32] AWS Lamba. URL: https://aws.amazon.com/es/lambda/features/