Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería Electrónica de Comunicaciones

Trabajo Fin de Grado

Estudio del módulo Heltec Cubecell-GPS para aplicaciones de localización y monitorización remota de sensores

Autor: Alberto Garmendia Gutiérrez

Tutor: José Manuel Villadangos Carrizo

2021

UNIVERSIDAD DE ALCALÁ
Escuela Politécnica Superior

GRADO EN INGENIERÍA ELECTRÓNICA DE COMUNICACIONES

Trabajo Fin de Grado

Estudio del módulo *Heltec Cubecell-GPS* para aplicaciones de localización y monitorización remota de sensores

Autor: Alberto Garmendia Gutiérrez

Tutor: José Manuel Villadangos Carrizo

TRIBUNAL:

Presidente: Pedro Martín Sánchez

Vocal 1º: Carlos Julián Martín Arguedas

Vocal 2º: José Manuel Villadangos Carrizo

FECHA: 21 de Junio de 2021

ÍNDICE

Índice de Figuras

Índice de Tablas

1. Resumen

El objetivo del Trabajo Fin de Grado es el estudio del módulo *Heltec Cubecell-GPS* y la obtención de los valores de los diferentes sensores representados de forma gráfica en diferentes plataformas IoT (Internet de las Cosas) a través de TTN (*The Things Network*) Console.

En este proyecto se han utilizado diferentes sensores, tanto analógicos como digitales. Para determinar la calidad del aire interior se ha empleado el sensor CCS811. En cuanto a la obtención de la intensidad de la radiación ultravioleta (Índice UV) se ha empleado el sensor UVM30A y para establecer la medición de la temperatura y humedad se ha empleado el sensor DTH22.

Se utiliza un RTC (Reloj en Tiempo Real) y una tarjeta de memoria (micro SD) donde se podrá almacenar en un fichero, un log con los eventos asociados con las diferentes magnitudes de los sensores.

Respecto a las plataformas para representar los datos a tiempo real, se usarán ThingSpeak, Cayenne My Devices y ThingsBoard Cloud.

2. Abstract

The objective of the Final Degree Project is to study the Heltec Cubecell-GPS module and obtain the values of the different sensors represented graphically on different IoT (Internet of Things) platforms through TTN (The Things Network) Console.

Different sensors, both analog and digital, have been used in this project.

The CCS811 sensor was used to determine the indoor air quality. As for obtaining the intensity of ultraviolet radiation (UV Index), the UVM30A sensor has been used and the DTH22 sensor has been used to establish the measurement of temperature and humidity.

An RTC (Real Time Clock) and a memory card (micro SD) are used where a record with the events associated with the different magnitudes of the sensors can be stored in a file.

Regarding the platforms to represent the data in real time, ThingSpeak, Cayenne My Devices and ThingsBoard Cloud will be used.

3. Resumen extendido

Debido al gran avance de las tecnologías actuales, la conectividad entre dispositivos se encuentra en un momento álgido. La demanda de una red de bajo consumo de energía y alta cobertura ha llevado al desarrollo de nuevas plataformas como el Internet de las cosas (IoT), propuesto por primera vez por Kevin Ahston en 1999.

IoT permite conectar cualquier dispositivo a Internet sin necesidad de intervención humana, este fenómeno es conocido como una interacción *machine-to-machine* (M2M). Existen multitud de fabricantes de dispositivos IoT que han desarrollado sus propios protocolos de comunicación. El protocolo más utilizado es el denominado *MQTT* (*Message Queuing Telemetry Transport*).

Estas tecnologías *IoT* permiten conectar cualquier tipo de sensores a la red y desarrollar aplicaciones que controlen y administren los mismos. Se deben cumplir los siguientes requisitos:

- Recoger los datos enviados por cada dispositivo.
- Almacenar y analizar la información.
- Representar la información de forma gráfica.

El proyecto consiste en el estudio del módulo *Heltec Cubecell-GPS*. Para la transmisión de los datos se utiliza *The Things Indoor Gateway*, que es una puerta de enlace que permite la conexión de dispositivos compatibles con el protocolo *LoRaWan*. Dicho protocolo se caracteriza por:

- Bajo consumo de energía.
- Conexiones encriptadas de extremo a extremo.
- Largo alcance de comunicación.
- Baja transferencia de datos.
- Topología estrella.
- 3 tipos de frecuencias de trabajo: 868 Mhz(Europa), 915 Mhz(EEUU) y 433 Mhz(Asia).

Los elementos y las magnitudes a medir son: el monóxido de carbono (CO), los compuestos volátiles (VOC), el índice de intensidad ultravioleta (UV), la latitud, la longitud, la altura, la temperatura, la humedad relativa y la presión atmosférica absoluta.

Para la obtención de las magnitudes de la latitud, la longitud y la altura se emplea el módulo Air530Z. Este módulo utiliza el posicionamiento por satélite y tiene un bajo consumo de energía.

En la actualidad existen diversas plataformas *IoT*, compatibles con *The Things Network*, que permiten visualizar y analizar los datos en tiempo real. Estas plataformas reducen los costes del desarrollo y facilitan la innovación en los modelos de negocio.

En este proyecto se llevará a cabo el estudio de las siguientes plataformas: *ThingSpeak*, *ThingsBoard* y *Cayenne Mydevices*.

El siguiente diagrama de bloques representa el proyecto a realizar:

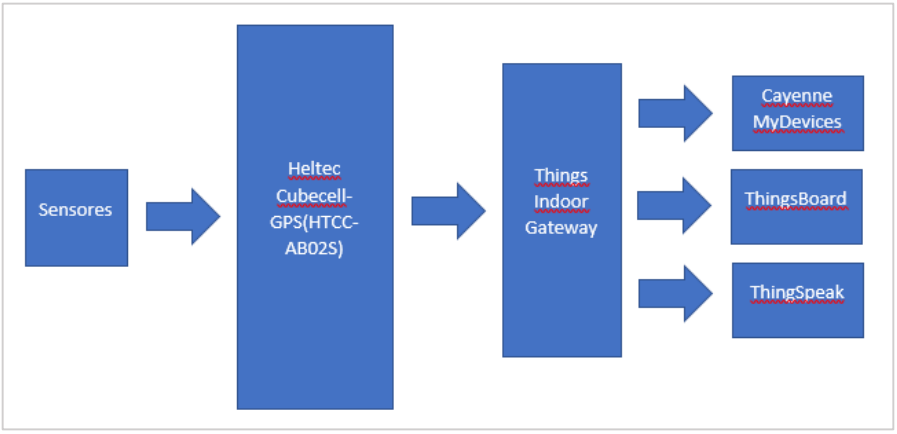

Figura 1. Diagrama de bloques del proyecto

4. Palabras clave

Palabras clave: IoT, TTN, ABP, *ThingSpeak*, *Cayenne myDevices* y *ThingsBoard*.

5. Memoria

5.1 Introducción

En los últimos años se están utilizando mucho las redes LPWAN (*Low-Power Wide Area Network*) o redes de bajo consumo y área extensa, siendo una alternativa para soportar millones de dispositivos.

Las redes de comunicaciones inalámbricas han evolucionado hacia una nueva industria denominada Internet de las cosas(*IoT*). El Internet de las cosas es una red de interconexión digital entre dispositivos a través del internet que permite el intercambio de datos entre ellos, permitiendo capturar información sobre el uso, el rendimiento de los dispositivos y mejores experiencias para los usuarios.

Constituye un cambio radical en la calidad de la vida de las personas ofreciendo una gran cantidad de nuevas oportunidades de accesos a datos.

El concepto de internet de las cosas fue propuesto en 1999 por Kevin Ashton, en el Auto-ID Center del MIT, en donde se realizaban investigaciones en el campo de la identificación por radiofrecuencia en red (RFID) y tecnologías de sensores.

Los dispositivos IoT se conectan con el proceso llamado M2M (machine to machine) en el que los dispositivos se comunican entre sí utilizando cualquier tipo de conectividad (Wifi, Bluetooth) sin necesidad de contacto humano.

Estos dispositivos conectados, generan una gran cantidad de datos que llegan a una plataforma IoT que procesa y analiza dichos datos, a través de dichos datos se pueden sacar conclusiones de los hábitos y preferencias del mismo.

Hoy en día, *IoT* se ha convertido en una de las tecnologías más importantes. Entre las tecnologías más utilizadas para el *IoT* están: *LoRa*, *SigFox*, *ZigBee*.

La tecnología *LoRa* fue desarrollada en 2012 por la startup de Grenoble (Francia) y posteriormente adquirida por Semtech. Es utilizada para la conexión de largo alcance, bajo coste y bajo consumo de energía. Funciona como una tecnología modulada para reducir costos y el consumo de energía de los dispositivos que se conectan.

Utilizando la tecnología LoRa, se pretende realizar la implementación de un prototipo de red de sensores IoT con el fin de monitorizar en tiempo real a través de TTN (*The Things Network*) Console), parámetros ambientales como: el monóxido de carbono (CO), los compuestos volátiles (VOC), el índice de intensidad ultravioleta (UV), la latitud, la longitud, la altura, la temperatura y la humedad relativa.

Para el desarrollo de este proyecto se empleará el módulo *Heltec Cubecell-GPS* totalmente compatible con el entorno de desarrollo integrado por Arduino IDE 1.8.13, la *Gateway LoraWan* (*Things Indoor Gateway*) para la transferencia de datos, y a la vez se mostrará a través de TTN (*The Things Network*) Console.

5.2 Redes LPWAN

La red de área amplia de baja potencia (LPWAN-red de área amplia de baja potencia) es un tipo de sistema de red de área amplia de telecomunicaciones inalámbricas diseñado para conectar objetos que permiten una comunicación remota de baja tasa de bits. El bajo consumo de energía, la baja tasa de bits y el uso previsto distinguen este tipo de red de las redes inalámbricas de área amplia, porque las redes de área amplia están diseñadas para conectar usuarios o empresas y utilizar más energía para transportar más datos.

Características de LPWAN:

- Alcance geográfico: diseñado para el transporte inalámbrico de datos entre dispositivos separados por distancias kilométricas.
- Velocidad de datos: varía entre 0,3 kbit /s a 50 kbit/s por canal.
- Bajo consumo eléctrico: las baterías de los dispositivos permiten una duración de 5 años.
- Cantidad de datos transmitidos: el objetivo de las redes LPWAN es regular el transporte de datos entre dispositivos.
- Es un protocolo de transporte inalámbrico de datos que hoy en día forma parte de la implementación de *IoT*.

La red LPWAN puede funcionar en la banda ISM libre sin una licencia.

Actualmente, existen tres grandes tecnologías LPWAN: *Sigfox*. *LoRa/LoRaWan* y *NB-IoT*.

5.2.1 LoRa

LoRa fue patentada por Semtech y utiliza un tipo de modulación en radiofrecuencia, esta tecnología de modulación se llama CSS (*Chirp Spread Spectrum*) permite lograr comunicaciones a kilómetros de distancia y tiene gran solidez frente a las interferencias.

Sus principales ventajas son:

- Bajo consumo (duración de las baterías hasta 10 años).
- Largo alcance de 10 a 20 km.
- Topología de estrella.
- Conexión punto a punto.
- Frecuencias de trabajo: 868 Mhz en Europa, 915 Mhz en América y 433 Mhz en Asia.
- Alta tolerancia a las interferencias.
- Sensibilidad de -168 Dbm.

Banda ISM

Lora utiliza el espectro sin licencia como parte de la banda de radio ISM (Industrial, Científica y Medica)

Se utiliza un plan de frecuencias sin licencia en Europa en torno a los 868 Mhz, mientras que en Asia 433 Mhz y en Estados Unidos es 915 Mhz.

Los parámetros de comunicación son:

- Canal de la banda: frecuencia central que representa el canal.
- Spreading Factor (SF): proporciona el número de bits usados para codificar un símbolo. LoRa utiliza un factor de propagación entre 7 y 12. Cuanto mayor sea SF, menor velocidad de transferencia de datos.
- Coding Rate (CR): nos permite añadir una corrección de errores hacia adelante (FEC) en cada transmisión de datos. Si existen muchas interferencias en el canal, se aconseja aumentar el valor de CR.
- Bandwith(BW): indica el ancho de frecuencia que se va a usar. *LoRa* puede utilizar canales con un ancho de banda de 125 kHz o 250 kHz, según la región o el plan de frecuencias.

LoRa es una tecnología ideal para conexiones de larga distancia y redes *IoT* en las que se necesiten sensores que no dispongan de corriente eléctrica y tiene una amplia gama de aplicaciones:

- Ciudades inteligentes.
- Lugares con poca cobertura (explotaciones agrícolas o ganaderas).
- Para construir redes privadas de sensores y/o actuadores.

Figura 2. LoRa [1]

5.2.2 SigFox

Es una compañía francesa fundada en 2009 que funciona con la tecnología de transmisión *Ultra narro band* (UNB) y consiste en emplear canales estrechos del espectro para alcanzar grandes distancias utilizando la mínima energía.

Es un servicio de datos que ha cambiado por completo el mundo de la comunicación M2M (*Machine-to-Machine*) diseñado para Internet de las cosas (*IoT*) y apuesta por la comunicación de baja velocidad con poca transmisión de datos.

Sigfox utiliza codificación diferencial de fase binaria o PSK y codificación de cambio de frecuencia gaussiana para permitir la comunicación mediante la red de radio ISM (bandas de radiofrecuencia industrial, científica y médica). La frecuencia de comunicación en Europa es de 868 MHz y la frecuencia de comunicación en los Estados Unidos es de 902 MHz. Utiliza una señal potente que puede pasar fácilmente a través de objetos sólidos denominada "banda ultra estrecha" y requiere muy poca energía, por lo que se denomina "red de área amplia de baja potencia (LPWAN)". La red se basa en una red en estrella y requiere que los operadores móviles gestionen el tráfico generado. Esta señal también se puede utilizar fácilmente para cubrir un área más grande y alcanzar objetos subterráneos.

Las principales ventajas que proporciona esta tecnología son:

- Bajo coste: tecnología fácil de integrar y protocolo de acceso gratuito.
- Bajo consumo: comunicaciones optimizadas que reducen el consumo de los dispositivos conectados.
- Dispositivos conectados: amplia conexión a la red.
- Regiones: 60 países están cubiertos con la red.
- La banda pública para el intercambio de mensajes en *SigFox* es en 200 Khz y en cada mensaje tiene un ancho de 100 Hz y su tasa de transmisión varía desde 100 bits a 600 bits dependiendo de la localización.

La cobertura de red en España está gestionada por Cellnex, la misma empresa que gestiona la infraestructura de red GSM / 3G / 4G.

Por último, el uso de la tecnología *SigFox* se puede utilizar para la monitorización de cualquier tipo de variable:

- Eficiencia energética.
- Variables de confort internas de los edificios.
- Sector Agrícola.
- Detección de presencia.
- Detección de inundaciones.

Figura 3. SigFox [2]

5.2.3 NB-IoT

Narrowband-IoT forma parte de las redes LPWAN (*Low-power wide-area networking*) y es la primera tecnología encargada de conectar los objetos cotidianos que necesitan pequeñas cantidades de datos en periodos de tiempos largos.

Su propósito es proporcionar una comunicación eficaz y una batería de larga duración para dispositivos distribuidos a gran escala. Utiliza la red móvil existente para conectar todos estos objetos. *NB-IoT* tiene como objetivo expandir el futuro de la conectividad de *IoT* de una manera más segura y confiable. Es una opción ideal para dispositivos que generan un tráfico de datos muy elevado y tienen un ciclo de vida prolongado.

Es una tecnología que utiliza las bandas celulares de comunicaciones. Permite tener una cobertura muy buena en entornos urbanos y tener mejores tiempos de respuesta que garantizan una mejor calidad de servicio.

Como principal ventaja se puede mencionar que los dispositivos *NB-IoT* hacen uso de la cobertura 4G. En áreas urbanas densas existen muchos repetidores, lo que hace que esta tecnología sea más rápida que la tecnología *LoRa*.

En cuanto a las desventajas, se pueden citar las siguientes:

- La dificultad a la hora de implementar el firmware por aire (*FOTA*) o la transferencia de archivos.
- Algunas de las especificaciones de diseño de *NB-IoT* no permiten enviar grandes cantidades de datos a un dispositivo.
- *NB-IoT* funciona mejor para sensores y medidores en una ubicación fija.

Por último, éstos son algunos de los mercados potenciales para los servicios *NB-IoT*:

- Ciudades Inteligentes.
- Casas y Edificios inteligentes: gestionar iluminación, climatización inteligentes y calidad del aire.
- Agricultura y ganadería: las granjas del futuro con el objetivo de conseguir una granja autónoma y conectada, a través de los sensores atmosféricos, seguimiento del ganado (con alertas de movimiento) y monitorización de la lluvia.
- Fabricación y Logística: se podrá localizar y hacer un inventario de manera fácil para optimizar la logística.

Figura 4. NB-IoT [3]

5.3 Protocolo LPWAN

Es un protocolo de capa de control de acceso a medios (*MAC*) que cumple con los requisitos *IoT*, es decir, la seguridad *end-to-end* y la comunicación bidireccional.

El protocolo *LoRaWan* es un estándar abierto, diseñado para utilizar la topología estrella donde cada nodo final se comunica con varias puertas de enlace que a su vez se comunican con el servidor de red.

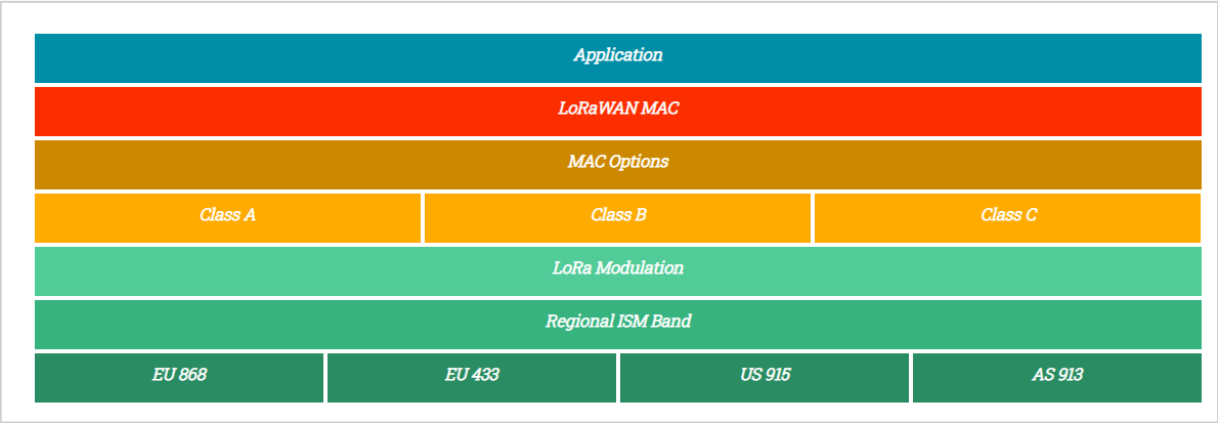

Figura 5. Clases de dispositivos LoRaWan [4]

5.3.1 Clases LoRaWaN

El protocolo define tres tipos de clases: clase A, B y C, que permiten la comunicación bidireccional y pueden iniciar una subida de datos a los servidores a través de una *Gateway*. Estas clases son otro ejemplo del diseño de *LoRaWan* ya que permiten indicar el consumo energético de un dispositivo.

Clase A

Es la clase que permite ahorrar la mayor energía. Después de enviar los datos se abren dos ventanas: la primera se abre un segundo después de la transmisión y la segunda está esperando a enviar las respuestas.

Clase B

Los dispositivos de clase B se sincronizan con señales periódicas, creando ventanas de escuchas para que los dispositivos puedan recibir señales o comandos. Al estar siempre activos consumen una mayor energía.

Clase C

Estos dispositivos tienen mejor latencia y menor ahorro de energía. Siempre están disponibles para la recepción de mensajes y sólo se interrumpen cuando se produce una transmisión.

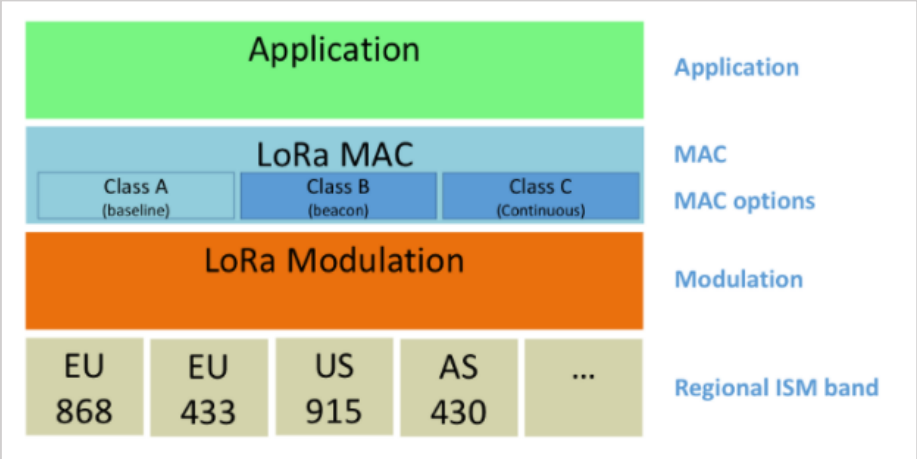

Figura 6. Clases LPWAN [5]

5.3.2 Arquitectura de Red

La arquitectura de Red *LoRaWan* utiliza una topología estrella-estrella para enviar los datos a un servidor central a través de *Gateways*.

Está formada por 4 elementos de red:

- Los nodos finales, que son los encargados de recoger los datos de los sensores, que son transmitidos a través de *upstream* y *downstream*.
- El concentrador/compuerta, que conecta los nodos finales con el servidor de red recibiendo y enviando datos.
- El servidor de red, que se conecta a varias puertas de enlace y sus funciones son eliminar mensajes duplicados, decidir que compuerta debe responder, gestionar las velocidades de trasmisión de datos para maximizar la capacidad de la red y aumentar la vida útil de la batería de los nodos finales.
- El servidor Web (servidor de la aplicación), encargado de recopilar y analizar los nodos finales y determinar las acciones del nodo final.

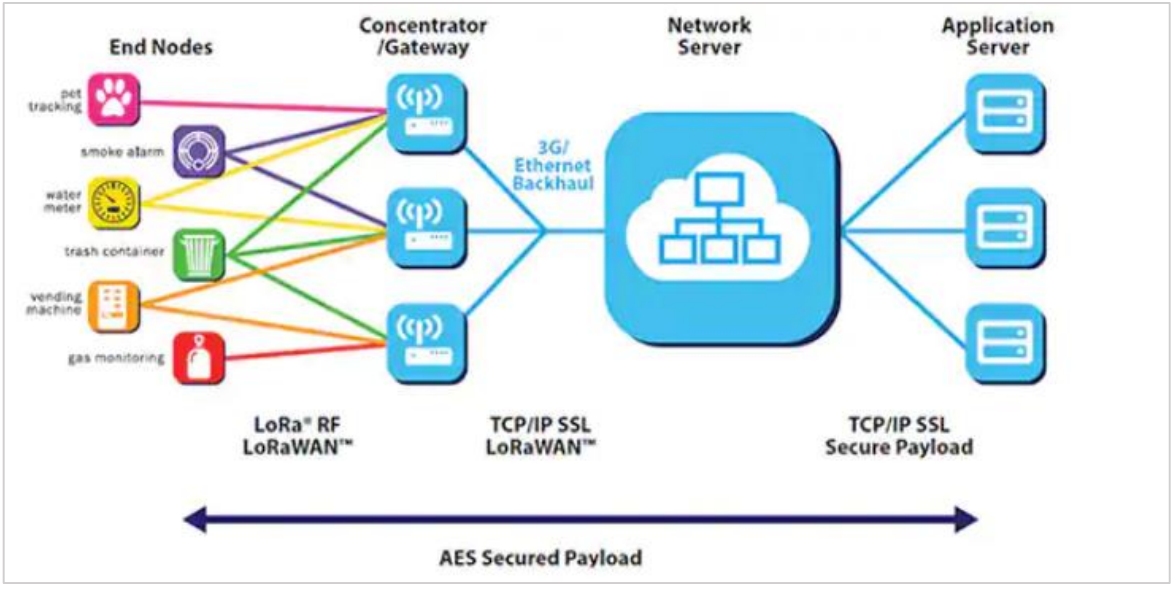

Figura 7. Red LoRa [6]

5.3.3 Seguridad del protocolo LoRaWan

La seguridad del protocolo *LoRaWan* se basa fundamentalmente en *IEEE 802.15.4*, que es un estándar que define el nivel físico y el control de acceso al medio de redes inalámbricas de área personal con tasas de transmisión bajas.

LoRaWan utiliza cifrado *AES* de 128 bits que es un estándar de encriptación de bloques basado en estas dos capas de seguridad: la clave de sesión de red(*NwKSKey*) y la clave de sesión de aplicación(*AppsKey*).

Hay dos métodos para implementas las claves:

1) **El método activación por aire (*OTAA*):** es uno de los métodos más seguro de conectarse a la red. Sus parámetros son:
   - *AppEUI*: es un ID de la aplicación global en el espacio de direcciones *IEEE EUI64*, que identifica el servidor de unión durante la activación por aire.
   - *AppKey*: es una clave secreta utilizada para determinar la clave de sesión.
   - *DevEui*: es un identificador único para cada dispositivo.

   Gracias a estos parámetros, la conexión se realiza de la siguiente forma:

   - El nodo solicita utilizar los datos de configuración para unirse (o iniciar sesión) a la red y abrir la ventana de recepción.
   - La pasarela recibe la solicitud y la envía al servidor.
   - El servidor verifica que el nodo esté registrado y la clave de cifrado sea correcta.
   - Si es correcto, asigna una sesión temporal y envía al nodo a través de la pasarela.
   - El nodo recibe la sesión temporal y permite enviar datos a la red.

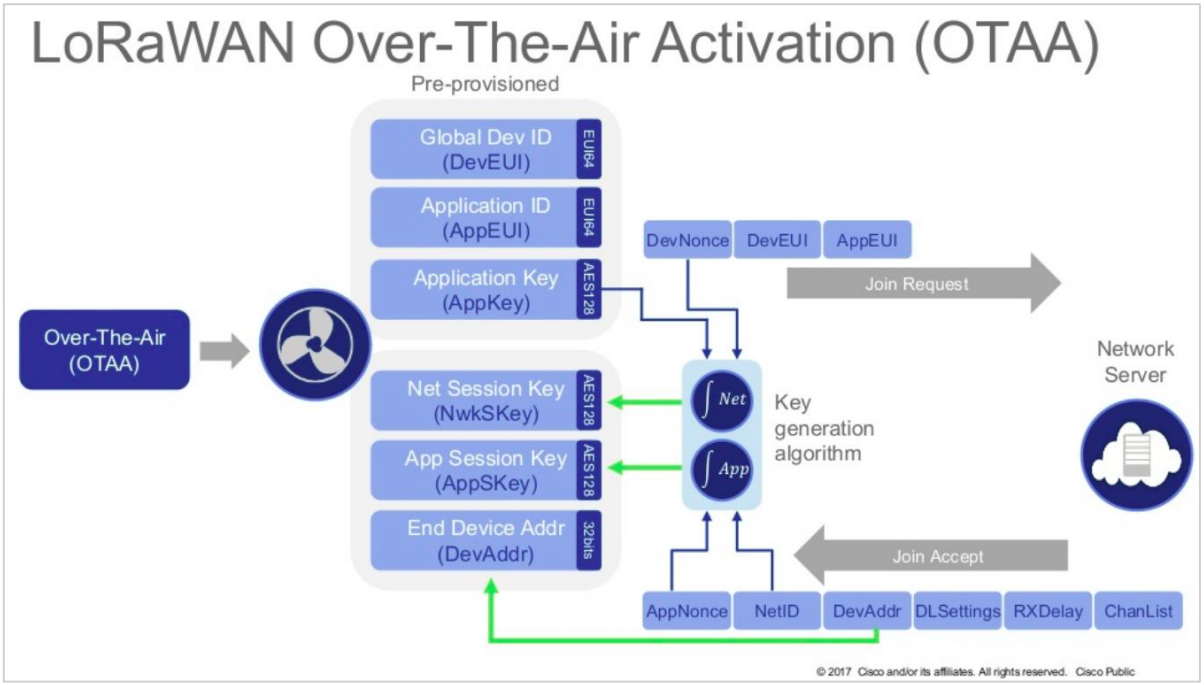

Figura 8. LoRaWan Over-The-Air Activation (OTAA) [7]

2) **Método *Activation-By-Personalisation* (*ABP*),** sus parámetros son:
   - *DevAddr*: es la dirección del nodo, que identifica el dispositivo final dentro de la red actual.
   - Network Session Key(*NwKSKey*): es la clave de sesión de red que interactúa entre el nodo y el servidor de red, validando la integridad de cada mensaje a través del código de integridad del mensaje (verificación *MIC*).
   - *AppSKey*: es una clave de sesión de aplicación específica para el dispositivo final. Se utiliza para cifrar y descifrar el campo de carga útil de los mensajes de datos de la aplicación. La carga útil se encuentra encriptada entre el nodo y el servidor de aplicaciones de *Things Network.*

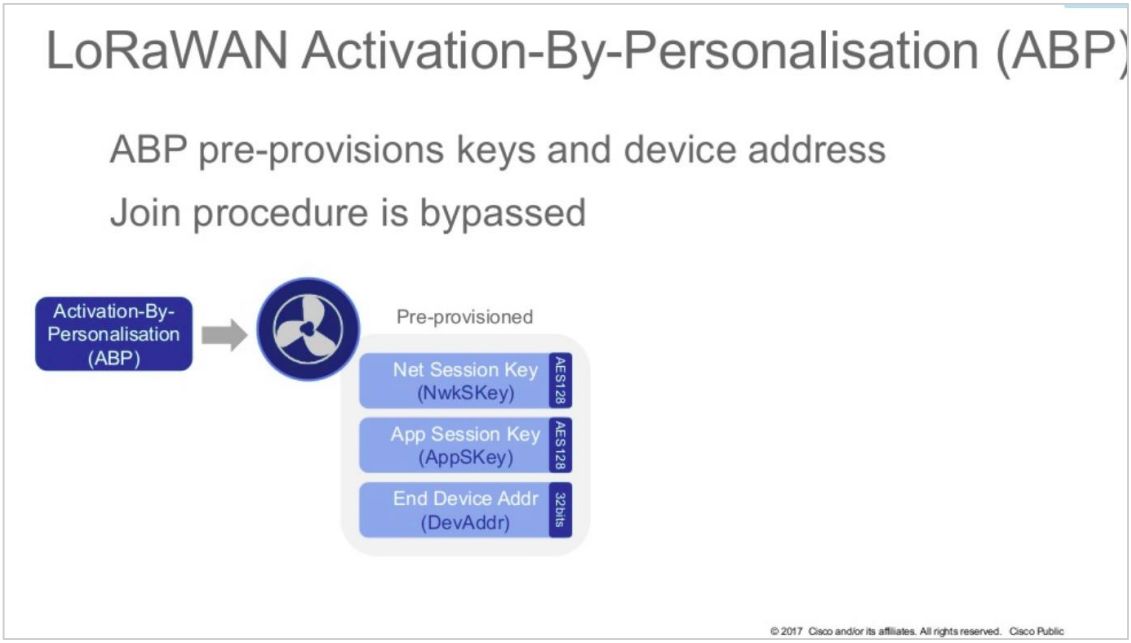

Figura 9. Activation-By-Personalisation (ABP) [8]

Con estos parámetros, la conexión se realiza de la siguiente manera:

- El dispositivo envía datos a la puerta de enlace.
- Los datos corresponden a la pasarela de la sesión.
- Si la sesión es correcta, entonces se procesarán los datos, en caso contrario los datos serán rechazados.
- La principal ventaja de este tipo de conexión es que no es necesario unirse a la red para enviar datos y no se requiere confirmación del lado del servidor porque la sesión se ha asignado manualmente (independientemente de si está activa o no). Con excelentes capacidades de recepción, este método de conexión es ideal. La desventaja es que después de encontrar la clave de cifrado en el dispositivo, un atacante puede extraer y clonar la clave.

5.4 Heltec Cubecell-GPS

CubeCell (TM) es una nueva serie de productos fabricada por el equipo de *Heltec*, principalmente para aplicaciones de nodo *LoRa / LoRaWAN*.

La serie *CubeCell* (TM) se basa en ASR605x (ASR6501, ASR6502), esos chips ya están integrados con el MCU de la serie PSoC ® 4000 (ARM ® Cortex ® M0 + Core) y SX1262. Es compatible con Arduino , puede ejecutar el protocolo *LoRaWAN* de forma estable, se puede conectar fácilmente las baterías de litio y paneles solares.

Cubecell-GPS es un módulo de desarrollo cuyas características son:

- Basado en ASR6052, estos chips están ya integrados con el MCU de la serie PSoC® 4000 (ARM ® Cortex ® M0 + Core) y SX1262.
- Diferentes bandas de trabajo LoRa: Europa(868MHz), Estados Unidos
- Soporte *LoRaWan* 1.0.2.
- Diseño de energía ultra baja, 50uA en sueño profundo.
- Sistema de gestión de energía solar a bordo, se puede conectar directamente con un panel solar de 5,5 ~ 7V.
- Interfaz de batería SH1.25-2 a bordo, sistema de administración de batería de litio integrado (administración de carga y descarga, protección de sobrecarga, detección de energía de la batería, conmutación automática de energía de batería/USB).
- Interfaz micro USB con protección ESD completa, protección contra cortocircuitos, blindaje RF y otras medidas de protección.
- Larga distancia de comunicación.
- Utiliza el modulo *GPS Air530Z*, soporte del sistema de posición GPS / Beidou de modo dual.
- Pantalla OLED de matriz de puntos 0,96 pulgadas 128 * 64 integrada, que se puede utilizar para mostrar información de depuración, energía de la batería y otra información.

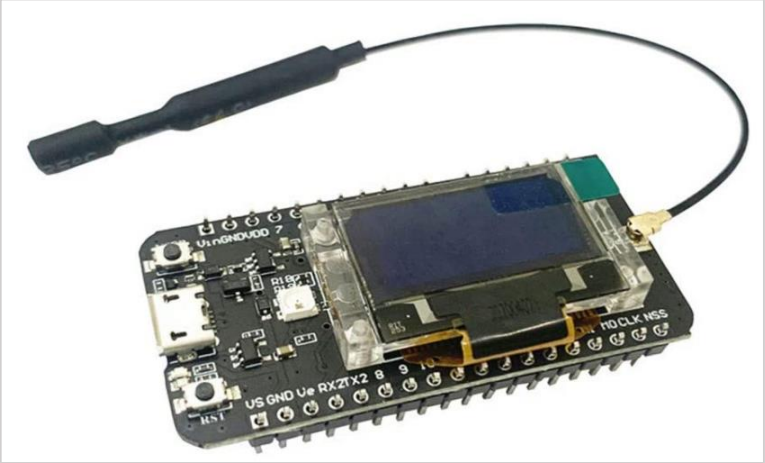

Figura 10. Heltec Cubecell-GPS [9]

5.4.1 Diagrama de pines

Los recursos de hardware:

- 2 UART's.
- 1 SPI
- 2 I2C's
- 1 SWD (Standing Wave Ratio).
- 1 ADC
- 12 Bits ADC, 8 canales motor DMA.
- 14 GPIO.
- Flash: 128 KB flash interno.

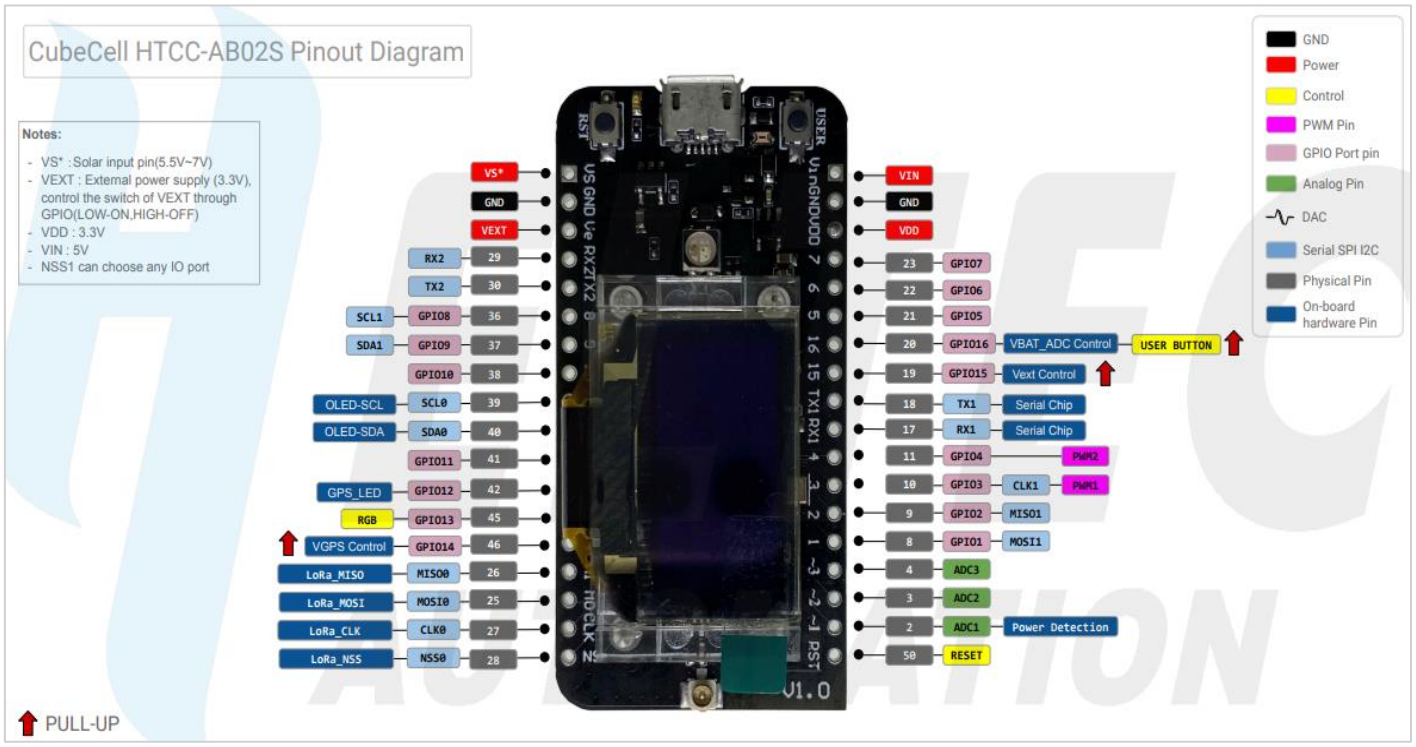

Figura 11. Diagrama de Pines Cubecell-GPS[10]

5.4.2 Módulo GPS AIR530Z

Es un módulo de navegación por satélite *GNSS/ BDS* de alto rendimiento que se pueden encontrar en drones, relojes inteligentes, rastreo de objetos, sistema de navegación de vehículos. Soporta *GPS/Beidou/Glonass/Galileo* permitiendo un posicionamiento *GNSS* (*Global Navigation Satellite System*). Es capaz de recibir información de más de 6 satélites al mismo tiempo. Tiene un bajo consumo de energía(<31μA).

Figura 12. Módulo GPS AIR530Z

Sus principales características son:

- Interfaz UART.
- Rango de Temperatura (-35ºC a 85ºC).
- Rango de Alimentación 3.3V/5V.
- Error de posición de unos 10 metros aproximados.
- Dispone de Real-Time Clock (RTC).
- Almacenamiento Flash.

Su diagrama de pines es el siguiente:

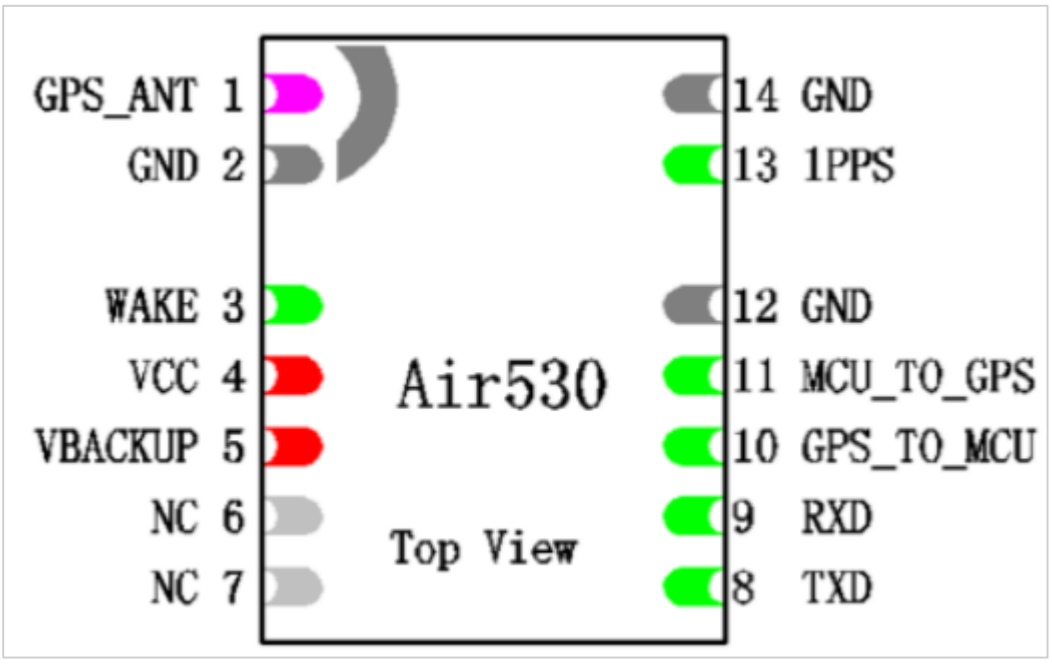

Figura 13. Diagrama de Pines del módulo GPS AIR530Z

Definición del Pin	Descripción del Pin
GPS_ANT	Entrada de la antena GPS
GND	Tierra
WAKE	Entrada(2.8V)
VCC	Alimentación de Entrada (2.8-4.2V)
VBACKUP	Entrada de memoria. Requiere alimentación continua
NC	Pin Reservado
NC	Pin Reservado
TXD	Datos enviados (2.8V). Baud Rate por defecto es 9600 bps
RXD	Datos recibidos (2.8V)
GPS_TO_MCU	Pin Reservado
MCU_TO_GPS	Pin Reservado
GND	Tierra
1PPS	1 Pulso por segundo
GND	Tierra

Tabla 1: Diagrama de pines Air530Z

Las funciones utilizadas para obtener la latitud, la longitud y la altura son:

- **Air530.begin()** -> Inicializar el módulo GPS.
- **Air530.location.lat()**-> Devuelve la magnitud de la latitud
- **Air530.location.lng()** -> Devuelve la magnitud de la longitud
- **Air530.altitude.meters()** -> Devuelve la altura en metros.

```
Serial.print("Latitud: ");
Serial.print(Air530.location.lat());
Serial.print(", Longitud: ");
Serial.print(Air530.location.lng();
Serial.print(", Altitud(m): ");
Serial.print(Air530.altitude.meters());
```

Figura 14. Función Módulo GPS Air530Z para obtener la longitud, la latitud y la altitud.

5.4.3 Configuración de la serie CUBECELL

A continuación, se explica cómo instalar las dos formas de la serie *CubeCell*:

1) El administrador de la placa Arduino

Abrir el IDE de Arduino y hacer clic en la pestaña Archivo-> Peferencias-> Configuración

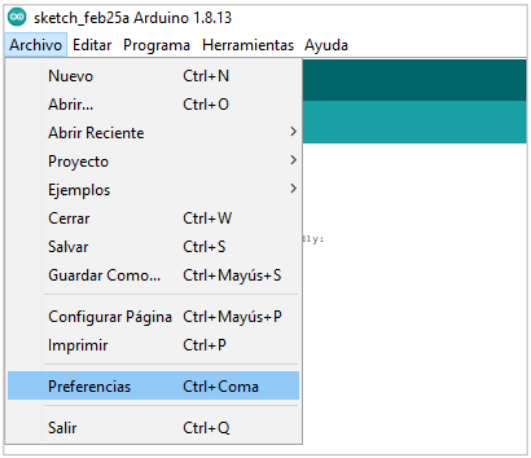

Figura 15. Preferencias IDE Arduino [11]

Al acceder a preferencias, permite ingresar direcciones URL para poder añadir un gestor de tarjetas adicionales.

En el campo de gestor URLs Adicionales de tarjetas, se añade esta dirección URL:

https://resource.heltec.cn/download/package_CubeCell_index.json

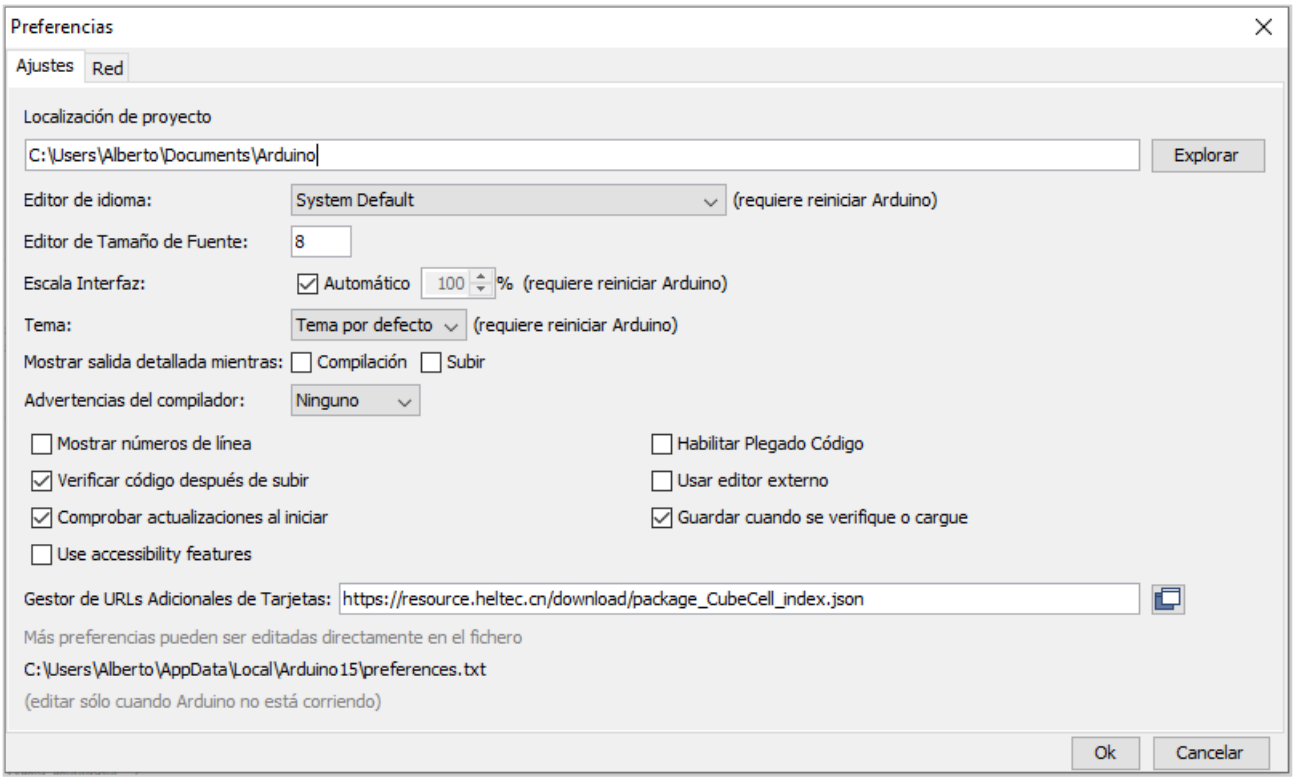

Figura 16. Gestión de URLs Adiccionales de Tarjetas [11]

Se accede a Herramientas-> Placa. En el nuevo cuadro de diálogo emergente, se selecciona Administrador de placas-> Heltec cubecell -> releases -> install.

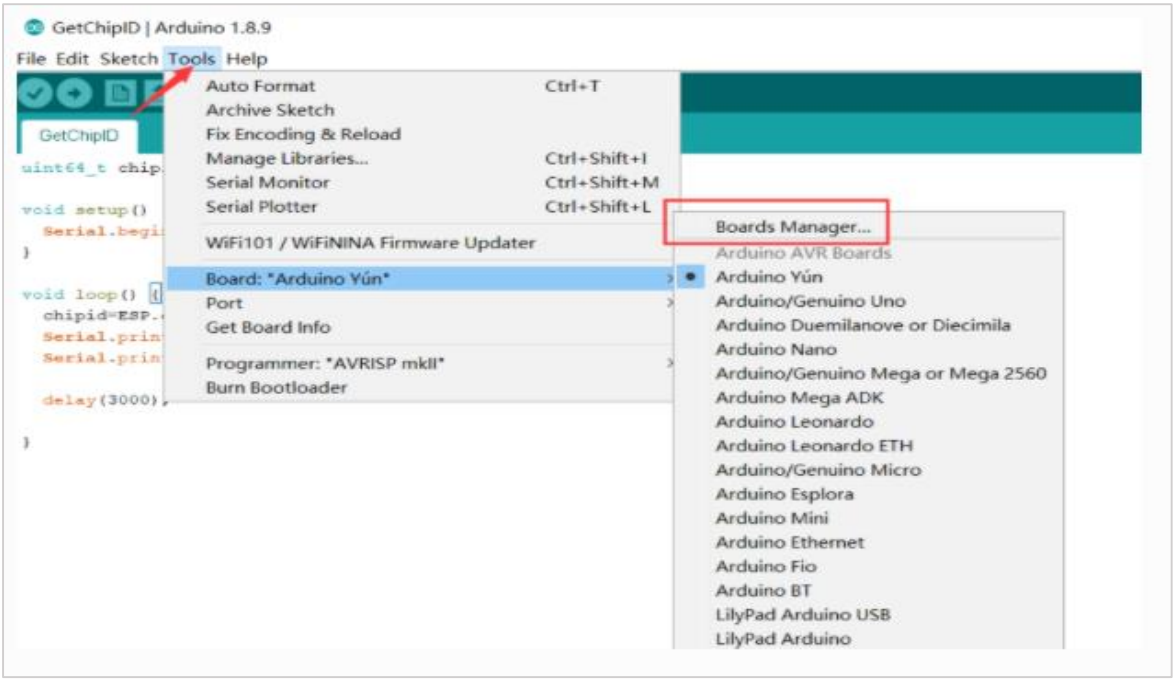

Figura 17. Gestor de Tarjetas[11]

Se instala la librería para poder seleccionar la serie *Cubecell*:

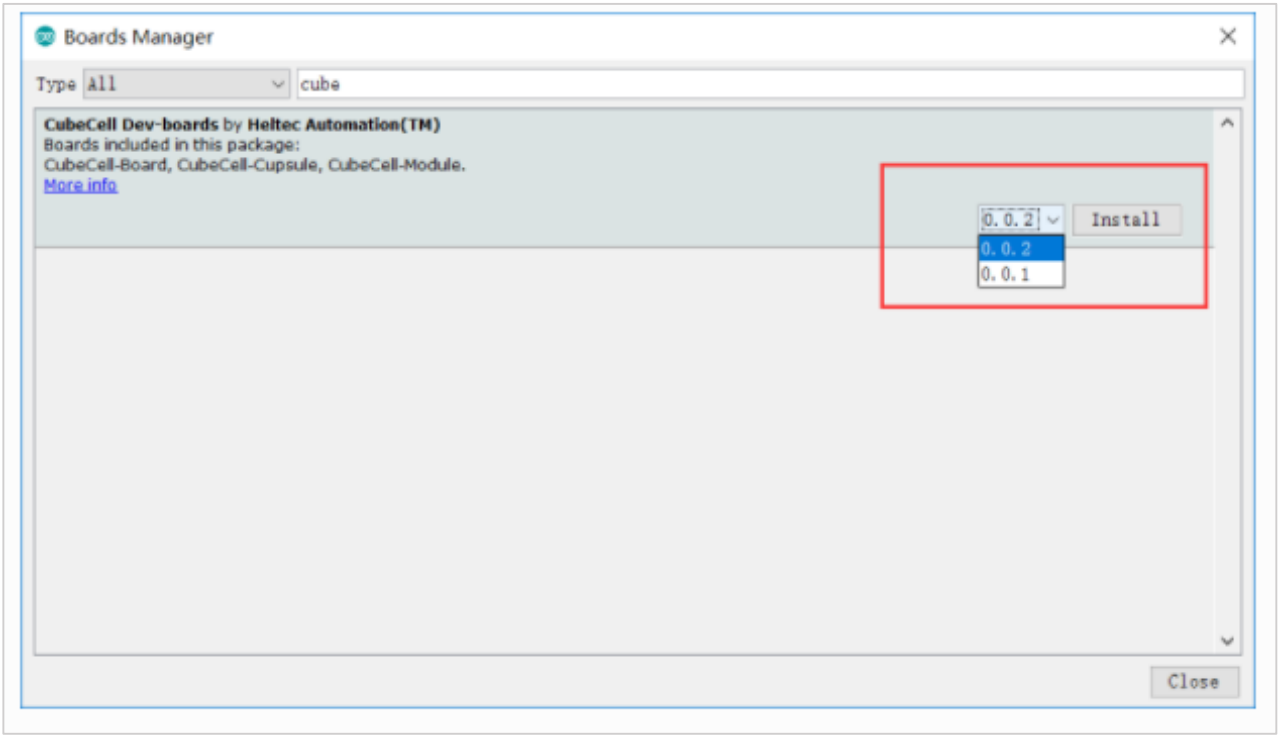

Figura 18. Instalación Librería Cubecell [11]

Por último, se selecciona nuestro módulo *Cubecell-GPS*:

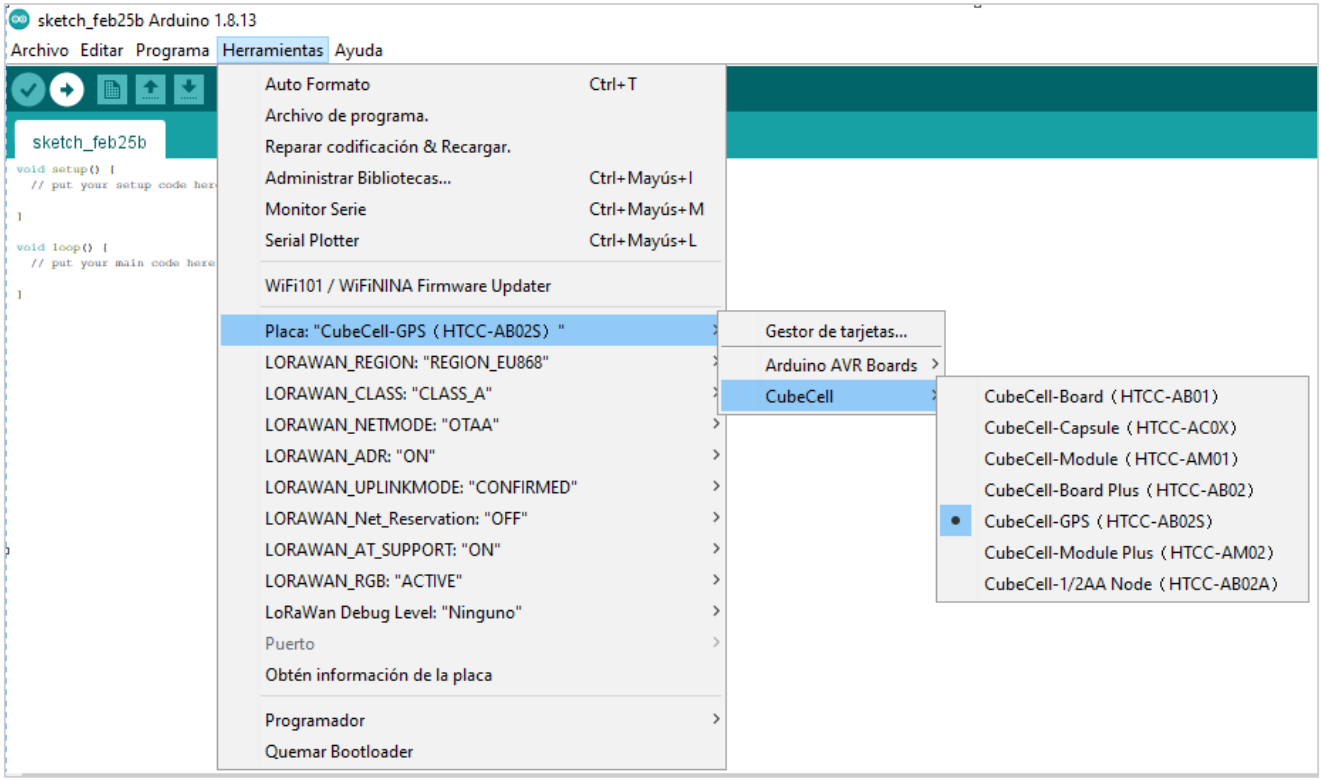

Figura 19. Selección Módulo Cubecell-GPS [11]

2) Vía Git

- **Windows**

Se ejecuta el siguiente comando https://github.com/HelTecAutomation/ASR650x-Arduino.git en la ruta \ Documents \ Arduino \ hardware \ CubeCell (debe estar en esta ruta). Se inicia Git Bash y se introduce git clone.

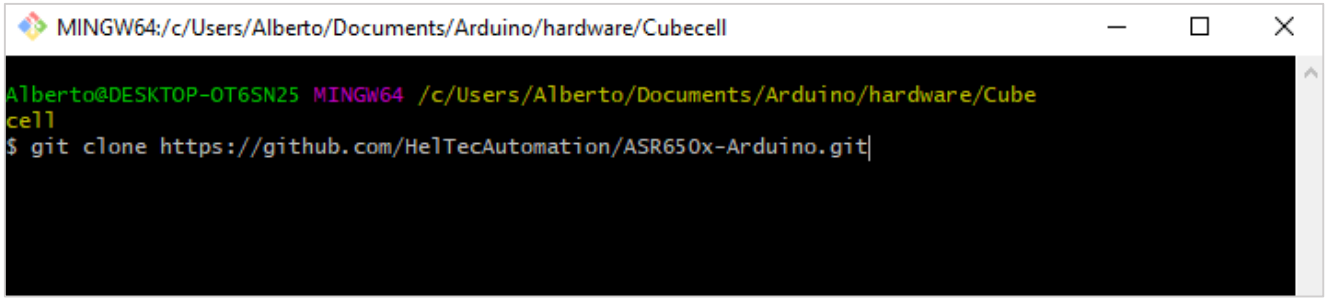

Figura 20. Ingresar ruta git clone [11]

Se abre la dirección /Documents/Arduino/hardware/CubeCell/ASR650x-Arduino/tools y se hace doble clic en el ejecutable “get.exe”.

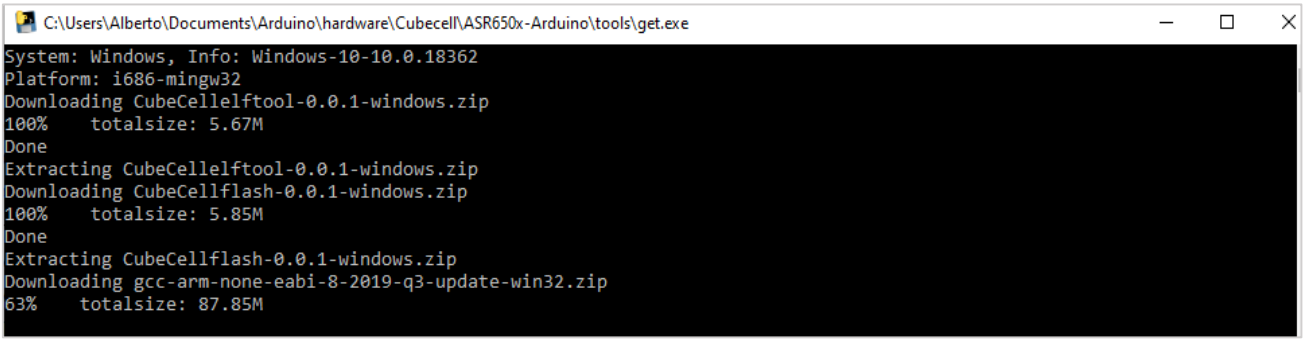

Figura 21. Instalación ASR650x-Arduino [11]

Cuando "get.exe" finalice la ejecución, se podrán observar los siguientes archivos en el directorio.

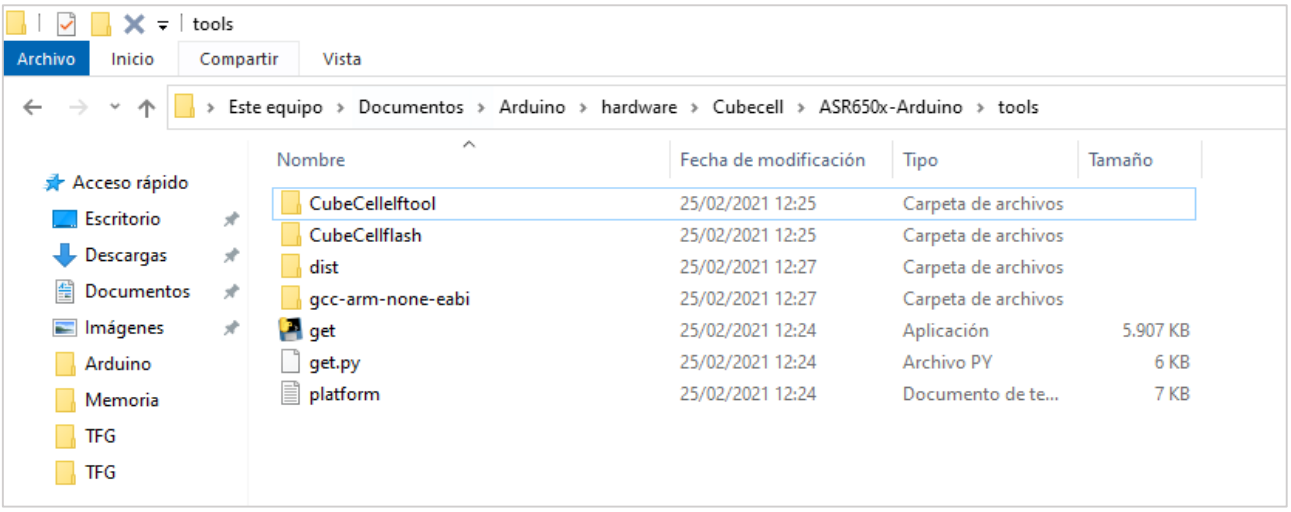

Figura 22. Directorio CubeCell [11]

Se conecta el módulo *CubeCell-GPS* a través de un cable Micro USB de alta calidad y se inicia Arduino IDE. A continuación, se selecciona la placa correspondiente en Herramientas > Placa.

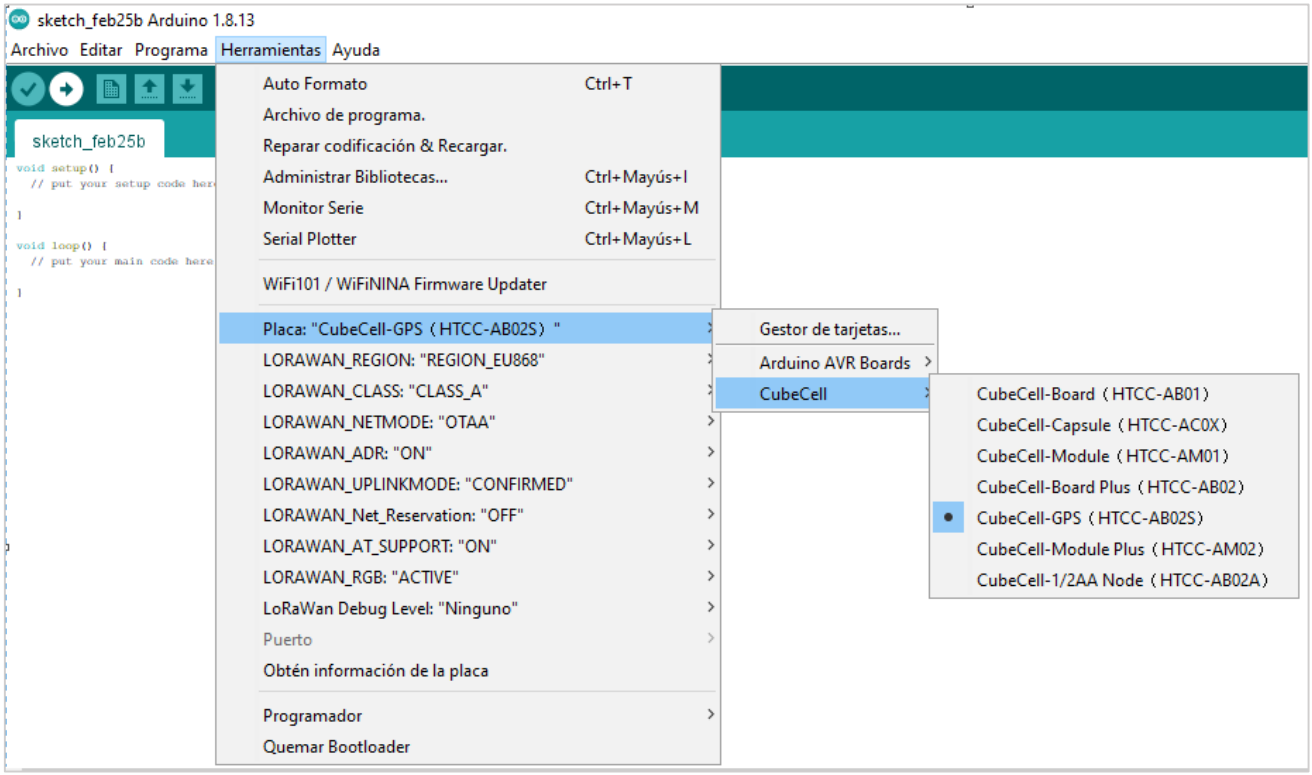

Figura 23. Selección Módulo Cubecell-GPS [11]

- **Mac OS**

Se procede a realizar la instalación del último IDE de Arduino desde su página web.

Abra Terminal y ejecute los siguientes comandos:

- **Crear un directorio**

  mkdir -p ~ / Documents / Arduino / hardware / CubeCell

- **Acceder al directorio creado**

  cd ~ / Documentos / Arduino / hardware / CubeCell

- **Ingresar en la consola-> git clone https://github.com/HelTecAutomation/ASR650x-Arduino.git**

- **Acceder a la carpeta herramientas**

  cd CubeCell / herramientas

- **Por último, instalar Python get.py**

5.5 Sensores Utilizados

5.5.1 Sensor de Calidad del Aire(CCS811)

CCS811 es un sensor utilizado por el fabricante de AMS para medir la calidad del aire interior. Es un sensor de MOX (óxido metálico) multigás, que incluye medición de compuestos y monóxido de carbono (CO). Se puede utilizar para determinar sustancias volátiles (COV) equivalentes al dióxido de carbono (eCO2), como el etanol o el hidrocarburo aromático. El rango de medición de eCO2 es de 400 a 8192 ppm y el rango de medición de TVOC es de 0 a 1187 ppb.

Figura 24. Módulo CCS811 [12]

Incluye un convertidor *ADC* y un procesador interno para cálculo y comunicación a través del bus *I2C*. El módulo puede operar a un voltaje de 1.8 V a 3.3 V. Además, el módulo de sensor tiene una salida de interrupción ajustable que puede informar los resultados de la medición o informar que se excede un umbral establecido previamente. El intervalo de medición se puede seleccionar entre los siguientes valores: 250 ms, 1 s, 10 s, 60 s. Se pueden obtener lecturas precisas a partir de 20 minutos.

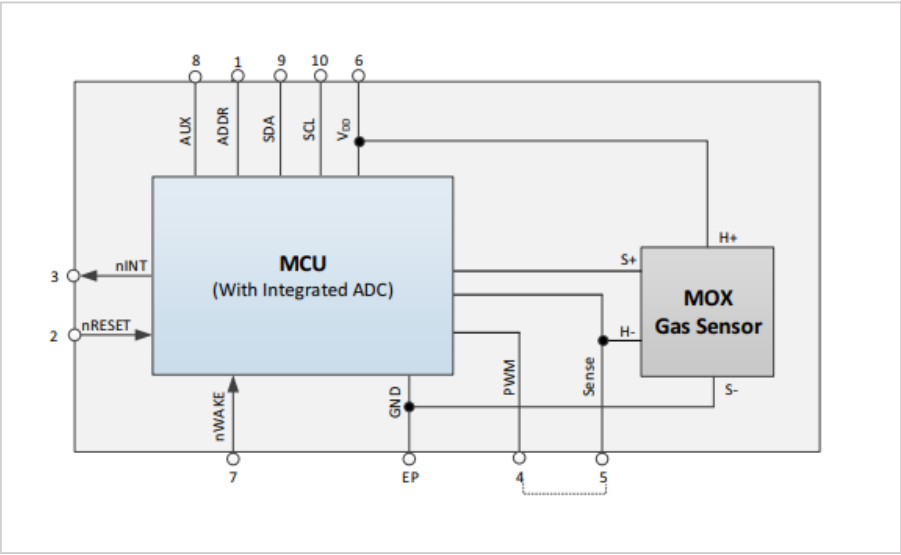

Figura 25. Diagrama de bloques del sensor CCS811 [13]

Las funciones principales utilizadas para obtener el valor de Co2 y Tvoc son:

- **geteCO2** -> Devuelve el valor de medición de dióxido de carbono estimado almacenado como un entero de 16 bits
- **getTvoc** -> Devuelve la medición de compuestos orgánicos volátiles totales almacenados como un entero de 16 bits.

```
while(i<5)
{
  ccs.available();
  {
    if(!ccs.readData())
    {
       co2 += ccs.geteCO2();
     tvoc += ccs.getTVOC();
      i++;
    }
  }
}
```

Figura 26. Leer 5 veces para obtener el valor promedio

Se obtiene 5 veces el valor de la medición de dióxido de carbono y los compuestos orgánicos volátiles totales para determinar el promedio.

5.5.2 Sensor de temperatura y humedad relativa (DHT22)

DHT22 (o AM2302) permite la medición simultánea de temperatura y humedad. Este sensor tiene un procesador interno que realiza el proceso de medición y proporciona los valores de medición a través de señales digitales, lo que facilita la obtención de los valores de medición del microprocesador.

El modelo DHT22 dispone de estas características:

- El rango de medición de temperatura es de -40 a 125 y la precisión es de 0,5ºC.
- El rango de medición de la humedad se encuentra entre 0 y 100%, con una precisión de 2-5%.
- Frecuencia de muestreo de 2 muestras por segundo (2 Hz).

Figura 27. Sensor de temperatura y humedad relativa (DHT22) [14]

Las funcionas principales utilizadas para obtener la temperatura y la humedad:

- **dht.readHumidity()** -> Devuelve el valor de medición de la humedad.
- **readTemperature()** -> Devuelve el valor de medición de la temperatura.

Solo podemos tomar 1 lectura cada 2 segundos, por eso hacemos una pausa de 2 segundos antes de tomar la lectura para asegurarnos de que hayan pasado 2 segundos desde la última lectura.

```
hum = dht.readHumidity();
temp= dht.readTemperature();

Serial.print("Humedad: ");
Serial.print(hum);
Serial.print(" %, Temperatura: ");
Serial.print(temp);
Serial.println(" Grados");
delay(2000);
```

Figura 28. Obtención de la medición de la temperatura y humedad relativa del sensor DHT22

5.5.3 Sensor de radiación ultravioleta (Índice UVM30A)

El índice ultravioleta es una indicación de la intensidad de los rayos ultravioleta del sol a la superficie de la tierra. Su escala comienza en 0 y no tiene límite superior. El índice UV también indica la capacidad de la radiación ultravioleta para dañar la piel. La cantidad de luz ambiental no siempre está relacionada con el índice UV. Dado que el índice y sus representantes varían de un lugar a otro, la Organización Mundial de la Salud, la Organización Meteorológica Mundial, el Programa de las Naciones Unidas para el Medio Ambiente y la Comisión Internacional para la Prevención de las Radiaciones Ionizantes han publicado conjuntamente un sistema estándar de métodos de medición del índice UV.

COLOR	INDICE UV	RIESGO
	<2	Bajo
	3-5	Moderado
	6-7	Alto
	8-10	Muy Alto
	>11	Extremadamente alto

Tabla 2. Índice UV

Los factores que afectan al índice UV son:

- Latitud: el sol tiene la posición más grande en la línea vertical, que ocurre entre los trópicos.
- Altitud: la intensidad de la radiación aumenta entre un 10% y un 12% por cada 1000 m de elevación.
- Nubosidad.
- La cantidad de ozono en la atmósfera.
- Índice de reflectancia del suelo: por lo general tiene poco efecto, pero la arena, la hierba, especialmente el agua (tan dura como la nieve, el líquido en el océano y los lagos) reflejarán mucha radiación ultravioleta.

El sensor UV se utiliza para detectar la intensidad de la radiación incidente, es decir, el rayo ultravioleta. Esta forma de radiación electromagnética tiene una longitud de onda más corta que la radiación visible. El sensor UVM-30A tiene una longitud de onda que varía entre 200 nm y 370 nm.

Figura 29. Módulo Sensor Ultravioleta UVM-30A [15]

Sus características principales son:

- Voltaje de funcionamiento: 3 ~ 5Vdc.
- Corriente: 0.06mA (Standard) /0.1mA (Max).
- Temperatura de trabajo: -20 ~ 85°C.

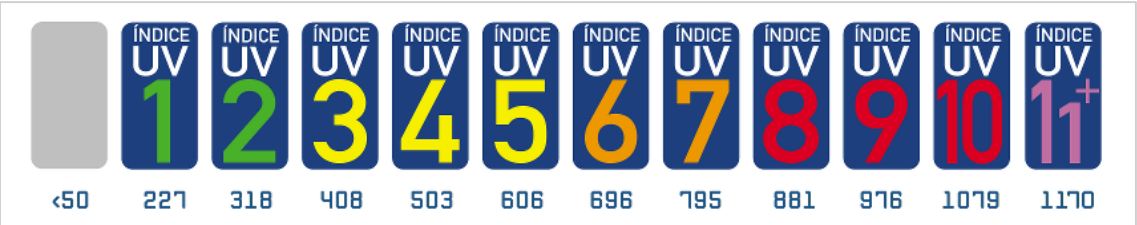

Figura 30. Índice UV respecto a los MV de salida [16]

La función utilizada para obtener el índice UV, midiendo la radiación ultravioleta es:

- **AnalogRead()** -> Lee el valor analógico existente en el pin indicado, en mi caso ADC3 y devuelve el voltaje en mV, el valor máximo que se puede leer es 2400Mv.
- **IF** -> utilizado para evaluar la figura 14.

```
  voltage=analogRead(ADC3);//return the voltage in mV, max value can be read is 2400mV


if(voltage<50)
  {
    UVIndex = 0;
  }else if (voltage>50 && voltage<=227)
  {
    UVIndex = 0;
  }else if (voltage>227 && voltage<=318)
  {
    UVIndex = 1;
  }
  else if (voltage>318 && voltage<=408)
  {
    UVIndex = 2;
  }else if (voltage>408 && voltage<=503)
  {
    UVIndex = 3;
  }
  else if (voltage>503 && voltage<=606)
  {
    UVIndex = 4;
```

Figura 31. Obtención Índice UV

Se realiza la medición de la variable ”voltage” y se compara con los valores mostrados en la figura 30 para obtener el valor del índice de radiación ultravioleta proveniente del sol (Índice UV).

5.5.4 Sensor de presión absoluta (LPS22HB)

El LPS22HB es un sensor piezorresistivo ultracompacto de presión absoluta que funciona como un barómetro de salida digital. El dispositivo comprende un elemento sensor, que se comunica a través del protocolo I2C o SPI.

El elemento sensor, que detecta la presión absoluta, garantiza que funcione en un rango de temperatura que va desde -40 ° C a +85 ° C. El paquete está perforado para permitir que la presión externa alcance el elemento sensor.

Sus principales características son:

- Rango de presión absoluta de 260 a 1260 hPa.
- Consumo de corriente hasta 3 µA.
- Compensación de temperatura integrada.
- Salida de datos de presión de 24 bits.
- Salida de datos de temperatura de 16 bits.
- Interfaces SPI e I2C.
- FIFO integrado.
- Funciones de interrupción: datos listos, banderas FIFO, umbrales de presión.
- Voltaje de suministro: 1,7 a 3,6 V.
- Alta capacidad de supervivencia al impacto: 22.000 g.

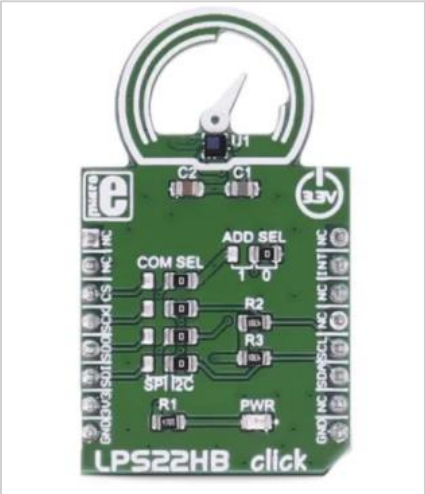

Figura 32. Módulo del sensor de presión atmosférica (LPS22HB) [17]

Las funciones principales usadas son:

- **Wire.begin()** -> Se inicia la comunicación I2C como MASTER.
- **Wire.avaliable()** ->Devuelve el número de bytes disponibles para la lectura.
- **Wire.read** -> Se encarga de leer los bytes que se transmitió de un dispositivo esclavo a un maestro.

```
if(Wire1.available() == 3)
{
  data[0] = Wire1.read();
  data[1] = Wire1.read();
  data[2] = Wire1.read();
}
delay(300);
```

Figura 33. Obtención de la presión atmosférica

Se obtienen los 3 bytes y se guardan en un array de enteros sin signo, para posteriormente convertirlo en la presión atmosférica (hPa).

5.5.5 Reloj en tiempo real RTC(DS3231)

Un reloj en tiempo real (*RTC*) es un dispositivo electrónico que nos permite obtener medidas del tiempo. El término *RTC* fue creado para distinguir este tipo de reloj del electrónico habitual, que mide el tiempo solo contando pulsos de señal y no tiene relación directa con la unidad de tiempo. Por el contrario, *RTC* es más similar al reloj y calendario que usamos habitualmente, y puede usar segundos, minutos, horas, días, semanas, meses y años. El *RTC* generalmente consta de un resonador de cristal, que se integra con la electrónica necesaria para contar el tiempo correctamente. Los equipos electrónicos de RTC tienen en cuenta las peculiaridades de la forma en que medimos el tiempo, como el sistema hexadecimal o años bisiestos.

DS3231 integra funciones de compensación de temperatura para garantizar una precisión de al menos 2 ppm, lo que equivale a un retraso máximo de 172 ms / día o 1 segundo cada 6 días. En el mundo real, suelen lograr una alta precisión, equivalente a un retraso de 1-2 segundos por mes. La comunicación de ambos tipos se realiza a través del bus I2C, por lo que es fácil obtener los datos de medición. El voltaje de la fuente de alimentación es de 2,3 a 5,5 V. La conexión es simple, sólo se necesita usar 5V y Gnd para alimentar el módulo desde el módulo de la batería de *Cubecell-GPS*. Por último, se conecta los pines del bus I2C al módulo de unidad de *Cubecell-GPS* correspondiente.

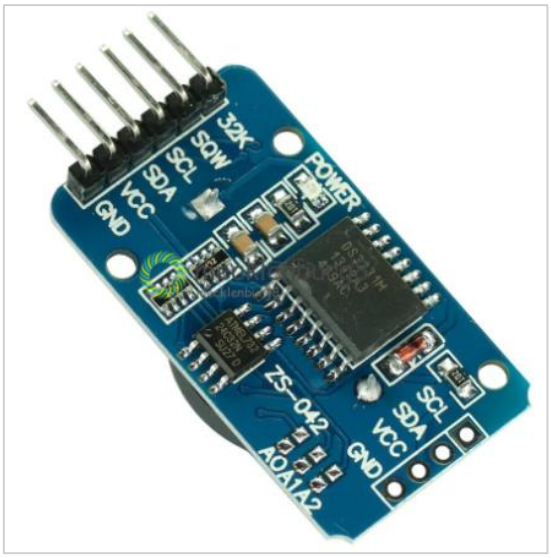

Figura 34. RTC(DS3231) [18]

```
void setup()  {
  Serial.begin(9600);
 setSyncProvider(RTC.get);
  if(timeStatus()!= timeSet)
     Serial.println("RTC no esta sincronizado");
  else
     Serial.println("RTC ha sido sincronizado");
}
```

Figura 35. Sincronizar con el RTC

Las dos primeras líneas cargan las librerías necesarias. Estas librerías están escritas de forma que se usa la función setSyncProvider () que se encarga de sincronizar el RTC.

En el setup se hace uso de timeStatus () para saber si el reloj está o no en hora. En caso de que no esté en hora se puede añadir al setup una línea más con la función setTime () para ajustar el reloj.

En mi caso he añadido:

```
SetTime (18,10,00,10,3,2021); // Las 18:10:00 del día 10 de marzo de 2021.
```

5.5.6 Lector tarjeta SD

Es un dispositivo que permite utilizar una tarjeta SD como dispositivo de almacenamiento. Las tarjetas SD y micro SD se han convertido en un estándar, reemplazando a otros medios de almacenamiento de datos debido a su gran capacidad y pequeño tamaño. Por lo tanto, se han integrado en una gran cantidad de dispositivos y actualmente se pueden encontrar en computadoras, tabletas y teléfonos inteligentes.

El lector SD apareció con anterioridad al micro SD. Por tanto, el módulo con micro SD es un modelo más actualizado que el lector SD. En ambos tipos, se puede leer a través del bus SPI, aunque puede tener otras interfaces, como buses *I2C* o *UART*.

Respecto a las tarjetas empleadas, se pueden encontrar en el mercado tarjetas SD, SDSC (Standard Capacity) o SDHC (High Capacity), pero no SDXC (Extended Capacity). La tarjeta debe ser formateada en sistema de archivos FAT16 o FAT32.

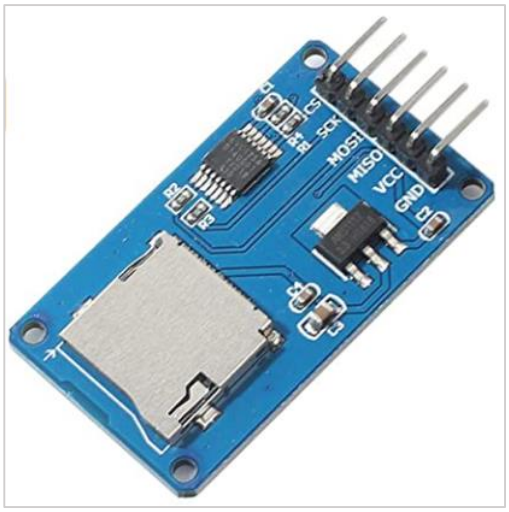

Figura 36. Lector tarjeta SD [19]

```
Serial.print("Iniciando SD card...");
digitalWrite(GPIO2, HIGH);

if (!SD.begin(GPIO2)) {
Serial.println("Fallo comunicacion o no existe SD");
return;}
Serial.println("SD Iniciada.");

dataFile = SD.open("LPS22HB.txt", FILE_WRITE);

if (dataFile) {
  Serial.println("<<< LPS22HB >>>");
  dataFile.println("<<< LPS22HB >>>");


  dataFile.close();}
  else {
  Serial.println("Error al abrir");}
```

Figura 37. Accediendo a escribir dentro de la tarjeta SD

Se inicializa el pin GPIO2 para la conexión con el lector SD. Se crea el objeto dataFile que permite crear un bloc de notas (“LPS22HB.txt”) donde se almacenaran los valores obtenidos del sensor LPS22HB.

A continuación, se puede ver cómo se guardan en el documento de texto (.txt) la fecha y la hora junto al valor de presión atmosférica (hPa) obtenido por el sensor LPS22HB.

LPS22HB: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
<<< LPS22HB >>>
18:26:19 10 5 2021
Pressure is : 759.68 hPa
18:26:20 10 5 2021
Pressure is : 759.68 hPa
18:26:21 10 5 2021
Pressure is : 759.68 hPa
18:26:22 10 5 2021
Pressure is : 759.68 hPa
18:26:23 10 5 2021
Pressure is : 759.68 hPa
18:26:24 10 5 2021
Pressure is : 759.68 hPa
18:26:25 10 5 2021
Pressure is : 759.68 hPa
18:26:26 10 5 2021
Pressure is : 759.68 hPa
18:26:27 10 5 2021
Pressure is : 759.68 hPa
18:26:28 10 5 2021
Pressure is : 759.68 hPa
18:26:29 10 5 2021
Pressure is : 759.68 hPa
18:26:30 10 5 2021
Pressure is : 759.68 hPa
```

Figura 38. LPS22HB.txt

5.6 Internet of Things (TTN)

Internet of Things construye una red de Internet de las cosas mediante la creación de conexiones de datos enriquecidas. La tecnología utilizada es *LoRaWAN*, que permite que las cosas se comuniquen con Internet sin 3G o WiFi. Cuenta con características como el bajo consumo de batería, el amplio radio de comunicación y el bajo ancho de banda.

La mayor contribución de TTN se produce en el desarrollo del servidor, donde la red se ha construido para admitir todas las puertas de enlace conectadas a través de él. Puede manejar mensajes duplicados, administrar descargas de mensajes, integrarse con la plataforma, etc. Otra ventaja proporcionada por TTN es la integración de API en lenguajes de programación como GO, Java, Node-RED o Node.js a través de *HTTP* y *MQTT*, lo que ayuda a crear una solución completa desde nodos sensores hasta aplicaciones de usuario final.

Figura 39. Red TTN Mundial

Una vez registrado en la plataforma, TTN nos proporcionará una consola para gestionar pasarelas y aplicaciones de forma sencilla e intuitiva.

Figura 40. Consola TTN [20]

5.6.1 The Things Indoor Gateway

Tiene como objetivo ser una puerta de enlace *LoRaWAN* totalmente compatible y de ultra bajo coste con WiFi como backhaul. La alimentación de la puerta de enlace se puede realizar a través de un enchufe de pared o a través de USB-C, lo que hace que la puerta de enlace sea la mejor opción para aplicaciones que requieren una cobertura dinámica.

The things Indoor Gateway LoRaWan tiene 8 canales y admite el protocolo BasicStation (el reenviador de paquetes de puerta de enlace de última generación de Semtech basado en WebSockets) y LBT(Listen-Before-Talk). Dispone de una configuración sencilla para conectarse a cualquier red a través de Wifi e incorpora una antena omnidireccional para uso en interiores. Se puede adquirir en diferentes versiones EU868, US915, AS923 y CN470.

Figura 41. The Things Indoor Gateway [21]

5.6.2 Conexión a la red de The Things Network

Para conectar esta puerta de enlace a la consola de *The Things Network*, se necesita registrar la puerta de enlace usando la opción “Estoy usando una versión anterior del reenviador de paquetes”.

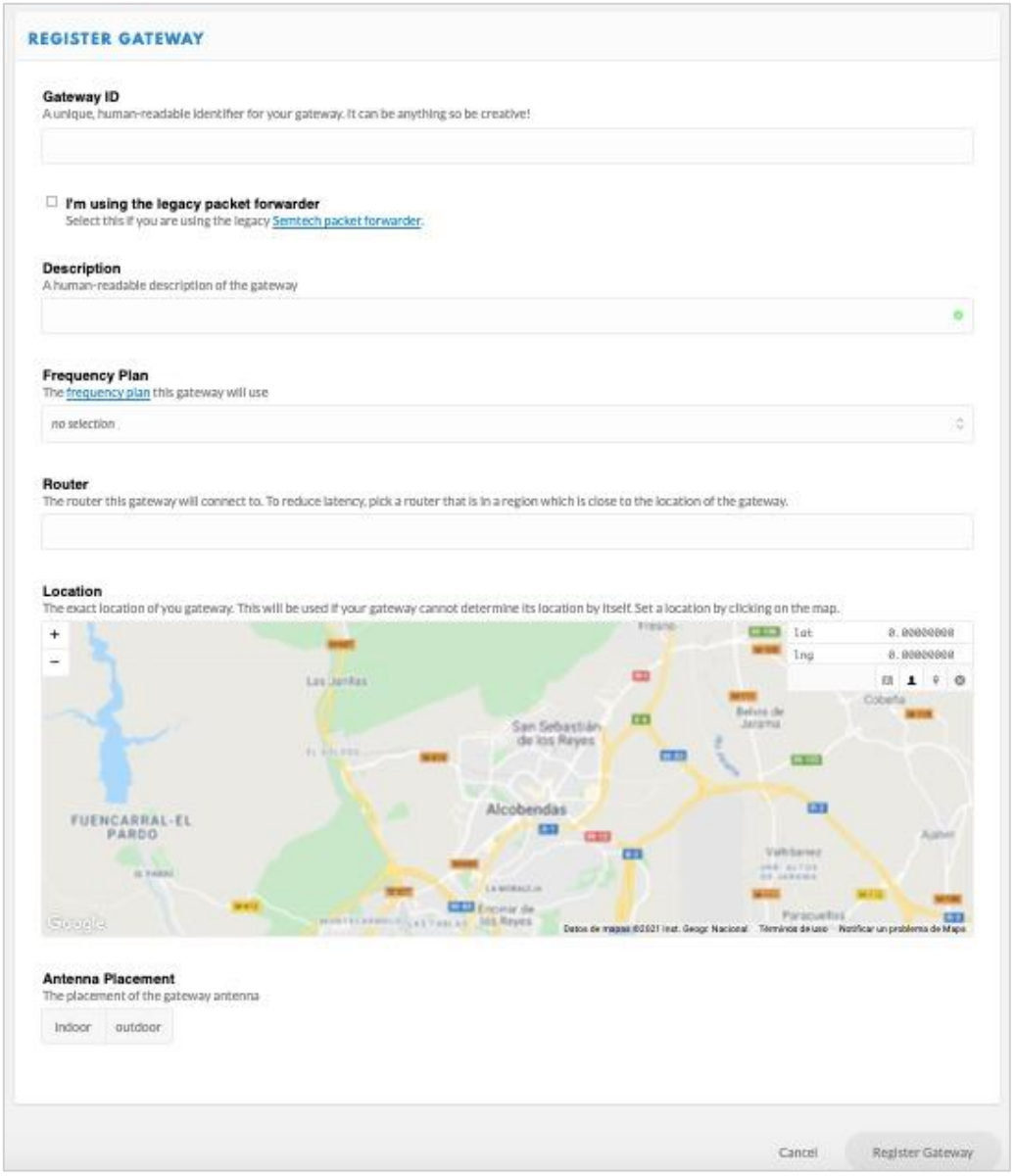

Figura 42. Registrar puerta de enlace [21]

La puerta de enlace EUI no es la dirección MAC de WiFi impresa en la parte posterior de *The Things Indoor Gateway*, sino que se deriva del primer número en la parte superior de la etiqueta que se encuentra debajo del código QR. Se puede configurar a través de la red WiFi.

Para obtener un EUI de 64 bits, se inserta FFFE después de los primeros 6 caracteres, que son el valor del campo "*Gateway EUI*" en la consola. Posteriormente, se añadirá la ubicación, el plan de frecuencia (868 Mhz) y el enrutador. Si la configuración se lleva a cabo con éxito, se empezará a recibir paquetes de datos.

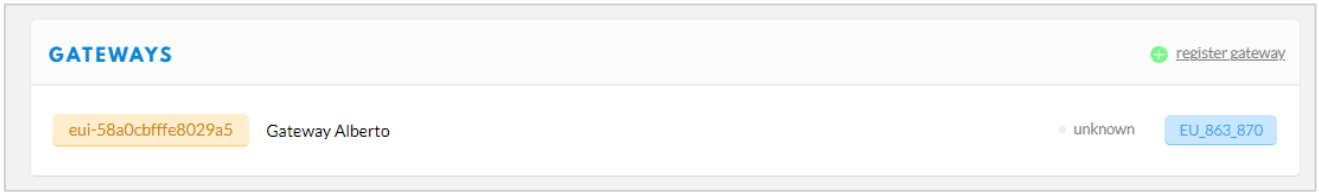

Figura 43. Gateway TTN

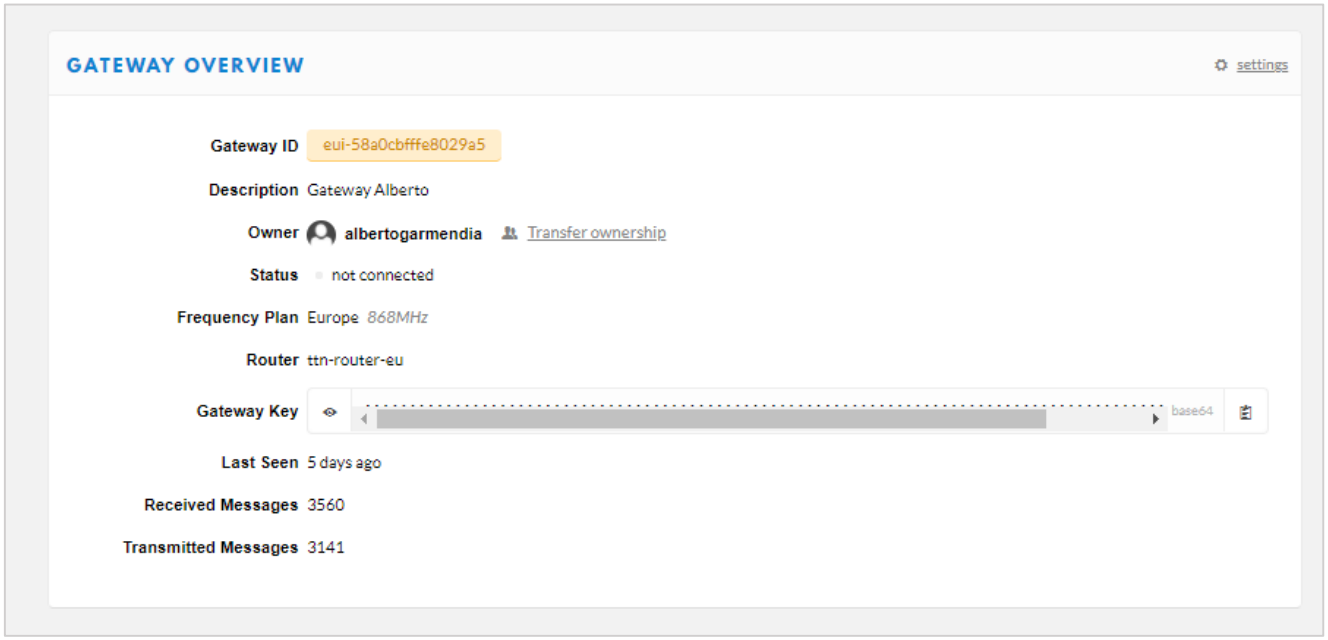

Figura 44. Configuración Gateway

Una vez completada la configuración, se mostrará el registro y el estado de nuestra pasarela en la plataforma. Si se accede a la pantalla de visualización, nos mostrará una descripción general del dispositivo.

5.6.3 Aplicaciones Internet of Things (TTN)

Estas aplicaciones permiten la integración de datos TTN con otras plataformas *IoT* (como *Node-RED*) y tienen una lista de *API* y *SDK* para la integración con otras plataformas.

- A continuación, se mostrará cómo agregar una aplicación:

El dispositivo puede comunicarse con aplicaciones que se encuentran en la plataforma. Para registrar un nuevo dispositivo, primero se debe agregar una aplicación.

Los pasos a seguir son:

- En la consola, hacer clic en Agregar aplicación.
- Para el ID de la aplicación, seleccionar un ID único con caracteres en minúscula, alfanuméricos y no consecutivos (no utilizar "- "y "_").
- Para el campo "Descripción de la aplicación", completar la descripción del dispositivo.
- Mantener la casilla de verificación habilitada para registrar automáticamente la aplicación en su zona predeterminada.

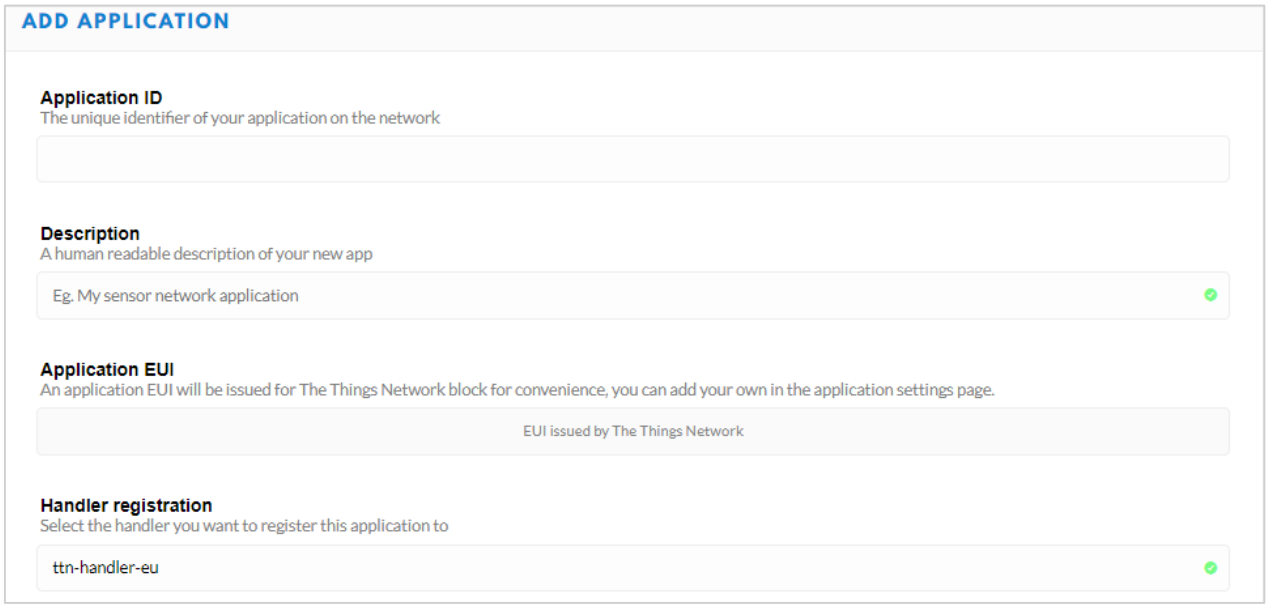

Figura 45. Agregar aplicación [22]

Una aplicación puede conectarse a *The Things Network* usando las siguientes formas:

1. **API**

El término API es una abreviatura de Interfaces de programación de aplicaciones. Es un conjunto de definiciones y protocolos que se utilizan para desarrollar e integrar software de aplicación, lo que permite la comunicación entre dos aplicaciones de software a través de un conjunto de reglas.

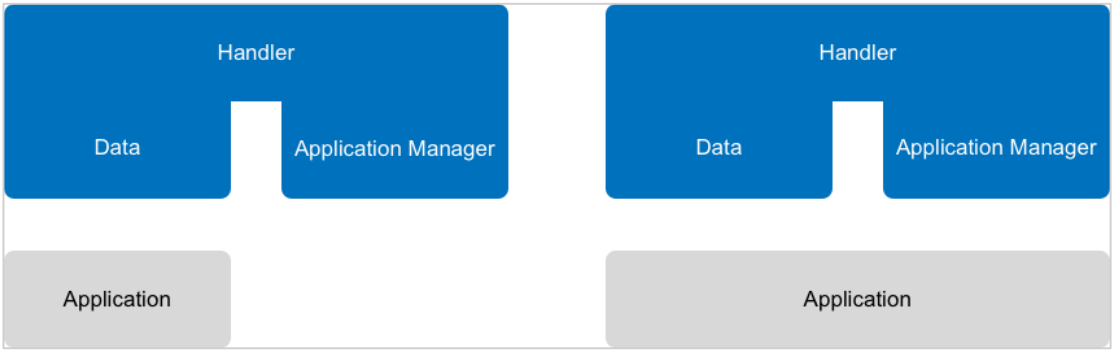

Figura 46. API de The Things Network (TTN) [23]

API de datos

La API de datos permite recibir eventos y mensajes desde el dispositivo. Se puede utilizar la API de datos de las siguientes formas: *MQTT* y *AMQP*.

API del administrador de aplicaciones

A través de la *API* de Application Manager se pueden administrar aplicaciones y dispositivos registrados. Para ello, se podrá utilizar la API de Application Manager de las siguientes formas: *gRPC* y *HTTP*.

2. **SDK**

Un kit de desarrollo de software (*SDK*) es un conjunto de herramientas y datos que pueden ayudar o incluso permitir a los programadores desarrollar programas en un lenguaje específico para desarrollar una plataforma o aplicación específica. La composición y distribución del SDK la realiza el desarrollador original del lenguaje, quien estará interesado en proporcionar software de terceros en el mercado para sus propios productos o lenguajes de programación. Por lo tanto, en la mayoría de los casos, los SDK son de uso gratuito, aunque sus fabricantes pueden restringir su uso a ciertas reglas y licencias.

El componente estándar incluido en casi todos los kits de desarrollo de software es la *API*, a través del cual se pueden vincular proyectos de software a nivel de código fuente. La interfaz de programación básica proporciona una gran cantidad de documentos, incluidas instrucciones sobre cómo usarla. Con ella, los desarrolladores pueden comprender rápidamente si su proyecto es factible y cómo desarrollarlo.

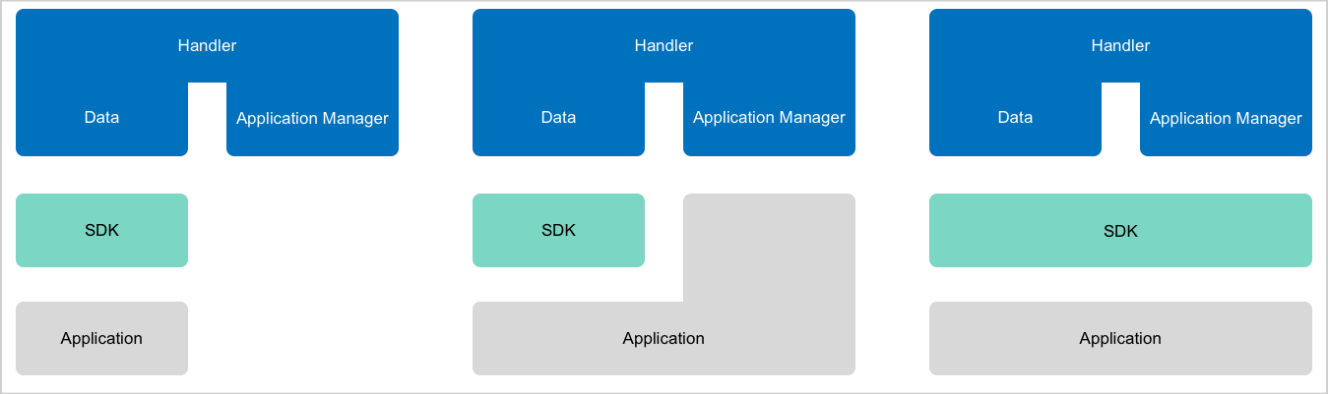

Figura 47. SDK [24]

3. Integraciones

La integración conecta su software con aplicaciones de terceros, por lo que pueden compartir información entre sí, así como datos de un sistema a otro, brindando de esta manera un mayor número de funciones y opciones al usar el software. Los datos que permiten la integración permanecerán en la nube, permitiendo modificar, eliminar y añadir cuando sea necesario.

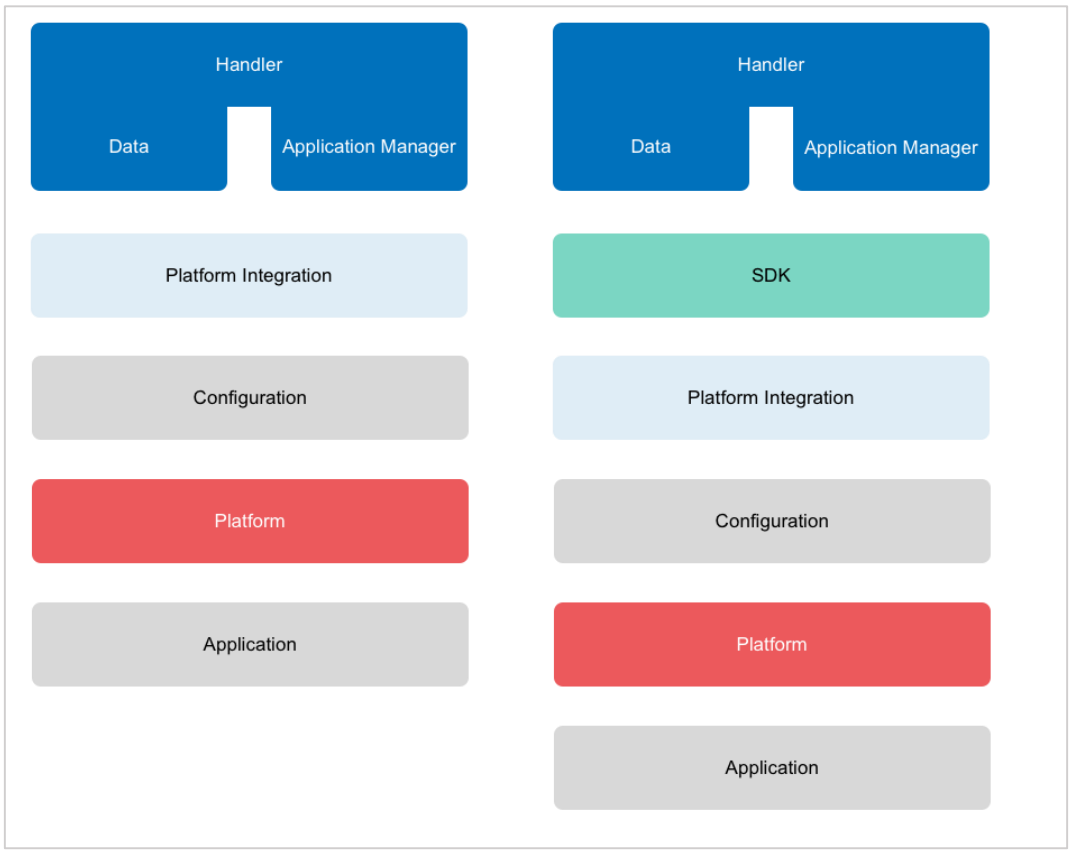

Figura 48. Integraciones [25]

Como ejemplos de la integración se pueden mencionar *ThingsBoard*, *Cayenne myDevices*, *ThingsSpeak*, *Azure IoT Hub* y *AWS IoT*, donde los usuarios pueden administrar sus aplicaciones y dispositivos. Esta integración es la encargada de sincronizar los datos con *The Things Network*.

5.7 Plataformas para representar gráficamente los datos a tiempo real

5.7.1 ThingSpeak

ThingSpeak ™ es un servicio de plataforma de análisis *IoT* que permite agregar, visualizar y analizar flujos de datos en tiempo real. Después de enviar datos desde el dispositivo a *ThingSpeak*, se puede crear una visualización de datos instantánea sin escribir ningún código.

Figura 49. ThingSpeak [26]

Ésta es una de las plataformas de código abierto más completas. La cantidad de posibilidades que ofrece sigue estando lejos de *AWS IoT* y *Azure IoT*, pero es capaz de suplir las carencias integrándose con plataformas de terceros. Está respaldada por una empresa como *MathWorks*, especializada en software de informática matemática.

Su código está disponible en GitHub y dispone de una gran comunidad de desarrolladores que dan soporte a una gran variedad de dispositivos.

Proporciona compatibilidad con los siguientes dispositivos:

- Arduino.
- Los módulos de fotones y electrones de partículas.
- Módulo WIFI ESP8266.
- Raspberry Pi.

Por otro lado, cualquier dispositivo que pueda comunicarse con la APIs RESTful también podrá hacerlo con la plataforma.

Esta solución tiene una amplia lista de aplicaciones y plataformas.

Finalmente, el análisis y la visualización de los datos será a través de MATLAB, que ha sido desarrollado por la misma empresa matriz como esta plataforma.

Sus principales ventajas son:

- Es gratuita.
- Sencilla configuración de la plataforma.
- Permite el uso de código de Matlab.

Sus principales inconvenientes son:

- No poder mostrar un intervalo de tiempo gráficamente en la visualización, pudiendo sólo seleccionar el número de días y la cantidad de puntos para el gráfico.
- Cuando el código MATLAB se inserta en los gráficos de datos desde un período de tiempo anterior, los datos no se pueden seleccionar en su fecha, siendo necesario saber cuántos mensajes se han enviado desde ese momento.

Funcionamiento

La plataforma funciona en un concepto básico llamado "*ThingSpeak Channel*", que es un canal que almacena la información enviada por los dispositivos. Se compone de las siguientes partes:

- Ocho campos para almacenar datos de todo tipo.
- Tres campos para almacenar información (latitud, elevación y longitud).
- Un campo llamado “Estado”, utilizado para describir la información del canal.

Los datos se transfieren a través de los protocolos *HTTP* y *MQTT*.

Una vez creado el canal, los dispositivos pueden enviar mensajes con los atributos que han sido definidos previamente.

Con la integración en MATLAB, se puede analizar la información y presentar de forma gráfica.

Finalmente, los eventos pueden configurarse para enviar notificaciones o usar dispositivos.

Sus principales características son:

- Recopilar datos a través de canales privados.
- Compartir datos con canales públicos.
- Protocolos (API RESTful y MQTT).
- Análisis de MATLAB®.
- Programación de eventos.
- Integraciones de aplicaciones.

Los pasos a seguir para configurar *ThingSpeak* son:

- Crear una cuenta MathWorks gratuita en ThingSpeak.
- Crear un canal nuevo.
- Completar los datos en el canal seleccionado:
  - Identificador del canal (ID) que se muestra en la parte superior de la vista.
  - Introducir la clave API, que se puede encontrar en la pestaña "Claves API".

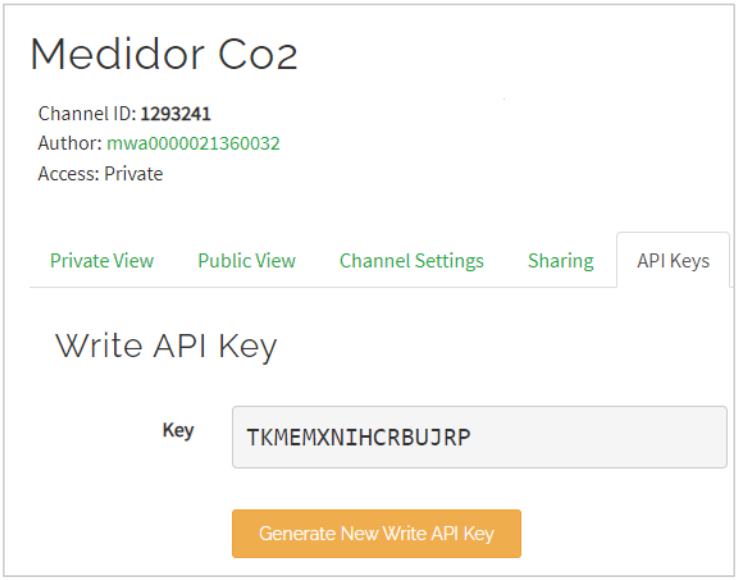

Figura 50. KEY e ID ThingsSpeak [27]

Creación de la integración en *The Things Network*

En *The Things Network Console* , se accede a la aplicación creada y se sigue la siguiente ruta:

"Integraciones>agregar una integración>ThingSpeak".

La pestaña "*Payload Formats*" de la aplicación *The Things Network* sirve para controlar la carga útil enviada a *ThingSpeak*. Su carga útil debe estar en un formato JSON.

Se introduce la clave API de escritura en el campo "Autorización" y la ID de canal en el campo "ID".

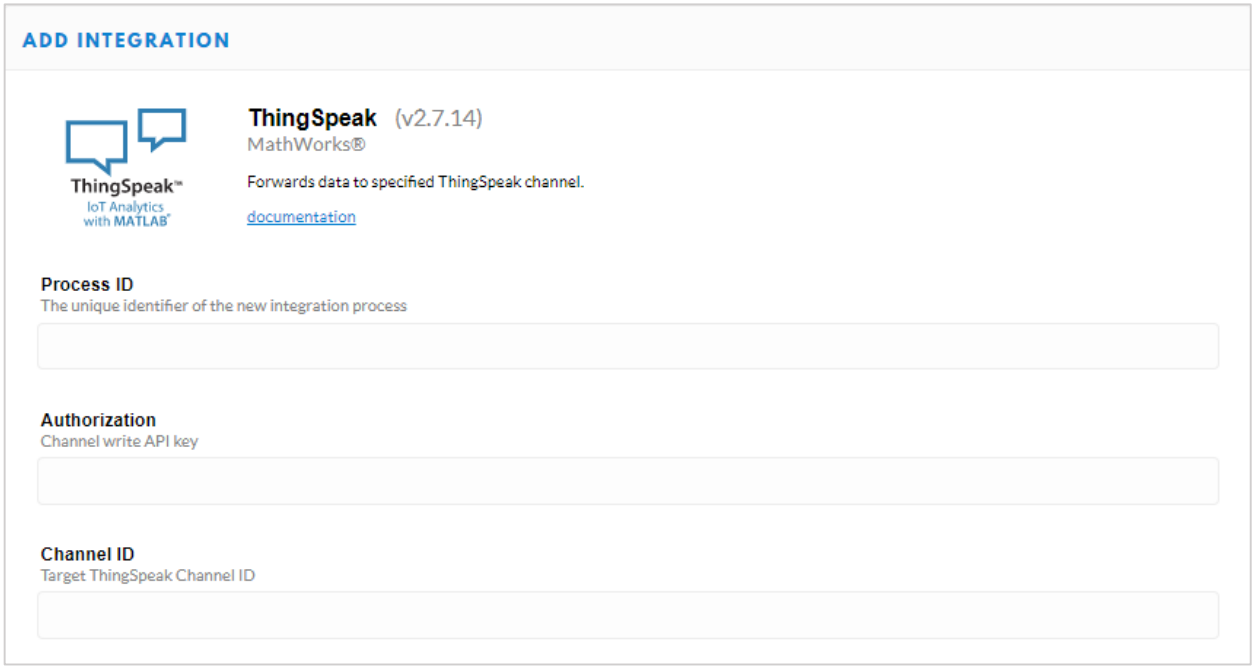

Figura 51. Integración ThingSpeak

Una vez introducida la clave API de escritura y la ID del canal, la integración queda configurada.

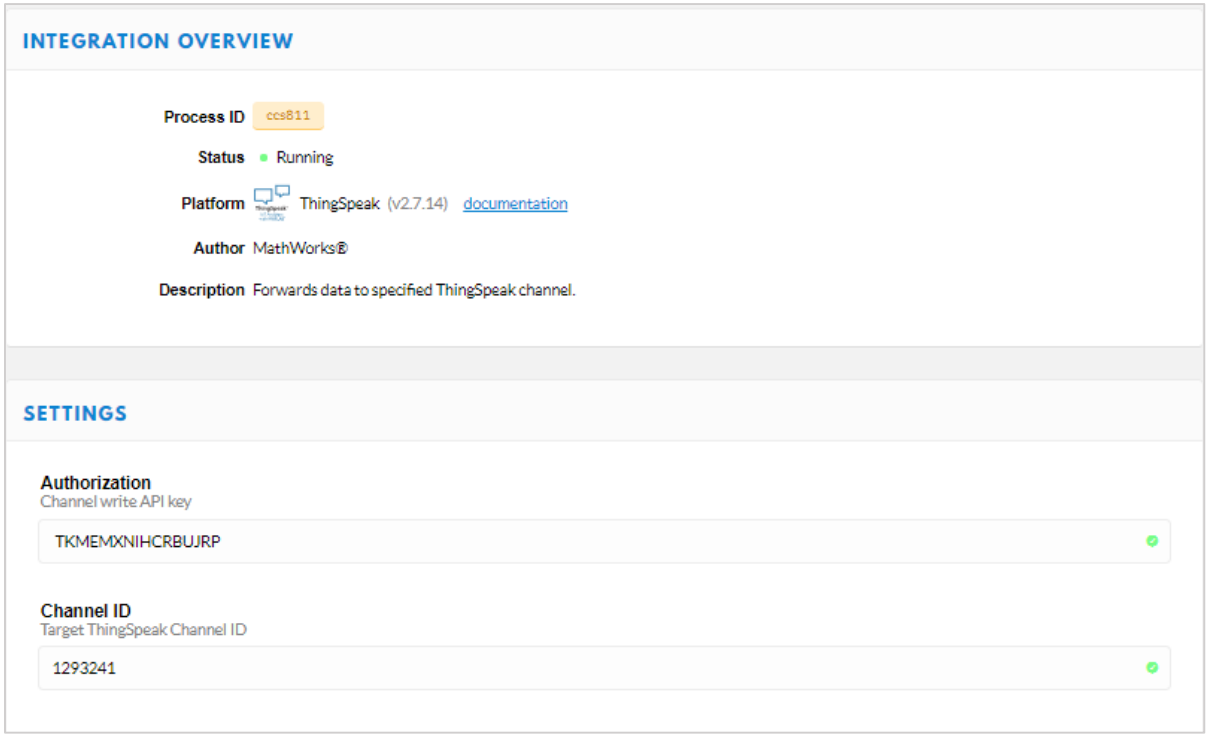

Figura 52. Configuración de la Integración ThingSpeak

Al tener creado el canal se procede a su configuración.

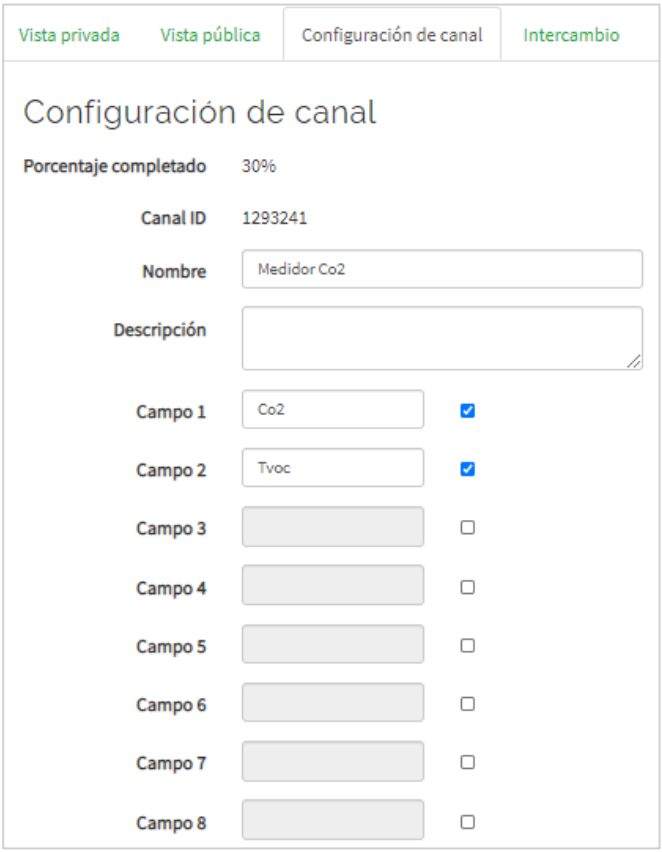

Figura 53. Configuración del canal de ThingSpeak (1) [27]

- Porcentaje completado: calculado en base a los datos ingresados en los distintos campos de un canal. Se debe introducir el nombre, la descripción, la ubicación, la URL, el video y las etiquetas para completar el canal.
- Nombre del canal: introducir un nombre único para el canal *ThingSpeak*.
- Descripción: introducir una descripción del canal *ThingSpeak*.
- Número de campo: seleccionar la opción para habilitar el campo e introducir un nombre de la variable a representar. Cada canal de *ThingSpeak* puede tener hasta 8 campos.

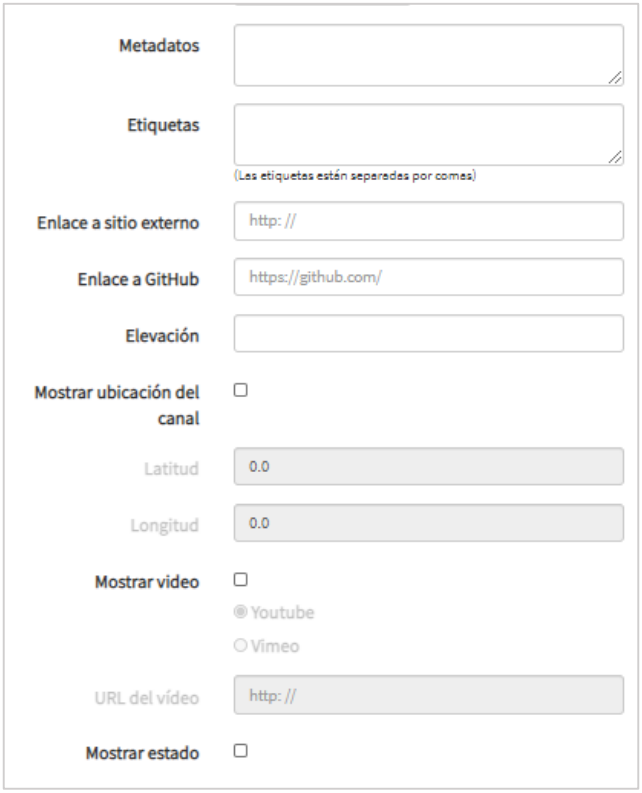

Figura 54. Configuración del canal de ThingSpeak (2) [27]

- Metadatos: introducir información sobre los datos del canal, incluidos datos JSON, XML o CSV.
- Etiquetas: introducir palabras clave que identifiquen el canal separando las etiquetas con comas.
- Enlace a un sitio externo: si tiene un sitio web que contiene información sobre su canal *ThingSpeak*, puede añadir la URL.
- Mostrar ubicación del canal.
- Latitud: especificar la posición en grados decimales.
- Longitud: especificar la posición en grados decimales.
- Elevación: especificar los metros.
- URL del video: si dispone de un video de YouTube ™ o Vimeo® que muestre la información de su canal, puede introducir la ruta completa de la URL del video.
- Enlace a GitHub: si se almacena el código *ThingSpeak* en GitHub®, puede introducir la URL del repositorio de GitHub.

Se dispone de 3 widgets para representar los datos:

Figura 55. Widget

Si se selecciona el widget “Indicador”, se mostrará el siguiente recuadro:

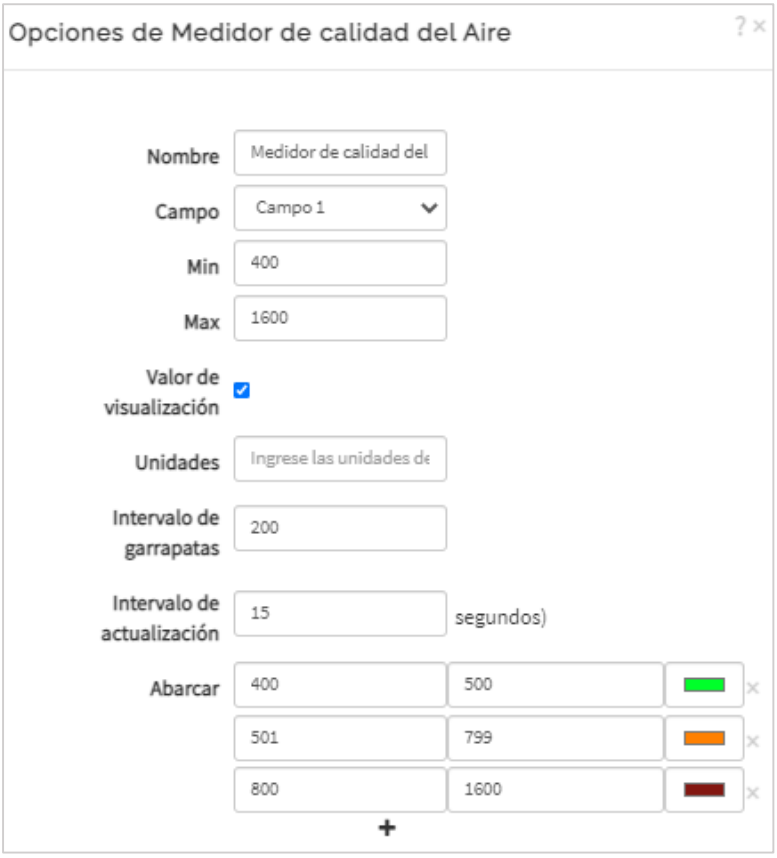

Figura 56. Configuración del indicador

En el campo “Abarcar”, se puede establecer una serie de colores en función de cada rango de valores.

Figura 57. Indicador del Medidor de calidad del Aire

También permite usar el widget ”Indicador de lámpara”, donde el color seleccionado se mostrará si se cumple la condición establecida.

En este caso, se mostrará en color rojo el indicador de lámpara si el valor del campo 1 es mayor de 799 ppm.

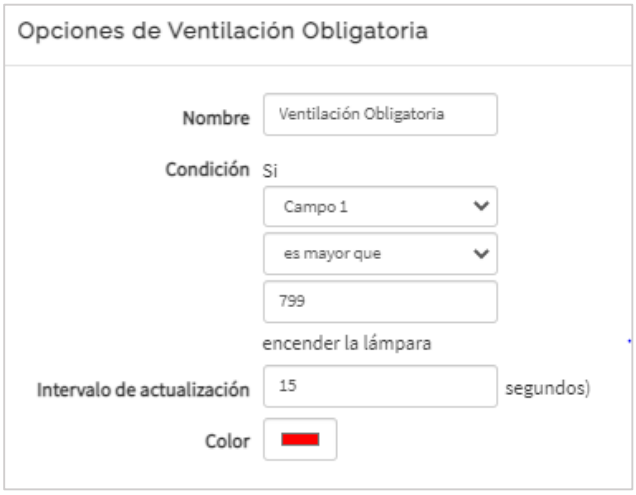

Figura 58. Configuración Ventana

Figura 59. Indicador de Lámpara

5.7.2 Cayenne myDevices

Cayenne myDevices es una plataforma de desarrollo de prototipos para dispositivos *IoT*. Una de sus mayores ventajas es una solución visual de arrastrar y soltar, es decir, nos permite configurar el sistema sin programar. Se utilizará *Cayenne* como herramienta para visualizar datos históricos y en tiempo real enviados a través de *The Things Network*. *Cayenne myDevices* ofrece una amplia gama de posibilidades. Cuenta con el respaldo de importantes empresas tecnológicas como Microchip y Smetch, además del hardware gratuito Arduino. Permite crear un panel de control de una forma muy sencilla, arrastrando y soltando widgets para visualizar, gestionar y controlar dispositivos *IoT* conectados a través del agente *MQTT*. Entre todos los servicios que tenemos al alcance, resalta el control y monitoreo remoto de alarmas, advertencias, programación de eventos, procesamiento de datos, rastreo de dispositivos, códigos personalizados, *API MQTT* y compatibilidad con redes *LoRaWan*.

La plataforma es conocida por su simplicidad para conectar dispositivos, analizar datos y representarlos gráficamente. Actualmente es necesario el uso de las librerías, por lo que la compatibilidad de hardware es bastante reducida.

Los dispositivos compatibles actualmente son:

- Raspberry Pi (1,2,3 y Zero).
- Arduino.
- ESP8266.
- Redes y dispositivos LoRaWan.

Estos dispositivos pueden comunicarse a través de la *API de MQTT*. La única restricción se produce al hacer llamadas a la API, ya que cada cliente solo puede enviar 60 mensajes por minuto usando su IP. Este límite no se refiere al número de dispositivos, sino al recuento global de mensajes enviados e intentos de conexión. El modelo de negocio de la plataforma se basa en el mercado de aplicaciones *IoT* y en crear soluciones completas para los clientes.

Figura 60. Cayenne MyDevices [28]

Creación de una cuenta myDevices

A través de la página web *Cayenne myDevices*, se creará una cuenta que nos permite la conexión entre múltiples placas con sensores. En esta cuenta, se accede a agregar un dispositivo siguiendo esta ruta: "Devices & Widgets" -> LoRa-> Actility-> Cayenne LPP.

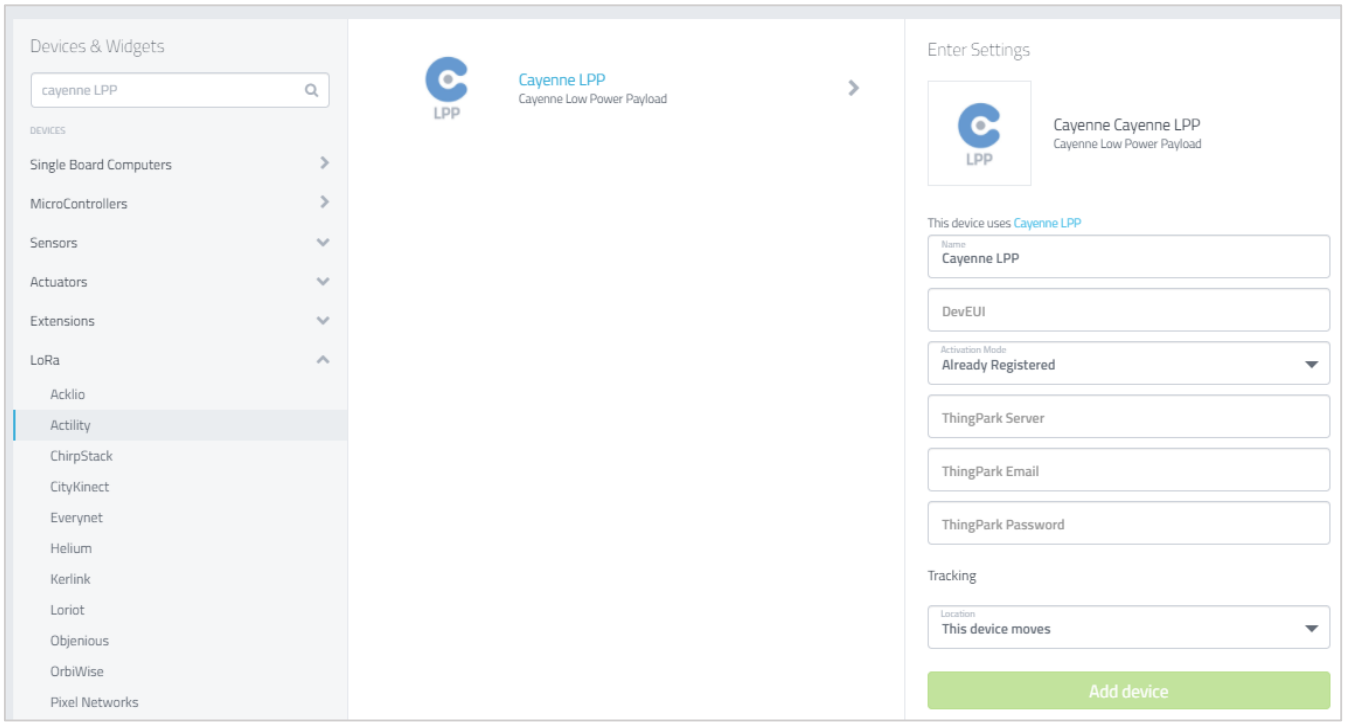

Figura 61. Creación de una cuenta MyDevices conectándola con el dispositivo[29]

Es necesario añadir en el campo "DevEUI" el identificador único de cada dispositivo para realizar la conexión con la plataforma *Cayenne Mydevices*.

Una vez creada la cuenta, se debe instalar la librería *Cayenne*, que permitirá conectar nuestro módulo *Heltec Cubecell-GPS* a dicha plataforma. Esta acción se realiza mediante el gestor de librerías, siguiendo esta ruta:

Programas > Incluir Librería > Gestionar Librerías.

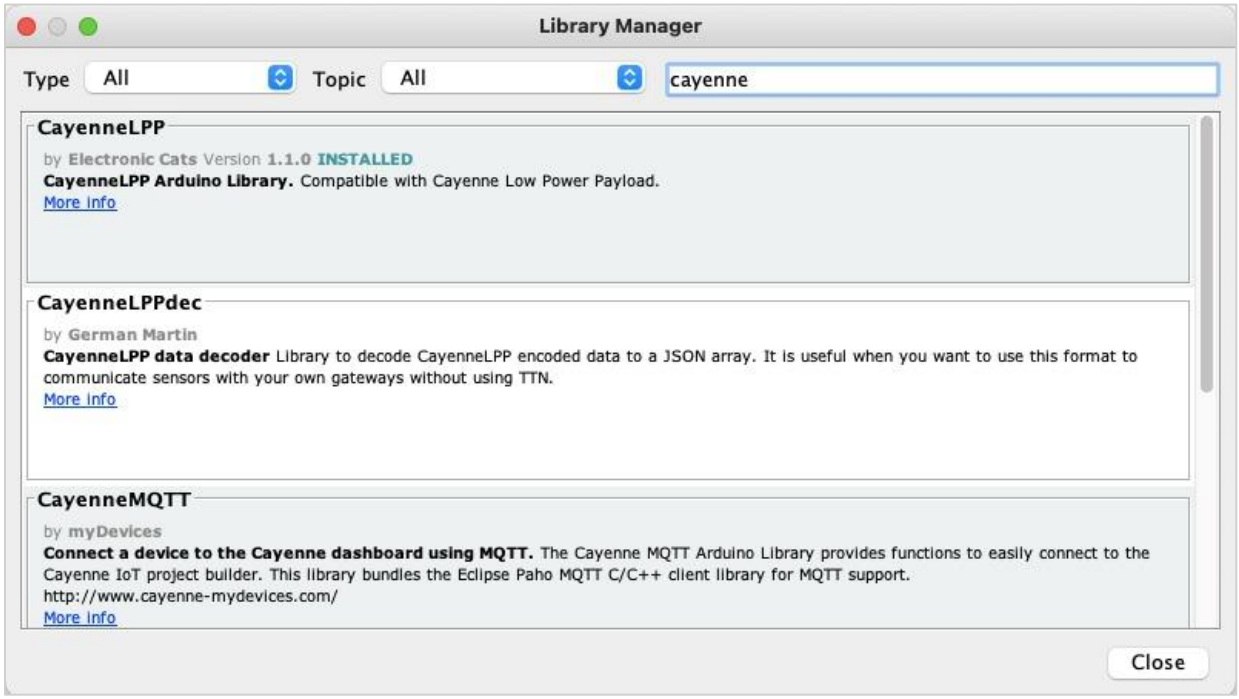

Figura 62. Librería Cayenne LPP [30]

Esta librería instalará tanto el código como los ejemplos dentro de ella, quedando todo preparado para empezar a configurar *Cayenne LPP*.

Se crea una aplicación en la consola *The ThingsNetwork*, donde se recibirán los datos obtenidos del sensor UVM30A.

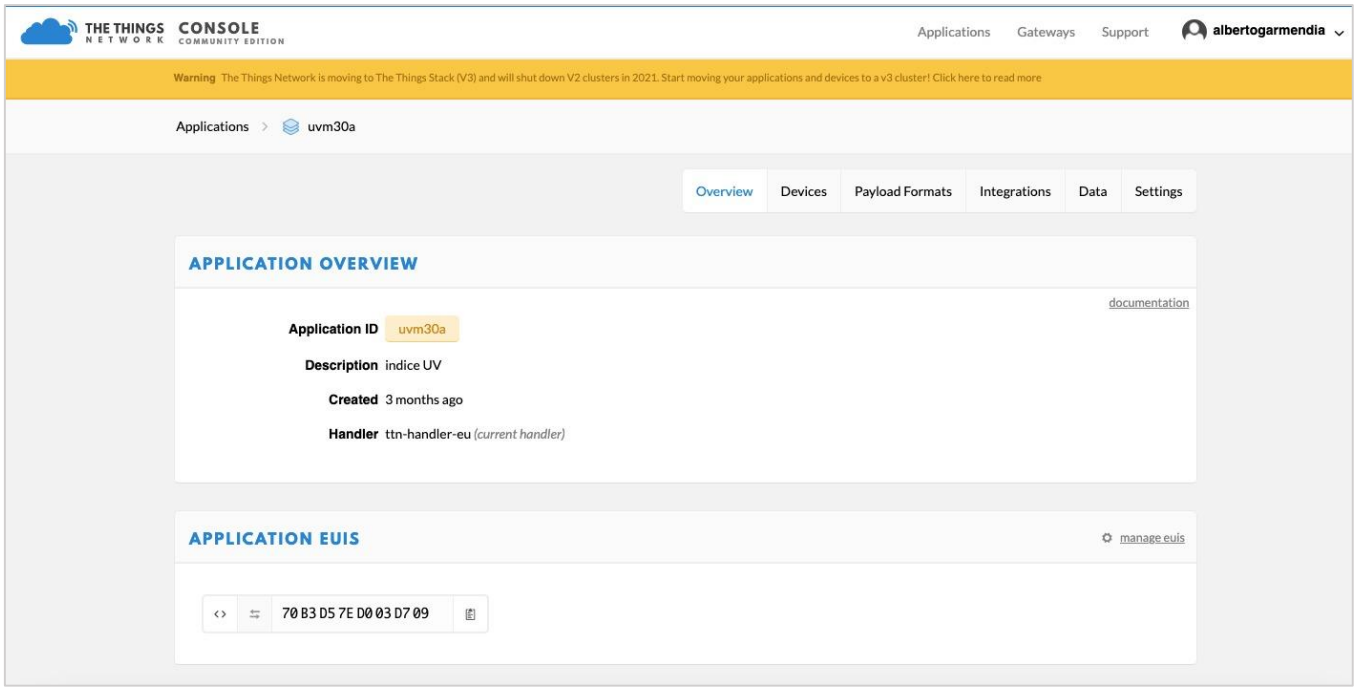

Figura 63. Aplicación de The Things Network para recibir la radiación ultravioleta del sensor UVM30A

Para mostrar su contenido en la plataforma de *Cayenne myDevices*, la carga útil debe estar codificada con *Cayenne Low Power Payload* (Cayenne LPP).

CayenneLPP contiene metadatos para que *Cayenne* comprenda qué datos se envían a su plataforma. Se necesita definir el identificador y qué tipo de datos son.

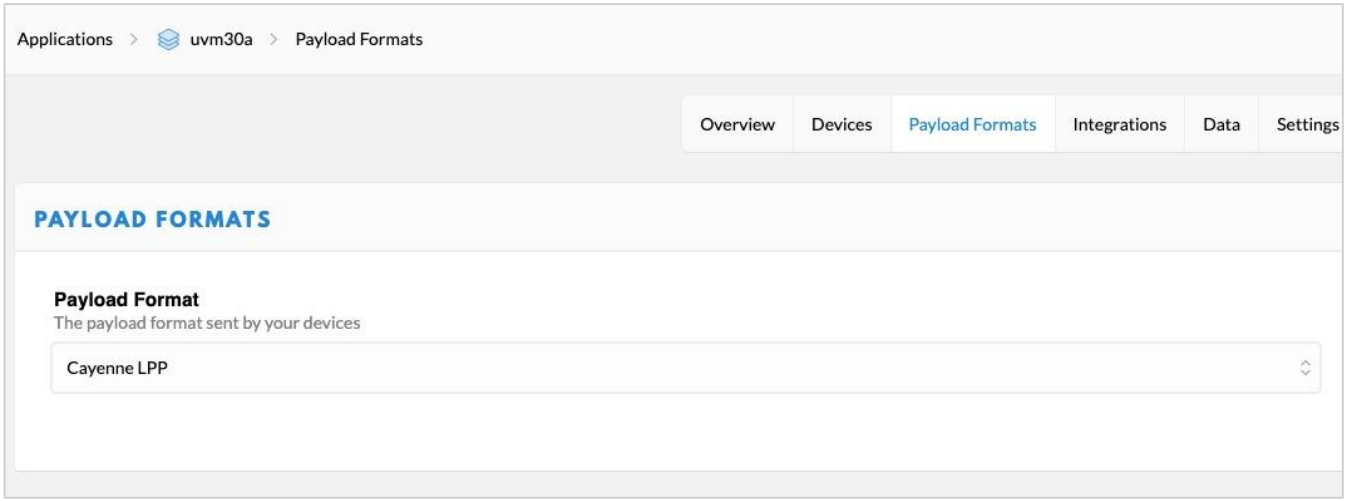

Figura 64. Payload Cayenne LPP [31]

Cayenne Low Power Payload (LPP) proporciona una forma cómoda y sencilla de enviar datos a través de redes *LPWAN* como *LoRaWAN*. Cayenne LPP cumple con la restricción de tamaño de carga útil, que se puede reducir a 11 bytes y permite que el dispositivo envíe múltiples datos de sensores a la vez. Además, *Cayenne LPP* permite que el dispositivo envíe diferentes datos de sensores en diferentes marcos. Para eso, los datos de cada sensor deben tener como prefijo dos bytes:

- **Data Channel**: identifica de forma única cada sensor en el dispositivo.
- **Data Type**: identifica el tipo de datos en el marco, por ejemplo, "temperatura".

El parámetro “channel” actúa como una clave para el campo de datos. Los campos de datos que se envían son dinámicos y se puede enviar datos de forma selectiva siempre que el canal coincida.

Lpp.addLuminosity(uint8_t channel, uint16_t lux) -> Devuelve los valores de los parámetros del Índice UV.

Lpp.addGPS(uint8_t channel, float latitud, float longitud, float metros) -> Devuelve la latitud, la longitud y la altura del módulo GPS Air530Z.

```
lpp.addLuminosity(1,voltage);
lpp.addLuminosity(2,UVIndex);
lpp.addGPS(3, lat,lon,alt);
lpp.getBuffer(),
appDataSize = lpp.getSize();
memcpy(appData,lpp.getBuffer(),appDataSize);
```

Figura 65. A través de Cayenne LPP enviando el voltaje y el índice UV.

A continuación, se necesita integrar *MyDevices* a la plataforma *The things Network* para poder recibir los datos.

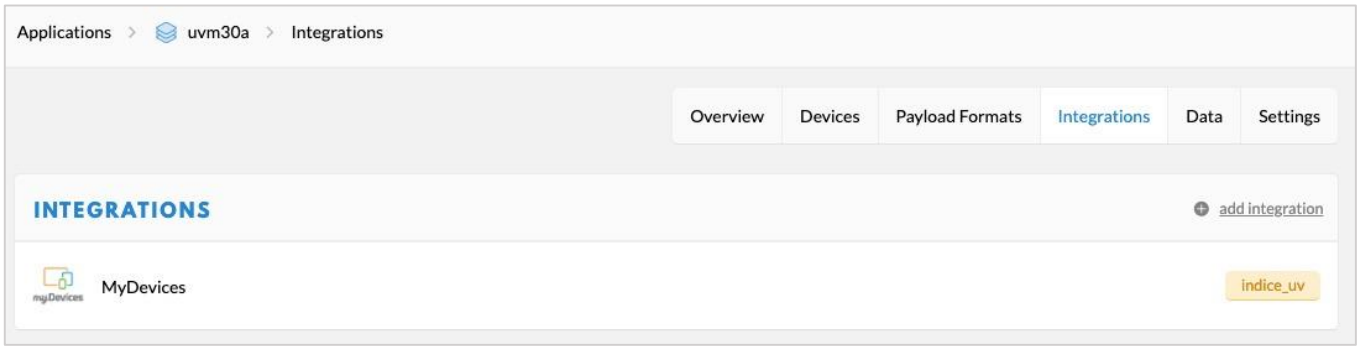

Figura 66. Intregración myDevices en The Thing Network

Por último, se visualiza el módulo GPS, el voltaje y el índice UV en la plataforma de *Cayenne*:

- Módulo Air530Z.

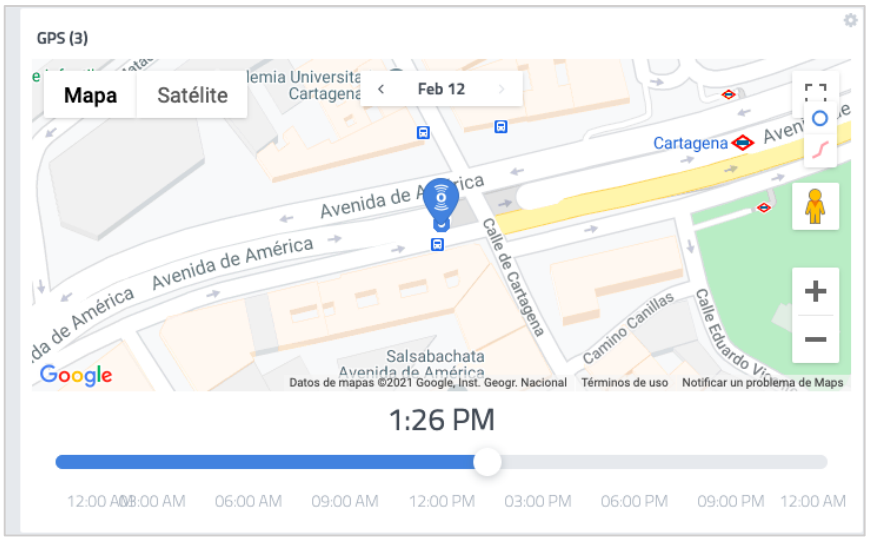

Figura 67. GPS donde se encuentra el dispositivo.

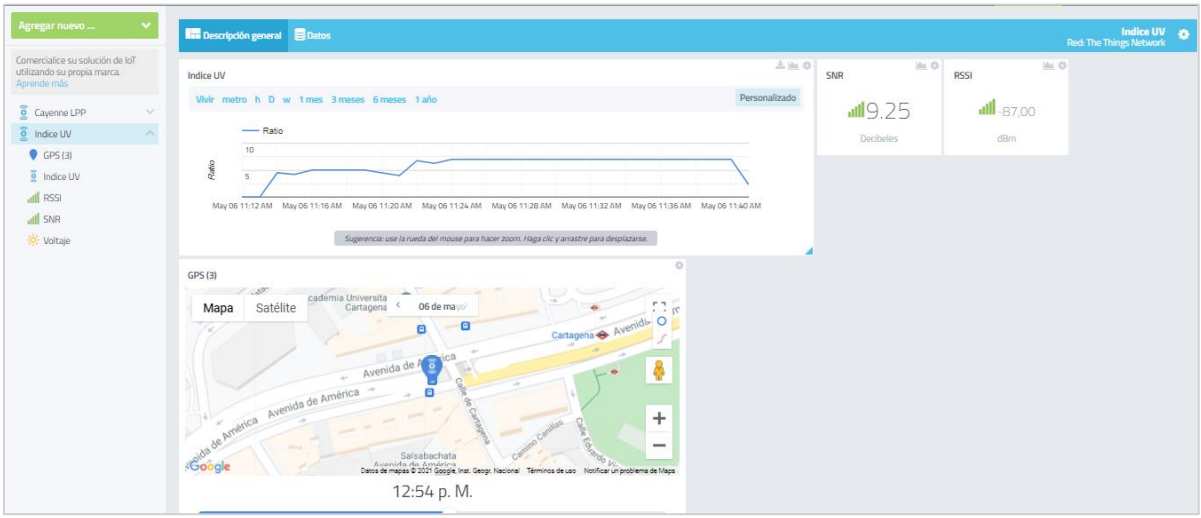

Figura 68. Plataforma Cayenne myDevices

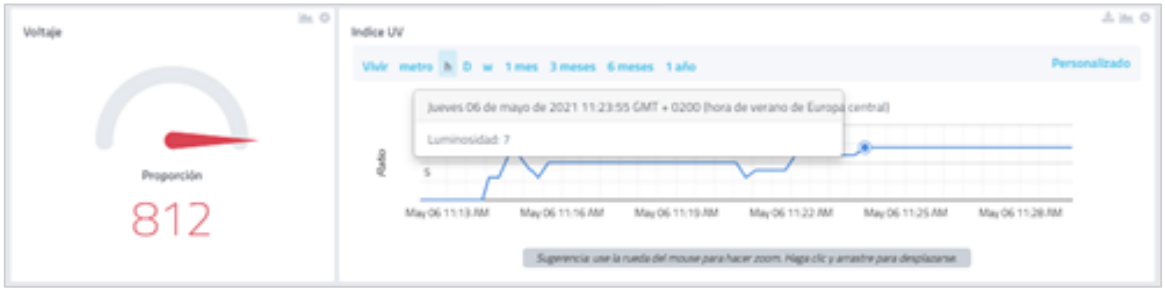

Figura 69. Plataforma Cayenne Luminosidad

5.7.3 Thingsboard

ThingsBoard es una plataforma *IoT* de código abierto que permite la gestión y escalabilidad de *IoT*. Dispone de dos soluciones, una local y otra en la nube.

Figura 70: ThingsBoard [32]

Características

- Analizar y representar los datos recibidos.
- Permite la utilización de bases de datos *SQL* y *NoSQL*.
- Paneles que muestran información en tiempo real (telemetría).
- Dispone de una gran variedad de plugins para visualizar la información.
- Desarrollada en Java 8.
- Posibilidad del uso de alarmas/notificaciones y de reglas personalizables.
- Controlar dispositivos en remoto (*RPC*).
- Posibilidad de envío de datos a otros sistemas compatibles.

Arquitectura

Esta plataforma fue diseñada para cumplir con los siguientes requisitos:

- Fiable: sin pérdida de datos.
- Eficiente y Robusto.
- Facilidad para implementar funciones con widgets personalizables.
- Escalable: respaldada con tecnologías líderes en código abierto.

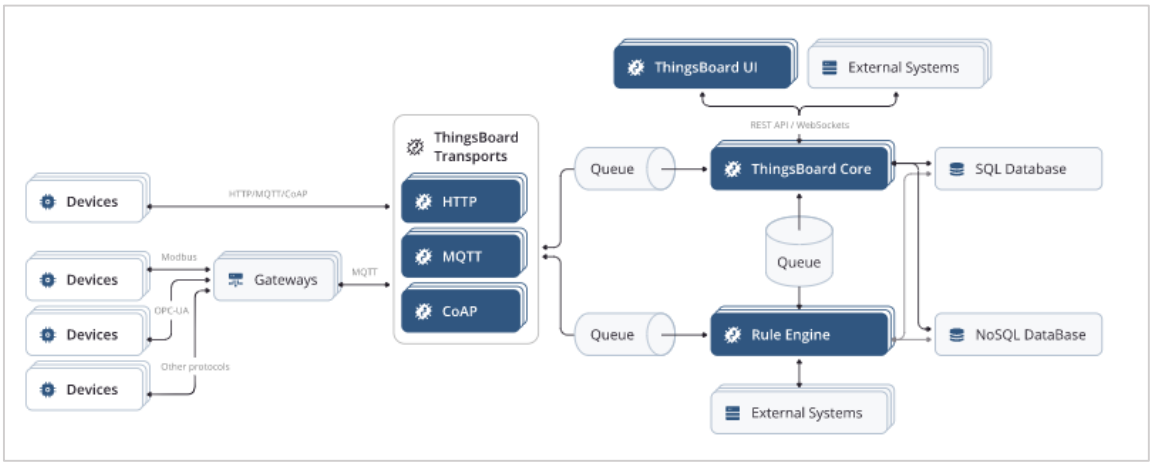

Figura 71. Diagrama de Arquitectura de ThingsBoard [33]

Transporte de ThingsBoard

Dispone de *API* basadas en *MQTT* , *HTTP* y *CoAP* que están disponibles para las aplicaciones. Cada *API* de protocolo forma parte de la "Capa de transporte" de *ThingsBoard*.

Una vez que se recibe el mensaje del dispositivo, se analiza y se envía a la cola de mensajes . La entrega del mensaje sólo será posible si la cola de mensajes ha reconocido el mensaje enviado.

ThingsBoard Core

ThingsBoard Core es el encargado de interactuar con las llamadas a la *API REST* y las suscripciones a *WebSocket* . Almacena información en tiempo real sobre los dispositivos y el estado de la conectividad.

Motor de reglas de ThingsBoard

ThingsBoard Rule Engine es el responsable de manejar los mensajes entrantes del sistema y puede operar como modo aislado o compartido. En el modo aislado, se puede programar para recibir sólo mensajes de un inquilino. En el modo compartido, el motor de reglas recibe mensajes de varios inquilinos.

Rule Engine se suscribe a la cola de datos entrantes y reconoce el mensaje sólo si ha sido procesado anteriormente. Existen varias estrategias que controlan el orden, el procesamiento del mensaje y los criterios de reconocimiento del mensaje.

Interfaz de usuario web de ThingsBoard

ThingsBoard proporciona una interfaz gráfica estática que ha sido desarrollada utilizando el framework "*Express.js*" (infraestructura de aplicaciones web). Una vez iniciada la interfaz gráfica, la aplicación empieza a usar la *API Rest* y la *API WebSockets* proporcionados por *ThingsBoard Core*.

Configuración de una aplicación en TheThingsNetwork

El primer paso es crear una aplicación en la consola *TheThingsNetwork*.

- ID de aplicación: es un identificador único para cada aplicación.
- Registro de controlador: se utiliza para identificar la región donde se registra la aplicación. En nuestro caso: - ttn-handler-eu.

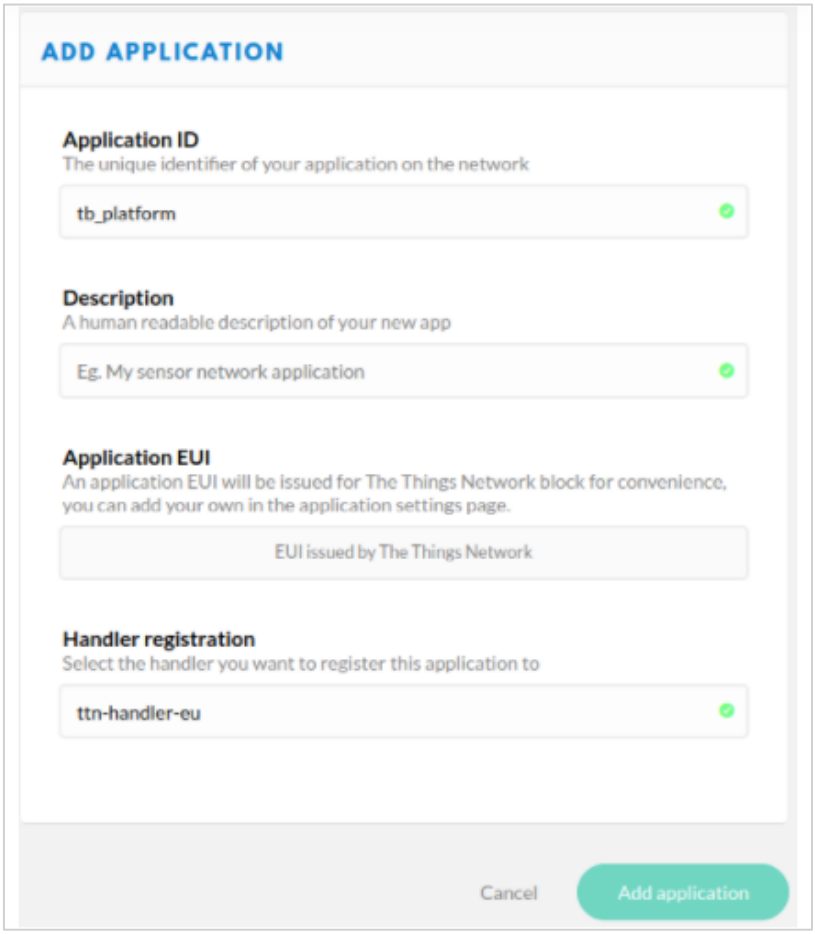

Figura 72. Agregar Aplicación para ThingsBoard [34]

Decodificador de carga útil

Existen 2 opciones para decodificar los datos:

- Decodificador de *TheThingsNetwork*: los datos se decodificarán antes de ingresar a la plataforma *ThingsBoard*.
- Convertidores de tabla de objetos: los convertidores de enlace ascendente / descendente se utilizarán para decodificar datos de formato binario en *JSON.*

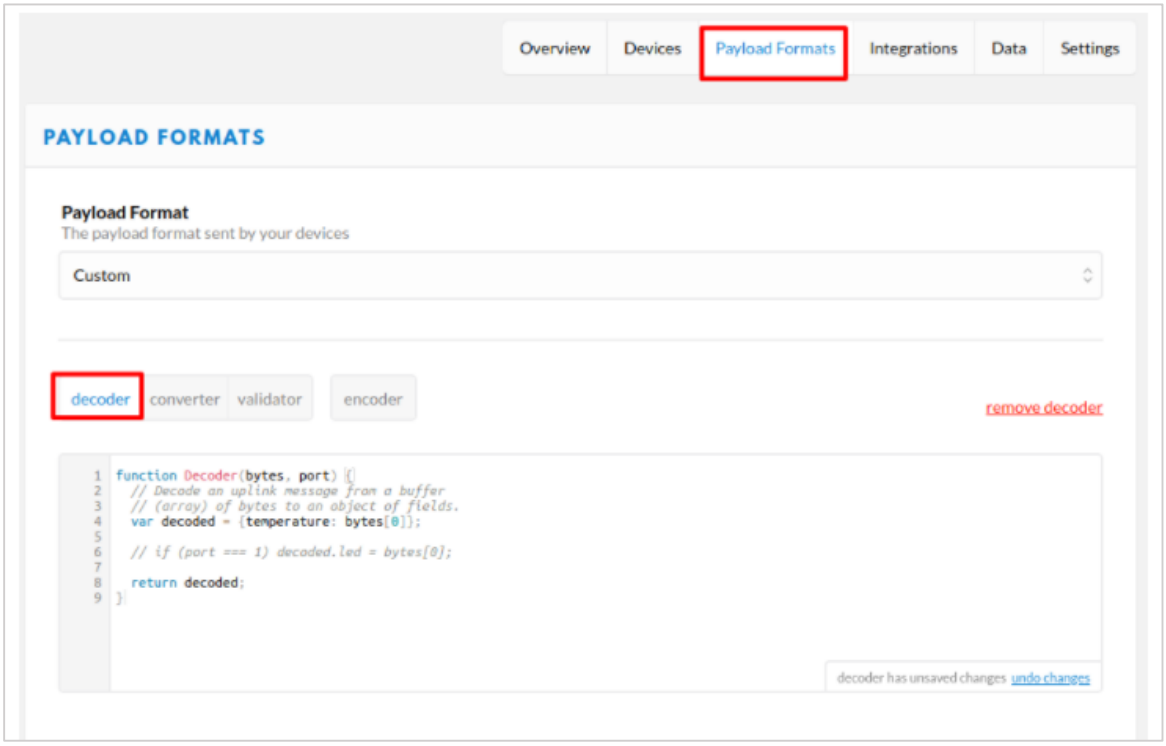

Figura 73. Payload Format [34]

Registro de dispositivos en TheThingsNetwork

El siguiente paso es la creación de un dispositivo en TTN, registrando el dispositivo:

- ID de dispositivo: es el identificador del dispositivo, en este caso “dht22_sensor”.
- Dispositivo EUI: presione el botón “Generar” para generar una identificación aleatoria.

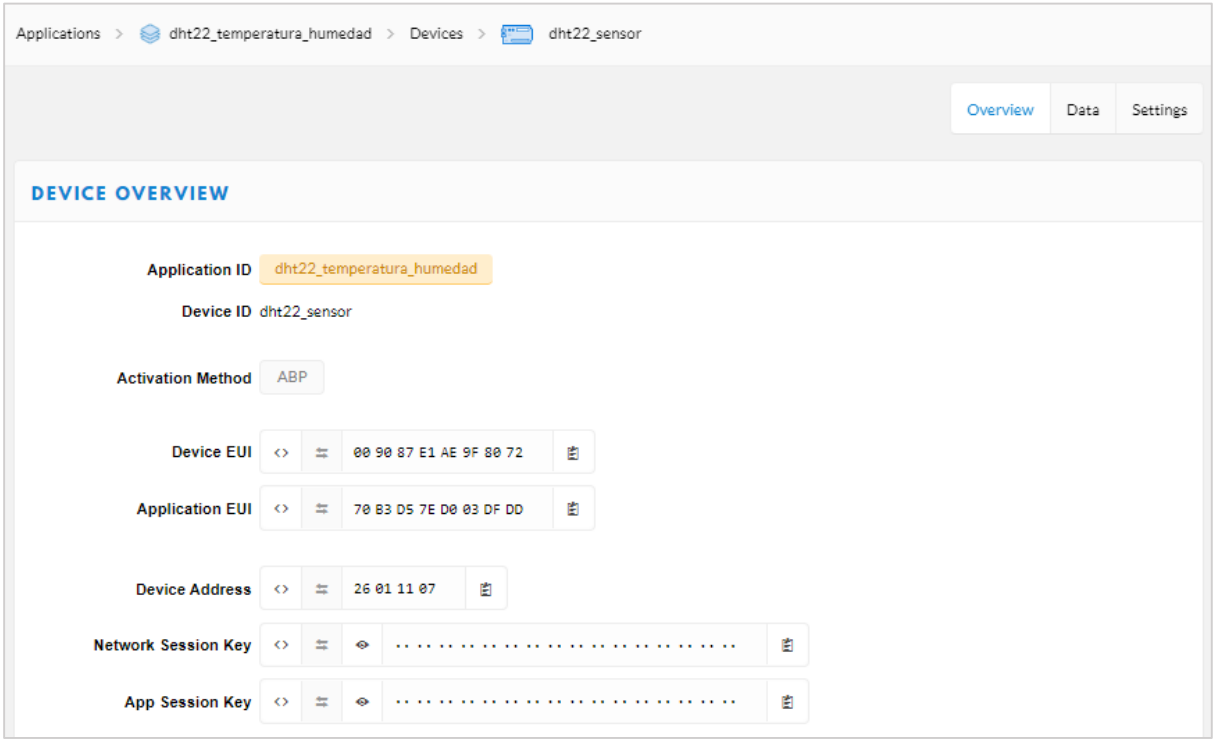

Figura 74. Dispositivo DTH22

Integración con Thingsboard

Una vez finalizada toda la configuración (registro de dispositivo, función de decodificador y registro de aplicación), se puede configurar *ThingsBoard*.

Convertidor de datos de enlace ascendente de Thingsboard

Se necesita crear un convertidor de datos de enlace ascendente que se utilizará para recibir mensajes de TTN. El convertidor debe transformar la carga útil entrante en el formato de mensaje requerido. El mensaje debe contener *deviceName* y *deviceType*. Estos campos se utilizan para enviar datos al dispositivo correcto.

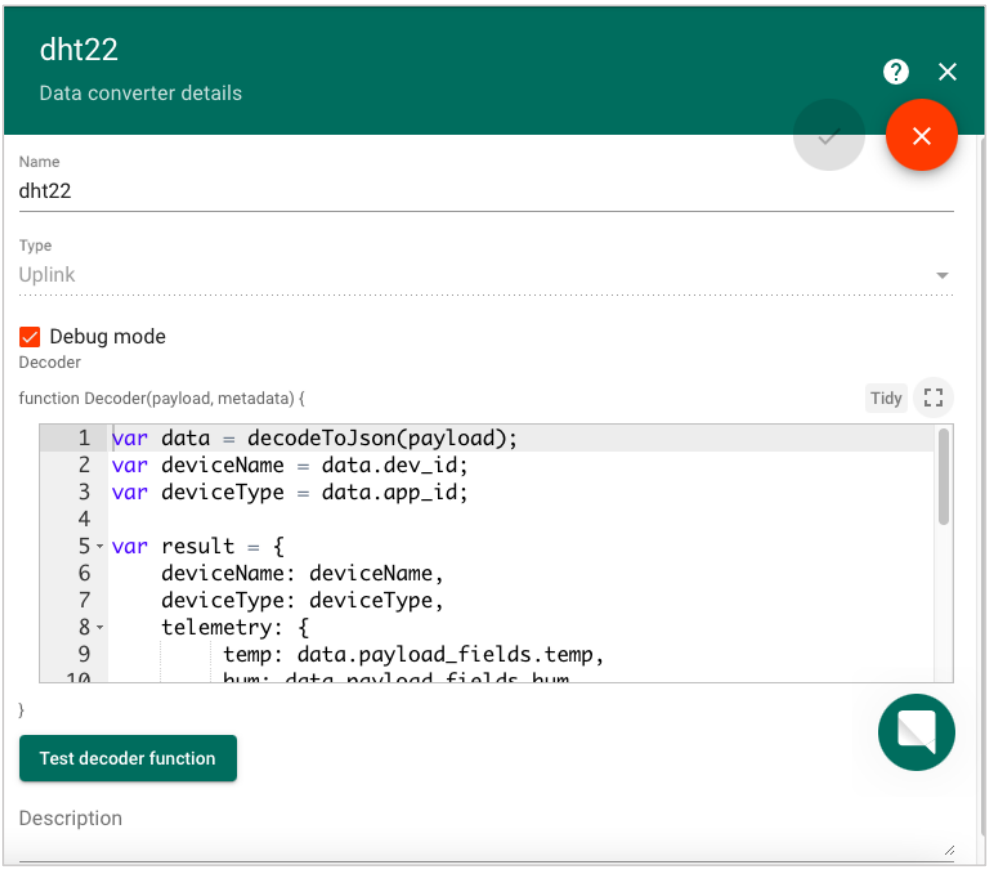

Figura 75. Convertidor de datos de enlace ascendente

Se crea la siguiente función, donde se toman los valores de “temperatura” y “humedad”, con el fin de poder mostrar dichos valores en la telemetría de *ThingsBoard*.

```
var data = decodeToJson(payload);
var deviceName = data.dev_id;
var deviceType = data.app_id;

var result = {
    deviceName: deviceName,
    deviceType: deviceType,
    telemetry: {
         temperature: data.payload_fields.temperature
       hum: data.payload_fields.hum
    }
};

function decodeToString(payload) {
    return String.fromCharCode.apply(String, payload);
}

function decodeToJson(payload) {
    var str = decodeToString(payload);
    var data = JSON.parse(str);
    return data;
}

return result;
```

Figura 76. Función del decoder de ThingsBoard

Integración TTN

A continuación, se crea una Integración con *TheThingsNetwork* dentro del *ThingsBoard* en la sección "Integraciones", siguiendo estos pasos:

- Nombre: ttn_integration.
- Tipo: TheThingsNetwork.
- Convertidor de datos de enlace ascendente: dht22.
- Región: eu.
- ID de aplicación: thingsboard_prueba (utilice el ID de aplicación de TTN).
- Clave de acceso: use la clave de acceso de TTN.

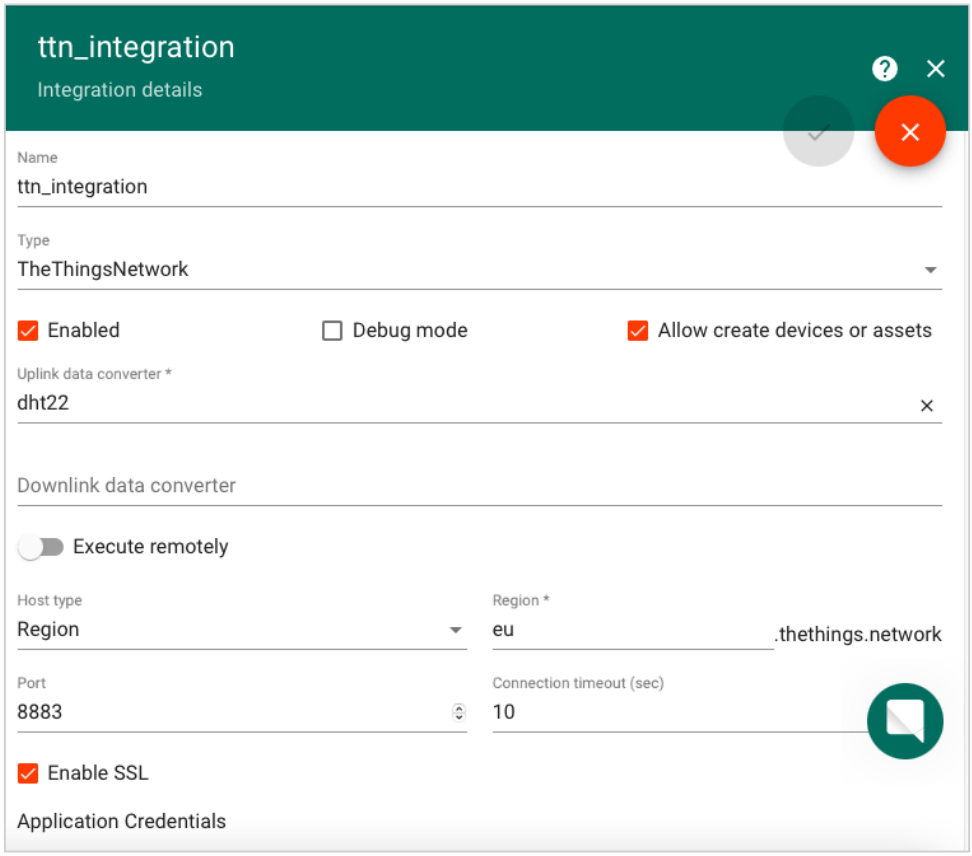

Figura 77. Integración TTN (1)

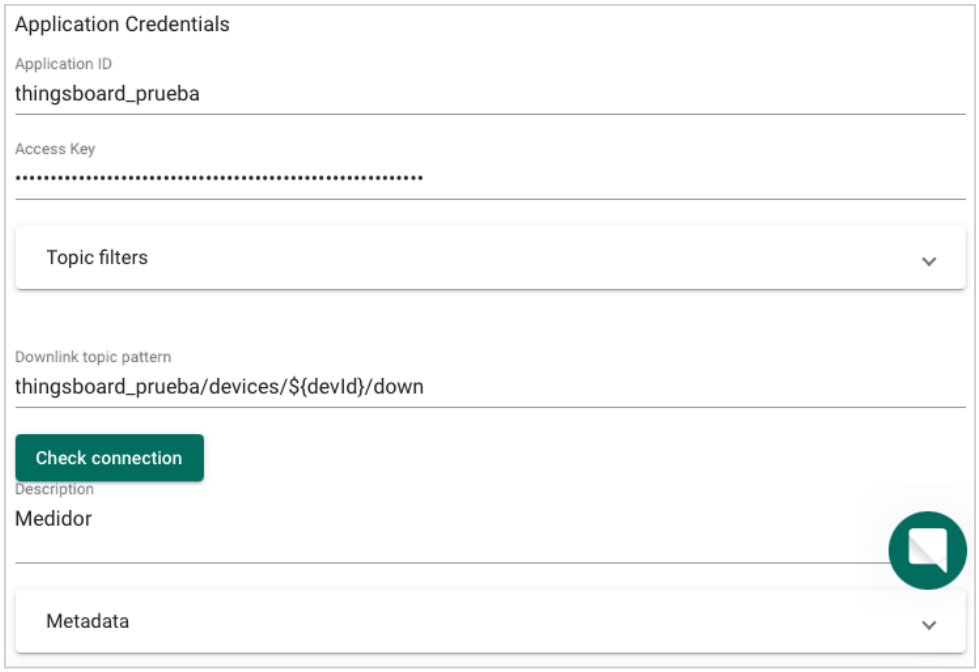

Figura 78. Integración TTN (2)

Por último, se accede a "Grupo de dispositivos -> Todos -> dth22_sensor", donde se puede observar:

- El registro de un nuevo dispositivo en *ThingsBoard*.
- En la sección "Última telemetría", se verá el último valor de temperatura y humedad recibida.

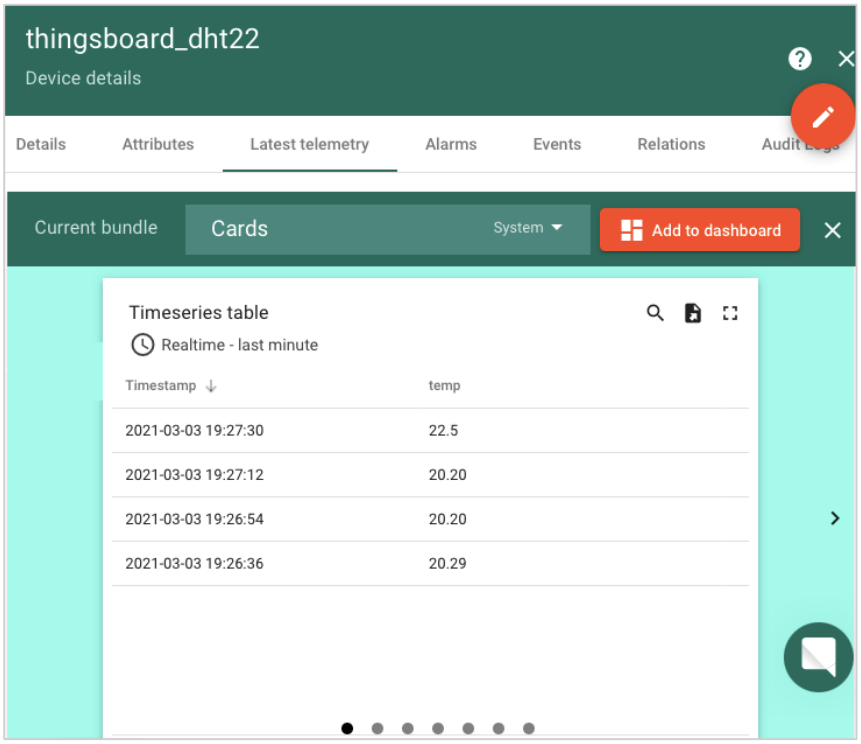

Figura 79. ThingsBoard Temperatura

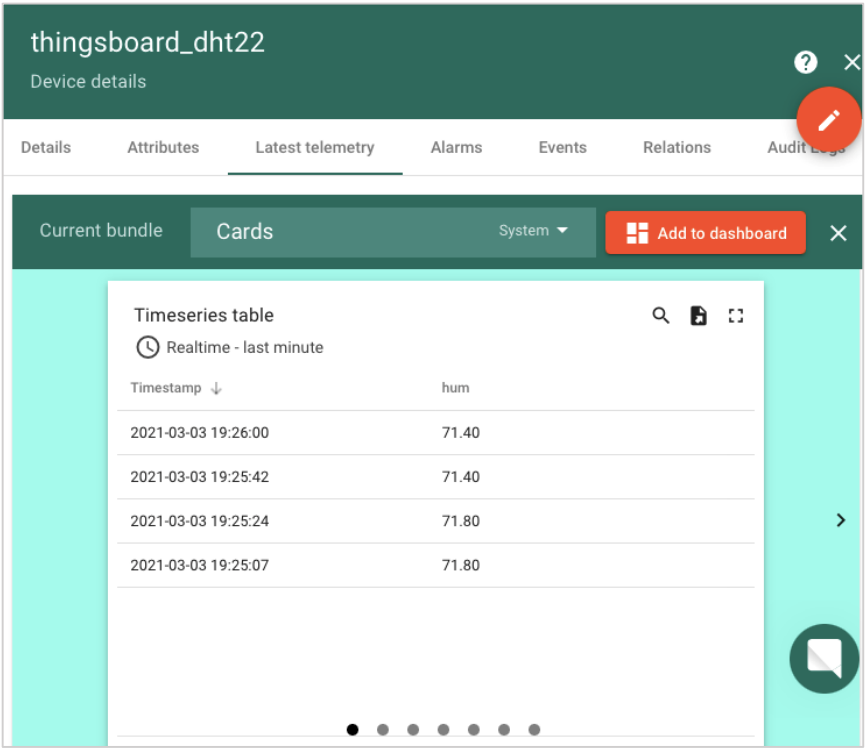

Figura 80. Thinsboard Humedad relativa

Para la visualización se crea este widget.

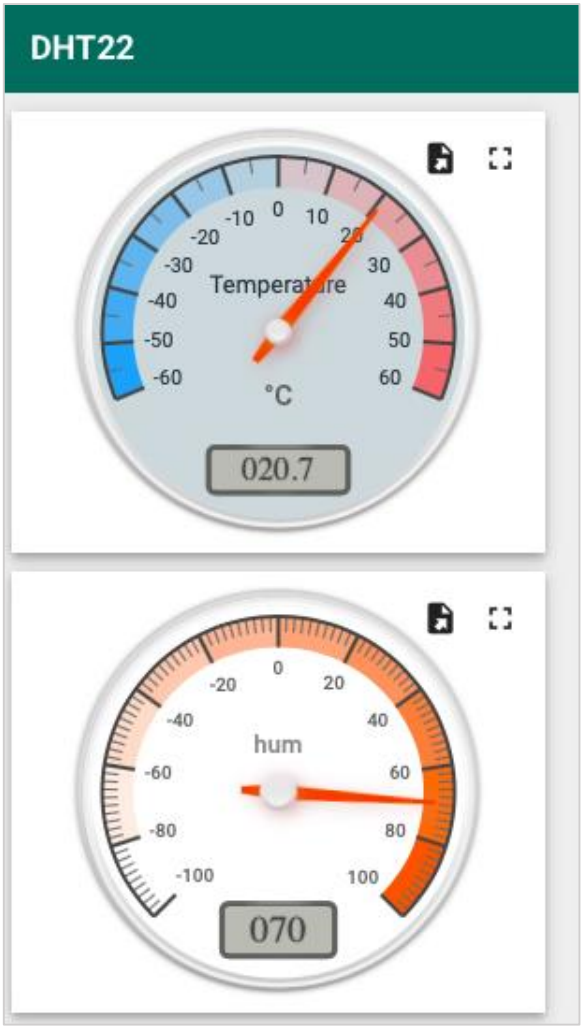

Figura 81. Widget DHT22

6. Presupuesto

En este apartado se realiza un desglose de los costes del proyecto, que engloban hardware, software y mano de obra.

6.1 Coste de Software

El desarrollo del software para este proyecto se ha realizado con los siguientes programas:

- **Arduino IDE 1.8.13**: tiene licencia gratuita.
- **Microsoft Office**: la universidad cuenta con licencia universitaria.

Por tanto, el coste de software total es de cero euros, debido a que la mayoría de ellos tienen licencias gratuitas o suministradas por la Universidad de Alcalá.

6.2 Coste de Hardware

Tabla 3. Coste del material

Descripción	Precio (€/ud)	Cantidad	Coste total
Ordenador Portátil Asus F550C	549 €	1	Duración = 72 meses Uso = 3 meses 27,45€
Heltec Cubecell-GPS	15,80 €	2	31,6 €
Kit de desarrollo LoRaWAN The Things Industries TTIG-868 (868MHZ)	85,28 €	1	85,28 €
CCS811-Sensor de Gas de calidad del aire	7,93 €	2	15,86 €
AZDelivery DHT22 AM2302	8,29 €	1	8,29 €
Detector de sensor UV UVM-30A	1,37 €	11	15,07 €
SPI Reader lector de tarjeta SD	4,29 €	2	8,58 €
Reloj en tiempo real Modulo RTC DS3231	7,49 €	1	7,49 €
Coste Total del Material			199,62 €

6.3 Coste de mano de obra

Tabla 4. Coste de mano de obra

Concepto	Categoría	€/hora	Horas totales	Coste total
Alberto Garmendia Gutiérrez	Ingeniero Junior	40	100	4000 €
Coste Total de mano de obra				4000 €

6.4 Coste de ejecución material

Tabla 5. Coste de ejecución material

Concepto	Coste Total
Coste de material	199,62 €
Coste de mano de obra	4000 €
Total	4199,62 €

6.5 Presupuesto de ejecución por contrata

Tabla 6. Presupuesto de ejecución por contrata

Concepto	Coste Total
Ejecución material	4199,62 €
Gastos generales (17%)	713,93 €
Beneficio industrial (6 %)	251,97 €
Total	5165,52 €

6.6 Presupuesto Total

Tabla 7. Presupuesto Total

Concepto	Coste Total
Ejecución por contrata	5165,52 €
I.V.A. (21 %)	1084,75 €
Presupuesto Total	6250,27 €

7. Bibliografía

[1] Tecnología LoRa.; https://alfaiot.com/tecnologias-iot/lora/

[2] SigFox.; http://productos-iot.com/sigfox-3/

[3] Nb-IoT. ; https://alfaiot.com/vs/nbiot-lorawan-sigfox/

[4] Clases de dispositivos LPWAN. ; https://pandorafms.com/blog/es/que-es-lpwan/#:~:text=comunicaciones%20para%20IoT-,LPWAN%3A%20introducci%C3%B3n%20al%20protocolo%20de%20comunicaciones%20de%20IoT,para%20la%20implementaci%C3%B3n%20de%20IoT.

[5] Clases de LoRaWan.; https://osiberia.org/que-es-una-red-iot-lpwan/

[6] Arquitectura LoRaWan.; https://medium.com/pruebas-de-laboratorio-de-la-modulaci%C3%B3n-lora/lorawan-d00f48384160

[7] LoRaWan Over-The-Air Activation (OTAA). ; https://medium.com/beelan/haciendo-iot-con-lora-capitulo-2-tipos-y-clases-de-nodos-3856aba0e5be

[8] Activation-By-Personalisation (ABP). ; https://medium.com/beelan/haciendo-iot-con-lora-capitulo-2-tipos-y-clases-de-nodos-3856aba0e5be

[9] Heltec Cubecell-GPS.; https://heltec.org/project/htcc-ab02s/

[10] Diagrama de Pines Heltec Cubecell-GPS.; https://resource.heltec.cn/download/CubeCell/HTCC-AB02S/HTCC-AB02S_PinoutDiagram.pdf

[11] Instalación CubeCell Arduino Development Environment a través del administrador de la placa Arduino.; https://heltec-automation-docs.readthedocs.io/en/latest/cubecell/quick_start.html#use-arduino-board-manager

[12] Sensor CO2 CCS811.; https://es.aliexpress.com/item/1005002303050387.html?

[13] Diagrama de bloques CCS811.; https://cdn.sparkfun.com/assets/learn_tutorials/1/4/3/CCS811_Datasheet-DS000459.pdf

[14] Sensor DTH22.; https://www.amazon.es/temperatura-humedad-Arduino-Raspberry-AZDelivery/dp/B078SVZB1X/ref=sr_1_1_sspa?

[15] Sensor Índice UVM30A.; https://www.amazon.es/Sensor-Detecci%C3%B3n-ultravioleta-M%C3%B3dulo-UVM-30A/dp/B01EZZO4I4

[16] Índice UV respecto a los MV de salida.; https://rambal.com/color-luz-forma/192-sensor-ultravioleta-uv-uvm30a.html

[17] Sensor de presión LPS22HB.; https://es.rs-online.com/web/p/kits-de-desarrollo-de-sensores/1683051/?

[18] AZDelivery Reloj en tiempo real Modulo RTC DS3231.; https://www.amazon.es/AZDelivery-RTC-incluida-Raspberry-microcontrolador/dp/B01M2B7HQB/ref=sr_1_1_sspa?

[19] Lector SD.; https://www.amazon.es/AZDelivery-Reader-Tarjeta-Memoria-Arduino/dp/B06X1DX5WS/ref=sr_1_1_sspa?

[20] The Things Network.; https://www.thethingsnetwork.org/

[21] Things Indoor Gateway.; https://www.thethingsnetwork.org/docs/gateways/thethingsindoor/

[22] Agregar aplicación TTN.; https://www.thethingsnetwork.org/docs/aplications/add/

[23] API de Handler de The Things Network.; https://www.thethingsnetwork.org/docs/applications/apis/

[24] SDK.; https://www.thethingsnetwork.org/docs/applications/sdks/

[25] Integraciones The Things Network.; https://www.thethingsnetwork.org/docs/applications/integrations/

[26] ThingSpeak.; https://www.thingspeak.com

[27] Configuración ThingSpeak.; https://descubrearduino.com/thingspeak/

[28] Cayenne Mydevices.; https://developers.mydevices.com/cayenne/features/

[29] Creación de una cuenta MyDevices.; https://accounts.mydevices.com/auth/realms/cayenne/protocol/openid-connect/registrations?

[30] Librería Cayenne LPP.; https://programarfacil.com/blog/arduino-blog/cayenne-mydevices-arduino-sensores-iot/

[31] Payload Cayenne LPP.; https://www.thethingsnetwork.org/docs/devices/arduino/api/cayennelpp/

[32] Plataforma ThingsBoard; https://thingsboard.io/

[33] Diagrama de la arquitectura de ThingsBoard.; https://thingsboard.io/docs/pe/reference/

[34] Agregar una aplicación para la plataforma ThingsBoard.; https://thingsboard.io/docs/user-guide/integrations/ttn/#integration-with-the-thingsboard

8. Anexos

8.1 Anexo 1. Código Índice UV para la plataforma Cayenne myDevices

```
/* Librerias*/

#include "LoRaWan_APP.h"

#include "Arduino.h"

#include <CayenneLPP.h>

#include <GPS_Air530.h>

/* Parametros OTAA */

uint8_t devEui[] = { 0x22, 0x32, 0x33, 0x00, 0x00, 0x88, 0x88, 0x02 };

uint8_t appEui[] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };

uint8_t appKey[] = { 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x66,

0x01 };

/* Parametros ABP */

uint8_t nwkSKey[] = { 0x2B, 0x56, 0x81, 0xD8, 0x72, 0xE2, 0xDD, 0x07, 0xEA, 0xD4, 0x5F, 0xBE, 0x53, 0x37, 0xF

F, 0x47 };

uint8_t appSKey[] = { 0xCF, 0xED, 0xCF, 0x39, 0x99, 0xED, 0xBF, 0x1E, 0xEB, 0xD1, 0x34, 0x05, 0xEB, 0x99, 0x6

3, 0x0C };

uint32_t devAddr =  ( uint32_t )0x26011F31;

/* Canales LoraWan, por defecto de 0-7*/

uint16_t userChannelsMask[6]={ 0x00FF,0x0000,0x0000,0x0000,0x0000,0x0000 };

/* Region LoraWan */

LoRaMacRegion_t loraWanRegion = ACTIVE_REGION;

/* Clase LoraWan */

DeviceClass_t  loraWanClass = LORAWAN_CLASS;

/*Ciclo de Trabajo.  valor en milisegundos.*/

uint32_t appTxDutyCycle = 15000;

/* Modo OTAA o ABP*/

bool overTheAirActivation = LORAWAN_NETMODE;
```

```
/* Habilitar ADR */
bool loraWanAdr = LORAWAN_ADR;
/* Fijar LORAWAN_Net_Reserve en modo ON */
bool keepNet = LORAWAN_NET_RESERVE;
/* Indica si el nodo ha enviado mensajes confirmados o no confirmados  */
bool isTxConfirmed = LORAWAN_UPLINKMODE;
/* Puerto de Aplicación */
uint8_t appPort = 2;
/* Numero de Pruebas */
uint8_t confirmedNbTrials = 4;
/* Definición de variables */
float voltage;
float UVIndex;
float lat,lon,alt;
/* Funcion Preparar la Trama */
static void prepareTxFrame( uint8_t port )
{
 /* Habilitacion Módulo GPS */
 Air530.begin();
 uint32_t starttime = millis();
 /* Adquisición de los valores GPS */
 while( (millis()-starttime) < 1000 )
 {
  while (Air530.available() > 0)
  {
   Air530.encode(Air530.read());
  }
 }
```

```
/* Variables */
 lat=Air530.location.lat();
 lon=Air530.location.lng();
 alt=Air530.altitude.meters();
 /* Inicialiación del protocolo Cayenne  */
 CayenneLPP lpp(LORAWAN_APP_DATA_MAX_SIZE);
 /* Lectura Analógica que devuelve el voltaje en mV. El valor maximo es 2400mV */
 voltage=analogRead(ADC3);
/* Diferentes Rangos del indice UV */
if(voltage<50)
 {
  UVIndex = 0;
 }else if (voltage>50 && voltage<=227)
 {
  UVIndex = 0;
 }else if (voltage>227 && voltage<=318)
 {
  UVIndex = 1;
 }
 else if (voltage>318 && voltage<=408)
 {
  UVIndex = 2;
 }else if (voltage>408 && voltage<=503)
 {
  UVIndex = 3;
 }
 else if (voltage>503 && voltage<=606)
 {
```

```
  UVIndex = 4;
 }else if (voltage>606 && voltage<=696)
 {
  UVIndex = 5;
 }else if (voltage>696 && voltage<=795)
 {
  UVIndex = 6;
 }else if (voltage>795 && voltage<=881)
 {
  UVIndex = 7;
 }
 else if (voltage>881 && voltage<=976)
 {
  UVIndex = 8;
 }
 else if (voltage>976 && voltage<=1079)
 {
  UVIndex = 9;
 }
 else if (voltage>1079 && voltage<=1170)
 {
  UVIndex = 10;
 }else if (voltage>1170)
 {
  UVIndex = 11;
 }
/* Funciones para visualizar los datos en Cayenne */
 lpp.addLuminosity(1,voltage);
```

```
  lpp.addLuminosity(2,UVIndex);
  lpp.addGPS(3, lat,lon,alt);
  lpp.getBuffer(),
  appDataSize = lpp.getSize();
  memcpy(appData,lpp.getBuffer(),appDataSize);
  /* Impresiones por pantalla de los valores obtenidos */
  Serial.println(voltage);
  Serial.println(UVIndex);
  Serial.println(lat);
  Serial.println(lon);
  Serial.println(alt);
}
/* Función principal */
void setup() {
  Serial.begin(115200);
#if(AT_SUPPORT)
  enableAt();
#endif
  deviceState = DEVICE_STATE_INIT;
  LoRaWAN.ifskipjoin();
}
/* Protocolo LoraWan */
void loop()
{
  switch( deviceState )
  {
    case DEVICE_STATE_INIT:
    {
```

```
#if(LORAWAN_DEVEUI_AUTO)
    LoRaWAN.generateDeveuiByChipID();
#endif
#if(AT_SUPPORT)
    getDevParam();
#endif
    printDevParam();
    LoRaWAN.init(loraWanClass,loraWanRegion);
    deviceState = DEVICE_STATE_JOIN;
    break;
  }
  case DEVICE_STATE_JOIN:
  {
    LoRaWAN.join();
    break;
  }
  case DEVICE_STATE_SEND:
  {
    prepareTxFrame( appPort );
    LoRaWAN.send();
    deviceState = DEVICE_STATE_CYCLE;
    break;
  }
  case DEVICE_STATE_CYCLE:
  {
    // Siguente paquete de la transmisión
    txDutyCycleTime = appTxDutyCycle + randr( 0, APP_TX_DUTYCYCLE_RND );
    LoRaWAN.cycle(txDutyCycleTime);
```

```
    deviceState = DEVICE_STATE_SLEEP;
    break;
   }
   case DEVICE_STATE_SLEEP:
   {
    LoRaWAN.sleep();
    break;
   }
   default:
   {
    deviceState = DEVICE_STATE_INIT;
    break;
   }
  }
}
```

8.2 Anexo 2. Código Sensor CCS811 para la plataforma ThingSpeak

```
/* Librerias*/

#include "LoRaWan_APP.h"

#include "Arduino.h"

#include <Wire.h>

#include <CCS811.h>

/* Parametros OTAA */

uint8_t devEui[] = { 0x22, 0x32, 0x33, 0x00, 0x00, 0x88, 0x88, 0x02 };

uint8_t appEui[] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };

uint8_t appKey[] = { 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x66,

0x01 };

/* Parametros ABP */

uint8_t nwkSKey[] = { 0xB7, 0x1E, 0xF4, 0x9C, 0xB8, 0x15, 0xCE, 0x15, 0x21, 0xAA, 0x15, 0x80, 0xF0, 0x3E, 0xD

1, 0xB1 };

uint8_t appSKey[] = { 0x47, 0xA0, 0xE7, 0x61, 0x71, 0xEE, 0x51, 0x31, 0x45, 0xAF, 0x54, 0x54, 0xC8, 0x6C, 0x05

, 0xB9 };

uint32_t devAddr =  ( uint32_t )0x260138AA;

/* Canales LoraWan, por defecto de 0-7*/

uint16_t userChannelsMask[6]={ 0x00FF,0x0000,0x0000,0x0000,0x0000,0x0000 };

/* Region LoraWan */

LoRaMacRegion_t loraWanRegion = ACTIVE_REGION;

/* Clase LoraWan */

DeviceClass_t  loraWanClass = LORAWAN_CLASS;

/*Ciclo de Trabajo.  valor en milisegundos.*/

uint32_t appTxDutyCycle = 15000;

/* Modo OTAA o ABP*/

bool overTheAirActivation = LORAWAN_NETMODE;
```

```
/* Habilitar ADR */
bool loraWanAdr = LORAWAN_ADR;
/* Fijar LORAWAN_Net_Reserve en modo ON */
bool keepNet = LORAWAN_NET_RESERVE;
/* Indica si el nodo ha enviado mensajes confirmados o no confirmados */
bool isTxConfirmed = LORAWAN_UPLINKMODE;
/* Puerto de Aplicación */
uint8_t appPort = 2;
/* Numero de Pruebas */
uint8_t confirmedNbTrials = 4;
/* Objeto ccs de la libreria CCS811 */
CCS811 ccs;
static void prepareTxFrame( uint8_t port )
{
   /* Fijar modo Entrada o Salida y asignar el Pin */
   pinMode(Vext, OUTPUT);
   digitalWrite(Vext, LOW);
   pinMode(GPIO3,OUTPUT);
   digitalWrite(GPIO3,LOW);
   pinMode(GPIO4,OUTPUT);
   digitalWrite(GPIO4,HIGH);
/* Variables a medir */
   float co2=0;
   float tvoc=0;
   float i=0;
   ccs.begin();
   while(!ccs.available());      /* ccs.available devuelve true si los datos estan disponibles para la lectura*/
```

```
 delay(5000);
  /* Extrae el valor medio de 5 muestras */
  while(i<5)
  {
     ccs.available();
     {
        if(!ccs.readData())    /* Devuelve True si ocurre un error durante la lectura */
        {
            co2 += ccs.geteCO2();  /* Devuelve el CO2 */
         tvoc += ccs.getTVOC();  /* Devuelve la lectura de TVOC */
           i++;
        }
     }
  }
  co2/=i;
  tvoc/=i;
  Wire.end();
  digitalWrite(Vext, HIGH);
  digitalWrite(GPIO3,HIGH);
  digitalWrite(GPIO4,LOW);
/* Formación del Payload para decodificar en TTN */
  unsigned char *puc;
puc = (unsigned char *)(&co2);
appDataSize = 8;
appData[0] = puc[0];
appData[1] = puc[1];
appData[2] = puc[2];
appData[3] = puc[3];
```

```
  puc = (unsigned char *)(&tvoc);
  appData[4] = puc[0];
  appData[5] = puc[1];
  appData[6] = puc[2];
  appData[7] = puc[3];
   /* impresion de Pantallas */
   Serial.print("CO2: ");
   Serial.print(co2);
   Serial.print(" ppm,");
   Serial.print(" TVOC: ");
   Serial.print(tvoc);
   Serial.print(" ppb,");
}
/* Función principal */
void setup() {
   boardInitMcu();
   Serial.begin(115200);
#if(AT_SUPPORT)
   enableAt();
#endif
   deviceState = DEVICE_STATE_INIT;
   LoRaWAN.ifskipjoin();
}
/* Protocolo LoraWan */
void loop()
{
   switch( deviceState )
   {
```

```
      case DEVICE_STATE_INIT:
      {
#if(LORAWAN_DEVEUI_AUTO)
        LoRaWAN.generateDeveuiByChipID();
#endif
#if(AT_SUPPORT)
        getDevParam();
#endif
        printDevParam();
        LoRaWAN.init(loraWanClass,loraWanRegion);
        deviceState = DEVICE_STATE_JOIN;
        break;
      }
      case DEVICE_STATE_JOIN:
      {
        LoRaWAN.join();
        break;
      }
      case DEVICE_STATE_SEND:
      {
        prepareTxFrame( appPort );
        LoRaWAN.send();
        deviceState = DEVICE_STATE_CYCLE;
        break;
      }
      case DEVICE_STATE_CYCLE:
      {
        // Siguente paquete de la transmisión
```

```
        txDutyCycleTime = appTxDutyCycle + randr( 0, APP_TX_DUTYCYCLE_RND );
        LoRaWAN.cycle(txDutyCycleTime);
        deviceState = DEVICE_STATE_SLEEP;
        break;
      }
      case DEVICE_STATE_SLEEP:
      {
        LoRaWAN.sleep();
        break;
      }
      default:
      {
        deviceState = DEVICE_STATE_INIT;
        break;
      }
    }
}
```

8.3 Anexo 3. Código del sensor DHT22 para la plataforma Thingsboard

```
/* Librerias*/

#include "LoRaWan_APP.h"

#include "Arduino.h"

#include <DHT.h>;

//Definición del Objeto DHT22 y Asignación del pin out

#define DHTTYPE DHT22   // DHT 22  (AM2302)

DHT dht(GPIO3, DHTTYPE); //// Inicializar el PIN GPIO3

/* Parametros OTAA */

uint8_t devEui[] = { 0x22, 0x32, 0x33, 0x00, 0x00, 0x88, 0x88, 0x02 };

uint8_t appEui[] = { 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 };

uint8_t appKey[] = { 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x66,

0x01 };

/* Parametros ABP */

uint8_t nwkSKey[] = { 0xDC, 0x19, 0x27, 0x88, 0x76, 0x16, 0xFA, 0xA5, 0xA9, 0x55, 0x9A, 0xA0, 0x21, 0x92, 0xC

2, 0x2F };

uint8_t appSKey[] = { 0x70, 0x8A, 0x5D, 0x8D, 0x1E, 0xA9, 0xED, 0x83, 0x8F, 0xE1, 0xE5, 0x73, 0x29, 0xDF, 0x8

5, 0xC4 };

uint32_t devAddr =  ( uint32_t )0x26011107;

/* Canales LoraWan, por defecto de 0-7*/

uint16_t userChannelsMask[6]={ 0x00FF,0x0000,0x0000,0x0000,0x0000,0x0000 };

/* Region LoraWan */

LoRaMacRegion_t loraWanRegion = ACTIVE_REGION;

/* Clase LoraWan */

DeviceClass_t  loraWanClass = LORAWAN_CLASS;

/*Ciclo de Trabajo.  valor en milisegundos.*/

uint32_t appTxDutyCycle = 15000;
```

```
/* Modo OTAA o ABP*/
bool overTheAirActivation = LORAWAN_NETMODE;
/* Habilitar ADR */
bool loraWanAdr = LORAWAN_ADR;
/* Fijar LORAWAN_Net_Reserve en modo ON */
bool keepNet = LORAWAN_NET_RESERVE;
/* Indica si el nodo ha enviado mensajes confirmados o no confirmados  */
bool isTxConfirmed = LORAWAN_UPLINKMODE;
/* Puerto de Aplicación */
uint8_t appPort = 2;
/* Numero de Pruebas */
uint8_t confirmedNbTrials = 4;
static void prepareTxFrame( uint8_t port )
{
  /* Variables a medir */
float hum;
float temp;
 dht.begin();
 hum = dht.readHumidity();
  temp= dht.readTemperature();
/* Formación del Payload para decodificar en TTN */
  unsigned char *puc;
  puc = (unsigned char *)(&hum);
  appDataSize = 8;
  appData[0] = puc[0];
  appData[1] = puc[1];
  appData[2] = puc[2];
  appData[3] = puc[3];
```

```
  puc = (unsigned char *)(&temp);
  appData[4] = puc[0];
  appData[5] = puc[1];
  appData[6] = puc[2];
  appData[7] = puc[3];
  /* Impresion de Pantalla */
    Serial.print("Humedad: ");
    Serial.print(hum);
    Serial.print(" %, Temperatura: ");
    Serial.print(temp);
    Serial.println(" Grados");
    delay(2000);
}
/* Función principal */
void setup() {
boardInitMcu();
  Serial.begin(115200);
#if(AT_SUPPORT)
  enableAt();
#endif
  deviceState = DEVICE_STATE_INIT;
  LoRaWAN.ifskipjoin();
}
/* Protocolo LoraWan */
void loop() {
  switch( deviceState )
  {
    case DEVICE_STATE_INIT:
```

```
  {
#if(LORAWAN_DEVEUI_AUTO)
    LoRaWAN.generateDeveuiByChipID();
#endif
#if(AT_SUPPORT)
    getDevParam();
#endif
    printDevParam();
    LoRaWAN.init(loraWanClass,loraWanRegion);
    deviceState = DEVICE_STATE_JOIN;
    break;
  }
  case DEVICE_STATE_JOIN:
  {
    LoRaWAN.join();
    break;
  }
  case DEVICE_STATE_SEND:
  {
    prepareTxFrame( appPort );
    LoRaWAN.send();
    deviceState = DEVICE_STATE_CYCLE;
    break;
  }
  case DEVICE_STATE_CYCLE:
  {
    // Siguente paquete de la transmisión
    txDutyCycleTime = appTxDutyCycle + randr( 0, APP_TX_DUTYCYCLE_RND );
```

```
    LoRaWAN.cycle(txDutyCycleTime);
    deviceState = DEVICE_STATE_SLEEP;
    break;
   }
   case DEVICE_STATE_SLEEP:
   {
    LoRaWAN.sleep();
    break;
   }
   default:
   {
    deviceState = DEVICE_STATE_INIT;
    break;
   }
  }
}
```

8.4 Anexo 4. Código para obtener la fecha actual a través del RTC DS3231

```
/* Librerias*/
#include <TimeLib.h>
#include <Wire.h>
#include <DS3231.h>
/* Funcion Setup */
void setup()  {
 Serial.begin(9600);
 while (!Serial) ;
 /* Sincronizar el tiempo con el RTC */
setSyncProvider(RTC.get);
 if(timeStatus()!= timeSet)
    Serial.println("No ha sido posible la sincronización con el RTC");
 else
    Serial.println("RTC ha sido sincronizado");
}
/* Función Loop */
void loop()
{
 if (timeStatus() == timeSet) {
   mostrartiempodigital();
 } else {
   Serial.println("No se ha podido sincronizar el tiempo”);
   Serial.println();
   delay(4000);
 }
 delay(1000);
}
/* Imprime el tiempo por pantalla */
```

```
void mostrartiempodigital(){
  Serial.print(hour());
  imprimirDigitos(minute());
  imprimirDigitos(second());
  Serial.print(" ");
  Serial.print(day());
  Serial.print(" ");
  Serial.print(month());
  Serial.print(" ");
  Serial.print(year());
  Serial.println();
}
/* Asigna un 0 delante si el numero es menor que 10 y separa los digitos con ":" */
void imprimirDigitos(int digitos){
  Serial.print(":");
  if(digitos < 10)
    Serial.print('0');
  Serial.print(digitos);
}
```

8.5 Anexo 5. Código para obtener la fecha actual a través del RTC (DS3231) y la presión atmosférica del LPS22HB

```
/* Librerias*/
#include <TimeLib.h>
#include<Wire.h>
#include <DS3231.h>
/* Asignar la dirección de I2C 0x5C(92)*/
#define Addr 0x5C
void setup()
{
 /* Iniciar la comunicación I2C */
 Wire1.begin();
 Serial.begin(9600);
 /* Sincronizar el tiempo con el RTC */
 setSyncProvider(RTC.get);
 if (timeStatus() != timeSet)
   Serial.println("No ha sido posible la sincronización con el RTC");
 else
   Serial.println("RTC ha sido sincronizado");
 /* Iniciar la transmisión */
 Wire1.beginTransmission(Addr);
 /* Seleccion del primer registro de control */
 Wire1.write(0x20);
 /* Fijar el modo activo y continuamente actualizarse */
 Wire1.write(0x90);
 /* Terminar la transmisión*/
 Wire1.endTransmission();
 delay(300);
}
```

```
/* Funcion LOOP */
void loop()
{
 unsigned int data[3];
 /* Empezar la transmisión */
 Wire1.beginTransmission(Addr);
 /* Seleccionar el registro de presión */
 Wire1.write(0x28);
 /* Terminar la transmisión*/
 Wire1.endTransmission();
 /* Maestro solicita 3 bytes al un dispositivo esclavo */
 Wire1.requestFrom(Addr, 3);
 /* Leer 3 bytes , Primer lsb presión*/
 if(Wire1.available() == 3)
 {
  data[0] = Wire1.read();
  data[1] = Wire1.read();
  data[2] = Wire1.read();
 }
 delay(300);
 /* Convertir la presion*/
 float presion = ((data[2] * 65536) + (data[1] * 256) + data[0]) / 4096.0;
 /* Impresion por Pantalla*/
 Serial.print("Presion es: ");
 Serial.print(presion);
 Serial.println(" hPa");
 delay(2000);
 /* Establecer el tiempo del RTC y el sistema al valor recibido*/
```

```
  if (Serial.available()) {
    time_t t = processSyncMessage();
    if (t != 0) {
      RTC.set(t);
      setTime(t);  }
  }
  mostrartiempodigital();
  delay(1000);
}
void mostrartiempodigital(){
  Serial.print(hour());
  imprimirDigitos(minute());
  imprimirDigitos(second());
  Serial.print(" ");
  Serial.print(day());
  Serial.print(" ");
  Serial.print(month());
  Serial.print(" ");
  Serial.print(year());
  Serial.println();
}
void imprimirDigitos(int digitos){
  /* Asigna un 0 delante si el numero es menor que 10 y separa los digitos con ":" */
  Serial.print(":");
  if(digitos < 10)
  Serial.print('0');
  Serial.print(digitos);}
```

8.6 Anexo 6. Código para obtener la fecha actual a través del RTC (DS3231), la presión atmosférica del LPS22HB y almacenar los datos en el Lector SD.

```
/* Librerias*/
#include <SD.h>
#include<Wire.h>
#include <TimeLib.h>
#include <DS3231.h>
/* Asignar la dirección de I2C 0x5C(92)*/
#define Addr 0x5C
/* Creación del Objeto dataFile de tipo FILE*/
File dataFile;
void setup(){
 Wire.begin();
 Serial.begin(9600);
setSyncProvider(RTC.get);
 if(timeStatus()!= timeSet)
   Serial.println("No ha sido posible la sincronización con el RTC");
 else
   Serial.println("RTC ha sido sincronizado");
 /* Iniciar la transmisión */
 Wire.beginTransmission(Addr);
 /* Seleccion del primer registro de control 1 */
 Wire.write(0x20);
 /* Fijar el modo activo y continuamente actualizarse */
 Wire.write(0x90);
 /* Terminar la transmisión*/
 Wire.endTransmission();
 delay(300);
 Serial.print("Iniciando SD card...");
```

```
  digitalWrite(GPIO2, HIGH);
  /* Iniciar la comunicación de la SD*/
  if (!SD.begin(GPIO2)) {
  Serial.println("Fallo comunicacion o no existe SD");
  return;}
  Serial.println("SD Iniciada.");
  dataFile = SD.open("LPS22HB.txt", FILE_WRITE);
/* Impresion por Pantalla*/
  if (dataFile) {
    Serial.println("<<< LPS22HB >>>");
    dataFile.println("<<< LPS22HB >>>");
    dataFile.close();}
    else {
    Serial.println("No se pudo abrir");}
}
void loop(){
if (timeStatus() == timeSet) {
    mostrartiempodigital();
  } else {
    Serial.println("El tiempo no ha sido sincronizado");
    delay(5000);
  }
  delay(1500);
  unsigned int data[3];
  /* Empezar la transmisión */
  Wire.beginTransmission(Addr);
  /* Seleccionar el registro de presion */
  Wire.write(0x28);
```

```
  /* Terminar la transmisión*/
  Wire.endTransmission();
   /* Maestro solicita 3 bytes al dispositivo esclavo*/
  Wire.requestFrom(Addr, 3);
  /* Leer 3 bytes, Primer lsb presión*/
  if(Wire.available() == 3)
  {
    data[0] = Wire.read();
    data[1] = Wire.read();
    data[2] = Wire.read();
  }
  delay(300);
  /* Escritura del archivo LPS22HB.txt*/
dataFile = SD.open("LPS22HB.txt", FILE_WRITE);
/* Escritura de la presion en el documento del texto*/
  if (dataFile) {
    float presion = ((data[2] * 65536) + (data[1] * 256) + data[0]) / 4096.0;
    dataFile.print("Presion : ");
    dataFile.print(presion);
    dataFile.print(" hPa");
        dataFile.close();
  }
  else {
    Serial.println("Fallo en la comunicacion");
    delay(2000); }
}
void mostrartiempodigital(){
  Serial.print(hour());
```

```
  imprimirdigitos(minute());
  imprimirdigitos(second());
  Serial.print(" ");
  Serial.print(day());
  Serial.print(" ");
  Serial.print(month());
  Serial.print(" ");
  Serial.print(year());
  Serial.println();
}
void imprimirdigitos(int digitos){
  Serial.print(":");
  if(digitos < 10)
    Serial.print('0');
  Serial.print(digitos);
}
```

Universidad de Alcalá

Escuela Politécnica Superior

ESCUELA POLITECNICA SUPERIOR