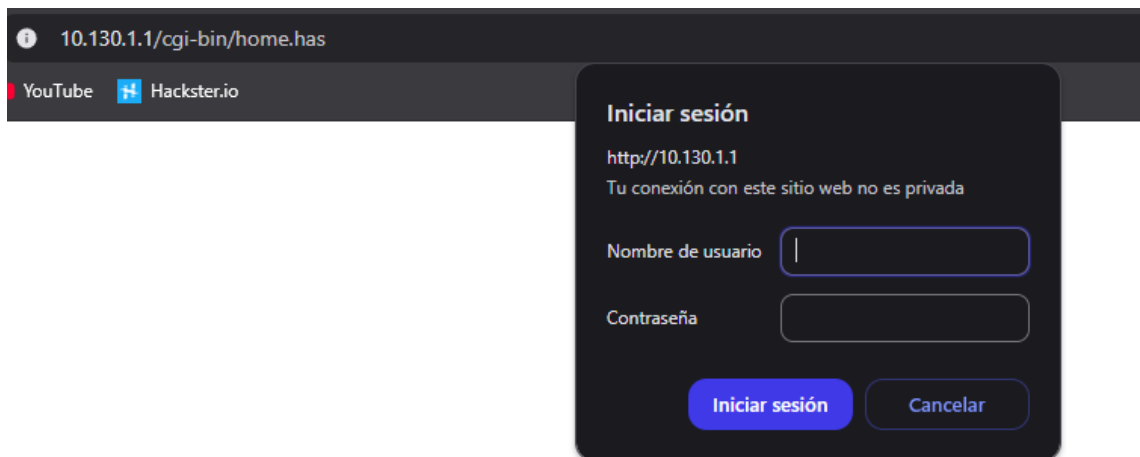


CONFIGURACION GATEWAY DRAGINO LPS8v2

1. Conectamos el Gateway, y accedemos a la red inalámbrica que crea por defecto, con la pass “dragino+dragino”.
2. Accedemos por un navegador a la configuración, accediendo a la IP del Gateway, por defecto suele ser: 10.130.1.1



Accedemos con las credenciales:

- Nombre de usuario: “root”
- Contraseña: dragino

3. Configuramos el acceso a una red que salga a internet:

WiFi Settings

WiFi AP Settings

Enable WiFi Access Point ☒

Wi-Fi Name SSID

Passphrase (8-32 char) [Show](#) Encryption

[Save&Apply AP](#) [Cancel](#)

WiFi WAN Client Settings

Enable WiFi WAN Client ☒

Host WiFi SSID WiFi Survey

Passphrase [Show](#) Proto Type Encryption

[Save&Apply STA](#) [Cancel](#)

Debemos configurar en la parte de abajo a la red que queremos que se conecte.

Seleccionamos LoRaWAN como modo de operación.

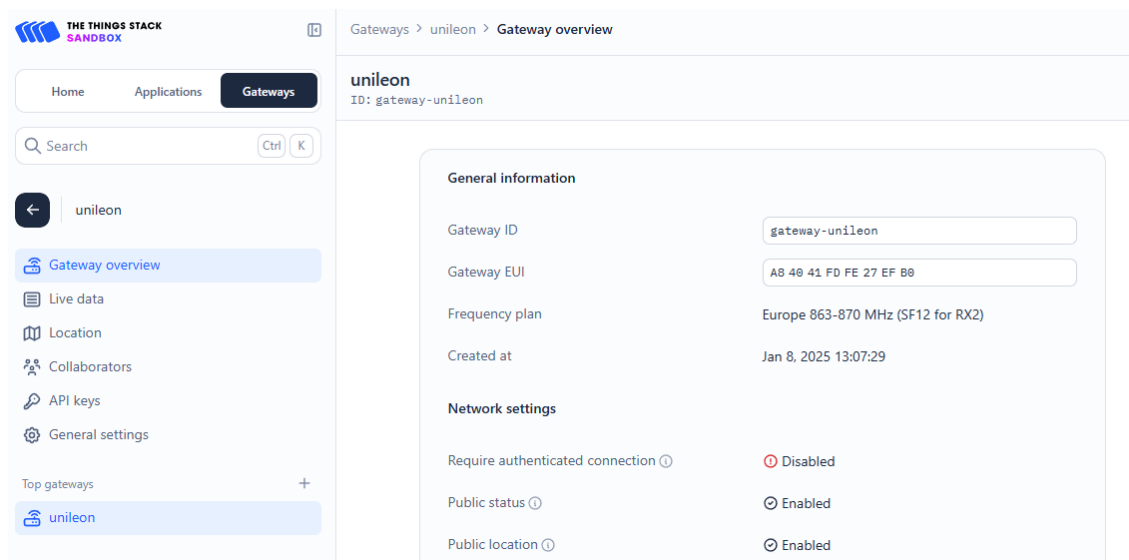
Una vez tengamos así la configuración del Gateway, vamos a TheThingsNetwork, nos creamos una cuenta, y le damos a añadir un Gateway, nos pedirá el EUI del mismo, podemos obtenerlo normalmente en la parte posterior del propio dispositivo, este en concreto tiene un QR en la parte inferior, que al escanearlo te proporciona este número.

Le damos un nombre y seleccionamos una frecuencia, en mi caso (España);

Europe 863-870 MHz (SF9 for RX2 - recommended)

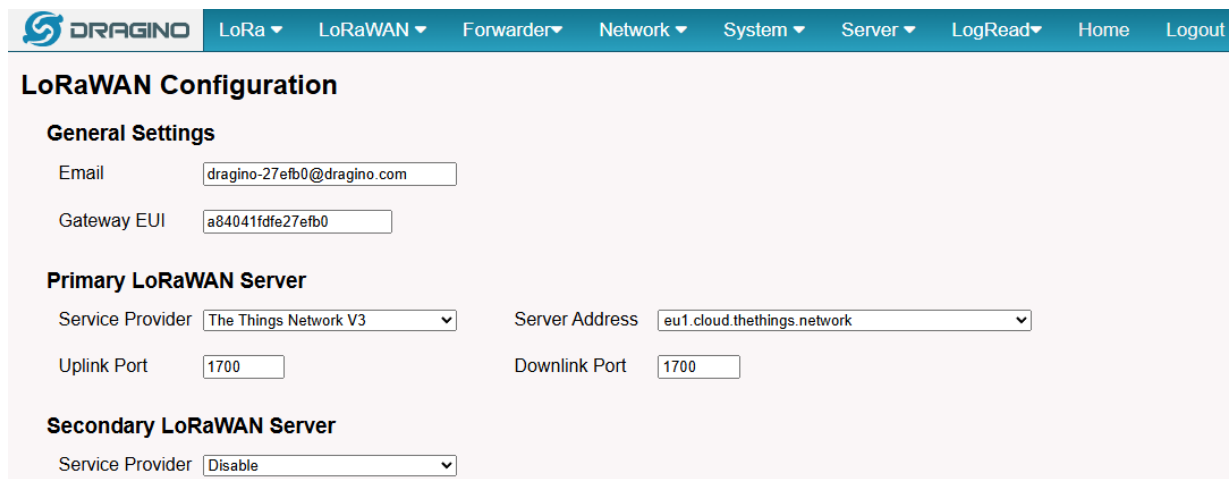
Una vez tengamos así la configuración del Gateway, vamos a TheThingsNetwork y nos creamos una cuenta

4. Nos registramos en TheThingsNetwork, para añadir nuestro Gateway: Una vez registrado el dispositivo, veremos algo como esto.



Donde podemos ver los datos de nuestro Gateway.

Ahora configuramos el LoRaWAN en el dragino con los datos de TTN.



Una vez configurado todo deberíamos ver algo como esto en la pantalla principal:



Ya tenemos configurado el Gateway, lo siguiente será crear una aplicación en TTN para poder añadir los dispositivos que deseamos conectar.

5. Entramos a la sección applications de TTN, y le damos a Add Application:



Le damos el nombre y el ID que deseemos y le damos a create application, una vez creada vamos a añadir dispositivos a la misma:

Create application

Within applications, you can register and manage end devices and their network
Learn more in our guide on [Adding Applications](#).

Application ID*

Application name

Description

Optional application description; can also be used to save notes about the applic

Create application

6. Dentro de la aplicación creada, vamos a entrar en “End devices”, y le daremos a derecha, al botón de “Register end device”:

+ Register end device

Rellenamos los campos, en mi caso para un cubecell HTCC-AB02:

Register end device

Does your end device have a LoRaWAN® Device Identification QR Code? Scan it to speed up onboarding.


 Scan end device QR code

 [Device registration help](#)

End device type

Input method 


- ☒ Select the end device in the LoRaWAN Device Repository
- ☐ Enter end device specifics manually

End device brand  *


HelTec AutoMati... | v

Model  *

HTCC-AB02(Class... | v

Hardware Ver.  *

Unknown... | v

Firmware Ver.  *

1.0 | v

Profile (Region) *

EU_863_870 | v


HTCC-AB02(Class A OTAA)

LoRaWAN Specification 1.0.2, RP001 Regional Parameters 1.0.2 revision B, Over the air activation (OTAA), Class A



The Heltec HTCC-AB02 CubeCell Dev-Board Plus is a LoRaWAN® development board based on ASR605x (ASR6501, ASR6502). The ASR605x chip is integrated with the PSoC® 4000 series MCU (ARM® Cortex® M0+ Core) and SX1262. HTCC-AB02 allows connecting various sensors and supports the Arduino® development environment.

[Product website](#) | [Data sheet](#)

Frequency plan  *

Europe 863-870 MHz (SF9 for RX2 - recommended) | v

Para el JoinEUI, podemos crearlo nosotros mismos, siempre que cumpla con las siguientes reglas:

- 8 bytes (16 caracteres hexadecimales)
- Lo ideal es que empiece por 70B3D57E, ya que es el prefijo común de The Things Network

Una vez este está puesto nos aparecerán mas campos, generamos un DevUI, una AppKey y le damos un ID:

Provisioning information

JoinEUI ? *

70 B3 D5 7E D0 05 7A C3

Reset

This end device can be registered on the network

DevEUI ? *

70 B3 D5 7E D0 06 FD 68

Generate

3/50 used

AppKey ? *

B5 E2 56 EF E0 DD 05 7F 35 06 6F E4 FB 0C 31 A2

Generate

End device ID ? *

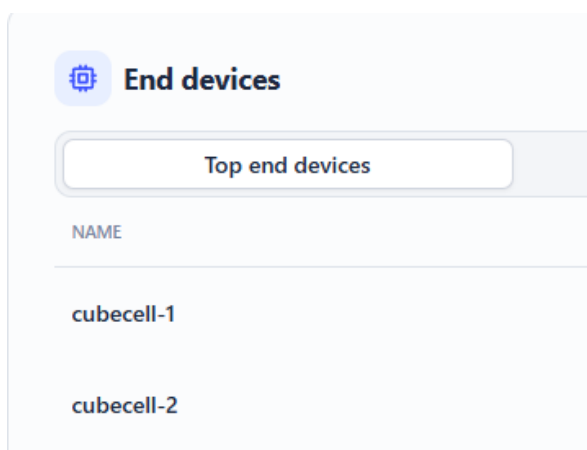
cubecell-1

After registration

- ☒ View registered end device
- ☐ Register another end device of this type

Register end device

Y repetiremos el mismo proceso, generando un distinto JoinEUI, para añadir el segundo dispositivo cubecell.



Nos quedará algo como esto.

7. Ahora vamos a subir el código a los cubecell, necesario para conectarse por OTAA al Gateway, vamos a ArduinoIDE, abriendo el código proporcionado en el repositorio.

Lo primero de todo, antes de empezar, debemos tener instaladas las dependencias:

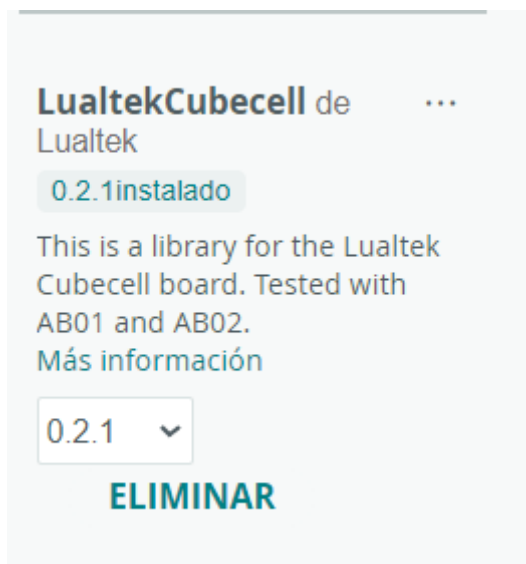
Archivo → Preferencias → URLs ... → y pegamos la siguiente URL:

https://github.com/HelTecAutomation/CubeCell-Arduino/releases/download/V1.5.0/package_CubeCell_index.json

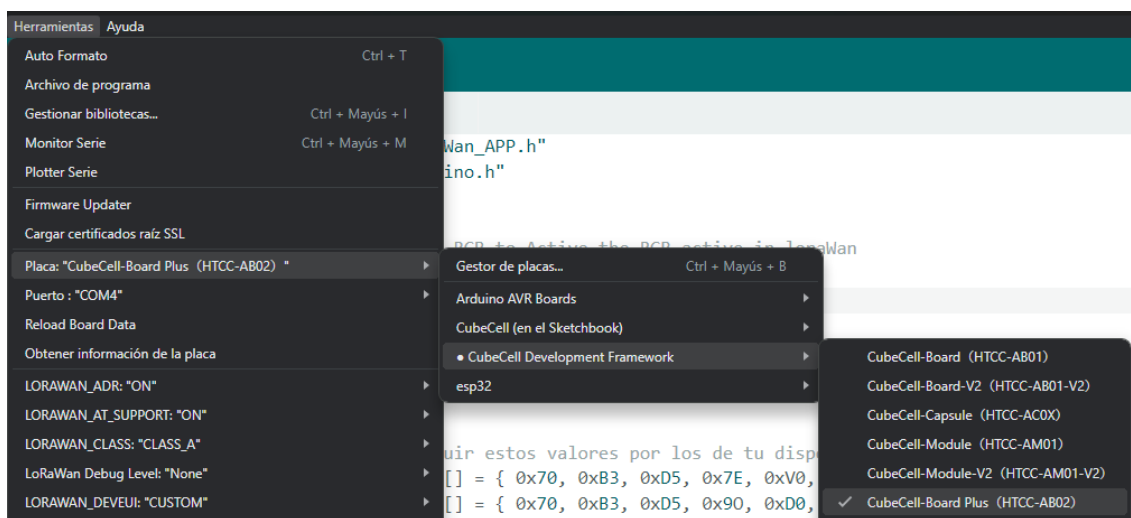
Ha de estar en una línea, no a continuación de otra URL.

Cuando hayamos puesto esto vamos a Herramientas → Gestionar bibliotecas:

Y buscamos e instalamos el siguiente:



Ya tenemos las dependencias, ahora para continuar solo debemos configurar el puerto COM al que tenemos conectado el cubecell y la placa:



Ahora, una vez dentro del archivo, debemos cambiar estas líneas:

```
/* OTAA sustituir estos valores por los de tu dispositivo*/
uint8_t devEui[] = { 0x70, 0xB3, 0xD5, 0x7E, 0xV0, 0x06, 0xFD, 0x09 };
uint8_t appEui[] = { 0x70, 0xB3, 0xD5, 0x90, 0xD0, 0x07, 0x7A, 0xC2 };
uint8_t appKey[] = { 0x16, 0x07, 0x02, 0x68, 0xFE, 0x62, 0x5C, 0x86, 0x6B, 0xAB, 0x9C, 0x06, 0xA8, 0x01, 0xAk, 0xED };
```

Por estos valores de tu dispositivo (se pueden ver en TTN):



Activation information

AppEUI	0x70, 0xB3, 0xD5, 0x7E, 0xD0, 0x... msb ↔
DevEUI	0x70, 0xB3, 0xD5, 0x7E, 0xD0, 0x... msb ↔
AppKey	0xAC, 0x5F, 0x87, 0x... msb ↔ <code></></code>  

Para verlos directamente en hexadecimal, le damos al icono de “</>” y luego copiamos y pegamos en el archivo .ino, sustituyendo los valores del archivo por los tuyos.

Asegúrate de tener esta línea así, ya que por defecto en los ejemplos empieza por “0x00FF”, y TTN utiliza el canal “0xFF00”.

```
/*LoraWan channelmask, default channels 0-7*/
uint16_t userChannelsMask[6]={ 0xFF00,0x0000,0x0000,0x0000,0x0000,0x0000 };
```

Una vez tengamos el archivo modificado con los datos de nuestro dispositivo, podemos compilarlo y subirlo, no debería darnos errores si hemos seguido todo el proceso.

Con esto subido veremos como cada 15 segundos se enciende el led del cubecell, lo que quiere decir que está enviando datos, en este caso:

```
/* Prepares the payload of the frame */
static void prepareTxFrame( uint8_t port )
{
    /*appData size is LORAWAN_APP_DATA_MAX_S:
    *appDataSize max value is LORAWAN_APP_DA
    *if enabled AT, don't modify LORAWAN_APP
    *if disabled AT, LORAWAN_APP_DATA_MAX_SI
    *for example, if use REGION_CN470,
    *the max value for different DR can be fr
    */
    appDataSize = 5;
    appData[0] = 0x00;
    appData[1] = 0x01;
    appData[2] = 0x02;
    appData[3] = 0x03;
    appData[4] = 0x04;
}
```

Estos son los datos que se van a enviar, ahora vamos a TTN, a comprobar que los está enviando:

Entramos en la aplicación, en end devices, y vamos al cubecell que hemos configurado, y veremos cómo está enviando los datos:

↑ 09:36:11	Forward uplink data message	DevAddr:	26 0B 30 C3	00 01 02 03 04	FPort: 2	Data rate: SF8BW125	SNR: 10.5	RSSI: -23
↑ 09:36:01	Forward uplink data message	DevAddr:	26 0B 30 C3	00 01 02 03 04	FPort: 2	Data rate: SF8BW125	SNR: 10.2	RSSI: -25
↑ 09:35:51	Forward join-accept message	DevAddr:	26 0B 30 C3	JoinEUI:	70 B3 D5 7E D0 03 4B DA	DevEUI:	70 B3 D5 7E D0 06 FC FF	

Para configurar el segundo cubecell, solo debemos volver a cambiar los valores:

```
/* OTAA substituir estos valores por los de tu dispositivo*/
uint8_t devEui[] = { 0x70, 0xB3, 0xD5, 0x7E, 0xV0, 0x06, 0xFD, 0x09 };
uint8_t appEui[] = { 0x70, 0xB3, 0xD5, 0x90, 0xD0, 0x07, 0x7A, 0xC2 };
uint8_t appKey[] = { 0x16, 0x07, 0x02, 0x68, 0xFE, 0x62, 0x5C, 0x86, 0x6B, 0xAB, 0x9C, 0x06, 0xA8, 0x01, 0xAK, 0xED };
```

Por los valores correspondientes del segundo cubecell, subirle el código y veremos como directamente nos sale como está enviando datos en TTN, y en este caso, al haber configurado todo por OTAA, no necesitamos ninguna configuración al reiniciar los dispositivos, se conectan automáticamente.