

Conectando el coche GreenPower a una plataforma IoT

Configuración de Adafruit

1. Crear y acceder a tu cuenta

- Accede a <https://io.adafruit.com/>

Sign In

Create an Adafruit Account

- Dale a **Sign In**, y luego abajo a **Create an Adafruit Account**. Utiliza tu correo de Educastur para crear la cuenta. Cuando la tengas creada, entra en ella.
- Una vez creada, deberá aparecer en la parte superior una barra similar a esta:



2. Crear feeds o variables online: Los feeds son lugares en la nube donde puedes guardar cosas, leerlas y escribirlas desde la propia plataforma o desde otros lugares.

- Ve a **Feeds**
- Crea 6 feeds nuevos **New Feed**, uno para cada variable que quieras ver desde tu plataforma IoT, llámalos: “temperatura”, “i_consumida”, “v_bateria”, “fan_on”, “p_consumida”, “q_restante”.

Default			
Feed Name	Key	Last value	Recorded
<input type="checkbox"/> fan_on	fan-on		less than a minute ago
<input type="checkbox"/> i_consumida	i-consumida		less than a minute ago
<input type="checkbox"/> p_consumida	p-consumida		less than a minute ago
<input type="checkbox"/> q_restante	q-restante		less than a minute ago
<input type="checkbox"/> temperatura	temperatura		less than a minute ago
<input type="checkbox"/> v_bateria	v-bateria		less than a minute ago

3. Guardar tus códigos de Adafruit

- Entra en el icono
- Aquí aparece tu clave actual de Adafruit, la **“Active Key”**, esta clave permite a cualquiera leer y escribir los feeds que acabas de crear, así que asegúrate de que sólo la utilices tú.
- Copia la forma de incluir tu usuario y Active Key en Arduino en el portapapeles, o mantén esta ventana abierta hasta que los necesites.

Arduino

```
#define IO_USERNAME
#define IO_KEY
```

Configuración de Arduino

1. Abre el archivo “receptor.ino” que creamos en la práctica anterior, que debería tener esto escrito. Tómate un tiempo para refrescar bien qué significaba cada parte.


```
1  #define HELTEC_POWER_BUTTON
2  #include <heltec_unofficial.h>
3  float temperatura;
4  float i_consumida;
5  float v_bateria;
6  float fan_on;
7  float p_consumida;
8  float q_restante;
9  //Funcion guardarMensaje a partir del mensaje recibido
10 void guardarMensaje(String mensaje){
11     //Encontrar comas: función "indexOf"
12     int coma1 = mensaje.indexOf(',');
13     int coma2 = mensaje.indexOf(',', coma1 + 1);
14     int coma3 = mensaje.indexOf(',', coma2 + 1);
15     int coma4 = mensaje.indexOf(',', coma3 + 1);
16     //Extraer subcadenas de texto: función "if" y "substring"
17     if (coma1 > 0 && coma2 > 0 && coma3 > 0 && coma4 > 0) {
18         String textoTemperatura = mensaje.substring(0, coma1);
19         String textoIConsumida = mensaje.substring(coma1 + 1, coma2);
20         String textoVBateria = mensaje.substring(coma2 + 1, coma3);
21         String textoFanOn = mensaje.substring(coma3 + 1, coma4);
22         String textoQRestante = mensaje.substring(coma4 + 1);
23         //Transformar texto a variables
24         float temperatura = textoTemperatura.toFloat();
25         float i_consumida = textoIConsumida.toFloat();
26         float v_bateria = textoVBateria.toFloat();
27         float fan_on = textoFanOn.toFloat();
28         float p_consumida = i_consumida*v_bateria;
29         float q_restante = textoQRestante.toFloat();
30     }
31 }
32
33 //Función setup, funciones radio.begin() y Serial.begin()
34 void setup() {
35     heltec_setup();
36     Serial.begin(115200);
37     int state = radio.begin(868.0);
38     if (state == RADIOLIB_ERR_NONE) {
39         Serial.println("Radio works");
40     } else {
41         Serial.printf("Radio fail, code: %i\n", state);
42         while (true);
43     }
44 }
```

```

45
46 //Función loop, funciones radio.receive y Serial.print()
47 void loop() {
48     String mensaje;
49     //Recibir mensaje en String
50     int state = radio.receive(mensaje);
51     if (state == RADIOLIB_ERR_NONE && mensaje.length() > 0) {
52         //Lee el mensaje y lo guarda en las variables globales
53         guardarMensaje(mensaje);
54         //Envía todos los datos por puerto serie, separados por comas:
55         Serial.print(temperatura, 2);
56         Serial.print(",");
57         Serial.print(i_consumida, 2);
58         Serial.print(",");
59         Serial.print(v_bateria, 2);
60         Serial.print(",");
61         Serial.print(fan_on);
62         Serial.print(",");
63         Serial.print(p_consumida, 2);
64         Serial.print(",");
65         Serial.println(q_restante, 3);
66     }
67 }

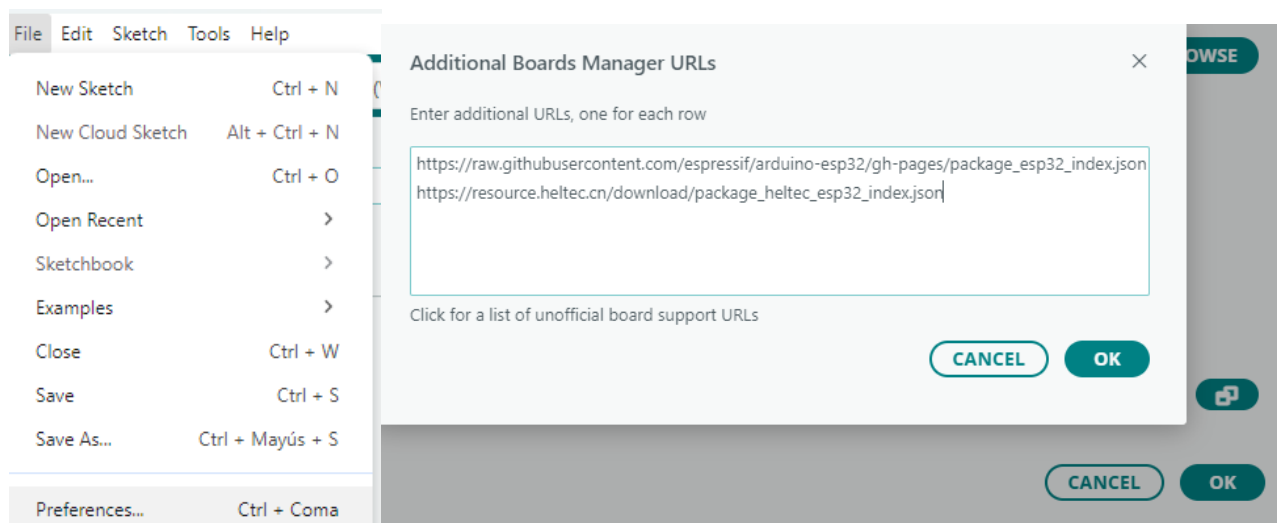
```


2. Añade comentarios con // en las partes que más te cueste recordar para qué servían

3. Añade las direcciones web de donde descargar las librerías. Para ello, ve a File>Preferences. Abajo, en “Additional Boards and...” dale al botón “” y pega ahí las siguientes direcciones:

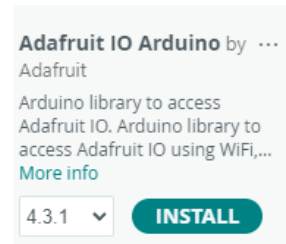
https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json


https://resource.heltec.cn/download/package_heltec_esp32_index.json




4. Instala las librerías necesarias. Para ello, ve al Gestor de Librerías () y busca e instala las siguientes librerías. Dale siempre a “**INSTALL ALL**” cuando pregunte.

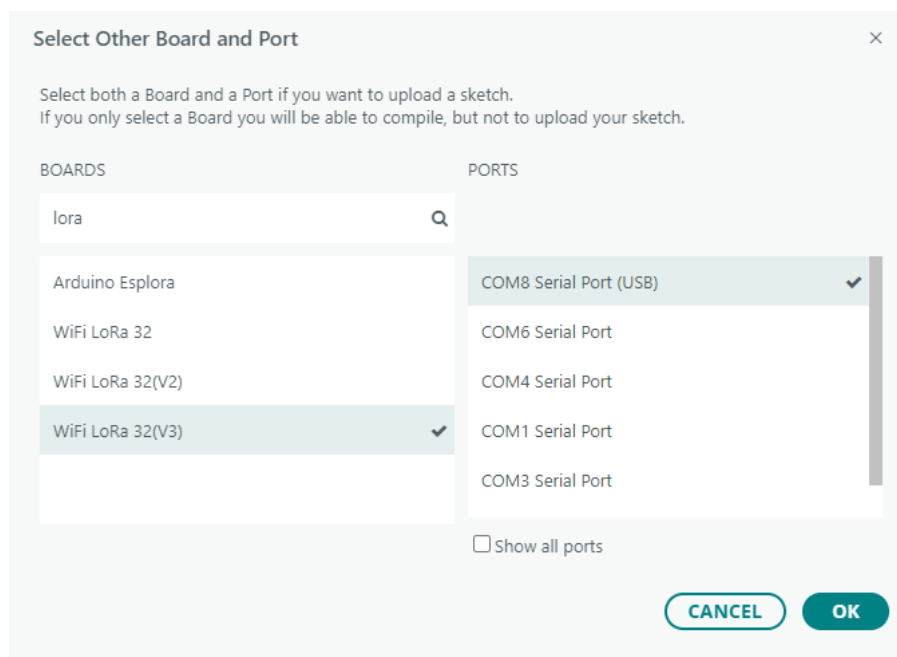
- Adafruit IO Arduino
- Adafruit MQTT Library
- Heltec ESP32 LoRa v3 (de Rop Gonggrijp)
- LoRa (de Sandeep Mistry)



5. Instala las tarjetas que vamos a usar. Para ello, ve al Gestor de tarjetas () y busca e instala:

- ESP32 de Espressif Systems

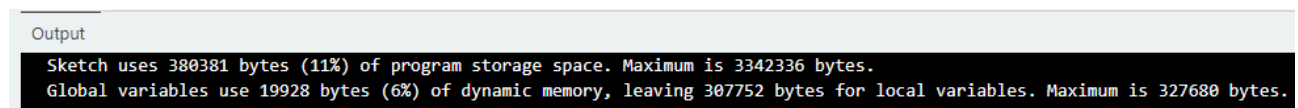
6. Escoge la placa instalada desde el desplegable de arriba, dándole a  y luego a **Select other board and port...**. Allí, busca la placa WiFi Lora 32 (V3) y elige el puerto donde tengas la placa conectada.



Deberá quedar así:



7. Compila el programa, dándole a . Si todo ha ido bien, deberá aparecer un mensaje en color blanco.



Programación en Arduino

1. Añade tus credenciales de Adafruit IO justo antes de las variables globales.

```
3  #define IO_USERNAME  "TU_USUARIO"
4  #define IO_KEY        "TU_ADAFRUIT_ACTIVE_KEY"
```

2. Añade las credenciales de la conexión WiFi a la que se vaya a conectar la placa receptora. Te recomiendo usar las de la propia red WiFi que crea la profesora, para que el sistema sea transportable:

```
5  #define WIFI_SSID     "CidercarNava"
6  #define WIFI_PASS     "tecnologia4eso"
```

3. Crea un objeto para interactuar con Adafruit que utiliza estas variables predefinidas

```
8  AdafruitIO_WiFi io(IO_USERNAME, IO_KEY, WIFI_SSID, WIFI_PASS);
```

Si te fijas, AdafruitIO_WiFi no se marca de un color como suelen hacerlo las variables, eso es porque no hemos incluido la librería que permite utilizarla.

4. Incluye arriba del todo la librería para poder usar este objeto

```
3  #include "AdafruitIO_WiFi.h"
```

Si le das ahora a compilar , deberá funcionar por el momento todo correcto.

5. Crea los objetos que te permitirán guardar valores en los feeds creados anteriormente. Tienen que tener exactamente el mismo nombre o no funcionará.

```
10  AdafruitIO_Feed *feed_i=io.feed("i_consumida");
11  AdafruitIO_Feed *feed_v=io.feed("v_bateria");
12  AdafruitIO_Feed *feed_temp=io.feed("temperatura");
13  AdafruitIO_Feed *feed_p=io.feed("p_consumida");
14  AdafruitIO_Feed *feed_q=io.feed("q_restante");
15  AdafruitIO_Feed *feed_fan=io.feed("fan_on");
```

6. En la función setup, asegúrate de conectar la placa a la nube al empezar el programa:

```
58  //Conexión a la nube IoT
59  io.connect();
60  while(io.status() < AIO_CONNECTED){
61    delay(500);
62  }
63  Serial.println("Conectado a Adafruit IO");
```

7. Guarda los datos en la nube justo encima de donde envías los datos por comunicación serie.

```
75  //Enviar los datos a la nube
76  feed_i->save(i_consumida);
77  feed_v->save(v_bateria);
78  feed_temp->save(temperatura);
79  feed_p->save(p_consumida);
80  feed_q->save(q_restante);
81  feed_fan->save(fan_on);
```

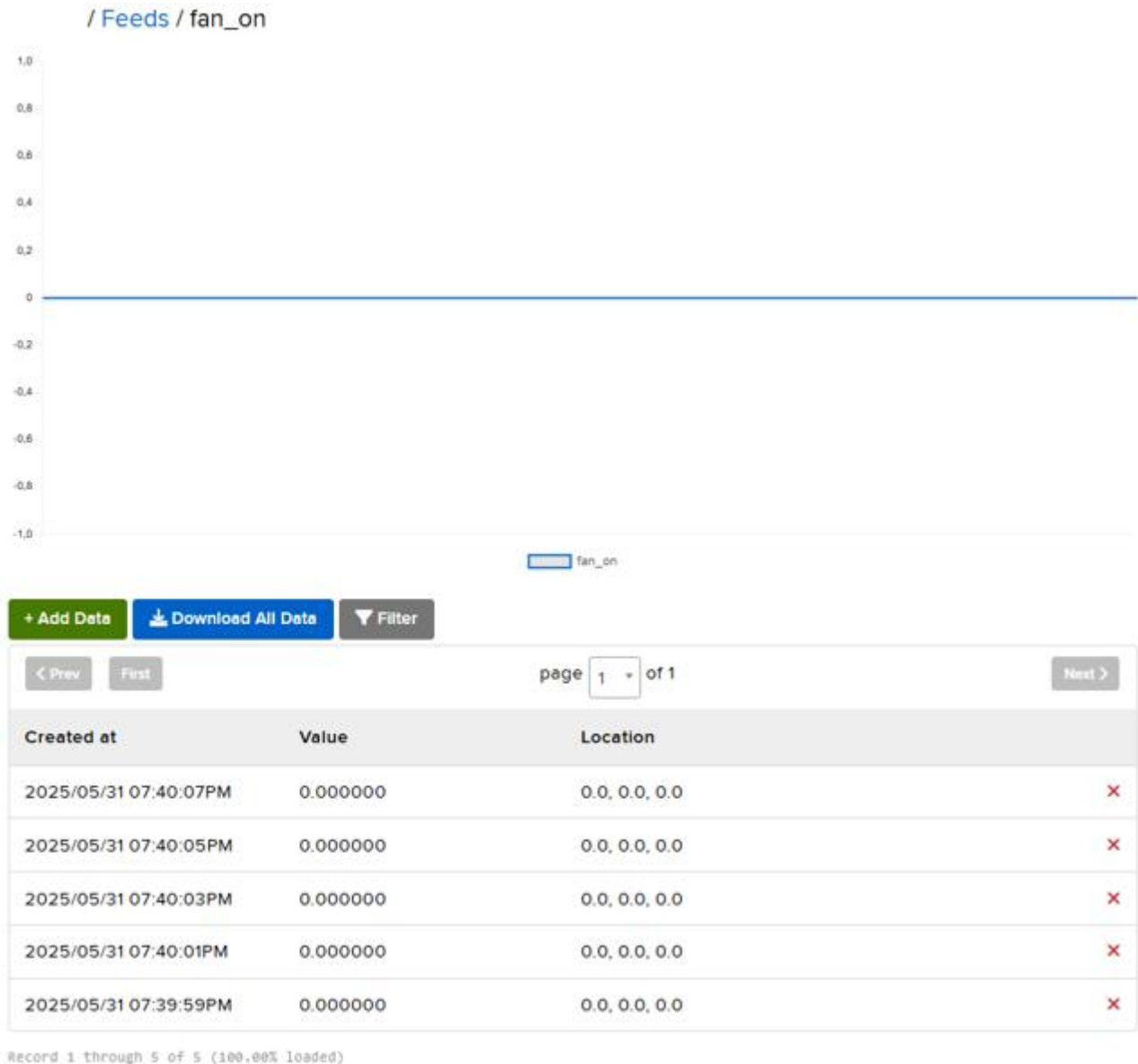
Fíjate en que esta vez la sintaxis es un poco diferente. Esto se debe a que estamos utilizando punteros, una herramienta de programación que permite acceder a objetos o variables a través de su dirección de memoria, en lugar de manipular directamente su contenido. En este caso, los punteros nos permiten trabajar con los feeds alojados en la nube de Adafruit IO, **sin tener que cargar todo el objeto completo en la memoria de la placa**.

Pruébalo

Cuando lo tengas, súbelo a la placa RECEPTORA de LoRa. Además:

- Asegúrate de que el WiFi al que se conecta la placa está activo y accesible.
- Abre el Monitor Serie en Arduino IDE y espera a que aparezca el mensaje: “Conectado a Adafruit IO”

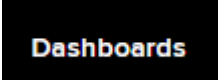
Cuando reciba un mensaje LoRa, verás los datos parseados en el monitor y enviados a la nube. Entra en io.adafruit.com, ve a tus Feeds, y comprueba que se actualizan los datos.



Si algo falla, revisa:

- Que el formato del mensaje sea correcto (con comas).
- Que los feeds estén creados en Adafruit IO.
- Que el código se haya subido correctamente a la placa.

Haz un Dashboard para ver

- Ve a . Esto sirve para mostrar los datos de forma un poco más estética

- Crea un nuevo Dashboard, llamado “Cidercar Telemetría”

/ Dashboards / Cidercar Telemetría

- Dale a  y . Ahí, puedes añadir las interfaces que prefieras para mostrar las 6 variables leídas. Por ejemplo, puedes añadir:

- Añade un Toggle para fan_on, seleccionando “Toggle”, escogiendo el feed “fan_on”.




- Añade un “Battery Gauge” para q_restante, indicando que el “Max Value” es 36Ah

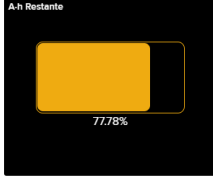
Block Title (optional)

Gauge Min Value

Gauge Max Value

☒ Show Feed Percentage
When checked, show the value of the feed as a percent.

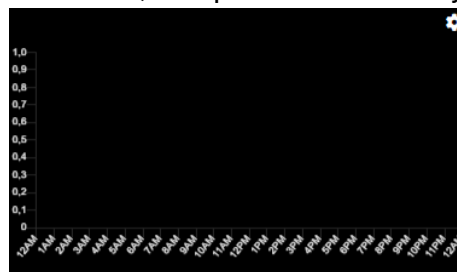
High Color


Block Preview


Line Gauge A line gauge is a read only block type that shows a fixed range of values.

Test Value

- Añade un LineChart para v_batería, otro para i_consumida y otro para p_consumida.



- Añade un Gauge para la temperatura.

Block Title (optional)

Gauge Min Value

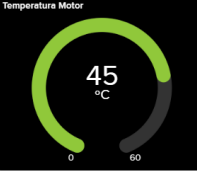
Gauge Max Value

Gauge Width

Gauge Label

Low Warning Value

Optional! If no low warning value is given

Block Preview


Gauge A gauge is a read only block type that shows a fixed range of values.

Test Value

¡Ahora solo queda probarlo con el coche en marcha!