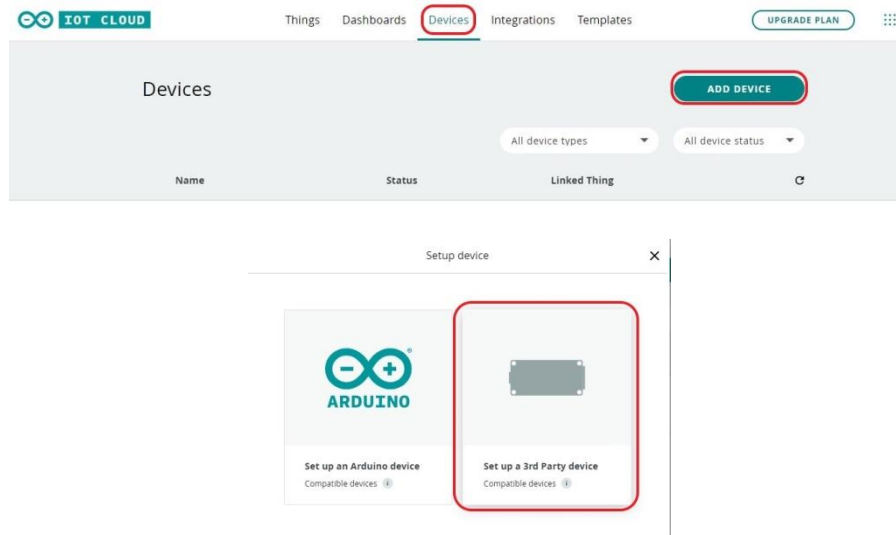


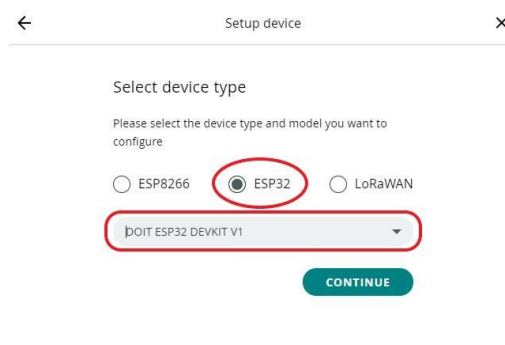
Ejercicio 1 d) Como implementaría un sensor inteligente de temperatura?

Una vez creada la cuenta en Arduino IoT Cloud, y accedido a la misma (se puede usar la cuenta de Arduino.cc si ya la teníamos creada).

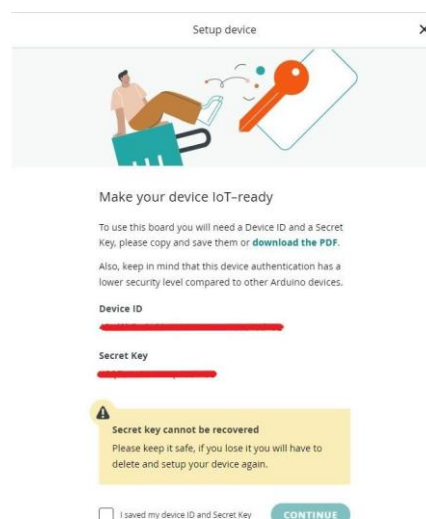
Lo primero que haremos es agregar un dispositivo para controlar nuestro sensor de temperatura, en este caso usaremos una placa de desarrollo ESP32 y un sensor temperatura y humedad modelo DHT11, como muestran las imágenes abajo.



Se selecciona ESP32 y se busca el modelo específico de la lista, luego de seleccionarlo se nombra el dispositivo.



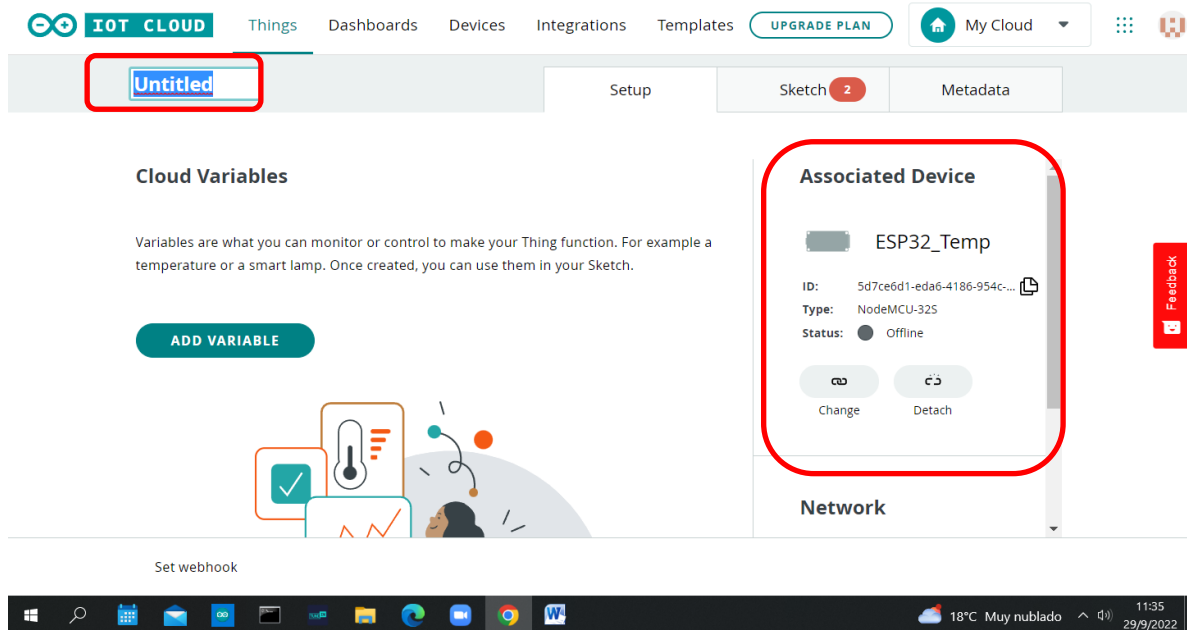
Entonces el dispositivo está ahora registrado. Se debe guardar el ID del dispositivo y la clave secreta, ya que serán necesarios en el futuro.



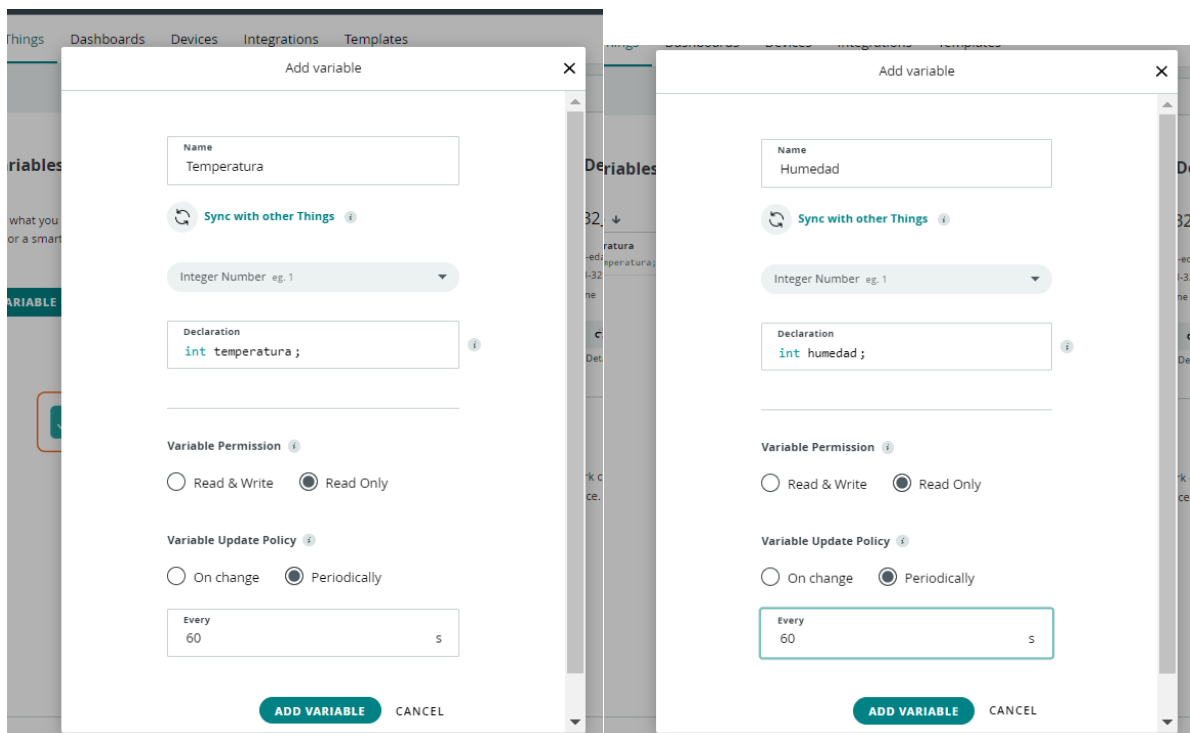
Ahora debemos crear una “cosa” como muestra la imagen abajo.



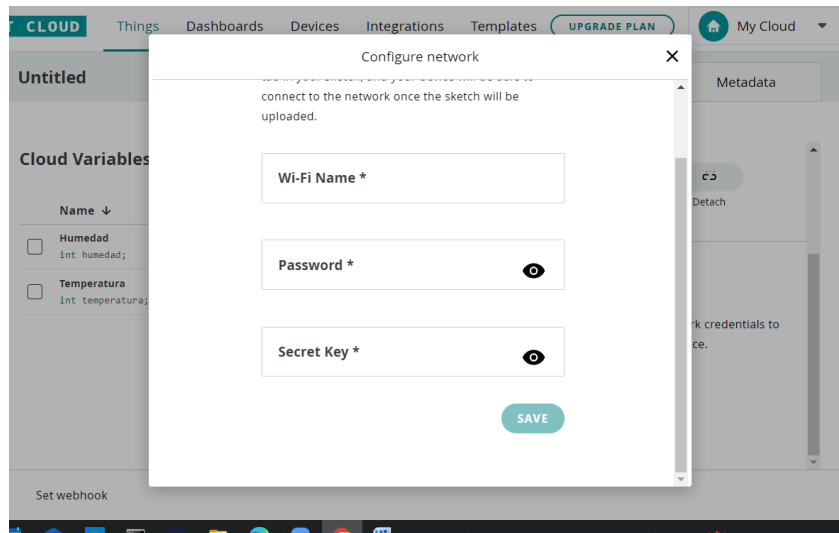
Una vez creada, se le debe dar un nombre, y vemos que la placa ya está asociada a nuestra cosa:



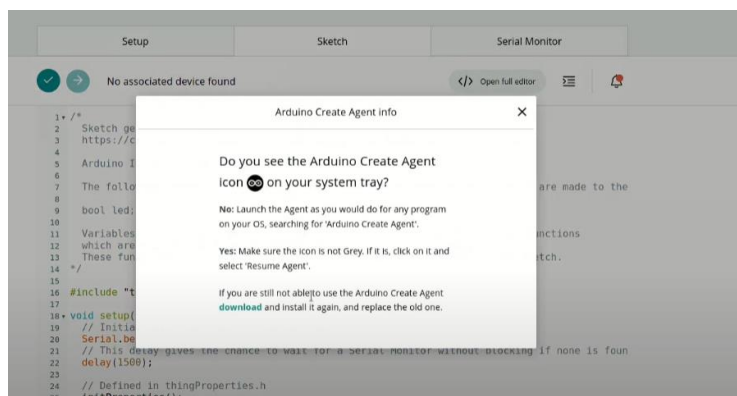
Luego, creamos y agregamos las variables, en nuestro caso una para la temperatura y otra para la humedad, que serán nuestros valores censados.



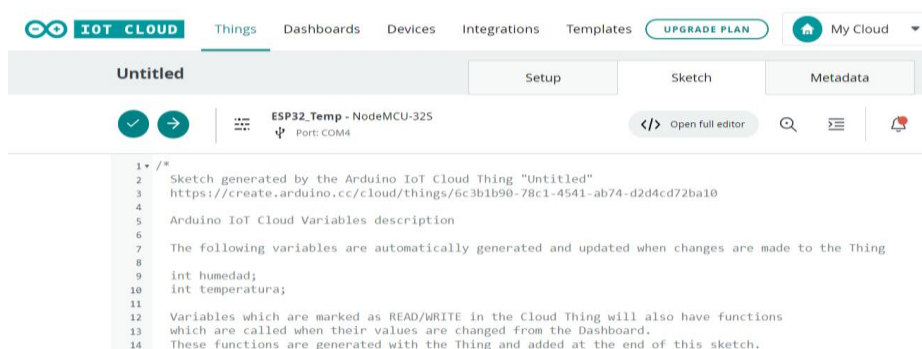
Ahora se debe configurar la red a la que se conectará la placa ESP32, con las credenciales de red y la clave secreta proporcionada por Arduino Cloud IoT:



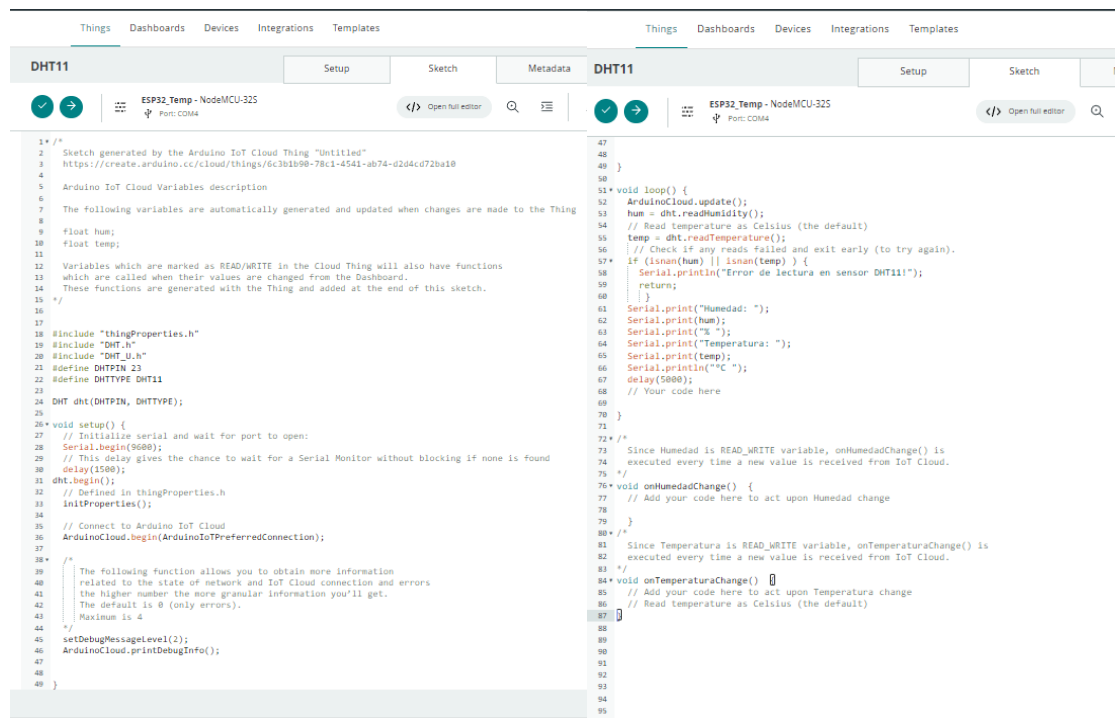
Configurada la conexión, debemos asegurarnos que éste instalado el agente de creación de Arduino para cargar el código a nuestra placa, si no lo está al ingresar a la solapa Sketch aparecerá este mensaje, se debe descargar el instalador y ejecutar, dándole permiso al firewall de Windows en este caso:



Hecho esto podemos ver que nuestra ESP32 está enlazada con Arduino Cloud IoT.

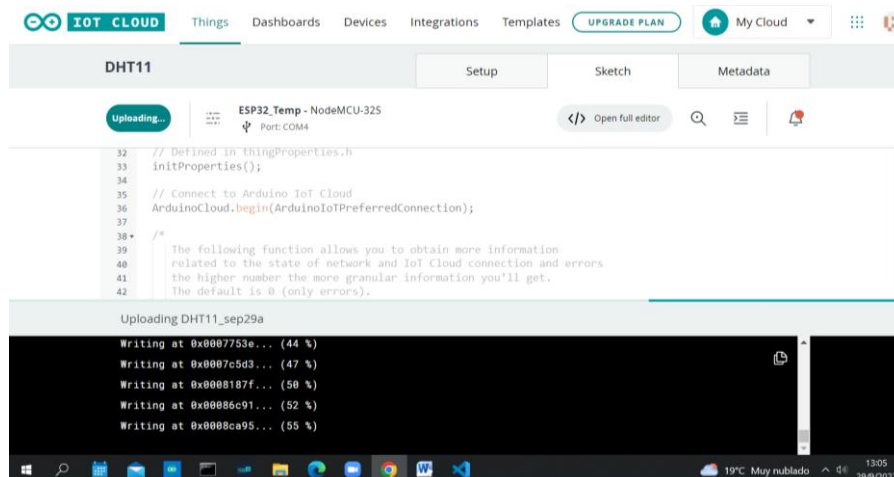


Desarrollamos simple código para registrar los valores de temperatura y humedad:

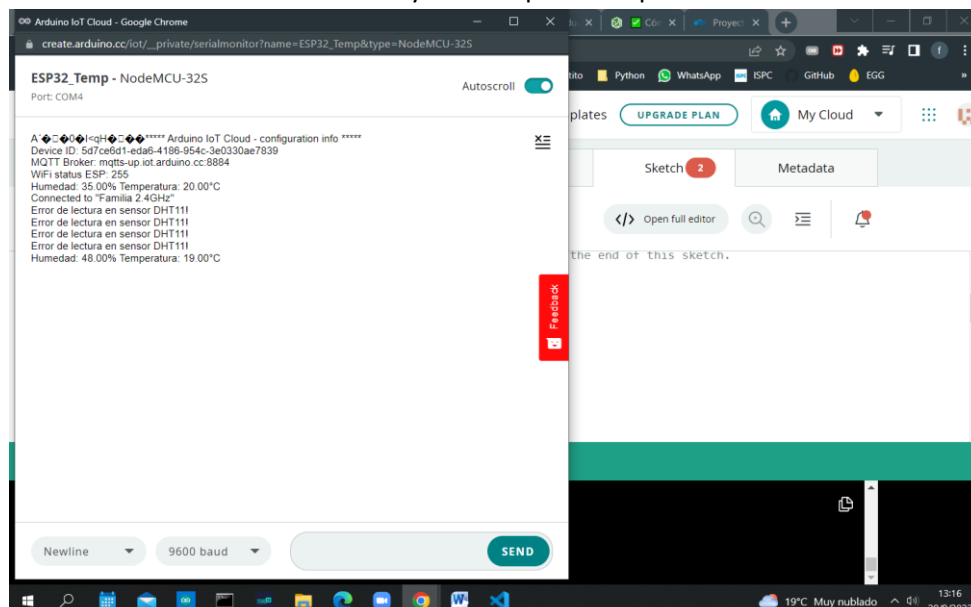


```
1 /*  
2 Sketch generated by the Arduino IoT Cloud Thing "Untitled"  
3 https://create.arduino.cc/cloud/things/6c3b1890-78c1-4541-ab74-d2d4c072ba18  
4  
5 Arduino IoT Cloud Variables description  
6  
7 The following variables are automatically generated and updated when changes are made to the Thing  
8  
9 float hum;  
10 float temp;  
11  
12 Variables which are marked as READ/WRITE in the Cloud Thing will also have functions  
13 which are called when their values are changed from the Dashboard.  
14 These functions are generated with the Thing and added at the end of this sketch.  
15 */  
16  
17 #include "thingProperties.h"  
18 #include "DHT.h"  
19 #include "DHT_U.h"  
20 #define DHTPIN 23  
21 #define DHTTYPE DHT11  
22  
23 DHT dht(DHTPIN, DHTTYPE);  
24  
25 void setup() {  
26 // Initialize serial and wait for port to open:  
27 Serial.begin(9600);  
28 // This delay gives the chance to wait for a Serial Monitor without blocking if none is found  
29 delay(1000);  
30 dht.begin();  
31 // Defined in thingProperties.h  
32 initProperties();  
33  
34 // Connect to Arduino IoT Cloud  
35 ArduinoCloud.begin(ArduinoIoTPreferredConnection);  
36  
37 /*  
38 The following function allows you to obtain more information  
39 related to the state of network and IoT Cloud connection and errors  
40 the higher number the more granular information you'll get.  
41 The default is 0 (only errors).  
42 Maximum is 4  
43 */  
44 setDebugLogLevel(2);  
45 ArduinoCloud.printDebugInfo();  
46  
47 }  
48  
49 }  
50  
51 void loop() {  
52 ArduinoCloud.update();  
53 hum = dht.readHumidity();  
54 // Read temperature as Celsius (the default)  
55 temp = dht.readTemperature();  
56 // Check if any reads failed and exit early (to try again).  
57 if (isnan(hum) || isnan(temp)) {  
58 Serial.println("Error de lectura en sensor DHT11!");  
59 return;  
60 }  
61 Serial.print("Humedad: ");  
62 Serial.print(hum);  
63 Serial.print(" ");  
64 Serial.print("Temperatura: ");  
65 Serial.print(temp);  
66 Serial.print("°C ");  
67 delay(5000);  
68 // Your code here  
69 }  
70 }  
71  
72 /*  
73 Since Humedad is READ_WRITE variable, onHumedadChange() is  
74 executed every time a new value is received from IoT Cloud.  
75 */  
76 void onHumedadChange() {  
77 // Add your code here to act upon Humedad change  
78 }  
79  
80 /*  
81 Since Temperatura is READ_WRITE variable, onTemperaturaChange() is  
82 executed every time a new value is received from IoT Cloud.  
83 */  
84 void onTemperaturaChange() {  
85 // Add your code here to act upon Temperatura change  
86 // Read temperature as Celsius (the default)  
87 }  
88  
89  
90  
91  
92  
93  
94  
95
```

Y lo cargamos a nuestra placa ESP32:

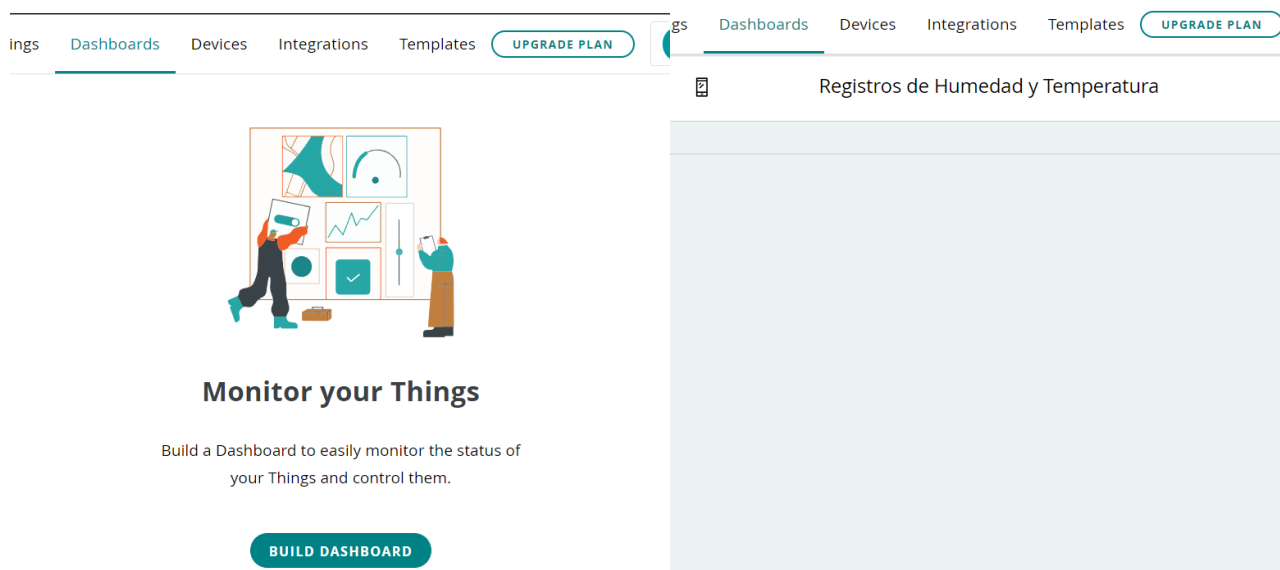


Con el Monitor Serie nos ayudamos para ver que está funcionando:

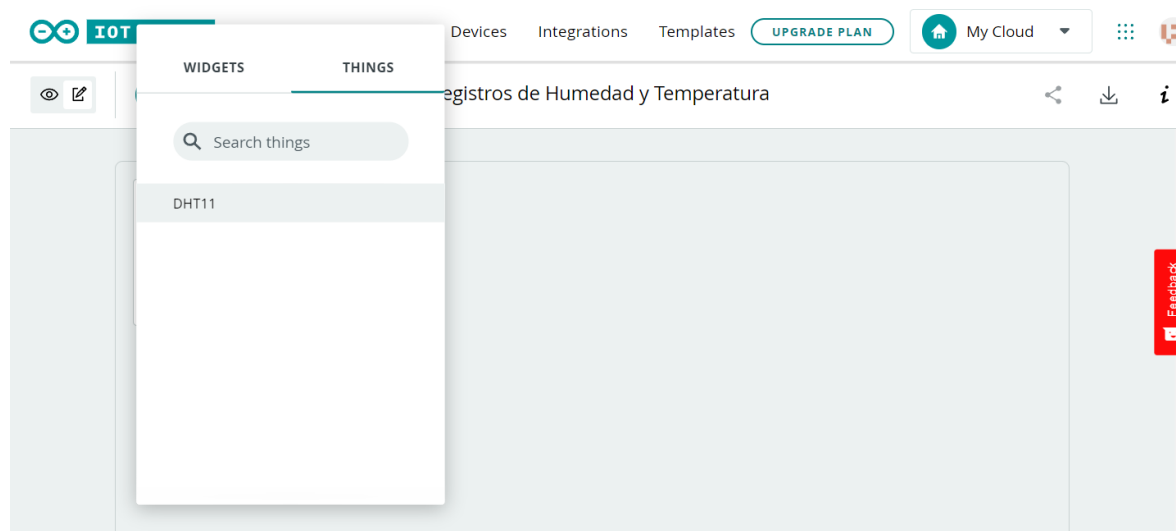


Para visualizar los datos de temperatura y humedad desde ESP32 y tener un registro de los mismos nos vamos a ayudar con los widgets de gráficos en el Dashboard de Arduino.

Creamos un dashborad y le ponemos nombre:



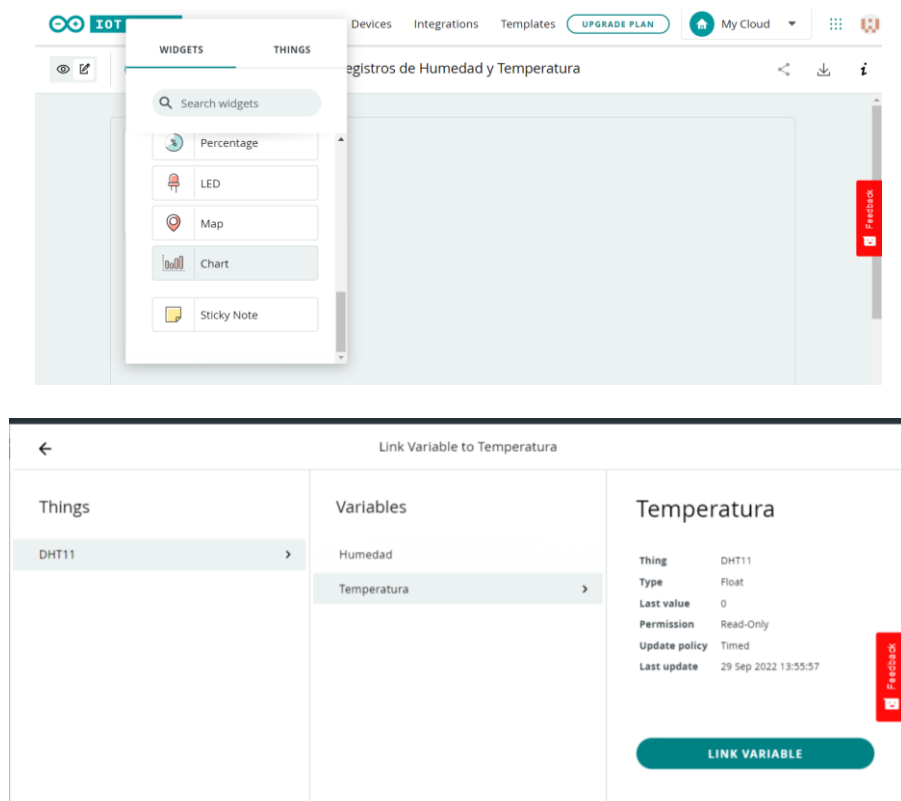
Agregamos nuestra “cosa” creada anteriormente al panel:



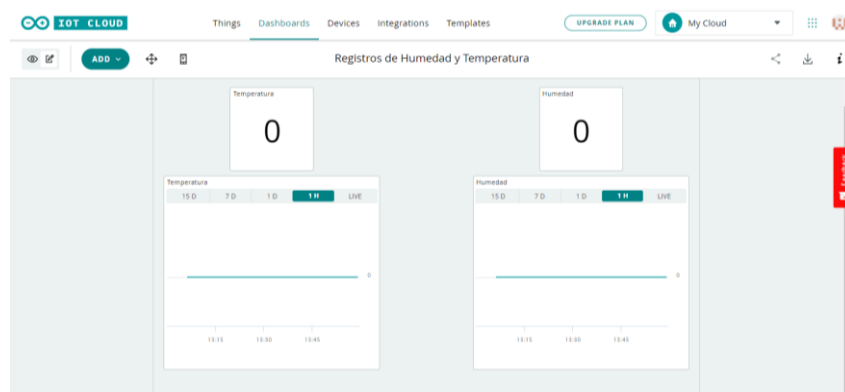
Nos agrega nuestras variables de temperatura y humedad:



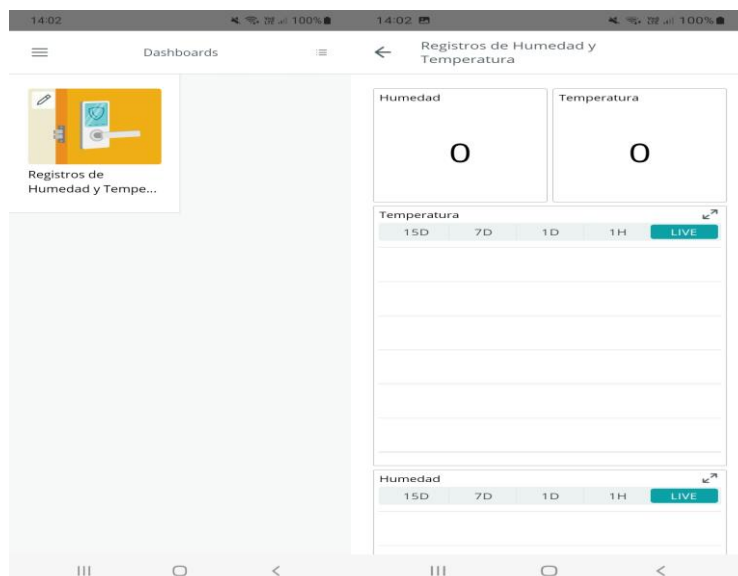
Ahora agregamos nuevos widgets que asociaremos a nuestras variables para tener una gráfica de las mediciones:



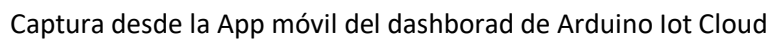
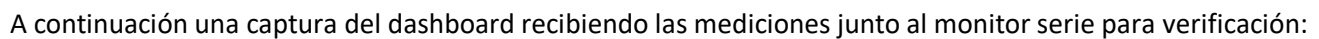
Quedando nuestro tablero resultante así:



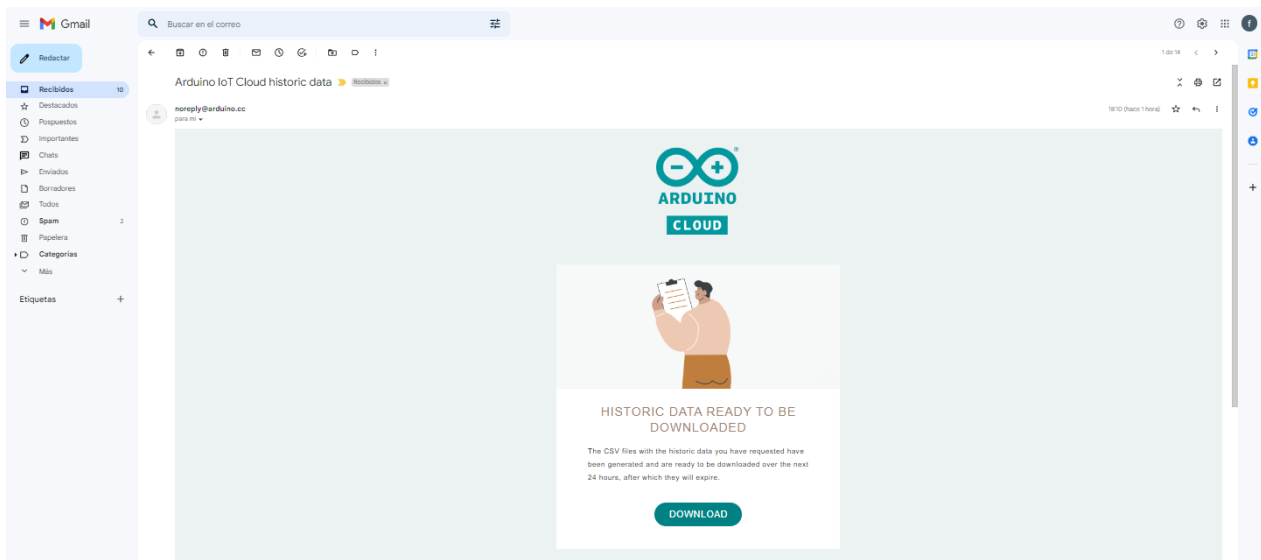
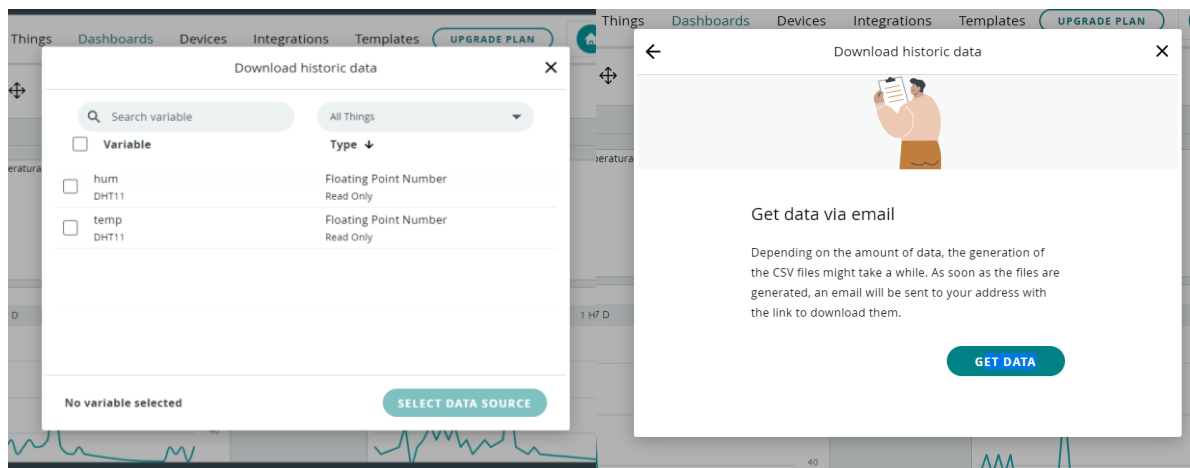
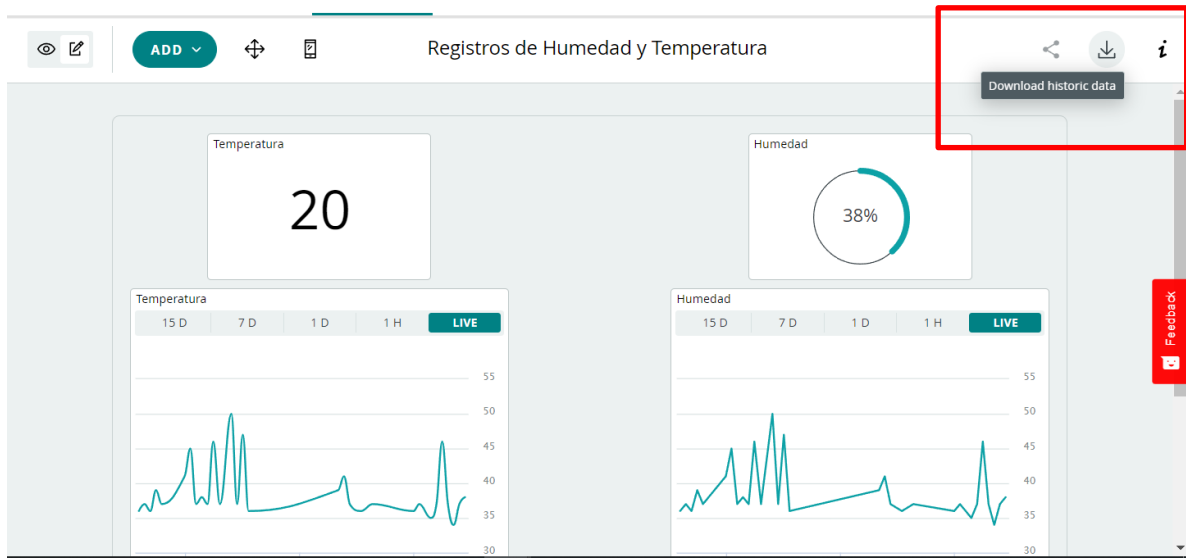
También podemos visualizarlo en el teléfono móvil, con la app de Arduino Cloud(Arduino Iot Cloud Remote)

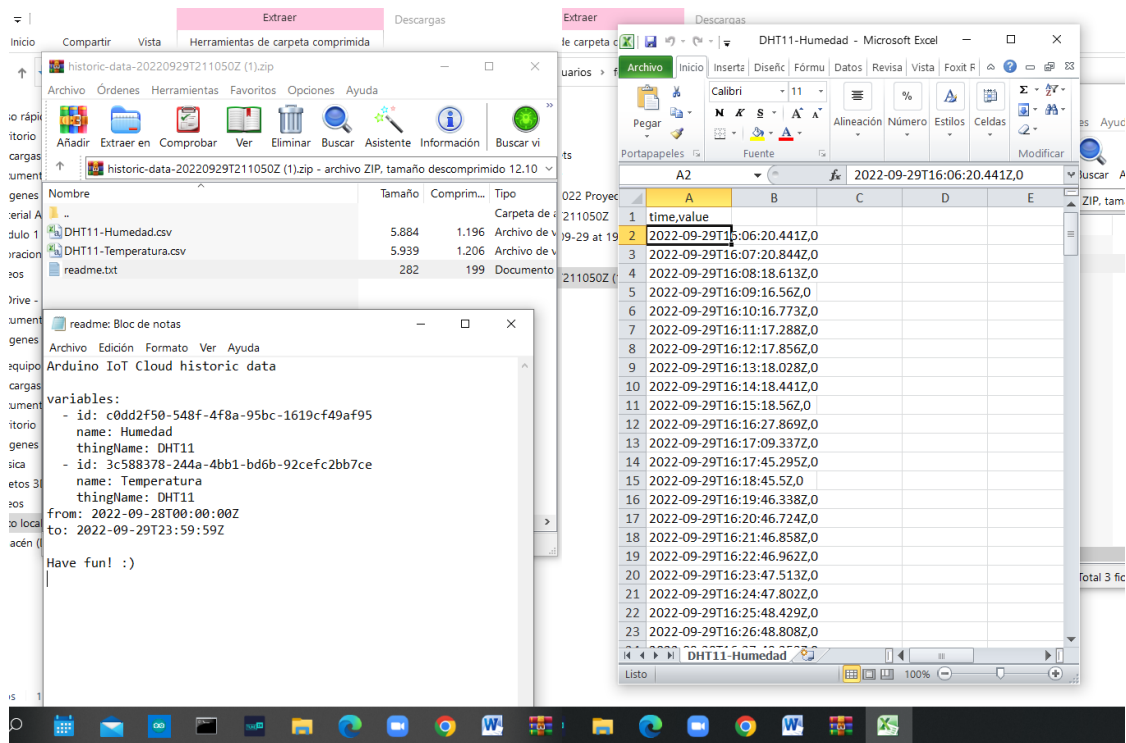


La placa ESP32 se encuentra en línea con Arduino IoT Cloud como muestra la siguiente imagen:

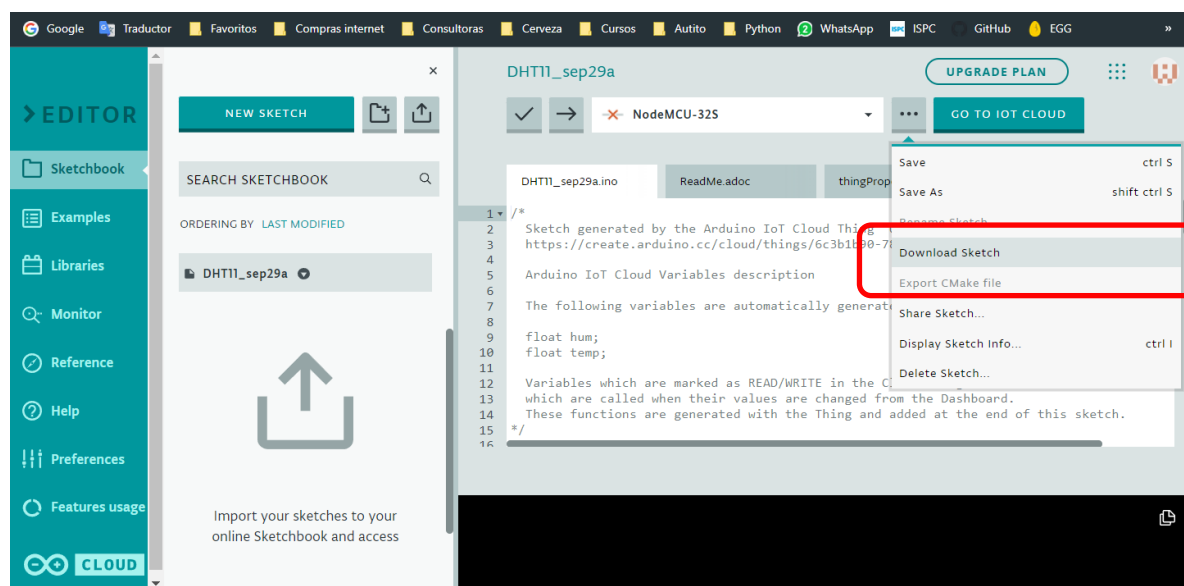


Además estos registros pueden ser descargados, la plataforma de Arduino envía un email un archivo .zip con una hoja de cálculo por cada variable involucrada:





De la misma manera es posible descargar todo el código desde el editor del sketch



Se descargará un archivo .zip con todo nuestro código.

Por último la plataforma Arduino IoT Cloud ofrece el proyecto de para un control de termostato con sus dispositivos de pago, pero que brinda un tutorial incluyendo el código de desarrollo para que se pueda tomar como referencia.

Link: <https://create.arduino.cc/iot/templates/thermostat-control>