Parcial 2 Rest y Plataforma IoT

Diego Iván Perea Montealegre (2185751) diego.perea@uao.edu.co

Facultad de Ingeniería, Universidad Autónoma de Occidente

Cali, Valle del Cauca

Siguiendo el contexto del primer parcial sobre las condiciones de almacenamiento óptimo para las principales especies de frutas y hortalizas como lo es limón y la cebolla , en donde se utilizará el DHT11 como sensor de temperatura y humedad ya explicados en el primer parcial , dado esto se debe enviar los datos obtenidos a una plataforma Iot , en donde esta permita realizar Rest , por lo que se deben realizar las acciones de GET (obtención de los datos publicados) y Post (publicación de los datos);Continuando los parámetros dispuestos se selecciona una plataforma Iot que permita lo anterior , esta plataforma es “Ubidots” .

Figura 1. Plataforma IoT, ubidots

Los datos a enviar serán la temperatura y la humedad, la fecha y hora no se enviarán debido a que la plataforma IoT ubidots nos proporciona esos datos cuando el dato de la temperatura y humedad sean entregados, con lo cual se realizan dos json para la fecha y hora y otro para la humedad y temperatura, este primero se realiza para visualizar en el serial monitor para tener certeza del tiempo de la información y se recalca que no se va enviar .

Para que realice el proceso de envío de los datos de la temperatura y humedad se utiliza la librería “HTTPClient.h” que realiza fácilmente solicitudes HTTP GET, POST y PUT a un servidor web. Funciona con cualquier clase derivada de Client, por lo que cambiar entre Ethernet, WiFi y GSMClient requiere cambios de código mínimos.

A continuación se muestra el código utilizado para la publicación de los datos y visualización de ellos:

```
#include <Arduino.h>

#include <HTTPClient.h>
#include <WiFi.h>
#include <ArduinoJson.h>
//LIBRERIAS PARA DHT11 (TEMPERATURA Y HUMEDAD)
#include <Adafruit_Sensor.h>
#include <DHT.h>
//LIBRERIAS PARA FECHA Y HORA
#include <NTPClient.h>
#include <WiFiUdp.h>
// Define NTP Client to get time
WiFiUDP ntpUDP;
NTPClient timeClient(ntpUDP);

// Variables to save date and time
String formattedDate;
String dayStamp;
String timeStamp;


//DEFINICION DE PINES DHT11
#define DHTPIN 4   // 4 = PIN D4
#define DHTTYPE    DHT11
DHT dht(DHTPIN, DHTTYPE);









const char* ssid = "****"; //El SSID de la red wifi a la que se conectará
const char* password = "*****"; //El password para conectarse a la red
inalambrica


void setup_wifi() {
  delay(10);
  // We start by connecting to a WiFi network
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
  delay(500);
```

```
  Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
  // Initialize a NTPClient to get time
  timeClient.begin();
  // Set offset time in seconds to adjust for your timezone, for example:
  // COLOMBIA -5 , entonces -5*3600 ->  -18000
  timeClient.setTimeOffset(-18000); //Thailand +7 = 25200
}
void setup() {
  Serial.begin(9600); //Serial connection
  setup_wifi(); //WiFi connection
  delay(1500);
}
void loop() {
  while(!timeClient.update()) {
  timeClient.forceUpdate();
  }
    // The formattedDate comes with the following format:
  // 2018-05-28T16:00:13Z
  // We need to extract date and time
  formattedDate = timeClient.getFormattedDate();
  // Extract date
  int splitT = formattedDate.indexOf("T");
  dayStamp = formattedDate.substring(0, splitT);
  //Serial.print("DATE: ");
  //Serial.println(dayStamp);
  // Extract time
  timeStamp = formattedDate.substring(splitT+1, formattedDate.length()-1);
  //Serial.print("HOUR: ");
  //Serial.println(timeStamp);

  //CODIGO----TEMPERATURA Y HUMEDAD---------------
  float h= dht.readHumidity();
  float t =dht.readTemperature();
  //JSON DE FECHA Y HORA no enviar
  String variable_fecha_y_hora;
  DynamicJsonDocument doc1(1024);
  doc1["Fecha"] = dayStamp;
  doc1["Hora"] = timeStamp;
  serializeJson(doc1, variable_fecha_y_hora);
```

```
  //JSON Humedad y temperatura a enviar platafoma IoT
  String variable;
  DynamicJsonDocument doc(1024); //creacion del json
  doc["temperatura(°C)"] = t;
  doc["humedad(%)"] = h;


  serializeJson(doc, variable);
  Serial.println("dato a enviar: "+ variable);
  Serial.println("dato no enviar: "+ variable_fecha_y_hora);
  HTTPClient http; //Declare object of class HTTPClient
  WiFiClient client;
  //Specify request destination
  //http.begin(client, "URL A INGRESAR");
  //http.begin(client, "http://192.16*.*.*:3000/datos/"); LOCAL URL
  //http.begin(client,
"http://things.ubidots.com/api/v1.6/devices/name_device/?token=your_token_ap
i_credentials");
  http.begin(client,
"http://things.ubidots.com/api/v1.6/devices/esp32/?token=BBFF-
baawaxKZeBQm3NHX7k0IlKxBDtWgVc");
  http.addHeader("Content-Type", "application/json"); //Specify contentype
header
  int httpCode = http.POST(variable); //Send the request
  String payload = http.getString(); //Get the response payload
  Serial.println(httpCode); //Print HTTP return code
  Serial.println(payload); //Print request response payload
  http.end(); //Close connection
  delay(5000); //Send a request every 5 seconds
}

//PARA POSTMAN
GET:http://things.ubidots.com/api/v1.6/devices/esp32/temperatura-
degc/values?token=BBFF-baawaxKZeBQm3NHX7k0IlKxBDtWgVc
//PARA POSTMAN
POST:  http://things.ubidots.com/api/v1.6/devices/esp32/?token=BBFF-
baawaxKZeBQm3NHX7k0IlKxBDtWgVc
```

Estando en la plataforma de ubidots se crea un dispositivo en blanco, en donde se le dará un nombre a ese dispositivo, por el que se estima con el nombre de “esp32” (observar figura 2 y figura 3)

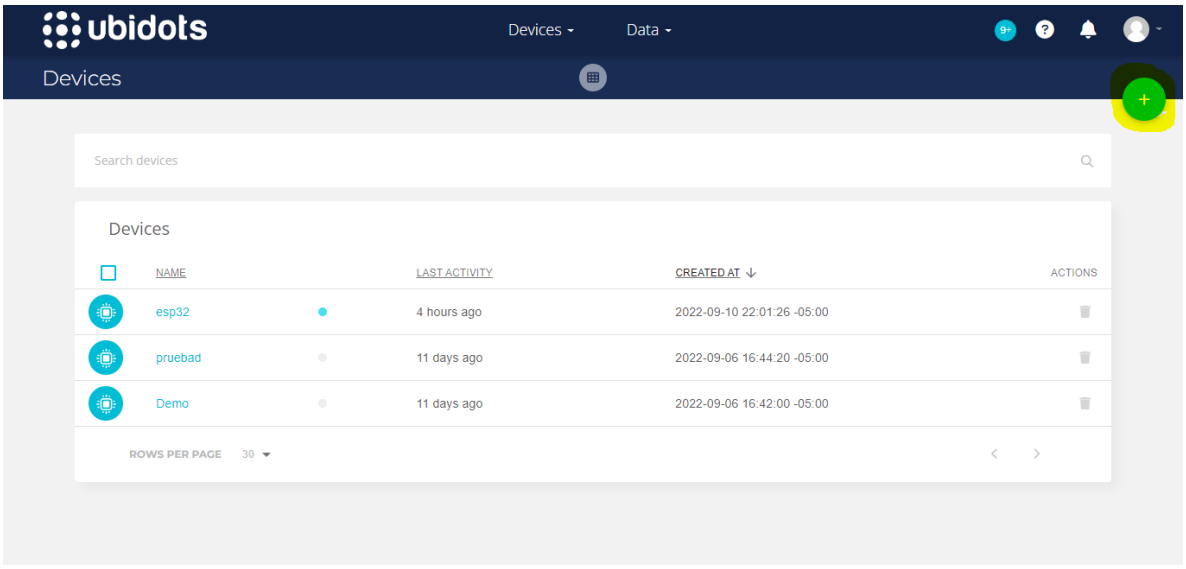

Figura 2. Página principal de ubidots de creación de dispositivo.

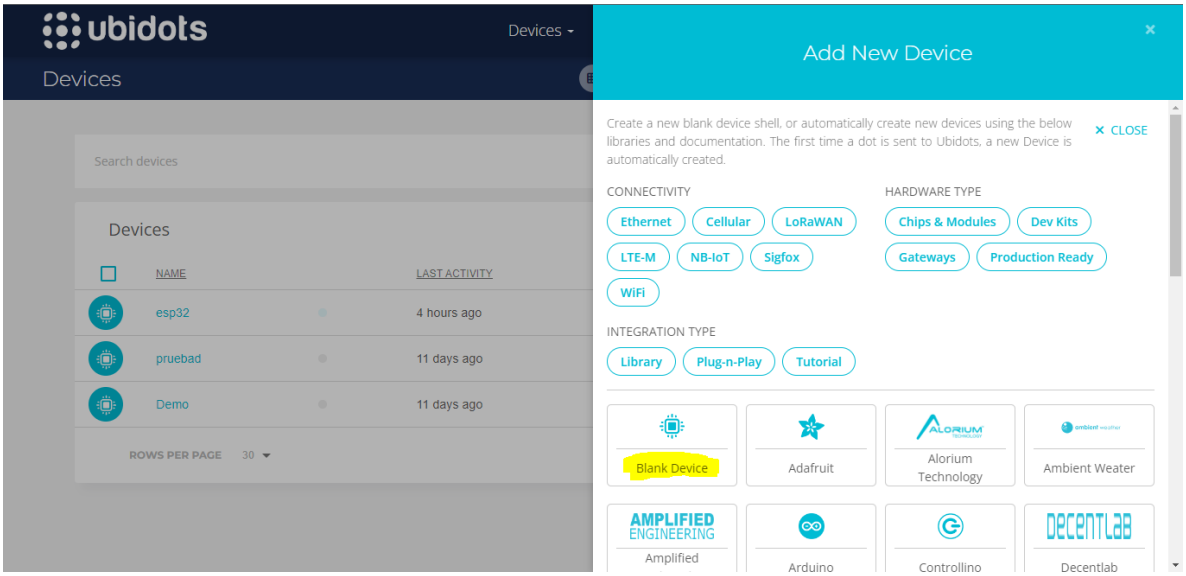

Figura 3. Creación de dispositivo y nombramiento en ubidots.

Tener en cuenta que el API label es el nombre en donde se identifica el dispositivo para que se pueda enviar los datos , el nombre de creación de las variables como la temperatura y humedad son dados por el nombre que se le adecuo en la creación del json (observar figura), como lo es en este caso en el json denominado en el almacenamiento “doc”.Estos nombres serán también tomados para la realización del Rest , que será realizada en el programa de “Postman” que permite realizar raw json con POST y GET y diferentes modalidades para la información .

Se abre el Serial monitor para visualizar los datos enviados de la temperatura y humedad, el dato a no enviar como lo es la fecha y la hora , en donde seobserva la efectividad del envio con el numero de 200 indicando el retorno del HTTP y la forma de status_code dando el envio del dato de la humedad y de la temperatura.

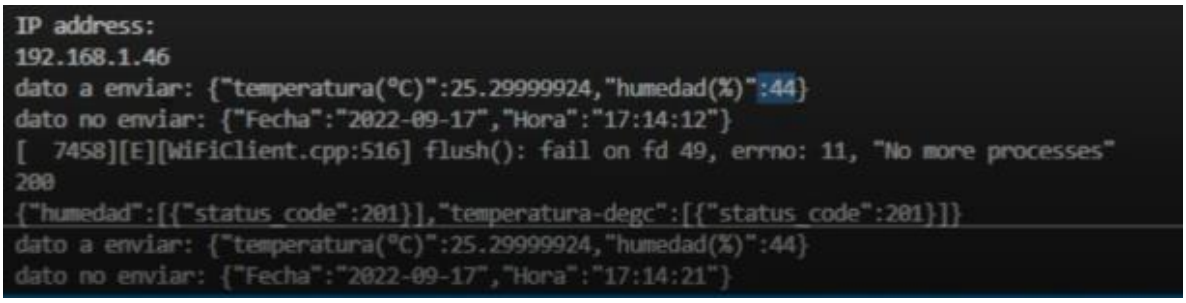

Figura 4. Serial monitor de ESP32 en POST de datos del sensor a ubidots.

Los datos de la temperatura y la humedad son visualizados en la plataforma de ubidots en el dispositivo seleccionado con el nombre de “esp32” .

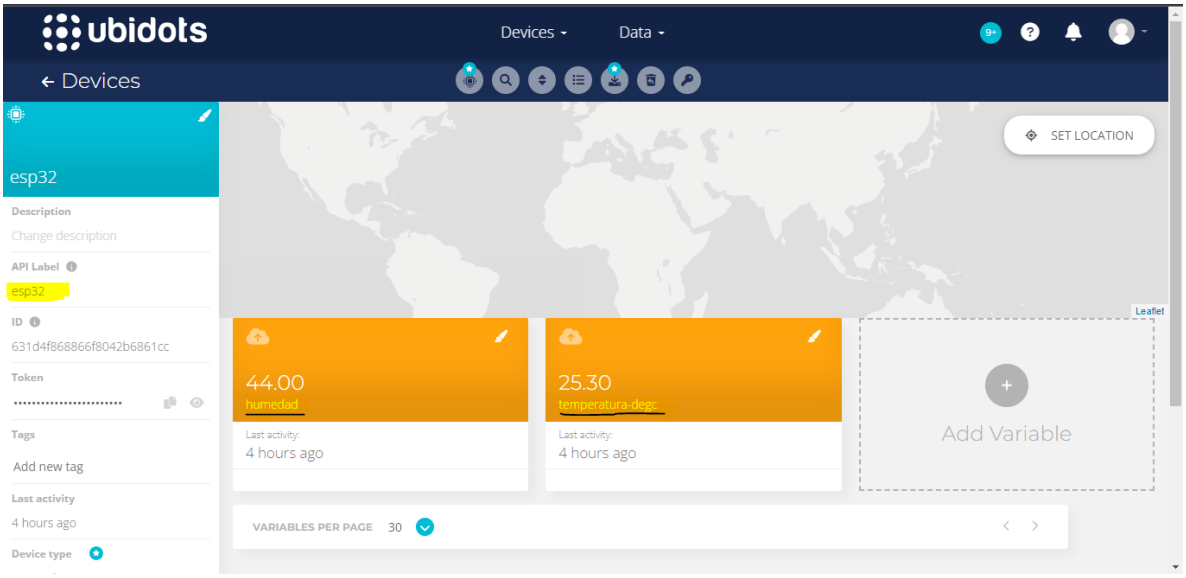

Figura 5. Visualización de datos de humedad y temperatura en ubidots.

El envío de datos es gracias al htpclient en donde en donde se envía a una url, en este caso la url esta dada por :

```
http://things.ubidots.com/api/v1.6/devices/name_device/?token=your_token_api_credentials
```

El token de la api credential esta en el perfil del usuario en donde se puede copiar y pegar en el código para que realiza el envío de los datos.

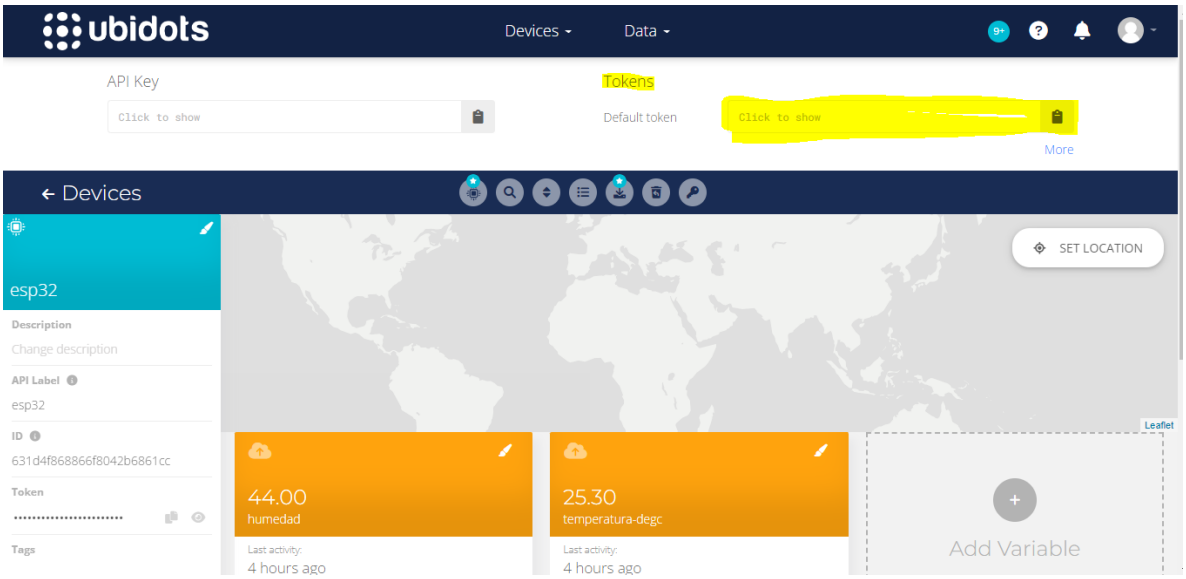

Figura 6. Token de api credentials en ubidots.

Dentro de las variables se puede examinar la fecha y hora en donde los datos han sido entregados , por esto no es necesario enviar los datos de fecha y hora ya anteriormente mencionados ; los datos de temperatura y humedad se pueden filtrar de diferentes forma de vista como raw,average,mínimum ,máximum, sum y count .

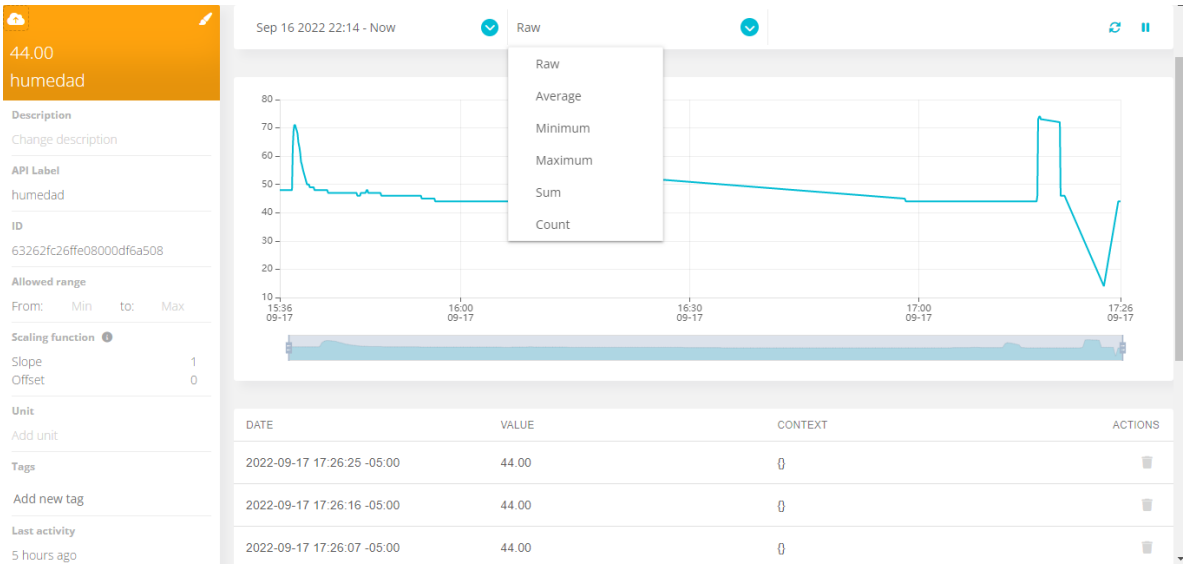

Figura 7. Visualización estadística de datos en ubidots.

Los datos también se pueden observar de una forma mas amigable , en esta se pincha en data dentro en el dashboard en donde se refleja la información de los datos de humedad y temperatura de forma más visual con iconos agradables y fondos configurables a través de css, permitiendo prestar atención rápidamente al dato y la fecha y hora en donde fue entregado.

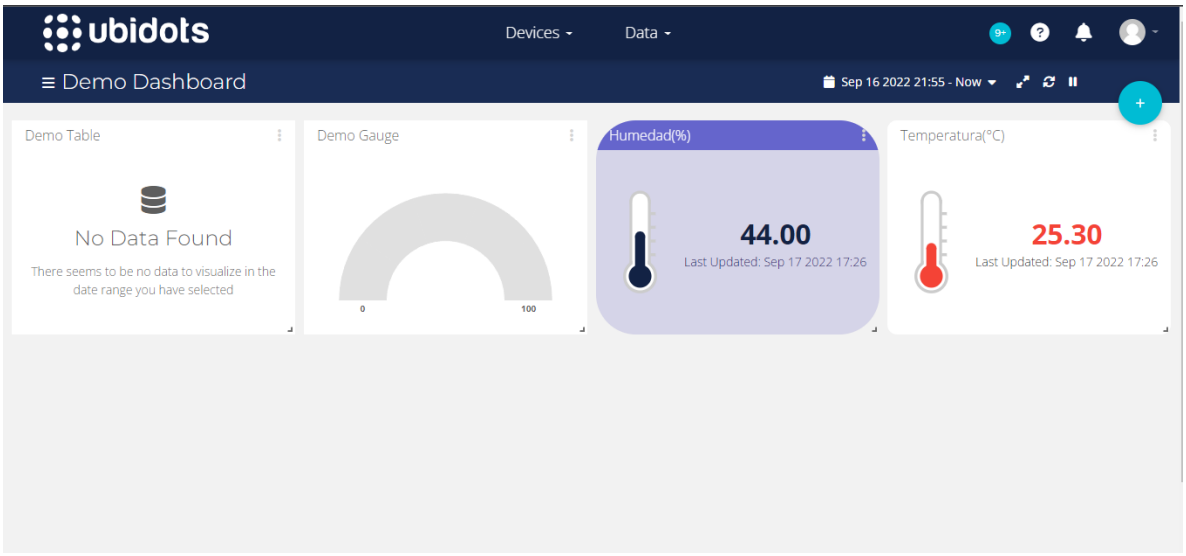

Figura 8. Visualización dashboard de datos en ubidots.

Ahora con el método Rest con el programa Postman se realizará el GET y POST de los datos.Con el método GET se utiliza la siguiente url a definir en postman , en donde solo se modifica el nombre del dispositivo , la eqtiqueta de la variable y el token.

```
http://things.ubidots.com/api/v1.6/devices/name_device
/label_variable/values?token=you_token
```

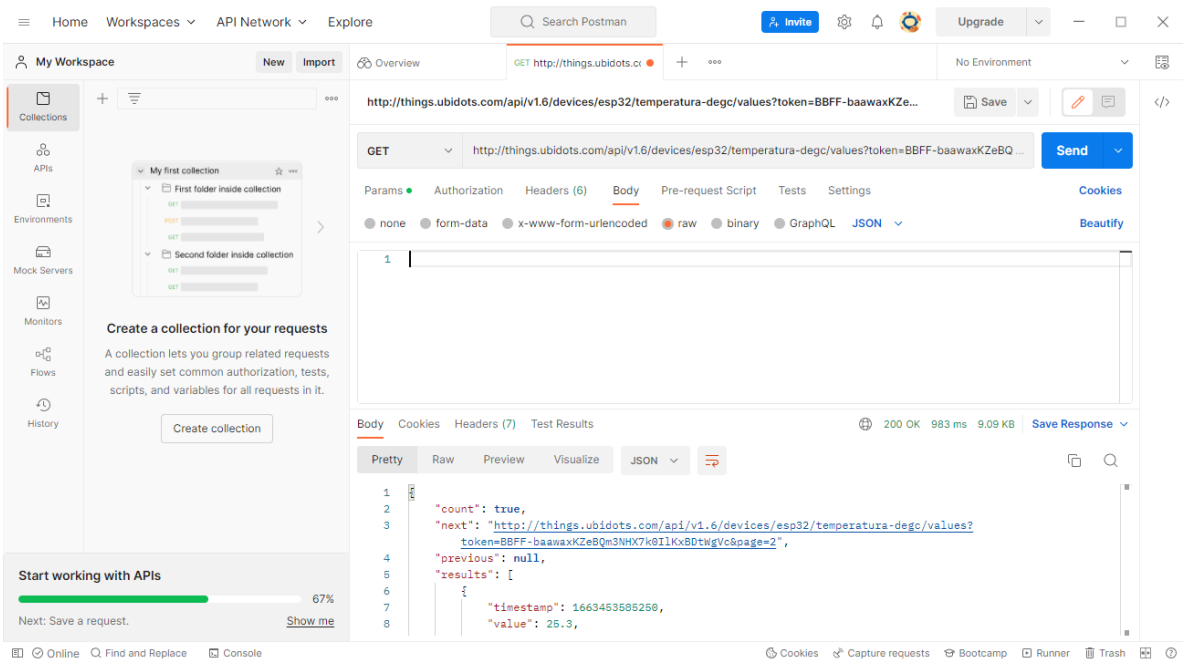

Figura 9. Visualización GET de datos temperatura de ubidots en Postman.

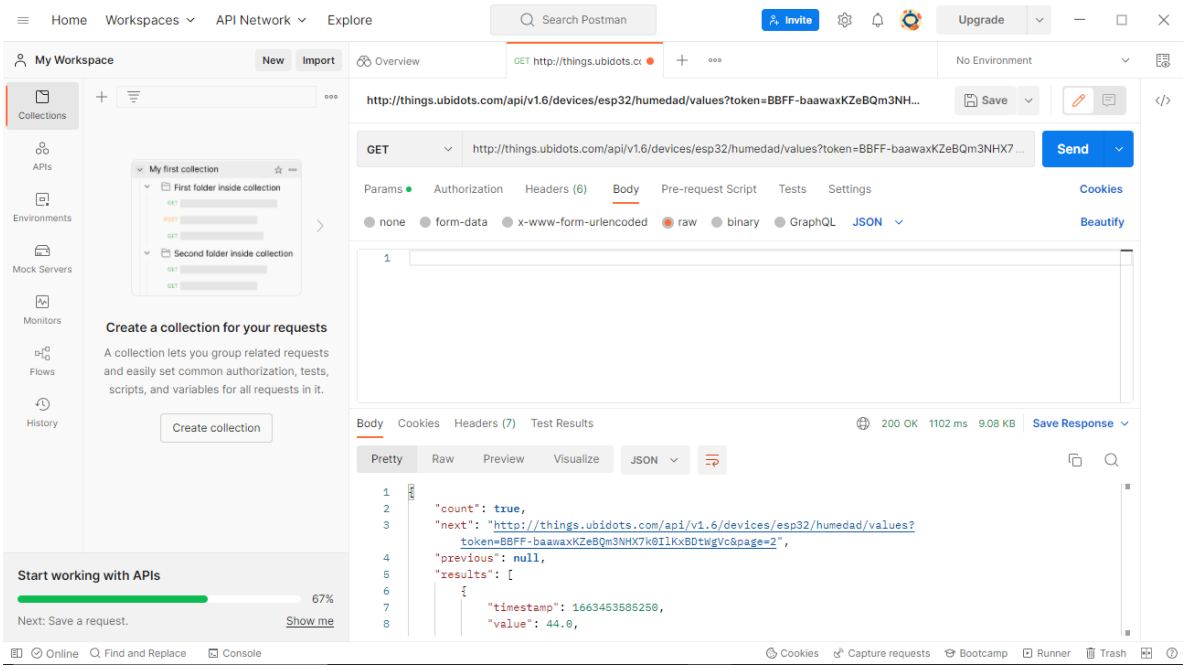

Figura 10. Visualización GET de datos humedad de ubidots en Postman.

Con el método POST se utiliza la siguiente url a definir en postman , en donde solo se modifica el nombre del dispositivo y el token.

```
http://things.ubidots.com/api/v1.6/devices/name_device/?token=you_token
```

y se escribe dentro de llaves con comillas dobles el nombre de variable a publicar con dos puntos y el dato a publicar.

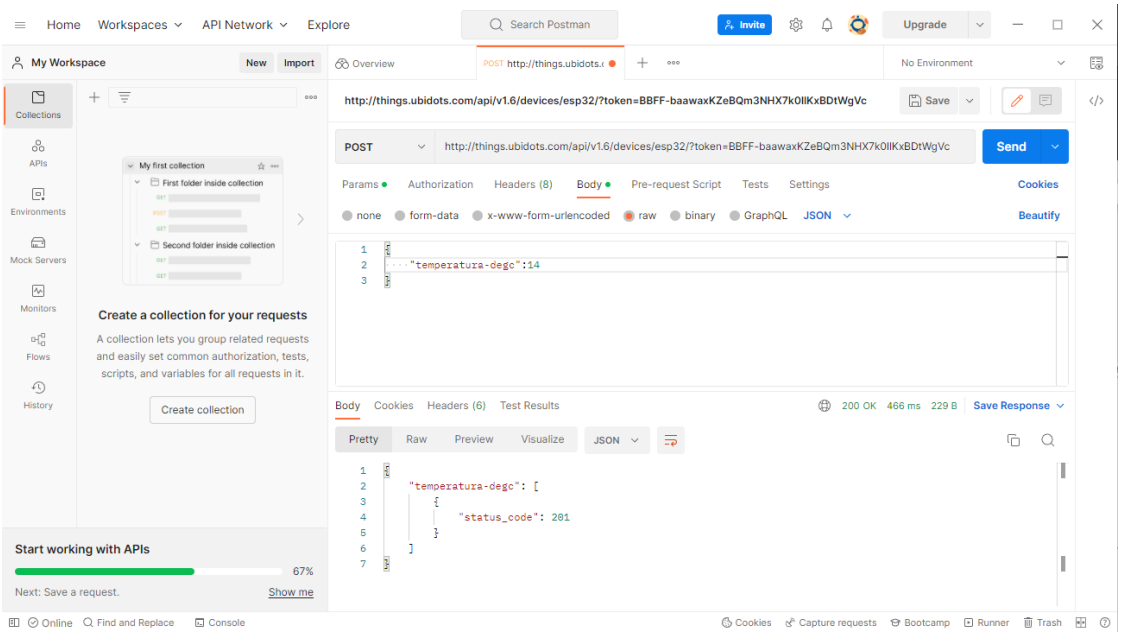

Figura 11. Visualización POST de dato temperatura de Postman en ubidots.

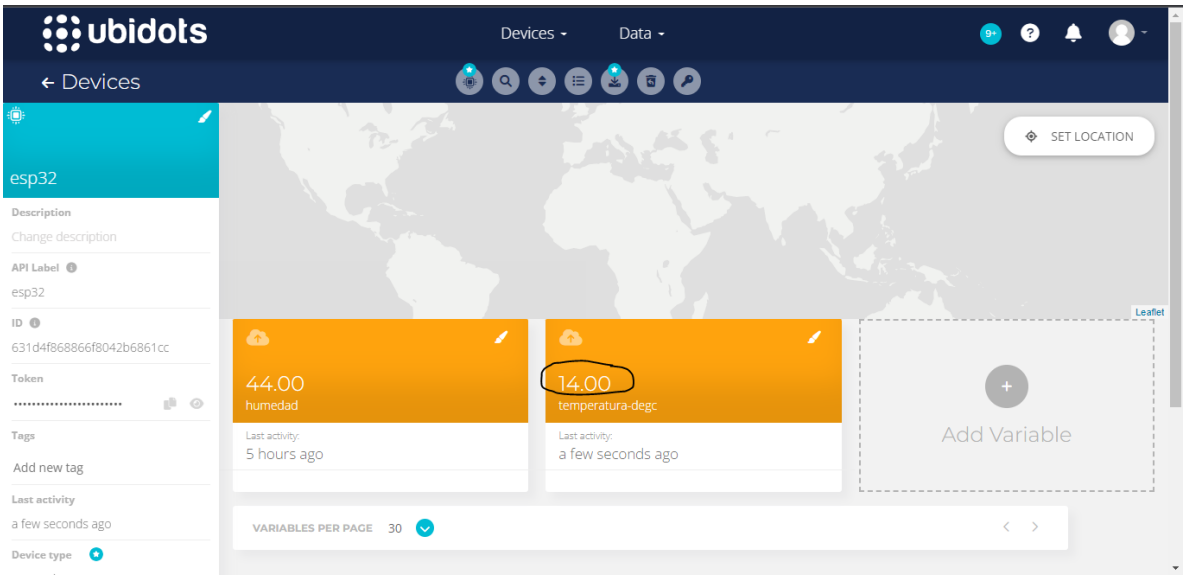

Figura 12. Visualización de publicación POST temperatura de Postman en ubidots.

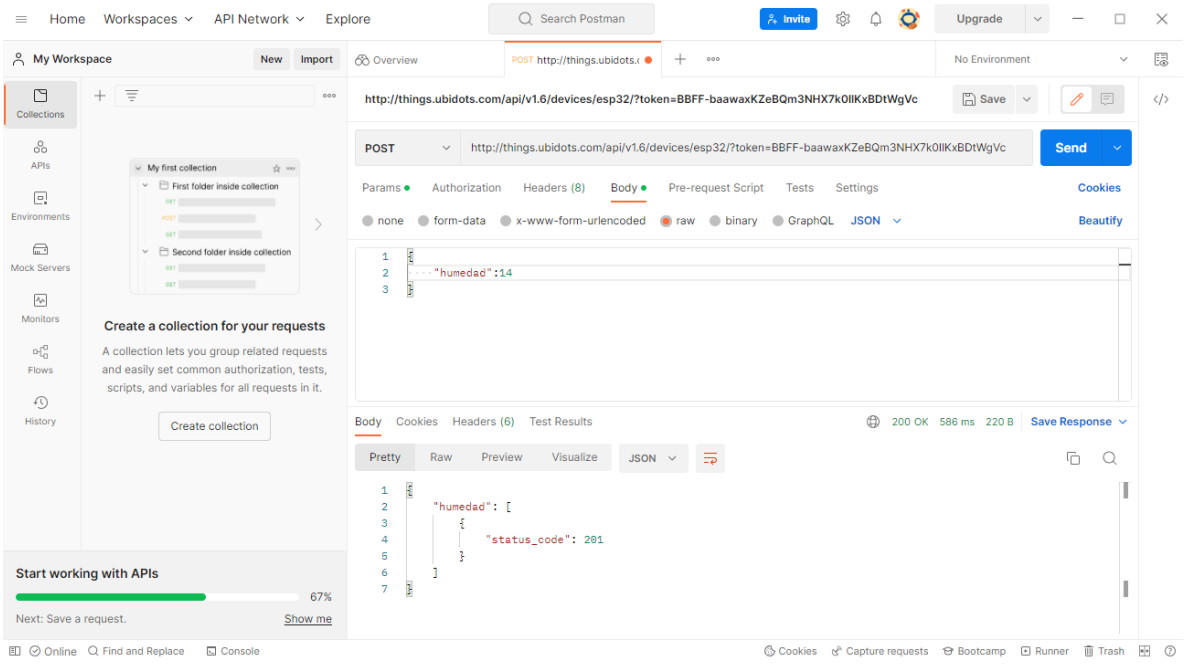

Figura 13. Visualización POST de dato humedad de Postman en ubidots.

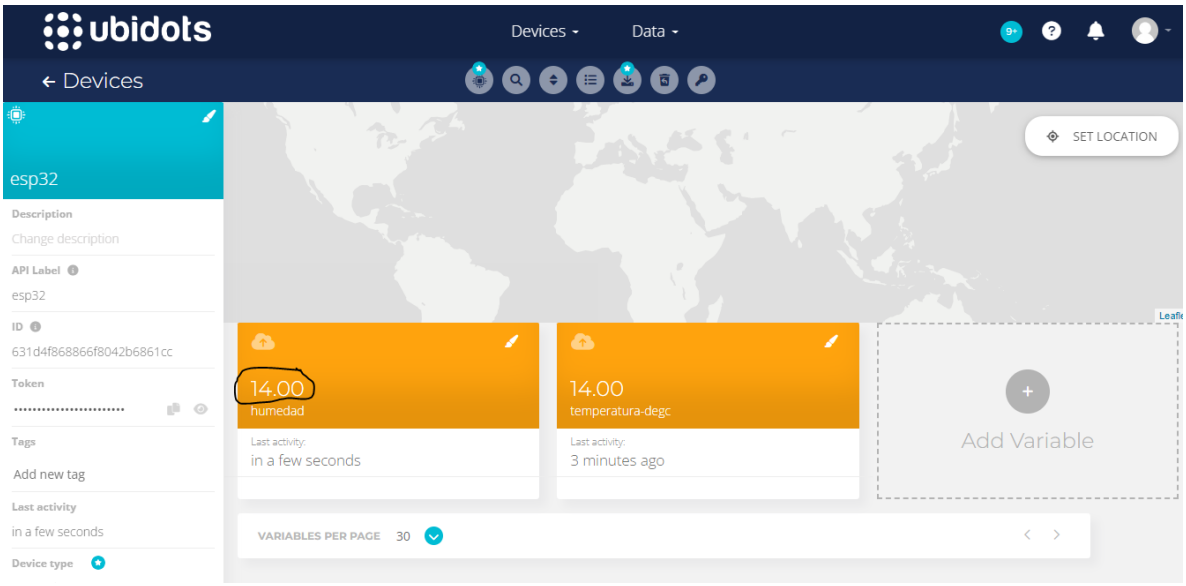

Figura 14. Visualización de publicación POST humedad de Postman en ubidots.

Referencias

[1]2022. [Online]. Available: https://ubidots.com/community/t/esp32-and-ubidots-subscribing-multiple-variables/3829

[2]2022. [Online]. Available: https://ubidots.com/community/t/esp32-and-ubidots-subscribing-multiple-variables/3829/2.

[3]"Programar ESP32 con IDE arduino", Talos Electronics, 2022. [Online]. Available: https://www.taloselectronics.com/blogs/tutoriales/programar-esp32-con-ide-arduino. [Accessed: 18-Sep- 2022]