Universidad Rafael Landívar

Facultad de Ingeniería

IoT

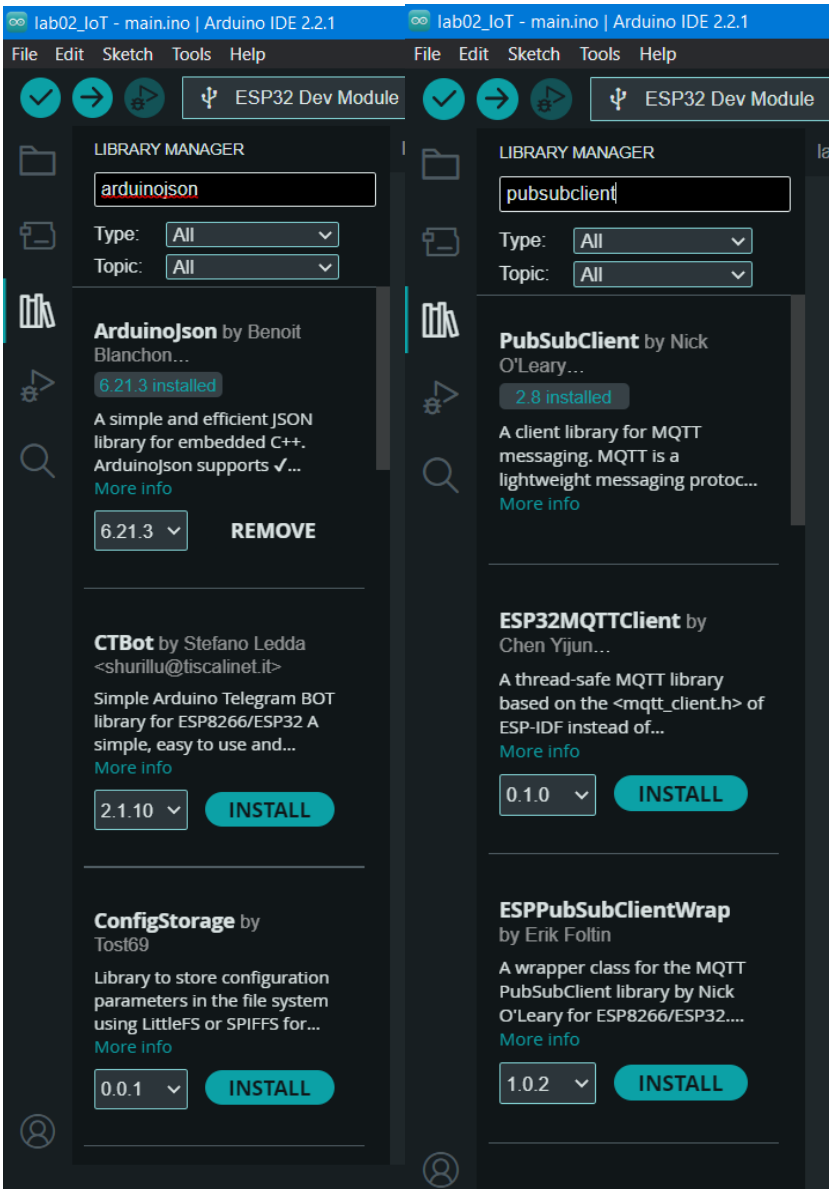
Ing. Diego Bran

Laboratorio #2

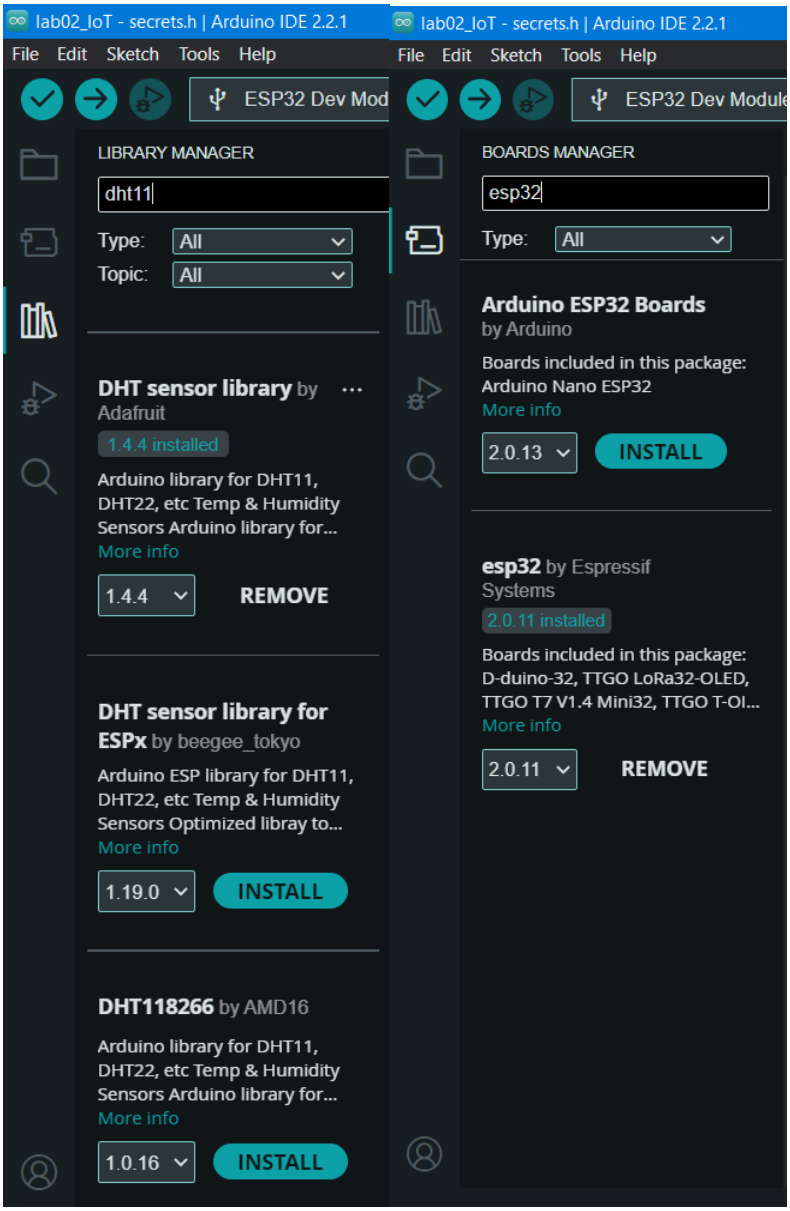
Things & Cloud

Guatemala, 29 de septiembre de 2023

- Como primer paso para el laboratorio, al abrir la IDE de Arduino, se necesitan instalar las siguientes librerías: ArduinoJSON y PubSubCLient. La primera se utiliza para poder gestionar los archivos de tipo JSON, la otra se utiliza para poder procesar y gestionar mensajes por medio del protocolo MQTT.



- La librería del sensor DHT11 también es necesaria agregarla para poder obtener información y funciones específicas para este sensor. En el apartado de Boards Manager de igual forma hay que agregar los drivers y ciertas funcionalidades para la tarjeta esp32 que se utilizará.



- Luego de agregar las librerías, se crea una nueva pestaña llamada main.ino, la cual tendrá el código para poder llevar a cabo el proceso de envío de datos de la tarjeta esp32 hacia la nube en AWS. Específicamente este archivo contiene la lógica empleada para establecer la comunicación WiFi, conectarse al broker MQTT, e interactuar por puerto serial con el Arduino.

lab02_IoT.ino | main.ino | secrets.h

```
#include "secrets.h" //I HAD PROBLEMS WITH THE POLICY OF MY "THING" IN AWS, SO TO CORRECT THIS I USED * TO BE ABLE TO COONNECT WITHOUT RESTRICTIONS.
#include <WiFiClientSecure.h>
#include <PubSubClient.h>
#include <ArduinoJson.h>
#include "WiFi.h"


#include "DHT.h"
#define DHTPIN 19   // Digital pin connected to the DHT sensor. THE DIGITAL PIN & THE PHYSICAL PIN BOTH WERE MY PROBLEMS TO CONNECT THE DHT11 SENSOR
#define DHTTYPE DHT11  // DHT 11

#define AWS_IOT_PUBLISH_TOPIC   "esp32/pub"
#define AWS_IOT_SUBSCRIBE_TOPIC "esp32/sub"

float h ;
float t;

DHT dht(DHTPIN, DHTTYPE); //ANOTHER PROBLEM THAT I HAD, IT WAS THIS LINE BECAUSE I USED "" TO DEFINE DHTPIN AND DHTTYPE SO IT WASN'T CORRECT BECASUSE THE AR

WiFiClientSecure net = WiFiClientSecure(); //Creating objects for wifi comunication
PubSubClient client(net);

void connectAWS() //Function for secure comunication between Arduino and AWS
{
  WiFi.mode(WIFI_STA);
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);

  Serial.println("Connecting to Wi-Fi");

  while (WiFi.status() != WL_CONNECTED)
  {
    delay(500);
    Serial.print(".");
  }
```

lab02_IoT.ino | main.ino | secrets.h

```
  }

  // Configure WiFiClientSecure to use the AWS IoT device credentials
  net.setCACert(AWS_CERT_CA);
  net.setCertificate(AWS_CERT_CRT);
  net.setPrivateKey(AWS_CERT_PRIVATE);

  // Connect to the MQTT broker on the AWS endpoint defined earlier
  client.setServer(AWS_IOT_ENDPOINT, 8883);

  // Create a message handler
  client.setCallback(messageHandler);

  Serial.println("Connecting to AWS IOT");

  while (!client.connect(THINGNAME))
  {
    Serial.print(".");
    delay(100);
  }

  if (!client.connected())
  {
    Serial.println("AWS IoT Timeout!");
    return;
  }

  // Subscribe to a topic
  client.subscribe(AWS_IOT_SUBSCRIBE_TOPIC);

  Serial.println("AWS IoT Connected!");
}
```

```
void publishMessage() //Function for publish the message into the topic with temperature and humidity data
{
  StaticJsonDocument<200> doc;
  doc["humidity"] = h;
  doc["temperature"] = t;
  char jsonBuffer[512];
  serializeJson(doc, jsonBuffer); // print to client

  client.publish(AWS_IOT_PUBLISH_TOPIC, jsonBuffer);
}

void messageHandler(char* topic, byte* payload, unsigned int length) //This function process the recieved messages and process the data in the MQTT message
{
  Serial.print("incoming: ");
  Serial.println(topic);

  StaticJsonDocument<200> doc;
  deserializeJson(doc, payload);
  const char* message = doc["message"];
  Serial.println(message);
}

void setup() //Functions calls
{
  Serial.begin(115200);
  connectAWS();
  dht.begin();
}
```

```
void loop() //Repeat the instruccions in it (print the data basically)
{
  h = dht.readHumidity();
  t = dht.readTemperature();


  if (isnan(h) || isnan(t) )  // Check if any reads failed and exit early (to try again).
  {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }

  Serial.print(F("Humidity: "));
  Serial.print(h);
  Serial.print(F("%  Temperature: "));
  Serial.print(t);
  Serial.println(F("°C "));

  publishMessage();
  client.loop();
  delay(1000);
}
```

- De igual forma, se procede a agregar otra pestaña denominada secrects.h la cual contiene información sensible que se utiliza en el archivo main.ino. Dentro de esto las credenciales del WiFi y los certificados del dispositivo IoT para autenticarse ante el broker MQTT de AWS.

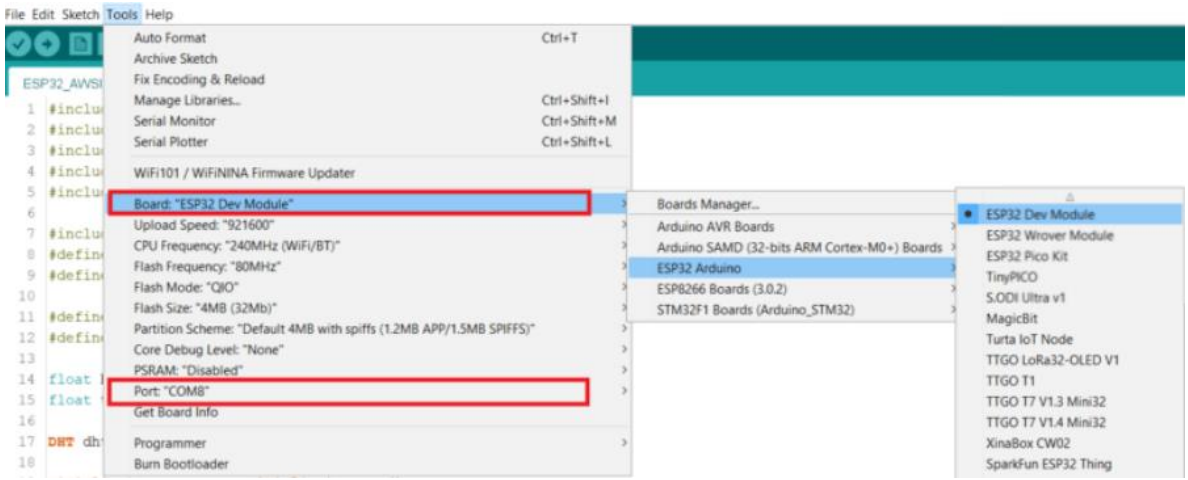
lab02_IoT.ino | main.ino | secrets.h

```
#include <pgmspace.h>

#define SECRET
#define THINGNAME "ESP32_DHT11"           //The same name of the Thing created on AWS

const char WIFI_SSID[] = "URL-WIFI";       //Defining the name of the wifi we are connected to
const char WIFI_PASSWORD[] = "";           //Establishing the password for the wifi
const char AWS_IOT_ENDPOINT[] = "af6de5brye0bj-ats.iot.us-east-2.amazonaws.com";

// Amazon Root CA 1
static const char AWS_CERT_CA[] PROGMEM = R"EOF(
-----BEGIN CERTIFICATE-----
MIIDQTCCAimgAwIBAgITBmyfz5m/jAo54vB4ikPmljZbyjANBgkqhkiG9w0BAQsF
ADA5MQswCQYDVQQGEwJVUzEPMA0GA1UEChMGQW1hem9uMRkwFwYDVQQDExBBbWF6
b24gUm9vdCBDQSAxMB4XDTE1MDUyNjAwMDAwMFoXDTM4MDExNzAwMDAwMFowOTEL
MAkGA1UEBhMCVVMxDzANBgNVBAoTBkFtYXpvbjEZMBcGA1UEAxMQQW1hem9uIFJv
b3QgQ0EgMTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBALJ4gHHKeNXj
ca9HgFB0fW7Y14h29Jlo91ghYPl0hAEvrAIthtOgQ3pOsqTQNroBvo3bSMgHFzZM
9O6II8c+6zf1tRn4SWiw3te5djgdYZ6k/oI2peVKVuRF4fn9tBb6dNqcmzU5L/qw
IFAGbHrQgLKm+a/sRxmPUDgH3KKHOVj4utWp+UhnMJbulHheb4mjUcAwhmahRWa6
VOujw5H5SNz/0egwLX0tdHA114gk957EWW67c4cX8jJGKLhD+rcdqsq08p8kDi1L
93FcXmn/6pUCyziKrlA4b9v7LWIbxcceVOF34GfID5yHI9Y/QCB/IIDEgEw+OyQm
jgSubJrIqg0CAwEAAaNCMEAwDwYDVR0TAQH/BAUwAwEB/zAOBgNVHQ8BAf8EBAMC
AYYwHQYDVR0OBBYEFIQYzIU07LwMlJQuCFmcx7IQTgoIMA0GCSqGSIb3DQEBCwUA
A4IBAQCY8jdaQZChGsV2USggNiMOruYou6r4lK5IpDB/G/wkjUu0yKGX9rbxenDI
U5PMCCjjmCXPI6T53iHTfIUJrU6adTrCC2qJeHZERxhlbI1Bjjt/msv0tadQ1wUs
N+gDS63pYaACbvXy8MWy7Vu33PqUXHeeE6V/Uq2V8viTO96LXFvKWlJbYK8U90vv
o/ufQJVtMVT8QtPHRh8jrdkPSHCa2XV4cdFyQzR1bldZwgJcJmApzyMZFo6IQ6XU
5MsI+yMRQ+hDKXJioaldXgjUkK642M4UwtBV8ob2xJNDd2ZhwLnoQdeXeGADbkpy
rqXRfboQnoZsG4q5WTP468SQvvG5
-----END CERTIFICATE-----
)EOF";
```

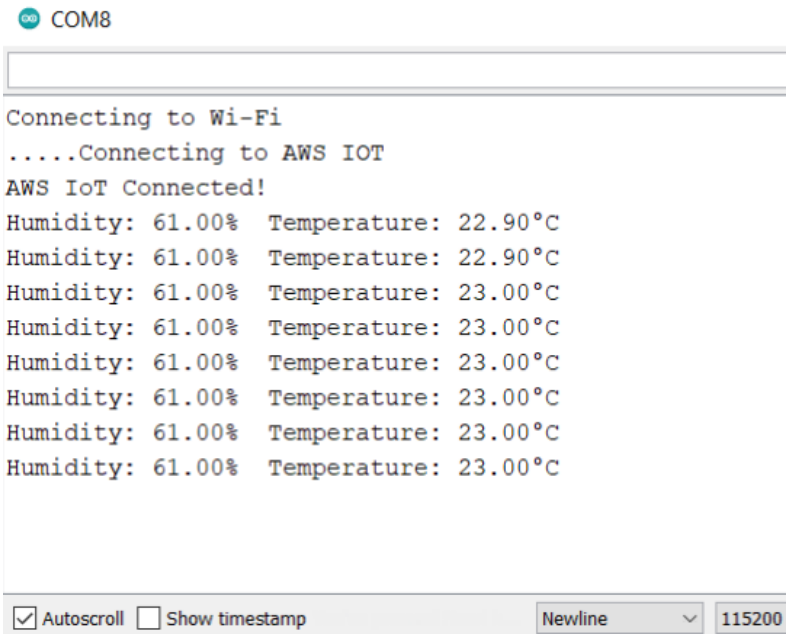
```
33
34  // Device Certificate
35  static const char AWS_CERT_CRT[] PROGMEM = R"KEY(
36  -----BEGIN CERTIFICATE-----
37  MIIDWjCCAkKgAwIBAgIVALd6v2gZwKWdIyVaZ4djm6I6UnbSMA0GCSqGSIb3DQEB
38  CwUAME0xSzBJBgNVBAsMQkFtYXpvbiBXZWIgU2VydmljZXMgTz1BbWF6b24uY29t
39  IEluYy4gTD1TZWF0dGxlIFNUPVdhc2hpbmd0b24gQz1VUzAeFw0yMzA5MjgxNzE3
40  NDZaFw00OTEyMzEyMzU5NTlaMB4xHDAaBgNVBAMME0FXUyBJb1QgQ2VydGlmaWNh
41  dGUwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQC9x+rVFbM4qgnvw1Jz
42  cCgjOtzuZCWJ45WHUPZGK+Boh2UXw8NJXhv65rHAf4nO6bIfTYOG4MlaQ2gK8zBJ
43  3QKjdXfFg8F6VZAN7iq7vRw608jgZkjQanIEYyaAhkR0lcz3WhtTovBEalrjOYgi
44  fAi9sMrN7G1vRYCnpZKZfZbI91jJhO10wUNkVRVCyMFdziKPbDyoz5LjN6LULlc3
45  lubvz5o5QyTTcYOmM3RmZdqiWCe05CWG2bBEMHGfQPzXFr4tksGSrRwkeYSKVD3d
46  lMDPiLR3LCvGzlJFV5UHbbXOctvP5av/KLOWq3X4rkQcj9uZhu7pjuvmJAaT84cY
47  K2LjAgMBAAGjYDBeMB8GA1UdIwQYMBaAFCYvcdgFFdYcTxd329jkeNvSDIImMB0G
48  A1UdDgQWBBQg+ywA0zZV5HB9rZ8M3IAu4rhMCDAMBgNVHRMBAf8EAjAAMA4GA1Ud
49  DwEB/wQEAwIHgDANBgkqhkiG9w0BAQsFAAOCAQEAhWiv+Lsru/epHmeM+CzdSnCK
50  0boDiDNxixMnDfSG303YLse/Zua+5mcl7Nd1FsXLahd9u0u4bWRwnku/NEEX7cNW
51  wULl1KA6prXAv09LPkvKTFfQGgsXAL/9n6xFmZU1GKP0wHBuPqDbcEzLpLmzQ9F2
52  8rzPxMYZiXOaHa3jX8lupwrZgUaJuNETFEgqY9szww+yD0WIx6ZnbXyErKh8+Dgk
53  RmVlzpY3gkD7fltCq7SzN1xAdH4IhDb+2QTOTZU+d6rAYLWoq+POLRUZQMEzZAAp
54  ZIfWFAQ+/c+4nvw7j3ufKsNikmEohKKrO2apncxN/9eKtwGxgERN2Q8XNc33ww==
55  -----END CERTIFICATE-----
56
```

```
59
60  // Device Private Key
61  static const char AWS_CERT_PRIVATE[] PROGMEM = R"KEY(
62  -----BEGIN RSA PRIVATE KEY-----
63  MIIEpQIBAAKCAQEAvcfq1RWzOKoJ78NSc3AoIzrc7mQlieOVh1D2RivgaIdlF8PD
64  SV4b+uaxwH+JzumyH02DhuDJWkNoCvMwSd0Co3V3xYPBelWQDe4qu70cOtPI4GZI
65  0GpyBGMmgIZEdJXM91obU6LwRGpa4zmIInwIvbDKzextb0WAp6WSmX2WyPdYyYTt
66  dMFDZFUVQsjBXc4ij2w8qM+S4zei1C5XN5bm78+aOUMk03GDpjN0ZmXaolgntOQl
67  htmwRDBxn0D81xa+LZLBkq0cJHmEilQ93ZTAz4i0dywrxs5SRVeVB221znLbz+Wr
68  /yizlqt1+K5EHI/bmYbu6Y7r5iQGk/OHGCti4wIDAQABAoIBACvRqSsiI4uF112A
69  39y/nMTlpnV0siYwYiooS3TQVLRN7s2Z6dFa5WL9OTWVPSCFXRdVMnVGGbe5sKA2
70  noOvgBh12v3D+jxg3S4aO8u1UntSwyLs+wsESOcybYVnew7gN781EvIwwQFK/UlR
71  01kDApGp7Gm3mV0D0zziQgMmuFYWQHq4VWzY7NGiiaa2jr5M8IZyecJkaX7H6XO2
72  m3MusbXumSYa1ne0YSSEa3IxM3PA1L39PAgO1WQIlfP7b/aHvKr38+EG1tt6ZeSp
73  FuuSpyD7YK3MVbhwdiKdguxhgf+c+h5H8eV5FZgQF55JudcBE03ZFtykXePeOwRu
74  PoMqtgECgYEA6T5hQfEepns5LWVFonWHhS1b0NtS7DFD7fUCTcH8PdM0C/7TCxLj
75  JJvqxjKble0TGg1jf44ymgxj6fT+TBDNIhpYeI9Qs453MpIQ6TE9ml9XeOy2zo/c
76  +f3y4WfOcoJHd1VYvwLENuJXxCbNXTCJjiJznlwCl83xVr75ougF/0ECgYEA0Ev9
77  BvxSH6z10aXiCeotDhDgKwbV1/kvl3DvSEut7oghw1aW3dH3legEEAK3stxPq1kb
78  G36GgzANDfuT5ZeqzIyv2Xl4BTSb6RhsjLGXmFXCFqPiupZ1H+ym/kzrsOfl7r5S
79  dHCIklb62QdiOC7ESL8SJvk53Lf3skpukwbHPSMCgYEAxskqfNfyhnL43S61tWsz
80  kBaDbTKvrquiOHkbeIf+4fZSInMRDRwQyM+1vSu+nH3MTOTiSwE3MUGkGe56mPus
81  yQbpJf8HCbE+97Ozg/s6hIoDdAVTHCNScyGzUdZ2BfPw0szDcS/7aMdpg7yKRIUs
82  hx3nqizC7PxRVtw2yFe1XMECgYEAwFaqmpy7qIwpoSEwDD+0WtxKYvC43ek3TPv3
83  uCZJwYzqwFlgjguoN0A7IB2iSwKsfmiVY1n+E313UIYSF1tpV+BBMOu9K/FcVm07
84  Lt9+QChQRF+oqOrt1EDOItwcdwprg30y3lpbgeXXVDhBNefZT+M/e8V+7JTjoDqi
85  yBjmOUkCgYEAvu0TckEdUAciMB6FfR/+ISbE7bTtZIu3xJQjvc47b6wMIE7xCyYG
86  Y/rw8F9sD5SbeXZ0I/yHNyzAkBu2Jd8xJTRZpf8lhzydG2bvUwoNWqrIOuLG+r/w
87  Y99vf3I5+F1NW+NeNiAqVd1SUq8q4dLx5FabK4IUdYy10gK7zCgKvJs=
88  -----END RSA PRIVATE KEY-----
89
90
91  )KEY";
```

- Al haber realiza las modificaciones necesarias del código anterior, estableciendo la red en la que el dispositivo se encuentra conectado, la contraseña, el número de puerto que se utilizará, se procede a conectar el módulo esp32 a la pc seleccionándola en el apartado de board y seleccionar en el puerto que está conectada en el apartado de port.



- Al verificar el código y al cargárselo al módulo, este empieza a recibir los datos de parte del sensor y a enviarlos a la nube en AWS que se pueden monitorear en el Serial Monitor.



- Como verificación, desde AWS se puede suscribir a un tema el cual será el esp32/pub definido en el código y, se puede apreciar que desde acá también se pueden recibir los mensajes proporcionados por el módulo esp32.