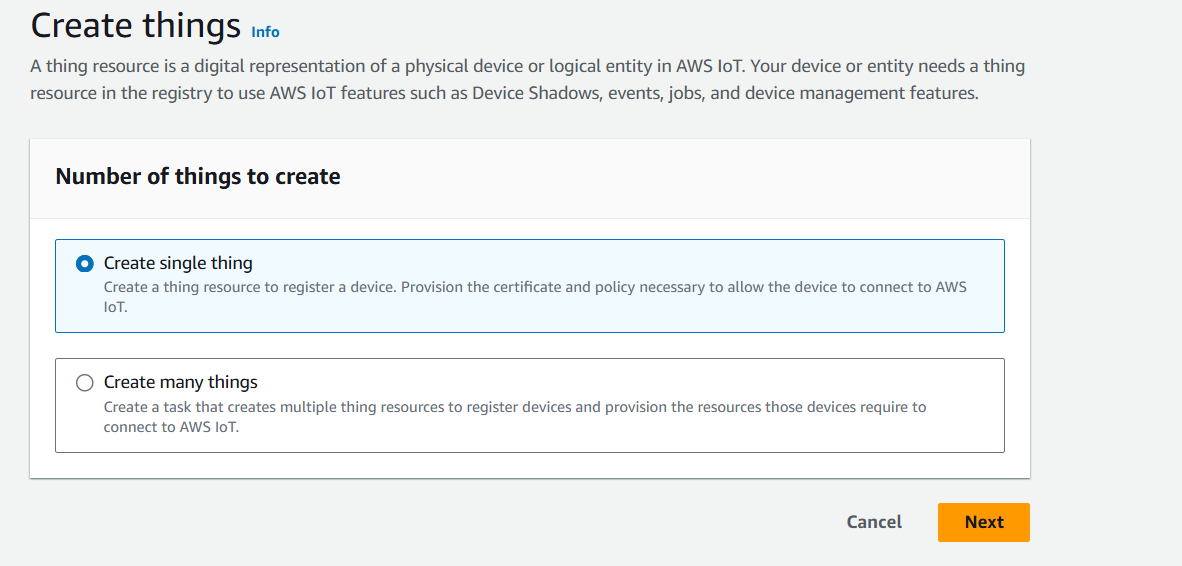
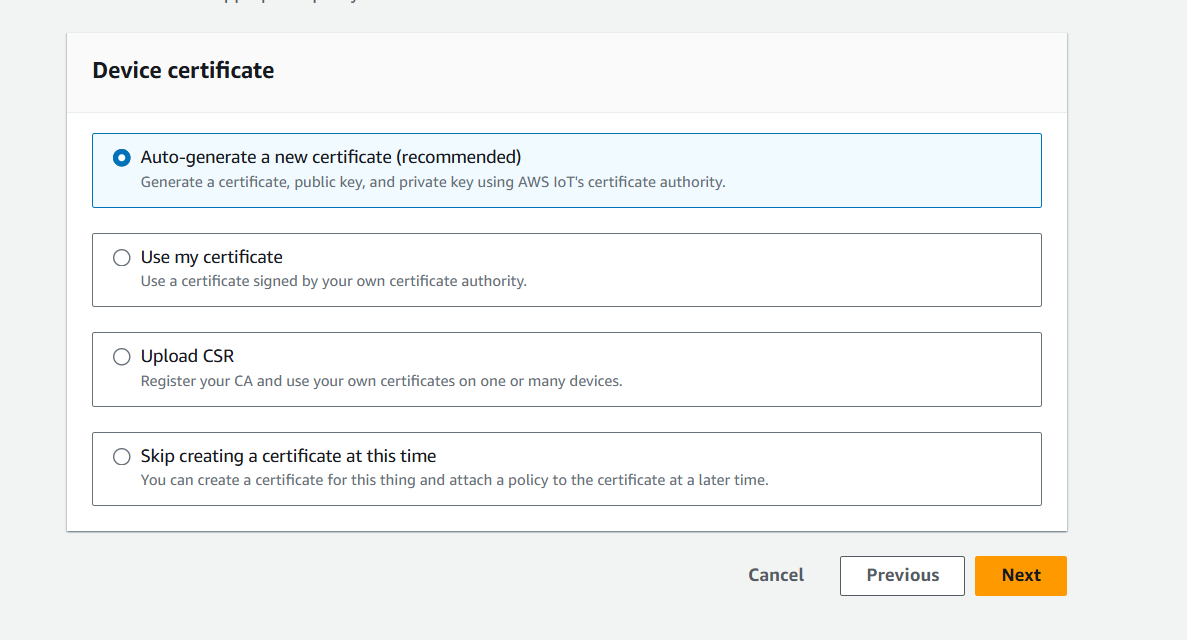
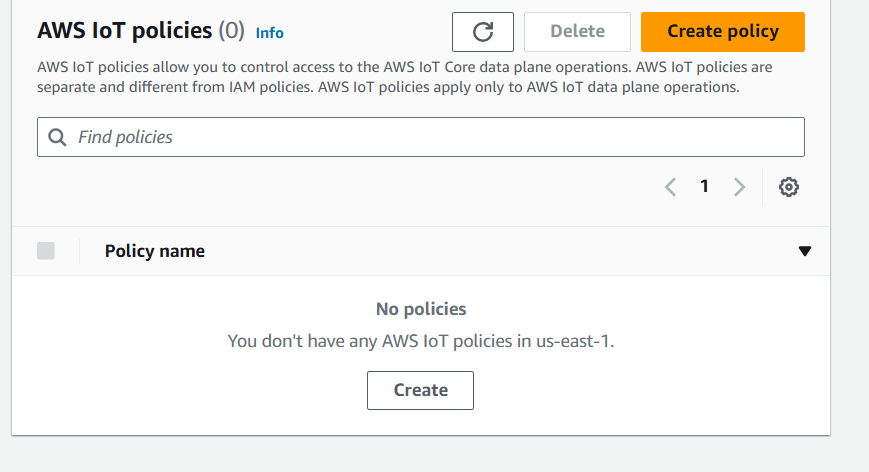
* teniendo las conecciones el NUID y WIFI haremos un obejeto del cual usaremos la herramien IOT de AWS el cual deberemos crear como thing o “objeto” para empezar el laboratorio:

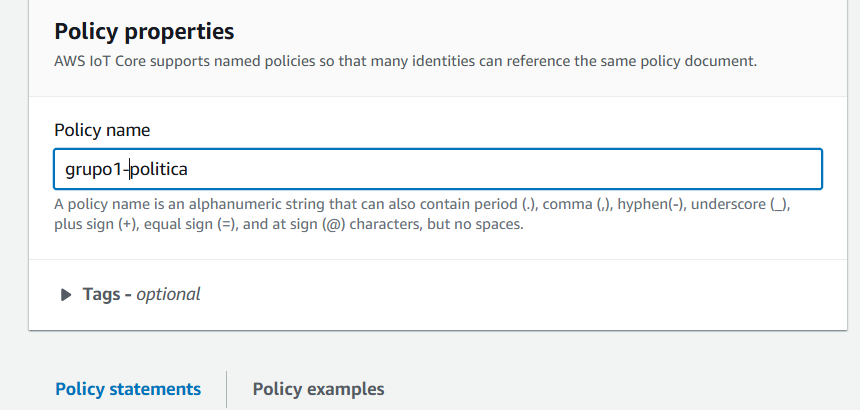


ahora testo: 

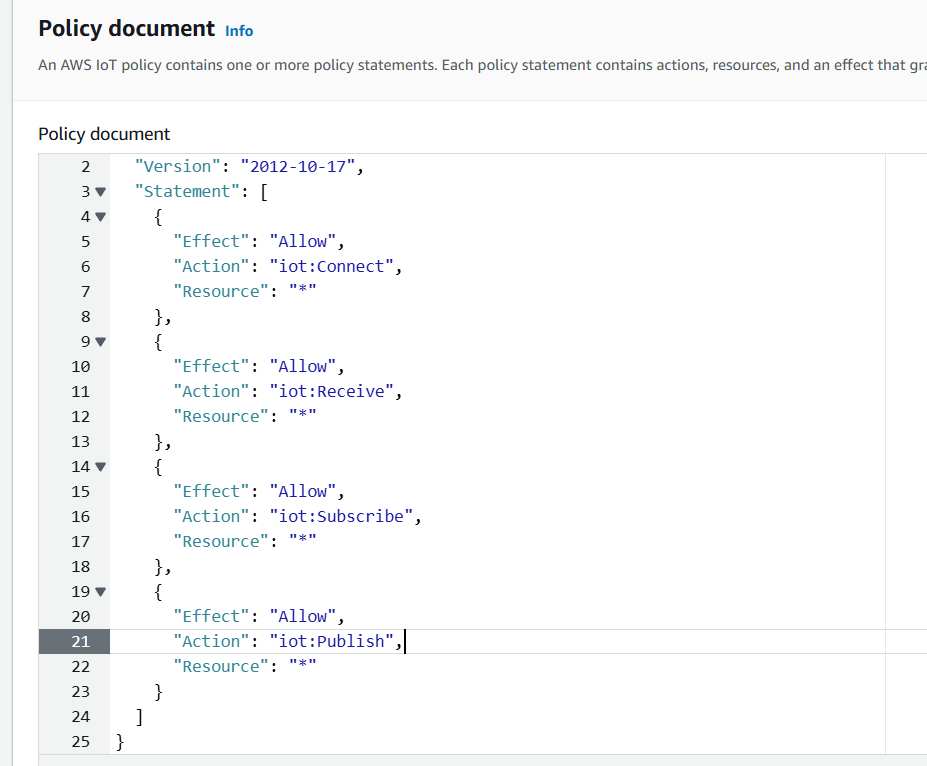
luego voy a crear una politicapara el esp32:



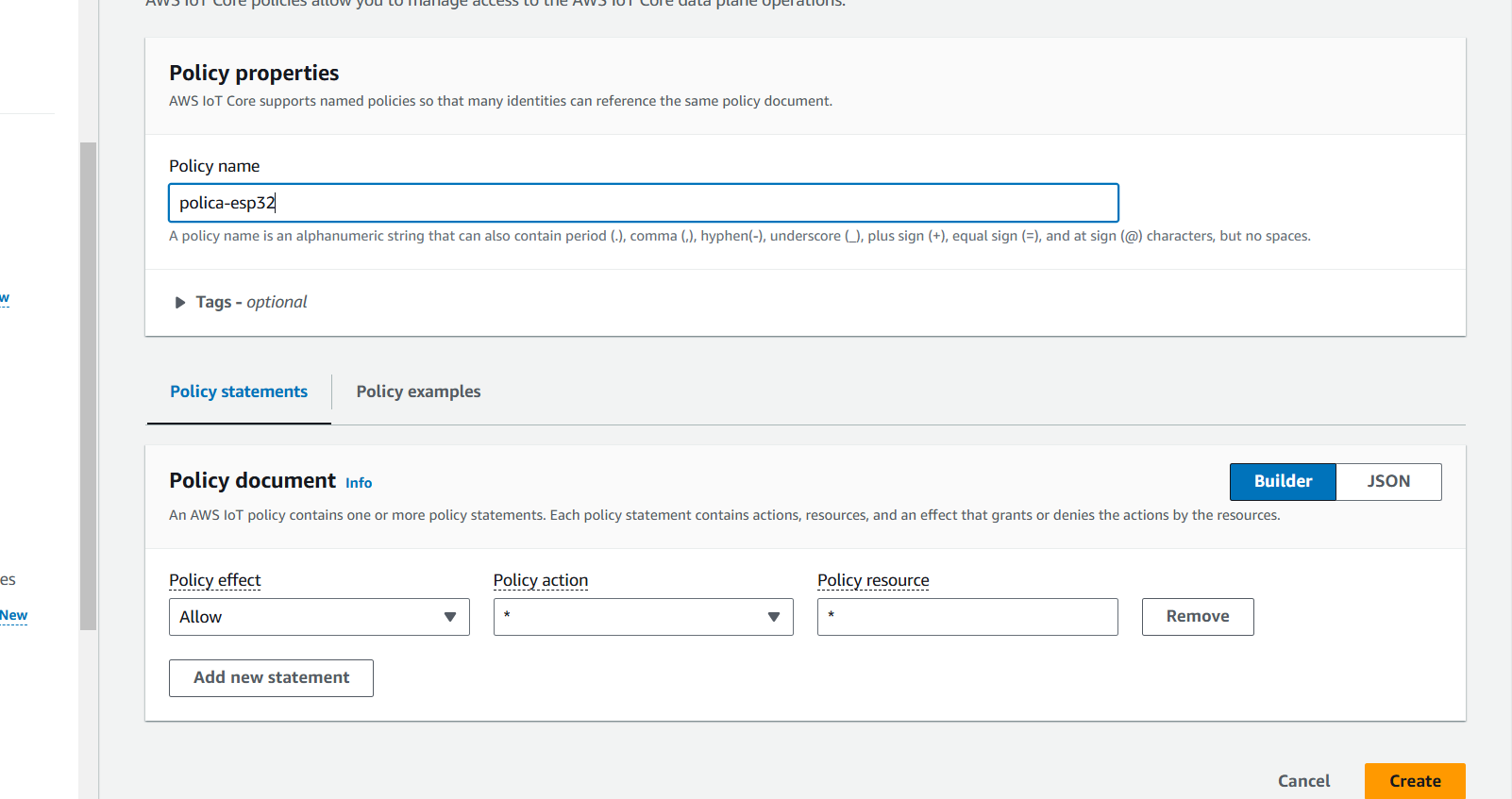
ahora creo la politica cpn el siguiente id:



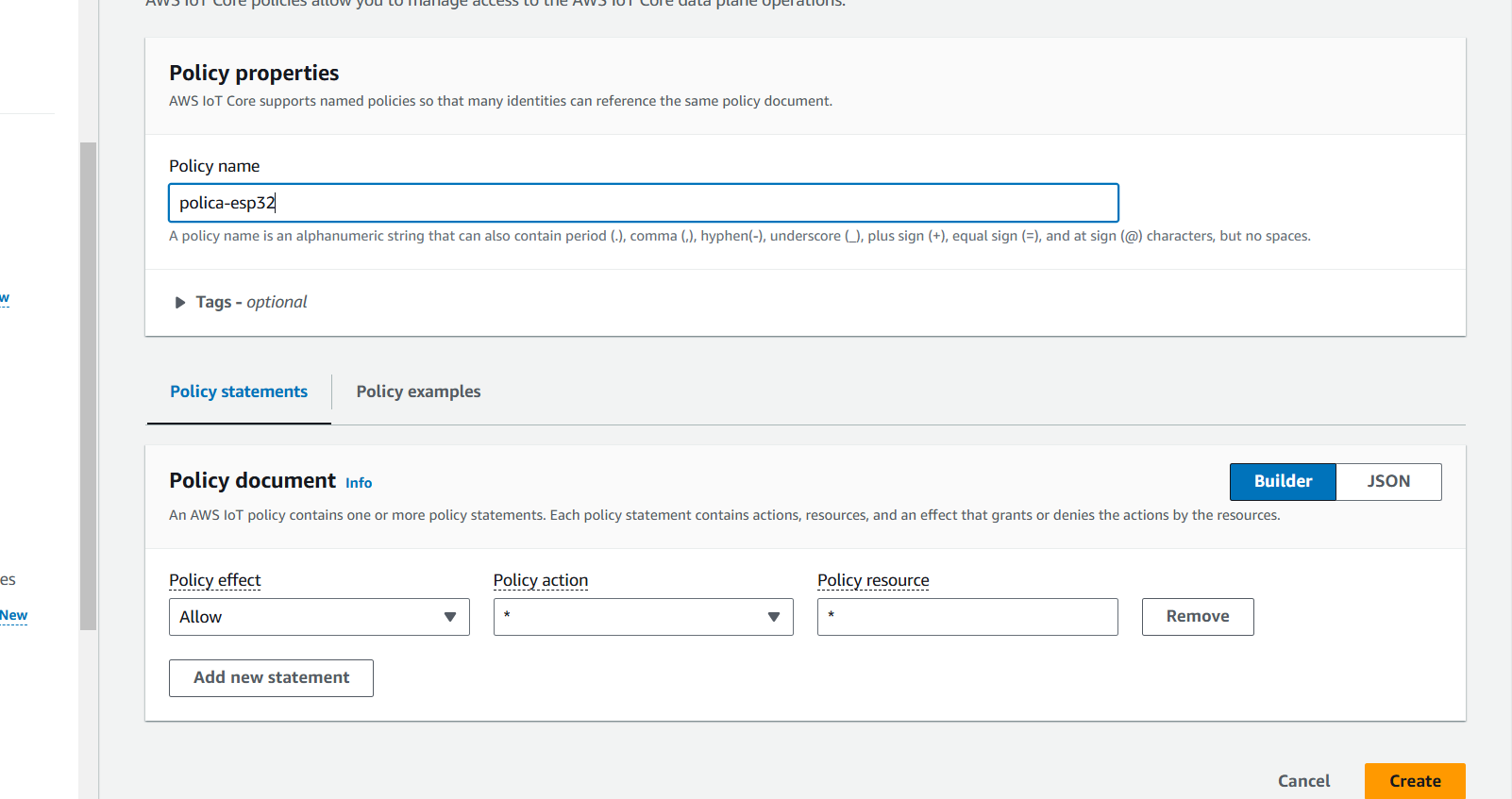
ahora creo la política copio y pego el code de la red y sigo esto JSON:

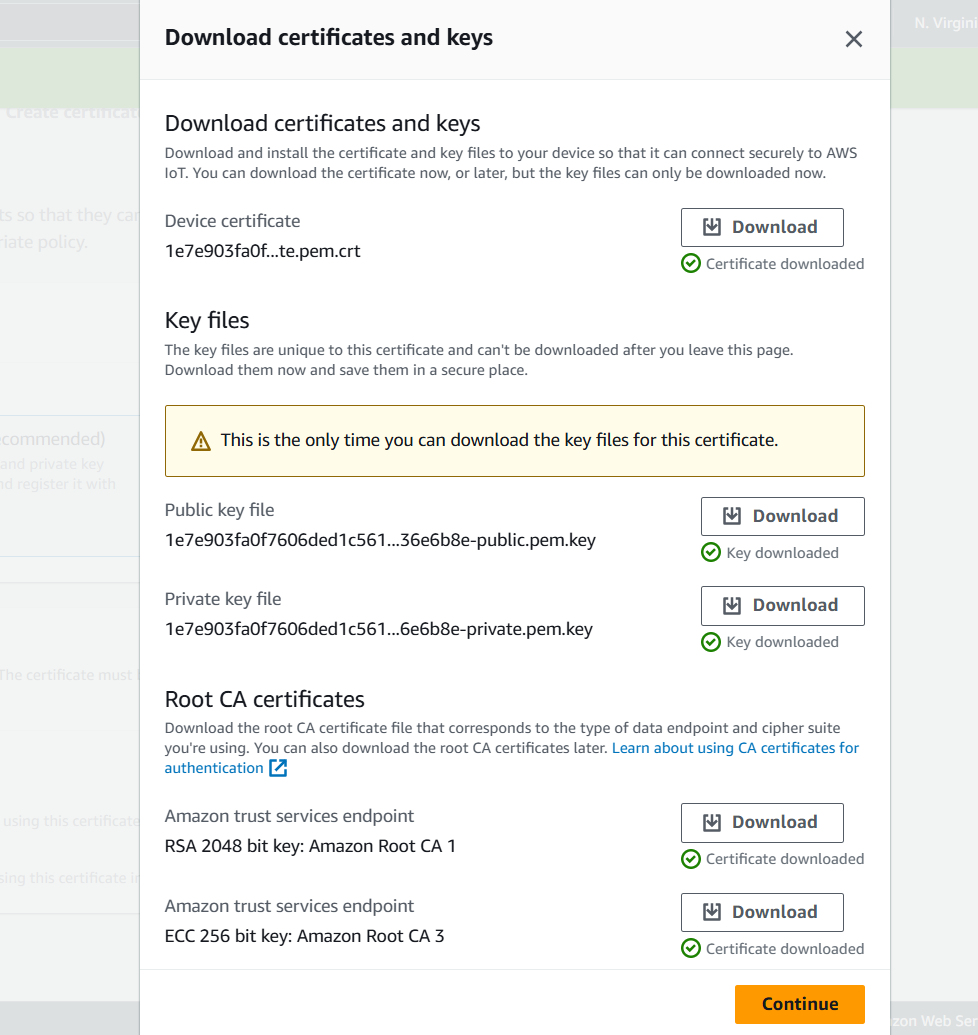


tambien si falla el json puede agregar la siguiente politicaen el thing;

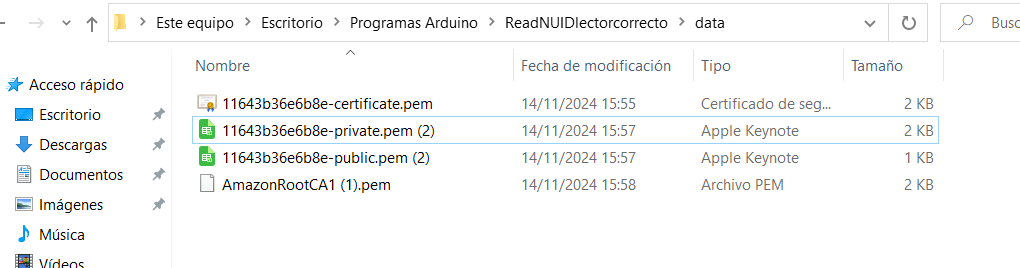


despues en certificado damos en crear certificados y después simplemente lo que vamos a hacer es descargar \*\*\*\*\*\*\* esto es en caso de que el json falle\*\* pero esto se sabra al momento de cargar el codigo en la placa esp32 y que esta se conecte y tome las lecturas:

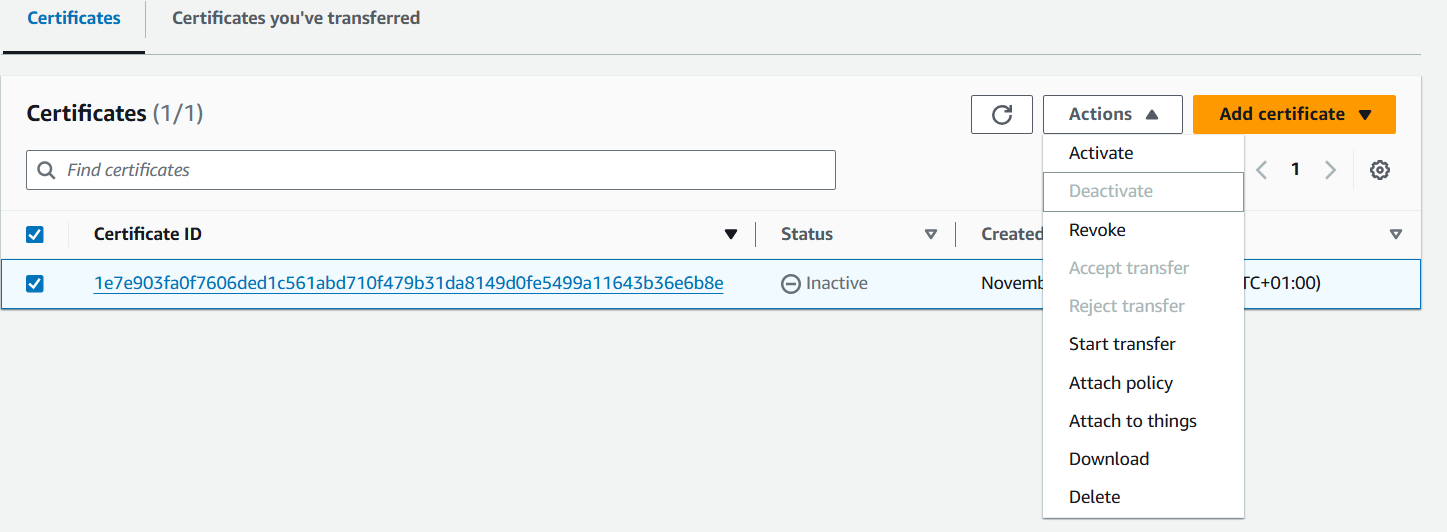


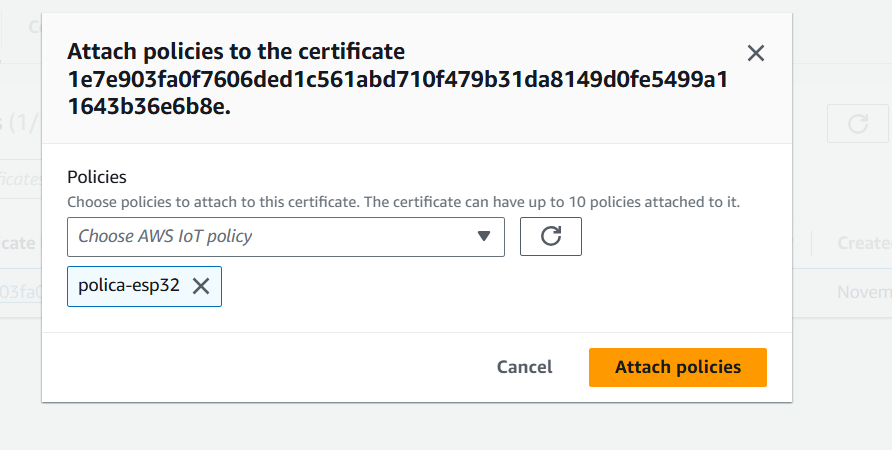
ahora esto: 

debería ser algo asi:

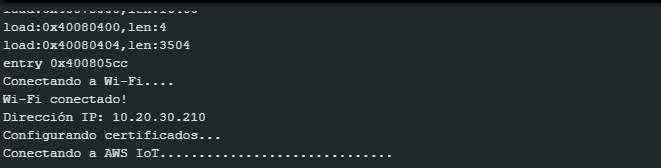


es hora de activar el certificado:



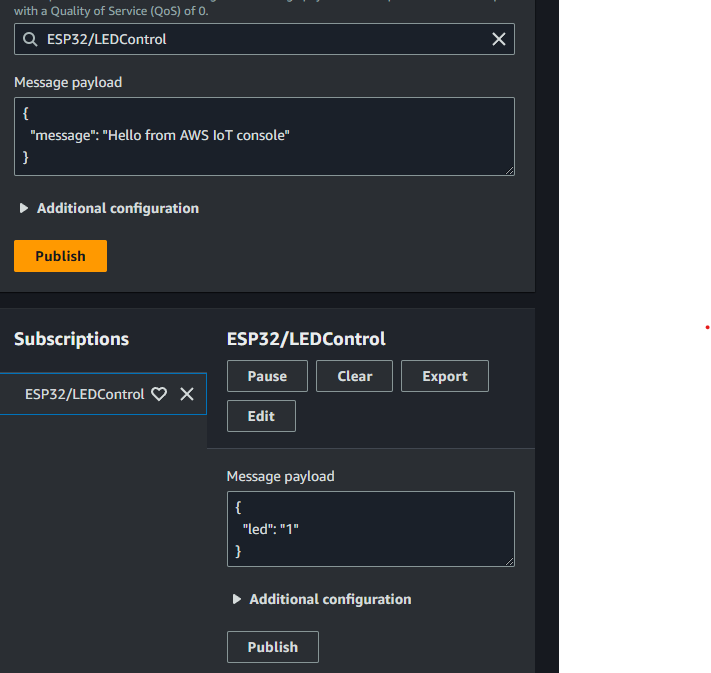
damos en attach polict y escojo esta de las cuales ya e creado y va lidenado con lo que e resuelto: 

después se agrega el código se deberá agregar las bibliotecas necesarias para ya despues tener la coneccion del esp32 al wifi usando los certificados que previamente se han descargado y generado todo ello desde el arduino ide lo cual arroja el siguiente resultado:

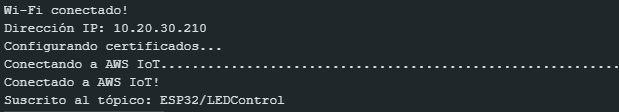


después de cargar el mqtt algo así deberá quedar:

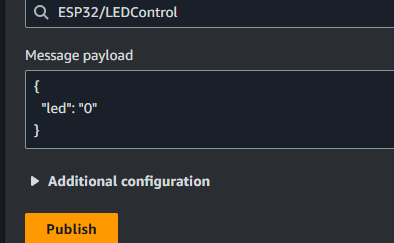
IMPORTANTE DEFINIR ENTRADA EN CÓDIGO TAL CUAL PARA QUE LA LECTURA SE HAGA CORRECTAMENTE



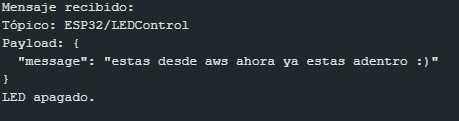
después se verifica y ahora si se lanza el programa

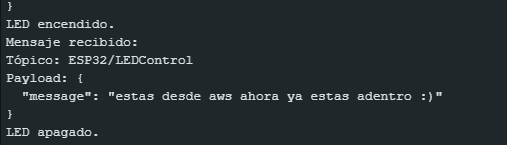


si queremos apagar simplemente deberemos hacer lo siguiente:

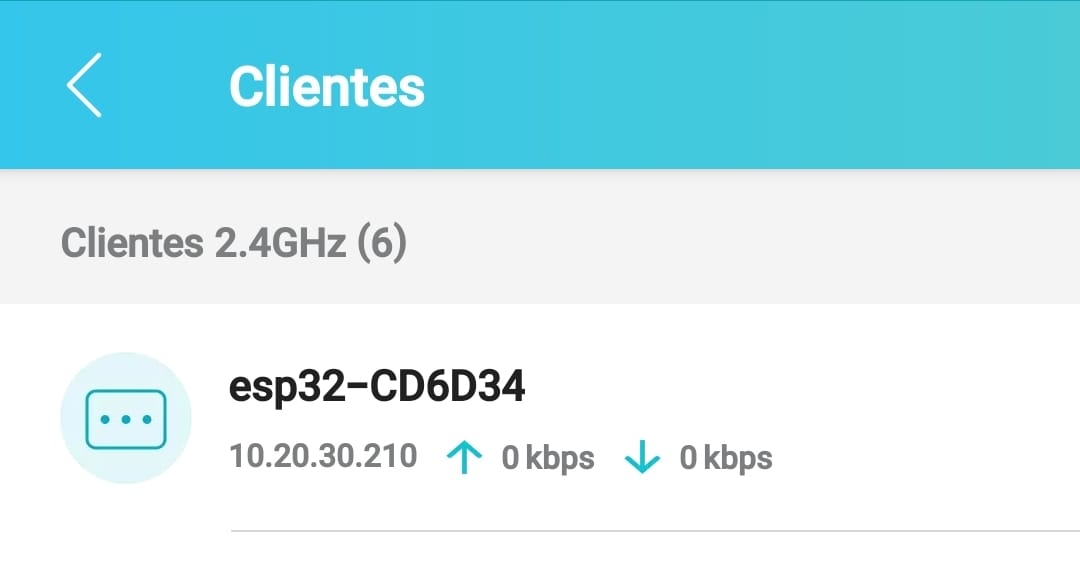


RESULTADO FINAL





red comprobacion:



CODIGO:

#include <WiFi.h>

#include <WiFiClientSecure.h>

#include <MQTTClient.h>

#include <ArduinoJson.h>

// Configuración Wi-Fi

const char WIFI\_SSID[] = "TP-Link\_62C2\_4G"; // Nombre de tu red Wi-Fi

const char WIFI\_PASSWORD[] = "RDsamana\*12"; // Contraseña de tu red Wi-Fi

// Configuración de AWS IoT

const char AWS\_IOT\_ENDPOINT[] = "a24nifur910c6a-ats.iot.us-east-1.amazonaws.com"; // Reemplaza con tu endpoint de AWS IoT

const char THINGNAME[] = "ESP32LED";

const char AWS\_IOT\_SUBSCRIBE\_TOPIC[] = "ESP32/LEDControl";

// Certificados

const char CA\_CERT[] 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";

const char DEVICE\_CERT[] PROGMEM = R"KEY(

-----BEGIN CERTIFICATE-----

MIIDWTCCAkGgAwIBAgIUMboMuS4KdsjrgSBH6QeyL69uteQwDQYJKoZIhvcNAQEL

BQAwTTFLMEkGA1UECwxCQW1hem9uIFdlYiBTZXJ2aWNlcyBPPUFtYXpvbi5jb20g

SW5jLiBMPVNlYXR0bGUgU1Q9V2FzaGluZ3RvbiBDPVVTMB4XDTI0MTExNTE3NDQx

OVoXDTQ5MTIzMTIzNTk1OVowHjEcMBoGA1UEAwwTQVdTIElvVCBDZXJ0aWZpY2F0

ZTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBALF5HsDBfNXqG5bhpxPz

KCu4Ua6xUS4HWAPl5pDZkNzldhxTOIHMmqEnYZTeHI/1WPWNpUadDcGRdumeoBtU

8WSWW23WXPh0Unp7ovQv5mBmwqu37czaCPrDtBhZRcmL0VHKAPBUi63+ADAAWnFW

FXvt+h2Ef1qWL182TZ5P52H2+8WU+FxMszNlcFavUYgKE9PoNW1CJ4QfSsRYrnLl

+2hqOl51uSGX3B70Be8werK42b3uiIsOX2G4IAKKJVZbzUhNfn+IFBd6ahLshZhT

7ADXPhlgg9+G1usnR2/hZPNDsYwJHoNbB8Tr/sECd+rCy2yADmMeV3n+dzIS6tcB

3CcCAwEAAaNgMF4wHwYDVR0jBBgwFoAU/bLhk30CndKYjLrMdrqpu9HiKzswHQYD

VR0OBBYEFEdqcy5zMEJDf8ZTeJZp0p8QwhmdMAwGA1UdEwEB/wQCMAAwDgYDVR0P

AQH/BAQDAgeAMA0GCSqGSIb3DQEBCwUAA4IBAQCk4GTiALFik3zJNOST/2BP2PKU

Wr/HWZMEUu6Ei3kaLFmyU54+oxcDpg/ApEsuWFn+glu57PFhqxzN0acDV+Vy30xv

Y0Uge1SsMClzVUmfbtEnOpwid776nXjAO5ZAGBfEDIocY/jXisQ9cahoNU0zqV8Q

3OpvUoSqYj6a7Vp1TWuslfxiJU7QzzrYIEXcNVu5Ooa7XzB2lZ3ef4ejmglaRwWq

SOQbl7ILY8B3mJSAnedVhKDmb6rWlO239SrG/UihI52HyVm1a7B/gTtApf7HFX/m

OvGLd6SXYPH46JDUjom/GhO+C4EC1QZGC2xJFBmpd7CEVVzoPm9jSqUjK+5N

-----END CERTIFICATE-----

)KEY";

const char PRIVATE\_KEY[] PROGMEM = R"KEY(

-----BEGIN RSA PRIVATE KEY-----

MIIEpAIBAAKCAQEAsXkewMF81eobluGnE/MoK7hRrrFRLgdYA+XmkNmQ3OV2HFM4

gcyaoSdhlN4cj/VY9Y2lRp0NwZF26Z6gG1TxZJZbbdZc+HRSenui9C/mYGbCq7ft

zNoI+sO0GFlFyYvRUcoA8FSLrf4AMABacVYVe+36HYR/WpYvXzZNnk/nYfb7xZT4

XEyzM2VwVq9RiAoT0+g1bUInhB9KxFiucuX7aGo6XnW5IZfcHvQF7zB6srjZve6I

iw5fYbggAoolVlvNSE1+f4gUF3pqEuyFmFPsANc+GWCD34bW6ydHb+Fk80OxjAke

g1sHxOv+wQJ36sLLbIAOYx5Xef53MhLq1wHcJwIDAQABAoIBAANtpYxvsGupdBL8

FPAruFDv5JQNVNYgyjB5FnUh3syFmjx91w+a1jnauuXXTJHzWV6U9+ZfTk648qUc

rVcZ0vfwd5+tjmo2RQmcrc8txbODCKush+l3JMlDUo1N3C1CiC6eSlpaJ8CasYI7

34Nk+FFGjZhYYOVtFUkvkVxHit63kHOmKZ8VymwXXRrYUUjJMZcTIsAjMrW0n/zQ

5K2z8YTKHSLJAciVgLYSUvNecITJ4M20T0IEFiH36UReAjcD4f8p9c5w1W+kMgBO

eyxienzSpgIgL2E6YmIXMSm6kmNQOBOXLK0FwI78DSYDmOe2A0WcQ7lSgP0SU+WJ

H5HxPdECgYEA3diiUahHrNqOxULrG1MnyKEf9r6mOHsU52Wkqjh/zTS+8+hU1eB1

awg2l996rnm33nGfWI3D4JQduawNFraF1/IEn2iYjhVQmTDqtKrG93lJ85/9JWf8

iAGUaiHncFuIpl/anjVtoEMCOjRHPzjA3e28ySkkpFi5Q+tZD3L7RmUCgYEAzMup

MZKSRKbvgCqH7S/vKMSh3CZULfFQaluxsTNl7vas6HQT1MckmzwkWeb8mDVIpaUO

sIe8gz2emT0oArBlunvnnRZm2m/WLC5Yd8ea7wJEU56OVn9FoogxjKNdA/vnp1GK

QF2dAkCte1X7vkuAt8h2KQAYxwhVlfuzkSpu+ZsCgYBvq2WBjyCNLPLi7t1jwsbq

4dgyHyOSmLocBHkPyiLiy6M3KYLSqgrQ1rCMYEzj2JcqXK1mKO4pdMVFugpYnYXf

/o4/I6pvEuGgSxRJ3xEKbz4/aRBHLQHcAFeR2QEj+J9fiC6GpsRJAJH1dG1MsKe8

zrfR1mkAZTzUqHoLIcjEVQKBgQCuIMBUSQaa3sGm3QLD1kzoAgCJugE7KwIv3JUC

UOxruUudPg0ajtR+NS3HTxIYrL1/Mg+CY3cSs9LlAk3Bs5BbdjyhoUmEkzCsUh+g

gJQogmnsG1V1EDP6FQjkRoaw+3+lETyWq8HzB2E6DArHa1Ufbo+hHtbybBCxPNka

JdVD3wKBgQDa76rYaFXNDSCeCZATJN/3bdAS6ZekuKZd5+52rvfAvplrigztYyJc

MPwC04MrHjz62Rhqz6Ifv1JQnJ1ZXHkM8xfAHLMLbJuxgtj4DO6g8oGtaG99HaJ7

30N8E7RZ3wlV0+XL9NJecvnAEgZiLAIH3s0qiiY+/sm0YN0gzN24bA==

-----END RSA PRIVATE KEY-----

)KEY";

// Variables globales

WiFiClientSecure wifiClient;

MQTTClient mqttClient(256);

const int ledPin = 2; // GPIO donde está conectado el LED

// Función para conectarse a Wi-Fi

void connectWiFi() {

Serial.print("Conectando a Wi-Fi...");

WiFi.begin(WIFI\_SSID, WIFI\_PASSWORD);

while (WiFi.status() != WL\_CONNECTED) {

delay(500);

Serial.print(".");

}

Serial.println("\nWi-Fi conectado!");

Serial.print("Dirección IP: ");

Serial.println(WiFi.localIP());

}

// Función para conectarse a AWS IoT

void connectAWS() {

Serial.println("Configurando certificados...");

wifiClient.setCACert(CA\_CERT);

wifiClient.setCertificate(DEVICE\_CERT);

wifiClient.setPrivateKey(PRIVATE\_KEY);

mqttClient.begin(AWS\_IOT\_ENDPOINT, 8883, wifiClient);

mqttClient.onMessage(messageHandler);

Serial.print("Conectando a AWS IoT...");

while (!mqttClient.connect(THINGNAME)) {

Serial.print(".");

delay(1000);

}

if (mqttClient.connected()) {

Serial.println("\nConectado a AWS IoT!");

mqttClient.subscribe(AWS\_IOT\_SUBSCRIBE\_TOPIC);

Serial.println("Suscrito al tópico: " + String(AWS\_IOT\_SUBSCRIBE\_TOPIC));

} else {

Serial.println("Error al conectar a AWS IoT.");

}

}

// Función para manejar mensajes MQTT

void messageHandler(String &topic, String &payload) {

Serial.println("Mensaje recibido:");

Serial.println("Tópico: " + topic);

Serial.println("Payload: " + payload);

StaticJsonDocument<200> doc;

DeserializationError error = deserializeJson(doc, payload);

if (error) {

Serial.println("Error al parsear el JSON.");

return;

}

int message = doc["led"]; // Lee el estado del LED del JSON

if (message == 1) {

digitalWrite(ledPin, HIGH);

Serial.println("LED encendido.");

} else if (message == 0) {

digitalWrite(ledPin, LOW);

Serial.println("LED apagado.");

}

}

// Configuración inicial

void setup() {

Serial.begin(115200);

pinMode(ledPin, OUTPUT);

digitalWrite(ledPin, LOW); // Asegúrate de que el LED esté apagado al inicio

connectWiFi();

connectAWS();

}

// Bucle principal

void loop() {

mqttClient.loop();

if (!mqttClient.connected()) {

Serial.println("Reconectando a AWS IoT...");

connectAWS();

}

}