Để có thể làm được phần này thì yêu cầu các bạn cần hiểu và làm qua các bước cần thiết từ việc tạo Thing, tạo chứng chỉ xác thực rồi kích hoạt chứng chỉ cho thiết bị, khởi tạo và liên kết Policy với chứng chỉ cũng như đính kèm chứng chỉ xác thực cho Thing. Đồng thời các bạn cũng cần nắm rõ mô hình mà AWS IoTs đưa ra cho việc tương tác giữa các thiết bị và ứng dụng với Device Shadow Service sử dụng giao thức MQTT. Những nội dung này đã được trình bày ở các phần trước:

Xem lại phần 1: Hướng dẫn tạo Thing, Device Certificate và Policy

Xem lại phần 2: Tương tác với Device Shadow sử dụng giao thức MQTT

***Phần 3: Sử dụng ESP32 kết nối tới Amazon IoT Service***

**1. Cài đặt các thư viện cần thiết**

Hai thư viện chính cần sử dụng là:

* WiFiClientSecure (cho ESP32)
* PubSubClient (tác giả Knolleary)

Để ESP32 có thể kết nối tới AWS IoT chúng ta sử dụng thư viện WiFiClientSecure (Đây là thư viện built-in của ESP32). Lưu ý phiên bản cũ của thư viện WiFiClientSecure không hỗ trợ cho việc gửi chứng chỉ gốc CA. Nếu bạn gặp lỗi WiFiClientSecure::setCACert() not found thì bạn cần phải nâng cấp lên phiên bản mới nhất.

Để ESP32 có thể tương tác với AWS IoT thông qua giao thức MQTT như một client thì chúng ta sẽ sử dụng thư viện có tên PubSubClient. Các bạn có thể tải thư viện ở đây: <https://github.com/knolleary/pubsubclient>

Sau khi đã cài đặt xong thư viện, có một thay đổi nhỏ cực kì quan trọng mà chúng ta cần lưu ý nằm trong thư viện PubSubClient. Ở thư mục chứa thư viện PubSubClient, mở file PubSubClient.h ra sẽ tìm thấy định nghĩa:

**#define MQTT\_MAX\_PACKET\_SIZE 128**

Việc cấu hình mặc định này sẽ gây ra một lỗi quan trọng khi nhận dữ liệu trả về từ AWS thông qua các topic. Bởi vì kích thước tối đa của gói tin khi gửi tới các topic như *shadow/update/delta*, *shadow/update/accepted*,... có thể lên tới 128KB trong khi thư viện PubSubClient lại định nghĩa kích thước tối đa của gói tin chỉ vỏn vẹn có 128Byte... nên nếu dữ liệu trả về lớn hơn 128Byte có thể làm mất dữ liệu. Để khắc phục vấn đề này chúng ta sẽ tăng giá trị của biến **MQTT\_MAX\_PACK\_SIZE** lên như sau:

**#define MQTT\_MAX\_PACKET\_SIZE 1024**

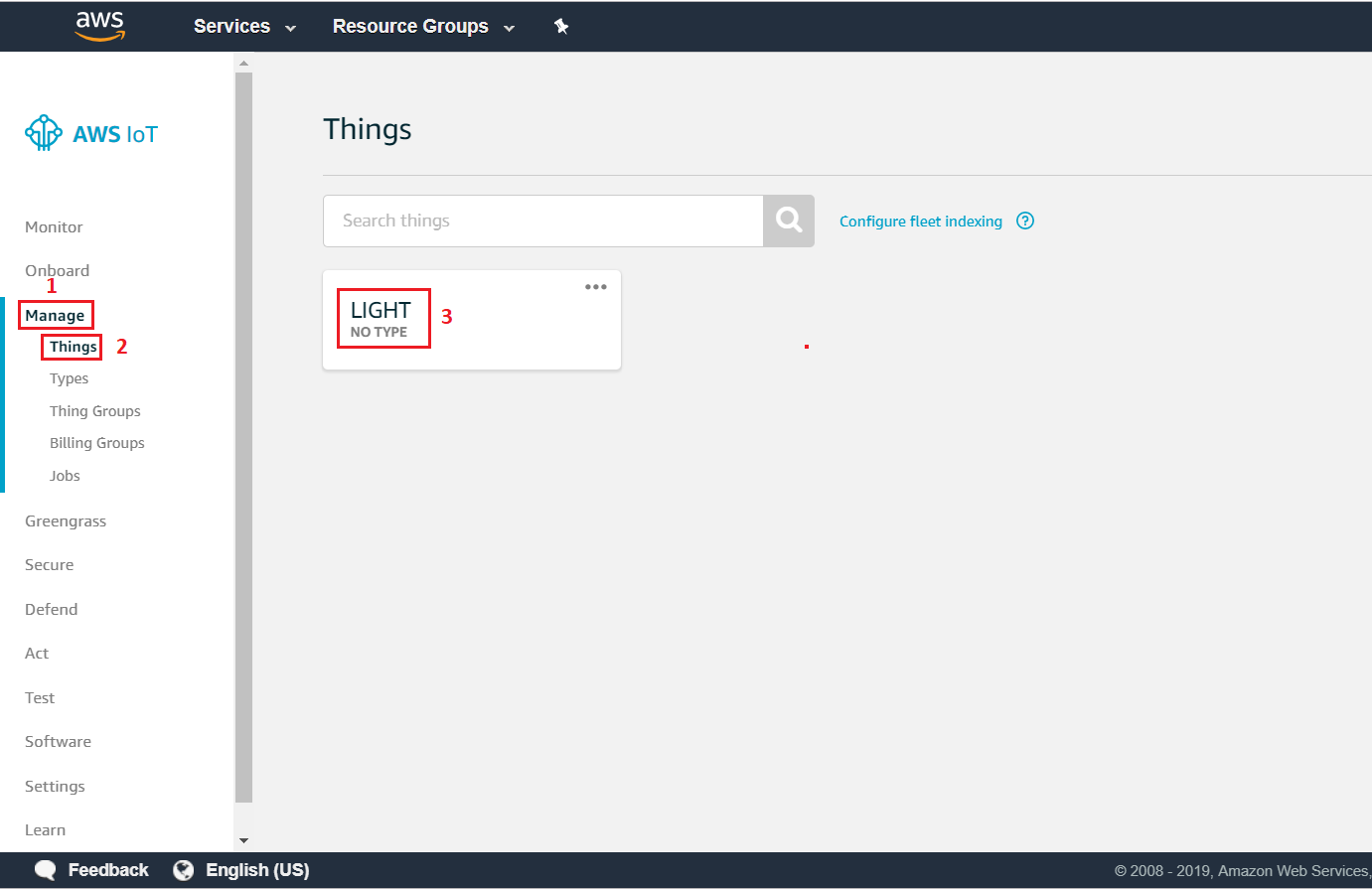
**2. Hướng dẫn viết source code cho ESP32**

Chương trình chính của chúng ta sẽ tuần tự thực hiện các bước như sau:

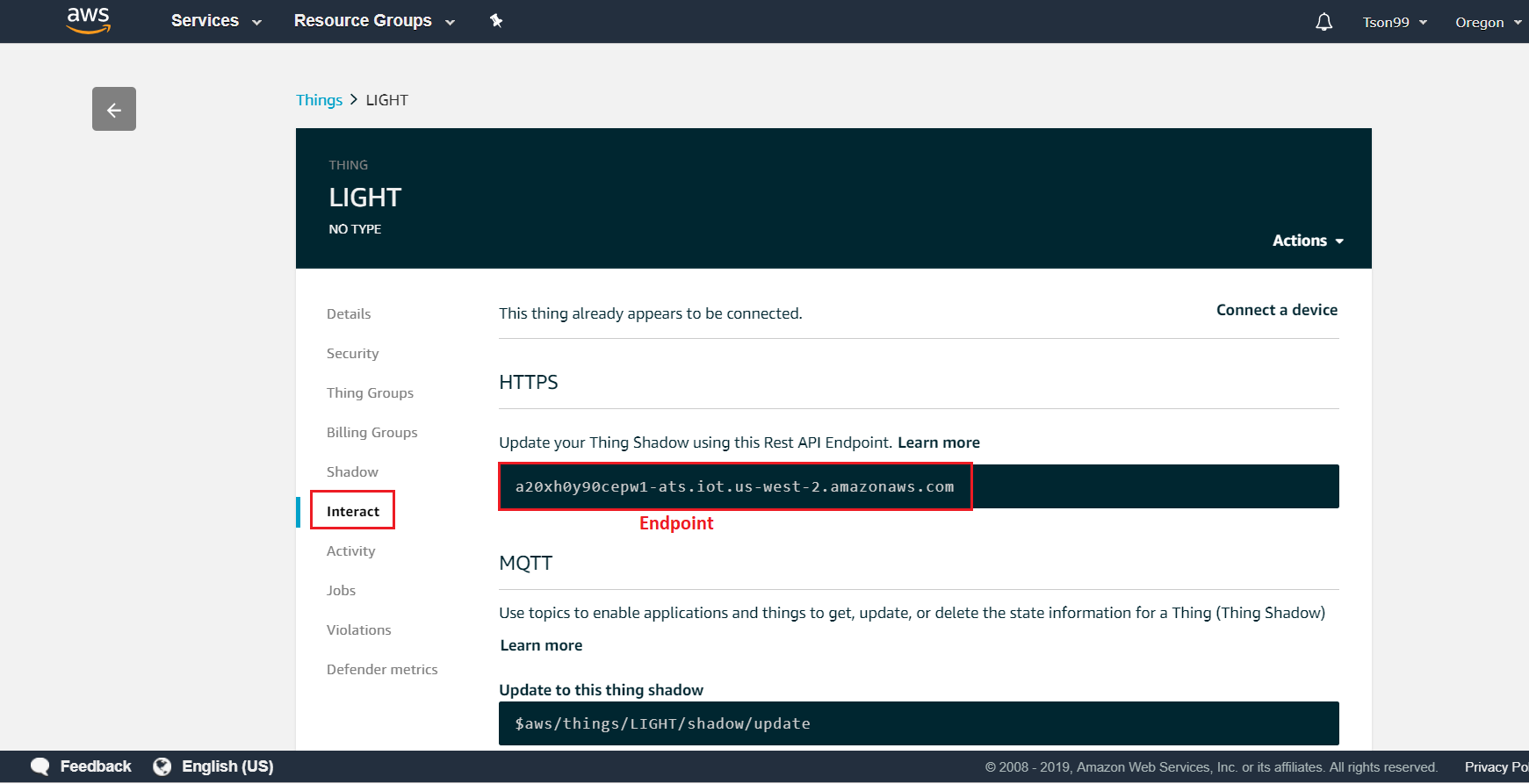
* Cài đặt và kết nối WiFi
* Cài đặt các chứng chỉ và privateKey trong thư viện WiFiClientSecure
* WiFiClientSecure::setCACert()
* WiFiClientSecure::setCertificate()
* WiFiClientSecure::setPrivateKey()
* Khởi tạo MQTT Client để kết nối tới AWS IoT Broker
* Port là 8883 (TLS)
* AWS IoT endpoint với định dạng xxxxxxx.iot.us-west-2.amazonaws.com
* Thực hiện Pub&Sub các topic

Để lấy được Certificate, privatekey cũng như rootCA các bạn lưu ý đọc lại phần 1 (Hướng dẫn tạo Thing, Device Certificate và Policy)

Để lấy được endpoint các bạn vào trang console của AWS IoT rồi thực hiện như hình dưới:



Các bạn chọn Thing mà mình muốn tương tác, ở đây mình ví dụ một Thing tên là LIGHT đã tạo từ trước. Chúng ta nhấn đúp để chọn, lập tức console sẽ chuyển hướng tới trang quản lý Thing này. Tiếp tục thực hiện như sau sẽ lấy được endpoint của Thing:



Sau khi đã cài đặt xong thư viện cũng như lấy được các thông tin cần thiết, chúng ta bắt đầu đi vào viết source code cho ESP32.

Các bạn sao chép đoạn mã sau đây vào máy và thực hiện việc cấu hình cho các biến.

#include <WiFiClientSecure.h>

#include <PubSubClient.h>

void connectAWSIoT();

void mqttCallback (char\* topic, byte\* payload, unsigned int length);

char \*ssid = "<YOUR\_SSID>";

char \*password = "<YOUR\_WIFI\_PASSWORD>";

const char \*endpoint = "<AWS\_IOT\_ENDPOINT>";

// Example: xxxxxxxxxxxxxx.iot.us-west-2.amazonaws.com

const int port = 8883;

char \*pubTopic = "$aws/things/<DEVICE\_NAME>/shadow/update";

char \*subTopic = "$aws/things/<DEVICE\_NAME>/shadow/update/delta";

const char\* rootCA = "-----BEGIN CERTIFICATE-----\n"

"......"

"-----END CERTIFICATE-----\n";

const char\* certificate = "-----BEGIN CERTIFICATE-----\n"

"......"

"-----END CERTIFICATE-----\n";

const char\* privateKey = "-----BEGIN RSA PRIVATE KEY-----\n"

"......"

"-----END RSA PRIVATE KEY-----\n";

WiFiClientSecure httpsClient;

PubSubClient mqttClient(httpsClient);

void setup() {

delay(1000);

Serial.begin(115200);

// Start WiFi

Serial.println("Connecting to ");

Serial.print(ssid);

WiFi.begin(ssid, password);

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

delay(500);

Serial.print(".");

}

Serial.println("\nConnected to WiFi.");

// Configure MQTT Client

httpsClient.setCACert(rootCA);

httpsClient.setCertificate(certificate);

httpsClient.setPrivateKey(privateKey);

mqttClient.setServer(endpoint, port);

mqttClient.setCallback(mqttCallback);

connectAWSIoT();

}

void connectAWSIoT() {

while (!mqttClient.connected()) {

if (mqttClient.connect("ESP32\_device")) {

Serial.println("Connected to AWS IoT.");

int qos = 0;

mqttClient.subscribe(subTopic, qos);

Serial.println("Subscribed to topics.");

} else {

Serial.print("Failed. Error state=");

Serial.print(mqttClient.state());

// Wait 5 seconds before retrying

delay(5000);

}

}

}

long messageSentAt = 0;

int dummyValue = 0;

char pubMessage[128];

void mqttCallback (char\* topic, byte\* payload, unsigned int length) {

Serial.print("Received. topic=");

Serial.println(topic);

for (int i = 0; i < length; i++) {

Serial.print((char)payload[i]);

}

Serial.print("\n");

}

void mqttLoop() {

if (!mqttClient.connected()) {

connectAWSIoT();

}

mqttClient.loop();

long now = millis();

if (now - messageSentAt > 5000) {

messageSentAt = now;

sprintf(pubMessage, "{\"state\": {\"desired\":{\"foo\":\"%d\"}}}", dummyValue++);

Serial.print("Publishing message to topic ");

Serial.println(pubTopic);

Serial.println(pubMessage);

mqttClient.publish(pubTopic, pubMessage);

Serial.println("Published.");

}

}

void loop() {

mqttLoop();

}

***\*Lưu ý***: Nếu các bạn sao chép trực tiếp các file *rootCA*, *certificate* và *privateKey* vào biến tương ứng sẽ gây ra lỗi. Định dạng của các biến này bắt buộc phải thỏa mãn các điều kiện sau:

* Mỗi dòng phải kết thúc bởi 1 kí hiệu \n
* Được bao quanh bởi dấu “ ”

Ví dụ:

const char\* rootCA = "-----BEGIN CERTIFICATE-----\n"

"MIIDQTCCAimgAwIBAgITBmyfz5m/jAo54vB4ikPmljZbyjANBgkqhkiG9w0BAQsF\n"

"ADA5MQswCQYDVQQGEwJVUzEPMA0GA1UEChMGQW1hem9uMRkwFwYDVQQDExBBbWF6\n"

"b24gUm9vdCBDQSAxMB4XDTE1MDUyNjAwMDAwMFoXDTM4MDExNzAwMDAwMFowOTEL\n"

"MAkGA1UEBhMCVVMxDzANBgNVBAoTBkFtYXpvbjEZMBcGA1UEAxMQQW1hem9uIFJv\n"

"b3QgQ0EgMTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBALJ4gHHKeNXj\n"

"ca9HgFB0fW7Y14h29Jlo91ghYPl0hAEvrAIthtOgQ3pOsqTQNroBvo3bSMgHFzZM\n"

"9O6II8c+6zf1tRn4SWiw3te5djgdYZ6k/oI2peVKVuRF4fn9tBb6dNqcmzU5L/qw\n"

"IFAGbHrQgLKm+a/sRxmPUDgH3KKHOVj4utWp+UhnMJbulHheb4mjUcAwhmahRWa6\n"

"VOujw5H5SNz/0egwLX0tdHA114gk957EWW67c4cX8jJGKLhD+rcdqsq08p8kDi1L\n"

"93FcXmn/6pUCyziKrlA4b9v7LWIbxcceVOF34GfID5yHI9Y/QCB/IIDEgEw+OyQm\n"

"jgSubJrIqg0CAwEAAaNCMEAwDwYDVR0TAQH/BAUwAwEB/zAOBgNVHQ8BAf8EBAMC\n"

"AYYwHQYDVR0OBBYEFIQYzIU07LwMlJQuCFmcx7IQTgoIMA0GCSqGSIb3DQEBCwUA\n"

"A4IBAQCY8jdaQZChGsV2USggNiMOruYou6r4lK5IpDB/G/wkjUu0yKGX9rbxenDI\n"

"U5PMCCjjmCXPI6T53iHTfIUJrU6adTrCC2qJeHZERxhlbI1Bjjt/msv0tadQ1wUs\n"

"N+gDS63pYaACbvXy8MWy7Vu33PqUXHeeE6V/Uq2V8viTO96LXFvKWlJbYK8U90vv\n"

"o/ufQJVtMVT8QtPHRh8jrdkPSHCa2XV4cdFyQzR1bldZwgJcJmApzyMZFo6IQ6XU\n"

"5MsI+yMRQ+hDKXJioaldXgjUkK642M4UwtBV8ob2xJNDd2ZhwLnoQdeXeGADbkpy\n"

"rqXRfboQnoZsG4q5WTP468SQvvG5\n"

"-----END CERTIFICATE-----\n";

Đoạn mã trên cứ 5 giây sẽ publish một chuỗi có định dạng JSON tới trường desired của topic $aws/things/<Your\_Device\_ID>/shadow/update. Đồng thời chúng ta cũng sẽ subscribe topic $aws/things/<Your\_Device\_ID>/shadow/update/delta để có thể nhận được dữ liệu trả về từ AWS Broker khi có sự sai khác dữ liệu giữa trường *desired* và *reported* trong topic *$aws/.../shadow/update.*

Sau khi nạp code thành công chúng ta mở Serial Monitor trong Arduino IDE ra để quan sát (Lưu ý cài đặt mức baudrate là 115200). Dữ liệu nhận được sẽ như sau:



Như vậy là ở phần này chúng ta đã biết cách làm thế nào để kết nối ESP32 tới AWS thông qua giao thức MQTT (có TLS). Tuy nhiên, source code ở trên chỉ ở mức giao tiếp MQTT căn bản. Để có thể xây dựng được một mô hình rộng lớn tích hợp nền tảng AWS IoT các bạn cần phải nắm rõ được lý thuyết cũng như kiến trúc do Amazon đề ra mà tự tạo cho mình một hệ thống tương tác rõ ràng. Chúc các bạn thành công!