Sau khi các bạn đã nắm được tổng quan về nền tảng Google IoT Core và biết cách làm sao để kết nối bất kỳ thiết bị IoT nào đến Google IoT Core, thì ở phần này mình sẽ tập trung hướng dẫn cho các bạn cách để lập trình một dòng MCU khá nổi tiếng là ESP32 thực hiện việc kết nối tới Google IoT Core thông qua giao thức MQTT nhé. Sau khi kết nối được rồi thì phần xử lý còn lại là phụ thuộc ở bạn!

**Phần 5: Kết nối ESP32 đến Google IoT Core thông qua giao thức MQTT**

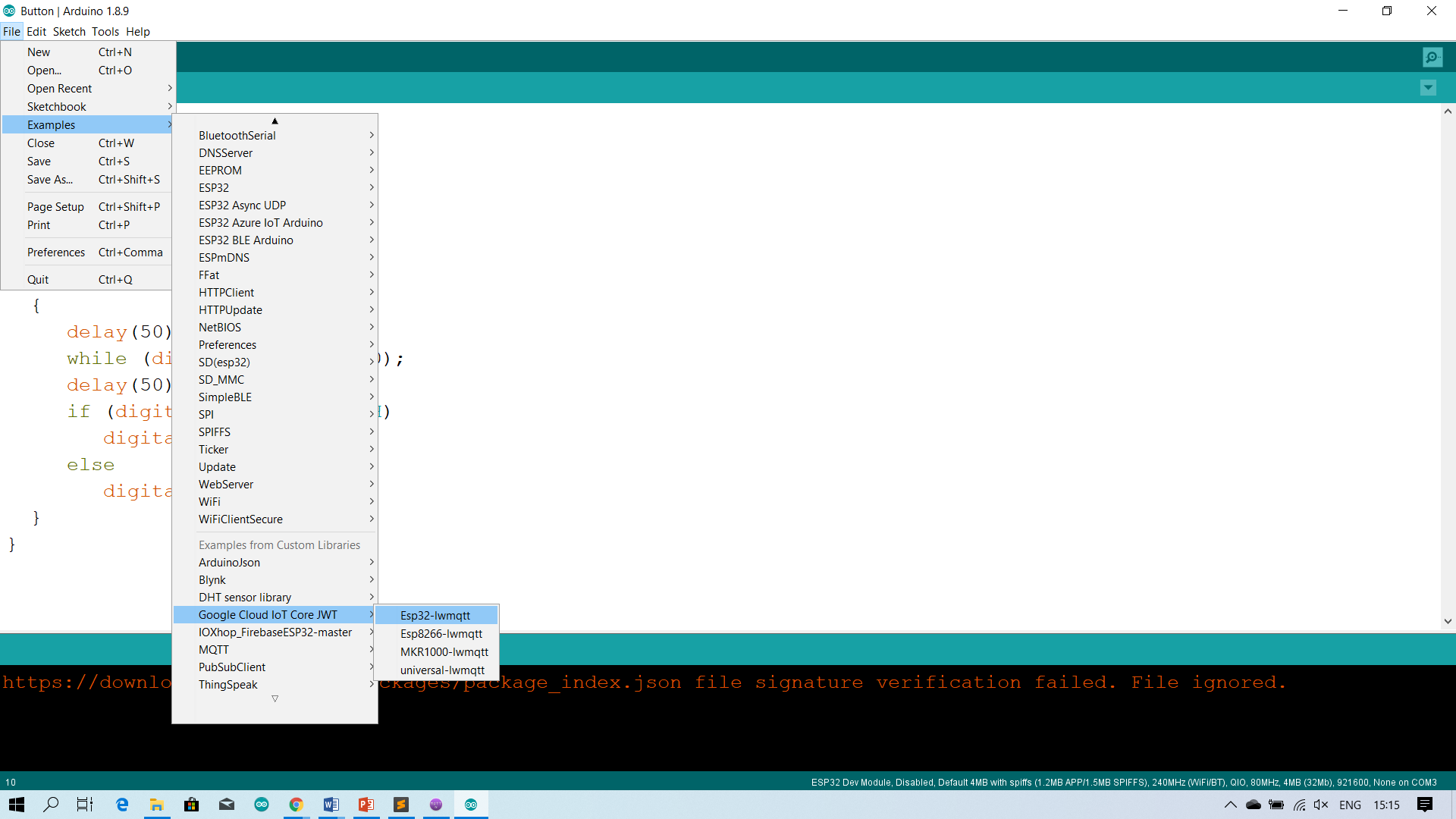
**I. Cấu hình cho ESP32**

Để kết nối ESP32 với Google IoT Core, chúng ta sử dụng thư viện “Google Cloud IoT Core JWT”, đây là thư viện chuẩn do lập trình viên của google cung cấp, chúng ta có thể thoải mái sử dụng mà không lo về lỗi.

Link tải: <https://github.com/GoogleCloudPlatform/google-cloud-iot-arduino>

Sau khi tải thư viện về và cài đặt, chúng ta sẽ sử dụng example để thực hiện việc cấu hình

Các bạn chọn **File** -> **Examples** -> **Google Cloud IoT Core JWT** -> **Esp32-lwmqtt**



Example đã cho chúng ta sẵn 3 file, chúng ta chỉ cần quan tâm tới file *ciotc\_config.h*. Mọi thông số mà chúng ta thay đổi trong file này sẽ giúp ESP32 kết nối được với Google IoT Core.

Trước tiên chúng ta sẽ cấu hình 6 thông số quan trọng cho ESP32 ở trong file *ciotc\_config.h* như sau:

// Wifi newtork details.

const char \*ssid = "SSID"; //Tên Wifi bạn muốn kết nối

const char \*password = "PASSWORD"; //Mật khẩu Wifi

// Cloud iot details.

const char \*project\_id = "tapitdemo"; //Project-ID

const char \*location = "asia-east1"; //Region

const char \*registry\_id = "Tapit-ID"; //Registry-ID

const char \*device\_id = "ESP32"; //Device-ID

Sau đó chúng ta sẽ thay đổi thông số của biến *private\_key\_str* dựa vào file **private key** (Lưu ý thư viện này sử dụng thuật toán sinh khóa EC để thực hiện việc kết nối).

Thông thường file private key sẽ có dạng như sau:

-----BEGIN EC PRIVATE KEY-----

MHcCAQEEIEQFiod5bfRSU7pOyskqbxAFpBhQMa+0hYHFycHkM4Y7oAoGCCqGSM49

AwEHoUQDQgAE4WbOWJBH3N+S24+39qwMiV4oLfcD4f/4hWntizpq36Pmfed+mjBB

qsPDpCGxrCXEOacRmMYLFlYiNlBR9UDXCg==

-----END EC PRIVATE KEY-----

Nhưng khi gán giá trị cho biến *private\_key\_str* thì phải chuyển đổi chuỗi base64 ở trên thành chuỗi hex. Để chuyển đổi, ở giao diện dòng lệnh gcloud ta điền:

*openssl ec -in <private-key.pem> -noout –text*

Lưu ý thay chuỗi bôi đỏ ở trên thành file private của chúng ta, gcloud sẽ trả về cho chúng ta chuỗi hex của file đó, sau đó ta copy và gán vào biến *private\_key\_str.* Ở đây mình ví dụ một chuỗi hex như sau:

const char \*private\_key\_str =

"6e:b8:17:35:c7:fc:6b:d7:a9:cb:cb:49:7f:a0:67:"

"63:38:b0:90:57:57:e0:c0:9a:e8:6f:06:0c:d9:ee:"

"31:41";

Tiếp theo ta có thể thay đổi biến *jwt\_exp\_secs* hoặc để mặc định, đây là biến xác định thời gian có hiệu lực của một chuỗi JWT (Việc sinh chuỗi JWT thì google đã viết hàm sẵn cho chúng ta).

// Time (seconds) to expire token += 20 minutes for drift

const int jwt\_exp\_secs = 3600; // Maximum 24H (3600\*24)

Cuối cùng là thay đổi thông số của biến *root\_cert*. Đây chính biến là khai báo chứng chỉ cho region mà chúng ta đang sử dụng cho ESP32.

Để lấy chứng chỉ cho region, tại giao diện dòng lệnh gcloud ta điền:

*openssl s\_client -showcerts -connect mqtt.googleapis.com:8883*

Sau đó bạn copy toàn bộ nội dung hiển thị trên giao diện dòng lệnh gcloud và gán vào biến *root\_cert*.

const char \*root\_cert =

"-----BEGIN CERTIFICATE-----"

"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"

"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"

"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"

"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"

"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"

"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"

"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"

"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"

"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"

"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"

"xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"

"-----END CERTIFICATE-----";

Sau khi đã cấu hình cho file **ciotc\_config.h** xong, chúng ta có thể thực hiện việc kết nối ESP32 tới Google IoT Core.

**II. Tương tác giữa ESP32 với Google IoT Core**

* Để Publish dữ liệu từ ESP32 -> Cloud ta chỉnh sửa code bên trong hàm *loop()* nằm ở file **Esp32**-**lwmqtt.ino**. Mình có làm sẵn một đoạn code mẫu trong hàm này, khi chúng ta nhấn button trên ESP32 thì sẽ đảo trạng thái led trên board đồng thời gửi tin nhắn “Bat den” hoặc “Tat den” tới Google IoT Core.

// TODO: replace with your code

// publish a message roughly every second.

/\*if (millis() - lastMillis > 60000) {

lastMillis = millis();

//publishTelemetry(mqttClient, "/sensors", getDefaultSensor());

publishTelemetry(getDefaultSensor());

}\*/

if (digitalRead(0) == 0)

{

delay(50);

while (digitalRead(0) == 0);

delay(50);

if (digitalRead(2) == HIGH)

{

digitalWrite(2, LOW);

publishTelemetry("Tat den");

}

else

{

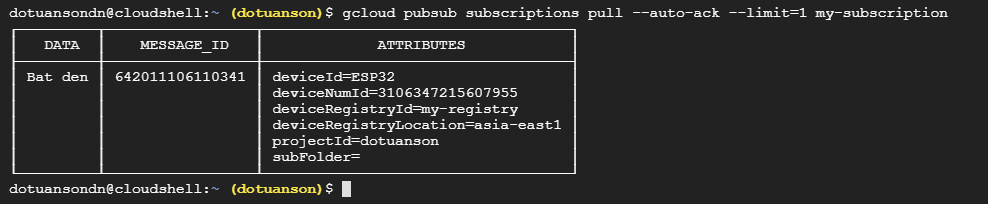
digitalWrite(2, HIGH);

publishTelemetry("Bat den");

}

}

Sau đó chúng ta sử dụng công cụ dòng lệnh gcloud để quan sát gói tin:



* Để Subscribe dữ liệu từ Cloud -> ESP32 chúng ta chỉnh sửa code bên trong hàm *messageReceived()* nằm ở file **esp32-mqtt.h**. Mình có làm sẵn một đoạn code mẫu trong hàm này, khi Google IoT Core gửi chuỗi “ON” hoặc “1” cho ESP32 thì led trên board sẽ bật. Ngược lại khi gửi chuỗi “OFF” hoặc “0” cho ESP32 thì led trên board sẽ tắt.

// !!REPLACEME!!

// The MQTT callback function for commands and configuration updates

// Place your message handler code here.

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

Serial.println("incoming: " + topic + " - " + payload);

if (payload == "1" || payload == "ON")

digitalWrite(2, HIGH);

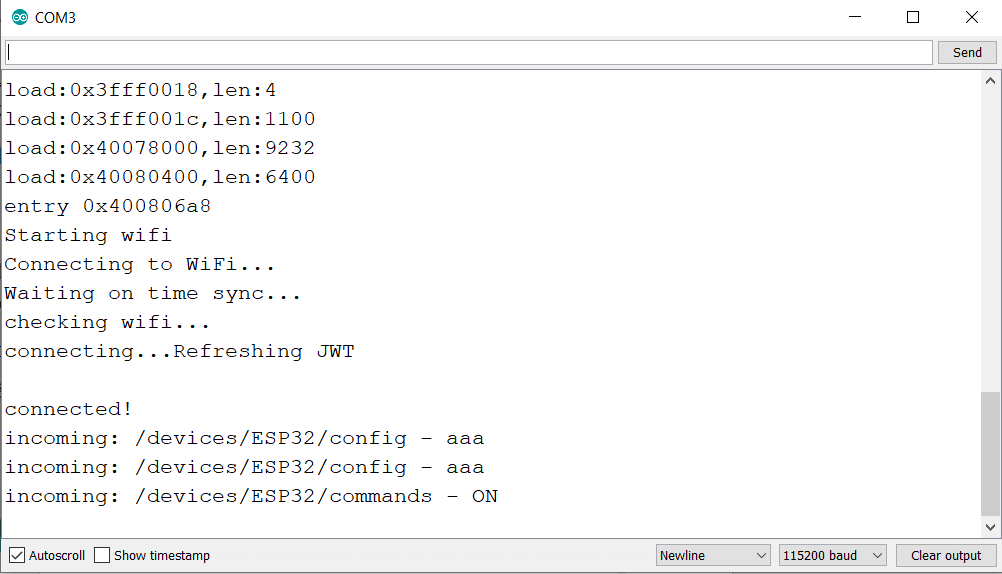
else

if (payload == "0" || payload == "OFF")

digitalWrite(2, LOW);

}

Sau khi upload code lên ESP32, chúng ta mở **Serial Monitor** để kiểm tra xem dữ liệu đã nhận được chưa. Nếu màn hình Serial hiển thị như hình bên dưới thì xin chúc mừng các bạn đã thiết lập thành công!



Trên đây mình đã giới thiệu cho các bạn cách để kết nối vi điều khiển ESP32 tới nền tảng dịch vụ Google IoT Core. Tương tự các bạn có thể mở rộng cho dòng vi điều khiển khác cũng nổi tiếng không kém chính là ESP8266. Mình mong là qua các bài viết do mình chia sẻ các bạn có thể nắm rõ hơn về mô hình của Google IoT Core cũng như có thể tự tạo cho bản thân một ứng dụng hay ho tích hợp nền tảng dịch vụ này. Bài viết của mình đến đây là kết thúc, hẹn gặp lại các bạn ở những bài hướng dẫn tiếp theo của nhóm TAPIT IoTs!