BÁO CÁO THỰC HÀNH

**Bài thực hành số 03: An toàn khi truyền dữ liệu**

**Môn học:** Bảo mật Internet of Things

**Lớp:** NT535.P21.1

**THÀNH VIÊN THỰC HIỆN (Nhóm xx):**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Họ và tên** | **MSSV** |  | **Điểm tự đánh giá** |
| 1 | Phạm Thiều Gia Khang | 21520967 | **10/10** |
| 2 | Phan Văn B | 20520002 |
|  |  |  |  |

**ĐÁNH GIÁ KHÁC:**

|  |  |
| --- | --- |
| Tổng thời gian thực hiện | 2 tuần |
| Phân chia công việc |  |
| Ý kiến *(nếu có)*  + Khó khăn  + Đề xuất, kiến nghị |  |

Phần bên dưới của báo cáo này là báo cáo chi tiết của nhóm thực hiện

MỤC LỤC

[A. BÁO CÁO CHI TIẾT 3](#_Toc195451093)

[1. Chứng thực khi tham gia vào MQTT 3](#_Toc195451094)

[a. Cài đặt mô hình: 3](#_Toc195451095)

[b. Demo: 6](#_Toc195451096)

[2. Mã hoá kênh truyền TLS/SSL cho MQTT: 10](#_Toc195451097)

[a. 10](#_Toc195451098)

[3. Mã hoá dữ liệu khi truyền cho MQTT: 10](#_Toc195451099)

[a. Kịch bản: 10](#_Toc195451100)

[b. Cài đặt mô hình: 10](#_Toc195451101)

[c. Demo: 12](#_Toc195451102)

[B. TÀI LIỆU THAM KHẢO 14](#_Toc195451103)

# BÁO CÁO CHI TIẾT

## Chứng thực khi tham gia vào MQTT

### Cài đặt mô hình:

Kịch bản bao gồm 2 Node là 2 WeMos D1 R2 và 1 Mosquitto MQTT Broker. Các node sẽ truy cập vào Broker thông qua tài khoản và mật khẩu. Thực hiện điều khiển đèn đồng bộ qua nut bấm trên từng Node.

Thực hiện cài tài khoản trên Broker:

Tạo tài khoản cho 2 Node, tạo 2 tài khoản wemos1 và wemos2 đặt mật khẩu là admin123:

$ sudo mosquitto\_passwd -c /etc/mosquitto/passwd wemos1

$ sudo mosquitto\_passwd /etc/mosquitto/passwd wemos2

Tạo file Access list cấp toàn quyền cho 2 user:

$ sudo nano /etc/mosquitto/acl

Thêm quyền vào file acl:

# Quyền cho wemos1

user wemos1

topic readwrite wemos/led

# Quyền cho wemos2

user wemos2

topic readwrite wemos/led

Thêm file mật khẩu và file acl vào file mosquito.conf:

listener 1883

allow\_anonymous false

acl\_file /etc/mosquitto/acl

password\_file /etc/mosquitto/passwd

Lưu file config và mở MQTT Broker chờ kết nối từ các node:

sudo mosquitto -c /etc/mosquitto/mosquitto.conf

mosquitto\_sub -t "wemos/led" -u "wemos1" -P "admin123"

mosquitto\_sub -t "wemos/led" -u "wemos2" -P "admin123"

Code WeMos:

# Gọi biến value để chỉnh giá trị đèn

int value = 0;

# Các thông số cho sự kiện giữ nút nhấn

bool buttonPressed = false;

unsigned long buttonPressTime = 0;

const long holdTime = 1000;

# Đăng nhập vào mạng

const char\* ssid = "WemosTest";

const char\* password = "01072003";

# Đăng nhập vào MQTT Broker

const char\* mqtt\_server = "192.168.43.40";

const int mqtt\_port = 1883;

const char\* mqtt\_user = "wemos1";

const char\* mqtt\_password = "admin123";

Hàm Callback() nhận phản hồi từ Broker:

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

Serial.print("Message arrived [");

Serial.print(topic);

Serial.print("] ");

String message = "";

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

message += (char)payload[i];}

Serial.println(message);

// Xử lý MQTT để bật/tắt LED từ xa

if (String(topic) == "wemos/led") {

// Tắt tất cả LED trước khi bật

digitalWrite(LED1, LOW);

digitalWrite(LED2, LOW);

digitalWrite(LED3, LOW);

# Xử lý bật đèn theo message

if (message == "LED1") {

digitalWrite(LED1, HIGH);

value = 1;

} else if (message == "LED2") {

digitalWrite(LED2, HIGH);

value = 2;

} else if (message == "LED3") {

digitalWrite(LED3, HIGH);

value = 0;

}

delay(500);

}

}

Hàm reconnect() để kết nối vào Broker và subscribe vào topic nhận tín hiệu đèn:

void reconnect() {

while (!client.connected()) {

Serial.print("Attempting MQTT connection...");

if (client.connect("wemos1", mqtt\_user, mqtt\_password)) {

Serial.println("Connected to MQTT Broker!");

client.subscribe("wemos/led");

} else {

Serial.print("Failed, rc=");

Serial.print(client.state());

Serial.println(" retrying in 5 seconds...");

delay(5000);

}

}

}

Xử lý nhận tín hiệu từ nút bấm gửi tín hiệu điều khiển đèn lên Broker:

bool buttonState = digitalRead(BTN);

if (buttonState == HIGH) { // Nhấn nút

if (!buttonPressed) {

buttonPressed = true;

buttonPressTime = millis();

}

else if (millis() - buttonPressTime >= holdTime) {

#Gửi tín hiệu đèn theo tín hiệu nút nhấn và thay đổi giá trị value

if (value == 0) {

client.publish("wemos/led", "LED1");

value += 1;

Serial.print("Value: ");

Serial.println(value);

}

else if (value == 1) {

client.publish("wemos/led", "LED2");

value +=1;

Serial.print("Value: ");

Serial.println(value);

}

else if (value == 2) {

client.publish("wemos/led", "LED3");

value = 0;

Serial.print("Value: ");

Serial.println(value);

}

buttonPressed = false; // Reset trạng thái sau khi đã đổi LED

}

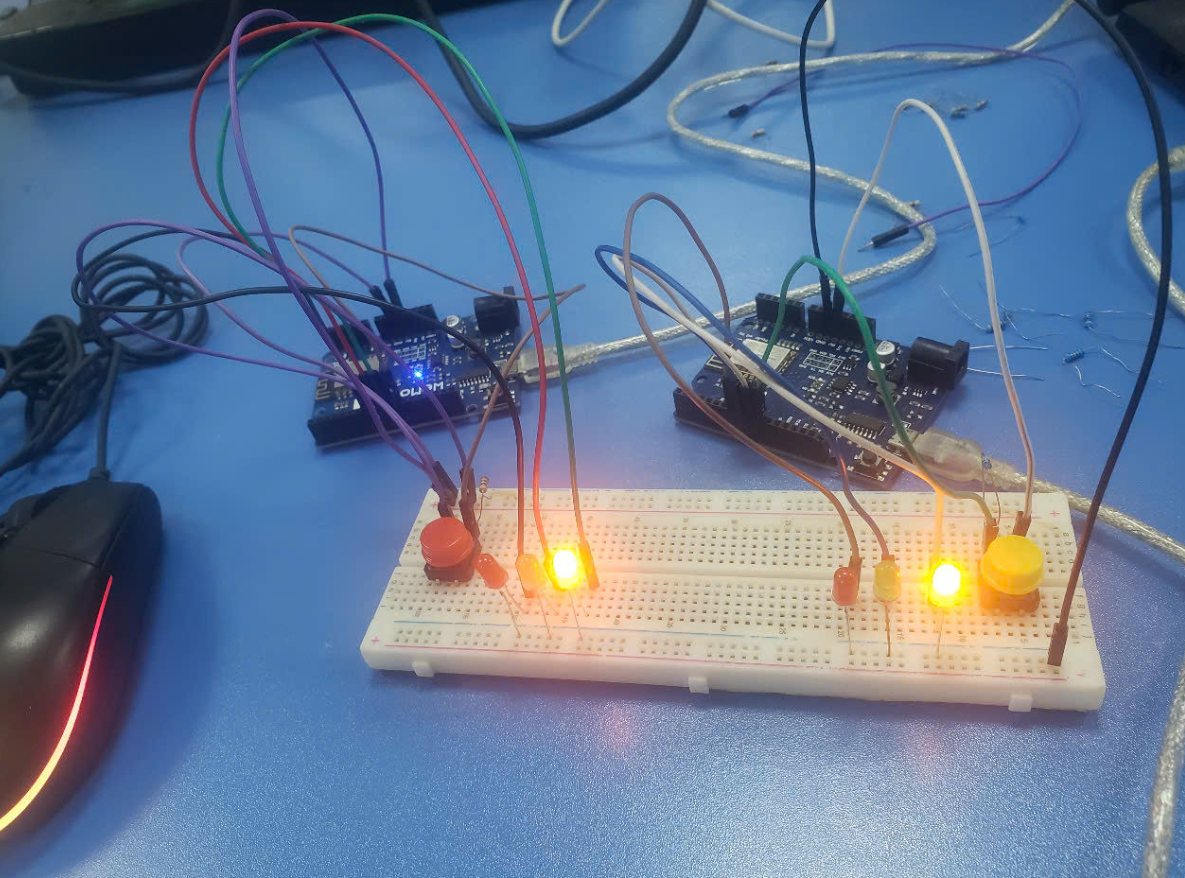
} else {

buttonPressed = false;

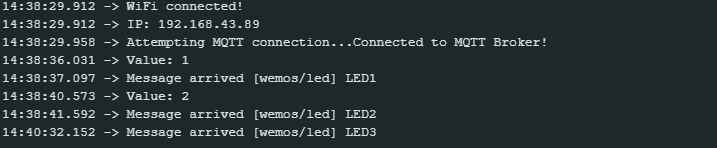
}

### Demo:

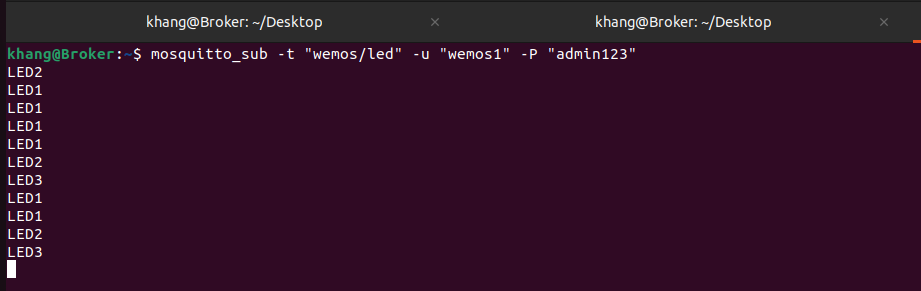
Mô hình:



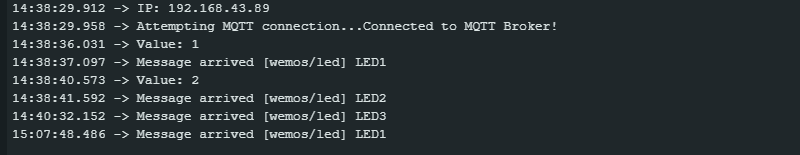
Trên 2 Node khi có tín hiệu thay đổi sẽ in ra giá trị value hiện tại cùng với tin nhắn nhận Callback từ broker:



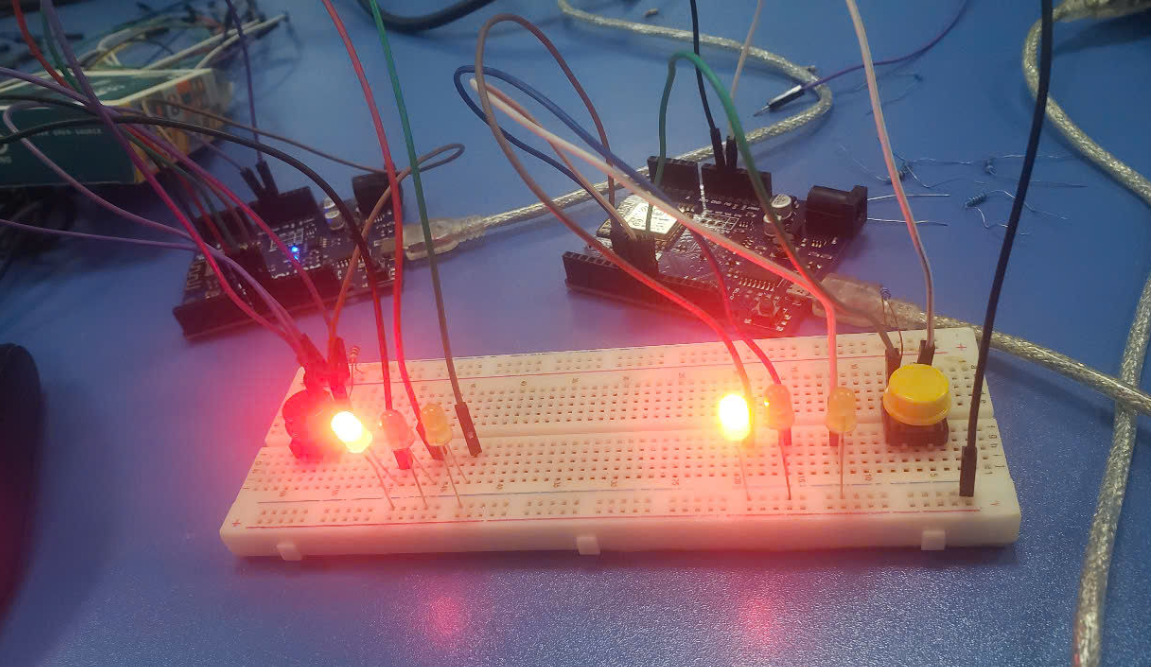
Trên Broker, dữ liệu gửi lên sẽ được ghi lại trên từng topic:



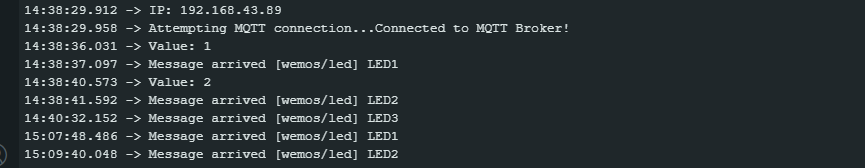
Khi giá trị là LED1:



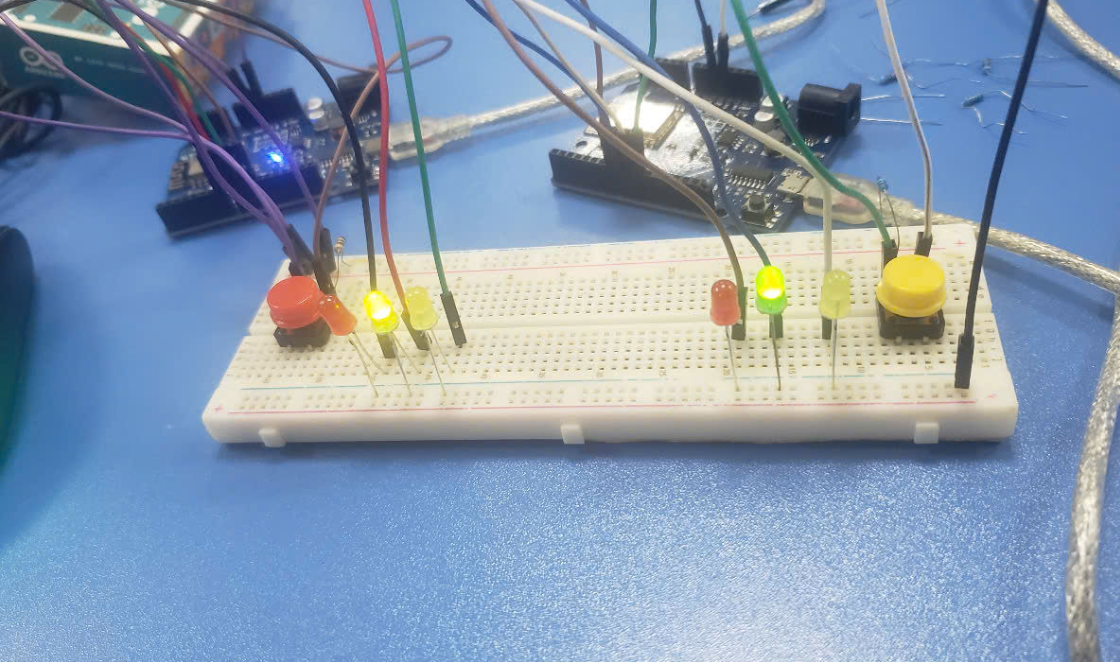
Đèn đỏ sáng:



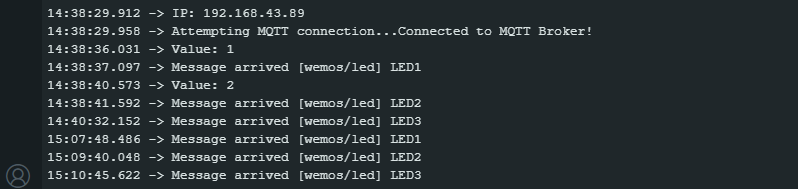
Khi giá trị là LED2:



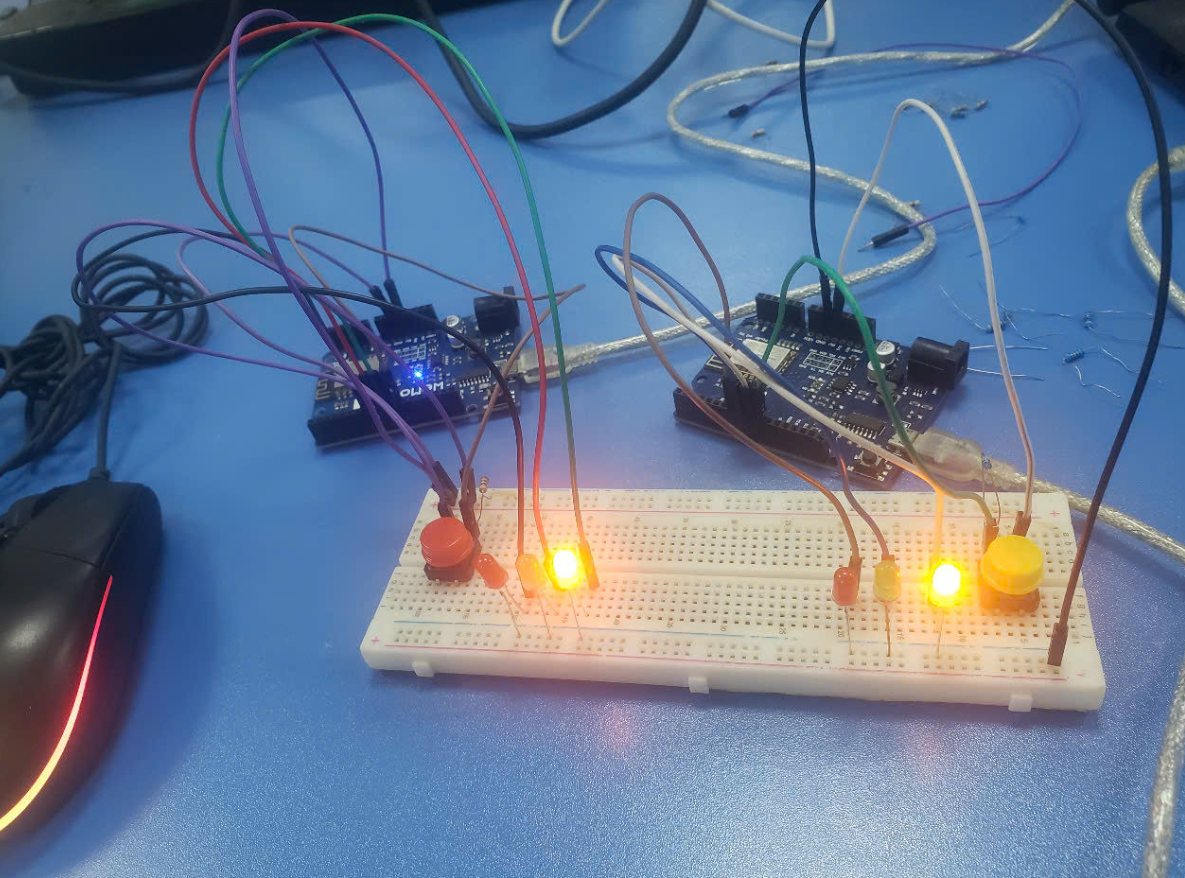
Đèn xanh sáng:



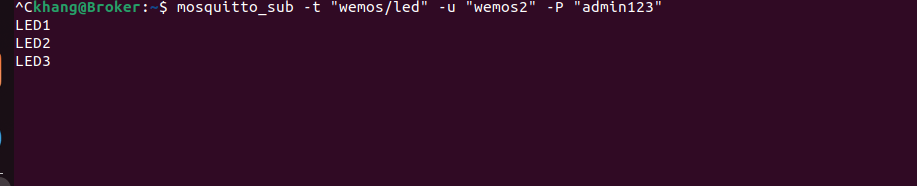
Khi giá trị là LED3:



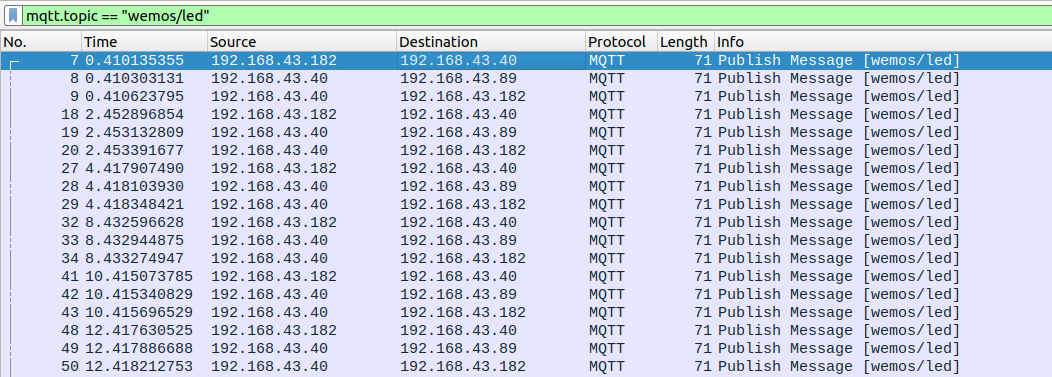
Đèn vàng sáng:



Log trên Broker:

****

Bắt gói tin bằng wireshark:

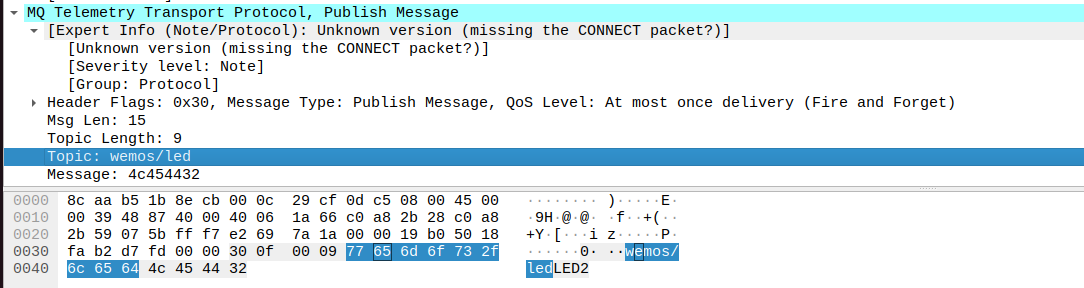


Quá trình thực hiện, Client sẽ gửi tuần tự các gói tín Request tới Broker xác nhận rằng mình vẫn còn đang hoạt động, Broker sẽ gửi trả lại 1 gói Response để trả lời:



Khi giao tiếp, các client sẽ gửi trực tiếp các gói tin MQTT chứa dữ liệu đến Broker và các client nhận subcribe từ topic này cũng sẽ nhận được các gói tin ấy. Cụ thể trong hình trên thì IP 192.168.43.40 là Broker, 192.168.43.182 là Node 1, 192.168.43.89 là Node 2.

Nội dung sẽ được ghi trong phần Payload của gói tin:



Video [demo](https://drive.google.com/file/d/1Hu169MBzddp6yzr3SM0hO-AVrVsDp5HD/view?usp=sharing) tại đây.

## Mã hoá kênh truyền TLS/SSL cho MQTT:

### 

## Mã hoá dữ liệu khi truyền cho MQTT:

### Kịch bản:

Mô hình bao gồm 2 Node và 1 Broker, mỗi node sẽ gửi dữ liệu về nhiệt độ, độ ẩm và điều kiện ánh sáng được đo bởi cảm biến DHT22 và BH1750 lên Broker. Thực hiện mã hóa và giải mã ở các Node. Mở wireshark trên Broker để kiểm tra nghe lén.

### Cài đặt mô hình:

Cài đặt Broker:

Khởi tạo một máy ảo với hệ điều hành Ubuntu 22.04 làm MQTT Broker, cài đặt Mosquitto

MQTT trên máy ảo.

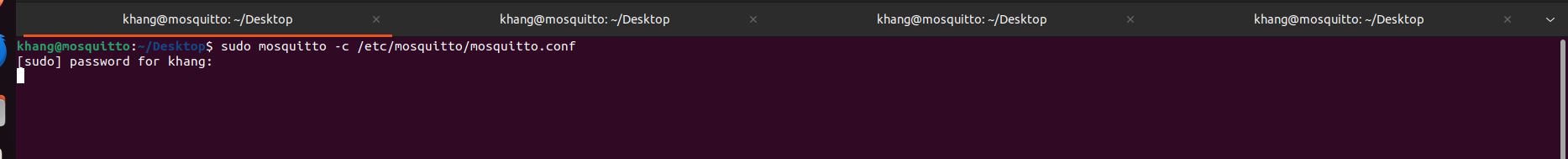
Thêm vào trong file mosquitto.conf để Broker có thể nghe được các các kết nối từ bên ngoài từ port 1883:

listener 1883

allow\_anonymos true

Chạy Mosquitto Broker qua lệnh:

sudo mosquito –c /etc/mosquito/mosquito.conf



Khởi tạo 3 topic để nhận dữ liệu lần lượt từ Node 1 (wemos1), Node 2 (wemos2) và topic led để nhận thông tin bật tắt đèn:

mosquito\_sub –t “wemos1/sensor”

mosquito\_sub –t “wemos2/sensor”

Code WeMos (2 Node có code tương tự nhau):

Khởi tạo khóa AES:

// AES config

AESLib aesLib;

byte aes\_key[] = {

0x30, 0x31, 0x32, 0x33,

0x34, 0x35, 0x36, 0x37,

0x38, 0x39, 0x41, 0x42,

0x43, 0x44, 0x45, 0x46

};

byte aes\_iv[N\_BLOCK] = {

0x00, 0x01, 0x02, 0x03,

0x04, 0x05, 0x06, 0x07,

0x08, 0x09, 0x0A, 0x0B,

0x0C, 0x0D, 0x0E, 0x0F

};

Hàm reconnect() để khởi tạo kết nối tới Broker:

void reconnect() {

while (!client.connected()) {

Serial.print("Attempting MQTT connection...");

if (client.connect("wemos1")) {

Serial.println("Connected to MQTT Broker!");

client.subscribe("wemos2/sensor");

} else {

Serial.print("Failed, rc=");

Serial.print(client.state());

Serial.println(" retrying in 5 seconds...");

delay(5000);

}

}

}

Hàm callback đăng ký nhận phản hồi từ Broker:

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

String base64Input = "";

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

base64Input += (char)payload[i];

}

// Giải mã gói Json

String decrypted = decryptPayload(base64Input);

Serial.print("MQTT Received from ");

Serial.print(topic);

Serial.print(" - Content: ");

Serial.println(decrypted);

}

Hàm sendMessage() đóng gói dữ liệu từ các cảm biến vào 1 gói Json, thực hiện mã hóa gói Json bằng AES + base64 rồi publish lên Broker:

void sendMessage(float h, float t, float lux) {

DynamicJsonDocument doc(256);

doc["humidity"] = h;

doc["temperature"] = t;

doc["light"] = lux;

String json;

serializeJson(doc, json);

char encrypted\_base64[512];

uint16\_t input\_length = json.length();

// Mã hóa gói Json

aesLib.encrypt64(

(byte\*)json.c\_str(), input\_length,

encrypted\_base64,

aes\_key, 128, aes\_iv

);

// Publish gói tin lên topic, kích hoạt Qó để tranh mất gói

Serial.println("Sending JSON:");

Serial.println(json);

client.publish("wemos1/sensor", encrypted\_base64, true);

}

Hàm decryptPayload thực hiện giải mã gói Json nhận được từ phàn hồi của Broker:

String decryptPayload(String base64Input) {

byte decrypted[256];

// Gọi bộ đệm để giải mã

char base64Buffer[512];

base64Input.toCharArray(base64Buffer, base64Input.length() \* 2);

uint16\_t decryptedLength = aesLib.decrypt64(

base64Buffer, strlen(base64Buffer),

decrypted,

aes\_key, 128, aes\_iv

);

// Nếu gói tin rỗng thì báo lỗi

if (decryptedLength <= 0) {

Serial.println("Decryption failed");

return "";

}

// Loại bỏ khoảng trống thừa

decrypted[decryptedLength] = '\0'; // null terminate

return String((char\*)decrypted);

}

Dùng millis() thay vì delay để tránh callback bị lặp với vòng loop gây lỗi gói tin:

if (millis() - lastSend > 5000) {

float lux = lightMeter.readLightLevel();

float h = dht.readHumidity();

float t = dht.readTemperature();

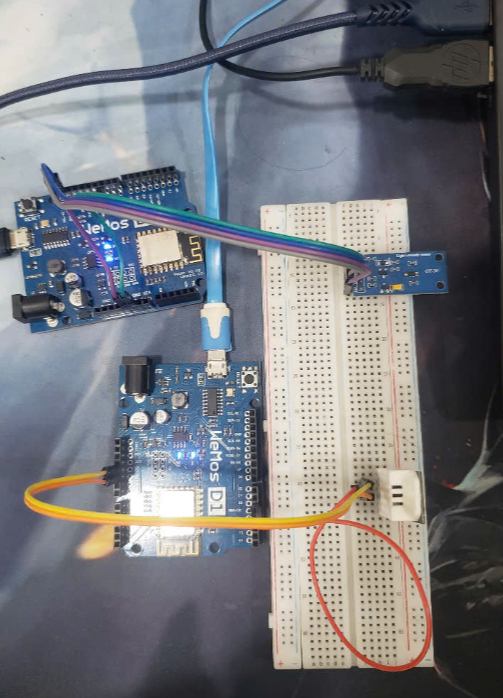
sendMessage(h, t, lux);

lastSend = millis();

}

### Demo:

Mô hình:

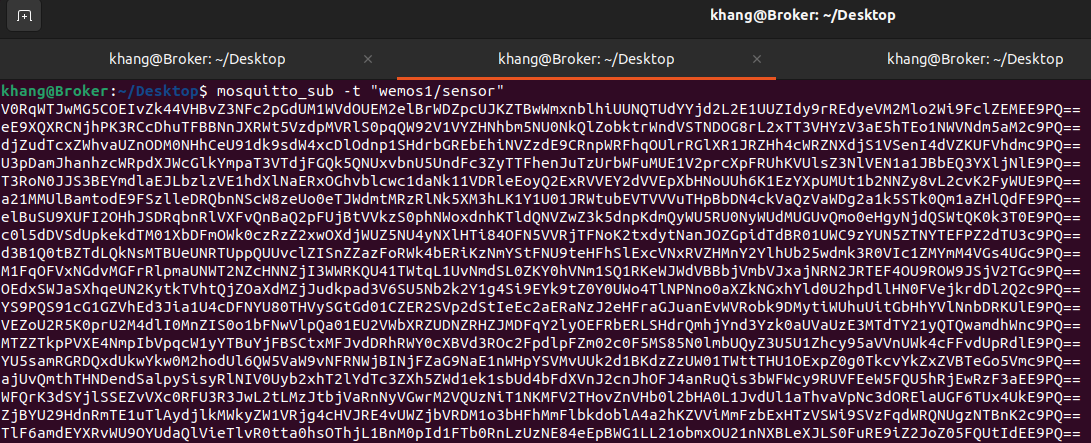


Dữ liệu sẽ được mỗi 5s trên mỗi Node:

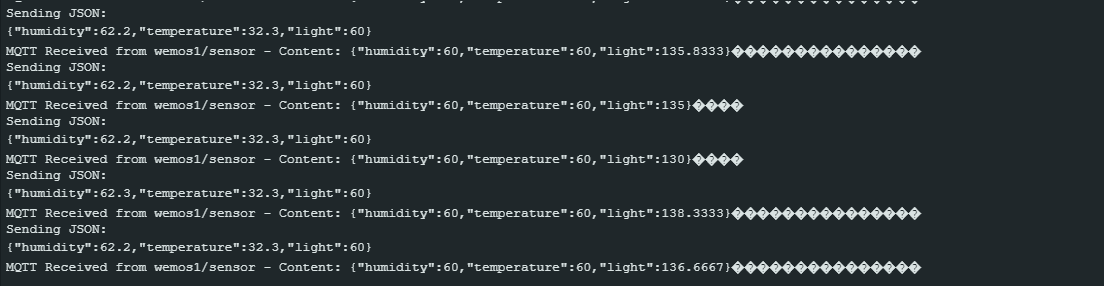
* Node 1:



* Topic wemos1/sensor:



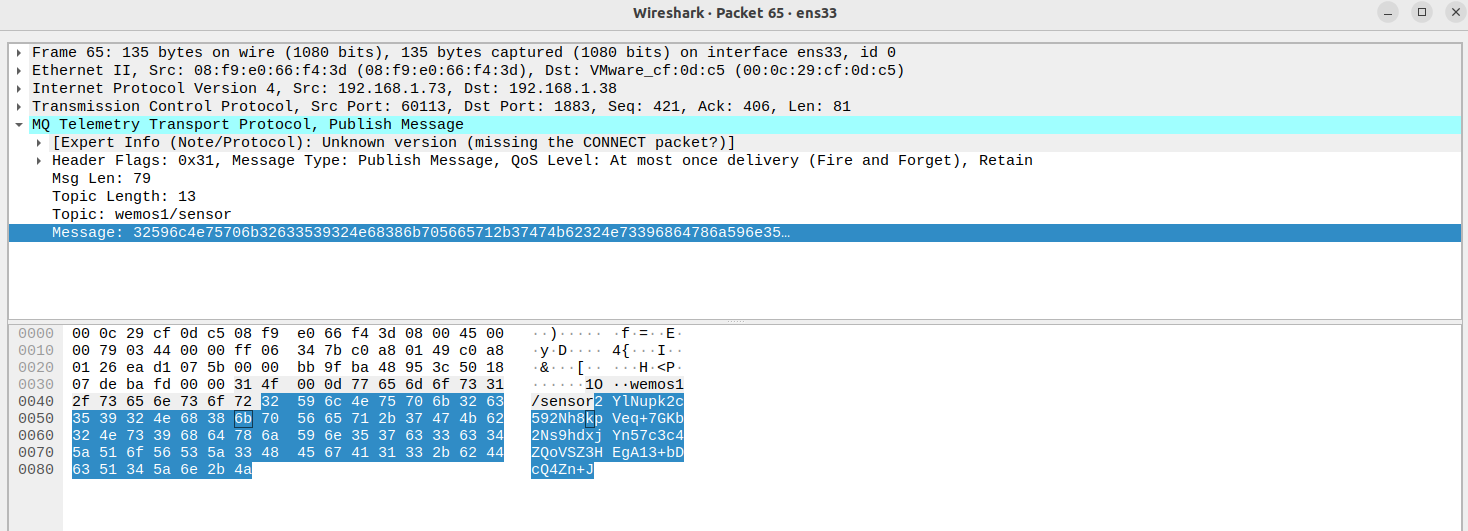
* Node 2:



* Topic wemos2/sensor:



Kiểm tra nghe lén trên Broker bằng wireshark, gói tin đã bị mã hóa hoàn toàn. Không thể nghe lén được:



# TÀI LIỆU THAM KHẢO

[1] Eclipse Foundation, “Mosquitto – An Open Source MQTT Broker,” GitHub Repository, [Online]. Available: https://github.com/eclipse-mosquitto/mosquitto. [Accessed: Apr. 13, 2025].

[2] D. Landman, “AESLib – AES Encryption Library for Arduino,” GitHub Repository, [Online]. Available: https://github.com/DavyLandman/AESLib. [Accessed: Apr. 13, 2025].

[3] B. Blanchon, “ArduinoJson – Efficient JSON serialization for embedded C++,” GitHub Repository, [Online]. Available: https://github.com/bblanchon/ArduinoJson. [Accessed: Apr. 13, 2025].