KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG - UIT

BÁO CÁO THỰC HÀNH

Lab 3

CÔNG NGHỆ INTERNET OF THINGS HIỆN ĐẠI

Làm quen với HTTP và MQTT trong mô hình IoT

GVHD: **Phan Trung Phát**

Lớp: **NT532.O21.MMCL.2**

Họ và tên	MSSV
Tống Võ Anh Thuận	21522652
Trịnh Vinh Đại	21521915
Lê Huỳnh Quanh Vũ	21522797

ĐÁNH GIÁ KHÁC (*):

Nội dung	Kết quả
Tổng thời gian thực hiện bài thực hành trung bình (1)	16 ngày
Link Video thực hiện (2) *(nếu có)*	Code n Video
Ý kiến (3) *(nếu có)* + Khó khăn + Đề xuất ...	
Điểm tự đánh giá (4)	10/10

(*): phần (1) và (4) bắt buộc thực hiện.

Phần bên dưới là báo cáo chi tiết của nhóm/cá nhân thực hiện.

WARNING LƯU HÀNH NỘI BỘ

Note: Due to some mistakes by our group, almost all MQTT topics in the video demo are labeled with "cl21/nhom2/check" when it should be "nhom1" instead. We apologize for the incorrect topic name, it was too late to reshoot the videos when we found out.

Câu hỏi 1. Lấy ý tưởng từ bài 5 - bài thực hành số 1 "thử tài đoán số", xây dựng kịch bản gồm 1 Wemos D1, 5 đèn LED. HTTP Server được dựng trên Wemos D1. Trò chơi được dừng lại khi người chơi không còn điểm. Bắt đầu với 5 đèn LED sáng từ trái sang phải và ngược lại, sau đó trò chơi sẽ ngẫu nhiên 1 số lượng đèn bất kỳ. Đèn được hiển thị trong khoảng thời gian là 2 giây sau đó tắt đi, vị trí của các đèn được thiết đặt một cách ngẫu nhiên. Người chơi cần nhanh trí đếm số lượng đèn và % 3, thực hiện chọn nút tương ứng trên giao diện website trong khoảng 2 giây tiếp theo. Nếu người dùng chọn đúng thì sẽ được cộng điểm và bị trừ điểm nếu chọn sai hoặc hết thời gian. Điểm số và tình trạng của trò chơi sẽ được hiển thị qua giao diện website.
Lưu ý: Sử dụng URL parameters (query string) để thể hiện việc chọn các con số trên giao diện website.

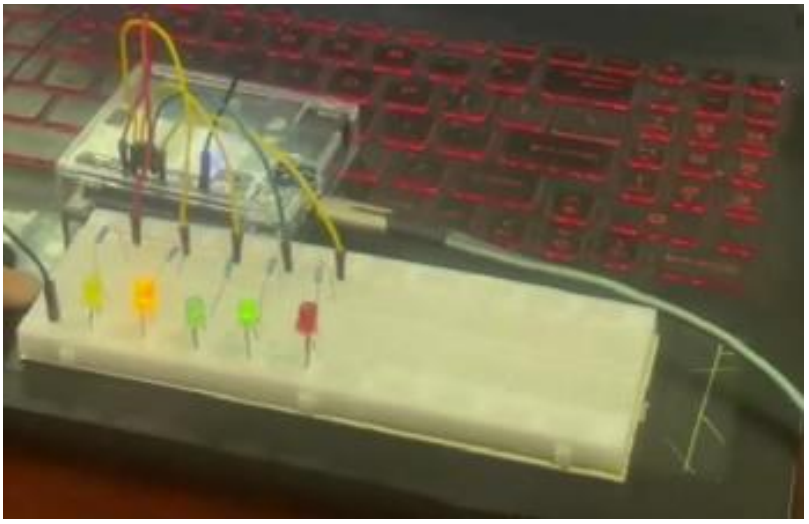
1. Setup:

Components in used:

- 5 LED.
- 5 resistors.
- 1 Wemos D1.
- 6 jumper wires (male-to-male).

Wire connection:

- 5 LED:
  - Anode: Use digital pins from D3 to D7 on Wemos. Those pins control LED output. Each led connect with one resistor.
  - Cathode: Connect to GND zone on bread board. This zone grounds the led.



2. Code explanation:

- Webserver setup:
  - This webserver display Score with three buttons named from Button 1 to Button 3.
  - When user click a button, it calls handleButtonClick function to take the request from Wemos, which is a led controller. Then update the score and isClicked status.
  - isClicked variable: a variable to tell the server that the user has already clicked the button.
  - checkTime function: a function used to check if the user has clicked any buttons. It is called every 2 seconds.

```
<script>
    let isClicked = false;
    let scr = 0;

    function updateScore(score) {
        document.getElementById('score').innerHTML = 'Score: ' + score;
    }

    function handleButtonClick(buttonId) {
      var xhr = new XMLHttpRequest();
      var url = "/" + buttonId; // Construct URL dynamically based on buttonId
      xhr.open('GET', url, true);
      xhr.onreadystatechange = function() {
        if (xhr.readyState == 4 && xhr.status == 200) {
          isClicked = true;
          scr = scr + parseInt(xhr.responseText);
          updateScore(scr);
        }
      };
      xhr.send();
    }

    function checkTime() {
      if (!isClicked) {
        scr--;
      }
      updateScore(scr);
      isClicked = false;
    }

    setInterval(checkTime, 2000);
</script>
```

- Wemos code:

```
void setup() {
// Settup for Wifi here
...
  startLed();
  server.on("/", handleRoot);
  server.on("/1", HTTP_GET, handleB1);
  server.on("/2", HTTP_GET, handleB2);
  server.on("/3", HTTP_GET, handleB3);
  server.begin();
  Serial.println("HTTP server started");
}

void loop() {
  server.handleClient();
  randomLED();
}

void handleRoot() {
  Serial.println("You called root page");
  String html = MAIN_page;
  server.send(200, "text/html", html);
}

void handleB1() {
  Serial.println("You press Button 1: ");
  checkNum(0);
}
```

```
1  void checkNum(int num) {
2    int score = 0;
3    if (ledNum % 3 == num)
4    {
5      Serial.print("Correct");
6      score++;
7    } else {
8      Serial.print("Incorrect");
9      score--;
10   }
11   Serial.println("");
12   server.send(200, "text/html", String(score));
13 }
```

- startLed: called once when wifi is connected, start leds from left to right, then vice versa.
- randomLED: called in loop function, random numbers of leds and position of those leds due to our algorithm.
- Server can handle four statuses:
    - handleRoot: call the main page.
    - handleButton(1-3): handle button 1 to 3.
    - handButton functions are called follow by user click on webserver. They can check the button using checkNum function. Then result to server.
    - checkNum: follow by the rule lednum % 3, if 0 then button 1 is correct, if 1 then button 2 is correct and if 2 then button 3 is correct.

Câu hỏi 2. Xây dựng kịch bản thiết bị gồm 1 Wemos D1, 1 cảm biến ánh sáng, 1 cảm biến khoảng cách và 3 đèn LED. Trong đó, máy tính cá nhân như một Server, sử dụng một ngôn ngữ lập trình bất kỳ để xây dựng các RESTful API (chủ yếu sử dụng hai phương thức GET và POST). Sử dụng Wemos D1 như một HTTP Client, định kỳ mỗi 5s Wemos D1 thu thập giá trị cảm biến ánh sáng và cảm biến khoảng cách, gửi các giá trị cảm biến với định dạng JSON trong payload của POST API ở trên. Đồng thời, Server sẽ trả về số lượng đèn sáng tùy theo cường độ ánh sáng. Nếu có người tiếp cận dựa vào giá trị của cảm biến khoảng cách thì trả về cho người dùng số lượng đèn sáng để cung cấp đủ ánh sáng. Nếu không có người tiếp cận thì tắt các đèn. Khi máy tính nhận được thông điệp thì xuất ra command line hoặc bất kỳ trình dạng nào có thể quan sát được. Các nội dung gửi đi và về đều ở dạng JSON.

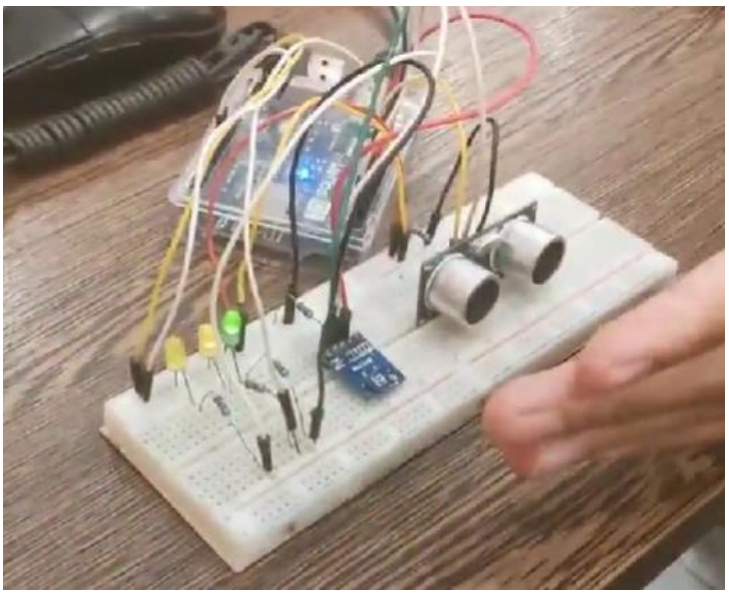
1. Setup:

Components in used:

- 1 Wemos D1.
- 1 light sensor.
- 1 ultrasonic sensor.
- 3 LED.
- 1 laptop.
- 5 resistors.

Wire connection:

- Light sensor:
  - VCC: connect to 5v zone with resistor on bread board. This help power the sensor.
  - GND: connect to GND zone on bread board. This grounds the sensor.
  - SCL: connect to D1 pin on Wemos. This pin carries the clock signal generated by the master device.
  - SDA: connect to D2 pin on Wemos. This pin carries the actual data being transmitted between the master and the slave devices.
- Ultrasonic sensor:
  - VCC: connect to 5v zone with resistor on bread board. This help power the sensor.
  - TRIG: connect to D6 pin on Wemos. This pin is used to trigger the sensor to send out an ultrasonic pulse.
  - ECHO: connect to D7 pin on Wemos. This pin is used to receive the echo of the ultrasonic pulse.
  - GND: connect to GND zone on bread board. This grounds the sensor.
- Three leds:
  - Anode: Use digital pins from D3 to D5 on Wemos. Those pins control led output. Each led connect with one resistor.
  - Cathode: connect to GND zone on bread board. This grounds the led.



2. Code explanation:

– Server code:

```go
import (
    "encoding/json"
    "fmt"
    "io"
    "net/http"
)

type SensorData struct {
    Error   bool   `json:"error"`
    Message string `json:"message,omitempty"`
    Data    struct {
        Distance float64 `json:"distance"`
        Light    float64 `json:"light"`
    } `json:"data"`
}
```

```go
// check light sensor value
lightsOn := 0
if sensorData.Data.Light < 100 {
    lightsOn = 3
} else if sensorData.Data.Light < 300 {
    lightsOn = 2
} else if sensorData.Data.Light < 500 {
    lightsOn = 1
}

// check distance senor value
if sensorData.Data.Distance > 6 {
    lightsOn = 0
}

fmt.Printf("Received data in POST request: Distance: %.2f, Light: %.2f\n",

responseData := struct {
    LightsOn int `json:"lightsOn"`
}{
    LightsOn: lightsOn,
}
responseJson, _ := json.Marshal(responseData)
w.WriteHeader(http.StatusOK)
w.Write(responseJson)
```

- When the server receives POST request from the client, which is Wemos, it decodes the body and gets the corresponding value in the JSON format defined in the struct SensorData.

- If the light level from the sensor is < 100, then return 3 LEDs. If the light level is < 300, then return 2 LEDs. If the light level < 500, then return 1 LED.
- If detected distance > 6, then return 0 led.
- After that, send the number of LED to light to the client.

– Wemos code:

```
void loop() {
  if ((millis() - lastTime) > timerDelay) {
    if(WiFi.status()== WL_CONNECTED){
      WiFiClient client;
      HTTPClient http;

      readDistance();
      readLight();

      http.begin(client, serverName);
      http.addHeader("Content-Type", "application/json");
      String jsonData = "{\"error\":false,\"message\":\"\",\"data\":{\"distance\":" + String(distance) + ",\"light\":" + String(light) + "}}";
      int httpResponseCode = http.POST(jsonData);

      String payload = "";
      if (httpResponseCode > 0) {
        payload = http.getString();
      }

      Serial.print("HTTP Response code: ");
      Serial.println(httpResponseCode);
      Serial.print("Data: ");
      Serial.println(payload);
      JsonDocument doc;
      deserializeJson(doc, payload);
      ledNum = doc["lightsOn"];
      Serial.print("LED num: ");
      Serial.println(ledNum);
      Serial.println("==========");

      if (ledNum == 0) {
        for (int i = 0; i < 3; i++) {
          digitalWrite(ledPins[i], LOW);
        }
      } else {
        for (int i = 0; i < 3; i++) {
          digitalWrite(ledPins[i], LOW);
        }
        for (int i = 0; i < ledNum; i++) {
          digitalWrite(ledPins[i], HIGH);
        }
      }
```

- The whole function is in loop function. It can execute every timeDelay, which is set in 5 seconds with WiFi is connected.
- At first, it read data from sensors through readDistance and readLight functions.
- Then create connections to the server, which is written in Go. Create a JSON string and parse the corresponding value to this string.
- After that, POST to the server with JSON string.
- When received a message from server, deserialize the message to get value, which is number of led to light, and light up LEDs.

Câu hỏi 3. SSH hoặc VNC vào Raspberry Pi được cung cấp sẵn để tiến hành kiểm tra hoạt động của MQTT Broker. Sử dụng một phần mềm MQTT Client bất kỳ (ví dụ như MQTTLens) để tiến hành publish vào topic "cl21/Y/nhomX/check", với X là số thứ tự nhóm. Trên Client 1 subcribe vào topic trên để nhận thông điệp được gửi từ Client 2.

1. Setup:

Components used:

- Raspberry Pi running as MQTT Broker provided by Lab E3.1.
- 2 laptops running mqtt-cli program.

2. Code explanation:

Note: The mqtt-cli program's source code is publicly available here: https://github.com/hivemq/mqtt-cli

Command explanation:

- On client 1, to subscribe to the topic "cl21/2/nhom1/check", run the following command:

```
mqtt-cli sub -h 192.168.120.88 -p 1883 -u mmcl21-2 --password
mmcl21-2@123 -t "cl21/2/nhom1/check"
```

- On client 2, to publish to the topic, run the following command:

```
mqtt-cli pub -h 192.168.120.88 -p 1883 -u mmcl21-2 --password
mmcl21-2@123 -t "cl21/2/nhom1/check" -m "test cau 3 lab 03"
```

Command explanation:

- "sub" command: To subscribe to a topic.
- "pub" command: To publish to a topic.
- "-h" flag: To specify the host (the broker).
- "-p" flag: To specify the port number.
- "-u" flag: To specify the username.
- "--password" flag: To specify the password.
- "-t" flag: To specify the topic name.
- "-m" flag: To specify the message.

Câu hỏi 4. Xây dựng kịch bản 1 Wemos D1, 1 Raspberry Pi, 1 máy tính cá nhân và 1 đèn LED. Raspberry Pi đóng vai trò là MQTT Broker. Triển khai MQTT Client trên Wemos D1 và máy tính cá nhân. Tại máy tính cá nhân người dùng thông qua topic "cl21/Y/nhomX/led", tiến hành publish các trạng thái của đèn LED. Tại Wemos D1, subscribe vào topic "cl21/Y/nhomX/led" để lắng nghe trạng thái của đèn, từ đó điều khiển bật hoặc tắt đèn LED.

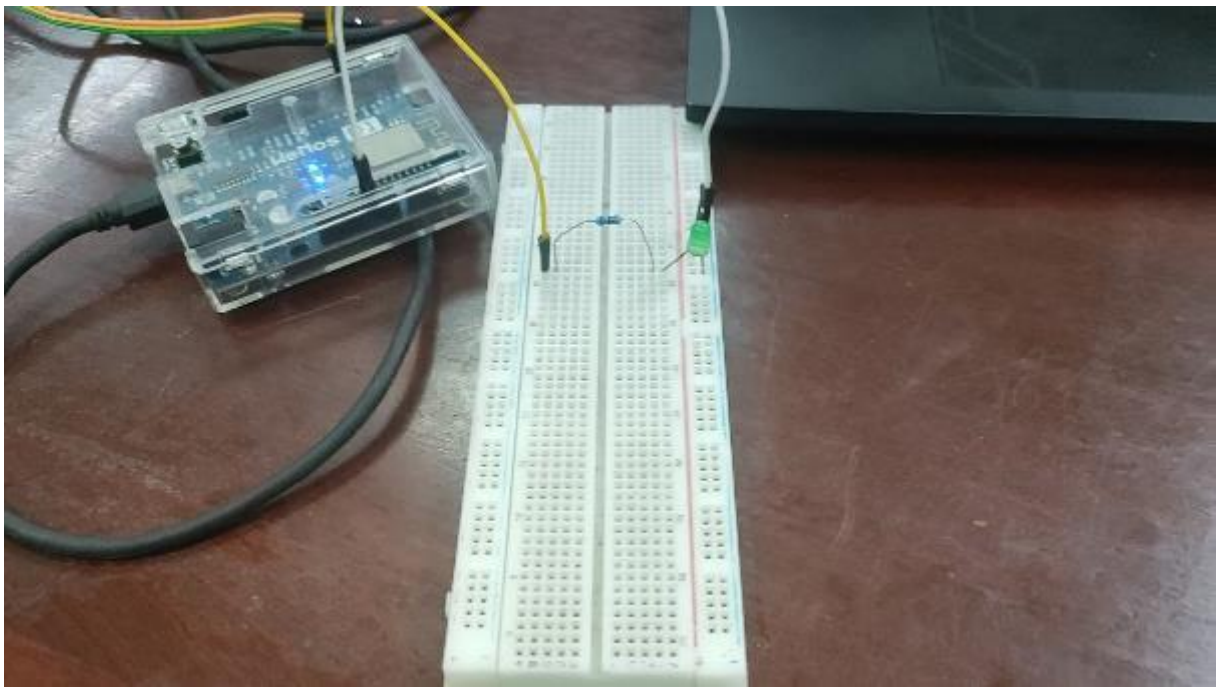
1. Setup:

Components used:

- 1 Wemos D1 board.
- 2 laptops.
- 1 LED.
- 1 resistor.

Wire connection:

- 5 LED:
    - Anode: Use digital pins from D3 on Wemos. Those pins control led output. Each led connect with one resistor.
    - Cathode: Connect to GND zone on breadboard. This zone grounds the led.



2. Code explanation:

Note: The majority of the MQTT-related code in this exercise was taken from here: https://github.com/knolleary/pubsubclient/blob/master/examples/mqtt_esp8266/mqtt_esp8266.ino

```
WiFiClient espClient;
PubSubClient client(espClient);

const char* topicToSubscribe = "cl21/2/nhom1/led";

void setup_wifi() {
  delay(10);
  // Connect to WiFi
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}
```

We use this code to register the requested topic "cl21/2/nhom1/led" and connect to WiFi. After successful connection, output the text "WiFi connected" and the IP address of the network.

```
// Check message content and control LED accordingly
if (message.equals("1")) {
  digitalWrite(ledPin, HIGH);  // Bật đèn LED
  Serial.println("LED turned ON");
} else if (message.equals("0")) {
  digitalWrite(ledPin, LOW);  // Tắt đèn LED
  Serial.println("LED turned OFF");
} else {
  Serial.println("Invalid message");
}
```

Print out the sent message. There are 3 cases:
- If the message is "1", turn on the LED.
- If the message is "0", turn off the LED.
- If the message is not 1 or 0, print the text "Invalid message".

```
58  void reconnect() {
59    // Loop until we're reconnected
60    while (!client.connected()) {
61      Serial.print("Attempting MQTT connection...");
62      // Create a random client ID
63      String clientId = "ESP8266Client-";
64      clientId += String(random(0xffff), HEX);
65      // Attempt to connect
66      if (client.connect(clientId.c_str())) {
67        Serial.println("connected");
68        // Once connected, subscribe to the topic
69        client.subscribe(topicToSubscribe);
70        Serial.print("Subscribed to: ");
71        Serial.println(topicToSubscribe);
72      } else {
73        Serial.print("failed, rc=");
74        Serial.print(client.state());
75        Serial.println(" try again in 5 seconds");
76        // Wait 5 seconds before retrying
77        delay(5000);
78      }
79    }
80  }
```

The reconnect function is called when the Wemos circuit is not connected to the network and needs to be reconnected.
- If connected, read the message from the subscribed message.
- If connection fails, print the failed connection result and return code, then continue to reconnect every 5 seconds.

Câu hỏi 5. Xây dựng kịch bản gồm 1 Wemos D1, 1 Raspberry Pi, 1 máy tính cá nhân, 1 cảm biến nhiệt độ, độ ẩm DHT22 và 1 đèn LED. Raspberry Pi đóng vai trò là MQTT Broker. Tại máy tính cá nhân, sử dụng một ngôn ngữ lập trình bất kỳ để subcribe vào lắng nghe dữ liệu cảm biến từ Wemos D1 thông qua topic "cl21/Y/nhomX/dht/value". Cho khoảng giá trị bình thường của nhiệt độ là từ 25 oC - 27oC hoặc độ ẩm 40% - 70%, nếu trong quá trình đo đạc phát hiện giá trị bất thường thì máy tính cá nhân publish tín hiệu bất thường vào topic "cl21/Y/nhomX/dht/detected". Wemos D1 subcribe vào topic "cl21/Y/nhomX/dht/detected", nếu nhận giá trị phát hiện bất thường thì chớp tắt đèn một cách liên tục để báo hiệu.

1. Setup:

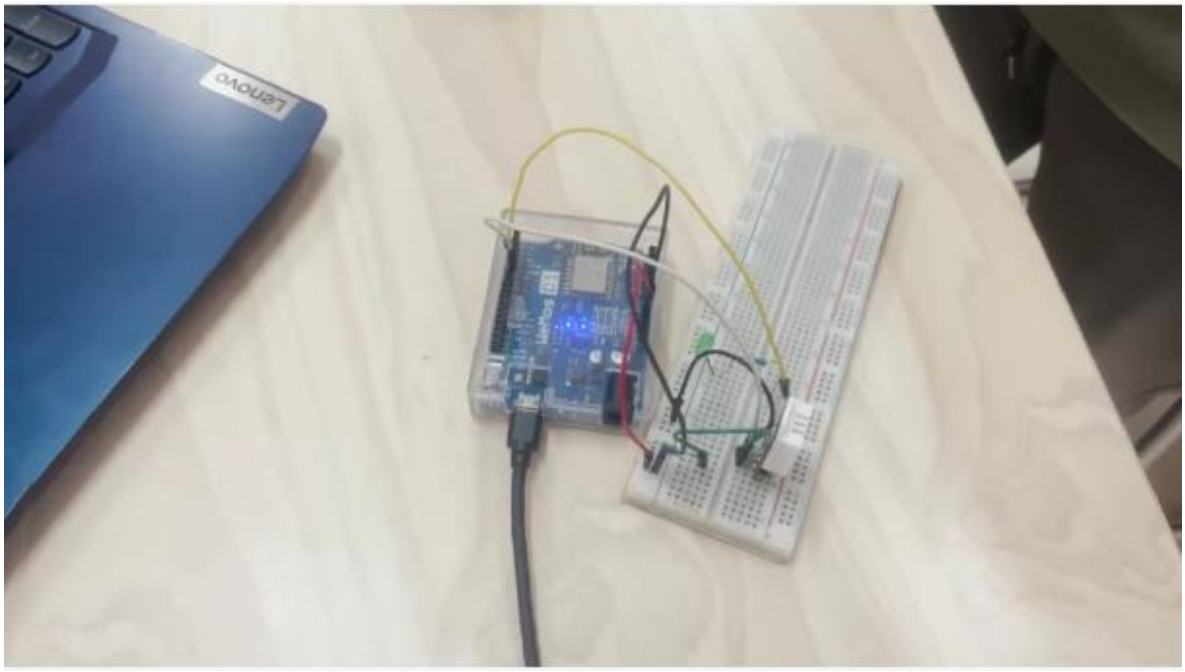
Components used:

- 1 Wemos D1 board.
- 1 laptop running an MQTT desktop program.
- 1 LED.

- 2 resistors.
- 1 DHT22 sensor.
- 1 breadboard.

Wire connection:

- DHT22:
  - VCC: connect to 5v zone on breadboard with resistor. This help supply power to sensor.
  - DATA: connect to D4 pin on Wemos. This pin is used to transmit data from the sensor to the microcontroller.
  - GND: connect to GND zone on breadboard. This grounds the sensor.
- LED:
  - Anode: Use digital pins from D3 on Wemos. Those pins control led output. Each led connect with one resistor.
  - Cathode: connect to GND zone on bread board. This grounds the led.



2. Code explanation:

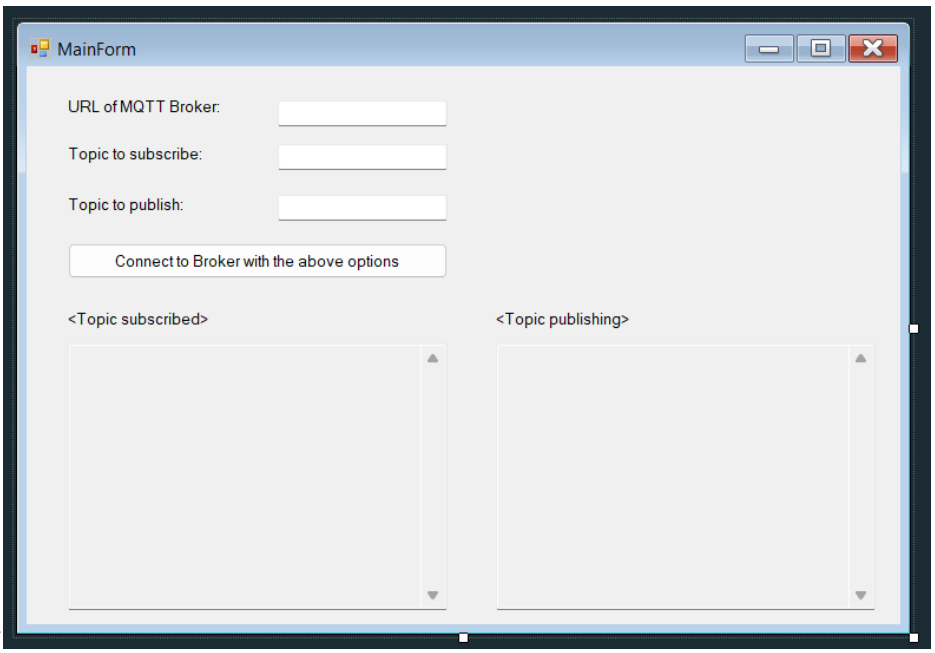
a. Wemos's code

```
void loop() {
  if (!client.connected()) {
    reconnect();
  }

  readDHT22();
  JsonDocument doc;
  doc["temperature"] = String(temp);
  doc["humidity"] = String(hum);

  char jsonString[128];
  serializeJson(doc, jsonString);

  Serial.println(jsonString);
  client.publish(topicVALUE, jsonString);
  Serial.println("Published to: " + String(topicVALUE));
  client.subscribe(topicDETECTED);
  delay(2000);

  client.loop();
}

void callback(char* topic, byte* payload, unsigned int length) {
  Serial.print("Message arrived [");
  Serial.print(topic);
  Serial.print("] ");
  for (int i = 0; i < length; i++) {
    Serial.print((char)payload[i]);
  }
  Serial.println();

  if ((char)payload[0] == '1') {
    blinkLed();
  }
}
```

- In loop functions, at first, Wemos attempts to connect to MQTT broker.
- If success, Wemos will read data from DHT22 and create a json string included those metrics.
- Wemos publish a json string to topic (cl21/2/nhom1/dht/value) and subcribe to topic (cl21/2/nhom2/dht/detected) to receive message.
- callback function use to receive message. If received message is '1', blink led. Else, do nothing.

b. MQTT desktop program's code:

The program was built using the Winforms UI framework (C#, .NET Core).



The code utilized the NuGet packet "**MQTTnet**" for all MQTT related operations.

```
using MQTTnet;
using MQTTnet.Client;
using MQTTnet.Packets;
using Newtonsoft.Json;


namespace mqtt_client
{
    public partial class MainForm : Form
    {
        private IMqttClient mqttClient;
        private string broker = "broker.hivemq.com";
        private string valueTopic = "cl21/2/nhom1/dht/value";
        private string detectTopic = "cl21/2/nhom1/dht/detected";

        public class SensorData
        {
            public double temperature { get; set; }
            public double humidity { get; set; }
        }
```

After defining the "mqttClient" variable, we go on to set some fixed values such as the MQTT broker's URL, the topic to subscribe to and the topic to publish to as required. We also define a class to receive data from the sensor in JSON format which includes temperature and humidity.

```
private async Task OnMessageReceived(MqttApplicationMessageReceivedEventArgs e)
 {
     string jsonString = Encoding.UTF8.GetString(e.ApplicationMessage.Payload);
     string message = "";
     double temperature = 25;
     double humidity = 40;
     bool alarm = false;

     try
     {
         SensorData data = JsonConvert.DeserializeObject<SensorData>(jsonString);
         temperature = data.temperature;
         humidity = data.humidity;
         message = "Temperate: " + temperature + ", Humidity: " + humidity + "\n";

         //Check values for anomaly
         if(temperature < 25 || temperature > 27 || humidity < 40 || humidity > 70)
         {
             alarm = true;
         }
     }
```

After subscribing to the predefined topic, we set up an asynchronous task for handling the received messages. First, we get the message string from the MQTT payload using UTF8 encoding. Since this is JSON data, we have to deserialize the string into our SensorData class object to get the temperature and humidity values. Now we just need to set up conditions for these values, if they go below or over a certain threshold, we set the bool “alarm” to true.

```
if(alarm)
{
    await Publish("1");
}
else
{
    await Publish("0");
}
```

If “alarm” is true, we will publish the message “1”, else we will publish “0”.

```
private void updateSubRcvTB(string msg)
{
    subRcvTB.Text += msg + System.Environment.NewLine;
}

private void updatePubNotiTB(string msg)
{
    pubNotiTB.Text += msg + System.Environment.NewLine;
}
```

Every time a message is received or published, the task will also call one of the two functions above to update the text boxes for the users to see on their screen.

Note: The logic behind the Connection(), Subscribe(), and Publish() functions is rather straightforward in the source code so it will not be mentioned in detail here. All of them involve using an asynchronous task to either connect, subscribe, or publish then add an event handler for when a message is received or sent successfully.

YÊU CẦU CHUNG

1) Đánh giá

- Chuẩn bị tốt các yêu cầu đặt ra trong bài thực hành.
- Sinh viên hiểu và tự thực hiện được bài thực hành, trả lời đầy đủ các yêu cầu đặt ra.
- Nộp báo cáo kết quả chi tiết những đã thực hiện, quan sát thấy và kèm ảnh chụp màn hình kết quả *(nếu có)*; giải thích cho quan sát *(nếu có)*.
- Sinh viên báo cáo kết quả thực hiện và nộp bài.

2) Báo cáo

- File .PDF hoặc .docx. Tập trung vào nội dung, giải thích.
- Nội dung trình bày bằng Font chữ Cambria hoặc Times New Roman (*tuy nhiên, phải chuyển đổi hết báo cáo này sang 1 font chữ thống nhất*) – cỡ chữ 13. Canh đều (Justify) cho văn bản. Canh giữa (Center) cho ảnh chụp.
- Đặt tên theo định dạng: Mã lớp-LabX_MSSV1_MSSV2. (trong đó X là Thứ tự buổi Thực hành).

  Ví dụ: NT532.O21.1-Lab01_25520001_25520002
- Nếu báo cáo có nhiều file, nén tất cả file vào file .ZIP với cùng tên file báo cáo.
- Không đặt tên đúng định dạng – yêu cầu, sẽ KHÔNG chấm điểm bài thực hành.
- Nộp file báo cáo trên theo thời gian đã thống nhất tại courses.uit.edu.vn.

Bài sao chép, trễ, … sẽ được xử lý tùy mức độ vi phạm.

HẾT