**BÁO CÁO TỔNG HỢP**

**TUẦN 1. BÁO CÁO BÀI TẬP TUẦN I VỚI GIAO THỨC HTTP**

Khai báo các thư viện cần dung

Kết quả thực hiện chương trình được hiện thị sau mỗi đoạn code.

Tại mục 1.1 tạo 3 request gồm:

Request thứ nhất để update dữ liệu bằng các thủ công tức là điền các trường field1 và field2 trực tiếp vào request.

Request thứ hai để thực hiện gửi dữ liệu bằng file json chi tiết được trình bày ở đoạn code sau.

Request thứ ba dùng để lấy dữ liệu về dưới dạng json.

Tại mục 1.2 là một hàm nhận vào một request và trả về các giá trị Temperature và Humidity được lấy ra từ các trường field1 và field2 của file json.



Mục 1.3 thực hiện gửi dữ liệu bằng hàm phương thức requests.get() của python. Trả về thành công nếu dữ nhận được mã code là 200 và có lỗi nếu nhận được mã khác.

Mục 1.4 là giá trị trả về sau khi thực hiện mục 1.3.

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phông chữ

Mô tả được tạo tự động

Mục 1.5 thực hiện cập nhật các giá trị bằng file json. Đầu tiên tạo một file json với các trường field1 và field2 lần lượt là 60 và 70. url = request\_update\_data\_2 ở mục 1.1.

Sau đó, gọi phương thức request.post() của python. Trả về thành công nếu như mà trả về là 200 và có lỗi xảy ra nếu như gặp mã lỗi khác.

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, màn hình

Mô tả được tạo tự động

**TUẦN 2. BÁO CÁO BÀI TẬP TUẦN II VỚI GIAO THỨC MTTQ**

1. **Khai báo các thư viện và các giá trị khởi tạo.**

**Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động**

1. **Viết hàm thực hiện kết nối một client với MQTT Broker qua client\_id, port và broker.**

*Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động*

*Viết hàm thiết lập kết nối mqtt của client*

1. **Thiết lập một file dạng json để gửi từ publish tới subscribe với một số trường đã được định nghĩa. Viết hàm gửi.**

*Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động*

*Thiết lập một kiểu data gửi dạng json và hàm thực hiện client thực hiện gửi file này. Hàm* ***run\_publish()*** *sẽ thực hiện kết nối và gửi dữ liệu*

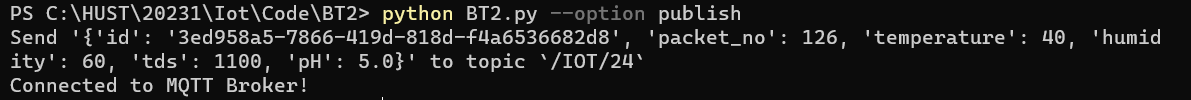
1. **Viết hàm lấy dữ liệu được gửi từ publish tới subscribe và hàm thực thi chương chình main.**

*Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm

Mô tả được tạo tự động*

*Hàm để thiết lập kết nối và trả về kết quả khi publish gửi dữ liệu gì lên subcribe. Hàm main để thực hiện chương trình.*

1. **Kết quả thực hiện chương trình và MQTTBox**
2. **Chạy chương trình để gửi file json**

****

**Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, số

Mô tả được tạo tự động**

1. **Chạy chương trình khi gửi dữ liệu từ MQTTBox**

**Ảnh có chứa văn bản, Phông chữ, ảnh chụp màn hình

Mô tả được tạo tự động**

**Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Trang web

Mô tả được tạo tự động**

**Link github:** [kaisacht/IOT (github.com)](https://github.com/kaisacht/IOT)

**TUẦN 3. Báo cáo bài tập tuần 3**

1. **Code lớp Publish.py**

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm

Mô tả được tạo tự động

Tạo các kết nối và thiết lập một file json

1. **Code lớp Subscribe.py**

Ảnh có chứa văn bản, ảnh chụp màn hình, màn hình, phần mềm

Mô tả được tạo tự động

**3 Kết quả thực hiện trên terminal và rabbitMQ**

Ảnh có chứa văn bản, hàng, số, Phông chữ

Mô tả được tạo tự động

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Mô tả được tạo tự động

**TUẦN 4. BÁO CÁO BÀI TẬP TUẦN IV VỚI ESP32**

1. **Khai báo các thư viện và các giá trị khởi tạo.**
2. #include <LiquidCrystal\_I2C.h>
3. #include "DHTesp.h"
4. LiquidCrystal\_I2C lcd(0x27,20,4);
5. #define DHT\_PIN 15
6. #define LED\_PIN 26
7. #define PIR\_PIN 13
8. DHTesp dhtSensor;
9. **Khởi tạo các giá trị**

void setup() {

  // put your setup code here, to run once:

  dhtSensor.setup(DHT\_PIN, DHTesp::DHT22);

  pinMode(LED\_PIN, OUTPUT);

  pinMode(PIR\_PIN, INPUT);

}

1. **Thực hiện vòng lặp để hiển thị dữ liệu liên tục**
2. int value = 1 ;
3. void loop() {
4. // put your main code here, to run repeatedly:
5. lcd.init(); // initialize the lcd
6. // Print a message to the LCD.
7. lcd.backlight();
8. TempAndHumidity data = dhtSensor.getTempAndHumidity();
9. int temp = data.temperature;
10. int humid = data.humidity;
11. lcd.setCursor(0,0);
12. lcd.print("Temperature:" + String(temp) +"C");
13. lcd.setCursor(0,1);
14. lcd.print("Humidity:" +  String(humid) + "%");
15. value = digitalRead(PIR\_PIN);
16. if(value == 1){
17. digitalWrite(LED\_PIN, HIGH);
18. }else{
19. digitalWrite(LED\_PIN, LOW);
20. }
21. delay(1000);
22. }
23. lcd.print("Temperature:" + String(temp) +"C");
24. lcd.setCursor(0,1);
25. lcd.print("Humidity:" +  String(humid) + "%");
26. value = digitalRead(PIR\_PIN);
27. if(value == 1){
28. digitalWrite(LED\_PIN, HIGH);
29. }else{
30. digitalWrite(LED\_PIN, LOW);
31. }
32. delay(1000);
33. }
34. **Kết quả mô phỏng nhiệt độ**

Ảnh có chứa văn bản, màn hình, biểu đồ, ảnh chụp màn hình

Mô tả được tạo tự động

1. **Mô phỏng chuyển động**

Ảnh có chứa văn bản, màn hình, biểu đồ, ảnh chụp màn hình

Mô tả được tạo tự động

1. **Link project**

[BTT4 Lê Anh Vũ - Wokwi ESP32, STM32, Arduino Simulator](https://wokwi.com/projects/379709635231761409)

**TUẦN 5. BÁO CÁO TUẦN BÀI TẬP 5**

1. **HTTP Get**
   1. **Khai báo các thư viện và các giá trị khởi tạo**

#include <LiquidCrystal\_I2C.h>

#include "DHTesp.h"

#include <WiFi.h>

#include <HTTPClient.h>

#include <ArduinoJson.h>

LiquidCrystal\_I2C lcd(0x27,20,4);

#define DHT\_PIN 15

#define LED\_PIN 26

#define PIR\_PIN 13

DHTesp dhtSensor;

const String serverName = "HTTP://postman-echo.com/get";

void setup() {

  Serial.begin(115200);

  Serial.println("Connecting to WiFi");

  WiFi.begin("Wokwi-GUEST", "",6);

  // put your setup code here, to run once:

  dhtSensor.setup(DHT\_PIN, DHTesp::DHT22);

  pinMode(LED\_PIN, OUTPUT);

  pinMode(PIR\_PIN, INPUT);

}

* 1. **Thực hiện vòng lặp và thực hiện lấy nhiệt độ và độ ẩm từ cảm biến bằng các phương thức đã được cài đặt được trình bày cụ thể trong bài tập tuần 4 sau đó sẽ tạo một request get lên sever bằng cách kết hợp các giá trị nhiệt độ và độ ẩm từ sensor.**

int value = 1 ;

void loop() {

  HTTPClient HTTP;

  TempAndHumidity data = dhtSensor.getTempAndHumidity();

  int temp = data.temperature;

  int humid = data.humidity;

  if (WiFi.status() ==  WL\_CONNECTED){

    String params = "?temp=" + String(temp) +"&humid=" + String(humid);

    String serverPath = serverName + params;

    HTTP.begin(serverPath.c\_str());

    int HTTPResponseCode = HTTP.GET();

    if (HTTPResponseCode == 200){

      String payload  = HTTP.getString();

      Serial.println(payload);

    } else{

      Serial.println("error");

      Serial.println(HTTPResponseCode);

    }

    delay(3000);

  }

1. **Kết quả thực hiện chương trình với HTTP get.**

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự động

1. **Link bài mô phỏng**

[HTTPGet Lê Anh Vũ - Wokwi ESP32, STM32, Arduino Simulator](https://wokwi.com/projects/380339619700505601)

1. **HTTP Post**
   1. **Khai báo các thư viện**

Phần này hoàn toàn giống với trong phần 1 phương thức HTTP get ( Nên không trình bày lại)

**2.2 Thực hiện vòng lặp và lấy các giá trị nhiệt độ và độ ẩm như trong phần trước phần này chỉ khác ở cách thực hiện HTTP post.**

void loop() {

  HTTPClient HTTP;

  TempAndHumidity data = dhtSensor.getTempAndHumidity();

  int temp = data.temperature;

  int humid = data.humidity;

  if (WiFi.status() ==  WL\_CONNECTED){

    HTTP.begin(serverName);

    HTTP.addHeader("Content-Type", "application/x-www-form-urlencoded");

    String HTTPRequestData = "&temp=" + String(temp) + "&humid=" + String(humid);

**Serial**.println(HTTPRequestData);

    int HTTPResponseCode = HTTP.POST(HTTPRequestData);

    if (HTTPResponseCode == 200){

      String payload  = HTTP.getString();

**Serial**.println(payload);

    } else{

**Serial**.println("error");

**Serial**.println(HTTPResponseCode);

    }

    delay(3000);

  }

* 1. **Kết quả mô phỏng trên wokwi**

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, màn hình

Mô tả được tạo tự động

* 1. **Link mô phỏng HTTP post**

[HTTPPost Lê Anh Vũ - Wokwi ESP32, STM32, Arduino Simulator](https://wokwi.com/projects/380922161513781249)

1. **HTTP post Json**
   1. **Khai báo thư viện và các giá trị khởi tạo tương tự như hai phần trước chỉ cần thêm một thư viện khác để làm việc với file json.**
   2. **Thực hiện vòng lặp để lấy nhiệt độ và độ ẩm từ cảm biến sau đó đóng gói trong file json bằng thư viện và thực hiện gửi đi.**
2. void loop() {
3. HTTPClient http;
4. TempAndHumidity data = dhtSensor.getTempAndHumidity();
5. int temp = data.temperature;
6. int humid = data.humidity;
7. if (WiFi.status() ==  WL\_CONNECTED){
8. http.begin(serverName);
9. http.addHeader("Content-Type", "application/json");
10. DynamicJsonDocument doc(1024);
11. String jsonstr;
12. JsonObject root = doc.to<JsonObject>();
13. root["temperature"] = temp;
14. root["humidity"] = humid;
15. serializeJson(doc, jsonstr);
16. String httpRequestData = jsonstr;
17. Serial.println(httpRequestData);
18. int httpResponseCode = http.POST(httpRequestData);
19. if (httpResponseCode == 200){
20. String payload  = http.getString();
21. Serial.println(payload);
22. } else{
23. Serial.println("error");
24. Serial.println(httpResponseCode);
25. }
26. delay(3000);
27. }

**3.3 Kết quả mô phỏng trên wokwi**

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Biểu tượng máy tính

Mô tả được tạo tự động

* 1. **Link bài mô phỏng với HTTP post json.**

[HTTPPostJson Lê Anh Vũ - Wokwi ESP32, STM32, Arduino Simulator](https://wokwi.com/projects/380921788926369793)

**TUẦN 6. BÁO CÁO TUẦN BÀI TẬP 6 MQTT VỚI ESP32**

1. **Khởi tạo các thư viện và các giá trị tham số.**

#include <LiquidCrystal\_I2C.h>

#include "DHTesp.h"

#include <WiFi.h>

#include <HTTPClient.h>

#include <ArduinoJson.h>

#include <PubSubClient.h>

LiquidCrystal\_I2C lcd(0x27,20,4);

#define DHT\_PIN 15

#define LED\_PIN 26

#define PIR\_PIN 13

const char\* mqtt\_server = "broker.hivemq.com";

WiFiClient espClient;

PubSubClient client(espClient);

unsigned long lastMsg = 0;

#define MSG\_BUFFER\_SIZE (50)

char msg[MSG\_BUFFER\_SIZE];

int value = 0;

1. **Hàm callback dung để điều khiển bóng đèn với các giá trị nhận được từ gói tin tương ứng.**
2. void callback(char\* topic, byte\* payload, unsigned int length) {
3. Serial.print("Message arrived [");
4. Serial.print(topic);
5. Serial.print("] ");
6. for (int i = 0; i < length; i++) {
7. Serial.print((char)payload[i]);
8. }
9. if ((char)payload[0] == '1') {
10. digitalWrite(LED\_PIN, LOW);
11. } else {
12. digitalWrite(LED\_PIN, HIGH);
13. }
14. }
15. **Hàm kết nối với MQTT.**

void reconnect() {

  // Loop until we're reconnected

  while (!client.connected()) {

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

    // Create a random client ID

    String clientId = "ESP8266Client-";

    clientId += String(random(0xffff), HEX);

    // Attempt to connect

    if (client.connect(clientId.c\_str())) {

      Serial.println("connected");

      // Once connected, publish an announcement...

      client.publish("/IOT/T6", "1");

      // ... and resubscribe

      client.subscribe("/IOT/T6");

    } else {

      Serial.print("failed, rc=");

      Serial.print(client.state());

      Serial.println(" try again in 5 seconds");

      // Wait 5 seconds before retrying

      delay(5000);

    }

  }

}

1. **Hàm setup() chỉ chạy một lần trong quá trình.**

DHTesp dhtSensor;

void setup() {

  Serial.begin(11150);

  WiFi.begin("Wokwi-GUEST", "",6);

  // put your setup code here, to run once:

  dhtSensor.setup(DHT\_PIN, DHTesp::DHT22);

  pinMode(LED\_PIN, OUTPUT);

  pinMode(PIR\_PIN, INPUT);

  client.setServer(mqtt\_server, 1883);

  client.setCallback(callback);

  lcd.init();

  lcd.backlight();

}

1. **Vòng lặp để gửi dữ liệu liên tục nhiệt độ và độ ẩm về máy sever MQTT với 5s một lần.**

void loop() {

  if (!client.connected()) {

    reconnect();

  }

  client.loop();

  TempAndHumidity data = dhtSensor.getTempAndHumidity();

  int temp = data.temperature;

  int humid = data.humidity;

  lcd.setCursor(0,0);

  lcd.print("Temperature:" + String(temp) +"C");

  lcd.setCursor(0,1);

  lcd.print("Humidity:" +  String(humid) + "%");

  String mg = "Temperature: " + String(temp) + "C  Humidity: " + String(humid) + "%";

  unsigned long now = millis();

  ++value;

  // Convert 'value' to a C-style string

  char valueStr[10]; // Assuming a maximum length for the number

  snprintf(valueStr, sizeof(valueStr), "%d", value);

  // Concatenate 'valueStr' to 'mg'

  mg += " Value: ";

  mg += valueStr;

  // Convert 'mg' (String) to a C-style string (char array)

  char msg[MSG\_BUFFER\_SIZE];

  mg.toCharArray(msg, MSG\_BUFFER\_SIZE);

  Serial.print("Publish message: ");

  Serial.println(msg);

  client.publish("/IOT/T6", msg);

  delay(5000);

}

1. **Kết quả mô phỏng.**
   1. **Điều khiển bóng đèn sáng với payload = “0”.**

**Ảnh có chứa văn bản, phần mềm, Phần mềm đa phương tiện, Biểu tượng máy tính

Mô tả được tạo tự động**

* 1. **Điều khiển bóng đèn sáng với payload = “1”.**

**Ảnh có chứa văn bản, phần mềm, Biểu tượng máy tính, Phần mềm đa phương tiện

Mô tả được tạo tự động**

* 1. **Gửi nhiệt độ và độ ẩm liên tục cứ 5s một lần.**

**Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Biểu tượng máy tính

Mô tả được tạo tự động**

**Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Biểu tượng máy tính

Mô tả được tạo tự động**

1. **Link bài mô phỏng**

[BTT6 Lê Anh Vũ - Wokwi ESP32, STM32, Arduino Simulator](https://wokwi.com/projects/380981340459403265)

**TUẦN 7. BÁO CÁO BÀI TẬP TUẦN 7 XÂY DỰNG WEB API**

1. **Package controller**

Lớp SensorData

namespace WA.Models

{

    public class SensorData

    {

        public int Id { get; set; }

        public string Name { get; set; }

        public float Value { get; set; }

        public DateTime ReceiveTime { get; set; }

    }

}

Gồm bốn trường thuộc tính ID, name, value và receiveTime.

Lớp WebAPIDemoConext

using Microsoft.EntityFrameworkCore;

using System.Collections.Generic;

namespace WA.Models

{

    public class WebAPIDemoContext : DbContext

    {

        public WebAPIDemoContext(DbContextOptions<WebAPIDemoContext> options) : base(options) { }

        public DbSet<WA.Models.SensorData> SensorDatas { get; set; }

        protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

        {

            optionsBuilder.UseSqlite("Data Source=Application.db;Cache=Shared");

            base.OnConfiguring(optionsBuilder);

        }

    }

}

1. **Package Controller**

Lớp SensorDatasController

Gồm các phương thức get, put, post, delete được tạo ra bằng cách tự động như trong slide.

1. **Package Migrations**

Package này nhiệm vụ khởi tạo một bảng trong cơ sở dữ liệu với các trường dữ liệu như trong lớp model.

**Hầu như các lớp đều tự sinh mà không cần thực hiện code.**

1. **Kết quả thực hiện**
   1. **Giao diện**

**Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Biểu tượng máy tính

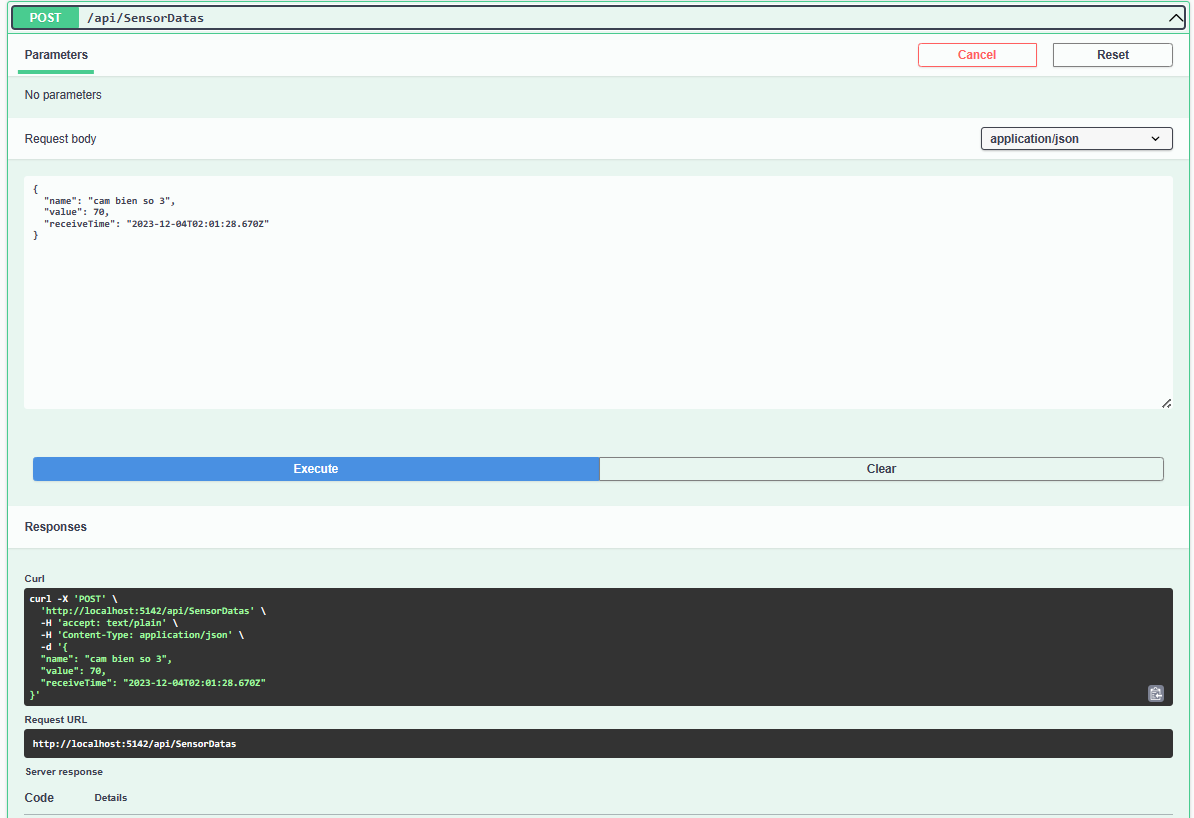
Mô tả được tạo tự động**

* 1. **Thực hiện phương thức GET**
     1. **Ảnh có chứa phần mềm, máy tính, ảnh chụp màn hình, văn bản

        Mô tả được tạo tự độngThực hiện trên Web**
     2. **Kiểm thử trên postman**

**Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự động**

* 1. **Thực hiện phương thức POST**
     1. **Thực hiện trên Web**
     2. **Kiểm thử GET sau khi POST**

**Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự động**

* 1. **Thực hiện phương thức PUT (UPDATE)**
     1. **Thực hiện trên Web**

**Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Biểu tượng máy tính

Mô tả được tạo tự động**

* + 1. **Kiểm thử trên postman dữ liệu sau khi sửa.**

**Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Mô tả được tạo tự động**

* 1. **Thực hiện phương thức DELETE**
     1. **Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, số

        Mô tả được tạo tự độngThực hiện trên Web**
     2. **Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

        Mô tả được tạo tự độngKiểm thử trên postman bản ghi có id bằng 10 đã bị xóa.**