BÁO CÁO GIỮA KÌ

1. Các linh kiện và cảm biến sử dụng trong đồ án
   - **ESP32**: Thu thập dữ liệu từ cảm biến, giao tiếp với hệ thống máy chủ bằng Mqtt thông qua WiFi và nhận lệnh điều khiển theo chiều ngược lại

Yêu Thích+ ESP32 WROOM 32 MICRO-C 30PIN CP2102, DEVKIT WIFI BLUETOOTH

5.0 ★★★★★ | 3 Đánh Giá | Đã Bán 40 | Tố cáo

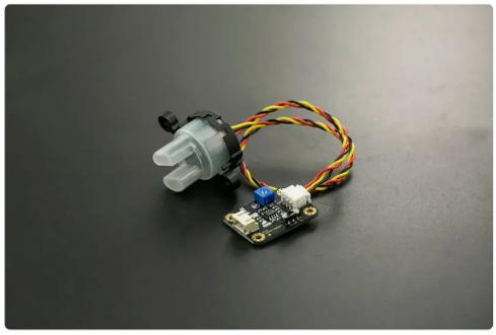
₫144.000 - ₫154.000

Voucher Của Shop | Giảm ₫5k | Giảm ₫10k

Voucher của Shop

   - **Cảm biến độ đục**: đo các hạt lơ lửng trong nước

Cảm biến đo độ đục của nước DFRobot Gravity: Analog Turbidity Sensor For Arduin



Loại: Default | Thương hiệu: DFRobot
Tình trạng: Còn hàng | Mã sản phẩm: HS0179

Giá bán:

320.000₫

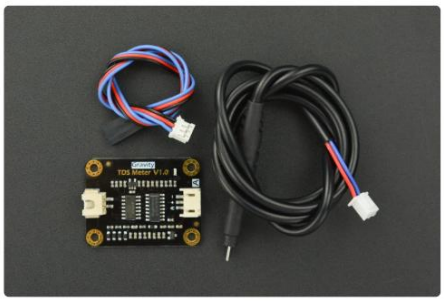
Số lượng: − 1 +

THÊM VÀO GIỎ

MUA NGAY | Liên hệ Zalo OA Hshop — Để được tư vấn và hỗ trợ ngay!!!

   - **Cảm biến TDS**: đo tổng chất rắn hòa tan trong nước (TDS - Total Dissolved Solids) giúp xác định chất lượng nước

Cảm biến tổng chất rắn hòa tan DFRobot Gravity: Analog TDS Sensor/Meter for Ard



Loại: Default | Thương hiệu: DFRobot
Tình trạng: Còn hàng | Mã sản phẩm: HS1472

Giá bán:

365.000₫

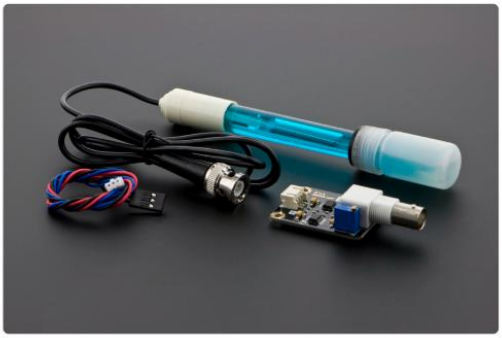
Số lượng: − 1 +

THÊM VÀO GIỎ

MUA NGAY | Liên hệ Zalo OA Hshop — Để được tư vấn và hỗ trợ ngay!!!

- **Cảm biến PH**

Cảm biến độ pH DFRobot Gravity: Analog pH Sensor / Meter Kit For Arduino



Loại: **Default**
Tình trạng: **Còn hàng**
Thương hiệu: **DFRobot**
Mã sản phẩm: **HS0180**

Giá bán:

760.000₫

Số lượng: 1

THÊM VÀO GIỎ

MUA NGAY

Liên hệ **Zalo OA Hshop**
Để được tư vấn và hỗ trợ ngay!!!

- **Cảm biến nhiệt độ**

Cảm biến nhiệt độ MKE-S15 DS18B20 waterproof temperature sensor

Loại: **Default**
Tình trạng: **Còn hàng**
Thương hiệu: **MakerLab.vn**
Mã sản phẩm: **HS1523**

Giá bán:

56.000₫

Title: Không kèm cáp (Not need cable)

- Không kèm cáp (Not need cable)
- Kèm cáp MakerEDU XH2.54 - XH2.54 3Wires 20cm cable
- Kèm cáp MakerEDU XH2.54 - Male Pin 3Wires 20cm cable

Số lượng: 1

- **Module LoRa**: truyền dữ liệu cảm biến đến hệ thống máy chủ từ xa

Module RF SPI Lora SX1278 433MHz Ra-02 Ai-Thinker Breakout

Loại: **Default**
Tình trạng: **Còn hàng**
Thương hiệu: **Ai-Thinker + MakerLab.vn**
Mã sản phẩm: **Đang cập nhật**

Giá bán:

135.000₫

Số lượng: 1

THÊM VÀO GIỎ

MUA NGAY

Liên hệ **Zalo OA Hshop**
Để được tư vấn và hỗ trợ ngay!!!

- **Module Relay 5V**: dung để bật tắt máy máy bơm khi các dữ liệu cảm biến vượt ngưỡng

Mạch 2 Relay Opto chọn mức kích High/Low (5/12/24VDC)

Loại: Default
Tình trạng: Còn hàng
Thương hiệu: Import
Mã sản phẩm: HS0998C

Giá bán:
35.000₫

Title: 5VDC
5VDC | 12VDC | 24VDC

Số lượng: 1

THÊM VÀO GIỎ

- **Máy bơm DC 12V**: Bơm nước theo lệnh điều khiển hoặc hiện diện của người dùng

Động cơ bơm nước R385 Water Pump 12VDC

Loại: Default
Tình trạng: Còn hàng
Thương hiệu: Import
Mã sản phẩm: HS0439

Giá bán:
47.000₫

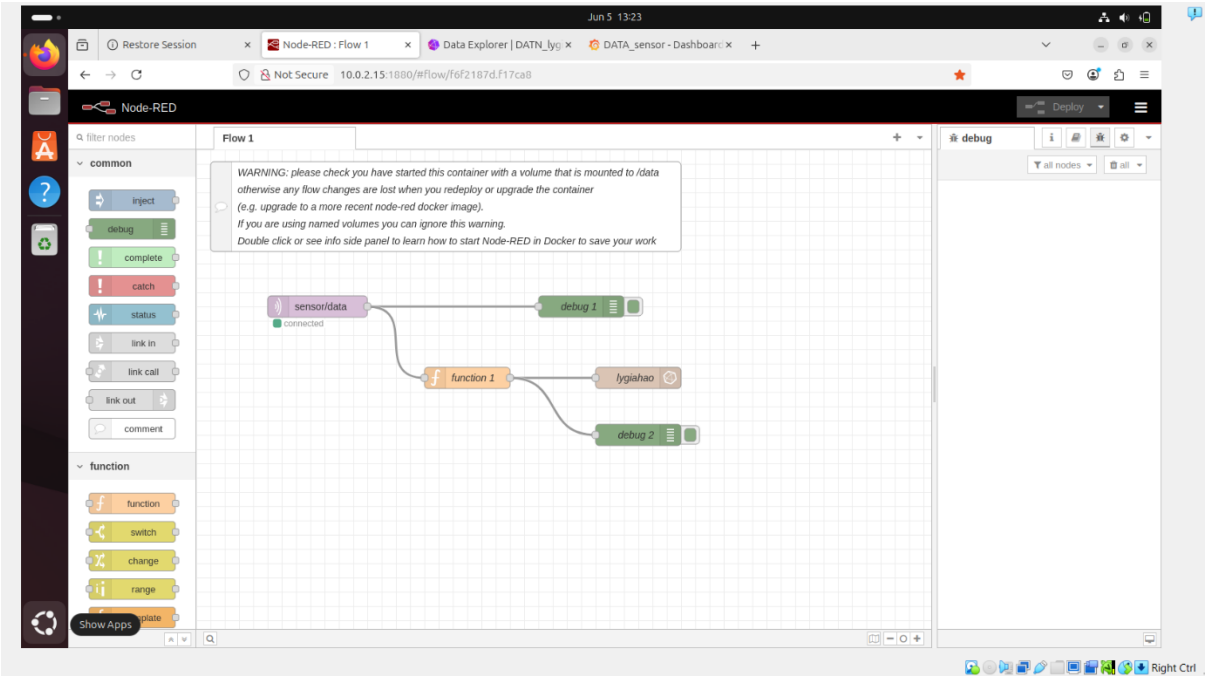
Số lượng: 1

THÊM VÀO GIỎ

MUA NGAY

Liên hệ Zalo OA Hshop
Để được tư vấn và hỗ trợ ngay!!!

2. Các phần mềm nền tảng sử dụng trong đồ án
   - **Ubuntu (Virtual Box)**
     Công nghệ ảo hóa nhẹ giúp đóng gói và triển khai các ứng dụng một cách độc lập.
     Dùng để triển khai các thành phần phần mềm như: MQTT broker, Home Assistant, Node-RED, InfluxDB, Grafana một cách tách biệt, dễ quản lý và nâng cấp.
     Đảm bảo hệ thống hoạt động ổn định, bảo mật và có thể sao lưu/phục hồi nhanh chóng nếu có sự cố.
     Hỗ trợ triển khai linh hoạt: trên máy tính cá nhân, server mini tại nhà hoặc server trên cloud
   - **Node-RED**
     Là một công cụ lập trình nhờ vào khả năng kéo và thả các thành phần (gọi là “nodes”) để xây dựng các luồng dữ liệu phức tạp một cách trực quan.

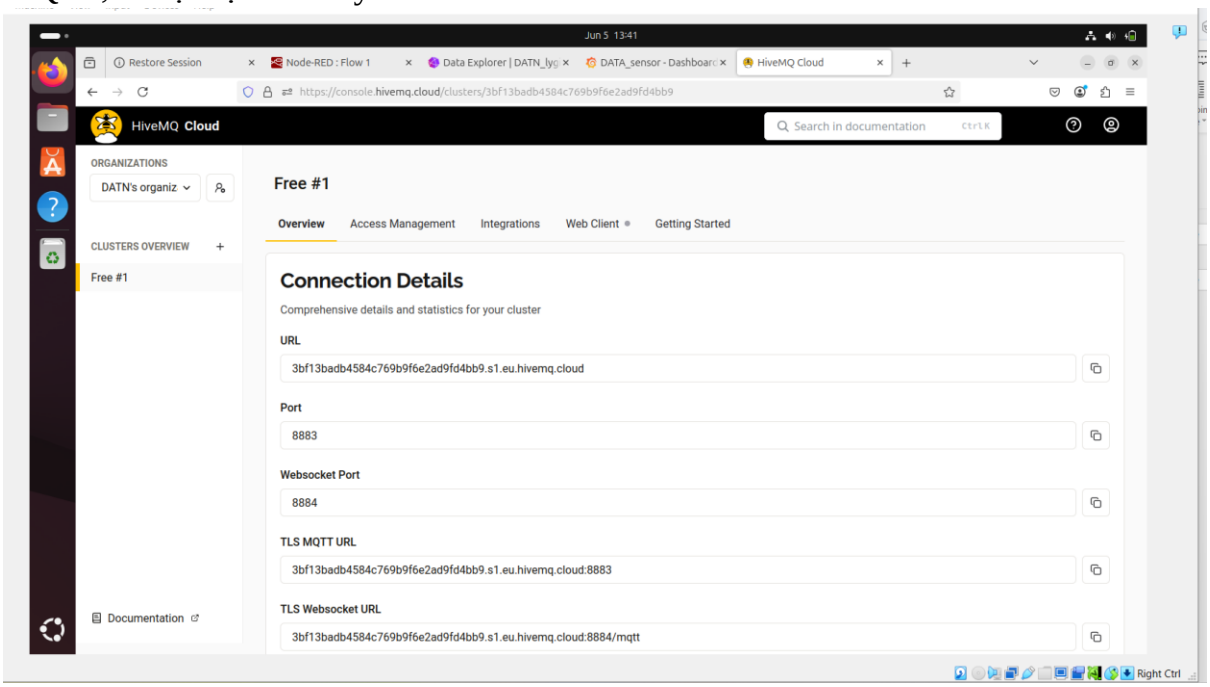


- **InfluxDB + Grafana**

  InfluxDB: Cơ sở dữ liệu thời gian thực, chuyên dùng lưu dữ liệu cảm biến.
  Grafana: Công cụ hiển thị dữ liệu từ InfluxDB dưới dạng biểu đồ.
  Lưu trữ dữ liệu thời gian thực về giá trị độ đục, TDS, nhiệt độ và độ PH.
  Phân tích và đánh giá xu hướng các giá trị trên nếu vượt ngưỡng thì tảo sẽ sống được trong bao lâu.
  Grafana giúp người dùng dễ dàng hình dung thông tin thông qua các biểu đồ động, trực quan.
  Hỗ trợ AI học từ dữ liệu lịch sử để đưa ra các đề xuất tối ưu.

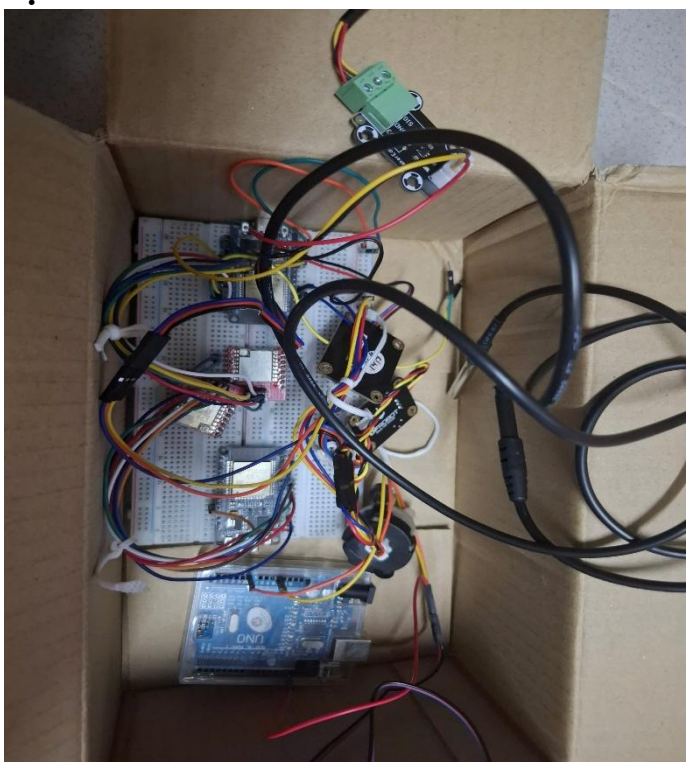
- **MQTT Broker (HiveMQ)**

Kết nối ESP32 với hệ thống máy chủ Ubuntu (Node-Red, InfluxDB và Grafana)
ESP32 sử dụng MQTT để gửi dữ liệu cảm biến như:
Độ đục, TDS, nhiệt độ, độ Ph
Trạng thái bơm
Đồng thời ESP32 cũng nhận lệnh điều khiển từ Node-RED thông qua MQTT, ví dụ: bật/ tắt máy bơm



3. Thi công mạch

- **Thử nghiệm trên testboard**



4. **Phụ lục code**
   - **Code Node-Red chuyển dữ liệu dạng String nhận từ Esp sang Json hoặc các dạng dữ liệu phù hợp để chuyển vào InfluxDB:**

```
// Nhận dữ liệu từ msg.payload
let inputData = msg.payload;

// Khởi tạo đối tượng JSON để lưu kết quả
let jsonData = {};

// Split dữ liệu theo dấu phẩy
let entries = inputData.split(',');

// Duyệt từng cặp key:value
entries.forEach(entry => {
    let [key, value] = entry.split(':');
    if (key && value !== undefined) {
        // Chuyển đổi giá trị sang dạng số nếu có thể
        let numValue = parseFloat(value);
        jsonData[key.trim()] = isNaN(numValue) ? value
    }
});

// Gán payload mới là dữ liệu JSON
msg.payload = jsonData;

// Trả về msg đã chỉnh sửa
return msg;
```

   - **Code Querry Script của InfluxDB để vẽ biểu đồ thông qua Grafana**

```
from(bucket: "DATN_01")
  |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
  |> filter(fn: (r) => r["_measurement"] == "Value")
  |> filter(fn: (r) => r["_field"] == "NTU" or r["_field"] == "T" or r["_field"] == "TDS" or r["_field"] == "pH")
  |> aggregateWindow(every: v.windowPeriod, fn: mean, createEmpty: false)
  |> yield(name: "mean")
```

   - **Code Esp thu dữ liệu từ các cảm biến độ đục, TDS, nhiệt độ và độ PH truyền dữ liệu qua Module LoRa**

nhanph-lora.ino

```
#include <LoRa.h>
#include <DallasTemperature.h>
#include <OneWire.h>

// Cảm biến nhiệt độ DS18B20
#define ONE_WIRE_BUS 4
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

// Các chân cảm biến khác
#define TdsSensorPin 32
#define analogPin 34  // cảm biến NTU
#define pHSerialBaud 115200

// LoRa pins
#define LORA_SCK  18
#define LORA_MISO 19
#define LORA_MOSI 23
#define LORA_SS   5
#define LORA_RST  14
#define LORA_DIO0 2

// Buffer đọc serial pH
String pH_from_Arduino = "";
bool pH_received = false;

// Biến chứa dữ liệu pH
float pH_value = 7.0; // Mặc định, sẽ cập nhật khi nhận được

// Các biến cảm biến khác
float temperature = 25.0;
unsigned long lastMeasurementTime = 0;
const unsigned long measurementInterval = 800; // ms

void setup() {
  Serial.begin(pHSerialBaud);
  // Khởi động cảm biến nhiệt độ
  sensors.begin();

  // Khởi tạo cảm biến TDS
  pinMode(TdsSensorPin, INPUT);
  pinMode(analogPin, INPUT);

  // Khởi động LoRa
```

```
45    LoRa.setPins(LORA_SS, LORA_RST, LORA_DIO0);
46    if (!LoRa.begin(433E6)) {
47      Serial.println("Không thể khởi động LoRa");
48      while (1);
49    }
50    Serial.println("LoRa đã sẵn sàng");
51  }
52
53  void loop() {
54    // --- 1. Đọc dữ liệu pH từ Arduino ---
55    while (Serial.available()) {
56      char incomingChar = Serial.read();
57      if (incomingChar == '\n') {
58        pH_received = true;
59      } else {
60        pH_from_Arduino += incomingChar;
61      }
62    }
63    if (pH_received) {
64      // Parse dữ liệu "pH:x.x"
65      int index = pH_from_Arduino.indexOf("pH:");
66      if (index != -1) {
67        String pH_str = pH_from_Arduino.substring(index + 3)
68        pH_value = pH_str.toFloat();
69      }
70      pH_from_Arduino = "";
71      pH_received = false;
72    }
73
74    // --- 2. Đo các cảm biến khác ---
75    // Nhiệt độ DS18B20
76    sensors.requestTemperatures();
77    float tempRead = sensors.getTempCByIndex(0);
78    if (tempRead != DEVICE_DISCONNECTED_C) {
79      temperature = tempRead;
80    }
81
82    // TDS
83    float tds = 0.0;
84    {
85      int sum = 0;
86      const int SCOUNT = 30;
87      int buffer[SCOUNT];
88      for (int i = 0; i < SCOUNT; i++) {
```

```
      buffer[i] = analogRead(TdsSensorPin);
      delay(40);
    }
    // Trung vị
    int medianVal = getMedian(buffer, SCOUNT);
    float voltage = medianVal * (5.0 / 4095.0); // ESP32 12-
    tds = calculateTDS(voltage, temperature);
  }

  // NTU
  float NTU = measureNTU();

  // --- 3. Gửi dữ liệu qua LoRa sau khoảng interval ---
  if (millis() - lastMeasurementTime >= measurementInterval)
    lastMeasurementTime = millis();

    // Chuẩn bị payload
    String payload = "";
    payload += "T:" + String(temperature, 2);
    payload += ",TDS:" + String((int)tds);
    payload += ",NTU:" + String(NTU >= 0 ? NTU : -1);
    payload += ",pH:" + String(pH_value, 2);

    // Gửi qua LoRa
    LoRa.beginPacket();
    LoRa.print(payload);
    LoRa.endPacket();

    // Xuất ra Serial để kiểm tra
    Serial.println("Gửi: " + payload);
  }
}

// Hàm trung vị
int getMedian(int arr[], int n) {
  for (int i = 0; i < n - 1; i++) {
    for (int j = i + 1; j < n; j++) {
      if (arr[i] > arr[j]) {
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
      }
    }
  }
  if (n % 2 == 0) {
    return (arr[n/2] + arr[n/2 - 1]) / 2;
  } else {
    return arr[n/2];
  }
}

// TDS calculation (ví dụ, bạn có thể chỉnh lại theo cảm biến của bạn)
```

```
141  float calculateTDS(float voltage, float temp) {
142    float compensationCoefficient = 1.0 + 0.02 * (temp - 25.0);
143    float compensatedVoltage = voltage / compensationCoefficient;
144    float tds = (133.42 * pow(compensatedVoltage, 3)
145                 - 255.86 * compensatedVoltage * compensatedVoltag
146                 + 857.39 * compensatedVoltage) * 0.5;
147    return tds;
148  }
149
150  // Hàm đo NTU (ví dụ)
151  float measureNTU() {
152    int sensorValue = analogRead(analogPin);
153    float Um = (sensorValue / 4095.0) * 5.0; // ESP32 12-bit ADC
154    if (Um > 5.0) {
155      Serial.println("Cảnh báo: Um vượt ngưỡng Vref!");
156      return -1;
157    }
158    float f = Um / 5.0;
159    if (f >= 0.98 && f <= 1.0) {
160      return 0.0;
161    } else {
162      float NTU = (1.0 - f) * 1000.0; // ví dụ, scale phù hợp
```

```
162      float NTU = (1.0 - f) * 1000.0; // ví dụ, scale phù hợp
163      return constrain(NTU, 0, 1000);
164    }
165  }
```

- **Code nhận dữ liệu từ module LoRa và gửi tới hệ thống máy chủ thông qua Mqtt**

```
1   #include <SPI.h>
2   #include <LoRa.h>
3   #include <WiFi.h>
4   #include <WiFiClientSecure.h>
5   #include <PubSubClient.h>
6
7   // Định nghĩa chân GPIO cho module LoRa (theo cấu hình của bạn)
8   #define LORA_SCK  18
9   #define LORA_MISO 19
10  #define LORA_MOSI 23
11  #define LORA_SS   5
12  #define LORA_RST  14
13  #define LORA_DIO0 2
14
15  // Thông tin WiFi
16  const char* ssid = "Ghao";
17  const char* password = "wacz6966";
18
19  // Thông tin MQTT (HiveMQ Cloud)
20  const char* mqtt_server = "3bf13badb4584c769b9f6e2ad9fd4bb9.s1.eu.hivemq.cloud";
21  const int mqtt_port = 8883;
22  const char* mqtt_topic = "sensor/data";
23
24  // Tài khoản MQTT
25  const char* mqtt_username = "DATN_LYGIAHAO";
26  const char* mqtt_password = "Lygiahao228";
27  const char* mqtt_clientID = "ESP32LoRaReceiver";
28
29  // MQTT Client bảo mật
30  WiFiClientSecure espClient;
```

```
PubSubClient client(espClient);

// Thời gian gửi test
unsigned long lastTest = 0;
const unsigned long testInterval = 10000;

void setup() {
  Serial.begin(115200);

  // Kết nối WiFi
  Serial.print("Đang kết nối WiFi");
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nWiFi đã kết nối!");
  Serial.print("Địa chỉ IP: ");
  Serial.println(WiFi.localIP());

  // Bỏ kiểm tra chứng chỉ (chỉ dùng để test)
  espClient.setInsecure();

  // Cấu hình MQTT server
  client.setServer(mqtt_server, mqtt_port);

  // Khởi động LoRa với chân đúng
  Serial.println("Khởi động LoRa...");
  LoRa.setPins(LORA_SS, LORA_RST, LORA_DIO0);
  if (!LoRa.begin(433E6)) { // Thay đổi tần số phù hợp
    Serial.println("Khởi động LoRa thất bại!");
    while (true);
  }
  Serial.println("LoRa đã khởi động");
}

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

  // Kiểm tra gói LoRa nhận được
  int packetSize = LoRa.parsePacket();
  if (packetSize > 0) {
    Serial.print("Nhận gói LoRa, kích thước: ");
    Serial.println(packetSize);

    String receivedData = "";
    while (LoRa.available()) {
      receivedData += (char)LoRa.read();
    }

    Serial.print("Nội dung: ");
    Serial.println(receivedData);

    // Gửi dữ liệu lên MQTT
    if (client.publish(mqtt_topic, receivedData.c_str())
      Serial.println("Gửi dữ liệu lên MQTT thành công");
    } else {
```

```
    } else {
      Serial.println("Gửi dữ liệu MQTT thất bại");
    }
  }

}

void reconnectMQTT() {
  while (!client.connected()) {
    Serial.print("Đang kết nối MQTT...");
    if (client.connect(mqtt_clientID, mqtt_username, mqtt_password)) {
      Serial.println("Kết nối MQTT thành công");
    } else {
      Serial.print("Thất bại, mã lỗi: ");
      Serial.println(client.state());
      Serial.println("Thử lại sau 5 giây");
      delay(5000);
    }
  }
}
```