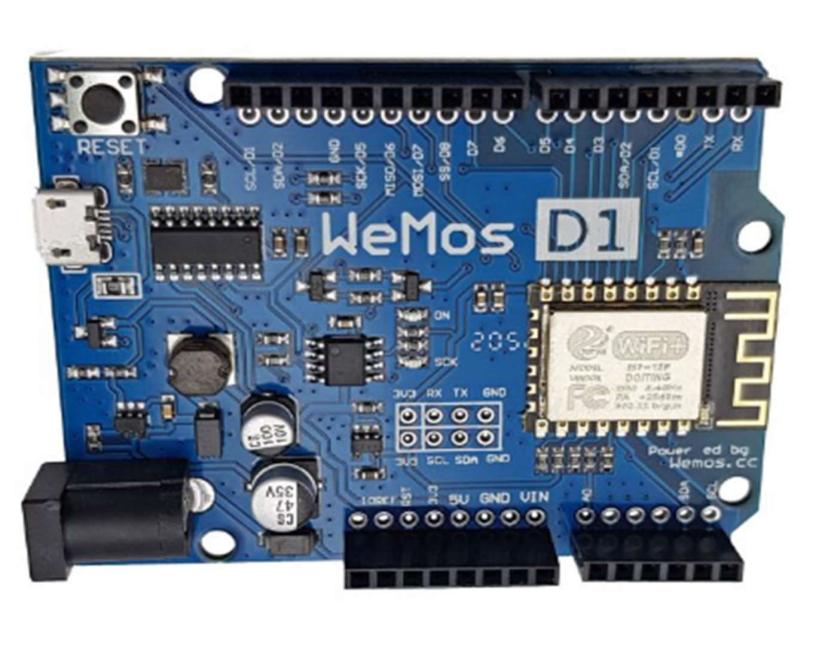
|  |
| --- |
| **ĐẠI HỌC HOA SEN**  **KHOA CÔNG NGHỆ THÔNG TIN**  Đại học Hoa sen — Sở Khoa học và Công nghệ Thành phố Hồ Chí Minh**---- ----**  **BÁO CÁO**  **PROJECT**  **Tên sinh viên: Lê Minh Quang**  **Mssv: 22122997**  **Tên đề tài: Cảm biến độ ẩm đất**  **Môn học: Thực hành công nghệ thông tin 3**  **Lớp: 0200**  **Giảng viên hướng dẫn: Trần Thị Trương Thi**  **THÀNH PHỐ HỒ CHÍ MINH – 2024** |

# NỘI DUNG

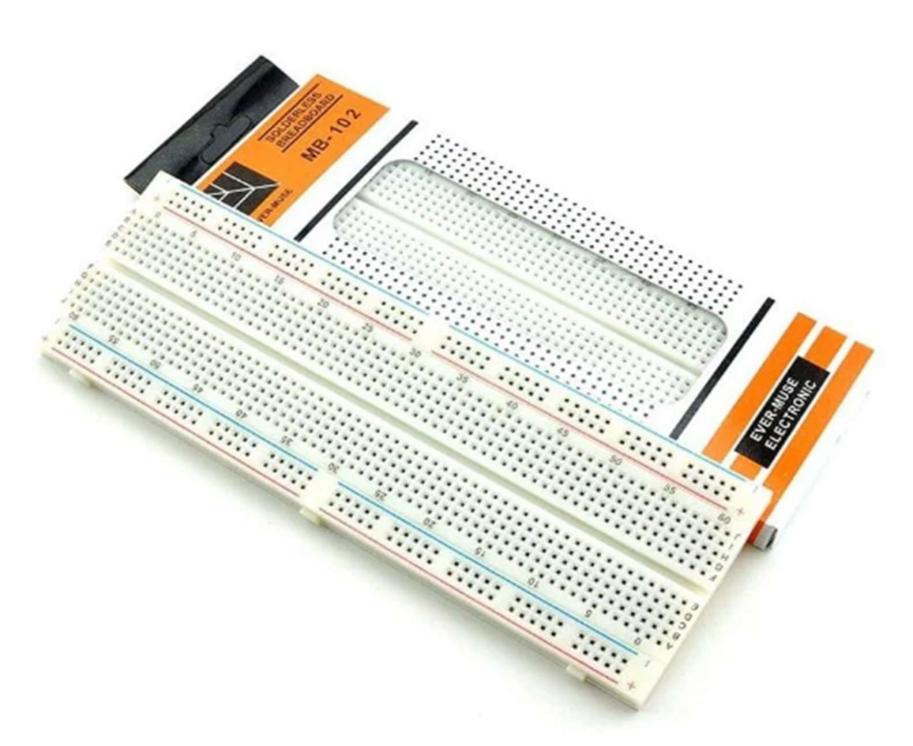
## Thiết bị

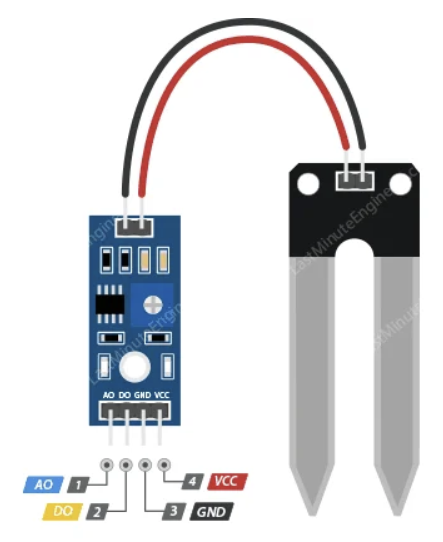
Những thiết bị sử dụng:

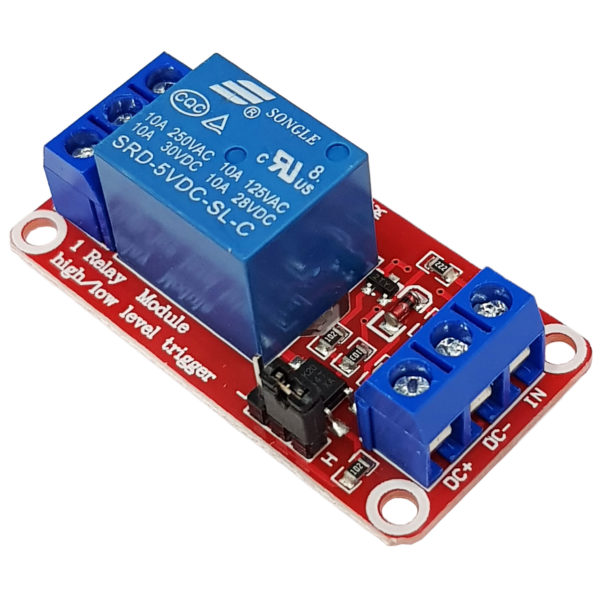
* ­Wemos D1 R2, cảm biến độ ẩm đất, máy bơm nước mini, Relay 5v, dây nối, breadboard MB-102 830 lỗ.









\



## Code Arduino

#include <ESP8266WiFi.h>

#include <PubSubClient.h>

#include <ArduinoJson.h>

#include <NTPClient.h>

#include <WiFiUdp.h>

#include <ESP8266HTTPClient.h>  // Include the HTTPClient library

// WiFi credentials

// #define ssid "HSU\_Students"

// #define password "dhhs12cnvch"

#define ssid "Tenda\_CBCCD0"

#define password "t12345678"

// MQTT broker details

#define mqtt\_server "broker.emqx.io"

const uint16\_t mqtt\_port = 1883;

#define mqtt\_topic\_pub\_led "ttcntt3/mqleeee/ledstatus"

#define mqtt\_topic\_sub\_led "ttcntt3/mqleeee/ledstatus"

#define mqtt\_topic\_pub\_soil "ttcntt3/mqleeee/soil"

#define mqtt\_topic\_sub\_soil "ttcntt3/mqleeee/soil"

#define mqtt\_topic\_pub\_time "ttcntt3/mqleeee/time"

#define mqtt\_topic\_sub\_time "ttcntt3/mqleeee/time"

// Sensor and LED pin definitions

#define soil A0

#define LED\_PIN D4

WiFiClient espClient;

PubSubClient client(espClient);

StaticJsonDocument<256> doc;

WiFiUDP ntpUDP;

NTPClient timeClient(ntpUDP, "pool.ntp.org", 7 \* 3600, 60000);  // UTC+7, update every 60 seconds

const char\* URL = "http://192.168.1.87:3003/data";  // Your server URL

char ledstatus[32] = "off";

void setup() {

  pinMode(soil, INPUT);

  pinMode(LED\_PIN, OUTPUT);

  digitalWrite(LED\_PIN, LOW);  // Set initial state to OFF

  Serial.begin(115200);

  setup\_wifi();

  client.setServer(mqtt\_server, mqtt\_port);

  client.setCallback(callback);

  timeClient.begin();

  reconnect();

}

void setup\_wifi() {

  delay(10);

  Serial.println();

  Serial.print("Connecting to ");

  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL\_CONNECTED) {

    delay(500);

    Serial.print(".");

  }

  Serial.println("");

  Serial.println("WiFi connected");

  Serial.println("IP address: ");

  Serial.println(WiFi.localIP());

}

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

  deserializeJson(doc, payload, length);

  if (String(topic) == mqtt\_topic\_sub\_led) {

    strlcpy(ledstatus, doc["status"] | "off", sizeof(ledstatus));

    String mystring(ledstatus);

    Serial.print("Message arrived [");

    Serial.print(topic);

    Serial.print("] ");

    Serial.println(ledstatus);

    if (mystring == "on") {

      Serial.println("Turning LED on");

      digitalWrite(LED\_PIN, HIGH);

    } else {

      Serial.println("Turning LED off");

      digitalWrite(LED\_PIN, LOW);

    }

  }

}

void reconnect() {

  while (!client.connected()) {

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

    if (client.connect("ttcntt3\_mqleeee")) {

      Serial.println("connected");

      // Xuất bản trạng thái LED ban đầu

      doc.clear();

      doc["name"] = "led";

      doc["status"] = ledstatus;

      char buffer[256];

      size\_t n = serializeJson(doc, buffer);

      client.publish(mqtt\_topic\_pub\_led, buffer, n);

      // Xuất bản thời gian ban đầu

      doc.clear();

      doc["timestamp"] = timeClient.getFormattedTime();

      n = serializeJson(doc, buffer);

      client.publish(mqtt\_topic\_pub\_time, buffer, n);

      // Xuất bản độ ẩm đất ban đầu

      doc.clear();

      doc["soilMoisture"] = analogRead(soil);

      n = serializeJson(doc, buffer);

      client.publish(mqtt\_topic\_pub\_soil, buffer, n);

      client.subscribe(mqtt\_topic\_sub\_led);

      client.subscribe(mqtt\_topic\_sub\_soil);

      client.subscribe(mqtt\_topic\_sub\_time);

    } else {

      Serial.print("failed, rc=");

      Serial.print(client.state());

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

      delay(5000);

    }

  }

}

void postData(int soilMoistureValue, const String& timestamp) {

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

    HTTPClient http;

    http.begin(espClient,URL);

    http.addHeader("Content-Type", "application/json");

    StaticJsonDocument<256> jsonDoc;

    jsonDoc["soilMoisture"] = soilMoistureValue;

    jsonDoc["timestamp"] = timestamp;

    String requestBody;

    serializeJson(jsonDoc, requestBody);

    int httpResponseCode = http.POST(requestBody);

    if (httpResponseCode > 0) {

      String response = http.getString();

      Serial.println(httpResponseCode);

      Serial.println(response);

    } else {

      Serial.print("Error on sending POST: ");

      Serial.println(httpResponseCode);

    }

    http.end();

  } else {

    Serial.println("WiFi Disconnected");

  }

}

void loop() {

  if (!client.connected()) {

    reconnect();

  }

  client.loop();

  static unsigned long lastPublish = 0;

  timeClient.update();

  unsigned long now = millis();

  if (now - lastPublish >= 15000) {  // Corrected to use >= instead of ==

    // Publish soil moisture periodically

    int soilMoistureValue = analogRead(soil);

    String formattedTime = timeClient.getFormattedTime();

    doc.clear();

    doc["soilMoisture"] = soilMoistureValue;

    char buffer[256];

    size\_t n = serializeJson(doc, buffer);

    client.publish(mqtt\_topic\_pub\_soil, buffer, n);

    Serial.print("Soil Moisture: ");

    Serial.print(soilMoistureValue);

    Serial.print(" at ");

    Serial.println(formattedTime);

    doc.clear();

    doc["timestamp"] = formattedTime;

    n = serializeJson(doc, buffer);

    client.publish(mqtt\_topic\_pub\_time, buffer, n);

    lastPublish = now;

    // Post data to the server

    postData(soilMoistureValue, formattedTime);

  }

}

## MQTTX

## 

## Front-end



## Back-end

const express = require('express');

const mysql = require('mysql2');

const cors = require('cors');

const app = express();

const port = 3002;

const bodyParser = require('body-parser');

// Middleware

app.use(cors());

app.use(express.json()); // Body parser middleware

// Thiết lập kết nối cơ sở dữ liệu MySQL

const db = mysql.createConnection({

  host: 'db',

  user: 'root',

  password: 'Quangdeptrai123', // Thay đổi mật khẩu tương ứng

  database: 'mydatabase'

});

app.use(bodyParser.json());  // Parse JSON bodies

// Kết nối đến MySQL

db.connect((err) => {

  if (err) {

    throw err;

  }

  console.log('MySQL connected...');

});

// Endpoint: Lấy tất cả các ngày trong tuần

app.get('/time\_schedule', (req, res) => {

  const sql = 'SELECT \* FROM time\_schedule ORDER BY id DESC LIMIT 1';

  db.query(sql, (err, results) => {

    if (err) {

      console.error(err);

      res.status(500).json({ error: 'Error fetching data' });

      return;

    }

    res.json(results);

  });

});

app.post('/time\_schedules', (req, res) => {

  const { timestart } = req.body;

  const sql = 'INSERT INTO time\_schedule (timestart) VALUES (?)';

  db.query(sql, [timestart], (err, result) => {

    if (err) throw err;

    res.send('Time schedule saved');

  });

});

app.post('/data', (req, res) => {

  const { soilMoisture, timestamp } = req.body;

  const sql = 'INSERT INTO data (soilMoisture, timestamp) VALUES (?, ?)';

  db.query(sql, [soilMoisture, timestamp], (err, result) => {

    if (err) {

      console.error('Error inserting data:', err);

      res.status(500).send('Error saving data');

      return;

    }

    res.send('Data saved');

  });

});

app.listen(port, () => {

  console.log(`Server running on port ${port}`);

});

## Docker

6.1 dockerfile

FROM node:lts-alpine

ENV NODE\_ENV=production

WORKDIR /usr/src/app

COPY ["package.json", "package-lock.json\*", "npm-shrinkwrap.json\*", "./"]

RUN npm install --production --silent && mv node\_modules ../

COPY . .

EXPOSE 5555

RUN chown -R node /usr/src/app

USER node

CMD [ "node", "api.js" ]

6.2 docker-compose.yml

version: '3.4'

services:

  db:

    image: mysql

    container\_name: mysql\_db

    environment:

      MYSQL\_ROOT\_PASSWORD: Quangdeptrai123

      MYSQL\_DATABASE: mydatabase

    ports:

      - "3308:3306"

    volumes:

      - mysql\_db-data:/var/lib/mysql

    networks:

      - network-mysql-api

  backend:

    depends\_on:

      - db

    networks:

      - network-mysql-api

    image: backend

    container\_name: cntt3-api

    build:

      dockerfile: ./Dockerfile

    environment:

      NODE\_ENV: production

      DB\_HOST: db

      DB\_USER: root

      DB\_PASSWORD: Quangdeptrai123

      DB\_NAME: mydatabase

      DB\_PORT: 3306

    ports:

      - "3003:3002"

volumes:

  mysql\_db-data:

networks:

  network-mysql-api:

## API

