

**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC HOA SEN**

**KHOA CÔNG NGHỆ THÔNG TIN**

----------------------------

**BÁO CÁO MÔN HỌC**

**ĐỀ TÀI:**

**ỨNG DỤNG QUẢN LÝ ĐIỀU KHIỂN VÀ TỐI ƯU ĐIỀU KIỆN MÔI TRƯỜNG (nhiệt độ môi trường , độ ẩm ) GIÚP TỰ ĐỘNG HÓA CHĂM SÓC CÂY TRỒNG**

**Môn học : THỰC HÀNH CÔNG NGHỆ THÔNG TIN 3**

**Lớp MH : IT205DV01\_0300**

**Sinh viên thực hiện : HOÀNG CÔNG KHA 22012134**

**Giảng viên hướng đẫn : ThS. Trần Thị Trương Thi**

**Thành Phố Hồ Chí Minh, 6/2024**



**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC HOA SEN**

**KHOA CÔNG NGHỆ THÔNG TIN**

----------------------------

**BÁO CÁO MÔN HỌC**

**ĐỀ TÀI:**

**ỨNG DỤNG QUẢN LÝ ĐIỀU KHIỂN VÀ TỐI ƯU ĐIỀU KIỆN MÔI TRƯỜNG (nhiệt độ môi trường , độ ẩm ) GIÚP TỰ ĐỘNG HÓA CHĂM SÓC CÂY TRỒNG**

**Môn học : THỰC HÀNH CÔNG NGHỆ THÔNG TIN 3**

**Lớp MH : IT205DV01\_0300**

**Sinh viên thực hiện : HOÀNG CÔNG KHA 22012134**

**Giảng viên hướng đẫn : ThS. Trần Thị Trương Thi**

**Thành Phố Hồ Chí Minh, 6/2024**

# LỜI CAM KẾT

Tôi cam kết rằng bài báo cáo này là kết quả của sự nghiên cứu và làm việc nghiêm túc của cá nhân tôi về đề tài Arduino. Toàn bộ nội dung trong bài viết đều là những hiểu biết, phân tích và kết quả thử nghiệm mà tôi đã thực hiện một cách trung thực và chính xác. Những thông tin và tài liệu tham khảo được sử dụng trong bài viết này đều đã được trích dẫn và ghi nguồn rõ ràng theo quy định.

Tôi xin chịu hoàn toàn trách nhiệm về tính chính xác và trung thực của bài báo cáo này. Nếu có bất kỳ sai sót hoặc vi phạm nào liên quan đến nội dung bài viết, tôi xin hoàn toàn chịu trách nhiệm và sẵn sàng chấp nhận các hình thức xử lý theo quy định của nhà trường và pháp luật. Tôi cam kết rằng không có bất kỳ hành vi sao chép hay gian lận nào trong quá trình thực hiện bài báo cáo này.

# TRÍCH YẾU

Bài báo cáo này cung cấp một cái nhìn tổng quan về sức mạnh và tiềm năng của nền tảng Arduino trong việc phát triển các ứng dụng điện tử và lập trình. Từ việc giới thiệu về nguồn gốc và phát triển của Arduino đến việc thực hiện các dự án thực tế, bài viết tập trung vào việc hướng dẫn và cung cấp kiến thức chi tiết cho người đọc.

Bài báo cáo bắt đầu bằng việc khám phá lịch sử và sự phát triển của Arduino, nhấn mạnh vào những tính năng và ưu điểm của nền tảng này. Sau đó, người đọc sẽ được dẫn dắt qua quá trình thiết lập môi trường lập trình Arduino và các bước cần thiết để bắt đầu làm việc với bo mạch.

Phần quan trọng tiếp theo của báo cáo tập trung vào các dự án thực hành, từ những ứng dụng đơn giản như đèn LED điều khiển đến những dự án phức tạp hơn như robot tự hành. Mỗi dự án sẽ được minh họa bằng các hướng dẫn chi tiết, mã nguồn và các kết quả thực nghiệm.

Cuối cùng, bài báo cáo đề cập đến những cơ hội và thách thức trong việc áp dụng Arduino trong giáo dục và công nghiệp. Nhận định và đánh giá sâu sắc được đề xuất, cùng với những gợi ý và khuyến nghị để tối ưu hóa hiệu quả sử dụng nền tảng Arduino.

# LỜI CẢM ƠN

Chúng tôi xin chân thành gửi lời cảm ơn sâu sắc đến tất cả những người đã đóng góp vào quá trình thực hiện bài báo cáo này. Sự hỗ trợ, động viên và kiến thức chuyên môn từ các giáo viên, bạn bè và các nhà nghiên cứu đã đóng vai trò quan trọng trong việc hoàn thành dự án này.

Đặc biệt, chúng tôi muốn bày tỏ lòng biết ơn đặc biệt đến [danh sách tên], người đã dành thời gian và nỗ lực để hướng dẫn và cung cấp phản hồi xây dựng quý báu trong suốt quá trình nghiên cứu và viết bài.

Những đóng góp của các bạn đã giúp chúng tôi hiểu sâu hơn về chủ đề và mang lại sự tự tin trong việc chia sẻ kiến thức của chúng tôi với cộng đồng. Một lần nữa, chân thành cảm ơn tất cả sự hỗ trợ và khích lệ mà chúng tôi đã nhận được.

# NHẬN XÉT CỦA GIẢNG VIÊN

Thành phố Hồ Chí Minh, Ngày 19 tháng 6 năm 2024

Người nhận xét

**MỤC LUC**

[LỜI CAM KẾT 3](#_Toc13434)

[TRÍCH YẾU 4](#_Toc4746)

[LỜI CẢM ƠN 5](#_Toc31518)

[NHẬN XÉT CỦA GIẢNG VIÊN 6](#_Toc5320)

[DANH MỤC HÌNH ẢNH 8](#_Toc31368)

[DANH MỤC BẢNG BIỂU 9](#_Toc27202)

[NHẬP ĐỀ 10](#_Toc19603)

[PHẦN CÔNG 11](#_Toc5070)

[NỘI DUNG 12](#_Toc11142)

[I. Giới thiệu thiết bị 12](#_Toc13397)

[1. Board wemos D1 12](#_Toc8562)

[2. Cảm biến DHT11 13](#_Toc7467)

[3. Buzzer 13](#_Toc5057)

[4.LCD I2C 14](#_Toc6637)

[II. Mô tả project 14](#_Toc5130)

[III. Mô hình arduino 15](#_Toc24707)

[IV. MQTTX 15](#_Toc26780)

[V. Nối mạch 16](#_Toc28151)

[VI. Code arduino 16](#_Toc4712)

[1. Code arduino 16](#_Toc7998)

[2. Code frontend react native 20](#_Toc815)

[3. Code backend expressjs 24](#_Toc3376)

[4. Dockerfile cho backend 26](#_Toc3613)

[5. docker-compose.yml 27](#_Toc11576)

[6. Database 27](#_Toc11828)

[7. Dockerfile cho database 28](#_Toc1155)

# DANH MỤC HÌNH ẢNH

[Hình 1 :Ảnh Wemos D1 R1 12](#_Toc20883)

[Hình 2 :DHT11 13](#_Toc9214)

[Hình 3 :Buzzer 13](#_Toc12893)

[Hình 4 :LCD I2C 14](#_Toc15734)

[Hình 5 : Mô hình arduino 15](#_Toc31170)

# DANH MỤC BẢNG BIỂU

[Bảng 1 : Wemos D1 R1 nối với DHT11 16](#_Toc27066)

[Bảng 2 : Wemos D1 R1 nối với LCD I2C 16](#_Toc29074)

[Bảng 3 : Wemos D1 R1 nối với Loa 16](#_Toc17578)

# NHẬP ĐỀ

Dự án này tập trung vào việc sử dụng nền tảng Arduino để phát triển một hệ thống thông minh quản lý điều khiển và tối ưu hóa môi trường cho việc chăm sóc cây trồng. Bằng cách tích hợp các cảm biến nhiệt độ và độ ẩm, Arduino giúp tự động điều chỉnh các thông số môi trường như nước, ánh sáng và nhiệt độ, tạo điều kiện lý tưởng cho sự phát triển của cây trồng một cách hiệu quả và tiết kiệm năng lượng.

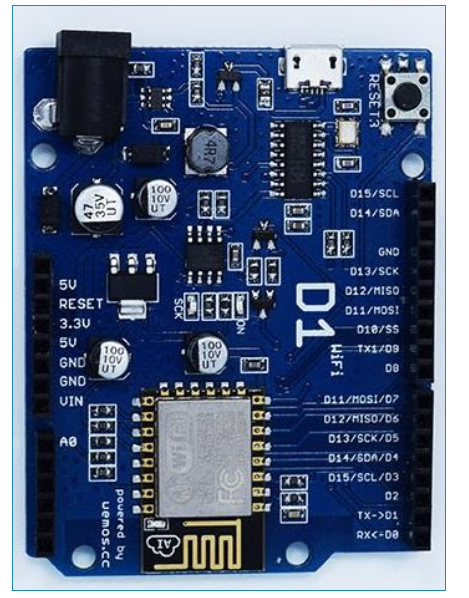
# PHẦN CÔNG

|  |  |  |
| --- | --- | --- |
| STT | HỌ TÊN | CÔNG VIỆC |
| 1 | Hoàng Công Kha | Viết code arduino, làm báo cáo, xây dựng giao diện, làm backend, cấu hình cho docker |
| 2 |  |  |

# NỘI DUNG

1. Giới thiệu thiết bị
2. Board wemos D1

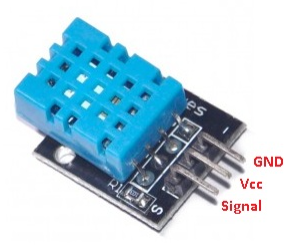
* Vi điều khiển: ESP8266EX
* Điện áp hoạt động: 3.3V
* Số chân I/O: 11 (tất cả các chân I/O đều có Interrupt/PWM/I2C/One wire, trừ chân D0)
* Số chân Analog Input: 1 (điện áp vào tối đa 3.3V)
* Bộ nhớ Flash: 4MB
* Điện áp vào: 9-24V
* Điện áp ra: 5V – Dòng max: 1A
* Giao tiếp: [Cable Micro USB](https://nshopvn.com/product/day-cap-cable-node-mcu-day-sac-micro-usb/)
* Wifi: 2.4 GHz
* Hỗ trợ bảo mật: WPA/WPA2
* Tích hợp giao thức TCP/IP
* Kích thước: 68.6mm x 53.4mm (2.701″ x 2.102″)
* Lập trình trên các ngôn ngữ: C/C++, Micropython, NodeMCU – Lua



**Hình 1****:**Ảnh Wemos D1 R1

1. Cảm biến DHT11

* Điện áp hoạt động: 3.5V đến 5.5V
* Dòng hoạt động: 0,3mA (đo) 60uA (chế độ chờ)
* Đầu ra: Dữ liệu nối tiếp
* Phạm vi nhiệt độ: 0 ° C đến 50 ° C
* Phạm vi độ ẩm: 20% đến 90%
* Độ phân giải: Nhiệt độ và Độ ẩm đều là 16-bit
* Độ chính xác: ± 1 ° C và ± 1%



**Hình 2****:**DHT11

1. Buzzer

Thông số kỹ thuật Buzzer:

* Chân dương ( + )
* Chân âm ( - )

**Hình 3****:**Buzzer

1. LCD I2C

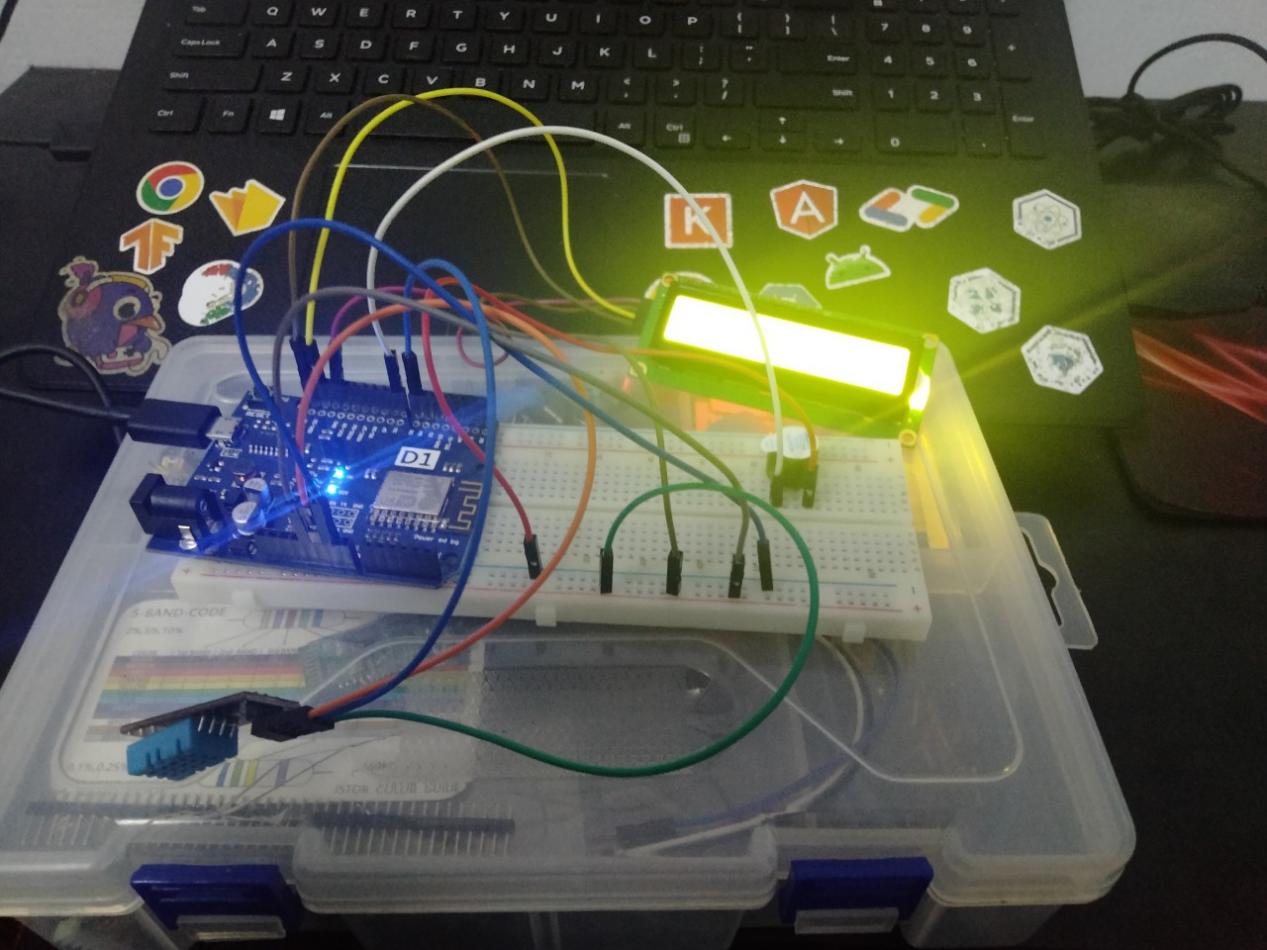
* Điện áp hoạt động: 2.5-6V DC.
* Hỗ trợ màn hình: LCD1602,1604,2004 (driver HD44780).
* Giao tiếp: I2C.
* Địa chỉ mặc định: 0X27 (có thể điều chỉnh bằng ngắn mạch chân A0/A1/A2).
* Tích hợp Jump chốt để cung cấp đèn cho LCD hoặc ngắt.
* Tích hợp biến trở xoay điều chỉnh độ tương phản cho LCD.

**Hình 4**:LCD I2C

1. Mô tả project

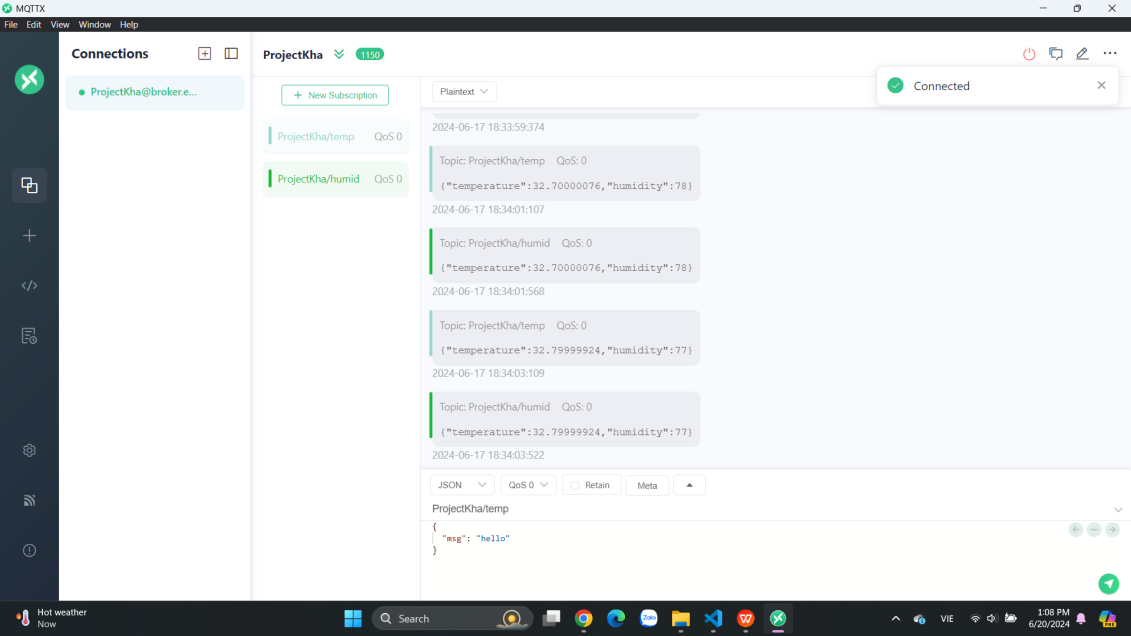
Dự án này sử dụng Arduino để theo dõi nhiệt độ và độ ẩm bằng cảm biến DHT11. Khi nhiệt độ vượt ngưỡng, còi báo động sẽ kêu để cảnh báo người dùng. Hệ thống còn tích hợp màn hình hiển thị giá trị nhiệt độ và độ ẩm hiện tại, đồng thời cung cấp giao diện điều khiển để bật/tắt còi thông qua nút bấm hoặc ứng dụng di động. Mục tiêu của dự án là duy trì môi trường lý tưởng cho cây trồng, cung cấp cảnh báo kịp thời và giúp quản lý, giám sát điều kiện môi trường một cách hiệu quả và tiện lợi.

1. Mô hình arduino



**Hình 5****:** Mô hình arduino

1. MQTTX



**Hình 4**:MQTTX

1. Nối mạch

|  |  |  |
| --- | --- | --- |
|  | Wemos D1 R1 | DHT11 |
| 1 | D5 | Signal |
| 2 | 5V | VCC |
| 3 | GND | GND |

Bảng 1: Wemos D1 R1 nối với DHT11

|  |  |  |
| --- | --- | --- |
|  | Wemos D1 R1 | LCD I2C |
| 1 | 5V | VCC |
| 2 | GND | GND |
| 3 | 014/SDA | SDA |
| 4 | 015/SCL | SCL |

Bảng 2: Wemos D1 R1 nối với LCD I2C

|  |  |  |
| --- | --- | --- |
|  | Wemos D1 R1 | Loa |
| 1 | D6 | VCC |
| 2 | GND | GND |

Bảng 3: Wemos D1 R1 nối với Loa

1. Code arduino
2. Code arduino

#include <ESP8266WiFi.h>

#include <PubSubClient.h>

#include <ArduinoJson.h>

#include <DHT.h>

#include <ESP8266WebServer.h>

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

#include <SoftwareSerial.h>

#define SSID "Nha tro sinh vien"

#define PASSWORD "39393939"

// MQTT Broker details

#define MQTT\_SERVER "broker.emqx.io"

const uint16\_t MQTT\_PORT = 1883;

#define MQTT\_TOPIC\_PUB\_TEMP "ProjectKha/temp"

#define MQTT\_TOPIC\_SUB\_TEMP "ProjectKha/temp"

#define MQTT\_TOPIC\_PUB\_HUMID "ProjectKha/humid"

#define MQTT\_TOPIC\_SUB\_HUMID "ProjectKha/humid"

WiFiClient espClient;

PubSubClient client(espClient);

StaticJsonDocument<256> doc;

ESP8266WebServer server(80);

// DHT sensor settings

#define DHTPIN D5  // Pin where the DHT sensor is connected

#define DHTTYPE DHT11  // Change this to DHT11 if you are using the DHT11 sensor

DHT dht(DHTPIN, DHTTYPE);

// Buzzer settings

#define BUZZER\_PIN D6  // Pin connected to the buzzer

int temperatureThreshold = 30;  // Default temperature threshold

bool buzzerState = LOW;

bool userBuzzerControl = true;  // User control flag for the buzzer

// LCD settings

LiquidCrystal\_I2C lcd(0x27, 16, 2);  // I2C address 0x27, 16 columns and 2 rows

void setup() {

    Serial.begin(115200);

    Serial.println();

    Serial.print("Connecting to existing Wifi network...");

    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());

    // Initialize LCD

    lcd.init();  // Initialize the LCD

    lcd.backlight();

    lcd.setCursor(0, 0);

    lcd.print("Initializing...");

    // Register URL handlers

    server.on("/Buzzer/on", HTTP\_GET, handleBuzzerOn);

    server.on("/Buzzer/off", HTTP\_GET, handleBuzzerOff);

    setup\_wifi();

    client.setServer(MQTT\_SERVER, MQTT\_PORT);

    client.setCallback(callback);

    dht.begin();

    pinMode(BUZZER\_PIN, OUTPUT);

    digitalWrite(BUZZER\_PIN, LOW);  // Turn off the buzzer initially

    server.begin();  // Start the web server

    Serial.println("HTTP server started");

    lcd.setCursor(0, 1);

    lcd.print("HTTP server ready");

    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 reconnect() {

    while (!client.connected()) {

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

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

            Serial.println("connected");

            client.subscribe(MQTT\_TOPIC\_SUB\_TEMP);

            client.subscribe(MQTT\_TOPIC\_SUB\_HUMID);

        } else {

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

            Serial.print(client.state());

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

            delay(5000);

        }

    }

}

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

    String message;

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

        message += (char)payload[i];

    }

    Serial.print("Message arrived [");

    Serial.print(topic);

    Serial.print("] ");

    Serial.println(message);

    // Handle MQTT messages here if needed

}

float roundToDecimal(float value, int decimals) {

    float scale = pow(10, decimals);

    return round(value \* scale) / scale;

}

void handleBuzzerOn() {

    userBuzzerControl = true;  // User wants to control the buzzer

    buzzerState = HIGH;  // Set flag to indicate that the buzzer is on

    digitalWrite(BUZZER\_PIN, HIGH);  // Turn on the buzzer

    server.send(200, "text/html", "Buzzer activated!");

}

void handleBuzzerOff() {

    userBuzzerControl = false;  // User wants to control the buzzer

    buzzerState = LOW;  // Clear flag to indicate that the buzzer is off

    digitalWrite(BUZZER\_PIN, LOW);  // Turn off the buzzer

    server.send(200, "text/html", "Buzzer is off");

}

void loop() {

    server.handleClient();

    // MQTT communication

    if (!client.connected()) {

        reconnect();

    }

    client.loop();

    // Sensor readings and actions

    static unsigned long lastPublish = 0;

    if (millis() - lastPublish > 2000) {  // Publish every 2 seconds

        lastPublish = millis();

        float h = dht.readHumidity();

        float t = dht.readTemperature();

        if (isnan(h) || isnan(t)) {

            Serial.println("Failed to read from DHT sensor!");

            lcd.setCursor(0, 0);

            lcd.print("Sensor error      ");

            return;

        }

        // Round temperature and humidity to 1 decimal place

        t = roundToDecimal(t, 1);

        h = roundToDecimal(h, 1);

        // Publish temperature and humidity

        doc["temperature"] = t;

        doc["humidity"] = h;

        char buffer[256];

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

        client.publish(MQTT\_TOPIC\_PUB\_TEMP, buffer, n);

        client.publish(MQTT\_TOPIC\_PUB\_HUMID, buffer, n);

        Serial.print("Temperature: ");

        Serial.print(t);

        Serial.print(" °C, Humidity: ");

        Serial.print(h);

        Serial.println(" %");

        // Display temperature and humidity on LCD

        lcd.clear();

        lcd.setCursor(0, 0);

        lcd.print("Temp: ");

        lcd.print(t);

        lcd.print(" C");

        lcd.setCursor(0, 1);

        lcd.print("Humidity: ");

        lcd.print(h);

        lcd.print(" %");

        // Check if temperature is above threshold to activate the buzzer

        if (t > temperatureThreshold && userBuzzerControl) {

            buzzerState = HIGH;  // Turn on the buzzer

        } else if (!userBuzzerControl) {

            buzzerState = LOW;  // Keep the buzzer off if user has turned it off

        }

        // If buzzer state is HIGH, turn on the buzzer

        if (buzzerState == HIGH) {

            digitalWrite(BUZZER\_PIN, HIGH);

        } else {

            digitalWrite(BUZZER\_PIN, LOW);

        }

        delay(100);  // Sleep 100ms to reduce CPU load

    }

}

1. Code frontend react native

import React, { useEffect, useState } from "react";

import { Text, View, StyleSheet, TouchableOpacity, Image } from "react-native";

import AsyncStorage from "@react-native-async-storage/async-storage";

import init from "react\_native\_mqtt";

import Icon from "react-native-vector-icons/FontAwesome"; // Import FontAwesome icon library

init({

    size: 10000,

    storageBackend: AsyncStorage,

    defaultExpires: 1000 \* 3600 \* 24,

    enableCache: true,

    sync: {},

});

const options = {

    host: "broker.emqx.io",

    port: 8083,

    path: "/ProjectKha",

    id: "id\_" + parseInt(Math.random() \* 100000),

};

const client = new Paho.MQTT.Client(options.host, options.port, options.path);

const TemperatureAndHumidityScreen = () => {

    const [temperature, setTemperature] = useState(null);

    const [humidity, setHumidity] = useState(null);

    useEffect(() => {

        connect();

        client.onMessageArrived = onMessageArrived;

    }, []);

    const connect = () => {

        client.connect({

            onSuccess: () => {

                console.log("connect MQTT broker ok!");

                subscribeTopic();

            },

            useSSL: false,

            timeout: 5,

            onFailure: () => {

                console.log("connect fail");

                connect();

                console.log("reconnect ...");

            },

        });

    };

    const subscribeTopic = () => {

        client.subscribe("ProjectKha/temp", { qos: 0 });

        client.subscribe("ProjectKha/humid", { qos: 0 });

    };

    const onMessageArrived = async (message) => {

        const jsondata = JSON.parse(message.payloadString);

        if (jsondata.temperature) {

            setTemperature(jsondata.temperature);

        }

        if (jsondata.humidity) {

            setHumidity(jsondata.humidity);

        }

    };

    const handleSpeakerOn = async () => {

        try {

            const result = await fetch('http://192.168.110.6/Buzzer/on');

            console.log('Speaker turned on');

        } catch (error) {

            console.error(error);

        }

    };

    const handleSpeakerOff = async () => {

        try {

            const result = await fetch('http://192.168.110.6/Buzzer/off');

            console.log('Speaker turned off');

        } catch (error) {

            console.error(error);

        }

    };

    // Function to determine icon color based on temperature

    const getTemperatureIconColor = () => {

        if (temperature >= 30) {

            return "#FF5733"; // Hot temperature color

        } else if (temperature <= 10) {

            return "#3498DB"; // Cold temperature color

        } else {

            return "#333"; // Default color

        }

    };

    // Function to determine icon color based on humidity

    const getHumidityIconColor = () => {

        if (humidity >= 70) {

            return "#2ECC71"; // High humidity color

        } else if (humidity <= 30) {

            return "#F39C12"; // Low humidity color

        } else {

            return "#333"; // Default color

        }

    };

    return (

        <View style={styles.container}>

            <View style={styles.header}>

                <Text style={styles.headerText}>Plant Manager</Text>

            </View>

            <View style={styles.imageContainer}>

                <Image

                    source={require('./img/Tree.jpg')} // Local image

                    style={styles.plantImage}

                />

            </View>

            <View style={styles.temperatureHumidityContainer}>

                <View style={styles.iconTextContainer}>

                    <Icon name="thermometer-half" size={30} color={getTemperatureIconColor()} />

                    <Text style={[styles.text, { color: getTemperatureIconColor() }]}>Temperature: {temperature} °C</Text>

                </View>

                <View style={styles.iconTextContainer}>

                    <Icon name="tint" size={30} color={getHumidityIconColor()} />

                    <Text style={[styles.text, { color: getHumidityIconColor() }]}>Humidity: {humidity} %</Text>

                </View>

            </View>

            <View style={styles.speakerControl}>

                <TouchableOpacity style={[styles.btn, styles.speakerOn]} onPress={handleSpeakerOn}>

                    <Icon name="volume-up" size={30} color="#FFF" />

                </TouchableOpacity>

                <TouchableOpacity style={[styles.btn, styles.speakerOff]} onPress={handleSpeakerOff}>

                    <Icon name="volume-off" size={30} color="#FFF" />

                </TouchableOpacity>

            </View>

        </View>

    );

};

const styles = StyleSheet.create({

    container: {

        flex: 1,

        backgroundColor: "#E6F0ED",

    },

    header: {

        width: "100%",

        padding: 20,

        backgroundColor: "#2ECC71",

        justifyContent: "center",

        alignItems: "center",

    },

    headerText: {

        color: "#FFF",

        fontSize: 24,

        fontWeight: "bold",

    },

    imageContainer: {

        alignItems: "center",

        marginVertical: 20,

        marginTop:90

    },

    plantImage: {

        width: 200, // Increased size

        height: 200, // Increased size

        borderRadius: 100, // Adjusted for larger size to maintain circular shape

        borderWidth: 1,

        borderColor: "#2ECC71",

    },

    temperatureHumidityContainer: {

        alignItems: "center",

        marginVertical: 20,

    },

    iconTextContainer: {

        flexDirection: "row",

        alignItems: "center",

        marginVertical: 10,

    },

    text: {

        fontSize: 20,

        marginLeft: 10,

    },

    speakerControl: {

        flexDirection: "row",

        justifyContent: "space-around",

        width: "80%",

        marginTop: 20,

        alignSelf: "center",

    },

    btn: {

        paddingVertical: 15,

        borderRadius: 50,

        width: 60,

        height: 60,

        justifyContent: "center",

        alignItems: "center",

    },

    speakerOn: {

        backgroundColor: "#4CAF50",

    },

    speakerOff: {

        backgroundColor: "#F44336",

    },

});

export default TemperatureAndHumidityScreen;

1. Code backend expressjs

const mqtt = require('mqtt');

const express = require('express');

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

const mysql = require('mysql2');

// MQTT Broker details

const MQTT\_SERVER = 'mqtt://broker.emqx.io';

const MQTT\_TOPIC\_TEMP = 'ProjectKha/temp';

const MQTT\_TOPIC\_HUMID = 'ProjectKha/humid';

const app = express();

app.use(bodyParser.json());

// MySQL database connection pool

const pool = mysql.createPool({

    host: 'mysql-container',

    user: 'root',

    password: '123456789',

    database: 'dht11\_data',

    waitForConnections: true,

    connectionLimit: 10,

    queueLimit: 0

});

pool.getConnection((err, connection) => {

    if (err) {

        console.error('Failed to connect to MySQL:', err);

        return;

    }

    console.log('Connected to MySQL');

    connection.release(); // Release the connection

});

// Connect to the MQTT broker

const client = mqtt.connect(MQTT\_SERVER);

client.on('connect', () => {

    console.log('Connected to MQTT broker');

    client.subscribe(MQTT\_TOPIC\_TEMP, (err) => {

        if (err) {

            console.error(`Failed to subscribe to ${MQTT\_TOPIC\_TEMP}:`, err);

        } else {

            console.log(`Subscribed to ${MQTT\_TOPIC\_TEMP}`);

        }

    });

    client.subscribe(MQTT\_TOPIC\_HUMID, (err) => {

        if (err) {

            console.error(`Failed to subscribe to ${MQTT\_TOPIC\_HUMID}:`, err);

        } else {

            console.log(`Subscribed to ${MQTT\_TOPIC\_HUMID}`);

        }

    });

});

client.on('message', (topic, message) => {

    try {

        const data = JSON.parse(message.toString());

        const temperature = data.temperature;

        const humidity = data.humidity;

        if (temperature !== null && humidity !== null) {

            pool.getConnection((err, connection) => {

                if (err) {

                    console.error('Error getting MySQL connection from pool:', err);

                    return;

                }

                const query = 'INSERT INTO sensor\_readings (temperature, humidity) VALUES (?, ?)';

                connection.query(query, [temperature, humidity], (err, results) => {

                    connection.release(); // Release the connection

                    if (err) {

                        console.error('Failed to insert data into MySQL:', err);

                    } else {

                        console.log('Data inserted into MySQL:', results);

                    }

                });

            });

        } else {

            console.log('Received null value for temperature or humidity');

        }

    } catch (error) {

        console.error('Error parsing MQTT message:', error);

    }

});

// Define routes

app.get('/sensor-data', (req, res) => {

    const query = 'SELECT \* FROM sensor\_readings ORDER BY reading\_time DESC LIMIT 1';

    pool.getConnection((err, connection) => {

        if (err) {

            console.error('Error getting MySQL connection from pool:', err);

            res.status(500).json({ error: 'Failed to retrieve data' });

            return;

        }

        connection.query(query, (err, results) => {

            connection.release(); // Release the connection

            if (err) {

                console.error('Failed to query data from MySQL:', err);

                res.status(500).json({ error: 'Failed to retrieve data' });

            } else {

                res.json(results[0]);

            }

        });

    });

});

// Start the Express server

const PORT = process.env.PORT || 3000;

app.listen(PORT, () => {

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

});

1. Dockerfile cho backend

# Use a specific version of Node.js

FROM node:14

# Set working directory

WORKDIR /usr/app

# Copy package.json and package-lock.json separately to leverage Docker cache

COPY package.json ./

COPY package-lock.json ./

# Install dependencies

RUN npm install

# Copy the rest of the application code

COPY . .

# Expose the port the app runs on

EXPOSE 3000

# Run the application

CMD ["npm", "start"]

1. docker-compose.yml

version: '3'

services:

  mysql-container:

    build:

      context: ./Database  # Thay đổi đường dẫn này cho phù hợp

      dockerfile: Dockerfile

    container\_name: mysql-container

    image: mysql:latest

    restart: always

    environment:

      MYSQL\_ROOT\_PASSWORD: 123456789

      MYSQL\_DATABASE: dht11\_data

    ports:

      - "3307:3306"  # Expose MySQL port 3306 inside the container to port 3307 on host

    volumes:

      - ./Database:/docker-entrypoint-initdb.d  # Mount the whole Database directory into the container

      - ./mysql\_data:/var/lib/mysql  # Mount the data directory from host into MySQL data directory

    expose:

      - 3306  # Expose MySQL port 3306 for other services to connect

  mqtt-broker:

    container\_name: mqtt-broker

    image: eclipse-mosquitto:latest

    ports:

      - "1883:1883"

    volumes:

      - ./mosquitto/config:/mosquitto/config

    command: mosquitto -c /mosquitto/config/mosquitto.conf

  backend:

    container\_name: backend-container

    build:

      context: ./backend

      dockerfile: Dockerfile

    ports:

      - "3000:3000"

    depends\_on:

      - mysql-container

      - mqtt-broker

    environment:

      - MQTT\_BROKER=mqtt-broker

      - MQTT\_PORT=1883

      - MYSQL\_HOST=mysql-container

      - MYSQL\_USER=root

      - MYSQL\_PASSWORD=123456789

      - MYSQL\_DATABASE=dht11\_data

1. Database

CREATE DATABASE IF NOT EXISTS dht11\_data;

USE dht11\_data;

CREATE TABLE IF NOT EXISTS sensor\_readings (

    id INT AUTO\_INCREMENT PRIMARY KEY,

    temperature FLOAT,

    humidity FLOAT,

    reading\_time TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

1. Dockerfile cho database

# Sử dụng image MySQL chính thức từ Docker Hub

FROM mysql:latest

# Đặt biến môi trường cho cơ sở dữ liệu

ENV MYSQL\_DATABASE dht11\_data

ENV MYSQL\_ROOT\_PASSWORD 123456789

# Sao chép tệp khởi tạo và dữ liệu SQL vào thư mục /docker-entrypoint-initdb.d/ trong container

# Tệp SQL này sẽ tự động thực thi khi container MySQL khởi động

COPY ./data.sql /docker-entrypoint-initdb.d/

# Cổng mặc định MySQL

EXPOSE 3306