TRƯỜNG ĐẠI HỌC KHOA HỌC

KHOA CÔNG NGHỆ THÔNG TIN

TÊN ĐỀ TÀI TIỂU LUẬN:

ĐIỀU KHIỂN QUẠT DỰA TRÊN NHIỆT ĐỘ VỚI ESP32

PHÁT TRIỂN ỨNG DỤNG IOT - 2024-2025.2.TIN4024.005

GIẢNG VIÊN HƯỚNG DẪN: VÕ VIỆT DŨNG

HUẾ, THÁNG 3 NĂM 2025

LỜI MỞ ĐẦU

Trong bối cảnh công nghệ phát triển mạnh mẽ, các hệ thống tự động hóa ngày càng đóng vai trò quan trọng trong đời sống hằng ngày. Một trong những ứng dụng phổ biến của tự động hóa là điều khiển thiết bị điện theo điều kiện môi trường, giúp tiết kiệm năng lượng và nâng cao tiện ích cho người dùng.

Tiểu luận này trình bày về hệ thống điều khiển quạt tự động dựa trên nhiệt độ, sử dụng vi điều khiển ESP32 và cảm biến nhiệt độ DHT22. Hệ thống cho phép điều chỉnh quạt theo nhiệt độ môi trường một cách tự động, đồng thời hỗ trợ điều khiển thủ công thông qua nút nhấn hoặc từ xa qua ứng dụng Blynk trên điện thoại. Với khả năng linh hoạt và dễ dàng mở rộng, hệ thống này có thể được ứng dụng trong nhiều lĩnh vực như nhà thông minh, phòng thí nghiệm, và các không gian yêu cầu kiểm soát nhiệt độ hiệu quả.

Mục tiêu của tiểu luận là giới thiệu nguyên lý hoạt động, mô hình thiết kế và cách lập trình hệ thống này, nhằm giúp người đọc có cái nhìn tổng quan về ứng dụng của IoT trong điều khiển thiết bị điện thông minh.

Mục lục

1. Giới thiệu chung

Hệ thống điều khiển quạt thông minh là một trong những ứng dụng phổ biến trong lĩnh vực IoT (*Internet of Things*). Các hệ thống này cho phép giám sát và điều chỉnh nhiệt độ trong môi trường bằng cách tự động bật hoặc tắt quạt khi nhiệt độ vượt ngưỡng cho phép. Điều này không chỉ giúp tiết kiệm điện năng mà còn mang lại sự tiện lợi và tối ưu hóa hiệu suất làm mát.

1.1. Giới thiệu về IoT và ứng dụng trong điều khiển

Internet of Things (IoT) là một mô hình công nghệ hiện đại, nơi các thiết bị có thể kết nối với nhau qua Internet để thu thập và trao đổi dữ liệu. Trong lĩnh vực điều khiển quạt thông minh, IoT cho phép người dùng giám sát và điều khiển thiết bị từ xa thông qua ứng dụng di động hoặc nền tảng đám mây.

Ứng dụng của IoT trong điều khiển quạt bao gồm:

- Tự động hóa: Hệ thống có thể tự động phản hồi theo nhiệt độ môi trường.
- Điều khiển từ xa: Người dùng có thể điều chỉnh quạt thông qua các ứng dụng di động như Blynk.
- Giám sát và báo cáo: Hệ thống có thể ghi lại dữ liệu nhiệt độ và trạng thái quạt để phân tích sau này.

1.2. Tổng quan về bộ vi điều khiển ESP32

ESP32 là một bộ vi điều khiển mạnh mẽ, hỗ trợ cả kết nối Wi-Fi và Bluetooth, giúp nó trở thành lựa chọn lý tưởng cho các dự án IoT. Một số đặc điểm chính của ESP32:

- Bộ xử lý: Dual-core 32-bit, tốc độ lên đến 240MHz.
- Kết nối: Wi-Fi 802.11 b/g/n, Bluetooth 4.2.
- GPIO: Hỗ trợ nhiều chân vào/ra để kết nối với cảm biến và thiết bị ngoại vi.
- Tiêu thụ điện năng thấp: Có chế độ Deep Sleep để tiết kiệm năng lượng.

ESP32 thường được sử dụng để thu thập dữ liệu từ cảm biến, xử lý thông tin và điều khiển thiết bị đầu ra như relay, LED, hoặc động cơ.

1.3. Cảm biến DHT22 và vai trò trong đo lường nhiệt độ

Cảm biến DHT22 là một trong những cảm biến đo nhiệt độ và độ ẩm phổ biến nhất, thường được sử dụng trong các hệ thống tự động hóa nhà thông minh. Một số thông tin quan trọng về DHT22:

- Dải đo nhiệt độ: -40°C đến 80°C, độ chính xác ±0.5°C.
- Dải đo độ ẩm: 0% đến 100% RH, độ chính xác ±2-5%.
- Giao tiếp: Sử dụng giao thức 1-Wire để truyền dữ liệu đến bộ vi điều khiển.

DHT22 có độ chính xác cao hơn so với DHT11, do đó được lựa chọn cho các ứng dụng cần độ tin cậy tốt hơn.

1.4. Ứng dụng của relay trong điều khiển quạt

Relay là một thiết bị đóng/ngắt mạch điện bằng tín hiệu điều khiển từ bộ vi điều khiển. Trong hệ thống này, relay được sử dụng để bật hoặc tắt quạt khi nhiệt độ thay đổi. Nguyên lý hoạt động của relay:

- Khi nhận tín hiệu từ ESP32, relay sẽ đóng mạch và cho phép dòng điện chạy qua để cấp nguồn cho quạt.
- Khi tín hiệu bị ngắt, relay sẽ mở mạch và tắt quạt.

Relay có nhiều loại, nhưng trong hệ thống này, loại relay 5V hoặc 12V thường được sử dụng để tương thích với mạch điều khiển.

1.5. Ứng dụng của Blynk trong giám sát và điều khiển từ xa

Blynk là một nền tảng IoT cho phép người dùng tạo ứng dụng di động để điều khiển thiết bị phần cứng. Blynk hoạt động dựa trên các widget có thể tùy chỉnh để giám sát và điều khiển thiết bị từ xa. Các tính năng chính của Blynk bao gồm:

- Điều khiển thiết bị từ xa thông qua Wi-Fi.
- Hiển thị dữ liệu thời gian thực từ cảm biến.
- Gửi thông báo khi có sự kiện xảy ra (ví dụ: nhiệt độ vượt ngưỡng).
- Ghi lại dữ liệu để phân tích.

Trong hệ thống này, Blynk sẽ hiển thị nhiệt độ hiện tại, cho phép người dùng chuyển đổi giữa chế độ tự động và thủ công, cũng như điều khiển quạt từ xa.

2. Giới thiệu về ESP32, cảm biến DHT22 và Relay Module

2.1. Bộ vi điều khiển ESP32

ESP32 là một bộ vi điều khiển (microcontroller) tích hợp Wi-Fi và Bluetooth do hãng Espressif Systems phát triển. Nó được xem là phiên bản nâng cấp mạnh mẽ của ESP8266, với khả năng xử lý nhanh hơn, nhiều chân I/O hơn và hỗ trợ nhiều chức năng hơn. Một số đặc điểm nổi bật của ESP32:

- **CPU:** Dual-core Xtensa LX6, tốc độ lên đến 240MHz.
- **RAM:** 520 KB SRAM nội bộ.
- **Bộ nhớ Flash:** Thường từ 4MB đến 16MB tùy loại module.
- **Wi-Fi:** Tích hợp Wi-Fi chuẩn 802.11 b/g/n và Bluetooth 4.2 BLE.
- **Hỗ trợ giao tiếp đa dạng:** SPI, I2C, UART, ADC, PWM, DAC…

ESP32 thường được sử dụng trong các dự án IoT nhờ khả năng giao tiếp mạng tốt, tiêu thụ điện năng thấp và có thể lập trình bằng Arduino IDE hoặc PlatformIO. Đặc biệt, nó có khả năng xử lý song song nhiều tác vụ với FreeRTOS – một hệ điều hành thời gian thực tích hợp sẵn.

Một ứng dụng cụ thể trong dự án này là ESP32 sẽ thu thập dữ liệu nhiệt độ từ cảm biến DHT22, xử lý so sánh với ngưỡng đặt trước và điều khiển relay bật/tắt quạt. Đồng thời, dữ liệu này cũng được gửi đến ứng dụng Blynk qua kết nối Wi-Fi.

2.2. Tổng quan về cảm biến DHT22

Cảm biến DHT22 là một module tích hợp dùng để đo nhiệt độ và độ ẩm. Nó là phiên bản nâng cấp của DHT11, cho độ chính xác cao hơn và dải đo rộng hơn, rất phù hợp cho các ứng dụng yêu cầu tính ổn định và độ tin cậy cao. Một số thông số kỹ thuật chính:

- **Dải đo nhiệt độ:** từ -40°C đến +80°C
- **Sai số nhiệt độ:** ±0.5°C
- **Dải đo độ ẩm:** 0 – 100% RH
- **Sai số độ ẩm:** ±2% đến ±5% RH
- **Tần suất lấy mẫu:** khoảng 0.5 Hz (2 giây/lần)

- **Điện áp hoạt động:** 3.3V – 6V
- **Giao tiếp:** Một dây (One-Wire)

Cảm biến DHT22 hoạt động bằng cách sử dụng một cảm biến điện dung để đo độ ẩm và một nhiệt điện trở (thermistor) để đo nhiệt độ. Dữ liệu được mã hóa theo định dạng số và truyền qua một chân tín hiệu duy nhất đến vi điều khiển. Trong dự án này, cảm biến DHT22 giúp cung cấp dữ liệu thời gian thực về nhiệt độ cho ESP32 để điều khiển hoạt động của quạt.

2.3. Relay Module

Relay là một công tắc điện tử điều khiển bằng tín hiệu điện áp thấp nhưng có khả năng bật/tắt các thiết bị điện có công suất cao hơn, như quạt hoặc đèn. Relay module thường tích hợp sẵn transistor, diode và đầu nối để dễ dàng kết nối với vi điều khiển như ESP32. Một số đặc điểm:

- **Điện áp điều khiển:** 3.3V hoặc 5V (phù hợp với ESP32)
- **Dòng điện tải:** có thể điều khiển thiết bị AC 220V hoặc DC lên đến 10A
- **Cách ly quang:** bảo vệ mạch điều khiển khỏi nhiễu và điện áp ngược
- **Hoạt động theo logic đảo:** mức LOW kích hoạt relay (do sử dụng transistor)

Trong hệ thống này, khi ESP32 phát hiện nhiệt độ vượt ngưỡng hoặc nhận tín hiệu điều khiển từ người dùng, nó sẽ cấp tín hiệu điều khiển cho relay module. Relay sẽ đóng mạch điện cấp nguồn cho quạt, giúp bật quạt lên. Khi không cần làm mát nữa, relay sẽ ngắt mạch và tắt quạt. Relay đóng vai trò trung gian giữa mạch điều khiển điện áp thấp và thiết bị tiêu thụ điện áp cao.

2.4. Lý do chọn ESP32-DevKitC V4 và DHT22 cho hệ thống

ESP32-DevKitC V4 là một board phát triển phổ biến dựa trên module ESP32-WROOM-32 do chính hãng Espressif sản xuất. Board này có thiết kế nhỏ gọn, giá thành thấp và được hỗ trợ rộng rãi bởi cộng đồng Arduino. Lý do chọn ESP32-DevKitC V4 cho hệ thống bao gồm:

- **Tích hợp Wi-Fi và Bluetooth:** Cho phép kết nối điều khiển thiết bị từ xa qua mạng không dây, rất lý tưởng cho các ứng dụng IoT như hệ thống quạt thông minh.

- **Đầy đủ chân GPIO:** ESP32-DevKitC V4 cung cấp nhiều chân I/O kỹ thuật số, cho phép kết nối với cảm biến DHT22, relay, nút nhấn, đèn LED, v.v.
- **Tương thích Arduino IDE:** Dễ lập trình, có nhiều thư viện hỗ trợ sẵn giúp giảm thời gian phát triển hệ thống.
- **Xử lý nhanh và ổn định:** Vi xử lý dual-core tốc độ cao giúp xử lý dữ liệu nhiệt độ, điều khiển thiết bị và giao tiếp mạng một cách hiệu quả.
- **Giá thành rẻ, dễ mua:** Board ESP32-DevKitC V4 phổ biến trên thị trường, dễ tiếp cận cho sinh viên và người mới bắt đầu.

DHT22 cũng là lựa chọn tối ưu cho hệ thống nhờ vào các yếu tố:

- **Độ chính xác cao:** Sai số nhiệt độ ±0.5°C, thích hợp cho các ứng dụng yêu cầu giám sát chính xác.
- **Dải đo rộng:** Từ -40°C đến +80°C, đáp ứng hầu hết các điều kiện nhiệt độ thường gặp.
- **Giao tiếp đơn giản:** Dễ kết nối với ESP32 qua một chân dữ liệu, tiết kiệm dây nối và chân GPIO.

Với sự kết hợp giữa ESP32-DevKitC V4 và cảm biến DHT22, hệ thống vừa có khả năng xử lý mạnh mẽ, vừa đảm bảo độ chính xác và độ ổn định cao trong quá trình đo nhiệt độ và điều khiển thiết bị. Đây là lựa chọn phù hợp cho các ứng dụng điều khiển tự động trong thực tế.

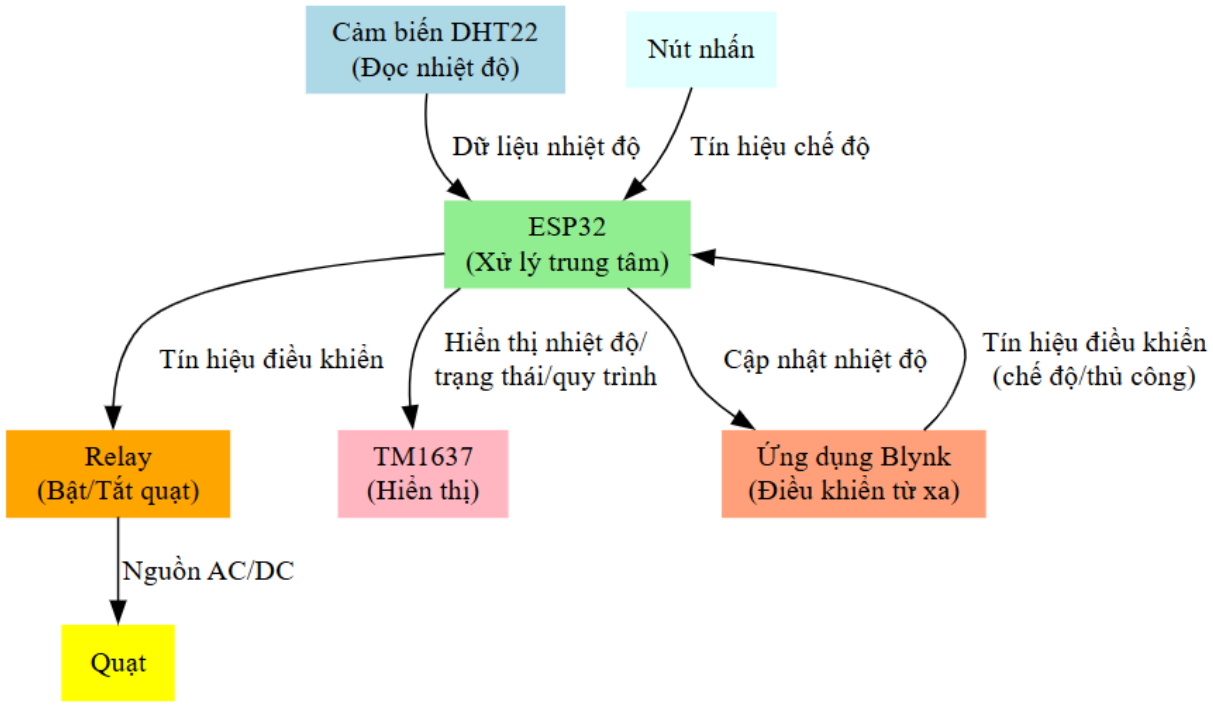
3. Nguyên lý hoạt động

Hệ thống điều khiển quạt dựa trên nhiệt độ sử dụng ESP32 hoạt động theo nguyên lý sau:

- Cảm biến DHT22 đo nhiệt độ môi trường và gửi dữ liệu đến ESP32 theo chu kỳ đều đặn.
- ESP32 sẽ xử lý giá trị nhiệt độ:
  + Nếu **chế độ tự động** được kích hoạt:
    - Quạt sẽ được **bật** nếu nhiệt độ vượt quá ngưỡng cài đặt (ví dụ: 32°C).
    - Quạt sẽ được **tắt** nếu nhiệt độ thấp hơn ngưỡng.
    - Số đếm ngược sẽ hiển thị 3 giây/lần để thể hiện chu kỳ kiểm tra nhiệt độ.

+ Nếu **chế độ thủ công** được chọn:
    - Người dung có thể bật/tắt quạt thủ công từ ứng dụng Blynk hoặc nút nhấn trên mạch
- Màn hình TM1637 sẽ hiển thị nhiệt độ và trạng thái quạt.
- Ứng dụng Blynk hiển thị nhiệt độ hiện tại và trạng thái quạt. Người dùng có thể chuyển đổi chế độ hoặc bật/tắt quạt từ xa.

3.1. Sơ đồ nguyên lý hoạt động



3.2. Chu trình xử lý của ESP32

Bước 1: Khởi động hệ thống

Bước 2: Kiểm tra chế độ hoạt động (thủ công hoặc tự động)

Bước 3.1: Nếu tự động:

- Đọc nhiệt độ từ DHT22
- So sánh với ngưỡng:
    - Nếu nhiệt độ ≥ ngưỡng → bật quạt.
    - Nếu nhiệt độ < ngưỡng → tắt quạt.

Bước 3.2: Nếu thủ công:

- Lắng nghe tín hiệu từ nút nhấn hoặc Blynk.
- Bật/tắt quạt theo yêu cầu.

Bước 4: Cập nhật TM1637 và ứng dụng Blynk.

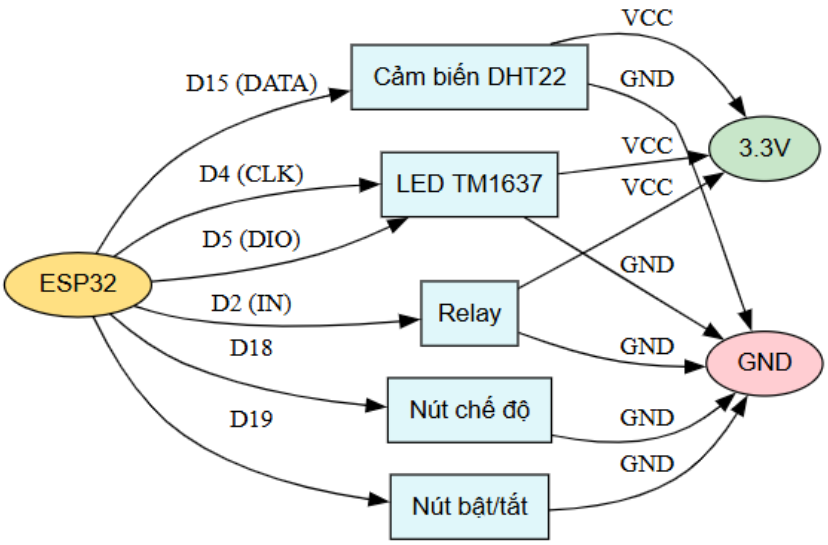
Bước 5: Lặp lại chu kỳ.

4. Mô hình thiết kế

4.1 Bảng sơ đồ kết nối phần cứng

Thiết bị	Chân thiết bị	Kết nối đến ESP32 (chân)	Ghi chú
DHT22	SDA(Data)	GPIO15	Đọc nhiệt độ
	VCC	5V	Nguồn cho cảm biến
	GND	GND	Mass
OLED	SDA	GPIO21	Xung đồng hồ
	SCL	GPIO22	Dữ liệu
	VCC	5V	Nguồn LED
	GND	GND	Mass
Relay	IN	GPIO23	Tín hiệu điều khiển relay
	VCC	5V	Nguồn
	GND	GND	Mass
Nút nhấn	Nút đỏ	GPIO26	Chuyển đổi chế độ
	Nút đen	GPIO32	Chuyển đổi trạng thái relay
	GND chung	GND	Mass
LED đỏ	Anode (A)	IN của Relay	Điều khiển gián tiếp qua Relay
LED tím	Anode (A)	GPIO12	Điều khiển trực tiếp từ ESP32

4.2. Sơ đồ khối

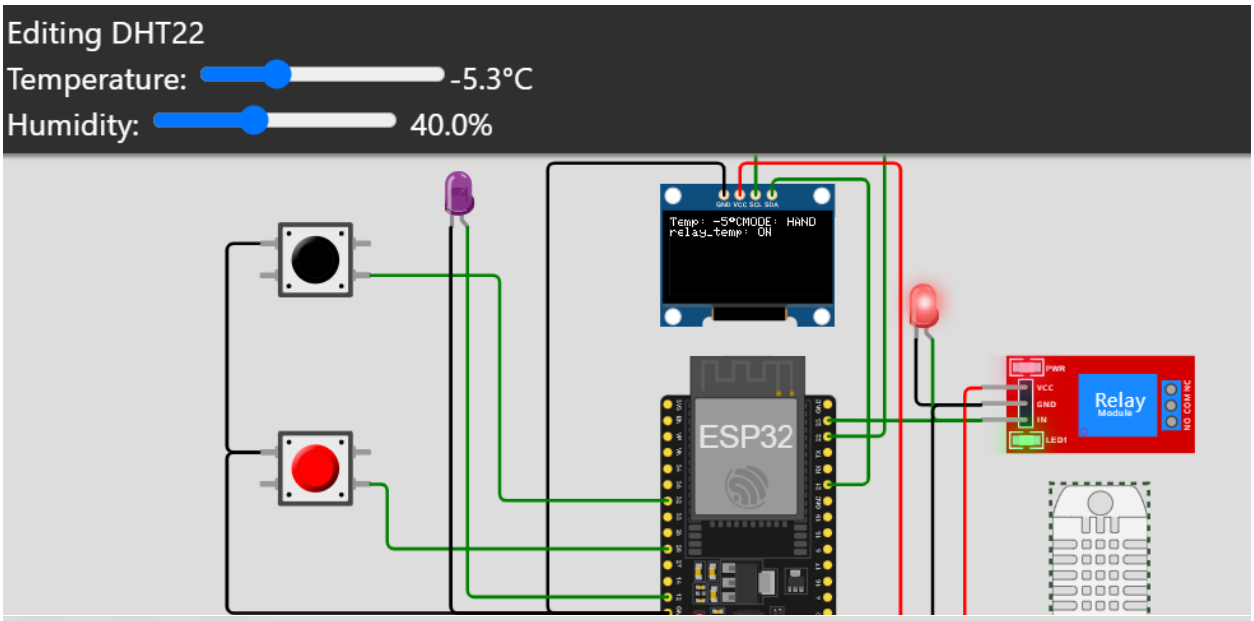


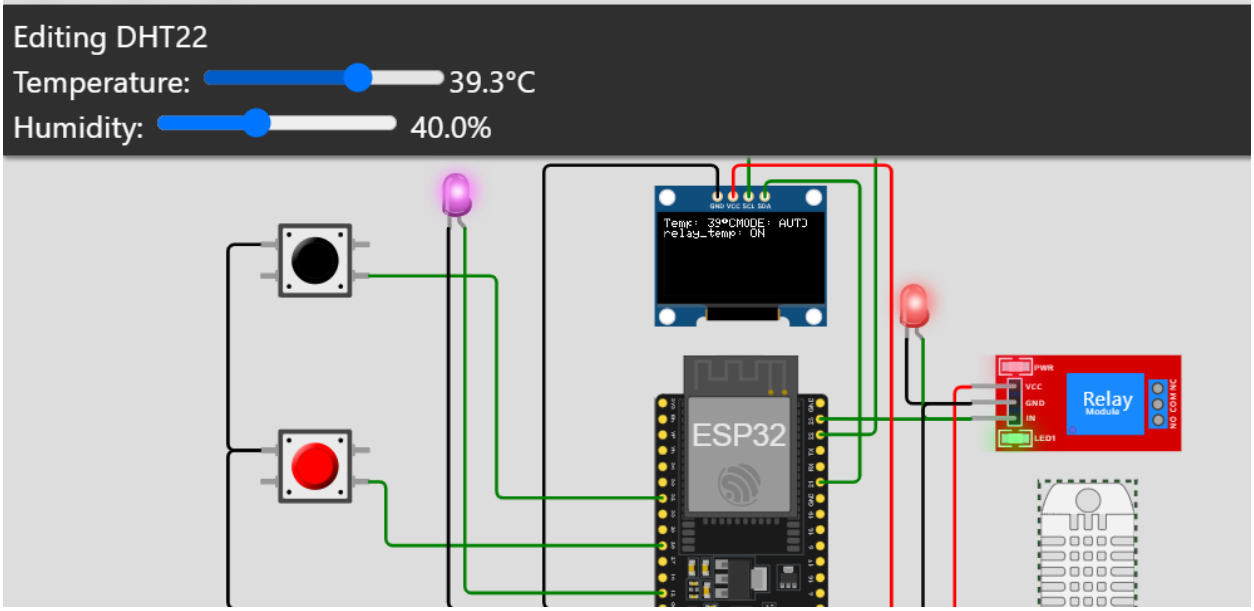
4.3. Sơ đồ Wokwi

```
{
  "version": 1,
  "author": "DPT",
  "editor": "wokwi",
  "parts": [
    { "type": "board-esp32-devkit-c-v4", "id": "esp", "top": 0, "left": 0,
"attrs": {} },
    {
      "type": "board-ssd1306",
      "id": "oled1",
      "top": -102.46,
      "left": 0.23,
      "attrs": { "i2cAddress": "0x3c" }
    },
    {
      "type": "wokwi-pushbutton",
      "id": "btn1",
      "top": 44.6,
      "left": -240,
      "attrs": { "color": "red", "bounce": "1" }
    },
    { "type": "wokwi-relay-module", "id": "relay2", "top": 0.2, "left": 192,
"attrs": {} },
    {
      "type": "wokwi-led",
```

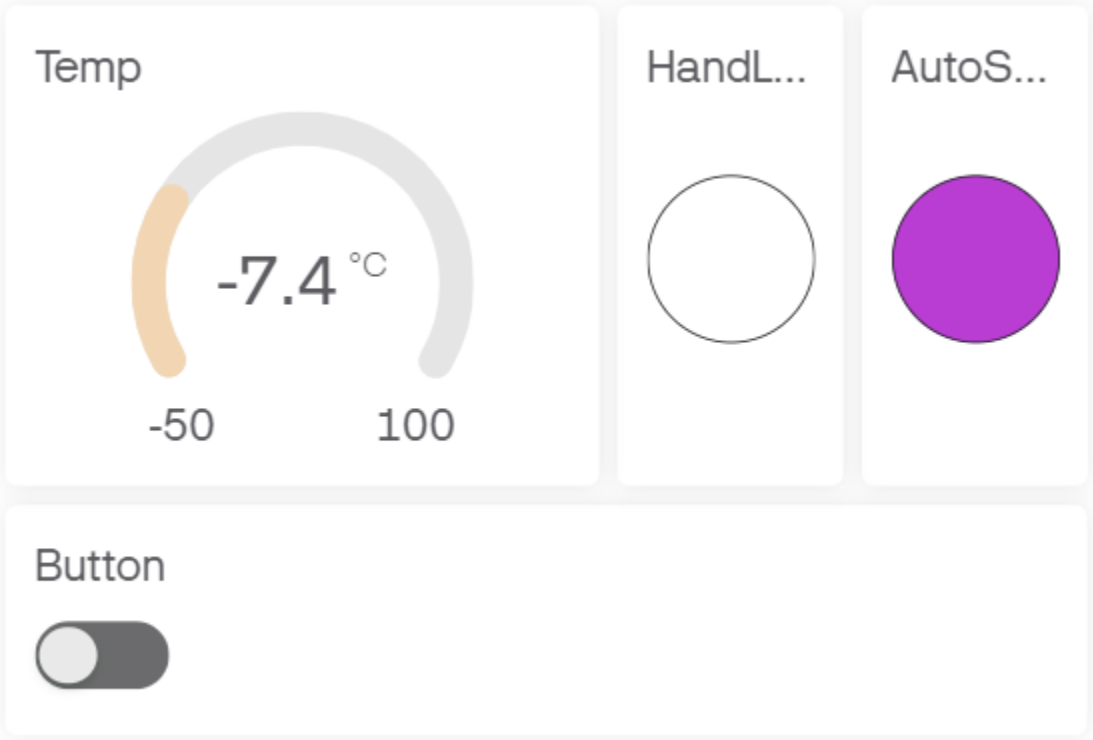
```
      "id": "led1",
      "top": -51.6,
      "left": 138.2,
      "attrs": { "color": "red" }
    },
    {
      "type": "wokwi-pushbutton",
      "id": "btn5",
      "top": -80.2,
      "left": -240,
      "attrs": { "color": "black" }
    },
    {
      "type": "wokwi-led",
      "id": "led5",
      "top": -118.8,
      "left": -140.2,
      "attrs": { "color": "purple" }
    },
    { "type": "wokwi-dht22", "id": "dht2", "top": 77.1, "left": 234.6, "attrs":
{} }
  ],
  "connections": [
    [ "esp:TX", "$serialMonitor:RX", "", [] ],
    [ "esp:RX", "$serialMonitor:TX", "", [] ],
    [ "oled1:SCL", "esp:22", "green", [ "v-28.8", "h77.1", "v172.8" ] ],
    [ "oled1:SDA", "esp:21", "green", [ "v-9.6", "h57.67", "v182.4" ] ],
    [ "relay2:IN", "esp:23", "green", [ "h-48", "v-38.6" ] ],
    [ "oled1:GND", "esp:GND.1", "black", [ "v-19.2", "h-105.6", "v268.8" ] ],
    [ "esp:26", "btn1:2.r", "green", [ "h-167.81", "v86.2" ] ],
    [ "led1:C", "relay2:GND", "black", [ "v0" ] ],
    [ "led1:A", "relay2:IN", "green", [ "v0" ] ],
    [ "btn5:2.r", "esp:32", "green", [ "h77", "v125" ] ],
    [ "led5:C", "esp:GND.1", "black", [ "v230.4", "h10" ] ],
    [ "led5:A", "esp:12", "green", [ "v0" ] ],
    [ "dht2:VCC", "esp:5V", "red", [ "v28.8", "h-316.8" ] ],
    [ "dht2:GND", "esp:GND.1", "black", [ "v38.4", "h-288", "v-76.8" ] ],
    [ "dht2:SDA", "esp:15", "green", [ "v9.6", "h-143.9", "v-28.8" ] ],
    [ "relay2:VCC", "dht2:VCC", "red", [ "h-9.6", "v163.2" ] ],
    [ "oled1:VCC", "dht2:VCC", "red", [ "v-19.2", "h96.15", "v307.2" ] ],
    [ "relay2:GND", "esp:GND.1", "black", [ "h-28.8", "v201.2", "h-172.8", "v-
76.8" ] ],
    [ "btn1:1.l", "esp:GND.1", "black", [ "h-19.2", "v96" ] ],
    [ "btn5:1.l", "btn1:1.l", "black", [ "h-19.2", "v124.8" ] ]
  ],
```

```
  "dependencies": {}
}
```





4.4. Giao diện Blynk



5. Chương trình điều khiển

```
#define BLYNK_TEMPLATE_ID "TMPL6lCsC-LTG"
#define BLYNK_TEMPLATE_NAME "ESP32FinalTest"
#define BLYNK_AUTH_TOKEN "8KSvJabgCjWzmnXJPMR2HxW2fSP2qXoB"

#include <DHT.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <WiFi.h>
#include <PubSubClient.h>
#include <BlynkSimpleEsp32.h>

// Định nghĩa các tham số màn hình OLED
#define SCREEN_WIDTH 128    // Chiều rộng màn hình OLED
#define SCREEN_HEIGHT 64    // Chiều cao màn hình OLED
#define OLED_RESET -1       // Reset mặc định cho màn hình OLED (không dùng)

// Khởi tạo màn hình OLED
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);

// Định nghĩa chân cảm biến và thiết bị
const int DHT_PIN = 15;      // Chân cảm biến DHT22
const int DHT_TYPE = DHT22;  // Loại cảm biến DHT22

const int LED = 12;  // Chân kết nối đèn LED

// Định nghĩa chân nút bấm và relay
const int button_temp = 26;   // Nút bấm điều khiển nhiệt độ

const int button_Mode = 32;   // Nút bấm thay đổi chế độ

const int relay_temp = 23;    // Relay điều khiển theo nhiệt độ
const int relay_dis = 16;     // Relay điều khiển thiết bị khác

// Định nghĩa các ngưỡng giá trị
float tempThreshold = 32;     // Ngưỡng nhiệt độ


volatile bool relayTemp_State = false;  // Trạng thái relay điều khiển nhiệt độ
volatile bool Mode_State = false;       // Biến lưu trạng thái của chế độ

// Biến lưu thời gian debounce cho các nút điều khiển (tránh rung nút)
volatile unsigned long lastDebounceTimeTemp = 0;
volatile unsigned long lastDebounceTimeMode = 0;
```

```
const unsigned long debounceDelay = 200;  // Thời gian chống rung nút 200ms

unsigned long previousMillis = 0;         // Lưu thời gian lần cập nhật trước cho
OLED
const long intervaloled = 100;             // Thời gian cập nhật OLED mỗi 100ms

DHT dht(DHT_PIN, DHT_TYPE);  // Khởi tạo đối tượng DHT với chân cảm biến và loại
DHT

const char* ssid = "Wokwi-GUEST";           // Tên mạng WiFi
const char* password = "";                  // Mật khẩu WiFi (trong trường hợp
này không cần)

WiFiClient espClient;         // Khởi tạo đối tượng WiFiClient cho ESP32
PubSubClient client(espClient);  // Khởi tạo đối tượng PubSubClient cho MQTT, sử
dụng WiFiClient

// Hàm kết nối WiFi
void connectWiFi()
{
    Serial.print("Connecting to WiFi...");
    WiFi.begin(ssid, password);  // Bắt đầu kết nối WiFi với SSID và password
    while (WiFi.status() != WL_CONNECTED)  // Kiểm tra trạng thái kết nối
    {
      delay(500);
      Serial.print(".");  // In ra dấu "." trong khi chờ kết nối
    }
    Serial.println(" Connected!");  // Thông báo đã kết nối thành công
}

// Ngắt ngoài để điều khiển relay (nhiệt độ)
void IRAM_ATTR buttonTempInterrupt() {
  if ((millis() - lastDebounceTimeTemp) > debounceDelay && !Mode_State) {  //
Kiểm tra chống rung và chế độ
    relayTemp_State = !relayTemp_State;  // Thay đổi trạng thái relay nhiệt độ
    digitalWrite(relay_temp, relayTemp_State ? HIGH : LOW);  // Bật/tắt relay
    lastDebounceTimeTemp = millis();  // Cập nhật thời gian debounce
  }
}

// Ngắt ngoài để điều khiển chế độ (Mode)
void IRAM_ATTR buttonModeInterrupt() {
  if ((millis() - lastDebounceTimeMode) > debounceDelay) {
    Mode_State = !Mode_State;  // Thay đổi trạng thái chế độ (AUTO/HAND)
```

```
    digitalWrite(LED, Mode_State ? HIGH : LOW);  // Bật/tắt LED báo hiệu chế độ
    lastDebounceTimeMode = millis();
  }
}

// Khởi tạo các cảm biến và thiết bị
void setup() {
  Serial.begin(115200);  // Khởi tạo cổng Serial
  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, password);  // Kết nối Blynk

  // Cấu hình chân cảm biến và relay
  pinMode(relay_dis, OUTPUT);
  pinMode(LED, OUTPUT);
  pinMode(button_temp, INPUT_PULLUP);
  pinMode(button_Mode, INPUT_PULLUP);
  pinMode(relay_temp, OUTPUT);
  dht.begin();  // Khởi động cảm biến DHT

  if (!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {  // Khởi động màn hình OLED
    Serial.println(F("SSD1306 allocation failed"));
    while (true);
  }
  display.clearDisplay();
  display.display();

  // Kết nối WiFi
  connectWiFi();

  // Thiết lập ngắt ngoài cho các nút bấm
  attachInterrupt(digitalPinToInterrupt(button_temp), buttonTempInterrupt,
FALLING);
  attachInterrupt(digitalPinToInterrupt(button_Mode), buttonModeInterrupt,
FALLING);
}

// Hàm đọc nhiệt độ và độ ẩm và điều khiển Relay
void readTemperatureAndControlLED(float &temperature) {
  temperature = dht.readTemperature();  // Đọc nhiệt độ từ cảm biến

  if (Mode_State) {  // Chỉ điều khiển trong chế độ AUTO
    if (temperature >= tempThreshold) {
      digitalWrite(relay_temp, HIGH); relayTemp_State = true;  // Bật relay nhiệt
độ nếu vượt ngưỡng
    } else if (temperature < tempThreshold) {
```

```
      digitalWrite(relay_temp, LOW); relayTemp_State = false;  // Tắt relay nhiệt
độ nếu dưới ngưỡng
    }
  }
}

// Hàm hiển thị dữ liệu lên OLED
void displayData(float temp) {
  display.clearDisplay(); // Xóa màn hình OLED
  display.setTextSize(0.8); // Thiết lập kích thước chữ
  display.setTextColor(WHITE); // Thiết lập màu chữ
  display.setCursor(0, 0); // Đặt vị trí con trỏ trên màn hình
  display.print("Temp: "); // In ra tiêu đề nhiệt độ
  display.print((int)temp); // In ra giá trị nhiệt độ
  display.print((char)247); // In ký hiệu độ C
  display.print("C");
  display.print("MODE: "); // In ra chế độ hoạt động
  display.println(Mode_State ? "AUTO" : "HAND"); // Hiển thị chế độ AUTO hoặc
HAND
  display.print("relay_temp: "); // In ra trạng thái relay nhiệt độ
  display.println(relayTemp_State ? "ON" : "OFF"); // Hiển thị ON/OFF
  display.display(); // Cập nhật màn hình
}

// Hàm hiển thị trạng thái hệ thống và relay lên Serial
void displayStatusToSerial(float temp) {
  Serial.println("===== System Status ====="); // In tiêu đề trạng thái
  Serial.print("Temperature: "); // In ra tiêu đề nhiệt độ
  Serial.print(temp); // In ra giá trị nhiệt độ
  Serial.println(" °C"); // Kết thúc dòng
  Serial.print("Mode: "); // In ra chế độ hoạt động
  Serial.println(Mode_State ? "AUTO" : "HAND"); // Hiển thị chế độ
  Serial.print("Relay Temp: "); // In ra trạng thái relay nhiệt độ
  Serial.println(relayTemp_State ? "ON" : "OFF"); // Hiển thị ON/OFF
  Serial.println("========================="); // In dòng ngăn cách
}
// Hàm gửi dữ liệu lên Blynk
void sendSensorDataToBlynk() {
  float temp = dht.readTemperature();
  Blynk.virtualWrite(V0, temp); // V0 là widget nhiệt độ trên app
  Blynk.virtualWrite(V1, Mode_State ? 1 : 0); // Cập nhật chế độ
  Blynk.virtualWrite(V2, relayTemp_State ? 1 : 0); // Cập nhật trạng thái relay
}
// Điều khiển chuyển đổi chế độ (AUTO/HAND) từ app Blynk
BLYNK_WRITE(V1) {
```

```
  int mode = param.asInt(); // Lấy giá trị từ app (0 hoặc 1)
  Mode_State = (mode == 1); // Nếu 1 thì AUTO, 0 là HAND
  digitalWrite(LED, Mode_State ? HIGH : LOW); // Cập nhật LED báo hiệu
}

// Điều khiển bật/tắt relay nhiệt độ từ app Blynk (chỉ hoạt động khi HAND)
BLYNK_WRITE(V2) {
  if (!Mode_State) { // Chỉ cho phép khi ở chế độ HAND
    int value = param.asInt(); // Lấy trạng thái nút từ app
    relayTemp_State = (value == 1);
    digitalWrite(relay_temp, relayTemp_State ? HIGH : LOW);
  }
}
unsigned long lastBlynkSend = 0;
const long intervalBlynk = 1000;

// Vòng lặp chính
void loop() {
  Blynk.run(); // Chạy Blynk

  unsigned long currentMillis = millis(); // Lấy thời gian hiện tại

  // Nếu đã đến lúc cập nhật (500ms trôi qua)
  if (currentMillis - previousMillis >= intervaloled) {
    previousMillis = currentMillis; // Cập nhật lại thời gian lần trước

    float temp; // Khai báo biến cho các cảm biến

    // Đọc cảm biến và điều khiển LED
    readTemperatureAndControlLED(temp); // Đọc nhiệt độ và độ ẩm

    // Hiển thị dữ liệu lên màn hình OLED
    displayData(temp); // Hiển thị dữ liệu trên OLED

    // Hiển thị trạng thái lên Serial
    displayStatusToSerial(temp); // Hiển thị trạng thái hệ thống trên Serial
  }

  // Blynk
  if (currentMillis - lastBlynkSend >= intervalBlynk) {
    lastBlynkSend = currentMillis;
    sendSensorDataToBlynk();
  }
}
```

Chương trình điều khiển này được lập trình cho một hệ thống sử dụng ESP32, Blynk, cảm biến DHT22, màn hình OLED và relay để điều khiển một thiết bị dựa trên nhiệt độ. Sau đây là giải thích chi tiết về các phần trong chương trình:

5.1. Khai báo các thư viện và thông số kết nối

- **Blynk**: Để kết nối với ứng dụng Blynk và điều khiển thiết bị từ xa.
- **DHT**: Thư viện để giao tiếp với cảm biến DHT22, dùng để đo nhiệt độ và độ ẩm.
- **Adafruit_SSD1306 và Adafruit_GFX**: Thư viện để điều khiển màn hình OLED 128x64.
- **Wi-Fi** : Thư viện cho việc kết nối WiFi.

5.2. Khai báo các thiết bị và chân kết nối

- **Chân cảm biến DHT22** (DHT_PIN) và **Loại cảm biến** (DHT_TYPE).
- **Các chân nút bấm** để thay đổi chế độ và điều khiển relay.
- **Relay** dùng để bật/tắt thiết bị hoặc hệ thống khi nhiệt độ vượt quá ngưỡng.

5.3. Kết nối Wi-Fi và Blynk

- Hàm connectWiFi() giúp kết nối ESP32 với WiFi.
- Blynk.begin() kết nối ESP32 với ứng dụng Blynk sử dụng **BLYNK_AUTH_TOKEN**, **SSID**, và **mật khẩu WiFi**.

5.4. Chế độ điều khiển tự động

- **AUTO**: Chế độ tự động điều khiển relay khi nhiệt độ vượt ngưỡng.
- **HAND**: Chế độ thủ công, người dùng có thể điều khiển relay qua ứng dụng Blynk hoặc nút bấm.
- **Đèn LED**: Bật khi ở chế độ **AUTO** và tắt khi ở chế độ **HAND**.

5.5. Điều khiển relay và cảm biến

- **Relay điều khiển theo nhiệt độ**: Khi nhiệt độ vượt quá ngưỡng (tempThreshold), relay được bật (nhiệt độ cao) hoặc tắt (nhiệt độ thấp).
- **Nút bấm**: Nút bấm cho phép người dùng chuyển đổi chế độ giữa **AUTO** và **HAND**, đồng thời điều khiển relay.

5.6. Cập nhật thông tin trên OLED

- Thông tin về nhiệt độ, chế độ và trạng thái relay được hiển thị trên màn hình OLED.

5.7. Gửi và nhận dữ liệu từ Blynk

- Dữ liệu nhiệt độ và trạng thái hệ thống được gửi lên ứng dụng Blynk qua **Blynk.virtualWrite()**.
- **Điều khiển từ Blynk**: Người dùng có thể thay đổi chế độ và bật/tắt relay từ ứng dụng Blynk, thông qua các hàm **BLYNK_WRITE(V1)** và **BLYNK_WRITE(V2)**.

5.8. Ngắt ngoài và debounce

- **Ngắt ngoài**: Được sử dụng để xử lý việc thay đổi trạng thái nút bấm mà không bị rung, giúp chuyển chế độ và điều khiển relay.
- **Debounce**: Giúp tránh việc kích hoạt nhiều lần khi nút bấm bị rung.

5.9. Vòng lặp chính (loop)

Chạy liên tục và thực hiện các chức năng như:

- Cập nhật màn hình OLED mỗi 100ms.
- Gửi dữ liệu lên ứng dụng Blynk mỗi 1 giây.
- Đọc và hiển thị thông tin về nhiệt độ và trạng thái hệ thống.

KẾT LUẬN

Dự án điều khiển tự động và thủ công dựa trên nhiệt độ sử dụng ESP32 đã đạt được mục tiêu trong việc xây dựng một hệ thống linh hoạt và dễ sử dụng. Hệ thống cho phép người dùng điều khiển relay và thay đổi chế độ hoạt động (AUTO hoặc HAND) thông qua ứng dụng Blynk, đồng thời hiển thị thông tin nhiệt độ và trạng thái của hệ thống lên màn hình OLED. Việc sử dụng cảm biến DHT22 giúp đo nhiệt độ và độ ẩm chính xác, trong khi các nút bấm và relay được tích hợp để điều khiển nhiệt độ theo yêu cầu. Các tính năng như chống rung nút bấm và truyền tải dữ liệu qua MQTT giúp hệ thống hoạt động ổn định và đáng tin cậy.

Hơn nữa, việc kết nối Wi-Fi và sử dụng nền tảng Blynk giúp người dùng có thể theo dõi và điều khiển hệ thống từ xa một cách dễ dàng. Chế độ tự động điều chỉnh relay theo nhiệt độ giúp tối ưu hóa hoạt động của hệ thống, trong khi chế độ thủ công cung cấp tính linh hoạt cho người sử dụng. Tuy nhiên, để cải thiện hệ thống, có thể xem xét bổ sung các tính năng như cảnh báo qua điện thoại khi nhiệt độ vượt ngưỡng hoặc phát triển giao diện người dùng trên ứng dụng Blynk để dễ dàng hơn trong việc điều khiển.

TÀI LIỆU THAM KHẢO

1. Tài liệu học tập, source code tham khảo của thầy Võ Việt Dũng, https://github.com/vvdung-husc/2024-2025.2.TIN4024.004

2. ESP32 Programming for Beginners, https://www.instructables.com/ESP32-Programming-for-Beginners/

3. DHT22 Sensor Tutorial, https://www.electronicwings.com/nodemcu/dht22-humidity-sensor

4. Blynk Documentation, https://docs.blynk.io/

5. Arduino to Blynk Integration, https://blynk.io/blog/how-to-use-blynk-with-arduino/

6. Adafruit SSD1306 OLED Tutorial, https://learn.adafruit.com/adafruit-gfx-graphics-library/using-fonts

7. PubSubClient Library, https://pubsubclient.knolleary.net/

8. WiFi Library for ESP32, https://github.com/espressif/arduino-esp32

9. Using MQTT Protocol with ESP32, https://randomnerdtutorials.com/esp32-mqtt-publish-subscribe-arduino/

10. WiFi ESP32 – Getting Started, https://www.electronicwings.com/nodemcu/esp32-wifi

11. Arduino IDE Setup for ESP32, https://www.arduino.cc/en/Guide/ESP32

12. Introduction to Blynk IoT, https://www.blynk.io/

13. Setting Up MQTT with Blynk, https://docs.blynk.io/en/getting-started

14. ESP32 and MQTT Tutorial, https://www.hackster.io/news/esp32-and-mqtt-a-tutorial-for-beginners-56d58f5440d5

15. DHT22 Sensor Data with Blynk, https://create.arduino.cc/projecthub/ryanchris99/blynk-and-dht22-9e313b

16. Blynk with Wi-Fi Module ESP32, https://www.circuitdigest.com/microcontroller-projects/esp32-blynk-app-based-iot-project

17. Building IoT Applications Using ESP32, https://www.tescaglobal.com/blog/building-iot-applications-using-esp32

18. Arduino Interrupts: A Complete Guide, https://www.electronicwings.com/arduino/interrupts-in-arduino

19. How to Use MQTT with Blynk for IoT Projects, https://www.iotdesignpro.com/projects/mqtt-with-blynk-for-iot-projects

20. Adafruit DHT22 Sensor and Arduino, https://learn.adafruit.com/dht-humidity-sensing-with-arduino/overview

21. ESP32 and MQTT with SSL, https://dev.to/

22. LED Relay Control Using Blynk and ESP32, https://www.circuitdigest.com/microcontroller-projects/led-relay-control-using-blynk-and-esp32

23. How to Control Relay from Blynk App, https://www.electronicwings.com/nodemcu/relay-module-control-using-blynk-app

24. Use MQTT with ESP32 and Blynk, https://www.hackster.io/news/how-to-use-mqtt-with-esp32-and-blynk-application-7c1f2d3d80fc

25. Programming the ESP32, https://www.esp32.com/

26. OLED Display on ESP32 Tutorial, https://www.hackster.io/news/esp32-oled-display-tutorial-25f8fdad1845

27. Blynk IoT SDK Documentation, https://docs.blynk.io/en/blynk-app/

28. Introduction to Relay Control with ESP32, https://www.robotshop.com/community/forum/t/controlling-relay-using-esp32/14590

29. Interrupt Handling in ESP32, https://www.espressif.com/en/products/socs/esp32

30. Handling Button Presses in ESP32, https://www.circuitdigest.com/microcontroller-projects/esp32-buttons-interrupt