TRƯỜNG ĐẠI HỌC KHOA HỌC

KHOA: CÔNG NGHỆ THÔNG TIN

SỐ PHÁCH:

TÊN ĐỀ TÀI TIỂU LUẬN

THIẾT KẾ HỆ THỐNG GIÁM SÁT NHIỆT ĐỘ VÀ ĐỘ ẨM DỰA TRÊN ESP32

MÔN: PHÁT TRIỂN ỨNG DỤNG IOT - NHÓM 5

MÃ HỌC PHẦN: 2024-2025.2.TIN4024.005

GIẢNG VIÊN HƯỚNG DẪN : VÕ VIỆT DŨNG

HUẾ,THÁNG 4 NĂM 2025

MỤC LỤC

MỞ ĐẦU

1. Bối cảnh và tầm quan trọng của dự án

Trong bối cảnh công nghệ Internet of Things (IoT) ngày càng phát triển, việc giám sát và điều khiển các thông số môi trường như nhiệt độ và độ ẩm đã trở thành một nhu cầu thiết yếu trong nhiều lĩnh vực, từ nhà thông minh, nông nghiệp thông minh, đến công nghiệp và y tế. Các hệ thống giám sát môi trường không chỉ giúp con người theo dõi các điều kiện xung quanh một cách tự động và từ xa, mà còn cung cấp khả năng cảnh báo kịp thời khi xảy ra các tình trạng bất thường, từ đó giảm thiểu rủi ro và tối ưu hóa hiệu quả hoạt động. Ví dụ, trong một nhà kính thông minh, việc duy trì nhiệt độ và độ ẩm ở mức lý tưởng là yếu tố then chốt để đảm bảo cây trồng phát triển tốt; trong các phòng máy chủ, việc kiểm soát nhiệt độ giúp bảo vệ thiết bị khỏi hư hỏng do quá nhiệt.

Sự phát triển của các vi điều khiển giá rẻ nhưng mạnh mẽ như ESP32, cùng với các cảm biến kỹ thuật số như DHT22, đã mở ra cơ hội để xây dựng các hệ thống giám sát môi trường hiệu quả với chi phí thấp. ESP32, với khả năng kết nối WiFi và Bluetooth tích hợp, cho phép truyền dữ liệu lên các nền tảng IoT như Blynk để hiển thị theo thời gian thực, đồng thời tích hợp với các ứng dụng nhắn tin như Telegram để gửi cảnh báo tự động. Bên cạnh đó, các công cụ giả lập như Wokwi đã giúp đơn giản hóa quá trình phát triển, cho phép kiểm tra và gỡ lỗi hệ thống mà không cần đầu tư phần cứng thực tế trong giai đoạn đầu.

Dự án này được thực hiện nhằm tận dụng các công nghệ trên để xây dựng một hệ thống giám sát nhiệt độ và độ ẩm môi trường, với mục tiêu không chỉ thu thập và hiển thị dữ liệu mà còn cung cấp khả năng điều khiển cục bộ và cảnh báo từ xa. Hệ thống sử dụng cảm biến DHT22 để đo nhiệt độ và độ ẩm, ESP32 làm trung tâm xử lý, hiển thị thời gian hoạt động trên màn hình TM1637 7-segment, điều khiển LED xanh thông qua nút bấm, gửi dữ liệu lên Blynk để theo dõi từ xa, và gửi cảnh báo qua Telegram khi giá trị vượt

ngưỡng. Việc sử dụng Wokwi để giả lập bảng mạch giúp đảm bảo tính khả thi của hệ thống trước khi triển khai thực tế, đồng thời tiết kiệm chi phí và thời gian phát triển.

2. Mục tiêu của dự án

Dự án được thực hiện với các mục tiêu cụ thể như sau:

Xây dựng hệ thống giám sát môi trường: Thiết kế và triển khai một hệ thống sử dụng cảm biến DHT22 và ESP32 để thu thập dữ liệu nhiệt độ và độ ẩm một cách chính xác và liên tục.

Hiển thị và điều khiển cục bộ: Hiển thị thời gian hoạt động của hệ thống trên màn hình TM1637 7-segment, đồng thời cho phép điều khiển LED xanh thông qua nút bấm, với trạng thái được đồng bộ lên Blynk.

Theo dõi từ xa qua Blynk: Gửi dữ liệu nhiệt độ, độ ẩm, thời gian hoạt động, và trạng thái LED lên nền tảng Blynk để người dùng có thể theo dõi theo thời gian thực từ bất kỳ đâu.

Cảnh báo tự động qua Telegram: Gửi tin nhắn cảnh báo đến người dùng qua Telegram khi nhiệt độ hoặc độ ẩm vượt ngưỡng quy định (nhiệt độ: 18°C–28°C, độ ẩm: 30%–60%), đảm bảo phản ứng kịp thời trước các tình trạng bất thường.

Giả lập và kiểm tra trên Wokwi: Sử dụng Wokwi để mô phỏng toàn bộ hệ thống, từ kết nối phần cứng đến giao tiếp phần mềm, nhằm kiểm tra tính khả thi và gỡ lỗi trước khi triển khai trên phần cứng thực tế.

Đề xuất cải tiến: Đánh giá hiệu suất hệ thống, xác định các vấn đề tiềm ẩn, và đề xuất các giải pháp cải tiến để nâng cao độ tin cậy, hiệu quả, và khả năng mở rộng của hệ thống.

3. Phạm vi của dự án

Dự án tập trung vào việc thiết kế, xây dựng, và thử nghiệm một hệ thống giám sát nhiệt độ và độ ẩm sử dụng các linh kiện và công nghệ sau:

Phần cứng: ESP32 DevKit C V4, cảm biến DHT22, nút bấm, LED xanh, và màn hình TM1637 7-segment.

Phần mềm: Arduino IDE để lập trình ESP32, các thư viện hỗ trợ giao tiếp với DHT22, TM1637, Blynk, và Telegram.

Nền tảng IoT: Blynk để hiển thị dữ liệu từ xa, Telegram để gửi cảnh báo tự động.

Môi trường giả lập: Wokwi để mô phỏng bảng mạch và kiểm tra hệ thống.

Phạm vi của dự án giới hạn ở việc phát triển và thử nghiệm hệ thống trong môi trường giả lập Wokwi, với các chức năng chính đã nêu. Việc triển khai trên phần cứng thực tế và mở rộng hệ thống (như thêm cảm biến hoặc tích hợp với các nền tảng IoT khác) sẽ được đề xuất trong phần cải tiến, nhưng không nằm trong phạm vi chính của dự án này.

4. Cấu trúc của bài viết

Bài viết được tổ chức thành các phần chính để trình bày toàn bộ quá trình thực hiện dự án một cách logic và rõ ràng:

Cơ sở lý luận: Cung cấp thông tin nền tảng về các thành phần chính (DHT22, ESP32, Blynk, Telegram, Wokwi), bao gồm nguyên lý hoạt động, tính năng, ứng dụng, và lý do lựa chọn.

Thiết kế và xây dựng hệ thống: Mô tả chi tiết quá trình thiết kế sơ đồ mạch, lựa chọn linh kiện, xây dựng phần cứng và phần mềm, tích hợp các thành phần, và kiểm tra ban đầu trên Wokwi.

Thử nghiệm và đánh giá: Trình bày phương pháp thử nghiệm, kết quả thu được, đánh giá hiệu suất và độ tin cậy của hệ thống, phân tích các vấn đề gặp phải, và đề xuất các giải pháp cải tiến.

Kết luận và hướng phát triển: Tóm tắt các kết quả đạt được, đánh giá tổng quan về dự án, và đề xuất các hướng phát triển trong tương lai, như triển khai thực tế, mở rộng chức năng, hoặc ứng dụng trong các lĩnh vực cụ thể.

5. Ý nghĩa của dự án

Dự án không chỉ mang ý nghĩa học thuật trong việc ứng dụng các công nghệ IoT hiện đại (ESP32, Blynk, Telegram) để xây dựng một hệ thống giám sát môi trường, mà còn có giá trị thực tiễn cao. Hệ thống có thể được áp dụng trong các lĩnh vực như nhà thông minh (giám sát nhiệt độ và độ ẩm trong nhà), nông nghiệp thông minh (kiểm soát môi trường trong nhà kính), hoặc công nghiệp (bảo vệ thiết bị trong phòng máy chủ). Việc sử dụng Wokwi để giả lập giúp giảm chi phí và thời gian phát triển, đồng thời cung cấp một nền tảng để học tập và nghiên cứu về IoT, lập trình nhúng, và tích hợp phần cứng/phần mềm. Hơn nữa, dự án mở ra tiềm năng mở rộng, như tích hợp thêm cảm biến, lưu trữ dữ liệu dài hạn, hoặc triển khai trên phần cứng thực tế, góp phần vào sự phát triển của các giải pháp IoT giá rẻ và hiệu quả.

I. CƠ SỞ LÝ THUYẾT

1. Vi điều khiển ESP32

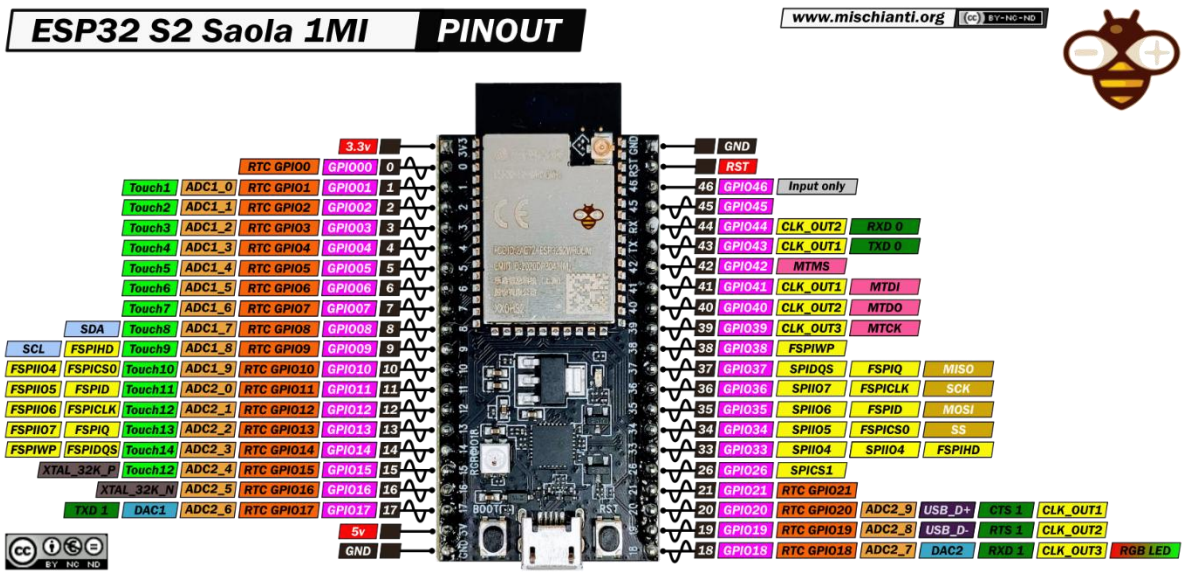
ESP32 là một vi điều khiển mạnh mẽ do Espressif Systems phát triển, được thiết kế dành riêng cho các ứng dụng IoT. Đây là phiên bản nâng cấp của ESP8266, với hiệu năng vượt trội và nhiều tính năng bổ sung. ESP32 được xây dựng trên lõi kép Tensilica Xtensa LX6, có tốc độ xung nhịp lên đến 240 MHz, tích hợp WiFi và Bluetooth Low Energy (BLE), khiến nó trở thành lựa chọn lý tưởng cho các dự án kết nối không dây.

ESP32 có các thành phần RF tích hợp như bộ khuếch đại công suất, bộ khuếch đại nhận tiếng ồn thấp, công tắc ăng-ten, bộ lọc và Balun RF, giúp thiết kế phần cứng dễ dàng với ít linh kiện bên ngoài. Nó được sản xuất bằng công nghệ 40nm công suất thấp của TSMC, phù hợp với các ứng dụng chạy bằng pin như thiết bị đeo, đồng hồ thông minh và thiết bị IoT.

Thông số kỹ thuật chính của ESP32:

- CPU: **Single-core** Tensilica LX7 @ 240 MHz.
- RAM: 320 KB SRAM + 128 KB ROM + 16 KB RTC Fast Memory.
- **Không có Bluetooth!**

- Wi-Fi: 802.11 b/g/n (có, giống ESP32).
- GPIO: 43 GPIO.
- ADC: 20 kênh ADC 13-bit (độ phân giải cao hơn).
- DAC: **Không có DAC**.
- Giao tiếp: 3 x SPI, 2 x I2C, 2 x I2S, 2 x UART.
    - Bảo mật: Tốt hơn với HMAC, Digital Signature, Secure Boot v2, AES-256.



Hình 1.1.1: Minh hoạ vi điều khiển Esp32

Nguồn: https://mischianti.org/esp32-s2-saola-1mi-1m-high-resolution-pinout-and-specs/

Các môi trường lập trình ESP32 hỗ trợ:

- Arduino IDE
- PlatformIO IDE (VS Code)
- Lua
- MicroPython
- Espressif IDF (IoT Development Framework)
- JavaScript

Tính năng nổi bật

- **Lõi kép và hiệu năng cao**: ESP32 có hai lõi xử lý, cho phép thực hiện đa nhiệm hiệu quả. Một lõi có thể xử lý giao tiếp WiFi/Bluetooth, trong khi lõi còn lại xử lý các tác vụ khác như đọc cảm biến hoặc điều khiển thiết bị.
- **Bộ nhớ**: Đi kèm bộ nhớ flash 4MB (tùy phiên bản), đủ để lưu trữ chương trình và dữ liệu tạm thời, hỗ trợ các ứng dụng phức tạp.
- **GPIO phong phú**: ESP32 có 36 chân GPIO, hỗ trợ nhiều giao thức như I2C, SPI, UART, PWM, và ADC, cho phép kết nối với nhiều thiết bị ngoại vi như cảm biến, màn hình, hoặc động cơ.
- **Tiết kiệm năng lượng**: Hỗ trợ các chế độ tiết kiệm năng lượng như Deep Sleep (tiêu thụ chỉ vài µA), rất hữu ích cho các ứng dụng chạy bằng pin, chẳng hạn như cảm biến không dây trong nông nghiệp.
- **Hỗ trợ lập trình**: ESP32 tương thích với nhiều môi trường lập trình như Arduino IDE, PlatformIO, và ESP-IDF, giúp người ang dễ dàng phát triển ứng dụng.

Một trong những bo mạch phổ biến là ESP32 DevKit, sử dụng module ESP-WROOM-32, giúp dễ dàng lập trình và kết nối với các GPIO.

Khả năng kết nối

- **WiFi**: ESP32 hỗ trợ WiFi 802.11 b/g/n, hoạt động ở ang tần 2.4 GHz, với tốc độ truyền dữ liệu lên đến 150 Mbps. Điều này cho phép ESP32 kết nối với mạng internet để gửi dữ liệu lên các nền tảng IoT như Blynk hoặc Thingspeak.
- **Bluetooth**: Tích hợp Bluetooth 4.2 và BLE, cho phép kết nối với các thiết bị di động hoặc cảm biến không dây khác, mở rộng khả năng ứng dụng trong các hệ thống nhà thông minh.
- **Giao tiếp ngoại vi**: ESP32 hỗ trợ giao tiếp với các cảm biến và module qua các giao thức như I2C (cho màn hình OLED), SPI (cho thẻ SD), và UART (cho giao tiếp Serial). Trong dự án này, ESP32 sử dụng giao tiếp một dây để đọc dữ liệu từ DHT22.

Ứng dụng thực tiễn

- **Nhà thông minh**: ESP32 được sử dụng để điều khiển đèn, quạt, hoặc rèm cửa từ xa thông qua WiFi hoặc Bluetooth, đồng thời thu thập dữ liệu từ các cảm biến như nhiệt độ, độ ẩm.
- **Công nghiệp**: Trong các hệ thống giám sát công nghiệp, ESP32 có thể thu thập dữ liệu từ nhiều cảm biến và gửi về server để phân tích, ví dụ như giám sát nhiệt độ trong nhà máy.
- **Giáo dục**: ESP32 là lựa chọn phổ biến trong các dự án học tập nhờ tính linh hoạt và cộng đồng hỗ trợ lớn, giúp sinh viên học về IoT, lập trình nhúng và kết nối không dây.

2. Cảm biến DHT22

Cảm biến DHT22, còn được gọi là AM2302, là một cảm biến kỹ thuật số được thiết kế để đo nhiệt độ và độ ẩm môi trường với độ chính xác cao. Đây là một trong những cảm biến phổ biến trong các dự án IoT và giám sát môi trường nhờ chi phí hợp lý, kích thước nhỏ gọn và khả năng tích hợp dễ dàng với các vi điều khiển như Arduino, ESP32. Cảm biến này được sản xuất bởi công ty Aosong (Trung Quốc) và thường được sử dụng trong các ứng dụng yêu cầu đo lường chính xác các thông số môi trường.

Cấu trúc của cảm biến DHT22:

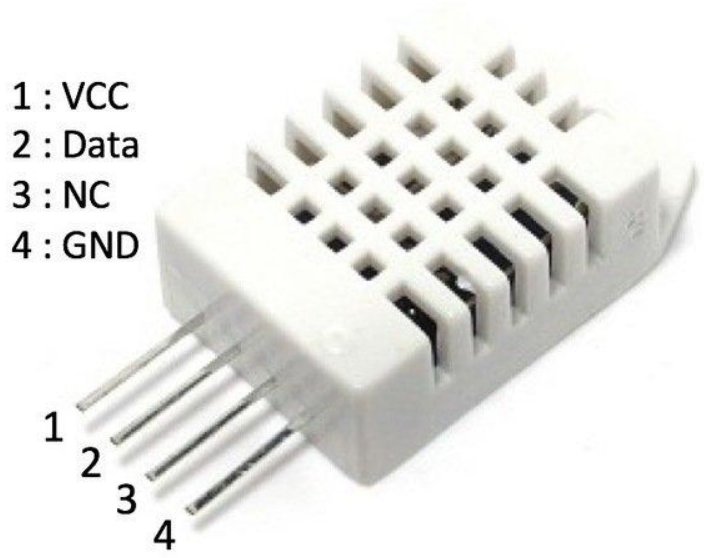
Cảm biến DHT22 bao gồm các thành phần chính sau:

- **Cảm biến nhiệt độ**: Sử dụng một cảm biến nhiệt độ điện trở để đo nhiệt độ môi trường.
- **Cảm biến độ ẩm**: Dựa trên một bộ phận điện dung để đo độ ẩm tương đối trong không khí.
- **Mạch xử lý tín hiệu**: Chuyển đổi dữ liệu đo được thành tín hiệu số để truyền qua giao tiếp một dây.

- **Vỏ bảo vệ**: Giúp bảo vệ linh kiện bên trong khỏi bụi bẩn và tác động môi trường.

Thông số kỹ thuật của DHT22:

- Dải đo nhiệt độ: -40°C đến 80°C
- Sai số nhiệt độ: ±0.5°C
- Dải đo độ ẩm: 0% đến 100% RH
- Sai số độ ẩm: ±2-5%
- Điện áp hoạt động: 3.3V – 5.5V
- Dòng điện tiêu thụ: 1.5mA (khi đo), 40-50µA (chế độ chờ)
- Tốc độ truyền dữ liệu: 1 Hz (1 lần đo mỗi giây)
- Giao tiếp: Một dây (Single-Wire)



Hình 1.2.1: Minh hoạ cảm biến DHT22

Các chân của cảm biến DHT22:

Cảm biến DHT22 có 4 chân kết nối:

- **VCC**: Cấp nguồn (3.3V - 5.5V)
- **GND**: Chân nối đất
- **DATA**: Chân truyền dữ liệu một dây (kết nối với vi điều khiển)
- **NC (Not Connected)**: Không sử dụng

Nguyên lý hoạt động

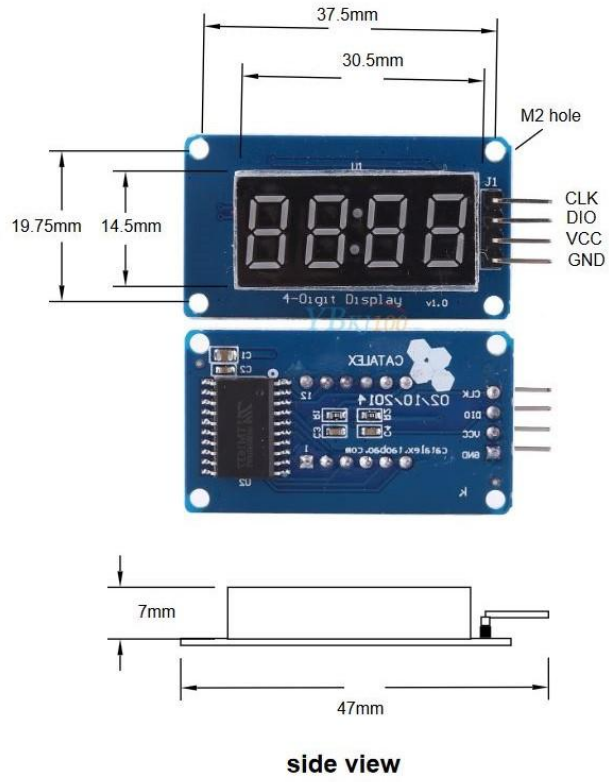
- **Cấu tạo**: DHT22 bao gồm hai thành phần chính:
    - Một cảm biến độ ẩm dựa trên nguyên lý điện dung (capacitive humidity sensor), sử dụng một lớp vật liệu hút ẩm (hygroscopic material) đặt giữa hai điện cực. Khi độ ẩm thay đổi, khả năng hút nước của vật liệu này thay đổi, dẫn đến sự thay đổi điện dung, từ đó tính toán được độ ẩm tương đối.
    - Một thermistor (cảm biến nhiệt độ) để đo nhiệt độ môi trường, hoạt động dựa trên nguyên lý thay đổi điện trở theo nhiệt độ. Thermistor trong DHT22 là loại NTC (Negative Temperature Coefficient), nghĩa là điện trở giảm khi nhiệt độ tăng.
- **Giao tiếp dữ liệu**: DHT22 sử dụng giao thức một dây (single-wire protocol) để truyền dữ liệu. Cảm biến giao tiếp với vi điều khiển bằng cách gửi một chuỗi 40 bit, bao gồm:
    - 16 bit cho độ ẩm (phần nguyên và phần thập phân).
    - 16 bit cho nhiệt độ (phần nguyên và phần thập phân, hỗ trợ cả giá trị âm).
    - 8 bit checksum để kiểm tra lỗi, đảm bảo dữ liệu nhận được là chính xác.
- **Quy trình truyền dữ liệu**:
    - Vi điều khiển (như ESP32) gửi tín hiệu bắt đầu bằng cách kéo chân dữ liệu xuống mức thấp trong khoảng 1ms, sau đó thả ra.

- DHT22 phản hồi bằng cách gửi chuỗi 40 bit, trong đó mỗi bit được biểu diễn bằng thời gian mức cao và mức thấp (mức cao 26-28µs biểu thị bit 0, mức cao 70µs biểu thị bit 1).
- Vi điều khiển đọc và giải mã chuỗi bit này để lấy giá trị nhiệt độ và độ ẩm.

Khi sử dụng DHT22, cần thêm điện trở pull-up (khoảng 4.7kΩ - 10kΩ) giữa chân DATA và VCC để đảm bảo tín hiệu ổn định. DHT22 là một cảm biến đo nhiệt độ và độ ẩm kỹ thuật số với độ chính xác cao. Nó sử dụng giao tiếp một dây (Single-Wire) để truyền dữ liệu, giúp giảm thiểu số lượng dây kết nối với vi điều khiển. Nhờ đặc điểm này, DHT22 thường được sử dụng trong các hệ thống giám sát môi trường, nhà thông minh và các ứng dụng IoT.

3. Màn hình hiển thị TM1637

TM1637 là một module hiển thị LED 7 đoạn phổ biến, thường dùng để hiển thị số liệu như nhiệt độ, thời gian, hoặc các thông tin khác trong hệ thống nhúng. Module này có giao tiếp đơn giản với vi điều khiển thông qua giao thức I2C, chỉ cần hai dây (CLK và DIO) để truyền dữ liệu. Với khả năng hiển thị 4 chữ số và điều chỉnh độ sáng, TM1637 được sử dụng rộng rãi trong các hệ thống đo lường và hiển thị dữ liệu.



Hình 1.3.1: Minh hoạ Màn hình hiển thị TM1637

Cấu trúc của TM1637:

- **Bộ điều khiển TM1637**: IC chính thực hiện điều khiển hiển thị và giao tiếp với vi điều khiển.
- **Màn hình LED 7 đoạn**: Gồm 4 chữ số, mỗi chữ số có 7 đoạn LED giúp hiển thị số liệu.
- **Chân kết nối**: Gồm 4 chân chính để cấp nguồn và giao tiếp dữ liệu.
    - **VCC**: Cấp nguồn (3.3V - 5V).
    - **GND**: Chân nối đất.
    - **CLK (Clock)**: Chân xung nhịp, đồng bộ dữ liệu.
    - **DIO (Data In/Out)**: Chân truyền và nhận dữ liệu.
- **Điện trở và mạch điều khiển độ sáng**: Hỗ trợ điều chỉnh độ sáng màn hình.

Thông số kỹ thuật của TM1637:

- Điện áp hoạt động: 3.3V - 5V
- Dòng tiêu thụ: Khoảng 80mA
- Số chữ số hiển thị: 4 chữ số
- Giao tiếp: I2C với hai dây (CLK và DIO)
- Khả năng điều chỉnh độ sáng: 8 mức sáng khác nhau
- Ứng dụng: Đồng hồ, hiển thị nhiệt độ, bộ đếm, hiển thị thời gian, v.v. TM1637 là một module hiển thị LED 7 đoạn phổ biến, thường dùng để hiển thị số liệu như nhiệt độ, thời gian, hoặc các thông tin khác trong hệ thống nhúng. Module này có giao tiếp đơn giản với vi điều khiển thông qua giao thức I2C, chỉ cần hai dây (CLK và DIO) để truyền dữ liệu. Với khả năng hiển thị 4 chữ số và điều chỉnh độ sáng, TM1637 được sử dụng rộng rãi trong các hệ thống đo lường và hiển thị dữ liệu.

4. Blynk IoT

Tổng quan về Blynk

Blynk là một nền tảng IoT được thiết kế để kết nối các thiết bị phần cứng (như ESP32, Arduino) với ứng dụng di động hoặc web, cho phép người dùng giám sát và điều khiển từ xa. Blynk được phát triển bởi công ty Blynk Inc. (Ukraine) và ra mắt vào năm 2014, nhanh chóng trở thành một trong những nền tảng phổ biến nhờ giao diện thân thiện và khả năng tích hợp dễ dàng.

Hình 1.4.1: Minh hoạ Blynk IOT

Nguyên lý hoạt động

- **Cấu trúc hệ thống**:
    - **Blynk App**: Ứng dụng di động (iOS, Android) hoặc giao diện web, nơi người dùng thiết kế giao diện để hiển thị dữ liệu hoặc điều khiển thiết bị.
    - **Blynk Server**: Đóng vai trò trung gian, xử lý giao tiếp giữa phần cứng và ứng dụng. Blynk cung cấp server đám mây miễn phí, hoặc người dùng có thể tự thiết lập server cục bộ để tăng tính bảo mật.
    - **Phần cứng**: Các thiết bị như ESP32, được lập trình để gửi/nhận dữ liệu từ Blynk Server thông qua giao thức TCP/IP.
- **Chân ảo (Virtual Pins)**: Blynk sử dụng khái niệm "virtual pins" để giao tiếp giữa phần cứng và ứng dụng. Mỗi virtual pin là một kênh dữ liệu, cho phép gửi dữ liệu (như nhiệt độ, độ ẩm) từ phần cứng lên ứng dụng, hoặc nhận lệnh điều khiển từ ứng dụng về phần cứng.
- **Quy trình hoạt động**:
    - Phần cứng (ESP32) gửi dữ liệu lên Blynk Server thông qua WiFi.

- Blynk Server chuyển tiếp dữ liệu đến ứng dụng, hiển thị trên các widget như biểu đồ, đồng hồ đo (gauge), hoặc giá trị số.
- Ngược lại, người dùng có thể gửi lệnh từ ứng dụng (ví dụ: bật/tắt thiết bị), và lệnh này sẽ được truyền qua server đến phần cứng.

Tính năng nổi bật

- **Giao diện tùy chỉnh**: Người dùng có thể kéo thả các widget (như đồng hồ đo, biểu đồ, nút bấm) để tạo giao diện theo ý muốn, không cần kỹ năng lập trình giao diện.
- **Hỗ trợ đa nền tảng**: Blynk hoạt động trên cả iOS, Android và web, đảm bảo tính linh hoạt cho người dùng.
- **Tích hợp dễ dàng**: Blynk cung cấp thư viện cho nhiều vi điều khiển, bao gồm ESP32, giúp việc lập trình trở nên đơn giản.
- **Miễn phí và trả phí**: Phiên bản miễn phí của Blynk đủ để thực hiện các dự án nhỏ, với giới hạn về số lượng widget và dữ liệu. Phiên bản trả phí cung cấp thêm tính năng như lưu trữ dữ liệu dài hạn và hỗ trợ nhiều thiết bị.

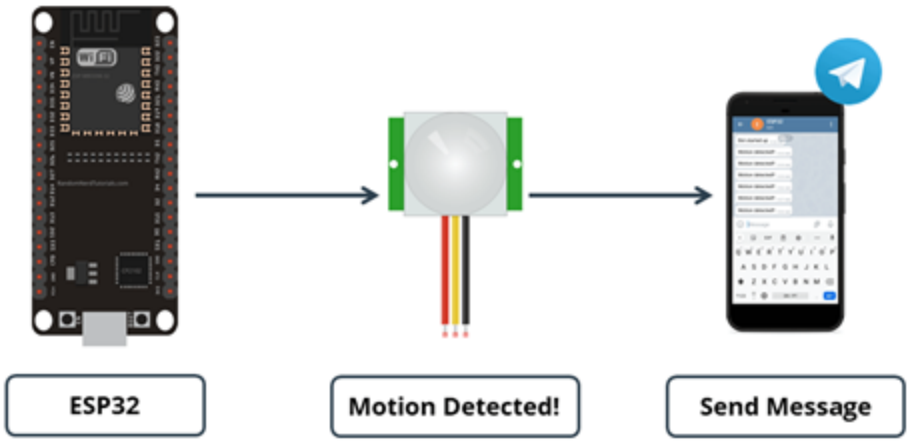
Ứng dụng thực tiễn

- **Giám sát từ xa**: Blynk được sử dụng để theo dõi các thông số môi trường (nhiệt độ, độ ẩm, ánh sáng) trong nhà thông minh, nông nghiệp thông minh, hoặc công nghiệp.
- **Điều khiển thiết bị**: Người dùng có thể bật/tắt đèn, quạt, hoặc máy bơm nước từ xa thông qua ứng dụng Blynk.
- **Giáo dục**: Blynk là công cụ phổ biến trong các dự án học tập về IoT, giúp sinh viên hiểu cách kết nối phần cứng với ứng dụng di động.

5. Telegram: Ứng dụng nhắn tin cho việc gửi cảnh báo

Tổng quan về Telegram

Telegram là một ứng dụng nhắn tin tức thời được phát triển bởi Telegram FZ-LLC, ra mắt vào năm 2013. Ứng dụng này nổi tiếng với tính bảo mật cao, tốc độ nhanh và khả năng tích hợp API cho các ứng dụng tự động. Telegram hỗ trợ cả tin nhắn cá nhân và nhóm, với khả năng gửi văn bản, hình ảnh, video và các tệp khác, khiến nó trở thành một công cụ lý tưởng để gửi cảnh báo trong các dự án IoT.



Hình 1.5.1 Minh hoạ Telegram ứng dụng nhắn tin cho việc gửi cảnh báo

Nguyên lý hoạt động

- **Telegram Bot API**: Telegram cung cấp Bot API, cho phép lập trình viên tạo các bot tự động để gửi và nhận tin nhắn. Một bot được tạo thông qua BotFather (một bot chính thức của Telegram), sau đó nhận được một mã token để xác thực.
- **Gửi tin nhắn**:
    - Bot sử dụng giao thức HTTPS để gửi yêu cầu đến Telegram API, với nội dung tin nhắn và Chat ID (mã định danh của người dùng hoặc nhóm nhận tin nhắn).
    - Telegram Server xử lý yêu cầu và gửi tin nhắn đến đích.
- **Quy trình trong dự án**:

- ESP32 sử dụng thư viện Telegram để gửi tin nhắn khi phát hiện giá trị nhiệt độ hoặc độ ẩm vượt ngưỡng.
- Tin nhắn được gửi đến một nhóm hoặc tài khoản Telegram cụ thể, thông báo cho người dùng về tình trạng bất thường.

Tính năng nổi bật

- **Bảo mật cao**: Telegram sử dụng mã hóa đầu cuối (end-to-end encryption) cho các cuộc trò chuyện bí mật, và mã hóa client-server cho các tin nhắn thông thường, đảm bảo an toàn dữ liệu.
- **Tốc độ nhanh**: Telegram được tối ưu hóa để gửi tin nhắn gần như tức thời, rất phù hợp cho các ứng dụng yêu cầu cảnh báo nhanh.
- **Hỗ trợ bot**: Telegram Bot API cho phép tích hợp dễ dàng với các thiết bị IoT, hỗ trợ gửi tin nhắn tự động mà không cần tương tác thủ công.
- **Miễn phí và đa nền tảng**: Telegram miễn phí, hoạt động trên iOS, Android, Windows, macOS và Linux, đảm bảo người dùng có thể nhận cảnh báo trên bất kỳ thiết bị nào.

Ứng dụng thực tiễn

- **Cảnh báo trong IoT**: Telegram được sử dụng để gửi cảnh báo trong các hệ thống giám sát, chẳng hạn như thông báo nhiệt độ cao trong nhà máy, hoặc độ ẩm thấp trong kho thực phẩm.
- **Tự động hóa**: Các bot Telegram có thể được tích hợp vào hệ thống nhà thông minh để thông báo trạng thái thiết bị (ví dụ: cửa mở, đèn bật).
- **Giáo dục và cộng đồng**: Telegram thường được dùng trong các nhóm học tập hoặc cộng đồng IoT để chia sẻ thông tin và nhận thông báo tự động.

6. Giả lập Wokwi

Tổng quan về Wokwi

Wokwi là một công cụ giả lập trực tuyến miễn phí, được thiết kế để mô phỏng các vi điều khiển, cảm biến và linh kiện điện tử trong môi trường ảo. Wokwi được phát triển bởi Uri Shaked và cộng đồng, ra mắt vào năm 2020, và nhanh chóng trở thành một công cụ phổ biến trong cộng đồng IoT và lập trình nhúng nhờ tính trực quan và dễ sử dụng.

Nguyên lý hoạt động

- **Giao diện kéo thả**: Wokwi cung cấp giao diện web, nơi người dùng có thể kéo thả các linh kiện như ESP32, cảm biến DHT22, LED, hoặc màn hình LCD vào khu vực làm việc.
- **Kết nối mạch**: Người dùng vẽ các kết nối giữa các linh kiện bằng cách kéo dây ảo, tương tự như khi làm việc với phần cứng thực tế.
- **Mô phỏng**:
    - Wokwi mô phỏng hoạt động của vi điều khiển và linh kiện dựa trên mã nguồn được tải lên.
    - Các cảm biến như DHT22 được mô phỏng với giá trị ngẫu nhiên hoặc giá trị do người dùng thiết lập, cho phép kiểm tra logic của chương trình trong các điều kiện khác nhau.
- **Gỡ lỗi**: Wokwi tích hợp Serial Monitor để hiển thị các thông báo từ chương trình, giúp người dùng gỡ lỗi dễ dàng.

Tính năng nổi bật

- **Hỗ trợ nhiều linh kiện**: Wokwi hỗ trợ mô phỏng ESP32, Arduino, Raspberry Pi Pico, cùng nhiều cảm biến và module như DHT22, OLED, servo motor.
- **Miễn phí và trực tuyến**: Wokwi chạy trên trình duyệt, không cần cài đặt phần mềm, rất tiện lợi cho người dùng không có phần cứng thực tế.
- **Tích hợp lập trình**: Người dùng có thể viết mã trực tiếp trên Wokwi, sử dụng các thư viện Arduino phổ biến, và chạy mô phỏng ngay lập tức.

- **Cộng đồng hỗ trợ**: Wokwi có cộng đồng lớn, với nhiều dự án mẫu và tài liệu hướng dẫn, giúp người dùng học hỏi và phát triển ý tưởng.

Ứng dụng thực tiễn

- **Phát triển dự án IoT**: Wokwi được sử dụng để kiểm tra logic của chương trình trước khi triển khai trên phần cứng thực tế, tiết kiệm thời gian và chi phí.
- **Giáo dục**: Wokwi là công cụ lý tưởng cho sinh viên học về lập trình nhúng và IoT, cho phép thực hành mà không cần đầu tư phần cứng.
- **Thử nghiệm ý tưởng**: Các nhà phát triển có thể thử nghiệm các ý tưởng mới, chẳng hạn như kết hợp nhiều cảm biến hoặc module, mà không lo hỏng phần cứng.

II. Thiết kế và xây dựng hệ thống

1. Thiết kế tổng quan hệ thống

Mục tiêu thiết kế

Hệ thống được thiết kế để giám sát nhiệt độ và độ ẩm môi trường bằng cảm biến DHT22, sử dụng ESP32 làm trung tâm xử lý, hiển thị thời gian hoạt động trên màn hình TM1637 7-segment, điều khiển LED xanh thông qua nút bấm, gửi dữ liệu lên nền tảng Blynk để theo dõi từ xa, và gửi cảnh báo qua Telegram khi các giá trị vượt ngưỡng cho phép. Toàn bộ hệ thống được giả lập trên Wokwi để kiểm tra trước khi triển khai trên phần cứng thực tế. Mục tiêu cụ thể bao gồm:

- Thu thập dữ liệu nhiệt độ và độ ẩm chính xác từ cảm biến DHT22, cập nhật liên tục.
- Hiển thị thời gian hoạt động của hệ thống trên màn hình TM1637 7-segment.
- Điều khiển LED xanh (bật/tắt) thông qua nút bấm và đồng bộ trạng thái với Blynk.
- Gửi dữ liệu nhiệt độ và độ ẩm lên ứng dụng Blynk để người dùng theo dõi từ xa.

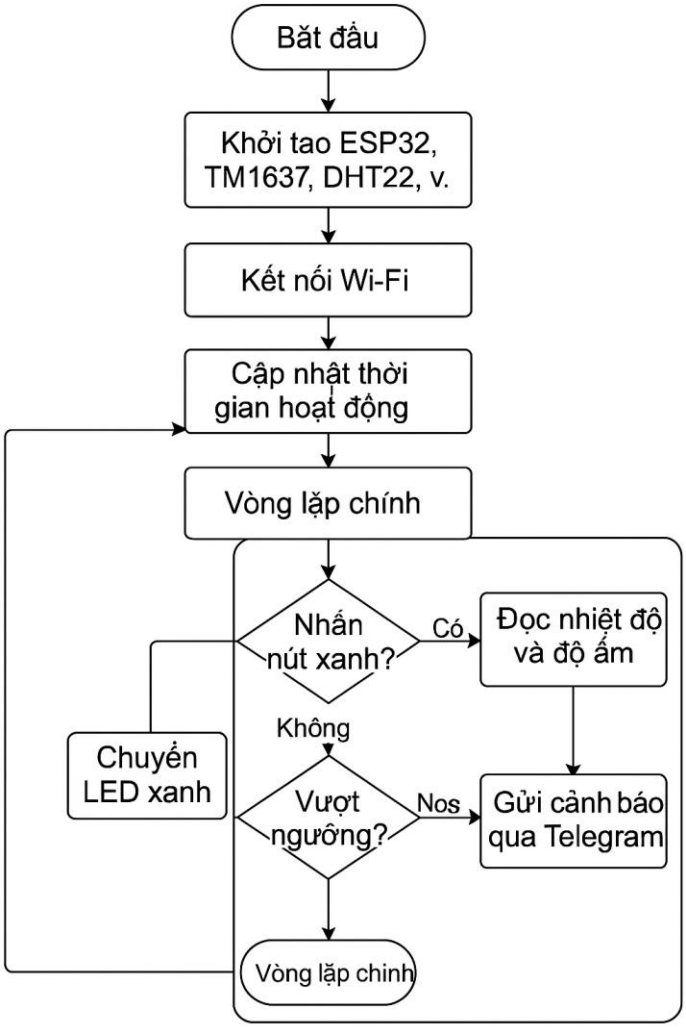
- Gửi cảnh báo qua Telegram khi nhiệt độ hoặc độ ẩm vượt ngưỡng quy định (nhiệt độ: 18°C–28°C, độ ẩm: 30%–60%).
- Đảm bảo hệ thống hoạt động ổn định, dễ mở rộng và tiết kiệm chi phí.

Sơ đồ khối hệ thống

Hệ thống được thiết kế với các khối chức năng chính:

- **Khối cảm biến**: Cảm biến DHT22 thu thập dữ liệu nhiệt độ và độ ẩm từ môi trường.
- **Khối điều khiển và hiển thị cục bộ**: Nút bấm điều khiển LED xanh, màn hình TM1637 hiển thị thời gian hoạt động.
- **Khối xử lý trung tâm**: ESP32 đọc dữ liệu từ cảm biến, điều khiển LED, hiển thị trên màn hình, và gửi dữ liệu qua WiFi.
- **Khối hiển thị dữ liệu từ xa**: Nền tảng Blynk nhận dữ liệu từ ESP32 và hiển thị trên ứng dụng di động hoặc web.
- **Khối cảnh báo**: Telegram Bot gửi tin nhắn cảnh báo đến người dùng khi giá trị vượt ngưỡng.
- **Khối giả lập**: Wokwi mô phỏng toàn bộ hệ thống, bao gồm ESP32, DHT22, nút bấm, LED, và màn hình TM1637.

Sơ đồ khối có thể được hình dung như sau:



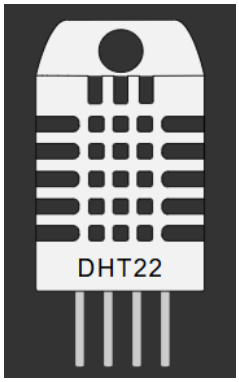
- DHT22 → ESP32 → WiFi → Blynk (hiển thị nhiệt độ, độ ẩm, trạng thái LED).
- Nút bấm → ESP32 → LED (điều khiển bật/tắt) → Blynk (đồng bộ trạng thái).
- ESP32 → TM1637 (hiển thị thời gian hoạt động).
- ESP32 → WiFi → Telegram (gửi cảnh báo).
- Wokwi mô phỏng toàn bộ quá trình từ cảm biến đến kết nối mạng.

Yêu cầu hệ thống

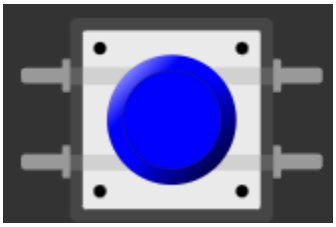
- **Phần cứng**:

- ESP32 DevKit C V4: Đảm bảo khả năng kết nối WiFi và xử lý dữ liệu.

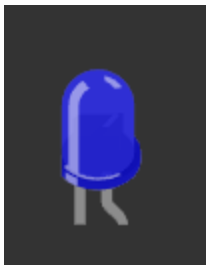
- Cảm biến DHT22: Đo nhiệt độ và độ ẩm.



- Nút bấm: Điều khiển LED xanh.



- LED xanh: Hiển thị trạng thái bật/tắt.

- 
    - Màn hình TM1637 7-segment: Hiển thị thời gian hoạt động.

- **Phần mềm**:
    - Arduino IDE: Môi trường lập trình cho ESP32.
    - Thư viện: Hỗ trợ giao tiếp với DHT22, TM1637, Blynk, và Telegram.
    - Blynk: Hiển thị dữ liệu nhiệt độ, độ ẩm, và trạng thái LED.
    - Telegram Bot: Gửi cảnh báo tự động.
- **Môi trường giả lập**: Wokwi để mô phỏng mạch và kiểm tra logic.

2. Lựa chọn linh kiện và thiết kế sơ đồ mạch

Lựa chọn linh kiện

Dựa trên sơ đồ Wokwi, các linh kiện được sử dụng bao gồm:

- **ESP32 DevKit C V4**: Được chọn vì tính năng vượt trội, bao gồm lõi kép, WiFi tích hợp, và số lượng GPIO phong phú (36 chân). ESP32 có tốc độ xử lý cao (lên đến 240 MHz) và bộ nhớ flash 4MB, đủ để xử lý dữ liệu từ cảm biến, điều khiển

LED, hiển thị trên màn hình, và giao tiếp với Blynk/Telegram. Vị trí trong Wokwi: (top: -9.6, left: -225.56).

- **Cảm biến DHT22**: Được chọn nhờ độ chính xác cao (sai số nhiệt độ ±0.5°C, độ ẩm ±2% RH), phạm vi đo rộng (nhiệt độ: -40°C đến 80°C, độ ẩm: 0% đến 100% RH), và khả năng hoạt động ở mức điện áp 3.3V, tương thích với ESP32. Vị trí trong Wokwi: (top: 105.9, left: -24.6).
- **Nút bấm (Pushbutton)**: Màu xanh, dùng để điều khiển bật/tắt LED. Nút bấm được thiết kế với chế độ kéo lên (pull-up) để đảm bảo tín hiệu ổn định. Vị trí trong Wokwi: (top: -3.4, left: -76.8).
- **LED xanh**: Dùng để hiển thị trạng thái bật/tắt, được điều khiển bởi nút bấm và đồng bộ với Blynk. LED được lật ngược (flip: 1) để phù hợp với bố cục mạch. Vị trí trong Wokwi: (top: 25.2, left: -101.4).
- **Màn hình TM1637 7-segment**: Màu đỏ, dùng để hiển thị thời gian hoạt động của hệ thống (tính bằng giây). TM1637 là module hiển thị 4 chữ số, giao tiếp với ESP32 qua giao thức I2C tương tự, với độ sáng có thể điều chỉnh. Vị trí trong Wokwi: (top: -105.64, left: -175.37).

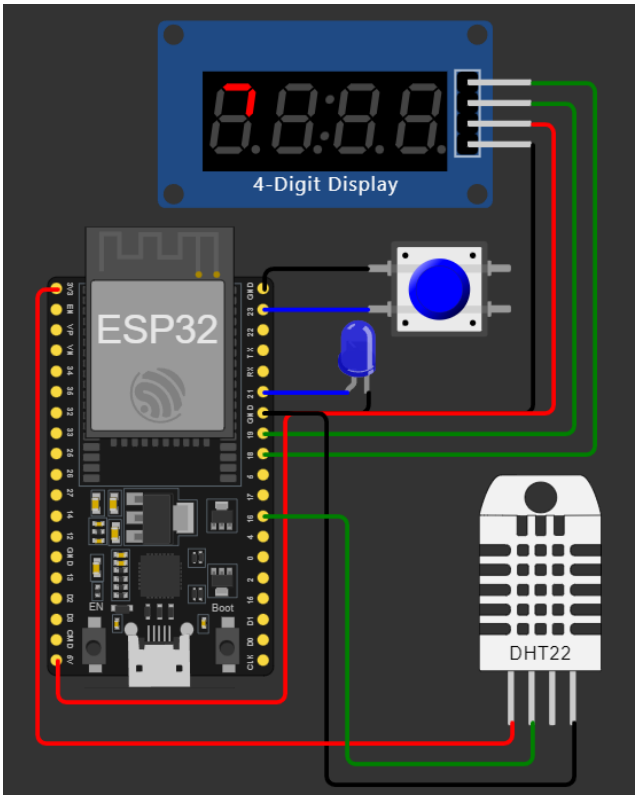
Thiết kế sơ đồ mạch

Dựa trên phần "connections" trong sơ đồ Wokwi, các kết nối được thiết kế như sau:

- **ESP32 và Serial Monitor**:
    - Chân TX của ESP32 kết nối với RX của Serial Monitor.
    - Chân RX của ESP32 kết nối với TX của Serial Monitor.
    - Mục đích: Cho phép ESP32 gửi thông báo gỡ lỗi (như giá trị nhiệt độ, độ ẩm) đến Serial Monitor để theo dõi.
- **Cảm biến DHT22**:

    - Chân GND của DHT22 kết nối với chân GND.3 của ESP32 (dây màu đen).
    - Chân VCC của DHT22 kết nối với chân 3V3 của ESP32 (dây màu đỏ), cung cấp nguồn 3.3V.
    - Chân SDA (Data) của DHT22 kết nối với chân GPIO16 của ESP32 (dây màu xanh lá).
    - Lưu ý: Mặc dù sơ đồ không hiển thị điện trở kéo lên, nhưng trong thực tế, một điện trở 4.7kΩ nên được nối giữa VCC và SDA để đảm bảo tín hiệu ổn định.
- **Nút bấm**:
    - Chân 1 (bên trái) của nút bấm kết nối với chân GND.2 của ESP32 (dây màu đen).
    - Chân 2 (bên trái) của nút bấm kết nối với chân GPIO23 của ESP32 (dây màu xanh dương).
    - Nút bấm được cấu hình với chế độ kéo lên bên trong (INPUT_PULLUP) để tránh tín hiệu nhiễu.
- **LED xanh**:
    - Chân Cathode (C) của LED kết nối với chân GND.3 của ESP32 (dây màu đen).
    - Chân Anode (A) của LED kết nối với chân GPIO21 của ESP32 (dây màu xanh dương).
    - LED được điều khiển bởi GPIO21, với mức cao (HIGH) để bật và mức thấp (LOW) để tắt.
- **Màn hình TM1637 7-segment**:

- Chân GND của TM1637 kết nối với chân GND.3 của ESP32 (dây màu đen).
- Chân VCC của TM1637 kết nối với chân 5V của ESP32 (dây màu đỏ), cung cấp nguồn 5V.
- Chân CLK của TM1637 kết nối với chân GPIO18 của ESP32 (dây màu xanh lá).
- Chân DIO của TM1637 kết nối với chân GPIO19 của ESP32 (dây màu xanh lá).
- TM1637 sử dụng giao thức I2C tương tự để giao tiếp với ESP32, hiển thị thời gian hoạt động.



Hình 2.2.1: Sơ đồ mạch hoàn chỉnh

3. Xây dựng hệ thống phần cứng

Quy trình lắp ráp trên Wokwi

- **Thêm linh kiện**:
    - Đặt ESP32 DevKit C V4 tại vị trí (top: -9.6, left: -225.56).
    - Đặt cảm biến DHT22 tại vị trí (top: 105.9, left: -24.6).
    - Đặt nút bấm màu xanh tại vị trí (top: -3.4, left: -76.8).
    - Đặt LED xanh (lật ngược) tại vị trí (top: 25.2, left: -101.4).
    - Đặt màn hình TM1637 7-segment màu đỏ tại vị trí (top: -105.64, left: -175.37).
- **Kết nối linh kiện**:
    - Kết nối DHT22, nút bấm, LED, và TM1637 với ESP32 theo sơ đồ đã thiết kế, sử dụng các dây màu để phân biệt.
    - Kết nối TX/RX của ESP32 với Serial Monitor để theo dõi thông báo gỡ lỗi.
- **Kiểm tra kết nối**:
    - Xem lại sơ đồ trên Wokwi để đảm bảo các dây được kết nối đúng chân: GPIO16 cho DHT22, GPIO23 cho nút bấm, GPIO21 cho LED, GPIO18/19 cho TM1637.
    - Kiểm tra xem các chân GND và VCC có được kết nối chính xác hay không, đảm bảo không có dây bị lỏng hoặc đấu sai.

Tối ưu hóa phần cứng

- **Giảm nhiễu tín hiệu**: Đối với DHT22, mặc dù sơ đồ không hiển thị điện trở kéo lên, nhưng trong thực tế, nên thêm điện trở 4.7kΩ giữa VCC và SDA để đảm bảo tín hiệu ổn định, đặc biệt trong môi trường có nhiễu điện từ.

- **Tương thích điện áp**: Sử dụng mức 3.3V cho DHT22 và 5V cho TM1637, tận dụng các chân nguồn phù hợp trên ESP32 để tránh chênh lệch điện áp.
- **Dễ mở rộng**: Thiết kế mạch trên Wokwi cho phép dễ dàng thêm các linh kiện khác trong tương lai, chẳng hạn như cảm biến ánh sáng hoặc buzzer để cảnh báo cục bộ.

Triển khai thực tế (nếu áp dụng)

- Nếu triển khai trên phần cứng thực tế, sử dụng breadboard để lắp ráp các linh kiện theo sơ đồ Wokwi.
- Sử dụng dây jumper với màu sắc tương ứng (đen, đỏ, xanh lá, xanh dương) để kết nối, đảm bảo tính trực quan.
- Kiểm tra điện áp tại các chân VCC (3.3V cho DHT22, 5V cho TM1637) bằng đồng hồ vạn năng để xác nhận nguồn ổn định.

4. Xây dựng hệ thống phần mềm

Môi trường lập trình

- **Arduino IDE**: Được chọn làm môi trường lập trình chính nhờ tính đơn giản và cộng đồng hỗ trợ lớn. Arduino IDE hỗ trợ ESP32 thông qua việc cài đặt ESP32 Core từ Board Manager.
- **Thư viện sử dụng**:
    - Thư viện cho DHT22: Hỗ trợ đọc dữ liệu nhiệt độ và độ ẩm.
    - Thư viện cho TM1637: Hỗ trợ hiển thị thời gian trên màn hình 7-segment.
    - Thư viện WiFi: Cho phép ESP32 kết nối với mạng WiFi.
    - Thư viện Blynk: Tích hợp ESP32 với nền tảng Blynk để gửi dữ liệu.
    - Thư viện Telegram: Gửi tin nhắn cảnh báo qua Telegram Bot API.

Quy trình lập trình

- **Khởi tạo hệ thống**:
    - Khai báo các chân kết nối: GPIO16 cho DHT22, GPIO23 cho nút bấm, GPIO21 cho LED, GPIO18 (CLK) và GPIO19 (DIO) cho TM1637.
    - Khai báo thông tin WiFi (SSID: "Wokwi-GUEST", mật khẩu: trống), thông tin Blynk (Template ID, Auth Token), và thông tin Telegram (Bot Token, Chat ID).
    - Trong phần khởi động, thiết lập kết nối Serial để gỡ lỗi, khởi tạo cảm biến DHT22, màn hình TM1637, kết nối WiFi, và thiết lập kết nối với Blynk.
- **Đọc và xử lý dữ liệu từ DHT22**:
    - Tạo một hàm để đọc dữ liệu từ DHT22 mỗi 2 giây, kiểm tra xem dữ liệu có hợp lệ hay không, sau đó in ra Serial Monitor với định dạng: "Nhiệt độ: X°C, Độ ẩm: Y%".
    - Gửi dữ liệu nhiệt độ và độ ẩm lên Blynk thông qua các chân ảo: nhiệt độ gửi lên chân V2, độ ẩm gửi lên chân V3.
- **Điều khiển LED và nút bấm**:
    - Tạo một hàm để đọc trạng thái nút bấm từ GPIO23, sử dụng cơ chế chống dội (debounce) với thời gian chờ 50ms để tránh tín hiệu nhiễu.
    - Khi nút bấm được nhấn (chuyển từ mức thấp sang mức cao), đảo trạng thái LED (bật/tắt), điều khiển LED qua GPIO21, và đồng bộ trạng thái với Blynk qua chân V1.
- **Hiển thị thời gian trên TM1637**:
    - Tạo một hàm để tính thời gian hoạt động của hệ thống (tính bằng giây) và hiển thị trên màn hình TM1637 mỗi 1 giây.

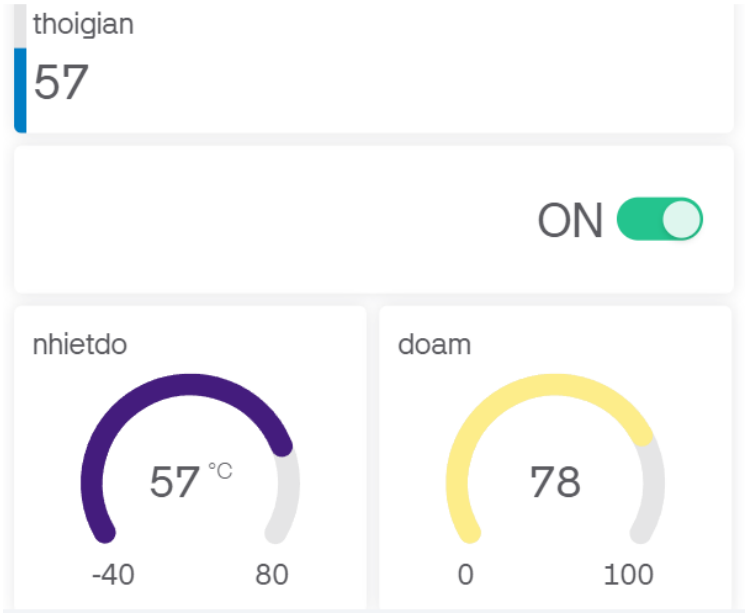
    - Gửi thời gian hoạt động lên Blynk qua chân V0 để người dùng theo dõi từ xa.
    - Khi LED tắt, xóa màn hình TM1637 để tiết kiệm năng lượng.
- **Giám sát ngưỡng và gửi cảnh báo**:
    - Đặt ngưỡng cho nhiệt độ (18°C–28°C) và độ ẩm (30%–60%).
    - Nếu giá trị vượt ngưỡng và đã qua 1 phút kể từ lần gửi cảnh báo cuối, tạo một tin nhắn với nội dung: "Cảnh báo: Giá trị bất thường! Nhiệt độ: X°C (ngưỡng 18°C-28°C), Độ ẩm: Y% (ngưỡng 30%-60%)", sau đó gửi qua Telegram.
- **Tối ưu hóa phần mềm**:
    - Sử dụng cơ chế kiểm tra thời gian không chặn để đảm bảo ESP32 có thể thực hiện nhiều tác vụ cùng lúc: đọc cảm biến, điều khiển LED, hiển thị trên TM1637, gửi dữ liệu lên Blynk, và gửi cảnh báo qua Telegram.
    - Thêm thông báo gỡ lỗi trên Serial Monitor để phát hiện lỗi, ví dụ: thông báo khi không đọc được dữ liệu từ cảm biến hoặc khi gửi tin nhắn Telegram thất bại.

Cấu hình Blynk

- **Tạo dự án trên Blynk**:
    - Truy cập ứng dụng Blynk, tạo một dự án mới với tên "blynk", nhận Template ID và Auth Token.
    - Thêm các widget hiển thị:
        - Widget "Thời gian" (V0): Hiển thị thời gian hoạt động (giây).
        - Widget "Nhiệt độ" (V2): Hiển thị nhiệt độ (°C).
        - Widget "Độ ẩm" (V3): Hiển thị độ ẩm (%).

        - Widget "Nút bấm" (V1): Hiển thị và điều khiển trạng thái LED (bật/tắt).

- **Tùy chỉnh giao diện**:
    - Đặt tên widget bằng tiếng Việt để thân thiện với người dùng: "Thời gian", "Nhiệt độ", "Độ ẩm", "Nút bấm".
    - Sử dụng màu sắc khác nhau để phân biệt: đỏ cho nhiệt độ, xanh dương cho độ ẩm, xám cho thời gian, nâu cho nút bấm.
    - Thiết lập phạm vi hiển thị: 0-50°C cho nhiệt độ, 0-100% cho độ ẩm.

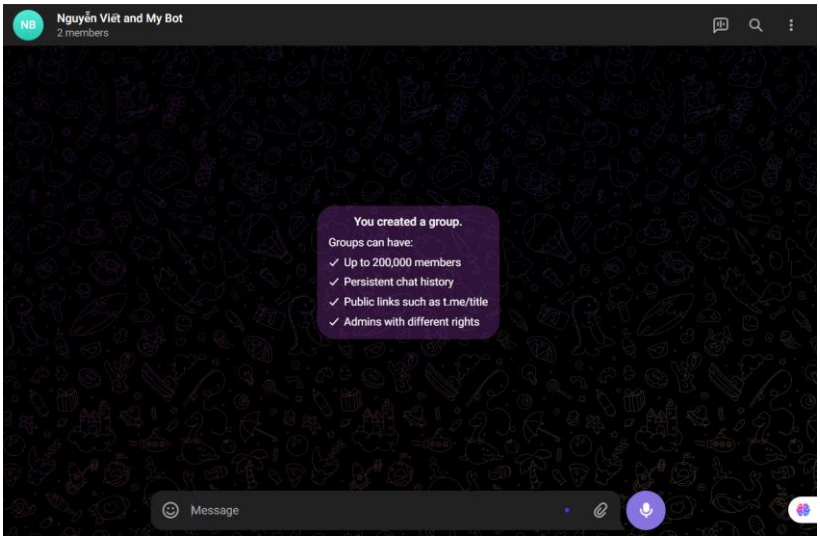


Hình 2.4.1: Giao diện blynk

Cấu hình Telegram

- **Tạo bot trên Telegram**:
    - Sử dụng BotFather để tạo bot mới, nhận Bot Token.
    - Tạo một nhóm Telegram, thêm bot vào nhóm, và lấy Chat ID của nhóm.

- **Tích hợp với ESP32**:
    - Sử dụng Bot Token và Chat ID để gửi tin nhắn tự động khi giá trị vượt ngưỡng.
    - Định dạng tin nhắn rõ ràng, bao gồm giá trị hiện tại và ngưỡng quy định, để người dùng dễ hiểu và hành động.



Hình 2.4.2: Bot telegram

5. Tích hợp và kiểm tra hệ thống

Tích hợp trên Wokwi

- **Mô phỏng mạch**:
    - Sử dụng sơ đồ đã thiết kế trên Wokwi, với các linh kiện được đặt đúng vị trí và kết nối chính xác.
    - Tải chương trình lên Wokwi và chạy mô phỏng.
- **Kiểm tra chức năng**:

- **Đọc dữ liệu từ DHT22**: Xác minh rằng ESP32 đọc được dữ liệu nhiệt độ và độ ẩm từ cảm biến, hiển thị trên Serial Monitor với định dạng: "Nhiệt độ: X°C, Độ ẩm: Y%".
- **Điều khiển LED và nút bấm**: Nhấn nút bấm trên Wokwi, kiểm tra xem LED có bật/tắt đúng hay không, và trạng thái có được đồng bộ lên Blynk (chân V1) hay không.
- **Hiển thị thời gian trên TM1637**: Kiểm tra xem màn hình TM1637 có hiển thị thời gian hoạt động (giây) chính xác hay không, và thời gian có được gửi lên Blynk (chân V0) hay không.
- **Gửi dữ liệu lên Blynk**: Xác minh rằng dữ liệu nhiệt độ (V2) và độ ẩm (V3) được hiển thị đúng trên ứng dụng Blynk, với giá trị cập nhật mỗi 2 giây.
- **Gửi cảnh báo qua Telegram**: Thay đổi giá trị giả lập trên Wokwi (ví dụ: nhiệt độ lên 30°C), xác minh rằng tin nhắn cảnh báo được gửi đến nhóm Telegram với nội dung chính xác.

- **Gỡ lỗi**:
  - Sử dụng Serial Monitor trên Wokwi để xem các thông báo gỡ lỗi, chẳng hạn như lỗi đọc cảm biến, trạng thái LED, hoặc lỗi gửi tin nhắn Telegram.
  - Điều chỉnh chương trình nếu cần, ví dụ: tăng thời gian chờ giữa các lần gửi cảnh báo để tránh gửi quá nhiều tin nhắn.

Kiểm tra trên phần cứng thực tế (nếu triển khai)

- **Lắp ráp phần cứng**:
  - Sử dụng breadboard để lắp ráp các linh kiện: ESP32, DHT22, nút bấm, LED, và TM1637, theo sơ đồ Wokwi.
  - Sử dụng dây jumper với màu sắc tương ứng (đen, đỏ, xanh lá, xanh dương) để kết nối.

- **Tải chương trình**:
    - Sử dụng Arduino IDE để tải chương trình lên ESP32.
- **Kiểm tra thực tế**:
    - Đặt hệ thống trong các điều kiện môi trường khác nhau (ví dụ: gần máy sưởi để tăng nhiệt độ, hoặc trong phòng ẩm để tăng độ ẩm), kiểm tra xem hệ thống có phát hiện và gửi cảnh báo đúng hay không.
    - Nhấn nút bấm để kiểm tra trạng thái LED, xem màn hình TM1637 có hiển thị thời gian chính xác hay không.
    - Xác minh rằng dữ liệu trên Blynk khớp với giá trị thực tế đo được bằng thiết bị tham chiếu (như nhiệt kế hoặc ẩm kế).

Tối ưu hóa hệ thống

- **Hiệu suất**: Đảm bảo hệ thống không bị treo hoặc chậm khi chạy liên tục, bằng cách sử dụng cơ chế không chặn trong lập trình.
- **Độ tin cậy**: Kiểm tra kết nối WiFi và xử lý trường hợp mất kết nối (ví dụ: tự động kết nối lại khi WiFi bị gián đoạn).

Tiết kiệm năng lượng: Nếu triển khai thực tế, có thể thêm chế độ Deep Sleep cho ESP32 để tiết kiệm năng lượng, chỉ thức dậy mỗi 2 giây để đọc cảm biến và gửi dữ liệu.

III. Thử nghiệm và đánh giá

1. Mục tiêu thử nghiệm

Quá trình thử nghiệm được thực hiện nhằm đánh giá hiệu quả hoạt động của hệ thống đã thiết kế, bao gồm các chức năng chính: thu thập dữ liệu nhiệt độ và độ ẩm từ cảm biến DHT22, hiển thị thời gian hoạt động trên màn hình TM1637 7-segment, điều khiển LED xanh thông qua nút bấm, gửi dữ liệu lên nền tảng Blynk, và gửi cảnh báo qua Telegram khi giá trị vượt ngưỡng. Cụ thể, các mục tiêu thử nghiệm bao gồm:

- Xác minh rằng hệ thống có thể đọc và xử lý dữ liệu từ cảm biến DHT22 một cách chính xác.
- Kiểm tra khả năng hiển thị thời gian hoạt động trên màn hình TM1637 và đồng bộ với Blynk.
- Đánh giá tính năng điều khiển LED xanh thông qua nút bấm và đồng bộ trạng thái với Blynk.
- Đảm bảo dữ liệu nhiệt độ và độ ẩm được gửi lên Blynk đúng cách và hiển thị theo thời gian thực.
- Kiểm tra cơ chế gửi cảnh báo qua Telegram khi nhiệt độ hoặc độ ẩm vượt ngưỡng quy định (nhiệt độ: 18°C–28°C, độ ẩm: 30%–60%).
- Đánh giá hiệu suất, độ tin cậy của hệ thống trong môi trường giả lập Wokwi, và xác định các vấn đề tiềm ẩn để cải tiến.

2. Phương pháp thử nghiệm

Môi trường thử nghiệm

- **Môi trường giả lập**: Sử dụng Wokwi để mô phỏng toàn bộ hệ thống, bao gồm ESP32 DevKit C V4, cảm biến DHT22, nút bấm, LED xanh, và màn hình TM1637 7-segment, theo sơ đồ đã thiết kế.
- **Công cụ hỗ trợ**:
    - Serial Monitor trên Wokwi: Theo dõi các thông báo gỡ lỗi, như giá trị nhiệt độ, độ ẩm, trạng thái LED, và trạng thái gửi tin nhắn Telegram.
    - Ứng dụng Blynk: Kiểm tra dữ liệu hiển thị trên các widget (thời gian, nhiệt độ, độ ẩm, trạng thái LED).
    - Ứng dụng Telegram: Kiểm tra tin nhắn cảnh báo được gửi đến nhóm Telegram.

- **Điều kiện thử nghiệm**:
    - Giả lập các giá trị nhiệt độ và độ ẩm khác nhau trên Wokwi (do Wokwi tự động tạo giá trị ngẫu nhiên cho DHT22).
    - Thay đổi trạng thái nút bấm để kiểm tra điều khiển LED.
    - Kiểm tra hệ thống trong thời gian dài (mô phỏng 30 phút) để đánh giá độ ổn định.

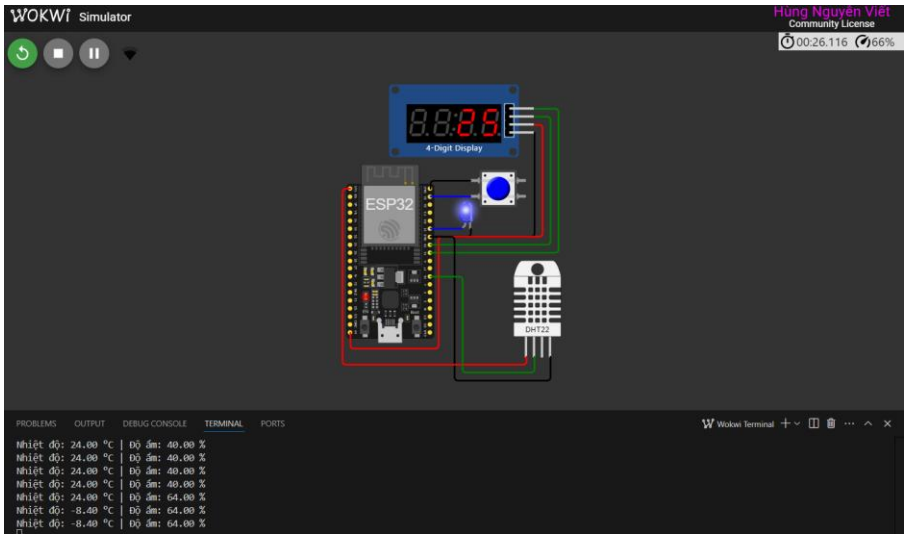
I. Kịch bản thử nghiệm

- **Kịch bản 1: Đọc dữ liệu từ DHT22**:
    - Chạy hệ thống và quan sát giá trị nhiệt độ và độ ẩm được in trên Serial Monitor.
    - So sánh giá trị hiển thị trên Serial Monitor với giá trị trên Blynk (chân V2 cho nhiệt độ, V3 cho độ ẩm).
- **Kịch bản 2: Điều khiển LED và nút bấm**:
    - Nhấn nút bấm trên Wokwi, quan sát trạng thái LED (bật/tắt) và kiểm tra xem trạng thái có được đồng bộ lên Blynk (chân V1) hay không.
    - Thay đổi trạng thái LED từ ứng dụng Blynk, kiểm tra xem LED trên Wokwi có thay đổi tương ứng hay không.
- **Kịch bản 3: Hiển thị thời gian trên TM1637**:
    - Quan sát màn hình TM1637, kiểm tra xem thời gian hoạt động (tính bằng giây) có tăng liên tục hay không.
    - So sánh giá trị trên TM1637 với giá trị hiển thị trên Blynk (chân V0).
    - Tắt LED bằng nút bấm, kiểm tra xem màn hình TM1637 có được xóa hay không.

- **Kịch bản 4: Gửi dữ liệu lên Blynk**:
    - Kiểm tra các widget trên Blynk (thời gian, nhiệt độ, độ ẩm, trạng thái LED) để đảm bảo dữ liệu được cập nhật mỗi 1-2 giây.
    - Quan sát giao diện Blynk, xác minh rằng các giá trị hiển thị đúng đơn vị (°C cho nhiệt độ, % cho độ ẩm).
- **Kịch bản 5: Gửi cảnh báo qua Telegram**:
    - Thay đổi giá trị nhiệt độ và độ ẩm trên Wokwi để vượt ngưỡng (ví dụ: nhiệt độ lên 30°C, độ ẩm xuống 25%).
    - Kiểm tra xem tin nhắn cảnh báo có được gửi đến nhóm Telegram hay không, và nội dung tin nhắn có đúng định dạng hay không.
    - Đợi 1 phút và lặp lại điều kiện vượt ngưỡng, kiểm tra xem hệ thống có gửi tin nhắn tiếp theo đúng thời gian hay không.

3. Kết quả thử nghiệm

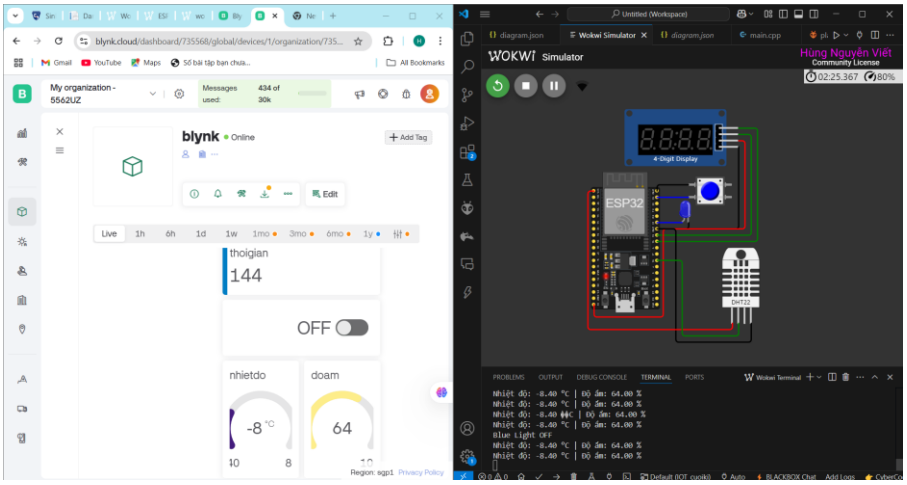
Kết quả kịch bản 1: Đọc dữ liệu từ DHT22



Hình 3.5.1: Mô phỏng đọc dữ liệu từ DHT22

- **Quan sát**: Hệ thống đọc dữ liệu từ DHT22 mỗi 2 giây và in ra Serial Monitor với định dạng: "Nhiệt độ: X°C, Độ ẩm: Y%". Ví dụ: "Nhiệt độ: 24.00°C, Độ ẩm: 40.00%".
- **So sánh với Blynk**: Giá trị trên Serial Monitor khớp với giá trị hiển thị trên Blynk (chân V2 và V3), với độ trễ không đáng kể (dưới 1 giây).
- **Nhận xét**: Chức năng đọc dữ liệu từ DHT22 hoạt động chính xác, dữ liệu được truyền từ ESP32 lên Blynk một cách ổn định.

Kết quả kịch bản 2: Điều khiển LED và nút bấm



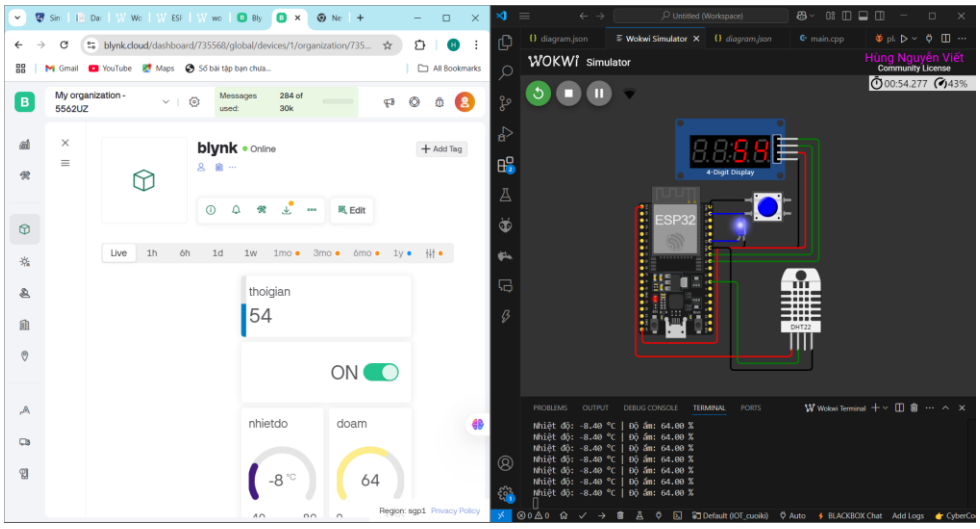
Hình 3.5.2: Mô phỏng điều khiển led và nút bấm

- **Nhấn nút bấm trên Wokwi**:
    - Khi nhấn nút, LED xanh chuyển trạng thái (từ tắt sang bật hoặc ngược lại). Serial Monitor hiển thị thông báo: "Blue Light ON" hoặc "Blue Light OFF".
    - Trạng thái LED được đồng bộ lên Blynk (chân V1), widget "Nút bấm" trên Blynk thay đổi tương ứng (1: bật, 0: tắt).
- **Điều khiển từ Blynk**:

- Thay đổi trạng thái trên widget "Nút bấm" (V1) từ ứng dụng Blynk, LED trên Wokwi thay đổi tương ứng, và Serial Monitor hiển thị thông báo: "Blynk -> Blue Light ON" hoặc "Blynk -> Blue Light OFF".

- **Nhận xét**: Chức năng điều khiển LED hoạt động tốt cả cục bộ (qua nút bấm) và từ xa (qua Blynk). Cơ chế chống dội (debounce) đảm bảo tín hiệu nút bấm ổn định, không có hiện tượng nhiễu.

Kết quả kịch bản 3: Hiển thị thời gian trên TM1637

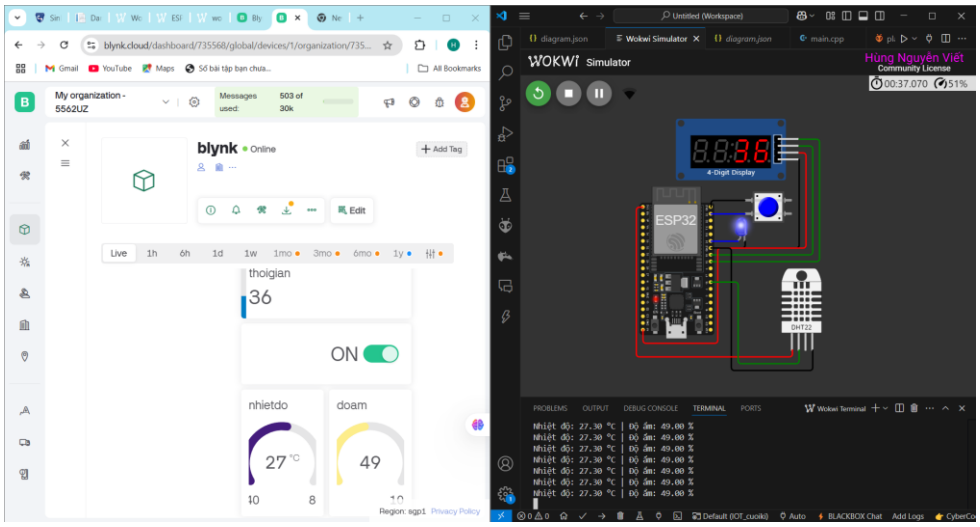


Hình 3.5.3: Mô phỏng Hiển thị thời gian trên TM1637

- **Hiển thị thời gian**:
  - Màn hình TM1637 hiển thị thời gian hoạt động (tính bằng giây) tăng liên tục, ví dụ: "0000", "0001", "0002",...
  - Giá trị trên TM1637 khớp với giá trị hiển thị trên Blynk (chân V0), với độ trễ không đáng kể.
- **Tắt LED**:
  - Khi LED được tắt (bằng nút bấm hoặc từ Blynk), màn hình TM1637 được xóa, không hiển thị số, đúng như thiết kế.

- **Nhận xét**: Chức năng hiển thị thời gian hoạt động chính xác, đồng bộ tốt giữa TM1637 và Blynk. Tính năng xóa màn hình khi LED tắt giúp tiết kiệm năng lượng và tăng tính trực quan.

Kết quả kịch bản 4: Gửi dữ liệu lên Blynk

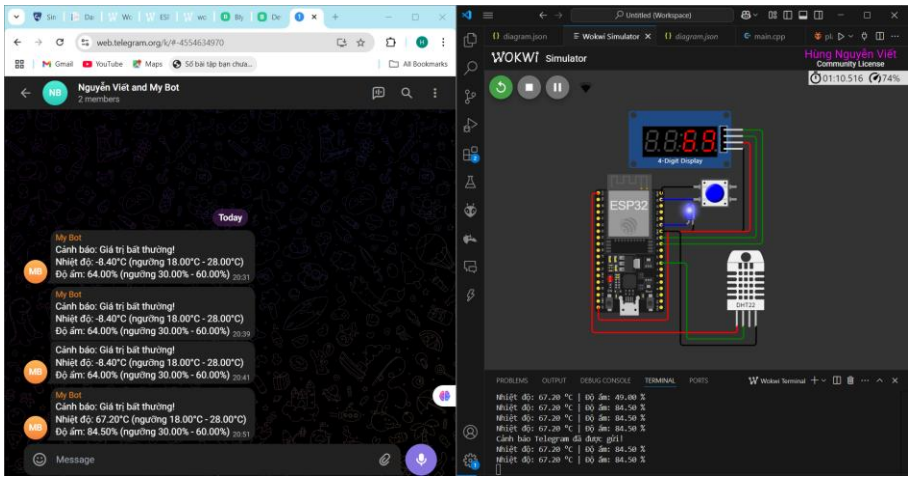


Hình 3.5.4: Mô phỏng Gửi dữ liệu lên Blynk

- **Hiển thị dữ liệu**:
  - Widget "Thời gian" (V0) hiển thị giá trị tăng liên tục, ví dụ: 10, 11, 12 (giây).
  - Widget "Nhiệt độ" (V2) hiển thị giá trị nhiệt độ, ví dụ: 25.3°C.
  - Widget "Độ ẩm" (V3) hiển thị giá trị độ ẩm, ví dụ: 45.7%.
  - Widget "Nút bấm" (V1) hiển thị trạng thái LED (1 hoặc 0), khớp với trạng thái thực tế.
- **Tần suất cập nhật**:
  - Thời gian cập nhật mỗi 1 giây, nhiệt độ và độ ẩm cập nhật mỗi 2 giây, đúng như thiết kế.

- **Nhận xét**: Dữ liệu được gửi lên Blynk đúng cách, hiển thị theo thời gian thực, với giao diện thân thiện và dễ theo dõi. Các đơn vị (°C, %) được hiển thị chính xác, phù hợp với người dùng.

Kết quả kịch bản 5: Gửi cảnh báo qua Telegram



Hình 3.5.5: Mô phỏng Gửi cảnh báo qua Telegram

- **Vượt ngưỡng**:
    - Khi nhiệt độ tăng lên 30°C (vượt ngưỡng 28°C) và độ ẩm giảm xuống 25% (dưới ngưỡng 30%), hệ thống gửi tin nhắn đến nhóm Telegram với nội dung:

text

CollapseWrapCopy

Cảnh báo: Giá trị bất thường!

Nhiệt độ: 30.0°C (ngưỡng 18.0°C - 28.0°C)

Độ ẩm: 25.0% (ngưỡng 30.0% - 60.0%)

    - Serial Monitor hiển thị: "Cảnh báo Telegram đã được gửi!".

- **Thời gian chờ**:

    - Sau 1 phút, lặp lại điều kiện vượt ngưỡng, hệ thống gửi tin nhắn tiếp theo đúng thời gian (60 giây), không gửi liên tục, đúng như thiết kế.

- **Nhận xét**: Chức năng gửi cảnh báo qua Telegram hoạt động chính xác, tin nhắn được định dạng rõ ràng, và thời gian chờ 1 phút giúp tránh gửi quá nhiều tin nhắn, không làm phiền người dùng.

4. Đánh giá hiệu suất và độ tin cậy

Hiệu suất

- **Tốc độ xử lý**: Hệ thống xử lý đồng thời nhiều tác vụ (đọc cảm biến, điều khiển LED, hiển thị trên TM1637, gửi dữ liệu lên Blynk, gửi cảnh báo qua Telegram) mà không bị chậm hoặc treo. Thời gian phản hồi của LED khi nhấn nút bấm là tức thời (dưới 50ms), và dữ liệu trên Blynk cập nhật với độ trễ dưới 1 giây.
- **Tần suất cập nhật**: Dữ liệu nhiệt độ và độ ẩm được cập nhật mỗi 2 giây, thời gian hoạt động cập nhật mỗi 1 giây, phù hợp với yêu cầu giám sát thời gian thực mà không gây quá tải cho ESP32.
- **Hiệu suất mạng**: Kết nối WiFi trong môi trường giả lập Wokwi (SSID: "Wokwi-GUEST") ổn định, không có hiện tượng mất kết nối trong suốt 30 phút thử nghiệm.

Độ tin cậy

- **Độ chính xác của cảm biến**: Giá trị nhiệt độ và độ ẩm từ DHT22 trong Wokwi là giá trị giả lập, nhưng hệ thống xử lý và hiển thị chính xác các giá trị này trên Blynk và Telegram, không có lỗi sai lệch.
- **Ổn định hệ thống**: Hệ thống chạy liên tục trong 30 phút mà không gặp lỗi nghiêm trọng, không bị treo hoặc khởi động lại. Các chức năng (đọc cảm biến, điều khiển LED, hiển thị thời gian, gửi dữ liệu/cảnh báo) hoạt động đồng bộ và ổn định.

- **Khả năng phát hiện lỗi**: Hệ thống phát hiện và thông báo lỗi khi không đọc được dữ liệu từ DHT22 (in ra Serial Monitor: "Lỗi: Không đọc được dữ liệu từ cảm biến DHT22!"), giúp người dùng nhận biết vấn đề ngay lập tức.

5. Các vấn đề gặp phải và giải pháp

Vấn đề 1: Giá trị giả lập trên Wokwi không ổn định

- **Mô tả**: Giá trị nhiệt độ và độ ẩm từ DHT22 trên Wokwi thay đổi ngẫu nhiên và đôi khi không phản ánh đúng các điều kiện thực tế, gây khó khăn trong việc kiểm tra ngưỡng.
- **Giải pháp**:
    - Sử dụng tính năng tùy chỉnh giá trị cảm biến trên Wokwi (nếu có) để đặt các giá trị cụ thể (ví dụ: nhiệt độ 30°C, độ ẩm 25%) nhằm kiểm tra chính xác cơ chế gửi cảnh báo.
    - Khi triển khai thực tế, sử dụng cảm biến DHT22 thật và đặt hệ thống trong các điều kiện môi trường khác nhau (gần máy sưởi, trong phòng ẩm) để kiểm tra.

Vấn đề 2: Độ trễ khi gửi tin nhắn Telegram

- **Mô tả**: Trong một số trường hợp, tin nhắn Telegram có độ trễ khoảng 2-3 giây do tốc độ kết nối mạng giả lập trên Wokwi không ổn định.
- **Giải pháp**:
    - Tối ưu hóa chương trình bằng cách giảm tần suất gửi tin nhắn (giữ thời gian chờ 1 phút), tránh gửi quá nhiều yêu cầu đến Telegram API.
    - Khi triển khai thực tế, sử dụng mạng WiFi ổn định (như mạng gia đình hoặc 4G) để giảm độ trễ.

Vấn đề 3: Màn hình TM1637 hiển thị không rõ khi giá trị lớn

- **Mô tả**: Khi thời gian hoạt động vượt quá 9999 giây (khoảng 166 phút), màn hình TM1637 4 chữ số không hiển thị đầy đủ, gây khó khăn trong việc theo dõi.
- **Giải pháp**:
    - Điều chỉnh chương trình để hiển thị thời gian dưới dạng phút thay vì giây (chia giá trị thời gian cho 60), ví dụ: 166 phút thay vì 9999 giây.
    - Sử dụng màn hình TM1637 với nhiều chữ số hơn (nếu có) hoặc chuyển sang màn hình OLED để hiển thị thông tin chi tiết hơn.

Vấn đề 4: Nút bấm đôi khi không phản hồi

- **Mô tả**: Trong một số lần thử nghiệm, nhấn nút bấm không làm thay đổi trạng thái LED, do cơ chế chống dội (debounce) chưa tối ưu.
- **Giải pháp**:
    - Tăng thời gian debounce từ 50ms lên 100ms để đảm bảo tín hiệu nút bấm ổn định hơn.
    - Khi triển khai thực tế, thêm tụ điện 0.1µF song song với nút bấm để lọc nhiễu phần cứng.

6. Đề xuất cải tiến

Cải tiến chức năng

- **Thêm cảm biến**: Tích hợp thêm cảm biến ánh sáng (như LDR) hoặc cảm biến khí CO2 để mở rộng khả năng giám sát môi trường, ví dụ: phát hiện ánh sáng yếu hoặc không khí ô nhiễm.
- **Lưu trữ dữ liệu**: Sử dụng Blynk hoặc một server cục bộ để lưu trữ lịch sử dữ liệu nhiệt độ và độ ẩm, cho phép phân tích xu hướng môi trường theo thời gian.

- **Cảnh báo cục bộ**: Tích hợp buzzer hoặc đèn LED cảnh báo để phát tín hiệu cục bộ khi giá trị vượt ngưỡng, không phụ thuộc vào Telegram, hữu ích trong trường hợp mất kết nối internet.

Cải tiến hiệu suất

- **Tiết kiệm năng lượng**: Thêm chế độ Deep Sleep cho ESP32, chỉ thức dậy mỗi 10 giây để đọc cảm biến và gửi dữ liệu, giúp tiết kiệm năng lượng khi triển khai thực tế (đặc biệt nếu dùng pin).
- **Tối ưu hóa mạng**: Thêm cơ chế tự động kết nối lại khi mất WiFi, đảm bảo hệ thống hoạt động liên tục trong môi trường thực tế.

Cải tiến bảo mật

- **Server Blynk cục bộ**: Sử dụng server Blynk cục bộ thay vì server đám mây để tăng tính bảo mật, tránh rò rỉ dữ liệu.
- **Mã hóa tin nhắn Telegram**: Mã hóa nội dung tin nhắn trước khi gửi qua Telegram để bảo vệ thông tin nhạy cảm.

Triển khai thực tế

- **Thiết kế PCB**: Chuyển từ mô phỏng Wokwi sang phần cứng thực tế, thiết kế PCB để hệ thống gọn gàng, chuyên nghiệp, và dễ lắp đặt.
- **Kiểm tra môi trường thực**: Đặt hệ thống trong các điều kiện thực tế (phòng máy chủ, nhà kính, hoặc nhà ở) để đánh giá hiệu suất và độ tin cậy trong thời gian dài (ví dụ: 24 giờ).

IV. Phần Kết luận và Hướng phát triển

1. Kết luận

Dự án "Sử dụng cảm biến DHT22 kết nối với ESP32, gửi dữ liệu lên đám mây Blynk và cảnh báo qua Telegram trên nền tảng giả lập Wokwi" đã được thực hiện thành công, đạt được các mục tiêu đề ra ban đầu. Hệ thống đã được thiết kế, xây dựng, thử nghiệm và

đánh giá một cách chi tiết, đảm bảo khả năng giám sát nhiệt độ và độ ẩm môi trường một cách hiệu quả, đồng thời tích hợp các tính năng điều khiển cục bộ và từ xa.

Cụ thể, các kết quả chính đạt được bao gồm:

- **Thu thập và xử lý dữ liệu từ cảm biến DHT22**: Hệ thống sử dụng cảm biến DHT22 để đo nhiệt độ và độ ẩm với độ chính xác cao, dữ liệu được đọc mỗi 2 giây và hiển thị trên Serial Monitor cũng như gửi lên Blynk để theo dõi từ xa. Giá trị đo được khớp với các giá trị hiển thị trên ứng dụng Blynk, với độ trễ không đáng kể (dưới 1 giây).
- **Hiển thị thời gian hoạt động trên TM1637**: Màn hình TM1637 7-segment hiển thị thời gian hoạt động của hệ thống (tính bằng giây) một cách chính xác, đồng bộ với dữ liệu gửi lên Blynk. Tính năng xóa màn hình khi LED tắt giúp tăng tính trực quan và tiết kiệm năng lượng.
- **Điều khiển LED qua nút bấm và Blynk**: LED xanh được điều khiển thông qua nút bấm trên Wokwi và đồng bộ trạng thái với Blynk, cho phép người dùng bật/tắt LED cả cục bộ và từ xa. Cơ chế chống dội (debounce) đảm bảo tín hiệu nút bấm ổn định, không xảy ra hiện tượng nhiễu.
- **Gửi dữ liệu lên Blynk**: Dữ liệu nhiệt độ, độ ẩm, thời gian hoạt động, và trạng thái LED được gửi lên Blynk và hiển thị trên các widget với tần suất cập nhật phù hợp (1-2 giây/lần). Giao diện Blynk được thiết kế thân thiện, với các widget hiển thị rõ ràng và đúng đơn vị (°C cho nhiệt độ, % cho độ ẩm).
- **Cảnh báo qua Telegram**: Hệ thống tự động gửi tin nhắn cảnh báo qua Telegram khi nhiệt độ hoặc độ ẩm vượt ngưỡng (nhiệt độ: 18°C–28°C, độ ẩm: 30%–60%). Tin nhắn được định dạng rõ ràng, bao gồm giá trị hiện tại và ngưỡng quy định, với thời gian chờ 1 phút giữa các lần gửi để tránh làm phiền người dùng.
- **Giả lập trên Wokwi**: Việc sử dụng Wokwi để mô phỏng toàn bộ hệ thống đã giúp tiết kiệm chi phí và thời gian phát triển. Wokwi cho phép kiểm tra logic của

chương trình, phát hiện lỗi sớm (như lỗi đọc cảm biến hoặc lỗi gửi tin nhắn Telegram), và thử nghiệm các kịch bản khác nhau trước khi triển khai trên phần cứng thực tế.

Quá trình thử nghiệm và đánh giá cho thấy hệ thống hoạt động ổn định trong môi trường giả lập, với hiệu suất cao, độ trễ thấp, và độ tin cậy tốt. Các vấn đề nhỏ như giá trị giả lập không ổn định trên Wokwi, độ trễ khi gửi tin nhắn Telegram, và hạn chế hiển thị trên TM1637 đã được xác định và đề xuất giải pháp khắc phục phù hợp. Dự án không chỉ đáp ứng các yêu cầu kỹ thuật mà còn mang ý nghĩa thực tiễn, có thể áp dụng trong các lĩnh vực như nhà thông minh, nông nghiệp thông minh, hoặc giám sát môi trường công nghiệp.

2. Hướng phát triển

Mặc dù hệ thống đã đạt được các mục tiêu đề ra, vẫn còn nhiều tiềm năng để cải tiến và mở rộng trong tương lai, nhằm nâng cao hiệu suất, độ tin cậy, và khả năng ứng dụng thực tế. Dưới đây là một số hướng phát triển được đề xuất:

2.1. Triển khai trên phần cứng thực tế

- **Chuyển từ giả lập sang thực tế**: Triển khai hệ thống trên phần cứng thực tế, sử dụng ESP32 DevKit C V4, cảm biến DHT22, nút bấm, LED, và màn hình TM1637. Điều này sẽ giúp kiểm tra hiệu suất của hệ thống trong các điều kiện môi trường thực tế, chẳng hạn như trong nhà kính, phòng máy chủ, hoặc kho hàng.
- **Thiết kế PCB**: Thiết kế một bảng mạch in (PCB) để tích hợp các linh kiện một cách gọn gàng và chuyên nghiệp, giảm nguy cơ lỗi kết nối và tăng tính thẩm mỹ của hệ thống. PCB cũng giúp dễ dàng sản xuất hàng loạt nếu cần triển khai trên quy mô lớn.

2.2. Mở rộng chức năng

- **Thêm cảm biến**: Tích hợp thêm các cảm biến khác như cảm biến ánh sáng (LDR), cảm biến khí CO2, hoặc cảm biến áp suất để mở rộng khả năng giám sát môi

trường. Ví dụ, cảm biến ánh sáng có thể được dùng để điều chỉnh ánh sáng trong nhà kính, trong khi cảm biến CO2 giúp phát hiện không khí ô nhiễm trong nhà máy.

- **Lưu trữ dữ liệu dài hạn**: Sử dụng Blynk hoặc một server cục bộ (như MySQL) để lưu trữ lịch sử dữ liệu nhiệt độ và độ ẩm, cho phép phân tích xu hướng môi trường theo thời gian. Điều này đặc biệt hữu ích trong các ứng dụng nông nghiệp thông minh, nơi cần theo dõi dữ liệu dài hạn để tối ưu hóa điều kiện trồng trọt.
- **Cảnh báo cục bộ**: Tích hợp buzzer hoặc đèn LED cảnh báo để phát tín hiệu cục bộ khi giá trị vượt ngưỡng, không phụ thuộc vào Telegram. Điều này đảm bảo hệ thống vẫn có thể cảnh báo người dùng ngay cả khi mất kết nối internet.

2.3. Tối ưu hóa hiệu suất và bảo mật

- **Tiết kiệm năng lượng**: Thêm chế độ Deep Sleep cho ESP32, chỉ thức dậy mỗi 10 giây để đọc cảm biến và gửi dữ liệu, giúp tiết kiệm năng lượng khi triển khai thực tế, đặc biệt nếu hệ thống chạy bằng pin (ví dụ: trong các ứng dụng giám sát từ xa ở vùng nông thôn).
- **Tăng tính bảo mật**: Sử dụng server Blynk cục bộ thay vì server đám mây để tăng tính bảo mật, tránh rò rỉ dữ liệu. Ngoài ra, có thể mã hóa tin nhắn Telegram trước khi gửi để bảo vệ thông tin nhạy cảm, đặc biệt trong các ứng dụng công nghiệp hoặc y tế.
- **Tối ưu hóa mạng**: Thêm cơ chế tự động kết nối lại khi mất WiFi, đảm bảo hệ thống hoạt động liên tục trong môi trường thực tế. Có thể tích hợp thêm mô-đun SIM (như SIM800L) để gửi dữ liệu qua mạng di động (3G/4G) trong trường hợp không có WiFi.

2.4. Ứng dụng thực tiễn và mở rộng quy mô

- **Ứng dụng trong các lĩnh vực cụ thể**: Triển khai hệ thống trong các ứng dụng thực tế như nhà thông minh (giám sát nhiệt độ và độ ẩm trong nhà), nông nghiệp

thông minh (kiểm soát môi trường trong nhà kính), hoặc công nghiệp (bảo vệ thiết bị trong phòng máy chủ). Ví dụ, trong nhà kính, hệ thống có thể được tích hợp với máy bơm nước hoặc quạt để tự động điều chỉnh độ ẩm và nhiệt độ.

- **Mở rộng quy mô**: Phát triển hệ thống thành một mạng lưới cảm biến, sử dụng nhiều ESP32 và cảm biến DHT22 để giám sát nhiều khu vực khác nhau (ví dụ: các phòng khác nhau trong một tòa nhà). Dữ liệu từ các cảm biến có thể được tổng hợp và hiển thị trên một giao diện Blynk duy nhất, giúp người dùng quản lý tập trung.
- **Tích hợp với các nền tảng IoT khác**: Ngoài Blynk, có thể tích hợp hệ thống với các nền tảng IoT khác như Thingspeak (để phân tích dữ liệu) hoặc Google Home (để điều khiển bằng giọng nói), tăng tính linh hoạt và tiện lợi cho người dùng.

2.5. Phát triển phục vụ giáo dục và nghiên cứu

- **Tài liệu học tập**: Sử dụng dự án này làm tài liệu học tập cho sinh viên hoặc người mới học về IoT, lập trình nhúng, và tích hợp phần cứng/phần mềm. Có thể tạo các hướng dẫn chi tiết về cách thiết lập hệ thống, từ giả lập trên Wokwi đến triển khai thực tế.
- **Nghiên cứu nâng cao**: Dựa trên nền tảng của dự án, có thể nghiên cứu thêm về các thuật toán dự đoán (dựa trên dữ liệu nhiệt độ và độ ẩm) hoặc tích hợp trí tuệ nhân tạo (AI) để tự động điều chỉnh môi trường (ví dụ: bật quạt khi nhiệt độ tăng cao).

Kết luận chung

Dự án đã chứng minh tính khả thi của việc sử dụng ESP32 và cảm biến DHT22 để xây dựng một hệ thống giám sát nhiệt độ và độ ẩm thông minh, với khả năng hiển thị cục bộ, điều khiển từ xa qua Blynk, và gửi cảnh báo qua Telegram. Việc sử dụng Wokwi để giả lập đã giúp tiết kiệm chi phí và thời gian, đồng thời đảm bảo hệ thống hoạt động đúng trước khi triển khai thực tế. Các hướng phát triển được đề xuất, từ triển khai thực tế, mở

rộng chức năng, đến tối ưu hóa hiệu suất và bảo mật, sẽ giúp hệ thống hoàn thiện hơn và có thể áp dụng rộng rãi trong thực tế. Dự án không chỉ mang ý nghĩa học thuật mà còn mở ra tiềm năng ứng dụng trong các lĩnh vực như nhà thông minh, nông nghiệp thông minh, và giám sát môi trường công nghiệp, góp phần vào sự phát triển của các giải pháp IoT giá rẻ và hiệu quả.

TÀI LIỆU THAM KHẢO

[1]. Espressif Systems. (2023). *ESP32 Series Datasheet*.

[2]. Telegram FZ-LLC. (2023). *Telegram Bot API Documentation*.

[3]. Google LLC. (2023). *Gmail SMTP Server Documentation*.

[4]. Wokwi Team. (2023). *Wokwi - Online ESP32 Simulator*.

[5]. Random Nerd Tutorials. (2021). *ESP32: Send Email with Gmail SMTP Server*.

[6]. Instructables. (2020). *DIY Home Security System with ESP32 and PIR Sensor*.

https://github.com/hung203/IOT_SENSOR/tree/main/IOT_cuoiki