

**TRƯỜNG ĐẠI HỌC KHOA HỌC  
KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỀ TÀI TIỂU LUẬN**

**ĐIỀU KHIỂN THIẾT BỊ GIA DỤNG  
QUA WIFI VỚI ESP32**

**GIẢNG VIÊN HƯỚNG DẪN: VÕ VIỆT DŨNG**

**HUẾ, THÁNG 4, NĂM 2025**

# MỤC LỤC

<b>LỜI MỞ ĐẦU .....</b>	<b>1</b>
<b>1. Tính cấp thiết của đề tài.....</b>	<b>1</b>
<b>2. Mục đích và nhiệm vụ nghiên cứu .....</b>	<b>2</b>
Mục đích .....	2
Nhiệm vụ .....	2
<b>3. Đối tượng và phạm vi nghiên cứu .....</b>	<b>2</b>
<b>PHẦN NỘI DUNG .....</b>	<b>4</b>
<b>1. Giới thiệu về ESP32.....</b>	<b>4</b>
a. Cấu trúc phần cứng và hiệu suất.....	4
b. Khả năng kết nối không dây .....	4
c. Tiêu thụ điện năng thấp và chế độ tiết kiệm năng lượng .....	5
d. Hệ thống chân GPIO linh hoạt.....	5
e. Vai trò của ESP32 trong hệ thống điều khiển thiết bị .....	6
2.1. PlatformIO trên Visual Studio Code.....	6
2.2. Mô phỏng trên Wokwi.....	7
2.3. Hiển thị dữ liệu với Blynk .....	8
2.4. Điều khiển từ xa bằng Telegram.....	8
<b>3. Giao thức truyền thông HTTP .....</b>	<b>9</b>
3.1. Giới thiệu về giao thức HTTP.....	9
3.2. Cách thức hoạt động trong hệ thống.....	10
3.2.1. Kết nối với Blynk.....	10
3.2.2. Kết nối với Telegram .....	11
3.2.3. Kết nối Wifi .....	11

3.2.4.Xử lý thông tin với luồng dữ liệu .....	11
3.3. Ví dụ về giao tiếp HTTP trong hệ thống .....	12
3.3.1. Yêu cầu HTTP từ ứng dụng đến ESP32 .....	12
3.3.2. ESP32 nhận yêu cầu và phản hồi.....	13
3.3.3. Giao tiếp với Telegram bằng HTTP.....	14
4.Thiết kế và triển khai hệ thống .....	17
4.1. Mô tả hệ thống.....	17
4.2. Các thành phần hệ thống .....	Error! Bookmark not defined.
4.3. Kiến trúc hệ thống .....	Error! Bookmark not defined.
4.4. Triển khai phần cứng.....	Error! Bookmark not defined.
4.6. Cấu trúc tệp cấu hình .....	27
5. Triển khai thực tế .....	28
5.1. Cài đặt môi trường mô phỏng .....	28
5.2. Triển khai phần mềm.....	29
KẾT LUẬN .....	32
TÀI LIỆU THAM KHẢO .....	32

## LỜI CẢM ƠN

Lời đầu tiên, em xin gửi lời cảm ơn chân thành và sâu sắc nhất đến thầy Võ Việt Dũng – người đã trực tiếp giảng dạy và hướng dẫn chúng em trong học phần Lập trình IoT học kỳ II, năm học 2024–2025. Trong suốt quá trình học tập và thực hiện tiểu luận, chúng em đã nhận được sự quan tâm, hướng dẫn tận tình và đầy tâm huyết từ thầy. Những kiến thức quý báu mà thầy truyền đạt không chỉ giúp em hiểu rõ hơn về Internet of Things (IoT) mà còn là nền tảng vững chắc để chúng em có thể hoàn thành bài tiểu

luan với đề tài: " Điều khiển thiết bị gia dụng qua Wi-Fi với ESP32." Thông qua quá trình nghiên cứu và xây dựng đề tài, em đã tích lũy thêm được nhiều kiến thức thực tiễn, tư duy ứng dụng công nghệ vào đời sống, đặc biệt là trong lĩnh vực điều khiển thiết bị gia dụng thông minh – một lĩnh vực tiềm năng và mang tính ứng dụng cao trong tương lai. Tuy đã nỗ lực hoàn thành bài tiểu luận với tinh thần học hỏi nghiêm túc, nhưng chắc chắn sẽ khó tránh khỏi những thiếu sót. Em rất mong nhận được sự góp ý, nhận xét từ thầy để bài làm được hoàn thiện hơn trong những lần sau.

Một lần nữa, em xin trân trọng cảm ơn!

# LỜI MỞ ĐẦU

## 1. Tính cấp thiết của đề tài

Trong thời đại công nghệ số phát triển mạnh mẽ, Internet of Things (IoT) đang ngày càng trở thành một xu hướng tất yếu, đóng vai trò quan trọng trong việc thúc đẩy quá trình tự động hóa và điều khiển thiết bị từ xa. Công nghệ này không chỉ góp mặt trong các lĩnh vực công nghiệp, y tế, nông nghiệp mà còn thâm nhập sâu rộng vào đời sống thường nhật, đặc biệt là trong mô hình nhà thông minh (Smart Home). Việc điều khiển thiết bị gia dụng như đèn, quạt, máy lạnh từ xa thông qua điện thoại thông minh hoặc ứng dụng web không chỉ giúp tiết kiệm thời gian, năng lượng mà còn tăng tính tiện nghi và an toàn cho người sử dụng.

Trong bối cảnh đó, vi điều khiển ESP32 nổi lên như một lựa chọn lý tưởng nhờ khả năng kết nối Wi-Fi mạnh mẽ, hiệu suất xử lý cao, mức tiêu thụ điện năng thấp, và hỗ trợ nhiều giao thức truyền thông hiện đại. Với việc tích hợp đa dạng các cổng kết nối và cảm biến, ESP32 cho phép xây dựng các hệ thống điều khiển thông minh một cách linh hoạt, hiệu quả và tiết kiệm chi phí.

Bài tiểu luận này được thực hiện với mục tiêu nghiên cứu, thiết kế và xây dựng một mô hình điều khiển thiết bị gia dụng (đèn, quạt) qua Wi-Fi sử dụng ESP32, kết hợp cùng các nền tảng hỗ trợ hiện đại như Blynk (web/mobile) để tạo giao diện điều khiển, Telegram chatbot để điều khiển từ xa qua tin nhắn, và Wokwi để mô phỏng hệ thống trước khi triển khai trên phần cứng thực tế. Ngoài ra, bài viết còn đề cập đến phương pháp đồng bộ trạng thái thiết bị giữa các nền tảng nhằm đảm bảo tính nhất quán trong điều khiển và giám sát thiết bị ở mọi thời điểm.

Đây là một đề tài có ý nghĩa thiết thực, giúp người học tiếp cận được công nghệ IoT hiện đại, hiểu và áp dụng vào thực tiễn một cách dễ dàng và hiệu quả, đồng thời cũng phù hợp với xu hướng chuyển đổi số trong đời sống hiện nay.

## **2. Mục đích và nhiệm vụ nghiên cứu**

### **Mục đích:**

- Nghiên cứu, thiết kế và xây dựng hệ thống điều khiển thiết bị gia dụng từ xa sử dụng vi điều khiển ESP32 thông qua kết nối Wi-Fi.
- Tạo giao diện điều khiển trực quan trên ứng dụng web hoặc mobile (Blynk), tích hợp điều khiển qua Telegram nhằm mở rộng tính linh hoạt của hệ thống.
- Đảm bảo khả năng đồng bộ trạng thái thiết bị giữa các nền tảng điều khiển để nâng cao độ chính xác và tính nhất quán trong vận hành.

### **Nhiệm vụ:**

- Tìm hiểu lý thuyết về Internet of Things, vi điều khiển ESP32, giao thức HTTP và các công nghệ hỗ trợ như Blynk, Telegram Bot API, PlatformIO, Wokwi.
- Lập trình ESP32 để kết nối mạng Wi-Fi, nhận lệnh từ ứng dụng và điều khiển thiết bị gia dụng như đèn và quạt.
- Xây dựng giao diện điều khiển người dùng trên nền tảng Blynk web/mobile và Telegram Bot để bật/tắt thiết bị từ xa.
- Mô phỏng hệ thống trên Wokwi để kiểm thử logic hoạt động trước khi triển khai thực tế.
- Đồng bộ hóa trạng thái thiết bị giữa các nền tảng (Blynk và Telegram) để duy trì sự thống nhất và tránh xung đột điều khiển.
- Đánh giá tính khả thi, hiệu quả của hệ thống và đề xuất hướng phát triển mở rộng.

## **3. Đối tượng và phạm vi nghiên cứu**

- Đối tượng nghiên cứu:
  - + Các thiết bị gia dụng cơ bản (đèn, quạt) có thể bật/tắt thông qua tín hiệu điện.

+Vi điều khiển ESP32, là trung tâm xử lý và kết nối giữa hệ thống và người dùng.

+Các ứng dụng điều khiển từ xa dựa trên nền tảng IoT (Blynk, Telegram).

- Phạm vi nghiên cứu:

+Tập trung xây dựng mô hình Smart Home cơ bản với chức năng bật/tắt thiết bị gia dụng thông qua kết nối Wi-Fi và điều khiển từ xa.

+Ứng dụng giao thức HTTP để truyền lệnh giữa người dùng và thiết bị.

+Sử dụng Wokwi để mô phỏng hệ thống trên nền tảng ảo trước khi triển khai thực tế.

+Không đi sâu vào các giao thức nâng cao như MQTT, WebSocket, hay hệ thống điều khiển phức tạp nhiều tầng.

+Giới hạn phần cứng thực tế ở ESP32 và một vài thiết bị đầu ra đơn giản (LED, LED thay thế cho quạt ).

# PHẦN NỘI DUNG

## 1. Giới thiệu về ESP32

ESP32 là một vi điều khiển mạnh mẽ do Espressif Systems phát triển, được thiết kế để phục vụ các ứng dụng IoT, nhúng và tự động hóa. Đây là phiên bản nâng cấp từ ESP8266, mang đến nhiều cải tiến đáng kể về hiệu suất, khả năng kết nối và các tính năng hỗ trợ phần cứng. Nhờ tích hợp Wi-Fi và Bluetooth ngay trên chip, ESP32 giúp việc xây dựng các hệ thống điều khiển từ xa và truyền dữ liệu qua mạng trở nên đơn giản, tiết kiệm chi phí mà không cần thêm module mở rộng.

### a. Cấu trúc phần cứng và hiệu suất

ESP32 được trang bị bộ xử lý lõi kép Xtensa LX6, với tốc độ xung nhịp có thể lên đến 240 MHz, giúp nó có thể thực thi nhiều tác vụ cùng lúc một cách hiệu quả. Ngoài ra, nó còn có bộ nhớ SRAM 520 KB, bộ nhớ flash mở rộng từ 4MB đến 16MB, phù hợp để lưu trữ chương trình và dữ liệu.

ESP32 cũng tích hợp một số module xử lý tín hiệu số (DSP) và hỗ trợ các giao thức truyền thông phổ biến như SPI, I2C, UART, CAN, PWM, ADC, DAC, giúp nó có thể giao tiếp với nhiều loại cảm biến và thiết bị ngoại vi khác nhau.

### b. Khả năng kết nối không dây

Một trong những ưu điểm lớn nhất của ESP32 là hỗ trợ kết nối Wi-Fi 2.4 GHz và Bluetooth (BLE + Classic), giúp nó có thể dễ dàng kết nối với các thiết bị di động, máy chủ hoặc hệ thống đám mây. Cụ thể:

- Wi-Fi 802.11 b/g/n: Hỗ trợ cả chế độ Station (kết nối vào mạng Wi-Fi) và Access Point (AP) (tạo điểm phát Wi-Fi).
- Bluetooth 4.2 (BLE & Classic): Có thể kết nối với các thiết bị Bluetooth khác như smartphone, loa thông minh hoặc cảm biến không dây.



Khả năng kết nối mạnh mẽ này giúp ESP32 trở thành một lựa chọn lý tưởng cho các ứng dụng nhà thông minh, hệ thống tự động hóa, giám sát môi trường và truyền dữ liệu không dây.

### **c. Tiêu thụ điện năng thấp và chế độ tiết kiệm năng lượng**

ESP32 được thiết kế để hoạt động trong nhiều chế độ tiêu thụ điện năng khác nhau, giúp tối ưu hóa thời gian sử dụng pin cho các ứng dụng chạy bằng pin hoặc năng lượng mặt trời. Một số chế độ tiết kiệm năng lượng bao gồm:

- **Modem Sleep:** Tắt Wi-Fi khi không cần thiết, nhưng vẫn giữ CPU hoạt động.
- **Light Sleep:** Giảm xung nhịp CPU, chỉ duy trì các nhiệm vụ quan trọng.
- **Deep Sleep:** Ngắt CPU, chỉ giữ một số module hoạt động (ví dụ: bộ đếm thời gian hoặc cảm biến chuyển động).
- **Hibernation Mode:** Ngừng tất cả hoạt động của vi điều khiển để tiết kiệm tối đa năng lượng.

Nhờ những tính năng này, ESP32 có thể hoạt động lâu dài mà không cần nguồn điện liên tục, phù hợp với các ứng dụng IoT cần tiết kiệm năng lượng.

### **d. Hệ thống chân GPIO linh hoạt**

ESP32 cung cấp hơn 30 chân GPIO (General Purpose Input/Output) có thể cấu hình để thực hiện nhiều chức năng khác nhau, chẳng hạn như:

- Điều khiển thiết bị (Relay, LED, động cơ servo, quạt...)
- Đọc dữ liệu cảm biến (Nhiệt độ, độ ẩm, ánh sáng, chuyển động...)
- Giao tiếp với màn hình hiển thị (OLED, LCD, TFT...)
- Giao tiếp với bộ nhớ ngoài và module mở rộng

Các chân GPIO này có thể hỗ trợ tín hiệu số và tín hiệu tương tự, giúp ESP32 dễ dàng kết nối với các thiết bị khác trong hệ thống IoT.

### **e. Vai trò của ESP32 trong hệ thống điều khiển thiết bị**

Trong hệ thống này, ESP32 sẽ đóng vai trò là bộ điều khiển trung tâm, thực hiện các nhiệm vụ chính sau:

- Nhận lệnh từ người dùng thông qua ứng dụng Blynk hoặc tin nhắn trên Telegram.
- Xử lý dữ liệu và điều khiển thiết bị như bật/tắt đèn, quạt hoặc các thiết bị gia dụng khác.
- Gửi phản hồi về trạng thái thiết bị để hiển thị trên Blynk và Telegram, giúp người dùng theo dõi tình trạng thiết bị từ xa.
- Đồng bộ trạng thái giữa các nền tảng để đảm bảo sự nhất quán trong quá trình điều khiển và giám sát.

Với sự kết hợp giữa khả năng xử lý mạnh mẽ, kết nối không dây linh hoạt, tiêu thụ năng lượng thấp và khả năng giao tiếp với nhiều thiết bị ngoại vi, ESP32 là một lựa chọn lý tưởng cho hệ thống điều khiển thông minh và sẽ giúp nâng cao trải nghiệm sử dụng thiết bị gia dụng trong các ứng dụng IoT.

## **2. Tổng quan về công nghệ sử dụng**

Hệ thống điều khiển thiết bị gia dụng qua Wi-Fi bằng ESP32 được xây dựng dựa trên nhiều công nghệ hỗ trợ để tối ưu hóa quá trình phát triển, mô phỏng và triển khai thực tế. Các công nghệ chính bao gồm PlatformIO, Wokwi, Blynk, và Telegram, với giao tiếp giữa các thành phần sử dụng giao thức HTTP.

### **2.1. PlatformIO trên Visual Studio Code**

PlatformIO là một môi trường phát triển tích hợp (IDE) dành cho các dự án nhúng, đặc biệt là các vi điều khiển như ESP32. Trong hệ thống này, PlatformIO được sử dụng thay cho Arduino IDE để lập trình và quản lý dự án, nhờ các ưu điểm sau:

- Hỗ trợ Visual Studio Code, giúp lập trình viên tận dụng các tính năng như tô màu cú pháp, tự động hoàn thành mã, debug.

- Tích hợp công cụ biên dịch mạnh mẽ, giúp quá trình biên dịch và tải chương trình lên ESP32 nhanh hơn.
- Hệ thống quản lý thư viện hiệu quả, tự động cập nhật và cài đặt các thư viện cần thiết như Blynk, WiFi, ArduinoJson, UniversalTelegramBot.
- Cấu trúc dự án chuyên nghiệp, sử dụng tệp cấu hình platformio.ini giúp dễ dàng quản lý thông số phần cứng và phần mềm.

Nhờ những lợi ích trên, PlatformIO được sử dụng để lập trình và triển khai mã nguồn lên ESP32 một cách hiệu quả.

## **2.2. Mô phỏng trên Wokwi**

Wokwi là một trình mô phỏng phần cứng mạnh mẽ, hỗ trợ các vi điều khiển như ESP32, Arduino và Raspberry Pi, giúp kiểm tra mã nguồn mà không cần sử dụng phần cứng thực tế. Trong hệ thống này, Wokwi đóng vai trò mô phỏng ESP32 và các thiết bị ngoại vi như đèn LED, cảm biến, giúp kiểm tra chương trình trước khi triển khai thực tế.

Những lợi ích khi sử dụng Wokwi bao gồm:

- Tiết kiệm thời gian và chi phí do không cần mua phần cứng để kiểm tra mã nguồn.
- Mô phỏng mạch điện dễ dàng, hỗ trợ kéo thả linh kiện và lập sơ đồ mạch trực quan.
- Giám sát dữ liệu theo thời gian thực, giúp theo dõi trạng thái thiết bị và debug lỗi hiệu quả.

Nhờ đó, Wokwi giúp đảm bảo mã nguồn hoạt động đúng trước khi được triển khai trên phần cứng thực tế.

## 2.3. Hiển thị dữ liệu với Blynk

Blynk là một nền tảng IoT cho phép tạo giao diện điều khiển trên thiết bị di động, giúp giám sát và điều khiển thiết bị từ xa. Trong hệ thống này, Blynk được sử dụng để hiển thị trạng thái thiết bị và gửi lệnh điều khiển đến ESP32 thông qua giao thức HTTP.

Các tính năng chính của Blynk bao gồm:

- Giao diện trực quan trên ứng dụng di động, giúp người dùng dễ dàng kiểm soát thiết bị.
- Giao tiếp với ESP32 qua HTTP API, cho phép gửi và nhận dữ liệu một cách nhanh chóng.
- Lưu trạng thái thiết bị bằng Blynk Virtual Pins, giúp đồng bộ dữ liệu giữa ESP32 và ứng dụng di động.
- Thông báo theo thời gian thực, gửi cảnh báo khi thiết bị thay đổi trạng thái.

Trong hệ thống này, ESP32 gửi và nhận dữ liệu từ Blynk thông qua các yêu cầu HTTP GET và HTTP POST, giúp đảm bảo tính ổn định và dễ triển khai trong môi trường IoT.

## 2.4. Điều khiển từ xa bằng Telegram

Telegram là một ứng dụng nhắn tin phổ biến, hỗ trợ tích hợp chatbot để giao tiếp với các thiết bị IoT. Trong hệ thống này, Telegram được sử dụng để điều khiển thiết bị từ xa thông qua bot Telegram, sử dụng Telegram Bot API kết hợp với giao thức HTTP.

Các tính năng chính của Telegram Bot trong hệ thống bao gồm:

- Nhận lệnh điều khiển từ người dùng, cho phép bật/tắt thiết bị thông qua tin nhắn.
- Gửi phản hồi trạng thái thiết bị, giúp người dùng biết thiết bị đang bật hay tắt.
- Bảo mật cao, sử dụng token API để đảm bảo chỉ người dùng được cấp quyền mới có thể điều khiển thiết bị.

- Tương thích với nhiều nền tảng, hoạt động trên điện thoại, máy tính bảng, PC mà không cần cài đặt thêm phần mềm.

Hệ thống sử dụng ESP32 kết nối với Telegram Bot API qua HTTP, cho phép gửi lệnh từ Telegram đến ESP32 và nhận phản hồi về trạng thái thiết bị theo thời gian thực.

### **3. Giao thức truyền thông HTTP**

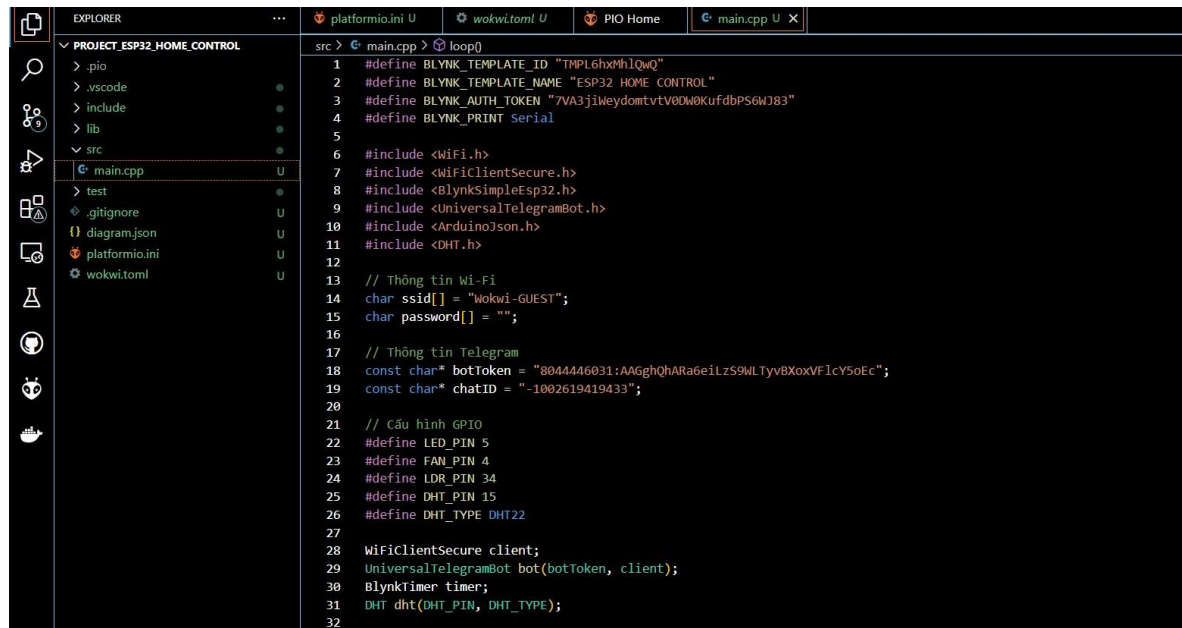
#### **3.1. Giới thiệu về giao thức HTTP**

HTTP (Hypertext Transfer Protocol) là giao thức truyền tải siêu văn bản được sử dụng rộng rãi trong các hệ thống web và IoT. HTTP hoạt động theo mô hình yêu cầu – phản hồi (request – response), trong đó client (thiết bị gửi yêu cầu) và server (thiết bị xử lý yêu cầu) trao đổi dữ liệu thông qua các phương thức như GET, POST, PUT, DELETE.

Trong hệ thống này, ESP32 đóng vai trò là một server, nhận yêu cầu từ ứng dụng Blynk hoặc Telegram và phản hồi trạng thái thiết bị. Ngược lại, ứng dụng web/mobile hoặc bot Telegram đóng vai trò là client, gửi yêu cầu đến ESP32 để điều khiển thiết bị.

## 3.2. Cách thức hoạt động trong hệ thống

### 3.2.1. Kết nối với Blynk



```
1 #define BLYNK_TEMPLATE_ID "TMPL6hxMh1QwQ"
2 #define BLYNK_TEMPLATE_NAME "ESP32 HOME CONTROL"
3 #define BLYNK_AUTH_TOKEN "7VA3jiWeydomtvtV0DW0KufdbPS6WJ83"
4 #define BLYNK_PRINT Serial
5
6 #include <WiFi.h>
7 #include <WiFiClientSecure.h>
8 #include <BlynkSimpleEsp32.h>
9 #include <UniversalTelegramBot.h>
10 #include <ArduinoJson.h>
11 #include <DHT.h>
12
13 // Thông tin Wi-Fi
14 char ssid[] = "Wokwi-GUEST";
15 char password[] = "";
16
17 // Thông tin Telegram
18 const char* botToken = "80444446031:AAgghQhARA6eilzS9WLTyvBXoxVflcY5oEc";
19 const char* chatID = "-1002619419433";
20
21 // Cấu hình GPIO
22 #define LED_PIN 5
23 #define FAN_PIN 4
24 #define LDR_PIN 34
25 #define DHT_PIN 15
26 #define DHT_TYPE DHT22
27
28 WiFiClientSecure client;
29 UniversalTelegramBot bot(botToken, client);
30 BlynkTimer timer;
31 DHT dht(DHT_PIN, DHT_TYPE);
32
```

-Sử dụng thư viện `#include <BlynkSimpleEsp32.h>` để kết nối với Blynk

-Khai báo mã để kết nối với Blynk

```
#define BLYNK_TEMPLATE_ID "TMPL6hxMh1QwQ"
#define BLYNK_TEMPLATE_NAME "ESP32 HOME CONTROL"
#define BLYNK_AUTH_TOKEN "7VA3jiWeydomtvtV0DW0KufdbPS6WJ83"
```

Hiển thị Serial Monitor lúc chạy mô phỏng

```
#define BLYNK_PRINT Serial
```

Kiểm tra lại kết nối Blynk định kỳ mỗi 10 giây

```
timer.setInterval(10000L, []() {
if (!Blynk.connected()) {
    Serial.println("[LỖI] Mất kết nối Blynk! Đang kết nối lại...");
    Blynk.connect();
}
}
```

Cập nhật trạng thái trên Blynk :

```
Blynk.virtualWrite(V0, digitalRead(LED_PIN));
Blynk.virtualWrite(V1, digitalRead(FAN_PIN));
```

### 3.2.2. Kết nối với Telegram

-Sử dụng thư viện

```
#include <WiFiClientSecure.h>
#include <UniversalTelegramBot.h>
```

Khai báo thông tin Telegram :

Thông tin Telegram

```
const char* botToken = "8044446031:AAGghQhARa6eiLzS9WLTyvBXoxVF1cY5oEc";
const char* chatID = "-1002619419433";
```

khởi

tạo:

```
WiFiClientSecure client;
```

```
UniversalTelegramBot bot(botToken, client);
```

Nhận lệnh và xử lý lệnh từ Telegram:

```
void handleNewMessages() {
    int numNewMessages = bot.getUpdates(bot.last_message_received + 1);
    for (int i = 0; i < numNewMessages; i++) {
        String text = bot.messages[i].text;

        if (text == "/den_on") digitalWrite(LED_PIN, HIGH);
        if (text == "/den_off") digitalWrite(LED_PIN, LOW);
        if (text == "/quat_on") digitalWrite(FAN_PIN, HIGH);
        if (text == "/quat_off") digitalWrite(FAN_PIN, LOW);
        if (text == "/trangthai") sendDeviceStatus();
    }
}
```

### 3.2.3.Kết nối Wifi

-Sử dụng thư viện `#include <WiFi.h>`

Thông tin Wifi:

```
char ssid[] = "Wokwi-GUEST"; char password[] = "";
```

### 3.2.4.Xử lý thông tin với luồng dữ liệu

Hệ thống sử dụng giao thức HTTP để giao tiếp giữa ESP32, ứng dụng Blynk và Telegram theo các bước sau:

Bước 1: Người dùng gửi yêu cầu từ ứng dụng web, mobile hoặc bot Telegram

- Khi người dùng nhấn nút BẬT/TẮT trên ứng dụng Blynk, ứng dụng sẽ gửi một yêu cầu HTTP GET hoặc HTTP POST đến ESP32.
- Nếu người dùng nhập lệnh "/bat\_den" trên Telegram, bot sẽ gửi một yêu cầu HTTP đến ESP32 để kích hoạt thiết bị.

Bước 2: ESP32 nhận yêu cầu và xử lý

- ESP32 đọc yêu cầu từ HTTP request, kiểm tra nội dung và thực hiện hành động tương ứng (bật/tắt đèn, quạt, thiết bị khác).
- Sau khi xử lý, ESP32 gửi phản hồi HTTP chứa trạng thái thiết bị về ứng dụng.

Bước 3: Cập nhật trạng thái thiết bị trong ứng dụng

- Blynk cập nhật giao diện để hiển thị trạng thái mới của thiết bị.
- Bot Telegram gửi phản hồi cho người dùng để xác nhận thiết bị đã thay đổi trạng thái.

Xử lý ngoài luồng:

- Cảm biến DHT22 cập nhật nhiệt độ hiện tại, nếu nhiệt độ lớn hơn 30C thì quạt tự động bật.
- Cảm biến LDR giúp cập nhật độ sáng của phòng, nếu như độ sáng dưới 200lx thì tự động bật đèn .

### **3.3. Ví dụ về giao tiếp HTTP trong hệ thống**

#### **3.3.1. Yêu cầu HTTP từ ứng dụng đến ESP32**

Giao tiếp với Blynk qua HTTP

Khi người dùng nhấn nút trong ứng dụng Blynk, ESP32 nhận lệnh thông qua BLYNK\_WRITE(), điều khiển thiết bị và gửi phản hồi.



```
cpp
Sao chép  Chỉnh sửa

// 🌐 Điều khiển đèn qua Blynk (V0)
BLYNK_WRITE(V0) {
    int state = param.asInt();
    digitalWrite(LED_PIN, state);
    sendDeviceStatus(); // Gửi trạng thái thiết bị lên Blynk và Telegram

    Serial.println(state ? "[LOG] 🟡 Đèn BẬT" : "[LOG] 🟡 Đèn TẮT");
    bot.sendMessage(chatID, state ? "🟡 Đèn đã BẬT" : "🟡 Đèn đã TẮT");
}

// 🌐 Điều khiển quạt qua Blynk (V1)
BLYNK_WRITE(V1) {
    int state = param.asInt();
    digitalWrite(FAN_PIN, state);
    sendDeviceStatus();

    Serial.println(state ? "[LOG] 🔵 Quạt BẬT" : "[LOG] 🔵 Quạt TẮT");
    bot.sendMessage(chatID, state ? "🔵 Quạt đã BẬT" : "🔵 Quạt đã TẮT");
}
```

- Cách hoạt động:

- + Khi người dùng nhấn nút trên Blynk, ứng dụng gửi yêu cầu HTTP đến máy chủ Blynk.
- + Máy chủ Blynk chuyển yêu cầu này đến ESP32 qua HTTP
- + ESP32 nhận lệnh, điều khiển thiết bị và gửi trạng thái phản hồi.

### 3.3.2. ESP32 nhận yêu cầu và phản hồi

Gửi phản hồi trạng thái về Blynk và Telegram

```

void sendDeviceStatus() {
    float temp = dht.readTemperature();
    int light = analogRead(LDR_PIN);
    String ledStatus = digitalRead(LED_PIN) ? "BẬT" : "TẮT";
    String fanStatus = digitalRead(FAN_PIN) ? "BẬT" : "TẮT";

    // Gửi dữ liệu lên Blynk
    Blynk.virtualWrite(V3, temp);          // Nhiệt độ
    Blynk.virtualWrite(V4, light);         // Cường độ ánh sáng
    Blynk.virtualWrite(V2, "💡 Đèn: " + ledStatus + "\n🌀 Quạt: " + fanStatus);

    // Gửi thông báo Telegram qua HTTP
    String status = "💡 Đèn: " + ledStatus +
        "\n🌀 Quạt: " + fanStatus +
        "\n🌡️ Nhiệt độ: " + String(temp) + "°C" +
        "\n💡 Ánh sáng: " + String(light);
    bot.sendMessage(chatID, status); // Gửi tin nhắn HTTP đến Telegram

    Serial.println("[LOG] " + status);
}

```

-Cách hoạt động: +ESP32 gửi dữ liệu lên Blynk qua Blynk.virtualWrite(), sử dụng HTTP để cập nhật trạng thái cảm biến và thiết bị.

+ESP32 gửi tin nhắn Telegram qua bot.sendMessage(), gọi một API HTTP của Telegram.

### 3.3.3. Giao tiếp với Telegram bằng HTTP

Nhận lệnh điều khiển từ Telegram

```
cpp                                                                    Sao chép    ✎    Chính sửa

void handleNewMessages() {
    int numNewMessages = bot.getUpdates(bot.last_message_received + 1);
    for (int i = 0; i < numNewMessages; i++) {
        String text = bot.messages[i].text;

        if (text == "/den_on") digitalWrite(LED_PIN, HIGH);
        if (text == "/den_off") digitalWrite(LED_PIN, LOW);
        if (text == "/quat_on") digitalWrite(FAN_PIN, HIGH);
        if (text == "/quat_off") digitalWrite(FAN_PIN, LOW);
        if (text == "/trangthai") sendDeviceStatus();

        // Cập nhật trạng thái lên Blynk
        Blynk.virtualWrite(V0, digitalRead(LED_PIN));
        Blynk.virtualWrite(V1, digitalRead(FAN_PIN));
    }
}
```

-Cách hoạt động:

+ESP32 gửi HTTP GET đến máy chủ Telegram để kiểm tra tin nhắn mới qua bot.getUpdates().

+Nếu có lệnh điều khiển từ người dùng (/den\_on, /den\_off), ESP32 thực hiện hành động tương ứng.

+Sau khi xử lý, ESP32 gửi phản hồi HTTP đến Telegram (bot.sendMessage()).

### 3.4. Ưu điểm của giao thức HTTP trong hệ thống

Giao thức HTTP được lựa chọn làm phương thức truyền thông giữa ESP32 và ứng dụng do các ưu điểm sau:

- Dễ triển khai: Không yêu cầu máy chủ trung gian, chỉ cần ESP32 kết nối Wi-Fi và chạy server HTTP đơn giản.
- Tương thích với nhiều nền tảng: Hoạt động tốt trên web, mobile và Telegram, giúp mở rộng khả năng điều khiển thiết bị từ nhiều môi trường khác nhau.

- Hỗ trợ bảo mật: Có thể sử dụng HTTPS để mã hóa dữ liệu, đảm bảo an toàn trong quá trình truyền tải.

- Linh hoạt: Hỗ trợ nhiều phương thức HTTP như GET, POST, phù hợp với các thao tác điều khiển và truy vấn trạng thái thiết bị.

## 4. Thiết kế và triển khai hệ thống

### 4.1. Mô tả hệ thống

Hệ thống này được thiết kế để điều khiển các thiết bị như đèn, quạt và giám sát các cảm biến như nhiệt độ (DHT22) và ánh sáng (LDR) thông qua giao diện ứng dụng di động (Blynk) và Telegram. ESP32 là bộ điều khiển trung tâm, kết nối với các cảm biến và thiết bị ngoại vi, đồng thời giao tiếp với ứng dụng Blynk và Telegram để nhận và gửi dữ liệu.

- **ESP32 Devkit V4**

Là bộ vi điều khiển trung tâm của hệ thống. ESP32 có khả năng giao tiếp Wi-Fi để kết nối với Blynk và Telegram. Nó chịu trách nhiệm thu thập dữ liệu từ các cảm biến, xử lý và điều khiển các thiết bị đầu ra như đèn và quạt.

- **Cảm biến nhiệt độ và độ ẩm DHT22**

Dùng để đo nhiệt độ và độ ẩm môi trường.

- **Kết nối:**

- VCC → chân 3.3V trên ESP32
    - DATA → chân GPIO15
    - GND → GND của ESP32

- **Chức năng:** Truyền dữ liệu môi trường đến ESP32 để xử lý, có thể làm cơ sở để điều khiển quạt.

- **Cảm biến ánh sáng LDR (Light Dependent Resistor)**

Dùng để phát hiện mức độ ánh sáng môi trường.

- **Kết nối:**

- VCC → chân 3.3V
    - AO (Analog Output) → chân GPIO34
    - GND → GND của ESP32

- **Chức năng:** Cung cấp tín hiệu analog dựa trên cường độ ánh sáng xung quanh, giúp hệ thống xác định khi nào cần bật đèn.

- **Đèn LED Đỏ (LED1)**

Đại diện cho thiết bị đèn trong nhà, có thể điều khiển từ xa qua Blynk hoặc Telegram, hoặc tự động bật khi ánh sáng yếu (dưới 200 lux).

- **Kết nối:**

- Cực dương (A) → chân GPIO22 của ESP32
    - Cực âm (C) → qua điện trở 220Ω → GND

- **Đèn LED Xanh (Fan\_LED)**

Mô phỏng thiết bị quạt trong hệ thống.

- **Kết nối:**

- Cực dương (A) → chân GPIO23

- Cực âm (C) → qua điện trở  $220\Omega$  → GND
  - **Chức năng:** Hoạt động như quạt, bật khi nhiệt độ vượt ngưỡng hoặc theo điều khiển từ người dùng qua Blynk/Telegram.
- **Nút nhấn (Button 1 & 2)**  
Dùng để điều khiển thủ công các thiết bị từ phần cứng.
  - **Nút đỏ (Button 1):**
    - Một chân nối GND
    - Chân còn lại nối GPIO19
  - **Nút xanh (Button 2):**
    - Một chân nối GND
    - Chân còn lại nối GPIO18
  - **Chức năng:** Cho phép người dùng bật/tắt thiết bị thủ công trực tiếp trên hệ thống mô phỏng.
- **Điện trở**
  - **Điện trở R1 ( $10k\Omega$ ):** Dùng làm pull-up cho cảm biến hoặc nút nhấn.
  - **Điện trở R4, R2 ( $220\Omega$ ):** Giới hạn dòng cho các LED đỏ và LED xanh.

### 4.3. Kiến trúc hệ thống

Hệ thống hoạt động dựa trên mô hình kết nối qua Wi-Fi:

- **ESP32** kết nối mạng không dây và trao đổi dữ liệu với:
  - **Blynk:** Hiển thị dữ liệu cảm biến và điều khiển đèn, quạt.
  - **Telegram:** Nhận lệnh điều khiển và phản hồi trạng thái thiết bị thông qua tin nhắn.
- Giao tiếp giữa ESP32 và các nền tảng được thực hiện thông qua giao thức **HTTP**.

### 4.4. Triển khai phần cứng

#### 4.4. Triển khai phần cứng

Trong hệ thống này, vi điều khiển **ESP32** đóng vai trò trung tâm, kết nối và điều khiển các cảm biến, thiết bị ngoại vi thông qua các chân GPIO. Việc mô phỏng được thực hiện trên nền tảng **Wokwi**, các chân của các thiết bị được kết nối cụ thể như sau:

#### LED1 (đèn đỏ):

- **Loại linh kiện:** LED màu đỏ
- **Mục đích sử dụng:** Đại diện cho thiết bị đèn chiếu sáng trong hệ thống.

- - **Kết nối:**
    - Cực dương (A) nối với chân **GPIO22** của ESP32 thông qua điện trở 220Ω.
    - Cực âm (C) nối với chân **GND** thông qua điện trở.
  - **Chức năng:** Được điều khiển bật/tắt khi ánh sáng môi trường giảm xuống dưới một mức ngưỡng do cảm biến LDR cung cấp, hoặc thông qua điều khiển từ xa bằng ứng dụng Blynk hoặc Telegram.
- 

#### Fan LED (đèn xanh - mô phỏng quạt):

- **Loại linh kiện:** LED màu xanh dương
  - **Mục đích sử dụng:** Mô phỏng thiết bị quạt trong hệ thống gia dụng thông minh.
  - **Kết nối:**
    - Cực dương (A) nối với chân **GPIO23** của ESP32.
    - Cực âm (C) nối với chân **GND** thông qua điện trở 220Ω.
  - **Chức năng:** Bật/tắt khi nhiệt độ môi trường vượt ngưỡng cho phép (do DHT22 cung cấp), hoặc theo điều khiển từ xa qua Blynk/Telegram.
- 

#### Cảm biến nhiệt độ và độ ẩm DHT22:

- **Loại linh kiện:** Cảm biến DHT22
  - **Mục đích sử dụng:** Đo nhiệt độ và độ ẩm môi trường để phục vụ việc tự động hóa điều khiển quạt hoặc thông báo môi trường.
  - **Kết nối:**
    - Chân **VCC** nối với chân **3.3V** của ESP32.
    - Chân **DATA** nối với chân **GPIO15**.
    - Chân **GND** nối với **GND** của ESP32.
  - **Chức năng:** Cung cấp dữ liệu nhiệt độ/độ ẩm cho hệ thống để xử lý và điều khiển thiết bị.
- 

#### Cảm biến ánh sáng LDR (quang trở):

- **Loại linh kiện:** Cảm biến ánh sáng quang trở
- **Mục đích sử dụng:** Đo cường độ ánh sáng môi trường để điều khiển đèn tự động.
- **Kết nối:**
  - Chân **VCC** nối với chân **3.3V** của ESP32.
  - Chân **AO** (analog output) nối với chân **GPIO34**.
  - Chân **GND** nối với **GND** của ESP32.

- - **Chức năng:** Gửi giá trị điện áp tương ứng với độ sáng môi trường, từ đó hệ thống quyết định việc bật/tắt đèn.
- 

#### **Nút nhấn đỏ (Button 1):**

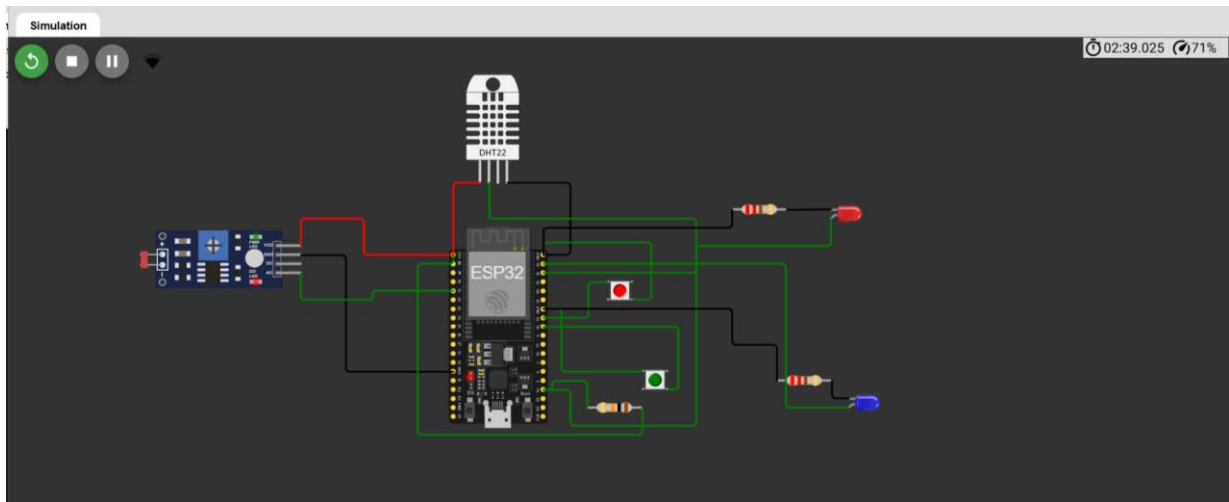
- **Loại linh kiện:** Push Button 6mm màu đỏ
- **Mục đích sử dụng:** Là nút điều khiển thủ công, ví dụ như khởi động/tắt đèn.
- **Kết nối:**
  - Một chân nối với **GND**
  - Chân còn lại nối với chân **GPIO19** của ESP32
- **Chức năng:** Gửi tín hiệu điều khiển thủ công khi được nhấn.

#### **Nút nhấn xanh lá (Button 2):**

- **Loại linh kiện:** Push Button 6mm màu xanh lá
- **Mục đích sử dụng:** Là nút điều khiển thủ công thứ hai, ví dụ như bật quạt hoặc reset hệ thống.
- **Kết nối:**
  - Một chân nối với **GND**
  - Chân còn lại nối với chân **GPIO18** của ESP32
- **Chức năng:** Điều khiển trực tiếp thiết bị theo chương trình cài sẵn.



## Sơ đồ kết nối phần cứng (Dựa trên diagram.json):



## Phần code sơ đồ diagram.json

```
{
  "version": 1,
  "author": "Duy",
  "editor": "wokwi",
  "parts": [
    { "type": "board-esp32-devkit-c-v4", "id": "esp", "top": 9.6, "left": 72.04,
    "attrs": {} },
    {
      "type": "wokwi-led",
      "id": "led1",
      "top": -30,
      "left": 448.2,
      "rotate": 90,
      "attrs": { "color": "red", "pin": "5" }
    },
    {
      "type": "wokwi-led",
      "id": "fan_led",
      "top": 171.6,
      "left": 496.2,
      "rotate": 90,
      "attrs": { "color": "blue", "pin": "4" }
    },
    { "type": "wokwi-dht22", "id": "dht", "top": -153.3, "left": 90.6, "attrs": {
    "pin": "15" } },
    {
      "type": "wokwi-photoresistor-sensor",
      "id": "ldr",
      "top": 12.8,
      "left": -258.4,
      "attrs": { "pin": "34" }
    },
    {

```

```

    "type": "wokwi-pushbutton-6mm",
    "id": "btn1",
    "top": 61.6,
    "left": 240.8,
    "rotate": 180,
    "attrs": { "color": "red", "xray": "1" }
  },
  {
    "type": "wokwi-pushbutton-6mm",
    "id": "btn2",
    "top": 161,
    "left": 278.4,
    "attrs": { "color": "green", "xray": "1" }
  },
  {
    "type": "wokwi-resistor",
    "id": "r1",
    "top": 196.25,
    "left": 219.4,
    "rotate": 180,
    "attrs": { "value": "10000" }
  },
  {
    "type": "wokwi-resistor",
    "id": "r4",
    "top": -34.45,
    "left": 364.8,
    "attrs": { "value": "220" }
  },
  {
    "type": "wokwi-resistor",
    "id": "r2",
    "top": 167.15,
    "left": 422.4,
    "attrs": { "value": "220" }
  }
],
"connections": [
  [ "esp:TX", "$serialMonitor:RX", "", [] ],
  [ "esp:RX", "$serialMonitor:TX", "", [] ],
  [ "dht:VCC", "esp:3V3", "red", [] ],
  [ "dht:GND", "esp:GND", "black", [] ],
  [ "dht:DATA", "esp:15", "green", [] ],
  [ "ldr:VCC", "esp:3V3", "red", [ "v-28.8", "h67.2", "v38.4" ] ],
  [ "ldr:GND", "esp:GND", "black", [] ],
  [ "ldr:GND", "esp:GND.1", "black", [ "h48", "v76.4" ] ],
  [ "dht:GND", "esp:GND.2", "black", [ "h67.2", "v105.6" ] ],
  [ "dht:SDA", "esp:15", "green", [ "v19.2", "h220.9", "v240", "h-134.4", "v-38.4" ] ],
  [ "ldr:A0", "esp:34", "green", [ "v28.1", "h76.8", "v-9.6" ] ],
  [ "esp:15", "r1:2", "green", [ "h9.6", "v-9.6", "h37.2" ] ],

```

```

[ "r1:1", "esp:3V3", "green", [ "v28.8", "h-240", "v-182.4", "h38.25" ] ],
[ "btn1:1.1", "esp:GND.2", "green", [ "h19.2", "v-61.1", "h-115.2" ] ],
[ "btn1:2.r", "esp:19", "green", [ "h-20", "v38" ] ],
[ "led1:A", "esp:22", "green", [ "h-144", "v-38.4" ] ],
[ "led1:C", "r4:2", "green", [ "v-18.8", "h39.6" ] ],
[ "r4:1", "esp:GND.2", "green", [ "h-96", "v38.4", "h-96" ] ],
[ "btn2:1.1", "esp:GND.3", "green", [ "h-86.4", "v-67.2" ] ],
[ "btn2:2.r", "esp:18", "green", [ "h10.4", "v-47.6" ] ],
[ "fan_led:A", "esp:23", "green", [ "h-67.2", "v-124.8" ] ],
[ "fan_led:C", "r2:2", "green", [ "h0" ] ],
[ "r2:1", "esp:GND.3", "green", [ "v-38.4", "h-48", "v9.6" ] ]
],
"dependencies": {}
}

```

### Phần code hoàn chỉnh Main.cpp:

```

                                                                    #define
BLYNK_TEMPLATE_ID "TMPL6hxMh1QwQ"
#define BLYNK_TEMPLATE_NAME "ESP32 HOME CONTROL"
#define BLYNK_AUTH_TOKEN "7VA3jiWeydomttvT0DW0KufdbPS6WJ83"
#define BLYNK_PRINT Serial


#include <WiFi.h>
#include <WiFiClientSecure.h>
#include <BlynkSimpleEsp32.h>
#include <DHT.h>
#include <ArduinoJson.h>
#include <UniversalTelegramBot.h>

// Wi-Fi
char ssid[] = "Wokwi-GUEST";
char password[] = "";

```

```

// Telegram
#define BOT_TOKEN "8044446031:AAGghQhARa6eiLzS9WLTyvBXoxVFlcY5oEc"
#define CHAT_ID "-1002619419433"
WiFiClientSecure secured_client;
UniversalTelegramBot bot(BOT_TOKEN, secured_client);


// DHT
#define DHTPIN 15 //  Đúng theo sơ đồ là chân 15
#define DHTTYPE DHT22
DHT dht(DHTPIN, DHTTYPE);

// GPIO (chân theo sơ đồ thực tế ESP32 trên Wokwi)
#define LED_PIN 22 // LED đỏ (đèn)
#define FAN_PIN 23 // LED xanh (quạt)
#define BTN_LED_PIN 19 // Nút nhấn điều khiển đèn
#define BTN_FAN_PIN 18 // Nút nhấn điều khiển quạt
#define LDR_PIN 34 // Cảm biến ánh sáng (photoresistor)

bool ledState = false;
bool fanState = false;

// Virtual Pins
#define VPIN_LIGHT_CONTROL V0
#define VPIN_FAN_CONTROL V1
#define VPIN_LIGHT_SENSOR V2
#define VPIN_TEMP_SENSOR V3
#define VPIN_STATUS V4


BlynkTimer timer;

//  Gửi dữ liệu lên Blynk + Telegram
void sendStatus() {
    float temp = dht.readTemperature();
    int ldrValue = analogRead(LDR_PIN);
    int brightness = map(ldrValue, 0, 4095, 0, 100);

    Blynk.virtualWrite(VPIN_TEMP_SENSOR, temp);
    Blynk.virtualWrite(VPIN_LIGHT_SENSOR, brightness);

    String status = "🌡️ Nhiệt độ: " + String(temp, 1) + "°C\n" +
        "⚙️ Ánh sáng: " + String(brightness) + "%\n" +
        "💡 Đèn: " + String(ledState ? "BẬT" : "TẮT") + "\n" +
        "🌀 Quạt: " + String(fanState ? "BẬT" : "TẮT");

    Blynk.virtualWrite(VPIN_STATUS, status);
    bot.sendMessage(CHAT_ID, status, "");
    Serial.println("[LOG] " + status);
}

//  Xử lý lệnh Telegram

```

```

void handleTelegram() {
    int numNewMessages = bot.getUpdates(bot.last_message_received + 1);
    for (int i = 0; i < numNewMessages; i++) {
        String text = bot.messages[i].text;

        if (text == "/led_on") {
            ledState = true;
            digitalWrite(LED_PIN, HIGH);
            Blynk.virtualWrite(VPIN_LIGHT_CONTROL, HIGH);
            bot.sendMessage(CHAT_ID, "💡 Đèn đã BẬT từ Telegram.");
        } else if (text == "/led_off") {
            ledState = false;
            digitalWrite(LED_PIN, LOW);
            Blynk.virtualWrite(VPIN_LIGHT_CONTROL, LOW);
            bot.sendMessage(CHAT_ID, "💡 Đèn đã TẮT từ Telegram.");
        } else if (text == "/fan_on") {
            fanState = true;
            digitalWrite(FAN_PIN, HIGH);
            Blynk.virtualWrite(VPIN_FAN_CONTROL, HIGH);
            bot.sendMessage(CHAT_ID, "🌀 Quạt đã BẬT từ Telegram.");
        } else if (text == "/fan_off") {
            fanState = false;
            digitalWrite(FAN_PIN, LOW);
            Blynk.virtualWrite(VPIN_FAN_CONTROL, LOW);
            bot.sendMessage(CHAT_ID, "🌀 Quạt đã TẮT từ Telegram.");
        } else if (text == "/status") {
            sendStatus();
        }
    }
}

// 🛠 Xử lý nút nhấn vật lý
void checkButtons() {
    if (digitalRead(BTN_LED_PIN) == LOW) {
        delay(200);
        ledState = !ledState;
        digitalWrite(LED_PIN, ledState);
        Blynk.virtualWrite(VPIN_LIGHT_CONTROL, ledState);
        bot.sendMessage(CHAT_ID, "💡 Đèn đã " + String(ledState ? "BẬT" : "TẮT") + "
bằng nút.");
    }

    if (digitalRead(BTN_FAN_PIN) == LOW) {
        delay(200);
        fanState = !fanState;
        digitalWrite(FAN_PIN, fanState);
        Blynk.virtualWrite(VPIN_FAN_CONTROL, fanState);
        bot.sendMessage(CHAT_ID, "🌀 Quạt đã " + String(fanState ? "BẬT" : "TẮT") + "
bằng nút.");
    }
}

```

```

// 🚀 setup()
void setup() {
  Serial.begin(115200);
  delay(1000);
  dht.begin();

  pinMode(LED_PIN, OUTPUT);
  pinMode(FAN_PIN, OUTPUT);
  pinMode(BTN_LED_PIN, INPUT_PULLUP);
  pinMode(BTN_FAN_PIN, INPUT_PULLUP);

  secured_client.setInsecure(); // Cho phép HTTPS không kiểm chứng

  WiFi.begin(ssid, password);
  Serial.print("🔌 Kết nối WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\n✅ Đã kết nối WiFi!");

  configTime(0, 0, "pool.ntp.org"); // Đồng bộ thời gian cho Telegram

  Blynk.begin(BLYNK_AUTH_TOKEN, ssid, password);
  delay(2000);

  timer.setInterval(5000L, sendStatus);

  timer.setInterval(10000L, []() {
    if (!Blynk.connected()) {
      Serial.println("[⚠️] Mất kết nối Blynk. Đang kết nối lại...");
      Blynk.connect();
    }
  });
}

// 🔄 loop()
void loop() {
  if (WiFi.status() != WL_CONNECTED) {
    Serial.println("[⚠️] Mất WiFi! Đang kết nối lại...");
    WiFi.disconnect();
    WiFi.begin(ssid, password);
    delay(5000);
    return;
  }

  Blynk.run();
  timer.run();
  checkButtons();
  handleTelegram();
}

```

```
}
```

## 4.5. Cấu hình phần mềm

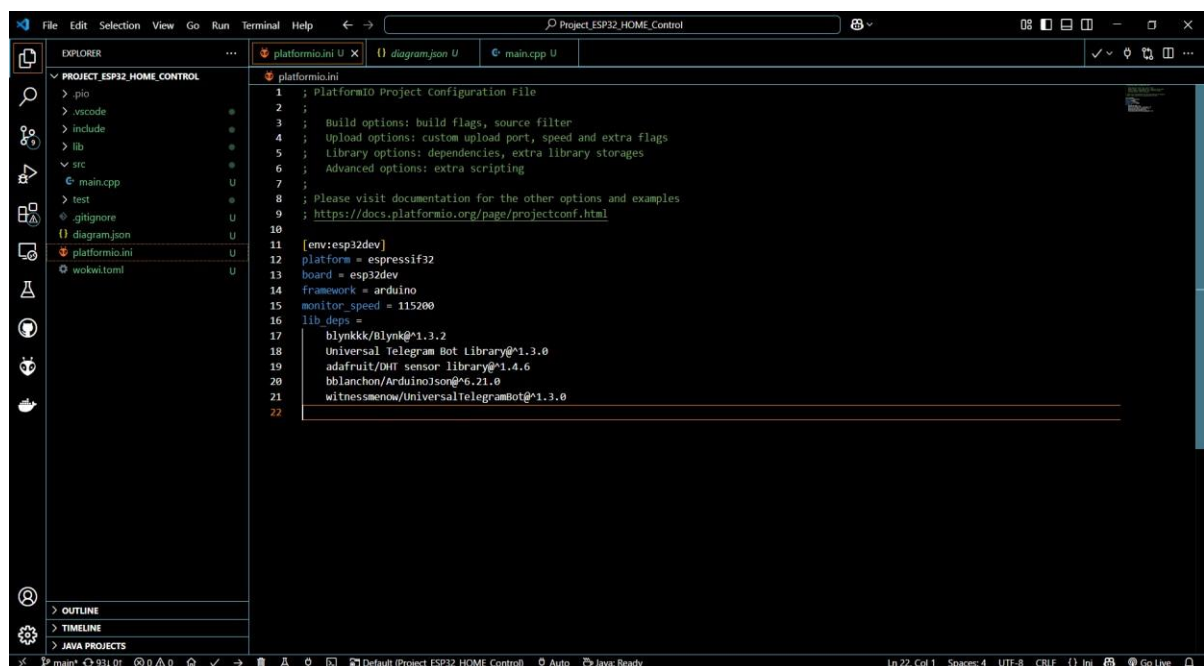
Để triển khai phần mềm trên ESP32, chúng ta sử dụng PlatformIO trong Visual Studio Code, kết hợp với các thư viện cần thiết như:

- Blynk: Quản lý kết nối và giao tiếp với ứng dụng Blynk.
- Telegram Bot Library: Gửi và nhận thông báo qua Telegram.
- DHT: Đọc giá trị từ cảm biến DHT22.
- ArduinoJson: Xử lý dữ liệu JSON từ các API.

Các cấu hình và thư viện này được chỉ định trong các tệp `platformio.ini` và `wokwi.toml` để đảm bảo phần mềm có thể chạy trên ESP32.

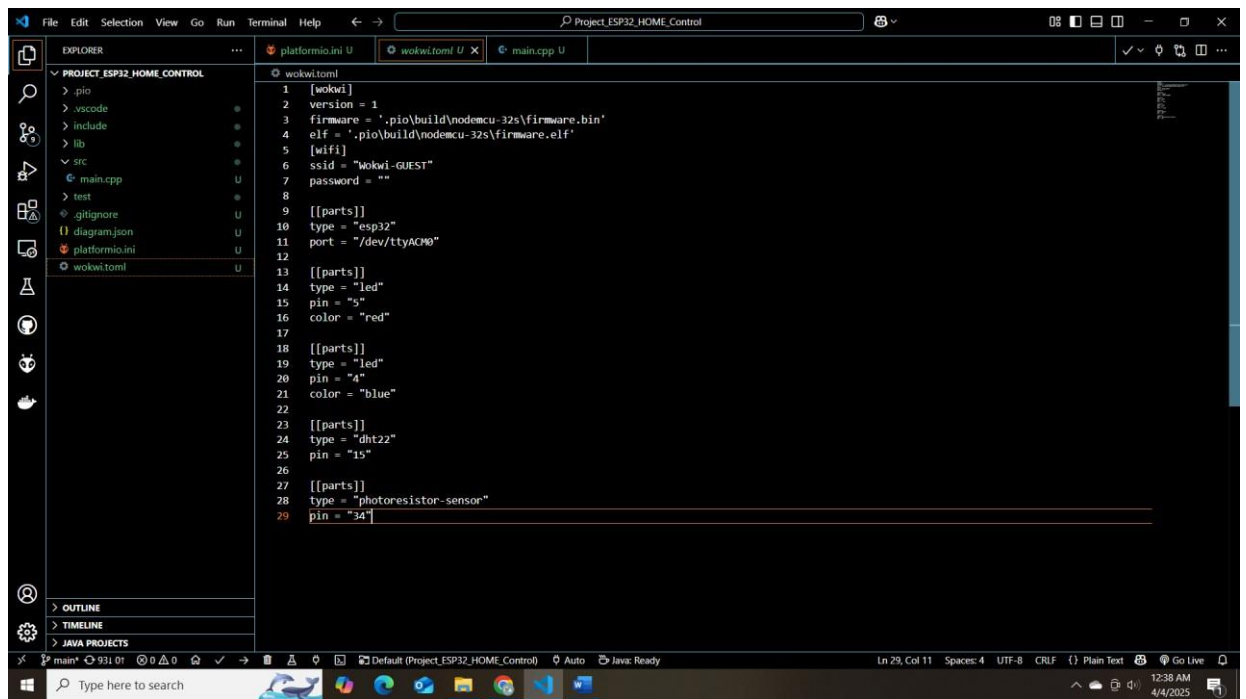
## 4.6. Cấu trúc tệp cấu hình

**Platformio.ini:**



```
1 ; PlatformIO Project Configuration File
2 ;
3 ; Build options: build flags, source filter
4 ; Upload options: custom upload port, speed and extra flags
5 ; Library options: dependencies, extra library storages
6 ; Advanced options: extra scripting
7 ;
8 ; Please visit documentation for the other options and examples
9 ; https://docs.platformio.org/page/projectconf.html
10
11 [env:esp32dev]
12 platform = espressif32
13 board = esp32dev
14 framework = arduino
15 monitor_speed = 115200
16 lib_deps =
17     blynkkk/Blynk@1.3.2
18     Universal Telegram Bot Library@1.3.0
19     adafruit/DHT sensor library@1.4.6
20     bblanchon/ArduinoJson@6.21.0
21     witnessmenow/UniversalTelegramBot@1.3.0
22
```

**Wokwi.toml:**



Hệ thống được triển khai dựa trên ESP32 kết hợp với Blynk và Telegram để điều khiển thiết bị và giám sát cảm biến từ xa. Qua đó, người dùng có thể theo dõi và điều khiển các thiết bị trong nhà thông qua ứng dụng di động hoặc Telegram một cách dễ dàng và tiện lợi. Việc sử dụng HTTP trong giao tiếp giữa ESP32, Blynk và Telegram giúp đơn giản hóa quá trình truyền tải dữ liệu mà không cần phải thiết lập một máy chủ phức tạp.

## 5. Triển khai thực tế

Do điều kiện hạn chế về thiết bị phần cứng, hệ thống đã được triển khai trên nền tảng mô phỏng Wokwi thay vì sử dụng phần cứng thực tế. Wokwi là một công cụ mô phỏng mạnh mẽ cho các vi điều khiển như ESP32, giúp kiểm tra và thử nghiệm hệ thống mà không cần thiết bị vật lý. Dưới đây là các bước triển khai và kết quả mô phỏng:

### 5.1. Cài đặt môi trường mô phỏng

- Wokwi được sử dụng để mô phỏng ESP32 và các cảm biến (DHT22 và LDR), cùng với các thiết bị ngoại vi như LED và quạt(thay thế bằng LED\_FAN)



- Các phần cứng được cấu hình qua giao diện Wokwi, với các chân GPIO của ESP32 được kết nối đúng với các thiết bị tương ứng (LED, cảm biến ánh sáng, cảm biến nhiệt độ).
- Cấu hình Wi-Fi và kết nối mạng được thực hiện thông qua tệp cấu hình wokwi.toml, cho phép ESP32 mô phỏng kết nối vào mạng và giao tiếp với Blynk và Telegram.

## 5.2. Triển khai phần mềm

- Các tệp cấu hình và mã nguồn đã được tải lên Wokwi để mô phỏng môi trường hoạt động của ESP32. Tại đây, mã nguồn Arduino cho ESP32 đã được biên dịch và tải vào mô phỏng.
- Các thư viện cần thiết (Blynk, DHT, Telegram Bot, ArduinoJson) đã được cài đặt và cấu hình trong PlatformIO và Wokwi.

## 5.3. Quá trình kiểm tra

Trong quá trình mô phỏng, các chức năng chính của hệ thống đã được kiểm tra:

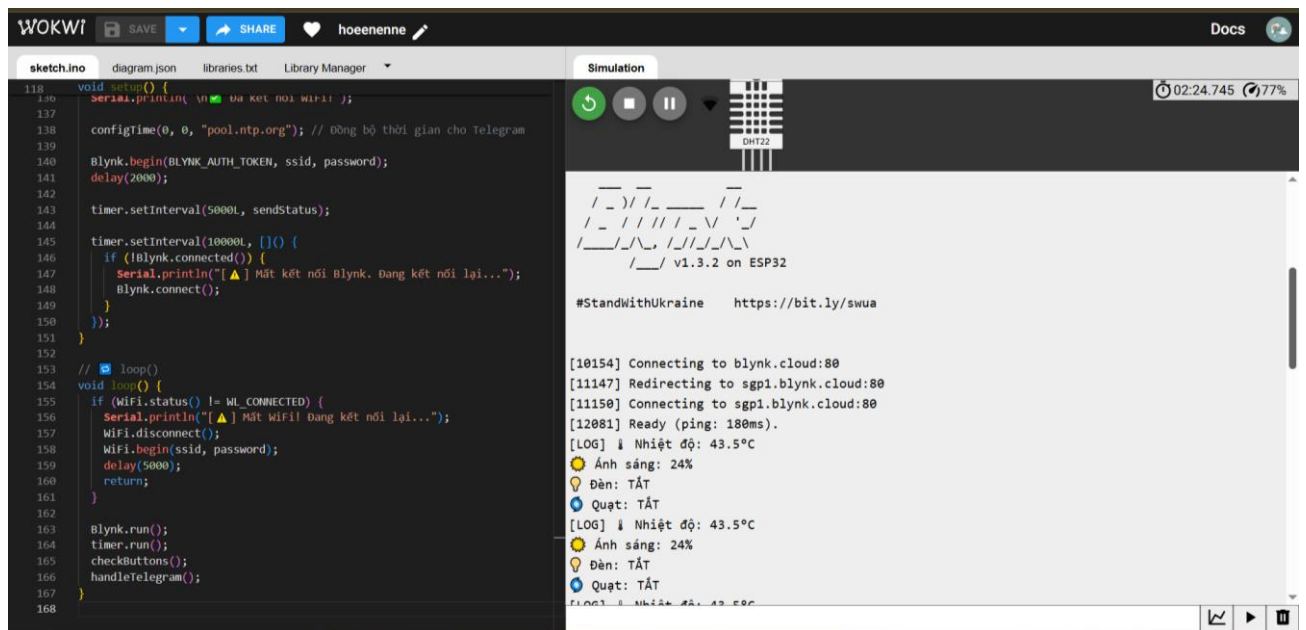
- Điều khiển đèn và quạt: Blynk đã được sử dụng để bật/tắt đèn LED và quạt, xác nhận rằng các tín hiệu được truyền đi và nhận đúng qua giao thức HTTP.
- Giám sát cảm biến: Các giá trị nhiệt độ và độ ẩm từ cảm biến DHT22 và độ sáng từ cảm biến LDR đã được hiển thị chính xác trên ứng dụng Blynk.
- Giao tiếp với Telegram: Hệ thống đã gửi thông báo qua Telegram khi các điều kiện nhiệt độ vượt ngưỡng, và cho phép điều khiển bật/tắt đèn và quạt qua Telegram.

## 5.4. Kết quả mô phỏng

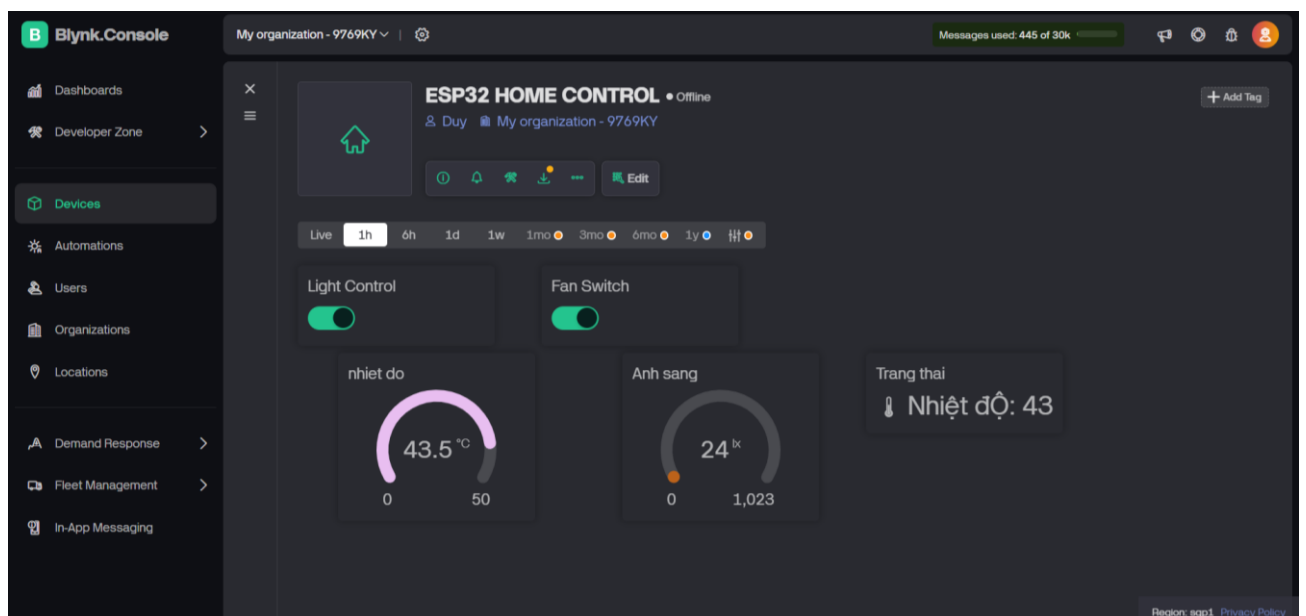
- Hoạt động ổn định: Toàn bộ hệ thống hoạt động chính xác trong môi trường mô phỏng, với khả năng điều khiển thiết bị từ xa qua Blynk và Telegram.
- Giao tiếp HTTP: Các yêu cầu HTTP được gửi và nhận đúng cách, hệ thống có thể tương tác với các dịch vụ Blynk và Telegram một cách hiệu quả.

- Độ chính xác cảm biến: Các giá trị cảm sensor DHT22 và LDR được hiển thị chính xác trên ứng dụng Blynk và có thể sử dụng chúng để đưa ra các quyết định điều khiển thiết bị.

### Hình ảnh LOG trên Serial Monitor

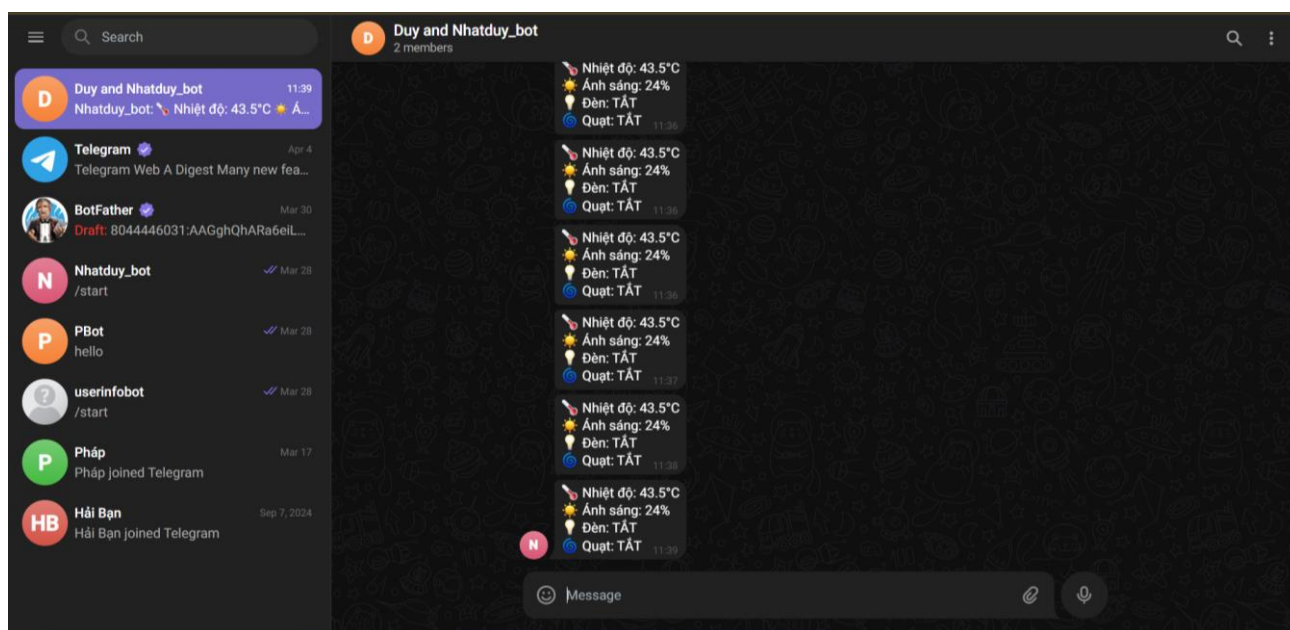


### Hình ảnh trên web Blynk



### Hình ảnh trong Blynk App(Mobile)

## Hình ảnh trên Telegram



### 5.5. Những thách thức và hạn chế

Mặc dù hệ thống đã hoạt động tốt trong môi trường mô phỏng, việc triển khai trên phần mềm có thể gặp phải một số thách thức, bao gồm:

- Kết nối Wi-Fi không ổn định: Mặc dù mô phỏng hoạt động tốt, kết nối Wi-Fi có thể gặp sự cố khi kết nối.
- Độ chính xác cảm biến: Các cảm biến có thể có độ chính xác khác nhau khi được sử dụng trên phần cứng thực tế.
- Giao tiếp Telegram: Việc kết nối và giao tiếp với Telegram có thể gặp phải sự cố nếu không cấu hình đúng.

# KẾT LUẬN

Trong thời đại công nghệ phát triển mạnh mẽ, Internet of Things (IoT) đã và đang trở thành xu hướng quan trọng, ứng dụng rộng rãi trong nhiều lĩnh vực, từ tự động hóa, y tế, đến giao thông và đời sống hằng ngày. Qua quá trình nghiên cứu và thực hiện tiểu luận, em đã tìm hiểu về nguyên lý hoạt động, kiến trúc hệ thống IoT, cũng như cách kết nối và điều khiển thiết bị từ xa thông qua nền tảng Blynk và ESP32.

Trong tiểu luận này, em đã xây dựng thành công một hệ thống điều khiển và giám sát từ xa sử dụng ESP32 kết hợp với các cảm biến như DHT22, LDR và các thiết bị điều khiển như đèn LED, quạt(LED\_FAN). Hệ thống không chỉ giúp người dùng giám sát các thông số môi trường mà còn có thể điều khiển các thiết bị thông qua ứng dụng Blynk trên web và điện thoại. Điều này minh chứng cho tiềm năng to lớn của IoT trong việc cải thiện chất lượng cuộc sống và tối ưu hóa hiệu suất hoạt động trong nhiều lĩnh vực.

Mặc dù đã cố gắng hoàn thành tiểu luận một cách tốt nhất, nhưng do thời gian nghiên cứu và thực nghiệm còn hạn chế, chắc chắn không thể tránh khỏi những thiếu sót. Em rất mong nhận được sự góp ý từ thầy cô để có thể hoàn thiện hơn kiến thức và ứng dụng IoT trong thực tế.

Cuối cùng, Em xin chân thành cảm ơn thầy Võ Việt Dũng đã tận tình hướng dẫn và hỗ trợ trong quá trình thực hiện bài tiểu luận này. Những kiến thức tích lũy được không chỉ giúp em hiểu sâu hơn về IoT mà còn là nền tảng quan trọng cho các dự án nghiên cứu và ứng dụng thực tế trong tương lai.

# TÀI LIỆU THAM KHẢO

## **Tài liệu học lập trình IoT với ESP32 và Arduino**

Nguyễn Văn Hiếu. *Hướng dẫn lập trình ESP32 với Arduino IDE*. NXB Khoa học

Kỹ thuật, 2021. <https://www.youtube.com/watch?v=QsoDg-bfD44&t=30s>

<https://viettelidc.com.vn/tin-tuc/iot-la-gi-nhung-ung-dung-noi-bat-cua-iot>

TRƯỜNG ĐẠI HỌC KHOA HỌC      CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM  
KHOA CÔNG NGHỆ THÔNG TIN      Độc lập – Tự do – Hạnh phúc

---

**PHIẾU ĐÁNH GIÁ TIỂU LUẬN**

Học kỳ I Năm học 2021-2022

Cán bộ chấm thi 1	Cán bộ chấm thi 2
Nhận xét: .....	
33	

.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....

Điểm đánh giá của CBChT1:

Điểm đánh giá của CBChT2:

Bảng số: .....

Bảng số: .....

Bảng chữ: .....

Bảng chữ: .....

Nhận xét: .....

Điểm kết luận: Bảng số..... Bảng chữ:.....

*Thừa Thiên Huế, ngày .... tháng .... năm*

20... CBChT1

CBChT2

*(Ký và ghi rõ họ tên)*

*(Ký và ghi rõ họ tên)*