**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**CƠ SỞ TẠI THÀNH PHỐ HỒ CHÍ MINH**

**KHOA KĨ THUẬT ĐIỆN TỬ II**

**----🙠🕮🙢----**

**A logo with a yellow star

Description automatically generated**

**BÁO CÁO TỔNG KẾT**

**ỨNG DỤNG IOT ĐIỀU KHIỂN THIẾT BỊ CHIẾU SÁNG THEO THỜI GIAN THỰC**

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**CƠ SỞ TẠI THÀNH PHỐ HỒ CHÍ MINH**

**KHOA KĨ THUẬT ĐIỆN TỬ II**

**----🙠🕮🙢----**

**A logo with a yellow star

Description automatically generated**

**BÁO CÁO TỔNG KẾT**

**ỨNG DỤNG IOT ĐIỀU KHIỂN THIẾT BỊ CHIẾU SÁNG THEO THỜI GIAN THỰC**

**Sinh viên thực hiện:**

LÂM CÔNG TRUYỀN - N21DCDT096

ĐOÀN QUANG HẢI - N21DCDT027

NGUYỄN ĐỨC TƯỜNG - N21DCDT099

HOÀNG TRỌNG QUYỀN - N21DCDT078

**Lớp*:*** D21CQDT01-N **Ngành:** Điện - Điện tử

**Giảng viên hướng dẫn:** TRẦN ĐÌNH ĐẠT

**LỜI CẢM ƠN**

Lời đầu tiên nhóm xin được gửi lời cảm ơn đến thầy Trần Đình Đạt đã đưa ra yêu cầu hoàn thành môn học một cách đầy đủ và hợp lý. Đồng thời, nhóm cũng rất cảm ơn về quyết định của thầy là đồng ý cho phép nhóm thực hiện đề tài có tên “Ứng dụng IOT điều khiển thiết bị chiếu sáng theo thời gian thực”.

Bên cạnh đó, nhóm cũng cảm ơn các nhóm khác trong lớp học đã hỗ trợ nhóm nghiên cứu trong quá trình học tâp.

Với kinh nghiệm còn hạn chế nên dự án nghiên cứu và bài báo cáo không tránh khỏi sai sót. Vì thế nhóm mong nhận được sự nhận xét và góp ý tận tình của thầy và cá nhóm khác.

Xin chân thành cảm ơn.

**MỤC LỤC**

[MỤC LỤC iv](#_Toc187598113)

[DANH MỤC HÌNH ẢNH vii](#_Toc187598114)

[DANH MỤC BẢNG BIỂU viii](#_Toc187598115)

[CHƯƠNG 1 TỔNG QUAN VỀ HỆ THỐNG 1](#_Toc187598116)

[1.1 Giới thiệu đề tài 1](#_Toc187598117)

[1.2 Tổng quan về hệ thống 1](#_Toc187598118)

[1.3 Lịch sử vấn đề 2](#_Toc187598119)

[CHƯƠNG 2 MỤC TIÊU DỰ ÁN 3](#_Toc187598120)

[2.1 Các tính năng hệ thống được triển khai 3](#_Toc187598121)

[2.2 Các trường hợp sử dụng 5](#_Toc187598122)

[CHƯƠNG 3 THIẾT KẾ PHẦN CỨNG 8](#_Toc187598123)

[3.1 Sơ đồ khối hệ thống 8](#_Toc187598124)

[3.2 Sơ đồ kết nối phần cứng 10](#_Toc187598125)

[3.3 Danh mục các linh kiện, vật tư được sử dụng 11](#_Toc187598126)

[3.4 Giới thiệu về các module chính 12](#_Toc187598127)

[3.5 Hình ảnh sản phẩm thực tế 16](#_Toc187598128)

[CHƯƠNG 4 THIẾT KẾ PHẦN MỀM 18](#_Toc187598129)

[4.1 Tổng quan 18](#_Toc187598130)

[4.1.1 Giới thiệu nền tảng tạo chương trình nhúng STM32 18](#_Toc187598131)

[4.1.2 Giới thiệu nền tảng tạo chương trình nhúng ESP32 18](#_Toc187598132)

[4.1.3 Giới thiệu nền tảng tạo chương trình ứng dụng 19](#_Toc187598133)

[4.1.4 Giới thiệu phần mềm mô phỏng sơ đồ mạch điện 21](#_Toc187598134)

[4.1.5 Giới thiệu phần mềm thiết kế hộp mica bảo vệ thiết bị 23](#_Toc187598135)

[4.2 Sơ đồ thiết kế phần mềm 23](#_Toc187598136)

[4.3 Thực thi các module phần mềm 25](#_Toc187598137)

[4.3.1 Hằng số và biến toàn cục chương trình STM32 25](#_Toc187598138)

[4.3.2 Hằng số và biến toàn cục chương trình ứng dụng 29](#_Toc187598139)

[4.3.3 Hằng số và biến toàn cục chương trình ESP32 30](#_Toc187598140)

[CHƯƠNG 5 KIỂM TRA HOẠT ĐỘNG CỦA HỆ THỐNG 33](#_Toc187598141)

[5.1 Đánh giá việc thực hiện các tính năng của hệ thống 33](#_Toc187598142)

[5.2 Đánh giá các trường hợp sử dụng 33](#_Toc187598143)

[5.3 Nhận xét 33](#_Toc187598144)

[KẾT LUẬN 34](#_Toc187598145)

[PHỤ LỤC 35](#_Toc187598146)

[Code STM32CubeIDE 35](#_Toc187598147)

[Code Arduino IDE- platform MIO 58](#_Toc187598148)

[Code chương trình ứng dụng ANDROID 72](#_Toc187598149)

[Code chương trình ứng dụng ANDROID 72](#_Toc187598150)

**NHẬN XÉT**

# **DANH MỤC HÌNH ẢNH**

[Hình 3.1 Sơ đồ khối tổng quát 8](#_Toc187597177)

[Hình 3.2 Sơ đồ khối cụ thể 9](#_Toc187597178)

[Hình 3.3 Sơ đồ kết nối phần cứng 10](#_Toc187597179)

[Hình 3.4 STM32F103C8T6 12](#_Toc187597180)

[Hình 3.5 ESP-WROOM-32 13](#_Toc187597181)

[Hình 3.6 Module PWM Mosfet 15A 14](#_Toc187597182)

[Hình 3.7 LCD 20x4 tích hợp module I2C 15](#_Toc187597183)

[Hình 3.8 Tổng hợp hình ảnh thực tế sản phẩm 16](#_Toc187597184)

[Hình 3.9. Mô hình lắp đặt thiết bị chiếu sáng 17](#_Toc187597185)

[Hình 3.10. Tổng hợp phần cứng 17](#_Toc187597186)

[Hình 4.1 STM32CubeIDE 18](#_Toc187597187)

[Hình 4.2 PlatformIO IDE 19](#_Toc187597188)

[Hình 4.3 Phần mềm Android Studio 19](#_Toc187597189)

[Hình 4.4 Giao diện đăng nhập ứng dụng 20](#_Toc187597190)

[Hình 4.5 Giao diện điều khiển trung tâm 20](#_Toc187597191)

[Hình 4.6 Giao diện điều khiển RGB 21](#_Toc187597192)

[Hình 4.7 Giao diện hẹn giờ 21](#_Toc187597193)

[Hình 4.8 Phần mềm Altium 22](#_Toc187597194)

[Hình 4.9 2D PCB 22](#_Toc187597195)

[Hình 4.10 3D PCB 22](#_Toc187597196)

[Hình 4.11 Phần mềm Inkscape 23](#_Toc187597197)

[Hình 4.12 Giao diện thiết kế hộp bảo vệ 23](#_Toc187597198)

[Hình 4.13 Lưu đồ giải thuật chương trình nhúng (STM32) 24](#_Toc187597199)

[Hình 4.14 Lưu đồ giải thuật chương trình nhúng (ESP32) 24](#_Toc187597200)

[Hình 4.15 Lưu đồ giải thuật chương trình ứng dụng 25](#_Toc187597201)

# **DANH MỤC BẢNG BIỂU**

[Bảng 2.2 Trường hợp 2 6](#_Toc187597545)

[Bảng 2.3 Trường hợp 3 6](#_Toc187597546)

[Bảng 2.4 Trường hợp 4 7](#_Toc187597547)

[Bảng 2.5 Trường hợp 5 7](#_Toc187597548)

[Bảng 2.6 Trường hợp 6 8](#_Toc187597549)

[Bảng 3.1 Chân kết nối LCD gắn I2C 10](#_Toc187597550)

[Bảng 3.2 Chân kết nối Module PWM 11](#_Toc187597551)

[Bảng 3.3 Chân kết nối Button 11](#_Toc187597552)

[Bảng 3.4 Chân kết nối biến trở 11](#_Toc187597553)

[Bảng 3.5 Các linh kiện được sử dụng 11](#_Toc187597554)

# **CHƯƠNG 1 TỔNG QUAN VỀ HỆ THỐNG**

## **1.1 Giới thiệu đề tài**

Trong thực tế khoa học công nghệ phát triển mạnh mẽ, nhu cầu của con người về các thiết bị thông minh hỗ trợ đời sống cũng ngày càng cao. Với mong muốn tạo nên được một sản phẩm giá thành rẻ có tính ứng dụng thực tiễn vào cuộc sống, vận dụng những kiến thức đã được học ở môn “Vi sử lý” và “Thiết kế mạch điện tử” nhóm đã lên ý tưởng và thực hiện “Ứng dụng IOT (Internet of Thing) điều khiển thiết bị chiếu sáng theo thời gian thực”. Đồng thời dự án môn học này cũng là cách giải quyết cho “giải pháp chiếu sáng mô hình nhà thông minh” đóng góp vào mô hình tổng thể mà thành viên nhóm đang xây dựng. Nhóm hi vọng sản phẩm của ý tưởng này giúp ích được vào cuộc sống và truyền thêm động lực cho các thành viên trong quá trình học tập.

## **1.2 Tổng quan về hệ thống**

Hệ thống gồm có các khối sau: khối xử lí, khối thực thi, khối sever và khối app. Quan trọng nhất là khối điều khiển trung tâm, có nhiệm vụ điều khiển toàn hệ thống. Nhóm lựa chọn vi điều khiển chính là STM32 và vi điều khiển thực hiện nhiệm vụ mạng là ESP32, vì các ưu điểm phù hợp với nhu cầu sau: giá rẻ, cộng đồng hỗ trợ lớn, dễ lập trình. Về phần mềm, để thực hiện lập trình cho STM32 nhóm lựa chọn phần mềm CubeIDE vì trước đây ở môn “Vi sử lý” các thành viên trong nhóm đã được tiếp cận qua phần mềm này. Về phần lập trình cho vi điều khiển ESP32, nhóm chọn PlatformMIO (là một plugin có khả năng lập trình Arduino) nhận thấy rằng tốc độ nạp chương trình nhanh và giao diện hiển thị thuận tiện hơn trong quá trình làm việc cũng như sửa lỗi cho nên đây là lựa chọn để thực hiện đề tài này.

Các khối liên quan khác:

+Khối sever: cơ sở dữ liệu thời gian thực Firebase.

+Khối app: ứng dụng android được thiết kế dựa trên phần mềm Android Studio.

+Khối thực thi: màn hình LCD20x4 có kết nối I2C và 2 module PWM

## **1.3 Lịch sử vấn đề**

Khảo sát các đề tài liên quan đến hệ thống IOT ứng dụng trong nhà ở đã có nhiều tác giả đưa ra các sản phẩm nghiên cứu đạt được yêu cầu của thị trường. Tuy nhiên ở đề tài mà nhóm đang thực hiện có những ưu điểm riêng biệt mà tôi cho rằng nó là cải thiện ưu việt của đề tài này đó là:

* Cải thiện tốt tốc độ điều khiển: tốc độ truyền nhận thông tin được thực hiện nhanh chóng ở thời gian thực.
* Ứng dụng android riêng biệt: với giao diện app ứng dụng có thể được tùy chỉnh dựa vào yêu cầu phát triển của đề tài mà không phụ thuộc vào đơn vị nào khác.
* Có màn hình hiển thị đầy đủ thông báo, trạng thái điều khiển.
* Dễ dàng áp dụng ngay trong các hộ gia đình để điều khiển các thiết bị chiếu sáng trong gia đình mà không giới hạn về khoảng cách.
* Giá thành sản phẩm của dự án (nếu sản xuất số lượng lớn và được thiết kế lại một số thứ như: PCB, hộp bảo vệ) có cải thiện so với các sản phẩm hiện tại có bán trên thị trường, thậm chí chức năng của dự án này có phần thông minh và dễ dàng cá nhân hóa theo yêu cầu của người dùng hơn

Song, đề tài cũng có những giới hạn khi triển khai: việc điều khiển bằng phần cứng với tần số nhanh có thể làm cho hệ thống bị gián đoạn chương trình vì tốc độ thực hiện các yêu cầu mạng trên ESP32 còn hạn chế, việc điều khiển chưa có chế độ offline nên phụ thuộc nhiều vào tình trạng Intenet hiện hành trong gia đình.

# **CHƯƠNG 2 MỤC TIÊU DỰ ÁN**

Dự án “Ứng dụng IOT điều khiển thiết bị chiếu sáng theo thời gian thực” được thực hiện nhằm mục đích thiết kế và triển khai hệ thống có các tính năng và thực hiện được các trường hợp như sau:

## **2.1 Các tính năng hệ thống được triển khai**

a) Chế độ hoạt động

Hệ thống có thể điều khiển trên cả phần cứng, phần mềm và được đồng bộ trạng thái với nhau, các trạng thái này được lưu trữ ở CSDL thời gian thực Firebase có thể giữ trạng thái cuối cùng các lựa chọn trong cả trường hợp phần cứng bị ngắt kết nối mạng hoặc nguồn điện.

b) Ứng dụng Android

Hệ thống có ứng dụng hỗ trợ điều khiển riêng biệt được thành viên nhóm tự phát triển. Với ngôn ngữ Java và phần mềm Android Studio, nhóm đã tạo một ứng dụng có chức năng cơ bản được đồng bộ với Firebase. Ứng dụng có giao diện đăng nhập yêu cầu gmail và mật khẩu, tài khoản đăng nhập này được tạo ra dựa trên Firebase và sẽ cấp kèm với sản phẩm cho người sử dụng. Firebase cũng hỗ trợ tạo nhiều tài khoản cho phép truy cập đồng thời vào Realtime DataBase, điều này giúp cho sản phẩm của dự án có thể phát triển với số lượng người dùng lớn.

c) Cơ sở dữ liệu thời gian thực

Firebase là dịch vụ cơ sở dữ liệu hoạt động trên nền tảng đám mây – cloud. Kèm theo đó là hệ thống máy chủ cực kỳ mạnh mẽ của Google. Chức năng chính là giúp người dùng lập trình ứng dụng bằng cách đơn giản hóa các thao tác với cơ sở dữ liệu. Khi tạo thành công một cơ sở dư liệu trên Firebase, sau khi cấp quyền đọc, ghi đã có một “sever” để lưu trữ dữ liệu của mình. Đồng thời nó cũng hỗ trợ đồng bộ thời gian thực đến các kết nối client trong dự án nếu muốn phát triển với số lượng người dùng lớn. Dù hệ thống có bị mất mạng, mất nguồn thì dữ liệu vẫn được sao lưu ở local của nền tảng này. Tuy nhiên, Firebase không có nhiều cơ sở dữ liệu ở nhiều quốc gia, trong dự án nhóm chọn máy chủ đặt ở Singapore (duy nhất ở Đông Nam Á) vì khoảng cách khá xa cho nên yêu cầu tốc độ xử lí phải nhanh trong khi tốc độ ESP32 lại hạn chế.

d) Hiển thị

Sử dụng màn hình LCD 20x4 hiển thị đầy đủ trạng thái các nút nhấn, chế độ điều khiển, công suất tiêu thụ điện năng, thời gian được người sử dụng cài đặt hẹn giờ.

e) Nguồn điện

Các thiết bị trong hệ thống được cấp nguồn 5V từ nguồn adapter 5V 2A. Theo lý thuyết các thiết bị trong hệ thống (STM32, ESP32, Module điều khiển PWM, LCD) yêu cầu dòng tối đa khoảng 800mA, tuy nhiên theo thực tế triển khai trước đó thì nguồn 5V 1A hầu như không đáp ứng đủ và gây ra tình trạng sụt áp trên ESP, nên nhóm đã thay đổi thành nguồn 5V 2A để đủ cho yêu cầu dòng điện cao khi ESP32 hoạt động ở chế độ Wifi và thời gian hoạt động liên tục của STM32.

f) Hộp bảo vệ

Các thiết bị trong hệ thống được kết nối và lắp đặt trong một hộp bảo vệ làm bằng mica. Hộp bảo vệ được thiết kế để chứa các linh kiện một cách cố định, đảm bảo tính gọn gàng. Ngoài ra, cũng được bố trí các cổng trống để dễ dàng đi dây vào ra, thuận lợi cho việc bảo trì và sử dụng.

h) Thời gian thực hiện và chi phí

Hoàn thành sau 3 tháng kể từ lúc thời gian bắt đầu dự án. Với bảng chi tiết chi phí cho các linh kiện như bên dưới:

Thiết bị mua:

* Hộp bảo vệ mica: 45.000 VND
* Module I2C thay thế: 18.000 VND
* 3 Nút nhấn, 2 biến trở vặn: 16.000 + 4.000 tổng 20.000 VND
* Module nguồn 3.3V AMS1117: 5.000 VND
* 2 Module điều khiển PWM Mosfet 15A: 26.000 VND
* Các bus cắm và dây điện: 20.000 VND
* Dụng cụ làm mạch PCB: 50.000 VND
* Formex làm mô hình: 40.000 VND

Thiết bị có sẵn:

* Vi điều khiển: ESP32-Wroom-32, STM32F103C8T6
* LCD 20x4 kèm Module I2C
* Dây cắm USB
* Adapter và Jack DC 5V

Tổng giá trị sản phẩm (không bao gồm các thiết bị có sẵn): 206.000 VND

## **2.2 Các trường hợp sử dụng**

**Trường hợp 1:** Hiển thị trạng thái hai đèn và thời gian cài đặt hẹn giờ

Bảng 2.1 Trường hợp 1

|  |  |
| --- | --- |
| Yếu tố | Định nghĩa |
| Kích hoạt | Hệ thống được cấp nguồn đúng yêu cầu. |
| Điều kiện tiên quyết | Chương trình được nạp sẵn và phần cứng được thi công chính xác. |
| Luồng cơ bản | Trạng thái cuối cùng các nút nhấn, chế độ điều khiển, thời gian cài đặt hẹn giờ được lưu ở server sau đó hiển thị lên màn hình LCD và cập nhật giá trị khi có thay đổi lên ứng dụng. |
| Luồng thay thế | 1.Người dùng chuyển đổi chế độ điều khiển độ sáng (trường hợp 2).  2.Người dùng kích hoạt nút nhấn hoặc điều khiển núm vặn (trường hợp 3). |

**Trường hợp 2:** Chuyển đổi chế độ điều khiển độ sáng

**Bảng 2.2 Trường hợp 2**

|  |  |
| --- | --- |
| Yếu tố | Định nghĩa |
| Kích hoạt | Nút nhấn chuyển chế độ được kích hoạt |
| Điều kiện tiên quyết | Hệ thống được cấp nguồn đúng yêu cầu và phần cứng được thi công chính xác. |
| Luồng cơ bản | Hệ thống cập nhật hiển thị chế độ điều khiển độ sáng lên LCD và cho phép chuyển đổi cách thức điều khiển từ phần cứng thành ứng dụng và ngược lại. |
| Luồng thay thế | Trường hợp 3.  Trường hợp 4. |

**Trường hợp 3:** Người dùng kích hoạt nút nhấn hoặc điều khiển núm vặn (phần cứng)

****Bảng 2.3 Trường hợp 3****

|  |  |
| --- | --- |
| Yếu tố | Định nghĩa |
| Kích hoạt | Người dùng kích hoạt nút nhấn “BTN1, BTN2” hoặc điều chỉnh núm vặn “VR1, VR2” trên phần cứng. |
| Điều kiện tiên quyết | Hệ thống được cấp nguồn đúng yêu cầu và phần cứng được thi công chính xác. |
| Luồng cơ bản | Các trạng thái nút nhấn, độ sáng đèn sẽ được điều khiển bằng *phần cứng*, khi có thay đổi dữ liệu sẽ gửi tới Firebase, cập nhật LCD và được đồng bộ hai chiều. |
| Luồng thay thế | Trường hợp 4. |

**Trường hợp 4:** Người dùng kích hoạt nút nhấn hoặc điều chỉnh thanh seekbar trên ứng dụng tại giao diện điều khiển trung tâm.

**Bảng 2.4 Trường hợp 4**

|  |  |
| --- | --- |
| Yếu tố | Định nghĩa |
| Kích hoạt | Người dùng kích hoạt nút nhấn “BTN1, BTN2” hoặc điều chỉnh thanh seekbar 1 hoặc 2 trên ứng dụng. |
| Điều kiện tiên quyết | Hệ thống được cấp nguồn đúng yêu cầu, ứng dụng được tải về trên điện thoại. |
| Luồng cơ bản | Các trạng thái nút nhấn, độ sáng đèn sẽ được điều khiển bằng *ứng dụng* tại giao diện điều khiển trung tâm, khi có thay đổi dữ liệu sẽ gửi tới Firebase, cập nhật LCD và STM32 sẽ thực hiện điều khiển thiết bị. |
| Luồng thay thế | Trường hợp 5  Trường hợp 6. |

**Trường hợp 5:** Người dùng cài đặt thời gian hẹn giờ.

**Bảng 2.5 Trường hợp 5**

|  |  |
| --- | --- |
| Yếu tố | Định nghĩa |
| Kích hoạt | Người dùng lựa chọn chức năng “hẹn giờ” trên giao diện ứng dụng. |
| Điều kiện tiên quyết | Hệ thống được cấp nguồn đúng yêu cầu, ứng dụng được tải về trên điện thoại |
| Luồng cơ bản | Ứng dụng sẽ chuyển sang giao diện hẹn giờ, tại đây người dùng có thể lựa chọn thời gian và ghi chú để bật, tắt thiết bị. |
| Luồng thay thế | Hệ thống quay trở lại giao diện điều khiển trung tâm. |

**Trường hợp 6:** Người dùng lựa chọn màu sắc đèn RGB.

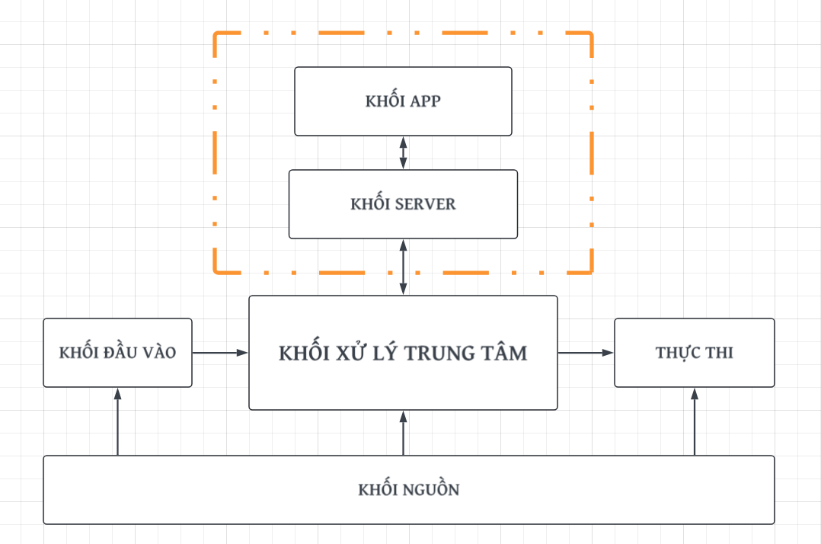
**Bảng 2.6 Trường hợp 6**

|  |  |
| --- | --- |
| Yếu tố | Định nghĩa |
| Kích hoạt | Người dùng lựa chọn chức năng “RGB” trên giao diện ứng dụng. |
| Điều kiện tiên quyết | Hệ thống được cấp nguồn đúng yêu cầu, ứng dụng được tải về trên điện thoại. |
| Luồng cơ bản | Ứng dụng sẽ chuyển sang giao diện lựa chọn màu RGB, tại đây người dùng có thể tự chọn màu sắc cho RGB hoặc lựa chọn các cài đặt có sẵn. |
| Luồng thay thế | Hệ thống quay trở lại giao diện điều khiển trung tâm. |

# **CHƯƠNG 3 THIẾT KẾ PHẦN CỨNG**

## **3.1 Sơ đồ khối hệ thống**

* Sơ đồ khối tổng quát:



**Hình 3.1 Sơ đồ khối tổng quát**

* Sơ đồ khối cụ thể:



**Hình 3.2 Sơ đồ khối cụ thể**

Nguyên lí hoạt động:

- Nguồn Adapter: cung cấp điện áp 5V 24/24 cho hệ thống hoạt động.

- Nút nhấn: gửi các tín hiệu digital khi người dùng kích hoạt các lựa chọn đến vi điều khiển STM32.

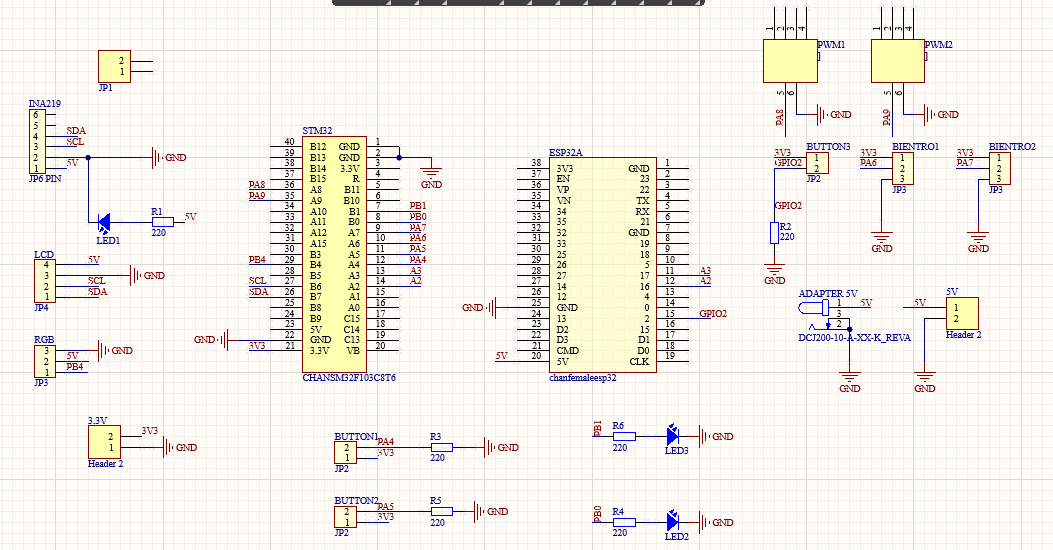
- Module PWM: 2 module pwm được kết nối với STM32 có nhiệm vụ bật tắt và điều chỉnh độ sáng hai đèn.

- Màn hình LCD 20x4 kết hợp module I2C: màn hình kết nối với STM32 qua giao thức I2C hiển thị các trạng thái nút nhấn, chế độ điều khiển độ sáng, công suất tiêu thụ và thông báo cho người dùng về thời gian đã được cài đặt hẹn giờ trước đó.

- STM32: sau khi tiếp nhận các thông tin từ cảm biến và nút nhấn, STM32 sẽ tiến hành xử lí thông tin, đưa ra tín hiệu điều khiển đến Module PWM và màn hình hiển thị LCD. Đồng thời gửi dữ liệu được gửi đến ESP32 sau đó dữ liệu được ESP32 đẩy lên cơ sở dữ liệu thời gian thực.

-ESP32: ngoài việc tiếp nhận dữ liệu từ STM32 và đẩy lên cơ sở dữ liệu, thì vi sử lý này còn lắng nghe dữ liệu ở CSDL (người dùng điều khiển bằng app ứng dụng) khi có thay đổi, dữ liệu sẽ cập nhật ngay lập và gửi đến cho STM32 thực thi. Chu trình đồng bộ hai chiều này được vi điều khiển sử lí nhanh chóng theo thời gian thực.

## **3.2 Sơ đồ kết nối phần cứng**

****

**Hình 3.3 Sơ đồ kết nối phần cứng**

Bảng trình bày các chân được kết nối với STM32:

**a) Màn hình LCD20x4**

**Bảng 3.1 Chân kết nối LCD gắn I2C**

|  |  |
| --- | --- |
| Module I2C | MCU |
| VCC | 5V |
| GND | GND |
| SCL | PB6 |
| SDA | PB7 |

**b) Module PWM**

**Bảng 3.2 Chân kết nối Module PWM**

|  |  |
| --- | --- |
| VCC | 12V |
| GND | GND |
| IN1 | PB3 |
| IN2 | PB4 |

**c) Button:**

**Bảng 3.3 Chân kết nối Button**

|  |  |
| --- | --- |
| Button1 | PA4 |
| Button2 | PA5 |
| Button3 | GPIO 2 |
| GND | GND |

**d) Biến trở:**

**Bảng 3.4 Chân kết nối biến trở**

|  |  |
| --- | --- |
| VCC | 3.3V |
| GND | GND |
| Biến trở 1 | PA6 |
| Biến trở 2 | PA7 |

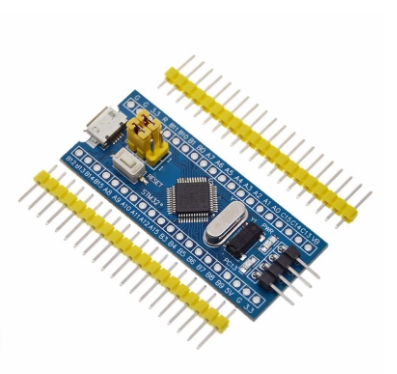
## **3.3 Danh mục các linh kiện, vật tư được sử dụng**

**Bảng 3.5 Các linh kiện được sử dụng**

|  |  |
| --- | --- |
| Vi sử lý (Microcontroller) | STM32F103C8T6, ESP-WROOM-32 |
| Ngoại vi (GPIO) | Nút nhấn, biến trở |
| Hiển thị (Display) | LCD 20x4 |
| Module I2C |
| Nguồn (Power) | Adapter 5V |
| Hộp bảo vệ (Case housing) | Mica |

## **3.4 Giới thiệu về các module chính**

**a) STM32F103C8T6**

****

**Hình 3.4 STM32F103C8T6**

Kit phát triển STM32F103C8T6 Blue Pill ARM Cortex-M3 là loại được sử dụng để nghiên cứu về ARM nhiều nhất hiện nay do có mức giá rẻ đồng thời có thể nạp bootloader Blue Pill để giao tiếp và lập trình với phần mềm Arduino một cách dễ dàng, kit có chất lượng gia công tốt, độ bền cao.

*Thông số kỹ thuật:*

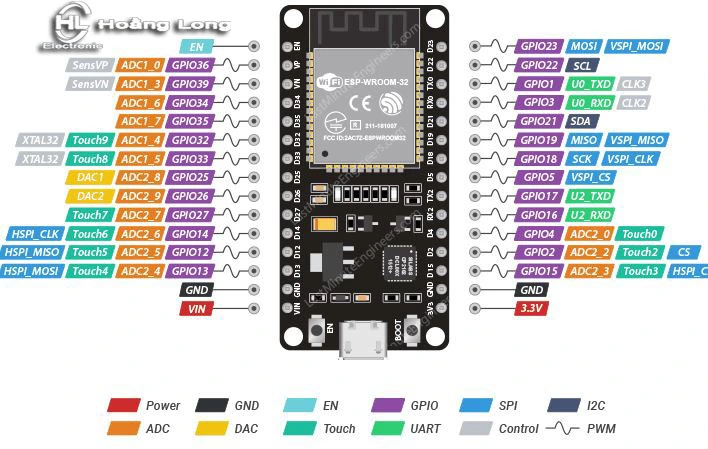
* Vi điều khiển: STM32F103C8T6.
* Điện áp cấp 5VDC qua cổng Micro USB sẽ được chuyển đổi thành 3.3VDC qua IC nguồn và cấp cho vi điều khiển chính.
* Tích hợp sẵn thạch anh 8Mhz.
* Tích hợp sẵn thạnh anh 32Khz cho các ứng dụng RTC.
* Ra chân đầy đủ tất cả các GPIO và giao tiếp: CAN, I2C, SPI, UART, USB,...
* Tích hợp Led trạng thái nguồn, Led PC13, nút Reset.
* Kích thước: 53.34 x 15.24mm.

**b) ESP-WROOM-32**

ESP-WROOM-32 là một mô-đun Wi-Fi + Bluetooth tích hợp dựa trên vi điều khiển ESP32 của **Espressif Systems**.

*Thông số kỹ thuật:*

* Vi xử lý: Dual-core Xtensa LX6, tốc độ lên đến 240 MHz, ROM 448KBytes
* SRAM 520 Kbytes, Flash 4 MB (tùy phiên bản).
* Kết nối: Wi-Fi 2.4 GHz (802.11 b/g/n), Bluetooth 4.2 và BLE.
* GPIO: 34 chân I/O, hỗ trợ ADC, DAC, PWM, UART, SPI, I2C, I2S, CAN, SDIO.
* Điện áp hoạt động: 3.3V(VSL) - 5V(Module).
* Tiêu thụ điện: 160-260 mA (hoạt động), 10 µA (Deep Sleep).
* Bảo mật: WPA2, TLS, mã hóa AES, RSA, SHA.



**Hình 3.5 ESP-WROOM-32**

**c) Module PWM Mosfet 15A**

Module PWM sử dụng Mosfet kích hoạt để chuyển đổi tín hiệu PWM tới tải (động cơ, đèn...) đầu ra từ 5 - 36V. Module cho công suất cao 15A - 400W đáp ứng được hầu hết các thiết bị.

Ứng dụng: Đầu ra có thể điều khiển các thiết bị công suất cao, bóng đèn, dải đèn LED, động cơ DC, máy bơm nước thu nhỏ, van điện từ và có thể nhập vào PWM để điều khiển tốc độ của động cơ và độ sáng của đèn.

*Thông số kỹ thuật:*

* Điện áp làm việc: DC 5V - 36V;
* Nguồn tín hiệu kích hoạt: DC 3.3V - 20V, có thể được kết nối với vi điều khiển cổng IO , giao diện PLC, .v.v, có thể được kết nối với tín hiệu PWM, tín hiệu tần số 0-20 KHZ
* Công suất đầu ra: DC 5V - 36V/15A/400W. Dòng điện tối đa có thể đạt 30A trong điều kiện làm mát phụ trợ.
* Nhiệt độ làm việc: -40 -> 85độC;
* Kích thước: 3,4cm x 1,7cm x 1,7cm



**Hình 3.6 Module PWM Mosfet 15A**

**d) LCD 20x4 kèm module I2C**

Màn hình text LCD2004 xanh dương sử dụng driver HD44780, có khả năng hiển thị 4 dòng với mỗi dòng 20 ký tự, màn hình có độ bền cao, rất phổ biến, nhiều code mẫu và dễ sử dụng thích hợp cho những người mới học và làm dự án.

*Thông số kỹ thuật:*

* Điện áp hoạt động là 5 V.
* Kích thước: 98 x 60 x 13.5 mm
* Chữ trắng, nền xanh dương/chữ đen nền xanh lá
* Khoảng cách giữa hai chân kết nối là 0.1inch tiện dụng khi kết nối với Breadboard.
* Tên các chân được ghi ở mặt sau của màn hình LCD hổ trợ việc kết nối, đi dây điện.
* Có đèn led nền, có thể dùng biến trở hoặc PWM điều chình độ sáng để sử dụng ít điện năng hơn.
* Có thể được điều khiển với 6 dây tín hiệu.
* Có bộ ký tự được xây dựng bằng tiếng Anh và tiếng Nhật.



**Hình 3.7 LCD 20x4 tích hợp module I2C**

**e) Hộp bảo vệ**

Hộp case được gia công laze bằng vật liệu mica trong suốt chứa các linh kiện điện tử bên trong. Có chức năng bảo vệ, bố trí các thành phần linh kiện một cách gọn gàng gia tăng tính thẩm mỹ, ngoài ra còn giúp quá trình lắp đặt thực tế được dễ dàng (chỉ cần một chỗ để cố định cứng cáp). Các vị trí linh kiện được bố trí hợp lí trên mạch PCB và dành một số khoảng trống cho trường hợp phát triển thêm chức năng sẽ có vị trí để lắp đặt.

## **3.5 Hình ảnh sản phẩm thực tế**

|  |  |
| --- | --- |
|  |  |
|  |  |

**Hình 3.8 Tổng hợp hình ảnh thực tế sản phẩm**



**Hình 3.9. Mô hình lắp đặt thiết bị chiếu sáng**



**Hình 3.10. Tổng hợp phần cứng**

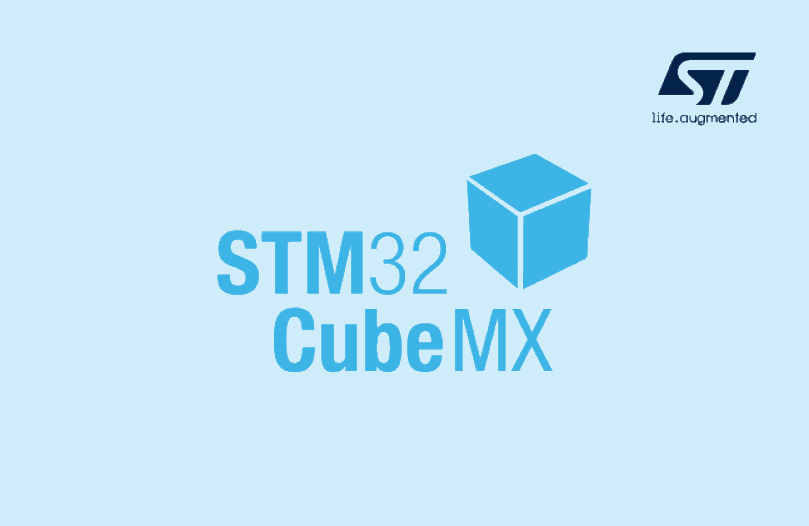
**CHƯƠNG 4 THIẾT KẾ PHẦN MỀM**

## **4.1 Tổng quan**

### **4.1.1 Giới thiệu nền tảng tạo chương trình nhúng STM32**

STM32CubeIDE là một nền tảng phát triển C/C++ tiên tiến với cấu hình ngoại vi, tạo mã, biên dịch mã và các tính năng gỡ lỗi cho vi điều khiển và vi xử lý STM32. Nó dựa trên khuôn khổ Eclipse ® /CDT™ và chuỗi công cụ GCC để phát triển và GDB để gỡ lỗi. Nó cho phép tích hợp hàng trăm plugin hiện có để hoàn thiện các tính năng của Eclipse ® IDE.

STM32CubeIDE tích hợp các chức năng cấu hình STM32 và tạo dự án từ STM32CubeMX để cung cấp trải nghiệm công cụ tất cả trong một và tiết kiệm thời gian cài đặt và phát triển. Sau khi chọn MCU hoặc MPU STM32 trống, hoặc vi điều khiển hoặc vi xử lý được cấu hình sẵn từ việc chọn bo mạch hoặc chọn ví dụ, dự án được tạo và mã khởi tạo được tạo. Tại bất kỳ thời điểm nào trong quá trình phát triển, người dùng có thể quay lại quá trình khởi tạo và cấu hình các thiết bị ngoại vi hoặc phần mềm trung gian và tạo lại mã khởi tạo mà không ảnh hưởng đến mã người dùng.



**Hình 4.1 STM32CubeIDE**

### **4.1.2 Giới thiệu nền tảng tạo chương trình nhúng ESP32**

Ngôn ngữ lập trình Arduino là một ngôn ngữ lập trình cấp cao được sử dụng để lập trình các board mạch vi điều khiển Arduino. Đây là một phần của nền tảng mã nguồn mở Arduino, được thiết kế nhằm giúp người dùng dễ dàng tạo ra các dự án điện tử và lập trình các vi điều khiển để điều khiển các thiết bị phần cứng.

Ngôn ngữ lập trình Arduino dựa trên ngôn ngữ C/C++, một ngôn ngữ lập trình phổ biến và mạnh mẽ. Điều này giúp người dùng có thể dễ dàng học và sử dụng nếu họ đã quen thuộc với C/C++.

**Giới thiệu về PlatformIO được sử dụng:**



**Hình 4.2 PlatformIO IDE**

PlatformIO là một môi trường phát triển tích hợp (IDE) và nền tảng phát triển mã nguồn mở dành cho các dự án nhúng, giúp lập trình và phát triển các ứng dụng trên vi điều khiển và các hệ thống nhúng dễ dàng hơn. Nó hỗ trợ nhiều loại bo mạch và vi điều khiển từ các nhà sản xuất khác nhau như ESP32, STM32, Arduino, và nhiều loại khác.

PlatformIO giúp người dùng viết chương trình nhanh chóng với các tính năng như hoàn tất code C/C ++ và công cụ viết hỗ trợ viết code thông minh Linter để phát triển dự án chuyên nghiệp và nhanh chóng. Các tính năng này vốn không có trong Arduino IDE. Ngoài ra, PlatformIO cung cấp các theme với các màu tối và sáng để thay đổi bất cứ lúc nào. Các tính năng cốt lõi bao gồm hệ thống xây dựng đa nền tảng, trình quản lý thư viện, theo dõi cổng nối tiếp, v.v.

### **4.1.3 Giới thiệu nền tảng tạo chương trình ứng dụng**



**Hình 4.3 Phần mềm Android Studio**

Android Studio là môi trường phát triển thích hợp IDE chính thức được Google cung cấp để phát triển ứng dụng Android. Một số thông tin cơ bản:

* Chủ yếu sử dụng Java (đang được sử dụng cho dự án này), Kotlin và XML để xây dựng giao diện và logic ứng dụng.
* Công cụ thiết kế giao diện kéo thả dễ dàng.
* Có trình giải lập thiết bị ảo với nhiều bản cập nhật hệ điều hành giúp kiểm tra ứng dụng một cách thuận tiện.
* Có trang web hỗ trợ, gợi ý giúp kiểm tra và sửa lỗi: [developer.android.com](file:///C:\Users\lamco\OneDrive\Máy%20tính\developer.android.com)

**Giới thiệu về ngôn ngữ Java**

Java là ngôn ngữ lập trình hướng đối tượng (Object-Oriented Programming – OOP) cho phép tổ chức mã nguồn thành các lớp và đối tượng. Java là một ngôn ngữ rất phổ biến hiện nay, được sử dụng nhiều để phát triển phần mềm và Web.

**Giới thiệu giao diện điều khiển ứng dụng**

|  |  |
| --- | --- |
| **Hình 4.4 Giao diện đăng nhập ứng dụng** | **Hình 4.5 Giao diện điều khiển trung tâm** |
| **Hình 4.6 Giao diện điều khiển RGB** | **Hình 4.7 Giao diện hẹn giờ** |

### **4.1.4 Giới thiệu phần mềm mô phỏng sơ đồ mạch điện**

Sau khi hoàn thành môn học Thực hành cơ sở, với những kiến thức nắm được tôi chọn Altium Designer là phần mềm hỗ trợ thiết kế sơ đồ nguyên lí cho dự án. Với các chức năng cơ bản, không cần kiểm tra lỗi trước nên việc chọn phần mềm này thuận tiện cho sau này phát triển mạch in PCB thay vì lựa chọn Proteus (đã được học trước đó).

A logo with a letter d

Description automatically generated

**Hình 4.8 Phần mềm Altium**

Altium Designer là một phần mềm thiết kế PCB (Printed Circuit Board) dựa trên đám mây toàn diện được sử dụng để chụp sơ đồ, mô hình 3D, bản vẽ lắp ráp và mô phỏng. Với Altium, các nhà thiết kế PCB có thể chia sẻ ảnh chụp nhanh và thiết kế trực tiếp với khách hàng và nhận phản hồi theo thời gian thực trong nháy mắt. Hơn nữa, Altium Designer là một phần mềm hỗ trợ công nghệ cao đã trở nên phổ biến trong ngành công nghiệp bo mạch điện tử.

**Giới thiệu bản vẽ PCB**

|  |  |
| --- | --- |
| **Hình 4.9 2D PCB** | **Hình 4.10 3D PCB** |

### **4.1.5 Giới thiệu phần mềm thiết kế hộp mica bảo vệ thiết bị**

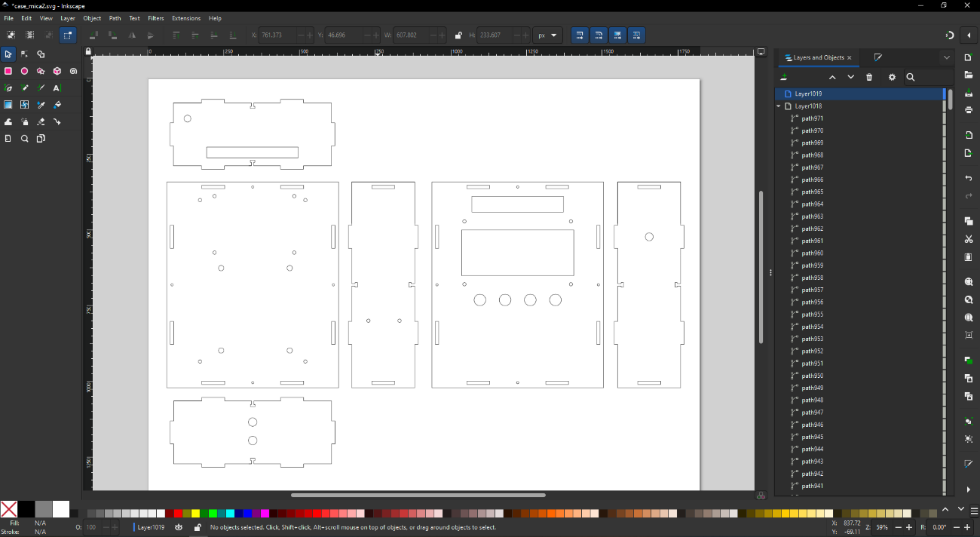


**Hình 4.11 Phần mềm Inkscape**

Inkscape là một trình chỉnh sửa đồ họa miễn phí cho Windows, có thể được sử dụng cho mục đích cá nhân và chuyên nghiệp. Chương trình có một bộ công cụ thiết kế đầy đủ cho phép bạn làm việc với các tệp đồ họa vector, được biết đến là SVG.

Ngoài các tệp SVG, phần mềm cũng hỗ trợ xuất ra các file .dxf, .hpgl,… hỗ trợ cho việc cắt laze. Đây là một phần mềm miễn phí và khá dễ dàng sử dụng đây là điều tôi ưu tiên để lựa chọn phần mềm này để thiết kế một hộp bảo vệ thiết bị.

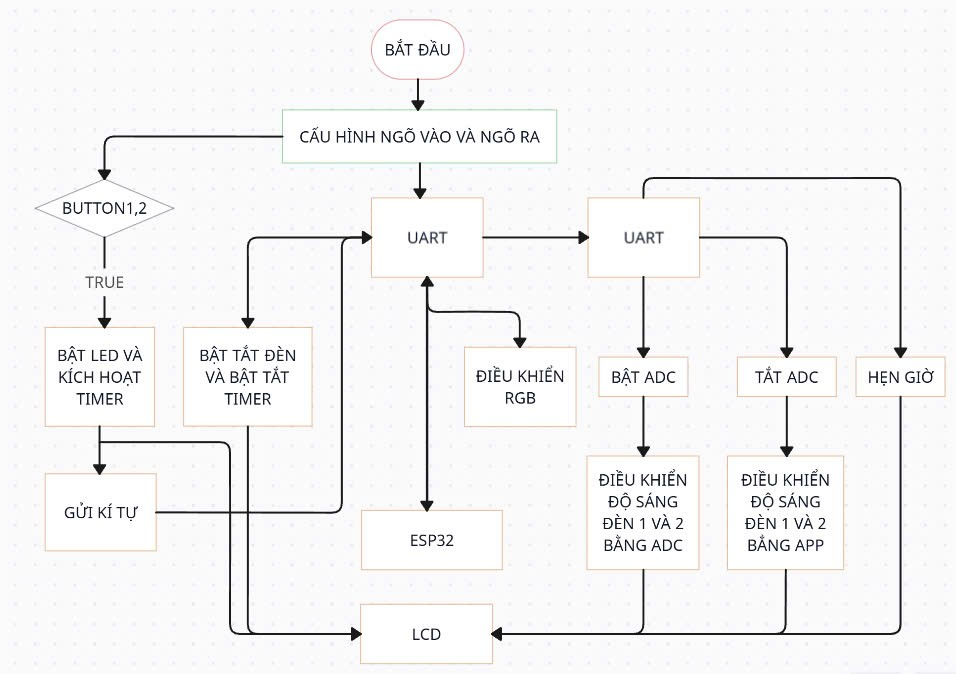
**Giới thiệu giao diện thực hiện trên phần mềm Inkscape**



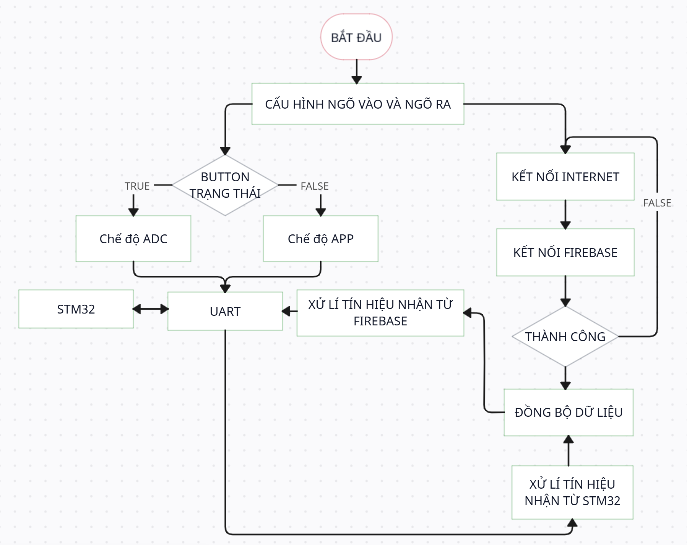
**Hình 4.12 Giao diện thiết kế hộp bảo vệ**

## **4.2 Sơ đồ thiết kế phần mềm**

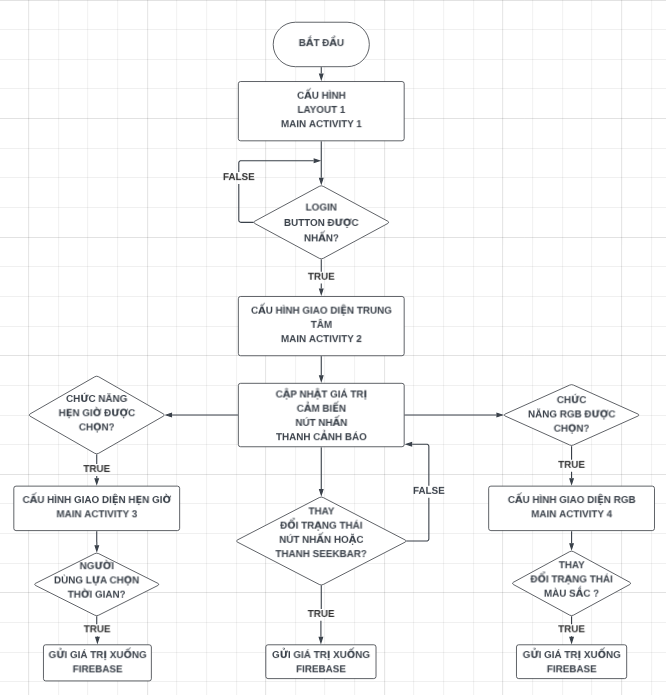
Lưu đồ giải thuật:



**Hình 4.13 Lưu đồ giải thuật chương trình nhúng (STM32)**



**Hình 4.14 Lưu đồ giải thuật chương trình nhúng (ESP32)**



**Hình 4.15 Lưu đồ giải thuật chương trình ứng dụng**

## **4.3 Thực thi các module phần mềm**

### **4.3.1 Hằng số và biến toàn cục chương trình STM32**

1. **Các biến toàn cục**

uint32\_t adcValue;

uint16\_t pwmValue;

uint32\_t adcValue2;

uint16\_t pwmValue2;

volatile bool flagRGBReceived = false;

volatile bool flagNumberReceived = false ;

volatile bool flagCommandReceived = false;

uint8\_t data\_valid = 1; // Biến để kiểm tra tính hợp lệ của dữ liệu

GPIO\_PinState button1State;

GPIO\_PinState button2State;

uint8\_t rx\_data;

uint8\_t uart2\_rx\_buffer[6]; // Expecting 6 bytes for RGB values

uint8\_t uart2\_rx\_index = 0;

char rx\_buffer[128]; // Buffer để nhận dữ liệu từ UART

int index1 = 0;

uint8\_t button1Flag = 0; // Trạng thái của Button 1 (1 hoặc 0)

uint8\_t button2Flag = 0;

uint8\_t rxBuffer[1];

const uint32\_t adcSendInterval = 500;

//static uint32\_t lastAdcSendTime = 0;

uint32\_t lastPressTime1 = 0;

uint32\_t lastPressTime2 = 0;

const uint32\_t debounceDelay = 300;

char rx\_buffer[RX\_BUFFER\_SIZE];

volatile int adcActive = 1;

char RxBuffer[50];

uint8\_t RxIndex = 0;

uint8\_t lcdInitialized = 0;

volatile int brightness1 = 0;

volatile int brightness2 = 0;

uint8\_t previousBrightness1 = 0;

uint8\_t previousBrightness2 = 0;

int deleteRequested = 0;

// Thêm mảng để lưu các lệnh xóa

char deleteNotes[4][7] = {"BAT.PK", "TAT.PK", "BAT.PN", "TAT.PN"};

int deleteFlags[4] = {0, 0, 0, 0}; // Đánh dấu các lệnh xóa đã được nhận

uint32\_t lastAdcTime = 0; // Thời gian thực hiện lần cuối của phần ADC

uint32\_t lastUpdateTime = 0; // Thời gian thực hiện lần cuối của updateLCD

uint32\_t adcInterval = 500; // Khoảng thời gian giữa hai lần đọc ADC (ms)

uint32\_t lcdInterval = 200; // Khoảng thời gian giữa hai lần cập nhật LCD (ms)

#define MAX\_ALARMS 20

#define MAX\_NOTE\_LENGTH 10

#define ADC\_MODE 1

#define APP\_MODE 0

uint8\_t controlMode = APP\_MODE;

#define LED1\_PIN GPIO\_PIN\_0

#define LED2\_PIN GPIO\_PIN\_1

#define LED\_GPIO\_PORT GPIOB

#define BUFFER\_SIZE 6

char messageBuffer[BUFFER\_SIZE];

#define noOfLEDs 16

uint16\_t pwmData[24\*noOfLEDs];

#define MAX\_ALARMS 20 // Số lượng báo thức tối đa

#define ALARM\_SWITCH\_INTERVAL 2000 // Thời gian chuyển đổi giữa các báo thức (ms)

typedef struct {

char time[10]; // Định dạng thời gian HH:MM:SS

char note[6]; // Ghi chú báo thức

} Alarm;

Alarm alarms[MAX\_ALARMS]; // Mảng lưu trữ các báo thức

int alarmCount = 0; // Đếm số lượng báo thức đã lưu

int currentDisplayAlarm = 0; // Chỉ báo thức hiện tại đang hiển thị

uint32\_t lastSwitchTime = 0; // Thời gian chuyển đổi lần cuối

**b) Các hàm được sử dụng**

void HAL\_TIM\_PWM\_PulseFinishedCallback(TIM\_HandleTypeDef \*htim)

void resetAllLED(void)

void setAllLED(void)

void setLED(int LEDposition, int Red, int Green, int Blue)

void ws2812Send(void)

void HAL\_GPIO\_EXTI\_Callback(uint16\_t GPIO\_Pin)

void deleteAlarmsByNoteAndUpdate(const char\* note)

void handleCommand(char receivedChar)

void UpdateBrightness(TIM\_HandleTypeDef \*htim, uint8\_t brightness, uint32\_t channel)

void DisplayBrightnessOnLCD(uint8\_t brightness1, uint8\_t brightness2)

void ProcessReceivedCommaSeparatedString(char \*data)

void updateBrightnessOnLCD(void)

void updateLedStatusOnLCD(void)

void initializeLCD(void)

void processAlarmData(char\* data)

void updateLCD(void)

void HAL\_UART\_RxCpltCallback(UART\_HandleTypeDef \*huart)

void ProcessReceivedRGBValues(char \*rgbData)

int main(void)

### **4.3.2 Hằng số và biến toàn cục chương trình ứng dụng**

1. **Các biến toàn cục trong chương trình main1 và main2**

**private EditText editemail, private** **EditText editpass**: biến này trỏ đến thành phần giao diện có đối tượng là editemail và editpass trong file XML.

**private Button btnlogin:** biến này trỏ đến thành phần giao diện button có đối tượng là btnlogin trong file XML

**private FirebaseAuth mAuth:** biến mAuth trỏ đến đối tượng FirebaseAuth nằm trong lớp thư viện Firebase Authentication.

**private Button btnlogout; private TextView nhietdoTextView; private TextView mq2TextView; private TextView humidityTextView; private TextView muccanhbao; private SeekBar thanhcanhbao; private ImageButton relay [1,2,3,4]; private ImageButton btnwarning**: tương tự các biến này cũng trỏ đến các đối tượng có chức năng.

1. **Các phương thức trong chương trình ứng dụng**

**login():**  phương thức này được gọi khi người dùng bỏ qua bước nhập tài khoản đăng nhập.

**onComplete(@NonNull Task<AuthResult> task):** phương thức này được gọi khi người dùng hoàn tất nhập chính xác tài khoản và nhấn nút LOGIN.

**onCreate(Bundle savedInstanceState):** phương thức này khởi tạo các thành phần dữ liệu và cài đặt ban đầu cho Activity.

**onDataChange(@NonNull DataSnapshot snapshot):** phương thức này lắng nghe sự thay đổi dữ liệu từ Firebase.

**onClick(View v):** lắng nghe sự kiện nhấn (click) vào giao diện đã được khai báo toàn cục, ví dụ: ImageButton.

**onProgressChanged(SeekBar seekBar, int i, boolean b):** phương thức này lắng nghe sự thay đổi từ giao diện thanh Seekbar .

### **4.3.3 Hằng số và biến toàn cục chương trình ESP32**

**a)Các biến toàn cục**

HardwareSerial Serialport(2);

#define BUTTON\_PIN 2

FirebaseData fbdo;

FirebaseData fbdo2;

FirebaseData fbdo5;

FirebaseData fbdo6;

FirebaseData fbdo7;

FirebaseData fbdo8;

FirebaseAuth auth;

FirebaseConfig config;

FirebaseData firebaseData;

String paths   = "/BT1";

String paths2  = "/BT2";

String paths5  = "/LB1";

String paths6  = "/LB2";

String paths7  = "/LEDRGB";

String paths8  = "/TIME";

const char ssid[] = "NHOM12";

const char pass[] = "01234567";

bool dataButton1 = false;

bool dataButton2 = false;

int dataButton7 = 0;

int button1State ;

int button2State;

unsigned long lastDebounceTime = 0;  // Thời gian lần nhấn cuối cùng

unsigned long debounceDelay = 300;

volatile bool sendToFirebaseFlag = false;

volatile bool buttonPressed = false;  // Flag for Button press

bool lastButtonState = LOW;

bool buttonState = false;

char currentTimeStr[9]; // Để lưu thời gian dạng HH:MM:SS

char currentDateStr[11]; //

unsigned long previousMillis = 0;  // Thời gian trước đó

const long interval = 1000;

struct Alarm {

    String time;  // Lưu thời gian báo thức dưới dạng chuỗi "HH:MM:SS"

    String note;  // Lưu chuỗi note tương ứng

};

// Tạo mảng để lưu nhiều báo thức và note

Alarm alarms[20]; // Lưu tối đa 20 báo thức với các note khác nhau

int alarmCount = 0;

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

**b)Các hàm được sử dụng**

void printStreamData(StreamData data)

void streamCallback(StreamData data)

void streamTimeoutCallback(bool timeout)

void receiveDataFromSTM32()

void IRAM\_ATTR handleButtonPress()

# **CHƯƠNG 5 KIỂM TRA HOẠT ĐỘNG CỦA HỆ THỐNG**

## **5.1 Đánh giá việc thực hiện các tính năng của hệ thống**

Các chức năng của hệ thống đều được thực thi, tuy nhiên về việc gửi dữ liệu lên FIREBASE chưa được tối ưu hoàn hảo, dù là vậy nhưng tốc độ phản hồi vẫn nhanh trông thấy. Sau một thời gian hoạt động, hệ thống xảy ra vấn đề đó là việc gửi dữ liệu lên Firebase bị gián đoạn và thời gian phản hồi khá lâu. Tuy nhiên, ngược lại với gửi dữ liệu lên Firebase thì việc điều khiển bằng app ứng dụng, quá trình gửi lệnh điều khiển xuống ESP vẫn hoạt động ở thời gian thực chính xác.

## **5.2 Đánh giá các trường hợp sử dụng**

Trường hợp 1, Trường hợp 2, Trường hợp 3, Trường hợp 4, Trường hợp 5: Hầu hết các trường hợp đều đạt yêu cầu ban đầu nhóm đặt ra, riêng trường hợp 6 kết quả đánh giá được nêu ở mục 5.1.

## **5.3 Nhận xét**

Hệ thống triển khai đạt được 100% khả năng về tính năng hệ thống, tuy nhiên hệ thống chỉ đạt được 70% các trường hợp mà nhóm đặt ra lúc bắt đầu.

Các tính năng chưa thực hiện được không quá quan trọng và không ảnh hưởng quá nhiều đến hệ thống (chế độ deepsleep trên ESP32 và STM32, AI hỗ trợ dự đoán giá tiền điện, tính toán công suất tiêu thụ của thiết bị, chế độ điều khiển offline BLE).

Tổng quát hệ thống triển khai nhóm cảm thấy khá hài lòng so với dự định ban đầu, thời gian tới nhóm sẽ phát triển thêm nhiều chức năng để vừa ý với yêu cầu chung của dự án.

# **KẾT LUẬN**

Tóm tắt nội dung và kết quả đã thực hiện trong dự án này: nghiên cứu, thiết kế và triển khai thành công ứng dụng điều khiển trên điện thoại Android, hệ thống điều khiển thiết bị chiếu sáng thời gian thực.

Những hạn chế, chưa đạt được của dự án:

* Chưa tối ưu hoàn toàn việc gửi dữ liệu lên Firebase và thông số cấu hình nạp vào bộ nhớ FLASH của ESP32
* Ứng dụng điều khiển chưa phát triển cho người dùng IOS.
* Chưa hoàn thành được các yêu cầu ban đầu đặt ra như: chế độ BLE, tính công suất tiêu thụ

Hướng phát triển tiếp theo:

* Cải thiện chương trình giúp tiết kiệm bộ nhớ FLASH tạo ra không gian trống để đáp ứng đủ với tốc độ ESP thực hiện kết nối Firebase tránh trường hợp tràn bộ nhớ hay xảy ra “timeout”.
* Xây dựng ứng dụng cho hệ điều hành IOS
* Tích hợp AI hỗ trợ kiểm soát điện năng tiêu thụ.
* Hoàn thành các yêu cầu đặt ra lúc bắt đầu dự án.
* Đồ án này cho ta cách giải quyết “giải pháp chiếu sáng cho mô hình nhà thông minh” nên trong tương lai dự án này sẽ được kết hợp với các giải pháp khác ví dụ: “giải pháp tiết kiệm năng lượng”, “giải pháp điều khiển thông minh” để xây dựng và hoàn thiện mô hình nhà thông minh góp phần vào xây dựng đồ án tốt nghiệp.

# **PHỤ LỤC**

**Code STM32CubeIDE**

|  |
| --- |
| **STM32CubeIDE** |
| #include "main.h"  /\* Private includes ----------------------------------------------------------\*/  /\* USER CODE BEGIN Includes \*/  #include <string.h>  #include <stdio.h>  #include <stdlib.h>  #include <stdbool.h>  #include "stm32f1xx\_hal.h"  #include "i2c-lcd.h"  #include <ctype.h>  uint32\_t adcValue;  uint16\_t pwmValue;  uint32\_t adcValue2;  uint16\_t pwmValue2;  // Định nghĩa LEDS bằng LED\_CNT  volatile bool flagRGBReceived = false;  volatile bool flagNumberReceived = false ;  volatile bool flagCommandReceived = false;  /\* USER CODE END Includes \*/  /\* Private typedef -----------------------------------------------------------\*/  /\* USER CODE BEGIN PTD \*/  /\* USER CODE END PTD \*/  /\* Private define ------------------------------------------------------------\*/  /\* USER CODE BEGIN PD \*/  /\* USER CODE END PD \*/  /\* Private macro -------------------------------------------------------------\*/  /\* USER CODE BEGIN PM \*/  /\* USER CODE END PM \*/  /\* Private variables ---------------------------------------------------------\*/  ADC\_HandleTypeDef hadc1;  I2C\_HandleTypeDef hi2c1;  TIM\_HandleTypeDef htim1;  TIM\_HandleTypeDef htim3;  DMA\_HandleTypeDef hdma\_tim3\_ch1\_trig;  UART\_HandleTypeDef huart2;  /\* USER CODE BEGIN PV \*/  uint8\_t data\_valid = 1; // Biến để kiểm tra tính hợp lệ của dữ liệu  GPIO\_PinState button1State;  GPIO\_PinState button2State;  uint8\_t rx\_data;  uint8\_t uart2\_rx\_buffer[6]; // Expecting 6 bytes for RGB values  uint8\_t uart2\_rx\_index = 0;  char rx\_buffer[128]; // Buffer để nhận dữ liệu từ UART  int index1 = 0; // Chỉ số cho STATE\_COMMAND  int index2 = 0; // Chỉ số cho STATE\_NUMBER  #define RX\_BUFFER\_SIZE 128  uint8\_t button1Flag = 0; // Trạng thái của Button 1 (1 hoặc 0)  uint8\_t button2Flag = 0;  uint8\_t rxBuffer[1];  const uint32\_t adcSendInterval = 500;  //static uint32\_t lastAdcSendTime = 0;  uint32\_t lastPressTime1 = 0; // Thời gian nhấn Button 1  uint32\_t lastPressTime2 = 0; // Thời gian nhấn Button 2  const uint32\_t debounceDelay = 300;  char rx\_buffer[RX\_BUFFER\_SIZE];  volatile int adcActive = 1;  char RxBuffer[50];  uint8\_t RxIndex = 0;  uint8\_t lcdInitialized = 0;  volatile int brightness1 = 0;  volatile int brightness2 = 0;  uint8\_t previousBrightness1 = 0;  uint8\_t previousBrightness2 = 0;  int deleteRequested = 0;  // Thêm mảng để lưu các lệnh xóa  char deleteNotes[4][7] = {"BAT.PK", "TAT.PK", "BAT.PN", "TAT.PN"};  int deleteFlags[4] = {0, 0, 0, 0}; // Đánh dấu các lệnh xóa đã được nhận  uint32\_t lastAdcTime = 0; // Thời gian thực hiện lần cuối của phần ADC  uint32\_t lastUpdateTime = 0; // Thời gian thực hiện lần cuối của updateLCD  uint32\_t adcInterval = 500; // Khoảng thời gian giữa hai lần đọc ADC (ms)  uint32\_t lcdInterval = 200; // Khoảng thời gian giữa hai lần cập nhật LCD (ms)  /\* USER CODE END PV \*/  /\* Private function prototypes -----------------------------------------------\*/  void SystemClock\_Config(void);  static void MX\_GPIO\_Init(void);  static void MX\_DMA\_Init(void);  static void MX\_ADC1\_Init(void);  static void MX\_TIM1\_Init(void);  static void MX\_USART2\_UART\_Init(void);  static void MX\_TIM3\_Init(void);  static void MX\_I2C1\_Init(void);  /\* USER CODE BEGIN PFP \*/  #define MAX\_ALARMS 20  #define MAX\_NOTE\_LENGTH 10  #define ADC\_MODE 1  #define APP\_MODE 0  uint8\_t controlMode = APP\_MODE;  #define LED1\_PIN GPIO\_PIN\_0  #define LED2\_PIN GPIO\_PIN\_1  #define LED\_GPIO\_PORT GPIOB  #define BUFFER\_SIZE 6  char messageBuffer[BUFFER\_SIZE];  #define noOfLEDs 16  uint16\_t pwmData[24\*noOfLEDs];  #define MAX\_ALARMS 20 // Số lượng báo thức tối đa  #define ALARM\_SWITCH\_INTERVAL 2000 // Thời gian chuyển đổi giữa các báo thức (ms)  typedef struct {  char time[10]; // Định dạng thời gian HH:MM:SS  char note[6]; // Ghi chú báo thức  } Alarm;  Alarm alarms[MAX\_ALARMS]; // Mảng lưu trữ các báo thức  int alarmCount = 0; // Đếm số lượng báo thức đã lưu  int currentDisplayAlarm = 0; // Chỉ báo thức hiện tại đang hiển thị  uint32\_t lastSwitchTime = 0; // Thời gian chuyển đổi lần cuối  /\* USER CODE END PFP \*/  /\* Private user code ---------------------------------------------------------\*/  /\* USER CODE BEGIN 0 \*/  void HAL\_TIM\_PWM\_PulseFinishedCallback(TIM\_HandleTypeDef \*htim)  {  HAL\_TIM\_PWM\_Stop\_DMA(&htim3, TIM\_CHANNEL\_1);  htim3.Instance->CCR1 = 0;  }  void resetAllLED(void)  {  for (int i = 0; i < 24 \* noOfLEDs; i++)  pwmData[i] = 0; // �?ặt lại tất cả giá trị LED v�? 0  }  void setAllLED(void)  {  for (int i = 0; i < 24 \* noOfLEDs; i++)  pwmData[i] = 0; // �?ặt lại giá trị của tất cả LED  }  void setLED(int LEDposition, int Red, int Green, int Blue)  {  // Kiểm tra xem LEDposition có hợp lệ không  if (LEDposition < 0 || LEDposition >= noOfLEDs) return;  //Đặt màu cho LED  for (int i = 0; i < 8; i++)  {  pwmData[24 \* LEDposition + 7 - i] = ((Green >> i) & 1) + 1; // Màu xanh  }  for (int i = 0; i < 8; i++)  {  pwmData[24 \* LEDposition + 15 - i] = ((Red >> i) & 1) + 1; // Màu đ�?  }  for (int i = 0; i < 8; i++)  {  pwmData[24 \* LEDposition + 23 - i] = ((Blue >> i) & 1) + 1; // Màu xanh dương  }  }  void ws2812Send(void)  {  HAL\_TIM\_PWM\_Start\_DMA(&htim3, TIM\_CHANNEL\_1, (uint32\_t \*)pwmData, 24 \* noOfLEDs);  }  uint32\_t mapValue(uint32\_t input)  {  return (input \* 1023)/4095;  }  uint32\_t mapValue2(uint32\_t input2)  {  return (input2 \* 1023)/4095;  }  void HAL\_GPIO\_EXTI\_Callback(uint16\_t GPIO\_Pin)  {  uint32\_t currentTime = HAL\_GetTick(); // Lấy thời gian hiện tại  // Kiểm tra Button 1  if (GPIO\_Pin == GPIO\_PIN\_4) // Button 1  {  if (currentTime - lastPressTime1 > debounceDelay)  {  if (HAL\_GPIO\_ReadPin(GPIOB, GPIO\_PIN\_0) == GPIO\_PIN\_RESET) { // Kiểm tra nếu LED1 tắt  HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_SET); // Bật LED1  button1Flag = 1;  HAL\_TIM\_PWM\_Start(&htim1, TIM\_CHANNEL\_2);// Set button1Flag lên 1  } else {  HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET); // Tắt LED1  button1Flag = 0; // Set button1Flag v�? 0  HAL\_TIM\_PWM\_Stop(&htim1, TIM\_CHANNEL\_2);  }  // Tạo chuỗi gửi  snprintf(messageBuffer, sizeof(messageBuffer), "%d,%d\n", button1Flag, button2Flag); // Tạo chuỗi gửi  // Gửi chuỗi  HAL\_UART\_Transmit\_IT(&huart2, (uint8\_t\*)messageBuffer, strlen(messageBuffer)); // Truyền không chặn  lastPressTime1 = currentTime; // Cập nhật thời gian nhấn  }  }  // Kiểm tra Button 2  if (GPIO\_Pin == GPIO\_PIN\_5) // Button 2  {  if (currentTime - lastPressTime2 > debounceDelay)  {  if (HAL\_GPIO\_ReadPin(GPIOB, GPIO\_PIN\_1) == GPIO\_PIN\_RESET) { // Kiểm tra nếu LED2 tắt  HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_SET); // Bật LED2  button2Flag = 1; // Set button2Flag lên 1  HAL\_TIM\_PWM\_Start(&htim1, TIM\_CHANNEL\_1);  } else {  HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_RESET); // Tắt LED2  button2Flag = 0; // Set button2Flag = 0  HAL\_TIM\_PWM\_Stop(&htim1, TIM\_CHANNEL\_1);  }  // Tạo chuỗi gửi  snprintf(messageBuffer, sizeof(messageBuffer), "%d,%d\n", button1Flag, button2Flag); // Tạo chuỗi gửi  // Gửi chuỗi  HAL\_UART\_Transmit\_IT(&huart2, (uint8\_t\*)messageBuffer, strlen(messageBuffer)); // Truyềnn không chặn  lastPressTime2 = currentTime; // Cập nhật thời gian nhấn  }  }  updateLedStatusOnLCD();  }  void deleteAlarmsByNoteAndUpdate(const char\* note) {  int i = 0;  bool alarmDeleted = false; // Biến để xác định nếu báo thức đã bị xóa  while (i < alarmCount) {  if (strcmp(alarms[i].note, note) == 0) {  // Dịch chuyển tất cả các báo thức phía sau lên một vị trí để xóa báo thức hiện tại  for (int j = i; j < alarmCount - 1; j++) {  alarms[j] = alarms[j + 1];  }  alarmCount--; // Giảm số lượng báo thức sau khi xóa  alarmDeleted = true; // Đánh dấu rằng báo thức đã bị xóa  // Không tăng i để kiểm tra lại vị trí hiện tại sau khi dịch chuyển  } else {  i++; // Chỉ tăng i nếu không xóa phần tử  }  }  // Nếu có báo thức đã bị xóa, cập nhật lại hiển thị trên LCD  if (alarmDeleted) {  currentDisplayAlarm = 0; // Đặt lại chỉ số hiển thị báo thức về 0  updateLCD(); // Cập nhật hiển thị trên LCD  }  }  void handleCommand(char receivedChar) {  switch (receivedChar) {  case 'a': // Bật LED1 và bắt đầu TIM1  if (button1Flag == 0) {  HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_SET);  button1Flag = 1;  HAL\_TIM\_PWM\_Start(&htim1, TIM\_CHANNEL\_2);  updateLedStatusOnLCD();  }  break;  case 'b': // Tắt LED1 và dừng TIM1  if (button1Flag == 1) {  HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_0, GPIO\_PIN\_RESET);  button1Flag = 0;  HAL\_TIM\_PWM\_Stop(&htim1, TIM\_CHANNEL\_2);  updateLedStatusOnLCD();  }  break;  case 'c': // Bật LED2 và bắt đầu TIM3  if (button2Flag == 0) {  HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_SET);  button2Flag = 1;  HAL\_TIM\_PWM\_Start(&htim1, TIM\_CHANNEL\_1);  updateLedStatusOnLCD();  }  break;  case 'd': // Tắt LED2 và dừng TIM3  if (button2Flag == 1) {  HAL\_GPIO\_WritePin(GPIOB, GPIO\_PIN\_1, GPIO\_PIN\_RESET);  button2Flag = 0;  HAL\_TIM\_PWM\_Stop(&htim1, TIM\_CHANNEL\_1);  updateLedStatusOnLCD();  }  break;  case 'e': // Kích hoạt vòng lặp ADC  adcActive = 1;  lcd\_put\_cur(3, 0);  lcd\_send\_string("Mode: ADC CONTROL");  break;  case 'f': // Dừng vòng lặp ADC  adcActive = 0;  lcd\_put\_cur(3, 0);  lcd\_send\_string("Mode: APP CONTROL");  break;  case 'g':  deleteAlarmsByNoteAndUpdate("a");  break;  case 'h':  deleteAlarmsByNoteAndUpdate("b");  break;  case 'i':  deleteAlarmsByNoteAndUpdate("c");  break;  case 'j':  deleteAlarmsByNoteAndUpdate("d");  break;  // Các lệnh khác giữ nguyên  default:  break;  }  }  // Function to process the received RGB values  void UpdateBrightness(TIM\_HandleTypeDef \*htim, uint8\_t brightness, uint32\_t channel)  {  uint32\_t pwm\_value = (brightness \* (htim->Init.Period)) / 100;  \_\_HAL\_TIM\_SET\_COMPARE(htim, channel, pwm\_value);  }  void DisplayBrightnessOnLCD(uint8\_t brightness1, uint8\_t brightness2)  {  char buffer[21];  // Xóa dòng cũ  lcd\_put\_cur(2, 0);  lcd\_send\_string(" "); // Xóa toàn bộ dòng 2 với 20 khoảng trắng  // Hiển thị độ sáng  lcd\_put\_cur(2, 0);  snprintf(buffer, sizeof(buffer), "LB1:%3d%% LB2:%3d%%", brightness1, brightness2);  lcd\_send\_string(buffer);  }  void ProcessReceivedCommaSeparatedString(char \*data)  {  uint8\_t brightness1 = 0;  uint8\_t brightness2 = 0;  // Tách chuỗi thành hai giá trị dựa trên dấu phẩy  char \*rest = data;  char \*token;  // Tách giá trị đầu tiên  token = strtok\_r(rest, ",", &rest);  if (token != NULL)  {  brightness1 = atoi(token); // Chuyển đổi giá trị đầu tiên sang số nguyên  }  // Tách giá trị thứ hai  token = strtok\_r(rest, ",", &rest);  if (token != NULL)  {  brightness2 = atoi(token); // Chuyển đổi giá trị thứ hai sang số nguyên  }  // Đảm bảo các giá trị trong khoảng 0-100  if (brightness1 <= 100 && brightness2 <= 100)  {  UpdateBrightness(&htim1, brightness1, TIM\_CHANNEL\_2); // Cập nhật độ sáng cho TIM1 Channel 1  UpdateBrightness(&htim1, brightness2, TIM\_CHANNEL\_1); // Cập nhật độ sáng cho TIM1 Channel 2  DisplayBrightnessOnLCD(brightness1,brightness2);  }  else  {  // Xử lý nếu giá trị nằm ngoài khoảng 0-100  printf("Giá trị không hợp lệ\n");  }  }  void updateBrightnessOnLCD(void) {  char buffer[21];  // Xóa toàn bộ dòng 2 trước khi cập nhật  lcd\_put\_cur(2, 5);  lcd\_send\_string(" "); // 20 khoảng trắng  lcd\_put\_cur(2, 14);  lcd\_send\_string(" "); // 20 khoảng trắng  // Cập nhật độ sáng cho cả LED1 và LED2  lcd\_put\_cur(2, 0);  snprintf(buffer, sizeof(buffer), "LB1:%3d%% LB2:%3d%%", brightness1, brightness2);  lcd\_send\_string(buffer);  }  void updateLedStatusOnLCD(void) {  char buffer[21];  // Cập nhật trạng thái LED1  lcd\_put\_cur(0, 17);  snprintf(buffer, sizeof(buffer), "%s", HAL\_GPIO\_ReadPin(GPIOB, GPIO\_PIN\_0) == GPIO\_PIN\_SET ? "ON " : "OFF");  lcd\_send\_string(buffer);  // Cập nhật trạng thái LED2  lcd\_put\_cur(1, 17);  snprintf(buffer, sizeof(buffer), "%s", HAL\_GPIO\_ReadPin(GPIOB, GPIO\_PIN\_1) == GPIO\_PIN\_SET ? "ON " : "OFF");  lcd\_send\_string(buffer);  }  void initializeLCD(void) {  HAL\_Delay(100);  lcd\_init();  HAL\_Delay(100);  lcd\_clear();  HAL\_Delay(50);  // Hiển thị layout cơ bản  lcd\_put\_cur(0, 0);  lcd\_send\_string("Time:");  lcd\_put\_cur(1, 0);  lcd\_send\_string("Note:");  lcd\_put\_cur(0,14);  lcd\_send\_string("L1:");  lcd\_put\_cur(1,14);  lcd\_send\_string("L2:");  lcd\_put\_cur(2, 0);  lcd\_send\_string("LB1:");  lcd\_put\_cur(2,14);  lcd\_send\_string("LB2:");  lcd\_put\_cur(3, 0);  lcd\_send\_string("Mode: ADC CONTROL");  // Cập nhật trạng thái LED ban đầu  updateLedStatusOnLCD();  // updateBrightnessOnLCD();  // updateControlMode();  lcdInitialized = 1;  }  // Function to update LCD without clearing  void processAlarmData(char\* data) {  if (alarmCount >= MAX\_ALARMS) {  return; // Nếu mảng đầy, không lưu thêm báo thức mới  }  Alarm newAlarm;  int comma\_count = 0;  int i = 1; // Bắt đầu từ index 1 vì ký tự đầu tiên là $  int segment\_index = 1;  // Đặt các biến tạm thời để lưu giờ, phút, giây  char hour[3] = {0};  char minute[3] = {0};  char second[3] = {0};  for (; data[i] != '\0'; i++) {  if (data[i] == ',') {  comma\_count++;  if (comma\_count == 1) {  strncpy(hour, &data[segment\_index], i - segment\_index); // Lưu giờ  hour[i - segment\_index] = '\0'; // Đảm bảo kết thúc chuỗi giờ  segment\_index = i + 1;  } else if (comma\_count == 2) {  strncpy(minute, &data[segment\_index], i - segment\_index); // Lưu phút  minute[i - segment\_index] = '\0'; // Đảm bảo kết thúc chuỗi phút  segment\_index = i + 1;  } else if (comma\_count == 3) {  strncpy(second, &data[segment\_index], i - segment\_index); // Lưu giây  second[i - segment\_index] = '\0'; // Đảm bảo kết thúc chuỗi giây  strncpy(newAlarm.note, &data[i + 1], sizeof(newAlarm.note) - 1); // Lấy ghi chú sau dấu phẩy cuối cùng  newAlarm.note[sizeof(newAlarm.note) - 1] = '\0';  break;  }  }  }  // Tạo chuỗi định dạng thời gian "HH:MM:SS"  snprintf(newAlarm.time, sizeof(newAlarm.time), "%s:%s:%s", hour, minute, second);  // Chuyển đổi ghi chú thành các chữ cái đơn "a", "b", "c", "d" để lưu trong mảng  if (strcmp(newAlarm.note, "1") == 0) {  strcpy(newAlarm.note, "a");  } else if (strcmp(newAlarm.note, "2") == 0) {  strcpy(newAlarm.note, "b");  } else if (strcmp(newAlarm.note, "3") == 0) {  strcpy(newAlarm.note, "c");  } else if (strcmp(newAlarm.note, "4") == 0) {  strcpy(newAlarm.note, "d");  } else {  strcpy(newAlarm.note, "NA"); // Trường hợp mặc định nếu không hợp lệ  }  alarms[alarmCount++] = newAlarm; // Thêm báo thức vào mảng  updateLCD(); // Hiển thị báo thức ngay lập tức  }  void updateLCD(void) {  char timeBuffer[10] ; // Chuỗi trống để xóa nội dung cũ  char noteBuffer[7] ; // Chuỗi trống để xóa nội dung cũ  uint32\_t currentTime = HAL\_GetTick();  // Kiểm tra thời gian để chuyển đổi báo thức  if (alarmCount > 0 && (currentTime - lastSwitchTime >= ALARM\_SWITCH\_INTERVAL)) {  currentDisplayAlarm = (currentDisplayAlarm + 1) % alarmCount; // Chuyển sang báo thức tiếp theo  lastSwitchTime = currentTime; // Cập nhật thời gian chuyển đổi  }  if (alarmCount > 0) {  // Định dạng dữ liệu thời gian từ báo thức hiện tại  snprintf(timeBuffer, sizeof(timeBuffer), "%s", alarms[currentDisplayAlarm].time);  // Chuyển đổi ghi chú từ "a", "b", "c", "d" thành chuỗi hiển thị đầy đủ trên LCD  if (strcmp(alarms[currentDisplayAlarm].note, "a") == 0) {  snprintf(noteBuffer, sizeof(noteBuffer), "BAT.PK");  } else if (strcmp(alarms[currentDisplayAlarm].note, "b") == 0) {  snprintf(noteBuffer, sizeof(noteBuffer), "TAT.PK");  } else if (strcmp(alarms[currentDisplayAlarm].note, "c") == 0) {  snprintf(noteBuffer, sizeof(noteBuffer), "BAT.PN");  } else if (strcmp(alarms[currentDisplayAlarm].note, "d") == 0) {  snprintf(noteBuffer, sizeof(noteBuffer), "TAT.PN");  } else {  snprintf(noteBuffer, sizeof(noteBuffer), "NA");  }  // Chỉ cập nhật vùng hiển thị báo thức trên LCD mà không xóa toàn bộ  lcd\_put\_cur(0, 5); // Vị trí thời gian ở dòng 1, cột 5  lcd\_send\_string(" "); // Xóa nội dung cũ  lcd\_put\_cur(0, 5);  lcd\_send\_string(timeBuffer); // Cập nhật thời gian mới  lcd\_put\_cur(1, 5); // Vị trí ghi chú ở dòng 2, cột 5  lcd\_send\_string(" "); // Xóa nội dung cũ  lcd\_put\_cur(1, 5);  lcd\_send\_string(noteBuffer); // Cập nhật ghi chú mới  } else {  // Hiển thị thông báo mặc định nếu không có báo thức  lcd\_put\_cur(0, 5);  lcd\_send\_string(" ");  lcd\_put\_cur(0, 5);  lcd\_send\_string("--:--:--");  lcd\_put\_cur(1, 5);  lcd\_send\_string(" ");  lcd\_put\_cur(1, 5);  lcd\_send\_string("NA");  }  }  void HAL\_UART\_RxCpltCallback(UART\_HandleTypeDef \*huart) {  if (huart->Instance == USART2) {  // Nhận ký tự từ UART và lưu vào buffer  char received\_char = rx\_buffer[index1];  // Kiểm tra nếu là lệnh điều khiển đơn khi buffer đang trống và ký tự không phải là chuỗi báo thức  if (index1 == 0 && received\_char != '$' &&  ((received\_char >= 'a' && received\_char <= 'd') || (received\_char >= 'e' && received\_char <= 'j'))) {  handleCommand(received\_char); // Xử lý lệnh điều khiển ngay lập tức  // Đặt lại chỉ số nhận và buffer để tránh xung đột  index1 = 0;  memset(rx\_buffer, 0, sizeof(rx\_buffer));  }  // Kiểm tra chuỗi dữ liệu báo thức bắt đầu bằng ký tự '$'  else if (received\_char == '$' && index1 == 0) {  // Nhận chuỗi báo thức, lưu ký tự '$' vào buffer  rx\_buffer[index1++] = received\_char;  }  // Nếu nhận được ký tự kết thúc là '#', xử lý chuỗi báo thức  else if (received\_char == '#') {  // Đặt ký tự kết thúc chuỗi  rx\_buffer[index1] = '\0';  // Xác nhận chuỗi bắt đầu bằng $  if (rx\_buffer[0] == '$') {  // Bỏ ký tự $ và #, sau đó xử lý phần dữ liệu bên trong  processAlarmData(&rx\_buffer[1]); // Truyền chuỗi mà không có ký tự $  }  // Đặt lại chỉ số nhận và buffer để chuẩn bị cho lần nhận tiếp theo  index1 = 0;  memset(rx\_buffer, 0, sizeof(rx\_buffer));  }  // Xử lý chuỗi có 1 hoặc 2 dấu phẩy  else if (received\_char == '\r' || received\_char == '\n') {  // Đặt ký tự kết thúc chuỗi  rx\_buffer[index1] = '\0';  // Đếm dấu phẩy để xác định loại dữ liệu  int comma\_count = 0;  for (int i = 0; i < index1; i++) {  if (rx\_buffer[i] == ',') {  comma\_count++;  }  }  // Xử lý dựa trên số lượng dấu phẩy  if (comma\_count == 1) {  ProcessReceivedCommaSeparatedString(rx\_buffer); // Xử lý chuỗi với 1 dấu phẩy  } else if (comma\_count == 2) {  ProcessReceivedRGBValues(rx\_buffer); // Xử lý chuỗi với 2 dấu phẩy  }  // Đặt lại chỉ số và buffer cho lần nhận tiếp theo  index1 = 0;  memset(rx\_buffer, 0, sizeof(rx\_buffer));  }  else {  // Lưu ký tự vào buffer và tăng chỉ số  rx\_buffer[index1] = received\_char;  index1 = (index1 + 1) % (sizeof(rx\_buffer) - 1); // Tránh tràn bộ nhớ  }  // Tiếp tục nhận dữ liệu từ UART  HAL\_UART\_Receive\_IT(&huart2, (uint8\_t \*)&rx\_buffer[index1], 1);  }  }  // Hàm xử lý giá trị RGB  void ProcessReceivedRGBValues(char \*rgbData) {  int red, green, blue;  sscanf(rgbData, "%d,%d,%d", &red, &green, &blue);  // Chuyển đổi giá trị RGB cho từng LED  for (int i = 0; i < 16; i++) {  setLED(i, red, green, blue);  }  ws2812Send(); // Gửi dữ liệu màu cho LED  }  void receiveDataFromSTM32() {  if (HAL\_UART\_Receive(&huart2, (uint8\_t \*)rx\_buffer, RX\_BUFFER\_SIZE, HAL\_MAX\_DELAY) == HAL\_OK) {  rx\_buffer[RX\_BUFFER\_SIZE - 1] = '\0'; // �?ảm bảo chuỗi kết thúc bằng null  // Goi hàm để xử lý chuỗi có dấu phẩy  ProcessReceivedCommaSeparatedString(rx\_buffer);  }  }  uint8\_t mapADCToPercentage(uint32\_t adcValue) {  float percentage = (adcValue \* 100.0f) / 4095.0f; // Tính phần trăm giá trị ADC  if (percentage < 10.0f) return 0; // Dưới 10%  else if (percentage < 20.0f) return 10; // Từ 10% đến dưới 20%  else if (percentage < 30.0f) return 20; // Từ 20% đến dưới 30%  else if (percentage < 40.0f) return 30; // Từ 30% đến dưới 40%  else if (percentage < 50.0f) return 40; // Từ 40% đến dưới 50%  else if (percentage < 60.0f) return 50; // Từ 50% đến dưới 60%  else if (percentage < 70.0f) return 60; // Từ 60% đến dưới 70%  else if (percentage < 80.0f) return 70; // Từ 70% đến dưới 80%  else if (percentage < 85.0f) return 80; // Từ 80% đến dưới 85%  else if (percentage < 90.0f) return 90; // Từ 85% đến dưới 90%  else return 100; // Trên 90%  }  /\* USER CODE END 0 \*/  /\*\*  \* @brief The application entry point.  \* @retval int  \*/  int main(void)  {  /\* USER CODE BEGIN 1 \*/  /\* USER CODE END 1 \*/  /\* MCU Configuration--------------------------------------------------------\*/  /\* Reset of all peripherals, Initializes the Flash interface and the Systick. \*/  HAL\_Init();  /\* USER CODE BEGIN Init \*/  /\* USER CODE END Init \*/  /\* Configure the system clock \*/  SystemClock\_Config();  /\* USER CODE BEGIN SysInit \*/  /\* USER CODE END SysInit \*/  /\* Initialize all configured peripherals \*/  MX\_GPIO\_Init();  MX\_DMA\_Init();  MX\_ADC1\_Init();  MX\_TIM1\_Init();  MX\_USART2\_UART\_Init();  MX\_TIM3\_Init();  MX\_I2C1\_Init();  /\* USER CODE BEGIN 2 \*/  HAL\_UART\_Receive\_IT(&huart2, (uint8\_t \*)&rx\_buffer[index1], 1);  lcd\_init();  // Initialize LCD with delay  HAL\_Delay(200); // Wait for system to stabilize  initializeLCD();  updateLCD();  /\* USER CODE END 2 \*/  /\* Infinite loop \*/  /\* USER CODE BEGIN WHILE \*/  while (1)  {  /\* USER CODE END WHILE \*/  /\* USER CODE BEGIN 3 \*/  while (1) {  uint32\_t currentTime = HAL\_GetTick(); // Lấy thời gian hiện tại  // Kiểm tra và xử lý phần ADC nếu adcActive và đủ thời gian  if (adcActive && (currentTime - lastAdcTime >= adcInterval)) {  lastAdcTime = currentTime; // Cập nhật thời gian thực thi  // Cấu hình ADC và đọc giá trị từ PA6 (ADC1 Channel 6)  ADC\_ChannelConfTypeDef sConfig = {0};  sConfig.Channel = ADC\_CHANNEL\_6;  sConfig.Rank = 1;  HAL\_ADC\_ConfigChannel(&hadc1, &sConfig);  HAL\_ADC\_Start(&hadc1);  HAL\_ADC\_PollForConversion(&hadc1, HAL\_MAX\_DELAY);  uint32\_t adcValue = HAL\_ADC\_GetValue(&hadc1);  HAL\_ADC\_Stop(&hadc1);  // Cấu hình ADC và đọc giá trị từ PA7 (ADC1 Channel 7)  sConfig.Channel = ADC\_CHANNEL\_7;  HAL\_ADC\_ConfigChannel(&hadc1, &sConfig);  HAL\_ADC\_Start(&hadc1);  HAL\_ADC\_PollForConversion(&hadc1, HAL\_MAX\_DELAY);  uint32\_t adcValue2 = HAL\_ADC\_GetValue(&hadc1);  HAL\_ADC\_Stop(&hadc1);  // Xác định ngưỡng và chuyển đổi giá trị ADC thành % độ sáng  uint8\_t brightnessPercent1 = mapADCToPercentage(adcValue);  uint8\_t brightnessPercent2 = mapADCToPercentage(adcValue2);  brightness2 = brightnessPercent1;  brightness1 = brightnessPercent2;  // Ánh xạ độ sáng thành giá trị PWM  uint32\_t pwmValue = (htim1.Init.Period \* brightness2) / 100;  uint32\_t pwmValue2 = (htim1.Init.Period \* brightness1) / 100;  // Đặt giá trị PWM cho TIM1 Channel 2 và Channel 1  \_\_HAL\_TIM\_SET\_COMPARE(&htim1, TIM\_CHANNEL\_2, pwmValue);  \_\_HAL\_TIM\_SET\_COMPARE(&htim1, TIM\_CHANNEL\_1, pwmValue2);  // Cập nhật độ sáng trên LCD  updateBrightnessOnLCD();  }  // Kiểm tra và xử lý phần updateLCD nếu đủ thời gian  if (currentTime - lastUpdateTime >= lcdInterval) {  lastUpdateTime = currentTime; // Cập nhật thời gian thực thi  updateLCD();  }  }  }  /\* USER CODE END 3 \*/  } |

**Code Arduino IDE- platform MIO**

|  |
| --- |
| **Code ARDUINO IDE- PLATFORM MIO** |
| #include <FirebaseESP32.h>  #include <HardwareSerial.h>  #include <WiFi.h>  #include <time.h>  #include <cstring>  /////////////////  HardwareSerial Serialport(2);  #define BUTTON\_PIN 2  /////////////////  FirebaseData fbdo;  FirebaseData fbdo2;  FirebaseData fbdo5;  FirebaseData fbdo6;  FirebaseData fbdo7;  FirebaseData fbdo8;  ///////////  FirebaseAuth auth;  FirebaseConfig config;  FirebaseData firebaseData;  /////////  String paths   = "/BT1";  String paths2  = "/BT2";  String paths5  = "/LB1";  String paths6  = "/LB2";  String paths7  = "/LEDRGB";  String paths8  = "/TIME";  ///////////////  // const char ssid[] = "Cà Phê Võng";  // const char pass[] = "";  // const char ssid[] = "GIANG L2";  // const char pass[] = "0393286953";  // const char ssid[] = "Nhà Thông Minh";  // const char pass[] = "0981805945";  // const char ssid[] = "Iphone 6s";  // const char pass[] = "12345678";  const char ssid[] = "NHOM12";  const char pass[] = "01234567";  //////////  ////////////////  bool dataButton1 = false;  bool dataButton2 = false;  int dataButton7 = 0;  int button1State ;  int button2State;  unsigned long lastDebounceTime = 0;  // Thời gian lần nhấn cuối cùng  unsigned long debounceDelay = 300;  volatile bool sendToFirebaseFlag = false;  volatile bool buttonPressed = false;  // Flag for Button press  bool lastButtonState = LOW;  bool buttonState = false;  char currentTimeStr[9]; // Để lưu thời gian dạng HH:MM:SS  char currentDateStr[11]; //  unsigned long previousMillis = 0;  // Thời gian trước đó  const long interval = 1000;  struct Alarm {      String time;  // Lưu thời gian báo thức dưới dạng chuỗi "HH:MM:SS"      String note;  // Lưu chuỗi note tương ứng  };  // Tạo mảng để lưu nhiều báo thức và note  Alarm alarms[20]; // Lưu tối đa 20 báo thức với các note khác nhau  int alarmCount = 0;  unsigned long currentMillis = millis(); // Lấy thời gian hiện tại  //////////  void printStreamData(StreamData data) {      int value;      Serial.print("Path: " + data.streamPath());      // Nhận dữ liệu từ Firebase      if (data.dataType() == "boolean") { // Kiểm tra kiểu dữ liệu là boolean          bool boolValue = data.boolData(); // Lấy giá trị boolean từ Firebase          value = boolValue ? 1 : 0;  // Chuyển true thành 1 và false thành 0          Serial.println("Giá trị nhận được (boolean): " + String(boolValue));      } else if (data.dataType() == "int") {          value = data.intData();          Serial.println("Giá trị nhận được (int): " + String(value));      } else if (data.dataType() == "string") {          String valueStr = data.stringData();          value = valueStr.toInt();  // Chuyển đổi chuỗi sang số nguyên nếu cần          Serial.println("Giá trị nhận được (string): " + valueStr);      } else if (data.dataType() == "json") {          // Lấy dữ liệu JSON từ Firebase          FirebaseJson jsonData = data.jsonObject();          // Khai báo biến để lưu giá trị hour, minute và note          FirebaseJsonData jsonHourData, jsonMinuteData, jsonNoteData;          int hour = 0;          int minute = 0;          String note = "";          // Kiểm tra và lấy giá trị từ Firebase cho giờ, phút và note          if (jsonData.get(jsonHourData, "/hour")) {              hour = jsonHourData.intValue;              Serial.println("Giờ: " + String(hour));          }          if (jsonData.get(jsonMinuteData, "/minute")) {              minute = jsonMinuteData.intValue;              Serial.println("Phút: " + String(minute));          }          if (jsonData.get(jsonNoteData, "/note")) {              note = jsonNoteData.stringValue;              Serial.println("Note: " + note);          }          // Định dạng thời gian thành "HH:MM:SS"          char buffer[9];          sprintf(buffer, "%02d:%02d:00", hour, minute);          String alarmTime = String(buffer);          // Lưu thời gian báo thức và note vào mảng  if (alarmCount < 20) { // Kiểm tra nếu mảng chưa đầy      alarms[alarmCount].time = alarmTime;      alarms[alarmCount].note = note;      alarmCount++;      Serial.println("Đã lưu báo thức: " + alarmTime + " với note: " + note);      // Chuyển đổi thời gian thành định dạng "HH,MM,SS"      String hour = alarmTime.substring(0, 2);      String minute = alarmTime.substring(3, 5);      String second = alarmTime.substring(6, 8);      // Chuyển đổi giá trị của note từ ký tự (a, b, c, d) thành số (1, 2, 3, 4)      String noteCommand;      if (note == "a") {          noteCommand = "1";      } else if (note == "b") {          noteCommand = "2";      } else if (note == "c") {          noteCommand = "3";      } else if (note == "d") {          noteCommand = "4";      } else {          noteCommand = "0"; // Trường hợp mặc định nếu không khớp      }      // Tạo chuỗi dữ liệu báo thức với định dạng "$HH,MM,SS,note#"      String command = "$," + hour + "," + minute + "," + second + "," + noteCommand + "#";      Serialport.print(command);      Serial.println("Đã gửi lệnh qua STM32: " + command);  } else {      Serial.println("Danh sách báo thức đã đầy!");  }      }          // Tiếp tục xử lý giá trị...          // Kiểm tra từng path và gửi ký tự tương ứng          if (data.streamPath() == paths) { // Kiểm tra SW1              char valueToSend = (value == 1) ? 'a' : 'b';  // Chuyển 1 thành 'a' và 0 thành 'b'              // Gửi ký tự tương ứng qua STM32              Serialport.print(valueToSend);              Serial.println("Gửi ký tự: " + String(valueToSend) + " qua STM32");          } else if (data.streamPath() == paths2) { // Kiểm tra SW2              char valueToSend = (value == 1) ? 'c' : 'd';  // Chuyển 1 thành 'c' và 0 thành 'd'              // Gửi ký tự tương ứng qua STM32              Serialport.print(valueToSend);              Serial.println("Gửi ký tự: " + String(valueToSend) + " qua STM32");          } else { // Đối với các đường dẫn khác              if (data.dataType() == "int") {                  value = data.intData();          } else if (data.dataType() == "string") {                  String valueStr = data.stringData();                  value = valueStr.toInt();  // Chuyển đổi chuỗi sang số nguyên nếu cần          } else {              Serial.print("Other data type");              return;              }              // Kiểm tra từng path và xử lý giá trị như trước          if (data.streamPath() == paths5 || data.streamPath() == paths6) { // Kiểm tra MUC SANG DEN 1 và 2              static int brightness1 = 0, brightness2 = 0;              if (value >= 0 && value <= 100) { // Chỉ nhận giá trị từ 0 đến 100                  if (data.streamPath() == paths5 && brightness1 != value) { // Chỉ gửi khi có sự thay đổi cho paths5                      brightness1 = value;                      Serial.println("Brightness1 (MUC SANG DEN 1): " + String(brightness1));                  } else if (data.streamPath() == paths6 && brightness2 != value) { // Chỉ gửi khi có sự thay đổi cho paths6                      brightness2 = value;                      Serial.println("Brightness2 (MUC SANG DEN 2): " + String(brightness2));                  }                      // Gửi giá trị từ cả MUC SANG DEN 1 và 2 dưới dạng chuỗi "brightness1,brightness2"                  Serialport.print(String(brightness1) + "," + String(brightness2) + "\n");                  Serial.println("Gửi giá trị: " + String(brightness1) + "," + String(brightness2) + " qua STM32");              } else {                  Serial.println("Invalid brightness value. Expected 0 to 100.");              }          } else if (data.streamPath() == paths7) { // Kiểm tra đường dẫn nhận dữ liệu              dataButton7 = value; // Lưu giá trị nhận được từ Firebase vào biến dataButton7.                  // In giá trị RGB ra Serial Monitor              Serial.println("RGB Data: " + String(dataButton7));              // Tách thành phần màu đỏ (R), xanh lá cây (G), và xanh dương (B)              int red = (dataButton7 >> 16) & 0xFF; // Tách thành phần màu đỏ              int green = (dataButton7 >> 8) & 0xFF; // Tách thành phần màu xanh lá cây              int blue = dataButton7 & 0xFF; // Tách thành phần màu xanh dương              Serial.println("R: " + String(red) + ", G: " + String(green) + ", B: " + String(blue));              // Gửi giá trị RGB đến STM32 dưới dạng chuỗi "R,G,B"              Serialport.print(String(red) + "," + String(green) + "," + String(blue) + "\n");              Serial.println("Gửi dữ liệu RGB: " + String(red) + "," + String(green) + "," + String(blue) + " qua STM32");          }     }  }  void streamCallback(StreamData data) {    Serial.println("Stream 1 received data!");    printStreamData(data);  }  void streamCallback2(StreamData data) {    Serial.println("Stream 2 received data!");    printStreamData(data);  }  void streamCallback5(StreamData data) {    Serial.println("Stream 5 received data!");    printStreamData(data);  }  void streamCallback6(StreamData data) {    Serial.println("Stream 6 received data!");    printStreamData(data);  }  void streamCallback7(StreamData data) {    Serial.println("Stream 7 received data!");    printStreamData(data);  }  void streamCallback8(StreamData data) {    Serial.println("Stream 8 received data!");    printStreamData(data);  }  void streamTimeoutCallback(bool timeout) {      if (timeout) {          Serial.println("Stream timeout, reconnecting...");          Firebase.beginStream(fbdo, paths);          Firebase.beginStream(fbdo2, paths2);          Firebase.beginStream(fbdo5, paths5);          Firebase.beginStream(fbdo6, paths6);          Firebase.beginStream(fbdo7, paths7);          Firebase.beginStream(fbdo8, paths8);      }  }  void receiveDataFromSTM32() {      if (Serialport.available()) {          String dataReceived = Serialport.readStringUntil('\n');          dataReceived.trim();          Serial.println(dataReceived);  // In dữ liệu đã nhận          // Kiểm tra giá trị hợp lệ          if (dataReceived == "0,0" || dataReceived == "0,1" || dataReceived == "1,0" || dataReceived == "1,1") {              int commaIndex = dataReceived.indexOf(',');              button1State = dataReceived.substring(0, commaIndex).toInt();              button2State = dataReceived.substring(commaIndex + 1).toInt();              // Chuyển đổi trạng thái nút từ int sang bool              bool button1Pressed = (button1State == 1);              bool button2Pressed = (button2State == 1);              // Cập nhật trạng thái của các nút              dataButton1 = button1Pressed;              dataButton2 = button2Pressed;              sendToFirebaseFlag = true;  // Kích hoạt biến cờ để gửi lên Firebase              Serial.println("button1Pressed: " + String(button1Pressed) + ", button2Pressed: " + String(button2Pressed));          } else {              Serial.println("Giá trị không hợp lệ! Chỉ chấp nhận 0,0; 0,1; 1,0; hoặc 1,1.");          }      }  }  void IRAM\_ATTR handleButtonPress() {      buttonPressed = true;  // Set flag when button is pressed  }  /////////////////  void setup(){      Serialport.begin(115200, SERIAL\_8N1, 16, 17);      Serial.begin(9600);      pinMode(BUTTON\_PIN, INPUT\_PULLDOWN);      Serial.print("Connecting to Wi-Fi");      WiFi.begin(ssid, pass);      while (WiFi.status() != WL\_CONNECTED)      {          Serial.print(".");      }      Serial.println();      Serial.print("Connected with IP: ");      Serial.println(WiFi.localIP());      Serial.println();    ///////////////////////      config.host = "iot-arduino-40f94-default-rtdb.asia-southeast1.firebasedatabase.app";      config.api\_key = "AIzaSyDglobI51-CImZeGcvW3uca3j399N1ArRE";      auth.user.email = "nhom12@gmail.com";      auth.user.password = "1234567";      Firebase.reconnectWiFi(true);      Firebase.begin(&config, &auth);      Firebase.beginStream(fbdo, paths);      Firebase.beginStream(fbdo2, paths2);      Firebase.beginStream(fbdo5, paths5);      Firebase.beginStream(fbdo6, paths6);      Firebase.beginStream(fbdo7, paths7);      Firebase.beginStream(fbdo8, paths8);      if(Firebase.ready()){      Serial.println("Connected with FIREBASE!");      }else{      Serial.println("Failed to connect with FIREBASE!");      }      Firebase.setStreamCallback(fbdo, streamCallback, streamTimeoutCallback);      Firebase.setStreamCallback(fbdo2, streamCallback2, streamTimeoutCallback);      Firebase.setStreamCallback(fbdo5, streamCallback5, streamTimeoutCallback);      Firebase.setStreamCallback(fbdo6, streamCallback6, streamTimeoutCallback);      Firebase.setStreamCallback(fbdo7, streamCallback7, streamTimeoutCallback);      Firebase.setStreamCallback(fbdo8, streamCallback8, streamTimeoutCallback);      attachInterrupt(digitalPinToInterrupt(2), receiveDataFromSTM32, CHANGE);      attachInterrupt(digitalPinToInterrupt(BUTTON\_PIN), handleButtonPress, CHANGE);      configTime(25200, 0, "pool.ntp.org", "time.nist.gov");  }    ///////////////  void loop() {      // Kiểm tra bộ nhớ      receiveDataFromSTM32();  // Nhận dữ liệu từ STM32      if (sendToFirebaseFlag) {          // Gửi dữ liệu lên Firebase          if (Firebase.setInt(firebaseData, "/BT1", dataButton1)) {              Serial.println("Cập nhật BT1 thành công");          } else {              Serial.println("Cập nhật BT1 thất bại");          }          if (Firebase.setInt(firebaseData, "/BT2", dataButton2)) {              Serial.println("Cập nhật BT2 thành công");          } else {              Serial.println("Cập nhật BT2 thất bại");          }          // Đặt lại biến cờ sau khi gửi xong          sendToFirebaseFlag = false;      }      if (buttonPressed) {          bool currentButtonState = digitalRead(BUTTON\_PIN);          // Kiểm tra xem trạng thái nút có thay đổi và đã qua thời gian debounce hay chưa          if (currentButtonState != lastButtonState && (millis() - lastDebounceTime) > debounceDelay) {              if (currentButtonState == HIGH) {                  Serialport.write('f');  // Gửi 'e' đến STM32 khi nút được nhấn (HIGH)                  Serialport.flush();      // Đảm bảo dữ liệu được gửi ngay lập tức                  Serial.println("Sent: f"); // In thông báo xác nhận              } else {                  Serialport.write('e');  // Gửi 'f' đến STM32 khi nút được thả (LOW)                  Serialport.flush();      // Đảm bảo dữ liệu được gửi ngay lập tức                  Serial.println("Sent: e"); // In thông báo xác nhận              }              lastButtonState = currentButtonState;  // Cập nhật trạng thái nút lần cuối              lastDebounceTime = millis();  // Cập nhật thời gian debounce          }          buttonPressed = false;  // Reset flag      }      // Lấy thời gian hiện tại      unsigned long currentMillis = millis(); // Lấy thời gian hiện tại      if (currentMillis - previousMillis >= interval) {          previousMillis = currentMillis; // Lưu thời gian hiện tại          struct tm timeinfo;          // Lấy thời gian          if (!getLocalTime(&timeinfo)) {              Serial.println("Không thể lấy thời gian");              return;          }          // Định dạng thời gian thành "HH:MM:SS"          snprintf(currentTimeStr, sizeof(currentTimeStr), "%02d:%02d:%02d",                   timeinfo.tm\_hour, timeinfo.tm\_min, timeinfo.tm\_sec);          // Định dạng ngày thành "DD/MM/YYYY"          snprintf(currentDateStr, sizeof(currentDateStr), "%02d/%02d/%04d",                   timeinfo.tm\_mday, timeinfo.tm\_mon + 1, timeinfo.tm\_year + 1900); // Cộng thêm 1900 cho năm          // In ra thời gian và ngày tháng          Serial.print("Thời gian hiện tại: ");          Serial.println(currentTimeStr);          Serial.print("Ngày hiện tại: ");          Serial.println(currentDateStr);          // Kiểm tra và kích hoạt báo thức         for (int i = 0; i < alarmCount; i++) {              if (alarms[i].time == String(currentTimeStr)) {                  // Báo thức kích hoạt                  Serial.println("Báo thức!!! Thời gian khớp với: " + alarms[i].time);                  // Gửi chuỗi `note` qua STM32                  Serialport.print(alarms[i].note);                  Serial.println("Gửi note qua STM32: " + alarms[i].note);                  if (alarms[i].note == "a") {                      Firebase.setBool(firebaseData, "/BT1", true);                      Serial.println("Gửi /BT1 = true lên Firebase");                      Serialport.print('g');  // Gửi 'g' khi note là 'a'                      Serial.println("Gửi ký tự 'g' qua STM32");                  } else if (alarms[i].note == "b") {                      Firebase.setBool(firebaseData, "/BT1", false);                      Serial.println("Gửi /BT1 = false lên Firebase");                      Serialport.print('h');  // Gửi 'h' khi note là 'b'                      Serial.println("Gửi ký tự 'h' qua STM32");                  } else if (alarms[i].note == "c") {                      Firebase.setBool(firebaseData, "/BT2", true);                      Serial.println("Gửi /BT2 = true lên Firebase");                      Serialport.print('i');  // Gửi 'i' khi note là 'c'                      Serial.println("Gửi ký tự 'i' qua STM32");                  } else if (alarms[i].note == "d") {                      Firebase.setBool(firebaseData, "/BT2", false);                      Serial.println("Gửi /BT2 = false lên Firebase");                      Serialport.print('j');  // Gửi 'j' khi note là 'd'                      Serial.println("Gửi ký tự 'j' qua STM32");                  }                  for (int j = i; j < alarmCount - 1; j++) {                      alarms[j] = alarms[j + 1];  // Dịch chuyển phần tử lên                  }                  alarmCount--; // Giảm số lượng báo thức                  i--; // Giảm chỉ số để không bỏ qua phần tử tiếp theo                }          }      }  } |

**Code chương trình ứng dụng ANDROID**

|  |
| --- |
| **Code chương trình ứng dụng ANDROID** |
| package com.example.light\_control;  import androidx.annotation.NonNull; import androidx.appcompat.app.AppCompatActivity;  import android.annotation.SuppressLint; import android.content.Intent; import android.os.Bundle; import android.view.View; import android.widget.Button; import android.widget.CompoundButton; import android.widget.ImageButton; import android.widget.SeekBar; import android.widget.Switch; import android.widget.TextView;  import com.google.firebase.database.DataSnapshot; import com.google.firebase.database.DatabaseError; import com.google.firebase.database.DatabaseReference; import com.google.firebase.database.FirebaseDatabase; import com.google.firebase.database.ValueEventListener;  import java.util.Objects;  public class MainActivity2 extends AppCompatActivity {  private TextView dongdien;  private DatabaseReference nhietdodata;  private TextView nangluongtieuthu;  private DatabaseReference kWhData;  private TextView dienap;  private DatabaseReference VoltData;  private TextView congsuat;  private DatabaseReference PowerData;  private TextView phantramden1;  private TextView phantramden2;  @SuppressLint("UseSwitchCompatOrMaterialCode")  private Switch swden1;  @SuppressLint("UseSwitchCompatOrMaterialCode")  private Switch swden2;  private SeekBar thanhden1;  private SeekBar thanhden2;  boolean sw1data\_status = false;  boolean sw2data\_status = false;  boolean isFirebaseUpdate = false; // private Button button; // private Button button2; // private Button button3;   private DatabaseReference den1\_data;  private DatabaseReference den2\_data;  private DatabaseReference sw1data;  private DatabaseReference sw2data; // private DatabaseReference btn2; // private DatabaseReference btn3;  private int dosang;   public MainActivity2() {  }   @SuppressLint("MissingInflatedId")  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.*activity\_main2*);  Button btnlogout = findViewById(R.id.*btnlogout*);  dongdien=findViewById(R.id.*AmpeValue*);  nangluongtieuthu=findViewById(R.id.*KWH*);  dienap=findViewById(R.id.*VOLT*);  congsuat=findViewById(R.id.*POWER*);  phantramden1=findViewById(R.id.*phantramden1*);  phantramden2=findViewById(R.id.*phantramden2*);  swden1=findViewById(R.id.*swden1*);  swden2=findViewById(R.id.*swden2*);  thanhden1=findViewById(R.id.*thanhden1*);  thanhden2=findViewById(R.id.*thanhden2*);  ImageButton btnToRGB = findViewById(R.id.*btnToRGB*);  ImageButton btnToclock = findViewById(R.id.*btnToclock*);   kWhData=FirebaseDatabase.*getInstance*().getReference().child("kWh");  sw1data=FirebaseDatabase.*getInstance*().getReference().child("BT1");  sw2data=FirebaseDatabase.*getInstance*().getReference().child("BT2");  // Xử lý đồng bộ nút switch 1  sw1data.addValueEventListener(new ValueEventListener() {  @Override  public void onDataChange(@NonNull DataSnapshot snapshot) {  if (snapshot.exists()) {  boolean value = Boolean.*TRUE*.equals(snapshot.getValue(boolean.class));  boolean switchStatus1 = (true == value);  swden1.setOnCheckedChangeListener(null);  swden1.setChecked(switchStatus1);  sw1data\_status = switchStatus1;  swden1.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {  @Override  public void onCheckedChanged(CompoundButton buttonView, boolean b) {  sw1data\_status=!sw1data\_status;  sw1\_onoff();  }  });  }  }  public void onCancelled(@NonNull DatabaseError error) {  //NOP  }  });  // Xử lý nút switch 1  swden1.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {  @Override  public void onCheckedChanged(CompoundButton compoundButton, boolean b) {  sw1data\_status=!sw1data\_status;  sw1\_onoff();  }  });  //Xử lý đồng bộ nút switch 2  sw2data.addValueEventListener(new ValueEventListener() {  @Override  public void onDataChange(@NonNull DataSnapshot snapshot) {  if (snapshot.exists()) {  boolean value2 = snapshot.getValue(boolean.class);  boolean switchStatus2 = (value2 == true);  swden2.setOnCheckedChangeListener(null);  swden2.setChecked(switchStatus2);  sw2data\_status = switchStatus2;  swden2.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {  @Override  public void onCheckedChanged(CompoundButton buttonView, boolean b) {  sw2data\_status=!sw2data\_status;  sw2\_onoff();  }  });  }  }  public void onCancelled(@NonNull DatabaseError error) {  //NOP  }  });  //Xử lý nút switch 2  swden2.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {  @Override  public void onCheckedChanged(CompoundButton compoundButton, boolean b) {  sw2data\_status=!sw2data\_status;  sw2\_onoff();  }  });  //Xử lí nút chuyển sang cài đặt RGB  btnToRGB.setOnClickListener(v -> {  Intent intent = new Intent(MainActivity2.this, MainActivity3.class);  startActivity(intent);  });  //Xử lí nt chuyển sang cài đặt hẹn giờ  btnToclock.setOnClickListener(v -> {  Intent intent = new Intent(MainActivity2.this, MainActivity4.class);  startActivity(intent);  });  // trượt thanh ngang đèn 1  den1\_data=FirebaseDatabase.*getInstance*().getReference().child("LB1");  den1\_data.addValueEventListener(new ValueEventListener() {  @Override  public void onDataChange(@NonNull DataSnapshot snapshot) {  if (snapshot.exists()) {  dosang = snapshot.getValue(Integer.class);  isFirebaseUpdate = true;  thanhden1.setProgress(dosang);  phantramden1.setText(dosang + "%");  }  }  public void onCancelled(@NonNull DatabaseError error) {  //NOP  }  });  thanhden1.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {  @Override  public void onProgressChanged(SeekBar seekBar, int i, boolean b) {  if(!isFirebaseUpdate){  den1\_data.setValue(i);  phantramden1.setText(i + "%");  }  isFirebaseUpdate = false;  } // if(i<10){ // den1\_data.child("MUC SANG DEN 1").setValue(0); // dosang=0; // phantramden1.setText(dosang+"%"); // }else if(i>=10 && i<20){ // den1\_data.child("MUC SANG DEN 1").setValue(10); // dosang=10; // phantramden1.setText(dosang+"%"); // }else if(i>10 && i<20){ // den1\_data.child("MUC SANG DEN 1").setValue(10); // dosang=10; // phantramden1.setText(dosang+"%"); // }else if(i>=20 && i<30){ // den1\_data.child("MUC SANG DEN 1").setValue(20); // dosang=20; // phantramden1.setText(dosang+"%"); // }else if(i>=30 && i<40){ // den1\_data.child("MUC SANG DEN 1").setValue(30); // dosang=30; // phantramden1.setText(dosang+"%"); // }else if(i>=40 && i<50){ // den1\_data.child("MUC SANG DEN 1").setValue(40); // dosang=40; // phantramden1.setText(dosang+"%"); // }else if(i>=50 && i<60){ // den1\_data.child("MUC SANG DEN 1").setValue(50); // dosang=50; // phantramden1.setText(dosang+"%"); // }else if(i>=60 && i<70){ // den1\_data.child("MUC SANG DEN 1").setValue(60); // dosang=60; // phantramden1.setText(dosang+"%"); // }else if(i>=70 && i<80){ // den1\_data.child("MUC SANG DEN 1").setValue(70); // dosang=70; // phantramden1.setText(dosang+"%"); // }else if(i>=80 && i<90){ // den1\_data.child("MUC SANG DEN 1").setValue(80); // dosang=80; // phantramden1.setText(dosang+"%"); // }else if(i>=90 && i<100){ // den1\_data.child("MUC SANG DEN 1").setValue(90); // dosang=90; // phantramden1.setText(dosang+"%"); // }else { // den1\_data.child("MUC SANG DEN 1").setValue(100); // dosang=100; // phantramden1.setText(dosang+"%"); // }///////////  @Override  public void onStartTrackingTouch(SeekBar seekBar) {  }  @Override  public void onStopTrackingTouch(SeekBar seekBar) {  }  });  // trượt thanh ngang đèn 2  den2\_data=FirebaseDatabase.*getInstance*().getReference().child("LB2");  den2\_data.addValueEventListener(new ValueEventListener() {  @Override  public void onDataChange(@NonNull DataSnapshot snapshot) {  if (snapshot.exists()) {  dosang = snapshot.getValue(Integer.class);  isFirebaseUpdate = true;  thanhden2.setProgress(dosang);  phantramden2.setText(dosang + "%");  }  }  public void onCancelled(@NonNull DatabaseError error) {  //NOP  }  });   thanhden2.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {  @Override  public void onProgressChanged(SeekBar seekBar, int i, boolean b){  if (!isFirebaseUpdate) {  den2\_data.setValue(i);  phantramden2.setText(i + "%");  }  isFirebaseUpdate = false;  } // if(i<10){ // den2\_data.child("MUC SANG DEN 2").setValue(0); // dosang=0; // phantramden2.setText(dosang+"%"); // }else if(i>=10 && i<20){ // den2\_data.child("MUC SANG DEN 2").setValue(10); // dosang=10; // phantramden2.setText(dosang+"%"); // }else if(i>10 && i<20){ // den2\_data.child("MUC SANG DEN 2").setValue(10); // dosang=10; // phantramden2.setText(dosang+"%"); // }else if(i>=20 && i<30){ // den2\_data.child("MUC SANG DEN 2").setValue(20); // dosang=20; // phantramden2.setText(dosang+"%"); // }else if(i>=30 && i<40){ // den2\_data.child("MUC SANG DEN 2").setValue(30); // dosang=30; // phantramden2.setText(dosang+"%"); // }else if(i>=40 && i<50){ // den2\_data.child("MUC SANG DEN 2").setValue(40); // dosang=40; // phantramden2.setText(dosang+"%"); // }else if(i>=50 && i<60){ // den2\_data.child("MUC SANG DEN 2").setValue(50); // dosang=50; // phantramden2.setText(dosang+"%"); // }else if(i>=60 && i<70){ // den2\_data.child("MUC SANG DEN 2").setValue(60); // dosang=60; // phantramden2.setText(dosang+"%"); // }else if(i>=70 && i<80){ // den2\_data.child("MUC SANG DEN 2").setValue(70); // dosang=70; // phantramden2.setText(dosang+"%"); // }else if(i>=80 && i<90){ // den2\_data.child("MUC SANG DEN 2").setValue(80); // dosang=80; // phantramden2.setText(dosang+"%"); // }else if(i>=90 && i<100){ // den2\_data.child("MUC SANG DEN 2").setValue(90); // dosang=90; // phantramden2.setText(dosang+"%"); // }else { // den2\_data.child("MUC SANG DEN 2").setValue(100); // dosang=100; // phantramden2.setText(dosang+"%"); // }////////////////  @Override  public void onStartTrackingTouch(SeekBar seekBar) {  }  @Override  public void onStopTrackingTouch(SeekBar seekBar) {  }  });   nhietdodata = FirebaseDatabase.*getInstance*().getReference();  final ValueEventListener nd\_dht11 = nhietdodata.child("AMPE").addValueEventListener(new ValueEventListener() {  @Override  public void onDataChange(@NonNull DataSnapshot snapshot) {  dongdien.setText(Objects.*requireNonNull*(snapshot.getValue()).toString() + " A");  }  @Override  public void onCancelled(@NonNull DatabaseError error) {  }  });   kWhData = FirebaseDatabase.*getInstance*().getReference();  final ValueEventListener watt = kWhData.child("kWh").addValueEventListener(new ValueEventListener() {  @Override  public void onDataChange(@NonNull DataSnapshot snapshot) {  nangluongtieuthu.setText(Objects.*requireNonNull*(snapshot.getValue()).toString() + " kWh");  }  @Override  public void onCancelled(@NonNull DatabaseError error) {  }  });   VoltData = FirebaseDatabase.*getInstance*().getReference();  final ValueEventListener volt = VoltData.child("VOLT").addValueEventListener(new ValueEventListener() {  @Override  public void onDataChange(@NonNull DataSnapshot snapshot) {  dienap.setText(Objects.*requireNonNull*(snapshot.getValue()).toString() + " V");  }  @Override  public void onCancelled(@NonNull DatabaseError error) {  }  });   PowerData = FirebaseDatabase.*getInstance*().getReference();  final ValueEventListener P = PowerData.child("WATT").addValueEventListener(new ValueEventListener() {  @Override  public void onDataChange(@NonNull DataSnapshot snapshot) {  congsuat.setText(Objects.*requireNonNull*(snapshot.getValue()).toString() + " W");  }  @Override  public void onCancelled(@NonNull DatabaseError error) {  }  });   // xu ly nut logout  btnlogout.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View view) {  finish();  }  });  }  private void sw2\_onoff() {  if(sw2data\_status){  sw2data.setValue(true);  }else {  sw2data.setValue(false);  }  }  private void sw1\_onoff() {  if(sw1data\_status){//sw1data\_status==true  sw1data.setValue(true);  }else {  sw1data.setValue(false);  }  } } |