**TRƯỜNG ĐẠI HỌC PHENIKAA**

**KHOA ĐIỆN – ĐIỆN TỬ**

A logo on a black background

AI-generated content may be incorrect.

**ĐỒ ÁN CHUYÊN NGÀNH**

**TÊN ĐỀ TÀI: THIẾT KẾ VÀ CHẾ TẠO BỘ ĐIỀU KHIỂN HỆ THỐNG ĐÈN TÍN HIỆU GIAO THÔNG**

Sinh viên thực hiện: Trần Sơn Dương Mã sinh viên: 21011224

Nguyễn Đình Khánh Mã sinh viên: 21012553

Nguyễn Công Khoa Mã sinh viên: 21011663

Lớp: K15 ĐK&TĐH1

Giảng viên hướng dẫn: ThS. Phạm Quang Vượng

*Hà Nội, 2025*

**MỤC LỤC**

[LỜI NÓI ĐẦU 1](#_Toc204282477)

[CHƯƠNG 1: TỔNG QUAN VÀ CÁC GIÁI PHÁP THỰC HIỆN 2](#_Toc204282478)

[1.1.Mở đầu 2](#_Toc204282479)

[1.2.Nguyên lý hoạt động của hệ thống đèn tín hiệu giao thông 2](#_Toc204282480)

[1.2.1.Cơ chế hoạt động của hệ thống đèn tín hiệu giao thông 2](#_Toc204282481)

[1.2.2.Các Bộ điều khiển đèn tín hiệu giao thông ngoài thị trường 3](#_Toc204282482)

[1.2.3.Các loại đèn giao thông thông dụng 3](#_Toc204282483)

[1.2.4.Bộ đếm đèn giao thông 5](#_Toc204282484)

[1.3.Cơ sở lý thuyết 7](#_Toc204282485)

[1.3.1.Module ESP32-WROOM-32D 7](#_Toc204282486)

[1.3.2.Module RTC DS3231 8](#_Toc204282487)

[1.3.3.Mạch hiển thị led 7 đoạn 74HC595 9](#_Toc204282488)

[1.3.4.Module LCD I2C 10](#_Toc204282489)

[1.3.5. Công tắc DIP switch 12](#_Toc204282490)

[1.3.6..MOSFET LR7843 13](#_Toc204282491)

[1.3.7.PC817 15](#_Toc204282492)

[1.3.8.LM2596 16](#_Toc204282493)

[1.3.9.Arduino IDE 17](#_Toc204282494)

[1.3.10.Visual Studio 17](#_Toc204282495)

[1.4.Giải pháp thực hiện 18](#_Toc204282496)

[1.4.1.Sử dụng ESP32 làm bộ điều khiển trung tâm 18](#_Toc204282497)

[1.4.2.Điều khiển theo chu kỳ và trạng thái 19](#_Toc204282498)

[1.4.3.Sử dụng 2 Module 74HC595 loại 2 led 7 thanh 19](#_Toc204282499)

[1.4.4.Sử dụng module thời gian thực RTC DS3231 19](#_Toc204282500)

[1.4.5.Hiển thị thông tin bằng module LCD I2C 19](#_Toc204282501)

[1.4.6.Giao diện giám sát trên máy tính bằng C# Window Form 19](#_Toc204282502)

[1.4.7.Tích hợp DIP switch để chuyển chế độ linh hoạt 20](#_Toc204282503)

[CHƯƠNG 2: THIẾT KẾ HỆ THỐNG 21](#_Toc204282504)

[2.1.Sơ đồ khối hệ thống 21](#_Toc204282505)

[2.1.1.Giao diện Winform 21](#_Toc204282506)

[2.1.2.Phần cứng 22](#_Toc204282507)

[2.1.3.Module RTC 22](#_Toc204282508)

[2.1.4.Bộ điều khiển trung tâm 22](#_Toc204282509)

[2.1.5.Output 22](#_Toc204282510)

[2.2.Lưu đồ thuật toán 23](#_Toc204282511)

[2.3.Thiết kế phần cứng 23](#_Toc204282512)

[2.3.1.Khối điều khiển trung tâm 23](#_Toc204282513)

[2.3.2.Khối nguồn 25](#_Toc204282514)

[2.3.3.Khối Input 26](#_Toc204282515)

[2.3.4.Khối Output 27](#_Toc204282516)

[2.4.Lập trình 30](#_Toc204282517)

[2.4.1.Lập trình trên Android IDE 30](#_Toc204282518)

[2.4.2.Lập trình Window Form 42](#_Toc204282519)

[CHƯƠNG 3: KẾT QUẢ THỰC HIỆN VÀ KẾT LUẬN 52](#_Toc204282520)

[3.1.Quá trình thử nghiệm và tối ưu hóa mạch điều khiển 52](#_Toc204282521)

[3.2.Kết quả thực nghiệm 54](#_Toc204282522)

[3.2.1.Bộ điều khiển đã hoàn thiện 54](#_Toc204282523)

[3.2.2.Quá trình thử nghiệm 54](#_Toc204282524)

[3.2.3.Kết quả đạt được 56](#_Toc204282525)

[3.3.Kết luận và hướng phát triển của đề tài 56](#_Toc204282526)

[3.3.1.Kết luận 56](#_Toc204282527)

[3.3.2.Hướng phát triển 56](#_Toc204282528)

[TÀI LIỆU THAM KHẢO 58](#_Toc204282529)

**MỤC LỤC HÌNH VẼ**

[Hình 1.1: Bộ điều khiển tín hiệu giao thông ATLC-C2 17](#_Toc204182900)

[Hình 1.2: Đèn giao thông model TL12-3 Light 18](#_Toc204182901)

[Hình 1.3: Bộ đếm đèn giao thông 19](#_Toc204182902)

[Hình 1.4: Module Esp32 – WROOM 32D 21](#_Toc204182903)

[Hình 1.5.Module RTC DS3231 22](#_Toc204182904)

[Hình 1.6: Module hiển thị 2 led 7 đoạn 74HC595 24](#_Toc204182905)

[Hình 1.7: Module hiển thị LCD I2C 25](#_Toc204182906)

[Hình 1.8: DIP switch 4 công tắc 27](#_Toc204182907)

[Hình 1.9: MOSFET LR7843 28](#_Toc204182908)

[Hình 1.10: PC817 29](#_Toc204182909)

[Hình 1.11: LM2596 30](#_Toc204182910)

[Hình 1.12: Arduino IDE 31](#_Toc204182911)

[Hình 1.13: Visual Studio 32](#_Toc204182912)

[Hình 2.1: Sơ đồ khối hệ thống 35](#_Toc204182913)

[Hình 2.2: Lưu đồ thuật toán 37](#_Toc204182914)

[Hình 2.3: Thiết kế khối điều khiển trung tâm 38](#_Toc204182915)

[Hình 2.4:Thiết kế khối nguồn 39](#_Toc204182916)

[Hình 2.5: Thiết kế sơ đồ chân kết nối của module RTC DS3231 trong khối Input 40](#_Toc204182917)

[Hình 2.6: Thiết kế sơ đồ nối chân của công tắc DIP Switch trong khối Input 41](#_Toc204182918)

[Hình 2.7: Thiết kế khối Optocoupler PC817 42](#_Toc204182919)

[Hình 2.8: Thiết kế sơ đồ đấu nối chân tại Module LCD I2C 43](#_Toc204182920)

[Hình 2.9: Thiết kế sơ đồ đấu nối với module LED 7 đoạn 44](#_Toc204182921)

[Hình 2.10: Giao diện Winform 57](#_Toc204182922)

[Hình 3.1: Mạch thử nghiệm lần thứ nhất 66](#_Toc204182923)

[Hình 3.2: Mặt trước của mạch thử nghiệm lần thứ hai 67](#_Toc204182924)

[Hình 3.3: Mặt sau của mạch thử nghiệm lần thứ hai 67](#_Toc204182925)

[Hình 3.4: Mạch thiết kết được đặt gia công từ các đối tác 68](#_Toc204182926)

[Hình 3.5: Bộ điều khiển ở chế độ Auto, thời gian đang là đêm khuya 69](#_Toc204182927)

[Hình 3.6: Giao diện Winform, hệ thống đang ở chế độ thủ công, thử nghiệm với trường hợp cảnh báo 69](#_Toc204182928)

**MỤC LỤC BẢNG BIỂU**

[Bảng 1.1: Thông số điện áp và dòng điện tiêu thụ cho từng màu ánh sáng 18](#_Toc204182929)

[Bảng 1.2: Thông số kỹ thuật của bộ đếm đèn giao thông 19](#_Toc204182930)

**PHIẾU ĐÁNH GIÁ QUÁ TRÌNH THỰC HIỆN ĐỒ ÁN**

Sinh viên thực hiện: Trần Sơn Dương Mã sinh viên: 21011224

Nguyễn Đình Khánh Mã sinh viên: 21012553

Nguyễn Công Khoa Mã sinh viên: 21011663

Giảng viên hướng dẫn: ThS. Phạm Quang Vượng

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TT** | **Ngày BC** | **Nội dung công việc** | | **Điểm danh** | **Kết quả %** | **Nhận xét** | **Đánh giá**  **(tối đa 10)** | **Người ĐG ký** |
| 1 | 07/05/2025 | Tìm hiểu về đề tài trong thực tế, hướng thực hiện đề tài. Phân chia công việc cho các thành viên. | | Đủ | 100% |  |  |  |
| 2 | 20/05/2025 | Đánh giá kết quả công việc, kiểm tra mạch thiết kế, đưa ra kế hoạch thực hiện tiếp theo. | | Đủ | 100% |  |  |  |
| 3 | 26/06/2025 | Đánh giá các kết quả của các mạch thử nghiệm, chốt thiết kế mạch thực tế, lên kế hoạch lập trình trên Winform | | Đủ | 100% |  |  |  |
| 4 | 16/07/2025 | Đánh giá kết quả thực tế của bộ điều khiển, hoàn thiện các vấn đề trong báo cáo | | Đủ | 100% |  |  |  |
| 9 |  | ***Điểm trung bình các lần báo cáo*** | |  |  |  |  |  |
| 10 |  | ***Số lần báo cáo*** | |  |  |  |  |  |
| 11 |  | ***Sản phẩm*** | |  |  |  |  |  |
| 12 |  | **Điểm thường xuyên** *(trung bình cộng các mục 9,10,11)* | |  |  |  |  |  |
| 13 |  | ***Điểm trình bày thuyết minh*** | |  |  |  |  |  |
|  |  |  | **Kết quả** *(trung bình cộng các mục 12,13)* | | | | | |

*Hà Nội, ngày tháng 07 năm 2025*

*Giảng viên hướng dẫn*

**KẾ HOẠCH THỰC HIỆN ĐỒ ÁN**

Tên đề tài: Thiết kế và chế tạo Bộ điều khiển hệ thống đèn tín hiệu giao thông

Sinh viên thực hiện: Trần Sơn Dương Mã sinh viên: 21011224

Nguyễn Đình Khánh Mã sinh viên: 21012553

Nguyễn Công Khoa Mã sinh viên: 21011663

Lớp: K15 ĐK&TĐH1

E\_mail: [21011224@st.phenikaa-uni.edu.vn](mailto:21011224@st.phenikaa-uni.edu.vn) ĐT: 0359036307

Giảng viên hướng dẫn: ThS. Phạm Quang Vượng

E\_mail: [vuong.phamquang@phenikaa-uni.edu.vn](mailto:vuong.phamquang@phenikaa-uni.edu.vn) ĐT: 0899.488.466

Kế hoạch thực hiện chi tiết:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **TT** | **NỘI DUNG** | **TG**  **(TUẦN)** | **ĐỊA ĐIỂM** | **TG**  **T.HIỆN** | **PHÂN CÔNG**  **NHIỆM VỤ** |
| 1 | Tìm hiểu   * Các sản phẩm đã được ứng dụng trong thực tế * Phân tích yêu cầu của đề tài * Thu thập các thông tin có liên quan * Các kiến thức cần có để phục vụ nghiên cứu đề tài * **Thiết kế sơ đồ khối thiết bị**   ***(Báo cáo giáo viên hướng dẫn)*** | 2 | Thư viện,  Phòng Thực hành 504, 503 | Từ ngày:  29/04/2025  đến ngày:  09/05/2025 | - Phân công nhóm trưởng.  - Cả nhóm cùng tham gia tìm hiểu và thảo luận nhóm  - Lập kế hoạch thực hiện  -Điện thoại: 0359036307  E\_mail: 21011224@st.phenikaa-uni.edu.vn |
| 2 | 1. Hoàn thiện sơ đồ khối thiết bị. Và phân tích chức năng các khối. 2. Chọn lựa giải pháp thực hiện 3. Phương pháp ghép nối giữa các khối với nhau. 4. **Thiết kế mạch nguyên lý các khối ,** phân tích chức năng các phần tử trong mạch và nguyên tắc làm việc của mạch điện. (viết tay) 5. Tính toán và lựa chọn các tham số của mạch điện (giá trị linh kiện, loại linh kiện sử dụng, điện áp, dòng điện, trong các mạch, công suất mạch, công suất nguồn….). 6. Chọn các linh kiện thực tế gần với các giá trị đã tính, Tính toán theo giá trị thực tế. 7. Viết thuyết minh báo cáo kết quả | 1 | Khoa Điện – Điện tử | Từ ngày:  10/05/2025  đến ngày:  20/05/2025 | Thảo luận nhóm  Phân công mỗi người phụ trách một khối công việc để làm việc  Trần Sơn Dương: 1,2,4  Nguyễn Đình Khánh: 4,5,6  Nguyễn Công Khoa: 3,5,7 |
| 3 | 1. Khảo sát mạch điện của thiết bị trên chương trình mô phỏng . Đồng thời thiết kế mạch 2. Làm mạch thực tế để thử nghiệm với mạch đã thiết kế 3. Hiệu chỉnh các tham số theo các giá trị tính toán. 4. Lập trình trên Arduino IDE chạy toàn mạch 5. Viết thuyết minh báo cáo kết quả sau khi khảo sát tực tế. | 6 |  | Từ ngày:  21/05/2025  đến ngày:  26/06/2025 | Trần Sơn Dương: 2,4,5  Nguyễn Đình Khánh: 1,2,3  Nguyễn Công Khoa: 3,4,5 |
| 4 | 1. Thiết kế mạch hoàn chỉnh 2. Đặt gia công mạch 3. Hàn mạch 4. Kiểm tra mạch + Hiệu chỉnh 5. Thiết kế phần mềm Winfrom 6. Hoàn thiện code Arduino cho toàn bộ dự án 7. Viết báo cáo sau khi kiểm tra hiệu chỉnh mạch. | 3 |  | Từ ngày:  27/06/2025  đến ngày:  16/07/2025 | Trần Sơn Dương; 5,6,7  Nguyễn Đình Khánh: 1,3,4,6  Nguyễn Công Khoa: 2,5,6 |
| 5 | 1. Lập phương án dự phòng. 2. Hướng phát triển và ứng dụng của đề tài 3. Hoàn thiện đề tài (thuyết minh, sản phẩm) 4. Hoàn thiện báo cáo, slide 5. Chuẩn bị bảo vệ. | 1 |  | Từ ngày:  17/07/2025  đến ngày:  24/07/2025 | - Cả nhóm |

|  |  |
| --- | --- |
|  | *Hà Nội, ngày tháng 07 năm 2025* |
| Giảng viên hướng dẫn | *Người lập* |

# LỜI NÓI ĐẦU

Sự phát triển như vũ bão của giao thông đô thị kéo theo nhu cầu quản lý và điều phối lưu lượng xe cộ một cách hiệu quả và an toàn. Trong đó, việc điều khiển hệ thống đèn tín hiệu giao thông đóng vai trò vô cùng quan trọng trong việc điều tiết phương tiện, giảm thiểu ùn tắc và tai nạn giao thông. Với sự phát triển của khoa học công nghệ hiện nay, việc tự động hóa và tối ưu hóa hoạt động của đèn tín hiệu giao thông đang trở nên cần thiết và cấp bách.

Xuất phát từ nhu cầu thực tiễn đó, nhóm chúng em thực hiện đề tài “Thiết kế và chế tạo Bộ điều khiển hệ thống đèn tín hiệu giao thông” nhằm mục tiêu xây dựng một mô hình điều khiển đèn giao thông hoạt động tự động, linh hoạt, dễ dàng lập trình và có khả năng mở rộng. Đề tài không chỉ giúp nhóm tiếp cận sâu hơn với kiến thức chuyên ngành về thiết kế mạch, lập trình hệ thống nhúng và kỹ thuật điều khiển, lập trình C# mà còn rèn luyện kỹ năng làm việc nhóm, phân tích hệ thống và triển khai ý tưởng vào thực tế.

Trong quá trình thực hiện đề tài, nhóm đã cố gắng hoàn thiện đề tài một cách nghiêm túc, từ khâu tìm hiểu lý thuyết, xây dựng giải pháp thiết kế đến việc làm mạch, lập trình, hoàn thiện sản phẩm. Tuy nhiên, do hạn chế về mặt thời gian và kinh nghiệm thực tiễn còn kém, chắc chắn đề tài vẫn còn một số điểm chưa hoàn thiện.

Nhóm xin chân thành cảm ơn sự hướng dẫn tận tình của thầy Phạm Quang Vượng cùng sự hỗ trợ từ các thầy cô trong bộ môn, đã giúp chúng em hoàn thành đề tài này. Chúng em cũng rất mong nhận được những ý kiến đóng góp quý báu từ thầy cô và bạn bè để hoàn thiện hơn đề tài trong tương lai.

# CHƯƠNG 1: TỔNG QUAN VÀ CÁC GIÁI PHÁP THỰC HIỆN

## 1.1.Mở đầu

Trong bối cảnh đô thị hóa và số lượng phương tiện giao thông ngày càng gia tăng, việc điều tiết lưu lượng giao thông một cách hiệu quả là yêu cầu cấp thiết nhằm đảm bảo an toàn, giảm thiểu ùn tắc và nâng cao chất lượng cuộc sống cho người dân. Hệ thống đèn tín hiệu giao thông là một trong những giải pháp phổ biến và hiệu quả nhất hiện nay để kiểm soát luồng xe tại các ngã tư, vòng xuyến và các điểm giao cắt quan trọng.

Tuy nhiên, nhiều hệ thống đèn tín hiệu hiện tại vẫn còn sử dụng phương pháp điều khiển thủ công hoặc các bộ điều khiển cứng với chu kỳ hoạt động cố định, không linh hoạt với sự thay đổi của lưu lượng giao thông. Điều này dẫn đến tình trạng kém hiệu quả trong phân phối thời gian đèn xanh - đỏ - vàng, gây lãng phí thời gian, nhiên liệu, tắc nghẽn thậm chí có thể gây tai nạn giao thông.

Xuất phát từ thực tế đó, đề tài “Thiết kế và chế tạo Bộ điều khiển hệ thống đèn tín hiệu giao thông” được thực hiện với mục tiêu xây dựng một hệ thống điều khiển đèn tín hiệu có tính tự động cao, linh hoạt trong lập trình thời gian chuyển đèn và có khả năng mô phỏng hoạt động của các ngã tư giao thông trong thực tế, có thể cài đặt thời gian với các trường hợp đặc biệt, tối ưu cho các ngã tư giao thông. Đề tài hướng đến việc ứng dụng các kiến thức về thiết kế mạch, vi điều khiển, lập trình để thiết kế và chế tạo một mô hình hoàn chỉnh, có thể mở rộng cho các ứng dụng thực tế hoặc tích hợp các công nghệ thông minh trong tương lai.

### 1.2.Nguyên lý hoạt động của hệ thống đèn tín hiệu giao thông

### 1.2.1.Cơ chế hoạt động của hệ thống đèn tín hiệu giao thông

Hệ thống đèn tín hiệu giao thông đóng vai trò điều tiết phương tiện tại các giao lộ. Mỗi hướng thường có ba đèn cơ bản: đỏ – vàng – xanh, trong đó:

* Đèn đỏ: yêu cầu phương tiện dừng lại;
* Đèn xanh: cho phép di chuyển;
* Đèn vàng: cảnh báo chuẩn bị chuyển pha;
* Đèn rẽ trái: hỗ trợ tách luồng rẽ trái tại các ngã tư có lưu lượng lớn.

Chu kỳ hoạt động sẽ luân phiên giữa các hướng, đảm bảo an toàn và trật tự. Hệ thống có thể điều khiển tự động theo khung thời gian nhất định đảm bảo cho việc điều chỉnh linh hoạt.

### 1.2.2.Các Bộ điều khiển đèn tín hiệu giao thông ngoài thị trường

Bộ điều khiển tín hiệu ATLC-C2 là một trong những bộ điều khiển đèn giap thông phổ biến trên thị trường



Hình 1.1: Bộ điều khiển tín hiệu giao thông ATLC-C2

Bộ điều khiển đèn tín hiệu ATLS-C2 có sự tích hợp firmware điều khiển TRAL, đầu vào/ đầu ra là 8/16, có khả năng tăng lên 16/32, 32/48, 48/64. Hỗ trợ nhiều kết nối mở rộng: 01 cổng RS232, 01 cổng RS485, 01 cổng RF (433MHz-9600). Điện áp hoạt động 12VDC và 24VDC, sử dụng lõi vi xử lý AVR 32bit, bộ nhớ dữ liệu RAM. Màn hình hiển thị LCD 16x2 có đèn nền, đồng bộ thời gian thực từ tín hiệu vệ tinh GPS, giám sát và cài đặt từ xa thông qua bộ kết nối không dây GSM. [1]

### 1.2.3.Các loại đèn giao thông thông dụng

Bộ đèn tín hiệu giao thông thường thấy được lắp đặt trên các ngã tư là loại đèn 3 LED xanh đỏ vàng

A close-up of a traffic light

AI-generated content may be incorrect.

Hình 1.2: Đèn giao thông model TL12-3 Light

Thông số chung của TL12 Series:

* Đường kính mỗi đèn: 12 inch (khoảng 305 mm)
* Số lượng LED mỗi bóng: 168 LED đảm bảo độ sáng cao
* Chất liệu vỏ: Vỏ nhôm bền chắc phù hợp dùng trong nhà và ngoài trời
* Chất liệu kính: Polycarbonate chịu được môi trường khắc nghiệt
* Tiêu chuẩn bảo vệ: IP55 (kháng nước và bụi)
* Tuổi thọ LED: Trên 50,000 giờ sử dụng
* Tiết kiệm năng lượng: Đèn tiết kiệm đến 90% điện năng so với đèn sợi đốt truyền thống
* Tuân thủ tiêu chuẩn: CE compliant
* Bảo hành: 2 năm [2]

Bảng 1.1: Thông số điện áp và dòng điện tiêu thụ cho từng màu ánh sáng [2]

| **Điện áp đầu vào** | **Dòng điện đỏ (Red)** | **Dòng điện vàng (Amber)** | **Dòng điện xanh (Green)** |
| --- | --- | --- | --- |
| 12 V DC | 950 mA | 950 mA | 750 mA |
| 110 V AC | 95 mA | 95 mA | 75 mA |
| 220 V AC | 45 mA | 45 mA | 37 mA |

### 1.2.4.Bộ đếm đèn giao thông

Trên thực tế, các bộ đếm đèn giao thông thường được đặt canh ngay bộ đèn giao thông để mọi người tiện theo dõi.

A close-up of a traffic sign

AI-generated content may be incorrect.

Hình 1.3: Bộ đếm đèn giao thông

Bảng 1.2: Thông số kỹ thuật của bộ đếm đèn giao thông [3]

| **STT** | **Thông số kỹ thuật** | **Chi tiết** |
| --- | --- | --- |
| 1 | **Model** | 5 phiên bản khác nhau: CD200 (đơn sắc, đa sắc), CD300 dạng 2 chữ số, 2 ½ chữ số và dạng vuông hoặc tròn |
| 2 | **Kích thước** | Đường kính 200 mm và 300 mm |
| 3 | **Kiểm tra trạng thái LED** | Mỗi phân đoạn gồm 2 nhánh LED |
| 4 | **Tự điều chỉnh độ sáng** | Có cảm biến bên trong để điều chỉnh tự động theo môi trường |
| 5 | **Số lượng LED** | - 256 LED với CD300 2 ½ chữ số  - 216 LED với CD300 2 chữ số  - 184 LED với CD200 đa sắc  - 68 LED với CD200 đơn sắc |
| 6 | **Loại LED** | High Flux LED |
| 7 | **Khoảng cách nhìn rõ** | - Dạng 200 mm: 60 m  - Dạng 300 mm: 100 m |
| 8 | **Màu sắc LED** | Đỏ, Vàng (Amber), Xanh lá (theo chuẩn EN12368:2006) |
| 9 | **Nguồn cấp điện** | 150 – 265 V AC, 47-63 Hz |
| 10 | **Công suất tiêu thụ** | 3 – 12 W |
| 11 | **Nhiệt độ làm việc** | -40°C đến +70°C (Class A, B, C theo EN12368:2006) |
| 12 | **Mức bảo vệ (Protection)** | IP65 theo EN 60598 (chống bụi, nước, thời tiết) |
| 13 | **EMC** | Tuân thủ EN50293 |

## 1.3.Cơ sở lý thuyết

### 1.3.1.Module ESP32-WROOM-32D

Kit ESP32 38 chân CP2102 TypeC - ESP32-WROOM-32D/ ESP32-WROOM-32U là KIT thu phát wifi, bluetooth dựa trên nền chip Wifi SoC ESP32-WROOM-D hoặc ESP32-WROOM-U và chip giao tiếp CP2102 mạnh mẽ. Được ứng dụng cho các thiết bị cần kết nối, thu thập dữ liệu và điều khiển qua sóng Wifi, qua Bluetooth đặc biệt là các ứng dụng liên quan đến IoT. Dễ dàng sử dụng trực tiếp trình biên dịch của Arduino IDE để lập trình và nạp code, điều này giúp cho việc sử dụng và lập trình các ứng dụng qua wifi, bluetooth trên ESP32 trở nên rất đơn giản.

A black circuit board with a silver and black chip

AI-generated content may be incorrect.

Hình 1.4: Module Esp32 – WROOM 32D

KIT sử dụng chip nạp và giao tiếp UART mới là CP2102 có khả năng tự nhận Driver trên tất cả các hệ điều hành Window và Linux.

Thông tin sản phẩm:

* IC chính: ESP32
* Chip nạp và giao tiếp UART: CP2102.
* Cổng nạp code: Cổng TypeC
* GPIO tương thích hoàn toàn với firmware Node MCU
* GIPO giao tiếp mức: 3.3VDC
* Nguồn sử dụng: 5VDC (qua MicroUSB hoặc Vin)
* Chuẩn Wifi: 802.11b/g/n/e/i
* Bluetooth: BR/EDR+BLE
* RAM: 520Kbytes
* ROM: 448Kbytes
* Tích hợp Led báo trạng thái, nút Reset, Flash.
* Tương thích hoàn toàn với trình biên dịch: Arduino IDE
* Các chuẩn giao tiếp : I²C, SPI, UART / USART, USB, CAN [4]

### 1.3.2.Module RTC DS3231

RTC có nghĩa là Đồng hồ thời gian thực . Mô-đun RTC chỉ đơn giản là các hệ thống ghi nhớ TIME và DATE có thiết lập pin giữ cho module được hoạt động trong trường hợp không có nguồn điện bên ngoài. Điều này giúp TIME và DATE luôn được cập nhật khi người dùng sử dụng



Hình 1.5.Module RTC DS3231

Các tính năng của MODULE DS3231 RTC

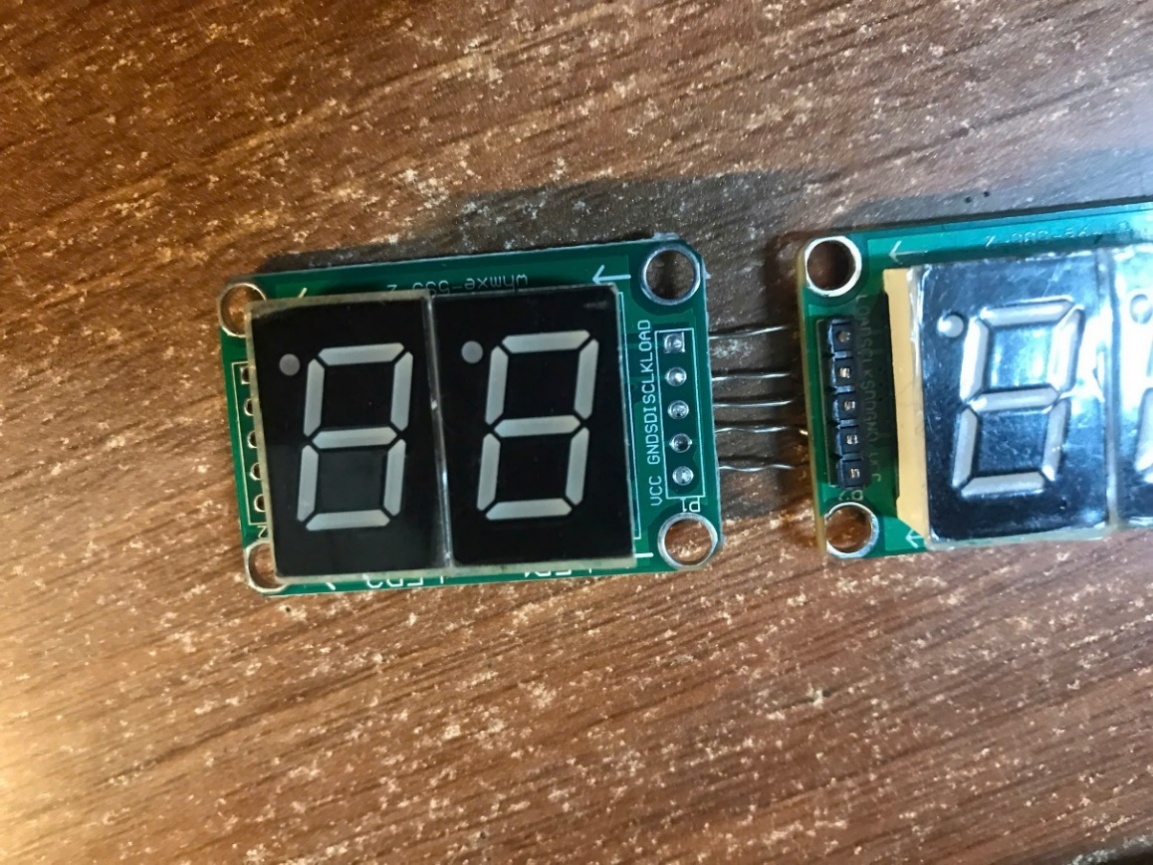
* RTC đếm giây, phút, giờ và năm
* Độ chính xác: + 2ppm đến -2ppm cho 0ºC đến + 40ºC, + 3,5ppm đến -3,5ppm cho -40ºC đến + 85ºC
* Cảm biến nhiệt độ kỹ thuật số với độ chính xác ± 3ºC
* Hai báo thức Thời gian trong ngày
* Đầu ra sóng vuông có thể lập trình
* Giao diện 400Khz I2C
* Sự tiêu thụ ít điện năng
* Mạch chuyển mạch pin tự động mất điện
* Pin dự phòng CR2032 với tuổi thọ từ hai đến ba năm

Thông số kỹ thuật MODULE DS3231 RTC

* Điện áp hoạt động của MODULE DS3231: 2.3V - 5.5V
* Có thể hoạt động trên điện áp THẤP
* Tiêu thụ 500nA khi pin dự phòng
* Điện áp tối đa tại SDA, SCL: VCC + 0,3V
* Nhiệt độ hoạt động: -45ºC đến + 80ºC. [5]

### 1.3.3.Mạch hiển thị led 7 đoạn 74HC595

Module này sử dụng chip 74HC595 để điều khiển các LED 7 đoạn một cách hiệu quả và đơn giản, giúp người dùng dễ dàng hiển thị các chữ số, ký tự, hay thậm chí là hình ảnh đơn giản. Module này giúp giảm số lượng chân IO cần dùng trên vi điều khiển, vì IC 74HC595 có khả năng ghi nối tiếp – xuất song song (Serial-In Parallel-Out Shift Register). [6]



Hình 1.6: Module hiển thị 2 led 7 đoạn 74HC595

Cấu tạo của Mạch hiển thị LED 7 đoạn 74HC595 anode chung

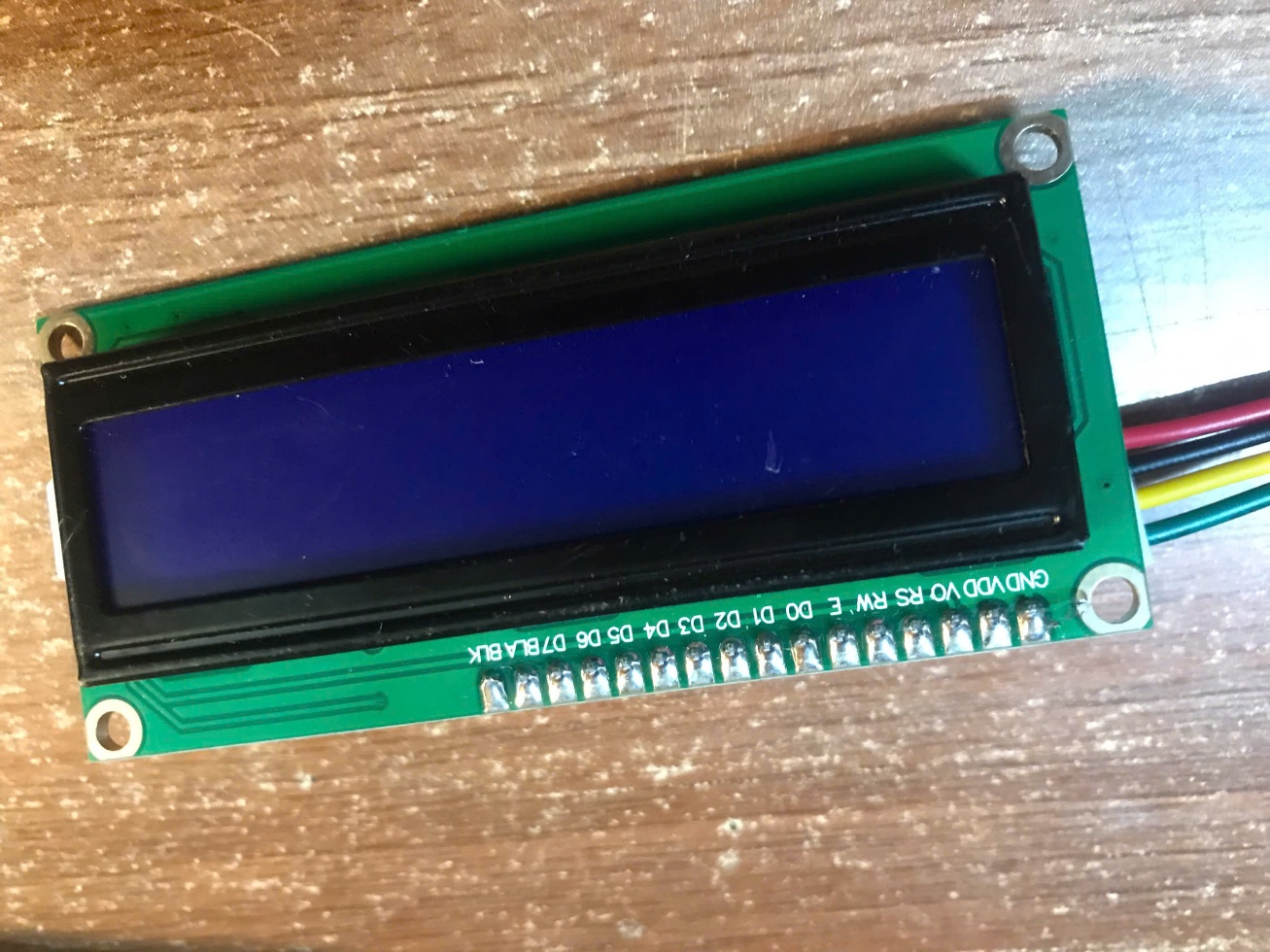
* Chip 74HC595: Đóng vai trò điều khiển và quản lý các LED 7 thanh, giúp hiển thị thông tin theo ý muốn của người dùng.
* Transistor: Sử dụng transistor để khuếch đại dòng điện cho các LED 7 thanh, đảm bảo độ sáng và ổn định khi hiển thị.
* Điện trở: Các điện trở được sử dụng để giới hạn dòng điện cho LED 7 thanh, bảo vệ chip 74HC595 và transistor.
* LED 7 thanh: LED 7 thanh kích thước 0.56 inch được sắp xếp theo cấu hình phù hợp để hiển thị 2, 3 hoặc 4 số.
* Đầu nối: Để kết nối với vi điều khiển.

Thông số kỹ thuật của mạch hiển thị LED 7 đoạn 74HC595

* IC sử dụng: 74HC595
* Nguồn sử dụng: 3- 6VDC.
* 3 chân giao tiếp: SCLK, RCLK, DIN.
* Điện năng tiêu thụ: 270 mW
* Nhiệt độ hoạt động: -30 ~ 70 độ C
* Storage Temperature: -40 ~ 85 độ C [7]

### 1.3.4.Module LCD I2C

Module LCD I2C giúp giảm số chân kết nối với vi điều khiển bằng cách dùng giao tiếp I2C chỉ với 2 dây (SCL, SDA) thay vì 6 hoặc 8 dây như cách truyền thống. Thông thường, các module này sử dụng màn hình LCD chuẩn HD44780 (như LCD 16x2, 20x4) tích hợp một mạch chuyển đổi I2C để điều khiển dễ dàng, tiết kiệm chân GPIO



Hình 1.7: Module hiển thị LCD I2C

Ưu điểm

* Tiết kiệm chân cho vi điều khiển.
* Dễ dàng kết nối với LCD.

Thông số kĩ thuật

* Tương thích với mạch Arduino hoặc các vi điều khiển/board khác hỗ trợ giao tiếp I2C.
* Loại hiển thị: Màn hình LCD với ký tự trắng trên nền đen (Negative white on Blue backlight).
* Địa chỉ I2C: 0x38 đến 0x3F (mặc định là 0x3F).
* Điện áp cung cấp: 5V DC.
* Giao tiếp: Giao tiếp I2C chuyển đổi thành 4 dây dữ liệu và các chân điều khiển LCD.
* Chỉnh độ tương phản: Có tích hợp sẵn biến trở để điều chỉnh trực tiếp trên module.
* Điều khiển đèn nền: Có thể điều chỉnh bằng phần mềm (firmware) hoặc qua dây nhảy (jumper wire).
* Kích thước bo mạch: 80 x 36 mm. [8]

### 1.3.5. Công tắc DIP switch

Trong thiết bị điện tử, DIP là viết tắt của "dual in-line package" khi được áp dụng cho các loại công tắc là viết tắt của 'gói nội tuyến kép'. Công tắc gói nội tuyến kép, hoặc công tắc DIP, là tập hợp các công tắc điện tử thủ công nhỏ được thiết kế để đóng gói với các mạch khác. Thuật ngữ công tắc DIP có thể đề cập đến một công tắc riêng lẻ trên một thiết bị nhiều công tắc hoặc toàn bộ thiết bị.

Vai trò của công tắc DIP là cho phép người dùng điều khiển dòng điện xung quanh bảng mạch in (PCB), thẻ mở rộng hoặc thiết bị ngoại vi điện tử/máy tính khác.

Khi được lắp đặt trên PCB cùng với các thành phần điện khác, công tắc DIP mang lại khả năng cho người dùng tùy chỉnh hoạt động của thiết bị, cho phép kiểm soát chính xác hoạt động mà thiết bị sẽ thực hiện. Công tắc DIP thường được thiết kế dưới dạng một dãy công tắc on-off nhỏ, giống như một loạt công tắc đèn tiêu chuẩn thu nhỏ, được gắn tuần tự trên một khối polyme. [9]

A red and white electrical device

AI-generated content may be incorrect.

Hình 1.8: DIP switch 4 công tắc

Cấu tạo DIP Switch

* Mỗi DIP Switch gồm nhiều công tắc nhỏ (switch)
* Có 2 trạng thái: ON (Đóng) chân tín hiệu được nối xuống GND (logic 0), OFF (Mở): chân tín hiệu được kéo lên mức cao (logic 1) nhờ điện trở pull-up.

Nguyên lý hoạt động

* Khi một công tắc bật ON, tín hiệu ở chân ra bị nối xuống mass (GND) → mức thấp (0).
* Khi công tắc OFF, điện trở pull-up (10kΩ) kéo tín hiệu lên mức cao (1).
* Vi điều khiển ESP32 sẽ đọc trạng thái các chân DIP để xác định chế độ hoạt động.

Các công tắc DIP thủ công cung cấp một tùy chọn tiết kiệm chi phí để điều khiển thiết bị và một tùy chọn cũng rất dễ dàng kết nối với mô hình thiết kế mạch PCB hoặc bảng mạch in hiện có. Công tắc DIP cũng cung cấp một cách nhanh chóng và thuận tiện để xem xét những cài đặt nào hiện được chọn mà không cần phải bật nguồn hệ thống trước và có thể giảm đáng phát sinh cho phần cứng bổ sung để thực hiện nhiều tác vụ đầu ra hơn.

### 1.3.6..MOSFET LR7843

MOSFET LR7843 là một loại MOSFET kênh N, điện trở dẫn thấp, dùng phổ biến trong các mạch điều khiển tải công suất như đèn, motor, relay…

**A black electronic device with white text

AI-generated content may be incorrect.**

Hình 1.9: MOSFET LR7843

Thông số cơ bản của LR7843

* Loại: MOSFET N-Channel.
* Điện áp Drain-Source tối đa (Vds): 30V.
* Dòng Drain tối đa (Id): 80A.
* Điện áp Gate-Threshold (Vgs(th)): khoảng 1V – 2.5V.
* Điện trở dẫn thấp (Rds(on)): rất nhỏ, chỉ khoảng 2-3 mΩ.
* Điều khiển được trực tiếp bằng mức tín hiệu logic 3.3V hoặc 5V (Logic-Level MOSFET). [10]

Cấu tạo chân MOSFET LR7843

* G (Gate): chân điều khiển tín hiệu (kích đóng/mở).
* D (Drain): chân nối với cực dương của tải (ví dụ đèn giao thông).
* S (Source): chân nối mass (GND) hoặc cực âm nguồn tải.

Nguyên lý hoạt động

* Khi có điện áp đủ lớn (thường >2.5V) đặt vào chân Gate, MOSFET dẫn điện (ON), dòng điện đi từ Drain qua Source → cấp điện cho tải.
* Khi không có điện áp kích Gate, MOSFET không dẫn (OFF), tải không được cấp điện.
* Điện trở dẫn rất nhỏ (Rds(on) thấp) giúp MOSFET ít tỏa nhiệt, chịu tải lớn mà không nóng nhiều. [10]

### 1.3.7.PC817

PC817 là một opto được sử dụng rất phổ biến, nó chứa một LED hồng ngoại và một transistor quang trong một gói. Opto hay còn được gọi là cách ly quang là những linh kiện dạng IC có từ 4 chân đến nhiều chân, chủ yếu được sử dụng để cách ly hai mạch với nhau.

Hoạt động của nó rất đơn giản, khi một điện áp được đặt vào LED hồng ngoại được nối trên chân 1 và 2, LED sẽ được kích hoạt và ánh sáng được nhận bởi transistor quang bên trong làm cho nó ở trạng thái bão hòa từ đó nối chân 3 và 4 với nhau. PC817 là một opto được sử dụng rộng rãi và hoạt động trong mạch điện tử chỉ với nhiệm vụ cách ly.

**A small black electronic device

AI-generated content may be incorrect.**

Hình 1.10: PC817

Thông số kỹ thuật PC817

* Loại gói: Dip 4 chân và SMT
* Loại transistor: NPN
* Dòng cực góp tối đa (IC): 50Ma
* Điện áp cực góp - cực phát tối đa (VCEO): 80V
* Điện áp bão hòa cực góp - cực phát: 0,1 đến 0,2
* Điện áp cực phát - cực gốc tối đa (VEBO): 6V
* Công suất tiêu tán cực góp tối đa (Pc): 200 mW
* Nhiệt độ lưu trữ và hoạt động phải là: -55 đến +120 độ C để lưu trữ và -30 đến +100 để hoạt động

Các ứng dụng

* Cách ly giữa hai mạch
* Đầu ra của vi điều khiển để điều khiển các thiết bị
* Mạch cảm biến
* Bộ cấp điện và bộ sạc
* Thiết bị văn phòng và đồ gia dụng
* Mạch kỹ thuật số [11]

### 1.3.8.LM2596

LM2596 là một IC nguồn switching regulator dạng buck converter (hạ áp), được dùng rất phổ biến để chuyển đổi điện áp DC từ mức cao xuống mức thấp hơn một cách hiệu quả

A close-up of a microchip

AI-generated content may be incorrect.

Hình 1.11: LM2596

Đặc điểm chính của LM2596

* Điện áp đầu vào (VIN): từ 4.5V đến 40V DC.
* Điện áp đầu ra (VOUT): có thể điều chỉnh từ 1.23V đến 37V (phiên bản Adjustable) hoặc cố định (ví dụ 5V).
* Dòng đầu ra tối đa: 3A.
* Hiệu suất chuyển đổi cao (lên đến 90%).
* Tần số chuyển mạch: khoảng 150 kHz.
* Bảo vệ quá nhiệt, quá dòng. [12]

### 1.3.9.Arduino IDE

Arduino IDE là một phần mềm với một mã nguồn mở, được sử dụng chủ yếu để viết và biên dịch mã vào module Arduino. Nó bao gồm phần cứng và phần mềm. Phần cứng chứa đến 300,000 board mạch được thiết kế sẵn với các cảm biến, linh kiện. Phần mềm giúp bạn có thể sử dụng các cảm biến, linh kiện ấy của Arduino một cách linh hoạt phù hợp với mục đích sử dụng [13]



Hình 1.12: Arduino IDE

Đây là phần mềm mã nguồn mở, miễn phí với tất cả mọi người. Sử dụng ngôn ngữ lập trình C/C++ thân thiện với các lập trình viên, với đa dạng các thư viện phong phú, hỗ trợ nhiều loại bo mạch như arduino, esp,..Đồng thời nó cũng có giao diện dễ dùng, hỗ trợ đa nền tảng từ Window đến Linux, MacOs.

### 1.3.10.Visual Studio

Visual Studio là một môi trường phát triển tích hợp (IDE) mạnh mẽ của Microsoft, được sử dụng phổ biến để lập trình C# Winform. Visual Studio hỗ trợ lập trình nhanh, trực quan với giao diện kéo thả, công cụ thiết kế Form đồ họa, trình gỡ lỗi, và tích hợp nhiều thư viện giúp phát triển ứng dụng Windows Forms dễ dàng.



Hình 1.13: Visual Studio

Dựa trên Visual Studio, nhóm đã thiết kế phần mềm Window Form để cài đặt thời gian các đèn sáng và chuyển giữa các chế độ của đèn giao thông.

## 1.4.Giải pháp thực hiện

### 1.4.1.Sử dụng ESP32 làm bộ điều khiển trung tâm

ESP32 được lựa chọn nhờ có:

* Số chân I/O dồi dào, phù hợp cho điều khiển nhiều LED;
* Tích hợp WiFi/Bluetooth (có thể mở rộng truyền thông trong tương lai);
* Hỗ trợ lập trình qua Arduino IDE, dễ tiếp cận và phát triển.

ESP32 chịu trách nhiệm chính trong:

* Điều khiển toàn bộ 8 đèn tín hiệu (4 hướng, mỗi hướng 4 đèn: đỏ, vàng, xanh, rẽ trái)
* Giao tiếp với RTC DS3231 để lấy thời gian thực
* Điều khiển 4 LED 7 thanh thông qua IC 74HC595 để hiển thị thời gian đếm ngược
* Hiển thị trạng thái hệ thống qua LCD I2C 16x2
* Đọc DIP switch để thay đổi chế độ hoạt động.

### 1.4.2.Điều khiển theo chu kỳ và trạng thái

Mỗi hướng luân phiên hoạt động theo trình tự: Xanh → Vàng → Đỏ;

Đèn rẽ trái được bật riêng theo thời gian quy định;

Hệ thống sử dụng các chế độ hoạt động linh hoạt:

* NORMAL: hoạt động bình thường;
* PRIORITY: ưu tiên hướng ngang hoặc dọc (tùy DIP);
* WARNING: bật nháy vàng cho cả 4 hướng;
* NIGHT: hoạt động ban đêm với chu kỳ rút gọn.

### 1.4.3.Sử dụng 2 Module 74HC595 loại 2 led 7 thanh

Để tiết kiệm chân I/O của ESP32, hệ thống sử dụng 2 module 74HC595 loại 2 led 7 thanh nối song song, hiển thị:

* Thời gian đếm ngược còn lại cho mỗi hướng;
* Dữ liệu được đẩy vào dạng mảng nhị phân tương ứng với từng chữ số.

Điều này giúp mô phỏng đúng hoạt động tại các ngã tư thực tế

### 1.4.4.Sử dụng module thời gian thực RTC DS3231

Module RTC giúp hệ thống:

* Lấy đúng thời gian thực để hiển thị và điều phối;
* Cho phép mở rộng lập lịch theo thời gian (giờ cao điểm, giờ thấp điểm…);
* Đồng bộ hóa giữa phần cứng và phần mềm giám sát (C#)

### 1.4.5.Hiển thị thông tin bằng module LCD I2C

Màn hình LCD 16x2 I2C hiển thị luân phiên:

* Thời gian thực lấy từ RTC;
* Trạng thái hoạt động: NORMAL, PRIORITY, WARNING, NIGHT;

Từ đó giúp người vận hành dễ dàng giám sát tình trạng hệ thống tại chỗ.

### 1.4.6.Giao diện giám sát trên máy tính bằng C# Window Form

ESP32 kết nối với máy tính qua Serial (UART) để giao tiếp với ứng dụng lập trình bằng C# WinForms. Phần mềm thực hiện các chức năng:

* Chọn cổng COM và kết nối với vi điều khiển;
* Điều chỉnh thời gian đèn: đèn xanh, vàng, rẽ trái (qua các trường nudGreenTime, nudYellowTime, nudLeftTime);
* Chuyển đổi chế độ: NIGHT MODE, WARNING MODE, ƯU TIÊN NGANG, ƯU TIÊN DỌC;
* Lưu cấu hình thời gian mới về ESP32 (qua nút “Save”);
* Hiển thị trạng thái hệ thống (qua txtStatus).

### 1.4.7.Tích hợp DIP switch để chuyển chế độ linh hoạt

4 công tắc DIP giúp thay đổi trực tiếp giữa các chế độ:

* NIGHT MODE – tiết kiệm năng lượng và giảm thời gian chờ;
* PRIORITY – ưu tiên luồng giao thông chính;
* WARNING – chuyển sang nháy vàng cảnh báo;
* NORMAL – hoạt động tuần tự như bình thường.

Giải pháp này cho phép thay đổi chế độ mà không cần kết nối máy tính, tiện lợi và linh hoạt.

# CHƯƠNG 2: THIẾT KẾ HỆ THỐNG

## 2.1.Sơ đồ khối hệ thống

A diagram of a computer

AI-generated content may be incorrect.

Hình 2.1: Sơ đồ khối hệ thống

Hệ thống điều khiển tín hiệu đèn được chia thành bốn khối chức năng chính: Giao diện Winform, Phần cứng, Trung tâm điều khiển, và Output. Quá trình hoạt động của hệ thống được mô tả như sau:

### 2.1.1.Giao diện Winform

Người dùng tương tác với hệ thống thông qua phần mềm Winform chạy trên máy tính. Giao diện này gồm các thành phần:

* Cổng COM để kết nối với vi điều khiển qua giao tiếp nối tiếp (serial).
* Nút nhấn để thực hiện kết nối.
* Các nút chọn chế độ hoạt động gồm: Chế độ ban đêm, chế độ ưu tiên đọc, chế độ ưu tiên ngang, chế độ cảnh báo
* Ô nhập thời gian cho từng pha đèn.
* Nút nhấn save để lưu thời
* Ô hiển thị kết quả phản hồi từ hệ thống.

Sau khi người dùng chọn chế độ và nhập thời gian, hệ thống sẽ gửi các lệnh tương ứng đến trung tâm điều khiển.

### 2.1.2.Phần cứng

Ngoài phần mềm, hệ thống còn có các switch vật lý gắn trên mạch để chọn chế độ hoạt động tương tự như trên Winform:

* 4 switch tương ứng với các chế độ: ban đêm, ưu tiên đọc, ưu tiên ngang, cảnh báo.
* Các switch này hoạt động song song với phần mềm, cho phép điều khiển thủ công trực tiếp khi không có phần mềm.

### 2.1.3.Module RTC

Module RTC sẽ tự động cập nhật dữ liệu thời gian thực, để từ đó gửi dữ liệu thời gian này về bộ điều khiển trung tâm. Từ đó để xác nhận khung thời gian riêng biệt để hiển thị đèn giao thông.

### 2.1.4.Bộ điều khiển trung tâm

Đây là bộ xử lý trung tâm (thường là vi điều khiển) có nhiệm vụ:

* Nhận lệnh điều khiển từ Winform hoặc tín hiệu từ các switch phần cứng.
* Xử lý thông tin chế độ và thời gian đèn đã được cài đặt.
* Ra lệnh điều khiển các đèn tín hiệu theo đúng logic và thời gian đã thiết lập.

### 2.1.5.Output

Dựa trên lệnh từ trung tâm điều khiển, hệ thống điều khiển đầu ra gồm:

* Đèn tín hiệu: hiển thị ba màu (xanh, vàng, đỏ) tùy theo chế độ hoạt động.
* LED 7 đoạn: hiển thị đếm ngược thời gian còn lại cho mỗi pha đèn.
* LCD: hiển thị trạng thái hoạt động của các chức năng và thời gian.

## 2.2.Lưu đồ thuật toán

A diagram of a flowchart

AI-generated content may be incorrect.

Hình 2.2: Lưu đồ thuật toán

Khi bắt đầu hệ thống sẽ khởi tạo các module đồng thời như ESP32, LCD, RTC và các chân I/O. Sau đó hệ thống sẽ kiểm tra xem có đang ở chế độ auto trên WinForm không, nếu đang ở chế độ auto thì hệ thống sẽ đọc dữ liệu thời gian thực từ RTC, qua đó hiển đèn giao thông theo các chế độ đã cài trước. Nếu không ở chế độ auto thì hệ thống sẽ đọc trạng thái của DIP Switch, từ đó xác định các chế độ hoạt động riêng biệt. Nếu chế độ WARNING hoạt động thì chớp vàng tất cả các đèn, nếu chế độ NIGHT hoạt động thì các đèn được bật nhanh đơn giản hơn, nếu chế độ PRIORITY hoạt động thì đèn giao thông được ưu tiên cho đèn xanh hướng chọn, nếu, ngoài các chế độ trên thì đèn sẽ hoạt động bật tắt theo chu kì thông thường.

## 2.3.Thiết kế phần cứng

### 2.3.1.Khối điều khiển trung tâm

Khối trung tâm của hệ thống sử dụng ESP32-DEVKITC-V4 loại 38 chân để làm bộ điều khiển chính. Nguồn cấp vào module thông qua IO 5V (Pin 19), lấy từ output của khối nguồn. GND (Pin 20) được nối chung mass với nguồn để đảm bảo mạch hoạt động ổn định.

A diagram of a computer

AI-generated content may be incorrect.

Hình 2.3: Thiết kế khối điều khiển trung tâm

Các IO tín hiệu được sử dụng trong hệ thống như sau:

* IO 32: xuất tín hiệu điều khiển đèn vàng ngang
* IO 33: xuất tín hiệu điều khiển đèn đỏ ngang
* IO 25: xuất tín hiệu điều khiển đèn xanh ngang
* IO 26: xuất tín hiệu điều khiển đèn rẽ trái ngang
* IO 27: xuất tín hiệu điều khiển đèn vàng dọc
* IO 14: xuất tín hiệu điều khiển đèn đỏ dọc
* IO 12: xuất tín hiệu điều khiển đèn xanh dọc
* IO 13: xuất tín hiệu điều khiển đèn rẽ trái dọc

Kết nối với các module ngoại vi:

* IO 22 và IO 21 được sử dụng để kết nối với module thời gian thực DS3231 và màn hình LCD, thông qua giao tiếp I2C.
* IO 19, IO 18, IO 5 được dùng để kết nối với module hiển thị Nex 7 thanh, phục vụ hiển thị số đếm hoặc trạng thái đèn giao thông.
* IO 17, IO 16, IO 4 và IO 2 lần lượt được sử dụng để nhận tín hiệu các chế độ điều khiển thông qua nút nhấn DIP Switch, cho phép người dùng lựa chọn các chế độ hoạt động khác nhau cho hệ thống giao thông.

### 2.3.2.Khối nguồn

Khối mạch nguồn hạ áp (DC-DC Buck Converter) sử dụng IC ổn áp LM2596-5.0, có nhiệm vụ chuyển đổi điện áp đầu vào từ 4v-40v thành điện áp ổn định 5V DC để cung cấp cho các khối mạch khác trong hệ thống.

A diagram of a circuit board

AI-generated content may be incorrect.

Hình 2.4:Thiết kế khối nguồn

Nguyên lý hoạt động của mạch

* Điện áp DC được cấp vào thông qua jack nguồn U27 (POWER-JACK).
* Từ jack nguồn, điện áp đi vào mạch qua cuộn cảm L1 (33uH).
* L1 (33uH): là cuộn cảm lọc, giúp giảm nhiễu xung và giữ dòng điện ổn định.
* Tụ điện U47 (680uF/25V) và U46 (680uF/25V): có nhiệm vụ lọc phẳng điện áp, giảm gợn sóng (ripple) ở đầu vào và đầu ra của mạch nguồn.
* D1 (1N5824): là diode Schottky, dẫn dòng trả về khi cuộn cảm xả năng lượng, đảm bảo mạch hoạt động ổn định trong chế độ chuyển mạch
* U48 (XH2596-5.0): là IC DC-DC Buck Converter, có khả năng giảm điện áp đầu vào xuống mức 5V ổn định ở đầu ra. Chân 1 (VIN) là ngõ vào điện áp DC từ cuộn cảm L1. Chân 2 (OUT) là điện áp đầu ra 5V ổn định. Chân 3 (GND) là nối mass. Chân 4 (FB) là chân phản hồi, dùng để ổn định điện áp đầu ra. Chân 5 (#ON/OFF) là chân điều khiển bật/tắt IC. Chân 6 (GND) nối với mass.
* Tín hiệu 5V sau khi ổn áp được đưa tới LED18, nối tiếp với điện trở R42 (330Ω) để giới hạn dòng điện.
* LED18 sẽ sáng khi có điện áp 5V, giúp báo hiệu nguồn đã hoạt động.

Kết quả đẩu ra ta được mạch cho ra điện áp ổn định 5V DC tại điểm OUT (điểm 5V trong sơ đồ), cấp nguồn cho các khối mạch khác. Mạch có hiệu suất cao nhờ sử dụng mạch hạ áp xung, đảm bảo ổn định điện áp 5V ngay cả khi điện áp đầu vào dao động, đồng thời có LED báo nguồn giúp người dùng quan sát tình trạng hoạt động.

### 2.3.3.Khối Input

Module thời gian thực RTC DS3231, giao tiếp với vi điều khiển thông qua chuẩn I2C.

A diagram of a circuit

AI-generated content may be incorrect.

Hình 2.5: Thiết kế sơ đồ chân kết nối của module RTC DS3231 trong khối Input

* Chân 1 (SCL): chân xung đồng hồ (Clock) của giao tiếp I2C.
* Chân 2 (SDA): chân dữ liệu của giao tiếp I2C.
* Chân 3 (5V): cấp nguồn 5V cho module.
* mChân 4 (GND): chân nối mass.

Module DS3231 có chức năng giữ thời gian thực với độ chính xác cao, giúp hệ thống điều khiển đèn giao thông hoạt động theo thời gian thực chính xác.

Khối DIP Switch có nhiệm vụ cho phép người dùng chọn các chế độ hoạt động khác nhau cho hệ thống điều khiển đèn giao thông.

A diagram of a circuit

AI-generated content may be incorrect.

Hình 2.6: Thiết kế sơ đồ nối chân của công tắc DIP Switch trong khối Input

* Gồm công tắc SW2 DIP có 4 switch nhỏ.
* Mỗi switch nối từ mass (GND) lên nguồn 3V3 thông qua các điện trở kéo lên (pull-up) R22, R23, R24, R25 có giá trị 10kΩ.
* Khi một switch đóng (ON), tín hiệu từ chân tương ứng sẽ được kéo xuống GND → tín hiệu mức thấp (0).
* Khi switch mở (OFF), nhờ điện trở pull-up, tín hiệu ở mức cao

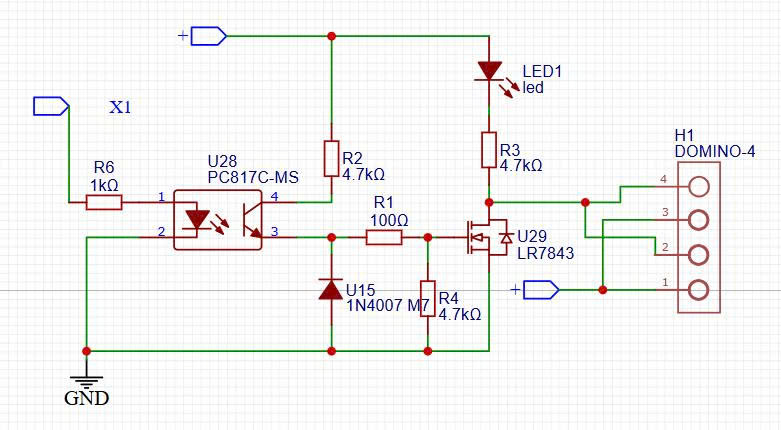
Cụ thể, các tín hiệu được lấy ra từ 4 chân DIP switch như sau:

* IO 2 (chân 8 SW2): chế độ A
* IO 4 (chân 7 SW2): chế độ B
* IO 16 (chân 6 SW2): chế độ C
* IO 17 (chân 5 SW2): chế độ D

Tín hiệu từ các chân này sẽ được đưa vào vi điều khiển ESP32 để xử lý và lựa chọn chế độ hoạt động phù hợp cho hệ thống giao thông (chế độ ban ngày/đêm, chế độ ưu tiên hướng dọc, chế độ ưu tiên hướng ngang, chế độ cảnh báo).

### 2.3.4.Khối Output

#### 2.3.4.1.Tín hiệu điều khiển vào Optocoupler PC817



Hình 2.7: Thiết kế khối Optocoupler PC817

Tín hiệu điều khiển từ ESP32 được đưa vào đầu X1. Qua điện trở hạn dòng R6 (1kΩ), dòng điện chạy qua LED trong optocoupler U28 (PC817C-MS), làm LED phát sáng. LED trong U28 phát sáng làm phototransistor phía trong dẫn điện. Điện áp ở chân 4 của U28 thay đổi, được kéo lên nguồn dương qua điện trở R2 (4.7kΩ). Tín hiệu từ chân 4 của opto đi qua điện trở R1 (100Ω) tới cổng (Gate) của MOSFET U29 (LR7843). Khi Gate có mức điện áp phù hợp, MOSFET bật, cho phép dòng điện chạy qua tải từ Drain xuống Source. R4 (4.7kΩ) là điện trở kéo xuống (pull-down) để đảm bảo MOSFET tắt hoàn toàn khi không có tín hiệu điều khiển. D1 (U15 – Diode 1N4007 M7) được mắc song song với tải để chống dòng điện ngược (flyback) sinh ra khi tải là cuộn cảm hoặc thiết bị cảm ứng, bảo vệ MOSFET khỏi hư hỏng. LED1 nối với nguồn qua điện trở R3 (4.7kΩ). Khi MOSFET dẫn, LED sáng để báo hiệu tải đang được kích hoạt. Đầu ra tải được đưa tới terminal Domino-4 (H1) với các chân, chân 1 nối với chân 3 là nguồn dương cấp vào mạch, chân 2 nối với 4 là các đường ra để kết nối đèn giao thông

Chức năng của khối mạch như việc cách ly tín hiệu điều khiển và phần công suất nhờ optocoupler giúp tăng độ an toàn, tránh nhiễu trở lại ESP32. Điều khiển tải dòng lớn thông qua MOSFET giúp phù hợp cho đóng cắt đèn giao thông hoặc các thiết bị có công suất vừa phải. Đèn LED giúp dễ dàng kiểm tra trạng thái hoạt động của mạch output.

#### 2.3.4.2.Module LCD I2C

A diagram of a circuit

AI-generated content may be incorrect.

Hình 2.8: Thiết kế sơ đồ đấu nối chân tại Module LCD I2C

Module LCD I2C là loại module sử dụng màn hình LCD giao tiếp qua giao thức I2C, giúp giảm số lượng dây kết nối so với LCD truyền thống.

* Chân 1 (SCL): chân xung đồng hồ (Clock) của giao tiếp I2C.
* Chân 2 (SDA): chân dữ liệu (Data) của giao tiếp I2C.
* Chân 3 (5V): cấp nguồn 5V cho module LCD.
* Chân 4 (GND): chân nối mass.

Module giúp hiển thị các thông tin cần thiết lên màn hình LCD

* Thời gian thực (đồng hồ từ DS3231).
* Trạng thái hoạt động của các đèn giao thông.
* Thông báo lỗi hoặc thông tin hệ thống.

#### 2.3.4.3.Module sử dụng LED 7 đoạn 74HC595

A diagram of a circuit

AI-generated content may be incorrect.

Hình 2.9: Thiết kế sơ đồ đấu nối với module LED 7 đoạn

Module LED 7 thanh này dùng đê hiển thị thời gian đếm ngược cho từng đèn xanh đỏ vàng, giúp chọ hệ thống có sự trực quan hóa nhất định.

* Tín hiệu điều khiển được nối từ các chân của esp32 gồm chân IO 19, IO 18, IO 5
* Nguồn nuôi module gồm chân 5V để cấp nguồn 5V cho LED 7 thanh, chân GND nối mass chung với hệ thống.

## 2.4.Lập trình

### 2.4.1.Lập trình trên Android IDE

Chương trình được viết trên nền tảng Arduino IDE, có chức năng điều khiển hệ thống đèn tín hiệu giao thông theo nhiều chế độ khác nhau. Người dùng có thể điều khiển hệ thống qua giao diện Winform hoặc switch phần cứng, trong đó có các chế độ như: bình thường, cảnh báo, ban đêm và ưu tiên. Ngoài ra, hệ thống còn hỗ trợ hiển thị thời gian thực bằng RTC và màn hình LCD I2C, đồng thời sử dụng IC 74HC595 để điều khiển LED 7 đoạn.

#### 2.4.1.1.Khai báo thư viện

Gọi các thư viện cần thiết để điều khiển LCD, đồng hồ thời gian thực (RTC), IC 74HC595 và lưu trữ thông tin

#include <Wire.h>

#include <LiquidCrystal\_I2C.h>

#include "RTClib.h"

#include <ShiftRegister74HC595.h>

#include <Preferences.h>

#### 2.4.1.2.Cấu hình các chân

Khai báo các chân được gán với từng module và chế độ

* DIP Switch dùng để chọn chế độ hoạt động (4 chân)
* Các chân điều khiển đèn giao thông (8 chân)
* 3 chân kết nối với IC 74HC595

// DIP switch pins

const int dip1 = 2;   // NIGHT MODE

const int dip2 = 4;   // PRIORITY DIRECTION (dọc)

const int dip3 = 16;  // PRIORITY HORIZONTAL (ngang)

const int dip4 = 17;  // WARNING MODE

// 74HC595 (4 IC)

ShiftRegister74HC595<4> sr(19, 18, 5);

// Traffic light pins

#define D\_YELLOW 32

#define D\_RED    33

#define D\_GREEN  25

#define D\_LEFT   26

#define N\_YELLOW 27

#define N\_RED    14

#define N\_GREEN  12

#define N\_LEFT   13

#### 2.4.1.3.Đọc cấu hình đèn từ bộ nhớ EEPROM (Preferences)

Thời gian đèn (xanh, vàng, rẽ trái) được cài đặt từ phần mềm Winform. Khi nhận lệnh "SET\_TIMES:...", các giá trị này sẽ được lưu vào bộ nhớ flash thông qua thư viện Preferences, đảm bảo dữ liệu vẫn giữ nguyên sau khi reset.

Preferences prefs; // Biến toàn cục cho Preferences

// Khởi động Preferences và đọc thời gian lưu

  prefs.begin("traffic", false);

  greenTime = prefs.getInt("green", 10);

  yellowTime = prefs.getInt("yellow", 3);

  leftTime = prefs.getInt("left", 3);

  redTime = greenTime + yellowTime + leftTime;

#### 2.4.1.4.Hiển thị thời gian thực và chế độ hoạt động

Hệ thống sử dụng chip RTC DS3231 để lấy thời gian thực. LCD I2C hiển thị giờ, ngày và chế độ hiện tại (NORMAL, NIGHT, PRIORITY, WARNING). Mỗi 5 giây, LCD sẽ luân phiên hiển thị thời gian và trạng thái hoạt động để cung cấp thông tin trực quan cho người dùng.

bool useDip = (manualMode && currentMode == "NORMAL");

  if (showTime) {

    lcd.setCursor(0, 0); lcd.print("Time:");

    if (now.hour() < 10) lcd.print("0"); lcd.print(now.hour()); lcd.print(":");

    if (now.minute() < 10) lcd.print("0"); lcd.print(now.minute()); lcd.print(":");

    if (now.second() < 10) lcd.print("0"); lcd.print(now.second());

    lcd.setCursor(0, 1); lcd.print("Date:");

    if (now.day() < 10) lcd.print("0"); lcd.print(now.day()); lcd.print("/");

    if (now.month() < 10) lcd.print("0"); lcd.print(now.month());

  } else {

    String modeDisplay = "NORMAL";

    if (currentMode == "WARNING" || (useDip && digitalRead(dip4) == LOW)) {

      modeDisplay = "WARNING ";

    } else if (currentMode == "NIGHT" || (useDip && digitalRead(dip1) == LOW)) {

      modeDisplay = "NIGHT   ";

    } else if (currentMode == "PRIORITY" || (useDip && (digitalRead(dip2) == LOW || digitalRead(dip3) == LOW))) {

      modeDisplay = "PRIORITY";

    }

    lcd.setCursor(0, 0);

    lcd.print("MODE: ");

    lcd.print(modeDisplay);

    lcd.setCursor(0, 1);

    if (modeDisplay == "PRIORITY") {

      lcd.print("PRIOR: ");

      if (currentMode == "PRIORITY") {

        lcd.print(priorityDir == "D" ? "VERTICAL " : "HORIZONTAL");

      } else {

        if (useDip && digitalRead(dip2) == LOW) lcd.print("DIRECTION ");

        else if (useDip && digitalRead(dip3) == LOW) lcd.print("HORIZONTAL");

        else lcd.print("UNKNOWN   ");

      }

    } else {

      lcd.setCursor(6, 1);

      lcd.print(manualMode ? "MANUAL" : "AUTO");

    }

  }

}

Cuối hàm loop(), chương trình sẽ tự điều chỉnh chế độ dựa theo giờ lấy từ RTC:

DateTime now = rtc.now();

#### 2.4.1.5.Chế độ tự động/ thủ công

Khi sử dụng chế độ tự động, bộ điều khiển sẽ tự hoạt động theo thời gian thực, hoạt động theo các khung giờ đã được thiết lập sẵn. Chế độ thủ công được thiết lập ra với mục đích cho việc kiểm thử hoạt động của bộ điều khiển, nhằm đảm bảo bộ điều khiển hoạt động tốt nhất. Khi ở chế độ thủ công thì lúc đó các chế độ như WARNING, NIGHT, PRIORITY sẽ được hoạt động dựa trên tác động của người dùng thông qua DIP Switch hoặc qua Winform app.

Khởi tạo biết điều khiển chế độ:

bool manualMode = false;

Giao tiếp Serial với Winform để chuyển đổi chế độ, Khi Winform gửi MANUAL\_MODE\_ON thì xác nhận bật chế độ thủ công. Khi Winform gửi: MANUAL\_MODE\_OFF thì xác nhận trở lại chế độ tự động.

else if (cmd == "MANUAL\_MODE\_ON") {

      manualMode = true;

      Serial.println("STATUS:MANUAL\_ON");

    }

    else if (cmd == "MANUAL\_MODE\_OFF") {

      manualMode = false;

      Serial.println("STATUS:MANUAL\_OFF");

    }

Xử lý logic chuyển chế độ bằng việc khởi tạo giá trị useDip

bool useDip = (manualMode && currentMode == "NORMAL");

Kiểm tra chế độ thủ công khi không bật DIP nào. Nếu chế độ thủ công đang bật, đồng thời tất cả các DIP (1–4) đều đang tắt (ở mức HIGH) và đang ở chế độ NORMAL thì không chọn chế độ nào cả đồng thời gọi handleManualIdle() để đèn tắt hoặc chờ

  if (manualMode &&

      digitalRead(dip1) != LOW &&

      digitalRead(dip2) != LOW &&

      digitalRead(dip3) != LOW &&

      digitalRead(dip4) != LOW &&

      currentMode == "NORMAL") {

    handleManualIdle();

    return;

  }

Thực hiện các chế độ dựa vào DIP hoặc dựa vào các trạng thái hiện tại:

bool mode\_night   = (currentMode == "NIGHT")   || (useDip && digitalRead(dip1) == LOW);

bool warning\_mode = (currentMode == "WARNING") || (useDip && digitalRead(dip4) == LOW);

bool priority\_d   = (currentMode == "PRIORITY" && priorityDir == "D") || (useDip && digitalRead(dip2) == LOW);

bool priority\_n   = (currentMode == "PRIORITY" && priorityDir == "N") || (useDip && digitalRead(dip3) == LOW);

Đồng thời sẽ tự động chuyển đổi chế độ theo giờ khi chế độ manualMode == False

 if (!manualMode) {

    DateTime now = rtc.now();

    int h = now.hour();

    if (currentMode == "NORMAL" || currentMode == "NIGHT" || currentMode == "WARNING") {

      if (h >= 6 && h < 22) currentMode = "NORMAL";

      else if (h >= 22 && h <= 23) currentMode = "NIGHT";

      else currentMode = "WARNING";

    }

  }

}

#### 2.4.1.5.Chế độ cảnh báo (WARNING)

Tại chế độ cảnh báo, đèn vàng nhấp nháy liên tục ở cả 2 hướng, đồng thời bỏ qua mọi logic điều khiển thời gian, thông báo qua LCD hiển thị cảnh báo.

Bật,tắt chế độ cảnh báo từ Serial, khi nhận chuỗi "WARNING\_MODE\_ON", chế độ sẽ chuyển sang "WARNING”. Khi nhận "WARNING\_MODE\_OFF", trở về "NORMAL"

else if (cmd == "WARNING\_MODE\_ON") {

      currentMode = "WARNING";

      Serial.println("STATUS:WARNING\_ON");

    }

    else if (cmd == "WARNING\_MODE\_OFF") {

      currentMode = "NORMAL";

      Serial.println("STATUS:WARNING\_OFF");

    }

Hiển thị trạng thái chế độ cảnh báo trên LCD trong hàm updateLCD():

} else {

    String modeDisplay = "NORMAL";

    if (currentMode == "WARNING" || (useDip && digitalRead(dip4) == LOW)) {

      modeDisplay = "WARNING ";

    }

...

lcd.setCursor(0, 0);

    lcd.print("MODE: ");

    lcd.print(modeDisplay);

}

Kích hoạt khi có lệnh WARNING\_MODE\_ON hoặc DIP switch tương ứng được bật.

if (warning\_mode) {

    allLightsOff();

    lcd.clear();

    lcd.setCursor(0, 0); lcd.print("CANH BAO: VANG");

    while ((currentMode == "WARNING") || (useDip && digitalRead(dip4) == LOW)) {

      digitalWrite(D\_YELLOW, HIGH);

      digitalWrite(N\_YELLOW, HIGH);

      displayCountdown(0, 0);

      updateLCD();

      delay(500);

      digitalWrite(D\_YELLOW, LOW);

      digitalWrite(N\_YELLOW, LOW);

      updateLCD();

      delay(500);

      if (Serial.available()) break;

    }

    lcd.clear();

    return;

  }

#### 2.4.1.6.Chế độ ban đêm (NIGHT)

Rút ngắn thời gian đèn xanh và vàng và tự động chuyển chế độ khi giờ hiện tại từ 22h đến 23h. Đọc tín hiệu từ DIP Switch để nhận diện chế độ NIGHT

bool mode\_night   = (currentMode == "NIGHT")   || (useDip && digitalRead(dip1) == LOW);

Khi ở chế độ NIGHT , thì hệ thống sẽ cài đặt thời gian đèn xanh chỉ còn 5s, đèn vàng 2s, đèn đỏ còn 3s

int gT = mode\_night ? 5 : greenTime;

int yT = mode\_night ? 2 : yellowTime;

Bật tắt chế độ ban đêm từ Serial, khi nhận "NIGHT\_MODE\_ON" thì chuyển currentMode = "NIGHT". Khi nhận "NIGHT\_MODE\_OFF" thì chuyển về "NORMAL"

else if (cmd == "NIGHT\_MODE\_ON") {

      currentMode = "NIGHT";

      Serial.println("STATUS:NIGHT\_ON");

    }

    else if (cmd == "NIGHT\_MODE\_OFF") {

      currentMode = "NORMAL";

      Serial.println("STATUS:NIGHT\_OFF");

    }

Đồng thời hiển thị chế độ NIGHT trên màn hình LCD trong hàm updateLCD()

 else if (currentMode == "NIGHT" || digitalRead(dip1) == LOW) {

      modeDisplay = "NIGHT   ";

    }

Tự động chuyển chế độ ban đêm theo giờ trong hàm loop()

if (!manualMode) {

    DateTime now = rtc.now();

    int h = now.hour();

    if (currentMode == "NORMAL" || currentMode == "NIGHT" || currentMode == "WARNING") {

      if (h >= 6 && h < 22) currentMode = "NORMAL";

      else if (h >= 22 && h <= 23) currentMode = "NIGHT";

      else currentMode = "WARNING";

    }

  }

#### 2.4.1.7.Chế độ ưu tiên (PRIORITY)

Nhận lệnh ưu tiên từ máy tính để bật tắt chế độ PRIORITY, khi nhận lệnh "PRIORITY\_MODE\_ON:D" thì bật chế độ ưu tiên cho hướng dọc. Khi nhận "PRIORITY\_MODE\_ON:N" thì bật ưu tiên cho hướng ngang. Khi nhận "PRIORITY\_MODE\_OFF" thì thoát khỏi chế độ ưu tiên và trở lại "NORMAL".

else if (cmd.startsWith("PRIORITY\_MODE\_ON:")) {

      currentMode = "PRIORITY";

      priorityDir = cmd.substring(cmd.lastIndexOf(":") + 1);

      Serial.println("STATUS:PRIORITY\_ON\_" + priorityDir);

    }

    else if (cmd == "PRIORITY\_MODE\_OFF") {

      currentMode = "NORMAL";

      priorityDir = "";

      Serial.println("STATUS:PRIORITY\_OFF");

    }

Khi phần mềm gửi lệnh PRIORITY\_MODE\_ON:D hoặc PRIORITY\_MODE\_ON:N thì chương trình bật chế độ PRIORITY và lưu hướng ưu tiên là D (Dọc) hoặc N (Ngang). Lệnh PRIORITY\_MODE\_OFF sẽ tắt chế độ và trở về trạng thái NORMAL.

Hiển thị trạng thái PRIORITY trên màn hình LCD trong hàm updateLCD()

else if (currentMode == "PRIORITY" || (useDip && (digitalRead(dip2) == LOW || digitalRead(dip3) == LOW))) {

      modeDisplay = "PRIORITY";

    }

...

lcd.setCursor(0, 0);

    lcd.print("MODE: ");

    lcd.print(modeDisplay);

    lcd.setCursor(0, 1);

    if (modeDisplay == "PRIORITY") {

      lcd.print("PRIOR: ");

      if (currentMode == "PRIORITY") {

        lcd.print(priorityDir == "D" ? "VERTICAL " : "HORIZONTAL");

      } else {

        if (useDip && digitalRead(dip2) == LOW) lcd.print("DIRECTION ");

        else if (useDip && digitalRead(dip3) == LOW) lcd.print("HORIZONTAL");

        else lcd.print("UNKNOWN   ");

      }

    } else {

      lcd.setCursor(6, 1);

      lcd.print(manualMode ? "MANUAL" : "AUTO");

    }

Hiển thị “PRIORITY” ở dòng 1 của LCD, còn hiển thị hướng ưu tiên là VERTICAL (dọc) hoặc HORIZONTAL (ngang) ở dòng 2.

Xác định hướng ưu tiên từ biến hoặc từ công tắc DIP Switch

bool priority\_d   = (currentMode == "PRIORITY" && priorityDir == "D") || (useDip && digitalRead(dip2) == LOW);

bool priority\_n   = (currentMode == "PRIORITY" && priorityDir == "N") || (useDip && digitalRead(dip3) == LOW);

Hướng dọc (priority\_d) được ưu tiên nếu chế độ là PRIORITY và priorityDir == "D" từ phần mềm hoặc DIP2 (ưu tiên dọc) được gạt xuống. Hướng ngang (priority\_n) tương tự với priorityDir == "N" hoặc DIP3.

Trường hợp ưu tiên dọc:

if (priority\_d && !priority\_n) {

    allLightsOff();

    digitalWrite(D\_GREEN, HIGH);

    digitalWrite(N\_RED, HIGH);

    displayCountdown(0, 0);

    updateLCD();

while ((currentMode == "PRIORITY" && priorityDir == "D") || (useDip && digitalRead(dip2) == LOW)) {

      delay(100);

      updateLCD();

      if (Serial.available()) return;

    }

    lcd.clear();

    return;

  }

Trường hợp ưu tiên ngang:

if (priority\_n && !priority\_d) {

    allLightsOff();

    digitalWrite(N\_GREEN, HIGH);

    digitalWrite(D\_RED, HIGH);

    displayCountdown(0, 0);

    updateLCD();

    while ((currentMode == "PRIORITY" && priorityDir == "N") || (useDip && digitalRead(dip3) == LOW)) {

      delay(100);

      updateLCD();

      if (Serial.available()) return;

    }

    lcd.clear();

    return;

  }

Điều chỉnh thời gian xanh để ưu tiên hướng đã chọn

 int gT\_d = gT, gT\_n = gT;

 if (priority\_d && !priority\_n) gT\_n = 0;

 if (priority\_n && !priority\_d) gT\_d = 0;

Nếu chỉ ưu tiên Dọc thì gT\_n = 0 (bỏ qua đèn xanh của hướng ngang). Nếu chỉ ưu tiên Ngang thì giá trị gT\_d = 0. Nếu cả 2 không ưu tiên thì bộ đèn vẫn hoạt động bình thường.

#### 2.4.1.8.Điều khiển chu kỳ đèn

Điều khiển đèn giao thông tại ngã tư cho hai hướng chính là dọc và ngang theo các pha

* Rẽ trái – Đèn xanh – Đèn vàng – Đèn đỏ
* Xử lý các chế độ đặc biệt như NIGHT, WARNING, PRIORITY

Vòng lặp chính loop() thực hiện điều khiển chu kỳ theo điều kiện thời gian (với delay). Thời gian đèn phụ phụ thuộc vào các chế độ, chế độ NIGHT thì thời gian đèn xanh đỏ được rút gọn, ở chế độ PRIORITY thì bỏ qua pha xanh của hướng không ưu tiên, ở chế độ WARNING thì chỉ hiện thị đèn vàng nhấp nháy. Đồng thời LED 7 đoạn cũng hiển thị đếm ngược đồng thời cùng với việc thời gian bật tắt của đèn.

Pha 1 khi hướng dọc hoạt động:

if (gT\_d > 0) {

    int totalRedN = gT\_d + yT + leftTime;

    allLightsOff();

    digitalWrite(D\_LEFT, HIGH);

    digitalWrite(D\_RED, HIGH);

    digitalWrite(N\_RED, HIGH);

    for (int i = leftTime; i >= 1; i--) {

      displayCountdown(i, totalRedN--);

      updateLCD();

      delay(1000);

      if (Serial.available()) return;

    }

    allLightsOff();

    digitalWrite(D\_GREEN, HIGH);

    digitalWrite(N\_RED, HIGH);

    for (int i = gT\_d; i >= 1; i--) {

      displayCountdown(i, totalRedN--);

      updateLCD();

      delay(1000);

      if (Serial.available()) return;

    }

    allLightsOff();

    digitalWrite(D\_YELLOW, HIGH);

    digitalWrite(N\_RED, HIGH);

    for (int i = yT; i >= 1; i--) {

      displayCountdown(i, totalRedN--);

      updateLCD();

      delay(1000);

      if (Serial.available()) return;

    }

  }

Pha 2 khi hướng ngang hoạt động:

  if (gT\_n > 0) {

    int totalRedD = gT\_n + yT + leftTime;

    allLightsOff();

    digitalWrite(N\_LEFT, HIGH);

    digitalWrite(N\_RED, HIGH);

    digitalWrite(D\_RED, HIGH);

    for (int i = leftTime; i >= 1; i--) {

      displayCountdown(totalRedD--, i);

      updateLCD();

      delay(1000);

      if (Serial.available()) return;

    }

    allLightsOff();

    digitalWrite(N\_GREEN, HIGH);

    digitalWrite(D\_RED, HIGH);

    for (int i = gT\_n; i >= 1; i--) {

      displayCountdown(totalRedD--, i);

      updateLCD();

      delay(1000);

      if (Serial.available()) return;

    }

    allLightsOff();

    digitalWrite(N\_YELLOW, HIGH);

    digitalWrite(D\_RED, HIGH);

    for (int i = yT; i >= 1; i--) {

      displayCountdown(totalRedD--, i);

      updateLCD();

      delay(1000);

      if (Serial.available()) return;

    }

  }

#### 2.4.1.9.Giao tiếp với phần mềm Winform

Cho phép phần mềm Winform gửi lệnh xuống ESP32, cập nhật thời gia của từng pha đèn, điều khiển bật tắt các chế độ NIGHT, WARNING, PRIORITY.

if (Serial.available()) {

    String cmd = Serial.readStringUntil('\n');

    cmd.trim();

Trích xuất các giá trị thời gian từ chuỗi lệnh, sau đó gán cho các biến. Ghi giá trị mới vào bộ nhớ EEPROM. Đồng thời nhận phản hồi từ ESP32 để xác nhận trạng thái.

if (cmd.startsWith("SET\_TIMES:")) {

      int p1 = cmd.indexOf(":") + 1;

      int p2 = cmd.indexOf(":", p1);

      int p3 = cmd.indexOf(":", p2 + 1);

      greenTime = cmd.substring(p1, p2).toInt();

      yellowTime = cmd.substring(p2 + 1, p3).toInt();

      leftTime = cmd.substring(p3 + 1).toInt();

      redTime = greenTime + yellowTime + leftTime;

      prefs.putInt("green", greenTime);

      prefs.putInt("yellow", yellowTime);

      prefs.putInt("left", leftTime);

      Serial.println("STATUS:TIMES\_SET");

    }

Xử lý các lệnh bật tắt của các chế độ

else if (cmd == "WARNING\_MODE\_ON") {

      currentMode = "WARNING";

      Serial.println("STATUS:WARNING\_ON");

    }

    else if (cmd == "WARNING\_MODE\_OFF") {

      currentMode = "NORMAL";

      Serial.println("STATUS:WARNING\_OFF");

    }

    else if (cmd == "NIGHT\_MODE\_ON") {

      currentMode = "NIGHT";

      Serial.println("STATUS:NIGHT\_ON");

    }

    else if (cmd == "NIGHT\_MODE\_OFF") {

      currentMode = "NORMAL";

      Serial.println("STATUS:NIGHT\_OFF");

    }

    else if (cmd.startsWith("PRIORITY\_MODE\_ON:")) {

      currentMode = "PRIORITY";

      priorityDir = cmd.substring(cmd.lastIndexOf(":") + 1);

      Serial.println("STATUS:PRIORITY\_ON\_" + priorityDir);

    }

    else if (cmd == "PRIORITY\_MODE\_OFF") {

      currentMode = "NORMAL";

      priorityDir = "";

      Serial.println("STATUS:PRIORITY\_OFF");

    }

### 2.4.2.Lập trình Window Form

#### 2.4.2.1.Giao diện và hiển thị thông tin trạng thái

A screenshot of a computer

AI-generated content may be incorrect.

Hình 2.10: Giao diện Winform

Giao diện phần mềm được xây dựng bằng WinForms trên Visual Studio với bố cục trực quan, dễ sử dụng. Các thành phần chính gồm:

* ComboBox comboBoxPort: chọn cổng COM để kết nối với ESP32.
* TextBox txtStatus: hiển thị toàn bộ thông tin phản hồi từ ESP32, giúp người dùng theo dõi trạng thái giao tiếp.
* Các Button điều khiển: cho phép gửi lệnh bật/tắt các chế độ: Ban đêm, Cảnh báo, Ưu tiên Dọc, Ưu tiên Ngang.
* Các NumericUpDown: để nhập giá trị thời gian cho các pha đèn xanh, vàng, rẽ trái.
* Label lblRedTime: hiển thị thời gian đèn đỏ được tính tự động.

Phần mềm cũng cập nhật lại trạng thái hiển thị cho người dùng mỗi khi có sự thay đổi từ phía thiết bị hoặc người dùng thao tác. Các nút điều khiển có thay đổi tên động để phản ánh trạng thái hiện tại.

#### 2.4.2.2.Kết nối và giao tiếp với ESP32

Phần mềm WinForms được thiết kế để kết nối và giao tiếp với vi điều khiển ESP32 thông qua giao thức UART sử dụng cổng COM. Khi người dùng chọn đúng cổng COM từ comboBoxPort và nhấn nút Connect, chương trình sẽ mở kết nối serial với tốc độ 115200 baud, đồng thời gán sự kiện DataReceived để xử lý dữ liệu phản hồi từ ESP32.

Thông tin phản hồi (trạng thái, xác nhận lệnh, thông báo lỗi) sẽ được hiển thị trực tiếp trên hộp văn bản txtStatus. Người dùng có thể chủ động ngắt kết nối khi cần, và hệ thống cũng tự động đóng cổng COM khi thoát phần mềm nhằm tránh lỗi treo cổng.

Kết nối đến ESP32:

private void btnConnect\_Click(object sender, EventArgs e)

{

try

{

if (serialPort == null || !serialPort.IsOpen)

{

if (comboBoxPort.SelectedItem == null)

{

MessageBox.Show("Vui lòng chọn cổng COM!");

return;

}

txtStatus.AppendText($"Đang thử kết nối tới {comboBoxPort.SelectedItem}...\r\n");

serialPort = new SerialPort(comboBoxPort.SelectedItem.ToString(), 115200);

serialPort.DataReceived += SerialPort\_DataReceived;

serialPort.Open();

btnConnect.Text = "Disconnect";

txtStatus.AppendText("Đã kết nối với ESP32!\r\n");

}

else

{

serialPort.Close();

btnConnect.Text = "Connect";

txtStatus.AppendText("Đã ngắt kết nối!\r\n");

}

}

catch (Exception ex)

{

txtStatus.AppendText($"Lỗi kết nối: {ex.Message}\r\n");

MessageBox.Show($"Lỗi kết nối: {ex.Message}", "Lỗi", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

Xử lý dữ liệu nhận được:

private void SerialPort\_DataReceived(object sender, SerialDataReceivedEventArgs e)

{

try

{

string data = serialPort.ReadLine();

this.Invoke((MethodInvoker)delegate

{

txtStatus.AppendText(data + "\r\n");

});

}

catch (Exception ex)

{

this.Invoke((MethodInvoker)delegate

{

txtStatus.AppendText("Lỗi nhận dữ liệu: " + ex.Message + "\r\n");

});

}

}

Ngắt kết nối khi đóng form:

protected override void OnFormClosing(FormClosingEventArgs e)

{

if (serialPort != null && serialPort.IsOpen)

serialPort.Close();

base.OnFormClosing(e);

}

Gửi các lệnh chuyển chế độ:

serialPort.WriteLine("WARNING\_MODE\_ON");

serialPort.WriteLine("WARNING\_MODE\_OFF");

serialPort.WriteLine("PRIORITY\_MODE\_ON:D");

serialPort.WriteLine("PRIORITY\_MODE\_OFF");

serialPort.WriteLine("PRIORITY\_MODE\_ON:N");

serialPort.WriteLine("PRIORITY\_MODE\_OFF");

serialPort.WriteLine("MANUAL\_MODE\_ON");

serialPort.WriteLine("MANUAL\_MODE\_OFF");

serialPort.WriteLine("NIGHT\_MODE\_ON");

serialPort.WriteLine("NIGHT\_MODE\_OFF");

Gửi thông tin cấu hình thời gian lên ESP32

string command = $"SET\_TIMES:{greenTime}:{yellowTime}:{leftTime}";

serialPort.WriteLine(command);

#### 2.4.2.3.Thiết lập thời gian cho từng pha đèn

Người dùng có thể điều chỉnh thời gian cho ba pha đèn: đèn xanh, đèn vàng, và đèn rẽ trái thông qua các điều khiển nudGreenTime, nudYellowTime, nudLeftTime. Tổng thời gian đèn đỏ sẽ được tính tự động và hiển thị trên lblRedTime.

Khi người dùng nhấn nút btnSave, chương trình sẽ đóng gói dữ liệu thành chuỗi lệnh định dạng:

SET\_TIMES:greenTime:yellowTime:leftTime

Lệnh này được gửi đến ESP32 qua Serial, giúp vi điều khiển cập nhật và lưu các giá trị thời gian mới vào bộ nhớ (Preferences)

private void TimeValueChanged(object sender, EventArgs e)

{

UpdateRedTime();

}

Hàm UpdateRedTime() bạn đưa ra có chức năng tính và hiển thị tổng thời gian đèn đỏ dựa trên 3 giá trị của đèn vàng, đèn xanh, đèn rẽ trái.

private void UpdateRedTime()

{

decimal yellow = nudYellowTime.Value;

decimal green = nudGreenTime.Value;

decimal left = nudLeftTime.Value;

lblRedTime.Text = (yellow + green + left).ToString();

}

Gửi lệnh thiết lập thời gian xuống ESP32

private void btnSave\_Click(object sender, EventArgs e)

{

if (serialPort != null && serialPort.IsOpen)

{

int yellowTime = (int)nudYellowTime.Value;

int greenTime = (int)nudGreenTime.Value;

int leftTime = (int)nudLeftTime.Value;

int redTime = yellowTime + greenTime + leftTime;

if (redTime > 0 && yellowTime > 0 && greenTime > 0)

{

string command = $"SET\_TIMES:{greenTime}:{yellowTime}:{leftTime}";

serialPort.WriteLine(command);

txtStatus.AppendText($"Gửi: {command}\n");

}

else

{

MessageBox.Show("Thời gian phải lớn hơn 0!");

}

}

else

{

MessageBox.Show("Chưa kết nối với ESP32!");

}

}

#### 2.4.2.4.Điều khiển chế độ thủ công/tự động

Khai báo biến xác định chế độ mặc định để ở chế độ tự động

private bool isAutoMode = true;

Hàm xử lý khi người dùng bấm nút chuyển chế độ:

private void btnManualMode\_Click(object sender, EventArgs e)

{

if (serialPort != null && serialPort.IsOpen)

{

if (isAutoMode)

{

serialPort.WriteLine("MANUAL\_MODE\_ON");

txtStatus.AppendText("Gửi: MANUAL\_MODE\_ON\n");

btnManualMode.Text = "Tự động";

ToggleManualControls(true); // Bật các nút điều khiển

}

else

{

serialPort.WriteLine("MANUAL\_MODE\_OFF");

txtStatus.AppendText("Gửi: MANUAL\_MODE\_OFF\n");

btnManualMode.Text = "Thủ công";

ToggleManualControls(false); // Tắt các nút điều khiển

}

isAutoMode = !isAutoMode;

}

else

{

MessageBox.Show("Chưa kết nối với ESP32!");

}

}

Khi người dùng nhấn nút btnManualMode, nếu đang ở chế độ tự động (isAutoMode == true) thì gửi lệnh "MANUAL\_MODE\_ON" đến ESP32, chuyển sang chế độ thủ công và đồng thời đổi tên nút thành "Tự động", bật các nút điều khiển. Ngược lại, nếu đang ở chế độ thủ công, khi nhấn button thì gửi "MANUAL\_MODE\_OFF" và đổi tên nút thành “Thủ công” đồng thời tắt các nút điều khiển.

Bật/Tắt các nút chức năng trong chế độ thủ công, hàm này sẽ được gọi mỗi khi thay đổi chế độ, để bật/tắt các nút tương ứng với việc người dùng có thể tự tay điều khiển đèn giao thông hay không.

private void ToggleManualControls(bool enable)

{

btnNightMode.Enabled = enable;

btnWarningMode.Enabled = enable;

btnPriorityD.Enabled = enable;

btnPriorityN.Enabled = enable;

btnSave.Enabled = enable;

}

#### 2.4.2.5.Điều khiển chế độ ban đêm

Chế độ Ban đêm (Night Mode) được thiết kế để giảm thời gian đèn sáng vào buổi tối, giúp giao thông linh hoạt hơn khi lượng xe thưa thớt. Người dùng có thể bật/tắt chế độ này thủ công bằng cách nhấn nút "Ban Đêm" trên giao diện WinForms.

Khi nút được nhấn, phần mềm sẽ gửi lệnh tương ứng qua Serial cho ESP32:

* "NIGHT\_MODE\_ON": kích hoạt chế độ ban đêm.
* "NIGHT\_MODE\_OFF": tắt chế độ ban đêm, trở về hoạt động bình thường.

Ngoài ra, trên ESP32 còn có chức năng tự động chuyển sang chế độ Ban đêm từ 22h đến 23h, giúp hệ thống vận hành linh hoạt theo thời gian thực.

private void btnNightMode\_Click(object sender, EventArgs e)

{

if (serialPort != null && serialPort.IsOpen)

{

if (btnNightMode.Text == "Ban Đêm")

{

serialPort.WriteLine("NIGHT\_MODE\_ON");

btnNightMode.Text = "Tắt Ban Đêm";

txtStatus.AppendText("Gửi: NIGHT\_MODE\_ON\n");

}

else

{

serialPort.WriteLine("NIGHT\_MODE\_OFF");

btnNightMode.Text = "Ban Đêm";

txtStatus.AppendText("Gửi: NIGHT\_MODE\_OFF\n");

}

}

else

{

MessageBox.Show("Chưa kết nối với ESP32!");

}

}

#### 2.4.2.6.Điều khiển chế độ cảnh báo

Chế độ Cảnh báo được sử dụng vào ban đêm muộn hoặc khi có sự cố giao thông, cho phép tất cả các đèn tín hiệu chuyển sang nhấp nháy màu vàng ở cả hai hướng nhằm cảnh báo người đi đường.

Người dùng có thể bật/tắt chế độ cảnh báo bằng nút "Cảnh báo" trên phần mềm. Khi nhấn:

* Gửi lệnh "WARNING\_MODE\_ON" đến ESP32 để bắt đầu chế độ cảnh báo.
* Gửi lệnh "WARNING\_MODE\_OFF" để kết thúc chế độ cảnh báo và quay về hoạt động bình thường.

Hệ thống hiển thị trạng thái gửi và phản hồi trên txtStatus, giúp người dùng kiểm tra giao tiếp dễ dàng.

private void btnWarningMode\_Click(object sender, EventArgs e)

{

if (serialPort != null && serialPort.IsOpen)

{

if (btnWarningMode.Text == "Cảnh báo")

{

serialPort.WriteLine("WARNING\_MODE\_ON");

btnWarningMode.Text = "Tắt Cảnh báo";

txtStatus.AppendText("Gửi: WARNING\_MODE\_ON\n");

}

else

{

serialPort.WriteLine("WARNING\_MODE\_OFF");

btnWarningMode.Text = "Cảnh báo";

txtStatus.AppendText("Gửi: WARNING\_MODE\_OFF\n");

}

}

else

{

MessageBox.Show("Chưa kết nối với ESP32!");

}

}

#### 2.4.2.7.Điều khiển chế độ ưu tiên

Chế độ Ưu tiên cho phép người dùng điều khiển hệ thống đèn tín hiệu ưu tiên một hướng cụ thể (Dọc hoặc Ngang) trong những tình huống đặc biệt như có xe cứu thương, cứu hỏa, hoặc mật độ giao thông không đều.

Trên giao diện WinForms, người dùng có thể chọn:

* "Ưu tiên Dọc" (PRIORITY\_MODE\_ON:D)
* "Ưu tiên Ngang" (PRIORITY\_MODE\_ON:N)

Nếu nhấn lại nút đã chọn, hệ thống sẽ hủy bỏ chế độ ưu tiên bằng lệnh PRIORITY\_MODE\_OFF.

Chế độ này được điều khiển thông qua SerialPort gửi lệnh đến ESP32, và trạng thái hiện tại được lưu bằng biến isPriorityOn và currentPriorityDir.

Xử lý ưu tiên nút dọc:

private void btnPriorityD\_Click(object sender, EventArgs e)

{

if (serialPort != null && serialPort.IsOpen)

{

if (!isPriorityOn || currentPriorityDir != "D")

{

serialPort.WriteLine("PRIORITY\_MODE\_ON:D");

txtStatus.AppendText("Gửi: PRIORITY\_MODE\_ON:D\n");

isPriorityOn = true;

currentPriorityDir = "D";

}

else

{

serialPort.WriteLine("PRIORITY\_MODE\_OFF");

txtStatus.AppendText("Gửi: PRIORITY\_MODE\_OFF\n");

isPriorityOn = false;

currentPriorityDir = "";

}

}

else

{

MessageBox.Show("Chưa kết nối với ESP32!");

}

}

Xử lý ưu tiên nút ngang:

private void btnPriorityN\_Click(object sender, EventArgs e)

{

if (serialPort != null && serialPort.IsOpen)

{

if (!isPriorityOn || currentPriorityDir != "N")

{

serialPort.WriteLine("PRIORITY\_MODE\_ON:N");

txtStatus.AppendText("Gửi: PRIORITY\_MODE\_ON:N\n");

isPriorityOn = true;

currentPriorityDir = "N";

}

else

{

serialPort.WriteLine("PRIORITY\_MODE\_OFF");

txtStatus.AppendText("Gửi: PRIORITY\_MODE\_OFF\n");

isPriorityOn = false;

currentPriorityDir = "";

}

}

else

{

MessageBox.Show("Chưa kết nối với ESP32!");

}

}

#### 2.4.2.8.Đóng kết nối và xử lý khi thoát chương trình

Để đảm bảo an toàn khi đóng ứng dụng và tránh treo cổng COM, chương trình đã xử lý sự kiện FormClosing, đảm bảo rằng kết nối SerialPort sẽ được đóng đúng cách trước khi thoát khỏi chương trình.

Việc đóng kết nối đúng lúc sẽ tránh tình trạng khóa cứng cổng COM, gây lỗi khi kết nối lại sau này.

protected override void OnFormClosing(FormClosingEventArgs e)

{

if (serialPort != null && serialPort.IsOpen)

serialPort.Close();

base.OnFormClosing(e);

}

Ngoài ra, người dùng cũng có thể chủ động ngắt kết nối thủ công bằng cách nhấn lại nút button “Disconnect” để đóng cổng COM.

serialPort.Close();

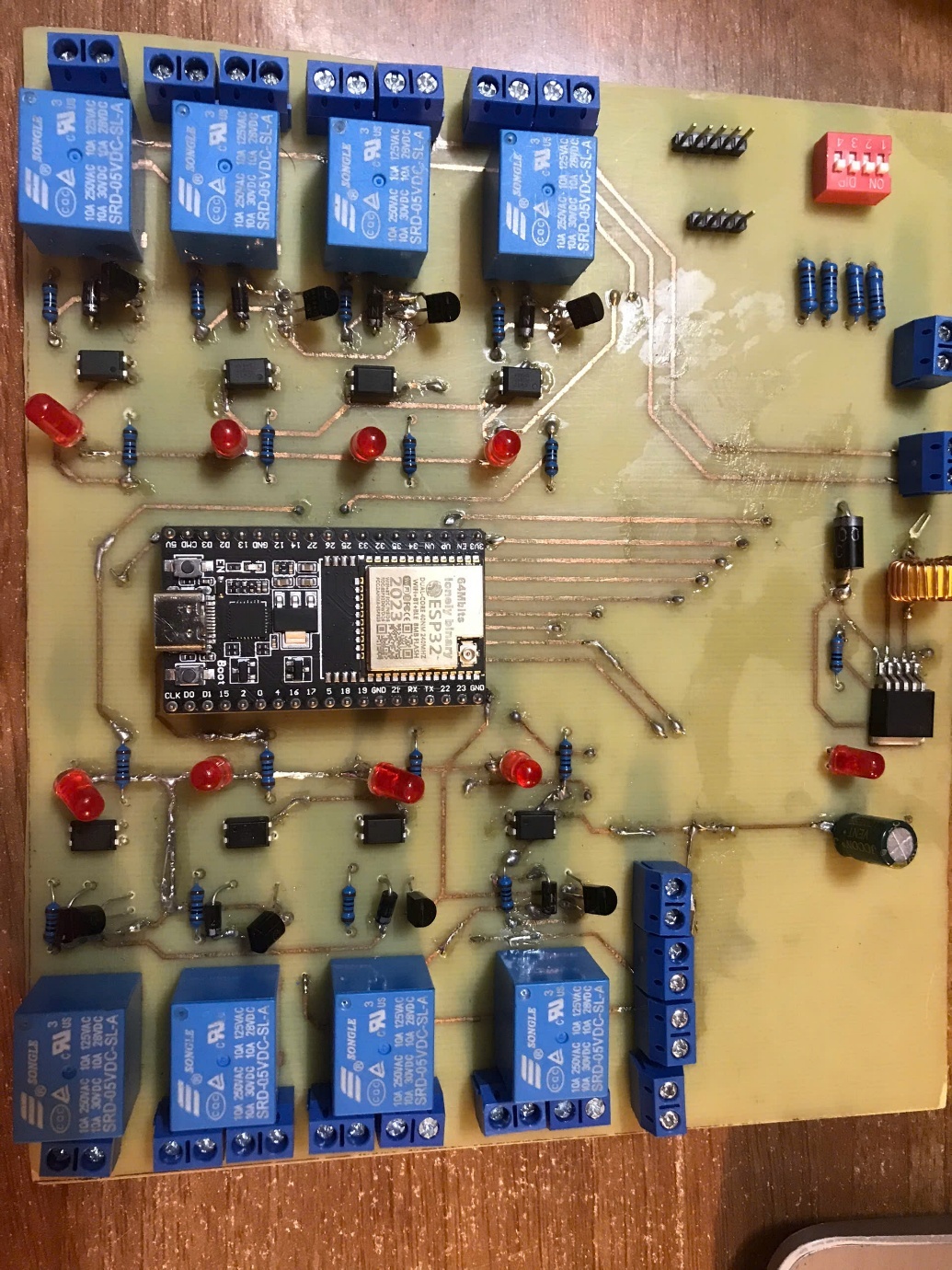
btnConnect.Text = "Connect";

txtStatus.AppendText("Đã ngắt kết nối!\r\n");

# CHƯƠNG 3: KẾT QUẢ THỰC HIỆN VÀ KẾT LUẬN

## 3.1.Quá trình thử nghiệm và tối ưu hóa mạch điều khiển

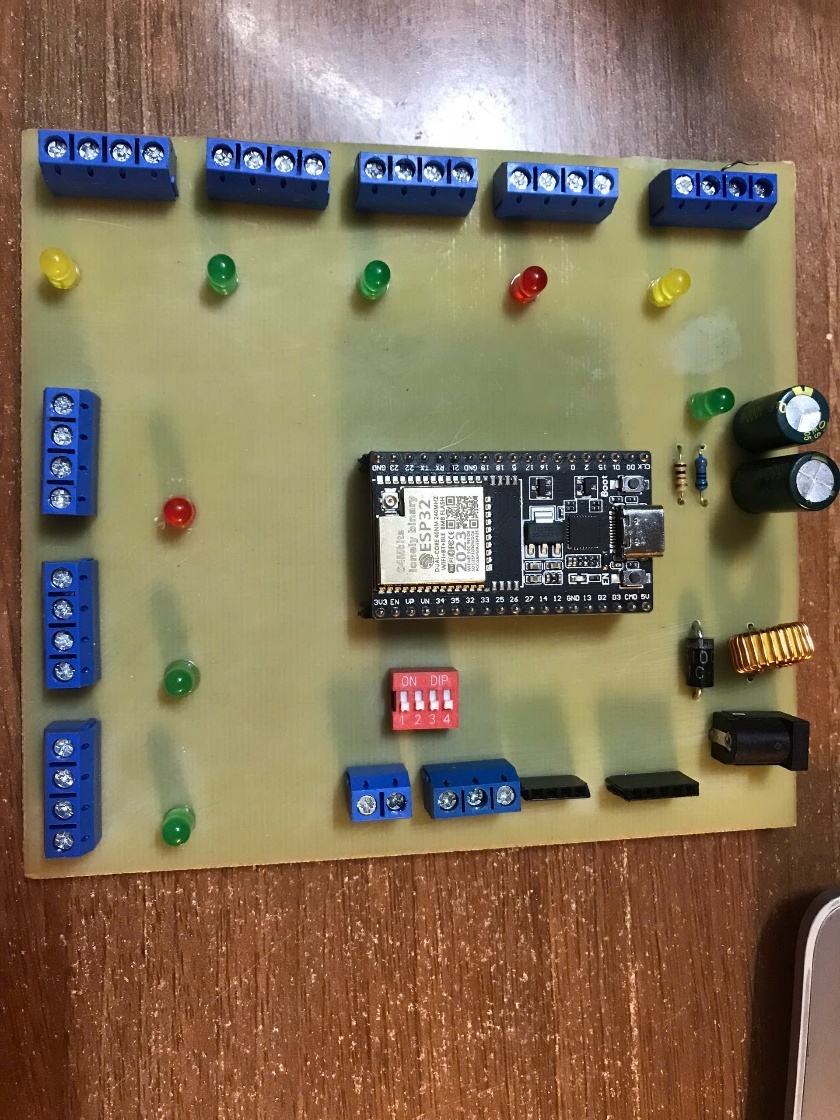
Trong quá trình thử nghiệm, với lần thử nghiệm đầu tiên, do chưa có thiết bị đo nên mạch bị bỏ sót nhiều vấn đề, các chân tín hiệu, nguồn điện không được đảm bảo đúng so với mục tiêu thiết kế ban đầu



Hình 3.1: Mạch thử nghiệm lần thứ nhất

Mạch lần thứ nhất này nhóm dùng các relay 5V để đóng cắt cơ bản trong việc bật tắt các đèn LED trong hệ thống đèn giao thông. Khi tham khảo mạch giảm áp thì nhóm đã có thiếu xót khi tìm hiểu về bộ giảm áp, bộ giảm áp khi không có tải sẽ có điện áp lớn hơn 5V, gây cháy các đèn LED 7 thanh. Vì vậy nhóm đã bổ sung điện trở 1k Ohm để đảm bảo rằng bộ giảm áp luôn có đầu ra ổn định là 5V.

Khi thực hiện lần thử nghiệm thứ 2, nhóm đã thay relay 5v bằng Optocoupler PC817, do nhận thấy PC817 có tốc độ đóng cắt nhanh, tuổi thọ cao và không có tiếng ồn như khi sử dụng relay, đặc biệt nó phù hợp với điều khiển logic của đề tài nhóm đang nghiên cứu.



Hình 3.2: Mặt trước của mạch thử nghiệm lần thứ hai

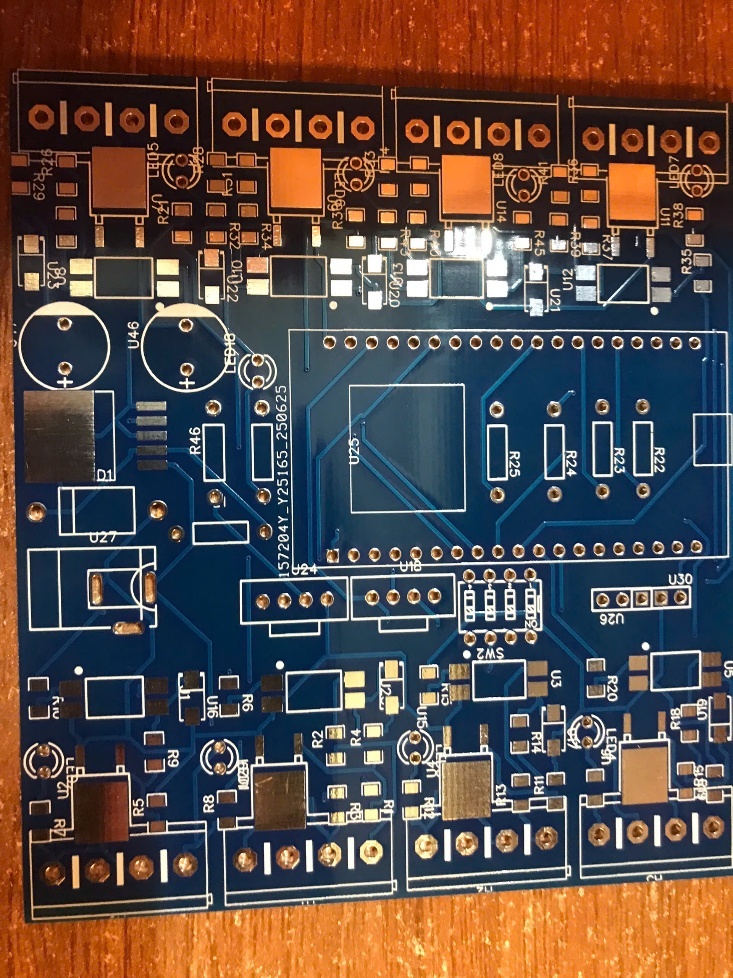
Ngoài ra nhóm cũng sử dụng linh kiện dán thay vì sử dụng cắm để tiết giảm diện tích



Hình 3.3: Mặt sau của mạch thử nghiệm lần thứ hai

Đến với lần thử nghiệm thứ hai này, nhóm đã có được thiết bị đo nên đã đảm bảo ổn định nguồn vào, các chân tín hiệu đúng so với mục tiêu thiết kế ban đầu.

Sau 2 làn thử nghiệm, nhận thấy mục tiêu nhóm đề ra đã đạt được,các chức năng cần thiết đều đã ổn định, vì vậy nhóm quyết định tối ưu phần thiết kế mạch và đưa ra mạch thiết kế cuối cùng sau đó gửi cho các đối tác gia công. Việc này giúp cho sản phẩm đầu ra của nhóm đạt được mục tiêu về mặt thẩm mỹ, giá thành tối ưu hơn, an toàn hơn khi chạy thực tế.



Hình 3.4: Mạch thiết kết được đặt gia công từ các đối tác

## 3.2.Kết quả thực nghiệm

### 3.2.1.Bộ điều khiển đã hoàn thiện

Mạch đã hoàn thiện việc hàn các linh kiện lên trên bảng mạch, kết nối với các module như LCD, LED 7 thanh, module thời gian thực RTC DS3231, và cắm module chính ESP32 vào mạch.

Đã nạp chương trình vào ESP32, đồng thời kết nối với phần mềm WinForms

### 3.2.2.Quá trình thử nghiệm

Hệ thống đèn tín hiệu giao thông được thiết kế hỗ trợ hai chế độ vận hành gồm chế độ tự động và chế độ thủ công.

Khi bật chế độ tự động, hệ thống sẽ dựa vào thời gian thực từ module DS3231 để tự động chuyển đổi giữa các chế độ:

* Ban ngày (06h – 21h59): hoạt động ở chế độ Bình thường
* Ban đêm (22h – 23h59): chuyển sang chế độ Ban đêm
* Đêm khuya (00h – 05h59): chuyển sang chế độ Cảnh báo

A digital display with wires

AI-generated content may be incorrect.

Hình 3.5: Bộ điều khiển ở chế độ Auto, thời gian đang là đêm khuya

Khi tắt chế độ tự động, người dùng có thể chủ động điều khiển hệ thống theo nhu cầu bằng cách:

* Nhấn nút chọn chế độ (Ban đêm / Cảnh báo / Ưu tiên) trên phần mềm WinForms

A screenshot of a computer

AI-generated content may be incorrect.

Hình 3.6: Giao diện Winform, hệ thống đang ở chế độ thủ công, thử nghiệm với trường hợp cảnh báo

* Gạt các DIP switch tương ứng trực tiếp trên mạch điều khiển

### 3.2.3.Kết quả đạt được

- Các chế độ chuyển đổi mượt mà, đúng thời gian, không xung đột.

- Thời gian đếm ngược hiển thị chính xác trên LED 7 đoạn.

- LCD hiển thị được đồng hồ và trạng thái chế độ.

- Phần mềm WinForms giao tiếp ổn định, gửi/nhận dữ liệu chính xác.

- Dữ liệu thời gian đèn lưu được trong bộ nhớ ESP32 (Preferences).

## 3.3.Kết luận và hướng phát triển của đề tài

### 3.3.1.Kết luận

Sau quá trình thiết kế, chế tạo và thử nghiệm, nhóm đã hoàn thiện thành công bộ điều khiển hệ thống đèn tín hiệu giao thông cho ngã tư với các chức năng chính:

* Điều khiển 8 LED tín hiệu gồm: đèn đỏ, vàng, xanh, rẽ trái cho cả hai hướng dọc và ngang.
* Hiển thị thời gian đếm ngược rõ ràng bằng LED 7 đoạn.
* Tích hợp LCD I2C để hiển thị thời gian thực và trạng thái chế độ.
* Giao tiếp ổn định với phần mềm C# WinForms, cho phép người dùng cấu hình thời gian cho các pha đèn, đồng thời có thể bật/tắt các chế độ Ban đêm, Cảnh báo, Ưu tiên.
* Hệ thống hỗ trợ chế độ Tự động và Thủ công, đảm bảo linh hoạt trong điều khiển và ứng dụng thực tế.
* Tối ưu quy trình từ tự làm mạch tay đến đặt gia công mạch, cho thấy sự tiến bộ và chuyên nghiệp trong triển khai kỹ thuật.
* Hệ thống hoạt động ổn định, đúng chức năng đã đề ra, thể hiện khả năng ứng dụng vào thực tế và phù hợp với yêu cầu của đề tài tốt nghiệp.

### 3.3.2.Hướng phát triển

Để nâng cao tính hoàn thiện và khả năng mở rộng, nhóm đề xuất một số hướng phát triển tiếp theo:

* Tích hợp AI để tính toán số lượng phương tiện, từ đó xác định khả năng tắc đường của mỗi hướng, điều chỉnh đèn giao thông tự động giúp giảm ùn tắc.
* Giao tiếp không dây (WiFi/MQTT) để điều khiển và giám sát từ xa qua smartphone hoặc máy chủ.
* Thiết kế giao diện phần mềm thân thiện hơn.
* Phát triển thêm tính năng giám sát lỗi, đảm bảo đèn luôn hoạt động ổn định
* Tích hợp camera để kết hợp với hệ thống giám sát giao thông thông minh trong tương lai.

# TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | "Bộ điều khiển tín hiệu giao thông," Công ty sản xuất và thi công đèn tín hiệu giao thông , [Online]. Available: https://dengiaothong.org/bo-dieu-khien-tin-hieu-giao-thong/. |
| [2] | Signawork, "TL12 Industrial Traffic Light PDF," [Online]. Available: https://www.signaworks.com/site/pdf/products/Traffic%20Lights/\_Signaworks\_PDF\_TL12\_23.pdf . |
| [3] | La Semaforica , "Countdown Traffic Light," [Online]. Available: https://sibel.bg/upl\_doc/COUNTDOWN\_eng.pdf . |
| [4] | L. Espressif Systems (Shanghai) Co., "esp32-wroom-32d\_esp32-wroom-32u\_datasheet\_en.pdf," 2025. [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32d\_esp32-wroom-32u\_datasheet\_en.pdf . |
| [5] | DALLAS [Dallas Semiconductor], "DS3231 Datasheet (PDF)," 2005. [Online]. Available: https://www.alldatasheet.vn/datasheet-pdf/pdf/112132/DALLAS/DS3231.html. |
| [6] | ONSEMI [ON Semiconductor], "74HC595 Datasheet (PDF)," [Online]. Available: https://www.alldatasheet.vn/datasheet-pdf/view/12198/ONSEMI/74HC595.html. |
| [7] | OptoSupply, "0.56 Inch Single Digit 7 Segment LED Display With 74HC595 Shift Register," [Online]. Available: https://akizukidenshi.com/goodsaffix/OSL10564-74HC595%20-X.PDF. |
| [8] | Handson Technology, "I2C Serial Interface 1602 LCD Module," [Online]. Available: https://www.handsontec.com/dataspecs/module/I2C\_1602\_LCD.pdf. |
| [9] | "Tìm hiểu về công tắc DIP switch," 09 02 2023. [Online]. Available: https://www.thegioiic.com/tin-tuc/tim-hieu-ve-cong-tac-dip-switch. |
| [10] | VBSEMI [VBsemi Electronics Co.,Ltd], "LR7843 Datasheet (PDF)," 2011. [Online]. Available: https://www.alldatasheet.vn/datasheet-pdf/view/1265884/VBSEMI/LR7843.html. |
| [11] | S. [. Corporation], "PC817 Datasheet (PDF)," [Online]. Available: https://www.alldatasheet.com/datasheet-pdf/view/43371/SHARP/PC817.html. |
| [12] | Wajid Hussain, "LM2596 Adjustable 3A Step-Down Voltage Regulator – Datasheet," 13 01 2022. [Online]. Available: https://www.circuits-diy.com/lm2596-adjustable-3a-step-down-voltage-regulator-datasheet/. |
| [13] | "Phần mềm Arduino IDE là gì? Chi tiết nhất," [Online]. Available: https://vn.got-it.ai/blog/phan-mem-arduino-ide-la-gi-chi-tiet-nhat. |