BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT
THÀNH PHỐ HỒ CHÍ MINH

ĐỒ ÁN TỐT NGHIỆP
NGÀNH CÔNG NGHỆ KỸ THUẬT ĐIỆN TỬ TRUYỀN THÔNG

THIẾT KẾ VÀ THI CÔNG HỆ THỐNG CHIẾU SÁNG ĐIỀU KHIỂN QUA MẠNG ZIGBEE

GVHD: PHAN VÂN HOÀN
SVTT: TRẦN BÌNH TRỌNG
MSSV: 13141387
SVTH: NGUYỄN ĐÌNH KHƯƠNG
MSSV: 15141189

Tp. Hồ Chí Minh, tháng 07/2020

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
KHOA ĐIỆN – ĐIỆN TỬ
BỘ MÔN ĐIỆN TỬ CÔNG NGHIỆP – Y SINH

ĐỒ ÁN TỐT NGHIỆP

NGÀNH CÔNG NGHỆ KỸ THUẬT ĐIỆN TỬ TRUYỀN THÔNG

ĐỀ TÀI:

THIẾT KẾ VÀ THI CÔNG HỆ THỐNG CHIẾU SÁNG ĐIỀU KHIỂN QUA MẠNG ZIGBEE

GVHD: ThS. Phan Vân Hoàn
SVTH: Trần Bình Trọng MSSV: 13141387
Nguyễn Đình khương MSSV: 15141189

Tp. Hồ Chí Minh 07/2020

TRƯỜNG ĐH SPKT TP. HỒ CHÍ MINH
KHOA ĐIỆN-ĐIỆN TỬ
BỘ MÔN ĐIỆN TỬ CÔNG NGHIỆP – Y SINH

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
ĐỘC LẬP - TỰ DO - HẠNH PHÚC
----o0o----

Tp. HCM, ngày 03 tháng 08 năm 2020

NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

Họ tên sinh viên:	Trần Bình Trọng	MSSV:	13141387
	Nguyễn Đình Khương	MSSV:	15141189
Chuyên ngành:	Điện tử công nghiệp	Mã ngành:	41
Hệ đào tạo:	Đại học chính quy	Mã hệ:	1
Khóa:	2013	Lớp:	13141DT1A
	2015	Lớp:	15141DT1B

I. TÊN ĐỀ TÀI: **THIẾT KẾ VÀ THI CÔNG HỆ THỐNG CHIẾU SÁNG ĐIỀU KHIỂN QUA MẠNG ZIGBEE**

II. NHIỆM VỤ

1. Các số liệu ban đầu:
   - Vi điều khiển Arduino Uno R3, Arduino Mega 2560 và ngôn ngữ lập trình
   - Tài liệu về Arduino, Xbee S2.
   - Thư viện Arduino, Xbee, DS1307, Nextion.
2. Nội dung thực hiện:
   - Nội dung 1: Nghiên cứu thiết kế giao diện để người dùng giao tiếp với hệ thống thông qua màn hình Nextion HMI 3.2Inch.
   - Nội dung 2: Tìm hiểu và nghiên cứu về các Mô-đun Arduino, Mô-đun Realtime và Mô-đun cảm biến ánh sáng.
   - Nội dung 3: Nghiên cứu cách kết nối, các chế độ hoạt động của Mô-đun Xbee S2 để xây dựng một hệ thống mạng không giây Zigbee.
   - Nội dung 4: Nghiên cứu và tìm hiểu về cách thức điều khiển độ sáng của đèn.
   - Nội dung 5: Lập trình điều khiển độ sáng và phản hồi trạng thái của đèn
   - Nội dung 6: Thi công phần cứng.
   - Nội dung 7: Thiết kế mô hình hệ thống.
   - Nội dung 8: Đánh giá kết quả thực hiện

III. NGÀY GIAO NHIỆM VỤ: 23/03/2020
IV. NGÀY HOÀN THÀNH NHIỆM VỤ: 03/08/2020
V. HỌ VÀ TÊN CÁN BỘ HƯỚNG DẪN: ThS. Phan Vân Hoàn

CÁN BỘ HƯỚNG DẪN BM. ĐIỆN TỬ CÔNG NGHIỆP – Y SINH

TRƯỜNG ĐH SPKT TP. HỒ CHÍ MINH
KHOA ĐIỆN-ĐIỆN TỬ
BỘ MÔN ĐIỆN TỬ CÔNG NGHIỆP – Y SINH

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
ĐỘC LẬP - TỰ DO - HẠNH PHÚC
----o0o----

Tp. HCM, ngày 03 tháng 08 năm 2020

LỊCH TRÌNH THỰC HIỆN ĐỒ ÁN TỐT NGHIỆP

Họ tên sinh viên 1: TRẦN BÌNH TRỌNG
Lớp: 13141DT1A MSSV: 13141387
Họ tên sinh viên 2: NGUYỄN ĐÌNH KHƯƠNG
Lớp: 15141DT1B MSSV: 15141189
Tên đề tài: **Thiết kế và thi công hệ thống chiếu sáng điều khiển qua mạng Zigbee.**

Tuần/ngày	***Nội dung***	***Xác nhận GVHD***
Tuần 1 (23/03 - 29/03)	- Gặp GVHD để nghe phổ biến yêu cầu làm đồ án, tiến hành chọn đề tài.	
Tuần 2 (30/03 - 05/04)	- GVHD tiến hành xét duyệt đề tài - Viết đề cương chi tiết.	
Tuần 3 (06/04 - 12/04)	- Gặp và báo cáo GVHD hướng thực hiện đề tài.	
Tuần 4 (13/04 - 19/04)	- Tìm kiếm một số tài liệu liên quan - Trao đổi với giáo viên về công nghệ Zigbee.	
Tuần 5 (20/04 - 26/04/)	- Tìm hiểu về cách cấu hình và thiết lập mô hình mạng Zigbee sử dụng module Xbee S2.	
Tuần 6 (27/04-03/05)	- Tìm hiểu mạch Arduino và cách lập trình	
Tuần 7 (04/05 - 10/05)	-Tìm hiểu về module cảm biến quang trở CDS. -Tìm hiểu về module thời gian thực DS1307.	
Tuần 8 (11/05 - 17/05)	- Tìm hiểu về mạch công suất điều khiển độ sáng đèn.	
Tuần 9 (18/05 - 24/05)	- Tìm hiểu về thiết kế giao diện hiển thị trên màn hình Nextion HMI.	

Tuần 10 (25/05 - 31/05)	- Thiết kế sơ đồ mạch và vẽ PCB. - Vẽ lưu đồ giải thuật và viết chương trình.	
Tuần 11 (01/06 - 07/06)	- Thi công phần cứng. - Thi công mô hình.	
Tuần 12 (08/06 - 14/06)	- Báo cáo tiến độ GVHD. - Chỉnh sửa phần cứng, phần mềm và hoàn thiện mô hình.	
Tuần 13 (15/06 - 21/06)	- Chạy, kiểm tra và hiệu chỉnh toàn bộ mô hình.	
Tuần 14 (22/06 - 28/06)	- Viết báo cáo.	
Tuần 15 (29/06 - 05/07)	- Chỉnh sửa và hoàn thiện báo cáo.	
Tuần 16 (06/07 - 12/07)	- Thiết kế slide thuyết trình.	
Tuần 17, 18 (13/07 -23/07)	- Làm slide báo cáo và nộp quyển báo cáo	

GV HƯỚNG DẪN
(Ký và ghi rõ họ và tên)

LỜI CAM ĐOAN

Đề tài này là do chúng tôi tự thực hiện dựa vào một số tài liệu trước đó và không sao chép từ tài liệu hay công trình đã có trước đó.

Người thực hiện đề tài

Trần Bình Trọng – Nguyễn Đình Khương

LỜI CẢM ƠN

Lời đầu tiên, chúng em xin trân trọng gửi lời cảm ơn đến thầy **ThS. Phan Vân Hoàn** người đã nhiệt tình hỗ trợ, giải đáp các khúc mắc, truyền đạt kiến thức, theo dõi sát sao quá trình làm đồ án, từ đó thầy đóng góp ý kiến định hướng cho đề tài cũng như động viên chúng em trong quá trình làm đề tài này.

Tiếp đến, chúng em cũng xin cảm ơn đến các thầy cô khoa Điện- Điện tử đã tạo mọi điều kiện cho chúng em thực hiện đề tài đặc biệt là thầy **PGS.TS Nguyễn Thanh Hải** và thầy **ThS.Võ Đức Dũng**. Chính những kiến thức, kỹ năng quan trọng mà các thầy cô đã truyền tải mà chúng em có thể vận dụng để giải quyết các vấn đề gặp phải khi thực hiện đề tài.

Mặc dù đã cố gắng nỗ lực hết sức của mình, song chắc chắn đồ án không tránh khỏi những thiếu sót. Chúng em kính mong nhận được sự thông cảm và chỉ bảo tận tình của thầy cô.

Người thực hiện

Trần Bình Trọng – Nguyễn Đình Khương

MỤC LỤC

DANH MỤC HÌNH ẢNH

DANH SÁCH CÁC BẢNG

CHƯƠNG 1: TỔNG QUAN

1.1 Đặt vấn đề

Ngày nay, khoa học công nghệ phát triển mạnh mẽ và đã tạo ra những sản phẩm thông minh, tiện lợi, đa dạng với nhiều chức năng khác nhau để phục vụ nhu cầu ngày càng cao của con người. Bên cạnh đó, việc tiết kiệm năng lượng luôn được đặc biệt quan tâm trong mọi lĩnh vực.

Trong nông nghiệp, ngoài những yếu tố như phân bón, nước, độ ẩm, nhiệt độ thì ánh sáng cũng là một phần thiết yếu để cây trồng phát triển tốt vì thế việc chiếu sáng cho các loài hoa, cây trồng ra kịp thời vụ, dịp lễ đang được đặc biệt quan tâm.

Tại nhiều trang trại trồng thanh long, hoa cúc, nuôi cấy mô… ở một số địa phương trên cả nước, từ nhiều năm nay, bóng đèn sợi đốt vẫn được sử dụng để chiếu sáng cho cây trồng nhằm kích thích cây tăng trưởng. Bóng đèn sợi đốt được ví như là "mặt trời nhân tạo" dành riêng cho nông nghiệp, không những cho ánh sáng khá đồng đều mà còn phát ra lượng nhiệt năng sưởi ấm cho cây trồng giúp kích thích rễ phát tiển và thực hiện phản ứng quang học, từ đó kích thích tăng trưởng. Cho nên, dù cây được trồng trong phòng nuôi cấy mô, không tiếp xúc với ánh sáng mặt trời hoặc vào ban đêm, cây vẫn thực hiện tốt quá trình quang hợp. Ngoài ra, đèn sợi đốt có công dụng xua đuổi sâu bọ và những con vật đêm gây hại cho vườn tược. Bên cạnh những ưu điểm trên thì đèn sợi đốt lại tiêu hao nhiều điện năng trong khi sử dụng.

Đặc biệt đối với trang trại trồng hoa cúc ở những nơi có khí hậu lạnh, người trồng phải dùng đèn sợi đốt để canh thời gian sáng và sưởi ấm cho cây để cây ra hoa đúng dịp Tết và mang lại giá trị kinh tế cao. Vì vậy tùy vào các mốc thời gian trong năm mà chúng ta phải điều chỉnh ánh sáng hợp lý để cây trồng phát triển đạt năng suất cao và mang lại hiệu quả kinh tế cao.

Nhận thấy tầm quan trọng của những vấn đề trên nhóm đã nghĩ ra ý tưởng thiết kế hệ thống chiếu sáng cho cây trồng nhằm tăng năng suất và tiết kiệm năng lượng góp phần nâng cao thu nhập cho người nông dân.

Trong phần đồ án, nhóm báo cáo sẽ đi sâu vào cách điều khiển độ sáng đèn và sử dụng ánh sáng nhân tạo để thắm sáng, cũng như sử dụng công nghệ không dây (Zigbee) để giao tiếp, điều khiển mạng lưới ánh sáng.

1.2 Mục tiêu

Thiết kế và thi công hệ thống có khả năng điều khiển và giám sát hệ thống chiếu sáng gồm có ba đèn chiếu sáng. Hệ thống gồm có một mạch trung tâm dùng để truyền dữ liệu và ba mạch phụ nhận dữ liệu để điều khiển hệ thống chiếu sáng. Sử dụng chuẩn truyền ZigBee để truyền nhận tín hiệu kết nối với các thiết bị thành một mạng để dễ điều khiển giám sát.

Hệ thống cho phép người sử dụng có thể bật tắt, điều chỉnh độ sáng đèn ở hai chế độ là bằng tay và tự động cũng như giám sát qua màn hình cảm ứng HMI.

1.3 Giới hạn

- Hệ thống chỉ có hai chế độ điều khiển: hẹn giờ tự động và hẹn giờ bằng nút cảm ứng cũng như hiển thị trạng thái hệ thống chỉ thông qua màn hình HMI.
- Khoảng cách truyền giữa mạch phụ và mạch trung tâm là khoảng 40m trong nhà và 120m ngoài trời.
- Hệ thống chỉ điều khiển được 3 đèn chiếu sáng và chỉ sử dụng cảm biến quang để phát hiện tình trạng hư hỏng của hệ thống chiếu sáng.
- Giao diện đơn giản chỉ có nhập mật khẩu và nút cảm ứng để điều khiển hệ thống chiếu sáng
- Sử dụng bộ sạc của các thiết bị di động để cấp nguồn cho các board mạch.

1.4 Nội dung thực hiện

- **Chương 1: Tổng quan**

- Giới thiệu lý do chọn đề tài, mục tiêu, giới hạn, các nội dung nghiên cứu.

- **Chương 2: Cơ sở lý thuyết**

- Trình bày, giới thiệu tổng quan về công nghệ Zigbee.
- Giới thiệu tổng quan về Arduino.
- Giới thiệu các chuẩn giao tiếp được sử dụng trong đề tài.

- **Chương 3: Thiết kế hệ thống điều khiển**

- Thiết kế sơ đồ khối cho hệ thống.
- Thiết kế mô hình mạng Zigbee
- Phân tích lựa chọn linh kiện sử dụng trong mạch.
- Thiết kế các board mạch.
- Lập lưu đồ giải thuật cho các chế độ điều khiển.

- **Chương 4: Thi công hệ thống**

- Cấu hình chế độ hoạt động cho các mô-đun Xbee S2 bằng phần mền XCTU.
- Lập trình cho hệ thống.
- Thi công các mạch đã được thiết kế và ghép các mô-đun với nhau.

- **Chương 5: Kết quả và đánh giá.**

- Trình bày kết quả đạt được.
- Đánh giá sự ổn định của hệ thống.

- **Chương 6: Kết luận và hướng phát triển**

- Hướng phát triển đề tài.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1 Tổng quan về mạng Zigbee

2.1.1 Tổng quan

Zigbee là một giao thức được xây dựng theo chuẩn IEEE 802.15.4 (một tiêu chuẩn kỹ thuật xác định hoạt động của mạng cá nhân). Giao thức này được tạo ra nhằm đáp ứng yêu cầu cho một mạng kết nối các thiết bị với giá thành thấp, công suất thấp và các ứng dụng yêu cầu phải có khả năng linh động trong phạm vi rộng.

Zigbee là một công nghệ không dây cung cấp một giải pháp truyền dữ liệu không dây với yêu cầu tốc độ truyền dữ liệu thấp, hiệu quả năng lượng và kết nối mạng an toàn. Nhờ vào kiểu thiết kế truyền thông đặc thù theo hình Zigzag tương tự như tổ ong mà ZigBee có khả năng cho phép nhiều nhóm thiết bị truyền thông tin với nhau trong cùng một khoảng thời gian. Giao thức Zigbee được xây dựng và phê chuẩn bởi các thành viên của Zigbee Alliance (hơn 300 nhà sản xuất bán dẫn hàng đầu, các công ty công nghệ, công ty dịch vụ).

2.1.2 Cấu trúc mạng Zigbee

Giao thức mạng Zigbee bao gồm 5 lớp mạng:

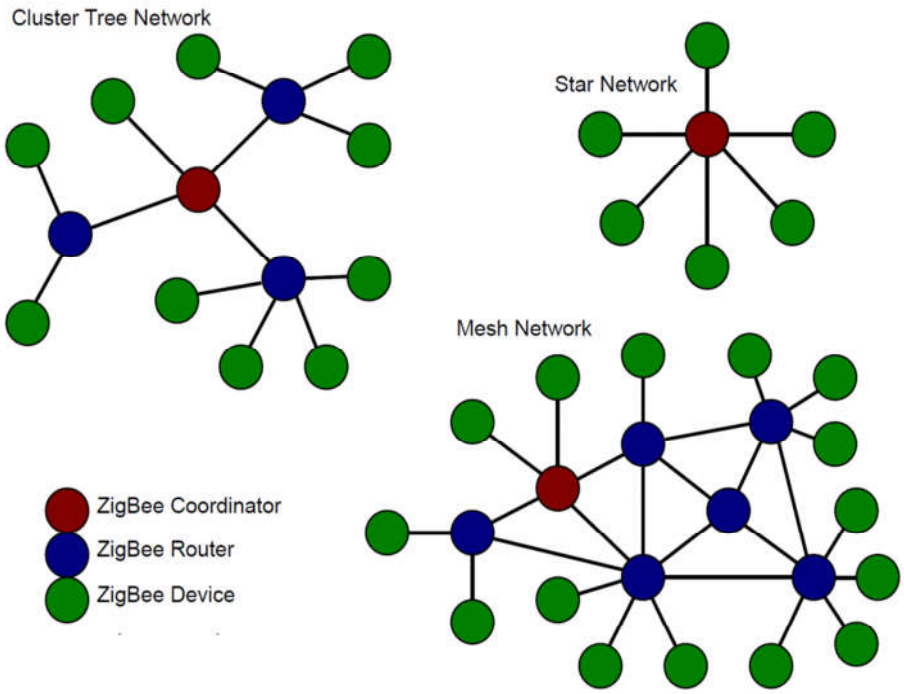
- **Lớp vật lý:** Đây là lớp giao thức thấp nhất và chịu trách nhiệm kiểm soát và kích hoạt bộ thu phát vô tuyến và cũng để chọn tần số kênh và giám sát kênh. Nó cũng chịu trách nhiệm liên lạc với các thiết bị radio. Truyền thông dữ liệu hoặc lệnh được thực hiện bằng cách sử dụng gói. Mỗi gói PHY bao gồm tiêu đề đồng bộ hóa (SHR) (chịu trách nhiệm đồng bộ hóa máy thu), tiêu đề vật lý (PHR) (chứa thông tin về độ dài khung) và tải trọng PHY (được cung cấp bởi các lớp trên dưới dạng khung và bao gồm dữ liệu hoặc lệnh).
- **Kiểm soát truy cập trung bình hoặc lớp MAC:** Nó hoạt động như một giao diện giữa lớp vật lý và các lớp mạng. Nó chịu trách nhiệm đồng bộ hóa các thiết bị trong mạng. Khung MAC (được điều phối viên sử dụng để truyền Beacons), khung dữ liệu, khung xác nhận hoặc khung lệnh. Nó bao gồm MAC Header (chứa thông tin về bảo mật và địa chỉ), MAC Payload có kích thước chiều dài thay đổi (chứa dữ liệu hoặc lệnh) và Footer MAC (chứa chuỗi kiểm tra khung 16 bit để xác minh dữ liệu)

- **Lớp mạng:** Lớp này kết nối lớp ứng dụng với lớp MAC. Nó quản lý sự hình thành và định tuyến mạng. Nó thiết lập một mạng mới và chọn cấu trúc liên kết mạng. Khung NWK bao gồm tiêu đề NWK và tải trọng NWK. Tiêu đề chứa thông tin liên quan đến việc kiểm soát địa chỉ cấp mạng. Tải trọng NWK chứa khung lớp con ứng dụng.
- **Lớp phụ hỗ trợ ứng dụng:** Nó cung cấp một tập hợp các dịch vụ thông qua hai thực thể - Thực thể ứng dụng Support Data và thực thể quản lý hỗ trợ ứng dụng, cho các lớp ứng dụng và mạng. Các thực thể này được truy cập thông qua điểm truy cập dịch vụ tương ứng (SAP)
- **Lớp ứng dụng:** Đây là lớp cao nhất trong mạng và chịu trách nhiệm lưu trữ các đối tượng ứng dụng chứa các ứng dụng người dùng và đối tượng thiết bị ZigBee (ZDOs). Một thiết bị ZigBee có thể chứa tới 240 đối tượng ứng dụng điều khiển và quản lý các lớp giao thức. Mỗi đối tượng ứng dụng có thể bao gồm một hồ sơ hoặc chương trình ứng dụng, được phát triển bởi người dùng hoặc liên minh ZigBee. Hồ sơ ứng dụng chịu trách nhiệm truyền và nhận dữ liệu trong mạng. Loại thiết bị và chức năng của từng thiết bị được xác định trong hồ sơ ứng dụng. Các đối tượng thiết bị ZigBee hoạt động như một giao diện giữa các đối tượng ứng dụng, cấu hình thiết bị và lớp phụ ứng dụng.

2.1.2 Mô hình mạng Zigbee

Mạng Zigbee chia thành 3 loại chính:

- **Mạng hình sao** (Star Network): Gồm một nút trung tâm, tất cả các nút khác đều được kết nối với nút trung tâm này, mạng hình sao bị hạn chế khoảng cách và sự mở rộng.
- **Mạng hình cây** (Cluster Tree Network): Gồm một nút trung tâm, các nút khác được liên kết với nhau theo mô hình giống một cái dễ cây, mạng này có khả năng mở rộng cao, tăng khoảng cách và quy mô của hệ thống.
- **Mạng hình lưới** (Mesh Network): Gồm một nút trung tâm, các nút trong mạng đều có thể kết nối với nhau (trừ các ZED chỉ có thể kết nối với ZR của nó). Khi một đường truyền bị lỗi, sẽ tự động tìm ra một đường truyền khác, tăng tính tin cậy và kết nối trong mạng.



Hình 2.1: Các dạng mô hình Zigbee

Trong đó:

- **Coordinator (ZC):** Chỉ có duy nhất một coordinator trong một mạng, phụ trách việc thiết lập mạng.
- **Routers (ZR):** Có thể có nhiều router, các router có thể chuyển tiếp tín hiệu với nhau.
- **Device (ZED):** Có thể có nhiều ZED, các ZED không thể chuyển tiếp tín hiệu cho nhau. Có thể kích hoạt chế độ ngủ để tiết kiệm năng lượng.

2.1.3 Các dải tầng sóng hoạt động của Zigbee

ZigBee hoạt động ở 1 trong 3 dải tầng sóng:

- Dải 915 MHz cho khu vực Bắc Mỹ: Trong giải này chỉ có 1 kênh (kênh số 0) và tốc độ truyền khá thấp chỉ khoảng 20kb/s
- Dải 868 MHz cho châu Âu, Nhật: Trong giải này chỉ có 1 kênh (kênh số 0) và tốc độ truyền khá thấp chỉ khoảng 20kb/s
- Và dải 2.4 GHz cho các nước khác: Có tới 16 kênh tín hiệu từ 11-26 và tốc độ truyền tải rất cao tới 250kb/s

2.1.4 Ưu và nhược điểm của giao thức Zigbee

- **Ưu và nhược điểm**

Bảng 2.1: Ưu điểm và nhược điểm của mạng Zigbee

Ưu điểm	Nhược điểm
- Giá thành thấp - Tiêu thụ công suất nhỏ, giúp tiết kiệm năng lượng - Kiến trúc mạng linh hoạt, có thể mở rộng dễ dàng - Số lượng các nút lớn (65000) - Độ trễ thấp - Tính bảo mật cao nhờ sử dụng mà hóa AES-128	- Lỗi ở một điểm chính có thể gây lỗi hệ thống. - Tốc độ truyền thấp - Chưa có đầy đủ các thiết bị để phát triển

Mỗi chuẩn không dây đều có ưu điểm và nhược điểm khác nhau, tùy vào mục đích sử dụng mà ta chọn chuẩn phù hợp để tiết kiệm chi phí hơn.

Để cho rõ ràng hơn, ta hãy làm một phép so sánh giữa chuẩn Zigbee và một chuẩn không dây cũng khá phổ biến khác:

Bảng 2.2: So sánh giao thức Zigbee, Wifi và Bluetooth

Đặc tính	Zigbee	Wi-Fi	Bluetooth
Tiêu chuẩn	IEEE 802.15.4	IEEE 802.11	IEEE 802.15.1
Tốc độ truyền	250 Kbits/s	11 – 105 Mbits/s	723 Kbits/s
Phạm vi làm việc	0 - 3200m	0 - 50m	0 - 10m
Tần số hoạt dộng	915 MHz (Bắc mỹ) 868 MHz (Châu Âu, Nhật) 2.4 GHz các nước khác	2.4 GHz và 5.0 GHz	2.4 GHz
Độ phức tạp	Thấp	Cao	Cao
Năng lượng tiêu thụ	Rất thấp	Cao	Cao
Độ an toàn	128 bit mã hóa	Xác thực SSID	64-128 bit mã hóa
Nút mạng	65000	32	8

Nhìn vào bảng 2.2 chúng ta có thể thấy rằng chuẩn Zigbee phù hợp hơn với những ứng dụng cho nhiều phần tử, yêu cầu độ linh hoạt cao, tiêu thụ công suất nhỏ so với mạng Wi-Fi và Bluetooth.

- **Các ứng dụng của công nghệ Zigbee**

Công nghệ Zigbee rất phù hợp triển khai các hệ thống mạng không dây diện rộng với chi phí thấp và tiết kiệm năng lượng, có khả năng hoạt động trong thời gian dài.

Hiện nay công nghệ Zigbee đã được áp dụng rất phổ biến trong các ứng dụng như:

- **Tự động hóa nhà:** Công nghệ ZigBee chứng tỏ là công nghệ đáng tin cậy nhất trong việc hiện thực hóa tự động hóa nhà (Smarthome). Các ứng dụng khác nhau như kiểm soát và giám sát mức tiêu thụ năng lượng, quản lý nước, kiểm soát ánh sáng, vv đã được thực hiện dễ dàng hơn thông qua tự động hóa bằng công nghệ ZigBee.
- **Tự động hóa công nghiệp:** Các thiết bị RFID dựa trên ZigBee giúp cung cấp quản lý truy cập đáng tin cậy trong các ngành công nghiệp. Các ứng dụng khác trong các ngành công nghiệp bao gồm kiểm soát quá trình, quản lý năng lượng, theo dõi nhân sự, v.v.
- **Tự động hóa chăm sóc sức khỏe:** Một ví dụ phổ biến của tự động hóa chăm sóc sức khỏe là theo dõi sức khỏe từ xa. Một người đeo thiết bị ZigBee với cảm biến đo thông số cơ thể thu thập thông tin sức khỏe. Thông tin này được truyền trên mạng ZigBee đến mạng giao thức Internet (IP) và sau đó đến nhân viên chăm sóc sức khỏe (bác sĩ hoặc y tá), người sau đó sẽ kê đơn thuốc phù hợp dựa trên thông tin nhận được.
- **Đo sáng thông minh:** Các hoạt động từ xa của Zigbee trong đo sáng thông minh bao gồm phản ứng tiêu thụ năng lượng, hỗ trợ giá, bảo mật đối với hành vi trộm cắp điện, v.v.
- **Giám sát lưới điện thông minh:** Các hoạt động của Zigbee trong lưới điện thông minh này bao gồm giám sát nhiệt độ từ xa, định vị lỗi, quản lý công suất phản kháng, v.v.

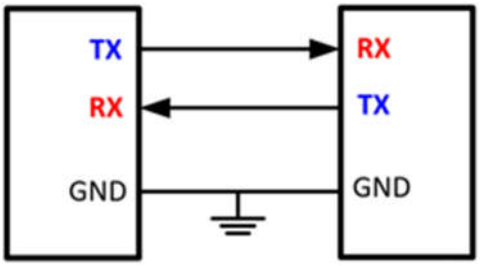
Hình 2.2: Các ứng dụng của giao thức Zigbee

2.2 Các chuẩn giao tiếp.

2.2.1 Chuẩn giao tiếp UART

UART có tên đầy đủ là Universal Asynchronous Receiver – Transmitter. Nó là một mạch tích hợp được sử dụng trong việc truyền dẫn dữ liệu nối tiếp giữa máy tính và các thiết bị ngoại vi.

UART có chức năng chính là truyền dữ liệu nối tiếp. Trong UART, giao tiếp giữa hai thiết bị có thể được thực hiện theo hai phương thức là giao tiếp dữ liệu nối tiếp và giao tiếp dữ liệu song song.

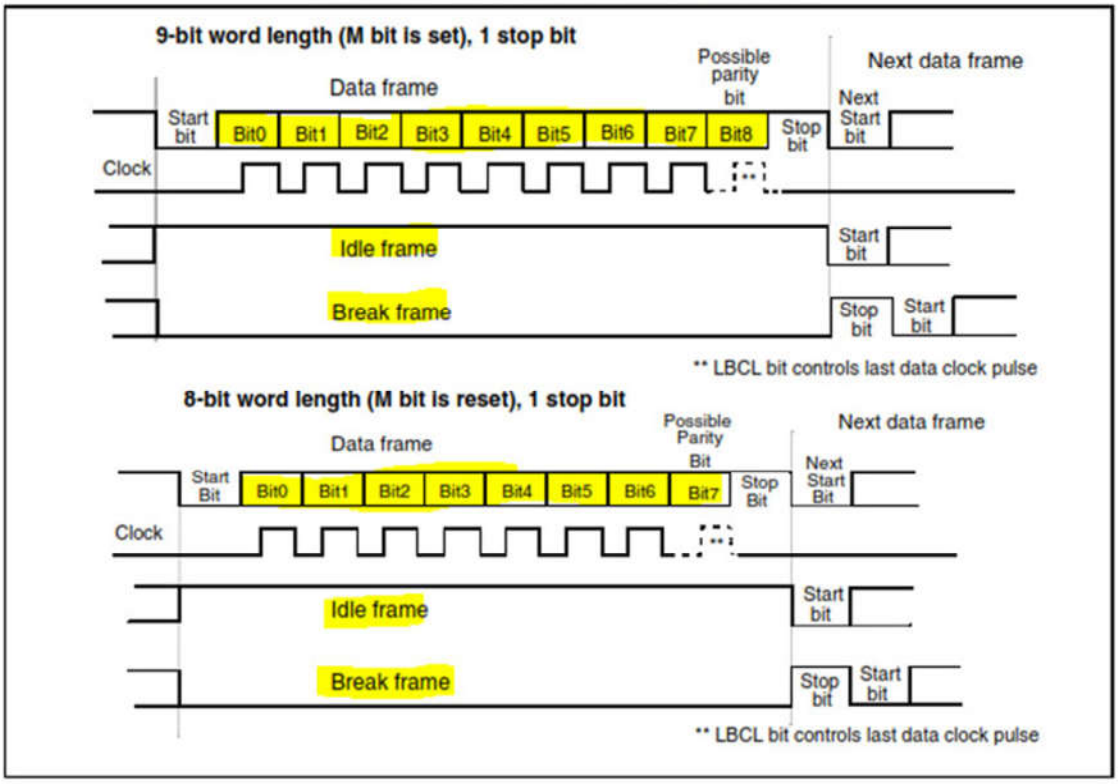


Hình 2.3: Truyền dữ liệu nối tiếp UART

Giao tiếp dữ liệu nối tiếp có nghĩa là dữ liệu có thể được truyền qua một cáp hoặc một đường dây ở dạng bit-bit và nó chỉ cần hai cáp. Nó yêu cầu số lượng mạch cũng như dây rất ít. Giao tiếp này rất hữu ích trong các mạch ghép hơn giao tiếp song song.

Giao tiếp dữ liệu song song có nghĩa là dữ liệu có thể được truyền qua nhiều cáp cùng một lúc. Truyền dữ liệu song song yêu cầu số lượng mạch và dây nhiều.

Vì vậy, giao tiếp song song tốn kém nhưng đổi lại rất nhanh, nó đòi hỏi phần cứng và cáp bổ sung.



Hình 2.4: Sơ đồ gói dữ liệu truyền nhận trong UART

Trong giao tiếp UART có các thông số chính:

- Baud rate (tốc độ baud): Khoảng thời gian để 1 bit được truyền đi. Phải được cài đặt giống nhau ở cả phần gửi và nhận
- Frame (khung truyền): Khung truyền quy định về mỗi lần truyền bao nhiêu bit
- Start bit: Là bit đầu tiên được truyền trong 1 Frame. Báo hiệu cho thiết bị nhận có một gói dữ liệu sắp đc truyền đến. Đây là bit bắt buộc
- Data: Dữ liệu cần truyền. Bit có trọng số nhỏ nhất LSB được truyền trước sau đó đến bit MSB.
- Parity bit: Kiểm tra dữ liệu truyền có đúng không
- Stop bit: Là 1 hoặc các bit báo cho thiết bị rằng các bit đã được gửi xong. Thiết bị nhận sẽ tiến hành kiểm tra khung truyền nhằm đảm bảo tính đúng đắn của dữ liệu. Đây là bit bắt buộc

➢ **Sơ đồ giao tiếp UART**

UART có sơ đồ bao gồm hai thành phần là thiết bị gửi và thiết bị thu. Phần thiết bị gửi bao gồm ba khối là thanh ghi giữ truyền, thanh ghi dịch chuyển và logic điều

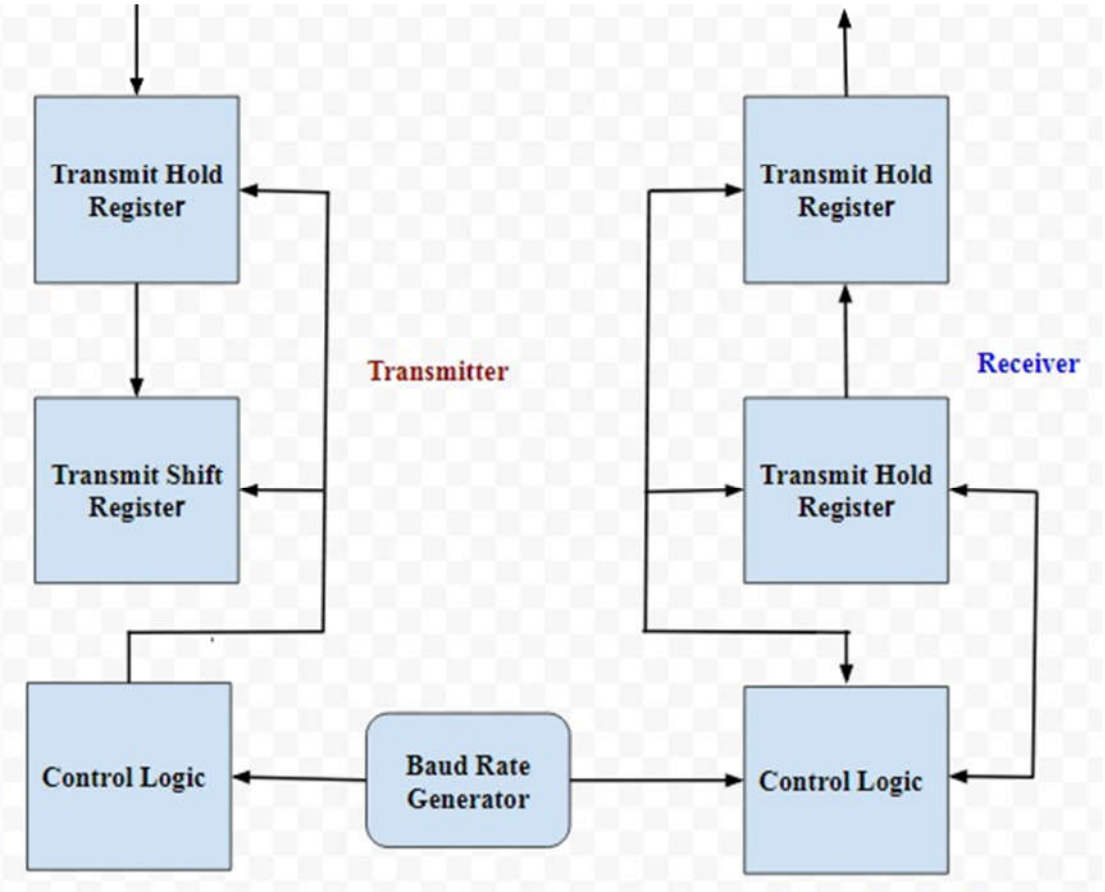
khiển. Tương tự, phần thiết bị thu bao gồm một thanh ghi giữ, thanh ghi thay đổi và logic điều khiển.

Hai phần này thường được cung cấp bởi một bộ tạo tốc độ baud. Trình tạo này được sử dụng để tạo tốc độ khi phần máy phát và phần máy thu phải truyền hoặc nhận dữ liệu.

Thanh ghi giữ trong thiết bị gửi bao gồm byte dữ liệu được truyền. Các thanh ghi thay đổi trong thiết bị gửi và nhận di chuyển các bit sang phải hoặc trái cho đến khi một byte dữ liệu được truyền hoặc nhận. Một logic điều khiển đọc (hoặc) ghi được sử dụng để biết khi nào nên đọc hoặc viết.

Máy phát tốc độ baud giữa thiết bị gửi và thiết bị nhận tạo ra tốc độ dao động từ 110 bps đến 230400 bps. Thông thường, tốc độ truyền của vi điều khiển là 9600 đến 115200.

Để bắt đầu cho việc truyền dữ liệu bằng UART, một Start bit được gửi đi, sau đó là các bit dữ liệu và kết thúc quá trình truyền là Stop bit.



Hình 2.5 Sơ đồ khối giao tiếp UART

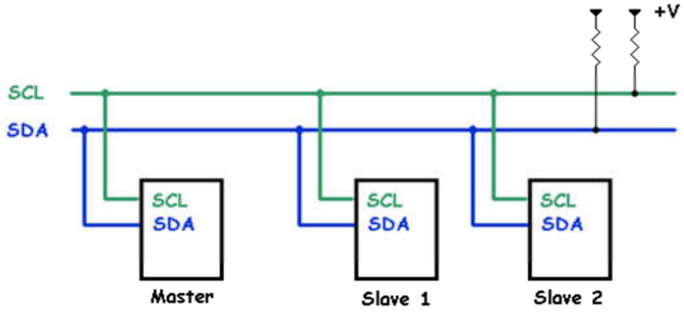
2.2.2 Chuẩn giao tiếp I2C

I2C là tên viết tắt của cụm từ Inter-Intergrated Circuit. Đây là đường Bus giao tiếp giữa các IC với nhau. I2C mặc dù được phát triển bởi Philips, nhưng nó đã được rất nhiều nhà sản xuất IC trên thế giới sử dụng. I2C trở thành một chuẩn công nghiệp cho các giao tiếp điều khiển, có thể kể ra đây một vài tên tuổi ngoài Philips như: Texas Intrument(TI), MaximDallas, analog Device, National Semiconductor ... Bus I2C được sử dụng làm bus giao tiếp ngoại vi cho rất nhiều loại IC khác nhau như các loại Vi điều khiển 8051, PIC, AVR, ARM... chip nhớ như: RAM tĩnh (Static Ram), EEPROM, bộ chuyển đổi tương tự số (ADC), số tương tự(DAC), IC điều khiển LCD, LED...

I2C sử dụng hai đường truyền tín hiệu: một đường xung nhịp đồng hồ chỉ theo một hướng (SCL) do Master phát đi (thông thường ở 100kHz và 400kHz. Mức cao nhất là 1Mhz và 3.4MHz) và một đường dữ liệu hai hướng (SDA). SCL và SDA luôn được kéo lên nguồn bằng một điện trở có giá trị từ 1 KOhm đến 4,7 KOhm tùy vào từng thiết bị và chuẩn giao tiếp. Các chế độ hoạt động của I2C bao gồm:

- Chế độ chuẩn (standard mode) hoạt động ở tốc độ 100 Kbit/s.
- Chế độ tốc độ thấp (low-speed mode) hoạt động ở tốc độ 10 Kbit/s.

Có rất nhiều thiết bị có thể cùng được kết nối vào một bus I2C, tuy nhiên sẽ không xảy ra chuyện nhầm lẫn giữa các thiết bị, bởi mỗi thiết bị sẽ được nhận ra bởi một địa chỉ duy nhất với một quan hệ chủ/tớ tồn tại trong suốt thời gian kết nối. Mỗi thiết bị có thể hoạt động như là thiết bị nhận hoặc truyền dữ liệu hay có thể vừa truyền vừa nhận. Hoạt động truyền hay nhận còn tùy thuộc vào việc thiết bị đó là chủ (master) hãy tớ (slave).



Hình 2.6: Sơ đồ kết nối I2C

Bản chất của I2C là dữ liệu trên đường SDA chỉ được ghi nhận ở sườn lên của chân CLK. Do vậy xung clock có thể không cần chính xác tốc độ là 1MHz hay 3.4Mhz. Lợi dụng điểm này có thể sử dụng 2 chân GPIO để làm chân giao tiếp I2C mềm mà không nhất thiết cần một chân CLK tạo xung với tốc độ chính xác.

Điểm mạnh của I2C chính là hiệu suất và sự đơn giản của nó: một khối điều khiển trung tâm có thể điều khiển cả một mạng thiết bị mà chỉ cần hai lối ra điều khiển.

2.3 Giới thiệu về Arduino

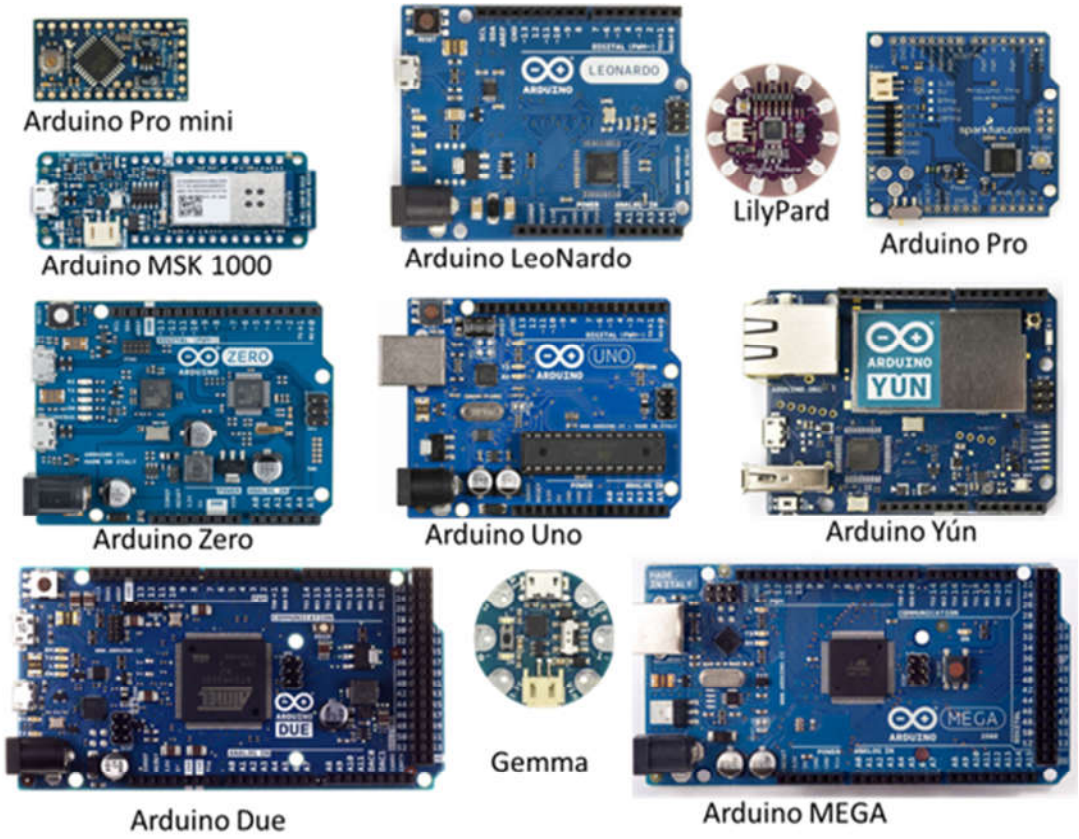
Arduino là một bo mạch vi xử lý được dùng để lập trình tương tác với các thiết bị phần cứng như cảm biến, động cơ, đèn hoặc các thiết bị khác. Arduino ra đời tại thị trấn Ivrea thuộc nước Ý và được đặt theo tên một vị vua vào thế kỷ thứ 9 là King Arduin. Arduino chính thức được đưa ra giới thiệu vào năm 2005 như là một công cụ khiêm tốn dành cho các sinh viên của giáo sư Massimo Banzi, là một trong những người phát triển Arduino, tại trường Interaction Design Instistute Ivrea (IDII).

Arduino sử dụng dòng vi xử lý AVR Atmel 8 bit với hai chip phổ biến nhất là ATmega328 và ATmega2560. Các dòng vi xử lý này cho phép lập trình các ứng dụng điều khiển phức tạp do được trang bị cấu hình mạnh với các loại bộ nhớ ROM, RAM và Flash, các ngõ vào ra digital I/O trong đó có nhiều ngõ có khả năng xuất tín hiệu PWM, các ngõ đọc tín hiệu analog và các chuẩn giao tiếp đa dạng như UART, SPI, I2C. Đi cùng với nó là một môi trường phát triển tích hợp (IDE) chạy trên các máy tính cá nhân thông thường và cho phép người dùng viết các chương trình cho Aduino bằng ngôn ngữ C hoặc C++.

Một mạch Arduino bao gồm một vi điều khiển AVR với nhiều linh kiện bổ sung giúp dễ dàng lập trình và có thể mở rộng với các mạch khác. Một khía cạnh quan trọng của Arduino là các kết nối tiêu chuẩn của nó, cho phép người dùng kết nối với CPU của board với các module thêm vào có thể dễ dàng chuyển đổi, được gọi là shield. Vài shield truyền thông với board Arduino trực tiếp thông qua các chân khác nhau, nhưng nhiều shield được định địa chỉ thông qua serial bus I2C nhiều shield có thể được xếp chồng và sử dụng dưới dạng song song.

Các phiên bản Arduino có trên thị trường hiện nay như: Arduino Mega 2560, Arduino Uno, Arduino 101, Arduino Pro, Arduino Zero, Arduino Due, Lilypad

Arduino USB, Lilypad Arduino Main Board, Lilypad Arduino Simple, Lilypad Arduino Simple Snap.



Hình 2.7: Các mạch Arduino thông dụng

CHƯƠNG 3: THIẾT KẾ HỆ THỐNG

3.1 Giới thiệu

Hệ thống gồm 2 bộ phận chính: bộ điều khiển chính và bộ điều khiển phụ (dùng để điều khiển đèn).

- Bộ điều khiển chính sẽ bao gồm: 1 Arduino Mega 2560, 1 module Xbee S2, 1 module DS1307, 1 màn hình cảm ứng HMI, dùng để so sánh thời gian người dùng cài đặt và thời gian thực để thực hiện gửi dữ liệu qua các bộ điều khiển phụ.
- Bộ điều khiển phụ sẽ bao gồm: 1 Arduino UNO, 1 module Xbee S2, 1 module cảm biến dòng, 1 đèn sợi tóc, 1 mạch điều khiển tải xoay chiều sử dụng Triac, dùng để điều khiển và hồi tiếp trạng thái đèn.

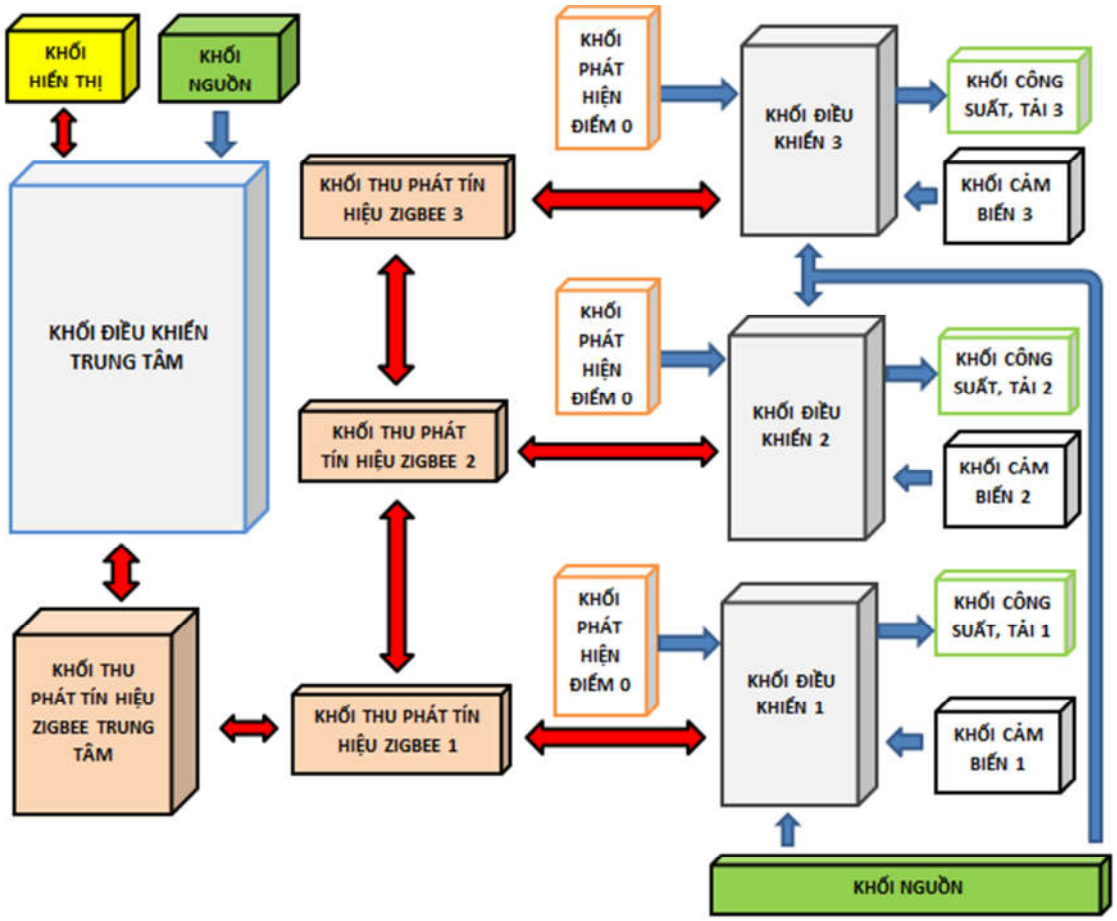
Nguyên lý hoạt động của hệ thống: hệ thống sẽ có 2 chế độ là chỉnh tay hoặc tự động.

- Chế độ tự động: khi người dùng cài đặt thời gian sáng và tắt theo mong muốn, bộ điều khiển chính sẽ gửi dữ liệu điều khiển tới bộ điều khiển phụ để điều khiển đèn. Bộ điều khiển chính sẽ luôn giám sát tình trạng hoạt động, hỏng hóc của đèn.
- Chế độ chỉnh tay: người dùng sẽ điều khiển thông qua màn hình cảm ứng để điều khiển độ sáng đèn, tắt, bật theo mong muốn.

3.2 Thiết kế hệ thống

3.2.1 Thiết kế sơ đồ khối của hệ thống

Dựa vào các yêu cầu đề ra nhóm đã thiết kế hệ thống theo sơ đồ khối sau:



Hình 3.1: Sơ đồ khối của hệ thống

Chức năng của từng khối:

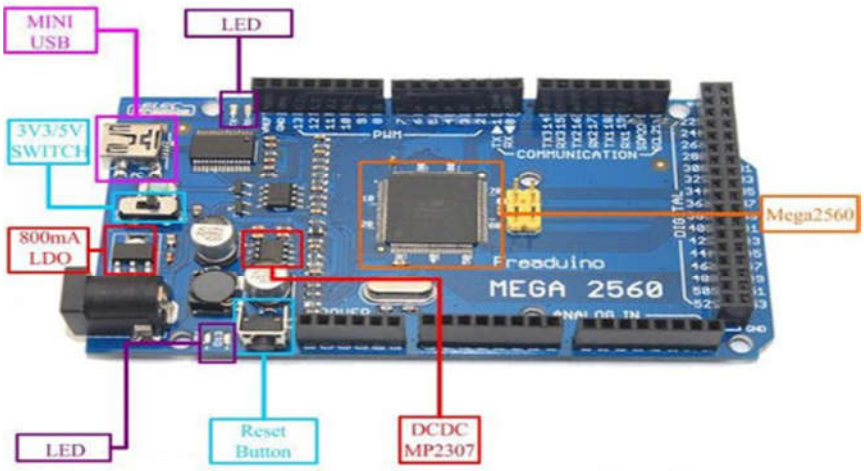
- **Khối điều khiển trung tâm:** Khối này có chức năng nhận lệnh điều khiển từ màn hình HMI sau đó gửi lệnh điều khiển đến bộ điều khiển đèn và thu thập dữ liệu phản hồi từ khối điều khiển đèn thông qua khối thu phát tín hiệu Zigbee sau đó hiển thị ra màn hình.
- **Khối điều khiển:** Khối này có chức năng nhận lệnh điều khiển từ khối điều khiển trung tâm thông qua khối thu phát tín hiệu Zigbee sau đó gửi lệnh điều khiển độ sáng của đèn và thu thập dữ liệu từ cảm biến để gửi về khối điều khiển trung tâm.
- **Khối phát hiện điểm 0:** Khối này có chức năng dò điểm 0 của điện xoay chiều sau đó gửi tín hiệu về cho vi điều khiển.
- **Khối công suất, tải:** Khối này có chức năng nhận lệnh điều khiển từ vi điều khiển sau đó điều khiển công suất của tải.

- **Khối thu phát tín hiệu Zigbee:** Khối này có chức năng truyền và nhận dữ liệu giữa hai khối điều khiển trung tâm và khối điều khiển đèn.
- **Khối hiển thị:** Khối này có chức năng hiển thị thông tin lên màn hình HMI để người dùng có thể dễ dàng điều khiển và giám sát hệ thống.
- **Khối thời gian thực:** Khối này có chức năng cung cấp thời gian hiện tại cho vi điều khiển để hiển thị ra màn hình.
- **Khối cảm biến:** Khối này có chức năng phát hiện được đèn sáng hay đèn hỏng thông qua cảm biến quang sau đó gửi dữ liệu về khối điều khiển đèn.
- **Khối nguồn:** Khối này là khối không thể thiếu trong mạch điện tử, nó có chức năng cung cấp điện áp cho mạch hoạt động. Trong đề tài nhóm sử dụng hai khối nguồn để cấp cho mạch. Khối nguồn cấp nguồn cho Arduino và khối nguồn 220AC để cấp cho đèn.

3.2.2 Tính toán và thiết kế mạch

a. Khối điều khiển trung tâm

- **Chức năng:** Khối này có chức năng nhận lệnh điều khiển từ màn hình HMI sau đó gửi lệnh điều khiển đến bộ điều khiển đèn và thu thập dữ liệu phản hồi từ khối điều khiển đèn thông qua khối thu phát tín hiệu Zigbee sau đó hiển thị ra màn hình.
- **Phân tích lựa chọn linh kiện:** Để đáp ứng được các chức năng trên nhóm đã chọn Arduino Mega 2560 làm bộ điều khiển chính. Arduino Mega 2560 là sản phẩm được phát triển trên con chip ATmega2560 được phát triển dựa trên dòng mạch Mega là dòng mạch được cải tiến so với Arduino Uno (54 chân digital IO và 16 chân analog IO). Đặc biệt bộ nhớ flash của MEGA được tăng lên một cách đáng kể, gấp 4 lần so với bảng mạch Arduino Uno R3. Việc được trang bị 3 timer và 6 cổng interrupt khiến Arduino Mega 2560 có thể xử lý song song nhiều luồng dữ liệu số cũng như tương tự. Ngoài ra Arduino Mega 2560 được trang bị 4 cổng kết nối UART giúp cho nó có thể kết nối với nhiều thiết bị cùng chuẩn giao tiếp UART. Arduino Mega 2560 cũng tương thích với tất cả Shield của Arduino Uno. Như các Shield sử dụng để điều khiển motor, kết nối wifi hay kết nối với các thiết bị điện tử trong nhà.



Hình 3.2: Mô-đun Arduino Mega 2560

➢ **Thông số kỹ thuật chính:**

Bảng 3.1: Thông số chính của Arduino Mega2560

Vi điều khiển	ATmega2560
Điện áp hoạt động	5V
Điện áp khuyến nghị	6-9V
Số chân I/O	54 chân(trong đó 15 chân PWM)
Số chân input analog	16 chân
Giao tiếp UART	4 bộ
Giao tiếp SPI	1 bộ(chân 50 - 53), dùng thư viện SPI của Arduino
Giao tiếp I2C	1 bộ
Cổng interrupt	6 cổng
Bộ nhớ Flash	256 KB, 8 KB sử dụng cho Bootloader
SRAM	8 KB
EEPROM	4 KB
Giao động	16 MHz

➢ **Tính toán mạch**: Mạch chỉ là Shield kết nối Arduino Mega 2560 với các thiết bị ngoại vi. Mạch sử dụng chỉ 1 đèn led để báo nguồn.

Công thức tính điện trở cho led:

$$R_{led} = \frac{V_S - V_f}{I} \quad (\Omega)$$ Trong đó:

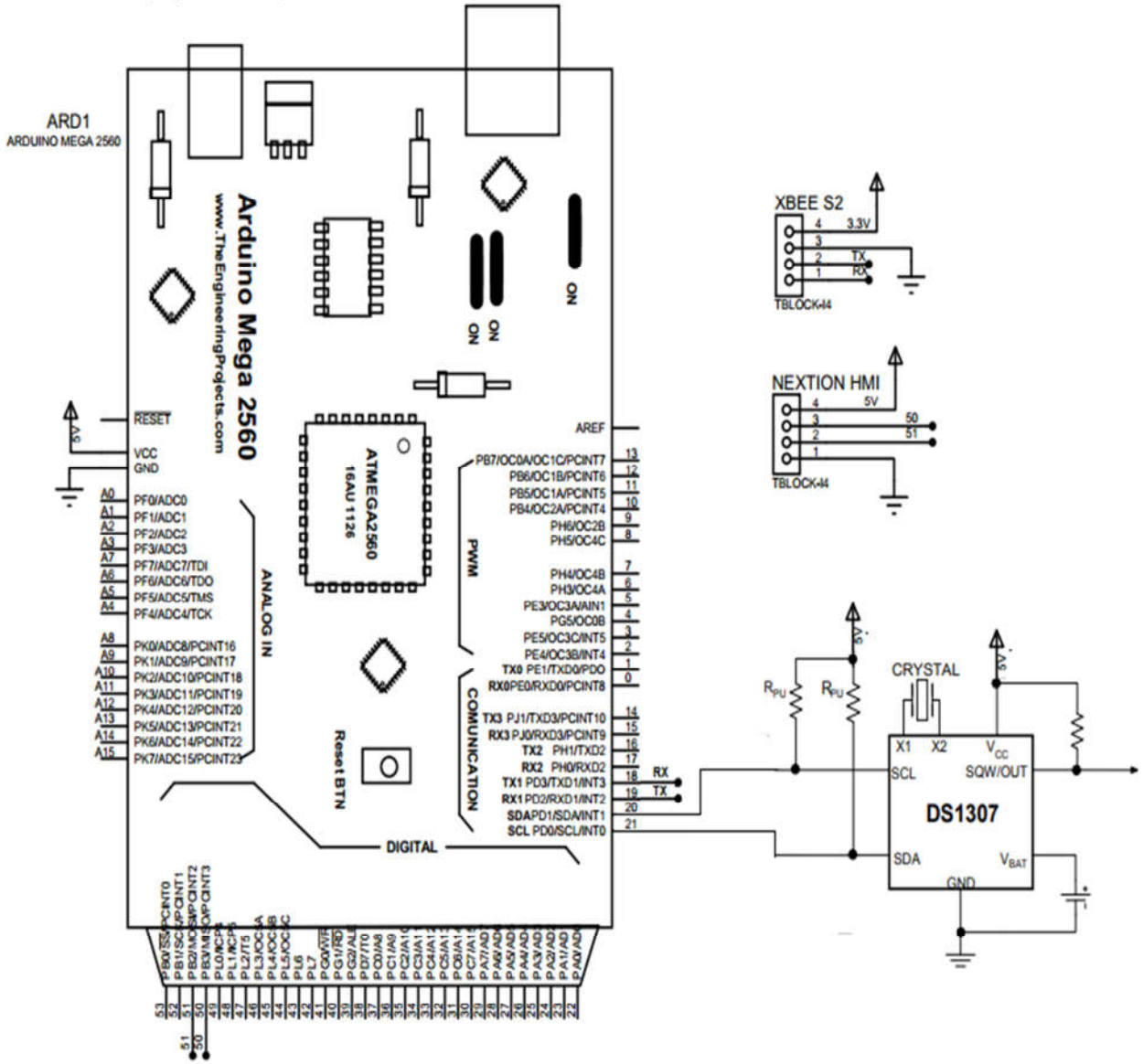
V_S là điện áp nguồn cấp, V_f là điện áp rơi trên led, I là cường độ dòng đi qua led, tính bằng mA.

Nguồn V_S = 5V, chọn dòng qua led khoảng 10mA

Điện áp rơi trên led V_f = 3V

$$\rightarrow R_{led} = \frac{5-3}{0.01} = 200\ (\Omega) \rightarrow$$ Chọn R_{led} thực tế 220 Ω

➢ **Sơ đồ nguyên lí mạch:**



Hình 3.3: Sơ đồ nguyên lý mạch điều khiển trung tâm

- **Giải thích sơ đồ nguyên lý:**

Khi có lệnh điều khiển từ chân Tx của màn hình HMI đến chân Rx của Arduino thì Arduino sẽ xử lí sau đó gửi tín hiệu đến Xbee để phát tín hiệu điều khiển đi. Khi Arduino nhận được tín hiệu gửi về từ Xbee và RTC DS1307 thì nó sẽ xử lý sau đó gửi dữ liệu từ chân Tx đến chân Rx của màn hình HMI để hiển thị ra màn hình.

b. Khối phát hiện điểm 0

- **Chức năng:** Khối này có chức năng dò điểm 0 của điện xoay chiều sau đó gửi tín hiệu về cho vi điều khiển.
- **Phân tích lựa chọn linh kiện:**

Điện áp đầu vào là 220 AC, tần số 50Hz để gửi được tín hiện về vi điều khiển phải chuyển đổi điện áp xoay chiều thành một chiều sau đó được cách ly và gửi tín hiệu về vi điều khiển. Ngõ ra có điện áp bằng 5VDC và cường độ dòng điện khoảng từ 5mA đến 20mA.

Nhóm quyết định chọn:

- Diode cầu KBP 307: V_{RMS} = 700V, I_{out} = 3A
- Diode Zener: V_{out} = 5.1V, I_{ZT} = 49mA, I_{ZM} = 890 mA(25^0C)
- PC817: V_F = 1.4V, I_F = 20mA, V_{CE}= 25V.

Vì các thông số dòng điện và điện áp của những linh kiện trên đáp ứng yêu cầu đề ra.

- **Tính toán mạch:**

- Điện áp đầu vào của mạch là 220VAC, tần số 50Hz đi qua Diode cầu chỉnh lưu dòng xoay chiều thành một chiều. Khi đó điện áp ra của Diode cầu sẽ có biên độ đỉnh bằng: $V_{in} = 220\sqrt{2}$ (V)

- Khi phân cực ngược Diode Zener sẽ ghim một mức điện áp tại hai đầu bằng V_Z = 5V, chọn dòng qua Diode Zener là 30mA

$$R = \frac{V_{in}-V_z}{I_z} = \frac{220\sqrt{2}-5}{0.03} = 10204 \rightarrow \text{Chọn R = 10 K}\Omega$$

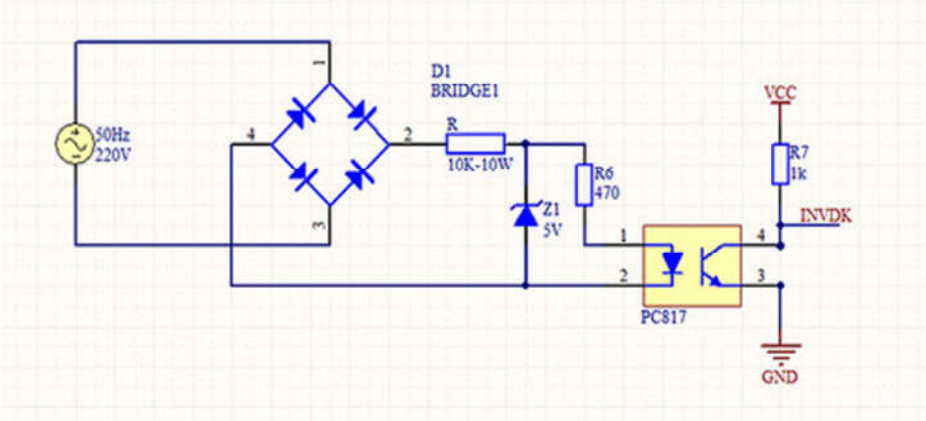
- Điện áp rơi trên led của PC817 bằng V_F = 1.2V mà điện áp V_z = 5V nên ta cần thêm điện trở để hạn dòng. Ta có:

$$5\text{mA} \leq I_p = \frac{V_Z-V_F}{R_6} = \frac{5-1.2}{R_6} \leq 20\text{mA}$$

Chọn dòng I_p khoảng 10mA → Chọn R_6 = 470 Ω

- Để đảm bảo Arduino hoạt động an toàn và tiết kiệm năng lượng ta chọn dòng tiêu thụ khoảng 5mA. Chọn $R_7 = \frac{V_{cc}}{5mA} = \frac{5}{0.005}$ =1 KΩ

➢ **Sơ đồ nguyên lí:**



Hình 3.4: Sơ đồ nguyên lý mạch phất hiện điểm 0

➢ **Giải thich sơ đồ nguyên lí:**

Khi cấp điện 220V AC cho mạch, lúc này dòng điện đi qua Diode cầu chỉnh lưu thành dòng điện một chiều có biên độ đỉnh là $220\sqrt{2}$V. Sau đó đi qua điện trở công suất để hạn dòng và nối tiếp với Diode Zener, lúc này Diode Zener sẽ phân cực ngược tại hai đầu Diode Zener sẽ ghim lại một mức điện áp cố định V_z = 5V bằng giá trị ghi trên nó.

Khi điện áp xoay chiều lớn hơn 5V, điện áp hai đầu Diode Zener đi qua điện trở 470Ω để hạn dòng cho PC817 làm led trong PC817 sáng để kích cho photo transistor dẫn khi đó sẽ tác động một điện áp mức thấp về cho Arduino.

Khi điện áp xoay chiều nhỏ hơn 5V và giảm dần về 0V, lúc này điện áp hai đầu diode zener sẽ không đủ cung cấp cho led trong PC817 sáng và Arduino sẽ nhận được tín hiệu điện áp mức cao. Như vậy cứ mỗi khi điện áp xoay chiều về 0V thì mạch sẽ phát hiện được và gửi tín hiệu về cho Arduino.

c. Khối công suất, tải:

➢ **Chức năng:** Khối này có chức năng nhận lệnh điều khiển từ vi điều khiển sau đó điều khiển công suất của tải.

➢ **Phân tích lựa chọn linh kiện:**

Điện áp ngõ vào là điện áp từ ngõ ra của vi điều khiển có mức điện áp bằng 5V và dòng điện khoảng 20mA. Lấy điện áp này điều khiển công suất của tải 220V xoay chiều nên phải được cách ly và ngõ ra chịu được dòng khoảng 1A.

Nhóm đã quyết định chọn các linh kiện sau:

- MOC3020: $V_f = 3V$, $I_f = 50mA$
- Triac BTA16: $V_{DRM} = 600V$, $I_T = 16A$, $I_{GT} = 100mA$
- Đèn sợi đốt: $V_{in} = 220AC$, P = 40W

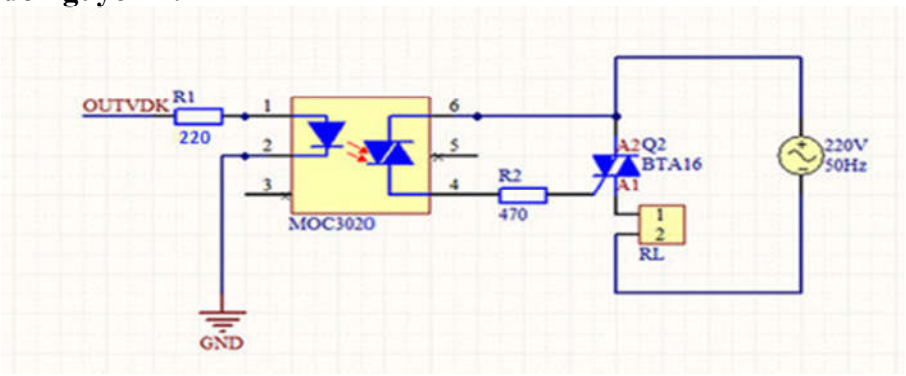
Vì các thông số của những linh kiện trên đáp ứng được yêu cầu đề ra.

➢ **Tính toán mạch:**

Điện áp đầu ra của Arduino là $V_O = 5V$ trong khi đó điện áp đầu vào của MOC3020 là khoảng $V_f = 1.2V$ nên ta phải cần có điện trở hạn dòng cho led. Dòng ra của Arduino là $I_o = 20mA$.

→ Chọn $R_1 = 220\ \Omega$

➢ **Sơ đồ nguyên lí:**



Hình 3.5: Sơ đồ nguyên lý mạch công suất

➢ **Giải thích sơ đồ nguyên lý:**

Sau một khoảng thời gian t1 khi phát hiện điểm 0 của điện xoay chiều, Arduino sẽ xuất một điện áp 5V cho led trong MOC3020 sáng để kích cho photo triac dẫn, khi đó cực G của Triac BTA16 sẽ có dòng làm cho hai đầu A2 và A1 thông với nhau và làm đèn sáng.

Xét bán kì dương của dòng điện xoay chiều 220V có tần số 50Hz

→ Chu kỳ T = 20ms:

- Với t1 ≤ 1ms lúc này biên độ hình sin chưa đủ lớn để tạo dòng kích tại chân G cho Triac, do đó trong khoảng góc mở từ (0 - 1ms) Triac khóa.

- Với t1 > 2ms lúc này biên độ hình sin đủ lớn kích dòng cho chân G và Triac mở dẫn dòng từ A2 qua A1 từ (2ms - 10ms). Tại thời điểm t1 = 10ms (Điểm 0) của chu kỳ do điện áp cực A1, A2, G bằng 0 do đó Triac khóa.

Vì vậy chỉ cần điều chỉnh thời gian t1 ta có thể điều chỉnh được điện áp trung bình ra tải, từ đó có thể điều chỉnh độ sáng của đèn hay tốc độ động cơ.

d. Khối điều khiển phụ

- **Chức năng:**

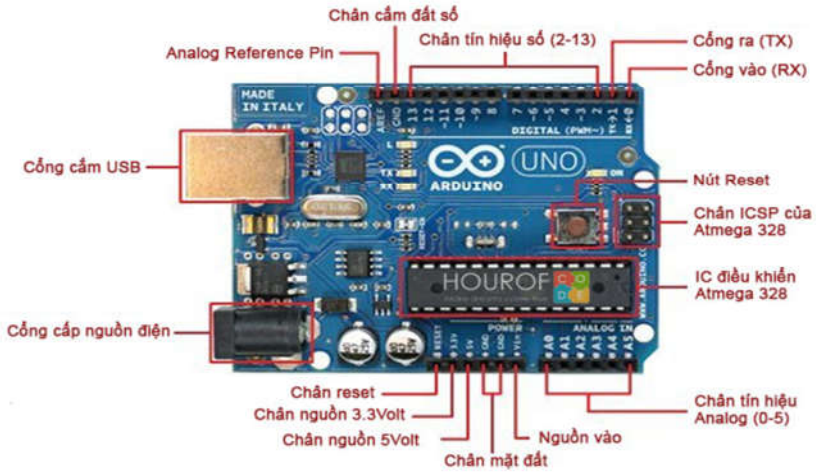
Khối này có chức năng nhận lệnh điều khiển từ khối điều khiển trung tâm thông qua khối thu phát tín hiệu Zigbee sau đó gửi lệnh điều khiển độ sáng của đèn và thu thập dữ liệu từ cảm biến để gửi về khối điều khiển trung tâm.

- **Phân tích lựa chọn linh kiện:**

Để đáp ứng được các chức năng trên nhóm đã lựa chọn Arduino Uno R3 làm linh kiện chính.

Arduino Uno R3 là một bảng mạch vi điều khiển mã nguồn mở dựa trên vi điều khiển AVR ATmega328, được kết nối trực tiếp với máy tính thông qua USB để giao tiếp với phần mềm lập trình IDE, tương thích với Windows, MAC hoặc Linux Systems.

Các ngôn ngữ lập trình như C và C ++ được sử dụng trong IDE. Aruino Uno R3 được sử dụng phổ biến trong việc tự thiết kế ra các mạch điện tử như điều khiển led, gửi dữ liệu lên lcd, điều khiển motor... hay được gắn thêm các Shield để kết nối nhiều module cảm biến khác để thực hiện thêm nhiều chức năng mở rộng như gửi dữ liệu qua wifi.



Hình 3.6: Mô-đun Arduino Uno R3

➢ **Các thông số kĩ thuật chính:**

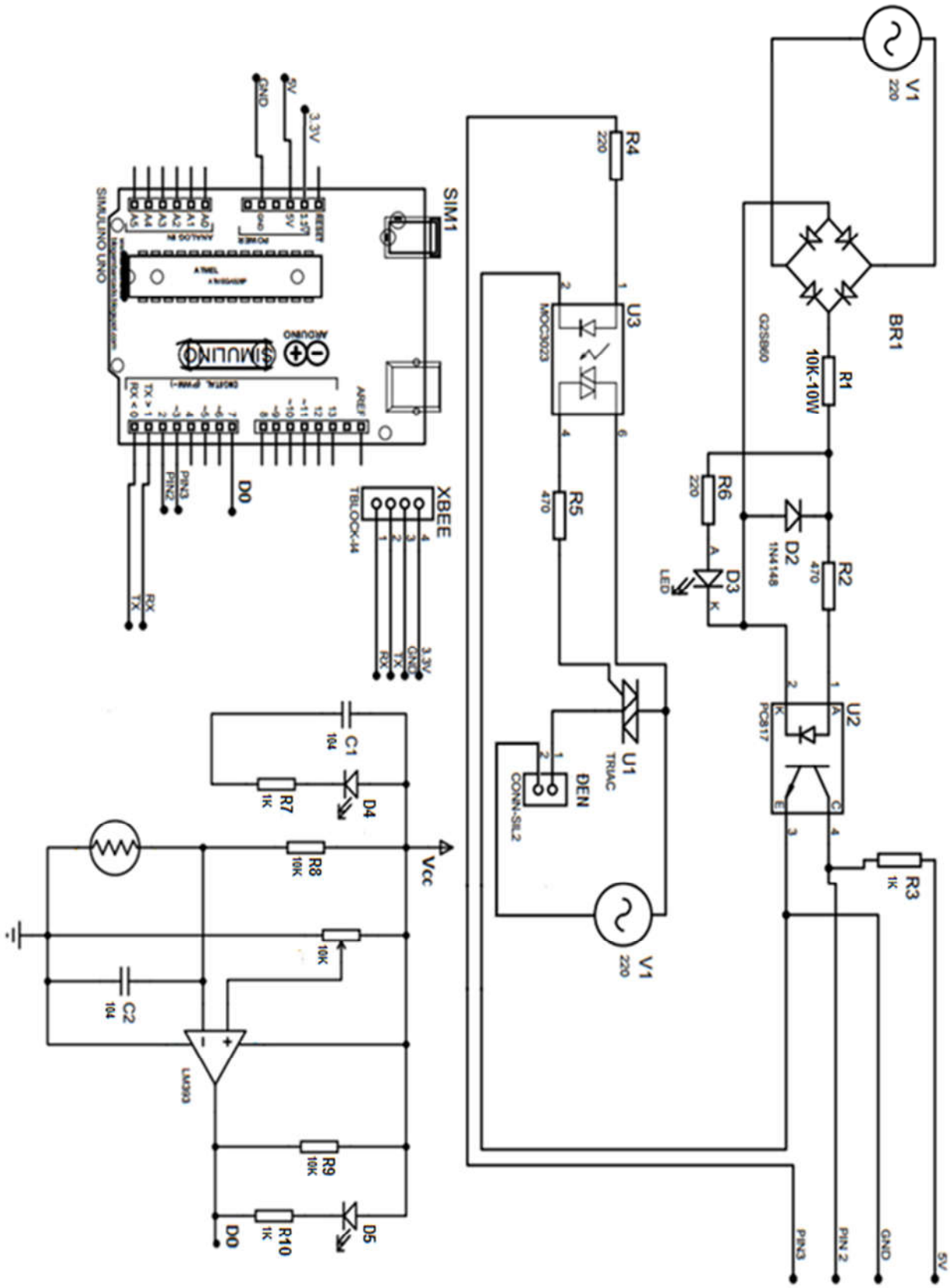
Bảng 3.2: Thông số chính của Arduino Uno R3

Vi điều khiển	ATmega328 họ 8bit
Điện áp hoạt động	5V DC (chỉ được cấp qua cổng USB)
Tần số hoạt động	16 MHz
Điện áp vào khuyên dùng	7-12V DC
Điện áp vào giới hạn	6-9V DC
Số chân Digital I/O	14 (6 chân PWM)
Số chân Analog	6 (độ phân giải 10bit)
Dòng tối đa trên mỗi chân I/O	20 mA
Dòng ra tối đa (5V)	500 mA
Dòng ra tối đa (3.3V)	50 mA
Bộ nhớ flash	32 KB (ATmega328) với 0.5KB dùng bởi bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Giao động thạch anh	16MHz

➢ **Tính toán mạch:**

Mạch dùng để kết nối Arduino với các thiết bị ngoại vi, trong mạch chỉ sử dụng đèn led để báo nguồn và trạng thái.

➢ **Sơ đồ nguyên lí**



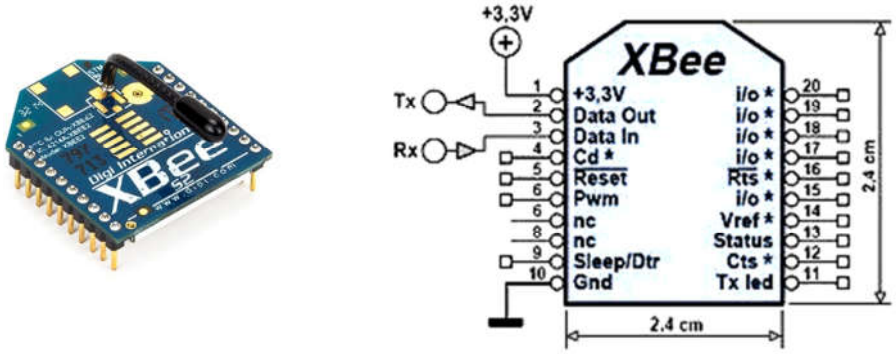
Hình 3.7: Sơ đồ nguyên lý mạch điều khiển phụ

- **Giải thích sơ đồ nguyên lí:**

  Khi mạch phát hiện điểm 0 bắt được điểm 0 của điện xoay chiều thì Arduino sẽ nhận được một tín hiệu điện áp gửi về, nếu Arduino nhận được lệnh điều khiển từ bộ điều khiển trung tâm gửi tới thông qua Xbee thì sau một khoảng thời gian t1 Arduino sẽ xuất một điện áp mức cao đến mạch điều khiển tải AC khi đó đèn sẽ sáng. Độ sáng của đèn phụ thuộc vào khoảng thời gian t1, t1 càng lớn thì đèn sáng càng yếu và ngược lại. Sau đó Arduino gửi dữ liệu nhận được từ cảm biến và gửi đến Xbee.

c. Khối thu phát tín hiệu Zigbee

- **Chức năng:** Khối này có chức năng truyền và nhận dữ liệu giữa hai khối điều khiển trung tâm và khối điều khiển đèn.
- **Phân tích lựa chọn linh kiện:** Để đáp ứng được các chức năng trên nhóm đã chọn Xbee S2 làm linh kiện chính để truyền nhận dữ liệu giữa các khối. Xbee là tên gọi của một họ các mô-đun giao tiếp không dây dùng để liên kết các thiết bị nối tiếp giao tiếp với vi điều khiển và được chế tạo bởi công ty phần cứng máy tính Digi International.



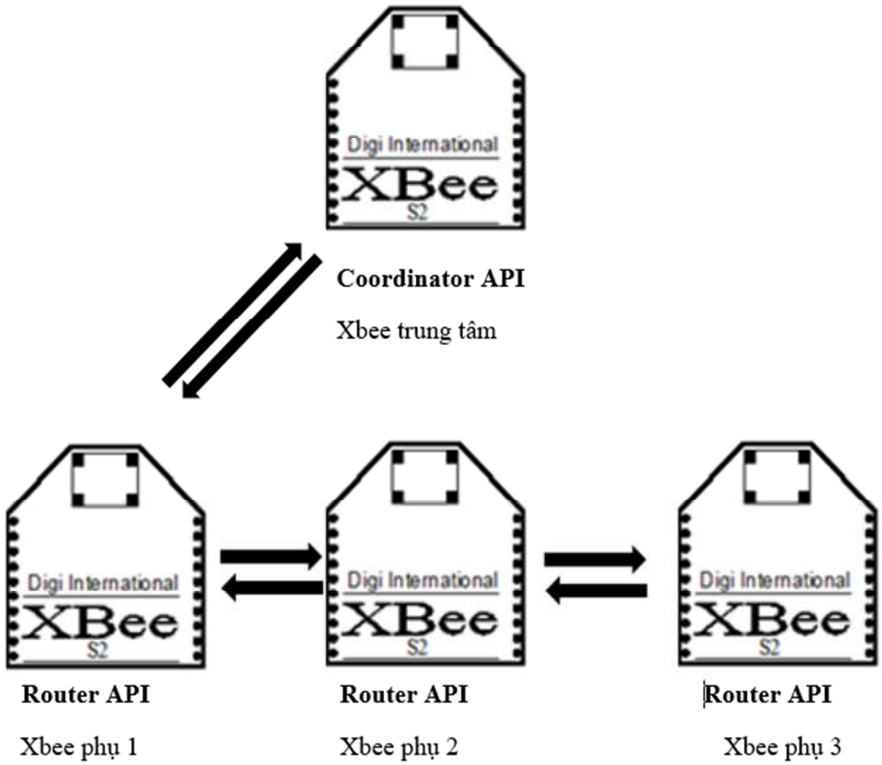
Hình 3.8: Mô-đun Xbee S2 và sơ đồ chân

- **Thông số kỹ thuật chính:**
  - Chuẩn giao tiếp: UART
  - Khoảng truyền nhận trong nhà (có vật cản): 40 m.
  - Khoảng truyền nhận ngoài trời (không có vật cản): 120 m.
  - Tốc độ truyền dữ liệu: 250kbps
  - Độ nhạy sóng: -96 dBm
  - Tần số hoạt động: 2.4GHz

- Công suất phát: 2mW (3 dBm)
- Nguồn cung cấp: 2.8-3.3 V
- Dòng ra chân Data Out: 45mA(3.3V)
- Dòng vào chân Data In: 50mA(3.3V)
- Nhiệt độ làm việc: -40^{0}C - 80^{0}C (trong công nghiệp)

➢ **Tính toán thiết kế mạch:**

Trong đề tài này nhóm sử dụng bốn trạm thu phát tín hiệu Zigbee và được thiết kế theo sơ đồ sau:



Hình 3.9: Mô hình mạng Zigbee

Trong khi kết nối Xbee với Arduino, phải sử dụng mạng điện trở hạn dòng giữa Tx của Arduino và Rx của Xbee vì chân Tx của Arduino xuất ra điện áp 5V trong khi đó chân Rx của Xbee chỉ chấp nhận mức điện áp từ 2.8V đến 3.3V. Điều này đảm bảo rằng mức điện áp tại chân Tx của Arduino có thể chấp nhận được khoảng 3,3V tại Rx của Xbee. Arduino coi mức 3,3v từ chân Tx của Xbee là mức cao vì vậy không cần điện trở giữa chân Tx của Xbee và Rx của Arduino.

Công thức tính điện áp vào chân Rx của Xbee như sau:

$$V_{RX} = \frac{R_2}{R_1 + R_2} V_O$$

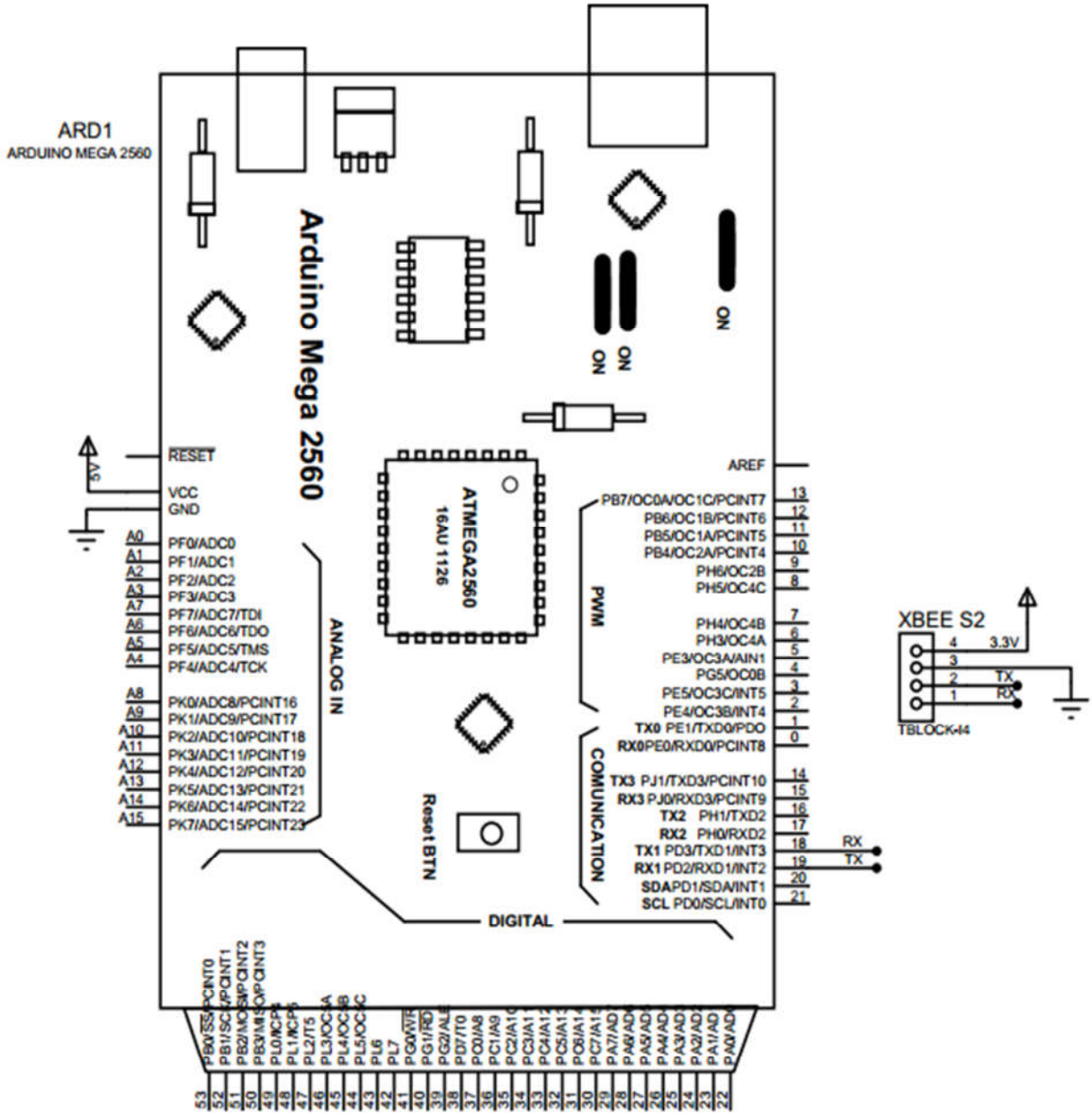
Trong đó:

R1, R2: giá trị điện trở cần tìm

V0: Điện áp ngõ ra của Arduino (5V)

V_{RX}: Điện áp vào chân Rx, theo Datasheet chọn mức điện áp vào chân Rx là V_{RX} = 3.3v

Chọn R1= 1KΩ → R2 = 2KΩ



Hình 3.10: Sơ đồ nguyên lý của khối thu phát tín hiệu Zigbee.

- **Giải thích sơ đồ nguyên lý:**

  Khi có lệnh điều khiển Arduino sẽ gửi tín hiệu từ chân Tx đến chân Rx của Xbee để phát tín hiệu đến trạm khác. Nếu nhận được tín hiệu từ trạm khác truyền tới thì Xbee sẽ gửi tín hiệu từ chân Tx đến chân Rx của Arduino.

d. Khối hiển thị

- **Chức năng:** Khối này có chức năng hiển thị thông tin lên màn hình HMI để người dùng có thể dễ dàng điều khiển và giám sát hệ thống.
- **Phân tích lựa chọn linh kiện:** Để đáp ứng được các chức năng trên nhóm đã chọn màn hình Nextion HMI làm linh kiện chính. Màn hình cảm ứng Nextion HMI, là một giao diện người và máy (HMI) cung cấp giao diện điều khiển và trực quan giữa con người và quy trình, máy, ứng dụng hoặc thiết bị.

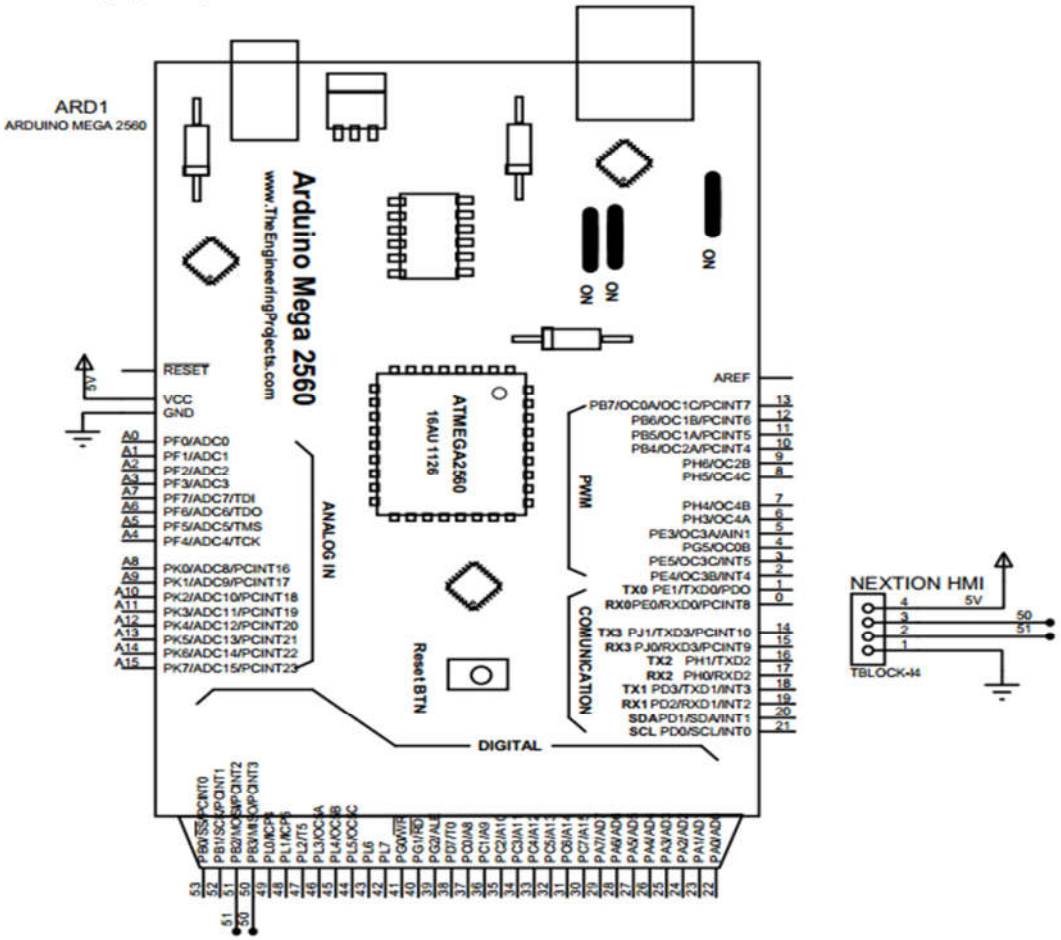
  Phần mềm thiết kế giao diện trên máy tính trực quan và dễ sử dụng, giao tiếp với màn hình qua giao tiếp UART Có bộ nhớ lưu trữ và xử lý hình ảnh, tích hợp khe thẻ nhớ, nên giảm thiểu được hầu hết các tác vụ về xử lý hình cho mạch điều khiển trung tâm, chỉ truyền về trung tâm các dữ liệu thao tác cảm ứng. Thiết kế cảm ứng điện trở giúp dễ dàng thao tác khi mang găng tay trong môi trường lao động.

Hình 3.11: Màn hình Nextion HMI 3.2 Inch

- **Các thông số kỹ thuật chính:**
  - Màn hình HMI 3.2 inch cảm ứng điện trở.
  - Độ phân giải: 400x200
  - Kích thước bảng: 95x47,6 mm
  - Giao tiếp UART mức TTL (3 – 5VDC).
  - Cấp nguồn 5VDC.
  - Vùng hiển thị: 69,60mm (L) × 41,76mm (W)

- Độ sáng có thể điều chỉnh: 0 ~ 230 nit, khoảng thời gian điều chỉnh là 1%
- Có phần mềm thiết kế giao diện đi kèm.
- Có bộ nhớ lưu trữ và xử lý hình ảnh.
- Giao tiếp UART, với chỉ 2 dây tín hiệu (TX, RX) rất dễ dàng giao tiếp và điều khiển.

➢ **Sơ đồ nguyên lý**



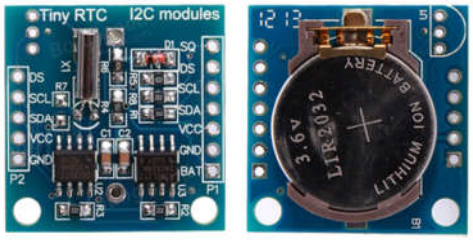
Hình 3.12: Sơ đồ nguyên lý của khối hiển thị

➢ **Giải thích sơ đồ nguyên lý:**
Khi có lệnh điều khiển từ người dùng, màn hình HMI sẽ gửi lệnh điều khiển từ chân Tx của HMI đến chân Rx của Arduino. Khi có sự thay đổi trạng thái của các thiết bị ngoại vi thì Arduino sẽ gửi dữ liệu từ chân Tx đến chân Rx của HMI để hiển thị ra màn hình.

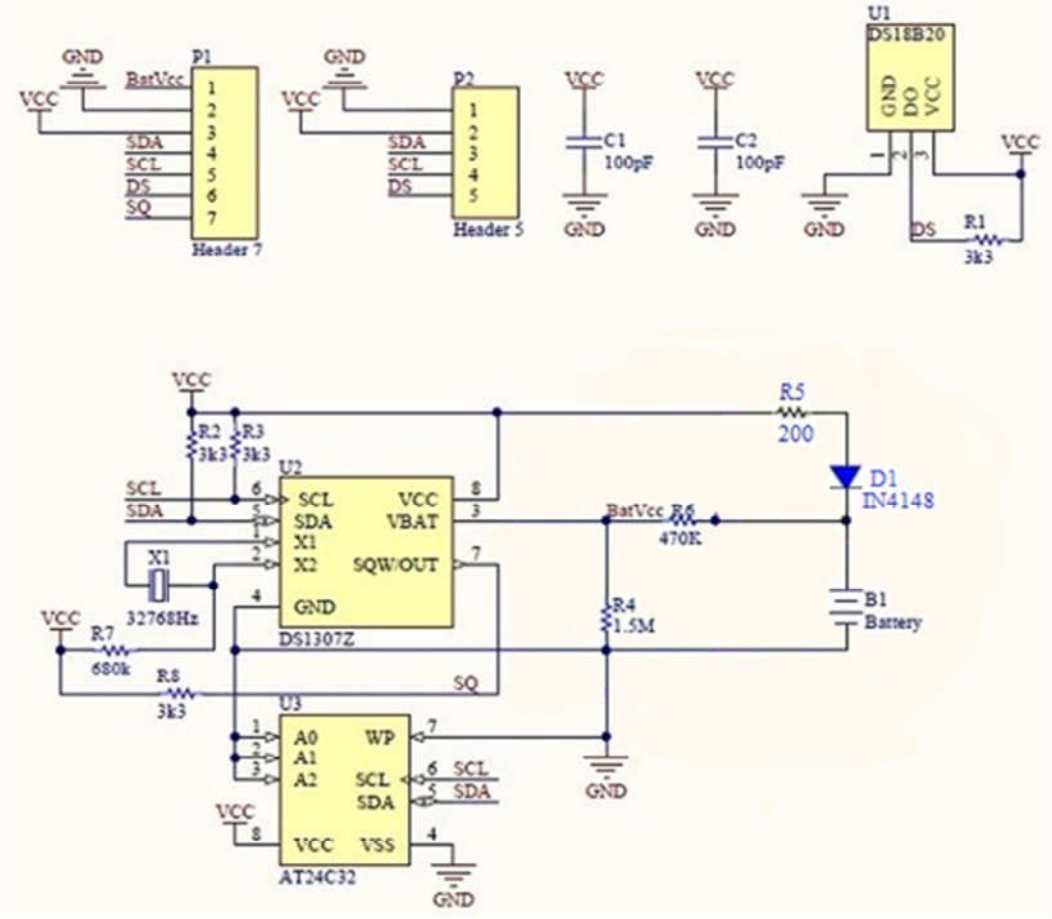
e. Khối thời gian thực

➢ **Chức năng:** Khối này có chức năng cung cấp thời gian hiện tại cho vi điều khiển để hiển thị ra màn hình.

- **Phân tích lựa chọn linh kiện:** Để đáp ứng được các chức năng trên nhóm đã chọn Mô-đun RTC DS1307 làm linh kiện chính. Mô-đun thời gian thực RTC DS1307 có chức năng lưu trữ thông tin ngày tháng năm cũng như giờ phút giây, nó sẽ hoạt động như một chiếc đồng hồ và có thể xuất dữ liệu ra ngoài qua giao thức I2C.Mô-đun thời gian thực RTC DS1307 được thiết kế kèm theo một viên pin đồng hồ có khả năng lưu trữ thông tin lên đến 10 năm mà không cần cấp nguồn 5V từ bên ngoài. Module đi kèm với EEPROM AT24C32 có khả năng lưu trữ thêm thông tin lên đến 32KBit.



Hình 3.13: Mô-đun RTC DS1307

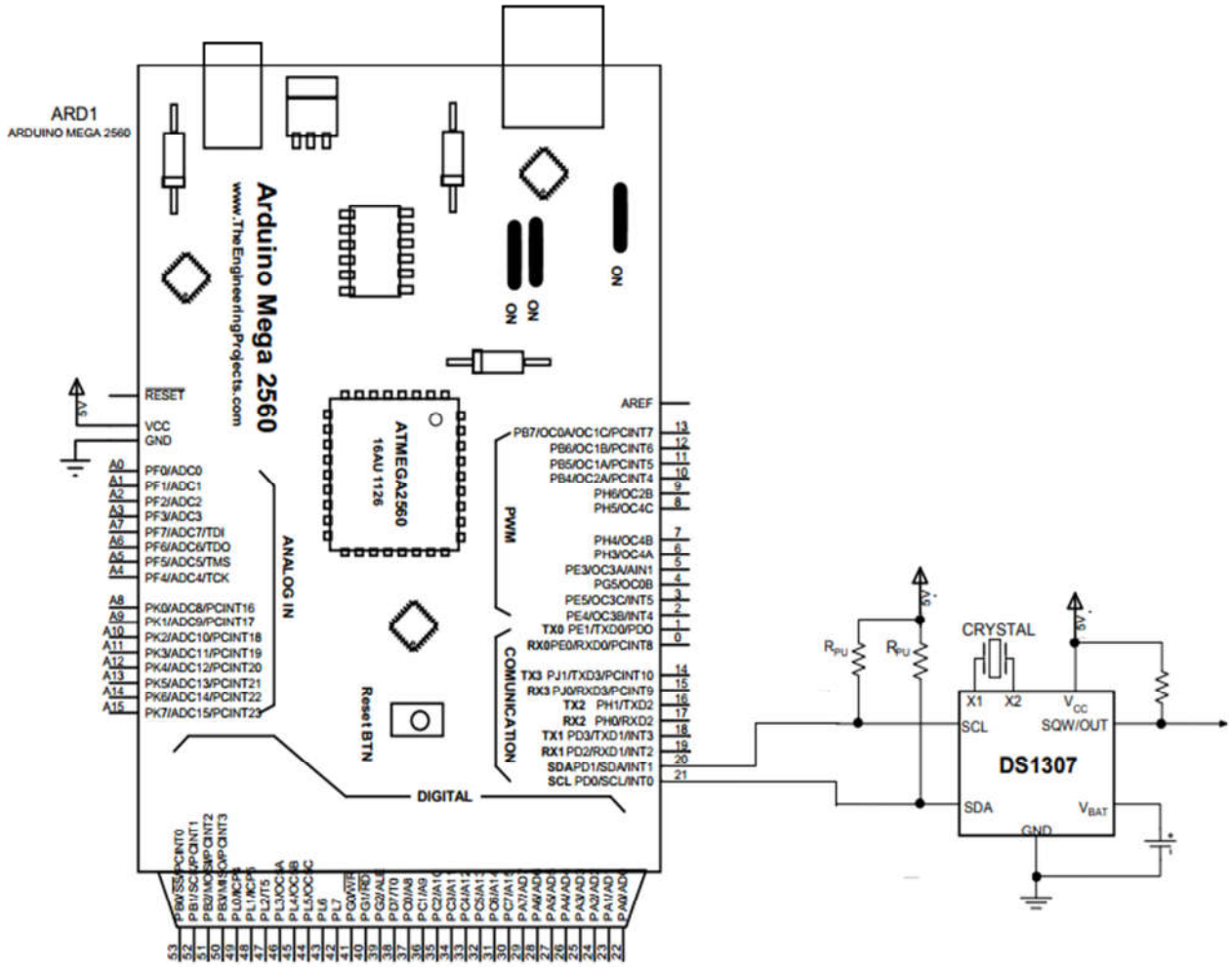


Hình 3.14: Sơ đồ nguyên lý RTC DS1307

- **Các thông số kĩ thuật chính:**

  - IC chính: RTC DS1307 + EEPROM AT24C32
  - Nguồn cung cấp: 5VDC.
  - Giao tiếp: I2C
  - Lưu trữ và cung cấp các thông tin thời gian thực: ngày, tháng, năm, giờ, phút, giây...
  - Pin duy trì thời gian trong trường hợp không cấp nguồn: Có
  - Tần số ngõ ra: 1Hz.
  - Kích thước: 27 x 28 x 8.4mm
  - PIN: 5-pin bao gồm giao thức I2C sẵn sàng giao tiếp INT (QWO), SCL, SDA, VCC và GND.

- **Sơ đồ nguyên lý:**



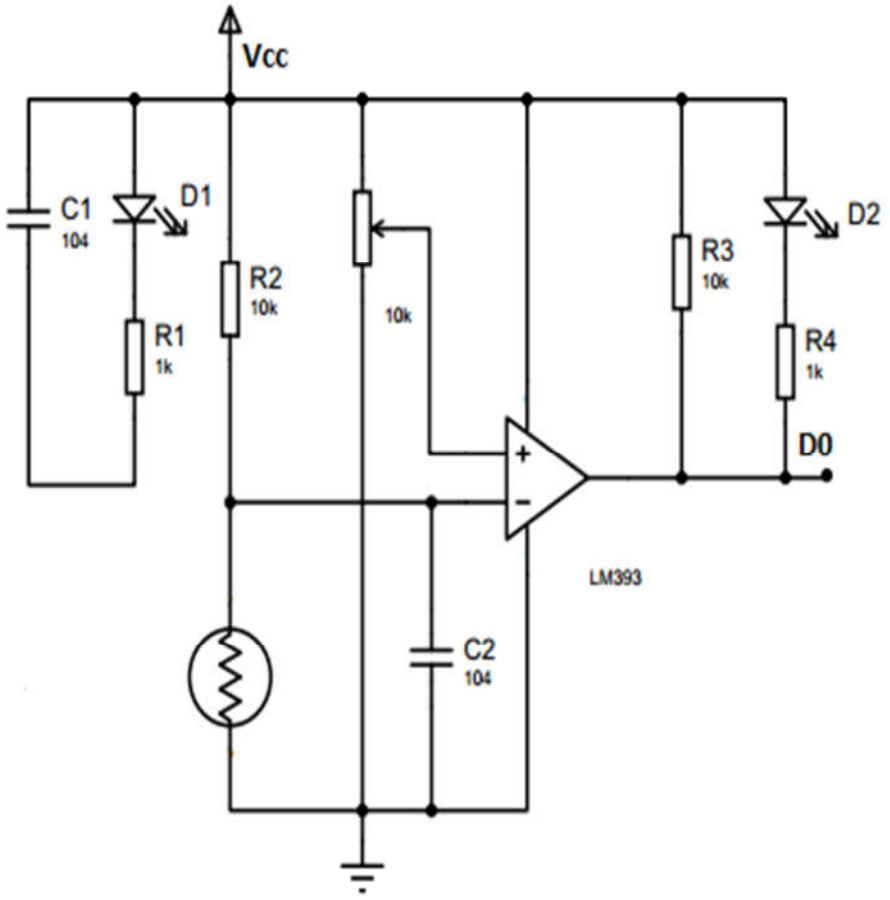
Hình 3.15: Sơ đồ nguyên lý khối thời gian thực

- **Giải thích sơ đồ nguyên lý:**

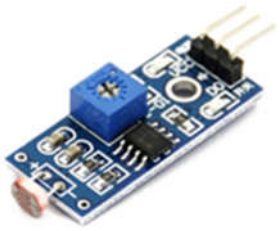
  Mô-đun RTC DS1307 sẽ cập nhật thời gian hiện tại và gửi dữ liệu về cho Arduino, sau đó Arduino sẽ lấy thời gian này hiển thị ra màn hình và so sánh với thời gian cài đặt để điều khiển đèn trong chế độ tự động.

f. Khối cảm biến:

- **Chức năng:** Khối này có chức năng phát hiện được đèn sáng hay đèn hỏng thông qua cảm biến quang sau đó gửi dữ liệu về khối điều khiển đèn.
- **Phân tích lựa chọn linh kiện:** Để đáp ứng được các chức năng trên nhóm đã chọn Mô-đun cảm biến ánh sáng quang trở CDS làm linh kiện. Cảm biến ánh sáng quang trở CDS có tích hợp sẵn Opamp và biến trở so sánh mức tín hiệu giúp cho việc nhận biết tín hiệu trở nên dễ dàng, sử dụng để nhận biết hay bật tắt thiết bị theo cường độ ánh sáng môi trường.



Hình 3.16: Sơ đồ nguyên lý Mô-đun cảm biến ánh sáng quang trở CDS

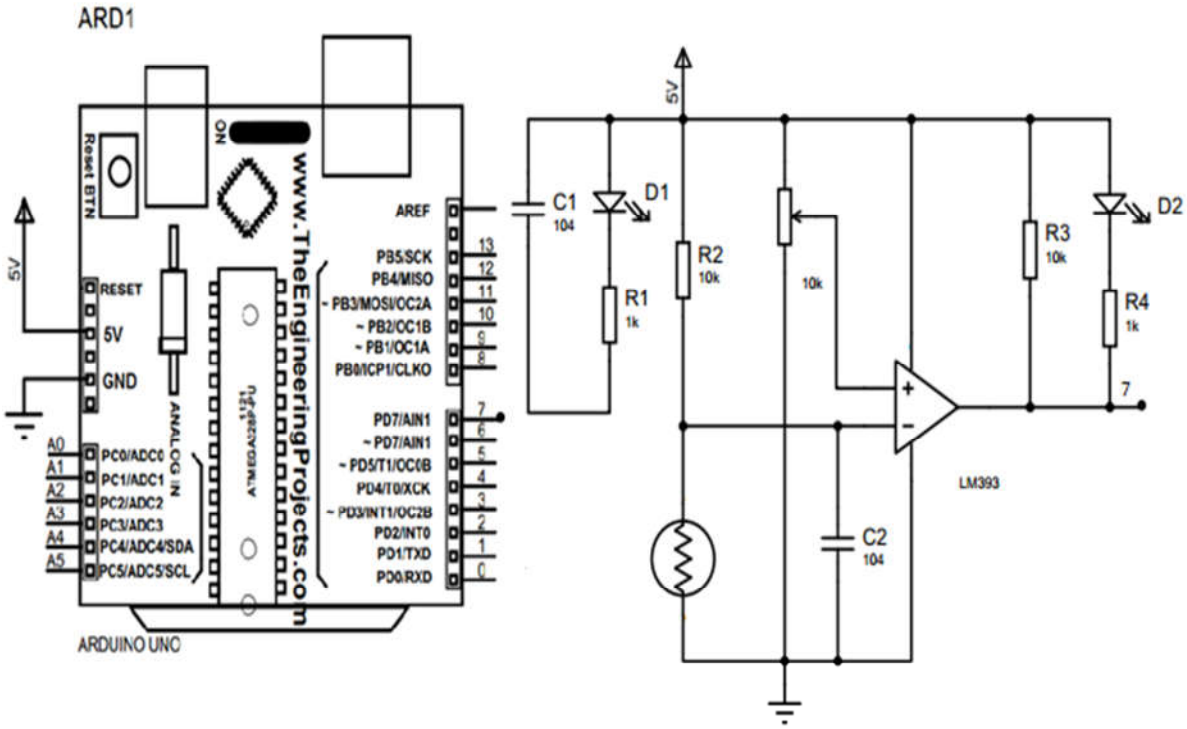


Hình 3.17: Mô-đun cảm biến ánh sáng quang trở CDS

- **Các thông số kĩ thuật chính:**

  - Điện áp đầu vào: 3.3V - 5V
  - Đầu ra: có đầu ra số và đầu ra tương tự tương ứng D0 và A0
  - Có chiết áp điều chỉnh cường độ sáng
  - Kích thước: 3.2cm x 1.4cm
  - Tải đầu ra số D0: 15mA
  - Đầu ra số D0 = 0 khi cường độ sáng cao và D0=1 khi cường độ sáng thấp.
  - Đầu ra tương tự A0 có thể kết nối với ADC để điều khiển thiết bị ở cường độ sáng mong muốn

- **Sơ đồ nguyên lý:**



Hình 3.18: Sơ đồ nguyên lý khối cảm biến

- **Giải thích sơ đồ nguyên lý:**

Khi đèn sáng sẽ có ánh sáng chiếu đến quang trở của Mô-đun cảm biến ánh sáng quang trở CDS lúc này Mô-đun sẽ xuất tín hiệu mức 0 đến chân số 7 của Arduino. Ngược khi đèn tắt thì Mô-đun sẽ xuất tín hiệu mức 1, vì vậy mà có thể biết được đèn sáng bình thường hay đèn hỏng.

g. Khối nguồn

- **Chức năng:** Khối này là khối không thể thiếu trong mạch điện tử, nó có chức năng cung cấp điện áp cho mạch hoạt động. Trong đề tài nhóm sử dụng hai khối nguồn để cấp cho mạch. Khối nguồn cấp nguồn cho Arduino và khối nguồn 220AC để cấp cho đèn.
- **Phân tích lựa chọn linh kiện**

Đề tài bao gồm 4 trạm thu phát nên sẽ sử dụng 4 nguồn riêng để cấp cho mạch. Một nguồn DC để cấp cho vi điều khiển và một nguồn 220AC để cấp cho đèn.

Tổng dòng tiêu thụ của từng mạch được thể hiện trong bảng sau:

- **Mạch điều khiển đèn**

Bảng 3.3: Tổng dòng tiêu thụ của mạch điều khiển đèn

STT	Tên linh kiện	Điện áp (VDC)	Dòng tiêu thụ (mA)	Số lượng	Tổng dòng điện tiêu thụ (mA)
1	Arduino UNO R3	5	30	1	30
2	Xbee S2	3.3	2	1	2
3	Cảm biến ánh sáng quang trở CDS	5	15	1	15
4	MOC3020	1.5	20	1	20
5	LED đơn	3	10	3	30
Tổng dòng tiêu thụ(mA)			~97		

- **Mạch điều khiển trung tâm**

Bảng 3.4: Tổng dòng tiêu thụ của mạch điều khiển trung tâm

STT	Tên linh kiện	Điện áp (VDC)	Dòng tiêu thụ (mA)	Số lượng	Tổng dòng điện tiêu thụ (mA)
1	Arduino Mega 2560	5	30	1	30
2	Xbee S2	3.3	2	1	2
3	Màn hình Nextion HMI 3.2 Inch	5	20	1	20
4	Mô-đun RTC DS1307	5	2	1	2
5	LED đơn	3	10	1	10
Tổng dòng tiêu thụ(mA)			~64		

Từ hai bảng thống kê trên, nhóm đã chọn nguồn Adapter 5V – 1A để cấp cho mạch. Nguồn có thiết kế nhỏ gọn, độ bền cao, dòng ra ổn định.

Hình 3.19: Nguồn adapter 5V – 1A

- **Các thông số kỹ thuật chính:**
  - Điện áp vào: 100~240AC,50/60 Hz
  - Điện áp ngõ ra: 5VDC
  - Dòng điện ngõ ra tối đa: 1A
  - Kiểu Jack ngõ ra: Chuẩn Jack DC tròn đường kính ngoài 5.5mm, đường kính trong phù hợp với lỗ kim từ 2.1~2.5mm.

CHƯƠNG 4: THI CÔNG HỆ THỐNG

4.1 THI CÔNG HỆ THỐNG

4.1.1 Thi công mạch thu phát tín hiệu Zigbee.

4.1.1.1 Cấu hình Xbee S2

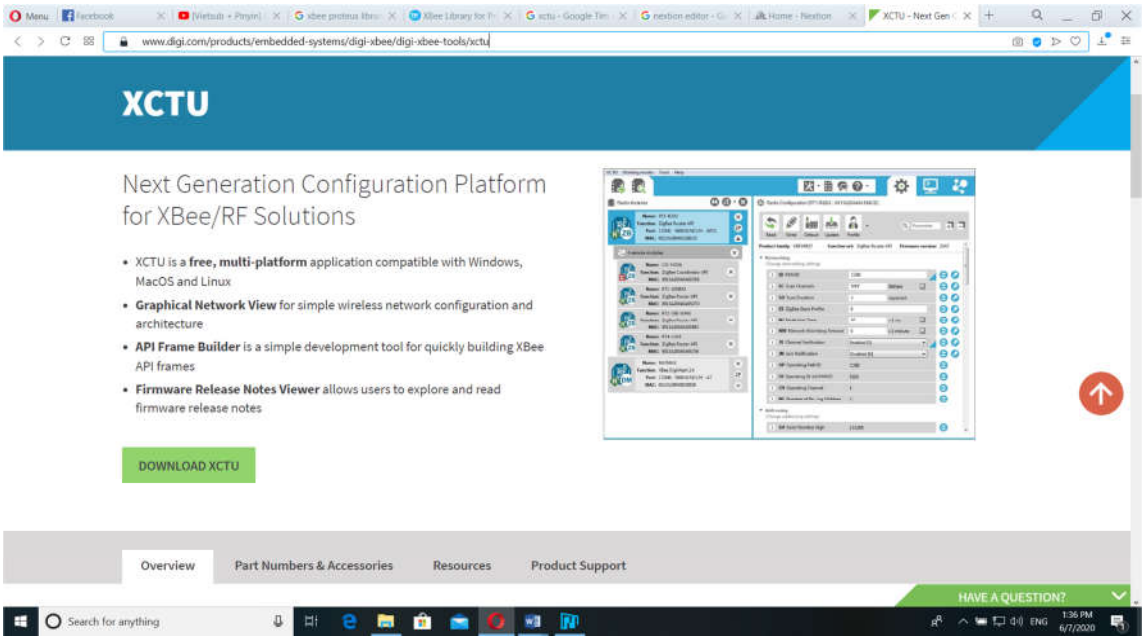
- **Giới thiệu phần mềm XCTU**

XCTU là một ứng dụng miễn phí được thiết kế để người dùng có thể tương tác với các mô-đun RF của Digi Inter thông qua một giao diện đơn giản, dễ sử dụng.

Hình 4.1: Icon XCTU khi đã cài đặt

- **Hướng dẫn cơ bản sử dụng phần mềm XCTU**

Truy cập trang chủ của Digi theo địa chỉ: https://www.digi.com/products/embedded-systems/digi-xbee/digi-xbee-tools/xctu



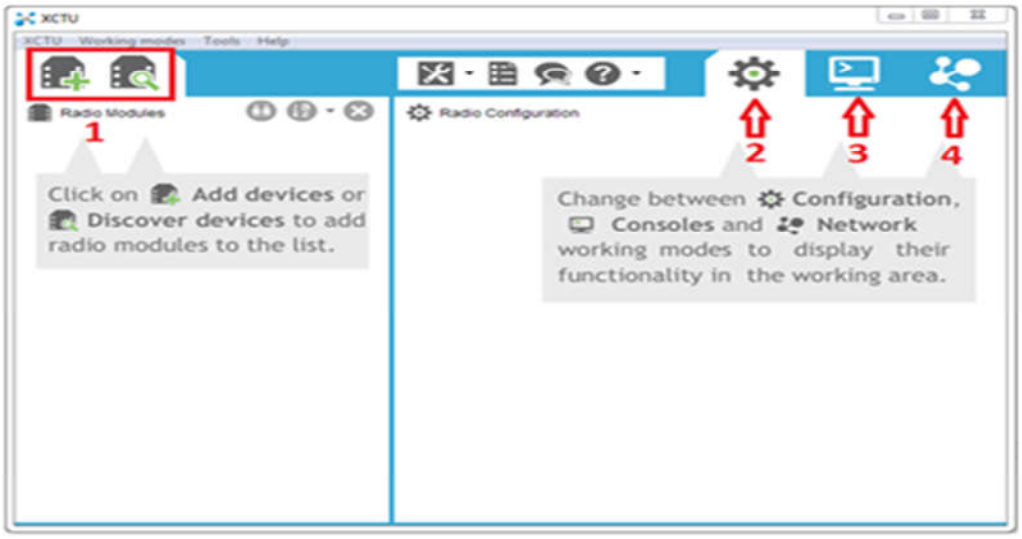
Hình 4.2: Giao diện website digi.com

Nhấn chọn *"Download XTCU"* lựa chọn phiên bản phù hợp với hệ điều hành của máy tính cá nhân.

Sau khi tải file cài đặt, tiến hành cài đặt như các phần mềm thông thường.

- **Hướng dẫn sử dụng cơ bản**

Bước 1: Khởi động phần mềm XTCU



Hình 4.3: Cửa sổ làm việc chính

Nhìn vào hình 4.3:

(1): Tìm kiếm các mô-đun Xbee được kết nối với máy tính

(2): Cấu hình cho các mô-đun Xbee

(3): Tiến hành việc truyền dữ liệu giữa các mô-đun

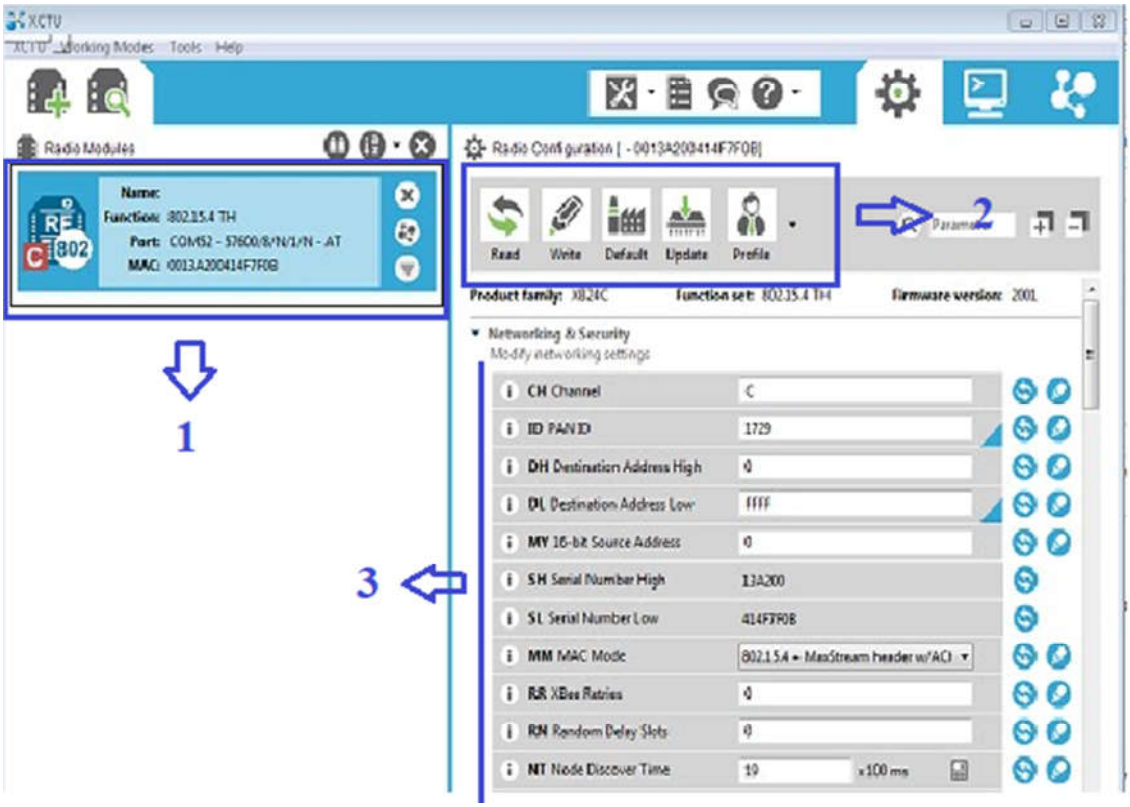
(4): Kiểm tra kết nối giữa các mô-đun

Bước 2: Để có thể cấu hình được, mô-đun Xbee cần kết nối với máy tính cá nhân thông qua đế mạch thu phát RF Xbee.

Hình 4.4: Đế mạch thu phát RF Xbee

Bước 3: Thiết lập module Xbee được kết nối

- **Cấu hình mô-đun Xbee**



Hình 4.5: Thiết lập cấu hình mô-đun Xbee

Nhìn vào hình 4.5:

(1): Thông tin cơ bản của mô-đun Xbee

- **Name**: Tên của mô-đun do người dùng đặt.
- **Function**: Phần mềm của Xbee được định dạng theo chuẩn nào.
- **Port**: cổng kết nối hiện tại của mô-đun với máy tính.
- **MAC**: địa chỉ của mô-đun được quy định bởi nhà sản xuất, mỗi mô-đun có một địa chỉ và là duy nhất (địa chỉ cũng được in ở mặt sau của mô-đun).

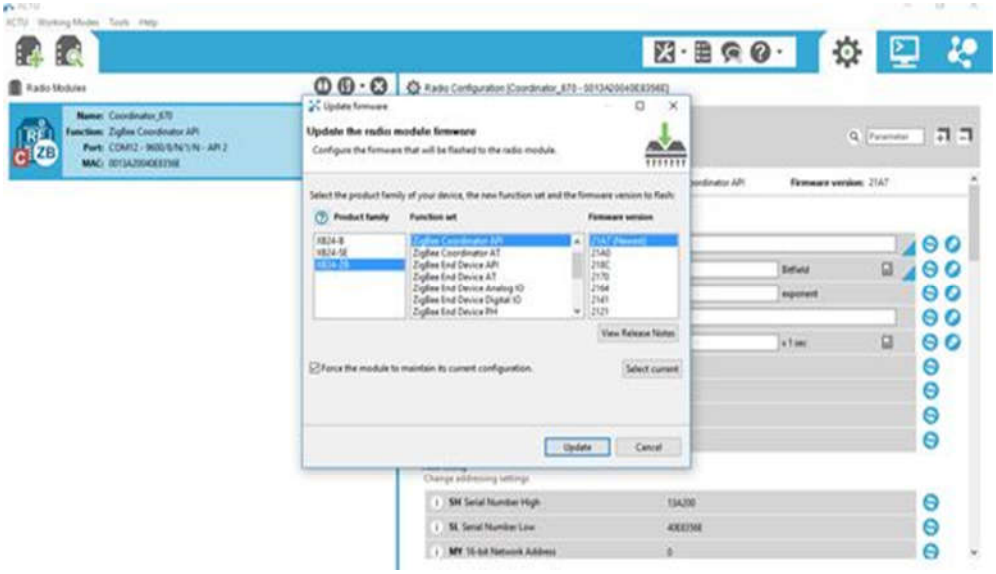
(2): Các công cụ để thao tác trên mô-đun Xbee

- **Read**: Đọc cấu hình đang có của Xbee
- **Write**: Lưu những thay đổi sau khi cấu hình
- **Default**: Cài đặt cấu hình Xbee về mặc định
- **Update**: Thay đổi phần mềm của Xbee
- **Profile**: Lưu cấu hình vào máy tính (định dạng .xml), hoặc mở một cấu hình đã được lưu.

(3): Nơi cấu hình cho Xbee tùy theo mong muốn của người dùng.

➢ **Thiết lập các chức năng cho mô-đun**

- Chọn **Update** trên công cụ để cập nhật bản mới nhất.
- Chọn **XB24-ZB**.
- Chọn **ZigBee Coodinator API**.
- Bảng chọn **21A7 (Newest*)*** *(bản mới nhất)*.



Hình 4.6: Thiết lập phần mềm chức năng cho module

Sau khi đã chọn, nhấn **Update** để cập nhật Firmware mới cho Xbee.

❖ **Cấu hình Coordinator API**

❖ **Phần Networking**

- **ID** Pan ID: 7777. Được xem như "mật khẩu" của một mạng Zigbee, chỉ những mô-đun có cùng Pan ID mới có thể giao tiếp được với nhau.
- **SC** Scan Channels: 1FFE.

❖ **Phần Adressing**

- **DH** Destination Address High: 0.
- **DL** Destination Address Low: FFFF.
- **NI** Node Identifier: Coordinator. Đặt tên cho Xbee chủ.

❖ **Phần Serial Interfacing**

- **AP** API Enable**:** 2
- **AO** API Output Mode: Native [0]

  Các thông số còn lại để mặc định. Chọn ***Write*** để cập nhật các thông số vừa thiết lập.

❖ **Cấu hình Router API**

❖ **Phần Networking**

- **ID** Pan ID: 7777. Được xem như "mật khẩu" của một mạng Zigbee, chỉ những mô-đun có cùng Pan ID mới có thể giao tiếp được với nhau.
- **SC** Scan Channels: 1FFE

❖ **Phần Addressing**

- **DH** Destination Address High: 0
- **DL** Destination Address Low: 0
- **NI** Node Identifier: API_1. Đặt tên cho mô-đun

❖ **Phần Serial Interfacing**

- **AP** API Enable**:** 2
- **AO** API Output Mode: Native [0]

Các thông số còn lại để mặc định. Chọn ***Write*** trên thanh công cụ để cập nhật các thông số vừa thiết lập cho Xbee

❖ **Cấu hình Coordinator AT**

❖ **Phần Networking**

- **ID** Pan ID: 8888: Được xem như "mật khẩu" của một mạng Zigbee, chỉ những mô-đun có cùng Pan ID mới có thể giao tiếp được với nhau
- **SC** Scan Channels: 1FFE
- **CE** Coordinator Enable: 1

❖ **Phần Addressing**

- **DH** Destination Address High: 0
- **DL** Destination Address Low: FFFF
- **NI** Node Identifier: AT_1. Đặt tên cho mô-đun

❖ **Phần Serial Interfacing**

- **AP** API Enable**:** 0

Các thông số còn lại để mặc định. Chọn ***Write*** trên thanh công cụ để cập nhật các thông số cho Xbee.

❖ **Cấu hình Router AT**

❖ **Phần Networking**

- **ID** Pan ID: 8888: Được xem như "mật khẩu" của một mạng Zigbee, chỉ những mô-đun có cùng Pan ID mới có thể giao tiếp được với nhau

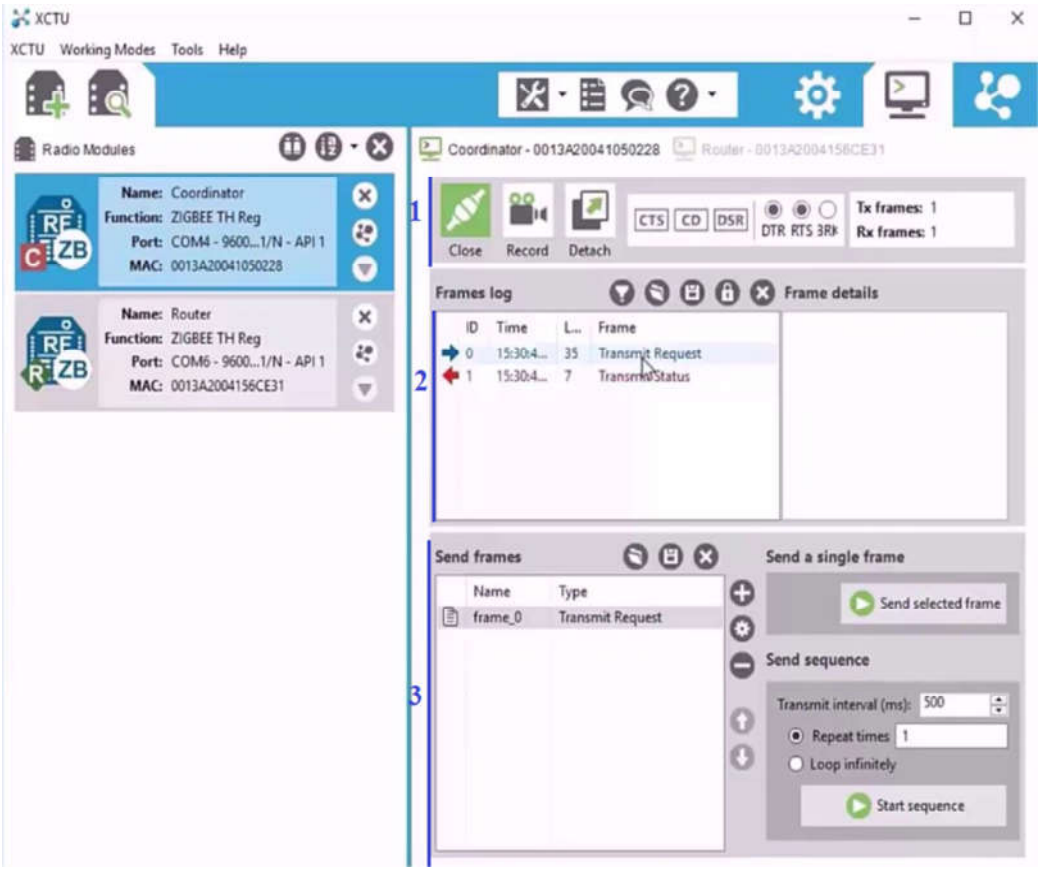
- **SC** Scan Channels: 1FFE
- **JV** Channel Verification: 1
- **CE** Coordinator Enable: 0

❖ **Phần Addressing**

- **DH** Destination Address High: 0
- **DL** Destination Address Low: FFFF
- **NI** Node Identifier: AT_1. Đặt tên cho mô-đun

Các thông số còn lại để mặc định. Chọn ***Write*** trên thanh công cụ để cập nhật các thông số cho Xbee.

❖ **Giao tiếp giữa các mô-đun**



Hình 4.7: Cửa sổ giao tiếp

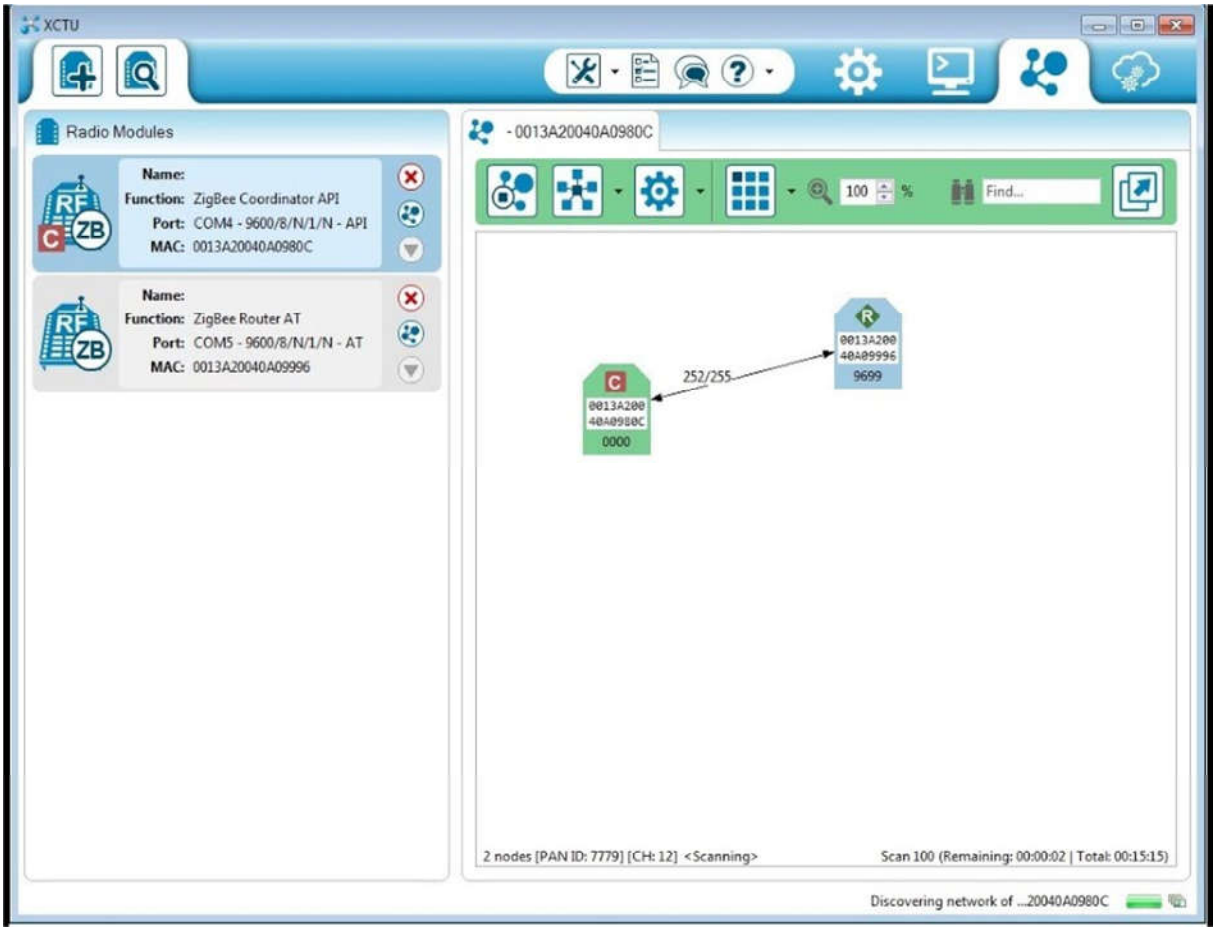
Nhìn vào hình 4.7:

Phần (1): Kết nối/ngắt kết nối Xbee

Phần (2): Khu vực hiển thị gói dữ liệu nhận về.

Phần (3): Khu vực gửi gói dữ liệu.

❖ **Kiểm tra kết nối các Xbee trong mạng**



Hình 4.8: Cửa sổ kiểm tra kết nối mạng

Chọn **Scan**, phần mềm sẽ tự động quét tìm các module trong mạng đang kết nối với nhau.

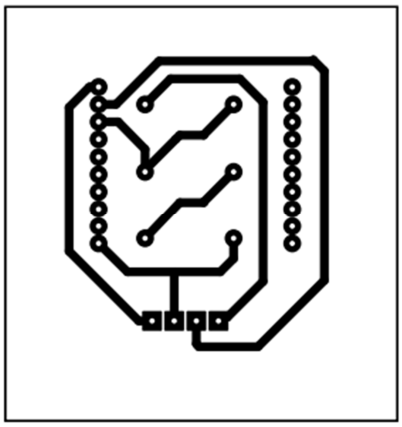
4.1.1.2 Thi công mạch

Trước khi vẽ mạch nguyên lý và PCB cho toàn mạch, ta nên lập danh sách các linh kiện sử dụng. Dưới đây là bảng tóm tắt tất cả các linh kiện sử dụng trong đề tài.

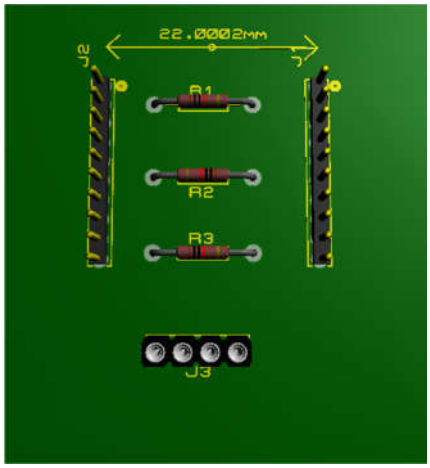
Bảng 4.1 Danh sách linh kiện của mạch thu phát tín hiệu Zigbee

STT	Tên linh kiện	Thông số
1	Xbee S2	V_{in} = 3.3V
2	Điện trở	1KΩ

Sơ đồ bố trí linh kiện lớp trên và lớp dưới của mạch:



Hình 4.9: Mặt dưới PCB mạch thu phát tín hiệu Zigbee



Hình 4.10: Mặt trên PCB mạch thu phát tín hiệu Zigbee

4.1.2 Thiết kế thi công giao diện điều khiển trên HMI bằng Nextion Editor

4.1.2.1 Giới thiệu và cài đặt phần mềm Nextion Editor

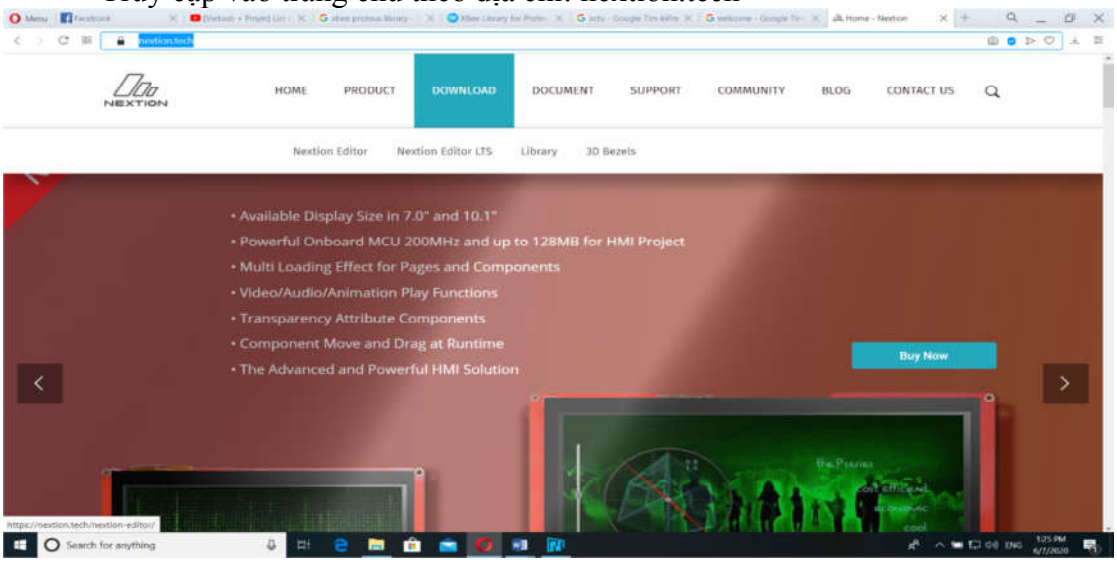
❖ **Giới thiệu phần mềm Nextion Editor**

- Nextion Editor là 1 phần mềm miễn phí để có thể tạo giao diện trên màn hình cảm ứng HMI.

Hình 4.11: Icon Nextion Editor sau khi cài đặt

❖ **Cài đặt phần mềm Nextion Editor**

- Truy cập vào trang chủ theo địa chỉ: nextion.tech



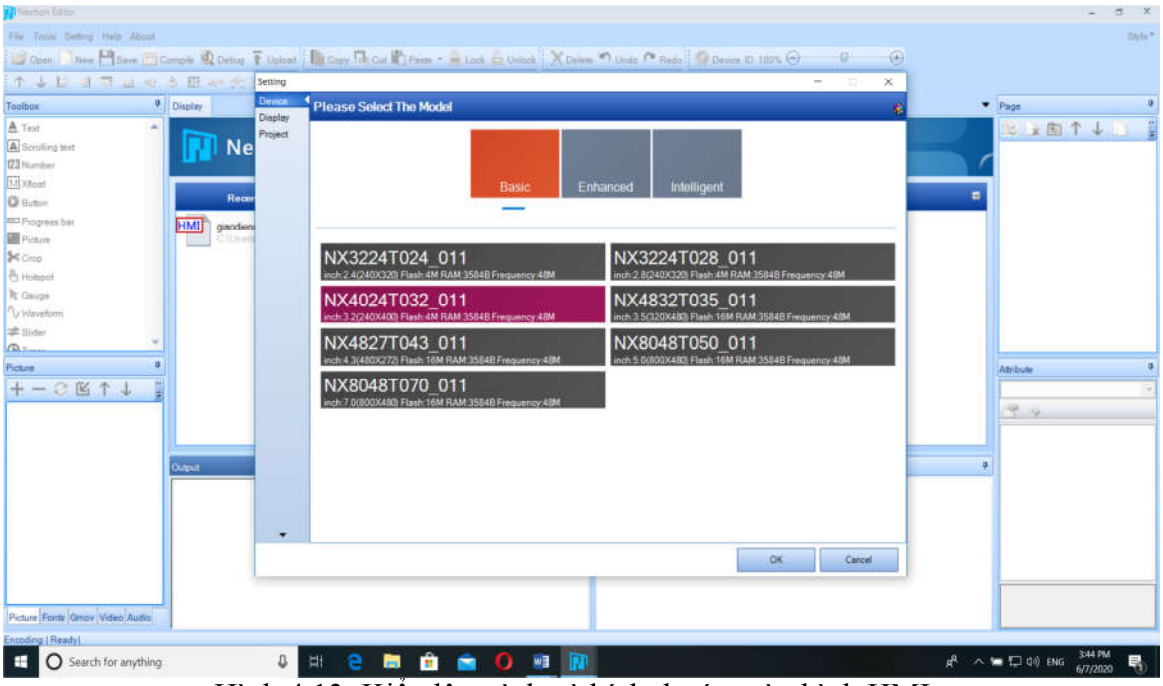
Hình 4.12: Trang chủ Nextion Editor

- Nhấn ***"Download"*** chọn ***"Nextion Editor"*** ⇒ tải về máy theo file Zip hoặc Exe tùy chọn.
- Sau khi tải file cài đặt, tiến hành cài đặt như các phần mềm thông thường.

4.1.2.2 Thiết kế giao diện HMI

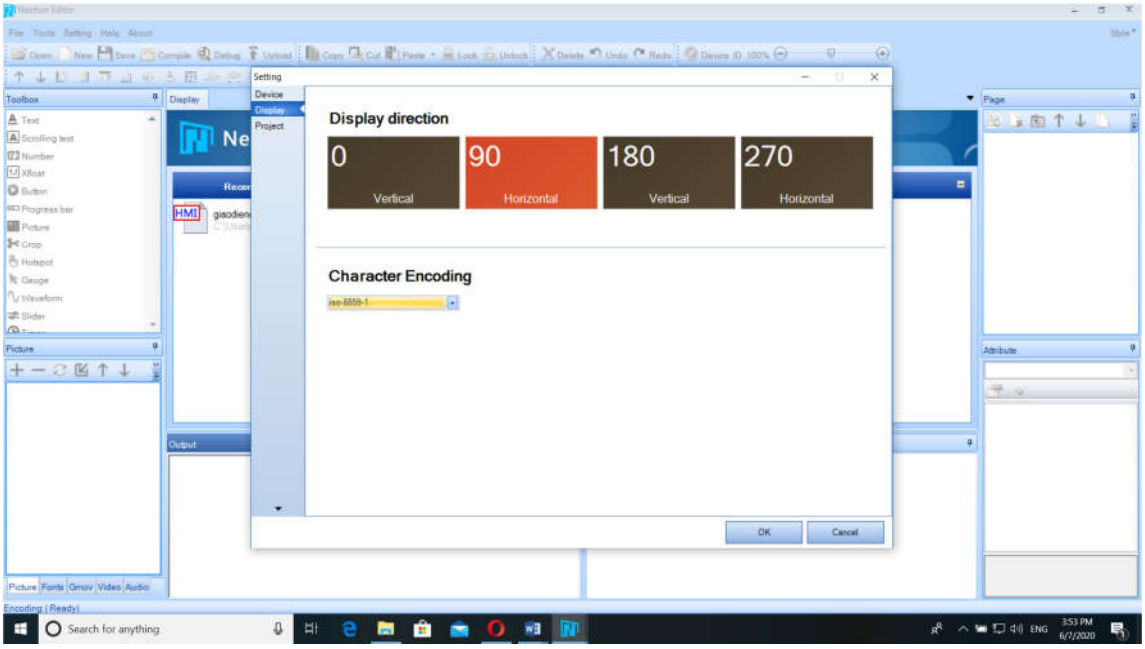
❖ **Tạo File phù hợp với thiết bị**

Mở phần mềm và chọn đúng tên màn hình đang sử dụng. Nhóm sử dụng màn hình 3.2 inch và kiểu là basic.



Hình 4.13: Kiểu lập trình và kích thước màn hình HMI

Tiếp theo là chọn thiết kế giao diện theo chiều ngang hoặc chiều dọc.



Hình 4.14: Chọn chiều thiết kế giao diện

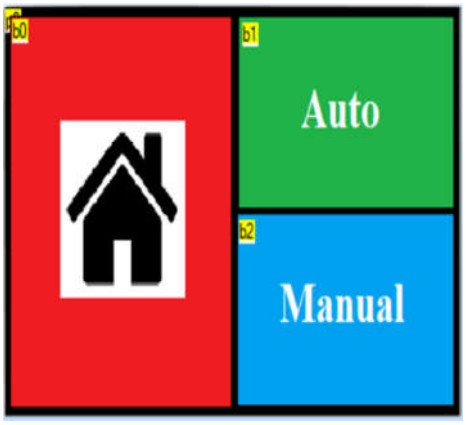
❖ **Thiết kế giao diện**

Nhóm thiết kế gồm 6 trang giao diện:

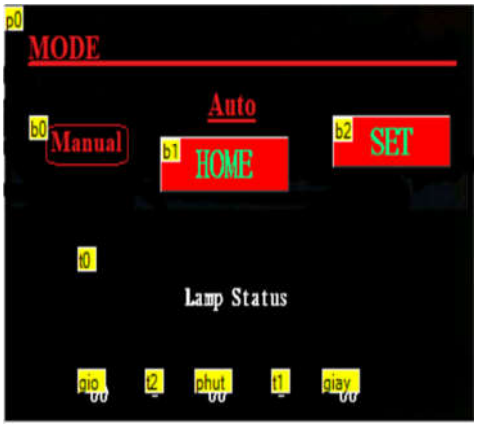
- Page 0: Hiển thị tên sinh viên thực hiện.
- Page 1: Nhập mật khẩu.
- Page 2: Chọn chế độ
- Page 3: Chế độ Auto
- Page 4: Chế độ Manuel
- Page 5: Cài đặt thời gian

Hình 4.15: Page0

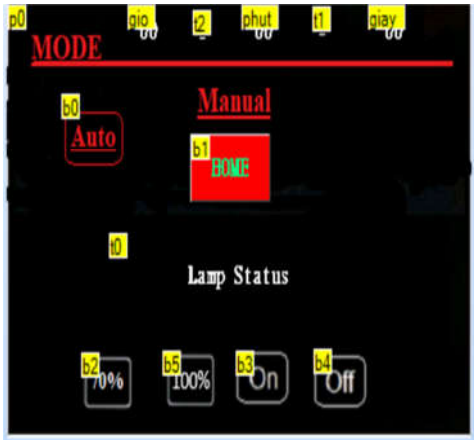
Hình 4.16: Page1



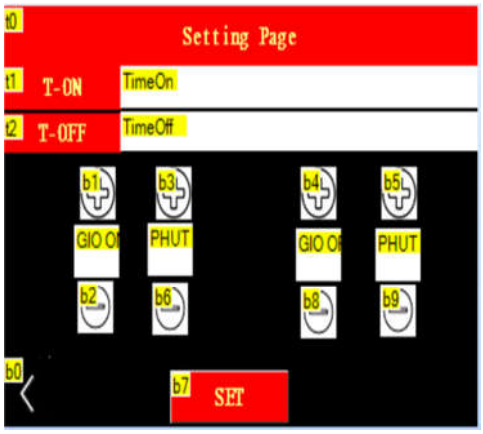
Hình 4.17: Page2



Hình 4.18: Page3



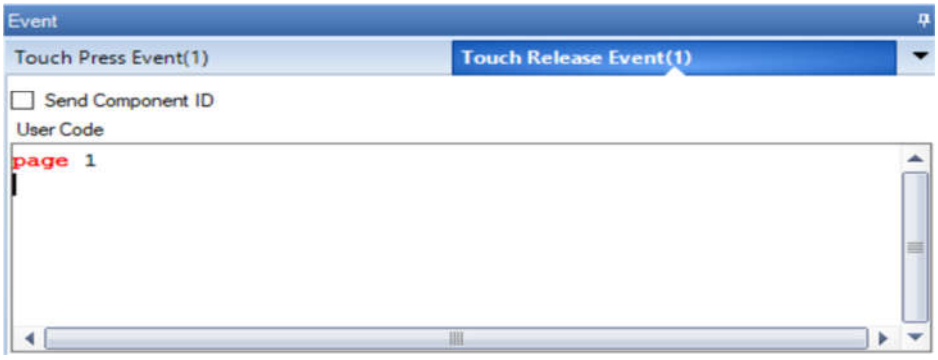
Hình 4.19: Page4



Hình 4.20: Page5

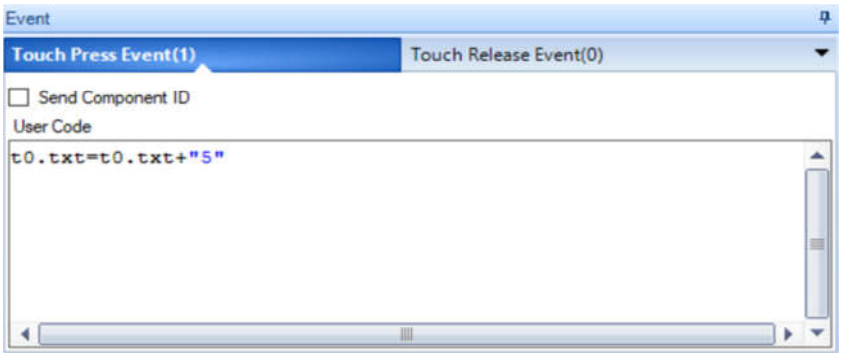
➢ Cách thực hiện các Page như sau:

- Ở Page 0: sẽ có một nút nhấn chuyển qua page2 (b0) và hình nền (p0). Cách lập trình nút nhấn:



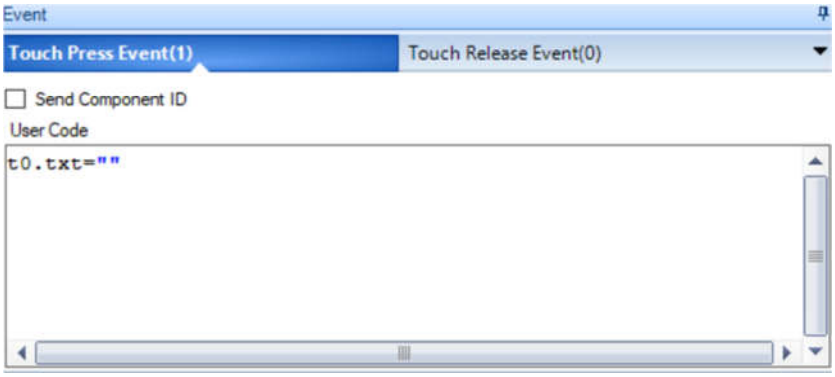
Hình 4.21: Lập trình nút chuyển page

- Ở Page 1: sẽ có các nút nhập từ 1 tới 9, 1 textbox hiển thị số nhập , 1 nút xóa toàn bộ dữ liệu nhập, 1 nút xóa 1 số vừa nhập, và nút chuyển trang nếu password đúng. Nút nhấn nhập số, số bao nhiêu thì sẽ cộng cho bấy nhiêu.

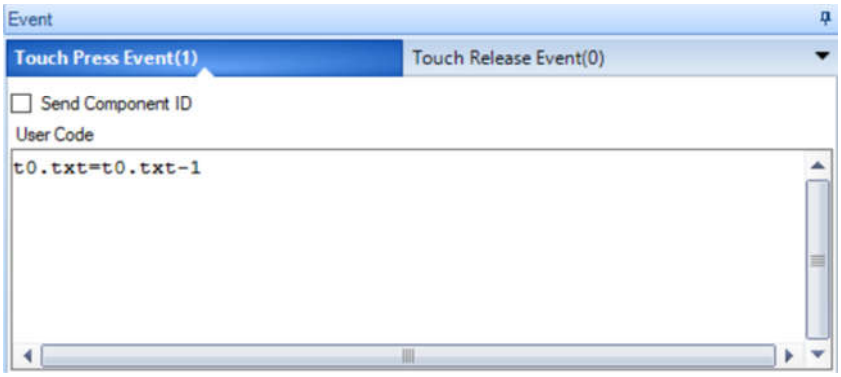


Hình 4.22: Nhập số 5

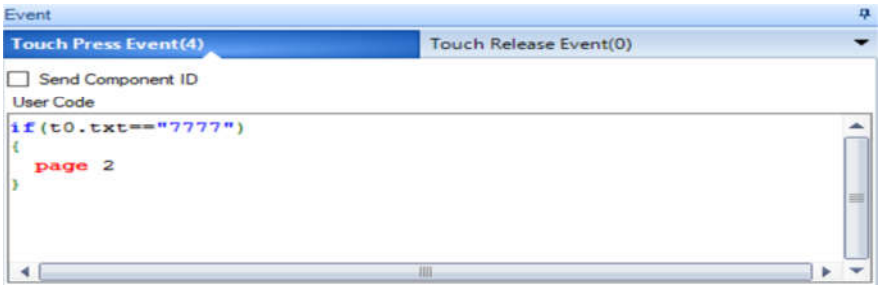
Tiếp theo là nút chuyển trang nếu mật khẩu đúng, xóa 1 ký tự và nút xóa hết.



Hình 4.23: Nút xóa dữ liệu nhập



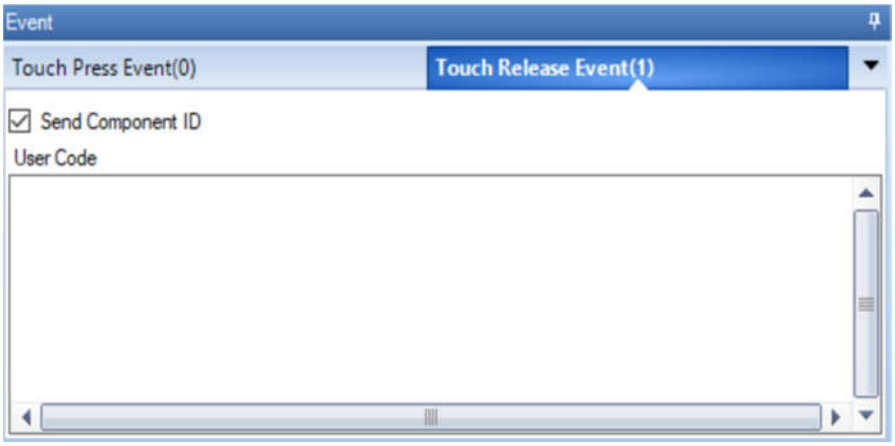
Hình 4.24: Nút xóa 1 dữ liệu nhập



Hình 4.25: Nút chuyển trang

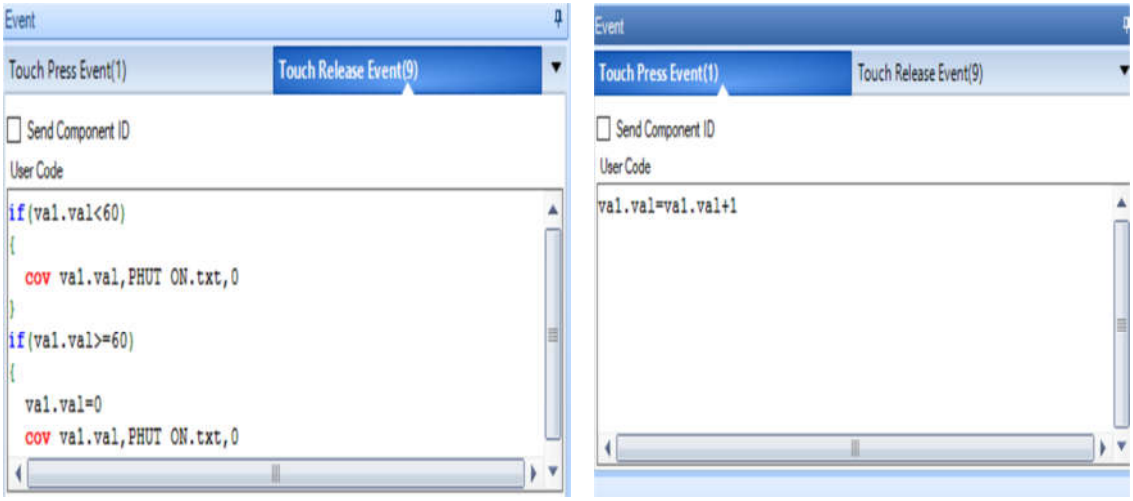
- Page 2: sẽ có tổng cộng 3 nút nhấn. Bao gồm: 1 nút chuyển về trang chủ (home), 1 nút chế độ auto, 1 nút chế độ manuel chỉ cần lập trình số page cần tới.

- Page 3: sẽ có các nút nhấn ON, OFF, 70%, 100%, hiển thị thời gian, trạng thái đèn. Các nút nhấn chỉ cần gửi ID của nút nhấn về cho Arduino để nhận biết lập trình.



Hình 4.26: Lập trình cho các nút nhấn

- Page 4: Tương tự page 3 chỉ có thêm 1 nút SET để chuyển tới page 5.
- Page 5: Bao gồm các nút tăng giảm để thiết lập thời gian sáng tắt của đèn. Và hiển thị trong textbox TimeOn, TimeOff. Cách lập trình cho nút tăng phút tương tự như vậy cho nút giảm giờ hoặc phút.

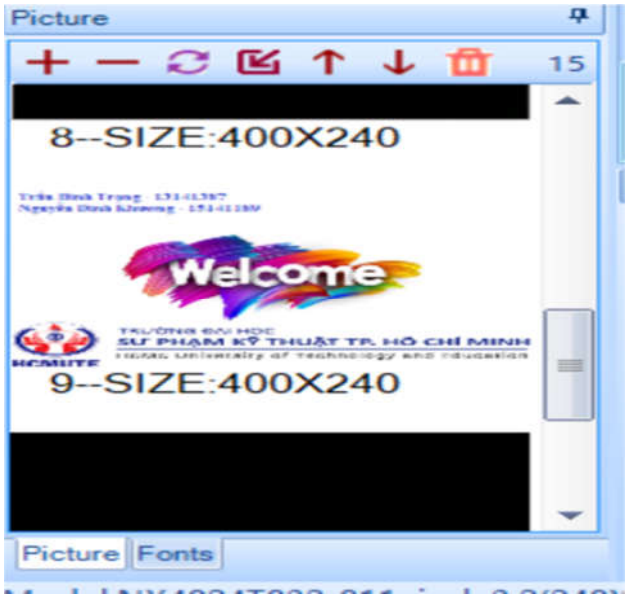


Hình 4.27: Thiết lập tăng giảm thời gian

Để có thể sử dụng hình ảnh làm hình nền hoặc nút nhấn thì ta phải lưu trữ hình ảnh vào thư viện ảnh của file giao diện đã tạo.

➢ Cách lưu hình ảnh vào thư viện ảnh của file nextion:

- Ta nhấn vào nút dấu "+" ở góc trái màn hình.
- Sau đó lựa chọn hình ảnh mà ta cần sử dụng ở trong máy tính.
- Nhấn **Open** là hình ảnh sẽ được đưa vào thư viện ảnh.



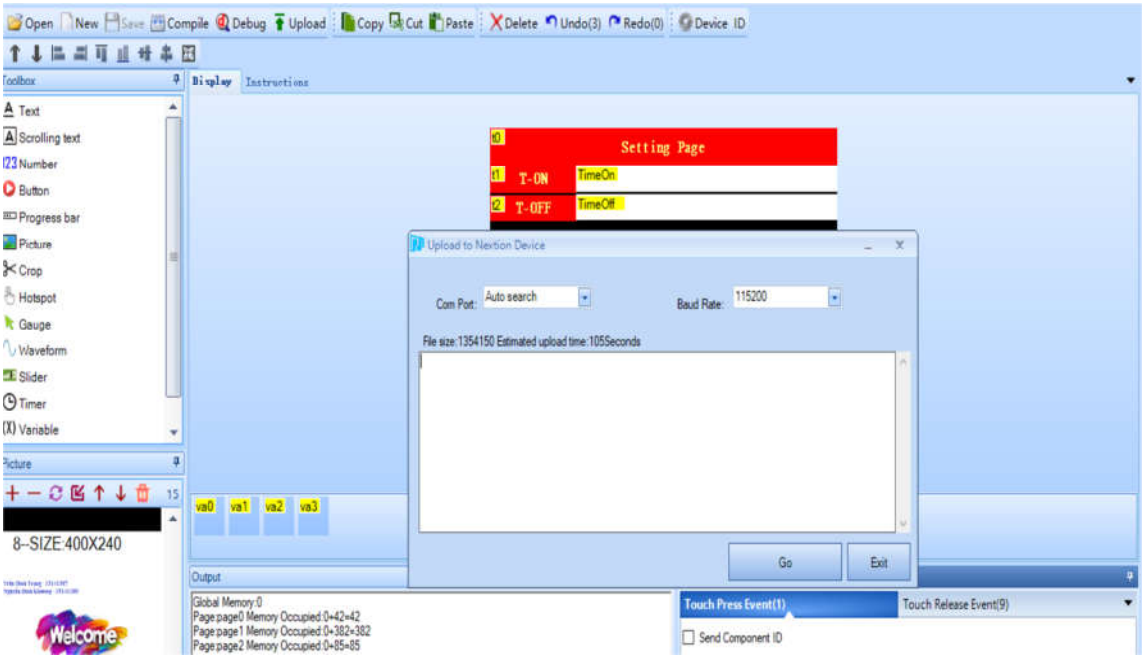
Hình 4.28: Thư viện ảnh

Lưu ý: Hình ảnh phải có kích thước 400x240 trở xuống mới có thể sử dụng được vì màn hình chỉ có 3.2 inch.

❖ **Nạp chương trình vào màn hình Nextion HMI**

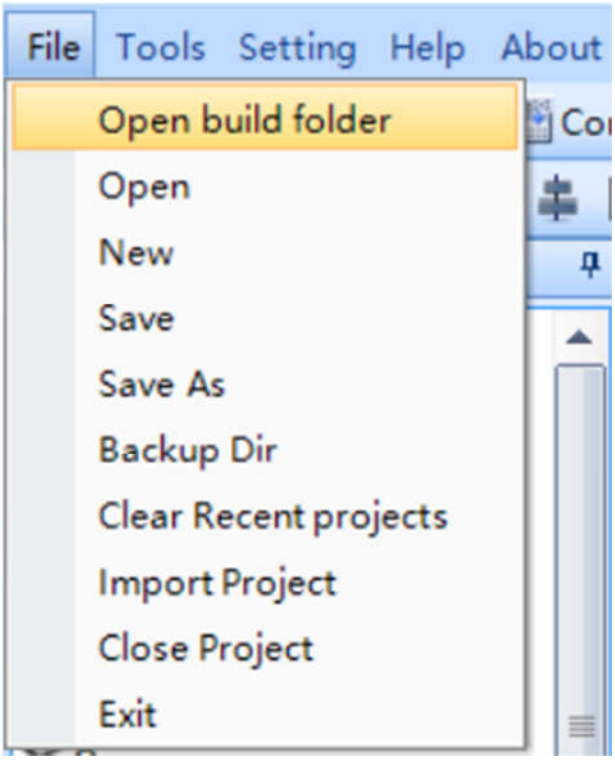
➢ Có 2 cách để nạp chương trình là:

- Nạp trực tiếp qua phần mềm Nextion Editor. Kết nối HMI vào máy tính và mở phần mềm lên nhấn vào **Upload** sau đó hiện ra bảng upload nhấn **Go** phần mềm sẽ tự động cập nhật vào HMI.



Hình 4.29: Nạp chương trình HMI

- Cách thứ 2 là gián tiếp thông qua thẻ nhớ SD. Mở file lưu đuôi .txt và copy vào thẻ nhớ. Sau đó cắm thẻ nhớ vào màn hình HMI sẽ tự động cập nhật sau khi hoàn thành rút thẻ nhớ ra thì chương trình đã được nạp.



Hình 4.30: File giao diện HMI .txt

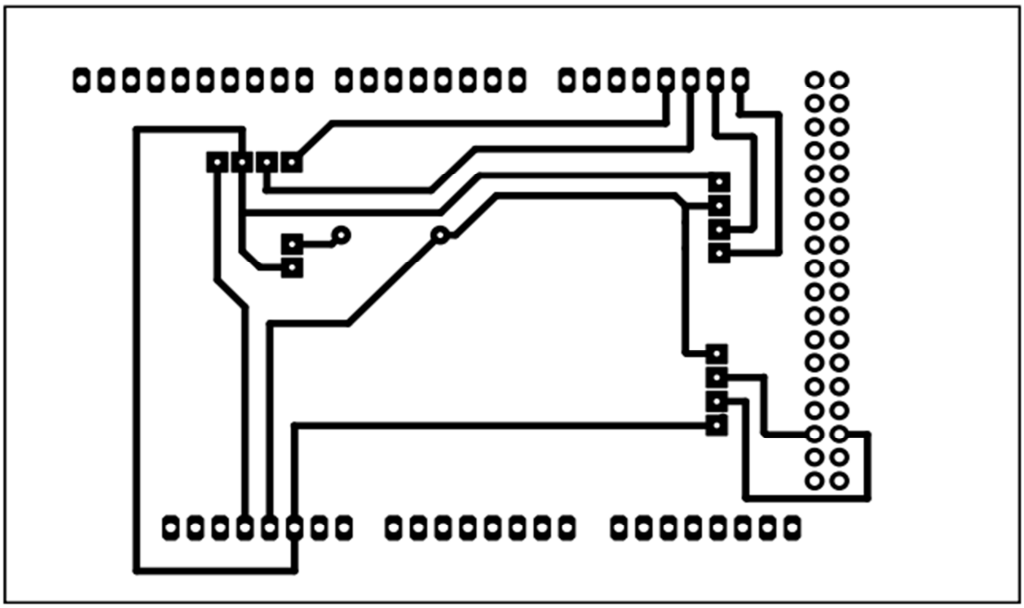
4.1.3 Thi công mạch điều khiển trung tâm

Trước khi vẽ mạch nguyên lý và PCB cho toàn mạch, ta nên lập danh sách các linh kiện sử dụng. Dưới đây là bảng tóm tắt tất cả các linh kiện sử dụng trong đề tài.

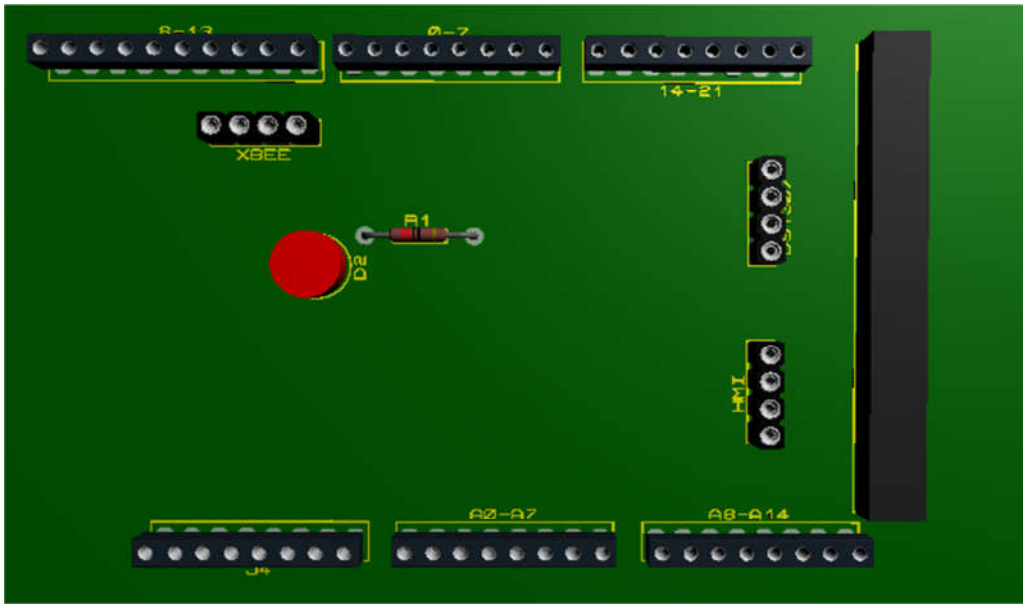
Bảng 4.2: Danh sách linh kiện của mạch điều khiển trung tâm

STT	Tên linh kiện	Thông số
1	Arduino Mega 2560	5 - 12V DC
2	Màn hình Nextion HMI	5V, 3.2 Inch
3	RTC DS1307	5V
4	Xbee S2	3.3V
5	Điện trở	220Ω
6	Led	3V, Đỏ

Sơ đồ bố trí linh kiện của mạch:



Hình 4.31: Mặt dưới PCB mạch điều khiển trung tâm



Hình 4.32: Mặt trên PCB mạch điều khiển trung tâm

4.1.4 Thi công mạch điều khiển phụ

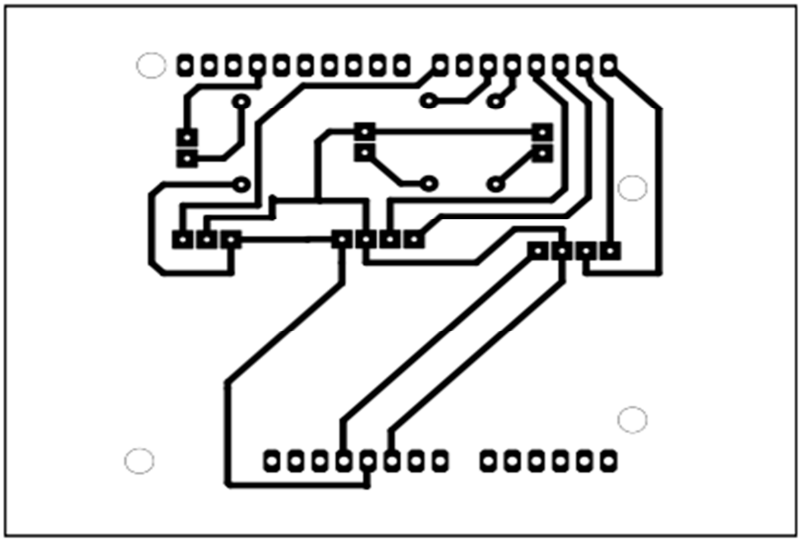
Về cơ bản các mạch điều khiển phụ đều giống nhau nên nhóm sẽ trình bày cách thi công một mạch.

Trước khi vẽ mạch nguyên lý và PCB cho toàn mạch, ta nên lập danh sách các linh kiện sử dụng. Dưới đây là bảng tóm tắt tất cả các linh kiện sử dụng trong đề tài.

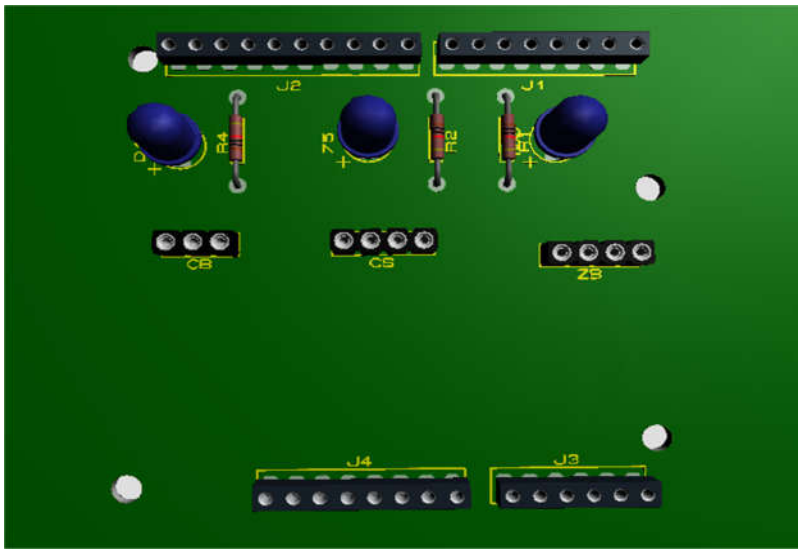
Bảng 4.3: Danh sách linh kiện của mạch điều khiển phụ

STT	Tên linh kiện	Thông số
1	Arduino Uno R3	5 - 12V DC
2	Cảm biến ánh sáng quang trở CDS	5V
3	Xbee S2	3.3V
4	Điện trở	1KΩ, 220Ω, 470Ω
5	Led	3V, Đỏ, Trắng

Sơ đồ bố trí linh kiện của mạch điều khiển phụ:



Hình 4.33: Mặt dưới PCB mạch Shield Arduino Uno R3



Hình 4.34: Mặt trên PCB mạch Shield Arduino Uno R3

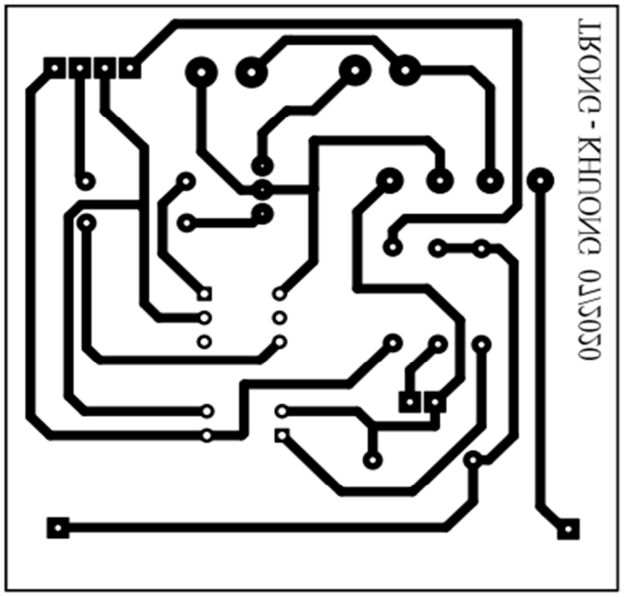
4.1.5 Thi công mạch phát hiện điểm 0 và mạch công suất

Sơ đồ bố trí linh kiện lớp trên và lớp dưới của mạch công suất:

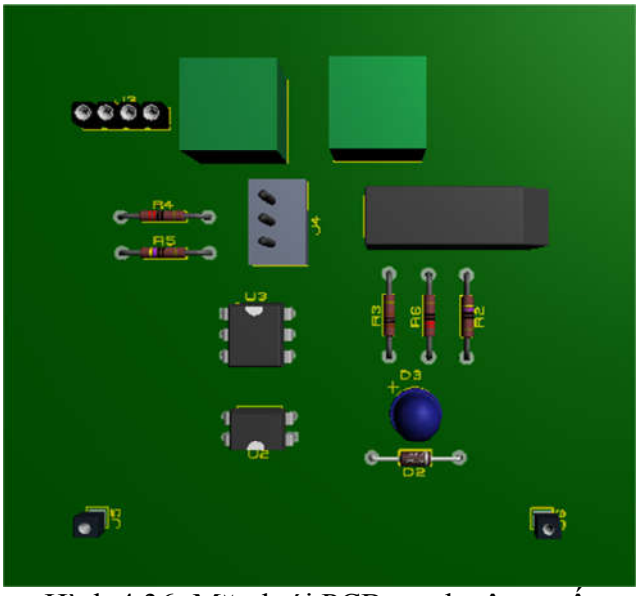
Bảng 4.4: Danh sách linh kiện của mạch phát hiện điểm 0 và mạch công suất

STT	Tên linh kiện	Thông số
1	Diode cầu KBP 307	V_{RMS} = 600V, I_F = 3A
2	Điện trở công suất	10K-10W
3	MOC3020	Ngõ ra: 400V, 100mA
4	PC817	V_R= 6V,I_F 50mA, P=70mW
5	Triac BTA16	V_{DRM}= 600V, I_T= 16A, I_{GT}=100mA
6	Diode Zener	5V, 1W
7	Điện trở	1KΩ, 220Ω, 470Ω
8	Đèn sợi đốt	220AC, 40W
9	Led	3V, Đỏ, Trắng

Sơ đồ bố trí linh kiện của mạch phát hiện điểm 0 và mạch công suất:



Hình 4.35: Mặt trên PCB mạch công suất

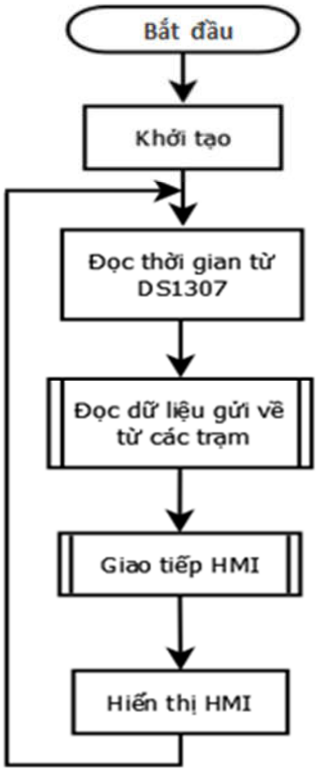


Hình 4.36: Mặt dưới PCB mạch công suất

4.2 Lưu đồ giải thuật

4.2.1 Lưu đồ mạch điều khiển trung tâm

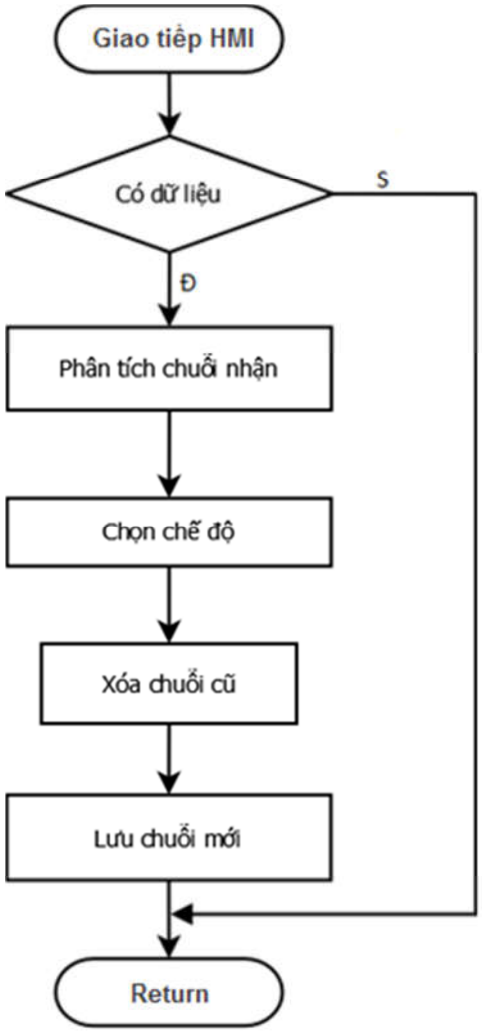
Yêu cầu chương trình: Đọc trạng thái đèn được gửi về từ các trạm đồng thời đọc thời gian thực để hiển thị lên màn hình HMI. Lập trình điều khiển chế độ chạy tự động và chế độ chạy bằng tay để điều khiển độ sáng đèn.



Hình 4.37: Lưu đồ chương trình chính của mạch điều khiển trung tâm

Giải thích lưu đồ:

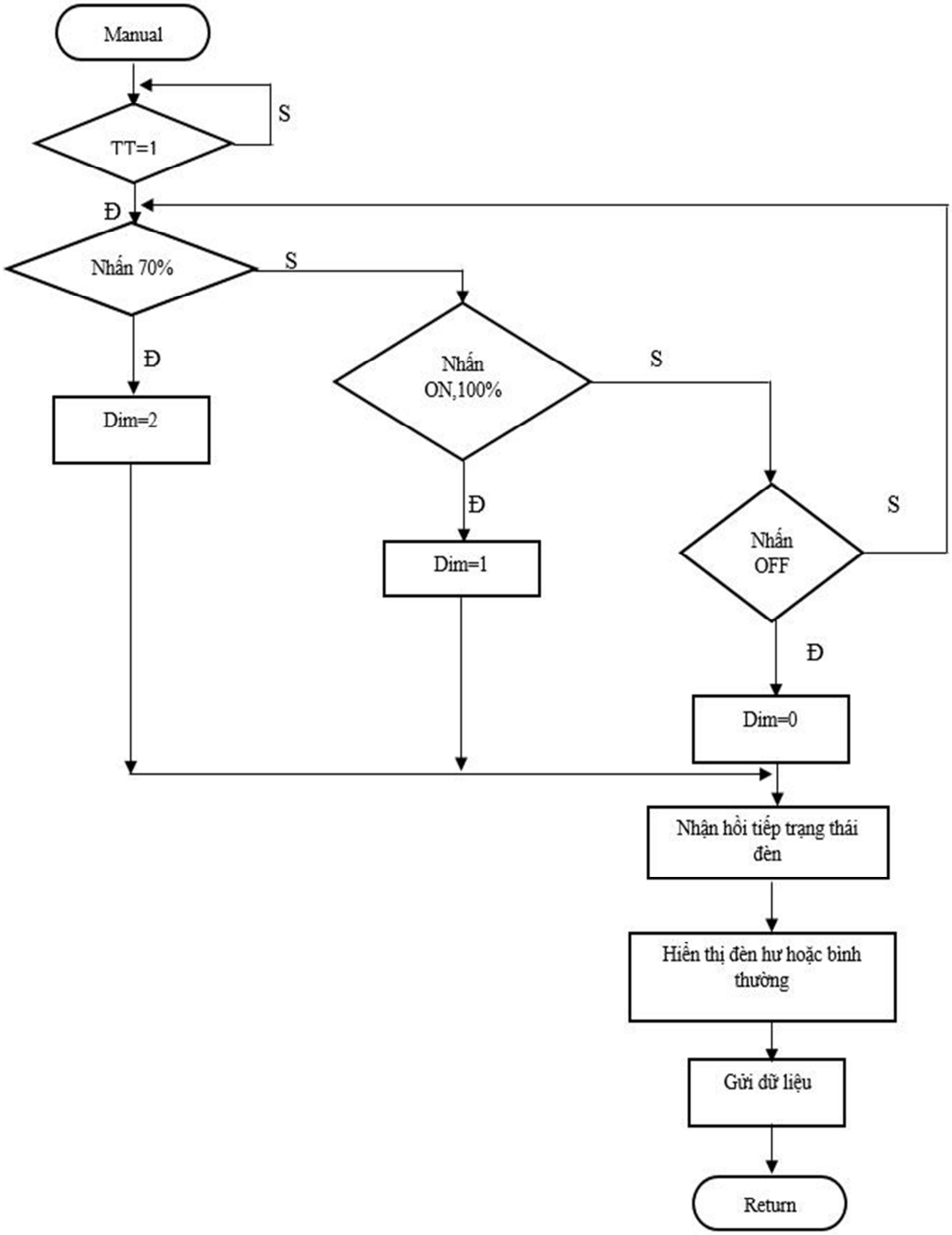
Để chương trình chạy, đầu tiên phải cấu hình hệ thống bằng cách khởi tạo Xbee, khởi tạo timer, khởi tạo giao tiếp Software Serial, khai báo địa chỉ truyền, khai báo địa chỉ nhận cũng như các thư viện của chúng. Tiếp đó vòng lặp được thực hiện để đảm bảo chương trình luôn được tuần hoàn, chương trình sẽ đọc dữ liệu theo trình tự đọc thời gian thực của DS1307, đọc trạng thái đèn gửi về từ các trạm, đọc lệnh điều khiển từ màn hình. Sau đó chương trình sẽ gửi dữ liệu ra màn hình để hiển thị.



Hình 4.38: Lưu đồ chương trình giao tiếp HMI

Giải thích lưu đồ:

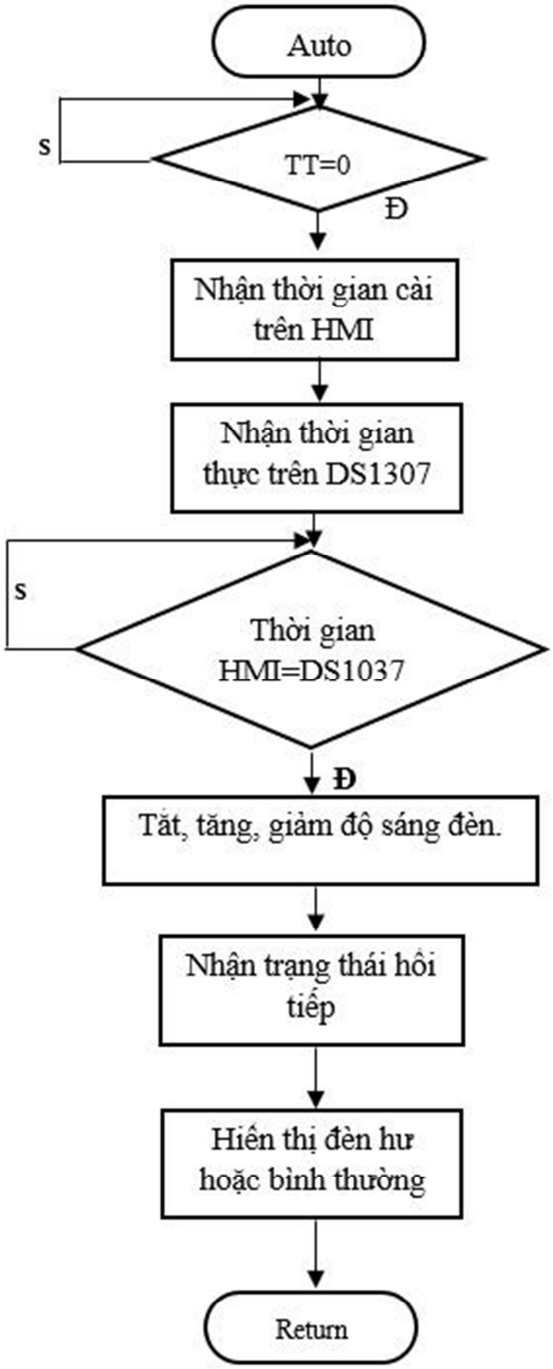
Nếu có dữ liệu điều khiển từ màn hình HMI chương trình sẽ phân tích chuỗi dữ liệu nhận được để chọn chế độ điều khiển và tiến hành xóa chuỗi dữ liệu cũ và thay bằng chuỗi dữ liệu mới được nhận. Sau đó chương trình sẽ gửi dữ liệu ra màn hình để hiển thị.



Hình 4.39: Lưu đồ chế độ chỉnh tay

Giải thích lưu đồ:

Đầu tiên kiểm tra biến trạng thái của chương trình. Nếu đúng tiếp tục kiểm tra lệnh từ màn hình có nhấn hay không, nếu có thì kiểm tra xem nhấn mức 100%, mức 70% hay nhấn OFF. Sau đó gán giá trị cho biến Dim và gửi dữ liệu đi điều khiển, nếu không thì nhận giá trị hồi tiếp trạng thái đèn và hiển thị ra màn hình.



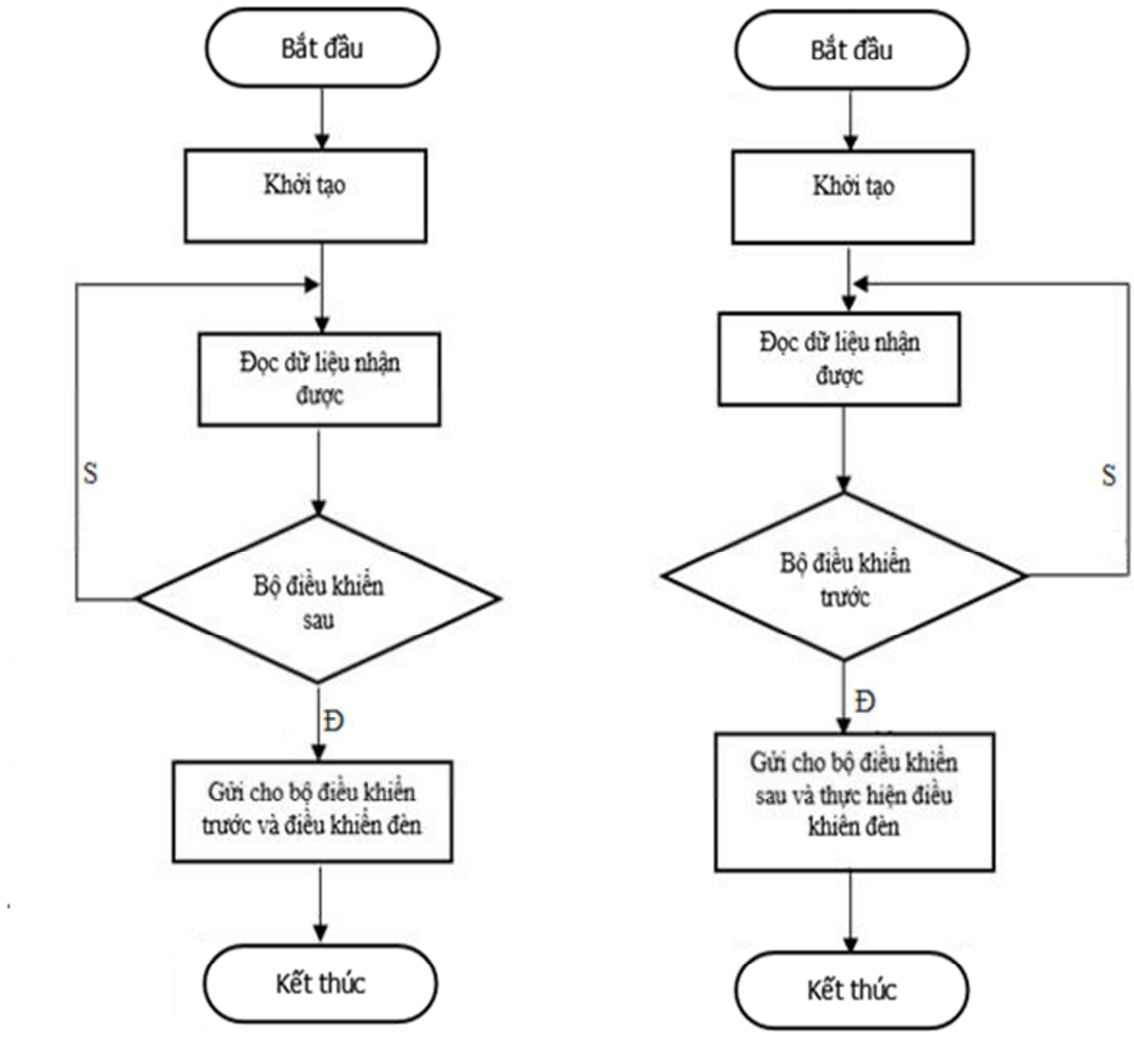
Hình 4.40: Lưu đồ chế độ tự động

Giải thích lưu đồ:

Đầu tiên kiểm tra biến trạng thái của chương trình. Nếu đúng tiếp tục đọc thời gian cài trên HMI và thời gian thực của DS1307. Sau đó so sánh hai giá trị, nếu đúng thì cho phép cài đặt độ sáng cho đèn, nếu không thì nhận giá trị hồi tiếp trạng thái đèn và hiển thị ra màn hình.

4.2.2 Lưu đồ mạch điều khiển phụ

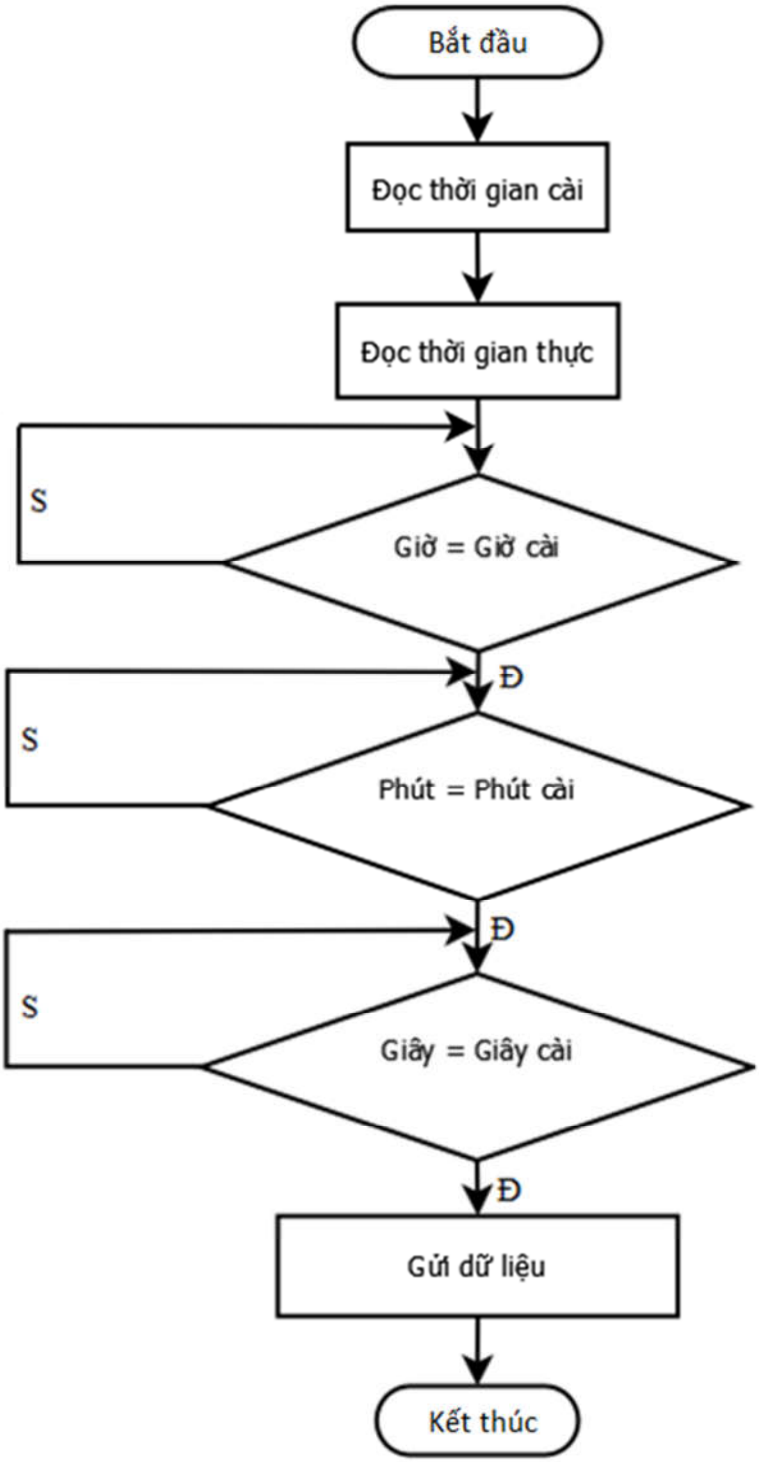
Yêu cầu chương trình: Đọc dữ liệu nhận được từ trạm khác truyền tới sau đó điều khiển độ sáng đèn và gửi dữ liệu nhận được cho trạm sau cũng như phản hồi trạng thái đèn cho trạm trước.



Hình 4.41: Lưu đồ chương trình của mạch điều khiển phụ

Giải thích lưu đồ:

Đầu tiên cấu hình hệ thống bằng cách khởi tạo Xbee, giao tiếp Software Serial, các địa chỉ truyền nhận, cũng như thư viện của chúng. Tiếp đó vòng lặp được thực hiện để đảm bảo chương trình luôn được tuần hoàn. Chương trình sẽ đọc dữ liệu nhận được từ các trạm phát đến sau đó sẽ kiểm tra địa chỉ của gói tin được gửi đến, nếu địa chỉ là của trạm trước thì chương trình sẽ điều khiển đèn và gửi dữ liệu nhận được đến các trạm sau, nếu địa chỉ là của trạm sau thì chương trình sẽ đọc giá trị cảm biến để biết trạng thái đèn sáng hay tắt và gửi dữ liệu đó đến trạm trước.



Hình 4.42: Lưu đồ chương trình cài đặt thời gian điều khiển đèn

Giải thích lưu đồ:

Đầu tiên chương trình sẽ đọc thời gian cài sau đó đọc thời gian hiện tại của DS1307. Tiếp đó chương trình tiến hành so sánh giờ phút giây đã cài và giờ phút giây hiện tại, nếu đúng thì sẽ gửi dữ liệu đến các trạm để điều đèn, nếu sai chương trình sẽ không gửi dữ liệu đi.

4.3 Phần mềm lập trình Arduino IDE

4.3.1 Giới thiệu phần mềm Arduino IDE

Arduino IDE là một phần mềm giúp chúng ta nạp code đã viết vào board mạch và thực thi ứng dụng. Arduino IDE là chữ viết tắt của Arduino Integrated Development Environment, một công cụ lập trình với các board mạch Arduino. Nó bao gồm các phần chính là Editor (trình soạn thảo văn bản, dùng để viết code), Debugger (công cụ giúp tìm kiếm và sửa lỗi phát sinh khi build chương trình), Compiler hoặc interpreter (công cụ giúp biên dịch code thành ngôn ngữ mà vi điều khiển có thể hiểu được và thực thi code theo yêu cầu người dùng).

Công cụ này được đội ngũ kĩ sư của Arduino phát triển và có thể chạy trên Windows, MAC OS X và Linux. Do có tính chất mã nguồn mở nên môi trường lập trình này hoàn toàn miễn phí.

Hình 4.43: Giao diện trang chủ của phần mềm Arduino IDE

4.3.2 Cài đặt và cách sử dụng phần mềm Arduino IDE

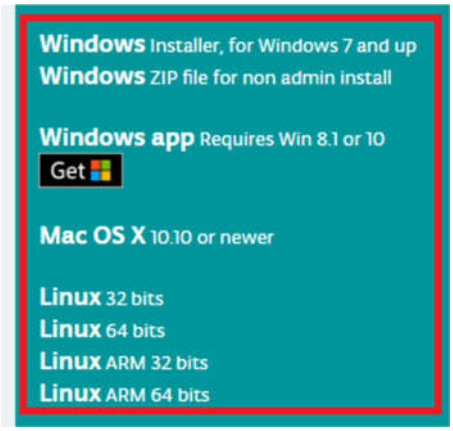
❖ **Cài đặt phần mềm**

Bước 1:

Để cài đặt phần mềm truy cập vào đường link sau để tải phần mềm về máy tính:

https://www.arduino.cc/en/Main/Software

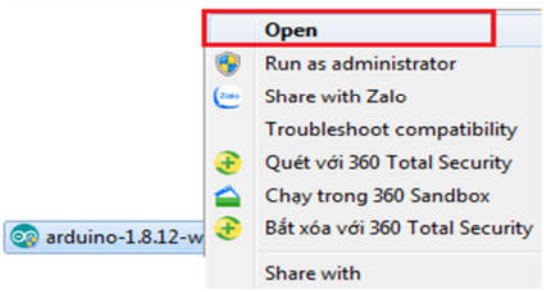
Sau đó chọn phiên bản dành cho máy tính.



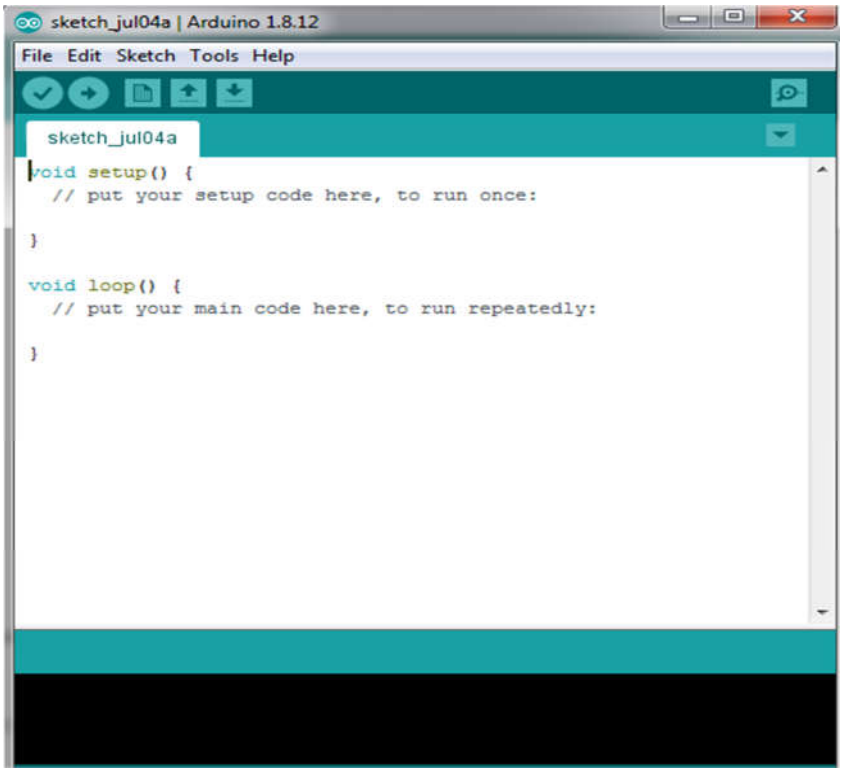
Hình 4.44: Chọn phiên bản cho máy tính

Bước 2: Tiến hành đi chuyển file đến nơi lưu trữ:

Bước 3: Cài đặt phần mềm



Hình 4.45: Chạy file cài đặt

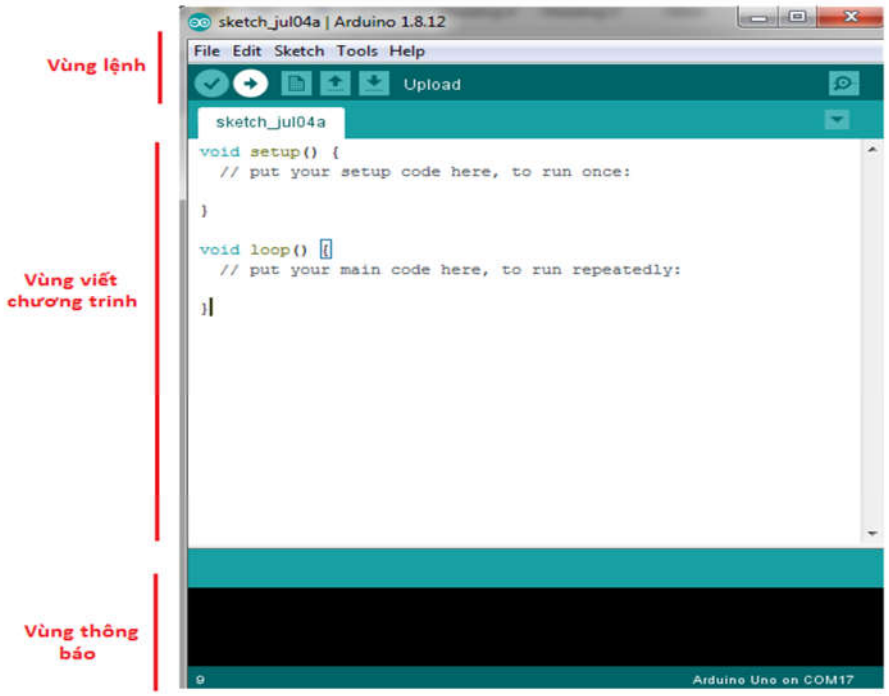


Hình 4.46: Giao diện sau khi cài đặt

❖ **Cách sử dụng phần mềm:**

Phần mềm sẽ có 3 vùng:

- Vùng lệnh
- Vùng viết chương trình
- Vùng thông báo



Hình 4.47 Các vùng làm việc của Arduimo IDE

- **Vùng lệnh:** Bao gồm các nút lệnh menu (File, Edit, Sketch, Tools, Help). Phía dưới là các icon cho phép sử dụng nhanh các chức năng thường dùng của IDE được miêu tả như sau:

Icon	Chức năng
	Biên dịch chương trình đang soạn thảo để kiểm tra các lỗi lập trình.
	Biên dịch và upload chương trình đang soạn thảo.
	Mở một trang soạn thảo mới.
	Mở các chương trình đã lưu.
	Lưu chương trình đang soạn.
	Mở cửa sổ Serial Monitor để gửi và nhận dữ liệu giữa máy tính và board Arduino.

Hình 4.48: Các chức năng của vùng lệnh

- **Vùng viết chương trình:** Các đoạn mã code sẽ được viết tại đây. Tên chương trình của bạn được hiển thị ngay dưới dãy các Icon. Để ý rằng phía sau tên chương trình có một dấu "§". Điều đó có nghĩa là đoạn chương trình chưa được lưu lại.
- **Vùng thông báo (debug)**

Hình 4.49: Vùng thông báo của Arduino IDE

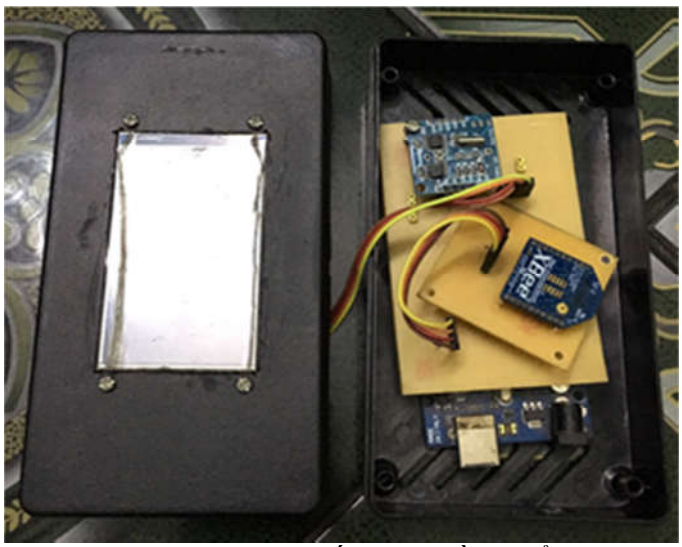
Những thông báo từ IDE sẽ được hiển thị tại đây. Để ý rằng góc dưới cùng bên phải hiển thị loại board Arduino và cổng COM được sử dụng. Luôn chú ý tới mục này bởi nếu chọn sai loại board hoặc cổng COM, phần mềm sẽ không thể upload được code.

4.4 Tiến hành lắp ráp hệ thống hoàn chỉnh

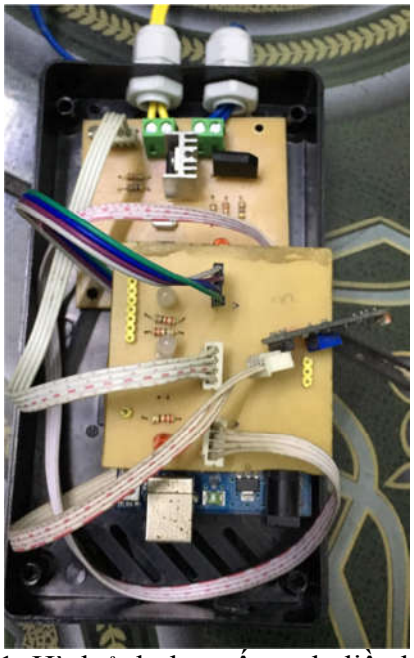
4.4.1 Lắp ráp hệ thống

Mô hình bao gồm 4 trạm: 1 trạm điều khiển trung tâm và 3 trạm điều khiển đèn

- Mạch sẽ được đặt vào hộp nhựa có kích thước: 160x90x80 (mm)
- Đế mạch làm bằng Mica có kích thước: 20x15 (cm)
- Ống PVC có đường kính 8mm



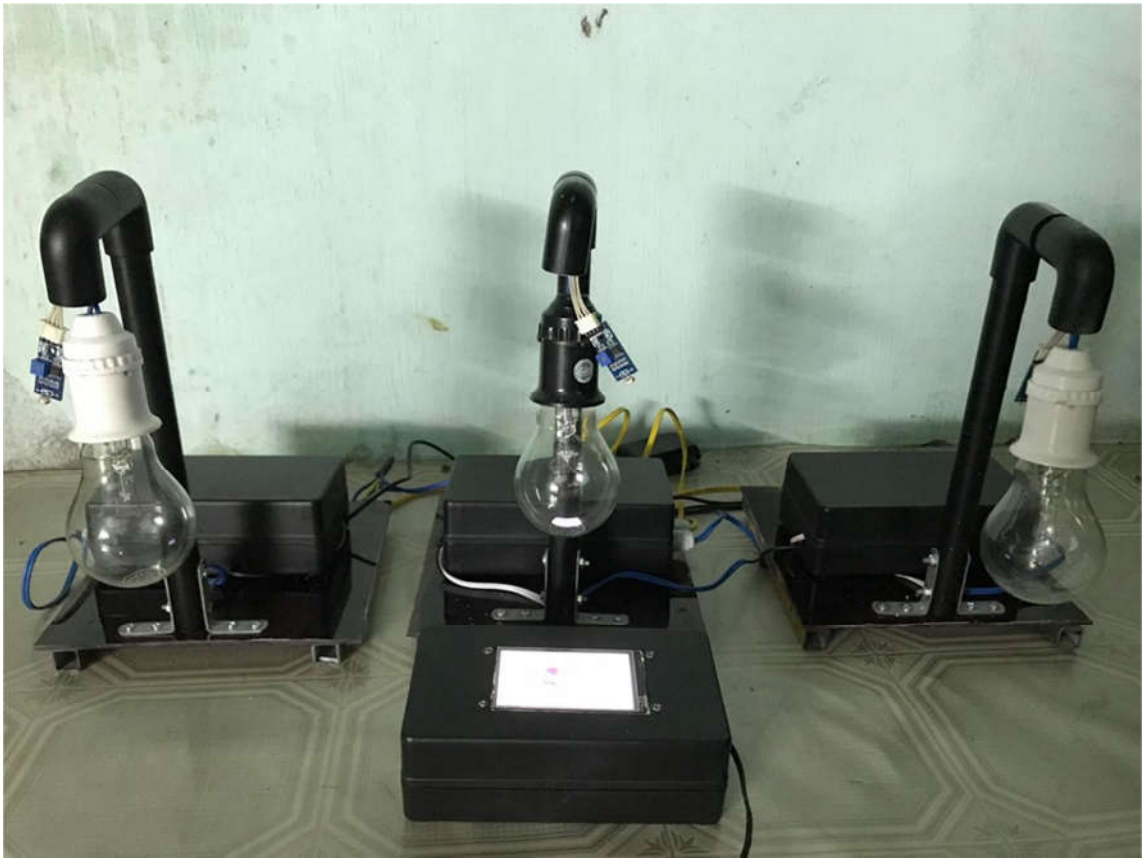
Hình 4.50: Hình ảnh thực tế mạch điều khiển trung tâm



Hình 4.51: Hình ảnh thực tế mạch điều khiển phụ

4.4.2 Mô hình hệ thống

Sau khi đã lắp ráp các mạch và đóng hộp xong nhóm đã tiến hành cố định mạch vào đế mica và được mô hình như sau:



Hình 4.52 Mô hình hệ thống

CHƯƠNG 5: KẾT QUẢ VÀ ĐÁNH GIÁ

5.1 Kết quả đạt được

Trong quá trình thực hiện đồ án, nhóm chúng em đã nghiên cứu các tài liệu chuyên ngành của trường, các tài liệu trên mạng Internet cùng với đó là sự hướng dẫn của thầy **ThS. Phan Vân Hoàn** mà chúng em đã hoàn thành được đề tài này.

Sau khi hoàn thành xong đề tài đề tài này, nhóm chúng em đã cũng cố lại một số kiến thức điện tử công suất đã được học thông qua thiết kế mạch điều chỉnh độ sáng của đèn, tiếp thu thêm được những kiến thức mới về giao thức Zigbee, cách truyền dữ liệu qua lại giữa 2 mô-đun Xbee S2, cũng cố lại các kiến thức cơ bản về lập trình trên Arduino (Arduino Mega 2560 và Arduino Uno R3) và cách giải quyết từng vấn đề thông qua làm việc nhóm.

Hệ thống đã đạt được những yêu cầu đề ra ban đầu. Cụ thể ở đây là việc có thể bật, tắt đèn đồng thời và điều khiển được độ sáng thông qua màn hình HMI.

Sau thời gian nghiên cứu và thi công đồ án tốt nghiệp với đề tài **"THIẾT KẾ VÀ THI CÔNG HỆ THỐNG CHIẾU SÁNG ĐIỀU KHIỂN QUA MẠNG ZIGBEE"** nhóm đã thu được kết quả như sau:

- Nhìn chung, mô hình hoạt động tương đối ổn định, có thể làm việc liên tục, đạt 95% yêu cầu đề ra ban đầu.
- Sản phẩm hoạt động phụ thuộc vào môi trường. Môi trường càng ít vật cản thì thiết bị hoạt động càng tốt.
- Khoảng cách truyền nhận trong nhà là 30m.
- Khoảng cách truyền nhận ngoài trời là 100m.
- Hệ thống hẹn giờ bật đèn chính xác theo thời gian cài.
- Có thể điều chỉnh trực tiếp độ sáng đèn thông qua màn hình.
- Giám sát được tình trạng hư hỏng của đèn qua màn hình.

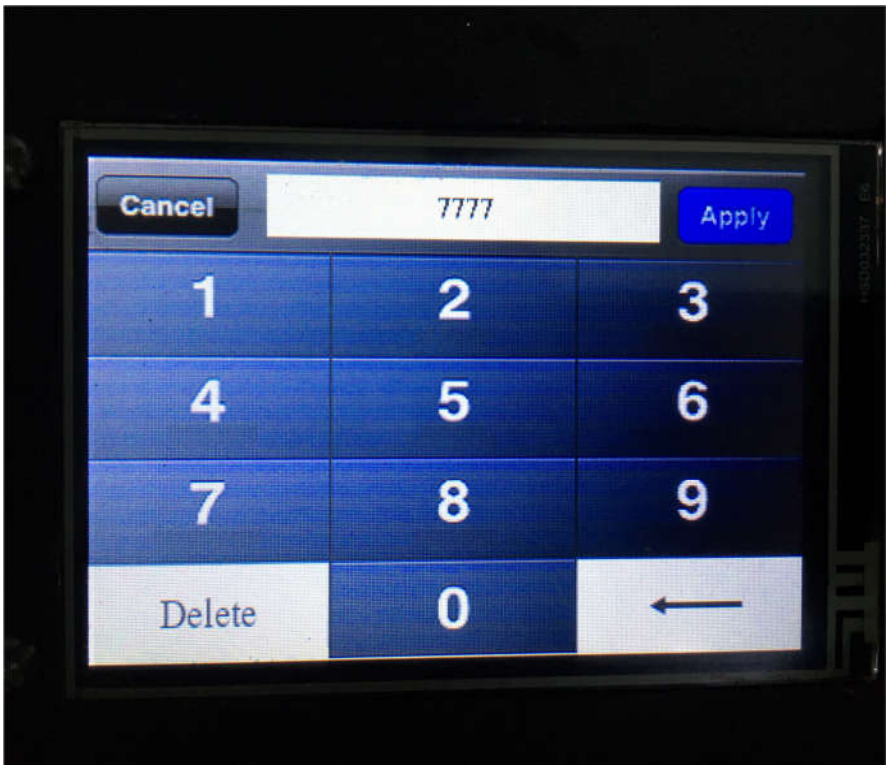
5.2 Kết quả chạy thử nghiệm hệ thống

5.2.1 Giao diện điều khiển hệ thống

Sau khi cấp nguồn cho phép hệ thống bắt đầu hoạt động, khi đó màn hình HMI sẽ hiển thị như sau:

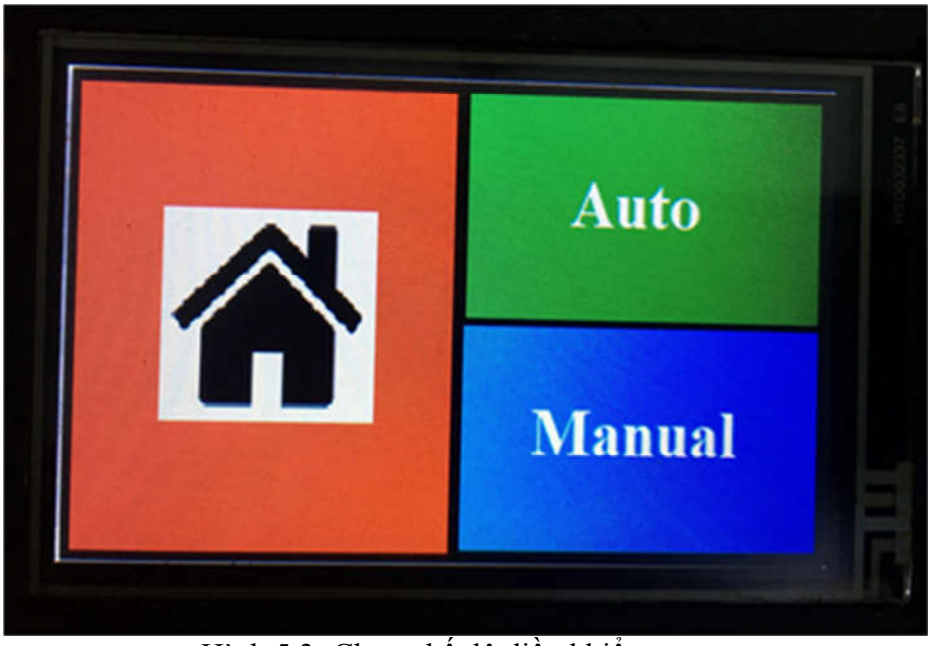
Hình 5.1: Màn hình chào

Màn hình sẽ hiển thị tên và mã số sinh viên của nhóm, khi nhấn vào mũi tên hệ thống sẽ yêu cầu nhập mật khẩu truy cập vào hệ thống như sau:



Hình 5.2: Nhập mật khẩu truy cập hệ thống

Mật khẩu của hệ thống là: **7777**. Sau khi nhập xong nhấn vào **Apply** nếu sai hệ thống sẽ yêu cầu nhập lại, nếu đúng màn hình sẽ hiển thị trình điều khiển như sau:



Hình 5.3: Chọn chế độ điều khiển

Màn hình lúc này sẽ hiển thị hai chế độ cho người dung lựa chọn:

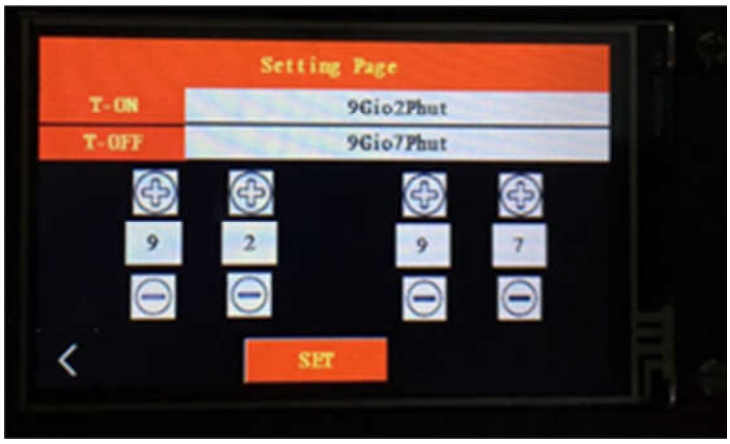
- Auto: Chế độ tự động
- Manual: Chế độ điều khiển bằng tay

- **Chế độ Auto**

Hình 5.4: Chế độ Auto

Giao diện chế độ Auto hiển thị thời gian để người dùng dễ theo dõi khi cài đặt tự động tắt mở. Nhấn nút **SET** trên màn hình sẽ chuyển qua trang cài đặt thời gian

tắt và mở đèn. Nhấn vào **Manual** thì sẽ chuyển chế độ chỉnh tay. Nhấn **HOME** sẽ trở về giao diện màn hình chào.



Hình 5.5: Cài đặt thời gian tắt mở đèn

Ở trang cài đặt thời gian tắt mở đèn, bên trái là cài thời gian mở đèn còn bên phải là cài đặt thời gian tắt đèn, các nhấn phím cộng trừ để tăng giảm thời gian muốn cài đặt. Sau đó ta nhấn nút **SET** trên màn hình để cài đặt thời gian và trên màn hình có hiển thị thời gian tắt mở mà ta vừa cài.

- **Chế độ Manual**

Hình 5.6: Chế độ Manual

Ở chế độ Manual bao gồm các nút nhấn:

- Nút Home: trở về màn hình chào như chế độ auto.
- Nút Auto: chuyển qua chế độ tự động.
- Nút 70%: sáng 70% công suất.
- Nút 100%: sáng 100% công suất.

- Nút ON: sáng 100% công suất (phải nhấn nút ON thì mới có thể nhấn nút 70% và 100%).
- Nút OFF: Tắt đèn

Hình 5.7: Trạng thái đèn

Trong hình 5.7 hệ thống hồi tiếp tình trạng hư hỏng và hoạt động của đèn. Khi đèn nào hoạt động bình thường thì sẽ hiển thị lên màn hình HMI là đèn sáng và khi nào đèn bị hư hỏng thì sẽ hiển thị đèn tắt ra màn hình. Ở hình 5.7 đèn 1 và 2 hoạt động bình thường còn đèn 3 hư.

Hình 5.8: Hình ảnh hệ thống đang hoạt động

❖ **Bảng thực nghiệm kết quả chạy thực tế:**

Bảng 5.1: Kết quả chạy thực nghiệm

Số lần nhấn	Nội dung	Kết quả
25	Phím Apply	24/25
25	Phím Home	23/25
25	Phím Auto	23/25
25	Phím Manual	24/25
25	Phím SET	23/25
25	Phím ON	23/25
25	Phím OFF	23/25
25	Phím 100%	22/25
25	Phím 70%	23/25
25	Hiện thị trạng thái đèn	20/25
25	Hiển thị thời gian cài đặt	24/25

5.3 Nhận xét và đánh giá

- Độ ổn định hệ thống đạt 88%
- Giao tiếp màn hình HMI giúp người dùng dễ dàng sử dụng và giám sát.
- Thời gian đáp ứng điều khiển khoảng từ 2 - 3 giây.
- Phản hồi trạng thái chưa có độ chính xác cao.
- Tuy nhiên, do sự hạn chế về kiến thức cũng như thời gian, nguồn tham khảo chủ yếu là thông qua mạng Internet nên đề tài còn có nhiều sai sót và một số hạn chế: Hoạt động chưa tốt ở vùng có nhiễu, tính thẩm mỹ và độ chính xác chưa cao, chưa có chức năng giám sát từ xa.

CHƯƠNG 6: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

6.1 Kết luận

Kiến thức về kỹ thuật rất sâu, rộng và không ngừng cập nhập từng ngày, chính vì thế cần phải có nhiều thời gian nghiên cứu để có thể thực hiện đề tài đạt kết quả tốt nhất. Trong giới hạn thời gian mà trường đưa ra, nhóm đã hoàn thành đề tài: **"THIẾT KẾ VÀ THI CÔNG HỆ THỐNG CHIẾU SÁNG ĐIỀU KHIỂN QUA MẠNG ZIGBEE"**. Đạt được 95% yêu cầu đề ra ban đầu, cụ thể như sau:

- Thiết kế và thi công được hệ thống mạng không dây Zigbee có thể giao tiếp giữa các trạm với nhau.
- Hệ thống có thể truyền dữ liệu đi được xa tương đối ổn định và chính xác.
- Thiết kế được giao diện điều khiển hệ thống ánh sáng cũng như giám sát trạng thái của hệ thống bằng màn hình Nextion HMI 3.2 Inch.
- Hệ thống có thể chạy trong thời gian dài và tương đối ổn định.
- Hệ thống có chức năng phản hồi trạng thái đèn giữa các trạm.

Dù đã cố gắng nghiên cứu và thực hiện các mục tiêu ban đầu đề ra nhưng hệ thống vẫn còn một số hạn chế như sau:

- Thời gian đáp ứng truyền tín hiệu đi xa từ 2- 3 giây.
- Hoạt động chưa tốt ở vùng có nhiễu.
- Hệ thống chưa có độ thẩm mỹ cao.
- Hệ thống phản hồi chưa có độ chính xát cao.

6.2 Hướng phát triển đề tài

- Hệ thống có thể mở rộng hệ thống lên nhiều đèn và có thể áp dụng vào thực tế.
- Tối ưu phần mềm, nâng cấp phần cứng để tăng độ tin cậy và hiệu quả.
- Kết hợp IoT (Internet of Things) vào hệ thống để điều khiển và giám sát đèn qua Internet cũng như qua đó có giúp người dùng đánh giá được hiệu quả về tiết kiệm năng lượng của hệ thống.

TÀI LIỆU THAM KHẢO

[1] Đào Xuân Sang và Ngô Đức Phú, "*Nghiên cứu, tìm hiểu và thi công tủ rau cho căn hộ*", Đồ án tốt nghiệp, trường ĐHSPKT, Tp.HCM, 2019.

[2] Nguyễn Đình Phú, "*Giáo Trình Vi Xử Lý*", Xuất bản ĐH Quốc Gia Tp.HCM, 2016

[3] Hoàng Ngọc Văn, "*Giáo Trình Điện Tử Công Suất*" Tp. Hồ Chí Minh 2014

[4] Bài viết Giao thức ZigBee trong truyền thông công nghiệp trên trang automation.net.vn

[5] Trần Thu Hà, Trương Thị Bích Ngà, Nguyễn Thị Lưỡng... *"Giáo Trình Điện Tử Cơ Bản*" Tp. Hồ Chí Minh: Đại học Quốc gia Tp. HCM, 2013

[6] Các bài viết trong http://arduino.vn

[7] Bài viết Điều khiển góc kích mở của Triac để thay đổi độ sáng của đèn trên trang hocdientu.vn

[8] Bài viết Tổng quan về công nghệ zigbee trên trang iotvietnam.com.

PHỤ LỤC

CHƯƠNG TRÌNH ĐIỀU KHIỂN CỦA BỘ ĐIỀU KHIỂN TRUNG TÂM

```
#include "Time.h"
#include <XBee.h>
#include <SoftwareSerial.h>
#include <Nextion.h>
#define COOR 0x419B5210 // địa chỉ XBee trung tâm
#define D1 0x409B9150 // địa chỉ XBee đèn 1
#define D2 0x40E74427 // địa chỉ XBee đèn 2
#define D3 0x418F1397 // địa chỉ XBee đèn 3
#define DCCOOR "1084889937" //địa chỉ dạng thập phân XBee trung tâm
#define DCD1 "1083937104" // địa chỉ dạng thập phân XBee đèn 1
#define DCD2 "1088898087" // địa chỉ dạng thập phân XBee đèn 2
#define DCD3 "1099895703" // địa chỉ dạng thập phân XBee đèn 3

// khởi tạo XBee
XBee xbee = XBee();
XBeeResponse response = XBeeResponse();

// khai báo địa chỉ các module Xbee giao tiếp
XBeeAddress64 addr64_1 = XBeeAddress64(0x0013A200, D1);
XBeeAddress64 addr64_2 = XBeeAddress64(0x0013A200, D2);

// khởi tạo phản hồi Xbee
ZBRxResponse rx = ZBRxResponse();
ZBRxIoSampleResponse ioSample = ZBRxIoSampleResponse();

// khai báo biến
const char gm[10] =
{0x30,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39};
uint8_t Data_send_a[1] = {0x00};
String va0;
String Data_2; // du lieu nhan duoc tu xbee
String Address;
int  doc,dim=0,mode=3,temp,ttd_auto,  Gui_dimming,  Gui_dimming_cu,
dimming_chuc, dimming_donvi;
int gio_on=18,phut_on=0;
int gio_off=6,phut_off=0;
int gio_c=0, phut_c=0, giay_c=0;
int ON, ON_cu, OFF, OFF_cu;
int ttd  ,hienthiden;
String ON_chuoi, OFF_chuoi;
float hienthiphantram;
String message,solieu;
SoftwareSerial nextion(50,51);
```

```
Nextion myNextion(nextion, 9600);
//-----------------------------------
void setup() {
  Serial.begin(9600);
  Serial1.begin(9600);
  myNextion.init();
   Wire.begin();
 //setTime(23,10,0,6,30,6,20);
 xbee.setSerial(Serial1);
 // khởi tạo TIMER 1
  cli();
    /* Reset Timer/Counter1 */
   TCCR1A = 0;
   TCCR1B = 0;
   TIMSK1 = 0;
    /* Setup Timer/Counter1 */
   TCCR1B |= (1 << CS12) | (0 << CS11) | (1 << CS10);
// prescale = 1024
   TCNT1 = 25000; // 0.5 s
   TIMSK1 = (1 << TOIE1);   // Overflow   interrupt enable
   sei();                // cho phép ngắt   toàn cục

}
//-----------------------------------
void chon_mode()
{
  if ((message=="65 2 2 0 ffff ffff ffff")|(message=="65 3 7 0 ffff
ffff ffff"))
  {
    dimming = 0;
    mode = 0;  //CHE DO AUTO
    Serial.println("AUTO");
    gio_cu = 0;
    phut_cu = 0;
    giay_cu = 0;
  }
  if ((message=="65 4 2 0 ffff ffff ffff")|(message=="65 2 1 0 ffff
ffff ffff"))
  {
    dimming = 0;
    mode = 1;  //CHE DO MANUAL
    ON=1;
    Serial.println("MANUAL");
    gio_cu = 0;
    phut_cu = 0;
    giay_cu = 0;
  }
  //-----------------------------------
```

```
//CHẠY CHẾ ĐỘ AUTO//
  while (mode == 0)
  {
    auto_mode();
    message = myNextion.listen();
    if(message != ""){ // if a message is received...
    Serial.println(message); //...print it out
  }
    truyen();
    nhan();
    if ((message=="65 2 2 0 ffff ffff ffff")|(message=="65 3 7 0
ffff ffff ffff"))
    {
      dim = 0;
      mode = 0;  //CHE DO AUTO
      ON=0;
      Serial.println("AUTO");
      gio_cu = 0;
      phut_cu = 0;
      giay_cu = 0;
    }
    if ((message=="65 4 2 0 ffff ffff ffff")|(message=="65 2 1 0
ffff ffff ffff"))
    {
      dim = 0;
      mode = 1;  //CHE DO MANUAL
      ON=1;

      gio_c = 0;
      phut_c = 0;
      giay_c = 0;
    }

  }
//-----------------------------------
// CHẠY CHẾ ĐỘ MANUAL//
  while (mode == 1)
  {
    manual_mode();
    truyen();
    nhan();
    if ((message=="65 2 2 0 ffff ffff ffff")|(message=="65 3 7 0
ffff ffff ffff"))
    {
```

```
      dim = 0;
      mode = 0;  //CHE DO AUTO
      gio_cu = 0;
      phut_cu = 0;
      giay_cu = 0;
    }
    if ((message=="65 4 2 0 ffff ffff ffff")|(message=="65 2 1 0
ffff ffff ffff"))
    {
      dim = 0;
      mode = 1;  //CHE DO MANUAL
      ON=1;
      gio_c = 0;
      phut_c = 0;
      giay_c = 0;
    }
  }
}
//-----------------------------------------
void truyen()
{
    Gui_dimming = dim;
    if ( Gui_dimming_cu != Gui_dimming)
 {
  if ( Gui_dimming <=9 )
  {
        Data_send_a[0]= gm[Gui_dimming];
        ZBTxRequest zbtx = ZBTxRequest(addr64_1, Data_send_a,
sizeof(Data_send_a));
        xbee.send(zbtx);
  }
   if ( Gui_dimming <=9 )
  {
        Data_send_a[0]= gm[Gui_dimming];
        ZBTxRequest zbtx = ZBTxRequest(addr64_2, Data_send_a,
sizeof(Data_send_a));
        xbee.send(zbtx);
```

```
  }
}
//-----------------------------------------
//NHAN DU LIEU TU XBEE//
  void nhan()
  {
     xbee.readPacket();
    if (xbee.getResponse().isAvailable())
    {
      if (xbee.getResponse().getApiId() == ZB_RX_RESPONSE)
        {
            xbee.getResponse().getZBRxResponse(rx);
            XBeeAddress64 senderLongAddress =
rx.getRemoteAddress64();
            Address = String(senderLongAddress.getLsb());//Lay gia
tri LSB
            Data_2="";
            for (int i= 0; i < rx.getDataLength(); i++)
                {
                  if (!iscntrl(rx.getData()[i]))
Data_2=Data_2+String(char(rx.getData()[i]));
                }
             ttd = Data_2.toInt();
        }
      }
    }
//-----------------------------------------
void auto_mode()
{
  if(message=="65 5 4 0 ffff ffff ffff")
   {
    gio_on = myNextion.getComponentValue("va0");
    phut_on = myNextion.getComponentValue("va1");
    gio_off = myNextion.getComponentValue("va2");
    phut_off = myNextion.getComponentValue("va3");
```

```
    Serial.println("gio on: " + String(gio_on) + " phut on: " +
String(phut_on) + "gio off: " + String(gio_off) + " phut off: " +
String(phut_off));
    dim = 0;
    message = myNextion.listen();
    gio_c = 0;
    phut_c = 0;
    giay_c = 0;
   }
  if(message=="65 5 17 0 ffff ffff ffff")
  {
    gio_c = 0;
      phut_c = 0;
      giay_c = 0;
  }
  readDS1307();
  if ((hour >= gio_on)&&(minute >= phut_on)&&(hour < 22))
  {
    dim = 1;
  }
  if ((gio_off >= hour)&&(phut_off>=minute)&&(hour <
gio_on)&&(minute < phut_on))
  {
    dim = 0;
  }
  if (giay_c != second)
  {
    myNextion.setComponentText("giay",String(second));
    myNextion.setComponentText("phut",String(minute));
     myNextion.setComponentText("gio",String(hour));
    if(phut_c != minute)
    {
      myNextion.setComponentText("phut",String(minute));
      if(gio_c != hour)
      {
        myNextion.setComponentText("gio",String(hour));
        gio_c = hour;
```

```
      }
      phut_c = minute;
    }
    giay_cu = second;
  }
}

//-----------------------------------------
void manual_mode()
{
    message = myNextion.listen();
    if(message != ""){ // if a message is received...
    Serial.println(message);} //...print it out
  if (message=="65 3 4 1 ffff ffff ffff")
  {
    if (ON==1)
    {
      dim = 1;
      myNextion.setComponentText("level","100");
      Serial.println(dimming);
      }
  }
  if(message=="65 3 1 0 ffff ffff ffff")
  {
    if(ON==1)
    {
      dim = 2;
      myNextion.setComponentText("level","70");
     Serial.println(dimming);
    }
  }
   if (message=="65 3 2 0 ffff ffff ffff") // NHAN ON
  {
    dim = 1;
    myNextion.setComponentText("level","100");
    ON=1;
    Serial.println(dimming);
```

```
    }
  if (message=="65 3 3 0 ffff ffff ffff")  //NHAN OFF
  {
    dim = 0;
    myNextion.setComponentText("level","0");
    ON=0;
  Serial.println(dimming);
  }
}
//----------------------------------------

void kiemtrattd()
{
  switch (ttd)
  {
    case 3:       // den 1 hu
      myNextion.sendCommand("vis sang_1,0");
      myNextion.sendCommand("vis sang_2,1");
      myNextion.sendCommand("vis sang_3,1");
      break;
    case 6:       // den 2 hu
      myNextion.sendCommand("vis sang_1,1");
      myNextion.sendCommand("vis sang_2,0");
      myNextion.sendCommand("vis sang_3,1");
      break;
    case 5:       // den 3 hu
      myNextion.sendCommand("vis sang_1,1");
      myNextion.sendCommand("vis sang_2,1");
      myNextion.sendCommand("vis sang_3,0");
      break;
     case 7:      // khong den nao hu
      myNextion.sendCommand("vis sang_1,1");
      myNextion.sendCommand("vis sang_2,1");
      myNextion.sendCommand("vis sang_3,1");
      break;
     case 2:      // đèn 1 và 2 hư
      myNextion.sendCommand("vis sang_1,0");
```

```
      myNextion.sendCommand("vis sang_2,0");
      myNextion.sendCommand("vis sang_3,1");
      break;
    case 1:      // đèn 1 và 3 hư
      myNextion.sendCommand("vis sang_1,0");
      myNextion.sendCommand("vis sang_2,1");
      myNextion.sendCommand("vis sang_3,0");
      break;
    case 4:      // đèn 2 và 3 hư
      myNextion.sendCommand("vis sang_1,1");
      myNextion.sendCommand("vis sang_2,0");
      myNextion.sendCommand("vis sang_3,0");
      break;
    case 0:      // 3 đèn hư
      myNextion.sendCommand("vis sang_1,0");
      myNextion.sendCommand("vis sang_2,0");
      myNextion.sendCommand("vis sang_3,0");
      break;
   }
}
//----------------------------------------

#include <Wire.h>
/* Địa chỉ của DS1307 */
const byte DS1307 = 0x68;
/* Số byte dữ liệu sẽ đọc từ DS1307 */
const byte NumberOfFields = 7;
int second, minute, hour, day, wday, month, year;
 int bcd2dec(byte num)
{
        return ((num/16 * 10) + (num % 16));
}
/* Chuyển từ Decimal sang BCD */
int dec2bcd(byte num)
{
        return ((num/10 * 16) + (num % 10));
}
```

```
void readDS1307()
{
    /* Chuyển từ format BCD (Binary-Coded Decimal) sang Decimal */
        Wire.beginTransmission(DS1307);
        Wire.write((byte)0x00);
        Wire.endTransmission();
        Wire.requestFrom(DS1307, NumberOfFields);
        second = bcd2dec(Wire.read() & 0x7f);
        minute = bcd2dec(Wire.read() );
        hour   = bcd2dec(Wire.read() & 0x3f); // chế độ 24h.
        wday   = bcd2dec(Wire.read() );
        day    = bcd2dec(Wire.read() );
        month  = bcd2dec(Wire.read() );
        year   = bcd2dec(Wire.read() );
        year += 2000;
}
/* Chuyển từ format BCD (Binary-Coded Decimal) sang Decimal */
void printDigits(int digits){
    // các thành phần thời gian được ngăn chách bằng dấu :
    Serial.print(":");
    if(digits < 10)
    Serial.print('0');
    Serial.print(digits);
}
/* cài đặt thời gian cho DS1307 */
void setTime(byte hr, byte minu, byte sec, byte wd, byte d, byte
mth, byte yr)
{
        Wire.beginTransmission(DS1307);
        Wire.write(byte(0x00)); // đặt lại pointer
        Wire.write(dec2bcd(sec));
        Wire.write(dec2bcd(minu));
        Wire.write(dec2bcd(hr));
        Wire.write(dec2bcd(wd)); // day of week: Sunday = 1,
Saturday = 7
        Wire.write(dec2bcd(d));
        Wire.write(dec2bcd(mth));
```

```
        Wire.write(dec2bcd(yr));
        Wire.endTransmission();
}
void digitalClockDisplay(){
    Serial.print(hour);
    printDigits(minute);
    printDigits(second);
    Serial.print(" ");
    Serial.print(day);
    Serial.print(" ");
    Serial.print(month);
    Serial.print(" ");
    Serial.print(year);
    Serial.println();
}
void loop()
{
  message = myNextion.listen(); //check for message
   if(message != ""){ // if a message is received...
    Serial.println(message); //...print it out
  }
  chon_mode();
}
ISR (TIMER1_OVF_vect)
{
TCNT1 = 25000; // reset lại TIMER
kiemtrattden();
}
```

CHƯƠNG TRÌNH BỘ ĐIỀU KHIỂN ĐÈN

Bộ điều khiển đèn 1

```
#include <XBee.h>       // thư viện Xbee
#define COOR 0x419B5210 // địa chỉ XBee trung tâm
#define D1 0x409B9150 // địa chỉ XBee đèn 1
#define D2 0x40E74427 // địa chỉ XBee đèn 2
#define D3 0x418F1397 // địa chỉ XBee đèn 3
#define DCCOOR "1084889937" //địa chỉ dạng thập phân XBee trung tâm
#define DCD1 "1083937104" // địa chỉ dạng thập phân XBee đèn 1
#define DCD2 "1088898087" // địa chỉ dạng thập phân XBee đèn 2
#define DCD3 "1099895703" // địa chỉ dạng thập phân XBee đèn 3

// chọn chân triac, chân cảm biến
unsigned char AC_LOAD = 3;
int cb = 7;

// khởi tạo XBee
XBee xbee = XBee();
XBeeResponse response = XBeeResponse();

// khai báo địa chỉ các module Xbee board đèn và cảm biến
XBeeAddress64 addr64_1 = XBeeAddress64(0x0013A200, COOR);
XBeeAddress64 addr64_2 = XBeeAddress64(0x0013A200, D1);
XBeeAddress64 addr64_3 = XBeeAddress64(0x0013A200, D2);
XBeeAddress64 addr64_4 = XBeeAddress64(0x0013A200, D3);

// khởi tạo phản hồi XBee
ZBRxResponse rx = ZBRxResponse();
ZBRxIoSampleResponse ioSample = ZBRxIoSampleResponse();

// khai báo biến
unsigned int dimtime , dim = 0 , dimming_cu,dlieu, tt_m, tt_c,
ttden;
int value_m, value_c;
const char
gm[10]={0x30,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39};
String Data_2; // du lieu nhan duoc tu xbee
String Address, diachi_s;
uint8_t Data_send_a[1] = {0x00};
//-----------------------------------------
void setup()
{
  pinMode(AC_LOAD, OUTPUT);

  pinMode(cb,INPUT);
  attachInterrupt(0, zero_crosss_int, FALLING);  //chân ngắt nối vào
chân số 2
  Serial.begin(9600);
  xbee.setSerial(Serial);
}

void zero_crosss_int()  // hàm ngắt để thực hiện kích TRIAC
{
```

```
  if ( dim == 0)  detachInterrupt(0);
   if (dim == 1) dimtime = 1;
   if (dim > 1) dimtime = 2200;
   delayMicroseconds(dimtime);    // thời gian off TRIAC
  digitalWrite(AC_LOAD, HIGH);   // kích TRIAC
  delayMicroseconds(10);         // trì hoãn để đảm bảo đã kích
TRIAC
  digitalWrite(AC_LOAD, LOW);    // ngắt xung kích
}
//----------------------------------------

void loop() {
    xbee.readPacket();
    if (xbee.getResponse().isAvailable())
    {
      if (xbee.getResponse().getApiId() == ZB_RX_RESPONSE)
        {
            xbee.getResponse().getZBRxResponse(rx);
            XBeeAddress64 senderLongAddress =
rx.getRemoteAddress64();
          Address = String(senderLongAddress.getLsb());//Lay gia
tri LSB
            Data_2="";
            for (int i= 0; i < rx.getDataLength(); i++)
                {
             if (!iscntrl(rx.getData()[i]))
Data_2=Data_2+String(char(rx.getData()[i]));
                }
             dlieu = Data_2.toInt();
          }
if ( Address == DCCOOR)
            {
              dim = dlieu;
              if ( dim == 0)  detachInterrupt(0);
        else  attachInterrupt(0, zero_crosss_int, RISING);
                if ( dimming != dimming_cu)
                {
                    if ( dim <= 9)
                    {
                        Data_send_a[0] = gm[dim];
 ZBTxRequest zbtx = ZBTxRequest(addr64_3,
Data_send_a,sizeof(Data_send_a));
                        xbee.send(zbtx);
                    }
                    if ( dim <= 9)
                    {
                        Data_send_a[0] = gm[dim];
 ZBTxRequest zbtx = ZBTxRequest(addr64_4, Data_send_a,
sizeof(Data_send_a));
                        xbee.send(zbtx);
                    }
            }

   value_m = digitalRead(cb);
            if (Address == DCD2)
            {
```

```
              tt_m = dlieu;
              if ( tt_c != stt_m)
              {
                    if ( !value_m == 0 )
                    {
                        if ( tt_m == 0) ttden = 0;
                        if ( tt_m == 1) ttden = 1;
                        if ( tt_m == 2) ttden = 2;
                        if ( tt_m == 3) ttden = 3;
                    }
                    if ( !value_m == 1)
                    {
                        if ( tt_m == 0) ttden = 4;
                        if ( tt_m == 1) ttden = 5;
                        if ( tt_m == 2) ttden = 6;
                        if ( tt_m == 3) ttden = 7;
                    }
                    Data_send_a[0] = kytu[ttd];
   ZBTxRequest
zbtx=ZBTxRequest(addr64_1,Data_send_a,sizeof(Data_send_a));
                    xbee.send(zbtx);
                    tt_c = tt_m;
              }
           }
           if ( value_c != value_m)
           {
                    if ( !value_m == 0 )
                    {
                        if ( tt_m == 0) ttden = 0;
                        if ( tt_m == 1) ttden = 1;
                        if ( tt_m == 2) ttden = 2;
                        if ( tt_m == 3) ttden = 3;
                    }
                    if ( !value_m == 1)
                    {
                        if ( tt_m == 0) ttden = 4;
                        if ( tt_m == 1) ttden = 5;
                        if ( tt_m == 2) ttden = 6;
                        if ( tt_m == 3) ttden = 7;
                    }
                    Data_send_a[0] = gm[ttd];
    ZBTxRequest
zbtx=ZBTxRequest(addr64_1,Data_send_a,sizeof(Data_send_a));
                    xbee.send(zbtx);
                    value_c = value_m;
           }
}
//----------------------------------------
```

Bộ điều khiển đèn 2

```
#include <XBee.h>       // thư viện Xbee
#define COOR 0x419B5210 // địa chỉ XBee trung tâm
#define D1 0x409B9150 // địa chỉ XBee đèn 1
#define D2 0x40E74427 // địa chỉ XBee đèn 2
#define D3 0x418F1397 // địa chỉ XBee đèn 3
```

```
#define DCCOOR "1084889937" //địa chỉ dạng thập phân XBee trung tâm
#define DCD1 "1083937104" // địa chỉ dạng thập phân XBee đèn 1
#define DCD2 "1088898087" // địa chỉ dạng thập phân XBee đèn 2
#define DCD3 "1099895703" // địa chỉ dạng thập phân XBee đèn 3

// chọn chân triac, chân cảm biến
unsigned char AC_LOAD = 3;
int cb = 7;

// khởi tạo XBee
XBee xbee = XBee();
XBeeResponse response = XBeeResponse();

// khai báo địa chỉ các module Xbee board đèn và cảm biến
XBeeAddress64 addr64_1 = XBeeAddress64(0x0013A200, COOR);
XBeeAddress64 addr64_2 = XBeeAddress64(0x0013A200, D1);
XBeeAddress64 addr64_3 = XBeeAddress64(0x0013A200, D2);
XBeeAddress64 addr64_4 = XBeeAddress64(0x0013A200, D3);

// khởi tạo phản hồi XBee
ZBRxResponse rx = ZBRxResponse();
ZBRxIoSampleResponse ioSample = ZBRxIoSampleResponse();

// khai báo biến
unsigned int dimtime , dim = 0 , dimming_cu,dlieu, tt_m, tt_c,
ttden;
int value_m, value_c;
const char
gm[10]={0x30,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39};
String Data_2; // du lieu nhan duoc tu xbee
String Address, diachi_s;
uint8_t Data_send_a[1] = {0x00};
//-----------------------------------------

void setup()
{
  pinMode(AC_LOAD, OUTPUT);

  pinMode(cb,INPUT);
  attachInterrupt(0, zero_crosss_int, RISING);  //chân ngắt nối vào
chân số 2
  Serial.begin(9600);
  xbee.setSerial(Serial);
}
//-----------------------------------------

void zero_crosss_int()  // hàm ngắt để thực hiện kích TRIAC
{
   if ( dim == 0)  detachInterrupt(0);
   if (dim == 1) dimtime = 1;
   if (dim > 1) dimtime = 2200;
  delayMicroseconds(dimtime);    // thời gian off TRIAC
  digitalWrite(AC_LOAD, HIGH);   // kích TRIAC
  delayMicroseconds(10);         // trì hoãn để đảm bảo đã kích
TRIAC
  digitalWrite(AC_LOAD, LOW);    // ngắt xung kích
```

```
}
//----------------------------------------
void doc()
{
xbee.readPacket();
    if (xbee.getResponse().isAvailable())
    {
      if (xbee.getResponse().getApiId() == ZB_RX_RESPONSE)
        {
            xbee.getResponse().getZBRxResponse(rx);
            XBeeAddress64 senderLongAddress =
rx.getRemoteAddress64();
           Address = String(senderLongAddress.getLsb());//Lay gia
tri LSB
            Data_2="";
            for (int i= 0; i < rx.getDataLength(); i++)
                {
           if (!iscntrl(rx.getData()[i]))
Data_2=Data_2+String(char(rx.getData()[i]));
                }
             dlieu = Data_2.toInt();
        }
    }
    if (( Address == DCD1)||(Address == DCCOOR))
    {
        dim = dlieu;
        if ( dim == 0)  detachInterrupt(0);
        else  attachInterrupt(0, zero_crosss_int, RISING);
                if ( dimming != dimming_cu)
                {
                    if ( dim <= 9)
                    Data_send_a[0] = gm[dim];
    ZBTxRequest zbtx=ZBTxRequest(addr64_4, Data_send_a ,sizeof
(Data_send_a));
                        xbee.send(zbtx);
                    }
    }
}
//----------------------------------------
void gui()
{
tt_m = digitalRead(cb);
    if ( Address == DCD3)
    {
        stt_m = dlieu;
        if ( stt_c != stt_m )
        {
              if ( stt_m == 0)
              {
                  if( !tt_m == 0) ttden=0;
                  if( !tt_m == 1) ttden=1;
              }
              if ( stt_m == 1)
              {
                  if ( !tt_m == 0) ttden=2;
```

```
                if ( !tt_m == 1) ttden=3;
            }
            Data_send_a[0] = gm[ttden];
   ZBTxRequest zbtx=ZBTxRequest (addr64_2, Data_send_a,
sizeof(Data_send_a));
            xbee.send(zbtx);
            stt_c = stt_m;
        }
    }
    if ( tt_c != tt_m)
    {

            if ( stt_m == 0)
            {
                if( !tt_m == 0) ttden=0;
                if( !tt_m == 1) ttden=1;
            }
            if ( stt_m == 1)
            {
                if ( !tt_m == 0) ttden=2;
                if ( !tt_m == 1) ttden=3;
            }
            Data_send_a[0] = gm[ttden];
   ZBTxRequest zbtx=ZBTxRequest(addr64_2,Data_send_a,
sizeof(Data_send_a));
            xbee.send(zbtx);
            tt_c = tt_m;
    }

}
//----------------------------------------


void loop() {
    void doc();
    void gui();
}
```

Bộ điều khiển đèn 3

```
#include <XBee.h>       // thư viện Xbee
#define COOR 0x419B5210 // địa chỉ XBee trung tâm
#define D1 0x409B9150 // địa chỉ XBee đèn 1
#define D2 0x40E74427 // địa chỉ XBee đèn 2
#define D3 0x418F1397 // địa chỉ XBee đèn 3
#define DCCOOR "1084889937" //địa chỉ dạng thập phân XBee trung tâm
#define DCD1 "1083937104" // địa chỉ dạng thập phân XBee đèn 1
#define DCD2 "1088898087" // địa chỉ dạng thập phân XBee đèn 2
#define DCD3 "1099895703" // địa chỉ dạng thập phân XBee đèn 3

// chọn chân triac, chân cảm biến
unsigned char AC_LOAD = 3;
int cb = 7;

// khởi tạo XBee
```

```
XBee xbee = XBee();
XBeeResponse response = XBeeResponse();

// khai báo địa chỉ các module Xbee board đèn và cảm biến
XBeeAddress64 addr64_1 = XBeeAddress64(0x0013A200, COOR);
XBeeAddress64 addr64_2 = XBeeAddress64(0x0013A200, D1);
XBeeAddress64 addr64_3 = XBeeAddress64(0x0013A200, D2);
XBeeAddress64 addr64_4 = XBeeAddress64(0x0013A200, D3);

// khởi tạo phản hồi XBee
ZBRxResponse rx = ZBRxResponse();
ZBRxIoSampleResponse ioSample = ZBRxIoSampleResponse();

// khai báo biến
unsigned int dimtime , dim = 0 , dimming_cu,dlieu, tt_m, tt_c,
ttden;
int value_m, value_c;
const char
gm[10]={0x30,0x31,0x32,0x33,0x34,0x35,0x36,0x37,0x38,0x39};
String Data_2; // du lieu nhan duoc tu xbee
String Address, diachi_s;
uint8_t Data_send_a[1] = {0x00};
//----------------------------------------

void setup()
{
  pinMode(AC_LOAD, OUTPUT);
  pinMode(cb,INPUT);
  attachInterrupt(0, zero_crosss_int, RISING);  //chân ngắt nối vào
chân số 2
  Serial.begin(9600);
  xbee.setSerial(Serial);
}
//----------------------------------------

void zero_crosss_int()  // hàm ngắt để thực hiện kích TRIAC
{
  if ( dim == 0)  detachInterrupt(0);
  if (dim == 1) dimtime = 1;
  if (dim > 1) dimtime = 2200;
  delayMicroseconds(dimtime);    // thời gian off TRIAC
  digitalWrite(AC_LOAD, HIGH);   // kích TRIAC
  delayMicroseconds(10);         // trì hoãn để đảm bảo đã kích
TRIAC
  digitalWrite(AC_LOAD, LOW);    // ngắt xung kích
}
Voi doc()
{
xbee.readPacket();
    if (xbee.getResponse().isAvailable())
    {
      if (xbee.getResponse().getApiId() == ZB_RX_RESPONSE)
        {
            xbee.getResponse().getZBRxResponse(rx);
            XBeeAddress64 senderLongAddress =
rx.getRemoteAddress64();
```

```
            Address = String(senderLongAddress.getLsb());//Lay gia
tri LSB
             Data_2="";
             for (int i= 0; i < rx.getDataLength(); i++)
                 {
               if (!iscntrl(rx.getData()[i]))
Data_2=Data_2+String(char(rx.getData()[i]));
                 }
              dlieu = Data_2.toInt();
         }
     }
}
//----------------------------------------

void gui()
{
if (( Address == DCD2)||(Address == DCD1))
     {
       if ( dim == 0)  detachInterrupt(0);
         else  attachInterrupt(0, zero_crosss_int, RISING);
         dim = dlieu;
     }
     value_m = digitalRead(cb);
     if ( value_m != value_c)
     {
         Data_send_a[0] = gm[!value_m];
   ZBTxRequest
zbtx=ZBTxRequest(addr64_3,Data_send_a,sizeof(Data_send_a));
               xbee.send(zbtx);
         value_c = value_m;
     }
}
//----------------------------------------

void loop() {
   doc();
   gui();

}
```


TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH
www.hcmute.edu.vn