

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM
KHOA CƠ KHÍ CHẾ TẠO MÁY

-----♦♦♦♦♦-----



ĐỒ ÁN MÔN HỌC HỆ THỐNG CƠ ĐIỆN TỬ

Đề tài:

**CÂN BẰNG CƯỜNG ĐỘ SÁNG CHO KHÔNG GIAN
PHÒNG HỌC**

GVHD: ThS. Nguyễn Vũ Lan

SVTH:	Võ Văn Đoàn	17146256
	Trần Minh Toàn	17146343
	Vũ Văn Phiêu	17146306
Khoa:	2017-2021	

TP Hồ Chí Minh, tháng 06 năm 2021

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM
KHOA CƠ KHÍ CHẾ TẠO MÁY

-----♦♦♦♦♦-----



ĐỒ ÁN MÔN HỌC HỆ THỐNG CƠ ĐIỆN TỬ

Đề tài:

**CÂN BẰNG CƯỜNG ĐỘ SÁNG CHO KHÔNG GIAN
PHÒNG HỌC**

GVHD: ThS. Nguyễn Vũ Lan

SVTH:	Võ Văn Đoàn	17146256
	Trần Minh Toàn	17146343
	Vũ Văn Phiêu	17146306
Khoa:	2017-2021	

TP Hồ Chí Minh, tháng 06 năm 2021

NHIỆM VỤ ĐỒ ÁN MÔN HỌC HỆ THỐNG CƠ ĐIỆN TỬ

Giảng viên hướng dẫn: ThS Nguyễn Vũ Lan

Sinh viên thực hiện:

Võ Văn Đoàn 17146256
Trần Minh Toàn 17146343
Vũ Văn Phiêu 17146306

1. Tên đề tài: Cân bằng cường độ sáng cho không gian phòng học.

2. Các số liệu, tài liệu ban đầu:

Kích thước mô hình phòng học: $1000 \times 600 \times 300$ (Dài \times rộng \times cao).

3. Nội dung chính của đồ án:

- Thiết kế, thi công, lắp đặt mô hình phòng học.
- Thi công, lắp đặt hệ thống động cơ, hệ thống đèn chiếu sáng.
- Lắp đặt hệ thống cảm biến ánh sáng.
- Viết chương trình điều khiển trên VDK và phần mềm điều khiển trên PC.

4. Các sản phẩm dự kiến:

- Mô hình phòng học.
- Hệ thống điều khiển.
- Phần mềm điều khiển.

5. Ngày giao đồ án: 09/03/2021

6. Ngày nộp đồ án: 25/06/2021

7. Ngôn ngữ trình bày:

- Bản báo cáo: Tiếng Việt.
- Trình bày phản biện: Tiếng Việt.

TRƯỞNG KHOA
(Ký, ghi rõ họ tên)

TRƯỞNG BỘ MÔN
(Ký, ghi rõ họ tên)

GVHD
(Ký, ghi rõ họ tên)

LỜI CẢM ƠN

Tri thức chỉ khi được áp dụng vào thực tế thì việc chiếm lĩnh tri thức đó mới có giá trị. Điều này đặc biệt đúng trong các khối ngành Kỹ thuật, trong đó có ngành CNKT Cơ điện tử. Ở học kỳ trước, chúng em đã được kiểm tra, học hỏi, nghiên cứu thêm các kiến thức thi công một hệ thống cơ khí thực tế bằng đồ án Truyền động điều khiển. Kỳ này, chúng em tiếp tục được học hỏi, áp dụng các kiến thức đã được học, đồng thời kiểm tra năng lực của bản thân bằng cách thực hiện Đồ án môn học Hệ thống Cơ điện tử.

Lời đầu tiên, chúng em xin cảm ơn các quý thầy cô ở Khoa Cơ khí Chế tạo máy – Trường Đại học Sư Phạm Kỹ Thuật Thành phố Hồ Chí Minh, trong suốt những năm qua vẫn luôn tận tụy truyền đạt, giảng dạy vốn kiến thức, kinh nghiệm quý báu của mình cho các thế hệ sinh viên, trong đó có các sinh viên chuyên ngành CNKT Cơ điện tử như chúng em.

Để hoàn thành được đồ án này thì không thể thiếu một người giáo viên hướng dẫn trực tiếp tận tâm, giàu kinh nghiệm. Chúng em xin chân thành cảm ơn thầy giáo **Nguyễn Vũ Lan**, là giảng viên hướng dẫn của nhóm em. Nhờ sự giúp đỡ nhiệt tình, những đóng góp quan trọng và những lời phê bình sâu sắc của thầy, chúng em mới có thể hoàn thành tốt đồ án này. Em xin gửi đến thầy lời cảm ơn chân thành nhất. Kính chúc thầy luôn dồi dào sức khoẻ và công tác tốt.

Đề tài: “**Cân bằng cường độ sáng cho không gian phòng học**” được thực hiện vào học kỳ 2 năm học 2020 – 2021. Đề tài này ở cả trong nước và nước ngoài còn khá mới, bước đầu tìm tòi và nghiên cứu với kiến thức còn hạn chế và còn nhiều bỡ ngỡ, chúng em chắc chắn không thể tránh khỏi những thiếu sót. Để việc thực hiện Đồ án tốt nghiệp sau này cũng với đề tài này được thuận lợi, chúng em rất mong có được những ý kiến đóng góp của các quý Thầy cô trong Khoa Cơ khí – Chế tạo máy để đề tài được thực hiện đầy đủ và hiệu quả hơn.

Sinh viên thực hiện

Võ Văn Đoàn
Trần Minh Toàn
Vũ Văn Phiêu

Mục lục

NHIỆM VỤ ĐỒ ÁN MÔN HỌC HỆ THỐNG CƠ ĐIỆN TỬ	i
LỜI CẢM ƠN	ii
DANH MỤC TỪ VIẾT TẮT	v
DANH SÁCH BẢNG	vi
DANH SÁCH HÌNH VẼ	viii
1 GIỚI THIỆU	1
1.1 Lý do chọn đề tài	1
1.2 Đối tượng nghiên cứu	2
1.3 Mục tiêu nghiên cứu	3
1.4 Giới hạn đề tài	3
1.5 Phương pháp nghiên cứu	3
1.6 Phương pháp thiết kế chung	3
2 TỔNG QUAN	4
2.1 Các nghiên cứu có liên quan ngoài nước	4
2.2 Các nghiên cứu có liên quan trong nước	5
2.3 Hướng nghiên cứu và phát triển	6
3 PHÂN TÍCH ĐỀ TÀI VÀ PHƯƠNG ÁN THIẾT KẾ	8
3.1 Thông số thiết kế	8
3.2 Phương hướng và giải pháp thực hiện	8
3.2.1 Phương án 1	9
3.2.2 Phương án 2	9
3.2.3 Phương án 3	9
3.2.4 Lựa chọn phương án tối ưu:	11
3.3 Chọn động cơ	11
3.4 Lựa chọn các chi tiết cho mô hình	12
3.5 Tính toán lựa chọn đèn	17
3.5.1 Chọn độ rọi	17
3.5.2 Chọn loại đèn	17
3.5.3 Chọn phương án chiếu sáng và bộ đèn	18
3.5.4 Bố trí các đèn	18
3.5.5 Thực nghiệm và chọn loại đèn phù hợp	19

4 THI CÔNG HỆ THỐNG ĐIỆN	21
4.1 Các thiết bị điện	21
4.1.1 Tổng quan về các phần tử điện	21
4.1.2 Khối nguồn	21
4.1.3 Động cơ bước	22
4.1.4 Driver cho động cơ bước	23
4.1.5 Mạch cầu H đầy đủ L298N	25
4.1.6 Kit STM32F411 - Discovery	26
4.1.7 Module mở rộng giao tiếp I2C 8 kênh TCA9548A	26
4.1.8 PCA9685	27
4.1.9 LED công suất 3W	28
4.1.10 Cảm biến ánh sáng	28
4.2 Sơ đồ kết nối	29
4.2.1 Sơ đồ kết nối tổng quát	29
4.2.2 Sơ đồ kết nối đèn và module L298N	29
4.2.3 Sơ đồ kết nối Driver TB6600 với động cơ bước	29
5 ĐIỀU KHIỂN VÀ LẬP TRÌNH	34
5.1 Tổng quan về phần điều khiển	34
5.1.1 Sơ đồ khối hệ thống điều khiển	34
5.1.2 Khối điều khiển trung tâm	35
5.1.3 Khối cảm biến	35
5.1.4 Khối điều khiển vị trí các dàn đèn	36
5.1.5 Khối điều khiển độ sáng các bóng đèn	36
5.2 Lập trình các chức năng Manual	38
5.2.1 Lưu đồ giải thuật chương trình điều khiển	38
5.2.2 Đọc dữ liệu từ cảm biến	39
5.2.3 Điều khiển toạ độ các dàn đèn	51
5.2.4 Điều khiển độ sáng của các bóng đèn	56
5.2.5 Nhận tín hiệu từ máy tính và thực hiện các thao tác điều khiển	63
5.2.6 Phần mềm điều khiển giao diện trên máy tính	65
6 TỔNG KẾT, ĐÁNH GIÁ VÀ HƯỚNG PHÁT TRIỂN	79
6.1 Tổng kết, đánh giá	79
6.2 Hướng phát triển	80
TÀI LIỆU THAM KHẢO	81

DANH MỤC TỪ VIẾT TẮT

STT	Ký hiệu chữ viết tắt	Chữ viết đầy đủ
1	GUI	Graphical User Interface
2	VĐK	Vi điều khiển

Danh sách bảng

3.1	So sánh các phương án thiết kế cơ cấu truyền động	11
3.2	Bảng so sánh động cơ Step và động cơ DC Servo	11
3.3	Bảng thông số kỹ thuật của động cơ NEMA17	12
4.1	Thông số nguồn cấp cần thiết của các phần tử điện	22
4.2	Bảng so sánh driver TB6600 và A4988	24

Danh sách hình vẽ

1.1	Bảng cường độ ánh sáng cần thiết cho các phòng	1
1.2	Ảnh hưởng của ánh sáng tự nhiên tới cường độ ánh sáng trong phòng	2
2.1	Cân bằng ánh sáng trong nhà kính trồng rau	4
2.2	Chiếu sáng trong nhà kho	5
2.3	Chiếu sáng trong lớp học	6
3.1	Khung nhôm định hình	8
3.2	Phương án 1	9
3.3	Phương án 2	10
3.4	Phương án 3	10
3.5	Động cơ bước NEMA17	12
3.6	Nhôm định hình 20x20	13
3.7	Ke góc vuông	13
3.8	Gối đỡ trực	14
3.9	Bảng kích thước gối đỡ trực	14
3.10	Con trượt vuông	15
3.11	Gối đỡ vòng bi ngang	16
3.12	Puly trực 5mm 20 răng	16
3.13	Dây đai GT2	16
3.14	Mô phỏng bố trí đèn chiếu trong phòng học sử dụng phần mềm DIALux	18
3.15	Đèn LED công suất 10W	19
3.16	Đèn LED công suất 5W	20
3.17	Đèn LED công suất 3W	20
4.1	Sơ đồ các khôi trong hệ thống điện	21
4.2	Nguồn tổ ong	22
4.3	Cầu tạo động cơ bước	23
4.4	Một số driver cho động cơ bước	23
4.5	Module L298N	25
4.6	Kit STM32F411 Discovery	26
4.7	Module TCA9548A	27
4.8	Module PCA9685	27
4.9	LED công suất 3W	28
4.10	Cảm biến ánh sáng GY-30 BH1750	29

4.11	Sơ đồ nguyên lý tổng quát	30
4.12	Sơ đồ kết nối điện thực tế	31
4.13	Sơ đồ nguyên lý kết nối đèn với module L298N	32
4.14	Sơ đồ kết nối thực tế đèn với module L298N	32
4.15	Sơ đồ nguyên lý kết nối Driver và động cơ	33
4.16	Sơ đồ kết nối thực tế Driver và động cơ	33
5.1	Sơ đồ khôi hệ thống điều khiển	34
5.2	Khôi điều khiển trung tâm	35
5.3	Khôi cảm biến	35
5.4	Khôi điều khiển vị trí các dàn đèn	36
5.5	Khôi điều khiển độ sáng các bóng đèn	37
5.6	Lưu đồ giải thuật chương trình chính	38
5.7	Lưu đồ giải thuật chương trình ngắn	39
5.8	Địa chỉ I2C của module TCA9548A	39
5.9	Quy cách gửi data cho module TCA9548A	40
5.10	Quy cách đọc data từ module TCA9548A	40
5.11	Bảng định nghĩa hàm dựa trên Control Register của TCA9548A	40
5.12	Gửi data cho module BH1750	44
5.13	Đọc data từ module BH1750	44
5.14	Bảng mã điều khiển của module BH1750	45
5.15	Địa chỉ của module PCA9685	57
5.16	Gửi data cho module PCA9685	57
5.17	Giao diện GUI của ứng dụng điều khiển	77

Chương 1

GIỚI THIỆU

1.1 Lý do chọn đề tài

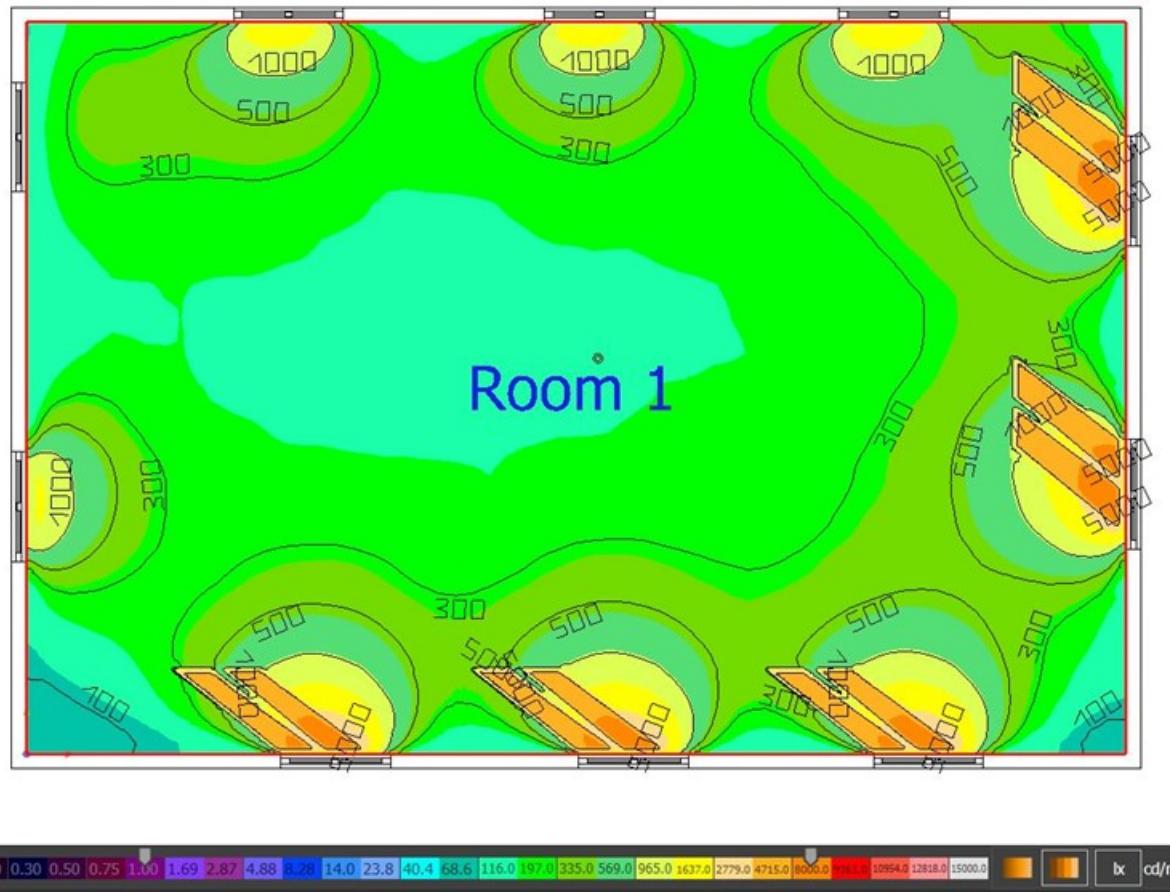
Trong hầu hết các hoạt động làm việc, sinh hoạt hằng ngày của con người, luôn cần một lượng ánh sáng nhất định. Tuỳ từng nơi sinh hoạt là phòng khách, phòng ăn (nhà ở) hay là phòng họp, nhà kho (công ty) mà chúng ta sẽ có mức cường độ ánh sáng cần thiết là khác nhau. Có thể thấy trong hình bên, trong nhà bếp, phòng ăn chúng ta chỉ cần 54 lux (đơn vị đo cường độ ánh sáng) còn trong khi đọc sách, chúng ta cần 431 lux. Hiện nay, gần như tất cả các phương án chiếu sáng đều đang sử dụng đèn cố định, tức là không thể thay đổi độ sáng lân góc chiếu, vị trí chiếu. Việc này dẫn tới hai nhược điểm sau đây:

Thứ nhất, vào ban ngày khi có ánh sáng tự nhiên, để tránh tình trạng hao phí điện năng thì chúng ta sẽ tắt đèn. Nhưng thường chỉ có những nơi gần cửa sổ, cửa ra vào mới được nhận lượng ánh sáng đầy đủ. Còn ở những nơi xa các nguồn sáng tự nhiên như góc phòng, giữa lớp sẽ có hiện tượng ánh sáng không đủ. Đối với các học sinh sinh viên khi học tập lâu dài trong điều kiện này sẽ dẫn tới các bệnh, tật về mắt.

Thứ hai, thường hệ thống đèn chiếu trong phòng học dùng một vài công tắc để bật tắt toàn bộ đèn trong phòng. Nếu chúng ta bật đèn để khắc phục tình trạng trên thì lại dẫn đến một nhược điểm khác, đó là những nơi có ánh sáng tự nhiên sẽ dư thừa ánh sáng không cần thiết. Khi thiết kế hệ thống chiếu sáng, người thiết kế đã đảm bảo hệ thống đèn sẽ chiếu đủ ánh sáng cần thiết trong điều kiện ban đêm. Khi kết hợp với

Activity	Lux	Footcandles
Direct Sunlight	32000-100000	2300 - 9300 (approx)
Daylight (not sun)	10000-25000	930 - 2300 (approx)
Full moon (clear)	1	0.1
Kitchen ambient	108	10
Kitchen task	538	50
Dining	54	5
Living Space	54	5
Living Space (task)	323	30
Desk lighting	431	40
Bedroom ambient	54	5
Bedroom reading	431	40
Bedroom dressing table		
Bathroom ambient	54	5

Hình 1.1: Bảng cường độ ánh sáng cần thiết cho các phòng



Hình 1.2: Ảnh hưởng của ánh sáng tự nhiên tới cường độ ánh sáng trong phòng

ánh sáng ban ngày thì sẽ bị dư thừa, gây hao phí điện năng không cần thiết. Hiện tại, thế giới đang đối mặt với tình trạng nóng lên toàn cầu, việc cắt giảm lượng dư thừa này được áp dụng trên phạm vi rộng chắc chắn sẽ cải thiện một phần nhỏ hiện tượng này. Ngoài ra, việc cắt giảm lượng ánh sáng dư thừa sẽ giúp cho các nhà trường tiết kiệm một khoản tiền điện, điều này có lợi cho ngân sách nhà nước đối với các trường công lập và giảm bớt chi phí phải bỏ ra trong chiếu sáng cho các trường tư, tự chủ.

Do đó, nhóm đã chọn đề tài “Cân bằng cường độ ánh sáng trong không gian phòng học” với mong muốn có thể cân bằng ánh sáng đều trên cả phòng học, giúp đảm bảo lượng ánh sáng cần thiết trong hoạt động học tập của học sinh, sinh viên và góp phần tiết kiệm điện năng.

1.2 Đối tượng nghiên cứu

Đối tượng nghiên cứu của đề tài “Cân bằng cường độ ánh sáng trong không gian phòng học” là: tập trung tìm hiểu, nghiên cứu các vấn đề liên quan, thiết kế và thi công mô hình điều khiển, đưa ra các giải thuật để cân bằng ánh sáng cho không gian

trong phòng học.

1.3 Mục tiêu nghiên cứu

- Đưa ra được các giải thuật cân bằng cường độ ánh sáng trong phòng học.
- Thực hiện được mô hình phòng học được cân bằng cường độ ánh sáng.

1.4 Giới hạn đề tài

Đề tài thực hiện nghiên cứu, thiết kế và thi công mô hình thử nghiệm ở mức độ:

- Điều khiển cường độ ánh sáng thông qua màn hình máy tính, nút nhấn.
- Cân bằng bằng cách điều khiển cường độ sáng của đèn hoặc tịnh tiến đèn để đạt được cường độ sáng cần thiết ở mọi điểm trong phòng.

1.5 Phương pháp nghiên cứu

Các vấn đề cần nghiên cứu trong đề tài:

- Phần cứng: Lắp ráp khung mô hình, bộ phận trượt cho dàn đèn, thi công đi dây cho mạch điện, driver điều khiển động cơ step, cảm biến và đèn công suất.
- Phần mềm: Phần mềm giao tiếp với VDK từ máy tính bằng giao tiếp UART, chương trình điều khiển hệ thống của VDK.

Phương pháp nghiên cứu:

- Tìm hiểu tài liệu: tìm hiểu các tài liệu về cường độ sáng, thiết kế chiếu sáng, các phương pháp điều khiển.
- Tự nghiên cứu: thông qua các tài liệu và các phương tiện thông tin, nhóm tự xây dựng giải thuật, cách xây dựng chương trình.
- Phương pháp thực nghiệm, thử và thử sai: thực nghiệm điều khiển cân bằng cường độ sáng, qua đó rút ra được kinh nghiệm.

1.6 Phương pháp thiết kế chung

- Sản phẩm: Lập trình điều khiển để cân bằng cường độ ánh sáng trong phòng.
- Năng suất: Mô hình có thể điều khiển cân bằng ánh sáng với thời gian ngắn và độ chính xác cao.
- Các thiết kế cơ khí được thiết kế phù hợp với phòng học thực tế.

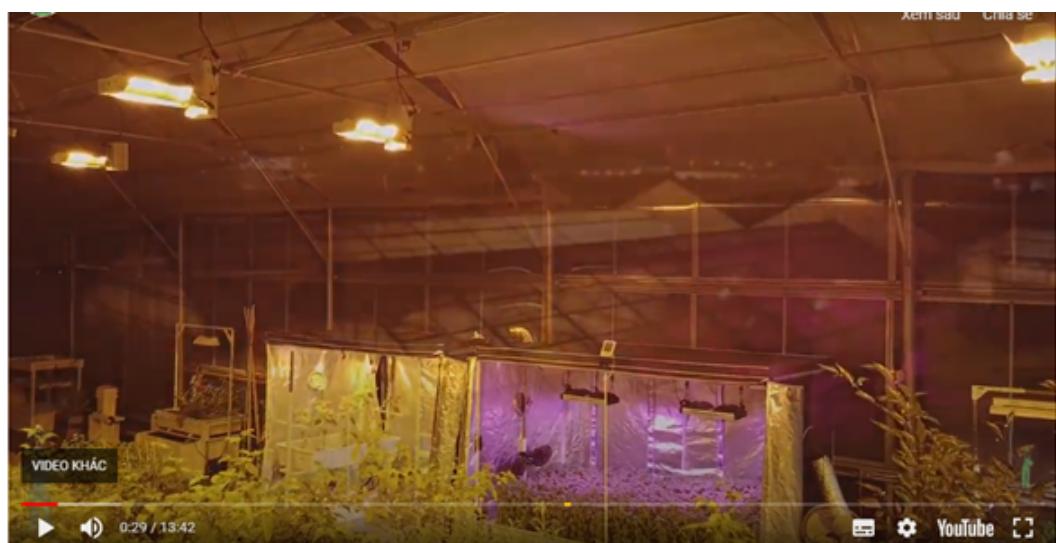
Chương 2

TỔNG QUAN

2.1 Các nghiên cứu có liên quan ngoài nước

Hiện nay tại các nước phát triển thì chiếu sáng nhân tạo là một vấn đề được rất nhiều công ty, tổ chức quan tâm và nghiên cứu. Chiếu sáng nhân tạo phục vụ các nhu cầu khác nhau: như trồng cây trong nhà kính, chiếu sáng các văn phòng, các phòng học... Tuy phục vụ các nhu cầu khác nhau nhưng điểm chung giữa các nghiên cứu ấy đều tập trung điều chỉnh cường độ ánh sáng và độ rọi lên bề mặt cần chiếu sáng.

Tại Farmertyler, một công ty tư vấn nông nghiệp chuyên về nông nghiệp đô thị, thủy canh, canh tác thẳng đứng và sản xuất nhà kính, ánh sáng được điều chỉnh và lựa chọn màu sắc phù hợp để kích thích và làm cho cây trồng phát triển và sinh trưởng nhanh. (Hình 2.1)



LIGHT INTENSITY

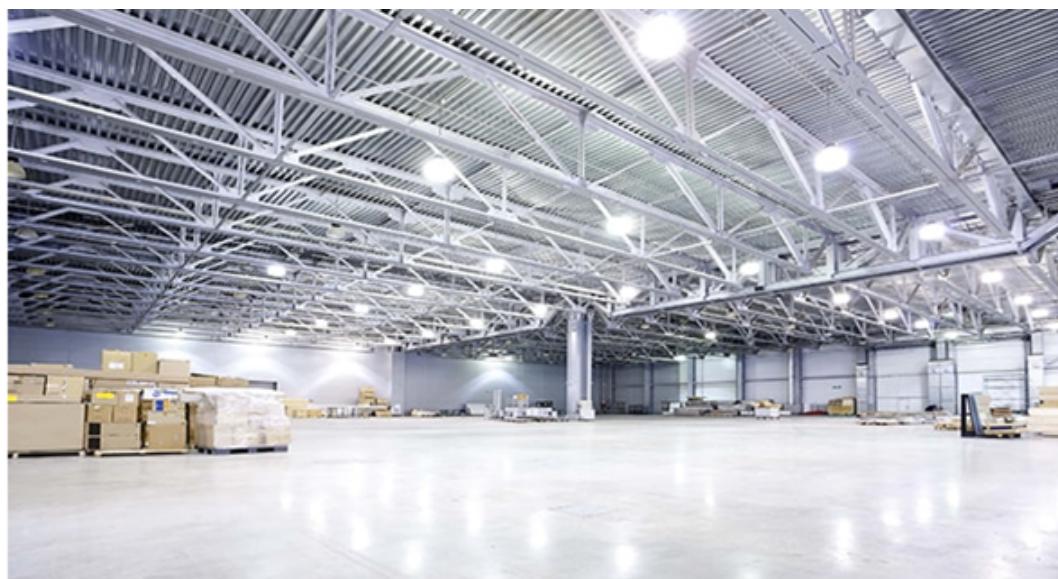
Hình 2.1: Cân bằng ánh sáng trong nhà kính trồng rau

Theo như bài nghiên cứu của tác giả: Sanaz Sanami thuộc Payame Noor University với bài nghiên cứu: “The Impact of Indoor Lighting on Students’ Learning Performance in Learning”. Bài nghiên cứu chỉ ra cho chúng ta thấy tác động của chiếu sáng trong phòng ảnh hưởng đến hiệu quả học tập của học sinh.

Hay như một bài nghiên cứu của Đại học Monash về ánh sáng nhân tạo tới nhịp sinh học thì PGS Cain cho biết: “Sự cân bằng âm và dương cũng là sự cân bằng giữa ánh sáng và bóng tối. Và sự cân bằng này rất quan trọng đối với việc điều tiết hệ thống sinh lý học của chúng ta. Ánh sáng nhân tạo phá vỡ nhịp hoạt động này, làm rối loạn giấc ngủ, ảnh hưởng đến sức khỏe và là nguyên nhân dẫn đến nhiều bệnh mãn tính hơn”.

2.2 Các nghiên cứu có liên quan trong nước

Hiện nay trong nước thì vấn đề về chiếu sáng trong phòng cũng chưa được quan tâm nhiều. Ở các phòng học và các công xưởng thì vấn đề chiếu sáng chỉ dừng ở mức độ chiếu đủ độ sáng. (Hình 2.2)



Hình 2.2: Chiếu sáng trong nhà kho

Ở các phòng công nghiệp thì ánh sáng được lắp và phân bố đều nhưng chỉ đảm bảo độ sáng chứ chưa có chuyên sâu về mức độ đảm bảo sức khỏe.

Công ty FSIVN, chuyên cung cấp các giải pháp thiết kế chiếu sáng trường học. Hình dưới là giải pháp của công ty cho một phòng học với ánh sáng trải đều, đèn điện có máng che đèn điện lắp so le và không có loáng của quạt. (Hình 2.3)

Từ hình ảnh trên chúng ta cũng thấy được ưu điểm của phương pháp này là ánh sáng của căn phòng luôn đạt được độ sáng chênh lệch và dao động từ 300-500 lux.

Nhưng nhược điểm của nó là có thể thấy vào những ngày thời tiết khác nhau cường độ ánh sáng bên ngoài tác động vào thì ánh sáng trong phòng sẽ khác, có nơi cường độ ánh sáng cao hơn mức cho phép và cũng có những nơi không đạt được độ sáng cần thiết.



Hình 2.3: Chiếu sáng trong lớp học

2.3 Hướng nghiên cứu và phát triển

Từ các bài nghiên cứu ở ngoài nước và trong nước thì chúng ta có thể thấy tầm quan trọng của ánh sáng nhân tạo hiện nay rất quan trọng. Tác động tích cực và tiêu cực của nó đối với con người và sinh vật rất lớn.

Nhận thấy tính quan trọng và cấp thiết của vấn đề. Nhóm chúng em lựa chọn nghiên cứu “Hệ thống cân bằng ánh sáng trong phòng kín” với mục đích là đề ra là tạo ra hệ thống cung cấp đủ ánh sáng trong phòng, cung cấp ánh sáng với độ rọi phù hợp. Hệ thống được xây dựng với định hướng nghiên cứu:

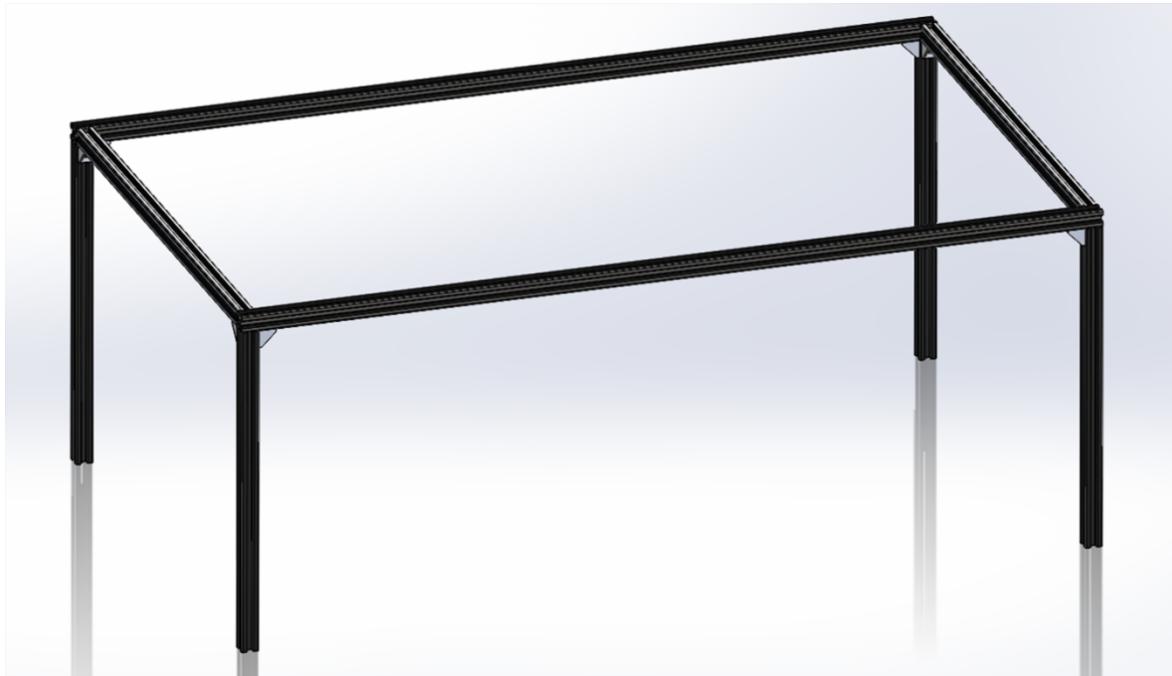
- Cung cấp ánh sáng trải đều trên diện tích chiếu sáng.
- Hệ thống tự điều chỉnh được bố cục đèn.
- Hệ thống tự đo và điều chỉnh cường độ ánh sáng khi có tác động ánh sáng khác từ bên ngoài.

- Báo cáo liên tục cho người dùng độ sáng của đèn.
- Đưa ra khuyến nghị khi ánh sáng trong phòng thay đổi.

Chương 3

PHÂN TÍCH ĐỀ TÀI VÀ PHƯƠNG ÁN THIẾT KẾ

3.1 Thông số thiết kế



Hình 3.1: Khung nhôm định hình

Kích thước khung: 1000 x 600 x 400 mm (dài x rộng x cao).

3.2 Phương hướng và giải pháp thực hiện

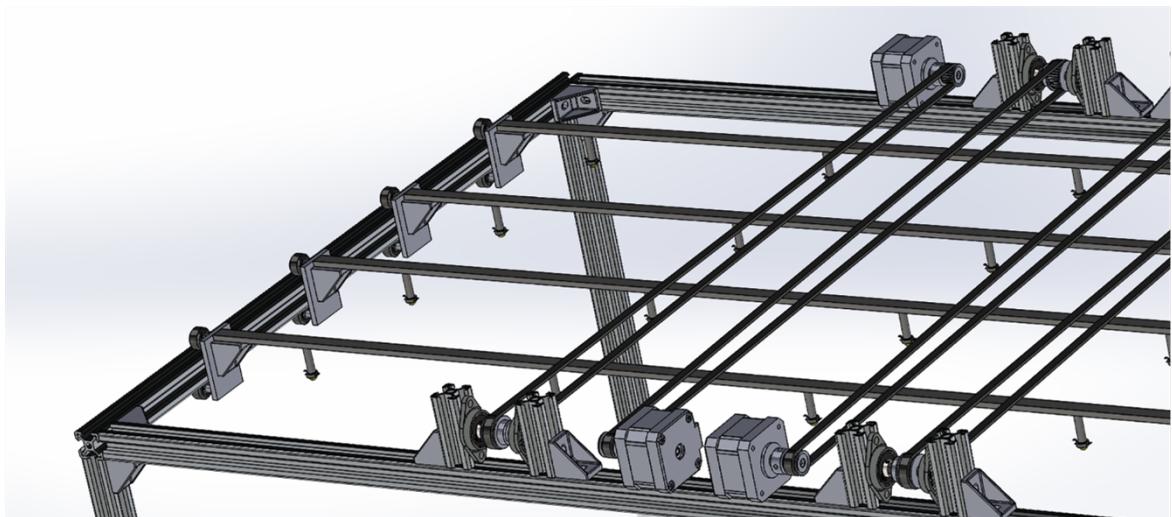
Để thiết kế được hệ thống đèn có thể điều chỉnh để nhằm ứng cường độ sáng tại các điểm, ta cần giải quyết được các vấn đề:

- Tịnh tiến được các dàn đèn
- Điều khiển dễ dàng, chính xác.
- Không gây tiếng ồn đáng kể.

Dựa theo đó, ta có các phương án sau:

3.2.1 Phương án 1

Ta tịnh tiến các dàn đèn bằng chuyển động trượt, dùng các con lăn nhôm V-slot trượt trên nhôm (Hình 3.2).



Hình 3.2: Phương án 1

3.2.2 Phương án 2

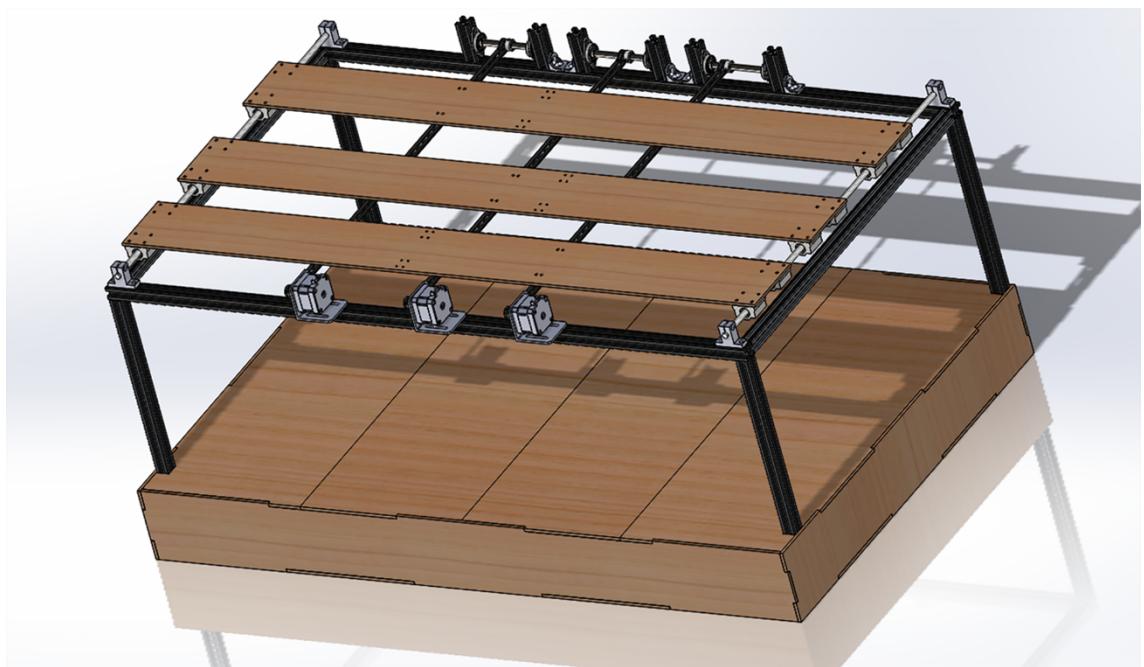
Ta tịnh tiến các dàn đèn bằng chuyển động trượt, dùng 2 con trượt vuông chuyển động trên ti trượt (Hình 3.3).

3.2.3 Phương án 3

Ta tịnh tiến các dàn đèn bằng chuyển động trượt, dùng 2 con trượt vuông chuyển động trên ti trượt (Hình 3.4).



Hình 3.3: Phương án 2



Hình 3.4: Phương án 3

3.2.4 Lựa chọn phương án tối ưu:

Đặc điểm	Phương án 1	Phương án 2	Phương án 3
Ôn định	+	-	+
Chi phí	-	+	+
Chính xác	-	-	+
Cơ cấu đơn giản	-	+	+
Khả năng lắp ráp, thay thế	+	+	+
Không gian hoạt động	+	+	-
Tổng	3+	4+	5+

Bảng 3.1: So sánh các phương án thiết kế cơ cấu truyền động

Sau khi so sánh ưu nhược điểm của từng phương án, nhóm quyết định lựa chọn phương án 3. (Bảng 3.1)

3.3 Chọn động cơ

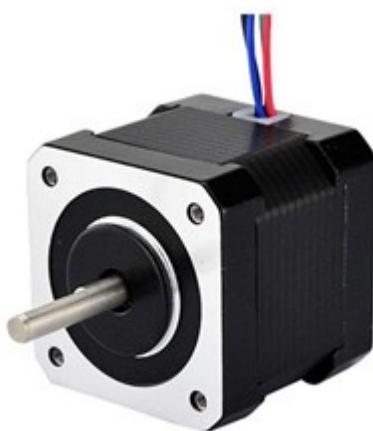
	Động cơ STEP	Động cơ DC Servo
Ưu điểm	<ul style="list-style-type: none"> - Tạo ra chuyển động gia tăng, giảm chi phí và độ phức tạp của bộ mã hóa. - Số cực cao nên có thể tạo ra mô-men xoắn rất cao ở tốc độ 0. - Kích thước nhỏ gọn, tiết kiệm chi phí. 	<ul style="list-style-type: none"> - Là hệ thống hồi tiếp vòng kín chính vì vậy động cơ servo DC rất dễ điều khiển, dễ sử dụng. - Mô-men xoắn khởi động lớn. - Có thể được sử dụng trong các ứng dụng công nghiệp và dân dụng nói chung nhạy cảm với chi phí.
Nhược điểm	<ul style="list-style-type: none"> - Tốc độ bị giới hạn, chạy tốt nhất ở 1200 vòng/phút hoặc thấp hơn. - Mặc dù tạo mô-men xoắn cao ở tốc độ 0, nhưng mô-men xoắn sẽ rời khi tốc độ tăng. 	<ul style="list-style-type: none"> - Vì cấu tạo có bộ phận chổi than nên điểm hạn chế lớn nhất của loại động cơ này chính là dễ gây ra tiếng ồn, nhiệt độ cao. - Quán tính cao khi giảm tốc độ. - Bảo trì bất tiện (thay thế chổi than).

Bảng 3.2: Bảng so sánh động cơ Step và động cơ DC Servo

Trong kỹ thuật không có giải pháp nào hoàn hảo, mà chỉ được coi là giải pháp tốt nhất, đặc biệt đối với động cơ SERVO và động cơ STEP. Cả hai đều được sử dụng rộng rãi trong ngành công nghiệp. Khi được áp dụng đúng cách, hai động cơ này đều có thể mang lại hiệu quả làm việc đáng mong đợi cho cả hệ thống máy. Để đi tới quyết định lựa chọn loại động cơ nào phù hợp thì phụ thuộc vào rất nhiều yếu tố, nhưng quan trọng nhất là tốc độ, giá tốc và giá cả.

Vì trong đề tài này, nhóm thực hiện mô hình với kích thước nhỏ, với mục đích điều khiển dễ dàng, chính xác và không gây tiếng ồn với chi phí thấp, dễ thay thế, bảo trì nên qua việc phân tích những ưu điểm và nhược điểm của hai loại động cơ này (Bảng 3.2), ta thấy động cơ STEP sẽ phù hợp với mô hình hơn.

Động cơ bước NEMA17: (Hình 3.5)



Hình 3.5: Động cơ bước NEMA17

Động cơ bước	Kích thước motor (mm)	Bước góc (°)	Cường độ dòng điện (A)	Điện trở pha (Ω)	Moment xoắn (Nmm)	Moment quán tính ($g.cm^2$)	Trọng lượng (g)
NEMA 17	42x42	1.8	1.7	5.75	280	100	300

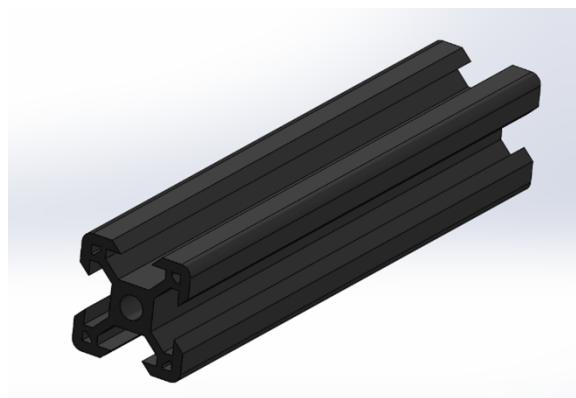
Bảng 3.3: Bảng thông số kỹ thuật của động cơ NEMA17

3.4 Lựa chọn các chi tiết cho mô hình

Nhôm định hình: nhóm đã sử dụng nhôm định hình 20 x 20 để làm khung. (Hình 3.6)

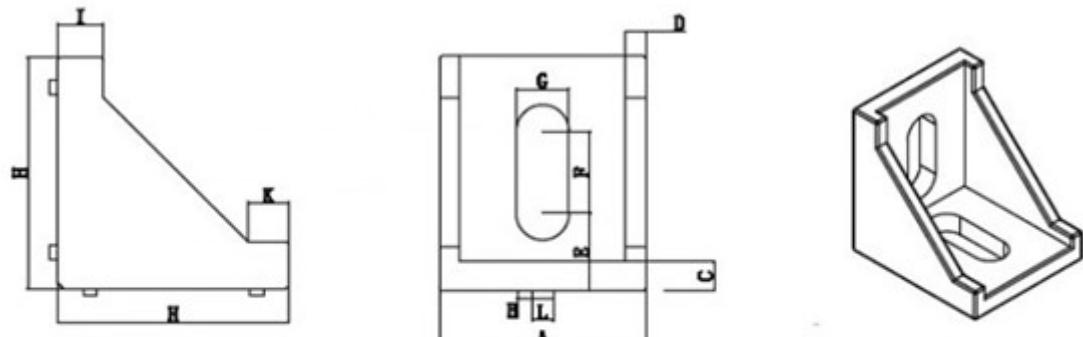
Ưu điểm:

- Có kết cấu vững chắc, bền vững với thời gian.
- Không bị cong vênh, không bị oxi hóa, không biến đổi do nhiệt.
- Ưu điểm vượt trội về khối lượng cũng như không bị mối mọt như gỗ.
- Được sơn tĩnh điện, không bị bong sơn, tính thẩm mỹ cao.



Hình 3.6: Nhôm định hình 20x20

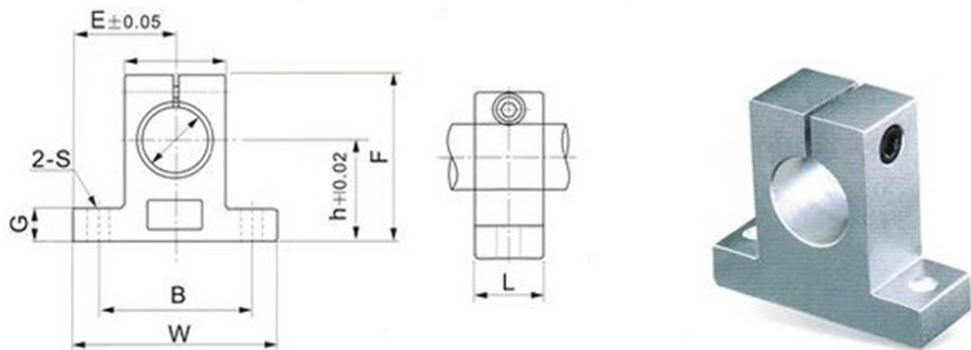
Ke góc vuông: Ke sử dụng để cố định và gá các thanh nhôm tạo khung. (Hình 3.7)



		A	B	C	D	E	F	G	H	I	J	K	重量
1	2028	20	1.2	4	2.6	14	2	6.2	28	6	5	5.8	13g
2	3030	28	2	4.58	3	13.5	9	6.9	35	6.9	5	5.7	32g
3	4040	34.8	1.45	4.84	3.57	13.1	13.7	9.07	39.9	8.1	6.18	6.44	33g
4	4545	43	3	7.86	3.7	22.5	9	8.5	41.8	8	4.8	7.6	48g

Hình 3.7: Ke góc vuông

Gối đỡ trực: Gối đỡ thanh trượt tròn 8mm dùng đỡ 2 đầu cho các thanh trượt tròn. (Hình 3.8)

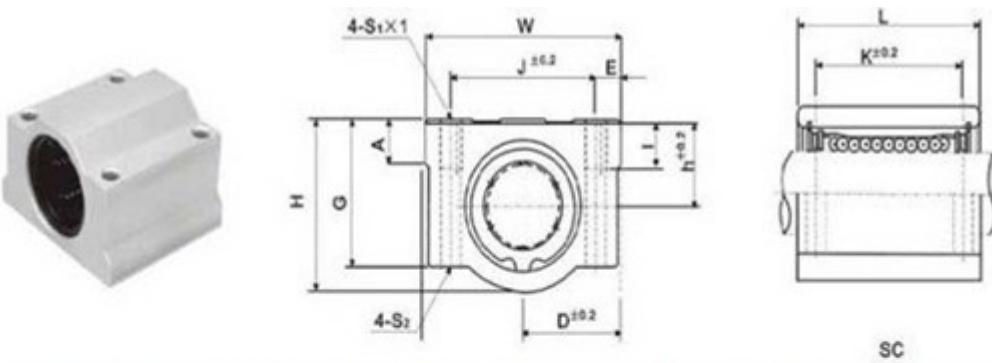


Hình 3.8: Gối đỡ trục

支撑型号 Support Designation	轴承尺寸 Shaft Dimensions	尺寸 Dimensions (mm)								锁紧螺栓 Locking Bolt	安全螺栓 Clamping Bolt	重量 Weight (g)
		h	E	W	F	G	P	B	S			
SH8A	8	20	21	42	32.8	6	18	32	5.5	M4	M5	24
SH10A	10	20	21	42	32.8	6	18	32	5.5	M4	M5	24
SH12A	12	23	21	42	38	6	20	32	5.5	M4	M5	30
SH13A	13	23	21	42	38	6	20	32	5.5	M4	M5	30
SH16A	16	27	24	48	44	8	25	38	5.5	M4	M5	40
SH20A	20	31	30	60	51	10	30	45	6.6	M5	M6	70
SH25A	25	35	35	70	60	12	38	56	6.6	M6	M6	130
SH30A	30	42	42	84	70	12	44	64	9	M6	M8	180
SH35A	35	50	49	98	85	15	50	74	11	M8	M10	270
SH40A	40	60	57	114	96	15	60	90	11	M8	M10	420
SH50A	50	70	63	126	120	18	74	100	14	M12	M12	750
SH60A	60	80	74	148	136	18	90	120	14	M12	M12	1100

Hình 3.9: Bảng kích thước gối đỡ trục

Con trượt vuông: cho phép các chi tiết máy móc chuyển động tinh tiến chính xác, mượt mà, nhẹ nhàng, êm ái, ít lực ma sát nhất. (Hình 3.10)



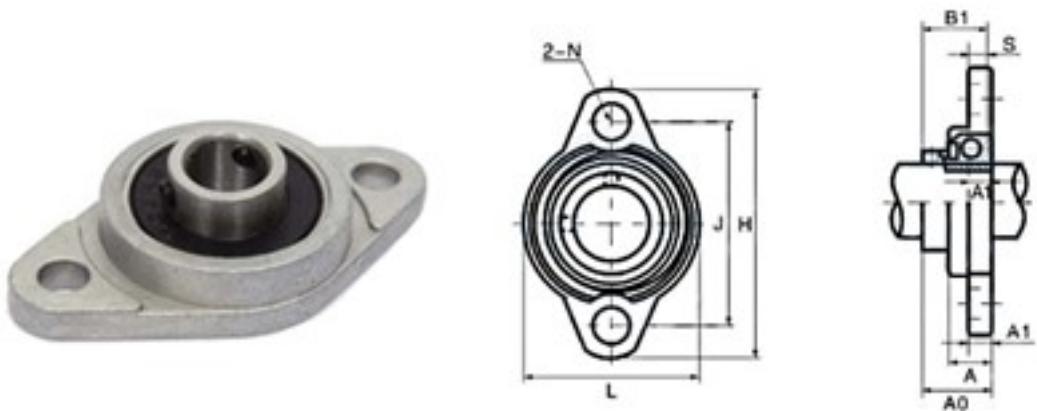
Model	Bearing ID(mm)	h(mm)	D(mm)	W(mm)	H(mm)	G(mm)	A(mm)	J(mm)	E(mm)	S1X1	S2	K(mm)	L(mm)
SC8UU	Φ8	11	17	34	22	18	6	24	5	M4X8	Φ3.4	18	30
SC10UU	Φ10	13	20	40	26	21	8	28	6	M5X10	Φ4.3	21	35
SC12UU	Φ12	15	22	44	30	24.5	8	33	5.5	M5X10	Φ4.3	26	39
SC13UU	Φ13	15	22	44	30	24.5	8	33	5.5	M5X10	Φ4.3	26	39
SC16UU	Φ16	19	25	50	38.5	32.5	9	36	7	M5X12	Φ4.3	34	44
SC20UU	Φ20	21	27	54	41	35	11	40	7	M6X12	Φ5.2	40	50
SC25UU	Φ25	26	38	76	51.5	41	12	54	11	M8X18	Φ6.8	50	67
SC30UU	Φ30	30	39	78	59.5	49	15	58	10	M8X18	Φ6.8	58	72
SC35UU	Φ35	34	45	90	68	54	18	70	10	M8X18	Φ6.8	60	80
SC40UU	Φ40	40	51	102	78	62	20	80	11	M10X25	Φ8.6	60	90
SC50UU	Φ50	52	61	122	102	80	24	100	11	M10X25	Φ8.6	60	110

Hình 3.10: Con trượt vuông

Gối đỡ vòng bi ngang: (Hình 3.11)

Pully: Pully có lỗ 5mm tương thích với đa số các động cơ bước, đặc biệt loại động cơ bước NEMA17. (Hình 3.12)

Dây đai GT2: sử dụng profin bo tròn, bước răng 2mm đảm bảo các răng của đai khít vừa vặn và chính xác. (Hình 3.13)



Hình 3.11: Gối đỡ vòng bi ngang



Hình 3.12: Puly trực 5mm 20 răng



Hình 3.13: Dây đai GT2

3.5 Tính toán lựa chọn đèn

Vấn đề chiếu sáng trong văn phòng, trong các phòng kín luôn là vấn đề được nghiên cứu và phát triển. Mục đích chiếu sáng tạo ra môi trường ánh sáng tốt, tiện nghi, làm cho người sử dụng cảm thấy dễ chịu. Ngoài ra ở các giảng đường, thư viện hay các văn phòng kín, việc chiếu sáng giúp cho người học hay người làm cảm thấy làm việc học tập có hiệu quả hơn.

Yêu cầu của đèn khi thiết kế cần đảm bảo:

- Đảm bảo được độ rọi để phục vụ cho công việc, việc học tập.
- Đảm bảo tiện nghi, không gây lóa mắt.
- Màu sắc và nhiệt độ trong phòng phù hợp.
- Vấn đề thẩm mỹ và tiết kiệm năng lượng.

3.5.1 Chọn độ rọi

Độ rọi của đèn thể hiện được bề mặt làm việc, nơi đó còn gọi là “bề mặt hữu ích” với mô hình nhóm lựa chọn có độ cao là 300mm so với mặt sàn.

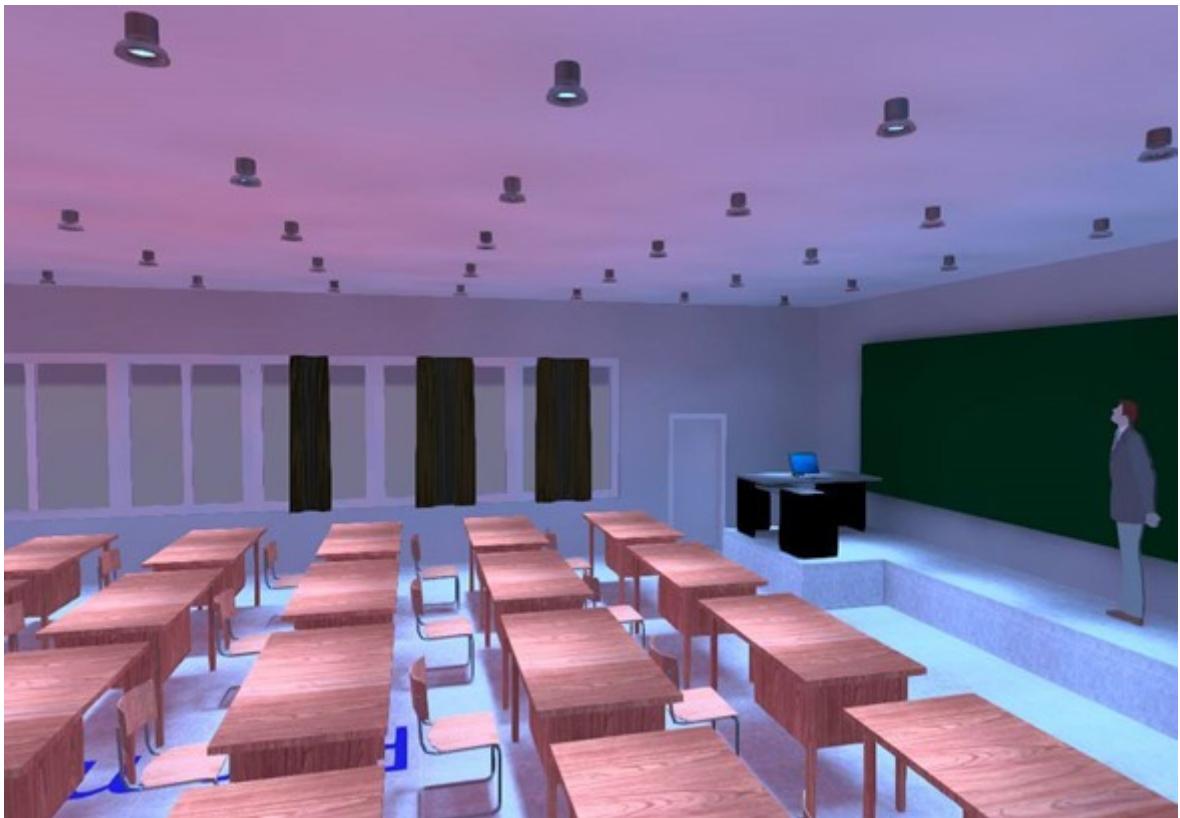
Độ rọi này phụ thuộc vào góc chiếu của đèn, của chóa đèn... ở cấp độ mô hình thì nhóm mong muốn là giữa các phần giao thoa của đèn thì cường độ ánh sáng được đảm bảo với cường độ ánh sáng mong muốn.

Hình 1.2 thể hiện ảnh hưởng của ánh sáng tự nhiên vào căn phòng, có thể thấy ở các vùng gần cửa sổ thì có cường độ ánh sáng cao nhưng ở nơi xa cửa sổ (trung tâm phòng học) có cường độ thấp hơn.

3.5.2 Chọn loại đèn

Việc lựa chọn đèn thích hợp nhất trong các loại đèn được nhóm triển khai theo các tiêu chuẩn sau đây:

- Góc chiếu của đèn đảm bảo được các vùng sáng giao thoa nhau.
- Đèn có cường độ ánh sáng có thể điều chỉnh phù hợp.
- Tuổi thọ của đèn.
- Hiệu quả ánh sáng của đèn.
- Chi phí.



Hình 3.14: Mô phỏng bố trí đèn chiếu trong phòng học sử dụng phần mềm DIALux

3.5.3 Chọn phương án chiếu sáng và bộ đèn

Trong cuộc sống chúng ta thường gặp những cách chiếu sáng trực tiếp và bán trực tiếp. Kiểu chiếu sáng phụ thuộc vào các yếu tố ngoại quan như ánh sáng môi trường, thời tiết, khí hậu, tính phản xạ,...

Đối với loại đèn khác nhau, thì nhà sản xuất đã cung cấp catalog cho phép chọn kiểu bộ đèn, cấp xác định và nếu có thể người sản xuất đảm bảo có đủ các loại công suất khác nhau.

3.5.4 Bố trí các đèn

Không gian của nhóm thực hiện là không gian hình chữ nhật nên nhóm lựa chọn phương án bố trí đèn theo từng dàn cách đều nhau.

Bằng cách tính tổng quan thông số của đèn thì ta tính được số đèn tương ứng để chiếu sáng. Vì thế số đèn chọn là nhỏ nhất nhưng bố trí đều đặn sao cho khi sử dụng thì đèn sẽ đồng đều và độ rọi sẽ tốt nhất.

3.5.5 Thực nghiệm và chọn loại đèn phù hợp

Sau khi tìm hiểu các yêu cầu mà bóng đèn cần có thì nhóm đã tiến hành mô phỏng tính toán để chọn ra các loại đèn tối ưu nhất đáp ứng các nhu cầu đề ra và tiến hành lắp ráp thực tế. Các loại đèn được nhóm lựa chọn: đèn công suất siêu sáng 3W, 5W, 10W.

Đèn LED công suất 10W: (Hình 3.15)

Đèn siêu sáng 10W cung cấp ánh sáng trắng với cường độ cao có thể thay đổi được, trong khi thực tiễn thì mặc dù đáp ứng được độ sáng cho mô hình nhưng vẫn còn tồn tại rất nhiều nhược điểm như:

- Bóng tỏa một lượng nhiệt rất lớn.
- Công suất tiêu thụ lớn.
- Giá thành của bóng cao.
- Việc điều khiển cường độ ánh sáng ở chế độ auto thì gấp nhiều lần trở do góc chiếu của đèn khá rộng.



Hình 3.15: Đèn LED công suất 10W

Đèn LED công suất 5W: (Hình 3.16)

Led 5W cung cấp ánh sáng trắng cho toàn bộ căn phòng, thỏa mãn các điều kiện cần của mô hình nhưng vẫn còn tồn tại là giá thành cao, và khi không có các yếu tố bên ngoài thì ánh sáng của led khi hoạt động ở trạng thái 100% thì vẫn vượt quá mức cần thiết rất nhiều.

Đèn LED công suất 3W: (Hình 3.17)

Led siêu sáng 3W cho cường độ ánh sáng với góc chiếu tính toán đạt yêu cầu. giá thành của đèn thì là rẻ nhất trong các loại đèn được nhóm đưa ra lựa chọn. Tuy còn điểm yếu chung là góc chiếu nhòm cần tìm và thiết kế chóa phù hợp với đèn.

Như vậy qua việc tìm hiểu và kiểm thử các loại đèn thì nhóm chọn được đèn phù hợp với đồ án của nhóm là led siêu sáng 3W với chóa đèn do nhóm tự thiết kế.



Hình 3.16: Đèn LED công suất 5W



Hình 3.17: Đèn LED công suất 3W

Chương 4

THI CÔNG HỆ THỐNG ĐIỆN

4.1 Các thiết bị điện

4.1.1 Tổng quan về các phần tử điện

Để hệ thống hoạt động được luôn cần phần điện. Hệ thống điện chịu trách nhiệm cung cấp nguồn điện, điều khiển các thiết bị trong kết cấu máy như động cơ, driver.



Hình 4.1: Sơ đồ các khía trong hệ thống điện

4.1.2 Khối nguồn

Khối nguồn là bộ phận cung cấp năng lượng cho toàn bộ hệ thống điện trong hệ thống. Để các bóng đèn và động cơ hoạt động ổn định nên nguồn cấp phải đảm bảo điều kiện điện áp và dòng điện luôn ổn định.

Ta lựa chọn nguồn tổ ong.

Để lựa chọn bộ nguồn phù hợp, phải chú ý đến các thiết bị sử dụng trong mạch điện. Dựa vào thông số về điện áp và dòng điện yêu cầu trên các linh kiện điện để có thể lựa chọn nguồn nuôi thích hợp. Dưới đây là một số linh kiện điện tử và điện áp yêu cầu của các linh kiện đó:



Hình 4.2: Nguồn tổ ong

Linh kiện	Số lượng	Thông số nguồn cấp
LED công suất 3W	12	4.2V
Driver TB6600	3	12V
Động cơ bước	3	12V
Module L298N	4	12V
TCA9548A	1	5V
PCA9685	1	5V

Bảng 4.1: Thông số nguồn cấp cần thiết của các phần tử điện

Các thiết bị điện trong máy có dải điện áp hoạt động từ $3.8V \sim 12V$.

Cường độ dòng điện cần thiết cho bóng đèn hoạt động: $0.789A \times 12 = 9.47(A)$

Cường độ dòng điện cần thiết cho 3 động cơ STEP: $0.5A \times 3 = 1.5(A)$

Tổng cường độ dòng điện định mức cần thiết để hệ thống hoạt động bình thường:

$$9.47A + 1.5A \approx 11A$$

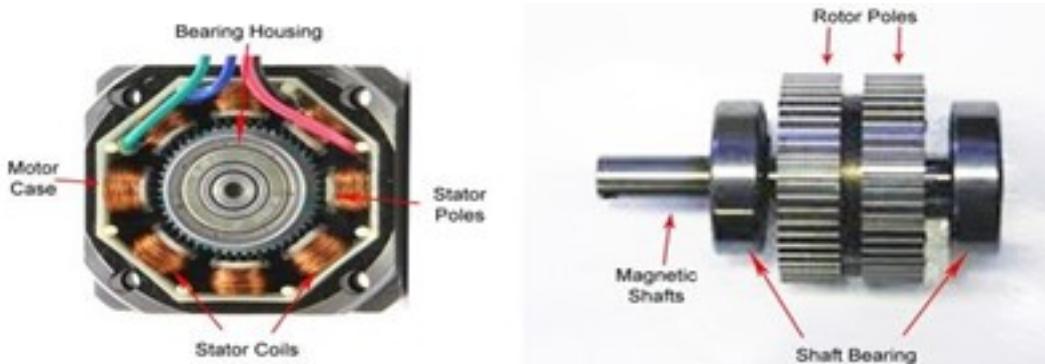
Bộ nguồn được chọn cần có điện áp cung cấp $12V$ và dòng điện định mức lớn hơn $11A$. Để thuận lợi cho việc nâng cấp hệ thống điện sau này và đảm bảo hệ thống điện hoạt động tốt nhất ta lựa chọn nguồn tổ ong $12V30A$

4.1.3 Động cơ bước

Động cơ bước (stepper motor) là một loại động cơ điện có nguyên lý và ứng dụng khác biệt với đa số các động cơ điện thông thường. Chúng thực chất là một động cơ đồng bộ dùng để biến đổi các tín hiệu điều khiển dưới dạng các xung điện rời rạc kế tiếp nhau thành các chuyển động góc quay hoặc các chuyển động của rôto có khả năng cố định rôto vào các vị trí cần thiết.

Động cơ bước không quay theo cơ chế thông thường, chúng quay theo từng bước nên có độ chính xác rất cao về mặt điều khiển. Chúng làm việc nhờ các bộ chuyển

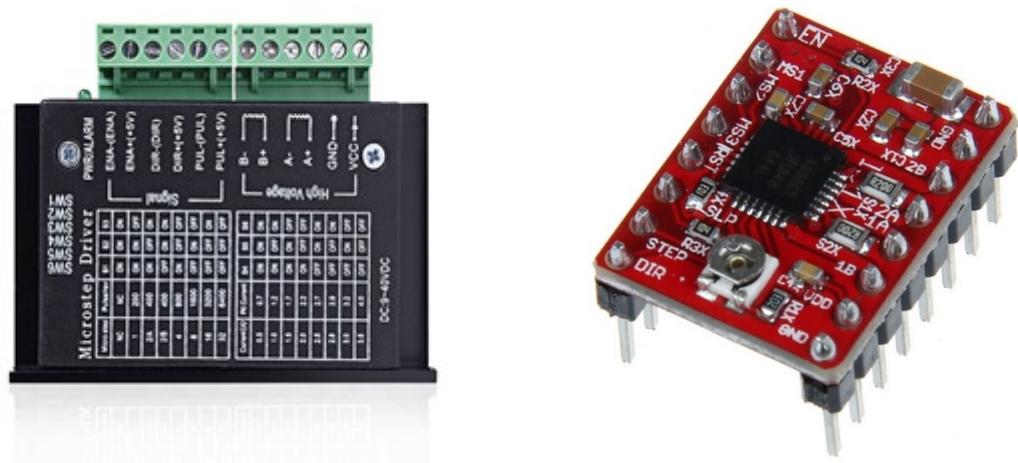
mạch điện tử đưa các tín hiệu điều khiển vào stator theo thứ tự và một tần số nhất định. Tổng số góc quay của rôto tương ứng với số lần chuyển mạch, cũng như chiều quay và tốc độ quay của rôto phụ thuộc vào thứ tự chuyển đổi và tần số chuyển đổi.



Hình 4.3: Cấu tạo động cơ bước

4.1.4 Driver cho động cơ bước

Một bộ phận không thể thiếu trong điều khiển động cơ bước đó là driver. Driver như là một mạch phân phối xung cho động cơ, làm nhiệm vụ cấp điện cho động cơ bước hoạt động. Có 2 loại driver được sử dụng khá nhiều trong thị trường hiện tại đó là : TB6600 và A4988.



(a) Driver TB6600

(b) Driver A4988

Hình 4.4: Một số driver cho động cơ bước

Từ Bảng 4.2, ta thấy tuy giá thành cao hơn, nhưng driver TB6600 có độ phân giải bước cao hơn, dòng điện điều khiển ngõ ra cũng lớn hơn đồng nghĩa với việc TB6600 sẽ điều khiển vị trí chính xác hơn và động cơ sẽ có momen lớn hơn. Vì thế, trong đồ án này ta sử dụng driver TB6600.

	TB6600	A4988
Dòng điện ngõ ra	0.5A – 4A	0.5 – 2A
Vị bước nhỏ nhất	1/64	1/16
Điện áp điều khiển	3.3V – 24V	0.3V – 5.5V
Nguồn cấp cho driver	9V – 42V	8V – 35V
Bảo vệ quá nhiệt	Có	Có
Bảo vệ ngắn mạch	Có	Không
Kích thước	96 × 71 × 37(mm)	5 × 5 × 0.9(mm)
Giá thành	140.000VND	25.000VND

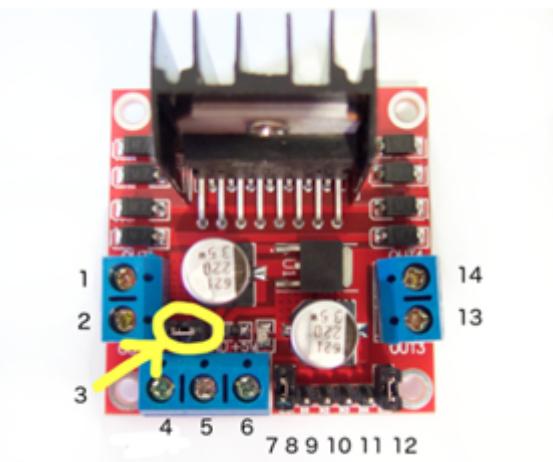
Bảng 4.2: Bảng so sánh driver TB6600 và A4988

Thông số kỹ thuật:

- IC Driver: TB6600HQ/HG Single-chip bipolar sinusoidal micro-step stepping motor driver, công nghệ mới nhất BiCD 0.13nm.
- Nguồn cấp tối đa: 40VDC.
- Dòng cấp $IIN_{MAX} = 5.0A$ và dòng ra $IOUT_{MAX} = 4$.
- Độ phân giải: 1/1, 1/2, 1/4, 1/8, 1/16, 1/32 và 1/64.
- Kích thước: 96 x 57 x 35mm.

Cách kết nối:

- DC+: Nối với nguồn điện từ 9 - 40VDC
- DC- : Điện áp (-) âm của nguồn
- A+ và A -: Nối vào cặp cuộn dây của động cơ bước
- B+ và B- : Nối với cặp cuộn dây còn lại của động cơ
- PUL+: Tín hiệu cấp xung điều khiển tốc độ (+5V) từ BOB cho M6600
- PUL-: Tín hiệu cấp xung điều khiển tốc độ (-) từ BOB cho M6600
- DIR+: Tín hiệu cấp xung đảo chiều (+5V) từ BOB cho M6600
- DIR-: Tín hiệu cấp xung đảo chiều (-) từ BOB cho M6600
- ENA+ và ENA -: khi cấp tín hiệu cho cặp này động cơ sẽ không có lực momen giữ và quay nữa
- Có thể đấu tín hiệu dương (+) chung hoặc tín hiệu âm (-) chung.



Hình 4.5: Module L298N

4.1.5 Mạch cầu H đầy đủ L298N

Thông số kỹ thuật:

- Driver: L298N tích hợp hai mạch cầu H
- Điện áp điều khiển: $+5V \sim +12V$
- Dòng tối đa cho mỗi cầu H là: 2A
- Điện áp của tín hiệu điều khiển: $+5V \sim +7V$
- Dòng của tín hiệu điều khiển : $0 \sim 36mA$
- Công suất hao phí : $20W$ (khi nhiệt độ $T = 75^\circ C$)
- Nhiệt độ bảo quản : $-25^\circ C \sim +130^\circ C$

Các chân tín hiệu quan trọng:

- 3: 12V jumper, tháo jumper ra nếu sử dụng nguồn trên 12V. Jumper này dùng để cấp nguồn cho IC ổn áp tạo ra nguồn 5V nếu nguồn trên 12V sẽ làm cháy IC nguồn.
- 4: Cắm dây nguồn cung cấp điện áp cho motor vào đây từ 6V đến 35V.
- 5: Cắm chân GND của nguồn vào đây.
- 6: Ngõ ra nguồn 5V, nếu jumper đầu vào không rút ra.
- 7: Chân Enable , chân này dùng để cấp xung PWM nếu dùng VDK thì rút jumper ra và cắm chân PWM vào đây.

- 8, 9, 10, 11: lần lượt là các chân IN1, IN2, IN3, IN4.
- 12: Chân Enable , chân này dùng để cấp xung PWM nếu dùng VDK thì rút jumper ra và cắm chân PWM vào đây.

4.1.6 Kit STM32F411 - Discovery

Thông số kỹ thuật:

- Bộ điều khiển STM32F411VET6 có 512KB bộ nhớ Flash, 128KB RAM trong gói LQFP100.
- Cấp nguồn : Thông qua bus USB hoặc cấp điện áp 5V từ bên ngoài.
- IC L3GD20: Cảm biến chuyển động ST MEMS 3-trục, đầu ra kĩ thuật số.
- IC LSM303DLHC: Hệ thống ST MEMS trong hệ thống bao gồm bộ cảm biến gia tốc tuyến tính kĩ thuật số 3D và cảm biến từ tính số 3 chiều
- IC MP45DT02: Bộ cảm biến âm thanh ST MEMS, micro kĩ thuật số.
- IC CS43L22 : âm thanh DAC với trình điều khiển loa lớp tích hợp D.



Hình 4.6: Kit STM32F411 Discovery

4.1.7 Module mở rộng giao tiếp I2C 8 kênh TCA9548A

Thông số kỹ thuật:

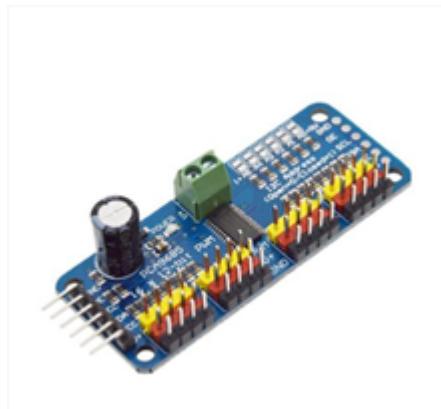
- Điện áp hoạt động: 1.65 – 5.5VDC
- Số kênh mở rộng: 8 kênh



Hình 4.7: Module TCA9548A

- Giao tiếp I2C, địa chỉ chọn được từ 0x70 đến 0x77
- Điện áp giao tiếp: 1.8, 2.5, 3.3, 5VDC
- Tần số hoạt động: 0 – 400kHz

4.1.8 PCA9685



Hình 4.8: Module PCA9685

Thông số kỹ thuật:

- Mạch điều khiển 16 Chanel PWM PCA9685
- IC chính: PCA9685
- Điện áp sử dụng: 2.3 ~ 5.5VDC.
- Số kênh PWM: 16 kênh, tần số: 40 ~ 1000Hz

- Độ phân giải PWM: 12bit.
- Giao tiếp: I2C (chấp nhận mức Logic TTL $3 \sim 5VDC$)
- Kích thước: $62.5mm \times 25.4mm \times 3mm$

4.1.9 LED công suất 3W



Hình 4.9: LED công suất 3W

Thông số kỹ thuật:

- Công suất tiêu thụ 3W
- Điện áp hoạt động : 3.5-4V
- Dòng tiêu thụ: 600-750mA
- Độ sáng : 180LM-200LM
- Nhiệt độ hoạt động: $-20^{\circ}C \sim 60^{\circ}C$

4.1.10 Cảm biến ánh sáng

Cảm biến cường độ ánh sáng GY-30 BH1750FVI là một cảm biến ánh sáng kỹ thuật số. Gồm một linh kiện điện tử IC cảm biến ánh sáng cho giao tiếp I2C. IC này là thích hợp nhất để nhận diện các dữ liệu ánh sáng xung quanh cho việc điều chỉnh màn hình LCD và bàn phím đèn nền sức mạnh của điện thoại di động. Nó có thể phát hiện nhiều ở độ phân giải cao (1-65535 lx).

Thông số kỹ thuật:

- Module cảm biến cường độ sáng sử dụng chíp BH1750FVI.
- Điện áp cung cấp: $3V - 5V$.



Hình 4.10: Cảm biến ánh sáng GY-30 BH1750

- Phạm vi phát hiện sáng: 0 – 65535 lux.
- Cảm biến sử dụng bộ chuyển đổi ADC 16 bit. Tín hiệu đầu ra là tín hiệu số.
- Giao tiếp với MCU theo chuẩn I2C.

4.2 Sơ đồ kết nối

4.2.1 Sơ đồ kết nối tổng quát

Sơ đồ nguyên lý: Hình 4.11.

Sơ đồ thực tế: Hình 4.12.

4.2.2 Sơ đồ kết nối đèn và module L298N

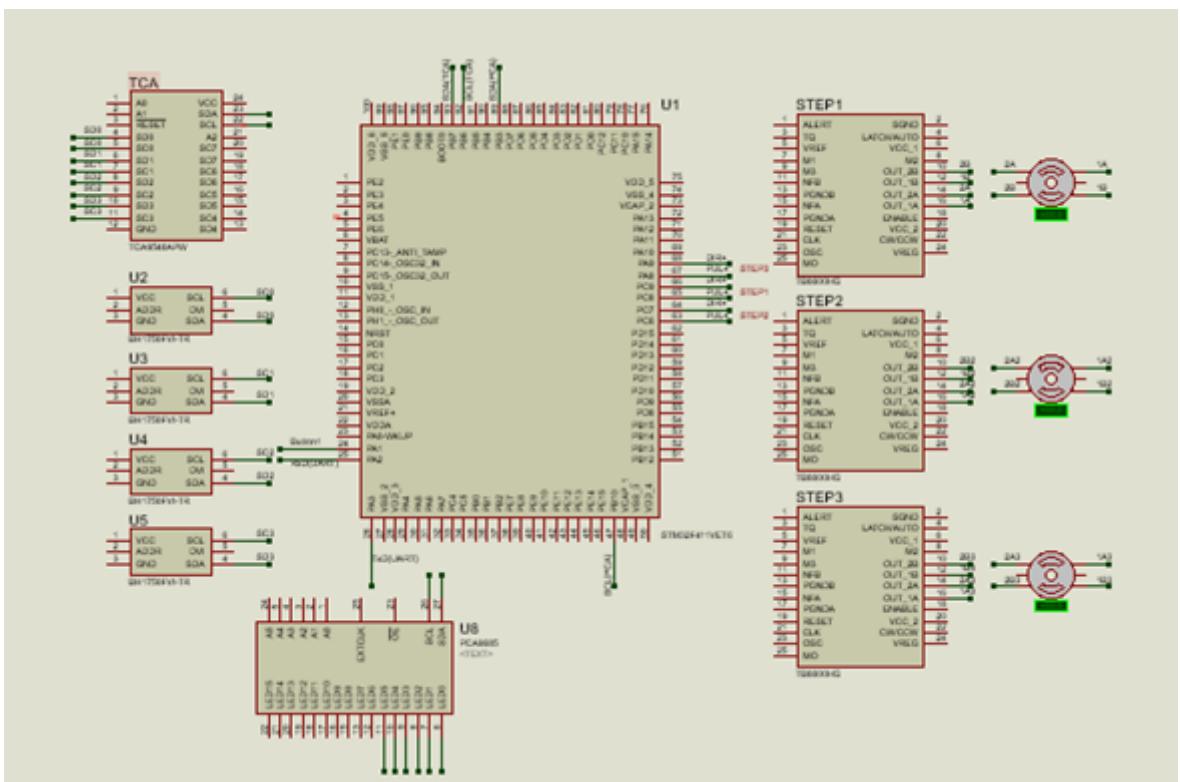
Sơ đồ nguyên lý: Hình 4.13.

Sơ đồ thực tế: Hình 4.14.

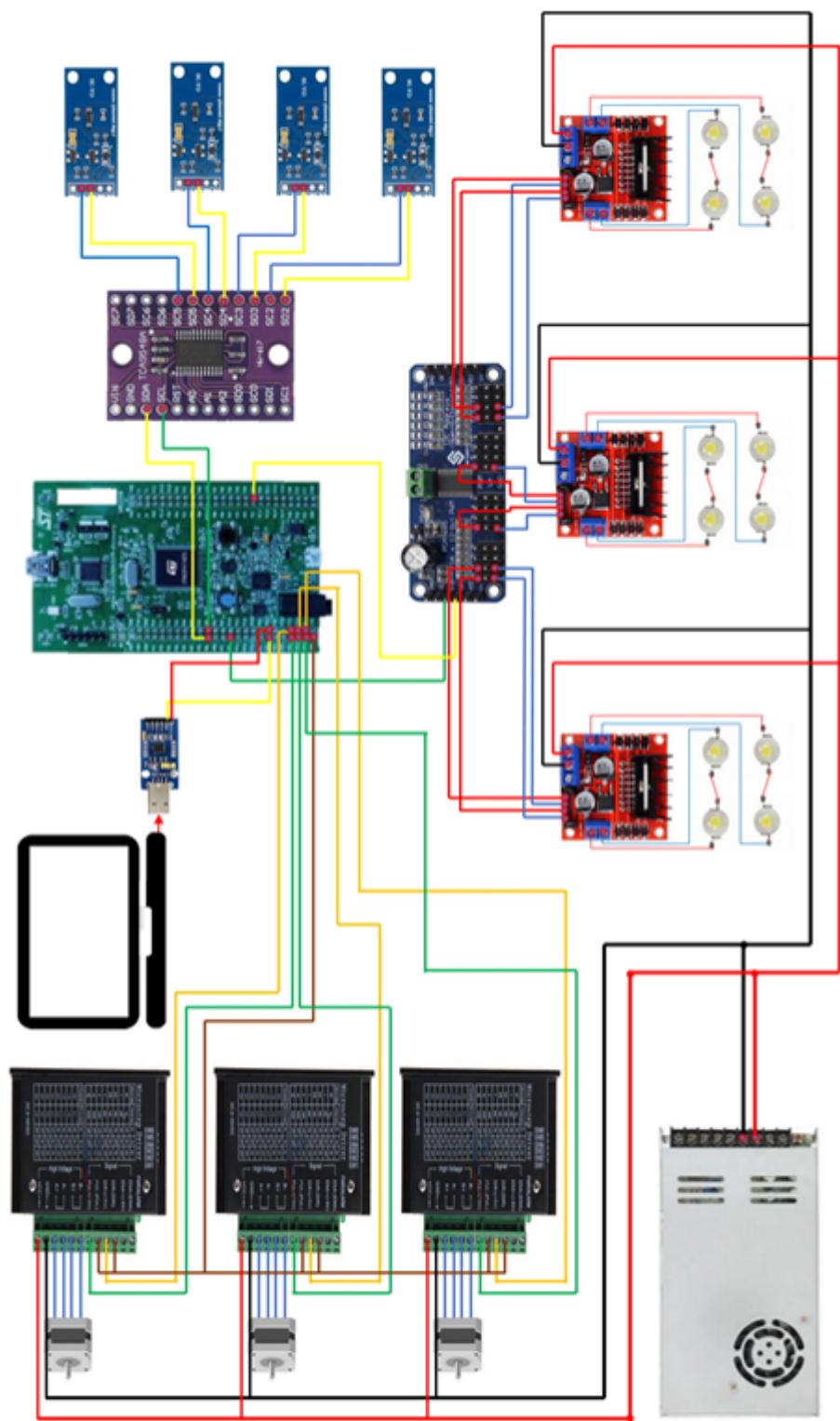
4.2.3 Sơ đồ kết nối Driver TB6600 với động cơ bước

Sơ đồ nguyên lý: Hình 4.15.

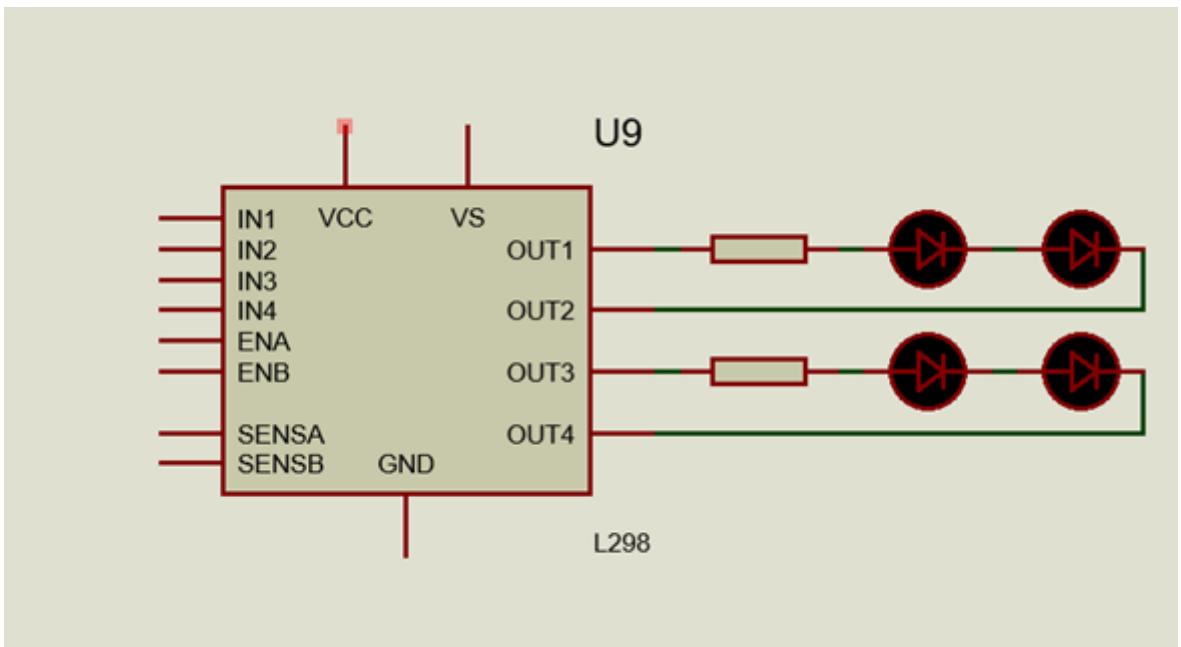
Sơ đồ thực tế: Hình 4.16.



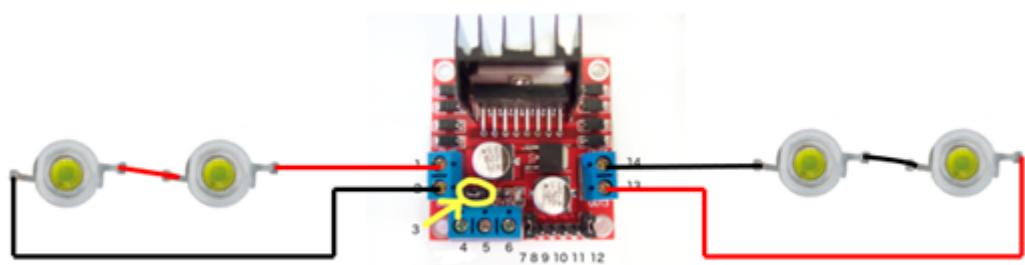
Hình 4.11: Sơ đồ nguyên lý tổng quát



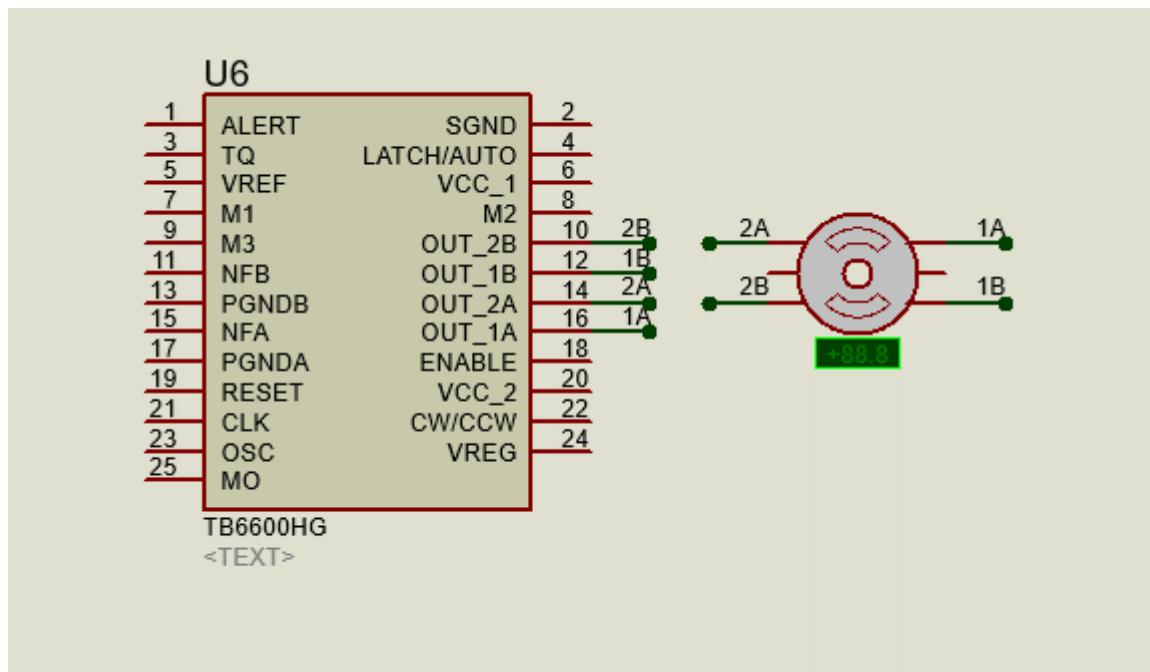
Hình 4.12: Sơ đồ kết nối điện thực tế



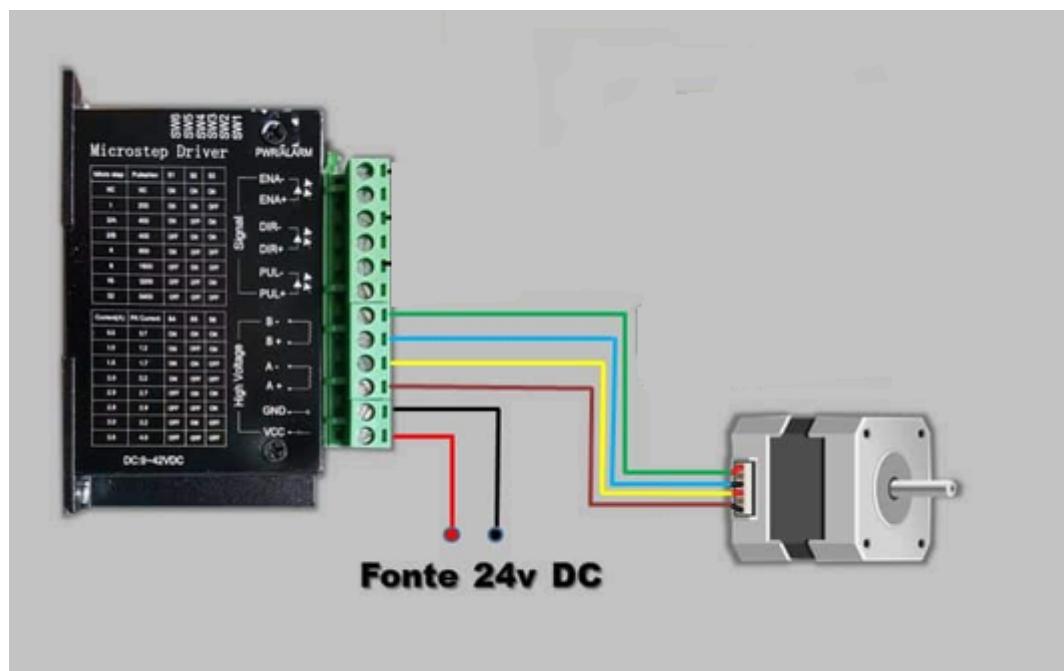
Hình 4.13: Sơ đồ nguyên lý kết nối đèn với module L298N



Hình 4.14: Sơ đồ kết nối thực tế đèn với module L298N



Hình 4.15: Sơ đồ nguyên lý kết nối Driver và động cơ



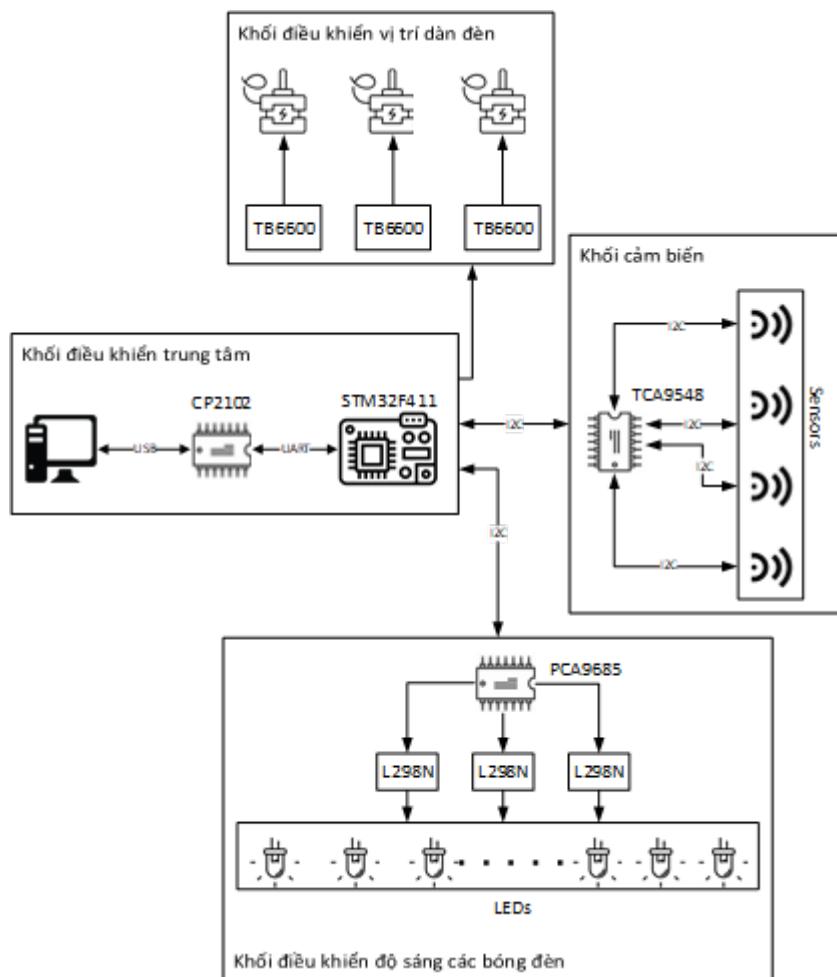
Hình 4.16: Sơ đồ kết nối thực tế Driver và động cơ

Chương 5

ĐIỀU KHIỂN VÀ LẬP TRÌNH

5.1 Tổng quan về phần điều khiển

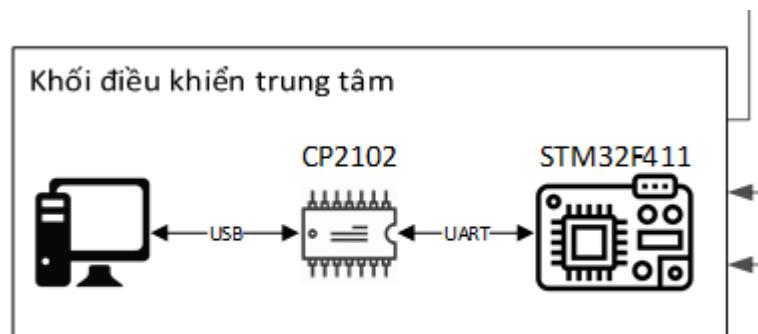
5.1.1 Sơ đồ khái niệm hệ thống điều khiển



Hình 5.1: Sơ đồ khái niệm hệ thống điều khiển

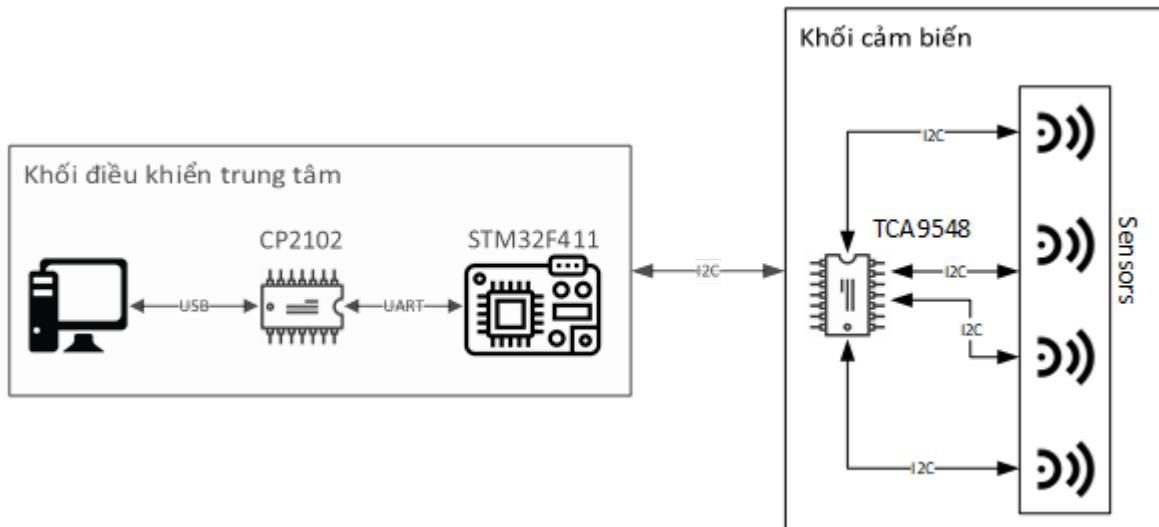
5.1.2 Khối điều khiển trung tâm

- PC: dùng 1 phần mềm giao diện để gửi tín hiệu điều khiển xuống VĐK và hiển thị dữ liệu do VĐK trả về.
- VĐK STM32F4: thực hiện các tác vụ chính của hệ thống: truyền nhận tín hiệu I2C với các module, xuất tín hiệu logic tới driver động cơ bước.
- Module USB to UART CP2102: chuyển đổi giao tiếp từ USB để PC và VĐK có thể giao tiếp với nhau qua chuẩn giao tiếp UART.



Hình 5.2: Khối điều khiển trung tâm

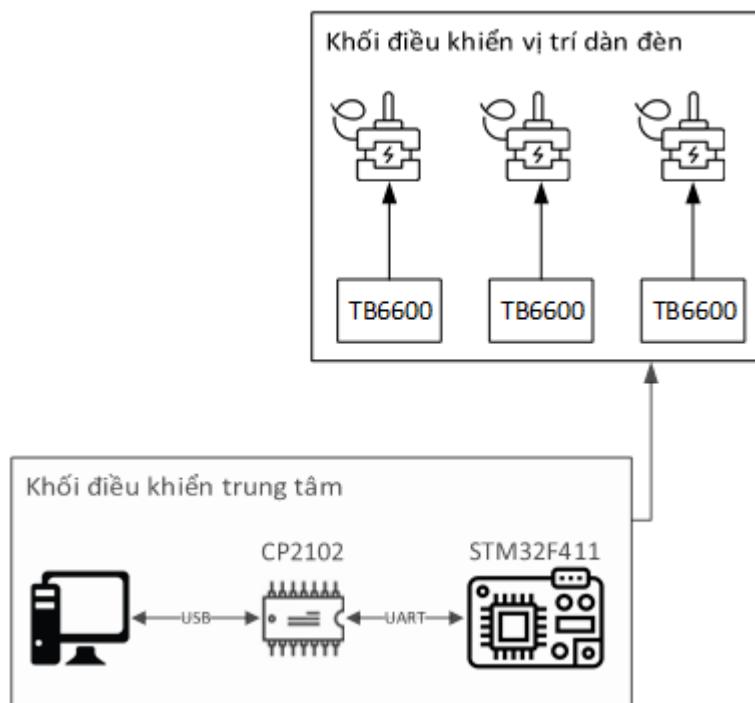
5.1.3 Khối cảm biến



Hình 5.3: Khối cảm biến

- Module TCA9548: mở rộng kênh I2C, khuếch đại tín hiệu trên 2 đường dẫn SCL và SDA.
- Sensors: các module cảm biến GY-30 BH1750. Đo lường cường độ ánh sáng và gửi về VĐK bằng chuẩn giao tiếp I2C.

5.1.4 Khối điều khiển vị trí các dàn đèn

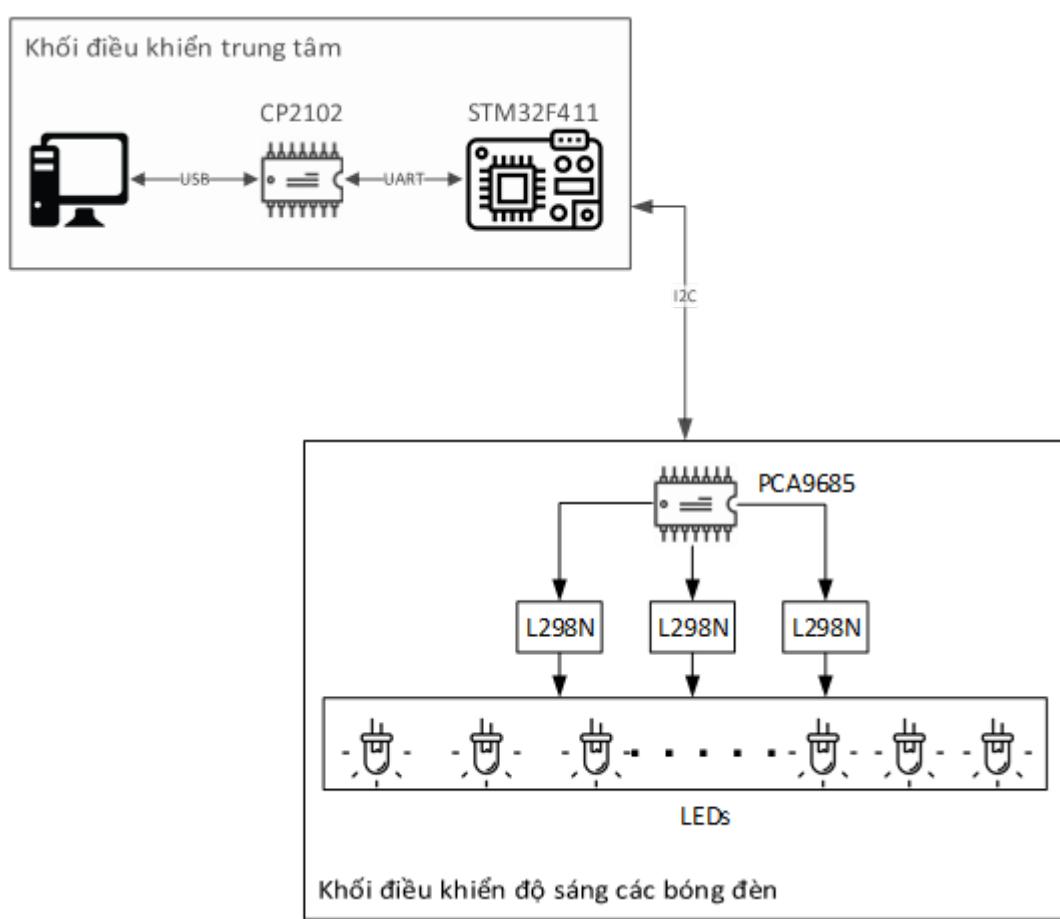


Hình 5.4: Khối điều khiển vị trí các dàn đèn

- Các module TB6600: Nhận xung điều khiển từ VĐK và điều khiển các động cơ quay theo xung điều khiển.
- Động cơ bước: thông qua bộ truyền đai kéo các dàn đèn để điều khiển dàn đèn tới vị trí mong muốn.

5.1.5 Khối điều khiển độ sáng các bóng đèn

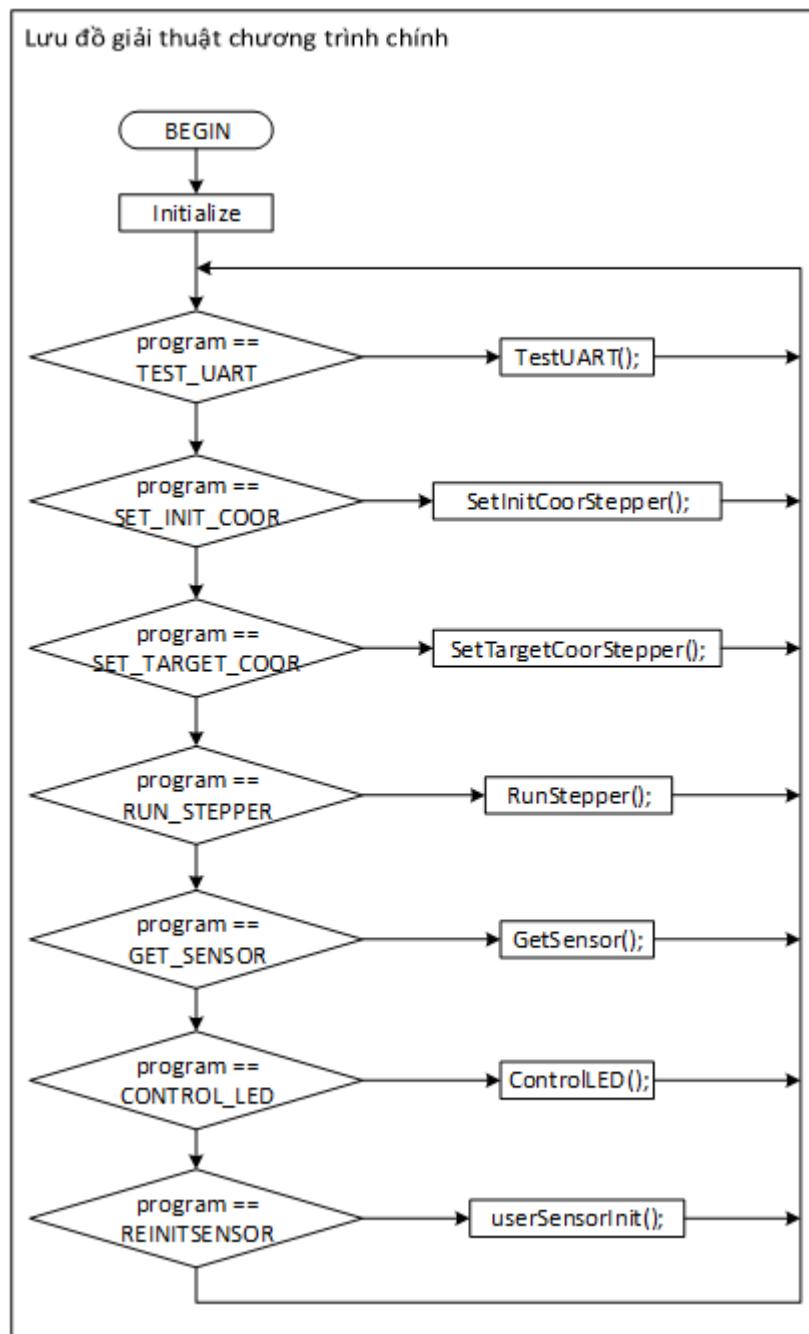
- PCA9685: Xuất xung PWM để điều khiển độ sáng các bóng đèn.
- L298N: Nhận xung PWM từ PCA9685 để điều khiển độ sáng bóng đèn lấy nguồn từ bộ nguồn 12V.
- Bóng đèn LED: Chiếu sáng cho mô hình.



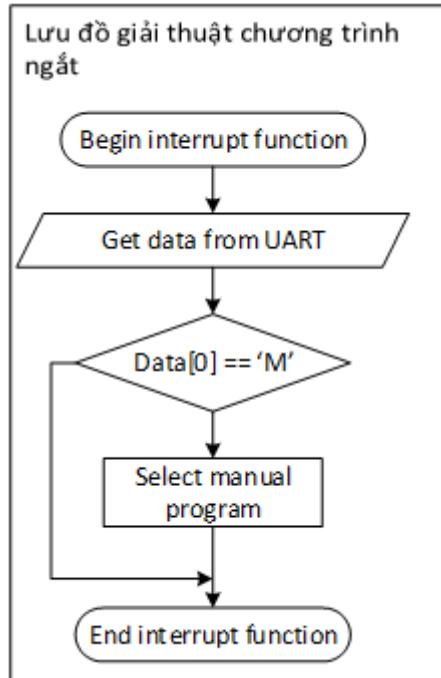
Hình 5.5: Khối điều khiển độ sáng các bóng đèn

5.2 Lập trình các chức năng Manual

5.2.1 Lưu đồ giải thuật chương trình điều khiển



Hình 5.6: Lưu đồ giải thuật chương trình chính



Hình 5.7: Lưu đồ giải thuật chương trình ngắn

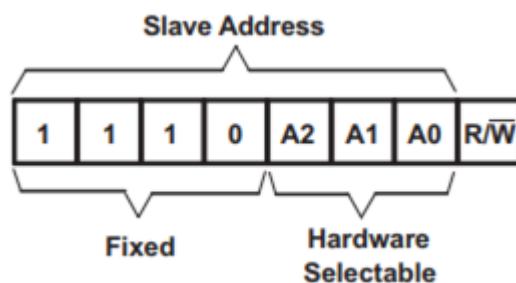
5.2.2 Đọc dữ liệu từ cảm biến

Một cảm biến ánh sáng GY-30 BH1750 chỉ có thể cài đặt được 2 địa chỉ I2C là “1011100” (0x5C) và “0100011” (0x23). Như vậy 1 kênh I2C chỉ có thể kết nối được tối đa 2 cảm biến. Vì điều khiển STM32F411VETx chỉ có 3 kênh I2C trong khi số cảm biến cần kết nối rất nhiều (16 cảm biến), nên nhóm dùng thêm module tăng số kênh I2C TCA9548A, mở rộng từ 1 kênh I2C lên 8 kênh I2C.

Nhà sản xuất chỉ có sẵn thư viện cho BH1750 và TCA9548A cho dòng VDK Atmega328P. Vì thế, nhóm phải tự viết lại thư viện riêng dùng bộ thư viện HAL cho dòng VDK STM32F4.

Module TCA9548A:

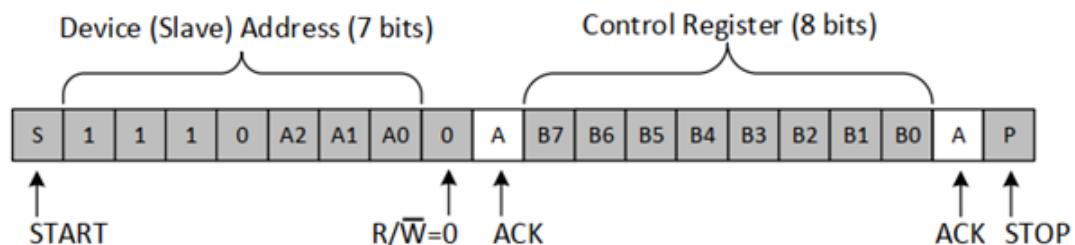
Địa chỉ I2C:



Hình 5.8: Địa chỉ I2C của module TCA9548A

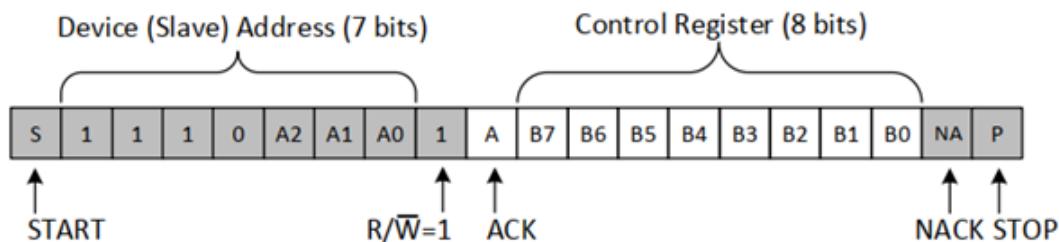
Khi 3 chân A0, A1, A2 trên module được thả nổi thì module sẽ có địa chỉ I2C là 0b1110000 hay 0x70.

Gửi data cho module:



Hình 5.9: Quy cách gửi data cho module TCA9548A

Đọc dữ liệu từ module:



Hình 5.10: Quy cách đọc data từ module TCA9548A

Bảng định nghĩa hàm dựa trên control register:

CONTROL REGISTER BITS								COMMAND
B7	B6	B5	B4	B3	B2	B1	B0	
X	X	X	X	X	X	X	0	Channel 0 disabled
							1	Channel 0 enabled
X	X	X	X	X	X	0	X	Channel 1 disabled
						1		Channel 1 enabled
X	X	X	X	X	0	X	X	Channel 2 disabled
					1			Channel 2 enabled
X	X	X	X	0	X	X	X	Channel 3 disabled
				1				Channel 3 enabled
X	X	X	0	X	X	X	X	Channel 4 disabled
			1					Channel 4 enabled
X	X	0	X	X	X	X	X	Channel 5 disabled
		1						Channel 5 enabled
X	0	X	X	X	X	X	X	Channel 6 disabled
	1							Channel 6 enabled
0	X	X	X	X	X	X	X	Channel 7 disabled
1								Channel 7 enabled
0	0	0	0	0	0	0	0	No channel selected, power-up/reset default state

Hình 5.11: Bảng định nghĩa hàm dựa trên Control Register của TCA9548A

Thư viện cho module TCA9548A

```
\\" TCA9548A_Driver.h
#ifndef TCA9548_H_
#define TCA9548_H_

#include "main.h"
#include "stm32f4xx_hal_i2c.h"

#define TCA_ADDRESS (0x70 << 1)

typedef enum {
    TCA9548A_OK = 0,
    TCA9548A_ERROR = 1
} TCA9548A_STATUS;

typedef struct TCA9548_Driver {
    I2C_HandleTypeDef *I2C_channel;
    uint8_t Address;
} TCA9548A_HandleTypeDef;

TCA9548A_Status TCA9548A_Init(TCA9548A_HandleTypeDef *htca9548a,
    I2C_HandleTypeDef *hi2c, uint16_t address);
TCA9548A_Status TCA9548A_SelectSingleChannel(TCA9548A_HandleTypeDef
    *htca9548a, uint8_t channel);
TCA9548A_Status TCA9548A_SelectMultiChannel(TCA9548A_HandleTypeDef
    *htca9548a, uint8_t channel);
uint8_t TCA9548A_ReadStatus(TCA9548A_HandleTypeDef *htca9548a);
TCA9548A_Status TCA9548A_DisableAllChannel(TCA9548A_HandleTypeDef
    *htca9548a);
```

```
#endif
```

```
\\" TCA9548A_Driver.c
#include "TCA9548A_Driver.h"

/*
 * @brief Initialize
 * @param htca9548a Pointer to a TCA9548A_HandleTypeDef
 * @param hi2c Pointer to a I2C_HandleTypeDef structure that contains
 *             the configuration information for the specified I2C.
 * @param address Target BH1750 device address. The device 7 bits
 *                address value in datasheet must
 *                be shifted to the left before calling by I2C function.
 * @retval TCA9548A Status
 */
```

```

TCA9548A_STATUS TCA9548A_Init(TCA9548A_HandleTypeDef *htca9548a,
    I2C_HandleTypeDef *hi2c, uint16_t address) {
    htca9548a->I2C_channel = hi2c;
    htca9548a->Address = address;

    if (TCA9548A_OK == TCA9548A_DisableAllChannel(htca9548a)) {
        return TCA9548A_OK;
    }
    return TCA9548A_ERROR;
}

/*
 * @brief Initialize
 * @param htca9548a Pointer to a TCA9548A_HandleTypeDef
 * @param channel From 0 to 7, is channel 0 to channel 7 on TCA9548A
 * @retval TCA9548A Status
 */
TCA9548A_STATUS TCA9548A_SelectSingleChannel(TCA9548A_HandleTypeDef
    *htca9548a, uint8_t channel) {
    if (channel > 7) {
        return TCA9548A_ERROR;
    }
    uint8_t tmp = (1 << channel);
    if (HAL_OK == HAL_I2C_Master_Transmit(htca9548a->I2C_channel,
        htca9548a->Address, &tmp, 1, 10)) {
        return TCA9548A_OK;
    }
    return TCA9548A_ERROR;
}

/*
 * @brief Initialize
 * @param htca9548a Pointer to a TCA9548A_HandleTypeDef
 * @param channel This have 8 bit to choose which channel is selected.
 *                 Ex: channel 0, 3, 5 is selected, channel = 0b00101001 =
 *                     0x29
 * @retval TCA9548A Status
 */
TCA9548A_STATUS TCA9548A_SelectMultiChannel(TCA9548A_HandleTypeDef
    *htca9548a, uint8_t channel) {
    if (HAL_OK == HAL_I2C_Master_Transmit(htca9548a->I2C_channel,
        htca9548a->Address, &channel, 1, 10)) {
        return TCA9548A_OK;
    }
    return TCA9548A_ERROR;
}

```

```
uint8_t TCA9548A_ReadStatus(TCA9548A_HandleTypeDef *htca9548a) {
    uint8_t tmp;
    if (HAL_OK == HAL_I2C_Master_Receive(htca9548a->I2C_channel,
        htca9548a->Address, &tmp, 1, 10)) {
        return tmp;
    }
    return 0;
}

TCA9548A_STATUS TCA9548A_DisableAllChannel(TCA9548A_HandleTypeDef
*htca9548a) {
    uint8_t tmp = 0x00;
    if (HAL_OK == HAL_I2C_Master_Transmit(htca9548a->I2C_channel,
        htca9548a->Address, &tmp, 1, 10)) {
        return TCA9548A_OK;
    }
    return TCA9548A_ERROR;
}
```

Module GY-30 BH1750

Địa chỉ I2C

Khi chân ADDR của module có tín hiệu mức cao ($\geq 0.7VCC$) thì module có địa chỉ là: 0b1011100 hay 0x5C, ngược lại module sẽ có địa chỉ là 0b0100011 hay 0x23.

Gửi data cho module:

BH1750FVI is not able to accept plural command without stop condition. Please insert SP every 1 Opcode.

ST	Slave Address	R/W 0	Ack	Opcode	Ack	SP
----	---------------	----------	-----	--------	-----	----

Hình 5.12: Gửi data cho module BH1750

Đọc data từ module:

ST	Slave Address	R/W 1	Ack	$2^{15} \ 2^{14} \ 2^{13} \ 2^{12} \ 2^{11} \ 2^{10} \ 2^9 \ 2^8$ High Byte [15:8]	Ack	
				$2^7 \ 2^6 \ 2^5 \ 2^4 \ 2^3 \ 2^2 \ 2^1 \ 2^0$ Low Byte [7:0]	Ack	SP

 from Master to Slave  from Slave to Master

Hình 5.13: Đọc data từ module BH1750

Bảng mã điều khiển:

Instruction	Opcode	Comments
Power Down	0000_0000	No active state.
Power On	0000_0001	Waiting for measurement command.
Reset	0000_0111	Reset Data register value. Reset command is not acceptable in Power Down mode.
Continuously H-Resolution Mode	0001_0000	Start measurement at 1lx resolution. Measurement Time is typically 120ms.
Continuously H-Resolution Mode2	0001_0001	Start measurement at 0.5lx resolution. Measurement Time is typically 120ms.
Continuously L-Resolution Mode	0001_0011	Start measurement at 4lx resolution. Measurement Time is typically 16ms.
One Time H-Resolution Mode	0010_0000	Start measurement at 1lx resolution. Measurement Time is typically 120ms. It is automatically set to Power Down mode after measurement.
One Time H-Resolution Mode2	0010_0001	Start measurement at 0.5lx resolution. Measurement Time is typically 120ms. It is automatically set to Power Down mode after measurement.
One Time L-Resolution Mode	0010_0011	Start measurement at 4lx resolution. Measurement Time is typically 16ms. It is automatically set to Power Down mode after measurement.
Change Measurement time (High bit)	01000_MT[7,6,5]	Change measurement time. ※ Please refer "adjust measurement result for influence of optical window."
Change Measurement time (Low bit)	011_MT[4,3,2,1,0]	Change measurement time. ※ Please refer "adjust measurement result for influence of optical window."

※ Don't input the other opcode.

Hình 5.14: Bảng mã điều khiển của module BH1750

Thư viện cho module GY-30 BH1750:

```
\\" BH1750_Driver.h
#ifndef BH1750_H_
#define BH1750_H_

/* Include ----- */
#include "stm32f4xx_hal.h"

#define BH1750_ADDRESS_HIGH      (0x5C<<1)
#define BH1750_ADDRESS_LOW       (0x23<<1)

#define BH1750_POWER_DOWN        0x00
#define BH1750_POWER_ON          0x01
#define BH1750_RESET              0x07

#define BH1750_DEFAULT_MTREG     69
#define BH1750_CONVERSION_FACTOR 1.2

typedef enum {
    BH1750_OK    = 0,
    BH1750_ERROR = 1
} BH1750_STATUS;

typedef enum {
    CONTINUOUS_H_RES_MODE = 0x10,
```

```

    CONTINUOUS_H_RES_MODE2 = 0x11,
    CONTINUOUS_L_RES_MODE = 0x13,
    ONETIME_H_RES_MODE   = 0x20,
    ONETIME_H_RES_MODE2  = 0x21,
    ONETIME_L_RES_MODE   = 0x23
} bh1750_mode;

typedef struct BH1750_Driver
{
    I2C_HandleTypeDef *I2C_channel;
    uint16_t Address;

} BH1750_HandleTypeDef;

BH1750_Status BH1750_Init(BH1750_HandleTypeDef *hbh1750,
                           I2C_HandleTypeDef *hi2c, uint16_t Address);
BH1750_Status BH1750_Reset(BH1750_HandleTypeDef *hbh1750);
BH1750_Status BH1750_PowerState(BH1750_HandleTypeDef *hbh1750, uint8_t
                                 PowerOn);
BH1750_Status BH1750_SetMtreg(BH1750_HandleTypeDef *hbh1750, uint8_t
                               Mtreg);
BH1750_Status BH1750_SetMode(BH1750_HandleTypeDef *hbh1750, bh1750_mode
                             Mode);
BH1750_Status BH1750_TriggerManualConversion(BH1750_HandleTypeDef
                                             *hbh1750);
BH1750_Status BH1750_ReadLight(BH1750_HandleTypeDef *hbh1750, uint16_t
                                *Result);

#endif /* BH1750_H_ */

```

```

\\ BH1750_Driver.c
/*
=====
##### How to use this driver #####
=====

Set up:
#1 @ref BH1750_Init()
    Ex: BH1750_Init(&sensor[i], &hi2c1, BH1750_ADDRESS_LOW);
#2 @ref BH1750_PowerState()
    Ex: BH1750_PowerState(&sensor[i], 1);
#3 @ref BH1750_SetMode()
    Ex: BH1750_SetMode(&sensor[i], CONTINUOUS_H_RES_MODE);

Let play: Read light @ref BH1750_ReadLight()
    BH1750_ReadLight(&sensor[i], &result[i]);
*/

```

```

#include "BH1750_Driver.h"

bh1750_mode    BH1750_Mode;
uint8_t         BH1750_Mtreg;

/*
 * @brief Initialize
 * @param hbh1750 Pointer to a BH1750_HandleTypeDef
 * @param hi2c Pointer to a I2C_HandleTypeDef structure that contains
 *               the configuration information for the specified I2C.
 * @param Address Target BH1750 device address. The device 7 bits
 *                 address value in datasheet must
 *                 be shifted to the left before calling by I2C function.
 * @retval BH1750 Status
 */
BH1750_STATUS BH1750_Init(BH1750_HandleTypeDef *hbh1750,
                           I2C_HandleTypeDef *hi2c, uint16_t Address) {
    hbh1750->I2C_channel = hi2c;
    hbh1750->Address = Address;
    if(BH1750_OK == BH1750_Reset(hbh1750))
    {
        if(BH1750_OK == BH1750_SetMtreg(hbh1750, BH1750_DEFAULT_MTREG))
            // Set default value;
            return BH1750_OK;
    }
    return BH1750_ERROR;
}

/*
 * @brief Reset all registers to default value.
 * @param hbh1750 Pointer to a BH1750_HandleTypeDef
 * @retval BH1750 status
 */
BH1750_STATUS BH1750_Reset(BH1750_HandleTypeDef *hbh1750) {
    uint8_t tmp = BH1750_RESET;
    if (HAL_OK == HAL_I2C_Master_Transmit(hbh1750->I2C_channel,
                                           hbh1750->Address, &tmp, 1, 10))
        return BH1750_OK;
    return BH1750_ERROR;
}

/*
 * @brief Set the power state.
 * @param hbh1750 Pointer to a BH1750_HandleTypeDef
 * @param PowerOn

```

```

*      @arg 0: Power down, low current, no active state.
*      @arg 1: Ready for measurement command.
* @retval BH1750 status
*/
BH1750_STATUS BH1750_PowerState(BH1750_HandleTypeDefDef *hbh1750, uint8_t
PowerOn) {
PowerOn = (PowerOn ? 1 : 0);
if (HAL_OK == HAL_I2C_Master_Transmit(hbh1750->I2C_channel,
hbh1750->Address, &PowerOn, 1, 10))
    return BH1750_OK;
return BH1750_ERROR;
}

/*
* @brief Adjust measurement result for influence of optical window.
(sensor sensitivity adjusting)
* @param hbh1750 Pointer to a BH1750_HandleTypeDefDef
* @param Mtreg The modified value of measurement time register. (31 <=
Mtreg <=254) (milliseconds)
* @retval BH1750 status
*/
BH1750_STATUS BH1750_SetMtreg(BH1750_HandleTypeDefDef *hbh1750, uint8_t
Mtreg) {
HAL_StatusTypeDef retCode;
if (Mtreg < 31 || Mtreg > 254) {
    return BH1750_ERROR;
}

BH1750_Mtreg = Mtreg;

uint8_t tmp[2];

tmp[0] = (0x40 | (Mtreg >> 5)); // High bit 01000_MT[7,6,5]
tmp[1] = (0x60 | (Mtreg & 0x1F)); // Low bit 011_MT[4,3,2,1,0]

retCode = HAL_I2C_Master_Transmit(hbh1750->I2C_channel,
hbh1750->Address, &tmp[0], 1, 10);
if (HAL_OK != retCode) {
    return BH1750_ERROR;
}

retCode = HAL_I2C_Master_Transmit(hbh1750->I2C_channel,
hbh1750->Address, &tmp[1], 1, 10);
if (HAL_OK == retCode) {
    return BH1750_OK;
}

```

```

        return BH1750_ERROR;
    }

/*
 * @brief Set the mode of converting. Look into bh1750_mode enum.
 * @param hbh1750 Pointer to a BH1750_HandleTypeDef
 * @param Mode The mode of converting. This parameter can be one of the
 * bh1750_mode enum values:
 *          @arg CONTINUOUS_H_RES_MODE: Start measurement at 1lx
 * resolution.
 *          Measurement Time is typically
 * 120ms.
 *          @arg CONTINUOUS_H_RES_MODE2: Start measurement at 0.5lx
 * resolution.
 *          Measurement Time is typically
 * 120ms.
 *          @arg CONTINUOUS_L_RES_MODE: Start measurement at 4lx
 * resolution.
 *          Measurement Time is typically
 * 16ms.
 *          @arg ONETIME_H_RES_MODE: Start measurement at 1lx
 * resolution.
 *          Measurement Time is typically
 * 120ms.
 *          It is automatically set to
 * Power Down mode after measurement.
 *          @arg ONETIME_H_RES_MODE2: Start measurement at 0.5lx
 * resolution.
 *          Measurement Time is typically
 * 120ms.
 *          It is automatically set to
 * Power Down mode after measurement.
 *          @arg ONETIME_L_RES_MODE: Start measurement at 4lx
 * resolution.
 *          Measurement Time is typically
 * 16ms.
 *          It is automatically set to
 * Power Down mode after measurement.
 * @retval BH1750 status
 */
BH1750_STATUS BH1750_SetMode(BH1750_HandleTypeDef *hbh1750, bh1750_mode
Mode) {
    if(!((Mode >> 4) || (Mode >> 5))) return BH1750_ERROR;
    if((Mode & 0x0F) > 3) return BH1750_ERROR;
}

```

```

BH1750_Mode = Mode;
if(HAL_OK == HAL_I2C_Master_Transmit(hbh1750->I2C_channel,
    hbh1750->Address, &Mode, 1, 10))
    return BH1750_OK;

return BH1750_ERROR;
}

// BH1750_STATUS BH1750_TriggerManualConversion(uint16_t Address);

/*
 * @brief Read the converted value and calculate the result.
 * @param hbh1750 Pointer to a BH1750_HandleTypeDef
 * @param Result Pointer to your variable for getting result.
 * @retval BH1750 Status
 */
BH1750_STATUS BH1750_ReadLight(BH1750_HandleTypeDef *hbh1750, uint16_t
    *Result) {
    uint16_t result;
    uint8_t tmp[2];

    if(HAL_OK == HAL_I2C_Master_Receive(hbh1750->I2C_channel,
        hbh1750->Address, tmp, 2, 10))
    {
        result = (tmp[0] << 8) | (tmp[1]);

        if(BH1750_Mtreg != BH1750_DEFAULT_MTREG)
        {
            result *= ((float)(uint8_t)(BH1750_DEFAULT_MTREG) /
                (float)BH1750_Mtreg);
        }

        if(BH1750_Mode == ONETIME_H_RES_MODE2 || BH1750_Mode ==
            CONTINUOUS_H_RES_MODE2)
        {
            result /= 2.0;
        }

        *Result = result / (float)BH1750_CONVERSION_FACTOR;
        return BH1750_OK;
    }
    return BH1750_ERROR;
}

```

Phần chương trình đọc dữ liệu cảm biến trong file main.c:

```
// Variables for Sensor and TCA
```

```

TCA9548A_HandleTypeDef i2cHub;
BH1750_HandleTypeDef sensor[4];
char dataSensorMessage[24] = {};
uint16_t dataSensor[4];
uint8_t ret;
void userSensorInit() {
    TCA9548A_Init(&i2cHub, &hi2c1, TCA_ADDRESS);
    for (uint8_t i = 0; i < 4; i++) {
        TCA9548A_SelectSingleChannel(&i2cHub, i);
        BH1750_Init(&sensor[i], &hi2c1, BH1750_ADDRESS_LOW);
        BH1750_PowerState(&sensor[i], 1);
        BH1750_SetMode(&sensor[i], CONTINUOUS_H_RES_MODE);
    }
}
void GetSensor() {
    for (uint8_t i = 0; i < 4; i++) {
        TCA9548A_SelectSingleChannel(&i2cHub, i);
        BH1750_ReadLight(&sensor[i], &dataSensor[i]);

        dataSensorMessage[i*6 + 5] = (i < 3) ? ' ' : '\n';
        dataSensorMessage[i*6 + 4] = dataSensor[i] % 10 + '0';
        dataSensorMessage[i*6 + 3] = dataSensor[i] / 10 % 10 + '0';
        dataSensorMessage[i*6 + 2] = dataSensor[i] / 100 % 10 + '0';
        dataSensorMessage[i*6 + 1] = dataSensor[i] / 1000 % 10 + '0';
        dataSensorMessage[i*6 + 0] = dataSensor[i] / 10000 % 10 + '0';
    }
    HAL_UART_Transmit(&huart2, (uint8_t *)dataSensorMessage, 24, 24);
    program = IDLE;
}

```

5.2.3 Điều khiển tọa độ các dàn đèn

Tọa độ của các dàn đèn được điều khiển bởi các động cơ bước sử dụng TB6600 làm driver.

Thư viện cho Driver TB6600:

```

\\ Stepper_Driver.h
#ifndef STEPPERDRIVER_H_
#define STEPPERDRIVER_H_

#include "stm32f4xx_hal.h"

// Time delay write HIGH and LOW to PIN PULSE in microsecond
#define TdelayON 50
#define TdelayOFF 300

```

```

// diameter of pulley (millimeters)
#define DIAMETER 13.2
#define PI      3.14159

// PulTranfer = 200 * stepper->Microstep / Diameter / PI = 160
#define FACTOR 160

typedef enum {
    STEPPER_OK    = 0,
    STEPPER_ERROR = 1
} STEPPER_STATUS;

typedef struct StepperDriver {
    GPIO_TypeDef* Port;
    uint16_t GPIO_Pin_Dir;
    uint16_t GPIO_Pin_Pulse;
    // uint16_t GPIO_Pin_Enable;
    uint8_t Microstep;
    uint32_t CurrentPulse;
    uint32_t TargetPulse;
} Stepper_HandleTypeDef;

STEPPER_STATUS StepperInit(Stepper_HandleTypeDef *stepper,
                           GPIO_TypeDef* port,
                           uint16_t pin_Dir,
                           uint16_t pin_Pulse,
                           uint8_t microstep,
                           uint16_t currentPos);
STEPPER_STATUS setCurrentPos(Stepper_HandleTypeDef *stepper, uint16_t
                             current);
STEPPER_STATUS setTargetPos(Stepper_HandleTypeDef *stepper, uint16_t
                            target);
STEPPER_STATUS setDirCCW(Stepper_HandleTypeDef *stepper);
STEPPER_STATUS setDirCW(Stepper_HandleTypeDef *stepper);
STEPPER_STATUS runToTarget(Stepper_HandleTypeDef *stepper);

#endif /* STEPPERDRIVER_H_ */

```

```

\\ Stepper_Driver.c
/*
 * StepperDriver.c
 *
 * Created on: Apr 25, 2021
 *      Author: Ice Cream
 */

```

```

=====
*      HOW TO USE THIS DRIVER
=====
* #1: Khoi tao dong co bang ham @ref StepperInit()
* #2: Dat vi tri ban dau bang ham @ref setCurrentPos()
* #3: Dat vi tri muon di toi bang ham @ref setTargetPos()
* #4: Them ham @ref runToTarget() vao vong lap de dong co se tu dong
      chay toi vi
      da chon.
*
*/

```

```

#include "StepperDriver.h"

/*
 * @brief Initialize
 * @param stepper Pointer to a Stepper Motor
 * @param Port is GPIOx, x is the port (A ... E) you connect the
 *          direction and pulse pins from the driver to stm32
 * @param microstep specifies the microstep value that is in use. (usually
 *          1, 2, 4, 8, 16, 32)
 * @param currentPos specifies the motor's current position
 * @retval Stepper Status
 */
STEPPER_STATUS StepperInit(Stepper_HandleTypeDef *stepper,
                           GPIO_TypeDef* port,
                           uint16_t pin_Dir,
                           uint16_t pin_Pulse,
                           uint8_t microstep,
                           uint16_t currentPos) {
    stepper->Port = port;
    stepper->GPIO_Pin_Dir = pin_Dir;
    stepper->GPIO_Pin_Pulse = pin_Pulse;
    stepper->Microstep = microstep;
    stepper->CurrentPulse = currentPos * FACTOR;
    stepper->TargetPulse = currentPos * FACTOR;

    return STEPPER_OK;
}

STEPPER_STATUS setCurrentPos(Stepper_HandleTypeDef *stepper, uint16_t
                             current) {
    stepper->CurrentPulse = current * FACTOR;
    stepper->TargetPulse = current * FACTOR;
    return STEPPER_OK;
}

```

```

STEPPER_STATUS setTargetPos(Stepper_HandleTypeDef *stepper, uint16_t
    target) {
    stepper->TargetPulse = target * FACTOR;
    return STEPPER_OK;
}

STEPPER_STATUS setDirCCW(Stepper_HandleTypeDef *stepper) {
    HAL_GPIO_WritePin(stepper->Port, stepper->GPIO_Pin_Dir,
        GPIO_PIN_SET);
    return STEPPER_OK;
}

STEPPER_STATUS setDirCW(Stepper_HandleTypeDef *stepper) {
    HAL_GPIO_WritePin(stepper->Port, stepper->GPIO_Pin_Dir,
        GPIO_PIN_RESET);
    return STEPPER_OK;
}

__weak void delay_us(uint16_t us)
{
    /* Prevent unused argument(s) compilation warning */
    UNUSED(us);
    /* TIM2 configure
     * Clock Source: Internal Clock
     * Prescaler: 50 - 1
     * Counter Period: 0xFFFF - 1
     * Clock Configure: 50MHz
     * don't forget to add this below USER CODE BEGIN 2
     * HAL_TIM_Base_Start(&htim2);
     * Code:
     * __HAL_TIM_SET_COUNTER(&htim2,0); // set the counter value a 0
     * while ((uint16_t)__HAL_TIM_GET_COUNTER(&htim2) < us);
     */
}
}

STEPPER_STATUS runToTarget(Stepper_HandleTypeDef *stepper) {

    if (stepper->TargetPulse != stepper->CurrentPulse) {
        // convert Position in millimeters to pulses base on microstep

        if (stepper->TargetPulse > stepper->CurrentPulse) {
            setDirCCW(stepper);
            HAL_GPIO_WritePin(stepper->Port, stepper->GPIO_Pin_Pulse,
                GPIO_PIN_SET);
            delay_us(TdelayON);
        }
    }
}

```

```

        HAL_GPIO_WritePin(stepper->Port, stepper->GPIO_Pin_Pulse,
                           GPIO_PIN_RESET);
        delay_us(TdelayOFF);
        stepper->CurrentPulse++;
    }
    else {
        setDirCW(stepper);
        HAL_GPIO_WritePin(stepper->Port, stepper->GPIO_Pin_Pulse,
                           GPIO_PIN_SET);
        delay_us(TdelayON);
        HAL_GPIO_WritePin(stepper->Port, stepper->GPIO_Pin_Pulse,
                           GPIO_PIN_RESET);
        delay_us(TdelayOFF);
        stepper->CurrentPulse--;
    }
    return STEPPER_ERROR;
}
return STEPPER_OK;
}

```

Phần điều khiển các động cơ trong file main.c:

```

void SetInitCoorStepper() {
    DecryptData();
    setCurrentPos(&Step0, info[0]);
    setCurrentPos(&Step1, info[1]);
    setCurrentPos(&Step2, info[2]);
    HAL_UART_Transmit(&huart2, InitStepperMessage,
                      sizeof(InitStepperMessage), 1000);
    program = IDLE;
}

void SetTargetCoorStepper() {
    DecryptData();
    setTargetPos(&Step0, info[0]);
    setTargetPos(&Step1, info[1]);
    setTargetPos(&Step2, info[2]);
    keepMotorSafe();
    uint32_t locationStep0 = Step0.TargetPulse / FACTOR;
    uint32_t locationStep1 = Step1.TargetPulse / FACTOR;
    uint32_t locationStep2 = Step2.TargetPulse / FACTOR;
    uint8_t locationMessage[20];
    locationMessage[0] = 'T';
    locationMessage[1] = 'a';
    locationMessage[2] = 'r';
    locationMessage[3] = 'g';
    locationMessage[4] = 'e';
}

```

```

locationMessage[5] = 't';
locationMessage[6] = ':';
locationMessage[7] = ' ';
// Step0
locationMessage[10] = locationStep0 % 10 + '0';
locationMessage[9] = locationStep0 / 10 % 10 + '0';
locationMessage[8] = locationStep0 / 100 % 10 + '0';
locationMessage[11] = ' ';
// Step1
locationMessage[14] = locationStep1 % 10 + '0';
locationMessage[13] = locationStep1 / 10 % 10 + '0';
locationMessage[12] = locationStep1 / 100 % 10 + '0';
locationMessage[15] = ' ';
// Step2
locationMessage[18] = locationStep2 % 10 + '0';
locationMessage[17] = locationStep2 / 10 % 10 + '0';
locationMessage[16] = locationStep2 / 100 % 10 + '0';
locationMessage[19] = '\n';

HAL_UART_Transmit(&huart2, locationMessage, sizeof(locationMessage),
1000);
program = RUN_STEPPER;
}

void RunStepper() {
    uint8_t flag = 0;
    if (runToTarget(&Step0) == STEPPER_ERROR) flag = 1;
    if (runToTarget(&Step1) == STEPPER_ERROR) flag = 1;
    if (runToTarget(&Step2) == STEPPER_ERROR) flag = 1;
    if (flag == 0) {
        HAL_UART_Transmit(&huart2, Done, sizeof(Done), 1000);
        program = IDLE;
    }
}

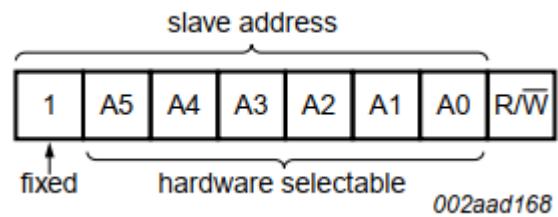
```

5.2.4 Điều khiển độ sáng của các bóng đèn

Độ sáng của các bóng đèn được điều khiển bằng module PCA9685. Tuy nhiên do module này chỉ có thể xuất xung PWM với 25mA, 5.5V trong khi bóng đèn sử dụng trong mô hình là đèn công suất với dòng cao. Vì thế nhóm sử dụng thêm module L298N để nhận xung PWM và cấp điện cho đèn.

Module PCA9685:

Địa chỉ I2C:



Hình 5.15: Địa chỉ của module PCA9685

Các chân A0 tới A5 đều đang thả nổi, địa chỉ của module hiện tại là: 0b1000000 hay 0x40.

Cài đặt xung PWM cho 1 channel:

Module tính toán thời điểm tắt và bật bằng counter với xung clock đầu vào là 27MHz. Các thanh ghi sử dụng (giả sử đối với channel 0):

LED0_ON_H, LED0_ON_L: 2 byte này lưu trữ thời điểm IC sẽ xuất tín hiệu ra chân PWM lên mức HIGH.

LED0_OFF_H, LED0_OFF_L: 2 byte này lưu trữ thời điểm IC sẽ xuất tín hiệu ra chân PWM xuống mức LOW.

Ví dụ cách tính giá trị cần ghi vào các thanh ghi trên:

$$Delaytime = 10\%$$

$$PWMDutyCycle = 20\% (LEDontime = 20\%, LEDofftime = 80\%)$$

Tính toán:

$$Delaytime = 4096 \times 10\% = 409.6 \approx 410 = 0x19A$$

Do counter đếm từ 0 tới 4096 nên Delaytime = 0x199.

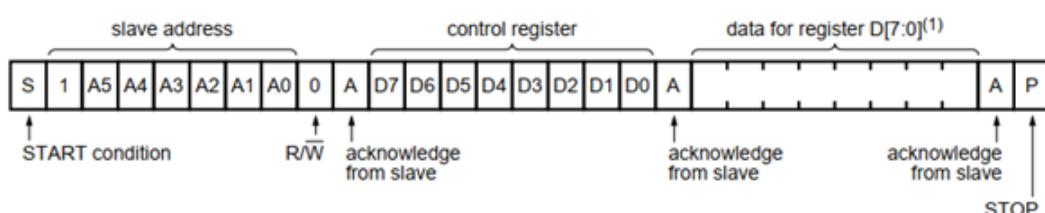
$$LED0_ON_H = 0x1, LED0_ON_L = 0x99$$

$$LED\ on\ time = 20\% = 819.2 \sim 819 counts$$

$$LED\ off\ time = 0x4CC$$

$$LED0_OFF_H = 0x4, LED0_OFF_L = 0xCC$$

Gửi data cho module:



Hình 5.16: Gửi data cho module PCA9685

```

\\ PCA9685_Driver.h
#ifndef PCA9685_H_
#define PCA9685_H_


// include
#include "stm32f4xx_hal.h"

// REGISTER ADDRESSES
#define PCA9685_MODE1      0x00    /**< Mode Register 1 */
#define PCA9685_MODE2      0x01    /**< Mode Register 2 */
#define PCA9685_SUBADR1    0x02    /**< I2C-bus subaddress 1 */
#define PCA9685_SUBADR2    0x03    /**< I2C-bus subaddress 2 */
#define PCA9685_SUBADR3    0x04    /**< I2C-bus subaddress 3 */
#define PCA9685_ALLCALLADR 0x05    /**< LED All Call I2C-bus address */
#define PCA9685_LED0_ON_L   0x06    /**< LED0 on tick, low byte*/
#define PCA9685_LED0_ON_H   0x07    /**< LED0 on tick, high byte*/
#define PCA9685_LED0_OFF_L  0x08    /**< LED0 off tick, low byte */
#define PCA9685_LED0_OFF_H  0x09    /**< LED0 off tick, high byte */

// etc all 16: LED15_OFF_H 0x45
#define PCA9685_ALLLED_ON_L 0xFA    /**< load all the LEDn_ON
    registers, low */
#define PCA9685_ALLLED_ON_H 0xFB    /**< load all the LEDn_ON
    registers, high */
#define PCA9685_ALLLED_OFF_L 0xFC    /**< load all the LEDn_OFF
    registers, low */
#define PCA9685_ALLLED_OFF_H 0xFD    /**< load all the LEDn_OFF
    registers,high */
#define PCA9685_PRESCALE     0xFE    /**< Prescaler for PWM output
    frequency */
#define PCA9685_TESTMODE      0xFF    /**< defines the test mode to be
    entered */

// MODE1 bits
#define MODE1_ALLCAL  0x01    /**< respond to LED All Call I2C-bus
    address */
#define MODE1_SUB3    0x02    /**< respond to I2C-bus subaddress 3 */
#define MODE1_SUB2    0x04    /**< respond to I2C-bus subaddress 2 */
#define MODE1_SUB1    0x08    /**< respond to I2C-bus subaddress 1 */
#define MODE1_SLEEP   0x10    /**< Low power mode. Oscillator off */
#define MODE1_AI      0x20    /**< Auto-Increment enabled */
#define MODE1_EXTCLK  0x40    /**< Use EXTCLK pin clock */
#define MODE1_RESTART 0x80    /**< Restart enabled */

// MODE2 bits
#define MODE2_OUTNE_0 0x01    /**< Active LOW output enable input */

```

```

#define MODE2_OUTNE_1 0x02    /**< Active LOW output enable input - high
                           impedance */
#define MODE2_OUTDRV 0x04    /**< totem pole structure vs open-drain */
#define MODE2_OCH     0x08    /**< Outputs change on ACK vs STOP */
#define MODE2_INVRT   0x10    /**< Output logic state inverted */

#define PCA9685_I2C_ADDRESS (0x40<<1)  /**< Default PCA9685 I2C Slave
                                         Address */
#define FREQUENCY_OSCILLATOR 25000000  /**< Int. osc. frequency in
                                         datasheet */

#define PCA9685_PRESCALE_MIN 3  /**< minimum prescale value */
#define PCA9685_PRESCALE_MAX 255 /**< maximum prescale value */

// Variable
typedef struct PCA9685_Driver {
    I2C_HandleTypeDef *hi2c;
    uint8_t Address;
} PCA9685_HandleTypeDef;

// function
void PCA9685_Init(PCA9685_HandleTypeDef *hpc9685,
                    I2C_HandleTypeDef *hi2c,
                    uint8_t addr,
                    uint8_t prescale);
void PCA9685_Reset(PCA9685_HandleTypeDef *hpc9685);
void PCA9685_Sleep(PCA9685_HandleTypeDef *hpc9685);
void PCA9685_Wakeup(PCA9685_HandleTypeDef *hpc9685);
void PCA9685_SetExtClk(PCA9685_HandleTypeDef *hpc9685, uint8_t
                       prescale);
void PCA9685_SetPWMFreq(PCA9685_HandleTypeDef *hpc9685, float freq);
uint8_t PCA9685_GetPWM(PCA9685_HandleTypeDef *hpc9685, uint8_t num);
void PCA9685_SetPWM(PCA9685_HandleTypeDef *hpc9685, uint8_t num,
                     uint16_t on, uint16_t off);

void PCA9685_SetOscillatorFrequency(PCA9685_HandleTypeDef *hpc9685,
                                     uint32_t freq);

void PCA9685_Write8(PCA9685_HandleTypeDef *hpc9685, uint8_t addr,
                     uint8_t d);
void PCA9685_Read8(PCA9685_HandleTypeDef *hpc9685, uint8_t addr,
                    uint8_t *data);

#endif /* PCA9685_H_ */

```

\\" PCA9685_Driver.c

```

/*
 * =====#
 * ##### How to use this driver #####
 * =====#
 * Set up:
 * #1: Run @ref PCA9685_Init()
 * #2: Run @ref PCA9685_SetOscillatorFrequency(), 27000000
 * #3: Run @ref PCA9685_SetPWMFreq(), 1600
 *
 * Let play: @ref PCA9685_SetPWM()
 * #1: GPIO
 * PCA9685_SetPWM(hPCA9685, pin, 4096, 0); // turns pin fully on
 * delay(100);
 * PCA9685_SetPWM(hPCA9685, pin, 0, 4096); // turns pin fully off
 *
 * #2: PWM
 * PCA9685_SetPWM(hPCA9685, pin, on, off);
 *
 */
#include "PCA9685Driver.h"
uint32_t _oscillator_freq;

void PCA9685_Init(PCA9685_HandleTypeDef *hPCA9685,
                   I2C_HandleTypeDef *hi2c,
                   uint8_t addr,
                   uint8_t prescale) {
    hPCA9685->Address = addr;
    hPCA9685->hi2c = hi2c;
    PCA9685_Reset(hPCA9685);
    if (prescale) {
        PCA9685_SetExtClk(hPCA9685, prescale);
    } else {
        // set a default frequency
        PCA9685_SetPWMFreq(hPCA9685, 1000);
    }
    // set the default internal frequency
    PCA9685_SetOscillatorFrequency(hPCA9685, FREQUENCY_OSCILLATOR);
}

void PCA9685_Reset(PCA9685_HandleTypeDef *hPCA9685) {
    PCA9685_Write8(hPCA9685, PCA9685_MODE1, MODE1_RESTART);
    HAL_Delay(10);
}

```

```

void PCA9685_Sleep(PCA9685_HandleTypeDef *h pca9685) {
    uint8_t awake;
    PCA9685_Read8(h pca9685, PCA9685_MODE1, &awake);
    uint8_t sleep = awake | MODE1_SLEEP; // set sleep bit high
    PCA9685_Write8(h pca9685, PCA9685_MODE1, sleep);
    HAL_Delay(5); // wait until cycle ends for sleep to be active
}

void PCA9685_Wakeup(PCA9685_HandleTypeDef *h pca9685) {
    uint8_t sleep;
    PCA9685_Read8(h pca9685, PCA9685_MODE1, &sleep);
    uint8_t wakeup = sleep & ~MODE1_SLEEP; // set sleep bit low
    PCA9685_Write8(h pca9685, PCA9685_MODE1, wakeup);
}

void PCA9685_SetExtClk(PCA9685_HandleTypeDef *h pca9685, uint8_t
prescale) {
    uint8_t oldmode;
    PCA9685_Read8(h pca9685, PCA9685_MODE1, &oldmode);
    uint8_t newmode = (oldmode & ~MODE1_RESTART) | MODE1_SLEEP; // sleep
    PCA9685_Write8(h pca9685, PCA9685_MODE1, newmode); // go to sleep,
        turn off internal oscillator

    // This sets both the SLEEP and EXTCLK bits of the MODE1 register
    to switch to
    // use the external clock.
    PCA9685_Write8(h pca9685, PCA9685_MODE1, (newmode |= MODE1_EXTCLK));

    PCA9685_Write8(h pca9685, PCA9685_PRESCALE, prescale); // set the
        prescaler

    HAL_Delay(5);
    // clear the SLEEP bit to start
    PCA9685_Write8(h pca9685, PCA9685_MODE1, (newmode & ~MODE1_SLEEP) |
        MODE1_RESTART | MODE1_AI);
}

void PCA9685_SetPWMFreq(PCA9685_HandleTypeDef *h pca9685, float freq) {
    // Range output modulation frequency is dependant on oscillator
    if (freq < 1) freq = 1;
    if (freq > 3500) freq = 3500; // Datasheet limit is
        3052=50MHz/(4*4096)

    float prescaleval = ((_oscillator_freq / (freq * 4096.0)) + 0.5) -
        1;
    if (prescaleval < PCA9685_PRESCALE_MIN)

```

```

prescaleval = PCA9685_PRESCALE_MIN;
if (prescaleval > PCA9685_PRESCALE_MAX)
    prescaleval = PCA9685_PRESCALE_MAX;
uint8_t prescale = (uint8_t)prescaleval;

uint8_t oldmode;
PCA9685_Read8(hpca9685, PCA9685_MODE1, &oldmode);
uint8_t newmode = (oldmode & ~MODE1_RESTART) | MODE1_SLEEP; // sleep
PCA9685_Write8(hpca9685, PCA9685_MODE1, newmode); // go to sleep
PCA9685_Write8(hpca9685, PCA9685_PRESCALE, prescale); // set the
    prescaler
PCA9685_Write8(hpca9685, PCA9685_MODE1, oldmode);
HAL_Delay(5);
// This sets the MODE1 register to turn on auto increment.
PCA9685_Write8(hpca9685, PCA9685_MODE1, oldmode | MODE1_RESTART |
    MODE1_AI);
}

void PCA9685_SetOscillatorFrequency(PCA9685_HandleTypeDefDef *hpca9685,
    uint32_t freq) {
    _oscillator_freq = freq;
}

void PCA9685_SetPWM(PCA9685_HandleTypeDefDef *hpca9685, uint8_t num,
    uint16_t on, uint16_t off) {
    uint8_t outputBuffer[5] = {PCA9685_LED0_ON_L + 4*num, on, (on >>
        8), off, (off >> 8)};
    HAL_I2C_Master_Transmit(hpca9685->hi2c, hpca9685->Address,
        outputBuffer, 5, 1);
}

/****************** Low level I2C interface */
void PCA9685_Write8(PCA9685_HandleTypeDefDef *hpca9685, uint8_t addr,
    uint8_t d) {
    HAL_I2C_Mem_Write(hpca9685->hi2c, hpca9685->Address, addr, 1, &d,
        1, 1);
}

void PCA9685_Read8(PCA9685_HandleTypeDefDef *hpca9685, uint8_t addr,
    uint8_t *data) {
    HAL_I2C_Mem_Read(hpca9685->hi2c, hpca9685->Address, addr, 1, data,
        1, 1);
}

```

Code điều khiển độ sáng bóng đèn trong file main.c

```

void userPCAIInit() {
    PCA9685_Init(&pcaHub, &hi2c2, PCA9685_I2C_ADDRESS, 0);
    PCA9685_SetOscillatorFrequency(&pcaHub, 27000000);
    PCA9685_SetPWMFreq(&pcaHub, 1600);
}

void sendInfoPWM() {
    uint16_t LEDOnTime;
    for (uint8_t channel = 0; channel < 6; channel++) {
        channelON[channel] = DELAY_LED - 1;
        LEDOnTime = intensity[channel / 2] * 4096 / 1000;
        channelOFF[channel] = (LEDOnTime + DELAY_LED > 4096
            ? LEDOnTime + DELAY_LED - 4096
            : LEDOnTime + DELAY_LED - 1);
        PCA9685_SetPWM(&pcaHub, channel, channelON[channel],
            channelOFF[channel]);
    }
}
void Controlled() {
    DecryptData();
    intensity[0] = info[0];
    intensity[1] = info[1];
    intensity[2] = info[2];
    sendInfoPWM();
    HAL_UART_Transmit(&huart2, controlLEDDone, sizeof(controlLEDDone),
        10);
    program = IDLE;
}

```

5.2.5 Nhận tín hiệu từ máy tính và thực hiện các thao tác điều khiển

Máy tính với VĐK STM32F411VETx đang giao tiếp với nhau bằng UART thông qua IC CP2102. Để VĐK nhận tín hiệu thì phải quy định trước số byte sẽ gửi, khi nhận đủ số byte này VĐK sẽ chạy hàm ngắn do người dùng định nghĩa. Kích thước gói thông tin nhóm đang quy định là 11 byte theo quy tắc như sau:

- Kiểm tra giao tiếp UART: **MT123123123**
- Đọc giá trị từ cảm biến: **MG123123123**
- Điều khiển độ sáng của 3 dàn đèn: **MLxxxxyyzzz** (xxx,yyy,zzz lần lượt là độ sáng của 3 dàn đèn theo mức độ từ 000 tới 999).

- Đặt giá trị toạ độ ban đầu của các dàn đèn: **MIxxxyyyzzz** (xxx, yyy, zzz lần lượt là toạ độ ban đầu của 3 dàn đèn).
- Gửi giá trị toạ độ mong muốn của các dàn đèn: **MTxxxyyyzzz** (xxx, yyy, zzz lần lượt là toạ độ mong muốn của 3 dàn đèn).

Code nhận tín hiệu trong file main.c:

```
// CHOOSE THE MANUAL CONTROL PROGRAM (It's still under UART interrupt
function)
void ManualControl() {
    switch (dataReceived[1]) {
        case 'T':
            program = TEST_UART;
            break;
        case 'I':
            program = SET_INIT_COOR;
            break;
        case 'A':
            program = SET_TARGET_COOR;
            break;
        case 'G':
            program = GET_SENSOR;
            break;
        case 'L':
            program = CONTROL_LED;
            break;
        case 'R':
            program = REINITSENSOR;
            break;
    }
}

// CHOOSE THE AUTO CONTROL PROGRAM (It's still under UART interrupt
function)
void AutoControl() {

}

// Select PROGRAM
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
    if (huart->Instance == huart2.Instance) {
        switch (dataReceived[0]) {
            case 'M':
                ManualControl();
                break;
```

```

        case 'A':
            AutoControl();
            break;
    }
    HAL_UART_Receive_IT(huart, dataReceived, SIZE_DATA);
}

}

int main(void)
{
    while (1)
    {
        /* USER CODE END WHILE */

        /* USER CODE BEGIN 3 */
        if (program == TEST_UART) {
            TestUART();
        } else if (program == SET_INIT_COOR) {
            SetInitCoorStepper();
        } else if (program == SET_TARGET_COOR) {
            SetTargetCoorStepper();
        } else if (program == RUN_STEPPER) {
            RunStepper();
        } else if (program == GET_SENSOR) {
            GetSensor();
        } else if (program == CONTROL_LED) {
            Controlled();
        } else if (program == REINITSENSOR) {
            HAL_UART_Transmit(&huart2, ReInitSensorMessage,
                sizeof(ReInitSensorMessage), 10);
            userSensorInit();
            program = IDLE;
        }
    }
    /* USER CODE END 3 */
}

```

5.2.6 Phần mềm điều khiển giao diện trên máy tính

Có rất nhiều cách để tạo giao diện khác nhau, mỗi ngôn ngữ lập trình như C++, C#, Python, Java đều cung cấp các thư viện hỗ trợ việc lập trình giao diện. Tuy nhiên nhóm lựa chọn WPF (Windows Presentation Foundation), một hệ thống API hỗ trợ việc xây dựng đồ họa trên nền Windows, sử dụng C# và xaml làm ngôn ngữ lập trình chính. WPF có các ưu điểm sau đây:

- Mới hơn và do đó phù hợp hơn với các tiêu chuẩn hiện tại.
- Microsoft đang sử dụng WPF cho rất nhiều ứng dụng mới, ví dụ: Visual Studio.
- WPF linh hoạt hơn, vì vậy bạn có thể làm nhiều việc hơn mà không phải viết hoặc mua các control mới.
- Khi bạn cần sử dụng các control của bên thứ 3, các nhà phát triển các control này có thể sẽ tập trung hơn vào WPF vì WPF mới hơn.
- XAML giúp dễ dàng tạo và chỉnh sửa GUI của bạn và cho phép công việc được phân chia giữa một nhà thiết kế (XAML) và một lập trình viên (C#, VB.NET, v.v.)
- Databinding, cho phép bạn có được một sự tách biệt hơn giữa data và layout
- Sử dụng tăng tốc phần cứng để vẽ GUI, để có hiệu suất tốt hơn
- WPF cho phép bạn tạo giao diện người dùng cho cả ứng dụng Windows và các ứng dụng web (Silverlight/XBAP)

```
\ \ MainWindows.xaml
<Window
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" x:Class="Serial_Communication_WPF.MainWindow"
    xmlns:ports="clr-namespace:System.IO.Ports;assembly=System"
    Icon="Data Resources\Serial.ico"
    Title="GUI Project" Height="450" Width="504">
    <Window.Resources>
        <ObjectDataProvider ObjectType="{x:Type ports:SerialPort}"
            MethodName="GetPortNames" x:Key="portNames"/>
        <XmlDataProvider x:Key="ComPorts" Source="CommsData.xml"
            XPath="/Comms/Ports" />
        <XmlDataProvider x:Key="ComSpeed" Source="CommsData.xml"
            XPath="/Comms/Baud" />
    </Window.Resources>
    <Grid>
        <Grid.Background>
            <ImageBrush ImageSource="wallpaper.jpg"/>
        </Grid.Background>
        <ComboBox x:Name="cmbComPorts" ItemsSource="{Binding
            Source={StaticResource portNames}}"
            SelectionChanged="cmbComPorts_SelectionChanged"
            HorizontalAlignment="Left" Margin="10,10,0,0"
            VerticalAlignment="Top" Width="120"/>
    </Grid>
</Window>
```

```

<TextBlock x:Name="tbStatus" HorizontalAlignment="Left"
    Margin="10,37,0,0" TextWrapping="Wrap" Text="Disconnected"
    VerticalAlignment="Top" Foreground="Red"/>
<Button x:Name="btnConnect" Content="Connect"
    HorizontalAlignment="Left" Margin="135,10,0,0"
    VerticalAlignment="Top" Width="75" Click="btnConnect_Click"/>
<TabControl HorizontalAlignment="Left" Margin="10,58,0,10"
    Width="476" Background="{x:Null}">
    <TabItem x:Name="tabManual" Header="Manual">
        <Grid>
            <Grid.Background>
                <SolidColorBrush Color="Black" Opacity="0.4"/>
            </Grid.Background>
            <Button x:Name="btnTestUART" Content="Test UART"
                HorizontalAlignment="Left" Margin="347,35,0,0"
                VerticalAlignment="Top" Width="111"
                Click="btnTestUART_Click" Height="30"/>
            <RichTextBox x:Name="Commdata"
                HorizontalAlignment="Left" Height="108"
                Margin="10,35,0,0" VerticalAlignment="Top"
                Width="316">
                <FlowDocument>
                    <Paragraph>
                        <Run Text="" />
                    </Paragraph>
                </FlowDocument>
            </RichTextBox>

            <Button x:Name="btnGetSensors" Content="Get Sensors
                Data" HorizontalAlignment="Left"
                Margin="347,70,0,0" VerticalAlignment="Top"
                Width="111" Height="30"
                Click="btnGetSensors_Click"/>
            <TextBlock x:Name="tbTerminal"
                HorizontalAlignment="Center"
                Margin="130,10,247,0" TextWrapping="Wrap"
                Text="Data Received" VerticalAlignment="Top"
                Height="20" Foreground="White"/>
            <Slider x:Name="slLevelLamp0"
                HorizontalAlignment="Left" Margin="144,148,0,0"
                VerticalAlignment="Top" Width="316" Maximum="999"
                TickFrequency="10" TickPlacement="BottomRight"
                IsSnapToTickEnabled="True"
                ValueChanged="slLevelLamp0_ValueChanged"
                PreviewMouseUp="slLevelLamp_PreviewMouseUp"/>
            <Slider x:Name="slLevelLamp1"

```

```

        HorizontalAlignment="Left" Margin="144,177,0,0"
        VerticalAlignment="Top" Width="316" Maximum="999"
        TickFrequency="10" TickPlacement="BottomRight"
        IsSnapToTickEnabled="True"
        ValueChanged="slLevelLamp1_ValueChanged"
        PreviewMouseUp="slLevelLamp_PreviewMouseUp"/>
    <Slider x:Name="slLevelLamp2"
        HorizontalAlignment="Left" Margin="144,206,0,0"
        VerticalAlignment="Top" Width="316" Maximum="999"
        TickFrequency="10" TickPlacement="BottomRight"
        IsSnapToTickEnabled="True"
        ValueChanged="slLevelLamp2_ValueChanged"
        PreviewMouseUp="slLevelLamp_PreviewMouseUp"/>

    <TextBlock x:Name="tbLevelLamp0"
        HorizontalAlignment="Left" Margin="10,149,0,0"
        TextWrapping="Wrap" Text="Level Lamp 0"
        VerticalAlignment="Top" Height="18"
        Foreground="White"/>
    <TextBlock x:Name="tbLevelLamp1"
        HorizontalAlignment="Left" Margin="10,178,0,0"
        TextWrapping="Wrap" Text="Level Lamp 1"
        VerticalAlignment="Top" Height="18"
        Foreground="White"/>
    <TextBlock x:Name="tbLevelLamp2"
        HorizontalAlignment="Left" Margin="10,207,0,0"
        TextWrapping="Wrap" Text="Level Lamp 2"
        VerticalAlignment="Top" Height="18"
        Foreground="White"/>
    <TextBox x:Name="txtValueL0"
        HorizontalAlignment="Left" Height="19"
        Margin="84,148,0,0" TextWrapping="Wrap" Text="0"
        VerticalAlignment="Top" Width="55"/>
    <TextBox x:Name="txtValueL1"
        HorizontalAlignment="Left" Height="19"
        Margin="84,177,0,0" TextWrapping="Wrap" Text="0"
        VerticalAlignment="Top" Width="55"/>
    <TextBox x:Name="txtValueL2"
        HorizontalAlignment="Left" Height="19"
        Margin="84,206,0,0" TextWrapping="Wrap" Text="0"
        VerticalAlignment="Top" Width="55"/>
    <TextBlock x:Name="tbStepper"
        HorizontalAlignment="Left" Margin="109,235,0,0"
        TextWrapping="Wrap" Text="Location of
        Light-Beams" VerticalAlignment="Top"
        Foreground="White"/>

```

```

<TextBox x:Name="txtLocStep0"
    HorizontalAlignment="Left" Height="23"
    Margin="10,280,0,0" TextWrapping="Wrap" Text="0"
    VerticalAlignment="Top" Width="57"
    PreviewTextInput="txtLocStep_PreviewTextInput"/>
<TextBox x:Name="txtLocStep1"
    HorizontalAlignment="Left" Height="23"
    Margin="78,280,0,0" TextWrapping="Wrap" Text="0"
    VerticalAlignment="Top" Width="57"
    PreviewTextInput="txtLocStep_PreviewTextInput"/>
<TextBox x:Name="txtLocStep2"
    HorizontalAlignment="Left" Height="23"
    Margin="144,280,0,0" TextWrapping="Wrap" Text="0"
    VerticalAlignment="Top" Width="57"
    PreviewTextInput="txtLocStep_PreviewTextInput"/>
<TextBlock x:Name="tbBeam0"
    HorizontalAlignment="Left" Margin="18,260,0,0"
    TextWrapping="Wrap" Text="Beam 0"
    VerticalAlignment="Top" Width="40" Height="16"
    Foreground="White"/>
<TextBlock x:Name="tbBeam1"
    HorizontalAlignment="Left" Margin="86,260,0,0"
    TextWrapping="Wrap" Text="Beam 1"
    VerticalAlignment="Top" Width="40" Height="16"
    Foreground="White"/>
<TextBlock x:Name="tbBeam2"
    HorizontalAlignment="Left" Margin="153,260,0,0"
    TextWrapping="Wrap" Text="Beam 2"
    VerticalAlignment="Top" Width="40" Height="16"
    Foreground="White"/>
<Button x:Name="btnSetTarget" Content="Set Target
    Location" HorizontalAlignment="Left"
    Margin="211,280,0,0" VerticalAlignment="Top"
    Width="115" Height="23"
    Click="btnSetTarget_Click"/>
<Button x:Name="btnsetCurrent" Content="Set Current
    Location" HorizontalAlignment="Left"
    Margin="332,280,0,0" VerticalAlignment="Top"
    Width="126" Height="23"
    Click="btnsetCurrent_Click"/>
<Button x:Name="btnReInitSensor" Content="Reset
    Sensors" HorizontalAlignment="Left"
    Margin="347,105,0,0" VerticalAlignment="Top"
    Width="111" Height="30"
    Click="btnReInitSensor_Click"/>
</Grid>

```

```

        </TabItem>
        <TabItem x:Name="tabAuto" Header="Auto">
            <Grid Background="#FFE5E5E5">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="5*"/>
                    <ColumnDefinition Width="108*"/>
                </Grid.ColumnDefinitions>
            </Grid>
        </TabItem>
    </TabControl>
</Grid>
</Window>

```

```

\\ MainWindows.xaml.cs
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.IO.Ports;
using System.Threading;
using System.Windows.Threading;
using System.Text.RegularExpressions;

namespace Serial_Communication_WPF
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        #region variables
        //Richtextbox
        FlowDocument mcFlowDoc = new FlowDocument();
        Paragraph para = new Paragraph();
        //Serial
        SerialPort serial = new SerialPort();
        string recieived_data;
    }
}

```

```

Boolean flagConnect = false;
// level LAMP
int levelLamp0;
int levelLamp1;
int levelLamp2;
#endregion

#region MainWindow and UART Function
public MainWindow()
{
    InitializeComponent();

}

private void cmbComPorts_SelectionChanged(object sender,
    SelectionChangedEventArgs e)
{
    var selectedComboItem = sender as ComboBox;
    string name = selectedComboItem.SelectedItem as string;
}
private void btnConnect_Click(object sender, RoutedEventArgs e)
{
    if (flagConnect == false)
    {
        //Sets up serial port
        string portName = cmbComPorts.SelectedValue as string;
        serial.PortName = portName;
        serial.BaudRate = 115200;
        serial.Handshake = System.IO.Ports.Handshake.None;
        serial.Parity = Parity.None;
        serial.DataBits = 8;
        serial.StopBits = StopBits.Two;
        serial.ReadTimeout = 200;
        serial.WriteTimeout = 50;
        serial.Open();
        flagConnect = true;

        //Sets button State and Creates function call on data
        //recieved
        btnConnect.Content = "Disconnect";
        this.tbStatus.Text = "Connected!";
        this.tbStatus.Foreground = new
            SolidColorBrush(Colors.Green);
        serial.DataReceived += new
            System.IO.Ports.SerialDataReceivedEventHandler(Recieve);

    }
}

```

```

        else
        {
            try // just in case serial port is not open could also
                 be acheived using if(serial.IsOpen)
            {
                serial.Close();
                btnConnect.Content = "Connect";
                this.tbStatus.Text = "Disconnected!";
                this.tbStatus.Foreground = new
                    SolidColorBrush(Colors.Red);
                flagConnect = false;
            }
            catch
            {
            }
        }
    }
#endregion

#region Recieving

private delegate void UpdateUiTextDelegate(string text);
private void Recieve(object sender,
                     System.IO.Ports.SerialDataReceivedEventArgs e)
{
    // Collecting the characters received to our 'buffer'
    // (string).
    recieved_data = serial.ReadExisting();
    Dispatcher.Invoke(DispatcherPriority.Send, new
        UpdateUiTextDelegate(WriteData), recieved_data);
}
private void WriteData(string text)
{
    // Assign the value of the recieved_data to the RichTextBox.
    para.Inlines.Add(text);
    mcFlowDoc.Blocks.Add(para);
    Commdata.Document = mcFlowDoc;
    Commdata.ScrollToEnd();
}

#endregion

#region Sending with Button
private void btnTestUART_Click(object sender, RoutedEventArgs e)
{
    SerialCmdSend("MT123123123");
}

```

```

    }
    private void btnGetSensors_Click(object sender, RoutedEventArgs
        e)
    {
        SerialCmdSend("MG123123123");
    }
    public void SerialCmdSend(string data)
    {
        if (serial.IsOpen)
        {
            try
            {
                // Send the binary data out the port
                byte[] hexstring = Encoding.ASCII.GetBytes(data);
                //There is a intermittent problem that I came across
                //If I write more than one byte in succesion without a
                //delay the PIC i'm communicating with will Crash
                //I expect this id due to PC timing issues ad they are
                //not directley connected to the COM port the solution
                //Is a ver small 1 millisecound delay between
                //chracters
                foreach (byte hexval in hexstring)
                {
                    byte[] _hexval = new byte[] { hexval }; // need
                        to convert byte to byte[] to write
                    serial.Write(_hexval, 0, 1);
                    Thread.Sleep(1);
                }
            }
            catch (Exception)
            {
                //para.Inlines.Add("Failed to SEND" + data + "\n" +
                    ex + "\n");
                //mcFlowDoc.Blocks.Add(para);
                //Commdata.Document = mcFlowDoc;
            }
        }
        else
        {
        }
    }
}

#endregion

#region Form Controls

```

```

private void Close_Form(object sender, RoutedEventArgs e)
{
    if (serial.IsOpen) serial.Close();
    this.Close();
}
private void Max_size(object sender, RoutedEventArgs e)
{
    if (this.WindowState != WindowState.Maximized)
        this.WindowState = WindowState.Maximized;
    else this.WindowState = WindowState.Normal;
}
private void Min_size(object sender, RoutedEventArgs e)
{
    if (this.WindowState != WindowState.Minimized)
        this.WindowState = WindowState.Minimized;
    else this.WindowState = WindowState.Normal;
}
private void Move_Window(object sender, MouseButtonEventArgs e)
{
    this.DragMove();
}
#endregion

#region Lights Level
private void slLevelLamp0_ValueChanged(object sender,
    RoutedEventArgs e)
{
    var slide = sender as Slider;
    txtValueL0.Text = slide.Value.ToString();
}
private void slLevelLamp1_ValueChanged(object sender,
    RoutedEventArgs e)
{
    var slide = sender as Slider;
    txtValueL1.Text = slide.Value.ToString();
}

private void slLevelLamp2_ValueChanged(object sender,
    RoutedEventArgs e)
{
    var slide = sender as Slider;
    txtValueL2.Text = slide.Value.ToString();
}

private void slLevelLamp_PreviewMouseUp(object sender,
    MouseButtonEventArgs e)

```

```

{
    var slide = sender as Slider;
    if (slide.Name == "slLevelLamp0")
    {
        levelLamp0 = Convert.ToInt16(slide.Value);
    }
    else if (slide.Name == "slLevelLamp1")
    {
        levelLamp1 = Convert.ToInt16(slide.Value);
    }
    else if (slide.Name == "slLevelLamp2")
    {
        levelLamp2 = Convert.ToInt16(slide.Value);
    }
    char[] lightMessage = new char[11];
    lightMessage[0] = 'M';
    lightMessage[1] = 'L';
    // Level Lamp0
    lightMessage[4] = (char)((levelLamp0 % 10) + 48);
    lightMessage[3] = (char)((levelLamp0 / 10 % 10) + 48);
    lightMessage[2] = (char)((levelLamp0 / 100 % 10) + 48);
    // Level Lamp1
    lightMessage[7] = (char)((levelLamp1 % 10) + 48);
    lightMessage[6] = (char)((levelLamp1 / 10 % 10) + 48);
    lightMessage[5] = (char)((levelLamp1 / 100 % 10) + 48);
    // Level Lamp2
    lightMessage[10] = (char)((levelLamp2 % 10) + 48);
    lightMessage[9] = (char)((levelLamp2 / 10 % 10) + 48);
    lightMessage[8] = (char)((levelLamp2 / 100 % 10) + 48);
    string s = new string(lightMessage);
    //WriteData(s);
    SerialCmdSend(s);
}
private void btnReInitSensor_Click(object sender,
    RoutedEventArgs e)
{
    SerialCmdSend("MR123123123");
}
#endregion

#region Stepper
private void btnSetTarget_Click(object sender, RoutedEventArgs e)
{
    char[] TargetMessage = new char[11];
    int tarStep0, tarStep1, tarStep2;
    Int32.TryParse(txtLocStep0.Text, out tarStep0);

```

```

        Int32.TryParse(txtLocStep1.Text, out tarStep1);
        Int32.TryParse(txtLocStep2.Text, out tarStep2);

        TargetMessage[0] = 'M';
        TargetMessage[1] = 'A';
        // Level Lamp0
        TargetMessage[4] = (char)((tarStep0 % 10) + 48);
        TargetMessage[3] = (char)((tarStep0 / 10 % 10) + 48);
        TargetMessage[2] = (char)((tarStep0 / 100 % 10) + 48);
        // Level Lamp1
        TargetMessage[7] = (char)((tarStep1 % 10) + 48);
        TargetMessage[6] = (char)((tarStep1 / 10 % 10) + 48);
        TargetMessage[5] = (char)((tarStep1 / 100 % 10) + 48);
        // Level Lamp2
        TargetMessage[10] = (char)((tarStep2 % 10) + 48);
        TargetMessage[9] = (char)((tarStep2 / 10 % 10) + 48);
        TargetMessage[8] = (char)((tarStep2 / 100 % 10) + 48);
        string s = new string(TargetMessage);
        SerialCmdSend(s);
    }

private void btnSetCurrent_Click(object sender, RoutedEventArgs
    e)
{
    char[] CurrentMessage = new char[11];
    int curStep0, curStep1, curStep2;
    Int32.TryParse(txtLocStep0.Text, out curStep0);
    Int32.TryParse(txtLocStep1.Text, out curStep1);
    Int32.TryParse(txtLocStep2.Text, out curStep2);
    CurrentMessage[0] = 'M';
    CurrentMessage[1] = 'I';
    // Level Lamp0
    CurrentMessage[4] = (char)((curStep0 % 10) + 48);
    CurrentMessage[3] = (char)((curStep0 / 10 % 10) + 48);
    CurrentMessage[2] = (char)((curStep0 / 100 % 10) + 48);
    // Level Lamp1
    CurrentMessage[7] = (char)((curStep1 % 10) + 48);
    CurrentMessage[6] = (char)((curStep1 / 10 % 10) + 48);
    CurrentMessage[5] = (char)((curStep1 / 100 % 10) + 48);
    // Level Lamp2
    CurrentMessage[10] = (char)((curStep2 % 10) + 48);
    CurrentMessage[9] = (char)((curStep2 / 10 % 10) + 48);
    CurrentMessage[8] = (char)((curStep2 / 100 % 10) + 48);
    string s = new string(CurrentMessage);
    SerialCmdSend(s);
}

```

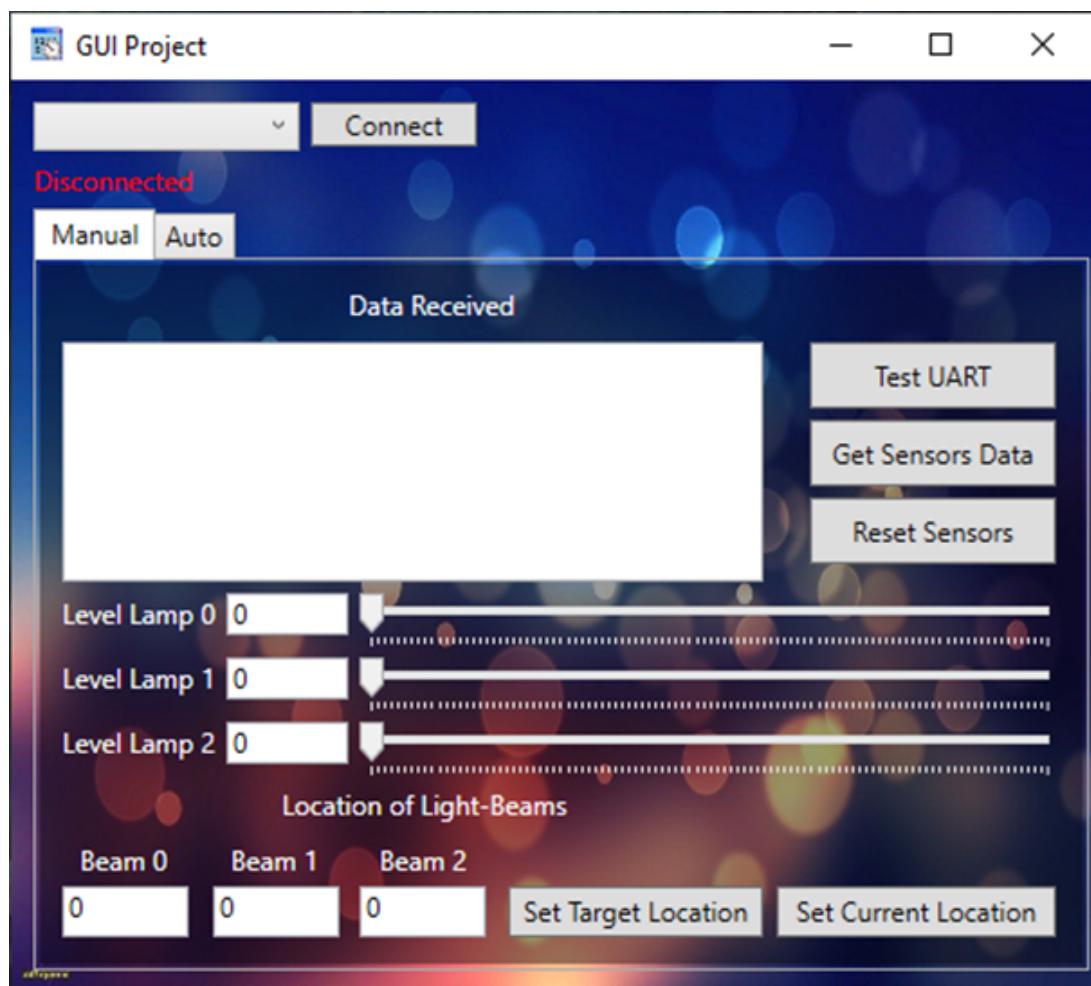
```

private void txtLocStep_PreviewTextInput(object sender,
    TextCompositionEventArgs e)
{
    Regex regex = new Regex("[^0-9]+");
    e.Handled = regex.IsMatch(e.Text);
}

#endregion
}

```

Giao diện của ứng dụng điều khiển (Hình 5.17):



Hình 5.17: Giao diện GUI của ứng dụng điều khiển

Các trình điều khiển có trên giao diện:

- Combobox chọn port kết nối với module UART CP2102. Nút nhấn Connect để kết nối với port đã chọn. Có textblock hiển thị trạng thái kết nối.

- Textbox “Data received” để nhận dữ liệu từ VĐK và hiển thị lên màn hình.
- Nút nhấn TestUART để kiểm tra kết nối UART có ổn định hay không. Nếu ổn định thì trên textbox “Data Received” sẽ hiển thị dòng chữ: "Welcome to Project: Automatic light intensity balance".
- Nút GetSensors Data để gửi lệnh đọc cảm biến. Dữ liệu trả về sẽ hiển thị lên textbox.
- Nút Reset Sensors để gửi lệnh reset và khởi tạo lại các cảm biến trong trường hợp dữ liệu trả về bị sai sót, không chính xác.
- 3 Slide Level Lamp để điều chỉnh độ sáng của 3 thanh dàn đèn.
- Nút Set Current Location để gửi tọa độ ban đầu của các thanh dàn đèn.
- Nút Set Target Location để gửi tọa độ mong muốn cho các thanh dàn đèn.

Chương 6

TỔNG KẾT, ĐÁNH GIÁ VÀ HƯỚNG PHÁT TRIỂN

6.1 Tổng kết, đánh giá

Sau hơn 4 tháng thực hiện đề tài với nhiều sự cố gắng nỗ lực của các thành viên trong nhóm cũng như sự hướng dẫn tận tình của thầy Nguyễn Vũ Lân, đồ án đã sơ bộ hoàn thành với những yêu cầu được đề ra. Bên cạnh đó còn tồn tại những yêu cầu đặt ra nhưng chưa thực hiện được.

Đạt được:

- Nhóm đã thống nhất và đưa ra được bản thiết kế của mô hình, lịch thực hiện của từng tuần. Trong quá trình thực hiện luôn thực hiện đúng những nhiệm vụ đã được đặt ra cũng như đúng với bản thiết kế.
- Nhóm đã hoàn thành được khung sườn của đồ án và chạy theo chế độ Manual. Điều khiển các dàn đèn đến các vị trí mong muốn với cường độ ánh sáng được thiết lập.
- Về phương diện giao diện điều khiển và lập trình, nhóm đã thiết kế được giao diện điều khiển các dàn đèn, hiển thị được các thông số của cảm biến ánh sáng.
- Đã viết được app điều khiển và source code cho chương trình hoàn thiện và chi tiết.

Chưa đạt được:

- Cơ chế auto tự điều chỉnh dàn đèn và cường độ ánh sáng khi tín hiệu ánh sáng bị ảnh hưởng chưa hoàn thành.
- Phần bao ngoài của hệ thống chưa được hoàn thiện.
- Phần chóa đèn cho góc chiếu lớn hơn chưa có triển khai được.

6.2 Hướng phát triển

Đề tài mới dừng ở việc thiết kế mô hình hệ thống nhỏ nên các thông số chưa phản ánh đúng hoàn toàn so với môi trường thực tế. Vì vậy để hệ thống phát triển hơn trong đời sống, nhóm đề xuất một số phương án cải thiện đề tài:

- Sử dụng cảm biến chuyên dụng để đo độ chính xác cao.
- Sử dụng thêm nhiều loại cảm biến để biết chính xác vị trí của từng dàn đèn và tránh va đụng giữa các dàn đèn.
- Tối ưu phương thức di chuyển của các dàn đèn.
- Điều khiển chính xác từng đèn ở từng vị trí.
- Phát triển được phần mềm quản lý đèn chuyên nghiệp hơn.
- Tìm ra được thuật toán di chuyển và cân bằng.
- Đề xuất tự tắt đèn khi trong phòng không có người hoặc tự mở nếu có người bước vào phòng.

TÀI LIỆU THAM KHẢO

[1] Trịnh Chất – Lê Văn Uyển, Tính toán thiết kế hệ dẫn động cơ khí tập 1, Trường Đại học Sư Phạm Kỹ Thuật Thành phố Hồ Chí Minh.

[2] Trịnh Chất – Lê Văn Uyển, Tính toán thiết kế hệ dẫn động cơ khí tập 2, Trường Đại học Sư Phạm Kỹ Thuật Thành phố Hồ Chí Minh.

[3] Nguyễn Đình Phú, Giáo trình Vi điều khiển PIC, Trường Đại học Sư Phạm Kỹ Thuật Thành phố Hồ Chí Minh.