

**ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG ĐIỆN - ĐIỆN TỬ**



BÁO CÁO
BÀI TẬP LỚN MÔN HỌC HỆ
THỐNG NHÚNG VÀ THIẾT KẾ
GIAO TIẾP NHÚNG

Đề tài:

HỆ THỐNG ĐIỀU KHIỂN CÁP TREO KHU
DU LỊCH

Sinh viên thực hiện: **PHÍ MẠNH TOÀN 20193142**
 PHẠM ĐỨC NAM 20193034
 TRẦN CÔNG TRƯỜNG 20193162
 HOÀNG VŨ AN 20192674
Giảng viên hướng dẫn: **TS. PHẠM VĂN TIẾN**

Hà Nội, 6 - 2023

LỜI NÓI ĐẦU

Cáp treo/ hệ thống cáp treo tên tiếng Anh là Aerial tramway, Sky tram, Cable cars hay Gondola lift là hệ thống phương tiện di chuyển trên không, đặc biệt dùng để vượt qua địa điểm khó khăn, hiểm trở. Trước đây cáp treo thường được sử dụng để vận chuyển đồ đạc, hàng hóa nhưng dần dần với sự phát triển của xã hội và nhu cầu của con người, cáp treo ngày càng được sử dụng nhiều như một phương tiện di chuyển trên không. Đặc biệt trong ngành du lịch thắng cảnh, nhờ có cáp treo mà lĩnh vực này trở nên cực kì phát triển trên toàn thế giới.

Cáp treo cung cấp trải nghiệm độc đáo cho du khách, cho phép họ chiêm ngưỡng khung cảnh tuyệt đẹp của khu du lịch từ một góc nhìn không thể có được bằng cách khác. Du khách có thể tận hưởng cảnh quan tự nhiên, địa danh nổi tiếng và các điểm tham quan từ độ cao, mang đến sự hứng thú và kích thích tâm hồn. Cáp treo cũng sẽ giúp cho du khách di chuyển nhanh chóng và thuận tiện từ điểm này đến điểm khác trong khu du lịch, giúp du khách tiếp cận các khu vực khó tiếp cận như các ngọn núi cao, hẻm lách hoặc các địa hình khắc nghiệt. Nó mở ra cơ hội cho du khách khám phá những vùng đất hẻm lách và có khí hậu đặc biệt mà không thể tiếp cận bằng các phương tiện khác.

Thêm nữa cáp treo được xây dựng và vận hành với các tiêu chuẩn an toàn cao, đảm bảo an toàn tuyệt đối cho du khách khi sử dụng. Các biện pháp an toàn như khóa an toàn, hệ thống cảnh báo và kiểm tra định kỳ đảm bảo rằng hệ thống hoạt động một cách tin cậy và không gây nguy hiểm cho người dùng.

Từ nhu cầu thực tiễn và lợi ích của xã hội ta thấy hệ thống cáp treo là rất cần thiết trong các khu du lịch. Chính vì vậy, nhóm chúng em quyết định chọn đề tài mô phỏng hệ thống điều khiển cáp treo du lịch với hệ thống đón trả khách tự động và điều khiển tốc độ hài hòa.

Em xin cảm ơn thầy Phạm Văn Tiến đã giúp chúng em lên ý tưởng, cung cấp cho chúng em những kiến thức phù hợp để có thể hoàn thành bài tập này.

LỜI NÓI ĐẦU.....	2
PHẦN I. BÁO CÁO NHÓM.....	6
CHƯƠNG 1. XÁC ĐỊNH CHỈ TIÊU KỸ THUẬT	6
1.1 Yêu cầu chức năng.....	6
1.2 Yêu cầu phi chức năng	6
CHƯƠNG 2. MÔ HÌNH HOÁ HỆ THỐNG	6
2.1. Nguyên lý hoạt động	6
2.2. Mô hình hóa hệ thống.....	7
2.1.1 Mô hình hóa hệ thống bằng FSM.....	7
2.1.2 Mô hình hóa bằng Use Case Diagram	8
CHƯƠNG 3. THIẾT KẾ HỆ THỐNG.....	9
3.1 Tìm hiểu link kiện	9
3.1.1 Board mạch NodeMCU Esp8266 V3.....	9
3.1.2 Module điều khiển động cơ L298N	11
3.1.3 Màn hình LCD tích hợp module giao tiếp I2C	13
3.1.4 Động cơ DC.....	14
3.1.5 Cảm biến đo tốc độ Encoder V2	14
3.1.6 Cảm biến đo khoảng cách siêu âm SRF05	15
3.2 Thiết kế mạch	17
3.2.1 Vẽ và mô phỏng trên phần mềm Proteus	17
3.2.2 Mạch thực tế.....	19
CHƯƠNG 4. THIẾT KẾ PHẦN MỀM.....	19
4.1 Điều khiển động cơ	19
4.2 Lập trình điều khiển sử dụng Blynk	22
CHƯƠNG 5. TRIỂN KHAI, THỬ NGHIỆM TOÀN HỆ THỐNG	28
5.1 Mô hình cấp treo	28
CHƯƠNG 6. ĐÁNH GIÁ KẾT QUẢ VÀ ĐỊNH HƯỚNG PHÁT TRIỂN.....	33
6.1 Đánh giá kết quả	33
6.2 Định hướng phát triển.....	33
PHẦN II. BÁO CÁO CÁ NHÂN.....	34
1. PHÍ MẠNH TOÀN.....	34

2. HOÀNG VŨ AN	37
2.1 Board mạch NodeMCU Esp8266 V3	37
2.2 Module L298N	39
2.3 Tìm hiểu và mô phỏng cảm biến khoảng cách HC-SR04 hiển thị LCD I2C.....	40
3 TRẦN CÔNG TRƯỜNG.....	42
3.1 Thiết kế phần mềm chạy bằng Blynk.....	42
3.2 Triển khai thử nghiệm toàn hệ thống	45
4 PHẠM ĐỨC NAM	49
4.1 Xác lập chỉ tiêu kĩ thuật	49
4.2 Nguyên lý hoạt động của hệ thống cấp treo	49
4.3 Mô hình hóa hệ thống bằng FSM và Use Case Diagram	49
KẾT LUẬN	53
TÀI LIỆU THAM KHẢO.....	54

Hình 2. 1 Hệ thống cấp treo khi về trạm	7
Hình 2. 2 Sơ đồ FSM hệ thống cấp treo	8
Hình 2. 3 Use Case Diagram	9

Hình 3. 1 ESP8266	10
Hình 3. 2 Sơ đồ chân ESP8266 NodeMCU.....	10
Hình 3. 3 Sơ đồ chân L298N	12
Hình 3. 4 Màn hình LCD 1602.....	13
Hình 3. 5 Module I2C.....	13
Hình 3. 6 Sơ đồ kết nối LCD1602 với module I2C	14
Hình 3. 7 Động cơ giảm tốc vàng 2 trục	14
Hình 3. 8 Cảm biến đo tốc độ Encoder V2 và đĩa quay encoder 20 xung	15
Hình 3. 9 Cảm biến siêu âm SRF05	16
Hình 3. 10 SRF05 Timing Diagram	16

PHẦN I. BÁO CÁO NHÓM

CHƯƠNG 1. XÁC ĐỊNH CHỈ TIÊU KỸ THUẬT

1.1 Yêu cầu chức năng

Hệ thống cáp treo khu du lịch bao gồm:

- Cabin
- Hệ thống dây cáp (rope system)
- Hệ thống khớp nối (grip system)
- Cột trụ nâng cáp
- Bộ máy kéo cáp (động cơ)
- Hệ thống điều khiển

Thiết kế hệ thống cáp treo với những chức năng sau đây:

- Có nút bấm khởi động, dừng hệ thống
- Cảm biến khoảng cách, khi khoảng cách từ cabin tới trạm $\leq 100\text{m}$ thì giảm tốc độ để về trạm đón khách
- Tăng tốc độ sau khi đón khách xong
- Cửa đóng, mở tự động để đón trả khách
- Dừng hệ thống khi gặp sự cố (thời tiết, kĩ thuật,...)

Một vài thông số kĩ thuật tiêu biểu:

- Số người: khoảng 15 người lớn/cabin
- Vận tốc di chuyển: 5m/s
- Chiều dài trung bình: 5km

1.2 Yêu cầu phi chức năng

Bên cạnh những yêu cầu về chức năng, hệ thống cáp treo cần đạt được những yêu cầu phi chức năng sau:

- Dễ điều khiển
- An toàn tuyệt đối cho người và thiết bị
- Xử lí các tình huống phát sinh chuẩn xác
- Tiết kiệm điện năng
- Dễ dàng nâng cấp chỉnh sửa, bảo trì
- Đảm bảo tính thời gian thực

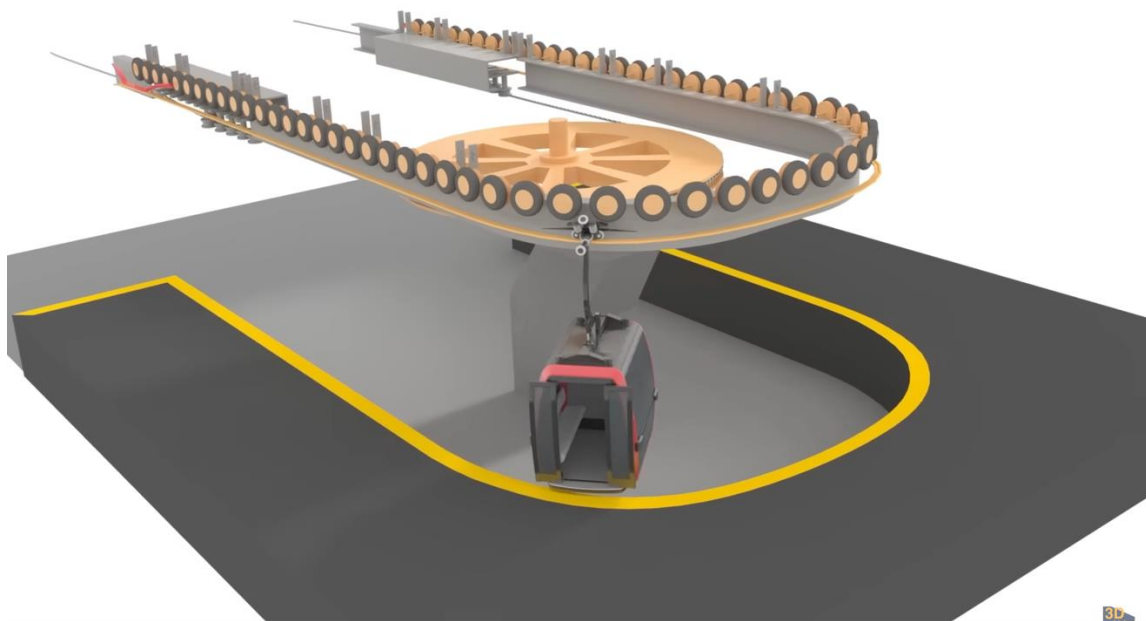
CHƯƠNG 2. MÔ HÌNH HOÁ HỆ THỐNG

2.1. Nguyên lý hoạt động

Trong thực tế, hệ thống cáp treo bao gồm một hoặc nhiều sợi cáp thép chắc chắn và cabin (hay gọi là gondola) để chở khách lên xuống. Các cabin được treo lên trên cáp và di chuyển theo đường dây cáp được cố định trên các trụ hoặc cột. Cáp treo sử dụng động cơ điện để cung cấp năng lượng và hệ thống truyền động để chuyển

động cho cáp và cabin. Động cơ tạo ra lực đẩy để kéo cáp di chuyển và đưa cabin lên trên hoặc hạ xuống theo hướng và tốc độ được điều khiển. Hệ thống cáp treo có trạm điều khiển tại các điểm xuất phát và điểm đích. Tại các trạm này, người điều khiển hoặc hệ thống tự động sẽ kiểm soát hoạt động của cáp treo, bao gồm khởi động, dừng và điều chỉnh tốc độ di chuyển của cabin.

Khi cabin tới gần trạm, với cảm biến, cabin sẽ giảm tốc độ lại để cho khách lên và xuống. Cửa hầu như luôn tự động đóng và mở. Cabin được điều khiển thông qua các thiết bị đầu cuối bằng bánh xe quay hoặc bằng hệ thống xích. Để được tăng tốc và giảm tốc so với tốc độ đường thẳng, các cabin được dẫn động bằng các lớp quay nhanh hơn (hoặc chậm hơn) dần dần cho đến khi chúng đạt đến tốc độ đường thẳng hoặc tốc độ cuối. Trên các hệ thống cũ hơn, cáp treo được tăng tốc thủ công bởi người điều khiển.



Hình 2. 1 Hệ thống cáp treo khi về trạm

Tuy nhiên khi thực hiện, do tài nguyên và khả năng còn hạn chế nên chúng em chỉ thực hiện thiết kế mô hình theo nguyên lý đơn giản như sau:

- Mô hình gồm 2 trạm lên và xuống, dây cáp và cabin
- Mạch điều khiển động cơ DC đặt ở trạm trên, sẽ tạo ra lực để kéo dây cáp di chuyển, từ đó cabin gắn trên dây sẽ di chuyển lên và xuống
- Khi tới các trạm, chúng em sẽ tiến hành giảm tốc độ động cơ để giảm tốc độ cabin, mô phỏng việc đón trả khách tại trạm.

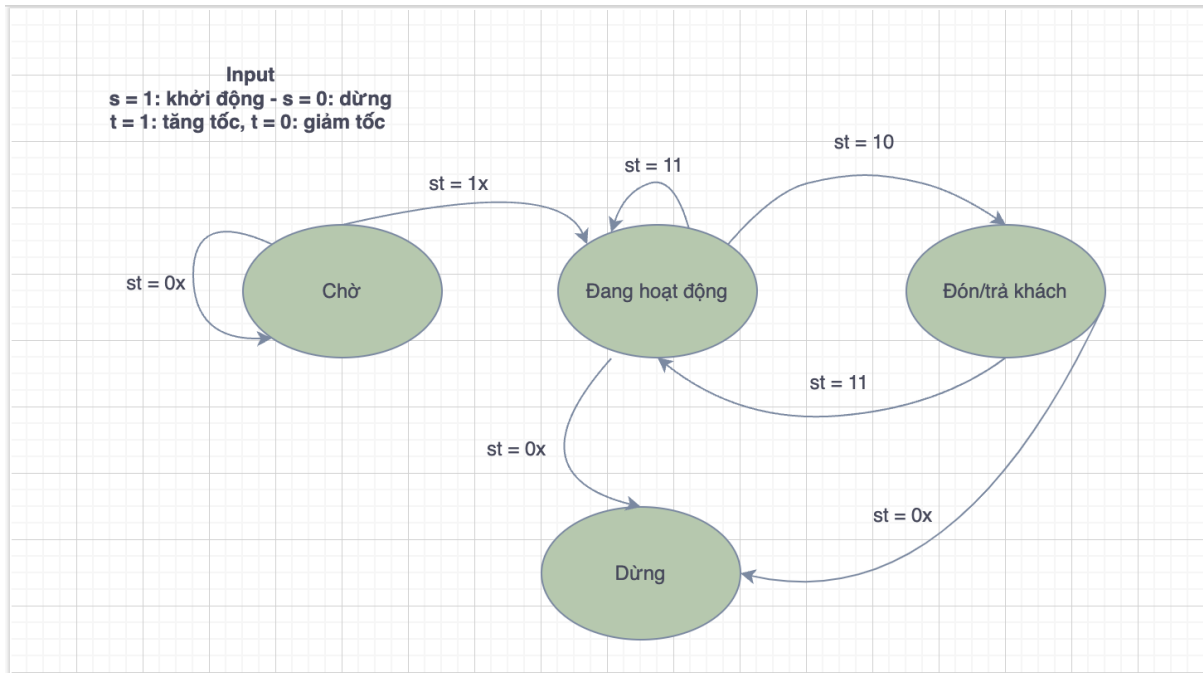
Sau đây chúng em sẽ mô hình hóa hệ thống để làm rõ hơn hoạt động của hệ thống cáp treo trên.

2.2. Mô hình hóa hệ thống

2.1.1 Mô hình hóa hệ thống bằng FSM

FSM (Finite state machine) - Máy trạng thái hữu hạn là một mô hình toán học biểu diễn trạng thái của hệ, trong đó bao gồm một số trạng thái hữu hạn, quá trình chuyển đổi giữa các trạng thái đó và các hành động. Từ mỗi trạng thái, máy có thể chuyển đổi qua 1 số trạng thái cố định khác, dựa trên các sự kiện, input.

Với mô tả nguyên lý hoạt động ở trên, chúng em sẽ mô hình hóa hệ thống bằng FSM. Trong đó có các trạng thái chính của hệ thống cáp treo là: Chờ, Đang hoạt động, Đón/Trả khách và Dừng, cùng với các input là các sự kiện khởi động/dừng, tăng tốc và giảm tốc.

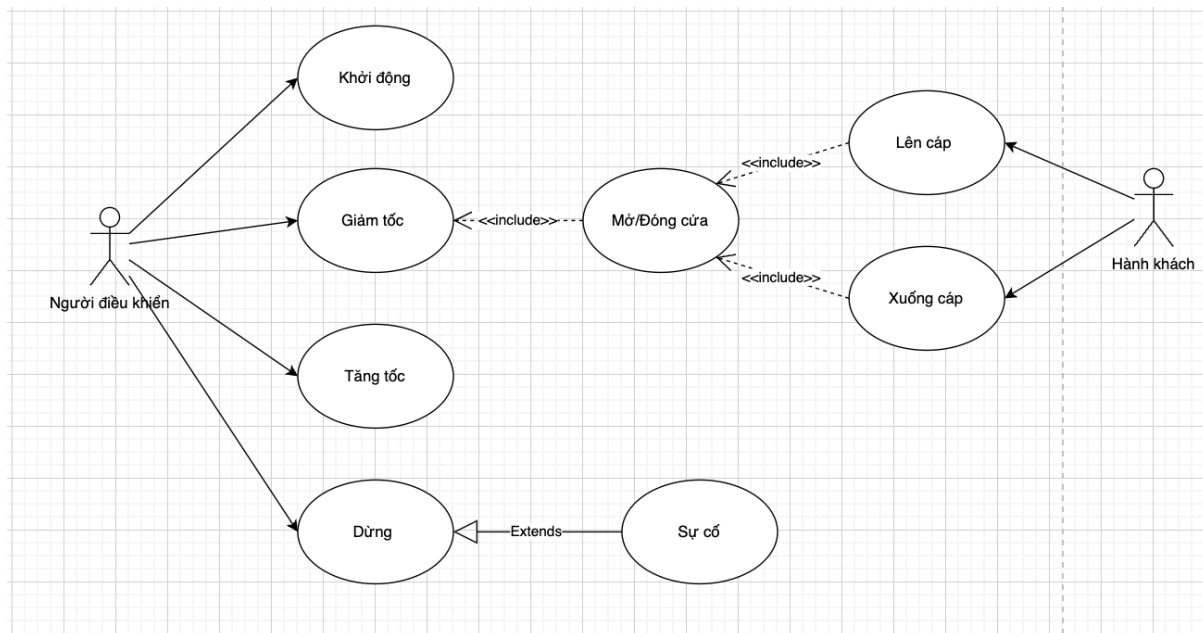


Hình 2. 2 Sơ đồ FSM hệ thống cáp treo

2.1.2 Mô hình hóa bằng Use Case Diagram

Biểu đồ ca sử dụng (Use case diagram), là một biểu đồ mô tả về các tương tác có thể xảy ra giữa người dùng (user) và hệ thống (system). Biểu đồ này bao gồm các use case cũng như các loại người dùng khác nhau của hệ thống. Các use case được thể hiện bằng hình tròn hoặc hình elip, còn tác nhân (actor) thường được biểu diễn dưới dạng hình người.

Với hệ thống cáp treo khu du lịch ta sẽ có hai tác nhân chính là người điều khiển hệ thống và khách hành. Người điều khiển có các use case vận hành hệ thống cáp treo còn hành khách sẽ chỉ có hai use case là lên cáp treo và xuống cáp treo.



Hình 2. 3 4Use Case Diagram

CHƯƠNG 3. THIẾT KẾ HỆ THỐNG

3.1 Tìm hiểu link kiện

3.1.1 Board mạch NodeMCU Esp8266 V3

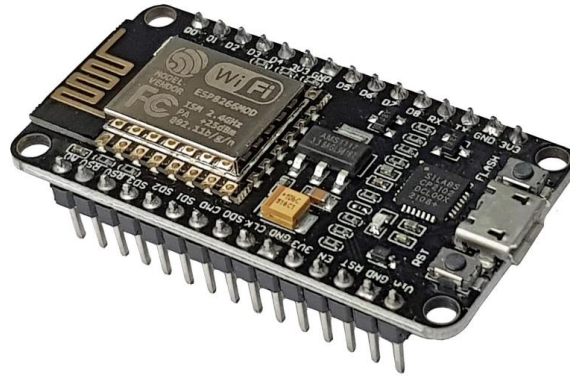
ESP8266 là một module Wi-Fi với khả năng kết nối Internet và được tích hợp sẵn trên một số board nhúng như NodeMCU, Wemos, và ESP-01. ESP8266 có thể hoạt động như một điểm truy cập (access point), một client kết nối đến một điểm truy cập khác, hoặc cả hai đều được. Nó được sử dụng rộng rãi trong các ứng dụng IoT (Internet of Things) như cảm biến thông minh, hệ thống kiểm soát thiết bị, hoặc các ứng dụng điều khiển từ xa. Module này có giá thành rẻ và rất dễ sử dụng, cùng với đó là khả năng tương thích với nhiều loại vi điều khiển khác nhau.

ESP8266 NodeMCU là một nền tảng IoT mã nguồn mở, được xây dựng trên ESP8266. NodeMCU cung cấp một bộ SDK để lập trình cho ESP8266 bằng ngôn ngữ Lua hoặc C++. Với các tính năng như Wi-Fi, GPIO, ADC, I2C, SPI, PWM và một số tính năng khác, NodeMCU ESP 8266 được sử dụng rộng rãi trong các ứng dụng IoT như kiểm soát thiết bị, thu thập dữ liệu và giao tiếp với các thiết bị khác

Thông số kỹ thuật sản phẩm :

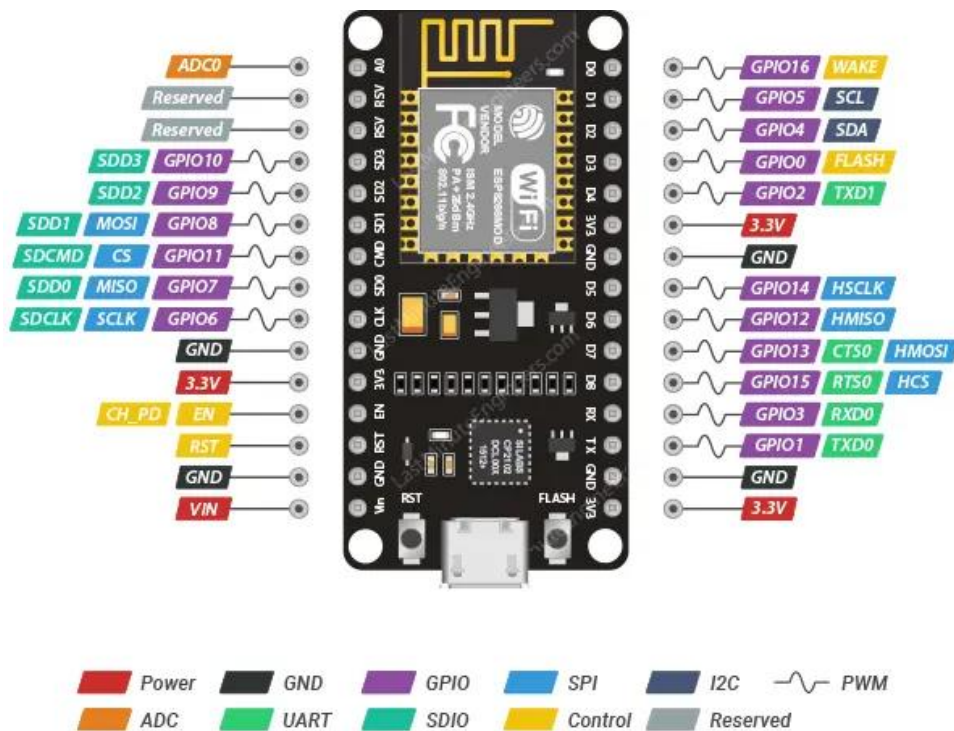
- Microcontroller: ESP8266EX
- Điện áp hoạt động: 3.3V DC
- Số chân I/O: 17 chân GPIO
- Kết nối mạng: WiFi 802.11 b/g/n
- Giao diện mạng: TCP/IP
- Đồng hồ thời gian thực (RTC): không tích hợp
- Bộ nhớ trong: 4MB

- RAM: 80KB
- Cổng nạp: Micro-USB
- Hỗ trợ các giao thức: MQTT, CoAP, HTTP/HTTPS
- Kích thước: 49 x 24.5 x 13mm



Hình 3. 1 ESP8266

Sơ đồ chân ESP8266 NodeMCU:



ESP8266 NodeMCU Pinout

Last Minute ENGINEERS.com

Hình 3. 2 Sơ đồ chân ESP8266 NodeMCU

Dưới đây là một số thiết bị ngoại vi và các chân I/O quan trọng trên ESP8266 NodeMCU:

17 chân GPIO	Được sử dụng để đọc dữ liệu từ các cảm biến, điều khiển các thiết bị đầu ra, hoặc giao tiếp với các thiết bị khác như LED, động cơ, nút nhấn, v.v.
1 kênh ADC	1 kênh ADC có độ chính xác 10 bit theo công nghệ SAR ADC.
2 giao tiếp UART	2 giao tiếp UART hỗ trợ điều khiển dòng dữ liệu.
4 đầu ra PWM	4 chân PWM để điều khiển tốc độ động cơ hoặc độ sáng của đèn LED.
2 giao tiếp SPI và 1 giao tiếp I2C	2 giao tiếp SPI và một giao tiếp I2C để kết nối các cảm biến và thiết bị ngoại vi khác.
Giao tiếp I2S	Một giao tiếp I2S để thêm âm thanh vào dự án của bạn.

3.1.2 Module điều khiển động cơ L298N

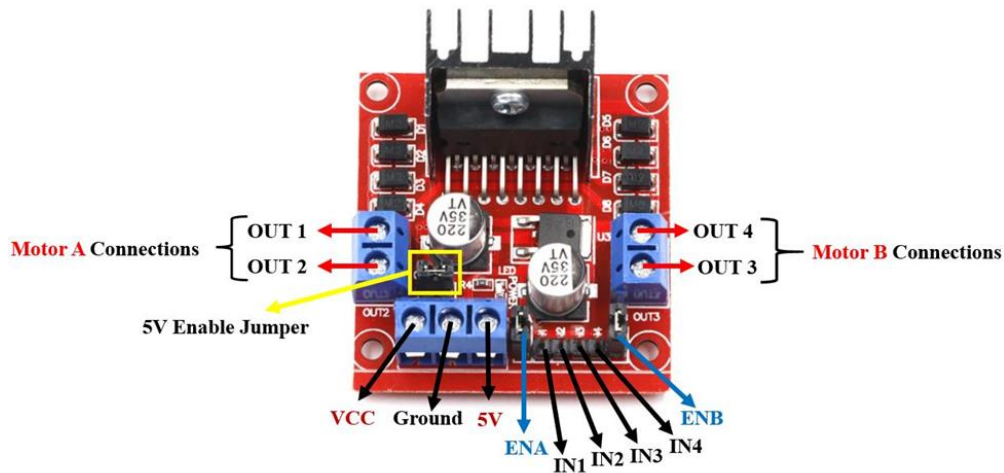
L298N là module điều khiển động cơ trong các xe DC và động cơ bước, rất dễ sử dụng với các bộ vi điều khiển và cũng tương đối rẻ.

Module có một IC điều khiển động cơ L298N, một bộ điều chỉnh điện áp 5V 78M05, 5V Enable Jumper, đèn LED nguồn, tản nhiệt, điện trở và tụ điện, tất cả được tích hợp trong một mạch. Module L298N có thể điều khiển tối đa 4 động cơ DC hoặc 2 động cơ DC với khả năng điều khiển hướng và tốc độ.

Thông số kỹ thuật:

- Module điều khiển: L298N
- Chip điều khiển: Cặp H-bridge L298N
- Công suất tối đa: 25W
- Điện áp tối đa cấp cho động cơ: 46V
- Dòng tối đa cấp cho động cơ: 2A
- Điện áp hoạt động của IC: 5-35V

- Dòng điện hoạt động IC: 2A



Hình 3. 3 Sơ đồ chân L298N

ENA điều khiển tốc độ của động cơ A và ENB điều khiển tốc độ của động cơ B. Nếu cả hai chân đều ở trạng thái logic CAO (5V), thì cả hai động cơ đều BẬT và quay ở tốc độ tối đa. Nếu cả hai chân đều ở trạng thái logic THẤP (tiếp đất), thì cả hai động cơ đều TẮT. Thông qua chức năng PWM, chúng ta cũng có thể kiểm soát tốc độ của động cơ. Bảng dưới đây minh họa các tín hiệu logic cần thiết để điều khiển Động cơ A:

ENA Pin State	Motor Action
1 (HIGH)	ON
0 (LOW)	OFF

Các chân điều khiển hướng là bốn chân đầu vào (IN1, IN2, IN3, IN4). Với hai chân IN1 và IN2 là đầu vào của động cơ A:

IN1	IN2	Motor Action
1 (HIGH)	1	OFF
1	0 (LOW)	Backward
0	1	Forward
0	0	OFF

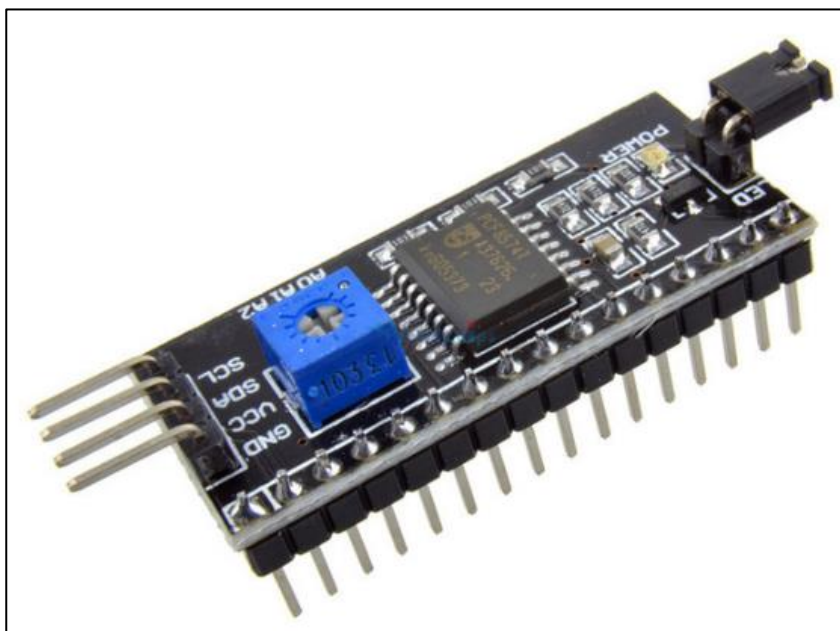
3.1.3 Màn hình LCD tích hợp module giao tiếp I2C



Hình 3. 4 Màn hình LCD 1602

Màn hình LCD 16x2 có 16 chân dữ liệu trong đó:

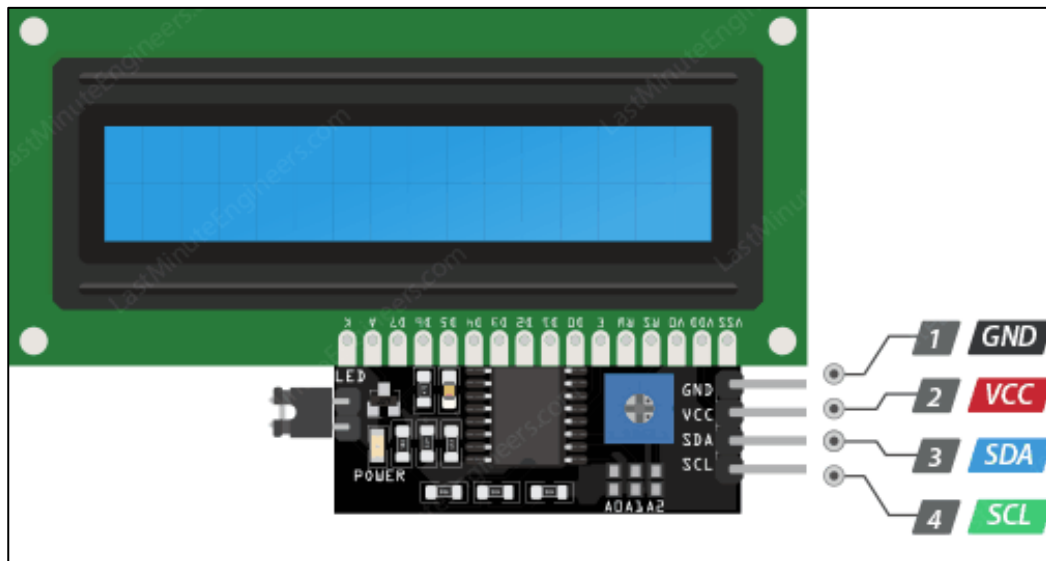
- 8 chân dữ liệu (D0- D7)
- 3 chân điều khiển RS, RW, EN
- Chân VE điều khiển độ sáng của màn hình
- VSS, VDD lần lượt là các chân GND, nguồn cho LCD hoạt động
- Chân A, K là các chế độ đèn nền cho LCD



Hình 3. 5 Module I2C

Do LCD có quá nhiều chân gây khó khăn trong quá trình kết nối với vi điều khiển nên sẽ được tích hợp với module I2C để giảm tải số chân kết nối. Giao thức I2C

chỉ sử dụng 2 chân dữ liệu là SDA và SCL. Module I2C này hỗ trợ các loại LCD sử dụng driver HD44780 (LCD16x2, LCD 20x4).



Hình 3. 6 Sơ đồ kết nối LCD1602 với module I2C

Trong hệ thống này, màn hình LCD sẽ được sử dụng để hiển thị các thông số như tốc độ vòng quay của trục, khoảng cách từ cabin tới trạm để có thể điều khiển động cơ tăng tốc, giảm tốc một cách tự động.

3.1.4 Động cơ DC

Động cơ DC có gắn giảm tốc 2 trục với dải điện áp hoạt động từ 3-12V. Số vòng/1 phút: 125 vòng/phút tại 3V, 208 vòng/phút tại 5V.



Hình 3. 7 Động cơ giảm tốc vàng 2 trục

3.1.5 Cảm biến đo tốc độ Encoder V2

Để đo được tốc độ của trục quay, nhóm chúng em tích hợp cảm biến đo tốc encoder V2, gắn đĩa quay vào trục quay để đo tốc độ.



Hình 3. 8 Cảm biến đo tốc độ Encoder V2 và đĩa quay encoder 20 xung

Cảm biến bao gồm 1 mắt phát và 1 mắt thu hồng ngoại đặt cách nhau qua 1 khe hở. Khi ánh sáng từ mắt phát đi được tới mắt thu (xuyên qua lỗ đĩa của encoder) thì sẽ có tín hiệu mức cao phát ra khỏi chân OUT, khi bị che lại thì chân OUT phát ra tín hiệu mức thấp.

Đĩa quay gồm có 20 lỗ, vậy công thức tính số vòng/phút: $RPM = (Số\ xung/20) \times 60$

Công thức tính tốc độ của trục quay:

$$v = \frac{Số\ xung}{20} \times 2 \times Bán\ kính\ đĩa\ quay \times \pi (m/s)$$

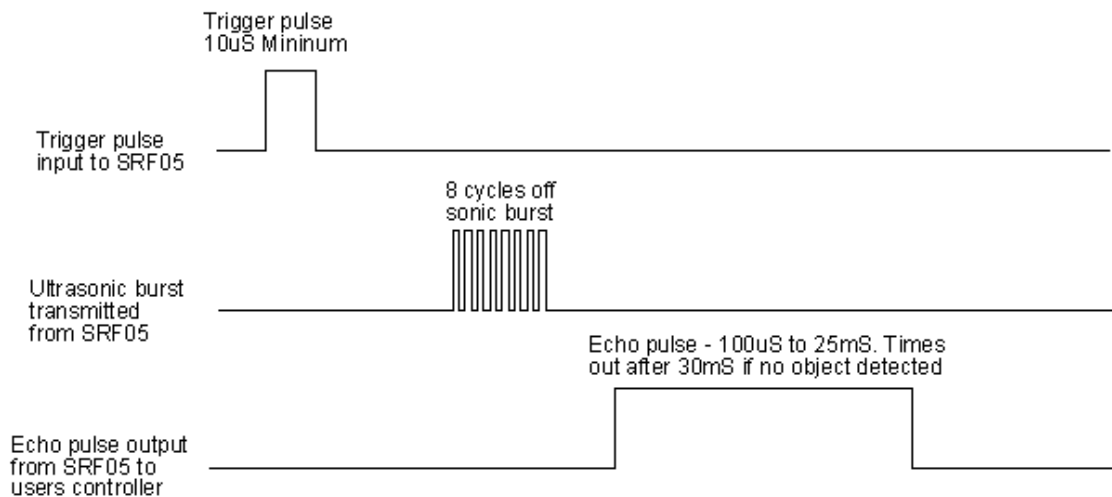
3.1.6 Cảm biến đo khoảng cách siêu âm SRF05

Do đề tài này chỉ dừng lại ở mức làm mô hình cỡ nhỏ nên cảm biến đo khoảng cách nhóm chúng em chọn là SRF05 (phạm vi đo khoảng cách <4,5m). Thông số kỹ thuật:



Hình 3. 9 Cảm biến siêu âm SRF05

Cảm biến SRF05 là một loại cảm biến đo khoảng cách dựa trên nguyên lý thu phát siêu âm. Cảm biến gồm một bộ phát và một bộ thu sóng siêu âm. Sóng siêu âm từ đầu phát truyền đi trong không khí, gặp vật cản sẽ phản xạ ngược lại và được đầu thu ghi lại. Vận tốc truyền âm thanh trong không khí là một giá trị được xác định trước, ít thay đổi (cỡ 3.10^8 m/s). Do đó, nếu xác định được khoảng thời gian từ lúc phát sóng đến lúc nhận được sóng phản xạ tới sẽ tính được khoảng cách từ vật cản đến cảm biến.



Hình 3. 100 SRF05 Timing Diagram

Cảm biến SRF05 có 2 chân TRIGGER và ECHO riêng biệt, khi chân MODE để trống, SRF05 sẽ sử dụng cả 2 chân chức năng TRIGGER và ECHO cho việc điều khiển hoạt động của cảm biến. Từ biểu đồ thời gian của chế độ hoạt động ta thấy để đo khoảng cách, đầu tiên ta phát 1 xung cỡ 5 us từ chân TRIG, sau đó cảm biến sẽ tạo ra xung HIGH ở chân ECHO cho đến khi nhận được xung phản xạ ở chân này. Chiều rộng của xung sẽ bằng với thời gian sóng siêu âm được phát từ cảm biến quay trở lại.

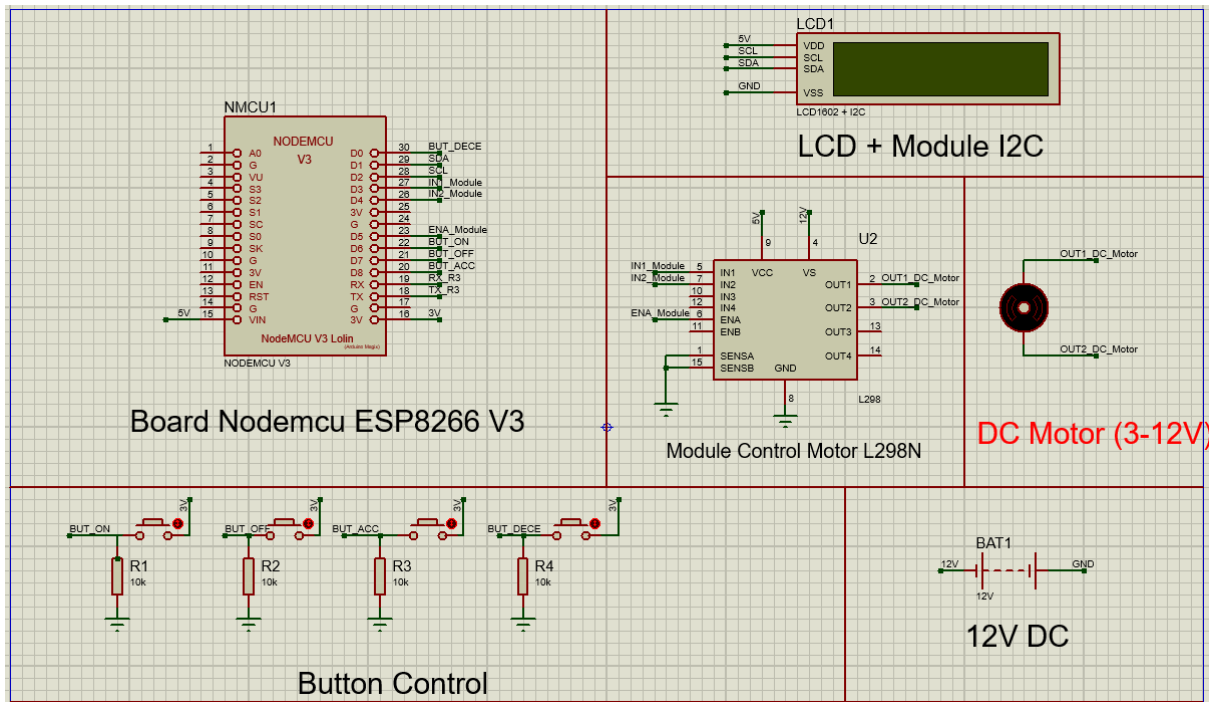
Tốc độ của âm thanh trong không khí tương đương $29,412 \text{ us/cm} = 3.108 \text{ m/s}$. Khi có khoảng cách từ cảm biến tới vật cản được tính bằng công thức:

$$d = \frac{\text{Thời gian từ lúc phát đến lúc thu } (\mu\text{s})}{29,412}$$

3.2 Thiết kế mạch

3.2.1 Vẽ và mô phỏng trên phần mềm Proteus

3.2.1.1 Mạch điều khiển động cơ sử dụng module L298N



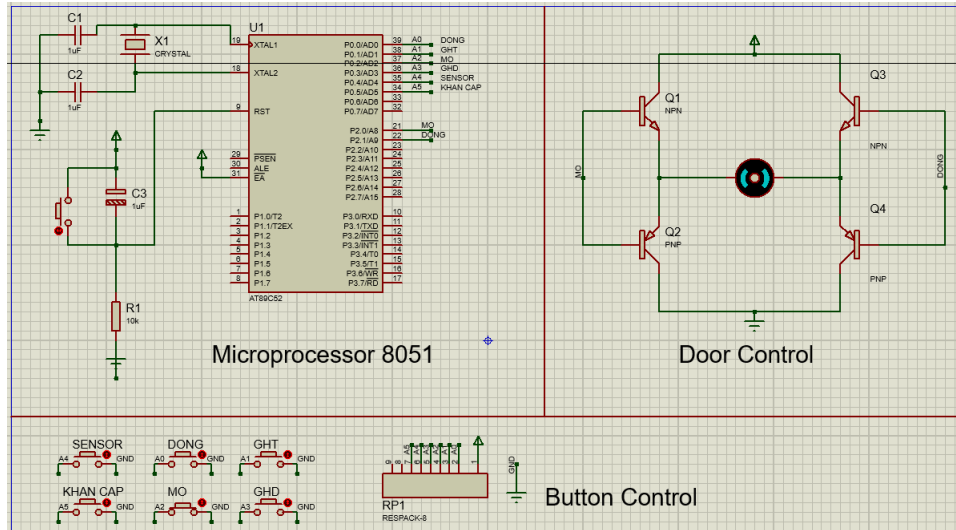
Hình 3. 111 Mạch điều khiển động cơ

Trên đây là sơ đồ nguyên lý khối mạch điều khiển động cơ sử dụng Board Nodemcu ESP8266 V3 giao tiếp với các ngoại vi như LCD sử dụng module I2C, module điều khiển động cơ L298N, các nút bấm điều khiển trạng thái của hệ thống.

- LCD được kết nối sẵn với module I2C từ đó giảm được số chân giao tiếp với Board mạch chính, thu gọn chỉ còn 4 chân VCC, GND, SDA, SCL.
- Board mạch Nodemcu giao tiếp với LCD theo giao thức I2C với 2 chân GPIO D1, D2 lần lượt theo datasheet được sử dụng là các chân SDA và SCL.
- Các chân D3, D4 lần lượt là các chân IN1 và IN2 điều khiển chiều quay của động cơ kết nối với module L298N.
- Chân D5 kết nối với chân ENA của L298N để điều khiển giá trị PWM- điều khiển tốc độ của động cơ.
- Các nút bấm ON, OFF, tăng tốc, giảm tốc lần lượt được kết nối với các chân GPIO D0, D6, D7, D8
- Module điều khiển động cơ L298N được cấp nguồn adapter 12V, điều khiển động cơ DC ở ngõ ra công A.

3.2.1.2 Mạch đóng mở cửa tự động dùng cảm biến và động cơ DC

Trong hệ thống cấp treo thì phần cửa cabin được thiết kế đóng mở tự động, kết hợp nhiều loại cảm biến để có thể đón trả khách tự động. Nhóm chúng em tìm hiểu và đưa ra một mô hình tổng quát mô tả chung một hệ thống đóng mở cửa như sau:

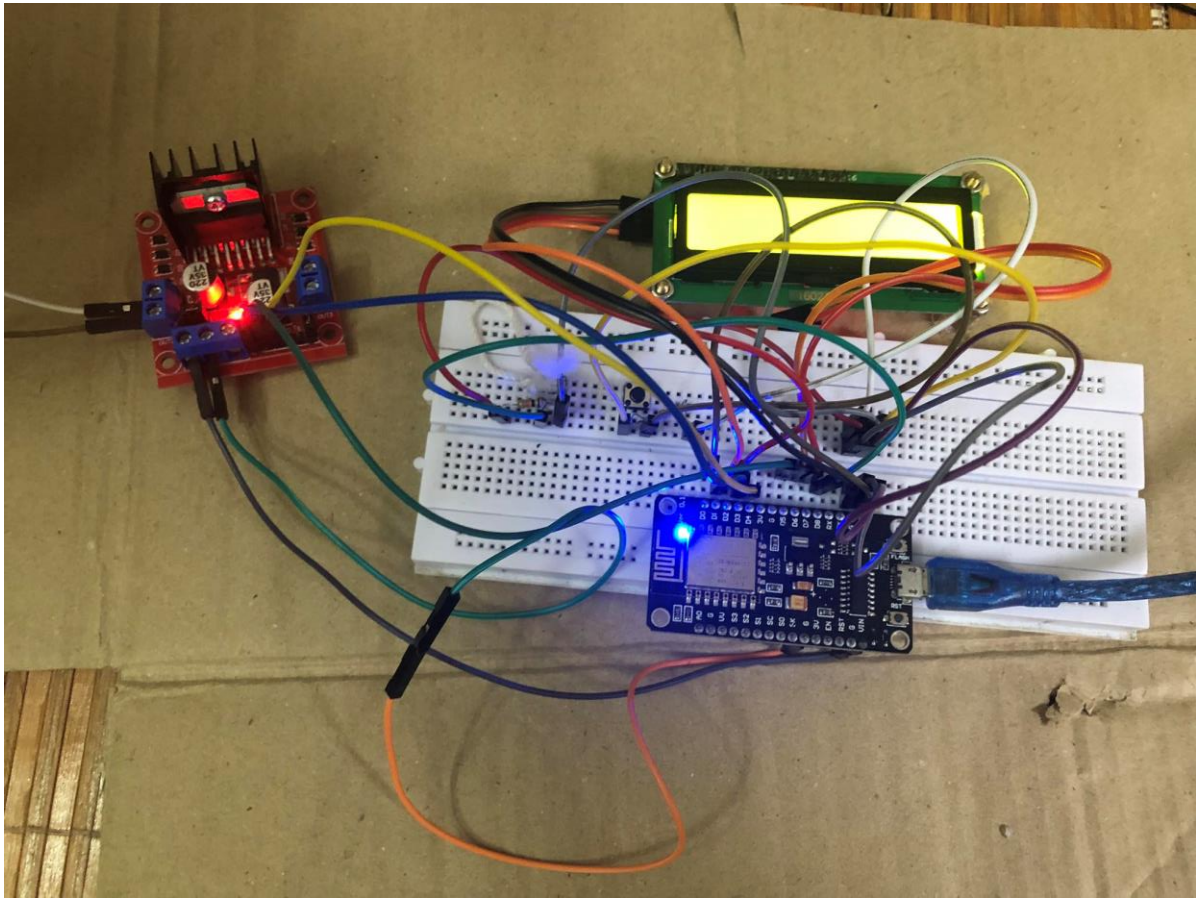


Hình 3. 122 Mạch nguyên lý đóng mở cửa tự động

Chúng em đưa ra mô hình mạch trên kết hợp giữa vi điều khiển 8051 và khối điều khiển động cơ gồm 4 transistor BJT. Nguyên lý hoạt động như sau:

- Cửa mở khi tín hiệu Mở và Sensor mức 1, khi gặp giới hạn trên thì motor dừng.
- Cửa đóng khi tín hiệu Đóng và Sensor mức 1, khi gặp giới hạn dưới thì dừng.
- Trong quá trình đóng hoặc mở nếu gặp tín hiệu Khẩn cấp, Motor dừng.

3.2.2 Mạch thực tế



Hình 3. 133 Mạch nguyên lý đóng mở cửa tự động

CHƯƠNG 4. THIẾT KẾ PHẦN MỀM

Trong phần này chúng em sẽ lập trình giao tiếp trình điều khiển động cơ L298N với ESP8266 NodeMCU. Chúng em sẽ điều khiển on/off hệ thống và tăng giảm tốc độ động cơ.

4.1 Điều khiển động cơ

Với nguyên lý hoạt động của hệ thống mô phỏng và các sơ đồ use case, FSM ở trên, thiết kế phần mềm điều khiển động cơ DC bằng module điều khiển động cơ L298N và NodeMCU ESP8266.

- Trước tiên, kết nối các chân điều khiển của L298N với ESP8266:

```
int ENA = D5;  
int IN1 = D3;  
int IN2 = D4;  
const int BUTTON_ON = D8;  
const int BUTTON_OFF = D0;
```

- Trong hàm setup(), chúng ta sẽ cấu hình các chân điều khiển đã kết nối ở trên thành các chân đầu ra và nút nhấn là các chân đầu vào bằng hàm pinMode():

```
pinMode(ENA, OUTPUT);
pinMode(IN1, OUTPUT);
pinMode(IN2, OUTPUT);
pinMode(BUTTON_ON, INPUT);
pinMode(BUTTON_OFF, INPUT);
```

Đồng thời, khởi tạo màn hình LCD, bật đèn nền và in tên hệ thống lên màn hình LCD:

```
lcd.init();
lcd.backlight();
lcd.print("Cap treo nhom 2");
```

Cuối cùng bằng cách sử dụng hàm digitalWrite(), chúng ta sẽ thiết lập các chân đầu vào ở trạng thái THẤP để ban đầu động cơ tắt:

```
digitalWrite(IN1, LOW);
digitalWrite(IN2, LOW);
```

- Có hai hàm để tăng và giảm tốc độ là speedUp() và slowDown():

```
void speedUp() {
    for (int i = 155; i <= 255; i++) {
        analogWrite(ENA, i);
        delay(20);
    }
}

void slowDown() {
    for (int i = 255; i >= 155; i--) {
        analogWrite(ENA, i);
        delay(20);
    }
}
```

Điều chỉnh tốc độ động cơ bằng cách tăng giảm tín hiệu PWM trên chân ENA. Tín hiệu PWM sẽ có giá trị từ 0 – 255. Qua nhiều lần mô phỏng và chạy thử mô hình, vì là động cơ DC cỡ nhỏ chúng em nhận thấy khi đặt tín hiệu PWM ở giá trị tối đa là 255 thì hệ thống cáp treo sẽ chạy với tốc độ hài hòa, còn khi giảm tín hiệu xuống khoảng 155 thì sẽ chạy rất chậm, mô phỏng đúng với cơ chế của cáp treo khi về trạm để đón/trả khách. Chính vì vậy mới có các thông số như ở trên.

Bằng cách sử dụng vòng lặp for, chúng ta sẽ giảm dần tốc độ của cáp treo để khi về trạm khách có thể lên và xuống, và sau đó khi khách lên thì sẽ tăng tốc độ để di chuyển.

- Hàm loop():

```

void loop() {
  int Push_button_on_state = digitalRead(BUTTON_ON);
  int Push_button_off_state = digitalRead(BUTTON_OFF);

  if (Push_button_on_state == HIGH) {
    //Khởi động động hệ thống
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, HIGH);
    lcd.clear();
    lcd.setCursor(0, 1);
    lcd.print("Start...");
    //Đặt tốc độ cho cáp treo, đi từ trạm đầu tới cuối hết 5s
    analogWrite(ENA, 255);
    delay(5000);

    //Sau đó giảm tốc độ, đón trả khách trong 5s
    slowDown();
    lcd.clear();
    lcd.setCursor(0, 1);
    lcd.print("Slow down");
    delay(5000);

    //Đón trả khách xong lại tăng tốc và tiếp tục đi
    speedUp();
    lcd.clear();
    lcd.setCursor(0, 1);
    lcd.print("Speed up...");
    delay(5000);
    lcd.clear();
    lcd.setCursor(0, 1);
    lcd.print("Running");
    //lcd.print("ON...");
    //delay(2000);
    //lcd.clear();
    //lcd.print("ON");
  }
}

```

Trước tiên, đọc trạng thái của chân nút nhấn bằng cách sử dụng hàm `digitalRead()` và lưu chúng vào các biến `Push_button_on_state` và `Push_button_off_state`.

Sau đó, khi biến `Push_button_on_state == HIGH` (được bật), ta sẽ bật động cơ, bằng cách sử dụng hàm `digitalWrite()` và chuyển chân đầu vào làm tham số đầu tiên và trạng thái làm tham số thứ hai. Ở đây động cơ quay theo hướng ngược kim đồng hồ (backward) vì IN1 ở mức thấp và IN2 ở mức cao.

Chúng em cũng xóa nội dung hiển thị ở LCD và thiết lập hiển thị trên màn hình theo trạng thái của hệ thống.

Qua quá trình chạy thử nhiều lần, chúng em ước lượng được thời gian trung bình của mô hình mô phỏng hệ thống. Cụ thể, khi đi từ trạm này qua trạm kia sẽ mất thời gian 5 giây, sau đó chúng em sẽ cho cáp treo giảm tốc và đi thật chậm lại trong 5 giây nữa để đón/trả khách tại trạm rồi sẽ tiếp tục tăng tốc và hệ thống sẽ tiếp tục vận hành như vậy.

Trước tiên ta sẽ đặt tốc độ cho cáp treo bằng hàm `analogWrite()`, sau đó dùng các hàm `speedUp()`, `slowDown()` và `delay()` để thiết lập giảm tốc, tăng tốc sau thời gian 5 giây.

- Cuối cùng chúng ta cũng có thể cho dừng hệ thống bằng cách thiết lập biến `Push_button_off_state == HIGH`:

```

if (Push_button_off_state == HIGH) {
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, LOW);
}

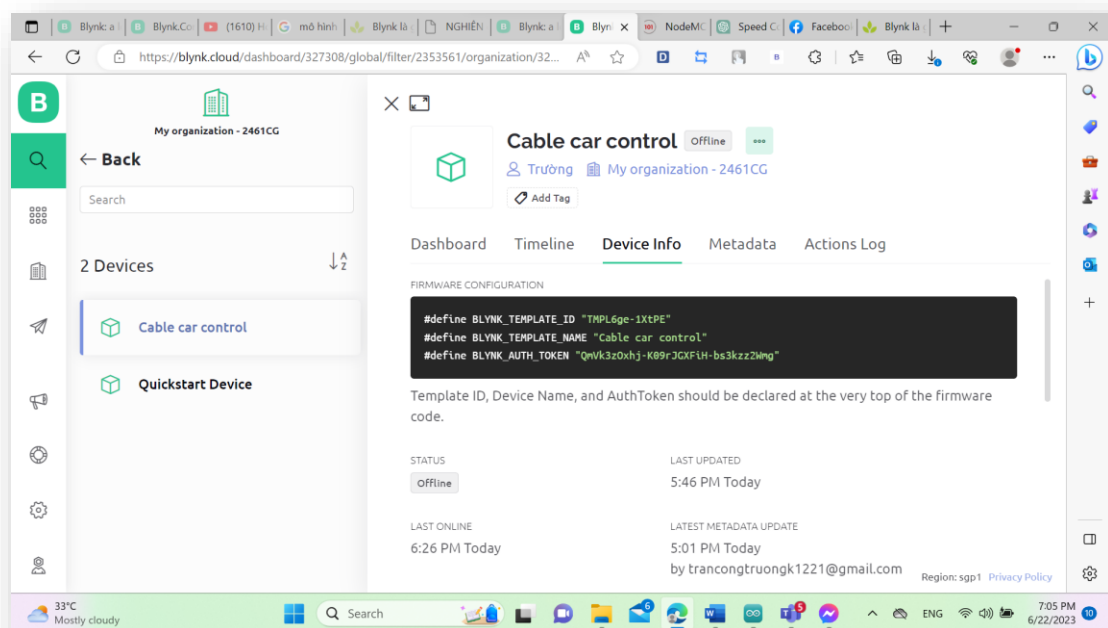
```

4.2 Lập trình điều khiển sử dụng Blynk

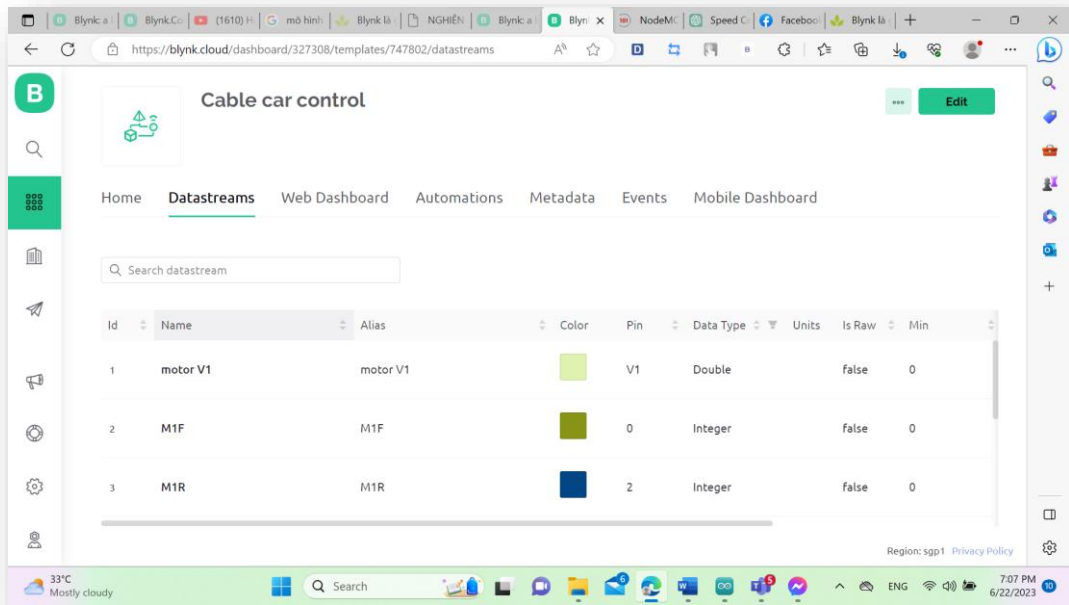
Blynk là một nền tảng IoT cung cấp các công cụ để kết nối, quản lý và điều khiển các thiết bị IoT từ xa thông qua mạng Internet. Điểm nổi bật của Blynk là sự dễ dàng và nhanh chóng trong việc tạo và quản lý các ứng dụng IoT, phù hợp cho cả những người mới bắt đầu và những nhà phát triển chuyên nghiệp.

Blynk cung cấp ứng dụng di động và API cho phép người dùng kết nối và điều khiển các thiết bị IoT bằng cách sử dụng các cảm biến và các tín hiệu đầu vào. Giao diện người dùng được tùy chỉnh linh hoạt để điều khiển thiết bị IoT theo cách tùy chỉnh và tạo ra các hành động và tương tác phức tạp thông qua mã code.

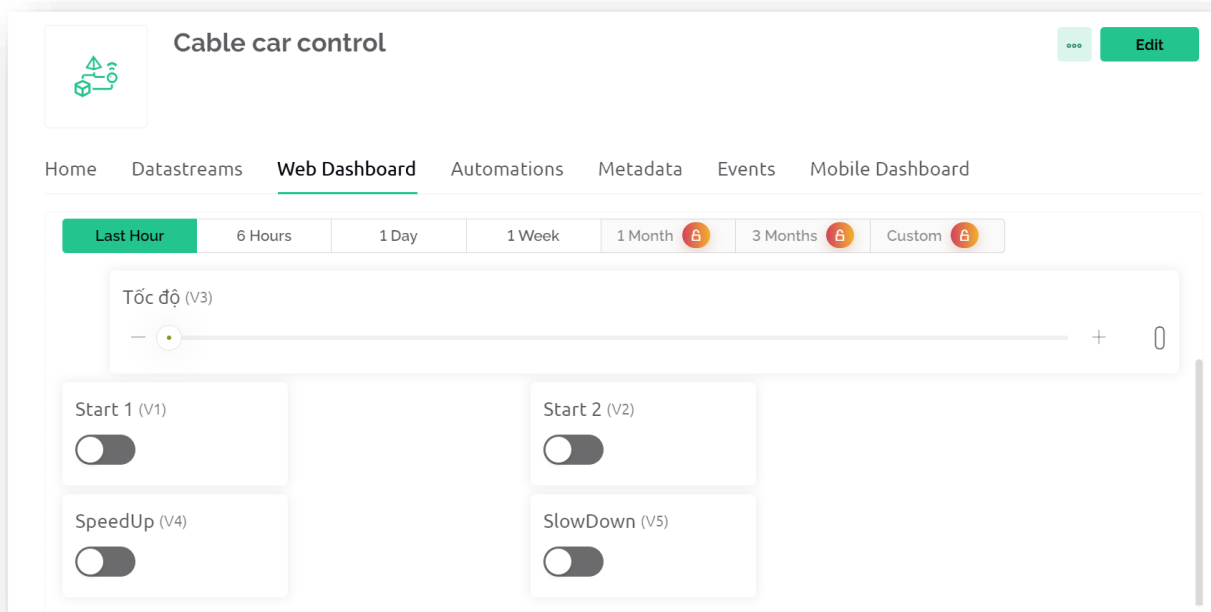
Cấu hình phần mềm: Template ID, Device Name, and AuthToken phải được khai báo ở đầu chương trình code.



Thiết lập Datastreams: thiết lập dữ liệu, kiểu dữ liệu, ...

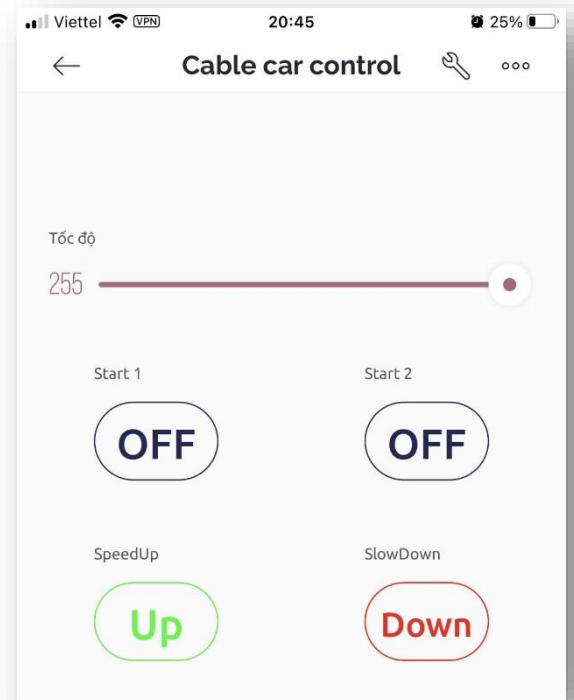


- Thiết lập Web Dashboard: Thiết kế nút trên web để điều khiển hệ thống.



Thiết lập điều khiển trên app. Các nút có các chức năng dưới đây:

- Chỉnh thanh slider để điều chỉnh tốc độ.
- Start 1: + ON: mô hình chạy theo chiều đồng hồ
+ OFF: mô hình dừng
- Start 2: mô hình chạy ngược chiều đồng hồ
- SpeedUp: Tăng tốc độ
- SlowDown: Giảm tốc độ



Cấu hình phần mềm, kết nối wifi.

```
#define BLYNK_TEMPLATE_ID "TMPL6ge-1XtPE"
#define BLYNK_TEMPLATE_NAME "Cable car control"
#define BLYNK_AUTH_TOKEN "QmVk3zOxhj-K09rJGXFih-bs3kzz2Wmg"

#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <LiquidCrystal_I2C.h>
#include <Wire.h>

char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "Nha Tro 25 C"; // Nhập tên mạng WiFi của bạn
char pass[] = "matmangroi"; // Nhập mật khẩu mạng WiFi của bạn
```

Định nghĩa hàm BLYNK_WRITE(V1) để xử lý sự kiện khi giá trị của nút kết nối với pin V1 trong ứng dụng Blynk thay đổi. Nếu giá trị nút là HIGH (được bật), động cơ sẽ được bật theo hướng chuyển tiếp (chân M1F HIGH, chân M1R LOW). Nếu giá trị nút là LOW (được tắt), động cơ sẽ dừng lại (cả hai chân M1F và M1R đều LOW).


```
BLYNK_WRITE(V1)
{
  int buttonState = param.asInt();
  if (buttonState == HIGH)
  {
    lcd.clear();
    lcd.setCursor(0, 1);
    lcd.print("Start...");
    digitalWrite(M1F, HIGH);
    digitalWrite(M1R, LOW);
  }
  else
  {
    digitalWrite(M1F, LOW);
    digitalWrite(M1R, LOW);
    lcd.clear();
    lcd.setCursor(0, 1);
    lcd.print("Stop...");
  }
}
```

Định nghĩa hàm BLYNK_WRITE(V1) để xử lý sự kiện khi giá trị của nút kết nối với pin V1 trong ứng dụng Blynk thay đổi.

```

BLYNK_WRITE(V2)
{
  int buttonState = param.asInt();
  if (buttonState == HIGH)
  {
    lcd.clear();
    lcd.setCursor(0, 1);
    lcd.print("Start...");
    digitalWrite(M1F, LOW);
    digitalWrite(M1R, HIGH);
  }
  else
  {
    digitalWrite(M1F, LOW);
    digitalWrite(M1R, LOW);
    lcd.clear();
    lcd.setCursor(0, 1);
    lcd.print("Stop...");
  }
}

```

Định nghĩa hàm BLYNK_WRITE(V3) để xử lý sự kiện khi giá trị của nút kết nối với pin V3 trong ứng dụng Blynk thay đổi. Thanh thay đổi giá trị tốc độ.

```

BLYNK_WRITE(V3)
{
  int sliderValue = param.asInt();
  analogWrite(M1PWM, sliderValue);
  pinValue1 = sliderValue;
}

```

Định nghĩa hàm BLYNK_WRITE(V4) để xử lý sự kiện khi giá trị của nút kết nối với pin V4 trong ứng dụng Blynk thay đổi. Có chức năng tăng tốc cho cabin.

```
BLYNK_WRITE(V4)
{
  int buttonState = param.asInt();
  if (buttonState == HIGH)
  {
    if (pinValue1 < 255 - increment)
    {
      pinValue1 += increment;
    }
    else
    {
      pinValue1 = 255;
    }
    analogWrite(M1PWM, pinValue1);
    Blynk.virtualWrite(V3, pinValue1);
  }
}
```

Định nghĩa hàm BLYNK_WRITE(V5) để xử lý sự kiện khi giá trị của nút kết nối với pin V5 trong ứng dụng Blynk thay đổi. Có chức năng giảm tốc cho cabin.

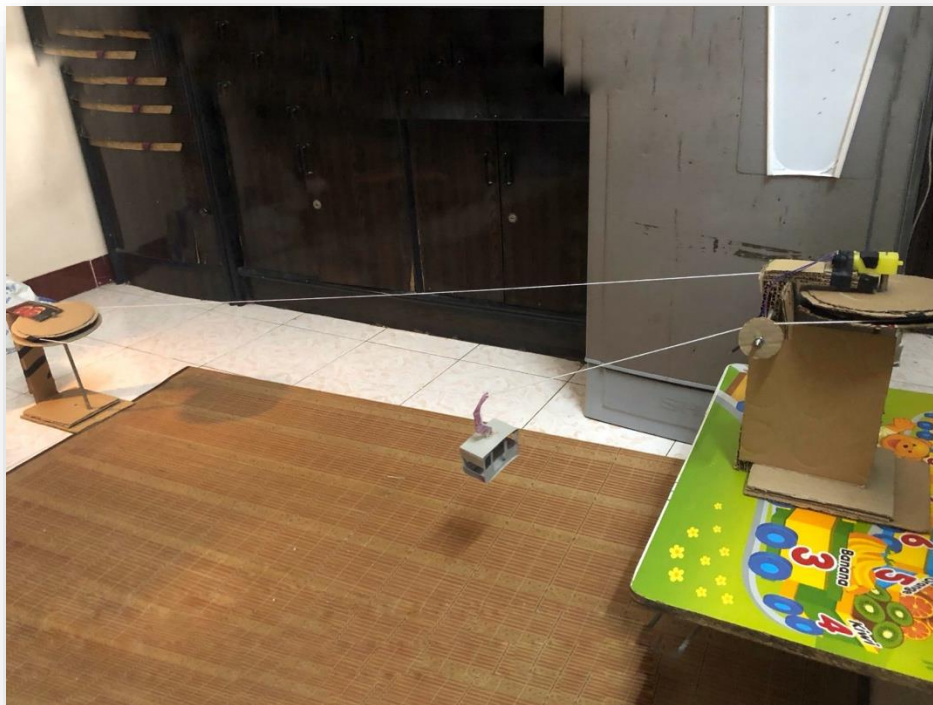
```

BLYNK_WRITE(V5)
{
  int buttonState = param.asInt();
  if (buttonState == HIGH)
  {
    if (pinValue1 > increment)
    {
      pinValue1 -= increment;
    }
    else
    {
      pinValue1 = 0;
    }
    analogWrite(M1PWM, pinValue1);
    Blynk.virtualWrite(V3, pinValue1);
  }
}

```

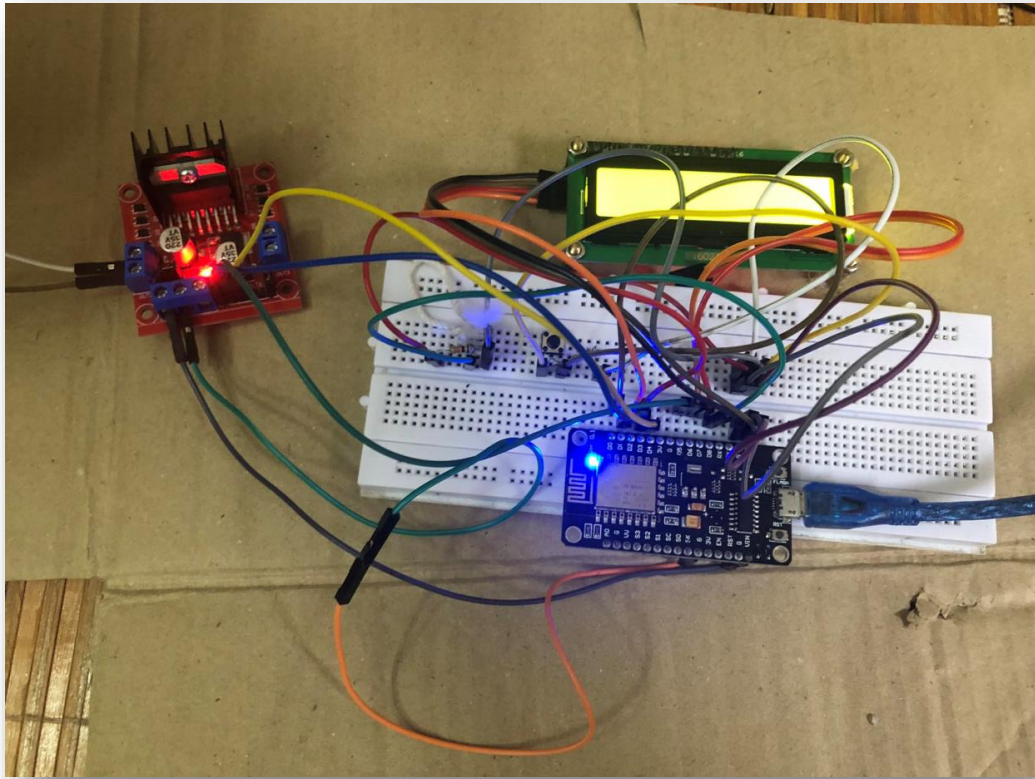
CHƯƠNG 5. TRIỂN KHAI, THỬ NGHIỆM TOÀN HỆ THỐNG

5.1 Mô hình cáp treo

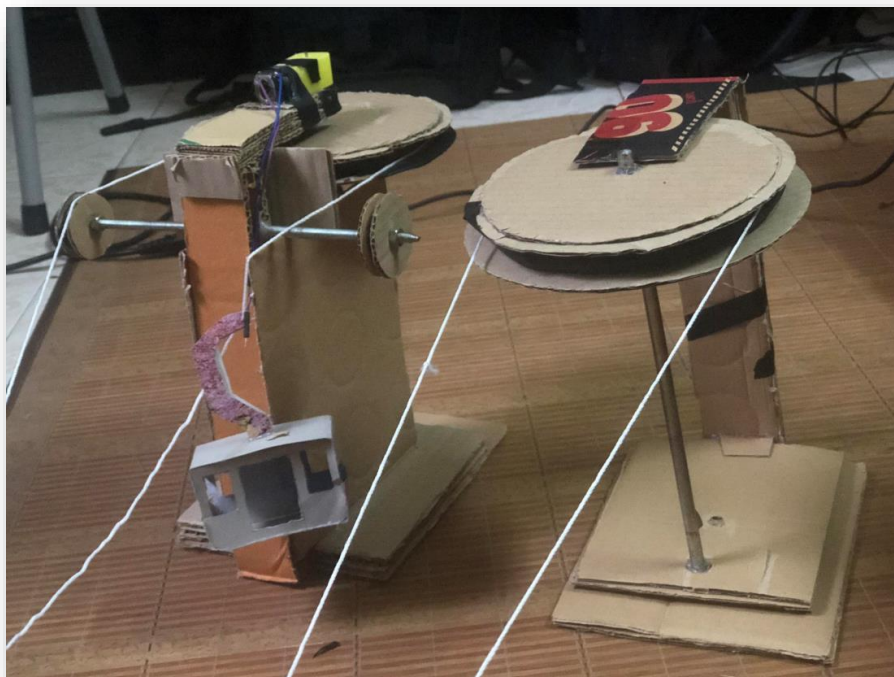


Mô hình bao gồm:

- Mạch mô phỏng



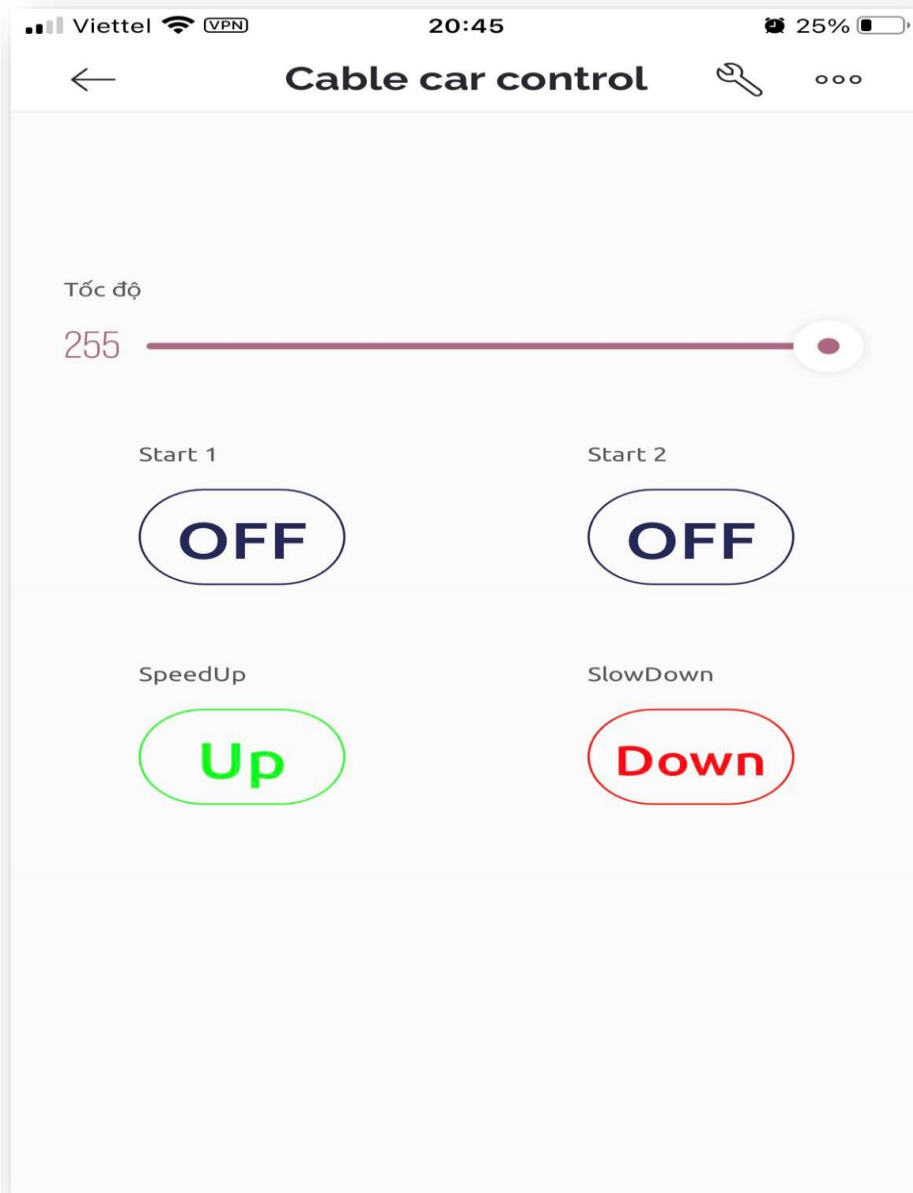
- Gồm 2 trạm đón khách:



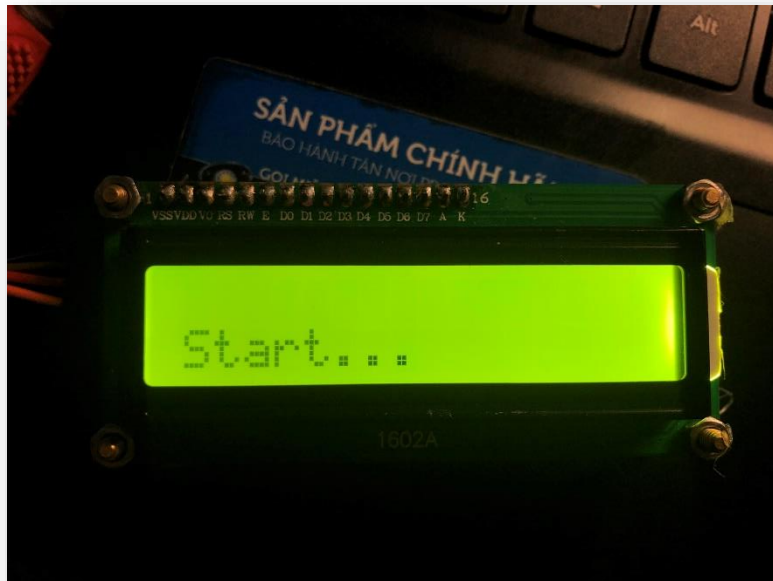
Khi nạp code vào ESP8266 thành công, màn hình LCD sẽ hiện lên như sau:



5.2 App Blynk điều khiển hệ thống



Sau khi mở app Blynk, thao tác ấn nút Start 1 thì cabin bắt đầu chuyển động đồng thời LCD hiện lên chữ Start... như hình.



Trong quá trình vận chuyển hành khách người điều khiển có thể điều chỉnh tăng giảm tốc độ bằng nút SpeedUp và SlowDown. Mỗi lần bấm sẽ tăng, giảm 15 đơn vị. Khi muốn hệ thống nghỉ người điều khiển bấm nút Start 1 để dừng ngay sau đó màn hình LCD hiện lên như hình sau:



CHƯƠNG 6. ĐÁNH GIÁ KẾT QUẢ VÀ ĐỊNH HƯỚNG PHÁT TRIỂN

6.1 Đánh giá kết quả

Sau quá trình tìm hiểu và thực hiện đề tài, nhóm chúng em có các đánh giá như sau:

- Mạch chạy ổn định, các phím điều khiển hoạt động đúng yêu cầu như nguyên lý đã đề ra.
- Do không làm mạch in nên các dây cắm trên board trắng còn phức tạp, gây khó khăn trong quá trình kiểm tra và hoạt động của mạch.
- Mô hình cáp treo làm bằng bìa giấy nên chưa chắc chắn, nếu có điều kiện cần làm bằng chất liệu khác để tăng chất lượng.
- Các thành viên trong nhóm đều có ý thức tìm hiểu, tham gia vào đề tài.

Tuy có nhiều hạn chế về kiến thức và thời gian nhưng về cơ bản nhóm đã hoàn thành các yêu cầu cơ bản của đề tài trong thời gian quy định.

6.2 Định hướng phát triển

Ngoài những chức năng đã phát triển được ở trên, trong tương lai nhóm chúng em sẽ tiếp tục tìm hiểu và tích hợp thêm một số chức năng như:

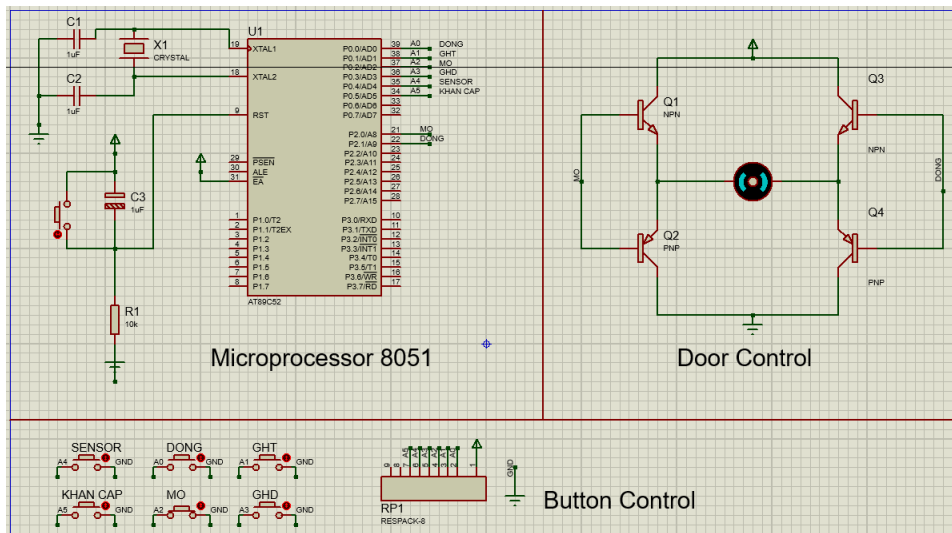
- Đặt cảm biến khoảng cách tại các trạm, từ đó việc tăng giảm tốc độ sẽ phụ thuộc vào cảm biến khoảng cách khi phát hiện cabin đã gần về trạm hay chưa.
- Phát triển mô hình cửa tự động đóng mở và tích hợp thêm các cảm biến như RFID, cảm biến hồng ngoại, cảm biến chuyển động.
- Chuyển các bo mạch từ board trắng thành mạch in để tín hiệu, dòng điện chạy ổn định hơn, thuận tiện cho việc lắp đặt và sử dụng.

PHẦN II. BÁO CÁO CÁ NHÂN

1. PHÍ MẠNH TOÀN

Trong quá trình thực hiện đề tài trên, em đã học hỏi và đóng góp cho nhóm được các phần sau:

Thực hiện vẽ và mô phỏng các mạch điều khiển trong phần mềm Proteus, sử dụng các phần mềm lập trình như Arduino IDE, Keil C u4 để lập trình và mô phỏng.



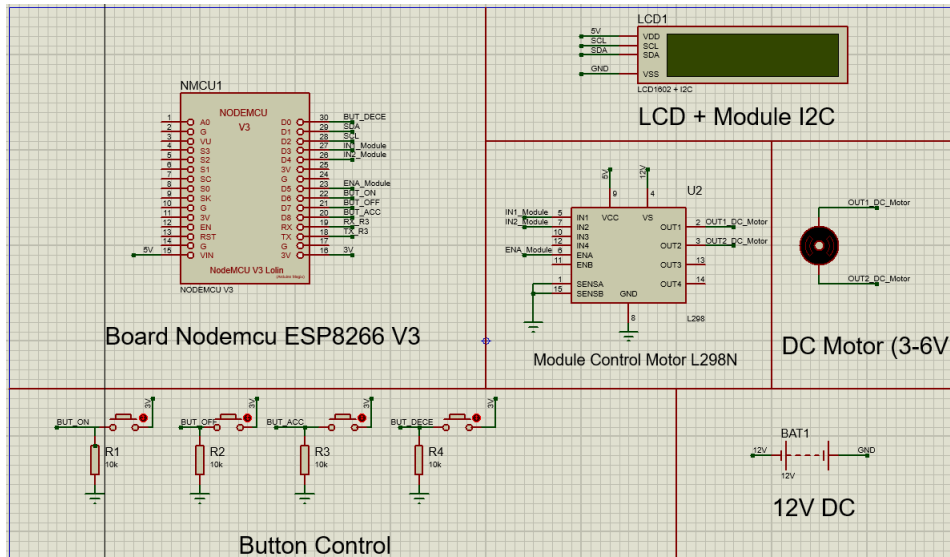
Qua mạch trên, em tìm hiểu được về cấu tạo của vi điều khiển 8051 (AT89C52), cách tạo khối dao động, mạch reset, các cổng GPIO, cách cài đặt các chân GPIO giao tiếp với nút nhấn

Hiểu được nguyên lý của khối điều khiển động cơ đóng mở cửa bằng cách phân cực các transistor BJT

Biết sử dụng phần mềm Keil C để viết code, biên dịch file hex, nạp vào trong phần mềm Proteus để mô phỏng hoạt động của mạch.

```
1 #include <REG52.h>
2
3 #define DONG = P0^0
4 #define GHT = P0^1
5 #define MO = P0^2
6 #define KHAN_CAP = P0^3
7 #define SENSOR = P0^4
8 #define KHAN_CAP = P0^5
9 #define THUAN = P2^0
10 #define NGUOC = P2^1
11
12 int main()
13 {
14     while(1)
15     {
16         if (DONG==1 && GHT==0 && MO==0 && SENSOR==1 && KHAN_CAP==0)
17         {
18             THUAN = 1;
19             NGUOC = 0;
20         }
21         if (DONG && GHT && MO && SENSOR && KHAN_CAP)
22         {
23             THUAN = 0;
24             NGUOC = 1;
25         }
26         if (DONG && GHT && MO && SENSOR && KHAN_CAP)
27         {
28             THUAN = 1;
29             NGUOC = 1;
30         }
31         if (DONG && MO && KHAN_CAP)
32         {
33             THUAN = 1;
34             NGUOC = 1;
35         }
36         if (DONG && MO && KHAN_CAP)
37         {
38             THUAN = 1;
39             NGUOC = 1;
40         }
41     }
42 }
```

Giao diện code bằng phần mềm Keil C



Từ việc xây dựng mạch nguyên lý điều khiển động cơ của cáp treo trên em đã tìm hiểu được các kiến thức như:

Cấu tạo của board nodemcu esp8266, các chân GPIO, 2 chân giao tiếp I2C với LCD.

Cấu tạo và nguyên lý hoạt động của module điều khiển động cơ L298N, cách điều khiển tốc độ, chiều của động cơ.

Cách sử dụng phần mềm Aduino IDE để code và nạp firmware cho vi điều khiển, cách sử dụng thư viện LiquidCrystal I2C để giao tiếp giữa Board điều khiển và LCD một cách thuận tiện hơn.

```

final_code_V1.ino
1 #include <LiquidCrystal_I2C.h>
2 #include <Wire.h>
3 int ENA = D5;
4 int IN1 = D3;
5 int IN2 = D4;
6 const int BUTTON_ON = D6;
7 const int BUTTON_OFF = D7;
8 LiquidCrystal_I2C lcd(0x27,16,2);
9
10
11 void setup() {
12   lcd.init();
13   lcd.backlight();
14   lcd.print("Cap treo nhom 2");
15   pinMode(ENA, OUTPUT);
16   pinMode(IN1, OUTPUT);
17   pinMode(IN2, OUTPUT);
18   pinMode(BUTTON_ON, INPUT);
19   pinMode(BUTTON_OFF, INPUT);
20   // pinMode(BUTTON_ON, INPUT);
21   // pinMode(BUTTON_OFF, INPUT);
22   digitalWrite(IN1, LOW);
23   digitalWrite(IN2, LOW);
24   //khởi động chức năng serial
25 }
26
27 void loop() {
28   //analogWrite(ENA, 100);
29   int Push_button_on_state = digitalRead(BUTTON_ON);
30   int Push_button_off_state = digitalRead(BUTTON_OFF);
31
32   if (Push_button_on_state == HIGH) {
33     //khởi động động hệ thống
34
35     digitalWrite(IN1, LOW);
36

```

Giao diện phần mềm Aduino IDE

Em cũng đã tìm hiểu được nguyên lý hoạt động của các loại cảm biến đo khoảng cách và cảm biến đo tốc độ Encoder V2:

Để đo được tốc độ của trục quay, nhóm chúng em tích hợp cảm biến đo tốc encoder V2, gắn đĩa quay vào trục quay để đo tốc độ.



Cảm biến đo tốc độ Encoder V2 và đĩa quay encoder 20 xung

Cảm biến bao gồm 1 mắt phát và 1 mắt thu hồng ngoại đặt cách nhau qua 1 khe hở. Khi ánh sáng từ mắt phát đi được tới mắt thu (xuyên qua lỗ đĩa của encoder) thì sẽ có tín hiệu mức cao phát ra khỏi chân OUT, khi bị che lại thì chân OUT phát ra tín hiệu mức thấp.

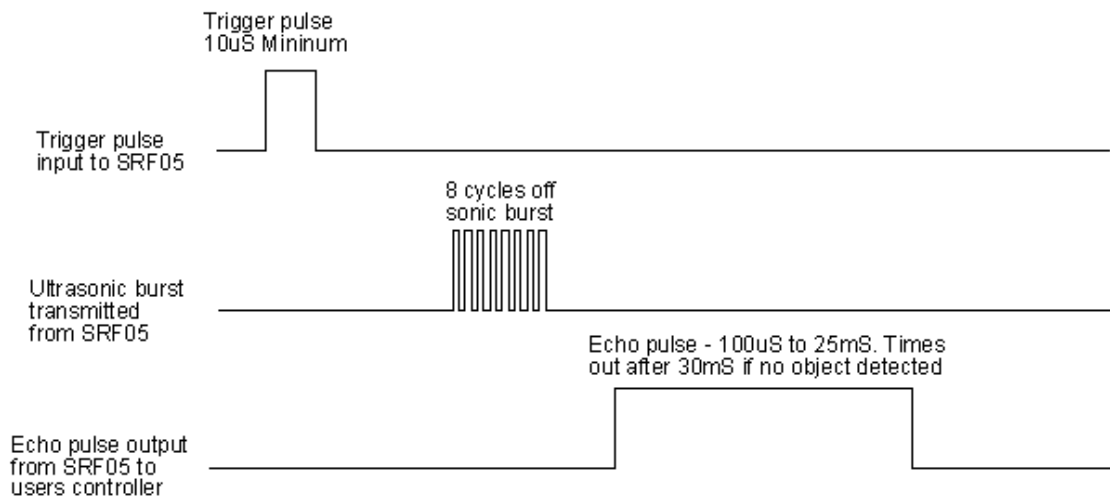
Do đề tài này chỉ dừng lại ở mức làm mô hình cỡ nhỏ nên cảm biến đo khoảng cách nhóm chúng em chọn là SRF05 (phạm vi đo khoảng cách <4,5m). Thông số kỹ thuật:



Cảm biến siêu âm SRF05

Cảm biến SRF05 là một loại cảm biến đo khoảng cách dựa trên nguyên lý thu phát siêu âm. Cảm biến gồm một bộ phát và một bộ thu sóng siêu âm. Sóng siêu âm từ đầu phát truyền đi trong không khí, gặp vật cản sẽ phản xạ ngược lại và được đầu thu ghi lại. Vận tốc truyền âm thanh trong không khí là một giá trị được xác định trước, ít

thay đổi (cỡ 3.10^8 m/s). Do đó, nếu xác định được khoảng thời gian từ lúc phát sóng đến lúc nhận được sóng phản xạ tới sẽ tính được khoảng cách từ vật cần đến cảm biến.



SRF05 Timing Diagram

Cảm biến SRF05 có 2 chân TRIGGER và ECHO riêng biệt, khi chân MODE để trống, SRF05 sẽ sử dụng cả 2 chân chức năng TRIGGER và ECHO cho việc điều khiển hoạt động của cảm biến. Từ biểu đồ thời gian của chế độ hoạt động ta thấy để đo khoảng cách, đầu tiên ta phát 1 xung cỡ 5 us từ chân TRIG, sau đó cảm biến sẽ tạo ra xung HIGH ở chân ECHO cho đến khi nhận được xung phản xạ ở chân này. Chiều rộng của xung sẽ bằng với thời gian sóng siêu âm được phát từ cảm biến quay trở lại. Tốc độ của âm thanh trong không khí tương đương $29,412 \text{ us/cm} = 3.108 \text{ m/s}$. Khi có khoảng cách từ cảm biến tới vật cần được tính bằng công thức:

2. HOÀNG VŨ AN

2.1 Board mạch NodeMCU Esp8266 V3

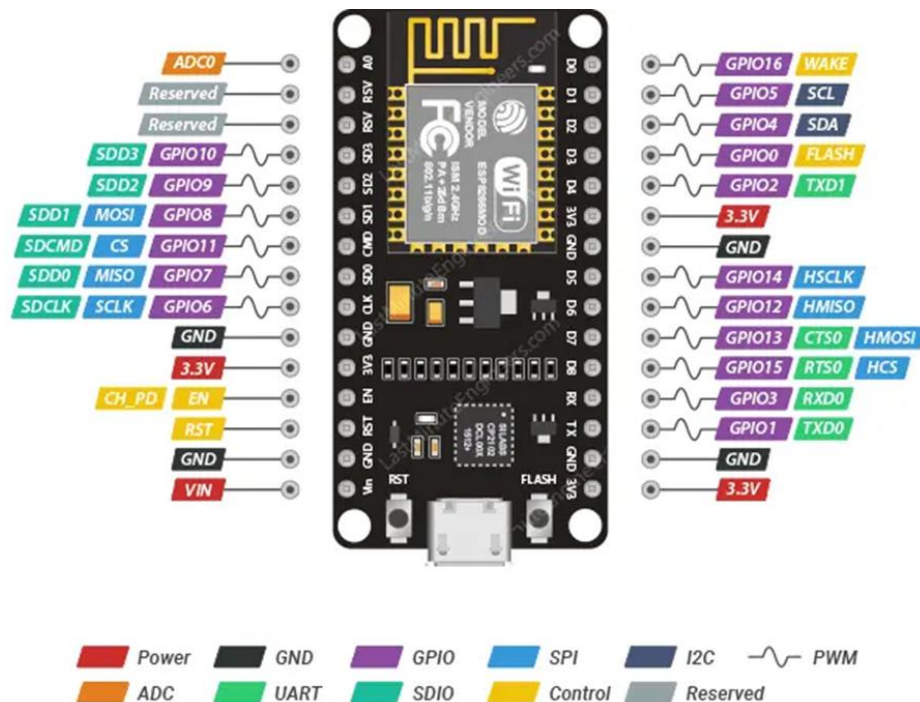
ESP8266 là một module Wi-Fi với khả năng kết nối Internet và được tích hợp sẵn trên một số board nhúng như NodeMCU, Wemos, và ESP-01. ESP8266 có thể hoạt động như một điểm truy cập (access point), một client kết nối đến một điểm truy cập khác, hoặc cả hai đều được. Nó được sử dụng rộng rãi trong các ứng dụng IoT (Internet of Things) như cảm biến thông minh, hệ thống kiểm soát thiết bị, hoặc các ứng dụng điều khiển từ xa. Module này có giá thành rẻ và rất dễ sử dụng, cùng với đó là khả năng tương thích với nhiều loại vi điều khiển khác nhau.

ESP8266 NodeMCU là một nền tảng IoT mã nguồn mở, được xây dựng trên ESP8266. NodeMCU cung cấp một bộ SDK để lập trình cho ESP8266 bằng ngôn ngữ Lua hoặc C++. Với các tính năng như Wi-Fi, GPIO, ADC, I2C, SPI, PWM và một số tính năng khác, NodeMCU ESP 8266 được sử dụng rộng rãi trong các ứng dụng IoT như kiểm soát thiết bị, thu thập dữ liệu và giao tiếp với các thiết bị khác.

Thông số kỹ thuật sản phẩm :

- Microcontroller: ESP8266EX
- Điện áp hoạt động: 3.3V DC
- Số chân I/O: 17 chân GPIO
- Kết nối mạng: WiFi 802.11 b/g/n
- Giao diện mạng: TCP/IP
- Đồng hồ thời gian thực (RTC): không tích hợp
- Bộ nhớ trong: 4MB
- RAM: 80KB
- Cổng nạp: Micro-USB
- Hỗ trợ các giao thức: MQTT, CoAP, HTTP/HTTPS
- Kích thước: 49 x 24.5 x 13mm

Sơ đồ chân ESP8266 NodeMCU:



ESP8266 NodeMCU Pinout

Last Minute ENGINEERS.com

Dưới đây là một số thiết bị ngoại vi và các chân I/O quan trọng trên ESP8266 NodeMCU:

17 chân GPIO	Được sử dụng để đọc dữ liệu từ các cảm biến, điều khiển các thiết bị đầu ra, hoặc giao tiếp với các thiết bị khác như LED, động cơ, nút nhấn, v.v.
--------------	--

1 kênh ADC	1 kênh ADC có độ chính xác 10 bit theo công nghệ SAR ADC.
2 giao tiếp UART	2 giao tiếp UART hỗ trợ điều khiển dòng dữ liệu.
4 đầu ra PWM	4 chân PWM để điều khiển tốc độ động cơ hoặc độ sáng của đèn LED.
2 giao tiếp SPI và 1 giao tiếp I2C	2 giao tiếp SPI và một giao tiếp I2C để kết nối các cảm biến và thiết bị ngoại vi khác.
Giao tiếp I2S	Một giao tiếp I2S để thêm âm thanh vào dự án của bạn.

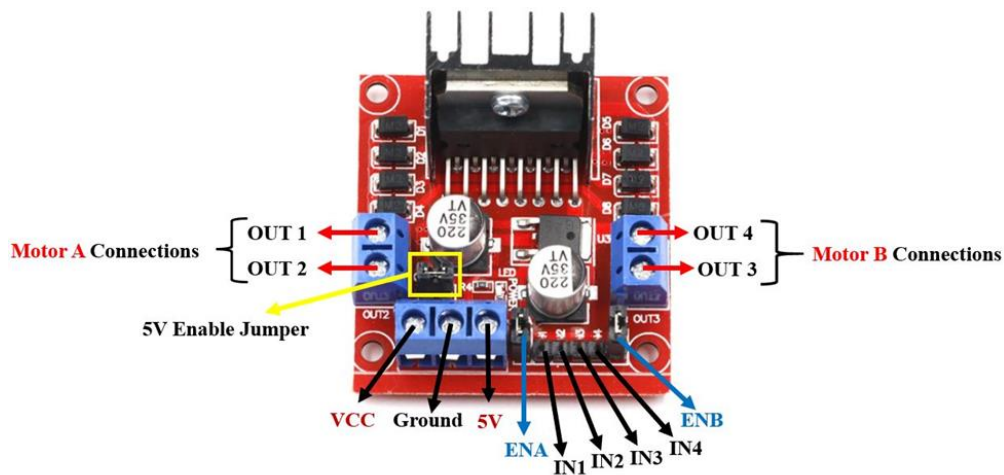
2.2 Module L298N

L298N là module điều khiển động cơ trong các xe DC và động cơ bước, rất dễ sử dụng với các bộ vi điều khiển và cũng tương đối rẻ.

Module có một IC điều khiển động cơ L298N, một bộ điều chỉnh điện áp 5V 78M05, 5V Enable Jumper, đèn LED nguồn, tản nhiệt, điện trở và tụ điện, tất cả được tích hợp trong một mạch. Module L298N có thể điều khiển tối đa 4 động cơ DC hoặc 2 động cơ DC với khả năng điều khiển hướng và tốc độ.

Thông số kỹ thuật:

- Module điều khiển: L298N
- Chip điều khiển: Cặp H-bridge L298N
- Công suất tối đa: 25W
- Điện áp tối đa cấp cho động cơ: 46V
- Dòng tối đa cấp cho động cơ: 2A
- Điện áp hoạt động của IC: 5-35V
- Dòng điện hoạt động IC: 2A



Sơ đồ chân L298N

ENA điều khiển tốc độ của động cơ A và ENB điều khiển tốc độ của động cơ B. Nếu cả hai chân đều ở trạng thái logic CAO (5V), thì cả hai động cơ đều BẬT và quay ở tốc độ tối đa. Nếu cả hai chân đều ở trạng thái logic THẤP (tiếp đất), thì cả hai động cơ đều TẮT. Thông qua chức năng PWM, chúng ta cũng có thể kiểm soát tốc độ của động cơ. Bảng dưới đây minh họa các tín hiệu logic cần thiết để điều khiển Động cơ A:

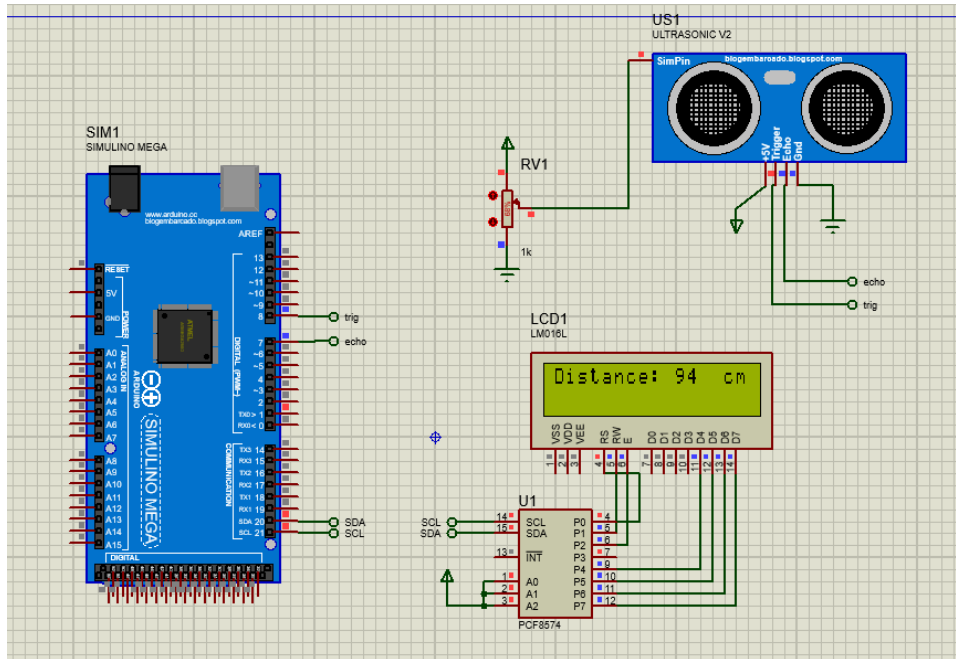
ENA Pin State	Motor Action
1 (HIGH)	ON
0 (LOW)	OFF

Các chân điều khiển hướng là bốn chân đầu vào (IN1, IN2, IN3, IN4). Với hai chân IN1 và IN2 là đầu vào của động cơ A:

IN1	IN2	Motor Action
1 (HIGH)	1	OFF
1	0 (LOW)	Backward
0	1	Forward
0	0	OFF

2.3 Tìm hiểu và mô phỏng cảm biến khoảng cách HC-SR04 hiển thị LCD I2C

Trong phần tìm hiểu này em sử dụng ARDUINO MEGA 2560 để thực hiện mô phỏng.



Code thực hiện mô phỏng:

```
void loop()
{
    unsigned long duration;
    int distance;
    digitalWrite(trig, 0);
    delayMicroseconds(2);
    digitalWrite(trig, 1);
    delayMicroseconds(5);
    digitalWrite(trig, 0);
    duration = pulseIn(echo, HIGH);
    distance = int(duration / 2 / 29.412);
    lcd.setCursor(10, 0);
    lcd.print(distance);
    if (distance < 10) {
        lcd.setCursor(11, 0);
        lcd.print(" ");
    }
    else if (distance < 100) {
        lcd.setCursor(12, 0);
        lcd.print(' ');
    }
    delay(200);
}
```

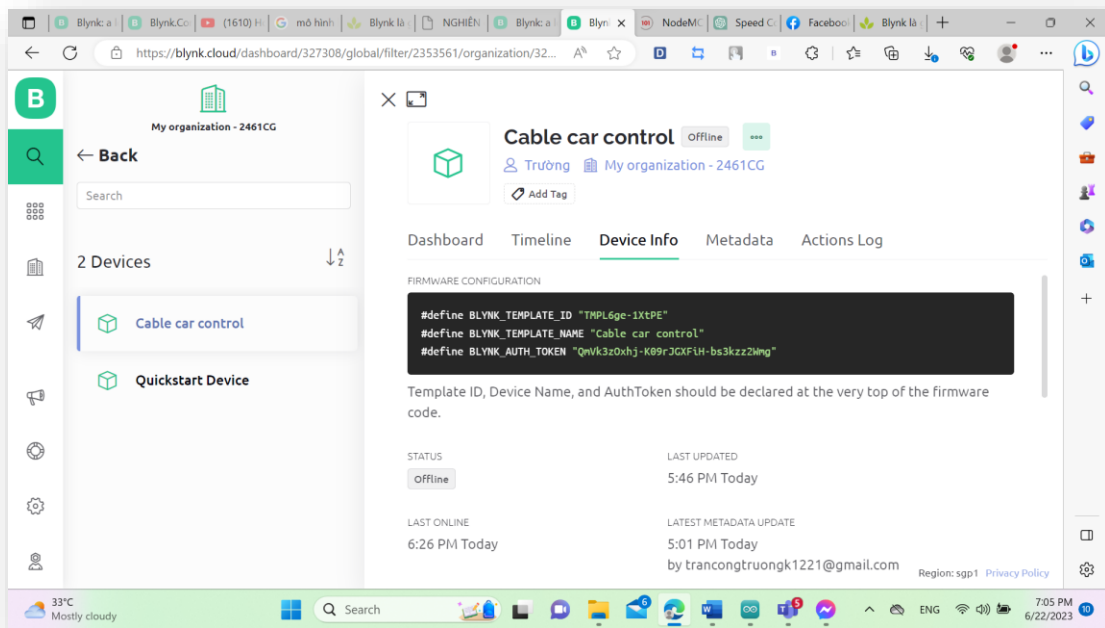
Trong bài này nhóm em mới chỉ thực hiện mô phỏng cảm biến để áp dụng trong việc giảm tốc cáp treo khi sắp tới các trạm đón trả khách. Nhóm dự tính sau này sẽ áp dụng công việc đó vào mô hình thực tế.

3 TRẦN CÔNG TRƯỜNG

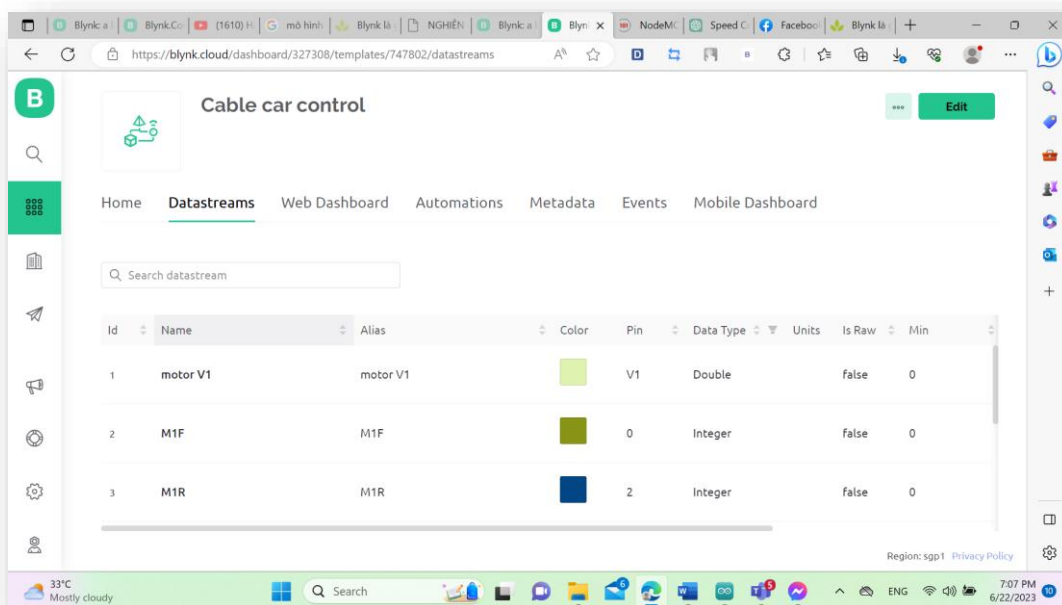
3.1 Thiết kế phần mềm chạy bằng Blynk

Tìm hiểu điều khiển hệ thống bằng Blynk trên app, web. Kết quả đạt được sau khi tìm hiểu, thiết kế, code hệ thống:

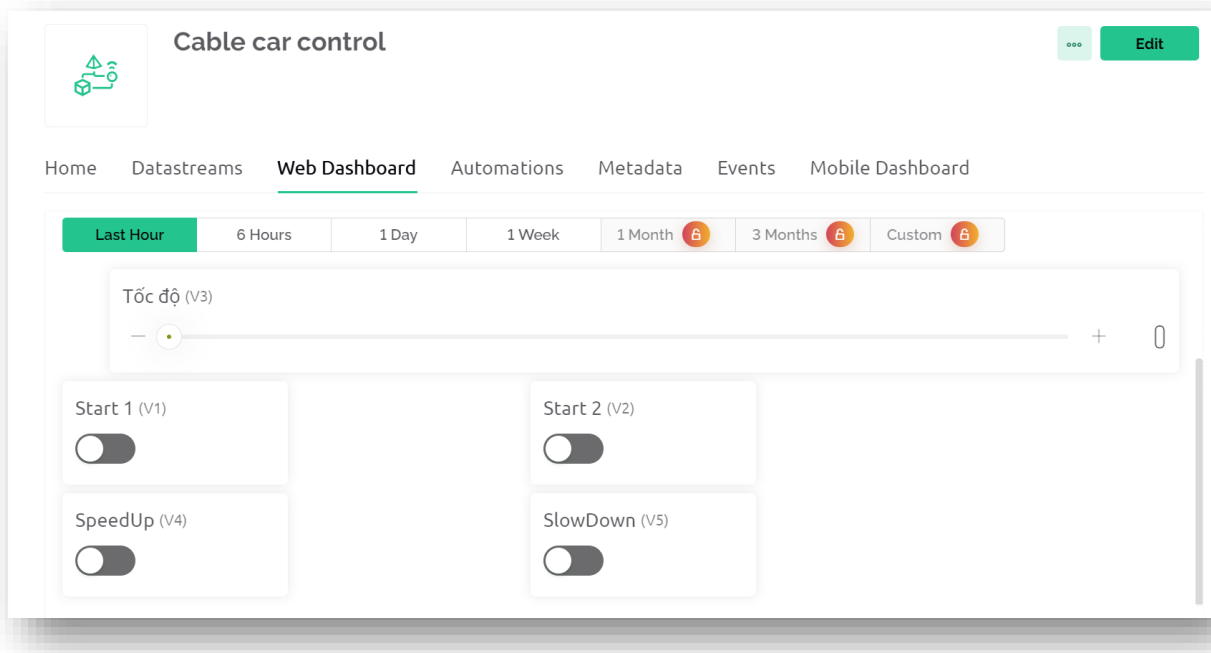
Cấu hình phần mềm: Template ID, Device Name, and AuthToken phải được khai báo ở đầu chương trình code.



Thiết lập Datastreams: thiết lập dữ liệu, kiểu dữ liệu, ...



Thiết lập Web Dashboard: Thiết kế nút trên web để điều khiển hệ thống.



Thiết lập điều khiển trên app. Các nút có các chức năng dưới đây:

- Chỉnh thanh slider để điều chỉnh tốc độ.
- Start 1: + ON: mô hình chạy theo chiều đồng hồ
+ OFF: mô hình dừng
- Start 2: mô hình chạy ngược chiều đồng hồ
- SpeedUp: Tăng tốc độ
- SlowDown: Giảm tốc độ

Cấu hình phần mềm, kết nối wifi.

```
#define BLYNK_TEMPLATE_ID "TMPL6ge-1XtPE"
#define BLYNK_TEMPLATE_NAME "Cable car control"
#define BLYNK_AUTH_TOKEN "QmVk3zOxhj-K09rJGXFih-bs3kzz2Wmg"

#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <LiquidCrystal_I2C.h>
#include <Wire.h>

char auth[] = BLYNK_AUTH_TOKEN;
char ssid[] = "Nha Tro 25 C"; // Nhập tên mạng WiFi của bạn
char pass[] = "matmangroi"; // Nhập mật khẩu mạng WiFi của bạn
```

Định nghĩa hàm BLYNK_WRITE(V1) để xử lý sự kiện khi giá trị của nút kết nối với pin V1 trong ứng dụng Blynk thay đổi. Nếu giá trị nút là HIGH (được

bật), động cơ sẽ được bật theo hướng chuyển tiếp (chân M1F HIGH, chân M1R LOW). Nếu giá trị nút là LOW (được tắt), động cơ sẽ dừng lại (cả hai chân M1F và M1R đều LOW).

```
BLYNK_WRITE(V1)
{
  int buttonState = param.asInt();
  if (buttonState == HIGH)
  {
    lcd.clear();
    lcd.setCursor(0, 1);
    lcd.print("Start...");
    digitalWrite(M1F, HIGH);
    digitalWrite(M1R, LOW);
  }
  else
  {
    digitalWrite(M1F, LOW);
    digitalWrite(M1R, LOW);
    lcd.clear();
    lcd.setCursor(0, 1);
    lcd.print("Stop...");
  }
}
```

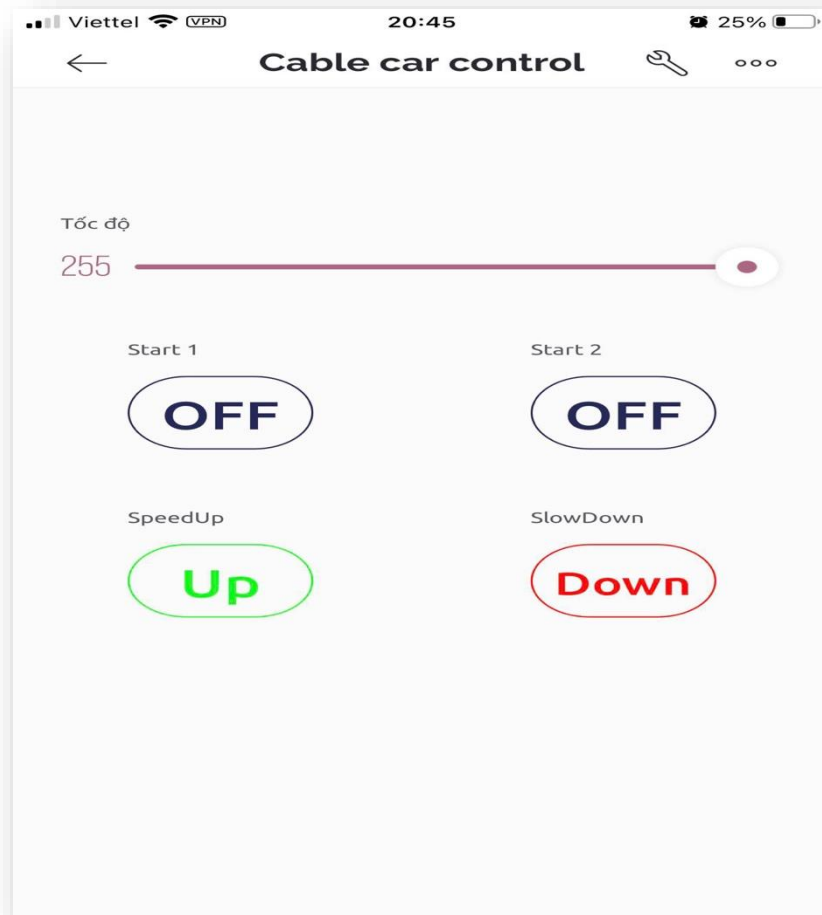
Định nghĩa hàm BLYNK_WRITE(V1) để xử lý sự kiện khi giá trị của nút kết nối với pin V1 trong ứng dụng Blynk thay đổi.

Định nghĩa hàm BLYNK_WRITE(V3) để xử lý sự kiện khi giá trị của nút kết nối với pin V3 trong ứng dụng Blynk thay đổi. Thanh thay đổi giá trị tốc độ.

Định nghĩa hàm BLYNK_WRITE(V4) để xử lý sự kiện khi giá trị của nút kết nối với pin V4 trong ứng dụng Blynk thay đổi. Có chức năng tăng tốc cho cabin.

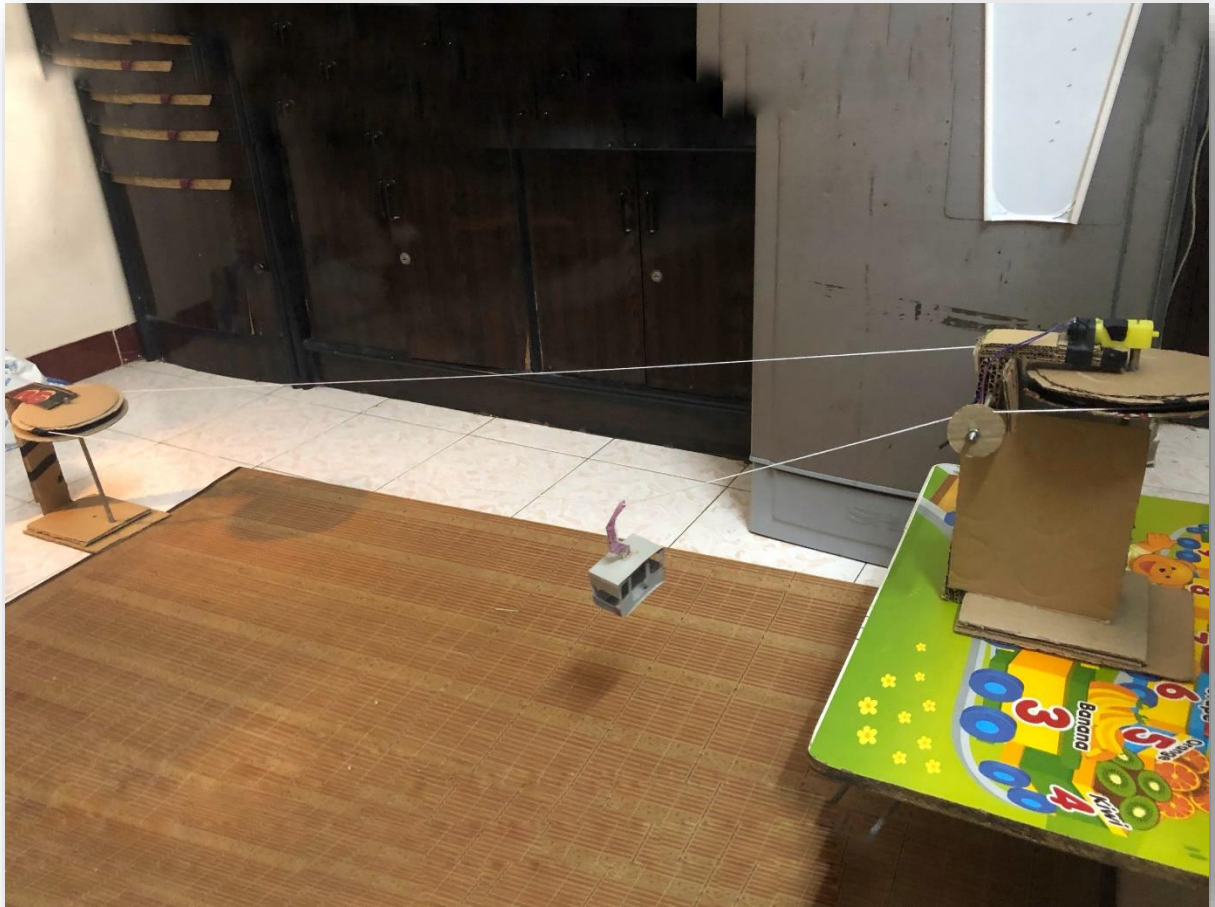
Định nghĩa hàm BLYNK_WRITE(V5) để xử lý sự kiện khi giá trị của nút kết nối với pin V5 trong ứng dụng Blynk thay đổi. Có chức năng giảm tốc cho cabin.

- Kết quả điều khiển được hệ thống một cách nhanh chóng, dễ dàng và hiệu quả hơn. Và được giao diện điều Blynk dễ nhìn.



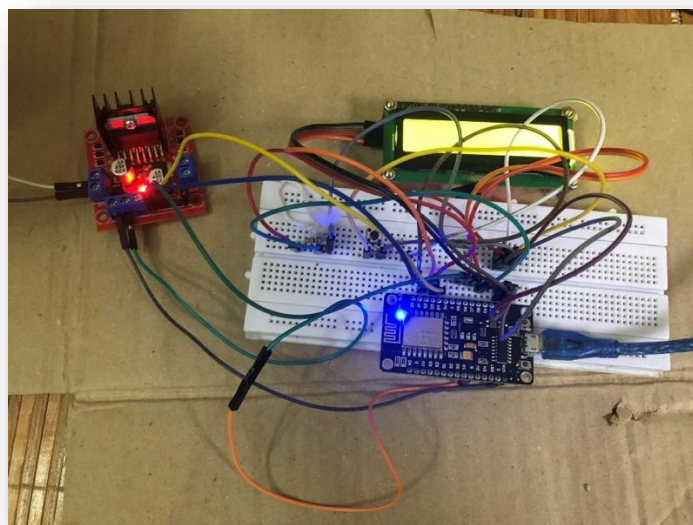
3.2 Triển khai thử nghiệm toàn hệ thống

Mô hình cáp treo được mô phỏng bằng bìa catton và một số các thiết bị, linh kiện điện tử để hoàn thiện ra được mô hình dưới đây.

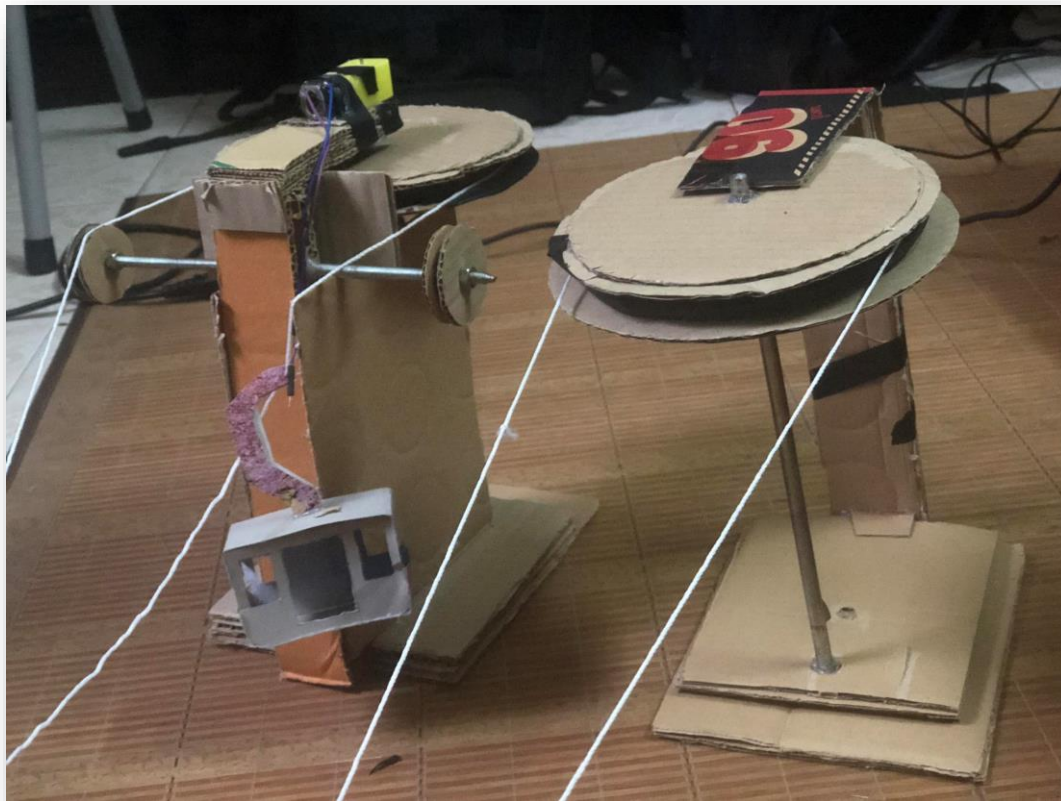


Mô hình bao gồm:

- Mạch mô phỏng 2 trạm đón trả khách



- Gồm 2 trạm đón khách, mô hình cũng được làm thủ công nên còn chưa được chắc chắn và kiên cố.

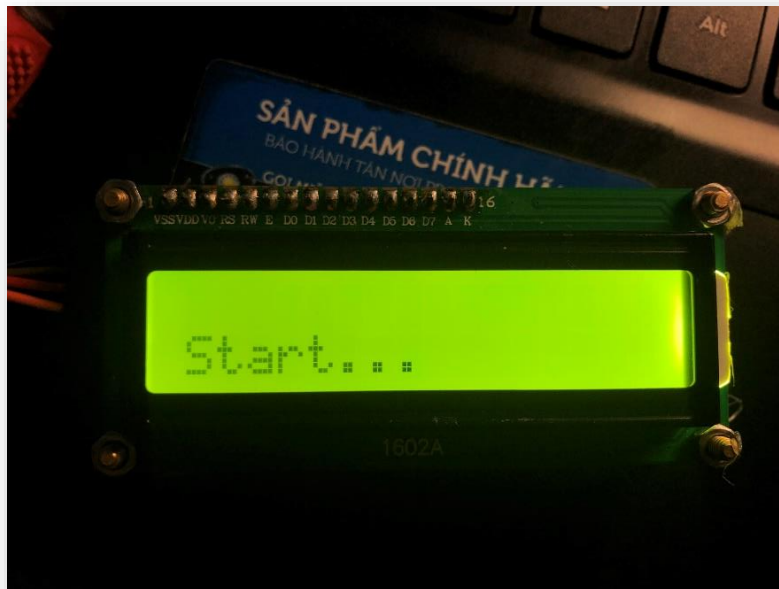


Khi nạp code vào ESP8266 thành công, màn hình LCD sẽ hiện lên như sau:

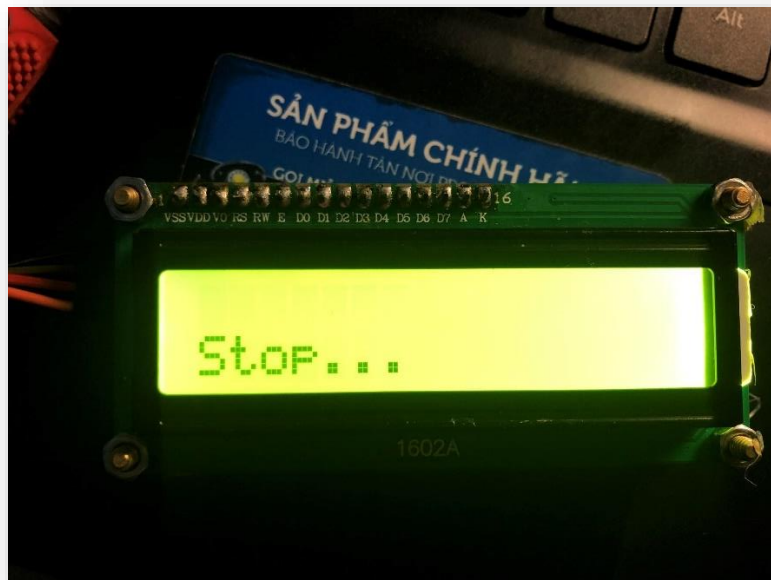


Tiếp theo tiến hành điều khiển hệ thống bằng app Blynk.

Sau khi mở app Blynk, thao tác ấn nút Start 1 thì cabin bắt đầu chuyển động đồng thời LCD hiện lên chữ Start... như hình.



Trong quá trình vận chuyển hành khách người điều khiển có thể điều chỉnh tăng giảm tốc độ bằng nút SpeedUp và SlowDown. Mỗi lần bấm sẽ tăng, giảm 15 đơn vị. Khi muốn hệ thống nghỉ người điều khiển bấm nút Start 1 để dừng ngay sau đó màn hình LCD hiện lên như hình sau:



4 PHẠM ĐỨC NAM

4.1 Xác lập chỉ tiêu kỹ thuật

- Yêu cầu chức năng:
 - Cấu tạo hệ thống cáp treo
 - Chức năng hệ thống
 - Một vài thông số kỹ thuật tiêu biểu của hệ thống cáp treo du lịch
- Yêu cầu phi chức năng

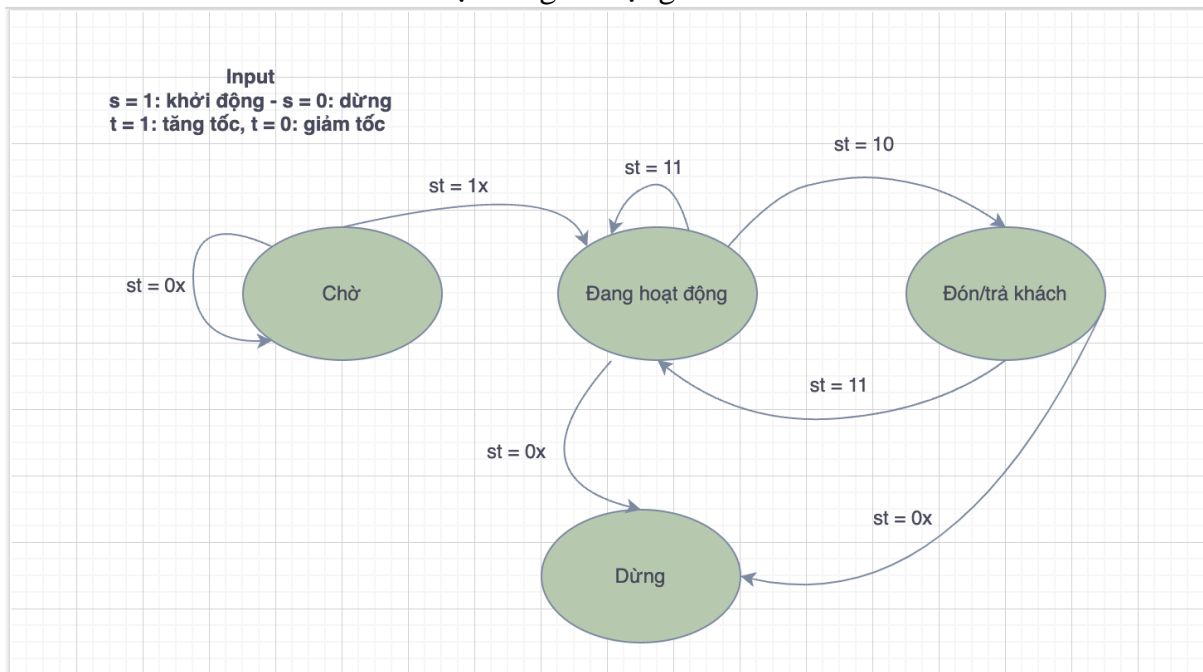
Kết quả đạt được: Em đã đọc và tìm hiểu nguyên lý hoạt động của hệ thống cáp treo ngoài đời thực, đồng thời xác định các chỉ tiêu kỹ thuật của hệ thống cáp treo khu du lịch. Từ đó bước đầu xác lập được ý tưởng về hệ thống mô phỏng mà nhóm thực hiện.

4.2 Nguyên lý hoạt động của hệ thống cáp treo

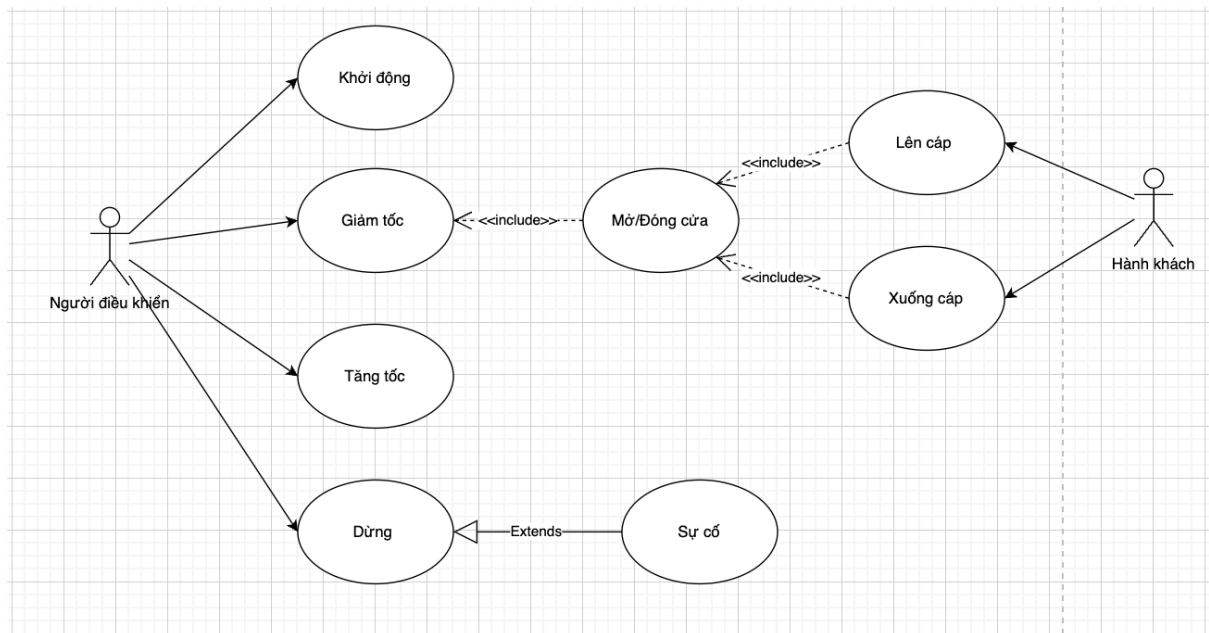
Kết quả đạt được: Hiểu nguyên lý cáp treo, từ đó chọn ra nguyên lý phù hợp cho mạch mô phỏng của hệ thống cáp treo cho phù hợp nhất với khả năng và chi phí. Vì chưa thể thực hiện được hệ thống dẫn động bằng các lớp và sử dụng được các cảm biến để cabin cáp treo giảm tốc độ tự động khi về trạm, chúng em đã thử nghiệm mô phỏng mô hình nhiều lần để tính toán thời gian trung bình và sau đó thực hiện thiết kế hệ thống bằng những hàm tăng giảm tốc độ tự động theo thời gian đo được.

4.3 Mô hình hóa hệ thống bằng FSM và Use Case Diagram

- Mô hình hóa hệ thống sử dụng FSM



- Mô hình hóa hệ thống sử dụng Use Case Diagram



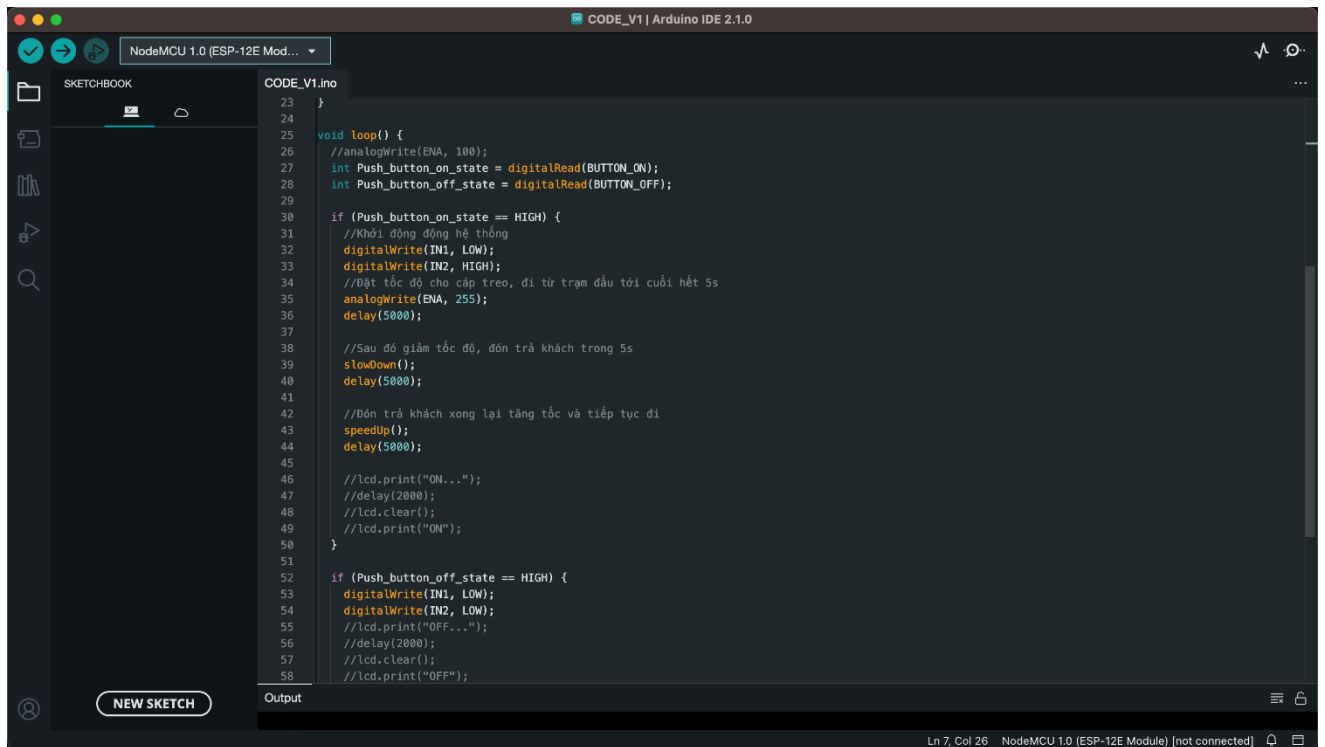
Kết quả đạt được: Thiết lập quy trình của hệ thống phục vụ cho viết chương trình

Kết quả chưa đạt được: Chưa thực hiện được mô hình hóa hệ thống bằng SystemC

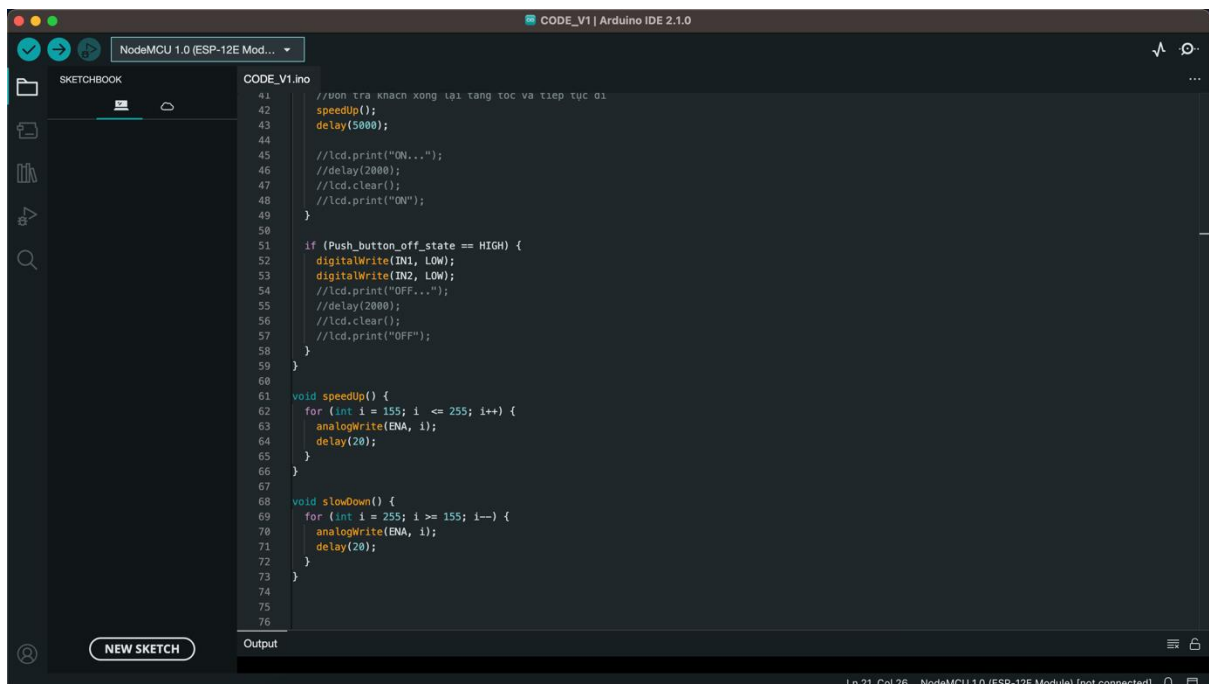
4.3 Thiết kế chương trình điều khiển động cơ DC bằng module điều khiển động cơ L298N và NodeMCU8266

```

1 #include <LiquidCrystal_I2C.h>
2 #include <Wire.h>
3 int ENA = 5;
4 int IN1 = D3;
5 int IN2 = D4;
6 const int BUTTON_ON = D8;
7 const int BUTTON_OFF = D0;
8 //LiquidCrystal_I2C lcd(0x27,16,2);
9
10
11 void setup() {
12   //lcd.init();
13   //lcd.backlight();
14   //lcd.print("Cap treo");
15   pinMode(ENA, OUTPUT);
16   pinMode(IN1, OUTPUT);
17   pinMode(IN2, OUTPUT);
18   pinMode(BUTTON_ON, INPUT);
19   pinMode(BUTTON_OFF, INPUT);
20   digitalWrite(IN1, LOW);
21   digitalWrite(IN2, LOW);
22 }
23
  
```



```
CODE_V1.ino
23 }
24
25 void loop() {
26 //analogWrite(ENA, 100);
27 int Push_button_on_state = digitalRead(BUTTON_ON);
28 int Push_button_off_state = digitalRead(BUTTON_OFF);
29
30 if (Push_button_on_state == HIGH) {
31 //Khởi động động hệ thống
32 digitalWrite(IN1, LOW);
33 digitalWrite(IN2, HIGH);
34 //Đặt tốc độ cho cấp treo, đi từ trạng đầu tới cuối hết 5s
35 analogWrite(ENA, 255);
36 delay(5000);
37
38 //Sau đó giảm tốc độ, đón trả khách trong 5s
39 slowDown();
40 delay(5000);
41
42 //Đón trả khách xong lại tăng tốc và tiếp tục đi
43 speedUp();
44 delay(5000);
45
46 //lcd.print("ON...");
47 //delay(2000);
48 //lcd.clear();
49 //lcd.print("ON");
50 }
51
52 if (Push_button_off_state == HIGH) {
53 digitalWrite(IN1, LOW);
54 digitalWrite(IN2, LOW);
55 //lcd.print("OFF...");
56 //delay(2000);
57 //lcd.clear();
58 //lcd.print("OFF");
59 }
60
61 void speedUp() {
62 for (int i = 155; i <= 255; i++) {
63 analogWrite(ENA, i);
64 delay(20);
65 }
66 }
67
68 void slowDown() {
69 for (int i = 255; i >= 155; i--) {
70 analogWrite(ENA, i);
71 delay(20);
72 }
73 }
74
75
76
```



```
CODE_V1.ino
41 //vận trả khách xong lại tăng tốc và tiếp tục đi
42 speedUp();
43 delay(5000);
44
45 //lcd.print("ON...");
46 //delay(2000);
47 //lcd.clear();
48 //lcd.print("ON");
49 }
50
51 if (Push_button_off_state == HIGH) {
52 digitalWrite(IN1, LOW);
53 digitalWrite(IN2, LOW);
54 //lcd.print("OFF...");
55 //delay(2000);
56 //lcd.clear();
57 //lcd.print("OFF");
58 }
59
60
61 void speedUp() {
62 for (int i = 155; i <= 255; i++) {
63 analogWrite(ENA, i);
64 delay(20);
65 }
66 }
67
68 void slowDown() {
69 for (int i = 255; i >= 155; i--) {
70 analogWrite(ENA, i);
71 delay(20);
72 }
73 }
74
75
76
```

Từ những mô hình Use Case, FSM đã được thiết kế, em đã viết một chương trình điều khiển động cơ bằng ngôn ngữ C trên Aduino IDE.

Kết quả đạt được: Chương trình chạy thành công, đúng với Use Case, FSM. Các button giả lập khởi động, dừng hệ thống hoạt động tốt, màn hình LCD hiển thị đúng và đã có tăng giảm tốc độ đúng như yêu cầu.

Kết quả chưa đạt được:

- Chưa tìm ra cách tối ưu hệ thống tự động hơn
- Chưa thực hiện được phần code cửa tự động đóng/mở cho cabin

Code đã được push lên github của nhóm: https://github.com/toanpm1011/BTL_IoT.git

KẾT LUẬN

Nhờ vào sự giảng dạy, hướng dẫn của Thầy Phạm Văn Tiến và qua quá trình tìm hiểu, chúng em đã hiểu biết thêm được nhiều kiến thức để thiết kế một hệ thống nhúng hoàn chỉnh, từ các bước thiết kế, các công cụ thiết kế tổng quan, các công cụ mô phỏng, kết nối mạch phân cứng và phần mềm,...

Trong khoảng thời gian của kì học, chúng em cũng đã cố gắng tìm hiểu về các kiến thức của môn học. Tuy nhiên, do môn học có rất nhiều kiến thức, mang tính hệ thống trong khi kiến thức và kĩ năng của chúng em còn hạn chế, nhiều phần kiến thức chúng em chưa hiểu sâu nên sản phẩm làm ra chưa được hoàn thiện và còn nhiều điểm thiếu sót, cần phát triển thêm. Chúng em sẽ cố gắng nhiều hơn nữa để nắm bắt tốt các phần kiến thức còn yếu, và để phát triển hệ thống của mình tốt và ngày càng hoàn thiện hơn nữa.

Nhóm chúng em xin chân thành cảm ơn thầy đã nỗ lực hết mình giúp đỡ chúng em!

TÀI LIỆU THAM KHẢO

[Interface L298N DC Motor Driver Module with ESP8266 NodeMCU](#)

[How Cable Cars Work and Detach From The Cable](#)

[How to make CABLE CAR very simple](#)

[ESP8266 NodeMCU Asynchronous Web Server using Arduino IDE and ESPAsyncWebServer library](#)

[Auto Door](#)

[DC Motor Control With Blynk](#)

[ESP8266EX - Datasheet](#)

[Cảm biến khoảng cách HC-SR04](#)