ĐẠI HỌC QUỐC GIA TP HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA ĐIỆN – ĐIỆN TỬ

□···☼···□

BÁO CÁO BÀI TẬP LỚN MÔN
ĐIỆN TỬ CÔNG SUẤT ỨNG DỤNG

ĐỀ TÀI:

THIẾT KẾ VÀ THI CÔNG MẠCH ĐIỀU KHIỂN TỐC ĐỘ VÀ VỊ TRÍ ĐỘNG CƠ DC BẰNG PHƯƠNG PHÁP PID VÀ ỨNG DỤNG ĐIỀU KHIỂN BẰNG SMART PHONE QUA APP BLYNK

Giảng viên hướng dẫn: Th.s Huỳnh Hữu Phương

Sinh viên thực hiện	Mã số sinh viên
Mai Khánh Duy	1912889
Đoàn Huỳnh Quát	1914815
Hoàng Ngọc Thành	1915131
Đỗ Thái Phước	1914760

Thành phố Hồ Chí Minh – tháng 12 năm 2022

MỤC LỤC **TRANG**

LỜI GIỚI THIỆU

Động cơ DC tuy không thông dụng bằng động cơ không đồng bộ 3 pha nhưng đối với những ứng dụng không cần yêu cầu công suất quá lớn và phức tạp thì động cơ DC vẫn có vai trò rất lớn đối với con người: quạt, cửa cuốn, kéo tời,… Do vậy để sử dụng động cơ DC một cách chuẩn chỉnh thì việc điều khiển qua một mạch lái là rất cần thiết.

Bộ điều khiển PID là bộ điều khiển được ra đời cách đây khá lâu ở thế kỷ 20, và là một trong những phát minh quan trọng nhất của lĩnh vực điều khiển tự động, gây nên tiếng vang rất lớn ở thời điểm ấy và đến tận bây giờ, nó vẫn vô cùng phổ biến, không chỉ trong công nghiệp, các hệ thống tự động (do mực nước, nhiệt độ, tốc độ động cơ,…) mà còn trong những ứng dụng nhỏ. Bằng sự phản hồi tín hiệu vòng kín và việc tính toán sai số, kết hợp với các khâu P, I, D, ta có thể điều khiển được ngõ ra theo giá trị đặt mà mình mong muốn

Từ những điều được học trong môn điện tử công suất ứng dụng cùng với tìm hiểu thông tin trên mạng. Thêm sự gợi ý từ thầy Huỳnh Hữu Phương về một ứng dụng điều khiển động cơ qua smartphone, chúng em đã quyết định chọn đề tài: “Thiết kế, thi công mạch lái động cơ DC bằng phương pháp PID và ứng dụng điều khiển bằng smartphone qua app Blynk ”

Xin chân thành cảm ơn những bài giảng của thầy về những kiến thức điện tử công suất để chúng em có thể ứng dụng nó vào một đề tài cụ thể như thế này.

Tp. Hồ Chí Minh, ngày 20 tháng 12 năm 2022

Trưởng nhóm

Đoàn Huỳnh Quát

CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI

TÌM HIỂU CHUNG VỀ ĐỘNG CƠ DC

1. Tổng quan về đề tài

Việc điều khiển một động cơ có vẻ như là một nhiệm vụ dễ dàng: mắc động cơ vào một nguồn thay đổi áp đặt lên phần ứng động cơ được là xong. Nhưng đây không phải là cách hoàn hảo để điều khiển một động cơ, đặc biệt là khi có các thành phần khác ảnh hưởng đến mạch. Do đó sự cần thiết của mạch cầu H là không thể bàn cãi: đóng ngắt nhanh, hiệu suất cao.

Tuy nhiên để đạt được mục tiêu vị trí và tốc độ bằng với giá trị mong muốn thì việc sử dụng thuật toán PID để điều khiển là một giải pháp tiết kiệm, đơn giản mang tính kinh tế. Dễ dàng trong thiết kế và hiệu quả, chính xác trong điều khiển và tính thích nghi cao là những ưu điểm rất quan trọng của bộ điều khiển, và PID có thể đạt được những điều ấy

Ứng dụng được sử dụng trong đề tài này chính là điều khiển động cơ qua mạng không dây thông qua ứng dụng Blynk. Ứng dụng này tuy đơn giản nhưng thể hiện được sự kết hợp của điện tử công suất và IOT

Trong giới hạn đề tài này: chương 2 sẽ trình bày các phần mềm hỗ trợ mà nhóm đã sử dụng để thực hiện đề tài; chương 3 sẽ trình bày về các phần cứng mà nhóm đã sử dụng; chương 4 sẽ trình bày phần lập trình và thuật toán cùng với kết quả đã thực hiện; chương cuối sẽ đánh giá lại kết quả và đưa ra hướng phát triển cho đề tài.

2. Khái niệm, ứng dụng và phân loại động cơ DC

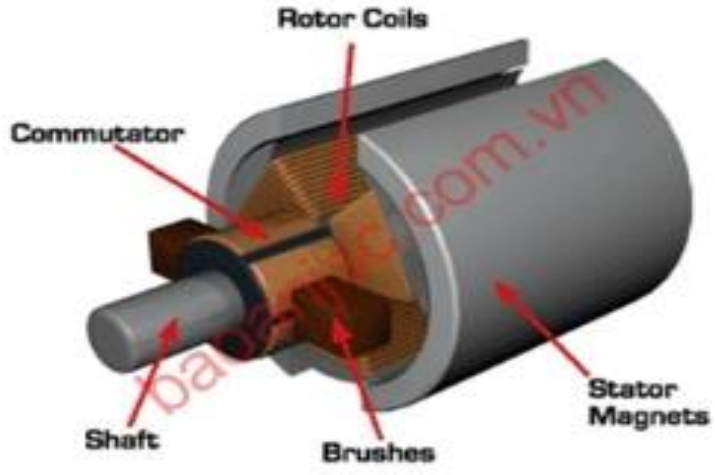
Động cơ DC (DC là viết tắt của từ Direct Current Motors) hay còn gọi là động cơ điện một chiều chính là động cơ được điều khiển bằng dòng có hướng xác định. Cũng có thể nói dễ hiểu hơn thì đây là một loại động cơ chạy bằng nguồn điện áp DC – điện áp 1 chiều (Khác với những điện áp AC xoay chiều). Đầu dây ra của động cơ này thường gồm có hai dây (dây nguồn – VCC và dây tiếp đất – GND). DC motor là một loại động cơ một chiều với động cơ quay liên tục.

Ứng dụng của động cơ DC cũng vô cùng đa dạng và hầu như có mặt trong mọi lĩnh vực của đời sống. Trong tivi, đài FM, máy in- photo, ổ đĩa DC, hay máy công nghiệp…v…v.

Động cơ điện 1 chiều được phân loại theo kích từ thành các loại sau: Kích từ độc lập, song song, nối tiếp và hỗn hợp

3. Cấu tạo động cơ DC:

Gồm có 3 phần chính đó là rotor (phần ứng), stato (phần cảm) và phần cổ góp – chỉnh lưu:

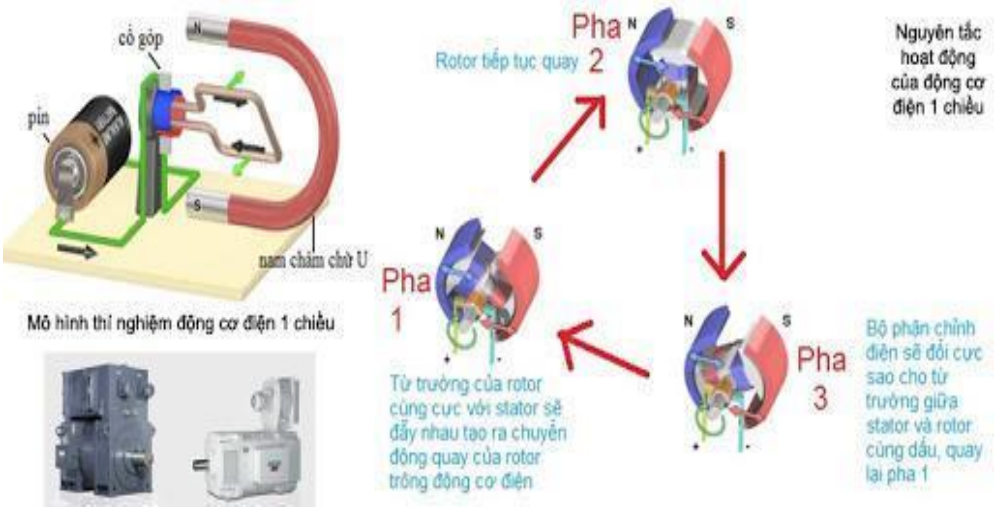


+ Stator của động cơ điện 1 chiều thường sẽ là 1 hoặc nhiều những cặp nam châm vĩnh cửu, hoặc là nam châm điện.

+ Rotor có những cuộn dây quấn và được nối với những nguồn điện một chiều.

+ Bộ phận chỉnh lưu có nhiệm vụ là đổi chiều của dòng điện trong khi chuyển động quay của roto là liên tục. Thông thường thì bộ phận này gồm có một bộ cổ góp và một bộ chổi than để tiếp xúc với cổ góp.

4. Nguyên lý hoạt động động cơ DC



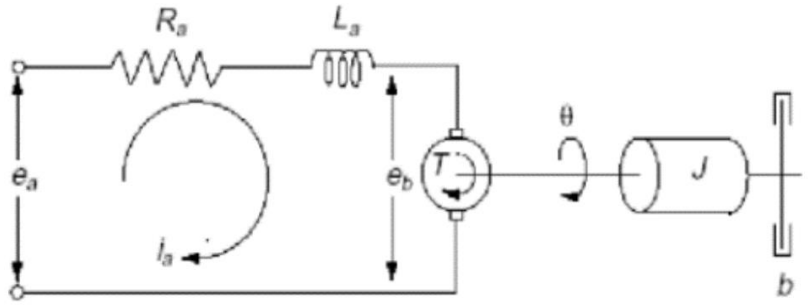
Pha 1: Từ trường của rotor cùng cực với stator nên sẽ đẩy nhau tạo nên chuyển động xoay của rotor.

Pha 2: Rotor sẽ tiếp tục quay

Pha 3: Bộ phận chỉnh điện sẽ đổi cực sao cho từ trường giữa stator và rotor cùng dấu sau đó trở lại pha 1

Nếu như trục của một động cơ điện 1 chiều được kéo bằng 1 lực bên ngoài, động cơ sẽ hoạt động như là một loại máy phát điện một chiều, nó sẽ tạo ra một sức điện động cảm ứng là Electromotive force (EMF). Khi vận hành bình thường thì rotor khi quay sẽ phát ra một điện áp được gọi là sức phản điện động của counter – EMF (hay CEMF) hoặc sức điện động đối kháng, bởi vì nó sẽ đối kháng lại với điện áp ở bên ngoài đặt vào động cơ. Sức điện động này nó sẽ tương tự như sức điện động được phát ra khi mà động cơ được sử dụng như là một máy phát điện (giống lúc chúng ta nối một điện trợ tải vào đầu ra của động cơ và kéo trục của động cơ bằng một ngẫu lực từ bên ngoài). Như vậy thì điện áp đặt trên động cơ sẽ bao gồm 2 thành phần: sức phản điện động và điện áp giáng để tạo ra do điện trở nội của những cuộn dây phần ứng.

5. Mô hình động cơ DC



$$U = E + RI + L\frac{dI}{dt} + \varepsilon$$

$$C_e = k\phi$$

$$M = C_e I = k\phi I$$

$$E = C_e \omega$$

Trong đó các thành phần:

+ R_a: điện trở phần ứng

+ L_a: điện cảm phần ứng

+ J: Moomen quán trình của động cơ

+ B: Hệ số ma sát nhớt

+ K_T: Hằng số momen xoắn

+ K_b: Hằng số sức phản điện

- Hàm truyền động cơ DC, ngõ vào điện áp U, ngõ ra tốc độ động cơ

$$G(s) = \frac{\Omega(s)}{U(s)} = \frac{K_T}{JL_a s^2 + (bL_a + JR_a)s + (bR_a + K_T K_b)}$$

- Do thành phần điện cảm của phần ứng rất nhỏ nên có thể bỏ qua, ta có thể xấp xỉ hàm truyền của động cơ DC thành bậc nhất như sau:

$$G(s) = \frac{\Omega(s)}{U(s)} = \frac{K_T}{JR_a s + (bR_a + K_T K_b)}$$

- Phương trình đặc tính cơ của động cơ một chiều $\omega = \frac{1}{C_e}(U - RI)$

=> Muốn điều chỉnh tốc độ động cơ nhanh hay chậm, ta thay đổi điện áp và dòng điện phần ứng đặt vào động cơ

CHƯƠNG 2: CÁC PHẦN MỀM HỖ TRỢ

1. Phần mềm thiết kế mạch Altinum designer

Altium Designer trước kia có tên gọi quen thuộc là Protel DXP, là một trong những công cụ vẽ mạch điện tử mạnh nhất hiện nay. Được phát triển bởi hãng Altium Limited. Altium designer là một phần mềm chuyên nghành được sử dụng trong thiết kế mạch điện tử. Nó là một phần mềm mạnh với nhiều tính năng thú vị, tuy nhiên phần mềm này còn được ít người biết đến so với các phần mềm thiết kế mạch khác như orcad hay proteus. Altium Designer có một số đặc trưng sau:

- Giao diện thiết kế, quản lý và chỉnh sửa thân thiện, dễ dàng biên dịch, quản lý file, quản lý phiên bản cho các tài liệu thiết kế.

- Hỗ trợ mạnh mẽ cho việc thiết kế tự động, đi dây tự động theo thuật toán tối ưu, phân tích lắp ráp linh kiện. Hỗ trợ việc tìm các giải pháp thiết kế hoặc chỉnh sửa mạch, linh kiện, netlist có sẵn từ trước theo các tham số mới.

- Mở, xem và in các file thiết kế mạch dễ dàng với đầy đủ các thông tin linh kiện, netlist, dữ liệu bản vẽ, kích thước, số lượng…

- Hệ thống các thư viện linh kiện phong phú, chi tiết và hoàn chỉnh bao gồm tất cả các linh kiện nhúng, số, tương tự…

- Đặt và sửa đối tượng trên các lớp cơ khí, định nghĩa các luật thiết kế, tùy chỉnh các lớp mạch in, chuyển từ schematic sang PCB, đặt vị trí linh kiện trên PCB.

- Mô phỏng mạch PCB 3D, đem lại hình ảnh mạch điện trung thực trong không gian 3 chiều, hỗ trợ MCAD-ECAD, liên kết trực tiếp với mô hình STEP, kiểm tra khoảng cách cách điện, cấu hình cho cả 2D và 3D

- Hỗ trợ thiết kế PCB sang FPGA và ngược lại.

=> Từ đó, chúng ta thấy Altium designer có nhiều điểm mạnh so với các phần mềm khác như đặt luật thiết kế, quản lý đề tài mô phỏng dễ dàng, giao diện thân thiện,…

2. Phần mềm lập trình cho vi điều khiển Arduino IDE

Arduino IDE là một phần mềm mã nguồn mở chủ yếu được sử dụng để viết và biên dịch mã vào module Arduino.

Đây là một phần mềm Arduino chính thức, giúp cho việc biên dịch mã trở nên dễ dàng mà ngay cả một người bình thường không có kiến thức kỹ thuật cũng có thể làm được.

Nó có các phiên bản cho các hệ điều hành như MAC, Windows, Linux và chạy trên nền tảng Java đi kèm với các chức năng và lệnh có sẵn đóng vai trò quan trọng để gỡ lỗi, chỉnh sửa và biên dịch mã trong môi trường.

Có rất nhiều các module Arduino như Arduino Uno, Arduino Mega, Arduino Leonardo, Arduino Micro và nhiều module khác.

Mỗi module chứa một bộ vi điều khiển trên bo mạch được lập trình và chấp nhận thông tin dưới dạng mã.

Mã chính, còn được gọi là sketch, được tạo trên nền tảng IDE sẽ tạo ra một file Hex, sau đó được chuyển và tải lên trong bộ điều khiển trên bo.

Môi trường IDE chủ yếu chứa hai phần cơ bản: Trình chỉnh sửa và Trình biên dịch, phần đầu sử dụng để viết mã được yêu cầu và phần sau được sử dụng để biên dịch và tải mã lên module Arduino.

Môi trường này hỗ trợ cả ngôn ngữ C và C ++.

3. Ứng dụng BLYNK

Blynk là một nền tảng với các ứng dụng iOS và Android để điều khiển Arduino, Raspberry Pi và các ứng dụng tương tự qua Internet. Nó là một bảng điều khiển kỹ thuật số nhờ đó bạn có thể xây dựng giao diện đồ họa cho dự án của mình bằng cách kéo và thả các widget.

Blynk không bị ràng buộc với một số bo hoặc shield cụ thể. Thay vào đó, nó hỗ trợ phần cứng mà bạn lựa chọn. Cho dù Arduino hoặc Raspberry Pi của bạn được liên kết với Internet qua Wi-Fi, Ethernet hoặc chip ESP8266, Blynk sẽ giúp bạn online và sẵn sàng cho IoT.

Blynk được thiết kế cho IoT. Nó có thể điều khiển phần cứng từ xa, nó có thể hiển thị dữ liệu cảm biến, nó có thể lưu trữ dữ liệu, trực quan hóa và làm nhiều thứ hay ho khác.

Có ba thành phần chính trong nền tảng:

+ Ứng dụng Blynk - cho phép bạn tạo giao diện cho các dự án của mình bằng cách sử dụng các widget khác nhau.

+ Blynk Server - chịu trách nhiệm về tất cả các giao tiếp giữa điện thoại thông minh và phần cứng. Có thể sử dụng Blynk Cloud hoặc chạy cục bộ máy chủ Blynk riêng của mình. Nó là mã nguồn mở, có thể dễ dàng xử lý hàng nghìn thiết bị và thậm chí có thể được khởi chạy trên Raspberry Pi.

+ Thư viện Blynk - dành cho tất cả các nền tảng phần cứng phổ biến - cho phép giao tiếp với máy chủ và xử lý tất cả các lệnh đến và lệnh đi. Mỗi khi bạn nhấn một nút trong ứng dụng Blynk, thông điệp sẽ truyền đến không gian của đám mây Blynk, và tìm đường đến phần cứng của bạn.

CHƯƠNG 3: THIẾT KẾ PHẦN CỨNG

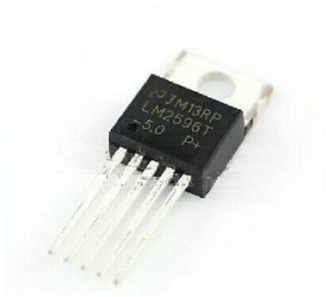
I. THÀNH PHẦN CẤU TẠO PHẦN CỨNG

1. Mạch giảm áp DC LM2596 3A

a) Giới thiệu IC LM2596

Mạch sử dụng IC LM 2596 có dòng ra 3A là 1 IC ổn áp dạng xung DC-DC. Điện áp đầu vào trong dải từ 4.5V – 40V. Điện áp đầu ra điều chỉnh được trong khoảng từ 1.5V – 37V, dòng điện đầu ra đạt 3A hiệu suất cao nhờ cơ chế băm xung ở tần số lên tới 150KHz. Trong quá trình hoạt động LM2596 luôn được đặt trong các chế độ bảo vệ quá nhiệt và quá dòng

DIP – Dạng chân cắm



SMD – Dạng chân dán

Thông số cơ bản của IC LM 2596

- IC LM 2596 có 5 chân

+ Chân 1: V in từ 4.5 ~ 40V

+ Chân 2: Vout

+ Chân 3: GND

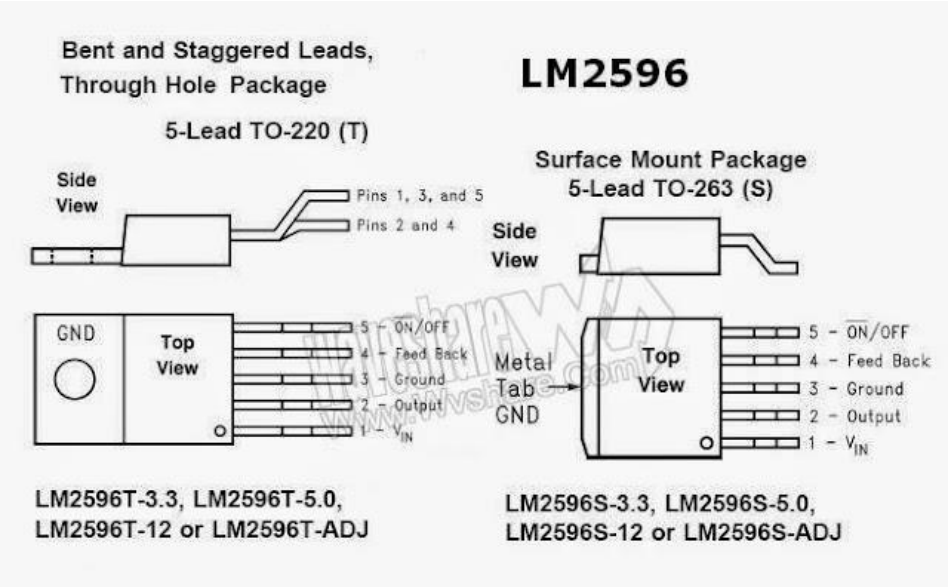
+ Chân 4: Feedback

+ Chân 5: ON/OFF chân bật/ tắt mức logic

- Điện áp đầu vào: 4.5 – 40V DC

- Điện áp đầu ra 1.5~37V

- Dòng ra max (3A)



Data Sheet của IC LM2596

b) Giới thiệu mạch giảm áp DC LM2596 3A

Hiệu suất và tính năng mô-đun:

+ Mô-đun BUCK

+ Điện áp đầu vào: 4-38V (Vui lòng cố gắng không vượt quá 38V làm đầu vào)

+ Điện áp đầu ra: 1.25-36V có thể điều chỉnh liên tục

+ Dòng điện đầu ra: 0-5A, được khuyến nghị sử dụng trong vòng 4,5A

+ Công suất đầu ra: Nên sử dụng trong vòng 75W.

+ Nhiệt độ làm việc: -40 ~ + 85 độ

+ Tần số làm việc: 180KHz

+ Hiệu suất chuyển đổi: lên đến 96%

+ Tỷ lệ điều chỉnh tải: S (I) ≤ 0,8%

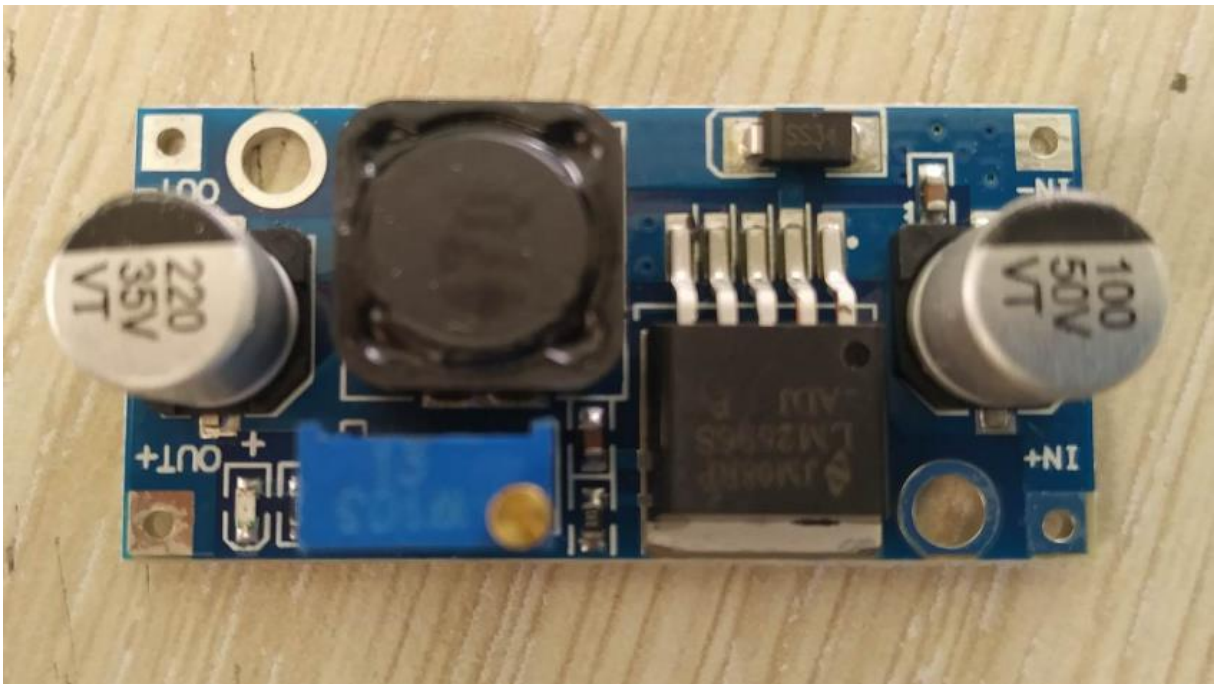
+ Tỷ lệ điều chỉnh điện áp: S (u) ≤ 0,8%

+ Bảo vệ ngắn mạch: Có (dòng giới hạn 8A)

+ Bảo vệ quá nhiệt: Có (tự động tắt đầu ra sau khi quá nhiệt)

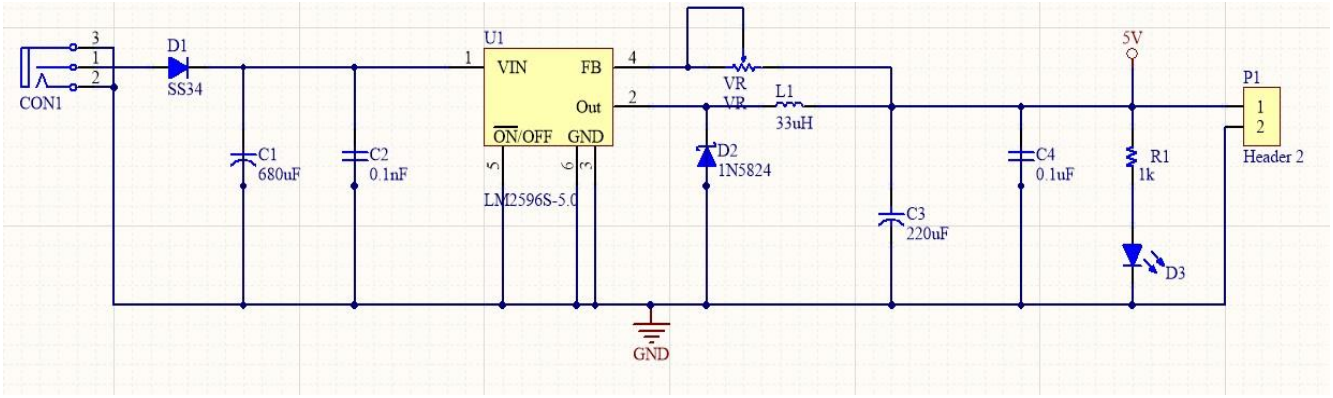
+ Bảo vệ kết nối ngược đầu vào: Không có

Phạm vi ứng dụng: Mô-đun này có thể được áp dụng cho trường buck trong đó điện áp đầu vào cao hơn điện áp đầu ra, chẳng hạn như, máy biến áp nguồn, bộ nguồn có thể điều chỉnh tự làm, nguồn điện bút ô tô 24V, thiết bị công nghiệp lùi,

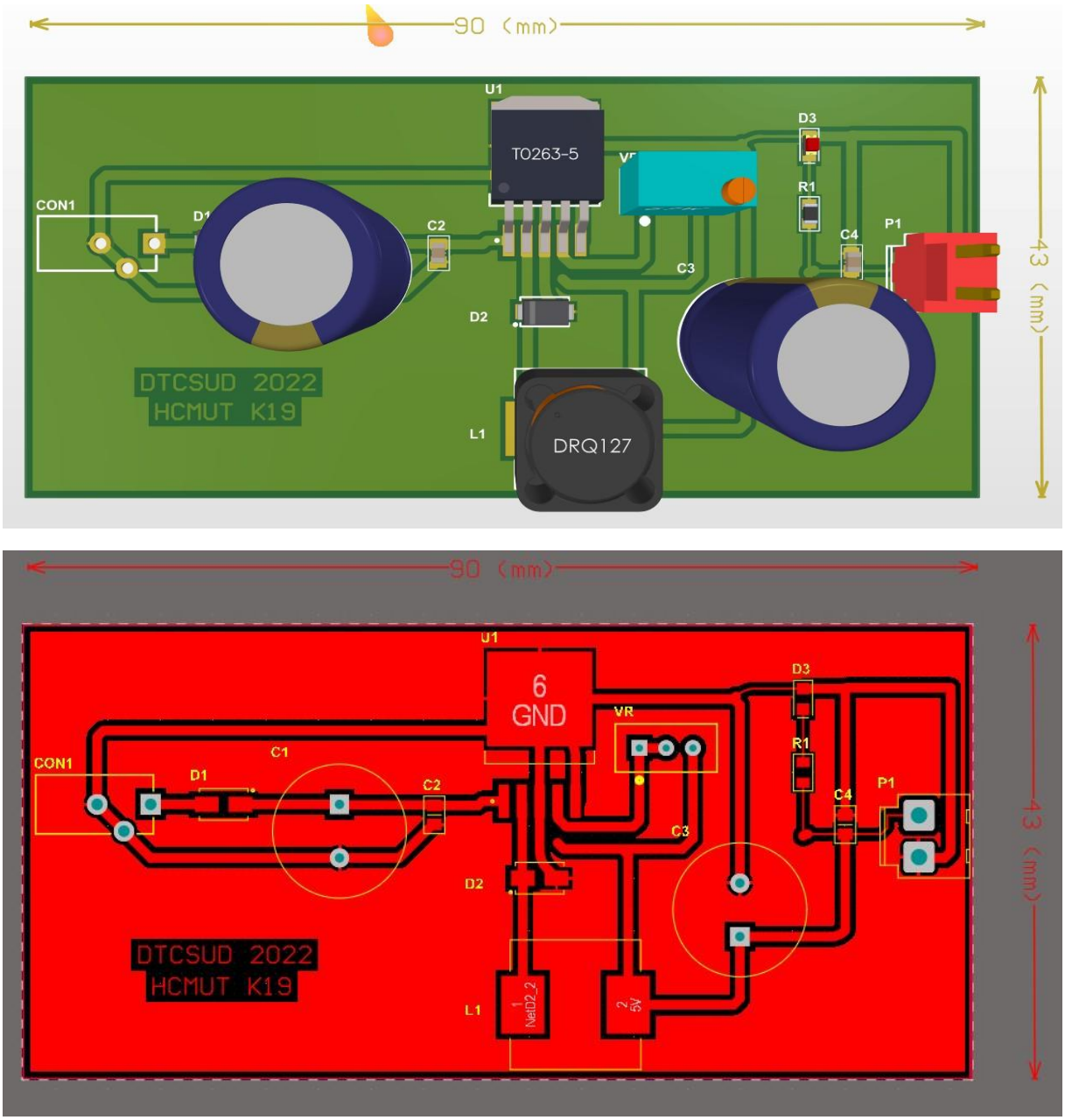


Mạch thực tế

Mạch nguyên lý thiết kế trên Alitinum



Mạch 3D và 2D trên Alitinum



Giá bán 16.000VND

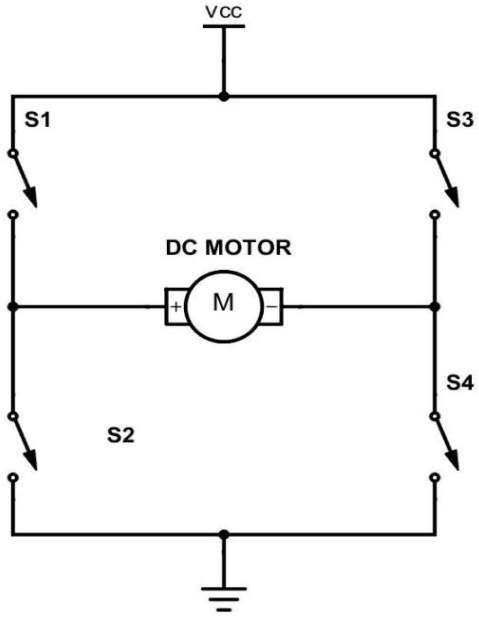
2. Mạch điều khiển động cơ DC-DC Motor Drive XY-160D

Mach sử dụng nguyên lý của mạch cầu H để điều khiển động cơ

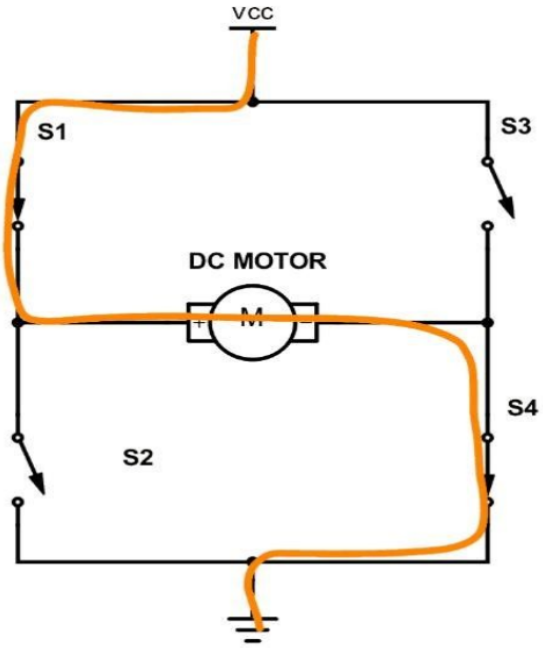
a) Giới thiệu về mạch cầu H

Mạch cầu H là một mạch đơn giản dùng để điều khiển động cơ DC quay thuận hoặc quay nghịch. Trong thực tế, có nhiều kiểu mạch cầu H khác nhau tùy vào cách chúng ta lựa chọn linh kiện có dòng điện, áp điều khiển lớn hay nhỏ, tần số xung PWM… Và chúng sẽ quyết định đến khả năng điều khiển của cầu H

Mô hình sơ đồ mạch cầu H

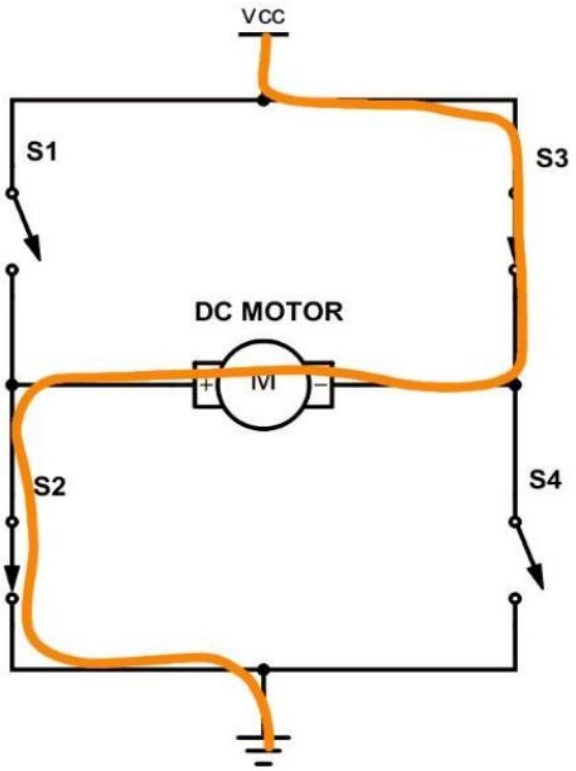


Một động cơ DC có thể quay thuận hoặc quay nghịch tùy thuộc vào cách bạn mắc cực âm và dương cho motor đó. Khi ta đóng S1 và S4, ta đã cho A nối với cực dương (+) và B nối với cực âm (-) của nguồn, một dòng điện chạy từ nguồn qua S1 qua động cơ qua S4 về mass làm động cơ quay theo chiều thuận.



Ngược lại khi ta đóng S2 và S3 thì động cơ quay nghịch

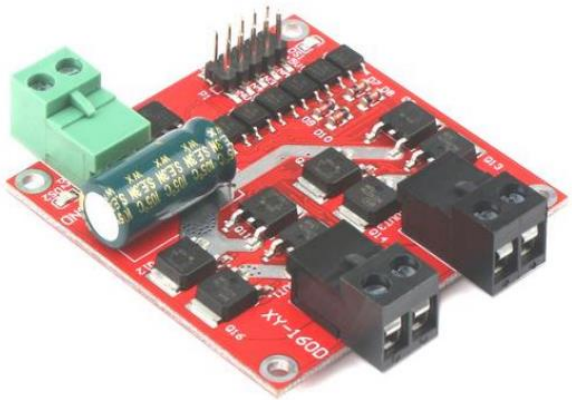
Như vậy, mạch cầu H dùng để đảo chiều quay động cơ .



Lưu ý, không bao giờ được phép đóng cùng lúc S1 và S2 hoặc S3 và S4 hoặc thậm chí là đóng cùng lúc 4 công tắc. Nếu bạn làm như vậy, bạn đã tạo ra một đường dẫn trực tiếp từ Vcc xuống GND và gây ra hiện tượng ngắn mạch. Acquy của bạn sẽ bị hỏng và nghiêm trọng hơn có thể cháy nổ mạch.

b) Giới thiệu bộ điều khiển động cơ cầu H kép 7A/160W XY-160D

Đây là bộ điều khiển động cơ DC kép cấu hình thấp siêu nhỏ dành cho các dự án hạn chế về không gian, có khả năng cung cấp công suất cao lên đến 7A trên mỗi kênh đầu ra. Mạch sử dụng logic tương tự như trình điều khiển động cơ L298, có điều khiển trình điều khiển bằng 3 chân tín hiệu (IN1, IN2, ENABLE). Trình điều khiển động cơ này được điều khiển bởi MOSFET công suất cao, với bộ ghép quang tín hiệu điều khiển được cách ly để bảo vệ mọi sự cố về mạch điện và vòng tiếp đất. Ngoài ra thể điều khiển trình điều khiển này bằng cả logic 3,3V và 5V.

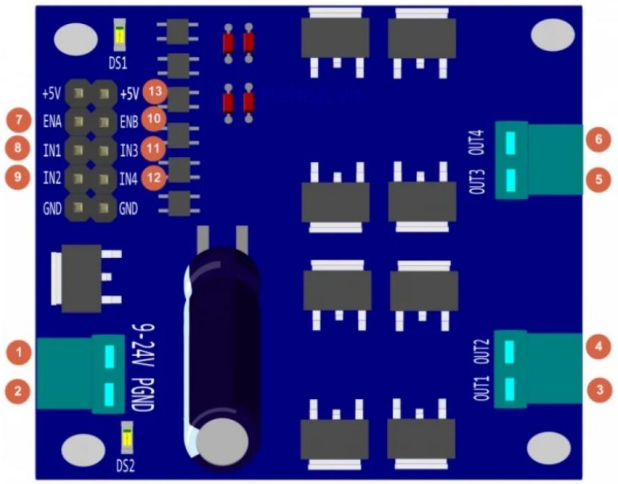


Thông số kỹ thuật

+ Điện áp nguồn: 7 ~ 24 VDC. (Giới hạn: 6.5~27VDC).

+ Mức tín hiệu điều khiển (Tương thích 3.3V/5V)
+ Logic Cao (H): DC 3.0 ~ 6.5V
+ Logic Thấp (L): DC0 ~ 0.8V
+ Kênh đầu ra: 2.
+ Dòng tín hiệu điều khiển: 3 ~ 11 mA (Từng tuyến).
+ Dòng hoạt động liên tục tối đa: 7A.
+ Dòng đỉnh: 50A.
+ Điều khiển tốc độ PWM: 0~10KHz.
+ Độ rộng xung hợp lệ tối thiểu: 5us.
+ Nhiệt độ làm việc: -25 ~ 85°C.
+ Kích thước (LxWxH): 55 x 55 x 13(mm).
+ Trọng lượng: 32g

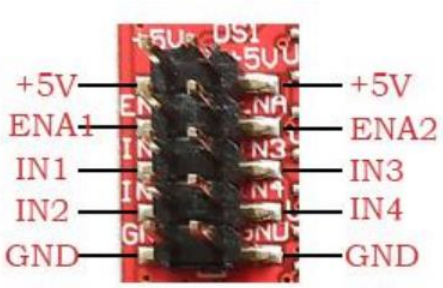
Sơ đồ chân:



STT	Tên chân	Mô tả
1	9~24VDC	Nguồn điện tải
2	PGND	Cung cấp điện cho Ground
3	OUT1	Động cơ 1 đầu ra +
4	OUT2	Động cơ 1 đầu ra -
5	OUT3	Động cơ 2 đầu ra +
6	OUT4	Động cơ 2 đầu ra -
7	ENA	Kích hoạt/điều khiển PWM đầu vào của motor 1
8	IN1	Motor 1 Control input
9	IN2	Motor 1 Control input
10	ENB	Kích hoạt/điều khiển PWM đầu vào của motor 2

11	IN3	Motor 2 Control input
12	IN4	Motor 2 Control input
13	13	Nguồn +5V cho mạch điều khiển logic
14	GND	Power Supply Ground for Logic Control Circuit

Control Logic Function:



Bảng logic điều khiển Motor-1

IN1	IN2	ENA1	OUT1-OUT2
0	0	X	Motor Braking
1	1	X	Floating
1	0	PWM	PWM thuận + Điều khiển tốc độ
0	1	PWM	PWM ngược + Điều khiển tốc độ
1	0	1	Tốc độ thuận tối đa
0	1	1	Tốc độ ngược tối đa

Bảng logic điều khiển Motor-2

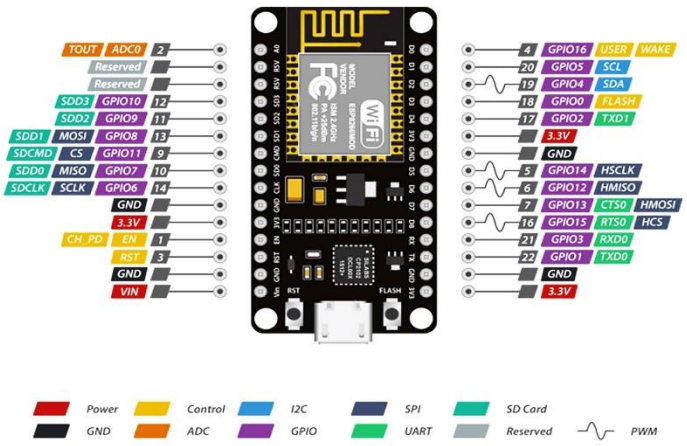
IN1	IN2	ENA2	OUT1-OUT2
0	0	X	Motor Braking
1	1	X	Floating
1	0	PWM	PWM thuận + Điều khiển tốc độ
0	1	PWM	PWM ngược + Điều khiển tốc độ
1	0	1	Tốc độ thuận tối đa
0	1	1	Tốc độ ngược tối đa

- Giá bán: 205.000 VNĐ

3. Kit thu phát wifi ESP 8266

Kit RF thu phát Wifi ESP8266 NodeMCU Lua CP2102 là kit phát triển dựa trên nền chip Wifi SoC ESP8266 với thiết kế dễ sử dụng và đặc biệt là có thể sử dụng trực

tiếp trình biên dịch của Arduino để lập trình và nạp code, điều này khiến việc sử dụng và lập trình các ứng dụng trên ESP8266 trở nên rất đơn giản.



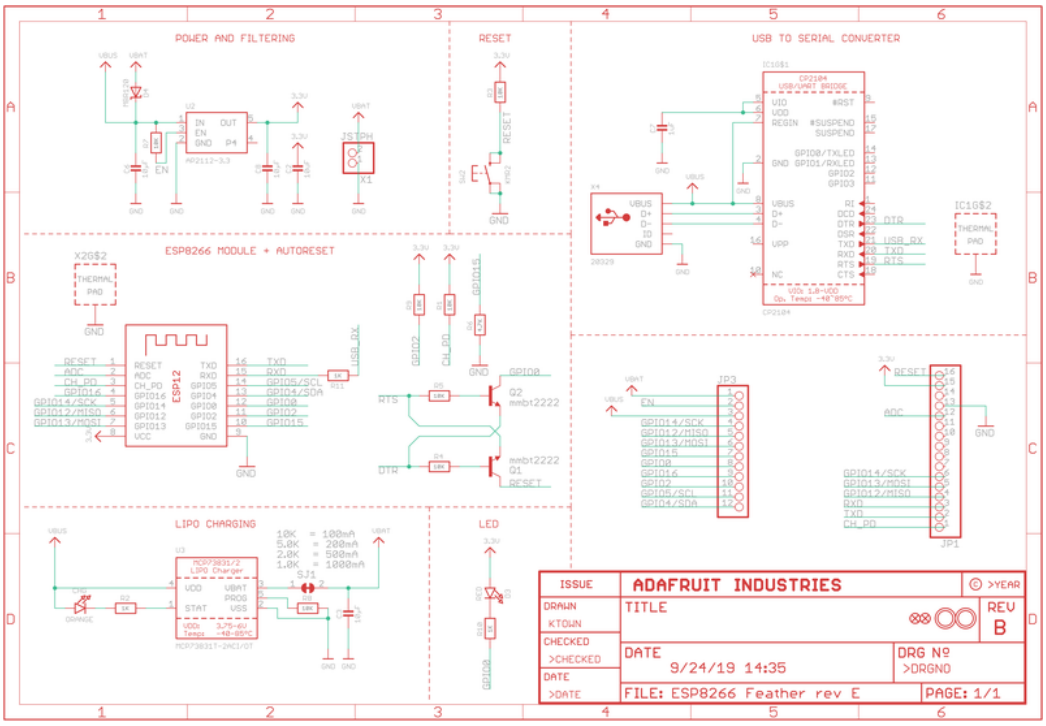
Kit RF thu phát Wifi ESP8266 NodeMCU Lua CP2102 được dùng cho các ứng dụng cần kết nối, thu thập dữ liệu và điều khiển qua sóng Wifi, đặc biệt là các ứng dụng liên quan đến IoT.

Kit RF thu phát Wifi ESP8266 NodeMCU Lua CP2102 sử dụng chip nạp và giao tiếp UART CP2102 có độ ổn định và độ bền cao và khả năng tự nhận Driver trên tất cả các hệ điều hành Windows và Linux.

Thông số kỹ thuật:

- Tương thích các chuẩn wifi : 802.11 b/g/n
- Hỗ trợ: Wi-Fi Direct (P2P), soft-AP
- Tích hợp TCP/IP protocol stack
- Tích hợp TR switch, balun, LNA, power amplifier and matching network
- Tích hợp bộ nhân tần số, ổn áp, DCXO and power management units
- +25.dBm output power in 802.11b mode
- Power down leakage current of <10uA
- Integrated low power 32-bit CPU could be used as application processor
- SDIO 1.1/2.0, SPI, UART
- STBC, 1×1 MIMO, 2×1 MIMO
- A-MPDU & A-MSDU aggregation & 0.4ms guard interval
- Wake up and transmit packets in < 2ms
- Dòng tiêu thụ ở Standby Mode < 1.0mW (DTIM3)

Sơ đồ Schematic:

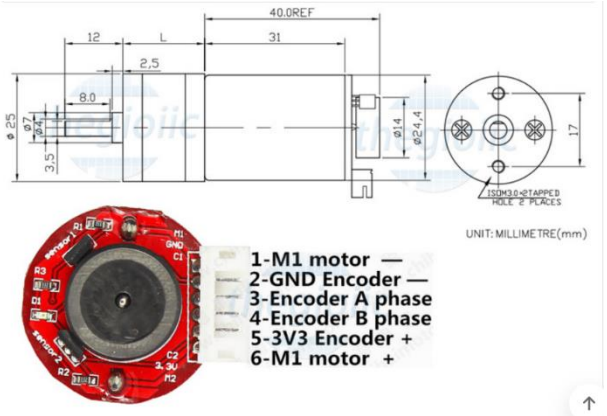


- Giá bán: 92.000VNĐ

4. Động cơ GA25-370 có encoder 12VDC, 130rpm

Động cơ giảm tốc có encoder GA25-370 hoạt động ở điện áp 12 VDC, tốc độ quay 130 vòng/phút, thường được sử dụng trong các ứng dụng cần xác định tốc độ, vị trí, chiều quay của động cơ DC: Robot dò đường, xe hai bánh tự cân bằng,...

Động cơ giảm tốc GA25-370 có encoder cấu tạo thực tế là động cơ giảm tốc DC bình thường có gắn thêm phần Encoder để có thể trả xung về vi điều khiển giúp xác định vị trí, vận tốc,... từ đó vi điều khiển tác động lại động cơ qua mạch công suất, sử dụng các thuật toán để điều khiển ngược lại động cơ.



Thông số kỹ thuật:

Tốc độ quay	130rpm
Điện áp định mức	12VDC

Điện áp hoạt động	6-12VDC
Loại	Có chổi than
Đường kính trục	4mm
Chiều dài động cơ	55mm
Đường kính động cơ	25mm
Tỷ số truyền	1:45
Xung / vòng	11 xung
Chiều dài trục	8mm

- Giá bán động cơ + encoder: 203.000VNĐ

5. Bộ nguồn tổ ong 12V 10A

Nguồn tổ ong 12V 10A Power Supply được thiết kế để chuyển đổi điện áp từ nguồn xoay chiều 180/240VAC thành nguồn một chiều 12VDC để cung cấp cho các thiết bị hoạt động. Nguồn tổ ong này được sử dụng rộng rãi trong các thiết bị công nghiệp và dân dụng.

Nguồn tổ ong 12V 10A Power Supply thích hợp với các hệ thống yêu cầu nguồn có công suất cao (dòng tải lớn) như: các hệ thống chiếu sáng led, led quảng cáo, camera, loa đài, máy in 3D,…

Thông số của Nguồn tổ ong 12V 10A Power Supply

- Model: S-120-12
- Điện áp đầu vào 180VAC hoặc 240VAC thay đổi qua công tắc gạt bên trong
- Điện áp đầu ra 9VDC~14VDC chỉnh được thông qua núm chỉnh ADJ có sẵn trên nguồn
- Công suất: 120W
- Dòng điện đầu ra: Max 10A
- Tần số đầu vào: 50-60Hz
- Nhiệt độ hoạt động: 0 ℃ ~ 40 ℃
- Kích thước sản phẩm: 190 * 99 * 42.5 mm

- Cổng vào: N (dây nguội) – L (dây nóng) – Cổng tiếp chất chống dòng rò
- Cổng ra: 2 (+) và 2 (-). Trọng lượng: 416g

- Giá bán: 150.000VNĐ

6. Modul hạ áp 12VDC xuống 3.3-5V

Mạch chuyển đổi điện áp 12V/5V/3.3V có chức năng chuyển đổi nhiều đầu ra điện áp từ điện 12V ngõ vào.

Ngõ ra của mạch gồm 3 cụm điện áp gồm 12V, 5V và 3.3V phù hợp với nhiều ứng dụng khác nhau.

Tính năng sản phẩm: Mạch chuyển đổi điện áp 12V/5V/3.3V có chức năng chuyển đổi nhiều đầu ra điện áp từ điện 12V ngõ vào. Ngõ ra của mạch gồm 3 cụm điện áp gồm 12V, 5V và 3.3V phù hợp với nhiều ứng dụng khác nhau.

Thông số kỹ thuật:

Điện áp ngõ vào: 6 – 12 VDC

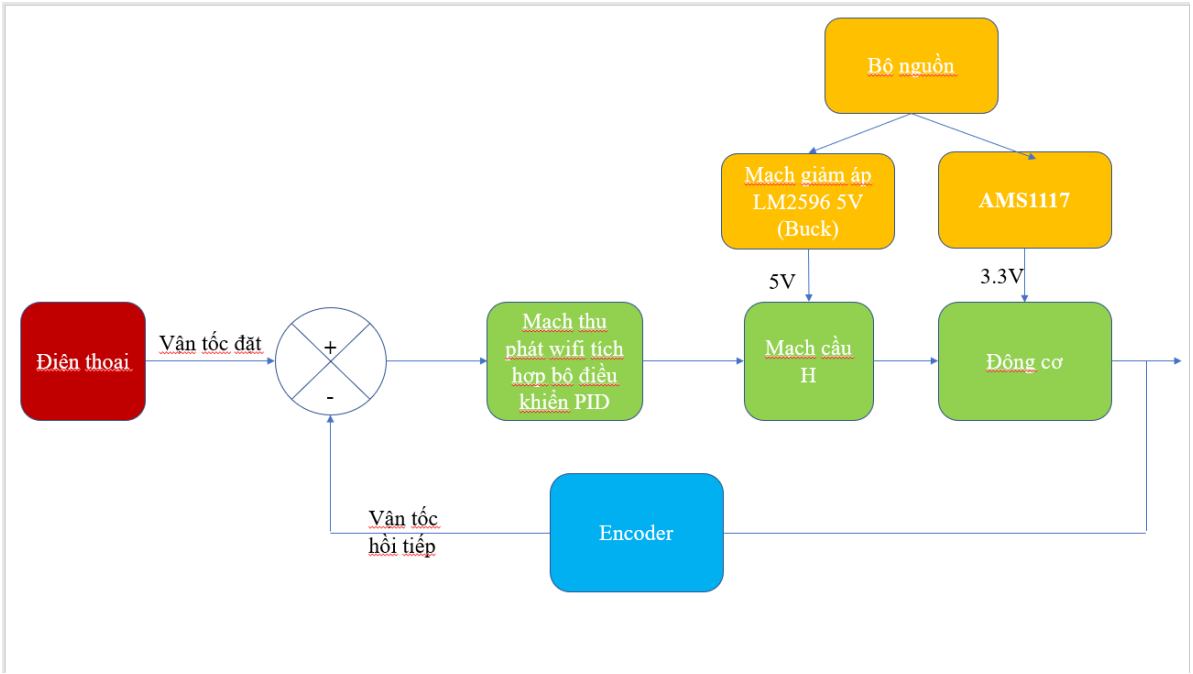
Điện áp ngõ ra: 3.3V – 5V – điện ngõ vào

Có đèn báo nguồn

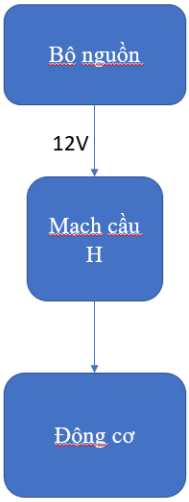
Có công tắc ON/OFF nguồn

II. SƠ ĐỒ KHỐI MẠCH THIẾT KẾ PHẦN CỨNG

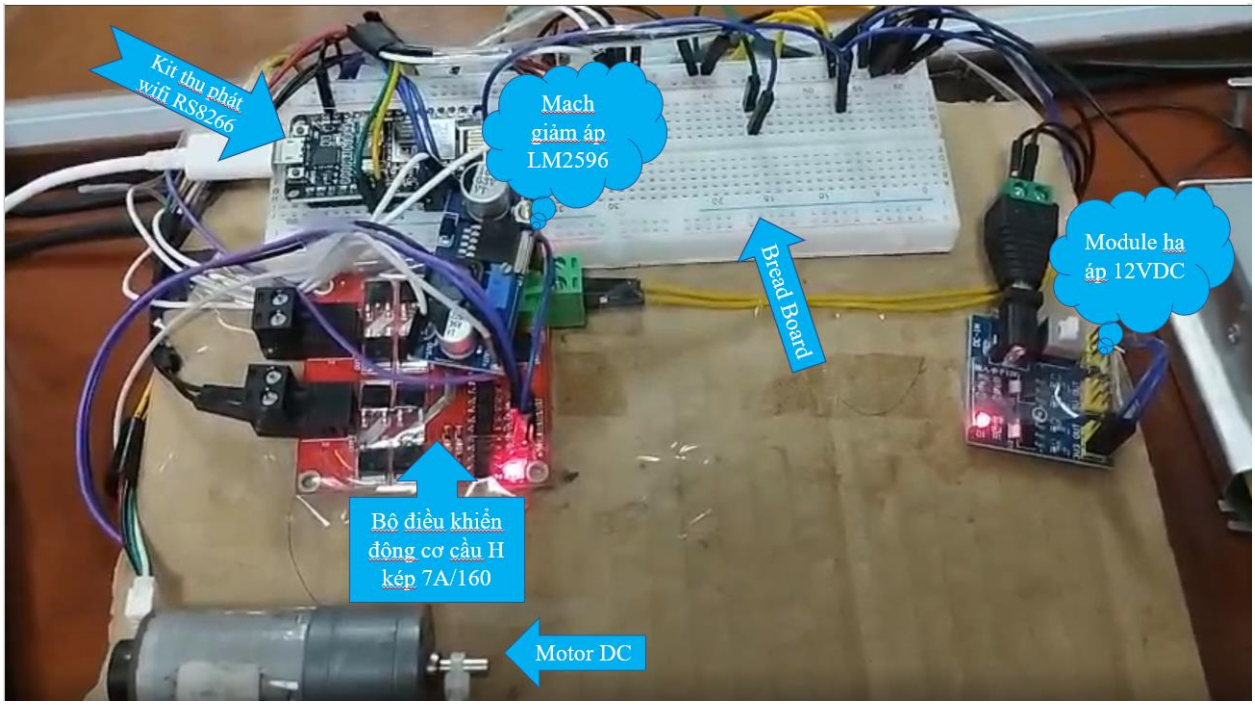
1. Mạch điều khiển



2. Mạch động lực



III. HÌNH ẢNH THỰC TẾ KHI THIẾT KẾ

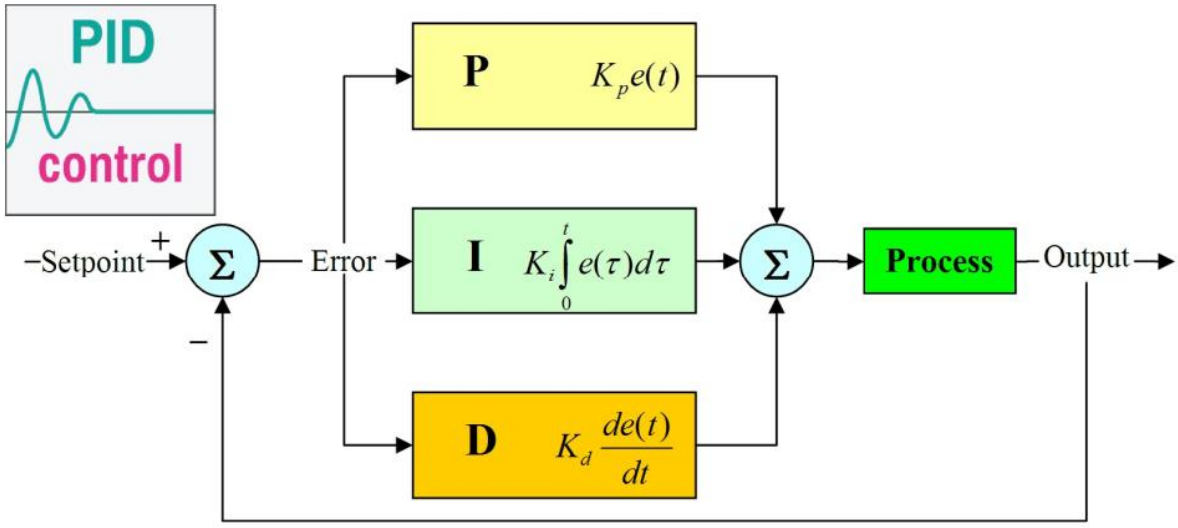


CHƯƠNG 4
LẬP TRÌNH VI ĐIỀU KHIỂN VÀ THỰC NGHIỆM HỆ THỐNG

I. LÝ THUYẾT BỘ ĐIỀU KHIỂN PID

1. Sơ đồ khối và thuật toán

Bộ điều khiển PID bao gồm 3 thông số riêng biệt, do đó đôi khi nó còn được gọi là bộ điều khiển ba khâu: tỉ lệ (P), tích phân (I) và đạo hàm (D). Giá trị tỷ lệ xác định tác động của sai số hiện tại, giá trị tích phân xác định tác động của tổng các sai số quá khứ và giá trị vi phân xác định tác động của tốc độ biến đổi sai số. Tổng chập của ba tác động này dùng để điều chỉnh quá trình thông qua một phần tử điều khiển như vị trí của van điều khiển hay bộ nguồn của phần tử gia nhiệt.



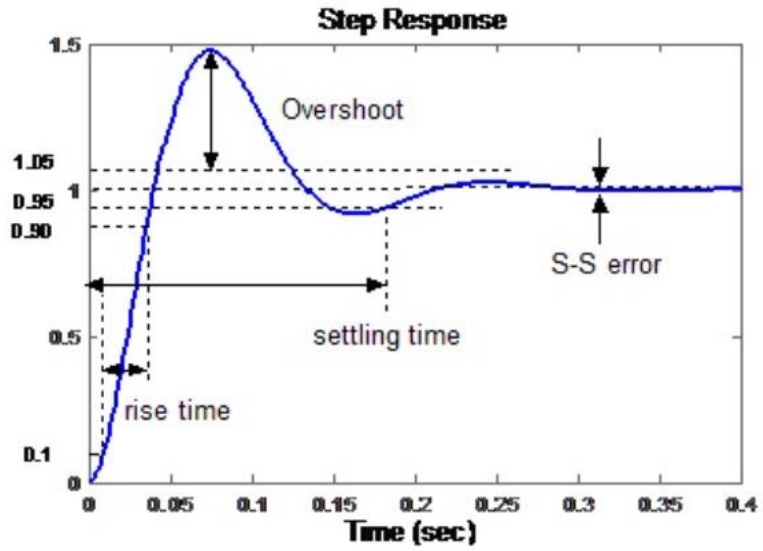
2. Hàm truyền hiệu chỉnh PID $HC = K_p + \frac{K_i}{s} + K_d s = \frac{(T_a s+1)(T_b s+1)}{T_i s}$

Trong đó: Kp, Ki, Kd: là các hệ số khuếch đại tương ứng với các khâu tỷ lệ, tích phân, vi phân, được sử dụng khi điều khiển bằng chương trình

Ta, Tb, Ti là các thời hằng, được sử dụng khi tính toán hiệu chỉnh. Hàm truyền PID có 2 zero và 1 cực ở gốc tọa độ. Theo nguyên lý khử cực – zero, các zero được dùng để loại bỏ các cực không mong muốn. Để hệ thống có đáp ứng nhanh, ta ưu tiên loại bỏ các thời hằng lớn (cực gần gốc tọa độ). Khâu tích phân được thêm vào nhằm triệt tiêu sai số xác lập, nhưng thời hằng tích phân Ti xác định đặc tính quá độ mong muốn

3. Ảnh hưởng của các thông số Kp, Ki, Kd lên chất lượng quá độ của hệ thống

Đ. ứng vòng kín (C.L. response)	T. gian tăng (Rise time)	Vọt lố (Overshoot)	T. gian quá độ (Settling time)	Sai số xác lập (Steady-state err.)
K_P	Giảm	Tă ng	Ít thay đổi	Tă ng
K_I	Giảm	Tăng	Tăng	Không xác định
K_D	Ít thay đổi	Giảm	Giảm	Thay đổi ít



4. Cách chỉnh thông số Kp, Ki, Kd :

Trình tự chỉnh bộ điều khiển PID bằng tay :

B1 : Đặt Ki= Kd = 0. Tăng dần Kp đến hệ số khuếch đại giới hạn Kgh (là hệ số làm cho hệ thống ở biên giới ổn đinh)

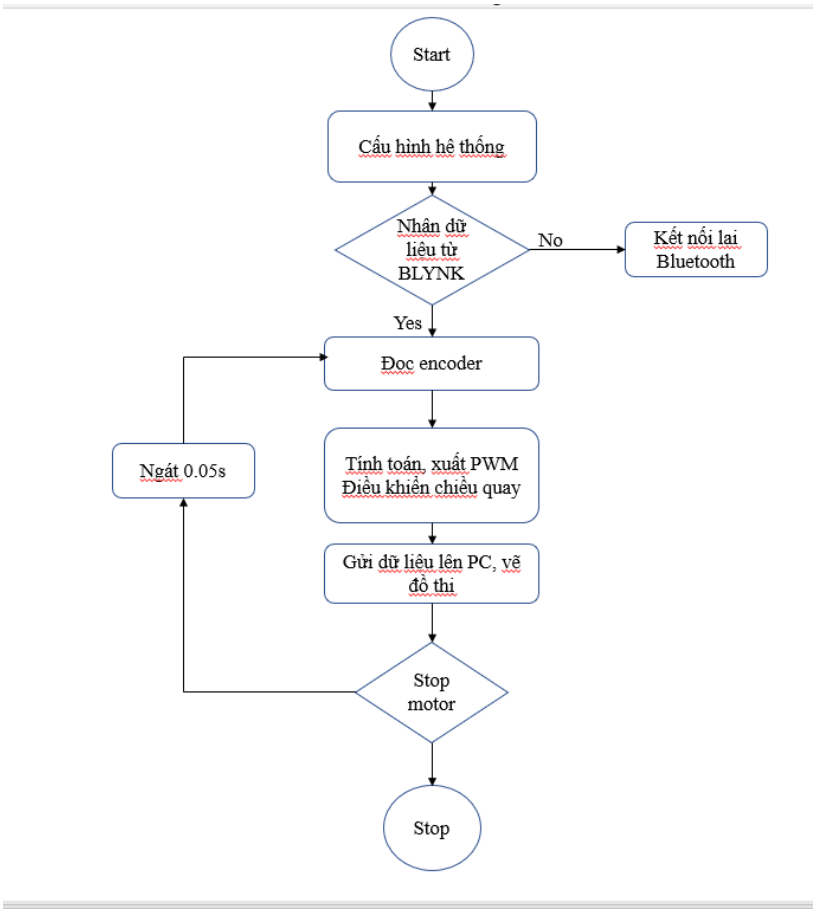
B2 : Đặt Kp = Kgh / 2 (hoặc chỉnh Kp để Pot = 25%)

B3 : Tăng dần Ki đến khi sai số xác lập triệt tiêu trong thời gian đủ nhanh (Ki quá lớn có thể làm hệ thống mất ổn đinh)

B4 : Tăng dần Kd để giảm độ vọt lố và thời gian xác lập (Kd quá lớn có thể làm hệ vọt lố trở lại)

B5 : Tinh chỉnh Kp,Ki,Kd để đáp ứng đạt yêu cầu mong muốn

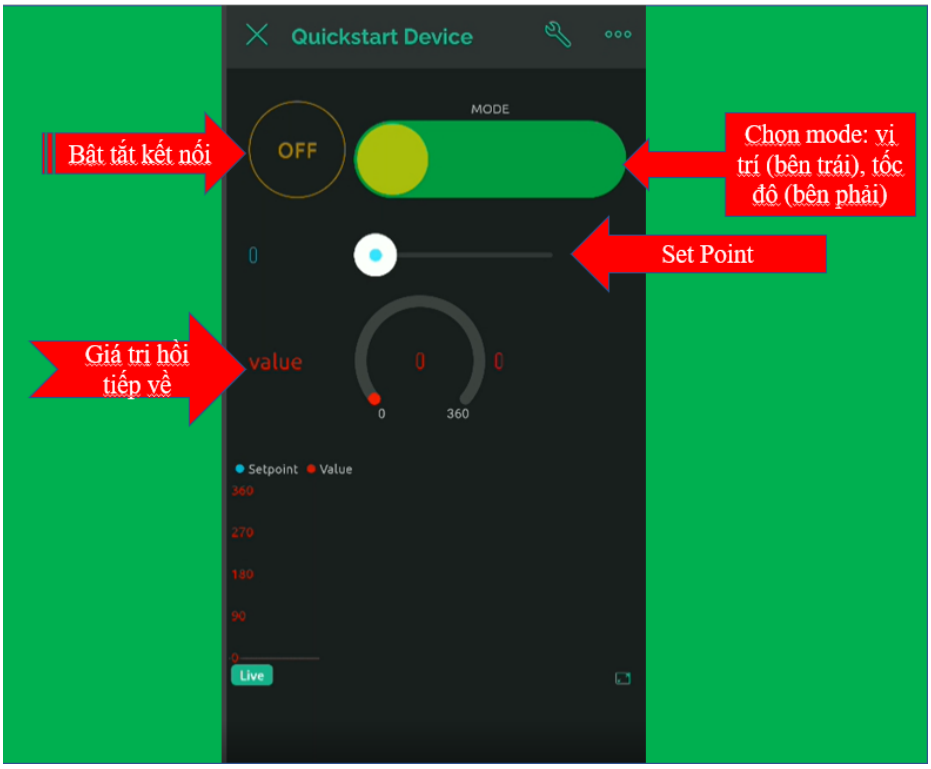
II. LƯU ĐỒ GIẢI THUẬT HỆ THỐNG:



III. THỰC NGHIỆM HỆ THỐNG KHI CHẠY

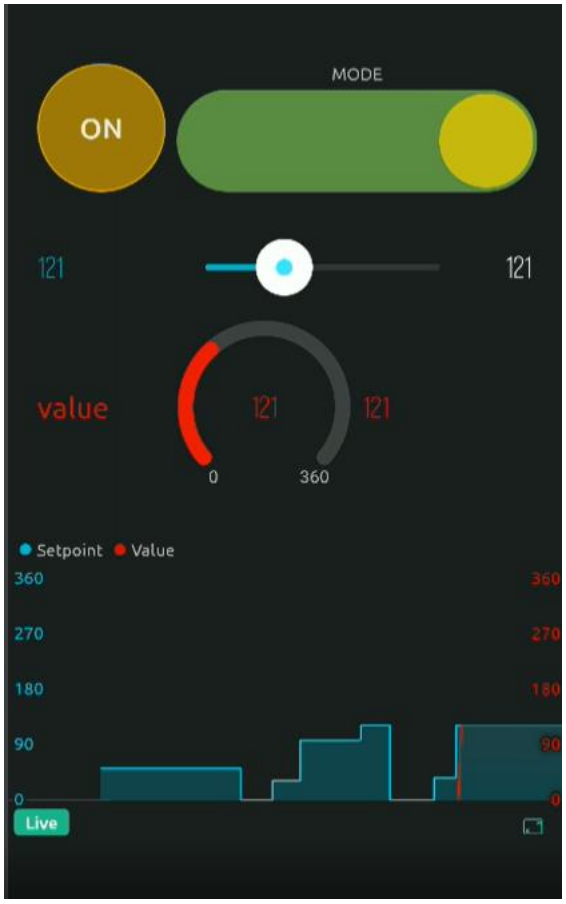
1. APP BLYNK

- Hình ảnh sau khi thiết kế giao diện điều khiển trên App



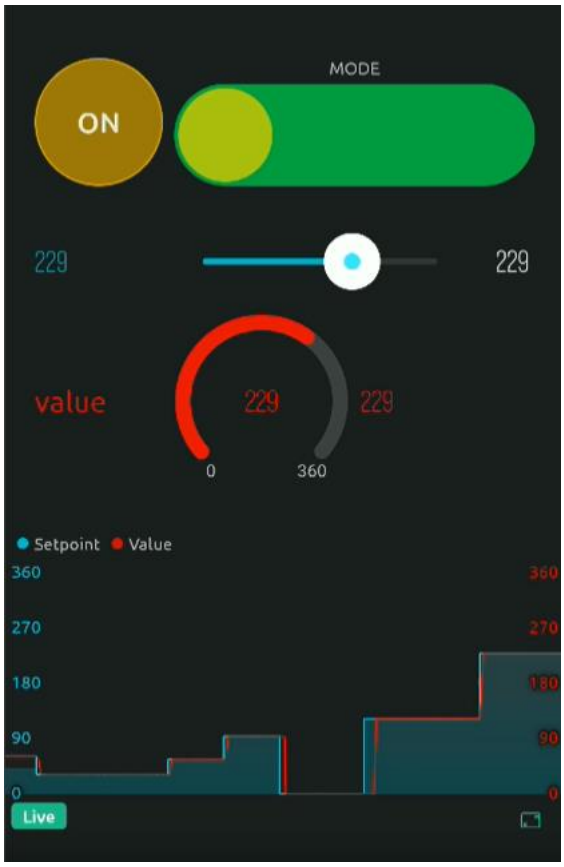
2. Khảo sát đáp ứng

- Đáp ứng vận tốc



Nhận xét: Đáp ứng vận tốc đảm bảo chất lượng điều khiển, thời gian xác lập nhanh, sai số điều khiển bằng 0 và không bị vọt lố. Thời gian đáp ứng khá nhanh tuy nhiên đồ thị không phải là đường cong. Điều này có thể lý giải do SERVER BLYNK không thể ghi và hiện dữ liệu với tần số quá nhanh, dù tần số lấy mẫu của hệ thống là 100ms nhưng 500ms mới gửi giá trị lên server để vẽ nên đồ thị nên đồ thị sẽ không phải là đường cong và đáp ứng hồi tiếp sẽ bị trễ. Chi tiết có thể xem video đính kèm.

- Đáp ứng vị trí:



Nhận xét: Đáp ứng vị trí là nhanh, sai số bằng 0 nhưng có độ trễ trong quá trình hồi tiếp vị trí lên app và dạng đồ thị không phải đường cong. Điều này có thể lý giải do SERVER BLYNK không thể ghi và hiện dữ liệu với tần số quá nhanh, dù tần số lấy mẫu của hệ thống là 100ms nhưng 500ms mới gửi giá trị lên server để vẽ nên đồ thị nên đồ thị sẽ bị trễ và dạng đồ thị không phải là đường cong. Chi tiết có thể xem video đính kèm.

3. Kết quả thực hiện

- Kết quả thực hiện được đính kèm trong link video sau: https://youtu.be/fyAJlLnCVB4

CHƯƠNG 5:
ĐÁNH GIÁ KẾT QUẢ THỰC HIỆN

Kết quả làm bài tập lớn của nhóm hầu như đã đáp ứng được các yêu cầu mà giảng viên đã đề ra

1. Các công việc nhóm đã làm được:

- Tìm hiểu về bộ điều khiển PID và phương pháp để điều khiển động cơ dựa trên nó.
- Ứng dụng được bộ điều khiển PID, thực hiện thi công mạch điều khiển động cơ đạt kết quả tốt : không có vọt lố, đáp ứng khá nhanh, sai số thấp
- Thiết kế và thi công mạch ra chân cho ESP 8266 cũng như kết nối phần cứng một cách hoàn chỉnh, tạo thành một khối hoàn chỉnh.
- Viết chương trình cho ESP8266 và xây dựng chương trình BLYNK trên smartphone để điều khiển vị trí và tốc độ động cơ DC bằng thuật toán PID. Người dùng hoàn toàn có quyền chọn điều khiển vị trí hoặc tốc độ động cơ bằng phần mềm mà không cần can thiệp phần cứng

2. Các công việc chưa làm được và hướng phát triển:

- Về thẩm mỹ, mạch thi công đường dây tuy khá gọn gàng nhưng nhìn chung sẽ tốt hơn nếu làm được mạch in.
- Đáp ứng động cơ là khá nhanh, tuy nhiên trong nhiều ứng dụng khác thì thời gian lấy mẫu vẫn hơi lâu, cần khắc phục, tìm được các chỉ số P, I, D tối ưu hơn nữa.
- Có thể thực hiện thêm bộ lọc Kalman để đáp ứng trở nên tốt hơn.
- Viết được app điều khiển, vẽ được đồ thị trên điện thoại chứ không cần dùng ứng dụng có sẵn như BLYNK.
- Nhận dạng mô hình động cơ và áp dụng các bộ điều khiển phi tuyến như hồi tiếp tuyến hoá, bộ điều khiển trượt hoặc sử dụng các bộ điều khiển thích nghi hoặc thông minh như Logic mờ (Fuzzy) để cải thiện chất lượng.

PHỤ LỤC

I. Datasheet mạch LM2596

https://datasheetspdf.com/pdf-file/1424873/CYStech/LM2596/1

II. Datasheet Kit thu phát wifi ESP 8266

https://www.electroschematics.com/wp-content/uploads/2015/02/esp8266-datasheet.pdf

III. Datasheet bộ điều khiển động cơ cầu H kép 7A/160W XY-160D

https://www.thanksbuyer.com/industrial-grade-2-channel-dc-motor-driver-module-7a-160w-cw-ccw-pwm-speed-control-l298-logic-xy-160d-63994

IV. Datasheet động cơ GA25-370

https://nshopvn.com/product/dong-co-dc-giam-toc-ga25-encoder/

V. Datasheet Modul hạ áp 12VDC

https://robocon.vn/detail/mdl202-module-ha-ap-12v-dc-xuong-33v-5v-pg2.html

VI. Full code lập trình

```
#define BLYNK_PRINT Serial
// Khai báo ID
#define BLYNK_TEMPLATE_ID "TMPL-a75PCCo"
#define BLYNK_DEVICE_NAME "Quickstart Template"
#define BLYNK_AUTH_TOKEN "HbkDrAvBEOHfdFhRziycOgy_VoHjJOpn"

#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>
#include <Ticker.h>
Ticker myTic;
char auth[] = BLYNK_AUTH_TOKEN;          // dia chi auuth cua Blynk
const char* ssid      = "VIETTEL";       // ten cua wifi
const char* pass      = "sonnguyen";     // mat khau cua wifi

//define cac chan của ESP vd chan D0 là GPIO16
static const uint8_t D0   = 16;
static const uint8_t D1   = 5;
```

```
static const uint8_t D2   = 4;
static const uint8_t D3   = 0;
static const uint8_t D4   = 2;
static const uint8_t D5   = 14;
static const uint8_t D6   = 12;
static const uint8_t D7   = 13;
static const uint8_t D8   = 15;

// define cac kenh va chan phan cung
#define Ts 0.1
#define res 495
#define IN1 D1
#define IN2 D2
#define ENCA D5
#define ENCB D6
#define ENA D3

// Khai bao cac bien
bool START=0,Mode=0;
// Gia tri cuoi cung sau khi tool PID
float SPEED_KP = 1.74;          //Kgh=3.5
float SPEED_KI = 0.52;          //~0.5
float SPEED_KD = 0.12;

float POSITION_KP = 0.382;      //Kgh~0.8
float POSITION_KI = 0.11;       //0.1
float POSITION_KD = 0.023;      //0.02


int encoderPos = 0;
int _speed, _speed1 ,_position, _position1;
```

```
int enable;
int cur_mode, pre_mode;
float error, pre_error=0, error_sum, d_error;
float control_signal;
int flag_speed =1, flag_position=0;
int setpoint;
int reset_position_signal ,reset_speed_signal;
int cnt = 0;
float my_duty_cycle;

BLYNK_WRITE(V0) // Nhân setpoint (Vận tốc và vị trí mong muốn) từ điện thoại
{
   setpoint = param.asInt();

}
BLYNK_WRITE(V2) // Nhận tín hiệu khởi động từ blynk
{
   START = param.asInt();

}
void Clear_all();
BLYNK_WRITE(V3) //Nhan tin hieu chon Mode (1: chay mode Speed 0: chay mode
Position)
{
  if(START) {
    Mode = param.asInt();
    if(Mode){
   flag_speed=1;
   flag_position=0;
   Serial.println("Speed mode ");
  }
```

```
 else {
   flag_speed=0;
   flag_position=1;
   Serial.println("Position mode ");
 }
 delay(100);
 Clear_all();
  }

}
float myPID(float KP,float KI,float KD,float current,int setpoint) ;
void Speed_motor();
void Position_motor();
void send_Serial();
// Hàm setup khởi tạo ban đầu cho chương trình.
void setup() {
  Serial.begin(115200);
  while (!Serial) {
    Serial.println(" COM is connected !! ");  //chờ cho cổng COM được kết nối
  }
  Blynk.begin(auth, ssid, pass);              // kết nối blynk điện thoại thông qua wifi
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(ENA, OUTPUT);
  pinMode(ENCA, INPUT);
  pinMode(ENCB, INPUT);
  attachInterrupt(14, ISR_encoder, RISING   ); // ngắt chân ENCA để update encoder
position
  myTic.attach(Ts, handle_interrupt);

}
```

```
// Hàm ngắt đọc encoder
ICACHE_RAM_ATTR void ISR_encoder() {
 int b=digitalRead(ENCB);
 if (b>0) encoderPos++;  // cạnh lên
 else encoderPos--;
}
// Hàm ngắt điều khiển theo tín hiệu từ blynk gửi về
void handle_interrupt() {
  if(START){
  if (flag_speed ) {
   Speed_motor();
  }
  else if(flag_position ) {
    Position_motor();
  }
  }
  else{
    Serial.println("Waiting for Start");
    Clear_all();

  }
}
// hàm main cho blynk chạy và kiểm tra serial port nhận dữ liệu gửi đến
void loop() {
  Blynk.run();
  if(Serial.available()){
    String teststr=Serial.readString();
    setpoint=teststr.toFloat();
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, LOW);
  }
```

```
 }
// Hàm dieu khiển tốc độ động cơ
void Speed_motor()
{
   if(setpoint>130) setpoint=130; // tốc độ tối đa của động cơ 130 vòng trên phút

   my_duty_cycle = myPID(SPEED_KP,SPEED_KI,SPEED_KD,_speed,setpoint);
   control_signal = my_duty_cycle*255/100.0;
   if (setpoint>0){
     digitalWrite(IN1, HIGH);
     digitalWrite(IN2, LOW);
   }
   else if (setpoint==0){
     digitalWrite(IN1, LOW);
     digitalWrite(IN2, LOW);
   }
   else {
     digitalWrite(IN1, LOW);
     digitalWrite(IN2, HIGH);
   }
   if(control_signal<0) control_signal=-control_signal;
   analogWrite(ENA, control_signal);


  _speed = (float)(encoderPos/(float)Ts/(float)res*60);
  encoderPos = 0;// reset coun
  send_Serial();
  Blynk.virtualWrite(V1,_speed1);
}
// hàm thực hiện chức năng
```

```
void Position_motor(){
   my_duty_cycle                                                    =
myPID(POSITION_KP,POSITION_KI,POSITION_KD,_position,setpoint);
   control_signal = my_duty_cycle*255/100;
   if (my_duty_cycle >= 0) {
     digitalWrite(IN1, HIGH);
     digitalWrite(IN2, LOW);
     analogWrite(ENA, control_signal);
   }
   else {
     digitalWrite(IN1, LOW);
     digitalWrite(IN2, HIGH);
     analogWrite(ENA, -control_signal);
   }

  _position = (float)(encoderPos)*360/(float)res;
  send_Serial();
  Blynk.virtualWrite(V1,_position1);



}
// Hàm gửi lên serial port
void send_Serial(){
  Serial.print(" Mode ");
  if (Mode) Serial.print(" Speed ");
  else Serial.print(" Position");
  Serial.print(" Setpoint: ");
  Serial.print(setpoint);
  if(Mode) {
    Serial.print(" (rpm) ");
  Serial.print(" Speed: ");
```

```
  Serial.print(_speed);
  Serial.print(" (rpm) ");
    Serial.print(" Kp ");
  Serial.print(SPEED_KP);
  Serial.print(" Ki ");
  Serial.print(SPEED_KI);
  Serial.print(" Kd ");
  Serial.print(SPEED_KD);
  }
  else {
  Serial.print(" (degree) ");
  Serial.print(" Position: ");
  Serial.print(_position);
  Serial.print(" (degree) ");
   Serial.print(" Kp ");
  Serial.print(POSITION_KP);
  Serial.print(" Ki ");
  Serial.print(POSITION_KI);
  Serial.print(" Kd ");
  Serial.print(POSITION_KD);
  }
  Serial.print(" Duty cycle ");
  Serial.print(my_duty_cycle);
  Serial.print(" Xung");
  Serial.println(control_signal);

}
// Hàm clear tất cả dữ liệu đưa về trạng thái ban đầu
void Clear_all(){
  _position=0;
  _speed=0;
```

```
  encoderPos=0;
  error_sum =0;
   setpoint = 0;
   digitalWrite(IN1, LOW);
   digitalWrite(IN2, LOW);
   Blynk.virtualWrite(V1,0);
}
// Hàm PID
float myPID(float KP,float KI,float KD,float current,int setpoint)
{
  error = setpoint - current;
  error_sum += error;
  d_error = (error - pre_error);
  float duty_cycle;
  duty_cycle = KP*error + KI*Ts*error_sum + KD*d_error/Ts ;
  pre_error = error ;
  if (duty_cycle > 100)
  duty_cycle = 99;
  else if(duty_cycle<-100)
  duty_cycle = -99;
  return(duty_cycle);
}
```