ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA ĐIỆN – ĐIỆN TỬ

BỘ MÔN TỰ ĐỘNG

.....o0o.....

BÀI TẬP LỚN

HỆ THỐNG ĐIỀU KHIỂN NHÚNG

GIẢNG VIÊN HƯỚNG DẪN: TS. Nguyễn Vĩnh Hảo

NHÓM 5

Tên sinh viên	MSSV
Dương Ngọc Hoàn	2010020
Lê Thế Hoan	2013203
Đỗ Hữu Toàn	2014769
Lê Hữu Phúc	2014163

I. PHẦN CỨNG:

1. Động cơ DC Servo:

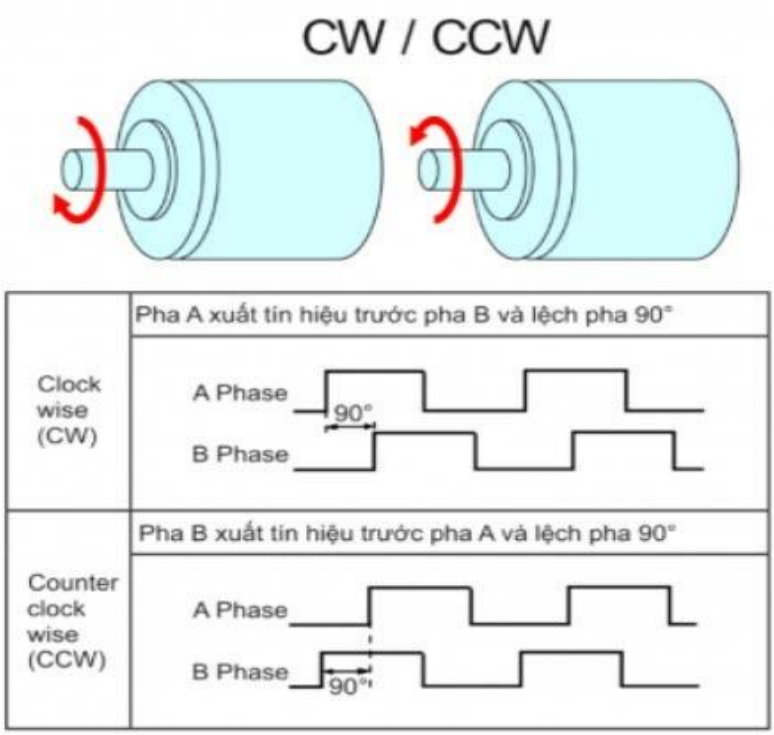
Trong khuôn khổ Đồ án này, phần tử chấp hành được sử dụng là động cơ DC Servo có giảm tốc. Động cơ có tích hợp đặc điểm kĩ thuật tiện ích là Encoder, hỗ trợ việc đọc chính xác tốc độ động cơ, hộp số cho phép tăng moment xoắn qua đó tăng khả năng mang tải của động cơ.

Hình 1. Động cơ DC JGB37-520 giảm tốc có encoder

Thông số kĩ thuật	
Động cơ	
Điện áp định mức	6 -24V/ 12VDC
Dòng tối đa	3A
Công suất	36W
Series	GB37
Tốc độ không tải (sau giảm tốc)	333 rpm
Tỉ lệ Hộp số	1:30
Encoder	
Điện áp nguồn	3.3 - 5VDC
Số xung/vòng (trước giảm tốc)	11
Số xung/vòng (sau giảm tốc)	330

Bảng 4.2. Thông số của động cơ JGB37-520

Dựa vào encoder ta có thể xác định được chiều quay của động cơ. Qui định xung A xuất tín hiệu trước xung B thì encoder đang quay cùng chiều kim đồng hồ CW(Clockwise), lúc này xung A đi trước và lệch pha 90 độ so với xung B. Ngược lại trục encoder quay ngược chiều kim đồng hồ CCW(Couter Clockwise), thì xung B xuất tín hiệu trước và lệch pha 90 độ xung A.

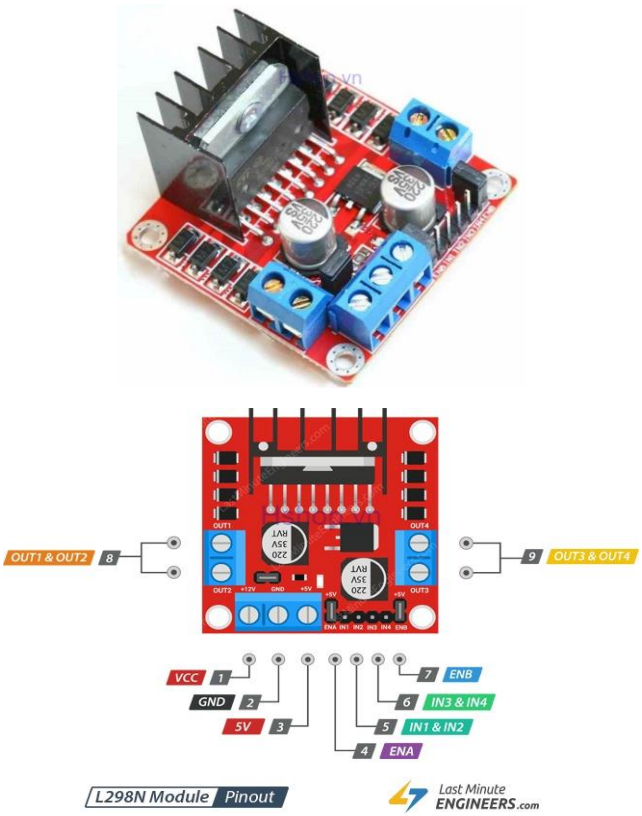


Hình 2. Xác định chiều quay động cơ

2. Module mạch cầu H:

Mạch điều khiển động cơ L298 DC Motor Driver có khả năng điều khiển 2 động cơ DC, dòng tối đa 2A mỗi động cơ, mạch tích hợp diod bảo vệ và IC nguồn 7805 giúp cấp nguồn 5VDC cho các module khác

Đồ án chọn dùng module L298 với nhiều ưu điểm như có thời gian giãn cách khi đổi chiều, công suất phù hợp với động cơ nhỏ, giá thành thấp, sử dụng đơn giản.

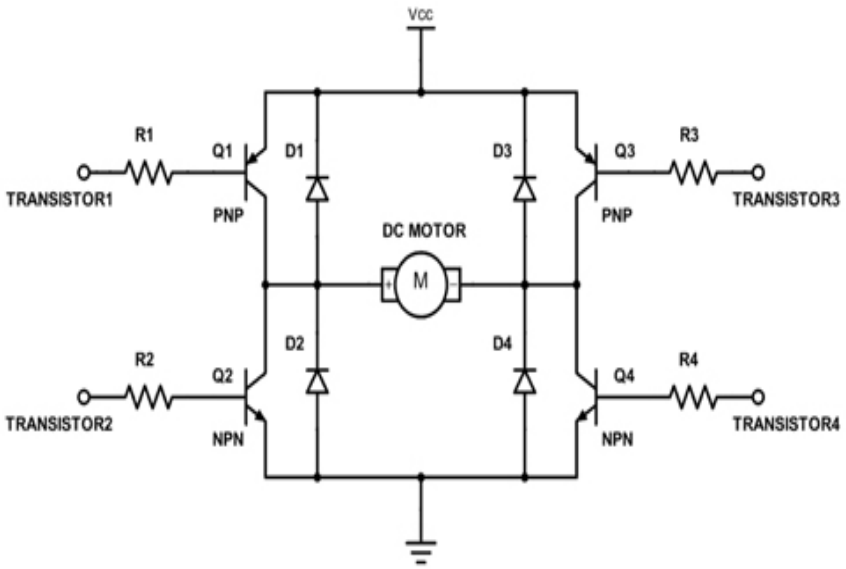


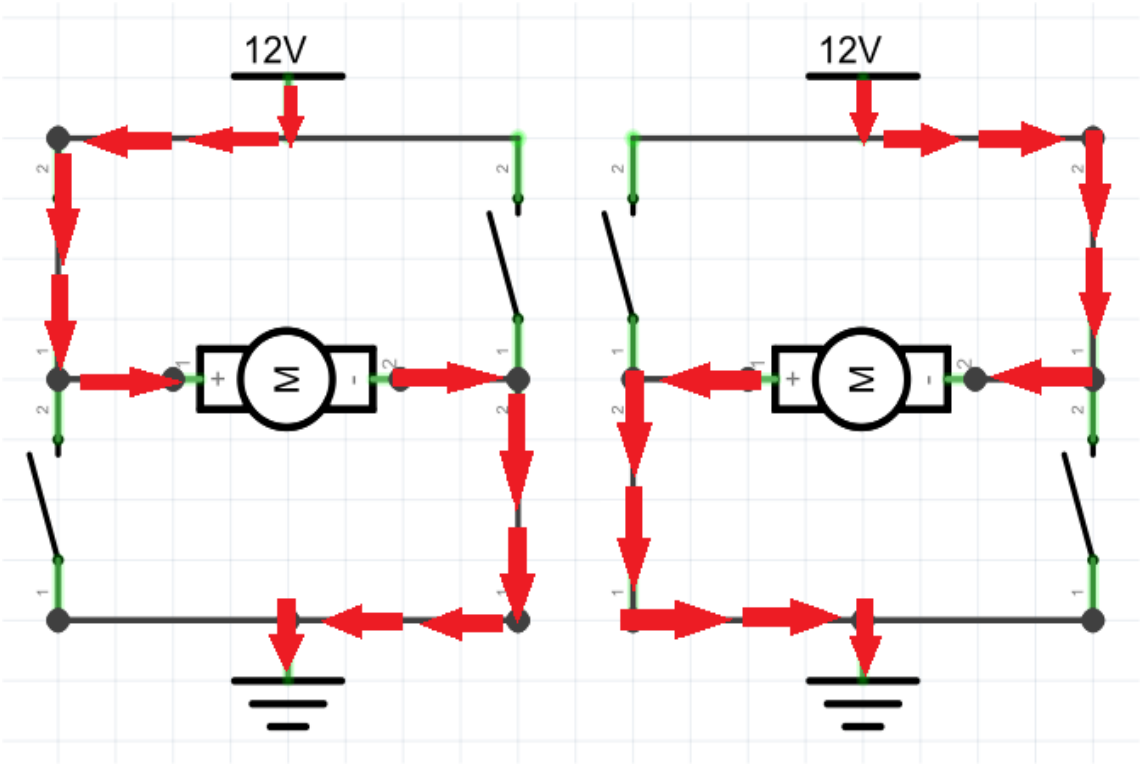
Hình 3. Module driver L298

+ IC chính: L298
+ Điện áp hoạt động: 5~30VDC
+ Công suất tối đa: 25W 1 cầu
+ Dòng tối đa cho mỗi cầu H là: 2A
+ Mức điện áp logic: Low -0.3V~1.5V, High: 2.3V~Vss
+ Kích thước: 43x43x27mm

Module hoạt động dựa trên nguyên lý mạch cầu H:

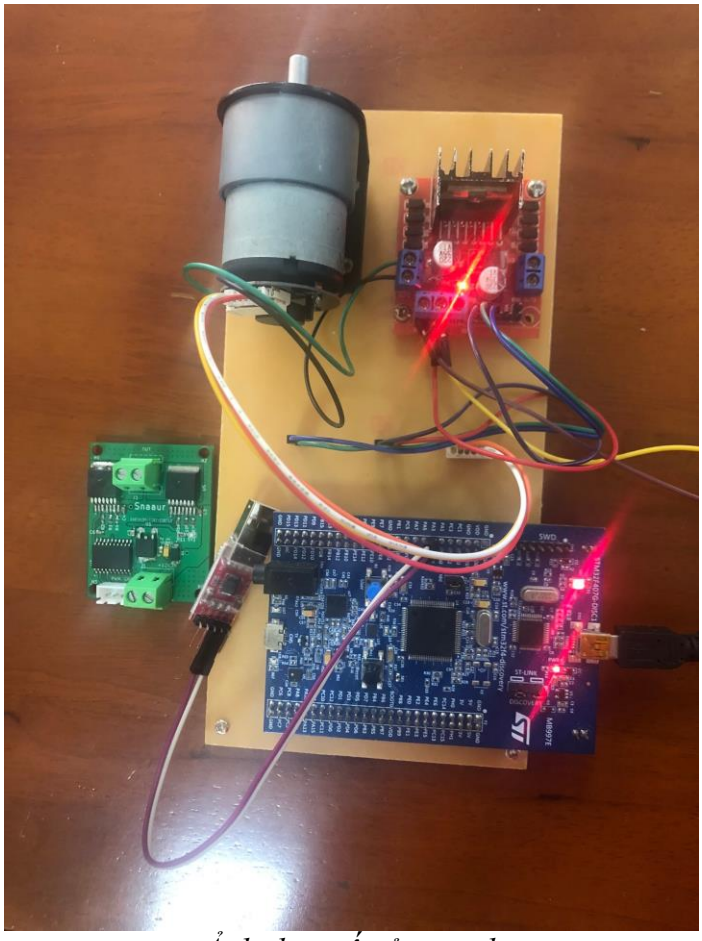
Xét một cách tổng quát, mạch cầu H là một mạch gồm 4 "công tắc" được mắc theo hình chữ H. Bằng cách điều khiển 4 "công tắc" này đóng mở, ta có thể điều khiển được dòng điện qua động cơ cũng như các thiết bị điện tương tự. Bốn "công tắc" này thường là Transistor BJT, MOSFET hay relay. Tùy vào yêu cầu điều khiển khác nhau mà người ta lựa chọn các loại "công tắc" khác nhau.





Nguyên lý mạch cầu H

Sản phẩm hoàn thiện:



Ảnh thực tế của mạch

II. LẬP TRÌNH NHÚNG VI ĐIỀU KHIỂN:

1. Giới thiệu vi điều khiển STMF407VG:

Để thực hiện được việc thiết kế bộ điều khiển PID điều khiển động cơ, ta sẽ dùng bộ xử lí trung tâm là STM32F4 Discovery với nhân chính là chip xử lí STM32F407VG; là dòng vi xử lí 32-bit lõi ARM Cortex-M4. Việc sử dụng board này sẽ giúp thúc đẩy khả năng của vi điều khiển làm việc với hiệu suất tốt nhất, nó cho phép người dùng dễ dàng phát triển các ứng dụng khác nhau. Kit bao gồm một mạch nạp ST-LINK/V2 để phục vụ việc nạp code và debug chương trình, cảm biến gia tốc ST-MEMS, một micro số, một audio DAC với trình điều khiển loa tích hợp lớp D, đèn LED, nút bấm và cổng micro-AB USB OTG

Thông số kĩ thuật	
RAM	192 KB
Flash memory	1 MB
Nguồn cấp qua cổng US	5VDC
Nguồn cấp ngoài	3VDC, 5VDC
Thạch anh	4 -26 MHz
Dao động RC	16 MHz
ADC	3 kênh - 12bits
DAC	2 kênh - 12bits
DMA	16 kênh
Timer	12 kênh -16bits, 2 kênh -32bits
I2C interface	3 kênh
USART/UART interface	4/ 2 kênh
SPI interface	3 kênh
CAN interface	2 kênh
USB interface	USB 2.0 full speed, high speed
Ethernet	10/100 Ethernet MAC có DMA, hỗ trợ chuẩn IEEE

Bảng thông số kỹ thuật STMF407VG

2. Cấu trúc chương trình nhúng:

Vi điều khiển đóng vai trò là bộ điều khiển trung tâm của hệ thống. Để có thể lập trình cho vi điều khiển ta sử dụng phần mềm hỗ trợ STM32 CubeIDE v1.4 viết bằng thư viện HAL (Hardware Abstraction Layer). Chương trình của vi điều khiển gồm các khối chức năng sau: Read Encoder, PID Algorithm, PWM Generator. Để thuận tiện, dễ dàng giao tiếp với máy tính, chương trình sẽ có thêm khối UART Communication.

2.1. Khối Read Encoder:

Khối này có chức năng đọc tín hiệu Encoder bằng mode đọc encoder được hỗ trợ sẵn trong chip sau đó sẽ chuyển thành dữ liệu vị trí trục quay với đơn vị độ (degree) và tốc độ động cơ theo đơn vị vòng/phút (rpm).

2.2. Khối PID:

Khối PID nhận tín hiệu đặt (Setpoint), các thông số bộ điều khiển từ giao diện trên máy tính thông qua khối UART Communication, đồng thời nhận tín hiệu hồi tiếp vị trí, tốc độ của khối Read Encoder, sau đó sẽ áp dụng giải thuật PID tính toán để tìm được tín hiệu điều khiển.

2.3. Khối PWM:

Khối PWM nhận tín hiệu điều khiển từ khối PID và tiến hành xuất xung PWM tương ứng với tín hiệu điều khiển nhận được.

Khi tín hiệu điều khiển không có hoặc bằng 0, khối tắt tín hiệu enable mạch lái. Khi tín hiệu khác 0, khối bật tín hiệu enable mạch lái.

Khi nhận tín hiệu điều khiển dương, khối chỉ xuất xung đến ngõ quay thuận của mạch lái, tắt xung ở ngõ quay ngược. Khi nhận tín hiệu điều khiển âm, khối thực hiện ngược lại.

2.4. Khối UART:

Khối này nhận tín hiệu đặt và thông số số bộ điều khiển từ máy tính, rồi gửi cho khối PID để tính toán tín hiệu điều khiển. Bên cạnh đó, khối UART sẽ lấy dữ kiệu từ khối đọc encoder rồi gửi lên máy tính để giao diện thực hiện vẽ đồ thị đáp ứng của động cơ theo thời gian thực, giúp ta có thể dễ dàng theo dõi, giám sát quá trình hoạt động của động cơ cũng như đánh giá chất lượng của bộ điều khiển.

Trong thực tế, khi truyền tải mỗi dữ liệu gồm một chuỗi các bit, các lỗi có thể phát sinh, bit 1 có thể biến thành bit 0 và ngược lại, có thể

lỗi một bit hoặc nhiều bit. Nguyên nhân của việc này có thể là do nhiễu, chuyển đổi nguồn điện, nguồn cung cấp điện kém, …

Việc nhận sai dữ liệu truyền nhận có thể làm cho hệ thống hoạt động sai, gây ra những hậu quả khó lường. Để tránh tình huống đó, ở đây em đã thêm các thông tin phụ vào bản tin chỉ nhằm mục đích giúp kiểm tra lỗi. Mã thừa sẽ được loại bỏ sau khi xác định xong độ chính xác của quá trình truyền. Phương pháp kiểm tra lỗi em dùng ở đây là phương pháp kiểm tra lỗi bằng thuật toán CRC (Cyclic Redundancy Check)

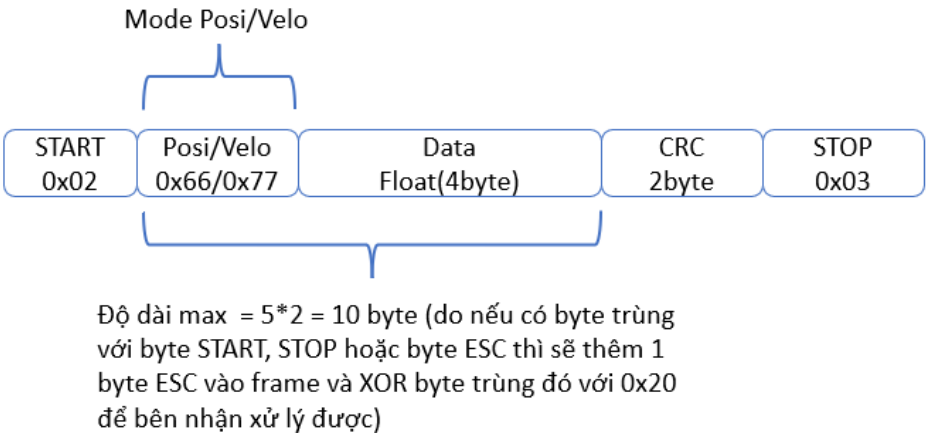
Thuật toán CRC:

Ý tưởng cơ bản của thuật toán CRC là coi dữ liệu được truyền là một số nhị phân rất dài các chữ số. Chia số này cho một số khác, phần số dư của phép chia này được nối với dữ liệu ban đầu dưới dạng dữ liệu kiểm tra.

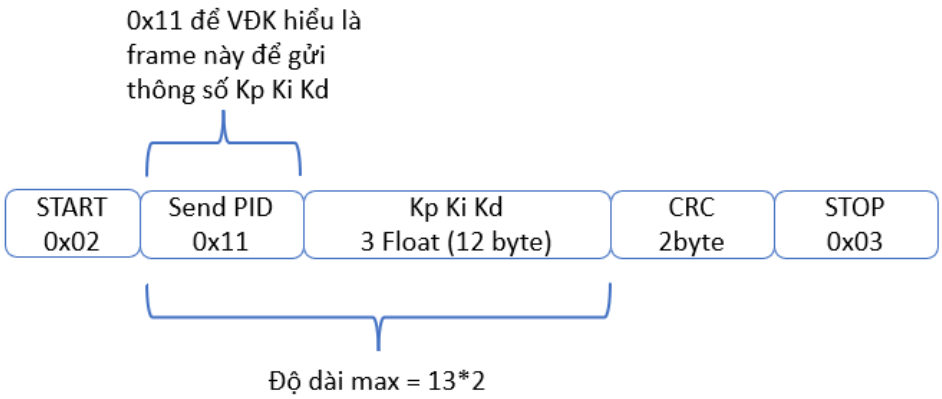
Thuật toán CRC sẽ tương tự với quy trình này: phép nhân và chia được sử dụng như là phép nhân và chia thông thường, trong khi đó phép cộng và phép trừ được thay bằng phép XOR bit. Do đó phép chia sẽ được thực hiện dễ dàng hơn vì không phải quan tâm đến cộng trừ có nhớ. Tại đây em không tự tính CRC mà dùng thư viện có sẵn để tính, công việc thực hiện chỉ là code tạo frame truyền – nhận và thêm 2 byte CRC tính từ thư viện đó vào.

Các frame truyền tự tạo như sau:

+ Frame truyền giá trị của vận tốc hiện tại hoặc vị trí hiện tại từ vi điều khiển lên GUI để vẽ đồ thị:

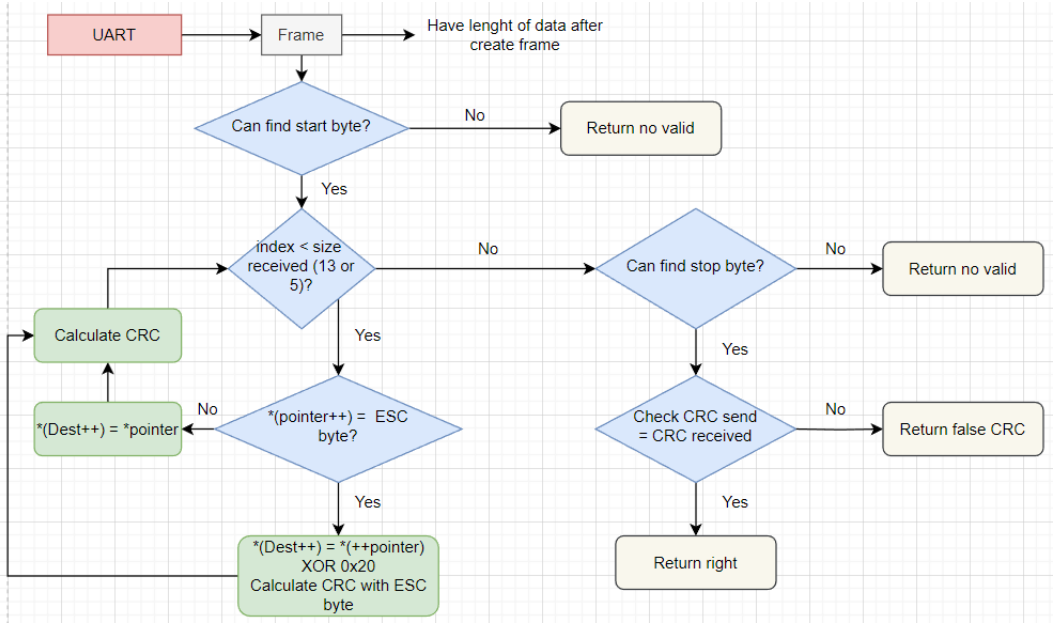


+ Frame truyền các thông số Kp Ki Kd xuống vi điều khiển:



+ Và một vài frame khác cũng tương tự với 2 frame trên các chức năng chọn PID1/PID2, gửi setpoint, run, reset.

Lưu đồ giải thuật nhận data từ vi điều khiển/Qt sau khi nhận được frame, với Dest là data thu được sau khi cắt ra từ frame, pointer là con trỏ tới địa chỉ byte đầu tiên của frame nhận được:



Lưu đồ giải thuật của hàm nhận frame và cắt lấy data

3. Cấu trúc chương trình nhúng:

Vi điều khiển đóng vai trò là bộ điều khiển trung tâm của hệ thống. Để có thể lập trình cho vi điều khiển ta sử dụng phần mềm hỗ trợ STM32 CubeIDE v1.4 viết bằng thư viện HAL (Hardware Abstraction Layer). Chương trình của vi điều khiển gồm các khối chức năng sau: Read Encoder, PID Algorithm, PWM Generator. Để thuận tiện, dễ dàng giao tiếp với máy tính, chương trình sẽ có thêm khối UART Communication.

2.5. Khối Read Encoder:

Khối này có chức năng đọc tín hiệu Encoder bằng mode đọc encoder được hỗ trợ sẵn trong chip sau đó sẽ chuyển thành dữ liệu vị trí trục quay với đơn vị độ (degree) và tốc độ động cơ theo đơn vị vòng/phút (rpm).

2.6. Khối PID:

Khối PID nhận tín hiệu đặt (Setpoint), các thông số bộ điều khiển từ giao diện trên máy tính thông qua khối UART Communication, đồng thời nhận tín hiệu hồi tiếp vị trí, tốc độ của khối Read Encoder, sau đó sẽ áp dụng giải thuật PID tính toán để tìm được

tín hiệu điều khiển.

2.7. Khối PWM:

Khối PWM nhận tín hiệu điều khiển từ khối PID và tiến hành xuất xung PWM tương ứng với tín hiệu điều khiển nhận được.

Khi tín hiệu điều khiển không có hoặc bằng 0, khối tắt tín hiệu enable mạch lái. Khi tín hiệu khác 0, khối bật tín hiệu enable mạch lái.

Khi nhận tín hiệu điều khiển dương, khối chỉ xuất xung đến ngõ quay thuận của mạch lái, tắt xung ở ngõ quay ngược. Khi nhận tín hiệu điều khiển âm, khối thực hiện ngược lại.

2.8. Khối UART:

Khối này nhận tín hiệu đặt và thông số số bộ điều khiển từ máy tính, rồi gửi cho khối PID để tính toán tín hiệu điều khiển. Bên cạnh đó, khối UART sẽ lấy dữ kiệu từ khối đọc encoder rồi gửi lên máy tính để giao diện thực hiện vẽ đồ thị đáp ứng của động cơ theo thời gian thực, giúp ta có thể dễ dàng theo dõi, giám sát quá trình hoạt động của động cơ cũng như đánh giá chất lượng của bộ điều khiển.

Trong thực tế, khi truyền tải mỗi dữ liệu gồm một chuỗi các bit, các lỗi có thể phát sinh, bit 1 có thể biến thành bit 0 và ngược lại, có thể lỗi một bit hoặc nhiều bit. Nguyên nhân của việc này có thể là do nhiễu, chuyển đổi nguồn điện, nguồn cung cấp điện kém, …

Việc nhận sai dữ liệu truyền nhận có thể làm cho hệ thống hoạt động sai, gây ra những hậu quả khó lường. Để tránh tình huống đó, ở đây em đã thêm các thông tin phụ vào bản tin chỉ nhằm mục đích giúp kiểm tra lỗi. Mã thừa sẽ được loại bỏ sau khi xác định xong độ chính xác của quá trình truyền. Phương pháp kiểm tra lỗi em dùng ở đây là phương pháp kiểm tra lỗi bằng thuật toán CRC (Cyclic Redundancy Check)

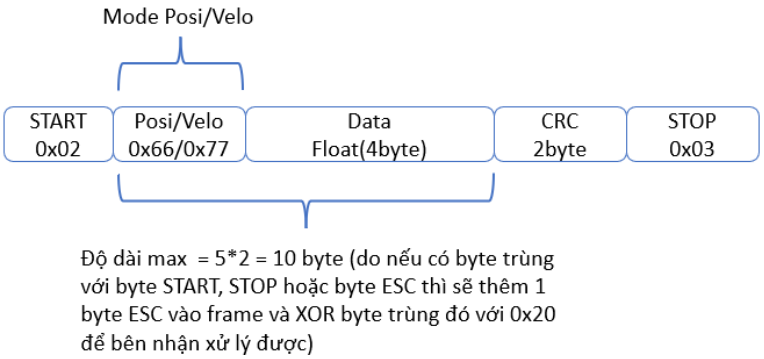
Thuật toán CRC:

Ý tưởng cơ bản của thuật toán CRC là coi dữ liệu được truyền là một số nhị phân rất dài các chữ số. Chia số này cho một số khác, phần số dư của phép chia này được nối với dữ liệu ban đầu dưới dạng dữ liệu kiểm tra.

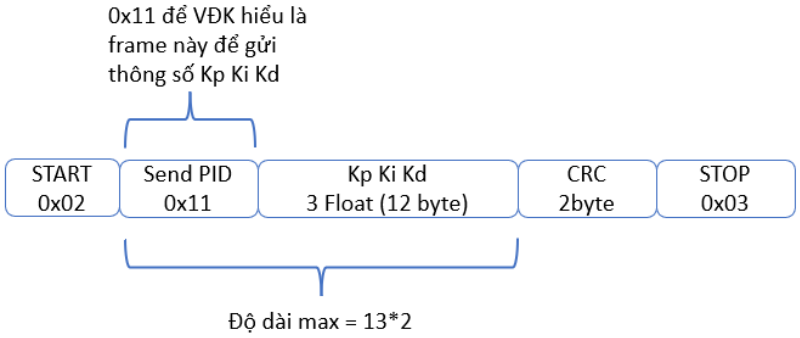
Thuật toán CRC sẽ tương tự với quy trình này: phép nhân và chia được sử dụng như là phép nhân và chia thông thường, trong khi đó phép cộng và phép trừ được thay bằng phép XOR bit. Do đó phép chia sẽ được thực hiện dễ dàng hơn vì không phải quan tâm đến cộng trừ có nhớ. Tại đây em không tự tính CRC mà dùng thư viện có sẵn để tính, công việc thực hiện chỉ là code tạo frame truyền – nhận và thêm 2 byte CRC tính từ thư viện đó vào.

Các frame truyền tự tạo như sau:

+ Frame truyền giá trị của vận tốc hiện tại hoặc vị trí hiện tại từ vi điều khiển lên GUI để vẽ đồ thị:

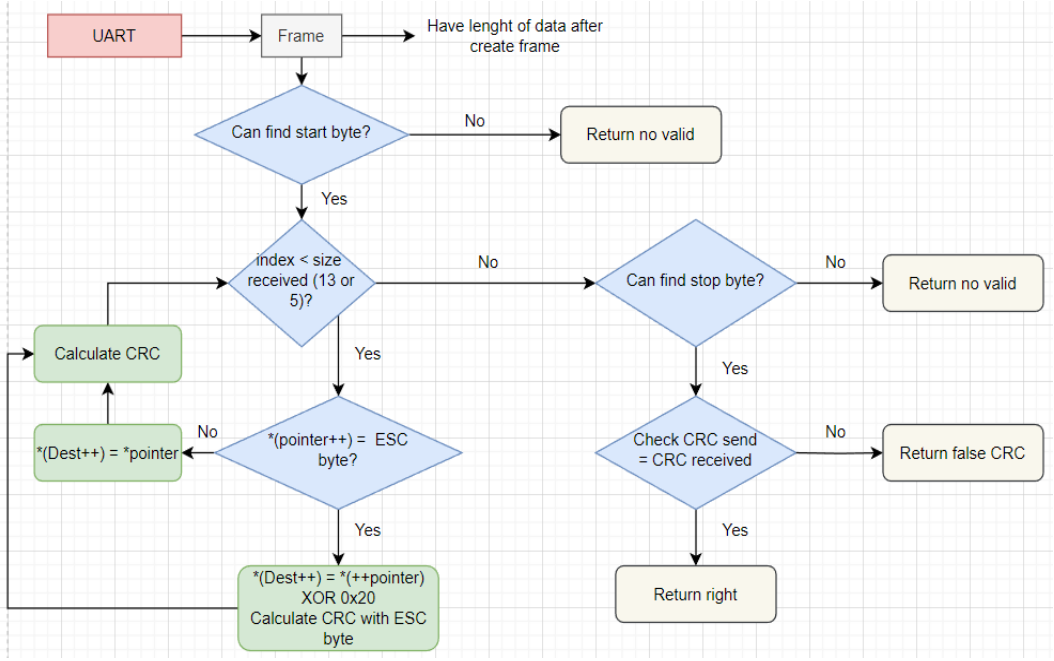


+ Frame truyền các thông số Kp Ki Kd xuống vi điều khiển:



+ Và một vài frame khác cũng tương tự với 2 frame trên các chức năng chọn PID1/PID2, gửi setpoint, run, reset.

Lưu đồ giải thuật nhận data từ vi điều khiển/Qt sau khi nhận được frame, với Dest là data thu được sau khi cắt ra từ frame, pointer là con trỏ tới địa chỉ byte đầu tiên của frame nhận được:



Lưu đồ giải thuật của hàm nhận frame và cắt lấy data

III. THIẾT KẾ GIAO DIỆN TRÊN MÁY TÍNH GIAO TIẾP NGƯỜI DÙNG:

1. Giới thiệu phần mềm Qt Creator:

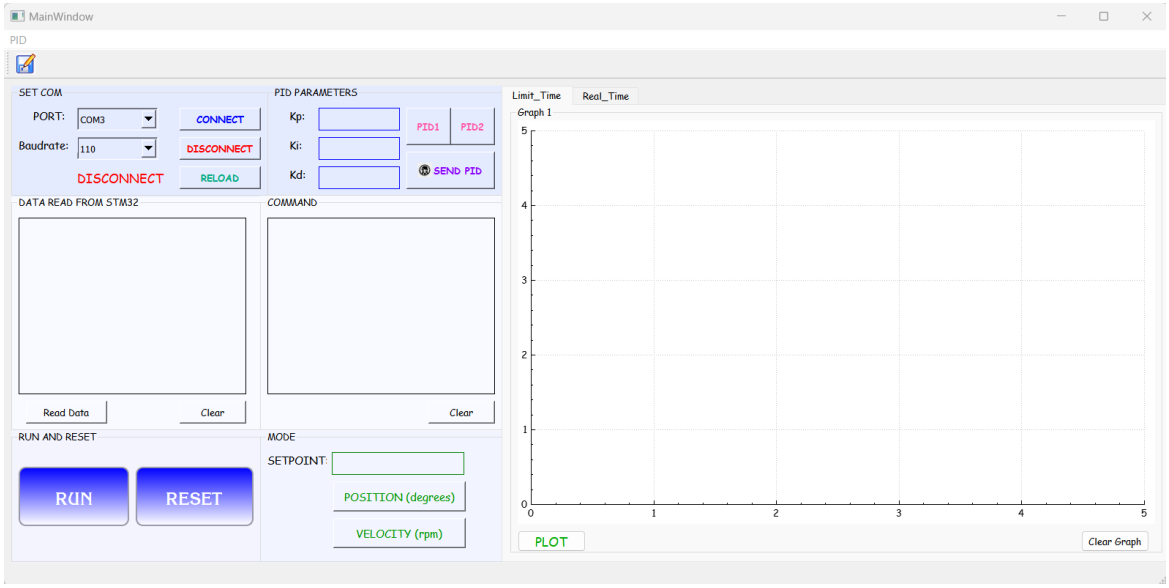
Qt là một framework đa nền tảng. Một số ứng dụng phổ biến được viết từ Qt có thể kể đến như KDE, Opera, Google Earth, và Skype. Qt lần đầu tiên được giới thiệu vào tháng 5 năm 1995. Qt có thể được dùng để phát triển ứng dụng mã nguồn mở lẫn các ứng dụng dành cho doanh nghiệp. Bộ công cụ phát triển Qt rất mạnh mẽ vì nó được cả một cộng đồng mã nguồn mở hỗ trợ. Có đến hàng ngàn các nhà phát triển mã nguồn mở sử dụng Qt trên toàn thế giới.

Qt được viết bằng C++ và được thiết kế để sử dụng trong C++. Tuy nhiên, hiện nay chúng ta đã có thể dùng thư viện này với nhiều ngôn ngữ khác như Java hay Python, …

Trên thực tế, Qt không phải một thư viện mà là 1 tập hợp các thư viện. Chúng rất rộng và thường thì người ta sử dụng thuật ngữ framework, nghĩa là 1 khối kiến trúc tập hợp cung cấp nhiều công cụ để việc lập trình của chúng ta trở nên hữu hiệu hơn.

Hình 4. Logo của phần mềm Qt Creator

2. Cấu trúc chương trình giao diện:



Hình 5. Giao diện tương tác trên máy tính

Giao diện có các khối tương tác như khối kết nối (Connect), hủy kết nối (Disconnect), chọn PID (PID1, PID2) và gửi thông số Kp Ki Kd xuống vi điều khiển (Send PID), khối để chọn chế độ điều khiển động cơ (Position hoặc

Velocity), khối hiển thị dữ liệu nhận về cũng như hiển thị các thông báo của hệ thống (Data read và Command), khối hiển thị đồ thị đáp ứng của hệ thống (Graph1 và Graph2) và khối System Control gồm Run và Reset

2.1. Khối Connect và Disconnect:

Có chức năng chọn cổng COM kết nối với USB UART đang kết nối giữa máy tính và vi điều khiển, có nút nhấn để cho phép kết nối cũng như ngắt kết nổi giữa 2 thiết bị. Khi chưa kết nối nút nhấn sẽ hiển thị chữ "CONNECT" màu chữ xanh dương và khi đã kết nối thành công thì sẽ hiển thị chữ "DISCONNECT" màu chữ đỏ để báo cho người sử dụng biết. Khi đang trong kết nối, bảng chọn cổng COM sẽ bị khóa lại. Khi đang không kết nối, các nút nhấn trong System Control cũng như nút gửi thông số bộ điều khiển khi nhấn sẽ hiện thông báo cảnh báo.

2.2. Khối System Control:

Khối này cho phép hệ thống chạy hoặc reset lại các thông số khi nhấn nút.

Khi nhấn nút "RUN", máy tính sẽ gửi xuống vi điều khiển một khung tin. Sau khi xác nhận được khung tin, vi điều khiển sẽ gửi trả một khung tin cho máy tính xác nhận động cơ đang chạy và sẽ bắt đầu khởi động bộ điều khiển và xuất xung PWM. Để có thể chạy được hệ thống phải có đầy đủ thông số PID và chọn mode điều khiển.

Khi nhấn nút "RESET", máy tính sẽ gửi xuống vi điều khiển một khung tin. Sau khi xác nhận được khung tin, vi điều khiển sẽ gửi trả một khung tin cho máy tính xác nhận đã ngừng lại động cơ, và sẽ ngắt bộ điều khiển PID và xuất xung PWM = 0 để dừng động cơ, đồng thời đưa biến đọc encoder về giá trị 0.

2.3. Khối PID Parameters và khối đặt setpoint:

Chứa các ô để nhập giá trị Kp Ki Kd gửi xuống vi điều khiển, PID1 là mode PID gốc thuần cơ bản, PID2 là mode PID được cải tiến từ kinh nghiệm (sẽ được nói tới ở phần phát triển thêm).

Các ô để nhập giá trị Kp Ki Kd, setpoint chỉ có thể nhập được số thực, đã lập trình các ô đấy không thể điền các ký tự hoặc chữ cái vào.

Sau khi kiểm tra thông số, nhấn nút "Send PID" để gửi thông số và mode chọn PID xuống vi điều khiển.

Khối đặt setpoint mục đích để đặt setpoint vị trí (position, tính theo độ) hoặc setpoint vận tốc (velocity, tính theo vòng/phút)

2.4. Khối Graph:

Gồm Graph1 và Graph2, Graph1 (limit_time) dùng để quan sát đồ thị đáp ứng, Graph2 (real_time) dùng để tune thông số liên tục theo thời gian mà không bị lag phần mềm Qt khi lượng dữ liệu thu được ngày càng lớn.

Khi có giá trị vị trí hoặc vận tốc được nhận trong lúc động cơ hoạt động, khối này sẽ được liên tục cập nhật đồ thị đáp ứng mức.

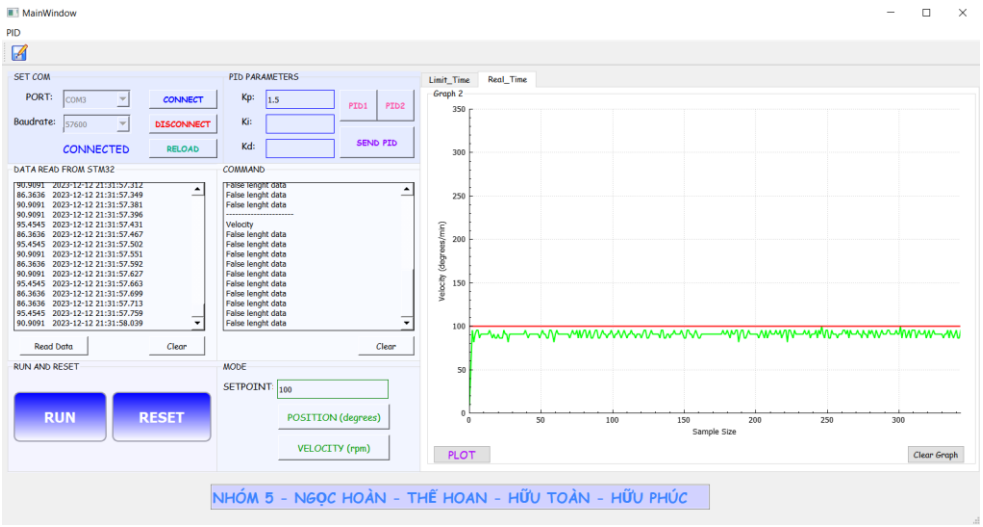
2.5. Khối data read và command:

Khối command có nhiệm vụ hiển thị các lệnh gửi từ GUI xuống vi điều khiển, có phản hồi nếu frame nhận được bị sai, khối data read dùng hiển thị giá trị vận tốc/vị trí hiện tại của động cơ (phía sau là thời gian tại vận tốc/vị trí đó).

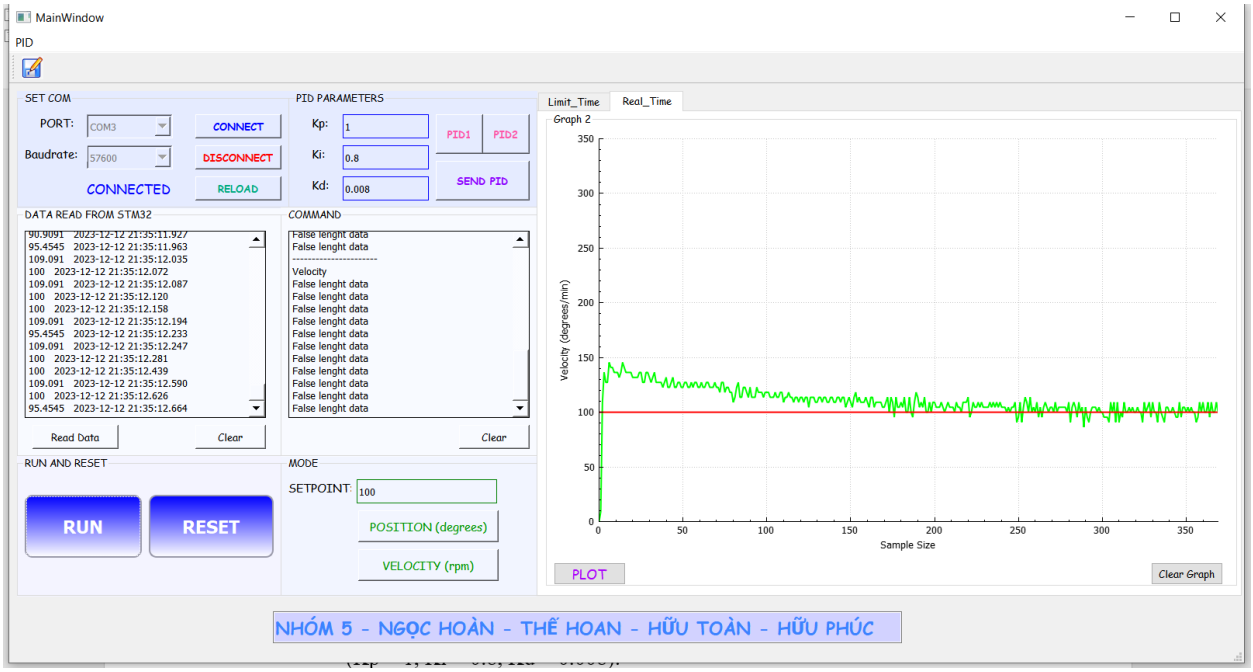
IV. ĐÁNH GIÁ CHẤT LƯỢNG:

1. Chạy thực nghiệm với vận tốc:

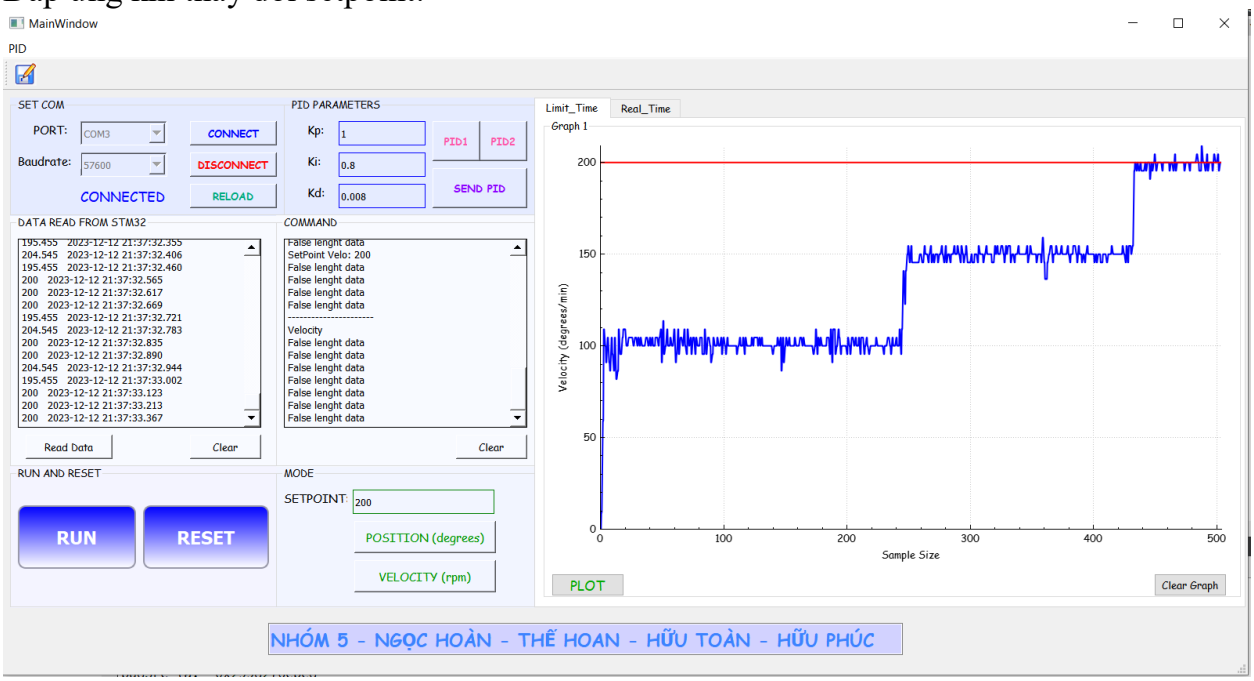
Đáp ứng vận tốc với Kp = 1.5, Ki = Kd = 0:



➔ Ta điều chỉnh thông số PID để đáp ứng tốt nhất có thể (Kp = 1, Ki = 0.8, Kd = 0.008):



Đáp ứng khi thay đổi setpoint:

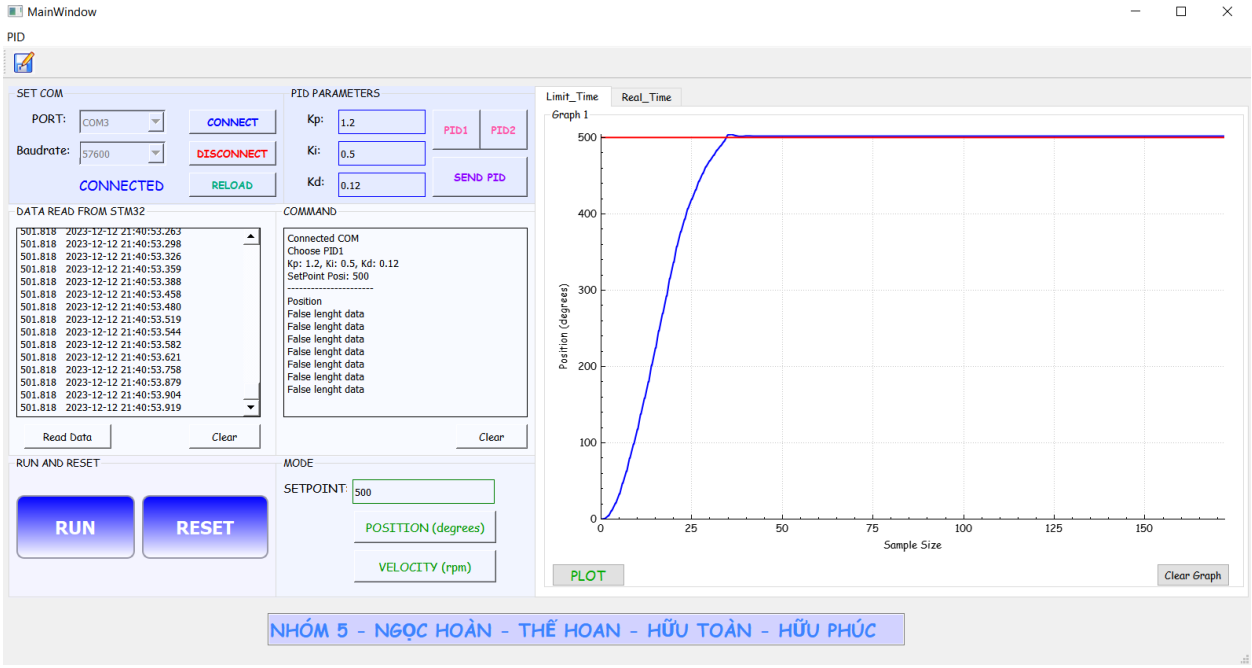


2. Chạy thực nghiệm với vị trí:

Ta tiến hành tune tìm thông số PID như sau:

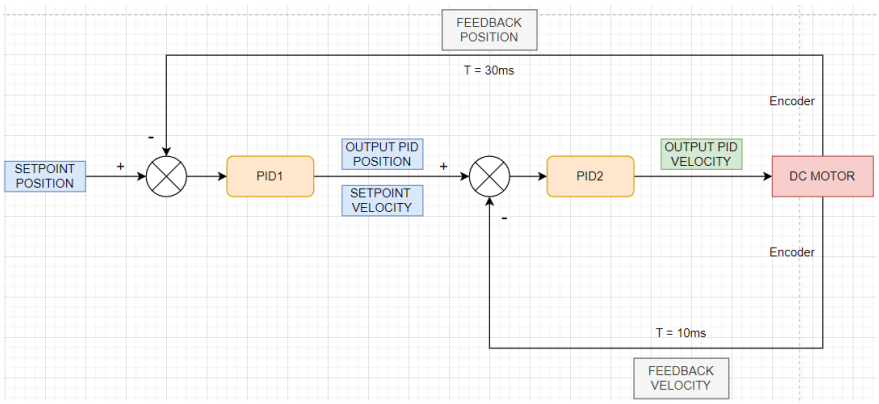
- Tune Kp đến khi có overshoot và undershoot để đủ năng lượng cung cấp cho hệ thống
- Tune Kd để giảm đi vọt lố do Kp tạo ra, giảm còn khoảng 5 - 10%
- Tune Ki để triệt tiêu sai số xác lập còn sót và giảm đi thời gian lên, không tăng quá lớn Ki để tránh độ vọt lố lại tiếp tục tăng

→ Điều chỉnh thông số PID để có đáp ứng tốt nhất có thể: (Kp = 1.2, Ki = 0.5, Kd = 0.12):



3. Kết luận:

Hiện tại, công thức tính toán PID được sử dụng ở đây là công thức cổ điển sơ khai nhất của bộ điều khiển PID số, nên khả năng điều khiển sẽ còn nhiều mặt hạn chế, không thể đạt được kết quả cao. Vì thế, thuật toán PID ngày càng phát triển tạo ra những bản thể khác để phục vụ tốt hơn thiết kế chuyên môn hóa hơn cho các đối tượng điều khiển khác nhau. Ở đây, chúng em có nêu ra một bộ PID phát triển khác như bộ PID 2 vòng sau có sơ đồ như sau:



Sơ đồ bộ PID cải tiến

Ý nghĩa sơ đồ: Khi điều khiển vị trí của động cơ thì ở lần đọc encoder đầu tiên sẽ tính error theo setpoint của vị trí và vị trí hiện tại từ đó tính ra output PID1 của vị trí, lấy output đó làm setpoint của vận tốc và tính ra output PID2 của vận tốc rồi mới output PID2 đó vào động cơ. Ở các lần tính sau thì cứ liên tục 3 lần tính output PID2 của vận tốc thì mới tính lại một lần output PID1 của vị trí và cứ thế lặp lại.

Vòng PID1 giúp điều chỉnh vị trí đến gần với setpoint, trong khi vòng PID2 giúp điều chỉnh vận tốc để đạt được sự ổn định và tránh hiện tượng dao động.

→ Việc tính toán PID 2 vòng giúp cải thiện hiệu suất và đáp ứng của hệ thống điều khiển động cơ DC-motor, đồng thời giảm thiểu sai số và tăng tính ổn định, ngoài ra PID 2 vòng có dễ dàng xác định thông số Kp Ki Kd hơn so với PID truyền thống.

Cho chạy thực nghiệm: