Mục lục

[**A.** **Cơ sở lý thuyết** 2](#_Toc376235146)

[I. Điều khiển PID rời rạc 2](#_Toc376235147)

[II. Điều khiển tốc độ động cơ DC 3](#_Toc376235148)

[III. Mạch động lực: Mạch cầu H 3](#_Toc376235149)

[**B.** **Giới thiệu thiết bị** 5](#_Toc376235150)

[I. Mạch cầu H 5](#_Toc376235151)

[II. Động cơ 7](#_Toc376235152)

[III. Mạch vi điều khiển 8](#_Toc376235153)

[IV. Lập trình vi điều khiển 8](#_Toc376235154)

[1. Đọc tốc độ từ encoder 9](#_Toc376235155)

[2. Gửi tốc độ đọc được về PC 10](#_Toc376235156)

[3. Thực hiện khâu PID 10](#_Toc376235157)

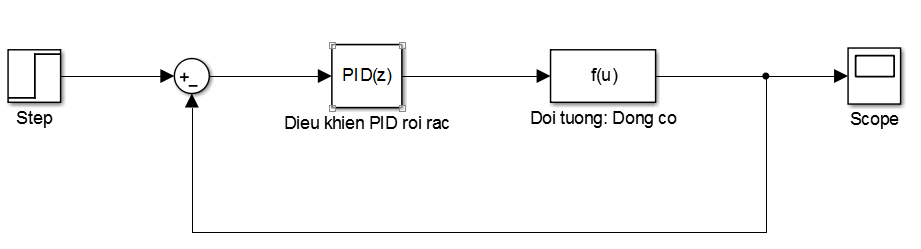
[4. Điều chỉnh xung PWM 11](#_Toc376235158)

[5. Nhận lệnh từ máy tính 11](#_Toc376235159)

[V. Giao diện trên máy tính 13](#_Toc376235160)

[**C.** **Kết quả thu được** 14](#_Toc376235161)

1. **Cơ sở lý thuyết**
2. Điều khiển PID rời rạc



Hình A 1.1: Sơ đồ hệ thống dùng PID rời rạc

Để dễ lập trình điều khiển PID cho vi điều khiển, ta sử dụng dạng PID rời rạc.

Bộ điều khiển PID rời rạc có hàm truyền như sau:

(1)

Sắp xếp lại, ta có:

(2)

Với:

Ta được:

(3)

Chuyển về miền thời gian:

Từ phương trình này ta có thể lập trình ứng dụng cho vi điều khiển để làm bộ điều khiển PID rời rạc, điều khiển tốc độ động cơ một chiều.

* Tuning: Có nhiều biện pháp tuning bộ PID, như phương pháp Ziegler – Nichols, nhưng ở đây do ta không biết rõ hàm truyền của đối tượng cũng như tìm nó nên ta sẽ thay thế các hệ số Ki, Kd, Kp để tìm ra hệ số đạt yêu cầu.

1. Điều khiển tốc độ động cơ DC

Mô hình của động cơ DC có thể được giới thiệu đơn giản như sau:



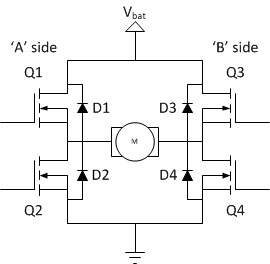
Hình A 2.1: Sơ đồ tương đương của động cơ

Động cơ DC là thiết bị chuyển hóa năng lượng điện sang năng lượng cơ học, và có thể chuyển từ năng lượng cơ học sang năng lượng điện nếu làm việc như một máy phát.

Mômen xoay của động cơ (Tm) tỉ lệ thuận với dòng điện (I) qua động cơ, nhưng ở đây ta chỉ quan tâm đến việc điều khiển tốc độ của động cơ bằng điện áp một chiều.

Tốc độ của động cơ DC (s) được điều khiển bằng đưa điện áp một chiều (Vg) vào hai cực của động cơ với công thức liên kết chúng:

1. Mạch động lực: Mạch cầu H



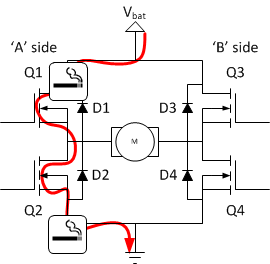
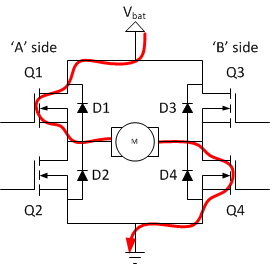
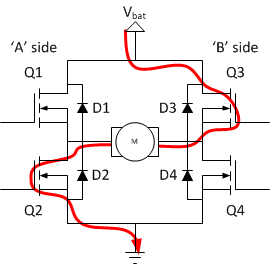
Hình A 3.1: Sơ đồ mạch cầu H

Mạch cầu H dùng để điều chỉnh điện áp một chiều, ứng dụng vào điều khiển động cơ.

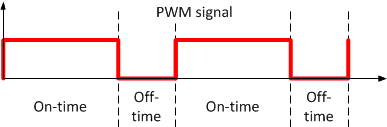
Mạch gồm có 4 linh kiện bán dẫn ở hai phía, thường là sử dụng FET, BJT hay IGBT

Các trạng thái của mạch:

|  |  |
| --- | --- |
| Thuận | Q1 và Q4 ON, Q2 và Q3 OFF |
| Nghịch | Q2 và Q3 ON, Q1 và Q4 OFF |
| Hãm nhanh | Q1 và Q3 ON hoặc Q2 và Q4 ON |
| Trùng dẫn | Đây là khi các linh kiện bán dẫn về một phía của mạch cầu H cùng hoạt động như Q1 và Q2 cùng ON hoặc Q3 và Q4 cùng on. Trạng thái này sẽ làm ngắn mạch và hư linh kiện bán dẫn, thêm nhiều khả năng làm hư vi điều khiển nếu không có mạch cách ly. Nên tránh trường hợp này. |



Để điều chỉnh điện áp, ta dùng vi điều khiển xuất sung PWM để điều khiển hoạt động của các linh kiện bán dẫn từ đó làm thay đổi điện áp theo mong muốn. Điều khiển được điện áp thì sẽ điều khiển được tốc độ của động cơ.





1. **Giới thiệu thiết bị**
2. Bộ nguồn

Bộ nguồn ATX máy tính Arrow 500W

* Hiệu điện thế vào: 230VAC-50Hz
* Hiệu điệu thế ra:12VDC,5VDC,3.3VDC

1. Mạch cầu H

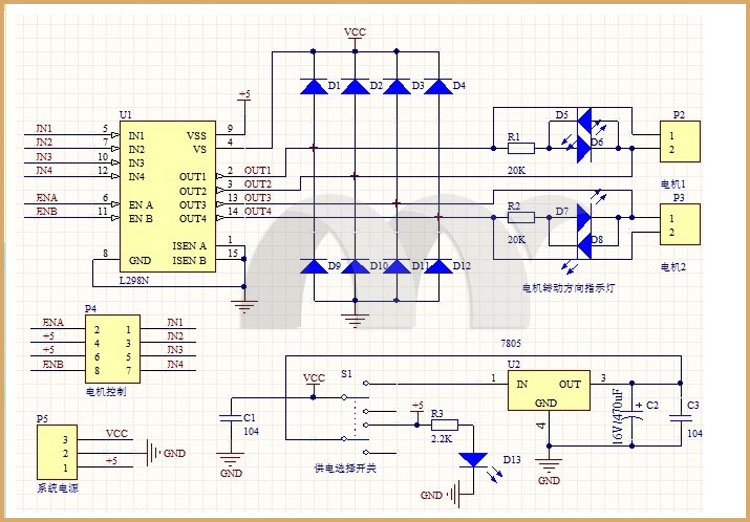


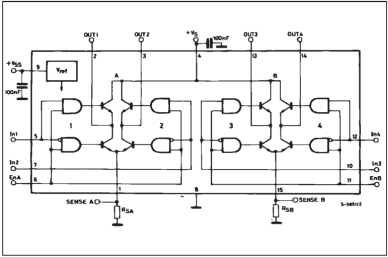
Sử dụng IC tích hợp mạch cầu H L298M gồm có 2 mạch cầu H, điều khiển với điện áp tối đa 46V, dòng tải 4A.

Để điều khiển mạch cầu H, sử dụng 2 chân: INA và INB.

Chân INA: Điều khiển xung PWM của động cơ khi động cơ quay theo chiều thuận

Chân INB: Điều khiển xung PWM của động cơ khi động cơ quay theo chiều nghịch





Hình B 1.2 Sơ đồ chân của IC L298M

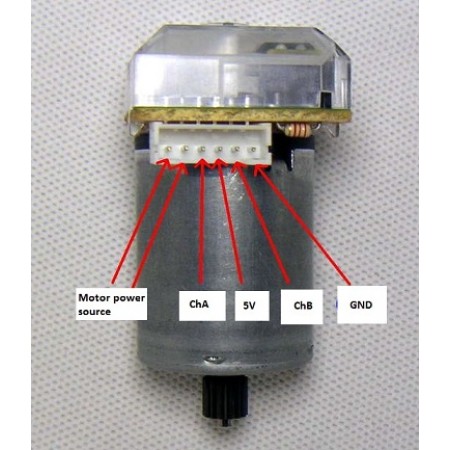
Ngoài ra các chân:

Chân VCC-GND: Cấp áp điều khiển động cơ 12VDC và 5VDC cấp nguồn cho mạch lái sau khi qua IC chuyển đổi điện áp và ổn áp 7805

Chân OutA-OutB: Xuất điện áp điều khiển ra động cơ

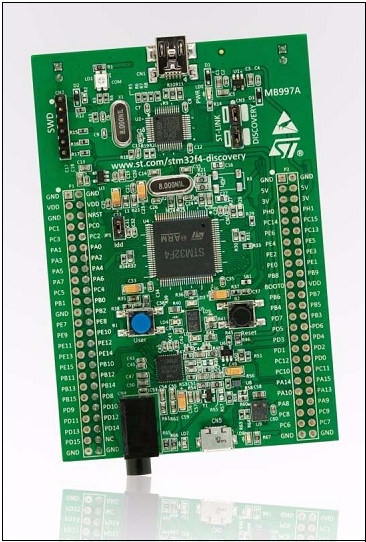
1. Động cơ

Động cơ DC Mitsumi M25N-2



* **Motor with Quad encoder which also gives direction of movement**
* **Motor Operating Voltages: 10V-31VDC**
* **Encoder Voltage: 5VDC**
* **Load current: 3A max**
* **No load Speed: 11,000 RPM**
* **Encoder 334 Counts per revolution**
* **14 tooth 6.2 mm diameter gear on output shaft**
* **Shaft Dia: 2 mm -0.2mm**
* **Motor Dia: 25.5mm**
* **Motor Length: 55mm including encoder and housing**
* **Stall Torque: 30 mNm**
* **Continuous torque: 21 mNm**

1. Mạch vi điều khiển

STM32F4 Discovery Kit được sử dụng để lập trình thành bộ điều khiển PID.

Sử dụng chức năng đọc Encoder để đọc tốc độ về, xử lý PID và xuất xung PWM cho mạch lái điều khiển tốc độ động cơ.

* Lập trình:

STM32F4 Discovery sử dụng ngôn ngữ C và dùng chương trình Keil C MDK-ARM để lập trình.

1. Lập trình vi điều khiển

Lập trình PID theo sơ đồ sau:

Bắt đầu

Khởi động Kit

Đọc encoder đo tốc độ

Khâu PID

Điều chỉnh xung PWM

Gửi tốc độ đọc được tới PC

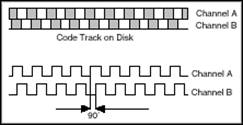
Nhận lệnh điều khiển từ PC

Hình B 4.1 Sơ đồ tư duy

1. Đọc tốc độ từ encoder

Encoder tích hợp của động cơ có 2 kênh A và B, chạy lệch pha nhau 90o để người dùng có thể đọc được động cơ đang quay thuận hay nghịch. Trong kit STM32F4 Discovery, vi điều khiển có tích hợp sẵn module đọc tín hiệu gửi từ encoder về để xử lý.

Encoder được đọc như sau:



* Khi phát hiện cạnh lên của kênh A, vi điều khiển đọc ngõ vào của kênh B, nếu là mức 0 thì cho counter tăng 1 và ngược lại nếu là mức 1 cho counter giảm 1.
* Khi phát hiện cạnh lên của kênh B, vi điều khiển đọc ngõ vào của kênh A, nếu là mức 0 thì counter giảm 1 và mức 1 thì counter tăng 1.
* Tương tự cho các cạnh xuống của 2 kênh A và B. Như vậy hết 1 chu kì của kênh A và B thì vi điều khiển đọc 4 lần encoder.

Theo thông số của nhà sản xuất thì encoder quay 334 xung / vòng cho một kênh, như vậy hết một vòng, vi điều khiển đọc từ encoder 334 x 4 = 1336 xung đếm cho counter của vi điều khiển.

* Code C lập trình đọc encoder:

|  |
| --- |
| //Encoder section  temp2 = temp1;  temp1 = EncoderValue();  diff = temp1 - temp2;    if (diff> 10000) diff=(temp1-0xFFFF)+(temp2);//overflow while counting down  if (diff<-10000) diff=(0xFFFF-temp2)+(temp1);//overflow while counting up    speed\_fb = diff\*1000.0\*60.0/1336.0; // RPM = (pulse/1336/0.001s)\*60 // encoder 334 pulses per round\*4= 1336  //roundC = (float)EncoderValue()/1336; |

Ta phải so sánh encoder với 10000 hay -10000 vì trường hợp counter của vi điều khiển bị tràn.

Counter của vi điều khiển là counter 16bit, tương đương đọc được từ 0 đến 65535. Khi bị tràn, counter nhảy từ 65535 xuống đến 0 và làm cho việc tính tốc độ của động cơ (temp1 – temp2) bị sai sót.

Để giải quyết trường hợp đó, ta chọn một mức max min để so sánh cho counter của vi điều khiển, nếu vượt quá tức là counter đã bị tràn và sẽ được xử lý như trong code: trừ đi 0xFFFF (65535).

Sau đó, nhân số counter đếm được với chu kì lấy mẫu và đổi ra vận tốc rpm.

1. Gửi tốc độ đọc được về PC

Sử dụng giao tiếp UART gửi tốc độ đọc được về PC qua cổng COM để hiển thị lên màn hình giao diện trên máy tính.

STM32F4 Discovery kit có hỗ trợ DMA để việc gửi thông tin vào thanh ghi truyền đi của UART nhanh và tiện hơn.

Dử liệu được truyền với định dạng như sau:

0xFF [Channel] [Data0] [Data1] [Data2] [Data3] 0xFE

0xFF: Báo hiệu cho phía nhận bắt đầu gửi dữ liệu.

0xFE: Báo hiệu cho phía nhận kết thúc gửi dữ liệu.

[Channel]: Kênh dữ liệu được gửi

[Datax]: Dữ liệu tốc đố được truyền đi, được tách ra từ định dạng real sang 4 ô nhớ để dễ gửi đi.

void SENDDATA(float Value, uint8\_t\* TxAddr, uint8\_t channel)

{

uint8\_t\* Data = (uint8\_t\*)&Value;

TxAddr[0] = 0xFF;

switch (channel) {

case 0: TxAddr[1] = 0xF0; break;

case 1: TxAddr[1] = 0xF1; break;

case 2: TxAddr[1] = 0xF2; break;

case 3: TxAddr[1] = 0xF3; break;

case 4: TxAddr[1] = 0xF4; break;

case 5: TxAddr[1] = 0xF5; break;

case 6: TxAddr[1] = 0xF6; break;

}

TxAddr[2] = Data[0];

TxAddr[3] = Data[1];

TxAddr[4] = Data[2];

TxAddr[5] = Data[3];

TxAddr[6] = 0xFE;

DMA\_TX();

}

1. Thực hiện khâu PID

Đây là phần chức năng chính của vi điều khiển, sử dụng công thức trong cơ sở lý thuyết, chương trình của PID như sau:

|  |
| --- |
| P\_part = Kp\* (e0 - e1);  I\_part = Ki\*T/2\* (e0 + e1);  D\_part = Kd/T\* (e0 - 2\*e1 + e2);    e2 = e1;  e1 = e0;  e0 = (speed\_sp - speed\_fb)/ (2\*max\_speed);  if (e0<-1) {e0=-1;}  if (e0>1) {e0= 1;}    U1 = U0;  U0 = U0\_100 / 100;  U0 = U1 + P\_part + I\_part + D\_part; |

Đây là công thức tính u[k] theo trên công thức ban đầu, từ u[k] biến đổi ra chu kỳ on của PWM để gửi vào mạch cầu H và điều khiển tốc độ động cơ.

Ngoài ra, còn các dòng lệnh bảo hòa, để ngăn cho u[k] tiến tới vô cùng.

*if (speed\_fb\*U0<0) U0=0;*

*if (U0<-1) U0=-1;*

*if (U0>1) U0=1;*.

1. Điều chỉnh xung PWM

|  |
| --- |
| if (U0>0) //Forward  {  duty = (uint16\_t)(U0 \*16800);  //Reset PD14  TIM\_SetCompare3(TIM4,0);  GPIO\_ResetBits(GPIOD,GPIO\_Pin\_14);  //TIM\_OC3PreloadConfig(TIM4, TIM\_OCPreload\_Disable);  //set PWM PD12  TIM\_SetCompare1(TIM4,duty);  }    if (U0<0) //Reverse  {  duty = (uint16\_t)(-U0 \*16800);  //Reset PD12  TIM\_SetCompare1(TIM4,0);  GPIO\_ResetBits(GPIOD,GPIO\_Pin\_12);  //TIM\_OC1PreloadConfig(TIM4, TIM\_OCPreload\_Disable);  //set PWM PD14  TIM\_SetCompare3(TIM4,duty);  } |

Xung PWM xuất ra sử dụng chức năng so sánh timer trong vi điều khiển, để đổi chu kì xung PWM, ta thay đổi giá trị so sánh timer bằng lệnh TIM\_SetCompare2(TIM4, Duty) ở mỗi cuối chu kì PWM.

Với 16800 là giá trị chu kì của xung PWM 5kHz

1. Nhận lệnh từ máy tính

Nhận lệnh với máy tính thông qua giao tiếp UART.

Khi nhận được tín hiệu truyền từ PC tới vi điều khiển ở cổng UART, sẽ có 1 chương trình interupt được thực hiện.

Trong chương trình interupt, vi điều khiển kiểm tra cú pháp của dữ liệu được gửi xuống với dạng sau:

void USART1\_IRQHandler(void)

{

if(USART\_GetITStatus(USART1, USART\_IT\_RXNE) != RESET)

{

/\* Read 28 byte from the receive data register \*/

temp = USART\_ReceiveData(USART1);

rxbuffer[rxindex] = temp;

rxindex++;

if (rxindex == rxsize) rxindex=0;

for (i=0;i<=rxsize-6;i++)

{

if ((rxbuffer[i] == 0xFF) && (rxbuffer[i + 6] == 0xFE))

{

memcpy(&rx\_data,&rxbuffer[i+2],4);

switch (rxbuffer[i+1])

{

case 0xE0:Kp = rx\_data;break;

case 0xE1:Ki = rx\_data;break;

case 0xE2:Kd = rx\_data;break;

case 0xE3:speed\_sp = rx\_data;break;

}

}

}

}

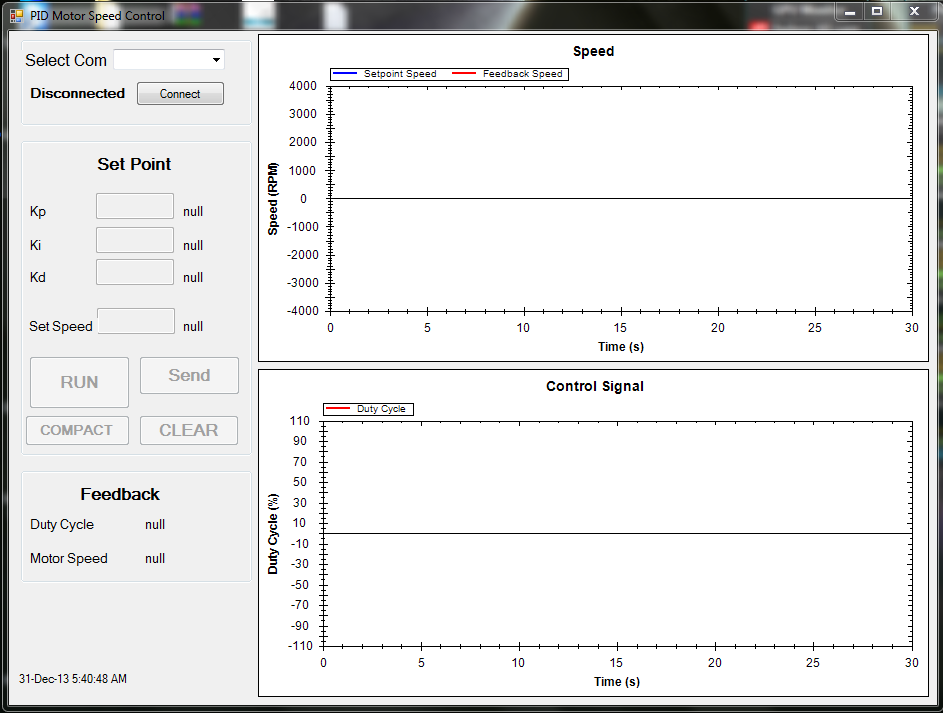
}

Cú pháp còn lại tương tự lúc gửi thông tin từ vi điều khiển lên máy tính, 0xFF gửi báo hiệu nhận thông số điều khiển với 4 byte data tách từ một số real.

Sau khi thực hiện xong bước này, chương trình sẽ quay lại bước đầu và tiếp tục làm để điều khiển tốc độ của động cơ đạt được đến tốc độ đặt mong muốn.

1. Giao diện trên máy tính

Giao diện trên máy tính được lập trình và thiết kế bằng visual C#. Giao diện như sau:



1. Kết nối cổng COM
2. Thông số gửi xuống vi điều khiển, gồm có: Kp, Ki, Kd và tốc độ đặt
3. Nút điều khiển, gồm có: Khởi động, gửi dữ liệu, chuyển đổi chế độ xem,xóa dữ liệu đang có
4. Tốc độ và duty cycle của động cơ

Code nhận dữ liệu từ Vi điều khiển

private void OnCom(object sender, System.IO.Ports.SerialDataReceivedEventArgs e) //UART RX

{

int buffersize = 42;// 6 channels \* 7 bytes

byte[] buffer = new byte[buffersize];

float data;

string s;

Com.Read(buffer, 0, buffersize);

for (byte i = 0; i <= (buffersize - 6); i++)

{

if ((buffer[i] == 0xFF) && (buffer[i + 6] == 0xFE))

{

data = BitConverter.ToSingle(buffer, i + 2);

s = data.ToString();

switch (buffer[i + 1])

{

case 0xF0: Display\_Kp(s); break;

case 0xF1: Display\_Ki(s); break;

case 0xF2: Display\_Kd(s); break;

case 0xF3: Display\_Speed\_sp(s); break;

case 0xF4: Display\_Cycle(s); break;

case 0xF5: Display\_Speed\_fb(s); break;

}

}

}

}

Code gửi dữ liệu cho Vi điều khiển

private void SENDDATA(float Value, int channel)

{

byte[] data\_encode = new byte[7];

byte[] data\_value = new byte[4];

byte tx\_buffersize = 7;

data\_value = BitConverter.GetBytes(Value).ToArray();

data\_encode[0] = 0xFF;

switch (channel)

{

case 0: data\_encode[1] = 0xE0; break;

case 1: data\_encode[1] = 0xE1; break;

case 2: data\_encode[1] = 0xE2; break;

case 3: data\_encode[1] = 0xE3; break;

}

for (int i = 2; i <= 5; i++)

{

data\_encode[i] = data\_value[i - 2];

}

data\_encode[6] = 0xFE;

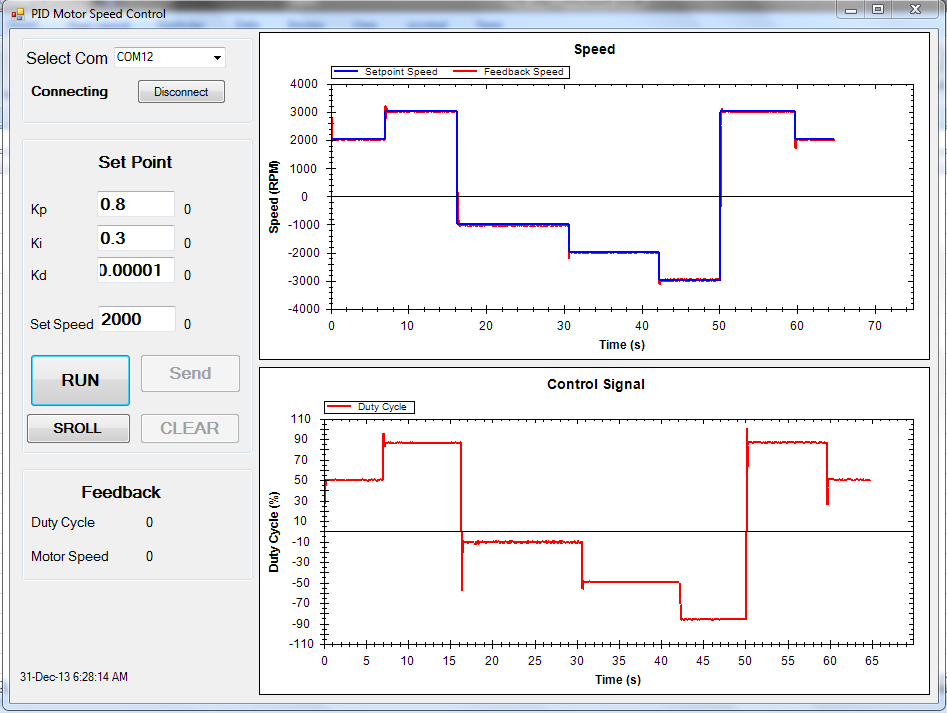
Com.Write(data\_encode, 0, tx\_buffersize);

}

1. **Kết quả thu được**

Sau đây là một số hình ảnh thu được của việc điều khiển tốc độ động cơ bằng mô hình được thực hiện như trên.

Thực nghiệm trên một số tốc độ 2000, 3000, -1000, -2000,-3000 (Kp = 0.8 Ki = 0.3 Kd = 0.00001)



Qua các kết quả thu được ta thấy mô hình điều khiển động cơ hoạt động khá tốt, thời gian xác lập khoảng 0,6 giây, vọt lố là không có. Nhưng có thể điều chỉnh thông số PID để có được kết quả tốt hơn.