**ĐẠI HỌC QUỐC GIA TP. HCM**

**TRƯỜNG ĐẠI HỌC BÁCH KHOA**


# BÁO CÁO BÀI TẬP NHÓM

## CHƯƠNG 4: ĐIỀU KHIỂN VỊ TRÍ ĐỘNG CƠ, ĐỌC ENCODER VÀ HIỂN THỊ ĐÁP ỨNG

**Giáo viên hướng dẫn: TS. Nguyễn Vĩnh Hảo**

**Nhóm thực hiện: Nhóm 9**

**Sinh viên thực hiện:**

| STT | MSSV | HỌ VÀ TÊN | LỚP |
|---|---|---|---|
| 1 | 2120064 | Nguyễn Văn Thành | L01 |
| 2 | 1751050 | Nguyễn Thanh Trung Kiên | L01 |
| 3 | 1814577 | Nguyễn Xuân Trường | L01 |
| 4 | 1913307 | Phan Trung Hậu | L01 |

TP. HỒ CHÍ MINH, NĂM 2022

# Mục lục

# I. CHUẨN BỊ THIẾT BỊ VÀ LINH KIỆN

## 1.1 Động cơ Servo GM25 – 370 DC Geared Motor:

- Động Cơ DC Servo GM25-370 DC Geared Motor được tích hợp thêm Encoder hai kênh AB giúp đọc và điều kiển chính xác vị trí, chiều quay của động cơ trong các ứng dụng cần độ có chính xác cao: điều khiển PID, Robot tự hành,....
- Động Cơ DC Servo GM25-370 DC Geared Motor có cấu tạo bằng kim loại cho độ bền và độ ổn định cao, được sử dụng trong các mô hình robot, xe, thuyền,..., hộp giảm tốc của động cơ có nhiều tỉ số truyền giúp bạn dễ dàng lựa chọn giữa lực kéo và tốc độ (lực kéo càng lớn thì tốc độ càng chậm và ngược lại), động cơ sử dụng nguyên liệu chất lượng cao.

➢ ***Thông số kỹ thuật:***

- Điện áp sử dụng: 12VDC
- Đường kính: 25mm
- Encoder: Cảm biến từ trường Hall, có 2 kênh AB lệch nhau giúp xác định chiều quay và vận tốc của động cơ, đĩa Encoder trả ra 11 xung/1 kênh/ 1 vòng (nếu đo tín hiệu đồng thời của cả hai kênh sẽ thu được tổng 22 xung / 1 vòng quay của Encoder).
- Cách tính số xung của mỗi kênh trên 1 vòng quay của trục chính động cơ = Tỉ số truyền x số xung của Encoder, ví dụ tỉ số 150:1 thì số xung Encoder trả ra cho 1 vòng quay của trục chính động cơ sẽ là 11 x 150 = 1650 xung / 1 kênh.
- Điện áp cấp cho Encoder hoạt động: 3.3~5VDC, lưu ý cấp quá áp hoặc ngược chiều sẽ làm cháy Encoder ngay lập tức!

  Nhóm sử dụng ***Loại 12VDC 250RPM:***

  - Tỉ số truyền 34:1 (động cơ quay 34 vòng trục chính hộp giảm tốc quay 1 vòng).
  - Dòng không tải: 150mA
  - Dòng chịu đựng tối đa khi có tải: 750mA
  - Tốc độ không tải: 250RPM (250 vòng 1 phút)
  - Tốc độ chịu đựng tối đa khi có tải: 140RPM (140 vòng 1 phút)

- Lực kéo Moment định mức: 4.3KG.CM
- Lực léo Moment tối đa: 5.2KG.CM
- Chiều dài hộp số L: 21mm
- Số xung Encoder mỗi kênh trên 1 vòng quay trục chính: 11 x 34 = 374 xung.

➢ ***Sơ đồ chân của động cơ:***

- M1 : Dây cấp nguồn cho động cơ.
- GND : Dây cấp nguồn cho Encoder, 0VDC.
- C1/A: Kênh trả xung A
- C2/B: Kênh trả xung B
- VCC: Dây cấp nguồn cho Encoder 3.3~5VDC
- M2: Dây cấp nguồn cho động cơ


***Hình 1: Động cơ 12VDC 250RPM***

## 1.2 L298 DC Motor Driver

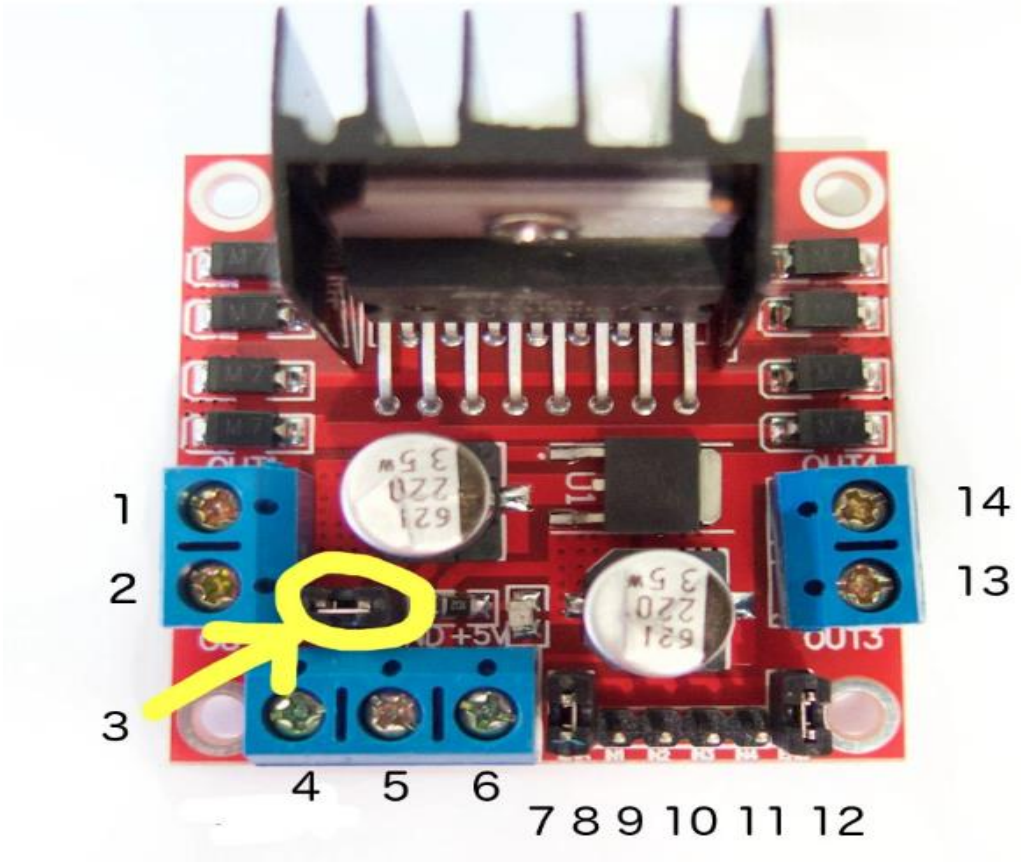
Module điều khiển motor L298N loại 1 có sẵn ốc gắn sử dụng IC điều khiển L298N có thể điều khiển 2 động cơ một chiều hoặc 1 động cơ bước 4 pha.

Module L298N loại 1:

+ Được thiết kế chắc chắn, có sẵn chỗ bắt ốc vào mô hình

+ Có gắn tản nhiệt chống nóng cho IC, giúp IC có thể điều khiển với dòng đỉnh đạt 2A.

IC L298N được gắn với các đi ốt trên board giúp bảo vệ vi xử lý chống lại các dòng điện cảm ứng từ việc khởi động/ tắt động cơ.




***Hình 2: L298 DC Motor Driver***

- **Sơ đồ chân L298 DC Motor Driver**

1. DC motor 1 “+” hoặc stepper motor A+

2. DC motor 1 “-” hoặc stepper motor A-

3. 12V jumper – tháo jumper ra nếu sử dụng nguồn trên 12V. Jumper này dùng để cấp nguồn cho IC ổn áp tạo ra nguồn 5V nếu nguồn trên 12V sẽ làm cháy IC Nguồn

4. Cắm dây nguồn cung cấp điện áp cho motor vào đây từ 6V đến 35V.

5. Cắm chân GND của nguồn vào đây

6. Ngõ ra nguồn 5V, nếu jumper đầu vào không rút ra.

7. Chân Enable của Motor 1, chân này dùng để cấp xung PWM cho motor nếu dùng VDK thì rút jumper ra và cắm chân PWM vào đây. Giữ nguyên khi dùng với động cơ bước

8. IN1

9. IN2

10. IN3

11. IN4

12. Chân Enable của Motor 2, chân này dùng để cấp xung PWM cho motor nếu dùng VDK thì rút jumper ra và cắm chân PWM vào đây. Giữ nguyên khi dùng với động cơ bước

13. DC motor 2 “+” hoặc stepper motor B+

14. DC motor 2 “-” hoặc stepper motor B-

### 1.3 Nguồn Adapter 12VDC 2A


***Hình 3 Nguồn Adapter 12VDC 2A***

***Thông số kỹ thuật***

- Điện áp đầu vào: AC100-240V  50/60HZ
- Đầu cắm AC
- Điện áp ra: 12VDC
- Dòng điện ra: Max 2A
- Chiều dài đường dây đầu ra: tổng chiều dài 1m
- Ổ cắm DC: 5.5 * 2.5mm (tương thích 5.5 * 2.1mm)
- Trọng lượng: 92g

### 1.4 USB Uart CH340


***Hình 4: USB Uart CH340***

**Thông số kỹ thuật:**

- Sử dụng điện áp 5VDC cấp trực tiếp từ cổng USB.
- Các chân tín hiệu ngõ ra:
- 5V: Chân cấp nguồn 5VDC từ cổng USB, tối đa 500mA.
- VCC: chân chọn mức tín hiệu TTL cho TX/RX, nếu nối với chân 5V sẽ là 5VDC, nối với 3V3 sẽ là 3.3VDC.
- 3V3:Chân nguồn 3.3VDC (dòng cấp rất nhỏ tối đa 100mA), không sử dụng để cấp nguồn, thường chỉ sử dụng để thiết đặt mức tín hiệu Logic.
- TXD: chân truyền dữ liệu UART TTL, dùng kết nối đến chân nhận RX của các module sử dụng mức tín hiệu TTL.
- RXD: chân nhận dữ liệu UART TTL, dùng kết nối đến chân nhận TX của các module sử dụng mức tín hiệu TTL.
- GND: chân cấp nguồn 0VDC.
- Tích hợp led tín hiệu TX, RX.
- Tương thích với hầu hết các hệ điều hành hiện nay: Windows, Mac, Linux.

## 1.5 Kit STM32F407


***Hình 5: Kit STM32F407***

- Kit STM32F407 Discovery hiện là loại kit được rất nhiều trường đại học hiện nay sử dụng trong giảng dạy vi điều khiển ARM, nếu so sánh về ngoại vi và sức mạnh của STM32 so với các dòng ARM của các hãng khác thì ở cùng 1 tầm giá, ARM của ST vượt trội về cấu hình và ngoại vi hơn rất nhiều.

**Thông số kỹ thuật:**

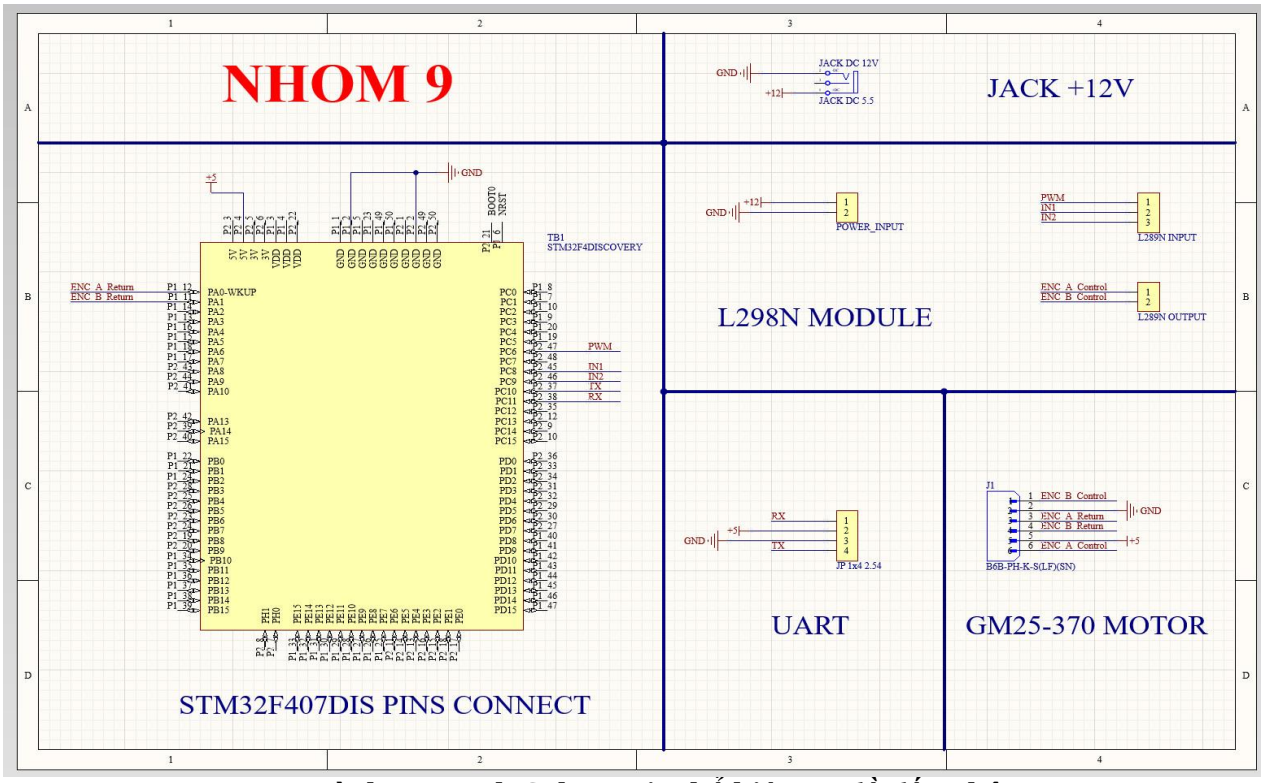
- Vi điều khiển chính: STM32F407VGT6 microcontroller featuring 32-bit ARM Cortex-M4F core, 1 MB Flash, 192 KB RAM in an LQFP100 package.
- Tích hợp sẵn mạch nạp và Debug ST-LINK/V2 with selection mode switch to use the kit as a standalone ST-LINK/V2 (with SWD connector for programming and debugging).
- Nguồn cấp từ cổng Mini USB qua các IC nguồn chuyển thành 3v3 để cấp cho MCU.
- Có sẵn các chân nguồn: 3 V and 5 V.

- Có sẵn cảm biến gia tốc: LIS302DL, ST MEMS motion sensor, 3-axis
- Có sẵn bộ xử lý âm thanh: MP45DT02, ST MEMS audio sensor, omni-directional digital microphone.
- Có sẵn bộ: CS43L22, audio DAC with integrated class D speaker driver.
- Có sẵn 8 Led.
- Có Led thông báo trạng thái nguồn
- Có nút nhấn và nút Reset tích hợp.
- Có cổng Micro USB OTG
- DSP Application.

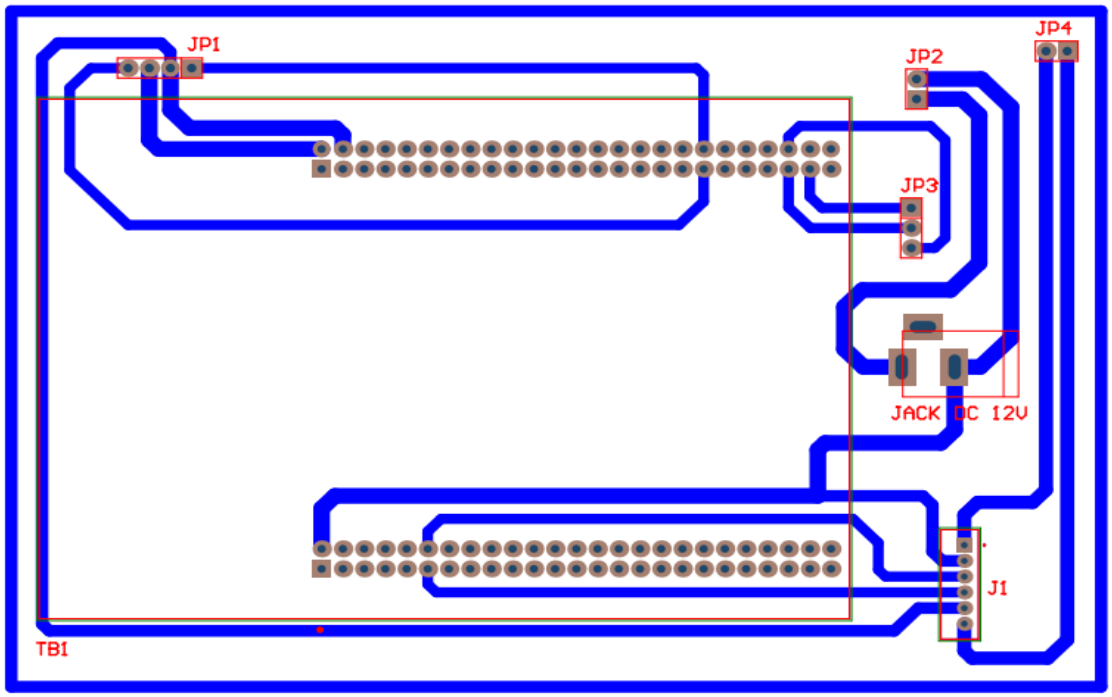
# II. SƠ ĐỒ KẾT NỐI CHÂN VÀ NGUYÊN LÝ HOẠT ĐỘNG

## 2.1 Sơ đồ kết nối chân

- Nhóm sử dụng phần mềm Altium Designer để thiết kế mạch Schematic và vẽ mạch in PCB để làm mạch phần cứng.




***Hình 6: Mạch Schematic thể hiện sơ đồ đấu chân***




***Hình 7: Mạch in  PCB***

***Qúa trình chọn chân đấu nối:***

- Đọc encoder sử dụng port A:

  +PA0: ENC A

  + PA1: ENC B
- Giao tiếp UART: Sử dụng port C giao tiếp UART4:

  +PC10: UART TX

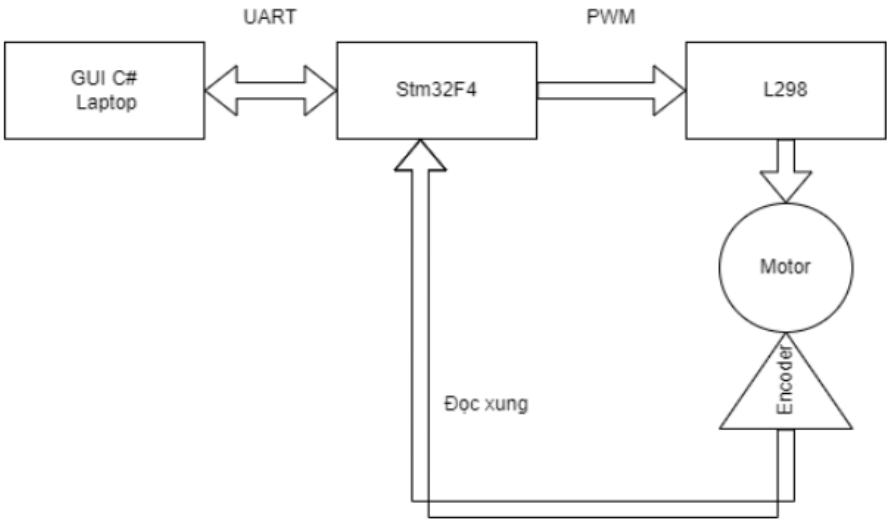
  +PC11: UART RX
- Điều rộng xung PWM và quy định chiều quay motor:

  + PC6: PWM

  +PC8: IN1

  +PC9: IN2

## 2.2 Nguyên lý hoạt động




***Hình 8: Sơ đồ nguyên lý***

+ Nhóm sẽ viết dao diện bằng C# để đọc và xuất UART và vẽ đáp ứng hiện thị vị trí động cơ dưới dạng góc quay.

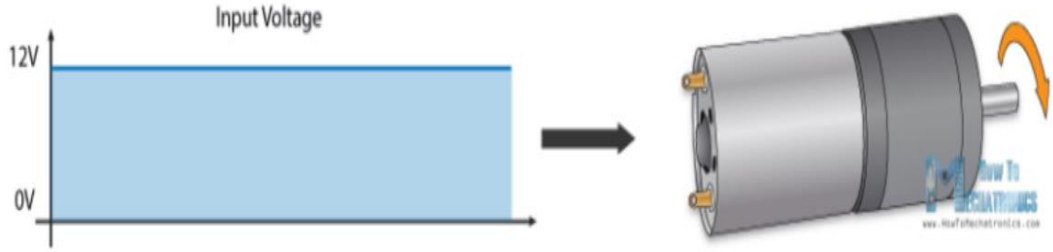
+ STM32F4: đọc xung encoder 2 kênh A,B theo 2 mode là X1 và X4. Điều rộng xung PWM để điều khiển động cơ.

+ L298 điều khiển động cơ thông qua việc thay đổi điện áp cấp vào động cơ qua 2 chân out1 và out2.

+ Motor hoạt động: trục chính motor quay được bao nhiêu vòng thì sẽ xuất được bấy nhiêu xung ENC A và ENC B ( với 1 vòng là 374 xung). Rồi trả xung về Stm32f4 để đọc xung theo mode đã chọn và hiển thị đáp ứng lên giao diện C# đã tạo.

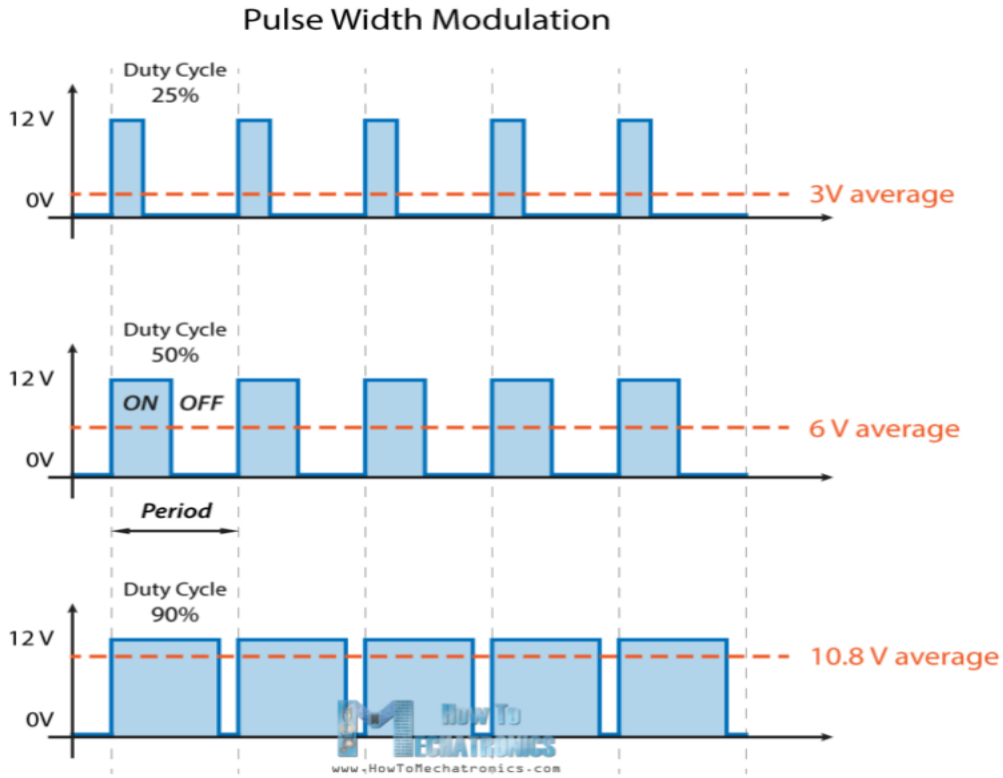
❖ ***Quá trình thay đổi PWM  để thay đổi điện áp như sau:***

- Thay đổi độ rộng xung kích để điều khiển linh kiện đóng ngắt (SCR hay Transistor) để bật và tắt nguồn với tốc độ nhanh từ đó điều khiển tốc độ động cơ.
- Trường hợp Duty cycle 100%: nguồn cấp trực tiếp cho động cơ là 12V và liên tục




***Hình 9: Duty cycle 100%***

- Các trường hợp còn lại khi thay đổi Duty cycle thay đổi từ 25%-50%-90%:

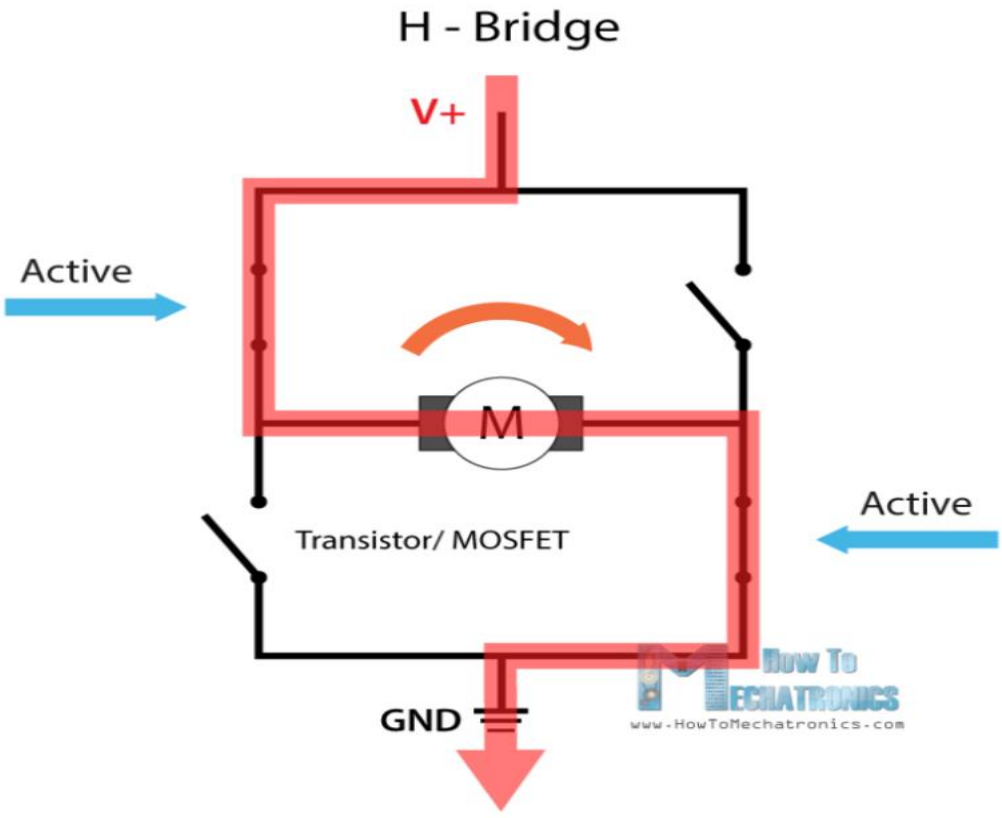



***Hình 10: Trường hợp thay đổi Duty cycle***

- Khi Duty cycle càng lớn thì điện áp trung bình cấp cho động cơ cào lớn, động cơ quay càng nhanh.

❖ ***Thay đổi chiều quay động cơ thông qua 2 chân IN1 và IN2:***

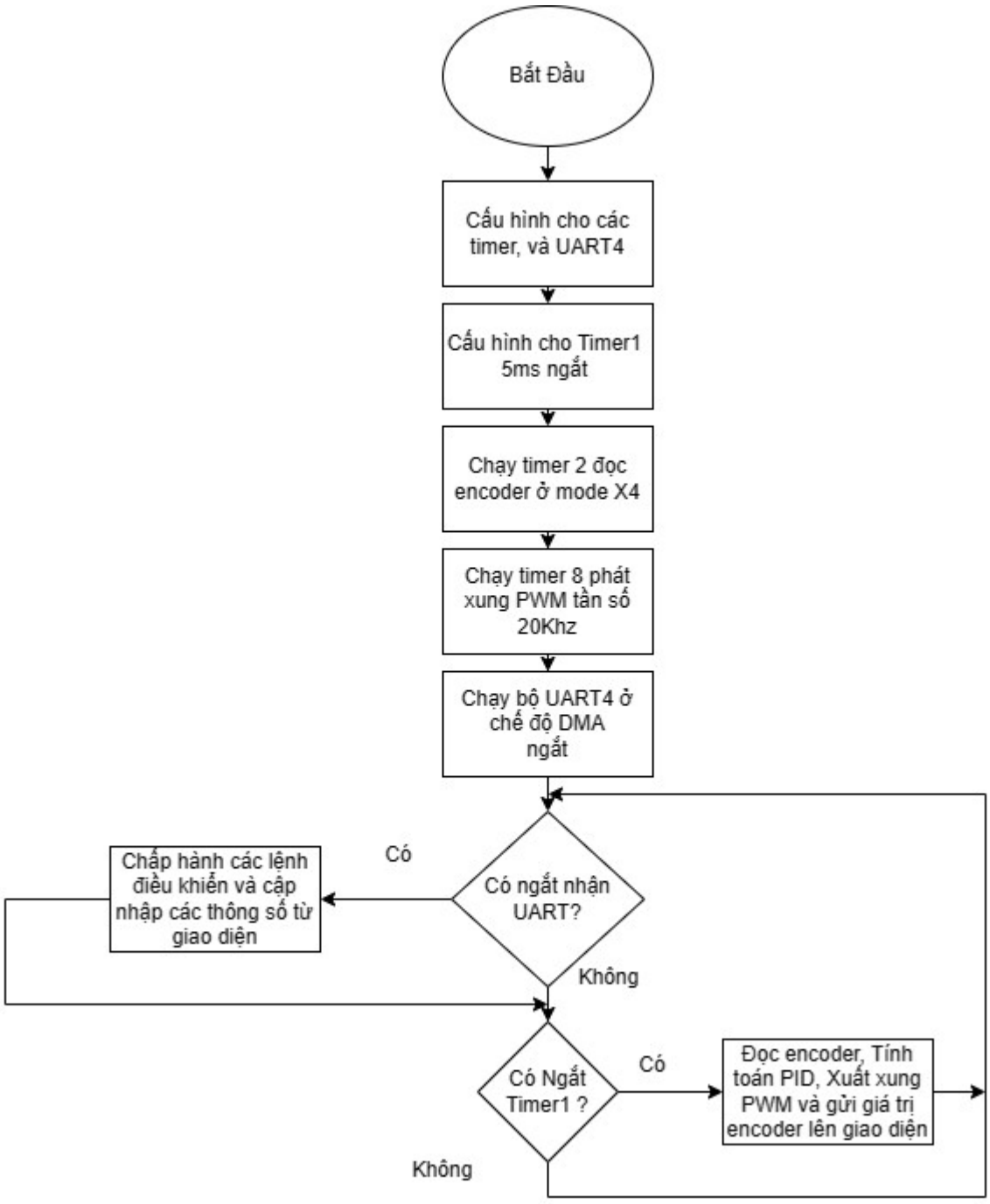
- Để điều khiển hướng quay, chúng ta chỉ cần đảo ngược hướng của dòng điện qua động cơ, và phương pháp phổ biến nhất để làm điều đó là sử dụng mạch cầu H. Một mạch cầu H chứa bốn chân chuyển mạch, điện trở hoặc MOSFET, với động cơ ở trung tâm tạo thành một cấu hình giống như chữ H. Bằng cách kích hoạt hai công tắc cụ thể cùng một lúc, chúng ta có thể thay đổi hướng của dòng điện, do đó thay đổi hướng quay của động cơ.




***Hình 11: Thay đổi chiều quay động cơ***

# III. VIẾT CHƯƠNG TRÌNH

## 3.1 Lưu đồ giải thuật để viết code trong Keil uVision 5



## 3.2 Code chương trình

- Sử dụng timer 2 đọc encoder (32bit)
- Timer 8 điều rộng xung PWM
- Đọc và xuất UART4
- Sử dụng Timer 1 để lặp và tính toán các thông số PID

```
SystemClock_Config();

/* USER CODE BEGIN SysInit */

/* USER CODE END SysInit */

/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_DMA_Init();
MX_TIM2_Init(); // Tim 2 encoder
MX_TIM8_Init(); // Tim 8 PWM
MX_UART4_Init();
MX_TIM1_Init(); // Tim 1 PID loop
/* USER CODE BEGIN 2 */
HAL_TIM_PWM_Start(&htim8,TIM_CHANNEL_1);
HAL_TIM_Encoder_Start(&htim2,TIM_CHANNEL_1|TIM_CHANNEL_2);
HAL_UART_Receive_DMA(&huart4,u8_Rx_Data,6);
```

- Scaler cho timer1:

```
/* USER CODE END TIM1_Init 1 */
htim1.Instance = TIM1;
htim1.Init.Prescaler = 499;   //  1680/2*1000/168Mhz = 499 => 5ms
htim1.Init.CounterMode = TIM_COUNTERMODE_UP;
htim1.Init.Period = 1679;
```

- Code cho các timer 2 và timer 8:

```
/* USER CODE END TIM2_Init 1 */
htim2.Instance = TIM2;
htim2.Init.Prescaler = 0;
htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
htim2.Init.Period = 0xFFFFFFFF; // 65535
htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
sConfig.EncoderMode = TIM_ENCODERMODE_TI12;
sConfig.IC1Polarity = TIM_ICPOLARITY_RISING;
sConfig.IC1Selection = TIM_ICSELECTION_DIRECTTI;
sConfig.IC1Prescaler = TIM_ICPSC_DIV1;
sConfig.IC1Filter = 0;
sConfig.IC2Polarity = TIM_ICPOLARITY_RISING;
sConfig.IC2Selection = TIM_ICSELECTION_DIRECTTI;
sConfig.IC2Prescaler = TIM_ICPSC_DIV1;
sConfig.IC2Filter = 0;
if (HAL_TIM_Encoder_Init(&htim2, &sConfig) != HAL_OK)
```

```
/* USER CODE END TIM8_Init 1 */
htim8.Instance = TIM8;
htim8.Init.Prescaler = 83; //168Mhz/ 84*x = 20kHZ
htim8.Init.CounterMode = TIM_COUNTERMODE_UP;
htim8.Init.Period = x;
htim8.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
htim8.Init.RepetitionCounter = 0;
htim8.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
```

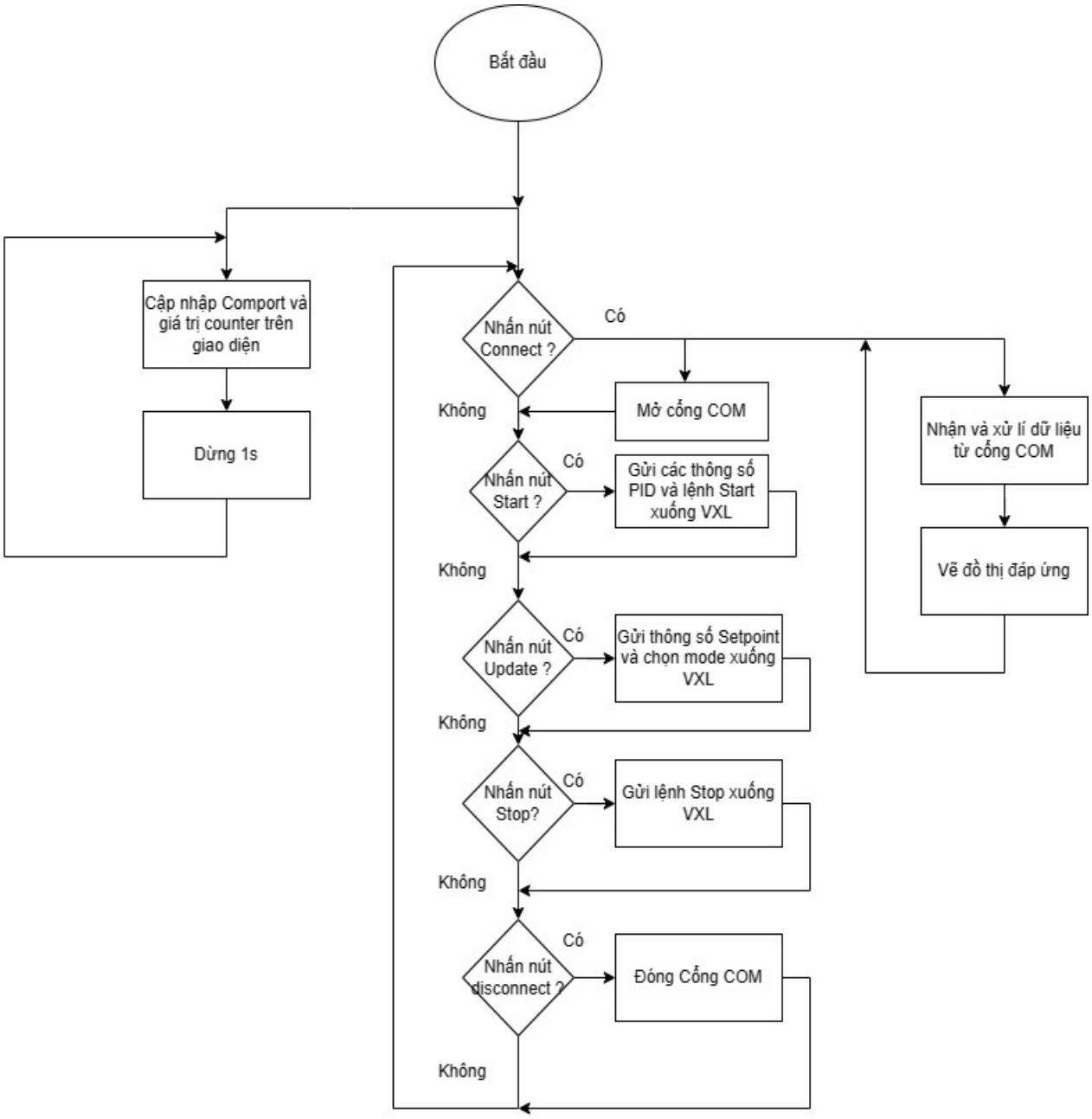
- Ngắt timer1 là 5ms:

```
// TIM1 interup 0.005s
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
  if (htim -> Instance == TIM1){
  enc_cnt  = __HAL_TIM_GetCounter(&htim2);
  //enc_cnt =1000;
  e = s_enc_cnt - enc_cnt;

  static float u=0,ul = 0 ,el = 0,e2 =0;
  u= ul + Kp*(e-el) + Ki*Ts*(e+el)/2 + Kd*(e-2*el+e2)/Ts;
  pid = u;
  if (u -pid>0.5) pid++;

  if (pid > x){
  pid = x;
  }
  else if (pid < -x){
  pid =-x;
  }
  e2= el;
  el = e;
  ul=pid;
  if (pid >=0){
    __HAL_TIM_SetCompare(&htim8,TIM_CHANNEL_1, pid);
   HAL_GPIO_WritePin(GPIOC,GPIO_PIN_8,0);
   HAL_GPIO_WritePin(GPIOC,GPIO_PIN_9,1);
}
else{
   __HAL_TIM_SetCompare(&htim8,TIM_CHANNEL_1, -pid);
   HAL_GPIO_WritePin(GPIOC,GPIO_PIN_8,1);
   HAL_GPIO_WritePin(GPIOC,GPIO_PIN_9,0);
}

   u8_Tx_Data[0] = enc_cnt>>8;
   u8_Tx_Data[1] = enc_cnt;
   u8_Tx_Data[2] = 0xFF;

HAL_UART_Transmit_DMA(&huart4,u8_Tx_Data,3);
}
```

- Ngắt UART4_RX để cập nhật các thông số từ giao diện:

```
// UART4 RX_interup
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart){
  if (u8_Rx_Data[5] != '#' ){
    //  Error
  }
  else{
    //double d_temp = ((u8_Rx_Data[2]) -48)*100 + (u8_Rx_Data[3]-48)*10 +(u8_Rx_Data[4]) -48;
    char* ptr ;

    switch (u8_Rx_Data[0]){
      case('P') :
        // update KP
        for (int i =0; i < 4 ; i++)
        {
          ptr = (char*)(&Kp);
          *(ptr+i) =  u8_Rx_Data[i+1];
        }
        break ;
      case ('I'):
        // update Ki
      for (int i =0; i < 4 ; i++)
        {
          ptr = (char*)(&Ki);
          *(ptr+i) =  u8_Rx_Data[i+1];
        }
        break ;
      case ('D'):
        // Update Kd
      for (int i =0; i < 4 ; i++)
        {
          ptr = (char*)(&Kd);
          *(ptr+i) =  u8_Rx_Data[i+1];
        }
        HAL_TIM_Base_Start_IT(&htim1);;

        break ;
    case ('S'):
      u8_Tx_Data[0] = u8_Rx_Data[1];
      u8_Tx_Data[1] = u8_Rx_Data[2];
      u8_Tx_Data[2] = 0xFE;
      HAL_UART_Transmit_DMA(&huart4,u8_Tx_Data,3);

      ptr = (char*)(&s_enc_cnt);
      *ptr = u8_Rx_Data[1];
      *(ptr+1) = u8_Rx_Data[2];
      if (u8_Rx_Data[3] == '1')
      {
        Enc_mode(MODE_X1);
      }
      else if (u8_Rx_Data[3] =='4')
      {
        Enc_mode(MODE_X4);
      }

      break ;
    default:
      //Stop
      HAL_TIM_Base_Stop_IT(&htim1);
      __HAL_TIM_SetCompare(&htim8,TIM_CHANNEL_1, 0);
  }
```

### 3.3 Lưu đồ giải thuật code C#



## 3.4 Lập trình giao diện Visual Studio

Khai báo tất cả thư viện được sử dụng trong phần thiết kế giao diện như sau:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading;
using System.Windows.Forms;
using System.IO.Ports;
using System.Runtime.Remoting.Contexts;
using ZedGraph;
using static System.Windows.Forms.VisualStyles.VisualStyleElement.Rebar;
using static System.Windows.Forms.VisualStyles.VisualStyleElement.TaskbarClock;
```

Phần khai báo biến sử dụng:

```
    #region Variable
    double time = 0;
    string dataIn;
    string dataOut="Hello\n";
   // Int16 vitri,vitri_hientai;
    byte[] bDataOut = {0x53,1,2,3,4,0x23} ;
    Int16[] DataBuffer = new Int16[4096];
    Int16[] VitriBuffer = new Int16[4096];
    static Int16 mode = 4;
    static int buff_cnt = -1, cnt = -1, Xmax = 50, Ymax = 400, Ymin = -400, S_arg, vitri = 0, myShort = 0;
    #endregion Variable
```

Phần thiết lập gửi dữ liệu lên từ STM32F407 lên máy tính:

- Vi xử lý sẽ gửi lên 3 byte
- 8 bit đầu Counter | 8 bit sau của Counter | 0xFF ( Gửi vị trí hiện tại)
- 8 bit đầu Setpoint | 8 bit sau của Setpoint Counter | 0xFE ( Gửi Setpoint)
- 2 byte dấu

```
#region Updata
1 reference
public void Re_Data()
{
    byte[] bDataIns = new byte[3];
    byte[] bDataIn = new byte[1];
    int a;
    while (true)
    {

        if (serialPort1.IsOpen)
        {
            a = serialPort1.Read(bDataIns, 0, 3);


            if (a < 3)
            {
                serialPort1.Read(bDataIns, a, 3 - a);
            }
                if (bDataIns[2] == 0xFF)
                {

                    myShort = (Int16)(bDataIns[0] << 8 | bDataIns[1]);
                    int arg = (myShort * 360 )/ (374 * mode);
                    draw(arg, vitri);

                }
                else if (bDataIns[2] == 0xFE ){

                    vitri = (((Int16)(bDataIns[0] | bDataIns[1] << 8))*360)/(374*mode) ;

            }


        }
    }
}
#endregion Updata
```

Thiết hàm Update để cập nhật dữ liệu xuống STM32F407

```
#region Update Fuction
1 reference
public void Update()
{
    while (true)
    {

        string[] ports = SerialPort.GetPortNames();

        Invoke(new MethodInvoker(() =>
        {
            cBoxCOMPORT.Items.Clear();
            cBoxCOMPORT.Items.AddRange(ports);
            label8.Text = myShort.ToString();
        }
        ));
        Thread.Sleep(1000);
    }

}
#endregion Update Function
```

Thiết lập vẽ biểu đồ bằng Zedgraph trên Visual Studio và Realtime

```
1 reference
private void Form1_Load(object sender, EventArgs e)
{
    string[] ports = SerialPort.GetPortNames();
    cBoxCOMPORT.Items.AddRange(ports);
    GraphPane mypanne = zedGraphControl1.GraphPane;
    mypanne.Title.Text = " Vị trí của động cơ theo thời gian";
    mypanne.YAxis.Title.Text = "Vị trí (độ)";
    mypanne.XAxis.Title.Text = "Thời gian (s)";

    RollingPointPairList list1 = new RollingPointPairList(60000);
    RollingPointPairList list2 = new RollingPointPairList(60000);

    LineItem dline1 = mypanne.AddCurve("Vị trí phản hồi", list1, Color.Red, SymbolType.None);
    LineItem dline2 = mypanne.AddCurve("Vị trí đặt", list2, Color.Blue, SymbolType.None);

    mypanne.XAxis.Scale.Min = 0;
    mypanne.XAxis.Scale.Max = Xmax;
    mypanne.XAxis.Scale.MinorStep = 0.005;
    mypanne.XAxis.Scale.MajorStep = 1;

    mypanne.YAxis.Scale.Min = Ymin;
    mypanne.YAxis.Scale.Max = Ymax;
    mypanne.YAxis.Scale.MinorStep = 1;
    mypanne.YAxis.Scale.MajorStep = 5;

    zedGraphControl1.AxisChange();

    string[] myport = SerialPort.GetPortNames();

    btnDisconnect.Enabled = false;

    ThreadStart Timer = new ThreadStart(Update);
    Thread Timer_update = new Thread(Timer);
    Timer_update.IsBackground = true;
    Timer_update.Start();

}
```

```
#region Draw
1 reference
public void draw(int line1, int line2)
{
    LineItem dline1 = zedGraphControl1.GraphPane.CurveList[0] as LineItem;
    LineItem dline2 = zedGraphControl1.GraphPane.CurveList[1] as LineItem;
    if (dline1 == null)
        return;
    if (dline2 == null)
        return;

    IPointListEdit list2 = dline2.Points as IPointListEdit;
    IPointListEdit list1 = dline1.Points as IPointListEdit;
    if (list1 == null)
        return;
    if (list2 == null)
        return;
    list1.Add(time, line1);
    list2.Add(time, line2);

    GraphPane mypanne = zedGraphControl1.GraphPane;
    if (time > Xmax)
    {
        Xmax = Xmax + 25;
        mypanne.XAxis.Scale.Max = Xmax;
    }
    if (line1 > Ymax )
    {
        Ymax = line1 + 200;
        mypanne.YAxis.Scale.Max = Ymax;
    }
    if (line2 > Ymax)
    {
        Ymax = line2 + 200;
        mypanne.YAxis.Scale.Max = Ymax;
    }
```

Thiết lập các đường vẽ cho các thông số:

```
        if (line1 < Ymin)
        {
            Ymin = line1 - 200;
            mypanne.YAxis.Scale.Min = Ymin;
        }
        if (line2 < Ymin)
        {
            Ymin = line2 - 200;
            mypanne.YAxis.Scale.Min = Ymin;
        }
        //zedGraphControl1.AxisChange();
        zedGraphControl1.Invalidate();
        time += 0.005;
    }
    #endregion Draw
```

Tạo nút Connect với STM32F407

```
#region Connect_Click
1 reference
private void Connect_Click(object sender, EventArgs e)
{
    try
    {
        serialPort1.PortName = cBoxCOMPORT.Text;
        serialPort1.BaudRate = int.Parse(cBoxBaudrate.Text);
        serialPort1.DataBits = 8;
        serialPort1.Parity = Parity.None;
        serialPort1.StopBits = StopBits.One;
        serialPort1.Open();
        btnConnect.Enabled = false;
        btnDisconnect.Enabled = true;
        progressBar1.Value = 100;
        lableComStatus.Text = "ON";

    }
    catch (Exception err)
    {
        MessageBox.Show(err.Message, "Not Find Your Device!!!", MessageBoxButtons.OK, MessageBoxIcon.Error);
        lableComStatus.Text = "OFF";
    }
    ThreadStart RecevieDatadref = new ThreadStart(Re_Data);
    Thread RecevieData = new Thread(RecevieDatadref);
    RecevieData.IsBackground = true;
    RecevieData.Start();

}
#endregion Connection_Click
```

Tạo nút Disconnect với STM32F407

```
#region Disconnect_Click
1 reference
private void btnDisconnect_Click(object sender, EventArgs e)
{
    try
    {
        serialPort1.Close();
        btnConnect.Enabled = true;
        btnDisconnect.Enabled = false;
        progressBar1.Value = 0;
        lableComStatus.Text = "OFF";

    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK);

    }
    // data reset ........
}
#endregion Disconnect_Click
```

Thiết lập nút Start để gửi dữ liệu bao gồm Kp Ki Kd tơi STM32F407

```
#region Start_Click
1 reference
private void btnStart_Click(object sender, EventArgs e)
{
    if (serialPort1.IsOpen)
    {
        btnStop.Enabled = true;
        btnStart.Enabled = false;
        /* Send Ki to stm */
        bDataOut[0] = 73; // 'I'
        byte[] byteArray = BitConverter.GetBytes(float.Parse(txtKi.Text));

        for (int i = 1; i < 5; i++)
        {
            bDataOut[i] = byteArray[i - 1];
        }
        for (int i = 1; i < 5; i++)
        {
            byteArray[i - 1] = bDataOut[i];
        }
        serialPort1.Write(bDataOut, 0, 6);
        /* Send Kp to Stm */
        bDataOut[0] = 0x50; // 'P'
        byteArray = BitConverter.GetBytes(float.Parse(txtKp.Text));
        for (int i = 1; i < 5; i++)
        {
            bDataOut[i] = byteArray[i - 1];
        }
        Thread.Sleep(70);
        serialPort1.Write(bDataOut, 0, 6);
        /* Send Kd to Stm */
        bDataOut[0] = 0x44; // 'D'
        byteArray = BitConverter.GetBytes(float.Parse(txtKd.Text));
```

```
            for (int i = 1; i < 5; i++)
            {
                bDataOut[i] = byteArray[i - 1];
            }
            Thread.Sleep(70);
            serialPort1.Write(bDataOut, 0, 6);
        }
        else
        {
            MessageBox.Show("Check your connecttion");
        }
    }
    #endregion Start_Click
```

Thiết lập nút Stop để dừng Encoder

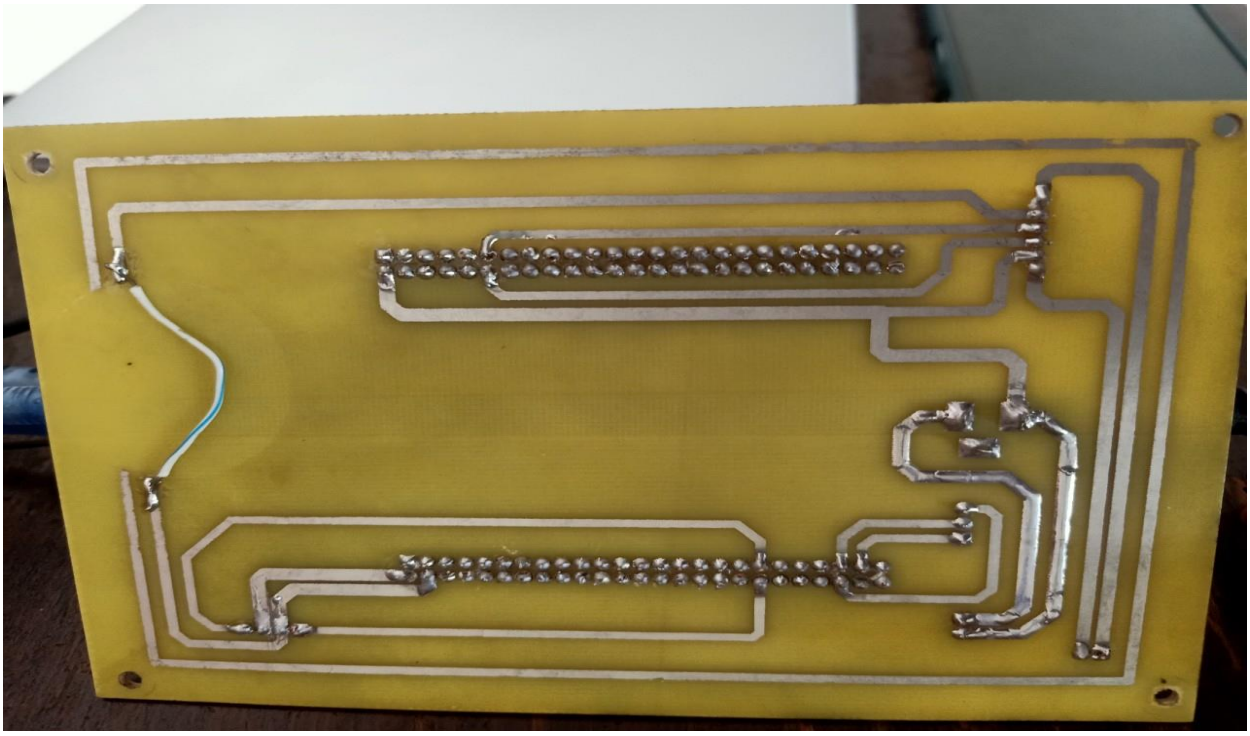
```
    #region Stop_Click
    1 reference
    private void btnStop_Click(object sender, EventArgs e)
    {
        if (serialPort1.IsOpen)
        {
            btnStart.Enabled = true;
            btnStop.Enabled = false;
            /* Send Ki to stm */
            bDataOut[0] = 0x21; //'!'
            string s1 = Encoding.UTF8.GetString(bDataOut);
            serialPort1.Write(s1);
        }
        else
        {
            MessageBox.Show("Check your connecttion");
        }
    }
    #endregion Stop_Click
```

Thiết lập nút Update để cập nhật dữ liệu được truyền xuống STM32F407

```
#region Update Button
1 reference
private void btnUpdate_Click(object sender, EventArgs e)
{

    if (serialPort1.IsOpen)
    {
        /* choose mode for encoder*/
        if (comboBox3.Text == "X1")
        {
            bDataOut[3] = 0x31; //'1'
            mode = 1;
        }
        else if (comboBox3.Text == "X4")
        {
            bDataOut[3] = 0x34; //'4'
            mode = 4;
        }

        /* Update param to stm */
        bDataOut[0] = 83; // 'S'
        S_arg = (Int16)((int.Parse(txtPoint.Text) * 374*mode) / 360);
        byte[] byteArray = BitConverter.GetBytes(S_arg);
        for (int i = 1; i < 3; i++)
        {
            bDataOut[i] = byteArray[i - 1];
        }

        serialPort1.Write(bDataOut, 0, 6);
    }
    else
    {
        MessageBox.Show("Check your connecttion");
    }
}
#endregion Update Button
```

# IV. KẾT QUẢ ĐẠT ĐƯỢC

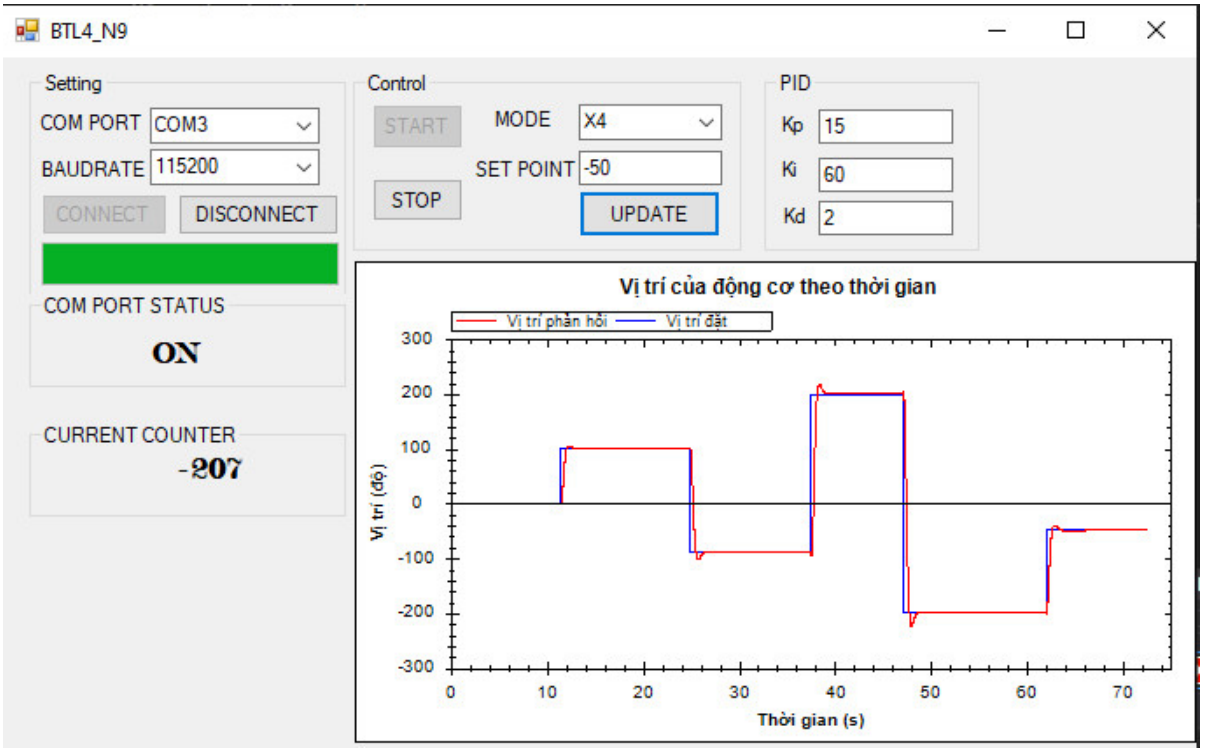
## 4.1 Mạch phần cứng



## 4.2 Kết quả hiện thị đáp ứng trên Visual Studio

- Mode X1:


- Mode X4:



- **Nhận xét:** Mode X4 sẽ cho hiện thị đáp ứng vị trí của động cơ tốt hơn Mode X1, sai số nhỏ và độ vọt lố thấp. Đường vị trí phản hồi từ động cơ bám sát với vị trí đặt cho trước hơn.