ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN ĐIỆN TỬ
BỘ MÔN ĐIỀU KHIỂN TỰ ĐỘNG

----------o0o----------

BÁO CÁO BÀI TẬP LỚN 2
LẬP TRÌNH STM32 ĐIỀU KHIỂN MOTOR

Môn Đo lường điều khiển bằng máy tính

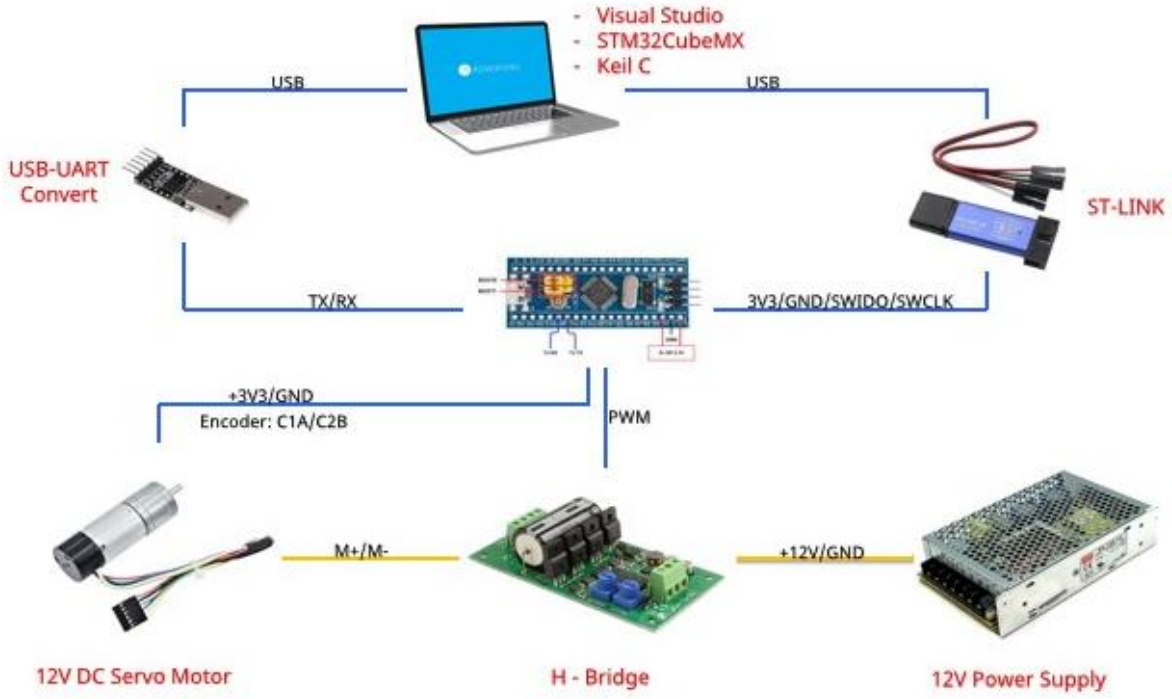
STT	HỌ VÀ TÊN	MSSV
1	Đoàn Huỳnh Quát	1914815

GVHD: Nhóm L02

TP. HỒ CHÍ MINH, THÁNG 5 NĂM 2022

MỤC LỤC

I.Giới thiệu



Các thiết bị em sử dụng:

Động cơ DC servo GA25 370 130rpm

Cầu L298H

Nguồn tổ ong 12V DC

STM32F103

Nguyên tắc hoạt động:

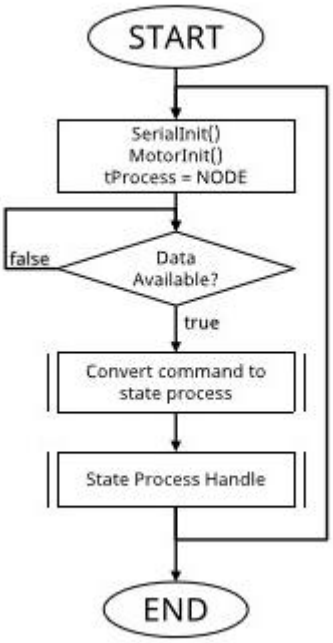
- ST Link thực hiện nạp code từ chương trình xuống STM

- USB TTL thực hiện giao tiếp uart giữa máy tính và STM

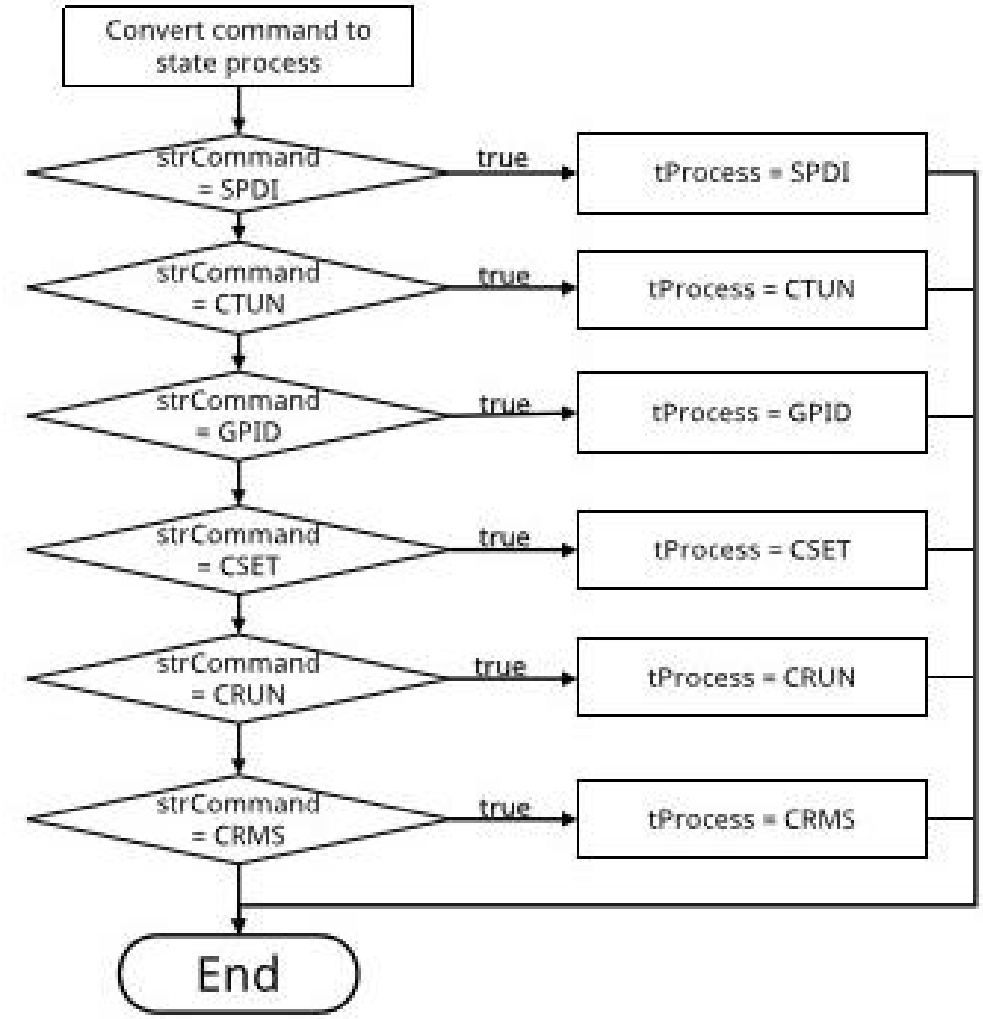
- Nguồn 12V cấp điện áp cho cầu H

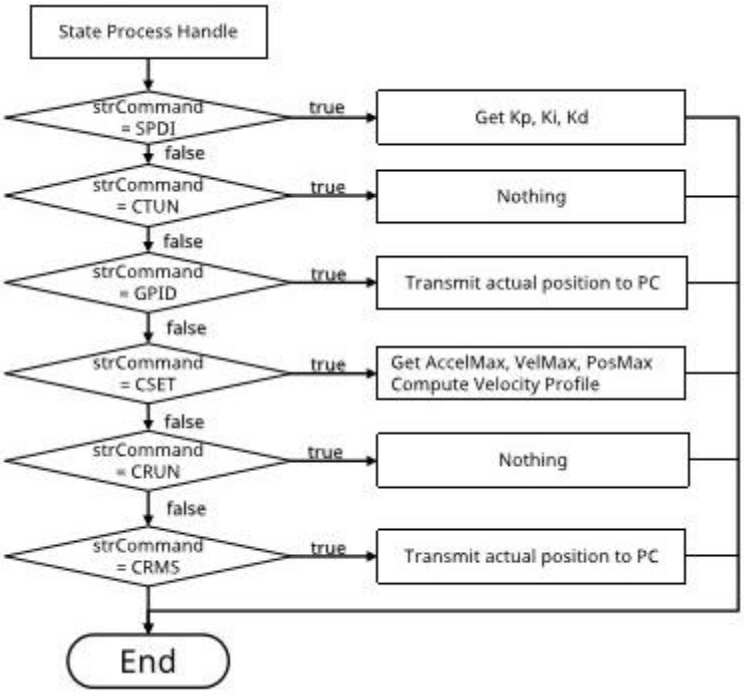
-STM32 phát xung PWM cho cầu H để điều khiển vị trí của động cơ DC

- Vị trí của động cơ này được đọc trở lại thông qua chân A, B của encoder
- Thực hiện so sánh giá trị điều khiển sử dụng thuật toán PID



Flowchart thực hiện chương trình


Convert command to state process
strCommand = SPDI
true
tProcess = SPDI
strCommand = CTUN
true
tProcess = CTUN
strCommand = GPID
true
tProcess = GPID
strCommand = CSET
true
tProcess = CSET
strCommand = CRUN
true
tProcess = CRUN
strCommand = CRMS
true
tProcess = CRMS
End

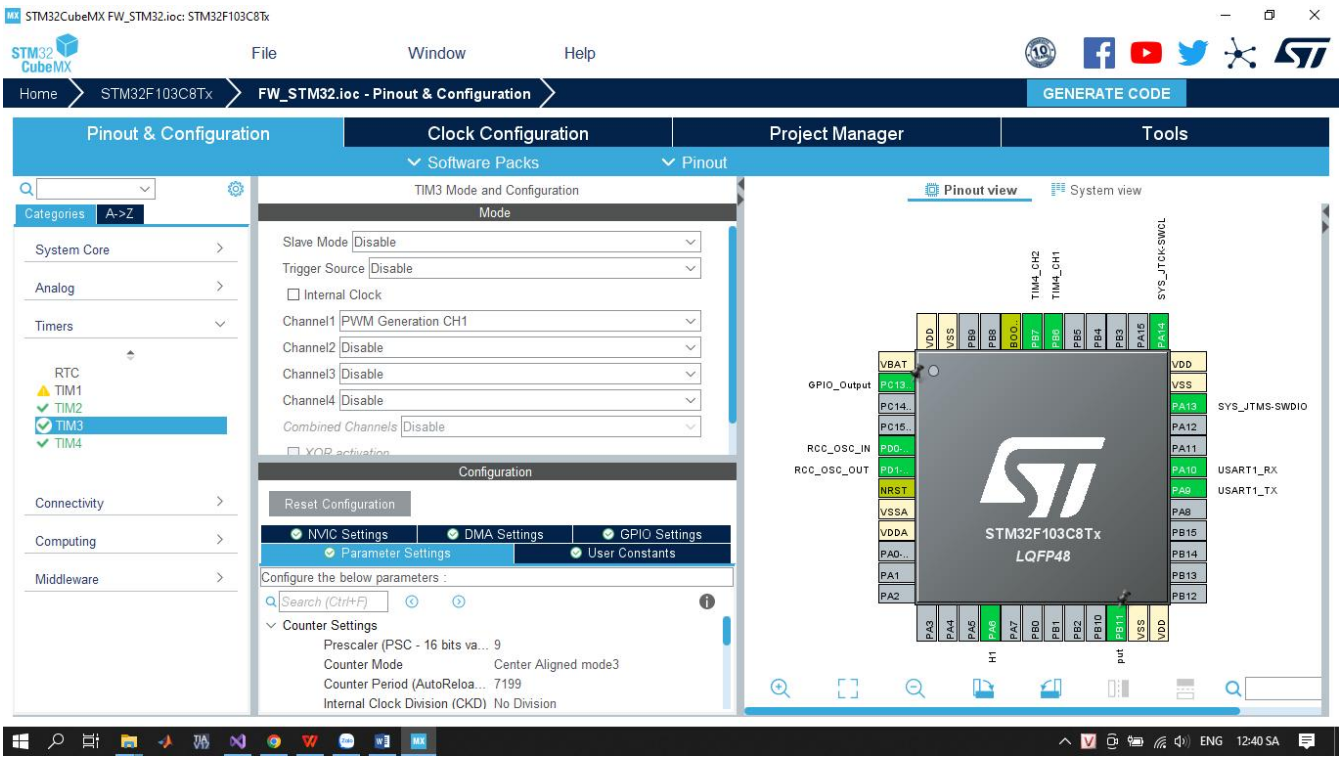


Trong hàm Process Handle nó sẽ kiểm lệnh command của mình đưa xuống có thỏa mãn các trường hợp có sẵn không.

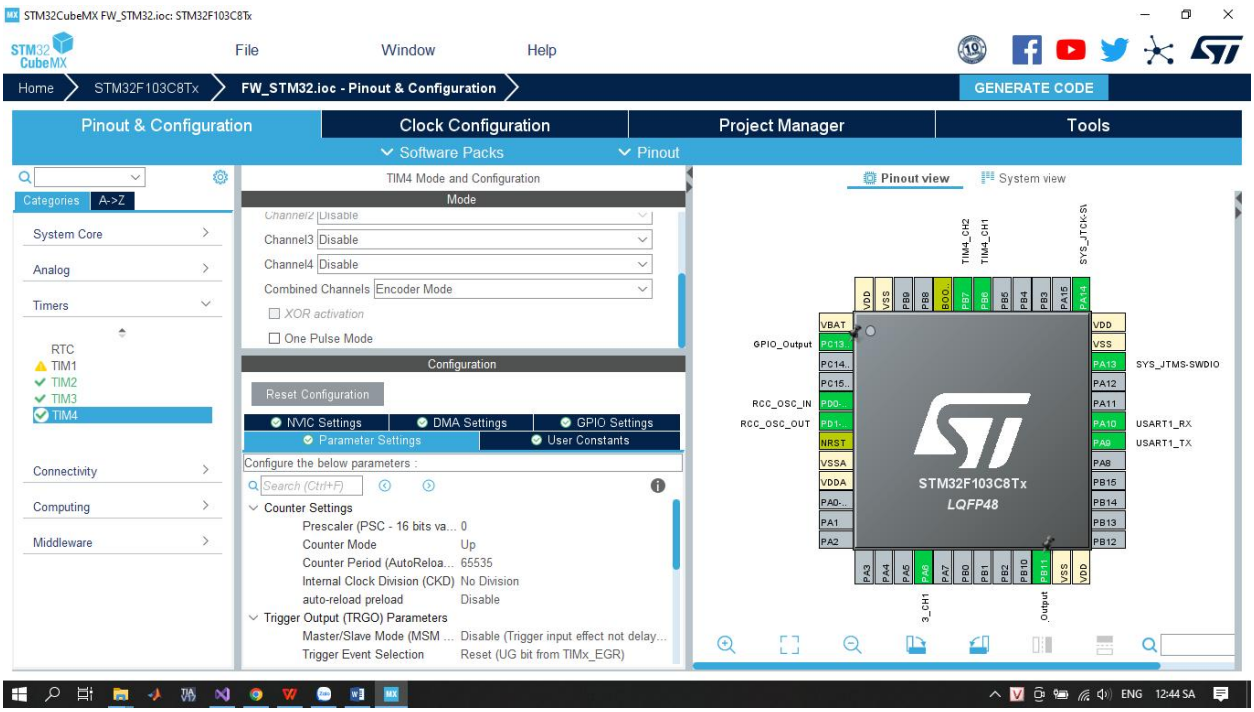
II. Firmware.

2.1 STM32CUBEMX

Việc thiết lập giống như BTL1



Setup cho timer phát xung PWM điều khiển động cơ



Setup Timer 4 thực hiện xác định tốc độ xung của encoder gắn với động cơ (Encoder

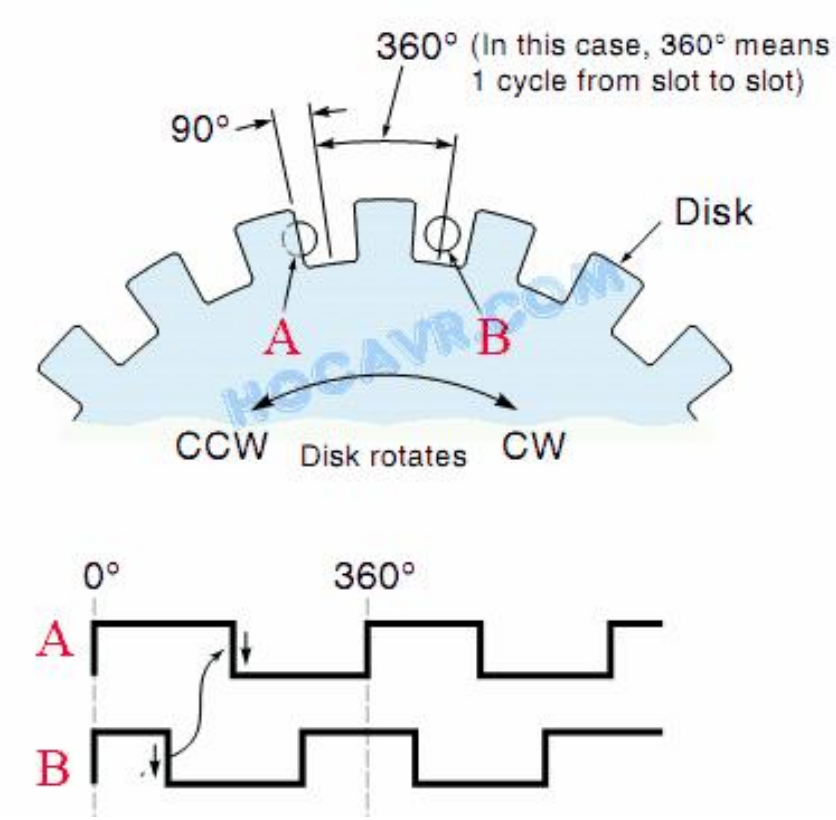
mode)

Cách thức xác định chiều quay của encoder

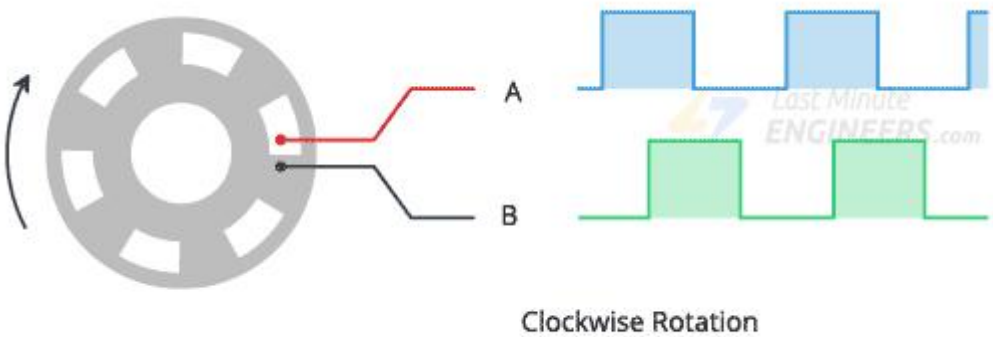
Encoder thường có 3 kênh (3 ngõ ra) bao gồm kênh A, kênh B và kênh I (Index). Trong hình 2 bạn thấy hãy chú ý một lỗ nhỏ bên phía trong của đĩa quay và một cặp phat-thu dành riêng cho lỗ nhỏ này. Đó là kênh I của encoder. Cứ mỗi lần motor quay được một vòng, lỗ nhỏ xuất hiện tại vị trí của cặp phát-thu, hồng ngoại từ nguồn phát sẽ xuyên qua lỗ nhỏ đến cảm biến quang, một tín hiệu xuất hiện trên cảm biến. Như thế kênh I xuất hiện một "xung" mỗi vòng quay của motor.

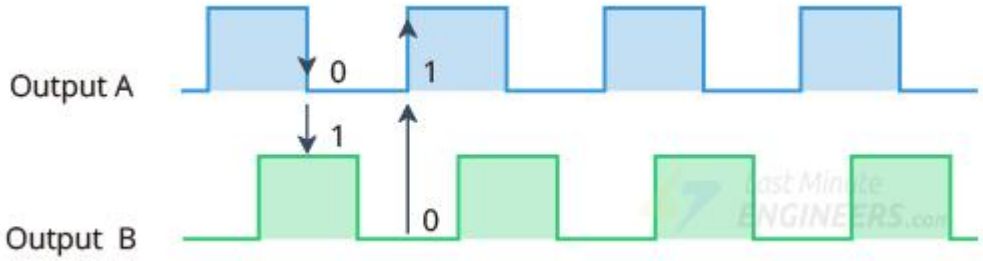
Bên ngoài đĩa quay được chia thành các rãnh nhỏ và một cặp thu-phát khác dành cho các rãnh này. Đây là kênh A của encoder, hoạt động của kênh A cũng tương tự kênh I, điểm khác nhau là trong 1 vòng quay của motor, có N "xung" xuất hiện trên kênh A. N là số rãnh trên đĩa và được gọi là độ phân giải (resolution) của encoder. Mỗi loại encoder có độ phân giải khác nhau, có khi trên mỗi đĩa chĩ có vài rãnh nhưng cũng có trường hợp đến hàng nghìn rãnh được chia. Để điều khiển động cơ, bạn phải biết độ phân giải của encoder đang dùng. Độ phân giải ảnh hưởng đến độ chính xác điều khiển và cả phương pháp điều khiển.

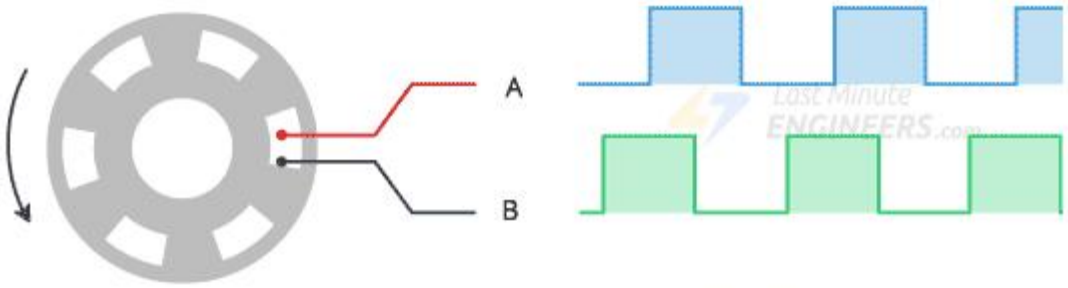
Trên các encoder còn có một cặp thu phát khác được đặt trên cùng đường tròn với kênh A nhưng lệch một chút, đây là kênh B của encoder. Với 2 tín hiệu xung A và B giúp chúng ta xác định chiều quay của động cơ. Tín hiệu xung từ kênh B có cùng tần số với kênh A nhưng lệch pha 90 độ.



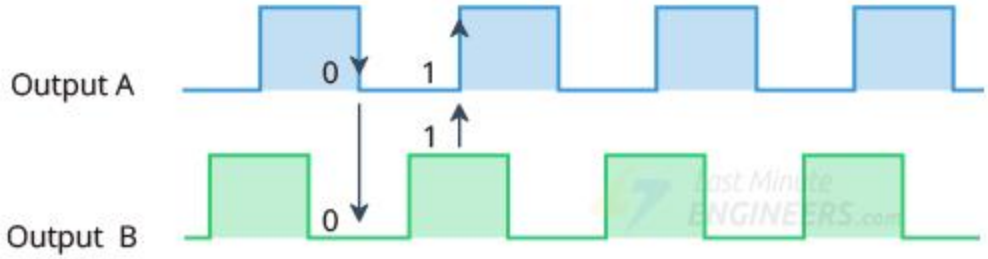
Khi cảm biến A bắt đầu bị che thì cảm biến B hoàn toàn nhận được hồng ngoại xuyên qua, và ngược lại. Hình trên là dạng xung ngõ ra trên 2 kênh. Xét trường hợp motor quay cùng chiều kim đồng hồ, tín hiệu "đi" từ trái sang phải. Bạn hãy quan sát lúc tín hiệu A chuyển từ mức cao xuống thấp (cạnh xuống) thì kênh B đang ở mức thấp. Ngược lại, nếu động cơ quay ngược chiều kim đồng hồ, tín hiệu "đi" từ phải qua trái. Lúc này, tại cạnh xuống của kênh A thì kênh B đang ở mức cao. Như vậy, bằng cách phối hợp 2 kênh A và B chúng ta không những xác định được góc quay (thông qua số xung) mà còn biết được chiều quay của động cơ (thông qua mức của kênh B ở cạnh xuống của kênh A).



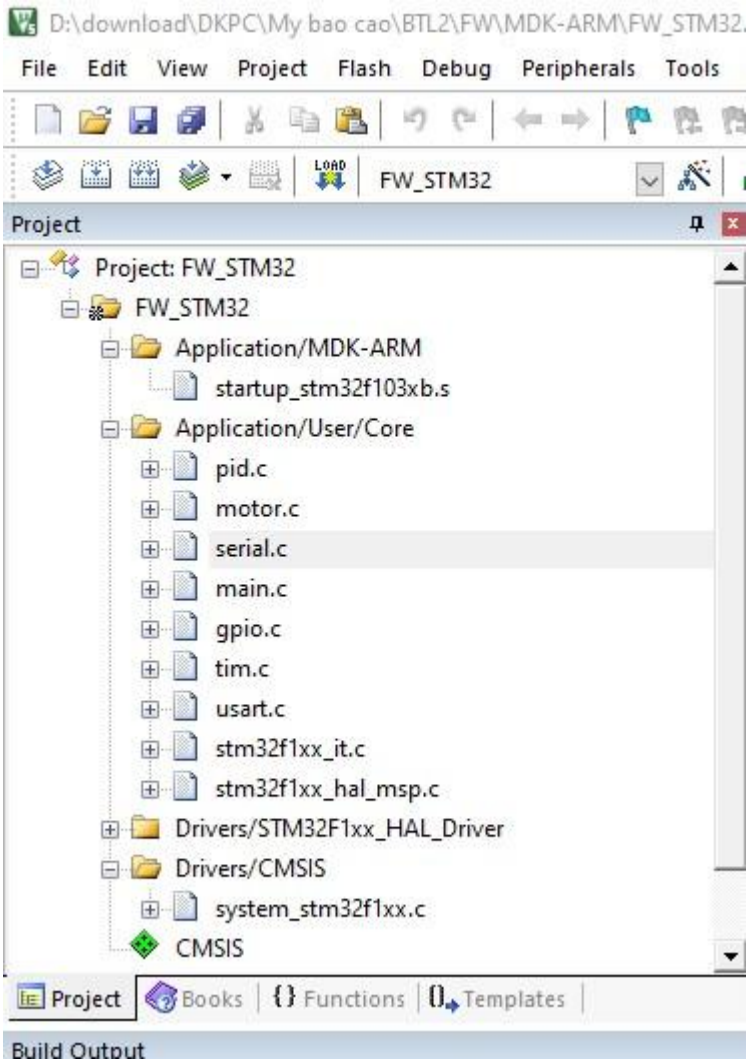




Counter Clockwise Rotation

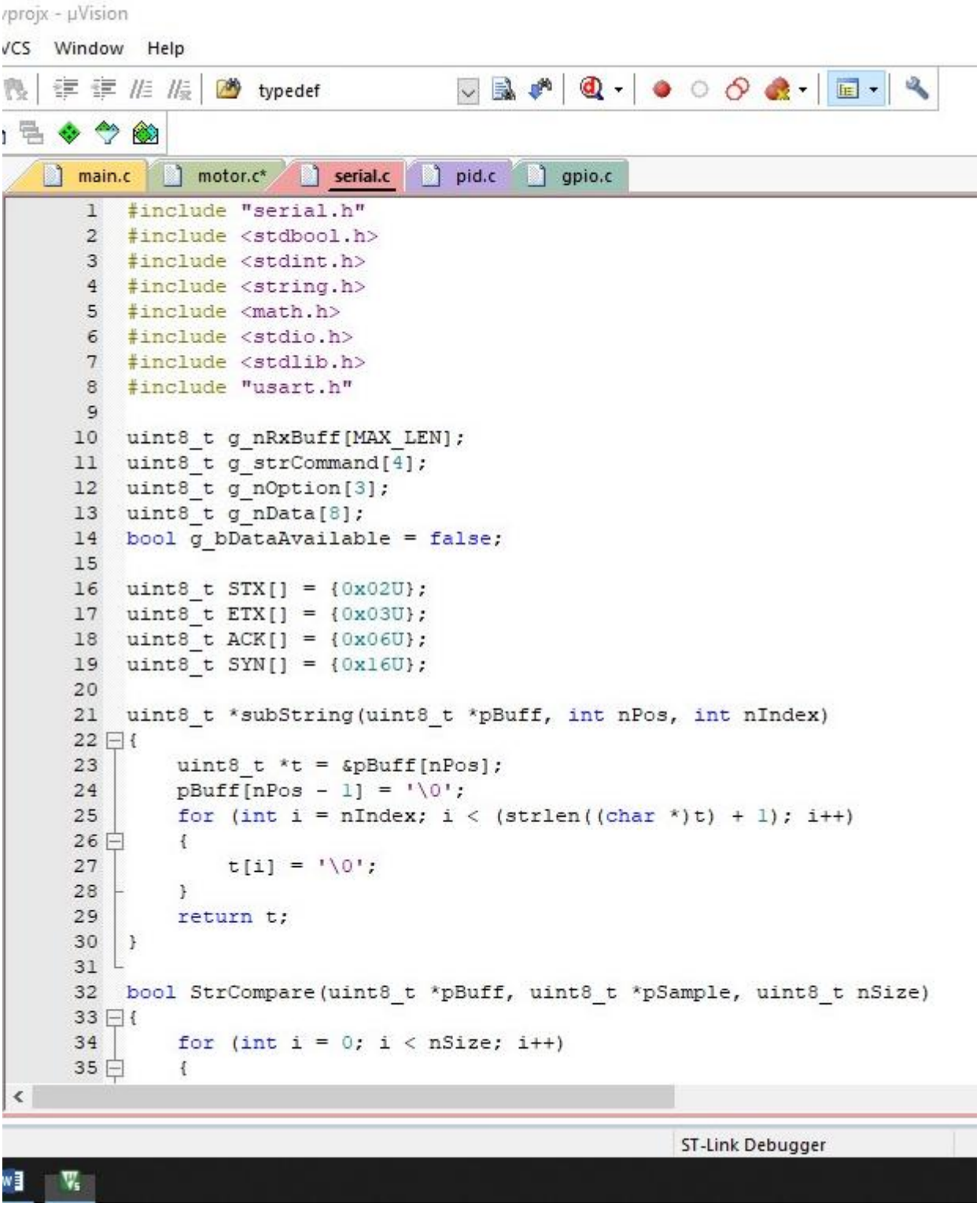


2.2 Keil C



Các hàm để điều khiển động cơ.

Module Serial.c



Trong serial them các thư viện cần thiết, khai báo biến

Hàm substring trả về chính xác ký tự trong chuỗi tại vị trí Pos và nhận index ký tự.

```
bool StrCompare(uint8_t *pBuff, uint8_t *pSample, uint8_t nSize)
{
    for (int i = 0; i < nSize; i++)
    {
        if (pBuff[i] != pSample[i])
        {
            return false;
        }
    }

    return true;
}
```

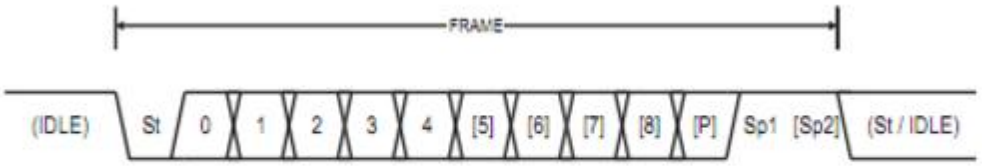
Thực hiện hàm so sánh 2 ký tự trong chuỗi. Nếu giống nhau thì return true

```
46  void SerialInit(void)
47  {
48      HAL_UART_Receive_IT(&huart1, (uint8_t *)g_nRxBuff, MAX_LEN);
49  }
50
51  void SerialAcceptReceive(void)
52  {
53      HAL_UART_Receive_IT(&huart1, (uint8_t *)g_nRxBuff, MAX_LEN);
54  }
55
```

Thực hiện ngắt nhận dữ liệu và nó được lưu trong biến g_nRxBuff

```
void SerialWriteComm(uint8_t *pStrCmd, uint8_t *pOpt, uint8_t *pData)
{
    uint8_t *pBuff;
    pBuff = (uint8_t *)malloc(18);
    uint8_t nIndex = 0;

    memcpy(pBuff + nIndex, STX, 1);
    nIndex += 1;
    memcpy(pBuff + nIndex, pStrCmd, 4);
    nIndex += 4;
    memcpy(pBuff + nIndex, pOpt, 3);
    nIndex += 3;
    memcpy(pBuff + nIndex, pData, 8);
    nIndex += 8;
    memcpy(pBuff + nIndex, ACK, 1);
    nIndex += 1;
    memcpy(pBuff + nIndex, ETX, 1);

    HAL_UART_Transmit(&huart1, pBuff, MAX_LEN, 1000);

    free(pBuff);
}
```

Viết hàm gửi data xuống serial port bằng protocol serial với frame truyền của nó



St	Start bit, always low.
(n)	Data bits (0 to 8).
P	Parity bit. Can be odd or even.
Sp	Stop bit, always high.
IDLE	No transfers on the communication line (RxDn or TxDn). An IDLE line must be high.

Trong module PID.c

```c
void PIDReset(PID_CONTROL_t *PID_Ctrl)
{
  PID_Ctrl->dIntergral = 0.0f;
  PID_Ctrl->dErrorTerm = 0.0f;
  g_dPIDError = 0;

}
void PIDInit(PID_CONTROL_t *PID_Ctrl, float dKp, float dKi, float dKd)
{
  PIDReset(PID_Ctrl);
  PID_Ctrl->dKp = dKp;
  PID_Ctrl->dKi = dKi;
  PID_Ctrl->dKd = dKp;
  __HAL_TIM_SetCounter(&htim4, 32768);
}
void PIDTuningSet(PID_CONTROL_t *PID_Ctrl, float dKp, float dKi, float dKd)
{
  // Check if the parameters are valid
  if(dKp < 0.0f || dKi < 0.0f || dKp < 0.0f)
  {
      return;
  }

  // Save the parameters for displaying purposes
  PID_Ctrl->dKp = dKp;
  PID_Ctrl->dKi = dKi;
  PID_Ctrl->dKd = dKd;
}
```

Có các hàm PID reset để reset thông số PID

PID unit khởi tạo giá trị mặc định PID và khởi động bộ đếm Encoder dùng timer4.

PID tunning để set các giá trị Kp Ki Kd

```c
float PIDCompute(PID_CONTROL_t *PID_Ctrl, float dCmdValue, float dActValue, float dTs)
{
  float dPIDResult;
  g_dPIDError = dCmdValue - dActValue;
  float dP = 0, dI = 0, dD = 0;

  dP = PID_Ctrl->dKp * g_dPIDError;
  PID_Ctrl->dIntergral += g_dPIDError;
  dI = PID_Ctrl->dKi * dTs / 2 * PID_Ctrl->dIntergral;
  dD = PID_Ctrl->dKd * (g_dPIDError - PID_Ctrl->dErrorTerm) / dTs;
  dPIDResult = dP + dI + dD;
  PID_Ctrl->dErrorTerm = g_dPIDError;

  return dPIDResult;
}
```

Tính toán PID

Tính sai số giữ góc thực và góc mong muốn điều khiển

Hệ số Kp nhân với sai số, Ki nhân với tích phân của sai số và Kd nhân với đạo hàm của sai số.

Module Motor.c

```
22  void MotorSetDir(int8_t nDir)
23  {
24    switch (nDir)
25    {
26    case 0: // CW
27      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_11, GPIO_PIN_RESET);
28      break;
29    case 1: // CCW
30      HAL_GPIO_WritePin(GPIOB, GPIO_PIN_11, GPIO_PIN_SET);
31      break;
32    default:
33      break;
34    }
35  }
36
37  void MotorSetDuty(uint16_t nDuty)
38  {
39    __HAL_TIM_SET_COMPARE(&htim3, TIM_CHANNEL_1, nDuty);
40  }
41
```

Viết các hàm MotorSetDir tức là set hướng của motor (quay bên phải hay bên trái)

Hàm setDuty là độ rộng băm xung điều khiển động cơ. PWM có độ rộng càng lớn thì tốc độ quay của motor càng cao và ngược lại

```
42  void MotorInit(void)
43  {
44    HAL_TIM_Base_Start_IT(&htim2);
45    HAL_TIM_PWM_Start(&htim3, TIM_CHANNEL_1);
46    HAL_TIM_Encoder_Start(&htim4, TIM_CHANNEL_1);
47    HAL_TIM_Encoder_Start(&htim4, TIM_CHANNEL_2);
48
49    PIDReset(&tPIDControl);
50    PIDInit(&tPIDControl, 1., 0., 0.00);
51    MotorSetDir(1);
52  }
53
```

Init motor với các thông số PID init và init ngắt timer, encoder ở timer 4 và timer pwm

```
54  uint16_t ConvertDegToPulse(uint16_t nDeg)
55  {
56    float dPulse = nDeg * 4 * 11 * 21.3 / 360;
57
58    return (uint16_t)dPulse;
59  }
60
61  uint16_t ConvertPulseToDeg(uint16_t nPulse)
62  {
63    float dDeg = nPulse * 360 / 4 / 11 / 21.3;
64    return (uint16_t)dDeg;
65  }
66
67  void MotorGetPulse(uint32_t *nPulse)
68  {
69    *nPulse = __HAL_TIM_GetCounter(&htim4);
70  }
```

Hàm viết sẵn chuyển từ giá trị từ độ về pulse và ngược lại

MotorGetPulse là lấy cái vị trí thực tế của encoder thông qua timer 4 mình đã set ở mode encoder

```
72  void MotorTuning(uint16_t nPos)
73  {
74    uint32_t nPulse;
75    MotorGetPulse(&nPulse);
76    g_nActPulse = nPulse - 32768;
77    g_nCmdPulse = ConvertDegToPulse(nPos);
78    g_nDutyCycle = (int16_t)PIDCompute(&tPIDControl, g_nCmdPulse, g_nActPulse, 0.01f);
79    if (g_nDutyCycle >= 0)
80    {
81      MotorSetDir(1);
82      MotorSetDuty(abs(g_nDutyCycle));
83    }
84    else if (g_nDutyCycle < 0)
85    {
86      MotorSetDir(0);
87      MotorSetDuty(abs(g_nDutyCycle));
88    }
89
90    // Store data
91    tPIDControl.nSampleTuningPID[g_nIndex] = g_nActPulse;
92    g_nIndex++;
93  }
94
```

Tunning động cơ, nhận sóng Pulse từ timer 4. Chuyển đổi góc mong muốn sang pulse.

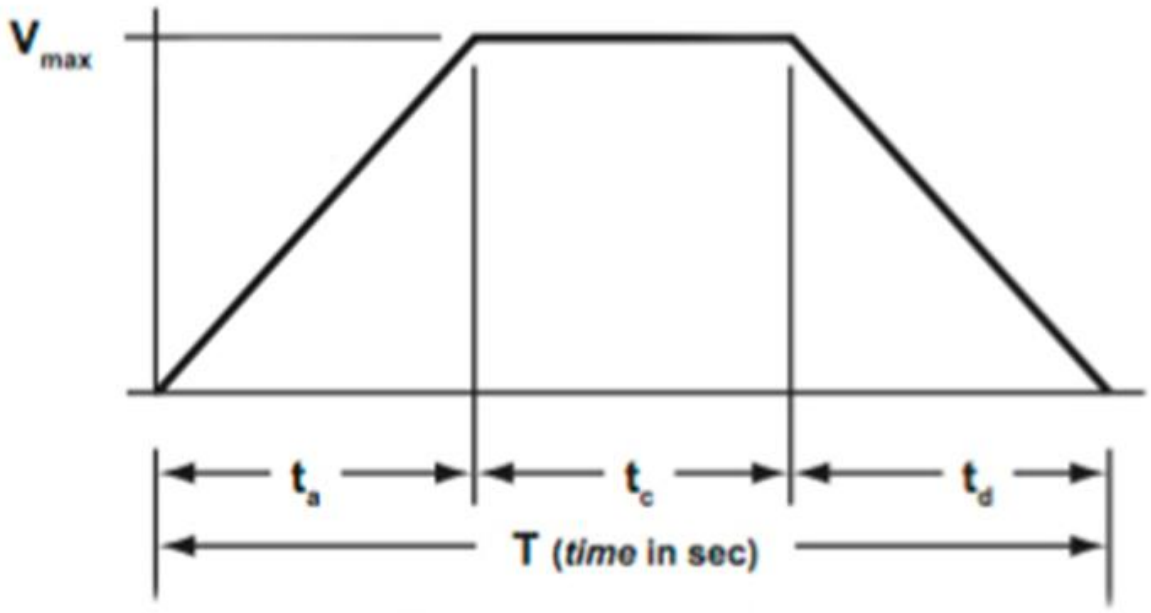
Set thông số Kp Ki Kd tính toán tín hiệu điều điều khiển (cần cân chỉnh) sao cho đáp ứng được thỏa mãn với kết quả mong đợi

```
95  void MotorMovePos(void)
96  {
97    uint32_t nPulse;
98    MotorGetPulse(&nPulse);
99    g_nActPulse = nPulse - 32768;
00    float dPosTemp = 0;
01
02    // Profile Trapezoidal Speed
03    if (tProfile.nTime <= tProfile.dMidStep1)
04    {
05      dPosTemp = (int32_t)(tProfile.dA1 * tProfile.nTime * tProfile.nTime);
06      g_dCmdVel = 2 * tProfile.dA1 * tProfile.nTime;
07    }
08    else if (tProfile.nTime <= tProfile.dMidStep2)
09    {
10      dPosTemp = (int32_t)(tProfile.dA2 * tProfile.nTime + tProfile.dB2);
11      g_dCmdVel = tProfile.dA2;
12    }
13    else if (tProfile.nTime <= tProfile.dMidStep3)
14    {
15      dPosTemp = (int32_t)(tProfile.dA3 * tProfile.nTime * tProfile.nTime + tProfile.dB3 * tProfile.nTime + tPro
16      g_dCmdVel = 2 * tProfile.dA3 * tProfile.nTime + tProfile.dB3;
17    }
18    else
19    {
20      dPosTemp = tProfile.dPosMax;
21    }
22
23    // Control PID
24    g_nCmdPulse = ConvertDegToPulse(dPosTemp);
25    g_nDutyCycle = (int16_t)PIDCompute(&tPIDControl, g_nCmdPulse, g_nActPulse, 0.01f);
26    if (g_nDutyCycle >= 0)
27    {
28      MotorSetDir(1);
```

ST-Link Debugger L:57 C:1 CAP NUM SCRL OVR R/W

ENG 9:52 SA

Điều khiển động cơ chạy với profile Trapezoidal



Trapezoidal Profile này có 3 giai đoạn là giai đoạn tăng tốc, vận tốc là hằng số và giảm tốc.

Nếu đưa gia tốc quá lơn thì hệ thống bị giật, rung rắc làm sao để cho nó hoạt động mượt mà.

Trong chương trình chính

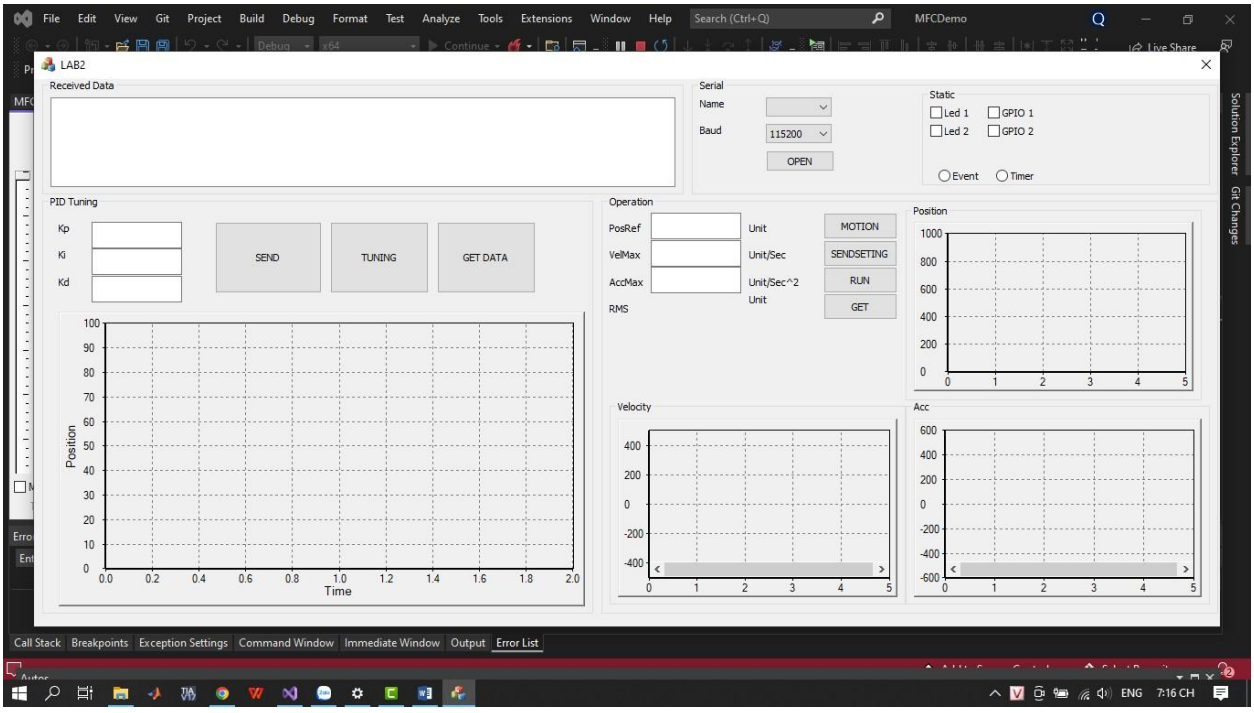
```
77    while (1)
78    {
79        if (g_bDataAvailable == true)
80      {
81        if (StrCompare(g_strCommand, (uint8_t *)"SPID", 4))
82        {
83          tProcess = SPID;
84        }
85        else if (StrCompare(g_strCommand, (uint8_t *)"CTUN", 4))
86        {
87          tProcess = CTUN_RES;
88        }
89        else if (StrCompare(g_strCommand, (uint8_t *)"GPID", 4))
90        {
91          tProcess = GPID;
92        }
93        else if (StrCompare(g_strCommand, (uint8_t *)"CSET", 4))
94        {
95          tProcess = CSET;
96        }
97        else if (StrCompare(g_strCommand, (uint8_t *)"CRUN", 4))
98        {
99          tProcess = CRUN_RES;
100       }
101       else if (StrCompare(g_strCommand, (uint8_t *)"GRMS", 4))
102       {
103         tProcess = GRMS;
104       }
105       else
106       {
107         tProcess = NONE;
```

```
109         g_bDataAvailable = false;
110       }
111       switch (tProcess)
112       {
113       case NONE:
114         SerialAcceptReceive();
115         break;
116       case SPID:
117         SerialWriteComm(g_strCommand, g_nOption, g_nData);
118         g_nCmdPulse = 0;
119         PIDReset(&tPIDControl);
120         __HAL_TIM_SetCounter(&htim4, 32768);
121         g_nIndex = 0;
122
123         /* Get PID params */
124         tPIDControl.dKp = (float)g_nData[0] + (float)g_nData[1] / 10;
125         tPIDControl.dKi = (float)g_nData[2] + (float)g_nData[3] / 10;
126         tPIDControl.dKd = (float)g_nData[4] + (float)g_nData[5] / (pow((float)10, (float)g_nData[6]));
127
128         tProcess = NONE;
129         break;
130       case CTUN_RES:
131         SerialWriteComm(g_strCommand, g_nOption, g_nData);
132         tProcess = CTUN;
133         break;
134       case CTUN:
135         break;
136       case GPID:
137         for (int index = 0; index < (g_nIndex - 1); index++)
138         {
139           sprintf((char *)g_strTxCommand, "%s", g_strCommand);
140           memset(g_nTxOption, '\0', 3);
141           g_nTxData[6] = (tPIDControl.nSampleTuningPID[index]&0xFF00) >> 8;
142           g_nTxData[7] = (tPIDControl.nSampleTuningPID[index]&0xFF);
143           g_nTxData[2] = (index&0xFF00) >> 8;
```

```
139           sprintf((char *)g_strTxCommand, "%s", g_strCommand);
140           memset(g_nTxOption, '\0', 3);
141           g_nTxData[6] = (tPIDControl.nSampleTuningPID[index]&0xFF00) >> 8;
142           g_nTxData[7] = (tPIDControl.nSampleTuningPID[index]&0xFF);
143           g_nTxData[2] = (index&0xFF00) >> 8;
144           g_nTxData[3] = (index&0xFF);
145           g_nTxData[0] = 0;
146           g_nTxData[1] = 199;
147
148           SerialWriteComm(g_strTxCommand, g_nTxOption, g_nTxData);
149           memset(g_strTxCommand, '\0', 4);
150           memset(g_nTxOption, '\0', 3);
151           memset(g_nTxData, '\0', 8);
152           HAL_Delay(40);
153         }
154         g_bDataAvailable = false;
155         SerialAcceptReceive();
156         tProcess = NONE;
157         break;
158       case CSET:
159         SerialWriteComm(g_strCommand, g_nOption, g_nData);
160
161         PIDReset(&tPIDControl);
162         g_nActPulse = 0;
163         g_nCmdPulse = 0;
164
165         // Get Pmax, Vmax, Amax
166         tProfile.dAccelMax = (float)((g_nData[2] >> 4) * 4096) + (float)((g_nData[2] & 0x0F) * 256) + (float)((g
167         tProfile.dVelMax = (float)((g_nData[4] >> 4) * 4096) + (float)((g_nData[4] & 0x0F) * 256) + (float)((g_nl
168         tProfile.dPosMax = (float)((g_nData[6] >> 4) * 4096) + (float)((g_nData[6] & 0x0F) * 256) + (float)((g_nl
169
170         // Calculate params for trapezoidal speed
171         tProfile.dA1 = 0.5f * tProfile.dAccelMax;
172         tProfile.dA2 = tProfile.dVelMax;
173         tProfile.dB2 = -0.5f * tProfile.dVelMax * tProfile.dVelMax / tProfile.dAccelMax;
```

```
175       tProfile.dB3 = tProfile.dPosMax * tProfile.dAccelMax / tProfile.dVelMax + tProfile.dVelMax;
176       tProfile.dC3 = -0.5f * tProfile.dPosMax * tProfile.dPosMax * tProfile.dAccelMax / (tProfile.dVelMax * tP
177       tProfile.dMidStep1 = tProfile.dVelMax / tProfile.dAccelMax;
178       tProfile.dMidStep2 = tProfile.dPosMax / tProfile.dVelMax;
179       tProfile.dMidStep3 = tProfile.dMidStep1 + tProfile.dMidStep2;
180
181       tProfile.nTime = 0;
182       tProcess = NONE;
183       break;
184     case CRUN_RES:
185       SerialWriteComm(g_strCommand, g_nOption, g_nData);
186       g_nCmdPulse = 0;
187       PIDReset(&tPIDControl);
188       __HAL_TIM_SetCounter(&htim4, 32768);
189       g_nIndex = 0;
190       tProcess = CRUN;
191       break;
192     case CRUN:
193       g_bDataAvailable = false;
194       SerialAcceptReceive();
195       break;
196     case GRMS:
197       for (int index = 0; index < (g_nIndex - 1); index++)
198       {
199         sprintf((char *)g_strTxCommand, "%s", g_strCommand);
200         memset(g_nTxOption, '\0', 3);
201         g_nTxData[6] = (tPIDControl.nActPosSample[index]&0xFF00) >> 8;
202         g_nTxData[7] = (tPIDControl.nActPosSample[index]&0xFF);
203         g_nTxData[4] = ((uint16_t)g_dPIDError&0xFF00) >> 8;
204         g_nTxData[5] = ((uint16_t)g_dPIDError&0xFF);
205         g_nTxData[2] = (index&0xFF00) >> 8;
206         g_nTxData[3] = (index&0xFF);
207         g_nTxData[0] = ((g_nIndex - 2)&0xFF00) >> 8;
208         g_nTxData[1] = ((g_nIndex - 2)&0xFF);
209
```

III. Software



Thiết lập giao diện như trên hình

```
void CMFCDemoDlg::OnCbnDropdownComboPort()
{
    // TODO: Add your control notification handler code here
    m_ccbSerialName.ResetContent();

    TCHAR lpTargetPath[5000]; // buffer to store the path of the COMPORTS
    bool gotPort = false; // in case the port is not found

    for (int i = 0; i < 255; i++) // checking ports from COM0 to COM255
    {
        CString str;
        str.Format(_T("%d"), i);
        CString ComName = CString("COM") + str; // converting to COM0, COM1, COM2


        DWORD test = QueryDosDevice(ComName, static_cast<LPWSTR>(lpTargetPath), 2000);

        // Test the return value and error if any
        if (test != 0) //QueryDosDevice returns zero if it didn't find an object
        {
            m_ccbSerialName.AddString(static_cast<CString>(ComName)); // add to the ComboBox
            gotPort = true; // found port
        }

        if (::GetLastError() == ERROR_INSUFFICIENT_BUFFER)
        {
            //lpTargetPath[2000]; // in case the buffer got filled, increase size of the buffer.
            continue;
        }

    }

    if (!gotPort) // if not port
        m_ccbSerialName.AddString(static_cast<CString>("No Active Ports Found")); // to display error message incase no ports found
}
```

Để thực hiện kết nối người dùng cần thiết lập chọn cổng COM serial nào để kết nối.

Tương tự vậy tạo ta tạo ra combo box baudrate (tốc độ truyền).

Nút nhấn button ta tạo event để kiểm tra nó open chưa, nếu chưa thì mình thực hiện open:

```
void CMFCDemoDlg::OnBnClickedButtonOpen()
{
    // TODO: Add your control notification handler code here
    if (bPortOpened == FALSE)
    {
        //DCB dcb;
        CString csPortName;
        CString csBaudRate;

        m_ccbSerialName.GetLBText(m_ccbSerialName.GetCurSel(), csPortName);
        m_ccbBaudrate.GetLBText(m_ccbBaudrate.GetCurSel(), csBaudRate);



        OpenPort(csPortName, csBaudRate);

    }
    else
    {
        ClosePort();

    }
}
```

Hàm onInitDialog: khởi tạo thông số ban đầu, hàm này được khởi tạo ra sẵn bởi MFC

```
BOOL CMFCDemoDlg::OnInitDialog()
{
    CDialogEx::OnInitDialog();

    // Set the icon for this dialog.  The framework does this automatically
    //  when the application's main window is not a dialog
    SetIcon(m_hIcon, TRUE); // Set big icon
    SetIcon(m_hIcon, FALSE); // Set small icon

    // TODO: Add extra initialization here

    m_ccbBaudrate.InsertString(0, static_cast<CString>("4800"));
    m_ccbBaudrate.InsertString(1, static_cast<CString>("9600"));
    m_ccbBaudrate.InsertString(2, static_cast<CString>("19200"));
    m_ccbBaudrate.InsertString(3, static_cast<CString>("115200"));
    m_ccbBaudrate.SetCurSel(3);



    for (int i = 0; i < 1000; i++)
    {
        if (i == 0)
        {
            dPIDPlotTime[i] = 0.01;
        }
        else dPIDPlotTime[i] = dPIDPlotTime[i - 1] + 0.01;
    }

    CChartCtrl ref;
    ref.RemoveAllSeries();
    pCharCtrlPID.EnableRefresh(true);
    pBottomAxis = pCharCtrlPID.CreateStandardAxis(CChartCtrl::BottomAxis);
    pLeftAxis = pCharCtrlPID.CreateStandardAxis(CChartCtrl::LeftAxis);
    pBottomAxis->SetMinMax(0, 2);
    pLeftAxis->SetMinMax(0, 100);
```

Send button.

Khi người dùng kích vào nút này thì việc đầu tiên truyền dữ liệu PID xuống slay sử dụng hàm lấy giá trị trong edit box: GetDlgItemDouble

Sau đó gán vào biến Kp Ki Kd

Chuẩn bị frame truyền biến các giá trị Kp Ki Kd thành cái chuỗi bằng lệnh FloatToByteArray.

Setup frame truyền bằng lệnh Memcpy.

Frame bắt đầu bởi **STX + Command+ option (byte rỗng)+Data + ETX**

Sau đó sử dụng hàm write để gửi dữ liệu ra ngoài

```
void CMFCDemoDlg::OnBnClickedButtonSend()
{
    // TODO: Add your control notification handler code here
    // TODO: Add your control notification handler code here
    CString strData;
    //DWORD dwRetCode;
    memset(bDATA, '\0', 8);

    double dKp = GetDlgItemDouble(IDC_EDIT_KP);
    double dKi = GetDlgItemDouble(IDC_EDIT_KI);
    double dKd = GetDlgItemDouble(IDC_EDIT_KD);
    // TODO: Add your control notification handler code here
    BYTE bProtocol[50] = {};
    UINT index = 0;

    uint8_t nLengTithes;
    BYTE bKp[2], bKi[2], bKd[3];

    FloatToByteArray(dKp, bKp, &nLengTithes);
    bDATA[0] = bKp[0];
    bDATA[1] = bKp[1];

    FloatToByteArray(dKi, bKi, &nLengTithes);
    bDATA[2] = bKi[0];
    bDATA[3] = bKi[1];

    FloatToByteArrayWithNipes(dKd, bKd, &nLengTithes);
    bKd[2] = nLengTithes;
    bDATA[4] = bKd[0];
    bDATA[5] = bKd[1];
    bDATA[6] = bKd[2];

    if (!GetPortActivateValue()) return;


    memcpy(bProtocol + index, bSTX, sizeof(bSTX));
    index += sizeof(bSTX);
```

```
memcpy(bProtocol + index, bSTX, sizeof(bSTX));
index += sizeof(bSTX);
memcpy(bProtocol + index, bSPID, sizeof(bSPID));
index += sizeof(bSPID);
memcpy(bProtocol + index, bOPT, sizeof(bOPT));
index += sizeof(bOPT);
memcpy(bProtocol + index, bDATA, sizeof(bDATA));
index += sizeof(bDATA);
memcpy(bProtocol + index, bSYNC, sizeof(bSYNC));
index += sizeof(bSYNC);
memcpy(bProtocol + index, bETX, sizeof(bETX));
index += sizeof(bETX);
Write((char*)bProtocol, index);

//Show on text box
CString cmd;
cmd.Format(static_cast<CString>("CMD: "));
m_listboxRead.InsertString(0, cmd);
cmd.Empty();
for (UINT i = 0; i < index; i++) {

    cmd.AppendFormat(static_cast<CString>("%02X "), bProtocol[i]);

}
m_listboxRead.InsertString(0, cmd);
```

Tunning button :

Tương tự vậy ta sử dụng event bởi button click

Setup protocol

1byte STX, 4 byte Command, 3 byte option, 4 byte data, 1 byte Syn, 1 byte ETX kết thúc frame

Sau đó sử dụng hàm write. (Trong module serial.c có sẵn hàm write này rồi, ta chỉ cần gọi nó ra)

```
void CMFCDemoDlg::OnBnClickedButtonTuning()
{
    BYTE bProtocol[50] = {};
    UINT index = 0;
    // TODO: Add your control notification handler code here
    if (!GetPortActivateValue()) return;
    memset(bDATA, '\0', 8);

    memcpy(bProtocol + index, bSTX, sizeof(bSTX));
    index += sizeof(bSTX);
    memcpy(bProtocol + index, bCTUN, sizeof(bCTUN));
    index += sizeof(bCTUN);
    memcpy(bProtocol + index, bOPT, sizeof(bOPT));
    index += sizeof(bOPT);
    memcpy(bProtocol + index, bDATA, sizeof(bDATA));
    index += sizeof(bDATA);
    memcpy(bProtocol + index, bSYNC, sizeof(bSYNC));
    index += sizeof(bSYNC);
    memcpy(bProtocol + index, bETX, sizeof(bETX));
    index += sizeof(bETX);
    Write((char*)bProtocol, index);

    //Show on text box
    CString cmd;
    cmd.Format(static_cast<CString>("CMD: "));
    m_listboxRead.InsertString(0, cmd);
    cmd.Empty();
    for (UINT i = 0; i < index; i++) {

        cmd.AppendFormat(static_cast<CString>("%02X "), bProtocol[i]);

    }
    m_listboxRead.InsertString(0, cmd);
}
```

Request button:

Nút bấm nhận dữ liệu, đầu tiên mình tạo protocol get data và truyền xuống.

Nhận dữ liệu thì ta sử dụng

```
void CMFCDemoDlg::OnBnClickedButtonRequestPid()
{
    BYTE bProtocol[50] = {};
    UINT index = 0;
    // TODO: Add your control notification handler code here
    memset(bDATA, '\0', 8);
    if (!GetPortActivateValue()) return;


    memcpy(bProtocol + index, bSTX, sizeof(bSTX));
    index += sizeof(bSTX);
    memcpy(bProtocol + index, bGPID, sizeof(bGPID));
    index += sizeof(bGPID);
    memcpy(bProtocol + index, bOPT, sizeof(bOPT));
    index += sizeof(bOPT);
    memcpy(bProtocol + index, bDATA, sizeof(bDATA));
    index += sizeof(bDATA);
    memcpy(bProtocol + index, bSYNC, sizeof(bSYNC));
    index += sizeof(bSYNC);
    memcpy(bProtocol + index, bETX, sizeof(bETX));
    index += sizeof(bETX);
    Write((char*)bProtocol, index);

    //Show on text box
    CString cmd;

    for (UINT i = 0; i < index; i++) {

        cmd.AppendFormat(static_cast<CString>("%02X "), bProtocol[i]);

    }
    m_listboxRead.InsertString(0, cmd);
}
```
;

Sử dùng hàm DrawPIDGraph.

Hàm Addpoint để vẽ ctừng điểm dữ liệu

Kết quả nhận được cái đồ thị đáp ứng của động cơ khi động cơ nó vận hành

```
VOID CMFCDemoDlg::DrawPIDGraph(){
    pChartPIDSeries->ClearSerie();
    for(int i = 0; i < nLenGraphPID;i++){
        pChartPIDSeries->AddPoint(dPIDPlotTime[i],dPIDPlotData[i]);
    }
}
VOID CMFCDemoDlg::DrawOperationGraph() {
    pChartPosSeriesRef = pChartCtrlPos.CreateLineSerie();
    pChartPosSeriesRef->SetColor(RGB(255, 0, 0));
    pChartPosSeries->SetWidth(3); //line width
    pChartPosSeriesRef->ClearSerie();

    for (int i = 0; i < nLengtPositionPlotData; i++) {
        pChartPosSeriesRef->AddPoint(dPIDPlotTime[i], fPositionPlotData[i]);
    }
```

ProcessData:

Nhận data và vẽ

Được đặt trong interrupt.

Trong đó nó có câu lệnh kiểm tra có phải thực hiện các lệnh. Ví dụ GRMS thì nó gửi xuống slave và slave trả ngược lại protocol một chuỗi liên quan. Và máy tính sẽ xử lý cái chuỗi đó và lấy dữ liệu.

```
}
VOID CMFCDemoDlg::ProcessData(unsigned char* data, int inLength) {

    CString str;
    uint16_t nIndex = 0;
    uint16_t nPosition = 0;
    CString csCmd,csOption,csData;
    for(UINT i = 0; i < static_cast<UINT>(inLength); i++){
        bProtocolBuffer[i] = static_cast<BYTE>(data[i]);
    }
    for (UINT i = 1; i <= 4;i++) {
        csCmd.AppendChar(static_cast<char>(bProtocolBuffer[i]));
    }
    for (UINT i = 5; i <= 7; i++) {
        bProtocolOption[i - 5] = bProtocolBuffer[i];
    }
    for (UINT i = 8; i <= 15; i++) {
        bProtocolData[i - 8] = bProtocolBuffer[i];
    }

    if(!csCmd.Compare (static_cast<CString>("SPID"))){

    }
    if (!csCmd.Compare(static_cast<CString>("GPID"))) {
        //m_btnREQ
        nLenGraphPID = (bProtocolData[0] << 8) + bProtocolData[1];
        //TODO:
        nIndex = (bProtocolData[2] << 8) + bProtocolData[3];
        nPosition = (bProtocolData[6] << 8) + bProtocolData[7];
        dPIDPlotData[nIndex] = nPosition;
        if (nIndex == (bProtocolData[0] << 8) + bProtocolData[1]) {
            DrawPIDGraph();
        }
    }
    if (!csCmd.Compare(static_cast<CString>("CTUN"))) {
```

```
}
if (!csCmd.Compare(static_cast<CString>("CTUN"))) {
    if (bProtocolBuffer[16] == 0x06) {
        //Receive SPID OK
    }
}
if (!csCmd.Compare(static_cast<CString>("CRUN"))) {
    if (bProtocolBuffer[16] == 0x06) {
        //Receive SPID OK
    }
}

if (!csCmd.Compare(static_cast<CString>("GRMS"))) {
    if (bProtocolBuffer[16] == 0x06) {
        //Receive SPID OK
    }
}
if (!csCmd.Compare(static_cast<CString>("CSET"))) {
    if (bProtocolBuffer[16] == 0x06) {
        //Receive SPID OK
    }
}
if (!csCmd.Compare(static_cast<CString>("GRMS"))) {
    //if (bProtocolBuffer[16] == 0x06) {
        //Receive SPID OK
    nLengtPositionPlotData = (bProtocolData[0] << 8) + bProtocolData[1];
    //if ((bProtocolData[2] << 8) + bProtocolData[3] >= nLengtPositionPlot
    nIndex = (bProtocolData[2] << 8) + bProtocolData[3];
    nPosition = (bProtocolData[6] << 8) + bProtocolData[7];

    fPositionPlotData[nIndex] = nPosition;

    if (nLengtPositionPlotData == nIndex) {
        DrawOperationGraph();
        uint16_t nRms = (bProtocolData[4] << 8) + bProtocolData[5];
```

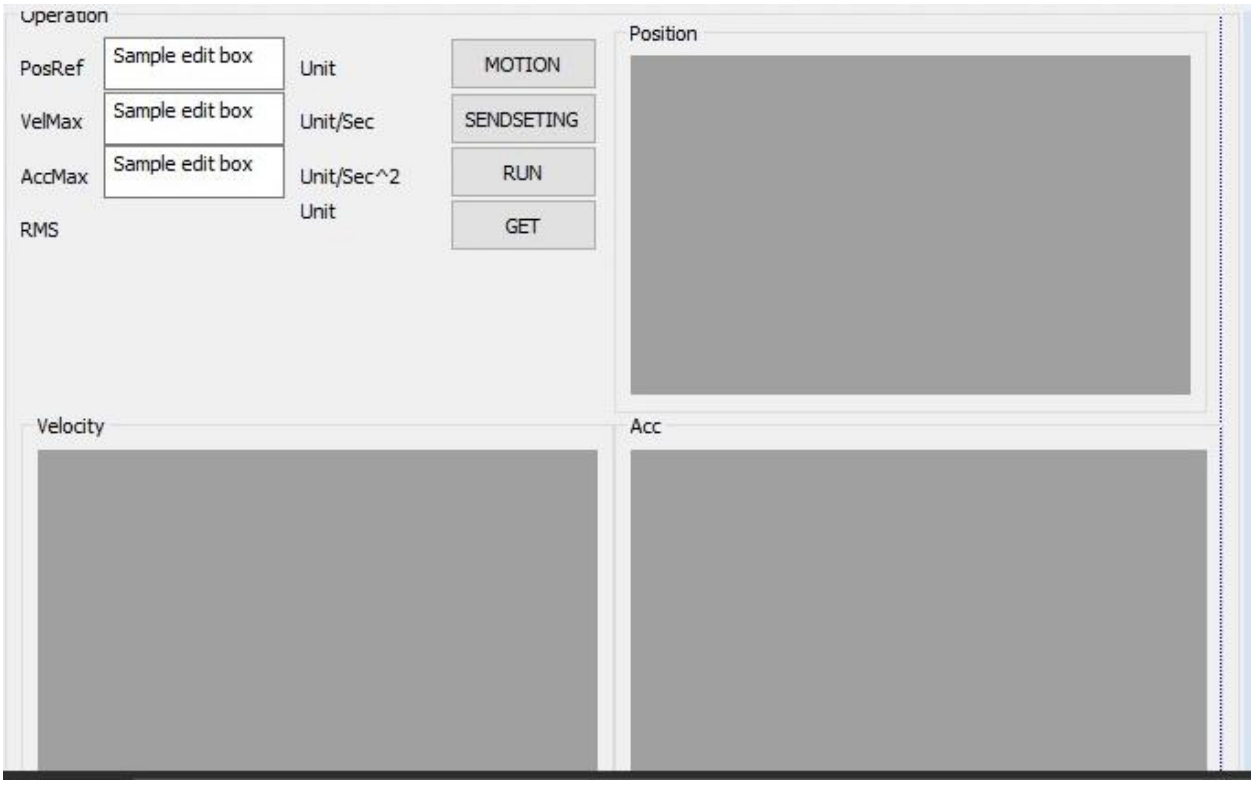
Control box:

Set các giá trị cho động cơ vận hành, chúng ta nhập vào giá trị vị trí động cơ, vận tốc max của động cơ, gia tốc max của động cơ.

Đơn vị unit là điều khiển theo dạng hình xung

Còn độ là degree.

RMS là sai số theo tiêu tiểu Euler (trung bình của căn bậc hai).



Motion button:

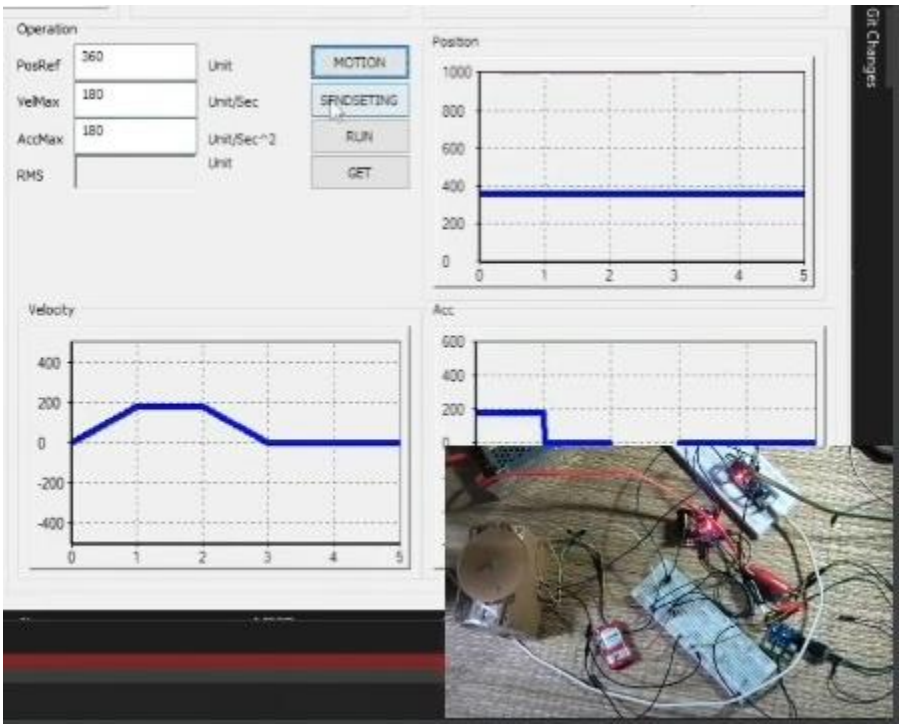
Vẽ lại mô phỏng lại cái profile trapezoidal

Nhận dữ liệu người dùng đưa vào là vị trí, vận tốc và gia tốc của động cơ.

Sau đó tạo ra hàm motion generator (hàm viết sẵn)

Vẽ lại cái dữ liệu sau tính toán.

```
void CMFCDemoDlg::OnBnClickedButtonDraw()
{
    double dPosRef = GetDlgItemDouble(IDC_EDIT_POS_MAX);
    double dVelMax = GetDlgItemDouble(IDC_EDIT_VELMAX);
    double dAccMax = GetDlgItemDouble(IDC_EDIT_ACC_MAX);

    MotionGenerator* trapezoidalProfile = new MotionGenerator(static_cast<float>(dVelMax), static_cast<float>(dAccMax),
                                                              0);

    float fCalPosition[200], fCalVel[200], fCalAcc[200];
    // Retrieve calculated position
    float positionRef = dPosRef;
    double positionRefC[200];
    //float position = trapezoidalProfile->update(positionRef);
    //float velocity, acceleration;

    double fTime[200];
    for (int i = 0; i < 200; i++){
        positionRefC[i] = dPosRef;
        if (i == 0) {
            fTime[i] = 0.025;
        }
        else fTime[i] = fTime[i - 1] + 0.025;
    }
    for(int i = 0; i < 200;i++){

        fCalPosition[i] = trapezoidalProfile->update(positionRefC[i], fTime[i]);  // NOLINT(bug prone-narrowing-conversions)
        // Check if profile is finished
        fCalVel[i] = trapezoidalProfile->getVelocity();

        //// Retrieve current acceleration
        fCalAcc[i] = trapezoidalProfile->getAcceleration();
```



Vận tốc gia tóc và vị trí

Send Setting button:

Gửi những giá trị vị trí vận tốc gia tốc này xuống cho slave.

```
void CMFCDemoDlg::OnBnClickedButtonSendpos()
{
    BYTE bProtocol[50] = {};
    UINT index = 0;
    memset(bDATA, '\0', 8);

    double dPosRef = GetDlgItemDouble(IDC_EDIT_POS_MAX);
    double dVelMax = GetDlgItemDouble(IDC_EDIT_VELMAX);
    double dAccMax = GetDlgItemDouble(IDC_EDIT_ACC_MAX);

    //dPosRef =
    bDATA[2] = (static_cast<UINT16>(dAccMax) & 0xFF00) >> 8;
    bDATA[3] = static_cast<UINT16>(dAccMax) & 0xFF;
    bDATA[4] = (static_cast<UINT16>(dVelMax) & 0xFF00) >> 8;
    bDATA[5] = static_cast<UINT16>(dVelMax) & 0xFF;
    bDATA[6] = (static_cast<UINT16>(dPosRef) & 0xFF00) >> 8;
    bDATA[7] = static_cast<UINT16>(dPosRef) & 0xFF;
    // TODO: Add your control notification handler code here
    if (!GetPortActivateValue()) return;


    memcpy(bProtocol + index, bSTX, sizeof(bSTX));
    index += sizeof(bSTX);
    memcpy(bProtocol + index, bCSET, sizeof(bCSET));
    index += sizeof(bCSET);
    memcpy(bProtocol + index, bOPT, sizeof(bOPT));
    index += sizeof(bOPT);
    memcpy(bProtocol + index, bDATA, sizeof(bDATA));
    index += sizeof(bDATA);
    memcpy(bProtocol + index, bSYNC, sizeof(bSYNC));
    index += sizeof(bSYNC);
    memcpy(bProtocol + index, bETX, sizeof(bETX));
    index += sizeof(bETX);
    Write((char*)bProtocol, index);
```

Tạo ra protocol rồi truyền nguyên frame đó xuống slave

Data mình có 8 byte thì sử dụng byte thứ 2 đến 7 để truyền .Byte 2 byte 3 là gửi gia tốc, byte 4 5 là vận tốc max và 2 byte còn lại là vị trí.

Run button

Tạo ra protocol truyền lênh run này xuống. Và hệ thống sẽ được hoạt động

```
void CMFCDemoDlg::OnBnClickedButtonOprun()
{
    BYTE bProtocol[50] = {};
    UINT index = 0;

    memset(bDATA, '\0', 8);
    // TODO: Add your control notification handler code here
    if (!GetPortActivateValue()) return;


    memcpy(bProtocol + index, bSTX, sizeof(bSTX));
    index += sizeof(bSTX);
    memcpy(bProtocol + index, bCRUN, sizeof(bCRUN));
    index += sizeof(bCRUN);
    memcpy(bProtocol + index, bOPT, sizeof(bOPT));
    index += sizeof(bOPT);
    memcpy(bProtocol + index, bDATA, sizeof(bDATA));
    index += sizeof(bDATA);
    memcpy(bProtocol + index, bSYNC, sizeof(bSYNC));
    index += sizeof(bSYNC);
    memcpy(bProtocol + index, bETX, sizeof(bETX));
    index += sizeof(bETX);
    Write((char*)bProtocol, index);

    //Show on text box
    CString cmd;
    cmd.Format(static_cast<CString>("CMD: "));
    m_listboxRead.InsertString(0, cmd);
    cmd.Empty();
    for (UINT i = 0; i < index; i++) {

        cmd.AppendFormat(static_cast<CString>("%02X "), bProtocol[i]);

    }
    m_listboxRead.InsertString(0, cmd);
}
```

Get button:

Nút này để nhận RMS (sai số)

Truyền xuống yêu cầu để board gửi lên giá trị RMS

```
void CMFCDemoDlg::OnBnClickedButtonGetRms()
{
    BYTE bProtocol[50] = {};
    UINT index = 0;
    memset(bDATA, '\0', 8);
    // TODO: Add your control notification handler code here
    if (!GetPortActivateValue()) return;


    memcpy(bProtocol + index, bSTX, sizeof(bSTX));
    index += sizeof(bSTX);
    memcpy(bProtocol + index, bGRMS, sizeof(bGRMS));
    index += sizeof(bGRMS);
    memcpy(bProtocol + index, bOPT, sizeof(bOPT));
    index += sizeof(bOPT);
    memcpy(bProtocol + index, bDATA, sizeof(bDATA));
    index += sizeof(bDATA);
    memcpy(bProtocol + index, bSYNC, sizeof(bSYNC));
    index += sizeof(bSYNC);
    memcpy(bProtocol + index, bETX, sizeof(bETX));
    index += sizeof(bETX);
    Write((char*)bProtocol, index);

    CString cmd;
    cmd.Format(static_cast<CString>("CMD: "));
    m_listboxRead.InsertString(0, cmd);
    cmd.Empty();
    for (UINT i = 0; i < index; i++) {

        cmd.AppendFormat(static_cast<CString>("%02X "), bProtocol[i]);

    }
    m_listboxRead.InsertString(0, cmd);
}
```