

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH  
TRƯỜNG ĐẠI HỌC BÁCH KHOA



**BÁO CÁO BÀI TẬP LỚN**

**HỌC PHẦN: ĐO LƯỜNG VÀ ĐIỀU KHIỂN BẰNG MÁY TÍNH**

**ỨNG DỤNG VI ĐIỀU KHIỂN STM32 TRONG ĐO LƯỜNG VÀ ĐIỀU KHIỂN  
BẰNG CÁC GIAO TIẾP CHUẨN CÔNG NGHIỆP**

**LỚP L03 – HK242**

***GVHD: TS. Nguyễn Hoàng Giáp***

Sinh viên thực hiện	MSSV
Võ Quế Long	2211910

Thành phố Hồ Chí Minh – 2025

## NỘI DUNG

<b>A. TỔNG QUÁT ĐỀ TÀI .....</b>	<b>1</b>
<b>I. MỤC TIÊU.....</b>	<b>1</b>
<b>II. CHUẨN BỊ.....</b>	<b>3</b>
<b>III. PROTOCOL FRAME.....</b>	<b>7</b>
<b>B. NỘI DUNG .....</b>	<b>12</b>
<b>I. CONTROL LED.....</b>	<b>12</b>
<b>II. CONTROL MOTOR.....</b>	<b>17</b>
<b>III. DIGITAL INPUT &amp; DIGITAL OUTPUT.....</b>	<b>23</b>
<b>IV. ADC &amp; DAC.....</b>	<b>28</b>
<b>V. SPI COMMUNICATION.....</b>	<b>40</b>
<b>VI. I<sup>2</sup>C COMMUNICATION.....</b>	<b>44</b>
<b>VII. USB COMMUNICATION .....</b>	<b>47</b>
<b>VIII. EXTERNAL RAM .....</b>	<b>50</b>
<b>IX. 4 – 20mA OUTPUT SENSOR .....</b>	<b>54</b>

## A. TỔNG QUÁT ĐỀ TÀI

### I. MỤC TIÊU

#### 1.1. UART

Điều khiển LED sử dụng giao tiếp UART.

Điều khiển động cơ sử dụng giao tiếp UART.

#### 1.2. DI & DO

Sử dụng các GPIO Port của MCU làm DI và DO.

#### 1.3. ADC & DAC

Sử dụng bộ ADC có sẵn trong KIT Blue Pill để nhận tín hiệu được chuyển từ Analog sang Digital.

Sử dụng bộ DAC ngoại để chuyển tín hiệu Digital từ MCU sang tín hiệu Analog bên ngoài.

#### 1.4. I<sup>2</sup>C

Đọc và ghi dữ liệu trên EEPROM sử dụng giao tiếp I2C.

#### 1.5. SPI

Hiển thị giá trị trên LED 7 đoạn sử dụng giao tiếp SPI.

#### 1.6. USB

Truyền & nhận dữ liệu sử dụng giao tiếp USB.

Truy xuất VIP & PID của USB.

#### 1.7. External RAM

Ghi và đọc dữ liệu trên SRAM ngoại.

Kiểm tra kết quả bằng cách đọc data về từ SRAM sau đó xuất tín hiệu ra LED.

### **1.8. Cảm biến dòng 4 – 20mA**

Sử dụng Pt100 công nghiệp cho dòng ngõ ra 4 – 20mA.

Biến đổi tín hiệu dòng thành áp và sử dụng ADC/VOM để đo/kiểm chứng tín hiệu analog áp.

## II. CHUẨN BỊ

### 2.1. Phần cứng

Tên linh kiện	Số lượng	Chức năng
KIT STM32F103C8T6 Blue Pill	1	Vi điều khiển đóng vai trò CPU
GA25-370 DC Geared Motor	1	Động cơ DC tạo chuyển động cơ học với moment lớn
L298 DC Motor Driver	1	Điều khiển chiều quay và tốc độ động cơ
ST-Link V2	1	Nạp chương trình cho STM32
CP2102 Serial UART Bridge	1	Giao tiếp nối tiếp với PC qua cổng USB
MCP4725 DAC Breakout Board	1	Chuyển đổi tín hiệu số sang analog
AT24C256 EEPROM	1	Bộ nhớ ngoài lưu dữ liệu khi mất điện
MAX7219 LED 7 Seg Module	1	Hiển thị số
TLP281 4 Channels Optocoupler	1	Cách ly bảo vệ mạch
CY6264-70SNX IC SRAM	1	Bộ nhớ lưu giữ liệu tạm thời

**2.2. Phần mềm:** Chủ yếu dùng hai phần mềm STM32 CubeMX và Keil C V.5



### III. PROTOCOL FRAME

Start Byte		Mode		Action				Reserved	Stop Byte	
0	0	0	0	0	0	0	0	0	1	1
				0	0	0	1			
				0	0	1	0			
				0	0	1	1			
		0	1	0	0	2	D			
				0	0	5	A			
				0	0	B	4			
				0	1	6	8			
		0	2	0	0	0	0			
				0	0	0	1			
		0	3	0	0	0	0			
				0	0	0	1			
				0	0	1	0			
				0	0	1	1			
		0	4	X	X	X	X			
				F	F	F	F			

Start Byte		Mode		Action				Reserved	Stop Byte	
0	0	0	5	0	0	0	1	0	1	1
				0	0	1	1			
		0	6	0	0	0	0			
				0	0	X	X			
		0	7	X	X	X	X			
				F	F	F	F			
		0	8	0	0	1	0			
				0	0	1	1			
		0	9	0	0	0	0			
				0	0	0	1			
		1	0	0	0	0	0			

## MODE DESCRIPTION

Descript	Mode	
Control LED	0	0
Control Motor Position	0	1
Active/Inactive Pulse Transfering (Draw Graph)	0	2
Active/Inactive Status Tracking	0	3
Control Motor Speed	0	4
Store LED Status & Motor Speed	0	5
DAC	0	6
Display On 7 Segment LED	0	7
Digital Output	0	8
ADC	0	9
External RAM	1	0

### **MODE 00 (CONTROL LED)**

Descript	Action Byte			
OFF LED 0	0	0	0	0
ON LED 0	0	0	0	1
OFF LED 1	0	0	1	0
ON LED 1	0	0	1	1

### **MODE 01 (CONTROL MOTOR POSITION)**

Descript	Action Byte			
Rotate 45D	0	0	2	D
Rotate 90D	0	0	5	A
Rotate 180D	0	0	B	4
Rotate 360D	0	1	6	8

### **MODE 02 (ACTIVE/INACTIVE PULSE TRANSFERING)**

Descript	Action Byte			
Inactive pulse transfering	0	0	0	0
Active pulse transfering	0	0	0	1

### **MODE 03 (ACTIVE/INACTIVE STATUS TRACKING)**

Descript	Action Byte			
Inactive LED status tracking	0	0	0	0
Active LED status tracking	0	0	0	1
Inactive motor position status tracking	0	0	1	0
Active motor position status tracking	0	0	1	1

### **MODE 04 (CONTROL MOTOR SPEED)**

Descript	Action Byte			
Control motor speed ( $000 \leq XXX \leq 100$ )	0	X	X	X
Reverse	1	0	0	0
Stop control motor speed	F	F	F	F

### **MODE 05 (STORE LED & MOTOR SPEED)**

Descript	Action Byte			
Store LED status	0	0	0	1
Store motor speed	0	0	1	1

### **MODE 06 (DAC)**

Descript	Action Byte			
Disable DAC	0	0	0	0
Output voltage ( $00 < XX \leq 30$ )	0	0	X	X

### **MODE 07 (DISPLAY ON 7 SEGMENT LED)**

Descript	Action Byte			
Active (XXXX # FFFF)	X	X	X	X
Inactive	F	F	F	F

### **MODE 08 (DIGITAL OUTPUT)**

Descript	Action Byte			
Low Ouput	0	0	1	0
High Ouput	0	0	1	1

### **MODE 09 (ADC)**

Descript	Action Byte			
Inactive	0	0	0	0
Active	0	0	0	1

### **MODE 10 (EXTERNAL RAM)**

Descript	Action Byte			
Check the parallel simulation	0	0	0	0

## B. NỘI DUNG

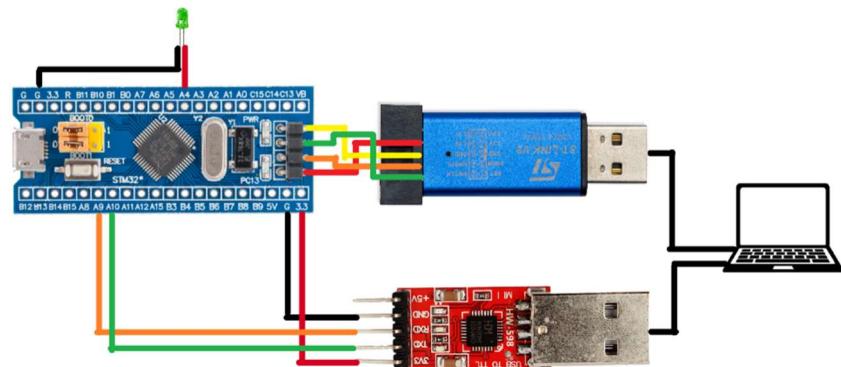
### I. CONTROL LED

#### 1.1. Tổng quát

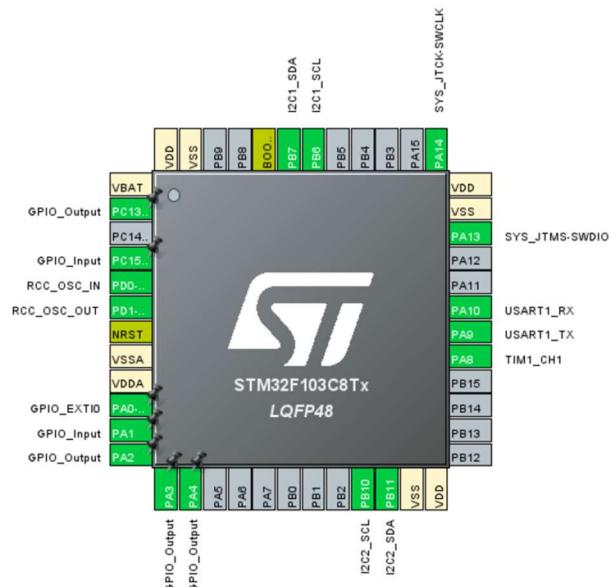
Sử dụng STM32 điều khiển trạng thái LED thông qua giao thức truyền thông UART.

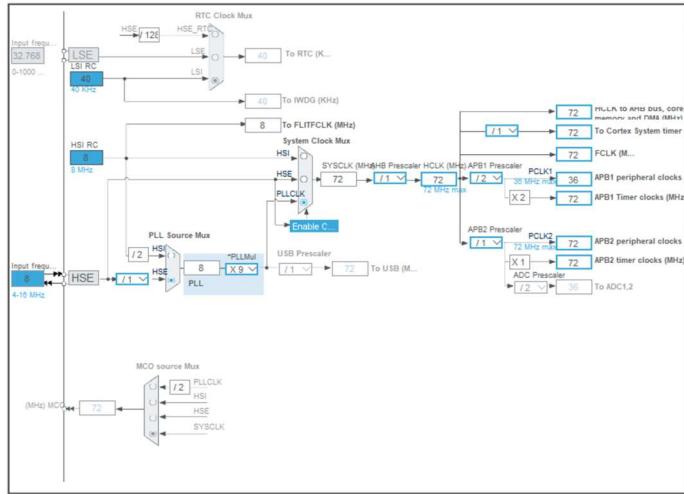
Theo dõi trạng thái hiện tại của LED theo thời gian thực.

#### 1.2. Kết nối phần cứng



#### 1.3. Cấu hình và lập trình phần mềm





	NVIC	Code generation	
	NVIC Interrupt Table		
	Enabled	Preemption Priority	Sub Priority
Non maskable interrupt	<input checked="" type="checkbox"/>	0	0
Hard fault interrupt	<input checked="" type="checkbox"/>	0	0
Memory management fault	<input checked="" type="checkbox"/>	0	0
Prefetch fault, memory access fault	<input checked="" type="checkbox"/>	0	0
Undefined instruction or illegal state	<input checked="" type="checkbox"/>	0	0
System service call via SWI instruction	<input checked="" type="checkbox"/>	0	0
Debug monitor	<input checked="" type="checkbox"/>	0	0
Pendable request for system service	<input checked="" type="checkbox"/>	0	0
Time base, System tick timer	<input checked="" type="checkbox"/>	15	0
PVD interrupt through EXTI line 16	<input type="checkbox"/>	0	0
Flash global interrupt	<input type="checkbox"/>	0	0
RCC global interrupt	<input type="checkbox"/>	0	0
EXTI line0 interrupt	<input checked="" type="checkbox"/>	0	0
DMA1 channel4 global interrupt	<input checked="" type="checkbox"/>	0	0
DMA1 channel5 global interrupt	<input checked="" type="checkbox"/>	0	0
TIM1 break interrupt	<input type="checkbox"/>	0	0
TIM1 update interrupt	<input type="checkbox"/>	0	0
TIM1 trigger and commutation interrupts	<input type="checkbox"/>	0	0
TIM1 capture compare interrupt	<input type="checkbox"/>	0	0
TIM2 global interrupt	<input checked="" type="checkbox"/>	0	0
TIM3 global interrupt	<input checked="" type="checkbox"/>	0	0
I2C1 event interrupt	<input type="checkbox"/>	0	0
I2C1 error interrupt	<input type="checkbox"/>	0	0
I2C2 event interrupt	<input type="checkbox"/>	0	0
I2C2 error interrupt	<input type="checkbox"/>	0	0
USART1 global interrupt	<input checked="" type="checkbox"/>	0	0

```
//MODE 00 - CONTROL LED/
if ((strncmp(Char_RX_Data, "0000", 4) == 0) && (strncmp(&Char_RX_Data[9], "1", 1) == 0))
{
    Mode_00_Flag = 1;

    if (strncmp(Char_RX_Data, "00000000011", sizeof(RX_Data)) == 0)
    {
        LED_1_Status_Flag = 0;
    }
    else if (strncmp(Char_RX_Data, "00000001011", sizeof(RX_Data)) == 0)
    {
        LED_1_Status_Flag = 1;
    }

    if (strncmp(Char_RX_Data, "00000010011", sizeof(RX_Data)) == 0)
    {
        LED_2_Status_Flag = 0;
    }
    else if (strncmp(Char_RX_Data, "00000011011", sizeof(RX_Data)) == 0)
    {
        LED_2_Status_Flag = 1;
    }
}
```

```

//=====CONTROL LED=====
//=====CONTROL LED=====
//=====CONTROL LED=====

void LED_Control ()
{
    Mode_00_Flag = 0;

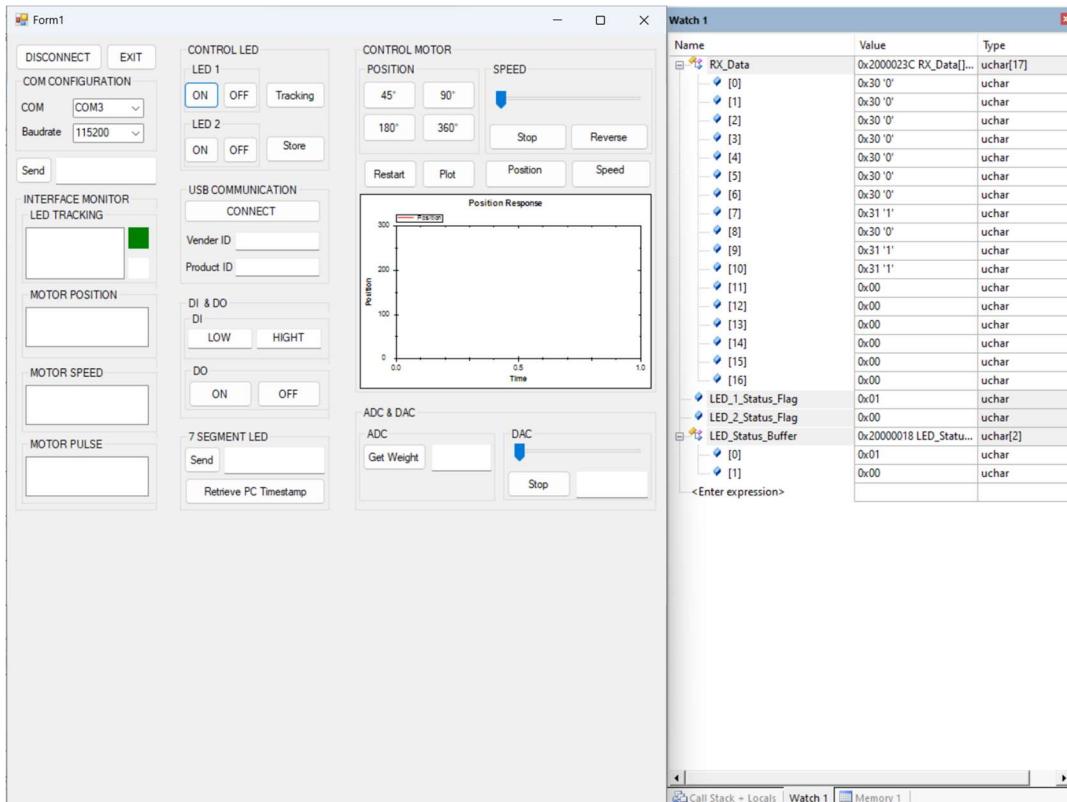
    if (LED_1_Status_Flag == 0)
    {
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, 1);
        LED_Status_Buffer [0] = 0;
    }
    else if (LED_1_Status_Flag == 1)
    {
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, 0);
        LED_Status_Buffer [0] = 1;
    }

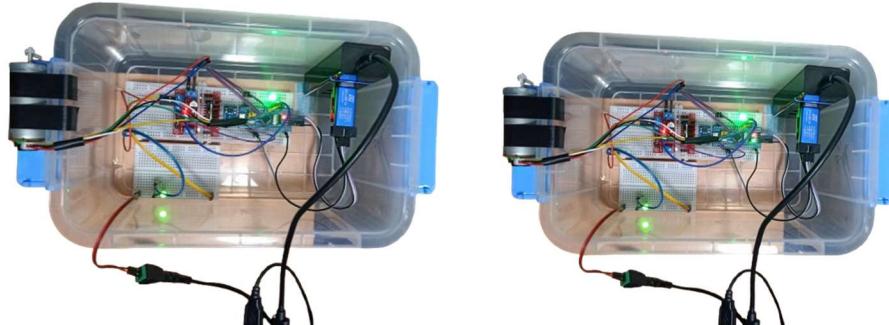
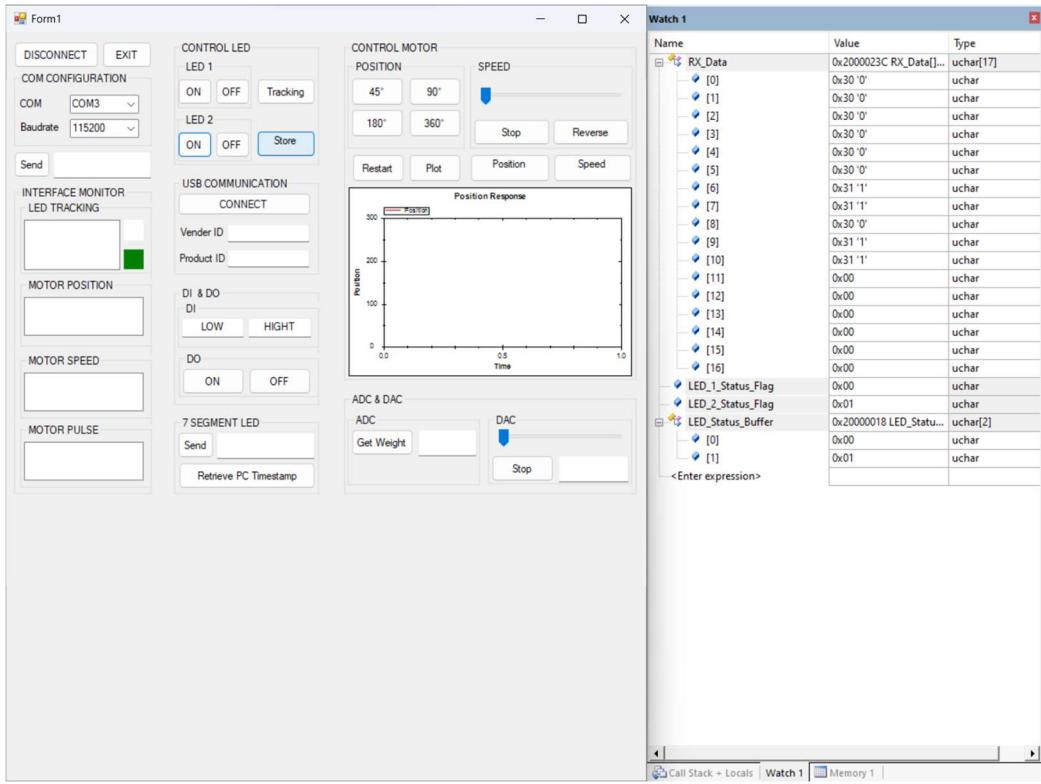
    if (LED_2_Status_Flag == 0)
    {
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_4, 0);
        LED_Status_Buffer [1] = 0;
    }
    else if (LED_2_Status_Flag == 1)
    {
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_4, 1);
        LED_Status_Buffer [1] = 1;
    }
}

```

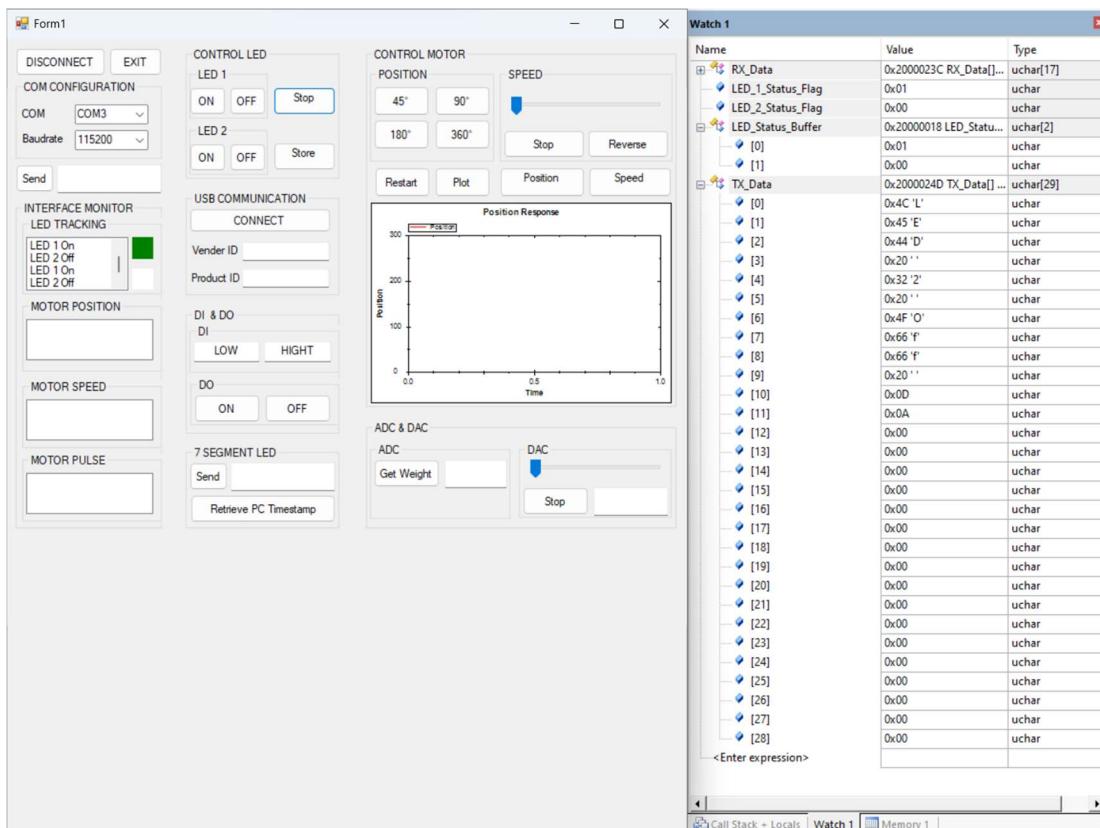
## 1.4. Kết quả

### 1.4.1. Điều khiển trạng thái LED





#### 1.4.2. Theo dõi trạng thái LED



## II. CONTROL MOTOR

### 2.1. Tổng quát

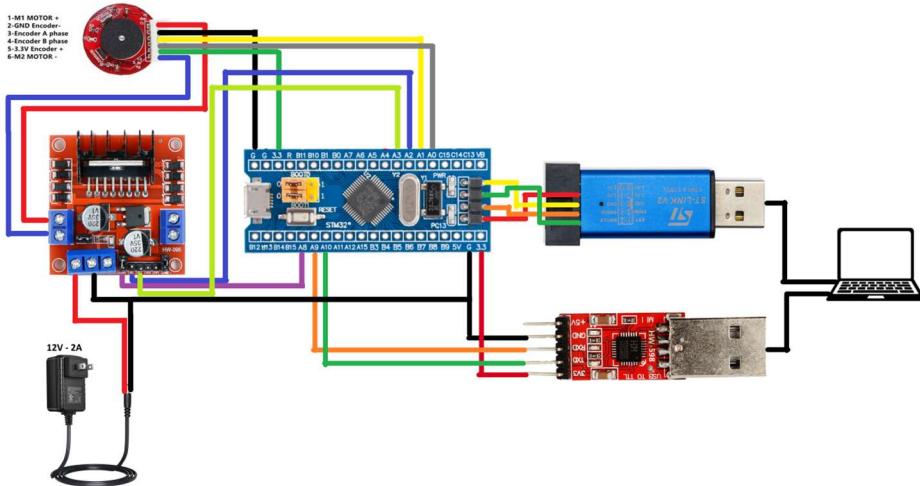
Điều khiển tốc độ của motor.

Điều khiển vị trí của motor.

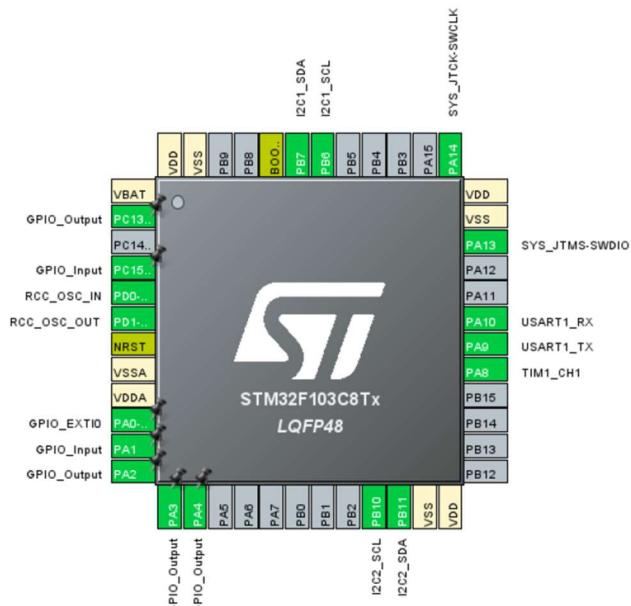
Theo dõi vận tốc của motor.

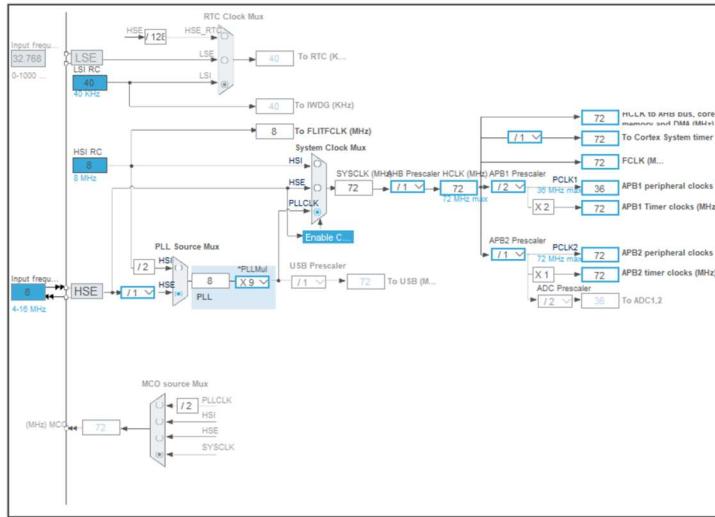
Theo dõi vị trí của motor

### 2.2. Kết nối phần cứng



### 2.3. Cấu hình và lập trình phần mềm





	Enabled	Preemption Priority	Sub Priority
Non maskable interrupt	<input checked="" type="checkbox"/>	0	0
Hard fault interrupt	<input checked="" type="checkbox"/>	0	0
Memory management fault	<input checked="" type="checkbox"/>	0	0
Prefetch fault, memory access fault	<input checked="" type="checkbox"/>	0	0
Undefined instruction or illegal state	<input checked="" type="checkbox"/>	0	0
System service call via SWI instruction	<input checked="" type="checkbox"/>	0	0
Debug monitor	<input checked="" type="checkbox"/>	0	0
Pendable request for system service	<input checked="" type="checkbox"/>	0	0
Time base System tick timer	<input checked="" type="checkbox"/>	15	0
PVD interrupt through EXTI line 16	<input type="checkbox"/>	0	0
Flash global interrupt	<input type="checkbox"/>	0	0
RCC global interrupt	<input type="checkbox"/>	0	0
EXTI line0 interrupt	<input checked="" type="checkbox"/>	0	0
DMA1 channel4 global interrupt	<input checked="" type="checkbox"/>	0	0
DMA1 channel5 global interrupt	<input checked="" type="checkbox"/>	0	0
TIM1 break interrupt	<input type="checkbox"/>	0	0
TIM1 update interrupt	<input type="checkbox"/>	0	0
TIM1 trigger and commutation interrupts	<input type="checkbox"/>	0	0
TIM1 capture compare interrupt	<input type="checkbox"/>	0	0
TIM2 global interrupt	<input checked="" type="checkbox"/>	0	0
TIM3 global interrupt	<input checked="" type="checkbox"/>	0	0
I2C1 event interrupt	<input type="checkbox"/>	0	0
I2C1 error interrupt	<input type="checkbox"/>	0	0
I2C2 event interrupt	<input type="checkbox"/>	0	0
I2C2 error interrupt	<input type="checkbox"/>	0	0
USART1 global interrupt	<input checked="" type="checkbox"/>	0	0

```
//MODE 01 - CONTROL MOTOR POSITION//
if ((strncmp(Char_RX_Data, "0001", 4) == 0) && (strncmp(&Char_RX_Data[9], "1", 1) == 0))
{
    Mode_01_Flag = 1;

    if (strncmp(Char_RX_Data, "0001002D011", sizeof(RX_Data)) == 0)
    {
        Forward_45D_Flag = 1;
    }

    if (strncmp(Char_RX_Data, "0001005A011", sizeof(RX_Data)) == 0)
    {
        Forward_90D_Flag = 1;
    }

    if (strncmp(Char_RX_Data, "000100B4011", sizeof(RX_Data)) == 0)
    {
        Forward_180D_Flag = 1;
    }

    if (strncmp(Char_RX_Data, "00010168011", sizeof(RX_Data)) == 0)
    {
        Forward_360D_Flag = 1;
    }
}
```

```

//=====CONTROL MOTOR=====
//=====CHANGE MOTOR POSITION=====
void Change_Motor_Position ()
{
//=====GENERATE PWM SIGNAL BY TIMER 1=====

void Generate_PWM (TIM_HandleTypeDef *htim, uint32_t Channel, float Duty_Cycle)
{
//=====CONTROL POSITION=====

void Control_Forward ()
{
void Control_Reverse ()
{
void Control_Stop ()
{
void Control_Position ()
{
//=====CONTROL MOTOR SPEED=====

void Control_Motor_Speed ()
{
//=====ENCODER PULSE COUNTING=====

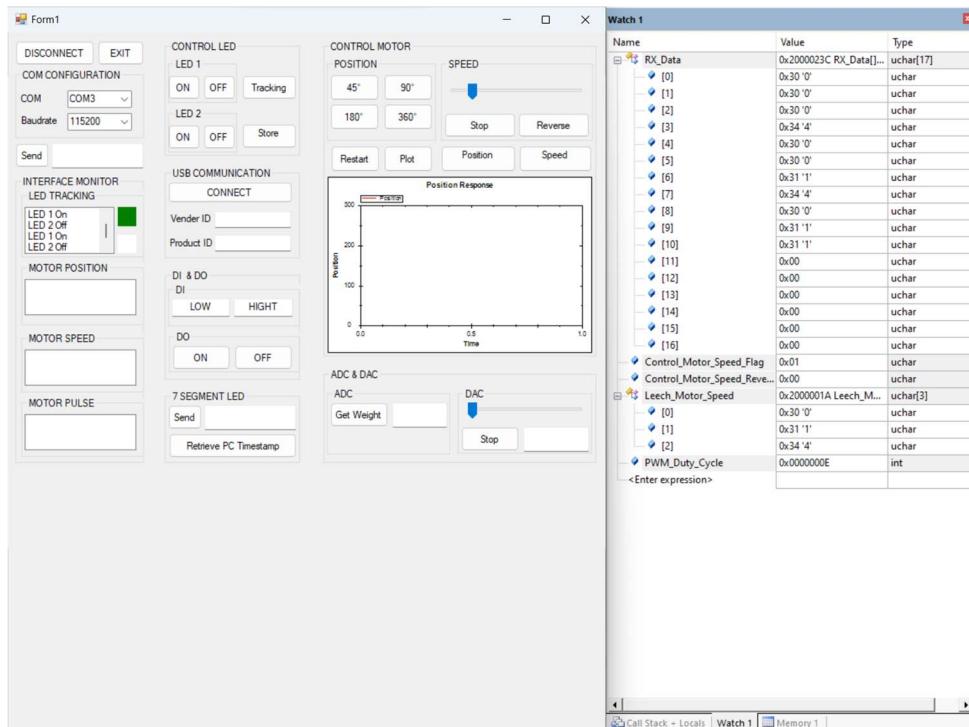
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
//=====PID CACULATION & ACTIVE TIMER INTERRUPT=====

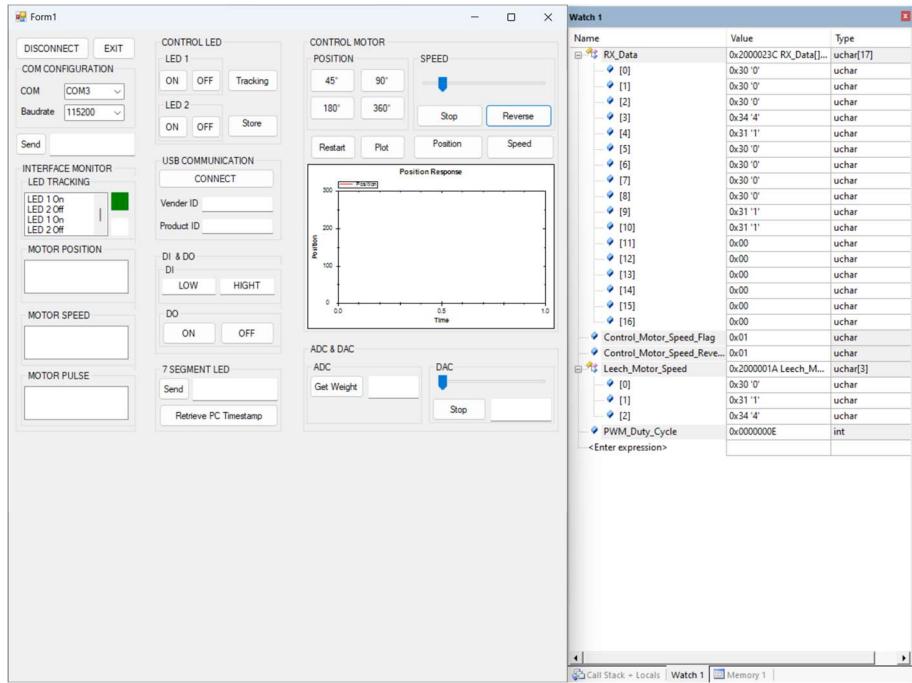
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{

```

## 2.4. Kết quả

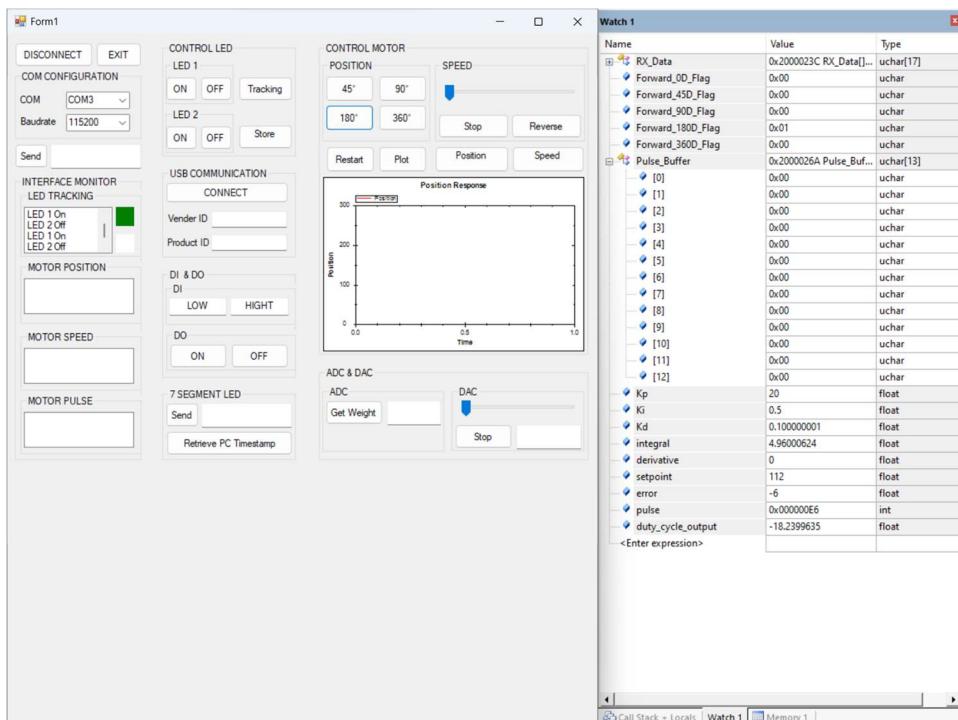
### 2.4.1. Điều khiển tốc độ động cơ

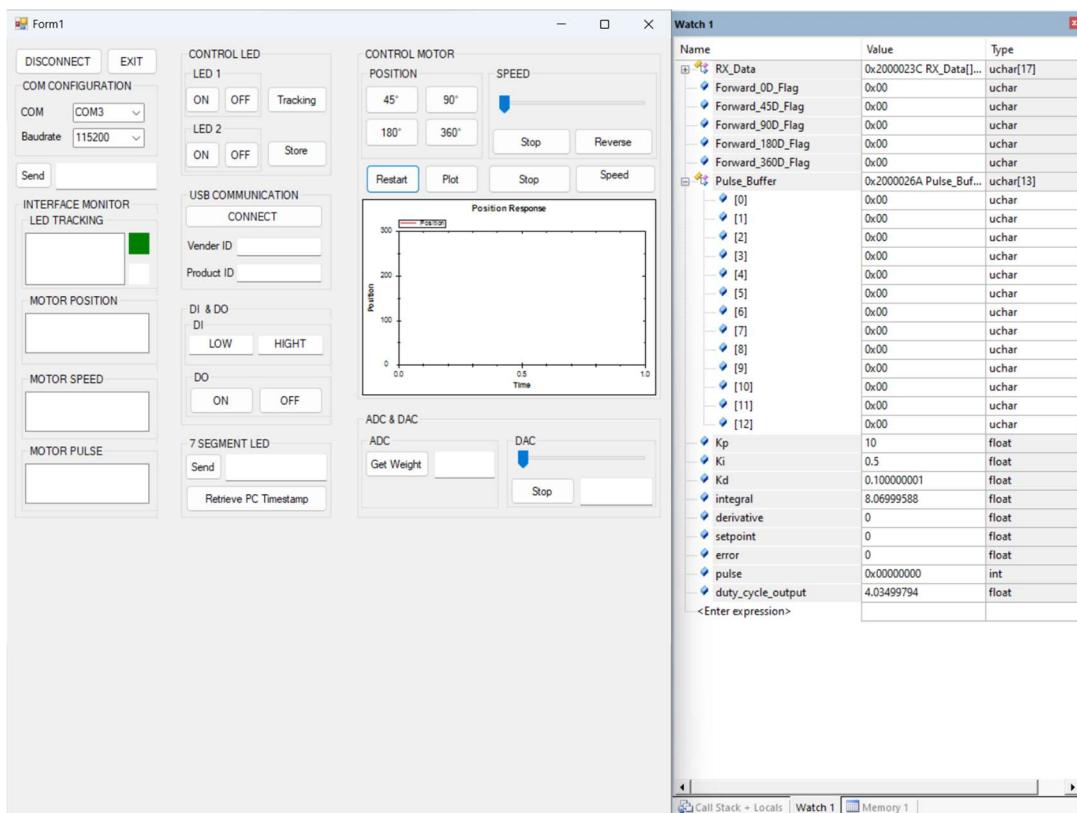
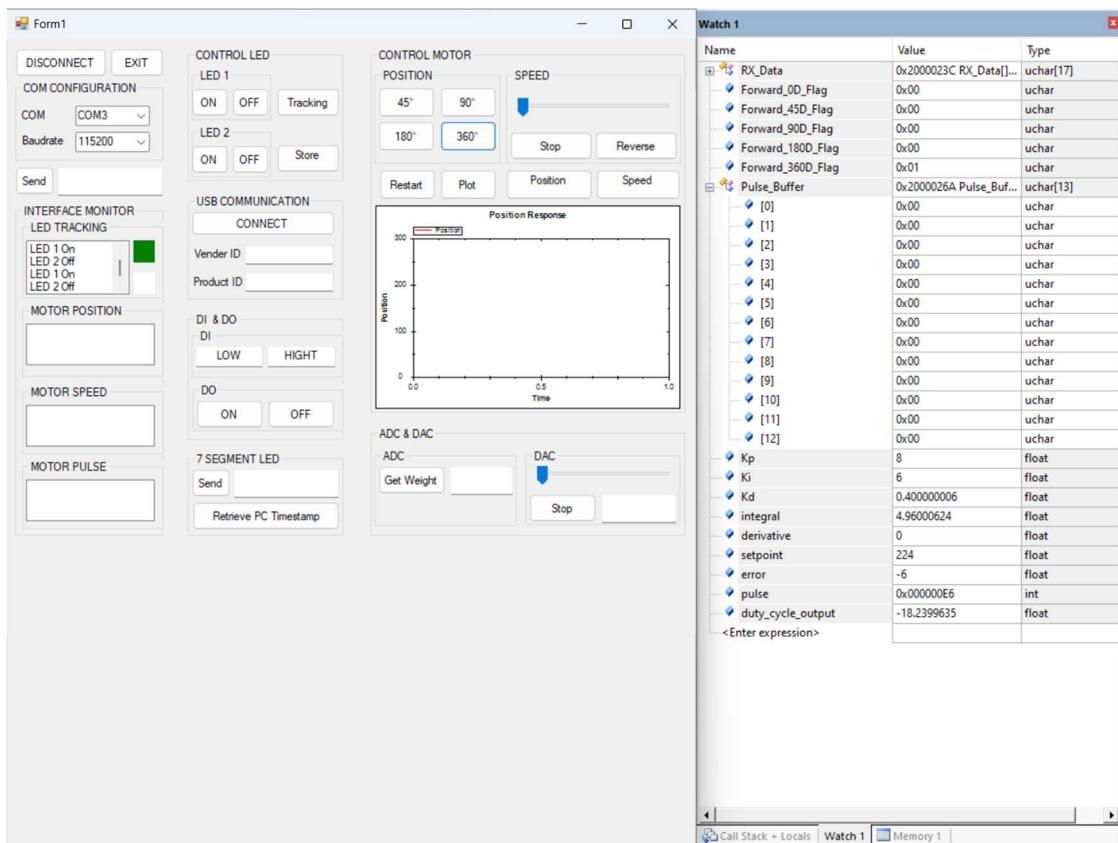




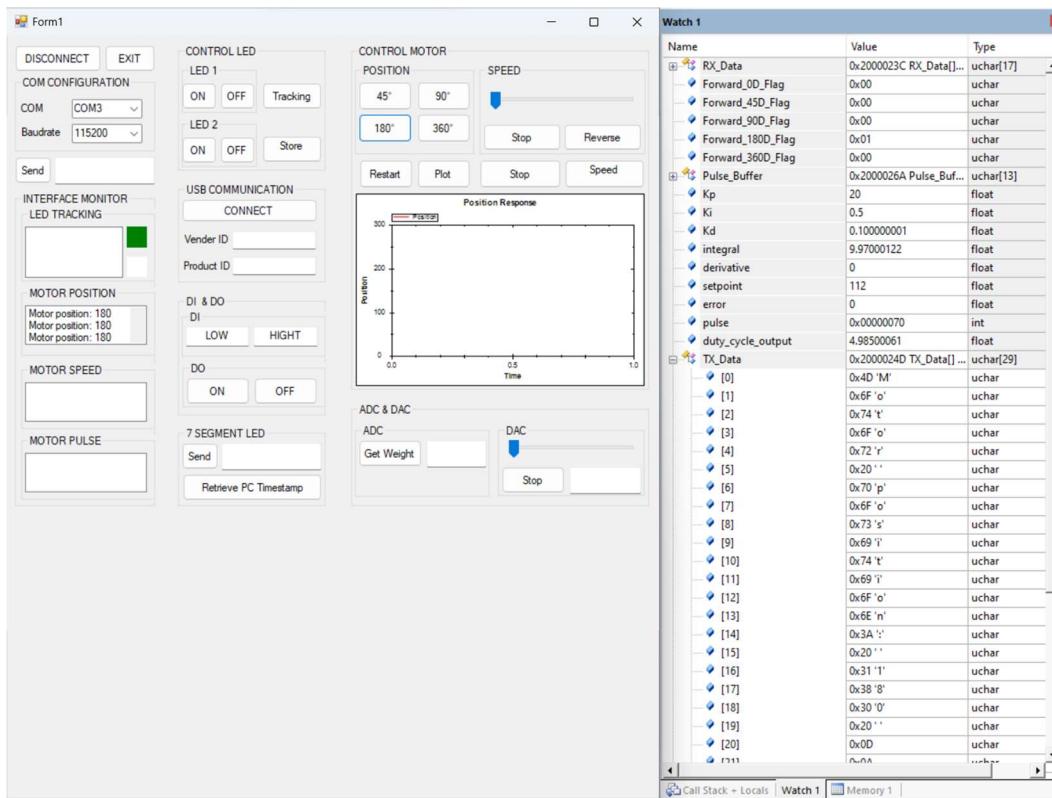
## 2.4.2. Vị trí động cơ

### Điều khiển vị trí

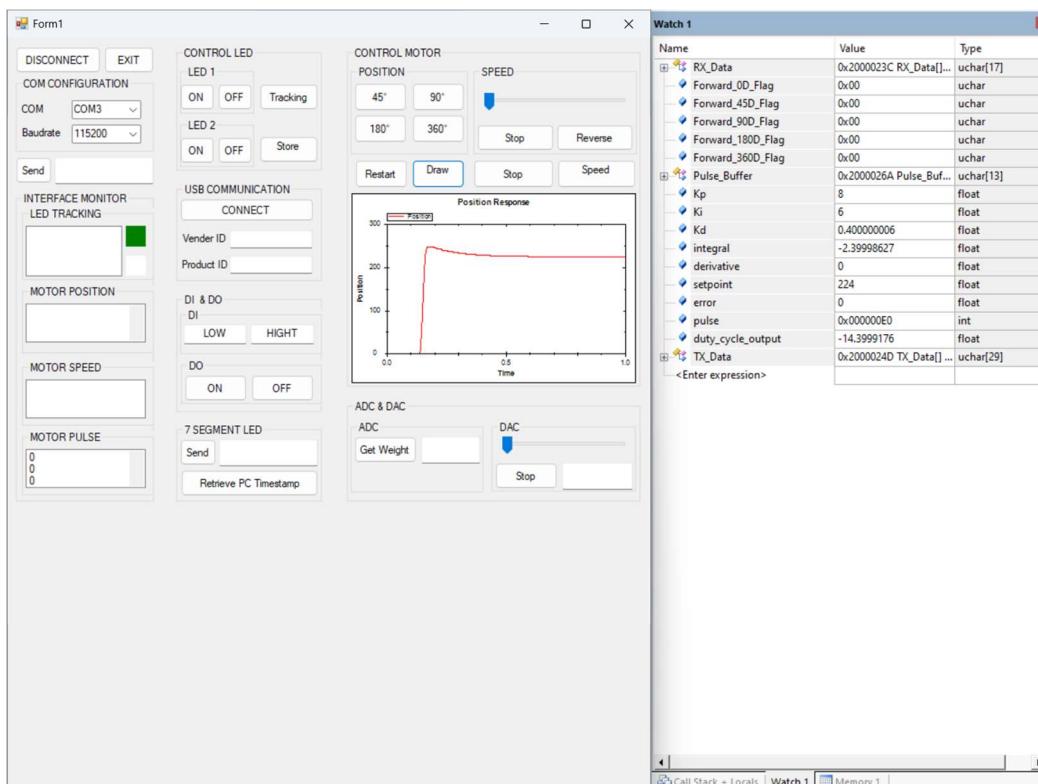




## Theo dõi vị trí



### 2.4.3. Đáp ứng vị trí động cơ sử dụng bộ điều khiển PID



### III. DIGITAL INPUT & DIGITAL OUTPUT

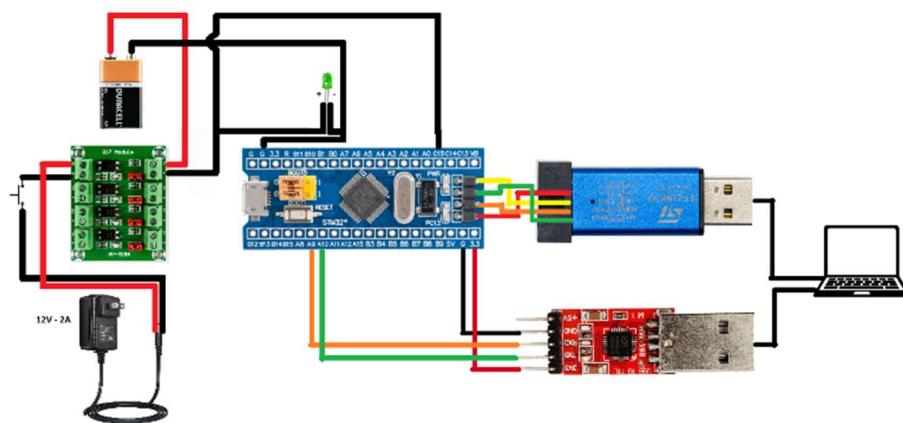
#### 3.1. Tổng quát

Sử dụng 2 channels của Optocoupler dùng làm DI và DO.

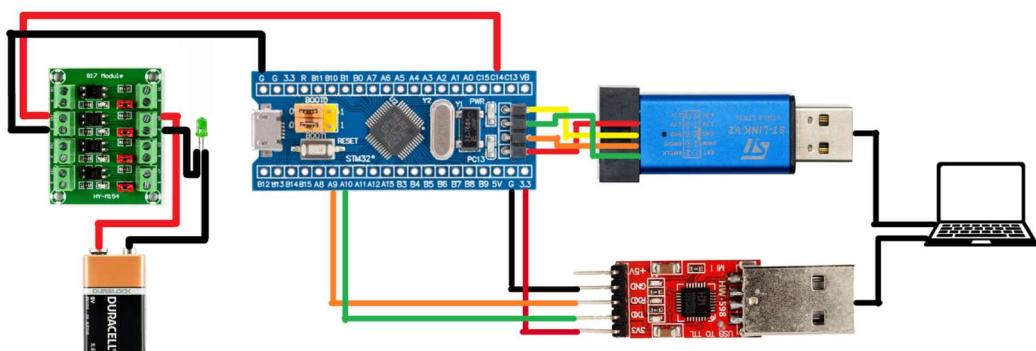
DI: Kết nối với một button ngoài, button đóng vai trò điều khiển tín hiệu ON/OFF, tương ứng với tín hiệu input vào MCU port hai mức HIGH/LOW

#### 3.2. Kết nối phần cứng

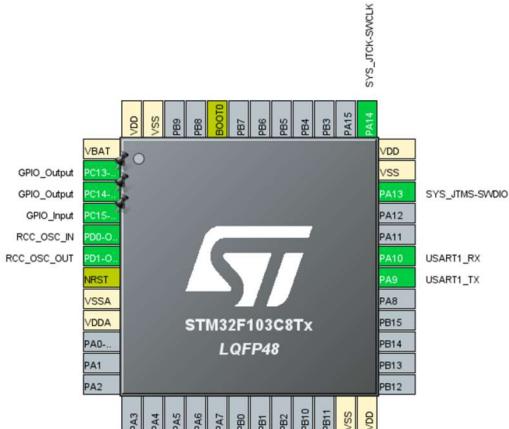
Input (Channel 1)



Output (Channel 2)



#### 3.3. Cấu hình và lập trình phần mềm



NVIC		Code generation		
Priority Group	4 b... ▾	<input type="checkbox"/> Sort by Preemption Priority and Sub Priority	<input type="checkbox"/> Sort by interrupts names	
Search	Show <input style="border: 1px solid #ccc; padding: 2px 10px; border-radius: 5px;" type="button" value="available interrupts"/>	<input checked="" type="checkbox"/> Force DMA channels Interrupts		
NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority	
Non maskable interrupt	<input checked="" type="checkbox"/>	0	0	
Hard fault interrupt	<input checked="" type="checkbox"/>	0	0	
Memory management fault	<input checked="" type="checkbox"/>	0	0	
Prefetch fault, memory access fault	<input checked="" type="checkbox"/>	0	0	
Undefined instruction or illegal state	<input checked="" type="checkbox"/>	0	0	
System service call via SWI instruction	<input checked="" type="checkbox"/>	0	0	
Debug monitor	<input checked="" type="checkbox"/>	0	0	
Pendable request for system service	<input checked="" type="checkbox"/>	0	0	
Time base: System tick timer	<input checked="" type="checkbox"/>	15	0	
PVD interrupt through EXTI line 16	<input type="checkbox"/>	0	0	
Flash global interrupt	<input type="checkbox"/>	0	0	
RCC global interrupt	<input type="checkbox"/>	0	0	
USART1 global interrupt	<input checked="" type="checkbox"/>	0	0	

```
int Opto_Digital_Input;
uint8_t Opto_RX_Buffer [1];
char *Char Opto RX Buffer;
```

```

Char_Opto_RX_Buffer = (char*)Opto_RX_Buffer;

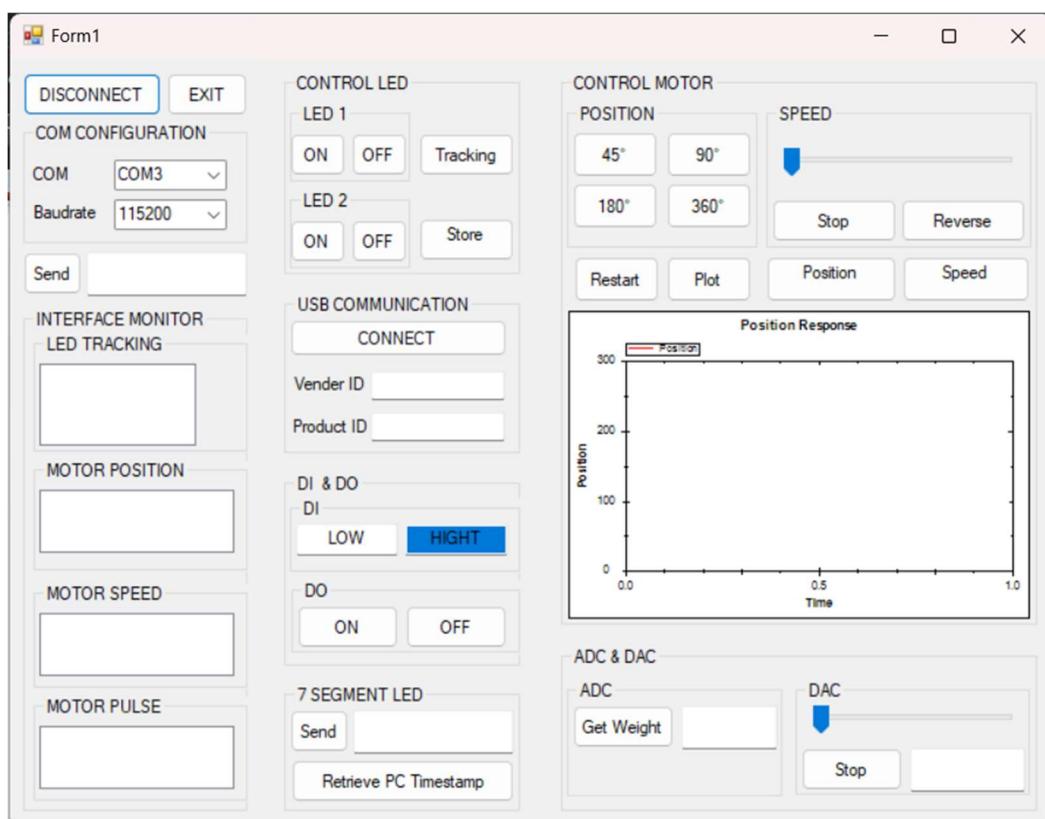
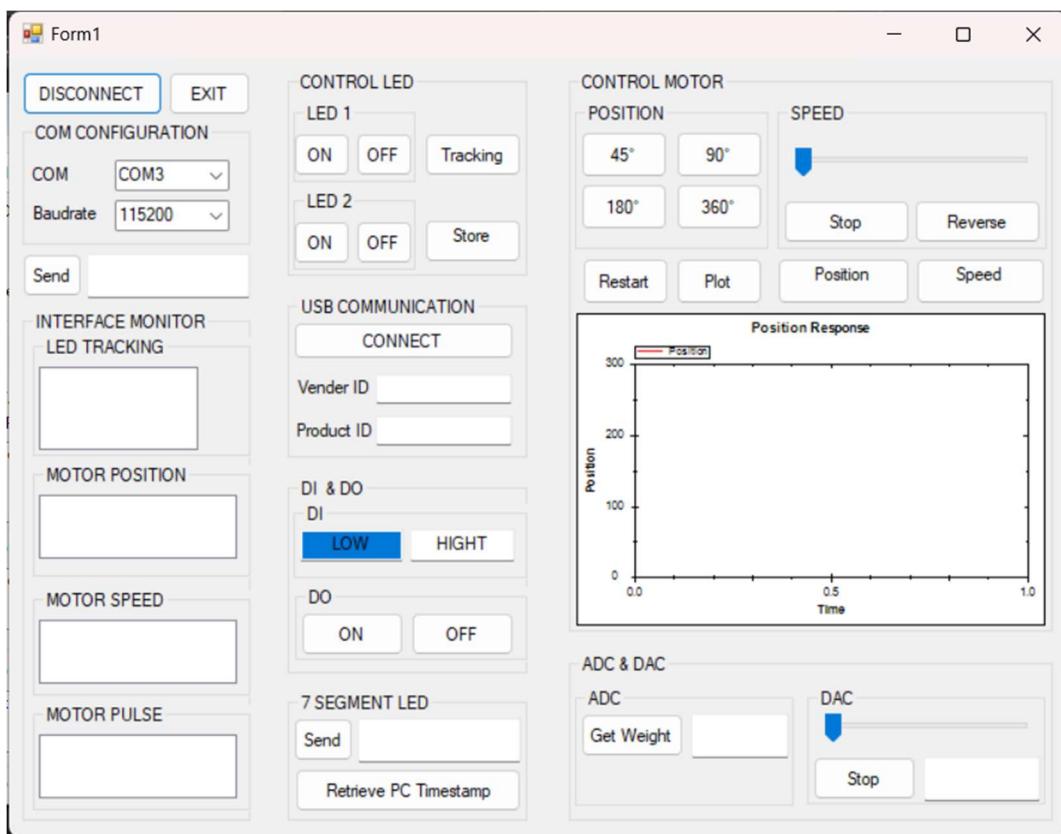
HAL_UART_Receive_IT (&huart1, (uint8_t *)Opto_RX_Buffer, sizeof(Opto_RX_Buffer));
if (HAL_GPIO_ReadPin (GPIOC, GPIO_PIN_15) == 1)
{
    Opto_Digital_Input = 1;
}
else
{
    Opto_Digital_Input = 0;
}

if (strcmp(Char_Opto_RX_Buffer, "1", sizeof(Opto_RX_Buffer)) == 0)
{
    HAL_GPIO_WritePin (GPIOC, GPIO_PIN_14, 1);
}
else if (strcmp(Char_Opto_RX_Buffer, "0", sizeof(Opto_RX_Buffer)) == 0)
{
    HAL_GPIO_WritePin (GPIOC, GPIO_PIN_14, 0);
}

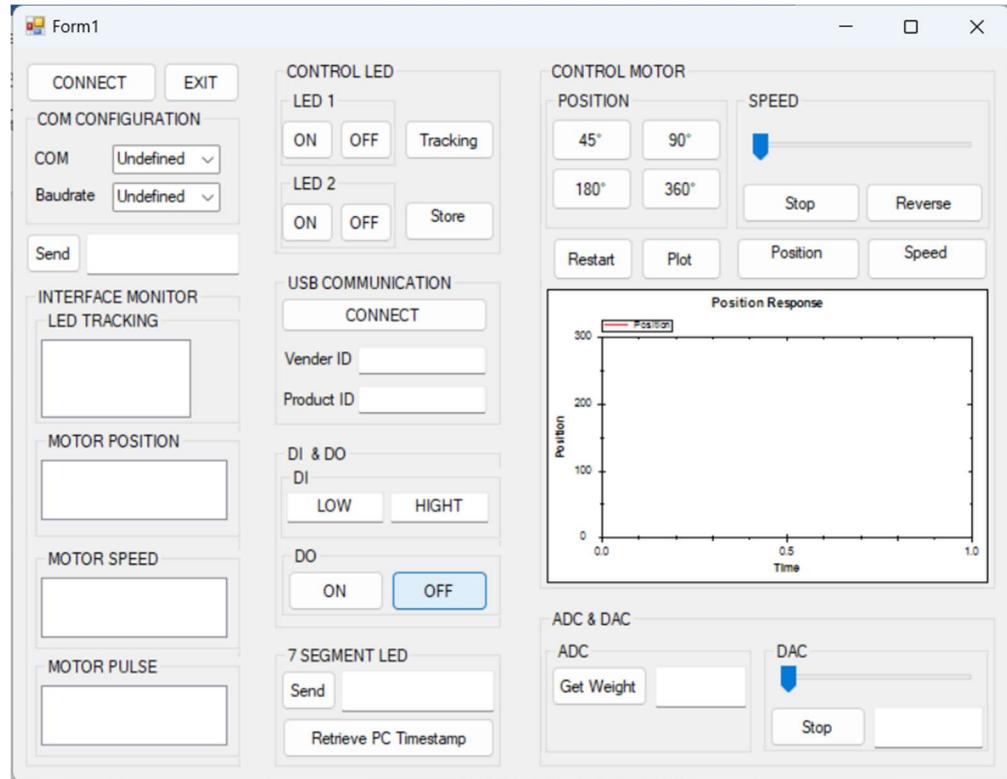
```

### 3.4. Kết quả

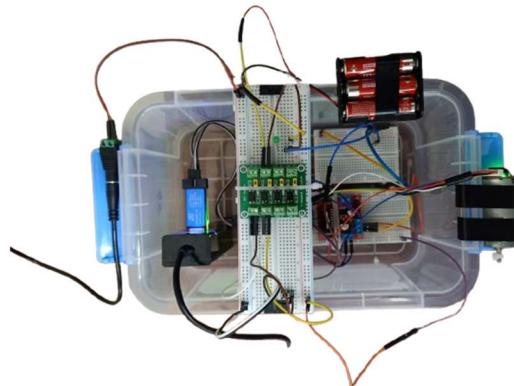
### Input (Channel 1)

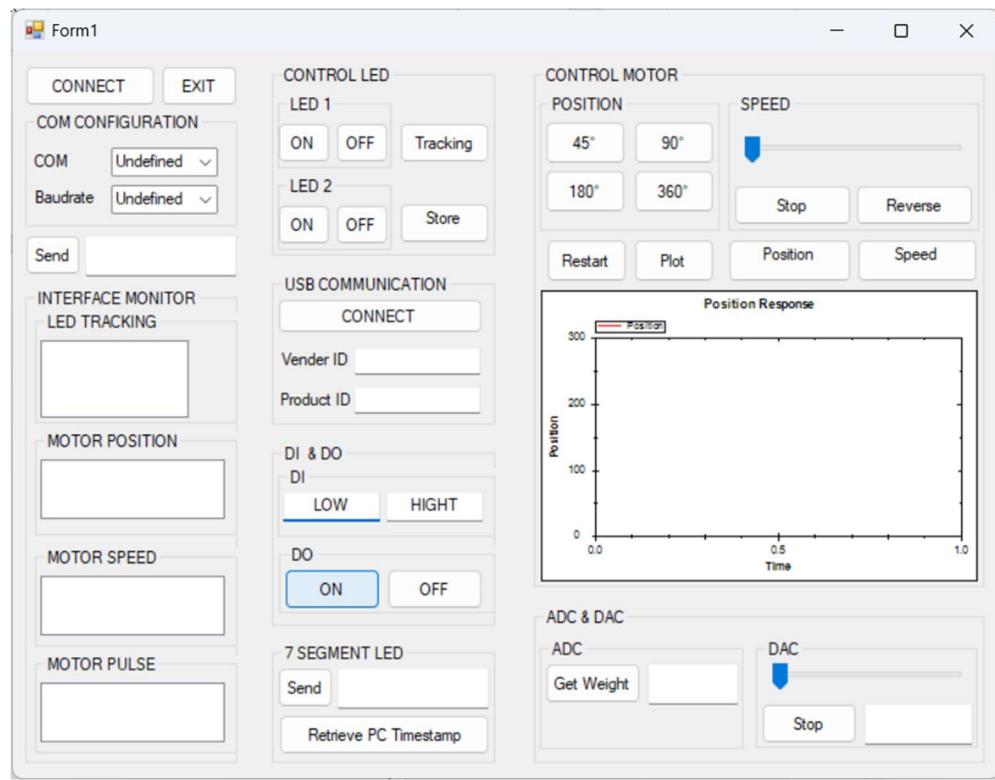


## Output (Channel 2)

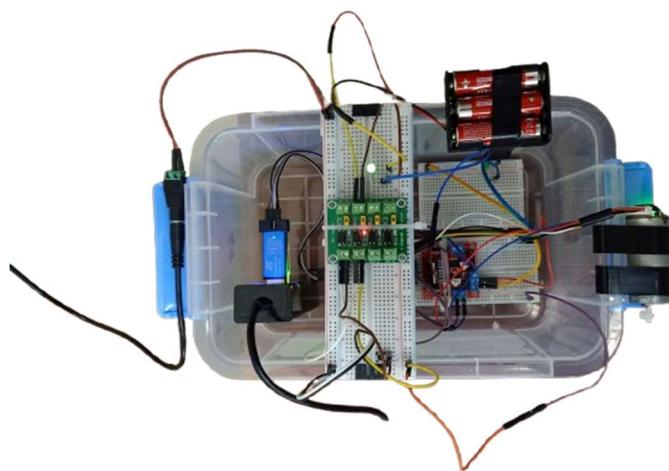


Name	Value	Type
Opto_Digital_Input	0x00000000	int
Opto_RX_Buffer	0x20000000 Opto_RX_...	uchar[1]
[0]	0x30 '0'	uchar
<Enter expression>		





Name	Value	Type
Opto_Digital_Input	0x00000000	int
Opto_RX_Buffer	0x20000000 Opto_RX_...	uchar[1]
[0]	0x31 '1'	uchar
<Enter expression>		



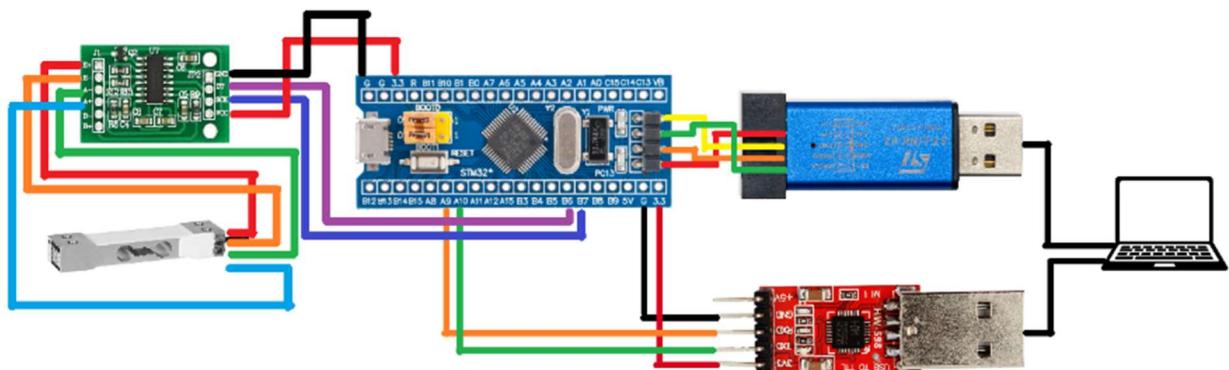
## IV. ADC & DAC

### 4.1. ADC

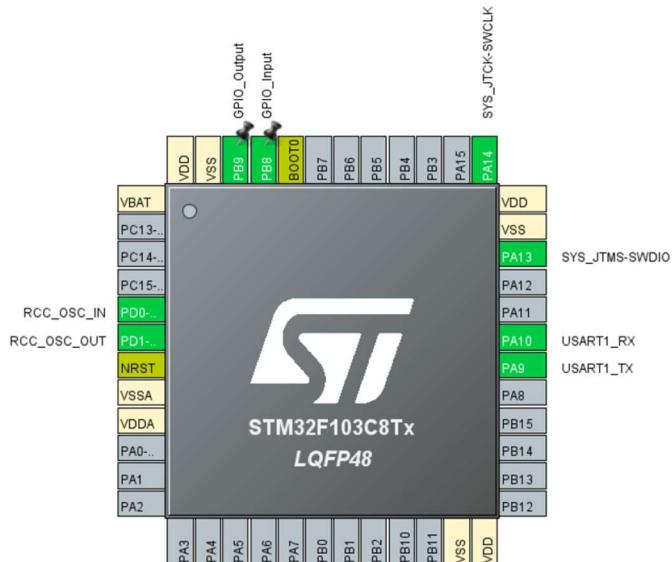
#### 4.1.1. Tổng quát

Sử dụng bộ chuyển đổi HX711 lấy tín hiệu khuếch đại áp từ Loadcell, sau đó mang tín hiệu analog này qua ADC của KIT STM để nhận được tín hiệu digital.

#### 4.1.2. Kết nối phần cứng



#### 4.1.3. Cấu hình và lập trình phần mềm



NVIC     Code generation

Priority Group   Sort by Preemption Priority and Sub Priority  Sort by interrupts names

Search  Show   Force DMA channels Interrupts

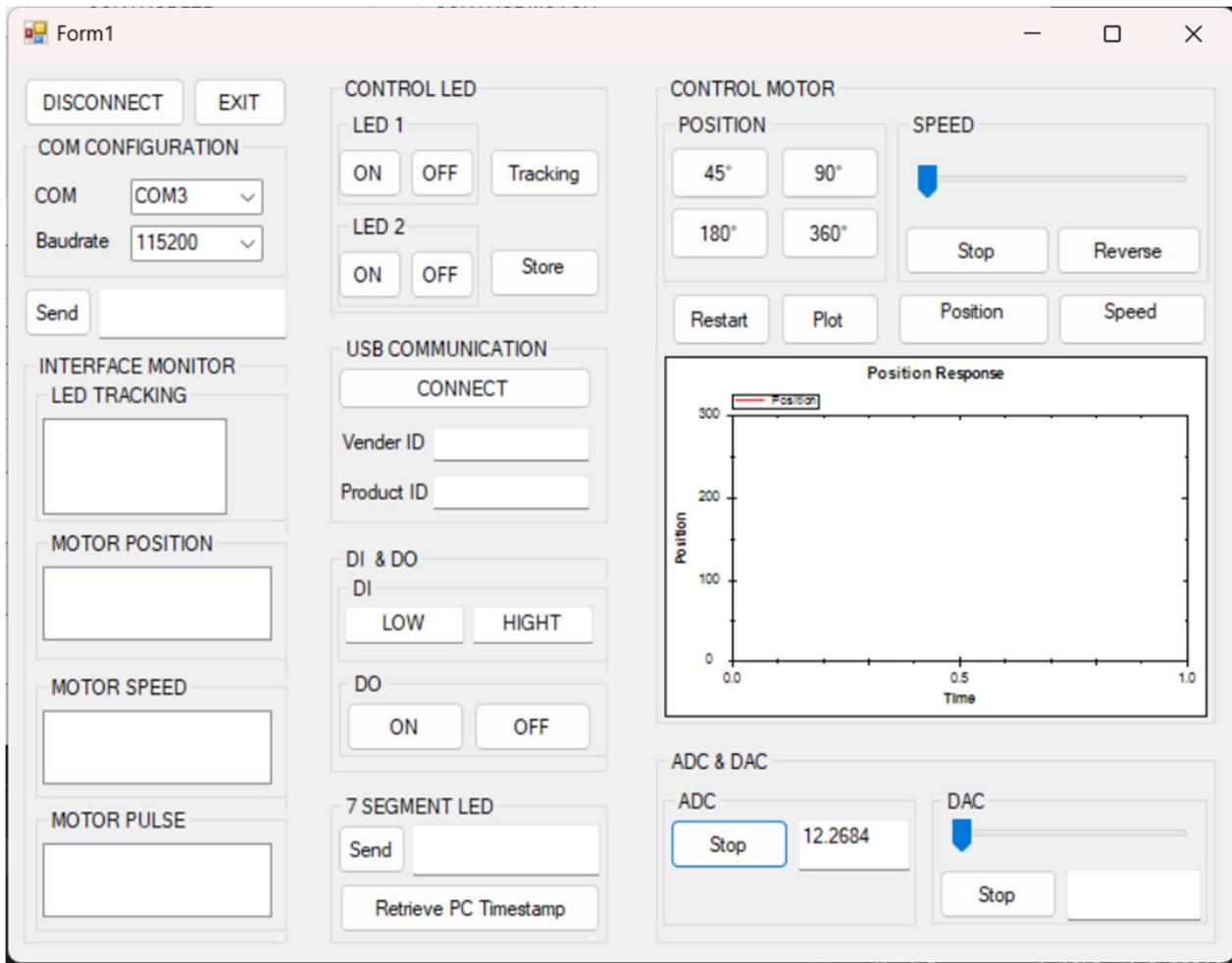
NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
Non maskable interrupt	<input checked="" type="checkbox"/>	0	0
Hard fault interrupt	<input checked="" type="checkbox"/>	0	0
Memory management fault	<input checked="" type="checkbox"/>	0	0
Prefetch fault, memory access fault	<input checked="" type="checkbox"/>	0	0
Undefined instruction or illegal state	<input checked="" type="checkbox"/>	0	0
System service call via SWI instruction	<input checked="" type="checkbox"/>	0	0
Debug monitor	<input checked="" type="checkbox"/>	0	0
Pendable request for system service	<input checked="" type="checkbox"/>	0	0
Time base: System tick timer	<input checked="" type="checkbox"/>	0	0
PVD interrupt through EXTI line 16	<input type="checkbox"/>	0	0
Flash global interrupt	<input type="checkbox"/>	0	0
RCC global interrupt	<input type="checkbox"/>	0	0
DMA1 channel4 global interrupt	<input checked="" type="checkbox"/>	0	0
DMA1 channel5 global interrupt	<input checked="" type="checkbox"/>	0	0
TIM2 global interrupt	<input checked="" type="checkbox"/>	0	0
USART1 global interrupt	<input checked="" type="checkbox"/>	0	0

```

void Mode_09 ()
{
void HAL_UARTEx_RxEventCallback (UART_HandleTypeDef *huart, uint16_t Size)
{
void UART1_RX_Processing ()
{
void microDelay(uint16_t delay)
{
int32_t getHX711(void)
{
    uint32_t data = 0;
    uint32_t startTime = HAL_GetTick();
    while(HAL_GPIO_ReadPin(DT_PORT, DT_PIN) == GPIO_PIN_SET)
    {
        if(HAL_GetTick() - startTime > 200)
            return 0;
    }
    for(int8_t len=0; len<24 ; len++)
    {
        HAL_GPIO_WritePin(SCK_PORT, SCK_PIN, GPIO_PIN_SET);
        microDelay(1);
        data = data << 1;
        HAL_GPIO_WritePin(SCK_PORT, SCK_PIN, GPIO_PIN_RESET);
        microDelay(1);
        if(HAL_GPIO_ReadPin(DT_PORT, DT_PIN) == GPIO_PIN_SET)
            data++;
    }
    data = data ^ 0x800000;
    HAL_GPIO_WritePin(SCK_PORT, SCK_PIN, GPIO_PIN_SET);
    microDelay(1);
    HAL_GPIO_WritePin(SCK_PORT, SCK_PIN, GPIO_PIN_RESET);
    microDelay(1);
    return data;
}
}

```

#### 4.1.4. Kết quả

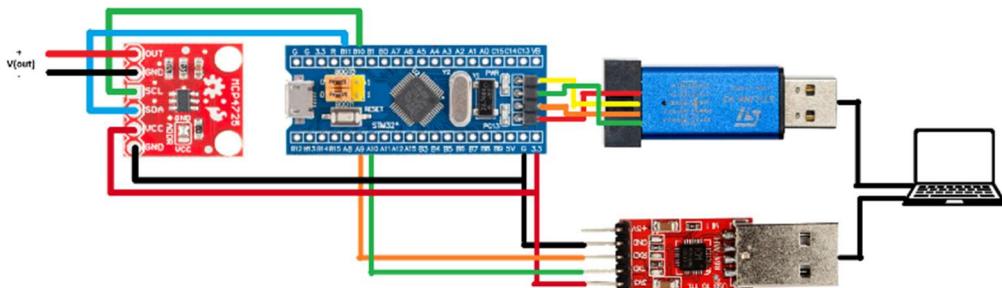


## 4.2. DAC

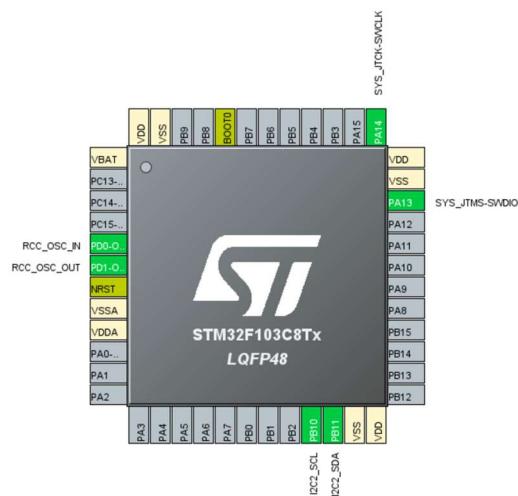
### 4.2.1. Tổng quát

Chuyển tín hiệu digital từ vi điều khiển thành tín hiệu analog thông qua module DAC. Ở đây ta cấp áp ngõ vào KIT là 3V nên analog ngõ ra sẽ được scale từ 0 – 3V.

### 4.2.2. Kết nối phần cứng



#### 4.2.3. Cấu hình và lập trình phần mềm



```

//=====DAC=====//
//=====DAC=====//
//=====DAC=====//

void MCP4725_SetVoltage(int Output_Voltage_Setpoint)
{
    uint8_t Output_Voltage_Buffer[3];

    Output_Voltage_Buffer[0] = 0x40;
    Output_Voltage_Buffer[1] = (Output_Voltage_Setpoint >> 4) & 0xFF;
    Output_Voltage_Buffer[2] = (Output_Voltage_Setpoint & 0x0F) << 4;

    HAL_I2C_Master_Transmit(&i2c2, MCP4725_ADDR, Output_Voltage_Buffer, 3, HAL_MAX_DELAY);
}

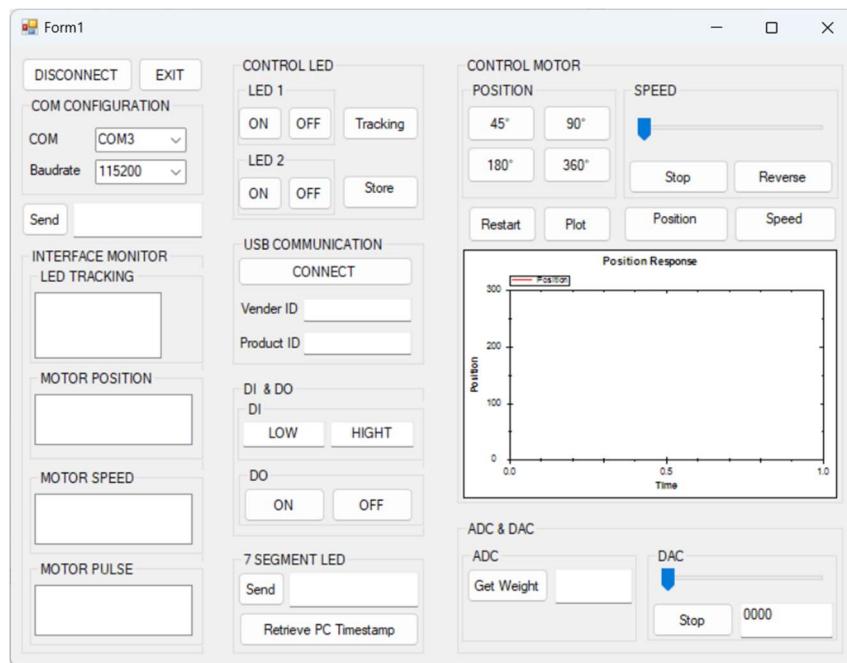
void Change_Output_Voltage ()
{
    if (strncmp(Char_RX_Data, "00060000011", sizeof(RX_Data)) == 0)
    {
        Mode_06_Flag = 0;
    }

    Output_Voltage_Setpoint = (RX_Data[6] - '0') + (RX_Data[7] - '0') / 10.0;
    MCP4725_SetVoltage ((Output_Voltage_Setpoint * 4095) / 3);
}

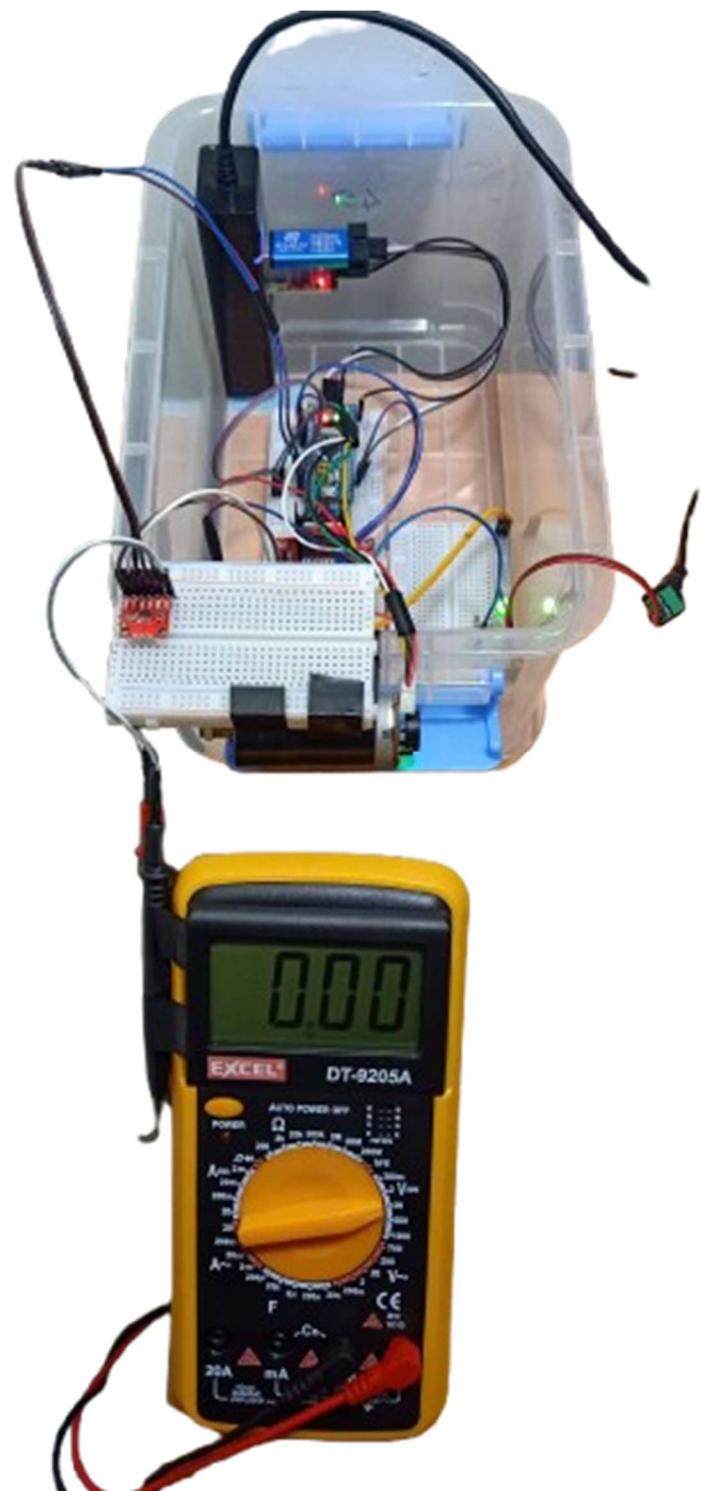
```

#### 4.2.4. Kết quả

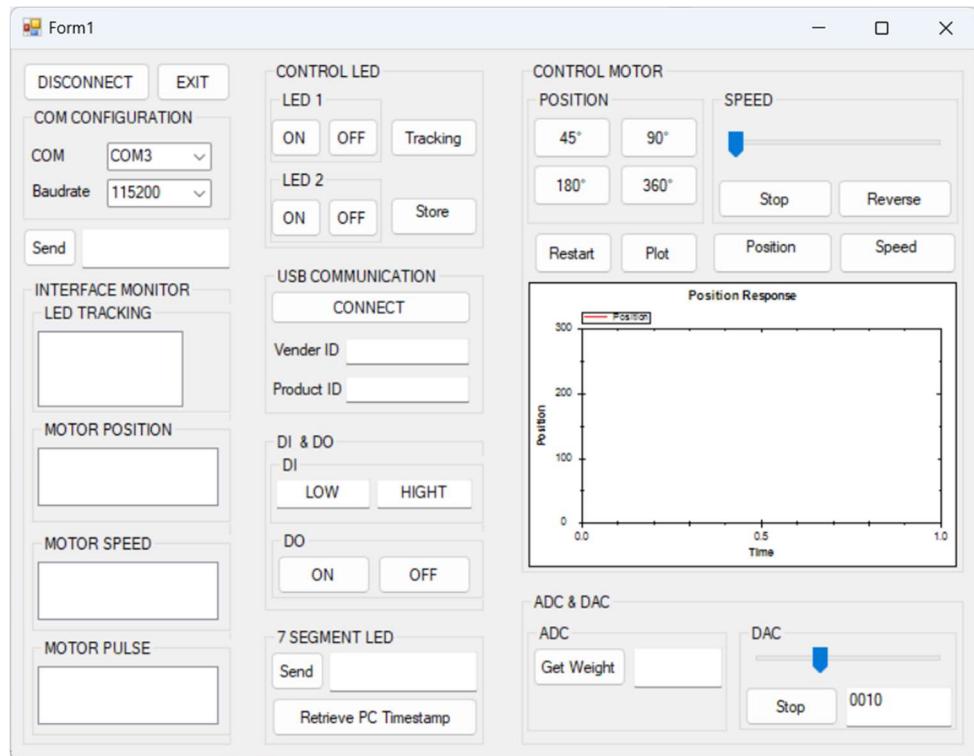
0V



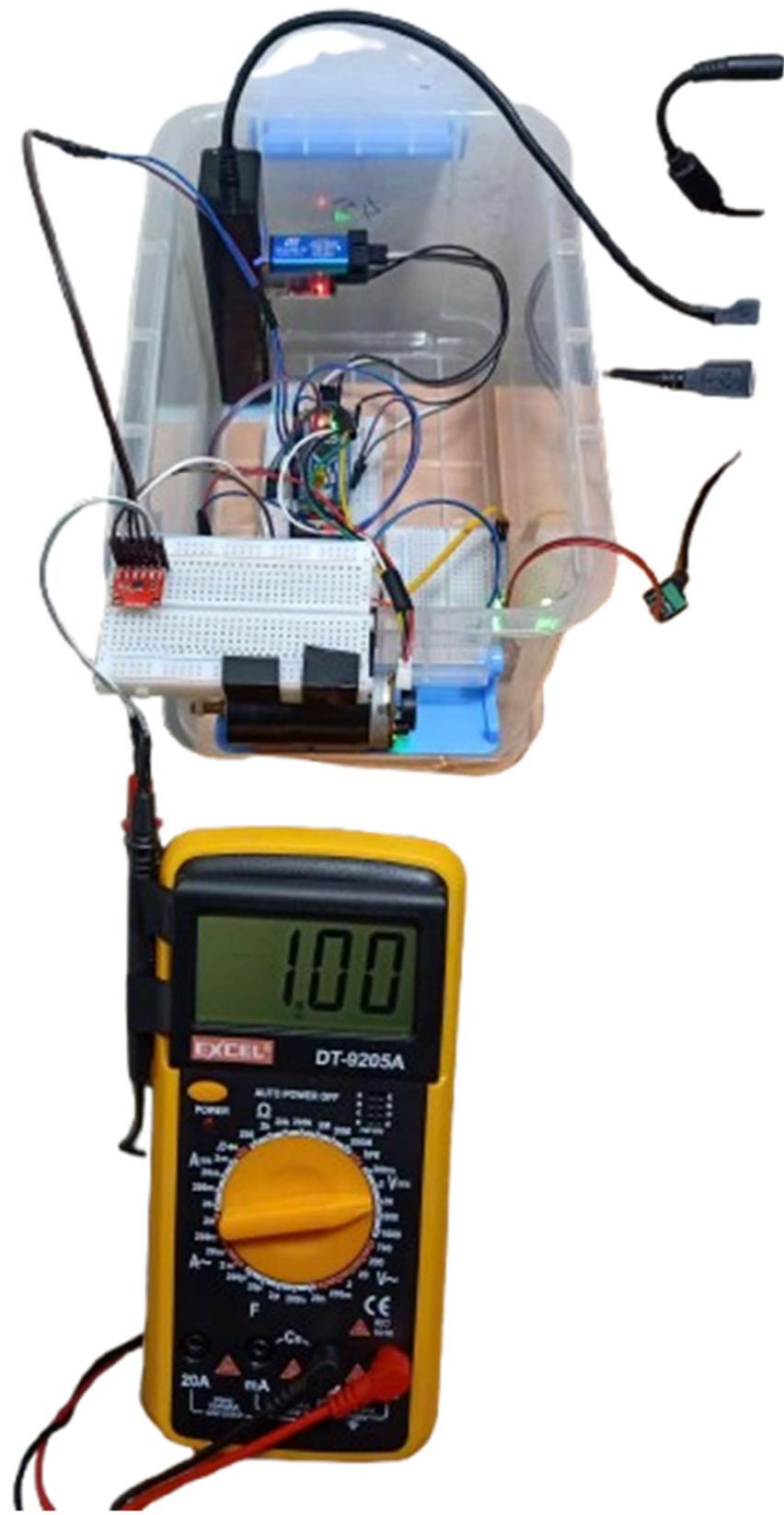
Name	Value	Type
RX_Data	0x200001DC RX_Data[...]	uchar[17]
[0]	0x00	uchar
[1]	0x00	uchar
[2]	0x00	uchar
[3]	0x00	uchar
[4]	0x00	uchar
[5]	0x00	uchar
[6]	0x00	uchar
[7]	0x00	uchar
[8]	0x00	uchar
[9]	0x00	uchar
[10]	0x00	uchar
[11]	0x00	uchar
[12]	0x00	uchar
[13]	0x00	uchar
[14]	0x00	uchar
[15]	0x00	uchar
[16]	0x00	uchar
Output_Voltage_Setpoint	0	float
<Enter expression>		



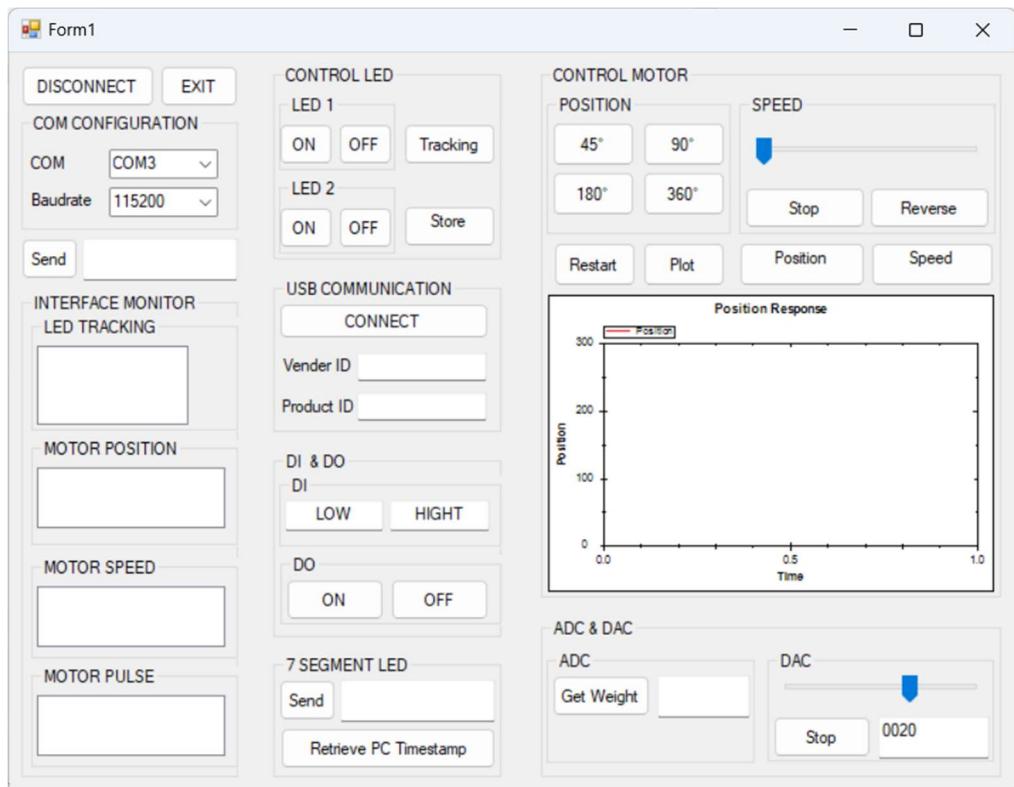
1V



Name	Value	Type
RX_Data	0x200001DC RX_Data[...]	uchar[17]
[0]	0x30 '0'	uchar
[1]	0x30 '0'	uchar
[2]	0x30 '0'	uchar
[3]	0x36 '6'	uchar
[4]	0x30 '0'	uchar
[5]	0x30 '0'	uchar
[6]	0x31 '1'	uchar
[7]	0x30 '0'	uchar
[8]	0x30 '0'	uchar
[9]	0x31 '1'	uchar
[10]	0x31 '1'	uchar
[11]	0x00	uchar
[12]	0x00	uchar
[13]	0x00	uchar
[14]	0x00	uchar
[15]	0x00	uchar
[16]	0x00	uchar
Output_Voltage_Setpoint	1	float
<Enter expression>		



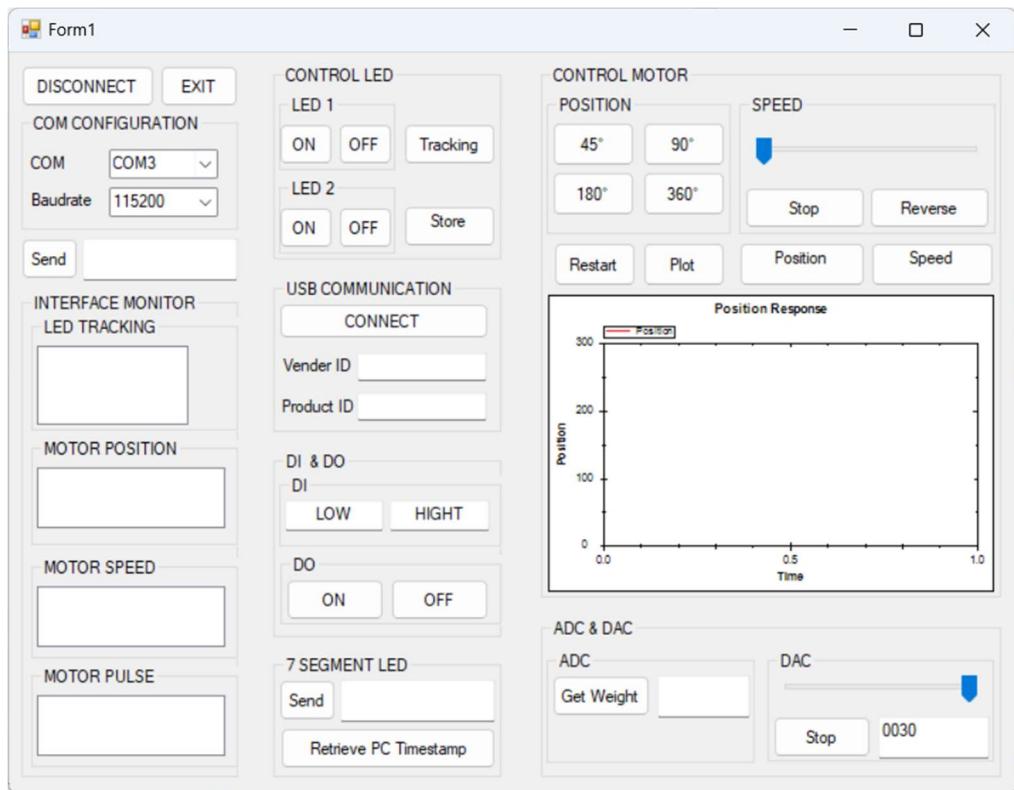
2V



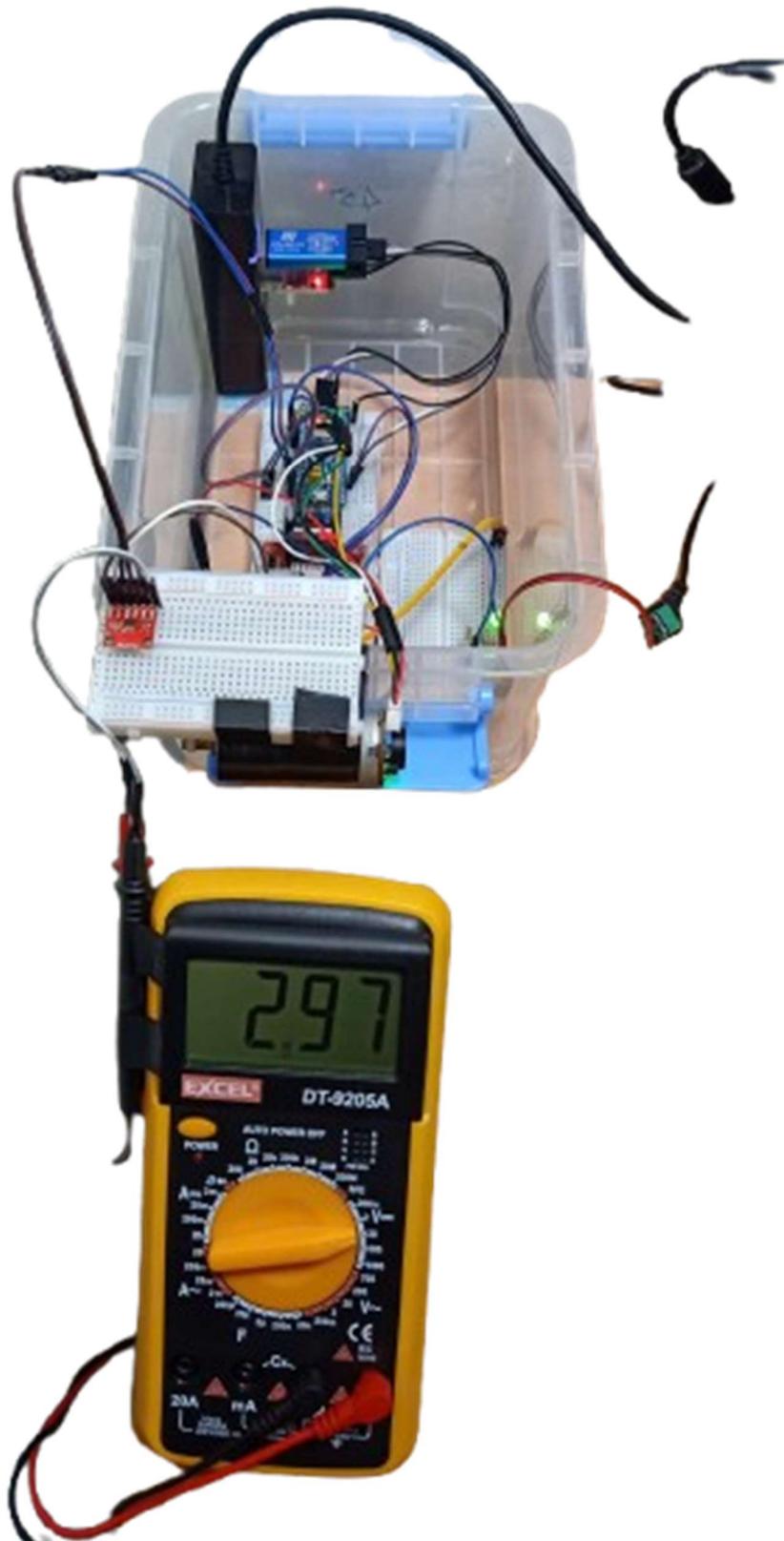
Name	Value	Type
RX_Data	0x200001DC RX_Data[...]	uchar[17]
[0]	0x30 '0'	uchar
[1]	0x30 '0'	uchar
[2]	0x30 '0'	uchar
[3]	0x36 '6'	uchar
[4]	0x30 '0'	uchar
[5]	0x30 '0'	uchar
[6]	0x32 '2'	uchar
[7]	0x30 '0'	uchar
[8]	0x30 '0'	uchar
[9]	0x31 '1'	uchar
[10]	0x31 '1'	uchar
[11]	0x00	uchar
[12]	0x00	uchar
[13]	0x00	uchar
[14]	0x00	uchar
[15]	0x00	uchar
[16]	0x00	uchar
Output_Voltage_Setpoint	2	float
<Enter expression>		



3V



Name	Value	Type
RX_Data	0x200001DC RX_Data[...]	uchar[17]
[0]	0x30 '0'	uchar
[1]	0x30 '0'	uchar
[2]	0x30 '0'	uchar
[3]	0x36 '6'	uchar
[4]	0x30 '0'	uchar
[5]	0x30 '0'	uchar
[6]	0x33 '3'	uchar
[7]	0x30 '0'	uchar
[8]	0x30 '0'	uchar
[9]	0x31 '1'	uchar
[10]	0x31 '1'	uchar
[11]	0x00	uchar
[12]	0x00	uchar
[13]	0x00	uchar
[14]	0x00	uchar
[15]	0x00	uchar
[16]	0x00	uchar
Output_Voltage_Setpoint	3	float
<Enter expression>		

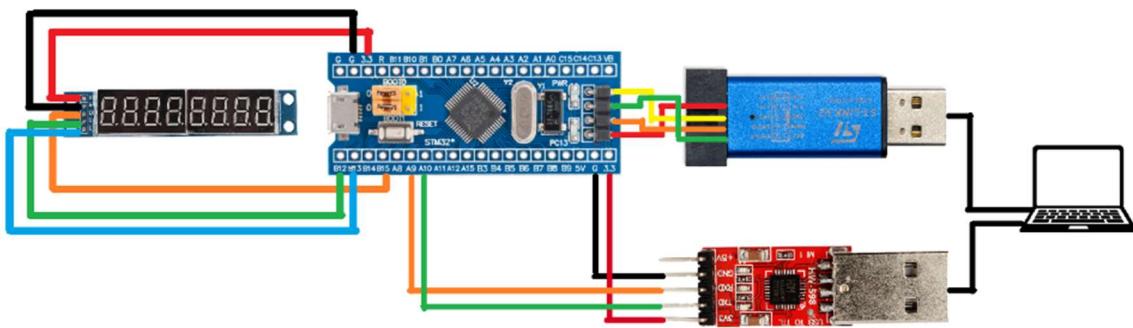


## V. SPI COMMUNICATION

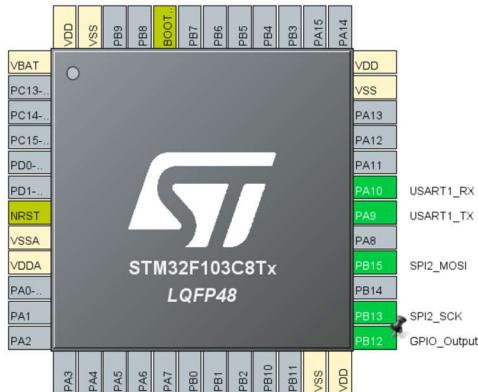
### 5.1. Tông quan

Sử dụng giao tiếp SPI truyền data với mục đích hiển thị chữ số lên module LED 7 đoạn.

### 5.2. Kết nối phần cứng



### 5.3. Cấu hình và lập trình phần mềm



NVIC | Code generation

Priority Group 4 bits ...  Sort by Preemption Priority and Sub Priority  Sort by interrupts names

Search  Show available interrupts  Force DMA channels Interrupts

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
Non maskable interrupt	<input checked="" type="checkbox"/>	0	0
Hard fault interrupt	<input checked="" type="checkbox"/>	0	0
Memory management fault	<input checked="" type="checkbox"/>	0	0
Prefetch fault, memory access fault	<input checked="" type="checkbox"/>	0	0
Undefined instruction or illegal state	<input checked="" type="checkbox"/>	0	0
System service call via SWI instruction	<input checked="" type="checkbox"/>	0	0
Debug monitor	<input checked="" type="checkbox"/>	0	0
Pendable request for system service	<input checked="" type="checkbox"/>	0	0
Time base: System tick timer	<input checked="" type="checkbox"/>	0	0
PVD interrupt through EXTI line 16	<input type="checkbox"/>	0	0
Flash global interrupt	<input type="checkbox"/>	0	0
RCC global interrupt	<input type="checkbox"/>	0	0
DMA1 channel5 global interrupt	<input checked="" type="checkbox"/>	0	0
SPI2 global interrupt	<input type="checkbox"/>	0	0
USART1 global interrupt	<input checked="" type="checkbox"/>	0	0

```

void HAL_UARTEX_RxEventCallback(UART_HandleTypeDef *huart, uint16_t Size)
{
    if (huart->Instance == USART1)
    {
        UART1_RX_Flag = 1;
        RX_Data[Size] = '\0';
    }
}

void UART1_RX_Processing ()
{
    UART1_RX_Flag = 0;
    HAL_UARTEX_ReceiveToIdle_DMA (&huart1, RX_Data, sizeof(RX_Data));

    memcpy(SPI_Buffer, &RX_Data[4], 4);

    SPI_Data = atoi((char *)SPI_Buffer);
}

void MAX7219_Send(uint8_t address, uint8_t data)
{
    uint8_t buffer[2] = {address, data};
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_12, GPIO_PIN_RESET);
    HAL_SPI_Transmit(&hspi2, buffer, 2, HAL_MAX_DELAY);
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_12, GPIO_PIN_SET);
}

void MAX7219_Init(void)
{
    MAX7219_Send(0x09, 0xFF);
    MAX7219_Send(0x0A, 0x0A);
    MAX7219_Send(0x0B, 0x07);
    MAX7219_Send(0x0C, 0x01);
    MAX7219_Send(0x0F, 0x00);
}

```

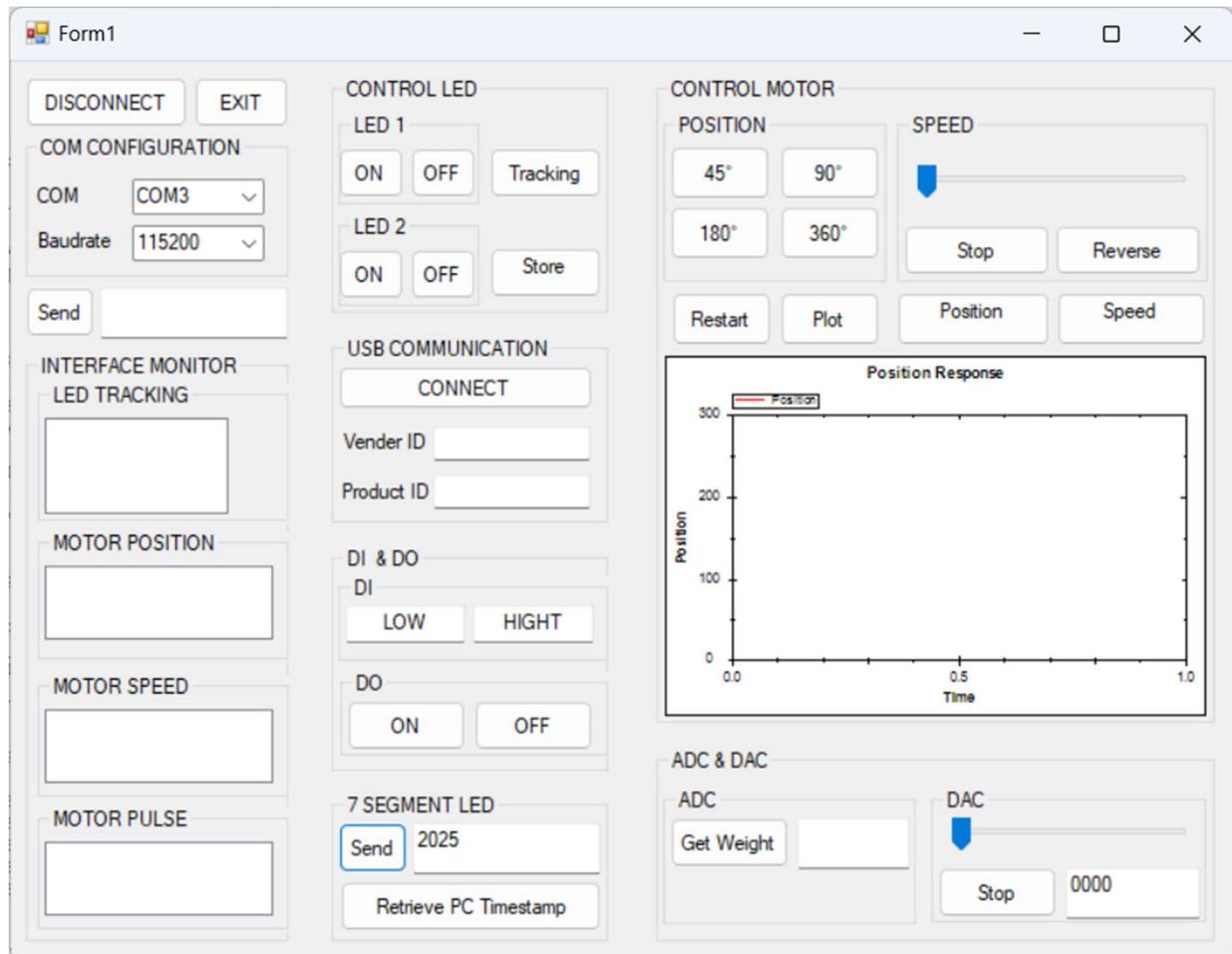
```

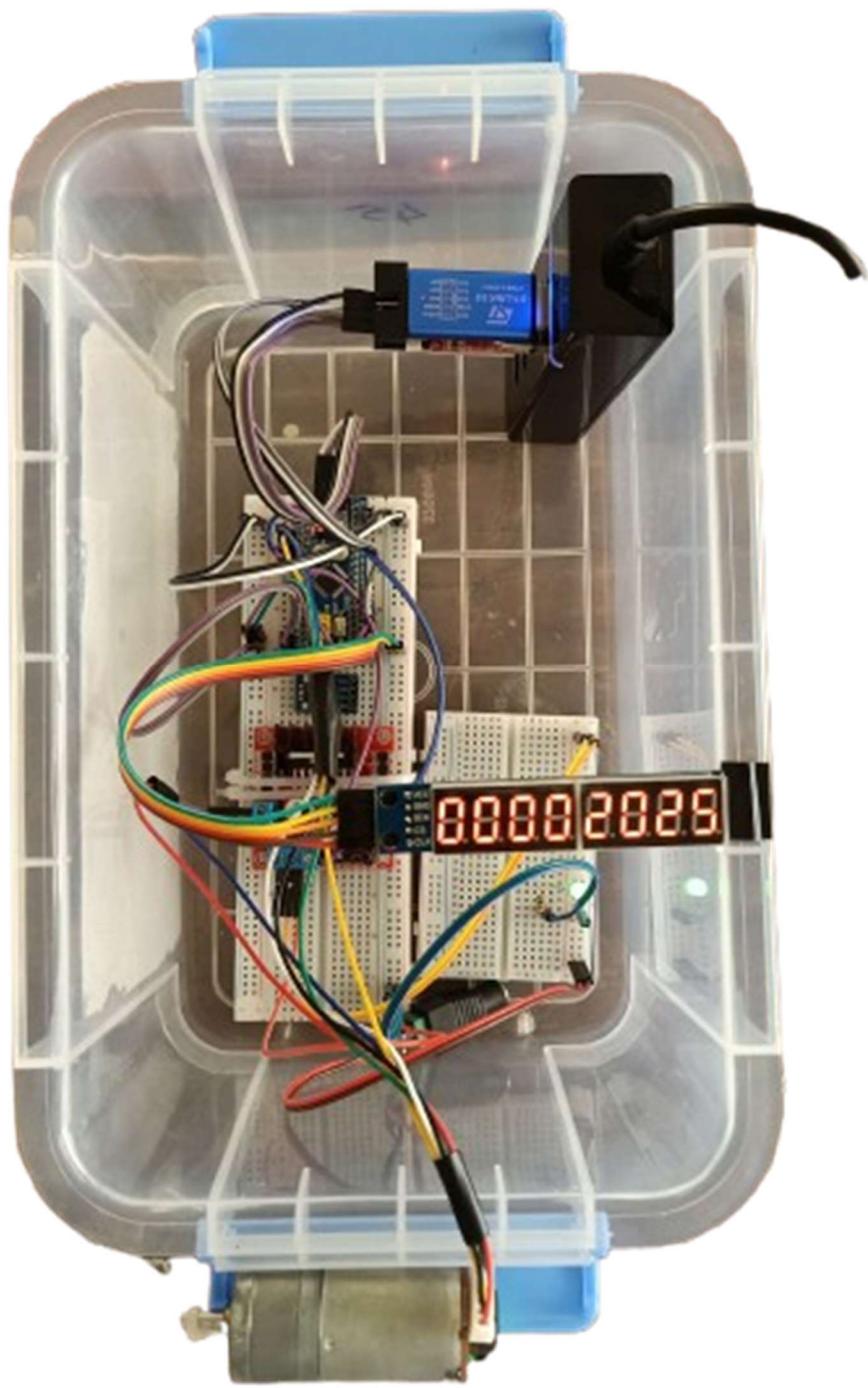
if (UART1_RX_Flag == 1)
{
    UART1_RX_Processing ();
}

MAX7219_Send(0x01, (SPI_Data / 1) % 10);
MAX7219_Send(0x02, (SPI_Data / 10) % 10);
MAX7219_Send(0x03, (SPI_Data / 100) % 10);
MAX7219_Send(0x04, (SPI_Data / 1000) % 10);
MAX7219_Send(0x05, (SPI_Data / 10000) % 10);
MAX7219_Send(0x06, (SPI_Data / 100000) % 10);
MAX7219_Send(0x07, (SPI_Data / 1000000) % 10);
MAX7219_Send(0x08, (SPI_Data / 10000000) % 10);

```

## 5.4. Kết quả

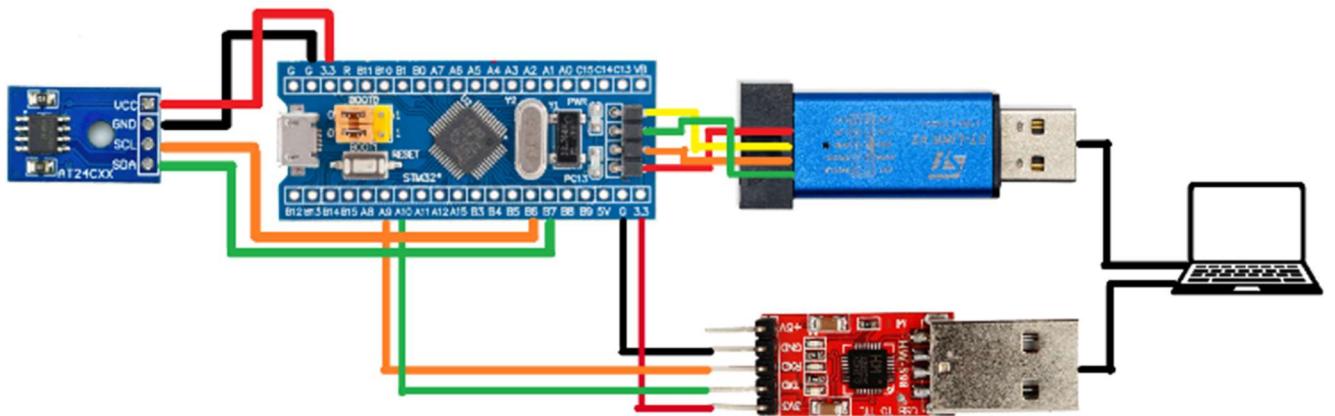




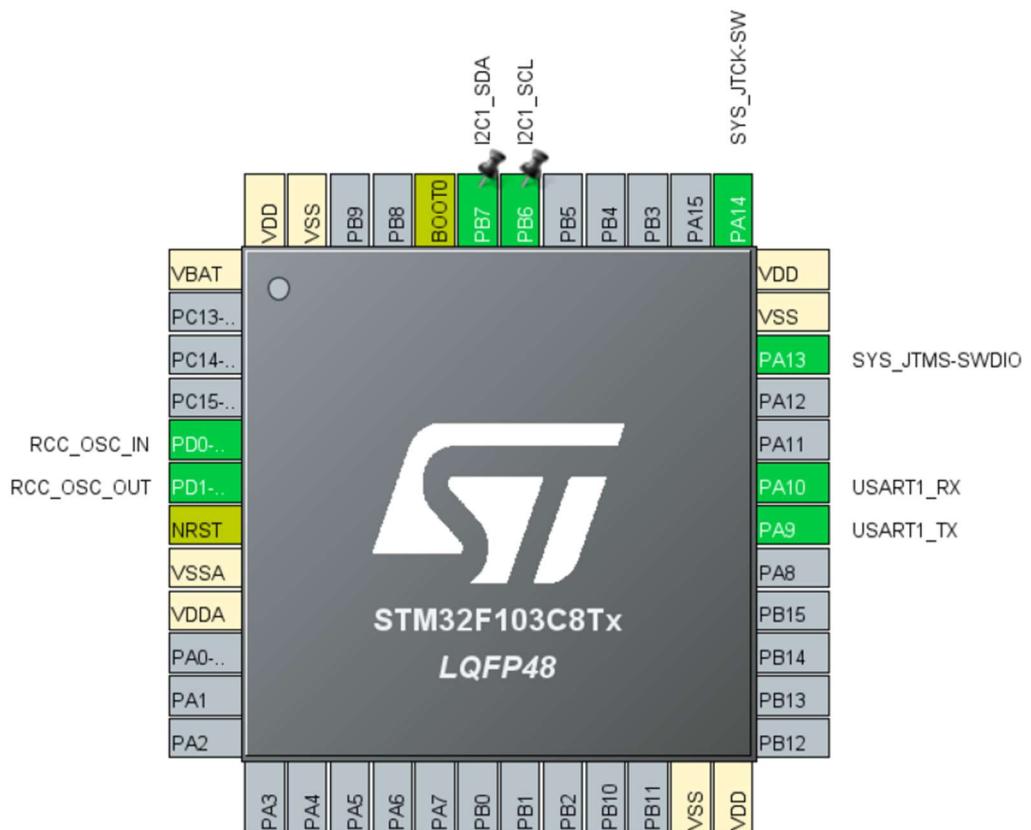
## VI. I<sup>2</sup>C COMMUNICATION

### 6.1. Tổng quan

### 6.2. Kết nối phần cứng



### 6.3. Cấu hình và lập trình phần mềm



```

//=====EEPROM=====
//=====
//=====

void Recover_Data ()
{
    Recover_Data_Flag = 1;

    for(int i = 0, j = 0; i <= 2; i++)
    {
        HAL_I2C_Mem_Read(&hi2c1, 0xA0, i, 2, Each_Eeprom_Element, 1, 100);
        LED_Status_Buffer[j] = Each_Eeprom_Element[0];
        j++;
    }

    if (LED_Status_Buffer[0] == 0)
    {
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, 1);

        Label_LED_Status = 1000;
        int len = sprintf(Label_LED_Status_Buffer, "%d\r\n", Label_LED_Status);
        HAL_UART_Transmit (&huart1, (uint8_t *)Label_LED_Status_Buffer, len, HAL_MAX_DELAY);
    }
    else if (LED_Status_Buffer[0] == 1)
    {
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, 0);

        Label_LED_Status = 1001;
        int len = sprintf(Label_LED_Status_Buffer, "%d\r\n", Label_LED_Status);
        HAL_UART_Transmit (&huart1, (uint8_t *)Label_LED_Status_Buffer, len, HAL_MAX_DELAY);
    }

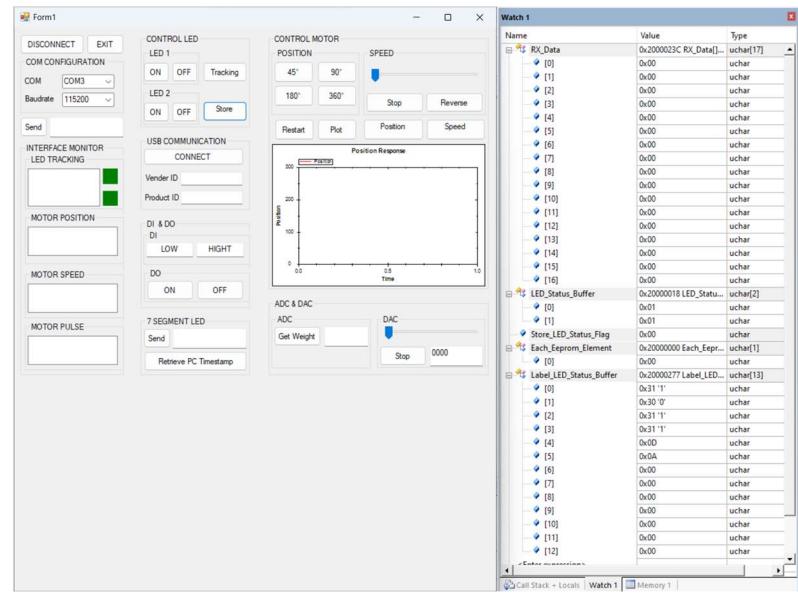
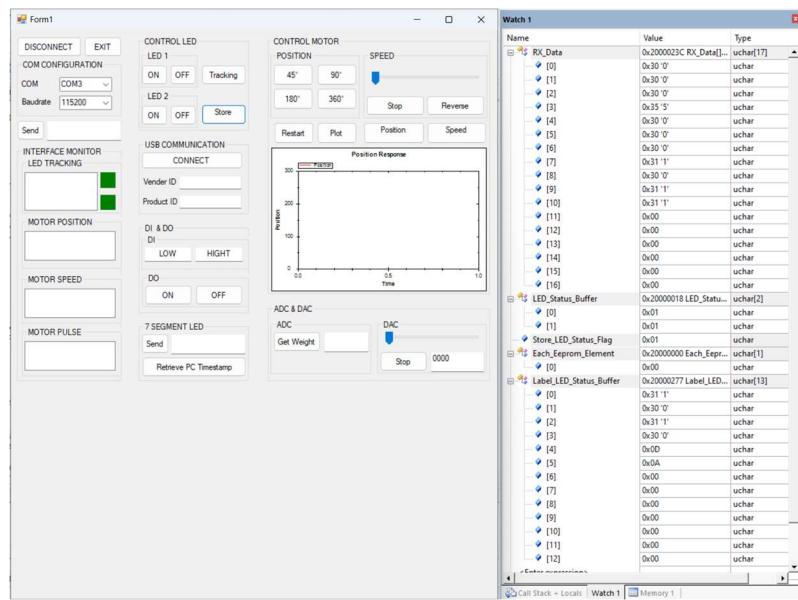
    if (LED_Status_Buffer[1] == 0)
    {
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_4, 0);

        Label_LED_Status = 1010;
        int len = sprintf(Label_LED_Status_Buffer, "%d\r\n", Label_LED_Status);
        HAL_UART_Transmit (&huart1, (uint8_t *)Label_LED_Status_Buffer, len, HAL_MAX_DELAY);
    }
    else if (LED_Status_Buffer[1] == 1)
    {
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_4, 1);

        Label_LED_Status = 1011;
        int len = sprintf(Label_LED_Status_Buffer, "%d\r\n", Label_LED_Status);
        HAL_UART_Transmit (&huart1, (uint8_t *)Label_LED_Status_Buffer, len, HAL_MAX_DELAY);
    }
}

```

## 6.4. Kết quả



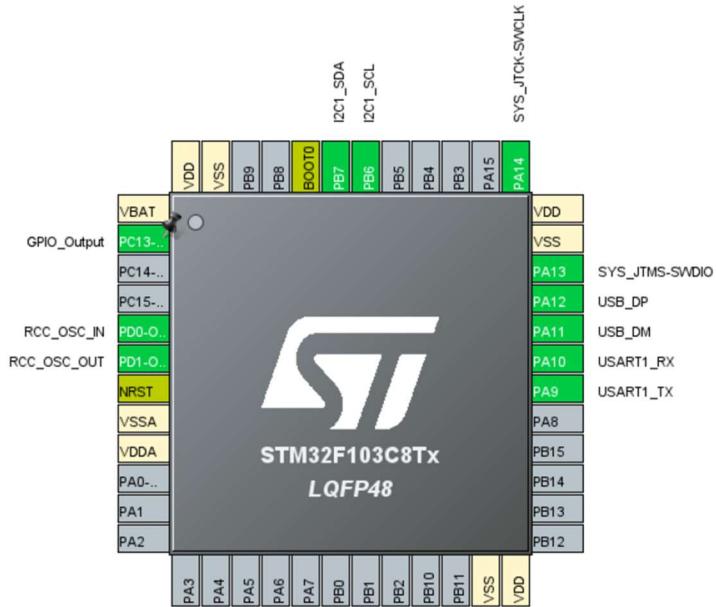
## VII. USB COMMUNICATION

### 7.1. Tổng quan

Giao tiếp USB dùng cho điều khiển trạng thái LED và truy xuất Vender ID & Product ID của USB.

**7.2. Kết nối phần cứng:** Dùng ST-Link V2 để nạp code và kết nối cổng USB (ở đây dùng Micro USB) từ PC đến KIT Blue Pill để giao tiếp.

### 7.3. Cấu hình và lập trình phần mềm

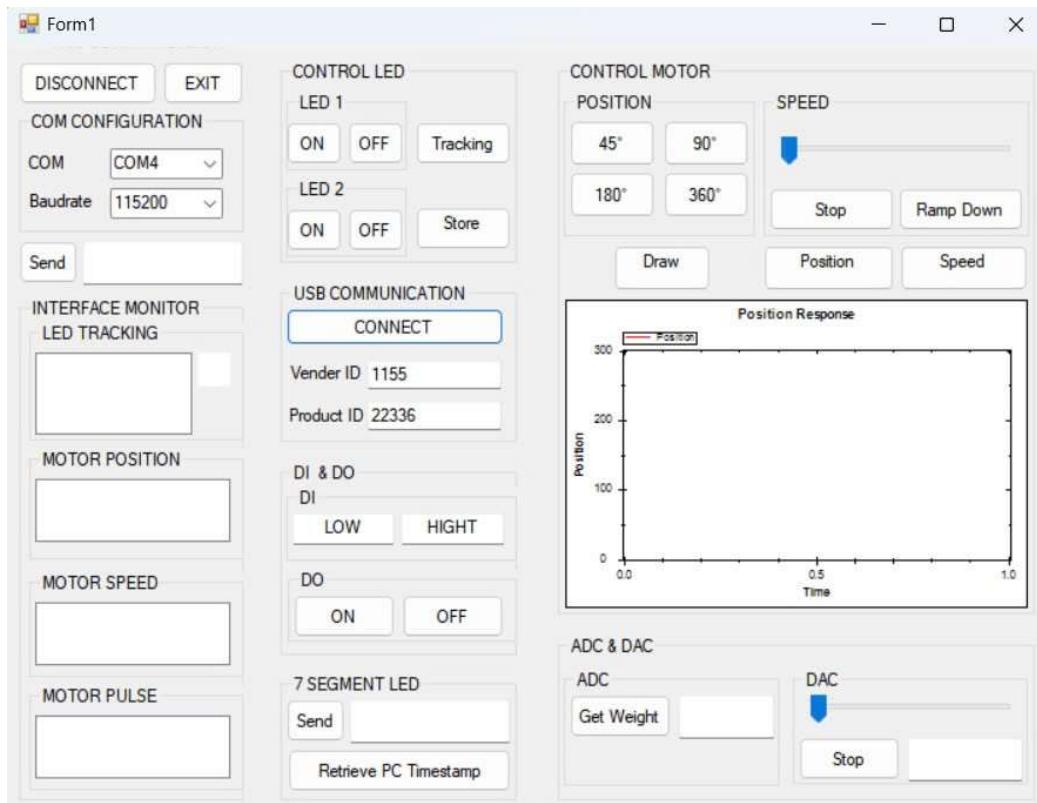


```

void ProcessRxBuffer(void)
{
    if (strcmp((char*)Rx_Buffer, "000000000011") == 0)
    {
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_SET);
        strcpy((char *)TX_Data, "LED Off \r\n");
        int len = strlen((char *)TX_Data);
        CDC_Transmit_FS((uint8_t *)TX_Data, len);
    }
    else if (strcmp((char*)Rx_Buffer, "00000001011") == 0)
    {
        HAL_GPIO_WritePin(GPIOC, GPIO_PIN_13, GPIO_PIN_RESET);
        strcpy((char *)TX_Data, "LED On \r\n");
        int len = strlen((char *)TX_Data);
        CDC_Transmit_FS((uint8_t *)TX_Data, len);
    }
}

```

## 7.4. Kết quả



Form1

**DISCONNECT**

**EXIT**

**COM CONFIGURATION**

COM

Baudrate

**Send**

**INTERFACE MONITOR**

**LED TRACKING**



**MOTOR POSITION**

**MOTOR SPEED**

**MOTOR PULSE**

**CONTROL LED**

**LED 1**

**LED 2**

**USB COMMUNICATION**

**CONNECT**

Vender ID

Product ID

**DI & DO**

**DI**

**DO**

**7 SEGMENT LED**

**Send**

**Retrieve PC Timestamp**

**CONTROL MOTOR**

**POSITION**

**SPEED**

**Stop**

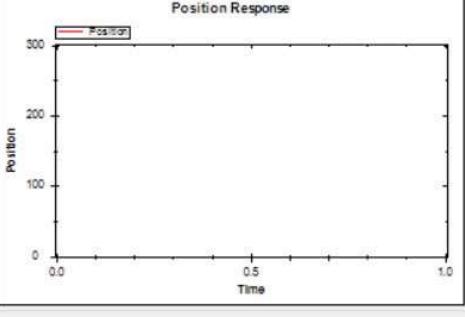
**Ramp Down**

**Draw**

**Position**

**Speed**

**Position Response**



Position

Time

**ADC & DAC**

**ADC**

**Get Weight**

**DAC**

**Stop**

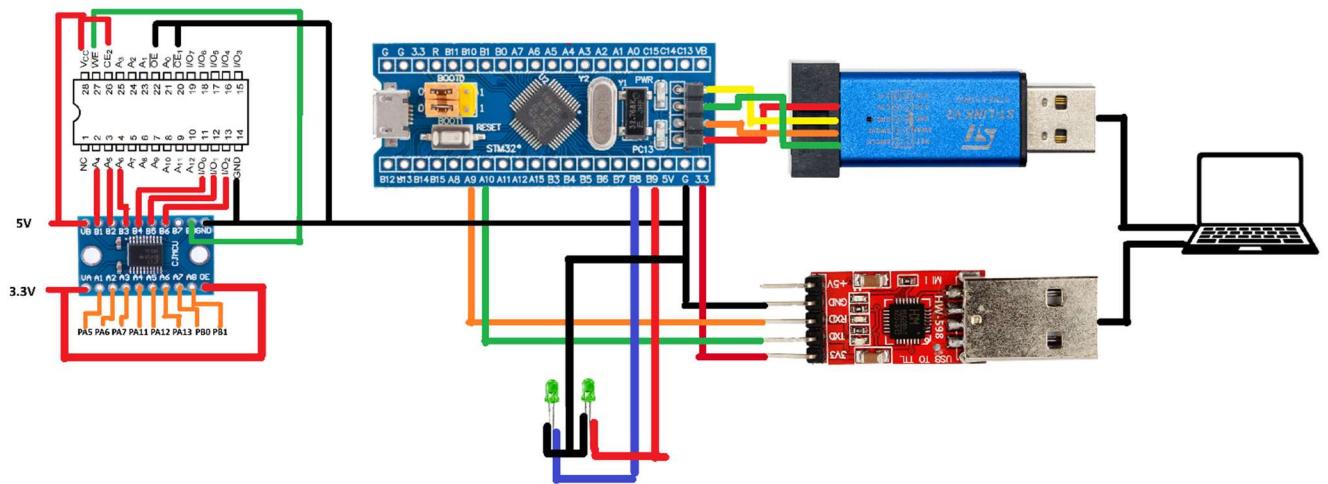
49

## VIII. EXTERNAL RAM

### 8.1. Mục tiêu

Truyền và đọc dữ liệu từ SRAM. Kiểm chứng kết quả bằng cách đọc dữ liệu đã gửi rồi xuất tín hiệu qua LED.

### 8.2. Kết nối phần cứng



```

//=====EXTERNAL RAM=====//
//=====//
//=====

void SRAM_Set_Address(uint8_t addr)
{
//=====CHANGE GPIO TO OUTPUT=====//
void SRAM_Write_Data(uint8_t data)
{
void SRAM_Write_Cycle(uint8_t address, uint8_t data)
{
//=====CHANGE GPIO TO INPUT=====//
uint8_t SRAM_ReadData(void)
{
uint8_t SRAM_Read_Cycle(uint8_t address)
{
//=====CHECK SRAM COMMUNICATION=====//
void LED_Show(uint8_t data)
{
void SRAM_Set_Data()
{
//=====READ DATA FROM SRAM=====//
void SRAM_Read_And_Check_Data()
{

void SRAM_Set_Address(uint8_t addr)
{
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, (addr & 0x01) ? GPIO_PIN_SET : GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_6, (addr & 0x02) ? GPIO_PIN_SET : GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_7, (addr & 0x04) ? GPIO_PIN_SET : GPIO_PIN_RESET);
}

//=====CHANGE GPIO TO OUTPUT=====//
void SRAM_Write_Data(uint8_t data)
{
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_11, (data & 0x01) ? GPIO_PIN_SET : GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_12, (data & 0x02) ? GPIO_PIN_SET : GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOA, GPIO_PIN_13, (data & 0x02) ? GPIO_PIN_SET : GPIO_PIN_RESET);
}

void SRAM_Write_Cycle(uint8_t address, uint8_t data)
{
    SRAM_Set_Address(address);
    SRAM_Write_Data(data);

    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, GPIO_PIN_RESET); // !WE = 0 start write
    HAL_Delay(1);
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, GPIO_PIN_SET); // !WE = 1 end write
    HAL_Delay(1);
}

```

```

uint8_t SRAM_ReadData(void)
{
    GPIO_InitTypeDef GPIO_InitStruct = {0};

    GPIO_InitStruct.Pin = GPIO_PIN_4 | GPIO_PIN_5 | GPIO_PIN_6;
    GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

    uint8_t value = 0;
    value |= HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_4) ? 0x01 : 0x00;
    value |= HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_5) ? 0x02 : 0x00;
    value |= HAL_GPIO_ReadPin(GPIOB, GPIO_PIN_6) ? 0x04 : 0x00;

    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

    return value;
}

uint8_t SRAM_Read_Cycle(uint8_t address)
{
    SRAM_Set_Address(address);

    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_9, GPIO_PIN_SET);
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_8, GPIO_PIN_RESET);
    HAL_Delay(1);

    uint8_t data = SRAM_ReadData();

    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_8, GPIO_PIN_SET);

    return data;
}

//=====CHECK SRAM COMMUNICATION=====
void LED_Show(uint8_t data)
{
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_8, (data & 0x01) ? GPIO_PIN_SET : GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_9, (data & 0x02) ? GPIO_PIN_SET : GPIO_PIN_RESET);
}

void SRAM_Read_And_Check_Data()
{
    Mode_10_Flag = 0;
    for (SRAM_Data_Address = 0; SRAM_Data_Address < 8; SRAM_Data_Address++)
    {
        SRAM_Set_Address(SRAM_Data_Address);
        SRAM_Read_Data = SRAM_Read_Cycle(SRAM_Data_Address);

        LED_Show(SRAM_Read_Data);

        HAL_Delay(500);
    }
}

```

#### **8.4. Kết quả**

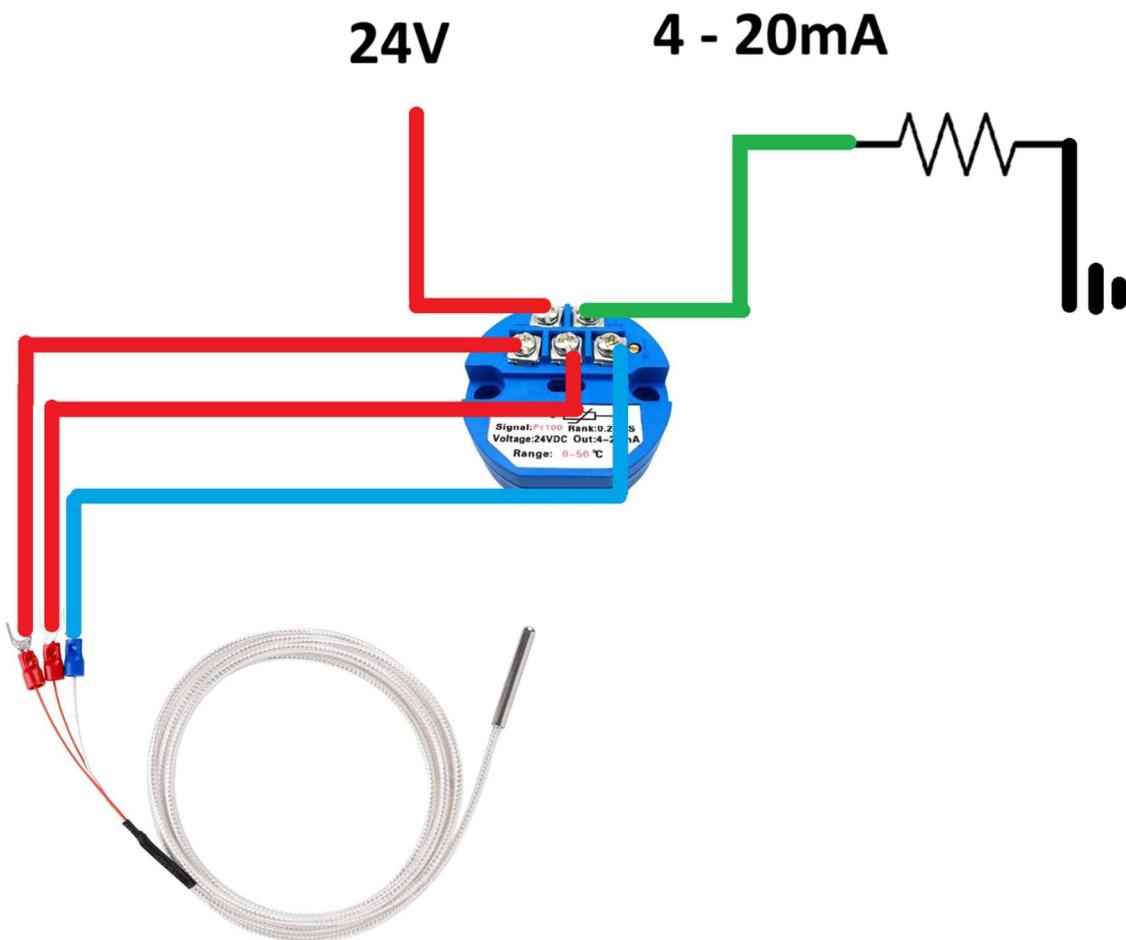
2 LED sáng tương ứng với data đọc về từ SRAM

## IX. 4 – 20mA OUTPUT SENSOR

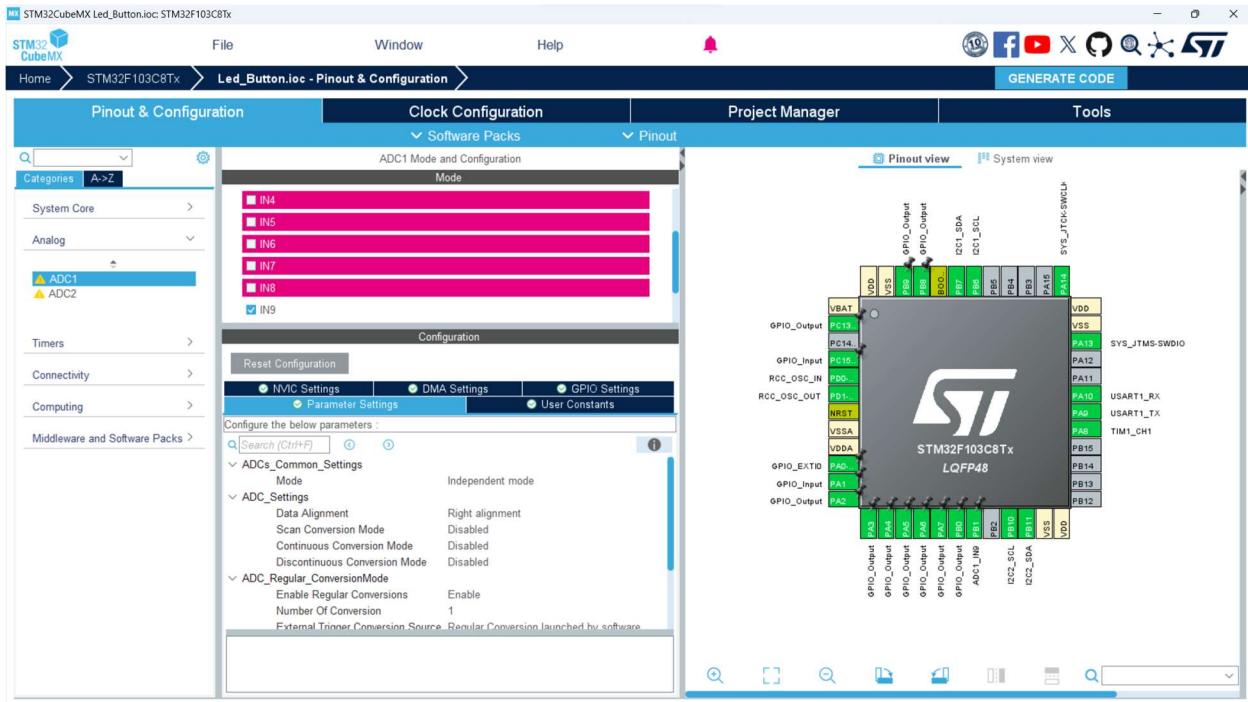
### 9.1. Mục tiêu

Sử dụng cảm biến nhiệt độ công nghiệp (Pt100) cho tín hiệu analog output là dòng 4 – 20mA, chuyển dòng này thành tín hiệu áp và dùng VOM để đo sự thay đổi áp.

### 9.2. Kết nối phần cứng



### 9.3. Cấu hình và lập trình phần mềm



```

HAL_ADC_Start(&hadc1);
HAL_Delay(50);
ADC_Pt100_Zero = HAL_ADC_GetValue(&hadc1);
HAL_ADC_Stop(&hadc1);

```

**9.4. Kết quả:** Chỉ đọc ADC và dùng VOM đo áp, chưa chuyên giá trị ADC thành nhiệt độ.

Watch 1		
Name	Value	Type
RX_Data	0x20000278 RX_Data[] ""	uchar[17]
SRAM_Init_Data	0x08	uchar
SRAM_Case_Flag	0x00000001	int
ADC_Pt100_Zero	1949	uint
<Enter expression>		

HẾT