ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN – ĐIỆN TỬ

Bài tập chương 4
MÔN HỆ THỐNG ĐIỀU KHIỂN NHÚNG
LỚP L01--- HK221

GVHD: **T.S Nguyễn Vĩnh Hảo**
SVTH: **Nguyễn Gia Phúc**
MSSV: **2010528**

TP. Hồ Chí Minh,Ngày 15 Tháng 12 Năm 2022

I. TỔNG QUAN VỀ DỰ ÁN: DAC + UART

DAC

- DAC kênh 1 và 2 lần lượt trên PA4 và PA5.
- DAC1 được kích hoạt bởi TIMER2 (có Ngắt), DAC2 được kích hoạt bởi TIMER7.
- DAC1 được hỗ trợ DMA1, Stream 5, Channel 7; DAC2 thuộc DMA1, Stream 6, Channel 7.

Các hàm cấu hình TIMER

```
void TIM2_Config(uint16_t Frequency, uint16_t Ns);
void TIM7_Config(uint16_t Frequency, uint16_t Ns);
```

Tham số thứ 1: Tần số mong muốn.

Tham số thứ 2: Số lần lấy mẫu của dạng sóng.

Các hàm cấu hình dạng sóng

```
void DAC_Ch1_WaveConfig(uint16_t* value, uint16_t BUFF);
void DAC_Ch2_WaveConfig(uint16_t* value, uint16_t BUFF);
```

Tham số thứ 1: địa chỉ của mảng sóng mong muốn (sóng sin, răng cưa, vuông).

Tham số thứ 2: số lần lấy mẫu của mảng đó.

```
void DAC_Ch1_TriangleConfig(void);
void DAC_Ch2_TriangleConfig(void);
```

Hàm tạo sóng tam giác do sử dụng chế độ có sẵn của DAC, nên giá trị biên độ được cố định.

UART

- UART4 hoạt động cả truyền (TX) và nhận (RX).

- UART4_RX sử dụng DMA1 stream2 channel 4; UART4_TX sử dụng DMA1 stream 4 channel 4.

UART4_RX mỗi lần nhận 8 Byte từ GUI, để cấu hình loại sóng và tần số. Khi nhận đủ 8 Byte, ngắt DMA xảy ra và nhảy vào chương trình cấu hình xung, tần số.

```
void DMA1_Stream2_IRQHandler(void)
{
```

UART4_TX truyền 1 byte là mã ASCII giá trị của thanh ghi DAC_DOR1 mỗi khi ngắt TIMER2 xảy ra.

```
void TIM2_IRQHandler(void) // GUI MA ASCII DAC LÊN UART_TX
{
   uint16_t DAC_value;
   uint8_t String_DAC_value[]="";
   if (TIM_GetITStatus(TIM2, TIM_IT_Update) != RESET)
   {
   DAC_value = DAC->DOR1;
   IntToStr4(DAC_value,String_DAC_value);
   Display1(String_DAC_value, 4);
   }
TIM_ClearITPendingBit(TIM2, TIM_IT_Update);
}
```

1. Digital-to-analog converter (DAC)

- The two 12-bit buffered DAC channels can be used to convert two digital signals into two analog voltage signal outputs.

This dual digital Interface supports the following features:

• two DAC converters: one for each output channel

• 8-bit or 12-bit monotonic output

• left or right data alignment in 12-bit mode

• synchronized update capability

• noise-wave generation

• triangular-wave generation

• dual DAC channel independent or simultaneous conversions

• DMA capability for each channel

• external triggers for conversion

• input voltage reference VREF+

Eight DAC trigger inputs are used in the device. The DAC channels are triggered through

the timer update outputs that are also connected to different DMA streams.

- Vi điều khiển STM32F407VG có 2 kênh DAC trên 2 chân PA4 và PA5.

Table 7. STM32F40xxx pin and ball definitions[1] (continued)

Pin number						Pin name (function after reset)[2]	Pin type	I / O structure	Notes	Alternate functions	Additional functions
LQFP64	WLCSP90	LQFP100	LQFP144	UFBGA176	LQFP176						
	D9			L4	48	BYPASS_REG	I	FT	-	-	-
19	E4	28	39	K4	49	V_{DD}	S	-	-	-	-
20	J9	29	40	N4	50	PA4	I/O	TTa	(5)	SPI1_NSS / SPI3_NSS / USART2_CK / DCMI_HSYNC / OTG_HS_SOF/ I2S3_WS/ EVENTOUT	ADC12_IN4 /DAC_OUT1
21	G8	30	41	P4	51	PA5	I/O	TTa	(5)	SPI1_SCK/ OTG_HS_ULPI_CK / TIM2_CH1_ETR/ TIM8_CH1N/ EVENTOUT	ADC12_IN5/DAC_ OUT2

- Cấu hình chức năng DAC trên 2 chân PA4,PA5 như sau:

```
int main(void)
{
    // Cau hinh chan
  GPIO_InitTypeDef GPIO_InitStructure;
  RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_DAC, ENABLE);
  // 2 kenh DAC PA4,PA5
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_4 | GPIO_Pin_5;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AN; //Chon mode analog
  GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL ;
  GPIO_Init(GPIOA, &GPIO_InitStructure);
```

Nguyên lý hoạt động

- Mỗi lần TIMER tràn, 1 giá trị được DMA chuyển từ bộ nhớ đến thanh ghi DAC_ DHRyyyx của DAC. Sau đó 3 chu kỳ xung APB1 thì giá trị từ thanh ghi DAC_DHRyyyx chuyển sang DAC_DORx và tạo ra điện áp đầu ra trong khoảng 0 đến VREF+. Điện áp đầu ra được tính theo công thức sau:

$$DAC_{Output} = V_{REF}\frac{DOR}{DAC\_MaxDigitalValue + 1} \quad \textbf{(1)}$$

Note:
For right-aligned 12-bit resolution: DAC_MaxDigitalValue = 0xFFF
For right-aligned 8-bit resolution: DAC_MaxDigitalValue = 0Xff

Cấu hình DAC

- Chương trình cấu hình DAC1, kích hoạt bởi TIMER2 và có sử dụng DMA1 Stream5channel7

```
void DAC_Ch1_WaveConfig(uint16_t* value, uint16_t BUFF)
{
  // Cau hinh DAC
  DAC->CR |= (6<<0)|(1<<5);   // Mode kich hoat ngoai TIM2; co bo dem
//Enable DMA1 clock
  RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_DMA1, ENABLE);// cho phep DMA1-CA 2 KENH DAC
// cau hình DMA1 cho DAC1 channel 7 stream 5
  DMA_InitTypeDef DMA_InitStructure;
  DMA_DeInit(DMA1_Stream5);
  DMA_InitStructure.DMA_Channel = DMA_Channel_7;
  DMA_InitStructure.DMA_Memory0BaseAddr = (uint32_t)value;//(uint32_t)sinvalue;
  DMA_InitStructure.DMA_PeripheralBaseAddr = (uint32_t)&(DAC->DHR12R1);//(uint32_t)0x40007408;
  DMA_InitStructure.DMA_DIR = DMA_DIR_MemoryToPeripheral;
  DMA_InitStructure.DMA_BufferSize = BUFF;
  DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
  DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Enable;
  DMA_InitStructure.DMA_PeripheralDataSize = DMA_PeripheralDataSize_HalfWord;//do phan giai DAC
  DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_HalfWord;
  DMA_InitStructure.DMA_Mode = DMA_Mode_Circular; //DMA_Mode_Circular;
  DMA_InitStructure.DMA_Priority = DMA_Priority_High;
  DMA_InitStructure.DMA_FIFOMode = DMA_FIFOMode_Disable;
  DMA_InitStructure.DMA_FIFOThreshold = DMA_FIFOThreshold_HalfFull;
  DMA_InitStructure.DMA_MemoryBurst = DMA_MemoryBurst_Single;
  DMA_InitStructure.DMA_PeripheralBurst = DMA_PeripheralBurst_Single;
  DMA_Init(DMA1_Stream5, &DMA_InitStructure);
  DAC_Cmd (DAC_Channel_1, ENABLE);
}
```

- Chương trình 2 tương tự, kích hoạt bởi TIMER7 và sử dụng DMA1 Stream 5 channel 6.

Cấu hình TIMER cho DAC và lẩy mảng giá trị sóng

- Tần số kích hoạt TIMER (f_TimerTRGO)

Tần số đếm cơ bản phụ thuộc vào tần số APB1, APB2 timer clock và có thể chia cho tham số PRESCALER và AUTORELOAD.

```
void TIM2_Config(uint16_t Frequency, uint16_t Ns)
{
// Cau hinh TIMER 2
  TIM_TimeBaseInitTypeDef    TIM_TimeBaseStructure;
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);
  TIM_TimeBaseStructInit(&TIM_TimeBaseStructure);
  TIM_TimeBaseStructure.TIM_Prescaler = 84-1;
  TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
  TIM_TimeBaseStructure.TIM_Period = 1000000/(Frequency*Ns)-1;
  TIM_TimeBaseStructure.TIM_ClockDivision = 0;
  TIM_TimeBaseInit(TIM2, &TIM_TimeBaseStructure);
  TIM_SelectOutputTrigger(TIM2, TIM_TRGOSource_Update); // THEM
  NVIC_InitTypeDef NVIC_InitStructure;
  TIM_Cmd(TIM2, ENABLE);
  NVIC_InitStructure.NVIC_IRQChannel = TIM2_IRQn;
  NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
  NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
  NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
  NVIC_Init(&NVIC_InitStructure);
}
```

Cách tạo từng loại tín hiệu

- **Sóng sin**

Số lần lấy mẫu: ns — Bước lấy mẫu: 2pi/ns

Công thức tính giá trị digital input của sóng sin:

$$y_{SineDigital}(x) = \left(\sin\left(x \cdot \frac{2\pi}{n_s}\right) + 1\right)\left(\frac{(0xFFF + 1)}{2}\right)$$

Tần số sóng sin:

$$f_{Sinewave} = \frac{f_{TimerTRGO}}{n_s}$$

Chương trình lấy 100 mẫu của sóng sin, và tạo mảng sóng sin như đoạn mã dưới.

```
#define   BUFF_SIZE_DAC_SIN     100   // SO LAN LAU MAU SONG SIN
void get_sinvalue_Ch1 ( float voltage);
uint16_t sinvalue_Ch1[BUFF_SIZE_DAC_SIN];
```

```
void get_sinvalue_Ch1 ( float voltage)  // TAO MANG SONG SIN CHANNEL 1
{
  for(int i =0;i<BUFF_SIZE_DAC_SIN;i++)
  {
    sinvalue_Ch1[i] = (sin(i*2*PI/BUFF_SIZE_DAC_SIN)+1)*(((0xFFF+1)*voltage)/(2*4*1.05));
  }
}
```

- **Sóng Răng Cưa**

$$y_{Digital(x)} = \frac{x}{n_s}\ 0xFFF$$

Chương trình lấy 100 mẫu.

```
#define   BUFF_SIZE_DAC_SRC     100 // SO LAN LAU MAU SONG RANG CUA
void get_srcvalue_Ch1( float voltage);
uint16_t srcvalue_Ch1[BUFF_SIZE_DAC_SRC];
```

```
void get_srcvalue_Ch1( float voltage) // TAO MANG SONG RANG CUA
{
  for(int i =0;i<BUFF_SIZE_DAC_SRC;i++)
  {
    srcvalue_Ch1[i] = (i)*(((0xFFF)*voltage)/(4*1.01*(BUFF_SIZE_DAC_SRC-1)));
  }
}
```

- **Sóng Vuông**

Chương trình lấy 100 mẫu sóng vuông: 50 mẫu đầu có giá trị 0, 50 mẫu sau tỉ lệ với điện áp.

```
#define   BUFF_SIZE_DAC_SV      100   // SO LAN LAU MAU SONG VUONG
void get_svvalue_Ch1( float voltage);
uint16_t svvalue_Ch1[BUFF_SIZE_DAC_SV];
```

```
void get_svvalue_Ch1( float voltage) // TAO MANG SONG VUONG
{
  for(int i =0;i<BUFF_SIZE_DAC_SV/2;i++)
  {
    svvalue_Ch1[i] = 0;
  }

  for(int i =BUFF_SIZE_DAC_SV/2;i<BUFF_SIZE_DAC_SV;i++)
  {
    svvalue_Ch1[i] = (((0xFFF)*voltage)/(4*1.08));
  }
}
```

- **Sóng tam giác**

Sóng tam giác được tạo ra nhờ sử dụng chế độ có sẵn của DAC.

Trong chương trình này cố định giá trị biên độ là 2046 = 1023(base) + 1023.

Gía trị digital sẽ tăng từ 1023 (base) lên 1 sau mỗi lần kích hoạt TIMER, và khi tăng thêm đủ 1023 giá trị, nó sẽ giảm 1 cho đến khi còn 1023(base) và tiếp tục tăng lại như trên.

```
void DAC_Ch1_TriangleConfig(void)
{
  DAC_InitTypeDef DAC_InitStructure;
 /* DAC channel2 Configuration */
  DAC_InitStructure.DAC_Trigger = DAC_Trigger_T2_TRGO;
  DAC_InitStructure.DAC_WaveGeneration = DAC_WaveGeneration_Triangle;
  DAC_InitStructure.DAC_LFSRUnmask_TriangleAmplitude = DAC_TriangleAmplitude_1023;
  DAC_InitStructure.DAC_OutputBuffer = DAC_OutputBuffer_Enable;
  DAC_Init(DAC_Channel_1, &DAC_InitStructure);
  /* Enable DAC Channel2 */
  DAC_Cmd(DAC_Channel_1, ENABLE);
  /* Set DAC channel2 DHR12RD register */
  DAC_SetChannel2Data(DAC_Align_12b_R, 1023);
}
```

- Chương trình ngắt UART_RX có sử dụng DMA khi nhận đủ 8 byte chứa các thông tin cần thiết để cấu hình xung.

Khi ngắt xảy ra, xóa cờ và cho phép DMA hoạt động ngắt lại.

```
void DMA1_Stream2_IRQHandler(void)  //NHAN du 8 BYTE se c
{
  /* Clear the DMA1_Stream2 TCIF2 pending bit */
  DMA_ClearITPendingBit(DMA1_Stream2, DMA_IT_TCIF2);
  DMA_Cmd(DMA1_Stream2, ENABLE);
  /* Kiem tra khi nao xay ra ngat */
```

Nếu chi phép xuất xung, data_Rx[2] = 1 thì chuyển giá trị biên độ và tần số từ mã ASCII về kiểu int. Sau đó xét byte data_Rx[7] chứa thông tin về dạng sóng.

```
if(!(data_Rx[2] - 0x31))
{ // TAN SO
  for(int i =0; i<4 ;i++)
  {
  Rx_Hz[i]=data_Rx[i+3];
  }
  Hz1=ATD(Rx_Hz,4);
  //BIEN DO
  for(int i=0;i<2;i++)
  {
  Rx_V[i]=data_Rx[i];
  }
  V1=ATD(Rx_V,2);
  // CAU HINH XUNG
  switch(data_Rx[7]-0x30)     //Chon kenh cau hinh
  {
    case 0:         //SONG SIN
    TIM2_Config(Hz1, BUFF_SIZE_DAC_SIN);
    get_sinvalue_Ch1(V1/10);
    DAC_Ch1_WaveConfig(sinvalue_Ch1, BUFF_SIZE_DAC_SIN);
    get_wavevalue_Ch2(V1/10);
    TIM7_Config(Hz1, BUFF_SIZE_DAC_SIN);
    DAC_Ch2_WaveConfig(wavevalue_Ch2,2);
    break;

    case 1:       //SONG VUONG
    get_svvalue_Ch1(V1/10);
    DAC_Ch1_WaveConfig(svvalue_Ch1, BUFF_SIZE_DAC_SV);
```

```
case 2:   //  SONG RANG CUA
TIM2_Config(Hz1, BUFF_SIZE_DAC_SRC);
get_srcvalue_Ch1(V1/10);
DAC_Ch1_WaveConfig(srcvalue_Ch1, BUFF_SIZE_DAC_SRC);
get_wavevalue_Ch2(V1/10);
TIM7_Config(Hz1, BUFF_SIZE_DAC_SRC);
DAC_Ch2_WaveConfig(wavevalue_Ch2,2);
break;

case 3:   //SONG TAM GIAC
TIM2_Config(Hz1,2000);
DAC_Ch1_TriangleConfig();
get_wavevalue_Ch2(V1/10);
TIM7_Config(Hz1, 2046);
DAC_Ch2_WaveConfig(wavevalue_Ch2,2);
break;
default:  // KHONG XUAT XUNG

break;
```

- Khi ngừng xuất xung, reset DAC và TIMER. Trong lúc này vẫn hoạt động UART4_Rx

```
else
{
DAC_DeInit();
TIM_DeInit(TIM2);
}
```

2. Universal synchronous/asynchronous receiver transmitters (USART)

- The STM32F407xx embed four universal synchronous/asynchronous receiver transmitters (USART1, USART2, USART3 and USART6) and two universal asynchronous receiver transmitters (UART4 and UART5).

Table 9. Alternate function map

Port		AF0	AF1	AF2	AF3	AF4	AF5	AF6	AF7	AF8
		SYS	TIM1/2	TIM3/4/5	TIM8/9/10 /11	I2C1/2/3	SPI1/SPI2/ I2S2/I2S2e xt	SPI3/I2Sext /I2S3	USART1/2/3/ I2S3ext	UART4/5/ USART6
	PA0	-	TIM2_CH1_ ETR	TIM 5_CH1	TIM8_ETR	-	-	-	USART2_CTS	UART4_TX
	PA1	-	TIM2_CH2	TIM5_CH2	-	-	-	-	USART2_RTS	UART4_RX

Đoạn chương trình chọn chức năng UART4_TX trên chân PA0 và UART4_RX trên chân PA1.

```
GPIO_InitTypeDef  GPIO_InitStructure;
USART_InitTypeDef USART_InitStructure;
DMA_InitTypeDef   DMA_InitStructure_Rx;
NVIC_InitTypeDef  NVIC_InitStructure;
/* Enable GPIO clock */
RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_GPIOA, ENABLE);
/* Enable UART clock */
RCC_APB1PeriphClockCmd(RCC_APB1Periph_UART4, ENABLE);
/* Enable DMA1 clock */
RCC_AHB1PeriphClockCmd(RCC_AHB1Periph_DMA1, ENABLE);
/* Connect UART4 pins to AF2 */
GPIO_PinAFConfig(GPIOA, GPIO_PinSource0, GPIO_AF_UART4);
GPIO_PinAFConfig(GPIOA, GPIO_PinSource1, GPIO_AF_UART4);
/* GPIO Configuration for UART4 Tx */
GPIO_InitStructure.GPIO_Pin   = GPIO_Pin_0;
GPIO_InitStructure.GPIO_Mode  = GPIO_Mode_AF;
GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
GPIO_InitStructure.GPIO_PuPd  = GPIO_PuPd_UP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_Init(GPIOA, &GPIO_InitStructure);
/* GPIO Configuration for USART Rx */
GPIO_InitStructure.GPIO_Pin   = GPIO_Pin_1;
GPIO_InitStructure.GPIO_Mode  = GPIO_Mode_AF;
GPIO_Init(GPIOA, &GPIO_InitStructure);
```

Cấu hình cho UART4

- UART4 configured as follow: BaudRate = 115200 baud, Word Length = 8 Bits, One Stop Bit1, No parity, Hardware flow control disabled (RTS and CTS signals), Receive and transmit enabled

```
USART_InitStructure.USART_BaudRate = 115200;
USART_InitStructure.USART_WordLength = USART_WordLength_8b;
USART_InitStructure.USART_StopBits = USART_StopBits_1;
USART_InitStructure.USART_Parity = USART_Parity_No;
USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
USART_Init(UART4, &USART_InitStructure);
/* Enable USART */
USART_Cmd(UART4, ENABLE);
```

Cấu hình DMA cho UART4_TX

UART4_Tx thuộc DMA1, stream 4, channel 4. Chiều truyền dữ liệu từ bộ nhớ ra ngoại vi.

```
USART_DMACmd(UART4, USART_DMAReq_Tx, ENABLE);
  /* DMA1 Stream4 Channel4 for UART4 Tx configuration */
DMA_InitStructure.DMA_Channel = DMA_Channel_4;
DMA_InitStructure.DMA_PeripheralBaseAddr = (uint32_t)&UART4->DR;
DMA_InitStructure.DMA_DIR = DMA_DIR_MemoryToPeripheral;
DMA_InitStructure.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
DMA_InitStructure.DMA_MemoryInc = DMA_MemoryInc_Enable;
DMA_InitStructure.DMA_PeripheralDataSize = DMA_PeripheralDataSize_Byte;
DMA_InitStructure.DMA_MemoryDataSize = DMA_MemoryDataSize_Byte;
DMA_InitStructure.DMA_Mode = DMA_Mode_Normal;
DMA_InitStructure.DMA_Priority = DMA_Priority_High;
DMA_InitStructure.DMA_FIFOMode = DMA_FIFOMode_Disable;
DMA_InitStructure.DMA_FIFOThreshold = DMA_FIFOThreshold_HalfFull;
DMA_InitStructure.DMA_MemoryBurst = DMA_MemoryBurst_Single;
DMA_InitStructure.DMA_PeripheralBurst = DMA_PeripheralBurst_Single;
```

```
void Display1( uint8_t* Addr_txbuff, uint16_t BUFF )
{
    DMA_InitStructure.DMA_Memory0BaseAddr = (uint32_t)Addr_txbuff;
    DMA_InitStructure.DMA_BufferSize = BUFF;//BUFF_SIZE_UART;
    DMA_Init(DMA1_Stream4, &DMA_InitStructure);
    DMA_ClearFlag(DMA1_Stream4, DMA_FLAG_TCIF4);
    DMA_Cmd(DMA1_Stream4, ENABLE);    // phai cho phep truyen lai
    while(DMA_GetFlagStatus(DMA1_Stream4, DMA_FLAG_TCIF4)  == RESET );
}
```

Cấu hình DMA cho UART4_RX, có ngắt DMA mỗi khi nhận đủ 8 byte

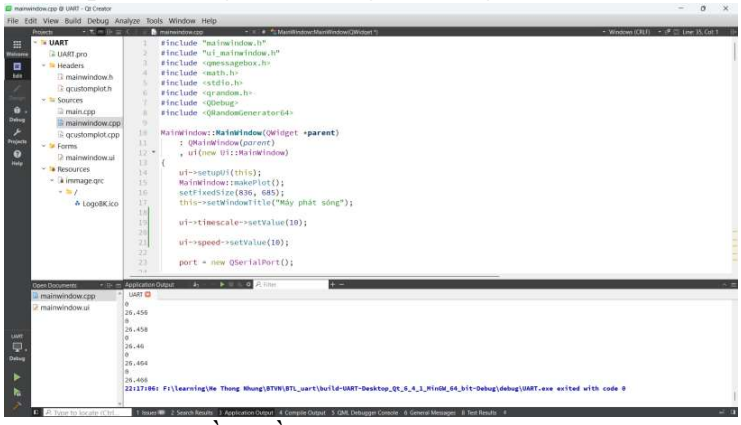
UART4_Rx có ngắt DMA1, stream 2, channel 4. Chiều truyền dữ liệu từ ngoại vi vào bộ nhớ.

```
USART_DMACmd(UART4, USART_DMAReq_Rx, ENABLE);
/* DMA1 Stream2 Channel4 for USART4 Rx configuration */
DMA_InitStructure_Rx.DMA_Channel = DMA_Channel_4;
DMA_InitStructure_Rx.DMA_PeripheralBaseAddr = (uint32_t)&UART4->DR;
DMA_InitStructure_Rx.DMA_Memory0BaseAddr = (uint32_t)&data_Rx;
DMA_InitStructure_Rx.DMA_DIR = DMA_DIR_PeripheralToMemory;
DMA_InitStructure_Rx.DMA_BufferSize = BUFF_SIZE_RX; //
DMA_InitStructure_Rx.DMA_PeripheralInc = DMA_PeripheralInc_Disable;
DMA_InitStructure_Rx.DMA_MemoryInc = DMA_MemoryInc_Enable;
DMA_InitStructure_Rx.DMA_PeripheralDataSize = DMA_PeripheralDataSize_Byte;
DMA_InitStructure_Rx.DMA_MemoryDataSize = DMA_MemoryDataSize_Byte;
DMA_InitStructure_Rx.DMA_Mode = DMA_Mode_Normal;//DMA_Mode_Circular;
DMA_InitStructure_Rx.DMA_Priority = DMA_Priority_High;
DMA_InitStructure_Rx.DMA_FIFOMode = DMA_FIFOMode_Disable;
DMA_InitStructure_Rx.DMA_FIFOThreshold = DMA_FIFOThreshold_HalfFull;
DMA_InitStructure_Rx.DMA_MemoryBurst = DMA_MemoryBurst_Single;
DMA_InitStructure_Rx.DMA_PeripheralBurst = DMA_PeripheralBurst_Single;
DMA_Init(DMA1_Stream2, &DMA_InitStructure_Rx);
DMA_Cmd(DMA1_Stream2, ENABLE);
/* Enable DMA Interrupt to the highest priority */
NVIC_InitStructure.NVIC_IRQChannel = DMA1_Stream2_IRQn;
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 15;
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&NVIC_InitStructure);
/* Transfer complete interrupt mask */
DMA_ITConfig(DMA1_Stream2, DMA_IT_TC, ENABLE);
```

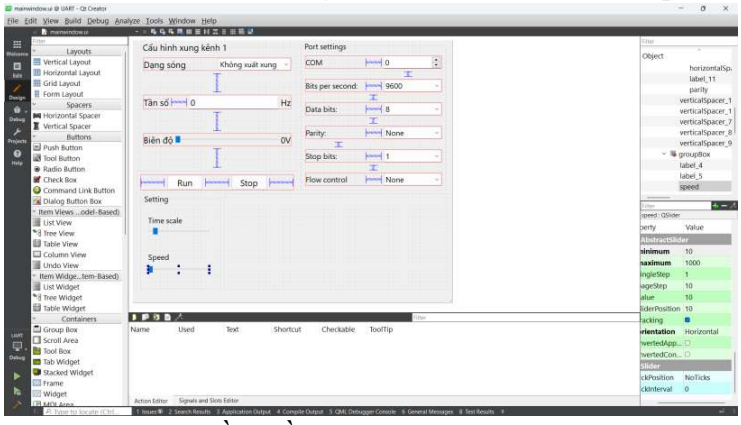
II. BÁO CÁO PHẦN MỀM GIAO TIẾP GIỮA MÁY TÍNH VỚI STM32F407 DÙNG ĐỂ PHÁT SÓNG

1. **Giới thiệu sơ lược về Qt**

   Qt là một khung ứng dụng đa nền tảng và bộ công cụ tiện ích để tạo giao diện người dùng đồ họa cổ điển và nhúng, và các ứng dụng chạy trên nhiều nền tảng phần mềm và phần cứng khác nhau hoặc ít thay đổi trong codebase cơ bản, trong khi vẫn là một ứng dụng gốc với khả năng và tốc độ cục bộ. Do đó nhóm chúng em sử dụng Qt để phát triển giao diện người dùng đồ họa (GUI).



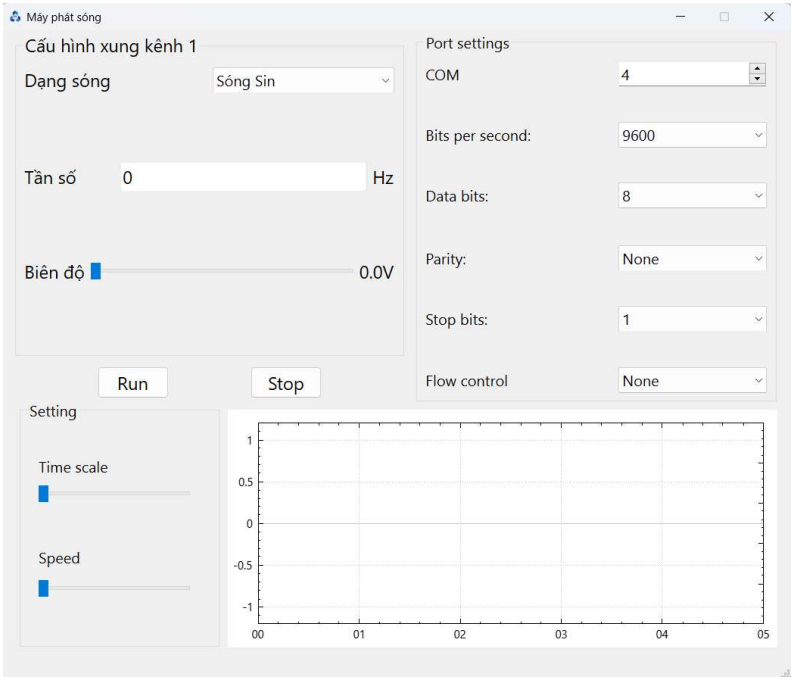
Hình 1: Phần mềm Qt Creater mainwindow.ccp



Hình 2: Phần mềm Qt Creater mainwindow.ui

2. **Giới thiệu về GUI "Máy phát sóng"**

   Sau khi tìm hiểu và sử dụng Qt nhóm chúng em đã tạo ra 1 phần mềm dùng để giao tiếp giữa máy tính với kit stm32f407 discovery và đặt tên là "Máy phát sóng".

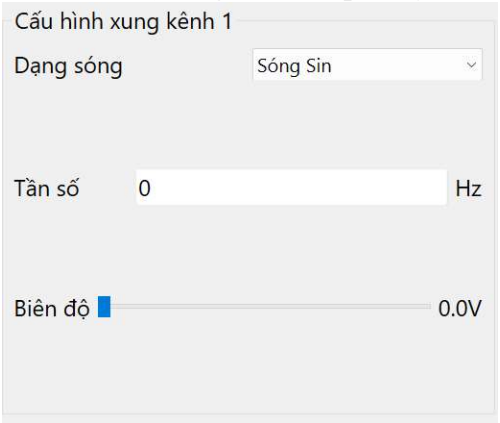


Hình 3: Giao diện của GUI "Máy phát sóng"

3. Các chức năng có trong GUI

GUI có 3 group box là Cấu hình xung kênh 1, Port settings và Setting.

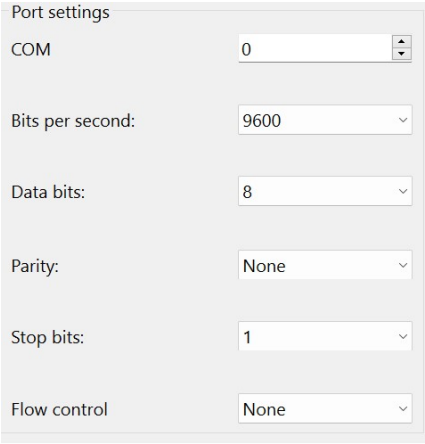
Trong group box Cấu hình xung kênh 1 ta có thể chỉnh được dạng sóng biên độ và tần số cần phát. Biên độ lớn nhất là 4V và nhỏ nhất là 0V, độ chia nhỏ nhất là 0.1V. Tần số lớn nhất là 5000Hz và nhỏ nhất là 0Hz. Nếu nhập giá trị lớn hơn 5000 thì hiển thị là 5000. Nếu nhập giá trị âm hoặc ký tự bất kỳ không phải số sẽ hiện cảnh báo (tham khảo hình 5) và giá trị nhập chuyển thành 0.



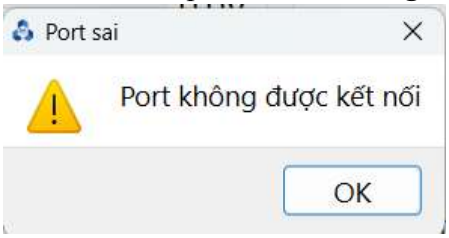
Hình 4: Group box Cấu hình xung kênh 1

Hình 5: Cảnh báo khi nhập ký tự không phải số

Trong group box Port settings (hình 6) ta có thể cấu hình giao tiếp UART. Nếu không kết nối được với cổng com đã cài đặt thì sẽ hiện thông báo như hình 7.

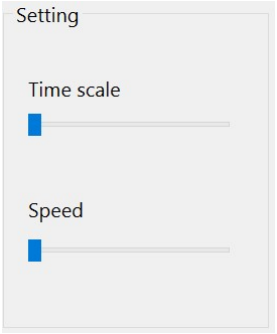


Hình 6: Group box Port settings

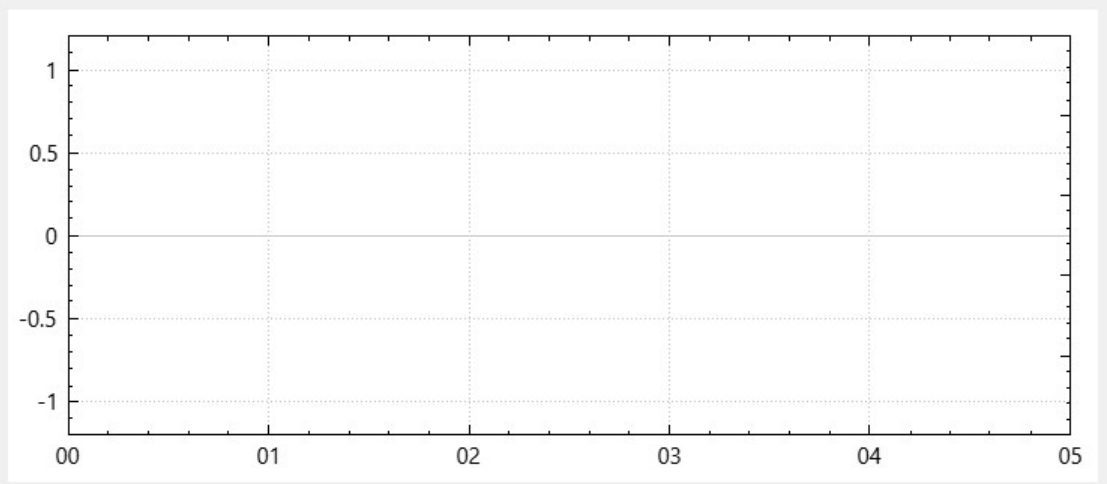


Hình 7: Cảnh báo khi không kết nối được với port

Trong group box Setting (hình 8) ta có thể điều chỉnh speed và timescale của dạng sóng vẽ trên cửa sổ vẽ đồ thị (hình 9). Cửa sổ vẽ đồ thị dùng để vẽ đồ thị của dữ liệu do UART gửi về.

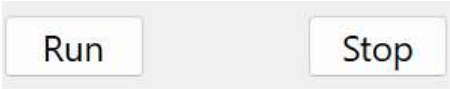


Hình 8: Group box Setting



Hình 9: Cửa sổ vẽ đồ thị

Ngoài ra, trong GUI còn có 2 nút Run và Stop (hình 10). Khi nhấn Run GUI sẽ gửi dữ liệu về biên độ dạng sóng và tần số xuống cho kit stm32f407 discovery xử lý. Ngoài ra khi nhấn Run ta cũng vẽ lại dạng sóng trên cửa sổ vẽ đồ thị đồng thời nó sẽ tắt group box Cấu hình xung kênh 1 và Port settings không cho phép chỉnh sửa trên 2 group này.. Nút Stop có chức năng dừng phát sóng. Khi nhấn nút Stop Gui sẽ gửi dữ liệu yêu cầu dừng phát sóng cho kit stm32f407 discovery đồng thời nó sẽ bật group box Cấu hình xung kênh 1 và Port settings cho phép chỉnh sửa trên 2 group này.



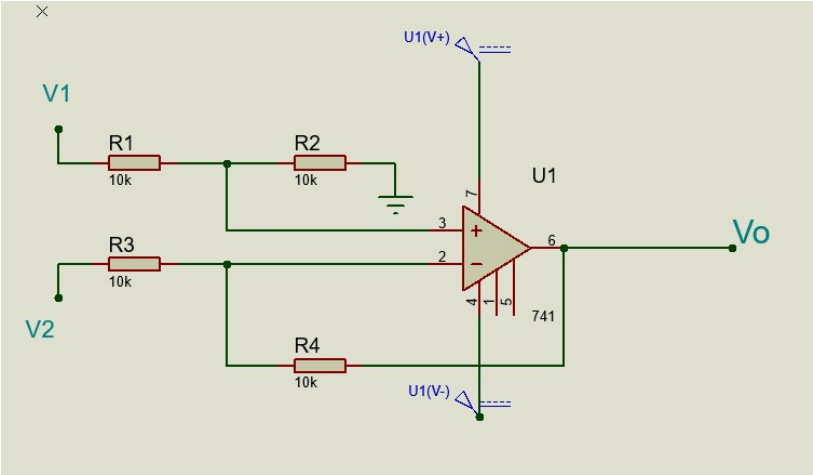
Hình 10: Nút Run và Stop

III. PHẦN CỨNG

Yêu cầu: Dạng sóng ngõ ra phải đạt được tầm giá trị từ -4V đến 4V

1) Mạch trừ

Sau khi xuất sóng từ vi điều khiển, ta sẽ trừ điện áp xuống sao cho sóng dao động quanh giá trị 0

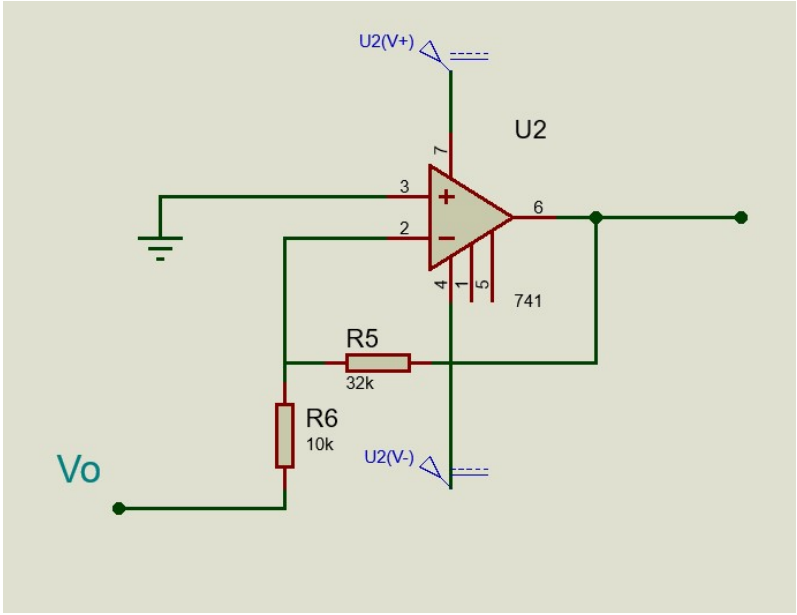


$$V_o = -\frac{R1}{R2}(V_2 - V_1) = -\frac{R4}{R3}(V_2 - V_1)$$

Với yêu cầu chỉ cần trừ điện áp nên hệ số $\frac{R1}{R2} = \frac{R4}{R3} = 1$, ta chọn 4 điện trở đó cùng giá trị bằng 10. Ta cấp sóng xuất ra từ vi điều khiển vào ngõ V_2 và tín hiệu cần trừ sau khi tính toán trong vi xử lý xuất ra và ngõ V_1. Lưu ý rằng ngõ ra Vo đang có giá trị đảo lại với giá trị ta mong muốn.

2) Mạch khuếch đại đảo

Tín hiệu Vo từ mạch trừ qua mạch khuếch đại đảo cho ra giá trị trong tầm từ -4V đến 4V

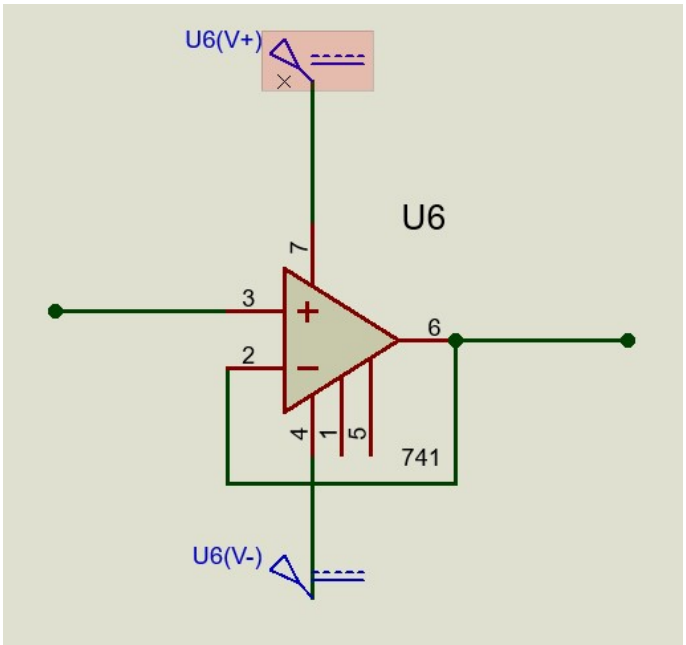


$$V_{out} = -\frac{R5}{R6} V_o$$

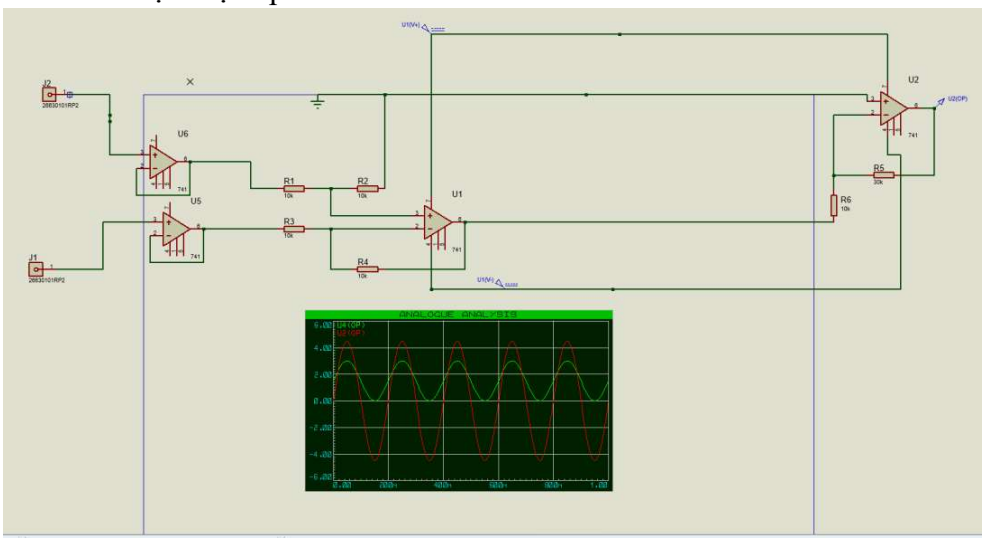
Tín hiệu ngõ ra sẽ có tầm từ -1.4V đến 1.4V theo như kiểm chứng thực tế, vậy ta cần nâng lên tầm 3.2 lần nên ta chọn điện trở R5 là 32k và R6 là 10k. Và sau khi đảo lại khi qua mạch khuếch đại đảo này, ta sẽ được sóng mong muốn.

3) Mạch đệm

Vì khi ta kết nối giữa vi điều khiển và mạch trừ , trở kháng đầu ra của vi xử lý cao mà trở kháng đầu vào của mạch trừ lại thấp, xảy ra hiện tượng không đáp ứng đủ điện áp cần. Bộ đệm điện áp được kết nối giữa hai mạch này ngăn không cho mạch trở kháng đầu vào thấp (mạch trừ) tải mạch thứ nhất thứ nhất (vi xử lý).



Lưu ý: ta cần phải cấp năng lượng nuôi cho cả 4 con opamp, thiết kế 2 pin 9V sao cho có được điện áp 9V và -9V



Mạch mô phỏng