ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

BÁO CÁO

THIẾT KẾ MÁY HIỂN THỊ DẠNG SÓNG

Họ và tên: Lý Bảo Khánh 21020920

1. Đề tài thực nghiệm

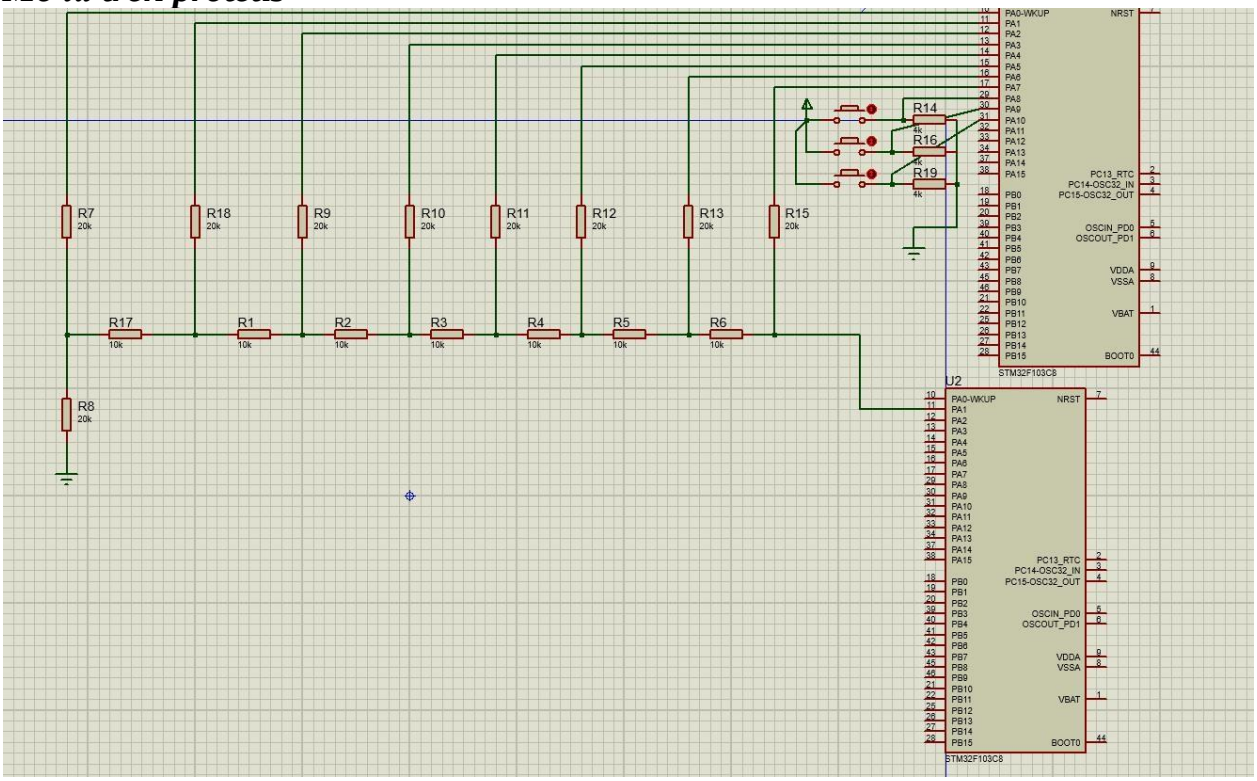
Thiết kế và thực thi thiết bị hiển thị dạng sóng (oscilloscope) đơn kênh, cho phép đo hiện thị dạng sóng trên máy tính sử dụng VĐK và bộ ADC có sẵn trên nó. Thiết bị có thể chỉnh được thang đo volt/div và time/div; chỉnh được đồng bộ tín hiệu. Cho biết dải tần hoạt động của thiết bị? Nó phụ thuộc vào yếu tố nào?

Linh kiện sử dụng:

- Vi điều khiển STM32
- CP2102 USB-to-UART Bridge Controller
- Máy phát chức năng đơn giản (Bài2)
- Màn hình hiển thị (Laptop)

2. Máy hiện thị dạng sóng

1. Mô tả trên proteus



Nguyên lý mạch:

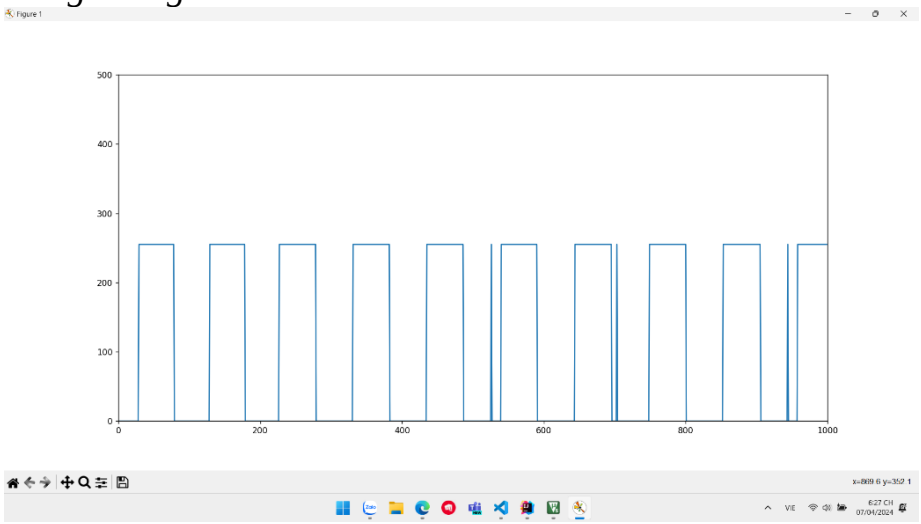
- Lấy tín hiệu xung từ máy phát xung đơn giản làm input vào PA0. Cấu hình PA0 như 1 chân để nhận tín hiệu analog rồi truyền vào mạch ADC được cài đặt sẵn trong STM32.
- Lập trình STM32 để nhận tín hiệu analog rồi chuyển thành digital. Lấy tín hiệu digital truyền qua giao tiếp UART đến máy tính để hiển thị xung analog.
- Do STM32F103C8T6 không hỗ trợ truyền trực tiếp bằng giao tiếp

UART-to-USB nên sử dụng CP2102 là cầu nối để truyền digital với 2 chân Rx, Tx ở STM32 lần lượt là PA9, PA10.

- Dùng code Python trên máy tính để đọc data từ Serial COM rồi vẽ lại dạng xung

2. *Mô tả thực nghiệm*

Tín hiệu xung vuông:



3. *Code*

Code hiển thị xung (Python)

```
import serial
import matplotlib.pyplot as plt
import numpy as np

ser = serial.Serial(
    port='COM11',
    baudrate=9600,
    bytesize=serial.EIGHTBITS,
    parity=serial.PARITY_NONE,
    stopbits=serial.STOPBITS_ONE
)

plt.figure()
plt.ion()
plt.show()

data = []
```

```
while True:
        ser.flushInput()
        bytes_data = ser.read(1)
        value = int.from_bytes(bytes_data, byteorder='little')
        data.append(value)
        print(value)
        if len(data) > 1000:
            data = data[-1000:]
        plt.cla()
        plt.plot(data)
        plt.xlim(0, 1000)  # Điều chỉnh time/div tại đây
        plt.ylim(0, 500) # Điều chỉnh volt/div tại đây
        plt.pause(0.01)
```

Code ADC và truyền UART

```
#include "stm32f10x.h"

#define ADC1_CR2_ADON (1 << 0)
#define ADC1_CR2_CONT (1 << 1)
#define ADC1_SMPR2_SMP1 (7 << 3)
#define ADC1_SQR3_SQ1 (1 << 0)
#define USART1_CR1_UE (1 << 13)
#define USART1_CR1_TE (1 << 3)
#define USART1_CR1_RE (1 << 2)
#define USART1_BRR_9600 ((uint16_t)0x341)
#define ADC1_DR_Address ((uint32_t)0x4001244C)

volatile uint16_t ADCConvertedValue;

void DMA_Config(void) {
   RCC->AHBENR |= RCC_AHBENR_DMA1EN;
   DMA1_Channel1->CCR = 0;
   DMA1_Channel1->CNDTR = 0;
   DMA1_Channel1->CPAR = 0;
   DMA1_Channel1->CMAR = 0;
   DMA1_Channel1->CPAR = ADC1_DR_Address;
   DMA1_Channel1->CMAR = (uint32_t)&ADCConvertedValue;
```

```
    DMA1_Channel1->CNDTR = 1;
    DMA1_Channel1->CCR |= DMA_CCR1_MINC | DMA_CCR1_PSIZE_0
| DMA_CCR1_MSIZE_0 | DMA_CCR1_TEIE | DMA_CCR1_EN;
}

void ADC_Config(void) {
    RCC->APB2ENR |= RCC_APB2ENR_ADC1EN |
RCC_APB2ENR_IOPAEN;
    GPIOA->CRL &= ~GPIO_CRL_CNF1;
    GPIOA->CRL &= ~GPIO_CRL_MODE1;
    ADC1->CR2 |= ADC1_CR2_ADON | ADC1_CR2_CONT;
    ADC1->SMPR2 |= ADC1_SMPR2_SMP1;
    ADC1->SQR3 |= ADC1_SQR3_SQ1;
    ADC1->CR2 |= ADC_CR2_DMA;
}

void USART_Config(void) {
    RCC->APB2ENR |= RCC_APB2ENR_USART1EN |
RCC_APB2ENR_IOPAEN;
    GPIOA->CRH |= GPIO_CRH_MODE9 | GPIO_CRH_CNF9_1;
    GPIOA->CRH &= ~GPIO_CRH_MODE10;
    GPIOA->CRH &= ~GPIO_CRH_CNF10_0;
    USART1->CR1 |= USART1_CR1_UE | USART1_CR1_TE |
USART1_CR1_RE;
    USART1->BRR = USART1_BRR_9600;
}

void USART_SendData(uint16_t Data) {
    ADC1->CR2 |= ADC1_CR2_ADON;
    while(!(ADC1->SR & ADC_SR_EOC));
    while(!(USART1->SR & USART_SR_TXE));
    USART1->DR = (Data >> 4) & (uint16_t)0xFFFF;
}

int main(void) {
    DMA_Config();
    ADC_Config();
    USART_Config();
    while(1) {
        USART_SendData(ADC1->DR);
```

```
    }
}
```