ĐẠI HỌC QUỐC GIA HÀ NỘI

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

MÔN CƠ SỞ ĐO LƯỜNG VÀ ĐIỀU KHIỂN SỐ

Bài tập C3.2

Thích ứng tín hiệu và chuyển đổi tương tự - số

Thành viên thực hiện: Nhóm 3

22022118 Phạm Văn Duy

22022103 Ngô Đức Hiếu

22022175 Nguyễn Quốc Toản

22022145 Tạ Đình Kiên

Hà Nội, tháng 3 năm 2025

Mục lục

Bài 1. 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?

Yêu cầu bài tập

- Thiết kế và thực thi oscilloscope đơn kênh trên LabVIEW, sử dụng STM32F401RE để thu thập và hiển thị tín hiệu.
- Hiển thị dạng sóng thời gian thực trên máy tính, hỗ trợ điều chỉnh Volt/div, Time/div và Trigger để quan sát tín hiệu chính xác.
- Đánh giá dải tần hoạt động của thiết bị dựa trên tốc độ lấy mẫu ADC, tốc độ truyền dữ liệu và băng thông phần cứng.
- Nâng cao kỹ năng thực hành với LabVIEW, giao tiếp UART, và xử lý dữ liệu tín hiệu số.

1. Nguyên tắc hoạt động

- LabVIEW giao tiếp với STM32 thông qua cổng UART để nhận dữ liệu ADC.
- Dữ liệu thu được được hiển thị trên Waveform Chart dưới dạng đồ thị tín hiệu theo thời gian.
- Người dùng có thể điều chỉnh Volt/div để thay đổi biên độ hiển thị của tín hiệu.
- Time/div được sử dụng để điều chỉnh tỷ lệ thời gian trên trục X, giúp quan sát tín hiệu rõ ràng hơn.
- Tần số lấy mẫu ảnh hưởng đến độ phân giải thời gian của dữ liệu hiển thị.

Chương trình chạy liên tục trong vòng lặp và dừng khi nhấn nút STOP.

2. Phân tích nguyên tắc hoạt động của hệ thống

Hệ thống này sử dụng LabVIEW để giao tiếp với Arduino qua UART (VISA), thu nhận dữ liệu ADC và hiển thị tín hiệu trên Waveform Chart. Dưới đây là trình tự hoạt động của từng khối:

a. Khối giao tiếp UART (VISA)

- VISA Resource Name: Xác định cổng COM kết nối với Arduino. Người dùng cần chọn đúng cổng để đảm bảo kết nối.

- VISA Serial Port:

+ Thiết lập cấu hình giao tiếp UART với tốc độ baud 9600 bps (hoặc theo cấu hình trên Arduino).

+ Chế độ giao tiếp nối tiếp giúp LabVIEW nhận dữ liệu từ Arduino.

- VISA Read:

+ Nhận chuỗi dữ liệu ADC từ Arduino qua cổng COM.

+ Dữ liệu được gửi theo từng chu kỳ từ Arduino và đọc liên tục trong LabVIEW.

b. Chuyển đổi dữ liệu nhận được

- Fract/Exp String To Number Function:

+ Chuyển đổi chuỗi dữ liệu ADC nhận được từ VISA Read thành giá trị số thực (floating-point).

+ Giúp LabVIEW xử lý dữ liệu đúng định dạng và hiển thị chính xác trên đồ thị.

c. Khối hiển thị tín hiệu (Waveform Chart)

- Waveform Chart:

+ Hiển thị dữ liệu ADC theo thời gian thực.

+ Tín hiệu hiển thị là dạng sóng thời gian, giúp người dùng quan sát tín hiệu dễ dàng.

d. Khối điều chỉnh tham số

- Khối đồng bộ tín hiệu (Trigger):

+ Đảm bảo tín hiệu hiển thị ổn định, không bị trôi theo thời gian.

+ Xác định mức điện áp làm điều kiện kích hoạt hiển thị tín hiệu.

- Amplitude (YScale Maximum & Minimum):

+ Điều chỉnh khoảng biên độ hiển thị của tín hiệu.

+ YScale.Maximum đặt giá trị tối đa của trục Y.

+ YScale.Minimum đặt giá trị tối thiểu của trục Y, có thể đảo ngược khi nhân với -1.

- Time/div (XScale.Multiplier):

+ Điều chỉnh tỷ lệ thời gian trên trục X của Waveform Chart.

+ Giá trị này giúp phóng to hoặc thu nhỏ tín hiệu theo thời gian.

- Frequency (XScale.Multiplier):

+ Điều chỉnh tần số lấy mẫu của tín hiệu.

+ Ảnh hưởng đến số lượng mẫu hiển thị trên đồ thị, giúp quan sát tín hiệu rõ ràng hơn.

e. Khối vòng lặp, chờ và dừng chương trình

- Vòng lặp While:

+ Giúp chương trình chạy liên tục, cập nhật dữ liệu ADC theo thời gian thực.

+ Dữ liệu mới được nhận liên tục từ Arduino và hiển thị trên đồ thị.

- Nút STOP:Khi nhấn, chương trình dừng hoạt động, thoát khỏi vòng lặp và kết thúc thu thập dữ liệu.

- **Wait Function (100ms)** giúp:

+ Giảm tải CPU, tránh chạy quá nhanh.

+ Cập nhật dữ liệu ổn định, làm mượt đồ thị.

+ Điều chỉnh tốc độ hiển thị tín hiệ

3. Các bước thực hiện

a. Cấu hình khối VISA Serial

- VISA Resource Name: Chọn đúng cổng COM của Arduino.

- Baud Rate: 9600.

- Data Bits: 8, Parity: None, Stop Bits: 1.

- VISA Read: Đọc dữ liệu ADC gửi từ Arduino.

Kết nối: Cổng đầu ra kết nối vào VISA Serial Port.

b. Khối VISA Serial Port (VISA Configure Serial Port)

- Chức năng: Cấu hình giao tiếp Serial giữa LabVIEW và Arduino.

- Cấu hình Data Bits: 8, Parity: None, Stop Bits: 1.

- Kết nối:

  + Đầu vào nhận VISA Resource Name.

  + Đầu ra kết nối với VISA Read.

c. Khối VISA Read

- Chức năng: Đọc dữ liệu ADC từ Arduino qua cổng Serial.

- Cấu hình: Số byte đọc: 100

- Kết nối:

  + Đầu vào nhận dữ liệu từ VISA Serial Port.

  + Đầu ra kết nối với Fract/Exp String To Number Function để chuyển đổi dữ liệu.

d. Khối Fract/Exp String To Number Function

- Chức năng: Chuyển đổi chuỗi ký tự (string) từ Arduino thành số thực (double).

- Kết nối

  + Đầu vào nhận dữ liệu từ VISA Read.

  + Đầu ra kết nối với Waveform Chart để hiển thị

e. Đồng bộ tín hiệu (Trigger)

- Chức năng: Giúp tín hiệu hiển thị ổn định trên Waveform Chart, tránh tín hiệu bị trôi.

- Cấu hình:

  + Xác định ngưỡng trigger (Trigger Level).

  + So sánh dữ liệu ADC với Trigger Level.

  + Chỉ hiển thị dữ liệu khi vượt ngưỡng trigger.

f. Khối Waveform Chart

- Chức năng: Hiển thị tín hiệu ADC dưới dạng đồ thị thời gian thực.

- Kết nối: Đầu vào nhận dữ liệu từ Fract/Exp String To Number Function.

g. Khối điều chỉnh tham số (Amplitude, Frequency, Time/div)

- **Amplitude (Điều chỉnh biên độ - Volt/div)**

+ Chức năng: Điều chỉnh thang đo biên độ của tín hiệu trên trục Y của Waveform Chart.Xác định giá trị tối đa và tối thiểu của trục Y để hiển thị tín hiệu theo đúng mức điện áp mong muốn.

+ Cấu hình:

· Amplitude (giá trị nhập từ người dùng) được gán cho YScale.Maximum của Waveform Chart.

· Một phép nhân với -1 được thực hiện để thiết lập YScale.Minimum, đảm bảo trục Y đối xứng với trục 0V.

+ Kết nối: Amplitude → Waveform Chart.YScale.Maximum

Amplitude × (-1) → Waveform Chart.YScale.Minimum.

- **Frequency (Điều chỉnh XScale.Multiplier - Tần số hiển thị)**

+ Chức năng: Điều chỉnh tần số hiển thị, tức là số điểm dữ liệu hiển thị trên đồ thị trong một khoảng thời gian.Xác định cách hiển thị tín hiệu theo trục X, giúp người dùng phóng to hoặc thu nhỏ dữ liệu theo thời gian.

+ Cấu hình: Giá trị nhập từ người dùng được nhân với một hệ số để thay đổi XScale.Multiplier của Waveform Chart.

+ Kết nối: Frequency → Waveform Chart.XScale.Multiplier

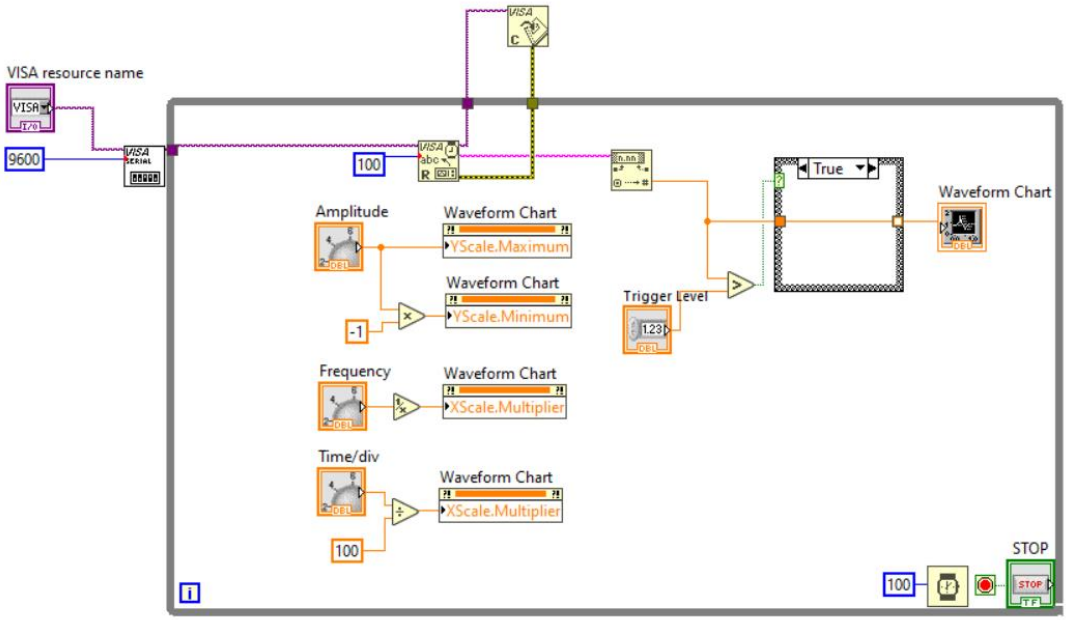
- **Time/div (Điều chỉnh khoảng thời gian hiển thị tín hiệu)**

+ Chức năng: Điều chỉnh khoảng thời gian hiển thị trên trục X, giúp thay đổi độ chi tiết của tín hiệu. Cho phép thu phóng tín hiệu theo thời gian, giống như tính năng Time/div trên oscilloscope thực tế.

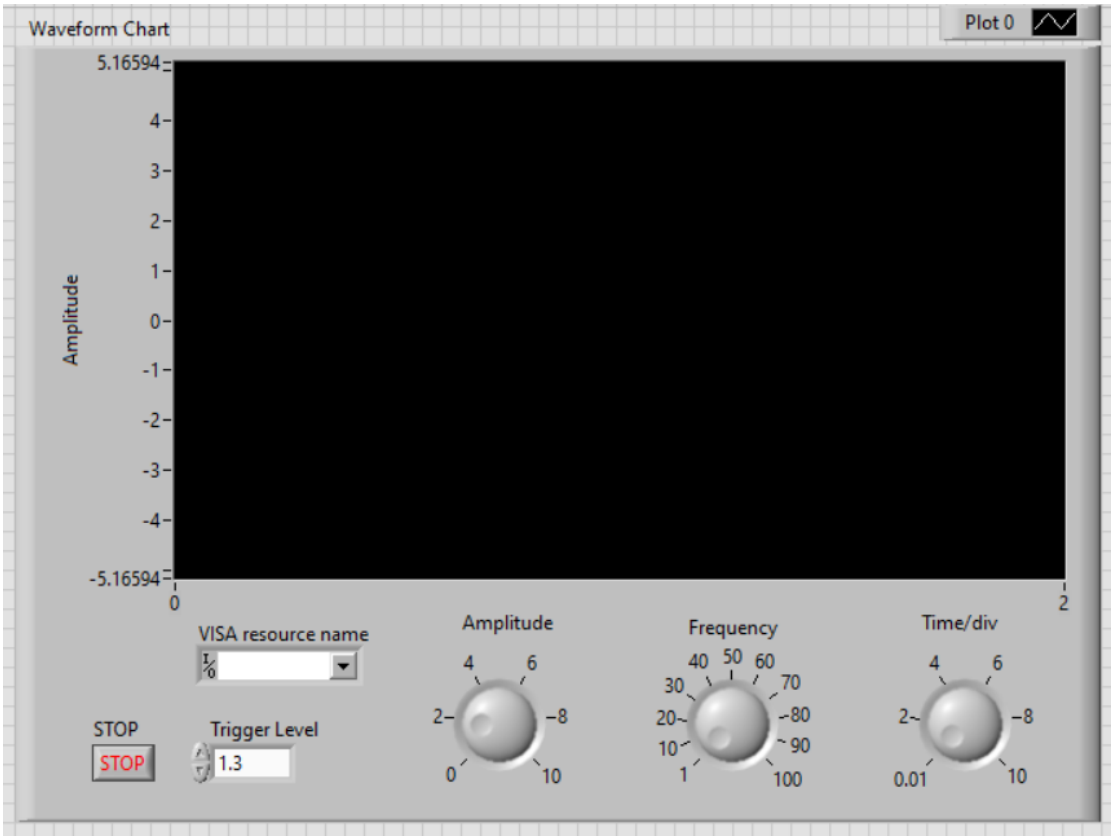
+ Cấu hình: Giá trị Time/div được chia cho 100 để tính toán giá trị thích hợp cho XScale.Multiplier của Waveform Chart.

+ Kết nối: Time/div → Chia cho 100 → Waveform Chart.XScale.Multiplier

➢ Mạch hoàn thiện



Giao diện:



4.Kiểm tra, đánh giá hiệu quả hệ thống và hiệu chỉnh

Đánh giá hiệu suất hệ thống

a. Độ chính xác của dữ liệu nhận từ UART

- Tốc độ baudrate là 9600 baud, tương đối chậm, có thể gây giảm tốc độ cập nhật dữ liệu trên đồ thị.
- Nếu tần số tín hiệu cao, dữ liệu có thể bị mất hoặc trễ khi hiển thị.

b. Độ chính xác của DAC (nếu có sử dụng để phát tín hiệu)

- Nếu tín hiệu đầu vào được tạo từ DAC R-2R, thì sai số điện trở sẽ ảnh hưởng đến độ tuyến tính của sóng.
- Nguồn cấp không ổn định có thể làm thay đổi biên độ tín hiệu.

c. Tốc độ cập nhật tín hiệu

- Việc lấy mẫu tín hiệu ADC có thể bị giới hạn bởi tốc độ truyền UART và tốc độ cập nhật của Waveform Chart.
- Nếu lấy mẫu không đủ nhanh, tín hiệu có thể bị méo hoặc mất chi tiết.

d. Chất lượng hiển thị trên LabVIEW

- Nếu tốc độ cập nhật dữ liệu không đồng bộ với Waveform Chart, tín hiệu có thể bị giật hoặc không chính xác.
- Cách thiết lập trigger level sẽ ảnh hưởng đến độ ổn định của dạng sóng hiển thị.

Hiệu chỉnh

a. Hiệu chỉnh nhận tín hiệu UART vào LabVIEW

Kiểm tra tốc độ truyền UART.

- Lý thuyết: Mạch hiện đang chạy với baud rate = 9600, nhưng tốc độ này có thể hạn chế số lượng mẫu nhận được.
- Thực tế: Với tốc độ này, dữ liệu có thể bị trễ khi truyền đến LabVIEW.
- Hiệu chỉnh: Tăng baud rate lên 115200 để cải thiện tốc độ cập nhật.

Kiểm tra độ trễ trong LabVIEW

- Vấn đề: Bộ đệm nhận dữ liệu có thể bị tràn nếu UART gửi quá nhanh.
- Hiệu chỉnh:
  + Giới hạn số lượng mẫu đọc mỗi lần trong LabVIEW để tránh quá tải.
  + Thêm Wait (ms) trong vòng lặp để giảm tải CPU.

b. Hiệu chỉnh hiển thị trên LabVIEW

Kiểm tra biên độ hiển thị

- Lý thuyết: Giá trị biên độ hiển thị trên Waveform Chart cần khớp với tín hiệu thực.
- Thực tế: Biên độ có thể chưa khớp với điện áp thực tế đo được từ DAC.

- Hiệu chỉnh: Điều chỉnh thang đo YScale.Maximum và YScale.Minimum dựa trên biên độ thực tế của tín hiệu.

Kiểm tra tần số hiển thị

- Lý thuyết: XScale.Multiplier phải đúng với tốc độ lấy mẫu.
- Thực tế: Nếu giá trị Time/div không đúng, dạng sóng sẽ bị co giãn không chính xác.
- Hiệu chỉnh:
  + Xác định thời gian lấy mẫu thực tế từ UART và cập nhật XScale.Multiplier.
  + Nếu Time/div đang dùng 100, phải kiểm tra lại giá trị thực tế để hiệu chỉnh.

c. Hiệu chỉnh Trigger Level

- Vấn đề: Nếu Trigger Level không đồng bộ với dữ liệu ADC, tín hiệu có thể hiển thị sai hoặc mất ổn định.
- Hiệu chỉnh:
  + Đảm bảo giá trị Trigger Level nằm trong khoảng tín hiệu ADC nhận được.
  + Nếu tín hiệu có biên độ từ 0-4.8V, chọn mức Trigger phù hợp, ví dụ 2.4V.

5. Dải tần hoạt động của thiết bị

Thiết bị có thể hoạt động ổn định trong dải tần từ 0 đến ~600 Hz.

Dải tần hoạt động của thiết bị phụ thuộc vào:

Tốc độ UART:

- baud rate = 115200, tốc độ truyền thực tế khoảng 3600 giá trị ADC/giây.
- Với 5 mẫu mỗi chu kỳ sóng, tần số tối đa hiển thị chính xác chỉ khoảng **680 Hz**.

Tốc độ ADC: Arduino có ADC chậm (~10 kSPS), khó theo kịp tín hiệu tần số cao.

Giới hạn phần cứng: Nếu tần số quá cao, mạch không đủ khả năng thu thập và hiển thị chính xác.

Bài 2: Thiết kế và thực thi máy phát chức năng (function generator) đơn giản có thể phát xung sin, vuông, tam giác sử dụng VĐK và DAC0808. Máy phát có thể chọn loại xung và tần số cần phát;Cho biết dải tần hoạt động của thiết bị? Nó phụ thuộc vào yếu tố nào?

I. Yêu cầu bài toán

Thiết kế một mạch phát tín hiệu sử dụng DAC

Mạch có thể phát các loại sóng sin, vuông, tam giác

Lập trình vi điều khiển để thực hiện truyền tín hiệu và giao tiếp với LabView qua UART để quan sát sóng tín hiệu đầu ra.

1. **Phân tích bài toán**

a. **Nguyên tắc hoạt động của mạch**

Vi điều khiển phát tín hiệu số 8 bit – 256 mức từ các chân digital.

Mạch DAC được mắc theo kiểu R2R (15k và 30k), nhận tín hiệu số 8 bit đầu vào, cho đầu ra là điện áp tương tự tương ứng với từng mức.

Vi điều khiển đo điện áp đầu ra tương tự qua chân analog, sau đó gửi dữ liệu qua UART đến máy hiện sóng trên LabView.

b. **Phân tích đối tượng đo và hệ thống điều khiển**

Hệ thống máy phát sóng gồm:

- **Phần phát sóng (Vi điều khiển Arduino + DAC R-2R)**: Tạo tín hiệu đầu ra ở dạng số và chuyển đổi sang tín hiệu tương tự (analog) thông qua mạch DAC.
- **Phần điều khiển (Nút nhấn + Vi điều khiển Arduino)**: Cho phép người dùng chọn loại sóng phát (sin, vuông, tam giác) bằng các nút bấm, vi điều khiển sẽ xử lý và xuất tín hiệu tương ứng.
- **Phần đo lường và hiển thị (Vi điều khiển Arduino + LabVIEW)**: Đọc tín hiệu sóng từ DAC bằng chân A0 của Arduino, sau đó truyền dữ liệu qua UART đến máy tính. Phần mềm LabVIEW trên máy tính sẽ hiển thị dạng sóng theo thời gian thực, hỗ trợ điều chỉnh thông số như Volt/Div, Time/Div.

c. **Yêu cầu phần cứng**

Để thực hiện được yêu cầu bài toán, ta cần:

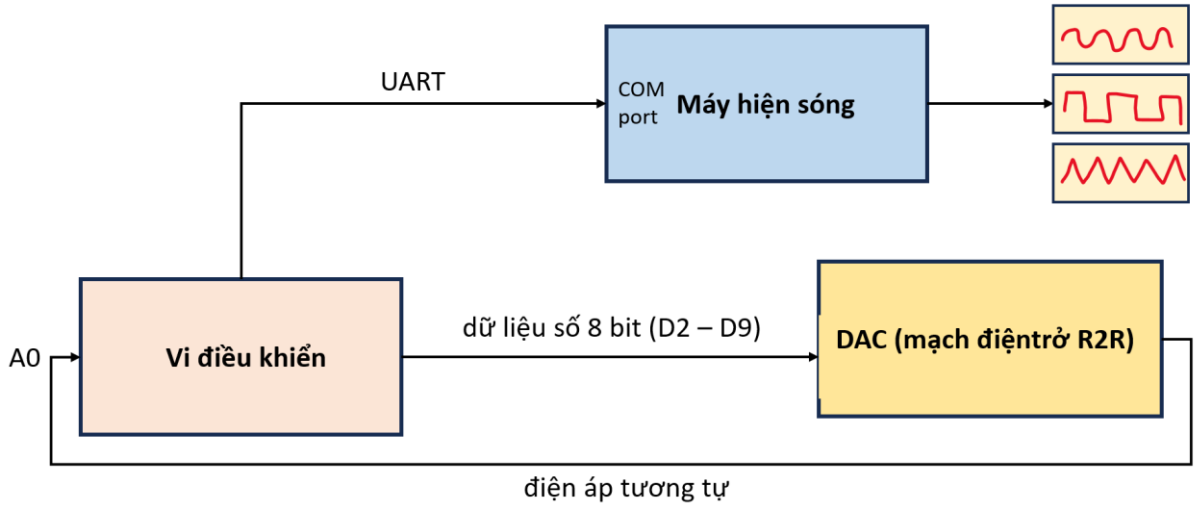
- Arduino để thực hiện phát tín hiệu số và đo đầu ra tương tự để gửi dữ liệu qua UART
- Mạch R2R 8 bit (7 điện trở loại 15k, 9 điện trở loại 30k), nhận đầu vào số từ chân D2 – D9 của vi điều khiển
- Mạch nút bấm để thực hiện việc chọn loại sóng phát ra dựa trên nút được bấm
- Máy hiện sóng được thiết kế trên phần mềm LabView để quan sát tín hiệu đầu ra

d. **Yêu cầu về hệ thống**

- Dữ liệu gửi đến DAC phải đồng nhất (mức 0 tương ứng ~0V, mức 1 tương ứng ~5V)
- Các điện trở R2R phải mắc theo đúng tỉ lệ ½.
- Dữ liệu gửi đến LabView là điện áp đo được (không gửi dữ liệu đo được trực tiếp từ chân A0)
- Máy hiện sóng trên LabView phải đọc được dữ liệu số thực.
- Sóng hiện lên phải đúng dạng tín hiệu mong muốn.

II. Thực thi hệ thống

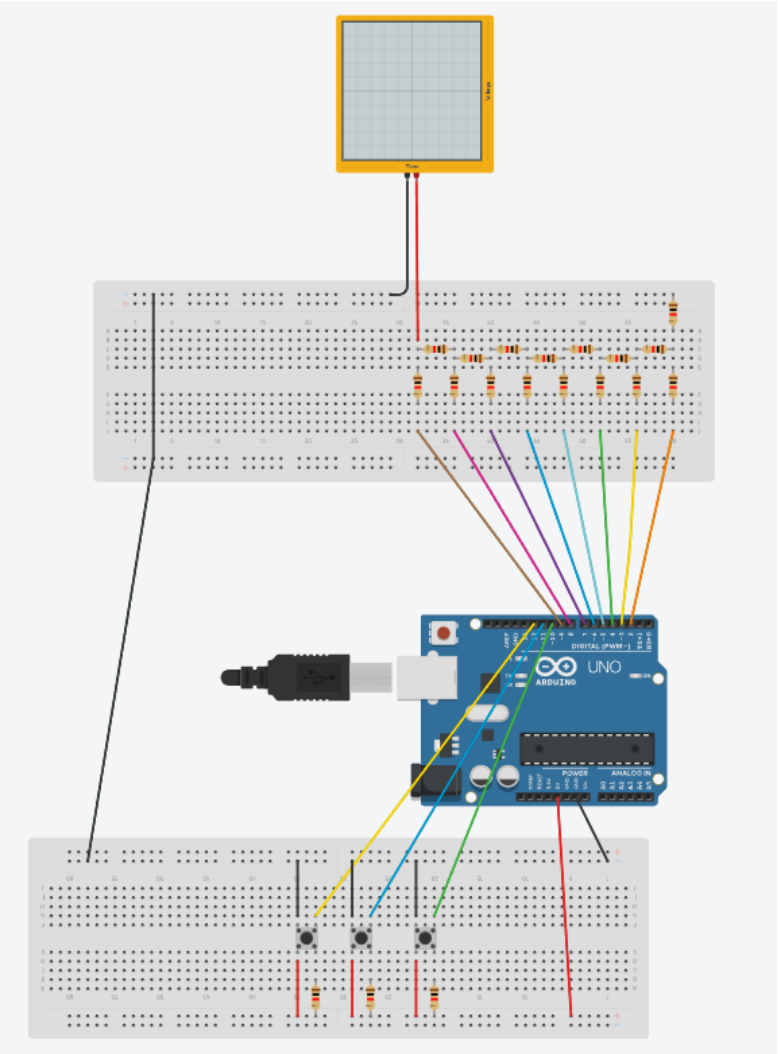
1. Sơ đồ khối hệ thống



2. Kết nối các module phần cứng

- **Mạch phát sóng (Arduino + DAC R-2R + Nút nhấn)**
- **Arduino** xuất dữ liệu dạng sóng qua **PORTD (D2-D9)** đến **mạch DAC R-2R**.
- **Nút chọn sóng** nối vào **D10 (Sin), D11 (Vuông), D12 (Tam giác)**.
- **Mạch đo lường (Arduino + UART)**
- **Tín hiệu từ DAC** vào **chân A0 (ADC) của Arduino**.
- **Arduino gửi dữ liệu** qua **UART** đến máy tính.
- **Kết nối đến máy hiện sóng**
- **Arduino gửi dữ liệu qua ST-Link Virtual COM Port tương ứng**, máy hiện sóng trên LabVIEW đọc, xử lí dữ liệu và hiển thị dạng sóng.

Cách lắp mạch được thể hiện như hình bên dưới.



3. **Viết chương trình cho vi điều khiển**

Chương trình cần có các chức năng chính:

- **Cấu hình các chân dữ liệu số cho DAC**
  - Chọn các chân Digital D2 – D9 và cấp đầu và số 8bit (256 mức) cho mạch DAC.
  - Dữ liệu đưa vào DAC dựa trên giá trị của bảng LUT tương ứng của loại sóng.
  - Điều chỉnh giá trị số đưa vào phù hợp và không vượt ngưỡng 255.
- **Cấu hình các chân để chọn loại sóng**
  - Cấu hình nút nhấn (D10, D11, D12) với INPUT_PULLUP để chọn dạng sóng
- **Cấu hình chân ADC để đọc điện áp tương tự tại đầu ra DAC**
  - Cấu hình chân A0 để đọc điện áp đầu ra
  - Chuyển đổi giá trị đo được thành điện áp tương ứng
- **Vòng lặp chính**

- Gửi mẫu sóng hiện tại đến DAC.
- Tạo độ trễ để điều chỉnh tần số.
- Đọc giá trị ADC, lọc tín hiệu và gửi dữ liệu đến UART.

Mã nguồn:

```
#define F_CPU 16000000UL // Xung nhịp 16MHz
#include <avr/io.h>
#include <util/delay.h>

#define NUM_SAMPLES 32
#define BAUD_RATE 9600

// Định nghĩa chân nút nhấn
#define BTN_SIN      PD2  // Nút chọn sóng sin
#define BTN_SQUARE   PD3  // Nút chọn sóng vuông
#define BTN_TRIANGLE PD4  // Nút chọn sóng tam giác

// LUT cho các dạng sóng
const uint8_t sineWave[NUM_SAMPLES] = {
  128, 152, 176, 200, 224, 247, 255, 247, 224, 200, 176, 152, 128, 104, 80, 56,
  32,  8,   0,   8,   32,  56,  80,  104, 128, 152, 176, 200, 224, 247, 255, 247
};

const uint8_t squareWave[NUM_SAMPLES] = {
  255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255, 255,
  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0
};

const uint8_t triangleWave[NUM_SAMPLES] = {
  0,  16,  32,  48,  64,  80,  96, 112, 128, 144, 160, 176, 192, 208, 224, 240,
  240, 224, 208, 192, 176, 160, 144, 128, 112, 96,  80,  64,  48,  32,  16,  0
};

uint8_t waveType = 0; // 0: Sin, 1: Vuông, 2: Tam giác

void UART_init() {
    uint16_t UBRR_VALUE = F_CPU / 16 / BAUD_RATE - 1;
    UBRR0H = (UBRR_VALUE >> 8);
    UBRR0L = UBRR_VALUE;
    UCSR0B = (1 << TXEN0); // Bật TX (Truyền)
    UCSR0C = (1 << UCSZ01) | (1 << UCSZ00); // 8-bit data, 1 stop bit
}

void UART_sendChar(char data) {
    while (!(UCSR0A & (1 << UDRE0))); // Chờ buffer trống
    UDR0 = data;
}


void UART_sendFloat(float value) {
    char buffer[10];
    dtostrf(value, 6, 2, buffer);
    for (char *ptr = buffer; *ptr; ptr++) {
        UART_sendChar(*ptr);
    }
    UART_sendChar('\n');
}

void ADC_init() {
    ADMUX = (1 << REFS0); // Chọn Vref = AVCC
    ADCSRA = (1 << ADEN) | (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0); // Chia tần số x128
}

uint16_t ADC_read() {
    ADCSRA |= (1 << ADSC); // Bắt đầu chuyển đổi
    while (ADCSRA & (1 << ADSC)); // Chờ ADC hoàn tất
    return ADC;
}

float readFilteredADC() {
    uint16_t sum = 0;
    for (int i = 0; i < 10; i++) {
        sum += ADC_read();
    }
    return (sum / 10.0) * (5.0 / 1023.0);
}

```

```
72  void debounceButtons() {
73      static uint8_t prevBtnState = 0xFF; // Lưu trạng thái trước của nút
74      uint8_t currentBtnState = PIND & ((1 << BTN_SIN) | (1 << BTN_SQUARE) | (1 << BTN_TRIANGLE));
75
76      if ((prevBtnState & (1 << BTN_SIN)) && !(currentBtnState & (1 << BTN_SIN))) {
77          waveType = 0;
78          _delay_ms(50);
79      }
80      if ((prevBtnState & (1 << BTN_SQUARE)) && !(currentBtnState & (1 << BTN_SQUARE))) {
81          waveType = 1;
82          _delay_ms(50);
83      }
84      if ((prevBtnState & (1 << BTN_TRIANGLE)) && !(currentBtnState & (1 << BTN_TRIANGLE))) {
85          waveType = 2;
86          _delay_ms(50);
87      }
88      prevBtnState = currentBtnState;
89  }
90
91  void sendToDAC(uint8_t value) {
92      PORTD = (PORTD & 0x03) | (value << 2); // 6 bit đầu vào PORTD
93      PORTB = (PORTB & 0xFC) | ((value >> 6) & 0x03); // Bit 6, 7 vào PORTB
94  }
95
96  void setup() {
97      UART_init();
98      ADC_init();
99  DDRD = 0b11111100; // D2-D7 OUTPUT (8-bit DAC)
100     DDRB |= (1 << PB0) | (1 << PB1); // PB0, PB1 OUTPUT (Bit 6, 7 của DAC)
101
102     DDRD &= ~((1 << BTN_SIN) | (1 << BTN_SQUARE) | (1 << BTN_TRIANGLE)); // Đặt chân nút là INPUT
103     PORTD |= (1 << BTN_SIN) | (1 << BTN_SQUARE) | (1 << BTN_TRIANGLE); // Kích hoạt pull-up
104 }
```

```
106 void loop() {
107     debounceButtons();
108     for (int i = 0; i < NUM_SAMPLES; i++) {
109         if (waveType == 0)
110             sendToDAC(sineWave[i]);
111         else if (waveType == 1)
112             sendToDAC(squareWave[i]);
113         else
114             sendToDAC(triangleWave[i]);
115
116         _delay_us(50);  // Điều chỉnh tần số phát
117
118         float voltage = readFilteredADC();
119         UART_sendFloat(voltage);
120     }
121 }
```

4. Kiểm tra, đánh giá hiệu quả hệ thống và hiệu chỉnh

Độ chính xác của hệ thống có thể bị ảnh hưởng bởi các yếu tố:

a) Độ chính xác của DAC R-2R

- Sai số của điện trở trong mạch R-2R DAC ảnh hưởng đến độ tuyến tính và chất lượng dạng sóng.
- Điện trở có sai số càng nhỏ thì chất lượng sóng càng tốt.

b) Tốc độ cập nhật tín hiệu DAC

- Chu kỳ lấy mẫu càng ngắn, tần số sóng phát ra càng cao.
- Giới hạn của vi điều khiển Arduino có thể làm giảm độ mượt của sóng, đặc biệt ở tần số cao.

c) Nhiễu tín hiệu ADC khi đo sóng

Khi DAC xử lí tín hiệu, dễ bị ảnh hưởng bởi nhiễu (ví dụ như chạm tay vào điện trở => làm thay đổi tỉ lệ 1: 2 của mạch).

d) Tốc độ truyền UART đến LabVIEW

UART chạy 115200 baud, nhưng nếu dữ liệu ADC gửi quá nhanh, LabVIEW có thể bị tràn bộ đệm, gây hiển thị sai hoặc lag.

e) Sai số do nguồn cấp

Nếu nguồn cấp không ổn định, điện áp đầu ra của DAC có thể dao động, ảnh hưởng đến chất lượng dạng sóng.

f) Độ chính xác hiển thị trên LabVIEW

Tốc độ lấy mẫu không phù hợp với tần số sóng, dữ liệu có thể bị hiển thị sai.

Quan sát số liệu đầu ra, dạng sóng trên oscilloscope và tiến hành hiệu chỉnh:

a. Mục tiêu hiệu chỉnh

Đầu ra sóng (DAC): Đảm bảo biên độ và tần số của sóng sin, vuông, tam giác đúng với thiết kế lý thuyết.

Đầu vào tín hiệu (ADC): Đo điện áp từ A0 chính xác, không bị nhiễu.

Tương tác với LabVIEW: Truyền dữ liệu qua Serial ổn định, hiển thị đúng trên giao diện.

b. Các bước hiệu chỉnh

Hiệu chỉnh DAC (Đầu ra sóng)

Kiểm tra biên độ:

Lý thuyết: Giá trị 0-255 từ bảng sóng tương ứng 0-5V qua R-2R DAC (Vref = 5V)

Thực tế:

- đo Vout tại đầu ra DAC (sau R-2R).
- Với `squareWave`: Giá trị 255 → ~5V, giá trị 0 → ~0V.

Hiệu chỉnh:

- Biên độ chỉ đạt: 4.8V
- Kiểm tra điện áp nguồn 5V từ Arduino (có thể thấp hơn do USB).

Điều chỉnh bảng sóng:

```
uint8_t scaleValue(uint8_t value) {
  return (uint8_t)((float)value * 5.0 / measured_max_voltage);
}
sendToDAC(scaleValue(sineWave[i]));
```

- Thay `measured_max_voltage` = 4.8V thay vì 5V

Kiểm tra tần số:

Lý thuyết: Tần số sóng = 1 / (NUM_SAMPLES × delayMicroseconds) = 1 / (32 × 50µs) = 625Hz.

Thực tế: Đo bằng oscilloscope: Tần số chỉ đạt 603Hz

Hiệu chỉnh:

Điều chỉnh `delayMicroseconds(50)`-> delayMicroseconds(48) để tăng tần số

Kiểm tra dạng sóng:

Đo sóng sin, vuông, tam giác:

- Sóng sin: Đảm bảo mượt (32 mẫu đủ cho tần số thấp).
- Sóng vuông: Cạnh dốc, không bị tròn.
- Sóng tam giác: Tuyến tính, không bị méo.

Hiệu chỉnh:

- kiểm tra mạch R-2R (điện trở không đều)
- tăng số mẫu (NUM_SAMPLES = 64).

Hiệu chỉnh ADC (Đầu vào A0)

Kiểm tra độ chính xác:

Lý thuyết: `readFilteredADC()` trả về điện áp 0-5V (từ giá trị ADC 0-1023).

Thực tế:

- Nối biến trở vào A0, đo điện áp thực tế bằng multimeter tại A0.
- So sánh với giá trị Serial in ra.

Hiệu chỉnh:

multimeter = 2.5V, Serial = 2.4V:

- Điều chỉnh hệ số trong `readFilteredADC()`:

```
float readFilteredADC() {
  uint16_t sum = 0;
  for (int i = 0; i < 10; i++) {
   sum += analogRead(ANALOG_PIN);
  }
  return (sum / 10.0) * (5.0 / 1023.0) * correction_factor;}
```

Hiệu chỉnh giao tiếp Serial

Kiểm tra tốc độ truyền:

- Baud rate 9600 gửi ~960 ký tự/giây. Với mỗi mẫu (ví dụ "2.50\n" ≈ 5 ký tự), tối đa ~192 mẫu/giây.
- Thực tế bị giới hạn bởi `delayMicroseconds(50)` (625Hz / 32 = ~19.5 mẫu/giây).

Hiệu chỉnh:

Cần nhanh hơn, tăng baud rate lên thành 9600*192/19.5 = 94523.

Ta có thể lặp đi lặp lại bước 4 để đạt được các thông số hệ thống mong muốn.

Video thực nghiệm

C3_2_FunctionGenerator_Oscilloscope.mp4

Dải tần hoạt động của mạch phát sóng

Phụ thuộc vào tốc độ cập nhật dữ liệu của DAC, cụ thể là **tần số gửi mẫu** trong vòng lặp:

$$f = {}^{1}/_{N \ * \ Tsample}$$

Với:

- N = 32 là số mẫu trên chu kỳ sóng
- Tsample: thời gian giữa hai mẫu (phụ thuộc vào delayMicroseconds(50) trong code)

⇨ **Tần số tối đa lý thuyết**: f = 1 / (32 * 50µs) = 625Hz

- Nếu giảm delayMicroseconds, tần số có thể tăng, nhưng bị giới hạn bởi tốc độ xử lý của vi điều khiển.

Các yếu tố ảnh hưởng đến dải tần hoạt động của thiết bị

1. **Vi điều khiển & tốc độ xử lý**:
   - Arduino có xung nhịp **16 MHz**, tốc độ cập nhật PORT bị giới hạn.
2. **Chu kỳ lấy mẫu**
3. **Độ phân giải DAC** (8-bit từ R-2R)
   - Độ phân giải thấp có thể gây méo dạng sóng ở tần số cao.
4. **Băng thông của mạch DAC & mạch đo**
   - Mạch R-2R có đáp ứng tần số giới hạn do điện dung ký sinh.
5. **Độ trễ UART khi gửi dữ liệu**
   - Nếu UART truyền quá nhanh, có thể làm chậm vòng lặp, ảnh hưởng đến tần số phát.

Đánh giá hiệu quả hệ thống

- Hệ thống phát sóng hoạt động ổn định, có thể tạo 3 sóng và điều chỉnh tần số.
- ADC 8-bit giúp đo tín hiệu chính xác.
- LabVIEW hiển thị dạng sóng theo thời gian thực.

- Tốc độ lấy mẫu ADC có thể chưa đồng bộ hoàn toàn với tần số sóng.

Bảng đánh giá thành viên

Sinh viên	Công việc	Đóng góp
22022118 Phạm Văn Duy	Tham gia nghiên cứu và lập trình, thực thi máy hiện sóng và phát sóng; tham gia đánh giá kết quả và hiệu chỉnh máy hiện sóng và phát sóng, làm báo cáo word cho phần bài tập 2 - máy phát sóng	**25%**
22022103 Ngô Đức Hiếu	Tham gia nghiên cứu và lập trình, thực thi máy hiện sóng và phát sóng; tham gia đánh giá kết quả và hiệu chỉnh máy hiện sóng và phát sóng, làm báo cáo word cho phần bài tập 2 - máy phát sóng	**25%**
22022175 Nguyễn Quốc Toản	Tham gia nghiên cứu và lập trình, thực thi máy hiện sóng và phát sóng; tham gia đánh giá kết quả và hiệu chỉnh máy hiện sóng và phát sóng, làm báo cáo word cho phần bài tập 1 - máy hiện sóng	**25%**
22022145 Tạ Đình Kiên	Tham gia nghiên cứu và lập trình, thực thi máy hiện sóng và phát sóng; tham gia đánh giá kết quả và hiệu chỉnh máy hiện sóng và phát sóng, làm báo cáo word cho phần bài tập 1 - máy hiện sóng	**25%**