

ĐẠI HỌC QUỐC GIA HÀ NỘI  
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

□ □ □



BÁO CÁO BÀI TẬP  
CƠ SỞ ĐO LƯỜNG VÀ ĐIỀU KHIỂN SỐ

THIẾT KẾ MÁY PHÁT CHỨC NĂNG  
(FUNCTION GENERATOR) SỬ DỤNG  
VĐK VÀ DAC

<i>Họ và tên sinh viên</i>	:	<i>MSV</i>
Nguyễn Trọng Nam		22022161
Doãn Đức Minh		22022135
Lê Thế Minh		22022215
Cao Song Toàn		22022158

*Hà Nội, tháng 3 năm 2025*

## **Nội dung báo cáo**

<b>I. Tổng quan</b>	<b>3</b>
1.Nội dung thực hành.....	3
2.Linh kiện chuẩn bị.....	3
3. Thiết kế bộ DAC.....	4
<b>II. Nguyên lý hoạt động:</b>	<b>5</b>
1.Ý nghĩa.....	5
2.Sơ lược về phần cứng và cách nối.....	5
3.Cách hoạt động thực tế.....	5
<b>III. Thực hành</b>	<b>6</b>
1. Lắp mạch thực tế.....	6
2. Code và nạp code cho vi điều khiển.....	6
<b>IV. Kết quả thực nghiệm</b>	<b>8</b>
<b>V. Kết luận</b>	<b>9</b>

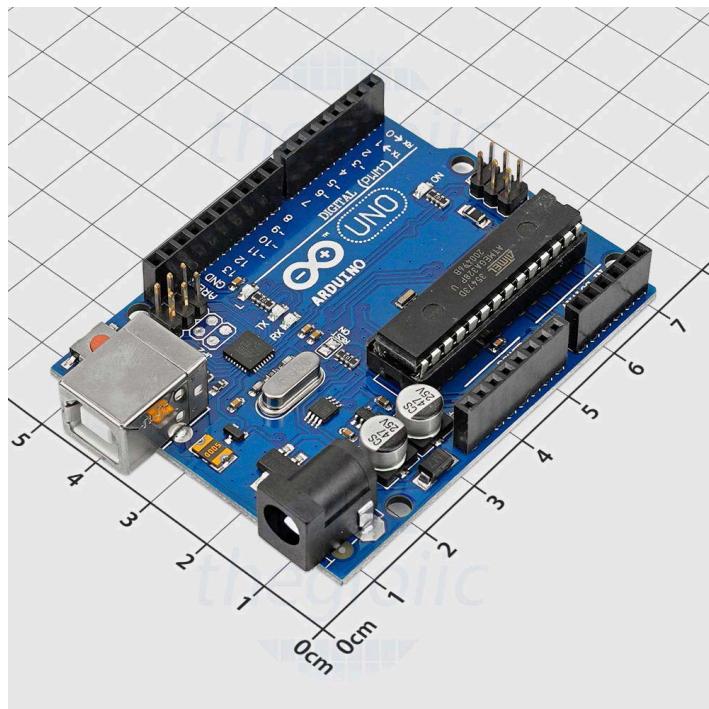
# I. Tổng quan

## 1. Nội dung thực hành

- Sử dụng labVIEW tạo ra máy phát sóng đó qua vi điều khiển, có thể chỉnh được tần số và dạng sóng. Sóng được đưa ra từ vi điều khiển cần chuyển đổi sang analog bằng bộ DAC.
- Lập trình ở mức thanh

## 2. Linh kiện chuẩn bị

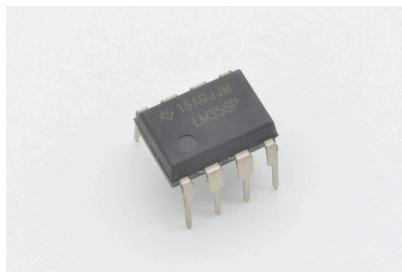
- Sử dụng vi điều khiển Adruino Uno



- Sử dụng mạch USB to TTL CP2102.



- Op-am LM358P.



- Một số dây nối, ST link.

### 3. Thiết kế bộ DAC

Mạch DAC (Digital-to-Analog Converter - Bộ chuyển đổi số sang tương tự) R-2R là một giải pháp đơn giản và tiết kiệm chi phí để chuyển đổi tín hiệu số thành tín hiệu tương tự tuy nhiên lại không có độ chính xác cao.

Nguyên lý hoạt động của mạch R-2R dựa trên việc chia áp theo trọng số của các bit đầu vào. Mỗi bit đóng góp một phần điện áp vào điện áp đầu ra tổng, và mức đóng góp này phụ thuộc vào vị trí của bit trong chuỗi nhị phân.

Công thức tổng quát để tính điện áp đầu ra của mạch R-2R DAC:

$$V_{output} = \frac{V_{b0}}{2^n} + \frac{V_{b1}}{2^{n-1}} + \frac{V_{b2}}{2^{n-2}} + \dots + \frac{V_{bn}}{2^1}$$

Trong trường hợp của mạch 8-bit ( $n = 8$ ), công thức cụ thể là:

$$V_{output} = \frac{V_{b0}}{2^8} + \frac{V_{b1}}{2^7} + \frac{V_{b2}}{2^6} + \frac{V_{b3}}{2^5} + \frac{V_{b4}}{2^4} + \frac{V_{b5}}{2^3} + \frac{V_{b6}}{2^2} + \frac{V_{b7}}{2^1}$$

$V_{b0}, V_{b1}, \dots, V_{b7}$ : Điện áp tại các bit đầu vào, thường là 0V (logic 0) hoặc 5V (logic 1) khi sử dụng nguồn 5V.

Mỗi bit trong mạch R-2R có trọng số giảm dần theo lũy thừa của 2:

- Bit b7 (MSB) có trọng số lớn nhất ( $1/2^7/2^{15}/2$ ), đóng góp nhiều nhất vào điện áp đầu ra.
- Bit b0 (LSB) có trọng số nhỏ nhất ( $1/2^{15}/2^{15}/2^7$ ), đóng góp ít nhất.

Điều này có nghĩa là khi bit b7 được đặt thành 1 (5V), nó sẽ đóng góp  $5 \times 1/2 = 2.5$  vào Vout, trong khi bit b0 chỉ đóng góp  $5 \times 1/2^7 = 0.0195V$ .

Lựa chọn điện trở cho mạch:

## Absolute Maximum Ratings\*

Operating Temperature ..... -55°C to +125°C

Storage Temperature ..... -65°C to +150°C

Voltage on any Pin except RESET  
with respect to Ground ..... -0.5V to  $V_{CC}+0.5V$

Voltage on RESET with respect to Ground.....-0.5V to +13.0V

Maximum Operating Voltage ..... 6.0V

DC Current per I/O Pin ..... 40.0mA

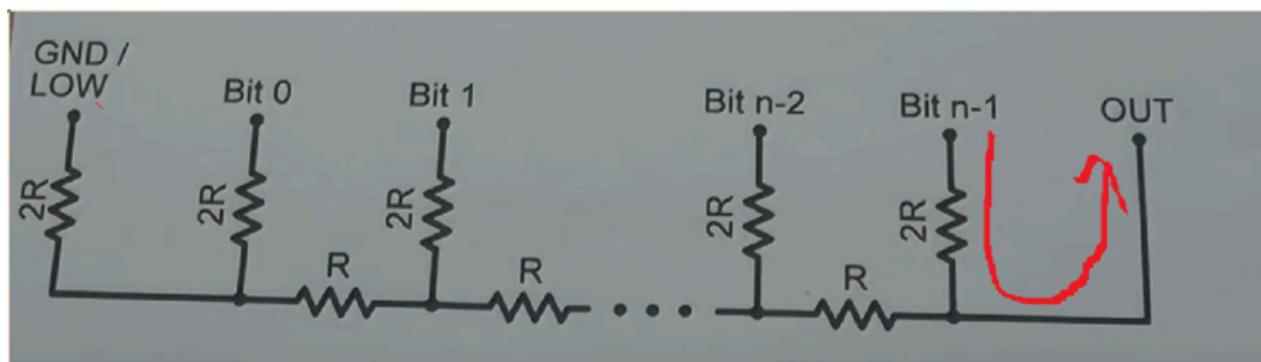
DC Current  $V_{CC}$  and GND Pins ..... 300.0mA

Source:

<https://www.alldatasheet.com/datasheet-pdf/view/80260/ATMEL/ATMEGA8L.html>

- Page 235

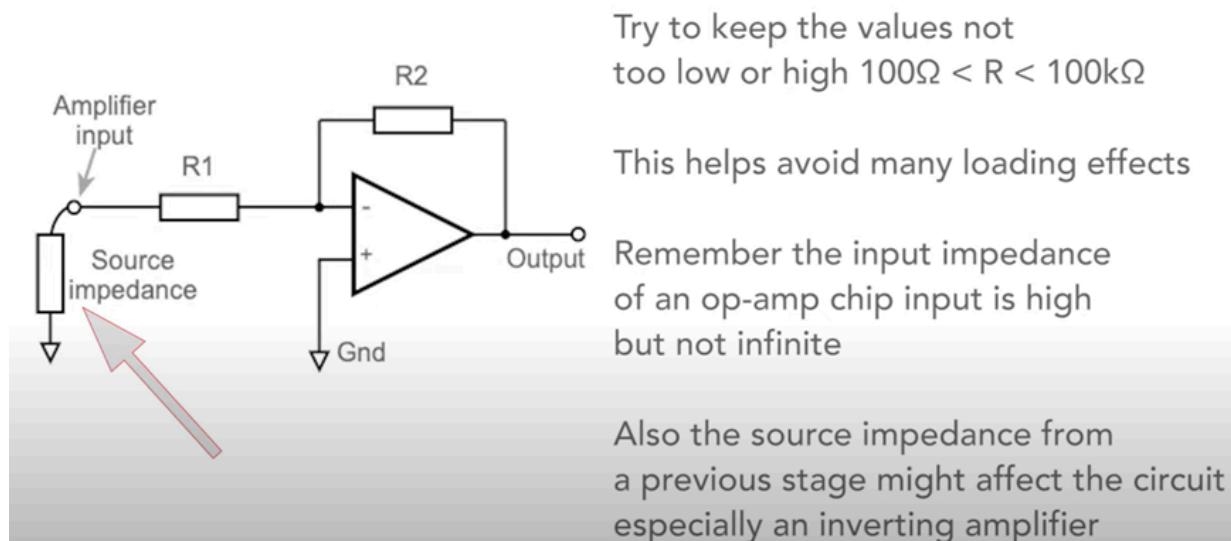
Dòng DC của các chân I/O là 40mA.



Vì vậy để mạch hoạt động tốt các trở phải có giá trị nhỏ nhất là:

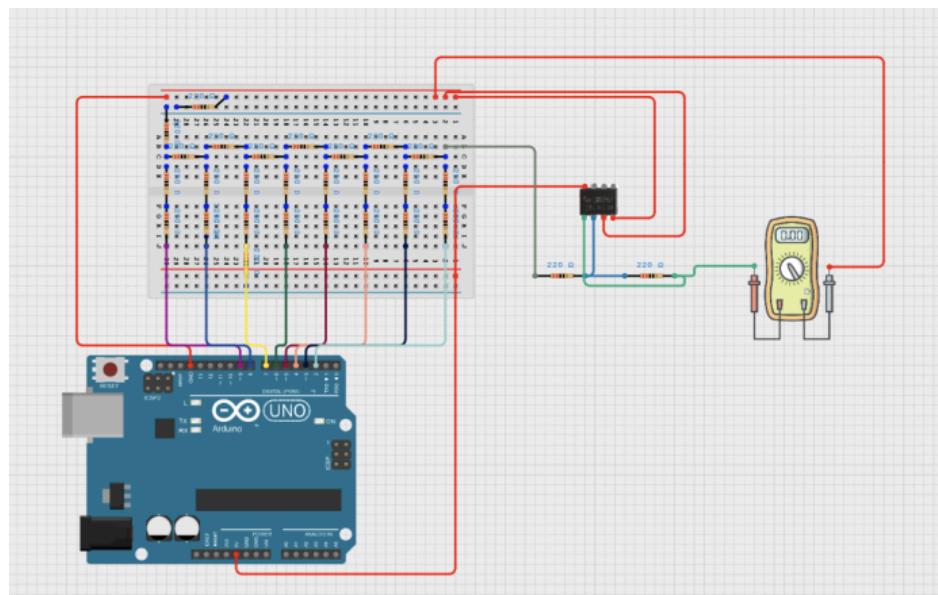
$$2R = 5V/40mA = 125\Omega \Rightarrow R > 75$$

Ta cũng có thể sử dụng thêm inverting Op-amp để giữ cho tín hiệu tương tự đầu ra được ổn định và inverting Op-amp thực hiện được chức năng tính tổng các dòng vào rất phù hợp cho DAC.



Để giữ cho giá trị không quá cao và thấp R sẽ nên bằng trong khoảng  $100 \rightarrow 100k$

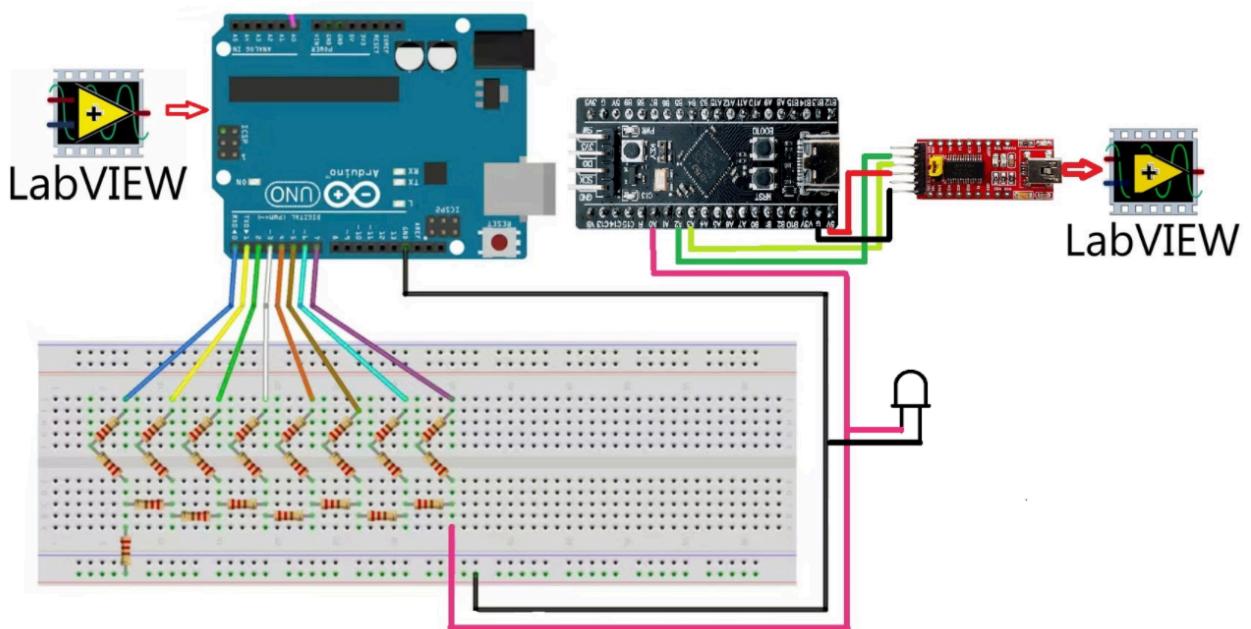
$\Rightarrow$  Trong bài tập này không cần thiết phải sử dụng R quá lớn vậy nên chúng tôi đã sử dụng loại điện trở R = 220.



<https://app.cirkitdesigner.com/project/01108f88-a4bf-4af1-a135-4cbd53b2b246>

## II. Nguyên lý hoạt động:

### 1. Sơ đồ phần cứng:



- 8 chân digital output (D2 -> D9 ) của Arduino ứng với 8 bit đầu vào.
- Đầu ra analog khi đi qua DAC sẽ được đọc vào chân A0 của stm32.
- Stm32 kết nối uart với mạch chuyển đổi qua các chân TX, RX.
- Led cũng được nối với đầu ra có độ sáng phụ thuộc vào đầu ra.

### 2. Nguyên lý:

Ta sử dụng labview để tạo ra các loại sóng ( có thể điều chỉnh loại sóng, tần số), tiến hành lấy mẫu các loại sóng được tạo ra gửi qua Arduino thành 1 chuỗi các giá trị liên tục của sóng chuyển đổi nó về dạng 8-bit ( bằng cách ướm nó với 256 level) => Đưa vào bộ DAC R/2R.

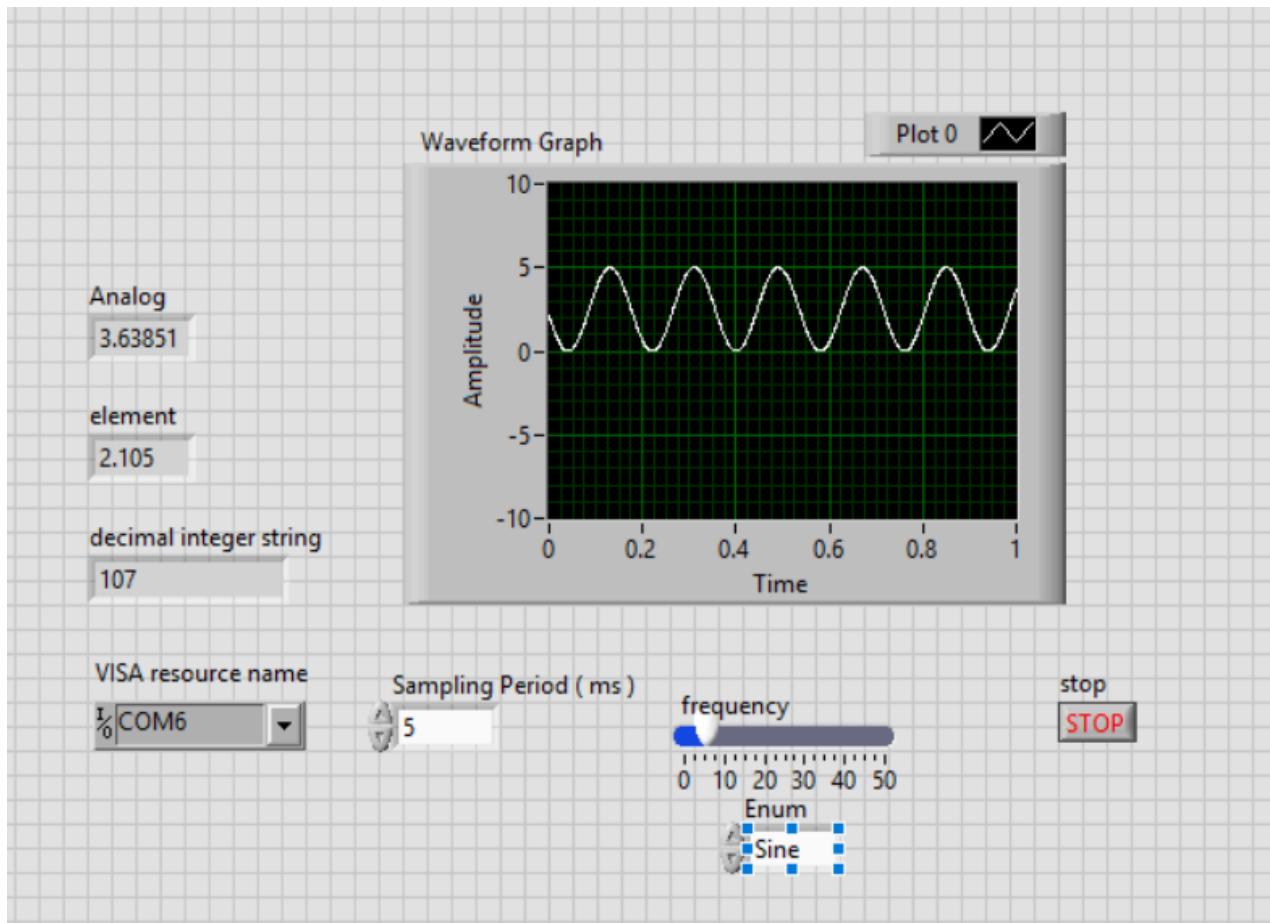
Mạch R-2R dựa trên việc chia áp theo trọng số của các bit đầu vào. Khi các chân GPIO của Arduino Uno được đặt ở mức cao (5V) hoặc thấp (0V), mạng thang R-2R sẽ tạo ra một điện áp đầu ra tương ứng trong khoảng từ 0V đến 5V ( điện áp này là tổng có trọng số của các bit đầu vào).

STM32 chỉ đảm nhận việc đọc giá trị đầu vào ( dạng analog) truyền lại lên Labview và Led để quan sát kết quả.

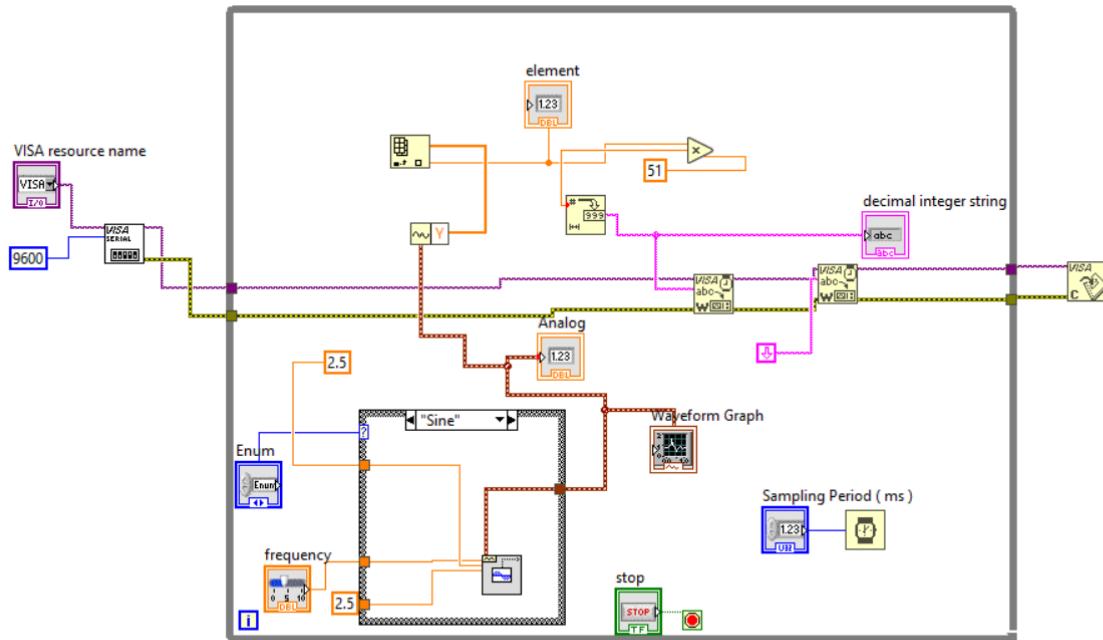
### III.Thực hành

#### 1. Mô phỏng trên labview

Chúng tôi chạy song song 2 chương trình Labview: 1 để phát và 1 để ghi. Máy ghi sóng đã được thực hiện và trình bày ở bài 1, dưới đây là phần máy phát sóng.



<Front panel>



<Block Diagram>

### Giải thích Block Diagram

Ý tưởng chính: Dùng LabVIEW lấy hàm signal generator => Lấy mẫu hàm đó => Chuyển giá trị lấy mẫu với độ phân giải 8bits (0-255) => Chuyển số đó qua string (để Serial có thể chuyển được) => Chuyển các mẫu qua Serial (9600 bits/s) => Chuyển các mẫu đó sang number (0->255) => Dùng giá trị các num đó chuyển thành 8 bits đưa ra 8 chân GPIO => dùng DAC lấy tín hiệu analog.

Đầu tiên, vi xử lý được kết nối giao thức serial qua việc chọn COM trên khôi VISA. Dùng các khôi VISA với chức năng đọc tín hiệu và xử lý tín hiệu với vi xử lý nối xuyên suốt qua 2 đường dây data (màu hồng) và dây error (màu vàng). Bên trong khôi while, có một khôi để tạo ra các tín hiệu sin, square, triangle với biên độ là 2.5, offset 2.5, tần số tự điều chỉnh được. Đưa tín hiệu đó qua waveform graph để có thể thấy rõ. Dùng khôi 'get waveform component' và 'array' theo sau nó để lấy mẫu tín hiệu. Do tín hiệu được lấy mẫu từ 0->5V kiểu double ta cần đưa tín hiệu vào vùng có độ phân giải 8 bits là 0->255 kiểu int, ta nhân với 51 và chuyển kiểu dữ liệu đó qua String đưa vào khôi VISA WRITE để giao tiếp chuyển dữ liệu String đó qua Serial và Adruino nhận rồi xử lý nó, Đằng sau khôi lấy mẫu có một khôi VISA WRITE nhận vào 'line feed constant' hay '/n' để phân tách giữa các mẫu, Adruino có thể biết các mẫu có giá trị như nào thay vì một loạt data dài từ LabVIEW đổ xuống không biết đâu là mẫu lấy được đâu là bit đầu giá trị đâu là bit cuối giá trị.

Cuối cùng, có một khối ‘wait time’ khôi này cũng cho biết rõ chu kỳ lấy mẫu, trong vòng while cứ sau ‘wait time’ một tín hiệu được lấy mẫu đưa vào Serial.

## 2. Code và nạp code cho vi điều khiển

Ở đây sử dụng 2 vi điều khiển Arduino và STM32. Vai trò của Arduino là đọc tín hiệu analog từ Labview chuyển đổi thành 8-bit theo 256 mức và bật/tắt các chân đầu ra tương ứng. Còn vai trò của STM32 chỉ là nhận tín hiệu analog đầu vào từ chân A0 sau đó chuyển và hiển thị lên Labview.

### a) Code Arduino Uno

```
1 #include <avr/io.h>
2
3 char c;
4 String num;
5
6 void setup() {
7     // Cấu hình các chân D2 -> D9 làm OUTPUT
8     DDRD |= 0b11111100; // Chân D2 -> D7 (PORTD) là OUTPUT
9     DDRB |= 0b00000011; // Chân D8, D9 (PORTB) là OUTPUT
10
11    // Khởi tạo UART baud rate 9600
12    UBRR0H = 0;
13    UBRR0L = 103; // Giá trị tương ứng với baud rate 9600 ở 16MHz
14    UCSR0B = (1 << RXEN0) | (1 << TXEN0); // Bật UART RX và TX
15    UCSR0C = (1 << UCSZ01) | (1 << UCSZ00); // Cấu hình UART 8-bit
16 }
17
18 char uartRead() {
19     while (!(UCSR0A & (1 << RXC0))); // Chờ dữ liệu nhận được
20     return UDR0; // Đọc dữ liệu từ UART
21 }
22
23 void loop() {
24     if (UCSR0A & (1 << RXC0)) { // Kiểm tra nếu có dữ liệu UART
25         c = uartRead();
26         num += c;
27
28         if (c == '\n') { // Khi gặp ký tự xuống dòng
29             int value = num.toInt(); // Chuyển chuỗi sang số nguyên
30             num = ""; // Xóa chuỗi để chuẩn bị dữ liệu mới
31
32             // Xuất dữ liệu ra các chân D2 -> D9
33             PORTD = (PORTD & 0x03) | ((value << 2) & 0xFC); // Xuất 6 bit cao vào PORTD (D2->D7)
34             PORTB = (PORTB & 0xFC) | ((value >> 6) & 0x03); // Xuất 2 bit thấp vào PORTB (D8, D9)
35         }
36     }
37 }
```

### b) Code STM32F4xx

< Code để đọc dữ liệu từ A0 và truyền lên Labview sử dụng y nguyên bài 1 >

## IV. Kết quả thực nghiệm

Video thực nghiệm: [https://youtu.be/g\\_ryws\\_oA5E](https://youtu.be/g_ryws_oA5E)

Source code: <https://github.com/iGhost1107/BTVN1.git>

### 1. Dải tần hoạt động của thiết bị là bao nhiêu?

Do arduino có tần số chuyển 9600 bits/s => với mỗi mẫu ta có 4 frames tương ứng với 40 bits ( 1 frames bao gồm String 1 bytes, 2 bit đầu cuối, ở đây chúng ta truyền 0->255 ( dạng string) tương đương với 3 bytes string và ‘/n’ (enter) là byte cuối)

Vậy ta có thể truyền 240 mẫu /s => tần số lấy mẫu cao nhất có thể là 240. Để an toàn ta chọn tần số lấy mẫu là 200 tương ứng với chu kỳ lấy mẫu là 5ms ( như labVIEW trên ). Với tần số lấy mẫu là 200, theo định lý Nyquist thì tần số phát phải cao nhất chấp nhận được là 100, ở đây ta chọn tần số phát cao nhất là 50 để mong muốn giữ đầy đủ thông tin về tín hiệu.

### 2. Dải tần đó phụ thuộc vào yếu tố nào?

Như bên trên, thì ta thấy dải tần đó phụ thuộc vào tần số lấy mẫu trên labVIEW, tốc độ xử lý của labVIEW, Tốc độ truyền dữ liệu qua serial, tốc độ xử lý của vi xử lý khi nhận được giá trị lấy mẫu.

## V. Kết luận

Trong bài thực hành này, chúng tôi đã thiết kế và triển khai thành công một máy phát chức năng (Function Generator) sử dụng vi điều khiển và bộ chuyển đổi số – tương tự (DAC) tuy chưa hoàn chỉnh. Hệ thống có thể tạo ra các dạng sóng khác nhau như sóng sin, sóng vuông, sóng tam giác với tần số có thể điều chỉnh. Kết quả thực nghiệm cho thấy thiết bị hoạt động ổn định trong dải tần số cho phép và có thể truyền dữ liệu chính xác đến LabVIEW để hiển thị và phân tích tín hiệu.

Trong quá trình thực hiện, nhóm đã rất cố gắng thử bổ sung thay thế 3 bộ khuếch đại thuật toán (op-amp) khác nhau vào mạch DAC R-2R nhằm ổn định tín hiệu đầu ra. Tuy nhiên, kết quả thu được không như mong đợi, gần như không có tín hiệu . Do đó, bộ DAC trong bài thực hành chỉ sử dụng mạng điện trở R-2R. Mặc dù hệ thống hoạt động đúng chức năng, nhưng tín hiệu đầu ra ghi lại vẫn còn nhiễu và biên độ bị suy giảm, có thể là do ảnh hưởng của phần cứng như điện trở, dây nối, hoặc đặc tính của vi điều khiển.

Độ chính xác của mạch DAC phụ thuộc khá lớn vào chất lượng linh kiện, cách bố trí mạch, và điều kiện hoạt động thực tế. Ngoài ra, việc sử dụng op-amp cần được tính toán cẩn thận để tránh làm suy hao hoặc méo tín hiệu. Qua bài thực hành này, nhóm đã có thêm kiến thức về cách hoạt động của DAC R-2R, phương pháp giao tiếp giữa vi điều khiển với máy tính qua UART, và cách kiểm tra, đánh giá chất lượng tín hiệu đầu ra. Trong tương lai, có thể cải tiến hệ thống bằng cách sử dụng bộ DAC có độ phân giải cao hơn hoặc tìm cách lọc nhiễu hiệu quả hơn nhằm nâng cao chất lượng tín hiệu.

