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

BÁO CÁO BÀI TẬP LỚN

MÔN: THIẾT KẾ HỆ THỐNG NHÚNG

Đề tài: Bàn Phím Âm Nhạc Đơn Giản

GVHD: Th.S. Nguyễn Phan Hải Phú

Lớp L01 - Nhóm 05

Thành viên:

Lê Đức Quý	2114597
Trương Anh Duy	2110943
Nguyễn Trung Trọng	2110620

TP.HCM, 11/2023

Mục lục

MỞ ĐẦU

Ngày nay, các hệ thống nhúng trở nên phổ biến và đóng vai trò quan trọng trong đời sống con người. Ví dụ quanh ta có rất nhiều sản phẩm nhúng như lò vi sóng, nồi cơm điện, điều hoà, ô tô, máy bay, tàu thuỷ, v.v… Ta có thể thấy hiện nay hệ thống nhúng có mặt ở mọi nơi và mọi lúc trong cuộc sống của chúng ta

Qua môn học thiết kế hệ thống nhúng, chúng em đã hiểu thêm về các hệ thống nhúng trong thực tế, về đặc điểm, tính ưu việt cũng như tính ứng dụng của chúng đối với con người. Với mong muốn làm rõ các kiến thức đã học và áp dụng các kiến thức đã học vào việc rèn luyện và giải trí, nhóm chúng em đã thiết kế sản phẩm Bàn phím âm nhạc đơn giản - một sản phẩm rất quen thuộc trong cuộc sống.

Do thời gian thực hiện và kiến thức còn hạn chế nên còn nhiều sai sót trong quá trình thực hiện đề tài, rất mong được sự bổ sung đóng góp của thầy.

Chúng em xin chân thành cảm ơn thầy Nguyễn Phan Hải Phú đã tận tình hướng dẫn và giúp đỡ chúng em thực hiện hoàn thành đề tài này!

CHƯƠNG 1: MÔ TẢ ĐỀ TÀI

1.1. Requirements

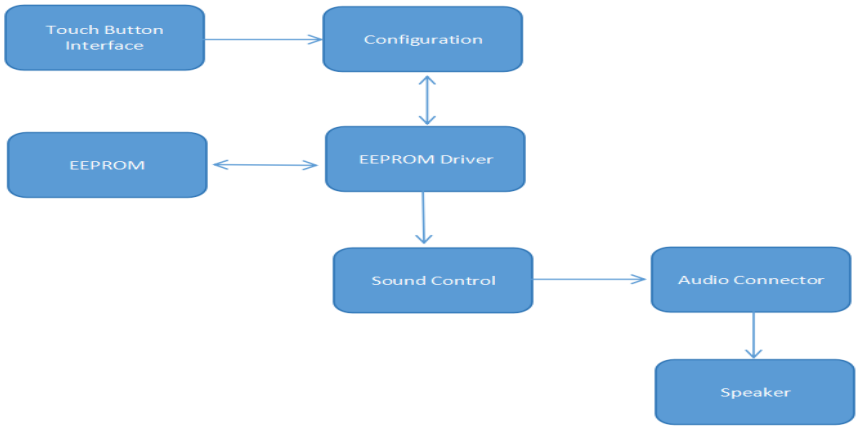
- Name: Bàn phím âm nhạc đơn giản
- Input: 13 phím đàn: C, C#, D, D#, E, F, F#, G, G#, A, A#, B, C
- Output: Tín hiệu âm thanh gửi ra loa phát âm thanh
- Functions: Khi tay người chơi chạm vào các phím đàn sẽ có sự thay đổi trở kháng, dẫn đến kích hoạt cảm biến trở kháng, xuất tín hiệu vào vi điều khiển

1.2. Performance:

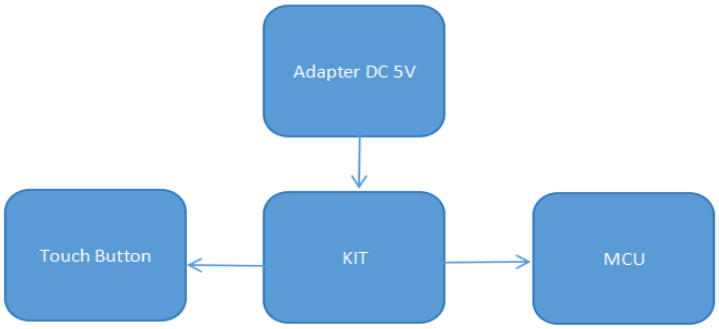
- Manufacturing costs:

Linh kiện	**Số lượng**	**Đơn giá**	**Thành tiền**
PIC 16F877A MCU	1	95.000	95.000
Đế kit PIC	1	75.000	75.000
2SC1815 Transistor NPN	13	400	5.200
Jack chuyển đổi Audio	1	12.000	12.000
Điện trở 10k, 2k ohm, 1k			5.000
Tụ điện 220nF			2.000
Tổng cộng			194.200

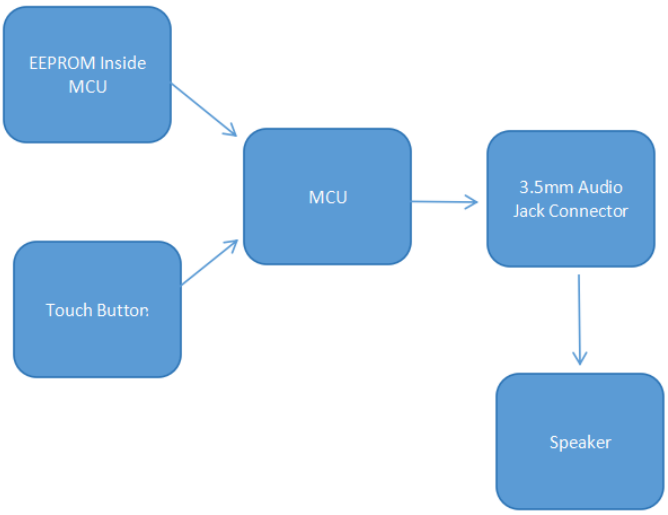
- Power: Adaptor 5V DC
- Sơ đồ khối:



Hình 1.1: System Block Diagram



Hình 1.2: Power Diagram



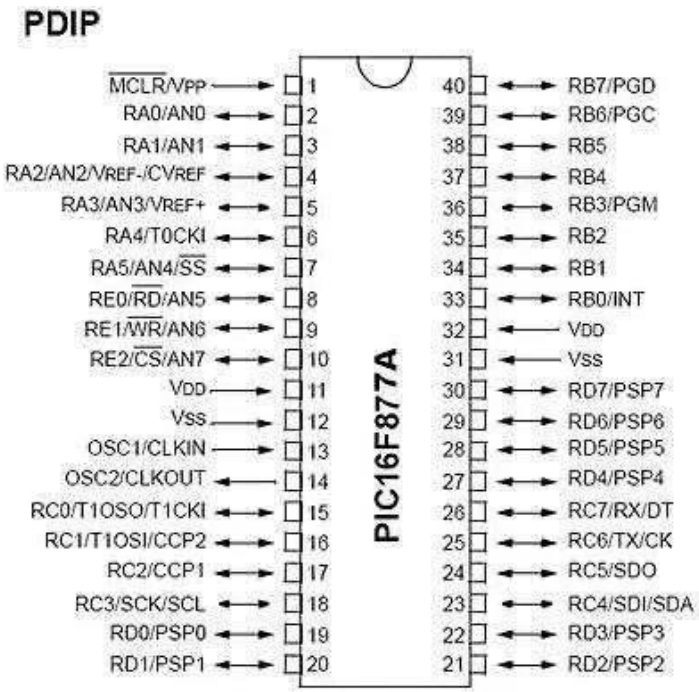
Hình 1.3: Hardware Diagram

CHƯƠNG 2: TỔNG QUAN VÀ GIỚI THIỆU CHUNG VỀ CÁC THÀNH PHẦN CỦA MẠCH

2.1. PIC 16F877A

2.1.1. PIC 16F877A là gì?

PIC16F877A là một Vi điều khiển PIC 40 chân và được sử dụng hầu hết trong các dự án và ứng dụng nhúng. PIC16F877A có năm cổng bắt đầu từ cổng A đến cổng E. Vi điều khiển này có ba bộ định thời trong đó có 2 bộ định thời 8 bit và 1 bộ định thời là 16 Bit. Nó hỗ trợ nhiều giao thức giao tiếp như giao thức nối tiếp, giao thức song song, giao thức I2C. PIC16F877A hỗ trợ cả ngắt chân phần cứng và ngắt bộ định thời.



Hình 2.1 : Hình ảnh PIC và sơ đồ các chân

PIC16F877A có thể được tìm thấy trên nhiều thiết bị điện tử, được sử dụng cho các thiết bị điều khiển từ xa, thiết bị bảo mật và an ninh, các thiết bị tự động trong gia đình và một số các thiết bị trong công nghiệp.

EEPROM được trang bị cho PIC16F877A giúp lưu trữ một số thông tin vĩnh viễn như mã bộ truyền, tần số bộ nhận và các dữ liệu liên quan khác. Chi phí cho bộ vi điều khiển này thấp.

Một số đặc tính kỹ thuật của PIC16F877A:

- Có 35 bộ lệnh.
- Hoạt động với tần số lên đến 20MHz.
- Điện áp hoạt động từ 4.2V đến 5V.
- Không có bộ dao động bên trong giống như PIC18F46K22, PIC18F4550.
- Dòng điện tối đa trên mỗi PORT khoảng 100mA. Do vậy, dòng điện tối đa cho mỗi chân GPIO của PIC 16F877A là 10mA.
- Có 4 package: PDIP 40 chân, PLCC 44 chân, TQFP 44 chân, QFN 44 chân.

2.1.2. Một số tính năng của PIC16F877A

Giống như tất cả các vi điều khiển khác, PIC16F877A cũng cung cấp các tính năng hữu ích được tích hợp sẵn như được đề cập trong danh sách sau:

Module chuyển đổi Analog sang Digital: Có module ADC 8 bit có 8 kênh. Chúng ta có thể sử dụng 8 cảm biến analog với bộ vi điều khiển này.

Timer - bộ định thời: Có ba bộ định thời là TIMER0, TIMER1 và TIMER2. Tất cả các timer này có thể được sử dụng ở chế độ định thời hoặc ở chế độ đếm. Các bộ định thời được sử dụng để tạo độ trễ, điều chế độ rộng xung, đếm các sự kiện bên ngoài và ngắt bộ định thời. TIMER0 là bộ định thời 8 bit và có thể hoạt động với tần số xung nhịp bên trong hoặc bên ngoài. Khi sử dụng TIMER0 ở chế độ định thời, chúng ta thường sử dụng tần số bên trong và ở chế độ bộ đếm, chúng ta nên sử dụng nguồn xung nhịp bên ngoài. Tương tự, TIMER1 là bộ định thời 16 bit và nó cũng có thể hoạt động ở cả hai chế độ. TIMER2 cũng là 8-bit. Nó còn được sử dụng với chế độ PWM làm cơ sở thời gian cho module CCP.

EEPROM: Được tích hợp cho module với 256 x 8 byte có thể được sử dụng để lưu trữ dữ liệu ngay cả khi bộ vi điều khiển bị tắt, dữ liệu sẽ không mất đi. Nó thường được sử dụng với các dự án liên quan đến electronics lock.

Module PWM: Có 2 module CCP. CCP là viết tắt của cụm từ capture compare PWM. Có thể dễ dàng tạo ra hai tín hiệu PWM với bộ vi điều khiển này. Độ phân giải

tối đa 10 bit. Có thể tham khảo hướng dẫn sử dụng PWM trong vi điều khiển PIC16F877A để biết thêm thông tin và lập trình.

Chân giao tiếp nối tiếp UART: Hỗ trợ một kênh UART. Chân UART được sử dụng để giao tiếp nối tiếp giữa các thiết bị digital. Chân RC7 (chân 26) là chân phát tín hiệu hoặc chân RX. RC6 (chân 25) là chân thu tín hiệu hoặc chân Tx. Để biết thêm chi tiết, hãy xem hướng dẫn về giao tiếp nối tiếp vi điều khiển pic16f877a .

Giao tiếp I2C: Có module giao tiếp I2C. Chân 18/RC3 và 230/RC4 lần lượt là chân SCL và SDA. SCL là đầu cấp xung nhịp và SDA là đường dữ liệu nối tiếp. Hướng dẫn giao tiếp I2C sẽ giúp bạn hiểu sâu hơn.

Ngắt: Ngắt có nhiều ứng dụng tuyệt vời trong lĩnh vực hệ thống nhúng. Vi điều khiển PIC16F877A cung cấp 8 loại ngắt là: Ngắt ngoài, ngắt bộ định thời, ngắt thay đổi trạng thái PORT, ngắt UART, ngắt I2C, PWM. bạn có thể đọc hướng dẫn này về ngắt vi điều khiển pic để biết thêm thông tin.

Module so sánh: Có một module so sánh bao gồm hai trình so sánh. Được sử dụng để so sánh các tín hiệu analog giống như bộ so sánh trong các mạch điện tử. Các chân đầu vào cho các bộ so sánh này là RA0, RA1, RA2 và RA3 và đầu ra là RA4 và RA5.

Bộ định thời watchdog: WDT là một bộ dao động riêng trên chip chạy tự do. Nó là một bộ dao động riêng biệt từ chân OSC1/CLKI. WDT cũng sẽ hoạt động ngay cả khi thiết bị ở chế độ ngủ. Được sử dụng để khởi động thiết bị từ chế độ ngủ và thiết lập lại bộ đếm thời gian watchdog.

Chế độ ngủ: PIC16F877A có chế độ ngủ. Ở chế độ này, thiết bị hoạt động ở mức công suất rất thấp. Tất cả các thiết bị ngoại vi sử dụng một dòng điện tối thiểu. Khởi động từ chế độ ngủ trong ngắt như ngắt timer1, ngắt uart, khi hoàn thành ghi EEPROM và nhiều thao tác khác.

Phát hiện brownout: Có một mạch phát hiện sự sụt giảm điện áp cung cấp. Nếu điện áp cung cấp giảm từ một giới hạn nhất định, nó sẽ tạo ra tín hiệu ngắt. Bit cấu hình này (BODEN) được sử dụng để tắt hoặc bật tính năng này.

Reset brownout: Tuỳ chọn này đặt lại thiết bị khi phát hiện sụt áp từ tín hiệu BODEN. Nếu điện áp cung cấp thấp hơn ngưỡng hơn 100 micro giây, code bảo vệ có

thể lập trình, thiết lập lại sụt áp sẽ xảy ra và thiết bị sẽ vẫn được đặt lại cho đến khi điện áp tăng đến giá trị định mức. Thiết bị kiểm tra điện áp sau mỗi 72ms.

2.1.3. Kit PIC Mini

Đế chân ra PIC được sử dụng giúp cho việc thiết lập và kết nối dễ dàng hơn.

Đế có tích hợp sẵn nút nhấn reset, cổng ICSP nạp chương trình, chân IO của các Port, dao động thạch anh, Jumper nguồn cho ngoại vi, Jumper giao tiếp UART, v.v…

Hình 2.2: Hình ảnh Kit PIC Mini ra chân dành cho PIC16F877A

2.1.4. Các linh kiện điện tử

- Op amp UA741
- Transistor C1815
- Điện trở loại 1K, 2K, 10K Ohm
- Jack 3.5mm female
- Thạch anh 12Mhz (tích hợp trong Kit PIC Mini)
- Tụ 220nF
- Dây nguồn DC 5V

2.1.5. Các thành phần khác:

- Breadboard

- Dây đồng làm phím nhấn
- Loa nghe nhạc

CHƯƠNG 3: CƠ SỞ LÝ THUYẾT VÀ XỬ LÝ PHẦN MỀM

3.1. Cơ sở lý thuyết

3.1.1. Phím cảm ứng chạm tay

Để thực hiện phím cảm ứng chạm tay, nhóm tác giả đã nghĩ ra một phương án là dựa vào nguyên lý hoạt động của transistor NPN và dùng transistor như một công tắc. Khi không có dòng ở B, transistor ở chế độ cut-off và không có dòng từ C xuống E. Khi có dòng vào B, transistor chuyển sang chế độ active và xuất hiện dòng lớn chảy từ C xuống E rồi xuống mass.

Áp dụng lý thuyết trên, nhóm tác giả cho mức điện áp đầu vào cực C theo mức điện áp hoạt động của PIC16F877A là +5V. Đồng thời trích 1 điểm trên đoạn từ cực E xuống mass vào chân input của vi xử lý, cực E được được nối đất thông qua điện trở 10K, còn cực B nhận tín hiệu từ tay ta chạm vào. Vì con người đóng vai trò là một chất dẫn điện nên khi tay ta chạm vào 2 thanh kim loại (cách ly) đặt gần nhau nối từ cực C và cực B, sẽ xuất hiện một dòng điện đi qua vào chân B và lúc này BJT sẽ hoạt động ở chế độ active, vi điều khiển nhận được tín hiệu 1.

3.1.2. Tạo âm thanh từ vi điều khiển

Âm thanh được tạo ra từ nhiều nốt nhạc có tần số (cao độ) khác nhau, mỗi nốt nhạc được tạo nên từ một tần số được quy định theo quy chuẩn quốc tế.

Note	Frequency (Hz)	Period (us)
C4	262	3817
C#4 / Db4	277	3610
D4	294	3401
Eb4	311	3215
E4	330	3030
F4	349	2865
F#4 / Gb4	370	2703
G4	392	2551
G#4 / Ab4	415	2410
A4	440	2273

A#4 / Bb4	466	2146
B4	494	2024
C5	523	1912

Con người có thể nghe âm thanh nhưng không thể thấy âm thanh, tuy nhiên chúng ta có thể biểu diễn nó dưới dạng sóng sin. Mỗi nốt nhạc là một sóng sin có biên độ khác nhau, các bản nhạc mà chúng ta thường nghe hằng ngày là tổng hợp của các sóng sin có tần số và biên độ khác nhau.

Để tạo âm thanh nốt nhạc bằng vi điều khiển, cụ thể ở đây là PIC16F877A, ta áp dụng lý thuyết tương tự như âm thanh trong đời sống hằng ngày chính là tạo sóng. Sóng càng mượt thì âm thanh sẽ càng mượt mà và êm ái. Tuy nhiên, vi điều khiển chỉ có thể tạo xung vuông cho nên phải có một hướng giải quyết phù hợp. Ở đây, nhóm tác giả nghĩ ra 2 phương án để tạo một tín hiệu âm thanh có tần số tương ứng với một nốt nhạc.

Đối với việc tạo xung vuông: Muốn tạo được một xung vuông mà chúng ta có thể nghe được giống như âm thanh của một note nhạc có tần số nhất định từ vi điều khiển, ta sẽ tạo xung vuông có tần số (chu kì) tương ứng với nốt nhạc đó với chu kỳ làm việc 50%. Cụ thể muốn tạo được âm thanh của note A4 (tần số 440Hz), ta sẽ tạo một xung vuông có tần số 440Hz (chu kì 2273ms), thời gian tín hiệu ở mức cao sẽ là 2273/2 = 1136ms, thời gian tín hiệu ở mức thấp sẽ là 1136ms. Như vậy với các nốt nhạc khác, ta sẽ lập trình cho vi điều khiển xử lý tương tự.

Đối với việc tạo sóng sine: Ta sẽ lấy mẫu biên độ của một sóng sine hoàn chỉnh và xuất các giá trị đó từ vi điều khiển ra các chân I/O. Sau đó, ta sử dụng mạch chuyển đổi tín hiệu Digital thành tín hiệu Analog để tạo các sóng sine có tần số tương ứng với các nốt nhạc được nhấn.

Nhóm sẽ tìm hiểu, nghiên cứu và thực hiện cả 2 phương án trên để so sánh kết quả đạt được. Sau đó nhóm sẽ chọn ra phương án tốt nhất, tối ưu nhất để trình bày phần cứng đến thầy.

3.2. Xử lý phần mềm

3.2.1. Ngôn ngữ, trình biên dịch, mạch nạp

Nhóm sẽ chọn lập trình cho vi điều khiển thực hiện dự án này bằng ngôn ngữ lập trình C với trình biên dịch CCS C Compiler, phần mềm PIC C Compiler. Đồng thời, nhóm sẽ dùng mạch nạp PICKIT2 với phần mềm PICkit2 để nạp chương trình cho vi điều khiển PIC16F877A

Nhóm đã thử qua việc lập trình trên phần mềm MPLAB. Tuy nhiên, quá trình nạp chương trình được viết từ MPLAB cho vi điều khiển thông qua mạch nạp PICKIT2 và phần mềm PICkit2 không như mong muốn. Khi import file hex của MPLAB vào phần mềm PICkit2, các mã lệnh không được đặt tại địa chỉ đầu tiên của bộ nhớ chương trình mà nằm tại vị trí khác và không cố định. Dẫn đến khi nạp chương trình cho vi điều khiển thì vi điều khiển không thể hoạt động như đã lập trình.

3.2.2. Phương án sử dụng module PWM được tích hợp trên vi điều khiển

Ý tưởng: Ta sẽ xuất xung vuông với các chu kì khác nhau bằng cách dùng module PWM được tích hợp trên vi điều khiển PIC16F877A tại chân RC1 hoặc RC2. Đầu tiên, ta phải cấu hình chân đó hoạt động ở chế độ PWM:

```
setup_ccp1(CCP_PWM);
```

Vì module PWM được tích hợp trên PIC16F877A lấy Timer2 làm cơ sở về thời gian, nên ta sẽ cài đặt các thông số cho Timer2:

```
setup_timer_2 (mode, period, postscale)
```

Chu kì (tần số) của xung PWM tạo được sẽ phụ thuộc vào các thông số như sau:

$$PWMPeriod = [(PR2) + 1] * 4 * TOSC * (TMR2PrescaleValue)$$

Như vậy, việc đặt các thông số phù hợp, ta sẽ tạo được xung PWM với chu kì mong muốn.

Kết quả: Đối với dự án này, ta cần phải xuất các xung vuông có chu kì khác nhau, tương ứng với các nốt nhạc (phím nhấn). Việc này đòi hỏi Timer2 cần phải báo tràn tại các thời điểm khác nhau. Nhóm viết một chương trình đặt một biến kiểu int vào vị trí Period của hàm setup_timer_2(mode,period, postscale). Sau đó, mỗi khi

phím nào được nhấn thì sẽ gán giá trị chu kì (đặt trước) cho biến đó. Và sau đó gọi lại hàm setup_timer_2(mode,period,postscale). Tuy nhiên, có vẻ hàm này yêu cầu ta phải nhập một con số cụ thể chứ không thể nhập vào đó một biến được. Việc này có lẽ do sự sai lệch thời gian do tốc độ gọi biến và thực thi lệnh của ngôn ngữ C không phù hợp với hàm setup_timer_2.

Quyết định: Chính vì vậy, nhóm quyết định không dùng phương án này và sẽ chuyển sang tiếp cận hướng xử lý khác.

3.2.3. Phương án sử dụng Timer để tạo xung

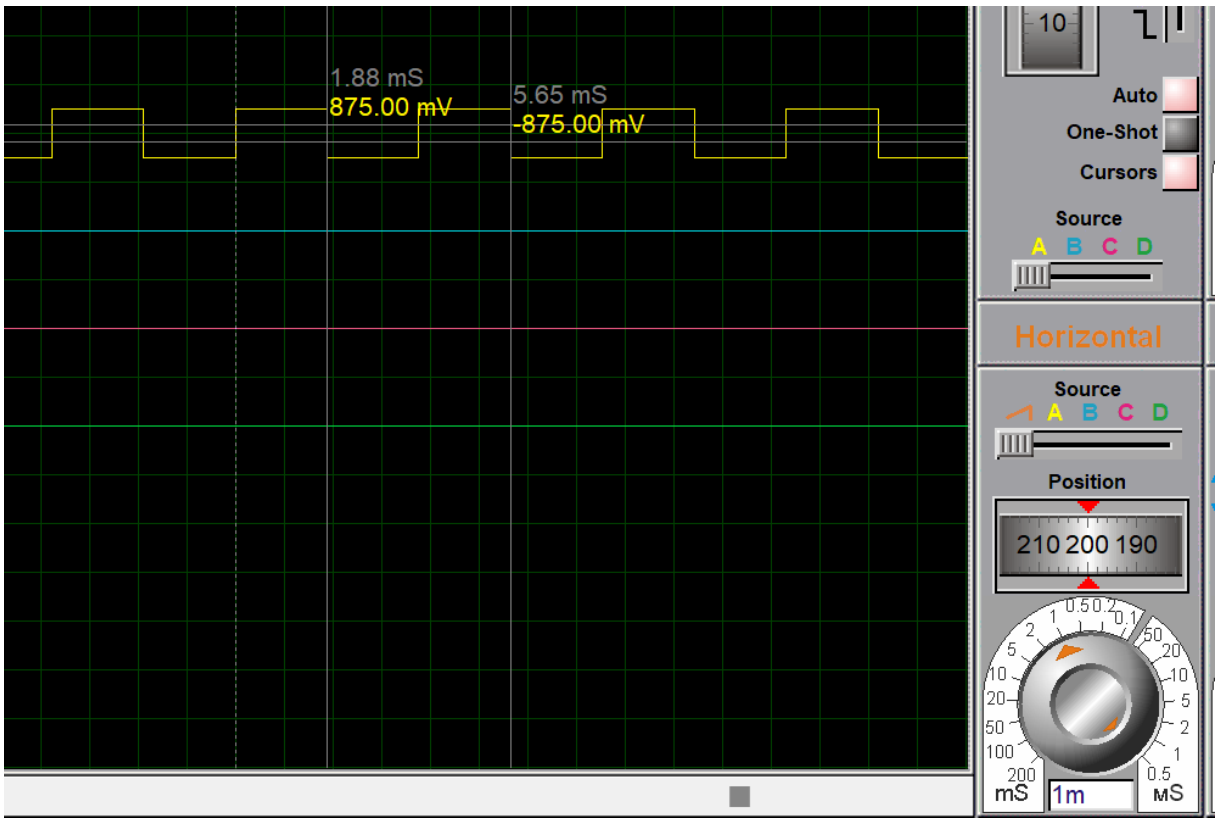
Ý tưởng: Ta có thể sử dụng Timer được tích hợp trên vi điều khiển để xuất xung. Khi có phím được nhấn, đầu tiên ta sẽ set timer bắt đầu từ 0 (cho timer chạy)

```
set_timer0(0);
```

Sau đó, chương trình sẽ vào vòng lặp và sẽ liên tục kiểm tra giá trị của Timer1, so sánh với giá trị đặt ra (tương ứng với chu kì mong muốn). Khi giá trị Timer1 đã được được giá trị mong muốn (mà ở đây là thời gian 1 nửa chu kì của một xung). Ta sẽ đảo trạng thái cho chân Output

```
while(button_0){
    set_timer0(0);
    value = 0;
    while(value<x[0]){
    value = get_timer0();}
    output_toggle(PIN_C0); }
```

Mô phỏng: Khi thiết kế và mô phỏng trên proteus, nhóm đã nhận thấy xung được tạo ra từ chân RC0 khá chính xác. Hình dưới đây thể hiện xung vuông với chu kì mong muốn 3817us được tạo từ Timer0. Với xung tạo được có chu kì thực tế 3770us.



Kết quả: Tuy nhiên, khi đổ code lên mạch thử nghiệm và chạy với loa phát nhạc (có tải), nhóm nhận thấy tín hiệu âm thanh bị suy giảm rất nhiều về biên độ và chất lượng, và tiếng không còn đúng với tần số của nốt nhạc.

Quyết định: Do đó, nhóm quyết định không dùng phương án này để thực hiện trên mạch thật. Nhóm sẽ chuyển qua nghiên cứu phương án khác.

3.2.4. Phương án sử dụng Delay để tạo xung

Ý tưởng: Ta có thể tạo xung vuông có chu kỳ mong muốn bằng việc cấu hình chân output ở mức cao trong một khoảng thời gian bằng nửa chu kì của xung, sau đó cấu hình ở mức thấp trong một khoảng thời gian bằng nửa chu kì của xung. Việc này đòi hỏi cần có một hàm delay để tạo trễ. Trên phần mềm PIC C Compiler có hỗ trợ hàm delay_us(time) cho ta.

Syntax:	delay_us (***time***)
Parameters:	***time*** - a variable 0-65535(int16) or a constant 0-65535 Note: Previous compiler versions ignored the upper byte of an int16, now the upper byte affects the time.
Returns:	undefined
Function:	Creates code to perform a delay of the specified length. Time is specified in microseconds. Shorter delays will be INLINE code and longer delays and variable delays are calls to a function. This function works by executing a precise number of instructions to cause the requested delay. It does not use any timers. If interrupts are enabled the time spent in an interrupt routine is not counted toward the time. The delay time may be longer than requested if an interrupt is serviced during the delay. The time spent in the ISR does not count toward the delay time.
Availability:	All devices
Requires:	#USE DELAY

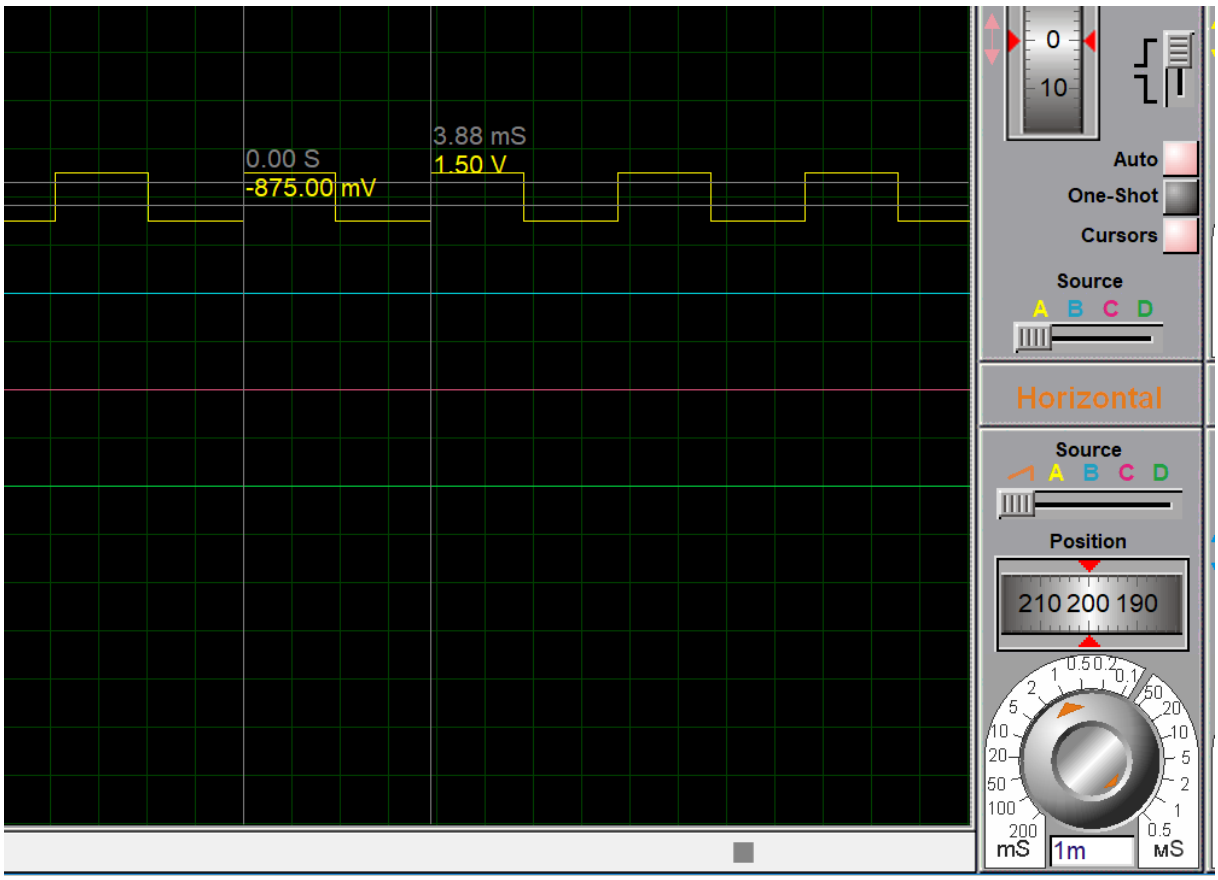
Hàm này yêu cầu ta phải khai báo trước xung clock hoạt động mà ta dùng cho vi điều khiển. Ở đây, nhóm dùng thạch anh 12MHz. Do đó nhóm sẽ khai báo như sau:

```
#use delay(crystal=12000000)
```

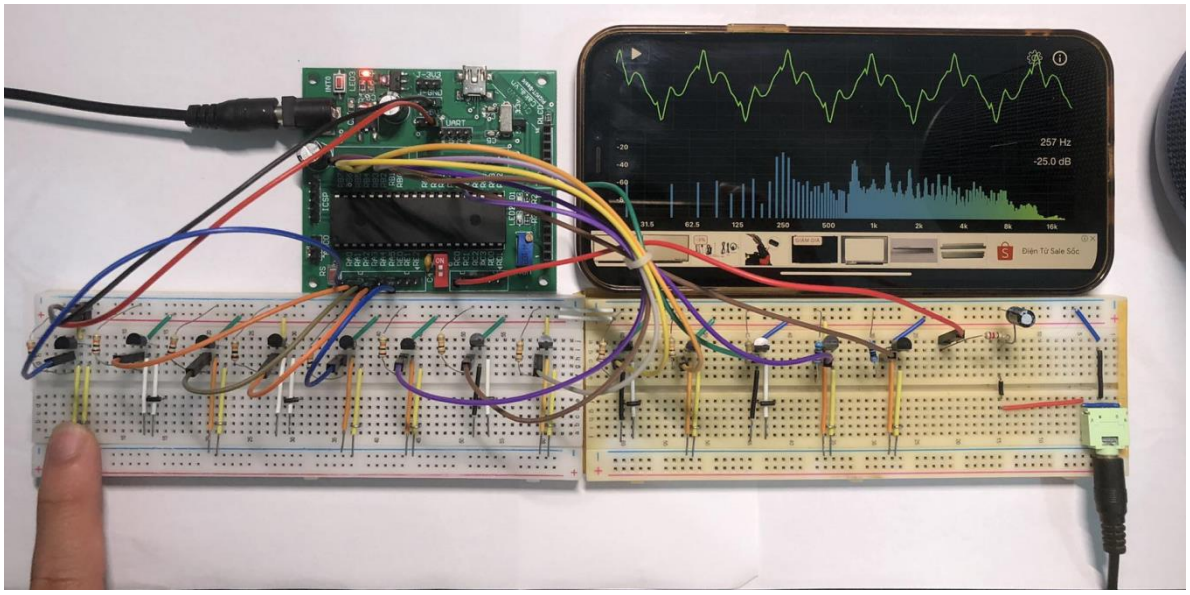
Sau đó, khi muốn tạo xung có chu kì 3817us (tương ứng với nốt nhạc C4 có tần số 262Hz). Ta cần phải tạo trễ khi cấu hình chân ở mức cao trong 1908us, tạo trễ khi cấu hình chân ở mức thấp trong 1908us.

```
delay_us(3818/2);
```

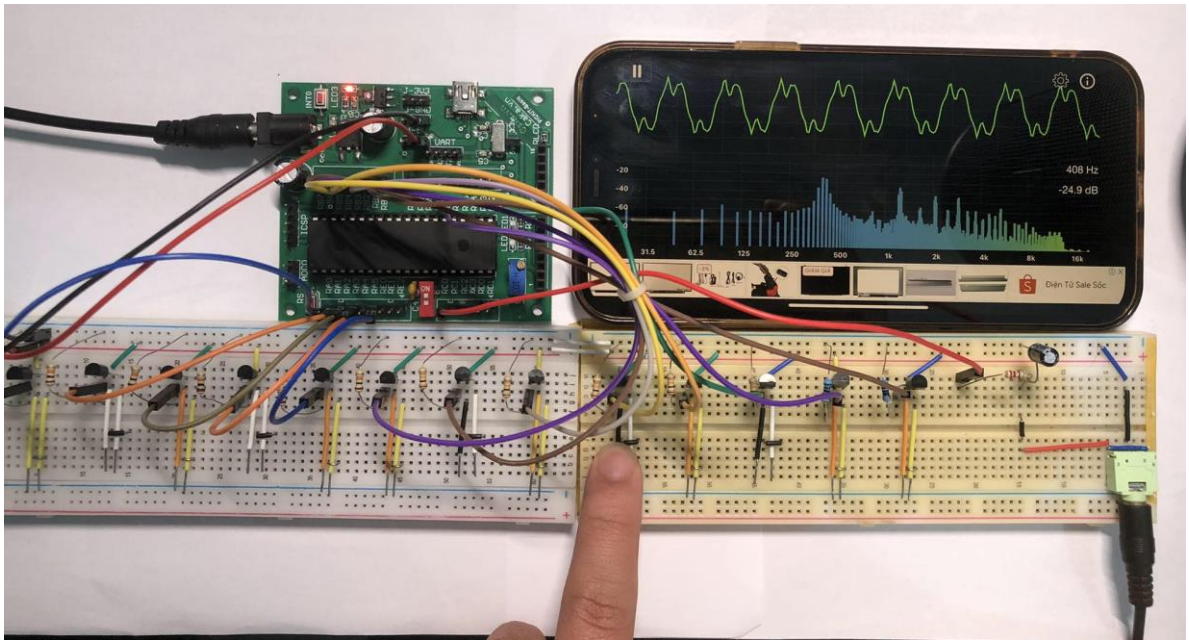
Mô phỏng: Khi mô phỏng trên Proteus, nhóm nhận thấy vi điều khiển tạo được xung vuông có chu kì thực tế là 3880us, khá chính xác so với chu kì mà ta mong muốn là 3817us.



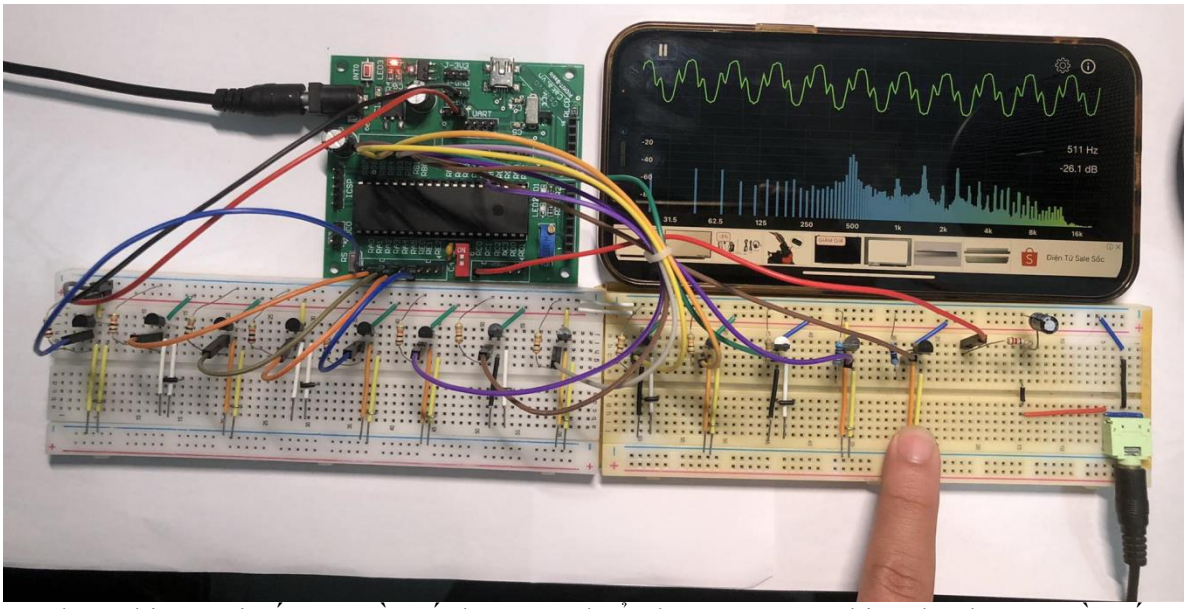
Kết quả: Khi nhóm thực hiện trên mạch thử nghiệm với loa phát nhạc (có tải). Âm thanh, được tạo từ xung vuông xuất từ vi điều khiển, có biên độ lớn và tín hiệu không bị suy giảm. Âm thanh phát ra chất lượng và gần đúng với tần số của nốt nhạc mà ta mong muốn.



Thử nghiệm với nốt C4 tần số theo quy chuẩn là 262 Hz. Tín hiệu thu được có tần số 252Hz



Thử nghiệm với nốt G#4 / Ab4 - tần số theo quy chuẩn là 415 Hz. Tín hiệu thu được có tần số 408 Hz



Thử nghiệm với nốt C5 - tần số theo quy chuẩn là 523 Hz. Tín hiệu thu được có tần số 511Hz

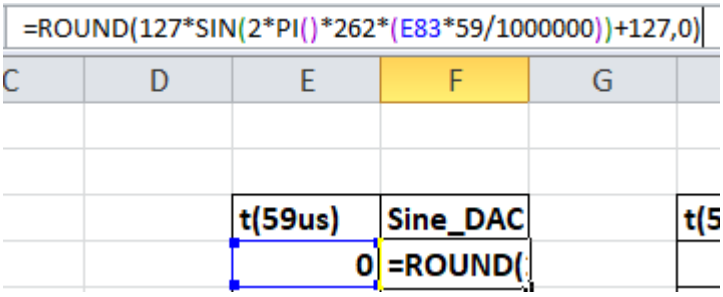
Kết luận: Như vậy, phương án sử dụng delay để tạo xung vuông đã đạt được kết quả tốt. Tuy nhiên, tín hiệu được tạo ra mang hình dạng xung vuông nên âm thanh chưa được mượt mà.

3.2.5. Tạo sóng sine với mạch DAC:

Như kết quả đã trình bày ở trên, việc sử dụng hàm delay để tạo xung đã tạo ra tín hiệu gần đúng với âm thanh của các nốt nhạc. Tuy nhiên, tín hiệu đó vẫn là xung vuông, hình dáng của xung vuông không mượt mà, chưa tối ưu cho việc tạo âm thanh - thứ mà ta cần hưởng thụ chúng một cách mượt mà. Do đó, nhóm đã thực hiện việc tạo sóng sine bằng vi điều khiển PIC16F877A với phương pháp DAC kiểu R-2R (hình T).

Đầu tiên, ta cần phải lấy mẫu một sóng sine tương đối chuẩn. Độ lớn của mẫu sẽ tương ứng với độ mượt mà và hoàn thiện của sóng sine ta sẽ tạo ra. Ở đây, ta sẽ lấy 64 mẫu của một sóng sine hoàn chỉnh. Ta thực hiện việc này bằng Excel. Hàm lấy mẫu như sau:

```
=ROUND(127*SIN(2*PI()*262*(E83*59/1000000))+127,0)
```

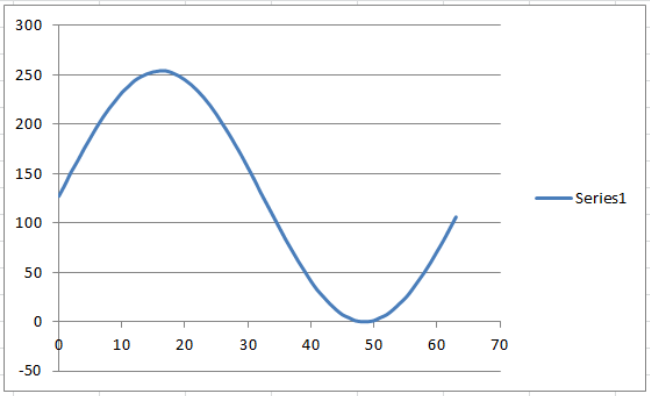


Trong đó: $sin(\frac{2\pi.t.262.59}{10^{-6}})$. Đây là hàm tạo sóng sine với tần số 262Hz, tốc độ lấy mẫu 17KHz

Vì sóng sine có biên độ dao động trong khoảng [-1;1]. Ta nhân hàm tạo sóng sine với 127 và cộng thêm một lượng 127 để tạo ra các mẫu dao động có giá trị biên độ từ 0 đến 255 (đây là khoảng giá trị mà vi điều khiển có thể xuất ra 1 port ở dạng nhị phân)

t(59us)	Sine_DAC
0	127
1	139
2	152
3	163
4	175
5	186
6	197
7	207
8	216
9	224
10	232
11	238
12	244
13	248
14	251
15	253
16	254
17	254
18	252
19	249
20	245
21	240
22	234
23	227
24	219
25	210
26	200
27	190
28	179
29	168
30	156
31	144

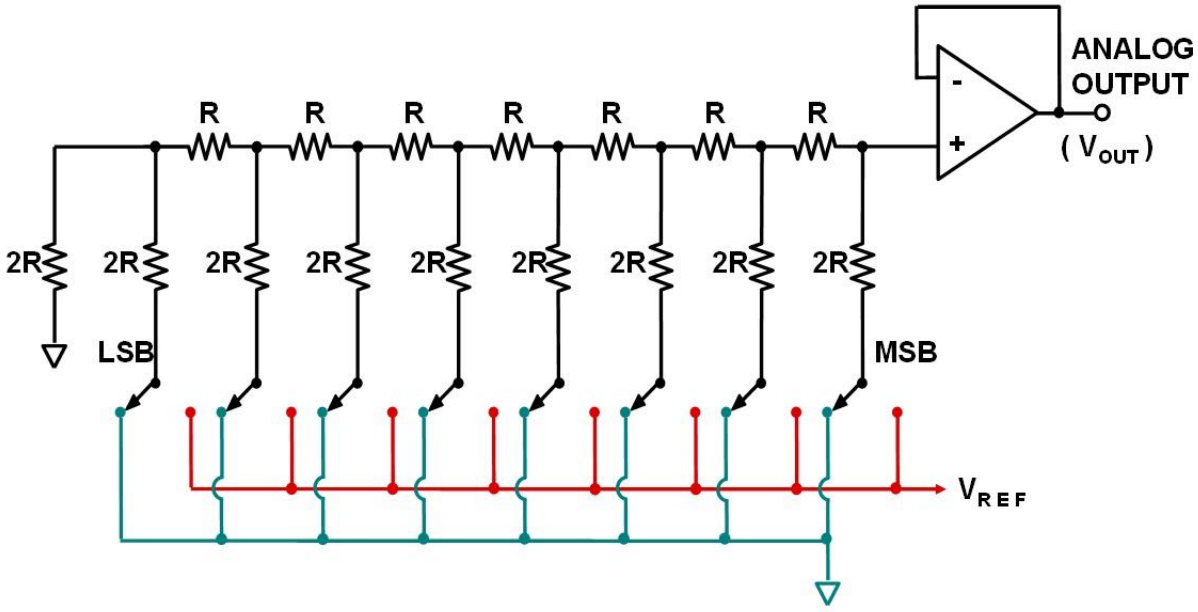
t(59us)	Sine_DAC
32	131
33	119
34	107
35	95
36	83
37	72
38	61
39	51
40	41
41	32
42	25
43	18
44	12
45	7
46	4
47	1
48	0
49	0
50	1
51	4
52	7
53	12
54	18
55	24
56	32
57	41
58	50
59	60
60	71
61	82
62	94
63	106



Kết quả lấy mẫu của một chu kỳ sóng sine chuẩn có tần số 262Hz với 64 mẫu.

Ta vẽ biểu đồ trên Excel để kiểm chứng việc lấy mẫu. Quan sát thấy rằng, đồ thị có dạng 1 chu kỳ sóng sine hoàn chỉnh, với biên độ cao nhất là 255, biên độ thấp nhất là 0. Vậy, việc lấy mẫu sóng sine đã thành công.

Tiếp đến, ta sẽ lựa chọn mạch DAC để thực hiện việc chuyển đổi tín hiệu Digital to Analog. Có 2 phương pháp thực hiện mạch DAC cho một tín hiệu Digital: phương pháp trở hồi tiếp và phương pháp R-2R. Vì lý do linh kiện đồng bộ, nhóm quyết định chọn phương pháp DAC kiểu R-2R. Phương pháp R-2R có sơ đồ mạch như sau:



Phương pháp DAC R-2R với các nhánh đường tín hiệu sẽ nối tiếp với nhánh chính thông qua điện trở 2R, các nhánh đường tín hiệu (đã bao gồm trở 2R) sẽ nối tiếp với nhau thông qua điện trở R. Mạch này sẽ chuyển 8 bit tín hiệu Digital thành 1 tín hiệu có dạng Analog. Ta khuếch đại tín hiệu Analog bằng 1 Op Amp.

Độ phân giải của mạch DAC-8 bit = $Q = \frac{Vcc}{2^B} = \frac{5}{2^8} = 0.195V$

Nhóm sử dụng port C trên vi điều khiển PIC16F877A để thực hiện việc xuất các giá trị lấy mẫu. Giá trị điện áp Vout được tính như sau:

$$Vo = (\frac{RCo}{256} + \frac{RC1}{128} + \frac{RC2}{64} + \frac{RC3}{32} + \frac{RC4}{16} + \frac{RC5}{8} + \frac{RC6}{4} + \frac{RC7}{2})x5V$$

Ta cần phải xuất 64 giá trị mà ta đã lấy mẫu ra port C để có thể chuyển đổi tín hiệu thành sóng sine khi đi qua mạch DAC. Để làm được điều này, nhóm tạo một chương trình con với biến cục bộ đưa vào là "period", thực hiện 64 vòng lặp, mỗi lần lặp sẽ xuất giá trị trong mảng DAC_Sin_Wave ra port C. Ta đặt vào biến đếm i để lấy lần lượt các giá trị trong mảng DAC_Sin_Wave.

```
void DAC(int period){
      for(int i =0; i<64; i++) {
         output_c(DAC_Sin_Wave[i]);
         delay_us(period);
      }
}
```

Mỗi khi chương trình chính phát hiện có phím được chạm, chương trình sẽ gọi chương trình con DAC ra và đặt giá trị period vào biến cục bộ của hàm.

Giá trị của biến cục bộ period đối với tần số 262Hz được tính như sau:

$$\text{Period of } 262\text{Hz} = \frac{T_{us}}{64} = \frac{3817\text{us}}{64} = 59$$

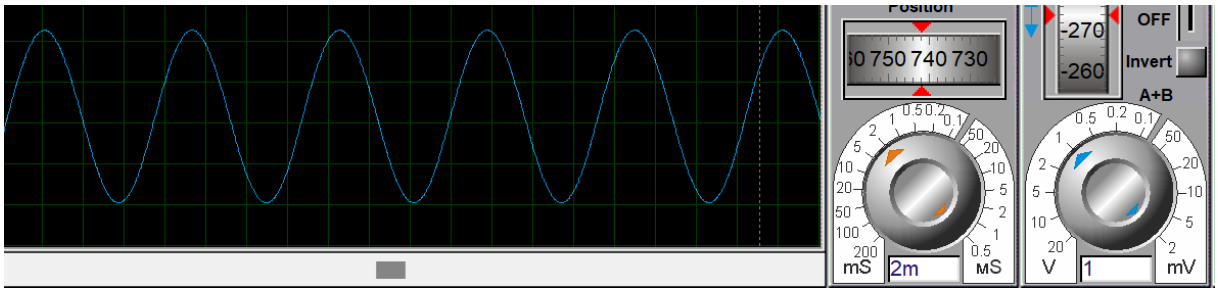
Kiểm tra lại = 1 vòng lặp x 64 lần x 59 us = 3776 us (gần đúng với chu kì của tần số 262Hz).

Chương trình được viết bằng ngôn ngữ C tốn nhiều thời gian xử lý lệnh và gọi mảng. Do đó, ta **trừ thêm 6us** vào mỗi giá trị Period tính được để tạo được tín hiệu có tần số chuẩn nhất. Vậy công thức cuối cùng sẽ như sau:

$$\text{Period of } 262\text{Hz} = \frac{T_{us}}{64} - 6 = \frac{3817\text{us}}{64} - 6 = 53$$

Đối với các tín hiệu tần số khác, ta tính toán, thay đổi biến cục bộ Period với công thức như trên để tạo các tín hiệu có tần số khác nhau.

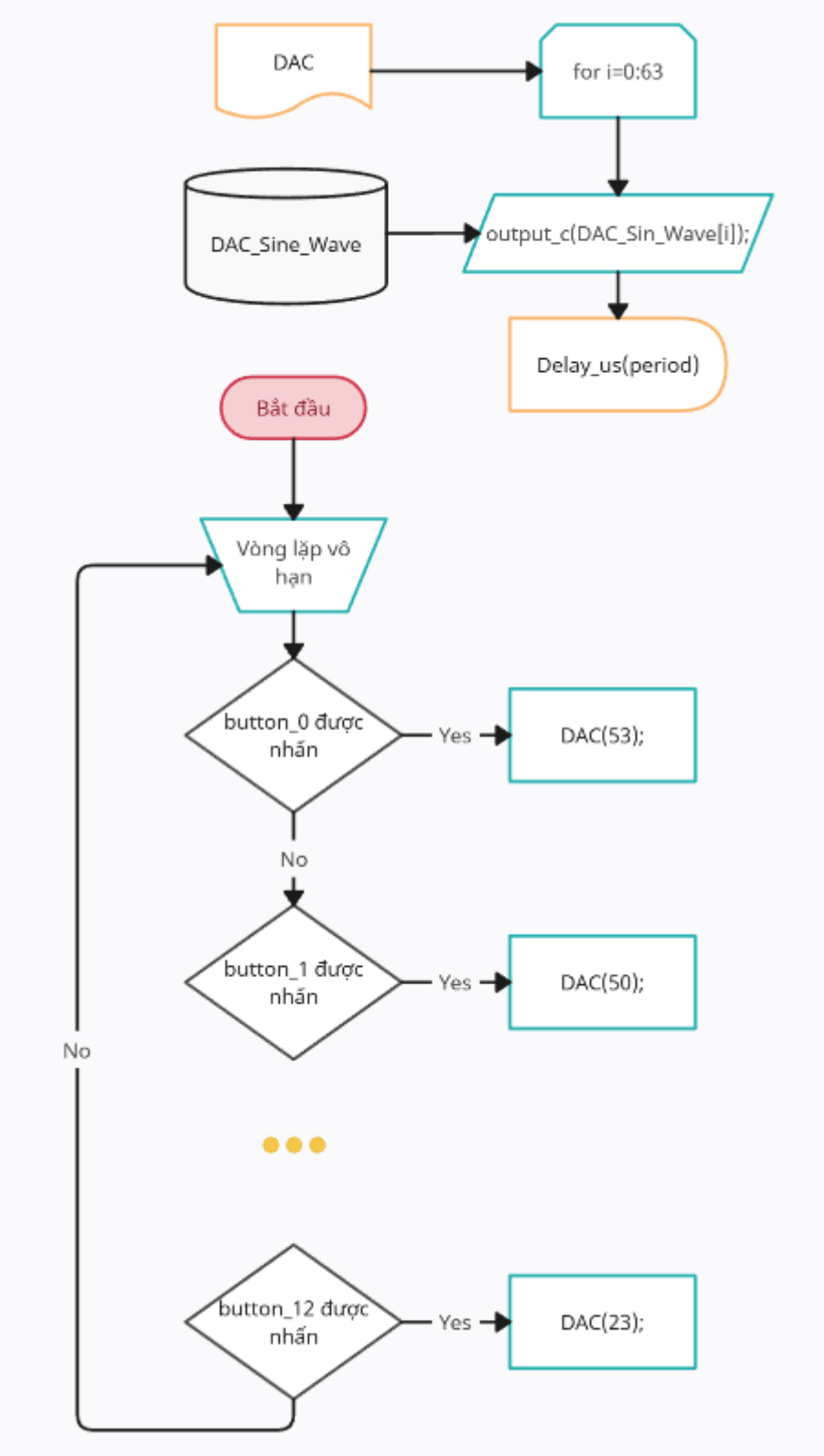
<u>Mô phỏng:</u> Khi mô phỏng sóng sine có tần số 262Hz trên Proteus:



<u>Kết luận:</u> Như vậy, phương án tạo sóng sine với mạch DAC kiểu R-2R đã được nhóm mô phỏng đạt kết quả thành công, sóng có dạng sine chuẩn, tần số đúng với tần số mà ta mong muốn. Khi kết nối ra loa, tín hiệu âm thanh mượt mà, không ngắt quãng.

<u>Quyết định:</u> Nhóm sẽ chọn phương án tạo sóng sine với mạch DAC kiểu R-2R để trình bày phần cứng với thầy Nguyễn Phan Hải Phú.

3.2.6. Lưu đồ giải thuật

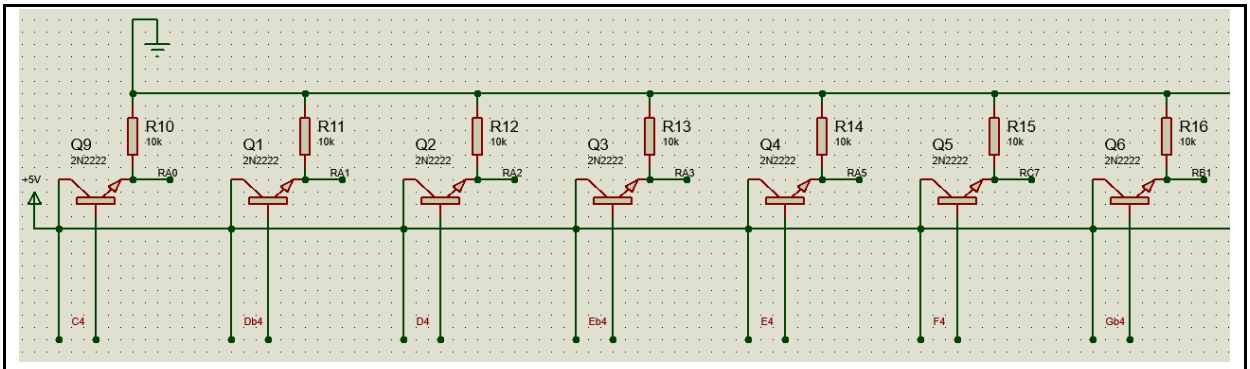


CHƯƠNG 4: THI CÔNG PHẦN CỨNG

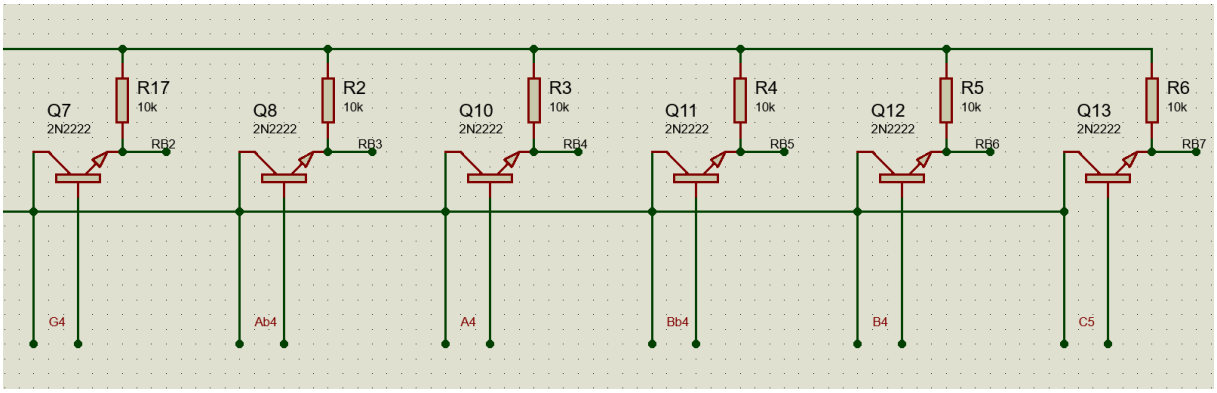
4.1. Mạch cảm biến điện trở chạm tay

Sử dụng một transistor, điện trở và 2 thanh dây dẫn cho mạch cảm biến chạm tay. Chân E của transistor sẽ nối mass thông qua một điện trở 10K, chân C sẽ được cấp nguồn, 2 thanh dây dẫn nối từ chân B và chân C (cách ly) làm cảm biến để tay chạm vào.

Khi có dòng vào B, transistor chuyển sang chế độ active và xuất hiện dòng lớn chảy từ C xuống E rồi xuống mass, một điểm trên đoạn E-mass sẽ chuyển từ tín hiệu mức thấp sang tín hiệu mức cao.



Hình 4.1: Sơ đồ mạch các cảm biến từ nốt Đô (C4) đến nốt Sol giáng (Gb4)

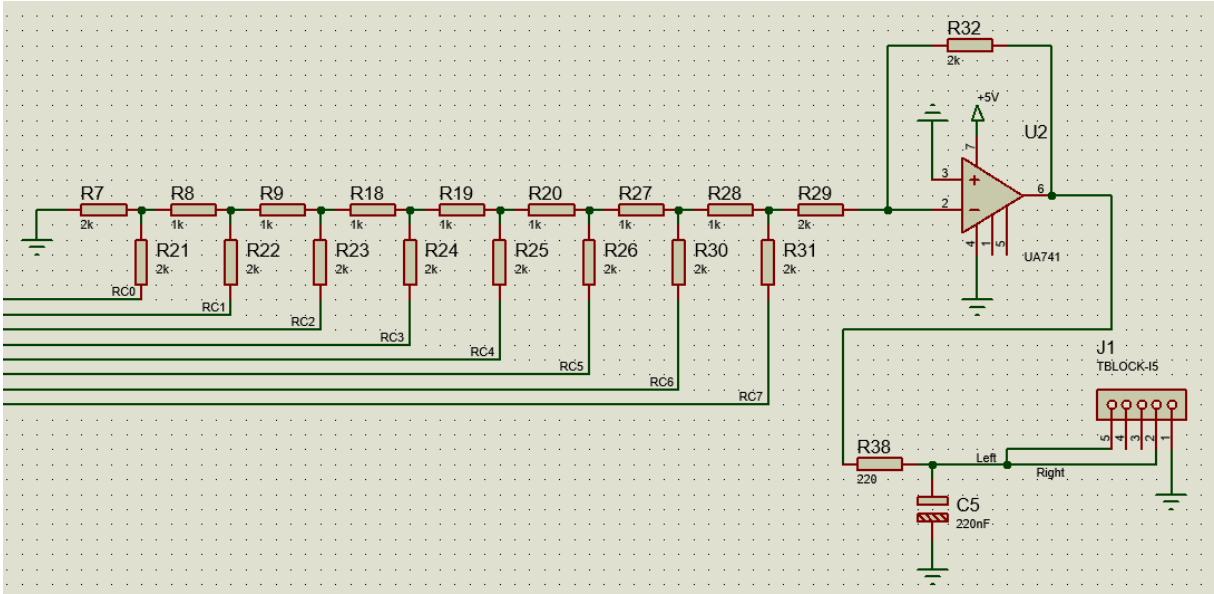


Hình 4.2: Sơ đồ mạch các cảm biến từ nốt Sol (G4) đến nốt Đô (C5)

4.2. Mạch DAC kiểu R-2R, 8 bit Digital Input

Các nhánh đường tín hiệu sẽ nối tiếp với nhánh chính thông qua điện trở 2K. Các nhánh đường tín hiệu (đã bao gồm trở 2K) sẽ nối tiếp với nhau thông qua điện trở

1K. Mạch này sẽ chuyển 8 bit tín hiệu Digital thành 1 tín hiệu có dạng Analog. Ta khuếch đại tín hiệu Analog bằng 1 Op Amp UA741 với trở hồi tiếp là 2K.



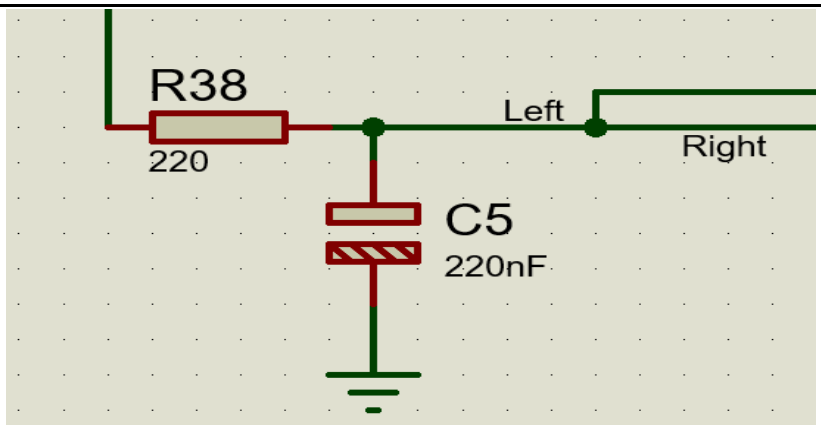
4.3. Mạch lọc thông thấp

Mạch lọc thông thấp là một mạch được thiết kế để loại bỏ tần số cao hơn không mong muốn của tín hiệu điện từ, tín hiệu âm thanh, tín hiệu điện và chỉ chấp nhận những tín hiệu được yêu cầu trong các mạch ứng dụng. Sự xuất hiện của mạch này trong sản phẩm giúp cho âm thanh nghe được trầm ấm hơn, và khử bỏ những tiếng rè không mong muốn.

$$f_c = \frac{1}{2\pi RC}$$

Ta chọn **C = 220 nF**, **R = 220 Ohm**

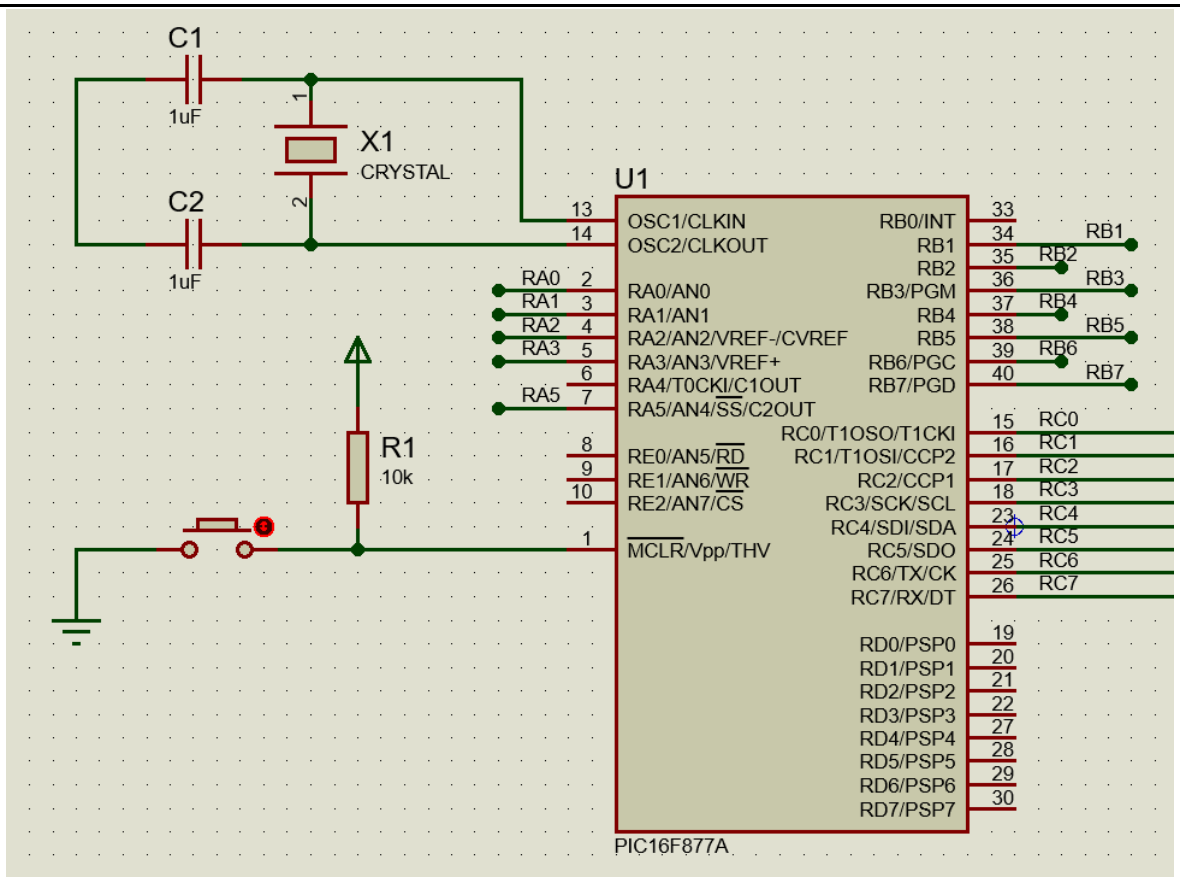
$$\text{--> } f_c = \frac{1}{2\pi.220.220.10^{-9}} \approx 3288\ Hz$$



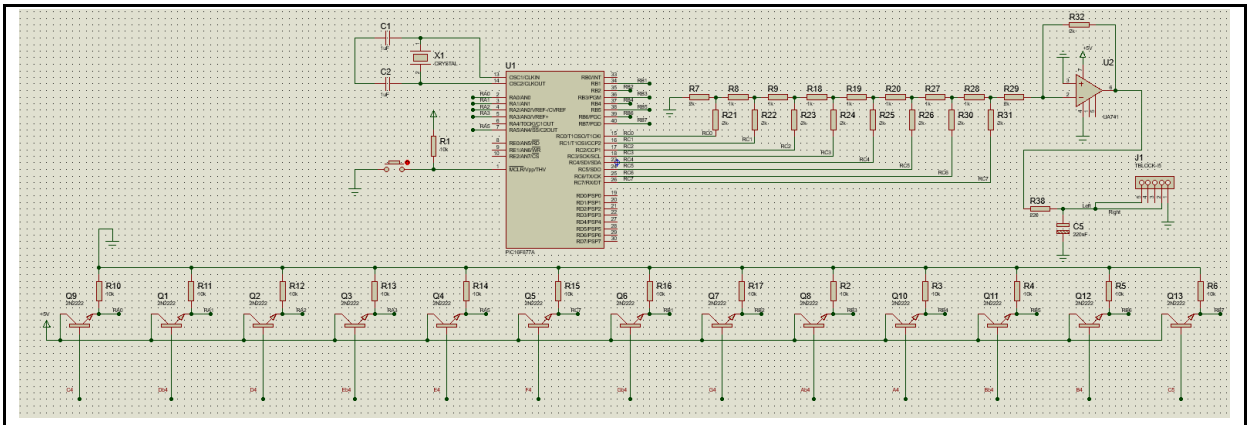
Hình 4.3: Mạch lọc thông thấp

Như vậy, tín hiệu sau khi qua mạch lọc thông thấp, sẽ chỉ còn những tín hiệu có tần số thấp hơn 3288 Hz, những tần số cao hơn 3288 Hz sẽ bị lọc bỏ. Tần số cao nhất của nốt nhạc mà nhóm thực hiện là 523 Hz (nốt C5). Do đó, tín hiệu tạo được từ vi điều khiển nằm trong băng thông cho phép.

4.4. Sơ đồ kết nối chân MCU

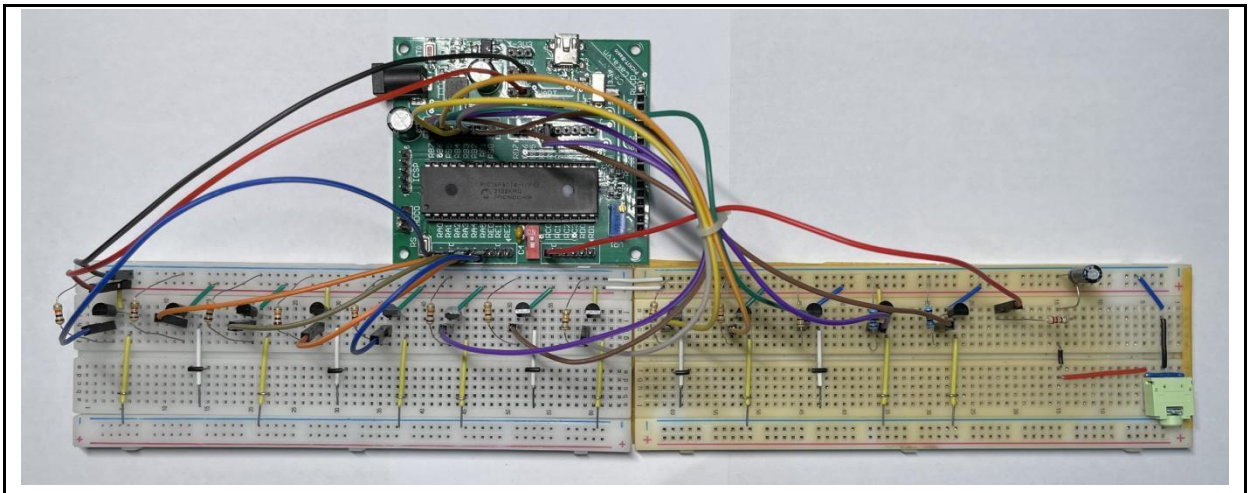


Hình 4.4: Khối vi xử lý

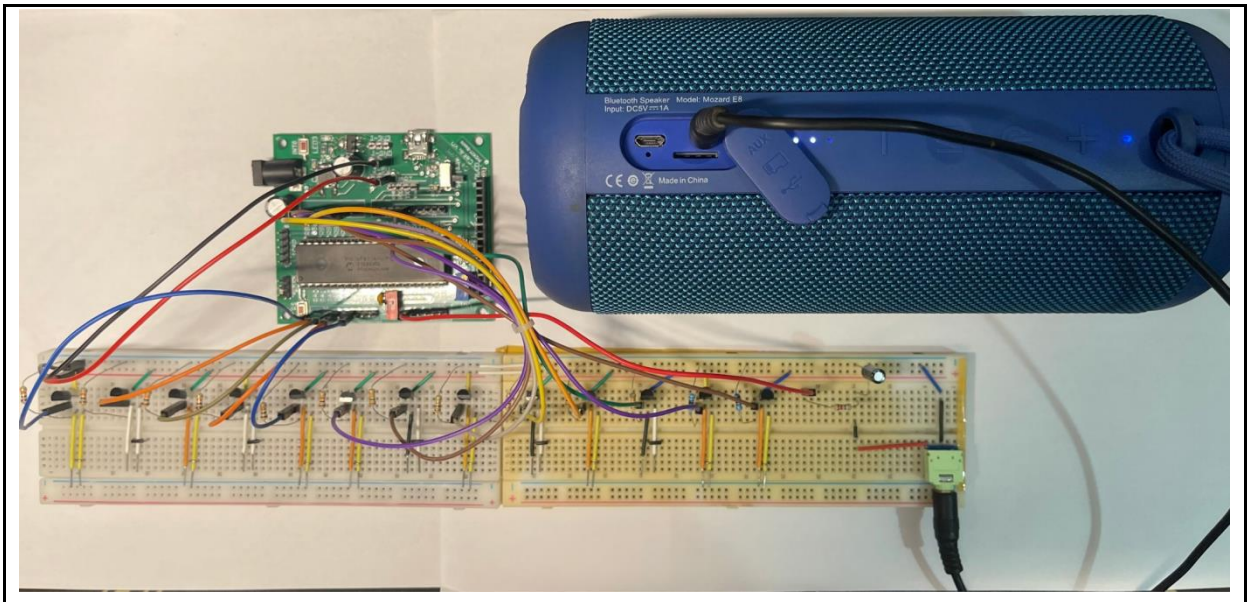


Hình 4.5: Toàn bộ sơ đồ mạch

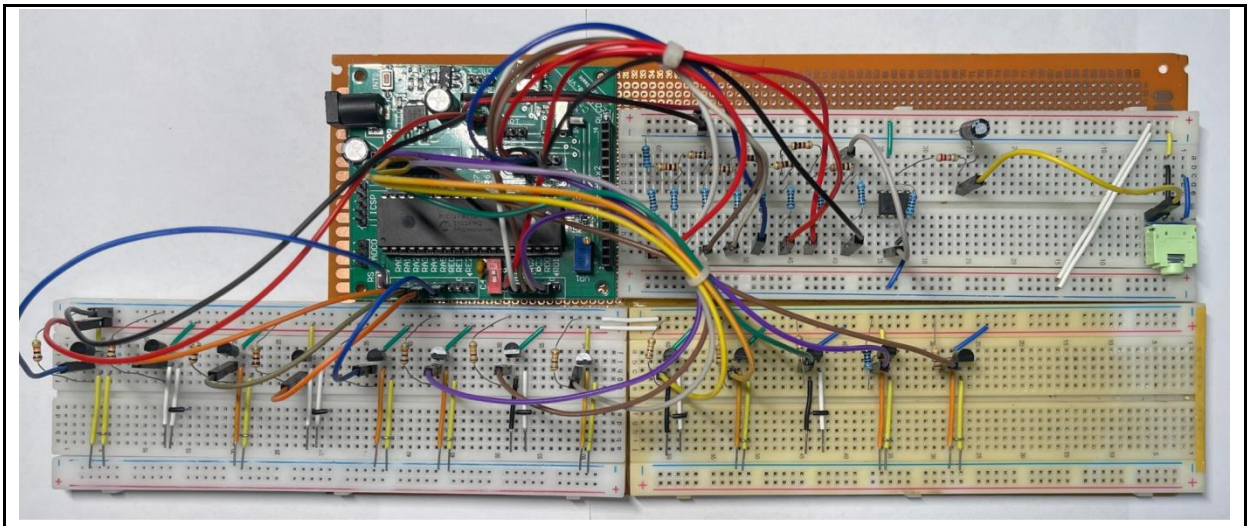
4.5. Hình ảnh sản phẩm



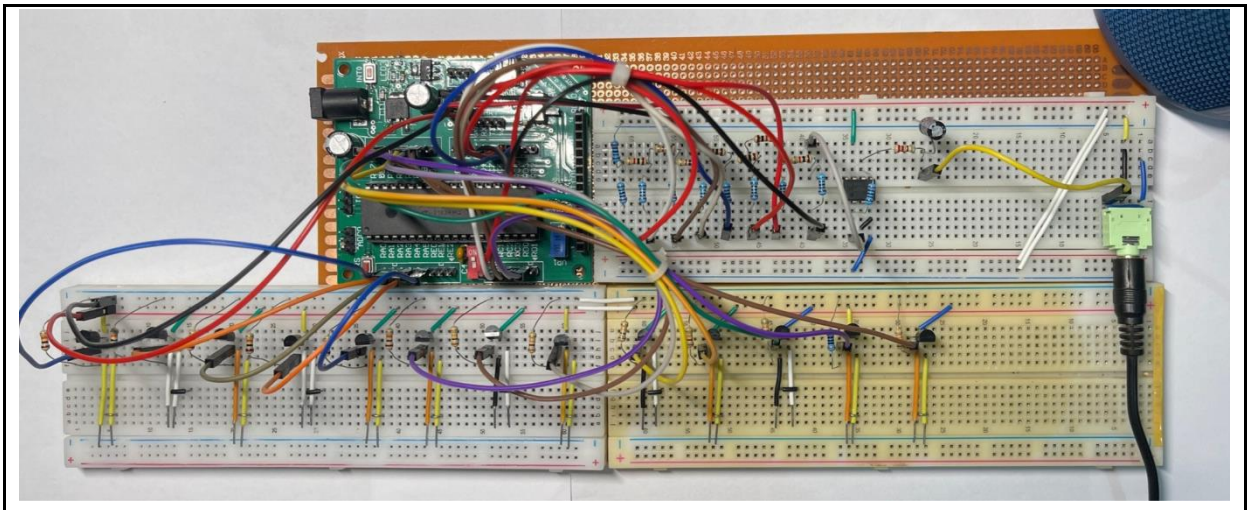
Hình 4.6: Sản phẩm theo phương án sử dụng delay để tạo xung



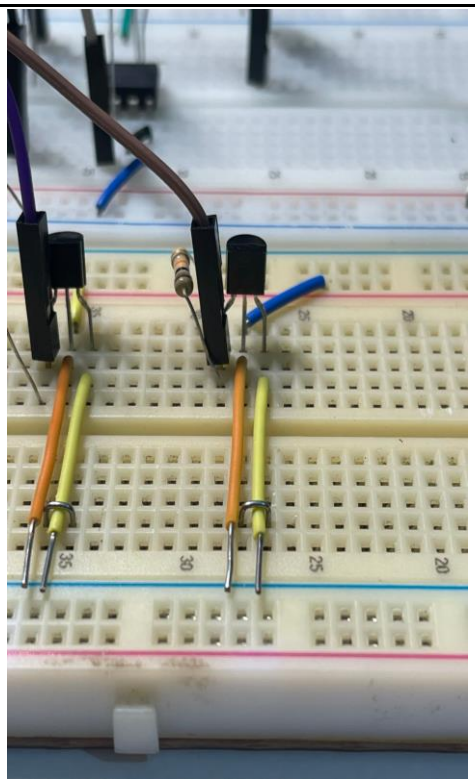
Hình 4.7: Sản phẩm được kết nối với loa nghe nhạc.



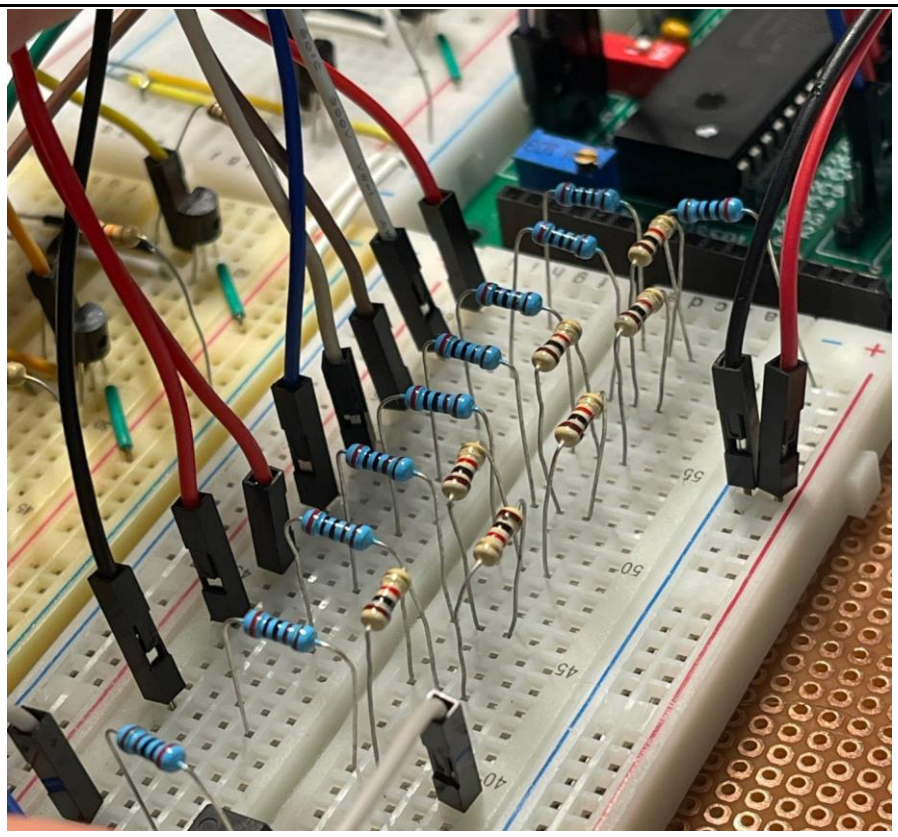
Hình 4.8: Sản phẩm đàn phím điện tử dùng cảm biến chạm tay với mạch DAC



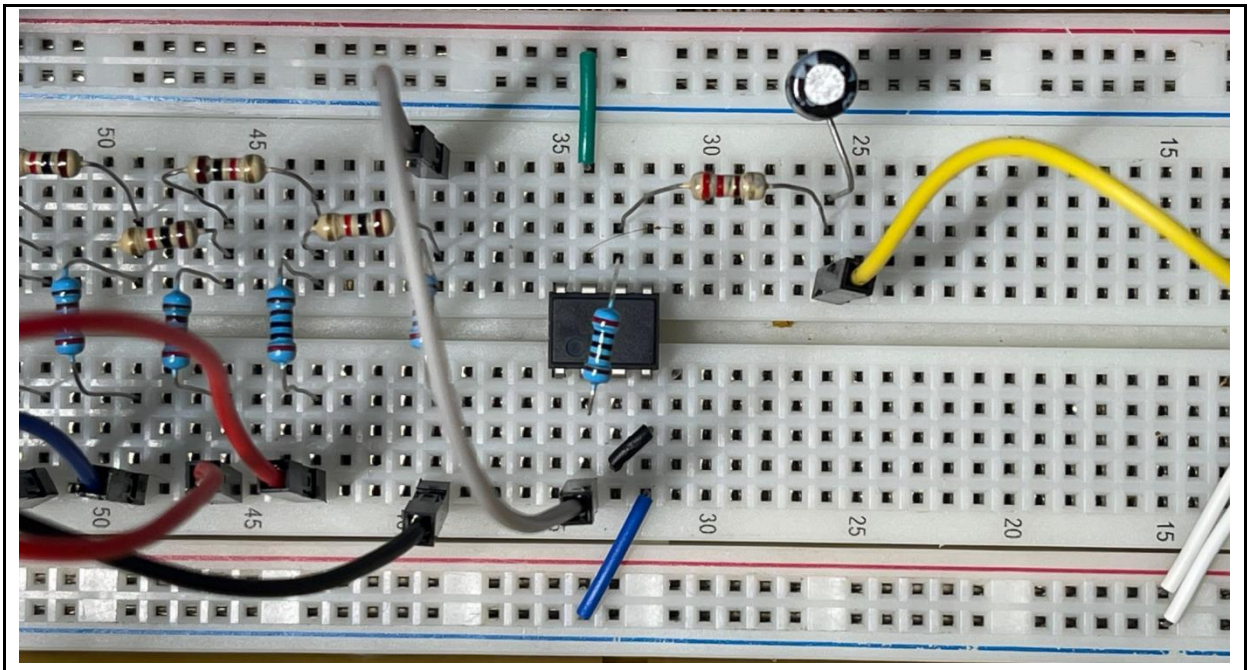
Hình 4.9: Sản phẩm được kết nối với loa thông qua cổng Audio Female 3.5mm



Hình 4.10: Cảm biến chạm tay



Hình 4.11: Các điện trở R và 2R trong mạch DAC



Hình 4.12: Mạch khuếch đại và mạch lọc thông thấp

CHƯƠNG 5: NHẬN XÉT VÀ KẾT LUẬN

5.1. Nhận xét

5.1.1. Mặt đạt được (Dimensions of Quality):

- **Performance:** Thời gian phản hồi khi phím được nhấn gần như tức thời.
- **Feature:** Thực hiện được việc thiết kế cảm biến chạm tay có độ nhạy cao và xử lý tạo ra tín hiệu dạng sine chuẩn có thể kết nối với loa phát âm thanh nốt nhạc.
- **Reliability:** Âm thanh tạo ra mượt mà, chính xác với cao độ mong muốn: độ sai lệch tần số thấp nhất là -5Hz, lớn nhất là -12Hz.
- **Durability:** Mạch hoạt động tốt trong điều kiện phòng.
- **Serviceability:** Linh kiện phần cứng dễ dàng thay thế và chi phí rẻ
- **Conformance:** Âm thanh tạo ra dựa trên cơ sở quy chuẩn quốc tế tần số của các nốt nhạc
- **Aesthetics:** Mạch được hoàn thiện tương đối gọn gàng và ổn định.

5.1.2. Mặt chưa đạt được:

Chưa có khả năng "synthesizer" (tổng hợp âm thanh) - tức là xử lý cùng lúc tạo ra 1 tín hiệu đa tần (hàm chứa nhiều tín hiệu sóng đơn tần).

5.2. Kết luận

Qua việc thực hiện đề tài này chúng em đã hiểu và biết cách làm từng bước để thực hiện một hệ thống nhúng cơ bản. Tuy nhiên với vốn kiến thức còn hạn chế và thời gian có hạn, nhóm chúng em mới hoàn thành sản phẩm ở mức độ thử nghiệm, đơn giản. Trong quá trình thực hiện dự án này, chúng em đã học tập được thêm nhiều kiến thức thực tế, trao đổi thêm giữa các thành viên, làm quen với tác phong làm việc nhóm và cách xử lý các khó khăn khi gặp phải.

Thiết kế hệ thống nhúng là một môn học hay nhưng khó, đây cũng là lần đầu tiên nhóm thực hiện một dự án như thế này, do đó bên cạnh việc tự học thì sự hướng dẫn của thầy đã giúp đỡ chúng em rất nhiều. Chúng em xin chân thành cảm ơn thầy Nguyễn Phan Hải Phú đã tận tình giúp đỡ chúng em trong thời gian qua!

TÀI LIỆU THAM KHẢO

1. “Introduction of Digital to Analog Converter and its Types”. Truy cập từ https://microcontrollerslab.com/dac-introduction-types/

2. “DAC (Digital to Analog) Conversion Using 8051 Microcontroller”. (16/07/2018). Truy cập từ https://codebloges.blogspot.com/2018/07/dacdigital-to-analog-conversion-using.html

3. “Low Pass Filter design”. Paul Thurst. (14/01/2012). Truy cập từ https://www.engineeringradio.us/blog/2012/01/low-pass-filter-design/

4. “Non-inverting Amplifier With UA741”. Truy cập từ https://www.instructables.com/Non-inverting-amplifier-with-uA741/

5. “PIC16F87XA Data Sheet”. Microchip Technology Inc. (2003) . Truy cập từ https://pdf1.alldatasheet.com/datasheet-pdf/view/82338/MICROCHIP/PIC16F877A.html