به نام خدا

دانشگاه صنعتی امیرکبیر (پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر

آزمایشگاه ریزپردازنده

گزارش‌کار آزمایش نهم

استاد درس: مهندس معصوم زاده

محمّد امین رضائی/ ۹۷۳۱۰۲۴

مهدی رحمانی/ ۹۷۳۱۷۰۱

«پیش گزارش»

پرسش اول: اسپیکر پیزوالکتریک ما چطور کار می‌کند؟ فکر می‌کنید چرا این روش کار انتخاب شده است؟

در اسپیکرهای پیزوالکتریک با اعمال ولتاژ کریستال داخل آن تغییر اندازه می‌دهد. با اعمال ولتاژ متغیر به این کریستال می‌توان نوسان‌هایی را در آن ایجاد کرد که منجر به تولید صوت می‌شود. با تغییر فرکانس تغییر ولتاژ می‌توان فرکانس‌های صوتی مختلف (که منجر به تولید نت‌های موسیقی مختلف می‌شود) را به راحتی تولید کرد. دلیل استفاده در اینجا هم همین سادگی استفاده از آن توسط آردوینو است. به این شکل که با تغییر فرکانس موج مربعی، نت‌های مختلف که از حدود ۳۲ هرتز شروع می‌شوند را می‌توان تولید کرد.

توضیحات بیشتر:

بلندگوی پیزوالکتریک یک بلندگو است که از اثر پیزوالکتریک برای تولید صدا استفاده می کند. حرکت مکانیکی اولیه با اعمال ولتاژ به یک ماده پیزوالکتریک ایجاد می شود و این حرکت معمولاً با استفاده از دیافراگم ها و تشدید کننده ها به صدای قابل شنیدن تبدیل می شود. بلندگوهای پیزوالکتریک مزایای مختلفی نسبت به بلندگوهای معمولی دارند: آنها در برابر اضافه بارهایی که به طور معمول باعث از بین رفتن بیشتر درایورهای فرکانس بالا می شوند ، مقاوم هستند و به دلیل خصوصیات الکتریکی می توان از آنها بدون کراس اوور استفاده کرد. بلندگوهای پیزوالکتریک می توانند دارای خروجی فرکانس بالا باشند و این در برخی شرایط خاص مفید است. به عنوان مثال ، برنامه های سونار که در آن انواع پیزولکتریک به عنوان دستگاه های خروجی (تولید صدای زیر آب) و هم به عنوان دستگاه های ورودی (به عنوان اجزای حسگر میکروفون های زیر آب) استفاده می شوند. آنها از مزایایی در این برنامه ها برخوردارند که حداقل ساخت ساده و جامد است که در برابر آب دریا بهتر از نوار یا مخروطی مقاومت می کند .برای این که ما قرار است انواع صدا های مختلف با طول موج های مختلف را تولید کنیم ، از این روش استفاده می شود (که ناشی از تغییر ولتاژ است.)

پرسش دوم: تایمری که دستور tone استفاده می‌کند با خیلی از پین‌های برد مشترک است. بررسی کنید که به چه روش‌هایی می‌توانید آن تایمر را به هم بریزید که دستور tone خراب شود و صداهای مطلوب را اجرا نکند.

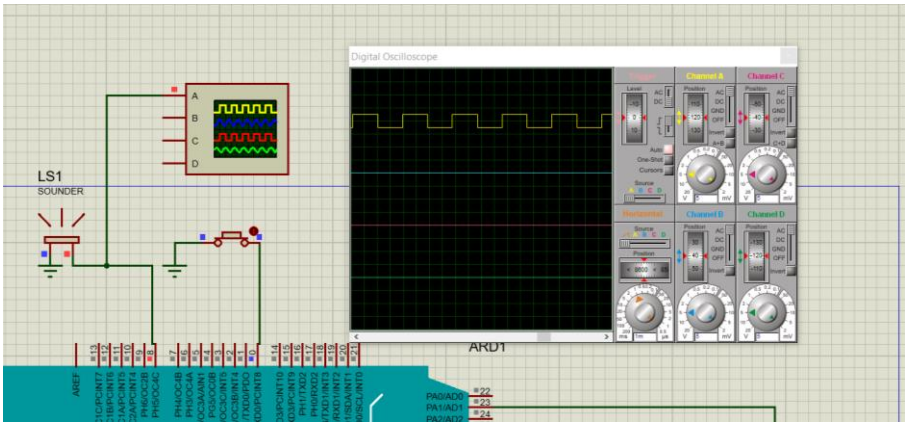
در بردهایی غیر از بردهای مگا استفاده از تابع ()tone با تایمر PWM پین‌های ۳ و ۱۱ اختلال دارد و دچار مشکل می‌شود.

درواقع دستور tone از تایمر شماره 2 استفاده می کند ، همچنین پین های 11 و 3 که PWMهستند نیز از این تایمر استفاده می کند ، و اگر حین استفاده از tone از این پین ها برای PWM استفاده کنیم ، صدا های نامطلوب اجرا می شود.

پرسش سوم: یک اسیلوسکوپ به سیم اسپیکر متصل کنید. چه اتفاقی دارد می افتد؟

با تغییر نت موسیقی و در نتیجه فرکانس صدا، می‌بینیم که فرکانس موج مربعی متصل به پیزوالکتریک هم متناسب با آن تغییر می‌کند و در واقع تغییر این فرکانس موجب تغییر صدای اسپیکر می‌شود.

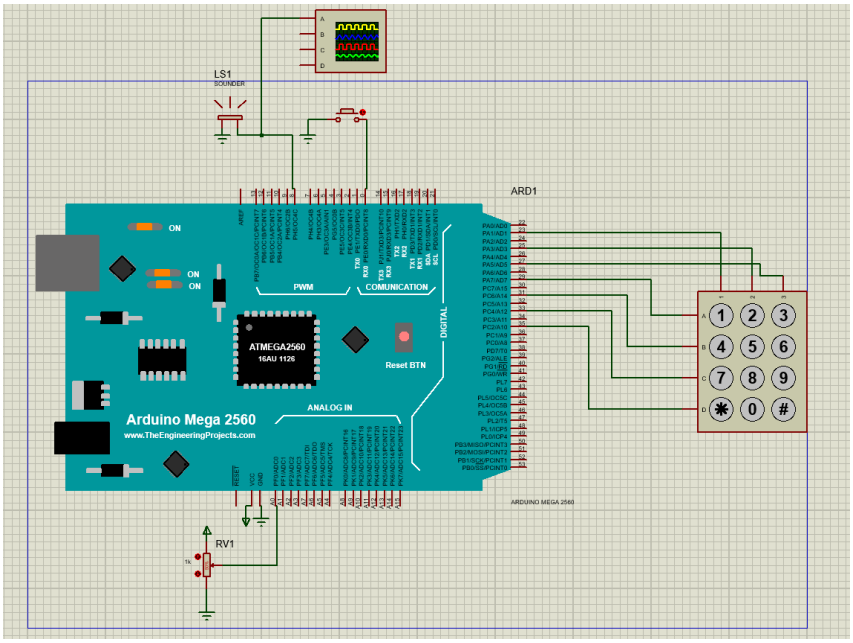
در حالت کلی طول موج سیگنال ارسالی کم و زیاد می شود ولی مقدارduty cycle ثابت می ماند ، و هر چقدر صدا زیر تر باشد طول موج کمتر است و هر چه صدا بم تر باشد ، طول موج بیشتر است.



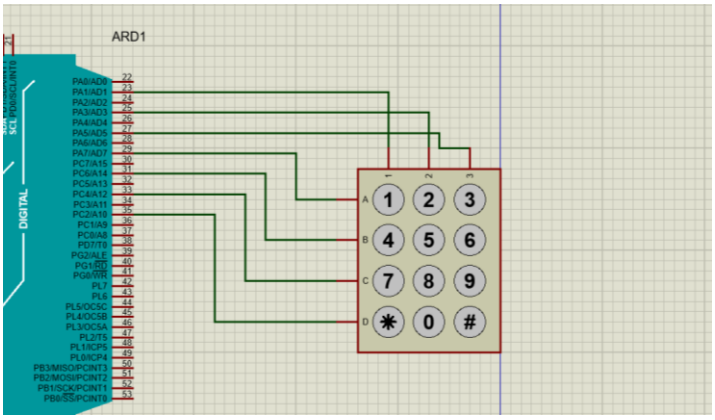
گزارش کار

مرحله اول) در پروتئوس یک دکمه و یک اسپیکر و یک صفحه کلید را به برد وصل می‌کنیم. برنامه‌ای بنویسید که به هنگام فشرده شدن هر کلید فایل ملودی متناظر با آن را پخش کند.

ابتدا مدار مورد نظر را به صورت زیر میبندیم:

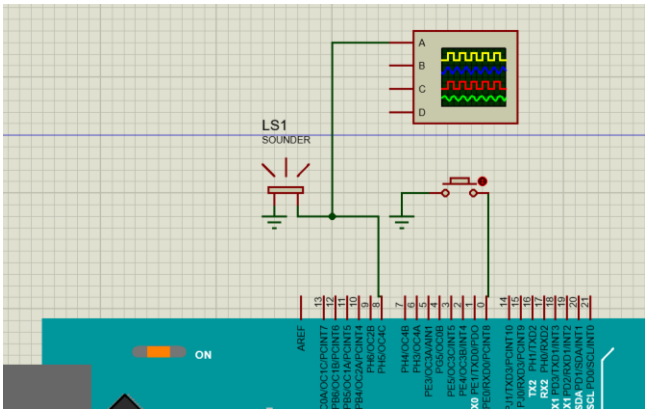


ابتدا یک صفحه کلید را به برد آردوینو وصل کرده ایم و پین های برد که به آن وصل شده اند به صورت زیر اند:



ستون های شماره 1 و 2 و 3 به ترتیب به پین های شماره 23 و 25 و 27 وصل شده اند. سطرهای A و B و C و D به ترتیب به پین‌های 29 و 31 و 33 وصل شده اند.

همچنین موارد مهم دیگر در این قسمت یک بازر یا درواقع SOUNDER میباشد که موسیقی ما را پخش کند و همان پیزو الکتریک موردنظر ماست که پیشتر راجع به آن صحبت کردیم. که یک سر آن را به پایه‌ی 8 وصل میکنیم. همچنین این پایه را به اسیلوسکوپ نیز وصل میکنیم تا ببینیم که ولتاژ ورودی آن به چه شکلی میباشد.



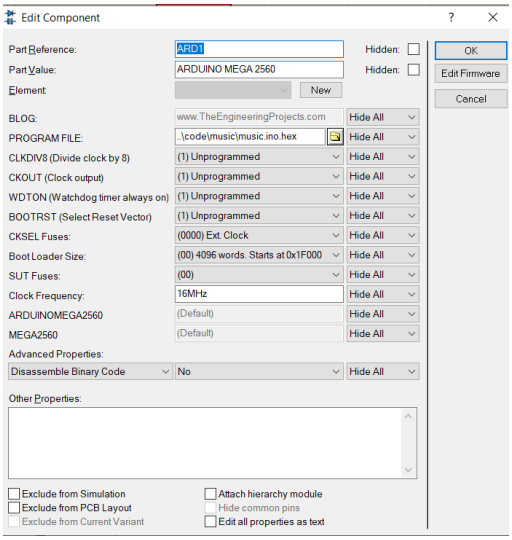
همانطور که در شکل مشخص است یک کلید یا BUTTON هم برای روشن و خاموش کردن بازر قرار میدهیم.

کلیت کار به این صورت است که 6 تا آهنگ آماده شده است و ابتدا کاربر یکی از دکمه های 1 تا 6 را فشار میدهد و بعد سپس Button را فشار میدهد و موسیقی مرتبط با آن کلید فشار داده شده پخش میشود.

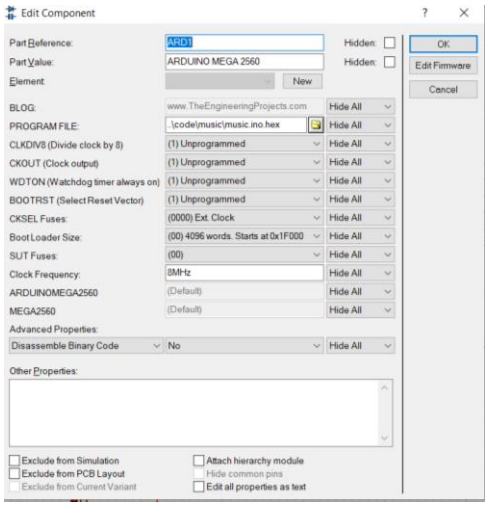
کد مربوط به این قسمت به همراه کد مربوط به مرحله‌ی سوم به صورت کامل در ادامه توضیح داده میشود.

مرحله دوم) هنگام پخش موسیقی از اسپیکر، پروتئوس با کارت صدا آن را به کامپیوتر شما می دهد. به احتمال زیاد هشدار اجرا نشدن کد در ریل تایم می‌گیرید، پس با کلیک راست روی برد، فرکانس کلاک آن را پایین بیاورید تا به جایی برسید که موسیقی به نرمی اجرا شود. متوجه می‌شوید که فرکانس نت‌ها هم عوض می شود، پس با ایجاد تغییری کوچک در کد، آن را هم رفع کنید.

در ابتدا به صورت پیشفرض کلاک آردوینو بر روی مقدار 16MHz میباشد:



سپس پس از برخورد با مشکل مقدار آن را روی 8MHz تنظیم میکنیم:



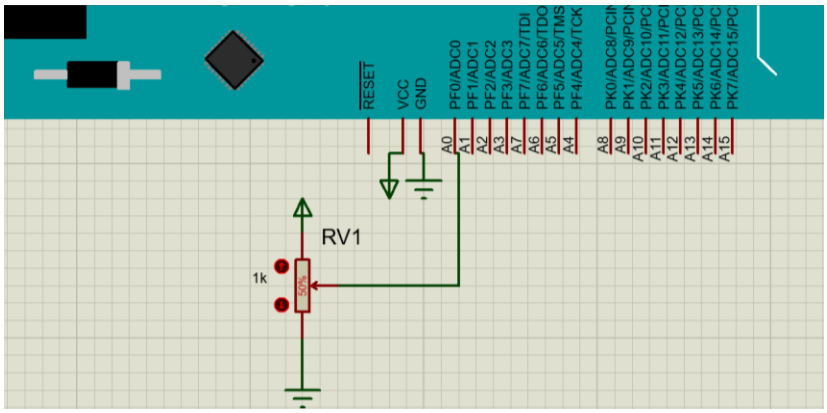
همچنین برای تنظیم بهتر در کد هم مقدار duration را تنظیم میکنیم و از متغیرهایی به نام tempo برای هریک از آهنگ ها استفاده میکنیم:

```
int duration = 2 * 60 * 1000.0 / (TEMPO1 * noteDurations1[note]);
tone(soundPin, (int) (melody1[note] * scale), duration);
delay( (int) duration );
if (digitalRead(buttonPin)) {
```

```
#define TEMPO1 210
#define TEMPO2 210
#define TEMPO3 150
#define TEMPO4 150
#define TEMPO5 135
#define TEMPO6 150
```

مرحله سوم) یک پتانسیومتر به برد وصل کنید. همان‌طور که قبلا تابع map ورودی آنالوگ را استفاده کردید، اکنون برنامه ای بنویسید که هر دفعه که دکمه فشرده می‌شود، بنا به وضعیت پتانسیومتر یک نت زیرتر یا بم‌تر اجرا کند.

همانطور که مشاهده میشود پتانسیومتر را نیز به پایه‌ی A0 وصل میکنیم:



حال به سراغ کد برنامه میرویم که به صورت کلی آن را تحلیل کنیم. این کد تمامی مراحل برنامه را پوشش میدهد:

music | Ode_to_Joy.h | Pirates_of_Caribbean.h | ey_iran.h | frere_jacques

```
#include <Keypad.h>
#include "pitches.h"

#include "ey_iran.h"
#include "jingle_bells.h"
#include "Ode_to_Joy.h"
#include "the_imperial_march.h"
#include "frere_jacques.h"
#include "Pirates_of_Caribbean.h"

#define TEMPO1 210
#define TEMPO2 210
#define TEMPO3 150
#define TEMPO4 150
#define TEMPO5 135
#define TEMPO6 150
```

```
const byte ROWS = 4; //four rows
const byte COLS = 3; //three columns

char keys[ROWS][COLS] = {
  {'1','2','3'},
  {'4','5','6'},
  {'7','8','9'},
  {'*','0','#'}
};

byte rowPins[ROWS] = {29, 31, 33, 35};
byte colPins[COLS] = {23, 25, 27};


const int potPin = A0, buttonPin = 0, soundPin = 8;
bool play = false;
float scale = 0;
int choose = 1;

Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );
```

در تصویر سمت چپ بالا ابتدا کتابخانه ی مربوط به صفحه کلید را اضافه کرده ایم. سپس فایل pitches که شامل تمامی نت ها میباشد را include کردیم و بعد از نیز یک سری فایل هدر برای هرکدام از آهنگ‌های مد نظرمان ایجاد کرده و آن ها را include کرده ایم. در نهایت هم یک سری TEMPO را define کردیم که پیش تر توضیح آن ها داده شد.

در تصویر سمت راست نیز در ابتدا متغیر های لازم برای initialize کردن کانستراکتور keypad را ایجاد و مقدار دهی کردیم و یک شی از آن درست کردیم. سپس یک سری متغیر که در ادامه از آن ها استفاده کرده ایم را تعریف کرده ایم.

```
void setup() {
  pinMode(buttonPin, INPUT_PULLUP);
}
```

همانطور که در تصویر بالا دیده میشود در قسمت setup فقط پایه مربوط به button را به صورت output تعریف کرده ایم.

```
void loop() {
  play = 1 - digitalRead(buttonPin);
  char recieve = keypad.getKey();
  if (recieve)
  {
    switch(recieve){
      case '1':
        choose = 1;
        break;
      case '2':
        choose = 2;
        break;
      case '3':
        choose = 3;
        break;
      case '4':
        choose = 4;
        break;
      case '5':
        choose = 5;
        break;
      case '6':
        choose = 6;
        break;
    }
  }
```

سپس در قسمت لوپ با توجه به اینکه کاربر چه عددی را در صفحه کلید فشار میدهد یک متغیر choose را مقدار دهی میکنیم تا براساس مقدار آن زمانی که کاربر دکمه play را فشار داد آهنگ متناسب با آن پخش شود.

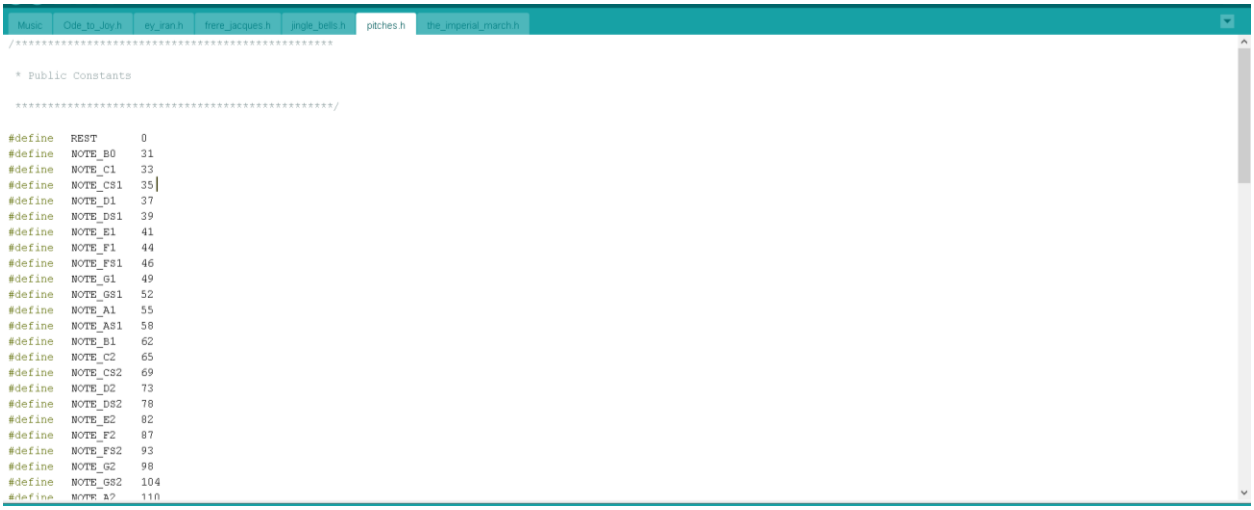
```
if(play && choose == 1)
{
  for (int note = 0; note < sizeof(melody1) / sizeof(int); note++) {
    scale = analogRead(potPin) / 512.0;
    int duration = 2 * 60 * 1000.0 / (TEMPO1 * noteDurations1[note]);
    tone(soundPin, (int) (melody1[note] * scale), duration);
    delay( (int) duration );
    if (digitalRead(buttonPin)) {
      break;
    }
  }
}else if(play && choose == 2){
  for (int note = 0; note < sizeof(melody2) / sizeof(int); note++) {
    scale = analogRead(potPin) / 512.0;
    int duration = 2 * 60 * 1000.0 / (TEMPO2 * noteDurations2[note]);
    tone(soundPin, (int) (melody2[note] * scale), duration);
    delay( (int) duration );
    if (digitalRead(buttonPin)) {
      break;
    }
  }
```

```
}else if(play && choose == 3){
  for (int note = 0; note < sizeof(melody3) / sizeof(int); note++) {
    scale = analogRead(potPin) / 512.0;
    int duration = 2 * 60 * 1000.0 / (TEMPO3 * noteDurations3[note]);
    tone(soundPin, (int) (melody3[note] * scale), duration);
    delay( (int) duration );
    if (digitalRead(buttonPin)) {
      break;
    }
  }
}else if(play && choose == 4){
  for (int note = 0; note < sizeof(melody4) / sizeof(int); note++) {
    scale = analogRead(potPin) / 512.0;
    int duration = 2 * 60 * 1000.0 / (TEMPO4 * noteDurations4[note]);
    tone(soundPin, (int) (melody4[note] * scale), duration);
    delay( (int) duration );
    if (digitalRead(buttonPin)) {
      break;
    }
  }
}else if(play && choose == 5){
  for (int note = 0; note < sizeof(melody5) / sizeof(int); note++) {
    scale = analogRead(potPin) / 512.0;
    int duration = 2 * 60 * 1000.0 / (TEMPO5 * noteDurations5[note]);
    tone(soundPin, (int) (melody5[note] * scale), duration);
    delay( (int) duration );
    if (digitalRead(buttonPin)) {
      break;
```

```
}else if(play && choose == 6){
  for (int note = 0; note < sizeof(melody6) / sizeof(int); note++) {
    scale = analogRead(potPin) / 512.0;
    int duration = 2 * 60 * 1000.0 / (TEMPO6 * noteDurations6[note]);
    tone(soundPin, (int) (melody6[note] * scale), duration);
    delay( (int) duration );
    if (digitalRead(buttonPin)) {
      break;
    }
  }
}
```

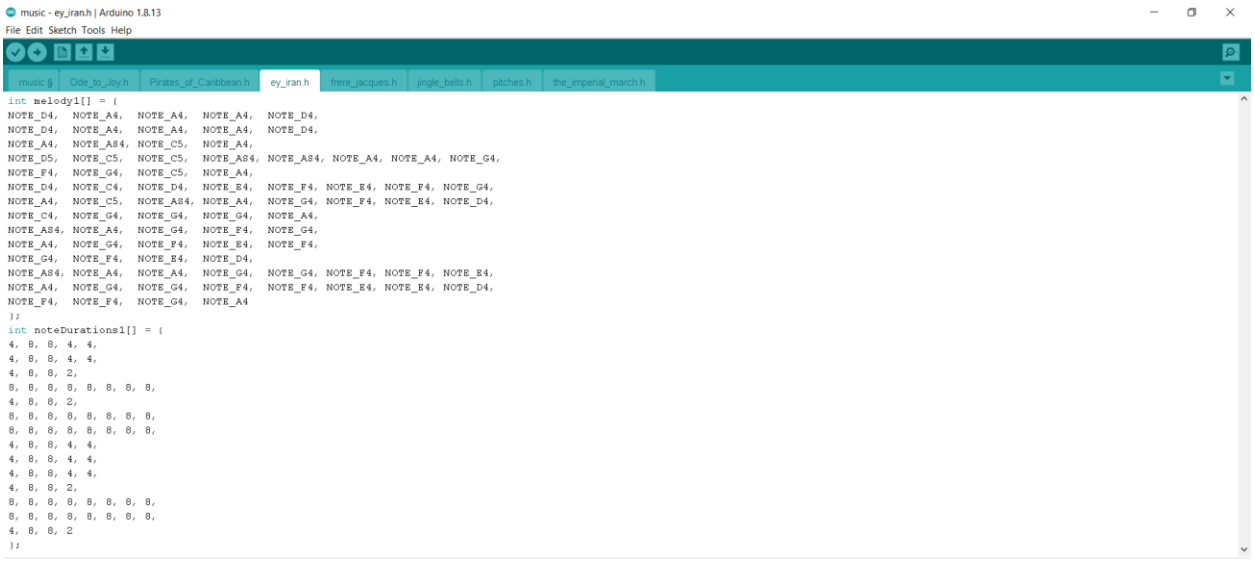
همانطور که در کد های بالا دیده میشود بعد از آنکه مقدار متغیر choose در مرحله قبل تعیین شد سپس به کمک مقدار آن و اینکه آیا دکمه play فشار داده شده است یا نه وارد یکی از دستورات شرطی بالا شده و آهنگ متناظر با آن دکمه ی فشار داده شده را پخش میکنیم.

در زیر قسمتی از عکس pitches که نت ها در آن تعریف شده اند آمده است:



در ادامه عکس هایی مربوط به ملودی ها و duration های آن ها که در فایل های هدر است را میبنید:

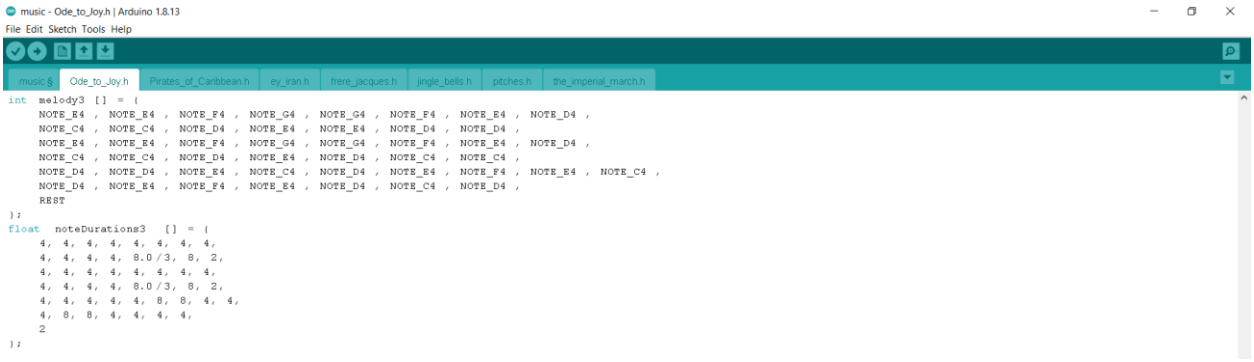
فایل هدر مربوط به آهنگ ای ایران:



فایل هدر مربوط به آهنگ jingle bells :

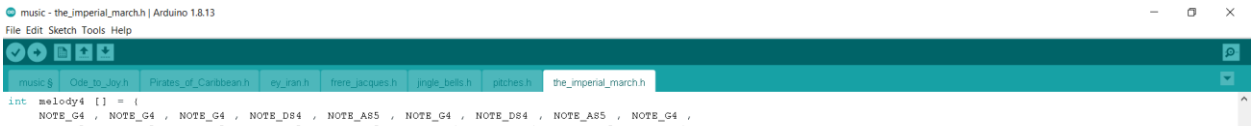

```
int  melody2 [] = {
     NOTE_E4 , NOTE_E4 , NOTE_E4 ,
     NOTE_E4 , NOTE_E4 , NOTE_E4 ,
     NOTE_E4 , NOTE_G4 , NOTE_C4 , NOTE_D4 ,
     NOTE_E4 ,
     NOTE_F4 , NOTE_F4 , NOTE_F4 , NOTE_F4 ,
     NOTE_F4 , NOTE_E4 , NOTE_E4 , NOTE_E4 , NOTE_E4 ,
     NOTE_E4 , NOTE_D4 , NOTE_D4 , NOTE_E4 ,
     NOTE_D4 , NOTE_G4
};
float  noteDurations2   [] = {
     4, 4, 2,
     4, 4, 2,
     4, 4, 8.0 / 3, 8,
     1,
     4, 4, 8.0 / 3, 8,
     4, 4, 4, 8, 8,
     4, 4, 4, 4,
     2, 2
};
```

فایل هدر مربوط به آهنگ Ode to Joy:




```
int  melody3 [] = {
     NOTE_E4 , NOTE_E4 , NOTE_F4 , NOTE_G4 , NOTE_G4 , NOTE_F4 , NOTE_E4 , NOTE_D4 ,
     NOTE_C4 , NOTE_C4 , NOTE_D4 , NOTE_E4 , NOTE_E4 , NOTE_D4 , NOTE_D4 ,
     NOTE_E4 , NOTE_E4 , NOTE_F4 , NOTE_G4 , NOTE_G4 , NOTE_F4 , NOTE_E4 , NOTE_D4 ,
     NOTE_C4 , NOTE_C4 , NOTE_D4 , NOTE_E4 , NOTE_D4 , NOTE_C4 , NOTE_C4 ,
     NOTE_D4 , NOTE_D4 , NOTE_E4 , NOTE_C4 , NOTE_D4 , NOTE_E4 , NOTE_F4 , NOTE_E4 , NOTE_C4 ,
     NOTE_D4 , NOTE_E4 , NOTE_F4 , NOTE_E4 , NOTE_D4 , NOTE_C4 , NOTE_D4 ,
     REST
};
float  noteDurations3   [] = {
     4, 4, 4, 4, 4, 4, 4, 4,
     4, 4, 4, 4, 8.0/3, 8, 2,
     4, 4, 4, 4, 4, 4, 4, 4,
     4, 4, 4, 4, 8.0/3, 8, 2,
     4, 4, 4, 4, 4, 8, 8, 4, 4,
     4, 8, 8, 4, 4, 4, 4,
     2
};
```

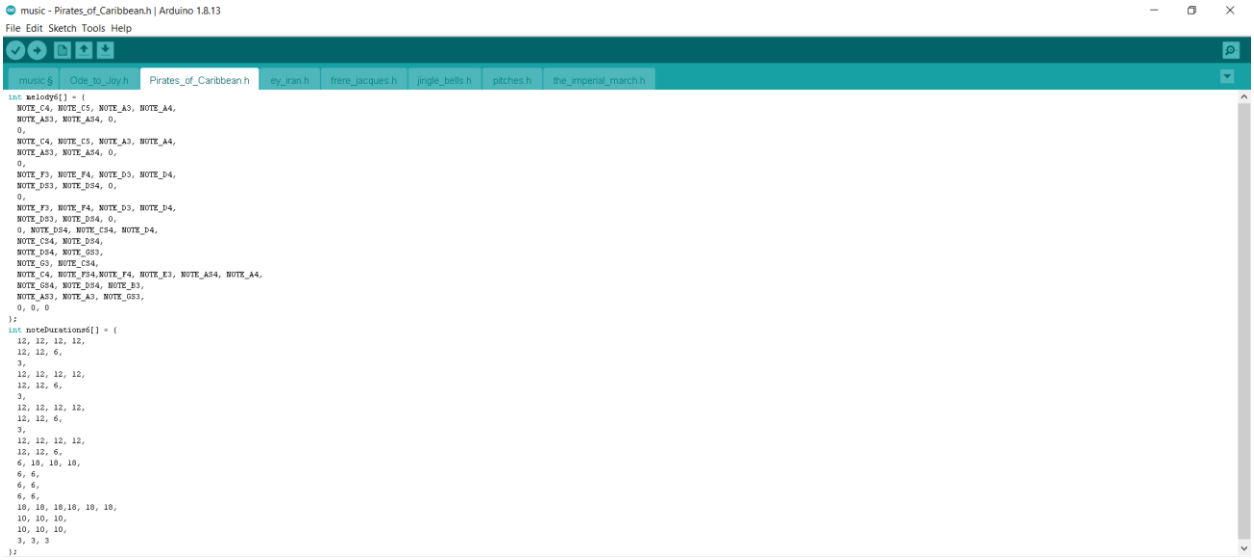
فایل هدر مربوط به آهنگ the imperial march :




```
int  melody4 [] = {
     NOTE_G4 , NOTE_G4 , NOTE_G4 , NOTE_DS4 , NOTE_AS5 , NOTE_G4 , NOTE_DS4 , NOTE_AS5 , NOTE_G4 ,
     NOTE_D5 , NOTE_D5 , NOTE_D5 , NOTE_DS5 , NOTE_AS5 , NOTE_FS4 , NOTE_DS4 , NOTE_AS5 , NOTE_G4 ,
     NOTE_G5 , NOTE_G4 , NOTE_G4 , NOTE_G5 , NOTE_FS5 , NOTE_F5 , NOTE_E5 , NOTE_DS5 , NOTE_DS5 , REST , NOTE_GS4 , NOTE_CS5 , NOTE_C5 , NOTE_B5 ,
     NOTE_AS5 , NOTE_A5 , NOTE_AS5 , REST , NOTE_DS4 , NOTE_FS4 , NOTE_DS4 , NOTE_FS4 , NOTE_AS5 , NOTE_G4 , NOTE_AS5 , NOTE_D5
};
float  noteDurations4   [] = {
     4, 4, 4, 16.0/3, 16, 4, 16.0/3, 16, 2,
     4, 4, 4, 16.0/3, 16, 4, 16.0/3, 16, 2,
     4, 16.0/3, 16, 4, 16.0/3, 16, 16, 16, 8, 8, 8, 4, 16.0/3, 16,
     16, 16, 8, 8, 8, 4, 16.0/3, 16, 4, 16.0/3, 16, 2
};
```

فایل هدر مربوط به آهنگ frere jacques:

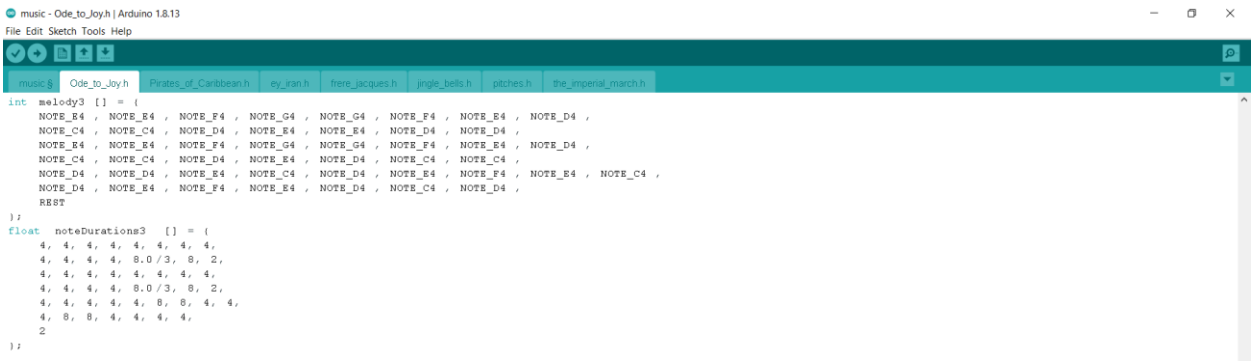
فایل هدر مربوط به آهنگ Pirates of Caribbean:



مرحله چهارم) با دانشی که دارید، برنامه ای بنویسید که ملودی Ode to Joy بتهوون را پخش کند.

همانطور که در قسمت قبل توضیح داده شد کد مربوط به پخش این آهنگ در کد اصلی آمده است. هدر مربوط به آن هم به صورت زیر است:

فایل هدر مربوط به آهنگ Ode to Joy:



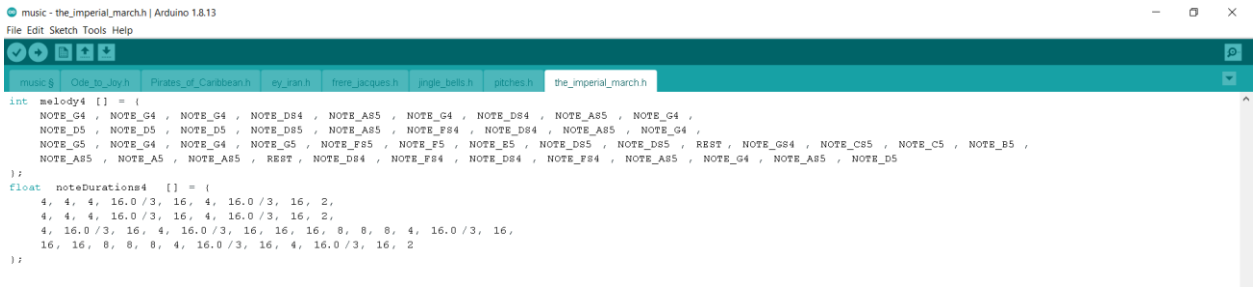
که چون سومی هست در این قسمت کد اصلی کد مربوط به پخش آن قرار گرفته است:

```
,
}else if(play && choose == 3){
  for (int note = 0; note < sizeof(melody3) / sizeof(int); note++) {
    scale = analogRead(potPin) / 512.0;
    int duration = 2 * 60 * 1000.0 / (TEMPO3 * noteDurations3[note]);
    tone(soundPin, (int) (melody3[note] * scale), duration);
    delay( (int) duration );
    if (digitalRead(buttonPin)) {
      break;
    }
  }
```

مرحله پنجم) ملودی امتیازی: (♭ همان # است ولی بجای یک قدم جلو، یک قدم عقب. ♮ نت را عادی می کند. ♭ های اول خط تا آخر خط باقی می‌مانند، مگر اینکه با ♮ عادی سازی شوند)

این کد نیز هم هدرش و هم دستور پخشش در کد اصلی قرار گرفت است:

فایل هدر مربوط به آهنگ the imperial march :




```
int  melody4 [] = {
     NOTE_G4 , NOTE_G4 , NOTE_G4 , NOTE_DS4 , NOTE_AS5 , NOTE_G4 , NOTE_DS4 , NOTE_AS5 , NOTE_G4 ,
     NOTE_D5 , NOTE_D5 , NOTE_D5 , NOTE_DS5 , NOTE_AS5 , NOTE_FS4 , NOTE_DS4 , NOTE_AS5 , NOTE_G4 ,
     NOTE_G5 , NOTE_G4 , NOTE_G4 , NOTE_G5 , NOTE_FS5 , NOTE_F5 , NOTE_E5 , NOTE_DS5 , NOTE_DS5 , REST , NOTE_GS4 , NOTE_CS5 , NOTE_C5 , NOTE_B5 ,
     NOTE_AS5 , NOTE_A5 , NOTE_AS5 , REST , NOTE_DS4 , NOTE_FS4 , NOTE_DS4 , NOTE_FS4 , NOTE_AS5 , NOTE_G4 , NOTE_AS5 , NOTE_D5
};
float  noteDurations4  [] = {
     4, 4, 4, 16.0/3, 16, 4, 16.0/3, 16, 2,
     4, 4, 4, 16.0/3, 16, 4, 16.0/3, 16, 2,
     4, 16.0/3, 16, 4, 16.0/3, 16, 16, 16, 8, 8, 8, 4, 16.0/3, 16,
     16, 16, 8, 8, 8, 4, 16.0/3, 16, 4, 16.0/3, 16, 2
};
```

که چون چهارمی هست در این قسمت کد اصلی کد مربوط به پخش آن قرار گرفته است:

```
  }
}else if(play && choose == 4){
  for (int note = 0; note < sizeof(melody4) / sizeof(int); note++) {
    scale = analogRead(potPin) / 512.0;
    int duration = 2 * 60 * 1000.0 / (TEMPO4 * noteDurations4[note]);
    tone(soundPin, (int) (melody4[note] * scale), duration);
    delay( (int) duration );
    if (digitalRead(buttonPin)) {
      break;
    }
  }
```