Arduino Based Piano and Song Player

Yatin Vadehra 20/10/2022

Abstract

This project displays the audio-programming capabilities of the Arduino microcontroller and Arduino IDE, which have been paired with resistors, push buttons, and a buzzer to make a piano-fashioned sound. After pressing a push button, the buzzer plays a pre-coded user-desired song. The programmer can set his/her desired notes to play by using the pitches.h library in Arduino IDE.

The song I chose for my project is “Unravel” from the Japanese Anime Series “Tokyo Ghoul”. There is no limitation to what song the programmer wants the buzzer to output. However, the project is only capable of playing the lead melody notes. In order to play the different voices and ornaments present in a typical song, one can use multiple buzzers and resistors- push buttons arrangements.

Introduction

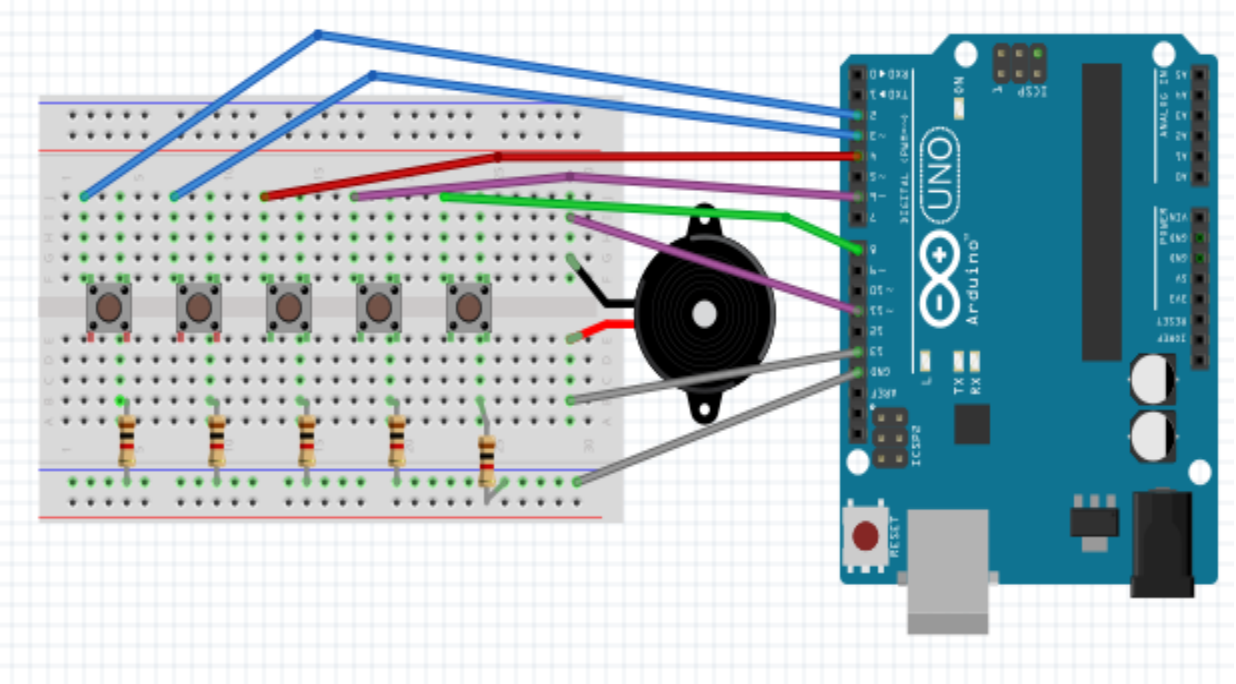
Many of us are fans of music. For some it’s a source of pleasure, for some it gives them motivation, for others, it acts as a friend in times of stress. No matter what the reason may be, it is a universal fact that music is an essential part of every human’s life and is always with us no matter where we go. It is rightly called the “Universal language”.

There are many different musical instruments. However, the most famous and well-recognized is the Piano. A Piano uses mechanical and acoustic actions to produce sounds. The sounds produced vary in pitches and frequency, called as “Octaves” and “Musical scales” in Music theory. Each pitch has a certain note value assigned to it. The musical convention defines what name a pitch gets.

An octave has 8 pitches and there are a total of 7 octaves in music theory. The notes are represented by the English alphabet A to G.

These notes can be defined in the Arduino IDE using the define function or by including the pitches.h library. This allows the user to assign the notes to any hardware and hence output the audio

Schematic diagram



Working and Explanation

Firstly, attach the buttons to the breadboard. One pair of legs of the button extending towards the + rail of the breadboard and the other pair of legs extending towards the - rail of the breadboard.

Then, Connect a resistor (1k Ohm) each per button. Connect one leg of the resistors towards the blue line of the breadboard in extension of a leg of a button. Connect the other leg of every resistor to the - side of the power rail.

Attach the buzzer in the 5th mounting hole next to the last button (hole 29). One side of the buzzer attached towards the + rail of the breadboard and the other to the - side.

Attach a jumper cable to a button each. Connect it to the other leg of the button on towards the + side of the breadboard. Attach 2 jumper cables to the buzzer: one towards the + side and one towards the -, both in row 29. Connect the last jumper cable to the - side of the power rail like the resistors.

Connect the jumper cables of the buttons to the pins 2, 3, 4, 6, 8 from left to right as shown in the picture with the arduino and connect the jumper cable of the buzzer pointing towards the + and - both to pin 11 and 13 respectively.

Lastly, connect the jumper cable from the - power rail to GND so you have a closed electric circuit.

Working and Explanation of the Code

We first use the pitches.h tab. It includes pre defined musical notes and their pitches.

The pitches.h tab contains over 8 octaves of notes, which will give you enough variety to play around with. After that comes naming what button is called and is using which pin and the void setup. The void loop contains the code which note will be played when you press the button. The note will play for as long as you press the button. For the melody you can see how the sequence of the notes work, no explanations needed except for if you want a pause/silence in your song, the 'note' is 0, while the noteduration is not. You'll have include in the code how long the pause is the same way as a normal note.

You can change the tone of a button in the loop to whatever note you'd like. This way you can change the note intervals between them and change it to your liking. Don't forget to also change the code at the beginning if you change the name of the button accordingly to the note.

The song in the code is Unravel, but you can put a song of your own in it by changing the notes. You can look up the notes using piano sheets and change the code along with the note durations accordingly to the song you want the arduino to play. The duration of a note in this code is 600ms/ the note duration you used as stated in the code in the void loop. You can change the speed of the song by changing this. E.g. change 600 to 1000 to make it play slower or to 500 to make it play faster.

The code used is given below-

```
#define NOTE_B0 31
#define NOTE_C1 33
#define NOTE_CS1 35
#define NOTE_D1 37
#define NOTE_DS1 39
#define NOTE_E1 41
#define NOTE_F1 44
#define NOTE_FS1 46
#define NOTE_G1 49
#define NOTE_GS1 52
#define NOTE_A1 55
#define NOTE_AS1 58
#define NOTE_B1 62
#define NOTE_C2 65
#define NOTE_CS2 69
#define NOTE_D2 73
#define NOTE_DS2 78
#define NOTE_E2 82
#define NOTE_F2 87
#define NOTE_FS2 93
#define NOTE_G2 98
#define NOTE_GS2 104
#define NOTE_A2 110
#define NOTE_AS2 117
```

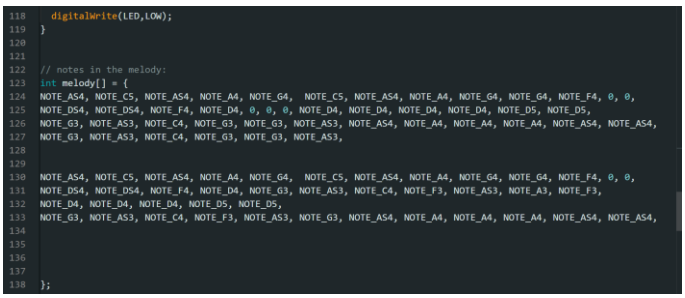
```
#define NOTE_B2 123
#define NOTE_C3 131
#define NOTE_CS3 139
#define NOTE_D3 147
#define NOTE_DS3 156
#define NOTE_E3 165
#define NOTE_F3 175
#define NOTE_FS3 185
#define NOTE_G3 196
#define NOTE_GS3 208
#define NOTE_A3 220
#define NOTE_AS3 233
#define NOTE_B3 247
#define NOTE_C4 262
#define NOTE_CS4 277
#define NOTE_D4 294
#define NOTE_DS4 311
#define NOTE_E4 330
#define NOTE_F4 349
#define NOTE_FS4 370
#define NOTE_G4 392
#define NOTE_GS4 415
#define NOTE_A4 440
#define NOTE_AS4 466
```

```
#define NOTE_B4 494
#define NOTE_C5 523
#define NOTE_CS5 554
#define NOTE_D5 587
#define NOTE_DS5 622
#define NOTE_E5 659
#define NOTE_F5 698
#define NOTE_FS5 740
#define NOTE_G5 784
#define NOTE_GS5 831
#define NOTE_A5 880
#define NOTE_AS5 932
#define NOTE_B5 988
#define NOTE_C6 1047
#define NOTE_CS6 1109
#define NOTE_D6 1175
#define NOTE_DS6 1245
#define NOTE_E6 1319
#define NOTE_F6 1397
#define NOTE_FS6 1480
#define NOTE_G6 1568
#define NOTE_GS6 1661
#define NOTE_A6 1760
#define NOTE_AS6 1865
```

```
#define NOTE_B6 1976
#define NOTE_C7 2093
#define NOTE_CS7 2217
#define NOTE_D7 2349
```

```
#define NOTE_DS7 2489
#define NOTE_E7 2637
#define NOTE_F7 2794
#define NOTE_FS7 2960
#define NOTE_G7 3136
#define NOTE_GS7 3322
#define NOTE_A7 3520
#define NOTE_AS7 3729
#define NOTE_B7 3951
#define NOTE_C8 4186
#define NOTE_CS8 4435
#define NOTE_D8 4699
#define NOTE_DS8 4978
#define ACTIVATED LOW
```

```
const int PIEZO = 11;
const int LED = 13;

int buttonSong = 8;
const int BUTTON_C = 6;
const int BUTTON_AS = 4;
const int BUTTON_A = 3;
const int BUTTON_G = 2;


void setup()
{
  Serial.begin(9600);
  pinMode(LED, OUTPUT);
  pinMode(BUTTON_C, INPUT);
  digitalWrite(BUTTON_C,HIGH);
  pinMode(BUTTON_AS, INPUT);
  digitalWrite(BUTTON_AS,HIGH);
  pinMode(BUTTON_A, INPUT);
  digitalWrite(BUTTON_A,HIGH);
  pinMode(BUTTON_G, INPUT);
  digitalWrite(BUTTON_G,HIGH);
  pinMode (buttonSong, INPUT);
  digitalWrite(buttonSong, HIGH);
```



```
  digitalWrite(LED,LOW);
}


// notes in the melody:
int melody[] = {
NOTE_AS4, NOTE_C5, NOTE_AS4, NOTE_A4, NOTE_G4,  NOTE_C5, NOTE_AS4, NOTE_A4, NOTE_G4, NOTE_G4, NOTE_F4, 0, 0,
NOTE_DS4, NOTE_DS4, NOTE_F4, NOTE_D4, 0, 0, 0, NOTE_D4, NOTE_D4, NOTE_D4, NOTE_D4, NOTE_D5, NOTE_D5,
NOTE_G3, NOTE_AS3, NOTE_C4, NOTE_G3, NOTE_G3, NOTE_AS3, NOTE_AS4, NOTE_A4, NOTE_A4, NOTE_A4, NOTE_AS4, NOTE_AS4,
NOTE_G3, NOTE_AS3, NOTE_C4, NOTE_G3, NOTE_G3, NOTE_AS3,


NOTE_AS4, NOTE_C5, NOTE_AS4, NOTE_A4, NOTE_G4,  NOTE_C5, NOTE_AS4, NOTE_A4, NOTE_G4, NOTE_G4, NOTE_F4, 0, 0,
NOTE_DS4, NOTE_DS4, NOTE_F4, NOTE_D4, NOTE_G3, NOTE_AS3, NOTE_C4, NOTE_F3, NOTE_AS3, NOTE_A3, NOTE_F3,
NOTE_D4, NOTE_D4, NOTE_D4, NOTE_D5, NOTE_D5,
NOTE_G3, NOTE_AS3, NOTE_C4, NOTE_F3, NOTE_AS3, NOTE_G3, NOTE_AS4, NOTE_A4, NOTE_A4, NOTE_A4, NOTE_AS4, NOTE_AS4,



};
```

```
// note durations: 4 = quarter note, 8 = eighth note, etc.:
int noteDurations[] = {
```

```
4.5, 2.25, 2.25, 4.5, 2.25, 2.25, 2.25, 2.25, 2.25, 4.5, 3, 9,
4.5,
4.5, 2.25, 4.5, 2.25, 1.125, 2.25, 4.5, 4.5, 2.25, 4.5, 2.25,
4.5, 2.25,
4.5, 4.5, 4.5, 2.25, 4.5, 4.5, 4.5, 2.25, 4.5, 2.25, 4.5, 2,
4.5, 4.5, 4.5, 2.25, 4.5, 4.5,

4.5, 2.25, 2.25, 4.5, 2.25, 2.25, 2.25, 2.25, 2.25, 4.5, 3, 9,
4.5,
4.5, 2.25, 4.5, 4.5, 4.5, 4.5, 4.5, 4.5, 3, 3, 4.5,
2.25, 4.5, 2.25, 4.5, 2.25,
4.5, 4.5, 4.5, 4.5, 2.25, 4.5, 4.5, 2.25, 4.5, 2.25, 4.5, 2.25,




} ;


void loop()
{
   while(digitalRead(BUTTON_C) == ACTIVATED)
  {
    tone(PIEZO,NOTE_C5);
    digitalWrite(LED,HIGH);
  }
  while(digitalRead(BUTTON_AS) == ACTIVATED)
  {
    tone(PIEZO,NOTE_AS4);
    digitalWrite(LED,HIGH);
  }

  while(digitalRead(BUTTON_A) == ACTIVATED)
```

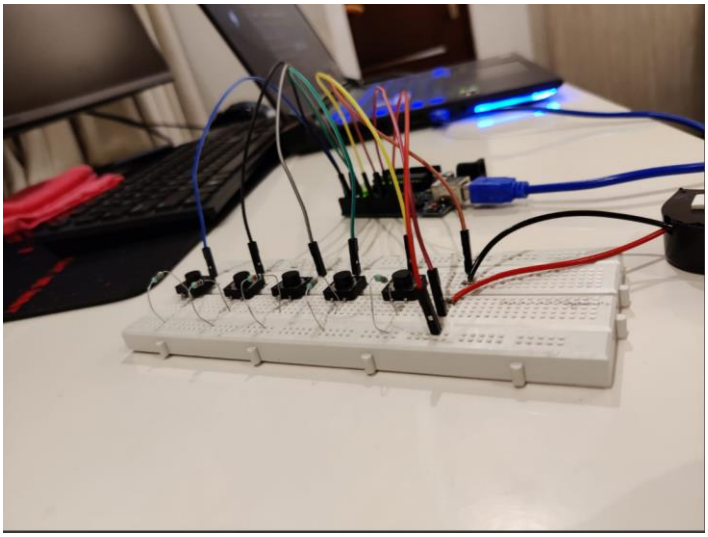
```
  {
    tone(PIEZO,NOTE_A4);
    digitalWrite(LED,HIGH);
  }

  while(digitalRead(BUTTON_G) == ACTIVATED)
  {
    tone(PIEZO,NOTE_G4);
    digitalWrite(LED,HIGH);
  }

  if(digitalRead(buttonSong) == ACTIVATED) {
    for (int thisNote=0; thisNote <85; thisNote++) {
      int noteDuration = 600 / noteDurations[thisNote];
      tone(11, melody[thisNote], noteDuration);
      int pauseBetweenNotes = noteDuration * 1.50;
      delay(pauseBetweenNotes);
      noTone(11);
    }

  }

  noTone(PIEZO);
  digitalWrite(LED,LOW);

}
```

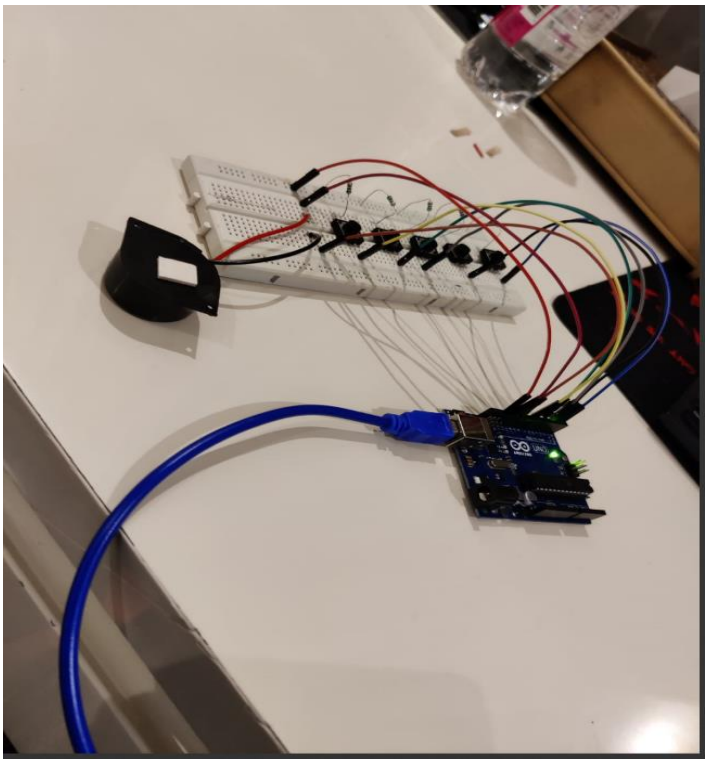
Hardware and software used

1) Arduino uno Microcontroller
2) Arduino IDE
3) Buzzer (5V)
4) Pushbuttons
5) Jumper wires
6) Resistors (1Kohm)

7) Breadboard

Working model screenshots





References

https://create.arduino.cc

https://www.arduino.cc/

https://github.com