HandsOn Music

Sabrina Moheydeen (sabrina7)

Elisabeth Martin (emm7)

ECE 110/120 Honors Project

HandsOn Music

## Introduction

### *Problem Description*

Our mutual interest in music and musical instruments motivated us to work on solving the problem of the cost, size, and complexity of musical instruments. We believe it is important for people to have easy access to music without needing to have large amounts of money or space; in this way, more people will be able to express themselves through music. Our design, HandsOn Music, must meet the constraints of portability, low cost, and ease of entrance for beginners.

### *Design Concept*

Our project is a modification of several other music projects that take inputs on a surface that represents a keyboard to play music; however, our project incorporates both the movement and keying combination and a teaching and learning mode that can teach beginners how to read music and translate it to finger motions. Our system, HandsOn Music, incorporates gloves and force sensitive resistors to play music according to finger movements. We also incorporate LEDs in a tutorial mode to allow users to learn piano fingerings if they have limited experience with the instrument, or provide a flashing display of color when not in tutorial mode (hereafter referred to as free play mode). Our system receives inputs from the pressure on the force sensitive resistors (FSRs) and push buttons. The push buttons interface with NAND gates to determine if our system is in tutorial mode. If in tutorial mode, the Arduino signals on an LED

which finger is to be played, and the FSRs send signals to the analog inputs of the Arduino to determine which note is played. When in free play mode, the FSRs send an input to the Arduino which signals an LED to light up to display the note played. We use a speaker to play the frequencies of various notes based on the user's motions.

**Analysis of Components**

*Characterization of Each Sensor*

To create our project, we used force sensitive resistors to detect when a key was played in order to play the corresponding note in the free play mode or determine whether the correct note has been played in the tutorial mode. The output values of these sensors increase when a force is applied and they are pressed. To use these sensors in the Arduino code, we needed to find the values of the sensors when they were pressed. Although we initially found that the sensors output a value of 0 when not pressed and greater than 15 when pressed, after adding components to our project, these values changed, and as a result, we needed to use a value of 300 or greater as the value to detect an applied force.

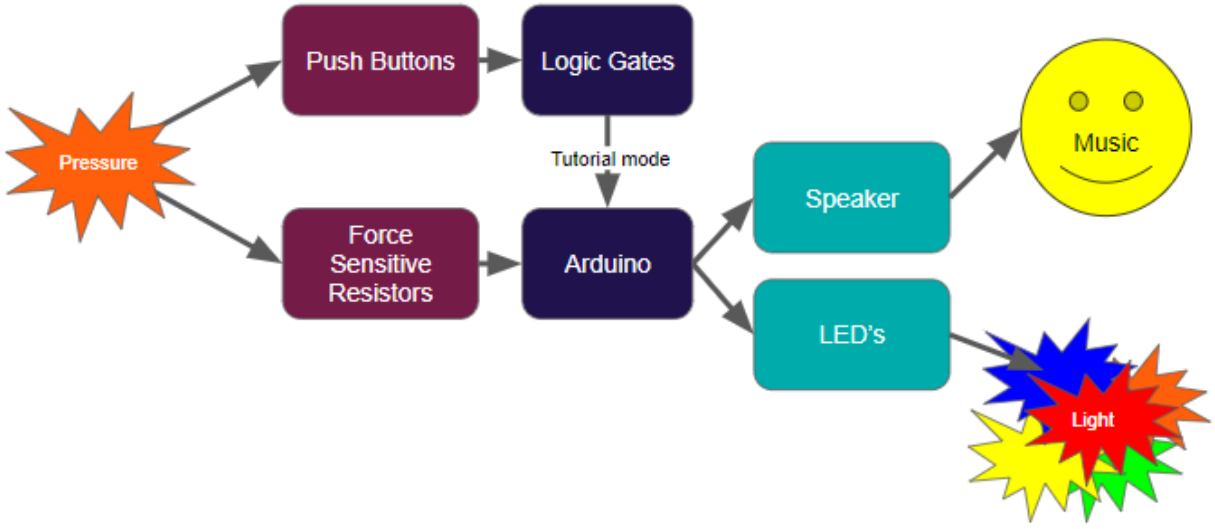
We also used NAND gates in our project to switch into the tutorial mode. We initially wanted to use two 7400 chips to turn on the tutorial when all five force sensors were pressed by NANDing two sensors together and NANDing the output with itself, doing the same with two other sensors, NANDing these two outputs together and inverting the output and NANDing this output with the final sensor and inverting this output. However, we ran into several issues while doing this and instead used two push buttons NANDed together and NANDed this output to send into the Arduino so the tutorial mode would turn on when the buttons were pressed at the same time.

*Design Considerations*

Before incorporating the FSR's into our circuit, we experimented with them using an ohmmeter and the Arduino readout. We determined that their resistances went extremely high once their threshold was met. We determined the threshold of "off" vs. "on" as we needed for our project by using the Arduino readout; we determined the force threshold value for our FSR's was around 300. We incorporated this value into our threshold for when the FSR's would register as pressed ("on") and not pressed ("off").

## Design Description

*Block Diagram*




Inputs:

The input to our design was pressure to either the push buttons or the FSR's. This allowed our further logic gates and microcontroller to know whether the user had pressed a finger to the surface or whether the user wanted to enter tutorial mode.

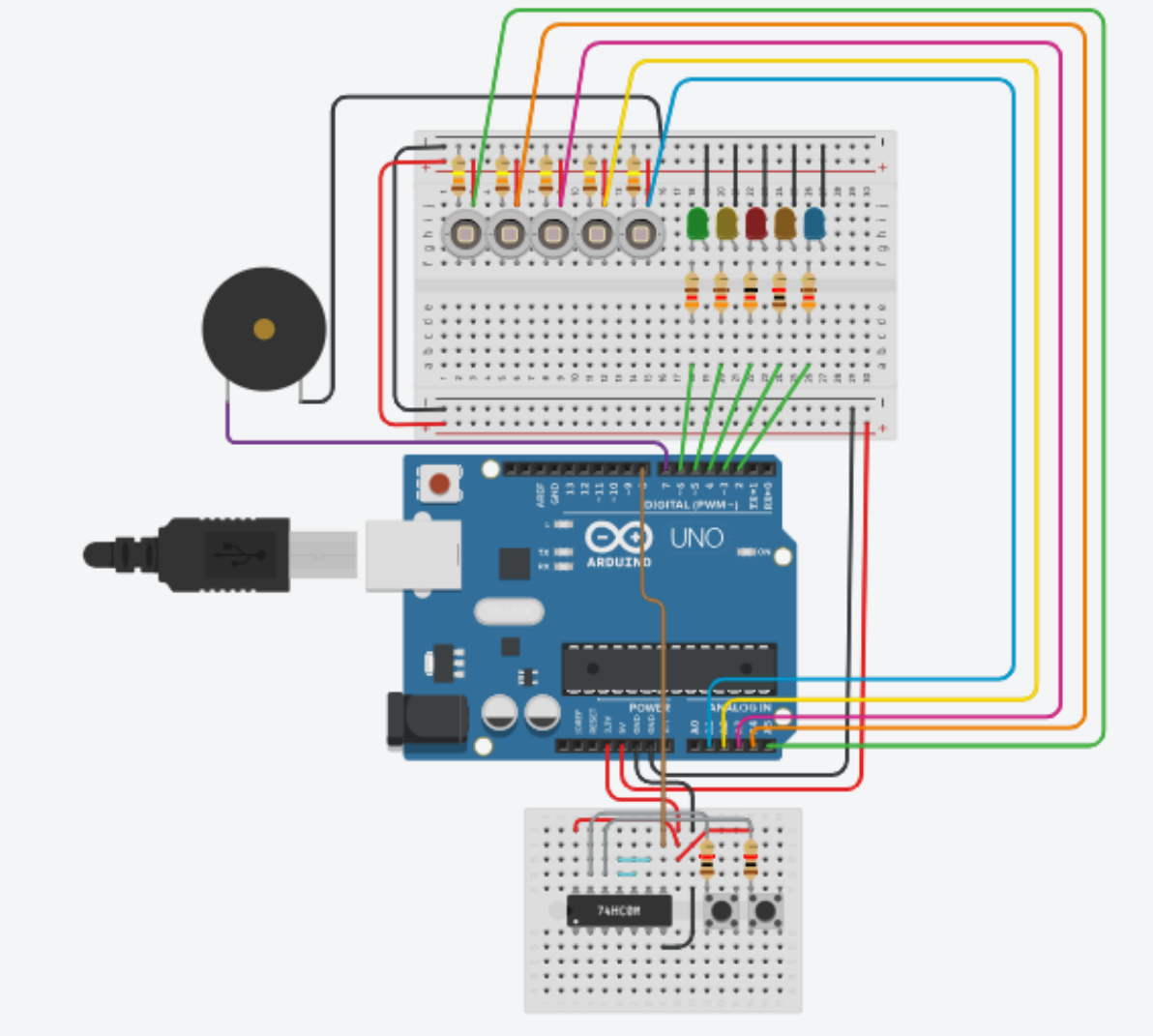
Logic and Processing:

The input of pressure to the push buttons was processed by the NAND gates which signaled to the Arduino whether the system should enter tutorial mode. Additionally, the Arduino microcontroller processed the input of pressure from the FSRs.

Outputs:

The LED's and speaker were the visible and audible outputs of our device. When in free play mode, the Arduino controlled the speaker and LED's to output both a flashing light and a tone for each input from the FSR (a finger pressed). When entering tutorial mode, the output of the logic gates was sent into a digital pin of the Arduino, which signaled the Arduino to cause all five LED's to flash simultaneously. The Arduino then controlled the LED's to flash to the tune of Jingle Bells, waiting to play the next note until the previous one was played.

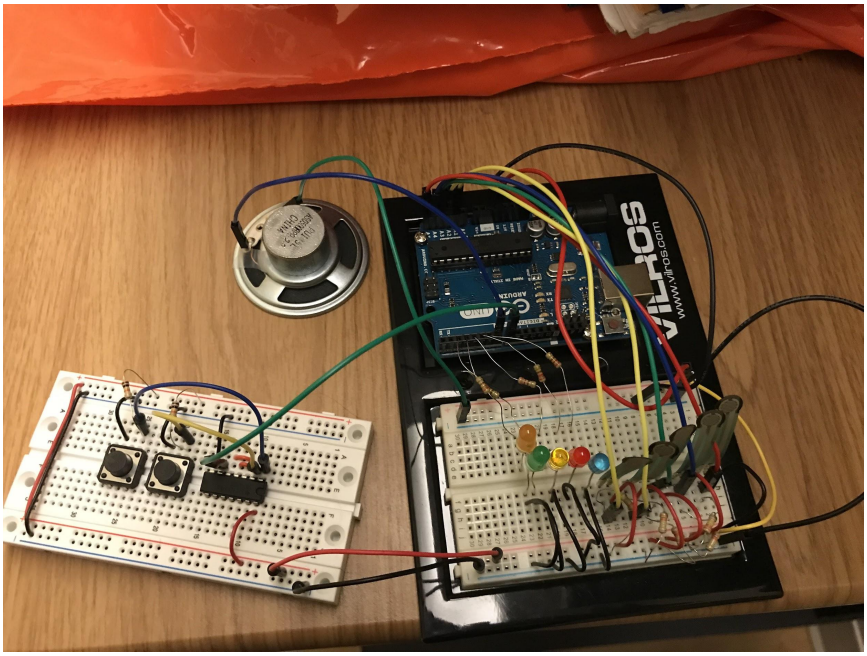
*Circuit Schematic*




Note: FSR's are represented as 

*Physical/Mechanical Construction*

Our design was constructed on a breadboard during prototyping. We placed the FSRs in the breadboard, along with the LED's and related wiring and resistors. We connected the speaker by the side of the breadboard.



**Conclusion**

*Lessons Learned*

One obstacle we encountered was the amount of time it took for our FSRs to arrive. Since we needed to order them online, we did not receive them for several weeks and were unable to begin building a prototype and testing our code until they arrived. This, along with the errors we encountered when attempting to implement the NAND gates, prevented us from creating a fully working glove, and we were only able to create a working prototype that was entirely connected to a breadboard instead of soldered together on a glove. During this process, we learned how to modify our ideas and goals in order to create a working design in the limited time we had left once we received our sensors and resolved any bugs. Additionally, when testing our logic gates and debugging our code in general, we learned that the Arduino serial monitor is incredibly helpful for isolating specific problems and determining what the values of certain inputs and variables are and whether these match the expected values.

*Self-Assessment*

Although we did not accomplish our initial goal of creating a fully working glove that could play music, we were able to create a prototype that could be soldered together and made into a glove, which we hope to complete in the future. Additionally, although we were not able to add in all of the features we initially wanted to, such as the ability to change octaves, we were able to create a working design that could play notes based on finger motions and incorporated the tutorial mode we had envisioned. In all, we are pleased with the progress we have made in developing a working prototype of our design and hope to expand upon this project to encompass more features in the future.

**References**

[1]"Piano TouchSense Glove", *Instructables.com*, 2018. [Online]. Available: https://www.instructables.com/id/Piano-TouchSense-Glove/. [Accessed: 22- Sep- 2018].

[2]"Scott Made This | Piano Gloves", *Scott.j38.net*, 2018. [Online]. Available: http://scott.j38.net/interactive/piano_gloves/. [Accessed: 22- Sep- 2018].

[3]S. Carroll and J. Talmage, "ECE 4760 Final Project: Piano Gloves", *People.ece.cornell.edu*, 2018. [Online]. Available: https://people.ece.cornell.edu/land/courses/ece4760/FinalProjects/f2015/swc63_ncm42_jmt329/swc63_ncm42_jmt329/webpage/index.html. [Accessed: 22- Sep- 2018].

[5] D. Ben Chan (Benechan) and V. profile, "Simple 5 finger Piano Songs - Easy Tunes for complete beginners", *Pianosage.blogspot.com*, 2018. [Online]. Available: http://pianosage.blogspot.com/2011/11/simple-5-finger-piano-songs- for.html. [Accessed: 15- Dec- 2018].

[6] "Play Sound on Pc With Arduino and Progduino", *Instructables.com*, 2018. [Online]. Available: https://www.instructables.com/id/play-sound-on-pc-with- arduino-and-progduino/. [Accessed: 15- Dec- 2018].

[7] S. Mullett, "Jingle Bells Very Easy Piano Sheet Music - Let's Play Music", *Let's Play Music*, 2018. [Online]. Available: https://www.letsplaykidsmusic.com/ jingle-bells-very-easy-piano-sheet-music/. [Accessed: 15- Dec- 2018].

**Appendix 1: Arduino Code**

```
#define NOTE_C4  262
#define NOTE_D4  294
#define NOTE_E4  330
#define NOTE_F4  349
#define NOTE_G4  392

int buzzer = 7;
int gates = 8;
int led1 = 2;
int led2 = 3;
int led3 = 4;
int led4 = 5;
int led5 = 6;
int fsr1Pin = 1;
int fsr1Reading = analogRead(fsr1Pin);
```

```
 int fsr2Pin = 2;
 int fsr2Reading = analogRead(fsr1Pin);
 int fsr3Pin = 3;
 int fsr3Reading = analogRead(fsr1Pin);
 int fsr4Pin = 4;
 int fsr4Reading = analogRead(fsr1Pin);
 int fsr5Pin = 5;
 int fsr5Reading = analogRead(fsr1Pin);
 int fsrs[] =
 {
  0, fsr1Reading, fsr2Reading, fsr3Reading, fsr4Reading, fsr5Reading
 };

 int song1Notes[] =
 {
  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,
  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_G4, NOTE_G4, NOTE_F4,
NOTE_D4, NOTE_C4
```

```
  };
  int song1Durations[] =
  {
    4, 4, 2, 4, 4, 2, 4, 4, 4, 4, 1, 4, 4, 4, 4, 4, 4, 4, 8, 8, 4, 4, 4, 4, 2, 4,
    4, 4, 2, 4, 4, 2, 4, 4, 4, 4, 1, 4, 4, 4, 4, 4, 4, 4, 8, 8, 8, 4, 4, 4, 4, 1
  };
  int song1LEDs[] =
  {
    led3, led3, led3, led3, led3, led3, led3, led5, led1, led2, led3, led4, led4, led4, led4, led4,
led3, led3, led3, led3, led3, led2, led2, led3, led2, led5,
    led3, led3, led3, led3, led3, led3, led3, led5, led1, led2, led3, led4, led4, led4, led4, led4,
led3, led3, led3, led3, led5, led5, led4, led2, led1
  };
  int song1Nums[] =
  {
    3, 3, 3, 3, 3, 3, 3, 5, 1, 2, 3, 4, 4, 4, 4, 4, 3, 3, 3, 3, 3, 2, 2, 3, 2, 5,
    3, 3, 3, 3, 3, 3, 3, 5, 1, 2, 3, 4, 4, 4, 4, 4, 3, 3, 3, 3, 5, 5, 4, 2, 1
  };


void setup() {
  // put your setup code here, to run once:
  pinMode(led1, OUTPUT);
  pinMode(led2, OUTPUT);
```

```
  pinMode(led3, OUTPUT);
  pinMode(led4, OUTPUT);
  pinMode(led5, OUTPUT);
  pinMode(buzzer, OUTPUT);
  pinMode(gates, INPUT_PULLUP);
  Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
  fsrs[1] = analogRead(fsr1Pin);
  fsrs[2] = analogRead(fsr2Pin);
  fsrs[3] = analogRead(fsr3Pin);
  fsrs[4] = analogRead(fsr4Pin);
  fsrs[5] = analogRead(fsr5Pin);
  int tutorial = digitalRead(gates);
  Serial.println(tutorial);

  if (fsrs[1] > 400 && tutorial == 1)
  {
    tone(buzzer, NOTE_C4, 1000/4);
    digitalWrite(led1, HIGH);
    delay(250);
```

```
  digitalWrite(led1, LOW);
}
else if (fsrs[2] > 400 && tutorial == 1)
{
  tone(buzzer, NOTE_D4, 1000/4);
  digitalWrite(led2, HIGH);
  delay(250);
  digitalWrite(led2, LOW);
}
else if (fsrs[3] > 400 && tutorial == 1)
{
  tone(buzzer, NOTE_E4, 1000/4);
  digitalWrite(led3, HIGH);
  delay(250);
  digitalWrite(led3, LOW);
}
else if (fsrs[4] > 400 && tutorial == 1)
{
  tone(buzzer, NOTE_F4, 1000/4);
  digitalWrite(led4, HIGH);
  delay(250);
  digitalWrite(led4, LOW);
}
```

```
else if (fsrs[5] > 400 && tutorial == 1)
{
  tone(buzzer, NOTE_G4, 1000/4);
  digitalWrite(led5, HIGH);
  delay(250);
  digitalWrite(led5, LOW);
}

if (tutorial == 0)
{
  Serial.println(tutorial);
  digitalWrite(led1, HIGH);
  digitalWrite(led2, HIGH);
  digitalWrite(led3, HIGH);
  digitalWrite(led4, HIGH);
  digitalWrite(led5, HIGH);
  delay(1000);
  digitalWrite(led1, LOW);
  digitalWrite(led2, LOW);
  digitalWrite(led3, LOW);
  digitalWrite(led4, LOW);
  digitalWrite(led5, LOW);
  delay(1000);
```

```
int wait = 1;
for (int currentNote = 0; currentNote < 51; currentNote++)
{
  delay(250);
  digitalWrite(song1LEDs[currentNote], HIGH);
  delay(250);
  while (wait == 1)
  {
    fsrs[1] = analogRead(fsr1Pin);
    fsrs[2] = analogRead(fsr2Pin);
    fsrs[3] = analogRead(fsr3Pin);
    fsrs[4] = analogRead(fsr4Pin);
    fsrs[5] = analogRead(fsr5Pin);
    if (fsrs[song1Nums[currentNote]] > 400)
    {
      int noteDuration = 1000 / song1Durations[currentNote];
      int note = song1Notes[currentNote];
      tone(buzzer, note, noteDuration);
      digitalWrite(led1, LOW);
      digitalWrite(led2, LOW);
      digitalWrite(led3, LOW);
      digitalWrite(led4, LOW);
      digitalWrite(led5, LOW);
```

```
        wait = 0;
       }
      }
      wait = 1;
    }
    digitalWrite(led1, HIGH);
    digitalWrite(led2, HIGH);
    digitalWrite(led3, HIGH);
    digitalWrite(led4, HIGH);
    digitalWrite(led5, HIGH);
    delay(1000);
    digitalWrite(led1, LOW);
    digitalWrite(led2, LOW);
    digitalWrite(led3, LOW);
    digitalWrite(led4, LOW);
    digitalWrite(led5, LOW);
    delay(1000);
  }
}
```