AIRduino Guitar

Andrew Bejjani
Electrical and Computer Engineering
American University of Beirut
aab68@mail.aub.edu

Mohammad Alkatranji
Electrical and Computer Engineering
American University of Beirut
maa342@mail.aub.edu

Nadim Dib
Electrical and Computer Engineering
American University of Beirut
ngd04@mail.aub.edu

I. Motivation

Guitars have been an instrumentational part of music for centuries and continue to be one of the most widely used instruments in today's world. They can expose multiple genres and styles such as rock, pop, country, blues, etc. However, there are a lot of restrictions in which not everyone can afford to buy a guitar. According to recent statistics published by the Statista Research Department, the average price of a guitar has increased dramatically between the years 2005 and 2020 in the U.S and reached an average price of 606 U.S. dollars for one guitar [1]. Even the people who can afford a guitar end up giving up playing the instrument since the learning curve for guitars is very steep. It is mostly due to the lack of dexterity between both hands, since the player must simultaneously change chords, which is painful for a newcomer's left-hand fingertips, and keep up the rhythm with the right hand [2]. For this reason, we propose to build an instrument known as the air guitar. Air guitar is mostly known as an invisible guitar in which people passionate about music can mimic playing a real guitar without holding one. It has become a popular form of entertainment and self-expression, with air guitar competitions held all over the world. Since 1996, Air Guitar World Championships have been arranged in Oulu where participants perform exaggerated movements, as if they are playing a guitar without any actual music [3]. The added value we will provide is that air guitar players will be able to generate music and sound while mimicking playing the guitar. Such a product can be revolutionary, especially for people this passionate about music.

II. Overview

Our instrument will be composed of an Arduino UNO microcontroller which will be used to interface with the needed sensors and a laptop to connect with a Digital Audio Workstation (DAW), which is defined as an electronic music generator.

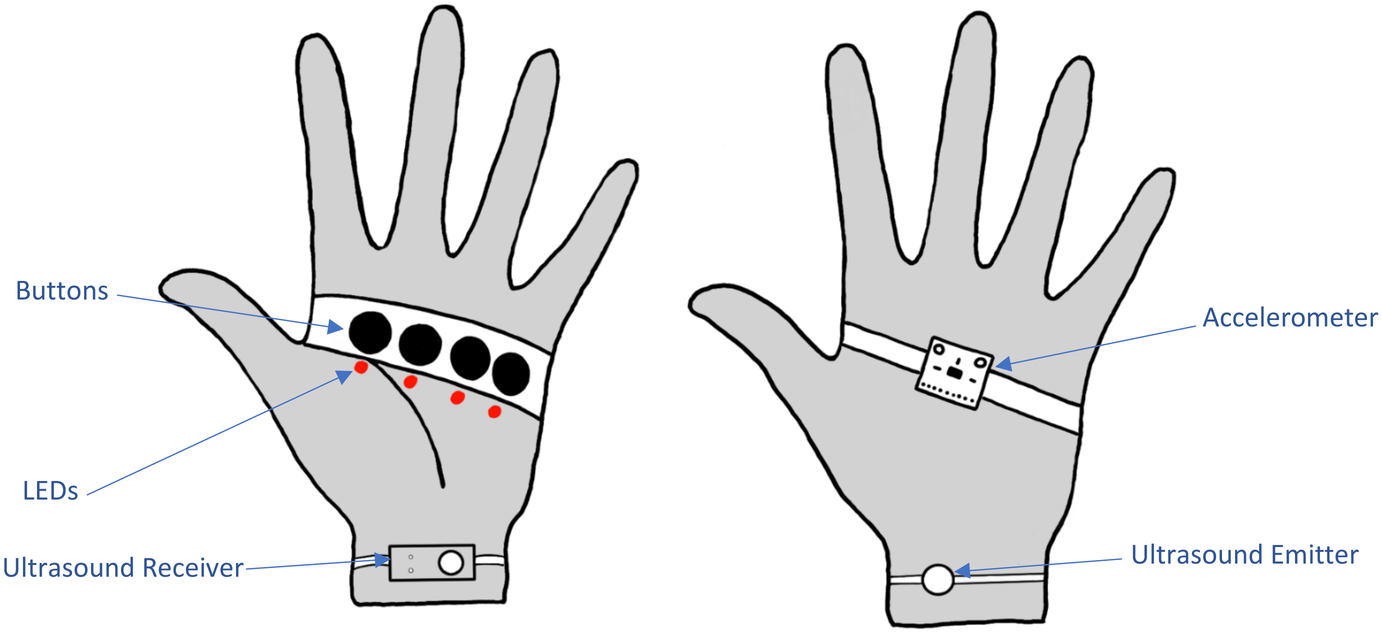


Figure 1: Sensor Placement Diagram

As shown by the figure above, our instrument will be composed of two main sensors, an ultrasonic sensor to measure the position between both playing hands and a 6-axis accelerometer used to strum the air guitar with the right hand. The accelerometer will measure whether the user has strummed the air guitar or not by detecting sudden rises in acceleration. In addition to these sensors, we will integrate four pushbuttons along with resistors and LEDs for each button on the palm of the left hand to map the needed chords. Each pushbutton will map to 2 chords, and the distance between both hands will select which chord the user is playing. We also have an LCD display on the user's left hand which will display the chord being played. Once a chord has been selected and a strum has been played, it will generate a signal to the Arduino UNO which will communicate with the DAW, to play the desired sound.

The Arduino will act as a MIDI controller, which transmits Musical Instrument Digital Interface (MIDI) data to MIDI-enabled devices. In our instrument, the Arduino will interface with CakeWalk (DAW) by sending it MIDI messages. CakeWalk will decode these messages into musical notes and will output guitar chords to the laptop speakers based on our sensor inputs.

III. Hardware Design

The list of hardware devices and components needed along with their functionalities in the instrument are presented below along with the hardware schematic of how they are connected:

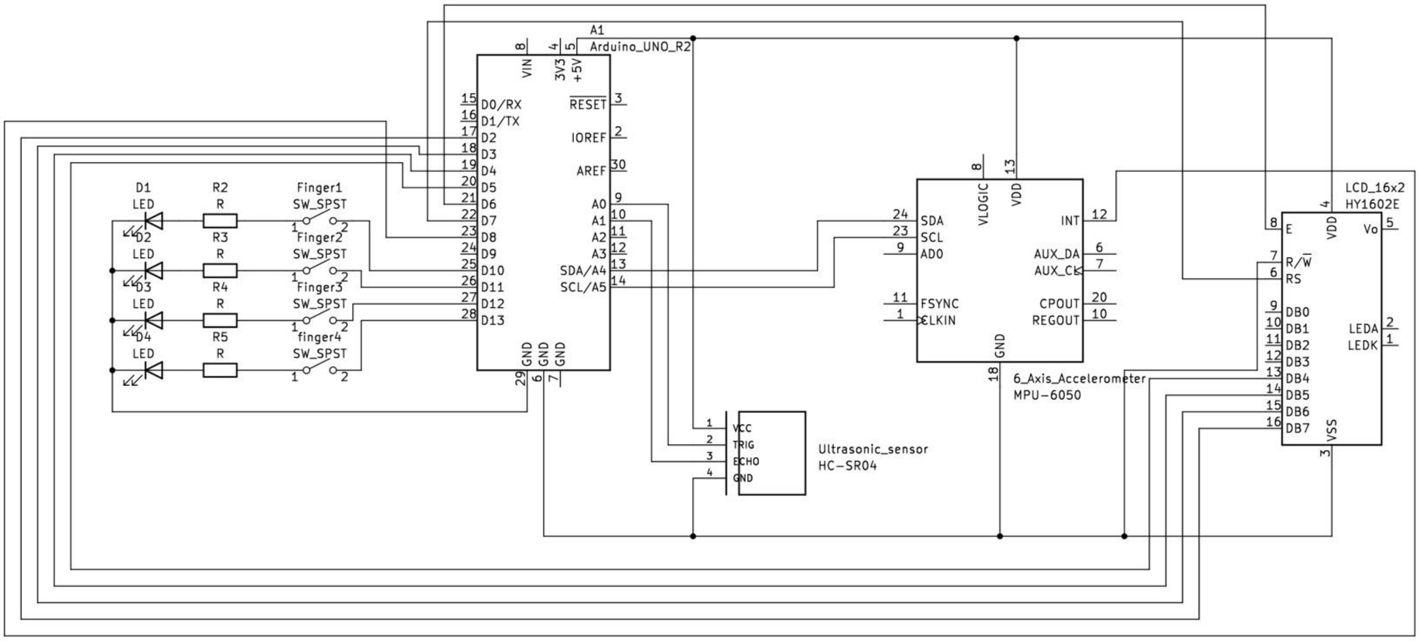


Figure 2: Hardware Schematic

<u>Arduino Uno Rev3:</u>

Programmable microcontroller that has 14 digital input/output pins, 6 analog inputs, a USB connection, a power jack, and an ICSP header for programming. It can interface with accelerometers and infrared transmitter-receivers which will be used in our case. Its role is to read the output of the sensors and send the needed signals based on the measurements we read.

<u>6-axis Accelerometer MPU-6050:</u>

MPU6050 sensor module is a complete 6-axis Motion Tracking Device. It combines a 3-axis Gyroscope, 3-axis Accelerometer, and Digital Motion Processor all in a small package. [4]

- Gyroscope:
  - 3-axis sensing with a full-scale range of ±250, ±500, ±1000, or ±2000 degrees per second (DPS)
  - Sensitivity of 131, 65.5, 32.8, or 16.4 LSBs per DPS
- Accelerometer:
  - 3-axis sensing with a full-scale range of ±2g, ±4g, ±8g, or ±16g
  - Sensitivity of 16384, 8192, 4096, or 2048 LSBs per g

HC-SR04 Ultrasonic Sensor (Modified):

This Ultrasonic sensor is a distance measuring sensor which is composed of two ultrasonic transducers. One is a transmitter which outputs ultrasonic sound pulses and the other is a receiver which listens for reflected waves. It has 4 pins that connect to Vcc, Trig (to send ultrasound wave), Echo (to receive the wave), and GND. It also has the following specifications:

- Operating Voltage: 5V DC
- Min Range: 2cm
- Max Range: 400cm
- Accuracy: 3mm (1.54%)

For our application, our distance will vary between 20cm and around 90cm and we will only have 1 threshold to distinguish the distance between both hands. So overall this sensor has good accuracy and range for our application. [5]

Pushbuttons:

Placed on the fingertips and the palm of the left hand. They will as a switch that closes when a button is pressed indicating that the user wants to play a chord.

LCD-016N002L1:

Displays the chord being played. Has 2 rows that hold 16 characters each, which is plenty. Since the target audience is beginner guitarists this will help them get familiar with the chords that sound good together. [2]

LEDs:

LEDs are Light Emitting Diodes that will be mounted on the left hand in series with the pushbuttons and light up whenever the user presses a button to indicate that the Arduino has received the signal.

Resistors:

Resistors will be mounted on the left hand in series with the diodes to limit the current passing through it whenever the user presses a pushbutton.

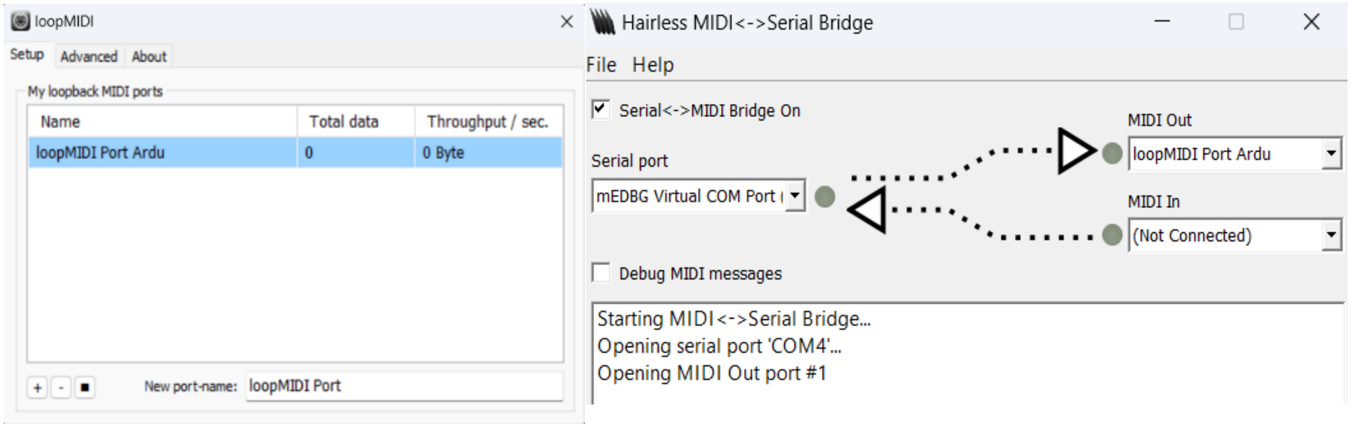
Laptop Dell G15 5510:

Dell G15 5510 high-performance laptop that will run the DAW and interface with the microcontroller. It will also be used as a speaker to output the chords after the user’s intervention from the DAW.

IV. Software Design

MIDI protocols:

MIDI is a set of instructions stored in a Standard MIDI File (SMF) interpreted by a DAW. We'll be sending our customized instructions from the Arduino which will be acting as a MIDI controller through software tools such as loop MIDI, which will allow us to classify our Arduino controller as a MIDI device, and Hairless MIDI Serial Bridge which will allow us to map a serial USB port to a MIDI controller and specify whether it's sending or receiving data.



My loopback MIDI ports

Name	Total data	Throughput / sec.
loopMIDI Port Ardu	0	0 Byte

New port-name: loopMIDI Port

☑ Serial<->MIDI Bridge On

Serial port: mEDBG Virtual COM Port (

MIDI Out: loopMIDI Port Ardu

MIDI In: (Not Connected)

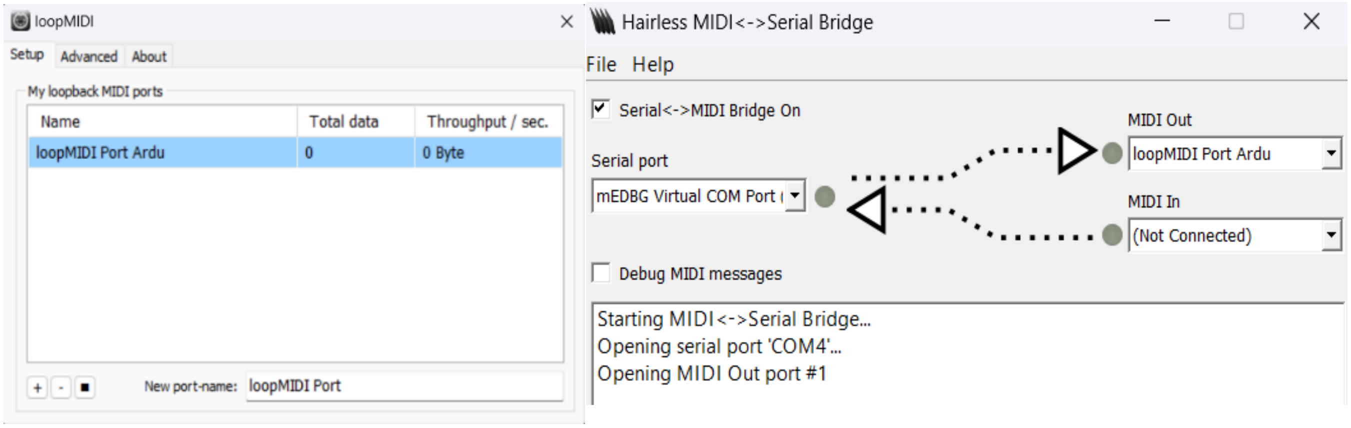
☐ Debug MIDI messages

```
Starting MIDI<->Serial Bridge...
Opening serial port 'COM4'...
Opening MIDI Out port #1
```

Figure 3: loopMIDI and Hairless MIDI Interface

MIDI protocols:

MIDI is a set of instructions stored in a Standard MIDI File (SMF) interpreted by a DAW. We'll be sending our customized instructions from the Arduino which will be acting as a MIDI controller through software tools such as loop MIDI, which will allow us to classify our Arduino controller as a MIDI device, and Hairless MIDI Serial Bridge which will allow us to map a serial USB port to a MIDI controller and specify whether it's sending or receiving data.



My loopback MIDI ports

Name	Total data	Throughput / sec.
loopMIDI Port Ardu	0	0 Byte

New port-name: loopMIDI Port

☑ Serial<->MIDI Bridge On

Serial port: mEDBG Virtual COM Port (

MIDI Out: loopMIDI Port Ardu

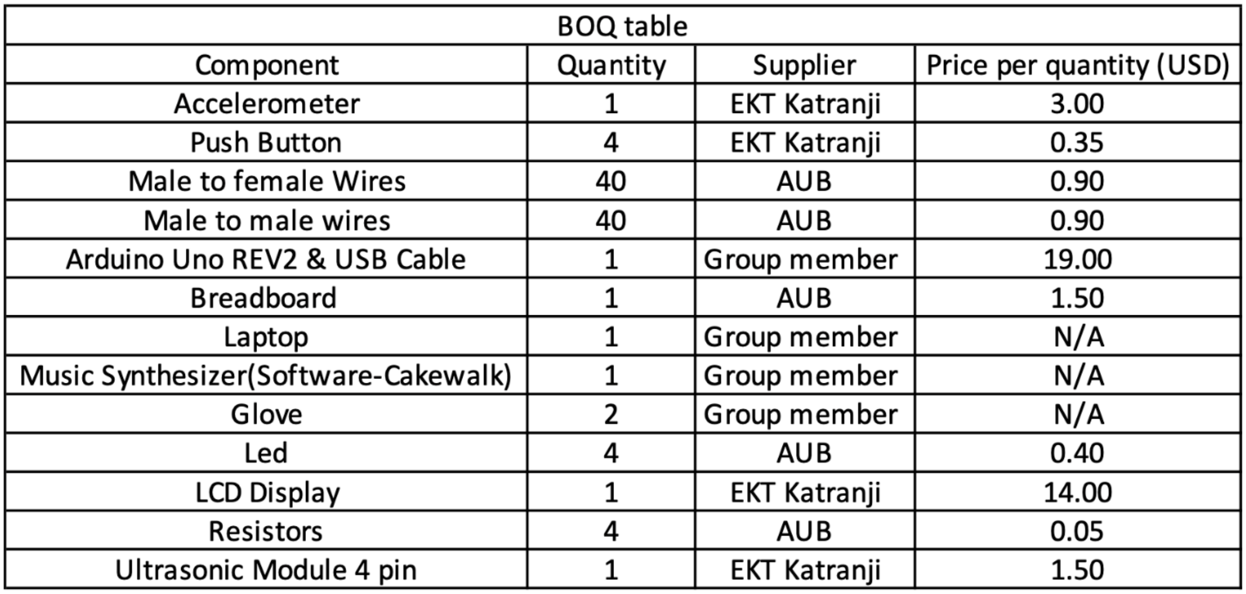
MIDI In: (Not Connected)

☐ Debug MIDI messages

```
Starting MIDI<->Serial Bridge...
Opening serial port 'COM4'...
Opening MIDI Out port #1
```

Figure 4: loopMIDI and Hairless MIDI Interface

V. BoQ Table



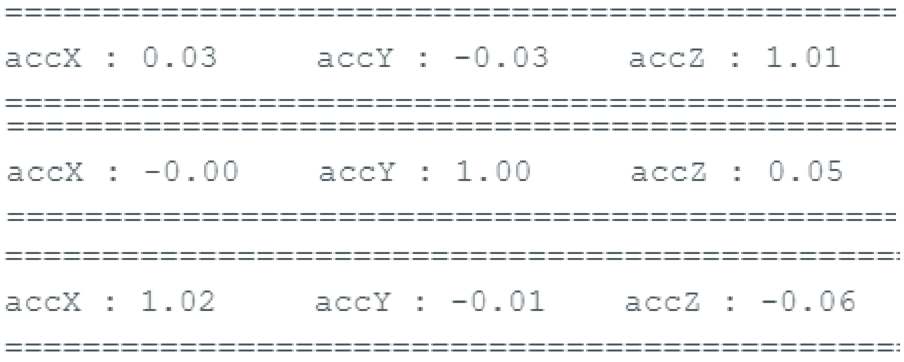
BOQ table			
Component	Quantity	Supplier	Price per quantity (USD)
Accelerometer	1	EKT Katranji	3.00
Push Button	4	EKT Katranji	0.35
Male to female Wires	40	AUB	0.90
Male to male wires	40	AUB	0.90
Arduino Uno REV2 & USB Cable	1	Group member	19.00
Breadboard	1	AUB	1.50
Laptop	1	Group member	N/A
Music Synthesizer(Software-Cakewalk)	1	Group member	N/A
Glove	2	Group member	N/A
Led	4	AUB	0.40
LCD Display	1	EKT Katranji	14.00
Resistors	4	AUB	0.05
Ultrasonic Module 4 pin	1	EKT Katranji	1.50

Figure 5: BoQ Table [7]

If price is stated when AUB or Group member is the supplier, then this is price is from EKT Katranji.

VI. MPU6050 Accelerometer Testing

Before proceeding with the design, we need to check whether the accelerometer is properly calibrated and has good readings, accuracy, and sensitivity. When the Arduino UNO is powered up, the MPU6050 automatically calibrates itself to calculate the offsets of the accelerometer. For this we should keep it stable for around five seconds for it to calibrate properly. Once calibrated, we would be able to read the acceleration values for the sensor by printing them in the serial monitor.



```
==================================================
accX : 0.03     accY : -0.03    accZ : 1.01
==================================================
==================================================
accX : -0.00    accY : 1.00     accZ : 0.05
==================================================
==================================================
accX : 1.02     accY : -0.01    accZ : -0.06
==================================================
```

Figure 6: Accelerometer Readings to Validate Calibration

The figure above shows 3 readings displayed for the accelerometer. The first reading is done when we positioned the accelerometer along the z-axis, for this reason we read a value around 1 for accZ since it can detect the effect of gravity g. The values displayed are normalized, so all values are divided by the gravitational constant g which is around 9.8m/s^2. The first reading has the accelerometer positioned along the z-axis, the second reading has it positioned along the y-axis, and the third reading has it positioned along the x-axis. From these readings, we can calculate the accuracy of the sensor:

Accuracy 1= $\frac{(measured\ value - actual\ value)}{actual\ value} * 100\% = \frac{\sqrt{(0.03-0.00)^2+(-0.03-0.00)^2+(1.01-1.00)^2}}{1} * 100\% = 4.36\%$

Accuracy 2= $\frac{(measured\ value - actual\ value)}{actual\ value} * 100\% = \frac{\sqrt{(-0.00-0.00)^2+(1.00-1.00)^2+(0.05-0.00)^2}}{1} * 100\% = 5.00\%$

Accuracy 3= $\frac{(measured\ value - actual\ value)}{actual\ value} * 100\% = \frac{\sqrt{(1.02-1.00)^2+(-0.01-0.00)^2+(-0.06-0.00)^2}}{1} * 100\% = 6.40\%$

Overall Sensor Accuracy = $\frac{Accuracy\ 1+Accuracy\ 2+Accuracy\ 3}{3} = 5.25\%$

This means that overall, a reading from the accelerometer can vary around ± 5.25%. This is decent since most accelerometers have an accuracy that varies between 5% and 10%. Note that this accuracy has been attained from experimental means and it is subject to real life errors (not positioning the sensor 100% along the x-axis etc.). However, we need the accuracy of the accelerometer to get the overall system accuracy later on, this is due to the absence of the sensor accuracy in the datasheet.
In our application, we need the accelerometer to detect sudden bursts in acceleration to detect any strums made by the users.

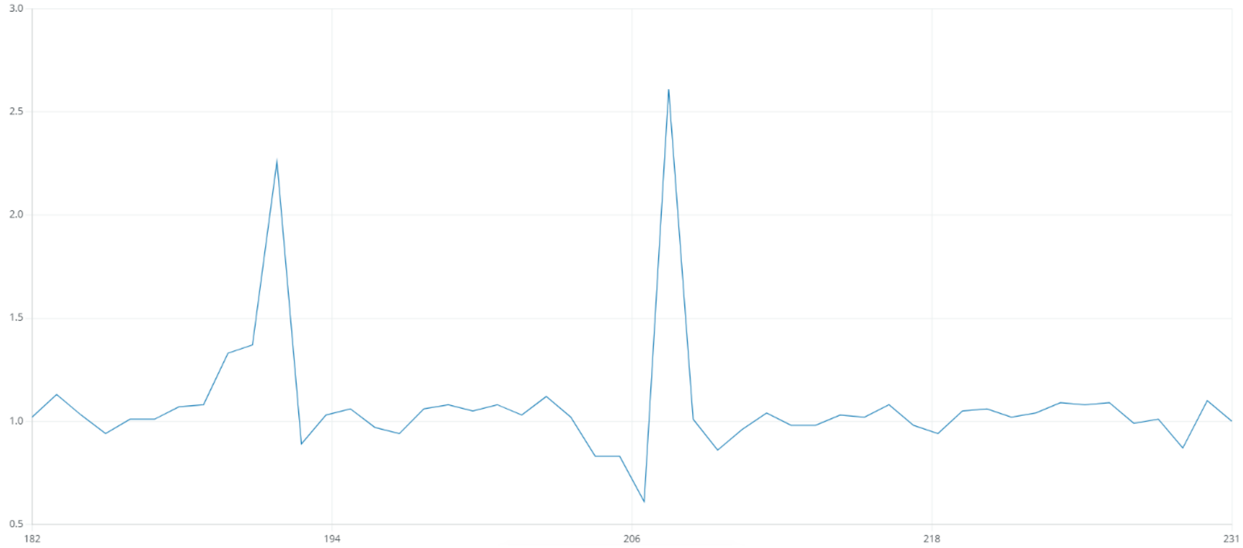


Figure 7: Serial Plotter of Average Acceleration in Arduino

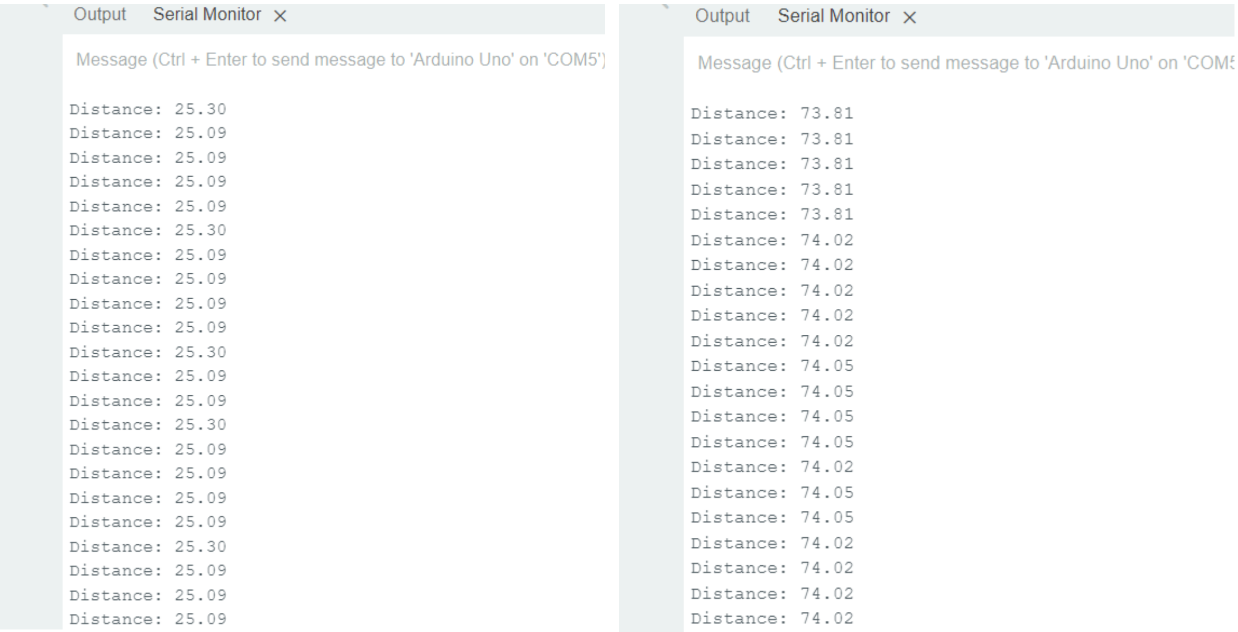
In the figure above, we have plotted the average acceleration of MPU6050 which is defined as follows:
$Average\ Accelertion = \sqrt{accX^2 + accY^2 + accZ^2}$
In this plotter, we can detect 2 sudden spikes which correspond to two strums done by the user. So, once we detect such spikes by setting a threshold for the acceleration, we play the corresponding chord selected by the user based on which button was pressed and the distance between both hands.

VII. Ultrasonic Distance Sensor (HC-SR04) Testing

For our application, we had to tweak the ultrasonic in a way to measure distance between two points in space. The ultrasonic is originally designed to measure distance between the user and a wall, so a wave would propagate back and forth to measure the distance. However, in our case, the chances of the sound wave hitting the other hand and propagating back are very low and it is very inconsistent. For this reason, we detached one of the transducers from the sensor and held it with the right hand and kept the original sensor module on the left hand and kept the internal wiring of the sensor intact. By doing this, we are now able to transmit sound directly without having the wave to bounce back and forth.

The way to do this is by setting the Trig to HIGH for 10µs and then to LOW. This will send out an 8-cycle ultrasonic burst which will travel at the speed of sound. Then we measure the duration that the bursts took to reach the receiver module. If no signal was received in 0.03s, we no longer wait for the signal to reach the receiver since it most probably overshot it. This saves a lot of delay time in the code since when the user strums, he might not always have the transmitter send signals directly in the receiver's way. Once we get the duration, we can get the distance by applying the formula:

$\text{distance} = \text{duration} * 0.034$ where 0.034 is the speed of sound in cm/s.


Output
Serial Monitor ×
Message (Ctrl + Enter to send message to 'Arduino Uno' on 'COM5')
Output
Serial Monitor ×
Message (Ctrl + Enter to send message to 'Arduino Uno' on 'COM5

```
Distance: 25.30
Distance: 25.09
Distance: 25.09
Distance: 25.09
Distance: 25.09
Distance: 25.30
Distance: 25.09
Distance: 25.09
Distance: 25.09
Distance: 25.09
Distance: 25.30
Distance: 25.09
Distance: 25.09
Distance: 25.30
Distance: 25.09
Distance: 25.09
Distance: 25.09
Distance: 25.09
Distance: 25.30
Distance: 25.09
Distance: 25.09
Distance: 25.09
```

```
Distance: 73.81
Distance: 73.81
Distance: 73.81
Distance: 73.81
Distance: 73.81
Distance: 74.02
Distance: 74.02
Distance: 74.02
Distance: 74.02
Distance: 74.02
Distance: 74.05
Distance: 74.05
Distance: 74.05
Distance: 74.05
Distance: 74.02
Distance: 74.05
Distance: 74.05
Distance: 74.02
Distance: 74.02
Distance: 74.02
Distance: 74.02
```

Figure 8: Readings from the Ultrasonic sensor in 2 different ranges

Above are readings taken from the sensor at 2 ranges of interest. The first reading was taken by setting the receiver 25cm away from the transmitter, this was done by measuring the distance between them using a tape measure and verifying that using the sensor. We observed that the sensor picked up around 25cm which demonstrates good accuracy and precision. The same was done with the second reading where we set the receiver 74cms away from the transmitter and recorded the values sent. We observe that most values were varying around 0.3cm of each other which is supported since the sensor's accuracy is 3mm.

VIII. Overall Testing

Overall, our system will consist of two sensors of known accuracies, along with other hardware components. Four push buttons will be used to select which chords will be played along with the distance read from the ultrasonic sensor. We designed a PCB to hold the push buttons, LEDs, and resistors for the user to play the chords in a very easy and neat way. This PCB was placed on the palm of the left hand and the Arduino UNO was placed on the back of the hand along with the LCD. The transmitter of the ultrasonic was held on the right hand and the receiver was held on the left hand. The accelerometer was also held on the right hand, which is the hand that the user uses to strum.

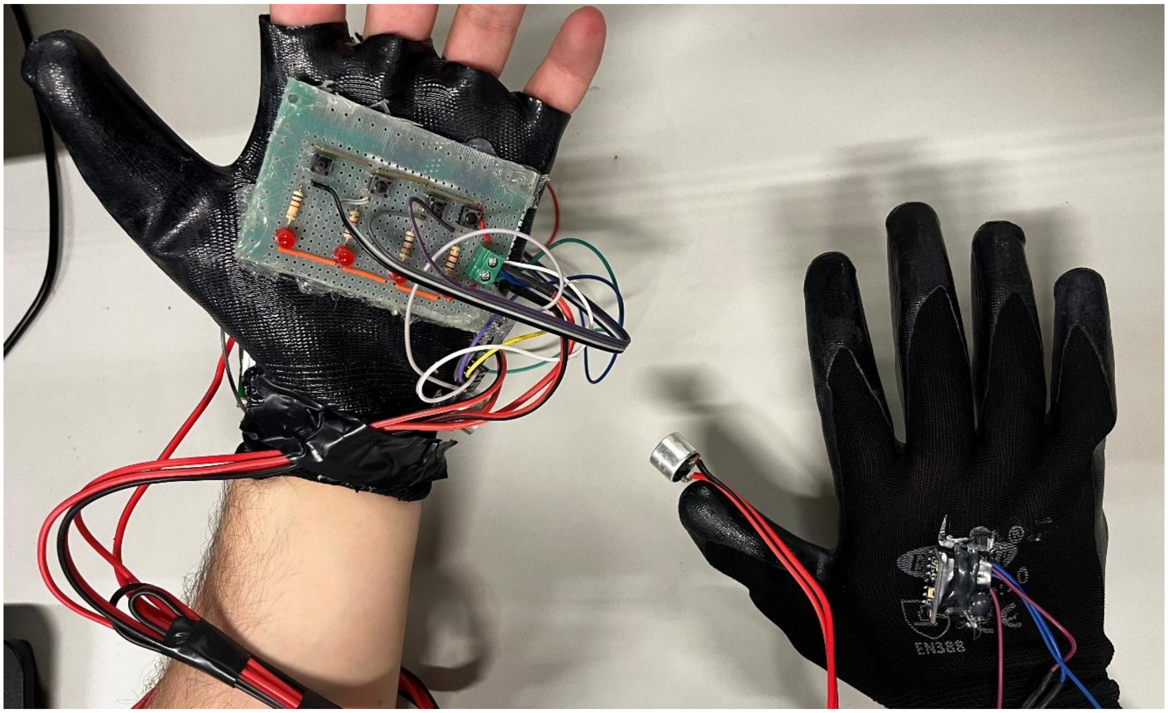


Figure 9: Final System Prototype

If we combine all of these together, we will get our system as the figure above, where the Arduino UNO and LCD are placed on the back of the left hand. Running this entire system with the sensors we have used; we can get a system accuracy of:

System Accuracy = $\sqrt{Acc\ MPU6050^2 + Acc\ Ultrasonic^2} = \sqrt{0.0525^2 + 0.0154^2} = 0.0547 = 5.47\%$

These results are descent since in our system, the accelerometer is used to detect sudden changes in acceleration which is what is happening and the readings from the ultrasonic are sufficient and accurate enough to determine which chords will be played.

IX. Results and Implementation

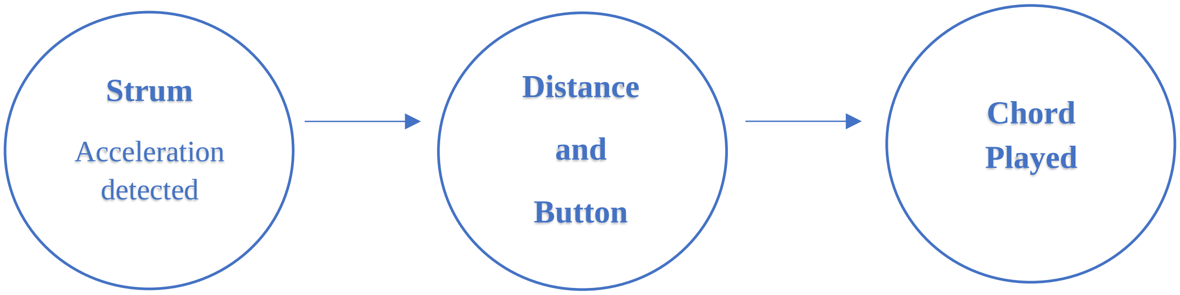
Ultrasonic:

Sensor is divided into Transmitter (which emits the wave) and Receiver (which reads the wave back). We calculate the distance between both and use the result to pick the correct chord based on the pressed button.

Accelerometer:

Measures the average acceleration of the right hand from the acceleration of on each axis (x, y, z) and the result is used to detect if a strum is taking place. If the average acceleration goes beyond a certain threshold a sound will be played.

System Diagram:



X. Conclusion

The goal of this instrument is to help beginners train their rhythm and help flatten the learning curve by showing them what they would be able to achieve if they went ahead and continued down that path. Simulating an actual guitar would be unintuitive for the user as he would eventually not know how to place his hands on the imaginary fretboard (a real guitar has dots marking the fret numbers making it easier on the physical instrument), but the air guitar would need the user to replicate the position from memory thus taking away from its simplicity. Our guitar captures this simplicity and therefore will attract any user interested in learning more about music and playing guitars.

XI. References

[1] D. H. Stipp, "Statista - Average unit price of a guitar in the U.S. from 2005 to 2021," March 2022. [Online].

[2] T. Fontana, "TheGuitarLesson.com - The 5 Biggest Challenges of Learning the Guitar," [Online].

[3] "Air Guitar World Championships," [Online].

[4] "MPU6050 (Gyroscope + Accelerometer + Temperature) Sensor Module," [Online]. Available: https://www.electronicwings.com/sensors-modules/mpu6050-gyroscope-accelerometer-temperature-sensor-module.

[5] U. S. H.-S. a. A. –. C. Guide, "How to Mechatronics," [Online]. Available: https://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/.

[6] "16 x 2 Character LCD," VISHAY, [Online]. Available: https://www.vishay.com/docs/37299/37299.pdf.

[7] "EKT Katranji," [Online]. Available: https://www.katranji.com .