A Project Report on

Motion Based Musical Instruments

Submitted by

Tilak Jilka (Roll no. 58)

Rohit Vaidya (Roll no. 59)

Preksha Vartak (Roll no. 60)

in partial fulfillment for the award of the degree

BACHELOR OF ENGINEERING

in

Electronics and Telecommunication Engineering

Under the Guidance of

Dr. Gautam Shah

St. Francis Institute of Technology, Mumbai

University of Mumbai

2023 - 2024

CERTIFICATE

This is to certify that Tilak Jilka, Rohit Vaidya, Preksha Vartak are the bonafide students of St. Francis Institute of Technology, Mumbai. They have successfully carried out the project titled "Motion Based Musical Intruments" under the domain Embedded Systems, in partial fulfilment of the requirement of B. E. Degree in Electronics and Telecommunication Engineering of Mumbai University during the academic year 2023-2024. The work has not been presented elsewhere for the award of any other degree or diploma prior to this.

Dr. Gautam Shah
(Project Guide)

Internal Examiner

External Examiner

Dr. Kevin Noronha
(EXTC HOD)

Dr. Sincy George
(Principal)

Project Report Approval for B.E.

This project entitled ***'Motion Based Musical Instruments'*** by **Tilak Jilka, Rohit Vaidya, Preksha Vartak** is approved for the degree of Bachelor of Engineering in Electronics and Telecommunication from University of Mumbai.

Examiners

1. - - - - - - - - - - - - - - - - - -

2. - - - - - - - - - - - - - - - - - -

Date:

Place:

ACKNOWLEDGEMENT

We are thankful to a number of individuals who have contributed towards our final year project and without their help; it would not have been possible. Firstly, we offer our sincere thanks to our project guide, Dr. Gautam Shah for his constant and timely help and guidance throughout our preparation.

We are grateful to the college authorities and the entire faculty for their support in providing us with the facilities required throughout this semester. We are thankful to Dr. Uday Pandit khot and Dr. Deepak Jayaswal for their constant motivation and guidance throughout.

We are also highly grateful to Dr. Kevin Noronha, Head of Department (EXTC), Principal, Dr. Sincy George, and Director Bro. Shantilal Kujur for providing the facilities, a conducive environment and encouragement.

Signatures of all the students in the group

(Tilak Jilka)

(Rohit Vaidya)

(Preksha Vartak)

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included; we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in this submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Signatures of all the students in the group

(Tilak Jilka)

(Rohit Vaidya)

(Preksha Vartak)

Abstract

A motion-based musical instrument is an innovative creation that allows musicians to produce sound and music through physical movements and gestures. These instruments leverage motion sensors, accelerometer, and gyroscope to translate different types of movement into musical input. The above sensors will be used to track the hand movement of users and will let users change multiple octaves. Also, it can be used for switching between types of instruments. The input from sensors will be passed through a micro-controller to the web server where the musical notes are assigned and mapped to the sensors. This idea combines music, technology, and interactive art, providing musicians new ways to express themselves and engage with music. This device provides users a platform that can embed multiple instruments of different categories such as string, keyboard, etc.

Keywords: *Hand movement, Motion sensor, Embedded System, Web Audio API*

Contents

List of Figures

List of Abbreviations

API Application Programming Interface

GPIO General Purpose Input Output

Chapter 1

Introduction

1.1 Motivation

We maintain a fundamental understanding of music and its theory, and this project harmoniously merge music theory and engineering technologies. Provide an interactive platform for learners to explore and express musical creativity. Integrate tools like web audio APIs, sensors, and real-time data processing. Foster cross-disciplinary collaboration between musicians and engineers. There is a need to design an innovative device that leverages motion sensors, such as accelerometers and gyroscopes, to detect the user's movements and translate them into musical notes or sounds.

1.2 Problem Statement

The aim is to develop a motion-based musical instrument that allows users to create music through intuitive movements. Traditional musical instruments may have a steep learning curve and may not be accessible to everyone. With time the instrument quality deteriorates and maintenance becomes a mandatory challenge.

1.3 Methodology

Firstly, there's need to create a simulator for the musical instrument using Web Audio API. After successfully executing the script on web server, next step is to interface the hardware i.e. the touch sensors and esp32 using the web socket. At start tested for static mode, but for playing in different scales and switching the instruments there's a need to interface gyroscope. After all the work is implemented with troubleshooting, place all the sensors and microcontroller as a single unit in a mould. Current plan is to keep it wired for time being, as progress is achieved implementation for wireless module can be started.

1.4 Organization of Project Report

This project report is organized as follows:

Chapter 2 presents the literature survey on the existing techniques.

Chapter 3 provides a brief explanation of design methodology and theory.

Chapter 4 is dedicated to the simulation and experimental results.

Chapter 5 presents the conclusions and future scope for this project.

Chapter 2

Literature Review

The system provided stable and intuitive 3-D CG graphics on a laptop PC, focusing on the guitar player's strumming hand. The camera angle from the head of the guitar to the strumming hand offered valuable guidance on correct playing form, akin to a professional guitarist's instructions. [1]

2.1 Wrist-worn Device

A simple C-chord piano playing, it was found that the wrist-worn motion-sensing device may estimate the strength of key touch. It was found that both the angular velocity and the linear acceleration-related parameter would have strong relationships with the onset MIDI velocity. As a result, the proposed system can be applied to evaluate the strength of piano key touch by using a wrist-worn motion-sensing device. [2]

- **Literature Survey:** In the initial stages of the conceptualization of this project, this study presents a survey of literature to study the work that has already been done in the field of Embedded systems and its application.

- **ESP32:** ESP32 is a series of low-cost, low-power system on a chip microcontrollers with integrated Wi-Fi and dual-mode Bluetooth. This microcontroller is mainly used for interfacing using wireless mode via Bluetooth or Wi-Fi. We are using GPIO pins for interfacing the touch sensors, ultrasonic sensors and accelrometer-gyroscope.

- **TTP223 Touch Sensor:** The TTP223 is a touch pad detector IC replicating a single tactile button. This touch detection IC is designed for replacing traditional direct button key with diverse pad size. Figure 2.1 shows the touch sensor.

- **MPU6050 Accelerometer and Gyroscope:** MPU6050 sensor module is complete 6-axis Motion Tracking Device. It combines 3-axis Gyroscope, 3-axis Accelerometer and Digital Motion Processor all in small package. It aids in the measurement of velocity, orientation, acceleration, displacement, and other motion-related features. It calculates the rotaion in form of "roll", "pitch" and "yaw". Raw refers to the rotation of an object around its longitudinal axis. Pitch is the rotation of an object around its lateral axis. Yaw refers to the rotation of an object around its vertical axis. Figure 2.2 shows the accelerometer and gyroscope.

- **HC-SR04 Ultrasonic Sensor** An HC-SR04 ultrasonic distance sensor actually consists of two ultrasonic transducers. One acts as a transmitter that converts the electrical signal into 40 KHz ultrasonic sound pulses. The other acts as a receiver and listens for the transmitted pulses. When the receiver receives these pulses, it produces an output pulse whose width is proportional to the distance of the object in front. This sensor provides excellent non-contact range detection between 2 cm to 400 cm (13 feet) with an accuracy of 3 mm. Figure 2.3 shows the ultrasonic sensor.

Figure 2.1: Touch Sensor Module

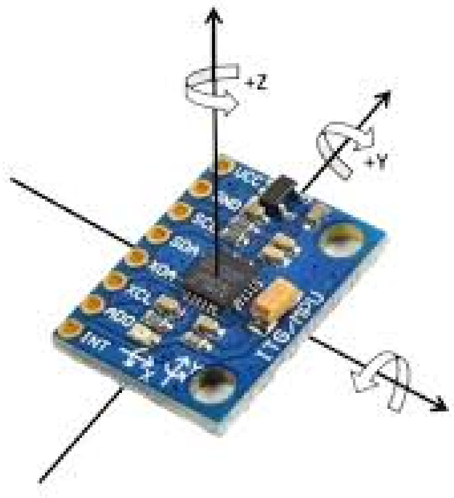


Figure 2.2: Accelerometer and Gyroscope

Figure 2.3: HC-SR04 Ultrasonic Sensor

Chapter 3

Theoretical Background/Design Methodology

3.1 Interfacing Sensors with ESP32

3.1.1 Touch Sensors

Touch sensors are hardware devices that detect touch or proximity. They are interfaced with the ESP32 microcontroller by connecting their output pins to specific GPIO pins on the ESP32. Each sensor is connected to a different GPIO pin to ensure we can read them individually. When a touch is detected, the sensor sends a signal to the ESP32, which will be used to trigger actions in the software. When a touch event occurs, data is collected such as which sensor was touched and any additional information. This data will be transmitted to the backend for further processing.

3.1.2 MPU6050 Accelerometer and Gyroscope

Interfacing an MPU6050 accelerometer and gyroscope with an ESP32 microcontroller for roll measurement entails programming the ESP32 to read data from the MPU6050 using I2C communication. Calibration ensures accuracy, and sensor fusion algorithms are applied to compute the roll angle from accelerometer and gyroscope data. The roll readings are parsed at every instance to avoid any delay in the instrument switching.

3.1.3 HCSR-04 Ultrasonic sensor

The HC-SR04 sensor emits ultrasonic waves and measures the time it takes for the waves to bounce back after hitting an object, allowing calculation of the distance. The ESP32 is programmed to trigger the sensor, measure the time of flight, and calculate the distance based on the speed of sound.

3.2 WebSocket Connection

3.2.1 WebSocket Client

There's a need set up a WebSocket client on the ESP32. This allows the microcontroller to establish a connection with the backend server. It's essential for sending sensor data in real-time.

3.2.2 WebSocket Server

On the backend,a WebSocket server will be established. WebSocket is a communication protocol that enables full-duplex, bidirectional communication channels over a single TCP connection. The server is responsible for listening to incoming WebSocket connections from clients, in this case, the ESP32.

3.2.3 ClientServer Communication

The ESP32 acts as a WebSocket client. It connects to the server using the server's IP address and port number. WebSocket allows real-time, low-latency communication. Both the client (ESP32) and server (backend) should have code to handle WebSocket messages.

3.3 Simulator Using Web Audio API

3.3.1 Music Simulator

Development of the backend music simulator, which is responsible for processing the data received from the ESP32. This simulator is written in a programming language

JavaScript which accurately captures movements of device and generates notes or sounds accordingly.

3.3.2 Web Audio API

The Web Audio API, available in modern web browsers, allows to create and manipulate audio in real-time. Within the simulator, the Web Audio API will be utilized to generate sound and control audio playback. Howler.js defaults to Web Audio API, it makes working with audio in JavaScript easy and reliable across all platforms.

3.3.3 Integration with Frontend

The simulator is integrated with a frontend interface, typically a web application, to allow users to interact with the motion-based musical instrument. The interface provides the user with a means to trigger the sensors and experience the generated music.

3.4 Design

Figure 3.1 and 3.2 illustrates the top view and bottom view respectively for basic prototype of the hand held device for user interaction. Top view has slots for 2x4 matrix consisting of 8 TTP223 touch sensors and the bottom plate acts as a support to the top plate and has slots for ESP32, ultrasonic sensor and MPU6050. It expresses the 3d model designed using software Fusion360.

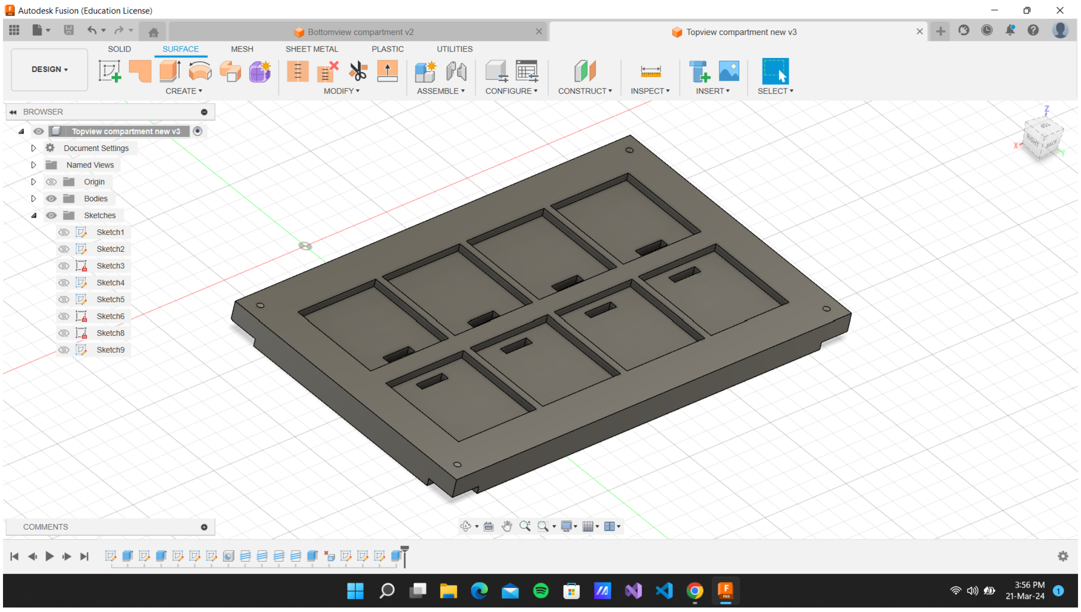


Figure 3.1: Top view of the design.

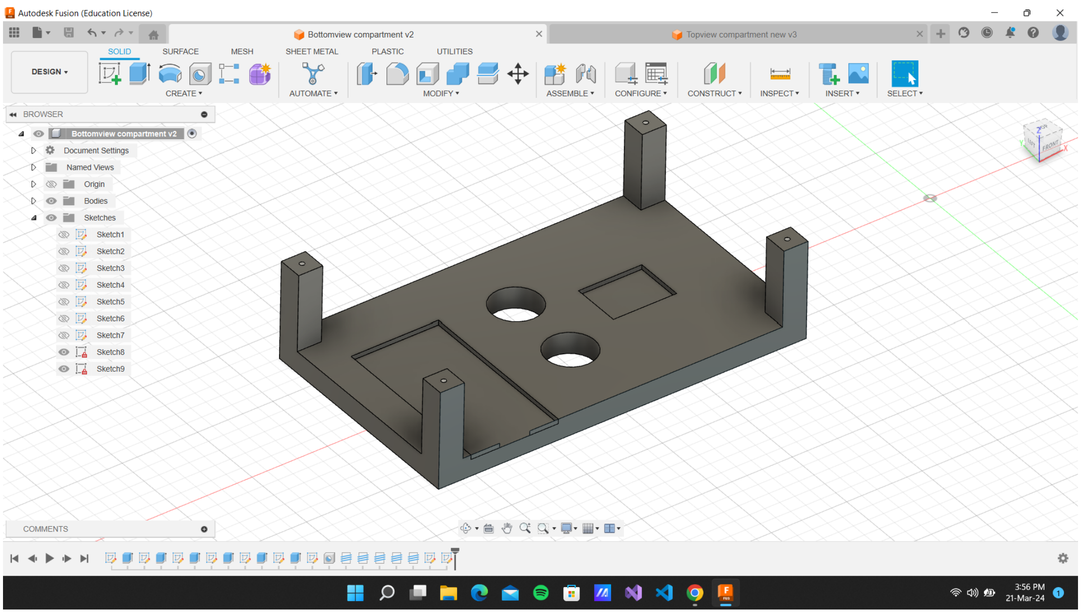


Figure 3.2: Bottom view of the design.

Chapter 4

Simulation and Experimental Results

Figure 4.1 illustrates the Outer view of the structure. The top plate is a 4x2 matrix of ttp223 touch sensors, the side and the bottom plate consists of one ultra-sonic sensor each.

Figure 4.2 illustrates the Inner view of the structure containing microcontroller, accelerometer and gyroscope (mpu6050).

Upon initialization, the web socket connection will be established, and the mpu6050 roll measurement, ultrasonic sensor, and touch sensor values will be transmitted to esp32 for processing. The sitar is the first instrument in this arrangement, active at roll of 0 degrees, followed by the piano at roll of 90 degrees. Following instrument switching, an octave switching check is performed using an ultasonic sensor to assess distance. One sensor will work at a time while the others are cut off. The javascript file maps the notes of three octaves that are defined at three threshold levels.

Figure 4.1: Outer structure

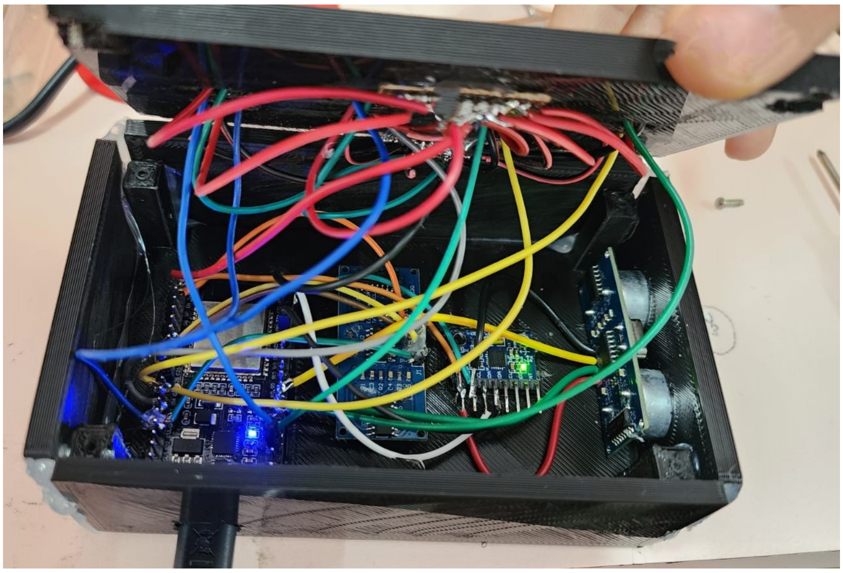


Figure 4.2: Inner structure

4.1 Code and Algorithm

4.1.1 main.cpp

main.cpp × index.html script.js

restructuring_final > src > main.cpp > trigPin2

```cpp
// Importing necessary libraries

#include <Arduino.h>
#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <ArduinoJson.h>
#include <WebSocketsServer.h>
#include <Wire.h>
#include <Adafruit_MPU6050.h>
#include <Adafruit_Sensor.h>

// defining variables for wifi connection
const char* ssid = "Wifi Name";
const char* password = "Wifi Password";
const int webSocketPort = 81; // Port for WebSocket communication

Adafruit_MPU6050 mpu;


// FOR ULTRASONIC SENSOR
const int trigPin1 = 32;
const int echoPin1 = 35;

const int trigPin2 = 16;
const int echoPin2 = 17;

//define sound speed in cm/uS
#define SOUND_SPEED 0.034

// To store distance of first ultrasonic sensor
long duration1;
int distanceCm1;
// To store distance of second ultrasonic sensor
long duration2;
int distanceCm2;

// FOR MPU6050 acclerometer and gyroscope

#define SDA_PIN 21   // Define the SDA pin (GPIO21 on ESP32)
#define SCL_PIN 22   // Define the SCL pin (GPIO22 on ESP32)


#define PRESS_DELAY 50   // Adjust this value based on your preference for press and hold duration
#define NUM_SENSORS 8
```

Figure 4.3: Code with hardware configuration

```cpp
// Pins for eight ultrasonic sensor
int touchPins[NUM_SENSORS] = {13, 12, 14, 27, 15, 26, 2, 4};

// variables for debounce logic for touch sensors
bool buttonStates[NUM_SENSORS] = {LOW};
bool lastButtonStates[NUM_SENSORS] = {LOW};
bool touchValue[NUM_SENSORS] = {LOW};
unsigned long lastDebounceTimes[NUM_SENSORS] = {0};
unsigned long lastPressTimes[NUM_SENSORS] = {0};

// websocket server intialization
AsyncWebServer webServer(80);
WebSocketsServer webSocket = WebSocketsServer(webSocketPort);

void webSocketEvent(uint8_t num, WStype_t type, uint8_t * payload, size_t length) {
    if (type == WStype_CONNECTED) {
        Serial.println("WebSocket client connected");
    }
}


void setup() {
    // baud rate
    Serial.begin(115200);

    // to connect to wifi for above mentioned credentials
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(1000);
        Serial.println("Connecting to WiFi...");
    }

    Serial.println("Connected to WiFi: ");
    Serial.println(WiFi.localIP());

    // defining gpio of touch sensor as input pins
    for (int i = 0; i < NUM_SENSORS; i++) {
        pinMode(touchPins[i], INPUT); // INPUT_PULLUP for ESP32
    }

    // starting websocket connection
    webSocket.begin();
    webSocket.onEvent(webSocketEvent);
```

```
  webServer.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(200, "text/html", "<html><body>Hello World</body></html>");
  });
  webServer.begin();


  pinMode(trigPin1, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin1, INPUT); // Sets the echoPin as an Input

  pinMode(trigPin2, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin2, INPUT); // Sets the echoPin as an Input


   // FOR MPU6050

  Serial.println("Adafruit MPU6050 test!");

  Wire.begin(SDA_PIN, SCL_PIN);
  // detecting mpu6050
  if (!mpu.begin()) {
    Serial.println("Failed to find MPU6050 chip");
    while (1) {
      delay(10);
    }
  }
  Serial.println("MPU6050 Found!");

// It sets the accelerometer range to ±8G and prints the selected range
  mpu.setAccelerometerRange(MPU6050_RANGE_8_G);
  Serial.print("Accelerometer range set to: ");
  switch (mpu.getAccelerometerRange()) {
  case MPU6050_RANGE_2_G:
    Serial.println("+-2G");
    break;
  case MPU6050_RANGE_4_G:
    Serial.println("+-4G");
    break;
  case MPU6050_RANGE_8_G:
    Serial.println("+-8G");
    break;
  case MPU6050_RANGE_16_G:
    Serial.println("+-16G");
    break;
```

```
}
// It sets the gyro range to ±500 degrees per second (deg/s) and prints the selected range.
  mpu.setGyroRange(MPU6050_RANGE_500_DEG);
  Serial.print("Gyro range set to: ");
  switch (mpu.getGyroRange()) {
  case MPU6050_RANGE_250_DEG:
    Serial.println("+- 250 deg/s");
    break;
  case MPU6050_RANGE_500_DEG:
    Serial.println("+- 500 deg/s");
    break;
  case MPU6050_RANGE_1000_DEG:
    Serial.println("+- 1000 deg/s");
    break;
  case MPU6050_RANGE_2000_DEG:
    Serial.println("+- 2000 deg/s");
    break;
}
  // It sets the filter bandwidth to 5 Hz and prints the selected bandwidth.
  mpu.setFilterBandwidth(MPU6050_BAND_5_HZ);
  Serial.print("Filter bandwidth set to: ");
  switch (mpu.getFilterBandwidth()) {
  case MPU6050_BAND_260_HZ:
    Serial.println("260 Hz");
    break;
  case MPU6050_BAND_184_HZ:
    Serial.println("184 Hz");
    break;
  case MPU6050_BAND_94_HZ:
    Serial.println("94 Hz");
    break;
  case MPU6050_BAND_44_HZ:
    Serial.println("44 Hz");
    break;
  case MPU6050_BAND_21_HZ:
    Serial.println("21 Hz");
    break;
  case MPU6050_BAND_10_HZ:
    Serial.println("10 Hz");
    break;
  case MPU6050_BAND_5_HZ:
    Serial.println("5 Hz");
    break;
}
```

```
  Serial.println("");
  delay(100);
}

void loop() {

    // starting websocket connecting in loop to trasmit readings continuously
    webSocket.loop();

    // Defining json data to pass sensor readings to frontend
    StaticJsonDocument<256> jsonDoc1;
    StaticJsonDocument<256> jsonDoc2;

    String jsonStr1;
    String jsonStr2;

    // FOR UNLTRASONIC SENSOR 1

      // Clears the trigPin
    digitalWrite(trigPin1, LOW);
    delayMicroseconds(2);
    // Sets the trigPin on HIGH state for 10 micro seconds
    digitalWrite(trigPin1, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin1, LOW);

    // Reads the echoPin, returns the sound wave travel time in microseconds
    duration1 = pulseIn(echoPin1, HIGH);

    // Calculate the distance
    distanceCm1 = duration1 * SOUND_SPEED/2;


    // FOR UNLTRASONIC SENSOR 2

      // Clears the trigPin
    digitalWrite(trigPin2, LOW);
    delayMicroseconds(2);
    // Sets the trigPin on HIGH state for 10 micro seconds
    digitalWrite(trigPin2, HIGH);
    delayMicroseconds(10);
    digitalWrite(trigPin2, LOW);
```

```
    // Reads the echoPin, returns the sound wave travel time in microseconds
    duration2 = pulseIn(echoPin2, HIGH);

    // Calculate the distance
    distanceCm2 = duration2 * SOUND_SPEED/2;

    // jsonDoc1["Distance"] = distanceCm;
    // jsonDoc2["Distance"] = distanceCm;


// mpu

    sensors_event_t a, g, temp;
    mpu.getEvent(&a, &g, &temp);


    // Calculate roll angle
    int roll = atan2(a.acceleration.y, a.acceleration.z) * 180 / M_PI;

    Serial.print("Roll: ");
    Serial.print(roll);
    Serial.println(" degrees");
    Serial.print("Distance 1: ");
    Serial.print(distanceCm1);
    Serial.println("cm");
    Serial.print("Distance 2: ");
    Serial.print(distanceCm2);
    Serial.println("cm");

    // delay(1000);

```

```
    // loop for detecting touch state of touch sensor
    for (int i = 0; i < NUM_SENSORS; i++) {
      int reading = digitalRead(touchPins[i]);

    // Debounce the button
    if (reading != lastButtonStates[i]) {
      lastDebounceTimes[i] = millis();
    }

    if (millis() - lastDebounceTimes[i] > PRESS_DELAY) {
      // Update the button state if it's been stable for a certain time
      if (reading != buttonStates[i]) {
        buttonStates[i] = reading;

        // Handle different press types
        if (buttonStates[i] == HIGH) {
          // Button is pressed
            lastPressTimes[i] = millis();
            Serial.print("Sensor ");
            Serial.print(i);
            Serial.println(" pressed");

            // define json objects in both json files
            jsonDoc1["roll"] = roll;
            jsonDoc2["roll"] = roll;

            jsonDoc1["Distance1"] = distanceCm1;
            jsonDoc2["Distance1"] = distanceCm1;

            jsonDoc1["Distance2"] = distanceCm2;
            jsonDoc2["Distance2"] = distanceCm2;

            jsonDoc1["touch1"] = buttonStates[0];
            jsonDoc1["touch2"] = buttonStates[1];
            jsonDoc1["touch3"] = buttonStates[2];
            jsonDoc1["touch4"] = buttonStates[3];

            jsonDoc2["touch5"] = buttonStates[4];
            jsonDoc2["touch6"] = buttonStates[5];
            jsonDoc2["touch7"] = buttonStates[6];
            jsonDoc2["touch8"] = buttonStates[7];
```

```
            // stringify json data
            serializeJson(jsonDoc1, jsonStr1);
            serializeJson(jsonDoc2, jsonStr2);
            // broadcast data into websocket server
            webSocket.broadcastTXT(jsonStr1);
            webSocket.broadcastTXT(jsonStr2);

            // print json data
            Serial.print("JSON 1: ");
            Serial.println(jsonStr1);
            Serial.print("JSON 2: ");
            Serial.println(jsonStr2);

          } else {
            // Button is released
            unsigned long pressDuration = millis() - lastPressTimes[i];
          }
        }
      }

      // Save the current button state for comparison in the next iteration
      lastButtonStates[i] = reading;
    }
  }
```

4.1.2 script.js

main.cpp | index.html | script.js

Restructuring_final > web > script.js > ...

```javascript
// defining websocket client
const socket = new WebSocket('ws://192.168.92.243:81'); // Replace with your ESP32 IP address

// checking for connections
socket.onopen = function(event) {
    console.log('WebSocket connection opened.');
};


// Octave 1 notes of Sitar
var C3_sitar = new Howl({
    src: ['./sitar_notes/C3.mp3'],
});
var D3_sitar = new Howl({
    src: ['./sitar_notes/D3.mp3']
});
var E3_sitar = new Howl({
    src: ['./sitar_notes/E3.mp3']
});
var F3_sitar = new Howl({
    src: ['./sitar_notes/F3.mp3']
});
var G3_sitar = new Howl({
    src: ['./sitar_notes/G3.mp3']
});
var A3_sitar = new Howl({
    src: ['./sitar_notes/A3.mp3']
});
var B3_sitar = new Howl({
    src: ['./sitar_notes/B3.mp3']
});
var C4_sitar = new Howl({
    src: ['./sitar_notes/C4.mp3']
});

// Octave 2 notes of Sitar
var C4_sitar = new Howl({
```

Figure 4.4: Code for back-end simulator

```
// Octave 2 notes of Sitar
var C4_sitar = new Howl({
    src: ['./sitar_notes/C4.mp3'],
});
var D4_sitar = new Howl({
    src: ['./sitar_notes/D4.mp3']
});
var E4_sitar = new Howl({
    src: ['./sitar_notes/E4.mp3']
});
var F4_sitar = new Howl({
    src: ['./sitar_notes/F4.mp3']
});
var G4_sitar = new Howl({
    src: ['./sitar_notes/G4.mp3']
});
var A4_sitar = new Howl({
    src: ['./sitar_notes/A4.mp3']
});
var B4_sitar = new Howl({
    src: ['./sitar_notes/B4.mp3']
});
var C5_sitar = new Howl({
    src: ['./sitar_notes/C5.mp3']
});

// Octave 3 notes of Sitar

var C5_sitar = new Howl({
    src: ['./sitar_notes/C5.mp3'],
});
var D5_sitar = new Howl({
    src: ['./sitar_notes/D5.mp3']
});
var E5_sitar = new Howl({
    src: ['./sitar_notes/E5.mp3']
});
```

```
var F5_sitar = new Howl({
    src: ['./sitar_notes/F5.mp3']
});
var G5_sitar = new Howl({
    src: ['./sitar_notes/G5.mp3']
});
var A5_sitar = new Howl({
    src: ['./sitar_notes/A5.mp3']
});
var B5_sitar = new Howl({
    src: ['./sitar_notes/B5.mp3']
});
var C6_sitar = new Howl({
    src: ['./sitar_notes/C6.mp3']
});


// Octave 1 for piano
var C3_piano = new Howl({
    src: ['./piano_notes/C3.mp3']
});
var D3_piano = new Howl({
    src: ['./piano_notes/D3.mp3']
});
var E3_piano = new Howl({
    src: ['./piano_notes/E3.mp3']
});
var F3_piano = new Howl({
    src: ['./piano_notes/F3.mp3']
});
var G3_piano = new Howl({
    src: ['./piano_notes/G3.mp3']
});
var A3_piano = new Howl({
    src: ['./piano_notes/A3.mp3']
});
```

```
var B3_piano = new Howl({
    src: ['./piano_notes/B3.mp3']
});
var C4_piano = new Howl({
    src: ['./piano_notes/C4.mp3']
});

// OCTAVE 2 FOR PIANO
var C4_piano = new Howl({
    src: ['./piano_notes/C4.mp3']
});
var D4_piano = new Howl({
    src: ['./piano_notes/D4.mp3']
});
var E4_piano = new Howl({
    src: ['./piano_notes/E4.mp3']
});
var F4_piano = new Howl({
    src: ['./piano_notes/F4.mp3']
});
var G4_piano = new Howl({
    src: ['./piano_notes/G4.mp3']
});
var A4_piano = new Howl({
    src: ['./piano_notes/A4.mp3']
});
var B4_piano = new Howl({
    src: ['./piano_notes/B4.mp3']
});
var C5_piano = new Howl({
    src: ['./piano_notes/C5.mp3']
});

// OCTAVE 3 FOR PIANO
var C5_piano = new Howl({
```

```
var C5_piano = new Howl({
    src: ['./piano_notes/C5.mp3']
});
var D5_piano = new Howl({
    src: ['./piano_notes/D5.mp3']
});
var E5_piano = new Howl({
    src: ['./piano_notes/E5.mp3']
});
var F5_piano = new Howl({
    src: ['./piano_notes/F5.mp3']
});
var G5_piano = new Howl({
    src: ['./piano_notes/G5.mp3']
});
var A5_piano = new Howl({
    src: ['./piano_notes/A5.mp3']
});
var B5_piano = new Howl({
    src: ['./piano_notes/B5.mp3']
});
var C6_piano = new Howl({
    src: ['./piano_notes/C6.mp3']
});
```

```
socket.onmessage = function(event) {

    const data = JSON.parse(event.data);
    console.log(data);


    let touchValue1 = data.touch1;
    let touchValue2 = data.touch2;
    let touchValue3 = data.touch3;
    let touchValue4 = data.touch4;
    let touchValue5 = data.touch5;
    let touchValue6 = data.touch6;
    let touchValue7 = data.touch7;
    let touchValue8 = data.touch8;

    console.log("touchvalue1: ", touchValue1);
    console.log("touchvalue2: ", touchValue2);
    console.log("touchvalue3: ", touchValue3);
    console.log("touchvalue4: ", touchValue4);
    console.log("touchvalue5: ", touchValue5);
    console.log("touchvalue6: ", touchValue6);
    console.log("touchvalue7: ", touchValue7);
    console.log("touchvalue8: ", touchValue8);

    let Distance1 = data.Distance1;
    console.log("Distance 1: ",Distance1);
    let Distance2 = data.Distance2;
    console.log("Distance 2: ",Distance2);

    let roll = data.roll;
    console.log("roll: ",roll);
```

```
socket.onmessage = function(event) {
    document.getElementById('sensorData1').textContent = `Sensor Data 1: ${touchValue1}`;
    document.getElementById('sensorData2').textContent = `Sensor Data 2: ${touchValue2}`;
    document.getElementById('sensorData3').textContent = `Sensor Data 3: ${touchValue3}`;
    document.getElementById('sensorData4').textContent = `Sensor Data 4: ${touchValue4}`;
    document.getElementById('sensorData5').textContent = `Sensor Data 5: ${touchValue5}`;
    document.getElementById('sensorData6').textContent = `Sensor Data 6: ${touchValue6}`;
    document.getElementById('sensorData7').textContent = `Sensor Data 7: ${touchValue7}`;
    document.getElementById('sensorData8').textContent = `Sensor Data 8: ${touchValue8}`;


    document.getElementById('Distance1').textContent = `Distance 1: ${Distance1} cm`;
    document.getElementById('Distance2').textContent = `Distance 2: ${Distance2} cm`;
    document.getElementById('roll').textContent = `Roll: ${roll} deg`;

    if (roll > -20 && roll < 20){
        if (Distance1 < 25){
            if (touchValue1 == 1){
                C3_sitar.play();
            }
            else if (touchValue2 == 1){
                D3_sitar.play();
            }
            else if (touchValue3 == 1){
                E3_sitar.play();
            }
            else if (touchValue4 == 1){
                F3_sitar.play();
            }
            else if (touchValue5 == 1){
                G3_sitar.play();
            }
            else if (touchValue6 == 1){
                A3_sitar.play();
            }
```

```
            else if (touchValue7 == 1){
                B3_sitar.play();
            }
            else if (touchValue8 == 1){
                C4_sitar.play();
            }
        }

        else if(Distance1 < 40){
            if (touchValue1 == true){
                C4_sitar.play();
            }
            else if (touchValue2 == 1){
                D4_sitar.play();
            }
            else if (touchValue3 == 1){
                E4_sitar.play();
            }
            else if (touchValue4 == 1){
                F4_sitar.play();
            }
            else if (touchValue5 == 1){
                G4_sitar.play();
            }
            else if (touchValue6 == 1){
                A4_sitar.play();
            }
            else if (touchValue7 == 1){
                B4_sitar.play();
            }
            else if (touchValue8 == 1){
                C5_sitar.play();
            }
        }

```

```
        else if (Distance1 < 60){
            if (touchValue1 == true){
                C5_sitar.play();
            }
            else if (touchValue2 == 1){
                D5_sitar.play();
            }
            else if (touchValue3 == 1){
                E5_sitar.play();
            }
            else if (touchValue4 == 1){
                F5_sitar.play();
            }
            else if (touchValue5 == 1){
                G5_sitar.play();
            }
            else if (touchValue6 == 1){
                A5_sitar.play();
            }
            else if (touchValue7 == 1){
                B5_sitar.play();
            }
            else if (touchValue8 == 1){
                C6_sitar.play();
            }
        }
    }
    else if ( roll > 80 && roll <= 100 ){

        if (Distance2 < 25){

            if (touchValue1 == 1){
                C3_piano.play();
            }
```

```
            else if (touchValue2 == 1){
                D3_piano.play();
            }
            else if (touchValue3 == 1){
                E3_piano.play();
            }
            else if (touchValue4 == 1){
                F3_piano.play();
            }
            else if (touchValue5 == 1){
                G3_piano.play();
            }
            else if (touchValue6 == 1){
                A3_piano.play();
            }
            else if (touchValue7 == 1){
                B3_piano.play();
            }
            else if (touchValue8 == 1){
                C4_piano.play();
            }
        }
        else if(Distance2 < 40){
            if (touchValue1 == true){
                C4_piano.play();
            }
            else if (touchValue2 == 1){
                D4_piano.play();
            }
            else if (touchValue3 == 1){
                E4_piano.play();
            }
            else if (touchValue4 == 1){
                F4_piano.play();
            }
```

```
            else if (touchValue5 == 1){
                G4_piano.play();
            }
            else if (touchValue6 == 1){
                A4_piano.play();
            }
            else if (touchValue7 == 1){
                B4_piano.play();
            }
            else if (touchValue8 == 1){
                C5_piano.play();
            }
        }
        else if (Distance2 < 60){
            if (touchValue1 == true){
                C5_piano.play();
            }
            else if (touchValue2 == 1){
                D5_piano.play();
            }
            else if (touchValue3 == 1){
                E5_piano.play();
            }
            else if (touchValue4 == 1){
                F5_piano.play();
            }
            else if (touchValue5 == 1){
                G5_piano.play();
            }
            else if (touchValue6 == 1){
                A5_piano.play();
            }
            else if (touchValue7 == 1){
                B5_piano.play();
            }
```

```
            else if (touchValue8 == 1){
                C6_piano.play();
            }
        }
    }
}
```

4.1.3 index.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="
    viewport" content="width=device-width, initial-scale=1.0">
    <title>Restructuring</title>
    <link rel="stylesheet" href="./style.css">
</head>
<body>

    <p id="sensorData1">Sensor data 1: 0</p>
    <audio id="audioElement1" preload="auto" allow="autoplay">
        <source id="sourceC" src="./C3.mp3" type="audio/mpeg">
        Your browser does not support the audio element.
    </audio>
    <button id="playButton1">Play Audio</button>



    <p id="Distance1">Ultrasonic sensor 1:</p>
    <p id="Distance2">Ultrasonic sensor 2:</p>
    <p id="roll">Gyroscope :</p>

    <p id="sensorData2">Sensor data 2: 0</p>
    <p id="sensorData3">Sensor data 3: 0</p>
    <p id="sensorData4">Sensor data 4: 0</p>
    <p id="sensorData5">Sensor data 5: 0</p>
    <p id="sensorData6">Sensor data 6: 0</p>
    <p id="sensorData7">Sensor data 7: 0</p>
    <p id="sensorData8">Sensor data 8: 0</p>


    <script type="module" src="./script.js"></script>
    <!-- <script type="module" src="./Notes.js"></script> -->
    <!-- <script type="module" src="./howler.core.js"></script> -->
    <script src="https://cdnjs.cloudflare.com/ajax/libs/howler/2.2.3/howler.min.js"></script>
</body>
</html>
```

Figure 4.5: Front-end

Chapter 5

Conclusion

5.1 Conclusion

In the study, the motion-based musical instrument project successfully combines hardware (ESP32 and touch sensors), communication (WebSocket), and software (backend music simulator and Web Audio API) to create an interactive musical experience. Users can trigger musical notes and sounds through physical gestures, offering an engaging and expressive form of musical performance. This project serves as an example of how technology and creativity can be blended to produce innovative and interactive musical instruments.

5.2 Future Scope

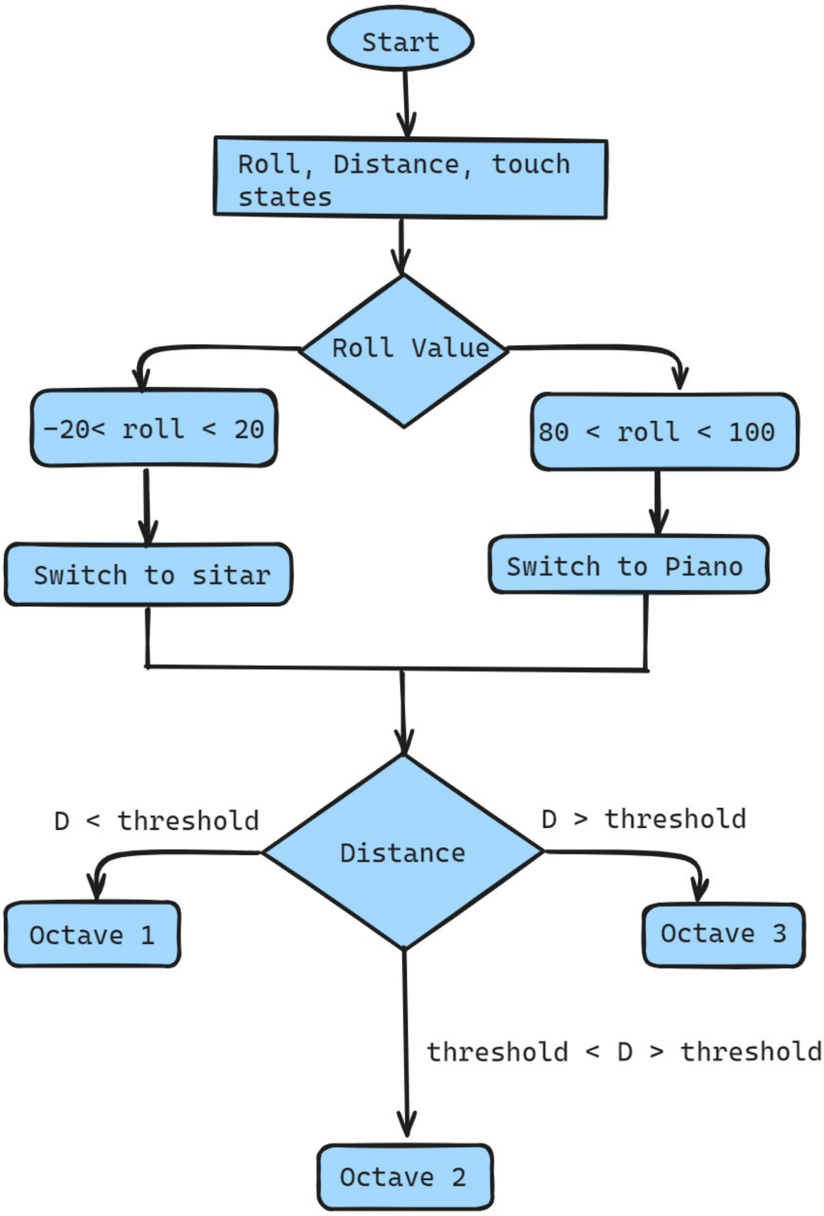
Future work will focus on increasing the accuracy of the model. Using the proposed model, a mobile-based application can be designed for embedding a wide range of instruments. Users can also be able to play more than one instrument at the same time in the case of chorus or simultaneously play the same chord of different scales. Such design can be integrated with a display that guides new users by prompting notes and can be used as a learning platform.

References

[1] S. Matsushita and F. Kamo, "Interactive Training System for Electric Guitar Strumming Form by Using Inertial Motion Sensors," *2022 IEEE 11th Global Conference on Consumer Electronics (GCCE), Osaka, Japan, 2022, pp. 97-100, doi: 10.1109/GCCE56475.2022.10014279.*

[2] K. Komatsu, F. Kamo and S. Matsushita, "Evaluating Strength of Piano Key Touch by Using a Wrist-Worn Inertial Motion Sensor," *2022 IEEE 11th Global Conference on Consumer Electronics (GCCE), Osaka, Japan, 2022, pp. 91-94, doi: 10.1109/GCCE56475.2022.10014314.*

Appendix-A: Flow Chart of Stage 1



Appendix-B: Time-Line Chart

WORK TASKS	JULY W1	JULY W2	JULY W3	JULY W4	AUGUST W5	AUGUST W6	AUGUST W7	AUGUST W8	SEPTEMBER W9	SEPTEMBER W10	SEPTEMBER W11	SEPTEMBER W12	OCTOBER W13	OCTOBER W14	OCTOBER W15	OCTOBER W16
1. PROBLEM DEFINITION																
Search for project topic	■	■														
Literature survey	■	■	■	■	■	■										
Identify the needs of project		■	■													
Identify the goals of project			■	■												
Give presentation					■											
Getting approval of the project					■											
Milestone: PROBLEM STATEMENT DEFINI																
2. PRE-DEVELOPMENT STAGE																
Preparing timeline chart						■	■									
Flowchart of the process								■								
Milestone: START OF THE PROJECT																
3. DEVELOPMENT STAGE																
Understanding software and hardware								■	■							
Study of touch sensor, Accelerometer & g									■	■	■					
operational feasibility											■					
Milestone: PROJECT SYNOPSIS																
4. DETERMINATION PHASE																
Input requirements												■				
Collection of components												■				
Project flow design													■	■		
Milestone: BASIS FOR IMPLEMENTATION																
5. DESIGNING																
Basic prototype implementation														■	■	
3D designing of Hand Held Device															■	■
Milestone: BEGIN BUILDING THE SYSTEM																

WORK TASKS / WEEK NO:	JANUARY				FEBRUARY				MARCH				APRIL			
	W17	W18	W19	W20	W21	W22	W23	W24	W25	W26	W27	W28	W29	W30	W31	W32
6. INTEGRATION OF SENSORS																
Mapping touch sensors	■															
Study of mpu6050		■														
Calibration and testing of mpu6050		■	■													
Merging mpu6050 with touch sensors			■	■												
7. FINAL CODE																
Testing hcsr04 sensor					■	■										
Merging readings with all sensors						■	■	■								
8. HARDWARE IMPLEMENTATION																
3D Printing									■	■						
Assembling components										■						
Connecting hardware										■	■	■				
Testing and Troubleshooting											■	■				
9. DOCUMENTATION																
Black book documentation													■	■	■	■