TECHNO ORCHESTRA
A TEAM ENVISAGE PROJECT

Authors:
Jeeva (AE17B029)
Ashutosh (ME17B001)
Lochana(CE18B003)
Karthiyalini(NA18B003)
Kumaran(MM18B024)
Madhav(EE18B070)
Abhishek(EE18B067)

April 20, 2020

Contents

0.1 INTRODUCTION

The aim of our project is to play multiple instruments at the same time, akin to an orchestra, using marbles.
A fully automated machine is used to achieve this. The xylophone and the drums are played by dropping marbles on them, which are controlled by solenoid valves. The guitar is played with the help of stepper motors.
The solenoid valves are controlled by the **Arduino UNO**, which contains information concerning the time to play each note, and the solenoid valves are switched on accordingly.
PCB designing was done to make the electrical connections easier. The designing was done using **Eagle** software.
The mechanical structure is one of the most important aspects of the project comprising the following structures

- To house the instruments
- The conveyor belt for moving the marbles
- Collectors and pathways for the marbles.

The movement of the marbles and the operation of the stepper motors proceed with the help of the **Arduino UNO** board. The musical notes are converted into binary with the help of python which is then feeded into the **Arduino**.
We were inspired the Marble Machine made by Wintergatan to make this project.

0.2 MECHANICAL MODULE

0.2.1 FUSION360)

The mechanism for xylophone, drums, guitar fret, marble holder and marble collector (for conveyor belt mechanism)and the ramp were designed using **Fusion360**. The exact dimensions were traced out in **Fusion**. The mechanism for guitar frets was designed based on the song requirements. These **AutoCAD** files were converted into **".stl"** files and were given for 3D printing.

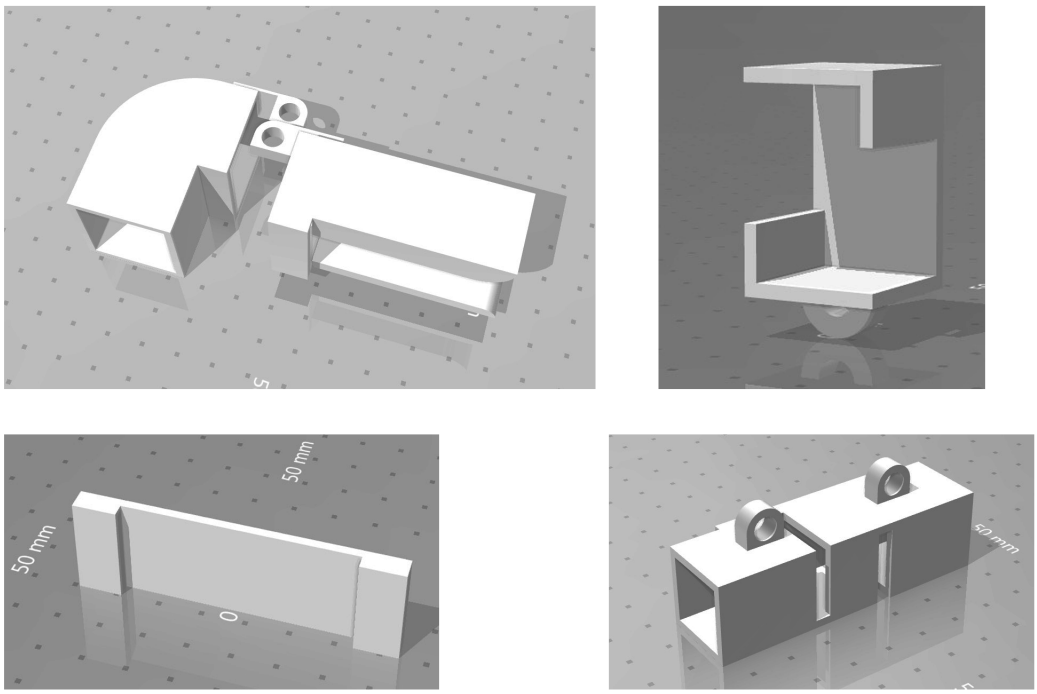

Figure 1: Marble Holder Components

These components(Refer fig 1) were fit into one another and used as a mechanism for controlling the flow of marbles as per the requirements of the song. The solenoid valves were connected to the mechanism, using the aluminium wire which was passed through the hoops provided. This mechanism

was adopted for both Xylophone as well as the Drums.

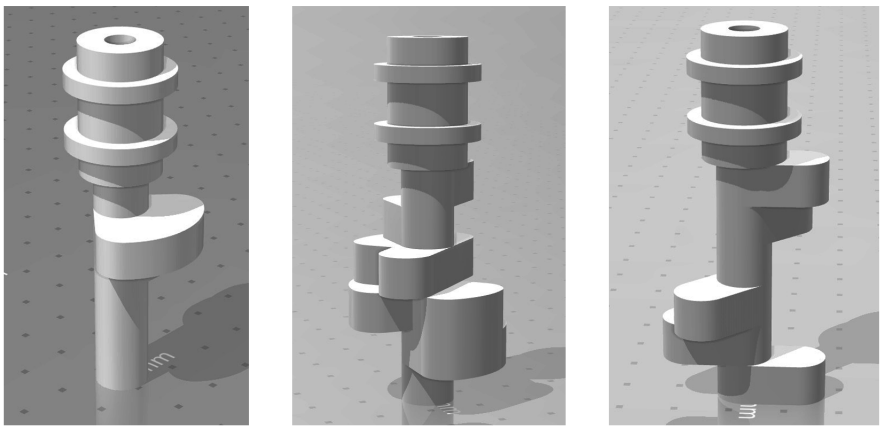

Figure 2: Stepper Motor Extension Components

The guitar was operated using three stepper motors(one for each fret) where the 3D printed components(Fig. 2) fitted to these motors were used for pressing the frets of the guitar. Only the first three frets were selected for operating as per the song requirements. The stepper motors where then operated by a code which depended on the songs characteristics(like chords to be played, number of strums,etc). The strumming was also automatic and this was achieved by an arm attached to a stepper motor which held a pick(plectrum). This structure wasn't too rigid to aid playing the guitar smoothly.

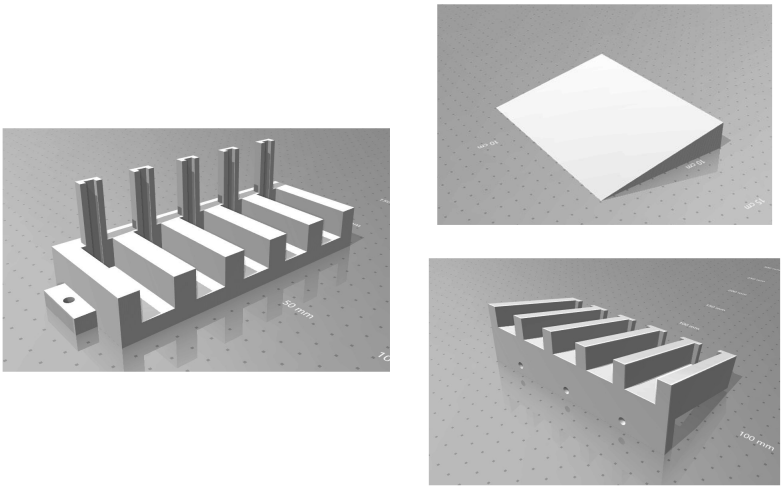

Figure 3: More 3D Printed Components

MARBLE COLLECTOR(Left figure): Which collects the marbles from the marble holder, as it continuously runs on the conveyor belt and transports them to the pipe, which further transports the marbles to the musical instruments.

MARBLE HOLDER(Right bottom figure): This is the one, where all the marbles stack up in a queue and are collected by the marble collector.It is designed in such a way that the marbles don't fall through any of the slits.

RAMP(Right top figure): This was later adopted as the marble stopped at the entry of the 3D printed component. Fixing this ramp made it easier for the marbles to climb onto the 3D component shown in fig.1 mechanism.

0.2.2 Deciding And Buying Materials

Initially, the entire mechanical structure was planned to be made from wood. But, due to a few errors, while cutting and joining wood, the marbles stopped in some places and the end result was not what we expected. So some parts, like the marble collector, marble holder, guitar fret cylinders and the ramp, were 3D printed for higher precision.
Three instruments were used in our project, namely the xylophone, guitar and drums.
The xylophone had 13 notes and we bought it online. There was a solenoid valve for each note.
The guitar we used was an electric guitar, which required an amplifier as well. As per our songs requirements,selected to simplify the guitar mechanism, we used only the first three frets.
Our drums consisted of three parts:-

- a cymbal,
- a snare drum
- a kick drum.

We used a box fitted with a contact microphone that mimics the sound of the kick drum instead of an original kick drum due to space constraints.
PVC pipes were used for transporting the marbles. For the conveyor belt mechanism, the first plan was to stick the marble collector to a timing belt. **But the number of the teeth on both sides didn't match and as a result, the collector started tilting after a few revolutions. So two cycle chains and two sprockets were used, but it wasn't effective as well since it was not easy to fix the marble collector to the cycle chains.**
The whole structure was mounted on aluminium extrusions so that it would be visible from a long distance.
We bought materials such as wood, **Araldite, Fevicol,** PVC and acrylic pipes, screws and nuts, clamps(L-clamps besides normal clamps) and **aluminium profiles(40*40mm)** as well as electrical components such as **Arduino UNO**, resistors, capacitors, diodes, transistors, motors, motor drivers, **RMC connectors**,multistrand wires. All electrical components were soldered to the **Printed Circuit Board(PCB)** with the help of soldering iron rod.
We bought materials from such shops as **Modi Electronics(in Ritchie Street), Om Sakthi Hardware and Wood(outside Velachery gate), Shanti Electronics(outside Velachery gate), Navarang Electricals**

and Hardware(outside Taramani gate), Mercy Electronics(near the Main Gate), Arihant Aluminum Shop (Parrys).

0.2.3 Making Of the Structure

The project required a lot of woodwork to be done. Initially since a rough model of the project was designed in **Fusion 360**, it was helpful in shaping various components, though the dimensions were later changed as per requirements. Woodwork included working with tools like jigsaw, drill and hacksaw.

0.2.4 Issues And Corrections

1. **CONVEYOR BELT:**
   Initially, the transportation system of marbles was ideated involving timing belts. But the marble collector was not stiff enough on the belts and tended to slip. Therefore, the cycle chain and sprockets mechanism was adopted.

2. **MECHANISM FOR CONTROLLING MARBLES WITH SOLENOID VALVES:**
   When the final structure for the drums was assembled and tested for the flow of marbles, it was found that the designed 3D printed component could pull off only 3 marbles without any effort and beyond that count, it failed to do so. A potential cause could be the solenoids not being that strong. Therefore, the idea of allowing only 3 marbles at a time by designing another gate was adopted. Also, the use of tiny 3D printed ramps was adopted, as there was inconvenience with regards to the marbles climbing onto the mechanism used for controlling their flow due to dimensional errors which occurred during the woodwork.

3. **3D PRINTING:**
   The 3D printing of the components was delayed.

4. **GUITAR:**
   Initially, the 3D printed parts for the guitar frets were thought of to be connected using the conveyor belt and all them being operated by using only a single stepper motor. This didn't seem to be working. Therefore, each fret led to be operated using a separate stepper motor.

0.3 ELECTRICAL MODULE

0.3.1 Circuitry

Controlling Solonoid Valves

Push-pull Solenoids are ideal for fast linear motion that can be switched between two states. **Small 12V Push-Pull Solenoids** have been used to control the gates for marble flow. A **transistor** and **diode** are required to control a solenoid.The transistor acts as a switch (when current is supplied to the base, the collector and ground gets shorted).

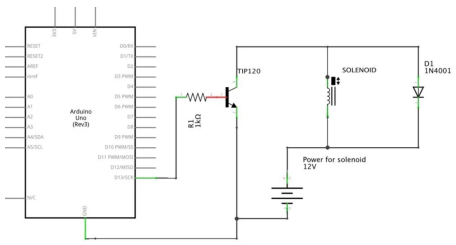

Figure 4: Circuit for controlling solonoid valves

Controlling Stepper Motor

Stepper motors are ideal for high torque and precise angle rotation applications. **12V NEMA 17 Stepper motor** has been used, which are controlled by **L298N motor drivers**. **Arduino** has a built-in library**(Stepper.h)** to control stepper motors.

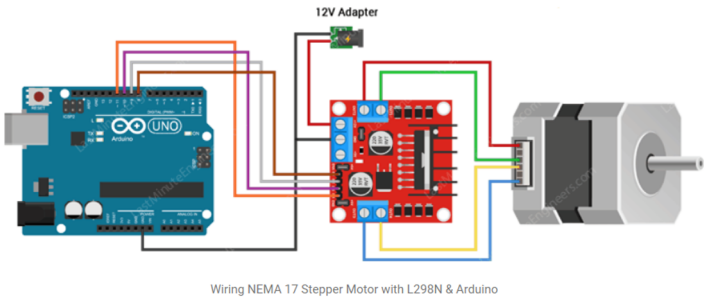

Figure 5: Arduino to Stepper Motor connections

Reading file from SD Card

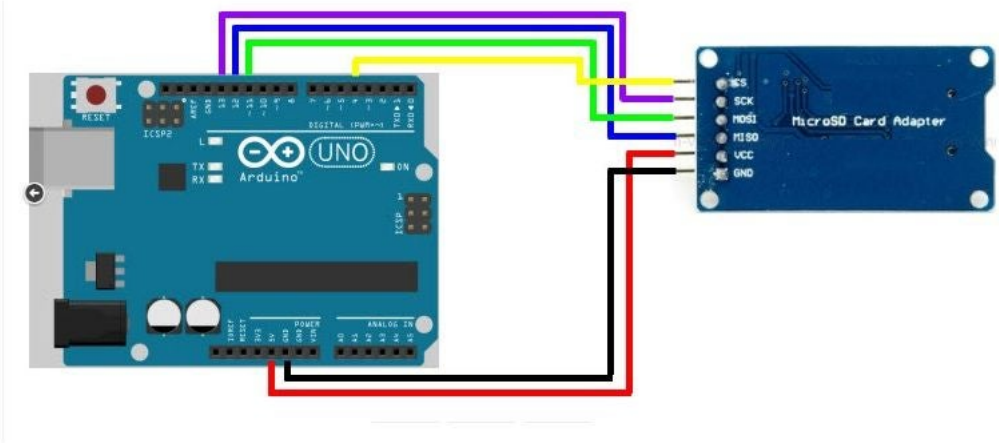

Figure 6: Circuit for reading file from SD Card

0.3.2 PCB Designing

Three **74595 shift registers(8-bit)** are chained to get a 24bit shift register, that can be controlled by a microcontroller like **Arduino Uno**. A Capacitor has been added between VCC and GND of ICs' to take care of fluctuations in voltages that may arise. RMC connectors have been used for external connections to PCB. 24 RMC connectors are used for 24 solenoids, and a 3 pin RMC connector is used for input to shift registers For every 6 solenoids, an **SMPS(Switch mode power supply)** has been used, which are further connected to PCB through the 4 corner RMC connectors

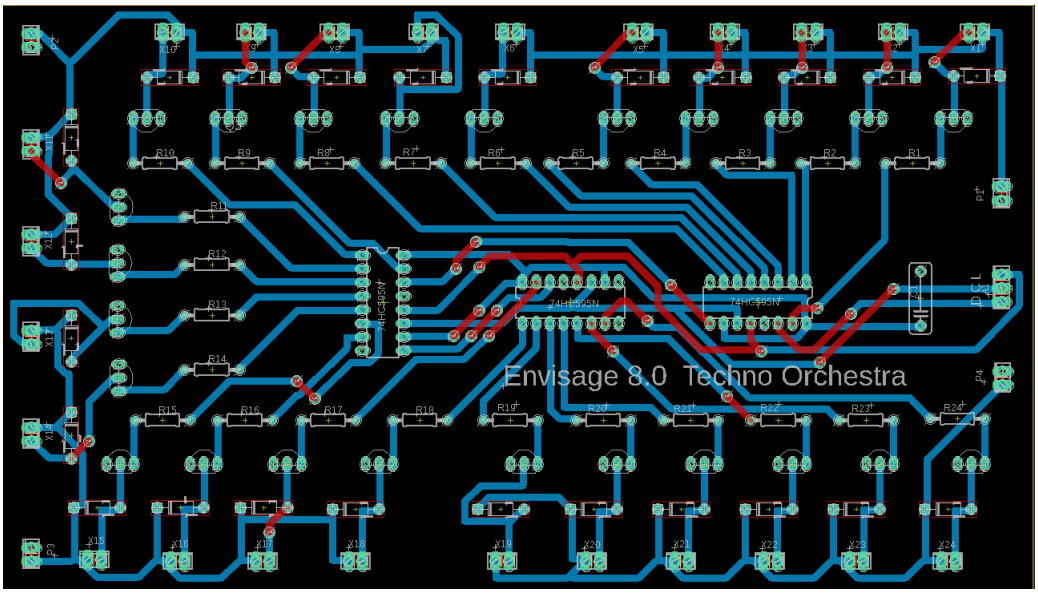

Figure 7: The detailed pcb design

0.3.3 Code

Python Code

Each note of guitar and drum has been labelled 1,2,3,etc. The input file consists of notes to be played against their timestamp The below code reads the file and encodes the notes in the required format and writes it to the output file.

```
#Python code to encode song notes

Xylo= {'e':16,'d':15,'c':14,'b':13,'a':12,'g':11,'F':10,
'E':9,'D':8,'C':7,'B':6,'A':5,'G':4}

Drum = {'S':1, 'K':2, 'H':3}

Guitar_notes = {'E': 0, 'e':1, 'G':2, 'D':3, 'A':4,
'a':5, 'C':6, 'F': 7}

tx = 300 # time delay for in ms xylophone
#(assumed constant for all notes)
td = 300 # time delay for in ms drums
#(assumed constant for all notes)
tf = 200 # max time delay in ms for stepper to attain
#correct position
tp = 1000
sotx = 400  # solenoid on time for xylophone in ms
sotd = 400  # solenoid on time for drums in ms

pcbxd = {}
```

```
22 strum = {}
23 chord = {}
24 count = 0
25 init_delay = 5000 # initial delay for song
26
27 def xylo(text):
28     s = 0
29     for char in text:
30         index = Xylo.get(char) - 1
31         s += 1<<index
32     return s
33
34 def drum(text):
35     s = 0
36     for char in text:
37         index = Drum.get(char) - 1
38         s += 1<<index
39     return s
40
41 def guitar(text):
42     return Guitar_notes.get(text)
43
44 fh = open("star wars.txt", "r")
45
46 mylist = fh.read().splitlines()
47
48 for line in mylist:
```

```
49      #print(line)
50      #print(type(line))
51      notes = line.split('\t')
52      print(notes)
53      ti = int(float(notes[0])*tp)#
54      if(notes[1] != ''):
55          print(ti + init_delay - tx,xylo(notes[1]))
56          if((ti + init_delay - tx) in pcbxd):
57              pcbxd[(ti + init_delay - tx)] += xylo(notes[1])
58          else:
59              pcbxd[(ti + init_delay - tx)] = xylo(notes[1])
60          if((ti + init_delay - tx + sotx) in pcbxd):
61              pcbxd[(ti + init_delay - tx + sotx)] += 0
62          else:
63              pcbxd[(ti + init_delay - tx + sotx)] = 0
64      if(notes[2] != ''):
65          print(ti + init_delay - td,drum(notes[2]))
66          if((ti + init_delay - td) in pcbxd):
67              pcbxd[(ti + init_delay - td)] += drum(notes[2])
68          else:
69              pcbxd[(ti + init_delay - td)] = drum(notes[2])
70          if((ti + init_delay - td + sotd) in pcbxd):
71              pcbxd[(ti + init_delay - td + sotd)] += 0
72          else:
73              pcbxd[(ti + init_delay - td + sotd)] = 0
74      if(notes[3] != ''):
75          chord[(ti + init_delay - tf)] = guitar(notes[3])
```

```
76      if(notes[4] != ''):
77          count += 1
78          strum[(ti + init_delay)] = count%2
79  fh.close()
80  print()
81  action = {0:[0,0,0]}
82
83  data1,data2,data3 = 0,0,0
84  for i in sorted(list(pcbxd.keys())+list(chord.keys())+
85  list(strum.keys())):
86      if(i in pcbxd):
87          data1 = pcbxd[i]
88      if(i in strum):
89          data2 = strum[i]
90      if(i in chord):
91          data3 = chord[i]
92      action[i] = [data1,data2,data3]
93
94  gh = open("envisage.txt","w")
95
96  for i in sorted(action):
97      print(str(i)+' '+str(action[i]))
98      gh.write(str(i)+'\n'+str(action[i][0])+'\n'+
99      str(action[i][1]*8+action[i][2])+'\n')
100
101 gh.close()
```

Arduino Code

Both solenoids and stepper motors need to be controlled parallely, so two arduinos, **a Master and a Slave** has been used to enable parallel processing Master arduino decodes the input file (stored in SD Card) and generates appropriate signals to solenoid valves It sends stepper motor encoded data to slave arduino through **I2C communication**, where it is decoded and generates appropriate signals to stepper motor.

```
Header files used:
include <SPI.h>
include <SD.h>
include<Wire.h>
define SLAVE_ADDRESS 9
```

```
#Master code to decode the input file (stored in SD card)

int CSpin = 4;
int latchPin = 5;
int clockPin = 6;
int dataPin = 7;
File myFile;
void updatepcb(long int data)
{
  byte a=0, b=0, c=0;
  c = lowByte(data);
  b = highByte(data);
  data = data>>8;
  a = highByte(data);
  Serial.print('[');
  Serial.print(a);
  Serial.print(' ');
```

```
18    Serial.print(b);
19    Serial.print(' ');
20    Serial.print(c);
21    Serial.print(']');
22    digitalWrite(latchPin, LOW);
23    shiftOut(dataPin, clockPin, MSBFIRST, a);
24    shiftOut(dataPin, clockPin, MSBFIRST, b);
25    shiftOut(dataPin, clockPin, MSBFIRST, c);
26    digitalWrite(latchPin, HIGH);
27  }
28  void updateguitar(int data)
29  {
30    Wire.beginTransmission(SLAVE_ADDRESS);
31    Wire.write(data);
32    Wire.endTransmission();
33  }
34  void setup() {
35    // put your setup code here, to run once:
36    pinMode(latchPin, OUTPUT);
37    pinMode(dataPin, OUTPUT);
38    pinMode(clockPin, OUTPUT);
39
40    long int activation_time;
41    long int current_time;
42    long int data1;
43    int data2;
44    Wire.begin(SLAVE_ADDRESS);
```

```
45
46    Serial.begin(9600);  # Open serial communications and
47    #wait for port to open:
48    while (!Serial)
49    {
50      ; # wait for serial port to connect.
51      #Needed for native USB port only
52    }
53
54    Serial.print("Initializing SD card...");
55    if (!SD.begin(4))
56    {
57      Serial.println("initialization failed!");
58      while (1);
59    }
60    Serial.println("initialization done.");
61    myFile = SD.open("envisage.txt");#open the file for reading:
62    if (myFile)
63    {
64      Serial.println('\n' "Reading from file:"); # read from the
65      #file until there's nothing left
66      long int starting_time = millis();
67      activation_time = myFile.readStringUntil('\n').toInt();
68      while (myFile.available())
69      {
70        current_time = millis() - starting_time;
71        if(millis()-starting_time >= activation_time)
```

```
72          {
73            Serial.print(activation_time);
74            Serial.print(" : ");
75            data1 = myFile.readStringUntil('\n').toInt();
76            Serial.print(data1);
77            updatepcb(data1);
78            Serial.print(',');
79            data2 = myFile.readStringUntil('\n').toInt();
80            Serial.print(data2);
81            updateguitar(data2);
82            activation_time = myFile.readStringUntil('\n').toInt();
83            Serial.println();
84          }
85        }
86        myFile.close();
87      }
88      else
89      {
90        Serial.println("Error opening file"); # if the file
91        #didn't open, print an error:
92      }
93    }
94    void loop() {
95
96    }
```

slave code to control the stepper motors
Header files required:

```
#include<Stepper.h>
#include<Wire.h>
#define SLAVE_ADDRESS 9
```

```
const int stepsPerRevolution = 200;

int strum_steps = 50; # no. of steps to strum once
int frets_steps = 25; # no. of steps to move frets to
#immediate next chord
int curr_frets_state = 0;
int curr_strum_state = 0;
int next_state=0;
int change_strum;
int change_frets;
int next_frets_state;
int next_strum_state;

Stepper Frets(200, 7, 6, 5, 4);
Stepper Strum(200, 11, 10, 9, 8);
int count=0;

void setup()
{
  // put your setup code here, to run once:
  Wire.begin(SLAVE_ADDRESS);
  Wire.onReceive(updatestate);

  Serial.begin(9600);  # Open serial communications
```

```
25    #and wait for port to open:
26    while (!Serial)
27    {
28      ; #wait for serial port to connect.
29      #Needed for native USB port only
30    }
31    Serial.println("Started ");
32
33    Strum.setSpeed(100);
34    Frets.setSpeed(240);
35  }
36  void updatestate(int b)
37  {
38    next_state = Wire.read();
39    #Serial.print("Next state ");
40    #USerial.println(next_state);
41  }
42  void updateguitar()
43  {
44    next_frets_state = next_state & 7;
45    next_strum_state = next_state >> 3;
46    change_frets = next_frets_state - curr_frets_state;
47    change_frets =
48    change_frets > 4 ? 200 - change_frets*25 : change_frets*25;
49    Frets.step(change_frets);
50    curr_frets_state = next_frets_state;
51    change_strum = (next_strum_state - curr_strum_state) * 25;
```

```
52    Strum.step(change_strum);
53    curr_strum_state = next_strum_state;
54  }
55  void loop()
56  {
57    # put your main code here, to run repeatedly:
58    updateguitar();
59    digitalWrite(4,LOW);
60    digitalWrite(5,LOW);
61    digitalWrite(6,LOW);
62    digitalWrite(7,LOW);
63    digitalWrite(8,LOW);
64    digitalWrite(9,LOW);
65    digitalWrite(10,LOW);
66    digitalWrite(11,LOW);
67  }
```

0.3.4 Issues faced

(a) **Heating of solenoid:** Solenoids could get hot due to improper connections causing a short-circuit. Before connecting to power supply it has to be made sure that every connection has been insulated properly.

(b) **Heating of stepper motor:** It's quite usual for stepper motor to get heated, One of the precautions taken was to cut off power to driver (by setting all 4 inputs to driver to low) whenever motor was not in use. Another was to limit the operation of motor to less than 2 mins so hence limiting the rise of temperature.

0.4 FUTURE PLANS

Because of a few errors in the previous mechanisms, we couldn't present the project during the show.We would like to continue with the project, rectifying our mistakes. We would like to come up with new mechanisms for most parts of the project. We would like to present our project in Envisage 9.0 and in various other platforms.

0.5 ACKNOWLEDGEMENTS

Firstly, we would like to thank our cores Tejas and Gautam, who constantly motivated us till the very end.Every minute we were fabulously guided by our supercoords Jeeva and Ashutosh.They helped us with many new mechanisms for the project, which were also useful to improve our technical skills.We would also like to thank Pravallika, the CoCAS for being with us, till the end of the show and giving us her valuable suggestions regarding the project presentation during the show.Throughout our journey, we also sought help and knowledge from our fellow supercoords and coords.We would like to thank Central Workshop staff Kumar Sir and 3D printing club for their help in providing us with numerous 3D printing components.We would be very thankful to all of them without whom our project wouldn't have taken shape.

0.6 APPENDIX A:Coding the songs

Now that we've built the structure, designed the printed circuit boards and got the code down for operating this machine, we come to the last part of the project,which is enabling the marble machine to play music by which we mean songs, since one man's noise is another's music.
To this avail, songs are to be broken down into notes on the corresponding instruments with time stamps on when the notes are to be played.
Before advancing further,here's some basic western music theory.
Basic Music Theory:
Just like atoms are(were) like the fundamental building blocks of matter, musical notes are like the fundamental building blocks of songs. You might have heard the following counterparts of musical notes.
Indian Classical: Sa Re Ga Ma Pa Dha Ni (Sa), the seven 'svaras' or the musical building blocks of indian classical music or Do Re Mi Fa So La Ti (Do), which are like the counterparts of the svaras ,these are how the notes "sound" when you play them on an instrument.
Now, coming to the notes itself, there are 7 notes ,that we're concerned about in this project.
They're C ,D ,E ,F ,G ,A ,B. Where C corresponds to 'Do' and B corresponds to 'Ti'.
Advancing to the instruments, there are three (types) instruments we're using, namely:
the xylaphone
a snare drum, kick drum, high hat
an electric guitar
Now that we know what music notes are, a look at our xylaphone will tell you that it comprises 13 wooden bars ,each corresponding to a different musical note.
A few of these notes will repeat,six actually, these are from a different 'octave'. These are the same type of sounds with its frequency doubled as in, a note with a frequency of 220hz will have a frequency of 440 hz in the next octave.
So our task for xylaphone is now rephrased as follows,
-¿ Choose a song which uses a variation of these notes.
-¿ Note down the timestamps at which these notes occur along with the corresponding note

We do something similar as far as the drums are concerned. We break down the drums segment of the song into the three types of drums we're using and then note the timestamps at which these beats occur along with the corresponding drum played.
Now to the guitar. We are playing chords on the guitar,where chords are a specific combination of notes played together.
The guitar mechanism in our design can currently play the following 8 chords:
A,Aminor,F,D,C,E,Eminor,G.
Now we break down the song in terms of these chords and then note the timestamps at which the chords are played and what chord is being played.

0.7 Appendix B: Some pictures

Figure 8: The outlet through the marbles start coming

Figure 9: The collector

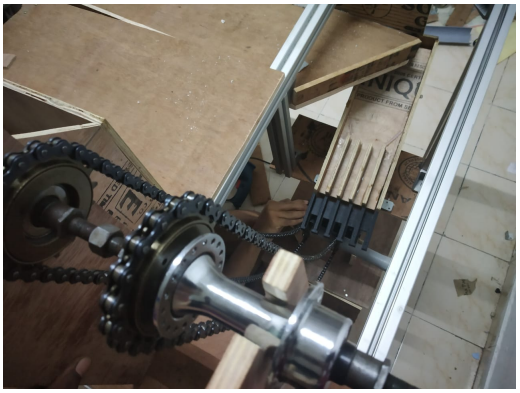

Figure 10: The conveyor belt mechanism

Figure 11: The 3D printed flaps aligned for the xylaphone

Figure 12: The team