SPECTULATION DOCUMENT
Laser Harp

Description:

A laser harp with a number of lasers with detectors to detect the blocking of each laser, according to which pre-defined musical tones would be played. The functionality can be extended to playing any note by modifying the MIDI output of the program.

Setting Up Arduino & Processing IDE:

Here are the details about how we got set up the environment:
Arduino IDE Setup: 1. Go to this link https://www.arduino.cc/en/Main/Software and download Arduino IDE according to your system.
2. Extract the zip file wherever you want.
3. Open the extracted folder and run Arduino.exe.
4. Your Arduino IDE is ready to use.

Working:

A laser harp is a musical instrument that produces sound by interrupting laser beams. It consists of multiple lasers, detectors, and a speaker to generate the desired notes. Here's a step-by-step explanation of how it works:

Laser Setup: The laser harp typically uses multiple lasers arranged in a vertical plane. In this case, there are five lasers, each pointing downward and positioned at different heights.

Laser Beam Emission: When the lasers are activated, they emit continuous laser beams toward the detector area. These beams act as virtual strings or beams of light that can be "plucked" or interrupted to produce sound.

Detector Setup: Corresponding to each laser, there is a detector placed opposite it. These detectors are typically photodiodes or phototransistors capable of detecting interruptions in the laser beams.

Interruption Detection: As the laser beams travel towards the detector area, they are continuously monitored by the detectors. When a laser beam is interrupted, such as when a hand or an object comes between the laser and the detector, the detector senses a decrease in the amount of light received.

Signal Generation: The interruptions detected by the detectors generate electrical signals. Each detector produces a separate signal when its corresponding laser beam is interrupted. The signals generated by the detectors are then processed to produce the desired musical notes.

Square Wave Generation: To create the square wave required for audio output, the electrical signals from the detectors are typically fed into a microcontroller or a digital signal processing unit. The microcontroller processes these signals and converts them into square waves.

Note Mapping: The microcontroller maps each laser to a specific musical note. When a laser beam is interrupted, the corresponding electrical signal is converted into a square wave with the frequency associated with that note. The duration of the interruption determines the duration of the square wave, which affects the note's sustain.

Audio Output: The square waves generated by the microcontroller are then sent to a speaker or an audio system capable of producing sound. The speaker converts the square waves into audible sound waves, which allows the player to hear the notes being played.

By interrupting the laser beams at different positions and durations, the player can play different notes and create melodies on the laser harp. The square wave generation and note mapping processes ensure that the desired musical notes are produced, providing a unique and visually impressive musical experience.

This was how the project was intended to go but due to the unavailability of parts, the lasers were simulated by switches in tinkercad which gave a +5V response when the laser was unblocked and a 0V response when the laser was blocked. The speaker could not be simulated, so an oscilloscope was used to show the square wave along with the MIDI outputs being printed on the Serial Monitor of the Arduino.

MIDI(MUSICAL INSTRUMENT DIGITAL INTERFACE)

MIDI, which stands for Musical Instrument Digital Interface, is a protocol that allows electronic musical instruments, computers, and other devices to communicate and control one another. It was introduced in the early 1980s and has become the industry standard for connecting and controlling musical devices. MIDI messages can carry various types of information, such as note data, control changes, pitch bend, and more.

Here are some example MIDI commands:

Note On: This command is used to trigger the start of a note. It includes information about the note number (pitch) and the velocity (how forcefully the note is played). Example: Note On - Channel 1, Note Number 60 (C4), Velocity 100.

Note Off: This command is used to indicate the release of a note that was previously triggered. It also includes the note number and velocity. Example: Note Off - Channel 1, Note Number 60 (C4), Velocity 0.

Control Change: This command is used to change parameters or control settings on a device. It includes a control number and a value. Example: Control Change - Channel 1, Control Number 7 (Volume), Value 127.

Program Change: This command is used to change the instrument or patch on a receiving device. It includes a program number. Example: Program Change - Channel 1, Program Number 5 (Electric Piano).

Pitch Bend: This command is used to apply pitch modulation to a note. It includes a value that determines the pitch bend amount. Example: Pitch Bend - Channel 1, Value 8192.

These are just a few examples of MIDI commands, and there are many more available for various purposes, such as channel mode messages, system exclusive messages, and more.

For more detailed information and references on MIDI, you can refer to the following resources:

MIDI Manufacturers Association (MMA): The official website of the organization that oversees the MIDI standard. It provides technical specifications, FAQs, and resources related to MIDI. Website: https://www.midi.org/

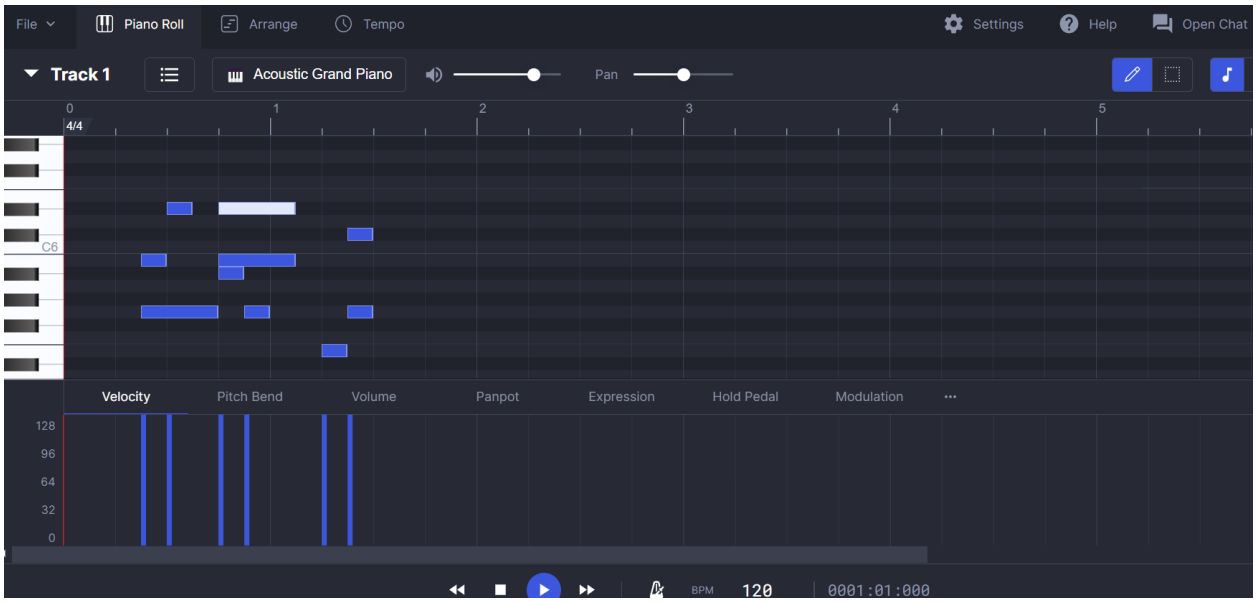
MIDI Association: A community-driven website with tutorials, articles, forums, and resources for MIDI enthusiasts and users. Website: https://www.midi.org/midi-articles

MIDI 1.0 Detailed Specification: The official MIDI 1.0 specification document that provides in-depth information about MIDI messages, data format, and protocols. PDF: https://www.midi.org/specifications-old/item/the-midi-1-0-specification

MIDI has been used in this project to extend the range of the audio output to play any instrument possible. It provides flexibility to the project

```
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
4D 54 68 64 00 00 00 06 00 01 00 02 00 60 4D 54
72 6B 00 00 00 13 00 FF 58 04 04 02 18 08 00 FF
51 03 06 8A 1B 00 FF 2F 00 4D 54 72 6B 00 00 00
2C 00 90 47 7F 30 80 47 00 00 90 47 7F 30 80 47
00 00 90 47 7F 30 80 47 00 00 90 47 7F 30 80 47
00 00 90 47 7F 30 80 47 00 00 FF 2F 00
```

A typical MIDI file



An example of a Digital Audio Workstation that can be used to play MIDI files

ARDUINO CODE:

```
void setup() {

  for (int i=2; i<=6; i++)
  {
    pinMode (i, INPUT);
```

```
  }
  pinMode (7,OUTPUT);   // pin 8 for speaker output
  Serial.begin(9600);
}

  int m,n,o,p,q;      // pin 2,3,4,5,6 are taking input from laser sensors
  int c[8]={956, 851, 758, 716, 638, 568, 506, 478};    //defining semi time
periods for various notes
int note7 = 0x64;
int note6 = 0x66;
int note5 = 0x68;
int note4 = 0x70;
int note3 = 0x71;
int note2 = 0x40;
int note1 = 0x47;
int note8 = 0x63;

  int d0=0;
  int d1=1;
  int s=7;
  int dis=8;
int cmd0=0;
  int pitch0=0;
  int velocity0=0;
int cmd00=0;
  int pitch00=0;
  int velocity00=0;
int prevnote=note1;

void loop() {

  m=digitalRead(2);
  n=digitalRead(3);
  o=digitalRead(4);
  p=digitalRead(5);
```

```
q=digitalRead(6);

if (m==d0 && n==d1 && o==d1 && p==d1 && q==d1)
{
  noteOff(0x80,prevnote,0x00);
  noteOn(0x90, note1, 0x7F);
  prevnote=note1;
  sound(0);
}
else if (m==d0 && n==d0 && o==d1 && p==d1 && q==d1)
{
  sound(1);
  noteOff(0x80,prevnote,0x00);
  noteOn(0x90, note2, 0x7F);
  prevnote=note2;
}
else if (m==d1 && n==d0 && o==d1 && p==d1 && q==d1)
{
  sound(2);
  noteOff(0x80,prevnote,0x00);
  noteOn(0x90, note3, 0x7F);
  prevnote=note3;
}
else if (m==d1 && n==d0 && o==d0 && p==d1 && q==d1)
{
  sound(3);
  noteOff(0x80,prevnote,0x00);
  noteOn(0x90, note4, 0x7F);
  prevnote=note4;
}
else if (m==d1 && n==d1 && o==d0 && p==d1 && q==d1)
{
  sound(4);
  noteOff(0x80,prevnote,0x00);
  noteOn(0x90, note5, 0x7F);
```

```
    prevnote=note5;
  }
  else if (m==d1 && n==d1 && o==d0 && p==d0 && q==d1)
  {
    sound(5);
    noteOff(0x80,prevnote,0x00);
    noteOn(0x90, note6, 0x7F);
    prevnote=note6;
  }
  else if (m==d1 && n==d1 && o==d1 && p==d0 && q==d1)
  {
    sound(6);
    noteOff(0x80,prevnote,0x00);
    noteOn(0x90, note7, 0x7F);
    prevnote=note7;
  }
  else if (m==d1 && n==d1 && o==d1 && p==d0 && q==d0)
  {
    noteOff(0x80,prevnote,0x00);
    sound(7);
    noteOn(0x90, note8, 0x7F);
    prevnote=note8;
  }
}

long readUltrasonicDistance(int triggerPin, int echoPin)
{
  pinMode(triggerPin, OUTPUT);  // Clear the trigger
  digitalWrite(triggerPin, LOW);
  delayMicroseconds(2);
  // Sets the trigger pin to HIGH state for 10 microseconds
  digitalWrite(triggerPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(triggerPin, LOW);
  pinMode(echoPin, INPUT);
```

```
  // Reads the echo pin, and returns the sound wave travel time in
microseconds
  return pulseIn(echoPin, HIGH);
}

void sound(int z)
{
  int x,y;

  x=readUltrasonicDistance(dis,dis);
  y= c[z]+c[z+1]*(x-162)*0.0000518;

  for(int i=0;i<100;i++)
  {
    digitalWrite(s,HIGH);
    delayMicroseconds(y);
    digitalWrite(s,LOW);
    delayMicroseconds(y);
  }
  delay(100);
}
void noteOn(int cmd, int pitch, int velocity)
{

  if(cmd!=cmd0||pitch!=pitch0||velocity!=velocity0)
  {
        Serial.write(cmd);
        Serial.write(pitch);
        Serial.write(velocity);
  }
   cmd0=cmd;
   pitch0=pitch;
  velocity0=velocity;

}
```

```
void noteOff(int cmd, int pitch, int velocity)
{

  if(cmd!=cmd00||pitch!=pitch00||velocity!=velocity00)
  {
        Serial.write(cmd);
        Serial.write(pitch);
        Serial.write(velocity);
  }
   cmd00=cmd;
   pitch00=pitch;
  velocity00=velocity;

}
```

OUR TEAM:

- **AMAL NARANG (2y)**
- **YASHVARDHAN SAINI(2y)**