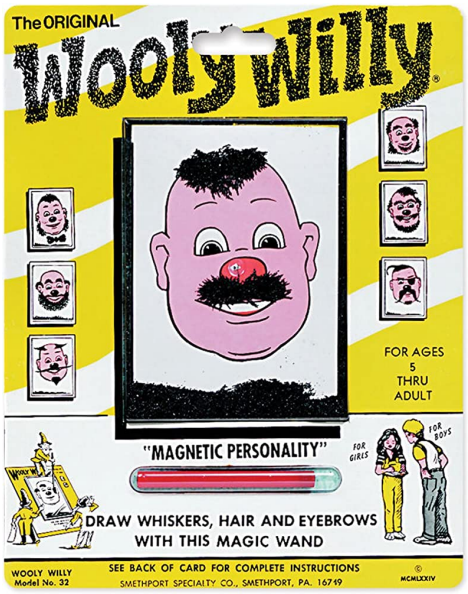
Annika Schaad

The Ferrous Wheel

Design Goals

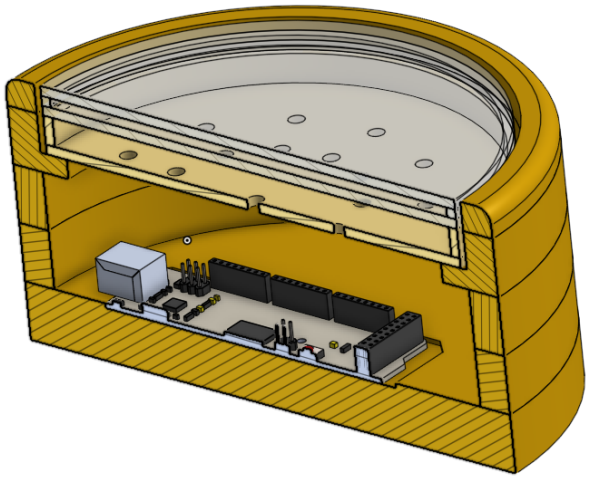
The goal of this project was to make the intuitive act of drawing into a musical instrument. We took inspiration from the mechanism of the Willy Wooly children's toy (below). The toy is made from inexpensive materials; iron filings are contained within a plastic enclosure, moved by the user with a magnetic wand. The "hairy" appearance of iron filings within a magnetic field is used by positioning the drawing space in front of a picture of a man's face. When brainstorming for this project, we discussed ways of turning a painting motion into control for a MIDI instrument. We landed on this sort of "drawing" because it is easily reset, and clear targets can be laid out below the drawing surface.



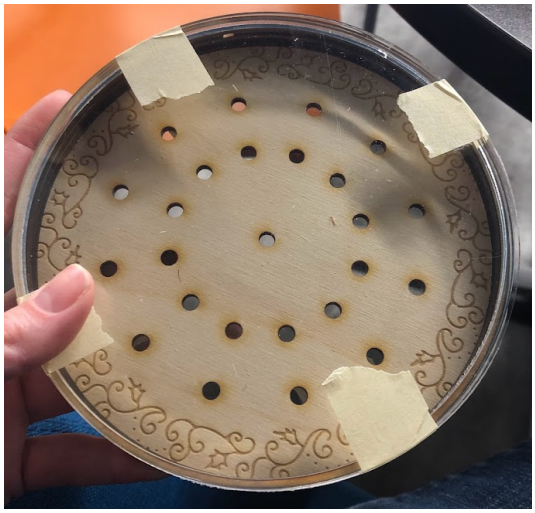
The instrument is composed of a network of photoresistors, laid out in the pattern of the circle of fifths, below a clear acrylic enclosure containing magnetic filings. The filings are moved using one of six magnetic wands. One of three sizes of magnets is attached to each wand. Half of the wands have the same polarity, such that there are two wands per size of magnet, with the magnets in opposing orientations. This allows the user to experiment with how much of the filings they are moving at once, as well as interactions between the magnetic fields of different wands. Either a note, chord, or sample, depending on the mode of the instrument, is played when a particular photoresistor is covered. The pitch of the note played is determined by the position of the covered sensor in the circle of fifths. The use of magnets, iron filings and photoresistors provide possibilities for future expansion of the instrument using the analog nature of the photoresistor or more direct measurement of the magnetic field.

Construction

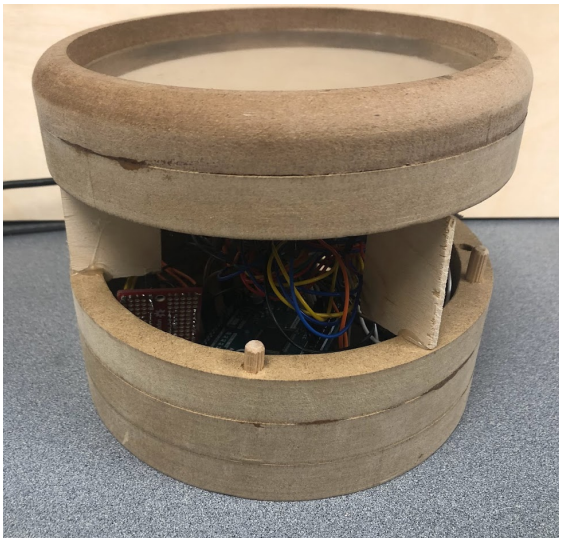
As a whole, this instrument was designed for easy assembly and machinability. It was designed to be assembled from cut pieces of flat material that could then be stacked for a robust, three-dimensional object. This instrument has two major components in its design. The inner section of the instrument is composed of the sensor plate and the housing for magnetic filings, as well as a spacer. The outer section is a case, which supports the inner section while also giving the user a larger interface and housing the arduino that collects the sensor data. The original design for the instrument is pictured to the right.



The inner section was machined using a laser cutter. The wooden sensor plate was engraved with a decorative border and 25 holes for the 25 photoresistors, arranged in two concentric circles of 8 with an additional sensor in the center. The photoresistors are attached to the bottom of this plate with the head of the photoresistor flush with the top of the plate, and secured in place with tape. There are three stacked spacer rings of ⅛ inch birch plywood between this plate and the acrylic layer above. The acrylic plate has a spacer ring between two solid layers of clear ⅛ inch acrylic. The three layers of acrylic are sealed with two-part epoxy, with iron filings between them. The wood pieces are attached to each other, and the acrylic pieces are attached to each other, but within the case the acrylic is resting atop the wooden piece for easy troubleshooting.



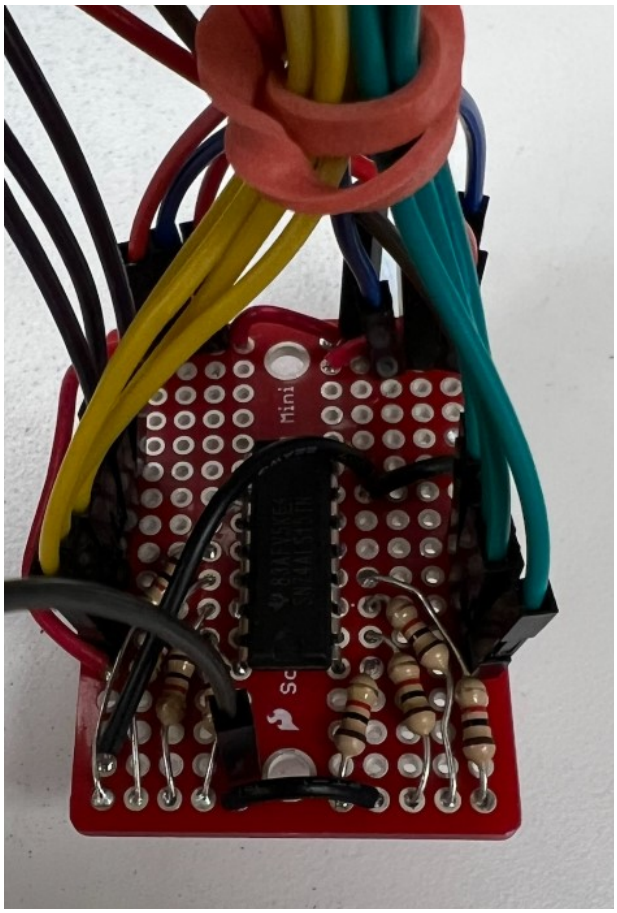
The outermost case was cut from medium-density fiberboard (MDF) using the Shopbot CNC router in Bray PALLS machine shop. The case could have been cut using the X-carve router in the EMID Lab as well, but I am more familiar with the Shopbot. Five rings were cut from 0.75” thick MDF, then were glued in place. The outer case was made assuming the sensors would take up approximately two inches of vertical space; towards the end of our project, it became clear that we would need more clearance. I fashioned four supports from ⅛ inch plywood, secured



them to the bottom half of the case with hot glue, and added a thin paperboard cover to keep everything contained (above). Originally, the two halves of the outermost case were attached via wooden pegs, which can be seen attached to the bottom half of the case, and slotted into holes in the top half.

The magnetic wands were designed using Adobe Illustrator and Onshape. They were printed from PLA using Bray’s Ender 3 3D printers. I fashioned a stand for them using scrap MDF from my earlier cuts as well as scrap foam from Bray. The magnets were attached using small pieces of foam as spacers and two-part epoxy. The foam is attached to the outer ring with hot glue. Three sizes of magnets were selected, and two of each size were attached to wands with opposing polarity.

Circuitry and Sensors



The circuitry for this project was overseen by Joe. This instrument contains 25 photoresistors as the primary sensors. Other than mode switching within the MAX patch, the user has no other direct controls of the instrument during normal operation. Since we want analog information from these sensors, and the Arduino Mega has only 16 analog pins, the photoresistors are wired through a multiplexer rather than sending information directly to the board. Each multiplexer chip can read on up to eight pins. The multiplexer chips can rapidly cycle through their pins so that the Arduino receives data from all 24 photoresistors, plus one that is plugged in directly, in a time frame conducive to the user making changes with the filings. To the right is a view of a protoboard that holds one multiplexer, eight 1.6 kΩ resistors, and the jumper wires for the photoresistors and connections to the Arduino Mega. Three of these circuits are contained within the instrument.

Arduino

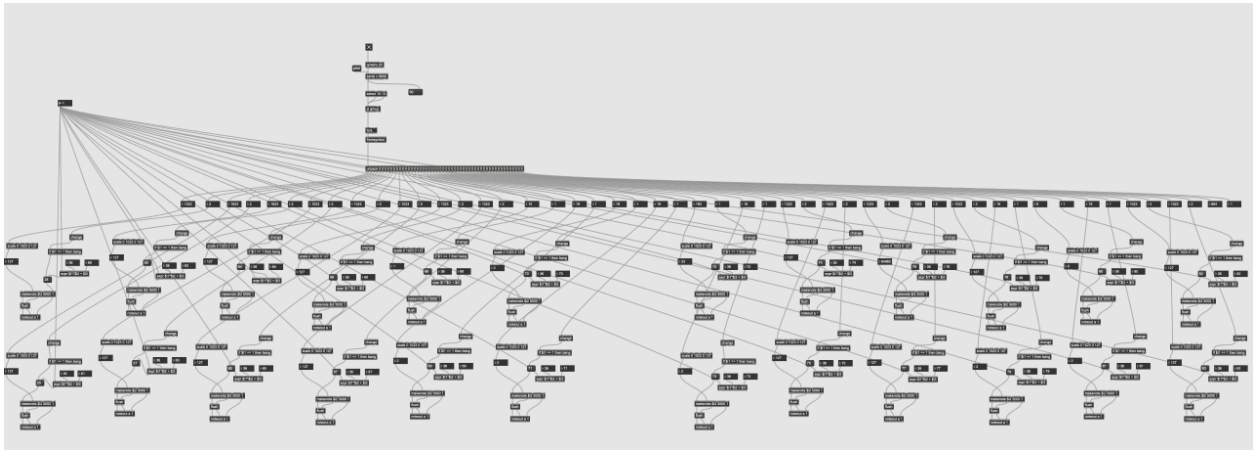
The arduino script for this project was overseen by Joe, and was built off the script we used for our midterm project. The Arduino code for this project has four main components. First, it cycles through the pins on the multiplexers. Second, it transforms the analog values from the multiplexer to a digital high or low signal. If the sensor value is above our threshold of 750 (approximately the halfway point of our photoresistors' range) then it is set to our "high" of 1023. This was done because the range of values we were getting for the sensors was difficult to work with, so we decided to switch to making them on or off. Third, the code sends an on and off signal to our Max patch depending on whether the sensor is reading high or low. It does this by writing the sensor values, as well as an on/off command, to the serial monitor. This serial

monitor is then read into Max. The lines of code for one sensor value to be sent are pasted below, and the code in its entirety is attached as appendix A.

```
int tolerance = 750;
if (sensor4 > tolerance) {
      sensor4 = 1023;
if (sensor4 < 1023 && sensor4_on != 1) {
      sensor4_on = 1;
}
if (sensor4 == 1023 && sensor4_on == 1) {
      sensor4_on = 2;
}
Serial.print(sensor4);
Serial.print(" ");
Serial.print(sensor4_on);
Serial.print(" ");
```

Max Patch

The MAX Patch for this project was overseen by Joe, and was built off the patch we used for our midterm project. First, the serial monitor is read in from Arduino and unpacked. Then, the sensor value and the on/off signal are sent as MIDI control signals to Reason. Each sensor corresponds to a different note, and the mode is changed by changing the octave. Each mode is two octaves wide. The note assignments can be found in appendix B.



Reason

Jordan headed up the reason patch for this project. She created the three modes of note, chord, and sampler by recording all of the played sounds into different octaves of an NN-19 sampler module. This allowed the MAX patch to be simplified, as changing modes is as straightforward as changing octaves. The Reason rack is pictured below. We decided to lay out the instrument's sensors in the circle of fifths in hopes that the music layed would be less dissonant than if the notes were arranged in ascending order of a scale, since it is difficult to trigger only one sensor at a time in this instrument.

Future Work

In line with our original vision, it would be nice in the future to more fully utilize the analog readings from the photoresistors. The timbre of the played sound could be affected by how completely the sensor is covered by means of a filter envelope. The volume of the played sounds could be determined by the coverage of the centermost photoresistor. Additionally, it would be beneficial to implement a way of switching modes of the instrument from the body of the instrument itself– either by cycling through with a button, multiple buttons, or a 3-position switch. This would allow the user to concentrate on playing the instrument rather than going between the software and the instrument.

Appendices

Appendix A: Arduino Code.

```
void setup() {
  Serial.begin(9600);

  pinMode(A1,INPUT);
  pinMode(A2,INPUT);
  pinMode(A3,INPUT);
  pinMode(A4,INPUT);

//mux1
  pinMode(2,OUTPUT);
  pinMode(3,OUTPUT);
  pinMode(4,OUTPUT);

//mux2
  pinMode(5,OUTPUT);
  pinMode(6,OUTPUT);
  pinMode(7,OUTPUT);

//mux 3
  pinMode(8,OUTPUT);
  pinMode(9,OUTPUT);
  pinMode(10,OUTPUT);
}

  int sensor0_on = 0;
  int sensor1_on = 0;
  int sensor2_on = 0;
  int sensor3_on = 0;
  int sensor4_on = 0;
  int sensor5_on = 0;
  int sensor6_on = 0;
  int sensor7_on = 0;
  int sensor8_on = 0;
  int sensor9_on = 0;
  int sensor10_on = 0;
  int sensor11_on = 0;
  int sensor12_on = 0;
  int sensor13_on = 0;
  int sensor14_on = 0;
  int sensor15_on = 0;
  int sensor16_on = 0;
  int sensor17_on = 0;
  int sensor18_on = 0;
  int sensor19_on = 0;
  int sensor20_on = 0;
  int sensor21_on = 0;
  int sensor22_on = 0;
```

```
  int sensor23_on = 0;
  int sensor24_on = 0;

void loop() {

  int sensor0 = 0;
  int sensor1 = 0;
  int sensor2 = 0;
  int sensor3 = 0;
  int sensor4 = 0;
  int sensor5 = 0;
  int sensor6 = 0;
  int sensor7 = 0;

  int sensor8 = 0;
  int sensor9 = 0;
  int sensor10 = 0;
  int sensor11 = 0;
  int sensor12 = 0;
  int sensor13 = 0;
  int sensor14 = 0;
  int sensor15 = 0;

  int sensor16 = 0;
  int sensor17 = 0;
  int sensor18 = 0;
  int sensor19 = 0;
  int sensor20 = 0;
  int sensor21 = 0;
  int sensor22 = 0;
  int sensor23 = 0;

  int sensor24 = 0;



  int tolerance = 750;

  int timedelay = 5;

  for (int i = 0; i<8; i++) {
       if (i == 0) {
       digitalWrite(4,LOW);
       digitalWrite(3,LOW);
       digitalWrite(2,LOW);

       digitalWrite(7,LOW);
       digitalWrite(6,LOW);
       digitalWrite(5,LOW);

       digitalWrite(10,LOW);
       digitalWrite(9,LOW);
       digitalWrite(8,LOW);
```

```
delay(timedelay);

sensor0 = analogRead(A1);
sensor8 = analogRead(A2);
sensor16 = analogRead(A3);
}
else if (i == 1) {
digitalWrite(4,LOW);
digitalWrite(3,LOW);
digitalWrite(2,HIGH);

digitalWrite(7,LOW);
digitalWrite(6,LOW);
digitalWrite(5,HIGH);

digitalWrite(10,LOW);
digitalWrite(9,LOW);
digitalWrite(8,HIGH);

delay(timedelay);

sensor1 = analogRead(A1);
sensor9 = analogRead(A2);
sensor17 = analogRead(A3);
}
else if (i == 2) {
digitalWrite(4,LOW);
digitalWrite(3,HIGH);
digitalWrite(2,LOW);

digitalWrite(7,LOW);
digitalWrite(6,HIGH);
digitalWrite(5,LOW);

digitalWrite(10,LOW);
digitalWrite(9,HIGH);
digitalWrite(8,LOW);

delay(timedelay);

sensor2 = analogRead(A1);
sensor10 = analogRead(A2);
sensor18 = analogRead(A3);
}
else if (i == 3) {
digitalWrite(4,LOW);
digitalWrite(3,HIGH);
digitalWrite(2,HIGH);

digitalWrite(7,LOW);
digitalWrite(6,HIGH);
digitalWrite(5,HIGH);

digitalWrite(10,LOW);
```

```
digitalWrite(9,HIGH);
digitalWrite(8,HIGH);

delay(timedelay);

sensor3 = analogRead(A1);
sensor11 = analogRead(A2);
sensor19 = analogRead(A3);
}
else if (i == 4) {
digitalWrite(4,HIGH);
digitalWrite(3,LOW);
digitalWrite(2,LOW);

digitalWrite(7,HIGH);
digitalWrite(6,LOW);
digitalWrite(5,LOW);

digitalWrite(10,HIGH);
digitalWrite(9,LOW);
digitalWrite(8,LOW);

delay(timedelay);

sensor4 = analogRead(A1);
sensor12 = analogRead(A2);
sensor20 = analogRead(A3);
}
else if (i == 5) {
digitalWrite(4,HIGH);
digitalWrite(3,LOW);
digitalWrite(2,HIGH);

digitalWrite(7,HIGH);
digitalWrite(6,LOW);
digitalWrite(5,HIGH);

digitalWrite(10,HIGH);
digitalWrite(9,LOW);
digitalWrite(8,HIGH);

delay(timedelay);

sensor5 = analogRead(A1);
sensor13 = analogRead(A2);
sensor21 = analogRead(A3);
}
else if (i == 6) {
digitalWrite(4,HIGH);
digitalWrite(3,HIGH);
digitalWrite(2,LOW);

digitalWrite(7,HIGH);
digitalWrite(6,HIGH);
```

```
     digitalWrite(5,LOW);

     digitalWrite(10,HIGH);
     digitalWrite(9,HIGH);
     digitalWrite(8,LOW);

     delay(timedelay);

     sensor6 = analogRead(A1);
     sensor14 = analogRead(A2);
     sensor22 = analogRead(A3);
     }
     else {
     digitalWrite(4,HIGH);
     digitalWrite(3,HIGH);
     digitalWrite(2,HIGH);

     digitalWrite(7,HIGH);
     digitalWrite(6,HIGH);
     digitalWrite(5,HIGH);

     digitalWrite(10,HIGH);
     digitalWrite(9,HIGH);
     digitalWrite(8,HIGH);

     delay(timedelay);

     sensor7 = analogRead(A1);
     sensor15 = analogRead(A2);
     sensor23 = analogRead(A3);
     }
     delay(timedelay);
}

sensor24 = analogRead(A4);

if (sensor0 > tolerance) {
     sensor0 = 1023;
}
if (sensor1 > tolerance) {
     sensor1 = 1023;
}
if (sensor2 > tolerance) {
     sensor2 = 1023;
}
if (sensor3 > tolerance) {
     sensor3 = 1023;
}
if (sensor4 > tolerance) {
     sensor4 = 1023;
}
if (sensor5 > tolerance) {
     sensor5 = 1023;
}
```

```
if (sensor6 > tolerance) {
     sensor6 = 1023;
}
if (sensor7 > tolerance) {
     sensor7 = 1023;
}
if (sensor8 > tolerance) {
     sensor8 = 1023;
}
if (sensor9 > tolerance) {
     sensor9 = 1023;
}
if (sensor10 > tolerance) {
     sensor10 = 1023;
}
if (sensor11 > tolerance) {
     sensor11 = 1023;
}
if (sensor12 > tolerance) {
     sensor12 = 1023;
}
if (sensor13 > tolerance) {
     sensor13 = 1023;
}
if (sensor14 > tolerance) {
     sensor14 = 1023;
}
if (sensor15 > tolerance) {
     sensor15 = 1023;
}
if (sensor16 > tolerance) {
     sensor16 = 1023;
}
if (sensor17 > tolerance) {
     sensor17 = 1023;
}
if (sensor18 > tolerance) {
     sensor18 = 1023;
}
if (sensor19 > tolerance) {
     sensor19 = 1023;
}
if (sensor20 > tolerance) {
     sensor20 = 1023;
}
if (sensor21 > tolerance) {
     sensor21 = 1023;
}
if (sensor22 > tolerance) {
     sensor22 = 1023;
}
if (sensor23 > tolerance) {
     sensor23 = 1023;
}
```

```
if (sensor24 > tolerance) {
     sensor24 = 1023;
}

if (sensor0 < 1023 && sensor0_on != 1) {
     sensor0_on = 1;
}
if (sensor0 == 1023 && sensor0_on == 1) {
     sensor0_on = 2;
}
if (sensor1 < 1023 && sensor1_on != 1) {
     sensor1_on = 1;
}
if (sensor1 == 1023 && sensor1_on == 1) {
     sensor1_on = 2;
}
if (sensor2 < 1023 && sensor2_on != 1) {
     sensor2_on = 1;
}
if (sensor2 == 1023 && sensor2_on == 1) {
     sensor2_on = 2;
}
if (sensor3 < 1023 && sensor3_on != 1) {
     sensor3_on = 1;
}
if (sensor3 == 1023 && sensor3_on == 1) {
     sensor3_on = 2;
}
if (sensor4 < 1023 && sensor4_on != 1) {
     sensor4_on = 1;
}
if (sensor4 == 1023 && sensor4_on == 1) {
     sensor4_on = 2;
}
if (sensor5 < 1023 && sensor5_on != 1) {
     sensor5_on = 1;
}
if (sensor5 == 1023 && sensor5_on == 1) {
     sensor5_on = 2;
}
if (sensor6 < 1023 && sensor6_on != 1) {
     sensor6_on = 1;
}
if (sensor6 == 1023 && sensor6_on == 1) {
     sensor6_on = 2;
}
if (sensor7 < 1023 && sensor7_on != 1) {
     sensor7_on = 1;
}
if (sensor7 == 1023 && sensor7_on == 1) {
     sensor7_on = 2;
}
if (sensor8 < 1023 && sensor8_on != 1) {
     sensor8_on = 1;
```

```
}
if (sensor8 == 1023 && sensor8_on == 1) {
    sensor8_on = 2;
}
if (sensor9 < 1023 && sensor9_on != 1) {
    sensor9_on = 1;
}
if (sensor9 == 1023 && sensor9_on == 1) {
    sensor9_on = 2;
}
if (sensor10 < 1023 && sensor10_on != 1) {
    sensor10_on = 1;
}
if (sensor10 == 1023 && sensor10_on == 1) {
    sensor10_on = 2;
}
if (sensor11 < 1023 && sensor11_on != 1) {
    sensor11_on = 1;
}
if (sensor11 == 1023 && sensor11_on == 1) {
    sensor11_on = 2;
}
if (sensor12 < 1023 && sensor12_on != 1) {
    sensor12_on = 1;
}
if (sensor12 == 1023 && sensor12_on == 1) {
    sensor12_on = 2;
}
if (sensor13 < 1023 && sensor13_on != 1) {
    sensor13_on = 1;
}
if (sensor13 == 1023 && sensor13_on == 1) {
    sensor13_on = 2;
}
if (sensor14 < 1023 && sensor14_on != 1) {
    sensor14_on = 1;
}
if (sensor14 == 1023 && sensor14_on == 1) {
    sensor14_on = 2;
}
if (sensor15 < 1023 && sensor15_on != 1) {
    sensor15_on = 1;
}
if (sensor15 == 1023 && sensor15_on == 1) {
    sensor15_on = 2;
}
if (sensor16 < 1023 && sensor16_on != 1) {
    sensor16_on = 1;
}
if (sensor16 == 1023 && sensor16_on == 1) {
    sensor16_on = 2;
}
if (sensor17 < 1023 && sensor17_on != 1) {
    sensor17_on = 1;
```

```
}
if (sensor17 == 1023 && sensor17_on == 1) {
     sensor17_on = 2;
}
if (sensor18 < 1023 && sensor18_on != 1) {
     sensor18_on = 1;
}
if (sensor18 == 1023 && sensor18_on == 1) {
     sensor18_on = 2;
}
if (sensor19 < 1023 && sensor19_on != 1) {
     sensor19_on = 1;
}
if (sensor19 == 1023 && sensor19_on == 1) {
     sensor19_on = 2;
}
if (sensor20 < 1023 && sensor20_on != 1) {
     sensor20_on = 1;
}
if (sensor20 == 1023 && sensor20_on == 1) {
     sensor20_on = 2;
}
if (sensor21 < 1023 && sensor21_on != 1) {
     sensor21_on = 1;
}
if (sensor21 == 1023 && sensor21_on == 1) {
     sensor21_on = 2;
}
if (sensor22 < 1023 && sensor22_on != 1) {
     sensor22_on = 1;
}
if (sensor22 == 1023 && sensor22_on == 1) {
     sensor22_on = 2;
}
if (sensor23 < 1023 && sensor23_on != 1) {
     sensor23_on = 1;
}
if (sensor23 == 1023 && sensor23_on == 1) {
     sensor23_on = 2;
}
if (sensor24 < 1023 && sensor24_on != 1) {
     sensor24_on = 1;
}
if (sensor24 == 1023 && sensor24_on == 1) {
     sensor24_on = 2;
}

Serial.print(sensor0);
Serial.print(" ");
Serial.print(sensor0_on);
Serial.print(" ");
Serial.print(sensor1);
Serial.print(" ");
Serial.print(sensor1_on);
```

```
Serial.print(" ");
Serial.print(sensor2);
Serial.print(" ");
Serial.print(sensor2_on);
Serial.print(" ");
Serial.print(sensor3);
Serial.print(" ");
Serial.print(sensor3_on);
Serial.print(" ");
Serial.print(sensor4);
Serial.print(" ");
Serial.print(sensor4_on);
Serial.print(" ");
Serial.print(sensor5);
Serial.print(" ");
Serial.print(sensor5_on);
Serial.print(" ");
Serial.print(sensor6);
Serial.print(" ");
Serial.print(sensor6_on);
Serial.print(" ");
Serial.print(sensor7);
Serial.print(" ");
Serial.print(sensor7_on);
Serial.print(" ");
Serial.print(sensor8);
Serial.print(" ");
Serial.print(sensor8_on);
Serial.print(" ");
Serial.print(sensor9);
Serial.print(" ");
Serial.print(sensor9_on);
Serial.print(" ");
Serial.print(sensor10);
Serial.print(" ");
Serial.print(sensor10_on);
Serial.print(" ");
Serial.print(sensor11);
Serial.print(" ");
Serial.print(sensor11_on);
Serial.print(" ");
Serial.print(sensor12);
Serial.print(" ");
Serial.print(sensor12_on);
Serial.print(" ");
Serial.print(sensor13);
Serial.print(" ");
Serial.print(sensor13_on);
Serial.print(" ");
Serial.print(sensor14);
Serial.print(" ");
Serial.print(sensor14_on);
Serial.print(" ");
Serial.print(sensor15);
```

```
  Serial.print(" ");
  Serial.print(sensor15_on);
  Serial.print(" ");
  Serial.print(sensor16);
  Serial.print(" ");
  Serial.print(sensor16_on);
  Serial.print(" ");
  Serial.print(sensor17);
  Serial.print(" ");
  Serial.print(sensor17_on);
  Serial.print(" ");
  Serial.print(sensor18);
  Serial.print(" ");
  Serial.print(sensor18_on);
  Serial.print(" ");
  Serial.print(sensor19);
  Serial.print(" ");
  Serial.print(sensor19_on);
  Serial.print(" ");
  Serial.print(sensor20);
  Serial.print(" ");
  Serial.print(sensor20_on);
  Serial.print(" ");
  Serial.print(sensor21);
  Serial.print(" ");
  Serial.print(sensor21_on);
  Serial.print(" ");
  Serial.print(sensor22);
  Serial.print(" ");
  Serial.print(sensor22_on);
  Serial.print(" ");
  Serial.print(sensor23);
  Serial.print(" ");
  Serial.print(sensor23_on);
  Serial.print(" ");
  Serial.print(sensor24);
  Serial.print(" ");
  Serial.println(sensor24_on);

}
```

Appendix B: MIDI Note Assignments

Midi 25 - G major

Midi 26 - D major

Midi 27 - A major

Midi 28 - E major

Midi 29 - B major

Midi 30 - Gb major

Midi 31 - Db major

Midi 32 - Ab major

Midi 33 - Eb major

Midi 34 - Bb major

Midi 35 - F major

Midi 36 - A minor

Midi 37 - E minor

Midi 38 - B minor

Midi 39 - F# minor

Midi 40 - C# minor

Midi 41 - G# minor

Midi 42 - Eb minor

Midi 43 - Bb minor

Midi 44 - F minor

Midi 45 - C minor

Midi 46 - G minor

Midi 47 - D minor

Midi 60 - C

Midi 61 - G

Midi 62 - D

Midi 63 - A

Midi 64 - E

Midi 65 - B

Midi 66 - Gb

Midi 67 - Db

Midi 68 - Ab

Midi 69 - Eb

Midi 70 - Bb

Midi 71 - F

Midi 72 - A

Midi 73 - E

Midi 74 - B

Midi 75 - F#

Midi 76 - C#

Midi 77 - G#

Midi 78 - Eb

Midi 79 - Bb

Midi 80 - F

Midi 81 - C

Midi 82 - G

Midi 83 - D

Midi 96 - calm-river-ambient

Midi 97 - nature sounds

Midi 98 - thunder

Midi 99 - 8-bit horror nosie

Midi 100 - video game death sound

Midi 101 - video game jump sound

Midi 102 - 8-bit sample music/beat

Midi 103 - coin collect sound

Midi 104- random chip tune bass loop

Midi 105 - long game over

Midi 106 - wolf howl

Midi 107 - typewriting

Midi 108 - c

Midi 109 - c#

Midi 110- d

Midi 111 - d#

Midi 112 - e

Midi 113- f

Midi 114 - f#

Midi 115 - g

Midi 116 - g#

Midi 117 - a

Midi 118 - a#

Midi 119 - b