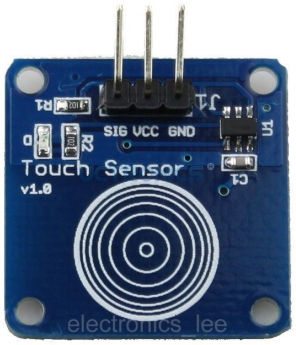
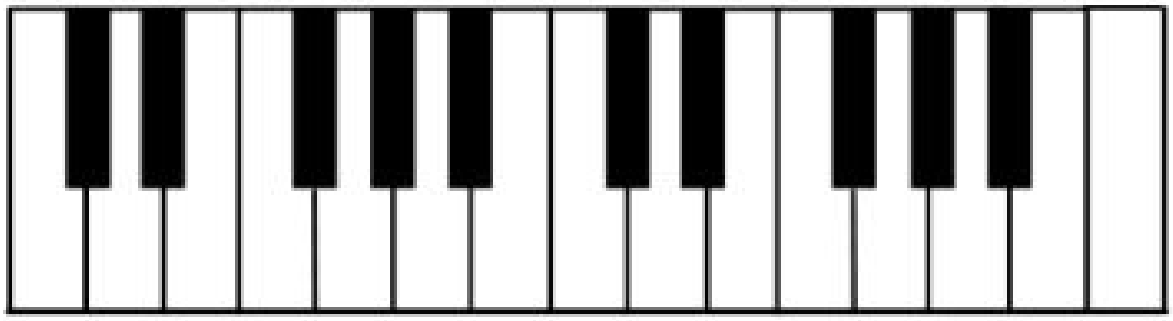
Project Write-Up for Synth-Hero

Members: Deon Zoss, Kathleen Mancillas, Dale Wesa

Project Summary: We have diverged somewhat from our initial thoughts on the project and have settled for something more digestible but also challenging. Our goal is to create a rudimentary synthesizer using touch-sensors as inputs, outputting to sound emitters.



The touch sensors will be organized in a manner to resemble the keys of a piano, spanning 2 octaves, which provides just enough keys to play simple songs with bass notes along with a melody. Further research needs to be done as to how to properly align the sensors on the circuit board, but this layout of the sensors will provide the easiest and most familiar way of playing.

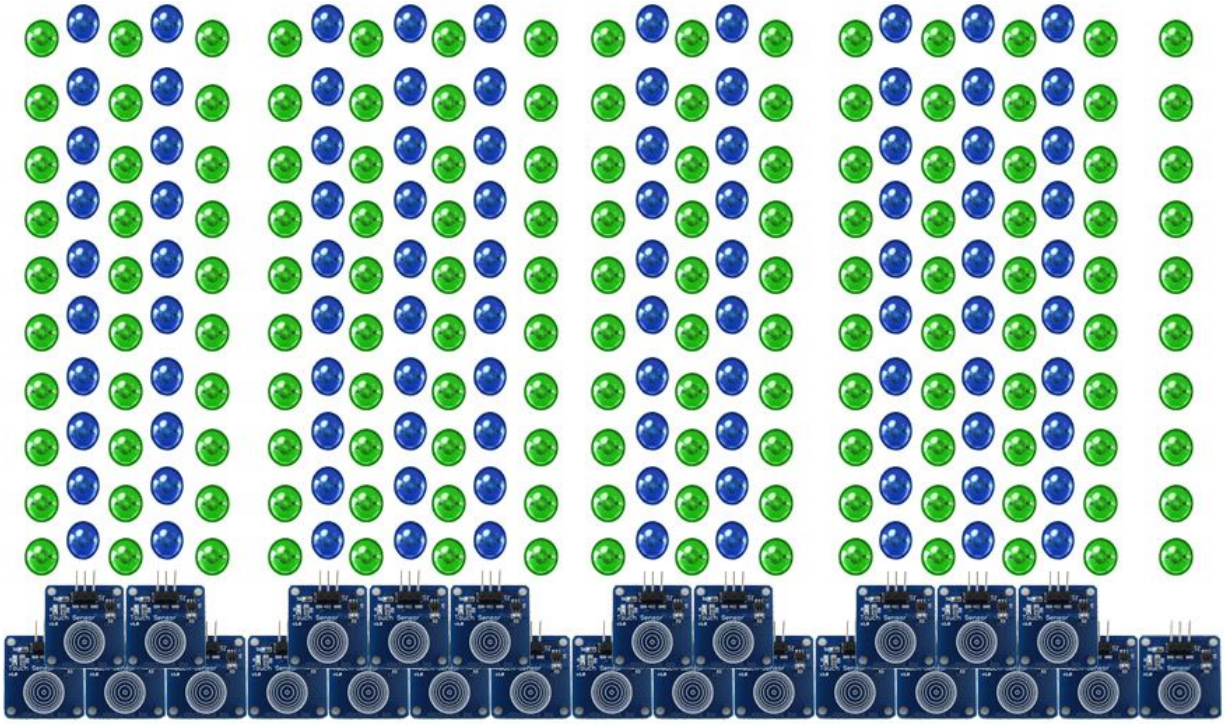


Each individual sensor will be mapped to output a certain note to the sound emitter, starting at C, and ending at another C two octaves higher.

The output for the sensors could either be a speaker or a buzzer. Since buzzers can only play one tone at a time, multiple buzzers would be needed if multiple keys wanted to be pressed simultaneously.

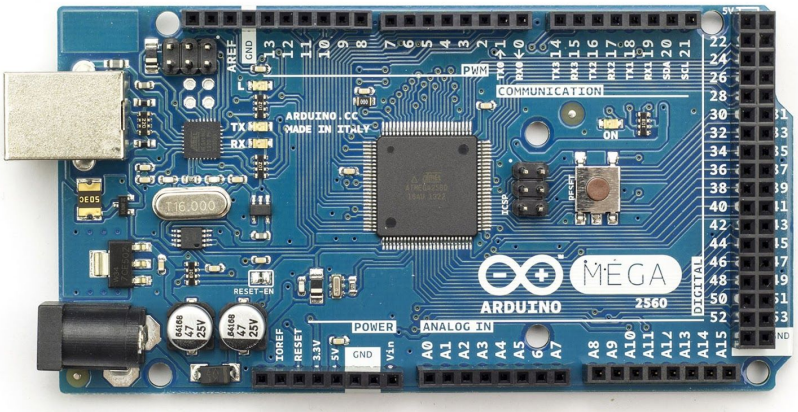
The rudimentary synthesizer will then be used as the tool to play “Synth-Hero”, a parody of the well-known game franchise “Guitar Hero”, in which the player is instructed to press certain notes at the right time and progress through a song. Our first thought for a display for the notes would be an lcd screen display that would be wide enough for all 25 sensors. However, looking for a cheap screen of this size was difficult, and we have since considered using LED’s instead.

A line of LED’s will be trailing down to each sensor. The LED’s will light up corresponding to the current song that is set to play. The LED that is lit will then continue to the next LED below it after a decided cycle time. Just after the very bottom LED is lit is when the sensor should be pressed. We have decided that 10 LED’s per sensor should be enough to produce the preferred playing experience. The more LED’s in each line, the more accurate you could display the timing between each note, and having 10 LED’s will provide enough reaction time for the user at a relatively quick tempo. Of course, the downside to using LED’s instead of a screen is all of the soldering and connections we will have to make on our circuit, which could be troublesome, since we would be using 250 LED’s (I go into detail about how to connect the LED’s later in the write-up). Having 10 LED’s in each line is not final, and may be subject to change later on.



The use of blue and green LED's, along with offsetting, can help to indicate which note is the one that should be played.

In terms of connecting all of the LED's and selecting which ones to light up, we will have no choice but to multiplex them. Looking into methods of multiplexing online has shown us that we could get the number of pins required for 250 LED's to at least 35. Of course, this isn't considering the touch sensors and speakers themselves (the sensors will most likely need to be multiplexed as well). The Arduino Mega along with each of our smaller arduinos should provide us enough pins to handle our project.



Here is a small sample code of synth-hero with only two buttons (notes), 3 LEDs per note, and no sound. In this sample, quarter notes are played between the left and right button at 60 bpm. When a note is hit or miss, or an extra note is added this is printed to the serial monitor.

```
const int LB = 12; // Left Button Pin
const int RB =  9; // Right Button Pin

bool L0b = false; // Left LED 0 on or off
bool R0b = false; // Right LED 0 on or off
bool L1b = false; // Left LED 1 on or off
bool R1b = false; // Right LED 1 on or off
bool L2b = false; // Left LED 2 on or off
bool R2b = false; // Right LED 2 on or off

// light pins
int L0 = 2;
int R0 = 3;
int L1 = 4;
int R1 = 5;
int L2 = 6;
int R2 = 7;

const int d = 500; // 1000 ms per division (60 divisions per minute)
String serial = "";

void setup() {
  Serial.begin(9600);
  pinMode(LB, INPUT);
  pinMode(RB, INPUT);
  pinMode(L0, OUTPUT);
  pinMode(R0, OUTPUT);
  pinMode(L1, OUTPUT);
  pinMode(R1, OUTPUT);
  pinMode(L2, OUTPUT);
  pinMode(R2, OUTPUT);
  attachInterrupt(digitalPinToInterrupt(RB), hitRight, RISING);
  attachInterrupt(digitalPinToInterrupt(LB), hitLeft,  RISING);
}

void loop() {
  // Simple left, right, left, right ...
  runSubdivision(true, false);
  runSubdivision(false, true);
}

void runSubdivision(bool left, bool right) {
  // slide notes down
  L0b = L1b; digitalWrite(L0, L0b ? HIGH : LOW);
  L1b = L2b; digitalWrite(L1, L1b ? HIGH : LOW);
```

```
  R0b = R1b; digitalWrite(R0, R0b ? HIGH : LOW);
  R1b = R2b; digitalWrite(R1, R1b ? HIGH : LOW);
  // set new notes at end
  L2b = left;  digitalWrite(L2, L2b ? HIGH : LOW);
  R2b = right; digitalWrite(R2, R2b ? HIGH : LOW);
  // wait (for behind the beat)
  delay(d);

  // check if hits good.
  for (char c : serial) {
    Serial.println(serial);
    switch (c) {
      case 'L':
        if (L0b) Serial.println("Nice!!!!!!");
        else     Serial.println("Ouch! xxxx");
        L0b = false;
        break;
      case 'R':
        if (R0b) Serial.println("Nice!!!!!!");
        else     Serial.println("Ouch! xxxx");
        R0b = false;
        break;
    }
  }
  if (L0b || R0b) Serial.println("Miss! xxxx");
  serial = "";


  // wait (for in front of the beat)
  delay(d);
}

void hitLeft() {
  serial.concat("L");
}

void hitRight() {
  serial.concat("R");
}
```