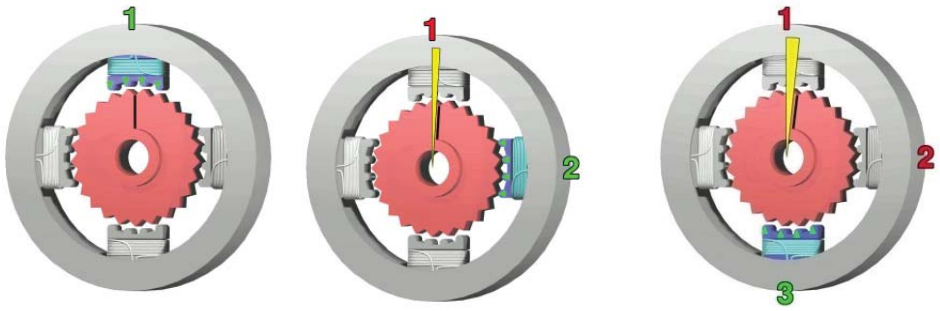
MOTORS_QUICK RECAP

TYPES OF MOTORS

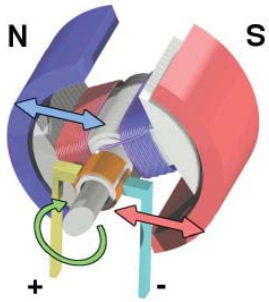
Steppers

_stepped movement



DC motors

_continuous movement



Servos

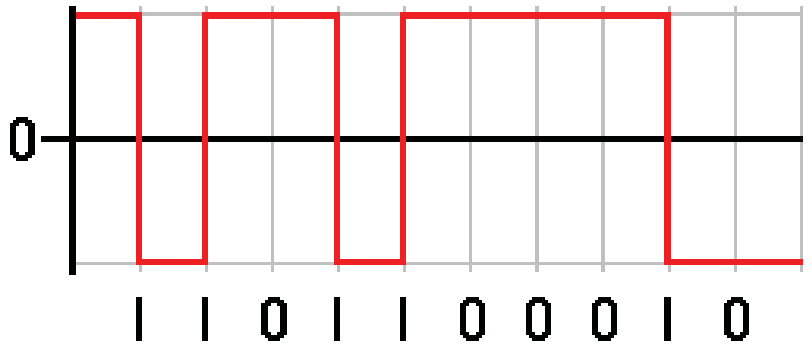
_tracked stepped movement

RISING/FALLING EDGE

In electronics, a **clock edge** is a transition in a clock signal from either low to high (0 to 1) or high to low (1 to 0). It is called an “edge” because the square wave which represents a clock has edges at those points.

A **rising edge** is the transition of a digital signal from low to high. It is also named positive edge. When a circuit’s rising edge triggered, it becomes active when the clock signal goes from **low to high, and ignores the high to low transition.**

A **falling edge** is the high to low transition. It’s also known as the negative edge. When a circuit’s falling edge triggered, it becomes active when the clock signal goes from **high to low, and ignores the low to high transition.**

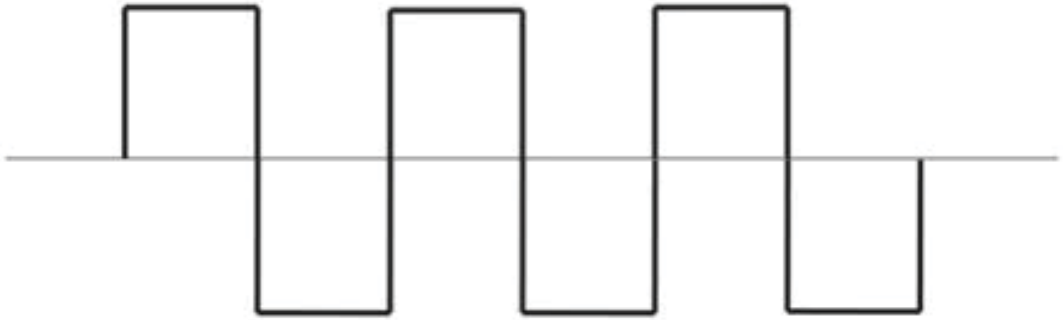


SOUND GENERATION/PROCESSING

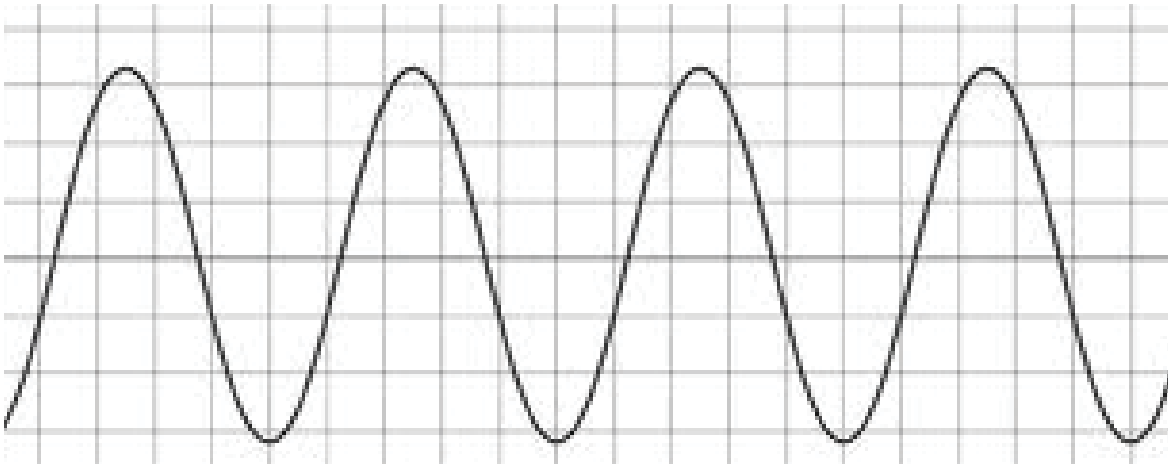
_ARDUINO
_PROCESSING

ARDUINO_SOUND

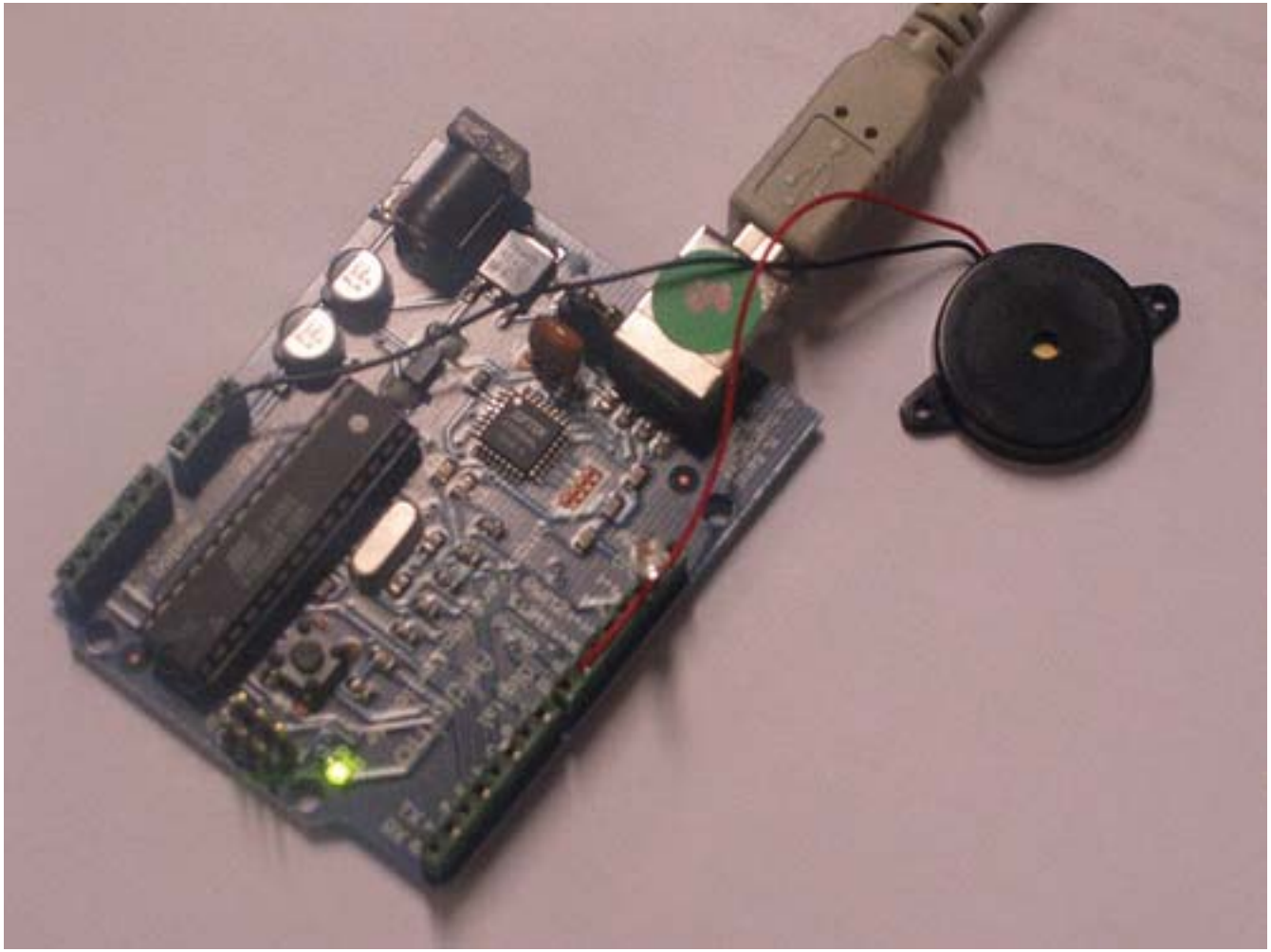
SQUARE WAVE



SINE WAVE(SOUND)



ARDUINO_SPEAKER



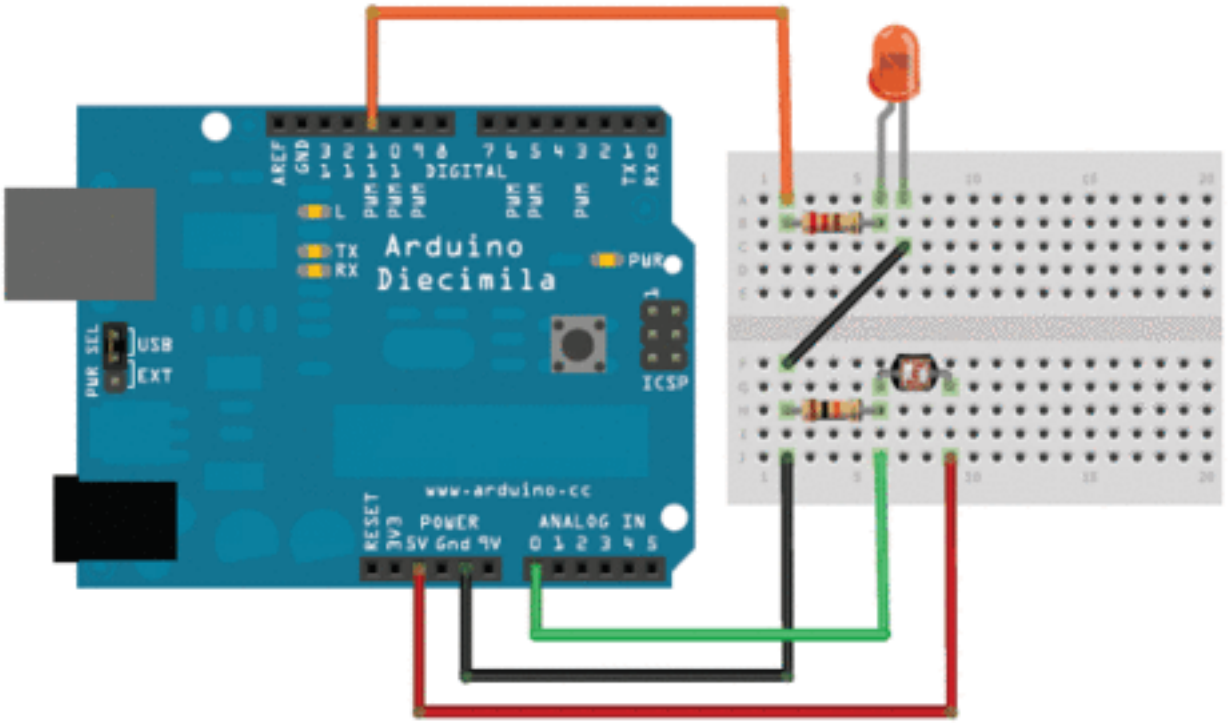
ARDUINO_TONE FUNCTION

OF TIMERS = # AUDIO OUTPUTS
* ATmega8: 2 (timers 2, and 1)
* ATmega168/328: 3 (timers 2, 1, and 0)
* ATmega1280: 6 (timers 2, 3, 4, 5, 1, 0)

```
void setup()
{
  tone(13,740,300);  //pin for output, freq., duration
  delay(100);
  tone(13,494,300);
  delay(200);
  tone(13,370,300);
  delay(100);
  tone(13,220,300);
  delay(200);
  tone(13,247,300);
  delay(100);
  tone(13,330,300);
  delay(200);
  tone(13,587,300);
  delay(100);
  noTone(13);
}

void loop()
{

}
```

Constant Name	Frequency (Hz)	Constant Name	Frequency (Hz)
NOTE_B2	123	NOTE_GS5	831
NOTE_C3	131	NOTE_A5	880
NOTE_CS3	139	NOTE_AS5	932
NOTE_D3	147	NOTE_B5	988
NOTE_DS3	156	NOTE_C6	1047
NOTE_E3	165	NOTE_CS6	1109
NOTE_F3	175	NOTE_D6	1175
NOTE_FS3	185	NOTE_DS6	1245
NOTE_G3	196	NOTE_E6	1319
NOTE_GS3	208	NOTE_F6	1397
NOTE_A3	220	NOTE_FS6	1480
NOTE_AS3	233	NOTE_G6	1568
NOTE_B3	247	NOTE_GS6	1661
NOTE_C4	262	NOTE_A6	1760
NOTE_CS4	277	NOTE_AS6	1865
NOTE_D4	294	NOTE_B6	1976
NOTE_DS4	311	NOTE_C7	2093
NOTE_E4	330	NOTE_CS7	2217
NOTE_F4	349	NOTE_D7	2349
NOTE_FS4	370	NOTE_DS7	2489
NOTE_G4	392	NOTE_E7	2637
NOTE_GS4	415	NOTE_F7	2794
NOTE_A4	440	NOTE_FS7	2960
NOTE_AS4	466	NOTE_G7	3136
NOTE_B4	494	NOTE_GS7	3322
NOTE_C5	523	NOTE_A7	3520
NOTE_CS5	554	NOTE_AS7	3729
NOTE_D5	587	NOTE_B7	3951
NOTE_DS5	622	NOTE_C8	4186
NOTE_E5	659	NOTE_CS8	4435
NOTE_F5	698	NOTE_D8	4699
NOTE_FS5	740	NOTE_DS8	4978
NOTE_G5	784		

ARDUINO_CONTROLLING TEMPO, PITCH AND VOLUME



ARDUINO_CONTROLLING TEMPO, PITCH AND VOLUME

```
int inPin = ;                    // analog in
int val = 0;                     // where to store info from analog pin
int val2 = 0;                    // analog in mapped to a tone
int led = 11;                    // output of led
int speakerOut = 13;             // output of speaker
int intval = 0;                  // inital read of sensor
//TIMING/MELODY ================================
int tempo = 50;
int beat = 0;
long duration  = 0;
char note1;
float tone1;
float prevtone = 0;
float temptone;
void setup()
{
 pinMode(speakerOut, OUTPUT);
 Serial.begin(9600);
 intval = analogRead(inPin);          //mapping the intial read as the limit
 val = analogRead(inPin);             //read the val coming from the sensor
 //initialize the first tone and set it as the previous tone
 if(val > 1){
  val = constrain(val, 1, 1028);
  val = map(val, 1, intval, 1, 255);
   //map values to light intensity
  analogWrite(led, val);
   //map values to specific note in the convert function
  val2 = map(val, 1, 255, 956, 1915);
  tone1 = val2;
 }
 else
 {
  tone1 = 0;
  analogWrite(led, 0);
 }
 convert(tone1);
 //the current tone becomes the previous tone
 prevtone = temptone;
}
```

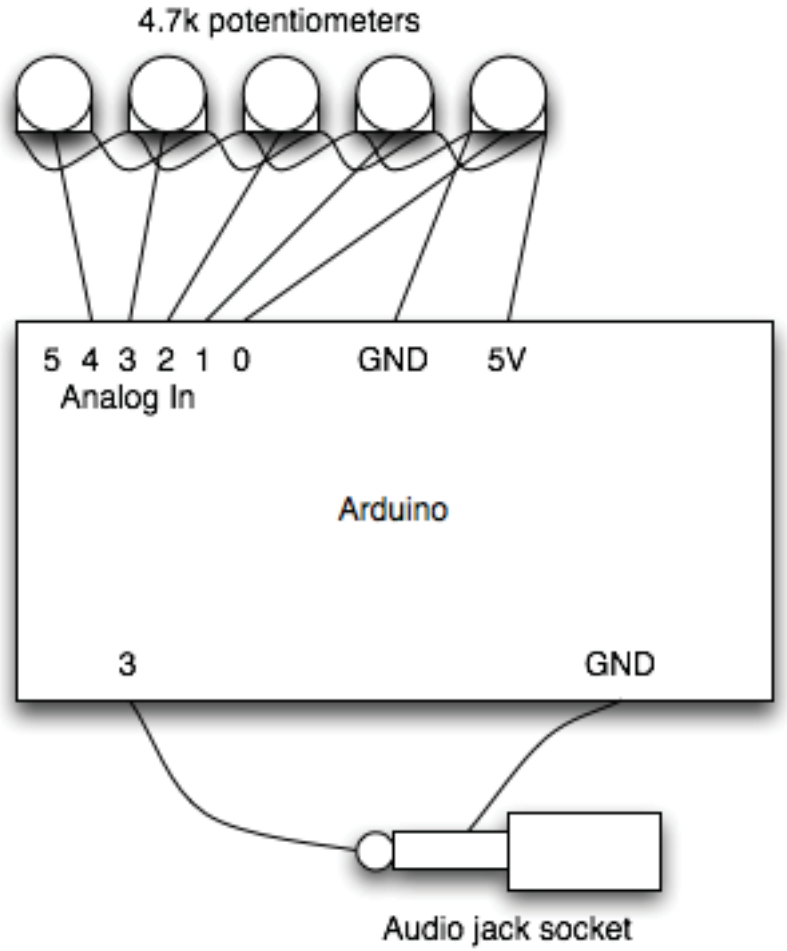
```
void convert(float temptone1)
{
 if (temptone1 >= 1915)
 {
   note1 =  'c';
   temptone = 2886/2;

 }
 if (temptone1 >= 1700 && temptone1 < 1915)
 {
   note1 =  'd';
   temptone = 1926/2;
 }
 if (temptone1 >= 1519 && temptone1 < 1700)
 {
   note1 =  'e';
   temptone = 1286/2;
 }
 if (temptone1 >= 1432 && temptone1 < 1519)
 {
   note1 =  'f';
   temptone = 858/2;
 }
 if (temptone1 >= 1275 && temptone1 < 1432)
 {
   note1 = 'g';
   temptone = 540/2;
 }
 if (temptone1 >= 1014 && temptone1 < 1136)
 {
   note1 =  'a';
   temptone = 360/2;
 }
 if (temptone1 >= 956 && temptone1 < 1014)
 {
   note1 = 'b';
   temptone = 240/2;
 }
 if (temptone1 <= 956)
 {
   note1 =  'C';
   temptone = 160/2;
 }
}
```

ARDUINO_CONTROLLING TEMPO, PITCH AND VOLUME

```
/*playTone function is called to play the temptone(current tone)
duration is the tempo
to change the volume, an analogWrite could be used 0-255
or by putting a potentiometer in circuit before the speaker
*/
void playTone(int tone2, int duration) {
 for (long i = 0; i < duration * 1000L; i += tone2*4)   //length of time the note is played
{
   digitalWrite(speakerOut, HIGH);
   delayMicroseconds(tone2/2);                          //speed of the pulse
   digitalWrite(speakerOut, LOW);
   delayMicroseconds(tone2/2);
 }
}
void loop() {
   val = analogRead(inPin);
   val = constrain(val, 1, 1028);
   val = map(val, 1, intval, 1, 255);
   analogWrite(led, val);
   val2 = map(val, 1, 255, 956, 1915);
   tone1 = val2;
 }
 else
 {
   tone1 = 0;
    analogWrite(led, 0);
 }
 convert(tone1);
  /*if the current tone is greater than the previous tone, add to previous tone until it
reaches the current tone to place the notes in between to create a gradient*/
 if(temptone > prevtone)
 {
    for(int j=prevtone; j<temptone; j++)
    {
      playTone(j,tempo);
    }
 }
 else
 {
  /*else if  the previous tone is greater than the previous tone place the  subtract from
the previous tone unitl it reaches the temptone to place  the notes in between to create a
gradient*/
    for(int i=prevtone; i>temptone; i--)
    {
       playTone(i,tempo);
    }
 }
 prevtone = temptone;
  //plays single note at one time
  /*
  if (note1 == ' ')  {
   delay(1 * tempo); // rest
  }
  else
  {
    playTone(temptone, 1 * tempo);
   }*/
}
```

ARDUINO_SYNTHESIZER

http://code.google.com/p/tinkerit/wiki/Auduino



PROCESSING _MINIM LIBRARY: LOAD SAMPLE

EXAMPLES > LIBRARY> MINIM> LOAD SAMPLE

```
import ddf.minim.*;
Minim minim;
AudioSample kick;
AudioSample snare;
AudioSample guitar;
int prevsec = 0;
int cursec = 0;

void setup()
{
  size(512, 200, P2D);
  // always start Minim before you do anything with it
  minim = new Minim(this);
  // load BD.mp3 from the data folder with a 1024 sample buffer
  // kick = Minim.loadSample("BD.mp3");
  // load BD.mp3 from the data folder, with a 512 sample buffer
  kick = minim.loadSample("BD.mp3", 2048);
  guitar = minim.loadSample("818_Koen_Gd_amch0.wav", 2048);
  snare = minim.loadSample("774_vitriolix_snare.wav", 2048);

  //initalize previous seconds()
  cursec = second();
  prevsec = cursec;
}
```

```
void draw(){
  background(0);
  stroke(255);
  // use the mix buffer to draw the waveforms.
  // because these are MONO files, we could have used the left or right      buffers
and got the same data
  for (int i = 0; i < kick.bufferSize() - 1; i++)
  {
    line(i, 100 - kick.left.get(i)*50, i+1, 100 - kick.left.get(i+1)*50);
  }
  for (int i = 0; i <snare.bufferSize() - 1; i++)
  {
    line(i, 100 - snare.left.get(i)*50, i+1, 100 - snare.left.get(i+1)*50);
  }
  for (int i = 0; i <guitar.bufferSize() - 1; i++)
  {
    line(i, 100 - guitar.left.get(i)*50, i+1, 100 - guitar.left.get(i+1)*50);
  }
 //loop sample using cumputer clock
  cursec = second();
  if( (cursec - prevsec) == 3)
  {
     kick.trigger();
     prevsec = cursec;
  }
}
void keyPressed()
{
  if ( key == 'k' ) kick.trigger();
  if ( key == 's' ) snare.trigger();
  if ( key == 'k' ) guitar.trigger();
}
void stop()
{
  // always close Minim audio classes when you are done with them
  kick.close();
  snare.close();
  guitar.close();
  minim.stop();
  super.stop();
}
```

freesound.com_audio samples

PROCESSING _ARDUINO LIBRARY/MIDI LIBRARY

EXAMPLES > LIBRARY> FIRMATA> STANDARDFIRMATA

```
import themidibus.*; //Import midibus library
import cc.arduino.*; //Import arduino library

MidiBus myBus;
Arduino arduino;
int channel = 0;
int velocity = 127;

void setup()
{
  size(640, 495);
 // List all available Midi devices on STDOUT. This will show each device's index and name.
   MidiBus.list();

            // or for testing you could ...
            //            Parent  In       Out
            //              |    |        |
            myBus = new MidiBus(this, "", "Java Sound Synthesizer");
// Create a new MidiBus with no input device and the default Java Sound Synthesizer as the
output device.
    arduino = new Arduino(this, Arduino.list()[0], 57600);
}
void draw()
{
            int pitchf = arduino.analogRead(0);
            int pitchi = map(pitchf, 0, 1023, 0, 100);
            //convert(pitchf);
            //float pitchi = temptone;

            myBus.sendNoteOn(channel, int(pitchi), velocity); // Send a Midi noteOn
            delay(200);
            myBus.sendNoteOff(channel, int(pitchi), velocity); // Send a Midi nodeOff
}
```

```
void convert(int val)
 {
   if (val >= 800)
   {
    temptone = 25;
   }
   if (val <= 800 && val > 700)
   {
    temptone = 32;
   }
   if (val <= 700 && val > 600)
   {
    temptone = 39;
   }
   if (val <= 600 && val > 500)
   {
    temptone = 46;
   }
   if (val <= 500 && val > 400)
   {
    temptone = 54;
   }
   if (val <= 400 && val > 300)
   {
    temptone = 61;
   }
   if (val <= 300 && val > 200)
   {
    temptone = 68;
   }
   if (val <= 200)
   {
    temptone = 75;
   }
 }
}
```

```
void keyPressed()
{
   if(key == CODED)
   {
    if (keyCode == UP)
    {
      channel++;
      println("channel:" + channel);
    }
    if (keyCode == DOWN)
    {
       channel--;
       println("channel:" + channel);
    }
    if (keyCode == RIGHT)
    {
       velocity--;
       println("velocity:" + velocity);
    }
    if (keyCode == LEFT)
    {
       velocity++;
       println("velocity:" +velocity);
    }
   }
}
```

PROCESSING _MINIM LIBRARY: LOAD FILE/USING .*TXT FILE TO DRAW SOUND WAVE

```
import ddf.minim.*;
import ddf.minim.signals.*;

Minim minim;
Minim minim1;
AudioPlayer player;
AudioOutput out;
SineWave sine;
String var[];
float freq = 0;
float prevfreq = 0;
float newfreq = 0;

void setup()
{
 size(512, 200, P2D);
 //intialize object for sine wave oscilator
 minim = new Minim(this);
 out = minim.getLineOut(Minim.STEREO);
 sine = new SineWave(0, 0.5, out.sampleRate());
 sine.portamento(200);                          //tempo
 out.addSignal(sine);                           //sine or saw wave
 //create background sound
 minim1 = new Minim(this);
 player = minim1.loadFile("background.mp3", 2048);
 player.play();
}

void draw()
{
 background(0);
 stroke(255);
 var = loadStrings("freq.txt");                 //read from text file
 newfreq = Float.valueOf(var[0]);               //cast String at index 0
 if(prevfreq != newfreq)
 {
  println(newfreq);
  sine.setFreq(newfreq);
  //sine.setPan(newfreq);
 }
 else
 {
  minim.stop();
 }
 // draw the waveforms
 for(int i = 0; i < out.bufferSize() - 1; i++)
 {
  float x1 = map(i, 0, out.bufferSize(), 0, width);
  float x2 = map(i+1, 0, out.bufferSize(), 0, width);
  line(x1, 50 + out.left.get(i)*50, x2, 50 + out.left.get(i+1)*50);
  line(x1, 150 + out.right.get(i)*50, x2, 150 + out.right.get(i+1)*50);
 }
 prevfreq = newfreq;
}

void stop()
{
 player.close();
 out.close();
 minim.stop();
 minim1.stop();
 super.stop();
}
```