# iSim Final Project: Theremin

Lisa Joelle Hachmann
Franton Lin

December 17, 2014

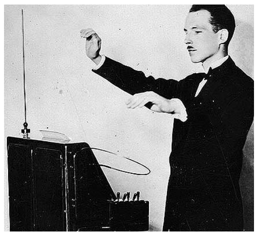


Figure 1: Leon Theremin playing a theremin

**Abstract**

We created a theremin using ultrasonic transducers, filters, a demodulator, and an Arduino. A theremin is an electrical instrument that generates a pitch based on the distance between the user's hand and an antenna/sensor. We used the ultrasonic transducers to measure the distance to the hand. The Arduino takes the processed signal to calculates the distance to the hand and then outputs an audio signal at a frequency proportional to the measured distance. Our theremin can directly power a small speaker or be connected to other devices/amplifiers via a 3.5mm headphone jack.

## 1 The Circuit

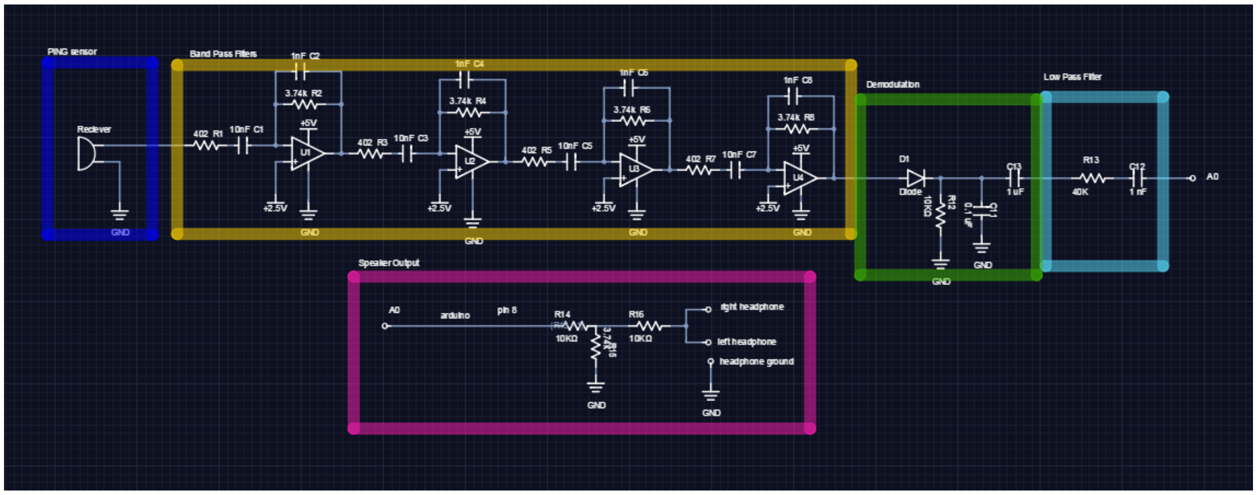


Figure 2: Schematic

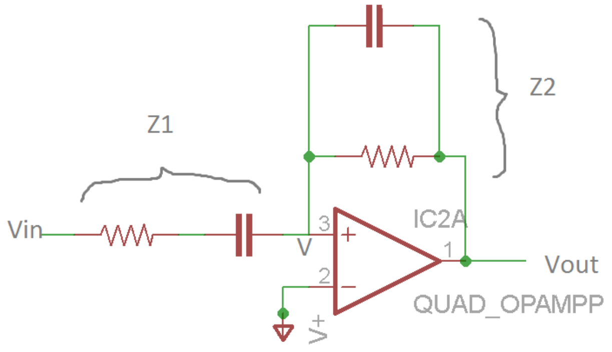


Figure 3: Active band pass filter schematic

## 1.1 Band Pass Filters

See Figure 3 for the schematic used to analyze the system.

Let:

$$Z1 = R1 + C1 Z2 = \frac{1}{\frac{1}{R2} + \frac{1}{C2}}$$

$$Z1 = R_1 + \frac{1}{jwC1}$$

$$Z2 = \frac{1}{\frac{1}{R_2} + \frac{1}{\frac{1}{jwC2}}}$$

The circuit analysis then is:

$$\frac{Vin - V}{Z1} = \frac{V - Vo}{Z2}$$

$$\frac{Vo}{Vin} = \frac{-Z2}{Z1}$$

$$\frac{Vo}{Vin} = \frac{\frac{-1}{jwC2 + \frac{1}{R2}}}{R1 + \frac{1}{jwC1}}$$

$$\frac{Vo}{Vin} = \frac{\frac{-R2}{1+jwR2C2}}{\frac{jwR1C1+1}{jwC1}}$$

$$\frac{Vo}{Vin} = \frac{jwC1}{1 + jwR1C1} * \frac{-R2}{1 + jwC2R2}$$

$$\frac{Vo}{Vin} = \frac{jwC1R1}{1 + jwR1C1} * \frac{1}{1 + jwR2C2} * \frac{-R2}{R1}$$

High pass $= \frac{jwC1R1}{1+jwR1C1}$

Low pass $= \frac{1}{1+jwR2C2}$

Amplitude $= \frac{-R2}{R1}$
Band pass filter = Low pass* High pass * Gain

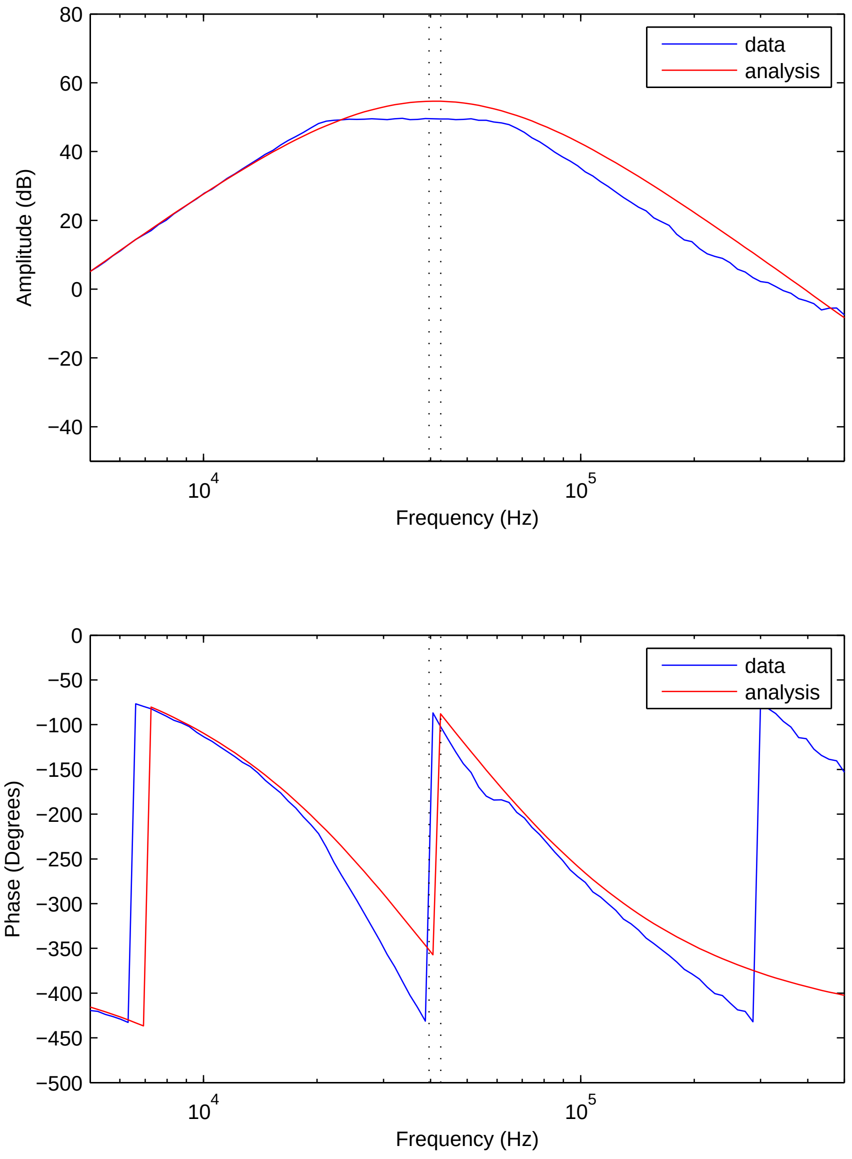


Figure 4: Bode plot of data and analysis of the four active band pass filters before demodulation

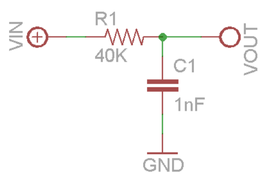


Figure 5: Low Pass Filter Schematic

After filtering and demodulating the received signal, we used a low pass filter to remove any remaining 40khz noise. The circuit analysis is as follows: Let R1 and C1 be Z1 and Z2, respectively.

Using the voltage divider rule

$$\frac{V_{out}}{V_{in}} = \frac{Z2}{Z1 + Z2}$$

Substituting in R1 for Z1 and $\frac{1}{j\omega C1}$ for Z2, we get

$$\frac{V_{out}}{V_{in}} = \frac{1}{j\omega R1C1 + 1}$$

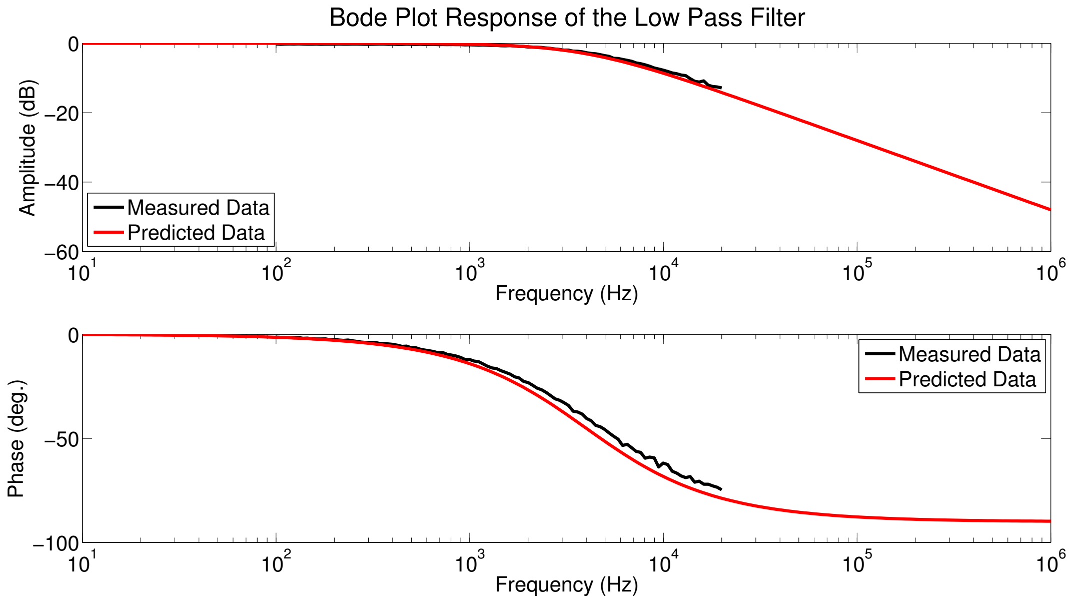
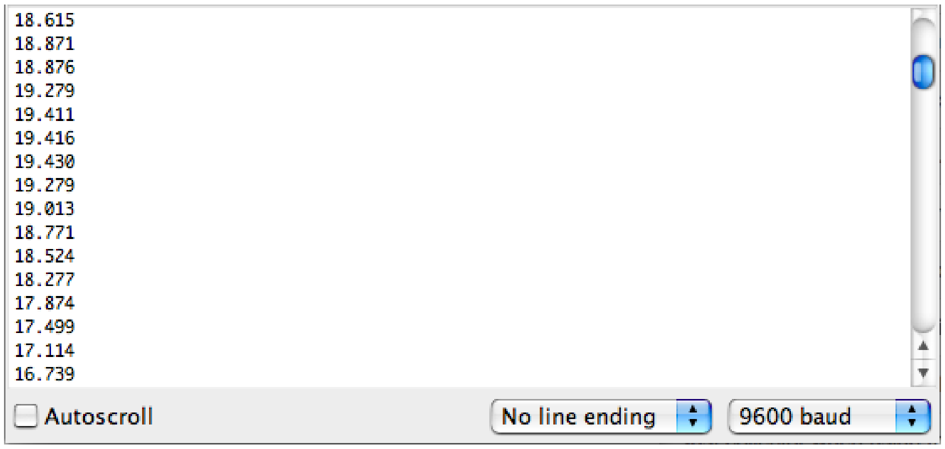


Figure 6

## 1.2 Sound Output

The red block shown in Figure 2 is part of the circuitry that deals with mapping the voltage and outputting sound. First, the output of the earlier circuitry goes into the Arduino, which averages out the distances and maps them to notes, which is then passed to a digital pin to a speaker or headphone.

The voltage peak from the ultrasonic range finder is passed to the arduino, which averages out 10 readings from the PING sensor before calculating which corresponding note to play. The reasoning here was that without the averaging, the theremin would play every note, and it was both very difficult to play consistently and it would rapidly change pitch while the musician puts his/her hand into the theremin's range. The averager also does not allow the theremin to play unless the majority of the 10 current readings are positive distances. This eliminated both problems of the difficulty level in playing the instrument and the theremin idly playing.



```
18.615
18.871
18.876
19.279
19.411
19.416
19.430
19.279
19.013
18.771
18.524
18.277
17.874
17.499
17.114
16.739
```

Figure 7: Arduino Serial Monitor of measured distance in cm

## 2 The Results

In order for the Arduino to correctly read the amplitude of the Ping response, a demodulation circuit was implemented right after the band pass filtering. We also implemented another low pass filter to remove any remaining 40kHz signal after demodulation. The results are shown in Figures 9, 9, and 10. See video attached in folder to see the theremin in action.

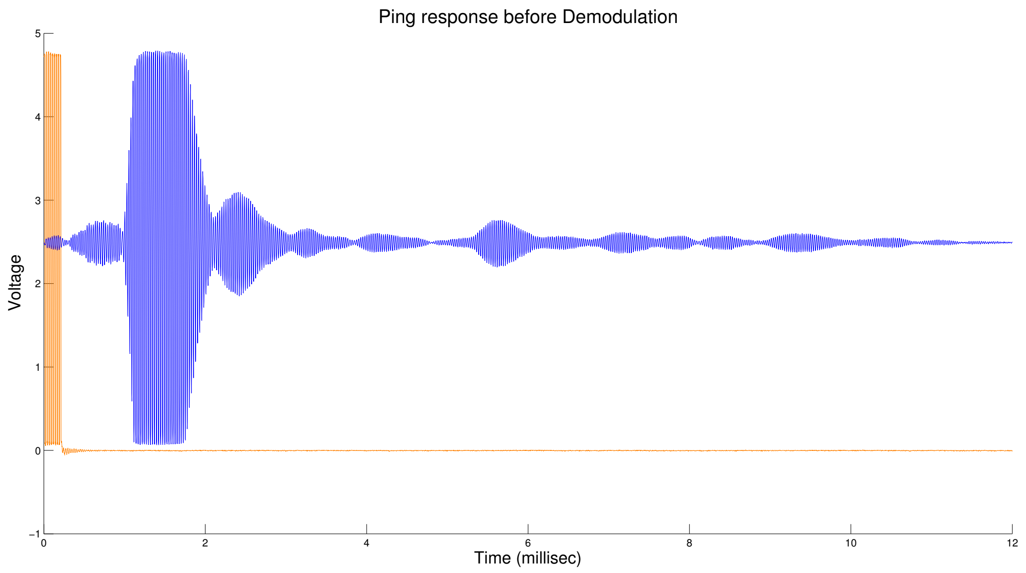


Figure 8

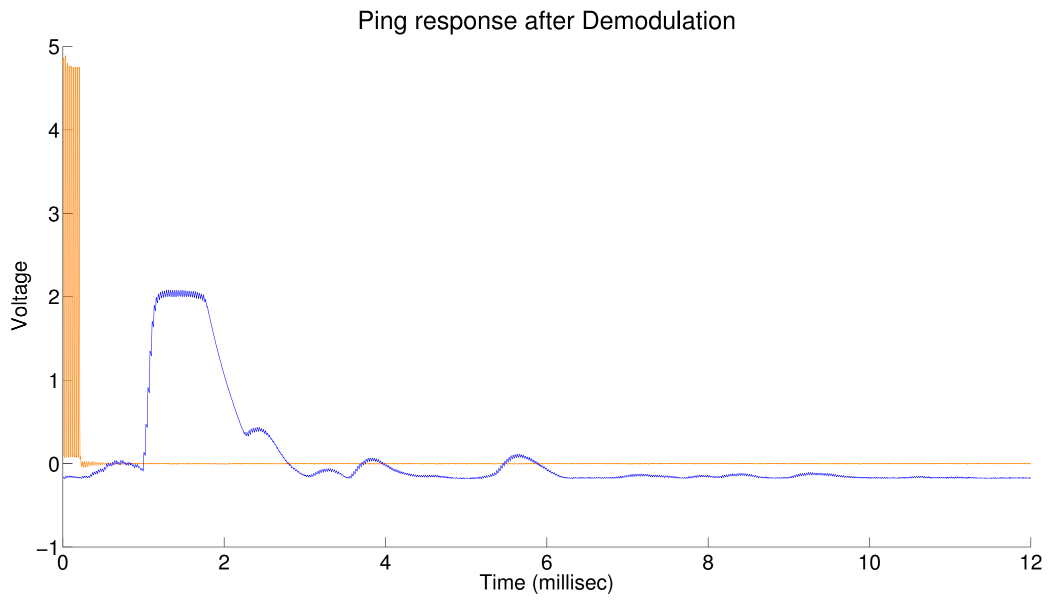


Figure 9

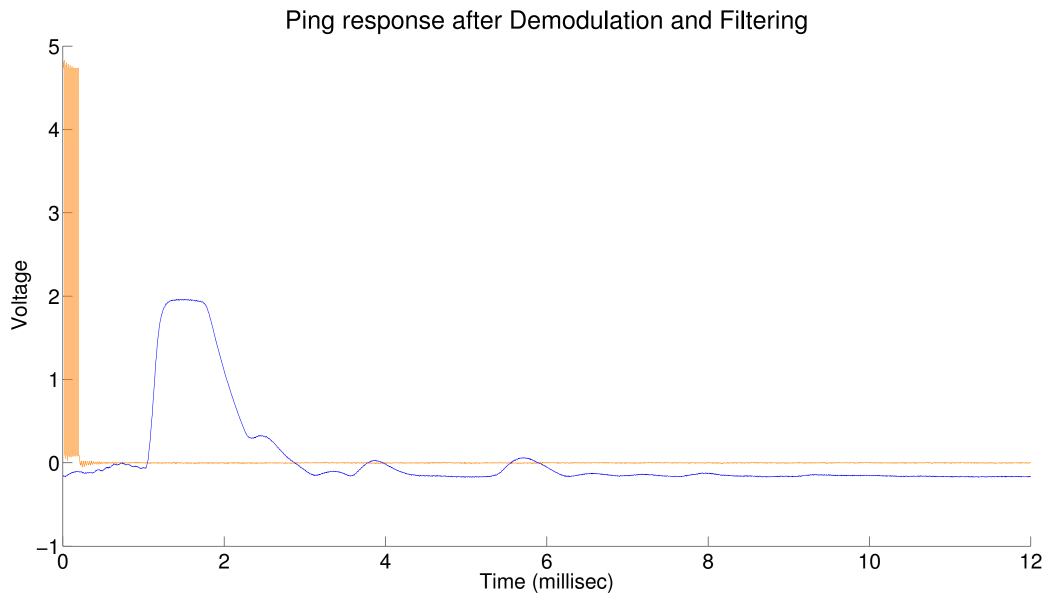


Figure 10

## 3 Reference - Arduino Code

*theremin.ino*

```
void delayMilliseconds(int ms) {
  for (int i = 0; i < ms; i++) {
    delayMicroseconds(1000);
  }
}

void stopTransducer()
{
  cli();
  TCCR1B = 0;
  sei();
  digitalWrite(9,LOW);
  digitalWrite(10,LOW);
}

void startTransducer(float freq, float dutyCycle)
{
  if (dutyCycle > 0.5) dutyCycle = 0.5;
  else if (dutyCycle < 0) dutyCycle = 0;

  cli();

  TCCR1B = _BV(WGM13) | _BV(CS10) | _BV(ICNC1);
  long topv = (long) ((float) F_CPU /(freq * 2.0 * 1.0));
  ICR1 = topv;

  OCR1A = (int) ((float) topv * dutyCycle);
  OCR1B = (int) ((float) topv * (1 - dutyCycle));
  DDRB |= _BV(PORTB1) | _BV(PORTB2);
  TCCR1A = _BV(COM1A1) | _BV(COM1B1);

  sei();
}

const float SPEED_OF_SOUND_20C = 0.0003432; //per micro-second
const int RESTING = 33;
const int NUMREADINGS = 15;
int a = 0;
unsigned long t_start = 0;
unsigned long t_peak = 0;
unsigned long t = 0;
int v_peak = 0;
float d = 0;
int index = 0;
float total = 0;
float avgDist = 0;
float pastDist[NUMREADINGS];
int play;
int numRelevant = NUMREADINGS;
```

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

  pinMode(10, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(8, OUTPUT);

  for(int i = 0; i < NUMREADINGS; i++)
    pastDist[i] = 0;
}

void loop()
{
  startTransducer(40150.0, 0.5);
  delayMicroseconds(200);
  stopTransducer();

  v_peak = 0;
  t_start =micros();
  t_peak = t_start;

  for (int i = 0; i < 120; i++) {
    a = analogRead(A0);
    t = micros();

    if (a > v_peak && abs(v_peak - a) > 8) {
      t_peak = t;
      v_peak = a;
    }
  }

  t = t_peak - t_start;
  if (v_peak > RESTING + 50) {
    d = (float) t * SPEED_OF_SOUND_20C / 2.0 * 100.0;
  }
  else {
    d = -1;
  }

  // subtract the last reading if it is not -1:
  if(pastDist[index] > -1)
    total = total - pastDist[index];
  else
    numRelevant++; // keep track of how many non -1 values we measured

  // read from the sensor:
  pastDist[index] = d;

  // add the reading to the total if it is not -1:
  if(pastDist[index] > -1)
    total = total + pastDist[index];
  else
    numRelevant--; // keep track of how many non -1 values we measured
  // advance to the next position in the array:
  index = index + 1;
  // if we're at the end of the array...
  if (index >= NUMREADINGS)
    index = 0; // ...wrap around to the beginning

  // calculate the average:
  avgDist = total / numRelevant;

  //Serial.println(v_peak);
  Serial.println(avgDist, 3);

  if(d > 0 && numRelevant > NUMREADINGS/2 && avgDist <= 50)
  {
    play = map(avgDist*100, 5*100, 50*100, 1047, 262); //C6: 1047
    tone(8, play);
  }
  else
  {
    noTone(8);
  }
}
```