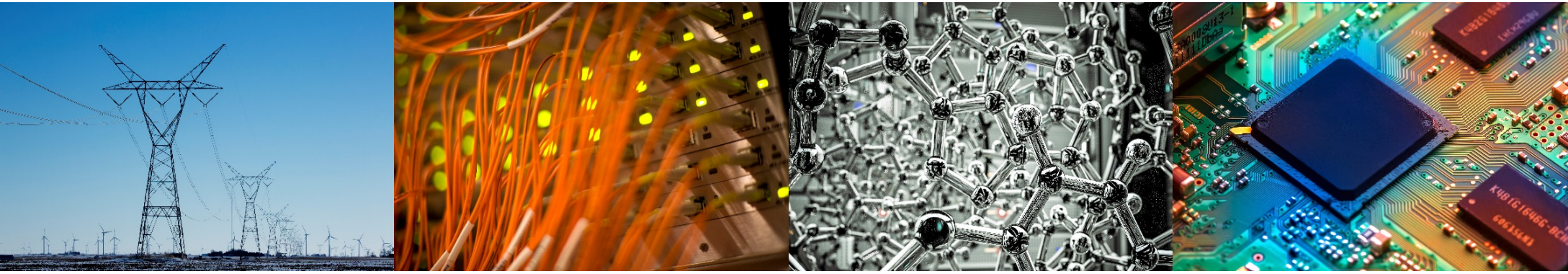
ThereminFreaks – Theremin Rhythm Game

Group 34
Esteban Looser-Rojas
Daniel Olivas Hernandez
Michael Recklein



ILLINOIS
Electrical & Computer Engineering
COLLEGE OF ENGINEERING

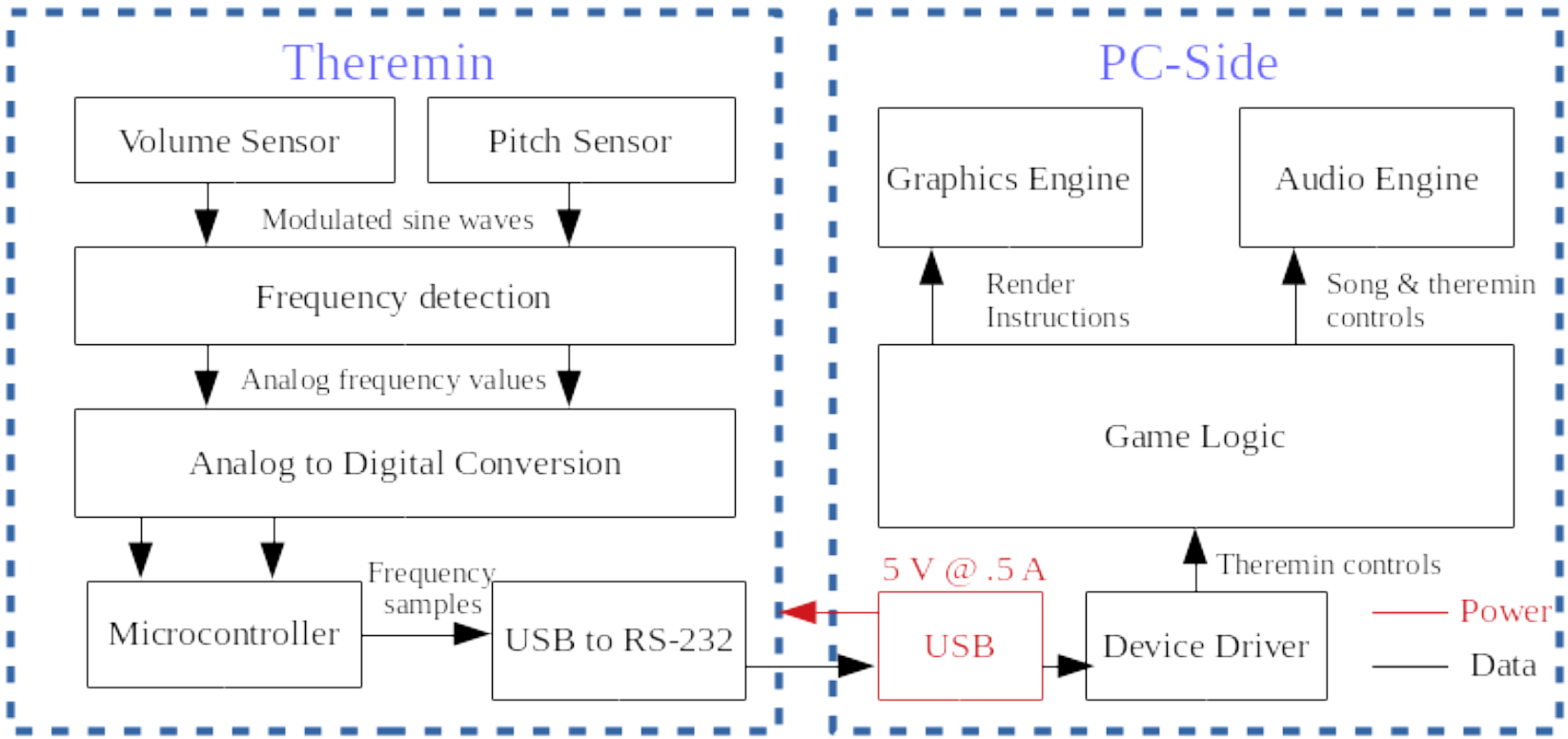
Introduction

- Desire to make a video game with unique hardware
- No rhythm games out there for simulating theremin
- Very unique instrument to make a rhythm game for: no contact + wave hands around

Objectives

- Create realistic and intuitive theremin simulator
- Responsive and enjoyable game
- Sound engine sounds like theremin and controls like one

Block Diagram

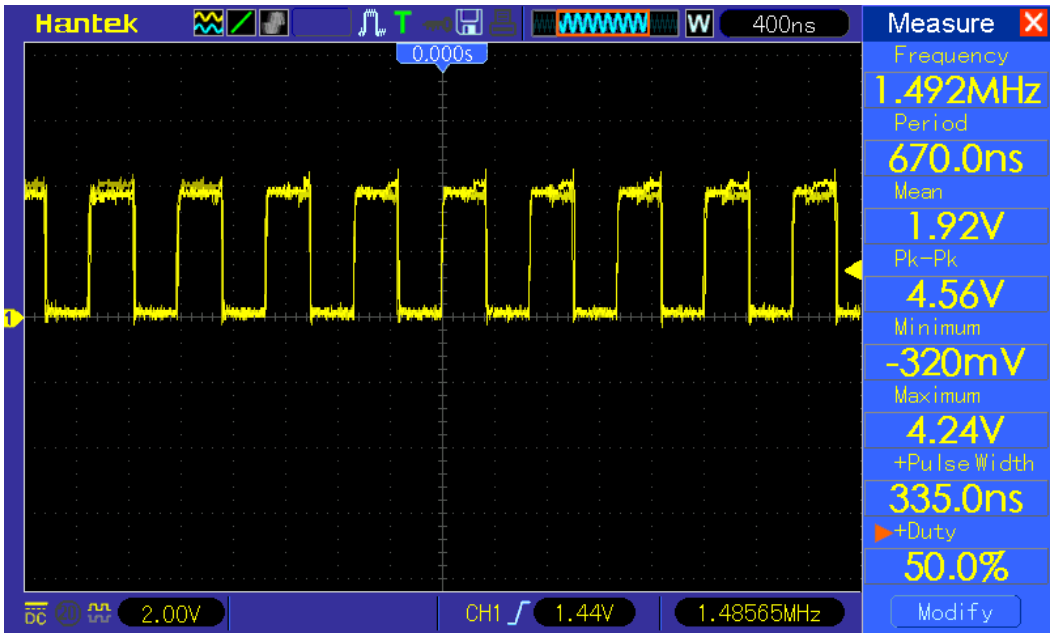


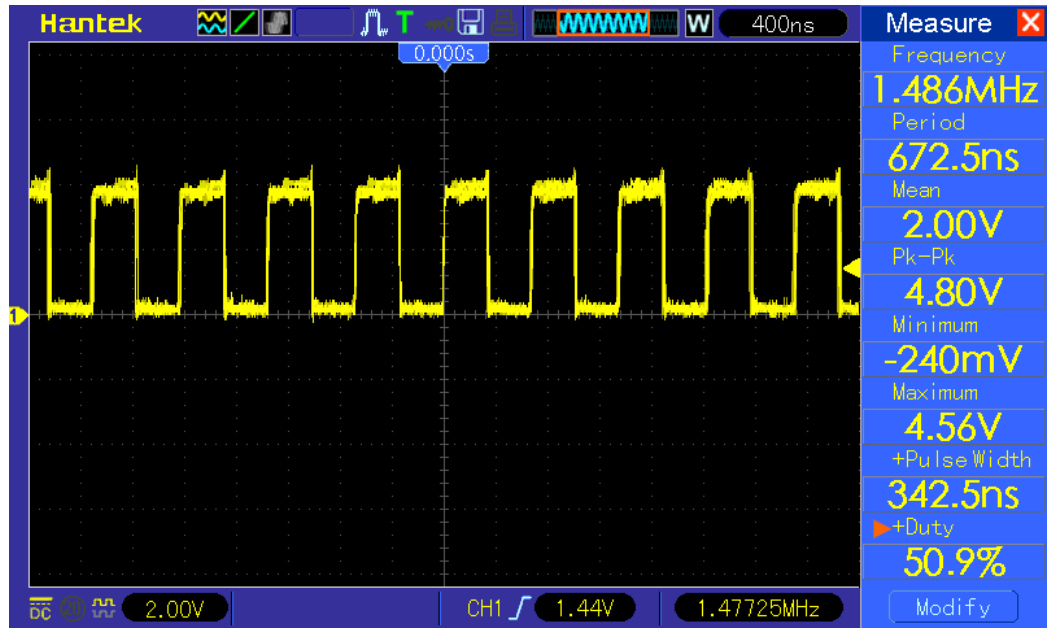
Pitch and Volume Sensors

- Volume antenna: 6” wide; 10” long; 3/8” dia
  Pitch antenna: 2’ long; 3/8” dia
- Design taken from DIY theremin guide
- Problem: unstable oscillators
- Theory: antennae load too large
- Solution: shorter antennae; better oscillator design

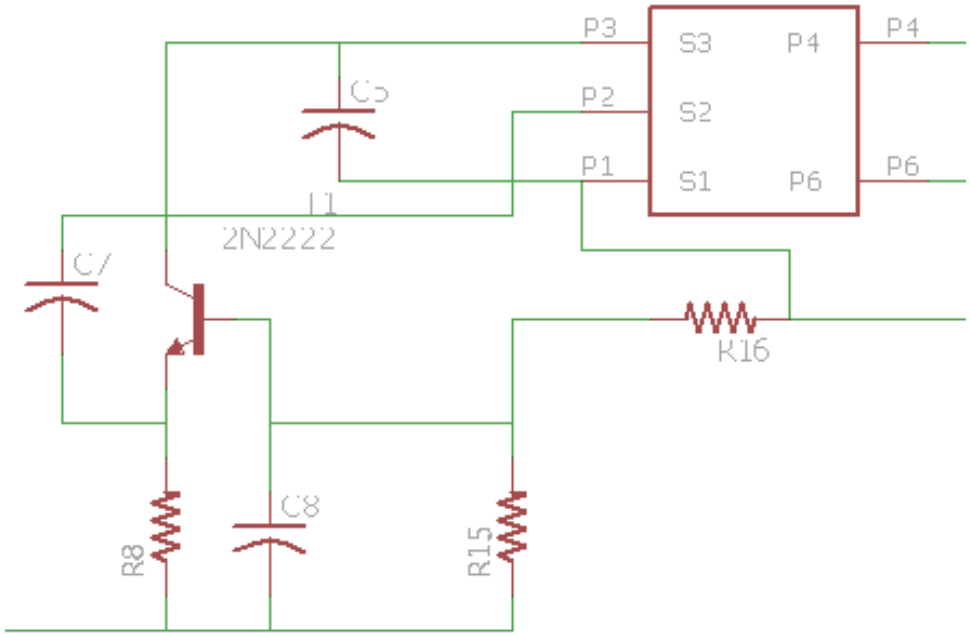
Oscillators

- First used Hartley oscillators w/SMD transformer – very weak signal
- Change: Hartley → Colpitts (thru-hole inductors)
- Requirement: Antennae change osc. freq by 5-10 kHz from base freq
- Measured antenna freq from no hand near (left) to hand near (right): ~6 kHz difference

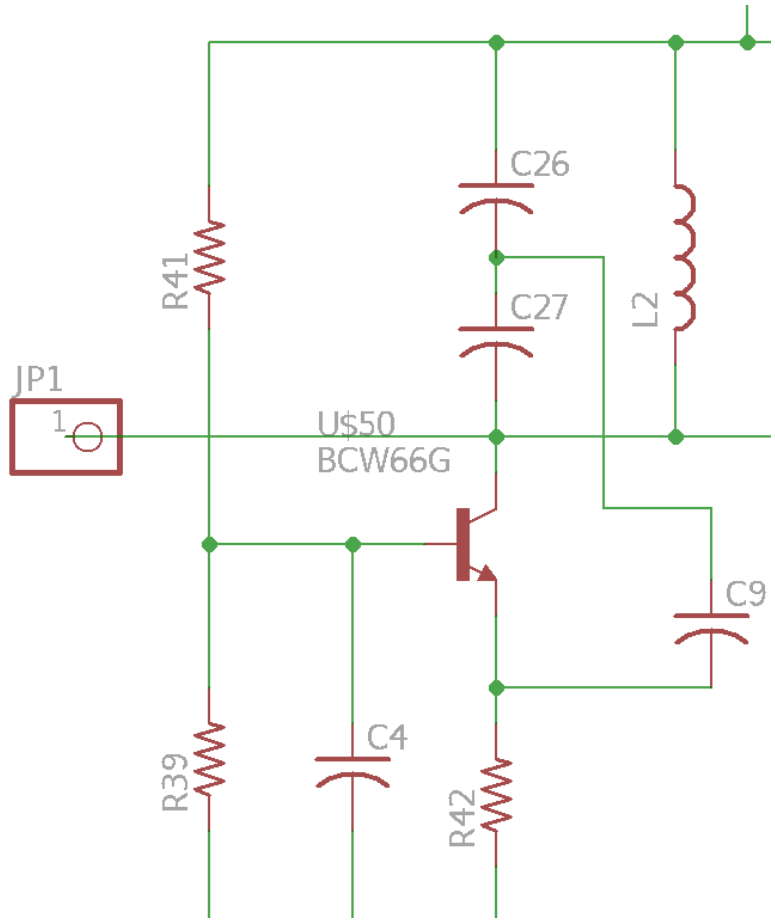




Old design: Hartley Oscillator

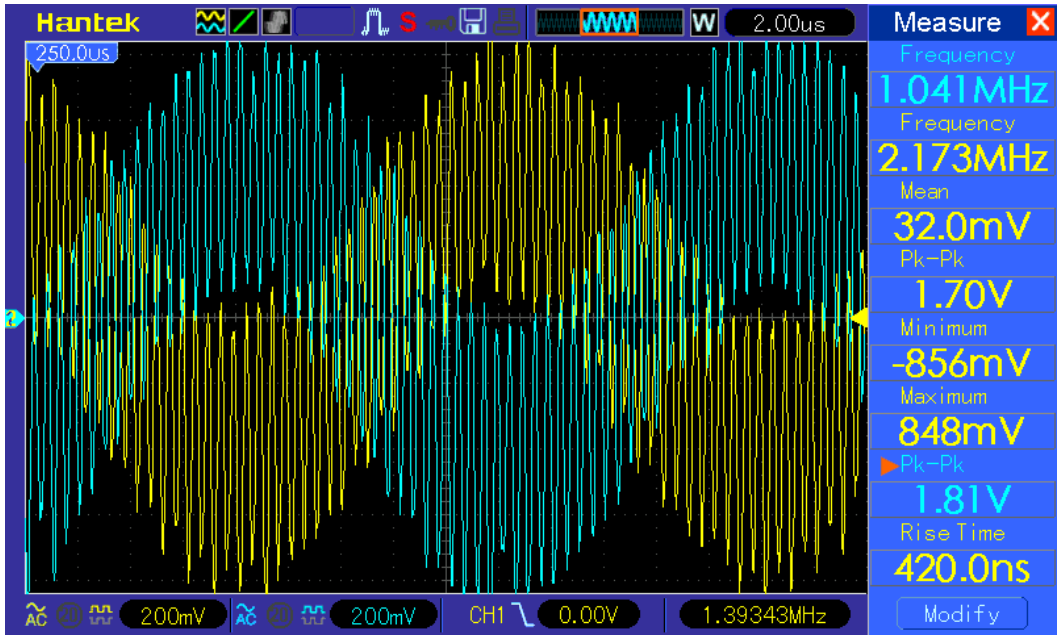


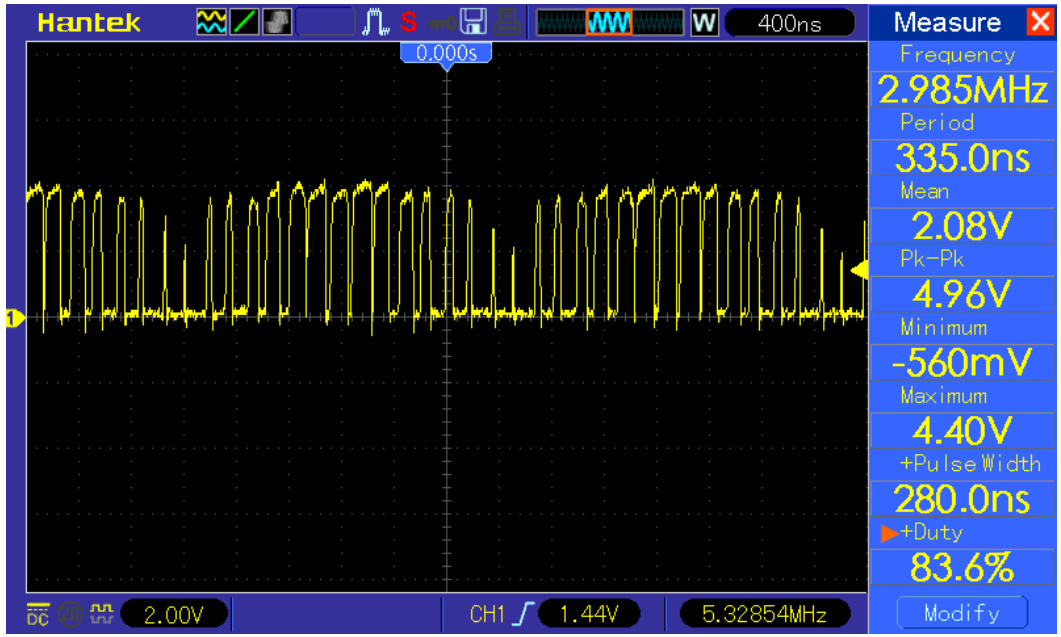
New design: Colpitts Oscillator



Mixers

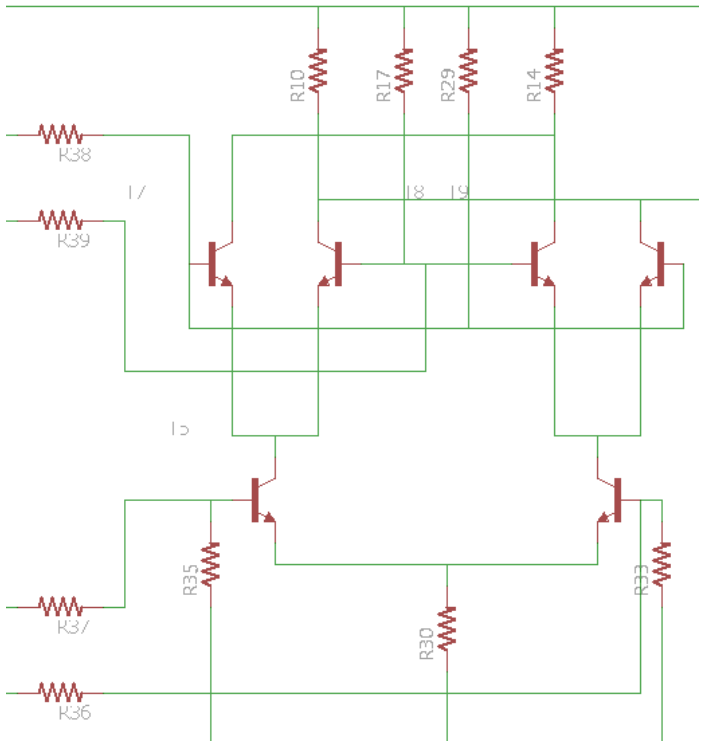
- Initially used Gilbert cells (on left) – sensitive; low output (1.7V pk-pk)
- Change: oscillator sine wave → square wave
- Requirement: modulated waves from 1-4 MHz input at logical high (2-5V)
- Use XOR gate to mix signals (on right) – get clean 5V output

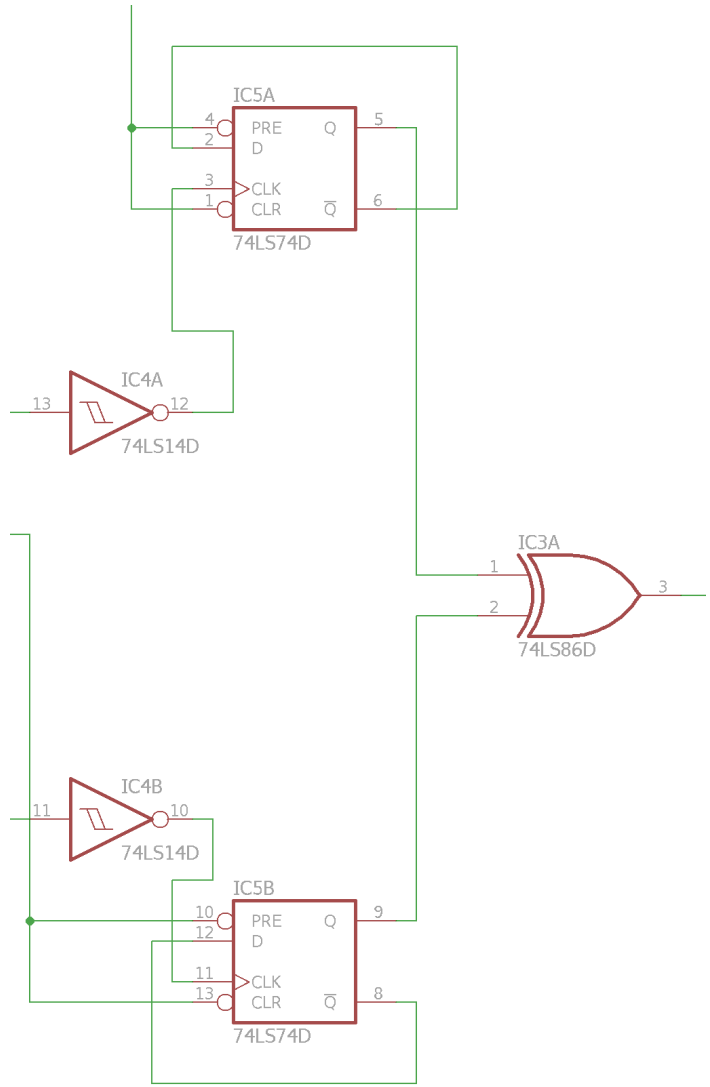




Square Wave & Digital XOR Mixer

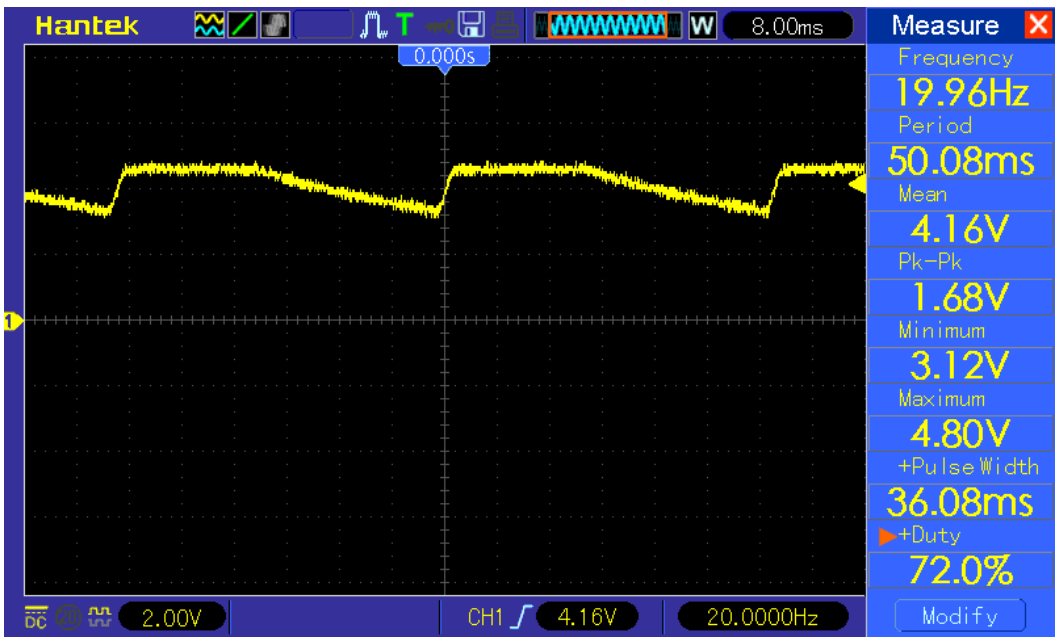
Gilbert Cell

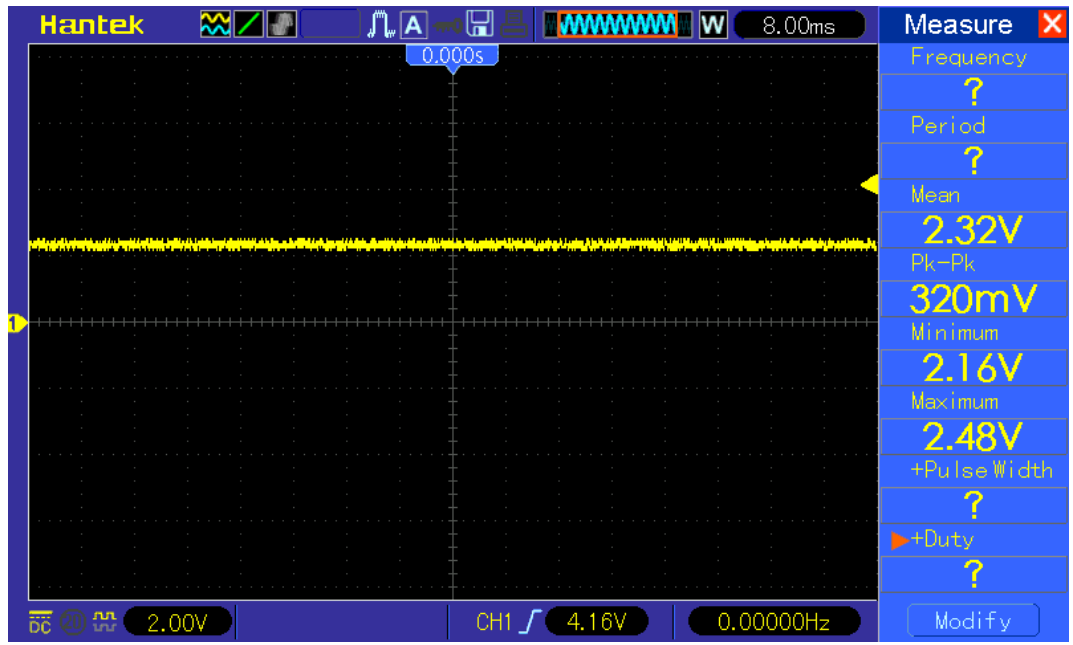


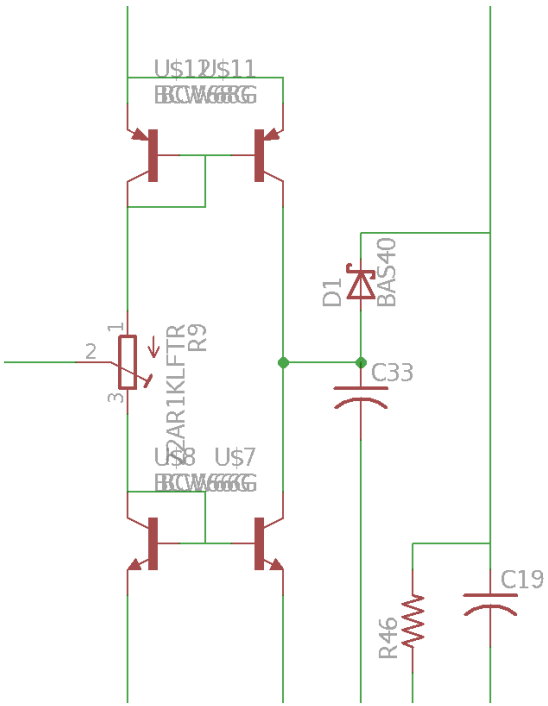


Detector and Integrator

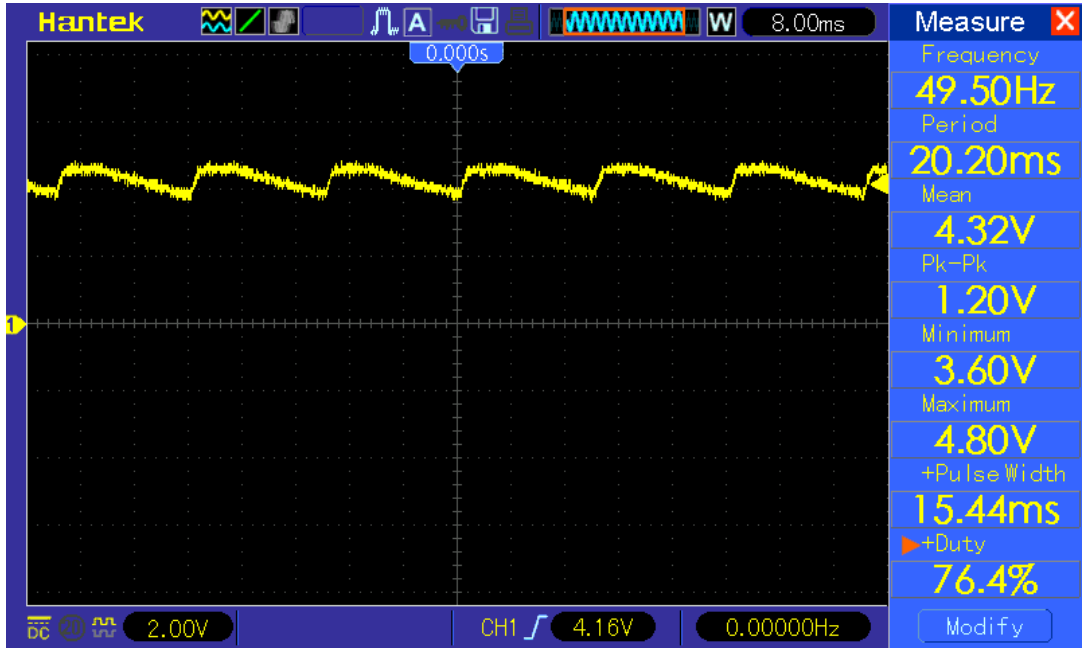
- Use BJTs for integrator circuit
- Simple diode and RC lowpass filter for detector
- Requirement: 30 kHz bandwidth
- Ran func. generator through 20 Hz - 30 kHz range



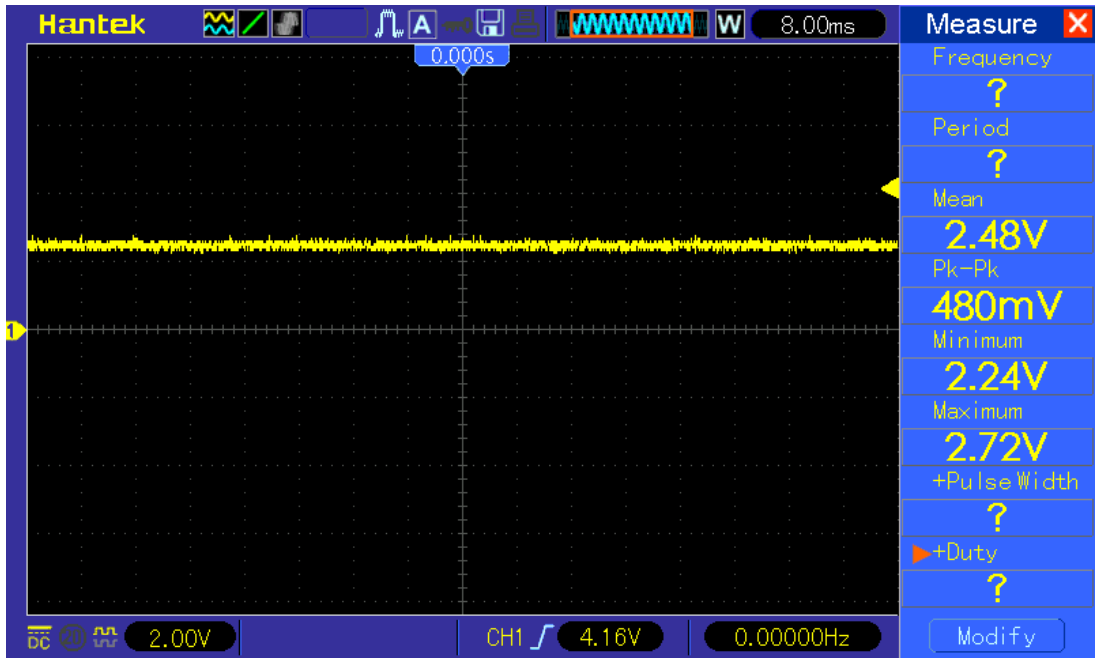




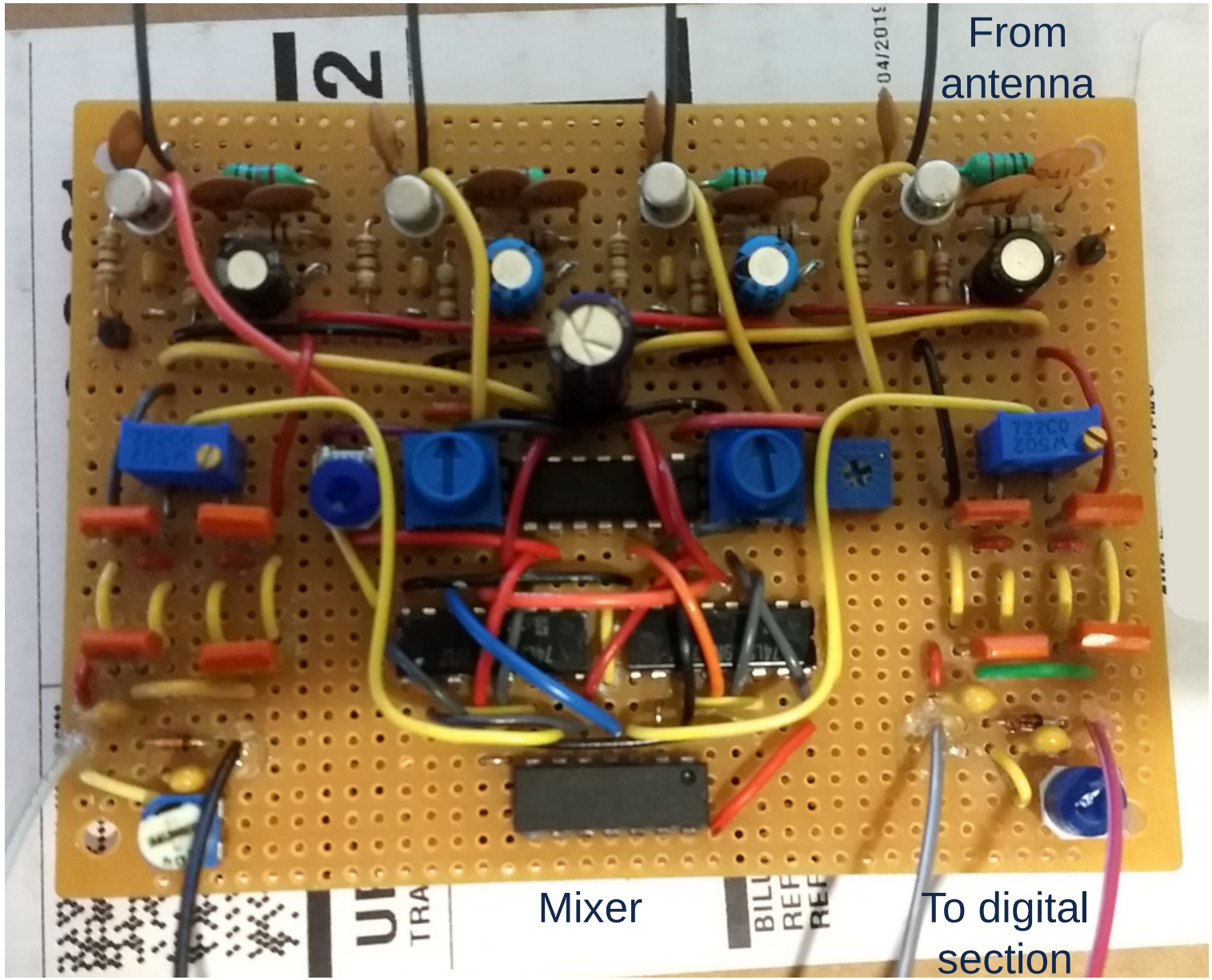
Integrator and Detector



Above: 50 Hz input
Below: 2 kHz input

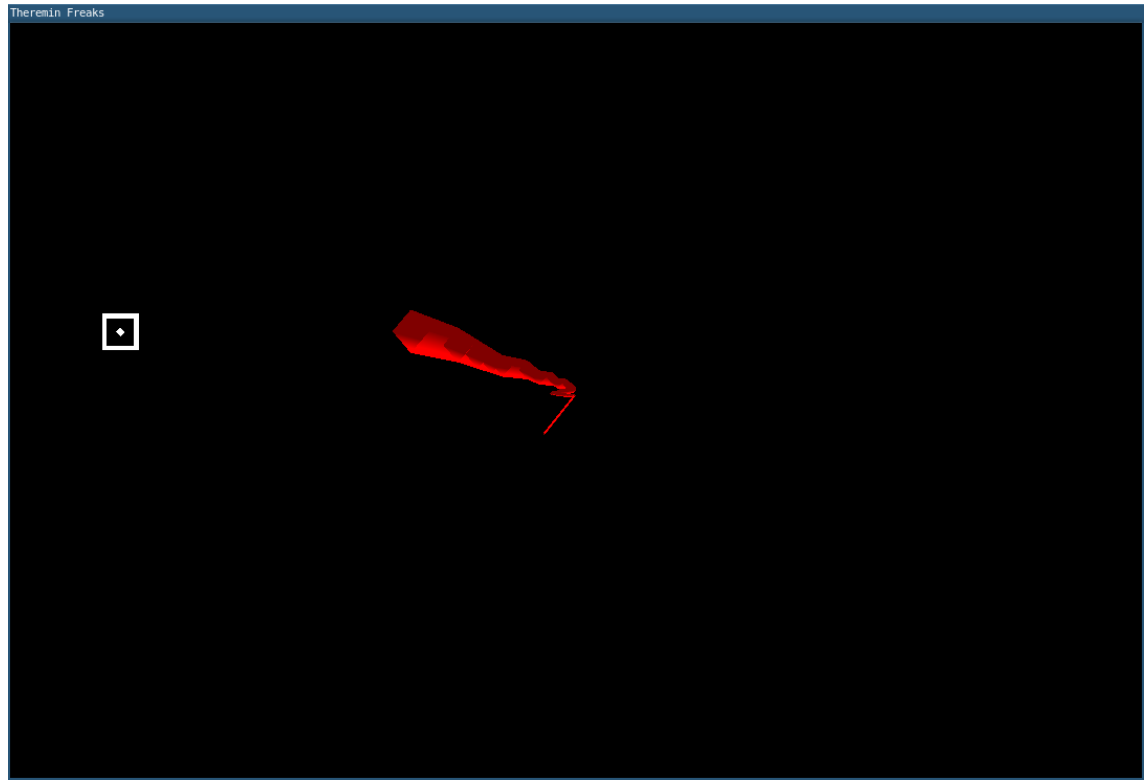


Analog Circuit



Game Design: Rendering Engine

- Create mesh to signal required pitch and volume
- Generated from text file + primitive meshes
- Requirement: want ~60 fps consistently – achieved (checked FPS counter)

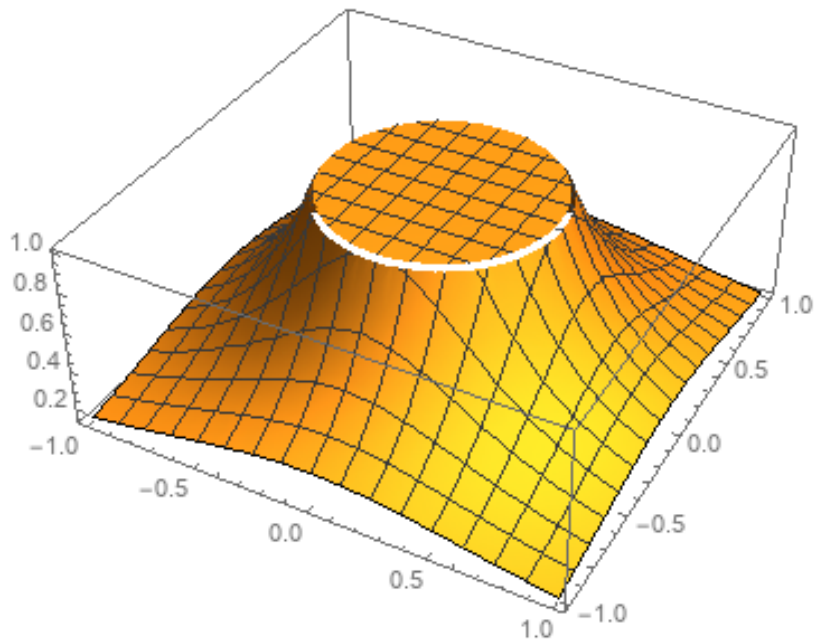


Game Design: Game Logic

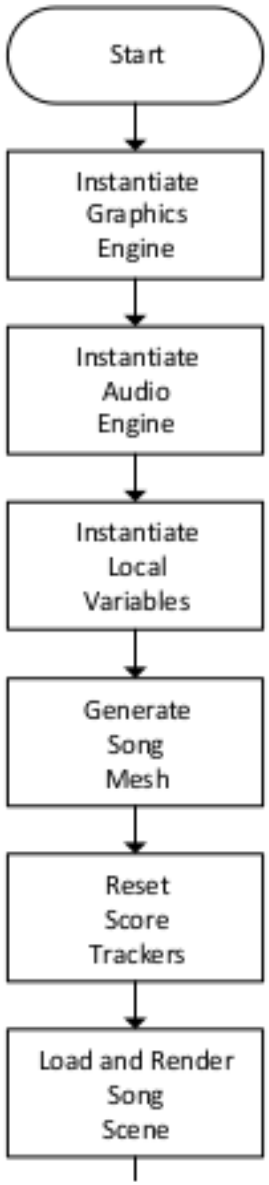
- Decoupled frame rate from game speed
- Scoring function based off cubic distance from “safe zone”
- Forgives shaky hands and sampling noise

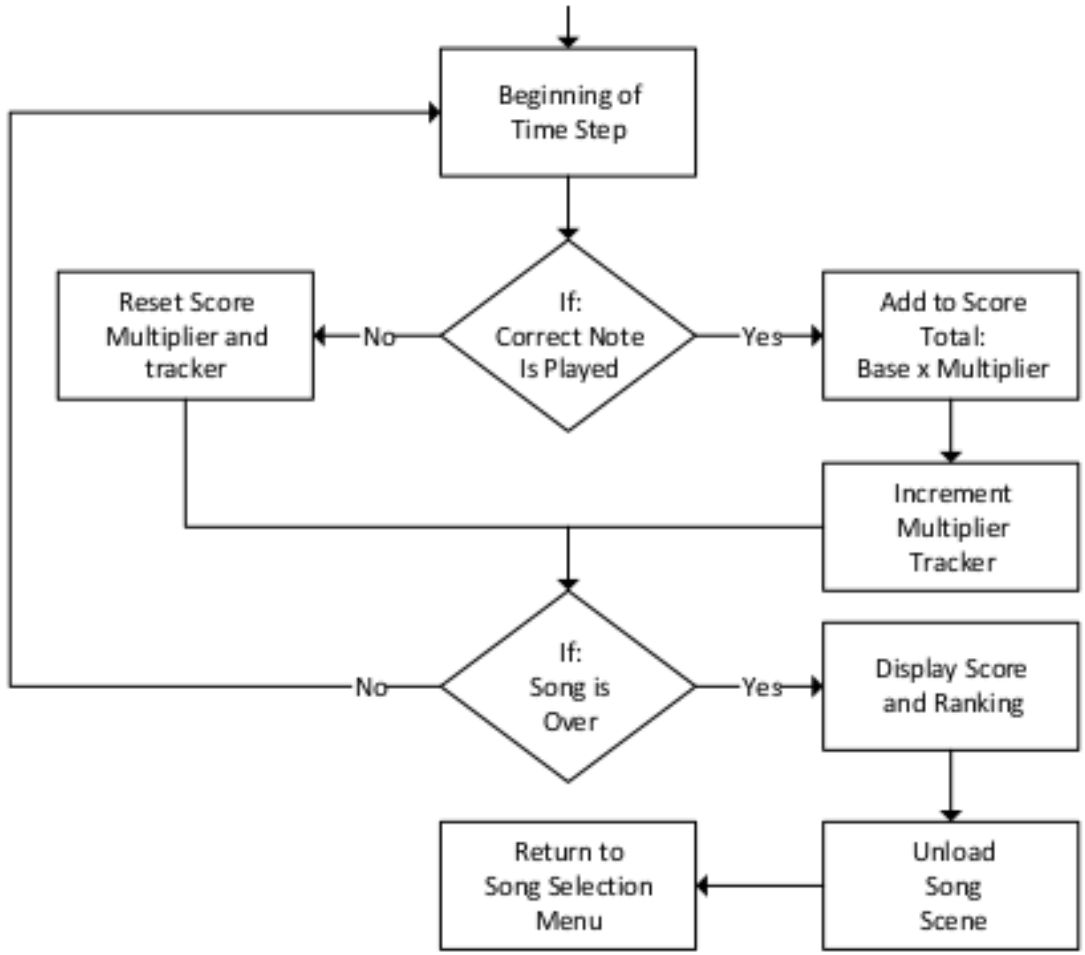
```
f = Min[1/(Sqrt[x^2 + y^2] + .5)^3, 1]
Plot3D[f, {x, -1, 1}, {y, -1, 1}, PlotRange → All]
```

$$\text{Min}\left[1, \frac{1}{\left(0.5 + \sqrt{x^2 + y^2}\right)^3}\right]$$



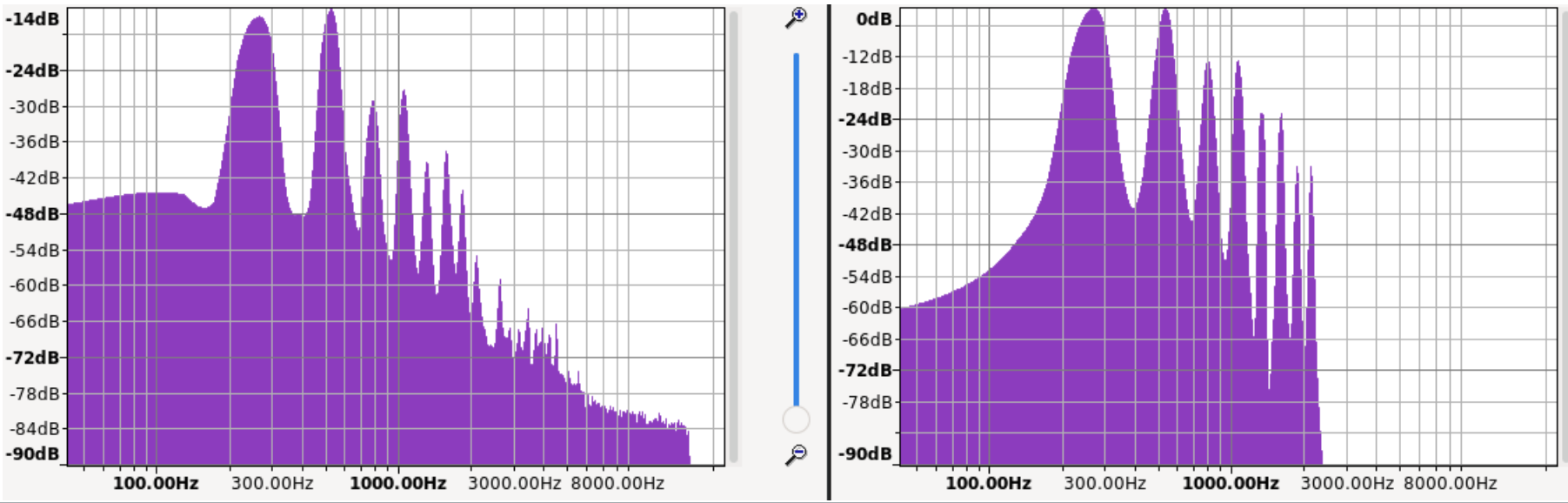
Game Flowchart





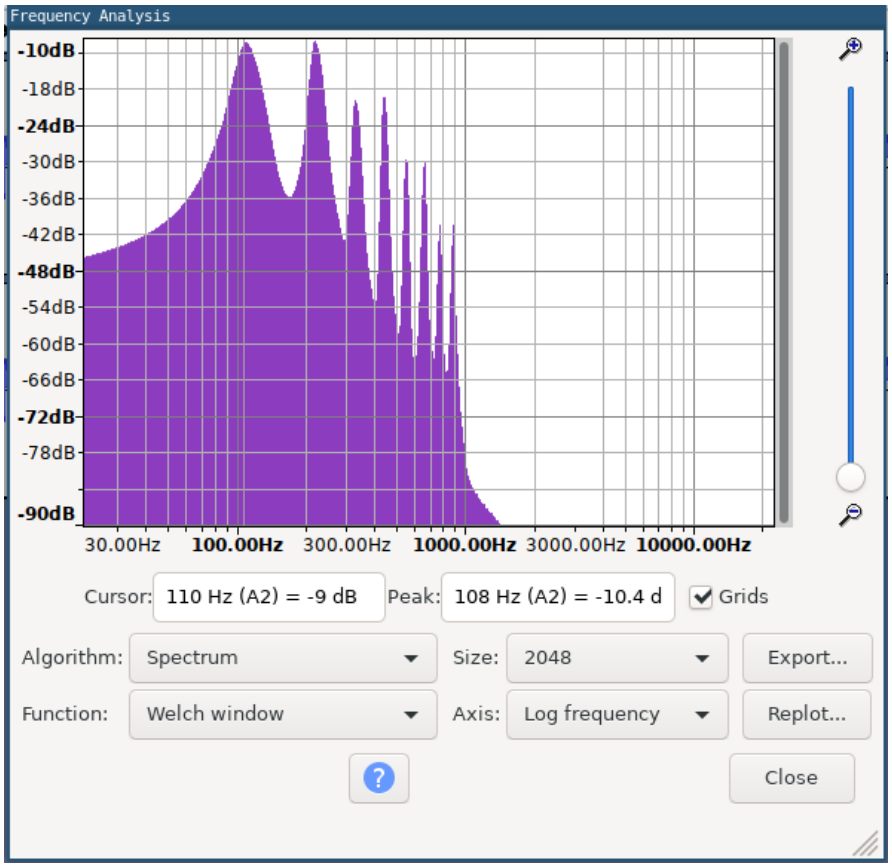
Audio Engine

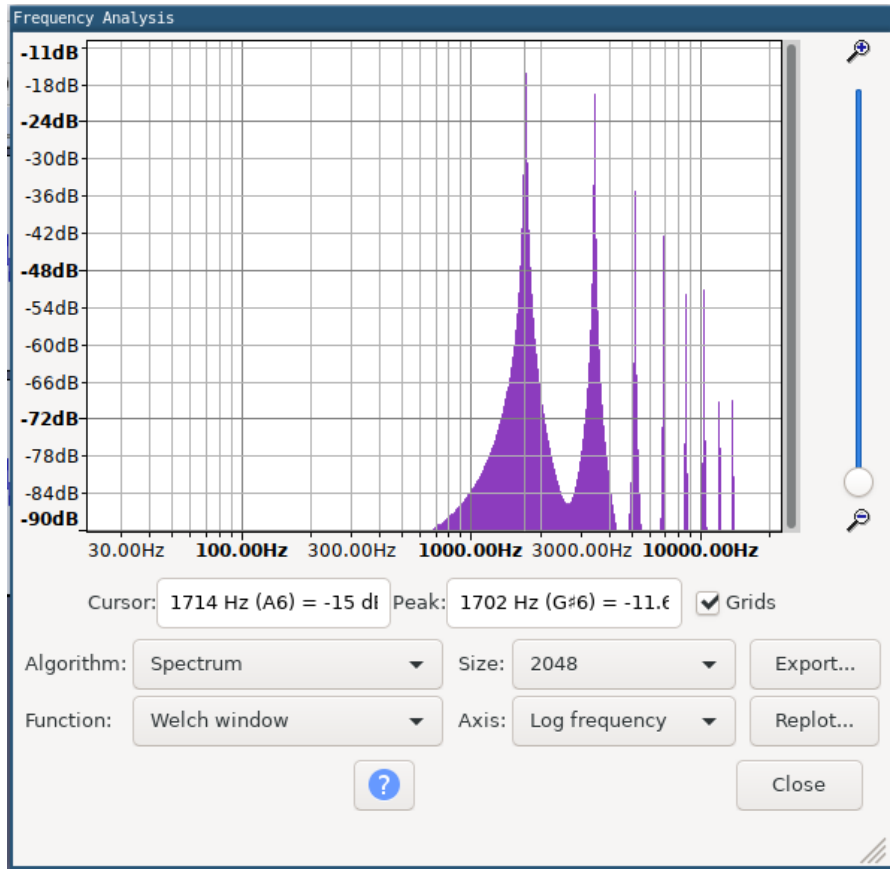
- Take ADC samples from driver and scale to a certain frequency range
- Analyzed recorded theremin sound with FFT (left)
- Simulate theremin sound using additive synthesis (right)



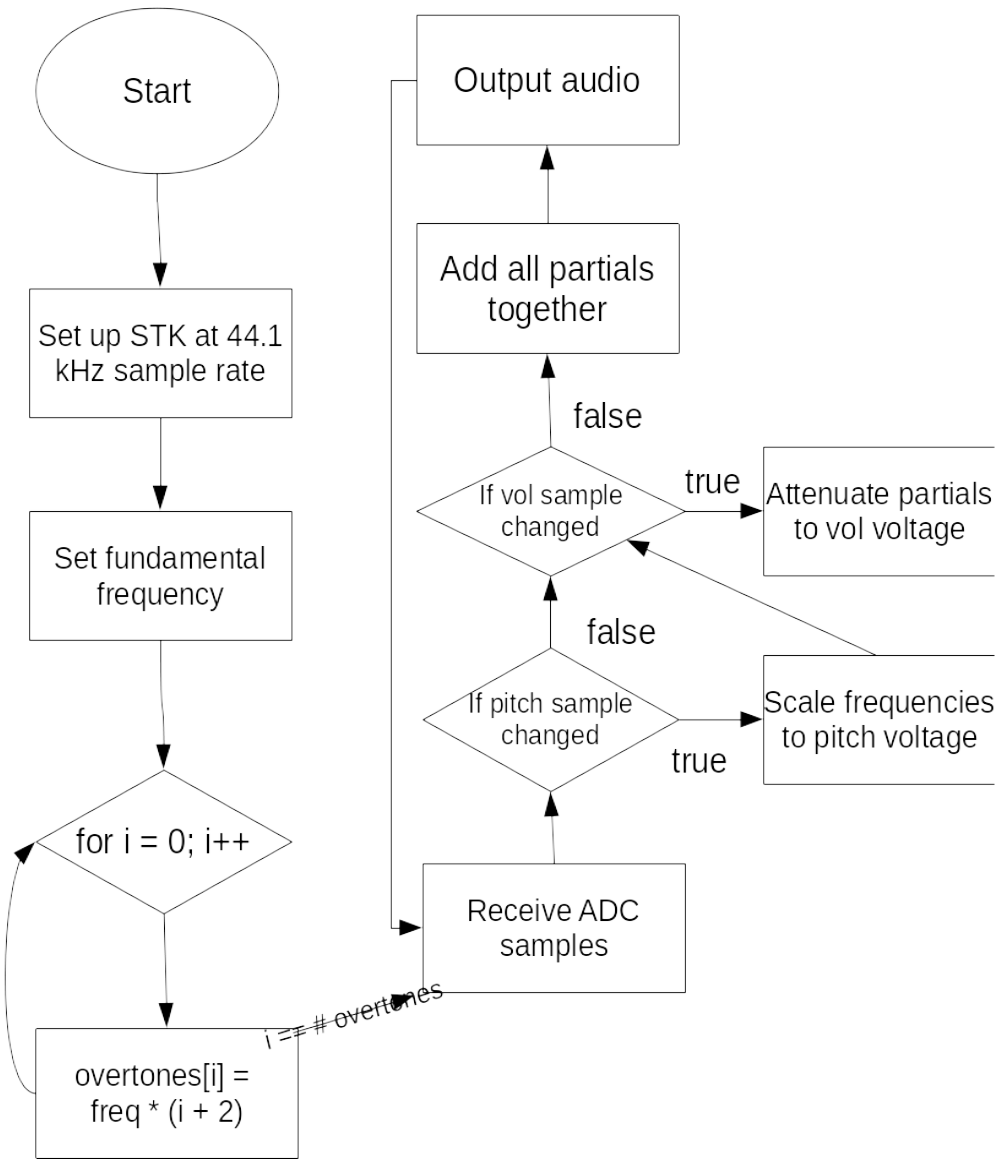
Audio Engine cont.

- Requirement: have at least four-octave sound range
- Used 110 Hz as lowest note (A2, left) and 1710 Hz as highest note (A6, right)
- Five-octave sound range result





Synthesis Flowchart



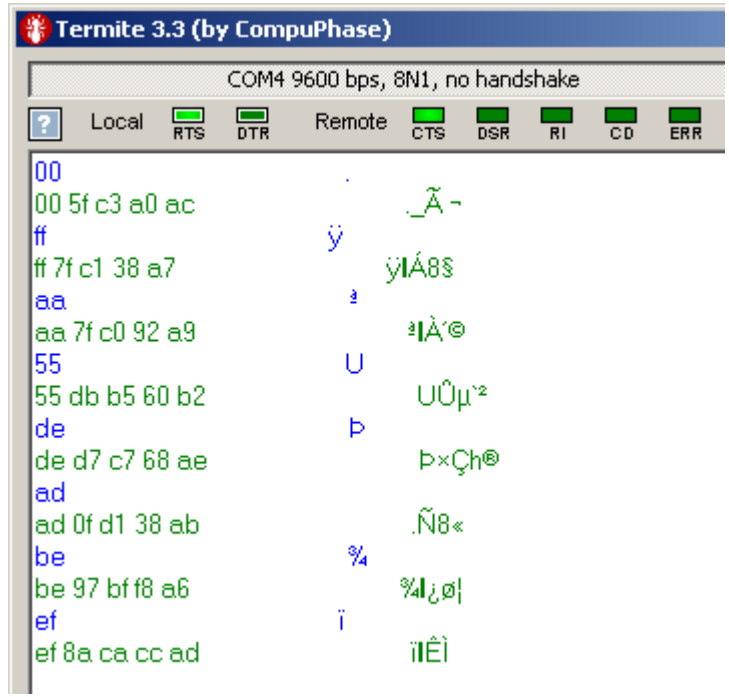
Analog-to-Digital Converters

- Off-the-shelf 16-bit Maxim Integrated σ-δ converters

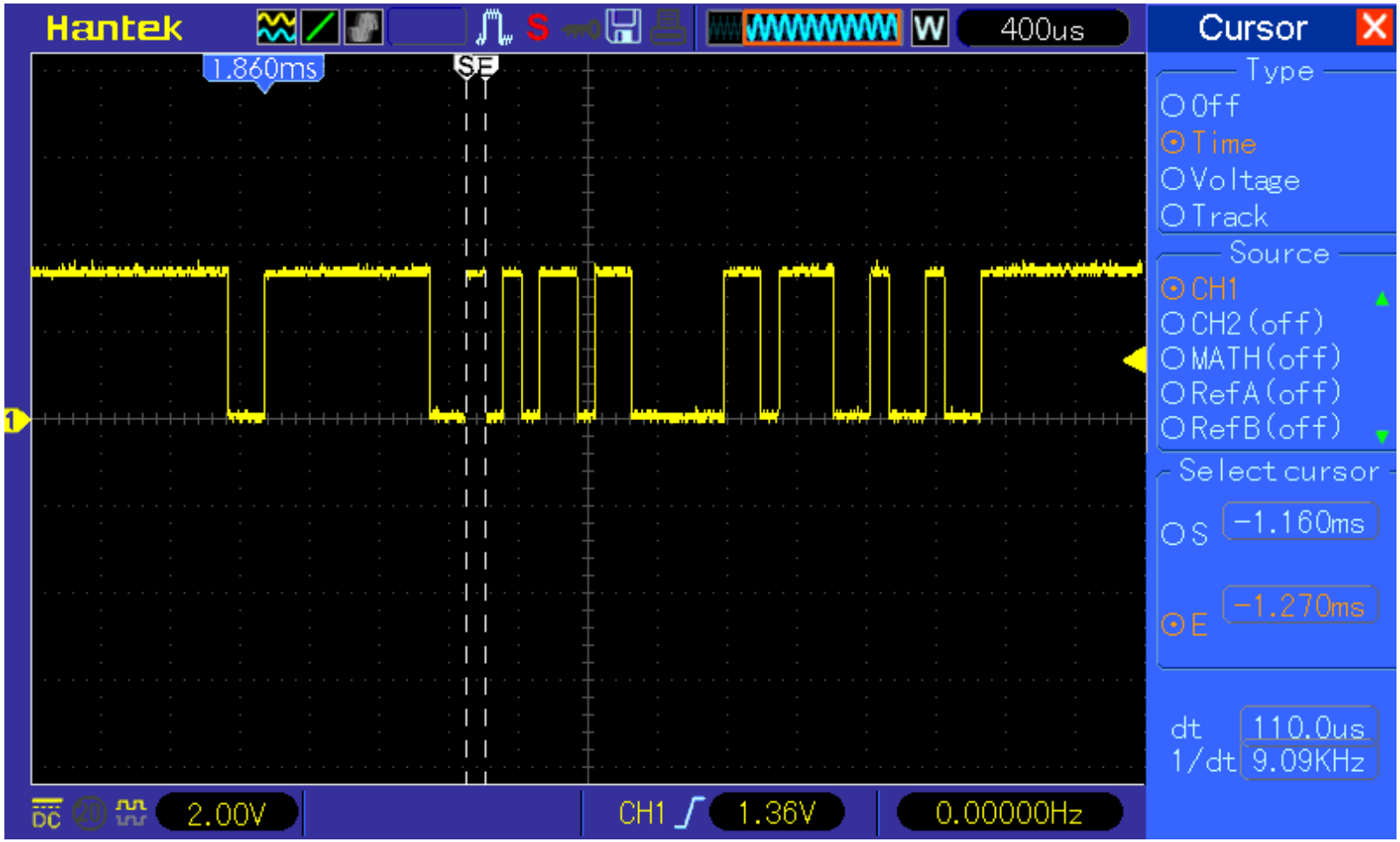
Requirement	Verification
• ADCs can send at least 480 samples per second	▪ Connect trimmer to ADC; ADC to Arduino Program Arduino to manipulate ADC and send voltage reading Verify trimmer position corresponds to value from ADC

PIC16 and RS-232 to USB chip

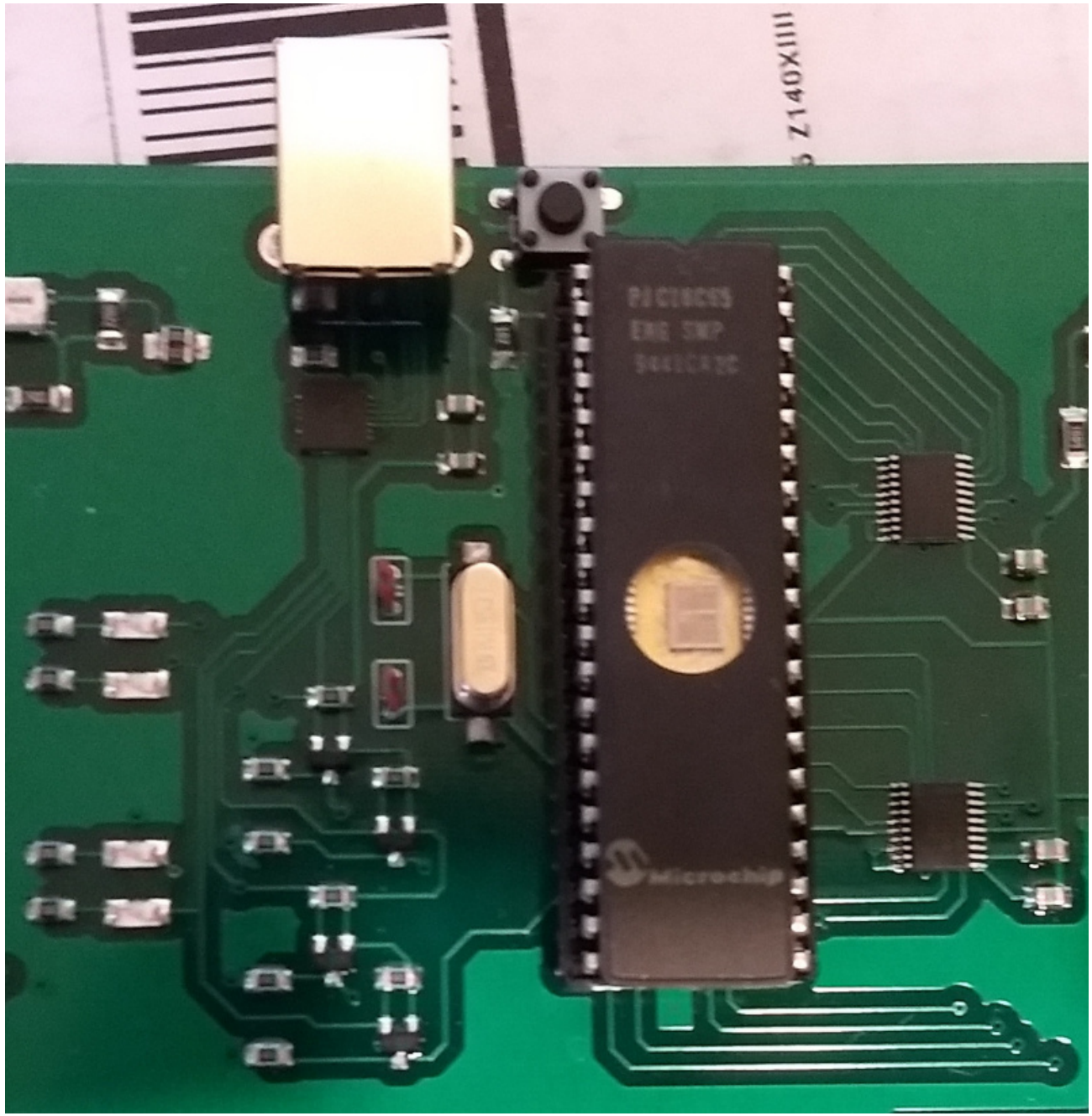
- PIC16 signals to ADC to take sample
- PIC16 then sends four bytes to PC thru RS-232 UART
- Requirement: PIC16 able to send samples from ADC to RS-232 interface at 9600 baud
- Verification: Send byte to PIC16, get same byte back + both ADC samples



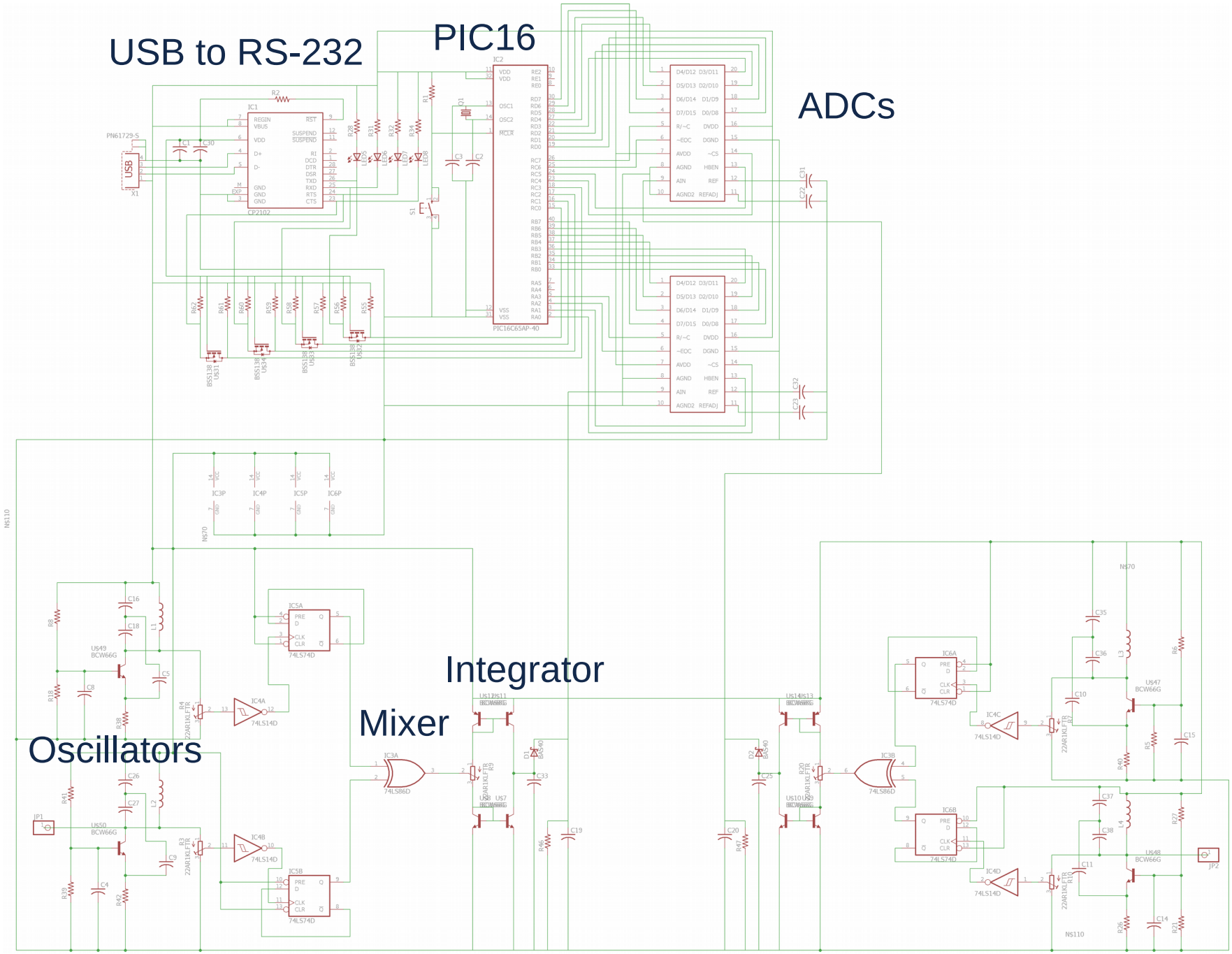
Oscilloscope Trace of RS-232 Transmission



Digital Circuit



Overall Schematic



Device Driver

- Uses system calls to open serial device and read/write from/to theremin controller
- Requirement: provide 2 16-bit samples from theremin with delay < 50 ms
- Verification: take time difference between PIC sampling input and driver receiving input
- ~9.5 ms delay

```
~/U/E/E/usb driver $ stty -F /dev/ttyUSB3 cs8 9600 i
non -iexten -echo -echoe -echok -echoctl -echoke nof
~/U/E/E/usb driver $ ./test /dev/ttyUSB3
transmission time: 9623
voltage level from top ADC, right of PCB: 4.096000
voltage level from bottom ADC, left of PCB: 4.096000
~/U/E/E/usb driver $ ./test /dev/ttyUSB3
transmission time: 9552
voltage level from top ADC, right of PCB: 3.977811
voltage level from bottom ADC, left of PCB: 3.953060
~/U/E/E/usb driver $ ./test /dev/ttyUSB3
transmission time: 9475
voltage level from top ADC, right of PCB: 3.720307
voltage level from bottom ADC, left of PCB: 3.803058
~/U/E/E/usb driver $ ./test /dev/ttyUSB3
transmission time: 9498
voltage level from top ADC, right of PCB: 3.525304
voltage level from bottom ADC, left of PCB: 3.850059
~/U/E/E/usb driver $ ./test /dev/ttyUSB3
transmission time: 9591
voltage level from top ADC, right of PCB: 3.322551
voltage level from bottom ADC, left of PCB: 3.591055
```

Conclusions

- Oscillators still need work done (detailed in next slide)
- Digital section of theremin reliable
- Synthesis engine decent
- Game barebones but working

Future Work

- Fix transistor biasing with inductors
  before biasing inductor 6V pk-pk (left); 32.4V pk-pk after (right)
- Improve upon sound synthesis engine
- Flesh out game
- Actual controller enclosure and rigid antennae