Analog Theremin with Audio Processing

Karthik Achar
Nikhil Mehta

November 27th, 2018

ECE 395: Advanced Digital Projects Lab
University of Illinois at Urbana-Champaign
Abstract
Our goal with this project was to build an analog Theremin, based off of Bob Moog’s Etherwave Theremin design, and apply our own real-time audio processing to the resulting audio signal using an Teensy 3.2 development board.
I. Introduction

The Theremin is an electronic musical instrument invented by Léon Theremin in 1920. It consists of several analog voltage oscillators that control the pitch and volume of an output tone. These parameters are controlled by moving your hands around the protruding antennae, one for pitch and one for volume. As you adjust your hand positioning, you can adjust the characteristics of the output tone. As seen in Figure 1 below, the long vertical antenna controls pitch, and the bent horizontal antenna controls volume.

![Figure 1: Clara Rockmore, one of the most famous thereminists, playing an early theremin.](image)

Each protruding antenna functions as one plate of a variable capacitor, with your hand being the ground plate. As you move your hands back and forth, you are adjusting capacitance, and in turn, adjusting the variable pitch and volume oscillators. This is what causes the output tone to change.

In addition to the analog theremin, we also wanted to add a digital component to this project. We decided to construct the theremin as a standalone analog piece, but add some digital, real-time processing to the output. This would be accomplished by feeding the direct line output of the theremin into a Teensy 3.2 development board, sampling the signal, applying digital processing effects, and converting it back to analog for output. The Teensy 3.2 development
board has an extensive library of functions and capabilities for audio processing and analysis, which we used for our digital processing.

II. Analog Design

The theremin has been refined over the years, and rather than attempt to start from scratch, we based our circuit design on Bob Moog’s Etherwave theremin schematic. Bob Moog is considered the grandfather of modern synthesizers, and his Etherwave theremin schematic and construction details are publicly accessible\(^1\).

\[\text{https://www.cs.nmsu.edu/~rth/EMTheremin.pdf,}\]
\[\text{http://www.suonelettronico.com/downloads/HotRodEtherwav.pdf}\]
After deciding on this analog design, we started to look for the necessary parts, but had to use quite a few different components due to lack of availability; the layout in Moog’s original design includes quite a few parts that are no longer available and because of this we had to make some changes to the board layout. Once we ordered the parts, we began constructing the circuit based on a recommended proto board layout also described in the Hot Rod Etherwave documentation, as referenced above. A specific part that gave us trouble were the variable inductors - the Toko inductors mentioned in the original design are still available, but obsolete and have no datasheet. After some testing we determined that the only active pins on the components are the two end pins on the side of the inductor with 3 pins, as is somewhat outlined in Figure 3. We attempted to make very few changes to this layout, as the Etherwave is designed to minimize unwanted interactions between components.
Figure 4: Final layout for the main board

Figure 5: Final layout for auxiliary boards
In Figures 6 and 7, you can see our completed main circuit board consisting of the oscillators, voltage controlled amplifier, processor, and power supply. We also have two smaller boards for some hardware relating specifically to the antenna, mainly a set of inductors. These
boards are separate from the main board to minimize wire length to the antennae, and are pictured below in Figures 8 and 9.

Figure 8: Volume antenna board          Figure 9: Pitch antenna board

Following the construction of these components, we began to assemble the Theremin as a complete unit. We ordered an ABS board to mount and organize all the components, and connected the front panel components (aux out, power switch, and potentiometers) via a 10-pin connector onto the same auxiliary board as the pitch antenna circuit. The antennae were first soldered to wires that connected them to their respective boards, then hot glued to the ABS board. For the volume antenna, we drilled 3/8 in. holes into the board and inserted the antenna into the holes before securing it on both sides with hot glue to ensure that the connection held up.
Figure 10: Completed theremin board
III. Software and DSP

The goal of the software component of this project was to process the analog audio output of our theremin using a microcontroller. After initially researching the capabilities of the LPC1114 we were given in class for audio processing, we found that with the time we had remaining, a better option would be to purchase a development board with audio input and output capabilities already built-in, and use that instead. This solution would also allow us to use the benefits of I2S. In order to capture all of this necessary functionality, we decided to use a Teensy 3.2 development board based on the ARM Cortex M4 architecture.

Because it took most of our time to construct the analog theremin, and much of our software development depended on the theremin’s working output, we don’t have a fully functioning digital component of this project. More research and testing also needs to be done on the pitch and volume antenna boards, as they also do not function as intended, while the main board’s three oscillator circuits are working properly. Instead, we have a few sample programs to test audio functionality on the Teensy board, and explore some of the features included in its audio library. Our sample programs are included below.
IV. Code

In this code sample, we implemented the FFT, peak recognition, and note frequency analysis tools all included in the Teensy Audio Library. The library can be found on their website and is publicly available and open source. The below code receives an audio stream in from the USB (substituting the analog audio input from the theremin temporarily, this can be changed easily), analyzes it using the functions listed, and outputs the data to the serial monitor.

```c
#include <Audio.h>
#include <Wire.h>
#include <SPI.h>
#include <SD.h>
#include <SerialFlash.h>

// GUItool: begin automatically generated code
AudioInputUSB            usb1;           //xy=355,176
AudioAnalyzeFFT1024      fft1024_1;      //xy=548,106
AudioOutputI2S           i2s1;           //xy=553,356
AudioAnalyzeNoteFrequency notefreq1;      //xy=558,153
AudioAnalyzePeak         peak1;          //xy=573,208
AudioConnection          patchCord1(usb1, 0, fft1024_1, 0);
AudioConnection          patchCord2(usb1, 0, notefreq1, 0);
AudioConnection          patchCord3(usb1, 0, peak1, 0);
AudioConnection          patchCord4(usb1, 0, i2s1, 0);
AudioControlSGTL5000     sgtl5000_1;     //xy=826,222
// GUItool: end automatically generated code

void setup() {
    // put your setup code here, to run once:
    AudioMemory(30);

    sgtl5000_1.enable();
    sgtl5000_1.inputSelect(AUDIO_INPUT_LINEIN);
    sgtl5000_1.volume(1);

    fft1024_1.windowFunction(AudioWindowHanning1024);

    notefreq1.begin(0.15);
}

void loop() {
    // put your main code here, to run repeatedly:
```
float n;
// float nn;
// float p;
int i;

// if(notefreq1.available()){
//   nn = notefreq1.read();
//   Serial.print(nn);
//   Serial.print(" ");
// }
//
// if(peak1.available()){
//   p = peak1.read();
//   Serial.print(p);
//   Serial.print(" ");
// }
//
// Serial.println();

if(fft1024_1.available()) {
  Serial.print("FFT: ");
  for(i=1; i<40; i++){
    n = fft1024_1.read(i);
    if(n >= 0.01){
      Serial.print(n);
      Serial.print(" ");
    } else {
      Serial.print(" - ");
    }
  }
  Serial.println();
}