ThereminFreaks – Theremin-based Rhythm Game

Team 34
Esteban Looser-Rojas – looserr2
Yhoas Olivas Hernandez – olivash2
Michael Recklein – recklei2
TA: Amr Martini
ECE 445 – Spring 2019

Introduction

Objective and Background

There are many different kinds of rhythm games on the market, stretching back all the way to the 90s and early 2000s with games like DanceDanceRevolution and Pump It Up, two dance games where one steps on panels; beatmania IIDX, which is a “DJ simulator” and uses a turntable and 7 rectangular buttons; and DrumMania which uses an electric drum set to simulate playing a drum. They typically involve the player pressing a button or a touch screen and/or simulating a conventional rock instrument like a guitar or a drum set. They also typically only allow discrete notes and when there is an analog or continuous control, it’s usually not that precise, like the knobs in a game like Sound Voltex. The closest thing to what we plan on making is the singing mode in Rock Band, which only takes the user’s voice pitch into account.

We are planning on creating a theremin-based rhythm game for the PC platform. The hardware component is a PC peripheral theremin where the capacitance is controlled by moving one's hands closer or farther from the antenna-like plates on the theremin controller. We plan on using this capacitance to affect an oscillator's frequency and capture this frequency as a variable on the PC program using USB. Thus the theremin circuit will be connected to a USB controller which will connect to the PC. On the software side, we will make a simple driver for this controller and a video game written in C++ using OpenGL. The way the game works is there is a stream of notes coming at the player. The vertical axis represents the volume to correspond with the loop antenna on the theremin. The horizontal axis represents pitch to correspond with the straight antenna.

What makes this project unique is therefore the fact that it uses an unconventional instrument where there is no contact between the player and the instrument. And the pitch and volume are continuous rather than being discrete like pressing a piano. A game like Rock Band has a singing mode (actually based off a previous game called Karaoke Revolution) that is somewhat similar to what we are doing, but it only takes pitch into account and not volume [1].

Rhythm games with realistic controllers like the aforementioned DrumMania also have the advantage of teaching people how the instruments they simulate are played. In the words of Ben Weinmann and Chris Pennie from The Dillinger Escape Plan [2], “Being big fans of Konami games, we are extremely happy to be a part of the newest versions of GUITARFREAKS and DRUMMANIA. We can remember our first trip to Japan and having the pleasure of playing earlier versions of these games in the Arcade. These Games are not only fun but a great way for young people to start looking at music in a more intricate and educational way. Thanks Konami!”

High-level requirements

- The user interface and hardware be intuitive enough that a new player can learn how a theremin works. They should then be able to play the theremin with some sort of skill outside of the game.
- Playtesters should reach a consensus that the game is responsive and bug-free along with being enjoyable and engaging.
- The sound engine should work properly such that it can sound like a real theremin and provide a similar mapping between the distance of the hands from the antennas and the in-game pitch and volume.

Design

Block Diagram

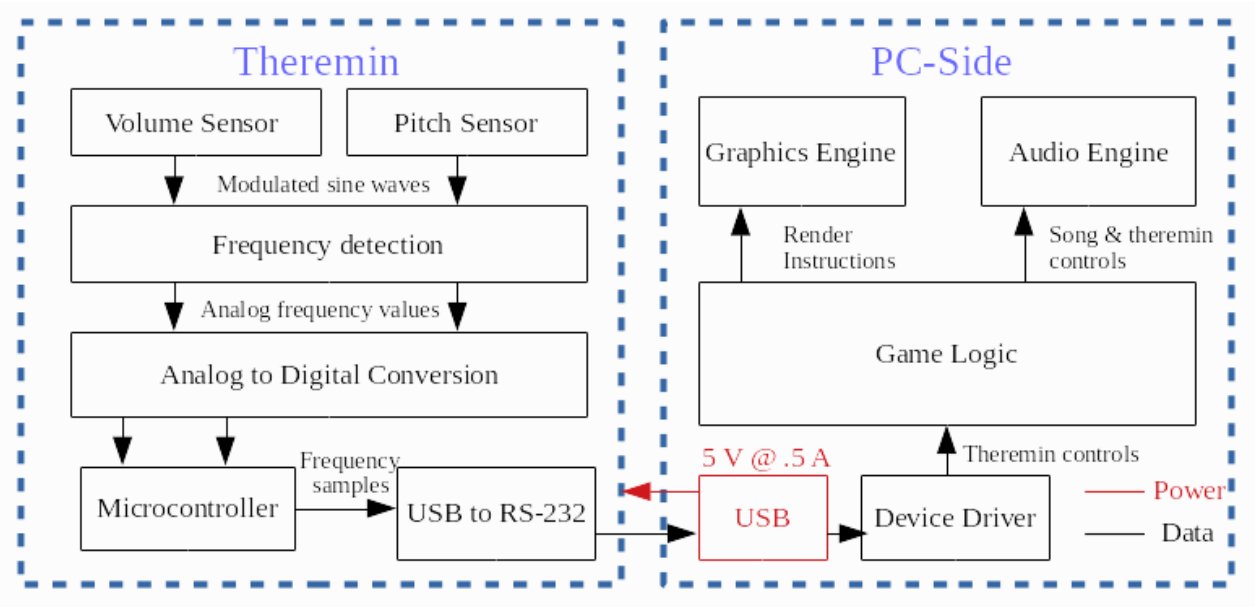


Figure 1: Block Diagram

The data flow starts from the hands to the antennae, which will create two capacitances used to change the frequencies of two Hartley oscillators. These oscillators will be mixed with another pair of reference oscillators to create values dependent on hand distance to the antennae, which will be fed to integrators and detectors to extract the frequency from the oscillating signal. The design then switches to digital as the two signals pass through ADCs and a microcontroller sends the data through a USB interface to the PC. With the amount of control we get from the software audio engine and game logic, we should then be able to finely tweak how the scaling of the theremin is configured and how it sounds while it is being played. The PC will run the rhythm game program.

The physical design, seen in figure 2, is a simple box to protect the PCB with holes for the two antennae, a USB-B port, and a reset button for the microcontroller. The size depends mostly on how big the PCB will be.

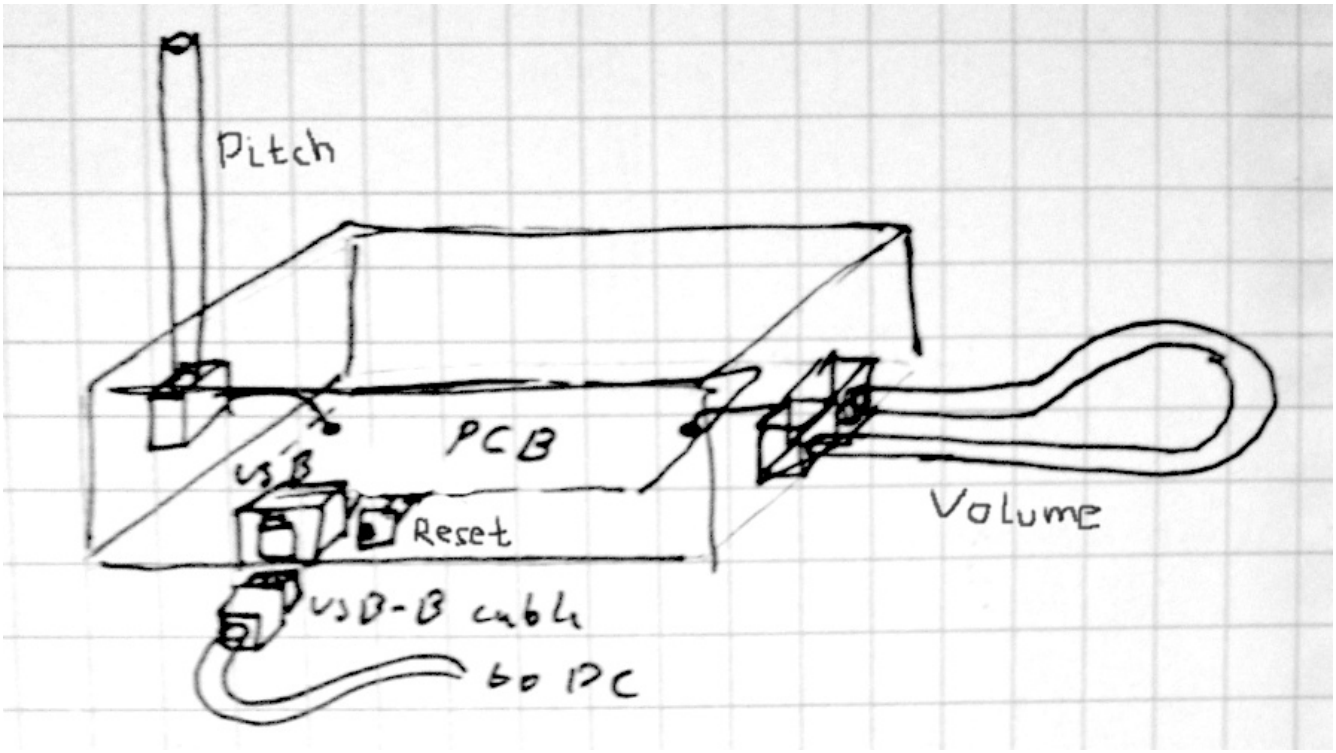


Figure 2: Theremin physical design

Hardware:

Pitch and volume sensors – The pitch and volume sensors start with two antennae that act as variable capacitors for two Hartley oscillators. Crude experimentation shows that a human touching a piece of metal creates a capacitance of around 100 pF, so the changes in capacitances are on the order of picofarads. Figure 3 shows a simulation of such Hartley oscillator, showing a frequency on the order of 4 MHz for the types of components we are using. There will be two more oscillators with fixed capacitors acting as reference oscillators. Modulating the reference oscillator with the variable-frequency oscillator using a mixer will result in a sinusoid with its frequency dependent on the distance from the hand to an antenna. Careful tuning is required to make sure that the output is very low-frequency when one has their hand away from the antennae.

Requirements	*Verification*
1. Mixer should create a stable sinusoid with frequency dependent on that of the difference between two input sinusoids, with input frequencies up to 1 to 4 MHz 2. Oscillators should be able to produce low-frequency signal below 10-20 Hz when mixed in order to create a “default” state where the reference and variable oscillators match	1. 1. Connect function generator to each mixer input, one to simulate reference oscillator and the other to simulate the variable oscillator 2. Probe output of mixer with oscilloscope 3. Turn on signal generators to MHz range, go through range of values from matched frequencies to differences in tens of kHz 4. Verify that oscilloscope shows product of the two sinusoids 2. 1. Probe output of reference oscillator with oscilloscope to have a reference of what the base frequency is. 2. Probe output of variable oscillator onto oscilloscope, make sure one can read the frequency of the signal. 3. Probe output of signal mixer with oscilloscope 4. Verify that oscilloscope shows a low frequency of 10-20 Hz at mixer output as both oscillators should be matched.

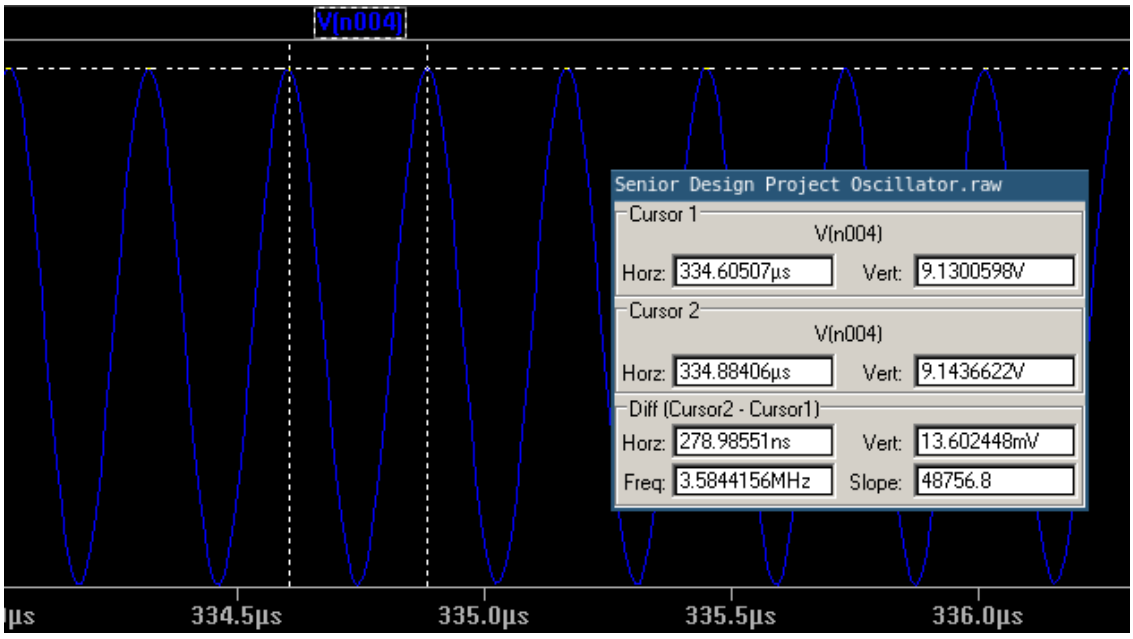


Figure 3: LTspice simulation of Hartley oscillator

Frequency detection – This module will be used to convert the output of the sensors into an analog signal that can be sampled. The output from a sensor will go into an integrator. This should give us back a sinusoid whose amplitude is proportional to its inverse frequency. After that we simply pass the signal through a peak detector to take the amplitude as a DC signal.

Requirement	***Verification***
1. Detector must have high enough bandwidth to respond to sensor output, up to 30 kHz	1. 1. Probe output of detector with respect to ground using voltmeter 2. Connect function generator to input of detector with respect to ground 3. Turn on function generator, send range of sinusoids from 20 Hz to 30 kHz 4. Verify on voltmeter that inverse dependency on frequency is achieved throughout frequency range

Analog to digital converter – This module is used to digitize the signal from the frequency detector. Because we don't expect the signal to change very frequently, being a musical instrument played by a human, we selected the 20-pin MAX1166 16-bit ADC due to it being small and low power and meeting our sampling frequency needs. In order to avoid having 8 more pins it alternates between sending the LSB and MSB, with a pin on the chip indicating which one is being sent.

Requirement	***Verification***
1. ADCs can send at least 1000 samples per second	1. 1. Probe 8-pin parallel interface and EOC pin, should give digital outputs 2. Send range of values appropriate for simulating frequency detection outputs made through experimentation 3. Check to see if data-out pins and EOC pin change accordingly

Microcontroller – The microcontroller must read data from the analog to digital converters and send it to the PC. A separate module will handle USB communication; all we need to do is to program the microcontroller to send the theremin data through RS-232 UART. We are using a PIC16C65 since we already have some and are familiar with using this kind of microcontroller.

Requirement	***Verification***
1. Able to send samples from ADC to RS-232 interface	1. 1. Create program that will read data in and send it through RS-232 2. Check to see if data is being sent to PC with USB cable plugged in

USB to RS-232 converter – In order to simplify the software design, a virtual RS-232 interface will be used. One of us has already made RS-232 interfacing software for the microcontroller we are using, so there shouldn't be much hassle communicating with the theremin this way.

Requirements	***Verification***

1. Able to interface with PC through USB
2. Able to interface with microcontroller through RS-232 UART
	1.
	  1. Connect USB cable from converter to PC
	  2. Verify that a new COM port device appears on PC
	2.
	  1. Program microcontroller with simple program that sends a string to the computer
	  2. Connect USB cable from converter to PC
	  3. Use RS-232 terminal program to read text input from virtual COM port
	  4. Verify that string appears in terminal

Schematic:

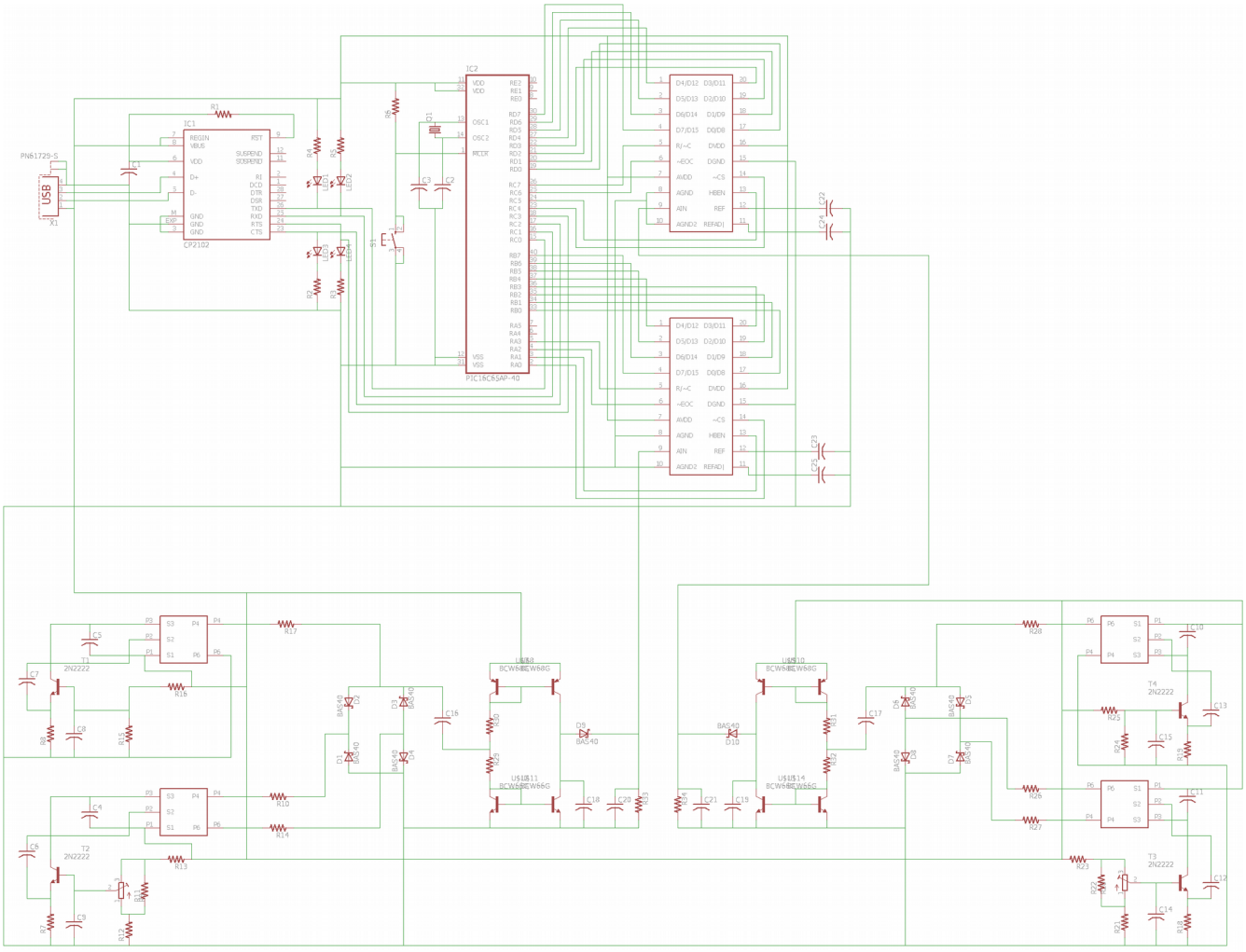


Figure 4: Theremin Schematic

Software:

Graphics engine and game logic – The video game will generate a “map” that the player has to try to imitate with the theremin and will judge the player based off of how close their pitch and volume is to said note map. We are using OpenGL to render the game menus and the game itself, using two dimensions on a plane for the pitch and volume and the z direction as time.

Requirement	***Verification***
1. The game should run at least at a constant 60 fps to ensure a smooth gaming experience	1. There will be a frames per second counter on the screen. We just have to make sure it doesn’t dip below 60.

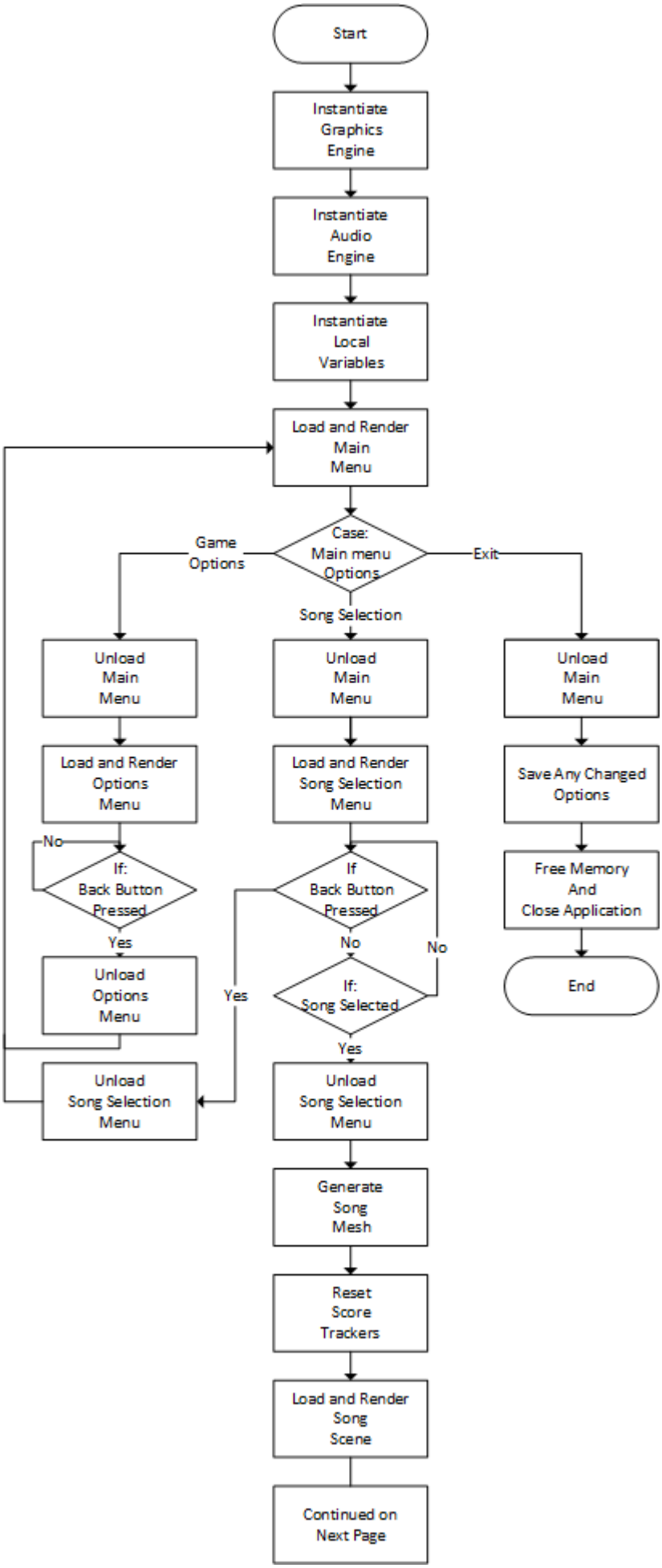
Audio engine – Takes in pitch and volume data from the theremin, along with background music files and plays the background music along with a synthesized theremin track manipulated by the player.

Requirements	***Verification***
1. Should be able to play a song synchronized with the video game so that there is no noticeable lag between what is on the screen and what is being heard 2. Should be able to generate theremin-like sound for a range of at least 4 octaves	1. 1. Have a decently-sized sample of people play the game (probably 4 or more) 2. If there is a consistent shift in timing towards late or early timing, the game is probably not synced correctly 2. A very subjective requirement, it can nevertheless be tested by going through the full 4 octave range and comparing it to recordings of real theremins

Device Driver – This piece of software will communicate with the USB COM port and will provide the game logic with a stream of 32-bit samples comprising of a 16-bit pitch sample and a 16-bit volume sample. Thus the game logic and audio engine should have easy access to the data arriving from the theremin, as a C struct consisting of two 16-bit integers.

Requirement	***Verification***
1. Should provide a steady stream of 32-bit samples from the theremin at a high enough bandwidth to minimize delay below 50 ms in order to prevent input lag.	1. 1. Set up microcontroller to send varying output through USB to PC 2. Print out stream of data coming from device driver 3. Check to see if time difference between microcontroller generating output and device driver output exceeds 50 ms

Software Flowchart:


Start
Instantiate Graphics Engine
Instantiate Audio Engine
Instantiate Local Variables
Load and Render Main Menu
Case: Main menu Options
Game Options
Song Selection
Exit
Unload Main Menu
Load and Render Options Menu
No
If: Back Button Pressed
Yes
Unload Options Menu
Unload Song Selection Menu
Unload Main Menu
Load and Render Song Selection Menu
If Back Button Pressed
No
Yes
If: Song Selected
No
Yes
Unload Song Selection Menu
Generate Song Mesh
Reset Score Trackers
Load and Render Song Scene
Continued on Next Page
Unload Main Menu
Save Any Changed Options
Free Memory And Close Application
End

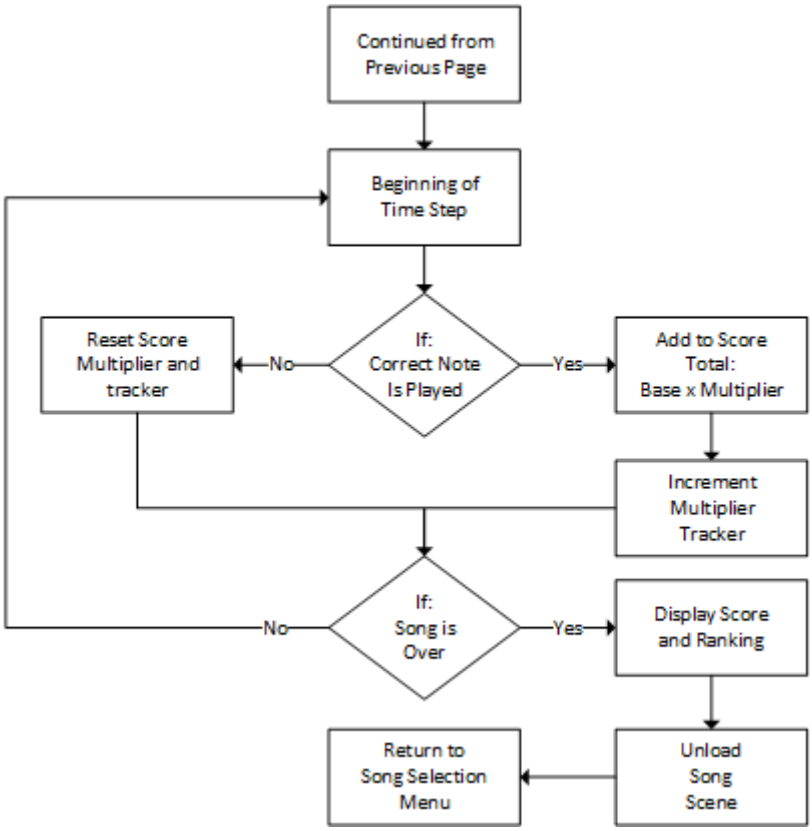


Figure 5: Flowchart of game logic

Tolerance Analysis

The oscillators pose the greatest risk for the success of our project. We have to tune them precisely so that the user's movements are tracked accurately and faithfully to how a real theremin operates. If we are careless with how we approach this part of the theremin circuit, we could have a very inaccurate frequency. For example, the oscillators work at megahertz frequencies, so if they are off by a kilohertz, then we'll have a tone of a kilohertz out of nowhere, throwing everything off.

Running LTspice simulations for our Hartley oscillator circuits, it is apparent that the oscillators we are using are going to be quite sensitive. A picofarad change in capacitance in the tuned circuit leads to about a 17 kHz change in frequency. We are unsure whether or not this will be a problem. On one hand, the change in capacitance between a hand being close to a metal rod and being farther away by a centimeter or so is going to be tiny. But we have no way to verify exactly by how much the capacitance changes until we actually build the theremin.

To try to remedy this, we have been looking at the different signal transformers available on the market, and we have come up with a list of three that we are going to test. They differ in turn ratios, having 1:2, 1:3, and 1:4, such that they each have different inductances. Since the resonance frequency of the tuned circuit depends on the inverse of the square root of the product of inductance and capacitance, we are using different transformers to give ourselves ample room to tune and tweak the oscillators so that they work reliably.

Costs and Schedule

Cost Analysis

The average salary for ECE graduates with a BS is around $40 an hour, and we have around eight weeks or so to work on the project. Assuming we work for ten hours a week with three people in the team, that amounts to about $24,000 in labor costs.

Costs for the parts necessary to build the theremin are included in the following table:

Parts	***Costs***
PIC16C65A microcontroller	$9.00
MAX1166 16-bit analog-to-digital converter x2	$23.90
UJ2-BH-1-TH USB-B connector	$0.65
Silicon Labs CP2102 USB to RS-232 interface	$1.33
2N222 transistors, any manufacturer x4	$1.00
Vishay Semiconductors 1N4148 diode x2	$0.40
BCW68GVL PNP transistor x4	$0.68
BCW66GVL NPN transistor x4	$0.72
Gilbert cell transistors x12	$4.00
TL3305AF260QG SMD push button	$0.23
Indicator LEDs	$0.80
WBC2-1TLB RF ferrite core transformer x4	$20.00
Total	***$62.71***

Total development costs are therefore $24,063

Schedule

Week	***Yhoas***	***Esteban***	***Michael***
2/25	Additive wave synthesis for theremin sound	Finish circuit design taking all options into account	Begin working on graphics engine to draw 3D line in space
3/4	Subtractive wave synthesis and feedback distortion; compare different methods	LTspice sim done; all component values assigned; PCB designs complete	Graphics engine complete
3/11	Background music program	Mixer and detector breadboard prototyping	Map generator using songs as inputs
3/18	Theremin setup – theremin antennae and oscillator testing	Oscillator prototyping; order parts; PCB revision complete	Mesh generator, compose first song
3/25	More testing and tweaking theremin to stabilize oscillator output	Finish MCU software for RS-232 interface	Finish mesh generator, compose second song
4/1	Theremin assembly – case and fix antennas on	Theremin assembly – soldering parts onto PCB	Gameplay mechanics – timing judgements, scoring

4/8	Tweak music program and synthesis program with completed theremin	Finish theremin – oscillators should be working fine by now	Main menu and options menu
4/15	Last-minute debugging for theremin before mock demo	More time in case oscillators are wonky before mock demo	Song select menu, compose third song
4/22	Compile together mock presentation	Apply any feedback from mock demo	Last-minute changes to video game
4/29	Work on final paper and final presentation	Final paper	Final presentation

Ethics and Safety

The ACM code of ethics talks about reducing harm, including physical and mental injury and unjustified harm to the environment [3]. We won't be working with any lethal voltages, so the risk of harming someone with our theremin is very low, comparable to the risk of someone having to plug in any household appliance. And our theremin doesn't use radio in order to communicate, so the risks associated with interference are negligible if we shield the components well enough. Our video game is simply a free musical instrument simulator without any micro-transactions, and the music we are composing for it won't have any explicit lyrics. So anyone of any age should be able to play it.

In another way, we will have to be responsible for our own safety when we use tools to solder and assemble the theremin. Thankfully, we have all taken lab safety training and we have experience in soldering electrical circuits. Much of the work involved in satisfying the design requirements for the hardware is soldering chips onto our PCB and probing it with lab equipment.

The IEEE and ACM codes of ethics also mention honesty and trustworthiness, along with abiding by copyright law [3][4]. We will disclose all the components we are using for the theremin and will keep track of all our source code with appropriate licenses. That includes code used to control the theremin's microcontroller and the code for the video game. We will also create our own circuit schematics and will only use software and libraries pursuant to their disclosed licenses. For example, OpenGL uses a free license that allows us to use it for our project. As for the music used in the video game, we will avoid infringing on copyright law by using either our own arrangements of public domain music or music we have created ourselves.

The ACM code of ethics discusses user privacy and confidentiality [3]. We are not designing the theremin or video game to communicate any information through the Internet or to store any kind of user information. So users' privacy rights are not at risk either.

References

[1] “Karaoke Revolution,” *Wikipedia*, May 17 2018. [Online]. Available: https://en.wikipedia.org/wiki/Karaoke_Revolution. [Accessed Feb 7, 2019].

[2] “Panasonic Youth,” *RemyWiki.com*, Dec. 27 2018. [Online]. Available: https://remywiki.com/Panasonic_Youth#Song_Production_Information. [Accessed Feb 1, 2019].

[3] Association for Computing Machinery, “ACM Code of Ethics and Professional Conduct,” *Association for Computing Machinery.* [Online]. Available: https://www.acm.org/code-of-ethics. [Accessed: Feb. 1 2019].

[4] IEEE, “IEEE Code of Ethics,” *IEEE*. [Online]. Available: https://www.ieee.org/about/corporate/governance/p7-8.html [Accessed: Feb. 1, 2019].