MIDI Zeusaphone

Design Document

Team Number: SDMAY19-11
Client: Dr. Joseph Zambreno
Advisor: Craig Rupp

Team:
Jacob Feddersen - Communications Specialist
William Brandt - Pulse Width Modulation Expert
Luke Heilman - Technical Architect
Gregory Harmon - Tesla Coil Construction Expert
Leo Freier - Interrupter and Microcontroller Lead
Gunnar Andrews - Webmaster

Team Email: sdmay19-11@iastate.edu
Team Website: http://sdmay19-11.sd.ece.iastate.edu

Revised: 12-2-2018 - Version 2.0

Table of Contents

List of Figures

List of Tables

List of Symbols

List of Definitions

CprE: Computer engineering, generally referring to the major or a Computer Engineering student.

DRSSTC: Double Resonant Solid State Tesla Coil - a tesla coil design which can be modulated, producing audio.

EcpE: Electrical and Computer Engineering. Usually refers to the EcpE Department at Iowa State University, which includes Electrical, Computer, and Software Engineering.

EE: Electrical engineering, generally referring to the major or an Electrical Engineering student.

MIDI: Musical Instrument Digital Interface. A technical standard for playing sounds through a digital interface. MIDI can also refer to the file type that computers use to play sounds based on the MIDI standard.

PPE: Personal protective equipment.

PWM: Pulse width modulation.In this case, a process for outputting analog signals on a microcontroller pin.

Tesla Coil: A resonating transformer circuit that produces very high voltages, generating arcs into the air.

WAP: WiFi Access Point

Zeusaphone: A special Tesla coil that releases voltages at specific frequencies, creating sound like a musical instrument

1 Introduction

1.1 Acknowledgement

The MIDI Zeusaphone team would like to extends thanks to our client Dr. Joseph Zambreno for providing the project, as well as the full financial support and other technical assistance during the project. The team would also like to thank our advisor Craig Rupp for being a reliable expert on the subject matter, being a professional mentor for the team, and always being available for us.

1.2 Problem Statement

When prospective students are given a tour through Iowa State, they are shown the accomplishments and senior design projects of past undergrad students. The Electrical and Computer Engineering Department currently has two inoperable arcade cabinets that were constructed by previous electrical and computer engineers. In order to continue attracting students to ECprE, the department needs a new showpiece to demonstrate what prospective students could be capable of if they choose to attend Iowa State.

Our solution to this problem is to construct a Tesla Coil that plays music, also called a Zeusaphone. The Zeusaphone will be able to play preset songs as well as have the ability to be played with a piano keyboard so that prospective students are engaged with the demonstration. The project includes specialized circuits and a microcontroller. It will appeal to prospective students with an interest in embedded systems or circuit design. Because it will be shown on tours, an operating manual will be written to ensure the operator is using the Zeusaphone properly. A safety manual, proper signage, and proper personal protective equipment (PPE) will also be provided so that no injuries occur when the device is in operation.

The team is comprised of four Computer Engineering students and two Electrical engineering students. It was determined to be a good fit for the project as the two EE students can tackle the tesla coil designs with assistance from the CprE students in building and operating the coil. The CprE knowledge can then be applied to converting MIDI messages, outputting messages to the circuit, creating a user interface, and setting up the interfacing keyboard and web client. The project appeals to the team as many have a musical background and all students find the tesla coil to be an intriguing subject on its own.

1.3 Operating Environment

The MIDI Zeusaphone will always be demonstrated indoors. It will be stored in Coover Hall and will be operated in the same place. There is no threat of moisture since it will not go outside. There may be a problem with dust build-up if it is stored for an extended period of time, but this can easily be handled by quickly dusting the project off or blowing the dust off.

1.4 Intended Users and Intended Uses

As the goal of the MIDI Zeusaphone is to be a showcase item for the EcpE Department, the operator of the Zeusaphone will always be a faculty member of the EcpE Department. However, the operator may not always be someone with previous knowledge or operation experience with the device. Therefore the MIDI Zeusaphone should be designed with simplicity and intuitive operation in mind.

The MIDI Zeusaphone will be used in demo scenarios in front of an audience. This audience could be a small private viewing or a large demo in front of a lecture hall.

1.5 Assumptions and Limitations

Assumptions:

On Usage

- The operator will be able to play a MIDI keyboard to produce sounds
- The operator can play pre-loaded MIDI songs to play through the web client.
- The operator can load MIDI songs through the web client to be played later.

On Safety

- The primary use of the Zeusaphone will be as a showcase item.
- The operator will be fully aware of the safety considerations and proper use of the Zeusaphone.
- During operation, all safety standards will be followed by the operator and the audience.
- When not being shown, the operator assumes responsibility as laid out by the provided safety standards.

On Reliability

- The system can be safely stored in any room safe enough to store high voltage circuits.
- The full project will be able to be reliably moved to and from storage with minimal assembly and disassembly
- Improper input will not result in a dangerous situation.
- The system can be safely disabled and shut down immediately at any time

Limitations:

- The end product will be no larger than 2 ft tall with a 1 x 1 square foot area
- It must be able to be run off of a wall outlet. (120V 60Hz)
- Can only play two different tones at once
- Operators must be associated with the EcpE Department.
- The Tesla coil will only be able to be activated using the project interfaces.

1.6 Expected End Product and Other Deliverables

- MIDI Zeusaphone (May 2019)
    - This will be the final product of our project. This will include a Tesla coil or coils that will play frequencies to make music while electricity arcs out of them. This will all be made by us and programmed by us. This device will be portable and easy to work so it can be used by a large number of people.
- Operating Manual (May 2019)
    - This will be a very detailed guide for working the zeusaphone. It will include all the steps to turn on the zeusaphone and make it play through all of the different interfaces. This manual will also include extensive safety details, so that whoever handles the project will know exactly what steps to take in order to ensure that the zeusaphone is operated safely. Finally the operating manual will also explain how to play the zeusaphone through all of the available interfaces (keyboard, MIDI, bluetooth, etc).
- Keyboard (May 2019)
    - Along with the zeusaphone a keyboard will be provided. There will be instructions inside the operating manual on how to connect the keyboard to the zeusaphone. This keyboard will be used to make music through the Zeusaphone.

2. Specifications and Analysis

2.1 Proposed Design

We propose a zeusaphone that is controlled by a microcontroller. The microcontroller processes MIDI events from a variety of inputs and controls the tesla coil accordingly. A very general overview of the layout can be seen below in figure (Fig.1).

We chose to use a Raspberry Pi for the microcontroller. The Raspberry Pi is a small, credit card sized computer that is capable of running a full Linux operating system. We chose to use the Raspberry Pi for a number of reasons:

- The Raspberry Pi has a number of general purpose I/O pins, including two with hardware-timed pulse-width modulation, which we use for outputting the audio waveforms.

- The Raspberry Pi is capable of acting as a wireless access point, and it can host its own web server. We use this to host an interface for controlling the tesla coil system.
- We have several Raspberry Pi's immediately available for testing, and several of our team members have experience setting them up and using them.

Music can be played on the tesla coil from two different sources. First, MIDI files can be stored on the Raspberry Pi and played back. These MIDI files would be loaded, managed, and played using the web interface. Second, a MIDI keyboard can be plugged into the Raspberry Pi and used to create live input. The web interface would be used to specify that the keyboard should be used for input.

The Raspberry Pi is connected to the tesla coil with a fiber optic cable, to avoid interference from the operation of the tesla coil. The output from this fiber line is used to modulate the tesla coil itself. When the line is active, the tesla coil is enabled and sparks are created. When the line is not active, the tesla coil is off. By modulating this output, music can be played through the sparks on the tesla coil.

In order for a user to control the Raspberry Pi itself, the Pi hosts a simple web interface accessible through http using any web browser. This web interface will have the following features:

- Upload a new MIDI file to the Raspberry Pi, to enable it to be played on the tesla coil
- Delete a MIDI file from the Raspberry Pi
- Play a MIDI file currently stored on the Raspberry Pi
- Enable/Disable the MIDI keyboard input functionality

To keep this control limited to the correct people, the Pi will transmit its own WiFi Access Point (WAP) protected by WPA2. The web page will only be available on this WAP, which will not be connected to the internet itself, thus providing a layer of security.

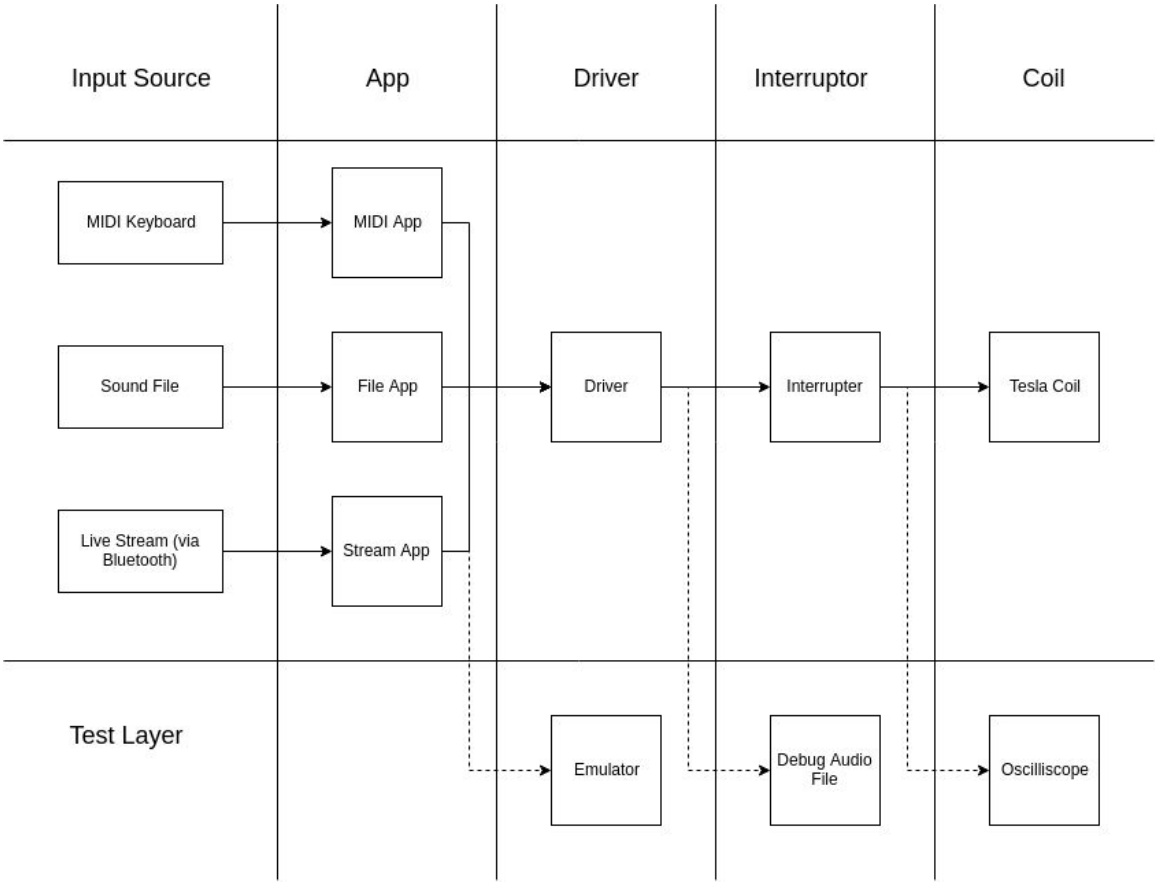

Figure 1: Overview of the Project Layout

The tesla coil itself is a dual resonant solid state tesla coil. The tesla coil uses transistors to switch the voltage across a primary capacitor and inductor on and off. This resonates with the secondary inductor which is recognized as the tall tower of the coil. The resonation steps the voltage up drastically and that causes the air to breakdown around the top of the coil. When the air breaks down, that is when the signature sparks appears. For the high frequency switching to occur, the transistors are driven by a driver circuit (Fig.2). This driver circuit (not to be confused with the driver layer mentioned beforehand) will take in the output from the interruptor layer seen in Figure 1 and send a matching signal to the gates of said transistors at the appropriate voltage in the bridge circuit (Fig.3). Within this driver circuit provided by oneTesla is another form of input that comes from a current transformer that are connected to the primary circuit with the primary capacitor and inductor. It is for feedback that allows the input from the interruptor to be synced with the already switching circuit to prevent any out of phase waves. Size is also an important factor that is considered as the device is designed to be moved around to certain events where it will be showcased. The size is approximately two feet tall to prevent disassembly when it is put into storage or transported.

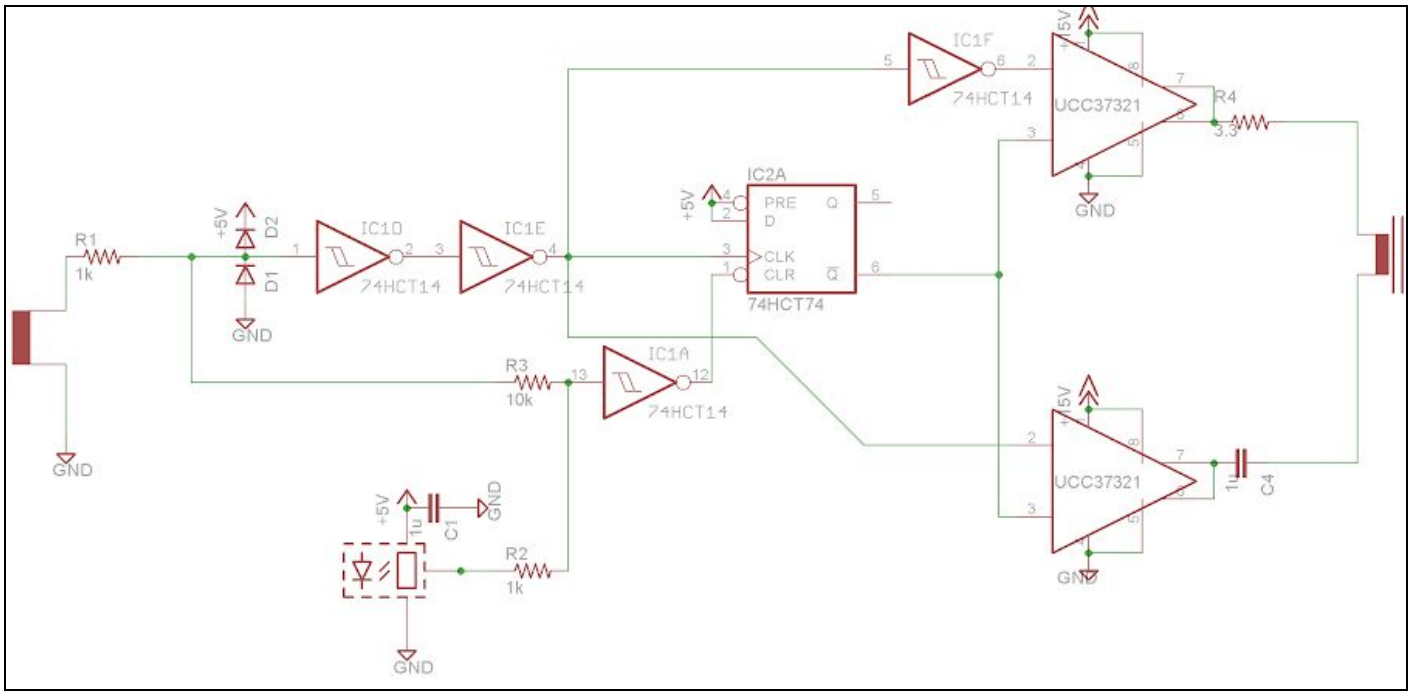

Figure 2: Tesla Coil Driver Circuit

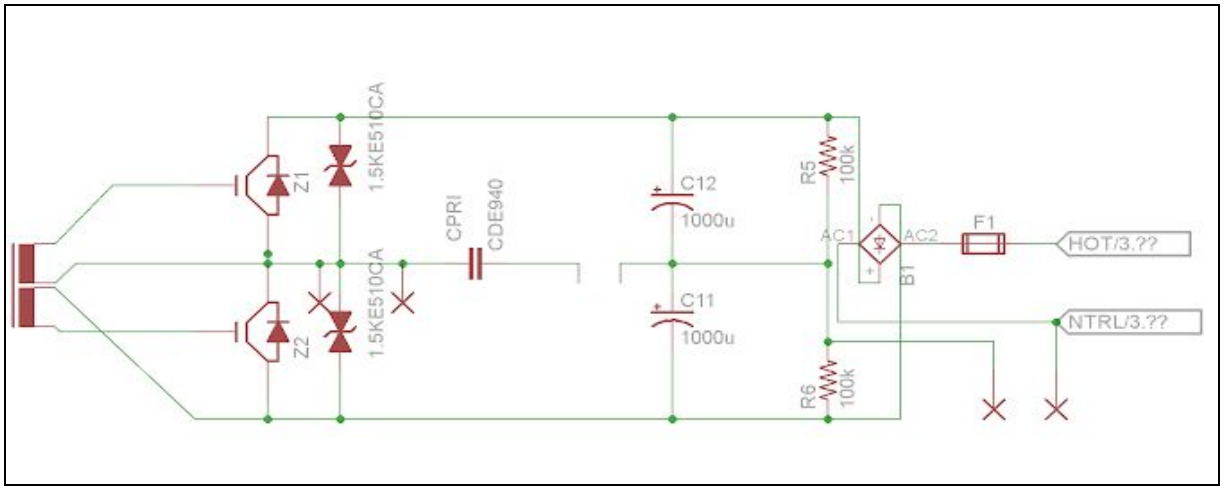

Figure 3: Tesla Coil Bridge Circuit

2.2 Design Analysis

The entire design can be broken down into two main components: the user interaction side, which contains the Raspberry Pi, the MIDI keyboard, and the interrupter circuit; and the tesla coil, which actually produces the output.

The software layers of the project are being actively developed. Since the layers are modularized, each layer can be built and tested without the other layers. The hardware layers are being researched and a prototype tesla coil will soon be built.

2.2.1 The Application Software

The app layer is where music is input into the system. There are two different applications, one for reading MIDI files from storage and one for reading live input from a MIDI keyboard. The app is responsible for reading and parsing the input data, converting it into a two-channel stream of notes, and sending it to the driver layer through a socket.

2.2.1.1 MIDI File App

The MIDI file reader opens and reads MIDI files from the filesystem using the midifile C++ library written by Craig Sapp. This library reads the MIDI file and parses it into a C++ object. All of the events in the MIDI file are stored in an array, in order. The MIDI file app reconstructs the timing of events using a wait loop, and reads the notes specified in the file. If more than two notes are played at once, some of the notes may be discarded; the MIDI file app will only play two channels at a time.

2.2.1.2 MIDI Keyboard App

The MIDI keyboard receiver program listens for MIDI messages from devices attached to the Raspberry Pi. These messages are initially processed by the Advanced Linux Sound Architecture library (ALSA) in the kernel. The program then makes use of the RtMidi library by Gary Scavone to parse these messages and setup a callback function to handle them. The MIDI keyboard app ignores all MIDI events except note on and note off. It only outputs two channels at a time, but it overwrites the oldest note that is still being held if a third note is played.

2.2.2 The Driver Software

The driver is the program on the Raspberry Pi responsible for interfacing with the tesla coil. It runs in the background, and acts as a server for the MIDI file app and MIDI keyboard app to connect to. It listens to the note messages input to the socket, and it generates output waveforms on the Raspberry Pi GPIO pins accordingly.

After experimenting with timing loops, threading, and interrupts, we finally decided to use the Raspberry Pi's hardware pulse width modulation pins to output the waveform. None of the other options provided the stability and timing accuracy that we needed. The hardware PWM pins have their own timing chip and counter that are seperate from the system clock of the Raspberry Pi. These counters will continue to output a wave at exactly the frequency set no matter what the workload of the processor is, and no kernel process can preempt or delay them. The duty cycle of each frequency waveform will be refined during testing, but we anticipate using a duty cycle of approximately 5% per channel. This will keep the total utilization of the tesla coil below 10%, which will help prevent the tesla coil from being overloaded or overheated.

Using the hardware PWM pins, each audio channel must be output on a seperate pin. The outputs cannot be combined in software. Therefore, the outputs are combined into a single output and sent to the tesla coil by the interrupter circuit.

2.2.3 The Interrupter Circuit

The interrupter circuit combines the two hardware PWM outputs by logically 'OR'ing them together. The output is then output over a fiber optic cable to be sent to the tesla coil. We chose to use a fiber optic connection for two reasons: optical transmission makes the signal impervious to electromagnetic interference from the tesla coil itself, and this also makes the interrupter circuit attached the the Raspberry Pi completely electrically isolated from the tesla coil circuit. The circuit diagram for the interrupter is shown below (Fig. 4):

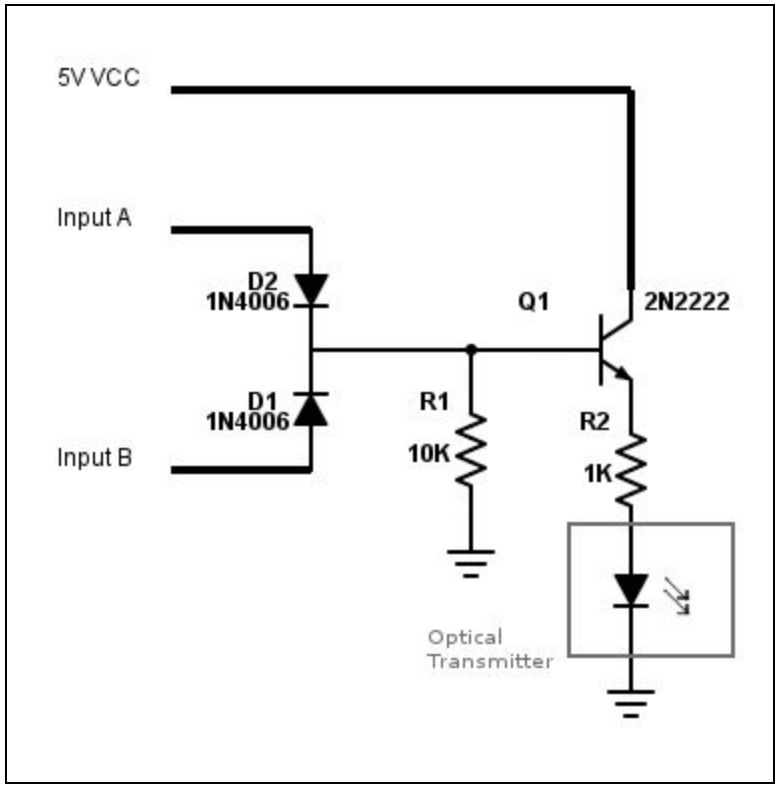

Figure 4: Interrupter Circuit

2.2.4 The Tesla Coil

The tesla coil is the part of the circuit where everything else done comes to fruition. The tesla coil starts by taking the input from the interruptor layer through a fiber optic connection. From that point, it will use a signal from a current transformer on the primary coil that will be fed through two inverting schmitt triggers to a flip-flop. This synchronizes the input from the interrupt layer with the current oscillations of the primary coil to ensure that no off-phase signals are being sent to the coil. The synced signal drives the

transistors in the bridge circuit which will trigger the oscillations in the primary coil. At this point the primary coil will resonate with the secondary coil and step up the voltage by a factor of 330. The high voltage will break down the air around the top of the coil and create the sparks. The frequency of the sparks turning on and off is controlled by the interrupt layer. The rapid expansion and contractions of the air caused by the sparks create a sound wave, that when it is at a frequency that humans can hear, creates audible sound.

3 Testing and Implementation

3.1 Interface Specifications

3.1.1 Interface between MIDI Keyboard and MIDI Keyboard App

The keyboard will be interfaced with the Raspberry Pi and the MIDI Keyboard application running on the Raspberry Pi via a USB cable. They will communicate using the USB MIDI protocol. However, we do not need to implement anything with regard to this protocol because it is all handled automatically in the Linux kernel ALSA libraries. The RtMidi library performs the system interactions to read the MIDI messages, and we can access them in turn using the RtMidi library API.

3.1.2 Interface between Software Application Layer and Driver Layer

The following protocol is used to interface between the App software and the Driver software. Messages are sent via a local UNIX TCP socket with the following format:

```
C<channel number>F<frequency>;
```

Each message must have a channel number and the frequency. The channel number is used to keep track of which notes are playing; only two notes can play at a time, so if a new frequency is sent on the same channel the old frequency will be overwritten. The frequency is a floating point value, specifying the sound frequency to be played on that channel. To turn a channel off, a frequency of “0” is sent. Each message must end with a semicolon.

3.1.3 Interface between Interrupter Circuit and Tesla Coil

The interrupter circuit and the actual tesla coil circuit will interface over a fiber optic communication link. This will allow us to keep the keyboard and Raspberry Pi microcontroller a safe distance away from the tesla coil while also preventing electromagnetic interference from the tesla coil from distorting the control signals.

The communication sent over the fiber optic channel will be a simple on/off state. When the channel is active, the tesla coil will be on, or producing sparks. When the channel is inactive, the tesla coil will be off. Turning the tesla coil on and off at a certain frequency will produce sound of that frequency.

3.1.4 Web Interface for User Interaction

The web interface used to remotely control the tesla coil will be hosted on a web server on the Raspberry Pi. This web server will only be accessible from a WiFi network also hosted on the Raspberry Pi. The network to access the web interface will be secured with WPA2, preventing unauthorized users from connecting to the interface and controlling the tesla coil.

The web interface would have only a few simple functions. First, it would allow the user to select between live keyboard input and MIDI file input. If MIDI file input is selected, the user will have three options: upload new MIDI files, play a saved MIDI file, or delete a MIDI file. The web interface will be designed to work with all modern browsers (Safari, Chrome, Firefox, and Edge), and it will also be compatible with both mobile and desktop browsers.

3.2 Hardware and software

To test if the App Layer works correctly and accurately records and transmit the MIDI data, a Driver Emulator software was written. This software has the same interface as the actual driver, receiving data on turning notes on/off for specific channels via a socket connection. However, instead of interfacing with a GPIO pin and outputting voltage, the emulator creates a wav sound file from the data. This file can then be listened to in order to determine if the information was processed correctly. This software has been used to test both the keyboard input and MIDI file input programs (in the App Layer).

An oscilloscope can be used to test the output voltages from the driver. This will ensure that the input into the interrupter circuit behaves as expected. In a similar way, the interrupter output voltage can be monitored on an oscilloscope. These voltages will be tested rigorously before the components are connected to the coil, to make sure the coil doesn't receive incorrect input and cause dangerous situations. An oscilloscope can also be utilized to test the driver circuit on the tesla coil before it drives the transistors to ensure that the output is at the proper voltages and the waveform is correct to ensure that the expensive transistors aren't being operated outside of their rating.

A waveform generator and a power supply is used to provide controlled inputs into the circuits. This enables testing of ideal and extreme cases of voltages and waveforms so that we can test the limits of the circuit without creating an uncontrolled and unsafe testing environment.

3.3 Functional Testing

Both forms of input (keyboard and MIDI files) need to be verified. This can be tested in stages, since the layers have been modularized.

3.3.1 Application Layer Testing

The reception of the input can be tested with the driver emulator, as mentioned earlier. This ensures that the input is interpreted correctly by the application. If the resulting

audio file matches the song that was inputted into the system, then it is working correctly.

There will also be a suite of unit tests written for the application layer covering the functionality of the code. If any of the unit tests fail, then there will be some code that needs to be revised.

3.3.2 Driver Layer Testing

The output voltages from the driver and interrupter can then be tested with an oscilloscope. A single output frequency can be sent through the driver layer, and the frequency of the output can be compared to the intended frequency. The duty cycle can also be tested with the oscilloscope - the total should be at most 10%. If only one channel is playing, this could be the full 10%. Otherwise, if two channels are playing, the output of each channel should have a duty cycle of 5%. The oscilloscope also shows how stable the output wave is. No jitters should be seen.

Again, unit tests will be written for this software to test its functionality. Tests will be performed for both .mid file input and keyboard input, as the two interfaces are very different. The outputting waveform will be tested for the above parameters such as wave accuracy, duty cycle, and wave stability. Failing to pass these tests means there is an issue to be resolved, as the functionality should be perfect in these regards. The driver needs to work in all scenarios as to not damage the coil with incorrect signals.

3.3.3 Tesla Coil Testing

Testing of the tesla coil will be done by breaking up the device into two modules and conducting a visual and electrical test on each one. The two modules are the driver and bridge circuit, and the secondary coil and toroid.

The visual inspection on the circuitry will have an emphasis of looking for burnt out components, cold solder joints, or any missing/incorrect components. To test the circuit electrically, an input will be given from the interrupt layer and we will observe the output of the bridge circuit using an oscilloscope. This output will be compared to the output of a commercially available driver and bridge circuit. This test will prevent the destruction of components due to an incorrect input signal.

For the secondary coil and toroid, a visual inspection will look for imperfections in the windings that may cause arcing to occur in between the windings. We will also be looking for any imperfections in the toroid that may prevent the sparks from emitting off of it. The electrical test will involve passing a lower voltage AC signal through the connected toroid and secondary coil and using a floating oscilloscope probe, we can measure the frequency at which the coil oscillates and ensure that it will match our design specifications.

3.4 Non-Functional Testing

3.4.1 Ease of Use

The final project needs to be simple to setup and use. A manual detailing how to do this will be produced for future users. When the project is at a more complete stage, we can

test the usability of the manual by letting people attempt to use it with only the manual for instruction. Of course, safety is also an issue here. We can be present during this testing to prevent a user from doing something that will cause harm.

3.4.2 Safety Testing

We need to be sure that the operation of the tesla coil does not endanger the operator and the audience. For this we need to ensure that the arcs are within safe standards and that nearby persons are not overexposed to electromagnetic waves. In order to do this we will be able to observe the length of the arcs in controlled settings to make sure that they are not longer than 40 cm. We will also test the strength of the electric and magnetic field in a controlled environment, so that if they exceed the maximum permissible exposure set by IEEE Std. C95.1-2005 (614 V/m, 16.3/freq. A/m), we will be able to implement proper shielding.

3.4.3 Reliability Testing

Testing the reliability of software should be straightforward. There are numerous test cases, but they are limited in scope as interface can only come from the keyboard or the web client. As long as the software passes unit tests and outputs what is expected, reliability will be ensured as there should be not outside interference outside of the normal interfaces. Web client reliability will be the most challenging as it is the most volatile. The client should be tested from various computers and operating systems to ensure a reliable connection and usability, which is an important aspect of the web client.

We will test to ensure that the coil can create output for the expected range of frequencies. In addition we will also check external conditions as they are available, such as testing in environments with different temperatures to ensure that hotter temperatures do not have a major effect on performance. When construction and initial testing is complete, tests will be performed around every week in an attempt to ensure reliability. Setting up and tearing down the coil numerous times should give the team a good idea of the system's reliability.

3.5 Modeling and Simulation

The project was made with a modular design, allowing for different layers to be tested by simulators. The software and microcontroller output was tested with an oscilloscope, to ensure that the waveform was smooth and at the intended frequency. By plugging in an audio jack and a pair of headphones into the circuit, the output could also be listened to. The sound should resemble the MIDI messages that were input.

There are a few different tesla coil simulator programs available for use. These programs can give an idea of potential problems with the coil design and work as some preliminary testing. The coil circuit can easily break if handled incorrectly, so these kind of tests are important prior to a live test of the coil. Simulations will mainly be used as a sanity and math check, but are important to ensure correctness.

3.6 Process

The general design of the project (especially in regards to the software) was planned with testing in mind - each layer is modular and can be tested without the other layers being present. Each program running on the raspberry pi will be tested on their own. When they are functioning as expected, we will connect them to ensure that they communicate correctly with each other. Before connecting the output of the GPIO pins on the microcontroller to any circuits, we will check it with an oscilloscope.

Along with the software, the hardware of the zeusaphone is designed to be modular in nature. Before each portion is wired to another, the outputs of each module will be monitored with a controlled input. The outputs of each module will be checked using an oscilloscope and the inputs can be created using a waveform function generator.

3.7 Results

Minimal testing on the software has been conducted so far. These have been to determine if the basic functionality of the programs work. However, none of them have undergone rigorous unit testing yet. As we progress further into the project, this will gain more attention.

As of now, we have programs that can grab MIDI messages from either a file or live keyboard input and parse the note on/off information. These messages can be sent to a driver emulator which outputs them into a sound file. The basics of this process has been tested, and both sources have produced sound files that sound as expected.

3.8 Implementation Issues and Challenges

It took some time and experimentation to iron out which software libraries would be used throughout the project. On the MIDI receiving end of the microcontroller, several libraries were tried before RtMidi was chosen. The other libraries proved to be overly complex and difficult to work with. More notably, on the other side of the microcontroller, it took a lot of testing to find a library that would enable outputting a reliable, steady waveform from the Pi. Since this output directly affects the power state of the Tesla Coil, it was imperative to find a way to create this output without jitter or other anomalies. Most libraries use the microcontroller's system clock and a process with a sleep command. These processes can be interrupted by the Pi's kernel, thus disturbing the output. The solution was to use the Raspberry Pi's PWM pins, since they use their own clock and continue to operate regardless of the processes being run by the Pi's kernel.

With regards to the tesla coil itself, it is by its nature a difficult circuit to work with. In order to get a better understanding of the tesla coil, a commercial Zeusaphone made by OneTesla was ordered. However, it has taken considerably longer to arrive than initially expected, thus delaying our research and design. Since a tesla coil produces such high voltages, it has the ability to destroy itself and harm people in its environment if designed incorrectly. This is why safety considerations are so important for this project.

4 Closing Material

4.1 Conclusion

As our society grows more embedded with technology, we will need more engineers with an electrical and computer background. To attract more students to the ECpE department at ISU, a musical Tesla coil (Zeusaphone) will be created. This Zeusaphone will be playable both by a MIDI keyboard and by MIDI files stored on a microcontroller. The microcontroller will emit its own WAP, allowing the presenter of the coil to easily connect to it and control it. The microcontroller will then control the Zeusaphone. The dazzling displays from the Zeusaphone will inspire prospective students and encourage them to join the ECpE department.

4.2 References

abyz.me.uk/rpi/pigpio/index.html. *pigpio library.* [online]. Available at: abyz.me.uk/rpi/pigpio/.

Csie.ntu.edu.tw. (n.d.). *MIDI Channel Mode Messages*. [online] Available at: https://www.csie.ntu.edu.tw/~r92092/ref/midi/midi_channel_mode.html [Accessed 1 Dec. 2018].

Instructables.com. (2018). *Build and Code a MONSTER Musical Tesla Coil With a Microcontroller*. [online] Available at: https://www.instructables.com/id/Build-a-Musical-Tesla-Coil-like-a-Pro/ [Accessed 1 Dec. 2018].

Kaizer Power Electronics. (2012). *Kaizer DRSSTC II*. [online] Available at: http://kaizerpowerelectronics.dk/tesla-coils/kaizer-drsstc-ii/ [Accessed 1 Dec. 2018].

Kaizer Power Electronics. (2016). *Musical SSTC/DRSSTC interrupter*. [online] Available at: http://kaizerpowerelectronics.dk/tesla-coils/musical-sstcdrsstc-interrupter/ [Accessed 1 Dec. 2018].

Learn.adafruit.com. (2018). *Setting up a Raspberry Pi as a WiFi access point*. [online] Available at: https://learn.adafruit.com/setting-up-a-raspberry-pi-as-a-wifi-access-point/overview [Accessed 1 Dec. 2018].

Midi.org. (2018). *Summary of MIDI Messages*. [online] Available at: https://www.midi.org/specifications-old/item/table-1-summary-of-midi-message [Accessed 1 Dec. 2018].

Music.mcgill.ca. (2017). *The RtMidi Tutorial*. [online] Available at: https://www.music.mcgill.ca/~gary/rtmidi/ [Accessed 1 Dec. 2018].

oneTesla.com. (n.d.). *oneTeslaTS Schematic*. [online] Available at: http://onetesla.com//media/wysiwyg/downloads/tsschem.png [Accessed 2 Dec. 2018].

Personal.kent.edu. (n.d.). *MIDI Channel Voice Messages*. [online] Available at: http://www.personal.kent.edu/~sbirch/Music_Production/MP-II/MIDI/midi_channel_voice_messages.htm [Accessed 1 Dec. 2018].

Sapp, C. (2018). midifile. [online] Available at: https://github.com/craigsapp/midifile [Accessed 2. Dec. 2018].

Thestk, "thestk/rtmidi," *GitHub*, 14-Sep-2018. [online]. Available: https://github.com/thestk/rtmidi.