MIDI Zeusaphone

Design Document

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

Team:
Jacob Feddersen
William Brandt
Luke Heilman
Gregory Harmon
Leo Freier
Gunnar Andrews

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

Revised: 5/8/2019 - Version 3.0

Table of Contents

List of Figures

List of Tables

List of Symbols

List of Definitions

CAD: Computer-Aided Design - software used to help create and test designs

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

DRSSTC: Dual Resonant Solid State Tesla Coil - a more advanced SSTC which adds a tank capacitor across the primary coil, which, when tuned properly, allows for greater flow of current through the primary (and thus bigger sparks from the secondary)

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.

MOSFET: Metal-Oxide Semiconductor Field-Effect Transistor - a common transistor, in this case used to switch the Tesla coil on and off

PCB: Printed circuit board.

PVC Pipe: Polyvinyl Chloride Pipe - a rigid plastic pipe often used in plumbing - in this case, used for construction of the tesla coil

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

SolidWorks: a CAD program used to design virtual parts and assemblies

SSTC: Solid State Tesla Coil - a tesla coil design which can be modulated, producing audio

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

Top Load: a large capacitive object placed on top of the secondary coil of the tesla coil that helps create an electric field - often in the shape of a toroid

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 also thanks Lee Harker and the rest of the ETG in Coover for sharing their invaluable knowledge and assisting in designing the physical aspects of the project. The team would also like to thank our 1st semester advisor Craig Rupp.

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 arcade cabinets that were constructed by previous electrical and computer engineers. They are rarely seen in use. 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 is able to play preset songs, as well as load songs to be played in this way. It can also be played with a MIDI music keyboard so there is a more interactive aspect. The project includes specialized circuits and a microcontroller. It will appeal to prospective students with an interest in embedded/low level software, circuit design, and power electronics. The wide range of topics should help the project be successful by potentially appealing to many different student interests. Because it will potentially be shown on tours, an operating manual is provided to ensure the operator is using the Zeusaphone properly. An additional safety manual and proper signage will also be provided so that operators are aware of safety when operating the device.

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 in a reasonable location. Outdoors in good weather should not pose a problem, but there might be an unforeseen risk from

doing so. It is recommended to operate the Zeusaphone indoors. There may be a problem with dust build-up if it is stored for an extended period of time. Due to ozone generation, make sure to operating the Zeusaphone in a well-ventilated and open room. However, the Zeusaphone can be run for a few minutes without ozone problems even if operated in a small room.

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. It is very loud when operating, so a lecture hall audience can definitely hear it. However, the sparks are probably not large enough to be visually impressive from the other end 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 for a safe startup and/or shutdown.

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.
- A combination of input will not result in a dangerous situation.

- The system can be safely shut down and de-energized immediately at any time.

Limitations:

- The end product is 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.
- User and Safety Manuals (May 2019)
    - The user manual, or operating manual, provides general information for an operator of the Zeusaphone. It is created to be thorough but brief enough so an operator will not need extensive training. The safety manual is written with the same philosophy, but provides thorough safety information while the user manual provides thorough operating information.
- Keyboard (May 2019)
    - Along with the Zeusaphone, a MIDI keyboard is provided. There are instructions inside the user manual on how to connect the keyboard to the Zeusaphone. This keyboard provides an alternative live input into the Zeusaphone, as opposed to playing loaded songs.

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 had 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 are 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 is used to select between keyboard mode or just playing a loaded song. When in keyboard mode, the pitch bending knob will (by intentional design) affect the frequency. This ranges from plus or minus two semitones from the original notes. It is essentially an added feature that a normal keyboard would have.

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
- 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.

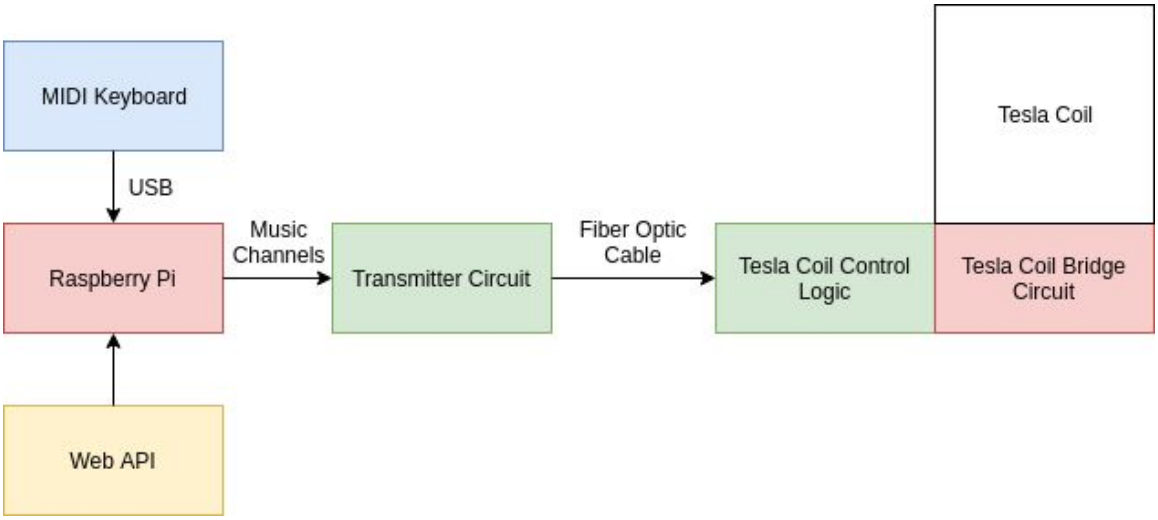

Figure 1: Overview of the Project Layout

The tesla coil itself is classified as a solid state tesla coil (SSTC), which means it uses semiconductors (not a spark-gap) to drive the primary coil. Our tesla coil uses power MOSFETs to switch the voltage across the primary coil, which then causes the secondary coil to resonate. This resonation steps the voltage up drastically in the secondary coil, and that causes the air to breakdown around the top of the coil, releases sparks and sound. For the high frequency switching to occur, the MOSFET's are driven by the control logic circuit. This circuit takes in the output from the transmitter circuit and syncs it with the output of the coil with an antenna. The synced signal is then used to drive the MOSFETs. The secondary coil is wound with 30 gauge magnet wire, and attaches to a conductive, toroidal topload. This topload adds capacitance to the secondary, allowing the coil to build up more energy before it releases it as sparks. The coil and its associated electronics has a base of 8 inches by 8 inches, and is shy of 2 feet tall.

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.

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. Both applications write the notes in the same way to a Unix socket. The apps are 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 the socket where notes will be read and generated.

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, note off, and pitch bend. 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 a socket, and it generates output waveforms on the Raspberry Pi GPIO pins accordingly. Frequency is varied based on the notes being played, which is what determines the pitch of the note.

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 series of pulses at exactly the frequency set no matter what the workload of the processor is, and no kernel process can preempt or delay them. Issues came from using other pins because other tasks on the Pi can stall the output wave, leading to notes being played incorrectly with lots of static sound. The final waveform on the PWM pin will not be affected by these problems. Each channel is then sent as two outputs into the transmitter circuit, so two channels of music can be played. The transmitter circuit smooths the waveform before it is finally sent to the coil.

2.2.3 The Transmitter Circuit

The purpose of the transmitter circuit is to convert the signals output from the Raspberry Pi to a waveform over a fiber optic transmitter. The circuit also contains some logic to add a minimum time before another wave can start, so two notes played directly on top of each other will not create an extended pulse. This extended pulse causes an unclean note that has the potential to damage the coil. This logic is done in hardware rather than software because of the time sensitivity; even small delays in this logic will drastically affect the output of the coil.

The three LEDs output from the Pi (LEDR, LEDY, and LEDG) are signal LEDs that show the state of the transmitter circuit. The red LED shows that the Pi is powered on. The

green LED signals that the Tesla Coil driver software is running. The yellow LED signals that a program (either a keyboard or a Midi file) is connected to the driver software and running on the Tesla Coil.

The data sheets for the LEDs state that 30 mA is the maximum forward DC current they can handle. We did not want to push this, so we decided to select resistor values that placed the current about 20 mA. The LEDs are powered by 3.3V rails. The green and yellow LEDs draw 2.4 V, while the red LED draws 2 V. The remaining voltage needs to go across the resistor. The following equations show the how the resistor values were selected:

$$\frac{3.3-2.4V}{20mA} = 45\Omega \text{ (green and yellow LED)}$$

$$\frac{3.3-2V}{20mA} = 65\Omega \text{ (red LED)}$$

Since we already had 47 and 68 Ohm resistors, we just used those. This makes the LEDs produce the maximum brightness we are willing to let them create. Combined they draw about 60 mA, which is acceptable for the Raspberry Pi power supply.

The Raspberry Pi's two PWM pins each carry info for a note channel. These two signals are combined with an OR gate in IC1. Since these output pins function at 3.3V, but the 555 timers require a 5V signal, a pull-up transistor (IC5) is used to bring up the voltage to 5V. This also inverts the signal which is then fed into a second OR gate (IC4), which operates with the 555 timers. Both 555 timers operate in monostable mode. The first 555 timer (IC6) acts as a timer - when it receives a pulse (active-low) it goes high for 110 µs. This output is fed back into the OR gate. When the 2nd 555 timer (IC7), which is also active-low, receives a pulse, it goes active for 90 µs. This signal is finally fed into the fiber optic transmitter.

The amount of time a 555 timer operating in monostable mode stays active after being enabled is given by the following equation [7]:

$$t = 1.1 * R * C$$

Rearranging to solve for R gives

$$R = \frac{t}{1.1*C}$$

With a capacitance of 0.01 µF, and times of 110 µs and 90 µs, the resistor values are calculated to be 10 kΩ and 8182 Ω, respectively. Thus, 10 kΩ and 8.2 kΩ resistor values were used as the closest standard resistance values available to what we calculated.

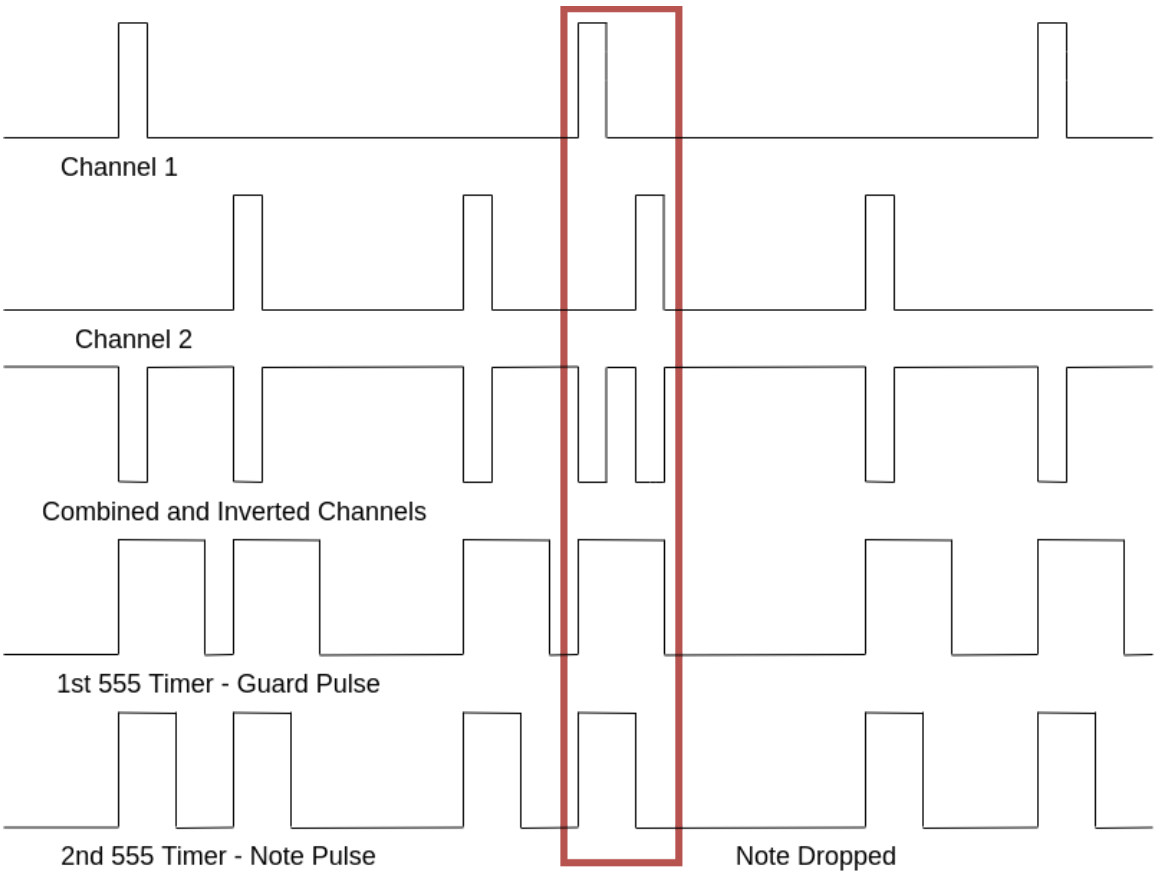

Figure 2: Graph of Transmitter Waveforms

The transmitter was also designed to operate within the power limits of the Raspberry Pi power supply, which supplies 2.5A max. The Raspberry Pi will use at most 1A. The LEDs require 60mA. The MIDI keyboard uses at most 500mA since it operates within USB 2.0 limitations. The fiber optic transmitter uses 30mA. This totals up to about 1.6A, which is well within the 2.5A limit.

The final output of the transmitter circuit is a single waveform being sent into a fiber optic transmitter into the fiber optic cable. All of the transmitter circuitry is assembled onto a single printed circuit board. The Tesla coil receives the waveform from the transmitter through the fiber optic receiver. This waveform is then used to trigger the coil on and off. A fiber optic connection is used as it is not affected by the magnetic field generated by the tesla coil.

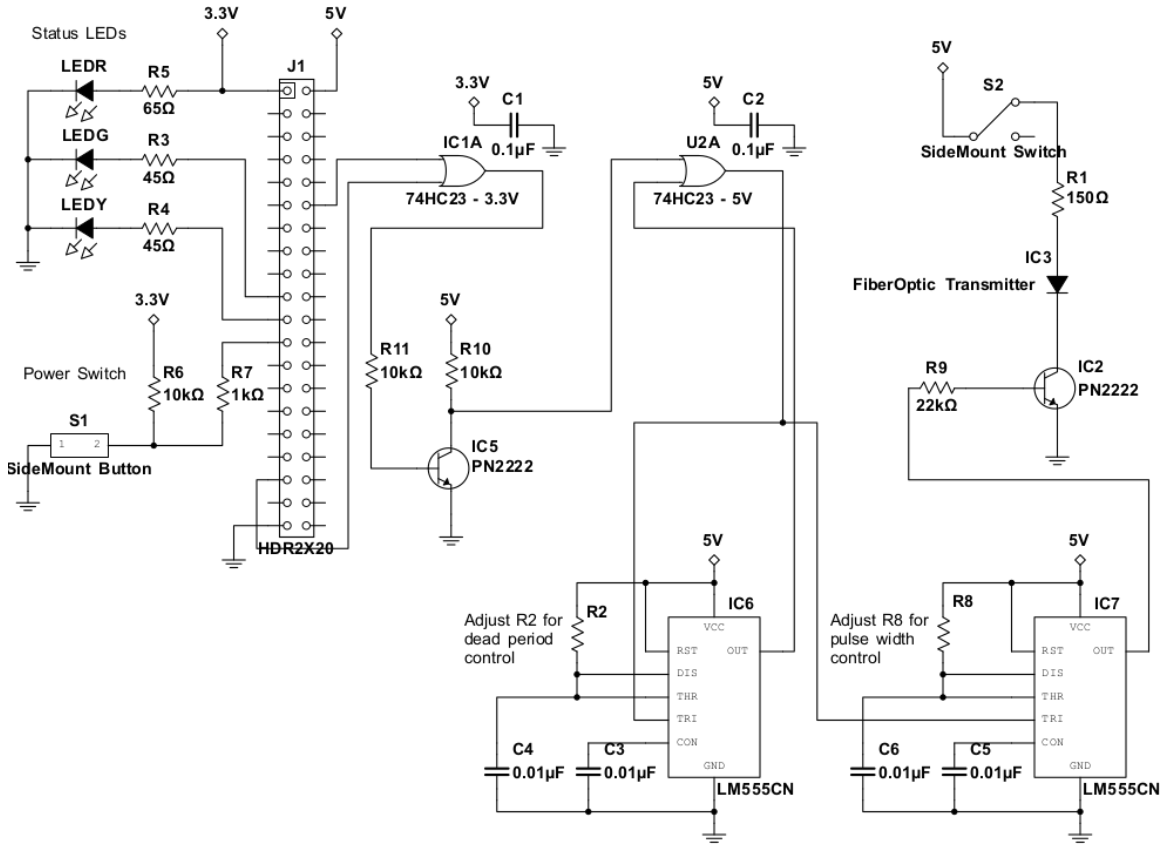

Figure 3: Transmitter Schematic

Schematic Reference	Description	Digi-Key Part Number
R1	150 Ω Resistor	-
R2, R6, R10, R11	10 kΩ Resistor	-
R3, R4	47 Ω Resistor	-
R5	68 Ω Resistor	-
R7	1 kΩ Resistor	-
R8	8.2 kΩ Resistor	-
R9	22 kΩ Resistor	-
C1, C2	0.1 µF Capacitor	478-7337-1-ND
C3, C4, C5, C6	0.01 µF Capacitor	C322C103K3G5TA7301-ND

S1	Side-Mount Button	450-1662-ND
S2	Side-Mount Switch	CKN9559-ND
LEDR	Red LED	751-1130-ND
LEDY	Yellow LED	751-1147-ND
LEDG	Green LED	TLHG4600-ND
IC1, IC4	OR Gate	296-1589-5-ND
IC2, IC5	PN2222 Transistor	PN2222TACT-ND
IC3	Fiber Optic Transmitter	FB162-ND
IC6, IC7	555 Timer	-
J1	2x20 Header	S6104-ND
IC1, IC4 Mount	DIP-14 Mount	AE9989-ND
IC6, IC7 Mount	DIP-8 Mount	AE9986-ND

Table 1: Transmitter Parts

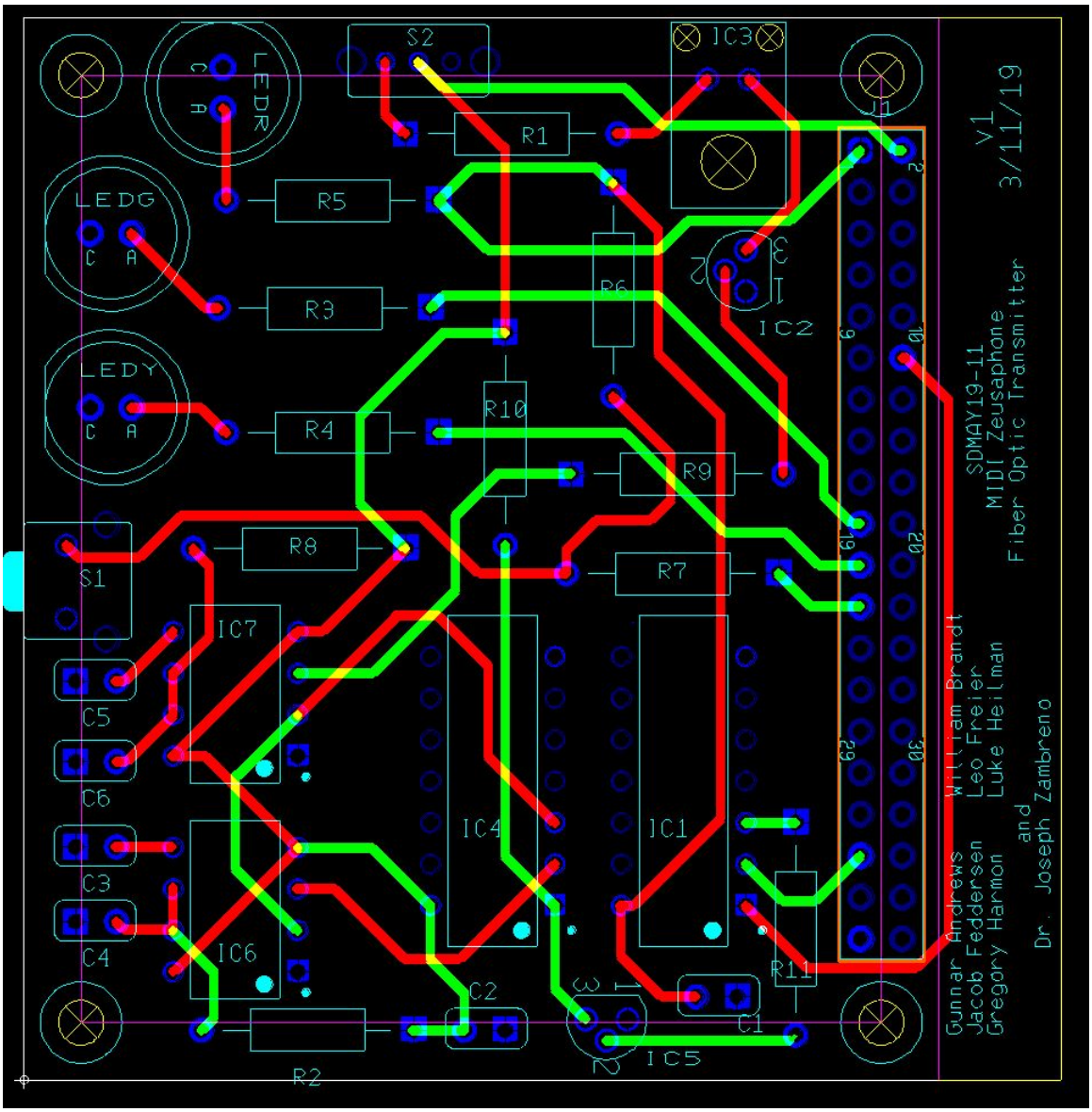

Figure 4: Transmitter Printed Circuit Board Layout

2.2.4 Power Provider and Input Logic

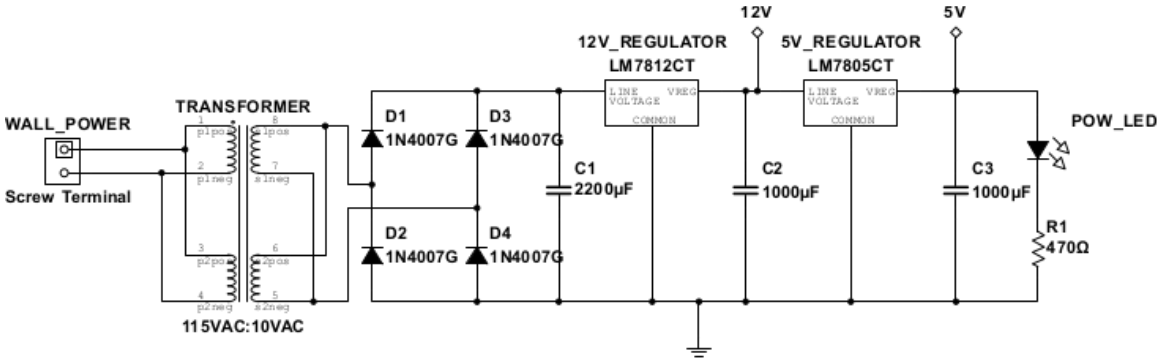

Figure 5: Tesla Coil Power Provider Schematic

The power provider circuit is simply a circuit that provides 12V and 5V rails for the rest of the tesla coil. It uses a transformer to convert 120V from the wall power to a lower voltage. We initially designed it to go from 120V to 20V. However, we discovered that transformers don't output a constant voltage - it depends on the current draw. Since the circuit does not draw much current, the output voltage of the transformer was actually closer to 40V, which is way too high for the 12V regulator. The transformer could be re-wired to output 10V instead, which when used with the circuit actually output around 16.2V, which works well with the 12V regulator.

After stepping down the voltage, the power is rectified with a full bridge rectifier before being powering the 12V regulator, which in turn powers the 5V regulator. A red LED was attached to the 5V rail as well to indicate if the power provider circuit is running. Note that it does take several seconds for resistor attached to the LED to bleed off remaining energy after the circuit is unplugged.

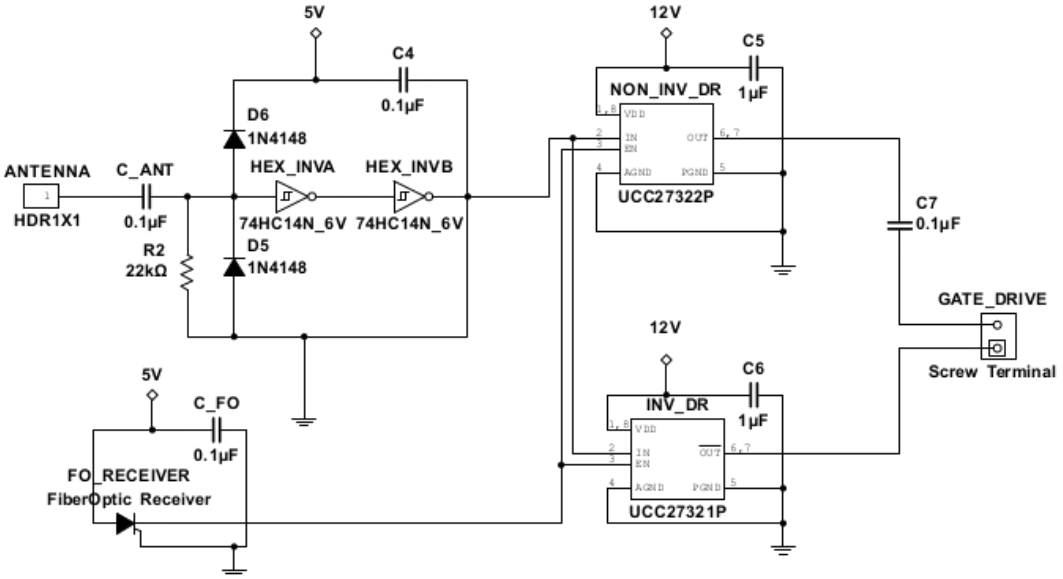

Figure 6: Tesla Coil Input Logic Schematic

The input logic circuit receives the signals generated by the transmitter via a fiber optic cable and synchronizes them with the resonance of the coil. The fiber optic receiver's signal is used to drive two MOSFET drivers (INV_DR and NON_INV_DR), which are designed to connect to MOSFETS and switch them on and off according to the signals they receive. One driver is an inverting driver, while the other is a non-inverting driver - this allows one driver to switch off while the other switches on. The output of these drivers are then fed to the bridge PCB.

The drivers are synchronized with the coil itself by attaching the enable pins of the drivers to the output of an antenna, which picks up the electric field generated by the coil. To convert the antenna's output to a digital signal, it is run through a Not Gate (the HEX_INV) twice. To prevent the gate from getting fried by large signals from the antenna, the antenna's output is clamped between 0V and 5V by two diodes, one tied to ground, one tied to 5V.

Schematic Reference	Description	Digi-Key Part Number
R1	470 Ω Resistor	-
R2	22 kΩ Resistor	-
C1	2200 µF Capacitor	493-1086-ND
C2, C3	1000 µF Capacitor	1189-1583-1-ND

C4, C7, C_FO	0.1 µF Capacitor	478-7337-1-ND
C5, C6	1 µF Capacitor	445-173436-1-ND
C_ANT	0.1 µF Capacitor (high voltage rating)	445-173324-1-ND
POW_LED	Red LED	751-1130-ND
D1, D2, D3, D4	1N4007 Diode	1N4007FSCT-ND
D5, D6	1N4148 Signal Diode	1N4148FSCT-ND
TRANSFORMER	115VAC:10VAC Transformer	VPP20-1000-B-ND
12V_REGULATOR	12 V Regulator	MC7812CTGOS-ND
5V_REGULATOR	5 V Regulator	MC7805CTGOS-ND
HEX_INV	Hex Inverter w/ Schmitt Trigger	296-1577-5-ND
INV_DR	Inverting Gate Driver	296-13686-5-ND
NON_INV_DR	Non-Inverting Gate Driver	296-13689-5-ND
FO_RECEIVER	Fiber Optic Receiver	FB123-ND
WALL_POWER, GATE_DRIVE	Screw Terminal	277-1667-ND

Table 2: Power Provider and Input Logic Circuit Parts

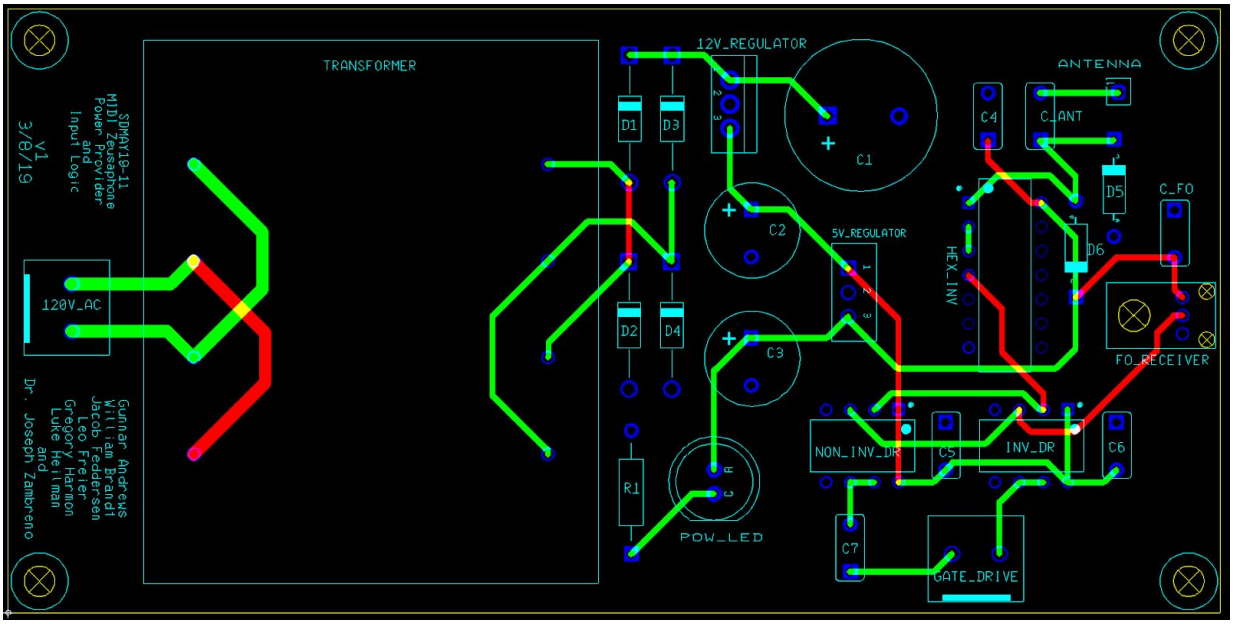

Figure 7: PP/IL Printed Circuit Board Layout

2.2.5 Bridge Circuit

The bridge circuit uses the output from the input logic circuit and a 120VAC source to create an alternating voltage across a load consisting of the primary coil (PRIMARY) and two capacitors (C2 & C3). This is achieved by first using a full bridge rectifier to rectify the AC voltage. C4, R5, and R6 are added to smoothen out the voltage ripple to create a more uniform DC voltage. Then the rectified voltage is used as the source on a full bridge single-phase inverter.

The four power MOSFETs (PWR_MOS_1-4) on the inverter are driven by the gate drive transformer (GDT) which is a one to one transformer that allows for the signal from the input logic circuit to be used while also isolating one circuit from the other. To achieve the functionality of the inverter from one signal, MOSFETs 3 and 2 are connected to the GDT in reverse of MOSFETs 1 and 4. Therefore, when a HIGH signal is coming from GATE_DRIVE_IN, MOSFETs 1 and 4 will conduct, while 3 and 2 will be open and vice versa for when a LOW signal is sent. This ultimately creates the desired square wave of +108V to -108V across the primary coil and parallel capacitors.

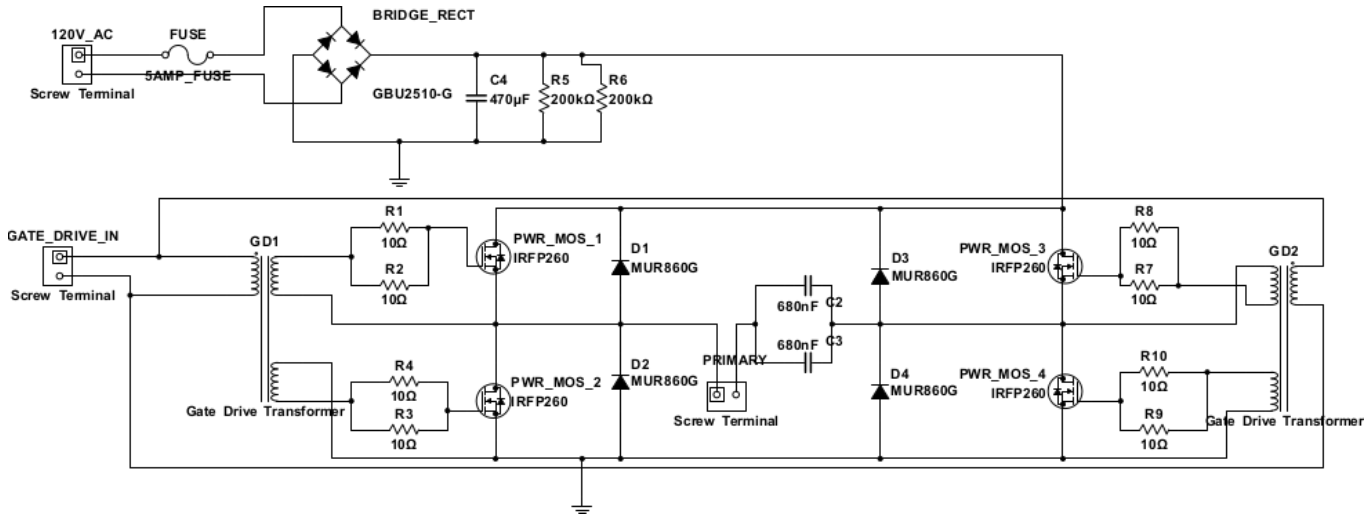

Figure 8: Tesla Coil Bridge Schematic

Schematic Reference	Description	Digi-Key Part Number
R1, R2, R3, R4, R7, R8, R9, R10	10 kΩ Resistor	-
R5, R6	200 kΩ Resistor	-
C2, C3	680 nF Capacitor	495-1299-ND
C4	470 uF Capacitor	493-7069-ND
D1, D2, D3, D4	Flyback Diode	MUR860GOS-ND
FUSE (brackets)	Brackets to hold fuse	F4189-ND
FUSE	5A Fuse	507-1232-ND
BRIDGE_RECT	Full-Bridge Rectifier	641-1374-5-ND
GD1, GD2	Gate Drive Pulse Transformer	553-1646-5-ND
PWR_MOS1, PWR_MOS_2, PWR_MOS_3, PWR_MOS_4	IRFP260NPBF Power MOSFET	IRFP260NPBF-ND
120V_AC, GATE_DRIVE_IN, PRIMARY	Screw Terminal	277-1667-ND

Table 3: Bridge Circuit Parts

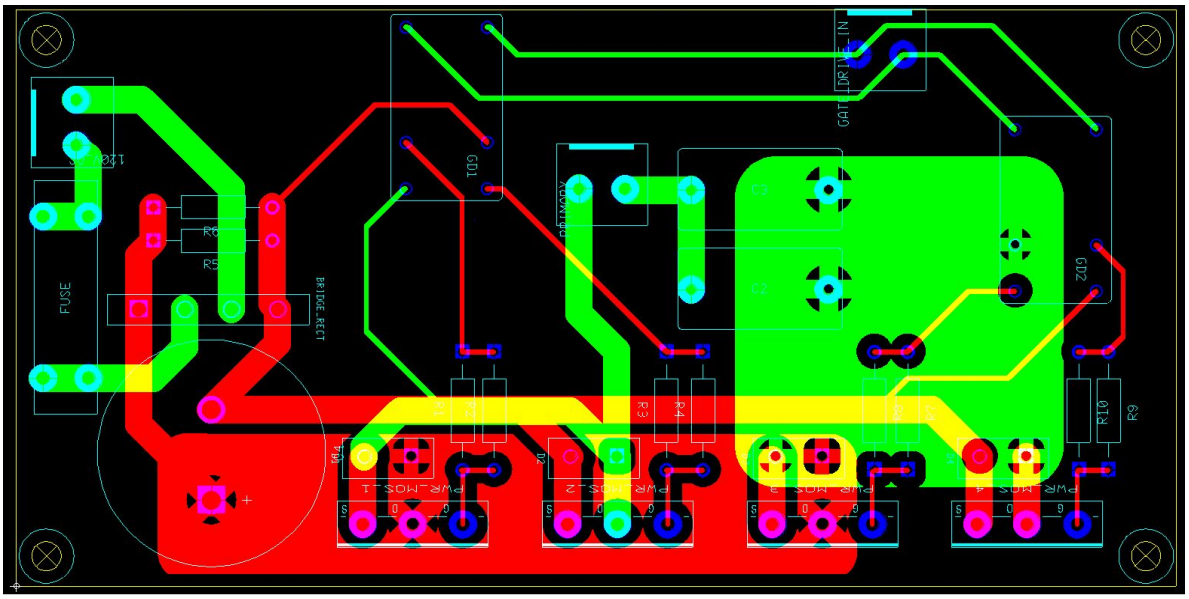

Figure 9: Bridge Printed Circuit Board Layout

2.2.6 Physical Construction

A case was made for both the transmitter component and the coil component of the project. The cases were made with acrylic, since the material is relatively cheap and durable. It is made with clear acrylic specifically to allow people to see the interiors of the components. The case designs were modeled in the SolidWorks CAD software and then cut with a laser cutter.

2.2.6.1 Transmitter Case

The transmitter case contains both the Raspberry Pi and the transmitter PCB. The transmitter PCB was created to fit on top of the Raspberry Pi and align with the Pi's screw holes. The case was made with ⅛ inch clear acrylic and cut with a laser cutter. The pieces are held together with screws and standoffs that fit through the screw holes of both the transmitter PCB and the Pi PCB. Care was taken to align holes in the case to access the buttons, switches, and ports on the PCBs.

The following figure shows the amount of the case that was designed within SolidWorks. The Raspberry Pi 3D model compliments of [2].

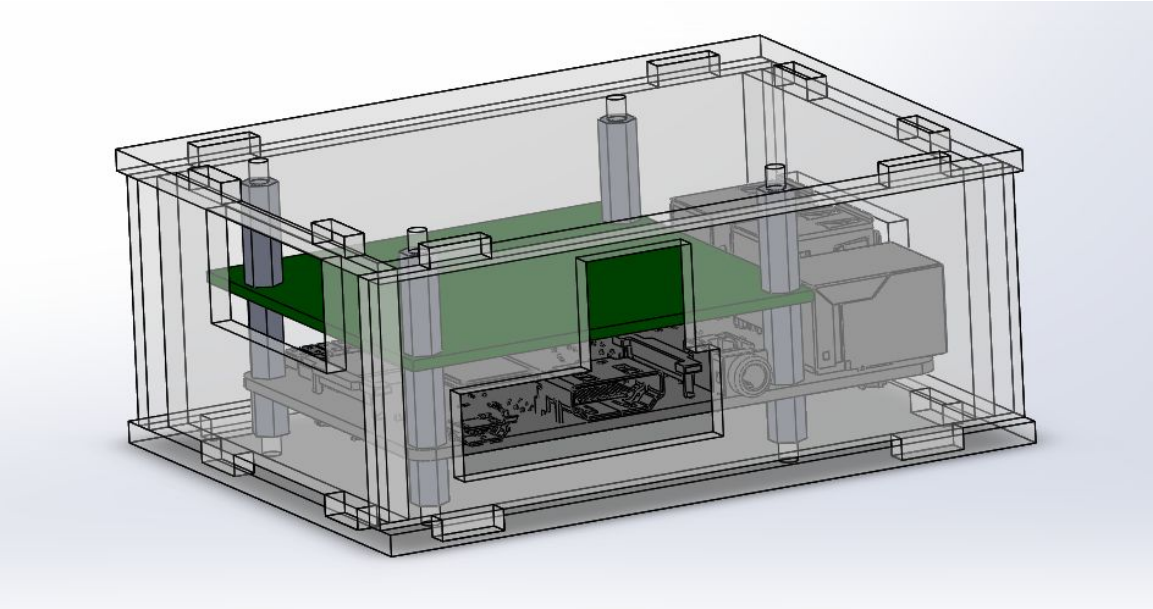

Figure 10: Transmitter Case

The following table shows which screws and standoffs are used in the transmitter case:

Purpose	**Material**	**Screw Length**	**Thread Size**
Bottom Case to Pi Standoff (M-F)	Metal	7mm	M2.5
Pi to Transmitter Standoff (M-F)	Metal	12mm	M2.5
Transmitter to Top Case Standoff (F-F)	Metal	15mm	M2.5
Top and Bottom Screws	Metal	6mm	M2.5

Table 4: Transmitter Screws and Standoffs

2.2.6.2 Tesla Coil Case

The case for the Tesla Coil contains the Input Logic/Power Provider PCB and the Coil Bridge PCB. The two PCBS are placed next to each other in the case, again held in place with standoffs and screws. The two PCBs are connected with wire that attaches to screw mounts on each PCB. Since the case has the secondary coil and topload attached to it as well, it was made with ¼ inch clear acrylic. The case itself was designed with an open side to allow an easy view of the eternal electronics. Two sides are made from acrylic, while the last side consists of the aluminum heat sink for the power MOSFETs in the bridge circuit. Screw holes were bored and tapped into the aluminum heat sink to allow the acrylic top and bottom to screw into the heat sink. The heat sink

also had holes that were bored and tapped to allow the MOSFETs to be screwed into the heat sink. The MOSFETs each had sil pad separating them from the heat sink, allowing thermal contact while preventing electrical contact.

The secondary coil is attached with a PVC flange. The flange is attached to the case with screws and bolts. The PVC pipe rests inside the flange - it is a snug fit. There are holes in the top of the case for the antenna and the primary coil wire.

The following figure shows the amount of the tesla coil case that was designed within SolidWorks. The 3D models of the AC Power Receptacle and the PVC flange compliments of [4] and [9], respectively. Note the PVC flange model is slightly different than the actual flange used - the footprint of the holes on the flat side is different.

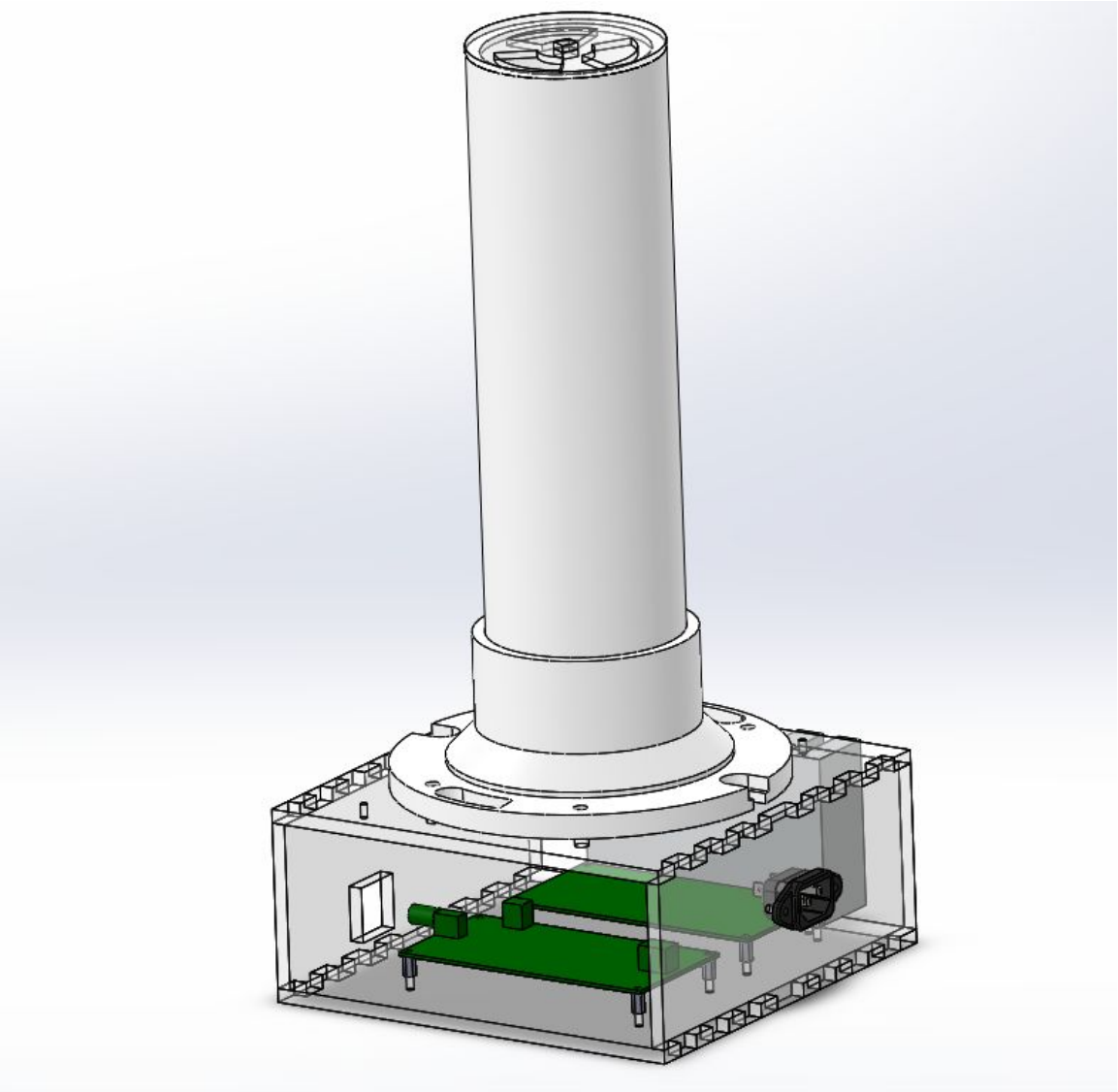

Figure 11: The Coil Case with Secondary PVC

The following table shows which screws and standoffs are used in the coil case:

Purpose	Material	Screw Length	Thread Size
MOSFET Heatsink	Metal	10mm	M3
GND Heatsink	Metal	8mm	M3
Case to Heatsink	Metal	12mm (except the one on the top close to the 470nF capacitor, which is 10mm)	M3
Case Top to Case Sides	Nylon	14mm	4-40
120V Wall Adapter	Metal	0.5inch	M3 or smaller
PCB to Standoff and Standoff to Case	Metal	9mm	4-40
PCB Standoff	Metal	7mm	4-40

Table 5: Tesla Coil Case Screws and Standoffs

2.2.6.3 Secondary Coil Winding

The secondary coil consists of 3" PVC pipe cut to 14" in length, with 30 gauge wire wound around the outside.

In order to wind and varnish the [number of turns here] turns on the secondary coil, a small rig was constructed out of 80/20 aluminum extrusion. The rig consists of frame slightly longer than the coil itself with arms on either end that stick up about 5 inches. ¼ inch screw holes were drilled into these arms. The coil could then be placed into the rig and spun about the center of its cylinder. In order to connect the framework to the coil, acrylic end caps were made for each end. Each end cap had two layers - one that fits just inside the cylinder, and one that fits right on top of the cylinder. The end caps had square holes cut in their center to allow a carriage bolt to fit into them. After gluing the two parts of an end cap together, a carriage bolt was secured in the hole with a washer and a nut, with the carriage bolt sticking out away from the cylinder. Special care was taken to ensure that the bolts were completely perpendicular to the end caps. These end caps were then glued onto the PVC cylinder. The bolts sticking out of the end caps can then be placed into the holes in the rig, allowing the whole coil to spin freely. A drill was attached

to on of the bolts, and used to spin the coil both for winding the wire and spraying/drying the varnish.

One end cap was made with spokes instead of being solid. This allowed the bolts to be unscrewed and then slipped out of the cylinder through the holes. The other end cap was left solid to allow the topload to be screwed onto it. Both end caps were left on the cylinder in the final coil.

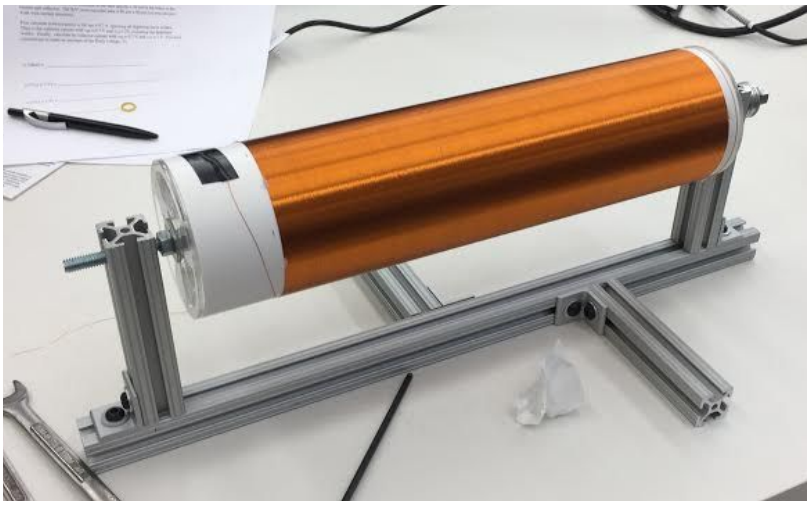

Figure 12: Winding Rig After Completion

The secondary coil has a measured resistance of 94 Ohms. With the topload, it resonates at around 290 kHz.

2.2.6.4 Topload

The toroidal shape of the topload was made with standard 3" heating duct, wrapped around to make a loop. The two ends are held together with aluminum tape. To attach the topload, two small disks that fit inside the heating duct loop were cut from cardboard and wrapped in aluminum foil. These were then taped to the heating duct loop. Holes were cut in the middle of the disks that are big enough to allow the bolt sticking from the top of the secondary pipe to fit through. The topload was then connected with washers and nuts to the bolt. The top of the secondary coil was soldered and taped to the topload, allowing electrical connection. Some copper wire was added on the top to serve as a breakout point.

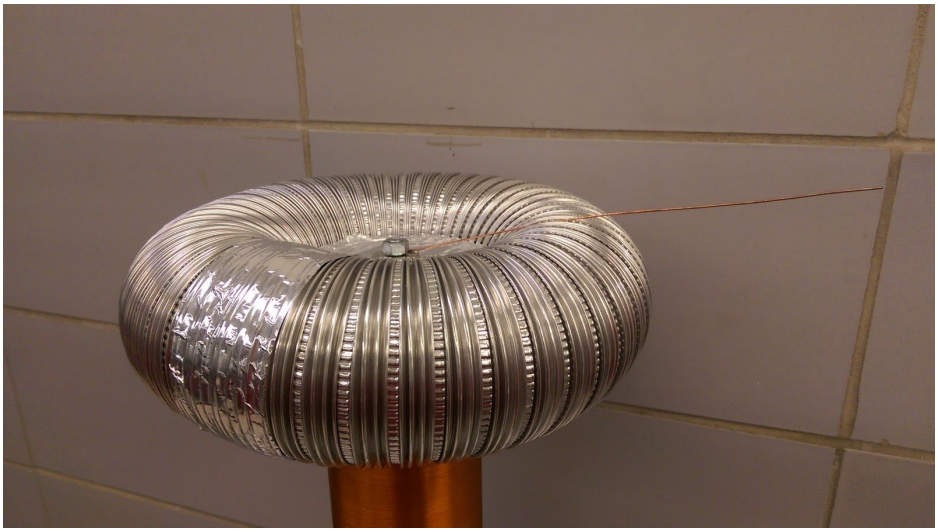

Figure 13: Tesla Coil Topload and Breakout Point

3 Testing and Implementation

3.1 Interface Specifications

3.1.1 Interface between MIDI Keyboard and MIDI Keyboard App

The keyboard is interfaced with the Raspberry Pi and the MIDI Keyboard application running on the Raspberry Pi via a USB cable. They communicate using the USB MIDI protocol. However, we did 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 Transmitter Circuit and Tesla Coil

The transmitter circuit interfaces with the input logic circuit 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 is a simple on/off state. When the channel is active, the tesla coil will be on, producing sparks. When the channel is inactive, the tesla coil will be off. Turning the tesla coil on and off at a certain frequency produces sound of that frequency. The transmitter sends a (90us) pulse at various frequencies. A standard pulse width is due to the design of the transmitter, even though duty cycle would be a much better solution.

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).

The output of the driver was tested by viewing the wave on an oscilloscope. It was compared to the OneTesla waveform before using it as the input into the OneTesla's coil. Once confirmed that the waveforms were similar in pulse width and frequency, it was sent to the coil with suboptimal results. The sound output contained lots of noise, and the coil would intermittently output a large spark accompanied by a larger noise. At this point, we developed the transmitter circuit to clean up the waveform. Then, we got the OneTesla running with the Raspberry Pi's wave generation through the breadboard transmitter. Our

whole transmitter was then confirmed to work similar to the OneTesla's interrupter, and produce slightly better arcs and louder sound.

A waveform generator and a power supply was used to provide controlled inputs into the circuits. This enabled testing of ideal and extreme cases of voltages and waveforms so that we could test the limits of the circuit without creating an uncontrolled and unsafe testing environment that could potentially damage test equipment, the Zeusaphone, or others nearby. This method of testing was used extensively when debugging the entire hardware system with the OneTesla coil.

3.3 Functional Testing

Both forms of input (keyboard and MIDI files) needed to be verified. This was tested in stages, since the layers have been modularized. As components were completed, they were tested in combination with the OneTesla system to confirm functionality.

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.

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. This was mainly tested by plugging the output into a speaker and listening to the song that was supposed to play. This helped us fix an issue that was solved by moving the output to the PWM pins on the Raspberry Pi and switching libraries for GPIO interfacing. The final level of driver testing was done by sending the output to the OneTesla's full coil setup, which was described in section 3.2.

3.3.3 Tesla Coil Testing

All circuits were tested in breadboard form with low voltages to make sure the overall design would work. We constructed a 'mini' coil to test our circuit designs at low voltages where a mistake would not lead to the destruction of the circuit or the coil. The 'mini' coil proved to be extremely useful as it added a layer of preliminary testing to debug some problems and finally confirm our designs. A major milestone was when we could play music on the mini coil using hardware entirely designed ourselves with no components from the OneTesla kit. This setup involved the Raspberry Pi running our software, the breadboard version of the transmitter, the breadboard low-voltage bridge circuit, and the mini coil. We used the oscilloscope to verify the waveforms on several pins throughout the circuit, and we were also able to view the induced voltage from the coil in the air to verify that the coil was oscillating properly and generating the correct frequencies.

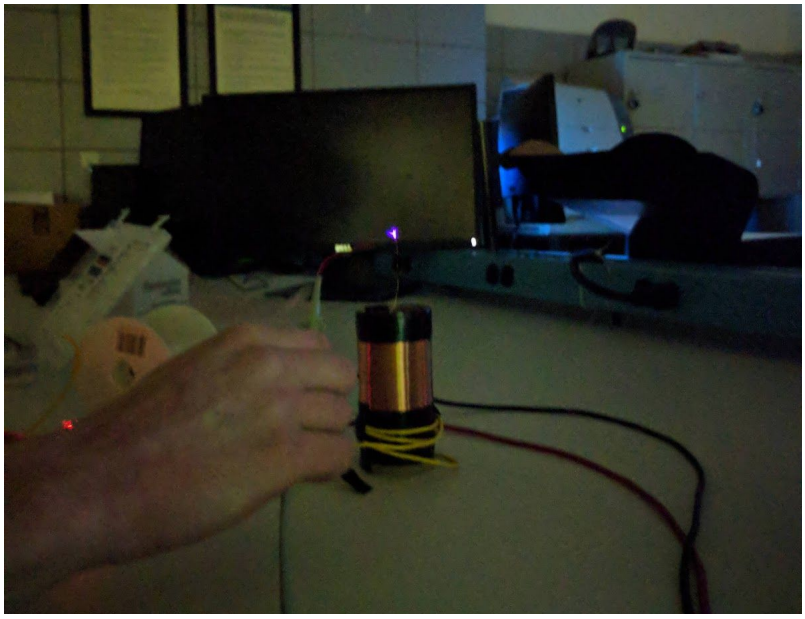

Figure 14: The Mini Coil

After confirming functionality on the breadboard, high voltage versions of the circuits were prototyped on perfboard, which was connected to the primary coil from the OneTesla kit. We did not use a perfboard iteration for the transmitter circuit, since it did not use high power components. Before designing and ordering PCBs, we made sure the circuits displayed the same behaviour when running on high voltage. The major issue we discovered at this stage was that the higher voltage coil suffered from noise induced by itself back in its control circuits. We traced the noise using the oscilloscope to the antenna input, and when we touched the oscilloscope to this pin the noise disappeared. We deduced that the pin was floating, and we solved the issue by adding a pull-down resistor.

Finally, we tested the driver circuits with our own coil. The main thing we needed to test was the control signal pulse-width needed to make the coil to output a good volume with visible sparks. This was done by sending pulses across a fiber optic transmitter using a function generator. With this setup, we could easily change the frequency and pulse widths of the signals sent. During this testing, we pushed the coil up to 100 µs pulse width, which was impressive but a little too loud. To be on the safe side, we used a resistor in our transmitter that gave about an 90 µs pulse width.

During this testing, we also discovered some glowing about the primary coil, as seen in the picture below. We reasoned that the issue was due to the primary coil being about an inch up the secondary coil, where the voltage had already been stepped up high enough that it was trying to arc back to the primary coil. Moving the primary coil windings down below the secondary coil solved this issue.

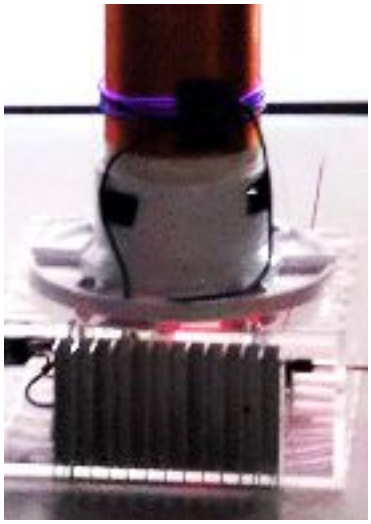

Figure 15: Glowing around the primary coil

3.4 Non-Functional Testing

3.4.1 Ease of Use

To facilitate the ease of use, we are providing a user manual to explain how to use the Zeusaphone. Hopefully this should make it easy to set up and use for someone with no knowledge of the system. Of course, the system is still designed to be easy to use, but is too complex for someone to reasonably use without instruction. The web interface is also a solution to ease of use problems. It is designed to be straightforward with only necessary UI elements. Connecting to WiFi is familiar to people, so the Raspberry Pi's access point is something that users should already understand. Since the end goal of the project relies on it being shown as a live demo, the parts that users interact with should be easy to use for them. The user manual is a comprehensive resource, but remains brief with the main points of operation.

3.4.2 Reliability Testing

Testing was done as components were developed. We tested the functionality of those components along with whatever previous components were completed at the time. There were dozens of configurations that we tested with combinations of breadboard, perfboard, and OneTesla components. When debugging a circuit, it would normally double as stress testing since we spent a good amount of time making sure everything worked as expected. Unexpected results from the coil or oscilloscope were our indicators of problems in the system. We tested the full tesla coil circuit with much more pulse width than what the Zeusaphone will use when being played. This was from normal debugging of the circuit, but also testing to see how large we could make the arcs since the full circuit was completed on perfboard during these tests.

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 entered, playing back the song.

The physical designs of the cases were modeled in SolidWorks, while the PCBs were modeled in Multisim. The modeling, along with modular design, allowed different aspects to be easily tweaked without requiring the whole system to be rebuilt.

JavaTC [3] is an online tool used by tesla coil creators to get estimates of different properties of a coil. By inputting characteristics such as the coil radius and number of turns in the secondary, it will calculate estimates like the resonating frequency of the secondary coil. This tool allowed us to model different coil designs without having to actually build them.

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 were monitored with a controlled input created by a waveform function generator.

Several iterations of the circuits were made. The first (and simplest) coil was built by hand wrapping some wire around a cardboard tube. This secondary was initially tested with a very simple circuit, consisting of a diode, resistor, and a BJT (called a slayer exciter). This circuit was not capable of being controlled beyond being turned on and off, and had a nasty tendency to burn out the transistor used. However, it showed that a tesla coil could be made and operated.

The next iteration used the same mini secondary coil, but used much more advanced circuitry. The circuits were split into four main sections - the transmitter, the power provider, the input logic, and the bridge sections. The transmitter was assembled in a breadboard, and then attached to a raspberry pi. The input logic and bridge circuits were assembled on another breadboard, which was then attached to the mini secondary coil. The power provider circuit was replaced in early testing with a voltage source plugged into the breadboards. The voltage across the primary coil was created with its own voltage source, allowing us to easily control the power output by the coil. This setup allowed us to

test the output of our software on a low powered coil, which allowed us to troubleshoot problems without a high voltage factor being present.

The next iteration kept the transmitter breadboard but replaced the input logic/bridge breadboard with a perfboard. The power provider circuit was also added to this perfboard. This iteration was first tested with the mini secondary coil, to ensure that both the power provider worked, and that the circuits were placed into the perfboard properly. The mini secondary coil was then replaced with the primary and secondary coil from the OneTesla and ran at a much higher voltage (from the wall as opposed to a voltage source).

These circuits were then fabricated into three PCBs - the transmitter PCB, the input logic/power provider PCB, and the bridge PCB. The transmitter PCB was attached directly to a raspberry pi. This combined circuit was then placed into the transmitter case, which is the final transmitter of the project. The other two PCBs were placed into the coil case, and the OneTesla coil was replaced with the coil created by us. This is the final coil of the project.

3.7 Results

By the end of the project, we had tested every component thoroughly by itself, and modularly with the rest of the components. We verified that the tesla coil could play songs and live keyboard input, and we confirmed that the note frequencies that the tesla coil played were exactly the pitches we 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.

When we stepped our perboard version of the circuit up from half-bridge circuit to a full-bridge circuit, we began to have issues with noise feedback disrupting the input logic circuit. We traced the noise back through the circuit to the antenna pin. When we touched the input of the antenna to the hex inverter with the oscilloscope, the noise went away. We determined that the pin was floating, and we added a pull-down resistor which eliminated the noise.

A major issue for the hardware was the team makeup - two electrical engineers and four computers engineers. With the project being so hardware intensive, most of the computer engineers had to switch to working only on the hardware in order for the project to be finished on time.

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) was created. This Zeusaphone is playable both by a MIDI keyboard and by MIDI files stored on a microcontroller. The microcontroller emits its own WAP, allowing the presenter of the coil to easily connect to it and control it. The microcontroller controls the Zeusaphone. The dazzling displays from the Zeusaphone will inspire prospective students and encourage them to join the ECpE department.

4.2 References

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

[2] Barber, Richard, "Raspberry Pi 3 Model B+," *grabcad*, 3-Mar-2018. [online] Available at: https://grabcad.com/library/raspberry-pi-3-model-b-2

[3] B. B. Anderson, "JAVATC," *JAVATC*. [Online]. Available: http://www.classictesla.com/java/javatc/javatc.html. [Accessed: 08-May-2019].

[4] cdowney, "IEC 320-C14 Receptacle Panel Mount," *grabcad.* 21-JUL-2016. [online] Available at: https://grabcad.com/library/tag/703w-00-04

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

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

[7] Kane, Phillip, "555 Timer Tutorial," *jameco.com,* 2019 [online] Available at: https://www.jameco.com/jameco/workshop/techtip/555-timer-tutorial.html

[8] 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].

[9] McMaster-Carr, "Flanged Socket-Connect Reducing Adapter, for PVC Pipe for Drain, Waste and Vent," 2389K91 Datasheet. [online] 2015 Available at: https://www.mcmaster.com/2389k91

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

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

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

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

[14] Steve Ward High Voltage. (2009). *New DRSSTC Driver*. [online] Available at: http://www.stevehv.4hv.org/new_driver.html