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: 10-12-2018 - Version 1.0

Table of Contents

List of Figures

List of Table

List of Symbols

List of Definitions

MIDI: Stands for Musical Instrument Digital Interface. It is 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.

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

PPE: Personal protective equipment.

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

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

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

WAP: WiFi Access Point

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 ECpE, 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. 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.

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.

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). A raspberry pi was chosen for the microcontroller in part because several were immediately available for testing, and also because of the simplicity of setting one up. The input of the music comes from different sources. MIDI files can be stored on the microcontroller and played. Alternatively, the tesla coil can be controlled by a MIDI keyboard connected to the microcontroller. This connection will be made with a wire to avoid interference from the tesla coil. Either method of input is processed by a program (in the layout, the App Layer). Regardless of the input, the resulting messages from the input are of the same format, which is sent to the Driver Layer. This layer converts note on and off signals to power on and off signals for the tesla coil. These signals are sent via a GPIO pin to the interrupter. The interrupter is a physical circuit that drives the tesla coil on and off. When oscillating on and off at correct frequencies, the tesla coil releases energy in the audible and visual spectrum, seen as the electrical sparks and heard as a musical note.

In order for a user to control the raspberry pi itself, the pi hosts a simple web page accessible through http using any web browser. This web page will enable a user to upload and remove MIDI files to the pi, as well as select to play these songs. 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.

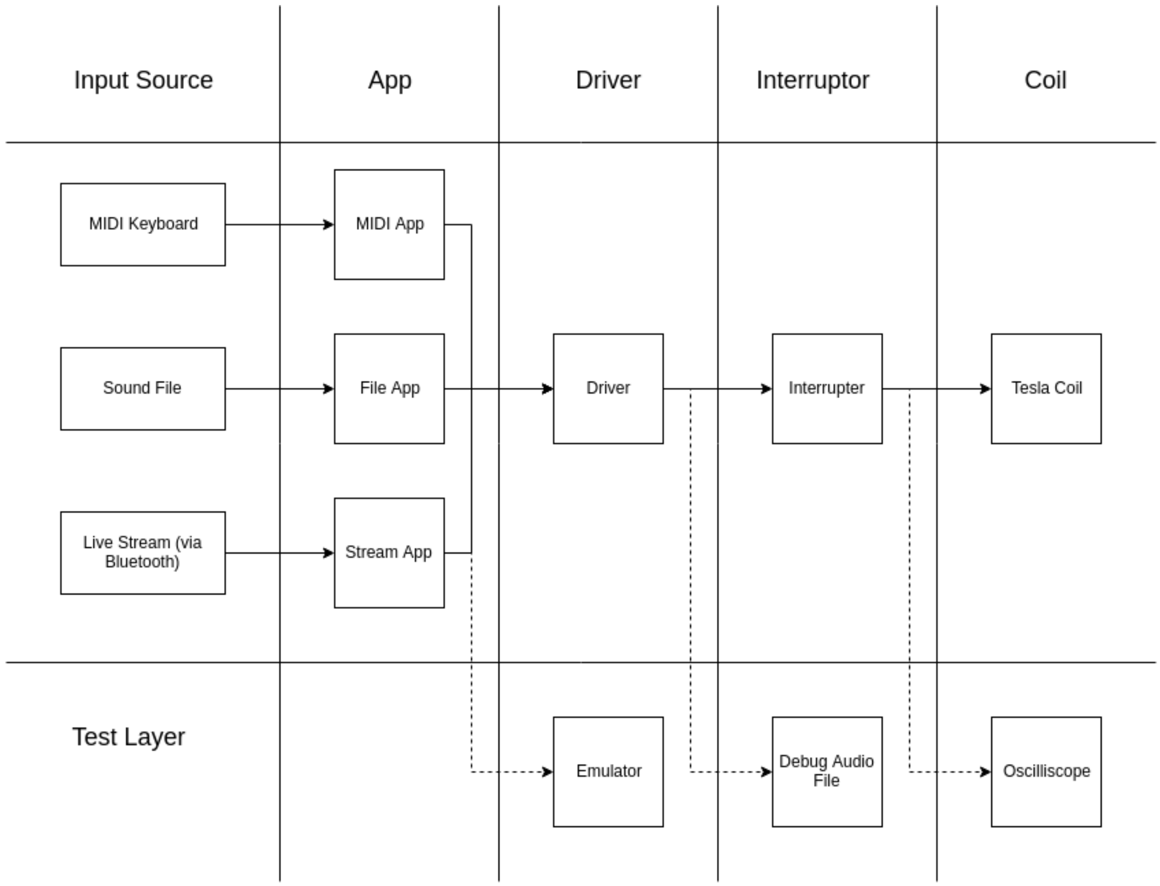

Figure 1: Overview of the Project Layout

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 lightning appears. For the high frequency switching to occur, the transistors are driven by a driver circuit. This driver circuit (not to be confused with the driver layer mentioned beforehand) will take in the output from the interrupter 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.2). Within this driver circuit is two other forms of input that come from current transformers that are connected to the primary circuit with the primary capacitor and inductor. One is for feedback that allows the input from the interrupter to be synced with the already switching circuit to prevent any out of phase waves. The other is used as overcurrent protection that will turn the coil off if the current becomes too great where components will start to break. 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.

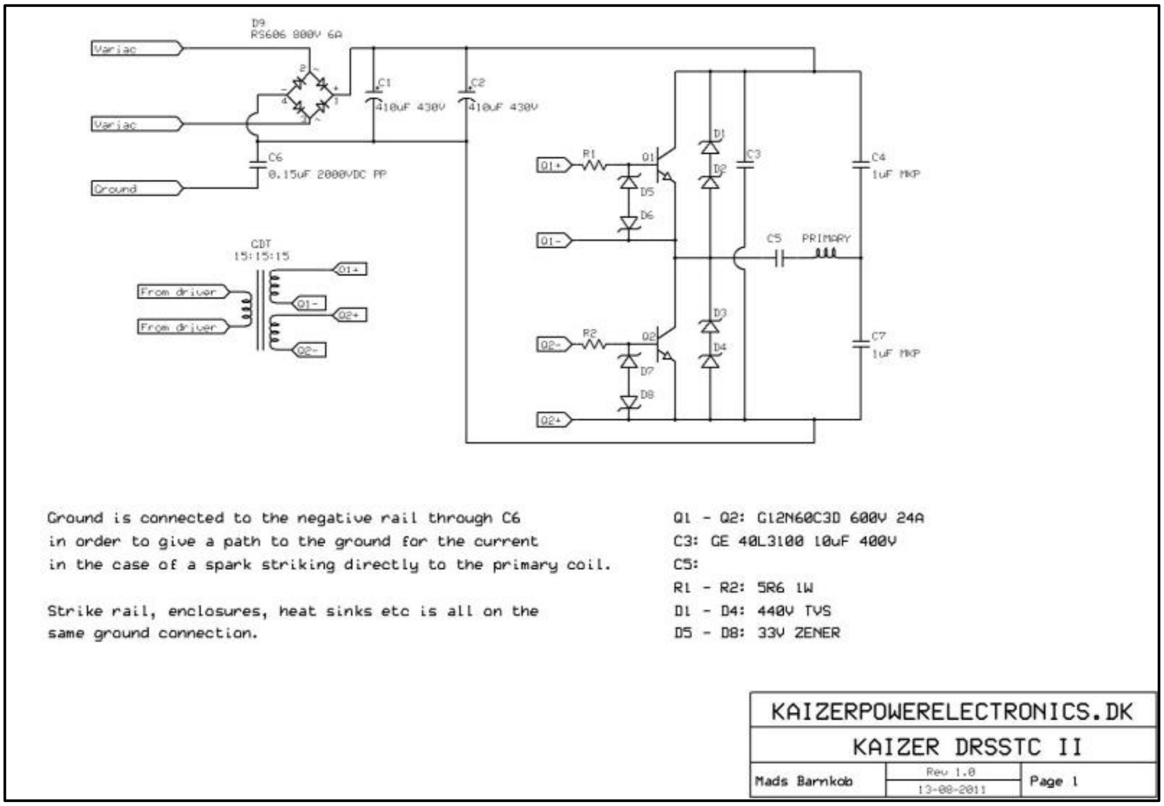

Figure 2: Bridge Circuit

2.2 Design Analysis

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 App Layer

Two main programs have been created for this layer. One to grab input from a MIDI file stored on the raspberry pi, and one to grab live input from a MIDI keyboard.

2.2.1.1 The MIDI File Reader

The MIDI file reader parses MIDI messages from a file and sends the relevant information through a socket. The MIDI file is interpreted as an object, which allows easy access to reading it. The only MIDI events that are processed are timing and note on/off events, since other effects will not be reproducible on the tesla coil. The messages sent in the socket carry channel and note info. Two channels are used, since the tesla coil should be able to play two notes at once. The timing of the notes comes from the order and timing of when the events are sent through the socket. The program was written in C++ because it needs to run quickly, and because there is native support for C++ on a raspberry pi.

2.2.1.2 THE MIDI KEYBOARD RECEIVER

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 to pick up these messages and provide an easy API to access them. These messages are processed and forwarded via a socket in the same format that the MIDI file reader creates. Again, two channels are supported. C++ was chosen for the same reasons as stated in the MIDI File reader program.

2.2.2 THE DRIVER LAYER

The driver is a program on the Raspberry Pi that reads off of the keyboard receiver socket. It converts these socket messages into square analog waves on GPIO pins. The socket messages are read and interpreted by the driver program, with one thread reading the socket and the other thread outputting the wave. The pins are interacted with using the WiringPi library in C. C was chosen as speed will be the biggest factor for the driver. Raspbian, the operating system for the Pi, is UNIX based so C integrates with it well. WiringPi turns the GPIO pins from digital high to digital low to create a square wave. These signals are sent to the interrupter where the circuit will handle them. Two channels are supported by simulating the effect of two notes being played at once.

2.2.3 THE INTERRUPTER LAYER

The interrupter is a circuit (Figure 3) that takes an input from the driver layer. We plan to start off by putting in the input through an op amp to ensure that our input signal is large enough for a proper response. The circuit can also take care of more modulation to ensure that any input to the circuit gets through as a square wave. An important part of the whole project is ensuring that the on-cycle of the output wave is not too long. If this section is not implemented in software, then we can include a 555 timer in this circuit to ensure that the pulse lengths are not too long.

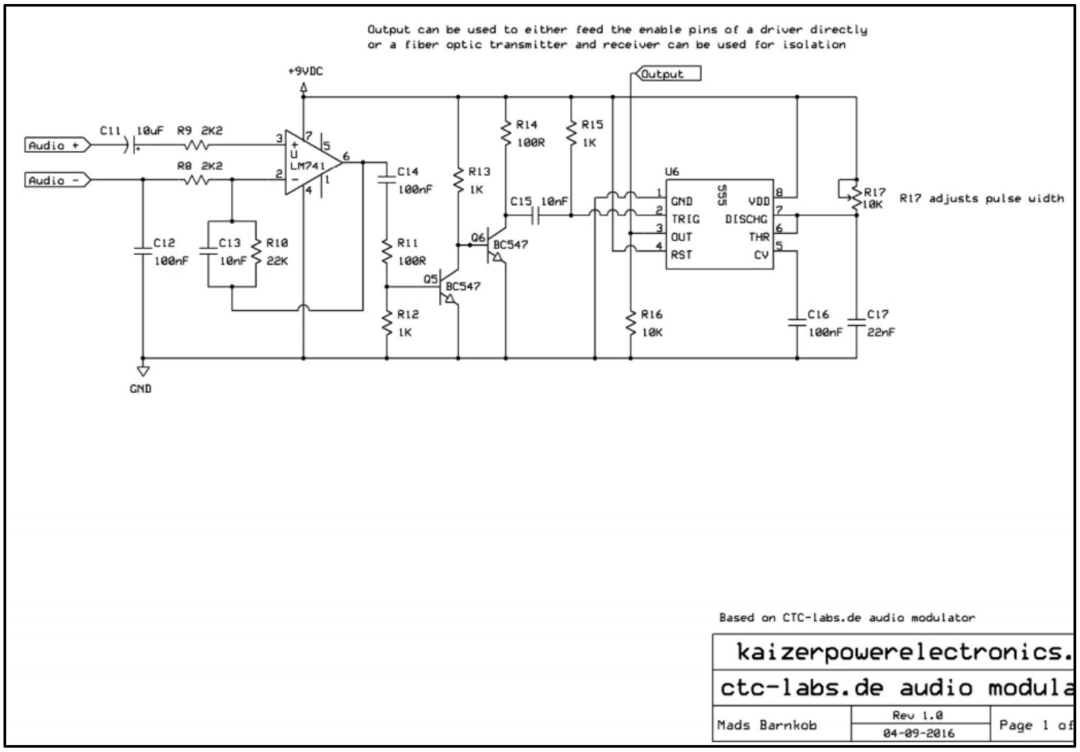

Figure 3: Interrupter Circuit

2.2.4 THE TESLA COIL LAYER

The tesla coil is the part of the circuit where everything else done comes to fruition. The tesla coil starts by taking in the input from the interrupter layer. From that point, it will sync it with the signal from the primary coil to ensure that no off-phase signals are being inputted to drive the bridge circuit. The synced signal drives the transistors in the bridge circuit which drives 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 lightning. The frequency of the lightning turning on and off is controlled by the interrupt layer and this creates audible sound when the frequency is within the range of human hearing.

3 Testing and Implementation

3.1 INTERFACE SPECIFICATIONS

One of the biggest challenges of this project will be creating the interface between the microcontroller software and the physical circuit that drives the coil. This will be implemented by the Driver Layer, which will use GPIO pins to output voltages to control the interrupter. The driver needs to quickly read from a socket and turn the data into an analog signal to be outputted. However, the WiringPi library makes this step much easier than manually changing register signals for the GPIO pin. It only requires calling functions on the pins to set them high or low. From this, the hardware just needs a physical connection to the correct GPIO pin to function properly.

The other major interface between hardware and software occurs on the other end of the project - connecting the MIDI keyboard to the microcontroller. The MIDI messages created by the keyboard need to be received in software. This is mostly accomplished using the ALSA and the MidiRt Library, which provide a simple API to record and analyze the MIDI messages in software. This interface has already been tested simply with a MIDI keyboard connected to a raspberry pi. MIDI events (i.e. notes being pressed) were recorded in the software and sent to the driver emulator (which is defined in more detail in the next section), which produced a wav file. This sound file accurately depicted what was played on the keyboard. This program will need to be tested more heavily with some unit tests, but the basic function is working.

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 are 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

To ensure that all software runs correctly, unit tests will be created that cover functionality. This includes the App and Driver Layers, which run on the microcontroller. If these unit tests discover bugs in the code, we can use this information to patch the code.

Both forms of input (keyboard and MIDI files) need to be verified. This can be tested in stages, since the layers have been modularized. First, the reception of the input can be tested with the driver emulator. The output voltages from the driver and interrupter can then be tested with an oscilloscope. Finally, when all the parts are ready, this input can be sent to the tesla coil itself. It will be easy to determine if the desired effect is produced here.

3.4 Non-Functional Testing

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.

3.5 Process

The general design of the project (especially in regard 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.

3.6 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.

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

MIDI Info Links:
https://www.csie.ntu.edu.tw/~r92092/ref/midi/midi_channel_mode.html
http://www.personal.kent.edu/~sbirch/Music_Production/MP-II/MIDI/midi_channel_voice_messages.htm
https://www.midi.org/specifications-old/item/table-1-summary-of-midi-message

RtMidi:
https://www.music.mcgill.ca/~gary/rtmidi/
Instructable Guide for Tesla Coil: https://www.instructables.com/id/Build-a-Musical-Tesla-Coil-like-a-Pro/

Setup Pi as WiFi access point:
https://learn.adafruit.com/setting-up-a-raspberry-pi-as-a-wifi-access-point/overview

Kaizer Power (Bridge Circuit):
http://kaizerpowerelectronics.dk/tesla-coils/kaizer-drsstc-ii/

Kaizer Power (Music Modder):
http://kaizerpowerelectronics.dk/tesla-coils/musical-sstcdrsstc-interrupter/

Steve's updated Driver Circuit:
http://www.stevehv.4hv.org/new_driver.html

4.3 Appendices

Further into the project, this will gain more attention.