

NESLA COIL

By

Julian Goldstein

Payton Baznik

Shane Zhao

Final Report for ECE 445, Senior Design, Spring 2018.

TA: Zipeng Wang

May 2nd, 2018

Project No. 32

Abstract

The NESLA Coil is a novel intersection between the Electrical Engineering sub-fields of power electronics and computer hardware systems. Principally, the NESLA Coil is a musical Tesla coil that serves as an audio output accessory for the Nintendo Entertainment System (NES). Our design exposes, a sound control interface that allows our coil to be driven directly by the NES' Audio Processing Unit (APU) in a manner that is plug and play. Using the APU's generated audio waveforms as an input, the NESLA Coil acoustically replicates the behavior of a standard television speaker by radiating arcs of lightning that vibrate the surrounding air and produce sound.

Contents

1. Introduction	1
1.1 Signal Processing Architecture	1
1.2 Block Diagram	2
2. Design	3
2.1 AC to DC Converter	3
2.1.1 Full Wave Rectifier	3
2.2 Pulse Width Modulator	4
2.2.1 Original PWM Design	4
2.2.2 Configuring the TL494	6
2.3 Audio Decoder	6
2.4 NES Software Block	7
2.4.1 Overview of the NES Emulator Software Architecture	7
2.4.3 GPIO Output Child Process	8
2.4.4 Integrating Functionality with the DAC	9
2.5 High Voltage Switching Circuit	10
2.5.1 Original Gate Driver Circuit	10
2.5.2 High Side-Low Side Gate Driver	11
2.5.3 IGBT Switches	12
2.6 Tesla Coil	13
2.6.1 Construction Overview	13
2.6.2 Operation Specifications and Behavior	13
3. Design Verification	15
3.1 AC to DC Converter	15
3.1.1 Full Wave Rectifier	15
3.2 Pulse Width Modulation	16
3.2.1 Original PWM Design	16
3.2.2 New PWM Circuit	17
3.3 Audio Decoder	17
3.4 High Voltage Switching Circuit	17

3.4.1 Old Switching Circuit	17
3.4.2 High Side-Low side gate Driver and Switching Circuit	17
4. Costs	19
4.1 Parts	19
4.2 Labor	20
5. Conclusion	20
5.1 Future work.....	20
References	21
Appendix A Requirement and Verification Table	22
Appendix B Additional Figures	28
Figure 1: DAC EAGLE Schematic.....	28
Figure 2: DAC TINA TI waveform.....	29
Figure 3: DAC PCB	29
Figure 4: Voltage Output of Full-Wave Rectifier Circuit with no Capacitor	30
Figure 5: Voltage Output of Full-Wave Rectifier Circuit with Capacitor (No-Load)	30
Figure 6: NES Emulator Flowchart	31
Figure 7: NES Ring Addition	31
Figure 8: Tesla Coil's Structural Component.....	32
Figure 9: Clean PWM Circuit	33
Figure 10: Old PWM Circuit.....	34
Figure 11: Triangle Wave Generator Output	34
Figure 12: PWM Output Using a 1 kHz Sine Wave as an Input	35
Figure 13: Summary of Our Coil's Mechanical Specifications.....	35
Appendix C Design Equations	36
Appendix D Coil Specification Tables	38
Table 1: Summary of Our Coil's Mechanical Specifications	38
Table 2: Summary of Our Coil's Passive Electrical Specifications	38
Appendix E Printed Circuit Board Layouts	39
Figure 1: Triangle Wave Generator PCB.....	39
Figure 2: Digital to Analog Circuit PCB	40
Figure 3: Error Amplifier PCB	41
Figure 4: Switching Circuit PCB	42

1. Introduction

The overall goal of the NESLA Coil is to build a musical Tesla coil that natively integrates as an audio peripheral accessory to the Nintendo Entertainment System (NES). At its core, the NESLA Coil is functionally congruent to a single-input, single-output signal processing system.

At our system's input end, the input signal is a stream of digital audio samples which are provided by the NES Audio Processing Unit's (APU) output interface. The system transforms the input signal into an audio format which can be used to modulate the electrical discharge frequency of the musical Tesla coil. The system's output is realized through the observation of plasma-induced sound which resembles the waveform intended by the APU.

The NESLA Coil's overall design, must effectively utilize various electrical engineering topics and principles in order to achieve its functional goal. These principles include electromagnetic theory, control systems, digital-analog conversion, power electronics, parallel programming, and computer engineering.

1.1 Signal Processing Architecture

For the NESLA Coil to accurately produce the sounds corresponding to the musical scores of NES games in real-time, the NES APU's waveform that is being input into our system must undergo a sequence of signal processing steps. The signal processing architecture can be abstracted as a linear cascade of simpler sub-systems. Each sub-system can be abstracted as a functional block in the NESLA Coil's signal processing path. These blocks are divided such that their purpose and contribution to the overall system is singular. In the context of our design, singularity means each module is responsible for carrying out one and only one type of signal transformation between its input and output. By making each module responsible for applying one and only one transformation, we can understand the NESLA Coil's complex signal transformation process as a cascaded composition of simpler processes. Figure 1.1 illustrates the linear cascade of functional blocks the APU's audio waveform must traverse through to become sound.

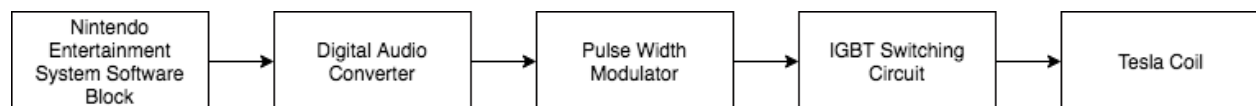
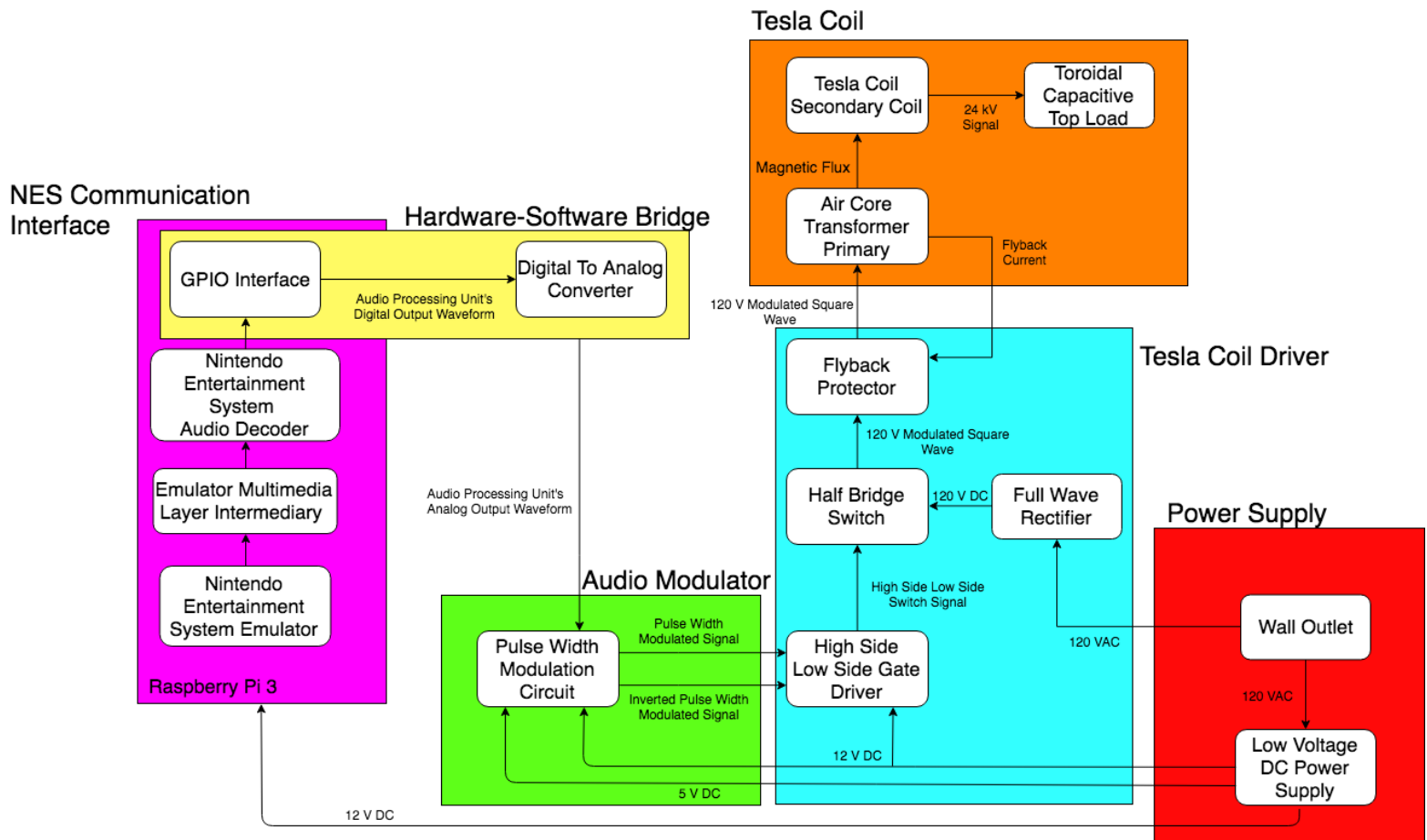


Figure 1.1: Signal Processing Cascade Architecture

1.2 Block Diagram



2. Design

2.1 AC to DC Converter

2.1.1 Full Wave Rectifier

The full wave rectifier is a simple circuit that uses 4 diodes in order to take an AC signal and obtain the absolute value of that signal as the output. It does this by having a very specific orientation of the diodes, so only two of them are on at one time. This configuration makes the voltage at the output always positive. After we obtain the absolute value of this AC signal we then put it through a capacitor in parallel with the load in order to smooth out our rectified AC signal so it resembles a DC voltage. Our design is shown in figure 2.1.1.

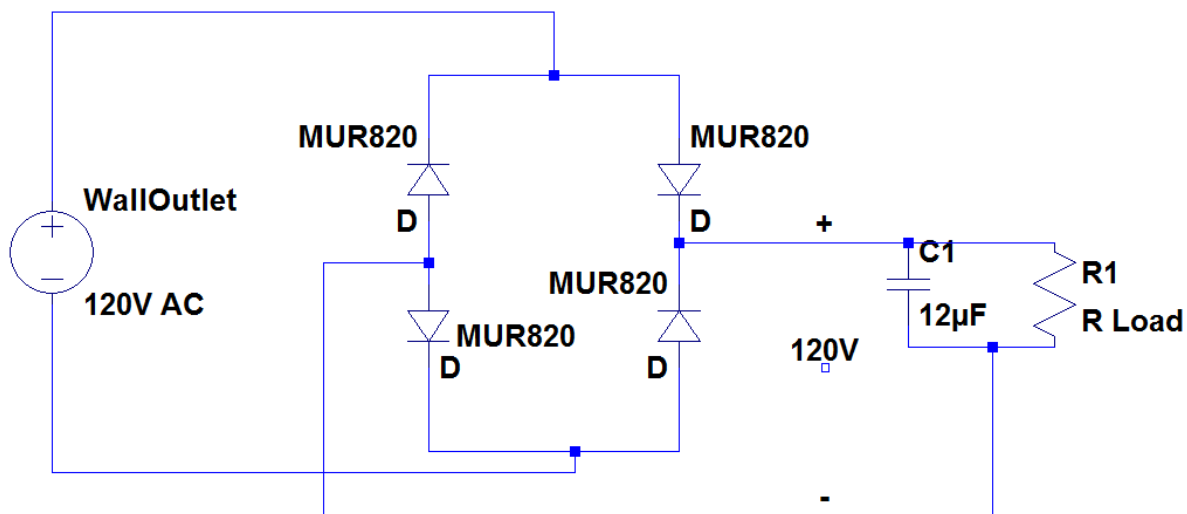


Figure 2.1.1: Full Wave Rectifier Design

The purpose of the full wave rectifier circuit is to create the high voltage input that we use for our coil. Because we designed our coil to be a 1:200 voltage relation between the primary and the secondary, we should be able to hit 24 kV on the secondary if we use the 120V DC from the output of the rectifier circuit. We originally decided to use a boost converter and convert the 12V rail from our power supply to a higher voltage. The problem that we ran into with this design is that we couldn't find a boost circuit we could purchase that would be able to handle the amperage requirement of our coil. Because of this we needed to find a solution that would be able to handle the power output of our coil. The best option is to use wall voltage because we will be able to get at least 1200W from the wall, which is more than enough to supply 120 V to the primary of our coil.

2.2 Pulse Width Modulator

2.2.1 Original PWM Design

Our PWM design has changed throughout the design process of our coil. Our original design made use of op-amps and comparators in order to generate a triangle wave at our coil's resonant frequency and compare it to our analog audio signal. The design can be divided into two different sections, the triangle wave generator and the comparator. Figure 2.2.1 was the original design for the triangle wave.

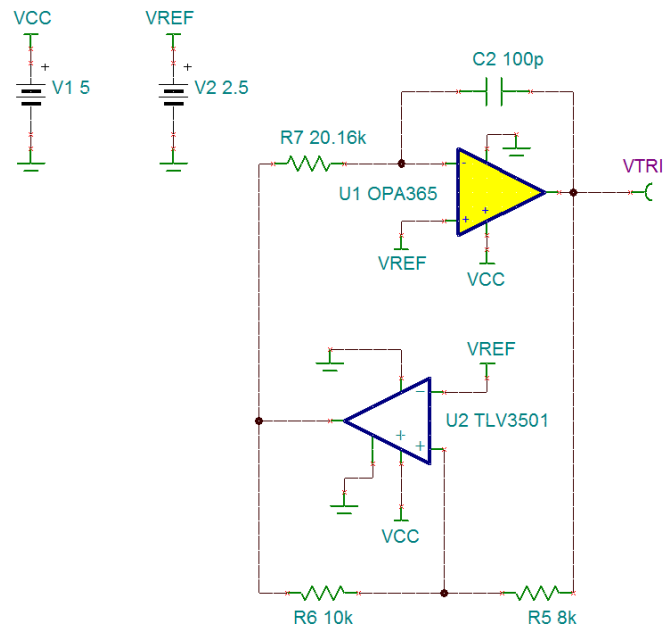


Figure 2.2.1 Triangle Wave Generator

The purpose of this design is to generate a triangle wave by configuring our op-amp as an integrator and forcing our comparator to switch between high and low voltage at the frequency of our coil's resonant frequency. Because of the design of this circuit, Equation (15) shows how to calculate the frequency of the triangle wave. We wanted our design to operate at 155 kHz because this was the resonant frequency of our coil. We also designed our triangle wave so that the peak to peak voltage would be higher than the analog voltage peak to peak. We needed this to be the case because if their voltages equaled each other then we would run into cases with our PWM output where it would be approaching 100% or 5% duty cycle. When this occurs, we have the possibility of our switches not being able to switch fast enough to see that high valued duty cycle. This would lead to an open or short circuit, with the short circuit being the worst possible outcome. With this in mind we used Equation (16) and set the voltage peak to peak to be 2 V. Because of this we needed to make sure that our signal input into the PWM will not have a peak to peak voltage greater than 1.9 V in order to avoid any complications with the switching circuit.

The next part of the design was the comparator circuit with the error amplifier. The very basic idea of this circuit is we have a comparator that compares both our triangle wave and the analog audio signal. Based on this it will output a high or low voltage and thus our PWM signal output. In reality the design is a bit more complicated. The design we used is shown in figure 2.2.2.

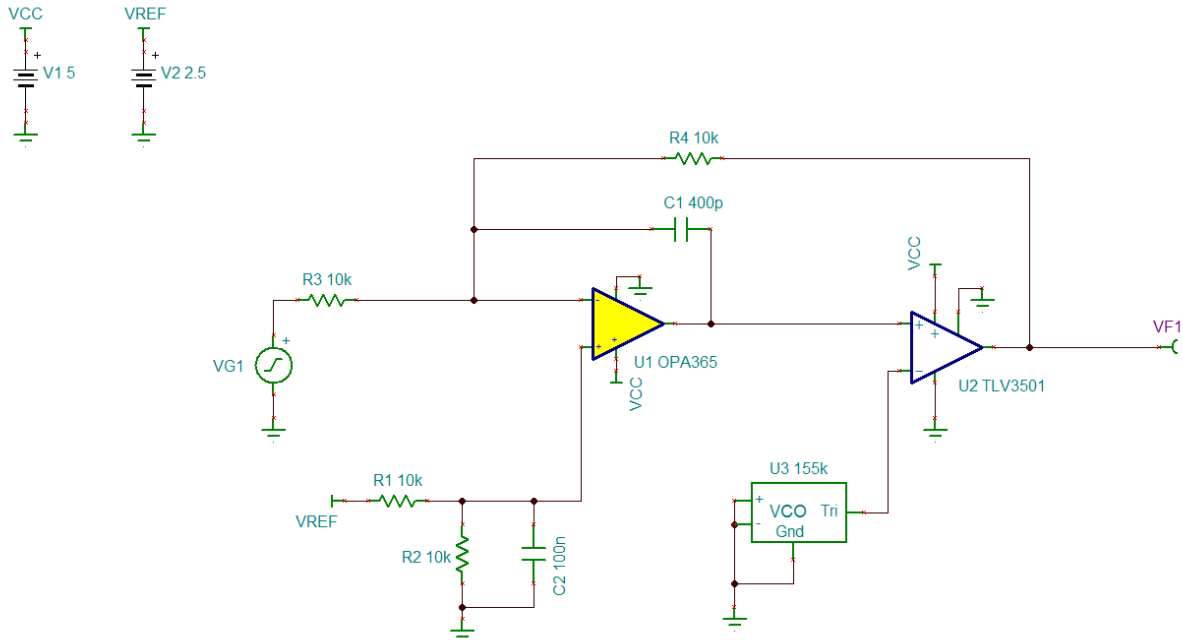
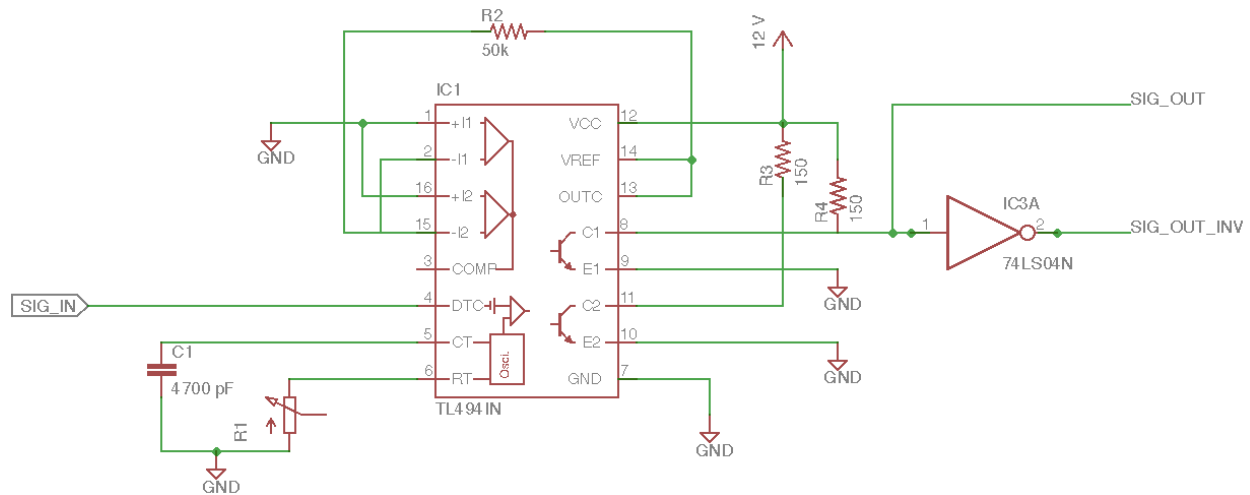


Figure 2.2.2 Comparator with Error Amplifier

This design works in a rather simple way. Our comparator is the TLV3501 and takes an input from the triangle wave generator shown in figure 2.2.1 as VCO and an input from our error amplifier which is the output of our OPA365. The output of this specific circuit is shown in equation 17. Because we wanted to obtain a gain of -1 for our output we set our R4 and R3 to be the same value. The next design that we needed to worry about was the pole we would create in the frequency response. The calculation of this pole is shown in equation 18. We want a pole that is below our switching frequency of 155 kHz and larger than any audio signal we would see, which is around 20 kHz. We set our capacitor to 400p in order to get our pole in-between these two values.

Because we want to use a comparator and we currently have our triangle wave offset by 2.5 volts we also need to find a way to offset our analog signal by 2.5 volts as well. This is where the op-amp in our error amplifier circuit comes into play. We can use Equation (19) in order to determine the voltage offset we would gain for our analog signal. Because we want the voltage offset to be 2.5 just like our triangle wave we use 10k resistors for all the passives. With this in place we were able to achieve the offset of 2.5 V, completing our design.

2.2.2 Configuring the TL494



To implement our new and improved pulse width modulation circuit, we used a dedicated PWM IC from Texas Instruments called the TL494. Since our application for the PWM circuit is push-pull, not single-ended, to compute the frequency of the oscillator we use Equation (9). Since there is inherent uncertainty on the exact resonant frequency we have to use to drive our coil, but we know within an order of magnitude where it should be, we decided to use a potentiometer instead of a resistor to set the oscillator's frequency. Fixing the capacitance at 4700 pF, we used a potentiometer that offered resistance ranges between 0 and 10 k . This way we could, just adjust the frequency we used to drive the coil with a twist of the potentiometer's knob as opposed to soldering on and off substituted resistors.

To successfully drive our high-side, low-side gate driver in the next module, we had to provide it with the pulse width modulated output, SIG_OUT, produced by pin 8, as well as the inverted complement provided by SIG_OUT_INV. To create this inverted complementary output, we used the Texas Instrument's 7404 Hex Inverter. The 7404 successfully carried out its inversion function to drive the next stage, however, the 7404's pitfall is that it cannot handle 12 V as its power supply. Instead, if we want to use it, we have to give it a separate 5 V supply. This extra need for a separate power supply, increases the cost of producing this circuit, and makes it more complex than it has to be.

2.3 Audio Decoder

The audio decoder module functions effectively as a digital to audio converter (DAC). This module takes in as input a 16-bit digital signal from the GPIO module. It will output an analog signal to the PWM module. Because the peak of audible sound is at 20 kHz, the DAC must be able to operate correctly at least 20 kHz.

Figure 1 in the Appendix B shows the DAC schematic in Eagle. Most of the pins are outside connections allowing for input and output. Figure 2 in the Appendix B shows the simulated 16-bit input signals and the corresponding analog signal in the Tina TI simulation software. The input signals are switching at 1MHz and the output is capable of handling the switching speed. This is much faster than required for the correct operation of the DAC.

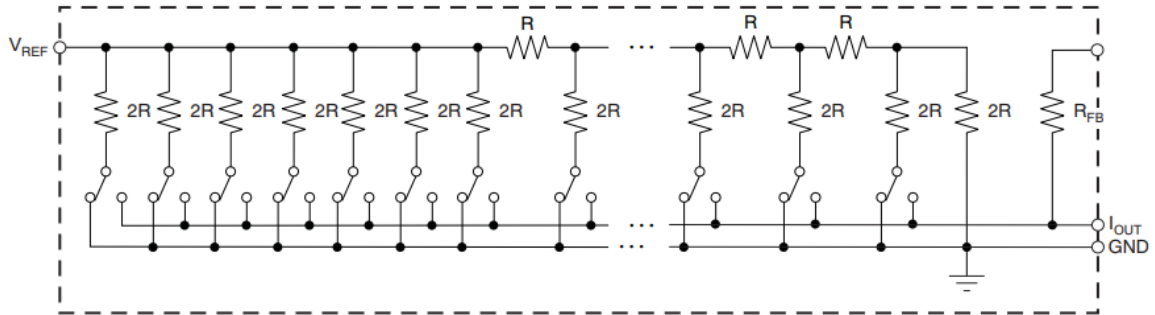


Figure 2.1 Equivalent DAC circuit [4]

Figure 2.1 shows the equivalent DAC circuit operation. The digital inputs determine whether their own switch is connected to I_{OUT} or GND. Equation (1) represents the expected output of the DAC and the equivalent circuit shown in figure 2.1. D in the equation corresponds to the number representation of the 16-bit input.

2.4 NES Software Block

2.4.1 Overview of the NES Emulator Software Architecture

After examining open-source implementations of NES emulators, made publically available through GitHub repositories, such as LaiNES[1], nes-emu[2], and SimpleNES[3]. I observed that across the various implementations, the source code is organized to reflect the underlying hardware architecture and organization of the real NES. The rationale for this pattern is rooted within the design requirement of the emulator itself, which is to emulate the NES accurately. In order to satisfy the emulation-accuracy requirement, the underlying hardware viewed from the perspective of the game must be identical to the NES. Thus, by virtualizing the NES hardware as software modules, the game's runtime dependencies become decoupled from the actual machine hardware it is executing on. The organizational pattern is illustrated in the figure 6 in the Appendix B.

In essence, what we would like to modify in this common organizational pattern is how information routes from the Game Audio Processing Modules into the Audio Output Peripheral. The solution should be based off of this emulator design structure. If we can assume an underlying structure, the task of modification would be more generalizable. In essence, if the design structure is known ahead of time, all we have to do to modify an emulator is find the code's integration point between the Game Audio Processing Modules into the Audio Output Peripheral and intercept the integration between the two

parts with a single function call to the modification code's entry point. Internally, our modification code should know what to do provided the outer emulator's structure is along the lines of this pattern.

2.4.3 GPIO Output Child Process

The biggest challenge in implementing the GPIO Audio Output was satisfying the criteria that individual 16-bit audio samples have to be placed on the GPIO peripheral pins at a rate very close to 44.1 kHz. If the GPIO output rate swivels over 20% off of the 44.1 kHz ideal, the analog signal seen at the output of the DAC aliases the sound signal beyond recognition. The reason why the GPIO output rate swivels and does so randomly is because the underlying Linux environment is configured to interrupt the process at random time intervals, so that it can use the CPU to react to more critical system events like hardware interrupts, checking resource availabilities, and page swapping. The probabilistic wait time for the GPIO child process is therefore characterized according to an Erlang distribution [13]. If we have k processes that have greater than or equal scheduling priority, and a process rescheduling rate of λ , we use the Erlang distribution to construct a probability density function that characterizes the random wait time, T_k of the GPIO child processes. The expectation of wait time for the process positioned at index k is given by Equation (6).

This equation shows, if we can raise the priority of the GPIO process to be above all other user-land processes we can reduce the expected wait time. To execute this step, immediately after the child's fork point away from the emulator parent process, we ask Linux to give the newly-born child its process ID, PID. Once the child is aware of its PID it can identify itself when re-configuring the Linux scheduler for the high scheduling priority wait time reduction. The mechanism used to make this happen is the child simply asking the bash shell to run the command: "sudo chrt -f -p 99 [PID]" where [PID] represents the child's obtained PID. What this command does is it invokes system administrator privileges to modify the provided PID's scheduling policy, so that it gets treated by the scheduler as a real-time process instead of the default class designation as an interactive process. Once the process classification is changed, we supply the "-f" flag to change the scheduling policy from the round robin default SCHED_RR to the alternative first in first out option, flagged internally as SCHED_FIFO. Directly after we change the class and scheduling policy, we maximize the priority using [14].

Thus, the expected GPIO 16-bit sample's time spacing, T should be the time cost of cycling through one iteration of the sample output loop, T_c , the sleep time in between iterations, T_s , and the time spent in the scheduling queue at position k , T_k . Ideally, we would like the mean of T to match the time period between 44.1 kHz pulses. This time period is computed by evaluating the reciprocal of frequency and yields 22.67 μs .

Overclocking the Raspberry Pi's ARMv7 CPU cores from 750 MHz to 950 MHz, we measured T_c to be approximately 11 μs . The uninterrupted loop cycle time, T_s I measured to be 6 μs . Changing the scheduling context to $k = 1$ with the real-time rescheduling rate, λ . We found the expected wait time for the process to be 1 μs . In total, we found the expected time spacing of the 16-bit samples being placed on the GPIO pins to be 18 μs . This is actually just below the maximum 22.67 μs . To lengthen it we inject

a busy wait loop just before each iteration moves to the next. We tune the total iterations we must cycle through to satisfy the busy wait and find that an iteration total of 3,630,000,000 was sufficient enough to add the extra 4.67 μ s needed between each sample output.

In our case of $k = 1$, the Erlang distribution used to define the probability of T has a standard deviation of λ^{-1} . Using the Central Limit Theorem, we are able to see that using the Erlang random variable to construct the following random variable, $X(T)$ [13]. We use the random variable standardization technique in equation 7.

This new construction, morphs the Erlang probability density function of domain T into the standard Gaussian form over the domain of X . The mean and standard deviation of the new Gaussian variable X matches that of the original random variable T . Using the Standard Gaussian's cumulative density function, we see that if we can set λ^{-1} such that $T \in (E[T]-5\lambda^{-1}, E[T]+5\lambda^{-1})$ is always within 20% of 22.67 μ s, we are almost surely guaranteed that T will lie within this interval 99.999942% of the time. Therefore, the constraint on λ^{-1} is given by equation 13.

Solving for λ^{-1} and using $E[T] = 22.67$: We see the absolute maximum of $\lambda^{-1} = 0.9068$ schedules/second. Using the `sched_yield` system call, which is what a user-land process uses to tell Linux it wants to be re-scheduled, I was able to measure the rescheduling rate by doing the real-time rescheduling procedure, and then simply reading the ARMv7's internal Tick register just before and just after the `sched_yield` call. Computing the absolute value of the difference between the Tick register read just before and just after `sched_yield`, I was able to then divide this by the running CPU frequency of 950 MHz and see that λ^{-1} is consistently on the order of 10^{-9} schedules / second. This result is over 8 orders of magnitude lower than our established maximum. Therefore, the CPU is guaranteed fast enough to keep up with the GPIO output rate, the challenge, however, completely lies within configuring Linux to stay out of the way, and not interrupt the GPIO child unless it absolutely has to.

2.4.4 Integrating Functionality with the DAC

Due to the disagreement between the unsigned 16-bit inputs expected by the DAC, and the rendered signed 16-bit samples from output audio waveform. In order for the DAC to properly be driven using our Pi over the GPIO, we had to modify the samples to stop the signed/unsigned disagreement from producing an integration failure between the two components. Luckily, the set of 16-bit binary integers in conjunction with the operation of modular 16-bit addition forms an algebraic structure known as a cyclic group. The DAC, essentially functions as a bijective map between the set of 16-bit binary integers and the interval of voltages between 0 V – 5 V. The map between the two cyclic groups, half rotates around the entire cyclic group if we add an offset equal to half the ring size. In our case, the ring size is 2^{16} . Half the ring size, the half rotation offset is 2^{15} . In hexadecimal notation, $2^{15} = 0x8000$. Therefore, applying ring addition to the cyclic group, of exactly $0x8000$ creates an algebraic group action and all members of the set 16-bit digital set rotate clockwise by $\frac{1}{2}$ a revolution while the mapped members of the analog set, 0 V to 5 V remain stationary. The result of the $0x8000$ offset's half rotation re-center's digital zero, $0x0000$, to map to 2.5 V. Without the offset's group action, $0x0000$, would have mapped to

0 V, and digital -1, 0xFFFF would have mapped to 5 V. A photo summarizing the overall group action of algebraic ring addition is shown in figure 7 in Appendix B.

2.5 High Voltage Switching Circuit

2.5.1 Original Gate Driver Circuit

Our original design for our driving circuit had us using one UCC37322 and one UC37721. These were both rated for 9A of output current, which was more than enough to handle the current our gates would be needing for their switching. They were both low side drivers, their only difference being that one was inverted so we could have inverted gate signals on our two IGBTs. The reasoning behind using gate drivers is a normal PWM circuit does not have the ability to provide enough current to drive the circuit. The PWM circuitry would die if it was directly attached to the gate itself. So instead we place a gate driver in-between the PWM signal and the gate. The PWM signal is taken into the gate driver, then the gate driver outputs it to a higher voltage and allows the gate to pull up to 9 A of current out of it without distorting the signal. Our original design is shown in figure 2.5.1.

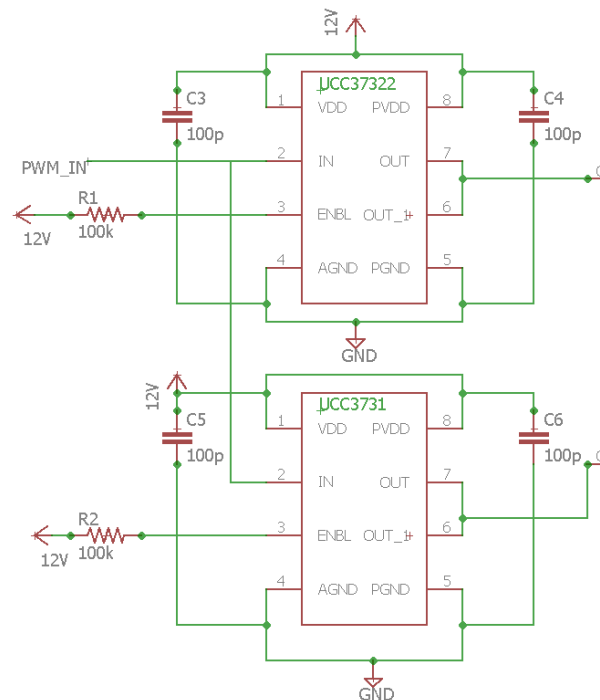


Figure 2.5.1 Old Gate Driver Circuit

In this diagram, we use a series of 100pF capacitors. These are placed in order to reduce the ground noise in our circuit, so we can have clean gate signals being supplied to our gates. We tie our enable pins to our 12V rails so that our drivers are always operating. We also need to tie our power voltage pin and our low voltage pin together, so they will have the same reference and be able to interact with each other.

2.5.2 High Side-Low Side Gate Driver

With the realization that we required a high side and a low side driver we needed to find a gate driver that could fit our needs. We decided to use the UCC27714. This is a high and low side driver that is able to operate at up to 600 Volts and supply up to 4 Amps. The main difference between this driver and the previous driver is the high side portion of the circuit utilizes a bootstrap circuit. The circuit design is shown in figure 2.5.2.

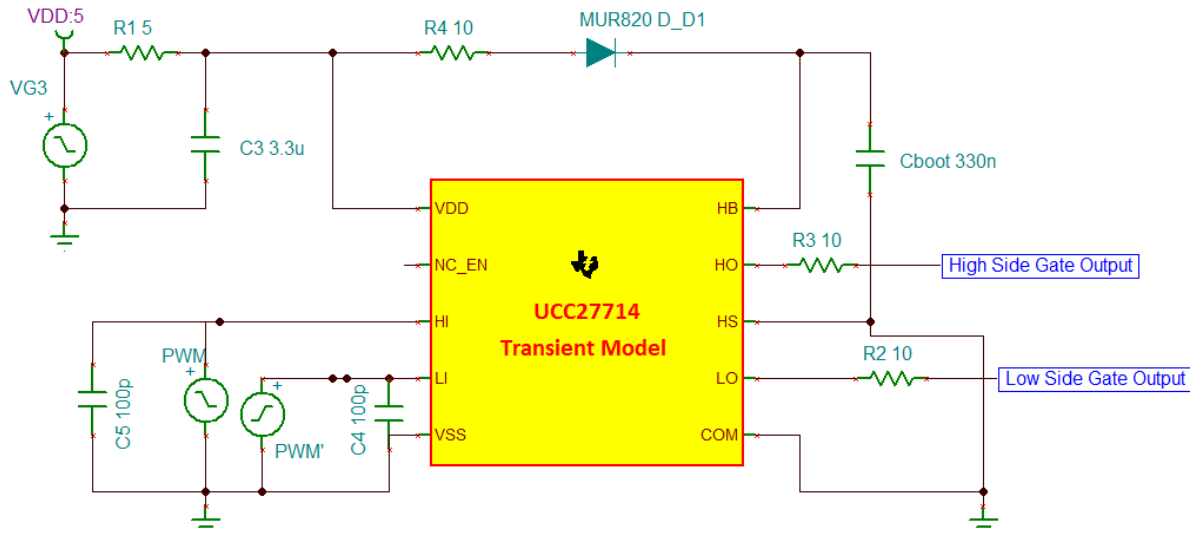


Figure 2.5.2 Revised Gate Driver Model

There are a few things that we needed to make sure we used when designing this circuit. The most important part is designing the bootstrap capacitor. By reading through the guide provided by the manufacturer we needed to pick a boot capacitance that was around 10 times the value of the capacitance of our gate. We calculated our gate capacitance using equation 20. We got a value of 27 nF for our gate capacitance which means our bootstrap capacitor should be around 270 nF. With the capacitance calculated we now needed to make sure that our time constant for our RC in the bootstrap was low enough to charge and discharge in time for the gate driver. For this specific gate driver, we needed to add a resistor to our bootstrap capacitor in order to lower our slew rate. The recommended value for such a resistor is between 2-10 Ω, and because we have such a large capacitance we opted to go with the 10 Ω resistor to slow our slew rate even more.

Another passive that we needed to design was the capacitance that was tied to our VDD input and our ground. This capacitance was used to help reduce the ground noise in our circuit. It was also recommended by the UCC27714 manual to make the capacitance value 10 times larger than the boot capacitor that we used. Because of that we picked a 3.3 μF capacitor to do the job. The capacitors that we used on the inputs of the PWM signals are simply there to clean out our input signals. The resistors

that we added to our gates are selected to reduce the ringing in our gate voltage. We used a 10 Ω resistor for both of these gates in order to dampen out this ringing to best of our ability.

2.5.3 IGBT Switches

The IGBT switching circuit is a simple half bridge circuit. It utilizes 2 IGBTs, one tied to a high voltage of around 120 VDC and the other IGBT tied to ground. The primary positive input of the coil is connected to the collector of the high side switch and the emitter of the low side switch while the negative is connected to two capacitors, one tied to high voltage and the other tied to ground. The diagram we used is shown in figure 2.5.3.

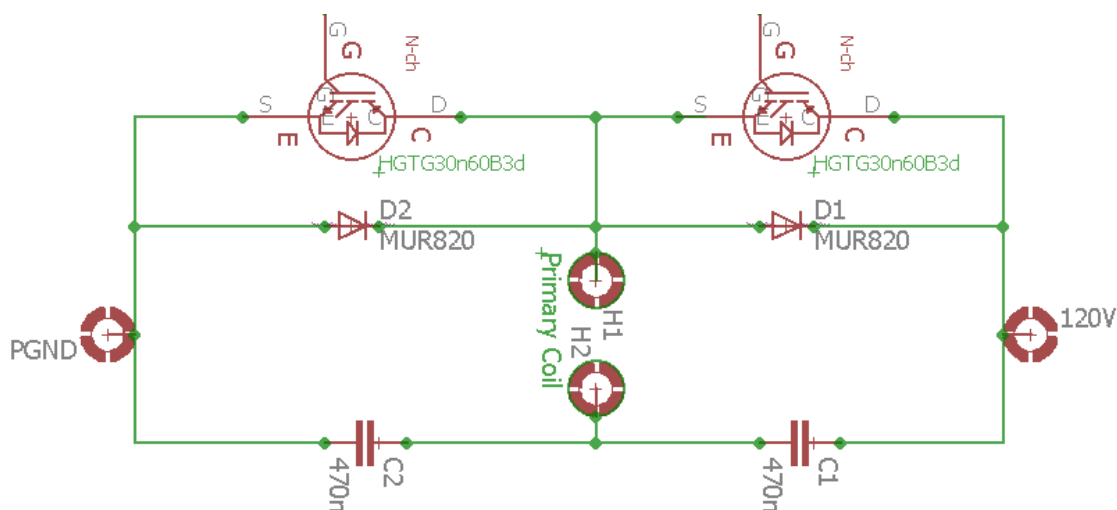


Figure 2.5.3 Switching Circuit

This switching circuit utilizes a half bridge design. When the high side switch turns on the primary of the coil sees 120 V. When the low side switch turns on the coil primary is now tied to the ground. These switches work as inverses of each other so they will never turn on or off at the same time. The IGBTs are rated for 30A of current and 600 V. This allows us to switch as the voltage we require and never have to worry about the amount of current going through the IGBTs because it will never approach 30 A in standard operation.

We added the high-power diodes as fly-back protection for our circuit. When the transformer has fly-back current our diodes that we add will be able to guide the current back and make sure that the current never flows into a part of the circuit that we don't want it to. These diodes are rated for 200 V, so they will never break down due to the voltage returning from the coil.

We also added two large capacitors tied to the negative of the primary of our coil. The purpose of these capacitors is to help isolate the primary from the rest of the circuit. They also make sure that the primary of the coil sees a clean input from our switching circuit.

2.6 Tesla Coil

2.6.1 Construction Overview

By design, the Tesla Coil is an inductively coupled resonant air-core transformer that provides a high voltage gain between its primary and secondary sides. The Tesla Coil is constructed by utilizing various common house-hold PVC and plastic pipes and fittings with conductive media wrapped around their surface areas. The reason we chose to make the Tesla Coil using only household pipes and fittings is because the fittings allow various pipe shapes to be attached and detached at any given time. Therefore, using pipes and fittings to create the Tesla Coil allows it to be taken apart into various modular components and naturally pieced back together as a monolithic device. This coil construction philosophy is advantageous over alternative construction philosophies as it favors the coil's portability and maximizes the coil's reliability by allowing malfunctioning structural pieces to be replaced without replacing the entire coil. Figure 4 in Appendix B shows a photograph of the final product and its various subcomponents.

2.6.2 Operation Specifications and Behavior

Using the mechanical measurements listed in Table 1 appendix D in conjunction with Equations (2-5), we were able to calculate the capacitance of our toroidal top load in addition to the primary and secondary coil inductances. We also used the RCL meter available in the senior design lab to physically measure the corresponding calculated results. In addition, we physically measured the resistance of the coil. These results help us determine the coil's passive behavior in the context of our coil driving circuits.

The coil produces audible electric arcs approximately 6 inches in length from the top needle. Using the measurement procedure outlined in Appendix B figure 14, we were able to make measurements on how the coil interacted with its surrounding electromagnetic field. We applied a constant amplitude, sinusoidal voltage signal to the coil's primary side, and monitoring the sinusoidal amplitude of the induced electromotive force by the secondary side. After sweeping through sinusoidal input frequencies between 100 kHz to 500 kHz, we found that the maximum amplitude of the secondary's induced electromotive force occurs at 155 kHz.

Applying Equation (2), we calculated the resonant frequency of the secondary side and neglected the effect of our air core transformer. In other words, we employed the case of zero magnetic coupling between the primary and secondary coils ($k = 0$). This yielded a resonant frequency of 170.5 kHz.

Looking back at Equation (2):

$$f_{res} = \frac{1}{2\pi\sqrt{L_s C(1 - k^2)}} \quad (2)$$

Applying algebraic manipulations and solve for k, the coupling coefficient of the air core transformer, we then conclude that:

$$k = \sqrt{\frac{1}{L_S C (2\pi f_{res})^2} - 1} \quad (2.1)$$

Using our measured resonant frequency of 155 kHz we use equation 2.1, to compute the coupling coefficient, k. This method, yields a coupling coefficient of approximately 0.46 for the air core transformer. The overall function of the air core transformer is to create the approximate 2100 V:10 V voltage gain between the primary and secondary. This expectation of gain is computed by evaluating the theoretical voltage gain of an ideal transformer in equation 14. Since realistically, we cannot expect the realistic case to occur, we adjust our voltage gain goal to be at a minimum 20% below the ideal gain of 2100 V: 10 V. Achieving this gain is limited by whether or not we are able transmit a sufficient magnitude of oscillating magnetic flux density between our primary and secondary coils. This limiting condition is a consequence of Maxwell's equations, Equation (23-24) which govern the electromagnetic field's interaction between an electromagnetic field's circulation and the corresponding field penetration.

The upper bound on the magnitude of magnetic flux density being transmitted completely depends on the relative magnetic permeability of the transmission medium. Exceeding this upper bound is called transformer saturation. Materials that have a high relative magnetic permeability reach their flux density limit at lower magnetic flux intensities.

By selecting a medium that offers a relative permeability closest to free space, which in our case is air, we can minimize the ratio of magnetic flux density per intensity. Minimizing this ratio raises the saturation limit at the cost of transmission efficiency. So, using air as the core is the most cost-effective way to stop saturation. However, since air is not an easily magnetizable material, we waste lots of energy attempting to magnetize it. The relationship between magnetic flux density and intensity is given in equation 20 and linearized in terms of relative permeability in equation 21.

Our coil requires a needle attached to the toroidal top load to produce the needed discharge because to achieve the effect of dielectric breakdown within the surrounding air. At some point on the surface of our capacitive toroidal top load, we need to emanate an electric flux with a density greater than or equal to 3 MV/m [7]. The needle's head achieves this flux density criterion without changing the capacitance of the toroid it is connected to. Since our needle is made of aluminum it is highly conductive. For conductive media, the effect of polarization is non-existent. Therefore, all electric flux lines will radiate perpendicular to the charged conductive material at each point on it. Applying Poisson's Equation using the Dirac delta function as the boundary condition for the free charge density at the needle's pin head. We see that the Laplacian of the voltage also resembles the Dirac delta function. Inverting the voltage's Laplacian under the point impulse boundary condition can be done by computing the second antiderivative of a 3D radial Gaussian distribution under the limiting condition

where its standard deviation drops to zero. Assuming spherical symmetry, if charge Q is deposited on the needle the equation is given by equation 13. Solving for V and computing E from equation 25 we derive equation 11:

The 3D radial Gaussian's small standard deviation in this mathematical model is correlated with the small area distributing the high voltage capacitor's radiative electric flux at the needles pin head. To get the electric field we compute the negative gradient of the radial Gaussian's second antiderivative, in turn we produce another related radial Gaussian. Resolving the electric field's limit as its Gaussian profile's standard deviation diminishes, we verify that at the needle's point the electric field's behavior exhibits impulse behavior. Therefore, the impulse approximation works if we choose a needle area sufficiently small enough at the tip to highly resemble the ideal impulse behavior. A 90/14 sewing needle was the best option available, since it implemented a sufficiently small point area to cause the electric field to resemble the impulse approximation.

The coupling coefficient is a good indicator of the coil's overall energy transfer efficiency and frequency selectivity. In the context of a Tesla Coil, researchers have found the ideal value for the coupling coefficient is 0.6. [5]. A coupling coefficient of 0.6 is ideal because it is low enough to keep saturation from becoming a problem, but also high enough to maximize the transmission efficiency. The fact that our magnetic coupling coefficient is only 23.3% below the ideal value means our Tesla coil's flux transmission energy loss is just above its ideal minimum. In other words, the PVC pipe fitting construction we engineered ranks relatively high in terms of efficiency when compared to other Tesla Coil constructions.

In appendix D table 2, we provide a table summarizing our physically measured results and compare them side by side with their corresponding calculated results. We also calculate the percentage of error between them.

3. Design Verification

3.1 AC to DC Converter

3.1.1 Full Wave Rectifier

Testing with the full wave rectifier was dangerous, so we needed to make sure that we took all the precautions necessary in order to keep everyone safe. The first test that we did was an open circuit test where we used a 20Vpp sine wave operating at 60Hz to represent a low voltage wall input. We discovered figure 4 in appendix B with no capacitors.

The voltage of the sine was successfully rectified, but the signal does not look DC-like. This is the exact reason we need to add a capacitor in parallel with our load. This capacitor will provide the voltage to

keep the voltage constant when the rectified sine wave starts to drop off after each maximum. Once we added the capacitors into our circuit we saw an output like appendix B figure 6.

This waveform looks much more DC-like when compared to the original waveform. This is merely a test at 20 volts with no load, so it would make sense that our voltage would remain constant. After we performed this test we went ahead and started testing using the test in the requirements and verification section. We were able to pass this test, and we also had a thermal camera looking at the diodes as they were switching. They got up to 90 degrees Celsius, which is within their related temperature of operation. With these tests completed, we determined that our circuit was working as intended.

3.2 Pulse Width Modulation

3.2.1 Original PWM Design

We designed our circuit using the models that TI provided for their op-amps and their comparators. Because of this we had the ability to simulate our circuit with supposed accuracy due to the parts parameters being directly provided by the maker of the hardware. With this being the case we simulated our triangle wave generator using the TI design software and got the graph shown in figure 11 in appendix B.

This data looked promising for us. The voltage peaks were exactly as calculated in our design and the switching frequency was operating as intended. With this solidified we moved onto the simulation of our error amplifier section. We performed a simulation with an ideal triangle wave running at 155 kHz as one input and a 1kHz sine wave as our imaginary audio signal as the other input. We achieved the PWM output seen in figure 12 in appendix B.

The PWM was simulating just as we intended. The output was following the sine input without achieving a duty cycle of greater than 95% or less than 5%. We had good simulations going for both our circuits. It appeared that everything was working as intended but once we ordered the PBCs and got the parts in the reality of the circuit was not the same. When we set the triangle wave generator up and began to probe the output we wouldn't see the triangle wave that we wanted. The wave had a peak to peak amplitude of 500 mV and was operating in the MHz range, which is far beyond the frequency that we wanted. The only thing that was working was the triangle wave being offset by around 2.5 volts, which is correct. After this we attempted to create the mock circuit that was provided in the documentation for this PWM circuit design. They ran the circuit at 500 kHz and got the triangle wave to output perfectly. We took the exact resistor and capacitor values that they used and assembled the circuit. We probed the output and saw that the circuit was working as intended. This implied that our circuit was working, and for some reason when we make the shift from high frequency to a lower frequency we obtained a different output than was thought. In the end we attempted many different ways to fix the triangle wave generator but ended up having to scrap the design due to the time restrictions on the project.

3.2.2 New PWM Circuit

To verify the correctness of our new PWM circuit, we examined its input output behavior using the oscilloscope. We show that using the TLS494, a dedicated PWM chip, as the means of producing a pulse width modulation as opposed to re-inventing the wheel and building our PWM circuitry up from op-amps and comparators, we see that the overall quality of our PWM modules ability to demonstrate the properties of accurate, highly tolerant, pulse width modulation on an arbitrary input signal has significantly increased. Doing a side by side visual comparison of figures 9 and 10 in Appendix B using our new circuit versus our old circuit, you can clearly distinguish which of the two produced a higher quality waveform.

3.3 Audio Decoder

The audio decoder was tested for correct operation by directly listening to the audio output by attaching the output signal to a headphone jack.

Shown in Figure 3 in the appendix is the finished PCB with the DAC. The left most 2x20 connectors will connect to the GPIO pins of the Raspberry Pi and provide the DAC with 16-bit digital input. The various wires are the power supplies and GND needed to connect to the OP amp and DAC. With everything wired together, one final wire is the output and connects to the headphone jack of a device we want to hear. If the sound we hear is the expected music, we can determine that the DAC is working correctly with the Raspberry Pi.

3.4 High Voltage Switching Circuit

3.4.1 Old Switching Circuit

The old switching circuit had one major design flaw. The high side gate was being driven by a low side gate driver. The main problem with this design is that a high side gate has to be isolated from the rest of the circuit. This needs to happen because we have such a large voltage being brought across it. If we were to keep the gate not isolated and attempt to run the switching circuit at our high voltage we would run into a problem where the gate to collector voltage would not be high enough to switch with full power. This is because if we fully open the high side switch the collector will see 120V. But if that is the case and our gate only has 10 volts across it, our gate to collector voltage would be negative, closing the switch.

Because of this great error in this circuit we knew that we needed to completely redesign the driver circuit. We needed to get a gate driver that could do both high and low side

3.4.2 High Side-Low side gate Driver and Switching Circuit

After we realized the mistake of our original gate driver we attempted to remedy our design by implementing a high side and low side combo gate driver. This gate driver was rated up to 600 volts so we had confidence that it would fit our needs. The first basic test that we decided to do was input a basic square wave operating at our resonant frequency into the high side gate input and see if the voltage is replicated on the gate of our switching circuit. When the high side gate was probed we saw that there was no gate voltage to be found. After doing some searching we found that we were operating our gate driver in Under-Voltage Lockout Mode. This mode is enabled when the gate driver detects something wrong with the voltage levels upon startup. There are many causes for this error, but in our case, it was happening because we did not have the correct resistance for our boost strap circuit. We had installed a resistor that was rated for 1 k Ω The resistor that we wanted was only 10 Ω . Because of this our RC constant was much lower than we needed and caused the voltage build up on our bootstrap capacitor to be too slow on start up. The gate driver detected this and locked itself down in order to preserve the driver. With this now known we installed the correct resistor value of 10 Ω and tested once again.

We once again ran a test on our gate driver where we supplied 25V to the high side of our high side switch. We input a simple square wave into our high side gate driver and measured the voltage on the high side gate. The gate was always high. The only thing that was good about this result is that we saw the full 25V on the output of our open circuit. This implies that our high side switch is isolated, so it will allow a voltage greater than 12 on the output. While this was good news, the fact that our gate was always high was not. After many hours of trial and error we realized that our PWM inputs were incorrect. The high side gate driver was getting a correct PWM input, but the low side was getting a phase shifted version of the high side gate, not an inverted version. Once we realized this we inverted the high side gate signal and input it into the low side gate input. With both of these gate signals being driven we were able to see the switching that we needed. When we used a low voltage for our high side reference (around 12 V) we saw a perfect output. But when we increased the high side voltage to around 25 V we began to see some massive ringing on our gate signals. The ringing's ripple was 8Vpp when our gate voltage only went up to 10 Volts. This ringing is large enough to turn the IGBT on or off at random, thus destroying our switching circuit. The worst part was as we increased the high side voltage, the ringing would increase even more. We deduced that most of this ringing was coming from the perf board that we soldered our components on to. These boards have a lot of stray inductance in them which can lead to the large ringing values that we saw before. Because we were in a time crunch we were unable to create a new PCB for our high side and low side gate driver circuit. If we were able to do that we would see our ringing on the gate decrease drastically.

4. Costs

4.1 Parts

Part	Manufacturer	Quantity	Retail Cost (\$)	Bulk Purchase Cost (\$)	Actual Cost (\$)
5" Pipe Fitting	Home Depot	1	\$4.63	\$3.70	\$4.63
3.5" Flange Pipe Fitting	Home Depot	1	\$3.14	\$3.14	\$3.14
3" PVC 90° Hub Elbow	Home Depot	4	\$3.44	\$3.44	\$13.76
Aluminum Foil	Reynolds Wrap	1	\$2.69	\$12.39	\$2.69
90/14 Sewing Needle	Schmetz Needles	1	\$2.99	\$8.90	\$2.99
All Purpose Cement	Oatley	1	\$8.54	\$21.09	\$11.15
36 AWG wire 12" wound on 14" long 3.5" Diameter PVC Pipe	McWilliams Tech	1	\$23.11	\$23.11	\$23.11
Raspberry Pi 3 Model B+	The Raspberry Pi Foundation	1	\$40.00	\$38.00	\$35.00
3.5" PVC Pipe Cap	Home Depot	1	\$3.40	\$6.60	\$3.40
APEVIA ATX-VS450W Venus 450W ATX Power Supply	Apevia	1	\$22.99	\$22.99	\$22.99
Break-away 0.1" 2x20-pin Strip Dual Male Header For Raspberry Pi Zero GPIO	Adafruit	2	\$1.03	\$0.90	\$2.06
HGTG30N60B3D IGBT	On Semiconductor	5	\$4.59	\$4.59	\$22.95
MUR820G Power Diodes	On Semiconductor	5	\$1.64	\$1.22	\$8.20
OPA365 Op-Amp	Texas Instruments	5	\$1.85	\$0.78	\$9.25
12mm Diameter CESS Amplifier Speaker Terminal Binding Post Banana Plug Jack Socket	CESS USA	5	\$7.49	\$5.49	\$37.45
M3 x 10 mm Hex Standoff Mounting Kit	RobotShop	1	\$5.49	\$3.80	\$4.10
UCC27714 High-Speed 4-A, 600-V High-Side Low-Side Gate Driver	Texas Instruments	5	\$2.94	\$1.42	\$14.70
Aluminum Cooler Radiator Heat Sink	UXCELL	3	\$8.53	\$5.58	\$25.59
DAC8820 16-bit Parallel Input	Texas Instruments	3	\$16.61	\$16.61	\$49.83

Multiplying Digital-to-Analog Converter					
THS4011 290 MHz Low-Distortion Voltage-Feedback Amplifier	Texas Instruments	6	\$4.60	\$4.60	\$27.60
Gorilla Glue	Gorilla Glue	1	\$5.97	\$6.35	\$5.97
OP277 High Precision Operation Amplifier	Texas Instruments	2	\$3.06	\$3.06	\$6.12
Total	N/A		\$519.29	\$197.76	\$336.68

Table 4.1: Parts Cost

4.2 Labor

Our labor costs are estimated to be around \$35/hour, 10hr/week for 3 people. This class will last for approximately 15 weeks. The total cost for labor equates to \$39,375.

$$\text{\$35/hour} * \text{10hours/week} * \text{15 weeks} * \text{3 people} * \text{2.5} = \text{\$39,375}$$

5. Conclusion

In the end our design fell short of being a fully working product. This is not to say that we didn't accomplish quite a bit in the design of our product. We were able to get audio out of our DAC by running an NES emulator. This is extremely hard because Linux is not a real time operating system so to be able to accomplish this alone was a great feat. We also were able to rectify wall voltage and create a DC-like voltage to run our high side voltage in our switching circuit. The design of the new PWM circuit created an inverted and non-inverted signal that we could provide to our gate drivers. Whether or not our design would work flawlessly if we fixed the ringing issue on our remains unseen, but we accomplished quite a large amount during this design process.

5.1 Future work

In the future we need fix up our switching circuit. The main reason that we cannot use our current circuit is because there is too much inductive ringing in our switching circuit. This is mostly due to the wires that come from the perf board. We also needed to use jumper wires to connect all our circuits together instead of having copper paths on a PCB. All these added wires create a large amount of inductance that introduce ringing. The ringing can be reduced by resistors and capacitors, but even when we added them to our circuit they didn't bring the design into working order. Therefore, if we want to have a working design we must create our PWM circuit and our switching circuit on their own PCB. This will allow much less ringing into our circuit and hopefully fix the circuit.

References

- [1] Texas Instruments, “Analog Pulse Width Modulation” *Texas Instruments*, 2013. [Online]. Available at: <http://www.ti.com/lit/ug/slau508/slau508.pdf>. [Accessed: May 1, 2018].
- [2] Texas Instruments, “UCC27714 High-Speed, 600-V High-Side Low-Side gate Driver with 4-A Peak Output” *Texas Instruments*, 2017. [Online] Available at: <http://www.ti.com/lit/ds/symlink/ucc27714.pdf>. [Accessed: May 1, 2018].
- [3] Texas Instruments, “TL494 Pulse-Width-Modulation Control Circuits” *Texas Instruments*, 2017. [Online] Available at: <http://www.ti.com/lit/ds/symlink/tl494.pdf>. [Accessed: May 1, 2018].
- [4] On Semiconductor, “Switch-mode Power Rectifiers” *On Semiconductor*, 2014. Available at: <https://www.onsemi.com/pub/Collateral/MUR820-D.PDF>. [Accessed: May 1, 2018].
- [5] “LaiNES”, github.com, 2017. [Online]. Available at: <https://github.com/AndreaOrru/LaiNES>. [Accessed: Mar. 25, 2018].
- [6] “NES Emulator”, github.com, 2016. [Online]. Available at: <https://github.com/amaiorano/nes-emu>. [Accessed: Mar. 25, 2018].
- [7] “Simple NES”, github.com, 2017. [Online]. Available at: <https://github.com/amhndu/SimpleNES>. [Accessed: Mar. 25, 2018].
- [8] “16-bit, Parallel Input Multiplying Digital-to-Analog Converter”, ti.com, 2008. [Online]. Available: <http://www.ti.com/lit/ds/symlink/dac8820.pdf>. [Accessed: April 30, 2018]
- [9] “IEEE Code of Ethics”, ieee.org, 2016. [Online]. Available at: <http://www.ieee.org/about/corporate/governance/p7-8.html>. [Accessed: May 1, 2018].
- [10] R. M. Craven, I. R. Smith and B. M. Novac, “Magnetic Coupling In Tesla transformers”, in *Applied Physics Research*, vol. 8, no. 6, pp. 101-105, October 2016.
- [11] “The Physics Factbook”, hypertextbook.com, 1997. [Online] Available at: <https://hypertextbook.com/facts/2000/AliceHong.shtml>. [Accessed: May 1, 2018].
- [12] M. Simmonds, “How to Make A musical Solid State Tesla Coil (SSTC) That Plays Guitar!” *Instructables.com*, 2015. [Online]. Available at: <http://www.instructables.com/id/How-To-Make-A-Musical-Solid-State-Tesla-Coil-SSTC/>. [Accessed: May 1, 2018].
- [13] “Statistical Analysis Handbook 2018 Edition”, www.statsref.com, 2018. [Online] Available at: <http://www.statsref.com/HTML/index.html>. [Accessed: May 1, 2018].
- [14] “The Linux Man Pages: chrt”, man7.org, January 2016. [Online] Available at: <http://man7.org/linux/man-pages/man1/chrt.1.html>. [Accessed: May 1, 2018].

Appendix A Requirement and Verification Table

Requirements	Verification	Y/N
NES APU Emulator		
The Emulator must read NES ROM image audio data and communicate with the Raspberry Pi's GPIO interface using the Wiring Pi library at the clock speed of the NES.	<ol style="list-style-type: none"> 1. Set the appropriate linker flags in the MakeFile so that Wiring Pi's code is included in the emulator's address space. 2. Then write a function to strobe the GPIO pin voltages 3. Measure the pin voltages with an oscilloscope 4. Confirm the physical behavior matches the test strobe signal specified by software. 	<ul style="list-style-type: none"> • Y
Must use the Simple DirectMedia Layer library as a way of generating multimedia output.	<ol style="list-style-type: none"> 1. View sections of the emulator's source code, specifically areas where the emulated APU is implemented. 2. Verify that the Simple DirectMedia Layer is integrated. 	<ul style="list-style-type: none"> • Y
Must execute smoothly within the Raspberry Pi's Linux Environment and accurately reflect the runtime behavior of the NES.	<ol style="list-style-type: none"> 1. Implement a software clock to measure the Emulator's cycle speed, and log clock measurements before and after critical sections. 2. Confirm that the interval displacing the two clocks approximates the same timing measurements made on the NES. 	<ul style="list-style-type: none"> • Y
Emulator Media Layer Intermediary		
Must function as a transparent intermediary interface located between APU's audio output and SDL's sound buffer.	<ol style="list-style-type: none"> 1. Measure the performance cost of the intermediary by measuring clock displacement, every time the NES is asked to fetch an instruction. 2. Verify that the intermediary marginal cycle performance cost is 0.05 milliseconds or lower. 	<ul style="list-style-type: none"> • Y

The Raspberry Pi environment must have the POSIX thread library available for linking at compile time.	<ol style="list-style-type: none"> 1. Compile the intermediary with the linker flag -lpthread set. 2. Verify that the POSIX threads are supported, when compilation should proceed without returning an error. 	<ul style="list-style-type: none"> • Y
Audio Decoder		
Must satisfy frequency operating conditions within the 1 kHz-100 kHz to correctly sample audio data.	<ol style="list-style-type: none"> 1. Directly reference the operational amplifiers datasheet and locate the Gain Bandwidth Product. 2. Check that the specified bounds form a valid subset of the frequency interval formed between 1 kHz and 100 kHz. 	<ul style="list-style-type: none"> • Y
Must be able to produce continuous time analog signals of the digital audio signal output by the GPIO pins.	<ol style="list-style-type: none"> 1. Connect audio signal to headphone jack 2. Listen for expected music 3. Check that music is audible and clear 	<ul style="list-style-type: none"> • Y
Power Supply Unit		
Be able to handle a high enough frequency to operate at the resonance frequency of the coil	<ol style="list-style-type: none"> 1. Find the resonance frequency of our coil 2. Confirm that the PWM circuit can operate at such a frequency 	<ul style="list-style-type: none"> • Y
Be able to supply multiple amps worth of current into our gate (2-3 amps)	<ol style="list-style-type: none"> 1. Select a gate driver based on our FET that we use. 2. Run current through this gate driver using a bench power supply to make sure that it can run at the rated current for the FET. 	<ul style="list-style-type: none"> • Y

The MOSFETs must be able to handle at least 300 volts across source terminals without the occurrence of junction breakdown.	<ol style="list-style-type: none"> 1. Apply 300 volts across the MOSFETs with a load that draws the correct amount of current to simulate the coil 2. Verify that MOSFETs behave correctly 	<ul style="list-style-type: none"> • Y
The MOSFETs must be able to carry out the switching behavior within 10-15% of the Tesla Coil's resonant frequency.	<ol style="list-style-type: none"> 1. Attempt switching the FETs at the rated frequency and clean the waves. 2. Ensure that MOSFETs operate correctly at resonant frequency 	<ul style="list-style-type: none"> • Y
The MOSFETs must be able to remain within their appropriate operating temperature ranges as specified by their datasheet.	<ol style="list-style-type: none"> 1. Operate MOSFETs as specified during previous requirements 2. Verify using a heat gun on the FETs that the temperature does not exceed ratings 	<ul style="list-style-type: none"> • N
Must have access to a 5V rail, and a 12V rail	<ol style="list-style-type: none"> 1. Check that the power supply has a 5V rail and 12V rail 2. Ensure both rails are operating at their respective voltages using oscilloscope 	<ul style="list-style-type: none"> • Y
The Full Wave Rectifying Circuit must be able operate under 120VAC	<ol style="list-style-type: none"> 1. Run the circuit using a 20Vpp sine wave, then step up the voltage in increments of 20 using a Variac under no load 2. Run the circuit under load for 1 minute at 120Vpp 	<ul style="list-style-type: none"> • Y

Step Up Air Core Transformer		
Must be rated for at least 150 W	<ol style="list-style-type: none"> 1. Check specifications given for the power supply 2. Ensure that it is rated for at least 150W 	<ul style="list-style-type: none"> • Y
The transformer should have an air core and step up the secondary terminal voltage 180-220 times the primary terminal voltage.	<ol style="list-style-type: none"> 1. Look at our primary and secondary coil winding ratio to ensure a proper step-up voltage ratio 2. Measure the output voltage 3. Verify that the output voltage 180-220 times the input voltage 	<ul style="list-style-type: none"> • Y
The transformer should be helical and the length between the inner and outer radii of the concentric coils should be small enough such that the primary and secondary coils magnetically interact, but far enough such that the primary and secondary coils electrically interact.	<ol style="list-style-type: none"> 1. Measure all defining attributes of the transformer: like magnetizing inductance, DC resistance, primary and secondary line impedance, core resistance 2. Use these measurements to ensure that the transformer will not saturate 	<ul style="list-style-type: none"> • Y
The wires used should be able to account for the skin effect when operating at our coils resonant frequency	<ol style="list-style-type: none"> 1. Calculate the current density of copper at the switching frequency to ensure that we are not current saturating our wires 2. Use enough wires in parallel to allow enough space for the current to transfer through 	<ul style="list-style-type: none"> • Y

Toroidal Top load

<p>The toroidal top load should function as a capacitor so it can store and release charge into the air. We want a capacitance in the range of 10-20 pF.</p>	<ol style="list-style-type: none"> 1. Measure the inductance and capacitance value of the coil, along with its parasitics 2. Ensure that we are seeing the capacitance granted by the top load 	<ul style="list-style-type: none"> • Y
<p>The Toroidal Top Load's aluminum coating should not noticeably deteriorate as a result of the electrical discharge.</p>	<ol style="list-style-type: none"> 1. Run the coil and look at the top load to ensure it does not break down. 2. After resting for an hour, rerun and verify that it still functions. 	<ul style="list-style-type: none"> • Y
<p>The toroidal top load should be the region of emission for the desired electrical discharge creating the arcs from 4-7 inches in length.</p>	<ol style="list-style-type: none"> 1. Run the coil 2. Ensure that the arcs produced are visible at the toroidal top and around 4-7 inches in length 	<ul style="list-style-type: none"> • Y

	Heat Sink	
The heat sink must regulate device temperatures such that they stay within their appropriate ranges as indicated by their data sheets throughout the entire Tesla Coil operation cycle.	1. Measure the temperature of all of our sensitive equipment for a 2 minute test at full load and ensure that they stay within specified temperature ranges	<ul style="list-style-type: none"> Y
The heat sink should cover the devices we want to thermally regulate.	1. Verify that the heatsink has a large surface area with fins on the outside to ensure maximum heat transfer	<ul style="list-style-type: none"> Y

Appendix B Additional Figures

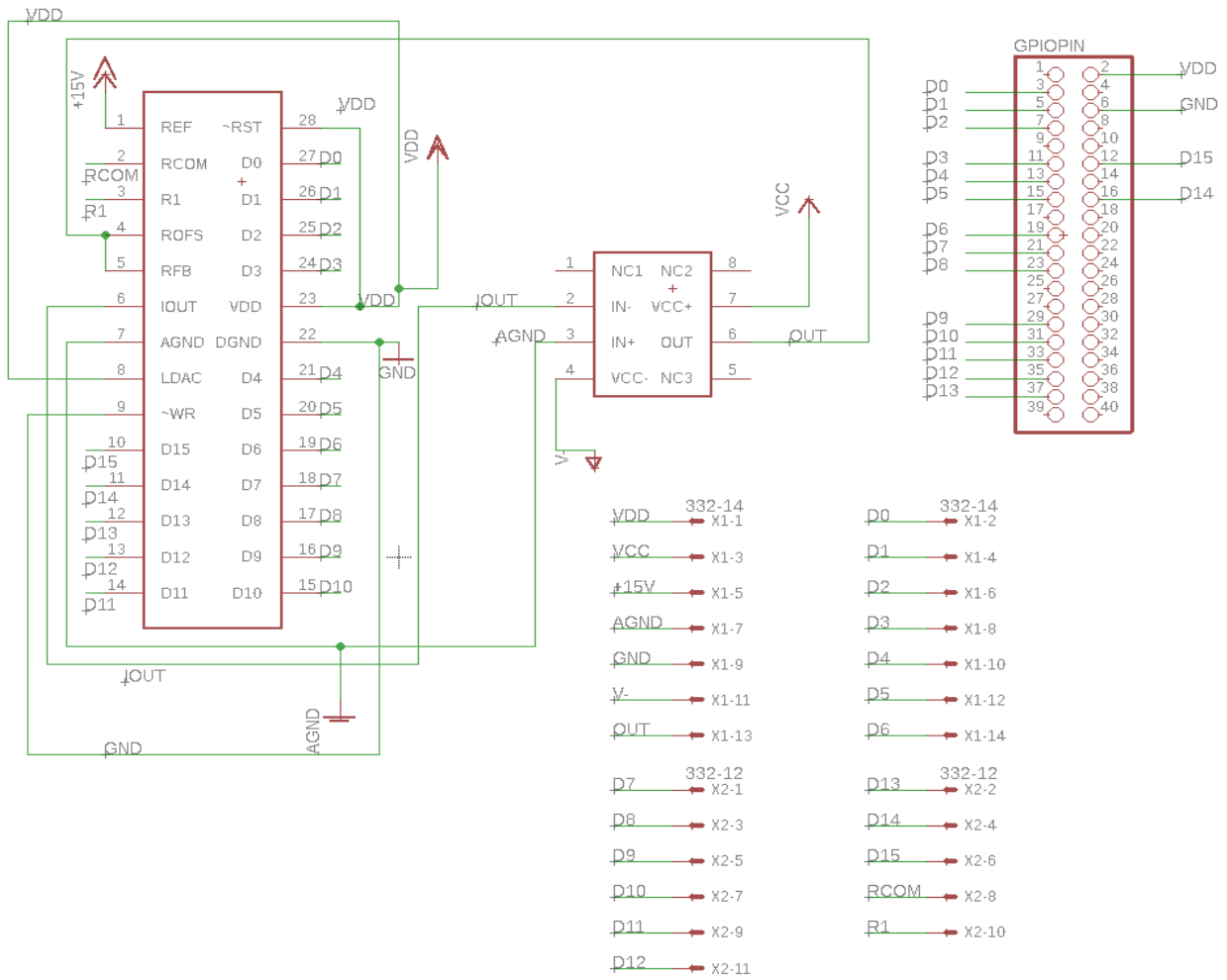


Figure 1: DAC EAGLE Schematic

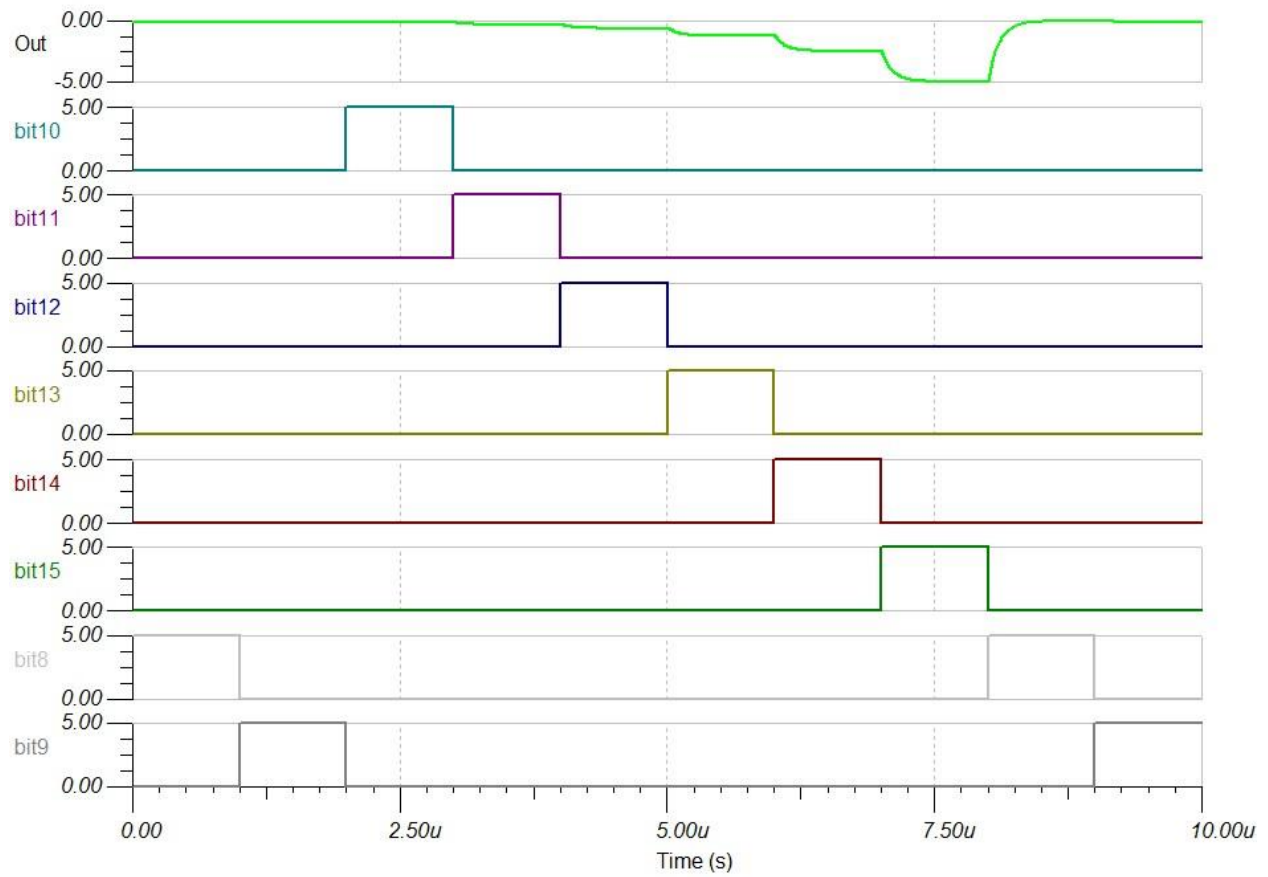


Figure 2: DAC TINA TI waveform

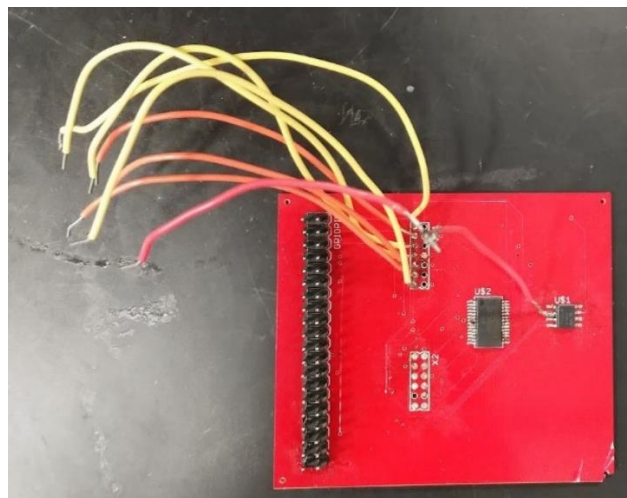


Figure 3: DAC PCB

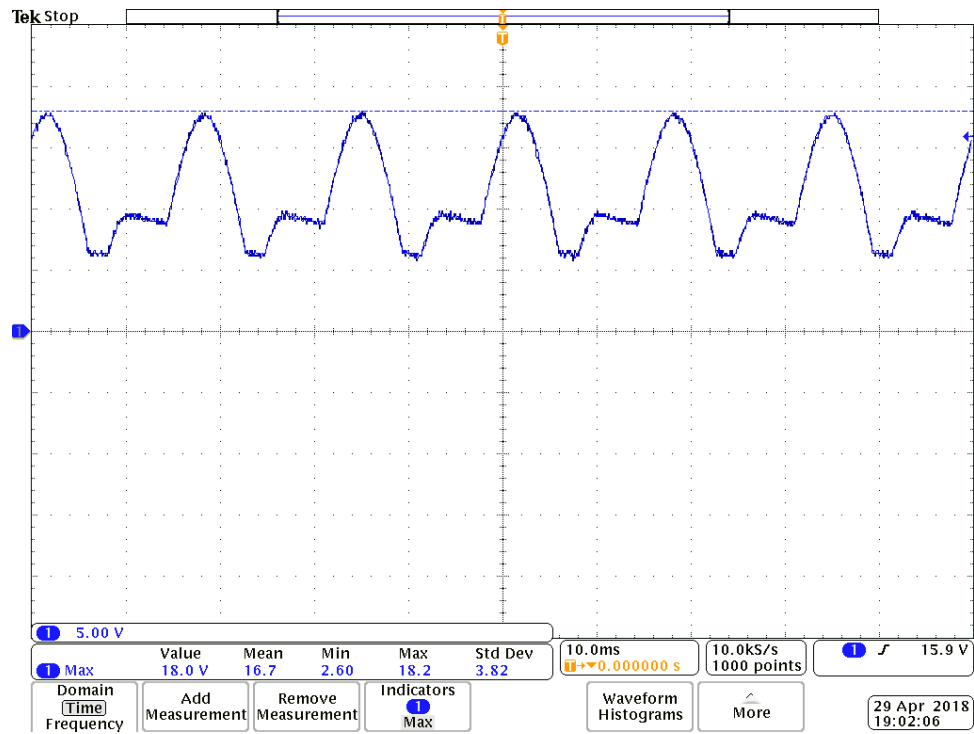


Figure 4: Voltage Output of Full-Wave Rectifier Circuit with no Capacitor

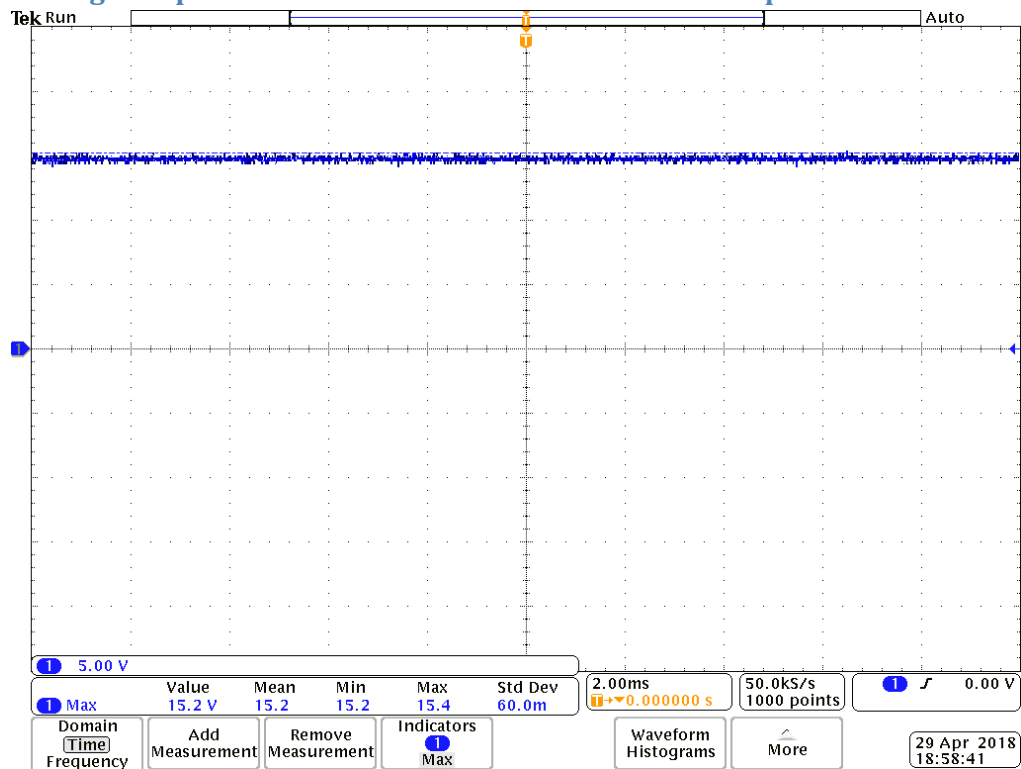


Figure 5: Voltage Output of Full-Wave Rectifier Circuit with Capacitor (No-Load)

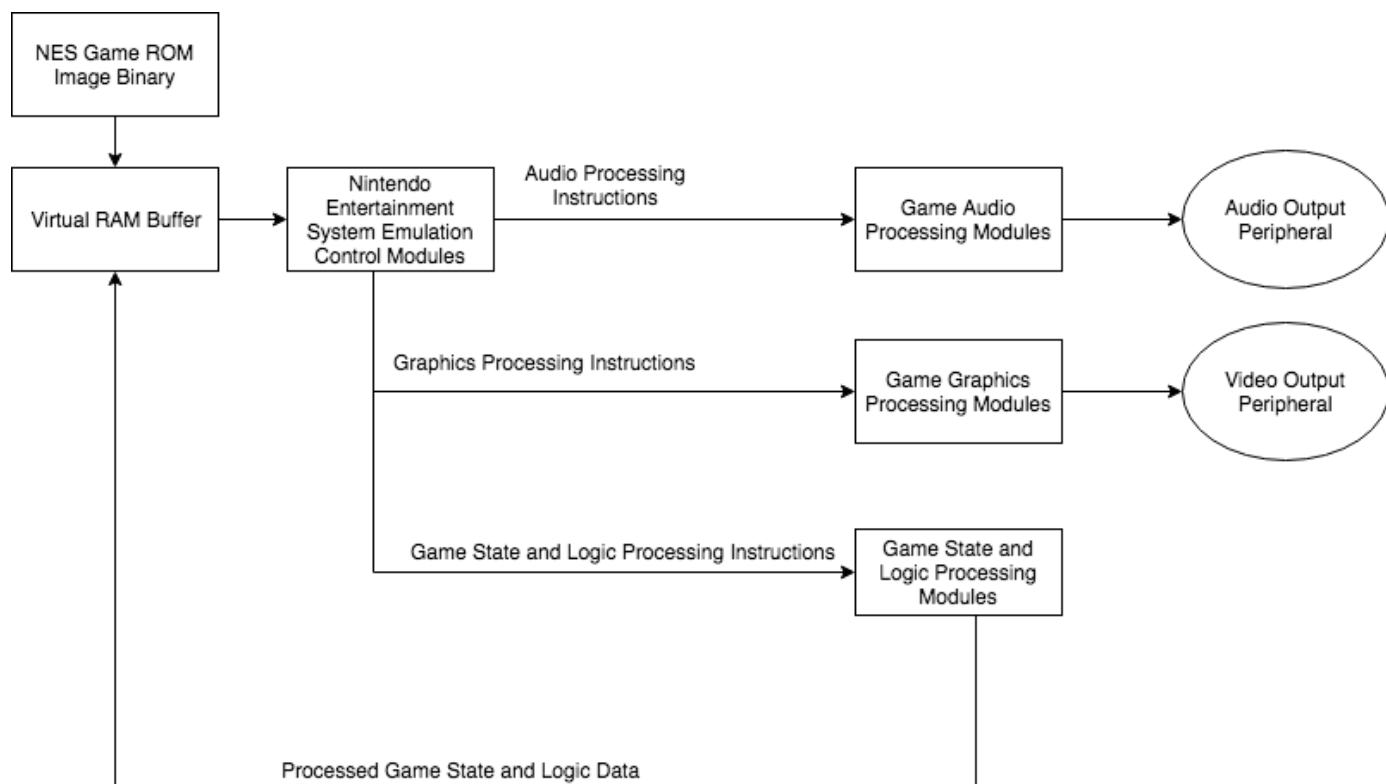


Figure 6: NES Emulator Flowchart

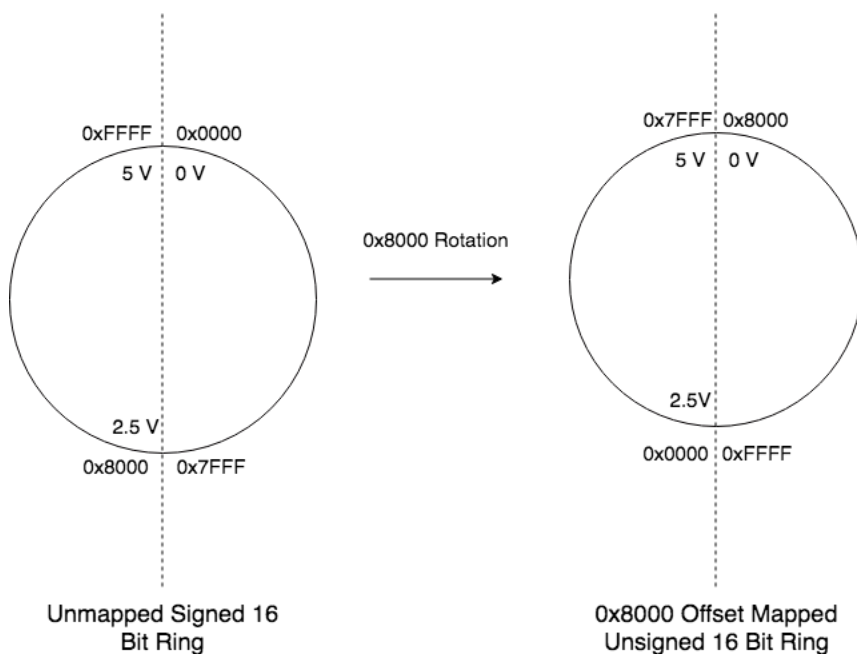


Figure 7: NES Ring Addition

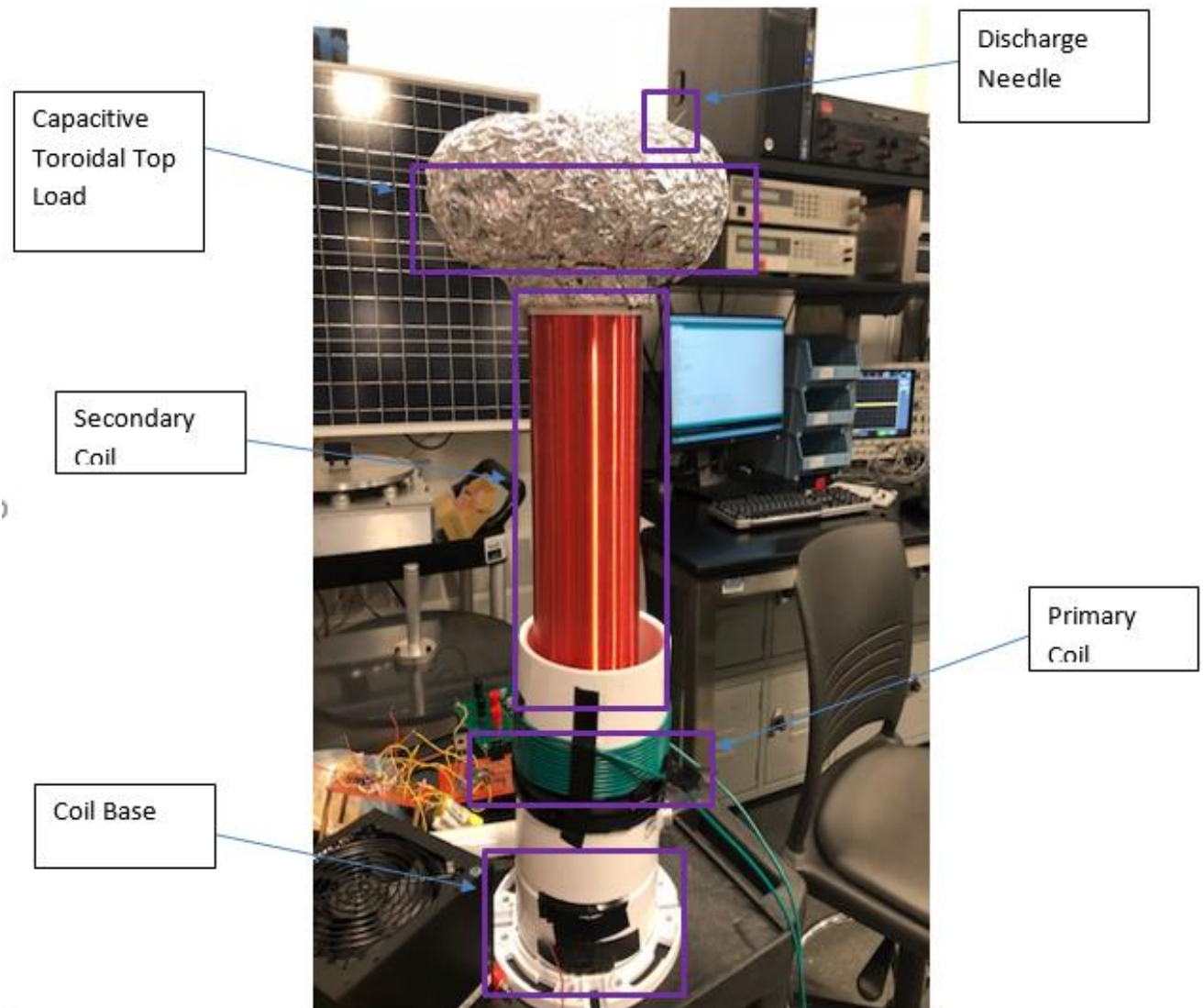


Figure 8: Tesla Coil's Structural Component.

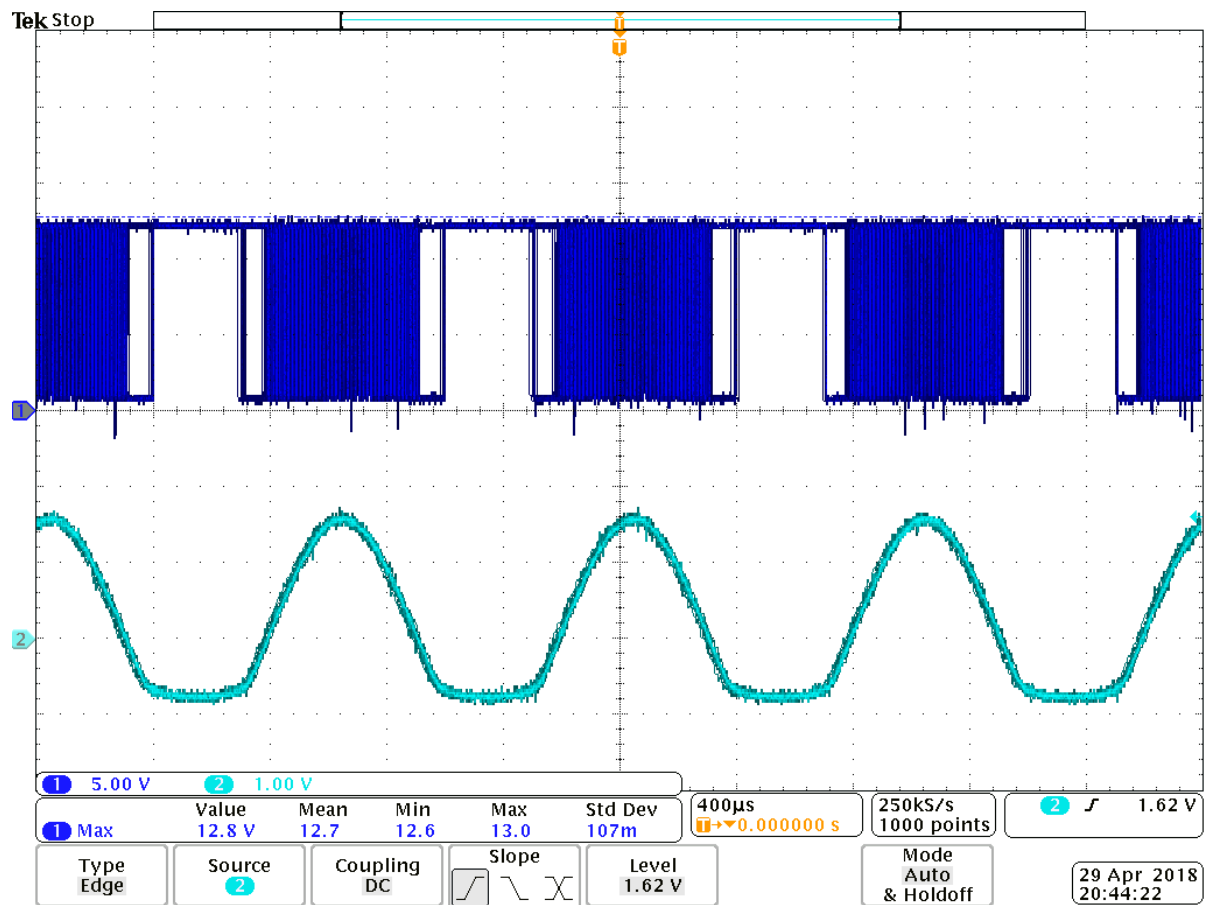


Figure 9: Clean PWM Circuit

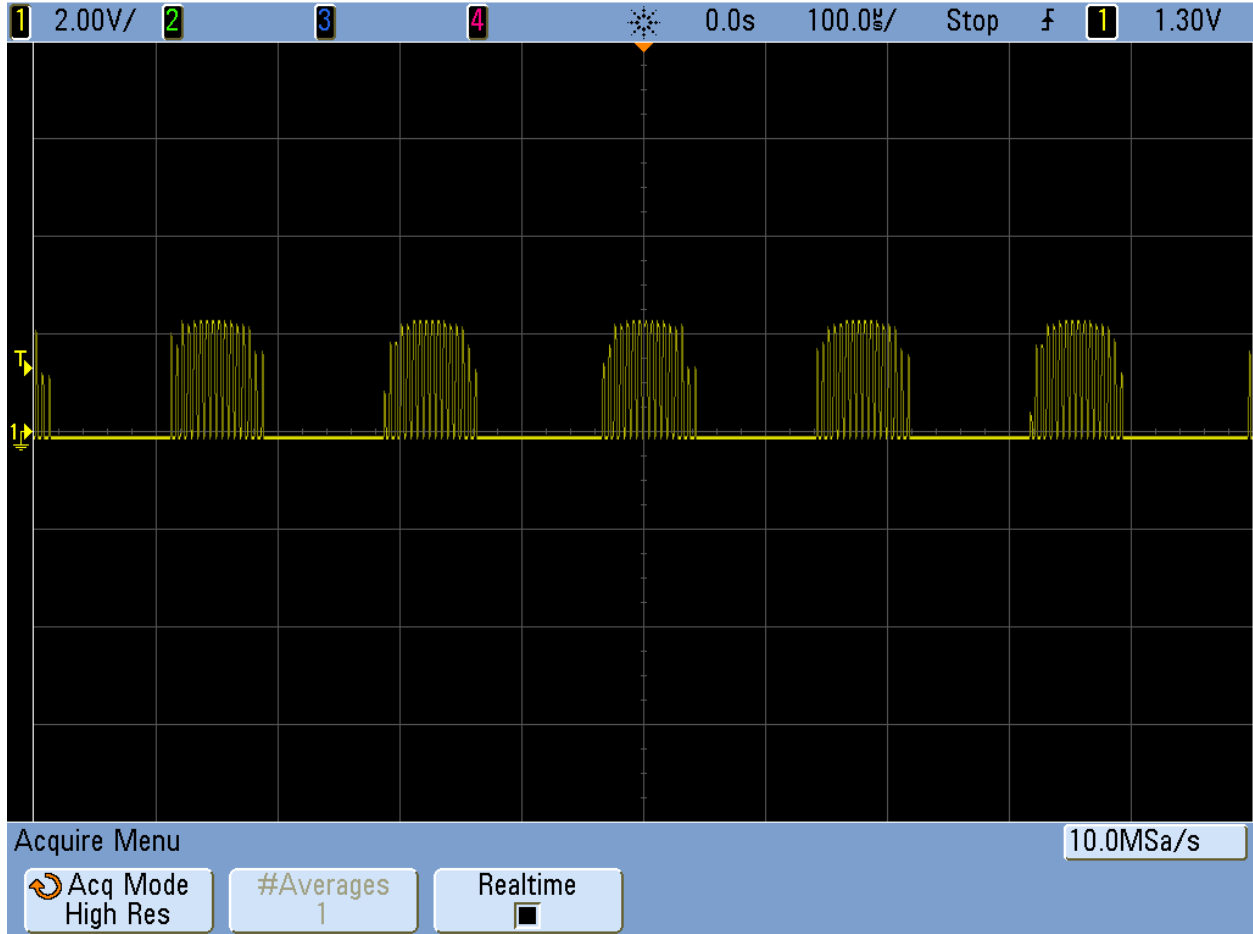


Figure 10: Old PWM Circuit

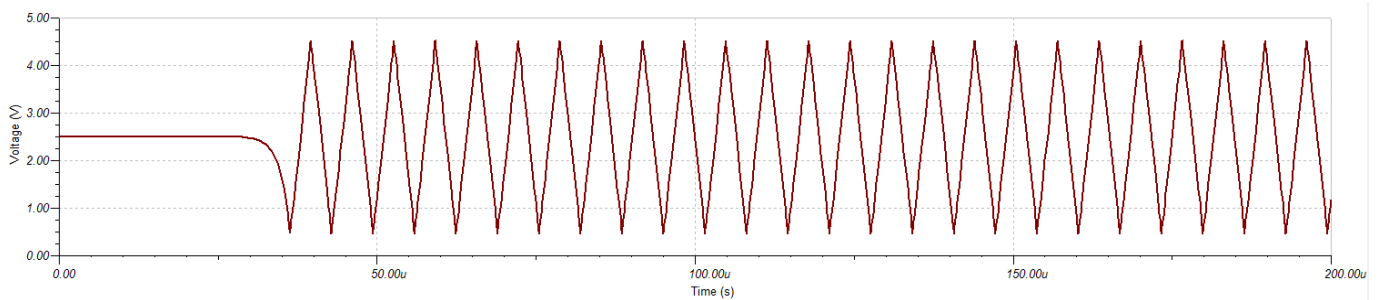


Figure 11: Triangle Wave Generator Output

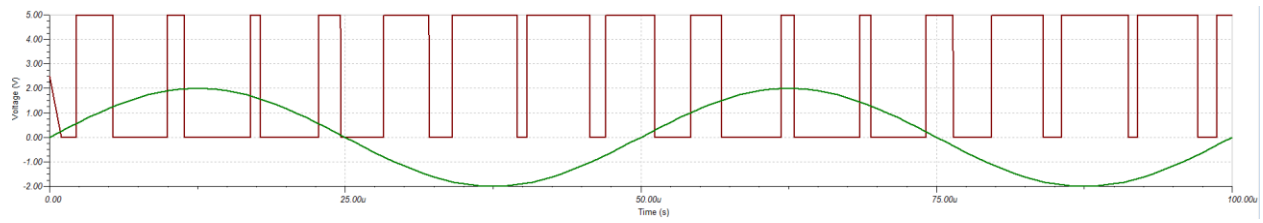


Figure 12: PWM Output Using a 1 kHz Sine Wave as an Input

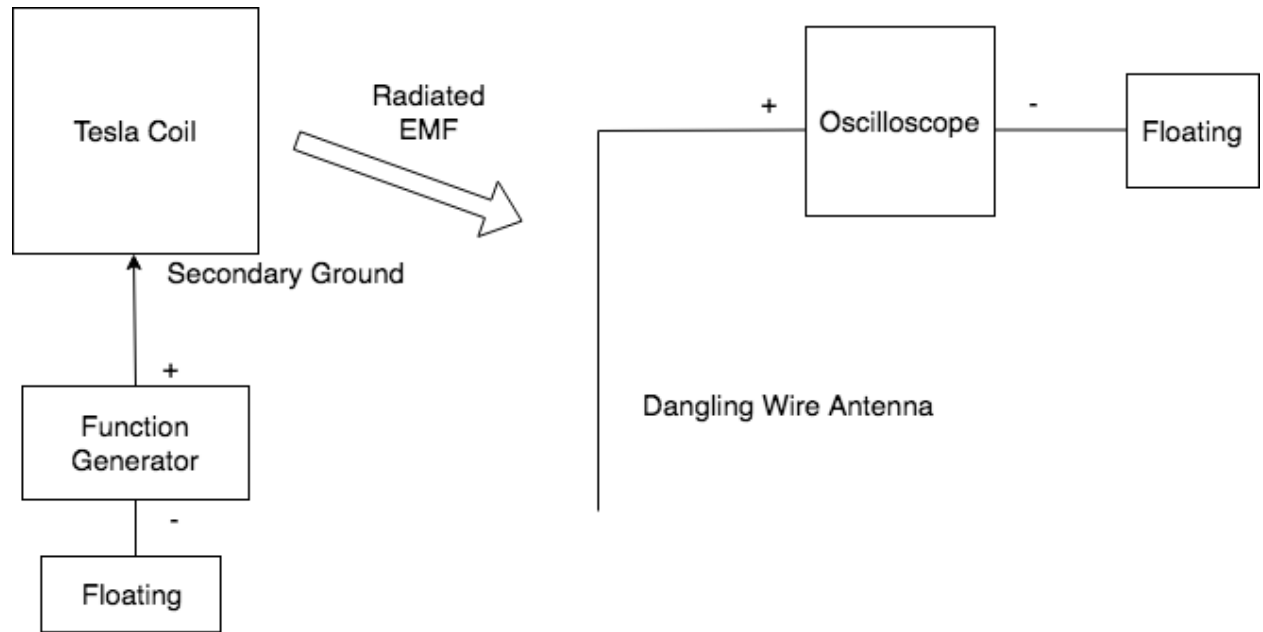


Figure 13: Summary of Our Coil's Mechanical Specifications

Appendix C Design Equations

$$V_{out} = -V_{ref} * \frac{D}{65536} \quad (1)$$

$$f_{res} = \frac{1}{2\pi\sqrt{L_S C(1-k^2)}} \quad (2)$$

$$k = \sqrt{1 - \frac{1}{L_S C(2\pi f_{res})^2}} \quad (2.1)$$

$$L_w = \frac{r^2 N^2}{9r + 10l} \quad (3)$$

$$C_S = 2.8(1 - \frac{d_2}{d_1}) \sqrt{\frac{2\pi^2(d_1 - d_2)}{4\pi}} (\frac{d_2}{2}) \quad (4)$$

$$R = \rho \frac{L}{A} \quad (5)$$

$$\mathbb{E}[T_k] = \frac{k}{\lambda} \quad (6)$$

$$X(T) = \frac{T - \mathbb{E}[T]}{\sqrt{Var(T)}} \quad (7)$$

$$C_{BOOT} \geq 10C_g \quad (8)$$

$$f = \frac{1}{2R_T C_T} \quad (9)$$

$$\ell_{spark} = 1.7\sqrt{P - P_0} \quad (10)$$

$$E = \lim_{\sigma \rightarrow 0} \frac{Q}{4\varepsilon_0 \sigma^3 \pi^{\frac{3}{2}} r^2} (\sqrt{\pi} \operatorname{erf}\left(\frac{r}{\sigma\sqrt{2}}\right) - \frac{r\sqrt{2}}{\sigma} e^{\frac{-r^2}{2\sigma^2}}) \quad (11)$$

$$\nabla^2 V = \frac{\partial}{\partial r} (r^2 \frac{\partial V}{\partial r}) = \lim_{\sigma \rightarrow 0} \frac{Q}{\sigma^3 (2\pi)^{3/2} \varepsilon_0} e^{\frac{-r^2}{2\sigma^2}} \quad (12)$$

$$|\frac{5\lambda^{-1}}{\mathbb{E}[T]}| \leq 0.2 \quad (13)$$

$$\frac{V_p}{V_s} \approx \frac{N_p}{N_s} \quad (14)$$

$$\nabla^2 V = -\frac{\rho}{\varepsilon} \quad (15)$$

$$f_{tri} = \frac{R_6}{4R_7 R_5 C_2} \quad (15)$$

$$V_{pp\ tri} = \frac{R_5}{R_6} V_{ref} \quad (16)$$

$$V_{PWM} = \left(-\frac{R_4}{R_3} * V_{IN} \right) + V_{REF} \quad (17)$$

$$f_p = \frac{1}{2\pi R_4 C_1} \quad (18)$$

$$V_{offset} = \left(1 + \frac{R_4}{R_3} \right) \left(\frac{R_2}{R_1 + R_2} \right) V_{REF} \quad (19)$$

$$Q = C * V \quad (20)$$

$$B = \mu_0 H + M \quad (21)$$

$$B = \mu_0 \mu_r H \quad (22)$$

$$\oint_{\partial\Gamma} E \cdot d\ell = -\frac{d}{dt} \iint_{\Gamma} B \cdot dS \quad (23)$$

$$\oint_{\partial\Gamma} B \cdot d\ell = \frac{1}{c} (4\pi \iint_{\Gamma} J \cdot dS + \frac{d}{dt} \iint_{\Gamma} E \cdot dS) \quad (24)$$

$$E = -\nabla V \quad (25)$$

Appendix D Coil Specification Tables

Diameter of the primary coil	5 inches
Diameter of the secondary coil	3.5 inches
Height of the primary coil	1.75 inches
Height of the secondary coil	12 inches
Length of the discharge needle	1.1 inches
Number of windings on the primary coil	10 Turns
Number of windings on the secondary coil	2,100 Turns
Outer Diameter of the Capacitive Toroid	8.5 inches
Height Diameter of the Capacitive Toroid	3.75 inches
Height of the secondary coil	12 inches
Length of the discharge needle	1.1 inches
Number of windings on the primary coil	10 Turns
Number of windings on the secondary coil	2,100 Turns
Outer Diameter of the Capacitive Toroid	8.5 inches
Height Diameter of the Capacitive Toroid	3.75 inches

Table 1: Summary of Our Coil's Mechanical Specifications

	Calculated Result	Measured Result	Magnitude of Percent Error Between Them
Inductance of the primary coil	15.625 μH	27.5 μH	76.0%
Inductance of the secondary coil	99.49 mH	278 mH	179.4%
Capacitance of the toroidal top load	8.76 pF	14.3 pF	63.2%
Resistance	1231.19 Ω	978 Ω	20.5%

Table 2: Summary of Our Coil's Passive Electrical Specifications

Appendix E Printed Circuit Board Layouts

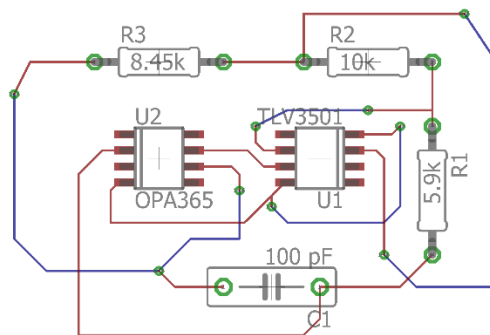


Figure 1: Triangle Wave Generator PCB

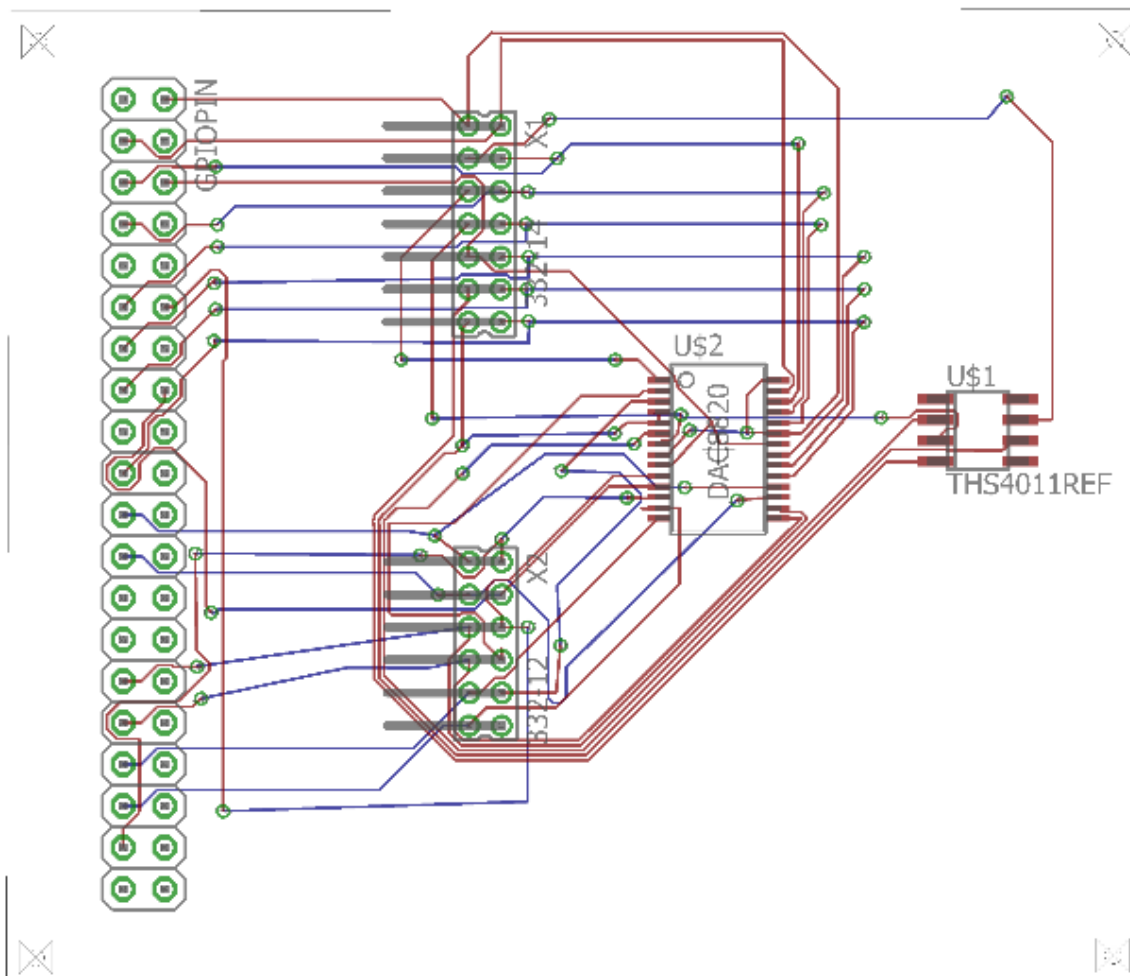


Figure 2: Digital to Analog Circuit PCB

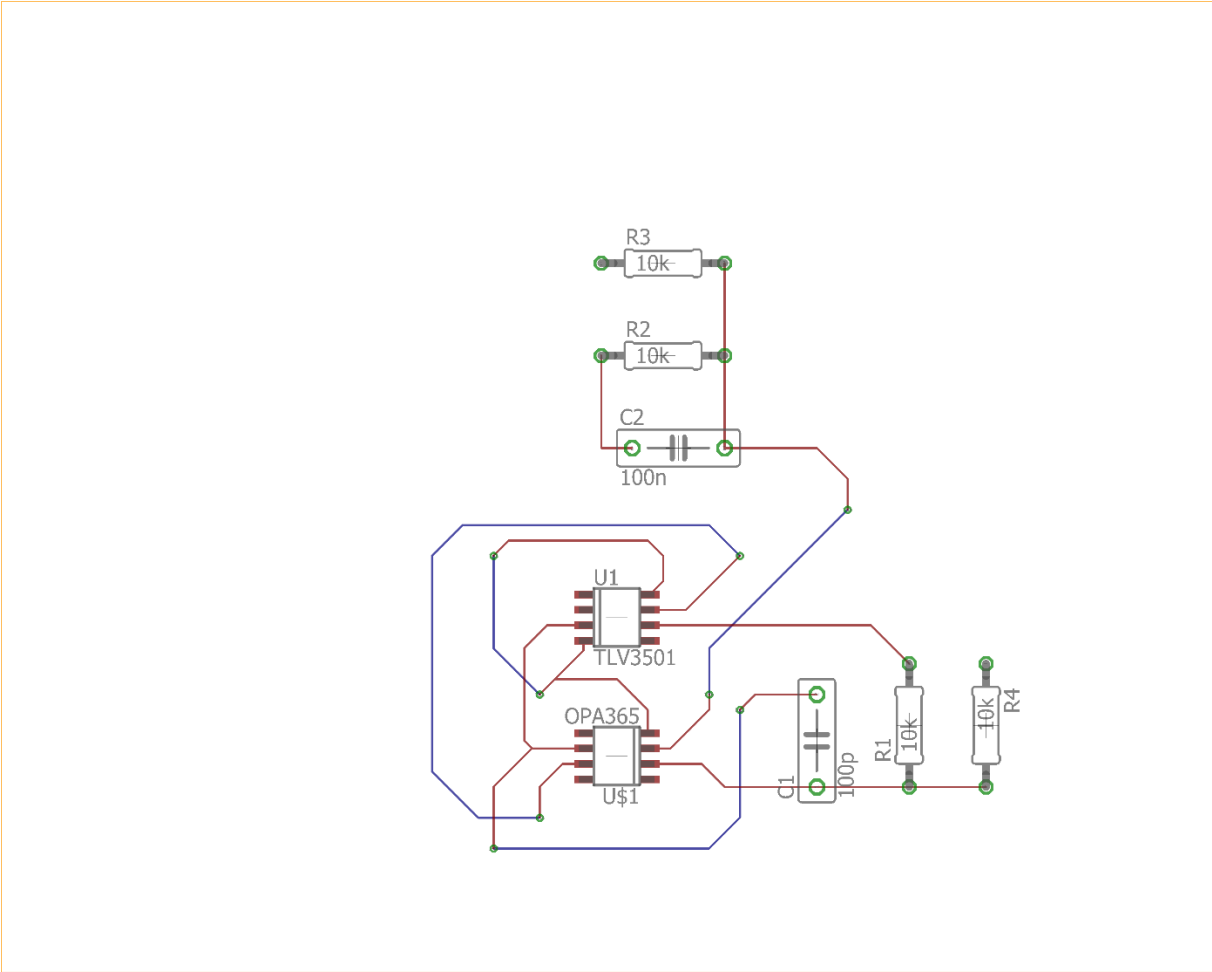


Figure 3: Error Amplifier PCB

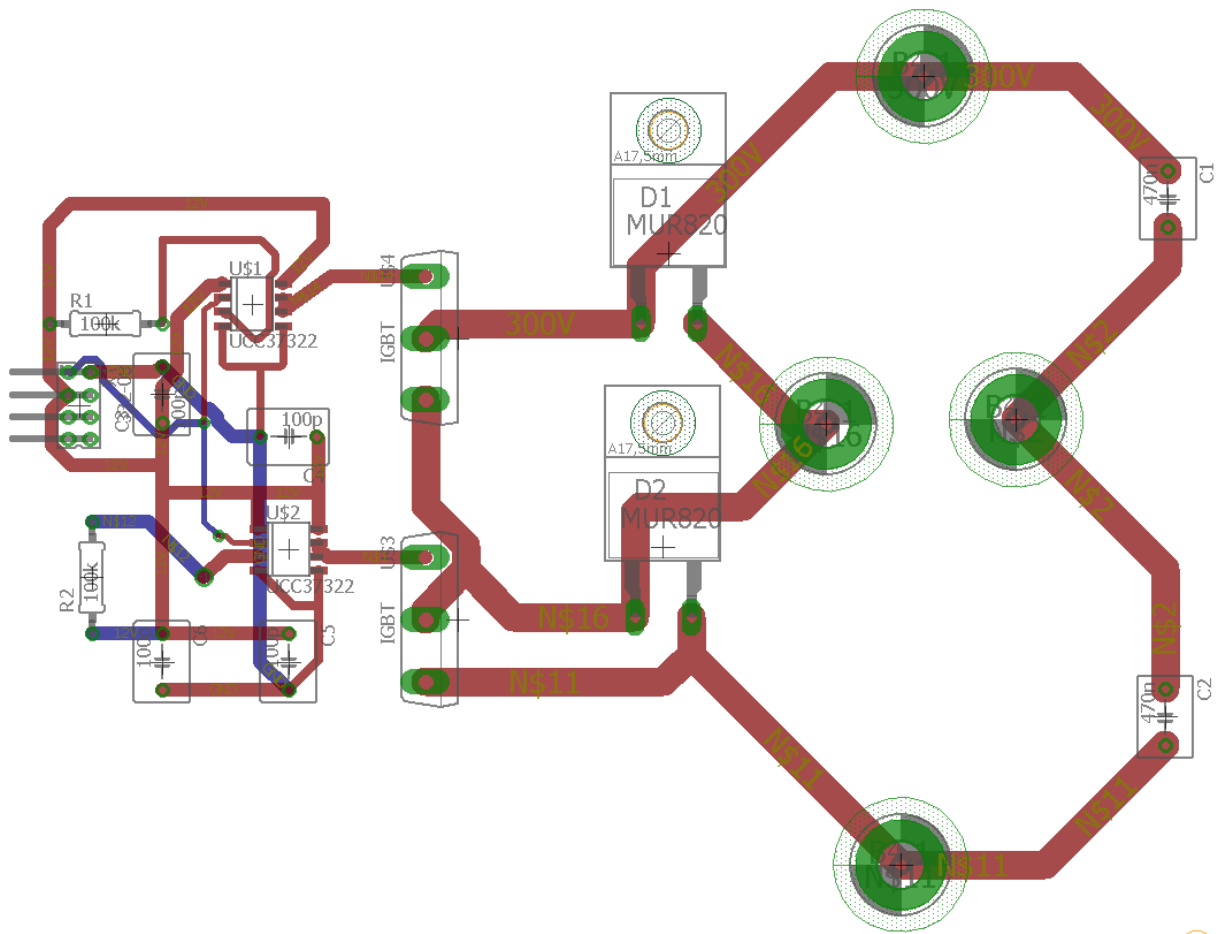


Figure 4: Switching Circuit PCB