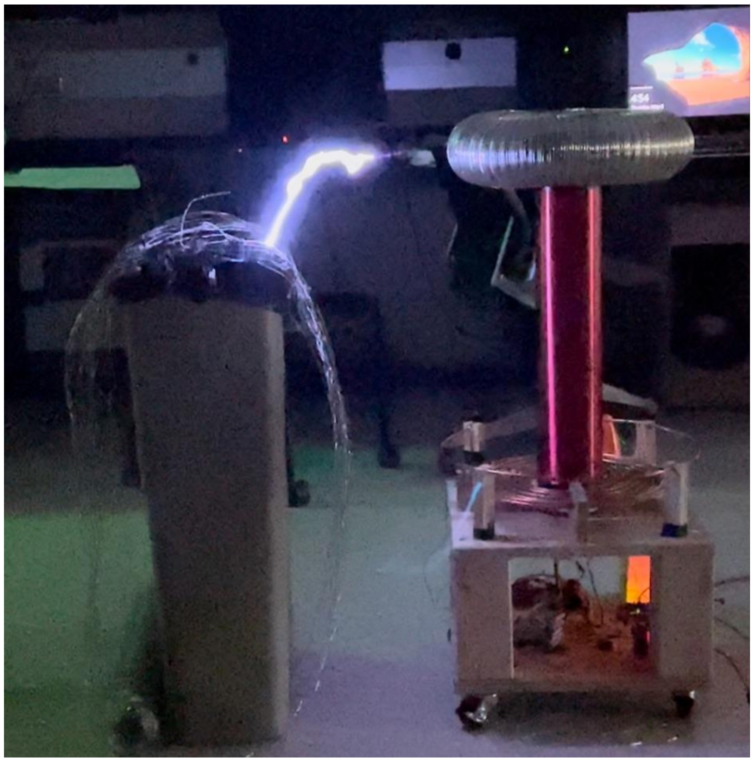
Musical Tesla Coil



Leo Corelli

Maximillian Domenech

Dennis Krivda

Table of Contents:

1. Introduction:

Tesla coils were first invented by famous inventor Nikola Tesla in 1891. It is an electrical resonant transformer circuit used to produce high-voltage, low-current, high frequency alternating-current (AC) electricity. Tesla initially intended for it to be a method of wireless power transmission, particularly for applications in wireless lightning, but ultimately decided that the design was not feasible or practical. Today, Tesla coils are an electrical engineer's dream, with many career EEs and electronics hobbyists seeking to recreate this magical circuit that shoots lightning around the room.

Tesla invented and patented his coil 50 years before the invention of the first transistor by Bell Labs in 1947. He used a spark gap to achieve the high voltage, high frequency switching necessary to operate the coil at the circuit's resonant frequency. Today, recent advancements in transistor technology has led to the rise of IGBTs (or insulated gate bipolar transistors), which allow us to control the high voltage, high frequency switching with an IGBT package about half the size of a modern smartphone. The IGBT allows us to control the operation of the coil with a program written in C that can output precise frequencies at the output of the coil. This is used to play musical notes (corresponding to a particular frequency) with the spark output from the breakout point of the coil.

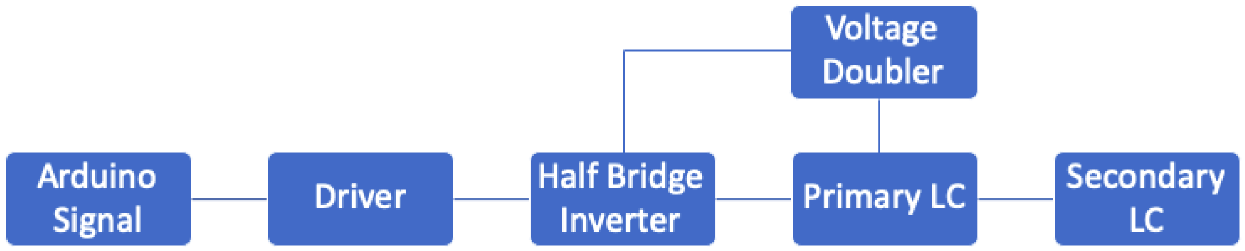
Our coil utilizes this IGBT technology. Therefore, it is a solid state (as opposed to spark gap) Tesla coil. We have also matched the resonant frequency of the primary LC circuit with that of the secondary coil's LC. This dual resonance between the primary and secondary leads to increased output (bigger sparks) at the breakout point of the coil because the primary and secondary coils share the same resonance frequency. Our coil is therefore classified as a dual resonant, solid state Tesla coil that can produce sparks of up to a foot in length.

2. Objectives:

1. Construct a solid state Tesla coil with respectable spark performance
2. Pulse sparks to make music
3. Play at least 5 songs

3. Block Diagram and High Level Descriptions:

1. Block Diagram:



2. High Level Descriptions:

   a. Arduino Signal

The Arduino will output a signal square wave at the resonant frequency of the primary LC. This square wave will be sent for enough time for the primary LC to oscillate in resonance and build up enough energy to induce enough energy in the secondary to generate a spark (~120us). This square wave pulse will then be repeated at a certain frequency to match the frequency of the note for the desired length of time of the note.

   b. Driver

The Driver will receive the output of the Arduino signal and amplify it to 18V, which it receives from two 9V batteries in series, with a MOSFET driver. It then outputs that to a gate driver transformer that is connected to the gates of the half bridge inverter. The gate driver transformer provides galvanic isolation for the control circuitry from the primary LC, eliminates the floating GND on the upper transistor in the inverter, and inverts the signal for the gate of the lower transistor. This circuit will also contain resistors, capacitors, and diodes to reduce stray inductance on the gates.

   c. Voltage Doubler

This takes the output from the wall voltage (120Vrms, 60Hz), and rectifies it using high voltage capacitors and high voltage, high current diodes into a 340V DC signal on the high side of the Half Bridge Inverter and a 0V DC signal on the lowside.

   d. Half Bridge Inverter

The Half Bridge Inverter receives the 18V squarewave output from the driver on the gate of the upper transistor and an inverted form of the signal on the lower transistor. The transistors are IGBTs (more on these later) to allow for efficient high current flow. This will take the 340 V input from the doubler and switch it on and off to supply the LC with a high voltage squarewave output at its resonant frequency.

   e. Primary LC

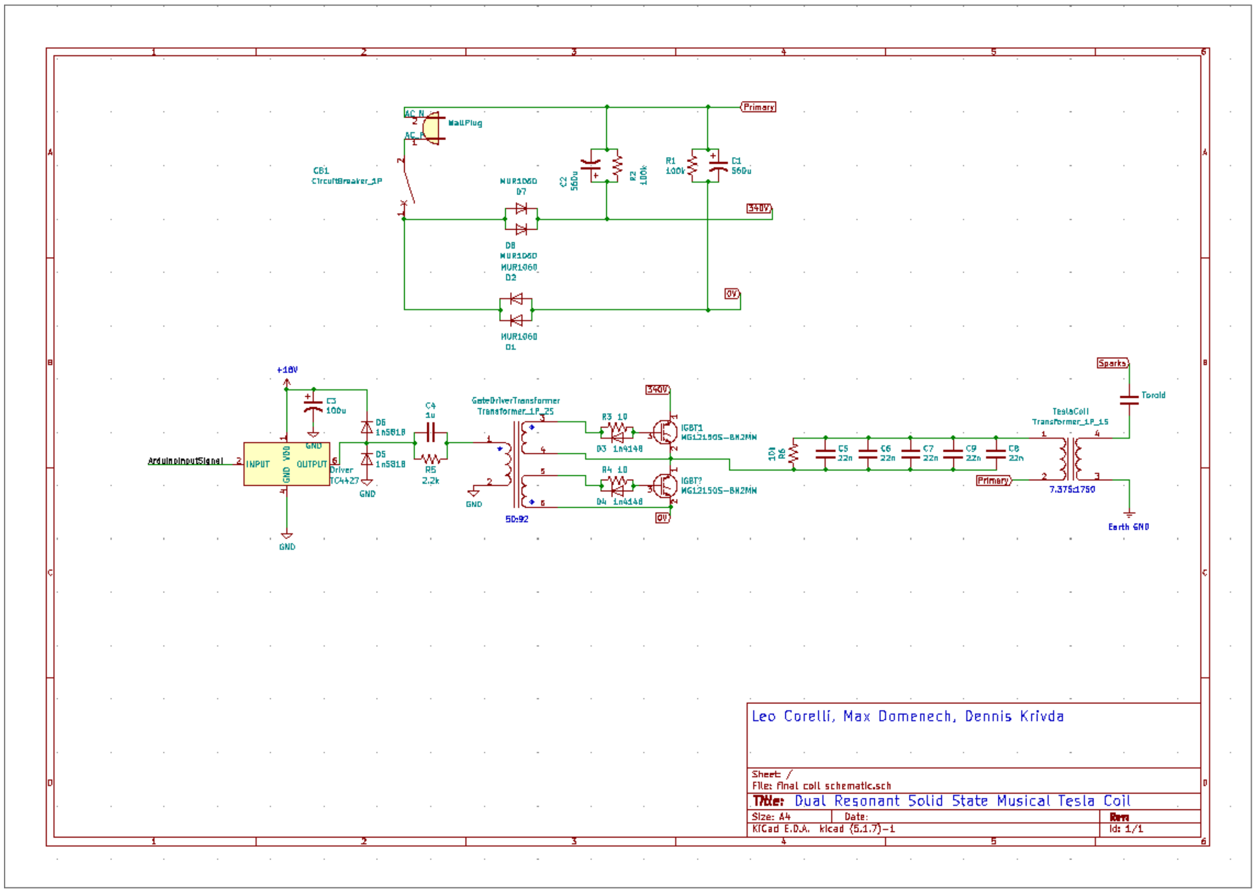
The Primary LC is a series LC circuit with a high voltage capacitor bank serving as the capacitor, and a ¼ inch copper tubing spiral with a few turns as the inductor. The inductor serves as the primary coil of a transformer, with the secondary coil serving as the secondary. The number of turns of the primary spiral is tapped to match the resonant frequency of the secondary, which is fixed. The resonant frequency is actually a little bit lower than that of the secondary because the secondary resonant frequency decreases when sparks are made because plasma has a lower impedance than air. It receives an input from the inverter at its resonant frequency and oscillates in resonance building up energy to transfer to the secondary.

f. Secondary Coil

The secondary coil (Tesla coil) is another series LC circuit with a thin, 30 gauge magnet wire wrapped 1750 times in a coil as the inductor, which is grounded on one side and connected to a capacitor on the other. An aluminium toroid top load serves as one plate of a capacitor and earth ground as the other. The inductor is the secondary coil of the transformer made with the Primary LC. The secondary LC is a very high voltage and low current system. The circuit oscillates in resonance with the primary, as the primary builds up energy and transfers it over. Once enough charge has built up on the toroid, the potential will overcome the air gap and flow through the air to ground through the breakout point in the form of sparks.

4. Hardware:

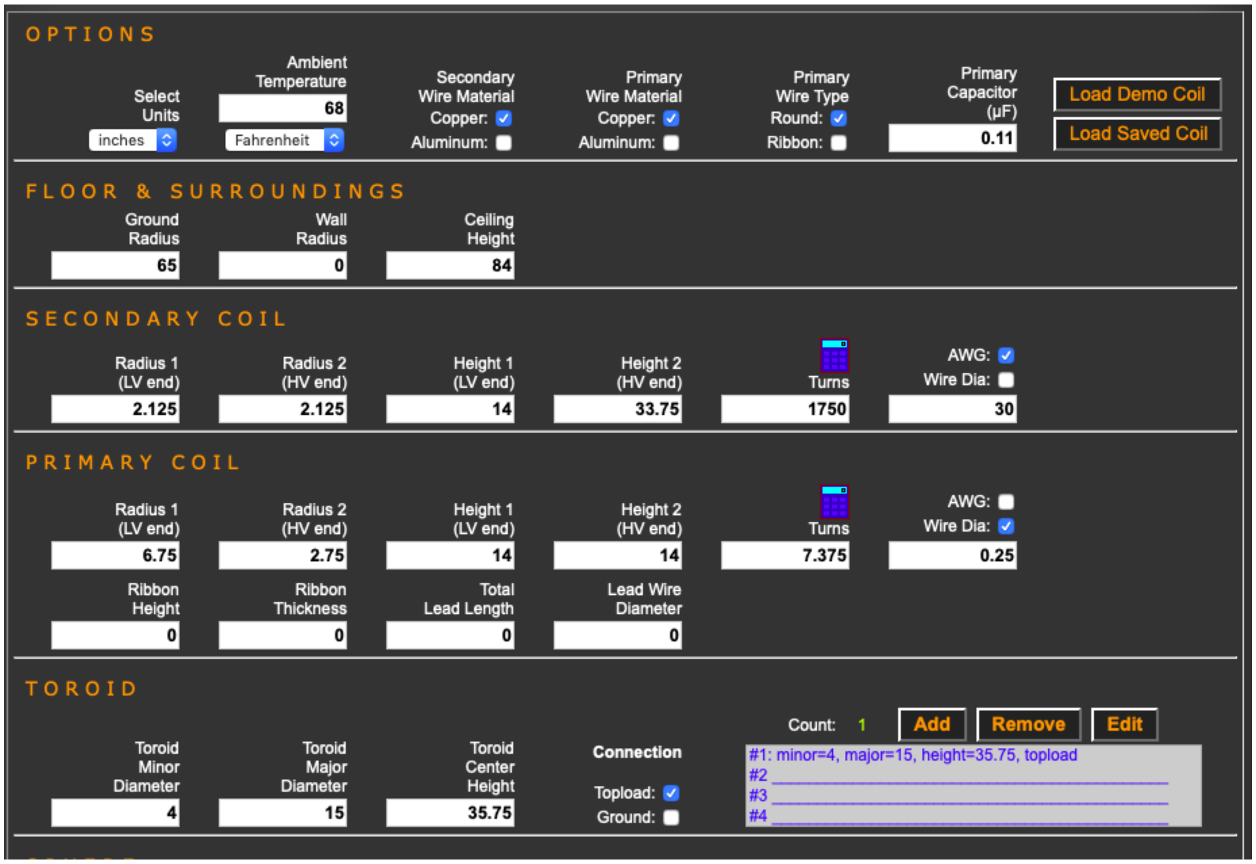
a. Overall Circuit Schematic:



b. Secondary Coil and Top Load:

i. Design/Simulation

We used the JavaTC website tesla coil calculator to plan our coil. Beforehand, we knew we wanted our coil to have a resonant frequency between 120kHz and 150kHz, and we wanted it to be a visually pleasing size, which to us meant approximately 2 feet from the bottom of the coil to the top of the top load. We also knew that a bigger secondary meant a larger voltage on it and bigger sparks, and a lower resonant frequency also meant bigger sparks. So, then we worked with Java TC until we got a reasonable design. We also wanted toroid top load, the most common Tesla coil top load. After construction, the height of the coil was slightly larger than calculated due to human error, so we recaulated with our new height and got this:



OPTIONS

Select Units	Ambient Temperature	Secondary Wire Material	Primary Wire Material	Primary Wire Type	Primary Capacitor (µF)	
inches	68 Fahrenheit	Copper: ✓ Aluminum: ☐	Copper: ✓ Aluminum: ☐	Round: ✓ Ribbon: ☐	0.11	Load Demo Coil / Load Saved Coil

FLOOR & SURROUNDINGS

Ground Radius	Wall Radius	Ceiling Height
65	0	84

SECONDARY COIL

Radius 1 (LV end)	Radius 2 (HV end)	Height 1 (LV end)	Height 2 (HV end)	Turns	AWG: ✓ Wire Dia: ☐
2.125	2.125	14	33.75	1750	30

PRIMARY COIL

Radius 1 (LV end)	Radius 2 (HV end)	Height 1 (LV end)	Height 2 (HV end)	Turns	AWG: ☐ Wire Dia: ✓
6.75	2.75	14	14	7.375	0.25

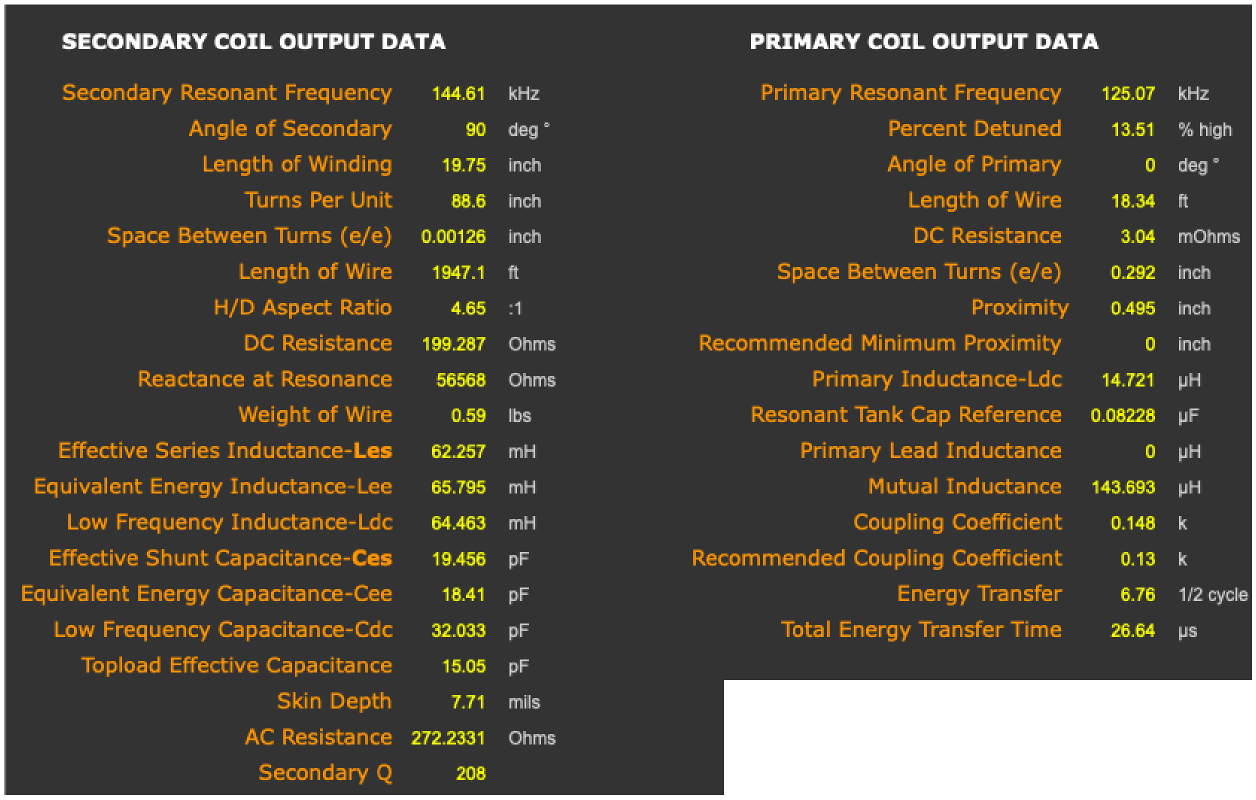
Ribbon Height	Ribbon Thickness	Total Lead Length	Lead Wire Diameter
0	0	0	0

TOROID

Toroid Minor Diameter	Toroid Major Diameter	Toroid Center Height	Connection
4	15	35.75	Topload: ✓ Ground: ☐

Count: 1 Add Remove Edit

#1: minor=4, major=15, height=35.75, topload
#2
#3
#4

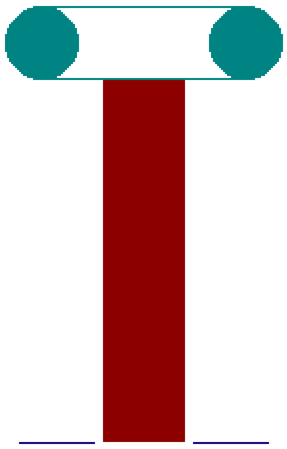


SECONDARY COIL OUTPUT DATA

Secondary Resonant Frequency	144.61	kHz
Angle of Secondary	90	deg °
Length of Winding	19.75	inch
Turns Per Unit	88.6	inch
Space Between Turns (e/e)	0.00126	inch
Length of Wire	1947.1	ft
H/D Aspect Ratio	4.65	:1
DC Resistance	199.287	Ohms
Reactance at Resonance	56568	Ohms
Weight of Wire	0.59	lbs
Effective Series Inductance-**Les**	62.257	mH
Equivalent Energy Inductance-Lee	65.795	mH
Low Frequency Inductance-Ldc	64.463	mH
Effective Shunt Capacitance-**Ces**	19.456	pF
Equivalent Energy Capacitance-Cee	18.41	pF
Low Frequency Capacitance-Cdc	32.033	pF
Topload Effective Capacitance	15.05	pF
Skin Depth	7.71	mils
AC Resistance	272.2331	Ohms
Secondary Q	208	

PRIMARY COIL OUTPUT DATA

Primary Resonant Frequency	125.07	kHz
Percent Detuned	13.51	% high
Angle of Primary	0	deg °
Length of Wire	18.34	ft
DC Resistance	3.04	mOhms
Space Between Turns (e/e)	0.292	inch
Proximity	0.495	inch
Recommended Minimum Proximity	0	inch
Primary Inductance-Ldc	14.721	µH
Resonant Tank Cap Reference	0.08228	µF
Primary Lead Inductance	0	µH
Mutual Inductance	143.693	µH
Coupling Coefficient	0.148	k
Recommended Coupling Coefficient	0.13	k
Energy Transfer	6.76	1/2 cycle
Total Energy Transfer Time	26.64	µs

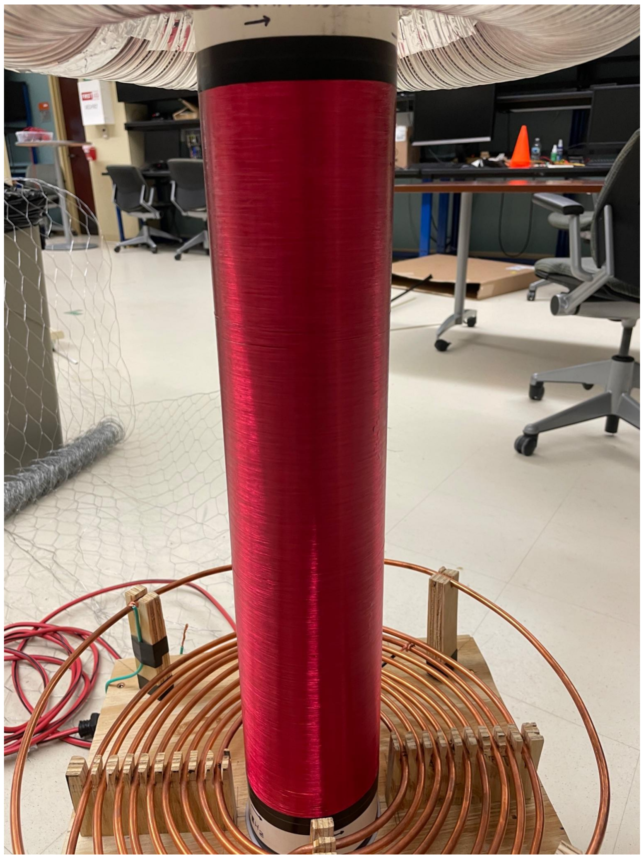


ii. Construction

1. Secondary Coil

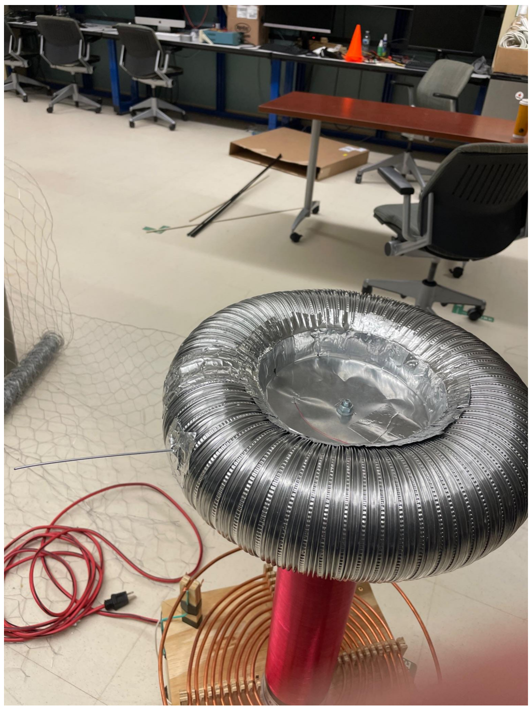
To construct the secondary coil we used a 10 ft by 4 inch diameter PVC pipe, 1 lb of 30 AWG magnet wire, polyurethane varnish, a drill, some screws, and a small piece of a 2 by 4. First, we wiped down and cleaned the PVC pipe. Then, We fastened the 2 by 4 onto one end of the PVC and drilled a screw into it positioned directly in the center of the PVC pipe. We drilled three small holes into the side of the PVC approximately 2 ft up, and threaded the magnet wire through it to secure it. Max was on the end with the 2 by 4 using a positioned in the centered screw to rotate the PVC counter clockwise (meaning the wire turns went clockwise). Dennis was holding the wire spool, guiding the wire with his hand to make sure it was wound snugly with no gaps or overlaps. Leo was on the other end of the PVC giving max a counter force to push into with the drill and keeping the pipe level. All three group members counted the turn number aloud to prevent error. The wire snapped about 100 turns in on the first attempt due to over tension, and we had to start over, but the second attempt worked perfectly. The total number of turns was 1750 exactly.

We applied two generous coats of polyurethane varnish to fix the windings in place and prevent unraveling and also increase the resistance between the turns. We sawed off the excess pipe leaving 2 inches on either end. We wrapped one turn of electrical tape around each end for greater security of the winding and to improve the visual aesthetics. And, we soldered a thicker, more durable wire on the bottom end to run to earth ground.



2. Top Load

The top load was toroid shape made using 4 inch diameter aluminium ducting; a 7 inch diameter aluminium pipe plate; aluminium tape; a piece of aluminium coat hanger; and a steel bolt, hex nut, and 2 washers. The lip of the plate was cut over because it prevented it from fitting in the center of the ducting. The ducting was then wrapped around the pie plate and secured using aluminium tape. We tried to solder it, but we could not get the ducting hot enough, so we used aluminium tape. The coat hanger was cut and inserted into the side of the coat ducting to create a break out point. We then put the bolt through the center of the pie plate and secured it to the center of the PVC pipe end cap with the hex nut. The end cap was fixed on the top end of the PVC pipe, and we wrapped the circumference of the PVC with several layers of electrical tape to make the fit more snug so it will not come off. Finally, we threaded the magnet wire from the end of the coil through the pie plate to the top of the toroid, shaved off a large area of its insulation, and connected it to the pie plate with aluminium tape.



iii. Testing

The final dimension of the secondary LC are as follows:

Secondary coil:
- Height: 19.75 inches
-Diameter: 4.25 inches
-1750 turns

Toroid:
-Outer diameter: 15 inches
-Inner diameter: 7 inches
-Ring diameter: 4 inches
-Center located 2 inches above end of secondary coil

To measure the resonant frequency, a function generator with a 10Vpp sine wave output was connected to the ground wire of the secondary coil, with the function generator's ground connected to the ground of an oscilloscope probe. The frequency of the function was varied until its measured amplitude reached a maximum on the scope. This was only an approximate measurement because the body holding the scope probe changed the resonant frequency, and it was unstable, changing by a few kHz just by moving the probe a small distance or the body holding it. However, it was measured to be between 130-136kHz.

c. <u>Primary LC</u>:

i. Design/Simulation

The primary coil was designed in the same program as the secondary as seen above. It's purpose is to match in resonant frequency to that of the secondary, which is fixed. It is the (relatively) low voltage, high current part of the transformer, and drives the secondary with a voltage step up of a=237.29.

ii. Construction

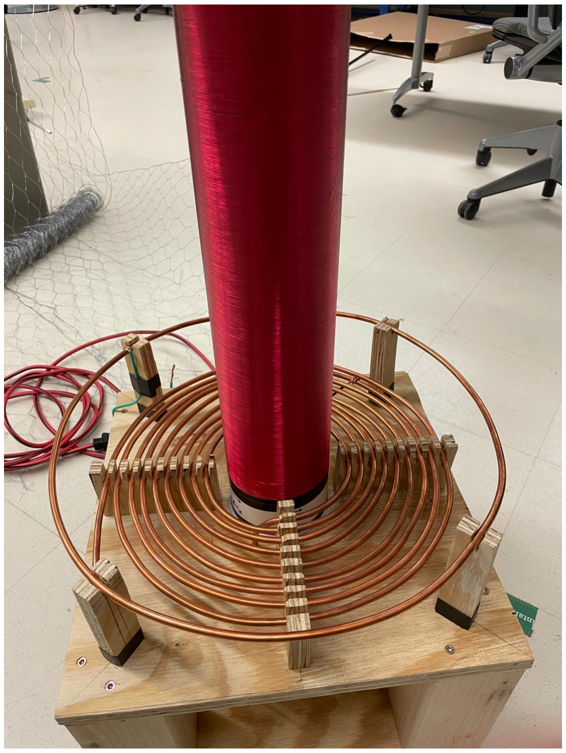
To construct the primary was used 50 ft by ¼ in copper tubing and ¾ in plywood. We cut a 16 in by 16 in square plywood to mount on. We then cut out four 5 in by 2.625 in rectangles of plywood and cut ¼ in wide, 0.625 in deep, notches in them with ¼ in between them. This would align the bottom of the primary coil up exactly with the bottom of the secondary. The first rectangle was placed .375 in out from the secondary, the second .5 in, the third, .625 in, and the fourth .75, so that there was a 0.625 in gap between the innermost point of the spiral and secondary. We then bent the tubing in a spiral rotating clockwise starting from the inner end, finishing and cutting at 8.75 turns. The inner end of the spiral bent down through a hole cut in the plywood into the area underneath the primary and tesla coil where the circuitry was. We originally drilled holes at the desired height and spacing and attempted to thread the tubing through it, but the friction was too great, we barely got more than a turn.

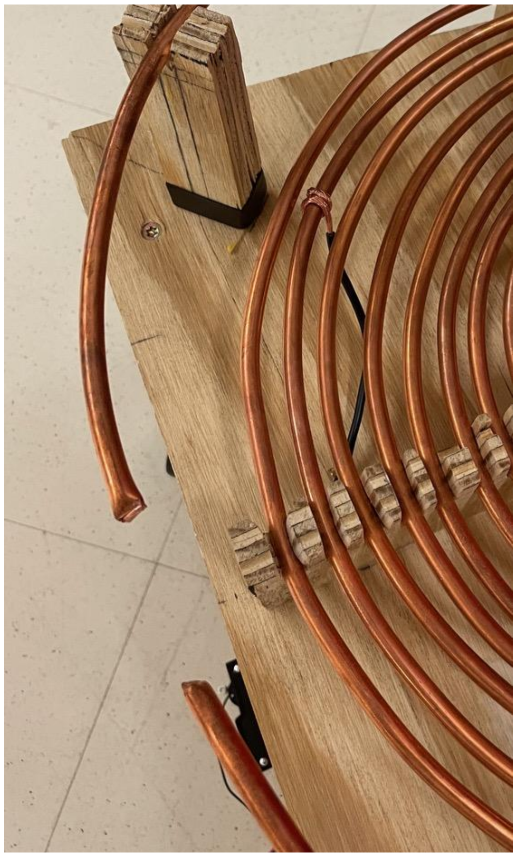
The coil positive end of the spiral would be the inner part. The negative end was terminated by a 14 AWG insulated wire with an exposed that was wrapped around the tubing at the desired number of turns, and was threaded back through a hole in the plywood base. This was done so the number of turns could be adjusted later on.

To create the primary capacitor, five 22nF 2kV polypropylene film capacitors (datasheet) were wired in parallel with each other and a 10kR 7W resistor (to discharge them while not in operation) on a solder breadboard. One end of the capacitor back was placed inside the copper tubing from the inside of the primary and crimped. The other end was soldered to 14 AWG wire and the connection was insulated with heat shrink tubing.

A single strike ring was created through the same process, 2 in greater in diameter than the outermost turn of the spiral, and 2 in higher. This was sent to earth ground through a 14 AWG wire wrapped in the same manner. It had a 1.5 in disconnect in the ring to prevent current flow around the ring and eddy currents induced by the transformer. This served to protect the primary coil and circuitry underneath from spark strikes.

We then created a second plywood platform of the same dimension 8 in below to mount the circuitry on, with 4 by 4s in each corner connecting them, and a wheel in each corner as the final support on the ground.


Primary Spiral


Wire tap and strike ring gap

iii. Testing

No tests were done on the primary, as it would be adjusted as needed via tuning later on; however, given the the series capacitance of 110nF and an approximate resonant frequency of 131kHz, its inductance can be calculated to be 13.42mH, using the formula:

$$f = \frac{1}{2\pi * \sqrt{L * C}}$$

It's final dimensions were:
-Inner radius: 2.75 in
-Outer radius at 8.75 turns (max): 7.125 in
-Outer radius are 7.375 turns (where it was tapped): 6.625 in

The bottom of both the primary and secondary coils was 14 in above the ground.

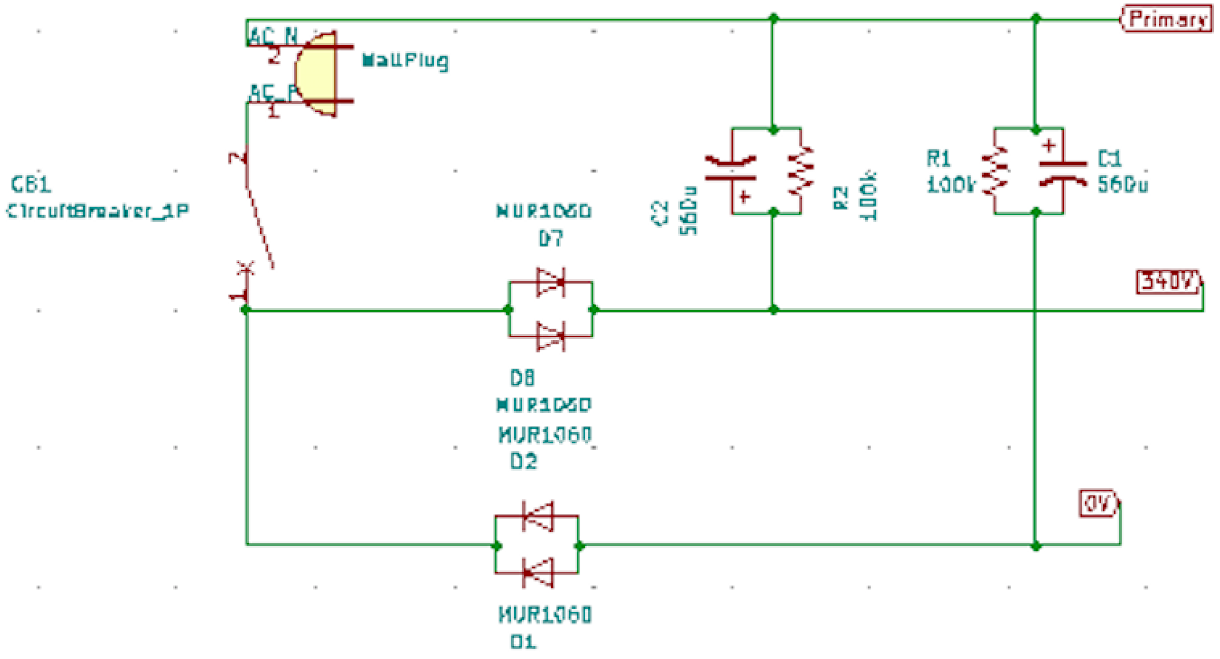
d. Voltage Doubler and Primary Circuitry:

i. Design/Simulation

The purpose of this circuit is to take the square wave input from the driver and the 120Vpp, 60Hz output from the wall and provide a high voltage input to the primary LC at its resonant frequency. The Voltage doubler provides power, while the half bridge inverter switches that power over the primary LC.

1. Voltage Doubler

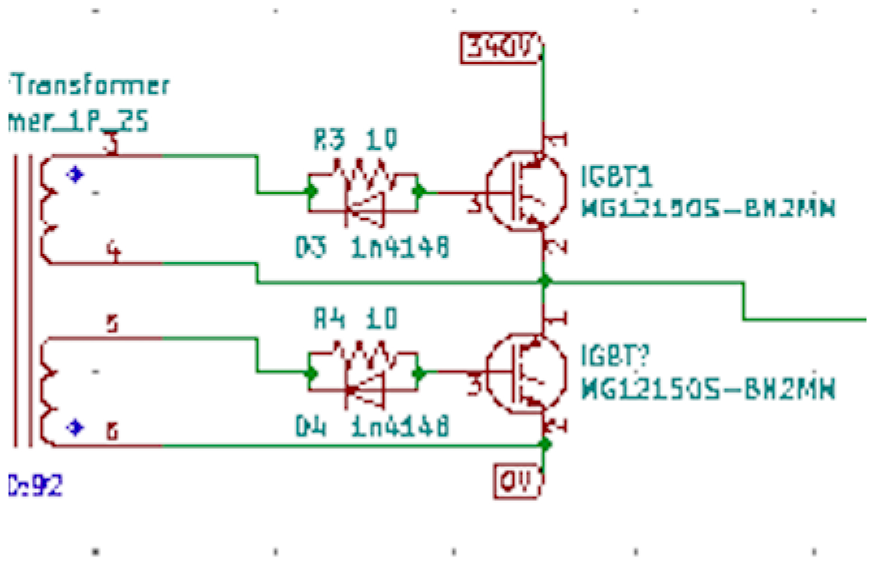
The voltage doubler receives the hot and neutral lines from the wall as input an. The neutral node is connected to the low side of the primary LC, the positive terminal of the lower capacitor, and the negative terminal of the upper capacitor. The hot side of the wall connects through a 10A 120V circuit breaker (9926251010 datasheet) through two sets of diodes to the top and bottom terminals of the half bridge inverter. It is forward biased through two parallel MUR1060 diodes (datasheet), rated for 10 A and 600V, thus allowing 20 A at 600V, to the high end of the inverter. This is in parallel with a high voltage capacitor, which is subsequently charged on the positive part of the 60Hz input and discharges into the node on the negative part providing the node with a constant voltage of 170V Dc (minus the 1.6V drop over the diodes). The same setup is situated on the low end of the inverter, providing it with -170V DC. Thus the Half Bridge inverter experiences a 340V potential difference over it. The two capacitors are 560uF, 450V DC Electrolytics (datasheet), and they are each in parallel with a 100kR 3W bleeder resistor to discharge them when the coil is not in operation. The specific capacitance does not particularly matter, as long as they are sufficiently large.



2. Half Bridge Inverter

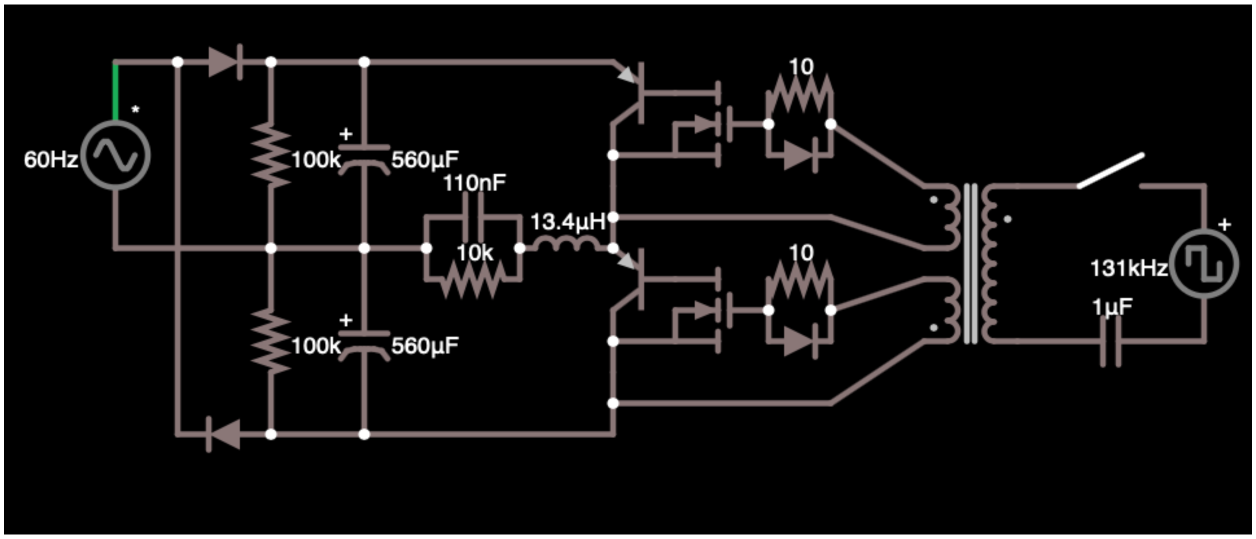
The Half Bridge Inverter is simply a high performance 1200V, 150A, 625W at 15Vge IGBT module (MG12150S-BN2MM) (datasheet). This module is two nIGBTs, with the high voltage end being the collector on the top IGBT, the emitter of the top connected to the collector of the bottom one, and the emitter of the bottom one being the low voltage end. P-type IGBTs are not used in practice because they have inferior characteristics. There are also internal fly back diodes between the emitter and collector of each. The node between the two is the output which connects to the capacitor bank in the primary LC. The top IGBT receives the Driver output, and the bottom one receives an inverter form. This way the top one will be on, while the bottom is off and vice versa, allowing it to oscillate voltage over the output in the form of a 340Vpp square wave.

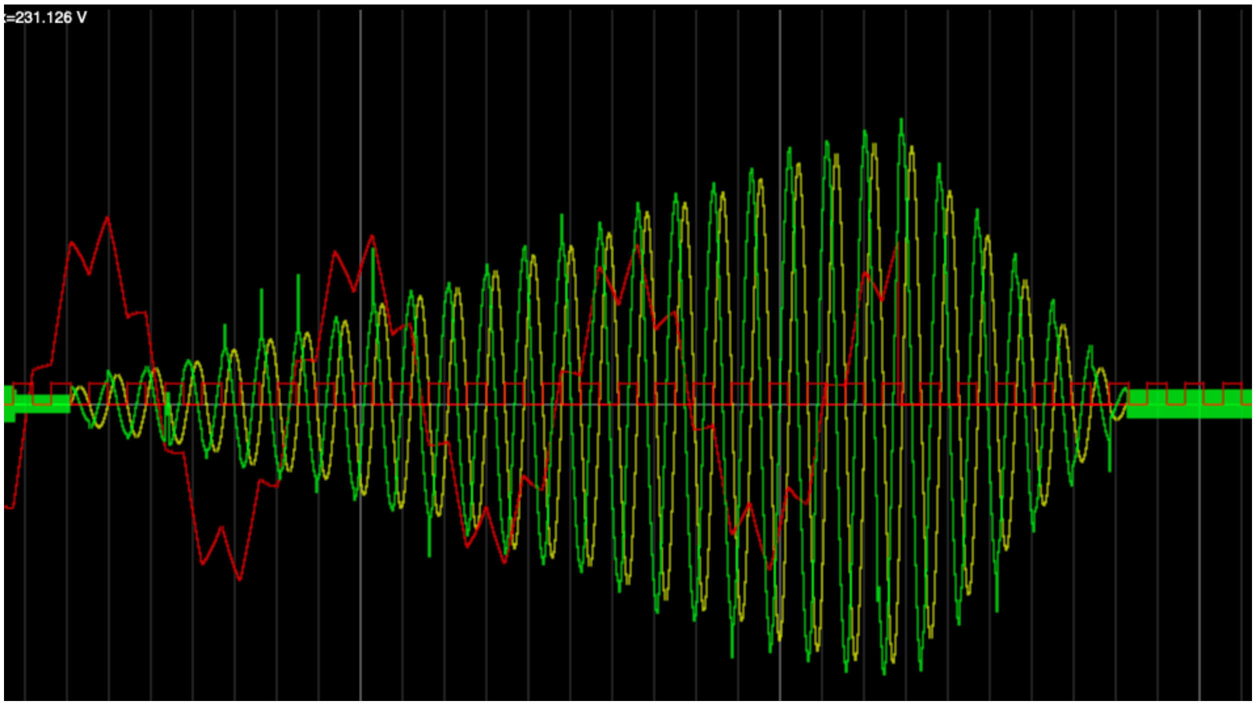
IGBT stands for Insulated-Gate Bipolar Transistor. In practice, they are like a high power BJT, with a MOSFET gate instead of a base. This allows it to have both the gate-drive characteristics of a MOSFET and the low-saturation-voltage and high-current characteristics of a BJT. Because of this and the high gate voltage required to drive (we used ~18V to allow a higher current and faster switching) the gates, a relatively large charge would be stored on the gate, which slows down switching because of the discharge time, so a 10R resistor and switching diode 1n4148 (datasheet) were placed in parallel on the gates to discharge the gates for faster switching. IGBTs can be modeled with the base of a PNP connected to the Drain of an nMOSFET and the source and emitter connected, which is what we did in our simulation. In the half bridge arrangement used in our project, it is common practice to charge the gates with an isolation transformer to prevent a floating GND on the top IGBT and galvanically isolate the input circuitry.



3. Simulation

We originally intended to simulate the system using LTspice; however, we could not find any suitable IGBTs, so we changed to the PNP-nMOS model of the IGBTs, but could not find PNPs or nMOSs that could handle the power. So, we ended up using Falstad to do our simulations because we knew how to declare our PNP and nMOS to have arbitrary large power characteristics. This was not a problem because we knew that our IGBT module would not be the limiting factor in the power that our circuit could handle. I could find a way to pulse the output, so I would just hold the switch for the desired length of time and release it. Here is the circuit tested and the simulation waveform:

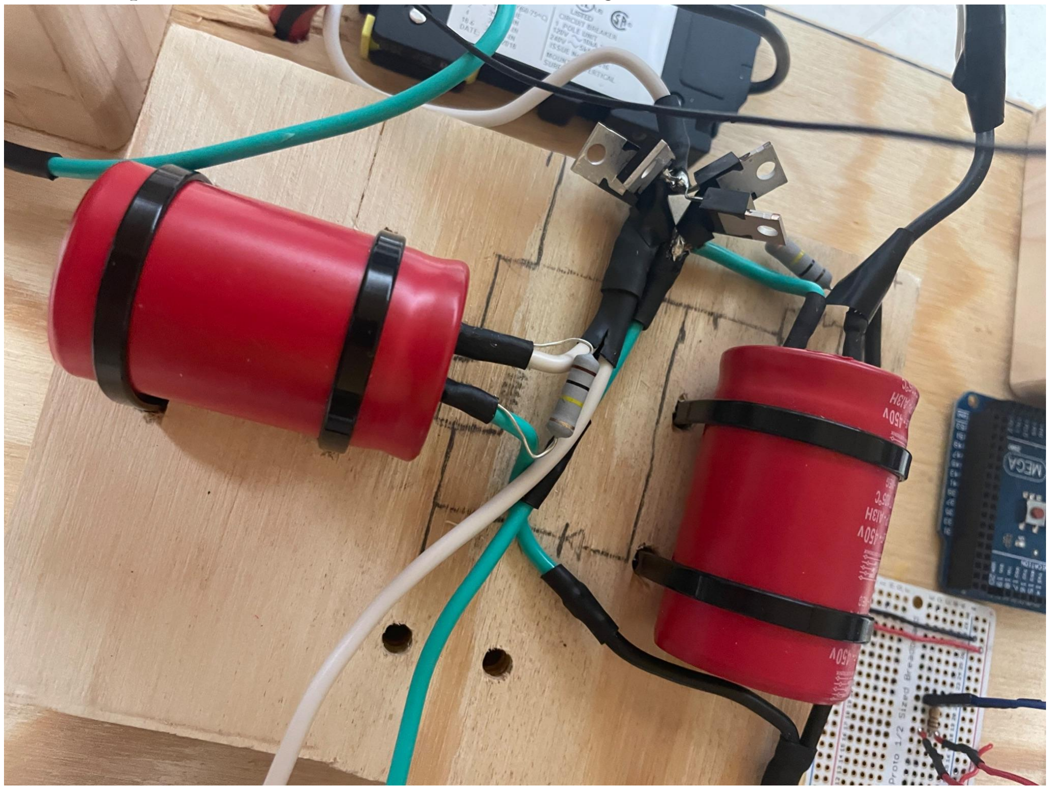




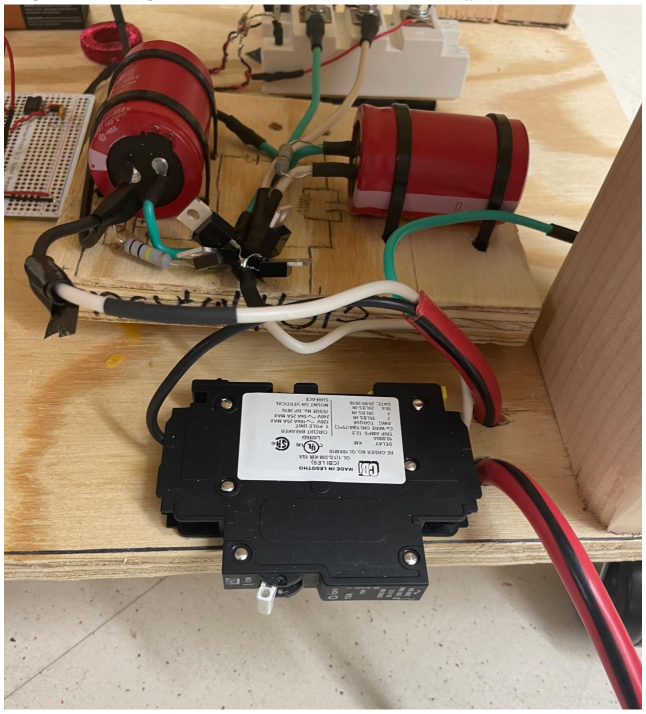
In this simulation of a single pulse, the primary current grows in resonance for duration of pulse, then rapidly degrades upon completion. Power transfers to secondary until enough charge is built up on secondary capacitor (toroid) to overcome air gap and transfer to ground in the form of a single spark. This simulation behaved exactly as intended, so we will begin construction.

ii. Construction

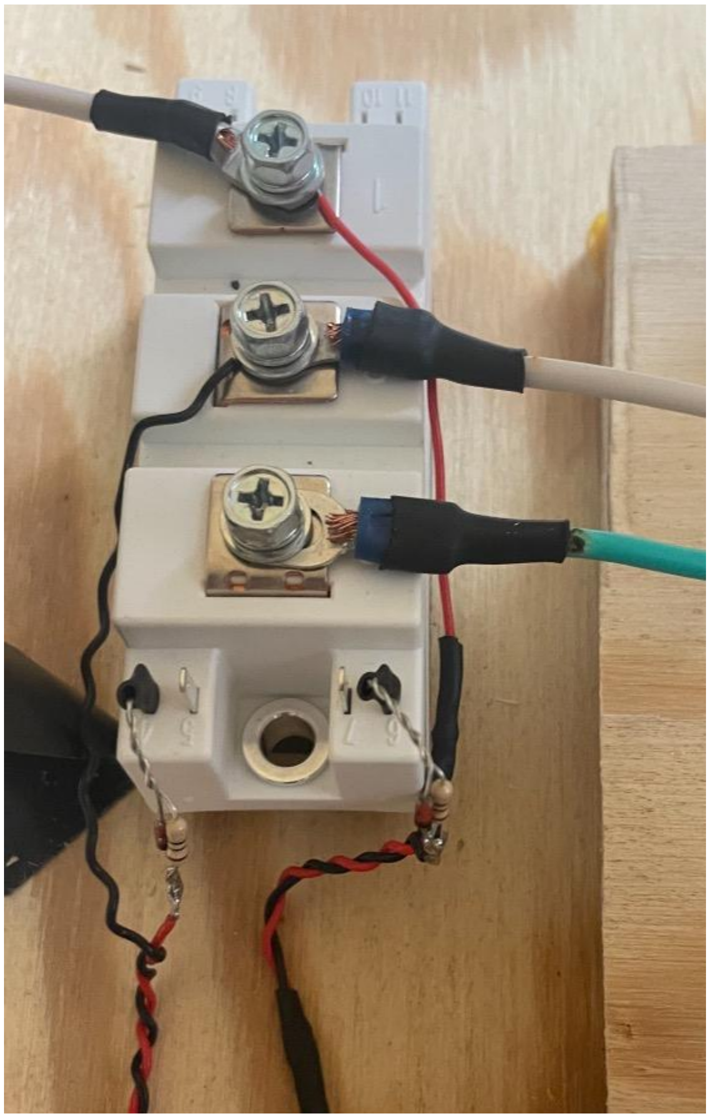
For the Voltage Doubler, we mounted the capacitors on a small square of plywood and then drilled holes through it and zip-tied them down. We then soldered the resistors to the capacitors and connected the diodes, using 14 AWG wire for all connections, and soldered all connections and insulated those connections with heat shrink tubing. The only parts physically secured to the board however, are the capacitors. The breaker is connected using its built in screw terminals.



The power was connected to the circuit using an extension cord to plug into the wall, and cutting off the other end to solder the hot and neutral wires to the appropriate nodes. The extension cord was secured by weaving it through two holes drilled in the plywood base.

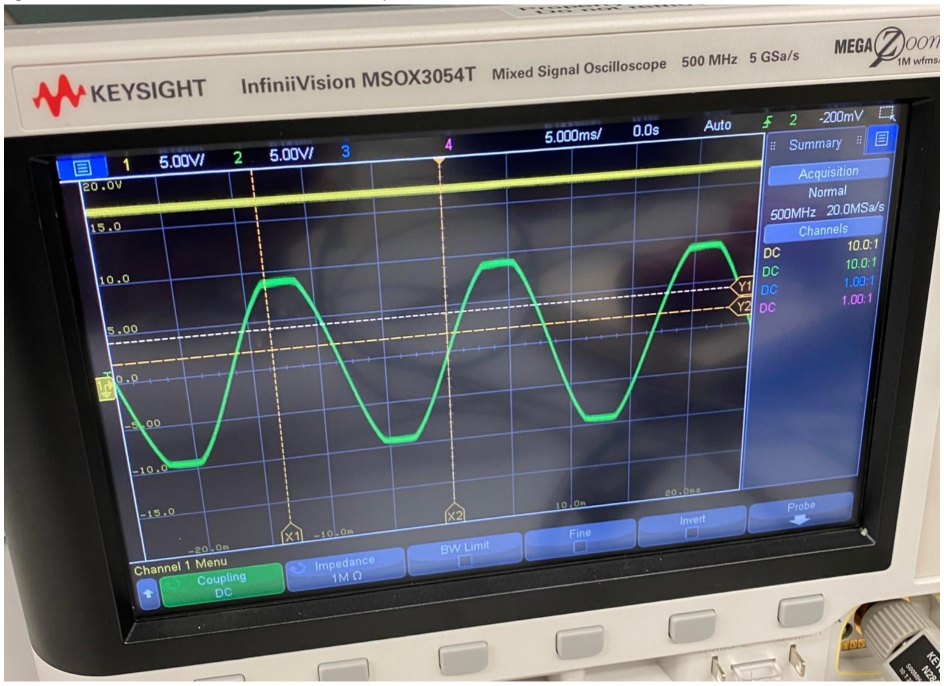


The IGBT module was mounted on 8 heat sinks (datasheet) and was connected as shown in the schematic using crimp screw terminals. Terminal 3 is the high side, 2 is the low side, 1 is the output to the primary LC, 4 is the high transistor gate, and 6 is the low transistor gate. On the gates, the 10R resistor and diode were twisted in parallel and soldered and onto the terminals and insulated with heat shrink tubing.

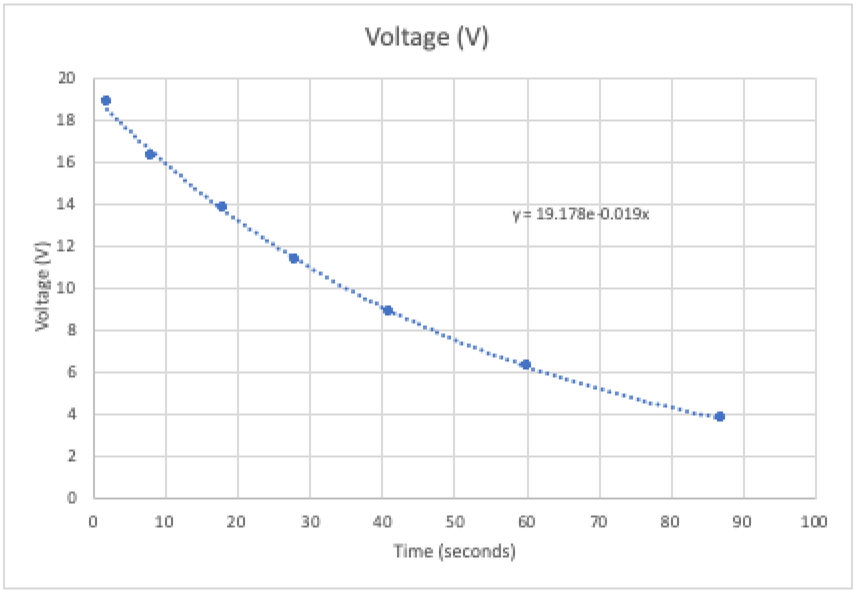


iii. Testing

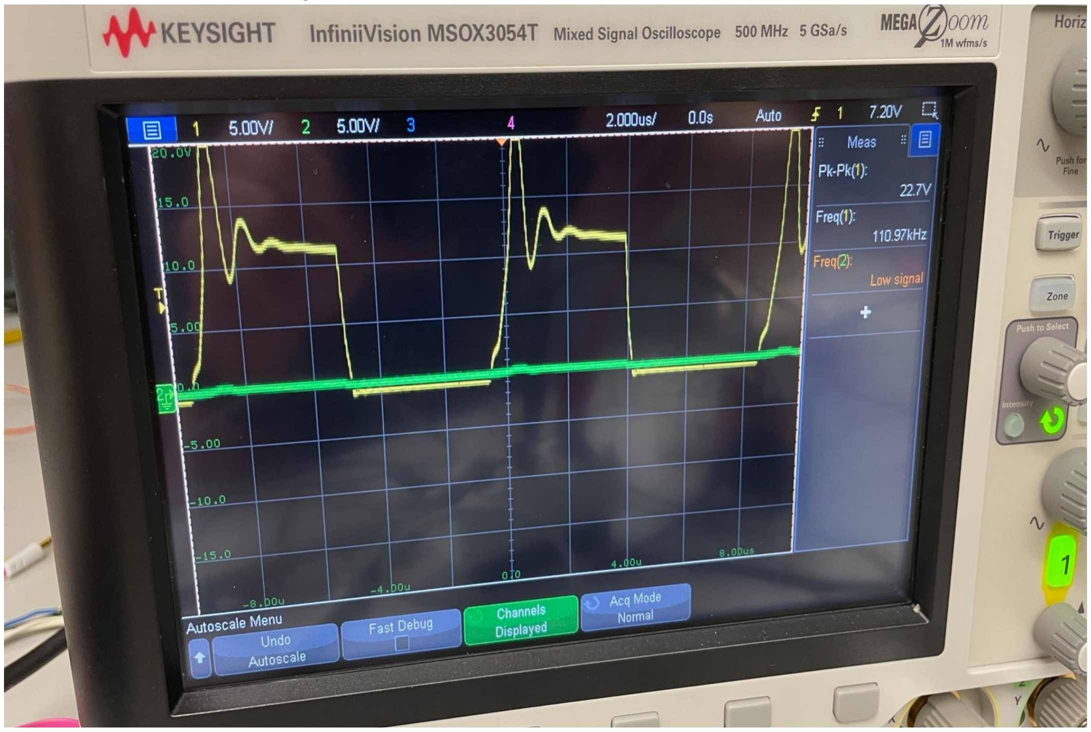
For safety, we experimentally determined how long it takes for the capacitors to discharge after the circuit is disconnected because they can source lethal amounts of current for long after the power is disconnected. To do this, we charged up the capacitors to 18.8V (10Vpp, 60Hz sine wave from the function generator, doubled, minus the voltage drop over the diodes), and recorded the time every time the voltage dropped 2.5V. We then plotted these points in excel and found the exponential trend line, giving us $\tau = 0.019$. We then used this to find how long it would take to discharge from 340V to 5V, which is 3 minutes and 42 seconds. So, we declared that after power was disconnected, we would wait 5 minutes before handling the circuit to air on the side of safety. This test simultaneously proved that our voltage doubler functioned correctly.



time (sec)	Voltage (V)
2	18.8
8	16.3
18	13.8
28	11.3
41	8.8
60	6.3
87	3.8



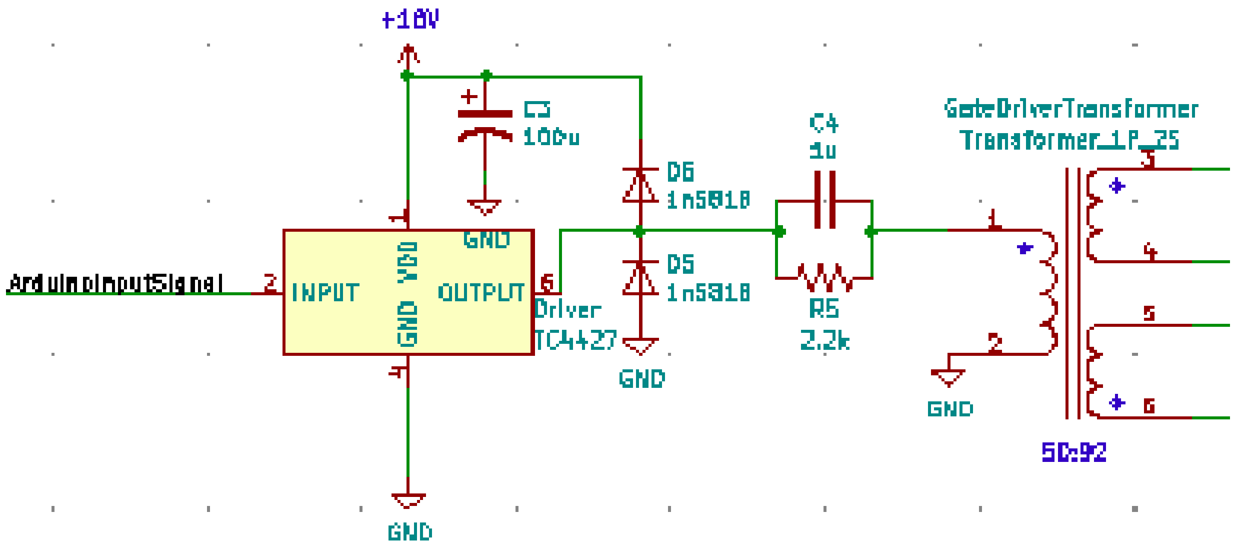
We tested the IGBTs by placing a 10kR resistor between terminals 1 and 2 (the output and low side), applying 30V DC and 700mA over terminal 3 and 2 (high and low side), and applying the output from the driver at each gate.



The output is a square wave at the desired frequency. All of the ripples are the result of parasitic stray inductance in the Driver circuit (the biggest issue in this project and hardest problem to combat), which was addressed later. But, those ripple do massively decrease spark length and performance.

e. <u>Gate-Drive Circuitry</u>:

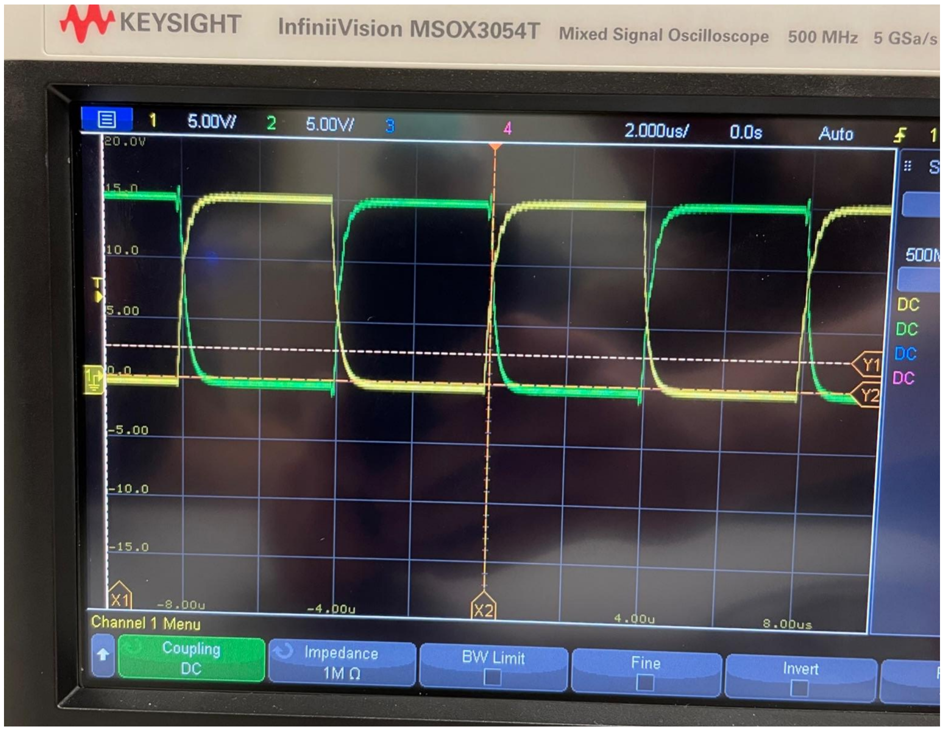
The Gate-Drive Circuitry receives the output of the Arduino and amplifies it up to 18V then outputs it to the gates of the inverter through the gate drive transformer. The two key components of the circuit are the MOSFET driver and the gate drive transformer.



i. MOSFET Driver

The Driver receives an input signal from the Arduino and an 18V VDD--two 9V batteries in series with a switch to turn them on and off. The advantage of MOSFET drivers is their very high switching speed at relatively high voltages (15-40V), which make them ideal for our high frequency 18V square wave, as opposed to the other amplifiers we looked at which either cannot provide a high enough voltage or fast enough switching speeds. The drivers we originally used were UCC27425 (datasheet); however their performance at our desired frequency was not perfect (see below), and they could only drive up to 15V, not 18V.

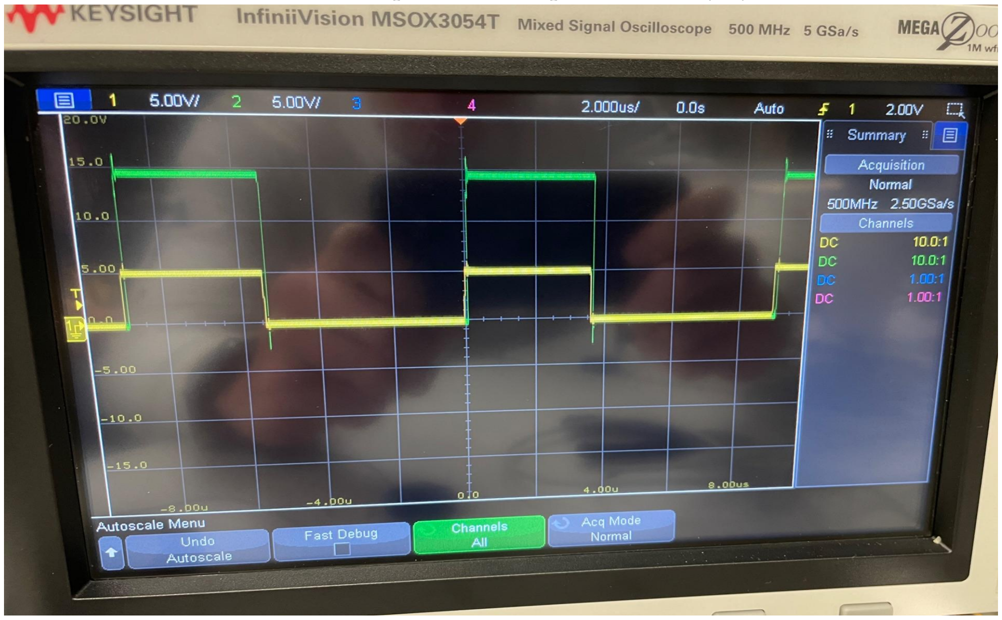
Output at 15V VDD, and a 5V, 125kHz square wave input:


KEYSIGHT
InfiniiVision MSOX3054T Mixed Signal Oscilloscope 500 MHz 5 GSa/s
1 5.00V/ 2 5.00V/ 3 4 2.000us/ 0.0s Auto
Channel 1 Menu
Coupling DC
Impedance 1M Ω
BW Limit
Fine
Invert

They were also surface mount, not through hole mounting style, which made it a challenge to solder to and bread board with and test. One was broken while trying to solder to it, and the other was destroyed by putting too much current through.

Once these broke, we switched to a different type of MOSFET driver: IR4427PBF (datasheet). This can drive up to 20V, is through hole mounted, and can produce a nearly perfect square wave at our desired frequency.

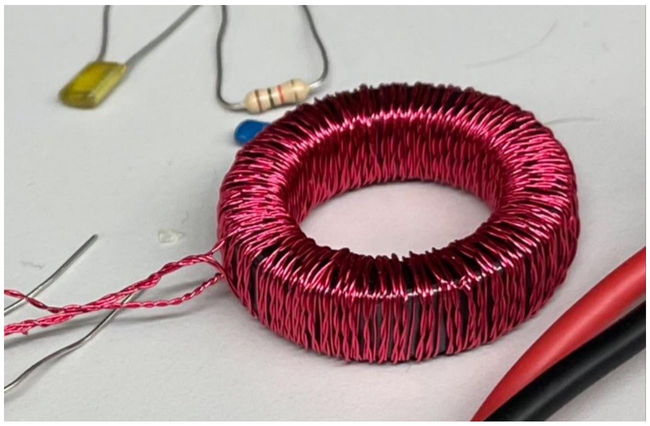
Output at 15V VDD, and a 5V, 111kHz square wave input, 40% duty cycle:



ii. Gate Driver Transformer (GDT)

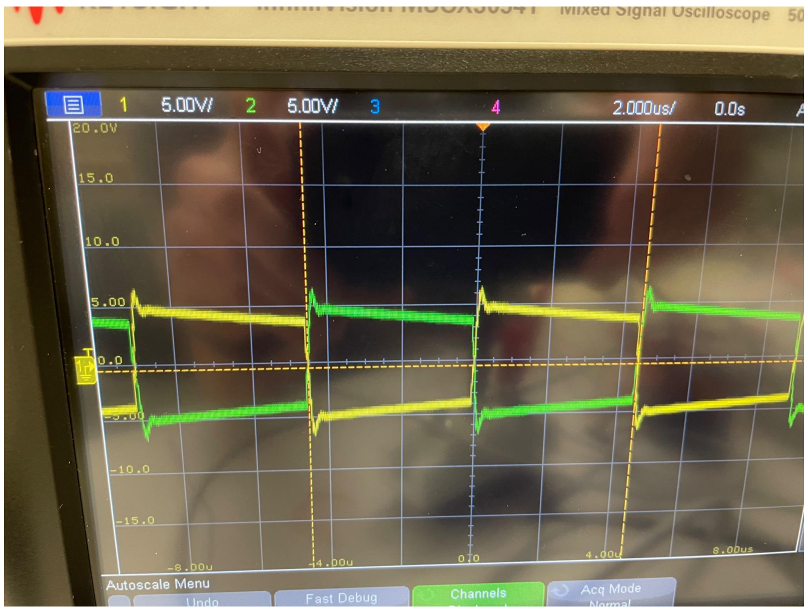
The gate driver transformer receives the 18V squarewave output from the MOSFET Driver and applies it to the gates of the IGBTs. It has one primary coil that is connected to the driver output on the positive side and ground on the negative side and two secondary coils, one with a positive side soldered to node 4 (the gate of the upper IGBT) through the diode and resistor and the negative side on node 1 screw terminal (the connection between the emitter of the upper IGBT and collector of the lower IGBT) and the other coil with the positive side on node 2 screw terminal (the low end of the inverter) and the negative side soldered to node 6 (the gate of the lower IGBT) through the diode and resistor, thus inverting the square wave. The transformer serves to galvanically isolate the low power driver circuit and arduino from the high power primary circuit to protect them. It also gets rid of the floating ground that will exist on node 1, providing better and safer performance of the Half Bridge Inverter.

The transformer needs to be able to receive the input square wave and output a waveform that looks reasonably square at the desired frequency. This can be done by increasing the number of turns, increasing the voltage it can handle before getting magnetically saturated. Unfortunately, we did not know much about how to choose a proper ferrite core, or that it really matters that much (it does!), so our core (DigiKey page) was not perfect for the job, but we did not have time to order a new one. It has some pretty serious stray inductance issues when connected with the driver (which we will see later), and some significant losses across. However, after several trials we came to the model we ended up using in our final build.



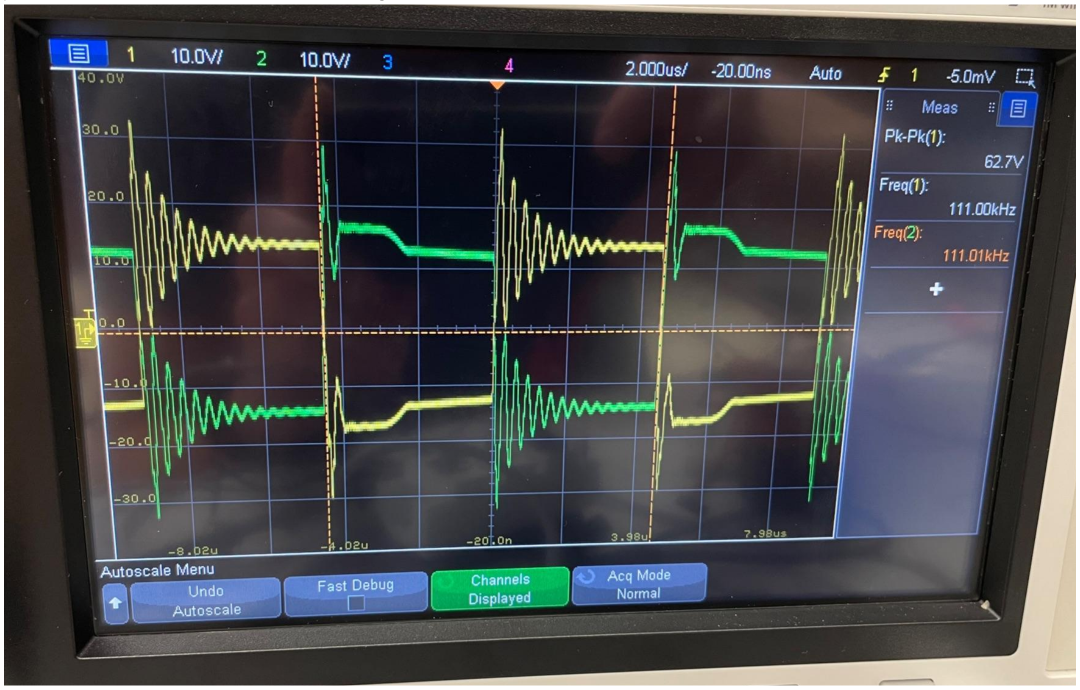
It was made using 30 AWG magnet wire (the same as the secondary coil), with 50 turns on the primary and 92 turns on the secondarys (a=1.84). The primary was individually wound and the two secondary wires were twisted together and then wound. The voltage step up was just enough to make up for the losses in the transformer. A parallel combination of 1uF non polarized capacitor and 2.2kR damping resistor were put between the driver output and GDT: The capacitor to block DC voltages and the resistor to damp the small voltage spike after the rising and falling of each edge of the squarewave.

Here is the output at 5V, 111kHz square wave input:



iii. Combating Stray Inductance

To demonstrate the significance of stray inductance, here is the output of the GDT when the entire system is connected and running:



It barely looks like a square wave and it reduces the spark size to around 1cm.

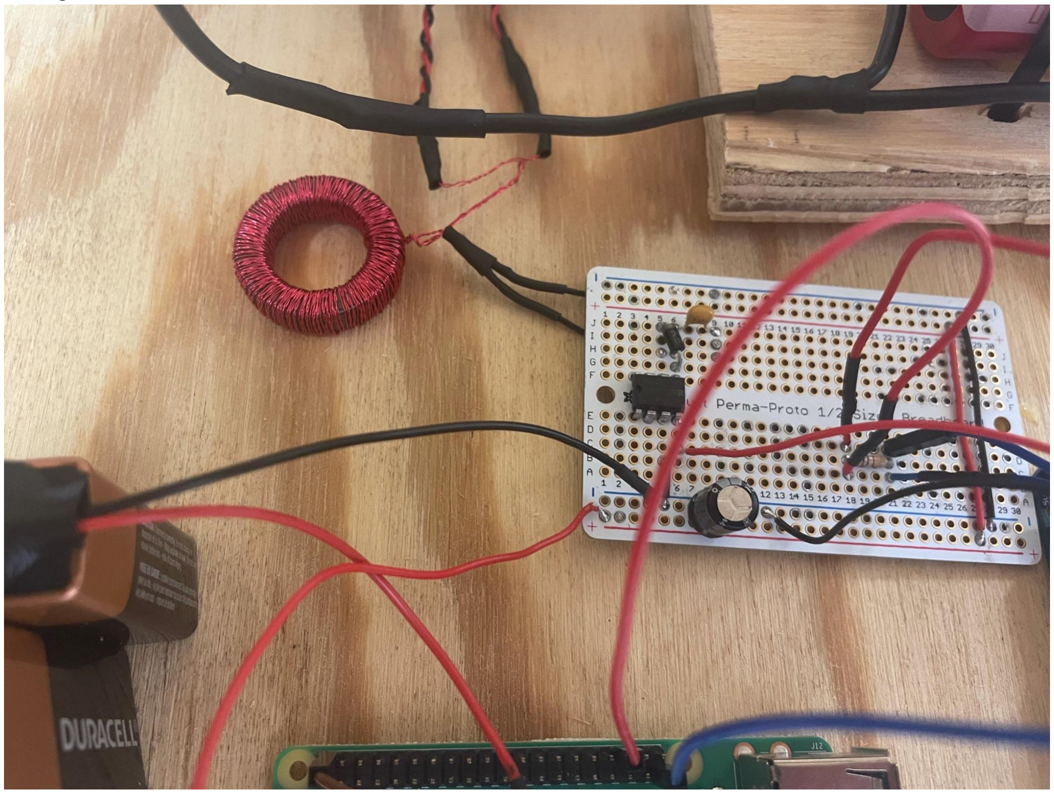
The first thing we did to combat this was place a 100uF 35V polarized capacitor between the 18V source and ground. This decoupling capacitor shunts noise caused by other circuit elements to ground, and helps to reduce the effect of stray inductance. Added this increased the spark length to about 3 inches.

The next step we took to reduce the effects of stray inductance was to place two clipping diodes

(1n5818 datasheet) on the output of the driver before the capacitor and resistor. One going from ground to the output and the other going from the output to the voltage source. These would clip any large voltage spikes (greater than the forward voltage of the diode (~0.33V)) on the output signal by shorting them to either power or ground through the diode. This increased spark length by about 1 inch.

Our final measure to combat the stray inductance itself, and not merely its effects. Every wire in the circuit acts as an inductor and thus contributes to the stray inductance problem. To decrease this effect, we reoriented and soldered the driver circuit, placing everything as close together as possible to reduce any excess wire, shortening any wires that could be shortened, and twisting pairs of wire together to cancel their inductances. This doubled the spark length, with the largest sparks reaching up to 1 foot in length.

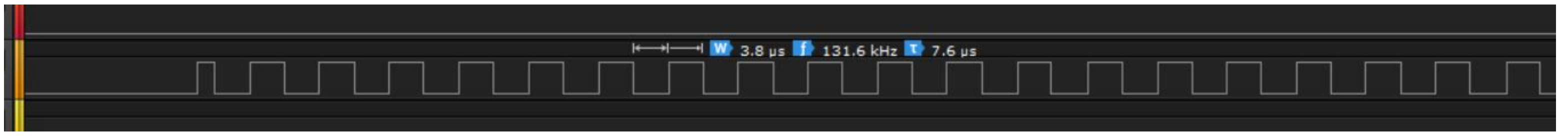
Other arrangements of capacitors were tested, but to little effect, so they were not included in the final design.



Here is the GDT and driver circuit. Many components are on the bottom of the solder breadboard in order to make them as close together as possible. The batteries can be seen in the lower left corner. The Arduino and Raspberry pi resistor to ground connections are also on the solder breadboard on the right hand side.

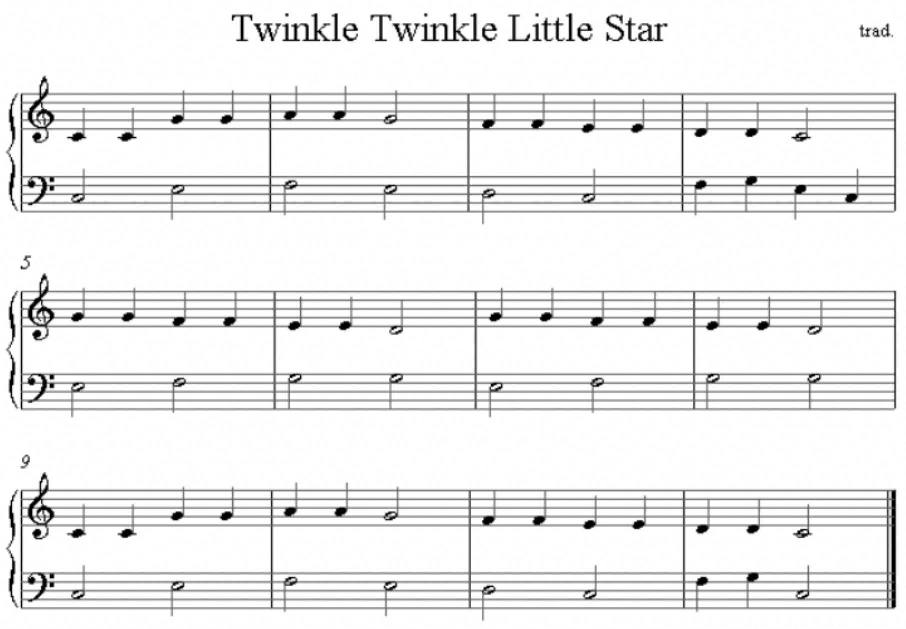
5. Software:

Our programming was written in C in the Arduino Integrated Development Environment (IDE) and in Geany for the raspberry pi python script . Since the necessary nominal frequency of our signal was the resonant frequency of our coil, 131 kHz, we needed to output a clean square wave at this frequency. The Arduino has libraries for square wave output, but they are all software based counting scripts that do not have the necessary level of precision or speed for our application. These library functions were maxing out at 2^{16}-1 Hz, or 65,535 Hz, which was not enough. Instead, we used hardware programming. We accessed the timers of the Arduino mega 2560, which are based off of the system's internal 16 MHz clock frequency. By using hardware programming and accessing individual registers within the mega's processor we were able to achieve a very clean square wave output at the 131 kHz frequency we needed.



Since registers are within the processor of the device, and because the mega's processor runs at 16 MHz, they are very quickly read from and written to. Utilizing them directly in code is the fastest possible method of accessing data.

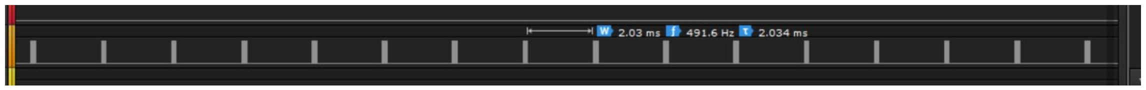
Anything that makes an audible noise produces frequencies, which vibrate the eardrum and sends a signal to the brain. The brain interprets this signal and we can hear. Pure musical notes are pure frequencies (A4 = 440 Hz, C5 = 523 Hz). A song is simply an ordered sequence of frequencies that are played for specific lengths of time.



This is the sheet music for the traditional lullaby, Twinkle Twinkle Little Star. The treble clef denotes the melody of the song, which is the most recognizable part of many songs. The placement of the notes on the treble clef staff encode the frequency information in sheet music. The first line of this song's melody is: C4, C4, G4, G4, A4, A4, G4, F4, F4, E4, E4, D4, D4, C4. Therefore, the first line of the song can be represented by a vector of frequencies like so: {262, 262, 392, 392, 440, 440, 392, 349, 349, 330, 330, 294, 294, 262}. The quarter note (pictured as: ♩) is played for 1 beat, assuming common time where a quarter note has the value of 1 beat. A half note (pictured as: 𝅗𝅥) is played for 2 beats. An eighth note is played for half a beat. These durations are important because they make the rhythm of the song, and the notes' frequencies make the melody. These durations can also be represented in a vector like so: {2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1}, where 2 is a quarter note and 1 is a half note. Due to the relationship between the clock and the registers, the "base" of the system is 120 beats per minute, which corresponds to two beats per second. Since we are writing our music in 4/4, common time (4 beats per measure corresponding to 1 beat per second), we divide each note value by 2. This is why a quarter note in our notearray is 2 (and not 4) and an eighth note is 4 (and not 8) and so on. With both of these vectors, you can successfully transcribe all of the information in the treble clef staff of any sheet music into two vectors that can be entered into our program.

In order to play a song, our program requires two vectors: **int myarray[]** with frequencies (corresponding to musical notes) in the order that we want to play them and **int notearray[]** with note lengths that encode the duration of the note. Therefore, the first entry in our myarray (myarray[0]) is the first note's frequency and the first entry in notearray (notearray[0]) is how long to play that note for. To play the first line of Twinkle Twinkle Little Star our vectors would be as follows:
int myarray[] = {262, 262, 392, 392, 440, 440, 392, 349, 349, 330, 330, 294, 294, 262};
int notearray[] = {2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1};

The output of our program is nominally a 131 kHz square wave that is on for 120 μs and off for a variable amount of time that corresponds to the frequency we want to play. The musical information is actually encoded in the "off" periods of the waveform. B4 has a frequency of 494 Hz, so its period is 1/494 = 2.024 ms. So, we will output the 131 kHz square wave for 120 µs and then turn it off for 2.024 ms. This will create square wave pulses over the course of a second that are separated by 2.024 ms, which corresponds to 494 Hz and therefore the coil will produce 494 which is B4.



We want our coil off for the majority of the time so that our IGBTs are not constantly conducting. The IGBTs get very hot and their performance is maximized with non continuous operation. We found experimentally that a period of more than 150 µs with our coil will degrade the signal and sound bad.

The myarray contains the information that determines the spacing between the pulses sent to the coil (therefore the frequency information) and the notearray contains the information that determines for how many actual seconds this same "on and off" signal is sent to the coil. Naturally, the array lengths must match or the code will not run due to a compiler error.

To accomplish our initial objective of playing 5 songs on our coil, we compiled around 10 different myarrays and notearrays, each corresponding to a particular song. We put these all in our code with the exact same name, and commented out all of the ones that we were not using at a given time. To play a different song in our code all you have to do is uncomment the myarray and the notearray of the song that you want to play and be sure to comment out the myarray and notearray of the previous song you were playing. Since the variable names are the same this will change the information that determines the coil pulse spacing and duration upon reupload to the arduino.

We had to use bitwise comparisons and direct register programming as opposed to conditional logic structures to accomplish the song selection and output due to the precise nature of our program running at such a high frequency. Even using these methods there was still a slight delay (~3µs) that affected the frequency of our notes by several Hz, depending on the note.

Upon upload of the program to the Arduino mega through USB cable, you can unplug the programmer cable and wirelessly initiate the coil operation through a VNC (virtual network computing) connection on the raspberry pi. By downloading the VNC viewer software on the pi and on our laptop, we were able to remotely gain complete control of the pi and run the simple script which outputs a high signal to the analog 4 pin on the arduino, then running the pulse code which fires the tesla coil. It is important to note that in order to establish a VNC connection, one must both manually check the IP address of the raspberry pi displayed in the VNC viewer app on the pi and also ensure that the pi and the laptop being used are on the same wireless network. Once the computer has connected to the pi once on the network, the IP address will not change unless the network goes down. This accomplishes our stated objective of remote control of our coil, which is equally a safety precaution and an impressive technical functionality.

6. Complete Design Testing and Tuning

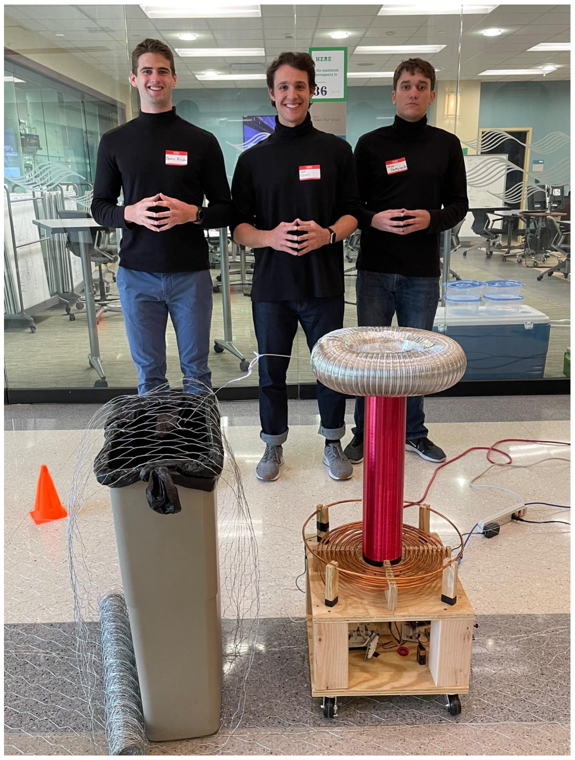
1. Setup:

Not mentioned previously: all components were glued into position with wood glue found in the lab. We did tests with it before using it to ensure it would not ruin our system somehow, to include measuring its resistance with a multimedia (too high to measure).

To set up the system for testing, a few more steps need to be taken. First, the chicken wire needs to be laid out along the floor over an area much greater than that of the coil. This acts as a Faraday cage to block the RF interference emitted by the coil. The earth ground connection from the secondary coil, the strike ring, and the wall outlet extension cord are all floating at this point. They need to be connected to the chicken wire by wrapping their uninsulated ends around some part of it. We always do several tight wraps to ensure a good connection. Then, a ground break out point needs to be placed 6-12 inches from the break out point on the top load. We used more chicken wire that was connected to our ground chicken wire and draped it over a trash can that happened to be the perfect height. Then you plug your computer that you use to place your songs into the Arduino; your computer being a safe distance away (i.e. not on the chicken wire). Lastly, you flip the switch on the batteries to turn them on and then plug the extension cord into the wall (in that order) and you are ready to fire.

Do not go near the system while it is plugged into the wall and for 5 minutes after it is unplugged. The primary coil can source dozens of amps, plenty enough to kill you.

Once it has been unplugged for 5 minutes, flip the switch to turn off the batteries, and it is now shut down.



Our group with the coil setup and operating.

2. Testing and Tuning:

We began testing actually before we had the final version of our driver circuit, but it was good enough to make about 1 cm sparks at this point.

Our first few tests produced nothing. We then remeasured the resonant frequency and found that it was different than we had initially measured weeks before. Initially, we measured 123kHz, hence why some of our test signals were at 111kHz (10% lower). But, this time we found it to be in the 130-136kHz range previously noted. This change can likely be due to the addition of the break out point, the addition of the primary coil and rest of the system, and how it was measured in a different environment this time.

We adjusted for this change in resonant frequency by tapping the secondary coil at 7.5 turns instead of 8.25. This produced sound but no sparks. An incredibly proud and happy moment for the team because even though we did not have sparks, we knew at this point we would get them soon. We then went about tuning the coil. This involved moving the primary ⅛ of turn and testing a range of frequencies, recording the results, and repeating the process.

7.375 Turns on primary, pulse width at ~122us (less)
66 - 121KHz no visible sparks
64 - 125KHz visible sparks with adequate sound quality
63 - 127KHz larger sparks compared to 125 but worse sound quality

7.25 Turns on primary pulse width at ~122us (less)
62 - 129KHz small sparks with poor sound quality
66 - 121KHz small sparks with decent sound quality
61 - 131KHz Better
59 - 135kHz Louder and biggest sparks yet

7.00 Turns on primary pulse width at ~122us (less)
62 - 129KHz barely any sparks
61 - 131KHz Best sound quality so far
60 - 133kHz lost sound quality but same loudness
59 - 135kHz similar to 60

7.50 Turns on primary pulse width at ~122us (less)
67 - 119.25kHz no visible sparks
64 - 125kHz sparks barely visible
65 - 123kHz
62 - 129kHz really bad

6.75 Turns on primary pulse width at ~122us (less)
61 - 131KHz decent sound quality no spark
60 - 133KHz bad sound quality and spark
59 - 135kHz bad sound quality and no spark
58 -
57 -
64- 125kHz bad
62- horrible

We actually did not record our output at 7.375 turns 131kHz, but that is what we ended up going with in our final design.

As we stated earlier, we had not fixed the stray inductance problem at this point, so all of the sparks were small and not very loud. This made testing very bearable because we ran many trials and did not have to deal with the very loud noise it would eventually produce, or the scent of Ozone.

7. Bill of Materials

Part Name	Quantity	Price	Totals	
IGBT modules	1	$92.96	$92.96	
High voltage capacitors	2	$18.40	$36.80	
Diodes	4	$7.46	$29.84	
Mosfet Drivers (UCC27425)	2	$1.35	$2.70	Not used in final system
Primary Capacitors	6	$2.75	$16.50	
CIR BRKR 10A 120VAC	1	$28.74	$28.74	
Raspberry pi 3 model B+	1	$43.57	$43.57	
Secondary Wire (30AWG x 1 lb)	1	$36.50	$36.50	
PVC (4in x 10ft)	1	$9.60	$9.60	
Primary Tubing (1/4in x 50ft)	1	$57.98	$57.98	
Aluminium Ducting (4in x 96in)	1	$10.68	$10.68	
Isolation transformer	2	$15.93	$31.86	Not used in final system
10k 7W bleeder resistor for primary	2	$1.09	$2.18	
Platform Wheels	2	$6.93	$13.86	
100k 2W bleeder resistor for doubler	10	$0.15	$1.50	
Heat sink	10	$1.48	$14.80	
Polyurethane varnish (0.5 pint)	1	$9.59	$9.59	
Plywood	2	$13.59	$27.18	
Aluminum pie plates for toroid	1	$1.99	$1.99	
Grounding rod	1	$10.70	$10.70	Not used in final system
Chicken wire	1	$10.00	$10.00	
Paint brush	2	$1.68	$3.36	
Insulated electrical rubber gloves	1	$17.00	$17.00	
Wide paint brush	1	$3.98	$3.98	
PVC caps	2	$2.08	$4.16	
extension cord 25ft	1	$19.97	$19.97	
MOSFET Driver Part 2 (IR4427PBF)	7	$3.10	$21.70	
Ferrite Core for Isolation Transformers	2	$1.13	$2.26	
			$561.96	

8. Conclusion

Overall this was an excellent choice of project for senior design. It called upon every aspect of electrical engineering from detailed and minute electronics knowledge in the IGBTs and MOSFET drivers, to knowledge of power systems in the GDT and Tesla coil itself, to hardware programming, to electromagnetic fields in the stray inductance and measuring the resonant frequency, treating the top load as an antenna. It was very challenging at times as every minute detail could make a large difference in performance and it certainly tested our general electrical engineering knowledge and craftsmanship. And, the end result was incredibly satisfying, as any onlooker would agree.

If we could redo the project, we would have 3-D printed much of the housing and structural components because it would have been much easier and more precise than using wood and power tools and hand tools. We would also construct all of the circuitry in such a way that all components are as close together as possible, so that there are practically no wires and very little stray inductance. We would do more research into which ferrite core to choose for our GDT. And, finally we would use a microcontroller, because it is much smaller than our Arduino/Raspberry Pi setup.