Dan Walker Mr. Ohm LiDAR Module Project Summarization

University of Texas at San Antonio, Department of Electrical and Computer Engineering Student Team

CPE/EE 4813 ECE Capstone Design II - Spring 2023

Submitted on 05/10/2023

Project: Mr. Ohm Discrete Component Continuous Wave LiDAR Module

Project Manager: Brandon Colon

Electronics & Control Engineer: Justin Moreno

Mission, Systems, and Test Engineer: Ryan Johannsen

Manufacturing Engineer: Efrain Rivera

PCB Design Engineer: Abdul-Muizz Atanda

1.0 Introduction

Light Detection And Ranging (LiDAR) is a sensor that is used to determine ranges by targeting an object or surface with a pulsed or continuous light and measuring the time it takes for the reflected light to return to the receiver. Better Bot's plan is to implement LiDAR sensors within an educational robot to allow students access to industry level sensors. To make this possible, the original proposal was to modify the original Better Bot's LiDAR sensor to have an improved signal output, maintain a five Megahertz frequency, stay within a set size footprint, all while ensuring that the entire sensor is made from discrete components. Discrete components are individual passive or active parts of a circuit for example a resistor, capacitor or a diode. Using discrete components and maximizing the infrared LED strength the original LiDAR sensor signal was improved from a signal strength of 50mA to 100mA while keeping a five Megahertz frequency. The proposal was adjusted to include adding an amplifier to the receiver circuit to improve signal readability within the oscilloscope. The project had to deviate from the original receiver design due to the fact that the signal being collected was measured below 50 millivolts, too small to used but the inclusion of the amplifier produces a signal above one volt amplitude, allowing for a better and more accurate analysis and data collection from the signal. The functionality of the LiDAR device was shown via a Python program using several scientific and data analysis libraries to take oscilloscope measurements and use them to calculate distance from the phase shift. Using this Python program the collected signal can be analyzed for a variety of uses like object detection and distance measurement, however it mainly serves to prove that students with access to the bot will be able to program it for its intended purpose.

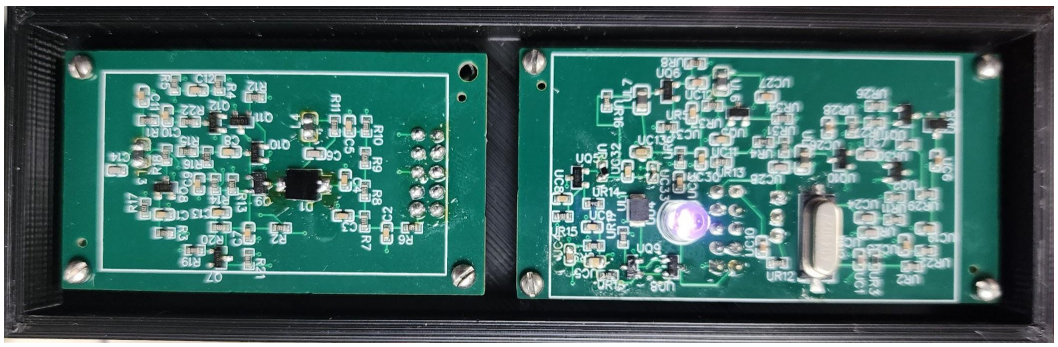


Fig 1.1 Final Module

2.0 Project Development

During the development of the project, our team divided itself into three main groups. One group handles LTspice simulations and prototype construction, the second group runs tests on the prototype and printed circuit board construction, and the third group handles programing and data acquisition. Part of the development process was learning key programs and tools like EasyEDA, LTspice, and a variety of tools for checking the health of the circuit.

2.1 Simulation

During the simulation stage group one's goal was to improve the Better Bots tuned driver circuit's 100mA peak-to-peak current to a 200mA peak-to-peak current through the infrared LED. The figure below was the original tuned driver circuit.

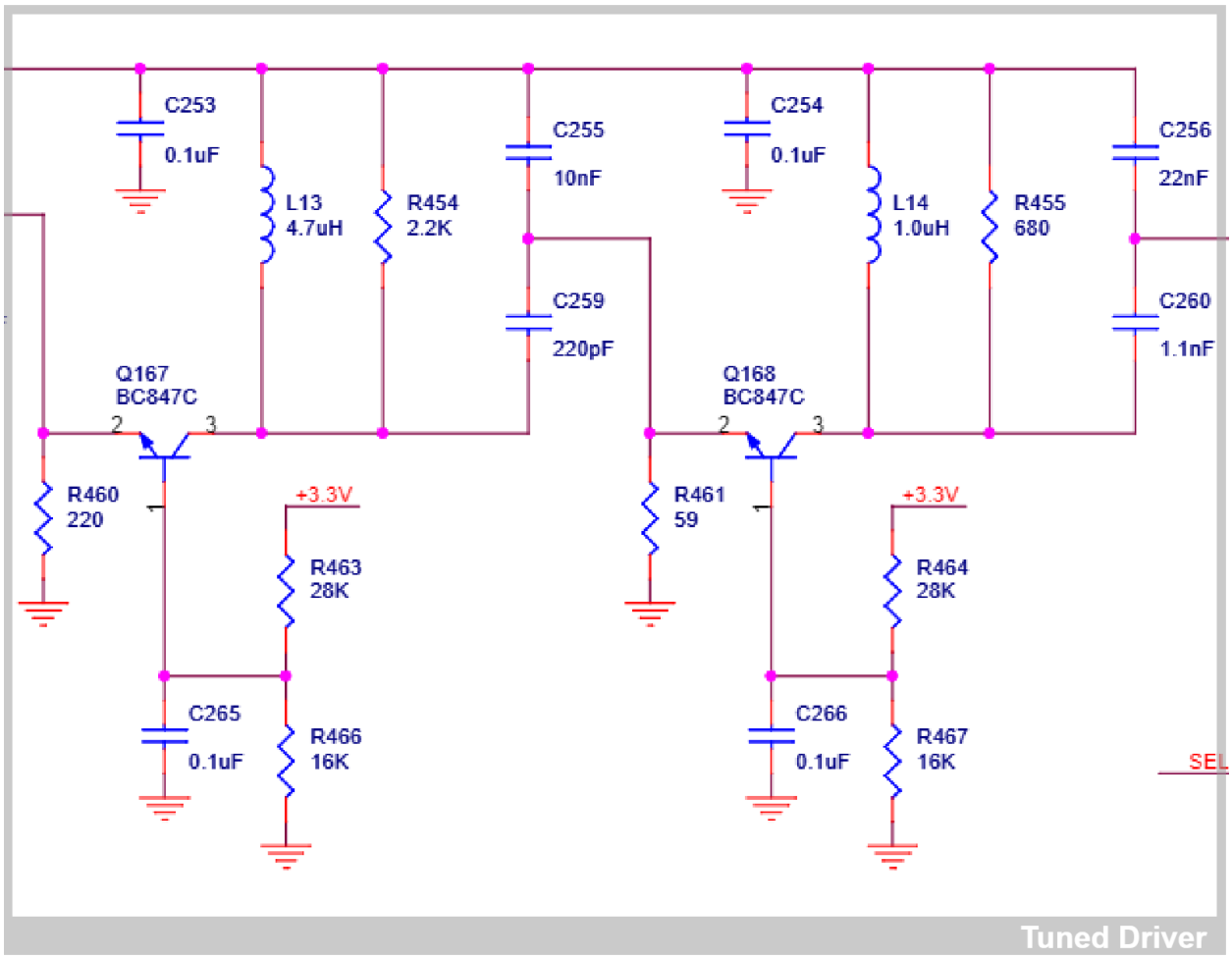


Fig. 2.1.1 Original Tuned Driver Schematic

Group one needed to learn how to use LTSpice from the ground up, which was essential for creating the correct modifications to the LiDAR module's tuned driver. To be able to increase the LiDAR signal strength, we utilized various LTSpice commands and an initial version of the emitter module as a starting point. The figure below is the schematic of the tuned driver translated into LTSpice.

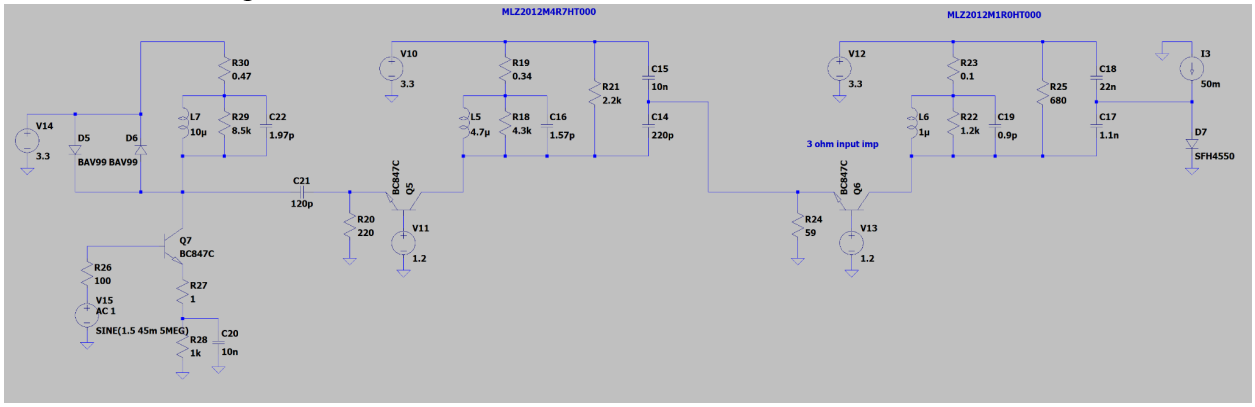


Fig. 2.1.2 LTSpice Tuned Driver Circuit

The original tuned driver circuit uses two stages of common-base amplifiers providing the starting 100mA to drive the IR LED. To adjust the bias current provided by the

Common-base amplifiers we need to change the resistor value on the emitter side of the transistor. Using the formula (Vbase-0.6)/Re) where Vbase is the voltage being applied to the base of the transistor and Re is the resistor at the emitter, we can ensure that the collector current never drops to zero. If the current drops to a value close to zero the transistor becomes unhealthy and distorts any signal being amplified. To increase the peak-to-peak a third common-base amplifier was added into the tuned driver, a LC tank load was attached to maintain the signal's frequency at 4.9 Megahertz. The figure below shows an early version of our simulated tuned driver circuit.

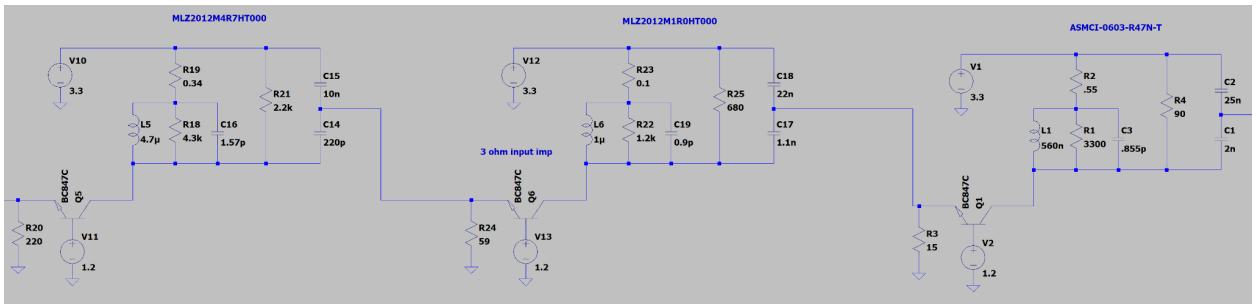


Fig. 2.1.3 Early Tuned Driver circuit

This early simulated version met the target 200mA current goal but had a transistor efficiency problem.

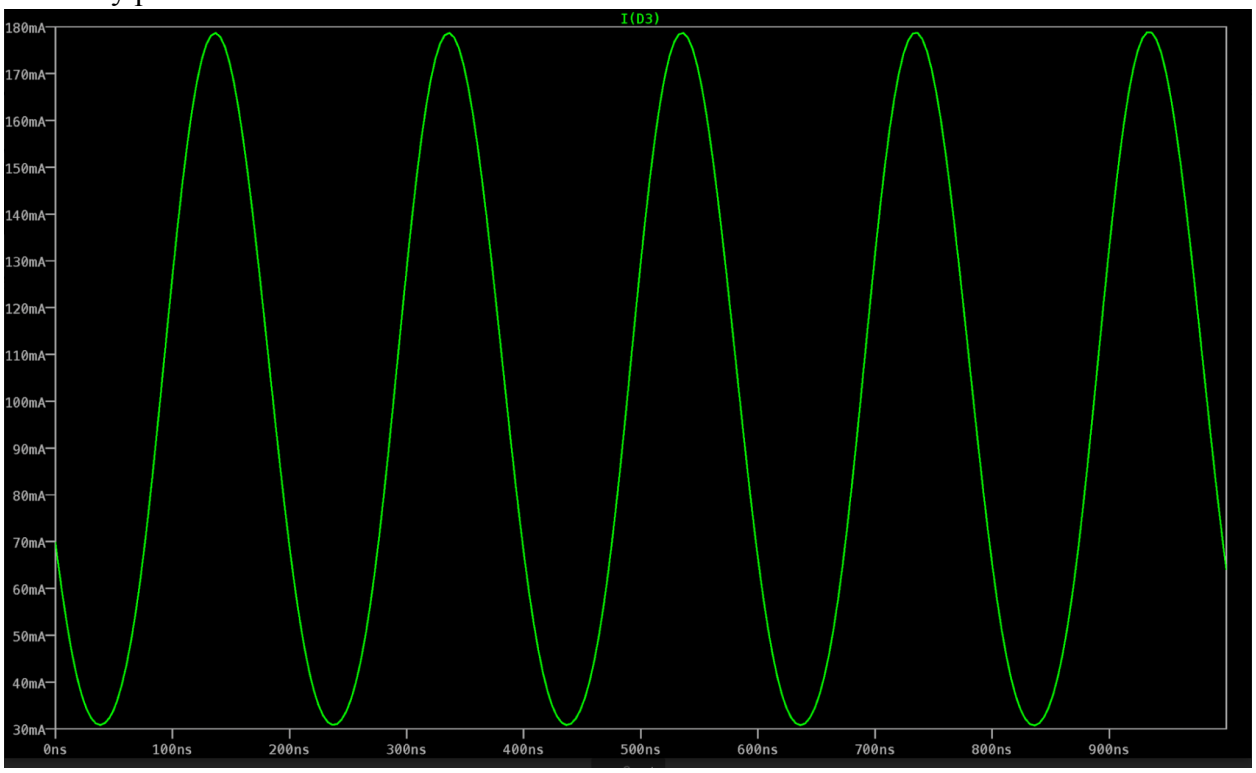


Fig 2.1.4 LTSpice final current across the LED

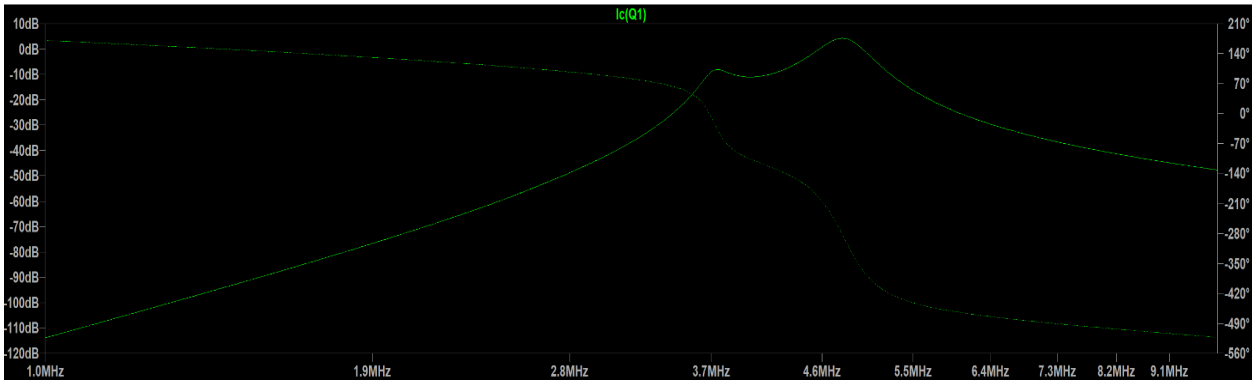


Fig. 2.1.5 Unhealthy Transistor Simulation

To solve the inefficiency problem tuning the Re at each common base amplifier and adjusting each LC tank load was important to ensure each transistor was efficient. Using the resonant frequency formula and swapping inductors the LC tank load was adjusted to help solve the transistor efficiency problem while keeping the signal frequency at 5MHz. Probing each node, probing for the current across the LED, as well as the power draw across the main amplifying transistors were methods of measuring the simulation in order to make value adjustments in each common base amplifier and LC tank stage within the tuned driver section of the circuit. Ensuring that the correct signal frequency is being maintained while the transistors are healthy, which could be checked by verifying their saturation at frequencies desired, as shown below:

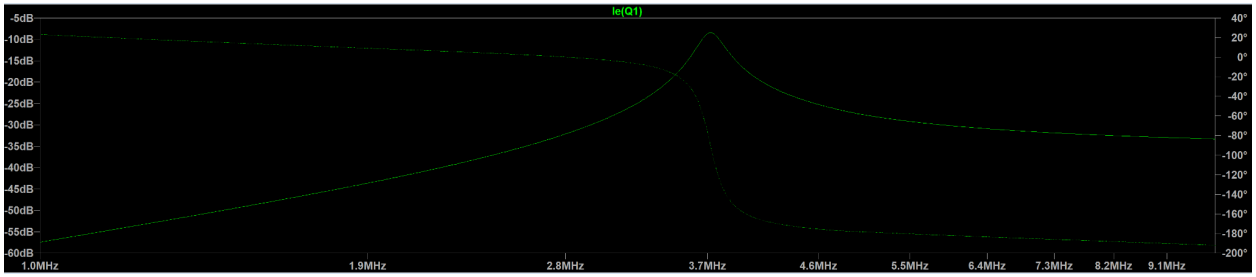


Fig 2.1.6 LTSpice graph of healthy transistor function

After transistor health verification was done, a current source was attached to the LED to ensure that the infrared LED emitted a continuous signal. A differential amplifier was added to keep the average current entering the LED at 100mA any higher than that and we risk damaging the IR LED since its average current max is 100mA.

In the final version of our simulated circuit we were able to achieve all of our target goals. We amplified our signal, ensured healthy transistor health and reduced the power consumption of the circuit which will allow this circuit to be efficiently run on a battery. In the figure below you can see the comparison between the original signal in blue and our new signal in green.

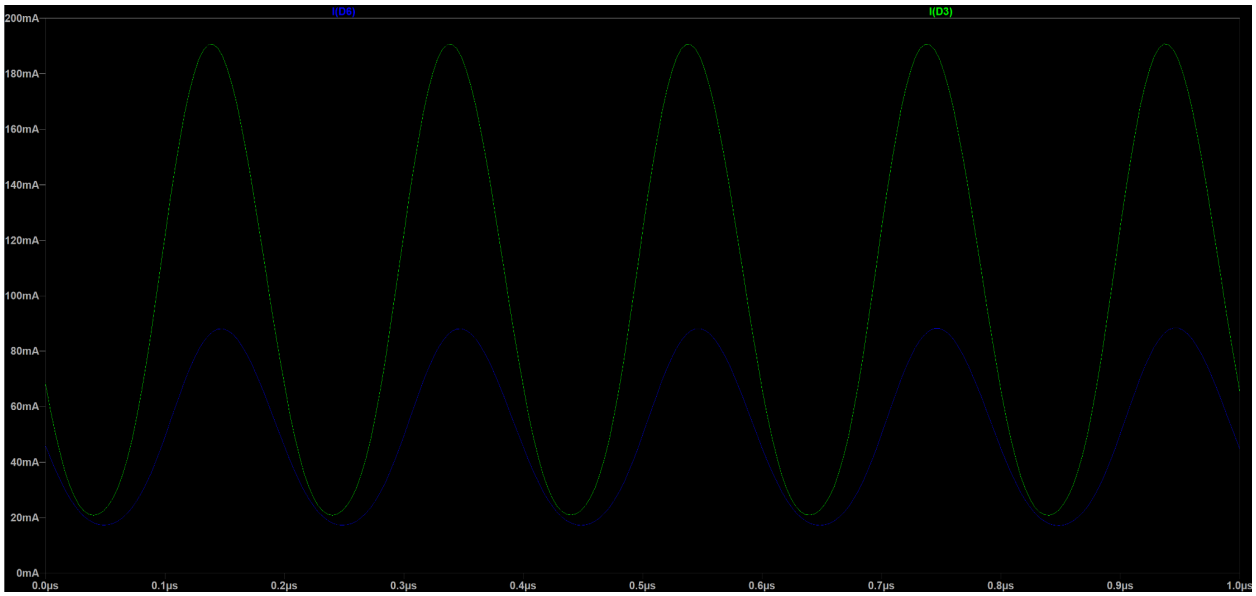


Fig. 2.1.7 Signal Comparison Old Vs New

The receiver was also simulated because without it we would not be able to test our emitter. Running the receiver within LTSpice we encountered that the output was too small to be easily seen on the oscilloscope. We created an amplifier circuit to increase the output signal to at least a one volt signal. The figure below shows the tuned photodiode receiver with our amplifier.

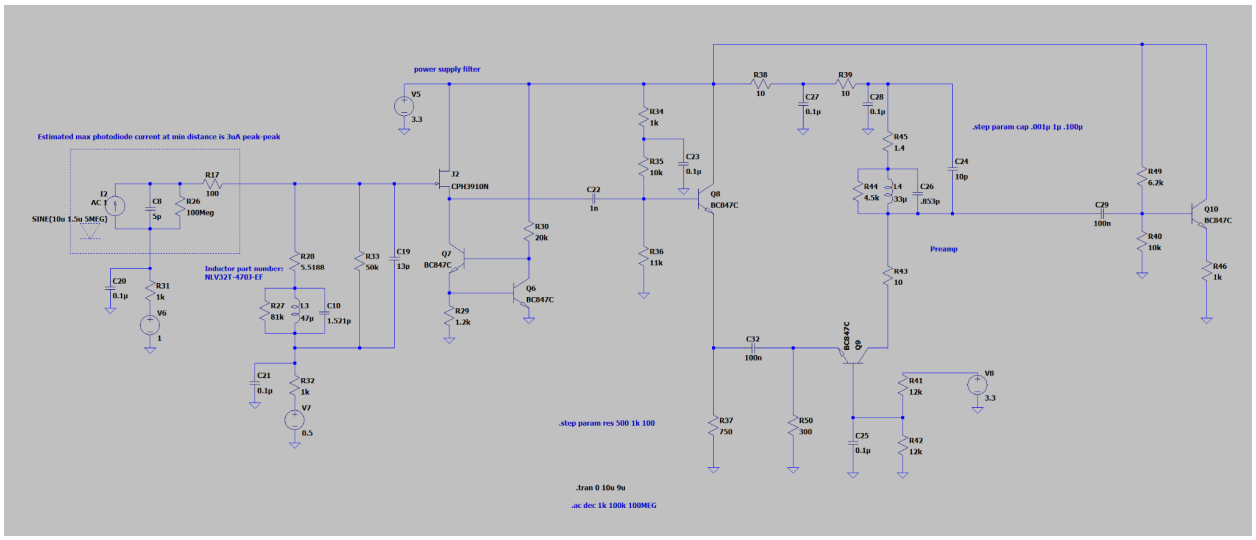


Fig. 2.1.8 Receiver with Amplifier

Using coupling capacitors we were able to adjust sections of the circuit without causing signal problems within other sections of the circuit. We also changed the output of the circuit into an emitter follower output which allows us to connect the oscilloscope probes without degrading the signal or its strength. In the figure below is the output signal without the amplifier in green and the output signal with the amplifier in blue.

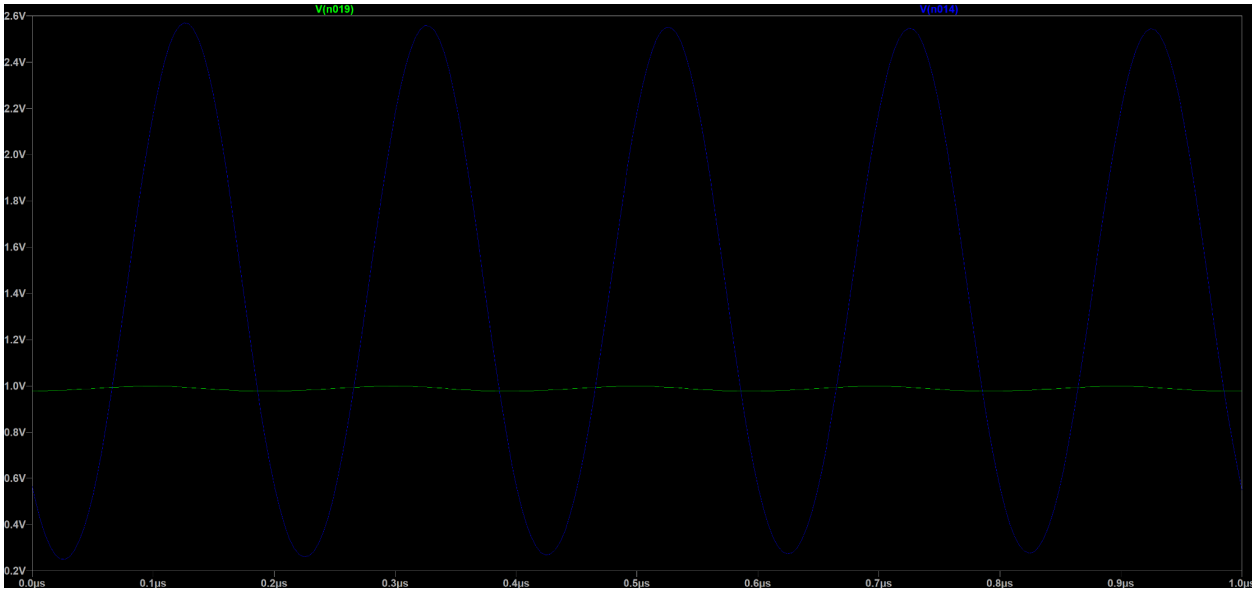
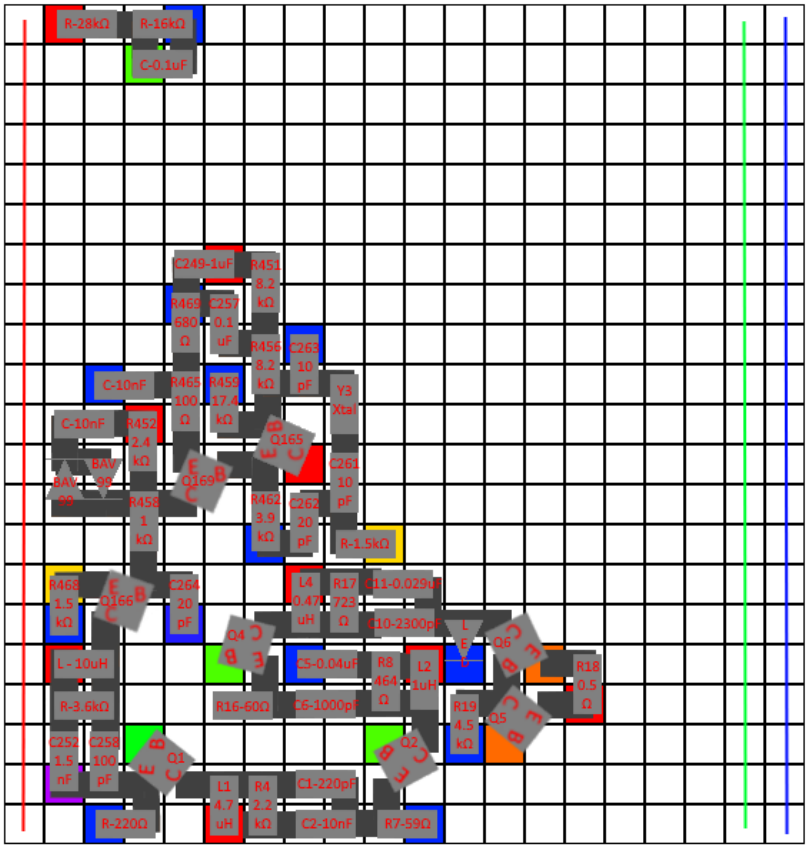


Fig. 2.1.9 Receiver Output with Amplifier and Without the Amplifier

2.2 Prototype

The key reason a prototype was built was to verify the functionality of the simulated circuit that was built in LTspice. LTspice simulates the circuit in a perfect world with no noise and perfect parts with zero tolerances; thus, building a physical prototype is essential to verify the simulation. Setting the prototype layout is key to ensure that there are minimal errors in the soldering process. While creating the layout, we allowed room for additional components to be added to the board; this space ended up being used for extra voltage dividers and wiring rails. The figure below is our wiring layout for the emitter prototype board. Each component is marked with a light gray rectangle and labeled with the type of component and value, dark gray paths are direct connections between components, red lines or red squares are locations that the power supply must connect to, blue lines or blue squares are locations that ground must be connected to, green squares are connected with the voltage divider and all other colors are locations that must be connected to each other but do not have a direct path.



Testing the prototype began in unison with the construction and integration of various components. As a component is soldered and connected with a node, continuity tests, and impedance checks were conducted to verify the connection between components. Figures 2.1 and 2.2 show the integration of each component from initial placement to final stage integration.

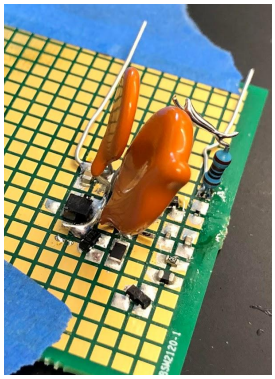


Fig 2.1

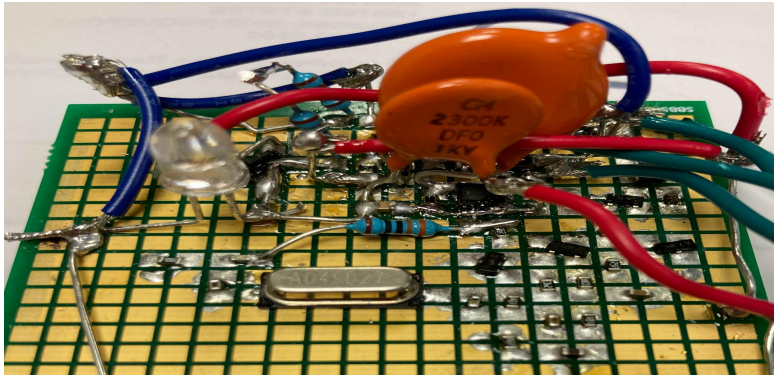


Fig 2.2

Once construction, continuity and impedance checks are done, a functionality test is performed for both the emitter and receiver board. Emitter is connected to a power supply and a function generator while using an oscilloscope to test various nodes within the tuned driver to verify an amplifying 4.9152 Megahertz signal. The measurements recorded on the oscilloscope are verified against the simulated values and the LED is tested to verify emission by visually seeing a dim light at the diode. If the LED does not emit any light or the recorded values do not match or is off then troubleshooting begins before claiming that the simulation is wrong.

Majority of the errors came from a part that was soldered wrong, incorrect wiring or insufficient power being applied to the circuit.

The receiver is connected to a power supply while using an oscilloscope to test various nodes within the receiver. Using a combination of a function generator and an infrared LED, a 4.9152 Megahertz signal is generated and pointed at then away from the receiver's photodiode to verify signal collection. To ensure that the receiver is collecting the correct signal a frequency sweep is conducted and the peak output is measured using an oscilloscope. Figure 2.3 and 2.4 are oscilloscope readings at the test nodes

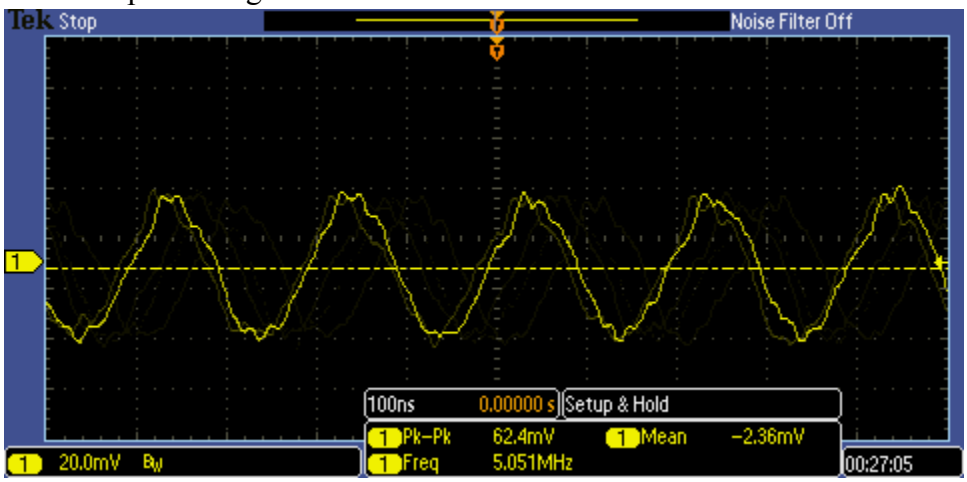


Fig 2.3 Emitter test node

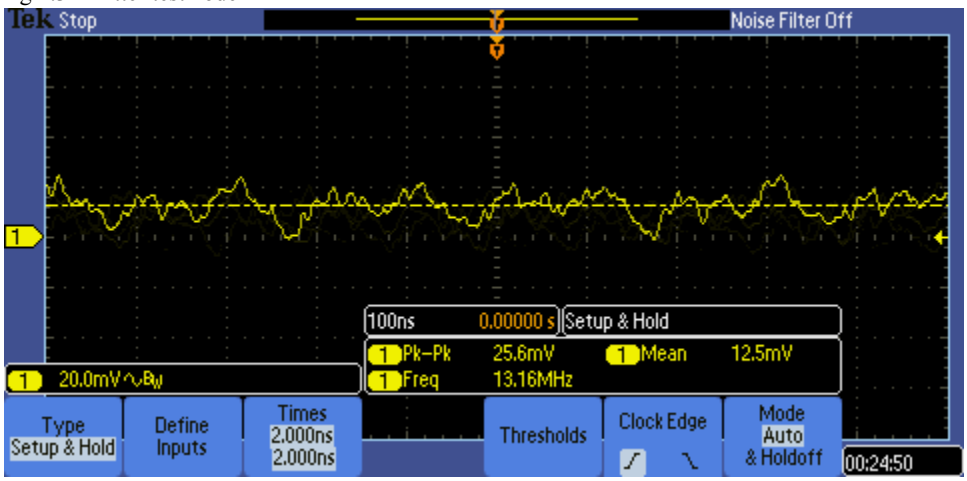


Fig 2.4 Receiver test node

Many errors were found in this stage, verifying solder connections and adjusting resistors to correct any amplification values while adjusting capacitor values to correct resonance frequency values. A major problem that was encountered involved external wires that ran over the board, which generated a small electric magnetic force (EMF). This small EMF was enough to generate a significant amount of noise that degraded the output signal. By rewiring and changing the oscilloscope probe, the EMF generated was reduced but not eliminated. However, the final build of the circuit was able to completely reduce the EMF by utilizing a four-layer process with multiple grounding layers to maintain signal integrity and minimize noise buildup.

The final step is to verify dual functionality that both the emitter and receiver can work in tandem. This was performed by powering on the emitter and receiver while connecting the oscilloscope to verify the correct signal is being emitted and collected. To check for changes in

the signal the emitter was pointed directly at the receiver, then pointed away and also covered up. Figure 2.5 shows the oscilloscope output of both boards

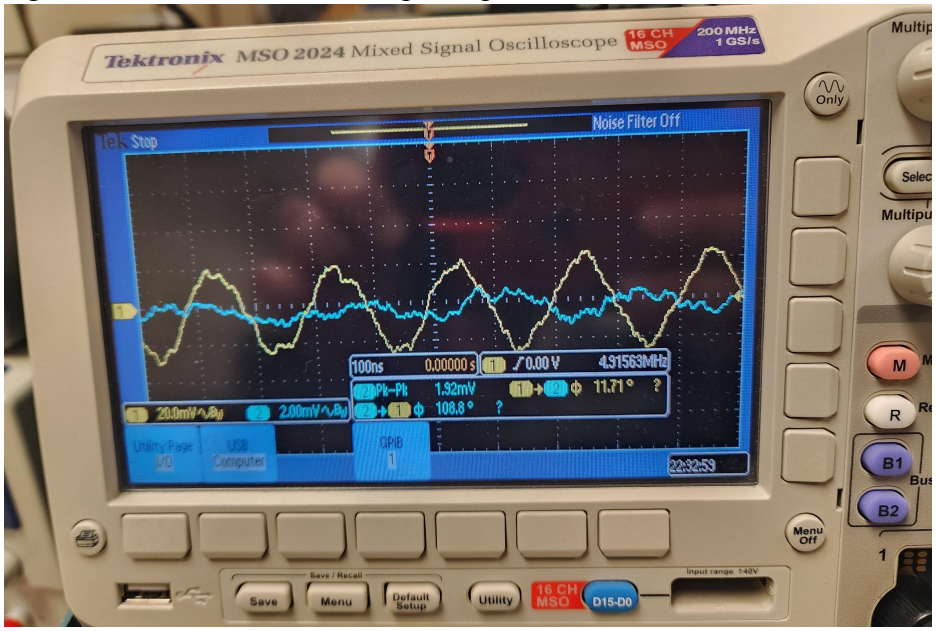


Fig 2.5 Dual functionality

Performing a final resistor and capacitor adjustment to ensure a 4.9152 Megahertz signal is correctly emitted and received, the boards are given to the printed circuit board design group and are designed on EasyEDA for the final product.

2.3 Final PCB

EasyEDA is a software that allows anyone to design, simulate, and order PCBs. EasyEDA has two services to be familiar with: JLCPCB and LSCS. JLCPCB is the part of EasyEDA that handles anything regarding your PCB orders. LSCS serves as the extensive parts library that EasyEDA users are able to access for their projects. EasyEDA users can work on their projects online via their website: https://easyeda.com/. This is possible since a user is able to make an account with EasyEDA. Any saved information is shared between both cloud/website and their local desktop app. A feature the team liked was that you can collaborate with other EasyEDA users on the same projects.

One example was when Justin worked on one of the Emitter versions. Abdul gave access to Justin so that Justin could work on it while he was temporarily unavailable for a week. Abdul was able to control how much editing control Justin had. Meanwhile, Justin could access Abdul's shared work anywhere he could access EasyEDA. This saved us valuable time.

A very valuable tool Justin learned to love is vias. The GND vias and the vias used for crowded traces made tracing those routes so much easier than either using auto-routing. You will encounter problems where a trace, or multiple, do not pass the Design Rule Check for whatever reason - not long enough width, overlapping traces in the same plane, and more. Vias are so useful for our design. By using vias and setting their net to GND, you eliminate signal noise and save space for routing traces. Another use

Another feature that even EasyEDA highlights is their parts library. The amount of options presented for available parts are expansive. This parts library - LSCS Parts Library- is generally sorted via common parts library and extended parts library. There is, at the time of writing this, an additional $3 USD surcharge to every single additional part that is added to your

PCB. Our anecdotal example was when we originally used about 15-20 parts from the extended library for the Receiver PCB. This made producing the PCBs and assembling most of the SMD parts cost around 3x more than it would've been if we solely used parts from the common parts library. This was later fixed by going back to your wiring diagram and finding the equivalent common parts library part of each component. We did this by comparing datasheets with specific characteristics this project required - SMD size, tolerances, throughput, and more. By doing this, the total cost of just physically creating, assembling, and shipping the PCBs dramatically decreased from around $250-$300 to about $80.

Receiver V1

Justin had previously used a software called KiCAD via workshops, but had rudimentary PCB experience. This was the first time he had ever used EasyEDAThis first version of the Receiver PCB development only included wiring the Tuned High-Z Photodiode Amp. I learned the basics of operating various commands and tools listed on EasyEDA. Previously, I used their desktop application to start drafting. At this stage, I was getting familiar with EasyEDA's LSCS Parts Library. At this stage (up to V5), we kept using parts from the extended and common parts library.

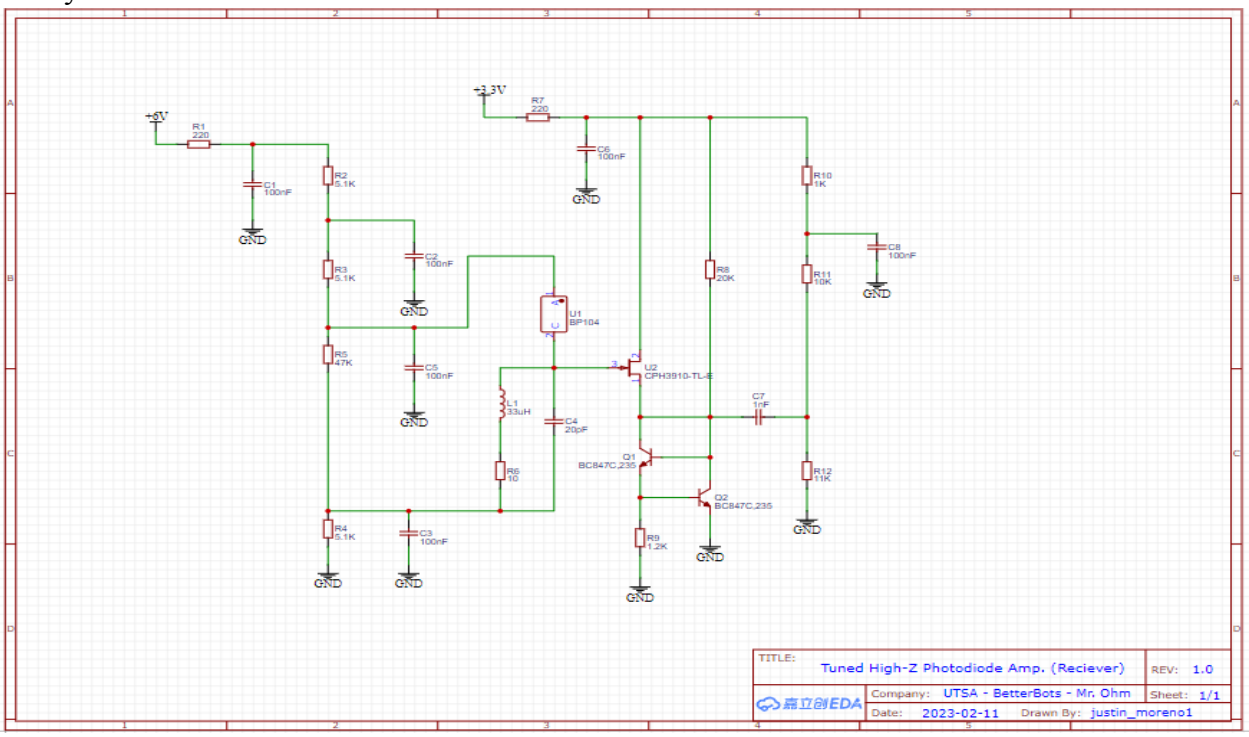


Fig. Original Wiring Diagram

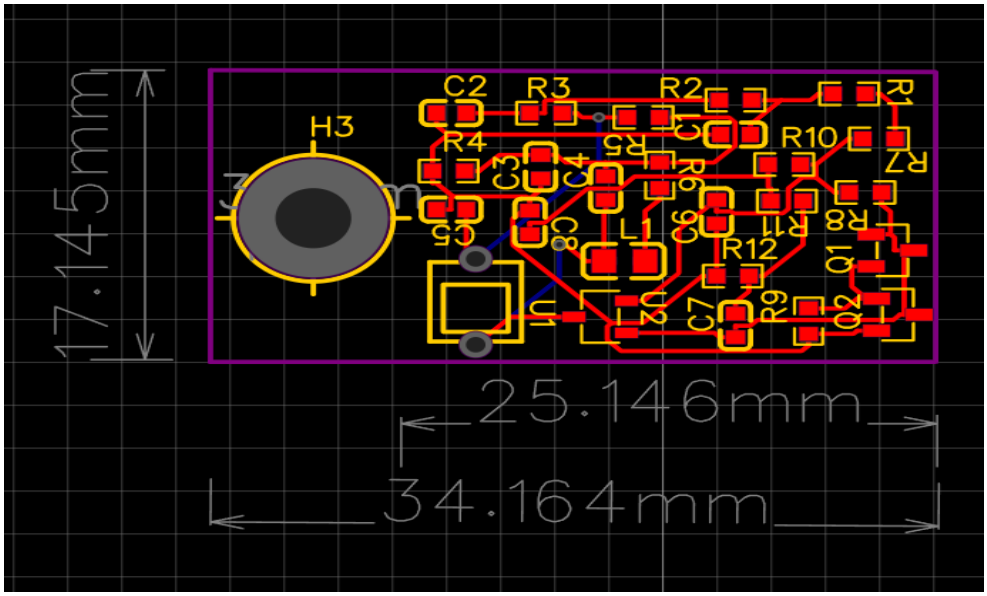


Fig. Receiver V1 PCB Schematic

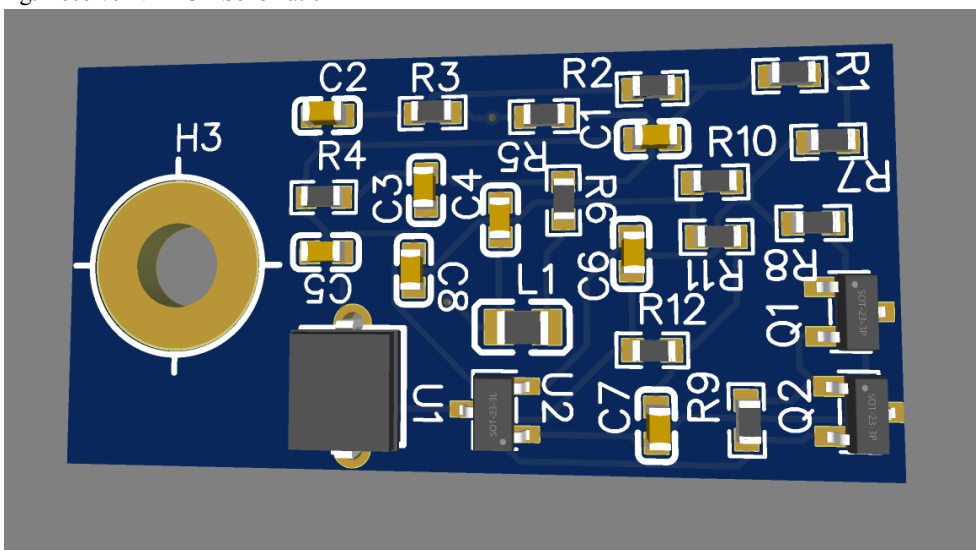


Fig. 3D Model of Top Layer

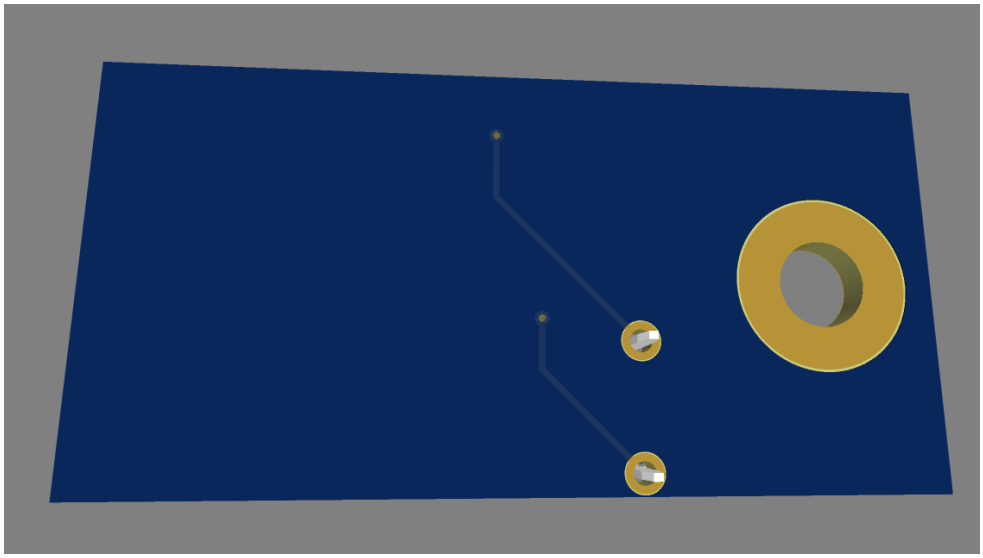


Fig. 3D Model of Bottom Layer

Receiver V2

This version adds a header into the wiring diagram. I did not know at the time that I needed to connect the VCC, S_Out, and GND lines to the rest of the circuit. Please do not draft your headers like how I did for this version's wiring diagram! I was still using Auto-Routing for the traces of the PCB. This version only had 2 layers. The main new additions were adding a header and adding a GND plane between the top and bottom layer to reduce signal noise.

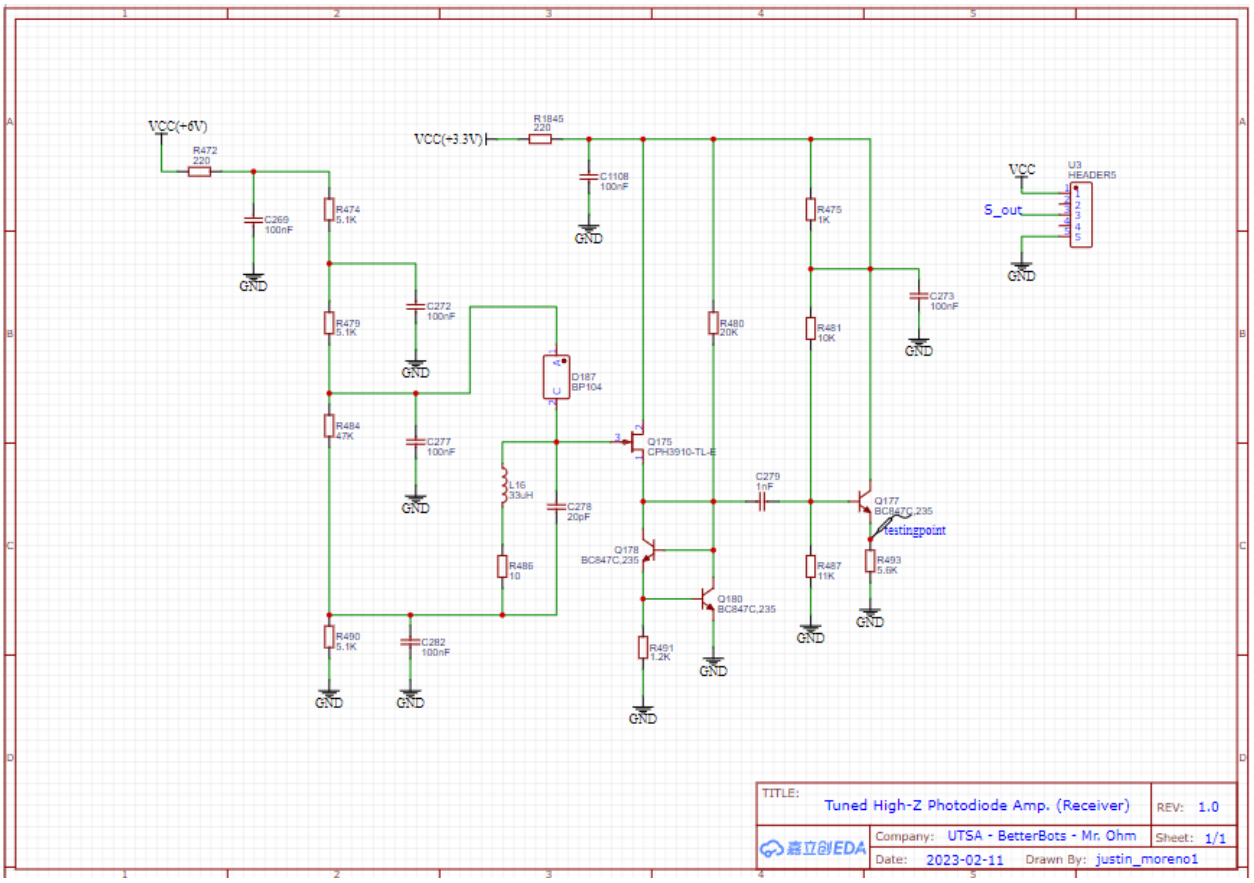


Fig. Header introduced for 1st time

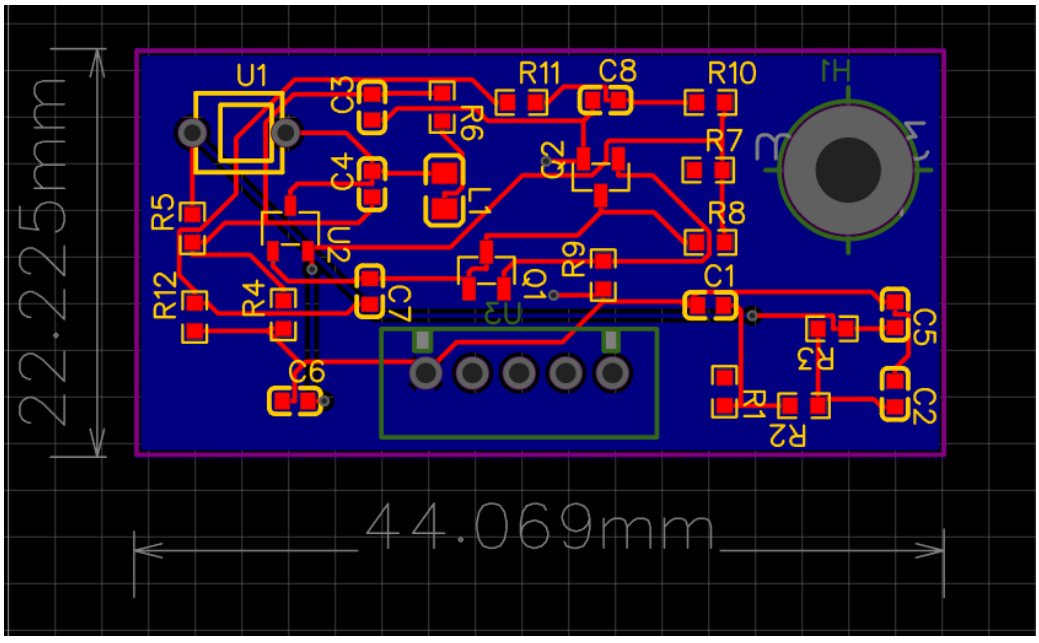


Fig: Addition of 5x1 Header to PCB Schematic

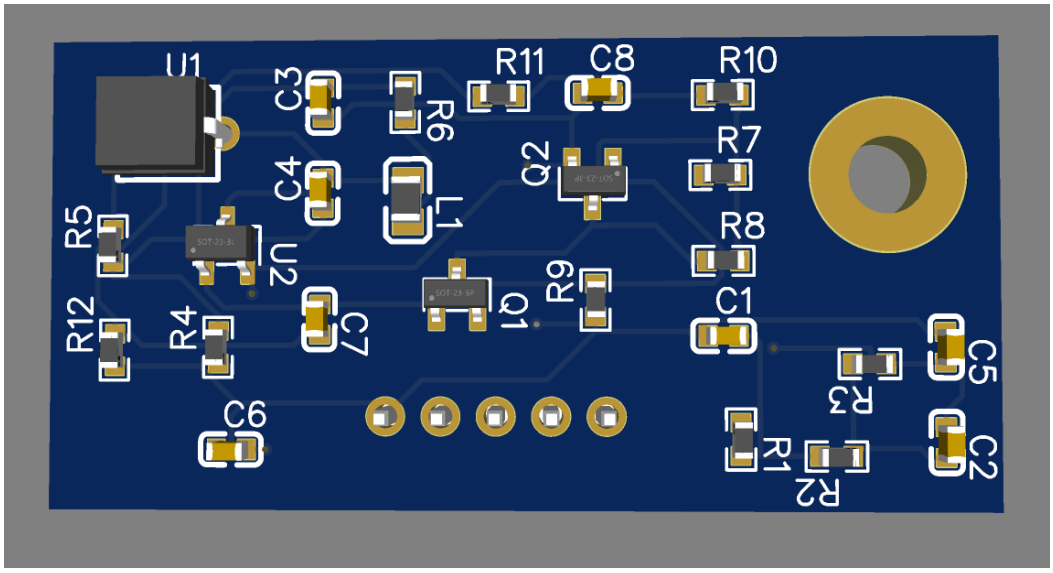


Fig. 3D Model of Top Layer

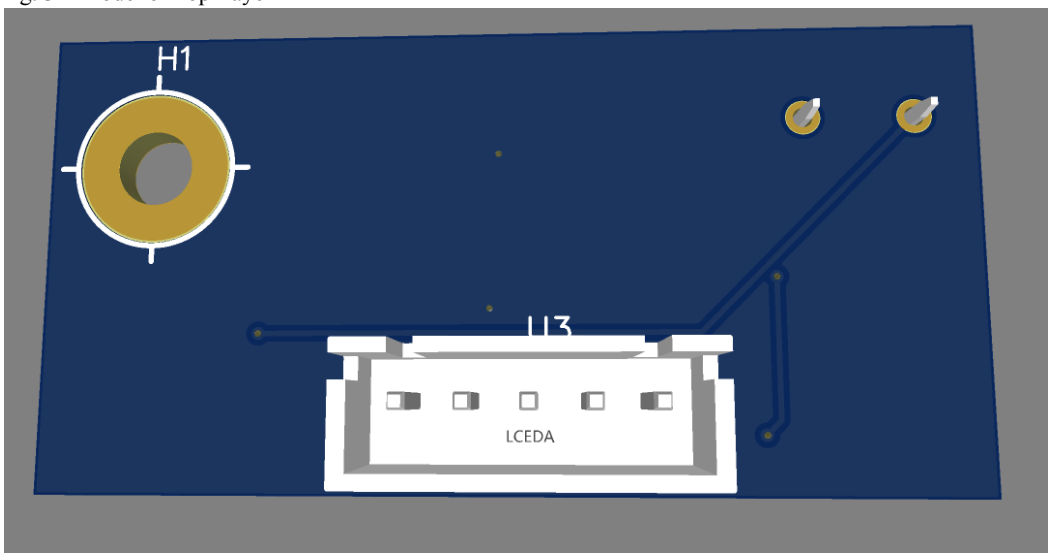


Fig. 3D Model of Bottom Layer

Receiver V3

The wiring diagram from this version did not really change until receiver V5. This version of the receiver is when the team started to hone in on the final version. We included multiple screw-holes on the corner of the PCB schematic since we were brainstorming 3D-Printing casings, were close to finalizing a generalized size of the PCB (with Dan's guidance), and we started to manually optimize the traces. One lesson learned from this version is to not have our SMD parts so spaced out in the PCB. We don't want to add delay to our signal.

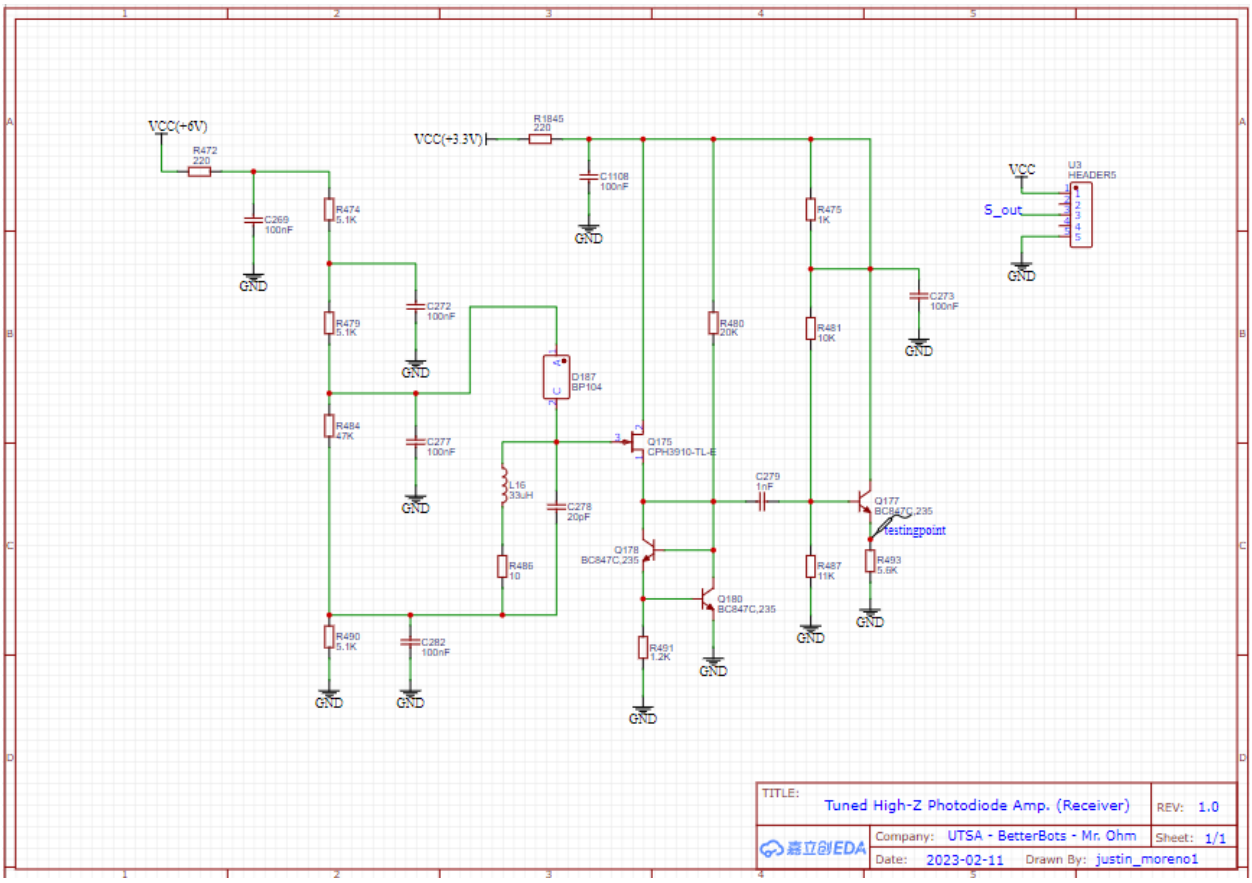


Fig. Wiring Diagram

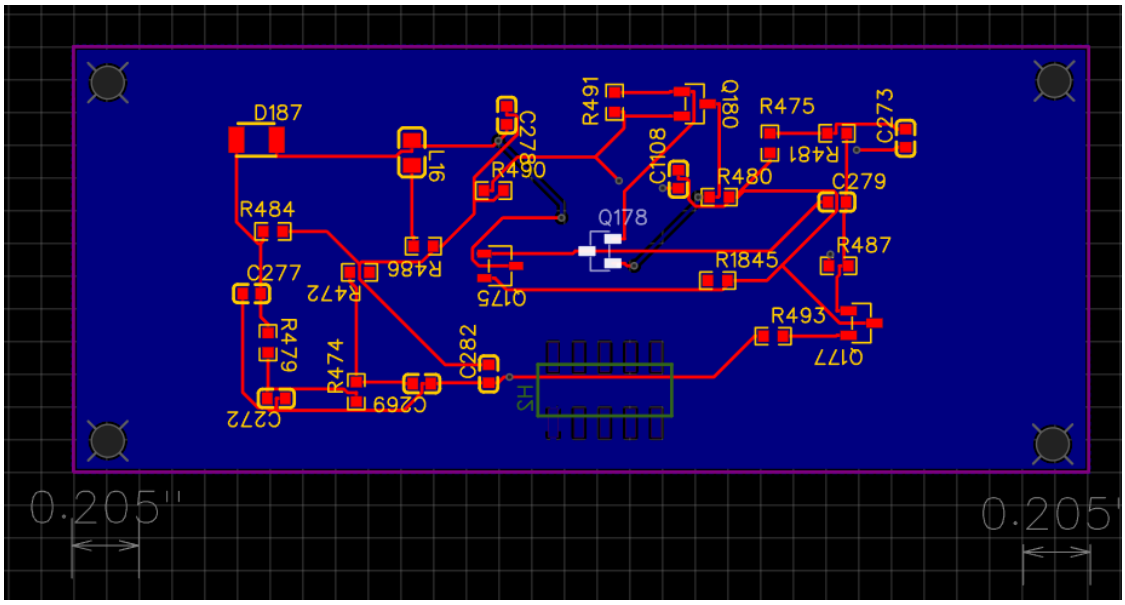


Fig. PCB Schematic

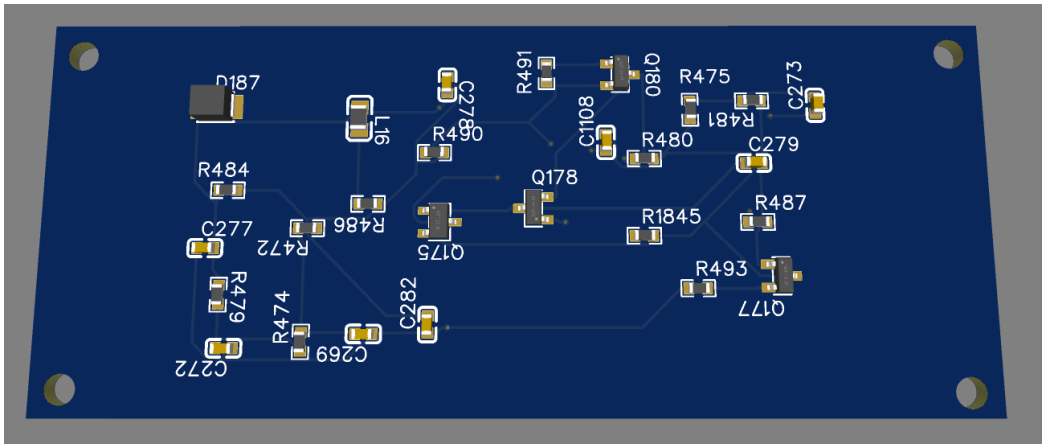


Fig. 3D-Model of Top Layer

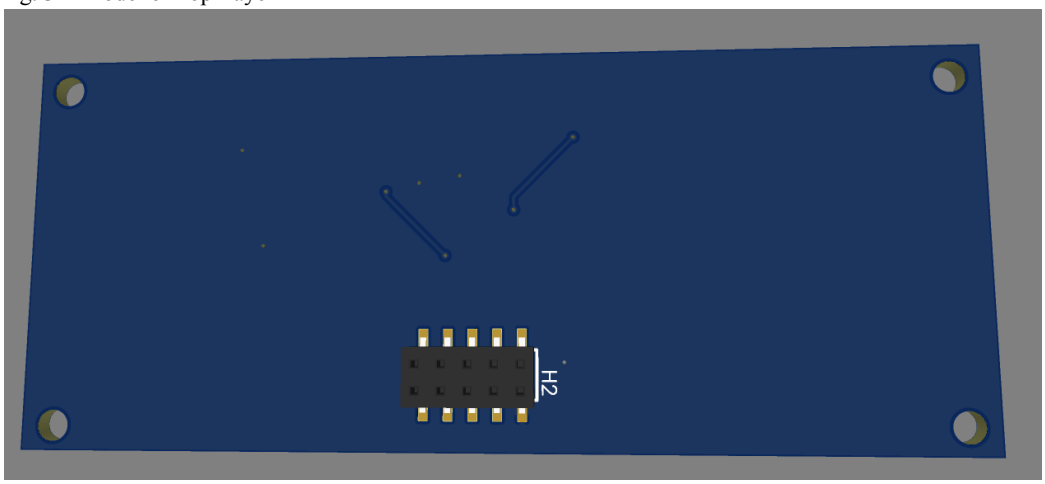


Fig. 3D-Model of Bottom Layer

Receiver V4

If I had to describe our thought process with V4 with one word, it would be: optimize. The PCB schematic underwent intense revisions for this version in order to maximize its spatial occupancy on the board. I also included the PCB's physical dimensions so that I wouldn't have to dig through multiple menus to find them. Also, this is where we decided to set the screw-hole's size to 2.54 mm (0.1") uniformly.

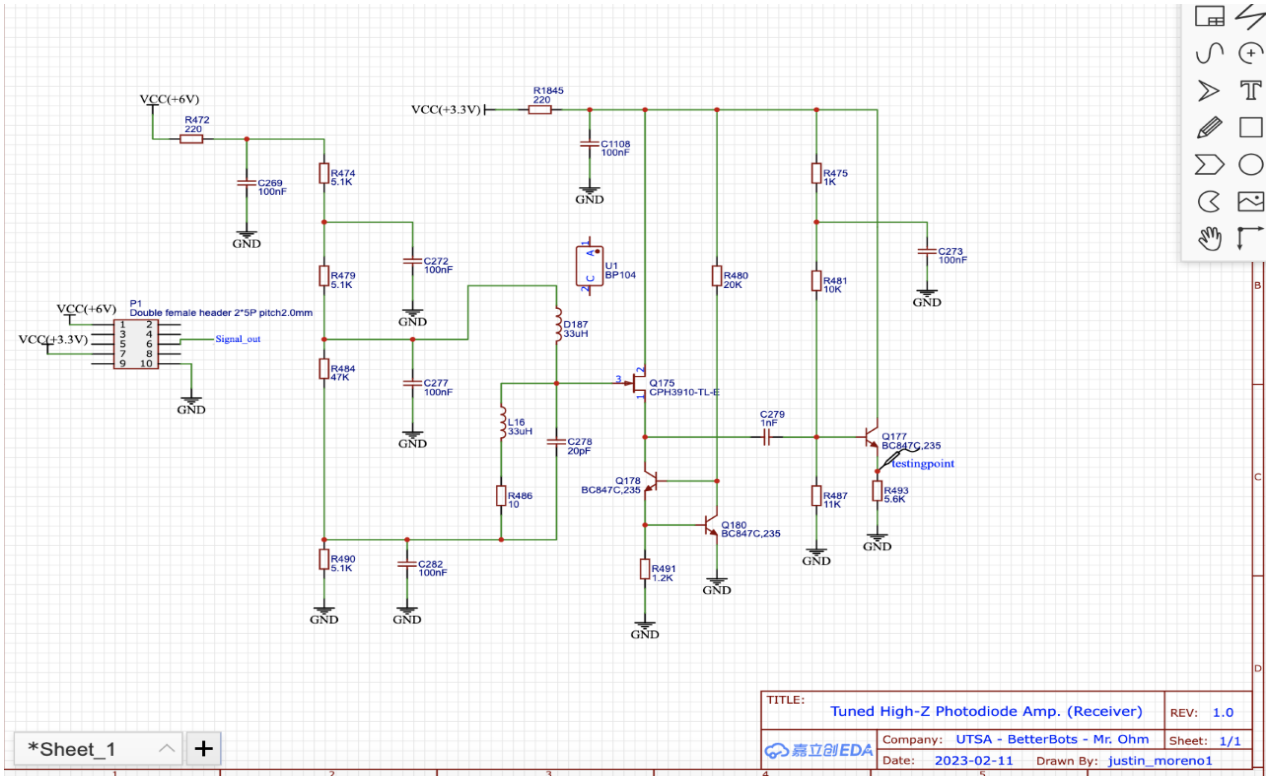


Fig. Wiring Diagram

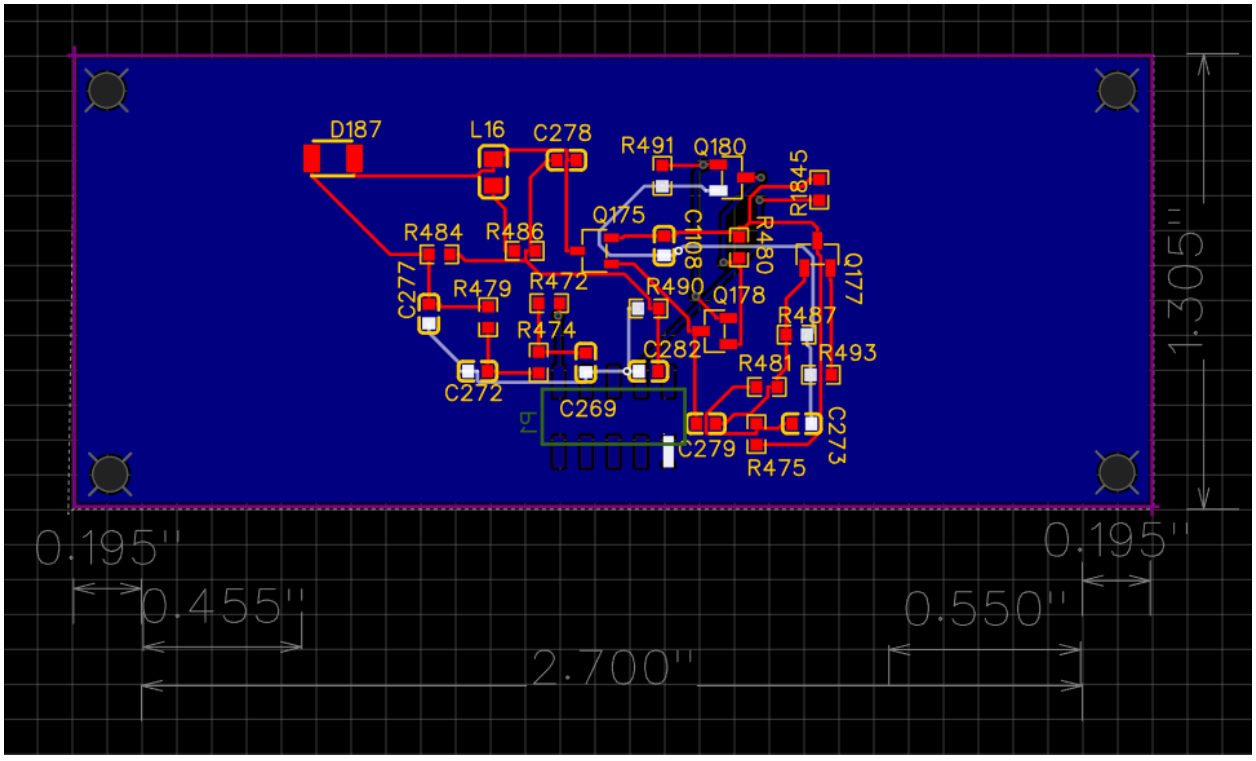


Fig. PCB Schematic

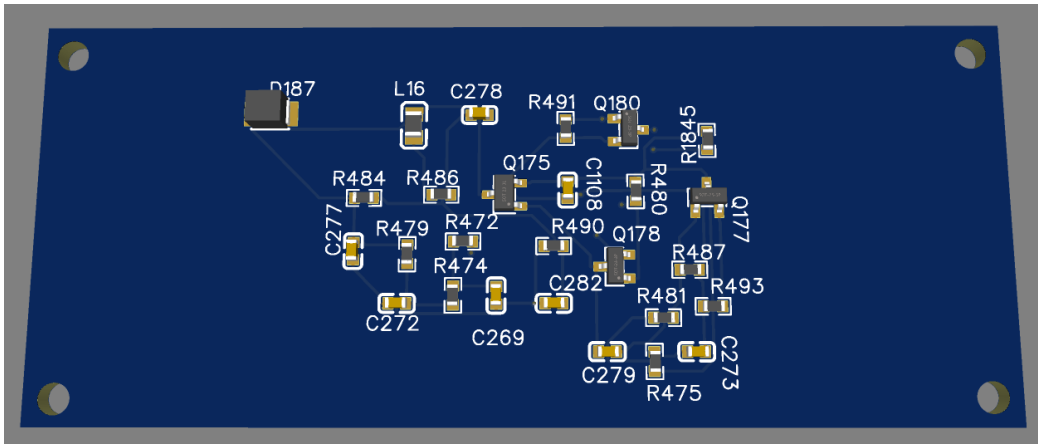


Fig. 3D Model of Top Layer

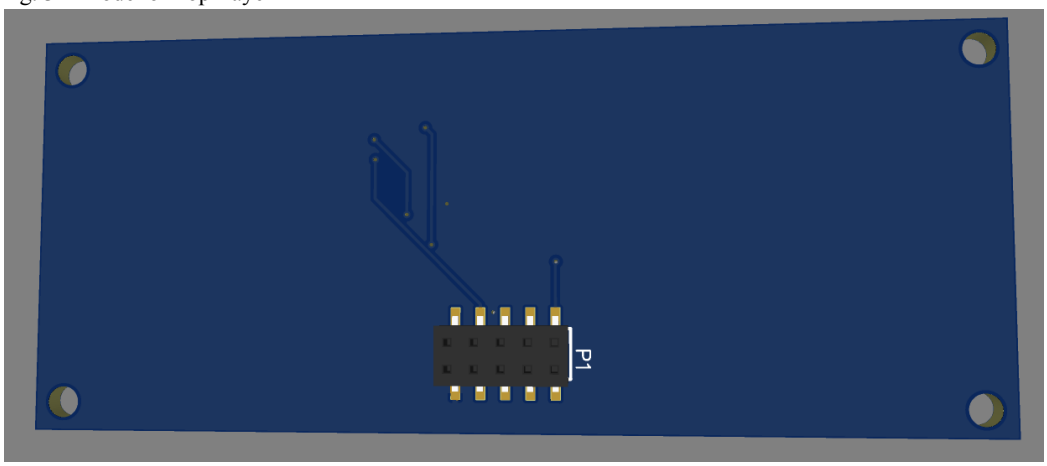


Fig. 3D Model of Bottom Layer

Receiver V5

Receiver V5 was the version where we fixed a lot of issues. The biggest addition we incorporated was the addition of an amplifying stage on the right half of the wiring diagram. The reason why we needed more amplification was during protoboard testing, our output signal was weaker than originally expected. This is where we decided to use a 5x2 (10 pin) header instead of the 5x1 (5 pin) we were using previously, replacing erroneous parts like BP104 - since it was not an inductor originally, and more. For the PCB schematic, I finally had figured out how to make a top silk layer box (in yellow) highlighting the circuitry. One day I figured out what to do by messing around with the various layers on EasyEDA. You can make the same Top Silk Layer box by clicking the yellow Top Layer object under 'Layers and Objects' on the right side of the website, press 'Rect' that's listed under PCB Tools, drag over the rectangle over the PCB, and for Rect Properties, select 'No' for 'Fill'.

By this stage, we already finalized the PCB's dimensions by adding more measurements on the PCB schematic. Also, this stage is where the team decided to go with 4 PCB Layers instead of 2 Layers previously. This was done after the team conducted protoboard testing and our initial results were very noisy. The configuration we set up was 1st Layer - Top Layer, 2 Layer - a.k.a. Inner1 - we set the entire net to be GND (entire layer acts as GND), Inner2 - we added an empty inner layer, and 4th Layer was the Bottom Layer. So I set a GND plane in between the Top Layer and the Inner1 (GND) Layer to make our signal even less noisy.

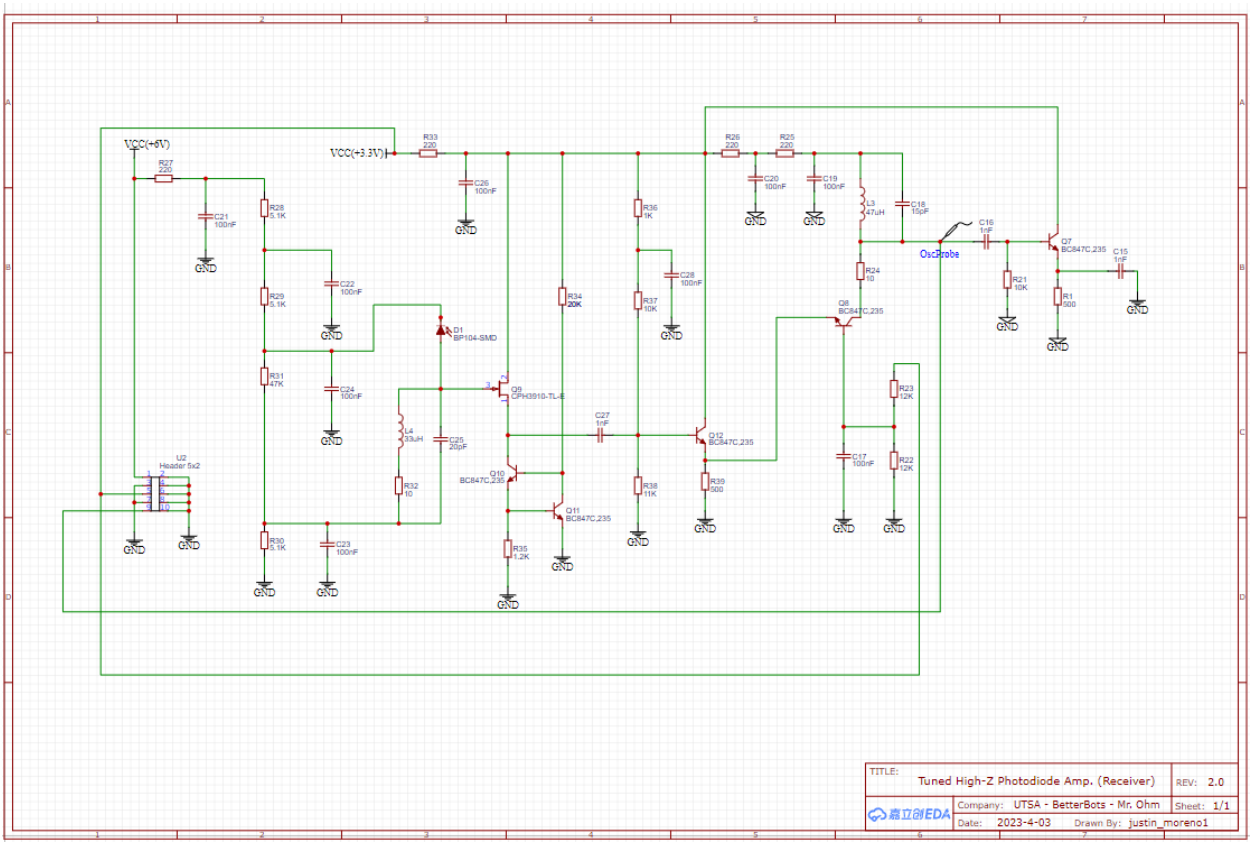


Fig. Addition of final header and swapped erroneous parts with correct ones

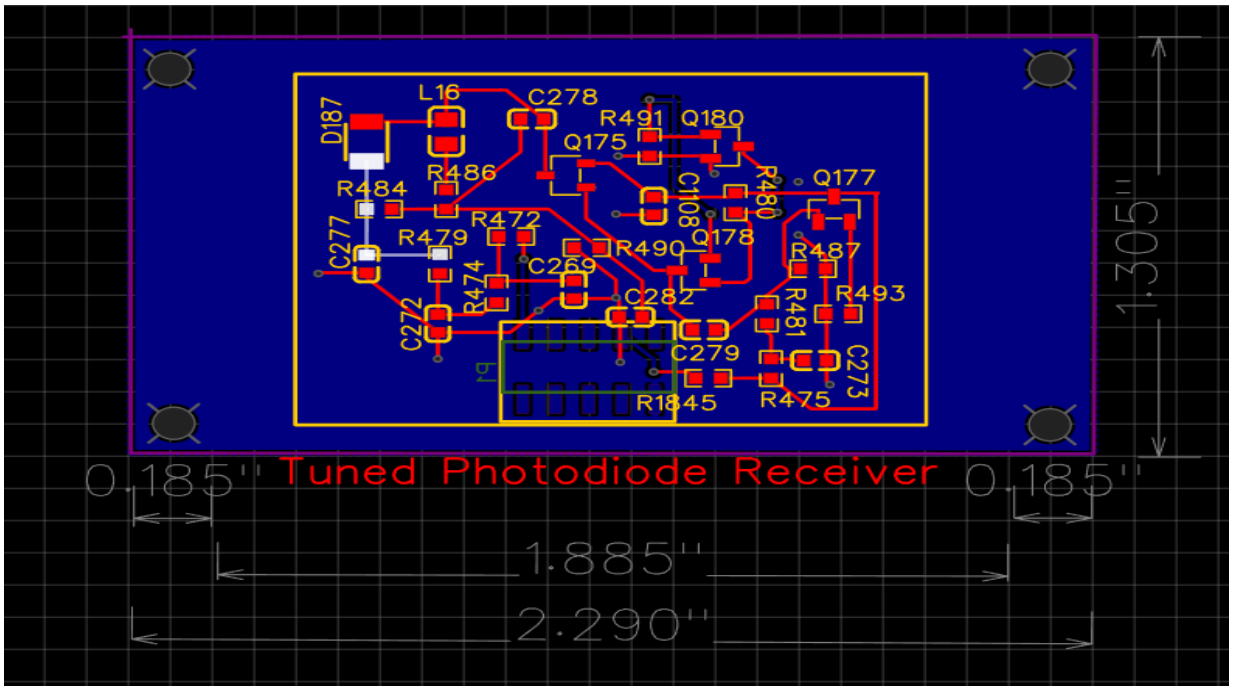


Fig. Addition of Top Silk Layer box and other useful tools

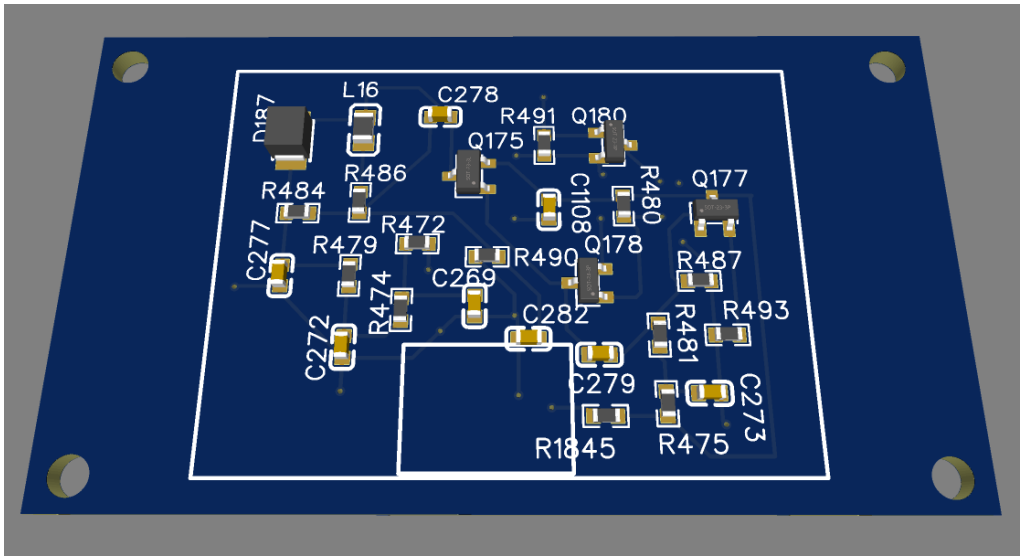


Fig. 3D Model of Top Layer

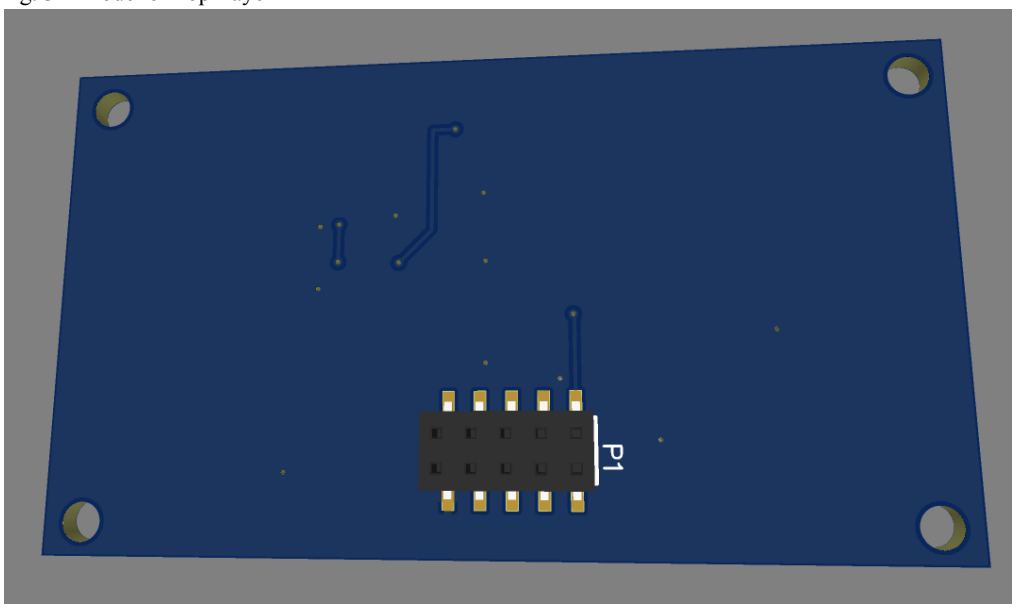


Fig. 3D Model of Bottom Layer

Receiver V6

The final version of the Receiver PCB we built! There are many issues we fixed between the last version compared to this version we used at the UTSA KCEID Tech. Symposium. This version saw us polishing and cleaning up various issues: replacing the extended parts library parts to common parts EasyEDA offers, clarifying different aspects of the circuit with labeling on both the wiring diagram and PCB schematic. The PCB schematic was reworked from scratch to include the new amp. stage and the common parts. Since I was redoing the PCB schematic by scratch, I did my best to optimize how much space the circuitry was going to take.

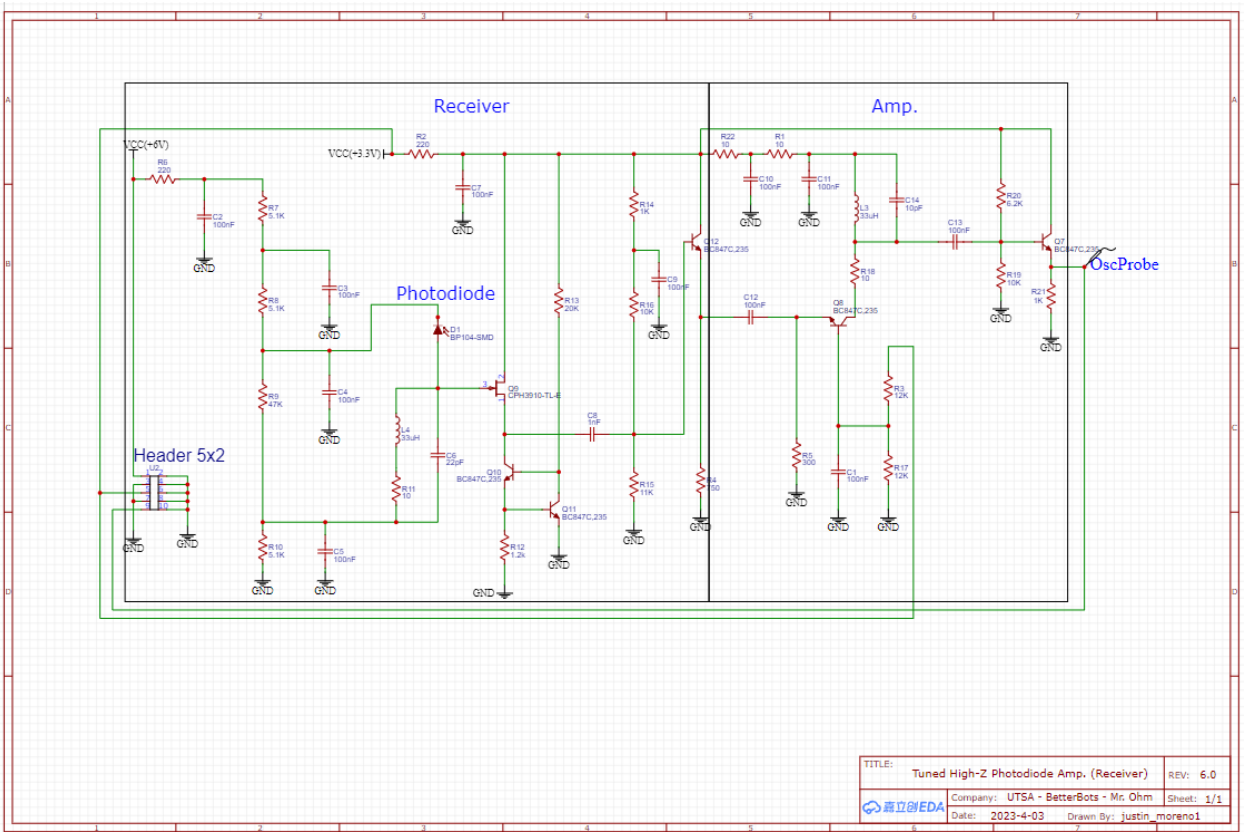


Fig. Final version of Wiring Diagram

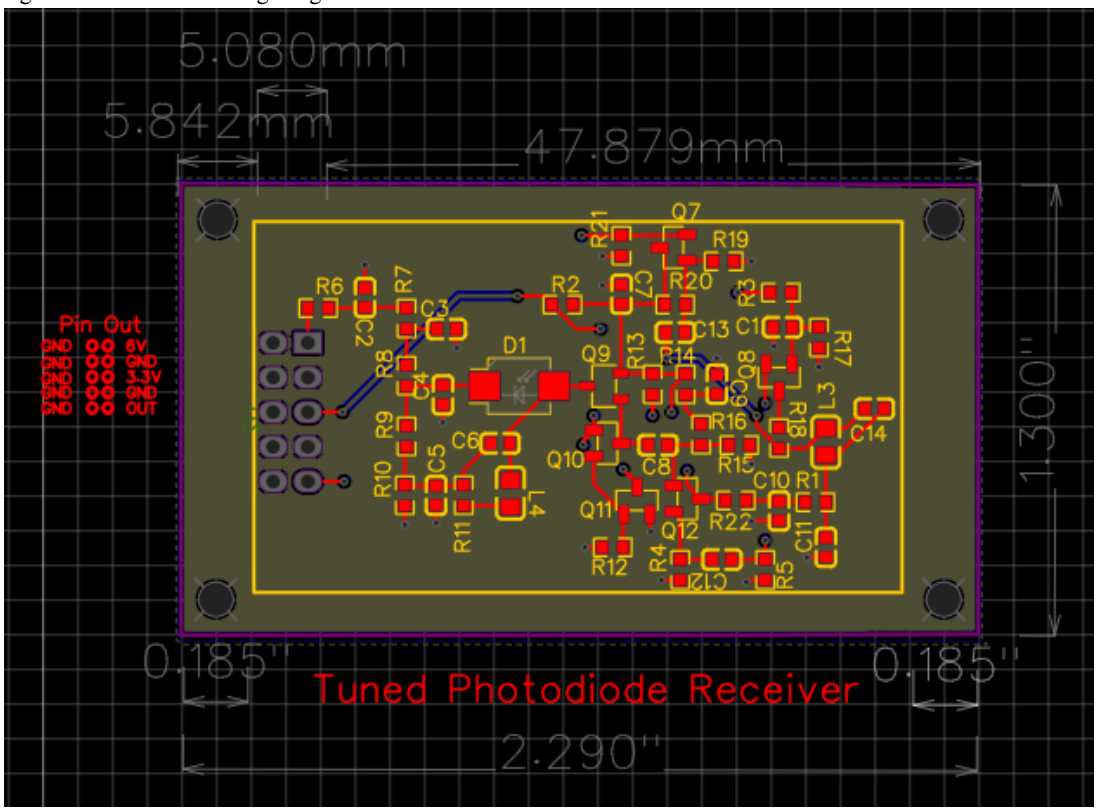


Fig. Finalized PCB Schematic

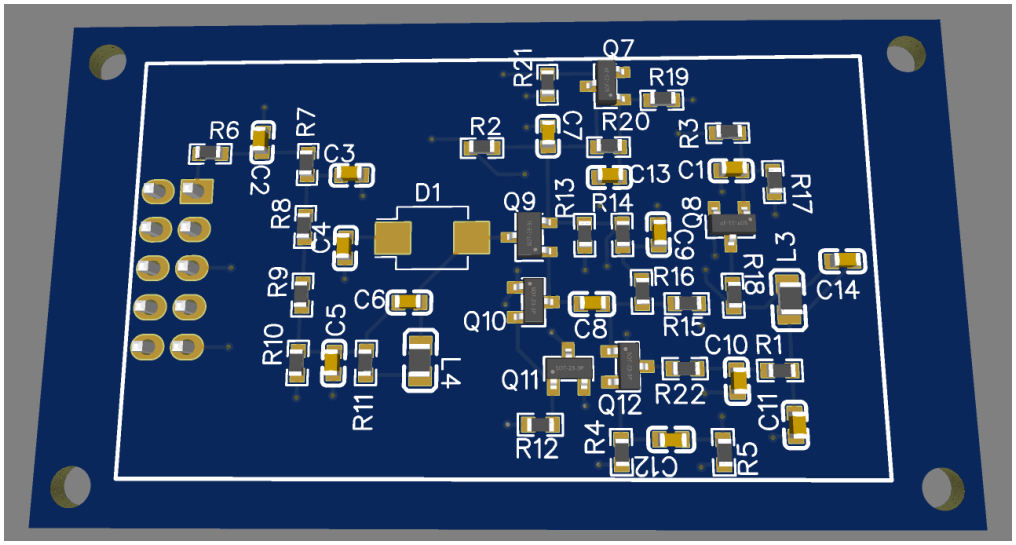


Fig. Finalized 3D-Model of Top Layer

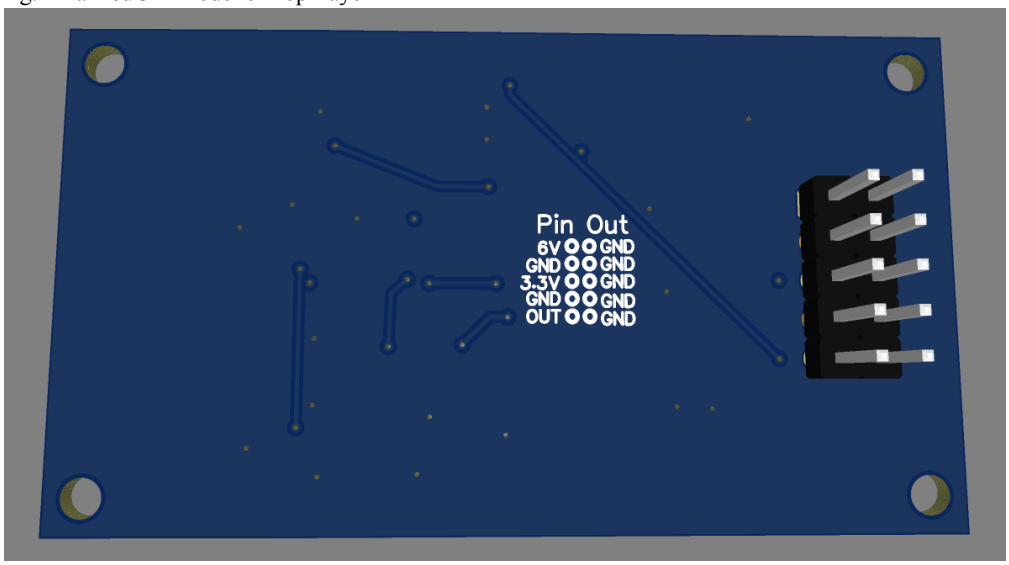


Fig. Finalized 3D-Model of Bottom Layer

In conclusion, the receiver was designed with the original receiving circuit, as well as an amplifying circuit, to make the signal more legible by the measuring oscilloscope, both of which were designed to filter for the crystal frequency. A 10-pin header was added to connect both 6V and 3.3V power, ground, and the output signal.

Emitter V1

The initial Emitter wiring diagram shown below is a model from the sponsor onto a PCB Design software called EasyEDA. In this diagram we have boxes indicating the crystal oscillator and the stages within the tuned driver from left to right. Also, the labels in the last three boxes are the special inductor parts that are being used within these stages to keep the signal consistent.

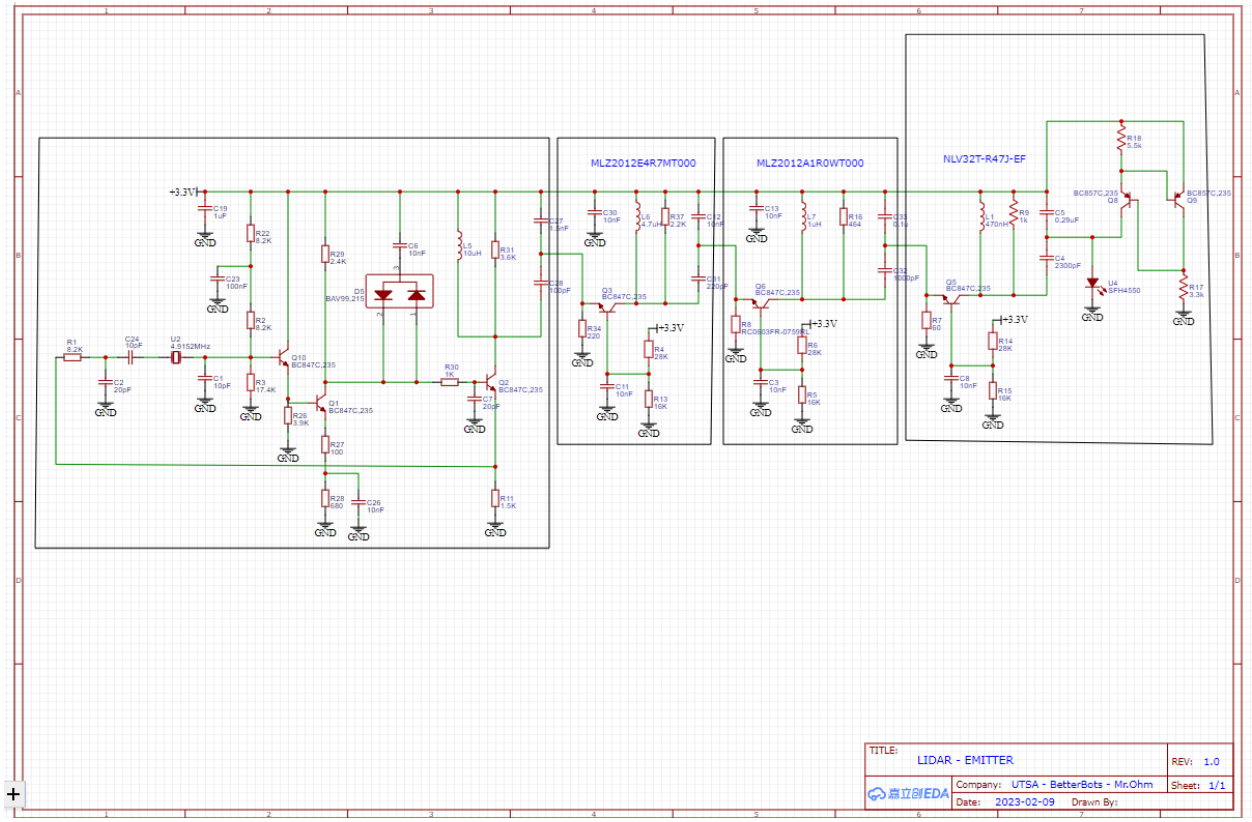


Fig Initial Emitter PCB Wiring Diagram

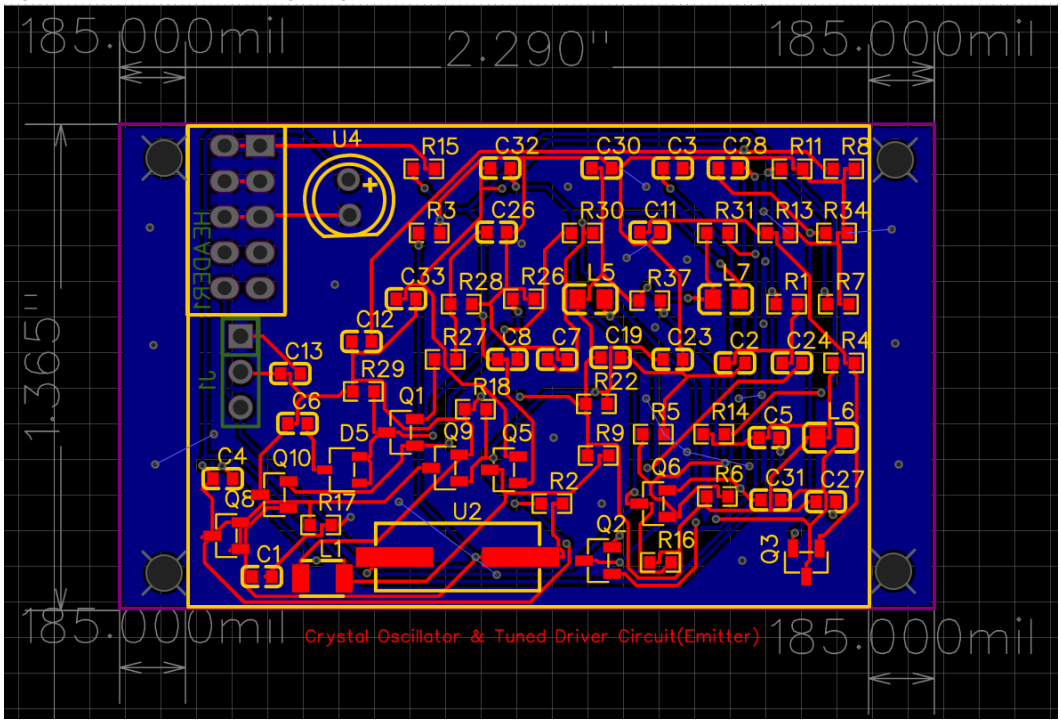


Fig Initial Emitter PCB Schematic

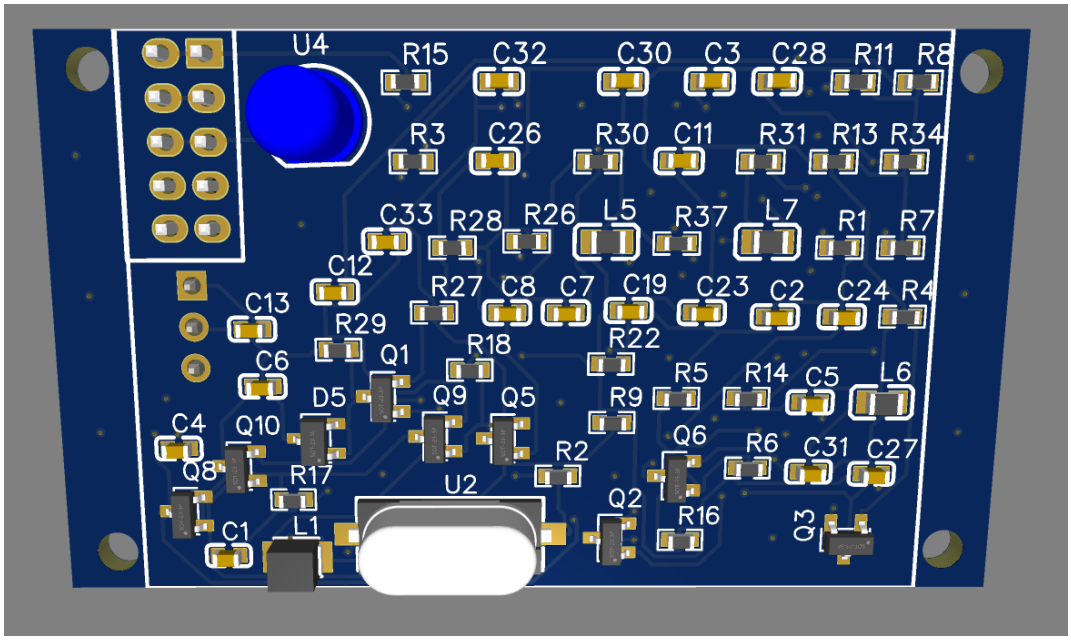


Fig Initial Emitter PCB 3D Schematic

Emitter V2

In version 2 of the Emitter, not much has changed other than correcting the placement of the components in the PCB schematic and updating the values as well.

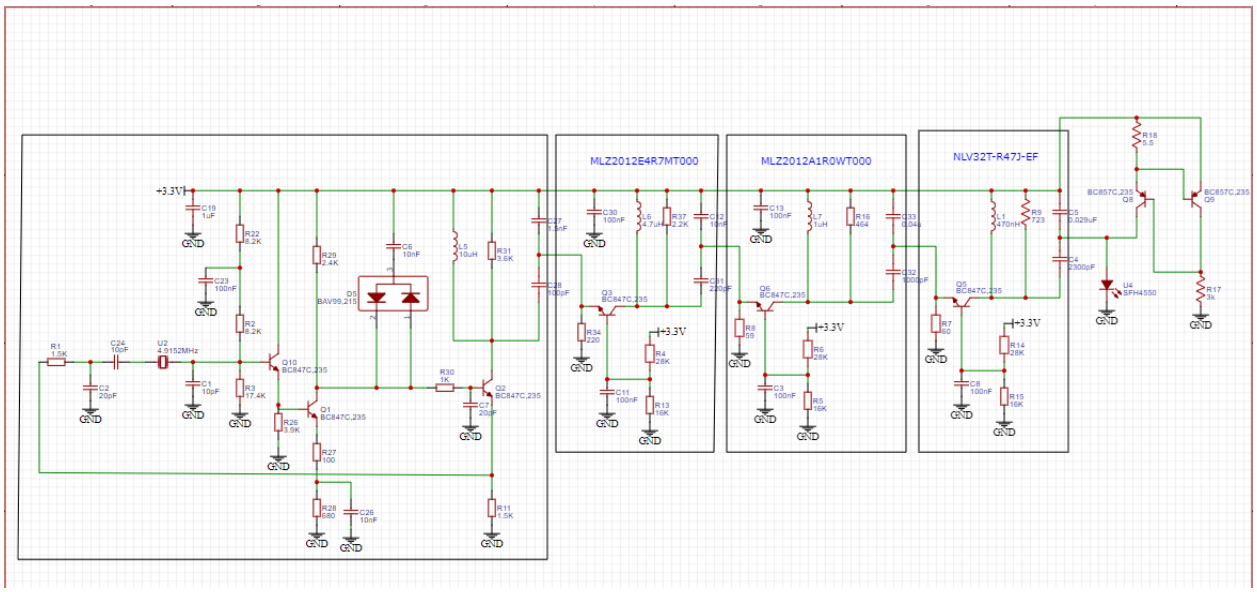


Fig V2 Emitter Wiring Diagram

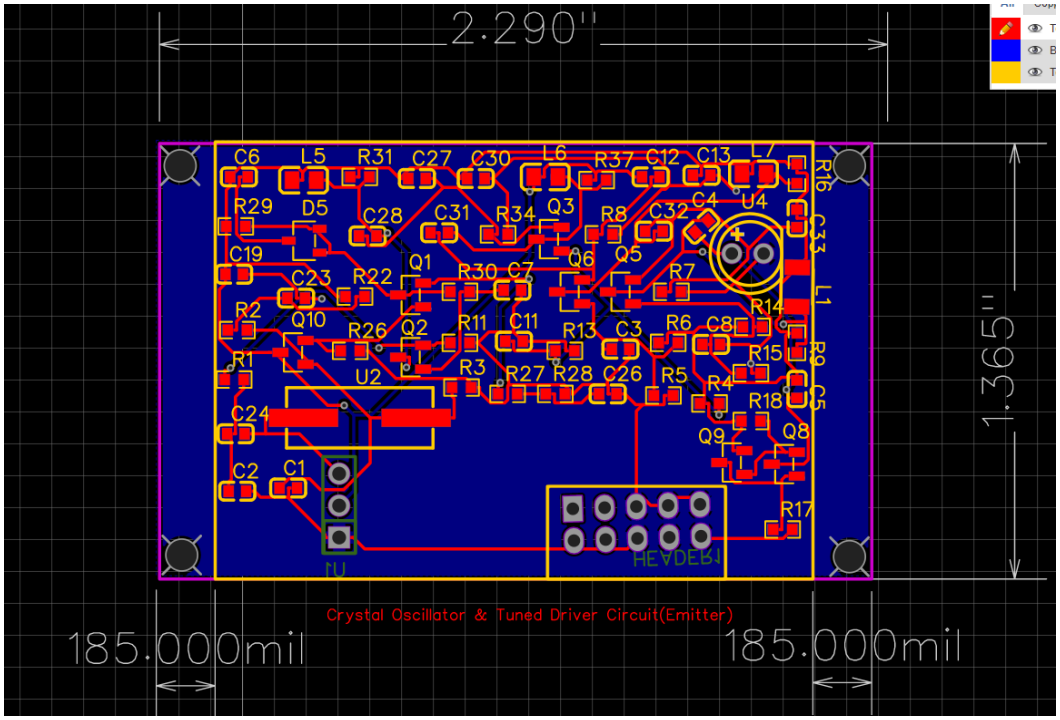


Fig V2 Emitter PCB Schematic

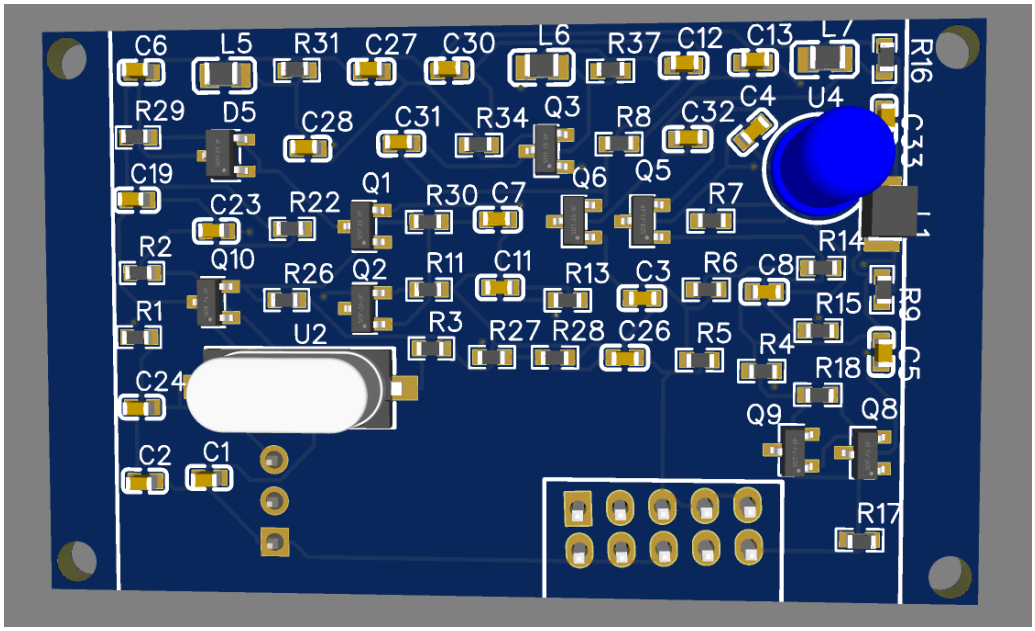


Fig V2 Emitter PCB 3D Schematic

Emitter V3

Version 3 of the Emitter was still fixing the placement of components with the addition of the wiring to switch the emitted signal between the function generator and the crystal oscillator; while also adding the additional ground layer to the PCB schematic.

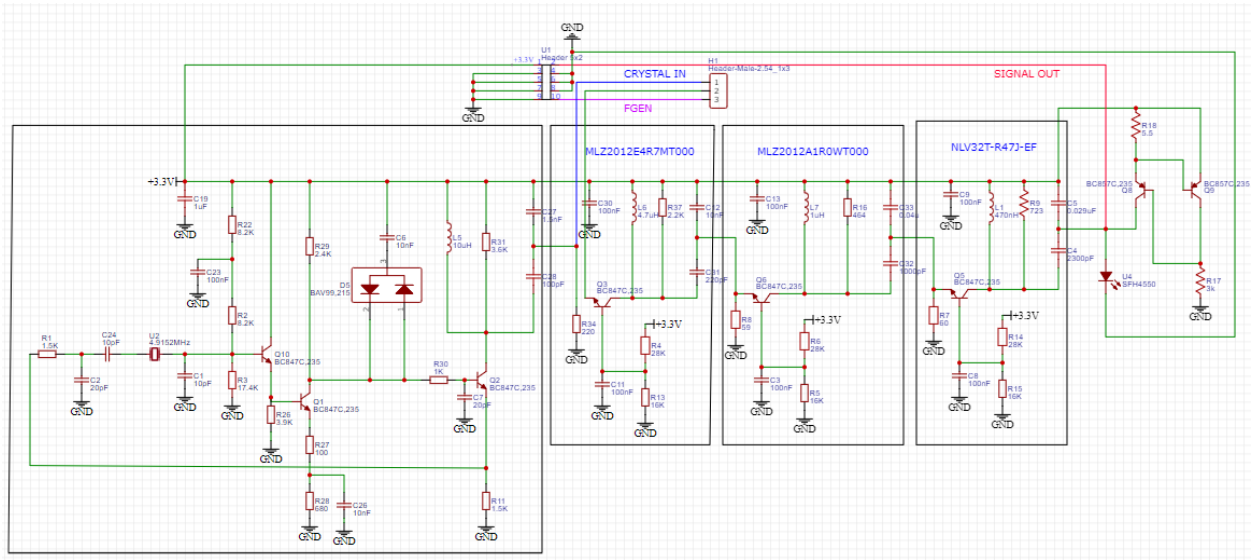


Fig V3 Emitter Wiring Diagram

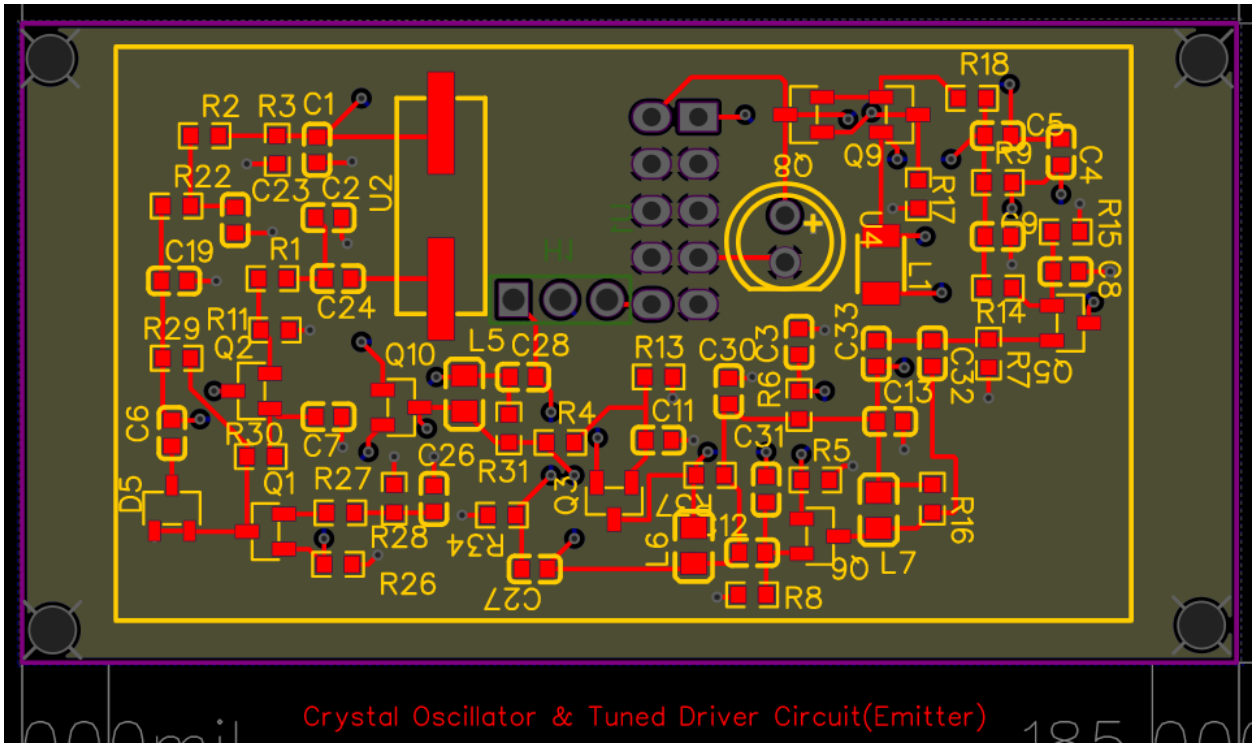


Fig V3 Emitter PCB Schematic

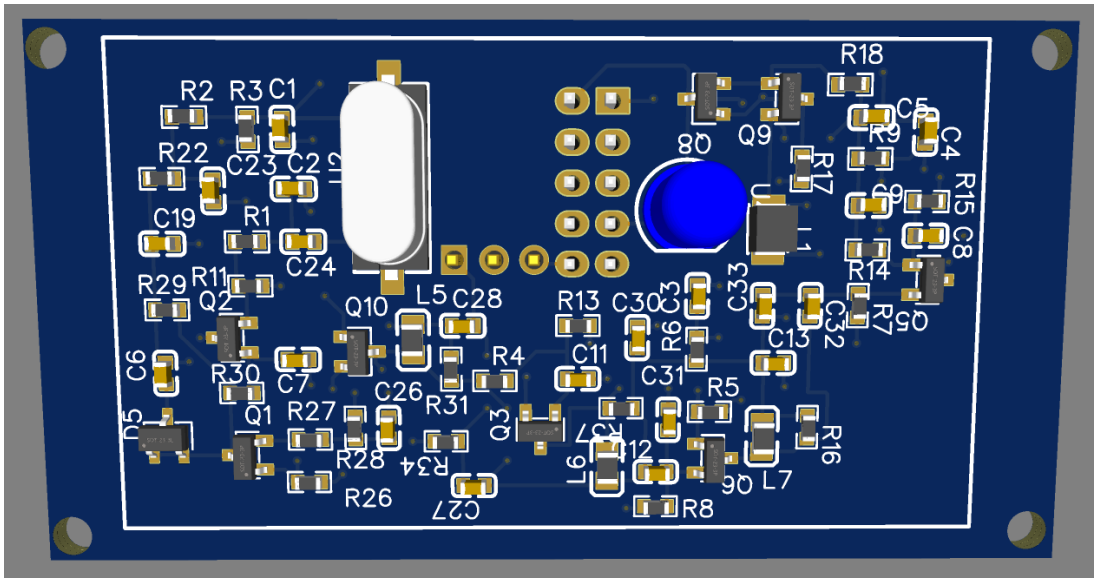


Fig V3 Emitter PCB 3D Schematic

Emitter V4

In version 4, the output reference signal was added to create that reference point from the collector side of the transistor within the crystal oscillator.

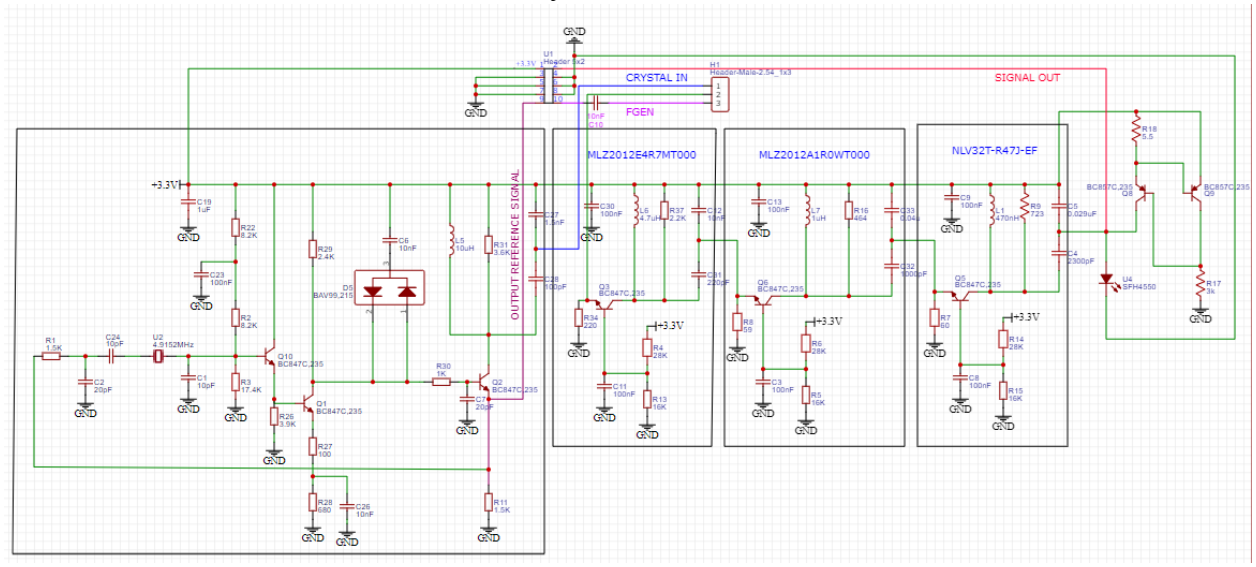


Fig V4 Emitter Wiring Diagram

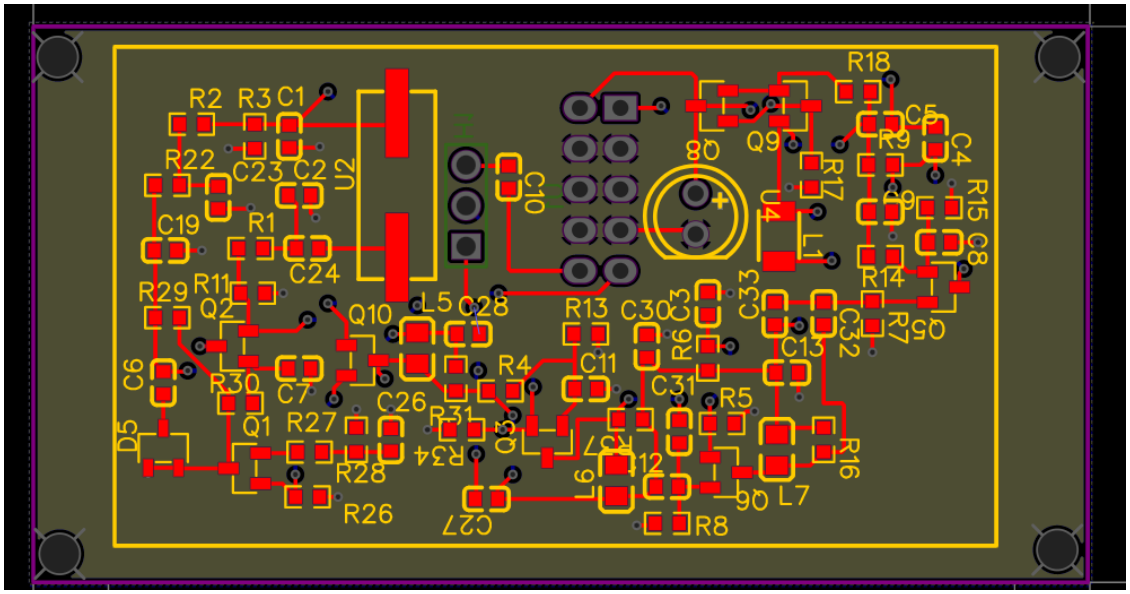


Fig V4 Emitter PCB Schematic

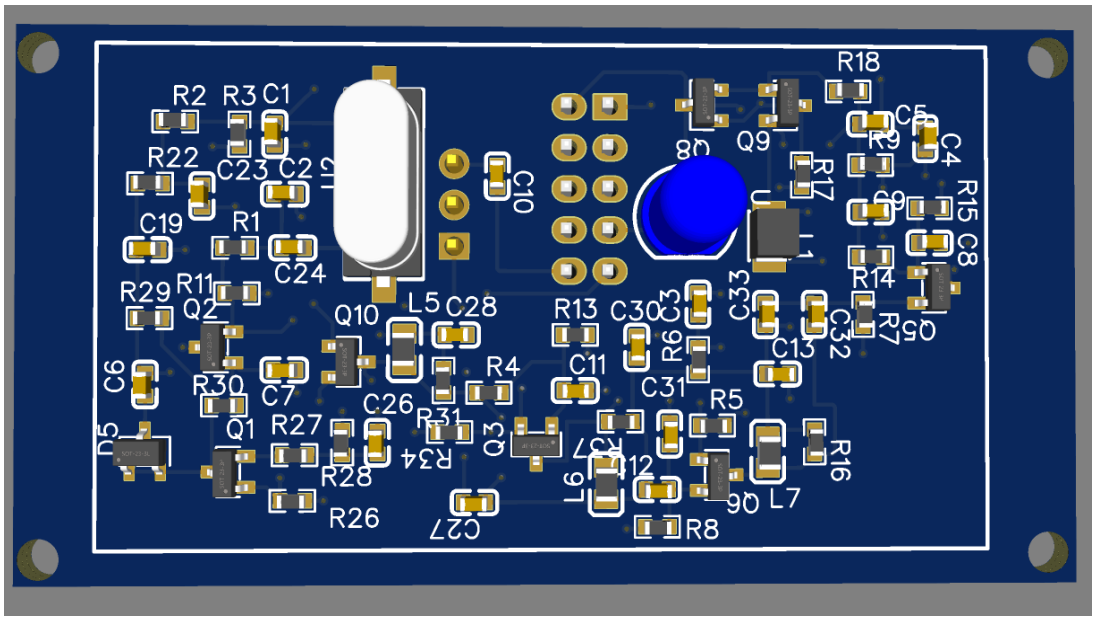


Fig V4 Emitter PCB 3D Schematic

Emitter V5

Lastly, version 5 was to update all the final values of the components and to switch them from the manufacturer parts to the basic parts in order to avoid the surcharge fee on those components before sending them out to be fabricated.

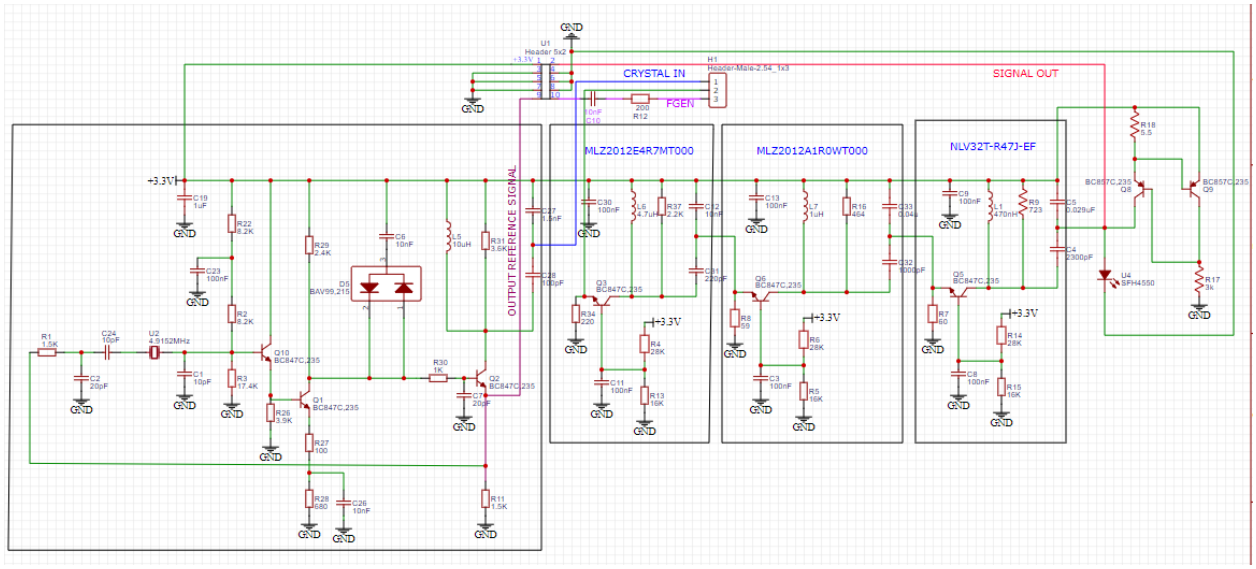


Fig Final Emitter PCB Wiring Diagram

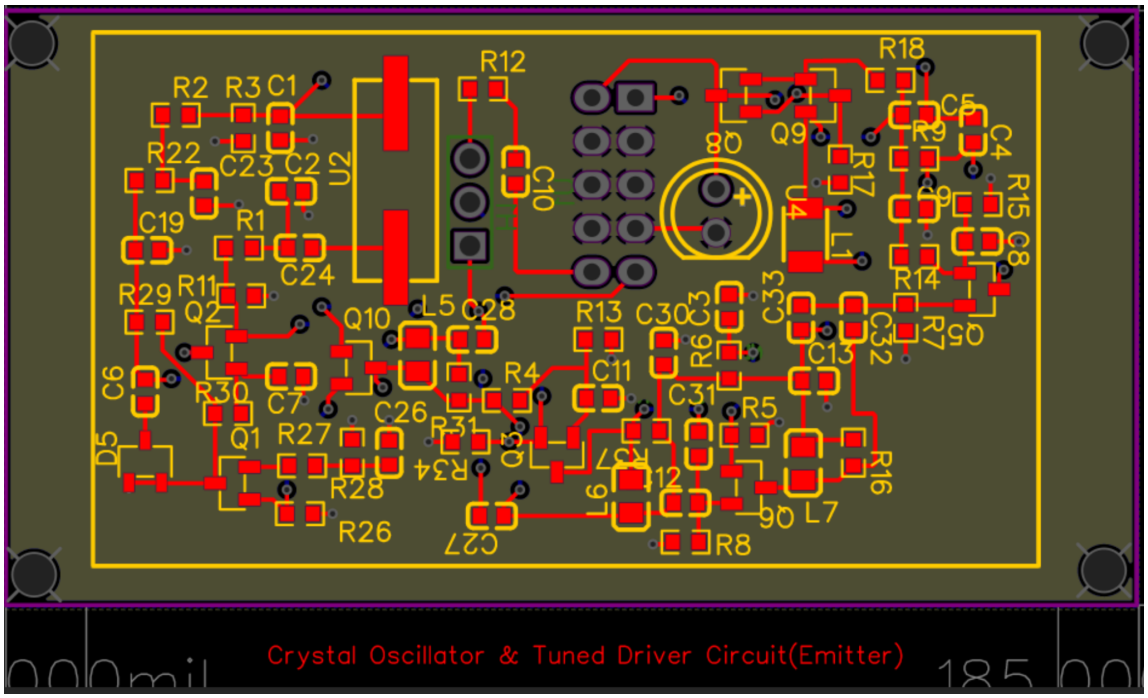


Fig Final Emitter PCB Schematic

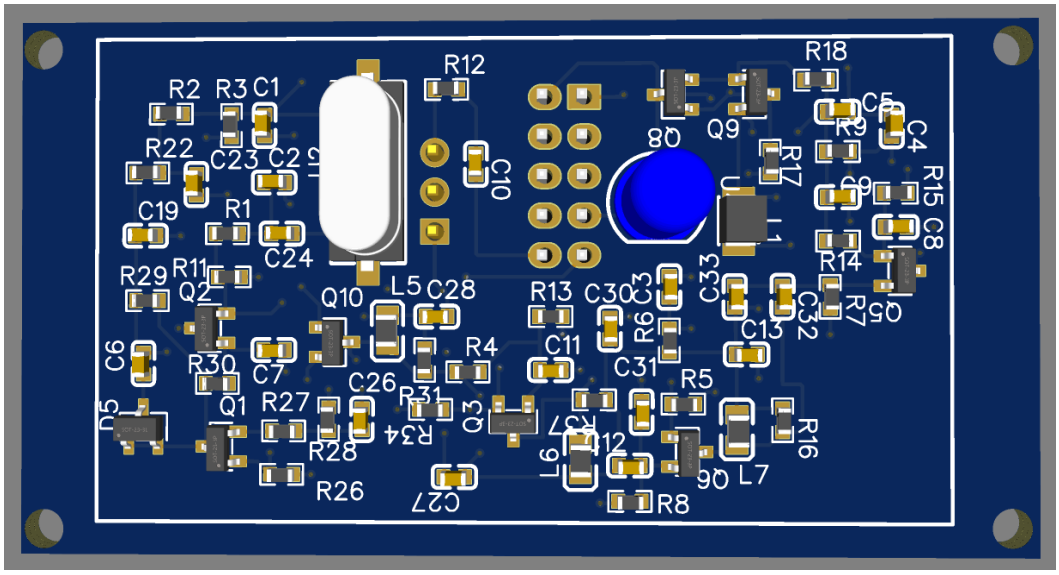


Fig Final Emitter PCB 3D Schematic

The emitter was then designed to separate the crystal oscillator function from the tuned driver in order to allow operation both from the crystal oscillator circuit as well as from a function generator, using a 3-pin header with a jumper to connect to one input or the other, while a 10 pin header was used in order to connect ground, 3.3V power, and the measurement tools, while voltage dividers were built onto the transistor bases to lower voltage to their required operating values. Also, the steps taken were to reorganize the whole schematic through multiple iterations to make sure no ground lines were touching the components, which would disrupt the current going through it. In addition, learning how to use the easyeda software or any PCB design builder and manually building it rather than auto-routing is essential because of the way certain components are rotated and implemented in a certain fashion.

3.0 Validation and Verification

Validation and Verification began shortly after the completion of the receiver protoboard. The first test was to make sure that the receiver did, in fact, receive the signal, and to test this, an infrared LED was soldered to a load resistor and a shunt resistor, in order to control and modulate the current so the LED would be getting a 200 mA peak to peak with average 100 mA to maximize its output. This small circuit was attached to a function generator providing the necessary voltage difference and offset at 5 MHz, and the LED was pointed at the photodiode, resulting in this graph:

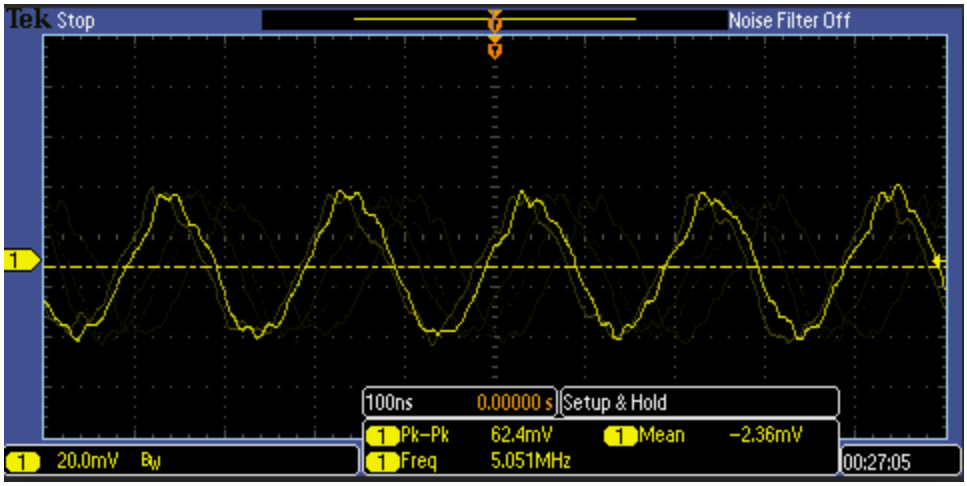


Fig 3.1 5 MHz Receiver Protoboard Test

In order to test the circuit filtration, this was then tested again with a 5 kHz signal, and finally, a control with no signal, pictured below:

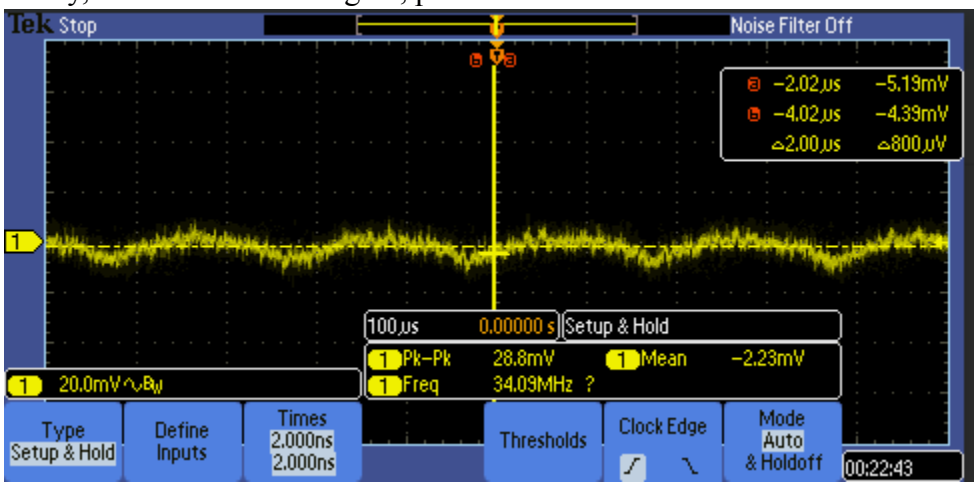


Fig 3.2 5 kHz Receiver Protoboard Test

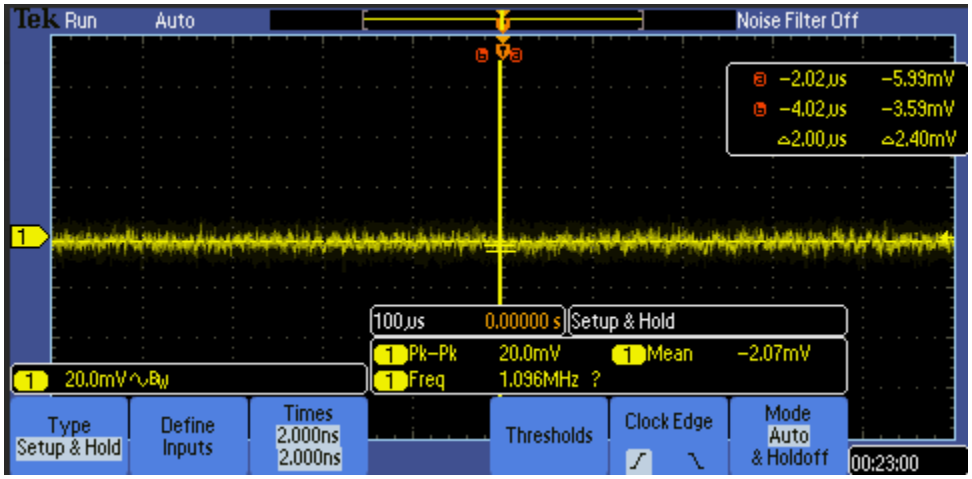


Fig 3.3 Receiver Control Test

While these tests showed that the circuit did filter out unwanted signals to some degree, they revealed another issue: induced emf which was the prime culprit behind the size of the signal shown above in Fig 3.1 Steps were taken from switching to the spring-like smaller oscilloscope probe heads, and using the more precise National Instruments VISA program to measure, as well as adding an amplifier to the receiver to increase the true signal out of it. When the experiment was repeated with the simple LED circuit, using the 5 MHz signal and pointing vs not pointing the LED, with both original signal out and the new amplified signal out these graphs were obtained:

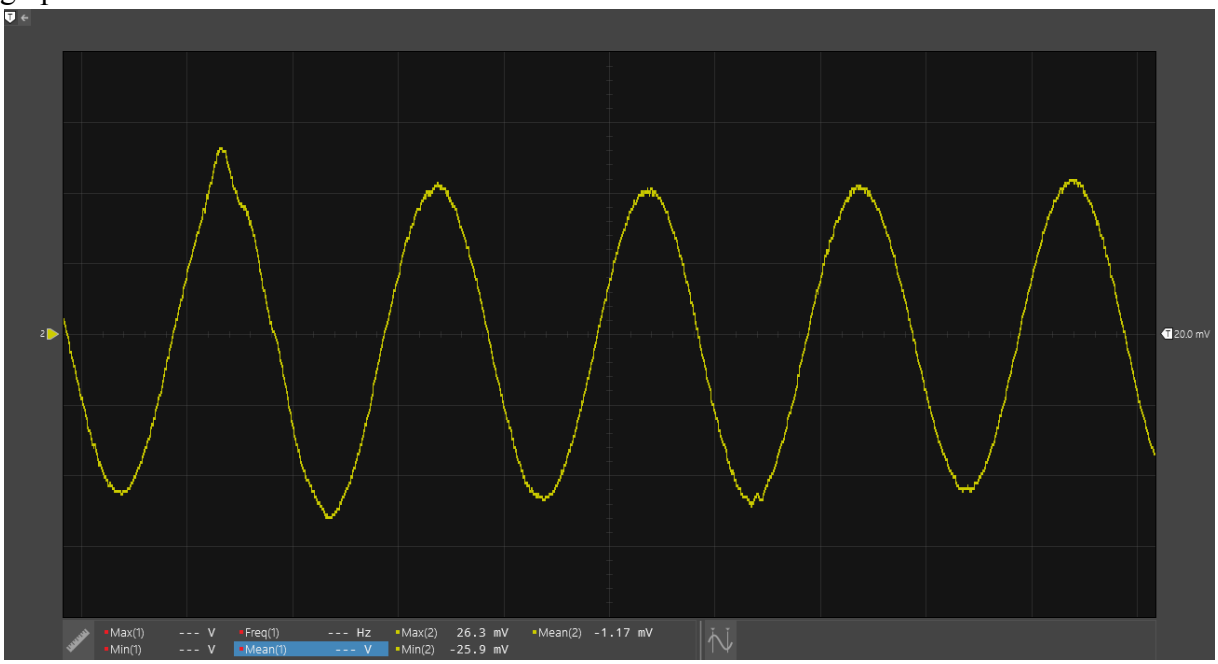


Fig 3.4 Receiver Test 2 - LED pointed at photodiode

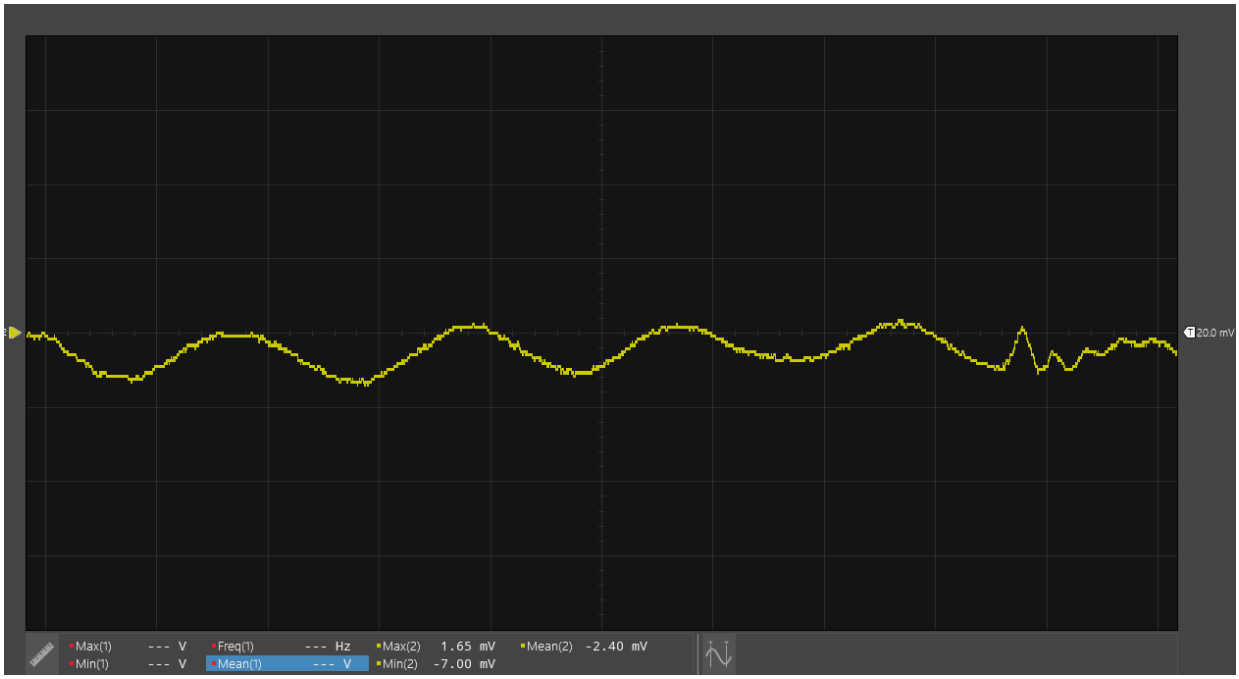


Fig 3.5 Receiver Test 2 - LED covered but powered to test induced emf

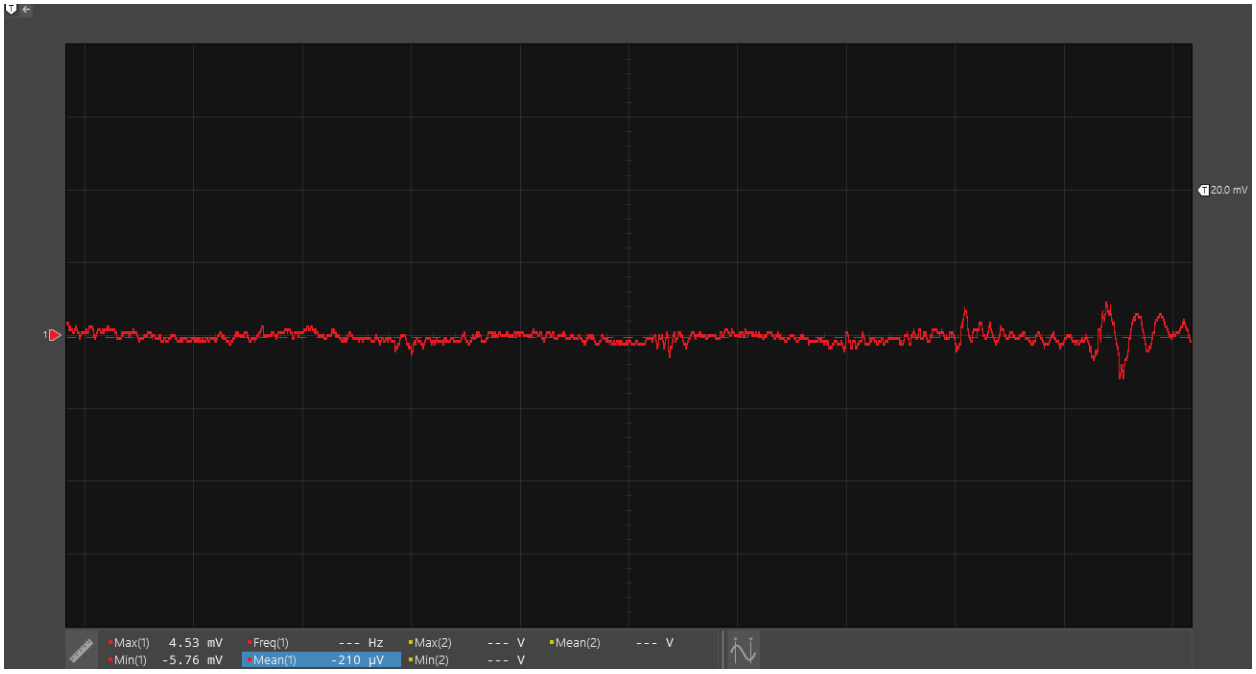


Fig 3.6 Receiver Test 2 - Unamplified signal with LED pointed at photodiode

Once the Emitter protoboard had been completed, a shunt resistor was added in order to find the proper function generator frequency for input into the emitter in the case of a non-functioning crystal oscillator, in order to find the proper offset and voltage, with the best results being obtained with a 50 mΩ resistor being a 1V peak to peak offset by 9. 45 V in order to make the current a 0 to 200 mA peak to peak signal.

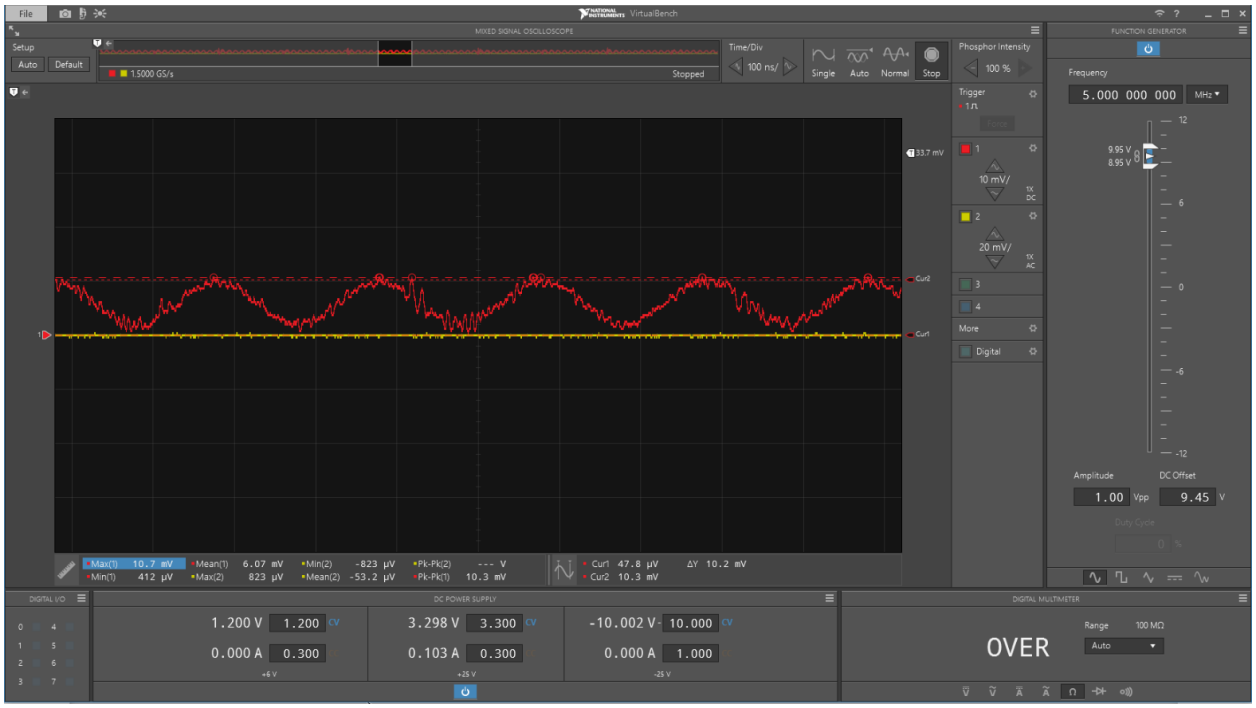


Fig 3.7 Emitter FGEN Test

Next, a full function test was conducted, pointing the emitter LED at the receiver photodiode, and then blocking the path between them with the function generator still running to give a control, where the yellow channel 2 signal is the signal in the receiver.

Fig 3.8 FGEN Emitter to Receiver Control

Fig 3.9 FGEN Emitter to Receiver Test

Upon reception of the PCB's, these tests were repeated, with a large improvement in the elimination of EMF thanks not only to the tighter system, but the move from the function generator to the on-board crystal oscillator. Once all functions were established, optimization began with the use of the function generated simplified LED circuit to frequency sweep across the photodiode, to see what its intrinsic filter's resonant frequency was, and to make sure it matched the 4.9152 MHz the crystal oscillated at. This sweep began at 1 MHz due to being the and would go all the way up to 5 MHz. While it was initially planned (and conducted) to go to 10, all attempts peaked below 5, and as such no frequencies above are of imminent relevancy. Pay close attention to the scale used, as the signals start very weak, but still seem large thanks to the smaller scaling.

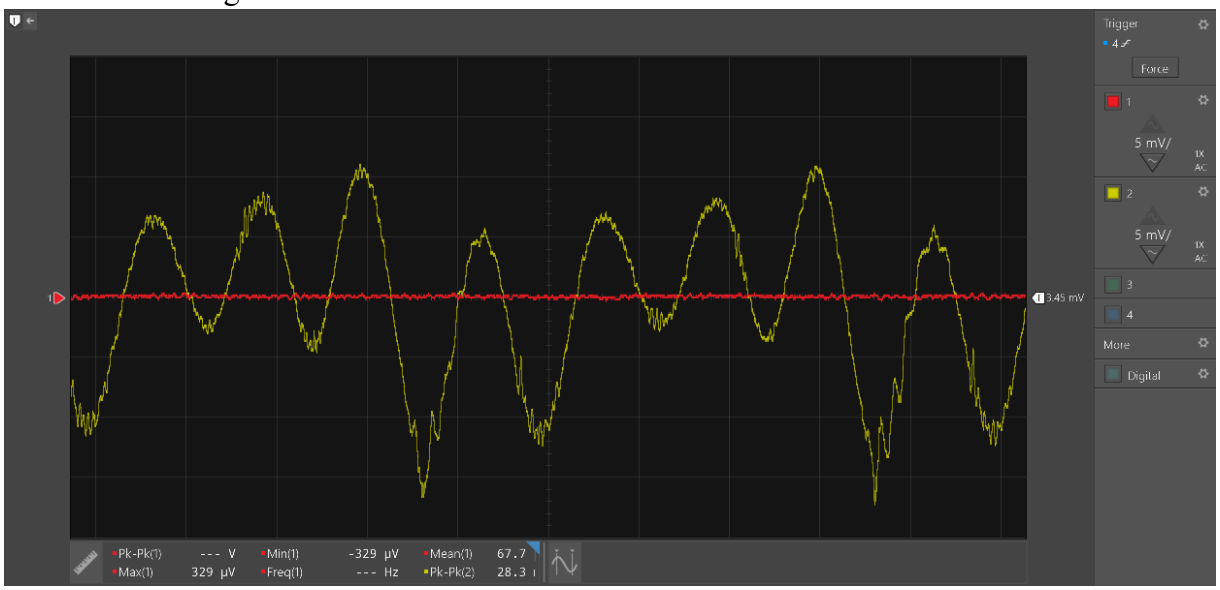


Fig 3.10 Frequency Sweep 1 MHz, scale 5 mV

Fig 3.11 Frequency Sweep 2 MHz, scale 20 mV

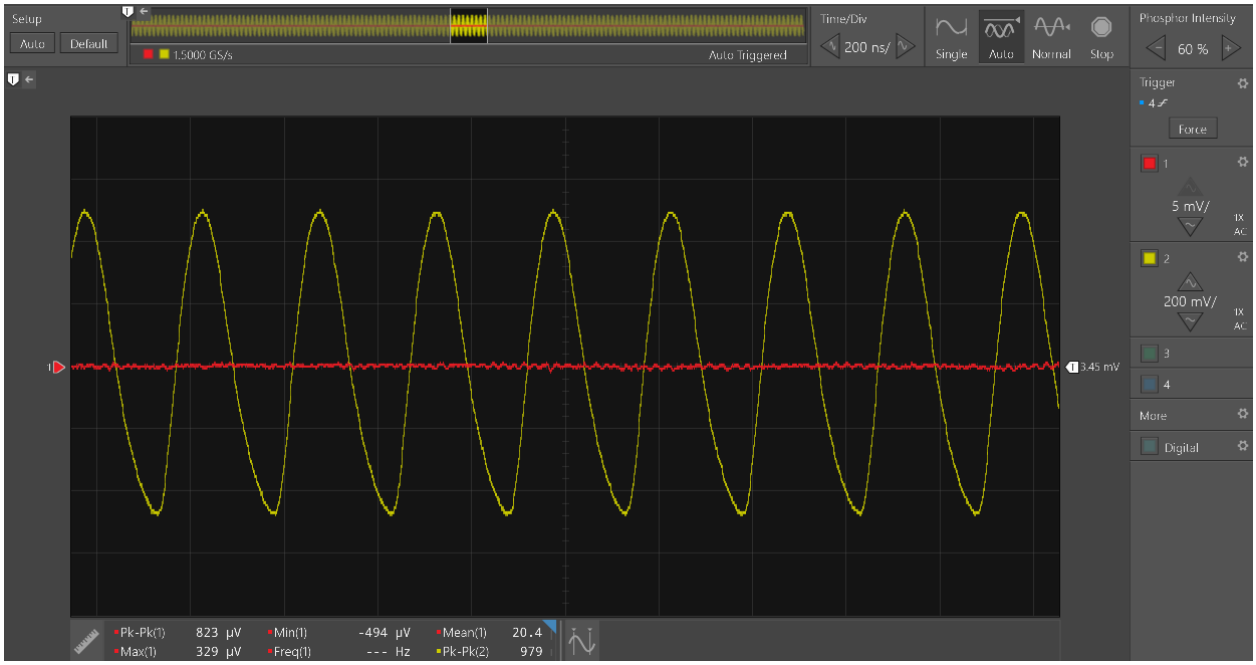


Fig 3.12 Frequency Sweep 3 MHz, scale 50 mV

Fig 3.13 Frequency Sweep 4 MHz, scale 200 mV

Fig 3.14 Frequency Sweep 5 MHz, scale 50 mV

From there, the frequency was stepped down from 4 MHz, eventually finding its peak at 3.8 MHz.

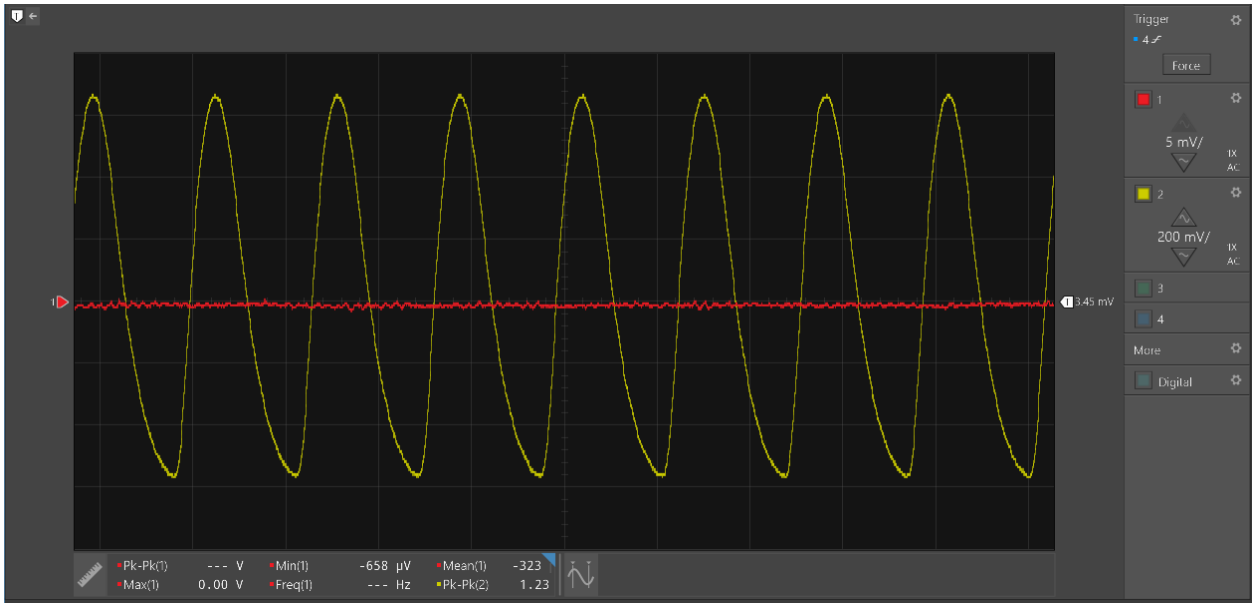


Fig 3.15 Frequency Sweep Peak, 3.8 MHz, scale 200 mV

At this point, it was determined that the best way to raise the resonant frequency would be to reduce the capacitance of capacitor C6 on the PCB. Tests were then repeated with a 10 pFcapacitor there to replace the designed 22pF, with the following results at its peak and at 5 MHz.

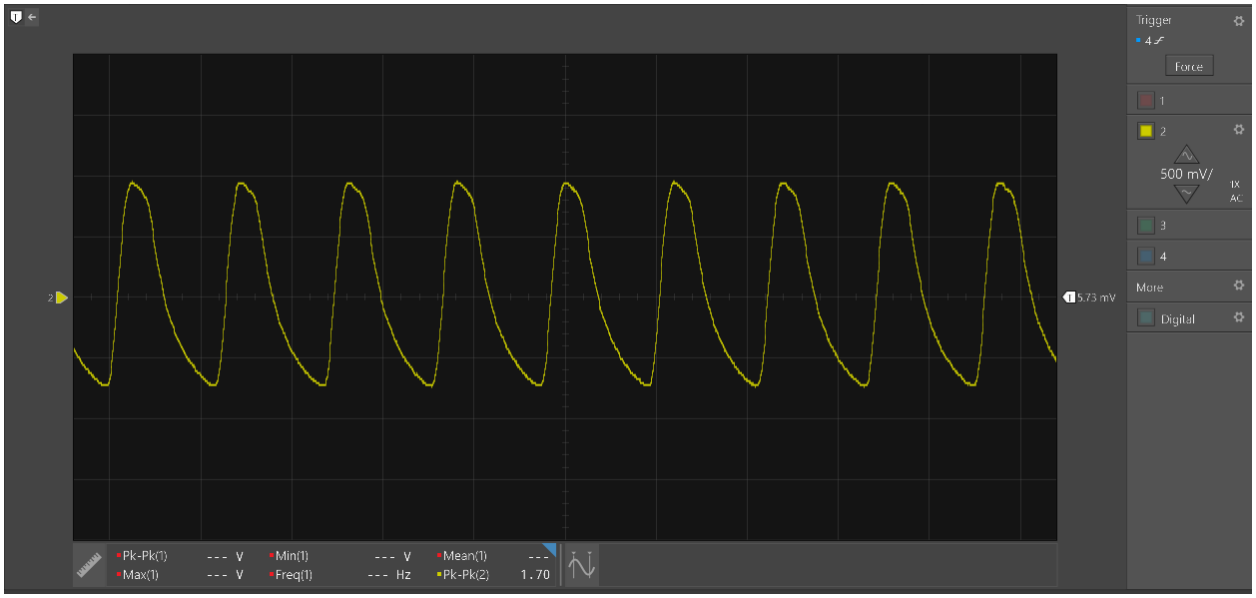


Fig 3.16 10 pF Frequency Sweep Peak, 4.2 MHz

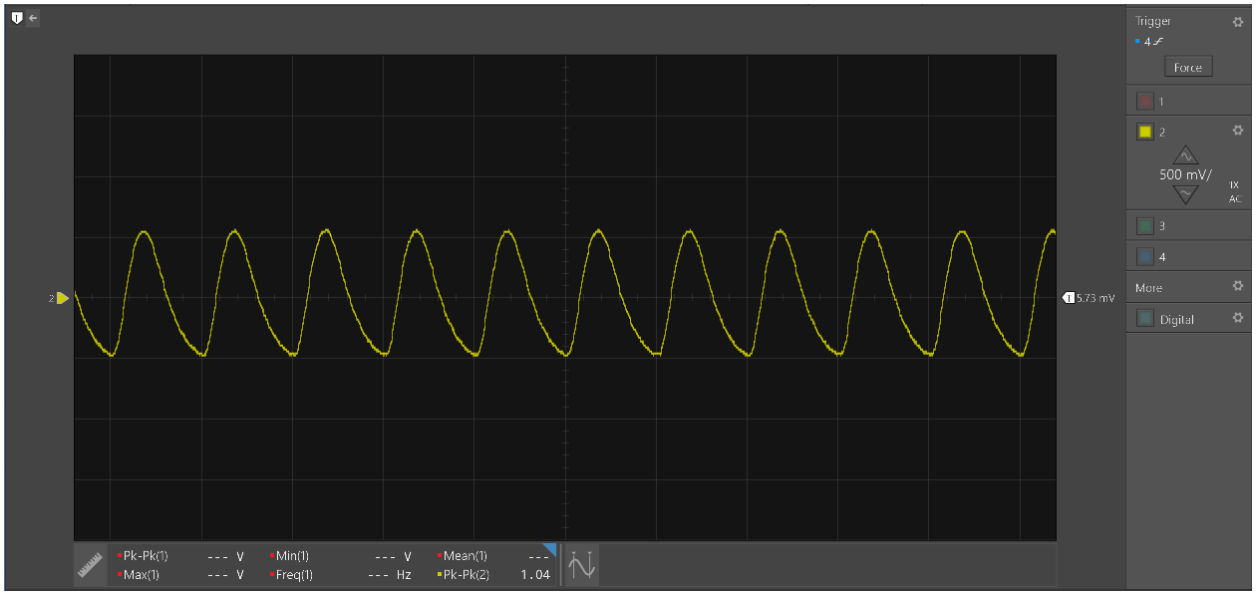


Fig 3.17 10 pF Frequency Sweep at 5 MHz

The capacitance still needed to be lowered, but without a 5 or lower pF capacitor, it was instead decided to use a jumper to connect two 10 pF capacitors in series to reduce the total capacitance to the desired 5 pF.

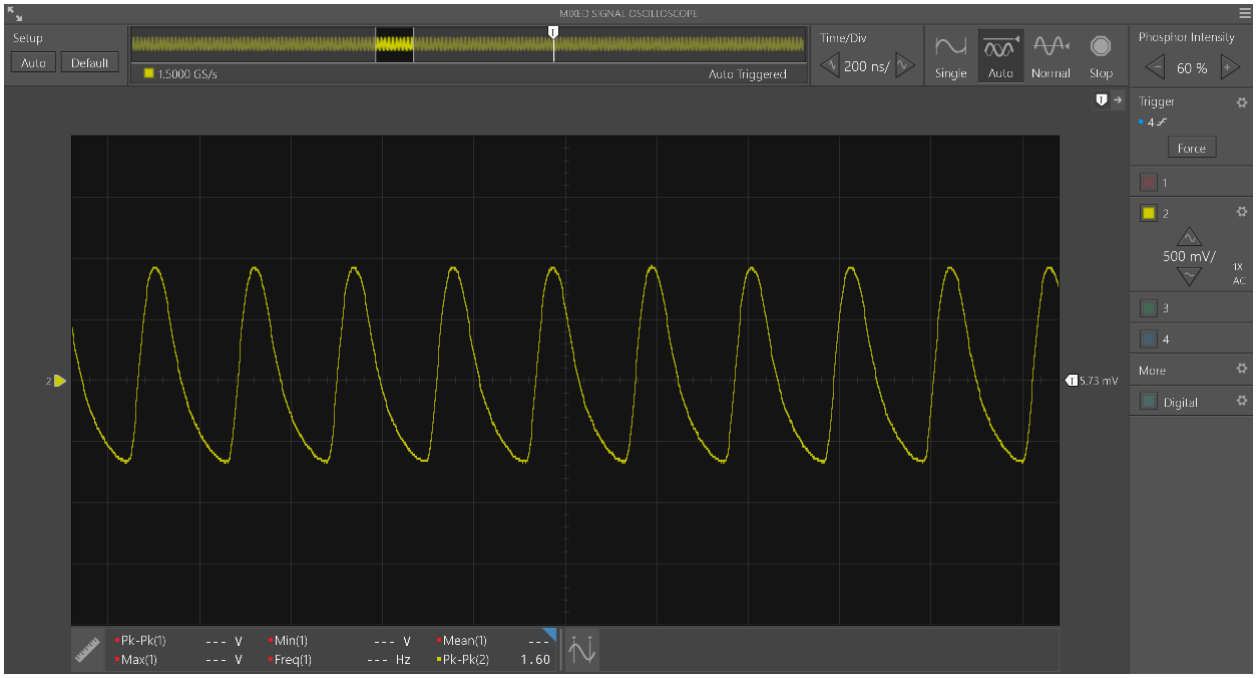
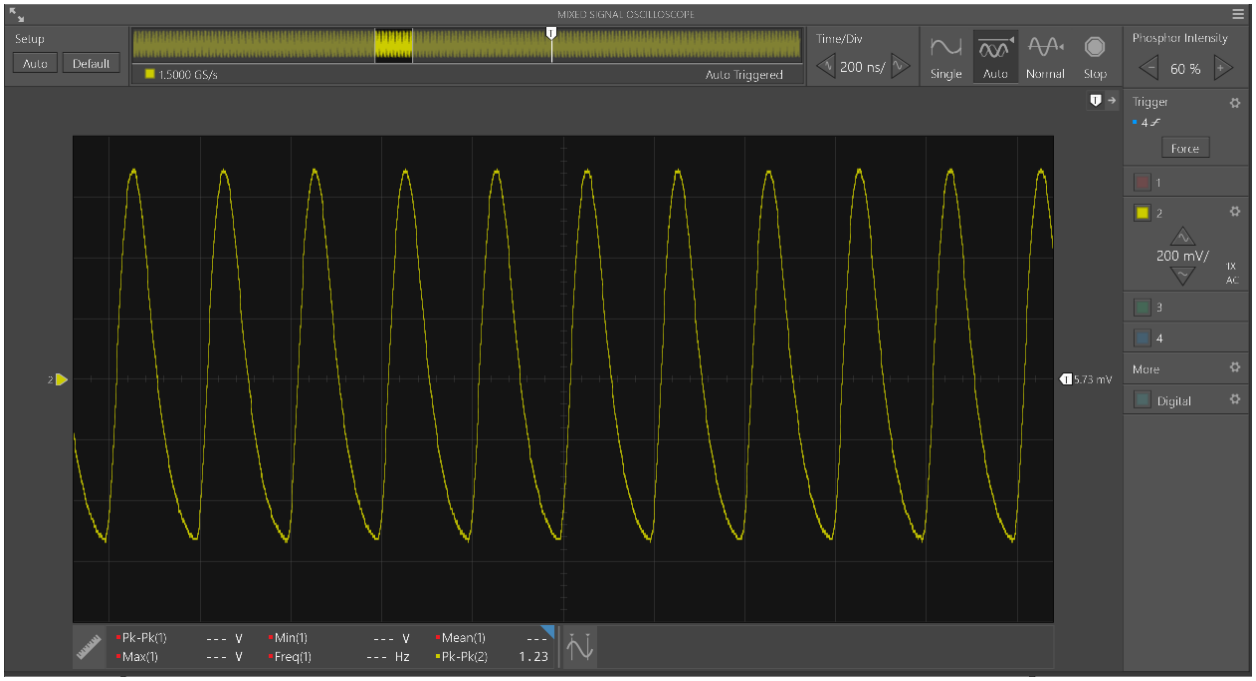


Fig 3.18 5 pF Frequency Sweep Peak 4.6 MHz, scale 500 mV



Fog 3.19 5 pF Frequency Sweep 5 Mhz, scale 200 mV

At this point, you may begin to notice a bit of a distortion to the 5MHz signal out of a sin wave, with clipping at the minimums. This became far worse however at 3.33 pF with a third capacitor added, where 5 MHz had become completely distorted.

Fig 3.20 3.33 pF Frequency Sweep 5 MHz

4.0 Frequency Modulated Waveform Analysis

During construction, the main idea was to use circuits' fundamental frequencies to filter out unwanted noise. As such, the circuits were designed around this equation in order to maintain the necessary frequency as amplifiers and tuned drivers were constructed for the chosen oscillator.

$$f_0 = \frac{1}{2\pi\sqrt{LC}}$$

For final rangefinding calculations, the primary focus was on the waveform of the signal being sent and received.

T = 1/f

This was a 4.9152 MHz signal, meaning that it had a period of 238 ns. However, knowing that is not enough, as you must also know how much of each of those waves offset means how much time, in order to calculate the distance from the phase shift, which can be done with this equation.

$2(x/c)/T = \phi/2\pi$

In this equation, x/c, or distance divided by the speed of light, gives the time it took to travel a single period. Light travels 1 foot in 1 ns, and it takes 2 to get from the emitter to the object, to the receiver if the object is 1 foot away. This means that each foot of distance traveled is 2/238 of a wave difference, which calculates to a 3-degree phase shift from light travel alone.

Meanwhile, the oscilloscope takes samples of 100 us, meaning there are approximately 417 waves to average across for one single phase difference.

Thanks to the hundreds of full waves collected in a single polling, the average of signals will be incredibly close to the means, and the partial waves at the beginning and end of the sample will be negligible, so the correlation equation simplifies to:

$$Correlation = \frac{Cov\ (x, y)}{\sigma x * \sigma y}$$

$$r \approx \frac{\int_0^{2\pi} sin(x)sin(x+\phi)}{\int_0^{2\pi} sin(x)^2} = \cos(\phi)$$

Where r is the correlation of the two waves, by taking the arccos of r, received from the covariance of the two waves, the phase shift can be obtained.

The software included in the initial function of this LiDAR module was merely a proof of concept for the final robot, as ultimately, the project's goal is to have students program the bot after reception rather than be shipped with any specific running code.

The software was designed by first reverse engineering a program designed to simply plot a single oscilloscope output in order to determine the functions of the necessary PyVISA package and find the best way to store the waveforms as they were picked up before removing the plotting and integrating a read of the second channel, and including comparison and covariance function to calculate phase shift before calculating distance from the shift between the two waveforms.

The final code worked by first beginning a connection with the measuring oscilloscope, setting it to optimal settings, and readying it for use. At this point, it would begin an infinite while loop, where each loop, the oscilloscope would be polled for a single one hundred microsecond sampling of the waveform across the LED and the receiver output.

Using settings gathered from the oscilloscope, the program would then run an error checking function, measuring the magnitude and frequency of the received signal, assuring that the magnitude was not too big, thus indicating the bot that the LiDAR module will be mounted on would be about to crash into something, leading to an alarm function, or that either the magnitude was too small or the frequency did not match the emitted frequency, indicating that the signal being received is more noise than signal, and thus should not be measured due to lack of accuracy.

If the signal passed these checks, it would then be compared to the emitted signal in order to calculate the phase shift by correlation, using the arccosine of the covariance between the signals.

A polynomial curve fit function, after sampling hundreds of samples at set distances, was used to determine a range-finding equation, using the phase difference as an independent variable to receive an estimated distance from the LiDAR module.

While the program did all of this, it would also keep track of stored distances in order to simulate the multiple orders of magnitude faster and more frequent sampling rate of the device

when not limited by oscilloscope polling like in its final placement in the bot. Pressing the A button on the keyboard would thus make it give a calculated value of the stored average of distances shown, while pressing the C button would clear the memory, allowing for a new average to be taken. Finally, the infinite loop could be killed at any time with a press of the space bar, closing out the oscilloscope so it was ready to receive new commands. The entire code is illustrated in the flowchart below.

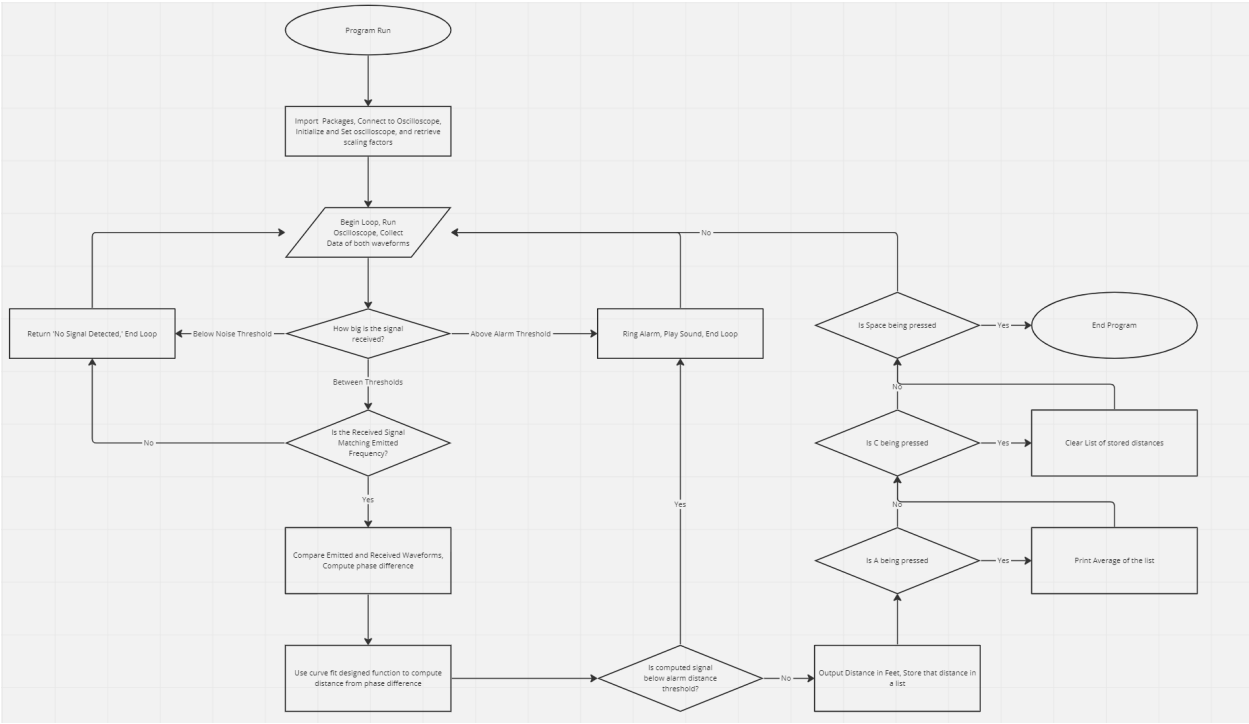


Fig 12.2.1 Software Flowchart

5.0 Comment and Conclusion

The design endeavor was an overall memorable experience and a tremendous learning opportunity, where a team of five students who never previously knew or worked with each other was able to come together with a virtual sponsor to implement and improve upon a complex product while bringing our own ideas and meeting course requirements. The task at hand was never simple, with very complex principles from day one that required our full attention and dedication to learn the vast complex circuit properties. The team was quick to separate into teams and individual tasks that contribute to the overall completion of the project, divided based on skills and interests. The following strategy was quite effective in ensuring the full use of everyone's time; the first team was the LtSpice Simulation team, the printed circuit board team, the software task, the housing task, and the Protoboard task shared by all members.

Designing, simulating, and upgrading the circuit design was the first major task of the LtSpice team. One of the most challenging aspects of the project where the team had to often work back and forth with the team sponsor to learn new tricks and discover simulation issues within components. LtSpice and the complexities of circuits offered an enormous challenge, but finally, after many iterations and inspections, the design was at a healthy point. The full team had originally intended to move along to a printed circuit board from the LtSpice design, but with much consideration and in order to meet a prototype requirement, a pivot was made to build a prototype.

The protoboard required a great deal of research and full team cooperation in order to have the correct components and assembly tools. The protoboard took the greatest amount of time compared to all other project tasks and allowed the discovery of some key flaws in the design. In the implementation of the board, many members struggle to keep up with the progress and current state of the design. The whole process was slightly rushed, and not every member had the patience or time to solder on surface mount components on tiny boards. The same issues persisted into the verification testing of the protoboards, where members were often confused about current testing steps, setup, and how to interpret results. The flaw may have stemmed from the division of labor and how each individual member may not have all the pieces to see the full picture, as in not knowing the exact simulation reference value to compare to the protoboard result. Nevertheless, these issues were addressed, and the team worked on transcribing available testing data and parameters and sharing the information in a regular meeting or in our folder system.

During this whole process, the printed circuit board team was hard at work on their own individual task of translating the simulation circuits into printed circuit boards. Another set of new skills needed to be developed once again, with regular meetings with the sponsor to confirm ideas and current designs. Having the dedicated roles and team did have one advantage once the error was discovered in the receiver protoboard, where the returned signal did not have a great enough amplitude to be recorded effectively by the oscilloscope. The team met, and it was decided the LtSpice team needed to create a new design to be simulated and implemented on the protoboard for real-world testing. Once complete, the design was passed to the printed circuit board team to place on the final product. Allowing the efficient use of each member's learned skill to quickly make a change that greatly influenced our final product. The software application was quick to follow after the completion of protoboard testing, where the implantation was more or less straightforward. The same was for the housing design, where measurements were transferred from the printed circuit board to the computer-aided design software, with very few errors or mistakes. Altogether the team worked together on all aspects, bringing in our own ideas and concerns for each other's work in a respectful and considerate manner. Ultimately developing a LiDAR device that we all share pride in accomplishing, demonstrating a product that ultimately supersedes the sponsor's expectations while building upon everlasting skills and friendships.

6.0 Advancing Development

Looking forward in future development endeavors of the Mr. Ohm LiDAR module, our team has identified key areas for future innovations in lensing the LED and photodiode, adding the second LED and Multiplexer, and full integration into a field programmable gate array FPGA. Our team also believes that the tuned driver can be refined even further to even include coupling capacitors in between each LC tank and common base amplifier. Regarding the parts of the LiDAR module, basic components found from the common parts library are best used for the implementation of the final PCB design since it can greatly reduce the labor cost, and eases the searching for specific manufacturer parts, specifically for the capacitors and resistors in the module.

7.0 Team Member Roles

Efrain Rivera - LTspice lead and Prototype Board Designer:

Efrain Rivera worked on the circuit design on LTspice along with Brandon Colon. Designed the infrared LED current source by using LTspice. Designed and simulated a new pre-amp stage to amplify the receiver's output signal. Designed the physical layout of the Receiver Protoboard and fully built the Emitter Protoboard. Designed the 3D-Housing for the LiDAR Module that was 3D printed and used for our PCBs during testing.

Abdul Muizz Atanda - PCB Designer:

Abdul's task was to design the Emitter PCB. Through multiple iterations, Abdul designed the Emitter PCB to fit the needs of our sponsor and the LTSpice team. Abdul ensured that the team had the correct surface-mounted components for our PCBs during development and for when the team needed to solder the protoboards and PCBs. From the LiDAR module, Abdul-Muizz learned a lot from the PCB design software EasyEDA, specifically the placement of components properly in order to reduce long wires. Also, this experience has helped tremendously in learning the different aspects of building a PCB board such as adding additional ground layers and how to efficiently place components manually rather than relying on the auto-route tool.

Justin Moreno - PCB Designer and Archivist:

Justin's main contributions were his work on developing the Receiver PCB and archiving the team's work. This project allowed Justin to learn not only how to draft PCBs, but also know and utilize helpful techniques used in industry.

Brandon Colon - Project Manager

Originally focused on the LtSpice simulation, working hand and hand with Efrain to learn and tackle our original task of circuit improvements. Once complete, the team moved into implantation, where it was quickly realized the team needed a manager to check on tasks and to keep the team on the right track. I then dedicated myself to creating divisions of labor for the main and sub task of the group while regularly checking progress and upgrading timelines. Throughout the process, I worked to ensure that each member knew their task goals and how their task fits into the full system; while maintaining full corporations and an even workload for all members.

Ryan Johannsen - Programer

Ryan Johannsen designed the layout of the protoboards, which aided the assembly team in building a quick, effective, and accurate design. Ryan had been in the lab most days working on the functionality of the protoboards and PCBs, such as repairing any damage to them. Ryan did various tests throughout the semester, such as AC sweeping, on the receiver to test its optimal frequency in order to compare it to its ideal values. Ryan worked on the programming side by obtaining and graphing oscilloscope data. Afterward, Ryan worked on analyzing the data from the protoboards and PCBs into a coding script. One example of validating the protoboard was by running the testing program and using its sampling rate to optimize our code.