Laser Arcade Machine

Final Report

SDMAY22-24

Client

Joseph Kenkel

Advisor

Dr. Neihart

Team Members

Joseph Kenkel

Ashley Robertson

Jonah Stoffer

Mark Kavars

Tyler Beveridge

Morgan Luecht

Zack Larson

Executive Summary

Development Standards & Practices Used

(ANSI/ANS 10.3-1995 Standard for Documentation of Computer Software);(ISO/IEC/IEEE 26514)

IEEE 2030.2.1-2019 - IEEE Guide for Design, Operation, and Maintenance of Battery Energy Storage Systems, both Stationary and Mobile, and Applications Integrated with Electric Power Systems

Summary of Requirements

- Backend server with database
- Multi-platform app development framework
- Blaster
    - Batteries
    - IR emitter
    - LED visuals
- Target
    - IR receiver
    - LED visuals

Applicable Courses from Iowa State University Curriculum

- CPR E 288: Embedded Systems I: Introduction
- CPR E 388: Embedded Systems II: Mobile Platforms
- E E 201: Electric Circuits
- E E 230: Electronic Circuits and Systems
- E E 285: Problem Solving Methods and Tools for Electrical Engineering
- E E 321: Communication Systems I
- E E 333: Electronic Systems Design
- S E 319: Construction of User Interfaces

New Skills/Knowledge acquired that was not taught in courses

- IR Protocol for sending and receiving data.
- Reading Datasheets of components
    - Deciphering Graphs
- Calculating Steradian and power loss with time
- Proper battery management

Table of Contents

1 The Team

1.1 Team Members

Electrical Engineering	Software Engineering
Joseph Kenkel	Tyler Beveridge
Ashley Robertson	Morgan Luecht
Jonah Stoffer	Zack Larson
Marcus Kavars	

1.2 Required Skill Sets

- PCB Design
    - At a minimum every target and blaster is going to need a Printed Circuit Board (PCB) to connect all of the hardware involved for the individual topics. On top of this, there will likely need to be a pcb
- Wireless Communication
    - Wireless communication will be important for our hardware to interact with each other from greater distances
- Sensor Signal Manipulation
    - We must be able to take the information gathered from our signal and get it to a place that our app can receive and process the information.
- Front End development skills
    - Use known languages to develop phone/ web applications
- Backend Development Skills
    - Develop app that is able to pull data from a piece of hardware to communicate scores to players
    - Store scoring data in database

1.3 Skill Sets Covered by Team

PCB Design
- Jonah Stoffer, Ashley Robertson

Wireless Communication
- Joseph Kenkel

Microcontroller Development
- Mark

Front end Development
- Morgan

Back end Development
- Zack

Stack Development
- Tyler

1.4 Project Management

Scrum and other principles within Agile methodologies to produce an effective and efficient project timeline.

1.5 Initial Project Management Roles

Team Leader: Joe Kenkel
Laser Generation Lead / General Hardware Lead: Jonah Stoffer
Backend Development Lead: Zack Larson

2 Introduction

2.1 Problem Statement

Laser arcades are entertaining to a wide range of people and demographics. Currently if someone wishes to shoot lasers at targets they need to go to a specific location that hosts that sort of entertainment. We intend to develop a mock system that will demonstrate the possibilities of a portable laser arcade machine.

Ideally the hosts of this portable laser arcade would be able to set targets around a room, turn on the system, and pass out laser blasters. The users would utilize an app on a tablet that provides a UI for the system and controls the game.

2.2 Requirements & Constraints

2.2.1 Requirements

2.2.1.1 Software

- The backend is hosted on a server on a Springboot application
  - Capability to connect, send, and receive data with the front end app
  - Capability to connect to the raspberry pi and receive data from it
- The database should hold key player stats and info
  - Capability to connect to the backend in order to transfer the player's data

    to the front end to display the players current stats/score to them
    - Capability to automatically update and refresh player's score
    - Secure and reliable database that we can only access
- A multi-platform app development framework
    - Capability to compile to different operating systems
    - Work on android tablets that will go with the system

2.2.1.2 Hardware

- Blaster
    - Power
        - Battery Powered to allow portability
    - Emitter
        - IR Transmitter
    - Visuals
        - shots/ammo left
        - color denote player
        - Spring loaded chamber to reload
        - Denote when fired
- Targets
    - Targets
        - 2-4 targets
            - To be placed 10-20 feet away from blaster
        - Varying in size
            - Area of 16 cm^2
        - Receiver IR
            - Identify each blaster
        - Visuals
            - LED denote
                - When to shoot
                - When hit
                - Which blaster hit it

2.2.2 Constraints

- Budget
    - $1000 - prototyping
    - $200 - Final Product
- Time
    - 10 weeks of design and planning
    - 15 weeks of development

    - Request and receive specific parts for the project in a timely manner
- Scope
    - Mock up of the system
        - Future development could turn this into a marketable product
        - The entire system including the laser blaster, targets, and controller should be extremely portable
            - Plan on having a case to store the system for easy transport

2.3 Engineering Standards

(ANSI/ANS 10.3-1995 Standard for Documentation of Computer Software);(ISO/IEC/IEEE 26514)

IEEE 2030.2.1-2019 - IEEE Guide for Design, Operation, and Maintenance of Battery Energy Storage Systems, both Stationary and Mobile, and Applications Integrated with Electric Power Systems

2.4 Intended Users and Uses

- Mock project to demonstrate capability of this sort of system
- Intended use is to determine interest and marketability
- Find potential investors or buyers

3. Project Plan

3.1 Project Proposed Milestones, Metrics, and Evaluation Criteria

3.1.1 Evaluation Criteria

3.1.1.1 Software

- Results of using application features with test data is successful
- Number of bugs found in code/ Number of refactored commits to Git
- Peer and Faculty Advisor review of UI

3.1.1.2 Hardware

- Finished pcb for the target should be under 16 in^2
- The target will have an 8 in tolerance from the center to register a hit.
- PCB schematic and layout are peer reviewed

3.2 Risks and Risk Management / Mitigation

- Disconnect between the hardware and software

3.2.1 Software

- Running into issues connecting the frontend and backend of application
- Running into issues connecting the application to the raspberry pi and hardware system
- Login data security
- Management risks (no real leader established)
- Schedule Risk/Technical Risk(Never worked with some of the things)

3.2.2 Hardware

- Difficulty finding parts or parts running out of stock when we try to order them.
- Possible issues when we connect system to interface with the software and app
- Possible issues when we begin to actually use the laser
- Lithium Batteries exploding

4. Design

4.1 Design Context

4.1.1 Broader Context

The design problem that we are tackling, is in a world where laser arcades are stationary at sparse locales. We want this design to assist college students as well as family households in their efforts to entertain themselves and others in times of boredom. College students would find a decent price point product that would be able to entertain large groups of people at once. Families would be able to entertain children and guests without requiring significant setup. This project would fill their need for easy and simple entertainment.

4.1.2 User Needs

A household (consisting of parents and kids / group of friends) needs to be able to set up their own laser arcade without an internet requirement to entertain kids or a party.

College students need to be able to bring an IR blaster and targets to dorm rooms or common areas to entertain themselves at a low cost.

4.1.3 Prior Work/Solutions

The main inspiration for this project came from a laser arcade solution found at Bass Pro shops. At these locations targets are placed in a “hunting” environment with targets like animals, cabin parts, and misc small items placed around. Each player has a fake hunting rifle that probably shoots IR light. When shooting these targets effects occur on hit in the environment which add to the enjoyment of the arcade.

A product that could be considered similar are the video game variety of shooting arcades. One that specifically has some overlap is the switch party game with a speed drawing mini game. In that game players hold the controller in a specific “holstered” direction and on the start sound quickly point the controller up and shoot.

Pros of our product

- Our product will be easily portable unlike Bass Pro Shop’s larger stationary arcade.
- Our product will be more tactile with a physical laser blaster and physical target that is better than just shooting targets on a screen as a mini game.

Cons of our product

- The advanced effects in the environment that Bass Pro Shop has is much better than what we plan to do.
- Our product will not be set up to shoot towards other players like the switch mini game or laser tag games in general.
- You can not see where you hit.

4.1.4 Technical Complexity

This project is of sufficient technical complexity mainly due to the components required and the connections needed between those components. Below is the list of components of this project:

1. IR emitter (Blaster)
2. IR receiver (Target)
3. Target microcontroller
4. Blaster microcontroller
5. Mobile android application (frontend/backend)
6. Raspberry PI backend server
7. Aesthetic Cases for the blaster and targets (3d printing)

Connections required

1. The IR receiver connected to the microcontroller and decoded

2. The microcontrollers bluetooth connected to the Raspberry PI
3. Mobile application wifi connected to the Raspberry PI
4. Frontend of mobile application to the backend of the mobile application
5. IR emitter connected to the IR receiver through IR light

The IR requirements for this project match the current solutions like Bass Pro Shop. Connecting an app through wifi and microcontrollers to bluetooth exceeds solutions like Bass Pro Shop or similar video game solutions.

4.2 Initial Design Exploration

4.2.1 Initial Design Decisions

One design decision that we made was to use IR emitters and receivers to handle the shooting of the targets. On these we need to determine and obtain microcontrollers that would read Pulse Wave Modulated signals from the blaster and send those to the central controller.

A second design decision we made was utilizing a Raspberry PI for the central control of the game system as well as housing the backend for the system and app. The PI is a powerful enough mini computer to handle the computing power of both requirements and has built in wifi routing capabilities and bluetooth connectivity.

A third design decision we made was to use React Native for the native application. React Native is a multiplatform javascript framework for developing native apps and provides HTML like UI development.

A fourth design decision we have yet to make is the exact emitter and receivers to use. The emitter will determine what receivers we can use through its PWM. Additionally, we have to watch out for stock shortages that may require us to use components that are less desirable.

4.3 Initial Design

This section of the document will go over in detail the design plans that were created during EE 491. This section will also talk about some of the design changes that were made during the implementation phase. For a more detailed discussion on the technical changes that were made to the design, refer to the implementation section.

4.3.1 Design Visual and Description

4.3.1.1 Blaster

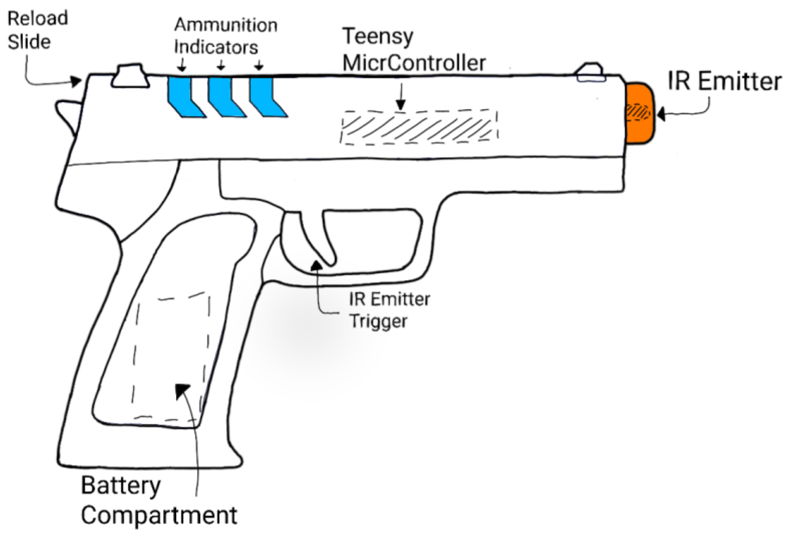

Figure 1: Blaster Visual

This was the initial sketch for the blaster of the project. As you can see in the original design, it had quite a realistic design that looked too realistic. The original plan was to have the battery in the handle of the model, the indicators on the top back of the model, teensy encased in the middle, and the IR emitter in the front of the barrel. The layout of all of these components stayed the same. The biggest change that was made for this project was the 3D model design. It was heavily modified to clearly resemble a toy to make sure that it cannot be easily mistaken for a real gun. The final design for this can be seen in the implementation section of the report.

4.3.1.2 Target

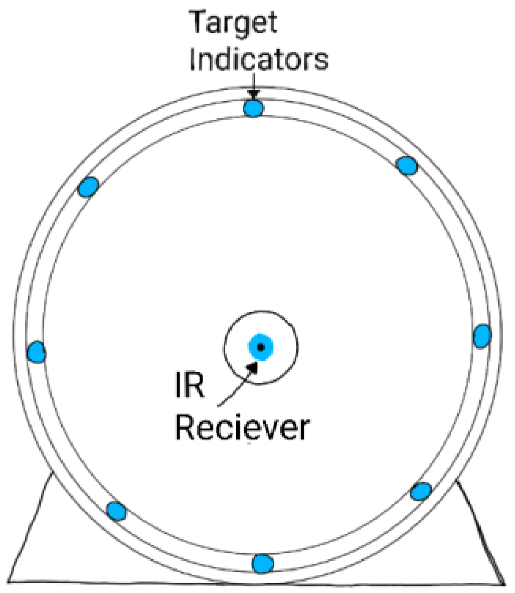

Figure 2: Target Initial Design Visual

Our initial design had the IR Receiver. We had wanted a range of target sizes as well. The idea for the shape of the target to be to not matter as much as long as the circuitry and battery would fit. We also planned to use RGB LEDs so that all of the LEDs would light the player's color. Our design from this semester has changed a few aspects of what we initially planned. The first thing we decided to change was that there would only be one size of target. This was because we decided that the receiver itself was not going to be anymore difficult to hit with the reduced size 3D printed casing. It was also decided that the RGB LEDs would be better off as static red and blue LEDs where they represent player one and two respectively so that we could have a game mode that allowed both players to hit the target. For the shape of the target we kept it similar to how we initially wanted it but imagined but was squared off the side sections to give more room on the inside of the case. The last change from the initial design was that we initially wanted the targets to be in a brighter color but when manufacturing them we chose to use the black filament because we only had enough red and blue filament for the blasters. This caused us to use filament that was what we had the most of when printing the cases to keep them all consistent.

4.3.1.3 Software

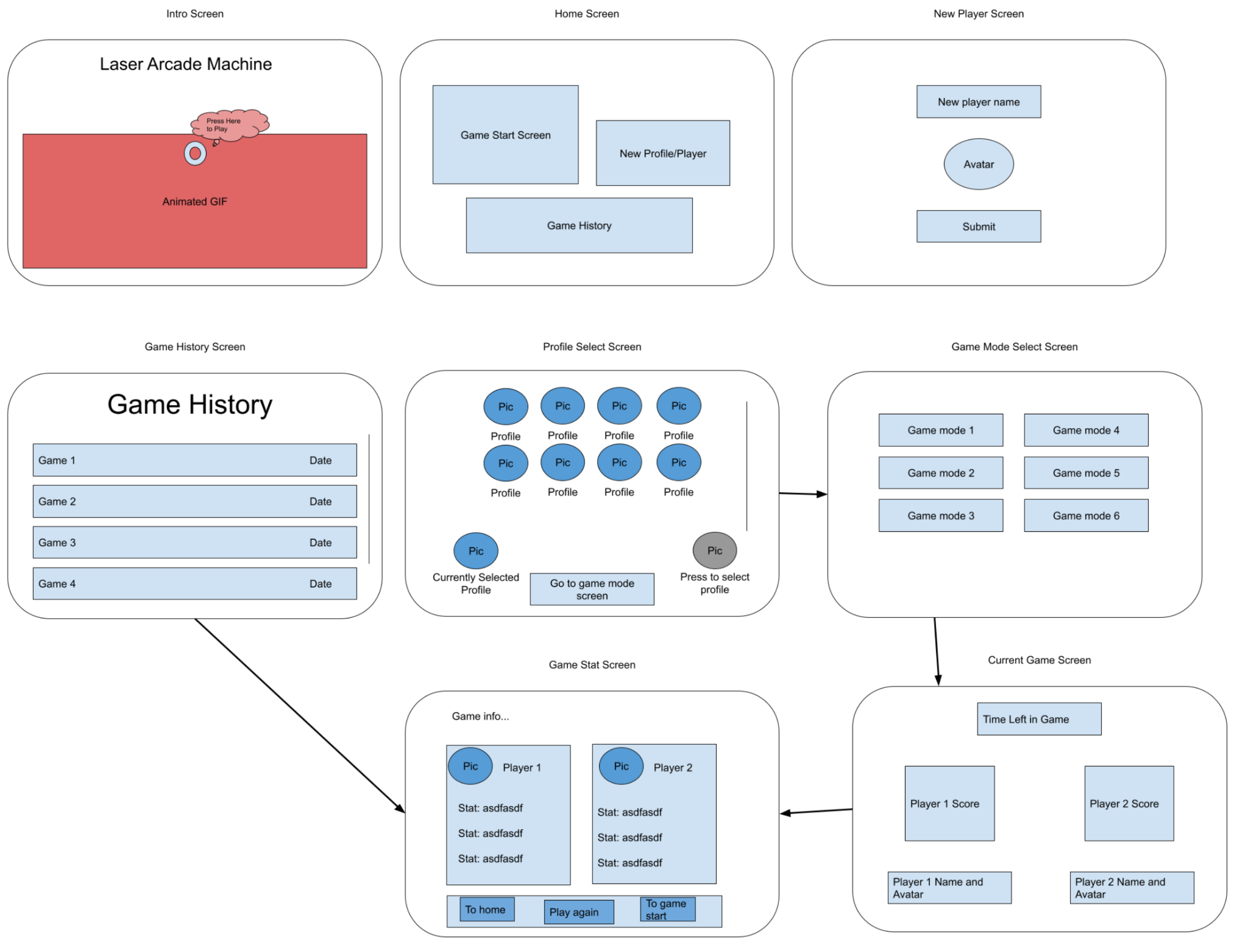

Figure 3: Application screen layouts

The UI for the application follows concepts from other multiplayer arcade style games. Most of the design (pictured above) matched nearly one to one with the final product minus some skipped screens and reduced game modes. For the structure of the bluetooth listener and server the only big change was the removal of websockets due to attempts being unable to make them communicate properly.

4.3.2 Functionality

4.3.2.1 Blaster

Last semester, we stated we were going to use a rechargeable lithium ion battery to power the blaster. This is no longer true. Because the battery would cost a lot of money, would be harder to implement, and was unnecessary, we instead decided to use replaceable 9V batteries. This battery powers the main processor for the blaster, a

Teensy LC. The Teensy would control the IR LED, some normal LEDs to indicate ammo, as well as a trigger and a reload button. Unlike the target, the blaster doesn't communicate to any other part of the system besides the target using a PWM wave. It doesn't need to have a bluetooth connection to the Pi, and doesn't need any information from anything else. It essentially acts like a fancy TV remote, sending IR waves to the target to interact with the system.

4.3.2.2 Target

Similarly to the blaster, we decided to use a 9V battery instead of a lithium ion battery to power the target. This will then power a Teensy LC microcontroller which controls the rest of the electrical components within the device. The target has RGB leds on the outside of the case that displays a variety of information to the user. When the target is ready to be shot by the user, the LEDs will shine white. Once the receiver receives a PWM signal from either blaster, the Teensy LC will detect which player shot the target and change the LEDs the color of the player who hit the target first. For any data that the blaster needs to either receive or transmit, it will do this through the bluetooth module that is connected via the Teensy LC module.

4.4 Technology Considerations

The IR beams are less precise than visible lasers. By choosing IR, you can not see where the shot landed and adjust your aim accordingly. Without the lazer, you can not shoot as far and the beamwidth is much larger causing the receiver to falsely trigger. On the other hand, we choose the IR because it is way less likely to blinde someone. In order to help deal with the issues caused by the large beamwidth, we chose an emitter with a low beam angle to make the range of the beam as small as possible.

We decided to wireless communicate from one board to another. This adds a layer of complexity over wired communication but also more freedom on where to put the target.

5 Implementation

In the previous semester we designed our systems, set out all of the requirements that we needed for system, got many of the key components working with each other (i.e: successfully created a bluetooth connection with the bluetooth module, successfully sent and received messages between the infrared emitter and receiver, tune the current of the infrared emitter correctly so that it met our specifications without causing unnecessary issues, etc). This semester was spent caring out those designs and plans.

5.1 Target Implementation

5.1.1 Hardware Implementation

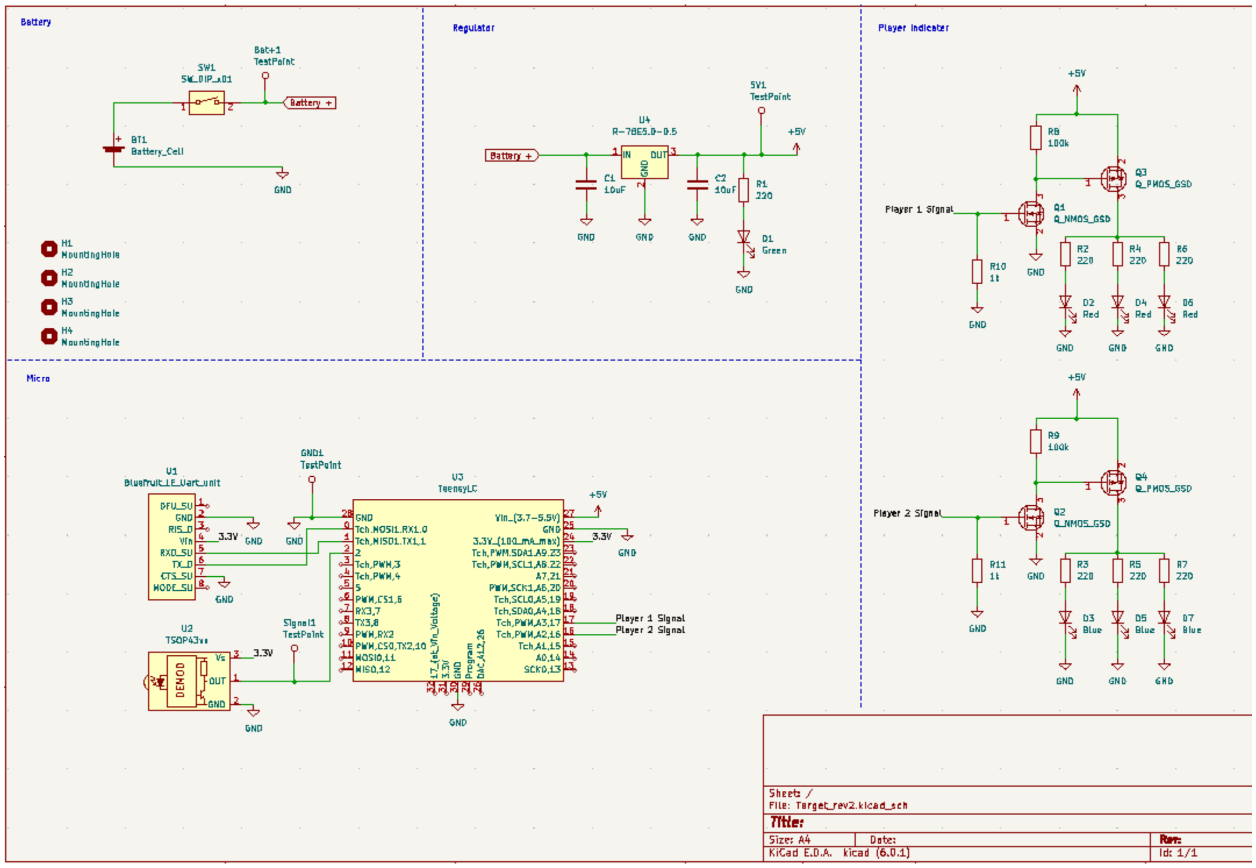
Figure 4: Target Schematic

Using the specifications of the infrared emitter and the correct bluetooth module we created a schematic for the target system of our game. The intention of this circuit is for the player indicator LEDs to light up once the infrared receiver detects either the first or second player's encoded

5.1.1.1 Battery and Power

The target is powered by a 9-Volt battery which leads into the switch to turn the system on and off. The 9V line is then fed into a DC-to-DC converter that then drops the 9V to 5V. This is used to power the microcontroller that we chose for this project (Teensy LC). The bluetooth module is powered by the Teensy's 3.3V output, as well as connected to receiving and transmitting pins on the Teensy. This allows for bluetooth connection between the targets and the Raspberry Pi that is hosting the server. This connection is how the target knows when to start up and enables the game to keep track of which player hit which target.

5.1.1.2 IR Receiver

The infrared receiver is also powered by the Teensy’s 3.3V out and has its signal output pin routed to an analog pin of the Teensy. The receiver that we chose demodulates a message that it receives. Because of this, it only reads signals when they are being frequency modulated by 38 kHz (this also allows the receiver better protection against picking up random noise, reflections and ambient infrared light). The player one and two codes that are being sent by the receiver were chosen to be the rewind and fast-forward buttons on a remote standard remote control. This was a decision based around gaining a better way to troubleshoot both the target and blaster systems, but also serves as the reason that we used the NEC IR transmission protocol when sending messages. This standard uses a 9 ms pulse burst followed by a 4.5 ms space as a 13.5 ms start sequence. After that, a logical '0' is a 562.5 µs pulse burst followed by a 562.5µs space, with a total transmit time of 1.125ms and the logical '1' is a 562.5µs pulse burst followed by a 1.6875ms space, with a total transmit time of 2.25ms.

5.1.1.3 Player Indication

Once a signal has been received and demodulated by the receiver, the Teensy decodes the message, transmits a message to the Raspberry Pi (via bluetooth) instructing which player hit the target, and sets the appropriate player signal pin high. The pins assigned to the first and second player signals (pins 17 and 16 respectively) are meant to light up three parallel LEDs in that respective player's color. Due to the fact that the Teensy is unable to provide enough current to light up the three parallel, we have implemented cascaded common source amplifiers to source the current necessary from the 5V line.

5.1.2 Mechanical Implementation

5.1.2.1 3D Model Design

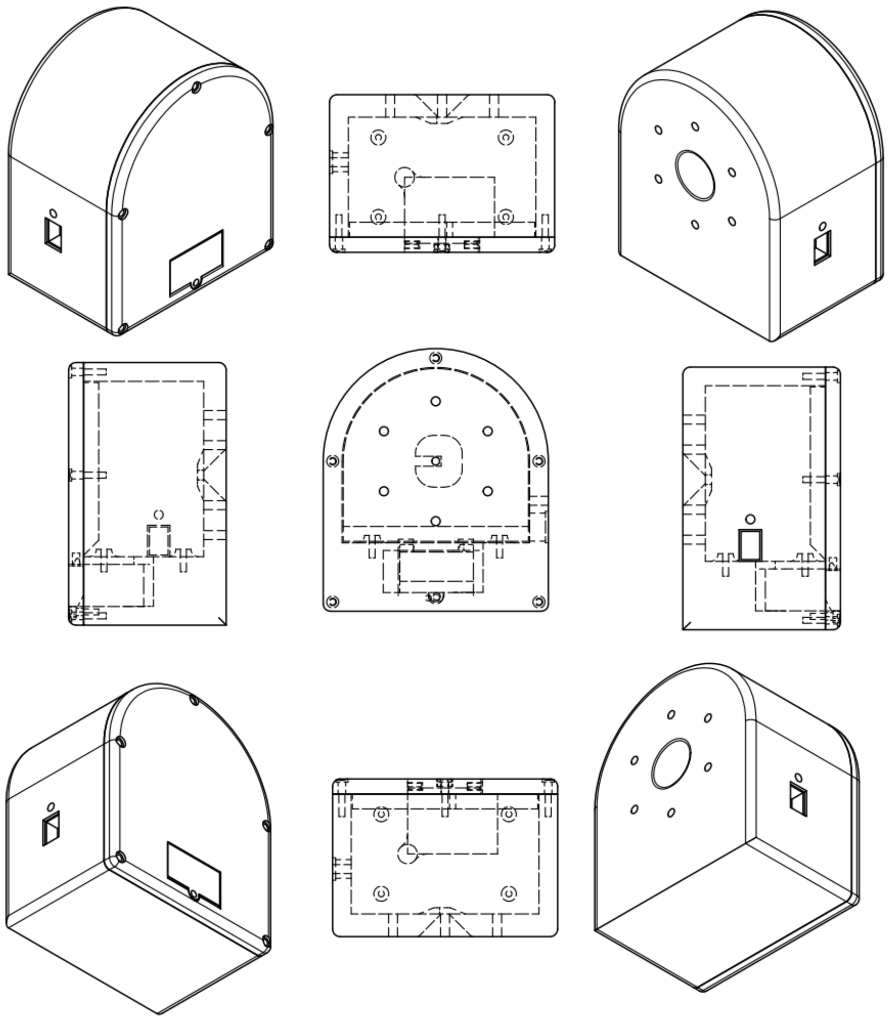

Figure 5: Target 3D Model

For the target design, we wanted something that was sturdy and stood up on its own. That was the reasoning for the flat bottom. The choice to make the top part round was to resemble a circular target with the bullseye in the middle. When you put these two designs together, we got our shape. We attached the PCB on top of the shelf inside of the target.

We added a lit to hide all the guts inside. This lit was screwed on to the front. Since changing the battery happens more often than changing the electronics inside, we created a little battery cover inside of the lid so you can remove one screw and gain access to the battery.

5.1.2.2 PCB Layout Design

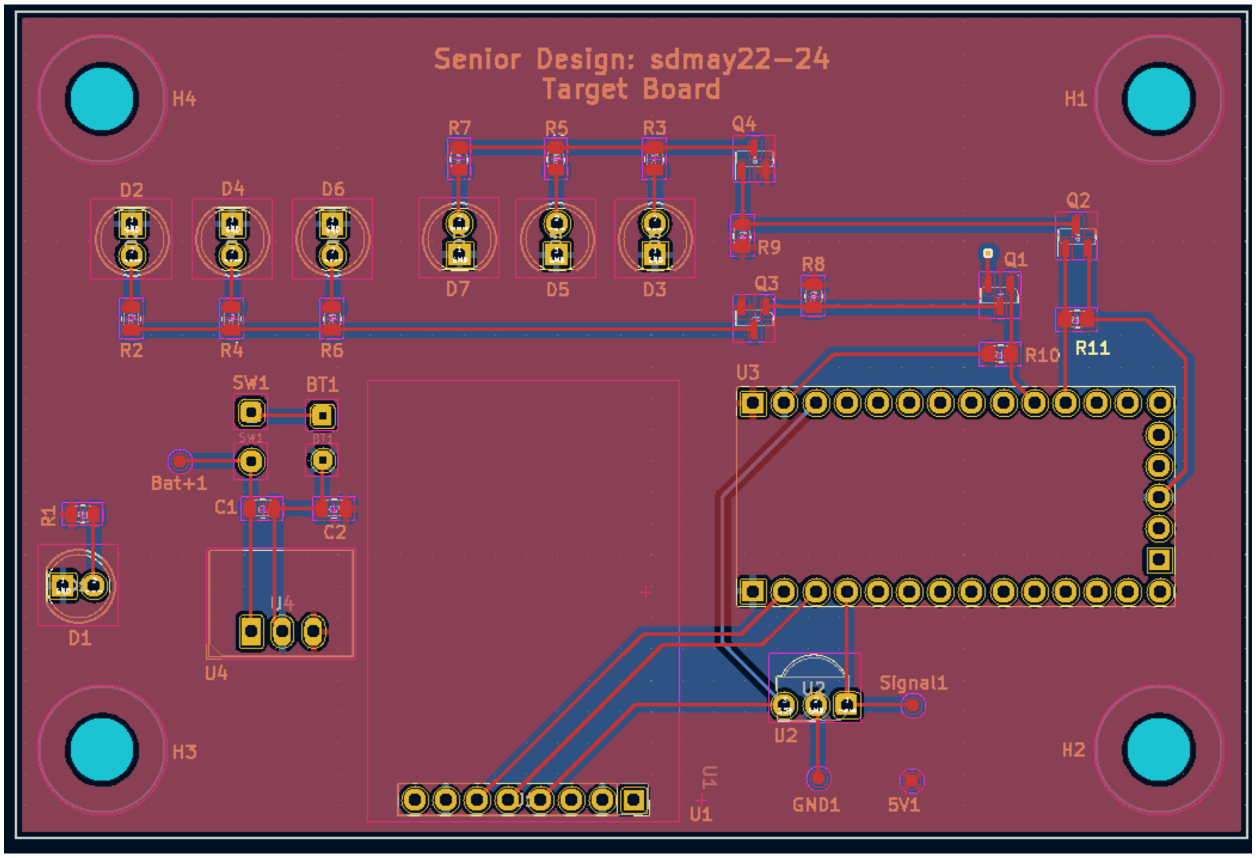

Figure 6: Target PCB

When creating the PCB layout for the target circuit we decided to make it a two layer board while making the top layer of the board 5V plane and the bottom layer the ground plane. This allowed for easier routing for the two networks that had the most connections. Because most of the circuit is DC we were not too concerned with a loss in signal integrity due to separations in the ground plains or islands in the 5V plain. We chose to use 0805 resistors and capacitors because they would be easier for people who were inexperienced with soldering to put on and we had plenty of room due to the other components. For the transistors used in our design we decided that the sot-23 package was what we have all used previously and are most familiar with. To make the manufacturing of the prototype with the 3D printed target shell easier, it was decided that it would be best to use wires to connect the LEDs and switch to the circuit board.

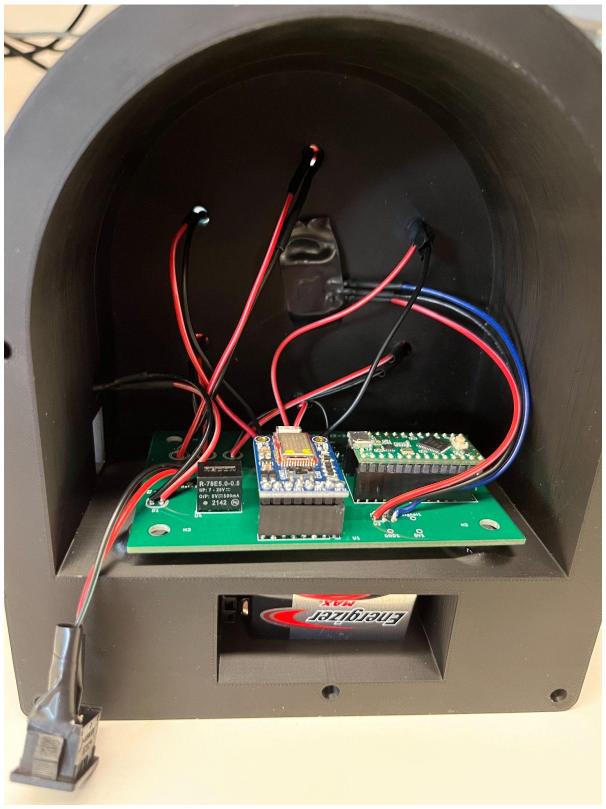

Figure 7: Physical Target Circuitry

5.2 Blaster Implementation

5.2.1 Hardware Implementation

Using the specifications of the IR receiver and Teensy Microcontroller, we were able to build a schematic for the PCB of the blaster which can be see in the following photo below.

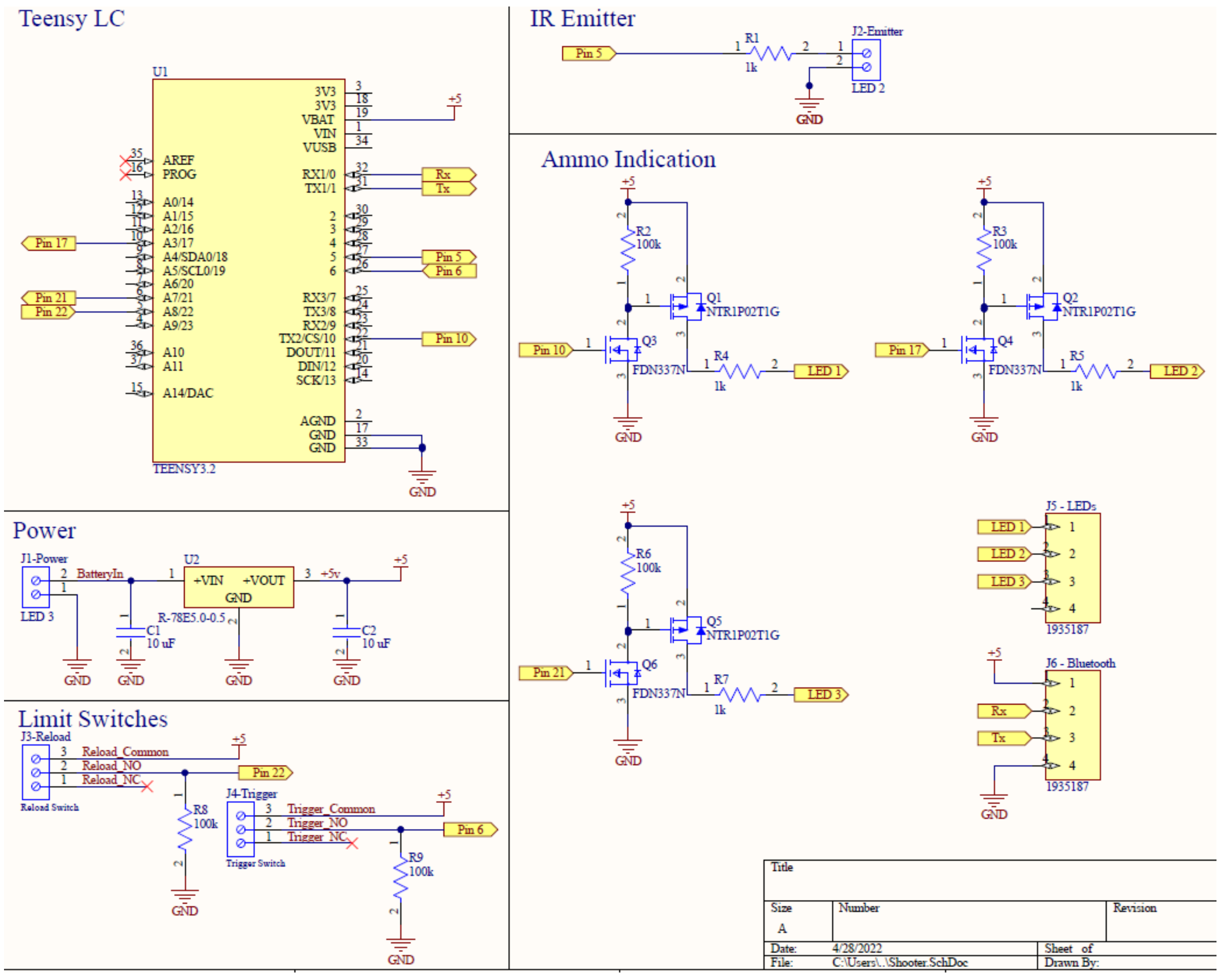

Figure 8: Blaster Schematic

5.2.1.1 Teensy Microcontroller:

The first block most people notice is the Tennsy LC Microcontroller block. This part acts as the brain of the blaster hardware. The code for the blaster is uploaded to this component and based on the various inputs it receives from different components will cause different impacts depending on what it received. These different interactions will be explained in more depth in the following sections of the schematic review.

5.2.1.2 Power:

The first section to look at in the schematic is the power supply. In our design, we use a 9 volt battery to power everything. However as the battery is used up, the voltage will slowly decrease from 9 volts. On top of that, the max voltage the teensy can handle is around 5 volts and could be damaged if above that. In order to prevent this, a 5 volt linear regulator was added to the board.

5.2.1.3 Limit Switches

The blaster uses two limit switches in its design for both the reload and trigger functionality. On pin one of each limit switch is connected to a 5 volt power rail to act as the power source for the limit switch. The normally open pin (Pin 2) of the limit switches are then connected to their respective pins of the microcontroller. This means that when the limit switch is pressed, this line will go high and trigger the microcontroller. Pin 2 also has a pull down resistor on the line to prevent floating and make sure that the line does not accidentally go high when the switch is not being pulled.

5.2.1.4 IR Emitter

The IR emitter in this case is a simple design. It receives a signal from the teensy microcontroller when the trigger limit switch is activated. The teensy will then send a PWM signal from the teensy into the Emitter which will then send a signal towards the IR receiver on the target.

5.2.1.5 Ammo Indication

The ammo indication LEDs are controlled by a series of mosfet buffers. Since most pins of the microscontroller can not source enough current in order to power the LEDs, a mosfet buffer is needed to source the current. This buffer is made up of one n-channel mosfet and one p-channel mosfet. The gate of the n-channel mosfet is connected directly to the output of the microcontroller. The gate of the p-channel mosfet is connected to the drain pin of the n-channel mosfet. When the n-channel mosfet is activated, it also activates the p-channel mosfet. The 5 volt source connected to the source of the p-channel mosfet is connected to the Ammunition LED indicator connected to the drain pin.

5.2.2 Mechanical Implementation

5.2.2.1 3D Model Design

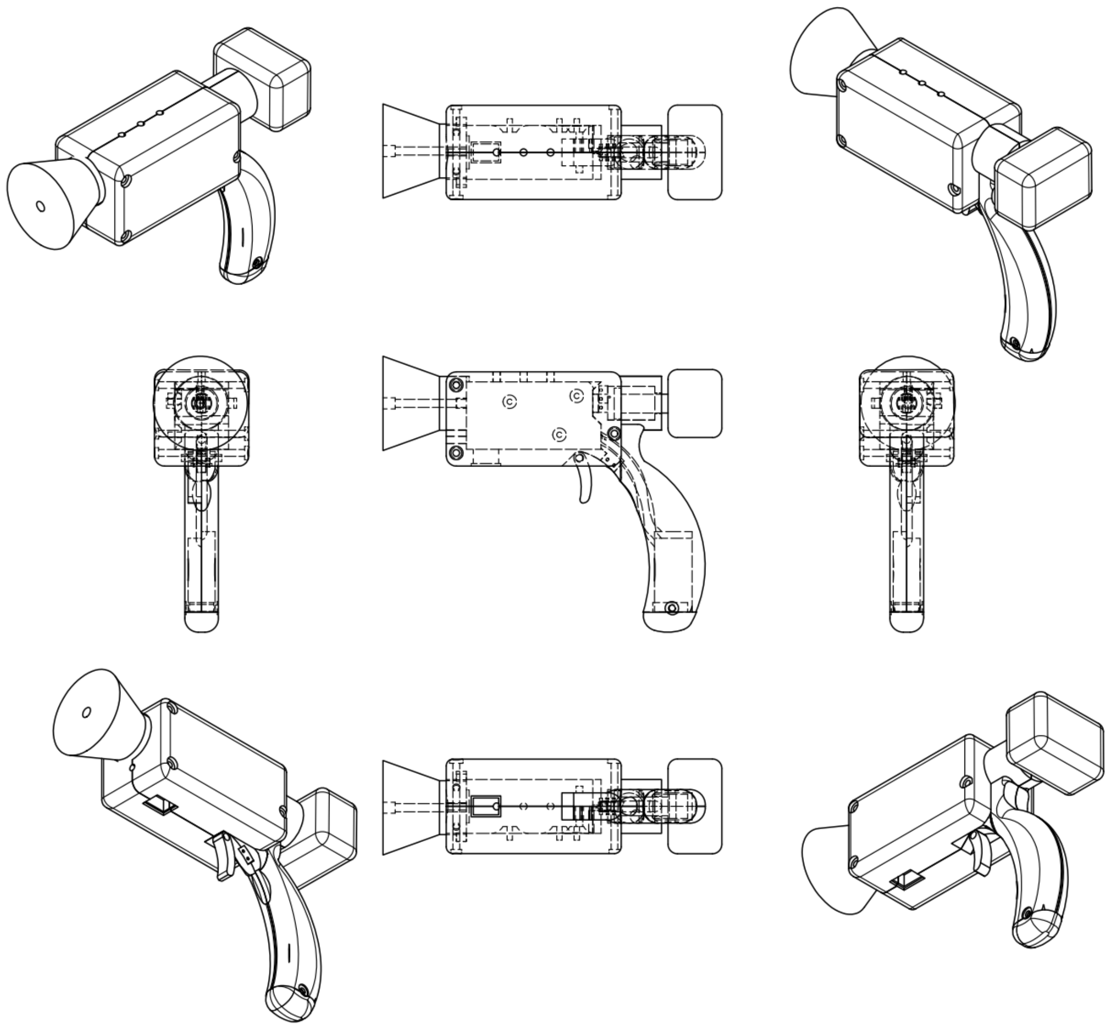
Figure 9: Blaster 3D Model

When it came to designing the blaster, we wanted a house to store the PCB. As you can see in the center drawing, there are 3 holes that are used to mount the blaster’s PCB into place. We designed this shoot easy to hold, and effective to aim. We wanted to do this while also making sure that it was clearly not real.   Next, we needed a place to hold the battery and chose the handle as an effective location.

5.2.2.2 PCB Layout Design
The following photo is the final PCB layout for the project.

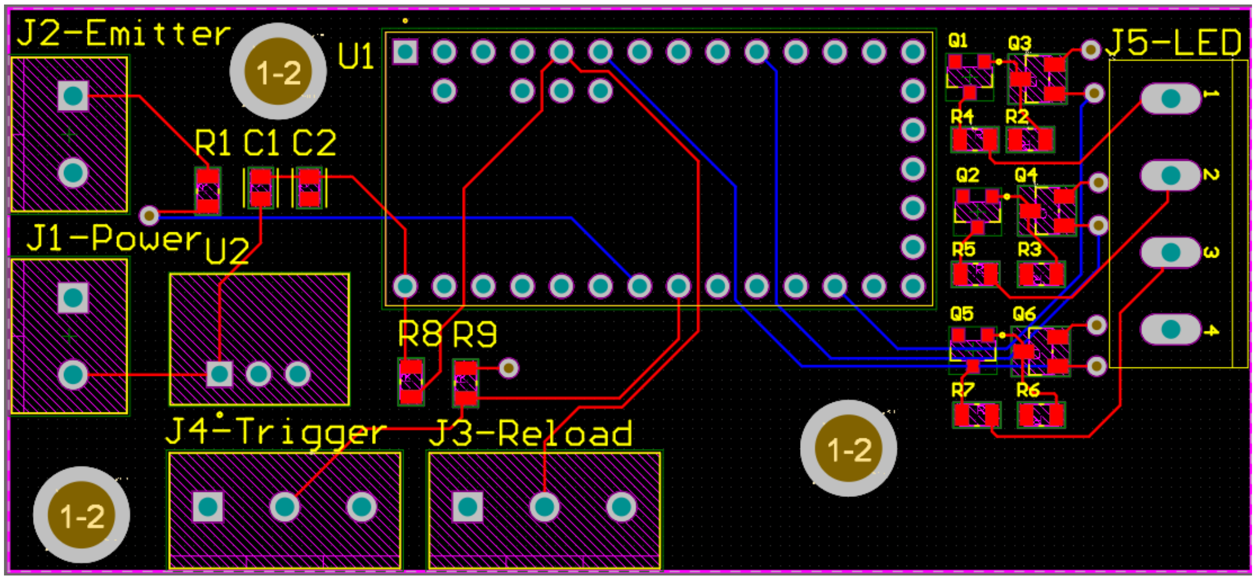

Figure 10: Blaster PCB

The PCB is roughly 81mm by 37 mm in size. The PCB for the blaster sits in the rectangular portion of the blaster 3D model. There are then wires that run from the PCB to the IR LED emitter, limit switches, battery switch, and the ammunition LED indicators.

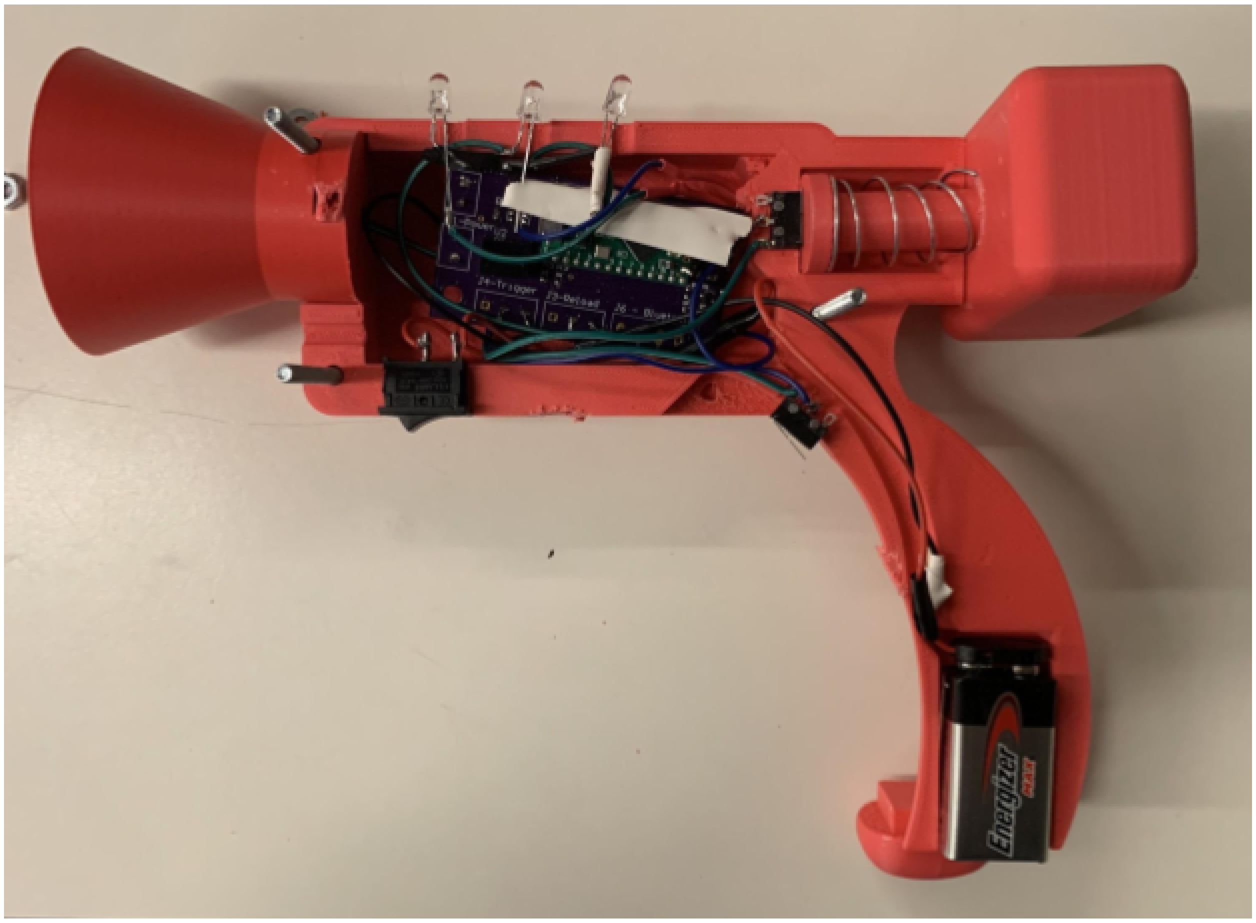

Figure 11: Physical Blaster Circuitry

5.2.3 Blaster Software Implementation

The blaster processing was all done by the Teensy LC, and to have it operate as intended, a program for the blaster would need to be designed. The program ended up being a somewhat simple state-based machine, with states being based on how much ammo the blaster has. Full ammo, 2 shots left, 1 shot left, and no ammo are the states. During all states except the no ammo states, the blaster would shoot the IR code, and then go down one stage. When the trigger is pulled while in the no ammo state, no code would be sent, and the state would not move. Whenever the reload button was pushed, the state would be returned to the full ammo state.

5.3 Software Implementation:

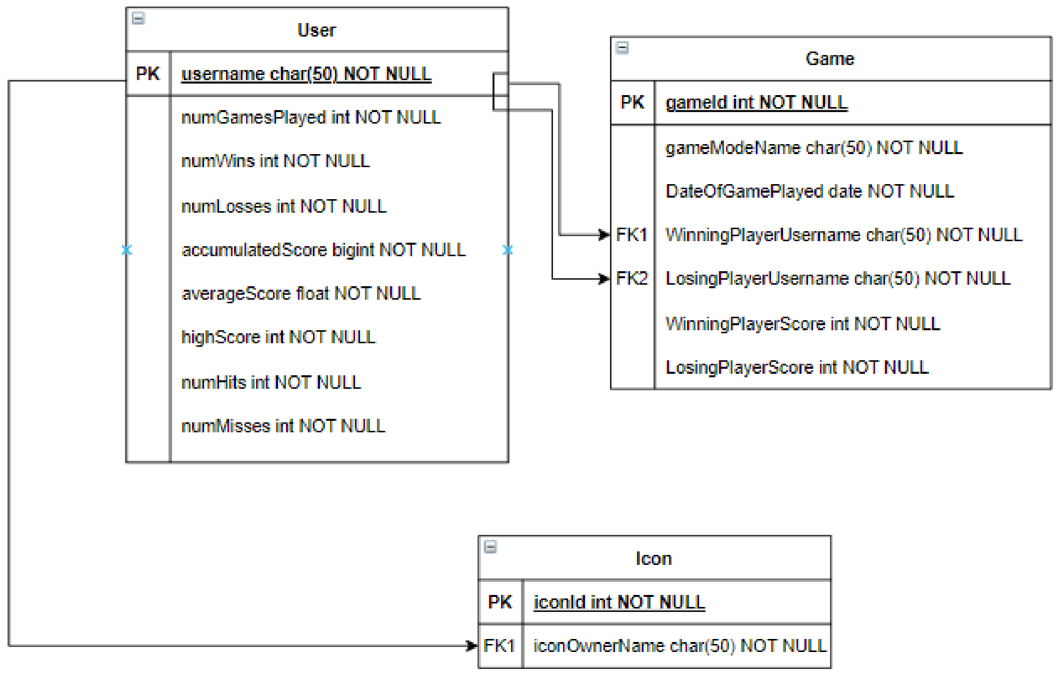
User
PK
username char(50) NOT NULL
numGamesPlayed int NOT NULL
numWins int NOT NULL
numLosses int NOT NULL
accumulatedScore bigint NOT NULL
averageScore float NOT NULL
highScore int NOT NULL
numHits int NOT NULL
numMisses int NOT NULL
Game
PK
gameId int NOT NULL
gameModeName char(50) NOT NULL
DateOfGamePlayed date NOT NULL
FK1
WinningPlayerUsername char(50) NOT NULL
FK2
LosingPlayerUsername char(50) NOT NULL
WinningPlayerScore int NOT NULL
LosingPlayerScore int NOT NULL
Icon
PK
iconId int NOT NULL
FK1
iconOwnerName char(50) NOT NULL

Figure 12: Database Schema

Starting with the backend of the application, we created a database schema then setup and initialized a mySQL database off of that schema. Once that was completed, we implemented a Springboot application server that would connect to the database we just setup. Within the Springboot application, we created different HTTP endpoints for the different screens implemented on the frontend of the application. For the frontend application we began by creating mock ups of the different screens to be able to set up all of the UI from our previous designs. Once we had the UI complete we implemented the HTTP endpoints to add in the screens functionality, and then worked to fix sizing issues and framing of the application into the tablet. Also in the final end product, we didn't need all of the screens we originally designed for on the frontend, so we did not implement those. For the communication between the raspberry pi and the application we originally decided on using websockets but after we integrated the sockets, we couldn't successfully connect and transfer data through websockets so we ended up using more HTTP endpoints to cover the necessary communication.

6 Testing

6.1 Integration Testing

The critical integration paths that needed to be tested were the connections between the targets/blasters to the raspberry pi server, the raspberry pi to the targets/blaster, and the tablet to the raspberry pi. The testing to ensure connectivity was done by checking that the bluetooth/wifi connections are made, and potentially adding startup sequences that indicate connectivity between the devices.

6.1.1 Target Testing

Once the target was fully soldered we tested its functionality in a number of different tests. The first was confirming that the 5V line was properly connected and that the DC-to-DC converter was fully functional in stepping down the 9V input. This was to make certain that the microcontroller would not be damaged when the system was turned on.

The next test we performed was uploading code for decoding the received signal and turning on the respective player's indicator LEDs. To make matters simply we used the infrared remote mentioned earlier to simulate the blaster emitting a decoded message for either player. Doing this not only made the testing of the target easier through the compact size of the remote and wider beam width of its infrared emitter, but it helped in the blaster testing by confirming that the targets were functional and that if the blaster failed to activate the target then it was an issue with the blaster.

The final test for the target was to confirm that the bluetooth was fully functional. This was done by uploading the finished code that included the bluetooth messages that the module should expect to receive and transmit. We then connected the bluetooth module on the target to a phone app to confirm the ability with devices like the Raspberry Pi being used for the central server. We sent a start up command that would be used to begin the game. After this was done we used the remote to simulate player signals in order to test that the target system was transmitting the correct strings that would tell the server which player hit which target.

6.1.2 Blaster Testing

Once the target was fully soldered, it was ready to test the functionality of the different components for both the hardware and the software. First up the hardware was tested to make sure that everything functioned properly and after that the software testing began to make sure that the hardware would not damage the microcontroller.

First up was testing the DC-DC converter stepping the 9 volts down to 5 volts. This was tested at a range of voltages between about 9.5 volts to 5.5 volts to make sure it worked at all values. This section worked perfectly fine which meant it was safe to connect the microcontroller without damaging it.

Now that the microcontroller was attached we started testing the IR LED. First off we needed to make sure that the teensy was reading the signal from the limit switch when it was pressed. Once we saw it was receiving the signal, we then measured the IR LED emitter pin and made sure that the correct signal was being sent.

Next up we needed to test the LEDs on the top of the blaster. First off we wanted to make sure that every one of the LEDs lit up when the code was reset. We measured each of the pins of the mosfet buffer to make sure that they were implemented properly. We would then pull the trigger limit switch to make sure that the mosfet buffers would go low one at a time and in each order. On top of that, we also needed to test the reload code. Whenever the reload limit switch was hit, each of the mosfet buffers needed to go back to reading 5 volts. Once we saw this, we knew that the entirety of the code was functional with the hardware and was ready to be permanently connected to the enclosure of the blaster.

6.1.3 Software Testing

Integration testing between the frontend and backend was initially handled manually through a locally running server and a locally emulated tablet. This changed mid semester to running the application on the physical tablet we have and the server on the Pi. There was also testing done between the bluetooth listening service and the server which ensured that communication was working and that the Pi could handle running the server and service at the same time which each had multiple threads to manage. The testing around the GUI brought up spacing issues with the emulator compared to the physical tablet.

6.2 System Testing

One important system test is checking that the Pi receives the signal once you transmit the IR signal from the transmitter. This involved setting up the initial bluetooth connection between the Pi and the bluetooth module, sending a player signal to the target in a place that the receiver could pick it up, and confirming that the bluetooth module and microcontroller transmit the correct message back to the Pi to recognize the hit and assign the player a point.

Another vital test that we performed was to confirm that the blaster and the target communicated properly. This was done by using the individually tested blasters and targets and having the blaster send its player signal in a place that the receiver can read the message. If successful the indicator LEDs of the respective blasters will light up. During our testing, we were able to calibrate the IR Emitter and receiver to work on a 15 foot range. At this range, there is a beam width of 6.3 degrees. The following photo shows the range that all of the systems work together at.

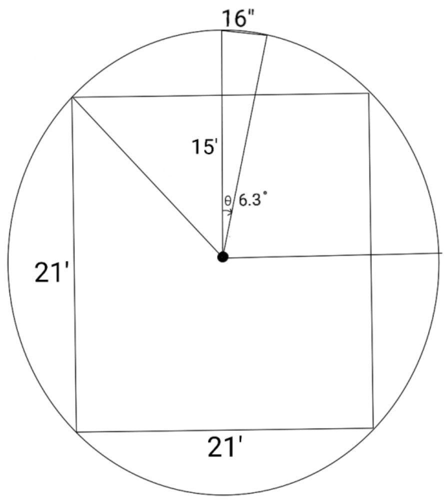

Figure 13: Blaster and Target Range Drawing

With the bluetooth communication and target-blaster connectivity confirmed we were able to test the game mode of our final system. For this we opened up the game's app and started the time. When shooting the target successfully we expected the correct player's indication lights to turn on and for the Pi to receive a bluetooth message which it would then use to award that player a point.

7 Conclusion

Currently, our team has created a blaster that can send an encoded signal that is picked up by the target. The target will only pick up that the blaster fired if IR LED was accurately pointed at the target. The target then registers who hit it and relay that information to the Pi. The Pi, using this information, sends this data to the server which updates the score then the application requests relevant game information to display.

The biggest problem the hardware team encountered was the part shortage. Parts were in short supply in online stores, and many of the ones left either had a long wait list or were very expensive. We ordered the best parts at our disposal given the limited choices for our project. In the future, our best option was to decide on parts sooner, as one of the reasons we had less options was because we waited a bit longer than we should have to order the components we needed for building. The other problem was the time it takes to 3D print something. Since all of our items were big, it takes way longer to print than anticipated because sometimes a print will get messed up and need to be restarted.

8 Appendices

8.1 Operation Manual

Supplies

- 2 - 9 volt batteries
- Philips screwdriver
- 2 - Blaster Shells
- 4 - Target Shells
- Raspberry Pi
- Mobile interface (Tablet)

Target

1. Step 1 -> replace the battery of the target
    a. Unscrew the screw holding the battery cover.
        i. Indicated by figure 14
    b. Take off the cover
    c. Pull out the 9 volt battery
    d. Detach the 9 volt battery
    e. Connect a new 9 volt battery to the connector
    f. Put the battery back in it’s housing
    g. Replace the battery cover

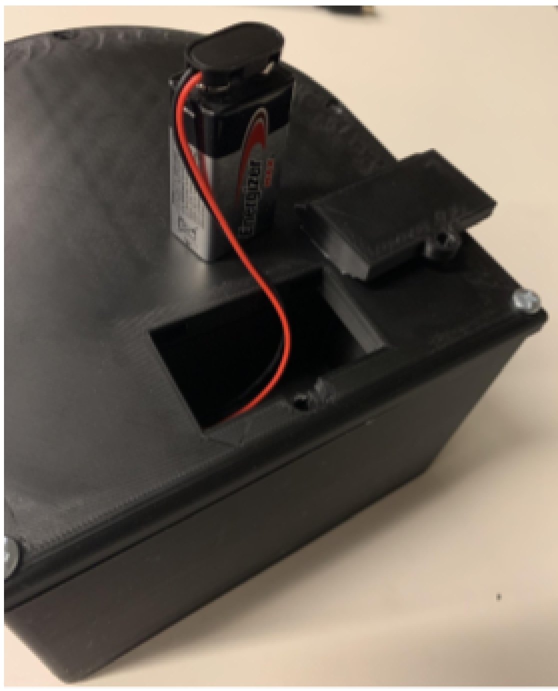
Figure 14: Battery for Target

2. Step 2 -> Turn on the Target
    a. Flip the power switch from the O to the I position
        i. Found on the right side of the target
        ii. Look at figure 15
    b. Check that the target is on
        i. The green LED above the target is on
        ii. If LED does not turn on, replace the battery by following step 5

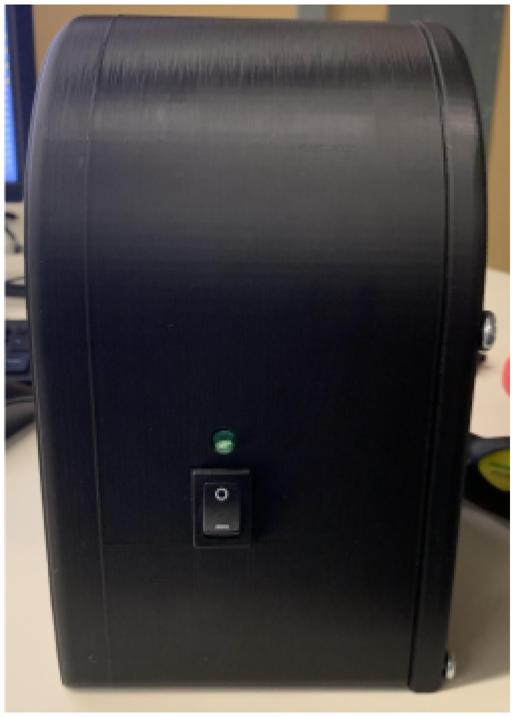

Figure 15: Target Power

Blaster

3. Step 3 -> Replace Blaster battery.
    a. Start by undoing the screw at the bottom of the blaster on figure 16
    b. Remove the battery cover
    c. Pull out the old 9 Volt battery
    d. disconnect the old battery
    e. Replace with a new battery
    f. Push battery back into its slot
    g. Replace the battery cover
    h. Redo the screw to hold the battery back into place

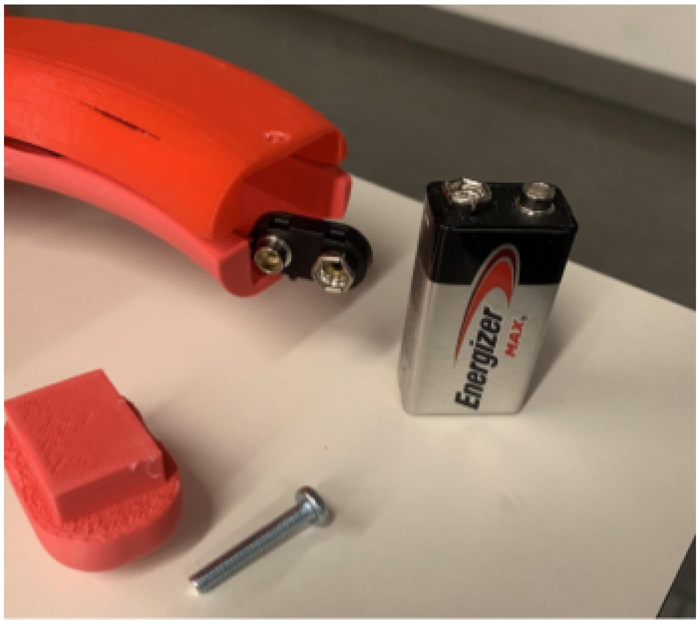

Figure 16: Blaster Battery

4. Step 4 -> Turn on the blaster
    a. Flip the power switch from the O to the I position
        i. Found on the bottom of the blaster under the barrel
        ii. This is the switch in figure 17
    b. Check that the target is on
        i. Green LED in the front by the switch is lit
        ii. 3 blue LEDs on the top of the blaster are lit
        iii. If it did not turn on, replace the battery using step 1

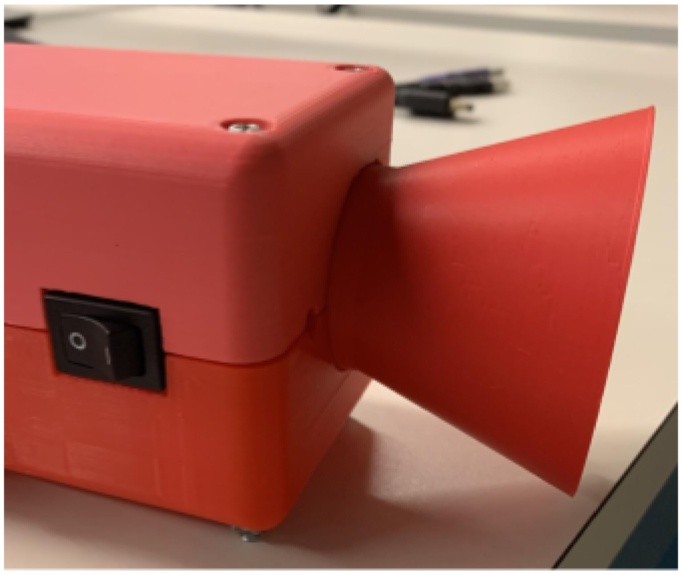
Figure 17: Blaster Power

5. Step 5 -> Shooting the blaster
    a. Aim at the target that you want to hit
    b. Pull the trigger
    c. See figure 18
    d. The LED at the top of the blaster indicates the ammo left.
        i. You start out with 3 shoots
        ii. If fired while out of ammo, the 3 LED will blink
            1. Go to step 4 to reload inorder to fire some more
6. Step 6 -> Reload the blaster
    a. Pull back the back
        i. See figure 19
        ii. All 3 of the LED will light back up if properly reloaded
        iii. It will not reload until the back goes back to its starting position

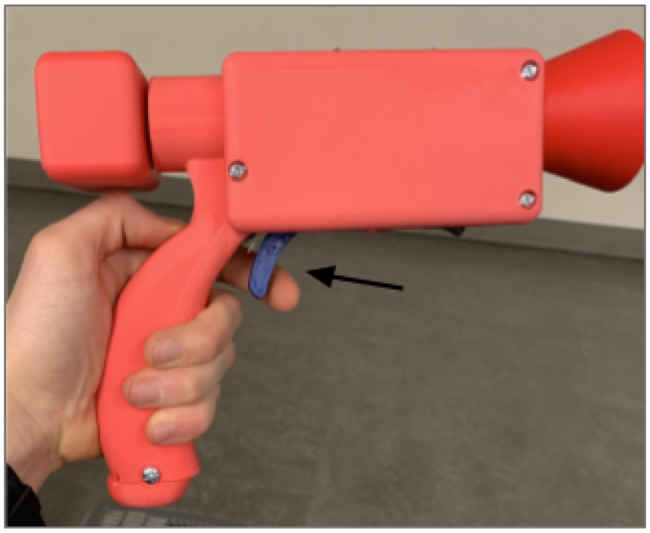
Figure 18: Firing

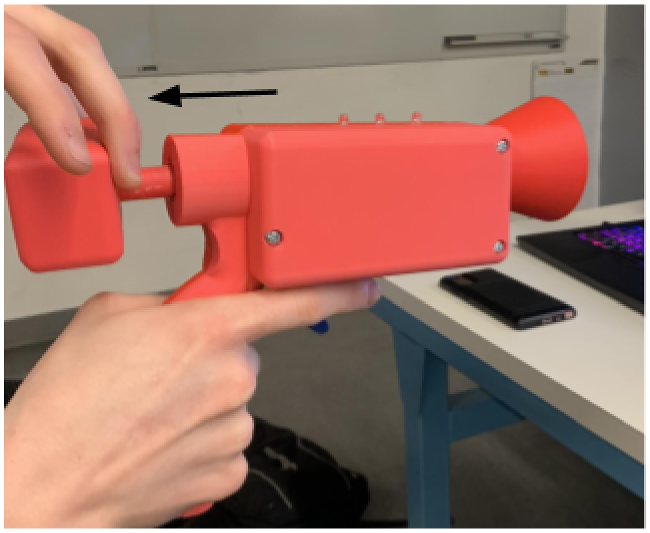
Figure 19: Reloading

Raspberry Pi

1. Connect to “RaspRouter” wifi network on computer
2. Open two terminals
3. SSH into the Raspberry Pi twice (ssh pi@192.168.4.1) (password: raspberry1)
    a. cd to the Pi’s desktop
    b. Run server on one (java -jar server.jar)
    c. Run the bluetooth service on the other (after targets are on) (python bluetooth_listener_service.py)

Android Application

1. Open up the application on the tablet
2. Create new profiles if necessary
    a. Select new profile from the home screen
    b. Type in a name and select a picture and submit
3. Play game
    a. Select player 1 from the list
    b. Tap the player 2 icon at the bottom right
    c. Select player 2 from the list
    d. Press the start game button
    e. Use the blasters to shoot at the targets to accumulate points within the time limit.

8.2 Initial Versions

We initially designed all of our 3D shells with only light consideration for the size of our 3D printer. This resulted in the blaster being too big for the Ender 3 that we used. In order to fix this, we made the barrel its own component. This resulted in a problem with securing it in a straight position. The solution to secure it was to add pegs on either side of the barrel to hold it from wiggling up and down.

On the software side, we originally had the websockets as the form of communication between the application and the raspberry pi for in-game communication to update the score. After implementing this, we tested the communication and could never find a stable connection and successfully transmit data back and forth between the two. With this, we decided to switch to using more HTTP endpoints to achieve this necessary communication. This way ended up working much more effectively and steadily compared to the websocket implementation.

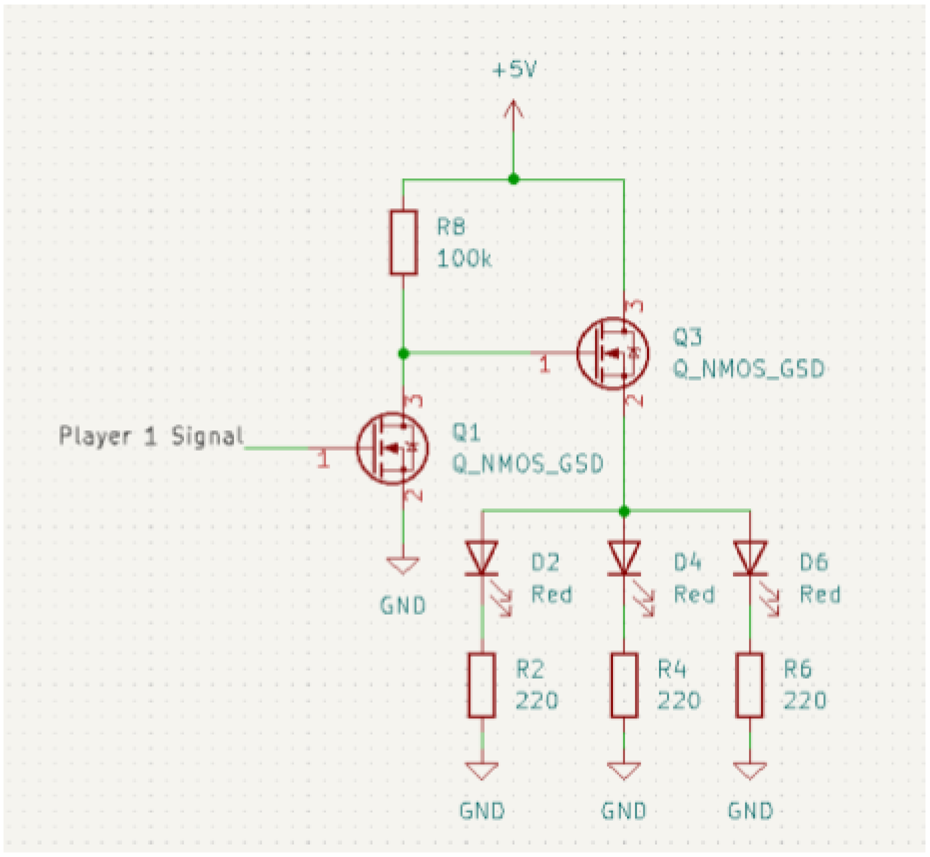

Figure 20: Revision 1 MOSFET Buffer

Another failure in our project occured in the first revision of the target's schematic and PCB. There were three main issues: two from the schematic and the other on the PCB layout. The first problem was that in the cascaded amplifier circuit we accidentally made the two transistors both n-channel mosfets. We wanted the second one to be a p-channel mosfet, so because of this mistake the buffer circuit drew less current than we were expecting and made it use the wrong mosfet logic. Neither of these issues were too bad to be fixed. We reduced the current limiting resistor so that more current was able to be drawn and made the light up code active low instead of the active high that we planned on. When we wanted the lights to be off the microcontroller pin needed to be high and when we wanted them to be on the microcontroller would pull the pin low.

The next problem on the first revision was that there were no pull down resistors. We should have added pull down resistors at the gate of the n-channel mosfet that was connected to the microcontroller. We initially believed that having the microcontroller pull low would create a digital ground and that would be good enough. This did not end up being the case so we added the pull down resistors in the second revision.

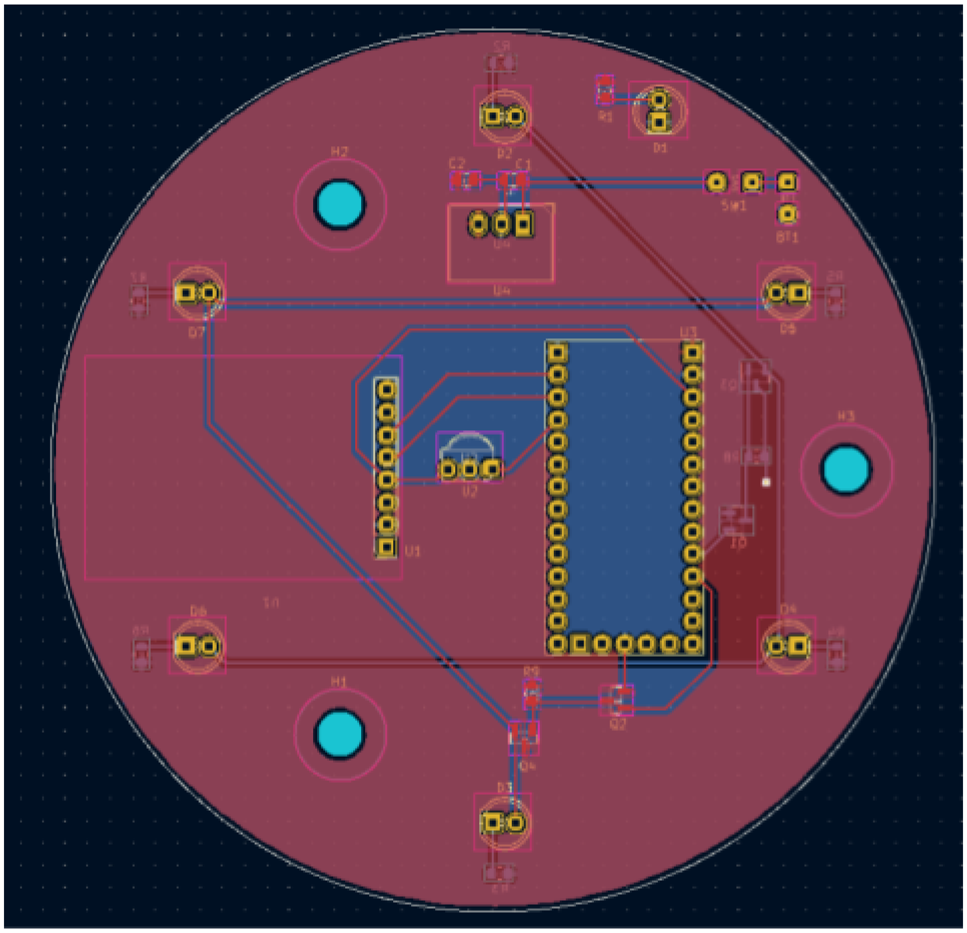

Figure 21: Revision 1 PCB Layout

The third issue was that the footprint for the bluetooth module was accidentally flipped so the pins were not connected to correctly. All of the problems from the first revision were addressed in the second revision along with adding test points and changing the shape of the PCB. The PCB was shaped as a circle so that the LEDs could line up against the holes for the target and allow the mounting of the PCB to be behind the face of the 3D shell.

The fourth issue we had was in the fabrication process of the blaster PCB. When the original file was made, the board outline was placed on the wrong layer. This caused about 13mm on the right side of the board to not be manufactured. Because of this, the LED connector and the ground plane were cut off. Some components were not properly functioning due to this so some modifications on the initial board revision needed to be made. While a new board was ordered, it did not arrive in time for this document. It wil however be arriving this weekend so it can be prepared for the presentation day.

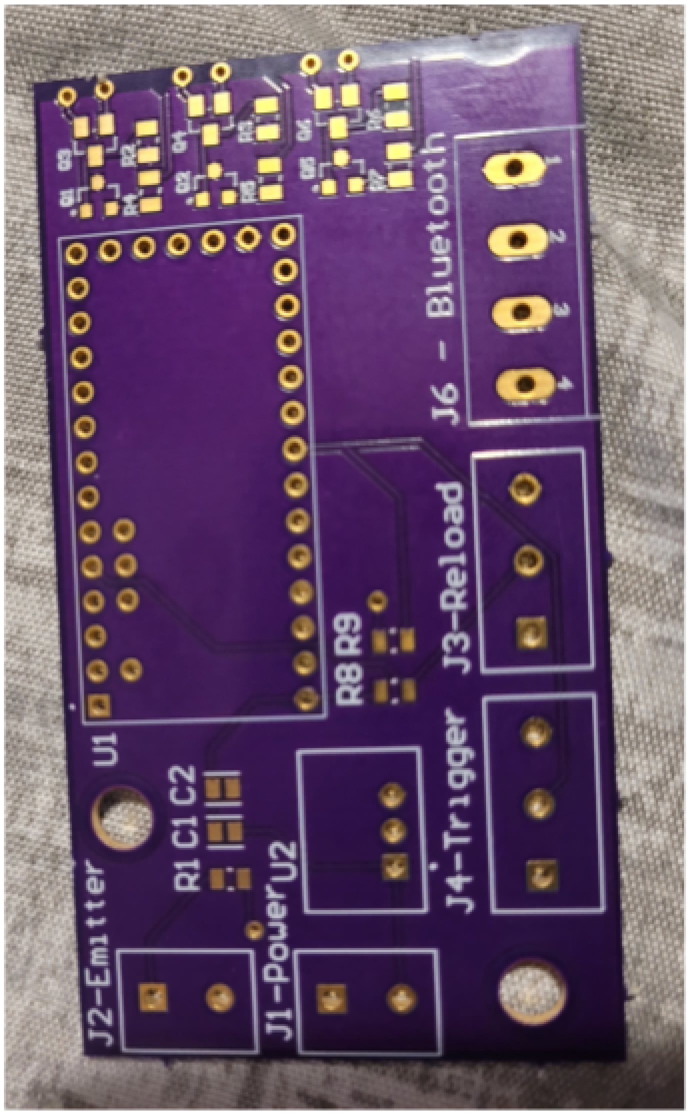

Figure 22: Messed up board

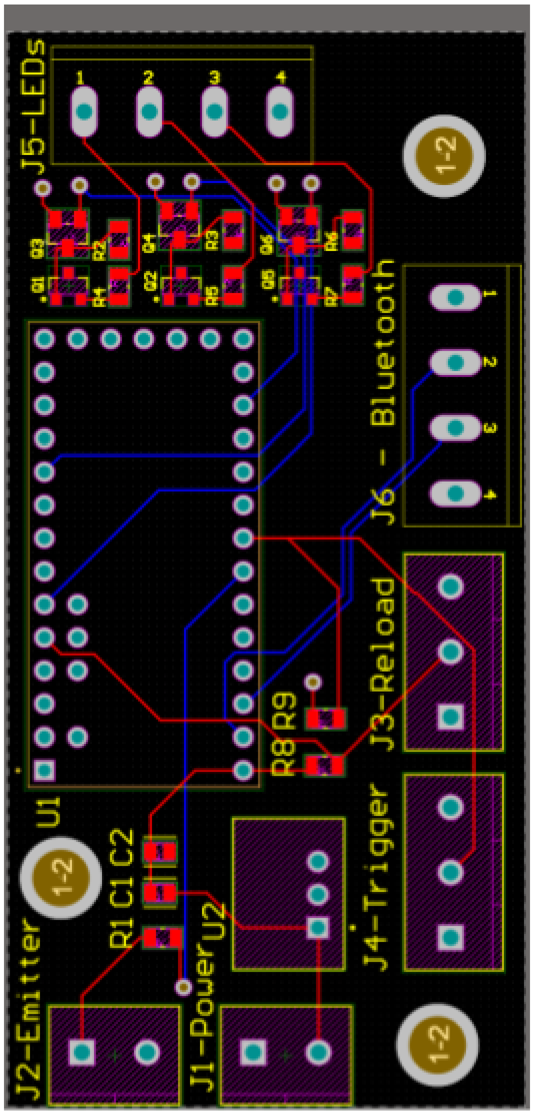

Figure 23: Correct Layout

8.3 Other Considerations

During this project, we learned about the process of how to design, test, and print out 3D components. Some of the skills learned here was the tolerance of 3D prints, buffer space between parts, and how to fasten parts together. We also learned about conserving filament while testing. Some of the techniques that we learned was just how thin of walls, floors and low the infil could be while still printing out a component.