Laser Arcade

Prepared by: Alex Wheelock, Andrew Keller

Robotics and Communications Systems Engineering Technology, Idaho State University

Prepared for: Shane Slack, Tim Rossiter

Spring 2025

Table of Contents

Table of Figures

1: Abstract

The Laser Arcade is an interactive, head-to-head game which will be used by the Idaho State University Robotics program for recruiting purposes. There are two main components of this project, the blaster and the target system. It features two blasters which allow two players to compete against one another by engaging targets for points. Each blaster has a frequency of either 38kHz or 56kHz, which corresponds to IR sensors built into each target. When the blaster is fired, the laser is modulated at one of the two receiver frequencies in order for the receivers to detect a hit on target, and determine which player hit the target. Targets will randomly enable and light up to indicate to the players which targets are active, then it will be a race to see which player can hit the target first. Players that hit an active target are awarded points, and the player with the most points at the end of the time limit wins.

2: Background

The Laser Arcade was initially started in the Fall 2024 semester by Alex Wheelock. In that semester the entirety of the project and how the game would function was planned out. However, primarily the only thing achieved in that semester was the blaster circuit and code being ironed out, leaving a lot to do in the Spring 2025 semester.

3: Proposed Problem

3.1: Overview

Previously the ISU Robotics program had a project similar to the Laser Arcade used for recruiting purposes, which was Nerf based rather than laser based (Shown in Figure 3.1.1 below). The Nerf Arcade was two motor-driven turrets which were controlled by game controllers, to shoot at targets. It was successful in attracting high school students to the ISU Robotics table at Tech Expo for recruiting. There were downsides to this project however. Primarily, the need to collect the Nerf balls once fired, and reload the guns leading to downtime of the game. A solution to this was to build a robot that can pick them up as people are playing to reduce the downtime. This robot unfortunately had its own issues however. Finally, the motors controlling the turrets died and rather than replacing them the project was scrapped. As a result, the Laser Arcade is being developed to replace the Nerf Arcade, eliminating the need for cleanup and downtime, and creating a game that is more fun and effective for recruiting.

Credit: Shane Slack

Figure 3.1.1: Nerf Arcade Project

3.2: Semester Goals

- Complete the target system.
    - Finish programming the target master and slave PICs, so that they function as intended and can communicate via I2C.
    - Write a Visual Basic program to control the target system, by communicating with the target master via UART.
- Find or create a target for players to shoot at, that allows both IR sensors to detect the laser when hit.
- Finalize schematics for the blaster, target master, and target slave.
    - Figure out power and audio for the blaster.
- Modify a Nerf gun to house electronics and function as a laser blaster.
- Design PCBs for the blaster, target master, and target slave circuits.

4: Project Technology Impact

Ethical implications must be considered when creating any new product, the Laser Arcade project is no different. While the Laser Arcade will be safe with low power lasers, there are still concerns of potential waste and general safety of the project. The primary concerns for the Laser Arcade would include environmental waste and gun handling safety.

The Laser Arcade project consists of several things that are wasted, or may have a negative impact on the environment if disposed of incorrectly. Electronic waste (e-waste) is a growing concern in the world, with the annual waste estimated to be, “more than 6 kg per person, totaling 44.7 million metric tons in 2016” (Awasthi, et al., 2019). The Laser Arcade PCBs and components, as well as the modified chassis’ should last a long time and are not disposable, the main concern comes from the batteries. Batteries, if disposed of improperly can leak harmful chemicals or even cause fires. Using rechargeable 3.7V batteries in the blasters will reduce the need to dispose of batteries. However, taking the proper steps to recycle these batteries when needing replaced is necessary to reduce the amount of e-waste of the project.

The safety concern of the Laser Game project comes not from the project itself but rather the potential gun safety implications of toy guns. High school students who would handle these blasters at recruiting events in a fun way may cause them to believe that all guns are toys, and handle real guns as such. A study conducted in the 1990s found that half of the kids were unable to tell the difference between a real gun and a toy one (Doh, et al., 2021, p. 6). Meaning that the teenagers handling these blasters may not have adequate firearm handling education. However, the Laser Arcade could be used somewhat as a training aid to teach teens about proper gun handling. Green Line Arms states that games offer dynamic ways to grasp important concepts, making training engaging and effective, making it easier to remember the details and making

safety second nature (2024). So the Laser Arcade may be used to actually help educate teens on gun handling safety, as long as players are supervised and instructed on how to handle the blasters.

Although there are potential negative impacts on both the environment and safety of gun handling, both concerns can be minimized. Throughout the life of the project the primary potential harmful waste, the batteries, can be reduced by using rechargeable batteries for the blasters, and ensuring the proper handling and recycling of them in the future. Additionally, the concern of gun handling safety of high school students who play the game can be reduced as long as proper, safe handling is required by supervisors of the game, using it somewhat as a safety teaching aid. As long as these steps are taken in the future, there are little to no ethical concerns within the Laser Arcade project.

5: System Overview

The Laser Arcade is broken up into 2 major components: the blaster, and the target system. The target system can then be broken down into 3 minor components: the target slave, the target master, and the Visual Basic program which controls the target system. The Laser Arcade block diagram showing how each component works together can be seen below in Figure 5.1.

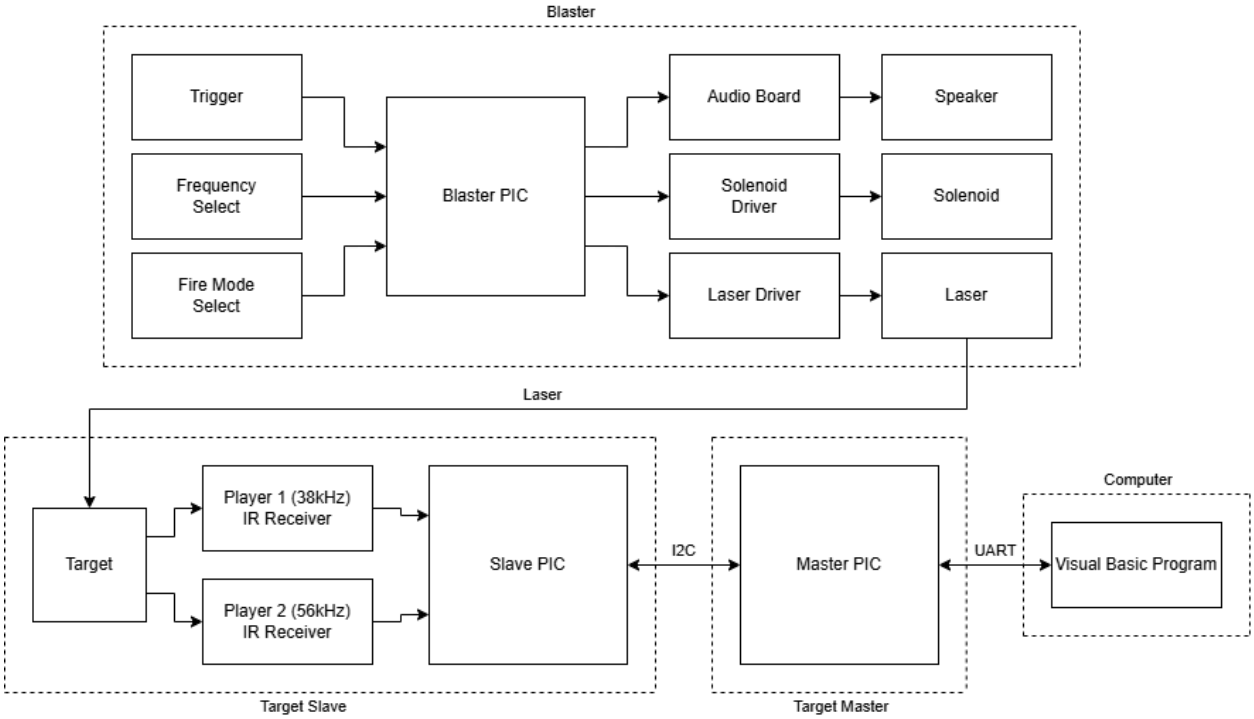


Figure 5.1: Laser Arcade Block Diagram

5.1: Blaster Overview

The blaster, shown in the top section of Figure 5.1, is the gun that is controlled by the players to engage active targets, and hit them for points. It is powered by a 12V DC power supply, which is then regulated down to 5V (6.2.6: Blaster Power) to power a PIC16F1788 which controls the circuit, and the laser.

There are three TTL inputs going into the PIC: the trigger, the frequency select (6.4.2: PWM Frequency Select), and the fire-mode select (6.4.3: Fire Mode Select). The trigger (on RB0) is a momentary switch that when pressed, will fire the blaster. The frequency select (on RB1) is a SPST switch that toggles the laser PWM frequency between 38kHz and 56kHz, allowing any blaster to be used as either player 1 or player 2 and changed quickly, in case a blaster goes down at an event. The last input is the fire mode select, which uses three input pins (RB4:2) to determine the fire-mode set by a rotary switch. This allows the user to choose between semi-auto, three-round burst, and full-auto firing modes.

The blaster also has three TTL outputs: the audio (6.4.5: Audio Output), solenoid (6.4.4: Solenoid Output), and laser outputs (6.4.1: Laser Output). The audio outputs (RA7:0, RC1:0, & RC7:6) connect to an Adafruit audio FX sound board which drives a speaker to play a blaster sound each time the blaster is fired. Each time the trigger is pressed, the PIC triggers on a logic low either TRIG0 or TRIG1 on the sound board depending what mode it is in, the other triggers are currently unused, but allow for additional audio files to be added and used in the future. The solenoid output (RC3) controls an N-channel MOSFET which switches on to energize a solenoid in tandem with the laser output to push the slide back, to simulate recoil felt with a real gun. The laser output (RC2) uses the CCP1 PWM module of the PIC to control a laser diode. It also uses a MOSFET driver to switch the laser on/off with the PWM frequency, either 38kHz or 56kHz as determined by the frequency select switch. The PWM is always running, but the output driver is disabled by default and only turned on when firing, pulsing the laser at one of the two IR sensor frequencies. The blaster schematics can be found in Section 11.1: Blaster.

5.2: Target System Overview

The Laser Arcade target system has three parts: the master and the slave which communicate via I2C, and a Visual Basic program which communicates with the master via UART. (Shown in the lower blocks in Figure 5.1). The system is powered by a 5V 12A DC power supply that plugs into the master board, which then distributes the 5V, along with GND, SCL, and SDA lines to the slave boards. The schematics for the target system can be found in sections 11.2: Target Master, and 11.3: Target Slave.

The slave boards are the targets themselves, with both IR sensors mounted onto the back of the board. Each slave has its own PIC16F1788 set to I2C slave mode for communicating with the master. The I2C address for each slave can be configured by adjusting the DIP switch (SW1) connected to PORTA, which is read on startup to set the address. The slave starts with the target in a disabled state until enabled by the master. The sensors for player 1 and 2 are connected to RB0 and RB1 respectively. When a sensor detects pulses at its corresponding frequency it pulls its output line low, triggering a change interrupt on the PIC and allowing it to quickly identify which player hit the target. The PIC will disregard any hits unless it has been told to enable by the master. Once the target has been hit, the PIC will then flash red (Player 1) or blue (Player 2), giving immediate feedback indicating which player shot it. It then stores a byte that tells which player hit the target, to send to the master on the next read. The slave has 4 RGB LEDs (D2-D5) connected to RB3:0. Green indicates to the players that the target is active, and the red and blue flash three times when the target is hit, to indicate which player hit the target.

The master board is essentially a UART to I2C converter to allow the VB program to control the targets. The master is a PIC16F1788, with a UART interface IC and a micro-USB port along with headers for providing 5V, GND, SCL, and SDA lines to the slaves. The master is

capable of maintaining communication with up to 8 enabled targets simultaneously with UART commands to change the I2C address of the enabled targets, allowing for the full range of 7-bit addresses. When VB commands a target to enable via UART, the master alters the signal to I2C and enables the target. Once a target is enabled, the master will periodically read from the slave to determine if a player has hit the target. When a hit is received from a target, the player number and I2C address of the target are transmitted to the VB program via UART.

5.3: Laser Arcade Calibration

There is currently nothing to calibrate on the blaster. However, if a red dot optic is mounted to the blaster with an IR laser in the future, it would require zeroing in the optic with the laser. The target system does not require any calibration, but the user has the option to modify the game settings. While not required, the user may want to adjust the timing, frequency or number of target activation/deactivation, as well as point values for certain targets or game length.

6: Blaster

6.1: Blaster Overview

The Laser Arcade blaster is the laser gun held by the players and used to engage targets with. The laser is modulated at either 38kHz or 56kHz corresponding to IR sensor frequencies, which is used to determine between player 1 or player 2 hitting the target. The blaster can select between these modulation frequencies on the fly using a switch. The user also has the option of the gun operating in full-auto, burst, or semi-auto, determined by adjusting a rotary switch mounted to the outside of the blaster. The blaster when firing also plays a Star Wars laser blaster audio file, and energizes a solenoid to move the slide in order to simulate recoil.

6.2: Blaster Mechanical Assembly

The Nerf gun chosen to be converted into the Laser Arcade blaster was the rival kronos pistol (shown below in Figures 6.2.1 and 6.2.2). This model was chosen for two reasons: it appeared to have plenty of room internally to mount all hardware and electronics required to make the blaster, and it had a moveable slide allowing a solenoid to move the slide and simulate recoil when fired.

There were several things that had to be done to this Nerf gun to convert it into the Laser Arcade blaster. A new barrel had to be designed in order to allow a laser to be mounted, a speaker mount had to be designed, and most trickily the recoil system which uses a solenoid to move the slide had to be designed. All of the Nerf guns original internals were removed or modified, the only things untouched were the exterior cosmetic components to keep the blaster looking like a normal, relatively unmodified Nerf gun.

Figure 6.2.1: Rival Kronos Nerf Gun

Figure 6.2.2: Disassembled Rival Kronos Nerf Gun

The barrel was designed to replicate the original Nerf gun barrel that allows us to mount a laser (Shown below in Figures 6.2.3 and 6.2.4). Additionally, a slot for a power switch to mount into the opposite end of the barrel was added. This allows the power switch to be tucked away and hidden from people handling the blaster at events and prevents them from playing with it. While hidden away however, it is still easily accessible by opening the hatch on the top of the blaster. See Section 13.1.2: Barrel for the blaster barrel drawings.

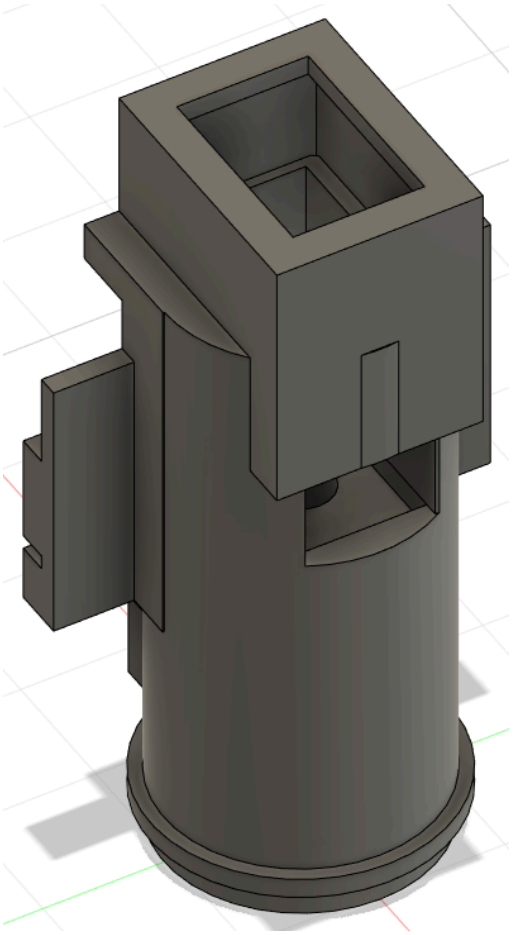


Figure 6.2.3: Blaster Barrel Rear

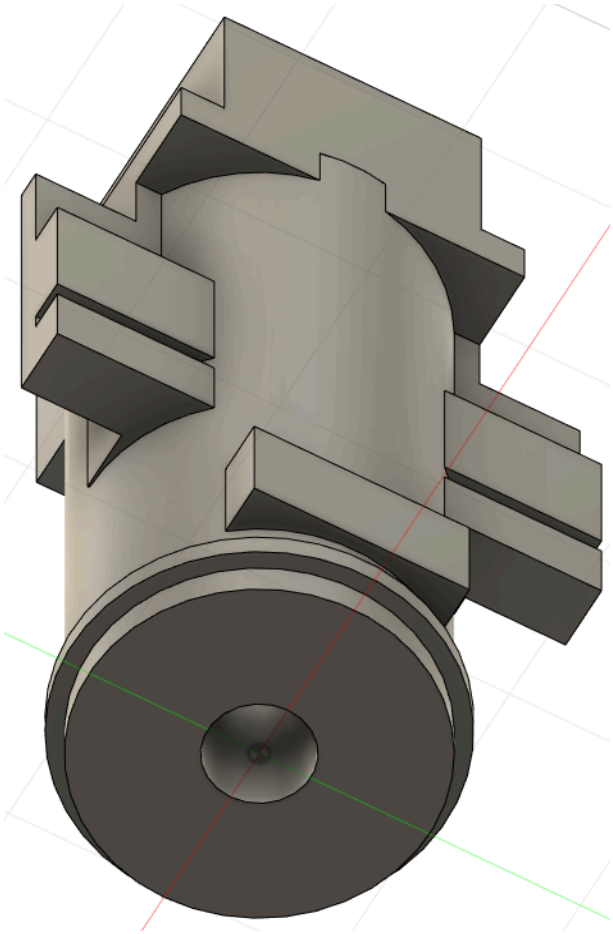


Figure 6.2.4: Blaster Barrel Front

The solenoid mount had to do two things: mount the solenoid in the perfect position to strike the slide to push it back, and hold one end of a spring stationary, to allow the spring to return the slide back to its original position after firing. So a piece was designed to slide into a slot at the rear of the blaster. A cutout was then made for the solenoid, as well as a backplate to fasten the solenoid to using two screws (Seen in Figure 6.2.6 below). Then at the top of the solenoid mount, a hook was designed for a spring to clip on to (Seen in Figure 6.2.5 below). See Section 13.1.3: Solenoid Mount for the solenoid mount drawings.

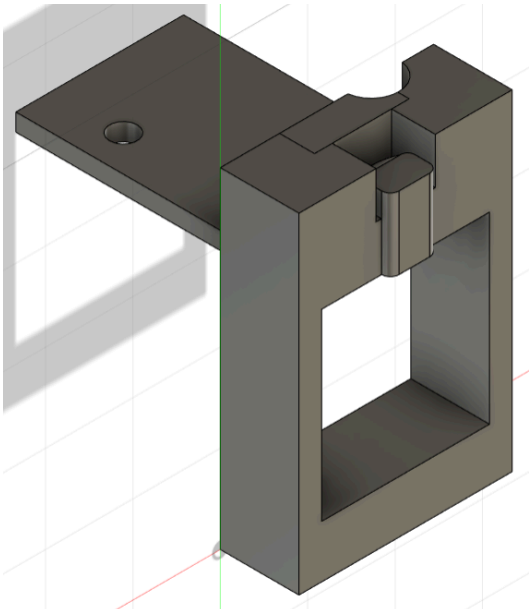


Figure 6.2.5: Blaster Solenoid Mount Front

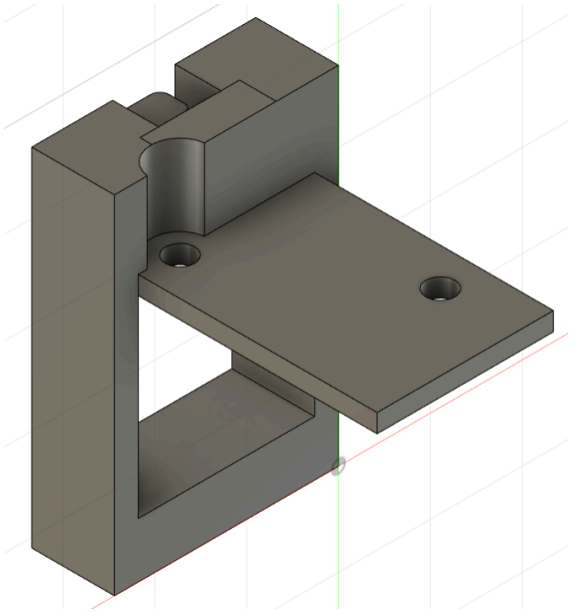


Figure 6.2.6: Blaster Solenoid Mount Rear

A stationary mount for the spring was designed, but then a mount for the other end of the spring, attaching it to the slide had to be made as well. The final model was a piece that fit through a slot in the back of the slide, which gets held in place between both sides of the slide. It extends slightly towards the solenoid with a slot for the ring on the end of the spring to slide into, and a hole going across for a screw to secure the end of the spring on (See Figure 6.2.7 below). To see the drawings for the slide spring mount, see Section 13.1.4: Slide Spring Mount.

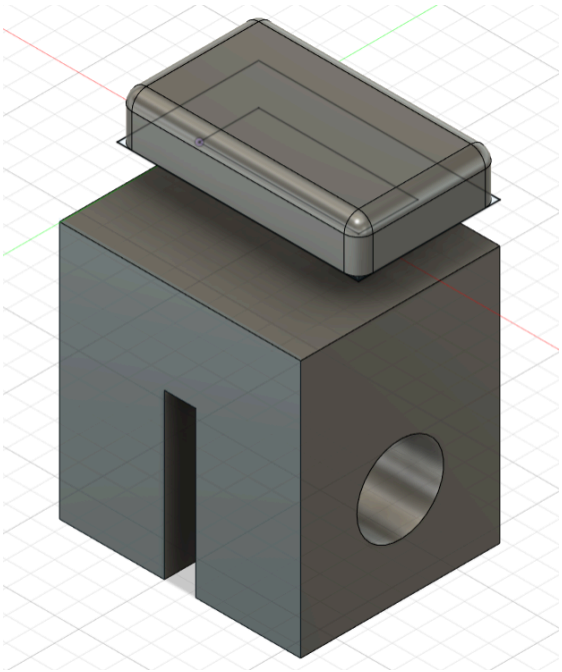


Figure 6.2.7: Blaster Slide Spring Mount

The final part that needed to be designed for the blaster, was a box to not only mount the 1 inch speaker chosen for this project, but also put something behind it to make the audio louder. The final model was a simple box with a slot for the speaker, and a hole for the wires to feed through, as well as a way to mount the box just below the barrel (Seen in Figure 6.2.8 below). Initially the box was square, but was later rounded on the bottom in order to give it additional clearance from the walls of the Nerf gun which prevented it from being able to be put back together. To see the drawings for the speaker box, see Section 13.1.1: Speaker Box.

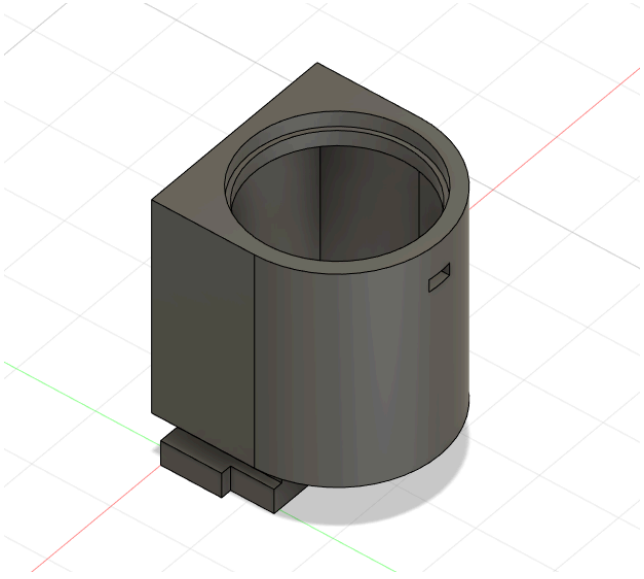


Figure 6.2.8: Blaster Speaker Box

After modeling all of the parts necessary for the blaster, it was time to put the blaster together. All modeled parts and PCB (See Section 6.3) can be seen mounted in the blaster below in Figure 6.2.9. It is easiest to mount everything on the right side of the blaster before putting it together. The barrel is mounted in the top left, just above the speaker box which is facing the PCB. In the rear of the blaster, the solenoid mount is in place with a spring attaching it to the slide spring mount on the slide itself. There are two mounted switches on the blaster: the fire mode selector switch mounted on the exterior of the left side of the blaster, and the trigger switch which is soldered to a cut down proto-board with a switch and two wires feeding to the PCB soldered onto it. The trigger switch is glued in place on a bracket already on the Nerf gun.

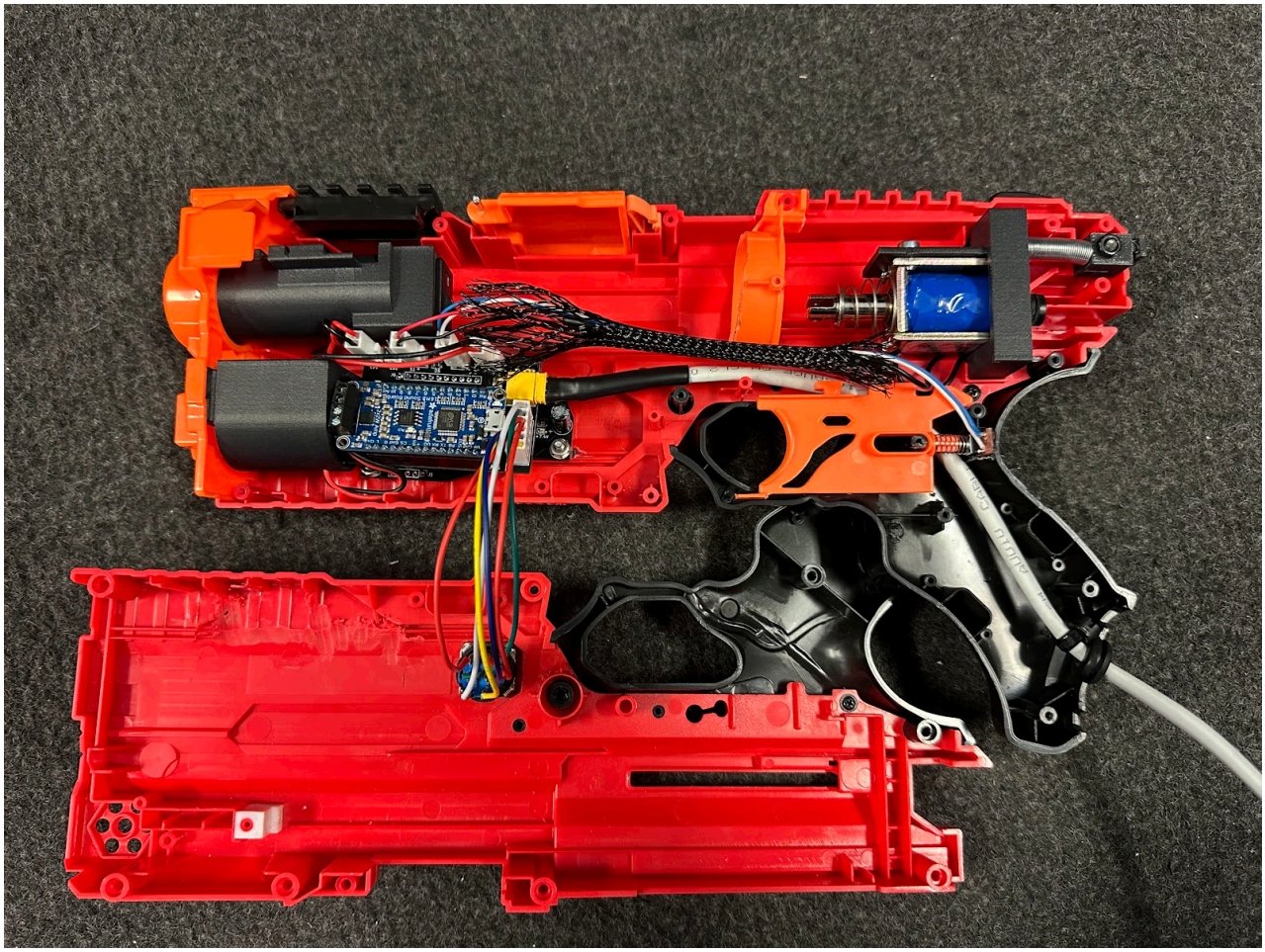


Figure 6.2.9: Internal View of Assembled Blaster

There were several modifications that had to be made to the Nerf gun in order to mount everything properly. First, everything had to be removed from the blaster. Then using a dremel, a notch was made to give the solenoid wires additional clearance. A hole was cut in the grip for a power cable, a cutout was made under the trigger switch to make room for the power cable, and the chamber was shortened down to only the part that keeps the slide together. Then a few holes had to be drilled: 4 to mount the PCB, and a larger one to mount the fire mode selector switch through. Then once this is all completed, the blaster is ready to be assembled as shown in Figure 6.2.9, and put together to look like the blaster shown in Figures 6.2.10, and 6.2.11.

Figure 6.2.10: Assembled Blaster Left Side

Figure 6.2.11: Assembled Blaster Right Side

6.3: Blaster PCB

Space inside of the Nerf gun was a concern when putting the blaster together, as it is very limited as seen in Figure 6.2.9. As a result, the blaster PCB could not be laid out flat, but was designed in a daughter board configuration with header footprints that lined up exactly so that one could be mounted on top of the other by soldering a socket on the bottom and headers on top as shown below in Figure 6.3.1. This allowed the PCB to have double the board space to mount everything for the blaster, while taking up a small footprint inside of the Nerf gun. To see the KiCad renders of the PCB looks like see Section 12.1: Blaster.

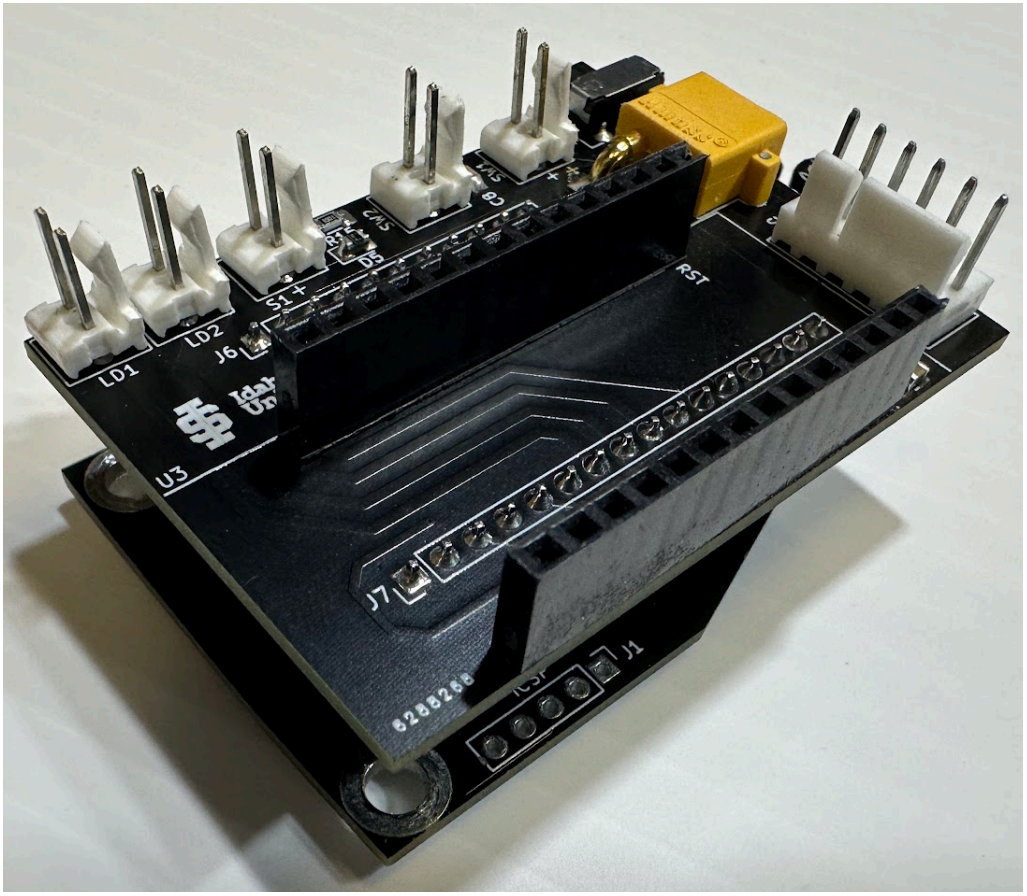


Figure 6.3.1: Blaster PCB

The current PCB design has the majority of the electronics including the PIC, regulator, and output drivers on the bottom board, while the top board acts as an I/O board to connect inputs and outputs to. The top has headers for two laser outputs, one solenoid output, the fire switch, power switch, the fire mode selector switch, a socket to mount the audio board, the power input connector, as well as the frequency select switch that is conveniently reachable by opening the hatch on the top of the blaster.

Not only are the headers being used for mounting, but they also are used to carry signals between the boards. The shorter 11 pin headers J4/J6 are used to carry the TRIGx signals from the PIC on the bottom to the audio board on the top. The J4/J6 headers have the following pinout:

Table 6.3.1: J4/J6 Pinout

Pin	Signal
1	TRIG10
2	TRIG9
3	TRIG8
4	TRIG7
5	TRIG6
6	TRIG5
7	TRIG4
8	TRIG3
9	TRIG2
10	TRIG1
11	TRIG0

The larger 16 pin headers J5/J7 are used to carry various input and output signals between the bottom boards with the following pinout:

Table 6.3.2: J5/J7 Pinout

Pin	Signal
1	Trigger Input
2	Frequency Select Input
3	Full-Auto Select Input
4	Burst Select Input
5	Semi-Auto Select Input
6	VOL+
7	Laser/PWM
8	Laser/PWM2
9	Solenoid
10	
11	GND
12	
13	5V
14	
15	12V
16	

Note: Signals using more than one pin are due to higher than 1A expected current for those specific signals.

6.4: Blaster Features

6.4.1: Laser Output

The Laser Arcade blaster has up to two laser outputs connected to the CPP1 and CCP2 modules of the PIC, and controls a MOSFET (Q1 or Q2) to switch the lasers on/off by shorting it from 5V to GND. The PWM output is constantly modulating at one of the two player sensor frequencies (38kHz-Player 1, 56kHz-Player 2) with a ~95% duty cycle to maximize the brightness of the laser. However, the output driver is disabled by default, resulting in nothing on the output. When the trigger (RB0) is pressed, the PWM output then enables for about 70 ms (controlled by timer 1) to fire, before disabling again for the same amount of time. This operation can be observed in the full-auto firing mode below in Figure 6.4.1.1. As the trigger (Ch1) is being pressed, the laser output (Ch2) is off for 50% of the time, modulating the laser at the selected player frequency, allowing the sensors on the target to determine which of the two players hit the target.

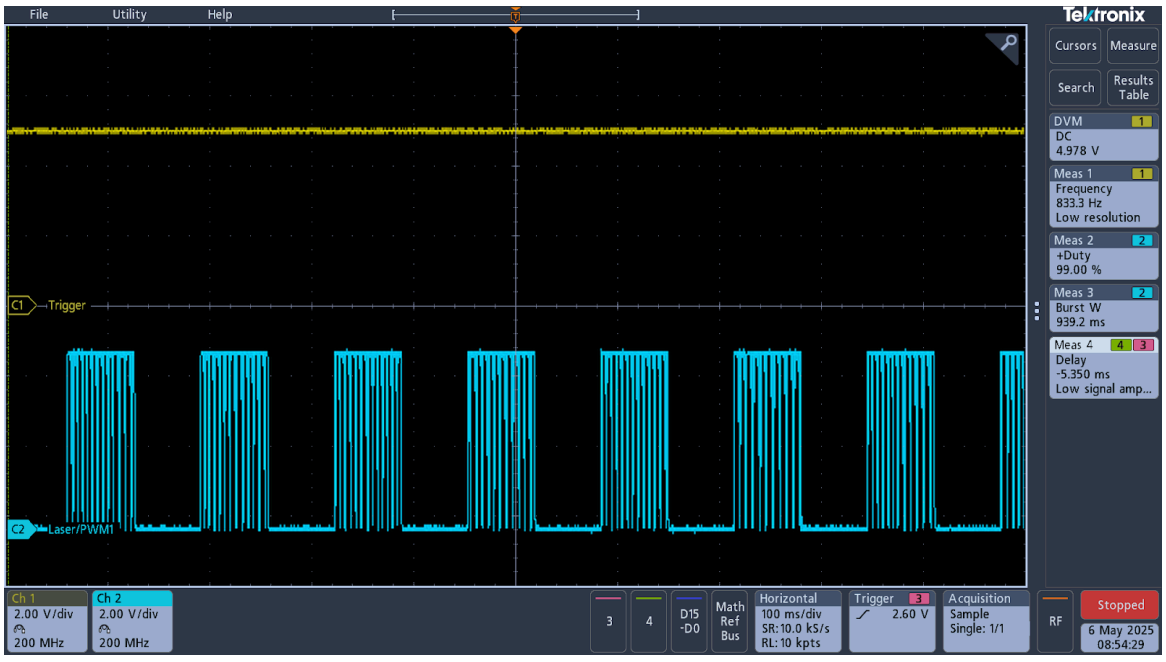


Figure 6.4.1.1: Laser Output Waveform

6.4.2: PWM Frequency Select

Each blaster is equipped with a frequency select switch, which is easily accessible on the top of the PCB after opening the hatch on the top of the blaster. It is a slide switch that determines whether a high or a low is on the frequency select pin (RB1) of the PIC. This allows any blaster to act as either player 1 or player 2, and be able to easily change it quickly. This adds redundancy in the event that a blaster stops working, so that any backup blaster will be able to take its place without requiring taking apart the blaster and reflashing the PIC.

Each time the blaster is fired, before anything else it determines the state of RB1 and sets the PWM frequency. A logic low on the frequency select pin results in a 38kHz modulation of the laser, while a logic high results in a 56kHz modulation of the laser (Refer to Table 6.4.2.1 below). To observe the waveforms demonstrating this feature, see Figures 6.4.2.1 and 6.4.2.2 below).

Table 6.4.2.1: Frequency Select Truth Table

Frequency Select	PWM Frequency
0	38kHz
1	56kHz

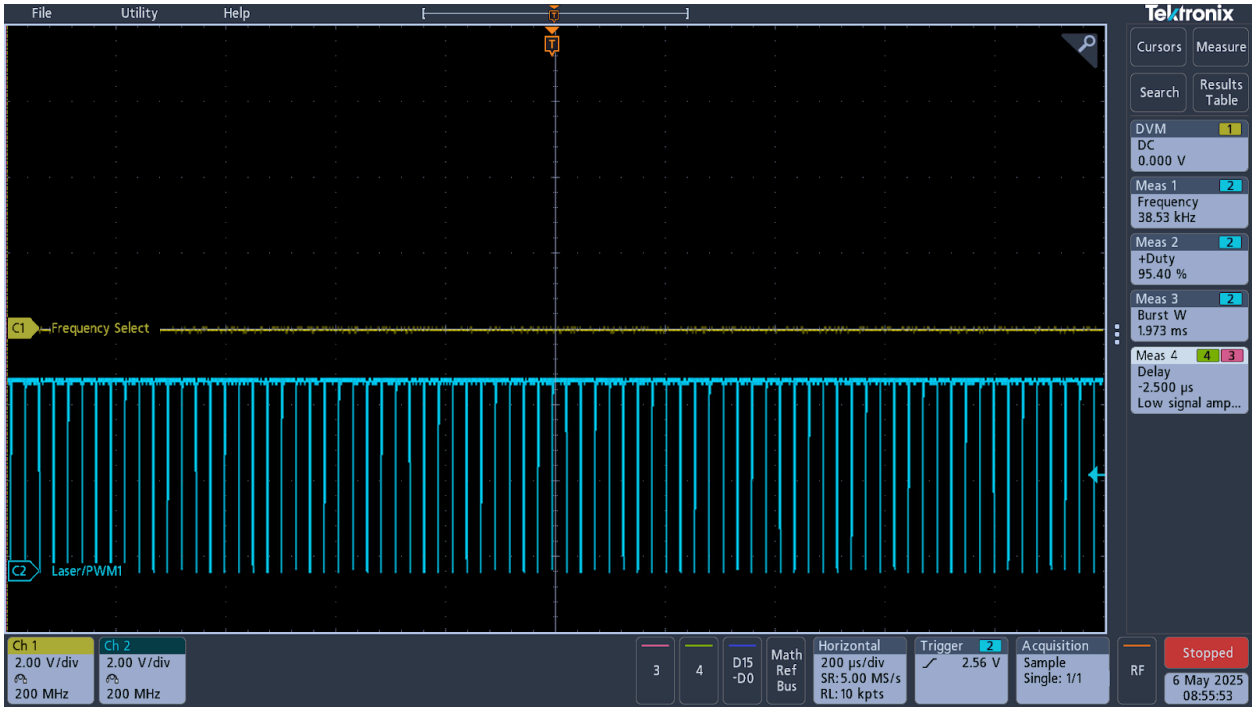


Figure 6.4.2.1: Frequency Select vs Modulation Frequency (RB1=0)

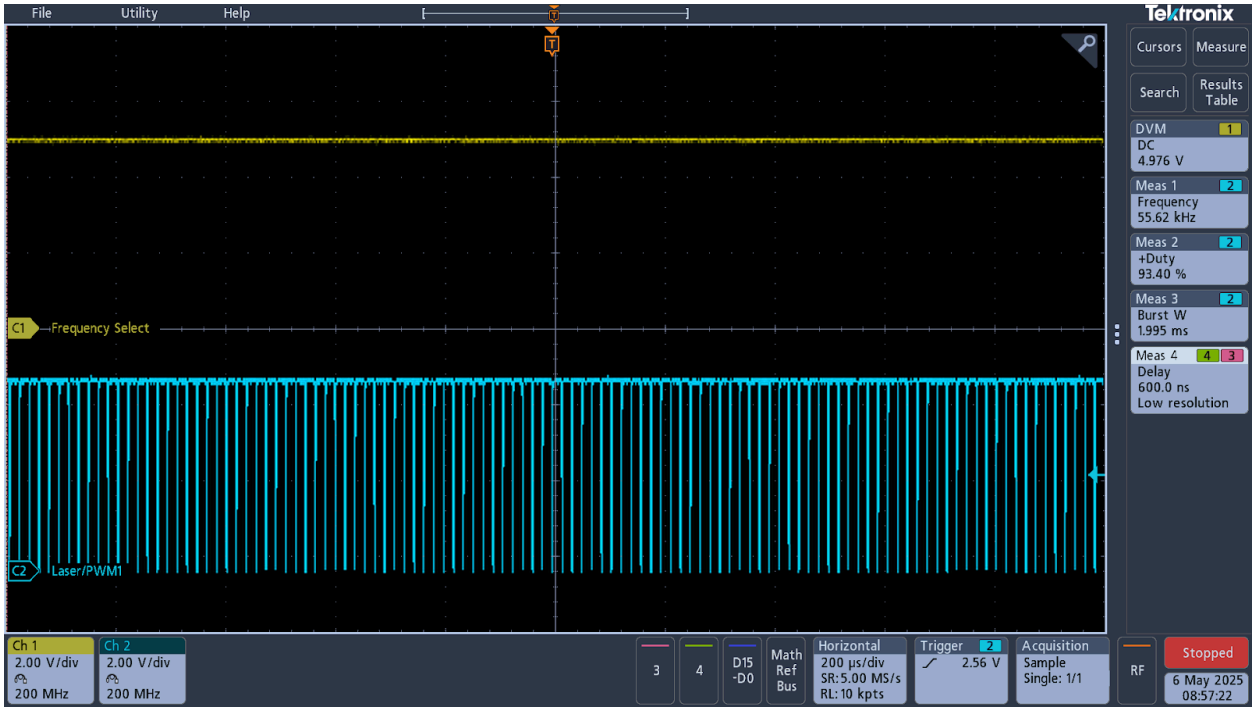
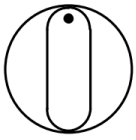
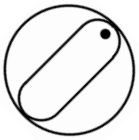


Figure 6.4.2.2: Frequency Select vs Modulation Frequency (RB1=1)

6.4.3: Fire Mode Select

The blaster has three firing mode operations: semi-auto, burst, and full-auto. The firing mode operation is determined by the position of the rotary fire mode select switch mounted on the outside of the blaster. The semi-auto firing mode (switch forward position) will toggle the outputs a single time to fire a single shot each time the trigger is pressed. The burst (switch center position) toggles the outputs to fire the blaster three times each time the trigger is pressed. And the full-auto (switch rear position) will toggle the outputs continuously as long as the trigger is pressed and held down. When firing, the modes are tested in the order of: full-auto, burst, and semi-auto. See Table 6.4.3.1 below for the firing operation of each position of the switch. Additionally, see Figures 6.4.3.1, 6.4.3.2, and 6.4.3.3 for waveforms demonstrating the operation of the fire mode select switch.

Table 6.4.3.1: Fire Mode Select Switch Position vs Fire Mode

Fire Mode Select Switch Position	Fire Mode
	Semi-Auto
	Burst
	Full-Auto

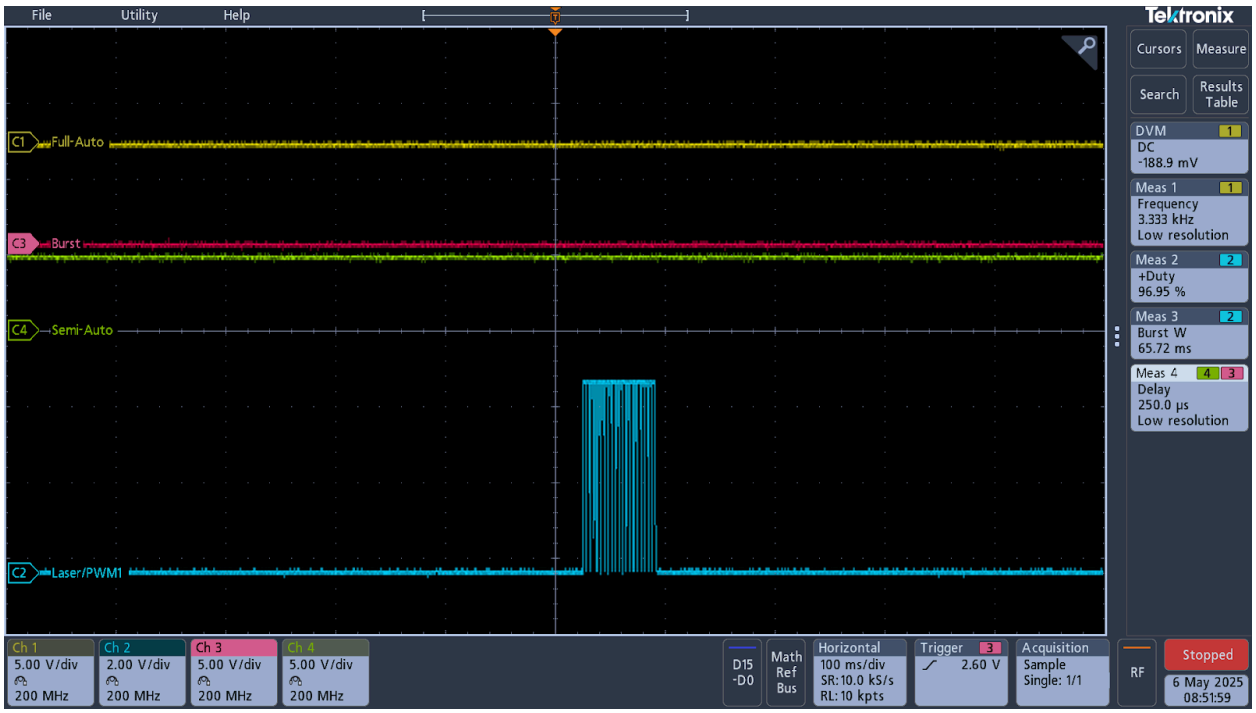


Figure 6.4.3.1: Fire Mode Select Switch Position vs Operation (Semi-Auto)

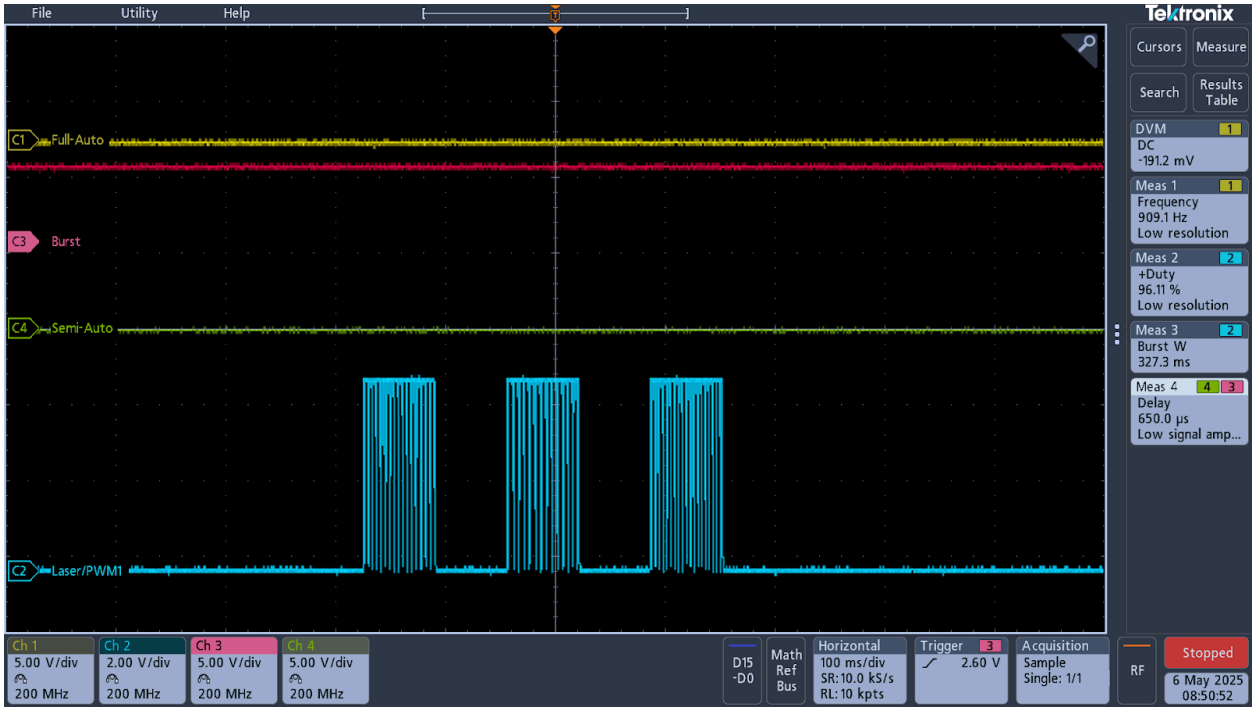


Figure 6.4.3.2: Fire Mode Select Switch Position vs Operation (Burst)

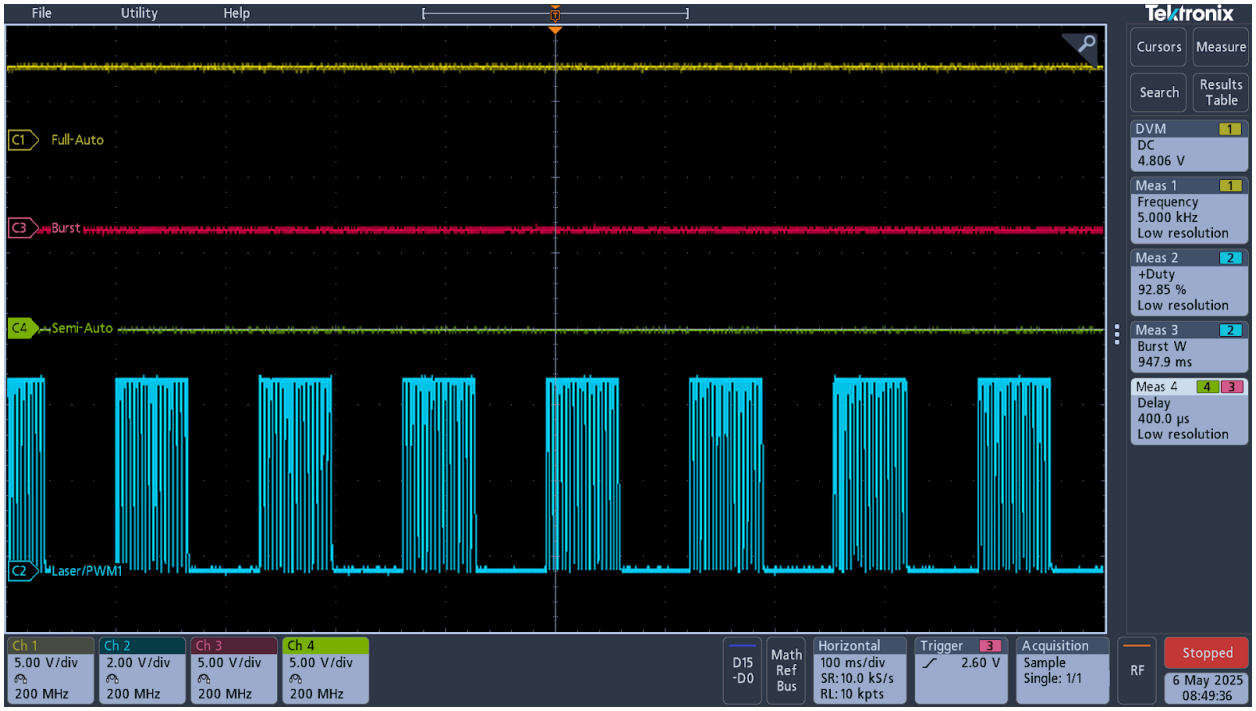


Figure 6.4.3.3: Fire Mode Select Switch Position vs Operation (Full-Auto)

6.4.4: Solenoid Output

The blaster has a single solenoid output responsible for moving the slide when the blaster is fired to simulate recoil. The PIC sets the solenoid output (RC3) high at the same time as it enables the PWM output drivers to keep the recoil and firing of the blaster synchronized (as seen in Figure 6.4.4.1 below). The solenoid output of the PIC controls a MOSFET (Q3) to drive the solenoid the same as the laser output, but gives it a continuous high rather than modulating for the same intervals, and shorts it from 12V to GND rather than 5V to GND.

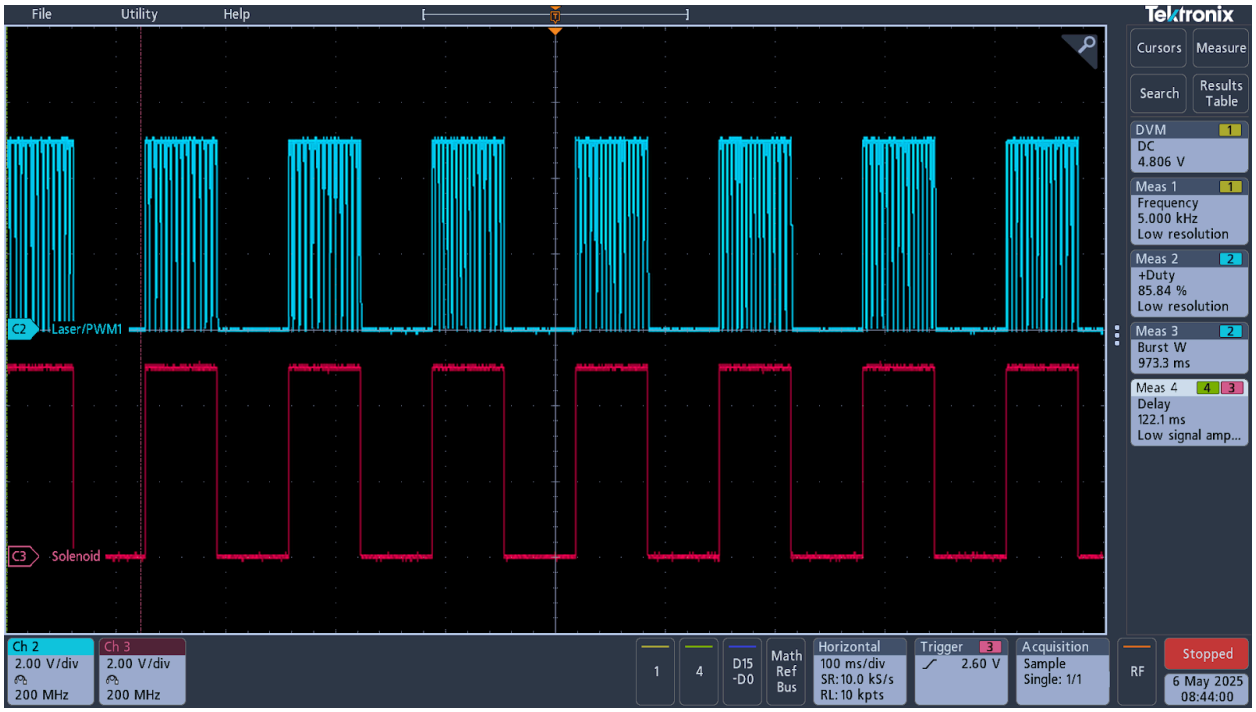


Figure 6.4.4.1: Solenoid Output vs Laser/PWM Output

6.4.5: Audio Output

The blaster features an AdaFruit Audio FX sound board with a built-in amplifier to play audio through a 1 inch speaker mounted to the front of the blaster each time the blaster is fired. The current audio files loaded onto the sound board are Star Wars blaster sounds with two files, one for the semi-auto and full-auto firing modes, and one for the burst firing mode (audio files can be found in the project GitHub repository, see Section 17: Appendix G (GitHub)).

The reason for two audio files is timing. The sound board is slow, and takes ~100 ms to trigger audio. In full-auto it doesn't matter because it can continuously repeat the file as long as the trigger is held low. It is harder to notice any lag when the blaster is constantly firing, thus there was no need for any delays in this mode (see Figure 6.4.5.1). For the burst fire mode, there simply isn't enough time to trigger the audio file three times, so it is easier to use a different trigger to play a different audio file with three blaster shots than it is to play a single one three times. A slight delay by triggering the audio and sitting in an in-line delay was added to synchronize the audio with the shots better (see Figure 6.4.5.2).

A significantly extended delay was required on the semi-auto firing mode however (see Figure 6.4.5.3). The current routine for the semi-auto firing mode is to enable the outputs, wait for timer 1 to overflow, and then leave the subroutine and disable all outputs. The issue with this is that the blaster is able to fire and complete that routine before the sound board has any chance to register the trigger. As a result the audio was not able to play at all initially, and a delay was added. Now, the blaster enables the audio trigger then sits in an in-line loop for ~65ms and then enables the other outputs, giving the soundboard enough time to play the audio file before the blaster stops firing.

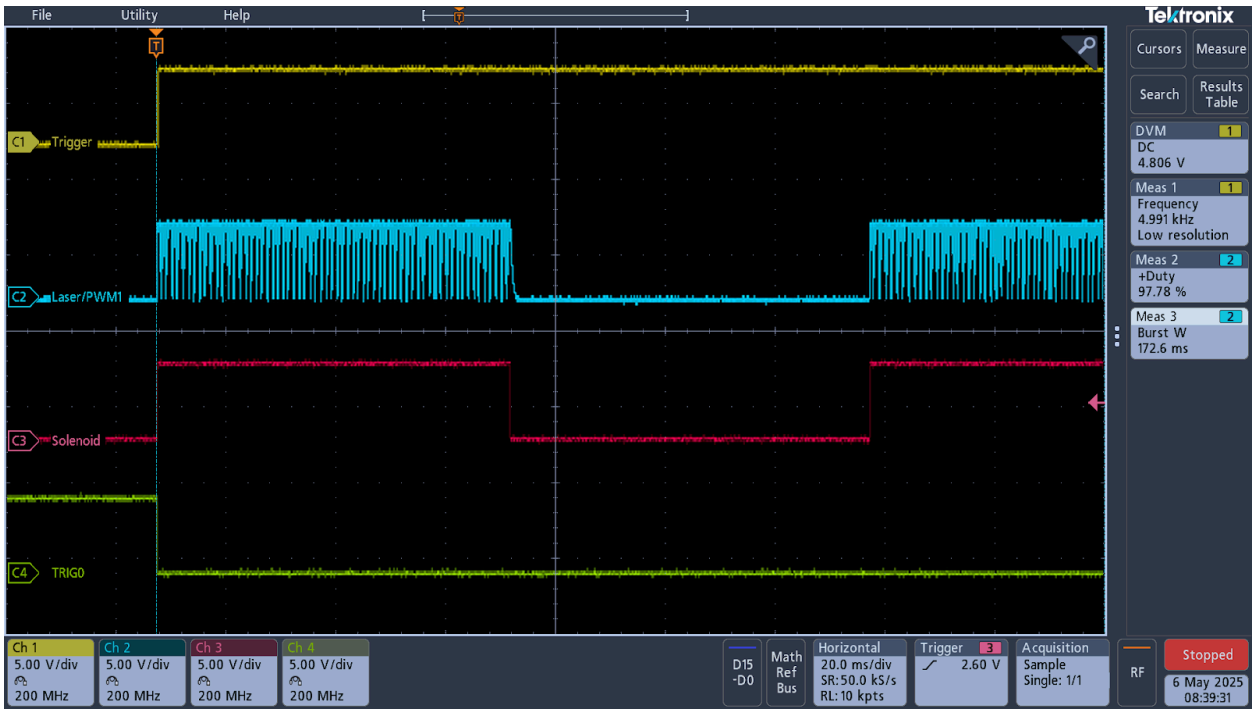


Figure 6.4.5.1: Audio Trigger Timing (Full-Auto)

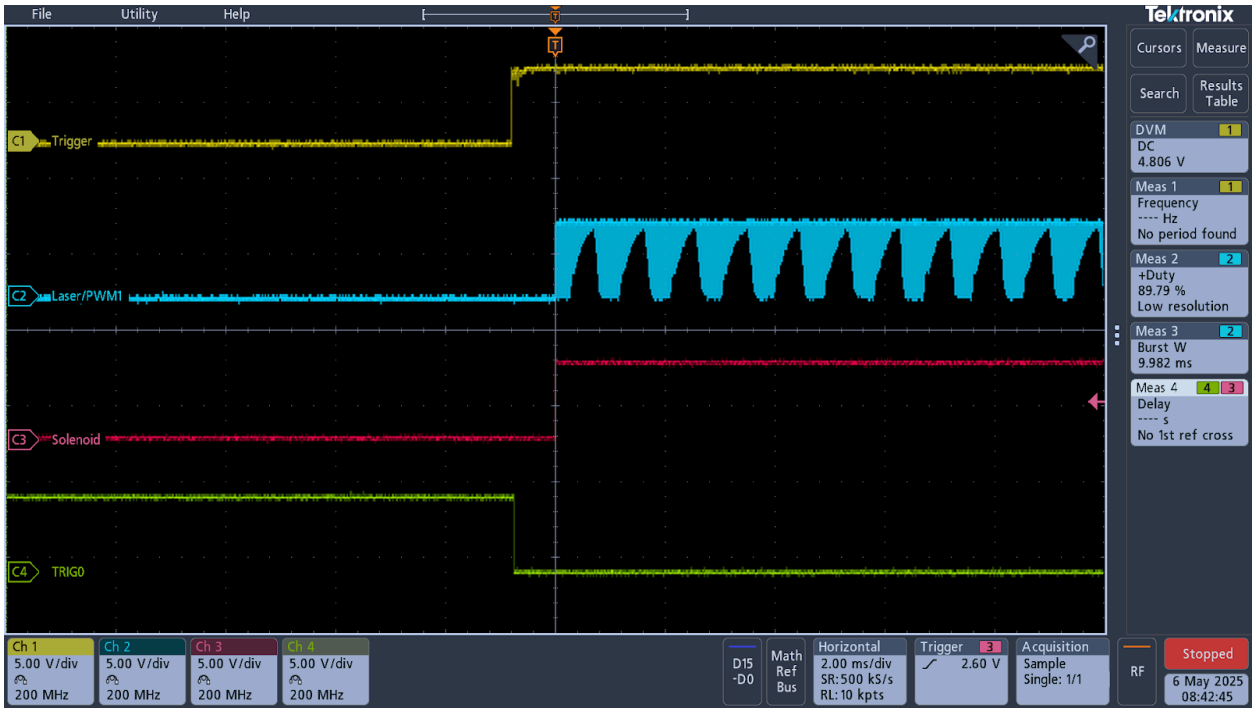


Figure 6.4.5.2: Audio Trigger Timing (Burst)

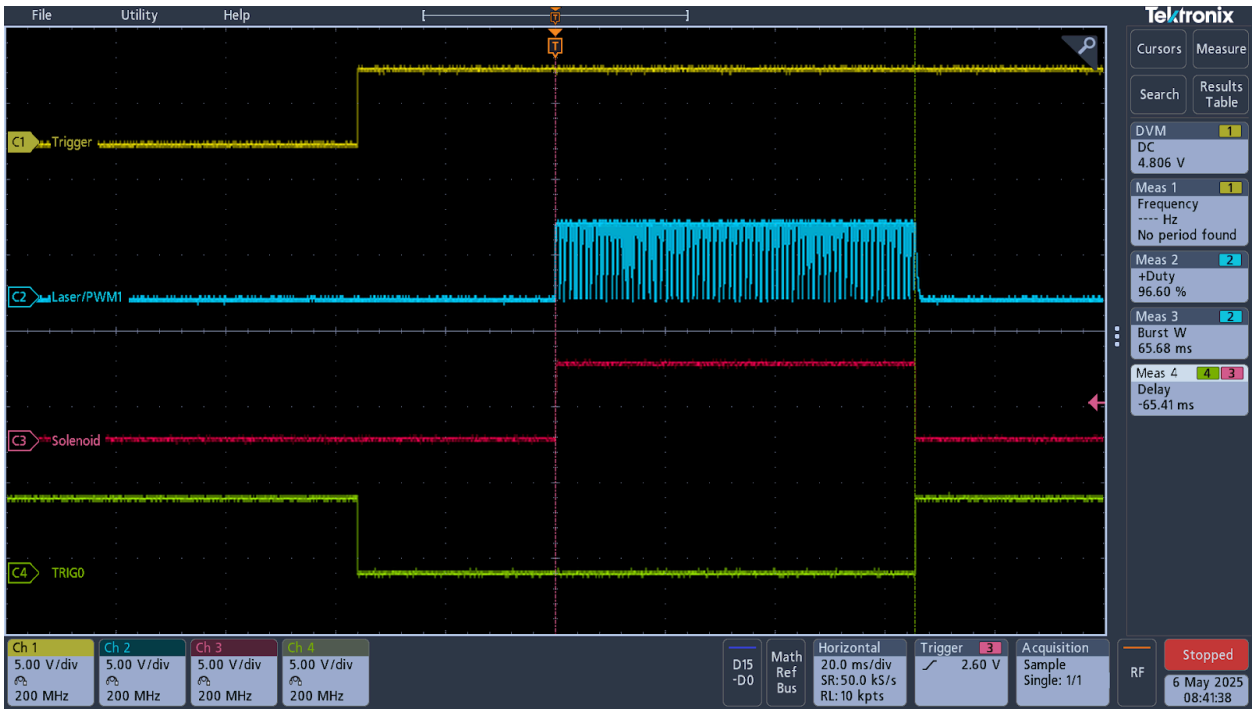


Figure 6.4.5.3: Audio Trigger Timing (Semi-Auto)

6.4.6: Blaster Power

The Laser Arcade blaster is powered by a wire fed through a hole in the grip of the blaster that connects from the battery connector on the top PCB to a 12V DC power supply. It is possible to run the blaster with a battery, however if it is less than 12V the solenoid will be less powerful and move the slide less, but will otherwise continue to function normally.

The PIC and laser diode would be damaged by 12V, both requiring 5V. U1 (TLV1117-50) is a set 5V output linear regulator to provide these components with 5V, with a max 800mA output (See Figure 11.1.2: Blaster Power Schematic).

6.5: Blaster Troubleshooting

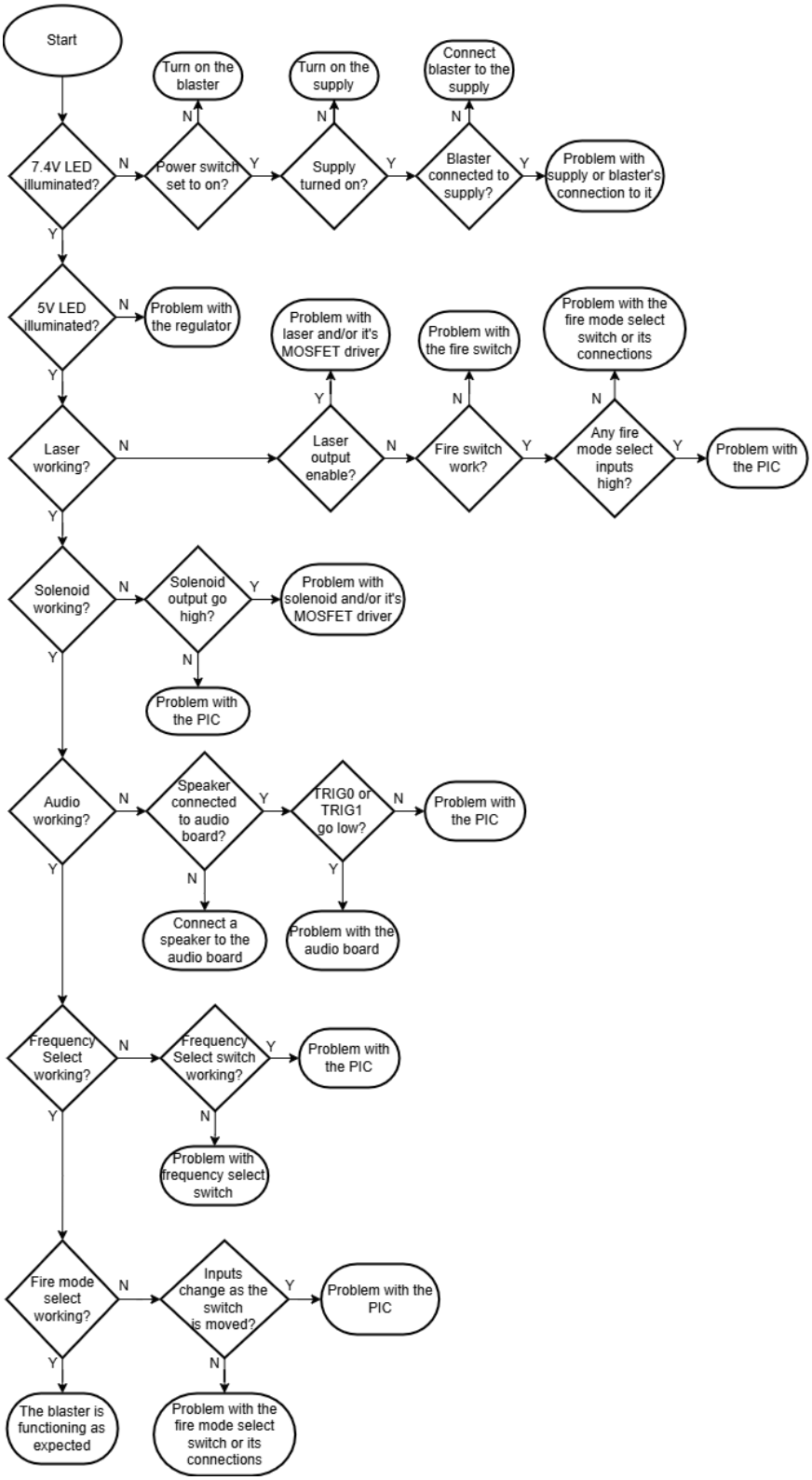


Figure 6.5.1: Blaster Troubleshooting Flowchart

To troubleshoot the blaster, follow the steps below, or follow Figure 6.3.1.

1. Is the 7.4V LED illuminated?
    a. If yes, move to step 2.
    b. If not, move to step 8.
2. Is the 5V LED illuminated?
    a. If yes, move to step 3.
    b. If not, there is a problem with the 5V linear regulator (U1).
3. Is the laser turning on when trying to fire the blaster?
    a. If yes, move to step 4.
    b. If not, move to step 12.
4. Is the solenoid energizing when firing the blaster?
    a. If yes, move to step 5.
    b. If not, move to step 16.
5. Is audio being played when firing the blaster?
    a. If yes, move to step 6.
    b. If not, move to step 18.
6. Is the frequency select working?
    a. If yes, move to step 7.
    b. If not, move to step 21.
7. Is the fire mode select working?
    a. If yes, then the blaster is functioning as expected.
    b. If not, move to step 22.

8. Is the power switch (SW1) closed? It is closed when the line "|" is pushed down.
    a. If yes, move to step 9.
    b. If not, flip the switch to turn the blaster on.
9. Is the 12V power supply plugged in and turned on?
    a. If yes, move to step 10.
    b. If not, ensure that the 12V supply is plugged in and turned on. If it is plugged in and not turned on, then there is a problem with the DC power supply.
10. Is the blaster connected to the supply, with one end of the power cable connected to the supply, and the other end properly inserted into the XT30 connector on the top board?
    a. If yes, move to step 11.
    b. If not, make sure that both ends of the power cable are properly connected.
11. Does the power switch (SW1) function properly? Test continuity between the two contacts with the switch in each position. It should be a short one way, and open the other.
    a. If yes, there is a continuity problem between the power input and the bottom board.
    b. If not, the problem is that SW1 is faulty.
12. Measure the corresponding Laser/PWM test point. When trying to fire the laser, does the PWM output enable?
    a. If yes, then move to step 13.
    b. If not, then move to step 14.
13. Unplug the laser, and connect it from 5V to GND. Does the laser turn on?

   a. If yes, then the problem is with the corresponding MOSFET driver, Q2 for Laser/PWM, and Q1 for Laser/PWM2
   b. If not, then there is a problem with the laser, and it needs to be replaced.

14. Measuring the Fire test point, does the measured voltage read 0V when the trigger is not pressed, and 5V when pressed?
   a. If yes, then move to step 15.
   b. If not, then there is a problem with the fire switch.
15. Measure the voltage on the fire mode select test points (labeled: Full, Burst, and Semi). Does at least one measure 5V?
   a. If yes, then there is a problem with the PIC.
   b. If not, then there is a problem with the fire mode select switch.
16. Measure the Solenoid test point. When pressing the fire switch, does the Solenoid output go high?
   a. If yes, move to step 17.
   b. If not, then there is a problem with the PIC.
17. Unplug the solenoid and short it from 12V to GND. Does the solenoid coil energize?
   a. If yes, then there is a problem with the solenoid's MOSFET driver Q3.
   b. If not, then there is a problem with the solenoid.
18. Is there a speaker properly connected to the audio board output?
   a. If yes, move to step 19.
   b. If not, connect a speaker to the output.
19. Measure the audio board trigger output (TRIG0 for full-auto and semi-auto firing modes, and TRIG1 for burst fire mode. Does the input go low when the fire switch is pressed?

   a. If yes, move to step 20.
   b. If not, there is a problem with the PIC.

20. Does the audio board have audio files?
   a. If yes, there is a problem with the audio board.
   b. If not, drop the audio files onto the audio board. The audio files can be found in the GitHub repository (see Section 17: Appendix G (GitHub)).

21. Measure the Freq. test point and slide the frequency select switch, does the measured voltage change between 5V and 0V?
   a. If yes, there is a problem with the PIC.
   b. If not, there is a problem with the frequency select switch or its continuity to the bottom board.

22. Measure the fire mode select test points (labeled: Full, Burst, and Semi) and rotate the fire select switch. Do the fire mode select inputs change as you rotate the switch?
   a. If yes, there is a problem with the PIC.
   b. If not, there is a problem with the fire mode select switch or its connections.

7: Target System

7.1: Target System Overview

The Laser Arcade target system comprises three parts: the master, the slave, and a Visual Basic program. The master and slave communicate with each other through the I2C protocol and the master communicates with the VB program through UART. The system utilizes a 5V 12A DC power supply connected to the master where it is distributed to the slave boards along with GND, SCL, and SDA lines.

The master board utilizes a system of eight target slots to support enabling up to eight targets simultaneously while still polling each target for player hits. These target slots store the I2C address and status of the corresponding target. Both the I2C address and the status of each target slot is changed with a UART command sent by the VB program. These commands allow VB to enable, disable, or change the I2C address of any of the target slots.

7.2: UART

7.2.1: UART Overview

The Laser Arcade utilizes UART for communication between the master board and the Visual Basic program. The master connects to a computer through micro USB. The USB data is converted to TTL UART using the FT232RL, a USB to UART converter IC. For connecting to the laser arcade, the UART baud rate is 9600 and transmits 8 data bits with no parity and a single stop bit

7.2.2: UART Command Structure

The UART command structure for the laser arcade consists of 5 different commands used by the VB program and two different responses from the master board. In all communication between the PIC and VB, a handshake byte is always sent first to signify the start of a data packet. The handshake byte for the laser arcade is an ASCII $ (0x24). The handshake is followed by a command byte and any other relevant data required for the command. Table 7.2.1 below is the target system UART command structure sent by the VB program.

Table 7.2.1: UART Command Structure

	Byte 1	Byte 2	Byte 3	Byte 4
Device Verification	$	V (0x56)		
Enable Target	$	E (0x45)	TargetSlots	
Disable Target	$	D (0x44)	TargetSlots	
Disable All Targets	$	C (0x43)		
Change Target	$	A (0x41)	ChangeSlot	I2C Address

Table 7.2.1 above lists the commands sent from the VB program to the master board. A single command is a maximum of 4 bytes of data with the minimum being 2 bytes of data. Any byte shown as blank in the table is representative of no transmission. Of the 5 UART commands, the device verification command is the only one requiring a direct response from the master via UART. All other commands do not solicit a UART response and instead wait for the master to send data when it is available. Table 7.2.2 below shows the structure of data transmitted from the master to the VB program.

Table 7.2.2: Master UART Transmission Structure

	Byte 1	Byte 2	Byte 3
Device Verification	$	L (0x4C)	A (0x41)
Target Hit	$	I2C Address	Player

Table 7.2.2 above shows the structure of data transmitted via UART from the target master board. The master transmits data back to the VB program for data verification or when a target slave reports having been hit by a player.

7.2.3: Device Verification

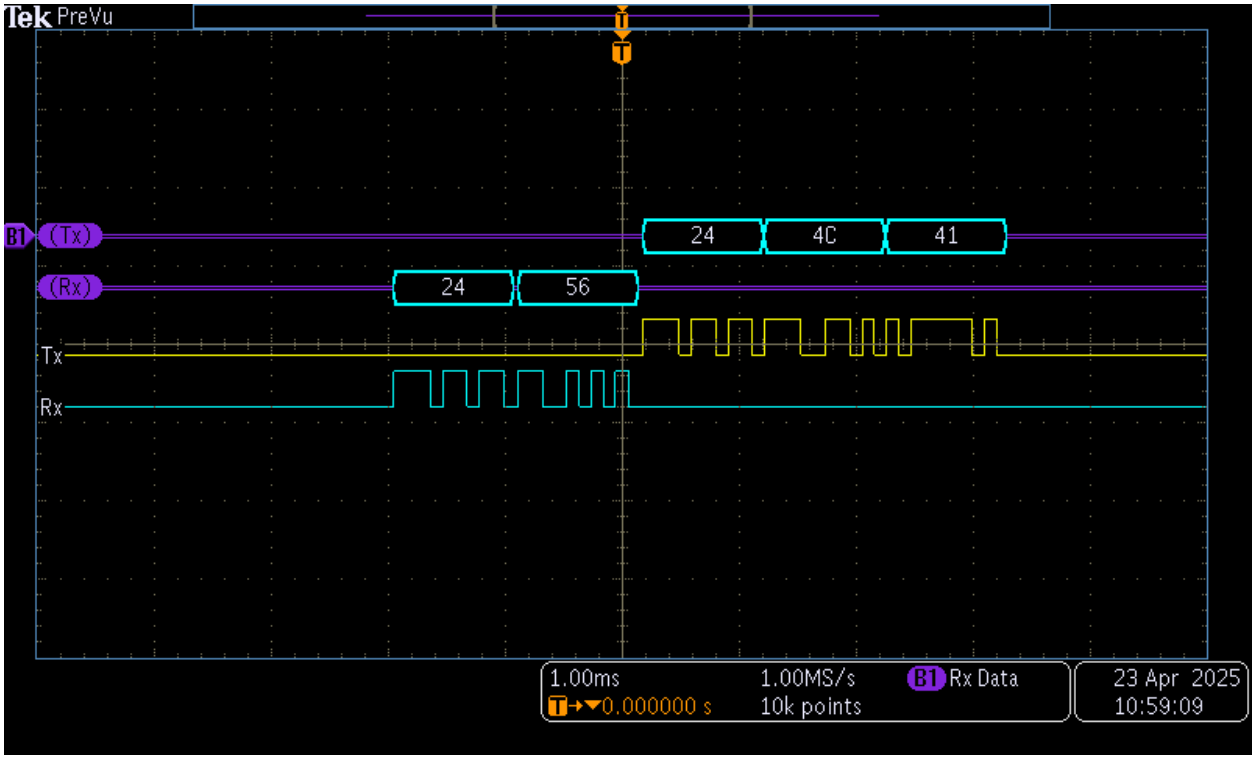


Figure 7.2.1: UART Device Verification

Figure 7.2.1 above is an oscilloscope capture of the UART communications between VB and the master board during the device verification command. The device verification command is used to ensure commands are being sent to the correct communications port in VB. This command consists of an ASCII $ (0x24) for the handshake followed by an ASCII V (0x56) for the command byte. When this is received, the device verification response is directly transmitted by the master. This response consists of the ASCII $ (0x24) followed by an ASCII L (0x4C) and

ASCII A (0x41). By comparing the master's response with the expected response, VB is able to verify the correct device is connected and is communicating properly.

7.2.4: Target Enable

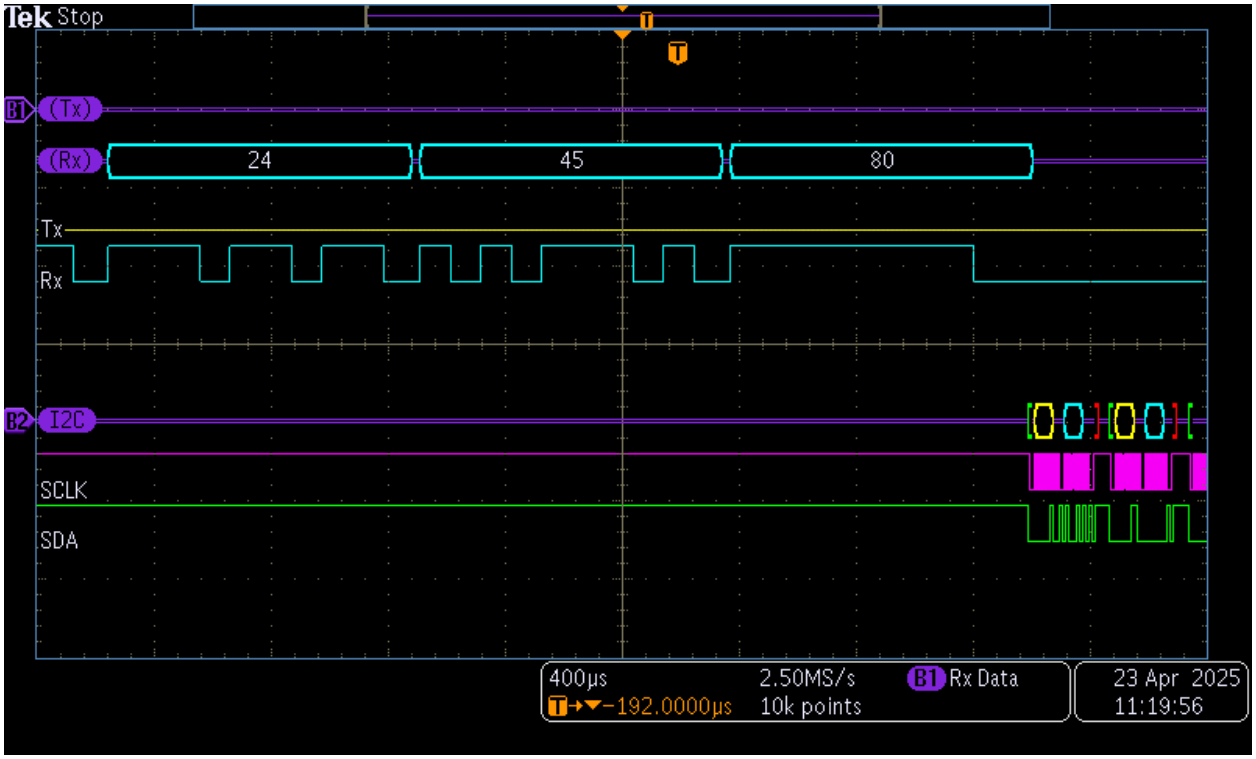


Figure 7.2.2: UART Target Enable

Figure 7.2.2 above is an oscilloscope capture of both the UART and I2C communication lines during a UART device enable command. The command packet consists of the handshake byte, an ASCII $ (0x24), followed by the enable command, ASCII E (0x45). The enable command requires an additional third byte used to determine which target slot in the master to use. Each bit in the TargetSlots byte represents a target slot. When a bit is high, the target associated with that slot will be enabled. When the bit is low, the target will remain in the same state as it is currently in. When received, the master ORs the TargetSlots byte with the

ENABLE_TARGETS register, requesting that the main loop of the PIC enable the target in the selected slot. Table 7.2.3 below maps each bit of the TargetSlots byte to its corresponding target slot.

Table 7.2.3: UART TargetSlots byte bitmap

Byte	B7	B6	B5	B4	B3	B2	B1	B0
TargetSlots	TGT_1	TGT_2	TGT_3	TGT_4	TGT_5	TGT_6	TGT_7	TGT_8

Table 7.2.3 above shows the bitmap of the TargetSlots byte used in the UART enable command. Each target slot has a specific bit associated with it. A logic high in any of the target slot bits commands the master to enable the corresponding target. Following the TargetSlots byte in UART, the master will write to the assigned target utilizing I2C communications. Figure 7.2.2 shows the command for enabling the target in slot 1. After receiving the command, the master will write to and begin polling the target for player hits using the I2C bus. See section 7.3 for the I2C communication structure.

7.2.5: Target Disable

The target system has two different UART commands for disabling targets. The main target disable command is three bytes and can disable individual target slots. The disable all targets command is two bytes and disables all target slots on the master. Each command will be explained in detail below.

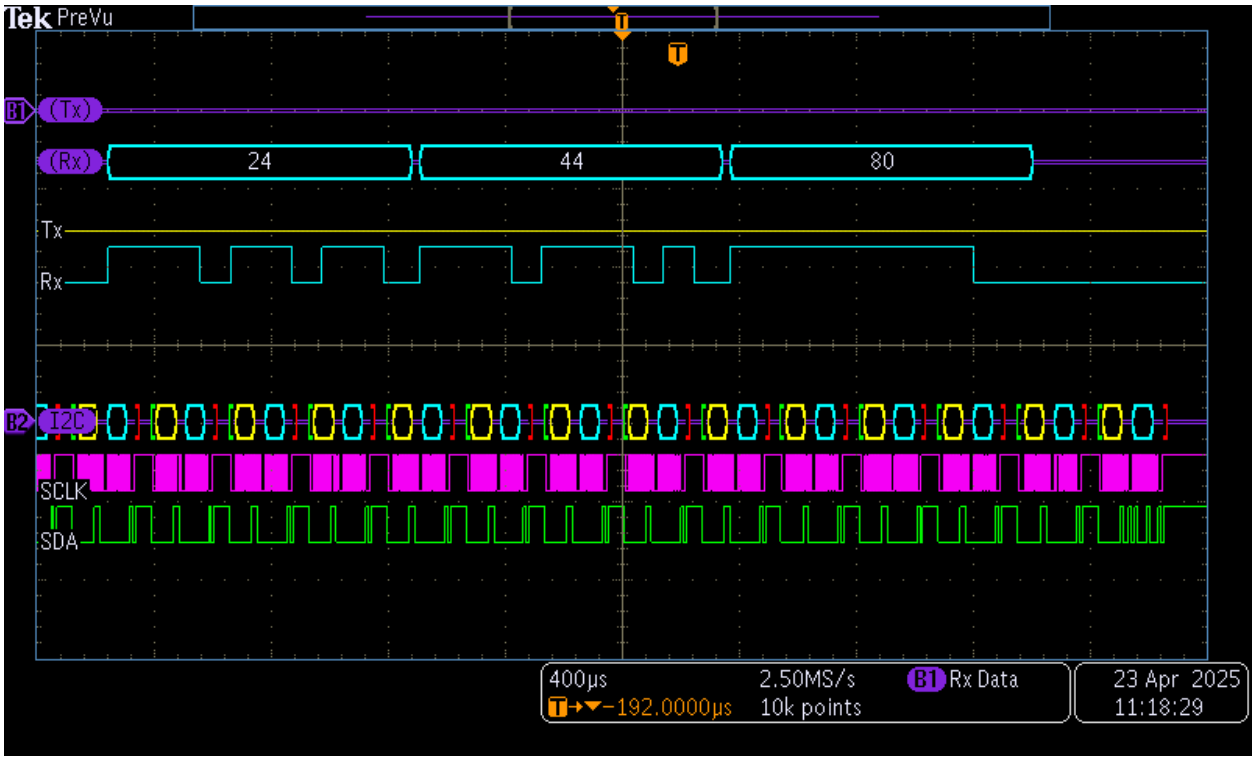


Figure 7.2.3: UART Target Disable

Figure 7.2.3 above is an oscilloscope capture of the UART Target Disable command. This command is made up of three bytes, a handshake byte of an ASCII $ (0x24) followed by an ASCII D (0x44) and a TargetSlots byte. The TargetSlots byte is shown in Table 7.2.3. Each bit in TargetSlots is associated with a target slot on the master. A logic high bit will request that the master disable the specified target. When the target disable command is received, the master ORs

the TargetSlots byte with the DISABLE_TARGET register to request the main loop disables the specified target slots. Any logic low bits will leave the target slot in its current state. As shown in figure 7.2.3, following the received UART disable command, the master stops polling and disables the target, causing the I2C communication lines to go to idle. See section 7.3 for the I2C communications.

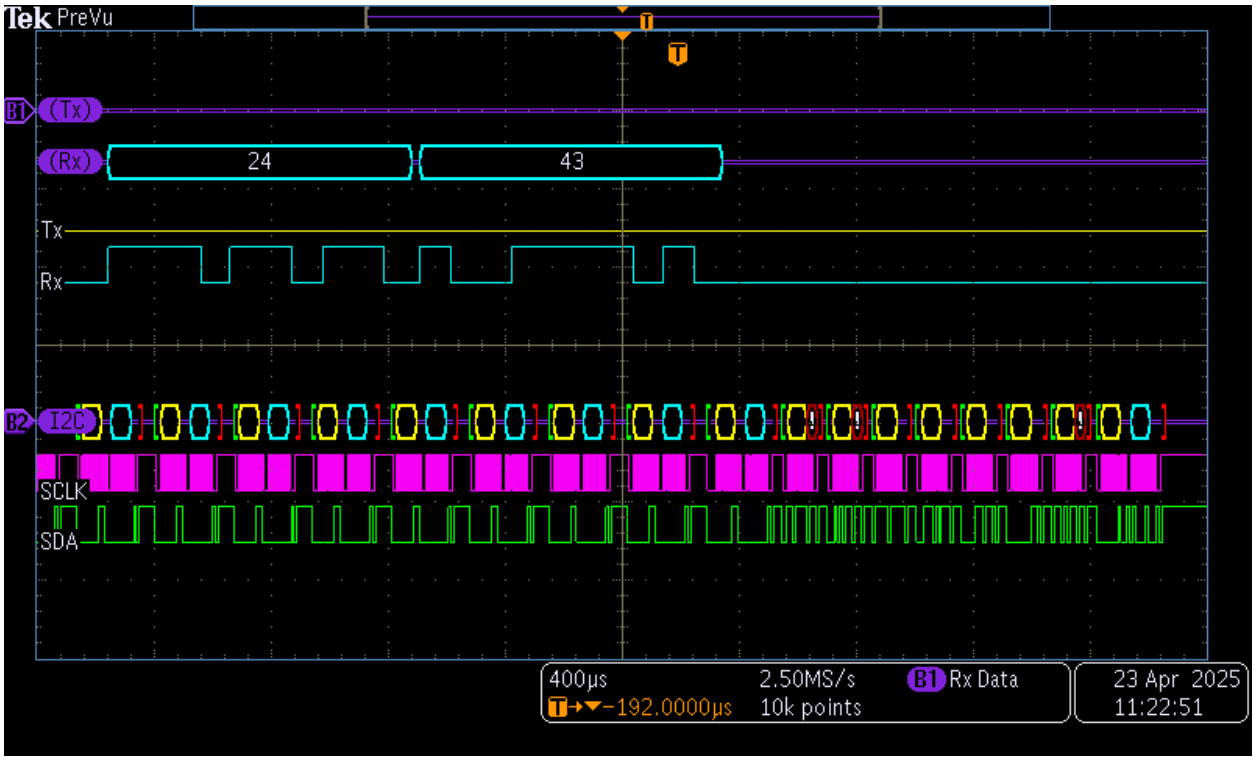


Figure 7.2.4: UART Disable All Targets

Targets may also be disabled using the Disable All/Clear command. This command consists of two bytes, the handshake byte of an ASCII $ (0x24) followed by an ASCII C (0x43). When this command is received by the master, polling is ended for all targets and each is individually disabled via I2C. This command is intended to serve as a quick way to disable all targets at the end of a game timer.

7.2.6: Target Address Change

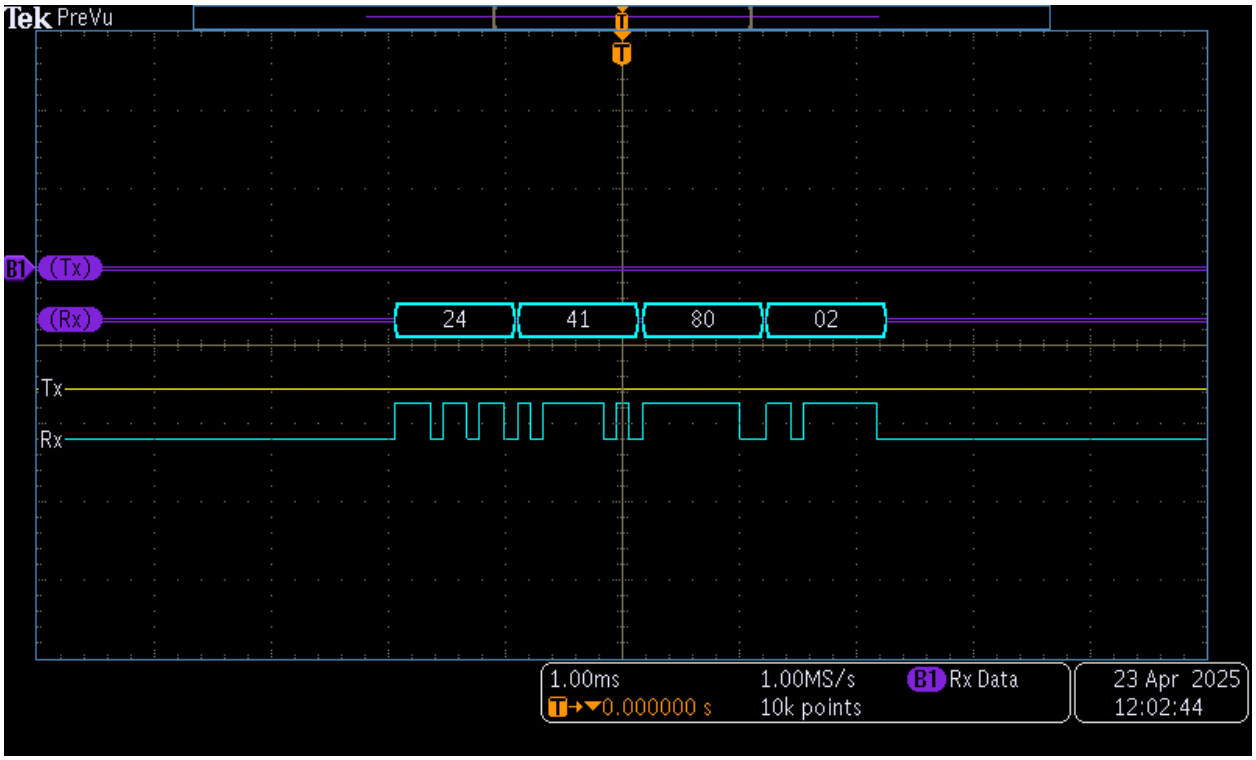


Figure 7.2.4: UART Target Address Change

Figure 7.2.4 above is an oscilloscope capture of the UART Target address change command. The address change command is 4 bytes, the first being the handshake byte of an ASCII $ (0x24). Following the handshake, the command byte of an ASCII A (0x41), the ChangeSlot, and the I2C address are sent. Both the ChangeSlot and I2C address bytes require specific formatting for the master to interpret properly. Table 7.2.4 below is a bitmap of both the ChangeSlot and I2C address UART bytes.

Table 7.2.4: UART Target Address Change Bitmap

Byte	B7	B6	B5	B4	B3	B2	B1	B0
ChangeSlot	TGT_1	TGT_2	TGT_3	TGT_4	TGT_5	TGT_6	TGT_7	TGT_8
I2C Address	7-bit I2C address							0

Table 7.2.4 shows both the ChangeSlot and I2C Address bytes of the UART target address change command. The ChangeSlot byte is similar to the TargetSlots byte used in both the enable and disable commands. A key aspect of the ChangeSlot byte is that it must be sent with only one bit high and all other bits low. The change address command cannot be used to set the address of more than one target slot at a time. If two bits are high simultaneously, one target will receive the new address while the other will not. This serves to prevent more than one target slot from communicating with the same target. The I2C address byte has special considerations as shown in Table 7.2.4. The master uses 7-bit I2C addresses to communicate with the slave boards. The I2C address byte sent via UART must contain a 7-bit address within the 8-bit byte with the unused bit (b0) as a logic low.

7.2.7: Player Score

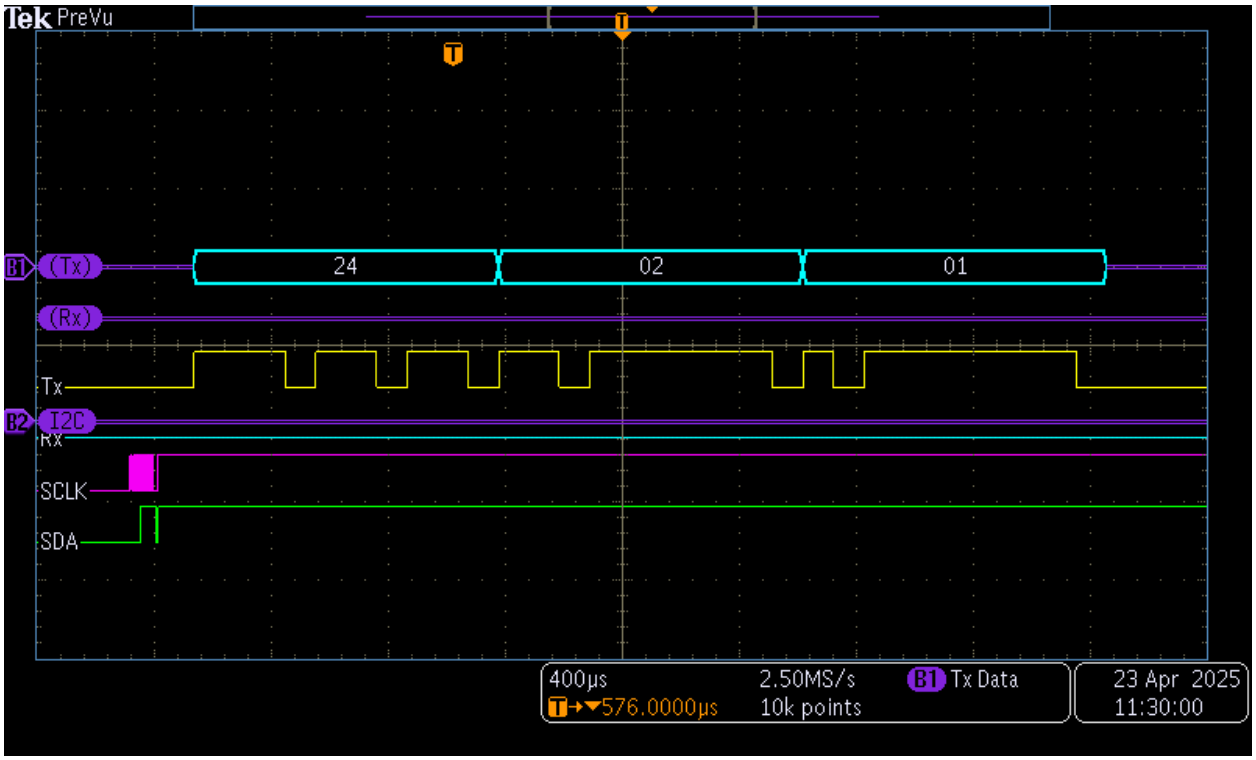


Figure 7.2.5: UART Player One Score

Figure 7.2.5 is an oscilloscope capture of the master board communicating a hit by player one on slave 1 back to the VB program. When a slave board reports to the master that a player has scored, the master transmits the target hit via UART back to the VB program. This transmission consists of three bytes, the handshake byte of an ASCII $ (0x24), the I2C address of the hit target, and the player number. The I2C address byte is formatted in the same way as with the address change command with the 7-bit address and B0 being a logic low. The player number byte is an 8-bit integer number representing the number of each player. A score by player one will be a 0x01 and a score by player two will be a 0x02

7.3: I2C

7.3.1: I2C Overview

Communications between the target system master and slave boards utilizes the I2C communications protocol. The I2C protocol utilizes a shared clock (SCL) and a shared data line (SDA) for all devices. These two lines originate from the master board and are connected to all target slave boards. I2C uses an addressing system to direct traffic between the master and a specific slave and back. The laser arcade uses an I2C baud rate of 100Kbps with 7-bit addressing, allowing for up to 127 different target slaves. The first byte of any read or write on the I2C bus is the address byte with the data byte following. The address byte contains 8 bits of information including the 7-bit address and a read/write indicator bit. Bit 0 of the address byte determines if a specific action is a read or a write. A logic low represents a write operation and a logic high represents a read operation.

7.3.2: I2C Write

The master writes to I2C when transmitting data to one of the slaves. For the laser arcade, the data written to the slave is interpreted as different commands for the operation of a target. There are two different commands for the slave, an enable command and a disable command. Figure 7.3.1 below is an oscilloscope capture showing the I2C enable command.

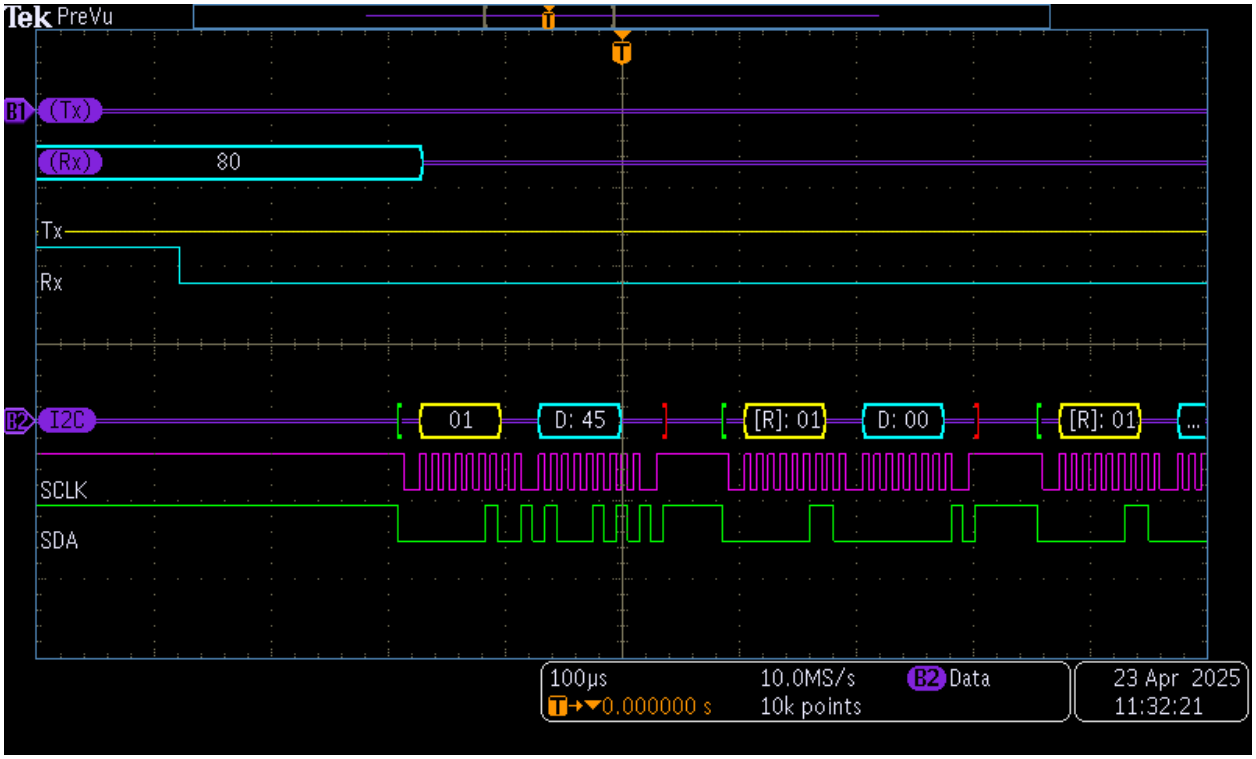


Figure 7.3.1: I2C Enable Command

Figure 7.3.1 shows part of the last byte of the UART enable command before the I2C enable command. The address is sent first, in this case writing to address 1. The data is sent second with the enable command being an ASCII E (0x45). Following the enable command, the slave will begin accepting player hits and will turn on the green LEDs indicating to the player that the target is enabled. Immediately after the enable command is sent, the master will begin to poll the target with an I2C read.

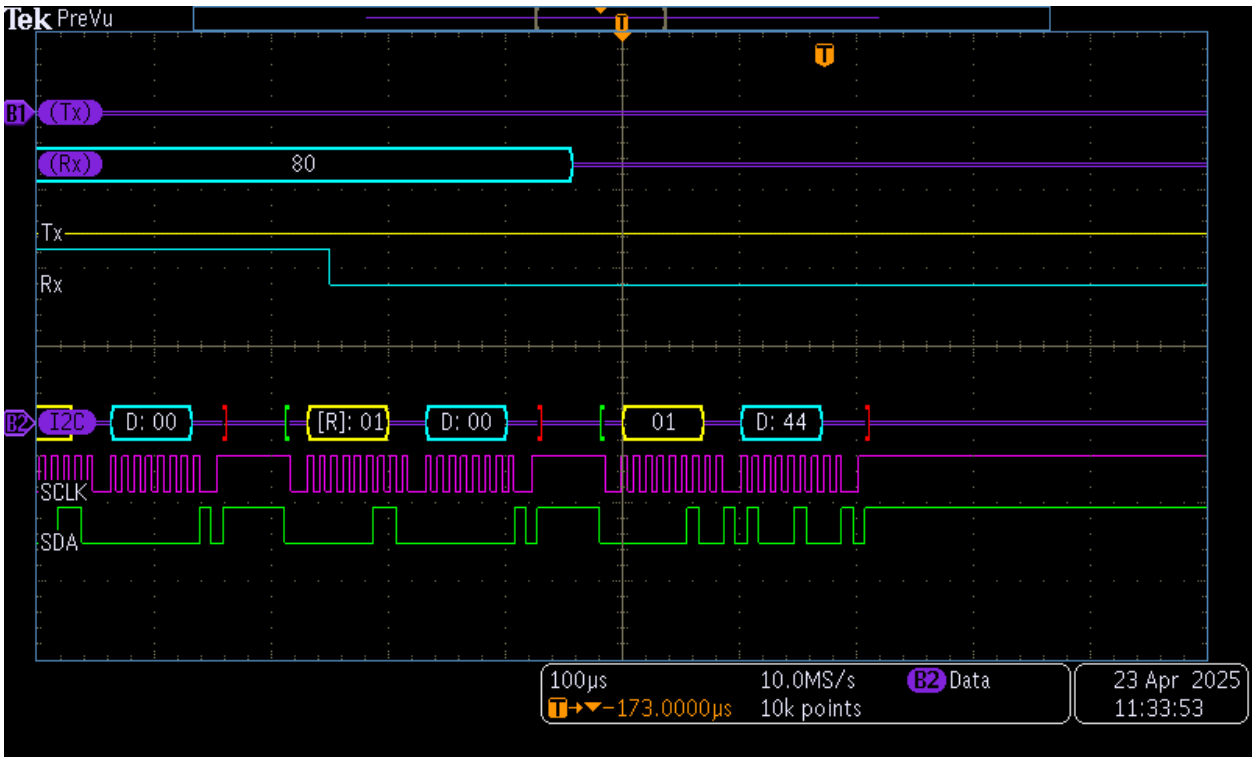


Figure 7.3.2: I2C Disable Command

Figure 7.3.2 shows the I2C disable command following the last bit of the UART disable command. Until the target is disabled, the master will continue to poll the slave for hits to the target by reading from the address as shown in the capture. Once the UART disable command has been received, the master will send the I2C disable command to the corresponding address. The I2C disable command is an ASCII D (0x44). When the master writes the disable command to the slave, the slave immediately disables itself and turns off all LEDs. Following the disable command, the master will no longer poll the slave for updates.

7.3.3: I2C Read

When any number of targets are enabled, the master will cycle through the target slots reading the status of the slave via I2C. An I2C read is characterized by the master sending 8 bits of information including the 7-bit address and the 0 bit as a logic high. This bit designates to the slave the master is reading data and is expecting a response. Following the address, the slave responds with its status byte. Figure 7.3.3 below shows the read both when the status has not changed and when the slave reports a hit by player one.

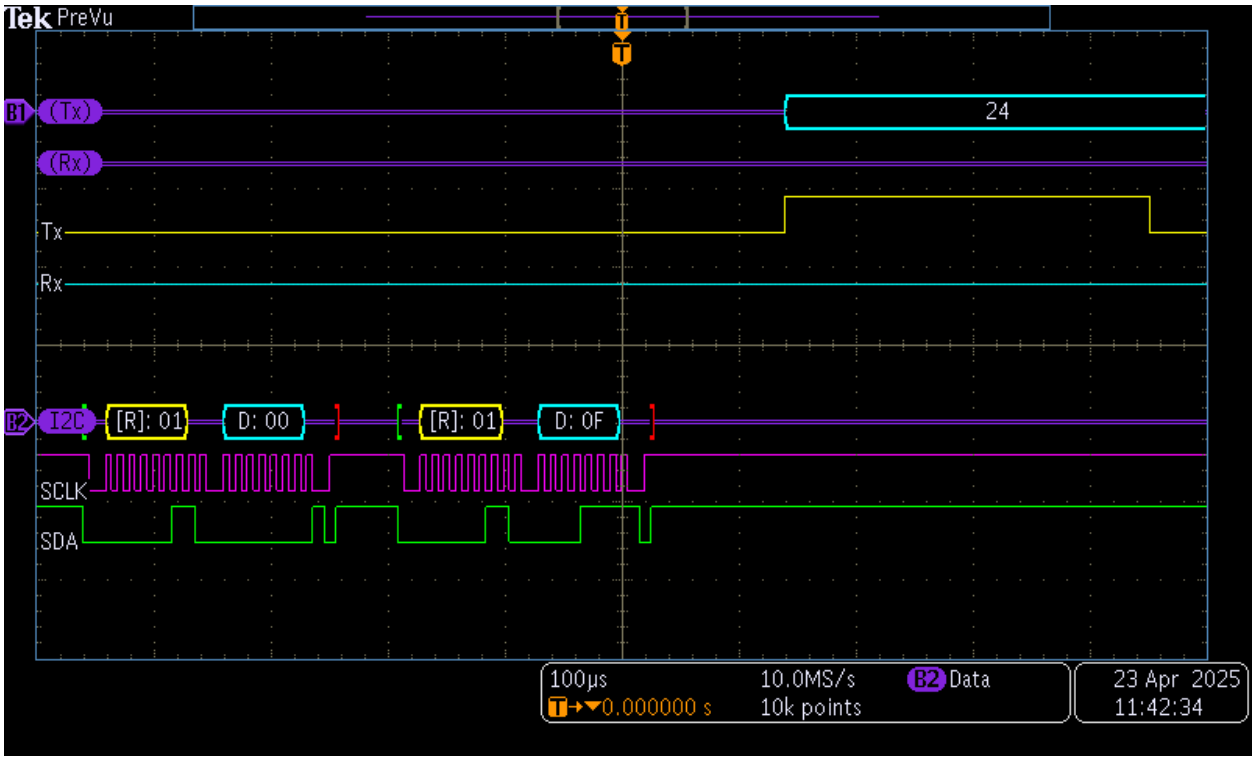


Figure 7.3.3: I2C Read Player One Score

Figure 7.3.3 is an oscilloscope capture showing the master polling the slave at address 1 until a hit is recorded. After the hit, the master begins to transmit the hit back to the VB program over UART. The data byte during an I2C read is the status of the target slave. When the target has not been hit, the status will read 0x00. When player one has hit the target, the status will read

0x0F. When player two has hit the target, the status will read 0xF0. Figure 7.3.3 shows a hit by player one. Figure 7.3.4 below shows the same capture when player two has hit the target.

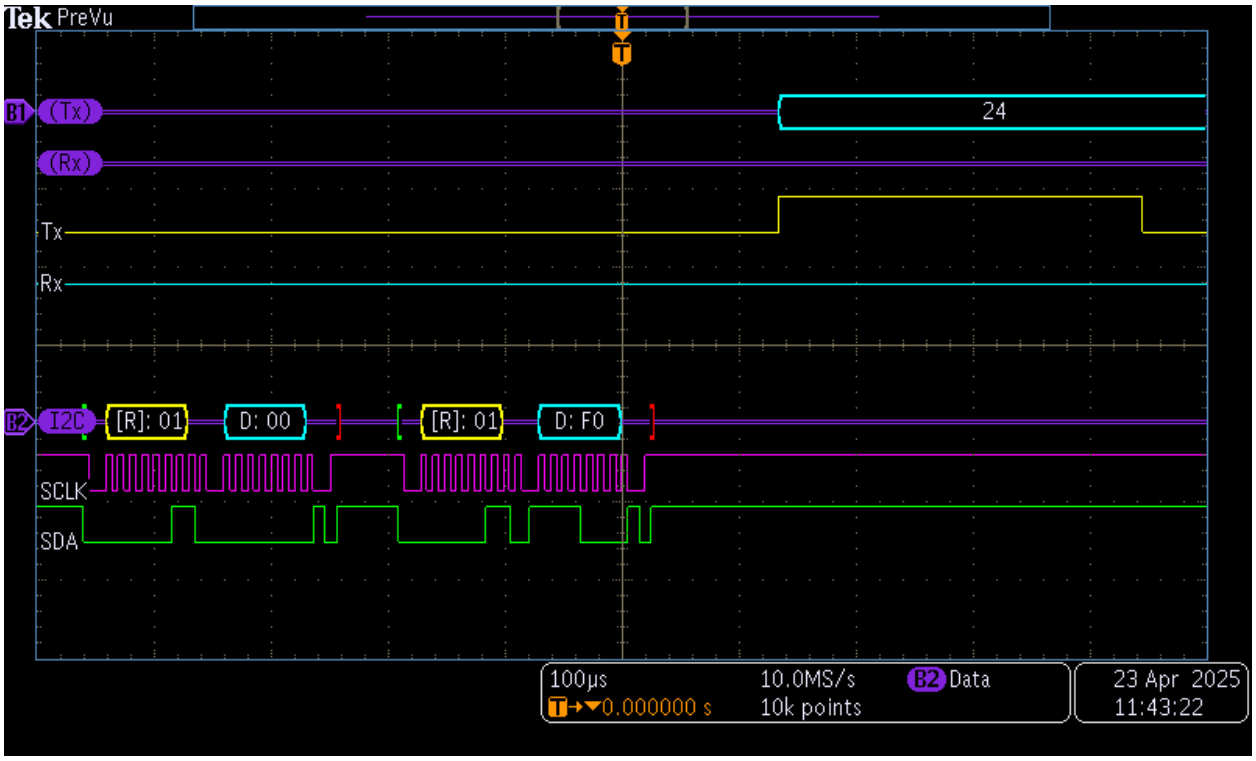


Figure 7.3.4: I2C Read Player Two Score

7.4: Visual Basic

7.4.1: Visual Basic Overview

The laser arcade target system utilizes a visual basic program to manage the player scores and the game countdown clock. The program consists of both a graphical user interface and an object oriented back end. Two object classes were created to manage the system, an ArcadeTarget class and a LaserArcade class. Combined, these two classes manage the UART communications and target management for the laser arcade. The LaserArcade class raises events for player scores and device connection failures, allowing it to be transferred between user interfaces and games. The user interface is a very basic player vs player game that tracks earned points and keeps a countdown. Game configuration can be temporarily changed or saved to file and reloaded as needed.

7.4.2: ArcadeTarget Class

The ArcadeTarget Class is used to track the characteristics of each master target slot. When a new instance of the class is created, the target slot is set and stored as a byte in the correct format for UART to transmit to the master board. Each instance of the class stores its I2C address as a byte, if the target is enabled as boolean, and if VB has previously set the I2C address as a boolean. At initialization, each instance will default to disabled with the I2C address not having previously been set. The class contains three different functions, EnableTarget, DisableTarget, and ChangeAddress. Each of these functions return a byte array of the full UART transmission to enable, disable, or change the I2C address of the specified target slot. The returned array can be directly transmitted through the serial port, as index 0 is always the first byte to transmit.

7.4.3: LaserArcade Class

The LaserArcade Class is used for complete control of the target system. The class contains all UART functionality as well as the ability to enable a random target. When an instance of the class is created, default values for the configuration are set and 8 instances of the ArcadeTarget class are loaded into an array.

The class has several public parameters to be used by the user interface to configure the class. The COMPort property accepts the string name of a windows communications port and is used to set the port used to connect to the master board. The Connected and DeviceVerified properties are read only boolean values. If the COM port is currently open, Connected will return true. If the COM port is open and the master successfully responds to the UART device verification command, DeviceVerified will return true. The NumberOfTargets property sets the number of targets available to the master with the value between 1 and 127. The LaserArcade class expects the I2C address of connected targets to begin at 1 and count up. The class has the ability to optionally disable targets after a random amount of time. The timed disable can be turned on or off using the TimedDisable property. The time range used by the timed disable can be set in milliseconds using the TimedDisableMinimum and TimedDisableMaximum properties.

The LaserArcade class has a variety of public subroutines. The StartConnection subroutine opens the serial connection and starts the device verification process. In addition, a connection timeout is started to close the connection if the target master does not respond to the device verification command. An EndConnection subroutine is available for closing the connection. For interfacing with targets, the class has three different public subroutines. The EnableTarget subroutine allows for enabling a single target or all targets. The DisableTarget subroutine will disable a single target or all targets. Both subroutines require an integer input

from 0 to 8. An input of 0 will enable or disable all target slots. An input between 1 and 8 will enable or disable a single target. All other inputs are ignored. The EnableRandomTargets subroutine will load a random I2C address between 1 and the number of targets into the first available target slot and enable it.

The LaserArcade class also has the ability to raise four different events to be handled by the user interface. The DeviceVerificationFailed event is raised when the target master fails to respond to the UART device verification command. The UnexpectedDisconnect event is raised when previously connected COM ports unexpectedly close to allow the user interface to handle the disconnect. The PlayerOneScore and PlayerTwoScore events occur when the corresponding player scores. The score events also send the I2C address of the target hit by the player to allow for alternate scoring methods to be handled by the user interface.

7.4.4: User Interface

The user interface consists of three different forms, a control form, a configuration form, and an about form. The default form is the control form, allowing the user to start and stop a game. The configuration form allows for changing, saving, or loading different game settings. The about form shown in figure 7.4.1 below contains a link to the github for this project.

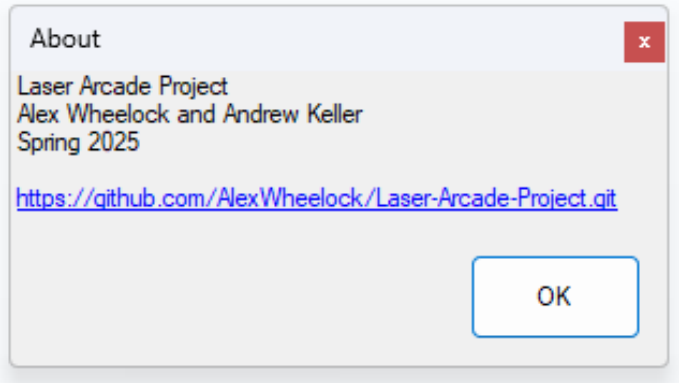


Figure 7.4.1: Laser Arcade About Form

Figure 7.4.2: Laser Arcade Control Form

Figure 7.4.2 above is the laser arcade control form, allowing the user to start and stop a game. In addition, the control form displays the game timer countdown as well as the current player scores. A status strip at the bottom of the page displays the current configuration, including the file path of saved configurations. The status strip also displays the current communications port status and allows the user to change the selected COM port.

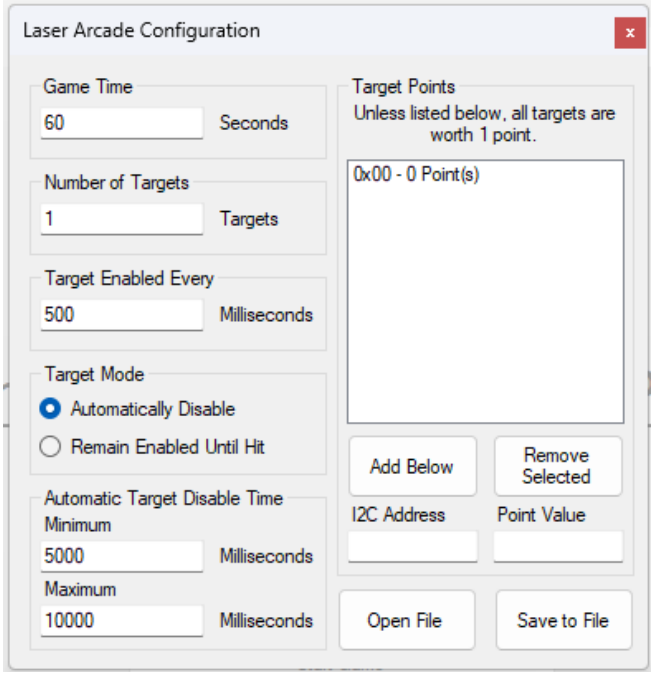


Figure 7.4.3: Laser Arcade Configuration Form

Figure 7.4.3 above is the laser arcade configuration form, allowing the user to change various aspects of the game. The number of targets, target mode, and target disable times are directly passed to the LaserArcade class. The game time setting allows for changing the length of a single game. During a game, the LaserArcade class EnableRandomTargets subroutine is called on a timer with a configurable interval using an input on the configuration form. In addition, the point value of a target can be changed from 1 to any other value by entering the hex I2C address, including the 8th bit as a low, and the new point value. A point override can be removed by selecting it from the list and clicking the "Remove Selected" button. The configuration form also has the ability to save to and open from configuration files using the "Open File" and "Save to File" buttons.

7.5: Target System Power

The target system will be powered by a single 5V 12A DC power supply. The supply will be connected to the master and be distributed through the I2C headers (J3, J4, and J5) on the master to the slave boards. The power supply being used is the LPV-100-5 from Mean Well. Due to the 5V power also being tied to the USB connector (J6) on the master, power can also be pulled from the connected computer. The full target system current draw is above what USB is capable of. To disconnect the USB power from the rest of the system, L1 will be removed and left as an open.

7.6: Target System Troubleshooting

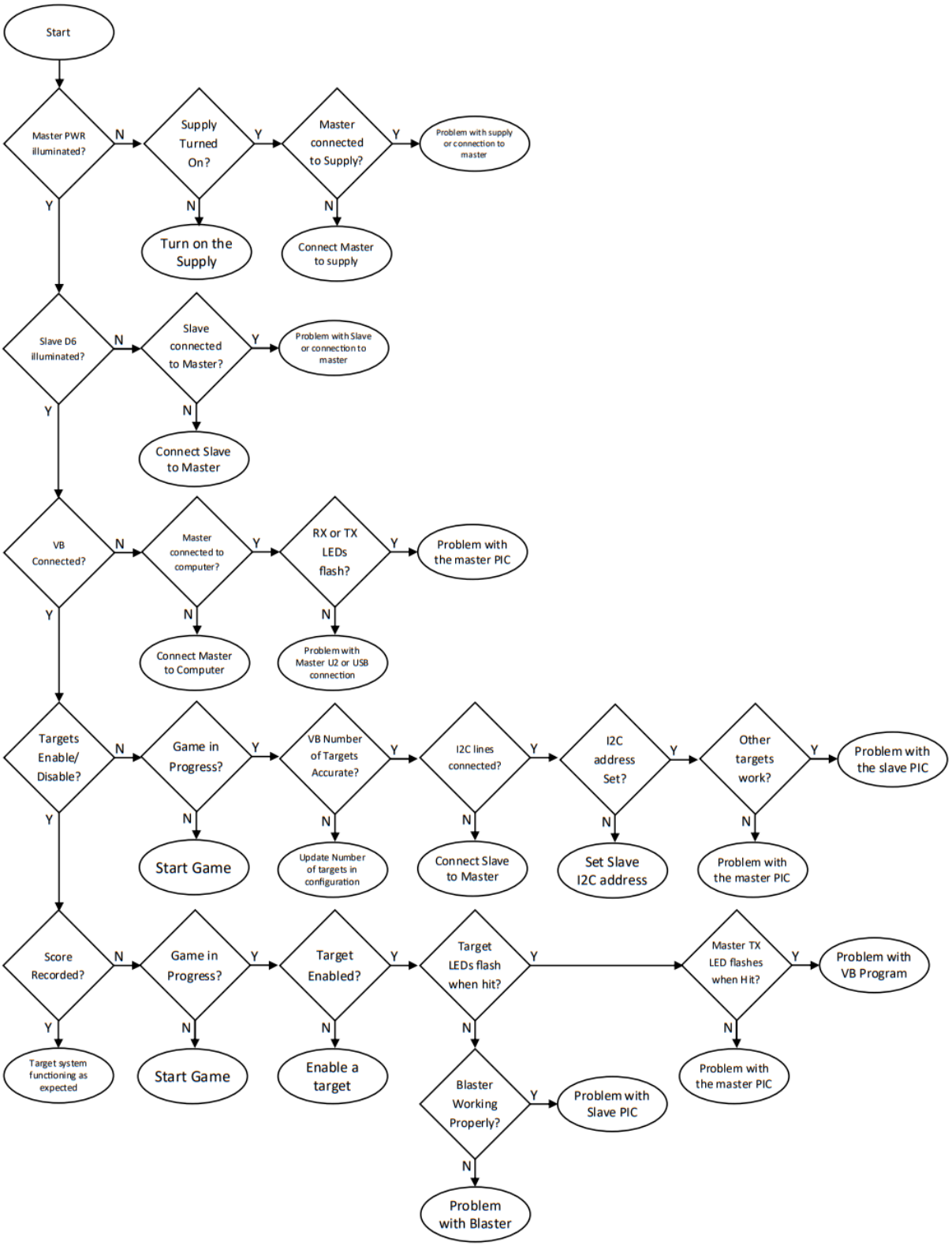


Figure 7.6.1: Target System Troubleshooting Flowchart

To troubleshoot the target system, follow the steps below, or follow Figure 7.6.1.

1. Is the master power LED (D1) illuminated?
    a. If yes, move to step 2.
    b. If not, move to step 6.
2. Is the slave power LED (D6) illuminated?
    a. If yes, move to step 3.
    b. If not, move to step 8.
3. Is the Visual Basic program successfully connected?
    a. If yes, move to step 4.
    b. If not, move to step 9.
4. Do targets enable and disable?
    a. If yes, move to step 5.
    b. If not, move to step 11.
5. Is a score reported in VB?
    a. If yes, the target system is functioning as expected. Skip all remaining steps.
    b. If not, move to step 16.
6. Is the power supply turned on?
    a. If yes, move to step 7.
    b. If not, turn on the power supply.
7. Is the master connected to the power supply?
    a. If yes, problem with supply or connection to master.
    b. If not, connect the master to the power supply

8. Is the slave connected to the master?
    a. If yes, problem with the slave or connection to master.
    b. If not, connect the slave to the master.
9. Is the master connected to the computer?
    a. If yes, move to step 10.
    b. If not, connect the master to the computer.
10. Do the master RX or TX LEDs flash?
    a. If yes, problem with the master PIC.
    b. If not, problem with master U2 or USB connection between master and computer.
11. Is a game in progress?
    a. If yes, move to step 12.
    b. If not, start a game in VB.
12. Is the number of targets in the VB configuration accurate?
    a. If yes, move to step 13.
    b. If not, update the number of targets in the VB configuration.
13. Are the I2C lines connected between the master and slave?
    a. If yes, move to step 14.
    b. If not, connect the slave to the master.
14. Is the slave I2C address correctly set using SW1 on the slave?
    a. If yes, move to step 15.
    b. If not, set the slave I2C address using SW1 and power cycle the target system.

15. Do other targets enable or disable?
    a. If yes, problem with the slave PIC.
    b. If not, problem with the master PIC.
16. Is a game in progress?
    a. If yes, move to step 17.
    b. If not, start a game in VB.
17. Is the target enabled with the green LEDs illuminated?
    a. If yes, move to step 18.
    b. If not, enable the target.
18. Do the target LEDs flash when hit?
    a. If yes, move to step 20.
    b. If not, move to step 19.
19. Is the blaster working properly?
    a. If yes, problem with the slave PIC.
    b. If not, problem with the blaster.
20. Does the master TX LED flash when hit?
    a. If yes, problem with the VB program.
    b. If not, problem with the master PIC

8: Engineering Log

Table 8.1: Alex Wheelock's Project Engineering Log

Date	Category	Time Started	Time Stopped	Duration	Description
3/10/25	Testing	11:00 AM	3:00 PM	4:00	Testing an audio board for the blaster, and testing solenoids
3/11/25	Programming	11:10 AM	3:15 PM	4:05	Start writing code for the target master
3/12/25	Programming	9:15 AM	3:00 PM	5:45	Writing code for the target slave, and testing the I2C communication
3/13/25	Programming	12:00 PM	3:00 PM	3:00	Working on I2C communication, troubleshooting code
3/14/25	Programming	11:00 AM	3:00 PM	4:00	Got I2C communication working between master/slave for writing to the slave
3/17/25	Programming	1:00 PM	3:30 PM	2:30	Working on adding reading to the I2C, so that it can determine hits
3/18/25	Programming	11:10 AM	3:00 PM	3:50	Finishing up the I2C communication (mostly) between master and slave
3/19/25	Programming	11:25 AM	3:00 PM	3:35	Testing the new audio boards for the blaster, setting it up and creating audio files
3/20/25	Programming	2:00 PM	3:25 PM	1:25	Finish setting up the audio board
3/21/25	Design	11:50 AM	2:50 PM	3:00	Test boost regulator and caps to supply current to the solenoid
4/1/25	Design	12:30 PM	3:15 PM	2:45	Updating schematics, testing RGB LEDs, testing switches, updating blaster code
4/2/25	Testing	11:00 AM	3:00 PM	4:00	Testing different materials to use as targets
4/3/25	Design	1:15 PM	3:15 PM	2:00	Working on cleaning up and updating all KiCad schematics
4/4/25	Organization	11:30 AM	3:15 PM	3:45	Working on cleaning up and sharing project files to send to Andrew, working on KiCad
4/7/25	Programming	1:00 PM	3:00 PM	2:00	Working on the I2C between master and slave
4/10/25	Design	1:00 PM	4:15 PM	3:15	Designing and testing the boost supply for the blaster with the new IC
4/11/25	Testing	9:00 AM	4:00 PM	7:00	Finishing up the blaster power circuit and testing it

4/14/25	Design	9:15 AM	3:25 PM	6:10	Finalized schematics & blaster footprints, designed a box for the speaker to mount into
4/14/25	Design	7:30 PM	8:00 PM	0:30	Starting the blaster PCB
4/15/25	Design	11:36 AM	2:30 PM	2:54	Worked on the blaster PCB, and testing the speaker box
4/16/25	Design	11:00 AM	3:00 PM	4:00	Working on the blaster PCB
4/16/25	Documentation	7:40 PM	10:30 PM	2:50	Starting report
4/17/25	Design	11:00 AM	2:00 PM	3:00	Working on fixing the blaster schematic for the separated boards fixing the nets, working on PCB
4/18/25	Design	12:15 AM	12:45	0:30	Recovering lost work for schematic
4/18/25	Design	11:00 AM	2:00 PM	3:00	Working on PCB, nearly complete
4/21/25	Design	9:15 AM	3:20 PM	6:05	Finished blaster PCB, documented the blaster
4/22/25	Design	11:00 AM	3:00 PM	4:00	Working on 3D Models for blaster, got the speaker box revision and barrel done
4/23/25	Design	11:00 AM	3:00 PM	4:00	Figuring out and desiging the spring and solenoid mount for the blaster
4/24/25	Design	11:00 AM	12:00 PM	1:00	Testing printed parts (barrel & speaker box), making grinding down barrel mounts
4/24/25	Design	1:00 PM	4:00 PM	3:00	Testing solenoid mount, fixing it and the barrel, designing rear hook, and rear bumper
4/25/25	Documentation	12:30 PM	5:40 PM	5:10	Taking blaster waveform & mechanical drawing captures and working on the report
4/28/25	Design	1:00 PM	4:30 PM	3:30	Working on modifying both nerf guns so that they are ready to be assembled when PCBs arrive
4/29/25	Documentation	11:00 AM	3:00 PM	4:00	Making some final modifications to the blaster, working on mechanical drawing images, cleaning up code
5/1/25	Soldering	11:00 AM	3:00 PM	4:00	Soldering up the first PCB and testing it, didn't work right
5/2/25	Troubleshooting	10:00 AM	3:30 PM	5:30	Got the PCB working, fixed some bugs in the code
5/4/25	Documentation	6:00 PM	8:00 PM	2:00	Working on flowcharts and writeup for the blaster, created the presentation file
5/5/25	Construction	12:00 PM	3:00 PM	3:00	Building blaster, putting together wire harnesses, making connectors, and installing all components

5/5/25	Construction	4:00 PM	5:40 PM	1:40	Building blaster, putting together wire harnesses, making connectors, and installing all components
5/5/25	Documentation	7:15 PM	2:00 AM	6:45	Writing up blaster and working on the presentation
5/6/25	Construction	9:30 AM	3:30 PM	6:00	Finishing up the presentation, putting the blaster together, taking final measurements and pictures for report
5/6/25	Documentation	7:15 PM	12:30 AM	5:15	Practicing presentation, and working on report
5/7/25	Documentation	1:00 PM	5:50 PM	4:50	Presenting the project and working on the report
5/7/25	Documentation	9:20 PM	3:20 PM	6:00	Working on finishing the report
5/8/25	Documentation	9:00 AM	11:00 AM	2:00	Making final changes to the report and printing it off to turn in
				Total Hours Worked	**147:55**

Table 8.2: Andrew Keller's Project Engineering Log

Date	Category	Time Started	Time Stopped	Duration	Description
4/10/25	Design	9:30 AM	12:15 PM	2.75	Developing UART Command structure
4/10/25	Programming	1:10 PM	3:05 PM	1.92	Visual Basic setup, serial port connection, sending device verification command
4/11/25	Programming	8:45 AM	3:00 PM	6.25	Master UART Assembly code
4/14/25	Programming	9:00 AM	12:30 PM	3.5	Moving serial communications from form to LaserArcade class, score events, randomly enable targets
4/14/25	Programming	1:15 PM	3:00 PM	1.75	Disable targets after timer, basic gameplay
4/15/25	Programming	9:15 AM	4:15 PM	7	Visual Basic game configuration menu, main GUI, tracking player scores
4/16/25	Design	9:00 AM	3:45 PM	6.75	Target Slave PCB design
4/17/25	Design	9:00 AM	12:15 PM	3.25	Target Master PCB design
4/18/25	Design	9:00 AM	12:00 PM	3	PCB Design verification
4/18/25	Construction	12:00 PM	3:00 PM	3	Building slave circuit on breadboard
4/21/25	Programming	9:00 AM	3:00 PM	6	I2C between master and slave
4/22/25	Programming	9:00 AM	3:45 PM	6.75	I2C protocol, fixing VB errors
4/23/25	Documentation	8:30 AM	3:00 PM	6.5	Master code comments, slave code comments, VB code comments
4/24/25	Design	9:00 AM	12:15 PM	3.25	3D Modeling target reflectors
4/24/25	Documentation	1:00 PM	3:30 PM	2.5	Working on project report
4/25/25	Documentation	9:00 AM	3:00 PM	6	Master and Slave updated flowcharts
4/28/25	Documentation	8:45 AM	12:00 PM	3.25	Working on Project Report (UART)

4/28/25	Documentation	12:45 PM	3:00 PM	2.25	Working on Project Report (I2C)
4/29/25	Documentation	9:00 AM	12:00 PM	3	Working on Project Report (Code in Appendix, VB)
4/29/25	Documentation	1:00 PM	3:00 PM	2	Working on Project Report (VB)
4/30/25	Soldering	9:00 AM	4:00 PM	7	Soldering master and slave boards
5/1/25	Soldering	8:45 AM	12:00 PM	3.25	Soldering slave boards
5/1/25	Soldering	1:00 PM	3:00 PM	2	Soldering Slave Boards
5/2/25	Soldering	9:00 AM	12:00 PM	3	Soldering Slave Boards
5/2/25	Construction	1:00 PM	3:30 PM	2.5	Target System construction
5/5/25	Construction	9:00 AM	4:00 PM	7	Target system construction
5/6/25	Documentation	8:30 AM	12:30 PM	4	Working on project report and presentation
5/6/25	Construction	12:30 PM	2:30 PM	2	Target System Construction
5/6/25	Documentation	2:30 PM	5:00 PM	2.5	Working on project report and presentation
5/7/25	Documentation	8:00 AM	4:00 PM	8	Final presentation, working on project report
5/8/25	Documentation	9:00 AM	11:00 AM	2	Finishing project report, printing it out
				Total Hours Worked	**124.92**

9: Cost Analysis

Table 9.1: Blaster Cost Analysis

Component Type	Value	Qty	Part Number	Cost/Part (Based on 100 Qty)	Cost/PCB
Power Supply	+12V	1	N/A	$27.99	$27.99
Connector	XT30	1	N/A	$0.48	$0.48
Capacitor	220uF	2	710-860020373010	$0.12	$0.24
Capacitor	0.1uF	2	187-CL21B104KACNFNC	$0.01	$0.02
Capacitor	1uF	3	581-KAF21KR71E105KL	$0.13	$0.39
Capacitor	50pF	1	791-0805N500J250CT	$0.03	$0.03
LED	Green	2	859-LTST-C171KGKT	$0.07	$0.14
Diode	1N4148	3	637-1N4148W	$0.04	$0.12
Header	1x5	3	M20-9990546	$0.14	$0.42
Header	1x11 (Male)	1	649-67996-112HLF	$0.72	$0.72
Header	1x11 (Female)	1	737-RS1-11-G	$0.42	$0.42
Header	1x14 (Female)	2	200-CES11402TS	$1.31	$2.62
Header	1x16 (Male)	1	737-PH1-16-UA	$0.15	$0.15
Header	1x16 (Female)	1	710-61301611821	$0.52	$0.52
Locking Header	1x2	5	571-6404562	$0.09	$0.45
Locking Header	1x6	1	571-6404566	$0.23	$0.23
MOSFET	BUK9832-55A	3	771-BUK9832-55A/CUX	$0.43	$1.29
Resistor	8.2kΩ	1	603-RE0805DRE078K2L	$0.03	$0.03
Resistor	4.7kΩ	1	594-MCU08050C4701FP5	$0.06	$0.06
Resistor	10kΩ	1	791-RMC1/10K1002FTP	$0.01	$0.01
Resistor	1kΩ	6	667-ERA-6APB102V	$0.23	$1.38
Resistor	100kΩ	3	754-RR1220P-104D	$0.03	$0.09
Resistor	0Ω	3	791-RMC1/10JPTP	$0.01	$0.03
Laser Diode	5V	1	N/A	$12.69	$12.69
Solenoid	12V/2A, 25Nm	1	JF-1039B	$12.43	$12.43
Switch	SPST	1	N/A	$1.40	$1.40

Switch	Momentary	1	N/A	$0.05	$0.05
Switch	SPST	1	113-LSSA12VB	$0.91	$0.91
Switch	Rotary	1	N/A	$8.15	$8.15
Regulator	TLV1117-50	1	595-TLV1117-50CDCYR	$0.19	$0.19
Microcontroller	PIC16F1788	1	579-PIC16F1788T-I/SS	$2.15	$2.15
Audio Board	AdaFruit FX Board	1	N/A	$14.95	$14.95
PCB	Blaster	1	N/A	$2.20	$2.20
Nerf Gun	Rival Kronos	1	N/A	$20.00	$20.00
				Total:	**$112.95**

Table 9.2: Target Master Cost Analysis

Component Type	Value	Qty	Part Number	Cost/Part (Based on 100 Qty)	Cost/PCB
Resistor	10kΩ	2	603-AC0805JR-0710KL	$0.010	$0.020
Capacitor	220uF	3	4191-EY1C221MP26311ERUCT-ND	$0.056	$0.167
Header	1x1 Header	4	538-22-28-4012	$0.047	$0.188
Microcontroller	PIC16F1788-IP	1	579-PIC16F1788-I/SS	$2.140	$2.140
Resistor	3.3kΩ	2	791-RMC1/10-332JTP	$0.007	$0.014
Capacitor	0.1uF	4	1276-1003-1-ND	$0.0088	$0.035
LED	TX LED	1	160-1423-1-ND	$0.0688	$0.069
Locking Header	1x4 Header	3	538-87891-0408	$0.166	$0.498
LED	RX LED	1	160-1423-1-ND	$0.0688	$0.069
Capacitor	10nF	1	1276-1015-1-ND	$0.0112	$0.011
Diode	PWR LED	1	160-1423-1-ND	$0.0688	$0.069
Connector	+5V	1	B0DM19KN2Y	$0.969	$0.969

Inductor	2uH	1	4816-FCI2012F-2R2MCT-ND	$0.0251	$0.025
Connector	USB_B_Mini	1	656-DX4R005JJ2R1800	$0.521	$0.521
USB to UART	FT232RL	1	895-FT232RNL-REEL	$3.950	$3.950
Header	1x6 Header	1	538-22-28-5061	$0.175	$0.175
Capacitor	1uF	2	1276-1066-1-ND	$0.019	$0.038
Capacitor	4.7uF	1	1276-1259-1-ND	$0.0282	$0.028
Capacitor	47pF	2	791-0805N470K250CT	$0.016	$0.032
Resistor	220Ω	2	791-RMC1/10-221JTP10	$0.007	$0.014
Resistor	4.7kΩ	1	652-CR0805FX-4701ELF	$0.010	$0.010
Diode	1N4148	1	5399-1N4148WSCT-ND	$0.0187	$0.019
PCB	PCB	1	N/A	$2.200	$2.200
Power Supply	Livewell LPV-100-5	1	709-LPV100-5	$24.60	$24.60
				Total:	**$35.86**

Table 9.3: Target Slave Cost Analysis

Component Type	Value	Qty	Part Number	Cost/Part (Based on 100 Qty)	Cost/PCB
Header	1x1 Header	4	538-22-28-4012	$0.047	$0.188
Capacitor	1uF	4	1276-1066-1-ND	$0.019	$0.077
Resistor	4.7kΩ	1	652-CR0805FX-4701ELF	$0.010	$0.010
Capacitor	22uF	1	1276-1202-1-ND	$0.0396	$0.040
Resistor	90Ω	2	708-RMCF0805JT91R0	$0.009	$0.018
Resistor	330Ω	8	791-WR08X331JTL	$0.007	$0.056
Locking Header	1x4 Header	2	538-87891-0408	$0.166	$0.332
Microcontroller	PIC16F1788-IP	1	579-PIC16F1788-I/SS	$2.140	$2.140
Capacitor	0.1uF	1	1276-1003-1-ND	$0.0088	$0.009
Header	1x6 Header	1	538-22-28-5061	$0.175	$0.175
Diode	1N4148	1	5399-1N4148WSCT-ND	$0.0187	$0.019
Resistor	150Ω	1	791-RMC1/10-151JTP	$0.007	$0.007
Switch	8x DIP Switch	1	2449-KG08R-ND	$0.5996	$0.600
Resistor	10kΩ	1	603-AC0805JR-0710KL	$0.010	$0.010
Diode	Green LED	1	160-1423-1-ND	$0.0688	$0.069
Diode	IN-P32TRRRGB	4	1830-IN-P32TRRRGBCT-ND	$0.1871	$0.748
IR Receiver	TSSP4038	1	782-TSSP4038	$0.638	$0.638
IR Receiver	TSOP53456	1	78-TSOP53456	$0.599	$0.599
PCB	PCB	1	N/A	$1.120	$1.120
				Total:	**$6.85**

Table 9.4: Labor Costs

Name	Hours	Rate	Cost
Alex Wheelock	147.92	$50	$7,395.83
Andrew Keller	124.92	$50	$6,246.00
Total Labor Cost:			**$13,641.83**

Table 9.5: Project Build Cost

Item	Quantity	Individual Cost	Total Cost
Blaster	2	$112.95	$225.90
Master	1	$35.86	$35.86
Slave	9	$6.85	$61.65
Total Build Cost:			**$323.41**

Table 9.6: Project Total Cost

Item	Cost
Labor	$13,641.83
Project Build	$323.41
Total Project Cost:	**$13,965.24**

10: Conclusion

There were high expectations regarding this semester's plans for the Laser Arcade project, with the goals to: finalize schematics and design PCBs, complete the target system, and complete the blaster. However, these goals were all accomplished this semester in the time allotted. All schematics were finalized, and PCBs were designed and tested. There is now a functional target system with working master/slave I2C communication, which is controlled by a VB program via UART. By now controlling the target system through VB, it greatly expands the capabilities of the target system. Additionally, there is now a working blaster mounted into a Nerf gun that players can now hold and interact with.

However, the Laser Arcade is not yet in a functional working state. The primary issue with this project's current state is the blaster's ability to hit targets. The problem is that the IR sensors on the targets filter out all visible light, not allowing them to see the light from the blaster's current laser to register hits properly. During the initial testing of the Laser Arcade project, the sensors were able to see the laser and register hits while the circuit was on the breadboard due to the proximity of the setup. While the blaster is very close to the sensors, the laser is still able to trigger the sensors, but it must be too close for the game to be playable. As a result, a future revision of the blaster will have to add an IR laser in some way. Fortunately, the addition of a second laser output on the PCB should make this problem easier to correct. The other problem with the project's current state is that there is currently only a single blaster built, and an additional blaster will be required for the game to reach a working state.

10.1: Recommendations

Recommendations for future development focus of the Laser Arcade project are to revise the blaster as well as construct a second blaster for the game to reach a functional state, and add additional modes to the VB program.

There are several things that could be changed in a revised version of the blaster. Primarily, it is recommended to add an IR laser so that it is possible to consistently hit targets. There are several ways to fix this issue. First it is possible to use the second Laser/PWM output of the current revision of the blaster PCB to add the additional IR laser, using the current laser for spotting and the IR laser for hitting the targets. Additionally, the current laser can be removed to leave only an IR laser, and use an optic such as a red dot sight for aiming the blaster. Once a blaster that works well with the target system is achieved, a second blaster will then need to be built in order for the game to be fully functional. Another recommendation for the future would be to design a custom blaster from scratch to be 3D printed which would reduce the costs of the blaster, and be able to better suit the needs of the blaster.

The target system currently is capable of running only a single game mode. While there is customization of how exactly this game mode plays, the control of the target system was moved to VB to make it easier to add on to. Which is why it is also recommended to program additional games. Some ideas include tic-tac-toe, a reaction test, a keyboard mode, etc.

11: Appendix A (Schematics)

11.1: Blaster

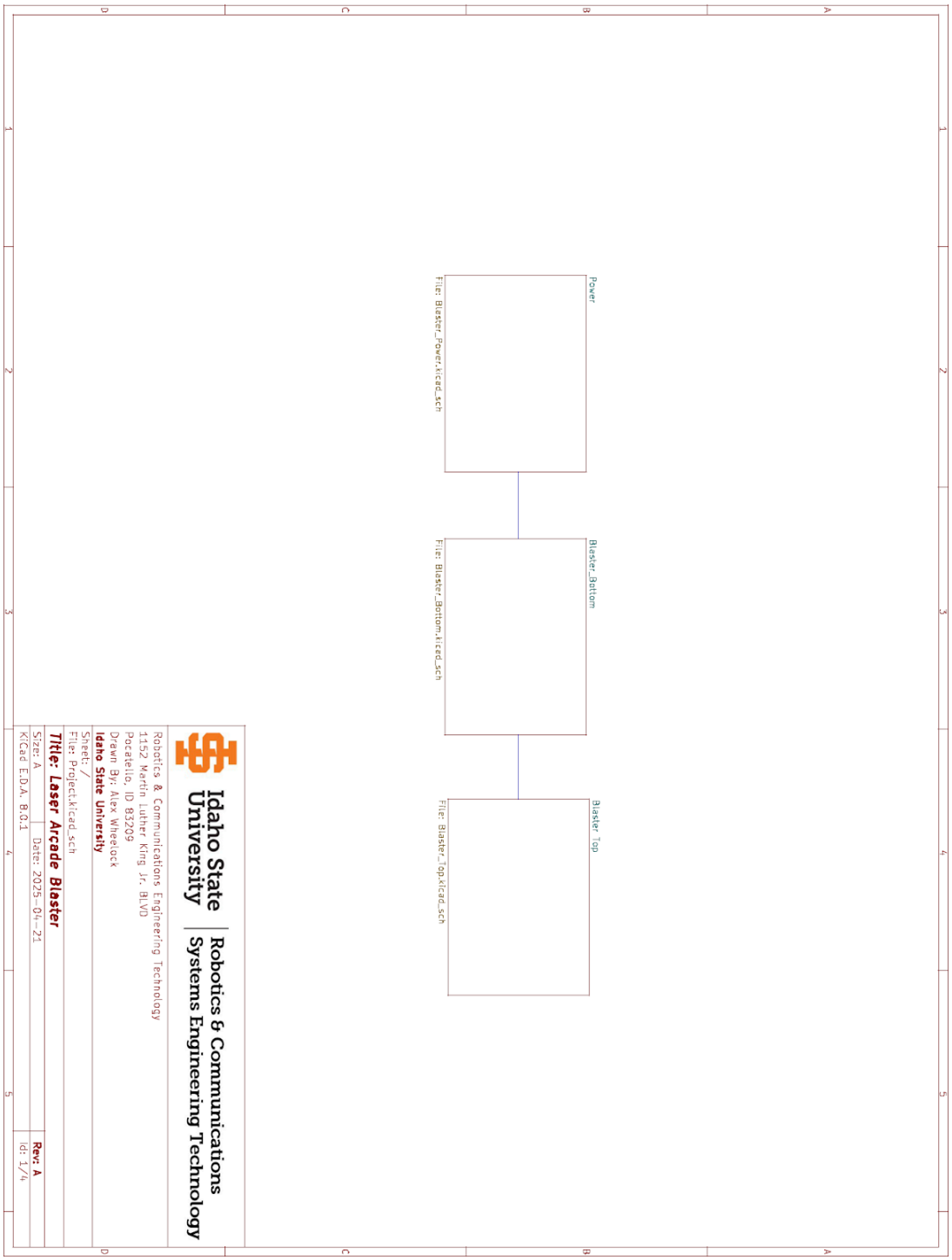


Figure 11.1.1: Blaster Main Schematic Sheet

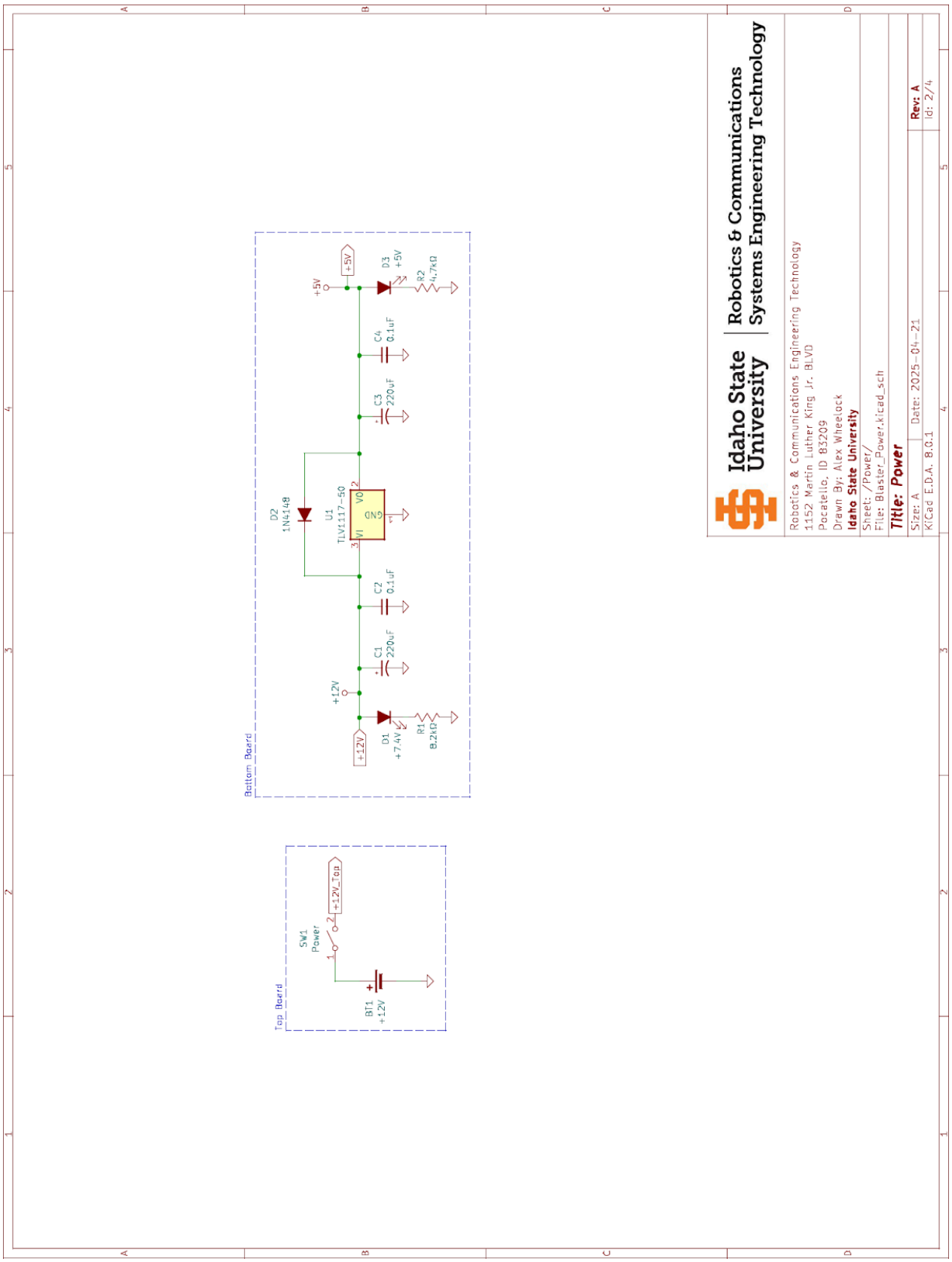


Figure 11.1.2: Blaster Power Schematic

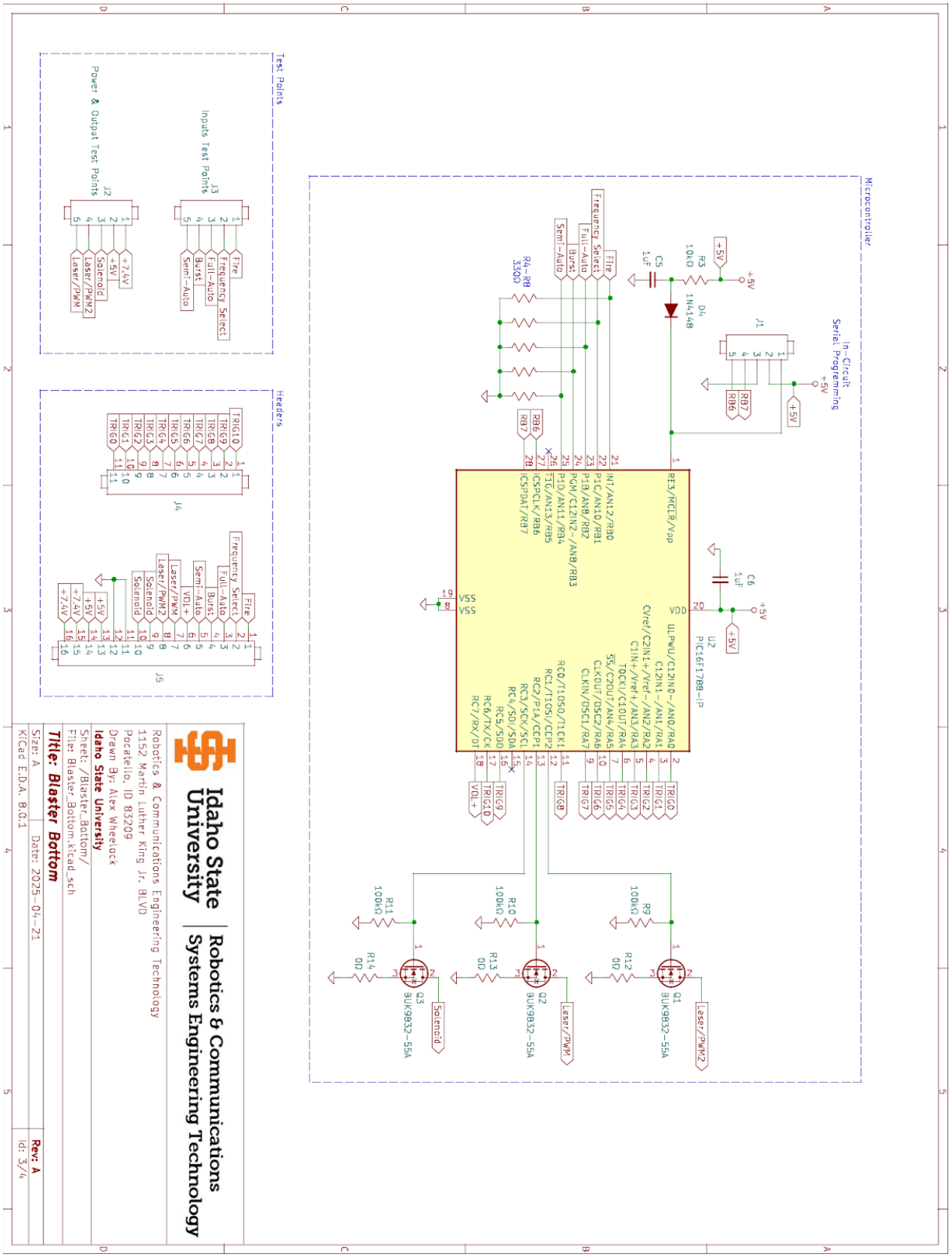


Figure 11.1.3: Blaster Bottom Board Schematic

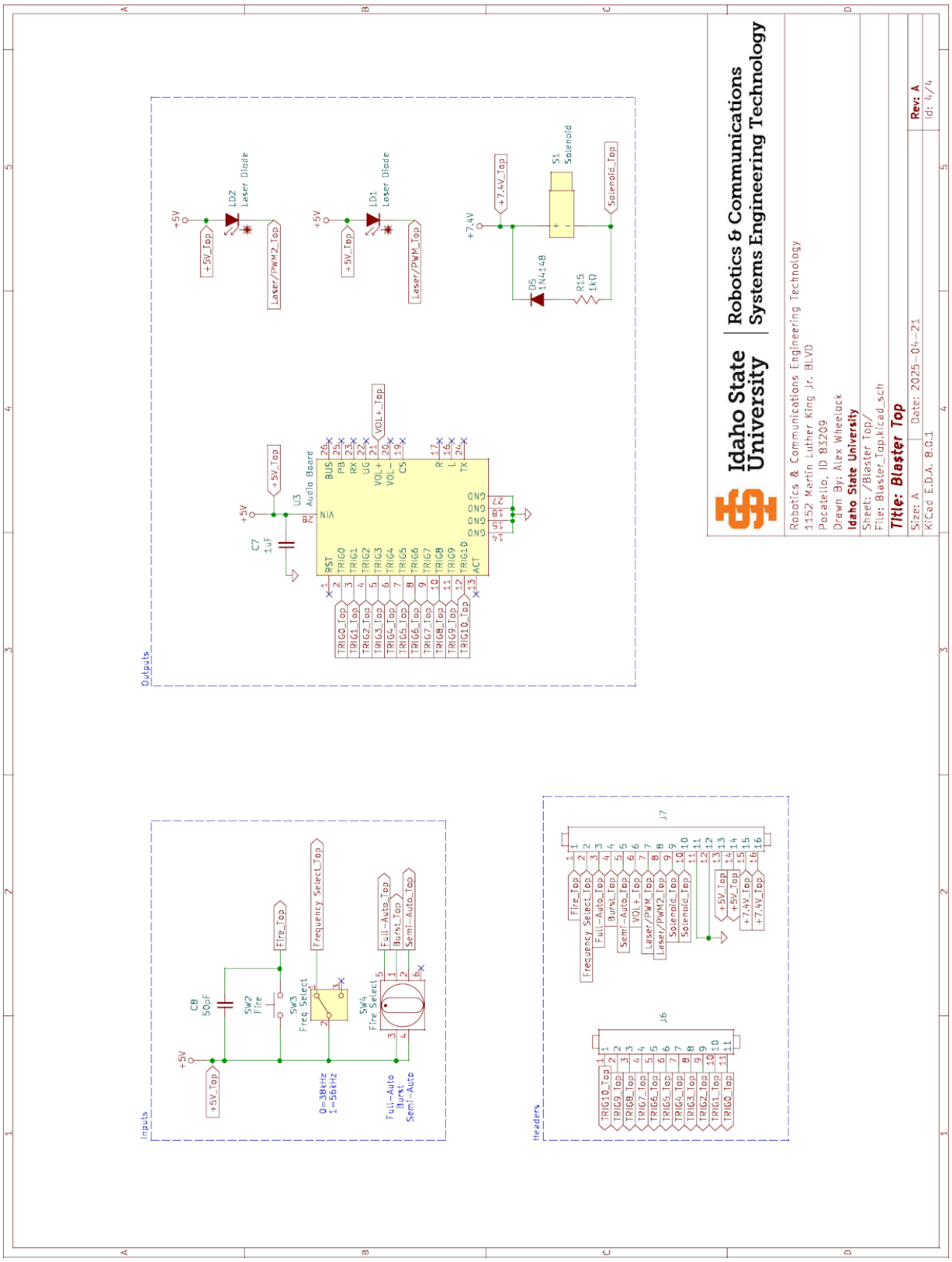


Figure 11.1.4: Blaster Top Board Schematic

11.2: Target Master

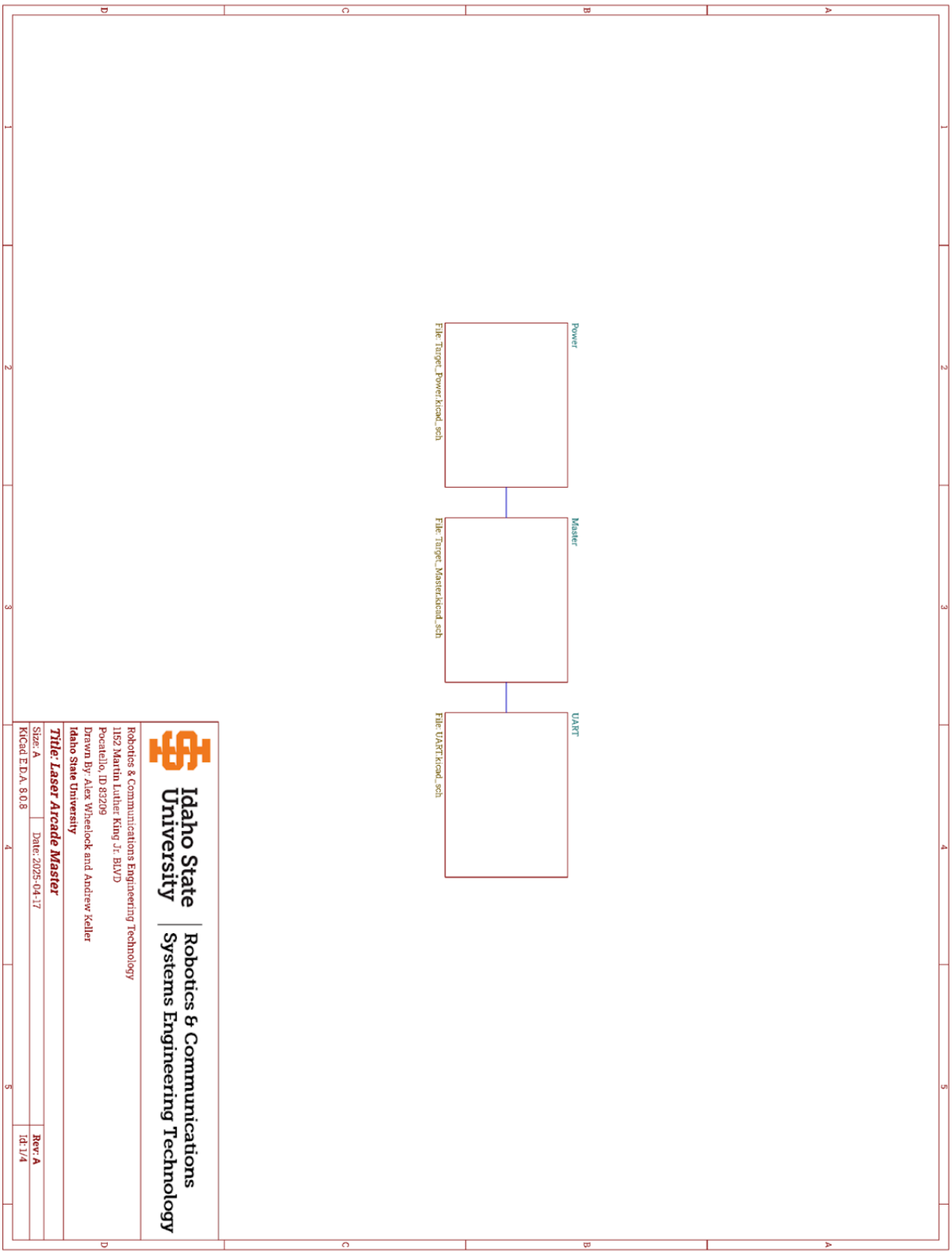


Figure 11.2.1: Target Master Main Schematic Sheet

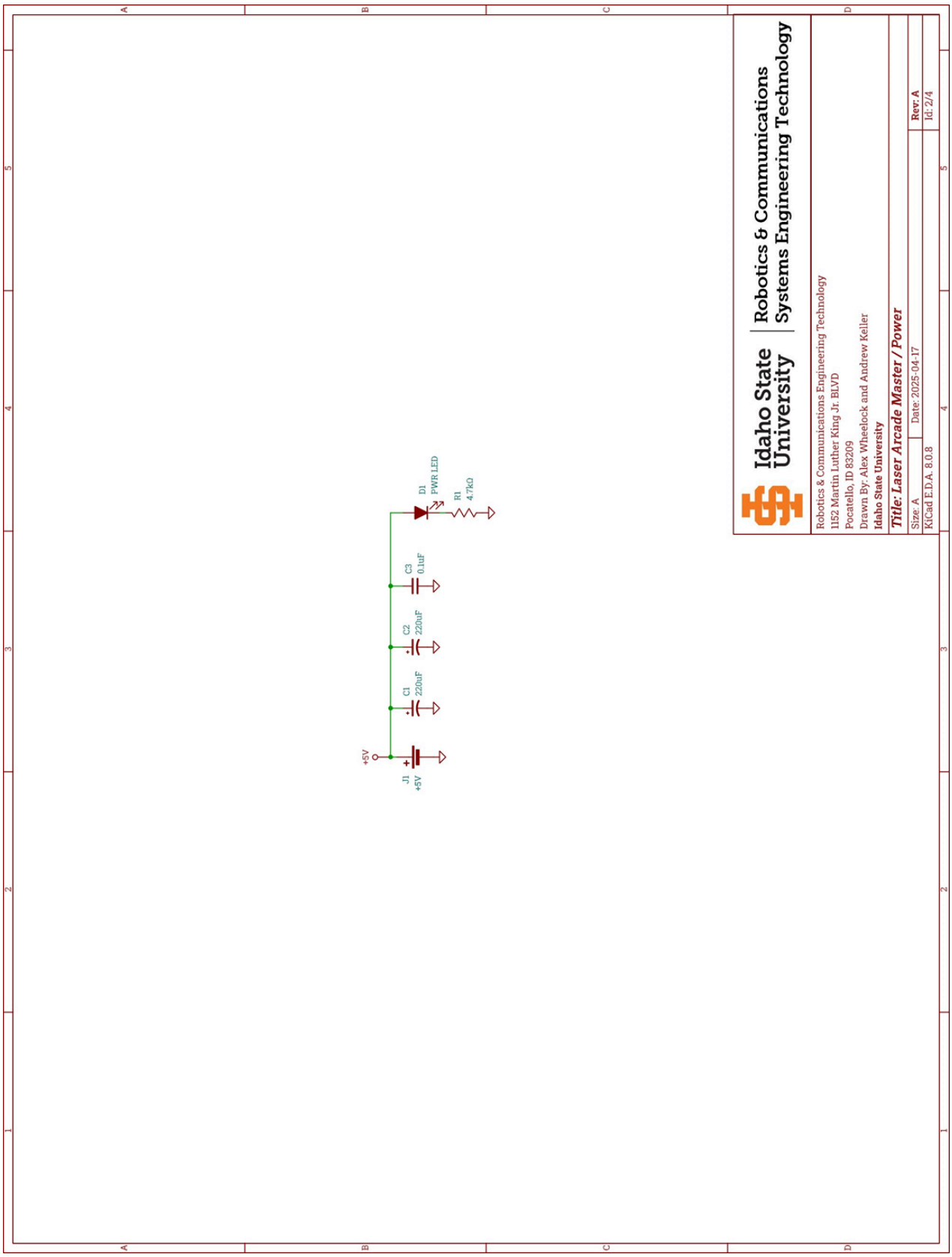


Figure 11.2.2: Target Master Power Schematic

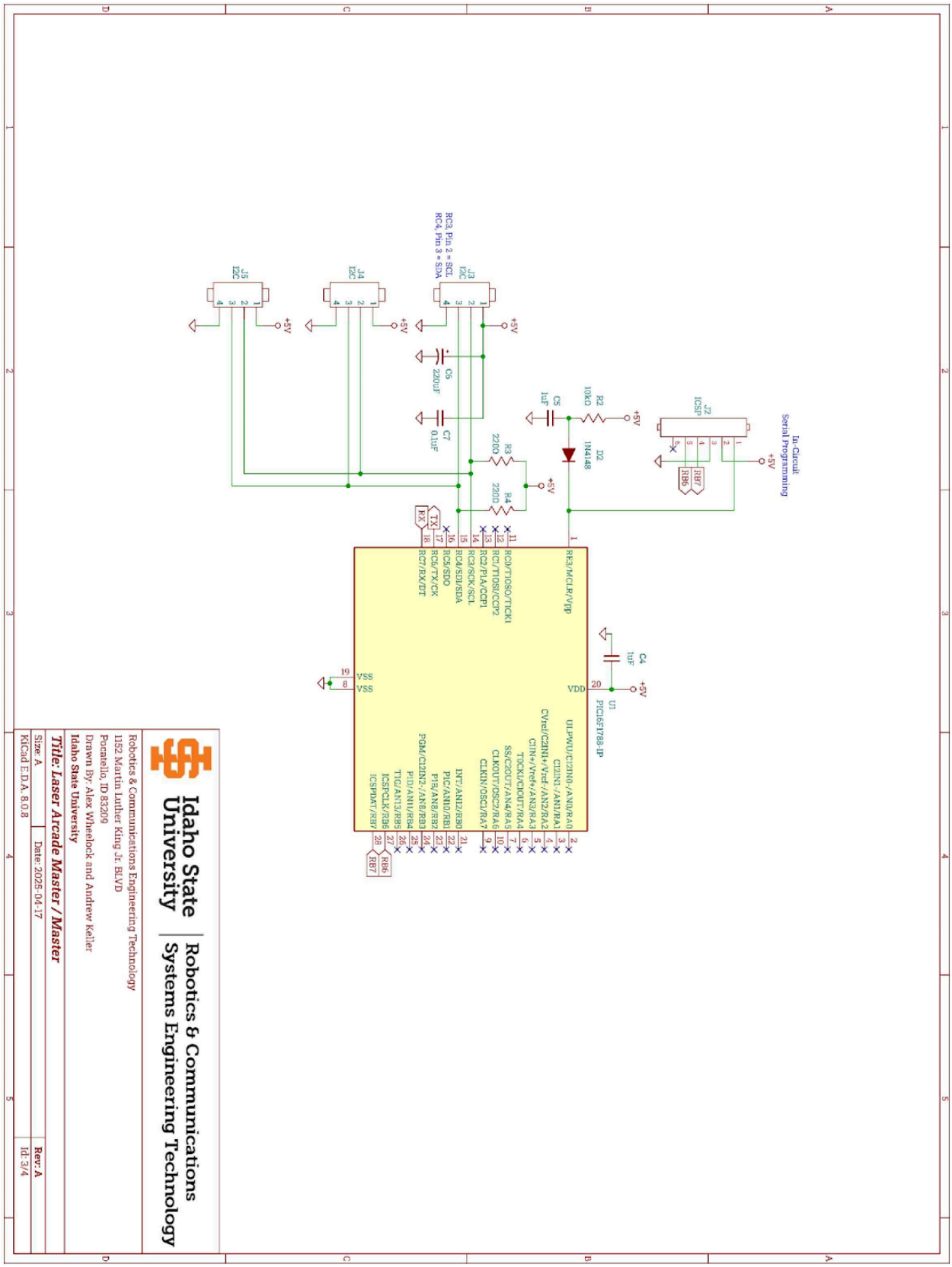


Figure 11.2.3: Target Master Circuit Schematic

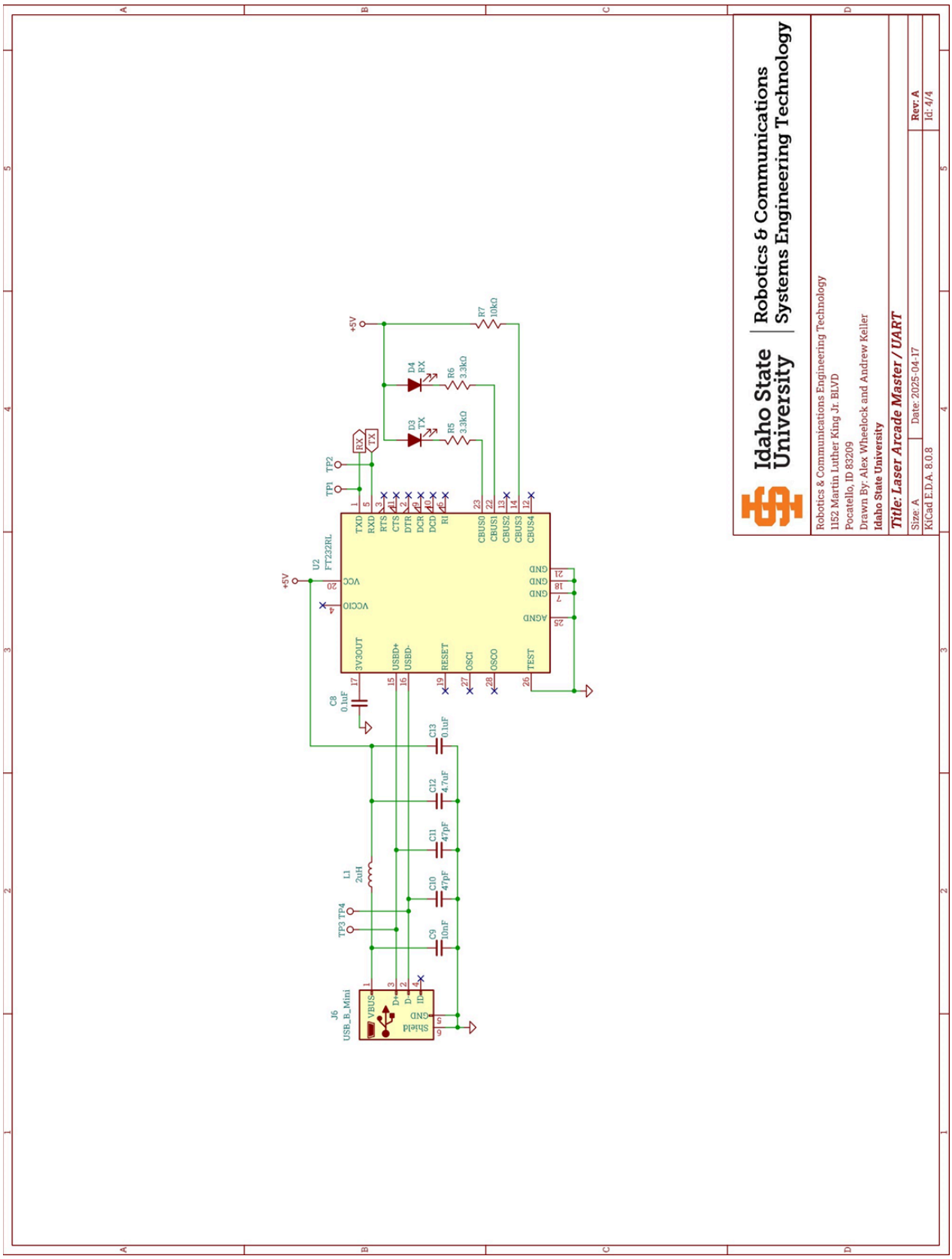


Figure 11.2.4: Target Master UART Schematic

11.3: Target Slave

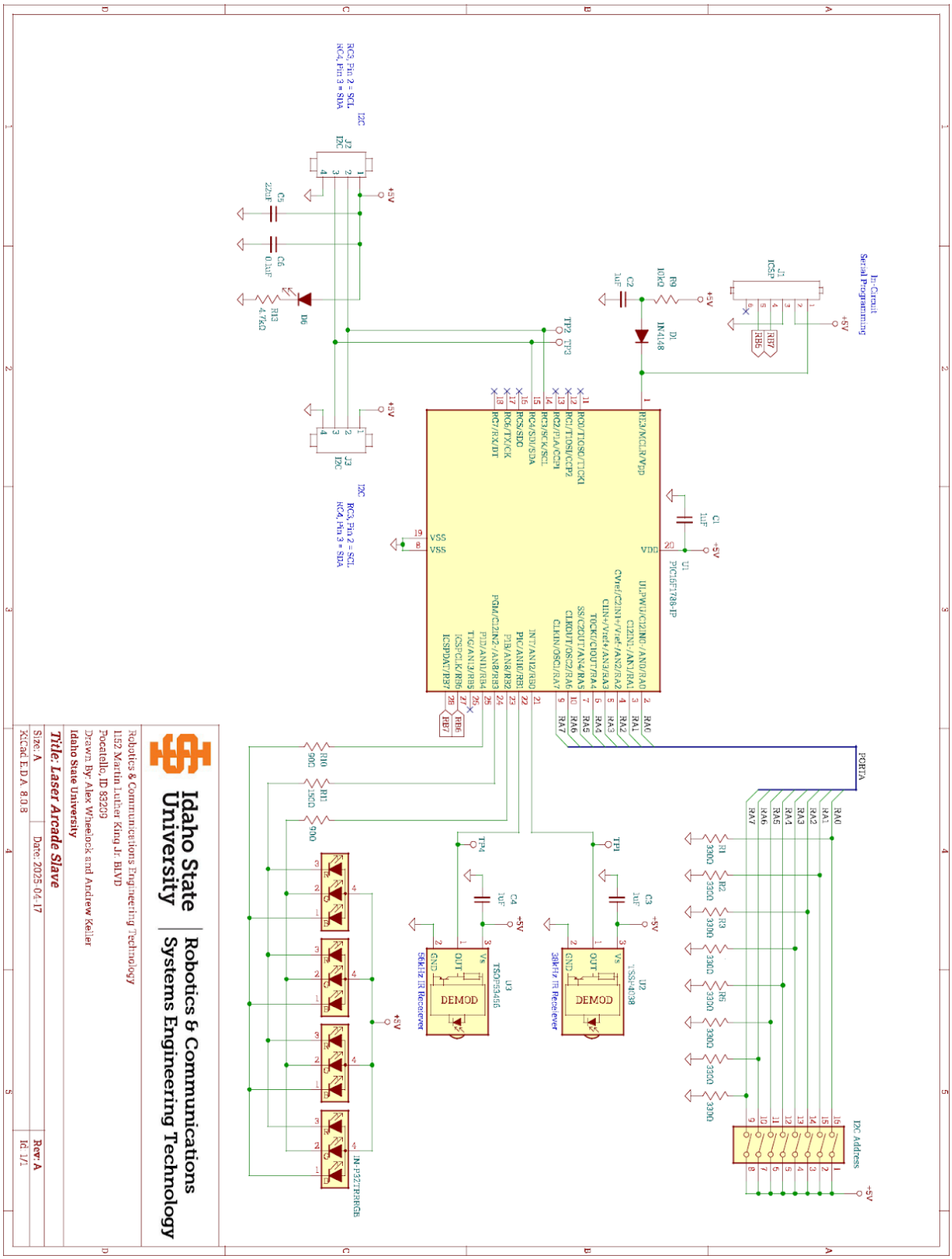


Figure 11.3.1: Target Slave Schematic

12: Appendix B (PCBs)

12.1: Blaster

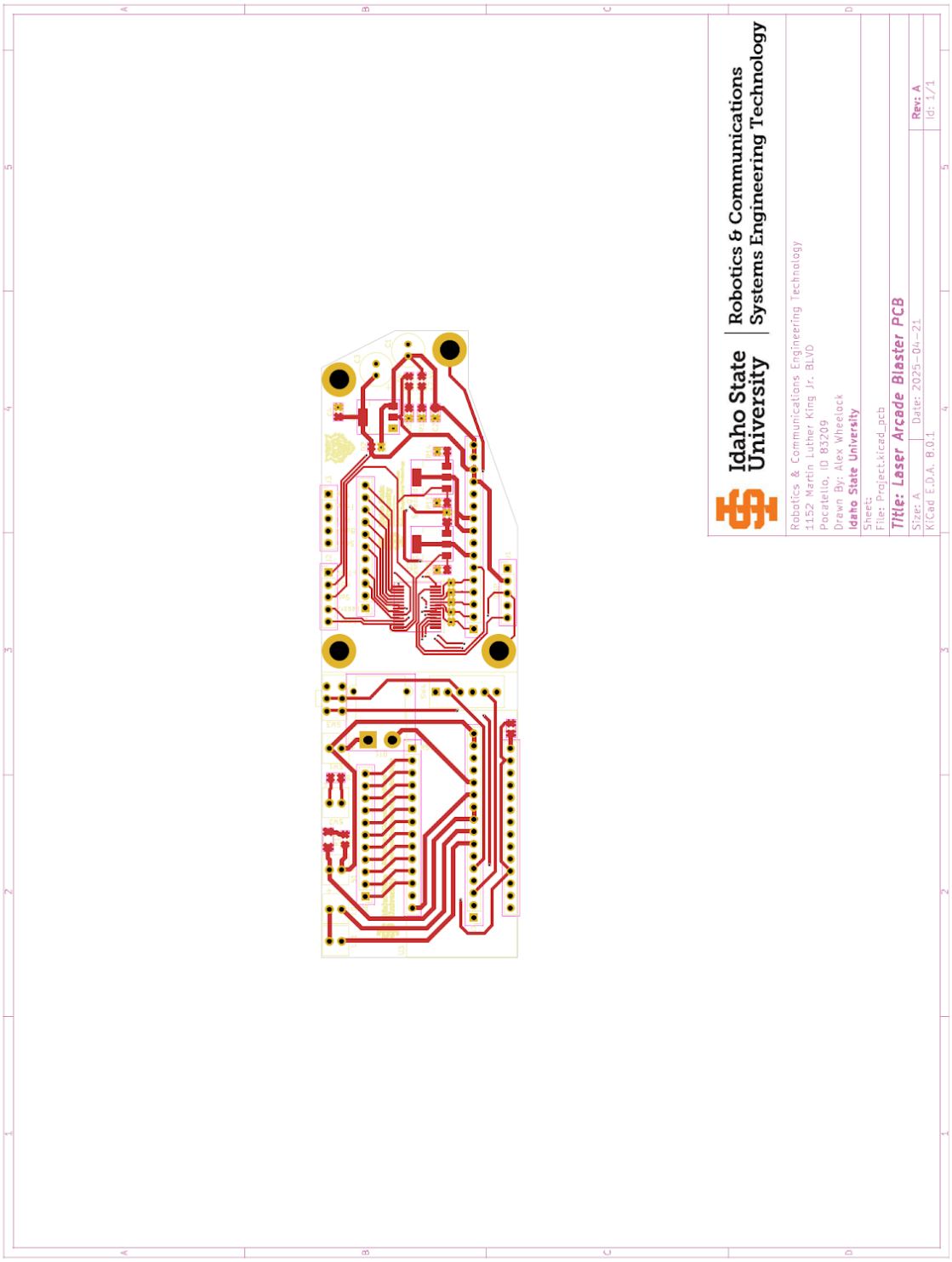


Figure 12.1.1: Blaster PCB Front

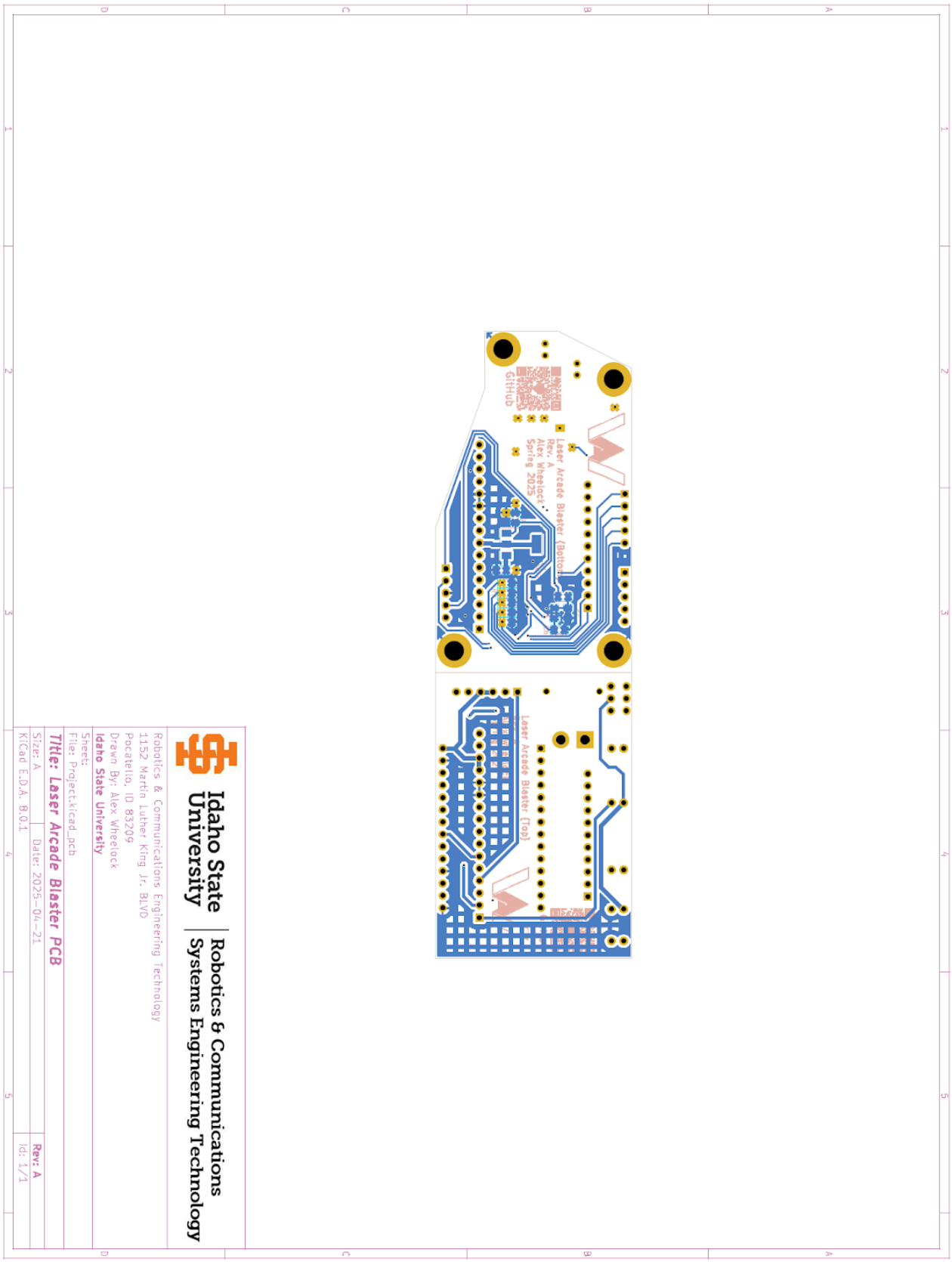


Figure 12.1.2: Blaster PCB Back

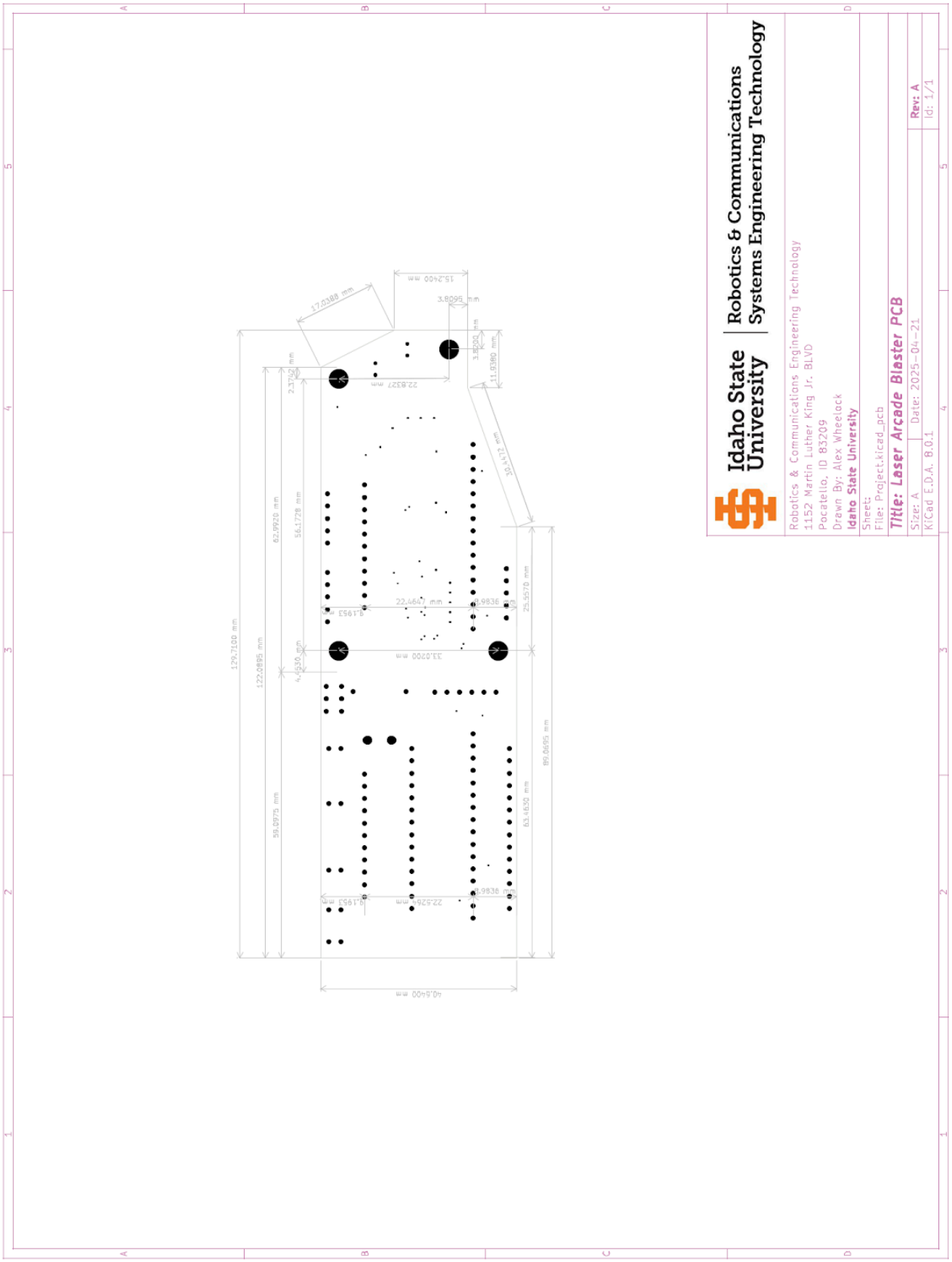


Figure 12.1.3: Blaster PCB Mechanical Drawing

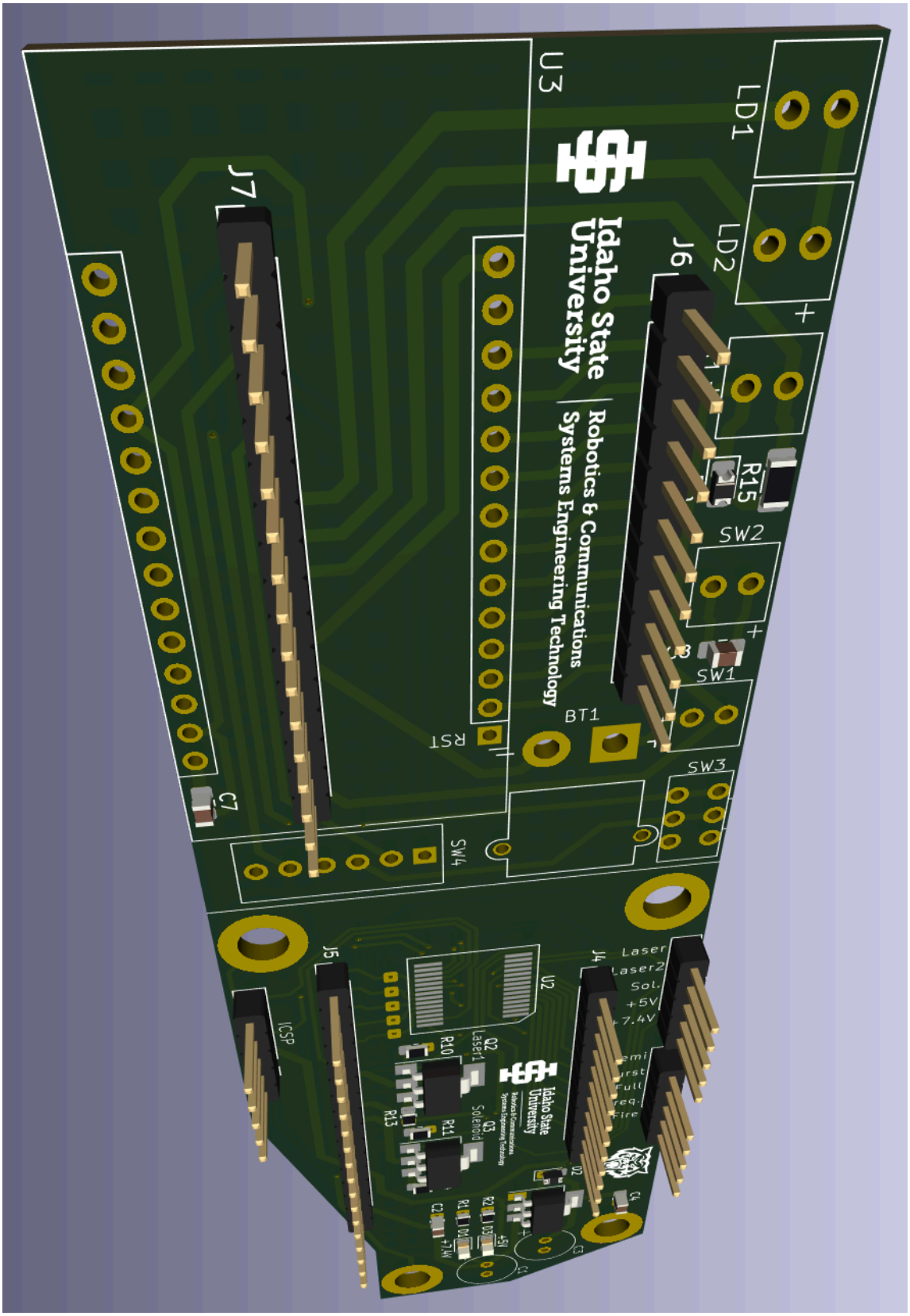


Figure 12.1.4: Blaster PCB 3D KiCad Render (Front)

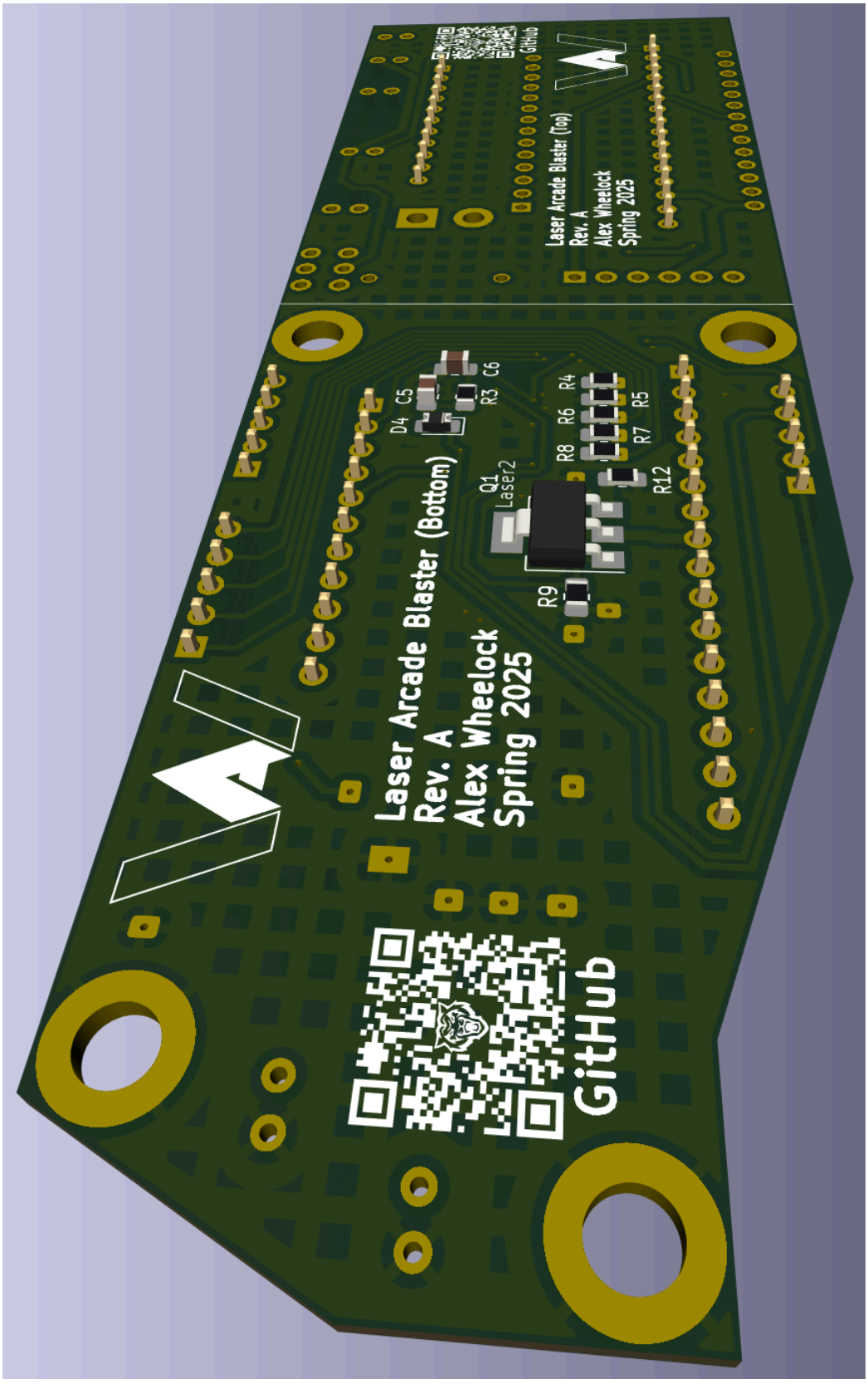


Figure 12.1.5: Blaster PCB 3D KiCad Render (Back)

12.2: Target Master

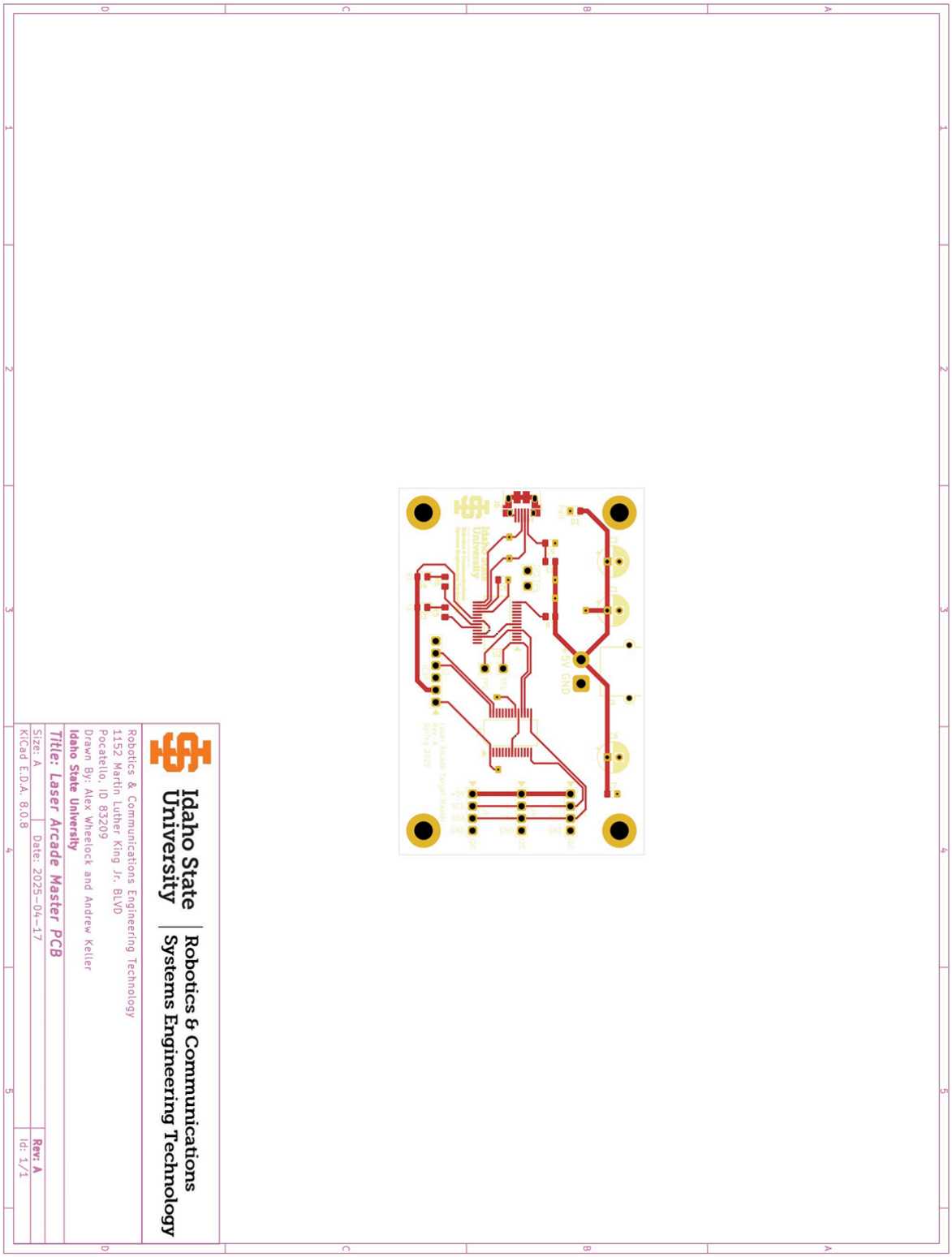


Figure 12.2.1: Target Master PCB Front

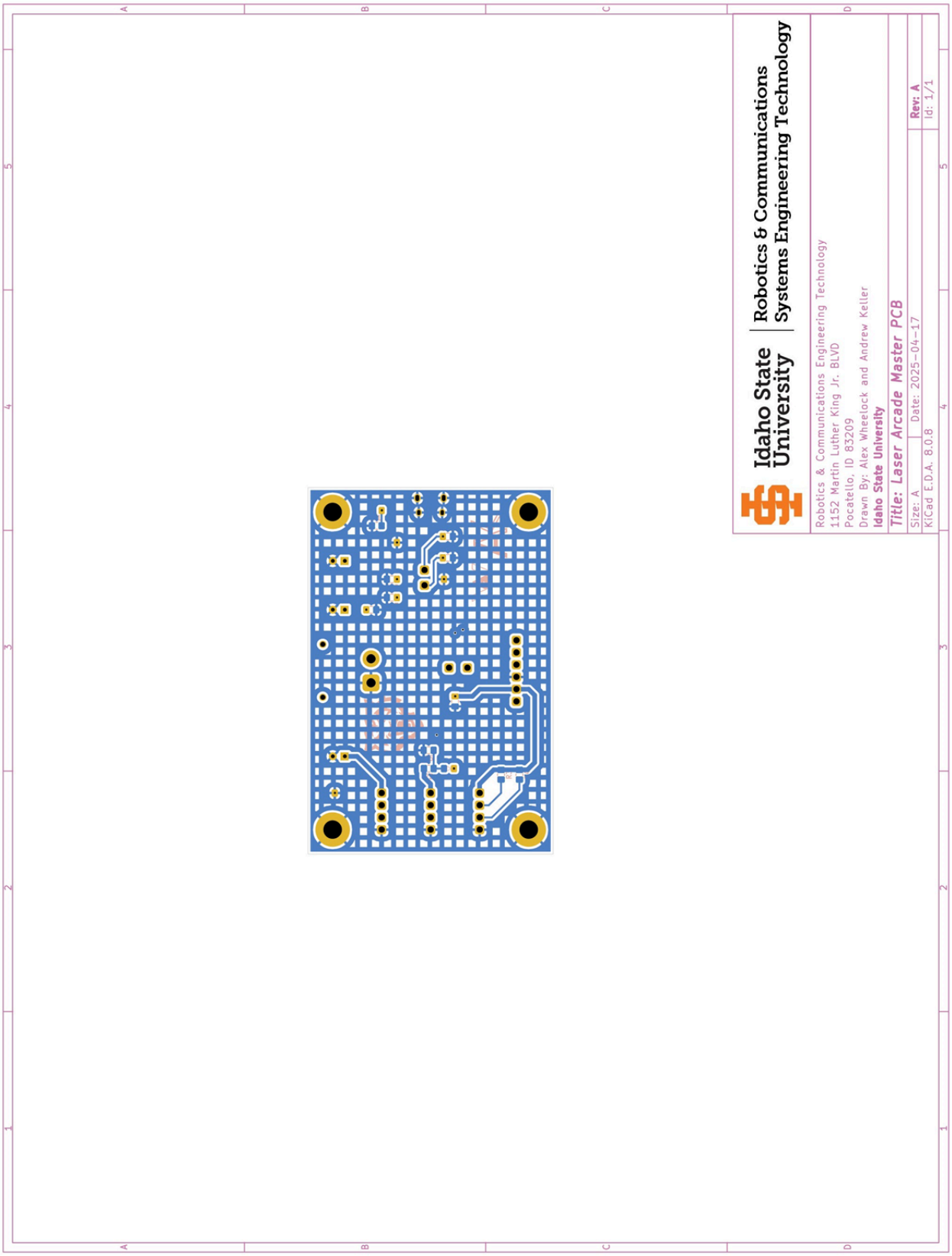


Figure 12.2.2: Target Master PCB Back

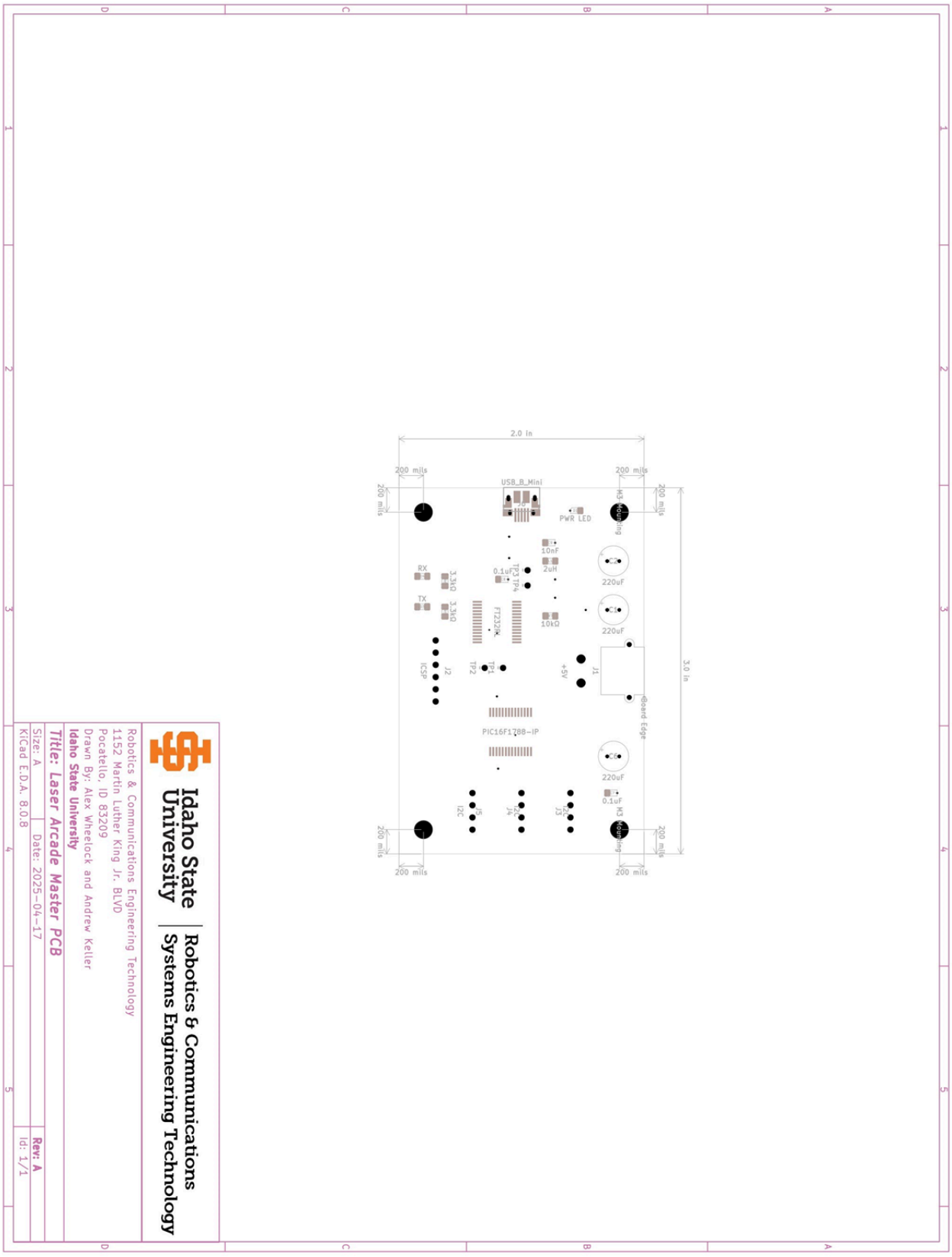


Figure 12.2.3: Target Master PCB Mechanical Drawing

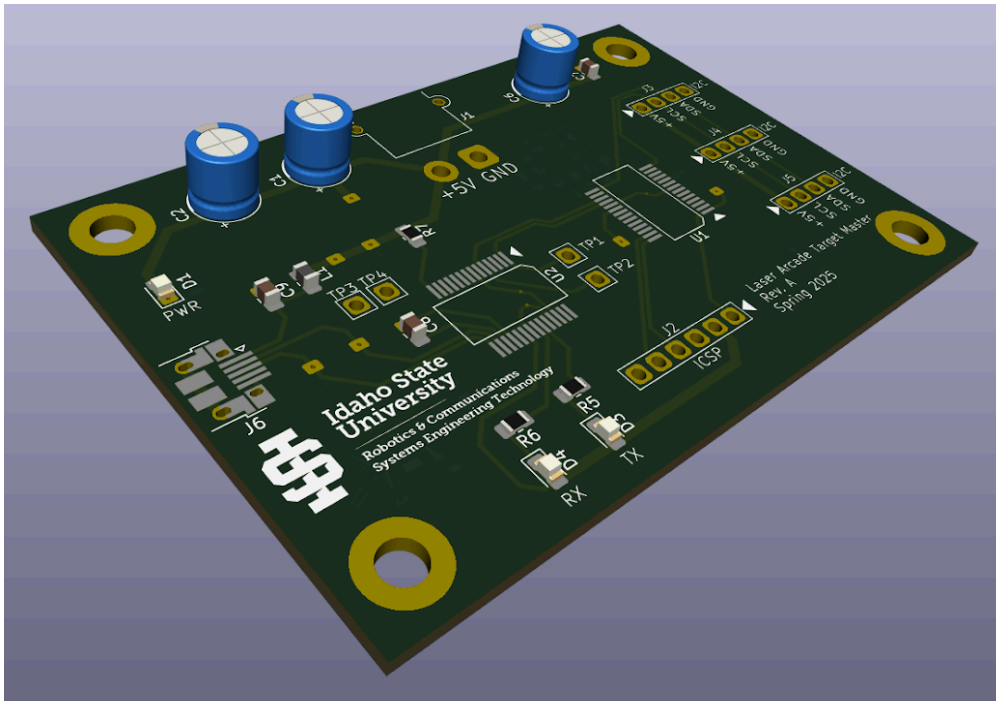


Figure 12.2.4: Target Master PCB 3D KiCad Render (Front)

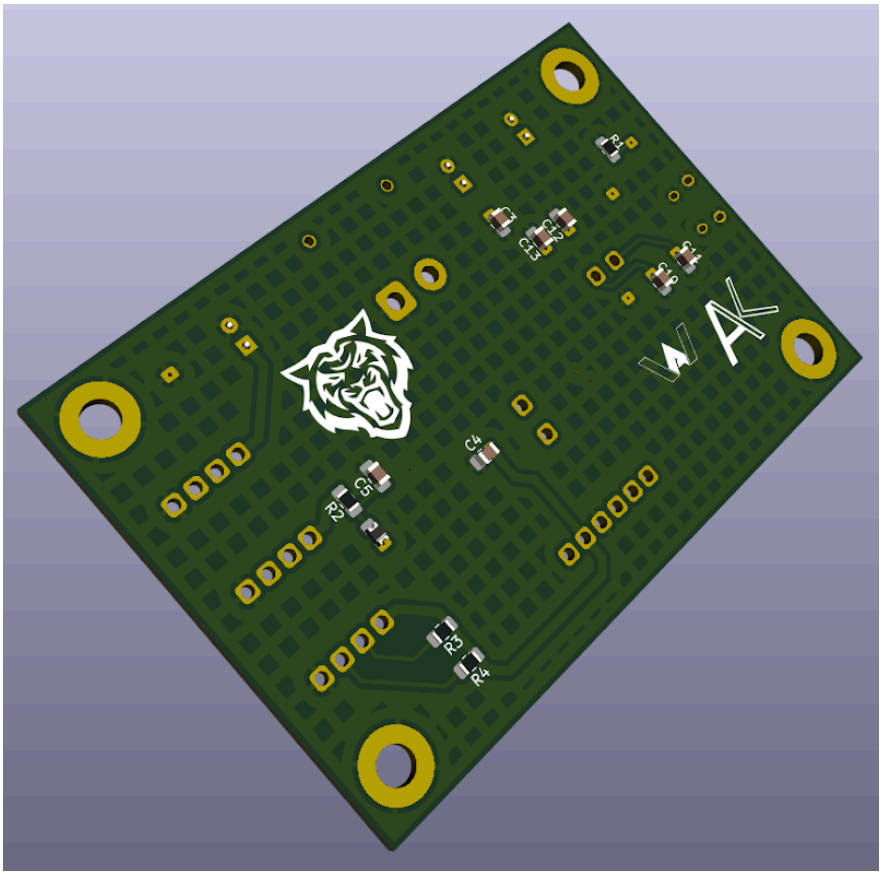


Figure 12.2.5: Target Master PCB 3D KiCad Render (Back)

12.3: Target Slave

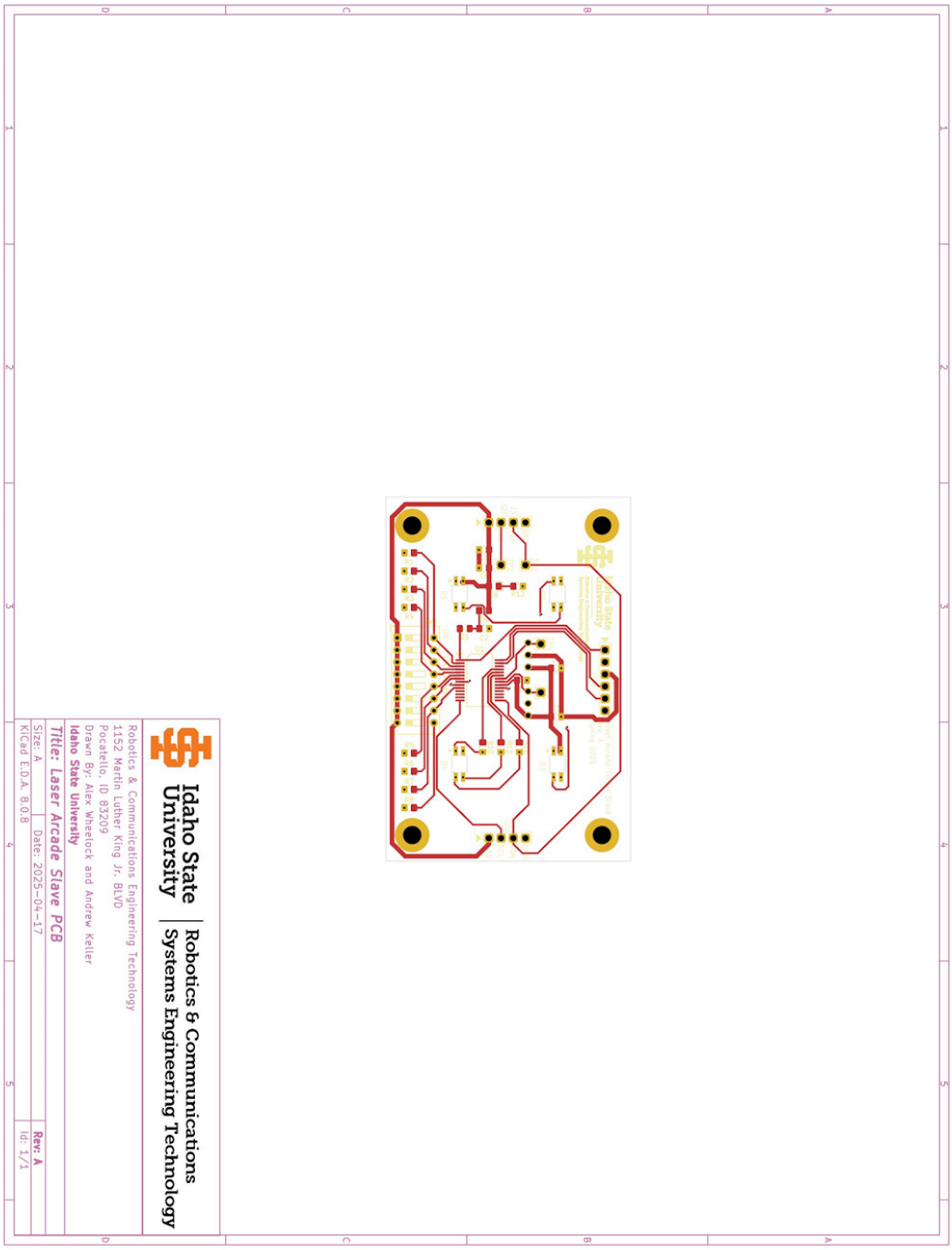


Figure 12.3.1: Target Slave PCB Front

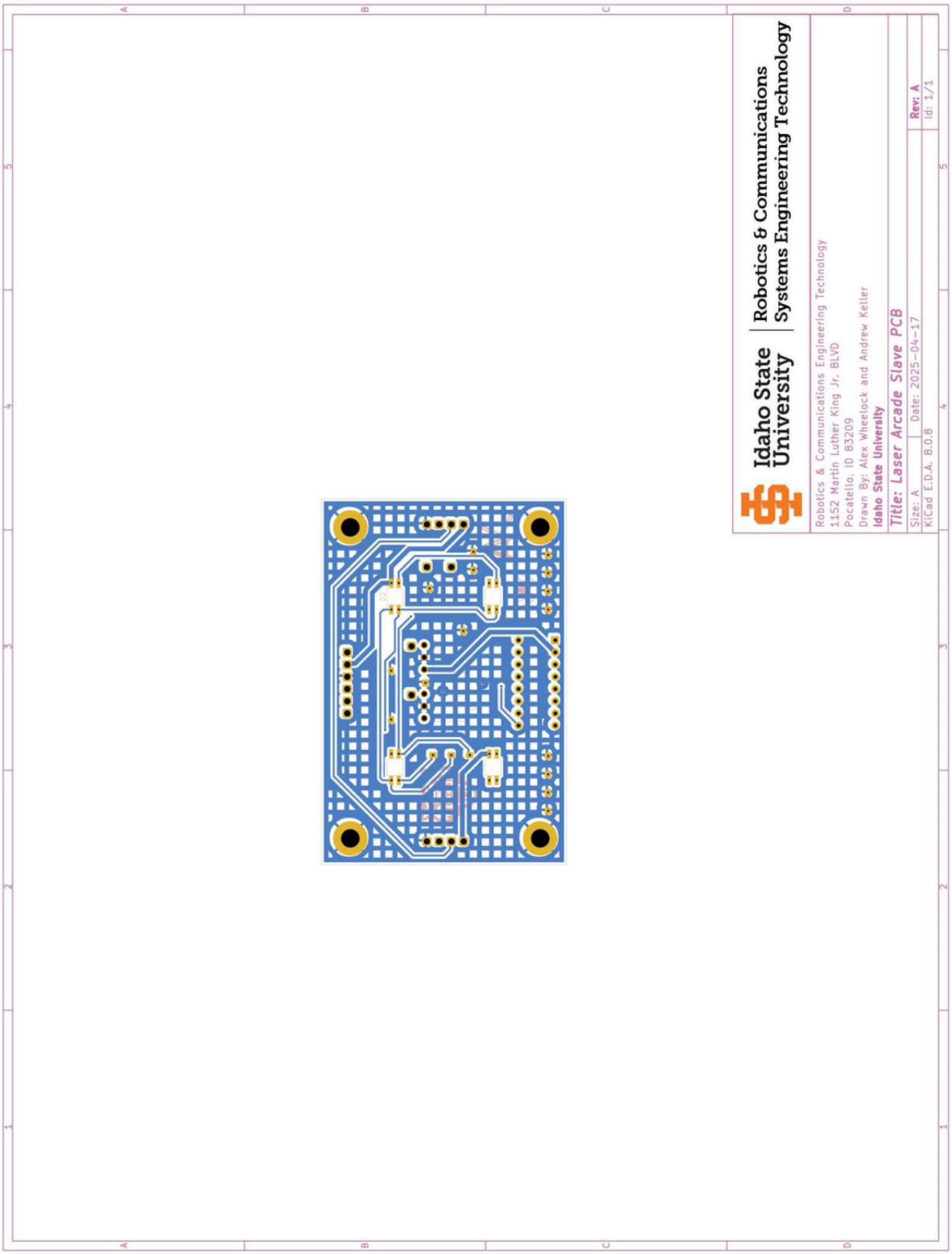


Figure 12.3.2: Target Slave PCB Back

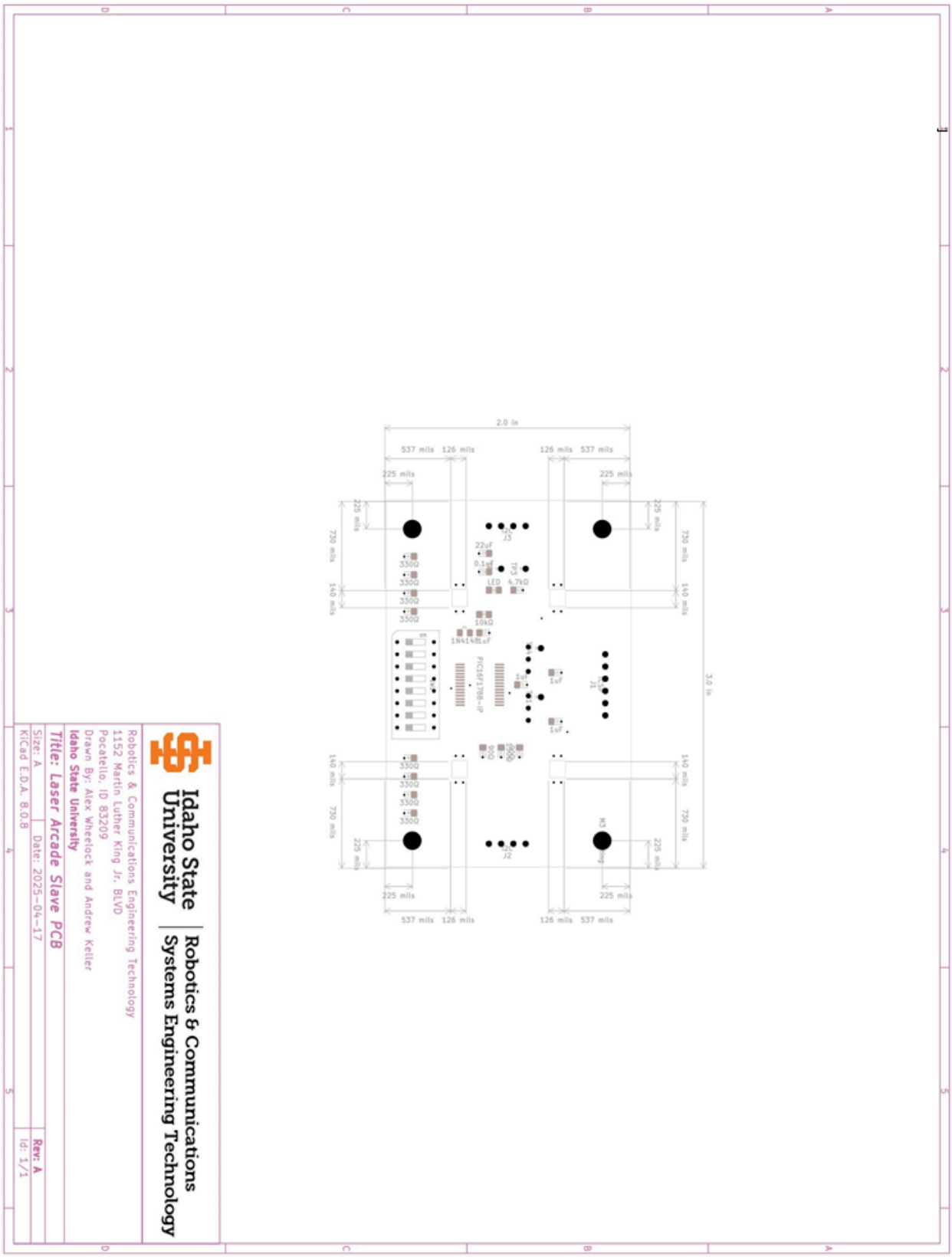


Figure 12.3.3: Target Slave PCB Mechanical Drawing

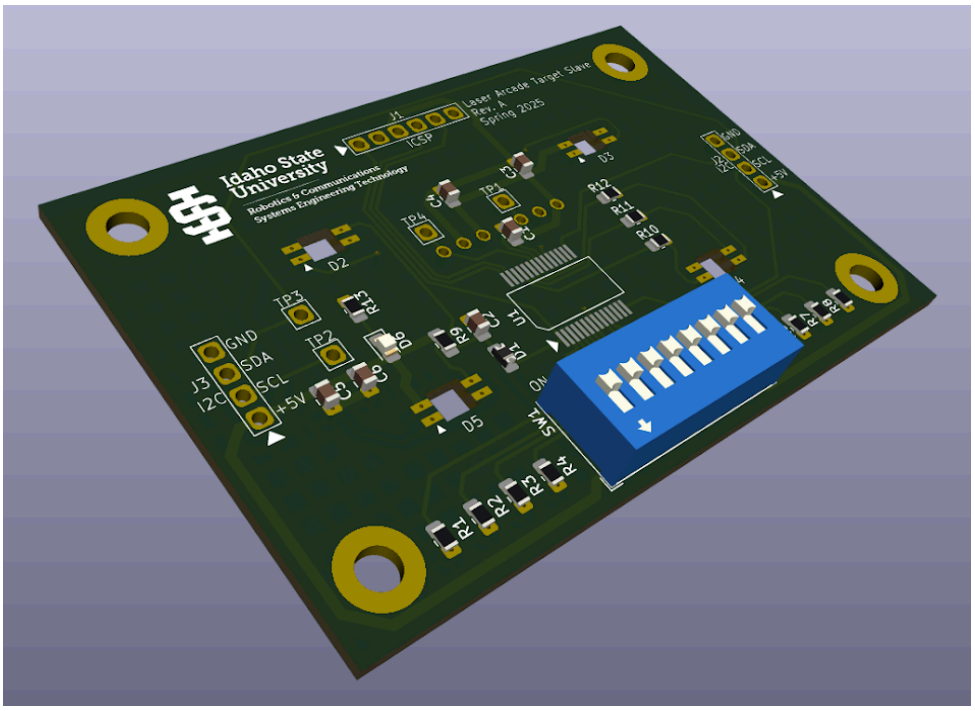


Figure 12.3.4: Target Slave PCB 3D KiCad Render (Front)

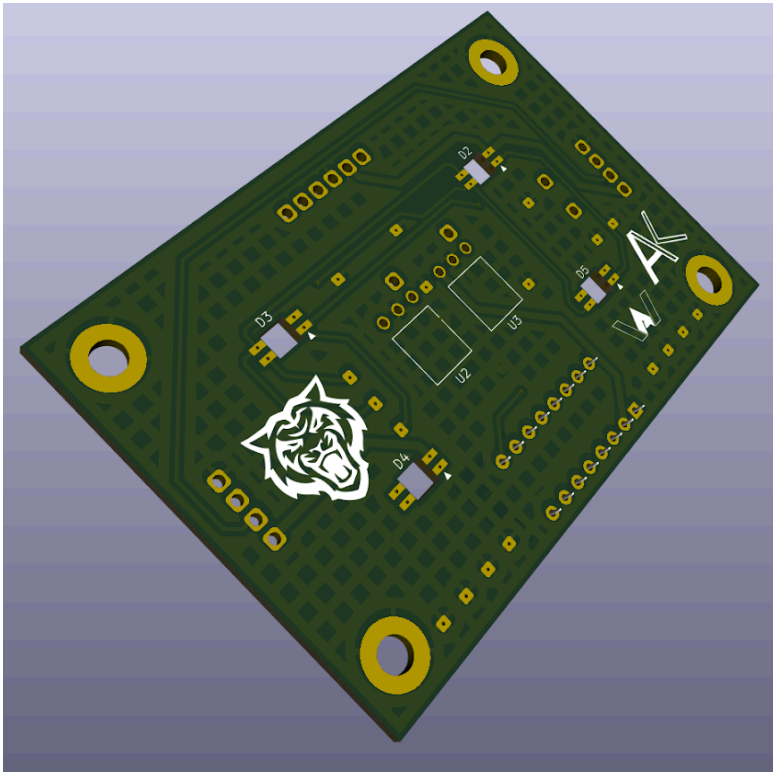


Figure 12.3.5: Target Slave PCB 3D KiCad Render (Back)

13: Appendix C (Mechanical Drawings)

13.1: Blaster

13.1.1: Speaker Box

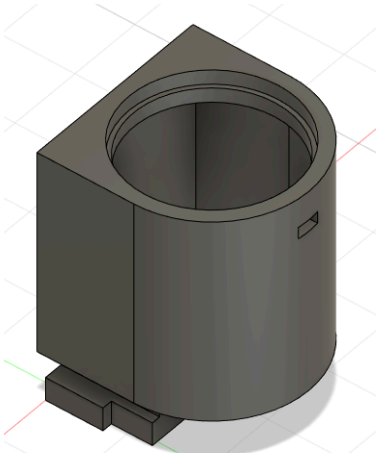


Figure 13.1.1.1: Blaster Speaker Box 3D View

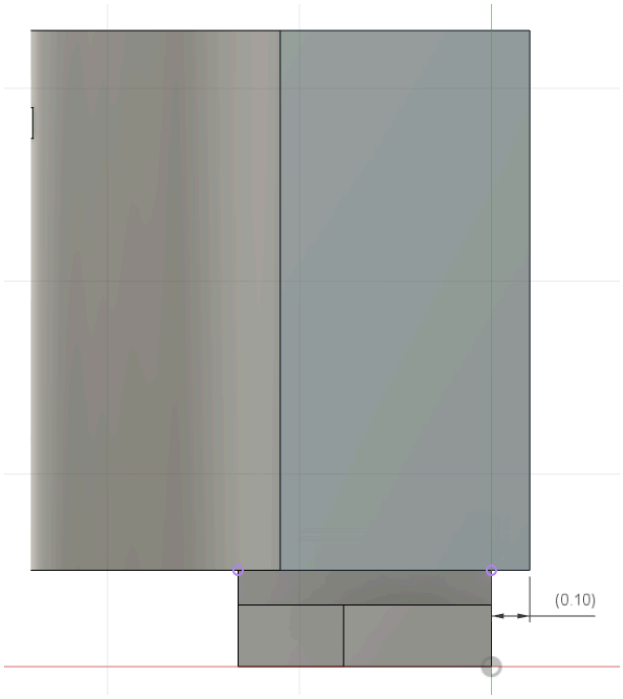


Figure 13.1.1.2: Blaster Speaker Box Side

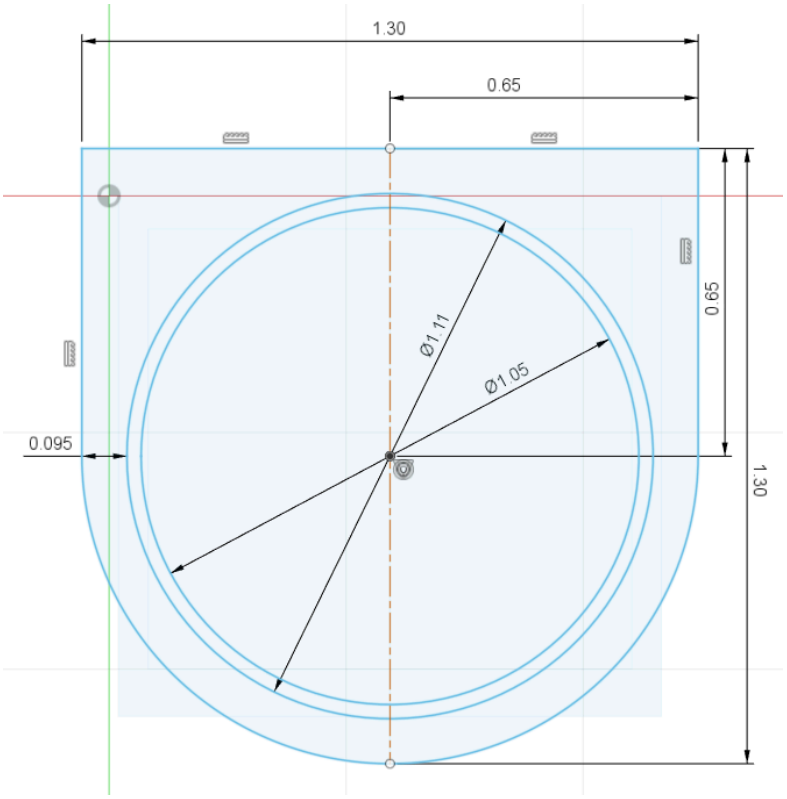


Figure 13.1.1.3: Blaster Speaker Face Mechanical Drawing

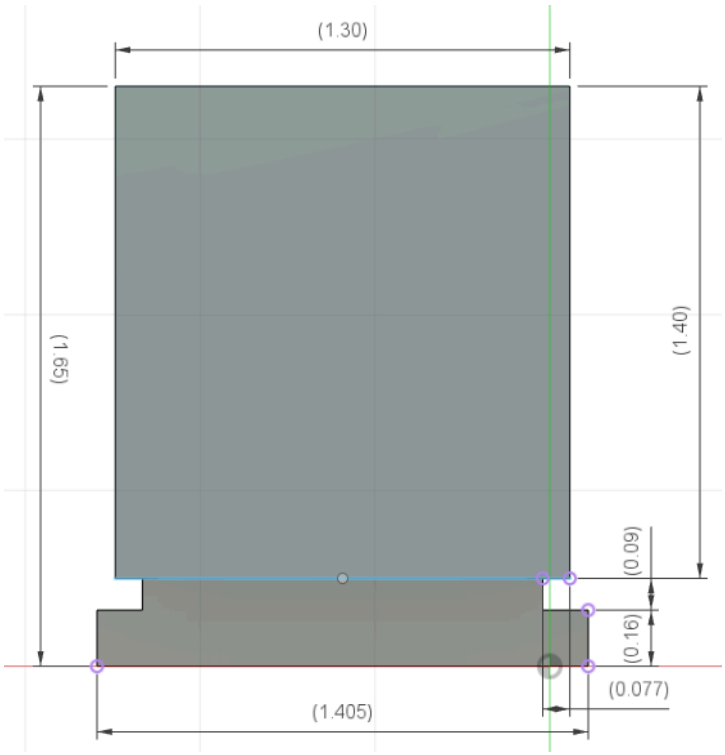


Figure 13.1.1.4: Blaster Speaker Box Back

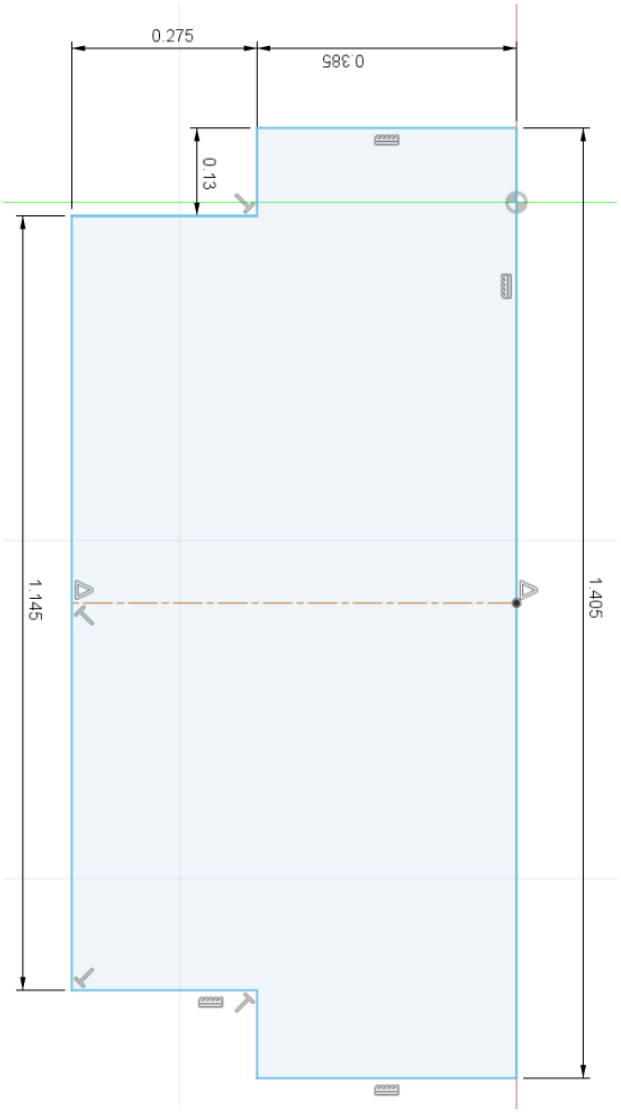


Figure 13.1.1.5: Blaster Speaker Box Mount Drawing

13.1.2: Barrel

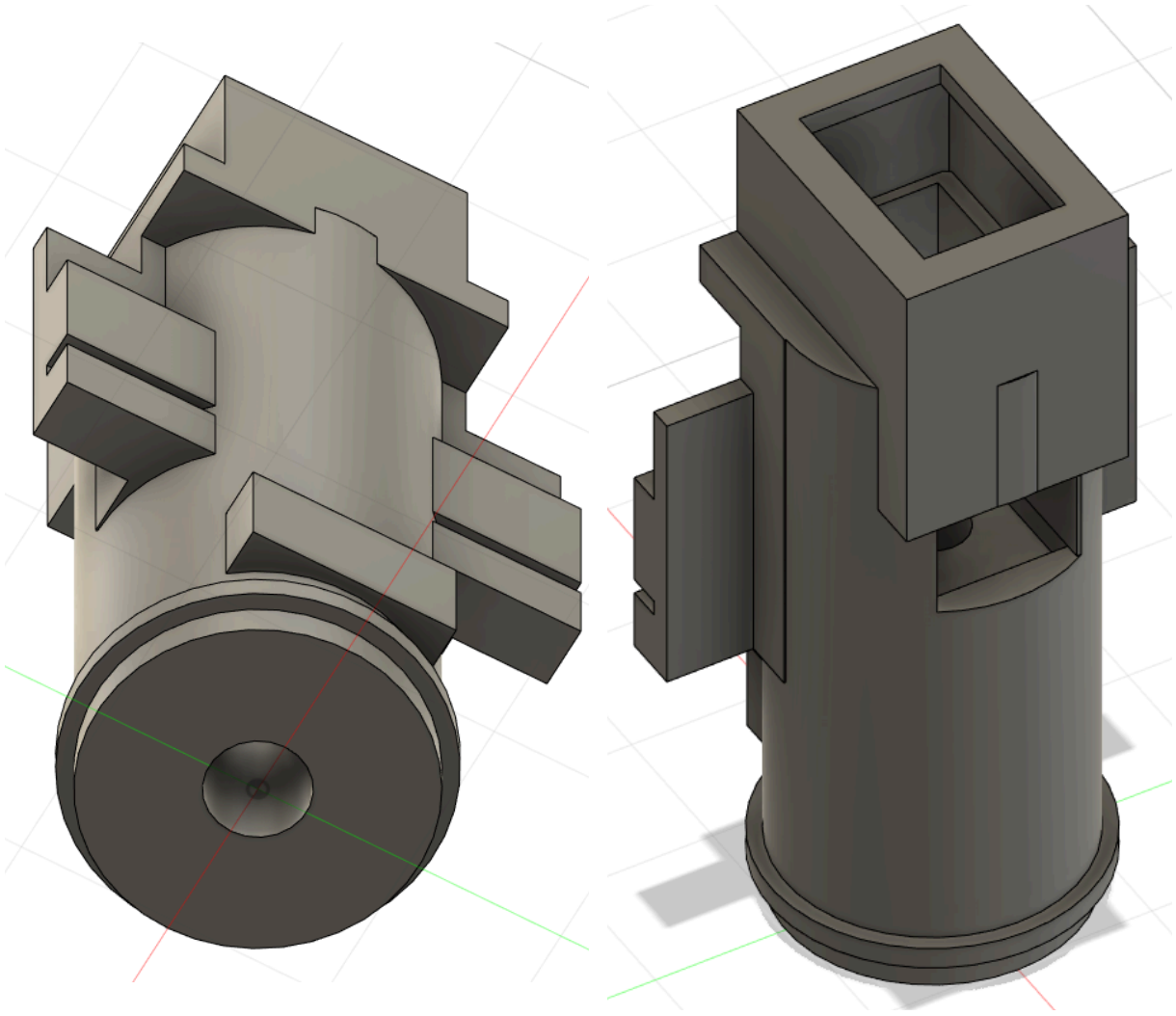


Figure 13.1.2.1: Blaster Barrel 3D Views

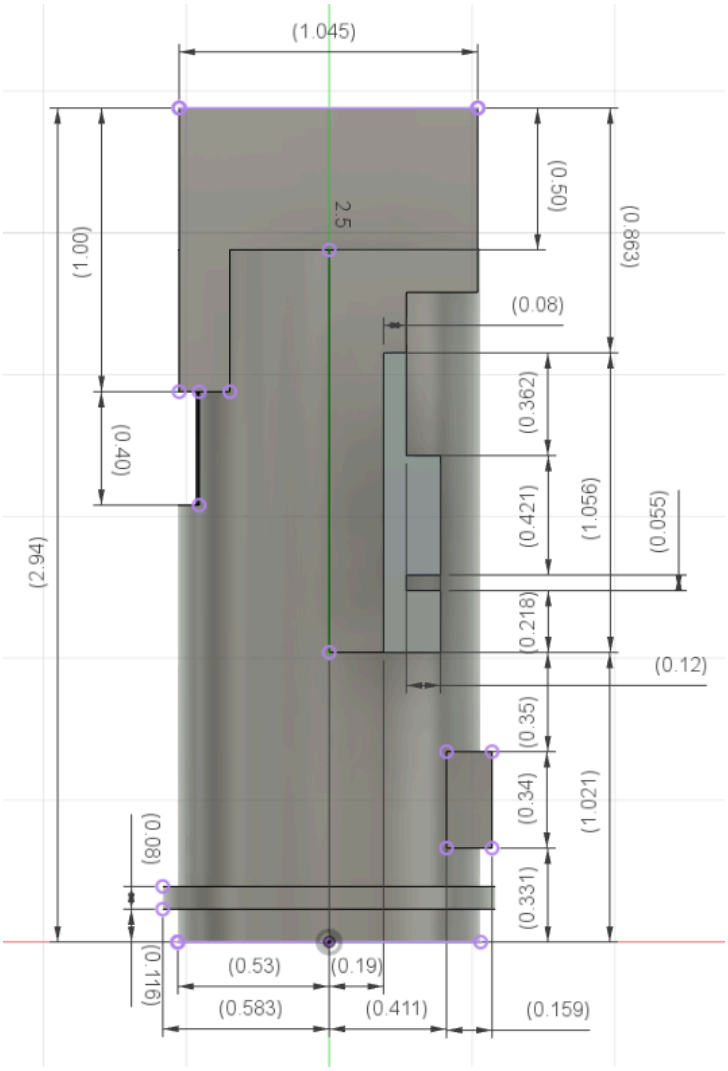


Figure 13.1.2.2: Blaster Barrel Side Drawing

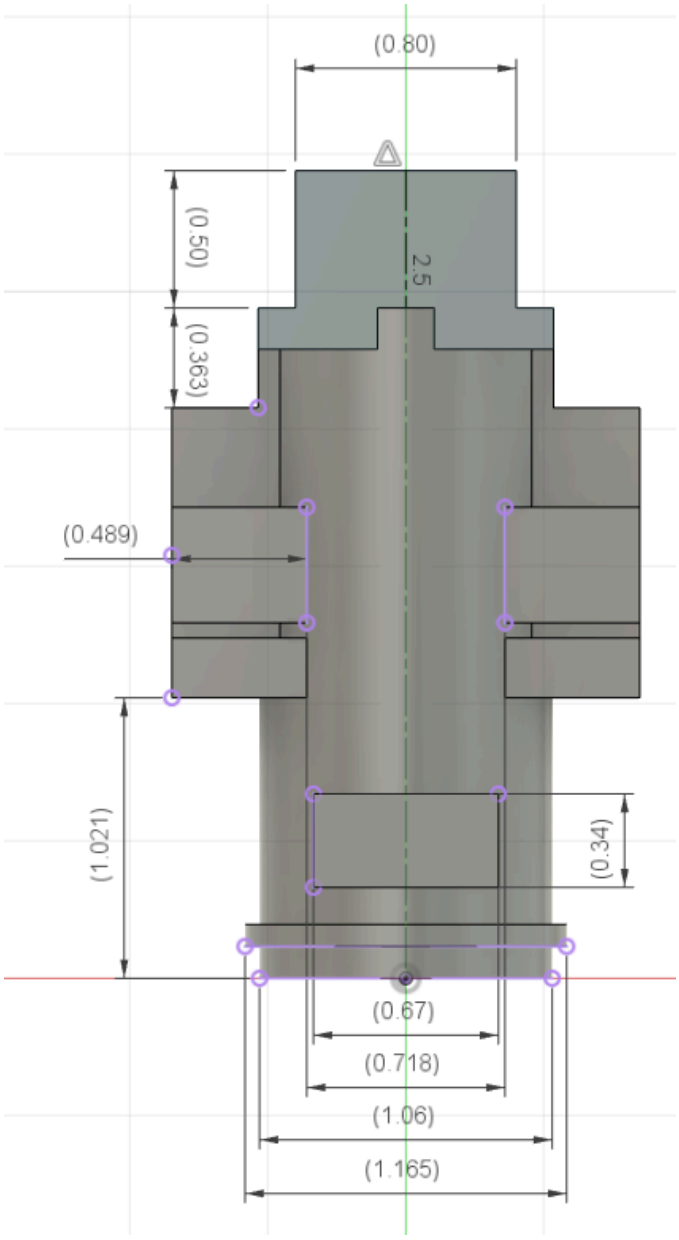


Figure 13.1.2.3: Blaster Barrel Top Drawing

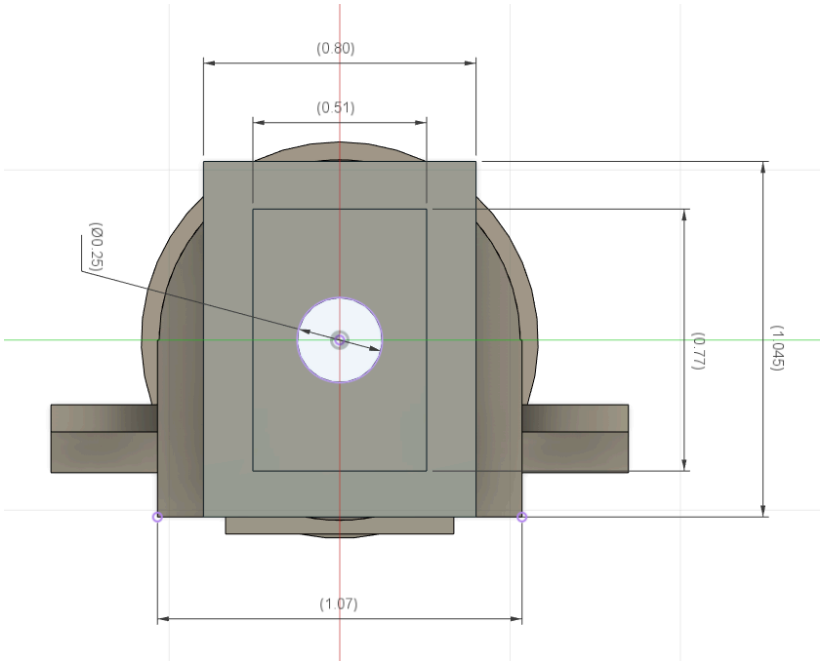


Figure 13.1.2.4: Blaster Barrel Rear Drawing

13.1.3: Solenoid Mount

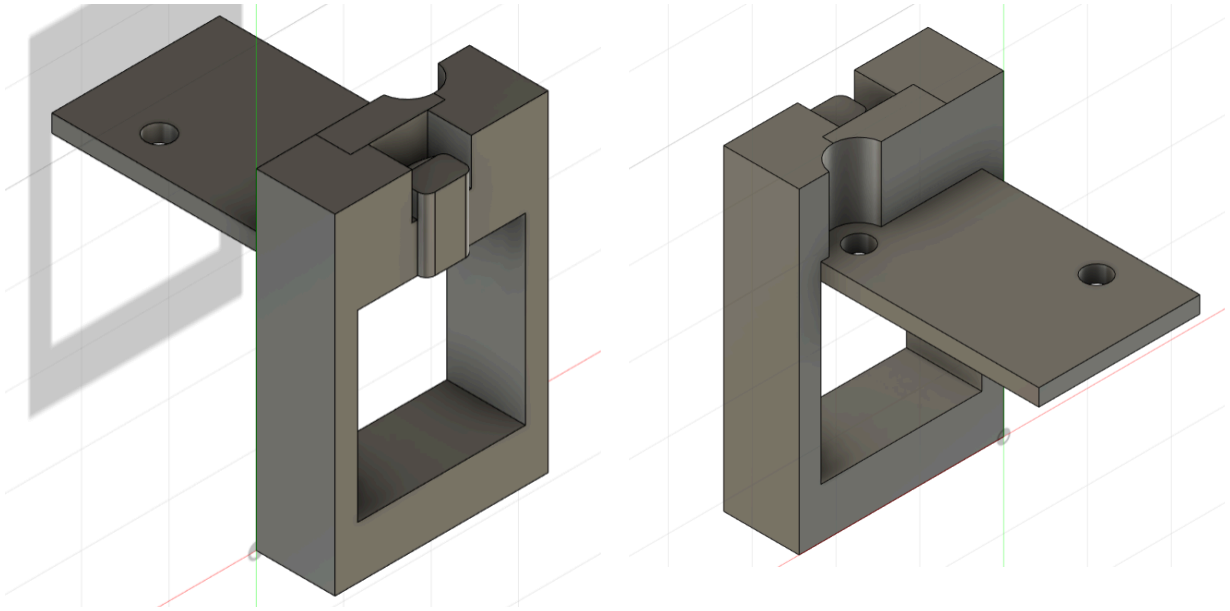


Figure 13.1.3.1: Blaster Solenoid Mount 3D Views

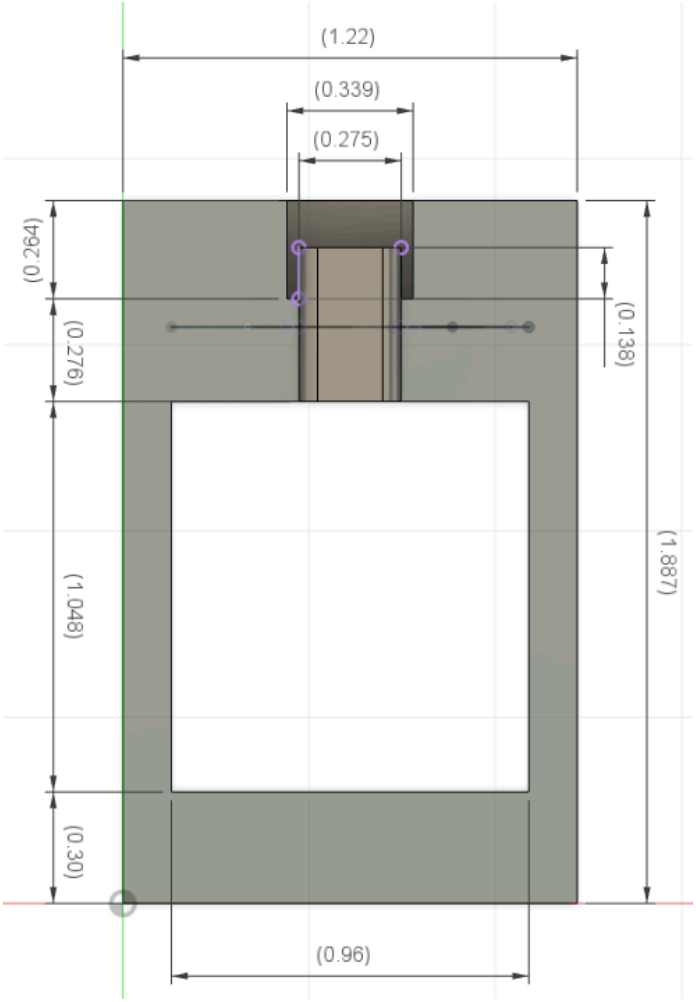


Figure 13.1.3.2: Blaster Solenoid Mount Front Drawing

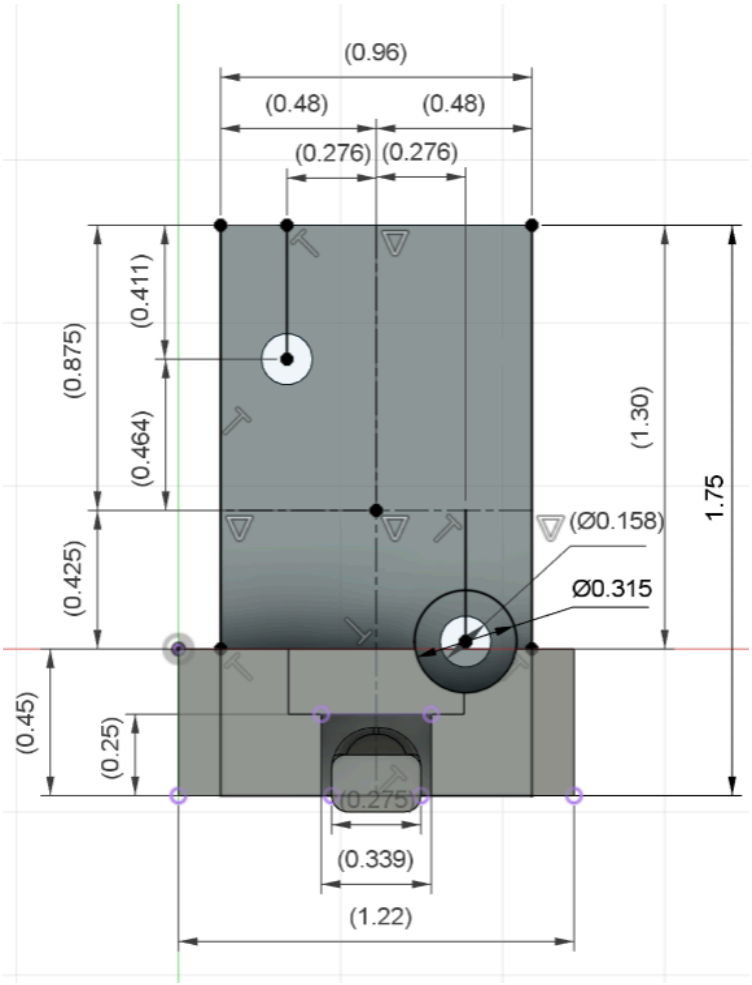


Figure 13.1.3.3: Blaster Solenoid Mount Top Drawing

13.1.4: Slide Spring Mount

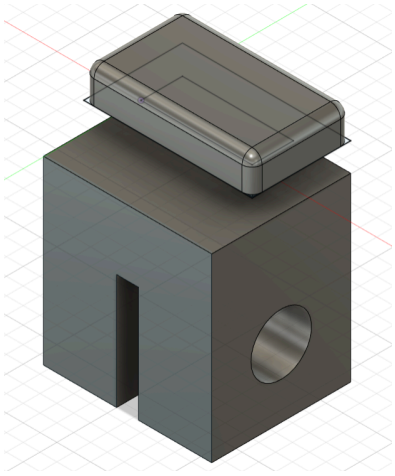


Figure 13.1.4.1: Blaster Slide Spring Mount 3D View

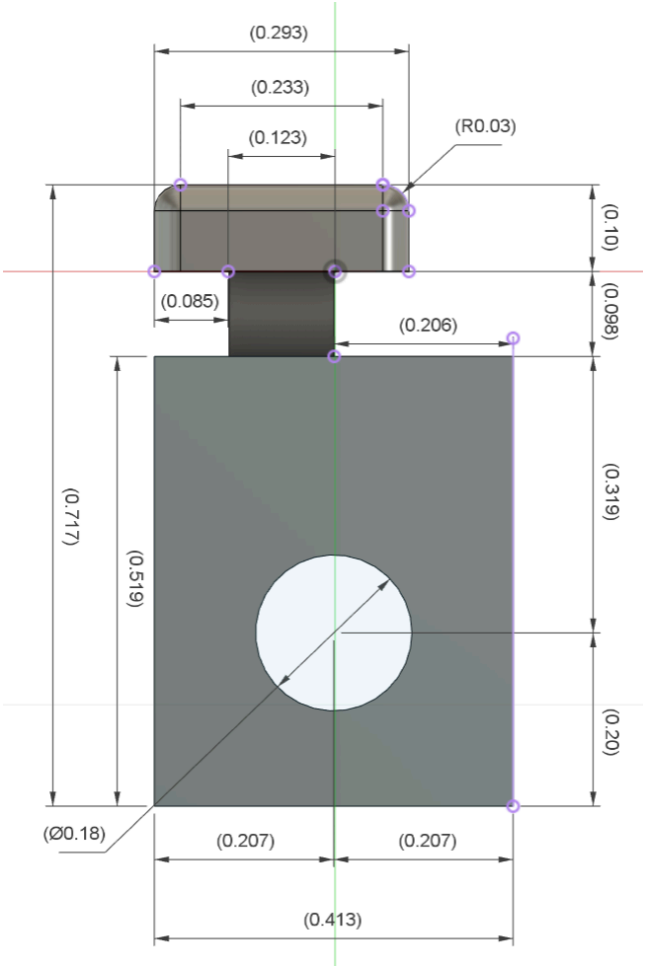


Figure 13.1.4.2: Blaster Slide Spring Mount Side Drawing

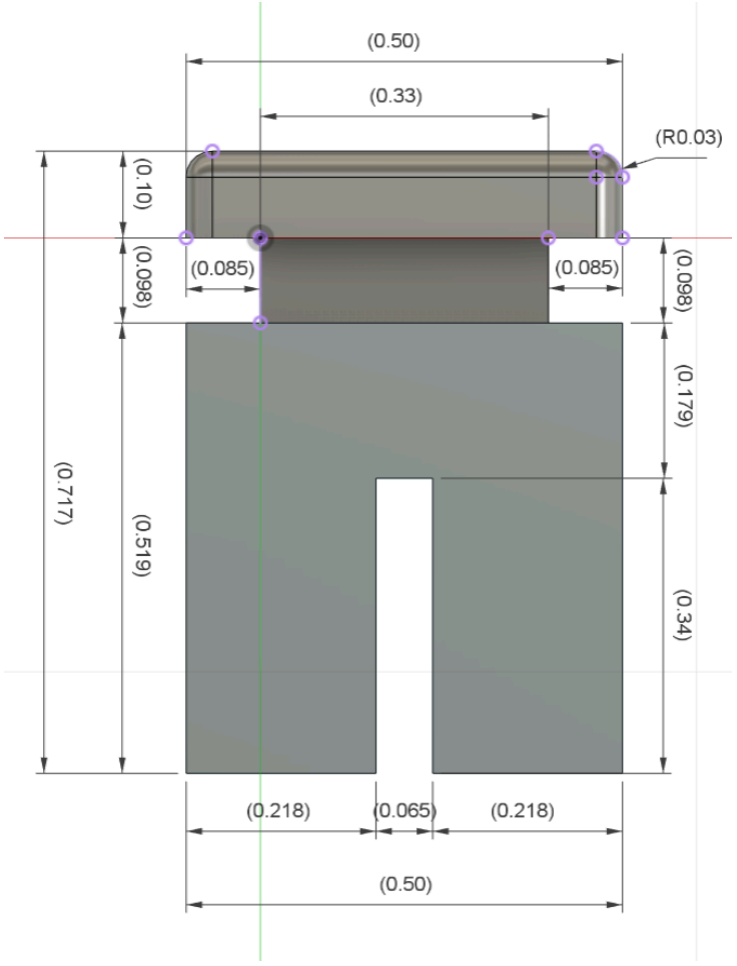


Figure 13.1.4.3: Blaster Slide Spring Mount Top Drawing

13.2: Target System

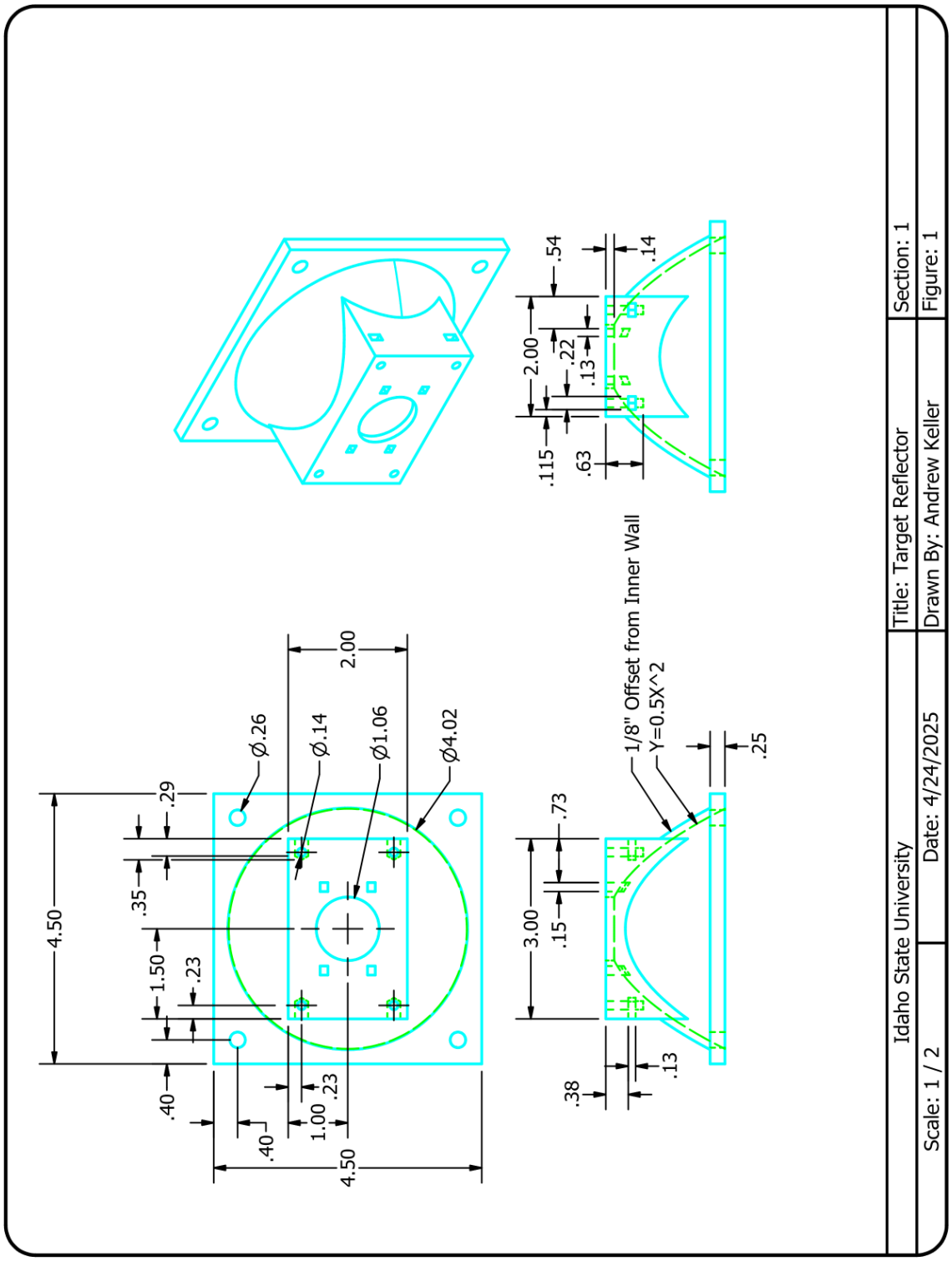


Figure 13.2.1: Target Reflector Mechanical Drawing

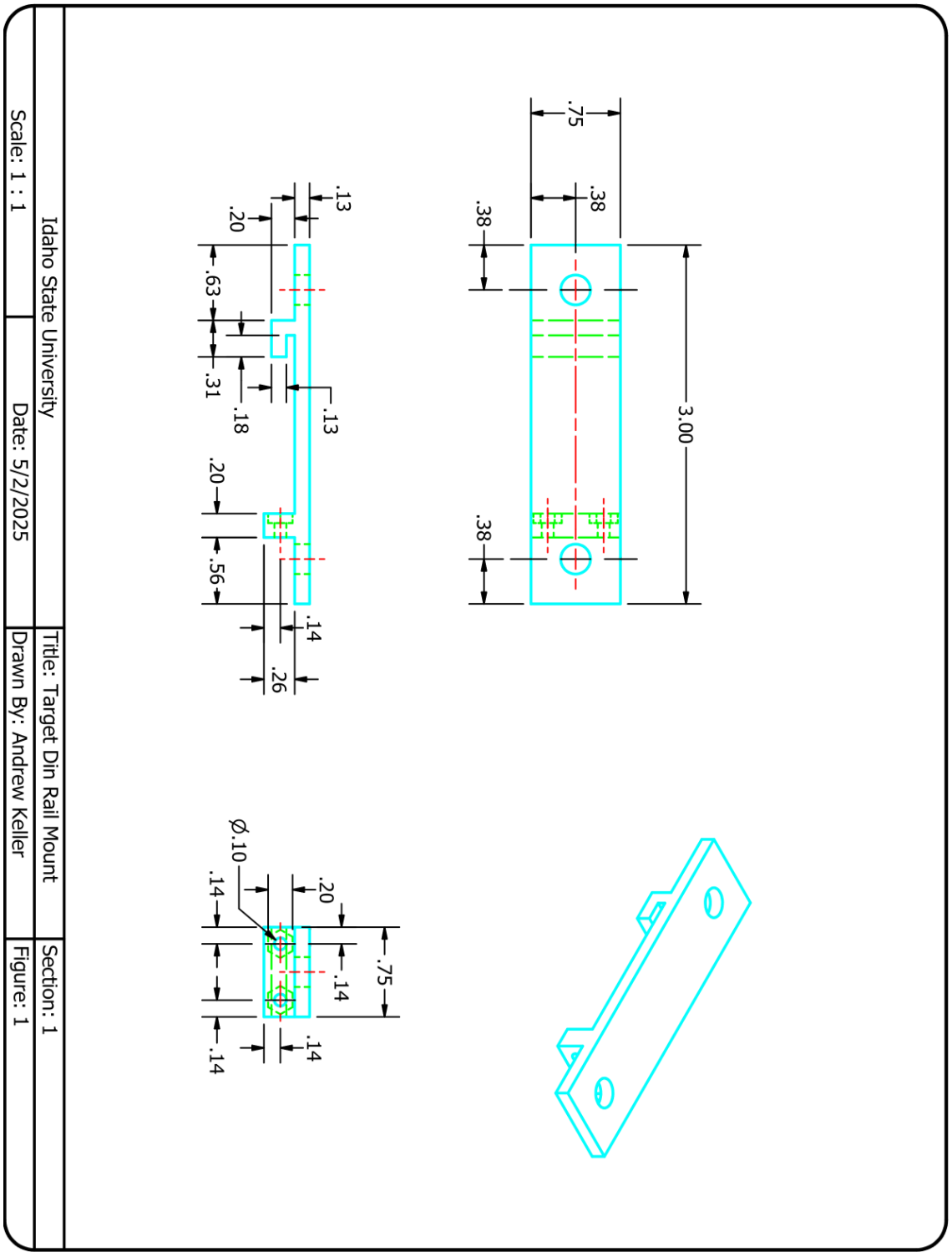


Figure 13.2.2: Target Reflector Din Rail Mount Mechanical Drawing

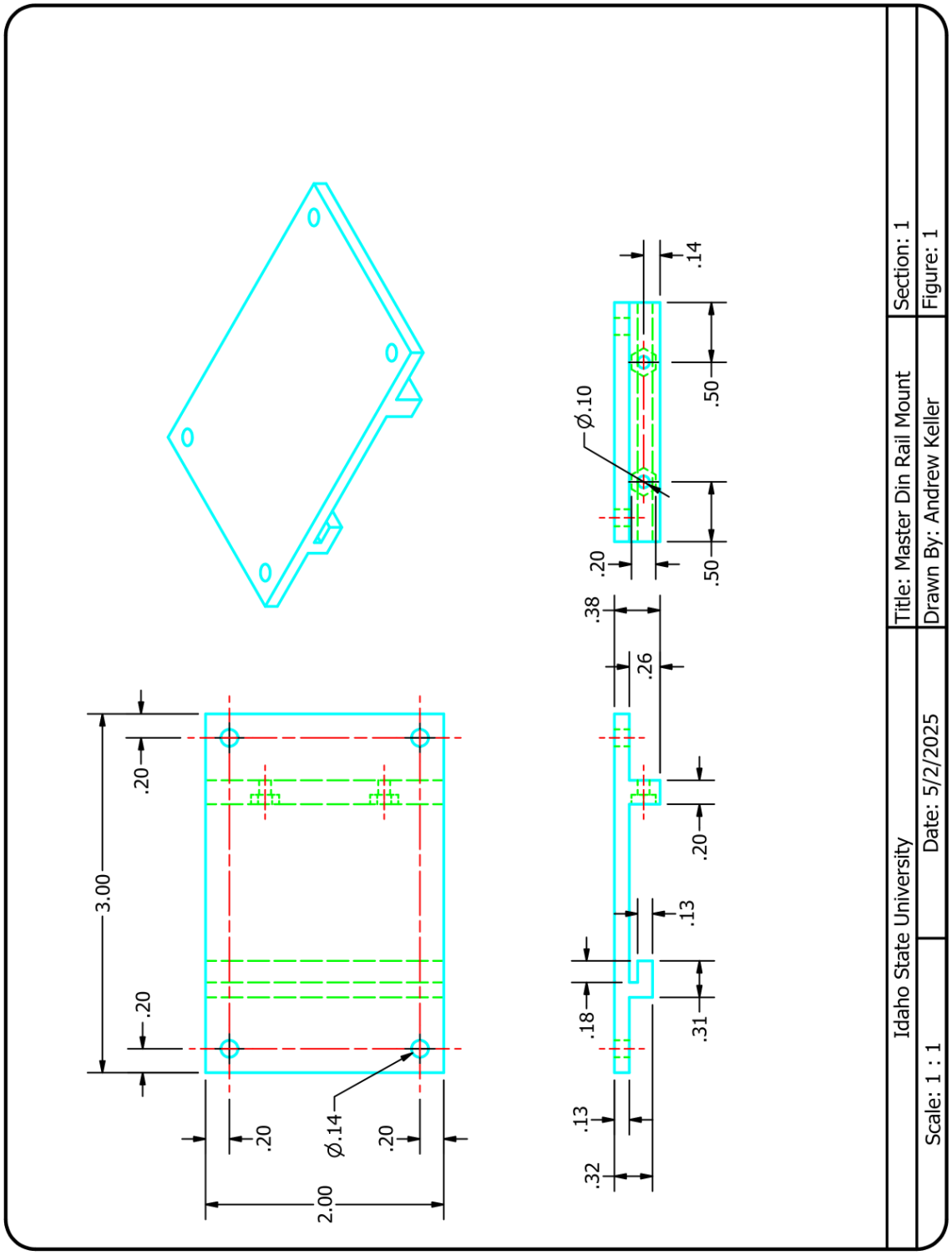


Figure 13.2.3: Target Master Din Rail Mount Mechanical Drawing

14: Appendix D (Flowcharts)

14.1: Blaster

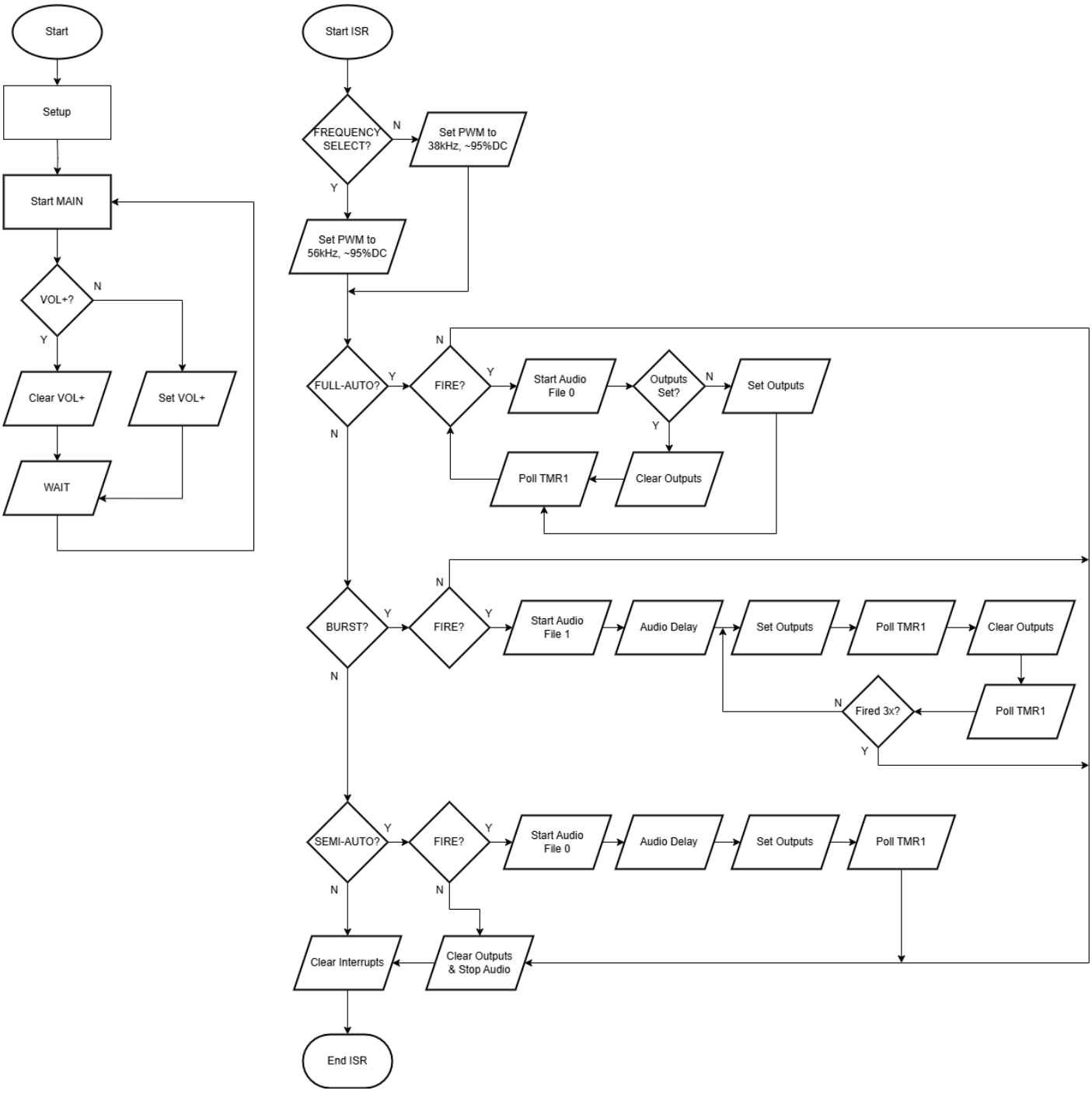


Figure 14.1.1: Blaster Program Flowchart

14.2: Target Master

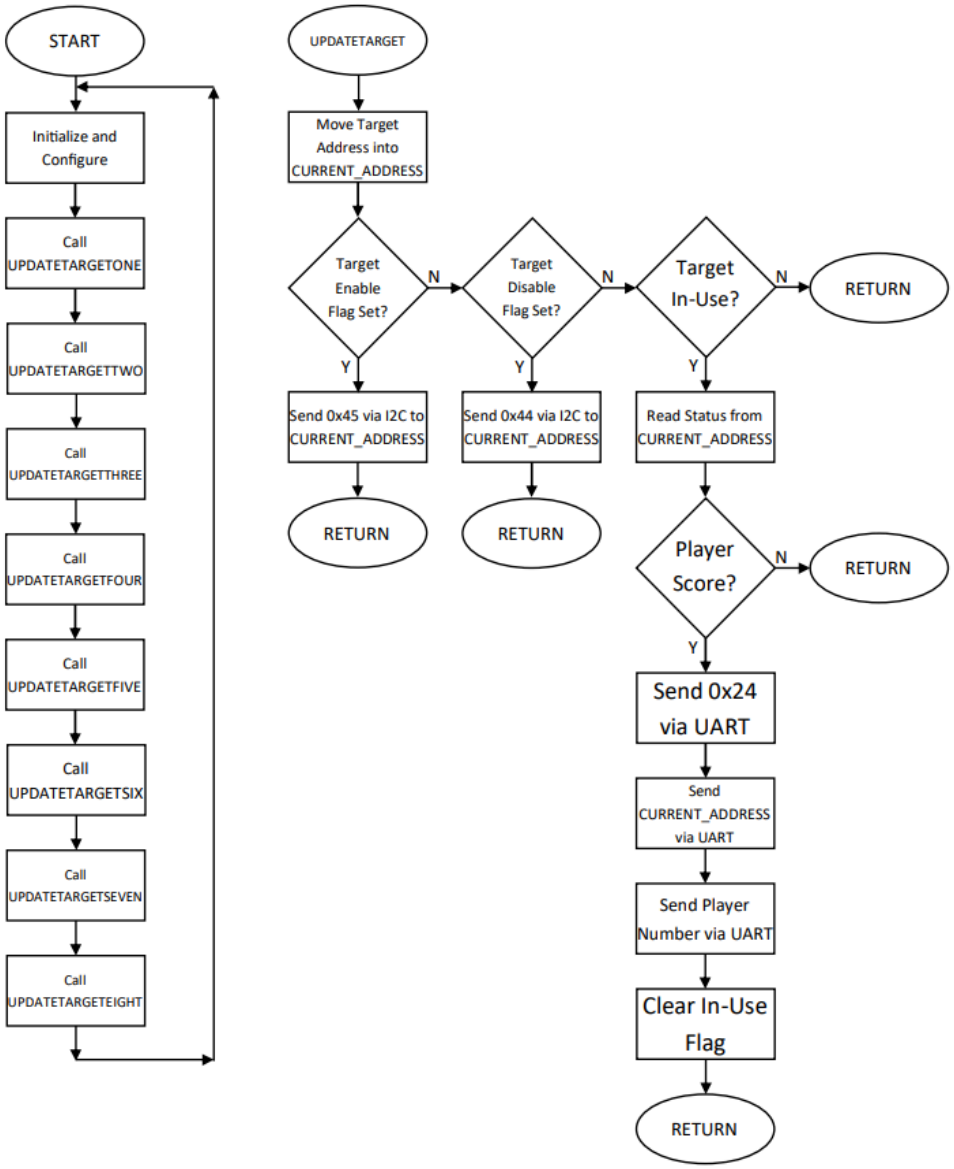


Figure 14.2.1: Target Master Program Main Flowchart

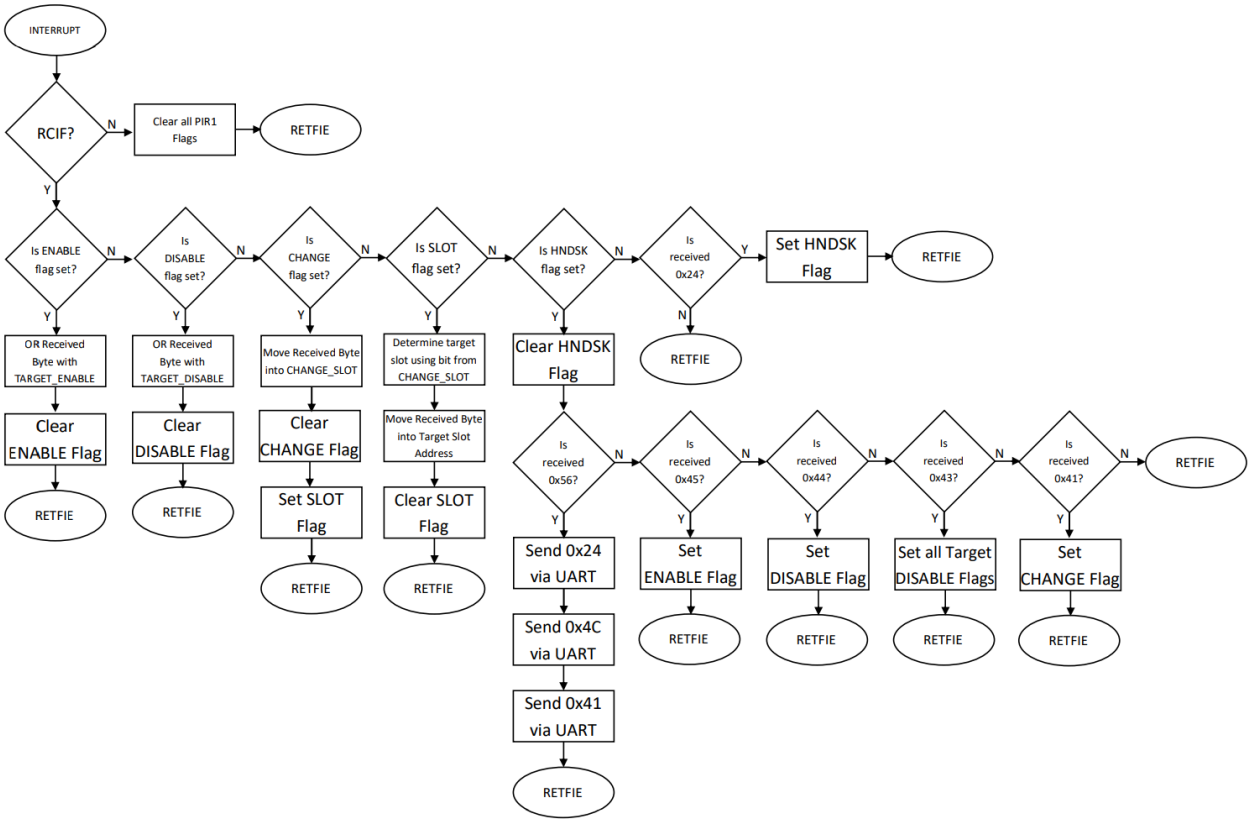


Figure 14.2.2: Target Master Interrupt Service Routine Flowchart

14.3: Target Slave

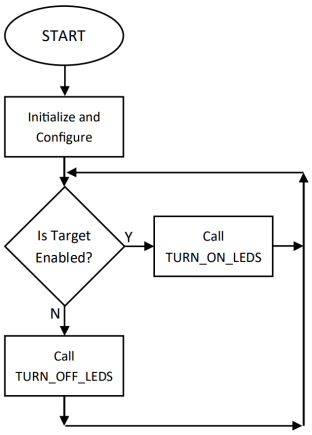


Figure 14.3.1: Target Slave Program Main Flowchart

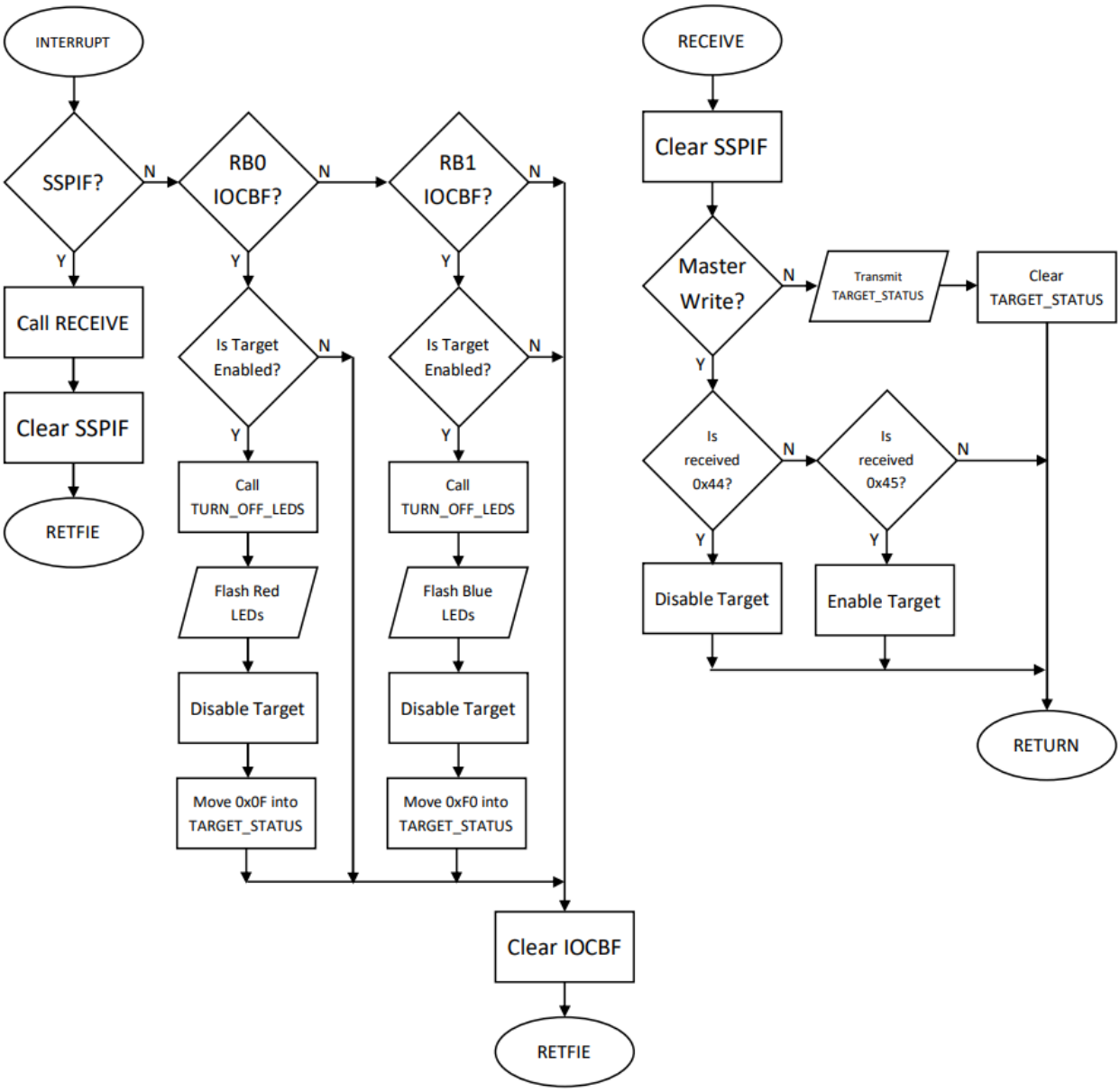


Figure 14.3.2: Target Slave Interrupt Service Routine Flowchart

15: Appendix E (Port Bitmaps)

15.1: Blaster

Table 15.1.1: Blaster I/O Port Bitmap

I/O Port	B7	B6	B5	B4	B3	B2	B1	B0
PORTA	TRIG7	TRIG6	TRIG5	TRIG4	TRIG3	TRIG2	TRIG1	TRIG0
PORTB	Unused	Unused	Unused	SEMI-AUTO	BURST	FULL-AUTO	FREQUENCY	FIRE
PORTC	VOL+	TRIG10	TRIG9	UNUSED	SOLENOID	LASER1	LASER2	TRIG8

Table 15.1.2: Blaster General Register Bitmap

Register	Address	B7	B6	B5	B4	B3	B2	B1	B0
AUDIO_DELAY	0x20	COUNT FOR AUDIO IN-LINE DELAY FOR SEMI-AUTO AND BURST							
AUDIO_DELAY2	0x21	COUNT FOR NESTED IN-LINE DELAY TIMING FOR SEMI-AUTO							
VOLUME_COUNT	0x22	COUNT FOR VOLUME IN-LINE DELAY TIMING							
VOLUME_COUNT2	0x23	COUNT FOR VOLUME NESTED IN-LINE DELAY TIMING							

15.2: Target Master

Table 15.2.1: Target Master I/O Port Bitmap

I/O Port	B7	B6	B5	B4	B3	B2	B1	B0
PORTA	Unused	Unused	Unused	Unused	Unused	Unused	Unused	Unused
PORTB	Unused	Unused	Unused	Unused	Unused	Unused	Unused	Unused
PORTC	UART RX	UART TX	Unused	I2C SDA	I2C SCL	Unused	Unused	Unused

Table 15.2.2: Target Master General Register Bitmap

Register	Address	B7	B6	B5	B4	B3	B2	B1	B0
UART_MODE	0x20	I2C ADDRESS CURRENTLY BEING USED							
UART_RX_BYTE	0x21	DATA TO BE WRITTEN TO I2C							
ENABLE_TARGET	0X22	DATA RECEIVED THROUGH I2C							
UART_MODE	0x30	HNDSK	VERIFY	PLAYER	ENABLE	DISABLE	CHANGE	SLOT	-
UART_RX_BYTE	0x31	TEMPORARY STORAGE FOR RECEIVED UART BYTE							
ENABLE_TARGET	0X32	EN_1	EN_2	EN_3	EN_4	EN_5	EN_6	EN_7	EN_8
DISABLE_TARGET	0X33	DIS_1	DIS_2	DIS_3	DIS_4	DIS_5	DIS_6	DIS_7	DIS_8
INUSE_TARGET	0X34	TGT_1	TGT_2	TGT_3	TGT_4	TGT_5	TGT_6	TGT_7	TGT_8
SCORE_ADDRESS	0X35	I2C ADDRESS OF THE HIT TARGET FOR UARTSENDSCORE							
SCORE_PLAYER	0X36	PLAYER NUMBER OF SCORING PLAYER FOR UARTSENDSCORE							
CHANGE_SLOT	0X37	SL_1	SL_2	SL_3	SL_4	SL_5	SL_6	SL_7	SL_8
TGT_ONE_ADDRESS	0x41	I2C ADDRESS OF ENABLED TARGET							
TGT_TWO_ADDRESS	0x42	I2C ADDRESS OF ENABLED TARGET							
TGT_THREE_ADDRESS	0x43	I2C ADDRESS OF ENABLED TARGET							
TGT_FOUR_ADDRESS	0x44	I2C ADDRESS OF ENABLED TARGET							
TGT_FIVE_ADDRESS	0x45	I2C ADDRESS OF ENABLED TARGET							
TGT_SIX_ADDRESS	0x46	I2C ADDRESS OF ENABLED TARGET							
TGT_SEVEN_ADDRESS	0x47	I2C ADDRESS OF ENABLED TARGET							
TGT_EIGHT_ADDRESS	0x48	I2C ADDRESS OF ENABLED TARGET							

15.3: Target Slave

Table 15.3.1: Target Slave I/O Port Bitmap

I/O Port	B7	B6	B5	B4	B3	B2	B1	B0
PORTA	I2C Address Switch	I2C Address Switch	I2C Address Switch	I2C Address Switch	I2C Address Switch	I2C Address Switch	I2C Address Switch	Unused Switch
PORTB	Unused	Unused	Unused	RGB LED (Blue)	RGB LED (Red)	RGB LED (Green)	56kHz IR Receiver	38kHz IR Receiver
PORTC	Unused	Unused	Unused	I2C SDA	I2C SCL	Unused	Unused	Unused

Table 15.3.2: Target Slave General Register Bitmap

Register	Address	B7	B6	B5	B4	B3	B2	B1	B0
RECEIVED_DATA	0x20	DATA RECEIVED THROUGH I2C							
TARGET_STATUS	0x21	CURRENT TARGET STATUS. 0x0F IS P1 SCORE, 0xF0 IS P2 SCORE.							
ACTIVE	0x22	-	-	-	-	-	-	-	ACTIVE

16: Appendix F (Code)

16.1: Blaster

16.1.1: Blaster Assembly Code

```
;*******************************************************************************
;                                                                              *
;  Filename:        LA_BLASTER.asm                                             *
;  Date:    November 5, 2024                                                  *
;  File Version:  2                                                           *
;  Author:          Alex Wheelock                                             *
;  Company:         Idaho State University                                    *
;                                                                              *
;  Description:   Assembly file for Laser Arcade blaster. A gun that can fire  *
;                   a laser at two frequencies (38kHz & 56kHz), with firing mode  *
;                   select (single-shot, three round burst, continuous). Different  *
;                   frequencies will be used to determine between player 1 (38kHz)  *
;                   and 2 (56kHz) for scoring.                                 *
;                                                                              *
;                   Uses CCP1 (RC2) for laser PWM, RC3 for solenoid control. RB0 for*
;                   trigger, RB1 for frequency select and RB4:2 used to select from *
;                   the firing modes. PORTA is used for trigger pins on the audio  *
;                   board, as are RC6, & RC1:0. RC7 continuously toggles in main to *
;                   ensure that the volume on the audio board is maxed out.    *
;                                                                              *
;                   The PWM output driver is enabled when the gun is fire, and  *
;                   disabled when not, or when cycling between "rounds/shots".  *
;                                                                              *
;*******************************************************************************

;*******************************************************************************
;                                                                              *
;  Revision History:                                                          *
;                                                                              *
;  1:       (NOV 2024) Got everything for the gun working the way that I think it  *
;           should with base features.                                         *
;                                                                              *
;  2:       (APRIL 3, 2025) Added an audio board, and connected all of PORTA, and  *
;           RC7,6,1, & 0 as triggers for the audio board, only RA0 & RA1 are being  *
;           used as triggers currently, RC7 toggles in MAIN to ensure the volume is  *
;           maxed out.                                                         *
;                                                                              *
;           Code was also updated to work properly with the fire select switch that  *
;           will be used in the final product.                                 *
;                                                                              *
;*******************************************************************************

                #include "BLASTER.inc"                  ; processor specific variable definitions
                #INCLUDE <BLASTER_SETUP.inc>            ; Custom setup file for the PIC16F883
micro-controller
                #INCLUDE <BLASTER_SUBROUTINES.inc>      ; File containing all used
subroutines
```

```
		LIST	p=16f1788				; list directive to define processor
		errorlevel -302,-207,-305,-206,-203	; suppress "not in bank 0" message,  Found label after column 1,
							; Using default destination of 1 (file),  Found call to macro in column 1

		; CONFIG1
; __config 0xC9A4
 __CONFIG _CONFIG1, _FOSC_INTOSC & _WDTE_OFF & _PWRTE_OFF & _MCLRE_OFF & _CP_OFF & _CPD_OFF & _BOREN_OFF & _CLKOUTEN_OFF & _IESO_OFF & _FCMEN_OFF
; CONFIG2
; __config 0xDFFF
 __CONFIG _CONFIG2, _WRT_OFF & _PLLEN_ON & _STVREN_ON & _BORV_LO & _LPBOR_OFF & _LVP_OFF



;*****************************************
;Interrupt Vectors
;*****************************************
		ORG	H'000'			;BEGINNING OF CODE
		GOTO	SETUP				;
		ORG	H'004'			;INTERRUPT LOCATION
		GOTO	INTERRUPT			;INTERRUPT OCCURRED, RUN THROUGH ISR

;*****************************************
;SETUP ROUTINE
;*****************************************
SETUP
		CALL	INITIALIZE			;CALL SETUP INCLUDE FILE TO INITIALIZE PIC
		GOTO	MAIN				;START MAIN CODE

;*****************************************
;INTERRUPT SERVICE ROUTINE
;*****************************************
INTERRUPT
		BANKSEL		PORTB
		BTFSC	PORTB,1				;TEST FREQUENCY SELECT SWITCH
		GOTO	SET_56KHZ			;RB1 IS HIGH, SET PWM FREQUENCY TO 56kHz
		GOTO	SET_38KHZ			;RB1 IS LOW, SET PWM FREQUENCY TO 38kHz

	TEST_MODE
		BANKSEL		PORTB
		BTFSC	PORTB,2				;DETERMINE IF IN CONTINUOUS/FULL-AUTO MODE
		GOTO	FIRE_CONTINUOUS		;GUN IS IN CONTINOUS/FULL-AUTO MODE, CHECK IF READY TO FIRE
		BTFSC	PORTB,3				;GUN IS NOT IN CONTINUOUS/FULL-AUTO MODE, TEST IF IT IS IN BURST-FIRE MODE
		GOTO	FIRE_BURST			;GUN IS IN BURST-FIRE MODE, CHECK IF READY TO FIRE
		BTFSC	PORTB,4				;GUN IS IN NEITHER FULL-AUTO OR BURST-FIRE MODES, DETERMINE IF IT IS IN SEMI-AUTO MODE
```

```
                GOTO    FIRE_SEMI                       ;GUN IS NOT IN CONTINUOUS OR BURST-FIRE
MODE, DEFAULT TO SEMI-AUTO MODE, TEST IF READY TO FIRE
        GOBACK
                BANKSEL         IOCBF
                CLRF    IOCBF                           ;CLEAR PORTB IOC FLAGS
                BCF     INTCON,IOCIF                    ;CLEAR IOCIF
                BANKSEL         PIR1
                BCF     PIR1,0                          ;CLEAR TMR1IF
                RETFIE                                  ;RETURN TO MAIN, RE-ENABLE GIE

;*****************************************
;Main Code
;*****************************************
MAIN                                                    ;CONSTANTLY PULSE THE VOL+ WHILE IN
MAIN TO ENSURE THE VOLUME IS MAXED OUT
                BANKSEL         PORTC
                BTFSC   PORTC,7                         ;DETERMINE IF THE VOL+ OUTPUT IS HIGH
OR LOW
                GOTO    _CLEAR                          ;OUTPUT IS HIGH
        _SET                                            ;OUTPUT IS LOW
                BSF     PORTC,7                         ;OUTPUT IS LOW, SET VOL+
                GOTO    WAIT
        _CLEAR
                BCF     PORTC,7                         ;OUTPUT IS HIGH, CLEAR VOL+
        WAIT                                            ;\
                MOVLW           0x7F                    ;\\
                MOVWF           VOLUME_COUNT                    ;\\\
          VOLUME_LOOP1                                  ;\\\\
                CLRF    VOLUME_COUNT2           ;\\\\\ GIVE A DELAY ON THE PW/PS TO ENSURE
THAT THE AUDIO BOARD HAS TIME
            VOLUME_LOOP2                                ;///// TO TRIGGER OFF OF IT, BECAUSE IT IS
VERY SLOW.
                DECFSZ          VOLUME_COUNT2                   ;////
                GOTO    VOLUME_LOOP2            ;///
                DECFSZ          VOLUME_COUNT                    ;//
                GOTO    VOLUME_LOOP1            ;/
                GOTO    MAIN
END

;*****************************************
;END PROGRAM DIRECTIVE
;*****************************************
```

16.1.2: Blaster Subroutine Code

```
;*******************************************************************************
;                                                                               *
;  Filename:          BLASTER_SUBROUTINES.inc                                   *
;  Date:     November 5, 2024                                                   *
;  File Version:   2                                                            *
;  Author:            Alex Wheelock                                             *
;  Company:           Idaho State University                                    *
;  Description:    Contains all subroutines needed for the Laser Arcade project.  *
;                     Subroutines include the ability to swap between the two player  *
;                     PWM frequencies of 56kHz and 38kHz, and the ability to shoot in *
;                     three different modes (single-shot, burst, & continuous), to    *
;                     move a solenoid and shoot the laser with the press of a button. *
;                                                                               *
;*******************************************************************************

;*******************************************************************************
;                                                                               *
;  Revision History:                                                            *
;                                                                               *
;  1:       (NOV 2024) Got everything for the gun working the way that I think it    *
;           should with base features.                                           *
;                                                                               *
;  2:       (APRIL 3, 2025) Added an audio board, and connected all of PORTA, and       *
;           RC7,6,1, & 0 as triggers for the audio board, only RA0 & RA1 are being    *
;           used as triggers currently, RC7 toggles in MAIN to ensure the volume is   *
;           maxed out.                                                           *
;                                                                               *
;           Code was also updated to work properly with the fire select switch that   *
;           will be used in the final product.                                   *
;                                                                               *
;*******************************************************************************

SUBROUTINES_CODE CODE

   FIRE_CONTINUOUS
                BANKSEL         PORTB
                BTFSS     PORTB,0                   ;TEST IF THE TRIGGER IS BEING HELD DOWN
                GOTO      STOP_FIRING                     ;TRIGGER IS NOT BEING HELD DOWN,
STOP FIRING
                BCF       PORTA,0                   ;ENABLE TRIGGER1 TO PLAY SOUND FROM
THE AUDIO BOARD
                BANKSEL         TRISC
                BTFSC     TRISC,2                   ;TEST IF OUTPUT DRIVER FOR PWM IS SET
OR NOT (FOR ALTERNATING BETWEEN SET/RESET)
                GOTO      FIRE                ;OUTPUT DRIVER WAS NOT ENABLED, ENABLE IT
                GOTO      _RESET                    ;OUTPUT DRIVER WAS ENABLED, DISABLE IT

        FIRE
                BCF       TRISC,2                   ;ENABLE OUTPUT DRIVER FOR PWM
                BANKSEL         PORTC
                BSF       PORTC,3                   ;ENERGIZE THE SOLENOID
                GOTO      CONTINUOUS_CHECK_TIMER          ;POLL TIMER 1 FOR TIMING
```

```
        _RESET ;SETS
                BSF     TRISC,2                 ;DISABLE OUTPUT DRIVER FOR PWM
                BANKSEL         PORTC
                BCF     PORTC,3                 ;DE-ENERGIZE THE SOLENOID
                GOTO    CONTINUOUS_CHECK_TIMER
        CONTINUOUS_CHECK_TIMER
                BCF     PIR1,0                  ;CLEAR TMR1IF
        POLL_TMR1_CONTINUOUS
                BTFSS   PIR1,0                  ;POLL TMR1IF UNTIL SET, THEN CONTINUE
THE CYCLE
                GOTO    POLL_TMR1_CONTINUOUS    ;TMR1 NOT DONE
                GOTO    FIRE_CONTINUOUS         ;TMR1 DONE, REPEAT CYCLE

  STOP_FIRING
                BANKSEL         TRISC
                BTFSS   TRISC,2                 ;DETERMINE IF PWM OUTPUT DRIVER IS
ALREADY DISABLED
                BSF     TRISC,2                 ;OUTPUT DRIVER NOT DISABLED, DISABLE
OUTPUT DRIVER
                BANKSEL         PORTC
                BCF     PORTC,3                 ;NO MATTER WHAT, VERIFY THAT THE
SOLENOID IS DE-ENERGIZED
                ;BSF    PORTA,0                 ;RESET TRIG0 OF THE AUDIO BOARD
                ;BSF    PORTA,1                 ;RESET TRIG1 OF THE AUDIO BOARD
                MOVLW           0xFF            ;\
                MOVWF           PORTA                   ;\\ SET ALL TRIG PINS LOW
                MOVLW           0xF1            ;// ENSURE NO FALSE TRIGGERS
                MOVWF           PORTC                   ;/
                GOTO    GOBACK                  ;LEAVE ISR

  FIRE_SEMI
                BANKSEL         IOCBF
                BTFSS   IOCBF,0                 ;DETERMINE IF A TRIGGER PRESS WAS
RECORDED
                GOTO    GOBACK                  ;DO NOTHING IF NO TRIGGER PRESS
                BANKSEL         PORTB
                BTFSS   PORTB,0                 ;VERIFY THAT THE BUTTON WAS PRESSED,
AND THAT THE INTERRUPT WAS NOT FROM A TRIGGER RELEASE
                GOTO    STOP_FIRING                     ;TRIGGER WAS BEING RELEASED FOR
THE INTERRUPT, CANCEL EVERYTHING
                BANKSEL         PORTC
                BCF     PORTA,0                 ;ENABLE TRIG0 TO PLAY AUDIO
                MOVLW           0x2F            ;\
                MOVWF           AUDIO_DELAY                     ;\\
        LOOP1                                   ;\\\
                CLRF    AUDIO_DELAY2            ;\\\\
        LOOP2                                   ;||||RUN THROUGH A DELAY TO ENSURE THAT
AUDIO IS PLAYED WHEN FIRING IN THE SEMI-AUTO FIRING MODE
                DECFSZ          AUDIO_DELAY2
;////(-------------------------------------------------------------------------------)
                GOTO    LOOP2                   ;/// (NOTE: THE AUDIO BOARD IF VERY SLOW
AND AS A RESULT IT NEEDS EXTRA TIME TO GET GOING.)
                DECFSZ          AUDIO_DELAY                     ;//  (     AS A RESULT OF
CONSTANTLY TOGGLING IN SEMI-AUTO, IT CANNOT KEEP UP WITH THE   )
```

```
                GOTO    LOOP1                   ;/  (    FIRING, WHEN THE BLASTER TRIGGER
IS SPAMMED.                     )
                BANKSEL         TRISC                   ;
(-------------------------------------------------------------------------------)
                BCF     TRISC,2                 ;ENABLE OUTPUT DRIVER
                BANKSEL         PORTC
                BSF     PORTC,3                 ;ENERGIZE SOLENOID
                CLRF    TMR1L                   ;/RESET TMR1 FOR FULL TIMING
                CLRF    TMR1H                   ;\
                BCF     PIR1,0                  ;CLEAR TMR1IF
        POLL_TMR1_SEMI
                BTFSS   PIR1,0                  ;POLL TMR1IF UNTIL COMPLETE
                GOTO    POLL_TMR1_SEMI          ;TMR1 NOT DONE
                BANKSEL         PORTC
                BCF     PORTC,3                 ;TMR1 DONE, DE-ENERGIZE SOLENOID
                GOTO    STOP_FIRING                     ;STOP FIRING

  FIRE_BURST
                BANKSEL         IOCBF
                BTFSS   IOCBF,0                 ;DETERMINE IF A TRIGGER PRESS WAS
RECORDED
                GOTO    GOBACK                  ;DO NOTHING IF NO TRIGGER PRESS
                BANKSEL         PORTB
                BTFSS   PORTB,0                 ;VERIFY THAT THE BUTTON WAS PRESSED,
AND THAT THE INTERRUPT WAS NOT FROM A TRIGGER RELEASE
                GOTO    STOP_FIRING                     ;TRIGGER WAS BEING RELEASED FOR
THE INTERRUPT, CANCEL EVERYTHING
                BANKSEL         PORTC
                BCF     PORTA,1                 ;ENABLE TRIGGER1 TO PLAY THE BURST
AUDIO FROM THE AUDIO BOARD
                CLRF    AUDIO_DELAY             ;\
        LOOP_AUDIO_DELAY                        ;\\PLAY A SHORT DELAY TO KEEP THE BURST
AUDIO RELATIVELY IN SYNC WITH THE SHOTS
                DECFSZ          AUDIO_DELAY                     ;//
                GOTO    LOOP_AUDIO_DELAY                ;/
                BANKSEL         TRISC
                BCF     TRISC,2                 ;ENABLE OUTPUT DRIVER FOR PWM
                BANKSEL         PORTC
                BSF     PORTC,3                 ;ENERGIZE THE SOLENOID
                CLRF    TMR1L                   ;\RESET TMR1 FOR FULL TIMING
                CLRF    TMR1H                   ;/
                BCF     PIR1,0                  ;CLEAR TMR1IF
        POLL_TMR1_BURST1
                BTFSS   PIR1,0                  ;POLL TMR1
                GOTO    POLL_TMR1_BURST1                ;TMR1 NOT DONE
                BANKSEL         TRISC
                BSF     TRISC,2                 ;TMR1 IS DONE, DISABLE PWM OUTPUT
DRIVER FOR RESET
                BANKSEL         PORTC
                BCF     PORTC,3                 ;DE-ENERGIZE SOLENOID
                BCF     PIR1,0                  ;CLEAR TMR1IF
        POLL_TMR1_BURST2
                BTFSS   PIR1,0                  ;POLL TMR1
                GOTO    POLL_TMR1_BURST2
                BANKSEL         TRISC
```

```
                BCF     TRISC,2                 ;ENABLE OUTPUT DRIVER FOR PWM
(SECOND SHOT)
                BANKSEL         PORTC
                BSF     PORTC,3                 ;ENERGIZE THE SOLENOID
                BCF     PIR1,0                  ;CLEAR TMR1IF
        POLL_TMR1_BURST3
                BTFSS   PIR1,0                  ;POLL TMR1
                GOTO    POLL_TMR1_BURST3
                BANKSEL         TRISC
                BSF     TRISC,2                 ;TMR1 IS DONE, DISABLE PWM OUTPUT
DRIVER FOR RESET
                BANKSEL         PORTC
                BCF     PORTC,3                 ;DE-ENERGIZE SOLENOID
                BCF     PIR1,0                  ;CLEAR TMR1IF
        POLL_TMR1_BURST4
                BTFSS   PIR1,0                  ;POLL TMR1
                GOTO    POLL_TMR1_BURST4
                BANKSEL         TRISC
                BCF     TRISC,2                 ;ENABLE OUTPUT DRIVER FOR PWM
(THIRD/FINAL SHOT)
                BANKSEL         PORTC
                BSF     PORTC,3                 ;ENERGIZE THE SOLENOID
                BCF     PIR1,0                  ;CLEAR TMR1IF
        POLL_TMR1_BURST5
                BTFSS   PIR1,0                  ;POLL TMR1
                GOTO    POLL_TMR1_BURST5
                BCF     PORTC,3                 ;DE-ENERGIZE SOLENOID
                GOTO    STOP_FIRING


  SET_38KHZ
                BANKSEL         CCPR1L
                MOVLW           0x19            ;SET PW TIME FOR ~95% DUTY CYCLE FOR
MAX BRIGHTNESS
                MOVWF           CCPR1L
                BANKSEL         PR2
                MOVLW           0x19            ;SET PR2 TO 25 FOR 38kHz PERIOD
                MOVWF           PR2
                BANKSEL         TMR2
                CLRF    TMR2                    ;RESET TMR2
                GOTO    TEST_MODE
  SET_56KHZ
                BANKSEL         CCPR1L
                MOVLW           0x11            ;SET PW TIME FOR ~95% DUTY CYCLE FOR
MAX BRIGHTNESS
                MOVWF           CCPR1L
                BANKSEL         PR2
                MOVLW           0x11            ;SET PR2 TO 17 FOR 56kHz PERIOD
                MOVWF           PR2
                BANKSEL         TMR2
                CLRF    TMR2                    ;RESET TMR2
                GOTO    TEST_MODE
```

16.1.3: Blaster Setup Code

```
;*******************************************************************************
;                                                                              *
;  Filename:          BLASTER_SETUP.inc                                        *
;  Date:     November 5, 2024                                                 *
;  File Version:   2                                                          *
;  Author:            Alex Wheelock                                           *
;  Company:           Idaho State University                                  *
;  Description:    Firmware for setting up a PIC16F1788 for the laser gun of the  *
;                     Laser ARCADE project.                                   *
;                                                                              *
;*******************************************************************************

;*******************************************************************************
;                                                                              *
;  Revision History:                                                           *
;                                                                              *
;  1:       (NOV 2024) Got everything for the gun working the way that I think it    *
;           should with base features.                                         *
;                                                                              *
;  2:       (APRIL 3, 2025) Added an audio board, and connected all of PORTA, and    *
;           RC7,6,1, & 0 as triggers for the audio board, only RA0 & RA1 are being   *
;           used as triggers currently, RC7 toggles in MAIN to ensure the volume is  *
;           maxed out.                                                         *
;                                                                              *
;           Code was also updated to work properly with the fire select switch that  *
;           will be used in the final product.                                 *
;                                                                              *
;*******************************************************************************

;================================================================================
================================================*
;                                                    PORTA BITMAP
                                *
;================================================================================
================================================*
;               |               |               |               |               |               |
        |               *
;       7       |       6       |       5       |       4       |       3       |       2       |
1       |       0       *
;---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------*
;DIGITAL        |DIGITAL        |DIGITAL        |DIGITAL        |DIGITAL        |DIGITAL
|DIGITAL        |DIGITAL        *
;OUTPUT                 |OUTPUT                 |OUTPUT                 |OUTPUT
|OUTPUT                 |OUTPUT                 |OUTPUT                 |OUTPUT                 *
;---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------*
;DRIVES TRIG7 |DRIVES TRIG6 |DRIVES TRIG5 |DRIVES TRIG4 |DRIVES TRIG3 |DRIVES TRIG2
|DRIVES TRIG1 |DRIVES TRIG0 *
;OF THE AUDIO           |OF THE AUDIO|OF THE AUDIO|OF THE AUDIO|OF THE AUDIO|OF THE AUDIO
        |OF THE AUDIO|OF THE AUDIO*
;BOARD                  |BOARD                  |BOARD                  |BOARD
|BOARD                  |BOARD                  |BOARD                  |BOARD                  *
;               |               |               |               |               |               |
        |               *
```

```
;-------------------------------------------------------------------------------------------------------------------*

;===================================================================================================
==================================================*
;                                                                PORTB BITMAP
                                  *
;===================================================================================================
==================================================*
;               |               |               |               |               |               |
        |               *
;       7       |       6       |       5       |       4       |       3       |       2       |
1       |       0       *
;---------------|---------------|---------------|---------------|---------------|---------------|---------------*
;DIGITAL        |DIGITAL        |DIGITAL        |DIGITAL        |DIGITAL        |DIGITAL
|DIGITAL        |DIGITAL        *
;INPUT          |INPUT          |INPUT          |INPUT          |INPUT          |INPUT          |INPUT
        |INPUT          *
;---------------|---------------|---------------|---------------|---------------|---------------|---------------*
;             USED FOR          |UNUSED                 |INPUT USED FOR       |INPUT USED FOR
|INPUT USED FOR       |INPUT USED FOR       |INPUT USED FOR       *
;            PROGRAMMING                |                 |SEMI-AUTO MODE     |BURST-FIRE
|CONTINUOUS/ |THE FREQUENCY       |TRIGGER BUTTON     *
;              ONLY                 |                 |SELECT               |MODE SELECT|FULL-AUTO
MODE  |SELECT SWITCH       |                 *
;                                   |                 |                 |               |SELECT
|               |               *
;-------------------------------------------------------------------------------------------------------------------*

;===================================================================================================
==================================================*
;                                                                PORTC BITMAP
                                  *
;===================================================================================================
==================================================*
;               |               |               |               |               |               |
        |               *
;       7       |       6       |       5       |       4       |       3       |       2       |
1       |       0       *
;---------------|---------------|---------------|---------------|---------------|---------------|---------------*
;DIGITAL        |DIGITAL        |DIGITAL        |DIGITAL        |DIGITAL        |PWM            |PWM
        |DIGITAL        *
;OUTPUT         |OUTPUT         |OUTPUT         |OUTPUT
|OUTPUT         |OUTPUT         |OUTPUT         |OUTPUT         *
;---------------|---------------|---------------|---------------|---------------|---------------|---------------*
;DRIVES VOL+ |DRIVES TRIG10      |DRIVES TRIG9 |UNUSED           |OUTPUT USED TO
|OUTPUT USED TO       |OUTPUT USED TO       |DRIVES TRIG8 *
;OF THE AUDIO       |OF THE AUDIO|OF THE AUDIO|                |CONTROL THE
|CONTROL/DRIVE       |CONTROL/DRIVE       |OF THE AUDIO*
;BOARD              |BOARD               |BOARD              |               |SOLENOID
|LASER1              |LASER2              |BOARD              *
;               |               |               |               |               |               |
        |               *
;-------------------------------------------------------------------------------------------------------------------*
```

```
INITIALIZE_CODE       CODE
INITIALIZE

;*** INTOSC SETUP *******

                BANKSEL         OSCCON
                MOVLW           0xEA                    ;ENABLE INTOSC, SET TO 4MHz
                MOVWF           OSCCON
                BANKSEL         CLKRCON
                CLRF    CLKRCON                         ;DISABLE CLOCK REFERENCE

;*** SFR SETUP **********


;*** SET OPTION_REG: ****

                BANKSEL         OPTION_REG
                BCF     OPTION_REG, PS0         ;\\
                BCF     OPTION_REG, PS1         ; >>PRESCALER RATIO SET 1:1
                BCF     OPTION_REG, PS2         ;//
                BSF     OPTION_REG, PSA         ;PRESCALER ASSIGNED TO WDT
                BCF     OPTION_REG, TMR0SE              ;TMR0 EDGE SET RISING
                BCF     OPTION_REG, TMR0CS              ;TMR0 CLOCK SOURCE SET TO
INTERNAL
                BSF     OPTION_REG, INTEDG              ;RB0/INT SET TO RISING EDGE
                BSF     OPTION_REG, NOT_WPUEN   ;WEAK PULLUP ENABLED

;*** SET INTCON REG: ****

                BANKSEL INTCON
                BSF     INTCON, IOCIE           ;ENABLE IOC INTERUPT
                BCF     INTCON, INTE            ;DISABLE INT EXTERNAL INTERUPT
                BSF     INTCON, PEIE            ;ENABLE PERIPHERAL INTERUPTS
                BSF     INTCON, GIE             ;ENABLE ALL UNMASKED INTERUPTS

;*** SET PIE1-4 REG: *****

                BANKSEL         PIE1
                MOVLW           0x01
                MOVWF           PIE1                    ;DISABLE ALL PIE1 INTERUPTS, EXCEPT
TMR1IE
                BANKSEL         PIE2
                CLRF    PIE2                    ;DISABLE ALL PIE2 INTERUPTS
                BANKSEL         PIE3
                CLRF    PIE3                    ;DISABLE ALL PIE3 INTERUPTS
                BANKSEL         PIE4
                CLRF    PIE4                    ;DISABLE ALL PIE4 INTERUPTS

;*** CLEAR INTERUPT FLAGS ***

                BANKSEL         PIR1
                CLRF    PIR1                    ;CLEAR ALL PIR1 INTERUPT FLAGS
                BANKSEL         PIR2
                CLRF    PIR2                    ;CLEAR ALL PIR2 INTERUPT FLAGS
                BANKSEL         PIR3
```

```
                CLRF    PIR3                ;CLEAR ALL PIR3 INTERUPT FLAGS
                BANKSEL PIR4
                CLRF    PIR4                ;CLEAR ALL PIR4 INTERUPT FLAGS

;*** SET CCP1CON REG: **

                BANKSEL TRISC
                BSF     TRISC,2                     ;DISABLE PWM OUTPUT DRIVER FOR CCP1
SETUP
                BANKSEL CCP1CON
                MOVLW   0x0C
                MOVWF   CCP1CON                     ;SELECT PWM MODE FOR CCP1
                BANKSEL CCPR1L
                MOVLW   0x19                ;PW FOR INITIALIZING 95%DC AT 38kHz
                ;MOVLW  0x11                ;PW FOR INITIALIZING 95%DC AT 56kHz
                MOVWF   CCPR1L

;*** TIMER 1 SETUP *****

                BANKSEL T1CON
                MOVLW   0xA8                ;ENABLE & SELECT TMR1 LP OSCILLATOR,
SET PRESCALE TO 1:4, DISABLE TMR1 AT THIS POINT
                BANKSEL T1GCON
                CLRF    T1GCON              ;DISABLE TMR1 GATE

;*** TIMER 2 SETUP *****

                BANKSEL T2CON
                CLRF    T2CON
                BANKSEL PR2
                MOVLW   0x19                ;SET PR2 TO 25 FOR 38kHz PWM FREQUENCY
                ;MOVLW  0x11                ;SET PR2 TO 17 FOR 56kHz PWM FREQUENCY
                MOVWF   PR2

;*** ALTERNATE PIN FUNCTION *****

                BANKSEL APFCON1
                CLRF    APFCON1             ;SELECTS PORT MAPPING
                CLRF    APFCON2             ;SEE DATASHEET (pg 132-133) FOR MAPPING
WITH BOTH CLEARED

;*** PORT A SETUP **** PORT A SET AS LOGIC OUTPUT

                BANKSEL PORTA
                MOVLW   0xFF
                MOVWF   PORTA                       ;SET PORTA
                BANKSEL LATA
                MOVWF   LATA                ;CLEAR LATA
                BANKSEL TRISA
                MOVLW   0x00
                MOVWF   TRISA                       ;SET PORTA ALL AS OUTPUTS
                BANKSEL ODCONA
                CLRF    ODCONA              ;DISABLE PORTA OPEN-DRAIN
                BANKSEL ANSELA
                CLRF    ANSELA              ;SET PORTA ALL AS DIGITAL INPUT
```

```
                BANKSEL         SLRCONA
                CLRF    SLRCONA                 ;SET THE SLEW RATE FOR PORT A TO
MAXIMUM

;*** PORT B SETUP **** PORT B USED AS LOGIC INPUT

                BANKSEL         PORTB
                CLRF    PORTB                   ;CLEAR PORTB
                BANKSEL         LATB
                CLRF    LATB            ;CLEAR LATB
                BANKSEL         TRISB
                MOVLW           0xFF
                MOVWF           TRISB                   ;SET PORTB ALL AS INPUTS
                BANKSEL         ODCONB
                CLRF    ODCONB                  ;DISABLE PORTB OPEN-DRAIN
                BANKSEL         ANSELB
                CLRF    ANSELB                  ;MAKE PORTB DIGITAL INPUTS
                BANKSEL         SLRCONB
                CLRF    SLRCONB                 ;SET THE SLEW RATE FOR PORT B TO
MAXIMUM
                BANKSEL         IOCBP
                MOVLW           0x0F                    ;ENABLE PORTB IOC FOR POSITIVE EDGES
ON RB3:0
                MOVWF           IOCBP
                BANKSEL         IOCBN
                MOVLW           0x0E                    ;ENABLE PORTB IOC FOR NEGATIVE EDGES
ON RB3:1
                MOVWF           IOCBN
                BANKSEL         INLVLB
                CLRF    INLVLB                  ;SET PORTB INPUT THRESHOLD TO TTL
(IGNORED NOISE BETTER FOR TRIGGER INPUT)

;*** PORT C SETUP ****

                BANKSEL         PORTC
                MOVLW           0xF1
                MOVWF           PORTC                   ;SET ALL OF THE AUDIO BOARD
OUTPUTS RC7:5 & RC0, CLEAR THE LASER/SOLENOID OUTPUTS
                BANKSEL         LATC
                MOVWF           LATC                    ;CLEAR LATC
                BANKSEL         TRISC
                MOVLW           0x00
                MOVWF           TRISC                   ;SET PORTC ALL AS OUTPUTS
                BANKSEL         ODCONC
                CLRF    ODCONC                  ;DISABLE PORTC OPEN-DRAIN
                BANKSEL         ANSELC
                CLRF    ANSELC                  ;MAKE PORTC ALL DIGITAL I/O
                BANKSEL         SLRCONC
                CLRF    SLRCONC                 ;SET THE SLEW RATE FOR PORT C TO
MAXIMUM

;*** BOOT CODE ****

                BCF     INTCON,0
                BANKSEL         T1CON
```

```
		BSF	T1CON,0			;ENABLE TIMER 1
		BANKSEL	T2CON
		BSF	T2CON,2			;ENABLE TIMER 2
		BANKSEL	TRISC
		BSF	TRISC,2			;DISABLE OUTPUT DRIVER ON STARTUP
(WAS ENABLED ON STARTUP WITHOUT THIS)
		BANKSEL	PORTA
		RETURN				;RETURN TO .asm FILE
```

16.2: Target Master

16.2.1: Target Master Assembly Code

```
;*******************************************************************************
;                                                                              *
;  Filename:          LA_Master.asm                                            *
;  Date:     April 22, 2025                                                   *
;  File Version:   2                                                           *
;  Author:            Alex Wheelock and Andrew Keller                          *
;  Company:           Idaho State University                                   *
;  Description:    Assembly file for the master controller for a laser shooting  *
;                     arcade. Includes the ability to communicate with a computer  *
;                     via UART. Can control up to 8 targets simultaneously with  *
;                     various target slots. Each slot can be changed to a different  *
;                     I2C address using UART commands.                         *
;                                                                              *
;*******************************************************************************

;*******************************************************************************
;                                                                              *
;  Revision History:                                                           *
;                                                                              *
;  1:       Got everything for the gun working the way that I think it should with  *
;           base features.                                                      *
;                                                                              *
;  2:       Added UART functionality, I2C capability to control up to 8 targets  *
;                                                                              *
;*******************************************************************************

                    #include "MASTER.inc"                    ; processor specific variable definitions
                    #INCLUDE <MASTER_SETUP.inc>                    ; Custom setup file for the PIC16F883
micro-controller
                    #INCLUDE <MASTER_SUBROUTINES.inc>        ; File containing all used subroutines

                    LIST      p=16f1788                      ; list directive to define processor
                    errorlevel -302,-207,-305,-206,-203 ; suppress "not in bank 0" message,  Found label after column
1,
                                                             ; Using default destination of 1 (file),  Found call to
macro in column 1

                    ; CONFIG1
; __config 0xC9A4
 __CONFIG _CONFIG1, _FOSC_INTOSC & _WDTE_OFF & _PWRTE_OFF & _MCLRE_ON & _CP_OFF &
_CPD_OFF & _BOREN_OFF & _CLKOUTEN_OFF & _IESO_OFF & _FCMEN_OFF
; CONFIG2
; __config 0xDFFF
 __CONFIG _CONFIG2, _WRT_OFF & _PLLEN_OFF & _STVREN_ON & _BORV_LO & _LPBOR_OFF &
_LVP_OFF



;*****************************************
;Interrupt Vectors
```

```
;*****************************************
                ORG     H'000'          ;BEGINNING OF CODE
                GOTO    SETUP                   ;
                ORG     H'004'          ;INTERRUPT LOCATION
                GOTO    INTERRUPT               ;INTERRUPT OCCURRED, RUN THROUGH ISR

;*****************************************
;SETUP ROUTINE
;*****************************************
SETUP
                CALL    INITIALIZE              ;CALL SETUP INCLUDE FILE TO INITIALIZE
PIC
                GOTO    MAIN                    ;START MAIN CODE

;*****************************************
;INTERRUPT SERVICE ROUTINE
;*****************************************
INTERRUPT
                BANKSEL         PIR1
                BTFSC   PIR1, RCIF              ;IF INTERRUPTED BY UART RECEIVE, GOTO
UARTRECEIVE
                GOTO    UARTRECEIVE
                BTFSC   PIR1, TXIF              ;IF INTERRUPTED BY UART TRANSMIT, GOTO
UARTTRANSMIT
                GOTO    UARTTRANSMIT
                GOTO    GOBACK                  ;ELSE GOBACK
        UARTRECEIVE
                CALL    UARTRX                  ;CALL THE UARTRX SUBROUTINE
                GOTO    GOBACK
        UARTTRANSMIT
                CALL    UARTTX                  ;CALL THE UARTTX SUBROUTINE
                GOTO    GOBACK
        GOBACK
                BANKSEL         PIR1
                CLRF    PIR1            ;CLEAR ALL PIR1 FLAGS
                RETFIE                          ;RETURN TO MAIN, RE-ENABLE GIE

;*****************************************
;Main Code
;*****************************************
MAIN
                CALL    UPDATETARGETONE         ;UPDATE EACH TARGET INDIVIDUALLY
                CALL    UPDATETARGETTWO
                CALL    UPDATETARGETTHREE
                CALL    UPDATETARGETFOUR
                CALL    UPDATETARGETFIVE
                CALL    UPDATETARGETSIX
                CALL    UPDATETARGETSEVEN
                CALL    UPDATETARGETEIGHT
                GOTO    MAIN
END

;*****************************************
;END PROGRAM DIRECTIVE
;*****************************************
```

16.2.2: Target Master Subroutine Code

```
;*******************************************************************************
;                                                                               *
;  Filename:        MASTER_SUBROUTINES.inc                                      *
;  Date:    April 22, 2025                                                     *
;  File Version:  2                                                            *
;  Author:          Alex Wheelock and Andrew Keller                            *
;  Company:         Idaho State University                                     *
;  Description:   Contains all subroutines needed for the Laser Shooting Game  *
;                   project. Subroutines include the ability to communicate via *
;                   UART for control, I2C write/read, and updating each target  *
;                   slot for I2C enabling, disabling or polling.                *
;                                                                               *
;*******************************************************************************

;*******************************************************************************
;                                                                               *
;  Revision History:                                                           *
;                                                                               *
;  1:     Initialize file, get everything working the way that I thought it should  *
;         work.                                                                 *
;  2:     Update to include UART and I2C for up to eight different targets active  *
;         simultaneously.                                                       *
;                                                                               *
;*******************************************************************************

SUBROUTINES_CODE CODE

   ;WRITES THE DATA STORED IN WRITE_DATA TO THE I2C ADDRESS IN CURRENT_ADDRESS
   WRITE_I2C:
            CALL        I2CIDLE             ;ENSURE THAT THE I2C LINES ARE IDLE
            BANKSEL     SSPCON2
            BSF         SSPCON2,0
            BTFSC       SSPCON2,0           ;WAIT UNTIL START CONDITION IS COMPLETE
            GOTO        $-1
            BANKSEL     PORTA
            MOVF        CURRENT_ADDRESS, W      ;SEND I2C ADDRESS OUT
            BANKSEL     SSPBUF
            MOVWF       SSPBUF
            BANKSEL     SSPSTAT
            BTFSC       SSPSTAT,0           ;WAIT UNTIL ALL DATA HAS BEEN SHIFTED OUT OF
SSPBUF
            GOTO        $-1
            CALL        I2CIDLE             ;ENSURE THAT THE I2C LINES ARE IDLE
            BANKSEL     SSPCON2
            BTFSC       SSPCON2,6           ;DETERMINE IF THERE IS AN ACK FROM THE SLAVE
            GOTO        BAD1                ;NACK FROM THE SLAVE, STOP COMMUNICATION
            BANKSEL     PORTA
            MOVF        WRITE_DATA, W
            BANKSEL     SSPBUF
            MOVWF       SSPBUF              ;TRANSMIT DATA TO SLAVE
            BANKSEL     SSPSTAT
            BTFSC       SSPSTAT,0           ;WAIT UNTIL ALL DATA HAS BEEN SHIFTED OUT OF
SSPBUF
```

```
        GOTO    $-1
        CALL    I2CIDLE         ;ENSURE THAT THE I2C LINES ARE IDLE
        BANKSEL SSPCON2
        BTFSC   SSPCON2,6       ;DETERMINE IF THERE IS AN ACK FROM THE SLAVE
        GOTO    BAD1            ;NACK FROM THE SLAVE, STOP COMMUNICATION
        BSF     SSPCON2,2       ;CREATE STOP CONDITION
        BTFSC   SSPCON2,2
        GOTO    $-1             ;WAIT FOR STOP CONDITION TO COMPLETE
        BANKSEL PORTA
        RETURN

;REQUESTS THE CURRENT STATUS OF THE TARGET FOUND AT CURRENT_ADDRESS
READ_TARGET_STATUS:
        CALL    I2CIDLE         ;ENSURE THAT THE I2C LINES ARE IDLE
        BANKSEL SSP1CON2
        BSF     SSP1CON2,0
        BTFSC   SSP1CON2,0      ;WAIT UNTIL START CONDITION IS COMPLETE
        GOTO    $-1
        BANKSEL PORTA
        MOVF    CURRENT_ADDRESS, W      ;SEND I2C ADDRESS OUT
        ADDLW   0X01            ;CONVERT THE WRITE ADDRESS TO THE READ ADDRESS
        BANKSEL SSP1BUF
        MOVWF   SSP1BUF
        BANKSEL SSP1STAT
        BTFSC   SSP1STAT,0      ;WAIT UNTIL ALL DATA HAS BEEN SHIFTED OUT OF
SSPBUF
        GOTO    $-1
        CALL    I2CIDLE         ;ENSURE THAT THE I2C LINES ARE IDLE
        BANKSEL SSP1CON2
        BTFSC   SSP1CON2,6      ;DETERMINE IF THERE IS AN ACK FROM THE SLAVE
        GOTO    BAD1
        BSF     SSP1CON2,3      ;ENABLE RCEN TO RECEIVE DATA FROM THE SLAVE
        BANKSEL SSP1STAT
        BTFSS   SSP1STAT,0      ;WAIT UNTIL ALL DATA HAS BEEN SHIFTED INTO
SSPBUF FROM THE SLAVE
        GOTO    $-1
        BANKSEL SSP1BUF         ;\
        MOVFW   SSP1BUF         ;\\READ STATUS OF CURRENT TARGET
        BANKSEL PORTA           ;//STORE BYTE INTO RECEIVED_BYTE REGISTER
        MOVWF   RECEIVED_BYTE   ;/
        BANKSEL SSP1CON2
        BSF     SSP1CON2,2      ;CREATE STOP CONDITION
        BTFSC   SSP1CON2,2
        GOTO    $-1             ;WAIT FOR STOP CONDITION TO COMPLETE
        CALL    TEST_SCORE      ;TEST TO SEE IF A PLAYER HAS SCORED
        RETURN

;TESTS THE SCORE BASED RECEIVED_BYTE
TEST_SCORE:
        ;PLAYER1 = 0x0F
        ;PLAYER2 = 0xF0
        ;NO SCORE = 0x00
        BANKSEL PORTA
        BTFSC   RECEIVED_BYTE,3 ;DETERMINE IF PLAYER 1 SCORED
        GOTO    PLAYER1_SCORED
```

```
            BTFSC       RECEIVED_BYTE,7   ;DETERMINE IF PLAYER 2 SCORED
            GOTO        PLAYER2_SCORED
            RETURN                        ;NEITHER PLAYER SCORED, RETURN
        PLAYER1_SCORED
            MOVLW       0x01
            MOVWF       SCORE_PLAYER      ;MOVE PLAYERONE INTO SCORE_PLAYER
            MOVFW       CURRENT_ADDRESS
            MOVWF       SCORE_ADDRESS     ;MOVE THE CURRENT I2C ADDRESS INTO THE
SCORE_ADDRESS
            CALL        UARTSENDSCORE     ;CALL THE UARTSENDSCORE SUBROUTINE
            RETLW       0xFF        ;RETURN WITH 0XFF IN W TO MARK A SCORE WAS
DETECTED
        PLAYER2_SCORED
            MOVLW       0x02
            MOVWF       SCORE_PLAYER      ;MOVE PLAYERTWO INTO SCORE_PLAYER
            MOVFW       CURRENT_ADDRESS
            MOVWF       SCORE_ADDRESS     ;MOVE THE CURRENT I2C ADDRESS INTO THE
SCORE_ADDRESS
            CALL        UARTSENDSCORE
            RETLW       0xFF

    ;VERIFIES THAT THE I2C LINES ARE IDLE
    I2CIDLE:
            MOVLW       0X1F        ;Load Bus Test Value
            BANKSEL     SSP1CON2
            ANDWF       SSP1CON2,W          ;Compare 1F to check for 5 busy conditions
            BANKSEL     STATUS
            BTFSS       STATUS,Z            ;Test Zero Bit
            GOTO        I2CIDLE             ;Z=0  Bus is still busy -repeat
        CHECKR_W                            ;Z=1 Not Busy
            BANKSEL     SSP1STAT
            BTFSC       SSP1STAT,2          ;see if SSP is transmitting data
            GOTO        CHECKR_W            ;R_W = 1 - Still transmitting data
            RETURN                          ;R_W = 0 - Transmit done

    ;BAD I2C STATE, GENERATES A STOP CONDITION
    BAD1:
            MOVLW       0XFF        ;SET RETURN CODE TO -1
            BANKSEL     SSP1CON2
            BSF         SSP1CON2,PEN        ;GENERATE A STOP CONDITION
        LOOP5
            BTFSC       SSP1CON2,PEN        ;IS STOP CONDITION DONE
            GOTO        LOOP5
            RETURN

    ;WHEN UART IS RECEIVED, DETERMINES IF IT IS THE CORRECT DEVICE AND HOW TO RESPOND
    UARTRX:
            BANKSEL     PORTA
            BTFSC       UART_MODE, 4        ;IF ENABLE COMMAND ALREADY RECEIVED, GOTO
ENABLECOMMAND
            GOTO        ENABLECOMMAND
            BTFSC       UART_MODE, 3        ;IF DISABLE COMMAND ALREADY RECEIVED, GOTO
DISABLECOMMAND
            GOTO        DISABLECOMMAND
```

```
        BTFSC   UART_MODE, 2    ;IF CHANGE COMMAND ALREADY RECEIVED, GOTO
STORECHANGESLOT
        GOTO    STORECHANGESLOT
        BTFSC   UART_MODE, 1    ;IF CHANGE SLOT ALREADY RECEIVED, GOTO
CHANGESLOTADDRESS
        GOTO    CHANGESLOTADDRESS
        BTFSC   UART_MODE, 7    ;IF HANDSHAKE ALREADY RECEIVED, GOTO
INTERPRETCOMMAND
        GOTO    INTERPRETUART
        BANKSEL RCREG           ;ELSE CHECK RECEIVED BYTE FOR HANDSHAKE
        MOVF    RCREG, W
        XORLW   0X24        ;COMPARE RECEIVED BYTE WITH $
        BANKSEL PORTA
        BTFSC   STATUS, Z       ;IF RECEIVED MATCHES $, SET HNDSK FLAG IN
UART_RX_MODE
        BSF     UART_MODE, 7
        RETURN

    ;THE HANDSHAKE BYTE WAS RECEIVED PREVIOUSLY, CHECK THE NEW BYTE FOR THE
COMMAND TO EXECUTE
    INTERPRETUART
        BANKSEL PORTA
        BCF     UART_MODE, 7    ;CLEAR THE HNDSK FLAG
        BANKSEL RCREG
        MOVF    RCREG, W
        BANKSEL PORTA
        MOVWF   UART_RX_BYTE    ;STORE RECEIVED BYTE IN UART_RX_BYTE
        XORLW   0X56
        BTFSC   STATUS, Z       ;IF RECEIVED BYTE IS DEVICE VERIFICATION
COMMAND, GOTO DEVICEVERIFICATION
        GOTO    DEVICEVERIFICATION
        MOVF    UART_RX_BYTE, W
        XORLW   0X45
        BTFSC   STATUS, Z       ;IF RECEIVED BYTE IS ENABLE COMMAND, SET
ENABLE FLAG FOR NEXT RECEIVE BYTE
        BSF     UART_MODE, 4
        MOVF    UART_RX_BYTE, W
        XORLW   0X44
        BTFSC   STATUS, Z       ;IF RECEIVED BYTE IS DISABLE COMMAND, SET
DISABLE FLAG FOR NEXT RECEIVE BYTE
        BSF     UART_MODE, 3
        MOVF    UART_RX_BYTE, W
        XORLW   0X43
        BTFSC   STATUS, Z       ;IF RECEIVED BYTE IS CLEAR COMMAND, GOTO
CLEAR TARGETS
        GOTO    CLEARTARGETS
        MOVF    UART_RX_BYTE, W
        XORLW   0X41
        BTFSC   STATUS, Z       ;IF RECEIVED BYTE IS CHANGE TARGET COMMAND,
SET CHANGE FLAG FOR NEXT RECEIVE BYTE
        BSF     UART_MODE, 2
        RETURN

    ;THE DEVICE VERIFICATION COMMAND WAS RECEIVED, RESPOND WITH HANDSHAKE
BYTE AND FIRST BYTE OF VERIFICATION
```

```
        DEVICEVERIFICATION
          BANKSEL     TXREG
          MOVLW       0X24
          MOVWF       TXREG               ;TRANSMIT HANDSHAKE BYTE
          MOVLW       0X4C
          MOVWF       TXREG               ;TRANSMIT FIRST HALF OF DEVICE VERIFICATION
          BANKSEL     PORTA
          BSF         UART_MODE, 6        ;SET THE VERIFY FLAG FOR NEXT TX INTERRUPT
          BANKSEL     PIE1
          BSF         PIE1, TXIE          ;ENABLE THE UART TRANSMIT INTERRUPT
          RETURN

        ;THE ENABLE COMMAND WAS RECEIVED, INCLUSIVE OR THE THIRD RECEIVED BYTE
WITH THE ENABLE_TARGET REGISTER
        ENABLECOMMAND
          BANKSEL     RCREG
          MOVF        RCREG, W            ;RETRIEVE THE ENABLE BYTE
          BANKSEL     PORTA
          IORWF       ENABLE_TARGET, W ;OR THE ENABLE BYTE WITH THE ENABLE_TARGET
REGISTER
          MOVWF       ENABLE_TARGET
          BCF         UART_MODE, 4        ;CLEAR ENABLE FLAG
          RETURN

        ;THE DISABLE COMMAND WAS RECEIVED, INCLUSIVE OR THE THIRD RECEIVED BYTE
WITH THE DISABLE_TARGET REGISTER
        DISABLECOMMAND
          BANKSEL     RCREG
          MOVF        RCREG, W            ;RETRIEVE THE DISABLE BYTE
          BANKSEL     PORTA
          IORWF       DISABLE_TARGET, f ;OR THE DISABLE BYTE WITH THE DISABLE_TARGET
REGISTER
          BCF         UART_MODE, 3        ;CLEAR DISABLE FLAG
          RETURN

        ;THE CLEAR COMMAND WAS RECEIVED, MOVE A 0XFF INTO DISABLE_TARGET
        CLEARTARGETS
          BANKSEL     PORTA
          MOVLW       0XFF
          MOVWF       DISABLE_TARGET    ;SET DISABLE FLAGS FOR ALL TARGETS
          RETURN

        ;THE CHANGE SLOT ADDRESS COMMAND WAS RECEIVED, STORE THE THIRD BYTE
RECEIVED IN THE CHANGE_SLOT REGISTER
        STORECHANGESLOT
          BANKSEL     RCREG
          MOVF        RCREG, W            ;RETRIEVE THE CHANGE SLOT
          BANKSEL     PORTA
          MOVWF       CHANGE_SLOT               ;STORE THE CHANGE SLOT
          BCF         UART_MODE, 2        ;CLEAR CHANGE FLAG
          BSF         UART_MODE, 1        ;SET SLOT FLAG
          RETURN

        ;THE CHANGE SLOT ADDRESS COMMAND WAS RECEIVED, STORE THE FOURTH BYTE
RECEIVED IN THE CORRESPONDING TARGET SLOT
```

```
        CHANGESLOTADDRESS
          BANKSEL     RCREG
          MOVF        RCREG, W            ;RETRIEVE THE I2C ADDRESS
          BANKSEL     PORTA
          BCF         UART_MODE, 1        ;CLEAR THE SLOT FLAG
          BTFSC       CHANGE_SLOT, 7      ;GOTO CHANGETARGET ASSOCIATED WITH CHANGE
SLOT BIT
          GOTO        CHANGETARGETONE
          BTFSC       CHANGE_SLOT, 6
          GOTO        CHANGETARGETTWO
          BTFSC       CHANGE_SLOT, 5
          GOTO        CHANGETARGETTHREE
          BTFSC       CHANGE_SLOT, 4
          GOTO        CHANGETARGETFOUR
          BTFSC       CHANGE_SLOT, 3
          GOTO        CHANGETARGETFIVE
          BTFSC       CHANGE_SLOT, 2
          GOTO        CHANGETARGETSIX
          BTFSC       CHANGE_SLOT, 1
          GOTO        CHANGETARGETSEVEN
          BTFSC       CHANGE_SLOT, 0
          GOTO        CHANGETARGETEIGHT
          RETURN

        CHANGETARGETONE
          BTFSC       INUSE_TARGET, 7     ;IF THE TARGET IS CURRENTLY IN USE, RETURN
          RETURN
          BANKSEL     PORTA
          MOVWF       TGT_ONE_ADDRESS ;ELSE UPDATE THE TARGET SLOT ADDRESS
          RETURN

        CHANGETARGETTWO
          BTFSC       INUSE_TARGET, 6     ;IF THE TARGET IS CURRENTLY IN USE, RETURN
          RETURN
          BANKSEL     PORTA
          MOVWF       TGT_TWO_ADDRESS;ELSE UPDATE THE TARGET SLOT ADDRESS
          RETURN

        CHANGETARGETTHREE
          BTFSC       INUSE_TARGET, 5     ;IF THE TARGET IS CURRENTLY IN USE, RETURN
          RETURN
          BANKSEL     PORTA
          MOVWF       TGT_THREE_ADDRESS       ;ELSE UPDATE THE TARGET SLOT ADDRESS
          RETURN

        CHANGETARGETFOUR
          BTFSC       INUSE_TARGET, 4     ;IF THE TARGET IS CURRENTLY IN USE, RETURN
          RETURN
          BANKSEL     PORTA
          MOVWF       TGT_FOUR_ADDRESS        ;ELSE UPDATE THE TARGET SLOT ADDRESS
          RETURN

        CHANGETARGETFIVE
          BTFSC       INUSE_TARGET, 3     ;IF THE TARGET IS CURRENTLY IN USE, RETURN
          RETURN
```

```
            BANKSEL     PORTA
            MOVWF       TGT_FIVE_ADDRESS;ELSE UPDATE THE TARGET SLOT ADDRESS
            RETURN

        CHANGETARGETSIX
            BTFSC       INUSE_TARGET, 2     ;IF THE TARGET IS CURRENTLY IN USE, RETURN
            RETURN
            BANKSEL     PORTA
            MOVWF       TGT_SIX_ADDRESS     ;ELSE UPDATE THE TARGET SLOT ADDRESS
            RETURN

        CHANGETARGETSEVEN
            BTFSC       INUSE_TARGET, 1     ;IF THE TARGET IS CURRENTLY IN USE, RETURN
            RETURN
            BANKSEL     PORTA
            MOVWF       TGT_SEVEN_ADDRESS       ;ELSE UPDATE THE TARGET SLOT ADDRESS
            RETURN

        CHANGETARGETEIGHT
            BTFSC       INUSE_TARGET, 0     ;IF THE TARGET IS CURRENTLY IN USE, RETURN
            RETURN
            BANKSEL     PORTA
            MOVWF       TGT_EIGHT_ADDRESS       ;ELSE UPDATE THE TARGET SLOT ADDRESS
            RETURN

    ;THE UART TX INTERRUPT HAS OCCURED, IF THE VERIFY FLAG IS SET, SEND THE LAST BYTE OF
THE VERIFY COMMAND
    ;IF THE PLAYER FLAG IS SET, SEND THE SCORING PLAYER NUMBER
    UARTTX:
            BANKSEL     PIE1
            BCF         PIE1, TXIE          ;DISABLE UART TX INTERRUPT
            BTFSC       PIE1, TXIE
            GOTO        $-2
            BANKSEL     PORTA
            BTFSC       UART_MODE, 6        ;IF THE VERIFY FLAG IS SET, GOTO TXVERIFY
            GOTO        TXVERIFY
            BTFSC       UART_MODE, 5        ;IF THE PLAYER FLAG IS SET, GOTO TXVERIFY
            GOTO        TXPLAYER
            RETURN
        TXVERIFY
            BANKSEL     TXREG
            MOVLW       0X41
            MOVWF       TXREG               ;TRANSMIT THE SECOND HALF OF DEVICE
VERIFICATION
            BANKSEL     PORTA
            BCF         UART_MODE, 6        ;CLEAR THE VERIFY FLAG
            RETURN
        TXPLAYER
            BANKSEL     PORTA
            BCF         UART_MODE, 5        ;CLEAR PLAYER FLAG
            MOVF        SCORE_PLAYER, W
            BANKSEL     TXREG
            MOVWF       TXREG               ;TRANSMIT THE SCORING PLAYER NUMBER
            RETURN
```

```
  ;A PLAYER HAS SCORED, SEND THE HANDSHAKE BYTE AND I2C ADDRESS OF THE SCORED
TARGET VIA UART. SET THE PLAYER FLAG FOR THE TX INTERRUPT
  UARTSENDSCORE:
        BANKSEL     TXREG
        MOVLW       0X24
        MOVWF       TXREG               ;TRANSMIT HANDSHAKE BYTE
        BANKSEL     PORTA
        MOVF        SCORE_ADDRESS, W
        BANKSEL     TXREG
        MOVWF       TXREG               ;TRANSMIT I2C ADDRESS OF SCORED TARGET
        BANKSEL     PORTA
        BSF         UART_MODE, 5        ;SET THE PLAYER FLAG FOR NEXT TX INTERRUPT
        BANKSEL     PIE1
        BSF         PIE1, TXIE          ;ENABLE THE UART TRANSMIT INTERRUPT
        RETURN


  ;CALLED IN MAIN TO EITHER ENABLE, DISABLE, OR POLL TARGET SLOT ONE
  UPDATETARGETONE:
        BANKSEL     PORTA
        MOVFW       TGT_ONE_ADDRESS ;MOVE AN ADDRESS TO THE CURRENT_ADDRESS
REGISTER
        MOVWF       CURRENT_ADDRESS
        BTFSC       ENABLE_TARGET, 7  ;IF THE TARGET IS REQUESTED TO BE ENABLED,
SEND THE ENABLE COMMAND
        GOTO        ENABLETARGETONE
        BTFSC       DISABLE_TARGET, 7 ;IF THE TARGET IS REQUESTED TO BE DISABLED,
SEND THE DISABLE COMMAND
        GOTO        DISABLETARGETONE
        BTFSC       INUSE_TARGET, 7     ;IF THE TARGET IS CURRENTLY IN USE, POLL THE
TARGET
        GOTO        POLLTARGETONE
        RETURN
      ENABLETARGETONE
        BANKSEL     PORTA
        MOVLW       0X45          ;MOVE AN ENABLE TO THE WRITE_DATA REGISTER
        MOVWF       WRITE_DATA
        CALL        WRITE_I2C           ;CALL THE WRITE_I2C SUBROUTINE TO ENABLE THE
TARGET
        BANKSEL     PORTA
        BCF         ENABLE_TARGET, 7  ;CLEAR THE ENABLE TARGET FLAG
        BSF         INUSE_TARGET, 7     ;MARK THE TARGET AS IN USE
        RETURN
      DISABLETARGETONE
        BANKSEL     PORTA
        MOVLW       0X44          ;MOVE A LETTER D TO THE WRITE_DATA REGISTER
        MOVWF       WRITE_DATA
        CALL        WRITE_I2C           ;CALL THE WRITE_I2C SUBROUTINE TO ENABLE THE
TARGET
        BANKSEL     PORTA
        BCF         DISABLE_TARGET, 7 ;CLEAR THE DISABLE TARGET FLAG
        BCF         INUSE_TARGET, 7     ;MARK THE TARGET AS NO LONGER IN USE
        RETURN
      POLLTARGETONE
        CALL        READ_TARGET_STATUS      ;ASK IF THE TARGET HAS BEEN SCORED ON
```

```
        XORLW   0XFF        ;IF THE TARGET HAD A SCORE, W RETURNED WITH 0XFF
        BANKSEL PORTA
        BTFSC   STATUS, Z           ;IF A SCORE WAS RECORDED, CLEAR THE INUSE
TARGET FLAG
        BCF     INUSE_TARGET, 7     ;TARGET AUTOMATICALLY DISABLED ITSELF
        RETURN


  ;CALLED IN MAIN TO EITHER ENABLE, DISABLE, OR POLL TARGET SLOT TWO
  UPDATETARGETTWO:
        BANKSEL PORTA
        MOVFW   TGT_TWO_ADDRESS;MOVE AN ADDRESS TO THE CURRENT_ADDRESS
REGISTER
        MOVWF   CURRENT_ADDRESS
        BTFSC   ENABLE_TARGET, 6  ;IF THE TARGET IS REQUESTED TO BE ENABLED,
SEND THE ENABLE COMMAND
        GOTO    ENABLETARGETTWO
        BTFSC   DISABLE_TARGET, 6 ;IF THE TARGET IS REQUESTED TO BE DISABLED,
SEND THE DISABLE COMMAND
        GOTO    DISABLETARGETTWO
        BTFSC   INUSE_TARGET, 6     ;IF THE TARGET IS CURRENTLY IN USE, POLL THE
TARGET
        GOTO    POLLTARGETTWO
        RETURN
      ENABLETARGETTWO
        BANKSEL PORTA
        MOVLW   0X45        ;MOVE AN ENABLE TO THE WRITE_DATA REGISTER
        MOVWF   WRITE_DATA
        CALL    WRITE_I2C           ;CALL THE WRITE_I2C SUBROUTINE TO ENABLE THE
TARGET
        BANKSEL PORTA
        BCF     ENABLE_TARGET, 6  ;CLEAR THE ENABLE TARGET FLAG
        BSF     INUSE_TARGET, 6     ;MARK THE TARGET AS IN USE
        RETURN
      DISABLETARGETTWO
        BANKSEL PORTA
        MOVLW   0X44        ;MOVE A LETTER D TO THE WRITE_DATA REGISTER
        MOVWF   WRITE_DATA
        CALL    WRITE_I2C           ;CALL THE WRITE_I2C SUBROUTINE TO ENABLE THE
TARGET
        BANKSEL PORTA
        BCF     DISABLE_TARGET, 6 ;CLEAR THE DISABLE TARGET FLAG
        BCF     INUSE_TARGET, 6     ;MARK THE TARGET AS NO LONGER IN USE
        RETURN
      POLLTARGETTWO
        CALL    READ_TARGET_STATUS      ;ASK IF THE TARGET HAS BEEN SCORED ON
        XORLW   0XFF        ;IF THE TARGET HAD A SCORE, W RETURNED WITH 0XFF
        BANKSEL PORTA
        BTFSC   STATUS, Z           ;IF A SCORE WAS RECORDED, CLEAR THE INUSE
TARGET FLAG
        BCF     INUSE_TARGET, 6     ;TARGET AUTOMATICALLY DISABLED ITSELF
        RETURN


  ;CALLED IN MAIN TO EITHER ENABLE, DISABLE, OR POLL TARGET SLOT THREE
```

```
  UPDATETARGETTHREE:
        BANKSEL     PORTA
        MOVFW       TGT_THREE_ADDRESS       ;MOVE AN ADDRESS TO THE
CURRENT_ADDRESS REGISTER
        MOVWF       CURRENT_ADDRESS
        BTFSC       ENABLE_TARGET, 5  ;IF THE TARGET IS REQUESTED TO BE ENABLED,
SEND THE ENABLE COMMAND
        GOTO        ENABLETARGETTHREE
        BTFSC       DISABLE_TARGET, 5 ;IF THE TARGET IS REQUESTED TO BE DISABLED,
SEND THE DISABLE COMMAND
        GOTO        DISABLETARGETTHREE
        BTFSC       INUSE_TARGET, 5     ;IF THE TARGET IS CURRENTLY IN USE, POLL THE
TARGET
        GOTO        POLLTARGETTHREE
        RETURN
      ENABLETARGETTHREE
        BANKSEL     PORTA
        MOVLW       0X45          ;MOVE AN ENABLE TO THE WRITE_DATA REGISTER
        MOVWF       WRITE_DATA
        CALL        WRITE_I2C           ;CALL THE WRITE_I2C SUBROUTINE TO ENABLE THE
TARGET
        BANKSEL     PORTA
        BCF         ENABLE_TARGET, 5  ;CLEAR THE ENABLE TARGET FLAG
        BSF         INUSE_TARGET, 5     ;MARK THE TARGET AS IN USE
        RETURN
      DISABLETARGETTHREE
        BANKSEL     PORTA
        MOVLW       0X44          ;MOVE A LETTER D TO THE WRITE_DATA REGISTER
        MOVWF       WRITE_DATA
        CALL        WRITE_I2C           ;CALL THE WRITE_I2C SUBROUTINE TO ENABLE THE
TARGET
        BANKSEL     PORTA
        BCF         DISABLE_TARGET, 5 ;CLEAR THE DISABLE TARGET FLAG
        BCF         INUSE_TARGET, 5     ;MARK THE TARGET AS NO LONGER IN USE
        RETURN
      POLLTARGETTHREE
        CALL        READ_TARGET_STATUS      ;ASK IF THE TARGET HAS BEEN SCORED ON
        XORLW       0XFF          ;IF THE TARGET HAD A SCORE, W RETURNED WITH 0XFF
        BANKSEL     PORTA
        BTFSC       STATUS, Z           ;IF A SCORE WAS RECORDED, CLEAR THE INUSE
TARGET FLAG
        BCF         INUSE_TARGET, 5     ;TARGET AUTOMATICALLY DISABLED ITSELF
        RETURN


  ;CALLED IN MAIN TO EITHER ENABLE, DISABLE, OR POLL TARGET SLOT FOUR
  UPDATETARGETFOUR:
        BANKSEL     PORTA
        MOVFW       TGT_FOUR_ADDRESS        ;MOVE AN ADDRESS TO THE
CURRENT_ADDRESS REGISTER
        MOVWF       CURRENT_ADDRESS
        BTFSC       ENABLE_TARGET, 4  ;IF THE TARGET IS REQUESTED TO BE ENABLED,
SEND THE ENABLE COMMAND
        GOTO        ENABLETARGETFOUR
```

```
        BTFSC   DISABLE_TARGET, 4 ;IF THE TARGET IS REQUESTED TO BE DISABLED,
SEND THE DISABLE COMMAND
        GOTO    DISABLETARGETFOUR
        BTFSC   INUSE_TARGET, 4   ;IF THE TARGET IS CURRENTLY IN USE, POLL THE
TARGET
        GOTO    POLLTARGETFOUR
        RETURN
      ENABLETARGETFOUR
        BANKSEL PORTA
        MOVLW   0X45        ;MOVE AN ENABLE TO THE WRITE_DATA REGISTER
        MOVWF   WRITE_DATA
        CALL    WRITE_I2C           ;CALL THE WRITE_I2C SUBROUTINE TO ENABLE THE
TARGET
        BANKSEL PORTA
        BCF     ENABLE_TARGET, 4  ;CLEAR THE ENABLE TARGET FLAG
        BSF     INUSE_TARGET, 4   ;MARK THE TARGET AS IN USE
        RETURN
      DISABLETARGETFOUR
        BANKSEL PORTA
        MOVLW   0X44        ;MOVE A LETTER D TO THE WRITE_DATA REGISTER
        MOVWF   WRITE_DATA
        CALL    WRITE_I2C           ;CALL THE WRITE_I2C SUBROUTINE TO ENABLE THE
TARGET
        BANKSEL PORTA
        BCF     DISABLE_TARGET, 4 ;CLEAR THE DISABLE TARGET FLAG
        BCF     INUSE_TARGET, 4   ;MARK THE TARGET AS NO LONGER IN USE
        RETURN
      POLLTARGETFOUR
        CALL    READ_TARGET_STATUS      ;ASK IF THE TARGET HAS BEEN SCORED ON
        XORLW   0XFF        ;IF THE TARGET HAD A SCORE, W RETURNED WITH 0XFF
        BANKSEL PORTA
        BTFSC   STATUS, Z           ;IF A SCORE WAS RECORDED, CLEAR THE INUSE
TARGET FLAG
        BCF     INUSE_TARGET, 4   ;TARGET AUTOMATICALLY DISABLED ITSELF
        RETURN


  ;CALLED IN MAIN TO EITHER ENABLE, DISABLE, OR POLL TARGET SLOT FIVE
  UPDATETARGETFIVE:
        BANKSEL PORTA
        MOVFW   TGT_FIVE_ADDRESS;MOVE AN ADDRESS TO THE CURRENT_ADDRESS
REGISTER
        MOVWF   CURRENT_ADDRESS
        BTFSC   ENABLE_TARGET, 3  ;IF THE TARGET IS REQUESTED TO BE ENABLED,
SEND THE ENABLE COMMAND
        GOTO    ENABLETARGETFIVE
        BTFSC   DISABLE_TARGET, 3 ;IF THE TARGET IS REQUESTED TO BE DISABLED,
SEND THE DISABLE COMMAND
        GOTO    DISABLETARGETFIVE
        BTFSC   INUSE_TARGET, 3   ;IF THE TARGET IS CURRENTLY IN USE, POLL THE
TARGET
        GOTO    POLLTARGETFIVE
        RETURN
      ENABLETARGETFIVE
        BANKSEL PORTA
```

```
            MOVLW       0X45            ;MOVE AN ENABLE TO THE WRITE_DATA REGISTER
            MOVWF       WRITE_DATA
            CALL        WRITE_I2C               ;CALL THE WRITE_I2C SUBROUTINE TO ENABLE THE
TARGET
            BANKSEL     PORTA
            BCF         ENABLE_TARGET, 3  ;CLEAR THE ENABLE TARGET FLAG
            BSF         INUSE_TARGET, 3     ;MARK THE TARGET AS IN USE
            RETURN
          DISABLETARGETFIVE
            BANKSEL     PORTA
            MOVLW       0X44            ;MOVE A LETTER D TO THE WRITE_DATA REGISTER
            MOVWF       WRITE_DATA
            CALL        WRITE_I2C               ;CALL THE WRITE_I2C SUBROUTINE TO ENABLE THE
TARGET
            BANKSEL     PORTA
            BCF         DISABLE_TARGET, 3 ;CLEAR THE DISABLE TARGET FLAG
            BCF         INUSE_TARGET, 3     ;MARK THE TARGET AS NO LONGER IN USE
            RETURN
          POLLTARGETFIVE
            CALL        READ_TARGET_STATUS      ;ASK IF THE TARGET HAS BEEN SCORED ON
            XORLW       0XFF            ;IF THE TARGET HAD A SCORE, W RETURNED WITH 0XFF
            BANKSEL     PORTA
            BTFSC       STATUS, Z               ;IF A SCORE WAS RECORDED, CLEAR THE INUSE
TARGET FLAG
            BCF         INUSE_TARGET, 3     ;TARGET AUTOMATICALLY DISABLED ITSELF
            RETURN


    ;CALLED IN MAIN TO EITHER ENABLE, DISABLE, OR POLL TARGET SLOT SIX
    UPDATETARGETSIX:
            BANKSEL     PORTA
            MOVFW       TGT_SIX_ADDRESS   ;MOVE AN ADDRESS TO THE CURRENT_ADDRESS
REGISTER
            MOVWF       CURRENT_ADDRESS
            BTFSC       ENABLE_TARGET, 2  ;IF THE TARGET IS REQUESTED TO BE ENABLED,
SEND THE ENABLE COMMAND
            GOTO        ENABLETARGETSIX
            BTFSC       DISABLE_TARGET, 2 ;IF THE TARGET IS REQUESTED TO BE DISABLED,
SEND THE DISABLE COMMAND
            GOTO        DISABLETARGETSIX
            BTFSC       INUSE_TARGET, 2     ;IF THE TARGET IS CURRENTLY IN USE, POLL THE
TARGET
            GOTO        POLLTARGETSIX
            RETURN
          ENABLETARGETSIX
            BANKSEL     PORTA
            MOVLW       0X45            ;MOVE AN ENABLE TO THE WRITE_DATA REGISTER
            MOVWF       WRITE_DATA
            CALL        WRITE_I2C               ;CALL THE WRITE_I2C SUBROUTINE TO ENABLE THE
TARGET
            BANKSEL     PORTA
            BCF         ENABLE_TARGET, 2  ;CLEAR THE ENABLE TARGET FLAG
            BSF         INUSE_TARGET, 2     ;MARK THE TARGET AS IN USE
            RETURN
          DISABLETARGETSIX
```

```
            BANKSEL     PORTA
            MOVLW       0X44        ;MOVE A LETTER D TO THE WRITE_DATA REGISTER
            MOVWF       WRITE_DATA
            CALL        WRITE_I2C           ;CALL THE WRITE_I2C SUBROUTINE TO ENABLE THE
TARGET
            BANKSEL     PORTA
            BCF         DISABLE_TARGET, 2 ;CLEAR THE DISABLE TARGET FLAG
            BCF         INUSE_TARGET, 2     ;MARK THE TARGET AS NO LONGER IN USE
            RETURN
          POLLTARGETSIX
            CALL        READ_TARGET_STATUS      ;ASK IF THE TARGET HAS BEEN SCORED ON
            XORLW       0XFF        ;IF THE TARGET HAD A SCORE, W RETURNED WITH 0XFF
            BANKSEL     PORTA
            BTFSC       STATUS, Z           ;IF A SCORE WAS RECORDED, CLEAR THE INUSE
TARGET FLAG
            BCF         INUSE_TARGET, 2     ;TARGET AUTOMATICALLY DISABLED ITSELF
            RETURN


   ;CALLED IN MAIN TO EITHER ENABLE, DISABLE, OR POLL TARGET SLOT SEVEN
   UPDATETARGETSEVEN:
            BANKSEL     PORTA
            MOVFW       TGT_SEVEN_ADDRESS       ;MOVE AN ADDRESS TO THE
CURRENT_ADDRESS REGISTER
            MOVWF       CURRENT_ADDRESS
            BTFSC       ENABLE_TARGET, 1  ;IF THE TARGET IS REQUESTED TO BE ENABLED,
SEND THE ENABLE COMMAND
            GOTO        ENABLETARGETSEVEN
            BTFSC       DISABLE_TARGET, 1 ;IF THE TARGET IS REQUESTED TO BE DISABLED,
SEND THE DISABLE COMMAND
            GOTO        DISABLETARGETSEVEN
            BTFSC       INUSE_TARGET, 1     ;IF THE TARGET IS CURRENTLY IN USE, POLL THE
TARGET
            GOTO        POLLTARGETSEVEN
            RETURN
          ENABLETARGETSEVEN
            BANKSEL     PORTA
            MOVLW       0X45        ;MOVE AN ENABLE TO THE WRITE_DATA REGISTER
            MOVWF       WRITE_DATA
            CALL        WRITE_I2C           ;CALL THE WRITE_I2C SUBROUTINE TO ENABLE THE
TARGET
            BANKSEL     PORTA
            BCF         ENABLE_TARGET, 1  ;CLEAR THE ENABLE TARGET FLAG
            BSF         INUSE_TARGET, 1     ;MARK THE TARGET AS IN USE
            RETURN
          DISABLETARGETSEVEN
            BANKSEL     PORTA
            MOVLW       0X44        ;MOVE A LETTER D TO THE WRITE_DATA REGISTER
            MOVWF       WRITE_DATA
            CALL        WRITE_I2C           ;CALL THE WRITE_I2C SUBROUTINE TO ENABLE THE
TARGET
            BANKSEL     PORTA
            BCF         DISABLE_TARGET, 1 ;CLEAR THE DISABLE TARGET FLAG
            BCF         INUSE_TARGET, 1     ;MARK THE TARGET AS NO LONGER IN USE
            RETURN
```

```
        POLLTARGETSEVEN
            CALL        READ_TARGET_STATUS      ;ASK IF THE TARGET HAS BEEN SCORED ON
            XORLW       0XFF            ;IF THE TARGET HAD A SCORE, W RETURNED WITH 0XFF
            BANKSEL     PORTA
            BTFSC       STATUS, Z               ;IF A SCORE WAS RECORDED, CLEAR THE INUSE
TARGET FLAG
            BCF         INUSE_TARGET, 1     ;TARGET AUTOMATICALLY DISABLED ITSELF
            RETURN


    ;CALLED IN MAIN TO EITHER ENABLE, DISABLE, OR POLL TARGET SLOT EIGHT
    UPDATETARGETEIGHT:
            BANKSEL     PORTA
            MOVFW       TGT_EIGHT_ADDRESS       ;MOVE AN ADDRESS TO THE
CURRENT_ADDRESS REGISTER
            MOVWF       CURRENT_ADDRESS
            BTFSC       ENABLE_TARGET, 0  ;IF THE TARGET IS REQUESTED TO BE ENABLED,
SEND THE ENABLE COMMAND
            GOTO        ENABLETARGETEIGHT
            BTFSC       DISABLE_TARGET, 0 ;IF THE TARGET IS REQUESTED TO BE DISABLED,
SEND THE DISABLE COMMAND
            GOTO        DISABLETARGETEIGHT
            BTFSC       INUSE_TARGET, 0     ;IF THE TARGET IS CURRENTLY IN USE, POLL THE
TARGET
            GOTO        POLLTARGETEIGHT
            RETURN
        ENABLETARGETEIGHT
            BANKSEL     PORTA
            MOVLW       0X45            ;MOVE AN ENABLE TO THE WRITE_DATA REGISTER
            MOVWF       WRITE_DATA
            CALL        WRITE_I2C               ;CALL THE WRITE_I2C SUBROUTINE TO ENABLE THE
TARGET
            BANKSEL     PORTA
            BCF         ENABLE_TARGET, 0  ;CLEAR THE ENABLE TARGET FLAG
            BSF         INUSE_TARGET, 0     ;MARK THE TARGET AS IN USE
            RETURN
        DISABLETARGETEIGHT
            BANKSEL     PORTA
            MOVLW       0X44            ;MOVE A LETTER D TO THE WRITE_DATA REGISTER
            MOVWF       WRITE_DATA
            CALL        WRITE_I2C               ;CALL THE WRITE_I2C SUBROUTINE TO ENABLE THE
TARGET
            BANKSEL     PORTA
            BCF         DISABLE_TARGET, 0 ;CLEAR THE DISABLE TARGET FLAG
            BCF         INUSE_TARGET, 0     ;MARK THE TARGET AS NO LONGER IN USE
            RETURN
        POLLTARGETEIGHT
            CALL        READ_TARGET_STATUS      ;ASK IF THE TARGET HAS BEEN SCORED ON
            XORLW       0XFF            ;IF THE TARGET HAD A SCORE, W RETURNED WITH 0XFF
            BANKSEL     PORTA
            BTFSC       STATUS, Z               ;IF A SCORE WAS RECORDED, CLEAR THE INUSE
TARGET FLAG
            BCF         INUSE_TARGET, 0     ;TARGET AUTOMATICALLY DISABLED ITSELF
            RETURN
```

16.2.3: Target Master Setup Code

```
;*******************************************************************************
;                                                                              *
;   Filename:           MASTER_SETUP.inc                                       *
;   Date:     April 22, 2025                                                   *
;   File Version:   2                                                          *
;   Author:             Alex Wheelock and Andrew Keller                        *
;   Company:            Idaho State University                                 *
;   Description:    Firmware for setting up a PIC16F1788 for the master of the *
;                       Laser Shooting Game project.                           *
;                                                                              *
;*******************************************************************************

;*******************************************************************************
;                                                                              *
;   Revision History:                                                          *
;                                                                              *
;   1:      Iniitialize file, setup how I think it should be setup for the project.  *
;                                                                              *
;   2:       Added UART functionality, I2C capability to control up to 8 targets  *
;                                                                              *
;*******************************************************************************

;================================================================================
==============================================*
;                                                   PORTA BITMAP
                                *
;================================================================================
==============================================*
;               |               |               |               |               |               |
        |               *
;       7       |       6       |       5       |       4       |       3       |       2       |
1       |       0       *
;---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------*
;DIGITAL        |DIGITAL        |DIGITAL        |DIGITAL        |DIGITAL        |DIGITAL
|DIGITAL        |DIGITAL        *
;OUTPUT                 |OUTPUT                 |OUTPUT                 |OUTPUT
|OUTPUT                 |OUTPUT                 |OUTPUT                 |OUTPUT                 *
;---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------*
;UNUSED                 |UNUSED                 |UNUSED                 |UNUSED
|UNUSED                 |UNUSED                 |UNUSED                 |UNUSED                 *
;               |               |               |               |               |               |
        |               *
;               |               |               |               |               |               |
        |               *
;               |               |               |               |               |               |
        |               *
;-------------------------------------------------------------------------------------------------------------------------------*

;================================================================================
==============================================*
;                                                   PORTB BITMAP
                                *
```

```
;=========================================================================================
==========================================*
;               |               |               |               |               |               |
        |               *
;       7       |       6       |       5       |       4       |       3       |       2       |
1       |       0       *
;---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------*
;DIGITAL        |DIGITAL        |DIGITAL        |DIGITAL        |DIGITAL        |DIGITAL
|DIGITAL        |DIGITAL        *
;OUTPUT                 |OUTPUT                 |OUTPUT                 |OUTPUT
|OUTPUT                 |OUTPUT                 |OUTPUT                 |OUTPUT                 *
;---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------*
;UNUSED                 |UNUSED                 |UNUSED                 |UNUSED
|UNUSED                 |UNUSED                 |UNUSED                 |UNUSED                 *
;               |               |               |               |               |               |
        |               *
;               |               |               |               |               |               |
        |               *
;               |               |               |               |               |               |
        |               *
;-----------------------------------------------------------------------------------------------------------------------------*

;=========================================================================================
==========================================*
;                                                       PORTC BITMAP
                                *
;=========================================================================================
==========================================*
;               |               |               |               |               |               |
        |               *
;       7       |       6       |       5       |       4       |       3       |       2       |
1       |       0       *
;---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------*
;DIGITAL        |DIGITAL        |DIGITAL        |DIGITAL        |DIGITAL        |DIGITAL
|DIGITAL        |DIGITAL        *
;OUTPUT                 |OUTPUT                 |OUTPUT                 |OUTPUT
|OUTPUT                 |OUTPUT                 |OUTPUT                 |OUTPUT                 *
;---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------*
;UNUSED         (UART) |UNUSED         (UART) |UNUSED                 |UNUSED (I2C)   |UNUSED (I2C)
|UNUSED                 |UNUSED                 |UNUSED                 *
;               |               |               |               |               |               |
        |               *
;               |               |               |               |               |               |
        |               *
;               |               |               |               |               |               |
        |               *
;-----------------------------------------------------------------------------------------------------------------------------*


INITIALIZE_CODE         CODE
INITIALIZE

;*** INTOSC SETUP *******

                BANKSEL         OSCCON
```

```
                MOVLW           0x7A                    ;ENABLE INTOSC, SET TO 16MHz
                MOVWF           OSCCON
                BANKSEL         CLKRCON
                CLRF    CLKRCON                         ;DISABLE CLOCK REFERENCE


;*** SFR SETUP **********


;*** SET OPTION_REG: ****

                BANKSEL         OPTION_REG
                BCF     OPTION_REG, PS0         ;\\
                BCF     OPTION_REG, PS1         ; >>PRESCALER RATIO SET 1:1
                BCF     OPTION_REG, PS2         ;//
                BSF     OPTION_REG, PSA         ;PRESCALER ASSIGNED TO WDT
                BCF     OPTION_REG, TMR0SE              ;TMR0 EDGE SET RISING
                BCF     OPTION_REG, TMR0CS              ;TMR0 CLOCK SOURCE SET TO
INTERNAL
                BSF     OPTION_REG, INTEDG              ;RB0/INT SET TO RISING EDGE
                BSF     OPTION_REG, NOT_WPUEN   ;WEAK PULLUP ENABLED

;*** SET INTCON REG: ****

                BANKSEL INTCON
                BCF     INTCON, IOCIE           ;ENABLE IOC INTERUPT
                BCF     INTCON, INTE            ;DISABLE INT EXTERNAL INTERUPT
                BSF     INTCON, PEIE            ;ENABLE PERIPHERAL INTERUPTS
                BSF     INTCON, GIE             ;ENABLE ALL UNMASKED INTERUPTS

;*** SET PIE1-4 REG: *****

                BANKSEL         PIE1
                MOVLW           B'00101000'
                MOVWF           PIE1                    ;DISABLE ALL PIE1 INTERUPTS, EXCEPT
SSP1IE, RCIE
                BANKSEL         PIE2
                CLRF    PIE2                    ;DISABLE ALL PIE2 INTERUPTS
                BANKSEL         PIE3
                CLRF    PIE3                    ;DISABLE ALL PIE3 INTERUPTS
                BANKSEL         PIE4
                CLRF    PIE4                    ;DISABLE ALL PIE4 INTERUPTS

;*** CLEAR INTERUPT FLAGS ***

                BANKSEL         PIR1
                CLRF    PIR1                    ;CLEAR ALL PIR1 INTERUPT FLAGS
                BANKSEL         PIR2
                CLRF    PIR2                    ;CLEAR ALL PIR2 INTERUPT FLAGS
                BANKSEL         PIR3
                CLRF    PIR3                    ;CLEAR ALL PIR3 INTERUPT FLAGS
                BANKSEL         PIR4
                CLRF    PIR4                    ;CLEAR ALL PIR4 INTERUPT FLAGS

;*** ALTERNATE PIN FUNCTION *****
```

```
                BANKSEL         APFCON1
                CLRF    APFCON1                 ;SELECTS PORT MAPPING
                CLRF    APFCON2                 ;SEE DATASHEET (pg 132-133) FOR MAPPING
WITH BOTH CLEARED

;*** PORT A SETUP **** PORT A SET AS LOGIC OUTPUT (NOT USED)

                BANKSEL         PORTA
                CLRF    PORTA                   ;CLEAR PORTA
                BANKSEL         LATA
                CLRF    LATA            ;CLEAR LATA
                BANKSEL         TRISA
                MOVLW           0x00
                MOVWF           TRISA                   ;SET PORTA ALL AS OUTPUT
                BANKSEL         ODCONA
                CLRF    ODCONA                  ;DISABLE PORTA OPEN-DRAIN
                BANKSEL         ANSELA
                CLRF    ANSELA                  ;SET PORTA ALL AS DIGITAL INPUT
                BANKSEL         SLRCONA
                CLRF    SLRCONA                 ;SET THE SLEW RATE FOR PORT A TO
MAXIMUM
                BANKSEL         IOCAP
                CLRF    IOCAP                   ;DISABLE PORTA INTERRUPT ON RISING
EDGE
                BANKSEL         IOCAN
                CLRF    IOCAN                   ;DISABLE PORTA INTERRUPT ON FALLING
EDGE

;*** PORT B SETUP **** PORT B SET AS LOGIC OUTPUT (NOT USED)

                BANKSEL         PORTB
                CLRF    PORTB                   ;CLEAR PORTB
                BANKSEL         LATB
                CLRF    LATB            ;CLEAR LATB
                BANKSEL         TRISB
                MOVLW           0x00
                MOVWF           TRISB                   ;SET PORTB ALL AS OUTPUTS
                BANKSEL         ODCONB
                CLRF    ODCONB                  ;DISABLE PORTB OPEN-DRAIN
                BANKSEL         ANSELB
                CLRF    ANSELB                  ;MAKE PORTB DIGITAL INPUTS
                BANKSEL         SLRCONB
                CLRF    SLRCONB                 ;SET THE SLEW RATE FOR PORT B TO
MAXIMUM
                BANKSEL         IOCBP
                CLRF    IOCBP                   ;DISABLE PORTB INTERRUPT ON RISING
EDGE
                BANKSEL         IOCBN
                CLRF    IOCBN                   ;DISABLE PORTB INTERRUPT ON FALLING
EDGE
                BANKSEL         INLVLB
                CLRF    INLVLB                  ;SET PORTB INPUT THRESHOLD TO TTL
(IGNORED NOISE BETTER FOR TRIGGER INPUT)
```

```
;*** PORT C SETUP ****  PORT C SET AS LOGIC INPUT (USED FOR I2C AND UART)

                BANKSEL         PORTC
                CLRF    PORTC                   ;CLEAR PORTC
                BANKSEL         LATC
                CLRF    LATC            ;CLEAR LATC
                BANKSEL         TRISC
                MOVLW           0xFF
                MOVWF           TRISC                   ;SET PORTC ALL AS INPUTS
                BANKSEL         ODCONC
                CLRF    ODCONC                  ;DISABLE PORTC OPEN-DRAIN
                BANKSEL         ANSELC
                CLRF    ANSELC                  ;MAKE PORTC ALL DIGITAL I/O
                BANKSEL         SLRCONC
                CLRF    SLRCONC                 ;SET THE SLEW RATE FOR PORT C TO
MAXIMUM

;*** I2C SETUP ****

                BANKSEL         SSPADD
                MOVLW           0x27                    ;SET BAUD RATE TO 100KHz w/ 16MHz
OSCILLATOR
                MOVWF           SSPADD
                BANKSEL         SSPCON
                MOVLW           0x08                    ;SET TO I2C MASTER MODE
                MOVWF           SSPCON                  ;
                BSF     SSPCON,5                ;ENABLE SERIAL PORT FOR I2C
                BANKSEL         SSPCON2
                CLRF    SSPCON2
                BANKSEL         SSPCON3
                CLRF    SSPCON3
                BANKSEL         SSPSTAT
                MOVLW           0xC0
                MOVWF           SSPSTAT

;*** UART SETUP ****

                BANKSEL         SPBRGH
                MOVLW           0x01                    ;SET UART BAUD RATE TO 9600 W/ 16MHz
OSCILLATOR
                MOVWF           SPBRGH
                BANKSEL         SPBRGL
                MOVLW           0xA0
                MOVWF           SPBRGL
                BANKSEL         BAUDCON
                MOVLW           B'00001000'             ;CONFIGURE FOR 16-BIT BAUD RATE
GENERATOR
                MOVWF           BAUDCON
                BANKSEL         RCSTA
                MOVLW           B'10010000'             ;ENABLE UART PORT, ENABLE
RECEIVER
                MOVWF           RCSTA
                BANKSEL         TXSTA
                MOVLW           B'00100110'             ;ENABLE TRANSMITTER, HIGH SPEED
MODE, ASYNCHRONOUS
```

```
                MOVWF           TXSTA

;*** BOOT CODE ****

                BANKSEL         INTCON
                BCF     INTCON,0
                BCF     INTCON,IOCIF            ;CLEAR IOCIF

                ;CLEAR GENERAL REGISTERS
                BANKSEL         PORTA
                CLRF    ENABLE_TARGET
                CLRF    CURRENT_ADDRESS
                CLRF    INUSE_TARGET


                RETURN                                  ;RETURN TO .asm FILE
```

16.3: Target Slave

16.3.1: Target Slave Assembly Code

```
;*******************************************************************************
;                                                                              *
;  Filename:         LA_Slave.asm                                              *
;  Date:             April 23, 2025                                            *
;  File Version:     2                                                         *
;  Author:           Alex Wheelock and Andrew Keller                                 *
;  Company:          Idaho State University                                     *
;  Description:    Assembly file for the target slave for a laser shooting arcade.                 *
;                    Communicates with the Master via I2C. May be enabled or   *
;                    disabled using I2C commands. When an I2C read is executed, the          *
;                    slave responds with the current status of the target. The I2C           *
;                    address is set during PIC setup utilizing switches connected            *
;                    to PORTA.                                                 *
;                                                                              *
;*******************************************************************************

;*******************************************************************************
;                                                                              *
;  Revision History:                                                           *
;                                                                              *
;  1:       Got everything for the gun working the way that I think it should with    *
;           base features.                                                      *
;                                                                              *
;  2:       I2C communications for enabling/disabling the target as well as reporting *
;        any player hits to the target when the target is enabled.             *
;                                                                              *
;*******************************************************************************

                    #include "SLAVE.inc"                ; processor specific variable definitions
                    #INCLUDE <SLAVE_SETUP.inc>          ; Custom setup file for the PIC16F883
micro-controller
                    #INCLUDE <SLAVE_SUBROUTINES.inc>          ; File containing all used subroutines
                    LIST      p=16f1788                 ; list directive to define processor
                    errorlevel -302,-207,-305,-206,-203 ; suppress "not in bank 0" message,  Found label after column
1,
                                                        ; Using default destination of 1 (file),  Found call to
macro in column 1

                    ; CONFIG1
; __config 0xC9A4
 __CONFIG _CONFIG1, _FOSC_INTOSC & _WDTE_OFF & _PWRTE_OFF & _MCLRE_OFF & _CP_OFF &
_CPD_OFF & _BOREN_OFF & _CLKOUTEN_OFF & _IESO_OFF & _FCMEN_OFF
; CONFIG2
; __config 0xDFFF
 __CONFIG _CONFIG2, _WRT_OFF & _PLLEN_OFF & _STVREN_ON & _BORV_LO & _LPBOR_OFF &
_LVP_OFF



;*****************************************
```

```
;Interrupt Vectors
;*****************************************
                ORG     H'000'                  ;BEGINNING OF CODE
                GOTO    SETUP                           ;
                ORG     H'004'                  ;INTERRUPT LOCATION
                GOTO    INTERRUPT                       ;INTERRUPT OCCURRED, RUN THROUGH ISR

;*****************************************
;SETUP ROUTINE
;*****************************************
SETUP
                CALL    INITIALIZE                      ;CALL SETUP INCLUDE FILE TO INITIALIZE
PIC
                GOTO    MAIN                            ;START MAIN CODE

;*****************************************
;INTERRUPT SERVICE ROUTINE
;*****************************************
INTERRUPT
                BANKSEL         PIR1
                BTFSC   PIR1,3                          ;CHECK SSPIF FLAG TO SEE IF DATA WAS
RECIEVED FROM MASTER
                CALL    RECEIVE                         ;CALL THE RECEIVE SUBROUTINE IF DATA
WAS RECEIVED
                BANKSEL         PORTA
                BANKSEL         IOCBF
                BTFSC   IOCBF,1                         ;IF THE PLAYER 2 IR RECEIVER IS TRIGGERED
(FALLING EDGE INTERRUPT ON CHANGE)
                GOTO    RECORD_PLAYER2_HIT              ;CALL RECORD_PLAYER2_HIT
SUBROUTINE
                BTFSC   IOCBF,0                         ;IF THE PLAYER 1 IR RECEIVER IS TRIGGERED
(FALLING EDGE INTERRUPT ON CHANGE)
                GOTO    RECORD_PLAYER1_HIT              ;CALL RECORD_PLAYER1_HIT
SUBROUTINE
        GOBACK
                BANKSEL         PIR1                    ;NOTE: GOBACK ONLY USED IN I2C
INTERRUPTS. IF USED FOR IOCBF, MISSES I2C REQUESTS AND MASTER CRASHES
                BCF     PIR1,3                          ;CLEAR SSP1IF
                RETFIE                                  ;RETURN TO MAIN, RE-ENABLE GIE
;Main Code
;*****************************************

MAIN
                BANKSEL         PORTA
                BTFSS   ACTIVE, 0                       ;IF THE TARGET IS NOT ACTIVE, TURN OFF
THE LEDS
                CALL    TURN_OFF_LEDS
                BTFSC   ACTIVE, 0                       ;IF THE TARGET IS ACTIVE, TURN ON THE
LEDS
                CALL    TURN_ON_LEDS
                GOTO    MAIN
END

;*****************************************
;END PROGRAM DIRECTIVE
```

16.3.2: Target Slave Subroutine Code

```
;*******************************************************************************
;                                                                              *
;  Filename:          SLAVE_SUBROUTINES.inc                                    *
;  Date:     April 23, 2025                                                   *
;  File Version:   2                                                          *
;  Author:            Alex Wheelock and Andrew Keller                         *
;  Company:           Idaho State University                                  *
;  Description:    Contains all subroutines needed for the Laser Shooting Game *
;                     target slave. Subroutines include I2C communications, recording *
;                     player hits, and enabling/disabling the target. The I2C  *
;                     communications includes the ability to both receive data from  *
;                     the master as well as the ability to respond to read requests  *
;                     from the master. When a hit is detected, the LED is flashed.   *
;                                                                              *
;*******************************************************************************

;*******************************************************************************
;                                                                              *
;  Revision History:                                                           *
;                                                                              *
;  1:      Initialize file, get everything working the way that I thought it should  *
;          work.                                                               *
;                                                                              *
;  2:       I2C communications for enabling/disabling the target as well as reporting *
;        any player hits to the target when the target is enabled.             *
;                                                                              *
;*******************************************************************************

SUBROUTINES_CODE CODE

   ;RECEIVES AND INTERPRETS I2C DATA, RESPONDS TO I2C READ REQUESTS
   RECEIVE:
            BANKSEL     PIR1
            BCF         PIR1, SSP1IF            ;IMMEDIATELY CLEAR INTERRUPT FLAG
            BANKSEL     SSP1BUF                     ;\READ BYTE FROM MASTER
            MOVFW       SSP1BUF                     ;/
            MOVWF       RECEIVED_DATA
            BANKSEL     SSP1STAT                ;DETERMINE IF THE RECEIVED BYTE WAS DATA OR
ADDRESS
            BTFSS       SSP1STAT,5              ;1=DATA, 0=ADDRESS
            GOTO        CHECK_READ_OR_WRITE     ;BYTE RECEIVED WAS ADDRESS, CHECK IF
R/W
            GOTO        SAVE_DATA               ;BYTE RECEIVED WAS DATA, SAVE BYTE

         CHECK_READ_OR_WRITE
            BTFSS       SSP1STAT,2              ;TEST IF ADDRESS BYTE WAS R/W: R = 1, W = 0
            GOTO        SAVE_DATA               ;MASTER WRITE: SAVE DATA
            GOTO        TRANSMIT_STATUS             ;MASTER READ: TRANSMIT THE TARGET
STATUS

         ;TESTS THE RECEIVED DATA AGAINST THE KNOWN ENABLE AND DISABLE COMMANDS
         SAVE_DATA
            BANKSEL     PORTA
```

```
            MOVWF       RECEIVED_DATA
            XORLW       0X45
            BTFSC       STATUS, Z           ;IF RECEIVED BYTE IS ENABLE COMMAND, SET TARGET
TO ACTIVE
            BSF         ACTIVE, 0
            MOVFW       RECEIVED_DATA
            XORLW       0X44
            BTFSC       STATUS, Z           ;IF RECEIVED BYTE IS DISABLE COMMAND, SET TARGET
TO INACTIVE
            BCF         ACTIVE, 0
            RETURN

        ;TRANSMIT THE CURRENT TARGET STATUS BACK TO THE MASTER IF A READ WAS
RECEIVED
        TRANSMIT_STATUS
            BANKSEL     PORTA
            MOVFW       TARGET_STATUS       ;RETREIVE THE CURRENT TARGET STATUS
            BANKSEL     SSP1BUF
            MOVWF       SSP1BUF             ;MOVE THE TARGET STATUS INTO THE I2C BUFFER
            BANKSEL     SSP1CON1
            BSF         SSP1CON1, CKP       ;START THE CLOCK TO BEGIN DATA
TRANSMISSION
            BANKSEL     PORTA
            BTFSC       TARGET_STATUS,3     ;IF THE TARGET_STATUS IS 0X0F, RESET THE
TARGET
            CALL        RESET_TARGET
            BTFSC       TARGET_STATUS,7     ;IF THE TARGET_STATUS IS 0XF0, RESET THE
TARGET
            CALL        RESET_TARGET
            RETURN

        ;CLEARS THE ACTIVE FLAG AND THE TARGET_STATUS REGISTER
        ;PREPARES THE TARGET TO BE ENABLED AGAIN
        RESET_TARGET
            CLRF ACTIVE
            CLRF TARGET_STATUS
            RETURN

   ;WHEN PLAYER ONE'S IR RECEIVER RECEIVES A HIT
   ;IF THE TARGET IS ACTIVE, FLASH THE LEDS FOR PLAYER ONE AND MOVE 0X0F INTO
TARGET_STATUS
   RECORD_PLAYER1_HIT:
            BANKSEL     PORTB
            BTFSS       ACTIVE, 0           ;IF THE TARGET IS NOT ACTIVE, RETURNFROMHIT
            GOTO        RETURNFROMHIT
            CALL        TURN_OFF_LEDS       ;TURN OFF ALL LEDS
            BCF         PORTB,3             ;TURN ON THE RED LEDS
            CALL        WAIT_FOR_TMR1       ;WAIT FOR A MOMENT
            CALL        WAIT_FOR_TMR1
            CALL        WAIT_FOR_TMR1
            CALL        WAIT_FOR_TMR1
            BSF         PORTB,3             ;TURN OFF THE RED LEDS
            CALL        WAIT_FOR_TMR1       ;WAIT FOR A MOMENT
            CALL        WAIT_FOR_TMR1
            CALL        WAIT_FOR_TMR1
```

```
        CALL    WAIT_FOR_TMR1
        BCF     PORTB,3             ;TURN ON THE RED LEDS
        CALL    WAIT_FOR_TMR1       ;WAIT FOR A MOMENT
        CALL    WAIT_FOR_TMR1
        CALL    WAIT_FOR_TMR1
        CALL    WAIT_FOR_TMR1
        BSF     PORTB,3             ;TURN OFF THE RED LEDS
        CALL    WAIT_FOR_TMR1       ;WAIT FOR A MOMENT
        CALL    WAIT_FOR_TMR1
        CALL    WAIT_FOR_TMR1
        CALL    WAIT_FOR_TMR1
        BCF     PORTB,3             ;TURN ON THE RED LEDS
        CALL    WAIT_FOR_TMR1       ;WAIT FOR A MOMENT
        CALL    WAIT_FOR_TMR1
        CALL    WAIT_FOR_TMR1
        CALL    WAIT_FOR_TMR1
        BSF     PORTB,3             ;TURN OFF THE RED LEDS
        BCF     ACTIVE,0        ;DISABLE THE TARGET
        MOVLW   0x0F
        MOVWF   TARGET_STATUS       ;MOVE 0X0F INTO TARGET_STATUS
        GOTO    RETURNFROMHIT

  ;WHEN PLAYER TWO'S IR RECEIVER RECEIVES A HIT
  ;IF THE TARGET IS ACTIVE, FLASH THE LEDS FOR PLAYER TWO AND MOVE 0XF0 INTO
TARGET_STATUS
  RECORD_PLAYER2_HIT:
        BANKSEL PORTB
        BTFSS   ACTIVE, 0       ;IF THE TARGET IS NOT ACTIVE, RETURNFROMHIT
        GOTO    RETURNFROMHIT
        CALL    TURN_OFF_LEDS       ;TURN OFF ALL LEDS
        BCF     PORTB,4             ;TURN ON THE BLUE LEDS
        CALL    WAIT_FOR_TMR1       ;WAIT FOR A MOMENT
        CALL    WAIT_FOR_TMR1
        CALL    WAIT_FOR_TMR1
        CALL    WAIT_FOR_TMR1
        BSF     PORTB,4             ;TURN OFF THE BLUE LEDS
        CALL    WAIT_FOR_TMR1       ;WAIT FOR A MOMENT
        CALL    WAIT_FOR_TMR1
        CALL    WAIT_FOR_TMR1
        CALL    WAIT_FOR_TMR1
        BCF     PORTB,4             ;TURN ON THE BLUE LEDS
        CALL    WAIT_FOR_TMR1       ;WAIT FOR A MOMENT
        CALL    WAIT_FOR_TMR1
        CALL    WAIT_FOR_TMR1
        CALL    WAIT_FOR_TMR1
        BSF     PORTB,4             ;TURN OFF THE BLUE LEDS
        CALL    WAIT_FOR_TMR1       ;WAIT FOR A MOMENT
        CALL    WAIT_FOR_TMR1
        CALL    WAIT_FOR_TMR1
        CALL    WAIT_FOR_TMR1
        BCF     PORTB,4             ;TURN ON THE BLUE LEDS
        CALL    WAIT_FOR_TMR1       ;WAIT FOR A MOMENT
        CALL    WAIT_FOR_TMR1
        CALL    WAIT_FOR_TMR1
        CALL    WAIT_FOR_TMR1
```

```
        BSF     PORTB,4                 ;TURN OFF THE BLUE LEDS
        BCF     ACTIVE,0        ;DISABLE THE TARGET
        MOVLW   0xF0
        MOVWF   TARGET_STATUS           ;MOVE 0XF0 INTO TARGET_STATUS
        GOTO    RETURNFROMHIT

    RETURNFROMHIT
        BANKSEL IOCBF
        CLRF IOCBF                  ;CLEAR THE INTERRUPT ON CHANGE PORTB FLAGS
        RETFIE

;LOOPS UNTIL TMR1 OVERFLOW
WAIT_FOR_TMR1:
        BCF     PIR1,0                  ;CLEAR TMR1IF
        BANKSEL T1CON
        BSF     T1CON,0                         ;ENABLE TIMER 1
        BANKSEL TMR1L
        CLRF TMR1L                  ;/RESET TMR1 FOR FULL TIMING
        CLRF TMR1H                  ;\
        BCF     PIR1,0                  ;CLEAR TMR1IF
        BTFSS   PIR1,0                  ;POLL TMR1IF UNTIL COMPLETE
        GOTO    $-1                     ;TMR1 NOT DONE
        BANKSEL T1CON
        BCF     T1CON,0                         ;DISABLE TIMER 1
        RETURN

;TURNS OFF ALL LEDS ON THE TARGET
TURN_OFF_LEDS:
        BANKSEL PORTB
        BSF     PORTB, RB2
        BSF     PORTB, RB3
        BSF     PORTB, RB4
        RETURN

;TURNS ON THE LEDS ASSOCIATED WITH AN ACTIVE TARGET
TURN_ON_LEDS:
        BANKSEL PORTB
        BCF     PORTB, RB2
        RETURN
```

16.3.3: Target Slave Setup Code

```
;*******************************************************************************
;                                                                              *
;   Filename:           SLAVE_SETUP.inc                                        *
;   Date:     April 23, 2025                                                   *
;   File Version:   2                                                          *
;   Author:             Alex Wheelock and Andrew Keller                        *
;   Company:            Idaho State University                                 *
;   Description:    Firmware for setting up a PIC16F1788 for the target slave of   *
;                       the Laser Shooting Game project.                       *
;                                                                              *
;*******************************************************************************

;*******************************************************************************
;                                                                              *
;   Revision History:                                                          *
;                                                                              *
;   1:      Iniitialize file, setup how I think it should be setup for the project.    *
;                                                                              *
;   2:      I2C communications for enabling/disabling the target as well as reporting   *
;       any player hits to the target when the target is enabled.              *
;                                                                              *
;*******************************************************************************

;================================================================================
================================================*
;                                                   PORTA BITMAP
                                *
;================================================================================
================================================*
;               |               |               |               |               |               |
        |               *
;       7       |       6       |       5       |       4       |       3       |       2       |
1       |       0       *
;---------------|---------------|---------------|---------------|---------------|---------------|---------------*
;DIGITAL        |DIGITAL        |DIGITAL        |DIGITAL        |DIGITAL        |DIGITAL
|DIGITAL        |DIGITAL        *
;INPUT          |INPUT          |INPUT          |INPUT          |INPUT          |INPUT          |INPUT
        |INPUT          *
;---------------|---------------|---------------|---------------|---------------|---------------|---------------*
;I2C Address    |I2C Address    |I2C Address    |I2C Address    |I2C Address    |I2C Address    |I2C
Address |I2C Address    *
;Switch         |Switch         |Switch         |Switch         |Switch         |Switch         |Switch
        |Switch         *
;               |               |               |               |               |               |
        |(Ignored in    *
;               |               |               |               |               |               |
        |7-bit address)     *
;-------------------------------------------------------------------------------------------------------------------------------*

;================================================================================
================================================*
;                                                   PORTB BITMAP
                                *
```

```
;==========================================================================================
===============================================*
;              |              |              |              |              |              |
       |              *
;      7       |      6       |      5       |      4       |      3       |      2       |
1      |      0       *
;--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------*
;DIGITAL       |DIGITAL       |DIGITAL       |DIGITAL       |DIGITAL       |DIGITAL
|DIGITAL       |DIGITAL       *
;INPUT         |INPUT         |INPUT         |OUTPUT                       |OUTPUT
|OUTPUT               |INPUT         |INPUT         *
;--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------*
;UNUSED               |UNUSED                       |UNUSED                       |RGB LEDS BLUE        |RGB
LEDS RED       |RGB LEDS GREEN       |56kHz IR Input  |38kHz IR Input  *
;              |              |              |              |              |              |
       |              *
;              |              |              |              |              |              |
       |              *
;              |              |              |              |              |              |
       |              *
;------------------------------------------------------------------------------------------------------------------------*

;==========================================================================================
===============================================*
;                                                      PORTC BITMAP
                              *
;==========================================================================================
===============================================*
;              |              |              |              |              |              |
       |              *
;      7       |      6       |      5       |      4       |      3       |      2       |
1      |      0       *
;--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------*
;DIGITAL       |DIGITAL       |DIGITAL       |DIGITAL       |DIGITAL       |DIGITAL
|DIGITAL       |DIGITAL       *
;OUTPUT               |OUTPUT                       |OUTPUT                       |OUTPUT
|OUTPUT               |OUTPUT                       |OUTPUT                       |OUTPUT               *
;--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------*
;UNUSED               |UNUSED                       |UNUSED                       |I2C SDA       |I2C SCL
|UNUSED               |UNUSED                       |UNUSED                       *
;              |              |              |              |              |              |
       |              *
;              |              |              |              |              |              |
       |              *
;              |              |              |              |              |              |
       |              *
;------------------------------------------------------------------------------------------------------------------------*

INITIALIZE_CODE       CODE
INITIALIZE

;*** INTOSC SETUP *******

              BANKSEL        OSCCON
              MOVLW          0x7A                  ;ENABLE INTOSC, SET TO 16MHz
```

```
                MOVWF           OSCCON
                BANKSEL         CLKRCON
                CLRF    CLKRCON                 ;DISABLE CLOCK REFERENCE

;*** SET OPTION_REG: ****

                BANKSEL         OPTION_REG
                BCF     OPTION_REG, PS0         ;\\
                BCF     OPTION_REG, PS1         ; >>PRESCALER RATIO SET 1:1
                BCF     OPTION_REG, PS2         ;//
                BSF     OPTION_REG, PSA         ;PRESCALER ASSIGNED TO WDT
                BCF     OPTION_REG, TMR0SE              ;TMR0 EDGE SET RISING
                BCF     OPTION_REG, TMR0CS              ;TMR0 CLOCK SOURCE SET TO
INTERNAL
                BSF     OPTION_REG, INTEDG              ;RB0/INT SET TO RISING EDGE
                BSF     OPTION_REG, NOT_WPUEN   ;WEAK PULLUP ENABLED

;*** SET INTCON REG: ****

                BANKSEL INTCON
                BSF     INTCON, IOCIE           ;ENABLE IOC INTERUPT
                BCF     INTCON, INTE            ;DISABLE INT EXTERNAL INTERUPT
                BSF     INTCON, PEIE            ;ENABLE PERIPHERAL INTERUPTS
                BSF     INTCON, GIE             ;ENABLE ALL UNMASKED INTERUPTS

;*** SET PIE1-4 REG: *****

                BANKSEL         PIE1
                MOVLW           0x08
                MOVWF           PIE1                    ;DISABLE ALL PIE1 INTERUPTS, EXCEPT
SSP1F
                BANKSEL         PIE2
                CLRF    PIE2                    ;DISABLE ALL PIE2 INTERUPTS
                BANKSEL         PIE3
                CLRF    PIE3                    ;DISABLE ALL PIE3 INTERUPTS
                BANKSEL         PIE4
                CLRF    PIE4                    ;DISABLE ALL PIE4 INTERUPTS

;*** CLEAR INTERUPT FLAGS ***

                BANKSEL         PIR1
                CLRF    PIR1                    ;CLEAR ALL PIR1 INTERUPT FLAGS
                BANKSEL         PIR2
                CLRF    PIR2                    ;CLEAR ALL PIR2 INTERUPT FLAGS
                BANKSEL         PIR3
                CLRF    PIR3                    ;CLEAR ALL PIR3 INTERUPT FLAGS
                BANKSEL         PIR4
                CLRF    PIR4                    ;CLEAR ALL PIR4 INTERUPT FLAGS

;*** ALTERNATE PIN FUNCTION *****

                BANKSEL         APFCON1
                CLRF    APFCON1                 ;SELECTS PORT MAPPING
                CLRF    APFCON2                 ;SEE DATASHEET (pg 132-133) FOR MAPPING
WITH BOTH CLEARED
```

```
;*** TMR1 SETUP ****

                BANKSEL         T1CON
                MOVLW           0xA8                    ;ENABLE & SELECT TMR1 LP OSCILLATOR,
SET PRESCALE TO 1:4, DISABLE TMR1 AT THIS POINT
                BANKSEL         T1GCON
                CLRF    T1GCON                          ;DISABLE TMR1 GATE

;*** PORT A SETUP **** PORT A SET AS LOGIC INPUT (I2C ADDRESS)

                BANKSEL         PORTA
                MOVLW           0xFF
                MOVWF           PORTA                           ;CLEAR PORTA
                BANKSEL         LATA
                MOVLW           0xFF                    ;CLEAR LATA
                MOVWF           LATA
                BANKSEL         TRISA
                MOVLW           0xFF
                MOVWF           TRISA                           ;SET PORTA ALL AS INPUTS
                BANKSEL         ODCONA
                CLRF    ODCONA                          ;DISABLE PORTA OPEN-DRAIN
                BANKSEL         ANSELA
                CLRF    ANSELA                          ;SET PORTA ALL AS DIGITAL INPUT
                BANKSEL         SLRCONA
                CLRF    SLRCONA                         ;SET THE SLEW RATE FOR PORT A TO
MAXIMUM
                BANKSEL         IOCAP
                MOVLW           0x00
                MOVWF           IOCAP
                BANKSEL         IOCAN
                CLRF    IOCAN

;*** PORT B SETUP **** PORT B USED AS LOGIC INPUT AND OUTPUT

                BANKSEL         PORTB
                MOVLW           B'00011100'
                MOVWF           PORTB                           ;CLEAR PORTB
                BANKSEL         LATB
                CLRF    LATB                    ;CLEAR LATB
                BANKSEL         TRISB
                MOVLW           B'00000011'
                MOVWF           TRISB                           ;SET PORTB ACCORDING TO BITMAP
                BANKSEL         ODCONB
                CLRF    ODCONB                          ;DISABLE PORTB OPEN-DRAIN
                BANKSEL         ANSELB
                CLRF    ANSELB                          ;MAKE PORTB DIGITAL INPUTS
                BANKSEL         SLRCONB
                CLRF    SLRCONB                         ;SET THE SLEW RATE FOR PORT B TO
MAXIMUM
                BANKSEL         IOCBP
                MOVLW           0x00                    ;DISABLE PORTB IOC FOR POSITIVE EDGES
                MOVWF           IOCBP
                BANKSEL         IOCBN
```

```
                MOVLW           0x03                    ;ENABLE PORTB IOC FOR NEGATIVE EDGES
ON RB0 AND RB1
                MOVWF           IOCBN
                BANKSEL         INLVLB
                CLRF    INLVLB                          ;SET PORTB INPUT THRESHOLD TO TTL
(IGNORED NOISE BETTER FOR TRIGGER INPUT)

;*** PORT C SETUP ****

                BANKSEL         PORTC
                CLRF    PORTC                           ;CLEAR PORTC
                BANKSEL         LATC
                CLRF    LATC                    ;CLEAR LATC
                BANKSEL         TRISC
                MOVLW           0xFF
                MOVWF           TRISC                           ;SET PORTC ALL AS INPUTS
                BANKSEL         ODCONC
                CLRF    ODCONC                  ;DISABLE PORTC OPEN-DRAIN
                BANKSEL         ANSELC
                CLRF    ANSELC                  ;MAKE PORTC ALL DIGITAL I/O
                BANKSEL         SLRCONC
                CLRF    SLRCONC                 ;SET THE SLEW RATE FOR PORT C TO
MAXIMUM
                BANKSEL         IOCCP
                MOVLW           0x00                    ;DISALBE PORTC IOC FOR POSITIVE EDGES
                MOVWF           IOCCP
                BANKSEL         IOCCN
                MOVLW           0x00                    ;DISABLE PORTC IOC FOR NEGATIVE EDGES
                MOVWF           IOCCN
                BANKSEL         INLVLC
                CLRF    INLVLC                          ;SET PORTC INPUT THRESHOLD TO TTL
(IGNORED NOISE BETTER FOR TRIGGER INPUT)

;*** I2C SETUP ****
                BANKSEL         PORTA
                MOVFW           PORTA
                ANDLW           B'11111110'                     ;IGNORE RA0 VALUE
                BANKSEL         SSPADD
                MOVWF           SSPADD                          ;SET I2C ADDRESS ACCORDING TO
SWITCHES
                BANKSEL         SSPCON
                MOVLW           0x16                    ;SET TO I2C 7-BIT ADDRESS SLAVE MODE
                MOVWF           SSPCON                          ;
                BSF     SSPCON,5                        ;ENABLE SERIAL PORT FOR I2C
                BANKSEL         SSPCON2
                MOVLW           0x80
                MOVWF           SSPCON2
                BANKSEL         SSPCON3
                CLRF    SSPCON3
                BANKSEL         SSPSTAT
                MOVLW           0xC0
                MOVWF           SSPSTAT

;*** BOOT CODE ****
```

```
                BANKSEL         PORTA
                CLRF    TARGET_STATUS           ;CLEAR TARGET_STATUS TO PREVENT
INACCURATE SCORE
                CLRF    ACTIVE                  ;ENSURE TARGET STARTS DISABLED
                BANKSEL         INTCON
                BCF     INTCON,0
                BANKSEL         IOCAF
                CLRF    IOCAF
                BCF     INTCON,IOCIF            ;CLEAR IOCIF
                BANKSEL         PORTA


                RETURN                                  ;RETURN TO .asm FILE
```

16.4: Visual Basic Code
16.4.1: Arcade Target Class Code

```
Option Strict On
Option Explicit On

'Laser Arcade Project
'Andrew Keller
'Spring 2025
'https://github.com/AlexWheelock/Laser-Arcade-Project.git

Public Class ArcadeTarget

    '######______Class Properties and variables______######

    ''' <summary>
    ''' Stores the TargetSlot byte for the target
    ''' </summary>
    Private _targetSlot As Byte
    Public ReadOnly Property TargetSlot() As Byte
        Get
            Return _targetSlot
        End Get
    End Property

    Private _ready As Boolean
    ''' <summary>
    ''' Set high the first time the address is set for the target
    ''' </summary>
    ''' <returns></returns>
    Public ReadOnly Property ReadyToEnable As Boolean
        Get
            Return _ready
        End Get
    End Property

    ''' <summary>
    ''' Stores whether or not a target is currently enabled
    ''' </summary>
    Private _enabled As Boolean
    Public Property Enabled() As Boolean
        Get
            Return _enabled
        End Get
        Set(value As Boolean)
            _enabled = value
        End Set
    End Property

    ''' <summary>
    ''' stores the i2C address of the target slot
    ''' </summary>
    Private _i2CAddress As Byte
    Public ReadOnly Property I2CAddress() As Byte
        Get
```

```
        Return _i2CAddress
    End Get
End Property

'######______Subroutines and Functions______######

''' <summary>
''' creates the target with a unique target slot
''' </summary>
''' <param name="slot"></param>
Sub New(slot As Integer)

    'convert the slot number integer into a bit in the Target_Slot byte
    Select Case slot
        Case 1
            _targetSlot = &B_10000000
        Case 2
            _targetSlot = &B_01000000
        Case 3
            _targetSlot = &B_00100000
        Case 4
            _targetSlot = &B_00010000
        Case 5
            _targetSlot = &B_00001000
        Case 6
            _targetSlot = &B_00000100
        Case 7
            _targetSlot = &B_00000010
        Case 8
            _targetSlot = &B_00000001
    End Select

    _enabled = False
End Sub

''' <summary>
''' changes the address of the slot to a new address represented by an integer
''' </summary>
''' <param name="newAddress"></param>
''' <returns></returns>
Public Function ChangeAddress(newAddress As Integer) As Byte()
    Dim _command(3) As Byte
    _command(0) = &H24
    _command(1) = &H41
    _command(2) = _targetSlot
    _command(3) = CByte(newAddress * 2) 'multiply by two to shift the address left by one bit

    _ready = True

    Return _command
End Function

''' <summary>
''' Returns a 3 byte array that commands the master to enable the set target slot
''' </summary>
```

```
    ''' <returns></returns>
    Public Function EnableTarget() As Byte()
        Dim _enableCommand(2) As Byte
        _enableCommand(0) = &H24 'handshake byte
        _enableCommand(1) = &H45 'enable command byte
        _enableCommand(2) = _targetSlot 'TargetSlot byte

        _enabled = True 'once the enable command is sent, set the target to enabled
        Return _enableCommand
    End Function

    ''' <summary>
    ''' Returns a 3 byte array that commands the master to disable the set target slot
    ''' </summary>
    ''' <returns></returns>
    Public Function DisableTarget() As Byte()
        Dim _disableCommand(2) As Byte
        _disableCommand(0) = &H24 'handshake byte
        _disableCommand(1) = &H44 'disable command byte
        _disableCommand(2) = _targetSlot 'target slog byte

        _enabled = False 'once the disable command is sent, set the target to disabled
        Return _disableCommand
    End Function
End Class
```

16.4.2: Laser Arcade Class Code

```
Option Explicit On
Option Strict On
Imports System.IO.Ports

'Laser Arcade Project
'Andrew Keller
'Spring 2025
'https://github.com/AlexWheelock/Laser-Arcade-Project.git

Public Class LaserArcade

    '######______Event Declarations______######

    ''' <summary>
    ''' Raised when the connected COM device fails to verify as the laser arcade master device
    ''' </summary>
    ''' <param name="message"></param>
    Public Event DeviceVerificationFailed(ByVal message As String)

    ''' <summary>
    ''' raised when player one scores on a target. The address is the I2C address associated with the score
    ''' </summary>
    ''' <param name="address"></param>
    Public Event PlayerOneScore(ByVal address As Byte)

    ''' <summary>
    ''' raised when player two scores on a target. The address is the I2C address associated with the score
    ''' </summary>
    ''' <param name="address"></param>
    Public Event PlayerTwoScore(ByVal address As Byte)

    ''' <summary>
    ''' Raised when the COM port disconnects unexpectedly
    ''' </summary>
    ''' <param name="message"></param>
    Public Event UnexpectedDisconnect(ByVal message As String)

    '######______Object With Events Declarations______######

    Private WithEvents arcadePort As New SerialPort
    Private WithEvents verificationTimer As New Timer
    Private WithEvents connectionTimeoutTimer As New Timer
    Private WithEvents unexpectedDisconnectTimer As New Timer

    '######______Class Properties and variables______######

    Private _targets(7) As ArcadeTarget
    Private _disableTimers(7) As Timer

    ''' <summary>
    ''' returns the open or closed status of the arcadePort serial connection
    ''' </summary>
    Public ReadOnly Property Connected() As Boolean
```

```
    Get
        Return arcadePort.IsOpen
    End Get
End Property

''' <summary>
''' get or set the current COM port for the arcade
''' </summary>
''' <returns></returns>
Public Property COMPort() As String
    Get
        Return arcadePort.PortName
    End Get
    Set(value As String)
        If Not arcadePort.IsOpen Then
            arcadePort.PortName = value
        End If
    End Set
End Property

Private _targetTimedDisable As Boolean
''' <summary>
''' If true, the targets will be disabled after a random amount of time in the disable range.
''' </summary>
''' <returns></returns>
Public Property TimedDisable() As Boolean
    Get
        Return _targetTimedDisable
    End Get
    Set(ByVal value As Boolean)
        _targetTimedDisable = value
    End Set
End Property

Private _disableTimeMinimum As Integer
''' <summary>
''' The minimum amount of time a target will remain enabled unless hit by a player in ms.
''' </summary>
''' <returns></returns>
Public Property TimedDisableMinimum() As Integer
    Get
        Return _disableTimeMinimum
    End Get
    Set(ByVal value As Integer)
        _disableTimeMinimum = value
    End Set
End Property

Private _disableTimeMaximum As Integer
''' <summary>
''' The maximum amount of time a target will remain enabled in ms.
''' </summary>
''' <returns></returns>
Public Property TimedDisableMaximum() As Integer
    Get
```

```
        Return _disableTimeMaximum
    End Get
    Set(ByVal value As Integer)
        _disableTimeMaximum = value
    End Set
End Property

''' <summary>
''' If the device was verified as the master of the laser arcade, _verified is true
''' </summary>
Private _verified As Boolean
Public ReadOnly Property DeviceVerified() As Boolean
    Get
        Return _verified
    End Get
End Property

''' <summary>
''' stores the number of I2C targets. Addresses start at 0x01.
''' </summary>
Private _numberOfTargets As Integer
Public Property NumberOfTargets() As Integer
    Get
        Return _numberOfTargets
    End Get
    Set(ByVal value As Integer)
        _numberOfTargets = value
    End Set
End Property

'######______Subroutines and Functions______######

''' <summary>
''' Configures the Laser Arcade class defaults for each instance.
''' </summary>
Sub New()
    'set the COM port Baudrate to 9600 with 8 bits and no parity
    arcadePort.BaudRate = 9600
    arcadePort.DataBits = 8
    arcadePort.Parity = Parity.None

    'sets default timer intervals
    verificationTimer.Interval = 500
    connectionTimeoutTimer.Interval = 10000
    unexpectedDisconnectTimer.Interval = 1000

    'set the disableTime from 5 to 10 seconds as default
    _disableTimeMaximum = 10000
    _disableTimeMinimum = 5000

    'initialize all targets and respective disable timers
    For i = 0 To 7
        _targets(i) = New ArcadeTarget(i + 1)
        _disableTimers(i) = New Timer
        AddHandler _disableTimers(i).Tick, AddressOf DisableTimer_Tick
```

```
        Next
    End Sub

    ''' <summary>
    ''' returns a random number between the max and min
    ''' </summary>
    ''' <param name="max"></param>
    ''' <param name="min"></param>
    ''' <returns></returns>
    Private Function GetRandomNumberBetween(max As Integer, min As Integer) As Integer
        Dim value As Integer

        Randomize(Date.Now.Millisecond)

        value = CInt(Int(Rnd() * ((max - min) + 1))) + min 'finds the difference between max and min, finds a random number in that range, adds min

        Return value
    End Function

    ''' <summary>
    ''' requests verification from the connected device that it is the laser arcade master board
    ''' </summary>
    Private Sub RequestVerification()
        Dim verification(1) As Byte 'the Laser Arcade verification command
        verification(0) = &H24 'handshake byte
        verification(1) = &H56 'command byte

        If arcadePort.IsOpen Then
            Try
                arcadePort.Write(verification, 0, 2) 'send the verification request
            Catch ex As Exception

            End Try
        End If
    End Sub

    ''' <summary>
    ''' opens the Laser arcade serial port connection and starts the device verification process
    ''' </summary>
    Public Sub StartConnection()
        If Not arcadePort.IsOpen Then 'if the serial port is not already open
            Try
                arcadePort.Open() 'open the port
                RequestVerification() 'request verification
            Catch ex As Exception

            End Try

            unexpectedDisconnectTimer.Start()
            verificationTimer.Start() 'start a timer for the verification
            connectionTimeoutTimer.Start() 'start a connection timeout
        End If
    End Sub
```

```
    ''' <summary>
    ''' closes the serial port if it is open
    ''' </summary>
    Public Sub EndConnection()
        If arcadePort.IsOpen Then
            connectionTimeoutTimer.Stop() 'stop the timeout timer
            verificationTimer.Stop() 'stop the verification timer
            arcadePort.Close() 'close the serial port
            _verified = False
        End If
    End Sub

    ''' <summary>
    ''' Enables specified target slot. 0 will enable all slots.
    ''' </summary>
    ''' <param name="slot"></param>
    Public Sub EnableTarget(slot As Integer)
        Select Case slot
            Case 0 'enable all targets
                For i = 0 To 7
                    If _targets(i).ReadyToEnable And arcadePort.IsOpen Then
                        Try
                            arcadePort.Write(_targets(i).EnableTarget(), 0, 3)
                        Catch ex As Exception

                        End Try
                        _disableTimers(i).Interval = GetRandomNumberBetween(_disableTimeMaximum,
_disableTimeMinimum)
                        If _targetTimedDisable Then
                            _disableTimers(i).Start()
                        End If
                    End If
                Next
            Case 1 'enable slot 1
                Try
                    arcadePort.Write(_targets(0).EnableTarget(), 0, 3)
                Catch ex As Exception

                End Try
                _disableTimers(0).Interval = GetRandomNumberBetween(_disableTimeMaximum,
_disableTimeMinimum)
                If _targetTimedDisable Then
                    _disableTimers(0).Start()
                End If
            Case 2 'enable slot 2
                Try
                    arcadePort.Write(_targets(1).EnableTarget(), 0, 3)
                Catch ex As Exception

                End Try
                _disableTimers(1).Interval = GetRandomNumberBetween(_disableTimeMaximum,
_disableTimeMinimum)
                If _targetTimedDisable Then
                    _disableTimers(1).Start()
                End If
```

```
        Case 3 'enable slot 3
            Try
                arcadePort.Write(_targets(2).EnableTarget(), 0, 3)
            Catch ex As Exception

            End Try
            _disableTimers(2).Interval = GetRandomNumberBetween(_disableTimeMaximum,
_disableTimeMinimum)
            If _targetTimedDisable Then
                _disableTimers(2).Start()
            End If
        Case 4 'enable slot 4
            Try
                arcadePort.Write(_targets(3).EnableTarget(), 0, 3)
            Catch ex As Exception

            End Try
            _disableTimers(3).Interval = GetRandomNumberBetween(_disableTimeMaximum,
_disableTimeMinimum)
            If _targetTimedDisable Then
                _disableTimers(3).Start()
            End If
        Case 5 'enable slot 5
            Try
                arcadePort.Write(_targets(4).EnableTarget(), 0, 3)
            Catch ex As Exception

            End Try
            _disableTimers(4).Interval = GetRandomNumberBetween(_disableTimeMaximum,
_disableTimeMinimum)
            If _targetTimedDisable Then
                _disableTimers(4).Start()
            End If
        Case 6 'enable slot 6
            Try
                arcadePort.Write(_targets(5).EnableTarget(), 0, 3)
            Catch ex As Exception

            End Try
            _disableTimers(5).Interval = GetRandomNumberBetween(_disableTimeMaximum,
_disableTimeMinimum)
            If _targetTimedDisable Then
                _disableTimers(5).Start()
            End If
        Case 7 'enable slot 7
            Try
                arcadePort.Write(_targets(6).EnableTarget(), 0, 3)
            Catch ex As Exception

            End Try
            _disableTimers(6).Interval = GetRandomNumberBetween(_disableTimeMaximum,
_disableTimeMinimum)
            If _targetTimedDisable Then
                _disableTimers(6).Start()
            End If
```

```
            Case 8 'enable slot 8
                Try
                    arcadePort.Write(_targets(7).EnableTarget(), 0, 3)
                Catch ex As Exception

                End Try
                _disableTimers(7).Interval = GetRandomNumberBetween(_disableTimeMaximum,
_disableTimeMinimum)
                If _targetTimedDisable Then
                    _disableTimers(7).Start()
                End If
        End Select
    End Sub

    ''' <summary>
    ''' Disables specified target slot. 0 will Disable all slots.
    ''' </summary>
    ''' <param name="slot"></param>
    Public Sub DisableTarget(slot As Integer)
        Dim disableAllCommand(1) As Byte
        disableAllCommand(0) = &H24
        disableAllCommand(1) = &H43

        Select Case slot
            Case 0 'enable all targets
                Try
                    arcadePort.Write(disableAllCommand, 0, 2)
                Catch ex As Exception

                End Try
                For i = 0 To 7
                    If _targets(i).ReadyToEnable And arcadePort.IsOpen Then
                        _disableTimers(i).Stop()
                    End If
                    _targets(i).Enabled = False
                Next
            Case 1 'enable slot 1
                Try
                    arcadePort.Write(_targets(0).DisableTarget(), 0, 3)
                Catch ex As Exception

                End Try
                _disableTimers(0).Stop()
            Case 2 'enable slot 2
                Try
                    arcadePort.Write(_targets(1).DisableTarget(), 0, 3)
                Catch ex As Exception

                End Try
                _disableTimers(1).Stop()
            Case 3 'enable slot 3
                Try
                    arcadePort.Write(_targets(2).DisableTarget(), 0, 3)
                Catch ex As Exception
```

```
            End Try
            _disableTimers(2).Stop()
        Case 4 'enable slot 4
            Try
                arcadePort.Write(_targets(3).DisableTarget(), 0, 3)
            Catch ex As Exception

            End Try
            _disableTimers(3).Stop()
        Case 5 'enable slot 5
            Try
                arcadePort.Write(_targets(4).DisableTarget(), 0, 3)
            Catch ex As Exception

            End Try
            _disableTimers(4).Stop()
        Case 6 'enable slot 6
            Try
                arcadePort.Write(_targets(5).DisableTarget(), 0, 3)
            Catch ex As Exception

            End Try
            _disableTimers(5).Stop()
        Case 7 'enable slot 7
            Try
                arcadePort.Write(_targets(6).DisableTarget(), 0, 3)
            Catch ex As Exception

            End Try
            _disableTimers(6).Stop()
        Case 8 'enable slot 8
            Try
                arcadePort.Write(_targets(7).DisableTarget(), 0, 3)
            Catch ex As Exception

            End Try
            _disableTimers(7).Stop()
    End Select
End Sub

''' <summary>
''' sets the address for and enables a random target
''' </summary>
Public Sub EnableRandomTarget()
    Dim targetSlots As Integer

    'only use up to 7 target slots
    If _numberOfTargets < 8 Then
        targetSlots = _numberOfTargets - 1 'if the number of targets is less than 7, use that many slots
    Else
        targetSlots = 7
    End If

    If arcadePort.IsOpen Then
        For i = 0 To targetSlots
```

```
            If Not _targets(i).Enabled Then 'only attempt changing the address if the target is not enabled
                Try
                    arcadePort.Write(_targets(i).ChangeAddress(GetRandomNumberBetween(_numberOfTargets, 1)), 0,
4) 'change the target address
                    EnableTarget(i + 1) 'enable the target
                Catch ex As Exception

                End Try
                Exit Sub
            End If
        Next
    End If
End Sub

'######_____Event Handlers_____######

''' <summary>
''' When serial data is received, if the device is verified, handle the data. If the device is not verified, check for the
verification
''' </summary>
''' <param name="sender"></param>
''' <param name="e"></param>
Private Sub SerialPort_DataReceived(sender As Object, e As SerialDataReceivedEventArgs) Handles
arcadePort.DataReceived
    Dim bytesToRead As Integer = arcadePort.BytesToRead
    Dim readBytes(bytesToRead - 1) As Byte

    If _verified Then
        If bytesToRead < 3 Then
            Exit Sub 'skip everything if the device is verified and there are not enough bytes to read
        End If
        'read three bytes at a time
        Try
            arcadePort.Read(readBytes, 0, 3)
        Catch ex As Exception

        End Try

        'based on the scoring player, raise the associated event
        Select Case readBytes(2)
            Case &H_01
                RaiseEvent PlayerOneScore(readBytes(1))
            Case &H_02
                RaiseEvent PlayerTwoScore(readBytes(1))
        End Select

        'if a hit target is in one of the target slots, set the target to disabled (slave automatically disabled it)
        For i = 0 To 7
            If _targets(i).I2CAddress = readBytes(1) Then
                _targets(i).Enabled = False
            End If
        Next

    Else
        'if the device has not yet been verified, read the entire buffer and look for verification
```

```
            Try
                arcadePort.Read(readBytes, 0, bytesToRead)
            Catch ex As Exception

            End Try
            For i = 0 To bytesToRead - 1 'for every byte in the read
                If readBytes(i) = &H24 And i + 2 <= bytesToRead - 1 Then 'if the byte is a $ and there are two more bytes
remaining to check
                    If readBytes(i + 1) = &H4C And readBytes(i + 2) = &H41 Then
                        _verified = True 'if the device verification is received, set _verified to True and exit the sub
                        Exit Sub
                    End If
                End If
            Next

            RequestVerification() 'if it was not verified again, request verification
        End If
    End Sub

    ''' <summary>
    ''' sends the RequestVerification command every 500ms when the verificationTimer is enabled. If device is
verified, disable the timer
    ''' </summary>
    ''' <param name="sender"></param>
    ''' <param name="e"></param>
    Private Sub verificationTimer_Tick(sender As Object, e As EventArgs) Handles verificationTimer.Tick
        If _verified Then
            verificationTimer.Stop() 'if the master board COM has been verified, stop the verification timer.
        Else
            RequestVerification() 'send the request verification command
        End If
    End Sub

    ''' <summary>
    ''' If the device verification times out, raise the verification failed event
    ''' </summary>
    ''' <param name="sender"></param>
    ''' <param name="e"></param>
    Private Sub connectionTimeoutTimer_Tick(sender As Object, e As EventArgs) Handles
connectionTimeoutTimer.Tick
        If _verified Then
            connectionTimeoutTimer.Stop() 'if the master board COM was verified, stop the timeout
        Else
            EndConnection() 'end the connection, incorrect device
            RaiseEvent DeviceVerificationFailed("Device verification timed out. Please check the COM port.") 'raise the
DeviceVerificationFailed event
        End If
    End Sub

    ''' <summary>
    ''' Disables a target when its disableTimer goes off.
    ''' </summary>
    ''' <param name="sender"></param>
    ''' <param name="e"></param>
    Private Sub DisableTimer_Tick(sender As Object, e As EventArgs)
```

```
        For i = 0 To 7 'scan through each target/timer
            If _disableTimers(i) Is sender Then 'if the current scan timer sent the tick
                DisableTarget(i + 1) 'disable the current scan timer
                Exit Sub
            End If
        Next
    End Sub

    ''' <summary>
    ''' Checks that the serial port is still open. Raises the UnexpectedDisconnect Event if not.
    ''' </summary>
    ''' <param name="sender"></param>
    ''' <param name="e"></param>
    Private Sub unexpectedDisconnectTimer_Tick(sender As Object, e As EventArgs) Handles
unexpectedDisconnectTimer.Tick
        If Not arcadePort.IsOpen Then
            connectionTimeoutTimer.Stop() 'stop the timeout timer
            verificationTimer.Stop() 'stop the verification timer
            _verified = False
            unexpectedDisconnectTimer.Stop()

            RaiseEvent UnexpectedDisconnect("Laser Arcade Disconnected. COM port closed.")
        End If
    End Sub
End Class
```

16.4.3: Arcade Control Form Code

```
Option Strict On
Option Explicit On
Imports System.ComponentModel
Imports System.IO.Ports

'Laser Arcade Project
'Andrew Keller
'Spring 2025
'https://github.com/AlexWheelock/Laser-Arcade-Project.git

Public Class ArcadeControlForm
    '######______Global Variables______######
    ''' <summary>
    ''' Instance of the LaserArcade class for the form
    ''' </summary>
    Public WithEvents laserArcade As New LaserArcade

    ''' <summary>
    ''' Stores the game time length in seconds.
    ''' </summary>
    Public gameTime As Integer

    ''' <summary>
    ''' Number of seconds left before end of game
    ''' </summary>
    Private gameCountdown As Integer

    ''' <summary>
    ''' True if a game is in progress
    ''' </summary>
    Private gameInProgress As Boolean

    ''' <summary>
    ''' Stores the points scored by player one.
    ''' </summary>
    Private playerOnePoints As Integer

    ''' <summary>
    ''' Stores the points scored by player two
    ''' </summary>
    Private playerTwoPoints As Integer

    ''' <summary>
    ''' stores any point overrides. If not in this array, points default to 1
    ''' 0,n is address. 1,n is point value
    ''' </summary>
    Public pointOverrides(,) As Integer


    '######______Subroutines______######

    ''' <summary>
    ''' If not currently connected to a serial device and the COM select is not open, update the available COM ports
```

```
    ''' </summary>
    Sub UpdateAvailableCOM()
        If Not laserArcade.Connected And Not COMPortComboBox.DroppedDown Then
            COMPortComboBox.Items.Clear()   'clear the Combobox items
            For Each sp As String In My.Computer.Ports.SerialPortNames 'add all available serial ports to the combo
box
                COMPortComboBox.Items.Add(sp)
            Next
            COMPortComboBox.Text = laserArcade.COMPort 'set the combobox text to the current COM port
        End If
    End Sub

    ''' <summary>
    ''' If connect is true, sets the status strip to connected to..., connect/disconnect button to disconnect, lock COM port
selection
    ''' If connect is false, sets the status strip to disconnected from, connect/disconnect button to connect, unlocks
COM port selection
    ''' </summary>
    ''' <param name="connect"></param>
    Sub UpdateSerialControls(connect As Boolean)
        If connect Then
            SerialStatusLabel.Text = $"Connected to {laserArcade.COMPort}" 'set the status bar label
            ConnectDisconnectStatusStripMenuItem.Text = "Disconnect" 'set the connect/disconnect button to
disconnect
            COMPortComboBox.Enabled = False 'disable COM port selection
            COMPortTimer.Enabled = False 'stop refreshing the available COM ports

            'cycle the timer to reset the count
            COMPortTimer.Stop()
            COMPortTimer.Start()
        Else
            ConnectDisconnectStatusStripMenuItem.Text = "Connect" 'set the connect/disconnect button to connect
            SerialStatusLabel.Text = $"Disconnected from {laserArcade.COMPort}" 'set the status bar label
            COMPortComboBox.Enabled = True 'enable COM port selection
            COMPortTimer.Start() 'starts the timer to periodically update the available COM ports
        End If
    End Sub

    ''' <summary>
    ''' Attempts to connect or disconnect from the laser arcade target master.
    ''' If currently disconnected, try connecting.
    ''' If currently connected, try disconnecting.
    ''' </summary>
    Sub ConnectDisconnectSerial()
        If laserArcade.Connected Then
            'Disconnect from serial device
            Try
                laserArcade.EndConnection() 'close the serial port

                'change the connect button text, serial status, and enable the COM selection
                UpdateSerialControls(False)

            Catch ex As Exception 'if failed to disconnect, alert user, change all controls back to
                MsgBox($"Failed to disconnect from device on {laserArcade.COMPort}")
                UpdateSerialControls(True)
```

```
            End Try
        Else
            'connect to serial device
            If COMPortComboBox.Text.Contains("COM") Then 'verify a COM port is selected]
                'set the serial port to the selected COM port
                laserArcade.COMPort = COMPortComboBox.Text

                'open the serial port and change the controls, notify user if failed
                Try
                    laserArcade.StartConnection()
                    UpdateSerialControls(True)
                Catch ex As Exception
                    MsgBox($"Failed to connect to device on {laserArcade.COMPort}")
                    UpdateSerialControls(False) 'set controls back to previous
                    Exit Sub 'skip the rest of the sub
                End Try

            End If
        End If
    End Sub

    ''' <summary>
    ''' Updates the GUI countdown label with the current remaining time
    ''' </summary>
    Private Sub UpdateCountdown()
        Dim minutes As Integer
        Dim seconds As Integer

        minutes = gameCountdown \ 60 'set minutes to the number of minutes remaining
        seconds = gameCountdown - (60 * minutes) 'set seconds to the remainder

        CountdownLabel.Text = $"{minutes}:{CStr(seconds).PadLeft(2, "0"c)}" 'display the countdown as a string
    End Sub


    '######_____Timer Event Handlers_____######

    ''' <summary>
    ''' Refreshes available COM ports when not connected.
    ''' </summary>
    ''' <param name="sender"></param>
    ''' <param name="e"></param>
    Private Sub COMPortTimer_Tick(sender As Object, e As EventArgs) Handles COMPortTimer.Tick
        If Not laserArcade.Connected Then
            'COM port is not connected, refresh available COM ports
            UpdateAvailableCOM()
        End If
    End Sub
=
    ''' <summary>
    ''' When the timer ticks, enable a random target.
    ''' </summary>
    ''' <param name="sender"></param>
    ''' <param name="e"></param>
```

```
    Private Sub TargetEnableTimer_Tick(sender As Object, e As EventArgs) Handles TargetEnableTimer.Tick
        laserArcade.EnableRandomTarget() 'enable a random target
    End Sub

    ''' <summary>
    ''' Occurs every 1000ms.
    ''' If a game is in progress, reduces the countdown and updates the labels.
    ''' If the countdown is 0, stop enabling targets, stop the game timer, and end the game
    ''' </summary>
    ''' <param name="sender"></param>
    ''' <param name="e"></param>
    Private Sub GameTimer_Tick(sender As Object, e As EventArgs) Handles GameTimer.Tick
        If gameInProgress Then
            gameCountdown -= 1 'decrease the game countdown by one
            UpdateCountdown() 'update the countdown label

            If gameCountdown = 0 Then 'if countdown is 0
                TargetEnableTimer.Stop() 'stop enabling targets
                GameTimer.Stop() 'stop counting down
                laserArcade.DisableTarget(0) 'disable all targets
                gameInProgress = False
                CountdownLabel.Text = "Game Over"
                StartStopGameButton.Text = "Start Game"
            End If
        End If
    End Sub

    ''' <summary>
    ''' Updates the scores on the GUI
    ''' </summary>
    ''' <param name="sender"></param>
    ''' <param name="e"></param>
    Private Sub ScoreUpdateTimer_Tick(sender As Object, e As EventArgs) Handles ScoreUpdateTimer.Tick
        PlayerTwoScoreLabel.Text = CStr(playerTwoPoints)
        PlayerOneScoreLabel.Text = CStr(playerOnePoints)
    End Sub

    '######_____LaserArcade Event Handlers_____######

    ''' <summary>
    ''' Occurs when the laserArcade class raises the DeviceVerificationFailed event.
    ''' Changes the labels and buttons back to a disconnected state and alerts the user
    ''' </summary>
    ''' <param name="message"></param>
    Private Sub laserArcadeGame_DeviceVerificationFailed(message As String) Handles
laserArcade.DeviceVerificationFailed
        UpdateSerialControls(False) 'update the labels and buttons
        MsgBox(message) 'alert the user
    End Sub

    ''' <summary>
    ''' Occurs when player one has scored on a target.
    ''' Adds the rewarded points to their score
    ''' </summary>
    ''' <param name="address"></param>
```

```
Private Sub laserArcade_PlayerOneScore(address As Byte) Handles laserArcade.PlayerOneScore
    Dim addressInt As Integer = CInt(address)
    Dim pointsScored As Integer = 1 'default point value of 1

    For i = 0 To (pointOverrides.Length \ 2) - 1
        If addressInt = pointOverrides(0, i) Then 'if the I2C address of the hit target is in points overrides
            pointsScored = pointOverrides(1, i)
        End If
    Next
    playerOnePoints += pointsScored 'add the points scored to the player points

End Sub

''' <summary>
''' Occurs when player two has scored on a target.
''' Adds the rewarded points to their score
''' </summary>
''' <param name="address"></param>
Private Sub laserArcade_PlayerTwoScore(address As Byte) Handles laserArcade.PlayerTwoScore
    Dim addressInt As Integer = CInt(address)
    Dim pointsScored As Integer = 1 'default point value of 1

    For i = 0 To (pointOverrides.Length \ 2) - 1
        If addressInt = pointOverrides(0, i) Then 'if the I2C address of the target has a point override
            pointsScored = pointOverrides(1, i)
        End If
    Next
    playerTwoPoints += pointsScored 'add the points scored to the player points

End Sub

''' <summary>
''' Occurs when the laserArcade class raises the DeviceVerificationFailed event.
''' Changes the labels and buttons back to a disconnected state and alerts the user
''' </summary>
''' <param name="message"></param>
Private Sub laserArcade_UnexpectedDisconnect(message As String) Handles laserArcade.UnexpectedDisconnect
    UpdateSerialControls(False) 'update the labels and buttons
    MsgBox(message) 'alert the user
End Sub

'######_____Form Event Handlers_____######

''' <summary>
''' Configures the defaults on the form when it loads.
''' </summary>
''' <param name="sender"></param>
''' <param name="e"></param>
Private Sub ArcadeControlForm_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    Dim defaultOverrides(1, 0) As Integer
    defaultOverrides(0, 0) = 0
    defaultOverrides(1, 0) = 0

    If Not laserArcade.Connected Then
        UpdateAvailableCOM() 'update the COM selection dropdown
```

```
            UpdateSerialControls(False) 'update the controls to reflect the current SerialPort COM port
        End If

        'set default configuration
        laserArcade.NumberOfTargets = 1 'single target
        laserArcade.TimedDisable = True 'targets turn off with timer
        gameTime = 60 '60 second gametime
        pointOverrides = defaultOverrides 'address 0x00 is worth 0 points
        laserArcade.TimedDisableMinimum = 5000 '5 second minimum target enable time
        laserArcade.TimedDisableMaximum = 10000 '10 second maximum target enable time
        TargetEnableTimer.Interval = 500 'target enabled every 500ms
        ConfigurationStatusStripLabel.Text = "Default Configuration"

        ScoreUpdateTimer.Start()
        COMPortTimer.Start()

    End Sub

    ''' <summary>
    ''' Handles when the connect/disconnect button is pressed
    ''' </summary>
    ''' <param name="sender"></param>
    ''' <param name="e"></param>
    Private Sub ConnectDisconnectStatusStripMenuItem_Click(sender As Object, e As EventArgs) Handles
ConnectDisconnectStatusStripMenuItem.Click
        If Not gameInProgress Then 'prevent changes in connection during game
            'connect or disconnect from the laser arcade
            ConnectDisconnectSerial()
        Else
            MsgBox("Connection cannot be changed with a game in progress.")
        End If
    End Sub

    ''' <summary>
    ''' When the serial icon in the status strip is clicked, open its dropdown
    ''' </summary>
    ''' <param name="sender"></param>
    ''' <param name="e"></param>
    Private Sub SerialStatusStripSplitButton_ButtonClick(sender As Object, e As EventArgs) Handles
SerialStatusStripSplitButton.ButtonClick
        'show the dropdown when the button is pressed
        SerialStatusStripSplitButton.ShowDropDown()
    End Sub

    ''' <summary>
    ''' Handles when the selected COM Port has changed in the ComboBox
    ''' </summary>
    ''' <param name="sender"></param>
    ''' <param name="e"></param>
    Private Sub COMPortComboBox_TextChanged(sender As Object, e As EventArgs) Handles
COMPortComboBox.TextChanged
        'update the controls when the selected COM port changes
        UpdateSerialControls(False)
    End Sub
```

```
    ''' <summary>
    ''' Handles when the start/stop game button is pressed.
    ''' If the game is in progress, the game is stopped.
    ''' If the game is not in progress, the game is started.
    ''' </summary>
    ''' <param name="sender"></param>
    ''' <param name="e"></param>
    Private Sub StartStopGameButton_Click(sender As Object, e As EventArgs) Handles StartStopGameButton.Click
        If gameInProgress Then
            'game is in progress, stop the game
            TargetEnableTimer.Stop() 'stop enabling targets
            GameTimer.Stop() 'stop the game timer
            laserArcade.DisableTarget(0) 'disable all targets
            gameInProgress = False 'mark that a game is no longer in progress
            CountdownLabel.Text = "Game Over" 'change the countdown text
            StartStopGameButton.Text = "Start Game" 'change the button back to start game
        Else
            'Game is not in progress
            If laserArcade.Connected And laserArcade.DeviceVerified Then
                'already connected to the target master board
                gameCountdown = gameTime 'reset the game countdown to the configured game time
                UpdateCountdown() 'update the countdown label

                'reset player points
                playerOnePoints = 0
                playerTwoPoints = 0
                PlayerOneScoreLabel.Text = "0"
                PlayerTwoScoreLabel.Text = "0"

                'start the game
                gameInProgress = True
                laserArcade.EnableRandomTarget() 'enable a target immediately
                GameTimer.Start() 'start the gametimer
                TargetEnableTimer.Start() 'start enabling other targets
                StartStopGameButton.Text = "Stop Game" 'change the button to stop game
            Else
                'not connected to the target master board, alert user
                MsgBox("Please connect to the Laser Arcade COM port.")
            End If
        End If
    End Sub

    ''' <summary>
    ''' Opens the configuration menu and disables this form
    ''' </summary>
    ''' <param name="sender"></param>
    ''' <param name="e"></param>
    Private Sub ConfigurationToolStripMenuItem_Click(sender As Object, e As EventArgs) Handles
ConfigurationToolStripMenuItem.Click
        If gameInProgress Then
            MsgBox("Configuration cannot be changed with a game in progress.")
        Else
            ArcadeConfigurationForm.Show()
            Me.Enabled = False
        End If
```

```
    End Sub

    ''' <summary>
    ''' Reload the form to reset the defaults
    ''' </summary>
    ''' <param name="sender"></param>
    ''' <param name="e"></param>
    Private Sub ResetTopMenuItem_Click(sender As Object, e As EventArgs) Handles ResetTopMenuItem.Click
        If gameInProgress Then
            MsgBox("Configuration cannot be changed with a game in progress.")
        Else
            ArcadeControlForm_Load(sender, e)
        End If
    End Sub

    ''' <summary>
    ''' Closes the form when the exit button is pressed.
    ''' </summary>
    ''' <param name="sender"></param>
    ''' <param name="e"></param>
    Private Sub ExitToolStripMenuItem_Click(sender As Object, e As EventArgs) Handles
ExitToolStripMenuItem.Click
        Me.Close()
    End Sub

    ''' <summary>
    ''' When the form closes and a game is in progress, disable all targets before closing.
    ''' Ensures target system disconnects properly.
    ''' </summary>
    ''' <param name="sender"></param>
    ''' <param name="e"></param>
    Private Sub ArcadeControlForm_Closing(sender As Object, e As CancelEventArgs) Handles Me.Closing
        If gameInProgress Then
            laserArcade.DisableTarget(0)
        End If
    End Sub

    ''' <summary>
    ''' Opens the About Form
    ''' </summary>
    ''' <param name="sender"></param>
    ''' <param name="e"></param>
    Private Sub AboutToolStripMenuItem_Click(sender As Object, e As EventArgs) Handles
AboutToolStripMenuItem.Click
        AboutForm.Show()
    End Sub
End Class
```

16.4.4: Arcade Configuration Form Code

```
Option Strict On
Option Explicit On
Option Compare Text
Imports System.ComponentModel
Imports System.Security

'Laser Arcade Project
'Andrew Keller
'Spring 2025
'https://github.com/AlexWheelock/Laser-Arcade-Project.git

Public Class ArcadeConfigurationForm

    '######______Global Variables______######

    ''' <summary>
    ''' Used to bypass textChanged events during form load
    ''' </summary>
    Private loaded As Boolean

    '######______Subroutines______######

    ''' <summary>
    ''' If the input textbox contains a number, the integer value is returned, else the user is warned and the last
character in the string is removed.
    ''' </summary>
    ''' <param name="inputBox"></param>
    ''' <returns></returns>
    Private Function VerifyNumber(inputBox As TextBox) As Integer
        Dim numberInput As String
        Dim newValue As Integer

        numberInput = inputBox.Text.Replace(" ", "") 'remove spaces

        If Not numberInput = "" Then 'verify the contents are not blank
            If Not "0123456789".Contains(numberInput.Last) Then 'check that the last character entered is a number,
remove it if not and warn user
                MsgBox($"{numberInput.Last} is not a number. Please enter a number.")
                numberInput = numberInput.Substring(0, numberInput.Length - 1)

                inputBox.Text = numberInput 'replace the current input text with the new text
                inputBox.Select(numberInput.Length, 0) 'move cursor back to end of input string
                Return Nothing

            Else 'input is a number
                Try
                    newValue = CInt(numberInput) 'convert to integer
                    Return newValue
                Catch ex As Exception
                    'number did not convert to integer, warn user
                    MsgBox($"{numberInput} is not a number. Please enter a number.")
                    Return Nothing
                End Try
```

```
            End If
        Else
            Return Nothing
        End If
    End Function

    ''' <summary>
    ''' Updates the Points Overrides list box entries to match the control form overrides array
    ''' </summary>
    Private Sub UpdatePointsListBox()
        PointsListBox.Items.Clear() 'clear the current items

        'for every item in the ArcadeControlForm.pointOverrides array, add it to the PointsListBox on this form
        For i = 0 To (ArcadeControlForm.pointOverrides.Length \ 2) - 1
            PointsListBox.Items.Add($"0x{Hex(ArcadeControlForm.pointOverrides(0, i)).PadLeft(2, "0"c)} -
{ArcadeControlForm.pointOverrides(1, i)} Point(s)")
        Next
    End Sub

    ''' <summary>
    ''' Adds a points override to the ArcadeControlForm.pointsOverride array
    ''' </summary>
    ''' <param name="sender"></param>
    ''' <param name="e"></param>
    Private Sub AddPointsButton_Click(sender As Object, e As EventArgs) Handles AddPointsButton.Click
        If loaded Then
            Dim newIndex As Integer = ArcadeControlForm.pointOverrides.Length \ 2
            Dim newOverrides(1, newIndex) As Integer 'dim a new array one index longer than before


            'move the entered data to the overrides
            If AddressTextBox.Text <> "" And ValueTextBox.Text <> "" Then
                'load current overrides into the new overrides
                For i = 0 To 1
                    For j = 0 To newIndex - 1
                        newOverrides(i, j) = ArcadeControlForm.pointOverrides(i, j)
                    Next
                Next

                'Add the new override to the array
                Try
                    newOverrides(0, newIndex) = Convert.ToInt32(AddressTextBox.Text, 16)
                    newOverrides(1, newIndex) = CInt(ValueTextBox.Text)

                    'load the newOverrides into the main overrides, clear the inputs
                    ArcadeControlForm.pointOverrides = newOverrides
                    AddressTextBox.Text = ""
                    ValueTextBox.Text = ""
                    ArcadeControlForm.ConfigurationStatusStripLabel.Text = "Modified Configuration"
                Catch ex As Exception
                    MsgBox("I2C address or Points value are not formatted correctly.")
                End Try

                'update the listbox
                UpdatePointsListBox()
```

```
            Else
                MsgBox("Please enter an I2C address and Point value")
            End If
        End If
    End Sub

    '######_____Form Event Handlers_____######

    ''' <summary>
    ''' Verifies that the I2C address entered is a valid Hex input.
    ''' </summary>
    ''' <param name="sender"></param>
    ''' <param name="e"></param>
    Private Sub AddressTextBox_TextChanged(sender As Object, e As EventArgs) Handles
AddressTextBox.TextChanged
        Dim hexInput As String

        hexInput = AddressTextBox.Text.Replace(" ", "") 'remove spaces
        hexInput = hexInput.ToUpper 'make the entire thing uppercase
        If Not hexInput = "" Then
            'if the input contains characters not used in hex, remove the last character and warn the user
            If Not "ABCDEF0123456789".Contains(hexInput.Last) Then
                MsgBox($"{hexInput.Last} is not used in Hex. Please enter a Hex value.")
                hexInput = hexInput.Substring(0, hexInput.Length - 1)
            End If

            'if the input hex is greater than one byte in length, remove the last character and warn the user
            If hexInput.Length > 2 Then
                MsgBox("The I2C address should not be greater than one byte.")
                hexInput = hexInput.Substring(0, hexInput.Length - 1)
            End If

            AddressTextBox.Text = hexInput 'replace the current input text with the new text
            AddressTextBox.Select(hexInput.Length, 0) 'move cursor back to end of input string
        End If
    End Sub

    ''' <summary>
    ''' Verifies the entered game time is a valid number and modifies the current configuration
    ''' </summary>
    ''' <param name="sender"></param>
    ''' <param name="e"></param>
    Private Sub GameTimeTextBox_TextChanged(sender As Object, e As EventArgs) Handles
GameTimeTextBox.TextChanged
        If loaded Then
            If Not VerifyNumber(GameTimeTextBox) = Nothing Then 'if the entered time is a valid number
                ArcadeControlForm.gameTime = VerifyNumber(GameTimeTextBox) 'change the configuration
                ArcadeControlForm.ConfigurationStatusStripLabel.Text = "Modified Configuration"
            End If
        End If
    End Sub

    ''' <summary>
    ''' Verifies the entered number of targets is valid for the laser arcade
    ''' </summary>
```

```
    ''' <param name="sender"></param>
    ''' <param name="e"></param>
    Private Sub TargetNumberTextBox_TextChanged(sender As Object, e As EventArgs) Handles
TargetNumberTextBox.TextChanged
        Dim targetNumber As Integer 'from 1 to 127

        If loaded Then
            If Not VerifyNumber(TargetNumberTextBox) = Nothing Then 'make sure the input is actually a number
                targetNumber = VerifyNumber(TargetNumberTextBox)

                If targetNumber > 0 And targetNumber <= 127 Then 'if the number of targets is from 1 to 127
                    ArcadeControlForm.laserArcade.NumberOfTargets = targetNumber 'update the configuration
                    ArcadeControlForm.ConfigurationStatusStripLabel.Text = "Modified Configuration" 'change the status
to modified config
                Else
                    MsgBox("The Laser Arcade does not support less than 1 or more than 127 targets.")
                End If
            End If
        End If
    End Sub

    ''' <summary>
    ''' Sets all labels and controls of the configuration form to match the current configuration
    ''' </summary>
    ''' <param name="sender"></param>
    ''' <param name="e"></param>
    Private Sub ArcadeConfigurationForm_Load(sender As Object, e As EventArgs) Handles Me.Load
        UpdatePointsListBox() 'update the points overrides listbox
        GameTimeTextBox.Text = CStr(ArcadeControlForm.gameTime) 'set the gametime textbox to the current game
time
        TargetNumberTextBox.Text = CStr(ArcadeControlForm.laserArcade.NumberOfTargets) 'sets the number of
targets textbox to the current number of targets
        TargetEnableTextBox.Text = CStr(ArcadeControlForm.TargetEnableTimer.Interval) 'sets the
TargetEnableTextBox to match the current interval

        'adjust the radio buttons according to current configuration
        If ArcadeControlForm.laserArcade.TimedDisable Then
            TargetAutoDisableRadioButton.Checked = True
            TargetRemainEnabledRadioButton.Checked = False
            AutoDisableGroupBox.Enabled = True 'enable the max and min inputs
        Else
            TargetAutoDisableRadioButton.Checked = False
            TargetRemainEnabledRadioButton.Checked = True
            AutoDisableGroupBox.Enabled = False 'disable the max and min inputs
        End If

        'set the max and min disable time inputs to match the current configuration
        AutoDisableMaximumTextBox.Text = CStr(ArcadeControlForm.laserArcade.TimedDisableMaximum)
        AutoDisableMinimumTextBox.Text = CStr(ArcadeControlForm.laserArcade.TimedDisableMinimum)

        loaded = True
    End Sub

    ''' <summary>
    ''' Removes a points override from the ArcadeControlForm.pointsOverride array.
```

```
    ''' </summary>
    ''' <param name="sender"></param>
    ''' <param name="e"></param>
    Private Sub RemovePointsButton_Click(sender As Object, e As EventArgs) Handles RemovePointsButton.Click
        If loaded Then
            Dim newLength As Integer = ArcadeControlForm.pointOverrides.Length \ 2 - 2
            Dim newOverrides(1, newLength) As Integer 'dim a new array one index shorter than before
            Dim offset As Integer = 0

            'move the modified data to the overrides array
            If PointsListBox.SelectedItem IsNot Nothing Then 'verify that an override has been selected

                'load current overrides into the new overrides, skipping selected index
                For i = 0 To 1
                    offset = 0
                    For j = 0 To newLength
                        If j >= PointsListBox.SelectedIndex Then
                            offset = 1
                        End If
                        newOverrides(i, j) = ArcadeControlForm.pointOverrides(i, j + offset)
                    Next
                Next

                'load the newOverrides into the main overrides
                ArcadeControlForm.pointOverrides = newOverrides

                'update the listbox
                UpdatePointsListBox()
                ArcadeControlForm.ConfigurationStatusStripLabel.Text = "Modified Configuration"
            Else
                'if an override has not been selected
                MsgBox("Please select an override to remove.")
            End If
        End If
    End Sub

    ''' <summary>
    ''' Verifies the entered target enable interval is a valid number and updates the configuration
    ''' </summary>
    ''' <param name="sender"></param>
    ''' <param name="e"></param>
    Private Sub TargetEnableTextBox_TextChanged(sender As Object, e As EventArgs) Handles
TargetEnableTextBox.TextChanged
        If loaded Then
            If Not VerifyNumber(TargetEnableTextBox) = Nothing Then 'if the entered value is a valid number
                ArcadeControlForm.TargetEnableTimer.Interval = VerifyNumber(TargetEnableTextBox) 'update the
configuration
                ArcadeControlForm.ConfigurationStatusStripLabel.Text = "Modified Configuration"
            End If
        End If
    End Sub

    ''' <summary>
    ''' Verifies the entered disable minimum is a valid number and updates configuration
    ''' </summary>
```

```
    ''' <param name="sender"></param>
    ''' <param name="e"></param>
    Private Sub AutoDisableMinimumTextBox_TextChanged(sender As Object, e As EventArgs) Handles
AutoDisableMinimumTextBox.TextChanged
        If loaded Then
            If Not VerifyNumber(AutoDisableMinimumTextBox) = Nothing Then 'if the input is a valid number
                ArcadeControlForm.laserArcade.TimedDisableMinimum =
VerifyNumber(AutoDisableMinimumTextBox) 'update configuration
                ArcadeControlForm.ConfigurationStatusStripLabel.Text = "Modified Configuration"
            End If
        End If
    End Sub

    ''' <summary>
    ''' Verifies the entered disable maximum is a valid number and updates current configuration
    ''' </summary>
    ''' <param name="sender"></param>
    ''' <param name="e"></param>
    Private Sub AutoDisableMaximumTextBox_TextChanged(sender As Object, e As EventArgs) Handles
AutoDisableMaximumTextBox.TextChanged
        If loaded Then
            If Not VerifyNumber(AutoDisableMaximumTextBox) = Nothing Then 'if input is a valid number
                ArcadeControlForm.laserArcade.TimedDisableMaximum =
VerifyNumber(AutoDisableMaximumTextBox) 'update configuration
                ArcadeControlForm.ConfigurationStatusStripLabel.Text = "Modified Configuration"
            End If
        End If
    End Sub

    ''' <summary>
    ''' Updates configuration when the radio buttons are changed. Disables the min and max inputs if timed disable is
false
    ''' </summary>
    ''' <param name="sender"></param>
    ''' <param name="e"></param>
    Private Sub TargetAutoDisableRadioButton_CheckedChanged(sender As Object, e As EventArgs) Handles
TargetAutoDisableRadioButton.CheckedChanged
        If loaded Then
            If TargetAutoDisableRadioButton.Checked Then
                ArcadeControlForm.laserArcade.TimedDisable = True
                AutoDisableGroupBox.Enabled = True 'enable the max and min inputs
            Else
                ArcadeControlForm.laserArcade.TimedDisable = False
                AutoDisableGroupBox.Enabled = False 'disable the max and min inputs
            End If
            ArcadeControlForm.ConfigurationStatusStripLabel.Text = "Modified Configuration"
        End If
    End Sub

    ''' <summary>
    ''' Saves the current configuration to file
    ''' </summary>
    ''' <param name="sender"></param>
    ''' <param name="e"></param>
    Private Sub SaveButton_Click(sender As Object, e As EventArgs) Handles SaveButton.Click
```

```
        'setup the SaveFileDialog to appear to the user
        SaveFileDialog.DefaultExt = ".config"
        SaveFileDialog.Title = "Save Configuration"
        SaveFileDialog.Filter = "Config files (*.config)|*.config|All files (*.*)|*.*"
        SaveFileDialog.FileName =
$"LaserArcadeConfig_{DateTime.Today.Now.ToString("yyyy-MM-dd-hhmm")}.config"
        SaveFileDialog.InitialDirectory = "../"

        'if the user clicked OK in the SaveFileDialog
        If SaveFileDialog().ShowDialog = DialogResult.OK Then
            FileOpen(1, SaveFileDialog.FileName, OpenMode.Output) 'open the new file as an output
            Write(1, ArcadeControlForm.gameTime) 'add the game time to the file
            Write(1, ArcadeControlForm.laserArcade.NumberOfTargets) 'add the targets to the file
            Write(1, ArcadeControlForm.TargetEnableTimer.Interval) 'add the enable interval to file
            Write(1, ArcadeControlForm.laserArcade.TimedDisable) 'add if the timed disable is on or not
            Write(1, ArcadeControlForm.laserArcade.TimedDisableMinimum) 'add minimum enable time
            WriteLine(1, ArcadeControlForm.laserArcade.TimedDisableMaximum) 'add maximum enable time
            For i = 0 To (ArcadeControlForm.pointOverrides.Length \ 2) - 1 'add the points overrides on a new line for
each
                Write(1, ArcadeControlForm.pointOverrides(0, i))
                WriteLine(1, ArcadeControlForm.pointOverrides(1, i))
            Next
            FileClose(1) 'close the file

            ArcadeControlForm.ConfigurationStatusStripLabel.Text = Split(SaveFileDialog.FileName, "\").Last 'change
the configuration status label on the control form
        End If
    End Sub

    ''' <summary>
    ''' opens a previously saved configuration from file
    ''' </summary>
    ''' <param name="sender"></param>
    ''' <param name="e"></param>
    Private Sub OpenButton_Click(sender As Object, e As EventArgs) Handles OpenButton.Click
        Dim currentLine As String
        Dim addressQueue As New Queue(Of Integer)
        Dim pointQueue As New Queue(Of Integer)

        'setup the openfiledialog to be shown to the user
        OpenFileDialog.Title = "Open Configuration"
        OpenFileDialog.Filter = "Config files (*.config)|*.config|All files (*.*)|*.*"
        OpenFileDialog.InitialDirectory = "../"

        'if the user clicked okay in the dialog
        If OpenFileDialog.ShowDialog = DialogResult.OK Then
            FileOpen(1, OpenFileDialog.FileName, OpenMode.Input) 'open the file as an input
            Input(1, ArcadeControlForm.gameTime) 'read the gametime
            Input(1, ArcadeControlForm.laserArcade.NumberOfTargets) 'read the number of targets
            Input(1, ArcadeControlForm.TargetEnableTimer.Interval) 'read the enable interval
            Input(1, ArcadeControlForm.laserArcade.TimedDisable) 'read whether or not the targets auto-disable
            Input(1, ArcadeControlForm.laserArcade.TimedDisableMinimum) 'read the minimum enable time
            Input(1, ArcadeControlForm.laserArcade.TimedDisableMaximum) 'read the maximum enable time
            Do Until EOF(1) 'read any and all points overrides and put them in a queue
                currentLine = LineInput(1)
```

```
            addressQueue.Enqueue(CInt(Split(currentLine, ",")(0)))
            pointQueue.Enqueue(CInt(Split(currentLine, ",")(1)))
        Loop

        'create a new points override array
        Dim newPointOverride(1, addressQueue.Count - 1) As Integer

        'load the input points overrides into the array
        For i = 0 To addressQueue.Count - 1
            newPointOverride(0, i) = addressQueue.Dequeue
            newPointOverride(1, i) = pointQueue.Dequeue
        Next

        'set the current pointOverrides to the new overrides
        ArcadeControlForm.pointOverrides = newPointOverride
        loaded = False 'bypass textChanged events for the Configuration form
        ArcadeControlForm.ConfigurationStatusStripLabel.Text = Split(OpenFileDialog.FileName, "\").Last 'update
the config status label

        FileClose(1) 'close the file
        ArcadeConfigurationForm_Load(sender, e) 'reload the configuration form
    End If
End Sub

''' <summary>
''' Re-enables the ArcadeControlForm when this form closes
''' </summary>
''' <param name="sender"></param>
''' <param name="e"></param>
Private Sub ArcadeConfigurationForm_Closing(sender As Object, e As CancelEventArgs) Handles Me.Closing
    ArcadeControlForm.Enabled = True
End Sub
End Class
```

16.4.5: About Form Code

```
Option Strict On
Option Explicit On

'Laser Arcade Project
'Andrew Keller
'Spring 2025
'https://github.com/AlexWheelock/Laser-Arcade-Project.git

Public Class AboutForm
    Private Sub OkButton_Click(sender As Object, e As EventArgs) Handles OkButton.Click
        Me.Close()
    End Sub

    Private Sub GithubLinkLabel_LinkClicked(sender As Object, e As LinkLabelLinkClickedEventArgs) Handles
GithubLinkLabel.LinkClicked
        Dim githubAddress As String = "https://github.com/AlexWheelock/Laser-Arcade-Project.git"

        Process.Start(githubAddress)
    End Sub
End Class
```

17: Appendix G (GitHub)

There is a GitHub repository for the Laser Arcade project which can be found by scanning the QR code shown in Figure 16.1 below, or by clicking the following link: https://github.com/AlexWheelock/Laser-Arcade-Project.git. It contains all necessary files for future work on the project including all KiCad files, assembly code, VB programs, etc.

Figure 17.1: Laser Arcade GitHub Repository QR Code

18: Appendix H: References

Awasthi, Abhishek Kumar, et al. "Circular economy and electronic waste." *Nature Electronics* 2.3 (2019): 86-89.

Doh, Kiesha Fraser, et al. "The relationship between parents' reported storage of firearms and their children's perceived access to firearms: a safety disconnect." *Clinical pediatrics* 60.1 (2021): 42-49.

Games and exercises for teaching gun safety. Green Line Arms. (2024, September 20). https://greenlinearms.com/games-and-exercises-for-teaching-gun-safety/