

PROJECT REPORT ON **LASER BLASTER**

SUBMITTED BY:

C LALDINLIANA ROLL NO. 2223BCA002

JOSHUA VANLALHNEHPUIA ROLL NO. 2223BCA007

RINSAWMZELA THLOHTHANG ROLL NO. 2223BCA017

Under the guidance of:

Mr. H. Lalruatkima

Assistant Professor

**Bachelor of Computer Application
(HATIM)**



MIZORAM UNIVERSITY, TANHRIL: AIZAWL

2024

HIGHER AND TECHNICAL INSTITUTE, MIZORAM
KAWMZAWL, LUNGLEI – 796701



CERTIFICATE

This is to certify that these records of the course “Minor Project” with course code **BCA/5/CC/29** submitted for the partial fulfilment of the requirement **BCA V Semester Examination** is an authentic and original work carried out under my supervision. Detail of student is given below:

Name: C Laldinliana

Roll No. 2223BCA002

Name: Joshua Vanlalhnehpuia

Roll No. 2223BCA007

Name: Rinsawmzela Thlohthang

Roll No. 2223BCA017

(H. LALRUATKIMA)

Project Guide

(K. LALMUANPUIA)

Head of Department

ACKNOWLEDGEMENT

First, we would like to give thanks to God for guiding us throughout the process of our project, giving us with good health so that we are able to get this far.

We thank to our Principal Mr. Vuansanga Vanchhawng. We thank him for the support, encouragement and permission, which we enjoy freely in order to accomplish our project, and in this time of Pandemic situation.

Our heartfelt gratitude goes to Mr. H. Lalruatkima, Project Guide and Mr. K. Lalmuanpuia, Head of Department in Department of Computer Science, Higher and Technical Institute, Mizoram for their continual advice, backup, confidence, patience and their encouraging words which gives us courage and confidence without which we would be nowhere.

We thank all the people in the Department of Computer Science for being always available and helpful over the semester.

Last but not the least; we thank our parents for their moral support and encouragement.

CONTENTS

Fig. No.	Figure Name	Page
	Introduction	1
	Objective of the Project	2
	Need of the System	3-4
	Feasibility Study	5-7
1	HARDWARE REQUIREMENTS	
	Laser Blaster System	8-10
	Arduino Uno	8
	LED Bar Graph Display	8
	Push Button	9
	Laser(638 nm)	9
	Vibrator Motor	10
	Target Module	11-13
	Arduino Uno	11
	Laser Receiver Module Non-Modulator Tub Sensor Output	11
	Servo Motor	11
	OLED Display	12
	TM1637 4 Segment Display	12
	DS-213K-SPST Push Button Momentary Switch	13
2	SOFTWARE REQUIREMENTS	
	Arduino IDE	14
	U9g2 Library	14
	Servo Library	15
	TM1637 library	16
	Custom Functions and Configurations	16-17
3	DIAGRAMS	
	Entity Relationship Model	18
	Data Flow Diagram	19-20
4	INPUT AND OUTPUT SCREEN ALONG WITH CODE	
	Code for Laser Module	21-28
	Code for Target Module	29-37
5	PHYSICAL COMPONENTS	
	Physical Components of Laser Module	38-41

	Physical Components of Target Module	42-44
	Additional Notes	45-47
6	LIMITATIONS OF PROJECT	
	Accuracy of Laser Detection	48
	Limited Ammunition Representation	48
	Response Time and Servo Movement	48
	Power Consumption	49
	Score and Time Limit Constraints	49
7	FUTURE SCOPE OF THE PROJECT	
	Improved Laser Detection with Photodiodes or IR Sensor	50
	Wireless Communication and Multi Target Setup	50
	Automatic Ammo Reload Mechanism	51
	Expanded Display Capabilities	51
	Battery Optimization and Renewable Power Sources	51-52
	Game Modes and Scoreboard Integration	52
8	BIBLIOGRAPHY	
	Arduino Documentation and Guides	53
	Servo Motor Control with Arduino	53
	OLED Display Library Documentation (U8g2)	53
	TM1637 4-Digit Display Interface	54
	Technical Datasheets	54
9	ADDITIONAL CODE AND PROJECT REFERENCES	
	Laser Module Integration	55
	Light Dependent Resistor	55
	LED Bar Graph and Visual	56
	Power Optimization Strategies	56
	Servo Motor Mechanics	56
	Wireless Communication and Multiplayer Enhancement	57
	Advanced Displays for Enhanced User Feedback	57
	Game Logic and Software Design	57
	Community Contribution and Troubleshooting	57-58

INTRODUCTION

We, the members of the project team, are thrilled to present our work on the **Target Module for Laser Blaster Game**. This project has been a collaborative journey of learning, creativity, and problem-solving, as we explored the intricacies of integrating hardware and software to create a hands-on interactive experience. Throughout this journey, we aimed not only to build a functional system but also to deepen our understanding of embedded systems, sensor technology, and digital game mechanics.

In recent years, interactive gaming systems have gained popularity, combining elements of technology and entertainment to create immersive experiences. This project, the **Target Module for Laser Blaster Game**, is designed to offer an engaging and skill-testing game in which players aim a laser blaster at multiple targets. The project integrates a range of electronic components, such as servo motors, LDR (Light-Dependent Resistor) sensors, and Arduino-based microcontrollers, to bring this interactive setup to life. When a target is hit by the laser, the target hides momentarily, rewarding the player by increasing their score, which is displayed on a digital screen. A timer tracks the duration of gameplay, and ammunition is visually represented by an LED bar display that depletes as shots are fired, adding an additional layer of strategy and challenge to the game.

This project not only provides entertainment but also serves as a hands-on application of electronic engineering, embedded systems, and basic programming concepts. The design of this system aims to balance fun with skill development, requiring players to be precise in their aim and quick in their decision-making. The project holds educational value as it introduces concepts of sensor integration, actuator control, and real-time data processing, making it suitable for students and hobbyists interested in electronics and game development.

Thank you for taking the time to review our project. We are excited to share our journey with you and hope you find it as engaging as we found it to build.

Objectives of the Project

The primary objectives of this project, **Target Module for Laser Blaster Game**, are as follows:

- **Develop an Interactive Target System:**
Design a responsive target module capable of detecting laser hits using Light-Dependent Resistors (LDRs) and activating servos to simulate target movement.
- **Implement Real-Time Score Tracking:**
Integrate a display system to record and update scores instantly when a target is successfully hit, enhancing user engagement and providing immediate feedback.
- **Utilize Sensor and Motor Integration:**
Combine LDR sensors, servos, and Arduino controls to create an integrated and efficient target detection mechanism.
- **Enhance Practical Understanding of Embedded Systems:**
Apply theoretical knowledge in embedded systems by working with various components such as LDR sensors, servos, and displays in a real-world application.
- **Provide a User-Friendly and Engaging Experience:**
Ensure that the module is intuitive and engaging, with clear visual feedback and dynamic movement to improve the gaming experience for users.
- **Enable Future Expansion and Scalability:**
Design the system to allow for easy modifications and additions, such as expanding the number of targets or integrating additional feedback elements like sound or vibration.

Need for the System

The **Target Module for Laser Blaster Game** project addresses a clear need for interactive and engaging gaming systems that integrate real-time feedback and precision tracking. As gaming technology advances, the demand for immersive, skill-based experiences has grown significantly. Our system fills this gap by creating a dynamic target response module, enhancing the excitement and realism of shooting-based games.

This system is necessary to provide:

- **Enhanced Gaming Experience:**
Traditional shooting games often lack real-time feedback mechanisms that accurately respond to the player's actions. This project introduces a system that uses LDR sensors to detect laser hits and servos to create reactive target movements, simulating real-life scenarios and improving user engagement.
- **Practical Application of STEM Skills:**
By designing, programming, and implementing this system, users can gain practical experience in embedded systems, circuit design, and sensor integration. This educational aspect encourages hands-on learning and demonstrates the application of technology in creating interactive systems.
- **Increased Accuracy and Efficiency in Target Recognition:**
Our system utilizes sensors with specific thresholds for laser detection, ensuring that target movements occur only upon successful hits. This feature not only adds to the game's difficulty level but also improves accuracy and reliability, providing a fair and skill-based challenge for users.
- **Flexibility and Future Scalability:**
The project allows for easy adjustments, such as the addition of

new targets or extra sensors, ensuring it can evolve as technology advances and user expectations grow. This flexibility provides an adaptable solution that can meet various gaming or educational needs over time.

- **Entertainment and Educational Value:**

In both recreational and educational environments, this system can engage users by combining entertainment with an interactive learning experience.

Feasibility Study

The feasibility study evaluates the viability of this project across multiple dimensions: technical, operational, economic, legal, and schedule feasibility. Each aspect is considered to determine the project's potential for success and sustainability.

- **Technical Feasibility:**
 - **Hardware Compatibility:** The project relies on components like Arduino Uno, servos, LED bar graph display, and laser diodes, which are compatible with one another and support easy programming through the Arduino IDE.
 - **Software Flexibility:** By using the Arduino IDE and supporting libraries, such as the Servo, U8g2, and TM1637 Display libraries, the system is equipped with the tools needed to manage different components in real-time. The IDE's integration capabilities simplify the development and troubleshooting processes.
 - **Scalability:** The system can be expanded with additional sensors or outputs (e.g., more LEDs or sensors) due to Arduino's flexibility, allowing future modifications to enhance or upgrade the project.

- **Operational Feasibility:**
 - **Ease of Use:** The system provides straightforward interaction with intuitive controls (trigger, reload, and scoring feedback). The OLED and LED bar graph displays provide real-time feedback, making the system easy to understand and engage with.
 - **Maintenance Requirements:** The modular nature of the design allows individual components to be replaced or upgraded with minimal disruption, enhancing its long-term operational feasibility.
 - **User Learning Curve:** The project's design and operation require minimal technical knowledge from users, making

it accessible to a broader audience, including those without a technical background.

-
- **Economic Feasibility:**
 - **Low Initial Cost:** The hardware components are affordable and widely available, keeping the project's startup costs low.
 - **Cost-Benefit Analysis:** The learning benefits and interactive nature of the project add value to the investment. Additionally, the use of open-source software eliminates the need for costly software licensing.
 - **Minimal Maintenance Cost:** Replacements for components like LEDs, resistors, and wiring are inexpensive, ensuring low maintenance costs over time.
- **Legal Feasibility:**
 - **Safety Compliance:** All components adhere to safety standards for educational and hobbyist use. Care is taken to use low-power components (e.g., 5V-12V power levels), reducing risk during operation.
 - **Open-Source Software Licensing:** The Arduino IDE and libraries used are open-source, ensuring no legal restrictions or fees associated with software usage in the project.
 - **Environmental Compliance:** Components chosen (e.g., reusable electronics and minimal waste materials) align with environmentally responsible practices, especially relevant for educational and project-based applications.
- **Schedule Feasibility:**
 - **Realistic Timeline:** Given the well-defined components and straightforward assembly, the project can be realistically completed within the proposed timeframe, including time for testing and refinement.

- **Minimal Development Delays:** The availability of online resources, tutorials, and support for Arduino and its components minimizes potential delays during development.
- **Support and Documentation:** Extensive Arduino documentation and community support reduce time spent troubleshooting, making it feasible to meet project deadlines.

Hardware Requirements

List the essential hardware components required for both the Laser Blaster and Target Module systems.

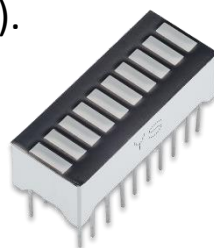
Laser Blaster System

- **Arduino Uno** – Core microcontroller for handling input and output.



- **Microcontroller:** ATmega328P
- **Operating Voltage:** 5V
- **Input Voltage:** 7-12V
- **Digital I/O Pins:** 14 (6 provide PWM output)
- **Analog Input Pins:** 6
- **Flash Memory:** 32 KB
- **Clock Speed:** 16 MHz
- **Usage:** The central processing unit for the project, handling sensor inputs, controlling servos, and managing displays.

- **LED Bar Graph Display** – Array of LEDs to represent ammo count (pins 2-10, A5).



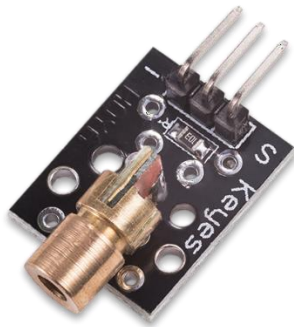
- **Type:** LED bar with 10 segments
- **Voltage:** Typically 2-3V per LED segment
- **Current:** 20mA per LED
- **Usage:** Displays the ammunition or energy levels in the system, where each segment represents a count of available shots.

Push Buttons – Trigger and reload buttons (pins 13, A0).



- **Type:** 3-pin push button
- **Operating Voltage:** Typically 250V
- **Usage:** Serves as a trigger button for firing and a reload button, allowing the user to control the laser and reload the system.

- Laser Module (638 nm) – Used for laser firing.



- **Wavelength:** 650 nm (visible red light)
- **Operating Voltage:** 2-3V
- **Current:** Around 20-40mA
 - **Usage:** Simulates the laser beam in the target system, aiming at LDR sensors to interact with the target.

- Vibration Motor – Simulates recoil on firing (pin A1).



- **Operating Voltage:** 3V-5V
 - **Rated Speed:** 12,000 RPM
 - **Rated Current:** 90-130mA
 - **Usage:** Adds a tactile feedback (vibration effect) each time the system is fired, enhancing the interactive experience.
- Power Supply – Battery or USB power for Arduino and components.

Target Module

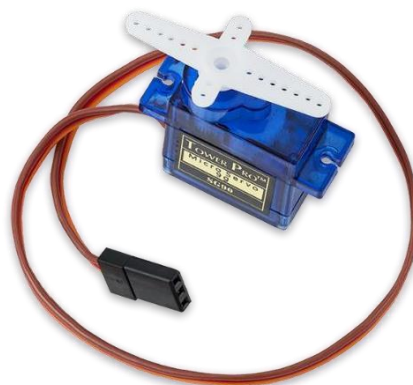
- Arduino Uno – Main controller for detecting laser and managing servo movement.



- Laser Receiver Module Non-Modulator Tub Sensor Output High Level – Detect laser hit (pins A0, A1, A2).



- **Operating Voltage:** 5V (with series resistor)
 - **Resistance:** Varies based on light intensity (typically 1K-10K ohms in bright light, 1M ohm in darkness)
 - **Usage:** Detects laser hits on the target by detecting changes in light intensity, triggering servo movement.
- Servos (x3) – Move in response to laser detection, creating target interaction (pins 9, 10, 11).



- **Operating Voltage:** 4.8-6V
- **Rotation Range:** 180 degrees
- **Speed:** 0.1 sec/60 degrees

- **Torque:** 1.8 kg/cm
- **Usage:** Moves the target out of view when hit, adding dynamic interaction to the system.

- OLED Display – Shows the current score on the target module.



- **Resolution:** 128x64 pixels
- **Interface:** I2C (two-wire interface)
- **Operating Voltage:** 3.3V-5V
- **Usage:** Shows real-time score updates, providing feedback on hits and overall scoring.

- TM1637 Display – Displays elapsed time during target interaction (pins 2, 3).



- **Voltage:** 3.3V-5V
- **Pins:** CLK and DIO for data communication
- **Usage:** Displays the elapsed time in seconds for scoring or tracking purposes.

- **Power Supply** – Power source compatible with Arduino and modules.



- DS-213K-SPST Push Button Momentary Switch – To turn on/restart the target module
- **Rating:** 1.5A 250VAC
- **Contact Resistance:** 100mΩ max
- **Electrical Life:** $\geq 10,000$ Cycles
- **Dielectric strength:** 500VAC 1 minute

Software Requirements

Outline the necessary software and libraries, including configuration and any specific libraries used in the code.

Software Environment and Libraries

1. Arduino IDE

- **Purpose:** Used for writing and debugging code, uploading it to the microcontroller, and monitoring serial output for real-time data analysis or troubleshooting.
- **Features:** Integrated development environment tailored for Arduino boards, offering a simple interface for beginners and powerful tools for advanced users.
- **Configuration:**
 - Install board-specific drivers (e.g., for Arduino Nano, Uno, or similar).
 - Ensure the correct COM port and board type are selected in the "Tools" menu.
 - Include required libraries using the Library Manager or manually add .zip library files.

2. U8g2 Library

- **Purpose:** Facilitates communication with OLED displays (monochrome or color).
- **Key Features:**
 - Supports multiple communication protocols (I2C, SPI).

- Provides efficient graphics rendering for text, shapes, and animations.
- **Configuration:**
 - Install via Arduino Library Manager (Sketch > Include Library > Manage Libraries).
 - Initialize the display with appropriate settings for your hardware, such as I2C address or screen dimensions (e.g., `U8G2_SSD1306_128X64_NONAME_F_HW_I2C`).

3. Servo Library

- **Purpose:** Manages servo motor operations for mechanical movements, such as adjusting the laser position.
- **Key Features:**
 - Simplifies angle-based control for servo motors.
 - Handles PWM signals for precise positioning.
- **Configuration:**
 - Include the library using `#include <Servo.h>`.
 - Define and attach the servo to a specific PWM pin (`myServo.attach(pinNumber)`).
 - Use `myServo.write(angle)` to adjust the servo's angle dynamically.

4. TM1637 Library

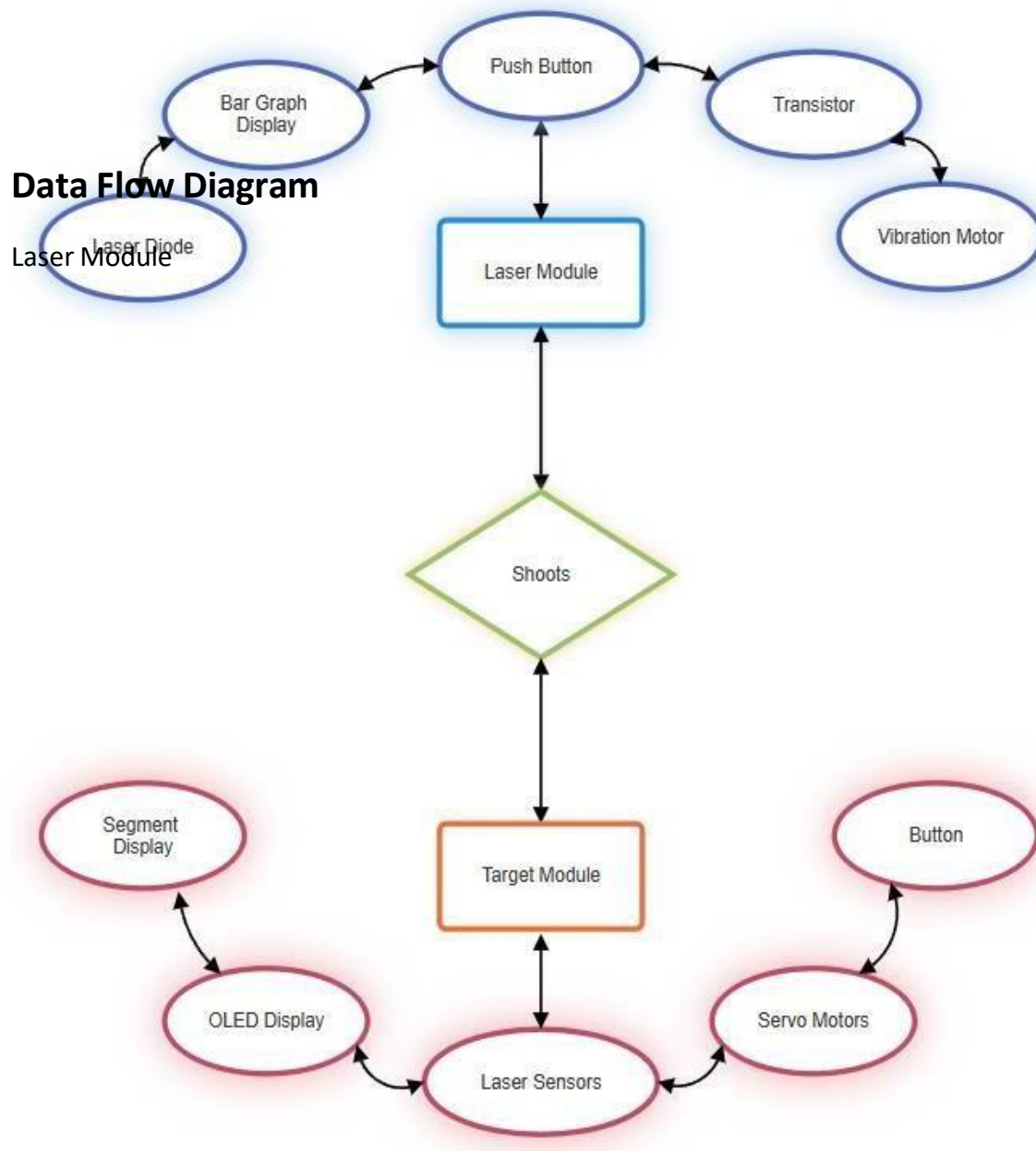
- **Purpose:** Interfaces with 4-digit 7-segment displays to track and display game-related metrics, such as time or score.
- **Key Features:**
 - Provides a simple API for setting digits and displaying numbers.
 - Reduces the complexity of managing multiplexed display data.
- **Configuration:**
 - Install the library via the Library Manager or manually.
 - Connect the TM1637 display to the appropriate pins (CLK and DIO), ensuring proper voltage levels.
 - Initialize with `TM1637Display display(CLK, DIO)` and set brightness or update displayed numbers using provided functions.

5. Custom Functions and Configurations

- **Purpose:** Central to the project's functionality, these functions handle various operations:
 - Ammo management: Tracks available ammunition and triggers reload sequences.
 - Firing mechanism: Detects when the laser is fired and registers hits.
 - Reloading: Simulates reloading actions, such as button presses or servo movements.

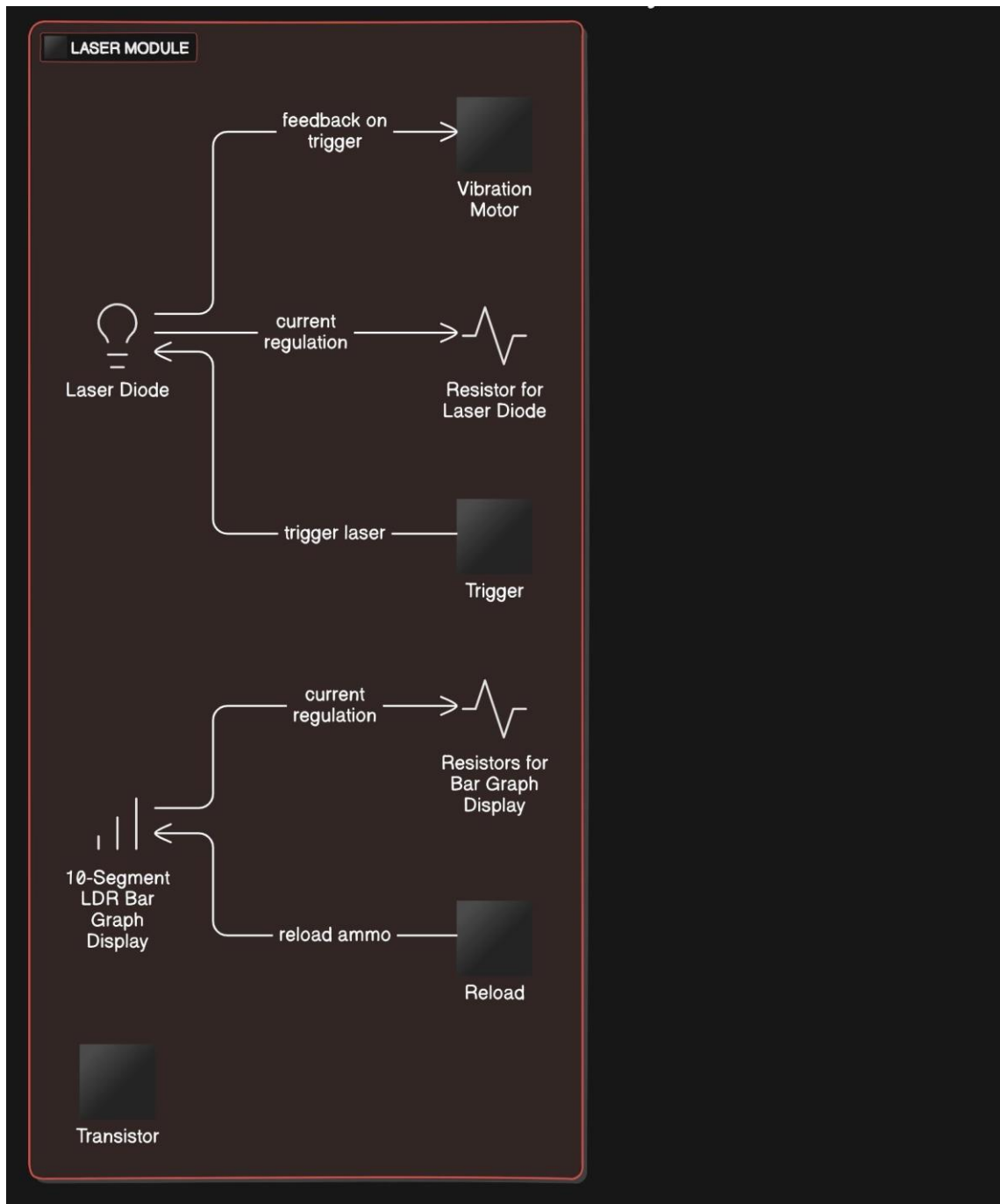
- Laser detection: Uses sensors (e.g., photodiodes) to determine successful laser hits.
- Score updates: Dynamically calculates and displays scores or time left.
- **Configuration:**
 - Functions should be optimized for real-time performance, avoiding delays or blocking code.
 - Use global variables or shared states to synchronize different modules (e.g., display, laser, sensors).
 - Implement interrupt-based triggers where possible for actions like hit detection or button presses.

Entity Relationship Diagram

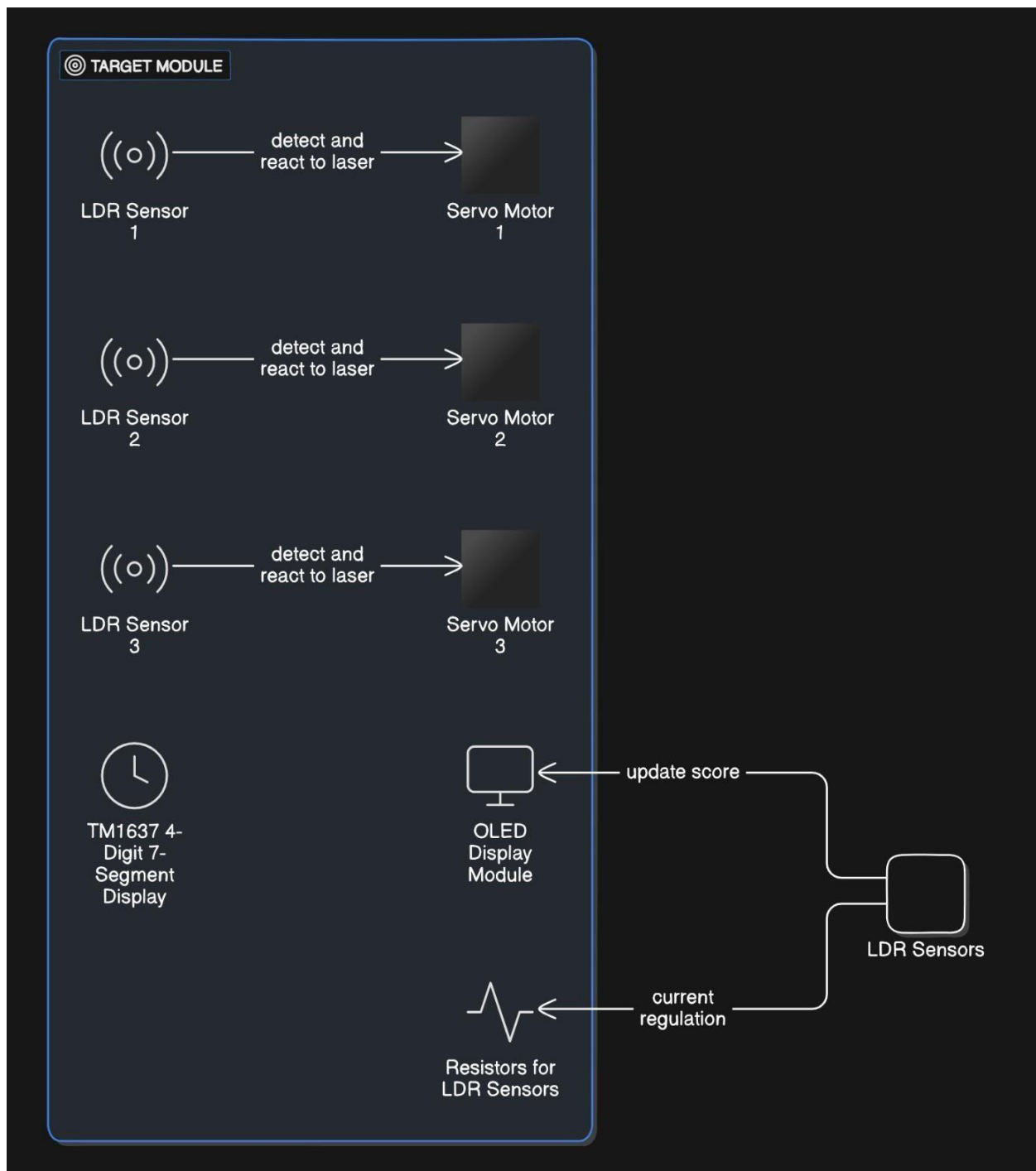


Data Flow Diagram

Laser Module



Target Module



Input and Output screen along with the code.

Laser Blaster

```
// Define pins for the LED bar graph display
int ledPins[] = {2, 3, 4, 5, 6, 7, 8, 9, 10, A5}; // Pin A5
for the last LED
int numLeds = 10;    // Total number of LEDs
int ammoCount = 10;  // Initial ammo count (all LEDs lit)

int ledPin = 11;      // LED pin (supports PWM)
int buttonPin = 13;   // Pin for the trigger button
int reloadButtonPin = A0; // Pin for the reload button
(analog pin A0)
int motorPin = A1;    // Pin for the vibration motor

int dimBrightness = 50; // Dim brightness level (always on
but dim)
int fullBrightness = 255; // Full brightness when button is
pressed

void setup() {
    // Initialize LED bar graph pins as OUTPUT
    for (int i = 0; i < numLeds; i++) {
        pinMode(ledPins[i], OUTPUT);
    }

    // Initialize LED, buttons, and motor pins
    pinMode(ledPin, OUTPUT);
    pinMode(buttonPin, INPUT_PULLUP); // Use internal pull-up
resistor for the trigger button
    pinMode(reloadButtonPin, INPUT_PULLUP); // Use internal
pull-up for reload button
    pinMode(motorPin, OUTPUT);

    // Initially light up all the LEDs (full ammo)
    for (int i = 0; i < ammoCount; i++) {
        digitalWrite(ledPins[i], HIGH);
    }
}
```

```

    }

    // Set the LED to dim brightness initially
    analogWrite(ledPin, dimBrightness);
}

void loop() {
    // Check if the trigger button is pressed
    if (digitalRead(buttonPin) == LOW) { // Button pressed
        (LOW because of pull-up resistor)
        if (ammoCount > 0) {
            fireLaser();
            delay(200); // Small delay to debounce the button
        }
    }

    // Check if the reload button is pressed
    if (digitalRead(reloadButtonPin) == LOW) { // Reload
        button pressed
        reloadAmmo();
    }
}

void fireLaser() {
    // Set LED to full brightness when firing
    analogWrite(ledPin, fullBrightness);

    // Activate the vibration motor for recoil effect
    digitalWrite(motorPin, HIGH);
    delay(100); // Vibrate for 100ms
    digitalWrite(motorPin, LOW);

    // Decrease ammo and turn off one LED
    ammoCount--;
    digitalWrite(ledPins[ammoCount], LOW); // Turn off one
    LED (ammo depletion)

    // Restore LED to dim brightness after firing
    analogWrite(ledPin, dimBrightness);
}

```

```

    // Optional: If ammo is depleted, stop firing
    if (ammoCount == 0) {
        // Indicate no ammo (optional logic can go here for
        reload)
    }
}

void reloadAmmo() {
    // Reload ammo over a period of 3 seconds
    for (int i = ammoCount; i < numLeds; i++) {
        delay(300); // Reload time for each segment (3 seconds
        / 10 LEDs = 300ms per LED)
        digitalWrite(ledPins[i], HIGH); // Light up one LED for
        each reload step
    }
    ammoCount = numLeds; // Reset ammo count to full
}

```

Breaking down of each module functions

Variable Definitions

```

int ledPins[] = {2, 3, 4, 5, 6, 7, 8, 9, 10, A5}; // Pin A5
for the last LED
int numLeds = 10; // Total number of LEDs
int ammoCount = 10; // Initial ammo count (all LEDs lit)

int ledPin = 11; // LED pin (supports PWM)
int buttonPin = 13; // Pin for the trigger button
int reloadButtonPin = A0; // Pin for the reload button
(analog pin A0)
int motorPin = A1; // Pin for the vibration motor

int dimBrightness = 50; // Dim brightness level (always on
but dim)
int fullBrightness = 255; // Full brightness when button is
pressed

```

- **ledPins[]**: This array defines the digital pins (2-10) and analog pin A5 that control the 10-segment LED bar graph display. Each element corresponds to an individual LED segment.
- **ammoCount**: This variable holds the current ammo count. Starting with 10, it decreases as the laser fires.
- **ledPin**: This pin (11) is used to control the external LED that simulates the laser. It supports Pulse Width Modulation (PWM) to adjust brightness.
- **buttonPin**: Pin 13 is used for the push button that simulates the trigger to fire the laser.
- **reloadButtonPin**: Pin A0 is assigned for the reload button.
- **motorPin**: Pin A1 controls the flat 1034 vibration motor, used to simulate the recoil effect when the laser is fired.
- **dimBrightness/fullBrightness**: These variables define two levels of brightness for the LED: dim for idle state and full brightness when the laser is fired.

Setup Function

```
void setup() {
    // Initialize LED bar graph pins as OUTPUT
    for (int i = 0; i < numLeds; i++) {
        pinMode(ledPins[i], OUTPUT);
    }

    // Initialize LED, buttons, and motor pins
    pinMode(ledPin, OUTPUT);
    pinMode(buttonPin, INPUT_PULLUP); // Use internal pull-up
resistor for the trigger button
    pinMode(reloadButtonPin, INPUT_PULLUP); // Use internal
pull-up for reload button
    pinMode(motorPin, OUTPUT);

    // Initially light up all the LEDs (full ammo)
    for (int i = 0; i < ammoCount; i++) {
        digitalWrite(ledPins[i], HIGH);
    }
}
```

```

}

// Set the LED to dim brightness initially
analogWrite(ledPin, dimBrightness);
}

```

- **pinMode():** Configures the pins for the 10-segment LED bar, the laser diode LED, the buttons, and the motor. OUTPUT is used for the LEDs and motor, and INPUT_PULLUP for the buttons (using the internal pull-up resistor to detect button presses).
- **LED Bar Initialization:** All the LEDs in the 10-segment bar graph are lit up initially to represent full ammo.
- **LED Brightness:** The external LED (representing the laser) is set to dim mode at the start using `analogWrite(ledPin, dimBrightness)`.

Main Loop

```

void loop() {
    // Check if the trigger button is pressed
    if (digitalRead(buttonPin) == LOW) { // Button pressed
        (LOW because of pull-up resistor)
        if (ammoCount > 0) {
            fireLaser();
            delay(200); // Small delay to debounce the button
        }
    }

    // Check if the reload button is pressed
    if (digitalRead(reloadButtonPin) == LOW) { // Reload
        button pressed
        reloadAmmo();
    }
}

```

- **Trigger Button:** This checks if the button connected to pin 13 is pressed. If pressed and ammo is available, it calls the `fireLaser()` function to simulate firing the laser.
- **Reload Button:** It checks the state of the reload button connected to pin A0. When pressed, it calls the `reloadAmmo()` function to reload the "ammo" and reset the LED bar graph.

Fire Laser Function

```
void fireLaser() {
  // Set LED to full brightness when firing
  analogWrite(ledPin, fullBrightness);

  // Activate the vibration motor for recoil effect
  digitalWrite(motorPin, HIGH);
  delay(100); // Vibrate for 100ms
  digitalWrite(motorPin, LOW);

  // Decrease ammo and turn off one LED
  ammoCount--;
  digitalWrite(ledPins[ammoCount], LOW); // Turn off one
  LED (ammo depletion)

  // Restore LED to dim brightness after firing
  analogWrite(ledPin, dimBrightness);

  // Optional: If ammo is depleted, stop firing
  if (ammoCount == 0) {
    // Indicate no ammo (optional logic can go here for
    reload)
  }
}
```

- **Laser Simulation:** When firing, the LED (connected to pin 11) simulates the laser firing by increasing brightness to `fullBrightness`. After firing, it returns to `dimBrightness`.

- **Vibration Motor Activation:** The vibration motor (flat 1034) is activated via pin A1 for 100 milliseconds to simulate recoil.
- **Ammo Depletion:** Each time the laser is fired, the `ammoCount` decreases, and one LED from the 10-segment bar is turned off to indicate ammo depletion.

Reload Ammo Function

```
void reloadAmmo() {
    // Reload ammo over a period of 3 seconds
    for (int i = ammoCount; i < numLeds; i++) {
        delay(300); // Reload time for each segment (3 seconds
// 10 LEDs = 300ms per LED)
        digitalWrite(ledPins[i], HIGH); // Light up one LED for
each reload step
    }
    ammoCount = numLeds; // Reset ammo count to full
}
```

- **Reload Simulation:** When the reload button is pressed, it reloads the ammo by gradually turning on the LEDs on the bar graph, with each LED lighting up after a 300ms delay (300ms * 10 LEDs = 3 seconds). The ammo count is restored to 10 after reloading.

Summary of Module Usage:

1. **10-Segment LED Bar Graph Display:** Shows the current ammo count by lighting up segments. Each segment turns off as ammo is depleted and lights up again during reload.
2. **Laser Diode (Replaced with LED):** Simulated by the external LED (connected to pin 11). Its brightness is controlled by PWM (Pulse Width Modulation) to indicate when the laser is fired.
3. **Push Button:** The trigger button (pin 13) is used to fire the laser and the reload button (pin A0) is used to reload the ammo.
4. **Flat 1034 Vibrator Motor:** Activated each time the laser is fired to provide tactile feedback (recoil effect).

5. **Transistor (2N2222):** Used to control the laser diode (now replaced with LED) and vibration motor, managing higher current requirements safely.

This setup allows for an interactive laser gun system with visual feedback on ammo count, vibration for recoil, and control over laser brightness.

Target Module

```
#include <Servo.h>
#include <U8x8lib.h>
#include <TM1637Display.h>

// Create Servo objects for three servos
Servo servo1, servo2, servo3;

// Define pin numbers for LDRs and Servos
int ldrPin1 = A0;    // LDR1 connected to analog pin A0
int ldrPin2 = A1;    // LDR2 connected to analog pin A1
int ldrPin3 = A2;    // LDR3 connected to analog pin A2

int servoPin1 = 9;    // Servo1 connected to digital pin 9
int servoPin2 = 10;   // Servo2 connected to digital pin 10
int servoPin3 = 11;   // Servo3 connected to digital pin 11

int threshold = 20; // Threshold for laser detection
                    (adjustable)
bool laserDetected1 = false;
bool laserDetected2 = false;
bool laserDetected3 = false;

int score = 0; // Initialize score

// Initialize the OLED display
U8X8_SSD1306_128X64_NONAME_HW_I2C u8x8(/* reset=*/
U8X8_PIN_NONE);

// TM1637 settings
#define CLK 2 // Define clock pin for TM1637
#define DIO 3 // Define data pin for TM1637
TM1637Display display(CLK, DIO);

// Time tracking
unsigned long startTime = 0;
int elapsedTime = 0; // Time in seconds
```

```

// Add a variable to track system stabilization
bool systemStabilized = false;

void setup() {
  // Attach servos to their respective pins
  servo1.attach(servoPin1);
  servo2.attach(servoPin2);
  servo3.attach(servoPin3);

  // Initialize OLED display
  u8x8.begin();
  u8x8.setFont(u8x8_font_px437wyse700b_2x2_r);
  u8x8.drawString(0, 0, "Score");
  u8x8.drawString(0, 2, "0");

  Serial.begin(9600); // Start serial communication

  // Initialize TM1637 display
  display.setBrightness(0x0f); // Set brightness to maximum
  display.showNumberDec(0);     // Display initial time (0)

  // Stabilization delay
  Serial.println("Stabilizing LDR readings...");
  delay(2000); // 2-second delay for stabilization

  // Read initial LDR values to avoid incorrect score
  increment
  laserDetected1 = confirmLaserDetection(ldrPin1,
threshold);
  laserDetected2 = confirmLaserDetection(ldrPin2,
threshold);
  laserDetected3 = confirmLaserDetection(ldrPin3,
threshold);

  // Start time tracking after stabilization
  startTime = millis();

  // Mark system as stabilized
  systemStabilized = true;

```

```

}

void loop() {
    // Read and display LDR values in the serial monitor
    int ldrValue1 = analogRead(ldrPin1);
    int ldrValue2 = analogRead(ldrPin2);
    int ldrValue3 = analogRead(ldrPin3);

    Serial.print("LDR1 Value: ");
    Serial.println(ldrValue1);
    Serial.print("LDR2 Value: ");
    Serial.println(ldrValue2);
    Serial.print("LDR3 Value: ");
    Serial.println(ldrValue3);

    // Check and control Servo 1, but only after the system
    has stabilized
    if (systemStabilized && confirmLaserDetection(ldrPin1,
threshold) && !laserDetected1) {
        Serial.println("Laser detected on LDR1!");
        servo1.write(90); // Move Servo 1 to 90 degrees
        delay(500);      // Wait for the servo to reach the
position
        laserDetected1 = true;
        score++;         // Increment score after moving the
servo
        updateScore();   // Update score on OLED
    } else if (ldrValue1 < threshold && laserDetected1) {
        Serial.println("Laser no longer detected on LDR1,
resetting servo.");
        servo1.write(0); // Reset Servo 1 to 0 degrees
        delay(500);      // Wait for the servo to reach the
position
        laserDetected1 = false;
    }

    // Check and control Servo 2
    if (systemStabilized && confirmLaserDetection(ldrPin2,
threshold) && !laserDetected2) {

```

```

    Serial.println("Laser detected on LDR2!");
    servo2.write(90); // Move Servo 2 to 90 degrees
    delay(500);       // Wait for the servo to reach the
position
    laserDetected2 = true;
    score++;          // Increment score after moving the
servo
    updateScore();    // Update score on OLED
} else if (ldrValue2 < threshold && laserDetected2) {
    Serial.println("Laser no longer detected on LDR2,
resetting servo.");
    servo2.write(0);  // Reset Servo 2 to 0 degrees
    delay(500);       // Wait for the servo to reach the
position
    laserDetected2 = false;
}

// Check and control Servo 3
if (systemStabilized && confirmLaserDetection(ldrPin3,
threshold) && !laserDetected3) {
    Serial.println("Laser detected on LDR3!");
    servo3.write(90); // Move Servo 3 to 90 degrees
    delay(500);       // Wait for the servo to reach the
position
    laserDetected3 = true;
    score++;          // Increment score after moving the
servo
    updateScore();    // Update score on OLED
} else if (ldrValue3 < threshold && laserDetected3) {
    Serial.println("Laser no longer detected on LDR3,
resetting servo.");
    servo3.write(0);  // Reset Servo 3 to 0 degrees
    delay(500);       // Wait for the servo to reach the
position
    laserDetected3 = false;
}

// Update elapsed time every second
if (millis() - startTime >= 1000) {

```

```

    elapsedTime++;
    startTime = millis();
    if (elapsedTime <= 60) {
        display.showNumberDec(elapsedTime); // Display time
in seconds (up to 60)
    } else {
        display.showNumberDec(60); // Display 60 when time
exceeds 60 seconds
    }
}

delay(100); // Small delay to avoid rapid sensor reading
}

// Function to confirm laser detection through multiple
readings
int confirmLaserDetection(int ldrPin, int threshold) {
    int stableReading = 0;
    for (int i = 0; i < 5; i++) { // Read the LDR value 5
times
        int ldrValue = analogRead(ldrPin);
        if (ldrValue > threshold) {
            stableReading++;
        }
        delay(10); // Short delay between readings
    }
    return stableReading > 3; // If more than 3 out of 5
readings are above the threshold, consider it stable
}

// Function to update the score on the OLED display
void updateScore() {
    char scoreText[10];
    sprintf(scoreText, "%d", score);
    u8x8.clearLine(2); // Clear the second
line where the score is displayed
    u8x8.drawString(0, 2, scoreText); // Display the updated
score on the second line
}

```

Breaking down of each module functions

```
#include <Servo.h>
#include <U8x8lib.h>
#include <TM1637Display.h>
```

- **Servo.h:** Controls the servo motors, providing commands to move the servos based on laser detection.
- **U8x8lib.h:** Manages the OLED display (SSD1306), displaying the current score.
- **TM1637Display.h:** Controls the TM1637 7-segment display to track and show elapsed time.

```
// Create Servo objects for three servos
Servo servo1, servo2, servo3;
```

- **Servo Objects:** Initializes three servo motors, each attached to a unique target. They rotate when a laser is detected by an LDR sensor.

Score, Time, and Stabilization Tracking

```
int score = 0; // Initialize score
unsigned long startTime = 0;
int elapsedTime = 0; // Time in seconds
bool systemStabilized = false;
```

- **Score:** Tracks successful laser hits on targets.
- **Elapsed Time:** Tracks the time in seconds on the TM1637 display.
- **Stabilization:** Ensures stable readings from LDRs before tracking hits.

Setup Function

```
void setup() {
    servo1.attach(servoPin1); // Attaching servos
    ...
}
```

```

display.showNumberDec(0); // Display initial time
delay(2000); // Stabilization delay for LDRs
startTime = millis(); // Start time tracking
systemStabilized = true; // Mark system as stable
}

```

- **Servo Initialization:** Attaches each servo motor to specific pins.
- **OLED and TM1637 Initialization:** Prepares displays, setting brightness and default display values.
- **Stabilization Delay:** Allows initial LDR values to stabilize, preventing false detections.

Main Loop

```

void loop() {
    int ldrValue1 = analogRead(ldrPin1); // Reading LDR
    values
    ...
    // Checking for laser detection
    if (systemStabilized && confirmLaserDetection(ldrPin1,
threshold) && !laserDetected1) {
        servo1.write(90); // Move servo
        ...
        laserDetected1 = true;
        score++;
        updateScore(); // Update OLED score
    }
    ...
    // Update elapsed time every second
    if (millis() - startTime >= 1000) {
        ...
        display.showNumberDec(elapsedTime); // Update TM1637
    }
}

```


- **LDR Reading:** Continuously reads LDR values, sending laser detection updates to the servos.
- **Servo Control:** Checks if laser detection occurs at each LDR, moves the corresponding servo, and increments the score.
- **Time Tracking:** Updates every second on the TM1637 display.

confirmLaserDetection Function

```
int confirmLaserDetection(int ldrPin, int threshold) {
    int stableReading = 0;
    for (int i = 0; i < 5; i++) {
        int ldrValue = analogRead(ldrPin);
        if (ldrValue > threshold) stableReading++;
        delay(10); // Short delay
    }
    return stableReading > 3; // Stable reading confirmation
}
```

- **Purpose:** Ensures accurate laser detection by reading the LDR multiple times.
- **Process:** Takes five readings, confirms detection if three or more are above the threshold.

updateScore Function

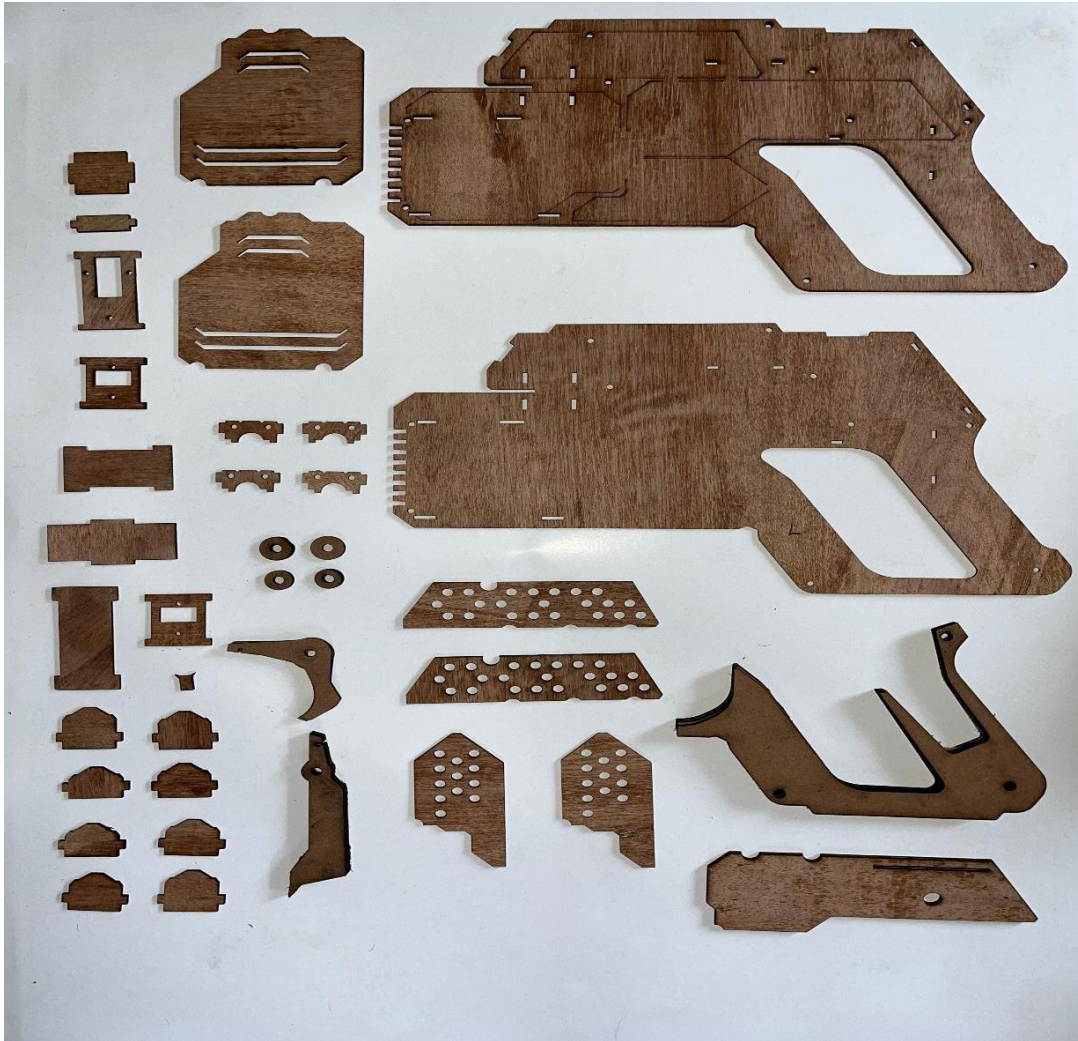
```
void updateScore() {
    char scoreText[10];
    sprintf(scoreText, "%d", score);
    u8x8.clearLine(2); // Clear previous score display
    u8x8.drawString(0, 2, scoreText); // Update OLED with
new score
}
```

- **Purpose:** Updates and displays the current score on the OLED.
- **Process:** Clears the previous score line and prints the updated score each time a target is hit.

This organized setup leverages each module for unique tasks:

- **Servos:** Target movement
- **LDR Sensors:** Laser detection
- **OLED:** Real-time score display
- **TM1637:** Elapsed time tracking

The Physical Components:



1. Laser Gun Module :

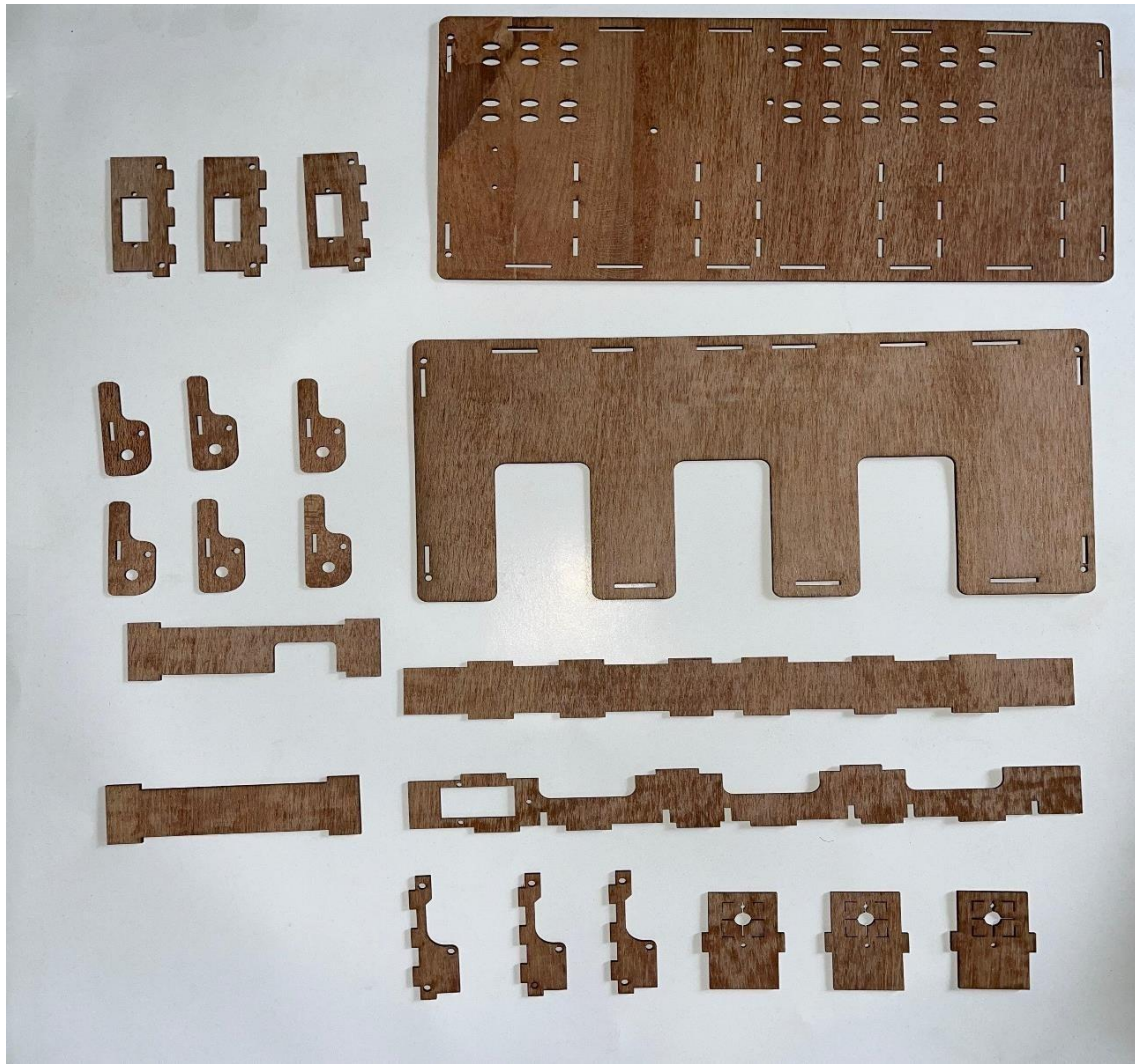
- **Purpose:** Serves as the central component of the system, functioning as the player's interactive device for firing the laser and displaying real-time feedback.
- **Key Features:**
 - **Arduino Uno Integration:**
 - Acts as the primary microcontroller, managing all components, including the laser, display, trigger, and vibration motor.

- Enables precise timing and control for features such as firing, reloading, and recoil simulation.
- **LED Bar Graph Display:**
 - Provides a visual indication of the loading mechanism, showing the player when the laser is ready to fire.
 - Enhances the gameplay experience by simulating weapon charging, adding a strategic element to the game.
- **Trigger Button:**
 - A physical button that initiates the firing sequence when pressed.
 - Provides tactile feedback to simulate real-world weapon interaction, ensuring intuitive user control.
- **Laser Module (638 nm):**
 - Emits a visible red laser beam with high precision, allowing accurate targeting of the opposing module.
 - Chosen wavelength ensures safety and compliance while being bright enough for clear visibility.
- **Push Button for Reloading:**
 - Simulates the reloading process, requiring players to interactively "reset" the weapon after a certain number of shots.
 - Adds realism and an additional challenge to the gameplay.

- **Vibrator Motor for Recoil Simulation:**
 - Activates during firing to mimic the recoil effect of a real gun, enhancing the immersive experience.
 - Compact and efficiently powered by the Arduino, with minimal impact on the overall weight of the gun.
- **Design and Assembly:**
 - The ergonomic layout ensures comfortable handling, with the trigger and reload buttons conveniently placed for easy access.
 - Laser cut panels provide secure slots for all components, maintaining durability and alignment.
 - Internal wiring and mounting points ensure a clean and organized assembly for optimal performance and maintenance.
- **User Interaction:**
 - The combination of the LED bar graph and vibration motor provides real-time feedback, making the gameplay experience engaging and responsive.
 - Reloading and firing mechanics promote strategic gameplay, enhancing the skill-based nature of the project.
- **Material and Portability:**
 - Lightweight wooden construction ensures the laser gun is easy to carry and operate for extended periods.
 - Durable design protects sensitive components like the Arduino and laser module during active use.

- **Educational and Gameplay Value:**

- Offers a practical demonstration of microcontroller-based systems and integration of multiple input/output devices.
- Serves as a hands-on project for learning concepts like pulse-width modulation (PWM) for the motor, LED control, and laser safety.
- The functional and interactive design creates a competitive, fun, and immersive experience for players.



2. Target Module :

- **Purpose:** Functions as the receiver for the laser gun system, designed to detect hits and display scores with enhanced functionality through servo-controlled features and visual displays.
- **Design Enhancements:**
 - **OLED Display:**
 - Provides real-time feedback by displaying the player's score or system status in a compact, high-resolution format.

- The integration ensures easy readability and supports custom animations or text for enhanced user experience.
- **TM1637 Display:**
 - A seven-segment display module to prominently show scores or countdown timers.
 - Offers clear and bright numeric feedback, useful for competitive or multi-player scenarios.
- **Servo Motors:**
 - Three servo motors will likely enable dynamic target movement or visual effects (e.g., flipping panels, moving obstacles, or hit indicators).
 - Enhance gameplay complexity by making the target mobile or adjustable for difficulty settings.
- **Assembly Features:**
 - Pre-drilled mounting points ensure stable attachment of components such as the OLED, TM1637 display, and servo motors.
 - The modular slots can house servo arms and ensure smooth operation without obstructing other mechanisms.
- **Interactive Features:**
 - A hit detection system (using sensors like LDRs or photodiodes) triggers servo-driven actions, such as resetting the target position or activating moving parts.
 - OLED and TM1637 displays provide synchronized visual feedback, keeping players engaged and informed.

- **Educational Integration:**

- Demonstrates multi-component integration, including sensors, displays, and actuators controlled by an Arduino.
- Offers insight into programming logic for scorekeeping, servo control, and display updates.

- **Material and Durability:**

- Wooden construction provides a stable platform for all electronics, ensuring durability during active use.
- Servos are mounted in reinforced sections, protecting against vibrations and wear.

- **Gameplay Versatility:**

- The combined features create an immersive and interactive experience, suitable for competitive or solo gaming.
- Modular design supports upgrades, such as adding additional displays, Wi-Fi/Bluetooth modules for remote score tracking, or even RGB LEDs for visual effects.

Additional Notes (Common to Both Modules):

1. Microcontroller Integration:

- Both modules rely on Arduino microcontrollers for coordinating all components, ensuring seamless communication between inputs (e.g., buttons, sensors) and outputs (e.g., displays, actuators).
- Demonstrates how microcontrollers can act as the brain of interactive systems, showcasing real-world applications of embedded systems.

2. Power Supply and Efficiency:

- The modules are designed to operate efficiently using portable power sources like batteries, ensuring long usage times without frequent recharging.
- Energy-efficient components like LEDs and small servos minimize overall power consumption.

3. Coding and Programming:

- Both modules require customized Arduino code to manage functions such as detecting hits, controlling displays, and managing triggers or servos.
- Offers an excellent opportunity to practice programming in C/C++ and learn about real-time event handling.

4. Modular Design for Easy Maintenance:

- The interlocking wooden panels ensure easy assembly and disassembly for troubleshooting or upgrades.
- Faulty components, such as a damaged servo or display, can be replaced without affecting the rest of the system.

5. Interactive Feedback:

- Both modules feature visual and/or tactile feedback systems, making gameplay intuitive and engaging.
- Feedback mechanisms, like the LED bar graph, OLED/TM1637 displays, and vibration motor, ensure players are constantly informed of their actions and system status.

6. Safety Considerations:

- The laser module (638 nm) is chosen for visibility and safety, complying with low-power laser safety standards to avoid harm.
- Wooden enclosures protect sensitive electronics while keeping sharp edges smooth for safe handling.

7. Expandability and Future Enhancements:

- Both modules are designed to accommodate future upgrades, such as adding wireless communication (e.g., Bluetooth or Wi-Fi) for real-time score tracking or introducing advanced sensors for hit detection.
- Additional game mechanics, like timed challenges or bonus rounds, can be implemented with minimal modifications.

8. Educational Focus:

- The project bridges multiple STEM disciplines, including electronics, programming, mechanical design, and optics.
- Serves as an excellent teaching tool for understanding systems integration and practical problem-solving in real-world projects.

9. Portability and Reusability:

- Lightweight materials and compact designs make both modules portable, ideal for classrooms, exhibitions, or outdoor gaming.
- Components can be reused for other projects, showcasing sustainable and modular design principles.

10. Aesthetic Appeal:

- The natural wooden finish of the modules enhances their visual appeal and adds a professional touch.
- Potential for customization through painting or engraving, making the system visually distinct and personalized.

Limitations of the Project

Accuracy of Laser Detection

The current project relies on Light Dependent Resistors (LDRs) to detect the laser beam on the target module. LDRs, while cost-effective, have relatively slow response times and are susceptible to ambient light interference, especially in brightly lit or outdoor environments. This can lead to inaccurate or delayed laser detection, which affects the project's precision and limits its use to controlled lighting conditions. Furthermore, adjusting sensitivity with LDRs alone can be challenging, as they are limited in dynamic range compared to other light-sensing components like photodiodes.

Limited Ammunition Representation

The LED bar graph display, which visually represents the remaining ammunition, is constrained by the number of available LEDs and Arduino pins. In the current setup, the bar can only show a maximum of 10 rounds, limiting the gameplay experience for users who may want extended or customizable ammo settings. This limitation restricts potential expansions for the system, such as different ammo capacities or power-up mechanisms without significant modification to the hardware or code.

Response Time and Servo Movement

The servo motors used to respond to laser detection have physical limitations in terms of speed and torque. This affects their responsiveness, especially if multiple laser detections occur in quick succession. The result is a delay in moving the servos, which may not correspond accurately with the laser hit feedback. Moreover, continuous servo movement can cause overheating or wear, affecting

long-term performance and reducing the system's reliability over time.

Power Consumption

The combination of components—such as the laser module, vibration motor, LED bar, OLED display, and servo motors—can quickly drain battery power or exceed a single USB power supply's capacity. This presents a challenge in maintaining stable power, especially during extended use, leading to potential performance drops. Users may find it necessary to frequently recharge or replace batteries, limiting the portability and convenience of the system. Optimizing power usage or adding renewable energy sources could address this issue but would require substantial modifications.

Score and Time Limit Constraints

The current display setup on the target module restricts score representation to single digits and time tracking to 60 seconds. This is due to the limited resolution and capabilities of the TM1637 display module, which can only show numbers in a 4-digit format. For longer or more complex game sessions, this constraint limits the potential for tracking detailed scores or extended timing, reducing the flexibility of gameplay options. Overcoming this limitation would require upgrading to a more advanced display, which might demand additional coding and wiring complexity.

Future Scope of the Project

Improved Laser Detection with Photodiodes or IR Sensors

The project could significantly benefit from using photodiodes or infrared sensors, which offer quicker response times and greater sensitivity than LDRs. Photodiodes, specifically designed for light detection, would reduce false positives and ensure accurate detection of the laser beam in a wider range of lighting conditions. This enhancement would make the target module more adaptable for indoor and outdoor use, allowing the system to function reliably in diverse environments and expanding its applicability beyond controlled spaces.

Wireless Communication and Multi-Target Setup

Adding wireless communication capabilities, such as Bluetooth or Wi-Fi, would open up possibilities for multiplayer interaction and remote score tracking. With a wireless system, multiple target modules could communicate with a central device, allowing for coordinated gameplay, competitive scoring, or team modes. This modification would not only add a collaborative element to the project but could also enable remote monitoring of scores and time limits, enriching the user experience and making the system adaptable for various group activities.

Automatic Ammo Reload Mechanism

Implementing an automatic reload system could increase gameplay realism and convenience. Instead of manually pressing a reload button, users could have a timed or sensor-based reload feature that automatically recharges ammo after a cooldown period. Additionally, adding audio or visual cues to indicate reload status (e.g., a buzzer or flashing LED) would enhance user engagement. This upgrade would streamline gameplay, making the experience more immersive and reducing the need for user intervention during action sequences.

Expanded Display Capabilities

Upgrading to a larger or multi-line display for the target module would allow the system to present more detailed information, such as score, ammo count, time elapsed, and even messages or animations. Using a larger OLED or an LCD display would increase the interface's versatility, providing players with richer feedback. It would also open opportunities for adding extra game modes, such as timed challenges or varying difficulty levels, making the project more versatile and user-friendly.

Battery Optimization and Renewable Power Sources

Integrating energy-efficient components and implementing renewable power options, such as solar charging, could extend the system's operating time. Using low-power components where possible, optimizing sleep modes for the Arduino, and switching to rechargeable lithium-ion batteries or solar panels would improve portability and reduce the need for frequent battery replacements.

This would enhance the project's sustainability and make it more convenient for outdoor or prolonged usage.

Game Modes and Scoreboard Integration

By adding different game modes, such as timed challenges or difficulty levels, users would have access to a variety of gameplay experiences. A scoreboard could be added to track scores over multiple rounds or display high scores, increasing the project's competitiveness and appeal. The addition of these features would make the system adaptable for different user skill levels and gameplay preferences, creating a more engaging experience for individual and group users alike.

Bibliography

These is the comprehensive list of all sources, articles, datasheets, and code references consulted during the project.

Arduino Documentation and Guides

Arduino. Getting Started with Arduino Uno. Available at <https://www.arduino.cc/en/Guide/ArduinoUno>. This document was referenced for understanding Arduino Uno's pin configurations, setup, and functionality as the primary microcontroller.

Servo Motor Control with Arduino

Adafruit. Using Servo Motors with Arduino. Retrieved from <https://learn.adafruit.com/using-servos-with-arduino>. This source provided insights on integrating servos with Arduino for interactive elements like target movement.

OLED Display Library Documentation (U8g2)

Arduino Libraries. U8g2 Library Documentation. Retrieved from <https://github.com/olikraus/u8g2>. Used as a reference for configuring and managing the OLED display for score tracking and visual feedback in the target module.

TM1637 4-Digit Display Interface

TM1637 datasheets and Arduino examples from sources such as GitHub repositories and Instructables projects were consulted to interface with and control the 4-digit display for time tracking on the target module. These examples provided code and wiring configurations necessary for integrating the TM1637 display.

Technical Datasheets

Component datasheets for LEDs, LDRs, laser modules, and other electrical components were consulted to understand electrical characteristics, such as voltage, current requirements, and operational limitations. These datasheets helped inform the design choices to ensure compatibility and optimal performance.

Additional Code and Project References

Open-source code and community projects from Arduino forums and electronics websites (e.g., Instructables) provided insights on best practices for laser and servo integration, as well as strategies for game logic and scoring systems. These resources were invaluable for troubleshooting and refining the project code and design.

Laser Module Integration

- **Laser Safety Guidelines:** Laser Institute of America. *Guide to Laser Safety Standards*. Available at <https://www.lia.org>. Referenced to ensure safe operation of laser modules within the project.
- **Laser Modules in Arduino Projects:** Open-source guides from DIY Electronics blogs (e.g., Circuit Digest) provided insights into safe wiring, calibration, and usage of low-power laser modules for interactive applications.

Light-Dependent Resistors (LDRs)

- **LDR Theory and Applications:** Electronics Tutorials. *Introduction to Light-Dependent Resistors*. Available at <https://www.electronics-tutorials.ws>. Consulted to understand the behavior, limitations, and practical considerations of LDRs in detecting laser beams.
- **Practical LDR Integration:** Hackster.io project references for LDRs and Arduino provided examples of ambient light calibration and noise reduction techniques.

LED Bar Graph and Visual Indicators

- **LED Control Libraries:** Arduino Libraries. *Adafruit_LED Backpack Library Documentation*. Retrieved from <https://github.com/adafruit>. Used for controlling LED displays to represent ammunition visually.
- **Circuit Design for LED Bars:** Tutorials from Maker Pro. *Using LED Bar Graphs in DIY Electronics Projects*. Available at <https://www.maker.pro>. Helped optimize LED wiring and power consumption.

Power Optimization Strategies

- **Arduino Power Management:** SparkFun. *Reducing Power Consumption with Arduino Projects*. Available at <https://www.sparkfun.com>. Provided techniques for optimizing Arduino power usage through sleep modes and efficient coding.
- **Battery Technology:** Technical datasheets for lithium-ion batteries from manufacturers such as Panasonic were reviewed to understand energy density, recharge cycles, and compatibility with the project.
- **Solar Integration:** Solarbotics. *Solar Power for Arduino Projects*. Retrieved from <https://solarbotics.com>. Consulted to explore renewable energy options for extending the system's runtime.

Servo Motor Mechanics

- **Advanced Servo Techniques:** Pololu. *Servo Controller Guides*. Available at <https://www.pololu.com>. Used for implementing precise and reliable servo motor control in response to laser detection.
- **Wear and Heat Management:** Practical recommendations from Mechatronics Engineering blogs on preventing servo overheating during continuous operations.

Wireless Communication and Multiplayer Enhancements

- **Bluetooth Modules:** HC-05/HC-06 Datasheets. Technical documentation on integrating Bluetooth modules with Arduino for wireless communication.
- **Wi-Fi Connectivity:** Espressif Systems. *ESP8266 Technical Reference*. Available at <https://www.espressif.com>. Used as a guide for incorporating Wi-Fi for multi-target coordination and remote data tracking.

Advanced Displays for Enhanced User Feedback

- **LCD Display Integration:** Arduino. *LiquidCrystal Library Documentation*. Available at <https://www.arduino.cc/en/Reference/LiquidCrystal>. Used for exploring upgrades from TM1637 to multi-line LCD displays.
- **OLED Enhancements:** Practical implementation examples from instructive YouTube channels (e.g., The Arduino Guy) for creating animated displays using U8g2 library.

Game Logic and Software Design

- **Interactive Game Coding:** Tinkercad. *Arduino Game Design Basics*. Online interactive tutorials helped structure the game logic for scoring, timing, and ammo systems.
- **Open-Source Game Projects:** Examples from GitHub, such as *LaserTag-Projects*, inspired ideas for incorporating game modes and competitive elements.

Community Contributions and Troubleshooting

- **Arduino Forums:** Arduino Community Discussions. Available at <https://forum.arduino.cc>. Provided solutions to specific coding and hardware challenges encountered during the project.

- **Stack Overflow:** Insights from active discussions on common issues with LDR calibration, servo response time, and power management in Arduino project