ICT 305 2.0 Embedded Systems

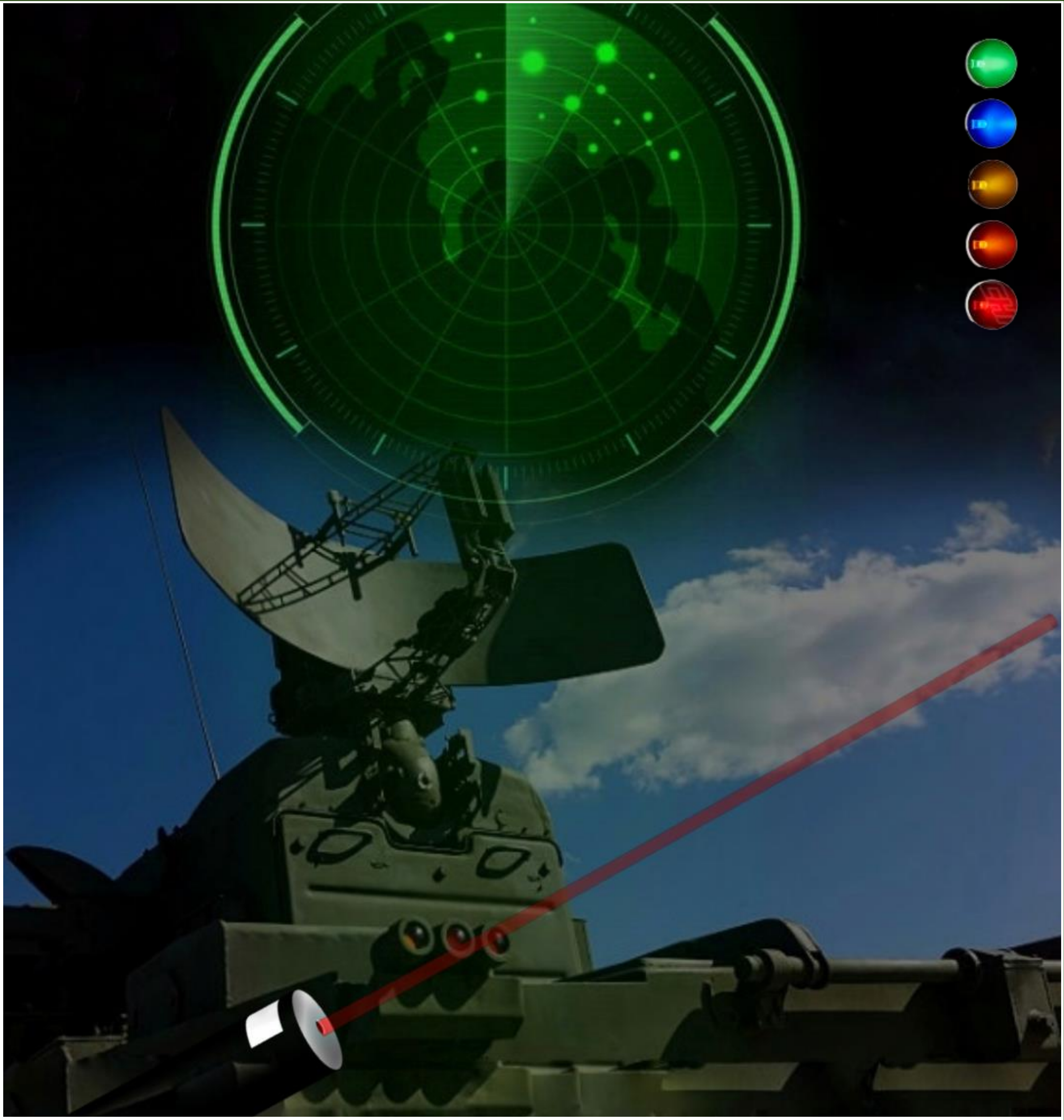
# Arduino Project

Year 03, Semester 01

2023

K.D.Y.Niwarthana

**AS2020979**



# DEFENSE SYSTEM

## Contents

## Introduction

Today, safety matters a lot. This project uses Arduino, a super-smart but easy-to-use tech tool, to make a smart security system. It's great at three things: recognizing stuff in a certain area, showing it on a screen, and quickly reacting if it spots something risky. By being smart with sensors and controls, it watches over spaces and acts fast when needed. This project is changing how we stay safe, all thanks to Arduino's cleverness!

This system does a couple of big jobs. First, it looks around and displays what it sees in real-time on a screen. That means it can keep an eye out and tell us if something seems off or unsafe. But the cool part is its ability to do something about it. It's set up to automatically respond to potential threats. So, if it spots something worrisome, it reacts swiftly and precisely, almost like having a vigilant, smart guard in place. There's another feature to it as well. This system doesn't just notice the threats; it also knows how dangerous they might be based on how far away they are. And here's the clever bit: it uses different colors of LEDs to signal the danger level. This way, the operators and anyone using the system can instantly grasp the level of risk, thanks to the specific colors lit up.

This project brings together security and technology, using Arduino's smarts to make places safer. The aim is to learn new ways to solve problems and improve safety by spotting, displaying, and acting upon possible threats. The report details how this defense system works and its role in boosting security through smart tech.

## Features

- **Object Detection**: Identifies nearby objects, offering a complete environmental view.
- **Radar Display**: Shows real-time object positions on a radar-style screen.
- **Position and Angle Calculation**: Accurately calculate objects' exact positions and angles instantly.
- **Danger Assessment and Warning Lights**: Assesses risk levels and activates alarming lights for safety.
- **Shooting Angle Calculation**: Determines the best angle for precise shooting based on input data.

Targeting and Shooting with Laser Technology: Precisely targets and engages objects using laser technology.

## Component Description

### Arduino Uno Board

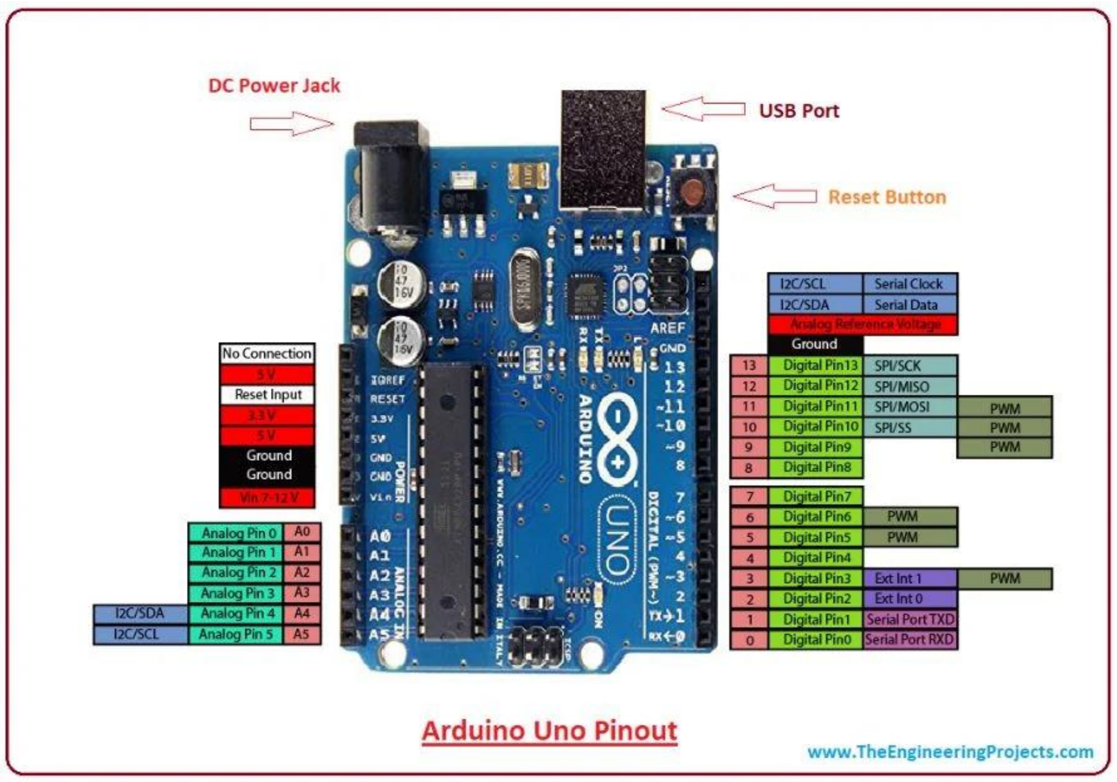


**Figure 01 - Figure of Arduino Uno Board**

The Arduino Uno is like the heart of the Arduino family. It's perfect for both newbies and pros. This board is built around the ATmega328P microcontroller and has pins that connect to different parts like sensors and motors.

Power Options:

USB: Powered through a computer's USB connection.

DC power Jack: Accepts power from an AC mains supply.

Crystal Oscillator:

Provides a 16 MHz frequency, aiding time-related functions.

Reset:

Allows resetting the board using a button or external pin connection.

Pins:

3.3V & 5V: Supplies power at different voltages.

GND (Ground): Multiple pins to ground circuits.

Vin: Additional power input from an external source.

Analog Pins:

Six input pins (A0-A5) convert analog sensor data to digital values.

Main Microcontroller:

Varied ICs from ATMEL, acting as the board's brain.

ICSP Pin:

Used for programming, featuring MOSI, MISO, SCK, RESET, VCC, and GND.

LED Indicators:

Power LED: Confirms correct board power.

Digital I/O:

14 pins (6 PWM) configurable for input or output.

AREF:

Sets an external reference voltage for analog input pins.

## Breadboard

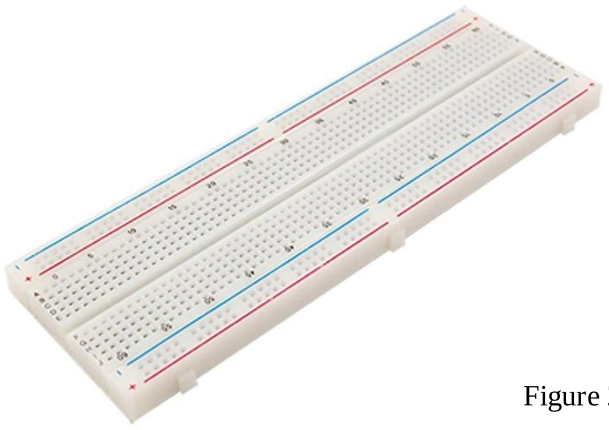


Figure 2 - Breadboard

Much like a connected electrical platform, a breadboard organizes components without soldering. It's highly convenient for circuit testing—parts easily fit into slots, simplifying setup and changes. Power rows interconnect everything and are perfect for experimenting before finalizing designs. It's a top choice for trying out electronic setups without the permanence of soldering.

## Ultrasonic Sensor

The ultrasonic sensor is a device that uses high-frequency sound waves to measure distances. It measures how long it takes for the waves to return after hitting an item.

The ultrasonic sensor typically runs on a 5V power supply when connected to an Arduino Uno, which is compatible with the Uno's power output. It requires no additional components and may connect straight to the Uno thanks to its low power consumption.

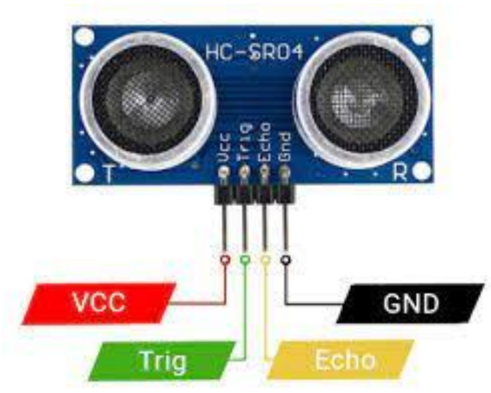


Figure 3 – Ultrasonic Sensor

Digital pins are used for connection; an Echo pin is used to record the reflected signal, and a Trigger pin is used to emit the ultrasonic signal.

This can be programmed to trigger the sensor, record the travel duration of the signal, and compute the distance based on sound speed by utilizing the Arduino IDE and certain libraries.

## Servo Motors

Servo motors designed for a 180-degree range offer precise control within this specific angular span, allowing controlled movement from 0 to 180 degrees. These motors excel in accurate angular positioning, making them ideal for tasks requiring fine control over rotational angles.

Figure 4 - Servo Motor

Usually operating on a 5V power supply, these servo motors consume low power, enabling direct connection without complex power management systems. Controlled by Pulse Width Modulation (PWM) signals, these motors achieve the desired angle within the 0 to 180-degree range by adjusting the signal's pulse width.

## Laser Pointer

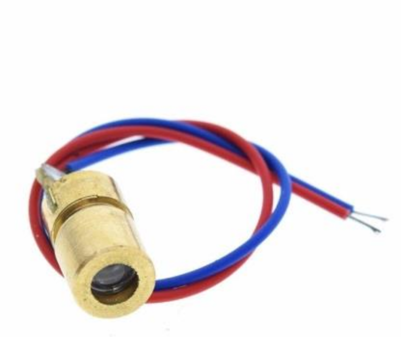


Figure 5 -Laser pointer

The Laser Pointer Red Dot for Arduino Uno R3 Mega Nano, designed for use with Arduino Uno R3, Mega, or Nano, emits a visible red dot at a wavelength of 650nm. This small pointer has an outer diameter of 6mm and operates on a standard 5V power supply.

It's important to note that the laser's operating current is less than 20mA, making it energy-efficient and compatible with common power sources. When in use, this laser projects a single dot, simplifying its application in various Arduino projects for precision pointing or alignment purposes.

## LEDs

Figure 6 – LED s

LEDs, or Light Emitting Diodes, are small, powerful lights used in various electronics. They come in different colors and sizes, and their key job is to produce light when electricity passes through them. They're energy-efficient, durable, and a common choice for indicators or displays in many devices, including Arduino projects. They're simple yet highly effective in providing visual cues or lighting in various applications.

## Resisters

A resistor is a small electrical part used in circuits. It's like a traffic controller for electricity. It helps control how much electricity flows in a circuit by resisting its flow.

Resistors are used in various ways. They can make sure too much electricity doesn't pass through, adjust how strong signals are, or divide how much power goes to different parts of a circuit. They're handy in many electronic devices to make them work just right.

Figure 7 -Resistors

## Libraries

**Servo Library in Arduino: <Servo.h>**

Enables accurate control of servo motors.

Sets specific angles for servo movement.Allows smooth and controlled motor operation.

**Math Library in Arduino: <math.h>**

Offers various mathematical tools.

Includes addition, subtraction, multiplication, and division.

Provides sine, cosine, and tangent functions.

Supports logarithmic calculations.

Assists in multiple mathematical operations in sketches.

**Processing Software**

Processing is a unique software sketchbook that also serves as a language for discovering the world of coding. It's been around since 2001, focusing on making technology understandable in the art world and making the visual world understandable in tech. Tons of students, artists, designers, and tech hobbyists use Processing to learn and try out new ideas. It's open source and works on macOS, Windows, and Linux. Whatever you create on Processing can easily be used across different platforms like macOS, Windows, Android, Raspberry Pi, and many other Linux platforms.

## Method

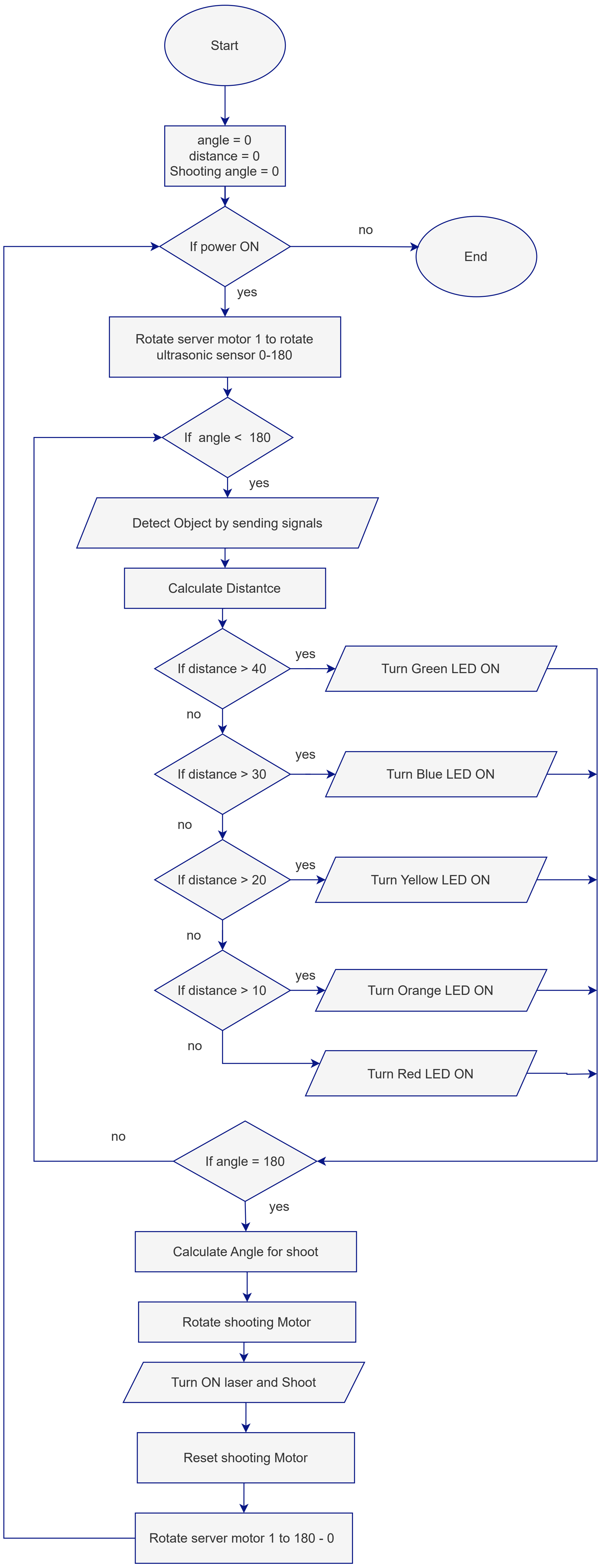
I. Configure servo motors to rotate the ultrasonic sensor within a 1–180-degree range.
II. Detect and display objects on a radar display created using Processing software.
III. Calculate distances between the sensor and detected objects.
IV. Illuminate five LEDs to indicate different proximity levels based on object distance.
V. Initiate a simulated shooting response within a specific range.
VI. Calculate the angle required for 'shooting'. Rotate the second servo meter and demonstrate shooting using a laser pointer.
VII. Reset the system after shooting and resume scanning the area.

## Design Phase

### Project Overview

This defense setup uses moving sensors to find stuff close by, and a computer program shows what it sees. It measures how far away the things are. Five lights show different colors:

Green means it's safe (40 cm or more).
Blue means something's coming (30-40 cm).
Yellow means it's close (20-30 cm).
Orange means it's a bit dangerous (10-20 cm).
Red means it's super close and risky (less than 10 cm).
If something's between 1-40 cm distance, it pretends to shoot it with a laser. Then, it goes back to looking around and repeats the whole thing again.


Start
angle = 0
distance = 0
Shooting angle = 0
If power ON
no
End
yes
Rotate server motor 1 to rotate ultrasonic sensor 0-180
If angle < 180
yes
Detect Object by sending signals
Calculate Distantce
If distance > 40
yes
Turn Green LED ON
no
If distance > 30
yes
Turn Blue LED ON
no
If distance > 20
yes
Turn Yellow LED ON
no
If distance > 10
yes
Turn Orange LED ON
no
Turn Red LED ON
no
If angle = 180
yes
Calculate Angle for shoot
Rotate shooting Motor
Turn ON laser and Shoot
Reset shooting Motor
Rotate server motor 1 to 180 - 0

## Block Diagram

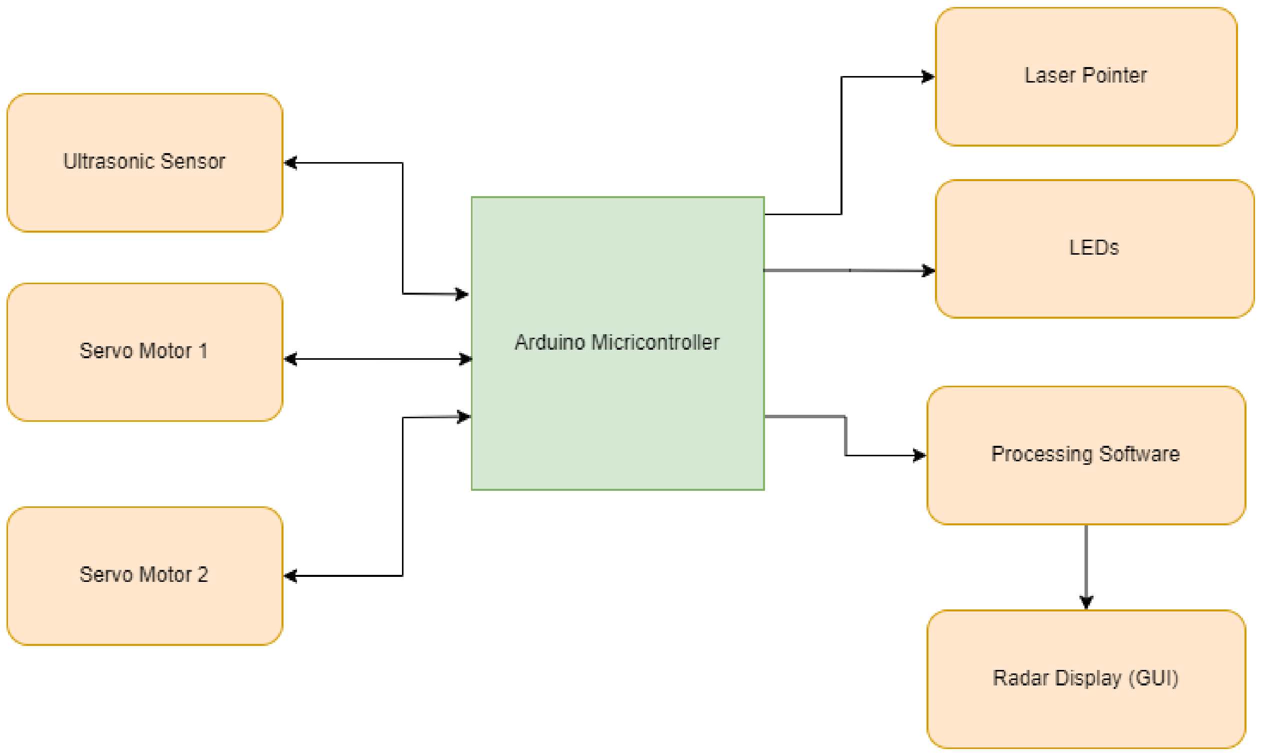


Figure 8 – Block Diagram

# Calculations Used

## Calculations for LED resisters

V = (power source) - (Voltage Drop)

= 5V - 1.8V

= 3.2 V

I = 25 mA

So,

R = V/I

= 3.2 V / 25 mA

R = 3.2/0.025

= 128 Ohms

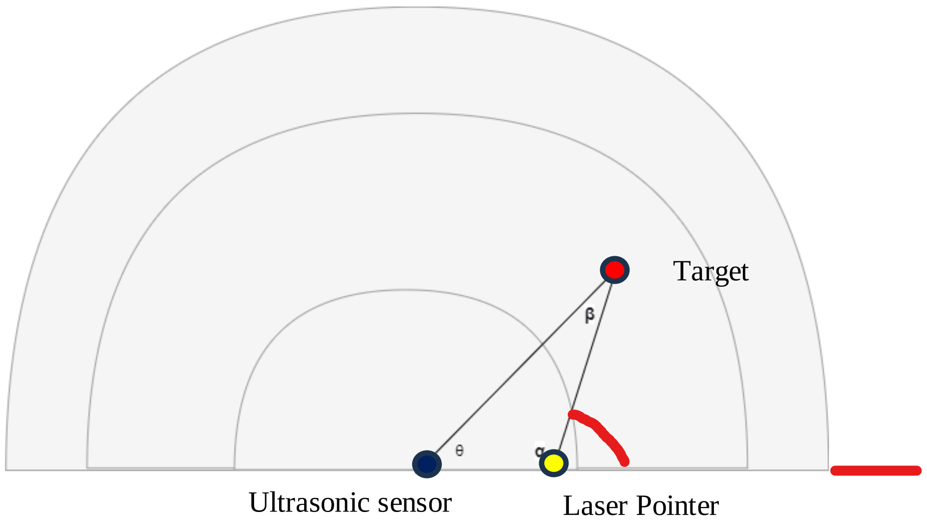
The closest match is 220 Ohms. But for more safety, I used 330 Ohms

## Calculations for Distance

Duration = Time to signal to go and return

Distance = (Duration x SpeedOfSound) / 2

## Calculations for Target Angle



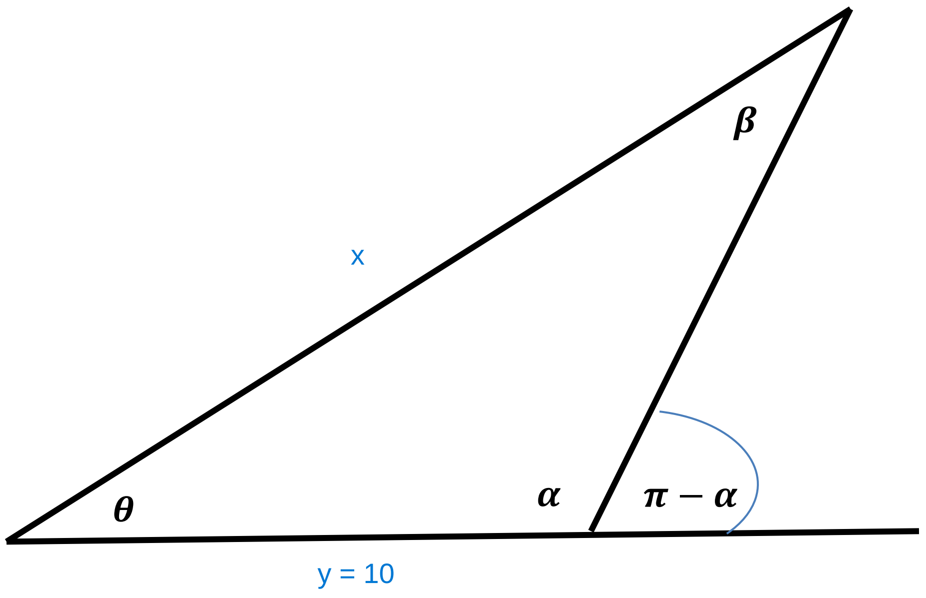


Figure 9 – Sin Triangle

Used  Law of Sines in trigonometry to calculate the angle

$$\frac{y}{\sin\beta} = \frac{x}{\sin\alpha}$$

$$\pi - \alpha = \theta + \beta$$

$$\pi - \alpha - \theta = \beta$$

$$\pi - (\alpha + \theta) = \beta$$

$$\frac{y}{\sin(\pi - (\alpha + \theta))} = \frac{x}{\sin\alpha}$$

$$\frac{y}{\sin(\alpha + \theta)} = \frac{x}{\sin\alpha}$$

$$\frac{y}{\sin\alpha\cos\theta + \cos\alpha\sin\theta} = \frac{x}{\sin\alpha}$$

$$\frac{y\sin\alpha}{\sin\alpha\cos\theta + \cos\alpha\sin\theta} = x$$

$$\frac{y}{\cos\theta + \cot\alpha\sin\theta} = x$$

$$\frac{y}{x} = \cos\theta + \cot\alpha\sin\theta$$

$$\frac{\frac{y}{x} - \cos\theta}{\sin\theta} = +\cot\alpha$$

$$\frac{y - \text{x} \cos\theta}{x \sin\theta} = \cot\alpha$$

$$\frac{x \sin\theta}{y - \text{x} \cos\theta} = \tan\alpha$$

$$\alpha = \tan^{-1}\left(\frac{x \sin\theta}{y - \text{x} \cos\theta}\right)$$

## Codes

### Arduino Code

```
/*
Project    : Arduino Project - Defense System
Student ID : AS2020979
Name       : K.D.Y.Niwarthana
*/


//LIBARIES

#include <Servo.h>
#include <math.h>

//Pin declarations

int trigPin = 8;
int echoPin = 9;
int laserPin = 10;
int redLED = 3;
int blueLED = 4;
int greenLED = 5;
int yellowLED = 6;
int orangeLED = 7;
```

```
//Varible declaration
long duration;
int distance;
Servo myServo; //for rotate ultrasonic sensor
Servo myServo2; //for rotate laser
int targetAngle = 0;
double targetAngleDegrees;

int angle1to40 = 0;  // Angle when distance is between 1-40
int loopCount = 0;  // Counter for the number of loop iterations


//Setup
void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(laserPin, OUTPUT);
  pinMode(redLED, OUTPUT);
  pinMode(blueLED, OUTPUT);
  pinMode(greenLED, OUTPUT);
  pinMode(yellowLED, OUTPUT);
  pinMode(orangeLED, OUTPUT);

  digitalWrite(laserPin, LOW);
  Serial.begin(9600);
  myServo.attach(2);
  myServo2.attach(12);
  myServo2.write(0);
  myServo.write(0);
}
```

```
//Function for calculate distance and get target angle
void calculateDistance() {

  //distance calculation
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = duration * 0.034 / 2;

  //calculate target angle for laser using sin Rule
  if (distance >= 1 && distance <= 40) {
    angle1to40 = myServo.read(); //get the current angle
    double targetAngle = atan((distance*sin(angle1to40)) / (10 - (distance *
cos(angle1to40))));
    // Convert radians to degrees
    targetAngleDegrees = targetAngle * 180.0 / M_PI;
  }

}

//Function for laser pointer shooting dimostration
void Shooting(){
    loopCount++;
    // Update myServo2 when one full rotation of myServo is complete only
    if (loopCount == 1) {
      myServo2.write(180 - targetAngleDegrees); //Target angle is the angle from
180. To ge the rotate angle minus it from 180
      digitalWrite(laserPin, HIGH); //on laser
      loopCount = 0;
      delay(4000);
      digitalWrite(laserPin, LOW);
      myServo2.write(0); // Reset the position
      loopCount = 0; //reset loop count
    }
  }
```

```
//FDunction to control LEDs
void controlLEDs() {
  if (distance > 40) {
    digitalWrite(greenLED, HIGH);
    digitalWrite(blueLED, LOW);
    digitalWrite(yellowLED, LOW);
    digitalWrite(orangeLED, LOW);
    digitalWrite(redLED, LOW);
  } else if (distance >= 30 && distance <= 40) {
    digitalWrite(greenLED, LOW);
    digitalWrite(blueLED, HIGH);
    digitalWrite(yellowLED, LOW);
    digitalWrite(orangeLED, LOW);
    digitalWrite(redLED, LOW);
  } else if (distance >= 20 && distance < 30) {
    digitalWrite(greenLED, LOW);
    digitalWrite(blueLED, LOW);
    digitalWrite(yellowLED, HIGH);
    digitalWrite(orangeLED, LOW);
    digitalWrite(redLED, LOW);
  } else if (distance >= 10 && distance < 20) {
    digitalWrite(greenLED, LOW);
    digitalWrite(blueLED, LOW);
    digitalWrite(yellowLED, LOW);
    digitalWrite(orangeLED, HIGH);
    digitalWrite(redLED, LOW);
  } else if (distance >= 0 && distance < 10) {
    digitalWrite(greenLED, LOW);
    digitalWrite(blueLED, LOW);
    digitalWrite(yellowLED, LOW);
    digitalWrite(orangeLED, LOW);
    digitalWrite(redLED, HIGH);
  }
}
```

```
void loop() {
  // Increment the loop count for each iteration to degree by degree unlit 180
  for (int i = 0; i <= 180; i++) {
    myServo.write(i);
    delay(30);
    calculateDistance(); //calculate distanse
    controlLEDs(); //call led function

    //print distance and angle to use in processor code. Used "," and "." to
seperate each record
    Serial.print(i);
    Serial.print(",");
    Serial.print(distance);
    Serial.print(".");

  }

  delay(3000);
  Shooting();

  //go back to 0 from 180

  for (int i = 180; i > 0; i--) {
    myServo.write(i);
    delay(180);
    calculateDistance();
    controlLEDs();
    Serial.print(i);
    Serial.print(",");
    Serial.print(distance);
    Serial.print(".");
  }

}
```

## Processing Code

```
//import libiaries
import processing.serial.*; //for serial comunication
import java.awt.event.KeyEvent;
import java.io.IOException;


Serial myPort; //declare variable to manage serial communication with arduino board

//variables for distance, angle and other data
String angle="";
String distance="";
String data="";

//for error handling
String noObject;

//variables for processed data
float pixsDistance;
int iAngle, iDistance;
int index1=0;
int index2=0;

void setup() {

 size (1280,720); //set display screen size to device resolution
 smooth();
 myPort = new Serial(this,"COM7", 9600); //get COM6 port for serial comunication at boud rate
og 9600
 myPort.bufferUntil('.'); //get and store data until period occurs and call SerialEvent function to
proccess buffer data

}

//when new data is available, this function will automatically caled
void serialEvent (Serial myPort) {

  data = myPort.readStringUntil('.'); //read data until period occurs
  data = data.substring(0,data.length()-1); //remove last character to ensure to remove period from
proccessing
```

```
  //we get data like aqngle , distance. "," seperate both parties

  index1 = data.indexOf(","); //assign "," as index 1
  angle= data.substring(0, index1); //get angle from the data before ","
  distance= data.substring(index1+1, data.length()); //get distance like before by accessing the
data after "," until the end of data


  iAngle = int(angle); //save angle as the numarical value from read data
  iDistance = int(distance); //save distance as the numarical value from read data
}

//main logic for sketch

void draw() {

  fill(98,245,31); //set default drawing color as green

  noStroke(); //remove outlines
  fill(0,4); //background color
  rect(0, 0, width, height-height*0.065); //to sketch fading lines set color

  //calling functions
  drawRadar();
  drawLine();
  drawObject();
  drawText();
}


void drawRadar() {
  pushMatrix();                          //save the current transformation matrix
  translate(width/2,height-height*0.074);   //centering the origin to the center of the canvas
  noFill();                          //thsi will only have outline, but not filled lines
  strokeWeight(2);                        //set outline weight
  stroke(98,245,31);                      //set trocke color
  // draws the arc lines. (center of arc by first 0,0 , next set width and height, start and end angles
of arc
  arc(0,0,(width-width*0.0625),(width-width*0.0625),PI,TWO_PI);
  arc(0,0,(width-width*0.27),(width-width*0.27),PI,TWO_PI);
  arc(0,0,(width-width*0.479),(width-width*0.479),PI,TWO_PI);
  arc(0,0,(width-width*0.687),(width-width*0.687),PI,TWO_PI);

  // Draw the angle lines at 10-degree intervals
  //stating point by 0,0  , use half od the canvus width as radius. minus because it starts from the
bottom , then cos() point out the horizontal end point   and the get vertical end point by sin()
  for (int i = 0; i <= 180; i += 10) {
    float angleRad = radians(i);
    line(0, 0, (-width/2) * cos(angleRad), (-width/2) * sin(angleRad));
  }

  popMatrix();
}


//draw target object od ratadar graph
void drawObject() {
  pushMatrix();//save current transformation
  translate(width/2,height-height*0.074); //posistion the object center at the horizontal center.
this is common practice.
  strokeWeight(9);
  stroke(255,10,10); // red color
  pixsDistance = iDistance*((height-height*0.1666)*0.025); //calculate the legth of the line to
display
  if(iDistance<40){
  line(pixsDistance*cos(radians(iAngle)),-pixsDistance*sin(radians(iAngle)),(width-
width*0.505)*cos(radians(iAngle)),-(width-width*0.505)*sin(radians(iAngle)));
  //line( start point by the calculated length and cos andle to get x cordinator, then sin with y
cordinator, then ending points
  }
  popMatrix();
}
```

```
//this get by the angle. same as above
void drawLine() {
  pushMatrix();
  strokeWeight(5);
  stroke(98,245,31);
  translate(width/2,height-height*0.074);
  line(0,0,(height-height*0.12)*cos(radians(iAngle)),-(height-height*0.12)*sin(radians(iAngle)));
  popMatrix();
}


//draw texts in graph
void drawText() {

  pushMatrix();
  if(iDistance>40) {
  noObject = "No threat";
  }
  else {
  noObject = "Attack detected";
  }

  textSize(20);
 fill(255,200,10);
 text("Defense System",100,100);

  fill(255,255,255);
  noStroke();
  rect(0, height-height*0.0648, width, height);
  fill(0,255,0);
  textSize(25);
  text("10cm",width-width*0.3854,height-height*0.0833);
  text("20cm",width-width*0.281,height-height*0.0833);
  text("30cm",width-width*0.177,height-height*0.0833);
  text("40cm",width-width*0.0729,height-height*0.0833);

  textSize(40);
  fill(0,0,0);
  text("Object: " + noObject, width-width*0.875, height-height*0.0277);
  text("Angle: " + iAngle +" °", width-width*0.48, height-height*0.0277);
  text("Distance: ", width-width*0.26, height-height*0.0277);
  if(iDistance<40) {
  text("  " + iDistance +" cm", width-width*0.140, height-height*0.0277);
  }

  textSize(25);
  fill(98,245,60);
  translate((width-width*0.4994)+width/2*cos(radians(30)),(height-height*0.0907)-
width/2*sin(radians(30)));
  rotate(-radians(-60));
  text("30°",0,0);
  resetMatrix();
  translate((width-width*0.503)+width/2*cos(radians(60)),(height-height*0.0888)-
width/2*sin(radians(60)));
  rotate(-radians(-30));
  text("60°",0,0);
  resetMatrix();
  translate((width-width*0.507)+width/2*cos(radians(90)),(height-height*0.0833)-
width/2*sin(radians(90)));
  rotate(radians(0));
  text("90°",0,0);
  resetMatrix();
  translate(width-width*0.513+width/2*cos(radians(120)),(height-height*0.07129)-
width/2*sin(radians(120)));
  rotate(radians(-30));
  text("120°",0,0);
  resetMatrix();
  translate((width-width*0.5104)+width/2*cos(radians(150)),(height-height*0.0574)-
width/2*sin(radians(150)));
  rotate(radians(-60));
  text("150°",0,0);
  popMatrix();
}
```

## Physical Design and Structure of Prototype

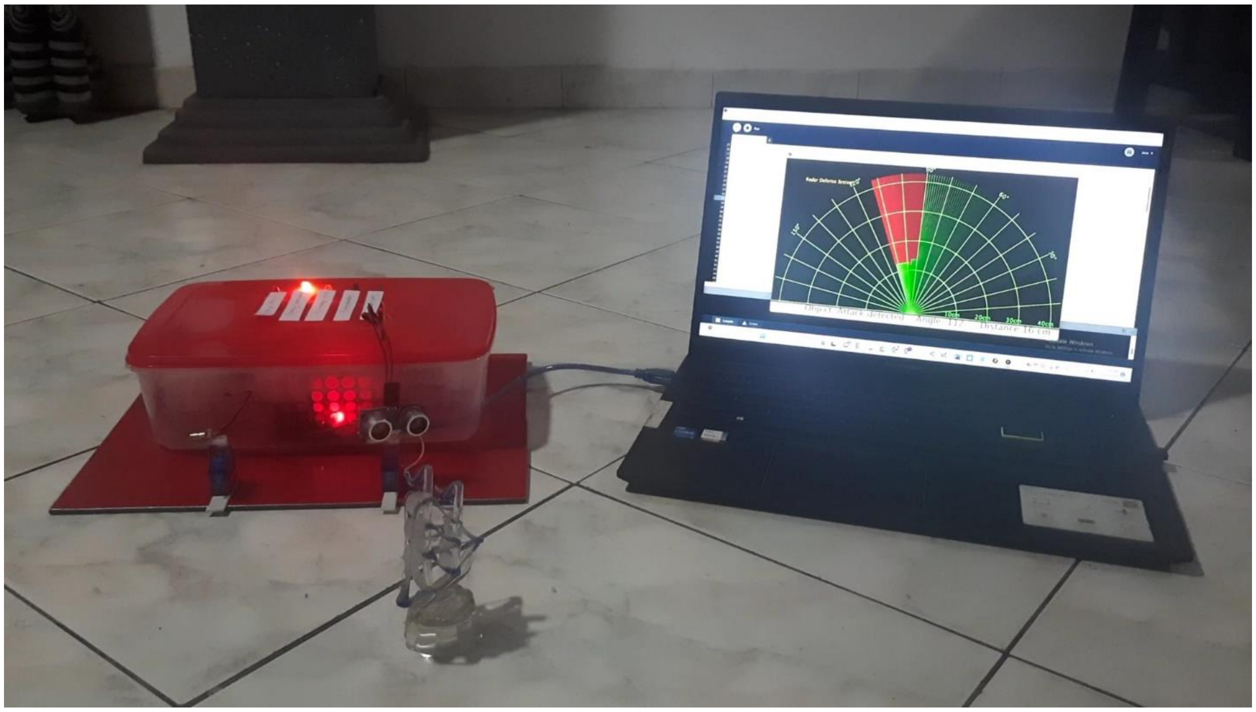


Figure 10 - Final View with Setup , object and Display

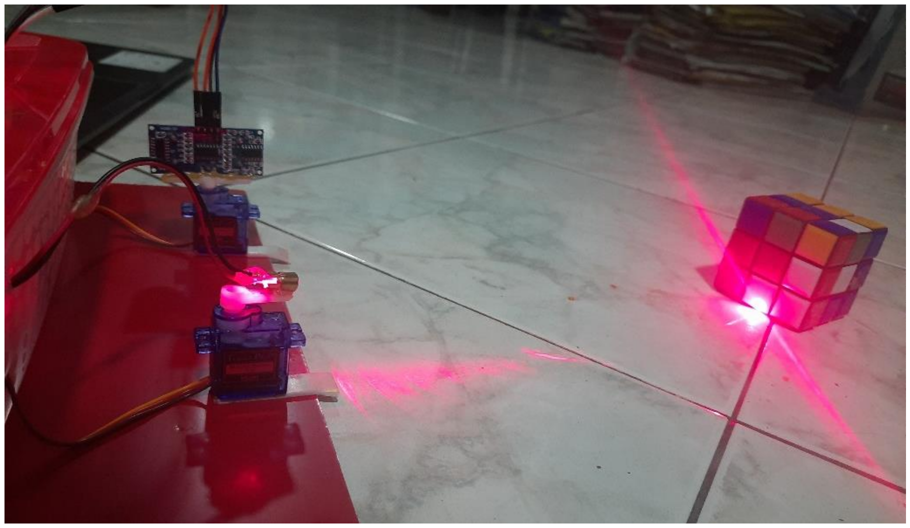


Figure 11 - In operation

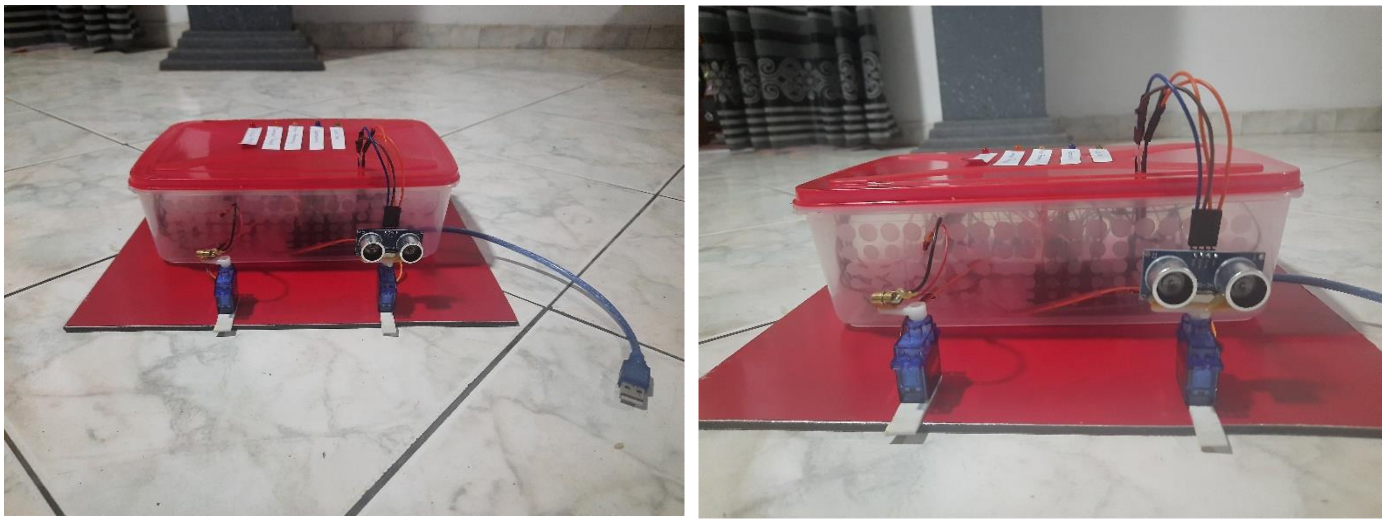


Figure 12 - Font View

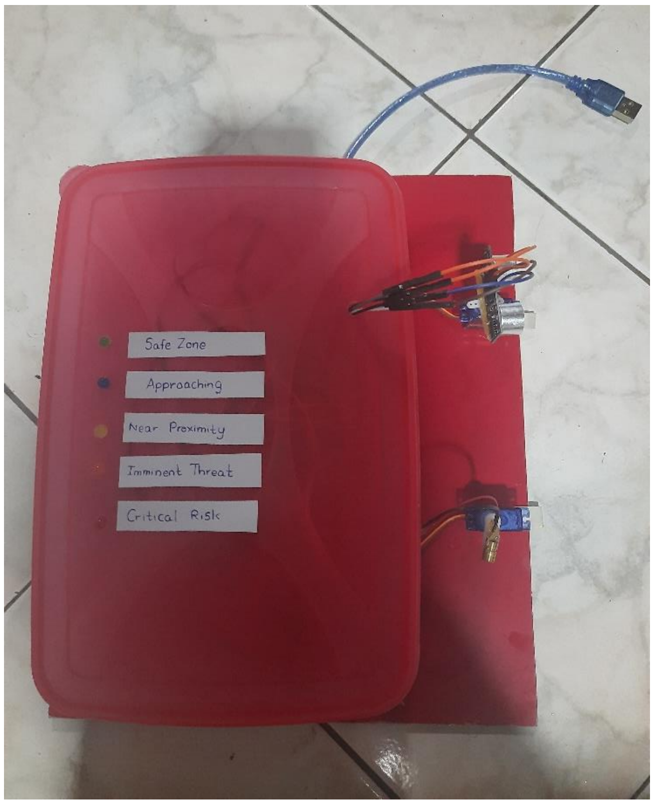


Figure 13 - Up View

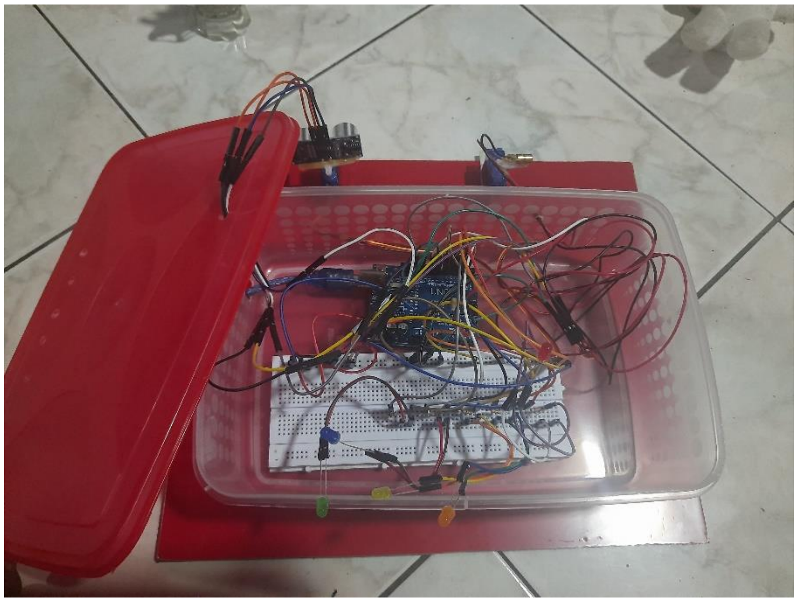


Figure 14 - Inside circuit

## Challenges and Issues

Before, I chose a 360-degree rotation servo motor but then realized I couldn't control the angle precisely. I then moved for a 180-degree motor. Prior to designing the system, I intended to use a laser to point out the exact angle from the moving motor, which carries the ultrasonic sensor. However, I encountered a problem. Aligning them in the same vertical line would make the laser not point directly at the object; it would end up slightly higher. To fix this, I used trigonometry (Sin Rule) to calculate another angle when the laser was positioned 10cm away from the ultrasonic sensor.

This was successful, but it led to another issue: the sensor detected the motor with the laser as an object. I do have solutions for this problem, but due to limited resources, I couldn't apply them to this project. When I enhance the project in the future, I can apply them. The solutions are:

- Firstly, since the project operates within a 40cm range, I could position the laser pointer beyond that range, as it wouldn't appear within the radar area.

- Secondly, I could use another servo motor. I could place the laser in the same vertical line as the ultrasonic sensor and use an additional servo to adjust the angle so it doesn't point higher. The angle adjustment would be calculated based on the distance.

Sometimes, due to vibrations in the ultrasonic sensor, the system detects locations even if there's no object present. In such cases, the laser would point at those locations. I have a suggestion: I could enhance this project in the future to only respond to larger area objects, greater than 10 degrees. This way, it would ignore other small object detections, treating them as noise or errors.

## Limitations

- This system operates within a distance range of 0-40cm.
- It only detects objects on the ground.
- It functions effectively for angles between 0 and 180 degrees, but it doesn't cover the backward direction from 180 to 0 degrees in this setup.

## Conclusion

In conclusion, the system reliably detects ground-based objects. However, occasional errors occur when the ultrasonic sensor mistakenly detects objects due to vibrations, even when there's no actual object present. To improve accuracy, implementing the solutions I suggested could enhance this project in the future.

## Future Enhancements

- Utilize Radar Display by Connecting the system to a radar display for visual representation.
- Position the laser where it doesn't disrupt the ultrasonic sensor's function.
- Create a mobile app for remote system monitoring and control.
- Introduce audio alerts to warn of detection or system status.
- Expand the system's range for broader coverage.
- Implement actual shooting mechanism.

## References

Draw.io, "Flowchart Maker & Online Diagram Software," *app.diagrams.net*. https://app.diagrams.net/

"Processing.org," *Processing.org*, 2019. https://processing.org/

YouTube, "YouTube," *YouTube*. Available: https://www.youtube.com/

MACFOS, "Home," *Robu.in | Indian Online Store | RC Hobby | Robotics*. https://robu.in/

wikipedia, "Wikipedia, the free encyclopedia," *Wikipedia*, 2023. https://en.wikipedia.org/

Arduino, "Arduino ," *Arduino.cc*, 2018. https://www.arduino.cc/