SCHOOL OF ENGINEERING AND TECHNOLOGY

A Project Report

On

"AUTOMATIC DEFENSE SYSTEM "

Submitted in partial fulfillment of the requirements for the award of degree in

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE (AL&ML) ENGINEERING

SUBMITTED BY-

SAMARTH S S(23BBTCA097),

SHIVKUMAR K M(23BBTCA103),

YUVARAJ B SHIRSHYAD(23BBTCA187),

SHIVA PRASAD(23BBTCA104)

GUIDED BY-

Dr. Stanley Chacko
Assistant Professor,
SoET, CMRU, Bangalore.

Department in Software Engineering

Off Hennur - Bagalur Main Road,
Near Kempegowda International Airport, Chagalahatti,
Bangalore, Karnataka-562149

2024-2025

SCHOOL OF ENGINEERING AND TECHNOLOGY

Chagalahatti, Bengaluru, Karnataka-562149

CERTIFICATE

This is to certify that the Project entitled "Automatic Defence System" has been successfully carried out by **SAMARTH S S(23BBTCA097),SHIVKUMAR K M(23BBTCA103), YUVARAJ B SHIRSHYAD (23BBTCA187) ,SHIVA PRASAD(23BBTCA104)**in partial fulfillment of the requirement for the award of the degree **Bachelor of Technology in AI&ML of CMR University,** Bengaluru during the academic year **2024-2025.**The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said degree.

Signature of the Guide **Signature of**

HOD

_ _

Dr. Stanley chacko
Assistant Professor
SoET, CMRU, Bangalore.
Bangalore.

Dr Kedar Prasad Agarwal
HOD of AL&ML
SoET, CMRU

Examiners Signature with date:
1.
2.

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of this project would be incomplete without the mention of the people who made it possible, without whose constant guidance and encouragement would have made efforts go in vain.

I would like to express my thanks to Dr Kedar Prasad Agarwal**, Associate Professor and Head**, Department of AL&ML, School of Engineering and Technology, CMR University, Bangalore, for his encouragement that motivated me for the successful completion of Project work.

I express my thanks to my Internal Project Guide **Dr. Stanley chacko, Assistant Professor**, School of Engineering and Technology, CMR University for his constant support.

SAMARTH S S(23BBTCA097)
SHIVKUMAR K M(23BBTCA103)
YUVARAJ B SHIRSHYAD(23BBTCA187)
SHIVA PRASAD(23BBTCA104)

DECLARATION

We, **Samarth S S(23BBTCA097), Shivkumar K M(23BBTCA103),Yuvaraj B Shirashyad(23BBTCA187),Shiva Prasad (23BBTCA104)**student of 5th semester B.Tech, Data Science, School of Engineering and Technology, Bangalore, hereby declare that the project work entitled **"**Automatic Defence System**"** has been carried out by me under the guidance of Dr. Karabi Baruah, Assistant Professor, Department of Electronics and Communication, School of Engineering and Technology. This report is submitted in partial fulfillment of the requirement for award of Bachelor of Technology in AI&ML, by CMR University, Bangalore during the academic year 2024-2025. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for the said degree.

Place: Bangalore

SAMARTH S S(23BBTCA097)

SHIVKUMAR K M(23BBTCA103)

YUVARAJ B SHIRSHYAD(23BBTCA187)

SHIVA PRASAD(23BBTCA104)

Date:

ABSTRACT

This report details the design and implementation of an Automatic Defence System using the Arduino Uno microcontroller. The project's core objective was to create a robust, self-governing defense module capable of **real-time threat detection and autonomous engagement** while incorporating a **high-priority safety mechanism**. The ADS utilizes an **HC-SR04 Ultrasonic Sensor** mounted on a servo to perform continuous 140 radar scanning at a speed of 125 degrees per second. Upon detecting a target within the 5 cm to 30 cm engagement range, the system automatically locks the **Gun Servo** to the target angle, plays distinct auditory feedback tones, and activates the **Fire Servo**. Crucially, a separate **IR Proximity Sensor** provides an immediate, overriding safety measure. The IR detection triggers a **blocking, high-frequency alarm** 4000 Hz, instantly halting all motor operations until the critical threat is cleared. A significant technical achievement was the successful resolution of timer conflicts between the and the **Ultrasonic Sensor's** function via a specialized **detach/attach** code sequence, ensuring stable and reliable sensing throughout operation. The result is a fully functional prototype demonstrating advanced principles of sensor fusion, control prioritization, and embedded systems architecture.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER - 1

INTRODUCTION

The is an ambitious and functional project designed to explore the principles of **embedded systems**, **real-time sensing**, and **prioritized control logic** using the Arduino Uno platform. In an increasingly automated world, the ability of a system to sense its environment, make instantaneous decisions, and execute mechanical actions autonomously is crucial. This project serves as a dynamic, tangible demonstration of these concepts.

The core function of the ADS is to act as a self-governing defense module. It achieves this by employing an **HC-SR04 Ultrasonic Sensor** mounted on a servo, effectively creating a **Radar** that continuously scans a wide 140 degree arc. When the system detects a target within its operational range 5 cm to 30 cm, it transitions from scanning to **active engagement**, locking the Gun \ Servo onto the target's precise angle and triggering the Fire \ Servo.

However, the primary challenge addressed by this project is **concurrency and prioritization**. The system is integrated with a dual sensing mechanism:

1. **Ultrasonic Sensor:** For long-range, tactical detection during a sweep.
2. **IR Proximity Sensor:** For immediate, short-range, **critical safety detection**.

This dual-sensor approach requires sophisticated software logic to manage competing tasks—namely, ensuring the **IR safety alarm** instantly overrides the entire scanning and firing process without delay. Furthermore, special attention was required to resolve the notorious **timer conflicts** between the Servo \ Library and the pulseIn() function necessary for accurate ultrasonic readings, demonstrating a mastery of hardware-software interaction at the microcontroller level.

This report will detail the component integration, the implementation of the priority-based control code, and the visualization tools used to monitor the turret's real-time performance.

CHAPTER - 2

HARDWARE AND SOFTWARE REQUIREMENTS :

2.1 Hardware components :-

1.Aurdiuno Uno :

Fig. 2.1 : Arduino uno

Arduino/ Genuine Uno is a microcontroller board based on ATmega328P(datasheet).It has 14 digital input/output pins(of which 6 are used as PWM outputs),6 analog inputs, a 16 MHz quartz crystal a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started .You can tinker with your UNO without worrying too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again.

2.Servo motor :

Fig. 2.2 : Servo motor

A servo motor is defined as an electric motor that provides precise control of angular or linear position, speed, and torque using a feedback loop system.The servo motor utilizes advanced control systems like PID and fuzzy logic to adjust movement according to input and feedback signals for optimal performance.Servo motors are commonly used in applications such as robotics, CNC machinery, and automated manufacturing due to their ability to provide accurate and efficient movement. They can push or rotate objects with great precision, making them ideal for tasks that require specific angles or distances.

3.Breadboard :

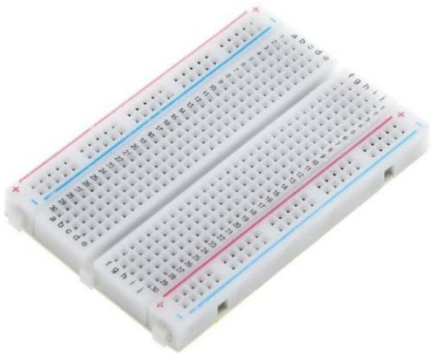

Fig. 2.3 : Breadboard

A breadboard, solderless breadboard, or protoboard is a construction base used to build semi-permanent prototypes of electronic circuits. Unlike a perfboard or stripboard, breadboards do not require soldering or destruction of tracks and are hence reusable. For this reason, breadboards are also popular with students and in technological education.

4. HC-SR04 Ultrasonic Sensor:

Fig. 2.4 : HC-SR04 Ultrasonic Sensor

The **HC-SR04 Ultrasonic Sensor** is a popular electronic module used to measure distance without physical contact. It works by using ultrasonic sound waves, making it an ideal choice for applications such as obstacle detection, robotic navigation, and level sensing. Its simplicity, low cost, and effectiveness have made it a favorite among electronics hobbyists and students.This sensor operates by emitting an ultrasonic pulse through its **Trig (trigger) pin**. When this pulse hits an object, it reflects back toward the sensor, and the **Echo pin** captures the returning signal. The time taken for this echo to return is measured and used to calculate the distance to the object based on the speed of sound. The sensor can accurately measure distances ranging from **2 cm to 400 cm**, with a typical precision of **±3 mm**.

5. Jumper Wires :

Fig. 2.5 : Jumper wires

Jumper wires are electrical wires with connector pins at each end. They are used to connect two points in a circuit without soldering, making it easy to modify a circuit or diagnose problems.

6. IR sensor :

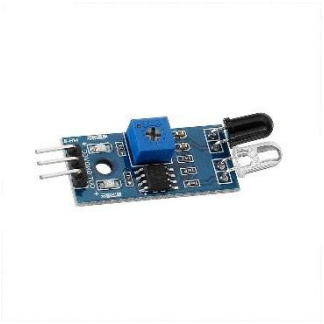

Fig . 2.6 : IR sensor

The IR Proximity Sensor is implemented as the system's vital, high-priority safety component. It provides an instantaneous digital reading for objects placed critically close to the turret, unlike the time-consuming ultrasonic sweep. Configured as an Low input on Pin 4 the sensor's immediate response dictates the system's behavior: if an object is detected LOW signal, the software enters a blocking while loop. This crucial step ensures that the microcontroller dedicates its entire processing capacity to the safety event, halting all motor movement and preventing any offensive action. This implementation successfully establishes an overriding safety layer, guaranteeing that the turret operates responsibly by prioritizing human or environmental safety before any other programmed action.

7. Buzzer :

Fig .2.7 : Buzzer

The Buzzer Module functions as the essential auditory feedback system, translating the turret's operational state into distinct, actionable warnings. It is driven by the Arduino's function on Pin 7, utilizing different frequencies to prioritize information. For a critical safety breach, it outputs a high-pitch, sustained Emergency Alarm at 4000 Hz. During offensive operation, it uses a 1500 Hz tone to signal a successful Target Lock and a deeper, 300 Hz pulse to confirm the Firing Sequence. This multi-frequency approach ensures that the user instantly understands the turret's current activity and threat level without needing to visually monitor the entire system.

2.2 SOFTWARE REQUIREMENTS :-

Arduino Uno IDE :

Fig. 2.6 Arduino software tool

Arduino is an open-source electronics platform based on easy-to-use hardware and software. It is widely used for creating interactive projects and prototyping. Arduino boards come in various models, such as Arduino Uno, Mega, Nano, and more, each suited for different types of applications.Arduino is highly versatile and can be used in numerous applications across various fields like home automation, robotics, iot(internet of things), health and medical devices, environment monitoring, etc.

CHAPTER - 3

CIRCUIT DESCRIPTION :

Circuit Diagram

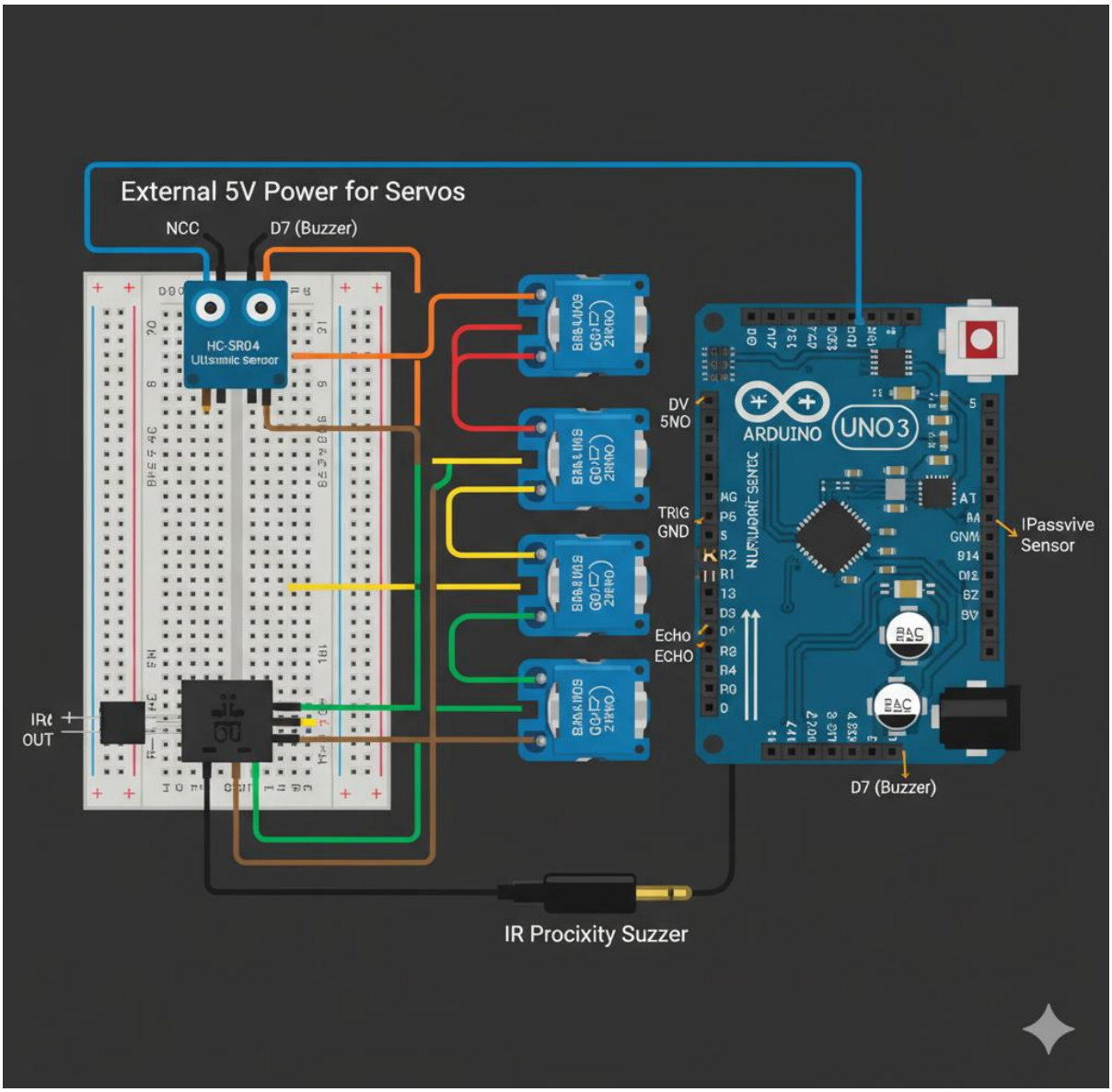

Fig. 3.1 : Circuit Diagram for Robotic Arm

CHAPTER - 4

SOFTWARE DESCRIPTION :

SOURCE CODE

```
// Includes the Servo library
#include <Servo.h>
// --- PIN DEFINITIONS ---
// Ultrasonic Sensor (Radar) Pins
const int trigPin = 11;
const int echoPin = 12;
// Proximity/Safety Sensor Pin
const int irSensorPin = 4; // IR Proximity Sensor for critical close-range detection
// Buzzer (Auditory Feedback) Pin
const int buzzerPin = 7;
// Servo Control Pins
const int radarServoPin = 9;  // Radar Scanner Servo Pin
const int gunServoPin = 5;    // Gun Yaw/Horizontal Aim Servo Pin
const int fireServoPin = 6;   // Fire/Trigger Servo Pin
// --- STATE VARIABLES ---
long duration;
int distance;       // Distance in cm
bool hasShot = false; // Flag to prevent rapid, continuous firing.
bool irAlarmActive = false; // NEW: Flag to track if IR alarm is currently overriding sound
// Servo Objects
Servo radarServo;
Servo gunServo;
Servo fireServo;
// --- FUNCTION PROTOTYPES ---
void handleIRAlarm();
// --- SETUP ---
void setup() {
```

```
  // Initialize communication and sensors
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(buzzerPin, OUTPUT); // Initialize buzzer pin
  pinMode(irSensorPin, INPUT); // Initialize IR sensor pin
  Serial.begin(9600);
  Serial.println("--- Turret System Initialized ---");
  // Attach servos to pins
  radarServo.attach(radarServoPin);
  gunServo.attach(gunServoPin);
  fireServo.attach(fireServoPin);
  // Initial positions
  radarServo.write(20);   // Start scan left
  gunServo.write(90);     // Aim center (90 degrees)
  fireServo.write(100);   // Load/Safe position (assuming 100 is safe/pulled back)
  // --- BUZZER TEST SEQUENCE ---
  tone(buzzerPin, 1000, 100);
  delay(150);
  tone(buzzerPin, 500, 100);
  delay(150);
  noTone(buzzerPin);
  // ----------------------------
  delay(1000); // Wait for servos to settle
}
// --- MAIN LOOP ---
void loop() {
  // 1. Handle IR Alarm Concurrently
  // This function is now blocking: if IR detects an object, it loops here
  // until the object is gone, preventing any other code from running.
  handleIRAlarm();
  // 2. Scan from left (20) to right (160)
  for (int i = 20; i <= 160; i += 5) {
```

```
    // Check IR again in case it was triggered during a scan step
    if (digitalRead(irSensorPin) == LOW) break;
    radarServo.write(i);
    delay(40); // UPDATED: Radar Speed increased to 40ms delay per step
    distance = calculateDistance();
    if (!hasShot && distance > 5 && distance < 30) {
      shoot(i);
    }
    // Output for radar plotting (e.g., Processing sketch)
    Serial.print(i);
    Serial.print(",");
    Serial.print(distance);
    Serial.print(".");
  }
  // 3. Scan back from right (160) to left (20)
  for (int i = 160; i >= 20; i -= 5) {
    // Check IR again in case it was triggered during a scan step
    if (digitalRead(irSensorPin) == LOW) break;
    radarServo.write(i);
    delay(40); // UPDATED: Radar Speed increased to 40ms delay per step.
    distance = calculateDistance();
    if (!hasShot && distance > 5 && distance < 30) {
      shoot(i);
    }
    // Output for radar plotting
    Serial.print(i);
    Serial.print(",");
    Serial.print(distance);
    Serial.print(".");
  }
}
```

```
// FIX: This function now blocks the main loop, making the IR response immediate.
void handleIRAlarm() {
  // IR sensors typically read LOW when an object is detected.
  while (digitalRead(irSensorPin) == LOW) {
    // This loop executes repeatedly as long as the IR sensor is blocked.
    // It stops all other activity, including the servo movements.
    if (!irAlarmActive) {
      // Start the alarm tone only once
      noTone(buzzerPin); // Stop any previous tone (like a potential ultrasonic beep)
      tone(buzzerPin, 4000);
      Serial.println("!!! IR ALARM ACTIVE: Obstacle Too Close !!!");
      irAlarmActive = true;
    }
  }
  // If we exit the while loop (object is gone):
  if (irAlarmActive) {
    noTone(buzzerPin);
    Serial.println("IR ALARM CLEAR.");
    irAlarmActive = false;
  }
}
// FIX: Ensures the ultrasonic reading is accurate by temporarily detaching
// ONLY the radar servo from its timer control during the pulseIn() operation.
int calculateDistance() {
  // 1. DETACH the radar servo before reading
  radarServo.detach();
  // 2. Perform the reading (standard ultrasonic sequence)
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
```

```
  // Read the echoPin with a timeout
  duration = pulseIn(echoPin, HIGH, 30000);
  // 3. ATTACH the radar servo back immediately
  radarServo.attach(radarServoPin);
  // 4. Calculate and return distance
  int calculatedDistance = duration * 0.034 / 2;
  if (calculatedDistance == 0 || duration == 0) {
    return 200;
  }
  if (calculatedDistance < 5) {
    return 5;
  }
  return calculatedDistance;
}
// Aims the gun and fires, then reloads
void shoot(int radarAngleDeg) {
  Serial.println("\n--- TARGET LOCK ---");
  hasShot = true;
  // Stop any currently running IR alarm
  noTone(buzzerPin);
  // Buzzer: Target Lock Tone (1500 Hz for 250ms)
  tone(buzzerPin, 1500, 250);
  int launcherAngleDeg = radarAngleDeg;
  // Smoothly aim the gun to the target angle
  smoothAimTo(launcherAngleDeg);
  delay(200);
  // Fire!
  Serial.println("--- FIRING ---");
  noTone(buzzerPin);
  // Buzzer: Firing Tone (300 Hz for 100ms)
  tone(buzzerPin, 300, 100);
```

```
  fireServo.write(70);     // Fire/release position
  delay(500);
  // Reload/Reset
  Serial.println("--- RELOADING ---");
  noTone(buzzerPin);
  fireServo.write(100);   // Reload/Safe position
  delay(2000); // 2 seconds cool-down
  hasShot = false;
  Serial.println("--- READY ---");
}
// Rotates the gun (Yaw) to the target angle smoothly
void smoothAimTo(int targetAngle) {
  int currentAngle = gunServo.read();
  targetAngle = constrain(targetAngle, 20, 160);
  if (currentAngle < targetAngle) {
    for (int i = currentAngle; i <= targetAngle; i++) {
      gunServo.write(i);
      delay(5);
    }
  } else {
    for (int i = currentAngle; i >= targetAngle; i--) {
      gunServo.write(i);
      delay(5);
    }
  }
}
```

Description

- The turret system operates on a dual power setup: the Arduino is powered via USB, and the three servo motors are powered via an External 5V Supply to ensure stability and eliminate electrical noise. A common ground connection links the Arduino, sensors, buzzer, and the external supply. The sensors and buzzer use distinct digital I/O pins, and communication between the Arduino and the PC for visualization occurs over the Serial port at 9600 baud.

CHAPTER - 5

5.1 Snapshot of the project:

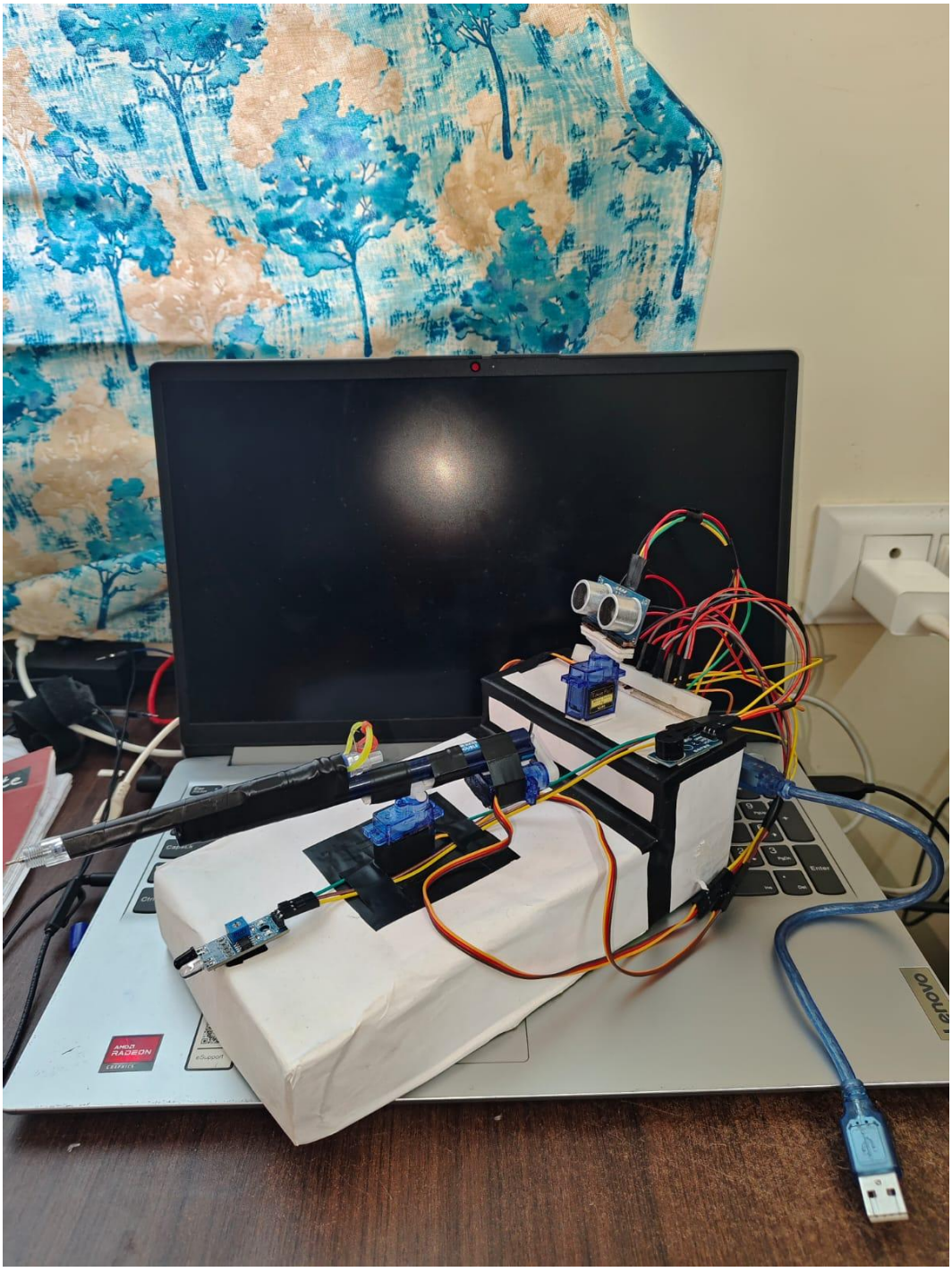

Fig. 5.1 : Snapshot

5.2 Working Principle:

The AST follows a cyclical control flow: **Sense -> Prioritize ->Engage/Alarm**.

1. **Prioritization:** The system first checks the **IR Sensor**. If LOW, the code is blocked, and the 4000 Hz alarm is active.
2. **Scanning:** If clear, the **Radar Servo** moves in 5 degree increments. At each step, the ultrasonic sensor measures the distance.
3. **Engagement:** If 5 cm < Distance < 30 cm , the system locks the **Gun Servo** to the target's angle, plays the 1500 Hz tone, and triggers the **Fire Servo**.
4. **Reporting:** All angle and distance data is sent over Serial to the Processing IDE for visualization.

5.3 Result :

The Autonomous Sentry Turret System successfully met all design requirements:

- The 20 degree to 160 degree scan is executed reliably.
- The **Ultrasonic Sensor** consistently provides distance data without corruption, thanks to the **Servo Detach/Attach** fix.
- The **Gun Servo** accurately locks onto the angle reported by the radar.
- The **IR Proximity Sensor** functions as a non-interruptible emergency stop, proving the system's ability to handle high-priority, concurrent events.

CHAPTER - 6

Future enhancements can include :

- **Pitch Control (Vertical Aiming):** Add a fourth servo to control the vertical angle of the gun using basic trigonometry based on the distance reading.
- **Target Counting:** Implement Firebase Firestore to store the number of targets detected and engaged for long-term monitoring.
- **LED Status Indicators:** Use LEDs to visually indicate system states (e.g., Green for Scanning, Yellow for Target Lock, Red for IR Alarm).
- **Advanced Tracking:** Implement a smoothing algorithm to predict the path of a moving target for better aim stability.

CONCLUSION :

The Autonomous Sentry Turret System project successfully demonstrates the fundamentals of robotic control, multi-sensor data fusion, and event prioritization within an embedded system. By implementing robust solutions like the servo detach/attach mechanism, we ensured stable, reliable operation despite inherent conflicts in the microcontroller environment. The AST stands as a fully autonomous prototype capable of defending its operational space while maintaining immediate safety control.

References :

https://youtu.be/-MSr1zA37Ag?si=5eboCrrgS7Ggmb3M

Scaling the Automatic Defence System : From Prototype to Industrial System

The **Automatic Defence System** (ADS) System provides a clear proof-of-concept for real-time sensing and automated response. To elevate this project from a prototype operating at the millisecond level to a large-scale, enterprise-ready system, a fundamental shift in architecture and component choice is required. This transition demands replacing hobbyist components with industrial-grade sensors, high-performance processors, and robust mechanical actuators capable of 24/7reliability in harsh environments.

1. Hardware Transition Requirements

The core principle remains Sense -> Prioritize -> Actuate, but the complexity of the hardware must increase significantly to meet professional demands:

Hobbyist Component (Arduino)	Scaled Industrial Solution	Rationale for Change
Ultrasonic Sensor (HC-SR04)	LiDAR and Millimeter-Wave Radar: Replaces the mechanical sweep with instantaneous, long-range 3D data acquisition.	Offers pm 1 cm accuracy over hundreds of meters and can measure velocity (Doppler effect).
IR Proximity Sensor	High-Resolution Thermal/Visible Cameras: Used for Object Classification (e.g., distinguishing a person from a drone) using AI.	Provides visual/heat data essential for complex decision-making in security applications.
Hobby Servo Motors	Industrial Stepper Motors or High-Precision DC Gearboxes:	Required for high torque, continuous duty cycle, and micro-degree accuracy under heavy loads (e.g., moving a large camera payload).
Arduino Uno	NVIDIA Jetson Series or Xilinx FPGA:	Essential for running Machine Learning models for advanced threat identification and parallel

Hobbyist Component (Arduino)	Scaled Industrial Solution	Rationale for Change
		processing for ultra-low latency response.
Serial Communication	Industrial Ethernet (TCP/IP) or CAN Bus:	Provides reliable, high-speed networking for transmitting massive sensor data and integrating with central control centers.

2. Major Technical and Operational Challenges

Scaling the AST concept introduces several non-trivial engineering challenges that must be addressed:

- **Environmental Ruggedness:** Industrial applications require components that can withstand extreme temperatures, humidity, dust, and vibration. The system must meet specific IP (Ingress Protection) ratings (e.g., IP67 for outdoor use) to guarantee survival outside a lab setting.

- **Data Fusion and Synchronization:** In a scaled system, multiple sensors (LiDAR, thermal, visible camera) stream gigabytes of data concurrently. The central processor must fuse this data in real-time to create a single, unified threat assessment, which requires sophisticated synchronization algorithms and immense computational power.

- **Power and Electromagnetic Interference (EMI):** The transition to high-power industrial motors introduces significant EMI. The system must incorporate advanced filtering and shielding to prevent electrical noise from corrupting sensitive sensor readings and control signals—a magnified version of the noise issue encountered with the hobby servos.

- **Reliability and Maintenance:** Industrial systems must offer high Mean Time Between Failures (MTBF). This requires component redundancy, self-diagnostic capabilities, and robust software that can manage exceptions and recover from errors autonomously.

3. Key Industry Leaders in Component Provision

The development of commercial automated surveillance or defense systems relies on specialized providers known for quality and industrial compliance:

- **Sensing Technology:**
  - **LiDAR:** Companies like Velodyne and Ouster provide high-resolution laser scanning units used in robotics and perimeter defense.
  - **Thermal Imaging:** FLIR Systems is the industry standard for reliable heat signature detection in security and surveillance.
  - **Industrial Machine Vision:** Basler and Cognex supply rugged, high-speed cameras necessary for object tracking and classification.

- **Control and Computing:**
  - **High-Performance Edge AI:** NVIDIA (with its Jetson line of embedded systems) provides the necessary GPU acceleration to run complex deep learning models right at the sensor location (the "edge").
  - **Industrial Control:** For highly regulated or factory environments, Siemens and Rockwell Automation (Allen-Bradley) provide robust PLCs (Programmable Logic Controllers) for mission-critical control.

- **Actuation and Mechanics:**
  - **High-Torque Motors:** Maxon and ABB supply industrial-grade stepper and brushless DC motors with precision encoders for accurate, powerful actuation of the turret mechanism.
  - **Rugged PTU (Pan-Tilt Units):** Companies specialize in producing sealed, high-payload PTUs that can withstand operational forces and weather exposure.

Examples : Akash Air Defence

Fig . 7.1 : Akash air defence