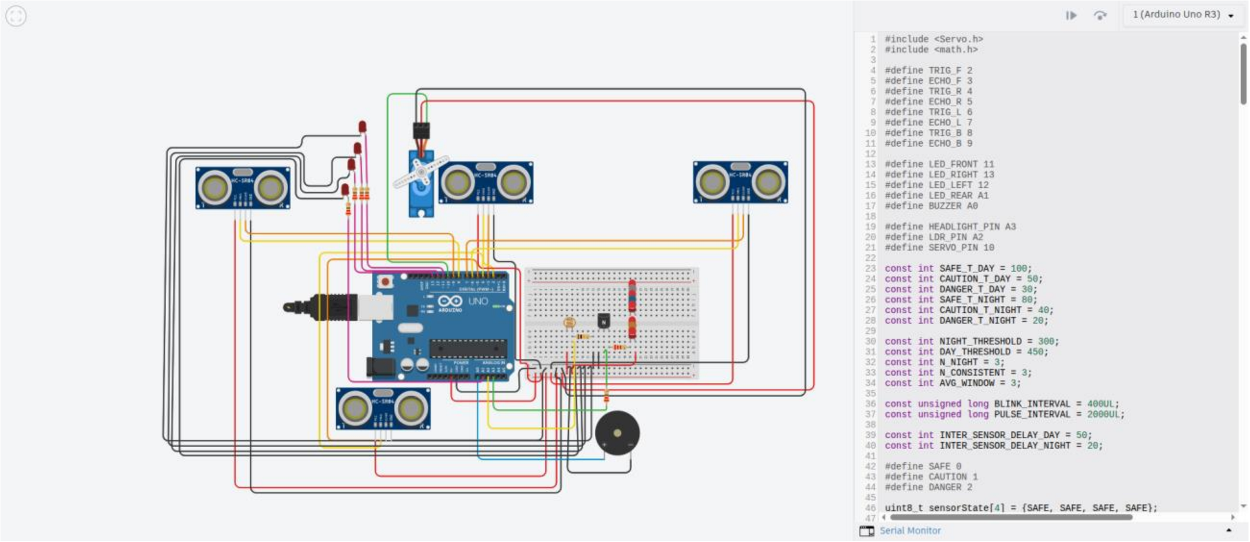
Advanced Radar Surveillance System with Multi-Sensor Fusion

Simulation Report

Name	Roll Number	Role
Izza Fatima	24I-0034	Dynamic Alerts & Night Path Detection
Adil Ibrahim	24I-0080	Object Detection, Servo Systems & Manual Control
Anum Ibrahim	24I-0113	Predictive Tracking & Movement Estimation
Abdul Azeem	24I-2013	Tilt Handling & Fault Management

Izza & Adil Simulation:



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

#define TRIG_F 2
#define ECHO_F 3
#define TRIG_R 4
#define ECHO_R 5
#define TRIG_L 6
#define ECHO_L 7
#define TRIG_B 8
#define ECHO_B 9

#define LED_FRONT 11
#define LED_RIGHT 13
#define LED_LEFT 12
#define LED_REAR A1
#define BUZZER A0

#define HEADLIGHT_PIN A3
#define LDR_PIN A2
#define SERVO_PIN 10

const int SAFE_T_DAY = 100;
const int CAUTION_T_DAY = 50;
const int DANGER_T_DAY = 30;
const int SAFE_T_NIGHT = 80;
const int CAUTION_T_NIGHT = 40;
const int DANGER_T_NIGHT = 20;

const int NIGHT_THRESHOLD = 300;
const int DAY_THRESHOLD = 450;
const int N_NIGHT = 3;
const int N_CONSISTENT = 3;
const int AVG_WINDOW = 3;

const unsigned long BLINK_INTERVAL = 400UL;
const unsigned long PULSE_INTERVAL = 2000UL;

const int INTER_SENSOR_DELAY_DAY = 50;
const int INTER_SENSOR_DELAY_NIGHT = 20;

#define SAFE 0
#define CAUTION 1
#define DANGER 2

uint8_t sensorState[4] = {SAFE, SAFE, SAFE, SAFE};
```

Responsible for:

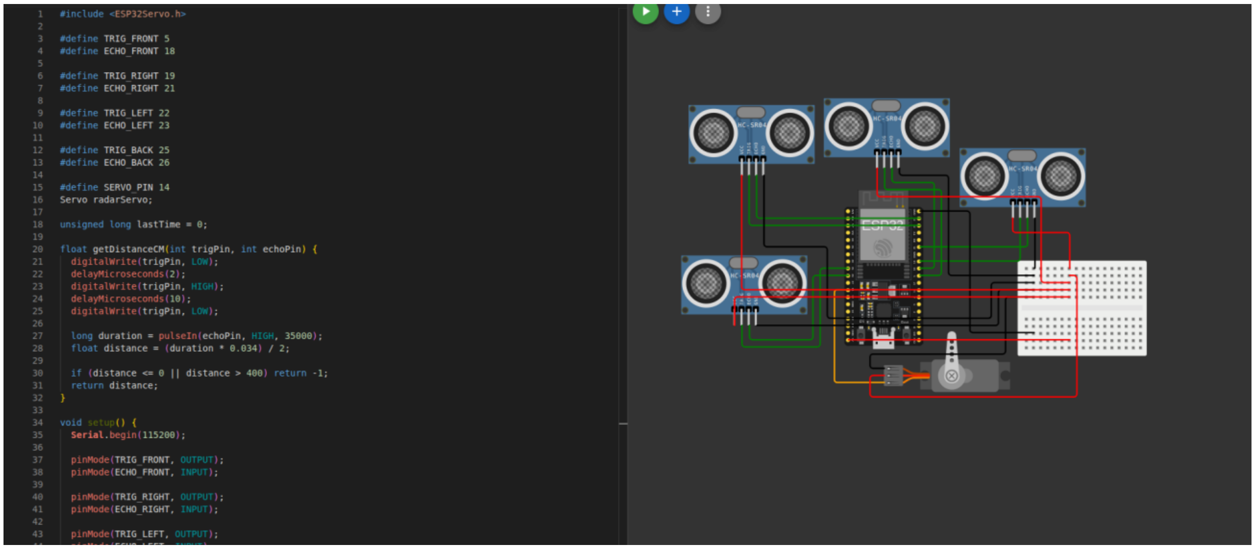
- **Feature 2:** Dynamic Alerts (LED + Buzzer)
- **Feature 6:** Night Path Detection
- **Feature 1:** Object Detection, Speed Calculation & Coordinate Logging

Description:
Implements the full alert system using LEDs and buzzer based on distance thresholds. Develops LDR-based night detection to activate headlights, adjust sweep speed, and enhance visibility during low-light conditions.

The code is attached to the folder.

Tinker cad Link: https://www.tinkercad.com/things/lHb5W4qITng-project-final/editel

Anum Simulation:



```
#include <ESP32Servo.h>

#define TRIG_FRONT 5
#define ECHO_FRONT 18

#define TRIG_RIGHT 19
#define ECHO_RIGHT 21

#define TRIG_LEFT 22
#define ECHO_LEFT 23

#define TRIG_BACK 25
#define ECHO_BACK 26

#define SERVO_PIN 14
Servo radarServo;

unsigned long lastTime = 0;

float getDistanceCM(int trigPin, int echoPin) {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  long duration = pulseIn(echoPin, HIGH, 35000);
  float distance = (duration * 0.034) / 2;

  if (distance <= 0 || distance > 400) return -1;
  return distance;
}

void setup() {
  Serial.begin(115200);

  pinMode(TRIG_FRONT, OUTPUT);
  pinMode(ECHO_FRONT, INPUT);

  pinMode(TRIG_RIGHT, OUTPUT);
  pinMode(ECHO_RIGHT, INPUT);

  pinMode(TRIG_LEFT, OUTPUT);
```

Responsible for:

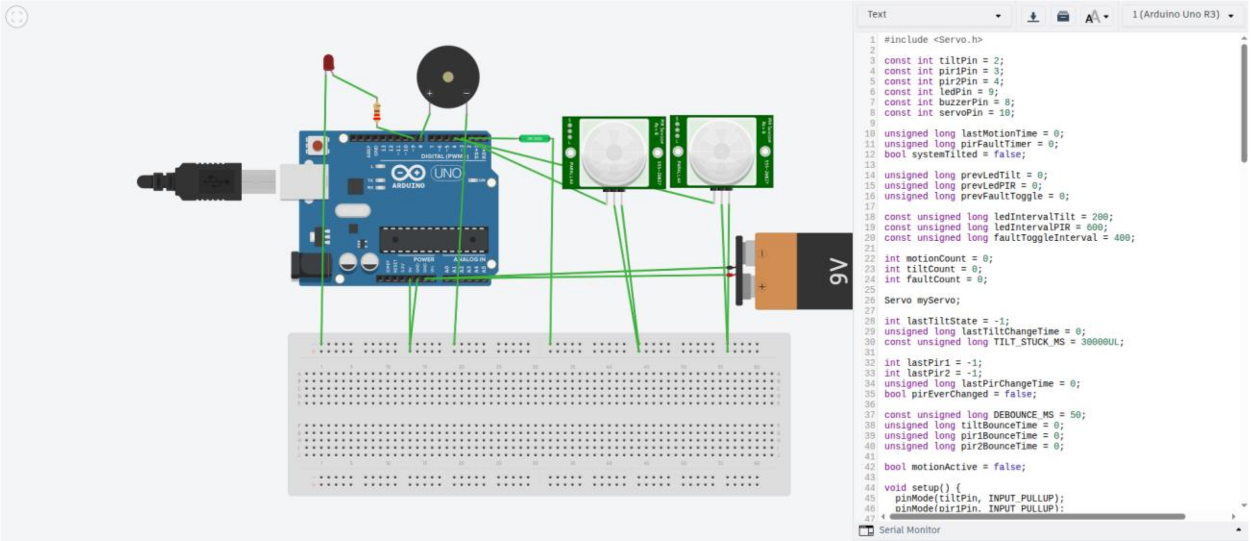
- **Feature 5:** Predictive Object Tracking

Description:
Builds predictive algorithms that estimate future object positions using sequential distance readings. Handles speed estimation, motion direction prediction, and integrates results into the Python dashboard for visualization.

The code is attached to the folder.

The wokwi Simulation link: https://wokwi.com/projects/448413997194664961

Azeem Simulation:



```
#include <Servo.h>

const int tiltPin = 2;
const int pir1Pin = 3;
const int pir2Pin = 4;
const int ledPin = 9;
const int buzzerPin = 8;
const int servoPin = 10;

unsigned long lastMotionTime = 0;
unsigned long pirFaultTimer = 0;
bool systemTilted = false;

unsigned long prevLedTilt = 0;
unsigned long prevLedPIR = 0;
unsigned long prevFaultToggle = 0;

const unsigned long ledIntervalTilt = 200;
const unsigned long ledIntervalPIR = 600;
const unsigned long faultToggleInterval = 400;

int motionCount = 0;
int tiltCount = 0;
int faultCount = 0;

Servo myServo;

int lastTiltState = -1;
unsigned long lastTiltChangeTime = 0;
const unsigned long TILT_STUCK_MS = 30000UL;

int lastPir1 = -1;
int lastPir2 = -1;
unsigned long lastPirChangeTime = 0;
bool pirEverChanged = false;

const unsigned long DEBOUNCE_MS = 50;
unsigned long tiltBounceTime = 0;
unsigned long pir1BounceTime = 0;
unsigned long pir2BounceTime = 0;

bool motionActive = false;

void setup() {
  pinMode(tiltPin, INPUT_PULLUP);
  pinMode(pir1Pin, INPUT_PULLUP);
```

Responsible for:

- **Feature 7:** Tilt & Uneven Surface Handling
- **Feature 8:** Fault Handling

Description:

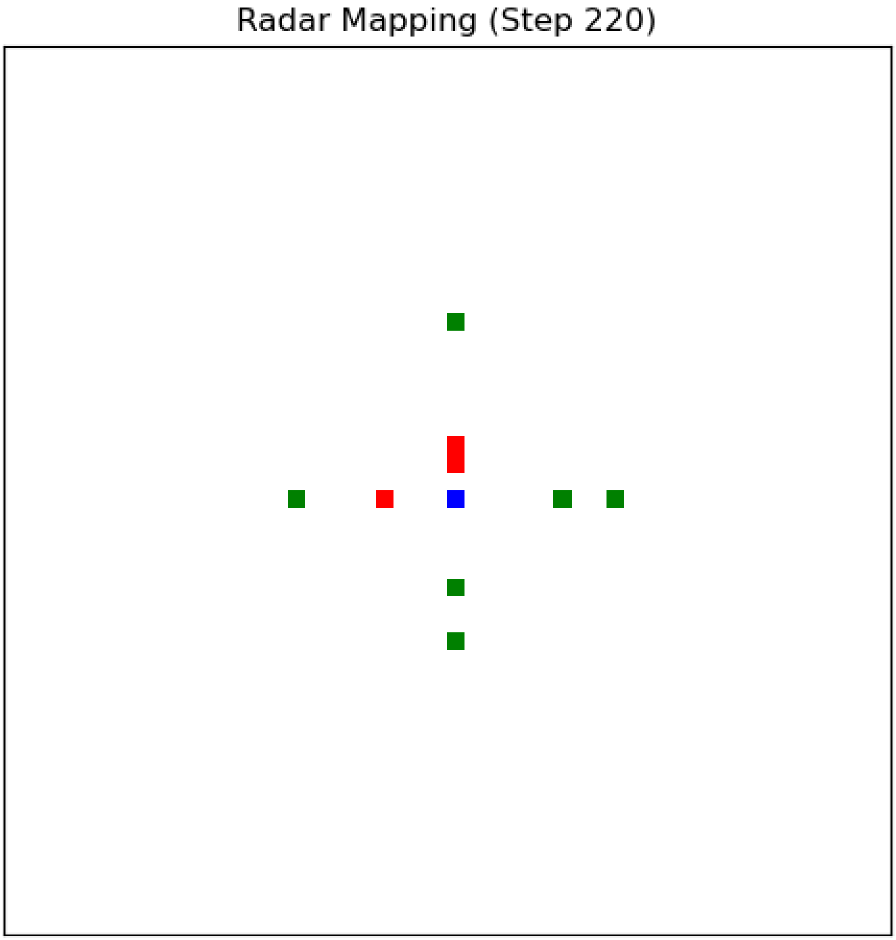
Implements tilt/gyroscope and IR sensor logic to detect uneven ground and directional instability. Adds robust error-handling mechanisms to keep the system operational even when specific sensors fail or return invalid readings.

The code is attached to the folder.

Tinker cad link is: https://www.tinkercad.com/things/j5AQiDa0vyp-mighty-habbi-juttuli/editel?returnTo=https%3A%2F%2Fwww.tinkercad.com%2Fdashboard&sharecode=iBktqsLriJ8pXefKNtw9AGKiKuRDEuoUFP8nATlakNg

2D Radar Mapping Python:

The code simulates a **2D radar mapping system** for a mobile robot/car using **sensor data** from multiple directions (front, back, left, right). It visualizes detected obstacles and the vehicle's position on a **grid**, providing a dynamic representation of the environment. The system can work with **live sensor data via MQTT** or **dummy data** for testing.



The code is attached to the folder.