Ultrasonic Radar System
IOT PROJECT

(By SEMESTER – VIII of IV Year M.Sc. (CA&IT) (2024-25))

Submitted By:

Name	Roll Number
Kadia Vedant Ketankumar	4031
Kanzariya Anand Dineshbhai	4118

GROUP ID: 08

Date of Submission : 05/05/2025

Submitted To

K. S. School of Business Management
M.Sc. - Computer Application and Information Technology.

ACKNOWLEDGEMENT

A project is a learning experience for the students. We consider it a great opportunity to be part of the project. It has been a privilege to have new experience and learn from it all. A lot of hard work, dedication and research have gone into making of this project.

However, it would not have been possible without the kind support and help of many individuals and organizations I would like to extend my sincere thanks to all of them.

We are indebted to our college ***KS School of Business Management*** for their guidance and constant supervision as well as for providing necessary information regarding the project and also for their support in completing the project.

We would like to express our special gratitude and thanks to the people related to this field for providing us with valuable information and giving us such attention and time.

Thanking You
Kadia Vedant
Kanzariya Anand

LIMITA

INDEX

CHAPTER : 1

PROJECT OVERVIEW

1.1 PROJECT TITLE : ULTRASONIC RADAR SYSTEM

The Ultrasonic Radar System IoT project is designed to detect objects using ultrasonic technology and display the results visually on a laptop screen. The system integrates components such as an ultrasonic sensor, Arduino Uno, servo motor, buzzer, LEDs, and basic prototyping tools like a breadboard and jumper wires. As the servo motor rotates the ultrasonic sensor, the system scans its surroundings and identifies objects based on distance measurements. Detected objects trigger alerts via LEDs and a buzzer, while real-time data is transmitted to a laptop for radar-style visualization. This project has potential applications in basic security systems, obstacle detection, and learning environments, demonstrating a practical blend of sensor technology, automation, and real-time visualization.

1.2 PROBLEM STATEMENT:

In today's world, surveillance and object detection systems play a critical role in ensuring security and automation across various sectors, including defense, transportation, and smart homes. However, many traditional systems are expensive, complex, and not easily accessible for small-scale or educational applications. There is a need for a cost-effective, compact, and easy-to-implement radar system that can detect objects and monitor surroundings in real-time.

This project aims to develop an IoT-based radar system using Arduino Uno, an ultrasonic sensor, servo motor, LED light, and buzzer. The system will scan the surroundings using a rotating ultrasonic sensor, detect nearby objects, and provide visual and auditory alerts. The addition of IoT capabilities enables remote monitoring and data visualization, making the system scalable and useful for a variety of practical applications.

1.3 OBJECTIVE OF THE PROJECT:

1) Enhanced Sensing Capabilities

- To develop a radar-like system using ultrasonic sensors and a servo motor that can accurately detect the presence and distance of objects within a defined scanning range.

2) Real-Time Monitoring and Analysis

- To implement real-time data acquisition and processing that enables immediate detection of obstacles and provides timely feedback via LEDs and buzzers.

3) IoT Integration for Remote Access

- To enable remote monitoring and control of the radar system through IoT connectivity, allowing users to access sensor data, receive alerts, and adjust parameters from any internet-enabled device.

4) Robust and Reliable Object Detection

- To ensure reliable object detection by optimizing sensor placement, sweep angles, and signal processing logic, thereby reducing false positives or negatives.

5) Scalability and Extensibility

- To design the system in a modular fashion so it can be extended with additional features such as cloud storage, mobile notifications, or AI/ML-based prediction and automation in future versions.

6) User-Friendly and Cost-Effective Design

- To create a compact, affordable, and easy-to-use prototype that can be used for educational purposes, home automation, or as a proof-of-concept for more advanced radar systems.

1.4 SCOPE OF THE PROJECT :

1) **Ultrasonic-Based Object Detection**
   - Utilizing ultrasonic waves to detect objects and measure their distance, simulating radar behavior through signal reflection and time-of-flight calculations.

2) **Rotational Scanning Mechanism**
   - Employing a servo motor to rotate the ultrasonic sensor and perform a continuous sweep of the surroundings to detect objects in a semi-circular area.

3) **Real-Time Alerts and Indicators**
   - Providing immediate feedback using LED lights and buzzers when an object is detected within a predefined range, enhancing system responsiveness and awareness.

4) **IoT Connectivity (Optional/If Implemented)**
   - Integrating IoT modules (e.g., ESP8266, NodeMCU) to allow wireless data transmission, enabling users to monitor radar outputs remotely via a web dashboard or mobile app.

5) **Visualization and Data Logging (If Implemented)**
   - Representing the scanning results visually on a serial monitor or external display (e.g., GUI or radar map) and optionally logging data for further analysis.

6) **Modular and Scalable Design**
   - Designing the system to be modular, making it easier to upgrade with additional features such as machine learning, AI-based pattern recognition, or integration into larger automation systems.

7) **Educational and Practical Use Cases**
   - Making the system accessible for educational use in embedded systems and IoT training, as well as potential applications in home security, robotics, and obstacle avoidance.

1.5 APPLICATIONS AND USE CASES :

1. Home and Property Security

- Detects unauthorized movement or intrusion in restricted areas.
- Can be integrated into home automation systems to trigger alarms, lights, or camera recording when motion is detected.

2. Obstacle Detection in Robotics

- Helps mobile robots or autonomous vehicles detect nearby objects and navigate safely.
- Used in line-following or maze-solving robots to avoid collisions.

3. Smart Parking Systems

- Detects the presence or absence of vehicles in parking spaces.
- Sends real-time data to a central system to manage available parking slots.

4. Industrial Automation and Safety

- Used in factories and warehouses for collision avoidance in automated guided vehicles (AGVs) or robotic arms.
- Detects human presence in restricted or dangerous zones.

5. Military and Surveillance Systems

- Serves as a basic prototype for perimeter surveillance and intruder detection in defense zones.
- Can be scaled for monitoring activity in specific areas using multiple units.

6. Environmental Monitoring

- Detects moving objects or animals in wildlife monitoring projects.
- Can be deployed in forests or rural areas to monitor animal activity without human presence.

7. Educational and Research Projects

- Ideal for students and researchers to learn about ultrasonic sensing, servo control, IoT integration, and embedded system design.
- Serves as a hands-on learning tool in electronics, robotics, and IoT courses.

8. Smart Traffic Monitoring

- Can be used to detect vehicles at intersections or pedestrian crossings.
- Helps in triggering traffic lights or sending vehicle count data to a central system.

CHAPTER : 2

REQUIREMENT ANALYSIS

2.1 FUNCTIONAL REQUIREMENT :

1) Object Detection:

➢ The system should be able to detect objects within its vicinity using ultrasonic technology.

2) Servo Motor Control:

♋☝ It should control the movement of the servo motor to scan the surrounding environment.

3) Data Processing:

♋☝ The system should process the sensor data to accurately identify objects and their distances.

4) Real-Time Monitoring:

♋☝ It should provide real-time monitoring of the detected objects.

5) Alerting System:

♋☝ The system should alert users when predefined conditions are met, such as detecting an obstacle.

6) User Interface:

♋☝ It should have a user-friendly interface for controlling and monitoring the system.

2.2 NON-FUNCTIONAL REQUIREMENT:

1) Reliability:

- The system should be reliable, providing accurate detection and minimal false alarms.

2) Performance:

- It should have high performance, with fast response times and efficient data processing.

3) Scalability:

- The system should be scalable to accommodate future upgrades or enhancements.

4) Security:

- It should have security features to protect against unauthorized access or data breaches.

5) Usability:

- The system should be easy to use, with a simple and intuitive interface.

6) Compatibility:

- It should be compatible with different devices and operating systems.

7) Maintainability:

- The system should be easy to maintain, with modular components that can be replaced or upgraded easily.

2.3 USER REQUIREMENT:

1) Detection Accuracy:

- Users expect the system to accurately detect objects and their distances.

2) Real-Time Monitoring:

- Users want to monitor the system in real-time to ensure the safety of their environment.

3) Customizable Settings:

- Users should be able to customize the detection range and other settings according to their requirements.

4) Ease of Use:

- Users expect the system to be easy to use, with a simple and intuitive interface.

5) Reliability:

- Users require the system to be reliable, providing consistent performance without frequent breakdowns.

2.4 SYSTEM REQUIREMENT:

1) Hardware:

- The system requires hardware components such as an ultrasonic sensor, Arduino Uno, servo motor, buzzer, LEDs, jumper wires, and a breadboard.

2) Software:

- Arduino IDE for programming and Processing or Python for radar visualization on a laptop.

3) Power Supply:

- Powered via USB from a laptop or a 9V battery/external adapter if used independently.

4) Connectivity:

- USB connection for data transfer between Arduino and laptop Compatibility.
- The system should be compatible with different devices and operating systems for ease of use.

5) Storage:

- It may require storage for storing sensor data or system settings.

CHAPTER: 3

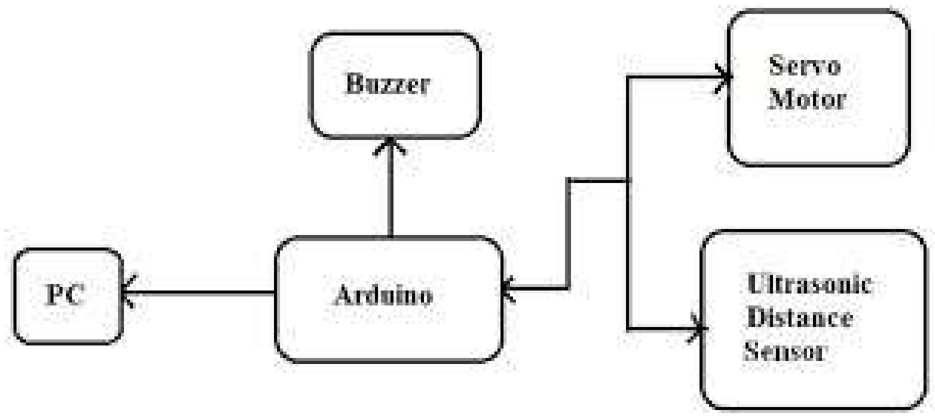
SYSTEM ARCHITECTURE AND DESIGN

3.1 SYSTEM OVERVIEW:

The **Ultrasonic Radar System** is a compact and cost-effective object detection system built using an Arduino Uno**,** ultrasonic sensor**,** servo motor**,** buzzer**,** and LEDs**,** with real-time visualization on a laptop screen. The system works by rotating the ultrasonic sensor using a servo motor to scan the surrounding area. It emits ultrasonic pulses and calculates distances by analyzing the reflected signals. Detected objects are indicated with visual alerts (LEDs), audio signals (buzzer), and a radar-style display on a computer screen via serial communication. This project serves as a foundational model for applications like basic surveillance, obstacle detection, and educational demonstrations. While primarily locally operated, it offers future scope for IoT integration such as data logging or remote access.

3.2 HARDWARE DESIGN:

3.2.1 DIAGRAMS:



3.2.2 COMPONENT LIST

Hardware Components :

1) Arduino Uno

- ➢ Microcontroller board serving as the brain of the system, responsible for processing sensor data, controlling the servo motor, and managing communication with other devices.

2) Ultrasonic Sensor

- ➢ Detects objects and measures their distance using ultrasonic waves, providing input to the Arduino for object detection and tracking.

3) Servo Motor

- ➢ Actuates the ultrasonic sensor, allowing it to scan the surrounding environment and detect objects within its range.

4) Jumper Wires

- ➢ Connectors used to establish electrical connections between the Arduino, ultrasonic sensor, servo motor, and other components, facilitating the assembly and wiring of the system.

5) Buzzer

- ➢ Audible alert component emitting sound signals based on input from the Arduino, providing a supplementary means of notification or warning in the radar system.

6) Laptop/PC/Android Phone

- ➢ Devices used to monitor the radar system remotely via IoT connectivity, accessing real-time data, receiving alerts, and controlling system parameters through a user-friendly interface

6) LED

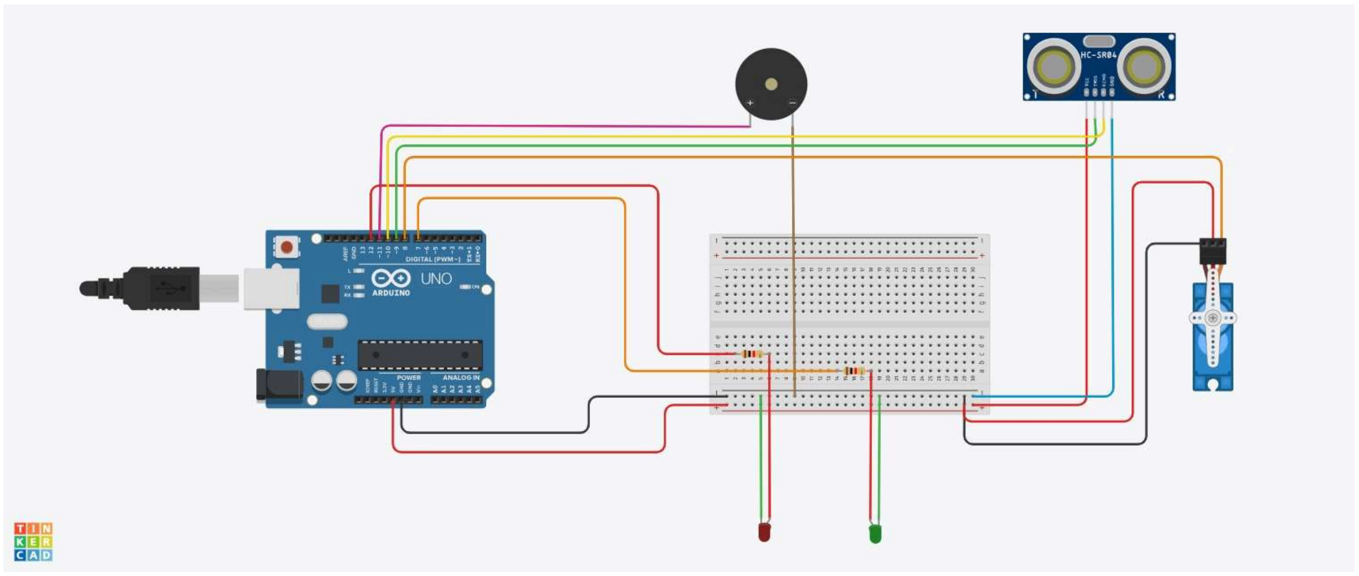
- ➢ Provides a *visual indicator* for object detection or system status. The LED can turn on or blink when an object is detected within a certain distance.

Software Tools:

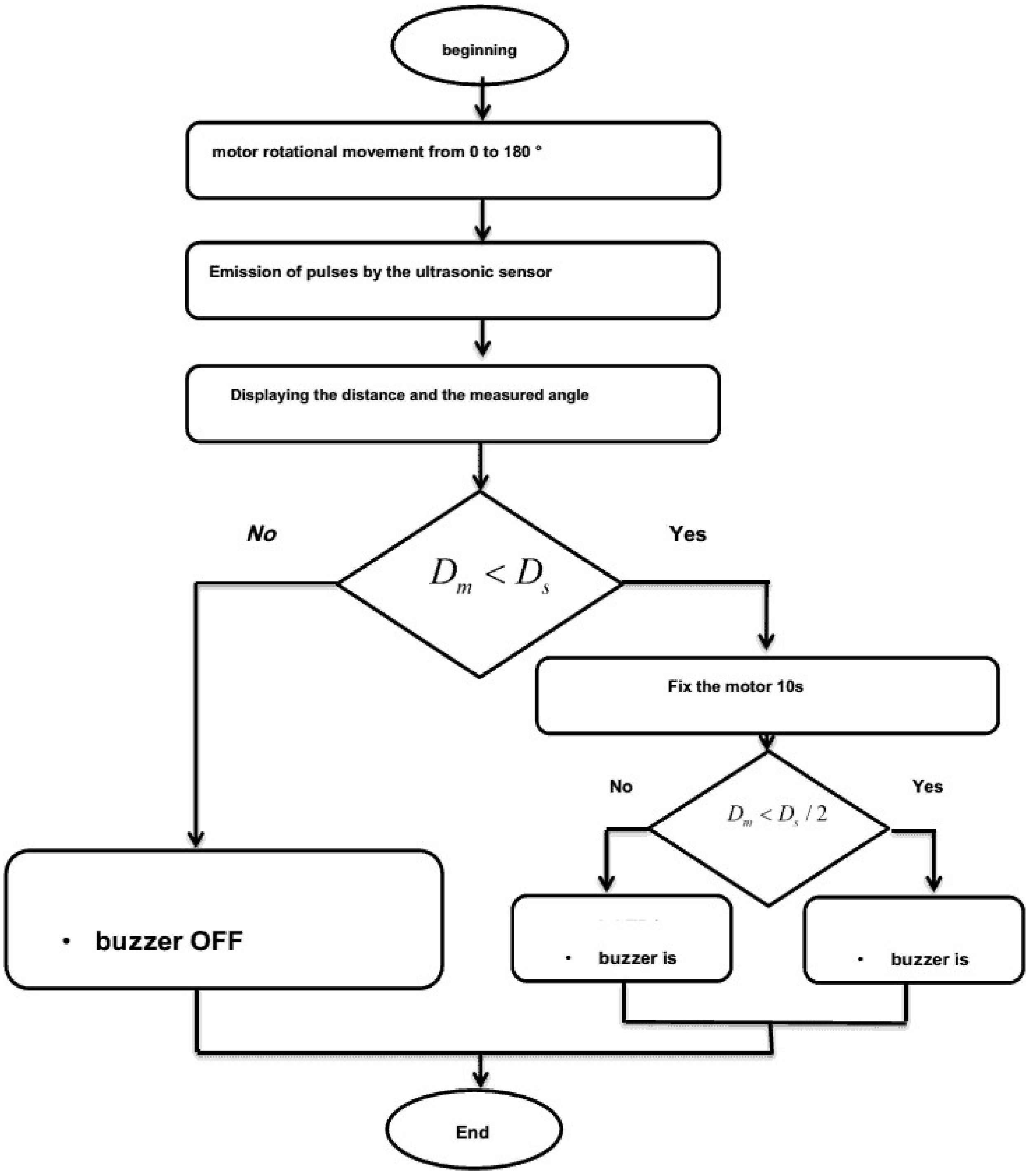
1) **Arduino IDE**
   - ➢ Used to write, compile, and upload code to the Arduino Uno.
   - ➢ Supports C/C++-based programming and serial communication for debugging and testing.
2) **Processing**
   - ➢ A flexible software sketchbook and language for visualizing data from the Arduino.
   - ➢ Used to create a graphical radar interface that displays object position and distance based on sensor input.

3.3 SOFTWARE DESIGN:

3.3.1 System Architecture Diagram:



3.3.2 Data Flow Diagrams:


beginning
motor rotational movement from 0 to 180 °
Emission of pulses by the ultrasonic sensor
Displaying the distance and the measured angle
No
Yes
$D_m < D_s$
Fix the motor 10s
No
Yes
$D_m < D_s / 2$
• buzzer OFF
• buzzer is
• buzzer is
End

CHAPTER : 4

IMPLEMENTATION DETAILS

4.1 HARDWARE SETUP:

STEP:1

- Mount the ultrasonic sensor at the front of the system to detect objects.

STEP2:

- Attach the servo motor to the ultrasonic sensor to enable scanning of the environment.

STEP3:

- Connect the ultrasonic sensor and servo motor to the Arduino board for data processing and control.

STEP4:

- Ensure a stable power supply for continuous operation of the system.

STEP5:

- Optionally, set up connectivity modules for remote monitoring and control.

4.2 SOFTWARE DEVELOPMENT:

STEP1:

- Develop Arduino code to read sensor data, control the servo motor, and process the data for object detection.

STEP2:

- Create a user-friendly interface for monitoring the system and controlling its parameters.

STEP3:

- Implement algorithms for analyzing sensor data and detecting objects within the system's vicinity.

4.3 INTEGRATION TECHNIQUES:

STEP1:

- Connect the hardware components (ultrasonic sensor, servo motor, Arduino board) according to the system architecture.

STEP2:

- Integrate the software components (Arduino code, user interface) to ensure seamless operation of the system.

STEP3:

- Conduct thorough testing to verify the integration of hardware and software components and ensure the system functions as expected.

4.4 CODE SNIPPETS:

- **Ardunio Code :**

```
#include <Servo.h>
const int greenLight = 7;
const int trigPin = 9;
const int echoPin = 10;
const int buzzer = 11;
const int redLight = 12;
// defining time and distance
long duration;
int distance;
float DistanceSec;
Servo myServo; // Object servo
void setup() {
  pinMode(trigPin, OUTPUT); // trigPin as an Output
  pinMode(echoPin, INPUT); // echoPin as an Input
  pinMode(buzzer,OUTPUT);
  pinMode(redLight, OUTPUT);
  pinMode(greenLight, OUTPUT);
  Serial.begin(9600);
  myServo.attach(8); // Pin Connected To Servo
  DistanceSec = 20;
```

```
}
void loop() {
// rotating servo i++ depicts increment of one degree

  for(int i=1;i<=180;i++){
    myServo.write(i);
    delay(10);
    distance = calculateDistance();
    Serial.print(i);
    Serial.print(",");
    Serial.print(distance);
    Serial.print(".");
    if(distance<=DistanceSec){
      if(distance<=DistanceSec/2){
        tone(buzzer,10); //send 1khz sound
        digitalWrite(redLight,HIGH);
        digitalWrite(greenLight,LOW);
        delay(30);
        noTone(buzzer);
        digitalWrite(redLight,LOW);
//        digitalWrite(greenLight,HIGH);
      }else{
        digitalWrite(buzzer, HIGH);
        digitalWrite(redLight, HIGH);
        digitalWrite(greenLight,LOW);
        delay(30);
        digitalWrite(buzzer, LOW);
      }
    }else{
      digitalWrite(buzzer, LOW);
      digitalWrite(redLight,LOW);
      digitalWrite(greenLight,HIGH);
    }
  }
// Repeats the previous lines from 165 to 15 degrees
  for(int i=180;i>1;i--){
    myServo.write(i);
    delay(30);
    distance = calculateDistance();
    Serial.print(i);
    Serial.print(",");
    Serial.print(distance);
    Serial.print(".");
    if(distance<=DistanceSec){
      if(distance<=DistanceSec/2){
        tone(buzzer,10); //send 1khz sound
        digitalWrite(redLight,HIGH);
        digitalWrite(greenLight,LOW);
        delay(30);
        noTone(buzzer);
        digitalWrite(redLight,LOW);
//        digitalWrite(greenLight,HIGH);
      }else{
        digitalWrite(buzzer, HIGH);
        digitalWrite(redLight, HIGH);
        digitalWrite(greenLight,LOW);
        delay(30);
        digitalWrite(buzzer, LOW);
      }
    }else{
      digitalWrite(buzzer, LOW);
      digitalWrite(redLight,LOW);
      digitalWrite(greenLight,HIGH);
    }
  }
}
int calculateDistance(){
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
// Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance= duration*0.034/2;
  return distance;
}
```

- **Processing Code (Output Display Code):**

```
import processing.serial.*; // imports library for serial communication
import java.awt.event.KeyEvent; // imports library for reading the data from the serial port
import java.io.IOException;
Serial myPort; // defines Object Serial
// defubes variables
String angle="";
String distance="";
String data="";
String noObject;
float pixsDistance;
int iAngle, iDistance;
int index1=0;
int index2=0;
PFont orcFont;
void setup() {

 size (1200, 700); // ***CHANGE THIS TO YOUR SCREEN RESOLUTION***
 smooth();
 myPort = new Serial(this,"COM4", 9600); // starts the serial communication
 myPort.bufferUntil('.'); // reads the data from the serial port up to the character '.'. So
actually it reads this: angle,distance.
}
void draw() {

  fill(98,245,31);
  // simulating motion blur and slow fade of the moving line
  noStroke();
  fill(0,4);
  rect(0, 0, width, height-height*0.065);

  fill(98,245,31); // green color
  // calls the functions for drawing the radar
  drawRadar();
  drawLine();
  drawObject();
  drawText();
}
void serialEvent (Serial myPort) { // starts reading data from the Serial Port
  // reads the data from the Serial Port up to the character '.' and puts it into the String
variable "data".
  data = myPort.readStringUntil('.');
  data = data.substring(0,data.length()-1);

  index1 = data.indexOf(","); // find the character ',' and puts it into the variable
"index1"
  angle= data.substring(0, index1); // read the data from position "0" to position of the
variable index1 or thats the value of the angle the Arduino Board sent into the Serial Port
  distance= data.substring(index1+1, data.length()); // read the data from position "index1"
to the end of the data pr thats the value of the distance

  // converts the String variables into Integer
  iAngle = int(angle);
  iDistance = int(distance);
}
void drawRadar() {
  pushMatrix();
  translate(width/2,height-height*0.074); // moves the starting coordinats to new location
  noFill();
  strokeWeight(2);
  stroke(98,245,31);
  // draws the arc lines
  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);
  // draws the angle lines
  line(-width/2,0,width/2,0);
  line(0,0,(-width/2)*cos(radians(30)),(-width/2)*sin(radians(30)));
  line(0,0,(-width/2)*cos(radians(60)),(-width/2)*sin(radians(60)));
  line(0,0,(-width/2)*cos(radians(90)),(-width/2)*sin(radians(90)));
  line(0,0,(-width/2)*cos(radians(120)),(-width/2)*sin(radians(120)));
  line(0,0,(-width/2)*cos(radians(150)),(-width/2)*sin(radians(150)));
  line((-width/2)*cos(radians(30)),0,width/2,0);
  popMatrix();
}
void drawObject() {
  pushMatrix();
  translate(width/2,height-height*0.074); // moves the starting coordinats to new location
  strokeWeight(9);
  stroke(255,10,10); // red color
  pixsDistance = iDistance*((height-height*0.1666)*0.025); // covers the distance from the 
sensor from cm to pixels
  // limiting the range to 40 cms
  if(iDistance<40){
    // draws the object according to the angle and the distance
  line(pixsDistance*cos(radians(iAngle)),-pixsDistance*sin(radians(iAngle)),(width-
width*0.505)*cos(radians(iAngle)),-(width-width*0.505)*sin(radians(iAngle)));
  }
  popMatrix();
}
void drawLine() {
  pushMatrix();
  strokeWeight(9);
  stroke(30,250,60);
  translate(width/2,height-height*0.074); // moves the starting coordinats to new location
  line(0,0,(height-height*0.12)*cos(radians(iAngle)),-(height-
height*0.12)*sin(radians(iAngle))); // draws the line according to the angle
  popMatrix();
}
void drawText() { // draws the texts on the screen

  pushMatrix();
  if(iDistance>40) {
  noObject = "Out of Range";
  }
  else {
  noObject = "In Range";
  }
  fill(0,0,0);
  noStroke();
  rect(0, height-height*0.0648, width, height);
  fill(98,245,31);
  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);
  text("Radar System ", width-width*0.875, height-height*0.0277);
  text("Angle: " + iAngle +"  ", width-width*0.58, height-height*0.0277);
  text("Distance: ", width-width*0.30, height-height*0.0277);
  if(iDistance<40) {
  text("  " + iDistance +" cm", width-width*0.175, 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();
}
```

CHAPTER : 5

TESTING AND VALIDATION

5.1 TEST PLANS :

❖ OBJECTIVE:

- ➢ To verify the functionality, accuracy, and reliability of the Ultrasonic Radar System.

❖ SCOPE:

- ➢ The test plan covers testing of hardware components, software functionality, and system integration.

❖ TEST ENVIROMENT:

- ➢ **Hardware:** Ultrasonic sensor, servo motor, Arduino Uno, power supply.
- ➢ **Software:** Arduino IDE, monitoring interface (web or mobile).

5.2 TEST CASES:

❖ TEST CASE 1: Hardware Setup and Configuration

- ➢ Verify proper mounting and alignment of the ultrasonic sensor and servo motor.
- ➢ Ensure correct wiring and connection of all hardware components.
- ➢ Check the power supply for stability and adequacy.

❖ TEST CASE 2: Software Functionality Testing

- ➢ Test the Arduino code for reading sensor data and controlling the servo motor.
- ➢ Verify the user interface for monitoring and controlling the system.
- ➢ Test the algorithm for object detection and distance calculation.

- **TEST CASE 3: System Integration Testing**
  - Test the integration of hardware and software components.
  - Verify real-time monitoring and control functionality.
  - Test the system's response to different scenarios (e.g., object detection, servo motor movement).

- **TEST CASE 4: Performance and Accuracy Testing**
  - Measure the system's detection range and accuracy.
  - Test the system's response time to object detection and servo motor movement.
  - Verify the system's performance under varying environmental conditions (e.g., different object sizes, distances, and angles).

5.3 PERFORMANCE METRICS:

- **Detection Range :**
  - Measure the maximum distance at which the system can detect objects accurately.

- **Accuracy:**
  - Evaluate the system's ability to accurately detect and track objects within its detection range.

- **Response Time:**
  - Measure the time taken by the system to detect an object and respond with servo motor movement.

- **Scanning Speed:**
  - Evaluate the speed at which the servo motor scans the environment for object detection.

- **Power Consumption:**
  - Measure the power consumption of the system to ensure it operates efficiently.

- **Reliability:**
  - Assess the system's reliability in detecting objects under different environmental conditions (e.g., lighting, noise).

5.4 VALIDATION PROCEDURES:

- **Detection Range Test:**
  - Place objects at varying distances from the sensor and measure the detection range.
  - Verify that the system can accurately detect objects within the specified range.

- **Accuracy Test:**
  - Place objects of different sizes and shapes at known distances and angles.
  - Verify that the system can accurately detect and track the objects.

- **Response Time Test:**
  - Measure the time taken by the system to detect an object and move the servo motor.
  - Ensure that the response time meets the project requirements.

- **Scanning Speed Test:**
  - Measure the speed at which the servo motor scans the environment.
  - Verify that the scanning speed is sufficient for real-time object detection.

- **Power Consumption Test:**
  - Measure the power consumption of the system during operation.
  - Ensure that the power consumption is within acceptable limits.

- **Reliability Test:**
  - Test the system's performance under different environmental conditions (e.g., light, noise).
  - Verify that the system can reliably detect objects in various scenarios.

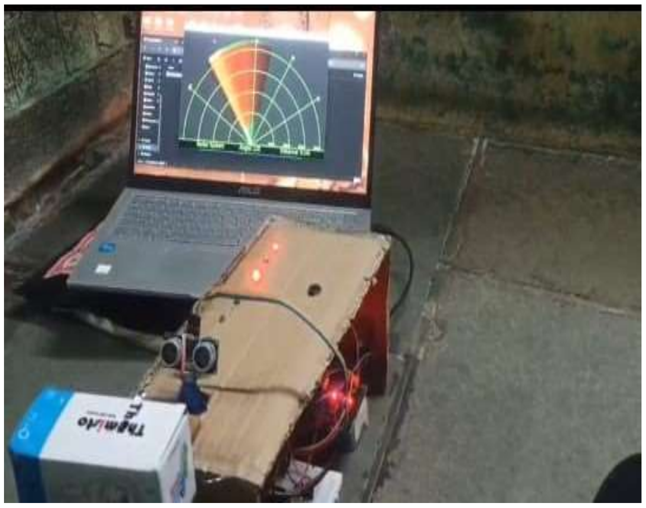
CHAPTER : 6
RESULT AND ANALYSIS

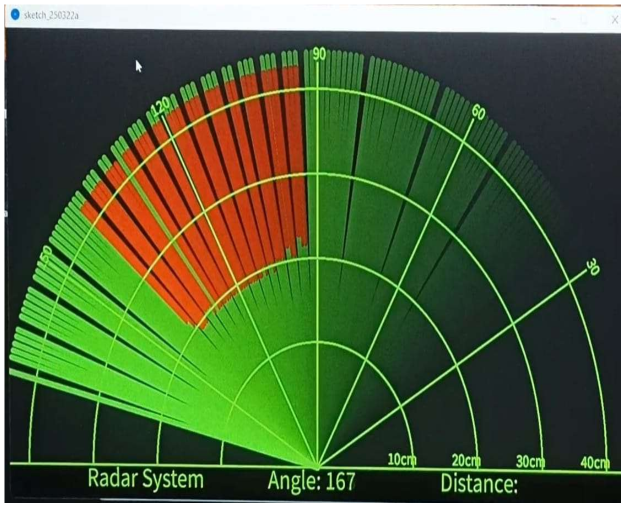
6.1 Summary of Findings:

The Ultrasonic Radar System IoT project aims to develop a sophisticated sensing system that utilizes ultrasonic technology for object detection. This innovative project combines ultrasonic sensing technology with radar principles to create a versatile detection system capable of accurately identifying objects and their distances. By emitting ultrasonic waves and analyzing their reflections, the radar system can effectively map the surrounding environment and detect any obstacles or targets within its range.

The Ultrasonic Radar System IoT project has various applications in security surveillance, industrial automation, and smart transportation. It provides a reliable and efficient solution for object detection and monitoring, contributing to enhanced safety, efficiency, and productivity. Overall, this project represents an innovative approach to utilizing ultrasonic technology for advanced sensing and detection systems.

6.2 OUTPUT SCREENS:





6.3 RESULT EVALUTATION:

Results Achieved:

1. **Object Detection and Distance Measurement**
   - ➢ The ultrasonic sensor accurately detected objects within a range of approximately 2 cm to 400 cm.
   - ➢ The system consistently measured distance based on time-of-flight of ultrasonic waves and displayed the data via the serial monitor or visual interface.
2. **Servo-Based Radar Scanning**
   - ➢ The servo motor successfully rotated the ultrasonic sensor in a defined arc (e.g., 0° to 180°), enabling a sweeping motion to cover a wider area.
   - ➢ Detected object positions were mapped to specific angles, simulating a radar sweep.
3. **Visual and Audible Alerts**
   - ➢ The buzzer and LED responded correctly when objects were detected within the defined threshold (e.g., less than 30 cm).
   - ➢ This provided immediate user feedback about the presence of an obstacle or target.
4. **Data Visualization**
   - ➢ A graphical interface was created using **Processing**, which plotted detected objects on a radar-like display.
   - ➢ The interface helped in visualizing object positions relative to the sensor's sweep angle and distance.
5. **IoT Integration** *(if implemented)*
   - ➢ The system was able to send data to a cloud platform (e.g., Blynk, ThingSpeak) or mobile app.
   - ➢ Users could monitor radar activity remotely and receive alerts in real time.

Evaluation and Performance Analysis:

- **Accuracy**:
  Object detection was reliable within the specified range, with minor fluctuations due to environmental noise or object shape/material.
- **Response**
  The system operated in near real-time, with minimal delay between object detection and alert output.
- **Coverage**:
  The scanning mechanism covered a semi-circular area (180°), providing adequate environmental awareness for basic applications.
- **Scalability**:
  The modular design allows for expansion, such as integrating additional sensors, increasing scanning resolution, or applying machine learning for pattern detection.

6.4 ADVANTAGES OF PROPOSED SYSTEM:

1) Low-Cost and Affordable

- Uses inexpensive components like Arduino Uno and ultrasonic sensors, making it ideal for students and prototype development.

2) Easy to Build and Program

- Simple hardware connections and Arduino-based code make the system easy to assemble, understand, and customize.

3) Real-Time Object Detection

- Accurately detects objects and measures distance in real time, providing instant visual (LED) and audible (buzzer) feedback.

4) Radar-Like Scanning

- Servo motor allows the ultrasonic sensor to scan a wide area, simulating radar functionality effectively.

5) IoT Integration (Optional)

- Can send data to a cloud platform or mobile app for remote monitoring and control, enhancing flexibility.

6) Portable and Compact Design

- Small footprint allows easy deployment in various environments like homes, labs, or classrooms.

7) Customizable Alert System

- Alerts (via buzzer and LED) can be easily configured for different distances or detection angles.

8) Scalable for Future Development

- The system can be expanded with machine learning, wireless modules, or camera integration for more advanced applications.

6.5 Limitation of Project:

- **Detection Range :**
  - The ultrasonic sensor's detection range may be limited, affecting the system's ability to detect objects at longer distances.
- **Environmental Factors:**
  - The system's performance may be affected by environmental factors such as noise, humidity, and temperature variations.
- **Obstacle Size and Material:**
  - The system may have limitations in detecting small or non-rigid objects, as well as objects made of certain materials that may not reflect ultrasonic waves effectively.
- **Interference :**
  - External interference from other ultrasonic devices or environmental factors may impact the system's accuracy and reliability.
- **Power Consumption :**
  - Continuous operation of the system may require a significant amount of power, limiting its battery life in portable applications.
- **Complexity of Implementation :**
  - Integrating hardware components and software applications, as well as ensuring proper calibration and alignment, may be complex and require technical expertise.
- **Cost:**
  - The cost of components, especially for high-quality sensors and connectivity modules, may be a limitation for some applications.
- **Data Processing and Analysis:**
  - Real-time data processing and analysis may pose challenges, especially in applications requiring quick decision-making based on sensor data.

6.6 FUTURE WORK:

- **Enhanced Detection Range :**
  - Research and implement techniques to extend the system's detection range, enabling detection of objects at greater distances.
- **Advanced Object Recognition:**
  - Integrate machine learning algorithms for improved object recognition and classification, allowing the system to distinguish between different types of objects.
- **Energy Efficiency:**
  - Develop energy-efficient algorithms and hardware designs to reduce the system's power consumption, particularly important for portable applications.
- **Integration with AI Assistants :**
  - Explore integration with AI assistants or voice-controlled devices for enhanced user interaction and control of the radar system.
- **Cloud Connectivity:**
  - Implement cloud connectivity for storing and analysing sensor data, enabling advanced analytics and remote access to the system.

CHAPTER : 7

REFERENCES

- https://www.youtube.com/
- https://www.google.com/
- https://www.tinkercad.com/
- https://www.arduino.cc/en/software
- https://processing.org/reference/

THANK YOU…!

We know we might have made some mistakes knowingly or unknowingly. But all the suggestions regarding to this system are always welcome