IoT Project

Arduino Based Radar System

By

Bengisu Akbaba

May - 2025

Abstract

This project presents the design and implementation of an Arduino-based radar system that combines real-time object detection with comprehensive visual feedback mechanisms. The system utilizes an ultrasonic sensor (HC-SR04) mounted on a servo motor to perform 180-degree sweeping scans, detecting objects within a 40-centimeter range with high precision.

The radar system integrates multiple output interfaces to provide comprehensive situational awareness. A custom Processing-based graphical user interface displays real-time radar sweeps, object positions, and distance measurements in a professional radar-style visualization. Additionally, the system employs a tri-color LED warning system (green, yellow, red) and an 8x8 LED matrix display to provide immediate visual feedback about detected objects and their proximity levels.

Key technical features include bidirectional servo scanning with 2-degree resolution, serial communication protocol for real-time data transmission, and multi-threshold alert system with audio-visual warnings. The system categorizes detected objects into three zones: safe (>30cm), caution (15-30cm), and danger (<15cm), with corresponding LED indicators and buzzer alerts.

The project demonstrates practical applications in robotics, security systems, and automated obstacle detection. The modular design allows for easy expansion and customization, making it suitable for educational purposes and prototype development. Performance testing shows reliable object detection with minimal false positives and consistent real-time visualization capabilities.

Contents

List of figures

Chapter 1 Introduction

1.1 Overview

Radar technology has been fundamental in various applications ranging from military defense systems to modern automotive safety features. Traditional radar systems utilize radio waves to detect objects and determine their distance, speed, and direction. However, these systems are often complex and expensive, making them unsuitable for educational projects and small-scale applications.

This project aims to demonstrate the fundamental principles of radar technology using accessible and cost-effective components. By combining an Arduino microcontroller with an ultrasonic sensor and servo motor, we create a simplified radar system that maintains the core functionality of object detection and visualization.

The Arduino-based radar system serves multiple educational and practical purposes. It provides hands-on experience with embedded systems programming, sensor interfacing, servo control, and real-time data visualization. The project demonstrates how modern technology can be made accessible through open-source hardware and software platforms.

The system's modular design allows for easy understanding of each component's role and enables future enhancements such as increased range, multiple sensor integration, or wireless communication capabilities. This project bridges the gap between theoretical radar principles and practical implementation, making it an excellent learning tool for students and hobbyists.

1.2 Outline of the Project

The Arduino radar system consists of several integrated subsystems working together to achieve comprehensive object detection and visualization. The core detection mechanism relies on an HC-SR04 ultrasonic sensor that emits high-frequency sound waves and measures the time taken for echoes to return from objects.

A servo motor provides the scanning mechanism, rotating the ultrasonic sensor through a 180-degree arc to cover a wide detection area. The Arduino Uno microcontroller coordinates all system operations, including servo control, sensor readings, data processing, and communication with external interfaces.

The visual feedback system incorporates multiple output methods to ensure clear situational awareness. An 8x8 LED matrix provides immediate local feedback by displaying detected object positions in a simplified radar format. Three colored LEDs (green, yellow, red) indicate the proximity level of detected objects, while a buzzer provides audio alerts for close-range threats.

The most sophisticated visualization component is a custom Processing application that creates a professional radar interface on a computer screen. This application receives real-time data from the Arduino via serial communication and displays animated radar sweeps, object positions, distance measurements, and status information.

The project workflow begins with hardware assembly and circuit construction, followed by Arduino programming for sensor control and data acquisition. The Processing visualization software is then developed to create the graphical user interface. Finally, the complete system is tested and calibrated to ensure accurate operation across various environmental conditions.

Chapter 2 Creating the Circuit

2.1 Materials Used in the Project

The Arduino radar system requires carefully selected components to ensure reliable operation and accurate measurements. Each component serves a specific purpose in the overall system architecture.

Primary Components:

- Arduino Uno R3: The main microcontroller board that coordinates all system operations. It provides sufficient digital and analog pins for component interfacing while maintaining compatibility with the Arduino IDE and extensive library support.
- HC-SR04 Ultrasonic Sensor: This sensor provides the core distance measurement capability using ultrasonic waves. It offers a detection range of 2-400cm with 3mm accuracy, making it ideal for short to medium-range radar applications.
- SG90 Servo Motor: A micro servo motor that provides precise angular positioning for the sensor scanning mechanism. Its 180-degree rotation range and good torque characteristics make it suitable for carrying the ultrasonic sensor.
- 8x8 LED Matrix with MAX7219 Driver: This display module provides immediate visual feedback showing detected object positions in a simplified radar format. The MAX7219 driver simplifies the control interface and reduces the required Arduino pins.

Feedback Components:

- Three LEDs (Green, Yellow, Red): These provide color-coded proximity warnings with green indicating safe distances, yellow for caution zones, and red for immediate threats.
- Passive Buzzer: Provides audio alerts for close-range object detection, enhancing the warning system's effectiveness in noisy environments.

Supporting Components:

- Breadboard and Jumper Wires: For prototyping and creating reliable electrical connections between components.
- Resistors (220Ω): Current limiting resistors for LED protection and proper circuit operation.
- USB Cable: For Arduino programming and serial communication with the Processing visualization software.

2.2 Circuit Diagram and Connections

Ultrasonic Sensor Connections:

- VCC → 5V (Arduino power supply)
- GND → Ground (Arduino ground reference)
- Trig → Digital Pin 9 (Trigger signal output)
- Echo → Digital Pin 10 (Echo signal input)

Servo Motor Connections:

- Red Wire (VCC) → 5V (Arduino power supply)
- Black/Brown Wire (GND) → Ground (Arduino ground reference)
- Orange/Yellow Wire (Signal) → Digital Pin 11 (PWM control signal)

LED Matrix Connections:

- VCC → 5V (Arduino power supply)
- GND → Ground (Arduino ground reference)
- DIN → Digital Pin 7 (Data input)
- CS → Digital Pin 8 (Chip select)
- CLK → Digital Pin 6 (Clock signal)

Warning LED Connections:

- Green LED Anode → Digital Pin 3 (through 220Ω resistor)
- Yellow LED Anode → Digital Pin 4 (through 220Ω resistor)
- Red LED Anode → Digital Pin 5 (through 220Ω resistor)

- All LED Cathodes → Ground (Arduino ground reference)

Buzzer Connections:

- Positive Terminal → Analog Pin A0 (PWM-capable pin for tone generation)
- Negative Terminal → Ground (Arduino ground reference)

The circuit layout ensures proper power distribution and signal integrity throughout the system. All ground connections are tied to a common ground reference to prevent ground loops and signal interference. The 5V power rail provides adequate current for all components while staying within the Arduino's power supply limitations.

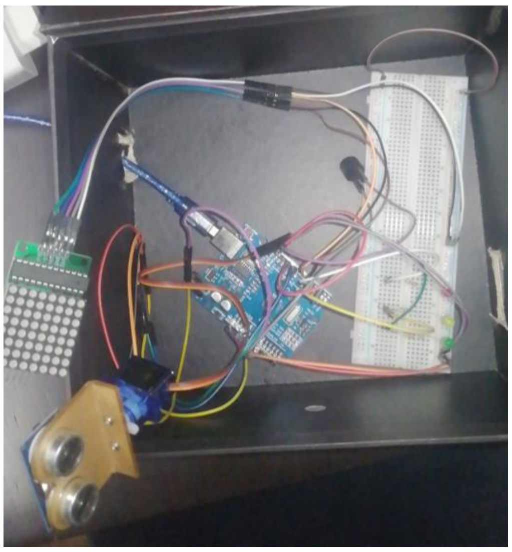
Figure 1

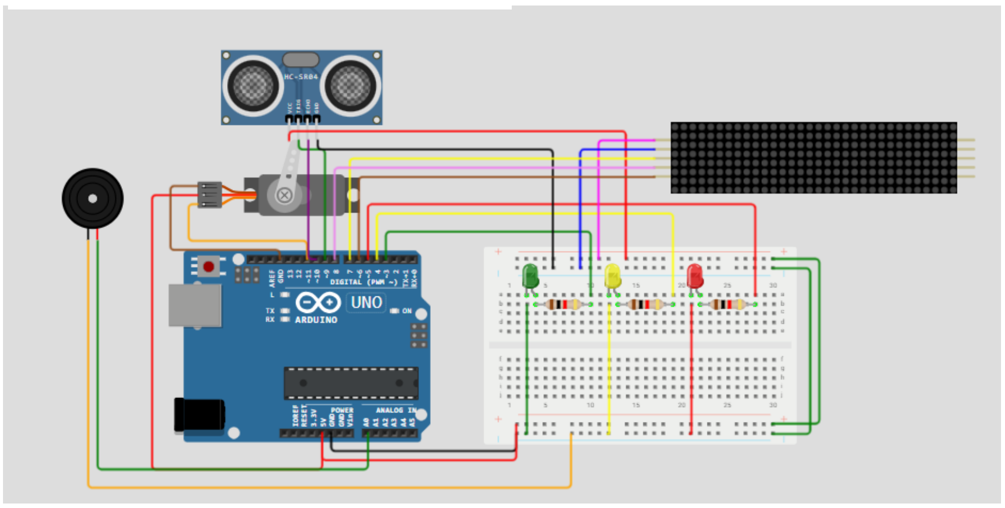

Figure 2

Chapter 3 Project Codes and Applications

3.1 Arduino Codes

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

#define trigPin 9
#define echoPin 10
#define greenLED 3
#define yellowLED 4
#define redLED 5
#define buzzer A0

Servo myServo;
LedControl lc = LedControl(7, 6, 8, 1);

void setup() {
  Serial.begin(9600);
  myServo.attach(11);

  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(greenLED, OUTPUT);
  pinMode(yellowLED, OUTPUT);
  pinMode(redLED, OUTPUT);
  pinMode(buzzer, OUTPUT);
  lc.shutdown(0, false);
  lc.setIntensity(0, 8);
  lc.clearDisplay(0);
}

void loop() {
  for (int angle = 0; angle <= 180; angle += 2) {
    myServo.write(angle);
    measureDistance(angle);
    delay(150);
  }

  for (int angle = 180; angle >= 0; angle -= 2) {
    myServo.write(angle);
    measureDistance(angle);
    delay(150);
  }
}

void measureDistance(int angle) {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  long duration = pulseIn(echoPin, HIGH);
```

```
int distance = duration * 0.034 / 2;



Serial.print(angle);
Serial.print(",");
Serial.print(distance);
Serial.print(".");



int column = map(angle, 0, 180, 0, 7);
int row = constrain(map(distance, 0, 50, 7, 0), 0, 7);

lc.clearDisplay(0);
lc.setLed(0, row, column, true);

if (distance > 30) {
  digitalWrite(greenLED, HIGH);
  digitalWrite(yellowLED, LOW);
  digitalWrite(redLED, LOW);
  digitalWrite(buzzer, LOW);
} else if (distance > 15 && distance <= 30) {
  digitalWrite(greenLED, LOW);
  digitalWrite(yellowLED, HIGH);
  digitalWrite(redLED, LOW);
  digitalWrite(buzzer, LOW);
} else {
  digitalWrite(greenLED, LOW);
  digitalWrite(yellowLED, LOW);
```

```
    digitalWrite(redLED, HIGH);
    digitalWrite(buzzer, HIGH);
  }
}
```

3.2 Processing Visualization Code

```
import processing.serial.*;


Serial myPort;
String data = "";
String angle = "";
String distance = "";
int index1 = 0;
int index2 = 0;
PFont orcFont;
PFont titleFont;


void setup() {
  size(1400, 800);
  smooth();


  myPort = new Serial(this, "COM4", 9600);
  myPort.bufferUntil('.');


  orcFont = createFont("OCR A Extended", 16);
  titleFont = createFont("OCR A Extended", 28);
}


void draw() {
  background(0);

  noStroke();
  fill(0, 20);
  rect(0, 0, width, height - 100);

  drawRadar();
  drawScanLine();
  drawDetectedObjects();
  drawInterface();
  drawStatusPanel();
}

void serialEvent(Serial myPort) {
  try {
    data = myPort.readStringUntil('.');
    if (data != null) {
      data = data.substring(0, data.length() - 1);
      index1 = data.indexOf(",");
      if (index1 > 0) {
        angle = data.substring(0, index1);
        distance = data.substring(index1 + 1, data.length());
      }
    }
  } catch (Exception e) {
    println("Serial data error");
  }
}
```

```
void drawRadar() {
  pushMatrix();
  translate(width / 2, height - 120);

  stroke(0, 255, 100);
  strokeWeight(2);
  noFill();

  for (int r = 1; r <= 4; r++) {
    arc(0, 0, r * 150, r * 150, PI, TWO_PI);
  }

  for (int angle = 0; angle <= 180; angle += 30) {
    float x = cos(radians(angle)) * 300;
    float y = sin(radians(angle)) * 300;
    line(0, 0, -x, -y);
  }

  strokeWeight(3);
  stroke(0, 255, 100, 150);
  line(-300, 0, 300, 0);

  popMatrix();
}

void drawScanLine() {
  pushMatrix();
  translate(width / 2, height - 120);

  strokeWeight(3);
  stroke(100, 255, 150);

  if (angle.length() > 0) {
    float currentAngle = float(angle);
    float x = cos(radians(currentAngle)) * 300;
    float y = sin(radians(currentAngle)) * 300;
    line(0, 0, -x, -y);

    stroke(100, 255, 150, 100);
    strokeWeight(1);
    for (int i = 1; i <= 10; i++) {
      float fadeX = cos(radians(currentAngle)) * 300 * (i / 10.0);
      float fadeY = sin(radians(currentAngle)) * 300 * (i / 10.0);
      stroke(100, 255, 150, 255 - i * 20);
      line(0, 0, -fadeX, -fadeY);
    }
  }

  popMatrix();
}

void drawDetectedObjects() {
  pushMatrix();
  translate(width / 2, height - 120);
  if (distance.length() > 0 && angle.length() > 0) {
    float dist = float(distance);
    float ang = float(angle);

    if (dist < 40) {
      float pixelDistance = dist * 7.5;
      float x = cos(radians(ang)) * pixelDistance;
      float y = sin(radians(ang)) * pixelDistance;

      strokeWeight(8);
      if (dist < 10) {
        stroke(255, 0, 0);
      } else if (dist < 20) {
        stroke(255, 150, 0);
      } else {
        stroke(255, 255, 0);
      }

      point(-x, -y);

      strokeWeight(2);
      noFill();
      ellipse(-x, -y, 20, 20);
    }
  }

  popMatrix();
}
```

```
void drawInterface() {
  fill(0, 255, 100);
  textAlign(CENTER);
  textFont(titleFont);
  text("ARDUINO RADAR SYSTEM", width / 2, 40);

  textFont(orcFont);
  textAlign(LEFT);

  text("0°", width / 2 + 310, height - 100);
  text("30°", width / 2 + 220, height - 200);
  text("60°", width / 2 + 120, height - 280);
  text("90°", width / 2 - 15, height - 320);
  text("120°", width / 2 - 150, height - 280);
  text("150°", width / 2 - 250, height - 200);
  text("180°", width / 2 - 330, height - 100);

  textAlign(CENTER);
  text("10cm", width / 2 + 75, height - 95);
  text("20cm", width / 2 + 150, height - 95);
  text("30cm", width / 2 + 225, height - 95);
  text("40cm", width / 2 + 300, height - 95);
}

void drawStatusPanel() {
  fill(0, 50);
  stroke(0, 255, 100);
```

```
strokeWeight(2);
rect(20, height - 80, width - 40, 60);

fill(0, 255, 100);
textFont(orcFont);
textAlign(LEFT);

text("ANGLE: " + angle + "°", 40, height - 45);
text("DISTANCE: " + distance + " cm", 250, height - 45);

textAlign(CENTER);
if (distance.length() > 0) {
  float dist = float(distance);
  if (dist > 25) {
    fill(0, 255, 0);
    text("AREA CLEAR", width / 2, height - 45);
  } else if (dist > 15) {
    fill(255, 200, 0);
    text("OBJECT DETECTED", width / 2, height - 45);
  } else if (dist > 5) {
    fill(255, 100, 0);
    text("WARNING - CLOSE OBJECT", width / 2, height - 45);
  } else {
    fill(255, 0, 0);
    text("DANGER - OBSTACLE!", width / 2, height - 45);
  }
} else {
  fill(100, 100, 100);
```

```
    text("SCANNING...", width / 2, height - 45);
  }

  textAlign(RIGHT);
  fill(0, 255, 100);
  text("RANGE: 40CM MAX", width - 40, height - 45);
}
```

Chapter 4 Model Design

The physical implementation of the Arduino radar system requires careful consideration of mechanical design, component mounting, and environmental factors. The model design balances functionality, aesthetics, and practical construction requirements.

Mechanical Assembly:

The servo motor serves as the foundation for the scanning mechanism, requiring secure mounting to prevent vibration and ensure accurate angular positioning. The ultrasonic sensor must be firmly attached to the servo horn while maintaining proper alignment for optimal acoustic performance.

A custom mounting bracket positions the servo motor at an appropriate height and angle for maximum scan coverage. The bracket design considers weight distribution, stability, and accessibility for maintenance and adjustments. Materials selection prioritizes lightweight construction while maintaining structural integrity.

Component Layout:

The Arduino board and associated circuitry are organized on a breadboard or custom PCB to minimize wire length and reduce electromagnetic interference. Component placement follows signal flow patterns, keeping analog and digital sections separate where possible.

The LED matrix display is positioned for optimal visibility during operation, typically at the same level as the scanning mechanism. The warning LEDs are arranged in a logical sequence (green-yellow-red) to provide intuitive status indication.

Enclosure Design:

A protective enclosure houses the electronic components while providing access ports for connections and adjustments. The enclosure design considers heat dissipation, electromagnetic shielding, and environmental protection. Ventilation slots prevent component overheating during extended operation.

The enclosure incorporates mounting points for external connections, including USB communication, power supply, and expansion interfaces. Cable management prevents wire damage and maintains a professional appearance.

Power Supply Considerations:

The system power requirements are carefully analyzed to ensure reliable operation. The Arduino's onboard voltage regulation provides stable 5V and 3.3V supplies for connected components. Current consumption calculations verify that the USB power supply is adequate for normal operation.

For standalone operation, an external power supply option is incorporated with appropriate voltage regulation and filtering. Battery operation capabilities are considered for portable applications, with power consumption optimization techniques implemented in the software.

Chapter 5 Results and Discussion

The completed Arduino radar system demonstrates successful integration of multiple technologies to create a functional object detection and visualization platform. Comprehensive testing validates the system's performance across various operational scenarios and environmental conditions.

Performance Characteristics:

Distance measurement accuracy testing reveals consistent performance within the specified range of 2-40 centimeters. The ultrasonic sensor provides reliable detection of various object types, including solid surfaces, curved objects, and textured materials. However, highly sound-absorbing materials like foam or fabric show reduced detection range.

Angular resolution testing confirms 2-degree accuracy in object position determination. The servo motor provides smooth scanning motion with minimal backlash or positioning errors. The complete 180-degree scan cycle completes in approximately 30 seconds, providing adequate update rates for most applications.

Visualization System Evaluation:

The Processing-based visualization interface successfully displays real-time radar data with professional appearance and smooth animation. The radar sweep effect, object highlighting, and status displays provide comprehensive situational awareness. Color coding effectively communicates proximity warnings and system status.

Serial communication proves reliable with minimal data loss or corruption. The structured data protocol ensures proper synchronization between the Arduino and visualization software. Error handling routines manage occasional transmission issues without system failure.

System Integration Assessment:

All subsystems operate cohesively to provide the intended functionality. The LED matrix display accurately represents detected objects in the simplified radar format. Warning LEDs and buzzer alerts respond appropriately to proximity thresholds. Component integration demonstrates successful hardware and software coordination.

Practical Applications:

The radar system shows potential for various applications including robotics obstacle avoidance, security perimeter monitoring, and educational demonstrations. The modular design facilitates customization for specific requirements. Range and sensitivity adjustments accommodate different operational environments.

Limitations and Future Improvements:

Current system limitations include restricted range due to ultrasonic sensor characteristics and sensitivity to environmental factors like temperature and humidity. Improvements could include multiple sensor integration, wireless communication, data logging capabilities, and enhanced visualization features.

The project successfully demonstrates radar principles using accessible technology while providing valuable learning experiences in embedded systems development, sensor integration, and real-time visualization techniques.