7/14/2025

RADAR-BASED OBJECT DETECTION SYSTEM USING ARDUINO

MINI PROJECT

**Submitted by:**

**HISANA S**

**1. INTRODUCTION**

Radar systems are widely used for detecting objects and measuring distances in applications such as autonomous vehicles, aviation, military surveillance, and industrial automation. Traditional radar systems use radio waves, but in embedded system prototyping and academic projects, ultrasonic sensors can be used to simulate similar functionality.

This mini project aims to demonstrate the basic working principle of radar by using an ultrasonic sensor (HC-SR04) mounted on a servo motor, interfaced with an Arduino Uno. The servo motor rotates the sensor to scan the surroundings, and the Arduino collects distance data by measuring the time taken for the ultrasonic signal to bounce back from an object. The results are displayed on a serial monitor or visualized using software like Processing IDE.

Through this project, students can understand key embedded concepts such as sensor interfacing, servo control, timing, and real-time data acquisition. This simplified radar system serves as a practical and educational tool for beginners in electronics and communication engineering.

**2. OBJECTIVES**

The main objectives of the Radar-Based Object Detection System using Arduino are as follows:

• To design a radar prototype capable of detecting nearby objects and measuring their distance.

• To simulate the working of a basic radar system using affordable and easily available components like Arduino, ultrasonic sensor, and servo motor.

• To gain practical experience in sensor interfacing, embedded programming, and real-time data collection.

• To visualize the scanning process and object detection through serial output or graphical interface using Processing IDE.

• To understand and apply the concepts of pulse timing, distance calculation, and motor control in embedded systems.

**3. COMPONENTS USED**

The following components were used to implement the radar-based object detection system:

• Arduino UNO – Acts as the microcontroller that controls the servo motor, triggers the ultrasonic sensor, and processes the returned signal.

• HC-SR04 Ultrasonic Sensor – Used to measure the distance to an object by sending and receiving ultrasonic waves.

• Servo Motor – Rotates the ultrasonic sensor in a sweeping motion from 0° to 180° to simulate radar scanning.

• Breadboard and Jumper Wires – Used for prototyping and connecting components without soldering.

• USB Cable – For uploading code from the computer to the Arduino UNO and for power supply.

• Computer with Arduino IDE and Processing IDE – For programming the Arduino and visualizing radar output in real-time.

**4. CIRCUIT DIAGRAM**

The circuit for the radar-based object detection system includes connections between the Arduino UNO, ultrasonic sensor (HC-SR04), and the SG90 servo motor.

🛠 Connections:

➤ Ultrasonic Sensor (HC-SR04):

• VCC → 5V on Arduino

• GND → GND on Arduino

• Trig → Digital Pin 10

• Echo → Digital Pin 11

➤ Servo Motor (SG90):

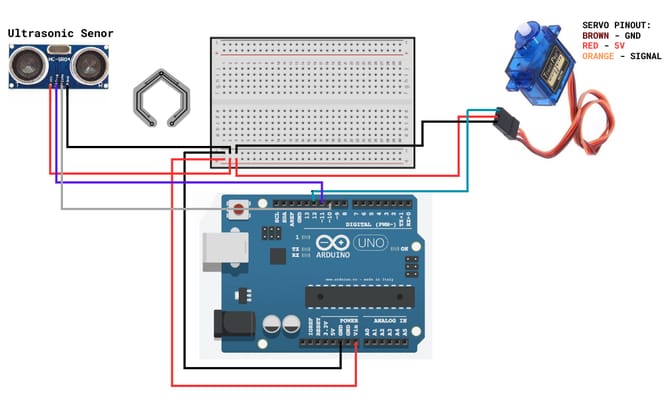
• VCC → 5V on Arduino

• GND → GND on Arduino

• Signal → Digital Pin 9

➤ Arduino UNO:

• Connected to PC via USB for uploading code and power



The ultrasonic sensor is mounted on top of the servo motor, allowing the system to scan the surroundings as the motor rotates between 0° and 180°. The Arduino controls the sweep, triggers the sensor, reads the echo signal, and calculates the distance based on time delay.

**5. WORKING PRINCIPLE**

The radar-based object detection system uses an ultrasonic sensor to measure the distance of objects based on the time delay of reflected sound waves. The sensor is mounted on a servo motor that sweeps from 0° to 180°, mimicking the scanning motion of a real radar system.

🔁 Step-by-Step Operation:

1. The Arduino rotates the servo motor gradually from 0° to 180°, and then reverses the direction.

2. At each angle, it sends a trigger pulse to the HC-SR04 ultrasonic sensor.

3. The sensor emits an ultrasonic wave (40kHz) and waits for it to reflect back from an object.

4. The Echo pin stays HIGH until the reflected wave is received. The time duration between sending and receiving is measured.

5. The Arduino calculates the distance using the formula:

\[

\text{Distance (cm)} = \frac{\text{Time (μs)} \times 0.0343}{2}

\]

6. The angle and distance are printed on the Serial Monitor or transmitted to the Processing IDE to visualize a radar screen.

🔍 Key Concepts Applied:

• Pulse timing and signal delay

• Real-time embedded data acquisition

• Servo positioning control

• Serial communication

• Sensor rotation synchronization

This project demonstrates how embedded systems can simulate real-world technologies using cost-effective hardware and basic programming.

**6.ARDUINO CODE**

// Includes the Servo library

#include <Servo.h>

// Defines Tirg and Echo pins of the Ultrasonic Sensor

const int trigPin = 10;

const int echoPin = 11;

int angle;

// Variables for the duration and the distance

long duration;

float distance;

Servo myServo; // Creates a servo object for controlling the servo motor

void setup() {

pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output

pinMode(echoPin, INPUT); // Sets the echoPin as an Input

Serial.begin(9600);

myServo.attach(12); // Defines on which pin is the servo motor attached

}

void loop() {

// rotates the servo motor from 15 to 165 degrees

for(int angle=0;angle<=180;angle++){

myServo.write(angle);

delay(15);

float distance = getDistance();// Calls a function for calculating the distance measured by the Ultrasonic sensor for each degree

Serial.print(angle); // Sends the current degree into the Serial Port

Serial.print(","); // Sends addition character right next to the previous value needed later in the Processing IDE for indexing

Serial.print(distance,1); // Sends the distance value into the Serial Port

Serial.print("."); // Sends addition character right next to the previous value needed later in the Processing IDE for indexing

}

// Repeats the previous lines from 165 to 15 degrees

for(angle=180;angle>=0;angle--){

myServo.write(angle);

delay(15);

float distance = getDistance();

Serial.print(angle);

Serial.print(",");

Serial.print(distance);

Serial.print(".");

}

}

// Function for calculating the distance measured by the Ultrasonic sensor

float getDistance(){

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

// Sets the trigPin on HIGH state for 10 micro seconds

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

float duration = pulseIn(echoPin, HIGH); // Reads the echoPin, returns the sound wave travel time in microseconds

float distance= duration\*0.034/2;

return distance;

}

**7.PROCESSING CODE**

import processing.serial.\*; // Serial communication

Serial myPort; // Serial object

// Variables

String angle = "";

String distance = "";

String data = "";

String noObject;

float pixsDistance;

int iAngle, iDistance;

int index1 = 0;

int index2 = 0;

void setup() {

size(1200, 700); // Adjust screen resolution if needed

smooth();

// Set correct COM port here

myPort = new Serial(this, "COM8", 9600); // ⚠ Make sure COM8 is your Arduino port

myPort.bufferUntil('.'); // Read data until '.'

}

void draw() {

// Clear screen with blur effect

noStroke();

fill(0, 4);

rect(0, 0, width, height - height \* 0.065);

fill(98, 245, 31); // Green color

// Drawing functions

drawRadar();

drawLine();

drawObject();

drawText();

}

// Read from Serial

void serialEvent(Serial myPort) {

data = myPort.readStringUntil('.');

if (data != null) {

data = trim(data); // Remove extra characters

index1 = data.indexOf(',');

if (index1 > 0 && index1 < data.length() - 1) {

angle = data.substring(0, index1);

distance = data.substring(index1 + 1);

// Convert to int

iAngle = int(angle);

iDistance = int(distance);

}

}

}

void drawRadar() {

pushMatrix();

translate(width/2, height - height \* 0.074);

noFill();

strokeWeight(2);

stroke(98, 245, 31);

// Arcs

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);

// Lines

for (int a = 30; a <= 150; a += 30) {

line(0, 0, (-width/2) \* cos(radians(a)), (-width/2) \* sin(radians(a)));

}

line(-width/2, 0, width/2, 0);

popMatrix();

}

void drawObject() {

pushMatrix();

translate(width/2, height - height \* 0.074);

strokeWeight(9);

stroke(255, 10, 10); // Red

pixsDistance = iDistance \* ((height - height \* 0.1666) \* 0.025);

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)));

}

popMatrix();

}

void drawLine() {

pushMatrix();

strokeWeight(9);

stroke(30, 250, 60); // Green

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();

}

void drawText() {

pushMatrix();

if (iDistance > 40) {

noObject = "Out of Range";

} else {

noObject = "In Range";

}

fill(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("SciCraft", 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 \* 0.225, height - height \* 0.0277);

}

textSize(25);

fill(98, 245, 60);

// Draw angle labels

drawAngleLabel(30, -60, "30°");

drawAngleLabel(60, -30, "60°");

drawAngleLabel(90, 0, "90°");

drawAngleLabel(120, 30, "120°");

drawAngleLabel(150, 60, "150°");

popMatrix();

}

void drawAngleLabel(int angleDeg, float rot, String label) {

resetMatrix();

translate((width - width \* 0.4994) + width / 2 \* cos(radians(angleDeg)),

(height - height \* 0.0907) - width / 2 \* sin(radians(angleDeg)));

rotate(radians(rot));

text(label, 0, 0);

}

**8. APPLICATIONS**

This radar-based object detection system can be adapted or expanded for various practical uses in both academic and industrial environments. Some potential applications include:

• Obstacle Detection in Robotics

Used in mobile robots or autonomous vehicles to detect and avoid obstacles in real-time.

• Surveillance and Security Systems

Detects motion or presence of objects/people within a predefined range.

• Blind Spot Detection

Can be used in vehicles to alert the driver about obstacles in blind spots.

• Proximity Warning Systems

Used in parking systems or warehouse automation to prevent collisions.

• Educational and Research Projects

Serves as a simple and effective way to understand radar principles, sensor interfacing, and embedded system design.

With enhancements like Wi-Fi or Bluetooth modules, this system can also be upgraded for remote monitoring or IoT applications.

**9. CONCLUSION**

The radar-based object detection system using Arduino was successfully designed and implemented. The project achieved its primary objective of simulating a basic radar functionality using affordable hardware components like the Arduino UNO, HC-SR04 ultrasonic sensor, and SG90 servo motor.

The system is capable of detecting objects, measuring distances, and scanning across a defined angular range. The angle–distance values were displayed effectively on the Serial Monitor and optionally visualized using Processing IDE.

Through this mini project, I gained hands-on experience in:

• Embedded C programming

• Servo motor control

• Ultrasonic sensor interfacing

• Serial communication

• Real-time data processing and display

The project enhanced my confidence in working with embedded systems and gave me deeper insights into real-world applications of electronics and communication engineering.