Radar sensing using Ultrasonic sensor

A Design Lab Project Report Submitted in Partial Fulfillment of the Requirements for the Degree of

Bachelor of Technology

By

Posham Navya-(220108043)

Vankudothu Dheeraj-(220108061)

under the guidance of

Dr. Manoj B R

to the

DEPARTMENT OF ELECTRONICS AND ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

GUWAHATI - 781039, ASSAM

<u>CERTIFICATE</u>

This is to certify that the work contained in this thesis entitled "Radar sensing using Ultrasonic sensor" is a bonafide work of Posham Navya (Roll No. 220108043) and Vankudothu Dheeraj (Roll No. 220108061), carried out in the Department Electronics and Electrical Engineering, Indian Institute of Technology Guwahati under my supervision and that it has not been submitted elsewhere for a degree.

Supervisor: **Dr. Manoj B R**

Assistant Professor,
Dept of Electronics & Electrical Engineering,
Indian Institute of Technology Guwahati, Assam.

May, 2025,
IIT Guwahati.

Acknowledgements

We wish to express our deep appreciation to Dr. Manoj B R for his valuable guidance, assistance, patience, and encouragement while developing this project.

We also want to express our sincere gratitude to the Electronics Club, IIT Guwahati, for constantly supporting us with their valuable resources and giving us a working environment.

Contents

Chapter 1

Theory

1.1 Arduino Uno

One of the most popular Arduino boards out there is the Arduino Uno. While it was not actually the first board to be released, it remains to be the most actively used and most widely documented on the market. Because of its extreme popularity, the Arduino Uno has a ton of project tutorials and forums around the web that can help you get started or out of a jam. We're big fans of the Uno because of it's great features and ease of use. **Board Breakdown:**

Here are the components that make up an Arduino board and what each of their functions are.

1. **Reset Button –** This will restart any code that is loaded to the Arduino board
2. **AREF** – Stands for "Analog Reference" and is used to set an external reference voltage
3. **Ground Pin –** There are a few ground pins on the Arduino and they all work the same
4. **Digital Input/Output –** Pins 0-13 can be used for digital input or output

5. **PWM –** The pins marked with the (˜) symbol can simulate analog output
6. **USB Connection –** Used for powering up your Arduino and uploading sketches
7. **TX/RX –** Transmit and receive data indication LEDs
8. **ATmega Microcontroller –** This is the brains and is where the programs are stored
9. **Power LED Indicator –** This LED lights up anytime the board is plugged in a power source
10. **Voltage Regulator –** This controls the amount of voltage going into the Arduino board
11. **DC Power Barrel Jack –** This is used for powering your Arduino with a power supply
12. **3.3V Pin –** This pin supplies 3.3 volts of power to your projects
13. **5V Pin –** This pin supplies 5 volts of power to your projects
14. **Ground Pins –** There are a few ground pins on the Arduino and they all work the same
15. **Analog Pins –** These pins can read the signal from an analog sensor and convert it to digital

The Arduino Uno needs a power source in order for it to operate and can be powered in a variety of ways. You can do what most people do and connect the board directly to your computer via a USB cable. If you want your project to be mobile, consider using a 9V battery pack to give it juice. The last method would be to use a 9V AC power supply. **How To Program Arduino:**
Once the circuit has been created on the breadboard, you'll need to upload the program (known as a sketch) to the Arduino. The sketch is a set of instructions that tells the board what functions it needs to perform. An Arduino board can only hold and perform one sketch at a time. The software used to create Arduino sketches is called the IDE which stands for Integrated Development Environment. The software is free to download and can be found at https://www.arduino.cc/en/Main/Software

1.2 Ultrasonic Sensor

HC-SR04 Ultrasonic Sensor is a very affordable proximity/distance sensor that has been used mainly for object avoidance in various robotics projects. It has also been used in turret applications, water level sensing, and even as a parking sensor.
This module is the second generation of the popular HC-SR04 Low Cost Ultrasonic Sensor. Unlike the first generation HC-SR04 that can only operate between 4.8V˜5V DC, this new version has wider input voltage range, allow it to work with controller operates on 3.3V.HCSR04 ultrasonic sensor provides a very low-cost and easy method of distance measurement. It measures distance using sonar, an ultrasonic (well above human hearing) pulse (˜40KHz) is transmitted from the unit and distance-to-target is determined by measuring the time required for the echo return. This sensor offers excellent range accuracy and stable readings in an easy-to-use package. An on board 2.54mm pitch pin header allows the sensor to be plugged into a solderless breadboard for easy prototyping.
Hardware Information:

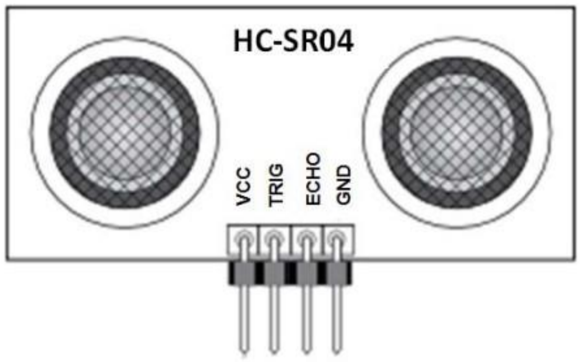

Fig. 1.1 Pins of Ultrasonic Sensor

VCC	3.3v ~ 5V
TRIG	Triggering Input Pin. 10uS TTL Pulses
ECHO	TTL Logic Output Pin. Proportional to distance
GND	Ground Pin

Table 1.1 Pin Description

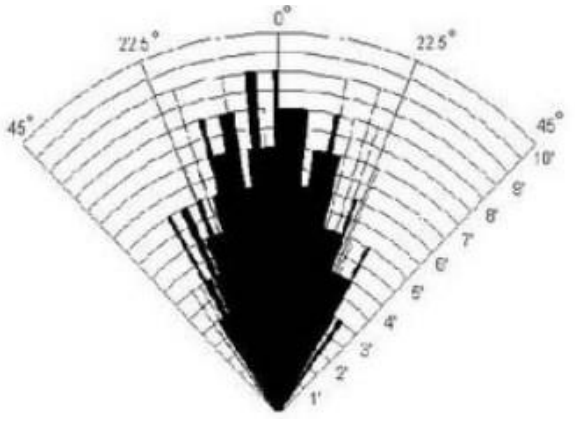

Fig. 1.2 Practical test of performance Best in 30 degree angle

1.3 Servo Motor

A **servo motor** is a type of motor commonly used in Arduino projects for precise control of angular postion. It is a compact device that includes a DC motor, gear system, position sensor (usually a potentiometer), and a control circuit inside one unit. Unlike regular DC motors, servo motors can rotate to a specific angle and hold that position.

Working Principle:

The servo motor works using a **PWM (Pulse Width Modulation)** signal sent from the Arduino. The width of the pulse determines the rotation angle of the motor shaft. The internal circuit continuously compares the desired angle with the current position (from the sensor) and adjusts the motor until the target position is reached.

- A 1ms pulse sets the motor to **0**
- A 1.5ms pulse sets it to **90**
- A 2ms pulse sets it to **180**

This control signal is usually sent every 20 ms (50 Hz).

Servo Motor Pins:

Servos have **three pins**:

Pin Name	Wire Color	Function
Vcc	Red	Power supply (usually 5V)
GND	Brown or Black	Ground
Signal	Orange, Yellow, or White	PWM control signal from Arduino

These pins are connected directly to the Arduino:

- **Vcc** to **5V**
- **GND** to **GND**
- **Signal** to any **PWM-capable digital pin** (e.g., D9)

Usage:

Servo motors are ideal for tasks where **limited-angle movement** is required, such as:

- Robotic arms
- Steering mechanisms in RC vehicles
- Camera positioning systems

Chapter 2

Hardware Implementation

2.1 Power Supply Unit

The Arduino and connected peripherals are powered via:

- USB (5V regulated) from the PC during development.
- 9V battery with a voltage regulator (for standalone operation).

Component	Voltage Supplied	Current Draw
Arduino Uno	5V	50 mA
HC-SR04	5V	15 mA
SG90 Servo	5V	˜250 mA peak

Ultrasonic Sensor (HC-SR04) specifications:

Parameter	Value
Operating Voltage	5V
Detection Range	2 cm – 400 cm
Accuracy	3 mm
Frequency	40 kHz

Servo Motor (SG90) specifications:

Parameter	Value
Operating Voltage	4.8 – 6V
Torque	1.8 kg·cm
Speed 0.1 sec/60	
Rotation Range 0 to 180	

2.2 Wiring and Signal Transmission

Jumper wires are used to establish reliable connections between components on a breadboard or PCB. The wiring ensures correct power delivery and signal routing.

Component	Pin	Arduino Connection
HC-SR04	Vcc	5V
HC-SR04	GND	GND
HC-SR04	Trig	Digital Pin 10
HC-SR04	Echo	Digital Pin 11
SG90 Servo	Vcc (Red)	5V
SG90 Servo	GND (Black)	GND
SG90 Servo	Signal	Digital Pin 9

2.3 Ultrasonic Sensor (HC-SR04) to Arduino

The **HC-SR04** operates by sending a high-frequency ultrasonic pulse and measuring the time it takes for the echo to return after hitting an obstacle. It has four pins:

HC-SR04 Pin	Arduino Connection	Description
Vcc	5V	Provides 5V power to the sensor.
GND	GND	Common ground to complete the circuit.
Trig	Digital Pin 10	Arduino sets this HIGH to initiate the ultrasonic pulse.
Echo	Digital Pin 11	Sensor sets this HIGH proportional to the distance.

Table 2.1 Ultrasonic Sensor to Arduino

Explanation:

- Arduino sends a 10 s HIGH signal to **Trig** to generate the pulse.

- The sensor emits an ultrasonic wave at 40 kHz.
- When the echo returns, the **Echo** pin stays HIGH for a time proportional to the distance.
- The Arduino uses pulseIn(Echo, HIG) to measure this time.

2.4 Servo Motor (SG90) to Arduino

The **SG90 servo motor** is controlled using PWM signals from the Arduino and has three wires:

Working Explanation:

Servo Wire	Arduino Connection	Description
Red (Vcc)	5V	Provides power to the servo.
Brown/Black (GND)	GND	Common ground connection.
Orange/Yellow (Signal)	Digital Pin 9	Arduino sends PWM to control servo angle.

Table 2.2 Servo Motor to Arduino

- The Arduino uses the Servo library to send PWM signals.
- PWM signal width determines angle:
  - ˜1 ms = 0
  - ˜1.5 ms = 90
  - ˜2 ms = 180
- The servo rotates the attached ultrasonic sensor accordingly for scanning.

2.5 Serial Communication (Arduino to PC)

The Arduino communicates the measured angle and distance to a PC using **USB serial**.

Working Explanation:

- The Arduino uses Serial.begin(9600) to initialize the communication.

- Data is sent using Serial.print() or Serial.println() functions.

  On the PC side, Processing reads the serial stream and interprets it for visualization.

Function	Connection	Description
Data Output	USB to PC	Sends angle and distance readings.
Serial Monitor/Processing	COM Port	Receives and displays data in text or graphical radar form.

Table 2.3 Serial Communication

2.6 Circuit Diagram

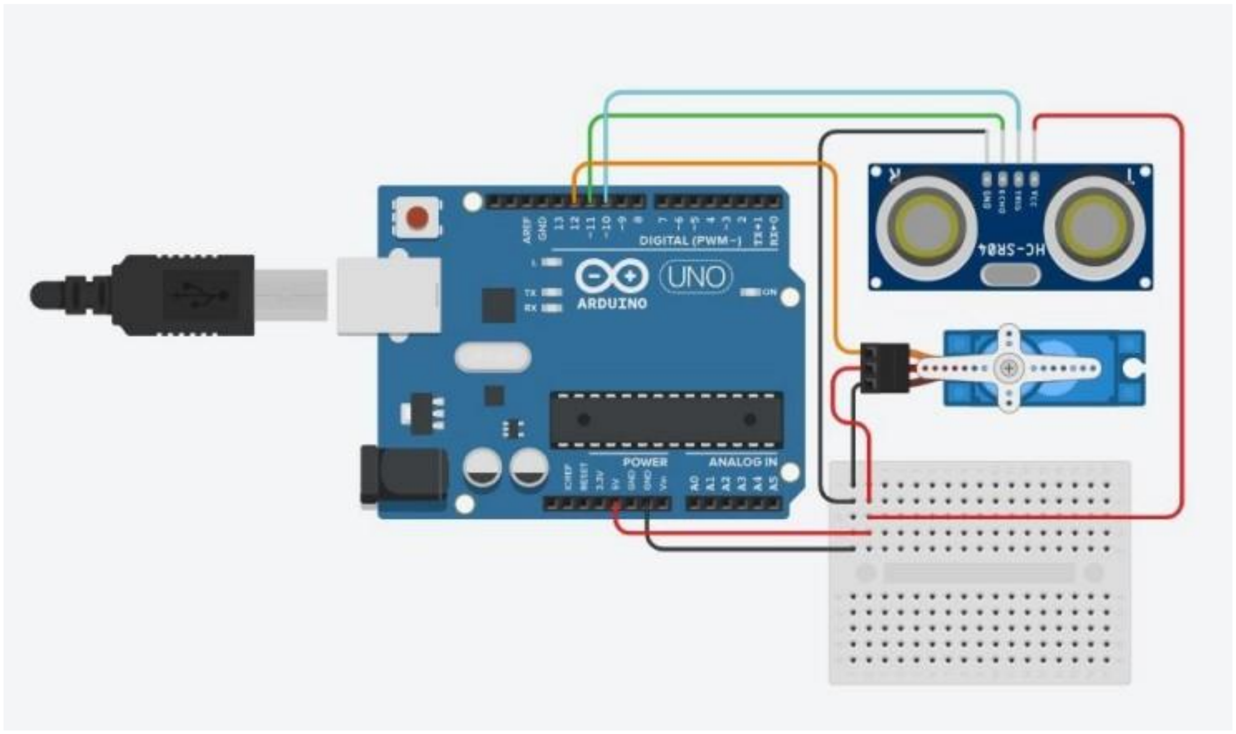

Fig. 2.1 Circuit diagram

Chapter 3

Software Implementation

3.1 Initialization and Library Inclusions

The code begins by including the required libraries:

- Servo.h for controlling the servo motor.
- SoftwareSerial or Serial for data transmission.

3.2 Setup Function

The setup() function initializes pin modes and serial communication.

Code:

```
void setup() {
myServo.attach(servoPin);
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
Serial.begin(9600);
}
```

3.3 Distance Measurement Logic

This block sends an ultrasonic pulse and calculates the distance based on the time taken for the echo to return.

Code:

```
long getDistance(){
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);

long duration = pulseIn(echoPin, HIGH);
long distance = duration * 0.034 / 2;
return distance;
}
```

3.4 Servo Scanning Loop

The servo is rotated from 0 to 180, and then back from 180 to 0, taking distance readings at each step.

Code:

```
 void loop(){
for(int angle=0; angle<=180; angle++){
myServo.write(angle);
delay(50);
long distance = getDistance();
Serial.print(angle);
Serial.print(",");
Serial.println(distance);
}


for (int angle = 180; angle >= 0; angle--){
myServo.write(angle);
delay(50);
long distance = getDistance();
Serial.print(angle);
Serial.print(",");
Serial.println(distance);
}
}
```

3.5 Serial Data Transmission

The angle and corresponding distance are transmitted in a comma-separated format via USB serial, which can be read by:

- Arduino Serial Monitor (textual display).
- Processing IDE (graphical radar plot).

Chapter 4

Result

The radar system was successfully implemented and tested using an Arduino Uno, HC-SR04 ultrasonic sensor, and SG90 servo motor. The device demonstrated real-time scanning capability across a 180 range, with accurate obstacle detection and responsive angular
control.

4.1 System Behavior

Upon powering the system:

- The **servo motor** began sweeping from **0 to 180** and back.
- At each angular position, the **ultrasonic sensor** emitted a pulse and measured the distance to the nearest object.
- The angle and distance values were transmitted via **serial communication** to the host computer.

4.2 Textual Output (Serial Monitor)

The Arduino Serial Monitor displayed live data in the format:

Angle,Distance (in cm):

```
0,50

10,48

20,47
```

...

This format provides easy logging and debugging for distance data corresponding to each angle.

4.3 Graphical Output (Radar Interface using Processing)

Using the Processing IDE, a **real-time radar interface** was implemented. The results included:

- A **semicircular radar sweep** line (simulated via graphics).
- **Red blips** indicating detected objects.
- **Green arcs** denoting areas with no obstacles.
- **Live plotting** that updated with each sweep of the servo.

After Running the processing Code the Visual Output Of Radar Look Like this:

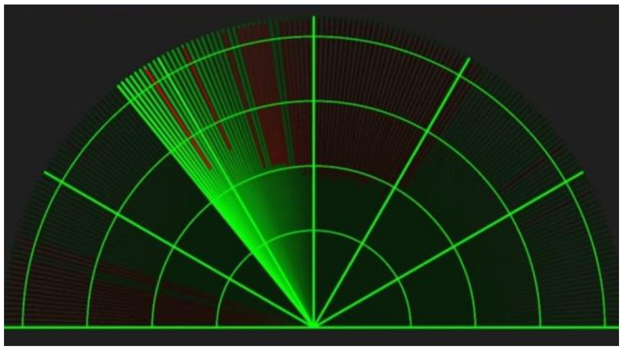

Fig. 4.1 Visual output

4.4 Object Detection Accuracy

Objects at various distances and angles were placed and tested. The system successfully identified objects and plotted them at correct angles. A table below summarizes a few test cases:

Actual Distance (cm)	Measured Distance (cm)	Error (cm)
50	47	3
80	82	2
100	98	2
120	125	5

Table 4.1 Comparision of actual and measured values

4.5 System Performance Observations

- **Update Rate:** ˜1 reading every 50 ms.
- **Coverage Angle:** 180 scan, with 1 resolution.
- **Operational Range:** 2 cm to 400 cm.
- **Servo Smoothness:** Stable and consistent sweeping without jitter.
- **Power Stability:** No resets or voltage drops under USB-powered operation.

4.6 Limitations Observed

Noise Sensitivity: Readings occasionally fluctuated due to reflections from small or irregular surfaces.

Servo Speed: The scanning speed is limited by the servo’s mechanical rotation and distance measurement delay.

Processing Lag: For high-resolution visualization, the Processing display may lag slightly depending on PC performance.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

The radar system designed using an Arduino Uno, HC-SR04 ultrasonic sensor, and SG90 servo motor successfully demonstrated real-time obstacle detection and environmental scanning over a 180 range. The system was able to detect objects with reasonable accuracy, convert the distance and angle data into meaningful visual outputs, and simulate a basic radar mechanism using open-source hardware and software tools.

The integration of hardware components with software visualization in Processing provided a clear understanding of embedded control systems, signal processing, and data visualization. This project proves that low-cost, microcontroller-based solutions can effectively emulate more complex systems at a prototype level.

5.2 Future Work

While the current system performs well for its intended scope, several enhancements can be introduced to improve functionality, range, and usability:

360 Scanning: Use of a continuous rotation servo or stepper motor to allow full-circle detection.

Distance Filtering: Implement filtering algorithms (e.g., median or Kalman filters) to reduce noise and improve reading stability.

Obstacle Classification: Use multiple sensors or combine with a camera module to identify object types, not just distance.

Wireless Communication: Incorporate Bluetooth, Wi-Fi (e.g., ESP8266), or LoRa for wireless data transmission to mobile apps or cloud dashboards.

Autonomous Systems: Integrate with mobile robots or drones to enable intelligent pathfinding and collision avoidance.

Battery Optimization: Develop a power-efficient version using sleep modes or rechargeable Li-ion battery modules.

3D Mapping: Mount the sensor on two servos (pan and tilt) to scan in both horizontal and vertical directions for depth mapping.

5.3 Applications

This low-cost radar system has a wide range of educational, prototyping, and practical applications, such as:

Obstacle Avoidance in Robots: Helps autonomous robots detect and avoid collisions

Surveillance Systems: Useful in simple motion detection systems for security.

Distance Sensing in Industrial Systems: Basic presence detection in assembly lines or automated doors.

Educational Projects: Ideal for teaching embedded systems, sensor integration, and real-time data visualization.

Smart Home Devices: Can be adapted for human presence detection in home automation systems.

Blind Spot Detection: Adapted versions can assist in automotive safety features.

References

1. "Exploring Arduino: Tools and Techniques for Engineering Wizardry" by Jeremy Blum

2. "Arduino Robotics" by John-David Warren, Josh Adams, and Harald Molle

3. Internet Documentations