# Project Report

**on**

# Smart Parking System

in partial fulfilment for the award of the degree of

**BACHELOR OF ENGINEERING**

IN

**Branch Name**

**Submitted by:**

Govind mittal(24BAI70128)

Praatibh(24BAI70266)

Ankush(24BAI70467)

….

**Under the Guidance of**

Dr. Divneet singh Kapoor
**ACADEMIC UNIT-I**

**Chandigarh University**

**April 2025**

# Index

## 1. Project Overview

This project is based on the development of an ESP32-based ultrasonic radar system. It combines real-time object detection and servo motor control to visually simulate radar scanning. The radar setup can detect obstacles using an ultrasonic sensor and display their distance and angular position through a web-based user interface.

This project is important as it demonstrates a real-world application of IoT, microcontroller programming, and web-based visualization. Such systems are used in robotics, automation, and smart surveillance applications

### 2. Objective and Problem Statement

The main goal of this project is to create a functional and affordable radar system that can:

Detect objects and obstacles in its vicinity.

Continuously scan using a servo motor.

Display the results visually in a web browser using a radar-style interface and a graph.

The problem it addresses is the complexity and cost of existing commercial radar systems. This project aims to build a low-cost, accessible alternative for learning and prototyping.

### 3. Proposed Solution & Methodology

Tools & Components Used:

ESP32 Development Board

Ultrasonic Sensor (HC-SR04)

SG90 Servo Motor

Web Browser

HTML, JavaScript (Chart.js)

Arduino IDE

***Block Diagram:***

```
           +-----------------+
           |   Web Browser   |
           | (Radar + Graph) |
           +--------+--------+
                    |
                  | WiFi |
                    V
             +------+------+
             |  ESP32 MCU  |
             +------+------+
                    |
      +------------+-------------+
      |                          |
      V                          V
+----------------+       +----------------+
| Ultrasonic     |       |   Servo Motor  |
| Sensor (HC-SR04)|      |    (SG90)      |
+----------------+       +----------------+
```

**Steps:**

1. Connect HC-SR04 sensor and servo to ESP32.
2. Program ESP32 to rotate servo and read distance.
3. Host a web server on ESP32.
4. Transmit data to web page using JSON.
5. Render live radar and graph using JavaScript.
6. Simultaneously send distance data to Ubidots for IoT cloud storage.

## 4. Key Findings / Results

The servo motor continuously sweeps from 0° to 180° and back.

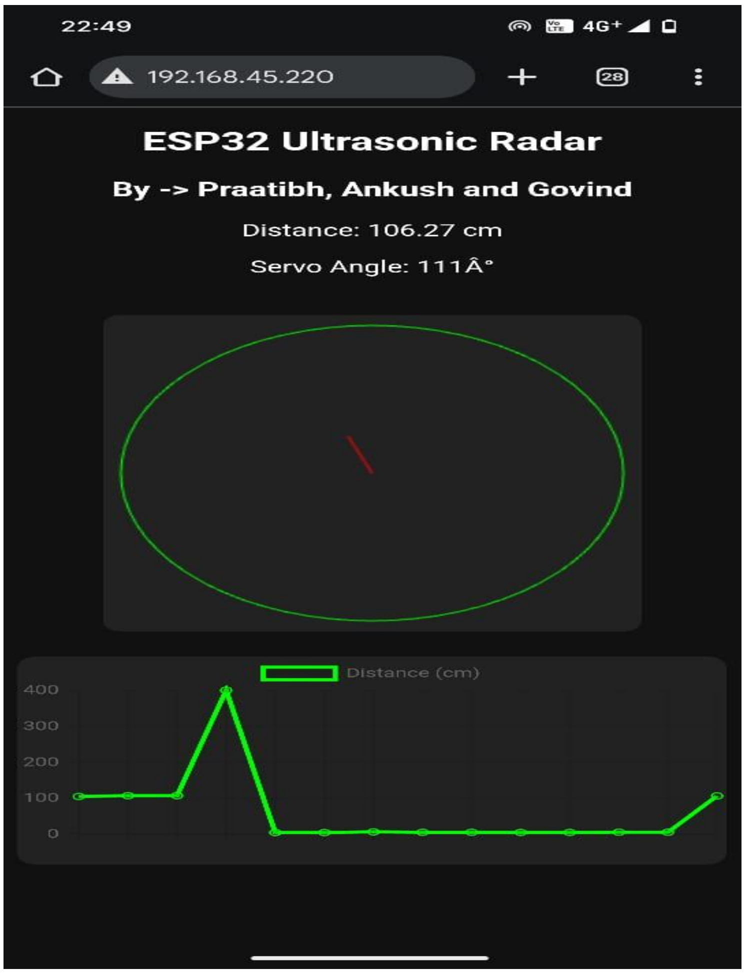
Ultrasonic sensor reads distance and updates every 200 milliseconds.

A radar UI with animation and real-time graph updates in the browser.

Ubidots receives data in real-time and logs it for remote monitoring.

Snapshots:

Screenshot of live radar UI in browser


22:49

192.168.45.220

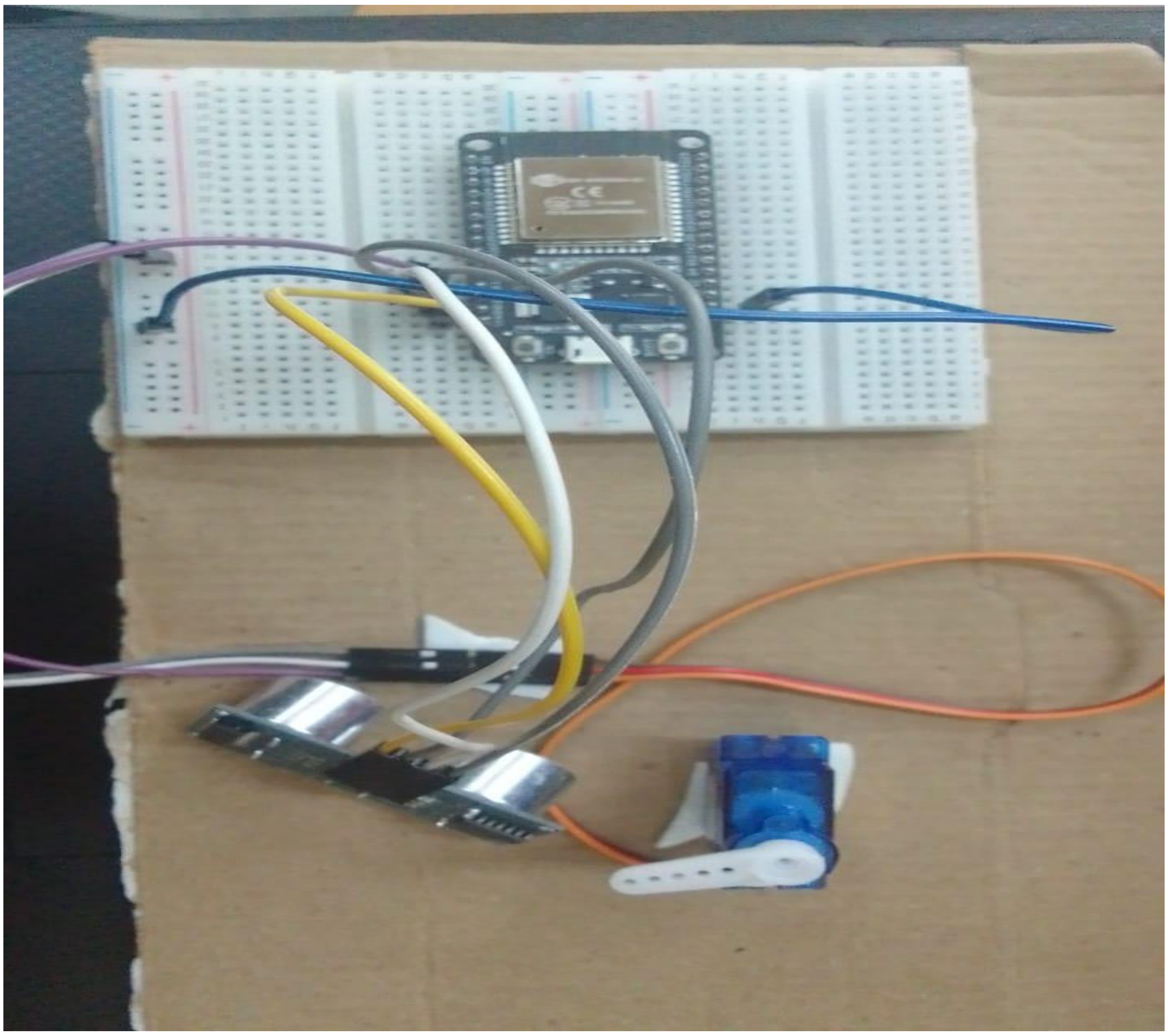
ESP32 Ultrasonic Radar

By -> Praatibh, Ankush and Govind

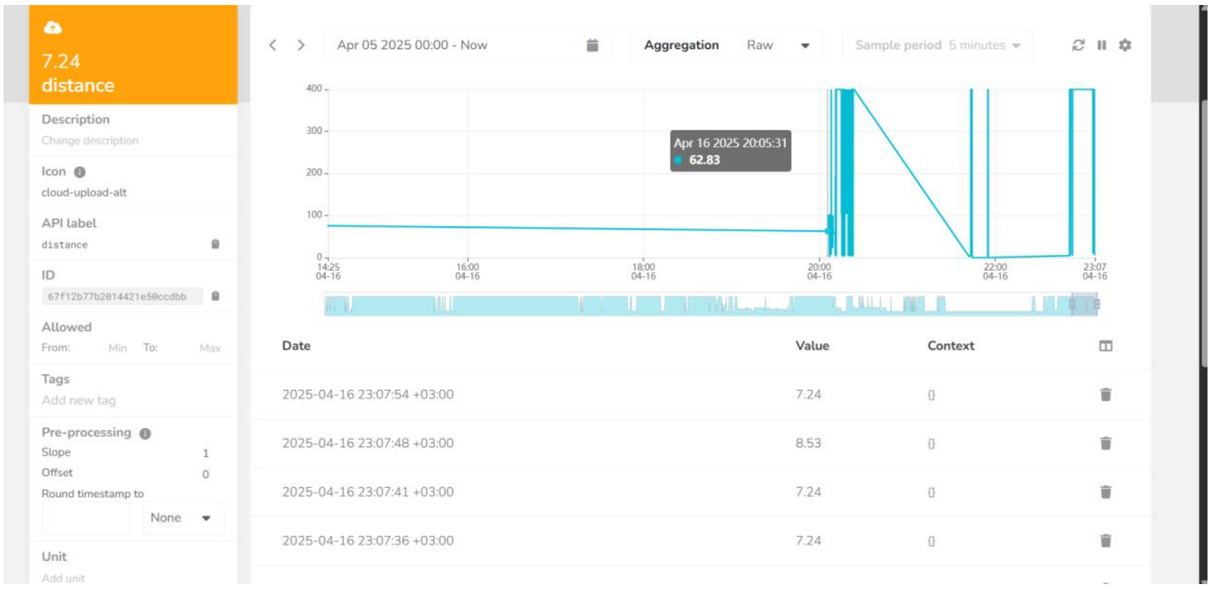
Distance: 106.27 cm

Servo Angle: 111Â°

ESP32 setup image with sensor and servo



Ubidots dashboard view



7.24
distance

Description
Change description

Icon
cloud-upload-alt

API label
distance

ID
67f12b77b2814421e58ccdbb

Allowed
From: Min To: Max

Tags
Add new tag

Pre-processing
Slope 1
Offset 0
Round timestamp to
None

Unit
Add unit

Apr 05 2025 00:00 - Now

Aggregation Raw

Sample period 5 minutes

| Date | Value | Context |
|---|---|---|
| 2025-04-16 23:07:54 +03:00 | 7.24 | {} |
| 2025-04-16 23:07:48 +03:00 | 8.53 | {} |
| 2025-04-16 23:07:41 +03:00 | 7.24 | {} |
| 2025-04-16 23:07:36 +03:00 | 7.24 | {} |

## 5. WebServer HTML Code

```
<!DOCTYPE html><html><head><title>ESP32 Radar</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
    <style>
      body { font-family: Arial; text-align: center; background: #111; color:
white; }
      canvas { background: #222; margin-top: 20px; border-radius: 10px; }
    </style></head><body>
    <h2>ESP32 Ultrasonic Radar</h2>
    <h3>By -> Praatibh, Ankush and Govind</h3>
    <p>Distance: <span id="distance">0</span> cm</p>
    <p>Servo Angle: <span id="angle">0</span>°</p>
    <canvas id="radar" width="300" height="300"></canvas>
    <canvas id="chart" width="300" height="150"></canvas>
    <script>
      const ctxRadar = document.getElementById('radar').getContext('2d');
      const ctxChart = document.getElementById('chart').getContext('2d');
      let chart = new Chart(ctxChart, {
        type: 'line', data: { labels: [], datasets: [{
          label: 'Distance (cm)', borderColor: 'lime', data: [], fill: false
        }]},
        options: { animation: false, scales: { y: { suggestedMin: 0,
suggestedMax: 100 } } }
      });

      let radarCtx = ctxRadar;
      function drawRadar(angle, distance) {
        radarCtx.clearRect(0, 0, 300, 300);
        radarCtx.beginPath();
        radarCtx.arc(150, 150, 140, 0, Math.PI * 2); radarCtx.strokeStyle =
'lime'; radarCtx.stroke();
        let rad = angle * Math.PI / 180;
        let x = 150 + Math.cos(rad) * distance * 0.35;
        let y = 150 - Math.sin(rad) * distance * 0.35;
        radarCtx.beginPath();
        radarCtx.moveTo(150, 150);
        radarCtx.lineTo(x, y);
        radarCtx.strokeStyle = 'red';
        radarCtx.stroke();
      }

      function fetchData() {
        fetch('/data').then(r => r.json()).then(data => {
          document.getElementById('distance').textContent = data.distance;
          document.getElementById('angle').textContent = data.angle;
          chart.data.labels.push('');
          chart.data.datasets[0].data.push(data.distance);
          if (chart.data.labels.length > 20) {
            chart.data.labels.shift(); chart.data.datasets[0].data.shift();
          }
          chart.update();
          drawRadar(data.angle, data.distance);
        });
      }
      setInterval(fetchData, 1000);
    </script>
    </body></html>
```

**6. Conclusion & Learnings**

This project provided hands-on experience with:

ESP32 microcontroller programming

Web-based real-time UI rendering

Servo control and ultrasonic data acquisition

IoT dashboard integration (Ubidots)

Future improvements could include obstacle classification, adding more sensors, or integration with mobile notifications or voice assistants.

**7. References**

https://randomnerdtutorials.com

https://ubidots.com/docs/

https://www.arduino.cc/

https://cdn.jsdelivr.net/npm/chart.js

## 8. Appendix

Arduino source code (main.ino)

Wiring diagrams

JSON data sample output

Screenshot of network traffic to /data endpoint