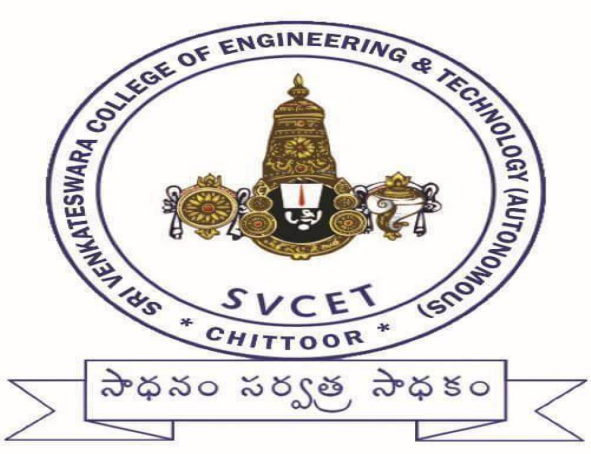


SRI VENKATESWARA COLLEGE OF ENGINEERING AND TECHNOLOGY(AUTONOMOUS)

R. V. S. NAGAR, CHITTOOR - 517127. (A.P).

(Approved by AICTE, New Delhi, Affiliated to JNTUA, Anantapur)

(Accredited by NBA, NEW Delhi AND NAAC, Bengaluru)

(An ISO 9001:2000 Certified Institution)

July 2024

PROJECT DETAILS:

CLASS : Third year ECE (B.Tech), second semester.

COURSE NAME: Cyber Physical Systems for Industrial Applications.

PROJECT NAME : Advanced radar sensor integration for industrial automation systems

BATCH: Batch 23(3)

1. **K.NAGA NEERAJ -(21781A0492)**
2. **K.JAGAN MOHAN REDDY -(21781A0493)**
3. **K.LAVANYA -(21781A041A6)**

CONTENTS

PROJECT

ADVANCED RADAR SENSOR INTEGRATION FOR INDUSTRIAL AUTOMATION SYSTEMS

AIM OF THE PROJECT :

To design automatic radar sensor system using suitable sensors and controllers

PROBLEM STATEMENT :

In industrial automation, there is an increasing demand for efficient and reliable detection systems to enhance safety, productivity, and operational efficiency. Traditional detection methods, such as proximity sensors, have limitations in detecting objects at longer distances and in adverse environmental conditions .To address these challenges design a system where the radar sensors interfaced with the controller can detect the objects and track it.

PROBLEM SOLUTION :

1. Hardware Selection:

Choose a suitable radar sensor module compatible with Arduino, such as the RCWL-0516 or HB100, based on detection range and sensitivity requirements.

2.Circuit Design:

Connect the radar sensor module to the Arduino board.
Connections: VCC to 5V pin, GND to GND pin, and OUT to a digital input pin (e.g., D2) on Arduino.
Additional Components: Depending on the module, you may need capacitors or resistors for stability and filtering.

3.Arduino Programming

Write a program (sketch) to interface with the radar sensor module.
Setup: Initialize serial communication and configure the input pin.
Loop: Continuously read the sensor output and respond to detection events.
Detection Logic: Use digitalRead() to monitor the sensor output. Upon detecting motion (HIGH signal), trigger appropriate actions (e.g., print message, activate an alarm).

4.Testing and Calibration:

Upload the code to Arduino and test the system.
Use the Serial Monitor to observe detection events and adjust sensitivity or detection parameters if needed.
Validate the system's performance in different environmental conditions to ensure reliable operation.

5.Integration and Application:

Integrate the radar detection system into your intended application or project.
Consider power management, enclosure design, and any additional features (e.g., integration with other sensors, data logging).
Documentation and Maintenance:

Document the system design, circuit diagram, and software implementation.
Ensure the system is maint
Certainly! If you're looking to solve a radar detection problem using Arduino, here's a structured problem statement and solution approach:

PROJECT DESIGN SPECIFICATION:

1. Objective:

- Design and implement a radar detection system using Arduino that detects motion within a defined range and provides real-time feedback.

2. Components:

- **Arduino Board**: Arduino Uno or similar.
- **Radar Sensor Module:** Choose a suitable module like RCWL-0516 or HB100.
- **Power Supply:** 5V DC for Arduino and appropriate voltage for radar module.
- **Optional Components:** Resistors, capacitors for stability; LEDs, buzzer for visual or auditory feedback.

3. System Requirements:

- **Detection Range:** Adjustable, minimum 5 meters.
- **Detection Angle:** Preferably 360 degrees or adjustable.
- **Detection Sensitivity:** Adjustable to minimize false positives.
- **Response Time:** Instantaneous detection with minimal delay.
- **Feedback Mechanism:** Serial output to monitor or integrate with other systems; optional visual or auditory alerts.

4. Hardware Design:

- **Connections:**

- **Radar Module:**

VCC to Arduino 5V.
GND to Arduino GND.
OUT to a digital input pin (e.g., D2) on Arduino.
Optional feedback components (LED, buzzer) connected as needed.

- **Power Considerations:**

Ensure stable power supply for both Arduino and radar module to prevent erratic behaviour.

- **Physical Setup:**

Securely mount radar module to ensure optimal detection coverage.
Enclose components in a suitable housing for protection and aesthetics.

5. Software Design:

- **Arduino Sketch:**

Initialize serial communication for debugging and feedback.
Configure radar module pin as input.
Continuously monitor radar module output using digitalRead().
Implement logic to detect motion (change in signal from radar module).
Trigger appropriate actions upon detection (e.g., print message, activate LED or buzzer).

- **Additional Features (Optional):**

Data logging to external storage (SD card).

Wireless communication (Bluetooth, Wi-Fi) for remote monitoring.
Integration with IoT platforms for cloud-based notifications.

6. Testing and Validation:

- **Functional Testing:**

Verify detection accuracy in various environments (indoor/outdoor).
Test response to different motion speeds and directions.
Validate sensitivity adjustments to minimize false positives.

- **Integration Testing:**

Ensure seamless integration with feedback components.
Verify data output consistency through serial monitor or external interface.

7. Documentation:

- **Project Documentation:**

Detailed circuit diagram with component specifications.
Arduino sketch with comments explaining each section.
Testing procedures and results.
Troubleshooting guide for common issues.

- **User Manual:**

Instructions for assembly and setup.
Operating procedures.
Maintenance and troubleshooting tips.

8. Project Deliverables:

Fully assembled and tested Arduino radar detection system.
Project documentation including circuit diagram, Arduino code, and user manual.
Optional enhancements or features based on project scope and requirements.

HARDWARE USED:

Arduino nano: http://bit.ly/2QJB8z8
Ultrasonic sensor: http://bit.ly/2zNsVQH
Servo motor: https://amzn.to/2NRXXAy
Oled display : https://amzn.to/3pGssa7
Jumper wires: https://amzn.to/2JWSR44
Mini bread board: https://amzn.to/2NP2UKL

FLOW EXPLANATION:

1. Setting Up the Ultrasonic Sensor:

Connections:
Connect VCC of the ultrasonic sensor to 5V on Arduino.

Connect GND of the ultrasonic sensor to GND on Arduino.
Connect Trig pin of the ultrasonic sensor to a digital pin (e.g., D7) on Arduino.
Connect Echo pin of the ultrasonic sensor to another digital pin (e.g., D6) on Arduino.

2. Setting Up the Servo Motor:

Connections:

Connect VCC of the servo motor to 5V on Arduino.
Connect GND of the servo motor to GND on Arduino.
Connect the control pin of the servo motor to a PWM pin (e.g., D9) on Arduino.

3. Writing the Arduino Code:

Use the Arduino IDE to write the code for the radar detection project.
The code will involve:
Initializing the ultrasonic sensor and servo motor.
Continuously sweeping the servo motor from 0 to 180 degrees.
For each position of the servo motor, triggering the ultrasonic sensor to measure distances.
Calculating and storing these distances.

4. Processing the Distance Data:

Determine if an object is within the radar's field of view.

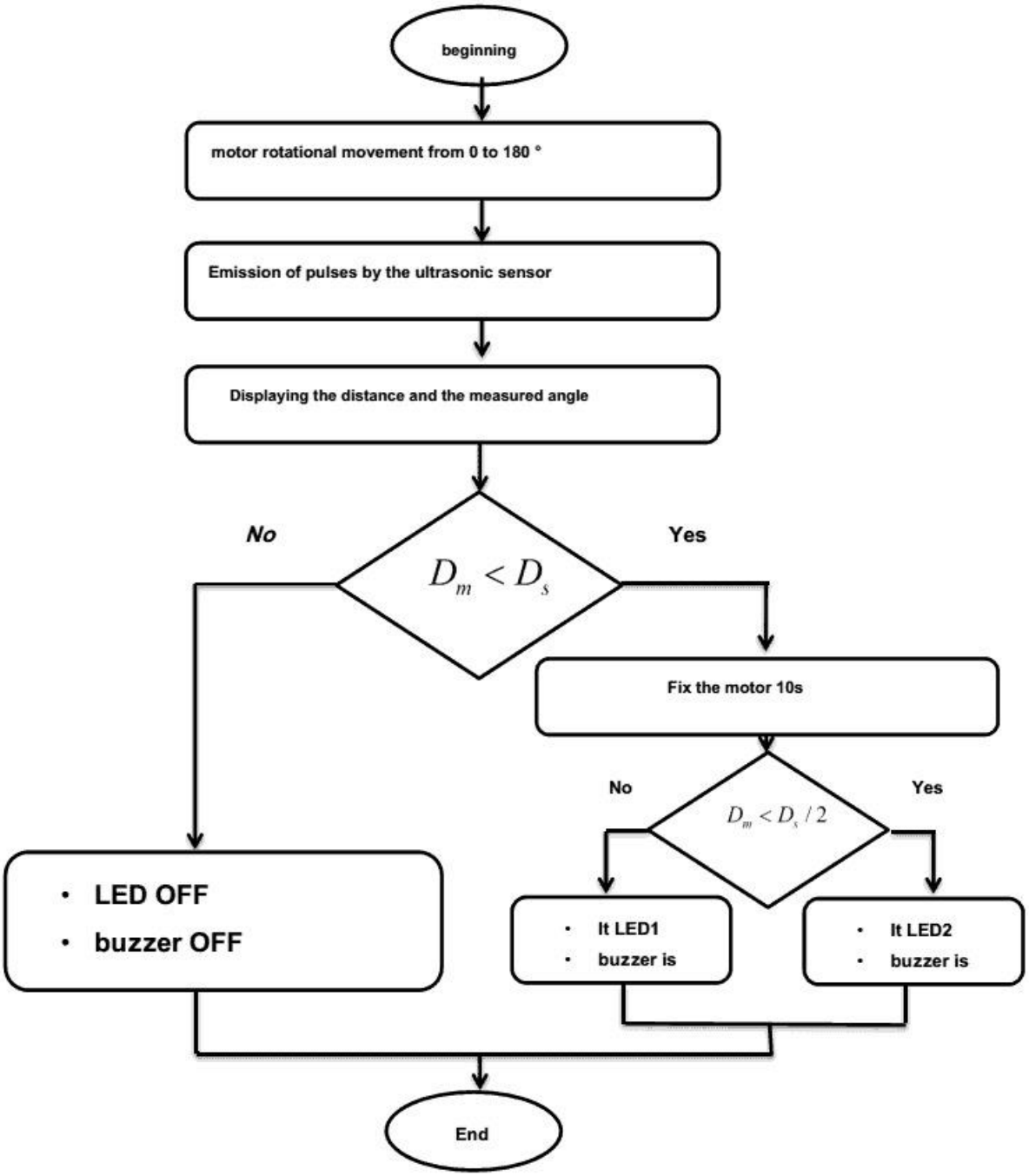
5. Display or Output:

You can output the detection information in various ways:
Serial monitor output for testing and debugging.
LED indicators or buzzer for immediate feedback.
Interface with a display (LCD or OLED) for visual feedback.

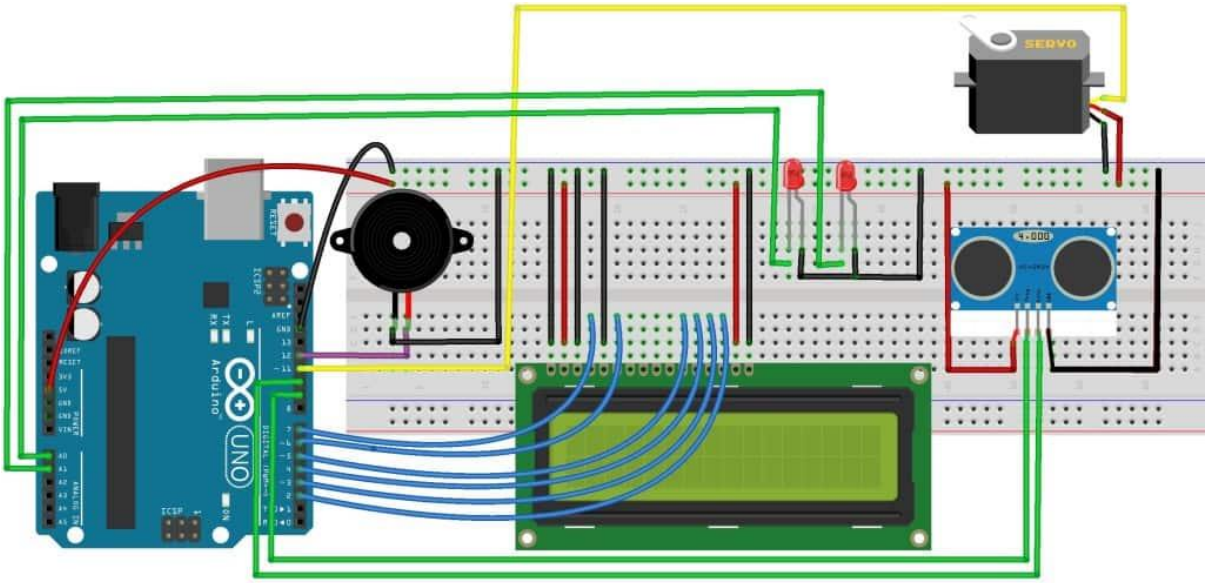
6. Fine-tuning and Testing:

Calibrate the servo motor sweep range and the detection thresholds based on your specific needs.
Test the project in different environments to ensure reliable detection.

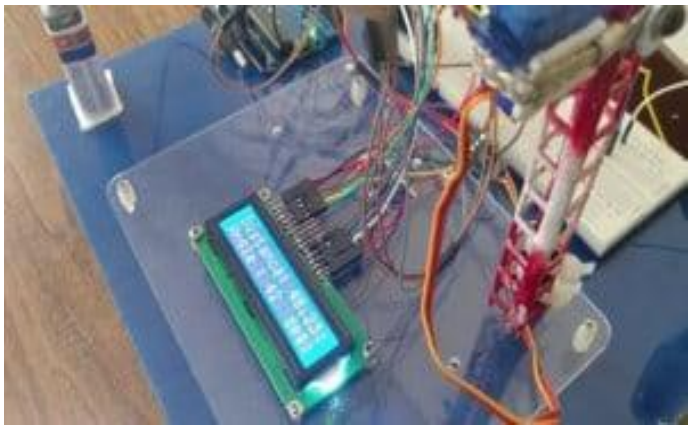
FLOWCHART:

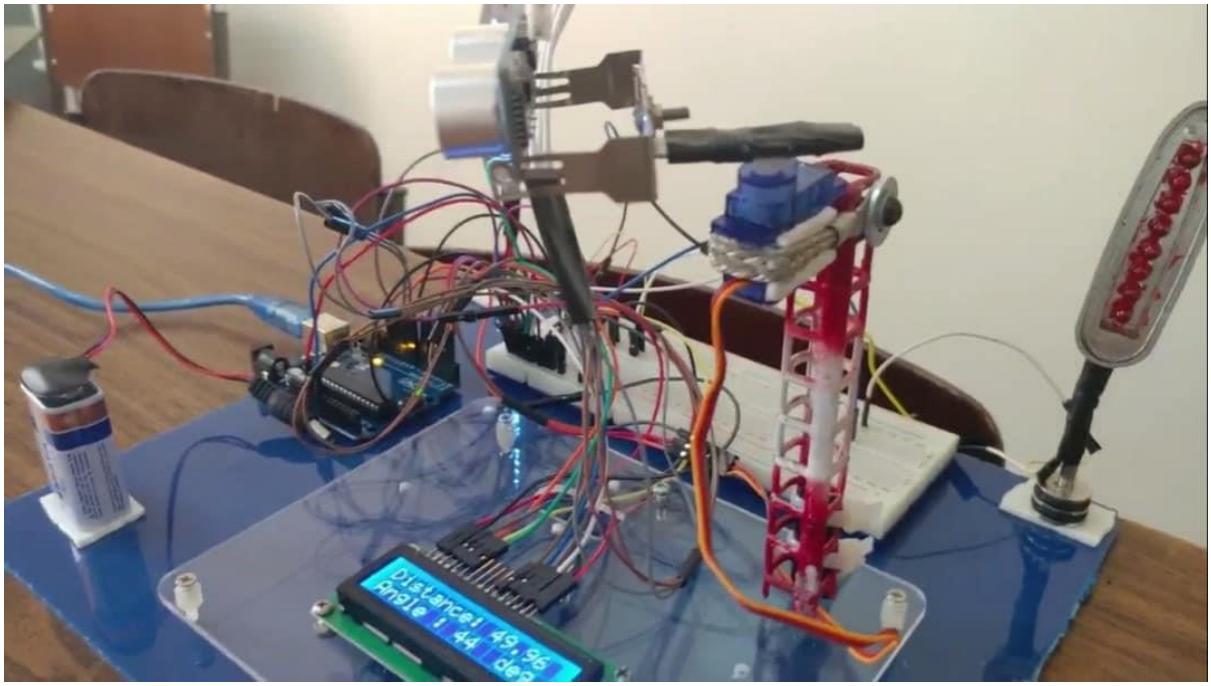

beginning
motor rotational movement from 0 to 180 °
Emission of pulses by the ultrasonic sensor
Displaying the distance and the measured angle
No
$D_m < D_s$
Yes
Fix the motor 10s
No
$D_m < D_s / 2$
Yes
• LED OFF
• buzzer OFF
• It LED1
• buzzer is
• It LED2
• buzzer is
End

WIRING DIAGRAM:



IMPLEMENTATION:



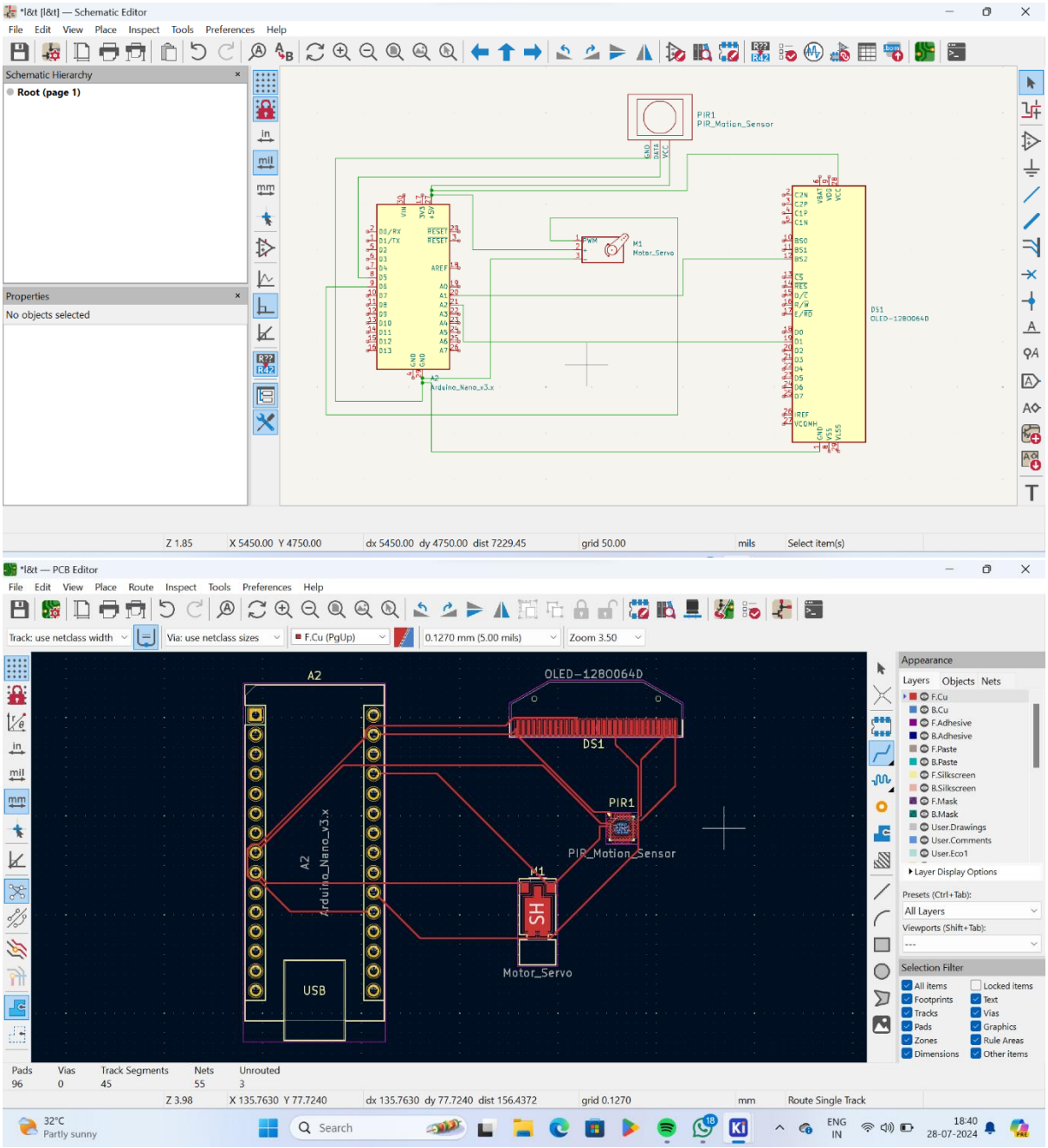


SOURCE CODE:

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

Servo myservo;
LiquidCrystal lcd(7, 6, 5, 4, 3, 2); // Creates an LCD object. Parameters: (rs, enable, d4, d5, d6, d7)

int pos = 0; // la position initiale du servo moteur
const int trigPin = 9;
const int echoPin = 10;
const int moteur = 11;
const int buzzer = 12;
const int ledPin1 = 14;
const int ledPin2 = 15;
float distanceCm, DistanceSec,duration;

void setup() {
myservo.attach(moteur); // attache le Servo moteur a la pin numéro 11
lcd.begin(16,2); // Initialiser l'interface de Lcd avec leurs Dimensions
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
pinMode(buzzer, OUTPUT);
pinMode(ledPin1, OUTPUT);
pinMode(ledPin2, OUTPUT);
DistanceSec=20;
```

```
}

void loop() {
for (pos = 0; pos <= 180; pos += 1) { // aller de 0 a 180 degée
// in steps of 1 degree
myservo.write(pos); // Programmer le Servo pour aller a la position (pos)
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH); //envoyer une impulsion de 10 micro seconds
delayMicroseconds(10);
digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);
distanceCm= duration*0.034/2;
if (distanceCm <= DistanceSec)
{

if(distanceCm <= DistanceSec/2)
{

tone(buzzer, 10); // Send 1KHz sound signal...
digitalWrite(ledPin1, LOW);
digitalWrite(ledPin2, HIGH);
delay(700);
noTone(buzzer); // Stop sound...
lcd.setCursor(0,0); // positionner le cursor a 0,0
lcd.print("Distance: "); // Printe "Distance" sur LCD
lcd.print(distanceCm); // Printe la valeur Obtenue sur LCD
lcd.print(" cm "); // Printe l'unité sur LCD
delay(10);
lcd.setCursor(0,1);
lcd.print("Angle : ");
lcd.print(pos);
lcd.print(" deg ");
delay(2000);
}
```

KICAD DESIGN:



COMPONENTS WORKING FUNCTIONALITY:

- **Arduino Board:** Acts as the central processing unit that controls all operations and interacts with other components.

- **Ultrasonic Sensor (HC-SR04):** Measures distance by sending ultrasonic waves and calculating the time it takes for the waves to bounce back. This helps in detecting objects and determining their distance from the sensor.

- **Servo Motor:** Rotates the ultrasonic sensor to sweep across a designated range. This allows the sensor to cover a wider area and detect objects in different directions.

- **LCD Display (optional):** Provides a visual output of detected objects, distances, or other relevant information.

- **Breadboard and Jumper Wires:** Used for connecting components and creating the circuit.

- **Power Supply:** Typically provided by a USB connection to a computer or a separate power source compatible with Arduino.

GITHUB LINK:

PROJECT OUTCOME:

1. **Object Detection and Distance Measurement**
2. **Real-Time Monitoring and Visualization**
3. **Obstacle Avoidance for Robotics**
4. **Education and Learning**
5. **Customization and Expansion**
6. **Skill Development**
7. **Community and Sharing**
8. **Demonstration and Prototyping**

APPLICATIONS:

Home Automation: Implementing a radar system for smart home applications, such as automatic door opening/closing based on detected presence.

Security Systems: Integrating radar for perimeter surveillance to detect unauthorized movements.

Environmental Monitoring: Using radar to monitor wildlife or detect intrusions in sensitive environments.

CONCLUSION:

An Arduino radar detection project not only demonstrates technical proficiency but also fosters a deeper understanding of sensor technologies and their applications. Whether for educational purposes, personal projects, or potential commercial applications, the project provides a solid foundation in electronics and programming while offering tangible outcomes in object

detection and distance measurement. It serves as an excellent platform for exploration, learning, and innovation in the realm of DIY electronics and beyond.