**DIY RADAR SYSTEM**

**Title:** Real-Time Obstacle Detection Radar Interface using Processing and Ultrasonic Sensor

**Objective:**

To build a visual radar system using an ultrasonic sensor and servo motor interfaced with a microcontroller (like Arduino), and visualize the real-time distance and angle of detected obstacles using the Processing IDE.

**Components Used:**

| **Component** | **Quantity** | **Description** |
| --- | --- | --- |
| Ultrasonic Sensor (HC-SR04) | 1 | For distance measurement |
| Servo Motor (SG90) | 1 | To rotate the sensor 0 to 180 degrees |
| Arduino Uno/ESP32 | 1 | Microcontroller to control everything |
| USB Cable | 1 | For communication with PC |
| Jumper Wires | As needed | For connections |

**Working Principle:**

1. Servo motor rotates the ultrasonic sensor from 0 to 180 degrees.
2. At each angle, the sensor measures the distance to the nearest obstacle.
3. Angle and distance are sent via Serial communication to the PC.
4. Processing software receives and visualizes the data as a radar sweep.

**Circuit connection:**

**(URL:-https://app.cirkitdesigner.com/project/feb29c64-f639-470a-a054-7464a828ee94)**

**Arduino Code:**

#include <Servo.h>

Servo myServo;

const int trigPin = 9;

const int echoPin = 10;

void setup() {

Serial.begin(9600);

myServo.attach(6);

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

}

void loop() {

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

myServo.write(angle);

delay(20);

long duration, distance;

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);

distance = duration \* 0.034 / 2;

Serial.print(angle);

Serial.print(",");

Serial.println(distance);

delay(20);

}

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

myServo.write(angle);

delay(20);

long duration, distance;

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);

distance = duration \* 0.034 / 2;

Serial.print(angle);

Serial.print(",");

Serial.println(distance);

delay(20);

}

}

**Processing Code:**

import processing.serial.\*;

Serial myPort;

float angle = 0;

float distance = 0;

void setup() {

size(1000, 800);

myPort = new Serial(this, Serial.list()[0], 9600);

myPort.bufferUntil('\n');

background(0);

stroke(0, 255, 0);

strokeWeight(1);

noFill();

}

void draw() {

background(0);

translate(width/2, height - 200);

drawGrid();

drawRadarSweep(angle, distance);

drawLiveCenterData(angle, distance);

drawText(angle, distance);

}

void drawGrid() {

stroke(0, 255, 0);

strokeWeight(2);

noFill();

int maxRadius = 300;

int step = 60;

for (int r = maxRadius; r >= step; r -= step) {

arc(0, 0, r\*2, r\*2, PI, TWO\_PI);

}

for (int a = 0; a <= 180; a += 30) {

float x = maxRadius \* cos(radians(a));

float y = -maxRadius \* sin(radians(a));

line(0, 0, x, y);

fill(0, 255, 0);

textAlign(CENTER, CENTER);

textSize(18);

text(a + "°", x \* 1.1, y \* 1.1);

}

for (int r = step; r <= maxRadius; r += step) {

fill(0, 255, 0);

textAlign(LEFT, CENTER);

textSize(16);

text(r + " cm", 10, -r + 10);

}

fill(0, 255, 0);

textSize(18);

textAlign(RIGHT);

text("Degrees →", maxRadius + 50, 30 - maxRadius);

textAlign(LEFT);

text("← Distance (cm)", -maxRadius - 90, 30 - maxRadius);

}

void drawRadarSweep(float a, float d) {

float maxDist = 300;

float dxFull = maxDist \* cos(radians(a));

float dyFull = -maxDist \* sin(radians(a));

float dxObstacle = d \* 3 \* cos(radians(a));

float dyObstacle = -d \* 3 \* sin(radians(a));

if (d > 0 && d <= 100) {

stroke(255, 0, 0);

strokeWeight(3);

line(0, 0, dxObstacle, dyObstacle);

stroke(0, 255, 0);

strokeWeight(2);

line(dxObstacle, dyObstacle, dxFull, dyFull);

fill(255, 0, 0);

noStroke();

ellipse(dxObstacle, dyObstacle, 12, 12);

fill(255);

textAlign(LEFT);

textSize(14);

text(int(d) + "cm @ " + int(a) + "°", dxObstacle + 10, dyObstacle - 10);

}

else {

stroke(0, 255, 0);

strokeWeight(3);

line(0, 0, dxFull, dyFull);

}

noStroke();

fill(0, 255, 0, 50);

beginShape();

vertex(0, 0);

for (float offset = -1; offset <= 1; offset += 0.2) {

float angleOffset = a + offset;

float x1 = 300 \* cos(radians(angleOffset));

float y1 = -300 \* sin(radians(angleOffset));

vertex(x1, y1);

}

endShape(CLOSE);

}

void drawText(float a, float d) {

fill(0, 255, 0);

textSize(24);

textAlign(LEFT);

text("Angle: " + int(a) + "°", -width / 2 + 30, -height + 50);

text("Distance: " + int(d) + " cm", -width / 2 + 30, -height + 90);

}

void drawLiveCenterData(float a, float d) {

fill(0, 255, 0);

textSize(26);

textAlign(CENTER);

text("Angle: " + int(a) + "°", 0, 40);

text("Distance: " + int(d) + " cm", 0, 70);

}

void serialEvent(Serial p) {

String data = p.readStringUntil('\n');

if (data != null) {

data = trim(data);

String[] values = split(data, ',');

if (values.length == 2) {

angle = constrain(float(values[0]), 0, 180);

distance = constrain(float(values[1]), 0, 100);

}

}

}

**Steps to Execute the Project:**

1. **Connect Components:**
   * Connect SG90 servo to pin 6 (PWM).
   * Connect HC-SR04 trigger to pin 9, echo to pin 10.
   * Power everything using 5V and GND.
2. **Upload Arduino Code:**
   * Use Arduino IDE to upload the sketch to your board.
3. **Run Processing Code:**
   * Open the provided Processing sketch in Processing IDE.
   * Ensure correct COM port is selected (Serial.list()[0] may need adjustment).
   * Run the sketch to see real-time radar sweep.
4. **Adjustments (if needed):**
   * Change Serial.list()[0] to match your port index.
   * Tune sweep speed and delay for accuracy.

**Output:**

* A radar-like green arc scanning left-to-right and back.
* When an object is detected:
  + A red line marks the object location.
  + A red dot is drawn.
  + Text label displays "distance cm @ angle°" near object.
* Real-time display of current angle and distance at top center and corner.

**Applications:**

* Robotics obstacle mapping.
* Short-range radar visualization.
* Education and demonstration of ultrasonic scanning systems.