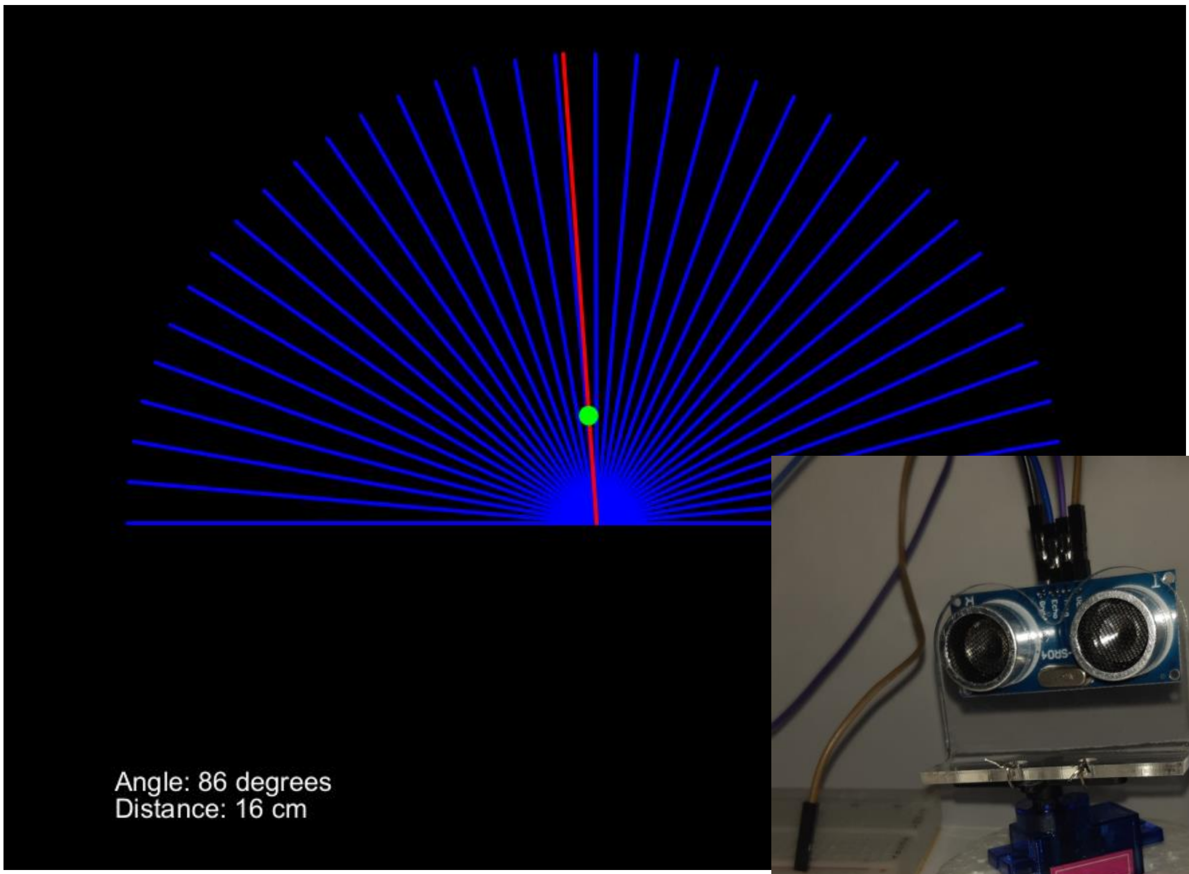
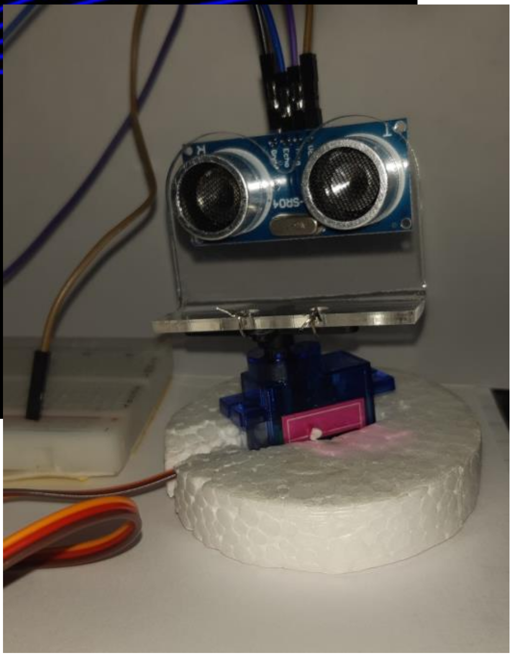
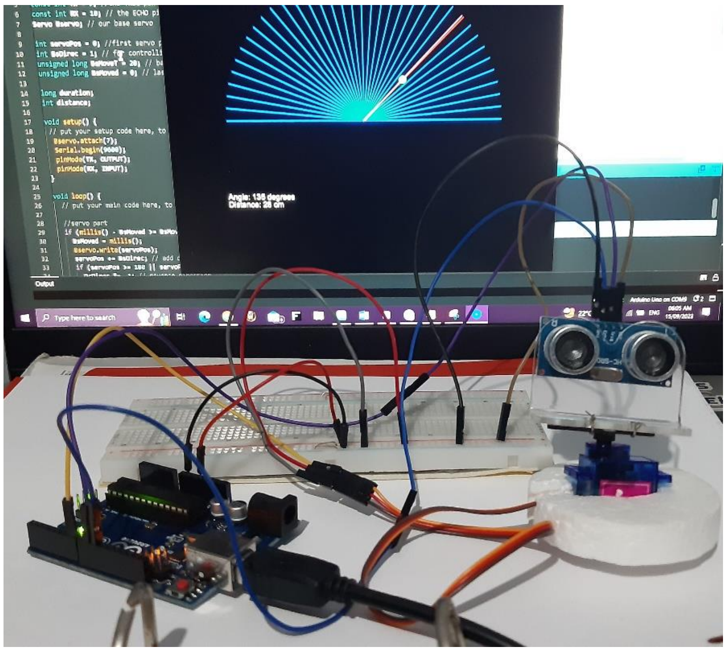
SONAR with arduino by issoupewd





SONAR with arduino by issoupewd

1. Introduction:



Sound Navigation and Ranging. It is a technique involving the use of propagation of sound for navigation, and it can be used for distant object detection and interaction with other objects.

Some may commonly refer to this project as a 'Radar', it is important to note that radar stands for Radio Detection and Ranging, uses radio frequency (RF) waves.

2. Resources:

In this project, you will need the Arduino IDE and the Processing Development Environment.

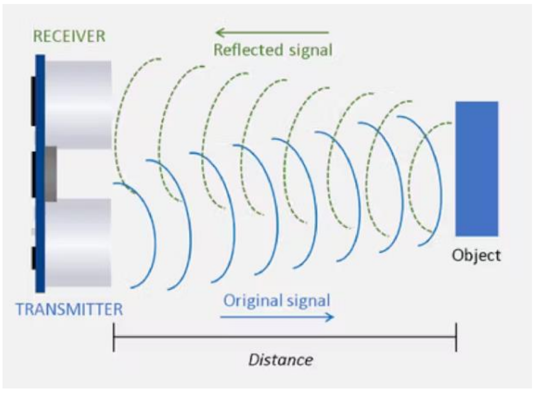
- Download Arduino: https://www.arduino.cc/en/software
- Download Processing: https://processing.org/download
- You can check the simulation of this project on Tinkercad: https://www.tinkercad.com/things/b7tNtF2Yhzf?sharecode=OXqOpS_5RJ6fwrNVpYZPAhQctfYCMyYtJTnrGhP1gCI

3. Components:

- Arduino
- Ultrasonic sensor HC-SR04
- Servo motor
- A breadboard or PCB and some jumper wires

3.1. Ultrasonic sensor HC-SR04:

Ultrasonic operate by emitting a sound wave at a frequency that is typically above the range of human hearing, often around 40 kHz. This sound wave travels through the air, and when it encounters an obstacle or object, it reflects or bounces back to the sensor. The ultrasonic sensor consists of two main components:



- **Transmitter** (Trig): This component emits the sound wave using piezoelectric crystals, which can generate sound when subjected to an electrical current.
- **Receiver** (Echo): This component detects the sound wave after it has travelled to the target and bounced back.

To calculate the distance between the sensor and the object, you can use the formula: D = ½ * T * V

- D represents the distance.
- T is the time it takes for the sound wave to travel to the object and back.
- V is the speed of sound, which is approximately 343 meters per second.

Lean to code it:

- You can find the time by using the following code:

```
duration =  pulseIn(RX,HIGH);
```

Here, RX is the pin connected to the Echo pin, and **pulseIn** measures the time it takes for a HIGH signal to go LOW, which corresponds to the sound wave traveling to the object and back.

- To calculate the distance, you can use this code:

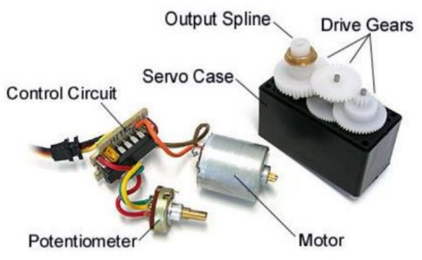
```
digitalWrite(TX, LOW);
delayMicroseconds(2);
digitalWrite(TX, HIGH);
delayMicroseconds(10);
digitalWrite(TX, LOW);
duration = pulseIn(RX, HIGH);
distance = 0.034*duration / 2;
```

TX is the pin connected to the Trig pin. This code sends a short pulse to trigger the sound wave and then measures the time it takes for the wave to bounce back. Finally, it calculates the distance by applying the formula mentioned earlier, taking half of the duration to account for the round trip of the sound wave.

3.2. Servomotor:

A servo motor is an electromechanical device that is used to precisely control the position, speed, and acceleration of an object, a servo motor consists of four main components:

- **DC Motor:** The servo motor contains a high-speed, low-torque DC motor.
- **Control Circuit:** The control circuit is responsible for controlling the motor, setting the angle of rotation.
- **Gears:** Gears are used to transform the high-speed rotation of the DC motor into slower but higher-torque rotation.
- **Potentiometer:** produces a voltage that corresponds to the absolute angle of the output shaft. This voltage is used by the control circuit to determine the servo's position.



Lean to code it:

- To move your servo to a specific angle you can use this code:

```
bservo.write(a);
```

Where **a** is the angle.

- To rotate a servo motor using an Arduino, you can use the Servo library. Here is an example of code that rotates a servo from 0 to 180 degrees and then back to 0 degrees while printing the angle to the serial monitor:

```
#include <Servo.h>
Servo bservo; // Declare your servo
```

```
void setup() {
bservo.attach(7); // Attach the servo to pin 7
Serial.begin(9600);
}
```

```
void loop() {
for (int a = 0; a <= 180; a++) {
bservo.write(a);
delay(30);
Serial.print("Angle:");
Serial.print(a);
Serial.println(" ");
}

for (int a = 180; a > 0; a--) {
bservo.write(a);
delay(30);
Serial.print("Angle:");
Serial.print(a);
Serial.println(" ");
}
}
```

In the loop function, we use two **for** loops to sweep the servo back and forth from 0 to 180 degrees adding 1 degree at each step, and pauses for the specified delay between each movement. While printing the angle to the serial monitor.

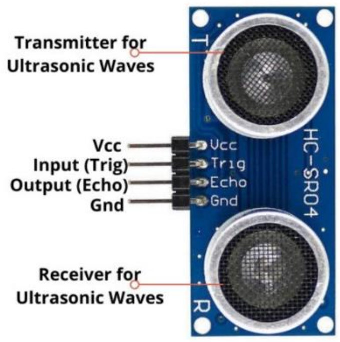
4. Circuit:

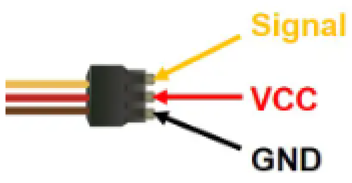
In the circuit setup, connect the Ultrasonic Sensor as follows:

- TRIG pin to Arduino pin 9
- ECHO pin to Arduino pin 10

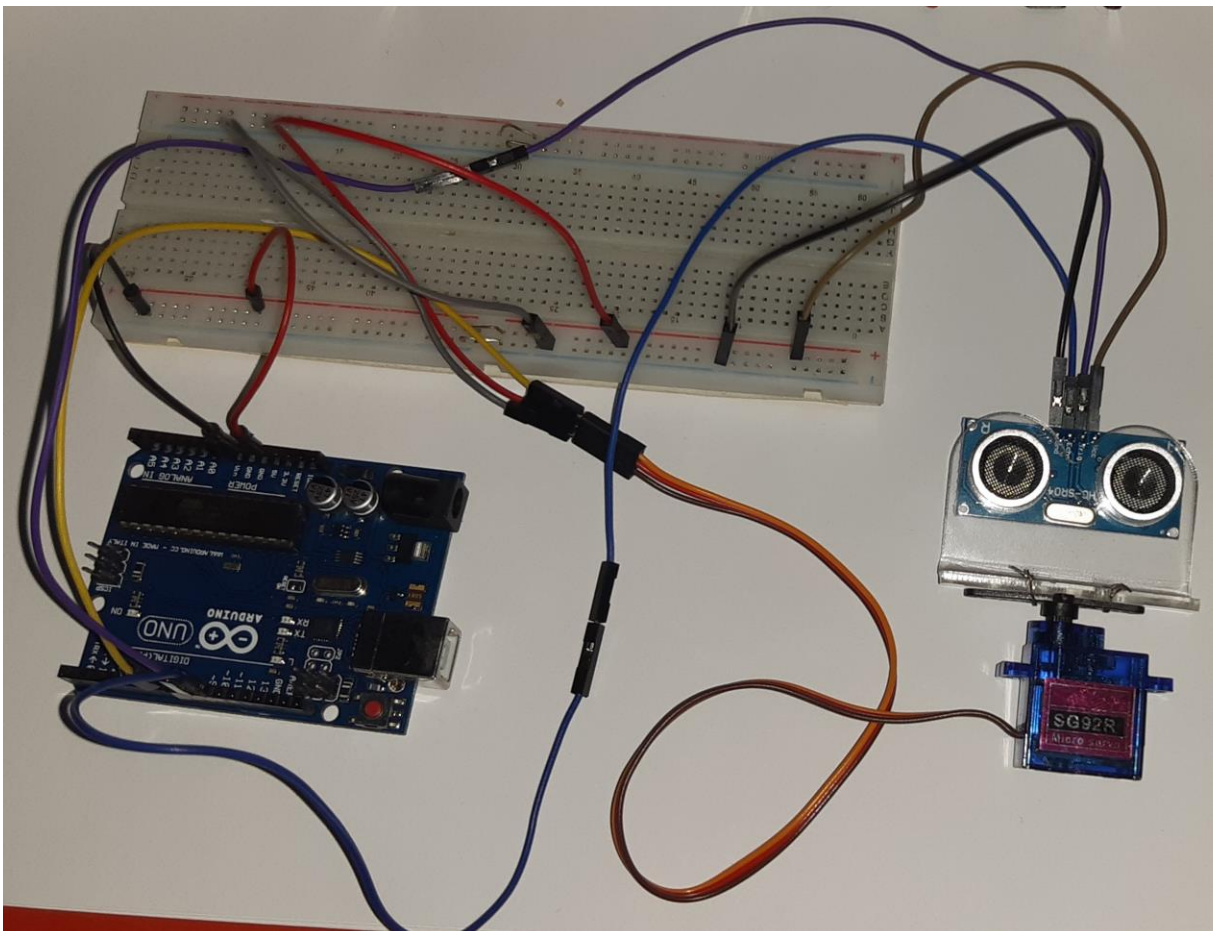
Connect the data pin of the servomotor (typically the orange wire) to Arduino pin 7.

Ensure that all Vcc pins are connected to the breadboard, which, in turn, is connected to the Arduino's 5V pin. Do not forget to connect the ground pins to create a complete circuit.

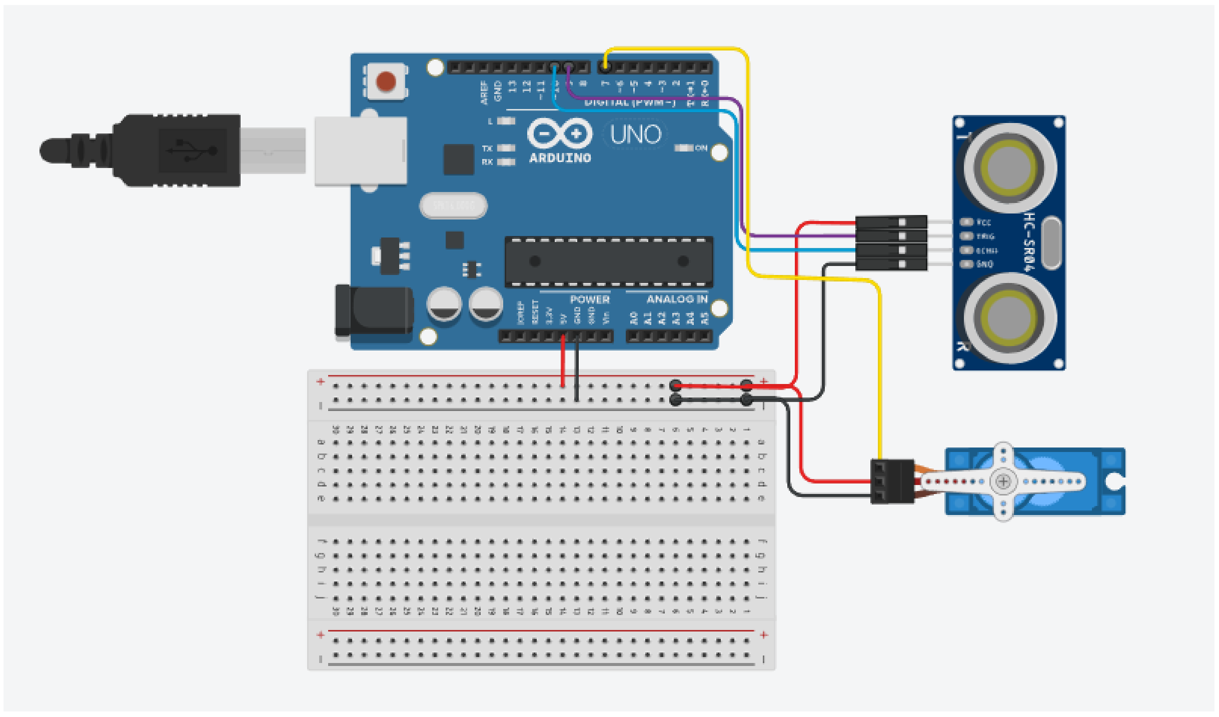

Transmitter for Ultrasonic Waves
Vcc
Input (Trig)
Output (Echo)
Gnd
Receiver for Ultrasonic Waves
T
Vcc
Trig
Echo
Gnd
HC-SR04
R



A real picture of the circuit:



The circuit schematic:



5. Source codes:

5.1. Arduino script:

Using Arduino IDE 2.0.4

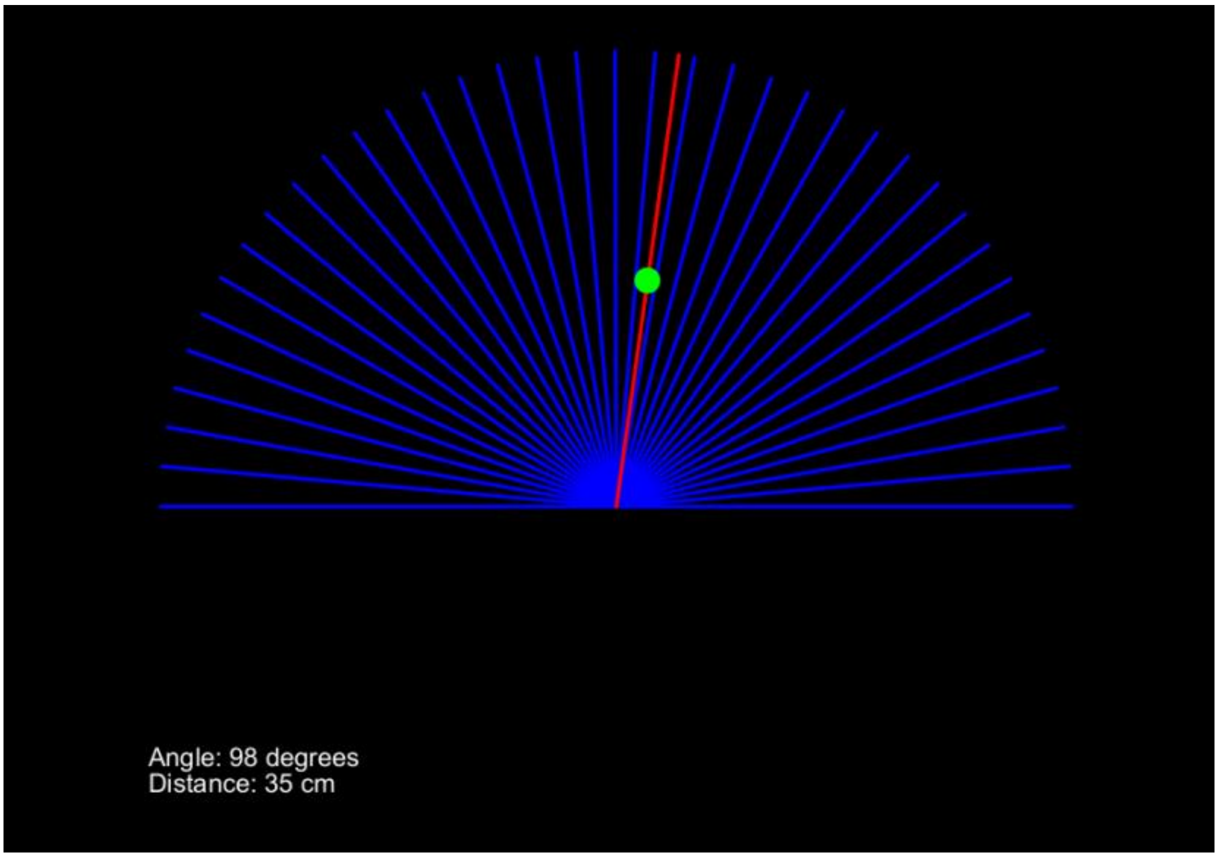
```
// SONAR : movement detection with ultrasonic sensor by issou : Arduino uno ,
servo motor , ultrason

#include <Servo.h>

const int TX = 9; //the TRIG pin
const int RX = 10; // the ECHO pin
Servo Bservo; // our base servo

int servoPos = 0; //first servo position
int BsDirec = 1; // for controlling base servo step and direction
unsigned long BsMoveT = 20; // base servo moving time , as a servo clock
unsigned long BsMoved = 0; // last move

long duration;
int distance;

void setup() {
 // put your setup code here, to run once:
  Bservo.attach(7);
  Serial.begin(9600);
  pinMode(TX, OUTPUT);
  pinMode(RX, INPUT);
}

void loop() {
  // put your main code here, to run repeatedly:

  //servo part
  if (millis() - BsMoved >= BsMoveT) {
    BsMoved = millis();
    Bservo.write(servoPos);
    servoPos += BsDirec; // add one degree
    if (servoPos >= 180 || servoPos <= 0) {
      BsDirec *= -1; // reverse direction
    }
  }

  //ultrason part
  digitalWrite(TX, LOW);
  delayMicroseconds(2);
  digitalWrite(TX, HIGH);
  delayMicroseconds(10);
  digitalWrite(TX, LOW);
  duration = pulseIn(RX, HIGH);
  distance = 0.034*duration / 2;

  //processing part
  Serial.print(distance);
  Serial.print(",");
  Serial.println(servoPos);

  delay(100);
}
```

When you upload the code, the servo will initially move to its starting position **servoPos**, which is 0 degrees. You can then check the Serial Monitor in the Arduino IDE for distance values. This allows you to verify that everything is functioning correctly.

5.2. Processing script:

To display the sonar graphics, we use processing to create a visual representation on the screen. This screen displays the servo position along with its angle. The red line indicates the current scanning or pointing direction of the sonar system. If an object is detected by the sonar system, it appears as a green dot on the screen, and the distance to the object is included as part of the display.

The sonar visual screen:



Using Processing 4.3:

Note, IMPORTANT:

In Arduino communication part, you must change the port COM9 to the port that you Arduino is using

```
import processing.serial.*;

PFont font;
Serial arduino;
int radius = 350; // Radius of the radar circle
int distance = 0;
int servoPos = 0;
color radarLineColor = color(255, 0, 0);
boolean objectDetected = false; //
float dotSize = 0;
float dotDistance = 0;

void setup() {
  size(1000, 800);
  font = createFont("Arial", 20);
  textFont(font);

  //Arduino communication
  String[] ports = Serial.list();
  for (String port : ports) {
    if (port.startsWith("COM9") || port.startsWith("/COM9")) { // Adjust based
on your system
      arduino = new Serial(this, port, 9600);
      break;
    }
  }
}

void draw() {
  background(0);
  translate(width / 2, height / 2);

  // Draw radar lines
  stroke(0, 0, 255);
  for (int i = 0; i <= 180; i += 5) {
    float rad = radians(i - 180);
    line(cos(rad) * radius, sin(rad)*radius, 0, 0);
  }

  // Read data from Arduino
  while (arduino.available() > 0) {
    String data = arduino.readStringUntil('\n');
    if (data != null) {
      data = trim(data);
      String[] values = split(data, ',');
      if (values.length == 2) {
        distance = int(values[0]);
        servoPos = int(values[1]);

        // Check if an object is detected
        objectDetected = (distance <= 70);
        if (objectDetected) {
          // Adjust the size of the red dot based on distance
          dotSize = map(distance, 0, 70, 10, 30);
          // Adjust the distance of the red dot from the center based on
distance
          dotDistance = map(distance, 0, 70, 0, radius);
        } else {
          dotSize = 0; //  no object detected
          dotDistance = 0; // Reset dot distance
        }
      }
    }
  }
  // Display angle and distance
  fill(255);
  textAlign(LEFT);
  text("Angle: " + servoPos + " degrees", -360, 200);
  text("Distance: " + distance + " cm", -360, 220);

  //the endpoint of the radar line
  float angle = radians(servoPos + 180);
  float x = radius * cos(angle);
  float y = radius * sin(angle);

  // radar line
  strokeWeight(3); // Thicker radar line
  stroke(radarLineColor);
  line(0,0, x, y);

  // red dot
  fill(0, 255, 0);
  noStroke();
  ellipse(dotDistance * cos(angle), dotDistance * sin(angle), dotSize,
dotSize);
}
```