Ultrasonic Radar with Arduino - Detailed Documentation

Overview

This project demonstrates how to create a simple ultrasonic radar system using an Arduino, an HC-SR04 ultrasonic sensor, and a servo motor. The radar scans a 180-degree area and detects objects within a specified range (up to 200 cm). The detected objects are displayed on a radar-like interface using the Processing IDE. The system is ideal for learning about sensor integration, servo control, and serial communication between Arduino and Processing.

Components and Supplies

- Arduino UNO - The microcontroller board used to control the system.
- HC-SR04 Ultrasonic Sensor - Measures the distance of objects.
- SG90 Micro-servo Motor - Rotates the ultrasonic sensor to scan the area.
- Jumper Wires - For connecting components.
- Breadboard - For easy prototyping (optional).
- Glue Gun - To temporarily attach the ultrasonic sensor to the servo motor.

Apps and Platforms

Arduino IDE - Used to program the Arduino.
Processing IDE - Used to visualize the radar interface.

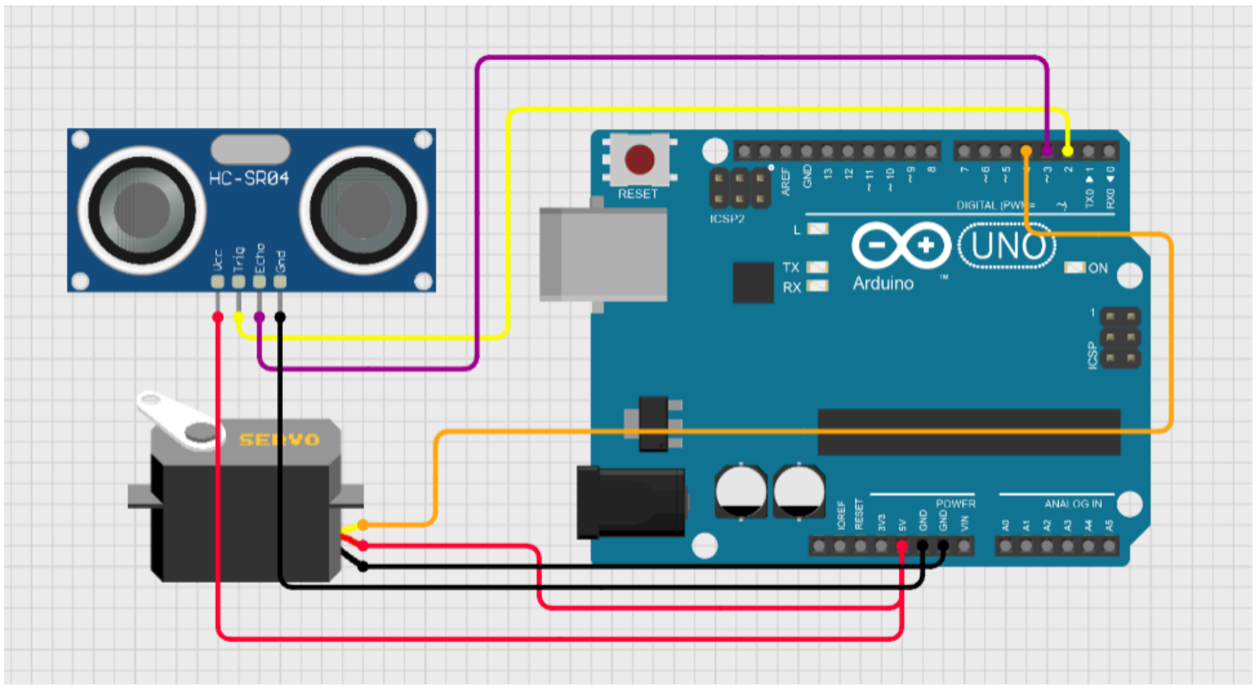
How It Works

- The ultrasonic sensor is mounted on a servo motor, which rotates from 15° to 165°.
- The sensor sends ultrasonic waves and measures the time taken for the echo to return, calculating the distance of any object in its path.
- The Arduino sends the angle and distance data to the Processing IDE via serial communication.
- Processing displays a radar-like interface, showing green lines for no detected objects and red lines for detected objects within a specified range (e.g., 40 cm).

Hardware Connections

Ultrasonic Sensor (HC-SR04)
VCC → 5V on Arduino
GND → GND on Arduino
TRIG → Digital Pin 2 on Arduino
ECHO → Digital Pin 3 on Arduino

Servo Motor (SG90)
Red Wire (VCC) → 5V on Arduino
Brown Wire (GND) → GND on Arduino
Orange Wire (Signal) → Digital Pin 4 on Arduino



Arduino Code

The Arduino code controls the servo motor and ultrasonic sensor, calculates distances, and sends data to the Processing IDE.

```
#include <Servo.h>

#define trigPin 2
#define echoPin 3

long duration;
int distance;

Servo myservo;

int calculateDistance() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = duration * 0.034 / 2;
  return distance;
}

void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  myservo.attach(4);
  Serial.begin(9600);
}

void loop() {
  for (int i = 15; i <= 165; i++) {
    myservo.write(i);
    delay(15);
    distance = calculateDistance();
    Serial.print(i);
    Serial.print(",");
    Serial.print(distance);
    Serial.print(".");
  }
  for (int i = 165; i >= 15; i--) {
    myservo.write(i);
    delay(15);
    distance = calculateDistance();
    Serial.print(i);
    Serial.print(",");
    Serial.print(distance);
    Serial.print(".");
  }
}
```

Processing Code

The Processing code visualizes the radar interface. It reads data from the Arduino and displays the radar with detected objects.

```
import processing.serial.*;
import java.awt.event.KeyEvent;
import java.io.IOException;

Serial myPort;
String angle = "";
String distance = "";
String data = "";
String noObject;
float pixsDistance;
int iAngle, iDistance;
int index1 = 0;
int index2 = 0;
PFont orcFont;

void setup() {
  size(1200, 700);
  smooth();
  myPort = new Serial(this, "COM7", 9600); // Change "COM7" to your
port
  myPort.bufferUntil('.');
}

void draw() {
  fill(98, 245, 31);
  noStroke();
  fill(0, 4);
  rect(0, 0, width, height - height * 0.065);
  fill(98, 245, 31);
  drawRadar();
  drawLine();
  drawObject();
  drawText();
}

void serialEvent(Serial myPort) {
  data = myPort.readStringUntil('.');
  data = data.substring(0, data.length() - 1);
  index1 = data.indexOf(",");
  angle = data.substring(0, index1);
  distance = data.substring(index1 + 1, data.length());
  iAngle = int(angle);
  iDistance = int(distance);
}

void drawRadar() {
  pushMatrix();
  translate(width / 2, height - height * 0.074);
  noFill();
  strokeWeight(2);
  stroke(98, 245, 31);
  arc(0, 0, (width - width * 0.0625), (width - width * 0.0625), PI,
TWO_PI);
  arc(0, 0, (width - width * 0.27), (width - width * 0.27), PI,
TWO_PI);
  arc(0, 0, (width - width * 0.479), (width - width * 0.479), PI,
TWO_PI);
  arc(0, 0, (width - width * 0.687), (width - width * 0.687), PI,
TWO_PI);
  line(-width / 2, 0, width / 2, 0);
  line(0, 0, (-width / 2) * cos(radians(30)), (-width / 2) *
sin(radians(30)));
  line(0, 0, (-width / 2) * cos(radians(60)), (-width / 2) *
sin(radians(60)));
  line(0, 0, (-width / 2) * cos(radians(90)), (-width / 2) *
sin(radians(90)));
  line(0, 0, (-width / 2) * cos(radians(120)), (-width / 2) *
sin(radians(120)));
  line(0, 0, (-width / 2) * cos(radians(150)), (-width / 2) *
sin(radians(150)));
  line((-width / 2) * cos(radians(30)), 0, width / 2, 0);
  popMatrix();
}

void drawObject() {
  pushMatrix();
  translate(width / 2, height - height * 0.074);
  strokeWeight(9);
  stroke(255, 10, 10);
  pixsDistance = iDistance * ((height - height * 0.1666) * 0.025);
  if (iDistance < 40) {
    line(pixsDistance * cos(radians(iAngle)), -pixsDistance *
sin(radians(iAngle)), (width - width * 0.505) * cos(radians(iAngle)),
-(width - width * 0.505) * sin(radians(iAngle)));
  }
  popMatrix();
}

void drawLine() {
  pushMatrix();
  strokeWeight(9);
  stroke(30, 250, 60);
  translate(width / 2, height - height * 0.074);
  line(0, 0, (height - height * 0.12) * cos(radians(iAngle)),
-(height - height * 0.12) * sin(radians(iAngle)));
  popMatrix();
}

void drawText() {
  pushMatrix();
  if (iDistance > 40) {
    noObject = "Out of Range";
  } else {
    noObject = "In Range";
  }
  fill(0, 0, 0);
  noStroke();
  rect(0, height - height * 0.0648, width, height);
  fill(98, 245, 31);
  textSize(25);
  text("10cm", width - width * 0.3854, height - height * 0.0833);
  text("20cm", width - width * 0.281, height - height * 0.0833);
  text("30cm", width - width * 0.177, height - height * 0.0833);
  text("40cm", width - width * 0.0729, height - height * 0.0833);
  textSize(40);
  text("N_Tech ", width - width * 0.875, height - height * 0.0277);
  text("Angle: " + iAngle + " ", width - width * 0.48, height -
height * 0.0277);
  text("Distance: ", width - width * 0.26, height - height * 0.0277);
  if (iDistance < 40) {
    text("        " + iDistance + " cm", width - width * 0.225,
height - height * 0.0277);
  }
  popMatrix();
}
```

Steps to Build the Project

Hardware Setup:

1. Connect the ultrasonic sensor and servo motor to the Arduino as described.
2. Attach the ultrasonic sensor to the servo motor using a glue gun.
3. Upload Arduino Code:
4. Open the Arduino IDE.
5. Copy and paste the provided Arduino code.
6. Upload the code to the Arduino.
7. Run Processing Code:
8. Open the Processing IDE.
9. Copy and paste the provided Processing code.
10. Change the serial port name (COM7) to the port your Arduino is connected to.
11. Run the code.

Test the System:

- Place objects within the sensor's range (up to 40 cm).
- Observe the radar interface for red lines indicating detected objects.

Troubleshooting

- No Data in Processing: Ensure the correct serial port is selected in the Processing code.
- Servo Not Rotating: Check the servo connections and ensure it is powered properly.
- Inaccurate Distance Measurements: Ensure the ultrasonic sensor is properly connected and there are no obstructions.

Conclusion

This project is a great way to learn about integrating sensors, servos, and serial communication. It can be extended by adding features like sound alerts, longer detection ranges, or integrating with other sensors. Have fun building and experimenting!

Download Links:

Arduino IDE: https://www.arduino.cc/en/software
Processing Software: https://processing.org/download