TITLE OF THE PROJECT: RADAR USING PROCESSING 3

SUBMITTED BY:

NAME	REGISTRATION NUMBER
TUNUGUNTLA SINDHU GAYATHRI	20MIS7007
PARRIPATI SAI PIYUSHA	20MIS7009
GUNDA PRAVEEN KUMAR	20MIS7021
YADLAPALLI SAI KIRAN	20MIS7050
KURRA VISHNU VARDHAN	20MIS7060
THOTA ANJALI SRI	21MIS7068

UNDER THE GUIDANCE OF Prof.Asish Kumar Dalai

VELLORE INSTITUTE OF TECHNOLOGY-AMARAVATI

ABSTRACT

Radar is an object detection system which uses radio waves to determine the range, altitude, direction, or speed of objects. The project is based on sonar technology as I will be using an ultrasonic sensor to determine the presence of any object in a particular range. It can be used to detect aircraft, ships, spacecraft, guided missiles, motor vehicles, weather formations, and terrain. The radar dish or antenna transmits pulses of radio waves or micro waves which bounce off any object in their path.The object returns a tiny part of the wave's energy to a dish or antenna which is usually located at the same site as the transmitter.

The modern uses of radar are highly diverse, including air traffic control, radar astronomy, air-defense systems, antimissile systems; marine radar start locate landmarks and other ships; aircraft anti-collision systems; ocean surveillance systems, outer space surveillance and rendezvous systems; meteorological precipitation monitoring; altimetry and flight control systems; guided missile target locating systems; and ground-penetrating radar for geological observations. High tech radar systems are associated with digital signal processing and are capable of extracting useful information from very high noise levels. The Arduino based project requires an ultrasonic sensor, the sensor released the waves which we want to measure the distance of an object. The microcontrollers of the Arduino board can be programmed using C and C++ languages. When a code is written in Arduino UNO IDE software and connected to the board through a USB cable, Arduino boards have lot of applications in the present-day scenario, so we have decided to do a small project on them.

INDEX

S.NO	CONTENTS	PAGE NO
1	INTRODUCTION	4
2	BACKGROUND	5-6
3	PROBLEM DEFINITION	7
4	OBJECTIVES	8
5	METHODOLOGY/PROCEDURE	09-15
6	RESULTS AND DISCUSSION	16-17
7	CONCLUSION AND FUTURE SCOPE	18
8	REFERENCES	19
9	CODES IN APPENDIX	20-25

INTRODUCTION

Defining Arduino:

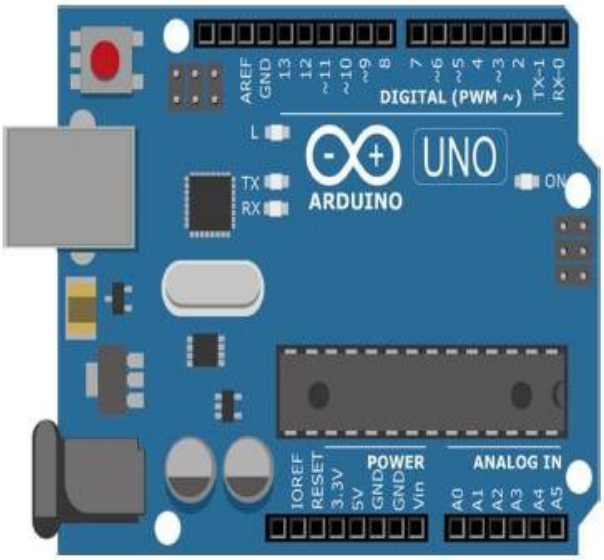
An Arduino is actually a microcontroller-based kit which can be either used directly by purchasing from the vendor or can be made at home using the components, owing to its open-source hardware feature. It is basically used in communications and in controlling or operating many devices.

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing.

BACKGROUND

Over the years Arduino has been the brain of thousands of projects, from everyday objects to complex scientific instruments. A worldwide community of makers - students, hobbyists, artists, programmers, and professionals - has gathered around this open-source platform, their contributions have added up to an incredible amount of accessible knowledge that can be of great help to novices and experts alike.

Arduino was born at the Ivrea Interaction Design Institute as an easy tool for fast prototyping, aimed at students without a background in electronics and programming. As soon as it reached a wider community, the Arduino board started changing to adapt to new needs and challenges, differentiating its offer from simple 8-bit boards to products for IoT applications, wearable, 3D printing, and embedded environments. All Arduino boards are completely open-source, empowering users to build them independently and eventually adapt them to their particular needs. The software, too, is opensource, and it is growing through the contributions of users worldwide.



How to program an Arduino?

The Arduino tool window consists of the toolbar with the buttons like verify, upload, new, open, save, serial monitor. It also consists of a text editor to write the code, a message area which displays the feedback like showing the errors, the text console which displays the output and a series of menus like the File, Edit, Tools menu. Thus, the code is uploaded by the bootloader onto the microcontroller.

ULTRASONIC SENSOR

As the name indicates, ultrasonic sensors measure distance by using ultrasonic waves. The sensor head emits an ultrasonic wave and receives the wave reflected back from the target. Ultrasonic Sensors measure the distance to the target by measuring the time between the emission and reception. An optical sensor has a transmitter and receiver, whereas an ultrasonic sensor uses a single ultrasonic element for both emission and reception. In a reflective model ultrasonic sensor, a single oscillator emits and receives ultrasonic waves alternately. This enables miniaturization of the sensor head.

Distance calculation

The distance can be calculated with the following formula:

Distance L = 1/2 × T × C

Where L is the distance, T is the time between the emission and reception, and C is the sonic speed.

(The value is multiplied by 1/2 because T is the time for go-and-return distance.)

Features

The following list shows typical characteristics enabled by the detection system. [Transparent object detectable]

Since ultrasonic waves can reflect off a glass or liquid surface and return to the sensor head, even transparent targets can be detected. [Resistant to mist and dirt]

Detection is not affected by accumulation of dust or dirt. [Complex shaped objects detectable]

Presence detection is stable even for targets such as mesh trays or springs.

SERVO MOTOR

A Servo Motor is a type of motor that can rotate with great precision. Normally this type of motor consists of a control circuit that provides feedback on the current position of the motor shaft, this feedback allows the servo motors to rotate with great precision. If you want to rotate an object at some specific angles or distance, then you use a servo motor. It is just made up of a simple motor which runs through a Servo Mechanism.

PROBLEM DEFINITION

How to make Arduino based ultrasonic radar. We could call it a Sonar because it is using sound, however, sonars are generally considered for use underwater. In this case, it is interesting that are used two sensors to scan the entire 360-degree space. Ultrasonic sensor modules are mounted on a servo motor that rotates between 0 and 180 °°. This method also simplifies the design because you don't have the problem of cables getting tangled around the servo. The process starts with a test of the servo rotation. First, it rotates a 0 to 180 and then 180 to 0 to check the correct movement of the headset. The process starts with a test of the servo rotation. First, it rotates a 0 to 180 and then 180 to 0 to check the correct movement of the headset. The process starts with a test of the servo rotation. First, it rotates a 0 to180 and then 180 to 0 to check the correct movement of the headset.

The device is extremely simple to build and consists of several elements:

- Arduino UNO Microcontroller
- Small 9g servo motor and two HC SR04 Ultrasonic modules

WORKING PRINCIPLE

At startup, need to test the movement of the servo motor. Immediately after that, the radar starts scanning. Next, we start the Processing application on the PC, and then we need to enter the correct serial port through which the Arduino communicates. This displays a Radar like monitor on which the objects are being monitored.

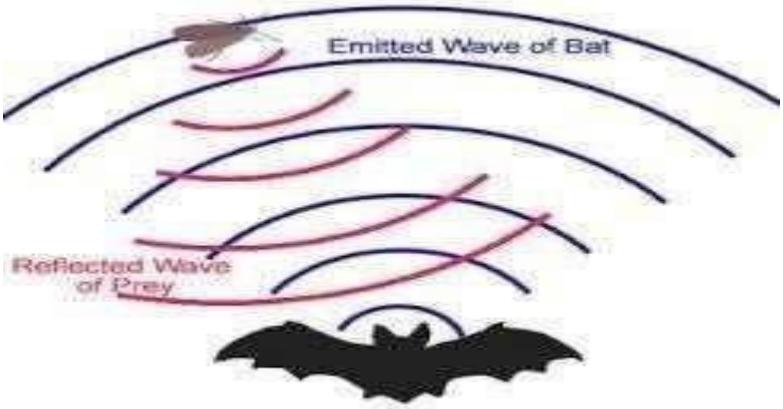
OBJECTIVES

- In order to eradicate the problems that are appeared in the vast and abandoned areas, like under the water.
- Similar to that in the forest areas and also in the border.
- To overcome this problem, we are using this approach using Arduino Uno.

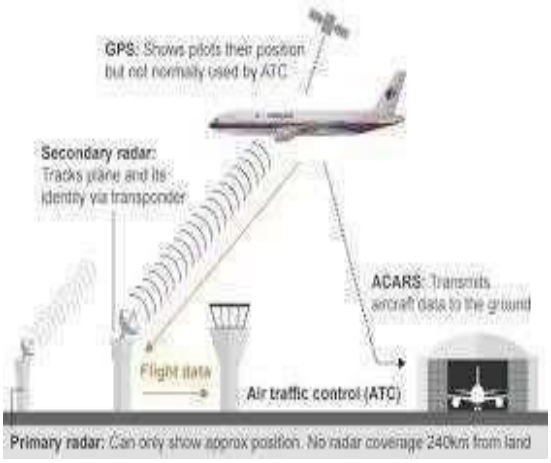
METHODOLOGY/PROCEDURE

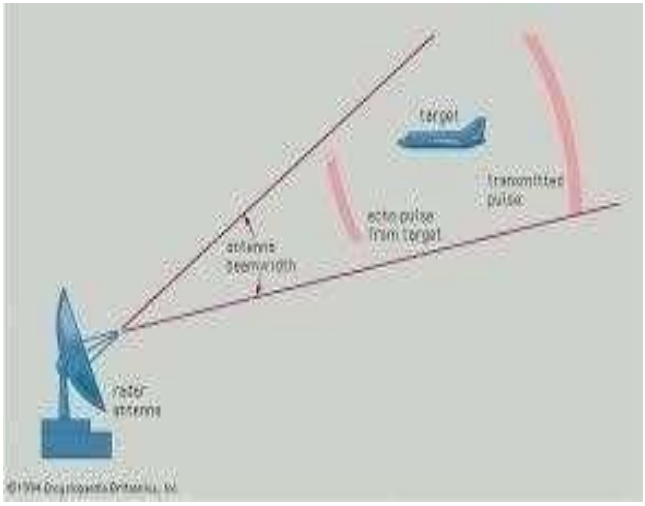
A radar system has a transmitter that emits radio waves called a radar signal in predetermined directions. When these come into contact with an object they are usually reflected or scattered in many directions Example: - let us take example for bat

Bat released the eco sound while travelling. if any object came in middle and it reflect back to the bat.



Here's a summary of how radar works:





• Magnetron generates high-frequency radio waves.

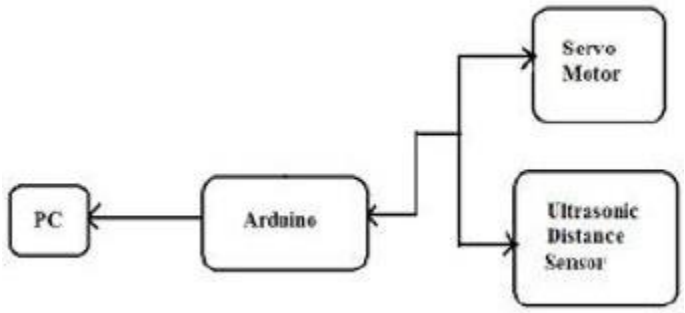
• Duplexer switches magnetron through to antenna.

• Antenna acts as transmitter, sending narrow beam of radio waves through the air.

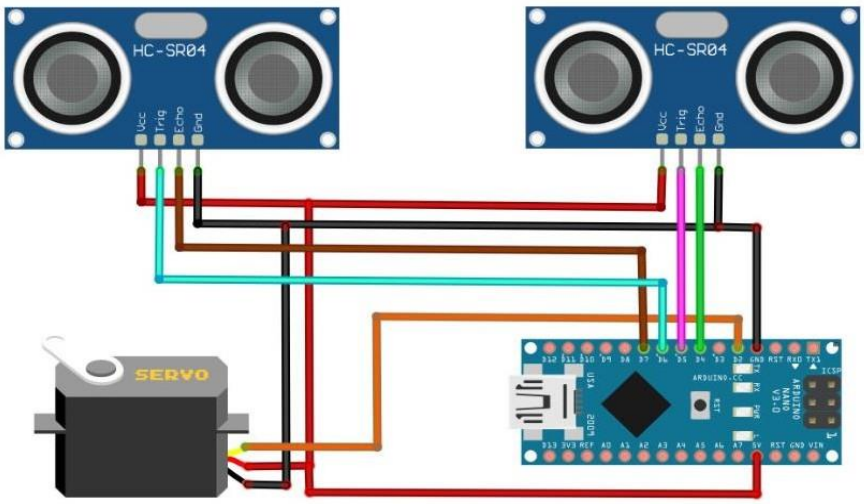
Radio waves hit enemy airplane and reflect back.

ARCHITECTURE OF THE PROJECT

BLOCK DIAGRAM:



CIRCUITDIAGRAM:



PROCEDURE

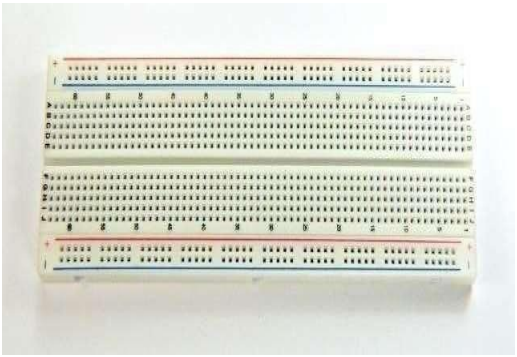
Components Required:

In this project we have used the Arduino and ultrasonic sensors along with the jumping wires and the relay motors and details list of the hard ware components are

- Arduino board and Arduino cable

- Jumper wires

- Bread board



- Ultrasonic sensor

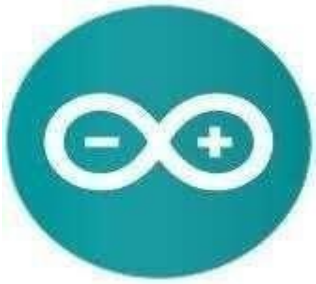
- Relay motor

- Double side plaster

- Gule gun

- Laptop

- Arduino software



- Processing software

WORKING

PRACTICAL IMPLEMENTATION:

A. Making on Arduino Board

Since, we believe in learning by doing. So, we decided to make our own Arduino board instead of using the readymade board. So, the steps required to make an Arduino board are as follows:

Boot-loading an Atmega328 using the Arduino board/AVR Programmer by uploading the boot loader to the Microcontroller. Making the connections on a general-purpose PCB, connecting the crystal oscillator, capacitors, connectors for the connections to Arduino board etc. Providing the power supply, usually 5 volts. Arduino is Ready to use. After you have done all this, then only the minimum circuitry like crystal oscillator, capacitors, connectors, power supply is required to complete the board. The same circuit can be made on the PCB, either designed or general purpose. Since, Arduino is an Open-Source. Hence, it is easy to make and can have any enhancements as per the requirements.

B. Connecting Servo Motor

A servomotor is a rotary actuator that allows for precise control of angular position, velocity and acceleration. A normal servo motor has three terminals:

1.VCC

2. GND

3. PULSE

A servo motor works at normally 4.8 to 6 volts. Ground is provided by connecting it to the Ground of the Arduino. The total time for a servo motor pulse is usually 20ms. To move it to one end of say 0-degree angle, a 1ms pulse is used and to move it to other end i.e., 180 degrees, a 2ms pulse is applied. Hence, according to this to move the axis of the servo motor to the center, a pulse of time 1.5 ms should be applied. For this, the pulse wire of the servo motor is connected to the Arduino that provides the digital pulses for pulse width modulation of the pulse. Hence, by programming for a particular pulse interval the servo motor can be controlled easily.

C. Connecting Ultrasonic Sensor: -

An Ultrasonic Sensor consists of three wires. One for Vcc, second for Ground and the third for pulse signal. The ultrasonic sensor is mounted on the servo motor and both of them further connected to the Arduino board. The ultrasonic sensor uses the reflection principle for its working. When connected to the Arduino, the Arduino provides the pulse signal to the ultrasonic sensor which then sends the ultrasonic wave in forward direction. Hence, whenever there is any obstacle detected or present in front, it reflects the waves which are received by the ultrasonic sensor. If detected, the signal is sent to the Arduino and hence to the PC/laptop to the processing software that shows the presence of the obstacle on the rotating RADAR screen with distance and the angle at which it has been detected.

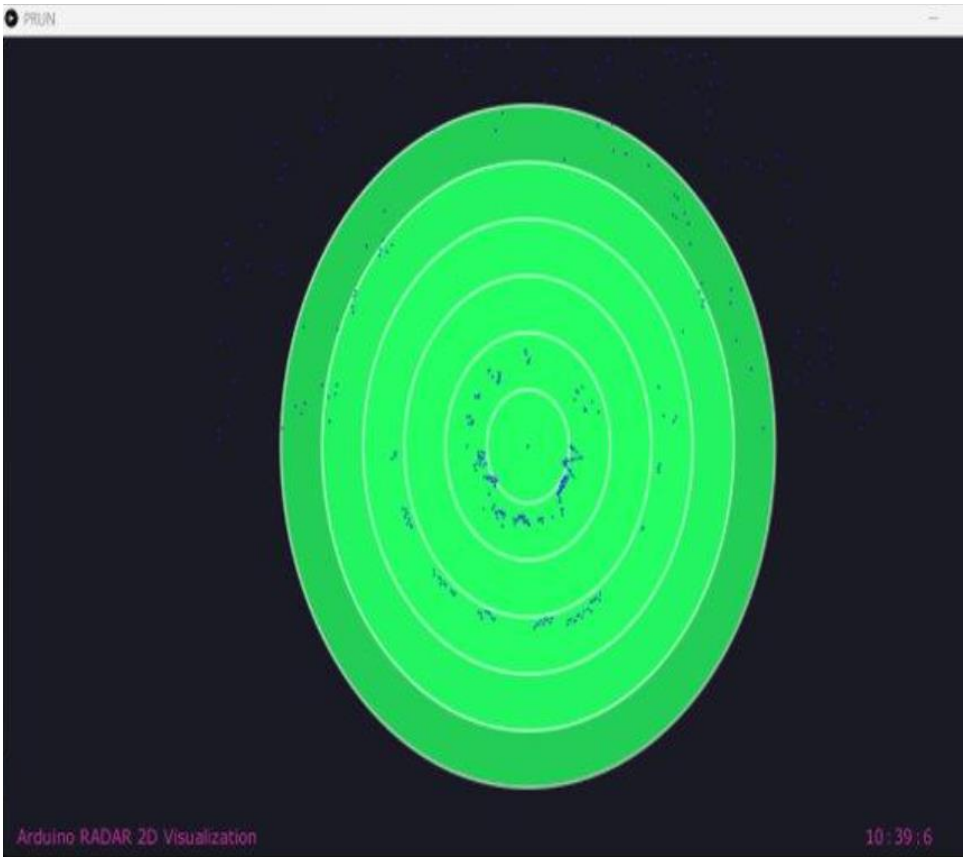
Using Processing Software

Processing is an open-source programming language and integrated development environment (IDE) built for the electronic arts, new media art, and visual design communities with the purpose of teaching the fundamentals of computer programming in a visual context, and to serve as the foundation for electronic sketchbooks. The project was initiated in 2001 by Casey Reas and Benjamin Fry, both formerly of the Aesthetics and Computation Group at the MIT Media Lab. One of the stated aims of Processing is to act as a tool to get non-programmers started with programming, through the instant gratification of visual feedback. The language builds on the Java language, but uses a simplified syntax and graphics programming models.

RESULTS AND DISCUSSION

OUTPUT:

We create a variable analog and assign it to 0. This is because the voltage value we are going to read is connected to the analog pin is A0. This voltage represents the voltage value falls across the resistor value we are measuring. Next, we create a variable name raw, which we will use to read in the analog voltage value. This later is our code get assigned to the analogue read () function.



THE BLUE COLOR INDICATES THE PRESENCE OF OBJECT

PROBLEMS FACED:

A. Making Own Arduino Board

The Arduino boards are available readily in the electronics market, but we decided to make our own Arduino board instead of buying one. So, the first problem was where to start to achieve this goal. Since, all parts on an Arduino board are SMD's, so we had to find a way to replace the SMD's with DIP IC's and also had to make an AVR programmer in order to pursue our further work. Hence, it took us some days to determine and plan our course of action. After that we had to boot load the AVR chip so as to make it compatible with the Arduino IDE software. Hence, we had to find a way to boot load the Arduino using the AVR programmer. It took us a long time to make the AVR programmer by researching on the type of communication and architecture of the AVR as it is not as same as 8051 microcontrollers.

B. Communicating with Arduino through PC

Another major problem related to the Arduino board was the communication with it from PC. Since, there is a requirement of an RS-232 to TTL conversion for the communication, so try some methods:

1. Firstly I used the MAX-232 IC to communicate with the Arduino as with the 8051 but due to large voltage drop and mismatch in the speed, it failed to communicate.

2. Next, I tried to use a dedicated AVR as USB to Serial converter as in the original Arduino board, the difference being DIP AVR used by us instead of the SMD Mega16U2 controller. But unfortunately, I was unable to communicate through it.

3. At last I had no other choice but to use the FTDI FT-232R chip for USB to Serial conversion. Finally, IT WORKED!!!

CONCLUSION AND FUTURE SCOPE

This project aims on the use of Ultrasonic Sensor by connected to the Arduino UNO R3 board and the signal from the sensor further provided to the screen formed on the laptop to measure the presence of any obstacle in front of the sensor as well as determine the range and angle at which the obstacle is detected by the sensor.

ADVANTAGES: -

1. The cost effective: our project below 1000rs only.

2. Improvised accuracy: The resistors with low value in milliohms are used in advanced cars with sensitive power steering and break circuits. Now a days these advancements have become the major cause for the severe accidents. Therefore, the components used in such circuits must have accurate and precise value for smooth working of such circuits. Ultimately this refers to the accurate testing of the resistors used. Improvised accuracy is thus the second primary aim of the sensor.

3. Reduced hardware complexity: Hardware complexity is one of the reasons for the high cost of the ultrasonic sensor. The use of Arduino Uno is to reduce the motherboard present in the conventional ohmmeter in Arduino based ultrasonic sensor. The Arduino acts as the central board. Since Arduino are readily available in market it leads to the reduction in the complexity of the design. The automated range selection is also the objective in order to speed up the testing process. This will also reduce the faults in range selection in manually operated conventional sensor.

APPLICATIONS:

- Military Security.
- Navy.
- Airlines or Air Traffic Control.
- Remote Sensing Environment.

The development of the radar technology took place during the World War II in which it was used for detecting the approaching aircraft and then later for many other purposes which finally led to the development of advanced military radars being used these days. Military radars have a highly specialized design to be highly mobile and easily transportable, by air as well as ground. Military radar should be an early warning, altering along with weapon control functions. It is specially designed to be highly mobile and should be such that it can be deployed within minutes.

REFERENCES

[1] http://www.arduino.cc/

[2] http://www.arduinoproducts .cc/

[3] http://www.atmel.com/atmega328/

[4] http://en.wikipedia.org/wiki/File:16MHZ_Crystal.jpg

[5] http://arduino.cc/en/Tutorial/BarGraph/

[6] http//: www.sproboticworks.com/ic%20pin%20configuration/7805/Pinout.jpg/

[7] http://www.sproboticworks.com/ic%ultrasonicsensor%20pinout.jpg

[8] http://www.instructables.com/id/ ATMega328-using-Arduino-/

[9] http://www.motherjones.com/files/blog_google_driverless_car.jpg

[10] http://www.google.co.in/imgres/Radar_antenna.jpg&w=546&h=697&ei=wuuK

CODES IN APPENDIX

ARDUINO CODE:

```
#include <HCSR04.h>
#include <Servo.h>

UltraSonicDistanceSensor distanceSensor(6, 7);            //Create the 1st sensor object
UltraSonicDistanceSensor distanceSensor2(5, 4);           //Create the 2nd sensor object
Servo servoMotor;          //Create the Servo object

int delayTime = 5;          //Delay value to wait for the servo to reach the 1 angle difference4f

long d;               //Distance from 1st sensor calculated
long d2;              //Distance from 2nd sensor calculated

void setup() {
  Serial.begin(9600);          //Initialize the Serial communication at 9600 bauds

  servoMotor.attach(2);        //Attach the servo to the Digital PWM pin 2
  servoMotor.write(180);       //Rotate the servo to the 180?
  delay(1000);             //Wait for the servo to reach 180?
  servoMotor.write(0);         //Rotate the servo to the 0?
  delay(1000);             //Wait for the servo to reach 0?

}

void loop() {
  for (int i = 1; i < 180; i++) {  //Move the Servo 180 degrees forward
```

```
    readSensors();            //Read the sensors
    Serial.print(i);          //Print the angle
    Serial.print(",");        //Print a ","
    Serial.print(d);          //Print the 1st distance
    Serial.print(",");        //Print a ","
    Serial.println(d2);       //Print the 2nd distance with end line
    servoMotor.write(i);      //Set the sensor at the angle
    delay(delayTime);         //Wait for the servo to reach i?
  }
  for (int i = 180; i > 1; i--) {  //Move the Servo 180 degrees backward
    readSensors();            //Read the sensors
    Serial.print(i);          //Print the angle
    Serial.print(",");        //Print a ","
    Serial.print(d);          //Print the 1st distance
    Serial.print(",");        //Print a ","
    Serial.println(d2);       //Print the 2nd distance with end line
    servoMotor.write(i);      //Set the sensor at the angle
    delay(delayTime);         //Wait for the servo to reach i?
  }
}

void readSensors() {
  d = distanceSensor.measureDistanceCm();
  d2 = distanceSensor2.measureDistanceCm();
}
```

PROCESSOR CODE:

```
import processing.serial.*;
```

```
import static javax.swing.JOptionPane.*;

Serial myPort;        // The serial port
String serialin;
int data[] = new int[360];
PFont f;

final boolean debug = true;

void setup() {
  String COMx, COMlist = "";
  size(1280, 720);
  f = createFont("Verdana", 32, true); // Arial, 16 point, anti-aliasing on
  textFont(f, 20);
  frameRate(60);
  for (int i = 0; i < 360; i++) {
    data[i] = 0;
  }
  try {
    if (debug) printArray(Serial.list());
    int i = Serial.list().length;
    if (i != 0) {
      if (i >= 2) {
        // need to check which port the inst uses -
        // for now we'll just let the user decide
        for (int j = 0; j < i; ) {
          COMlist += char(j+'a') + " = " + Serial.list()[j];
          if (++j < i) COMlist += ", ";
```

```
      }
      COMx = showInputDialog("Which COM port is correct? (a,b,..):\n"+COMlist);
      if (COMx == null) exit();
      if (COMx.isEmpty()) exit();
      i = int(COMx.toLowerCase().charAt(0) - 'a') + 1;
     }
     String portName = Serial.list()[i-1];
     if (debug) println(portName);
     myPort = new Serial(this, portName, 9600); // change baud rate to your liking
     myPort.bufferUntil('\n'); // buffer until CR/LF appears, but not required..
   } else {
     showMessageDialog(frame, "Device is not connected to the PC");
     exit();
   }
  }
  catch (Exception e)
  { //Print the type of error
   showMessageDialog(frame, "COM port is not available (may\nbe in use by another program)");
   println("Error:", e);
   exit();
  }
}


void draw() {
  background(26, 26, 36, 200);
  textSize(18);
```

```
  stroke(255, 255, 255, 150);
  fill(255, 50, 200, 200);
  text("Arduino RADAR 2D Visualization", 20, 710);
  text(hour(), 1050, 710);
  text(":", 1075, 710);
  text(minute(), 1085, 710);
  text(":", 1110, 710);
  text(second(), 1120, 710);
  fill(36, 255, 100, 200);
  strokeWeight(3);
  circle(640, 360, 600);
  circle(640, 360, 500);
  circle(640, 360, 400);
  circle(640, 360, 300);
  circle(640, 360, 200);
  circle(640, 360, 100);

  for (int i = 0; i < 360; i++) {
    fill(255, 10, 255, 200);
    stroke(50, 10, 255, 150);
    point(float(640) +  (map_values(data[i]))*cos(radians(i)), float(360) +
(map_values(data[i]))*sin(radians(i)));
  }

  while (myPort.available() > 0) {
    serialin = myPort.readStringUntil(10);
    try {
      String serialdata[] = splitTokens(serialin, ",");
```

```
      if (serialdata[0] != null) {
        serialdata[0] = trim(serialdata[0]);
        serialdata[1] = trim(serialdata[1]);
        serialdata[2] = trim(serialdata[2]);

        int i = int(serialdata[0]);
        data[179-i] = int(serialdata[1]);
        data[(179-i)+180] = int(serialdata[2]);
      }
    }
    catch (java.lang.RuntimeException e) {
    }
  }
}

float map_values(float x) {
  float in_min = 0, in_max = 200, out_min = 0, out_max = 700;
  return (x - in_min) * (out_max - out_min) / (in_max - in_min) + out_min;
}
```