Abstract :

A rangefinder is a device that measures the distance from the target to the observer, for the purposes of surveying, determining focus in photography, or accurately aiming a weapon [1]. In this technical project, we make a simple radar using the ultrasonic sensor, this radar works by measuring a range from 3cm to 40 cm as non-contact distance, with angle range between 15˚ and 165˚.The movement of the sensor is controlled by using a small servo motor. Information received from the sensor will be used by "Processing Development Environment" software to illustrate the result on a PC screen.

Introdution :

Radar is an object detection system that uses electromagnetic waves to identify range, altitude, direction, or speed of both moving and fixed objects such as aircraft, ships, vehicles, weather formations, and terrain. When we use ultrasonic waves instead of electromagnetic waves, we call it ultrasonic radar [2].

The main components in any ultrasonic radar are the ultrasonic Sensors. Ultrasonic sensors work on a principle similar to radar or sonar which evaluates attributes of a target by interpreting the echoes from radio or sound waves respectively.

Radar's information will appear in different ways. Basic and old radar station used sound alarm or LED, modern radar uses LCD display to show detailed information of the targeted object. We use Computer screen to show the information (distance and angle).

Fig(1).Google auto-drive car

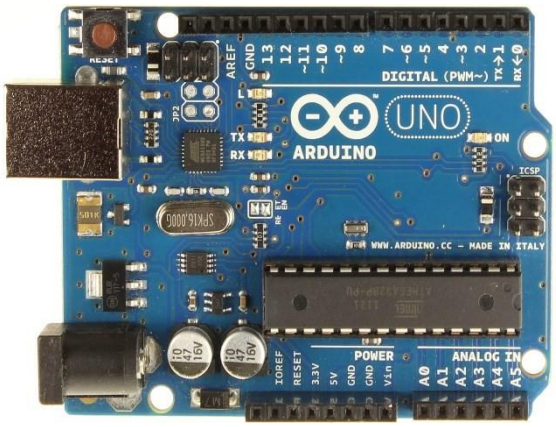
2-Tools

2-1 Arduino Board UNO Model :

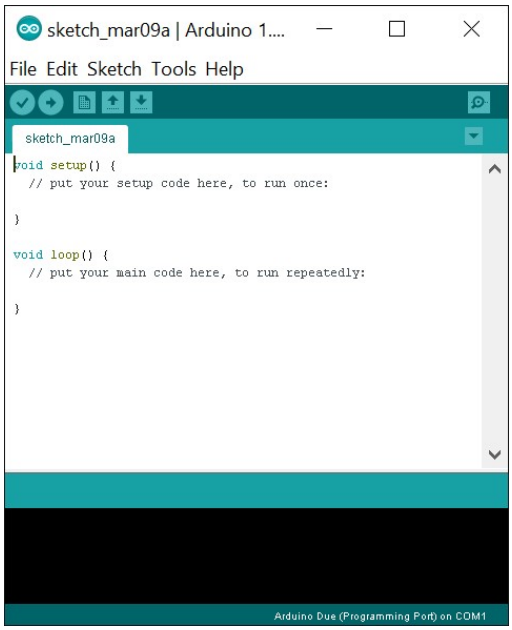
Arduino is a hardware and software company, project, and user community that designs and manufactures computer open-source hardware, open-source software, and microcontroller-based kits for building digital devices and interactive objects that can sense and control physical devices.

The project is based on microcontroller board designs. The board provides sets of digital and analog Input/output (I/O) pins that can interface to various expansion boards (termed shields) and other circuits Fig (2-1). The boards feature serial communication interfaces, including Universal Serial Bus (USB) on UNO model, for loading programs from personal computers [3].

For programming the microcontrollers, the Arduino project provides an integrated development environment (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software. This software can be used with any Arduino board Fig (2-2).



Fig(2.2) .Arduino UNO



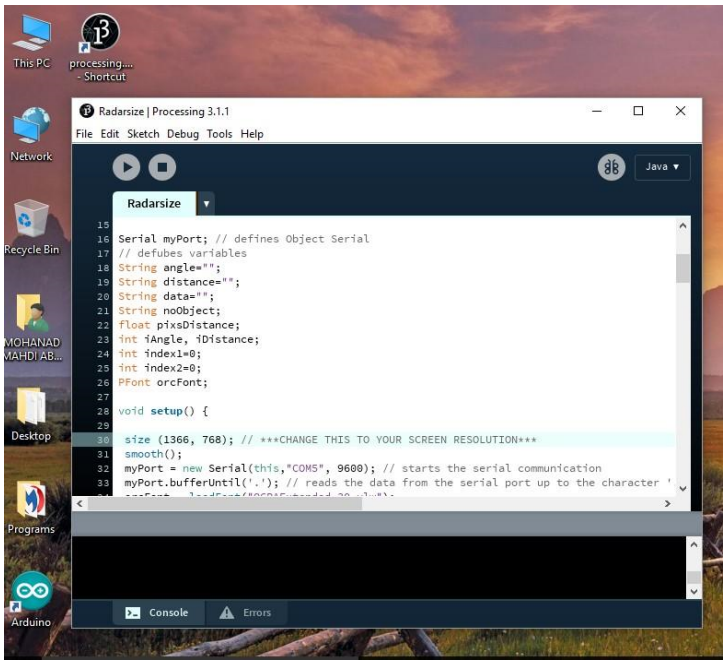
Fig(2.2). IDE Software.

Processing :

Processing is an open source computer 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 Fig (3).

❖ Specifications

- Free to download and open source
- Interactive programs with 2D, 3D or PDF output
- OpenGL integration for accelerated 2D and 3D
- For GNU/Linux, Mac OS X, and Windows
- Over 100 libraries extend the core software
- Well documented, with many books available



Fig(3). Software and processing

Ultrasonic sensors HC- SR04 :

Ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules include ultrasonic transmitters, receiver, and control circuit, within measuring angle 15 degrees Fig (4). [4-1]

❖ The basic principle of work. Fig (4.2)

(1) Using IO trigger for at least 10us high-level signal,

(2) The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.

(3) IF the signal back, through high level, time of high output IO duration is the time from sending ultrasonic to returning.

Test distance = (high level time × velocity of sound (340M/S) / 2.

❖ Wire connecting directly as following:

- 5V Supply
- Trigger Pulse Input
- Echo Pulse Output
- 0V Ground

4.1 UltrasonicHC- SR04

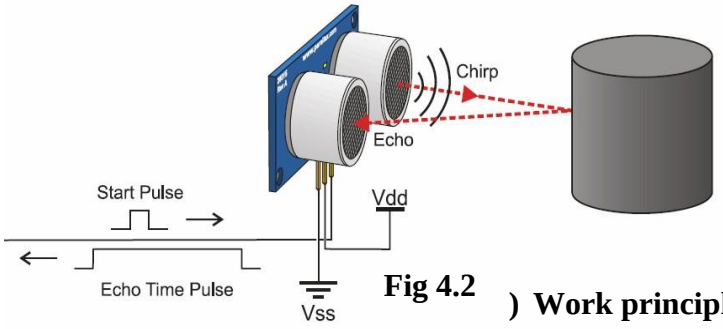


Fig 4.2) Work principle.

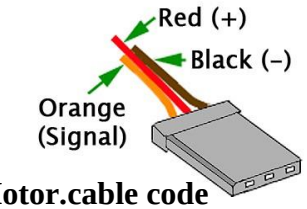
Servo Motor tower pro micro servo 9g :

Tiny and lightweight with high output power. The servo can rotate approximately 180 degrees (90 in each direction), and works just like the standard kinds but smaller Fig (5). You can use any servo code, hardware or library to control these servos.[5]

❖ **Specifications**

- Weight: 9 g
- Dimension: 22.2 x 11.8 x 31 mm approx.
- Stall torque: 1.8 kg f cm
- Operating speed: 0.1 s/60 degree
- Operating voltage: 4.8 V (~5V)
- Temperature range: 0 ºC – 55 ºC

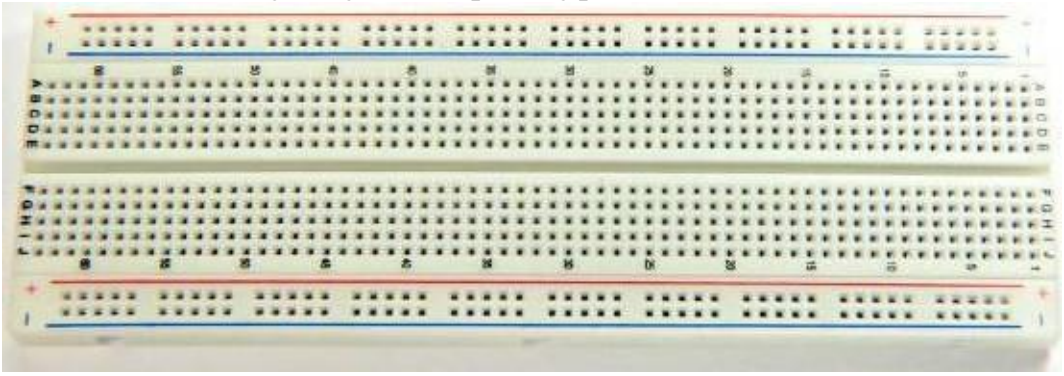
Fig(5).ServoMotor.Tower Pro Micro Servo 9g



Fig( )ServoMotor.cable code

2-5 Breadboards :

A breadboard is a construction base for prototyping of electronics. Originally it was literally a bread board, a polished piece of wood used for slicing bread. In the 1970s the solderless breadboard (AKA plug board, a terminal array board) became available and nowadays the term "breadboard" is commonly used to refer to these.
"Breadboard" is also a synonym for "prototype".



Fig(7) Breadboard

3-Work procedure :

❖ Components needed for this Project

- Arduino Board UNO Model.
- Processing software.
- Ultrasonic sensor HC- SR04.
- Servo Motor tower pro micro servo 9g.
- Breadboard and Jump Wires.

3-1 Circuit Diagram :

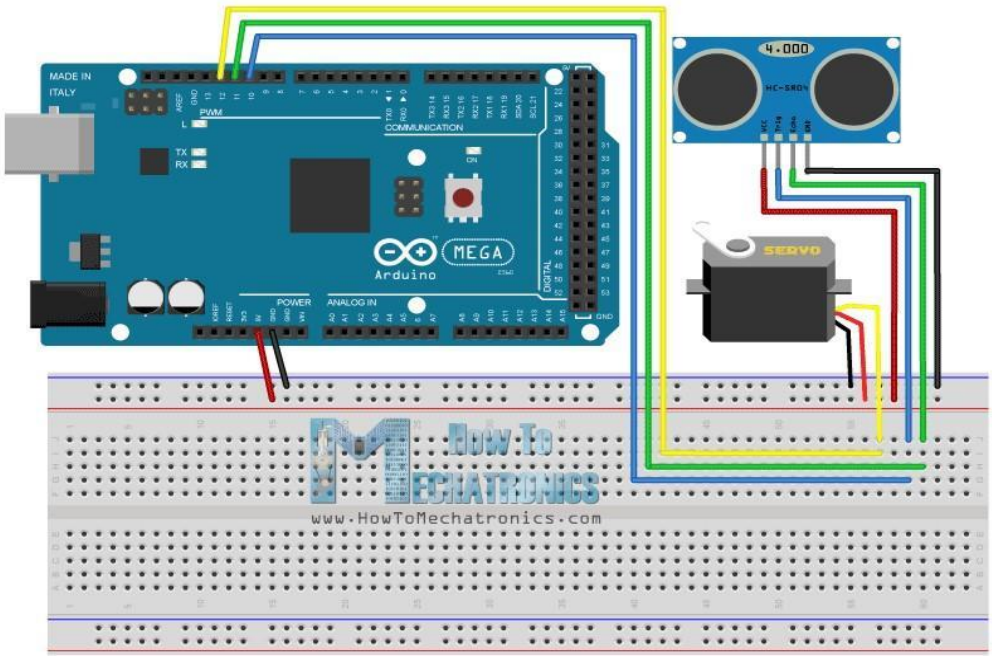
We connected the Ultrasonic Sensor HC-SR04 to the pins number 10 and 11 on the arduino.

TrigPin = 10.

EchoPin = 11.

And the servo motor to the pin number 12 on the Arduino Board. Fig (8) shows the circuit structure of the project(7).

MyServo = 13.



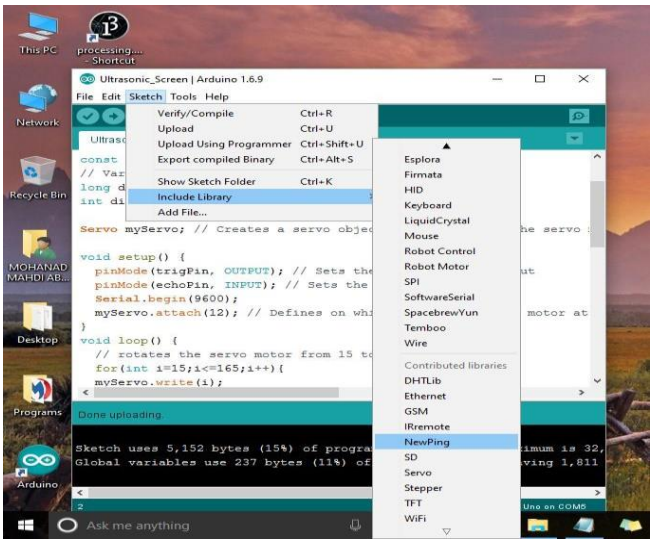
Fig(8). Circuit Diagram

❖ Here's the final appearance of the project:

Fig(9).Final appearance

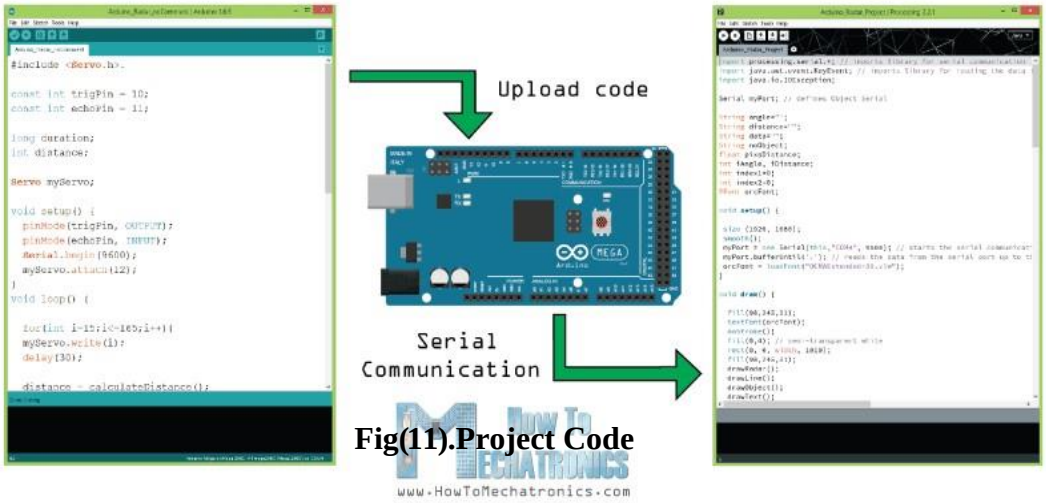
Write and upload sketch to Arduino :

❖ We wrote a sketch in IDE, for this project we need to include some libraries. We use (Serial.h) built-in library for transfer data through the serial port with processing software. Therefore, we add the last library for servo motor (Servo.h) and added NewPing library which includes the last update functions and features for the ultrasonic sensor. Fig (11).



Fig(10).Add library to IDE

❖ Then, we make a code and upload it to the Arduino board to enable the interaction between the Arduino and the Processing IDE Fig(11).



Fig(11).Project Code

Arudino Code :

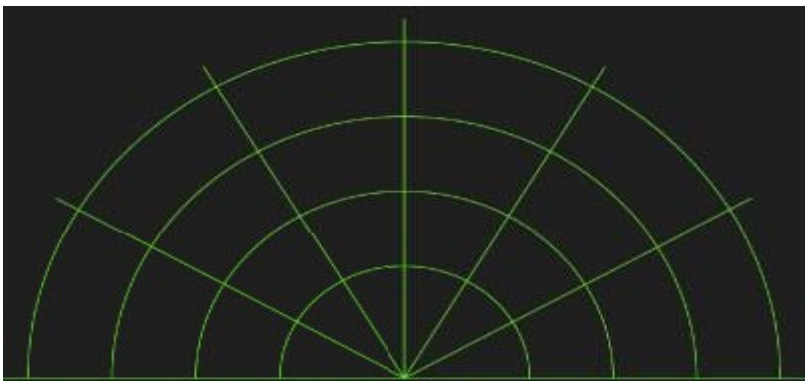
```
#include <Servo.h>.
const int trigPin = 10;
const int echoPin = 11;
long duration;
int distance;
Servo myServo;
void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  Serial.begin(9600);
  myServo.attach(12);
}
void loop() {
  for(int i=15;i<=165;i++){       // rotates the servo motor from 15 to 165 degrees
  myServo.write(i);
  delay(30);
  distance = calculateDistance();
```

```
   Serial.print(i);
  Serial.print(",");
  Serial.print(distance);
  Serial.print(".");
  }
  for(int i=165;i>15;i--){
  myServo.write(i);
  delay(30);
  distance = calculateDistance();
  Serial.print(i);
  Serial.print(",");
  Serial.print(distance);
  Serial.print(".");
  }
}
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;
}
```

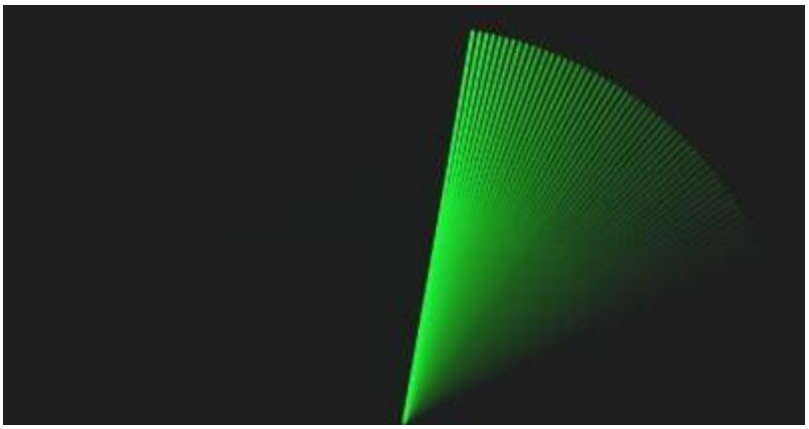
3-3 Write and upload sketch to Processing :

- ❖ Values for the angle and the distance measured by the sensor will be read from the Arduino board by the Processing IDE using the SerialEvent() function which reads the data from the Serial Port. These values will be used for drawing the lines, the detected objects and some texts.

- ❖ For drawing the radar display we make this function drawRadar() which consist of arc() and line() functions Fig(12).

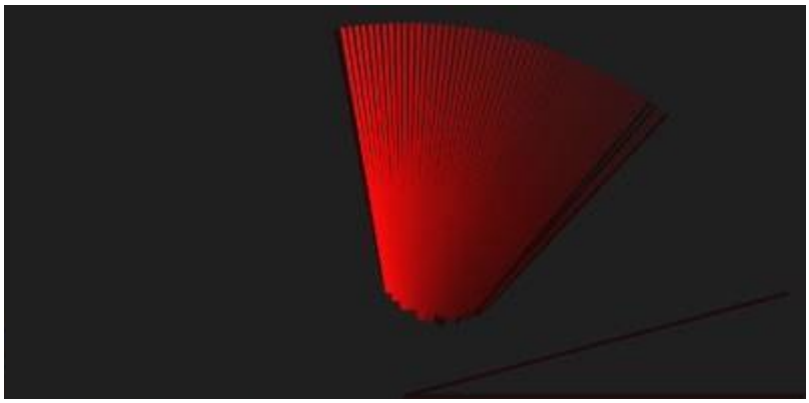


Fig(12). The radar workspace



Fig(13). Radar lines

❖ For drawing the moving lines we make this function *drawLine()*. Its center of rotation is set with the *translate()* function and using the *line()* function in which the iAngle variable is used to redraw the line for each degree. Fig (13).

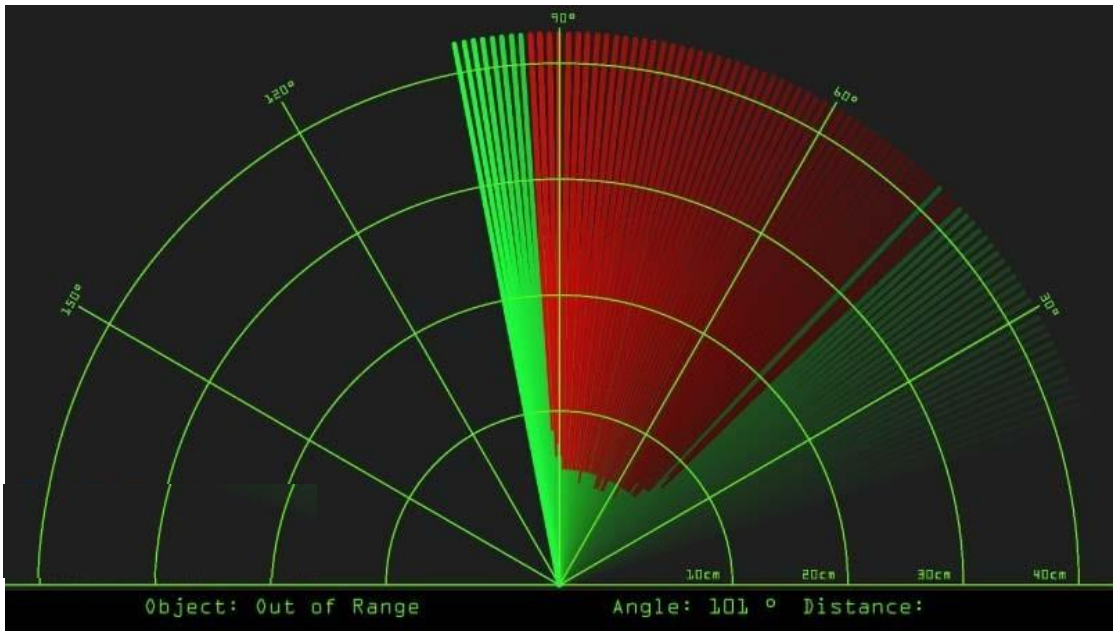


Fig(14). Radar detected lines

- For drawing the detected objects we made the *drawObject()* function. It receives the distance from the ultrasonic sensor, transforms it into pixels. Then, using the angle detected by the sensor it draws the object on the radar screen Fig (14).

- To illustrate the text on the screen, we make the *drawText()* function that draws texts on some particular locations. All of these functions are called in the main *draw()* function which is repeated in each iteration to draw the screen details.

We are using the *fill()* function with 2 parameters for simulating motion blur and slow fade of the moving line. Fig (15) shows the final appearance of the radar screen:



Fig(15). Radar Screen

Processing code:

```
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,"COM3", 9600);
 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("Radar ", 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);
  }
  textSize(25);
  fill(98,245,60);
  translate((width-width*0.4994)+width/2*cos(radians(30)),(height-
height*0.0907)-width/2*sin(radians(30)));
  rotate(-radians(-60));
  text("30°",0,0);
  resetMatrix();
  translate((width-width*0.503)+width/2*cos(radians(60)),(height-
height*0.0888)-width/2*sin(radians(60)));
  rotate(-radians(-30));
  text("60°",0,0);
  resetMatrix();
  translate((width-width*0.507)+width/2*cos(radians(90)),(height-
height*0.0833)-width/2*sin(radians(90)));
  rotate(radians(0));
  text("90°",0,0);
  resetMatrix();
  translate(width-width*0.513+width/2*cos(radians(120)),(height-
height*0.07129)-width/2*sin(radians(120)));
  rotate(radians(-30));
```

```
  text("120°",0,0);
  resetMatrix();
  translate((width-width*0.5104)+width/2*cos(radians(150)),(height-
height*0.0574)-width/2*sin(radians(150)));
  rotate(radians(-60));
  text("150°",0,0);
  popMatrix();
 }
```

4 - Conclusion :

Radar is normally used to determine velocity, range, and position of an object. In this technical project, we read the distance and angles of detected objects in order to convert these data into visual information. The performance of our project is so good. It works smoothly to detect objects within the designed range. The screen shows the information clearly with enough delay for the user to read it. This project could be helpful for object avoidance/ detection applications. This project could easily be extended and could be used in any systems may need it.

5 - Applications :

1. Military applications :

- Detecting and ranging of enemy targets.
- Aiming guns at aircraft and ship.
- Boming ships, aircrafts or cities even at night.
- Directing missiles

2 .Civilian applications :

- Navigational aid on ground and seas.
- Radar altimeter for determing the height of plane above ground.
- Airborne radar for satellite surveillance.
- Police use radar for directing and detecting speed of vehicle.

6 - Limitations:

- Radar cannot resolve in details like human eye, espescially at short distance.
- Radar cannot recognise the colour of the target.
- Radar cannot identify internal aspects of the target.
- Radar can detect an object at minimum range only.
- Deionisation delay

7 - **Futher Enhancement :**

In this project using Arduino board the radar system was implemented. It succeed in helping to be widely used to help detect objects in different environments. In the future it can be updated to used for a larger range and advancements.-

8 - References :

1 http://www.elecfreaks.com/estore/

2 Servo motor datasheet.

3 https://en.wikipedia.org/wiki/Breadboard.

4 http://howtomechatronics.com/projects/arduino-radar-project/