EC5511-ANALOG AND DIGITAL COMMUNICATION LABORATORY

ARDUINO RADAR FOR OBJECT DETECTION

Submitted by

MITHUL AKHASH 2022504549

PERARASAN 2022504551

ABHISHEK 2022504552

YUVARAJ 2022504554

BACHELOR OF ENGINEERING IN ELECTRONICS AND COMMUNICATION ENGINEERING

DEPARTMENT OF ELECTRONICS ENGINEERING

MADRAS INSTITUTE OF TECHNOLOGY

ANNA UNIVERSITY: CHENNAI – 600 004

TABLE OF CONTENTS

ABSTRACT

Radar is an object detection system which uses radio waves to determine the range, altitude, direction, or speed of objects. 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.

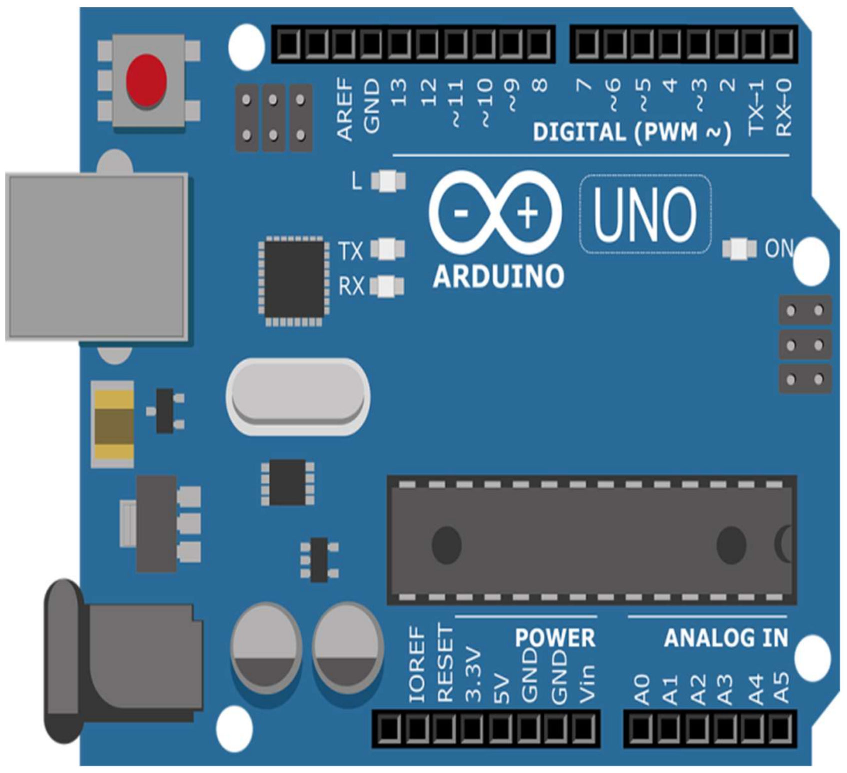
The modern uses of radar are highly diverse, including air traffic control, radar astronomy, air-defense systems, antimissile systems ;marine radar start locate landmarks and other ships; aircraft anti-collision systems; ocean surveillance systems, outer space surveillance and rendezvous systems; meteorological 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 a ultrasonic sensor, the sensor released the waves which we want to measure the distance of a 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.

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.

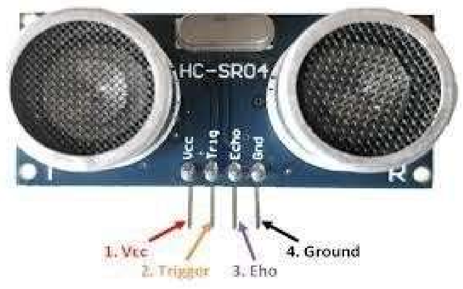
1. Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can 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.
2. 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.
3. 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.



1.1. 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



1.1. 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:

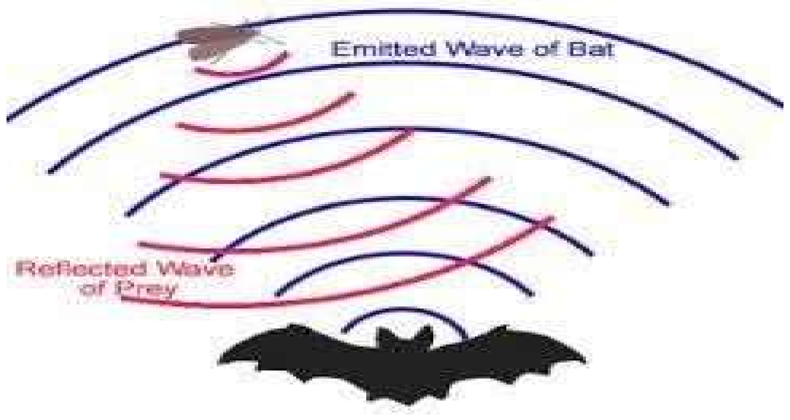
Distance $L = 1/2 \times T \times C$

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.

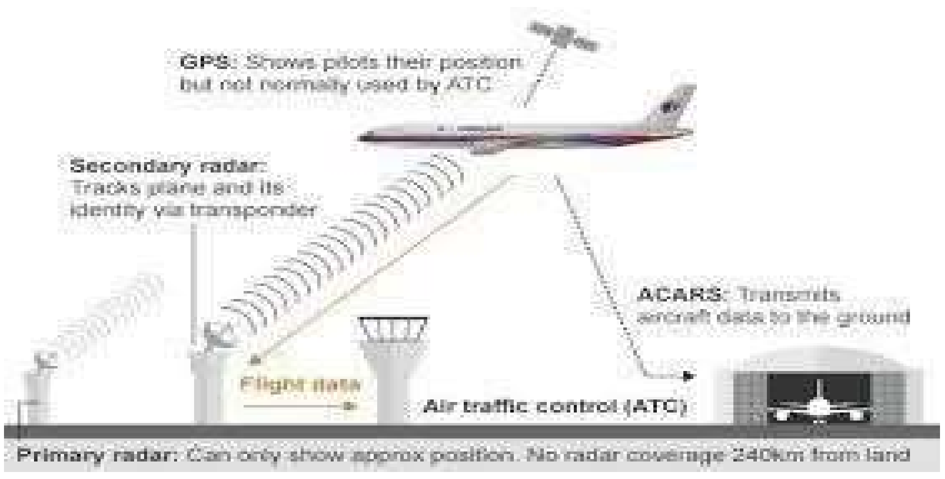
PRINCIPLE OR MEDTHODOLOGY

A radar system has a transmitter that emits radio waves called a radar signals 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

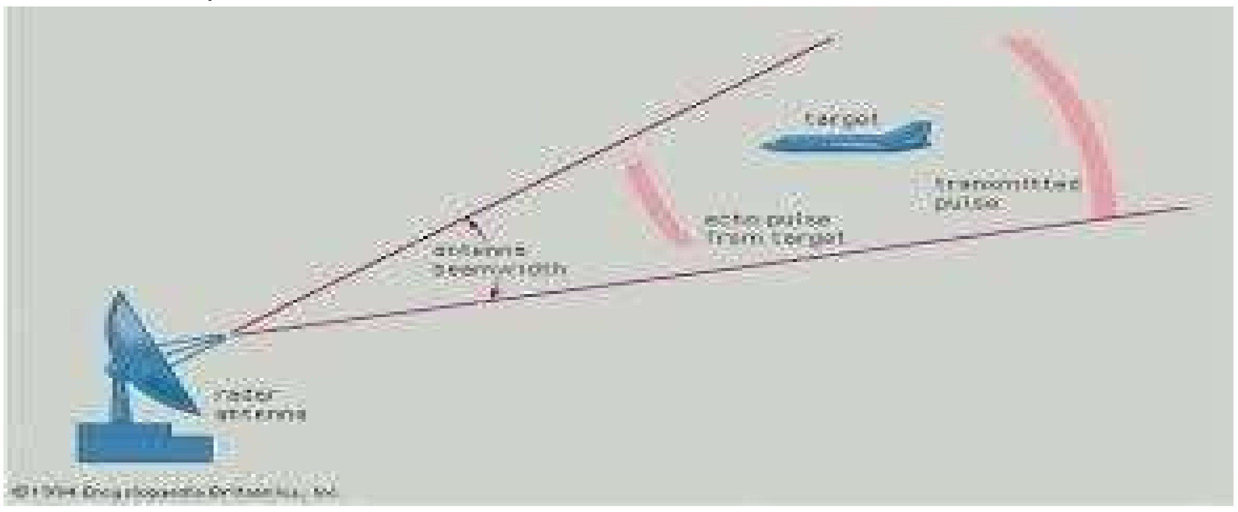


Applications and usages:-

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.


GPS: Shows pilots their position but not normally used by ATC
Secondary radar: Tracks plane and its identity via transponder
ACARS: Transmits aircraft data to the ground
Flight data
Air traffic control (ATC)
Primary radar: Can only show approx position. No radar coverage 240km from land

Here's a summary of how radar works:

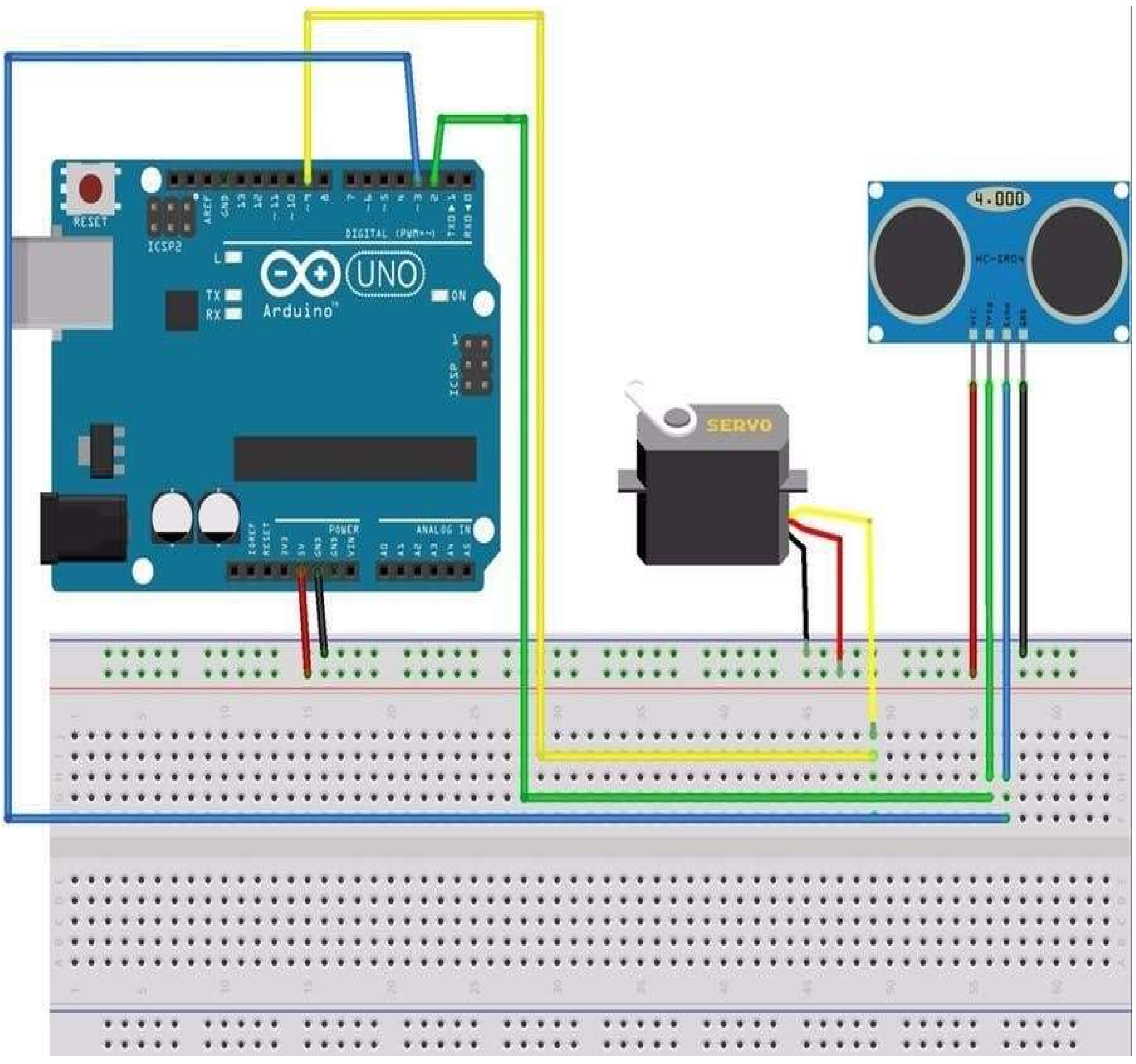


- Magnetron generates high-frequency radio waves.
- Duplexer switches magnetron through to antenna.
- Antenna acts as transmitter, sending narrow beam of radio waves through the air.

Radio waves hit enemy airplane and reflect back.

x

ARCHITECTURE OF PROJECT:-

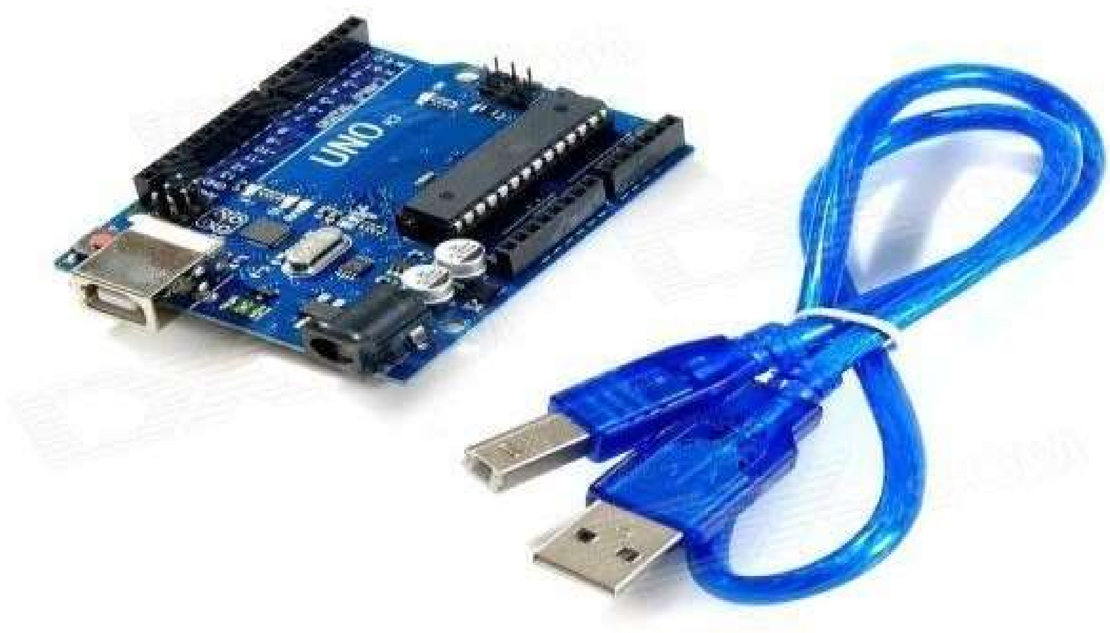


PROCEDURE

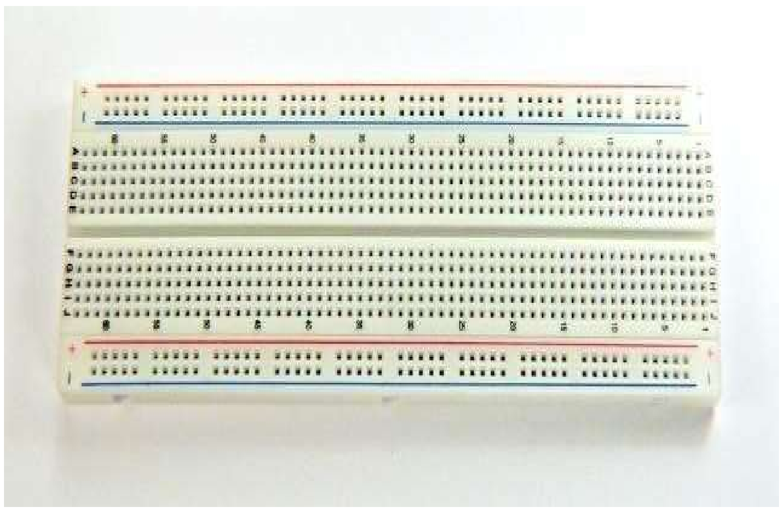
Components Required:

In this project we have used the arduino and ultrasonic sensor 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

 gum gun

 LAPTOP

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 osicillator, 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

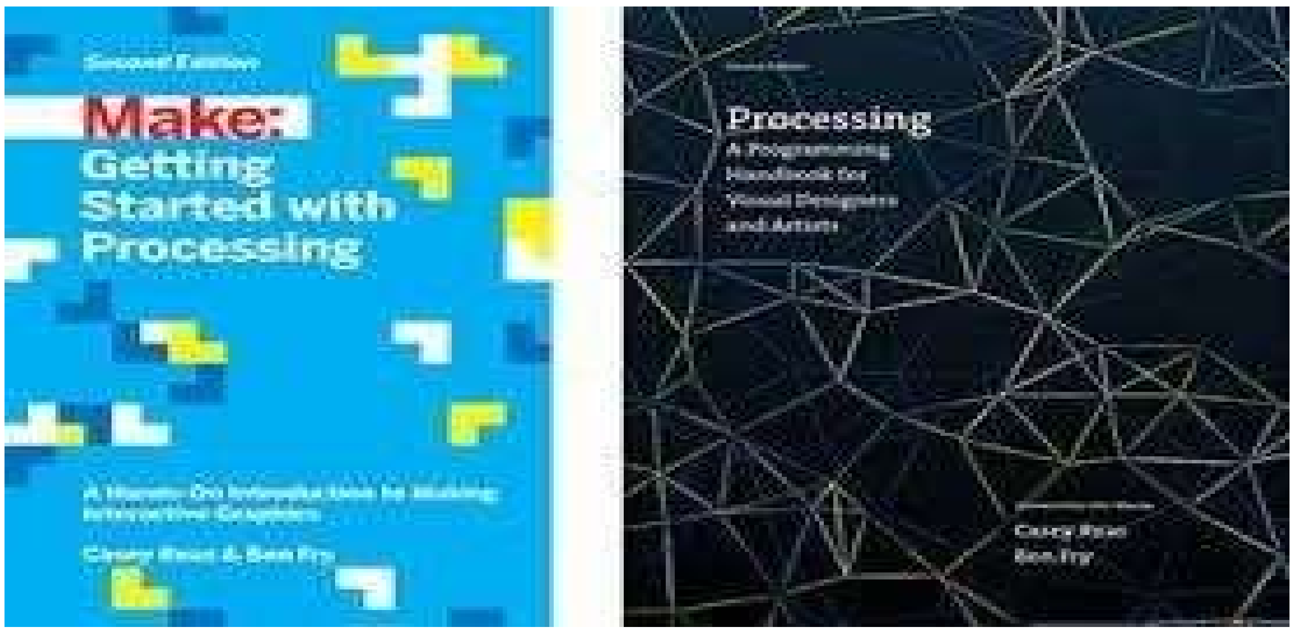
3. PULSE

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.5


Second Edition
Make:
Getting Started with Processing
Casey Reas & Ben Fry
Processing
A Programming Handbook for Visual Designers and Artists
Casey Reas
Ben Fry

VI. USING PROCESSING SOFTWARE

VII.

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.

VIII. 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 from 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 a 8051 microcontroller.

B. Communicating with Arduino through PC

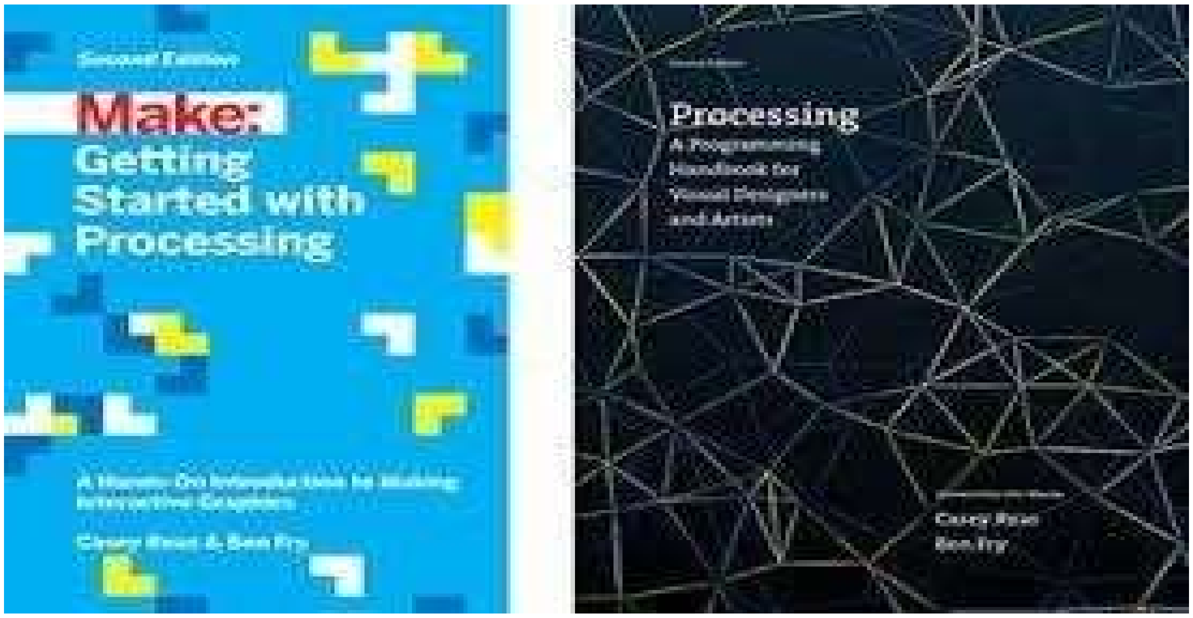
Another major problem related to the Arduino board was the communication with it from PC. Since, there is a requirement of an RS-232 to TTL conversion for the communication, so try some methods:

[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!!!


Second Edition
Make:
Getting Started with Processing
Casey Reas & Ben Fry
Processing
A Programming Handbook for Visual Designers and Artists
Casey Reas
Ben Fry

PROCESSING SOFTWARE:-

ARDUINO SOFTWARE :-

ARDUINO CODE

```
Includes the Servo library #include
<Servo.h>.
// Defines Tirg and Echo pins of the Ultrasonic Sensor
const int trigPin = 10; const int echoPin = 11;
// Variables for the duration and the distance
long duration; int distance;
Servo myServo; // Creates a servo object for controlling the servo motor
void setup() { pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
pinMode(echoPin, INPUT); // Sets the echoPin as an Input
Serial.begin(9600); myServo.attach(12); // Defines on which pin is the
servo motor attached
} void loop()
{
  // rotates the servo motor from 15 to 165 degrees
for(int i=15;i<=165;i++){ myServo.write(i);
delay(30);
  distance = calculateDistance();// Calls a function for calculating the distance measured by the
Ultrasonic sensor for each degree

  Serial.print(i); // Sends the current degree into the Serial Port
  Serial.print(","); // Sends addition character right next to the previous value needed later in
the Processing IDE for indexing
  Serial.print(distance); // Sends the distance value into the Serial Port
  Serial.print("."); // Sends addition character right next to the previous value needed later in
the Processing IDE for indexing
  }
  // Repeats the previous lines from 165 to 15 degrees
for(int i=165;i>15;i--){ myServo.write(i);
delay(30);
```

```
  distance = calculateDistance();
  Serial.print(i);
  Serial.print(",");
  Serial.print(distance);
  Serial.print(".");
  }
}
// Function for calculating the distance measured by the Ultrasonic sensor int
calculateDistance(){

  digitalWrite(trigPin, LOW);
delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
digitalWrite(trigPin,  HIGH);  delayMicroseconds(10);
digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH); // Reads the echoPin, returns the sound wave travel time in
microseconds distance= duration*0.034/2; return distance;
}
```

----

processer code:-

```
        import processing.serial.*; // imports library for serial communication import
java.awt.event.KeyEvent; // imports library for reading the data from the serial port import
java.io.IOException;
Serial myPort; // defines Object Serial
// defubes variables
String    angle="";
String  distance="";
String      data="";
String    noObject;
float  pixsDistance;
int           iAngle,
iDistance;        int
index1=0; int
```

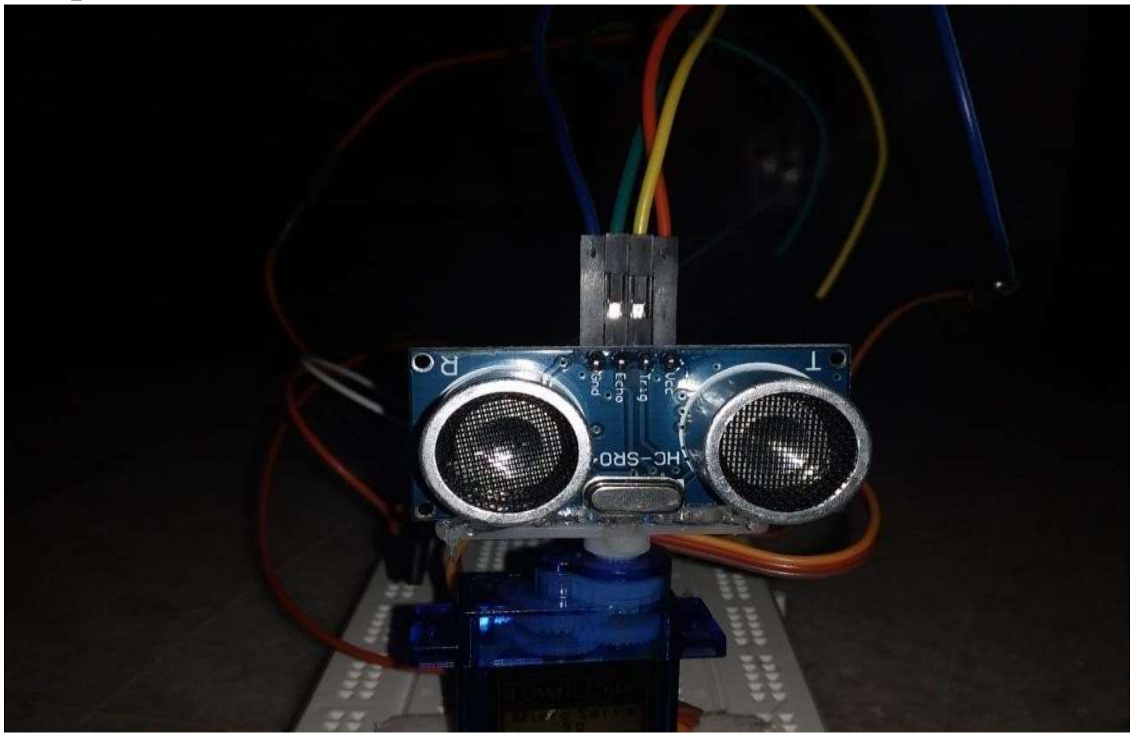
```
index2=0;        PFont
orcFont; void setup()
{

 size (1200, 700); // ***CHANGE THIS TO YOUR SCREEN RESOLUTION***
smooth();
 myPort = new Serial(this,"COM5", 9600); // starts the serial communication
myPort.bufferUntil('.'); // reads the data from the serial port up to the character '.'. So actually it
reads this: angle,distance.
} void draw()
{

  fill(98,245,31);
  // simulating motion blur and slow fade of the moving line
noStroke();  fill(0,4);
  rect(0, 0, width, height-height*0.065);

  fill(98,245,31); // green color
  // calls the functions for drawing the radar
drawRadar();   drawLine();
drawObject();  drawText();
} void serialEvent (Serial myPort) { // starts reading data from the Serial
Port
  // reads the data from the Serial Port up to the character '.' and puts it into the String variable
"data". data = myPort.readStringUntil('.'); data = data.substring(0,data.length()-1);

  index1 = data.indexOf(","); // find the character ',' and puts it into the variable "index1" angle=
data.substring(0, index1); // read the data from position "0" to position of the variable index1
or thats the value of the angle the Arduino Board sent into the Serial Port distance=
data.substring(index1+1, data.length()); // read the data from position "index1" to the end of the
data pr thats the value of the distance
```

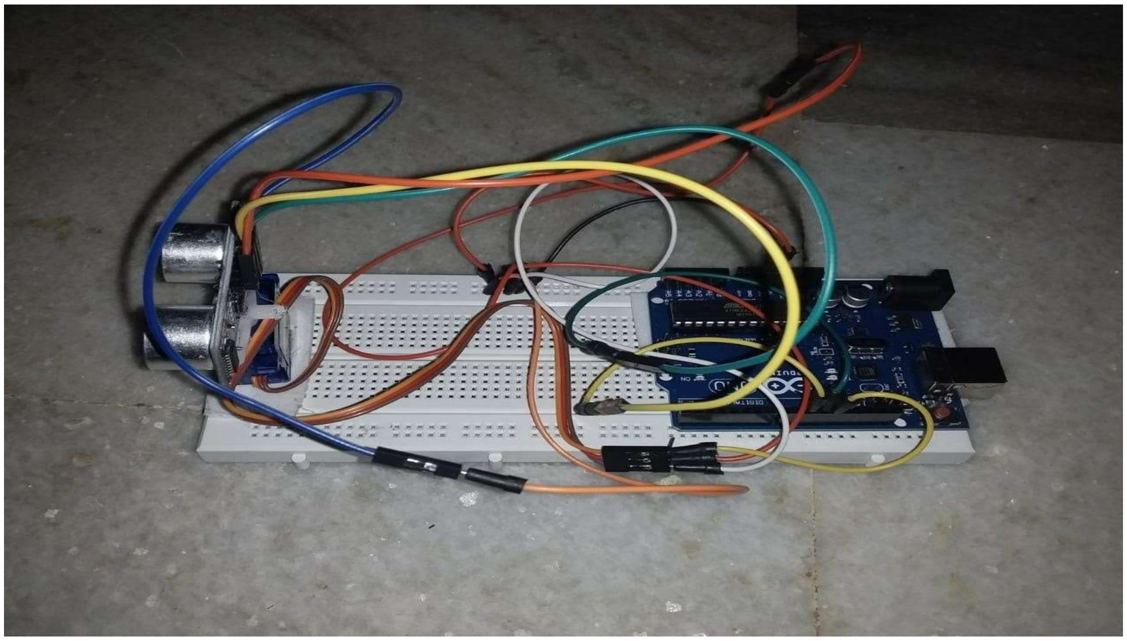
```
 // converts the String variables into Integer
iAngle = int(angle); iDistance =
int(distance);
}
void drawRadar() {  pushMatrix();  translate(width/2,height-height*0.074); // moves
the starting coordinats to new location noFill(); strokeWeight(2);  stroke(98,245,31);
// draws the arc lines
 arc(0,0,(width-width*0.0625),(width-width*0.0625),PI,TWO_PI);  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);

 // draws the angle lines 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); // moves the
starting coordinats to new location strokeWeight(9); stroke(255,10,10); // red color
pixsDistance = iDistance*((height-height*0.1666)*0.025); // covers the distance from the
sensor from cm to pixels // limiting the range to 40 cms  if(iDistance<40){
  // draws the object according to the angle and the distance
line(pixsDistance*cos(radians(iAngle)),-
pixsDistance*sin(radians(iAngle)),(widthwidth*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); // moves the starting coordinats to new location
line(0,0,(height-height*0.12)*cos(radians(iAngle)),-(height-
height*0.12)*sin(radians(iAngle))); // draws the line according to the angle popMatrix();
}
```

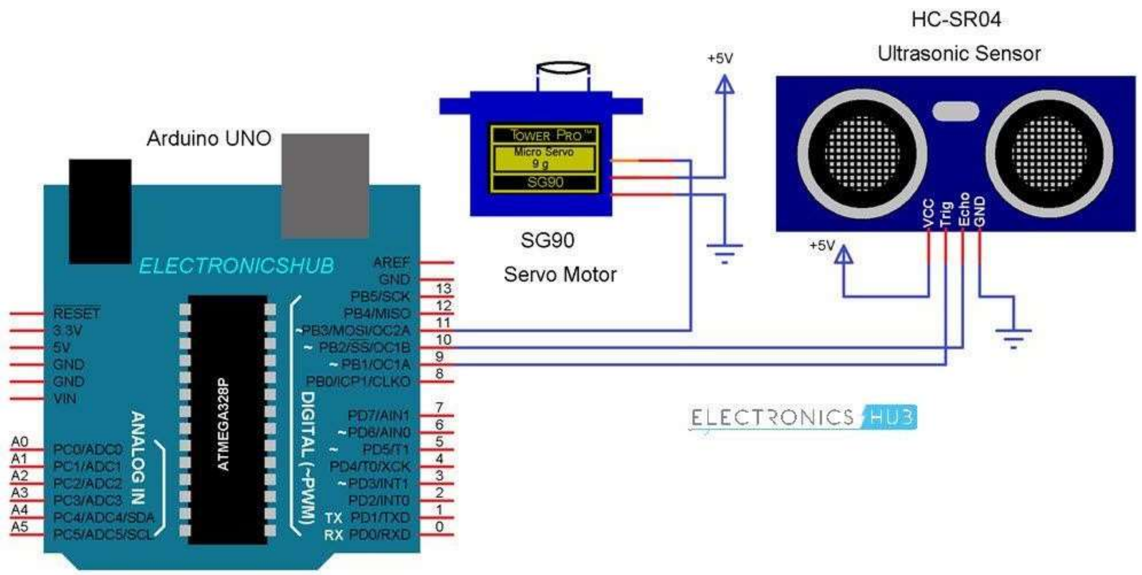
```
void drawText() { // draws the texts on the screen

  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 ("Indian Lifehacker
", 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();
```

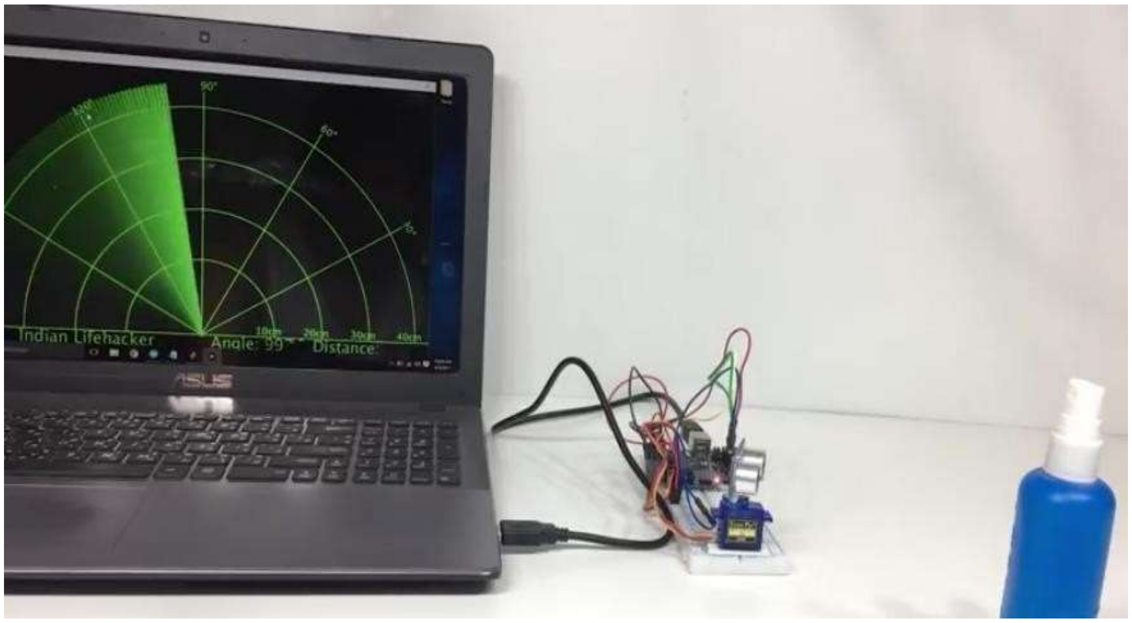
We create a variable analog and assign it to 0.This is because the voltage value we are going to read is connected to the analog pin is A0. This voltage represents the voltage value falls across the resistor value we are measuring. Next we create a variable name raw, which we will use to read in the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

Output: Screenshot






HC-SR04
Ultrasonic Sensor
Arduino UNO
SG90
Servo Motor
ELECTRONICSHUB
+5V
VCC
Trig
Echo
GND
ATMEGA328P
ANALOG IN
DIGITAL (~PWM)
ELECTRONICS HUB


Indian Lifehacker
Angle: 99°
Distance:

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 speedup the testing process. This will also reduce the faults in range selection in manually operated conventional sensor.

CONCLUSIONS:-

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.

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://www.google.co.in/imgres?imgurl=http://www.electrosome.com/wp-content/uploads/2012/06/ServoMotor.gif&imgrefurl=http://www.electrosome.com/tag/se
rvo
motor/&h=405&w=458&sz=67&tbnid=rcdlwDVt_x0DdM:&tbnh=100&tbnw=113&zoo
m=1
&usg=___6J2h0ZocdoSMrS1qgK1I2qpTQSI=&docid=lEfbDrEzDBfzbM&sa=X&ei=a_
OKU vTbD8O5rgeYv4DoDQ&ved=0CDwQ9QE

[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

[11] http://www.radomes.org/museum/photos/equip/ANSPS17.jpg [12]
http://www.wired.com/dangerroom/2011/07/ suicide- bombers-from-100-yards/

[13] http://upload.wikimedia.org/wikipedia/commons/Radaraccumulationseng.png

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

[15] http://arduino.cc/en/Tutorial/LiquidCrystal/

[16] http://fritzing.org