**IOT Based Moving Radar System with Light**

**1Al-Amin, 2Md. Golam Rabby**

**1***UG Student, Dept. Of Computer Science and Engineering, East West University*

***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. When radar can detect object properly then light is working properly and produced the signal. Information received from the sensor will be used by “Processing Development Environment” software* to *illustrate the result on a PC screen.***

**keywords:** **Ultrasonic sensor, Arduino, Servo Motor,light,wire etc.**

1. **Introduction**

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).

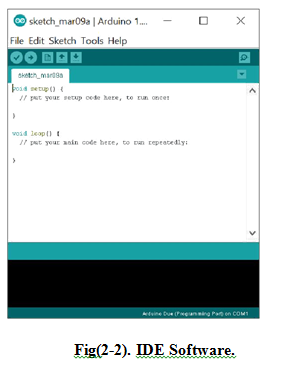
1. **Tools** 
   1. **Arduino Board UNO Model**

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.





1. **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.

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

**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

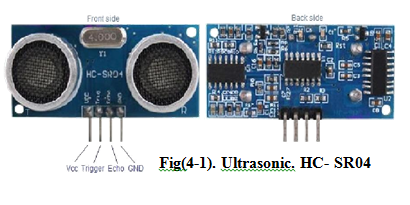
Wire connecting directly as following:

 5V Supply

 Trigger Pulse Input

 Echo Pulse Output

 0V Ground



**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]

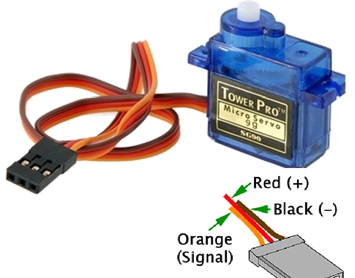
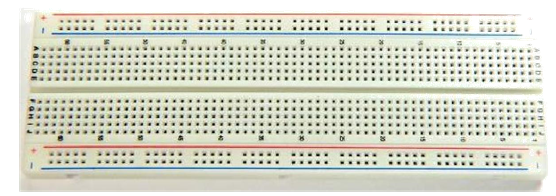


Fig: Servo motor

**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".

1. **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.

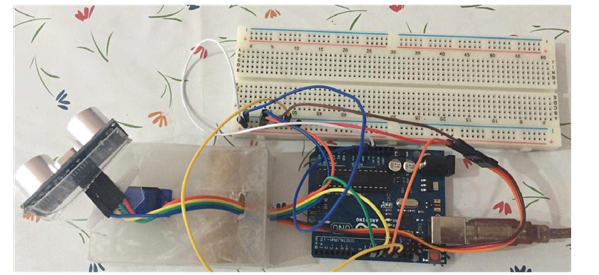


Fig: working procedure

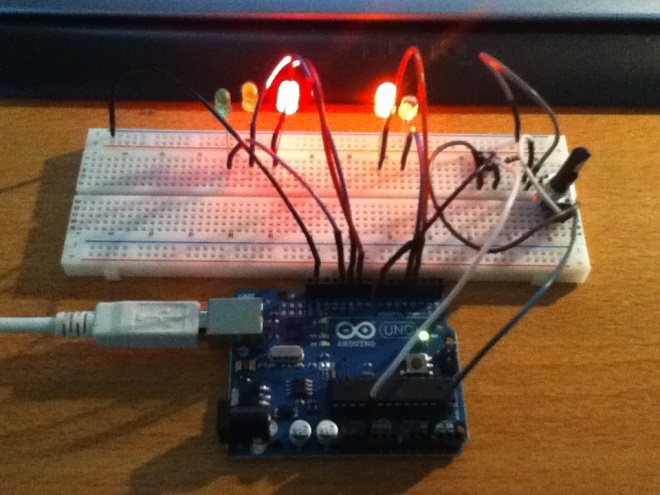


Fig: light is working

1. **Circuit diagram**

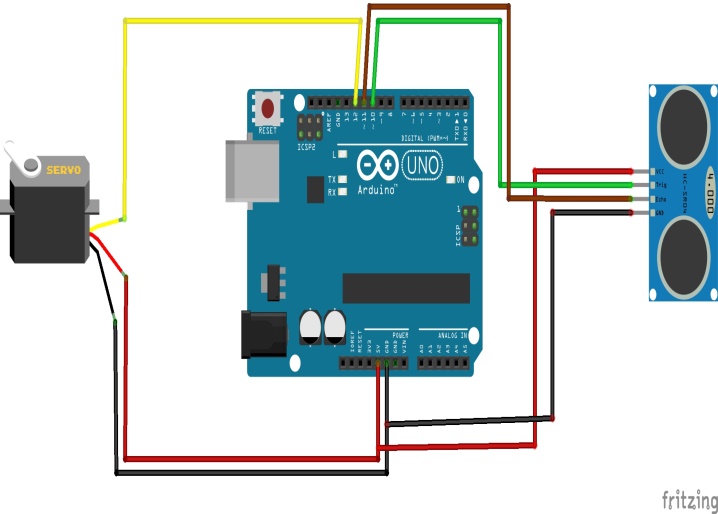
****

Fig: Circuit Diagram Connection

1. **Code**

**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;

}

**//For lighting code:**

#define echoPin 7 // Echo Pin

#define trigPin 8 // Trigger Pin

const int freq = 500;

const int dur = 20;

const int buzzer = 3;

const int red3 = 2;

const int red2 = 4;

const int red1 = 5;

const int yellow3 = 6;

const int yellow2 = 9;

const int yellow1 = 10;

const int green3 = 11;

const int green2 = 12;

const int green1 = 13;

int maximumRange = 200; // Maximum range needed

int minimumRange = 0; // Minimum range needed

long duration, distance; // Duration used to calculate distance

void setup() {

pinMode(buzzer, OUTPUT);

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

pinMode(green1, OUTPUT);

pinMode(green2, OUTPUT);

pinMode(green3, OUTPUT);

pinMode(yellow1, OUTPUT);

pinMode(yellow2, OUTPUT);

pinMode(yellow3, OUTPUT);

pinMode(red1, OUTPUT);

pinMode(red2, OUTPUT);

pinMode(red3, OUTPUT);

digitalWrite(green1, HIGH);

digitalWrite(green2, HIGH);

digitalWrite(green3, HIGH);

digitalWrite(yellow1, HIGH);

digitalWrite(yellow2, HIGH);

digitalWrite(yellow3, HIGH);

digitalWrite(red1, HIGH);

digitalWrite(red2, HIGH);

digitalWrite(red3, HIGH);

delay(300);

digitalWrite(green1, LOW);

digitalWrite(green2, LOW);

digitalWrite(green3, LOW);

digitalWrite(yellow1, LOW);

digitalWrite(yellow2, LOW);

digitalWrite(yellow3, LOW);

digitalWrite(red1, LOW);

digitalWrite(red2, LOW);

digitalWrite(red3, LOW);

tone(buzzer, freq, duration);

}

void loop() {

/\* The following trigPin/echoPin cycle is used to determine the

distance of the nearest object by bouncing soundwaves off of it. \*/

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);

//Calculate the distance (in cm) based on the speed of sound.

distance = duration/58.2;

if (distance < 7){

digitalWrite(green1, HIGH);

digitalWrite(green2, HIGH);

digitalWrite(green3, HIGH);

digitalWrite(yellow1, HIGH);

digitalWrite(yellow2, HIGH);

digitalWrite(yellow3, HIGH);

digitalWrite(red1, HIGH);

digitalWrite(red2, HIGH);

digitalWrite(red3, HIGH);

} else if (distance < 15){

digitalWrite(green1, HIGH);

digitalWrite(green2, HIGH);

digitalWrite(green3, HIGH);

digitalWrite(yellow1, HIGH);

digitalWrite(yellow2, HIGH);

digitalWrite(yellow3, HIGH);

digitalWrite(red1, HIGH);

digitalWrite(red2, HIGH);

digitalWrite(red3, LOW);

} else if (distance < 20){

digitalWrite(green1, HIGH);

digitalWrite(green2, HIGH);

digitalWrite(green3, HIGH);

digitalWrite(yellow1, HIGH);

digitalWrite(yellow2, HIGH);

digitalWrite(yellow3, HIGH);

digitalWrite(red1, HIGH);

digitalWrite(red2, LOW);

digitalWrite(red3, LOW);

} else if (distance < 25){

digitalWrite(green1, HIGH);

digitalWrite(green2, HIGH);

digitalWrite(green3, HIGH);

digitalWrite(yellow1, HIGH);

digitalWrite(yellow2, HIGH);

digitalWrite(yellow3, HIGH);

digitalWrite(red1, LOW);

digitalWrite(red2, LOW);

digitalWrite(red3, LOW);

} else if (distance < 30){

digitalWrite(green1, HIGH);

digitalWrite(green2, HIGH);

digitalWrite(green3, HIGH);

digitalWrite(yellow1, HIGH);

digitalWrite(yellow2, HIGH);

digitalWrite(yellow3, LOW);

digitalWrite(red1, LOW);

digitalWrite(red2, LOW);

digitalWrite(red3, LOW);

} else if (distance < 35){

digitalWrite(green1, HIGH);

digitalWrite(green2, HIGH);

digitalWrite(green3, HIGH);

digitalWrite(yellow1, HIGH);

digitalWrite(yellow2, LOW);

digitalWrite(yellow3, LOW);

digitalWrite(red1, LOW);

digitalWrite(red2, LOW);

digitalWrite(red3, LOW);

} else if (distance < 40){

digitalWrite(green1, HIGH);

digitalWrite(green2, HIGH);

digitalWrite(green3, HIGH);

digitalWrite(yellow1, LOW);

digitalWrite(yellow2, LOW);

digitalWrite(yellow3, LOW);

digitalWrite(red1, LOW);

digitalWrite(red2, LOW);

digitalWrite(red3, LOW);

} else if (distance < 45){

digitalWrite(green1, HIGH);

digitalWrite(green2, HIGH);

digitalWrite(green3, LOW);

digitalWrite(yellow1, LOW);

digitalWrite(yellow2, LOW);

digitalWrite(yellow3, LOW);

digitalWrite(red1, LOW);

digitalWrite(red2, LOW);

digitalWrite(red3, LOW);

} else if (distance < 50){

digitalWrite(green1, HIGH);

digitalWrite(green2, LOW);

digitalWrite(green3, LOW);

digitalWrite(yellow1, LOW);

digitalWrite(yellow2, LOW);

digitalWrite(yellow3, LOW);

digitalWrite(red1, LOW);

digitalWrite(red2, LOW);

digitalWrite(red3, LOW);

} else if (distance < 55){

digitalWrite(green1, LOW);

digitalWrite(green2, LOW);

digitalWrite(green3, LOW);

digitalWrite(yellow1, LOW);

digitalWrite(yellow2, LOW);

digitalWrite(yellow3, LOW);

digitalWrite(red1, LOW);

digitalWrite(red2, LOW);

digitalWrite(red3, LOW);

} else if (distance > 55){

digitalWrite(green1, LOW);

digitalWrite(green2, LOW);

digitalWrite(green3, LOW);

digitalWrite(yellow1, LOW);

digitalWrite(yellow2, LOW);

digitalWrite(yellow3, LOW);

digitalWrite(red1, LOW);

digitalWrite(red2, LOW);

digitalWrite(red3, LOW);

}

if(distance < 7){

tone(buzzer, freq, dur);

delay(5);

} else if (distance < 15){

tone(buzzer, freq, dur);

delay(10);

} else if (distance < 20){

tone(buzzer, freq, dur);

delay(15);

} else if (distance < 25){

tone(buzzer, freq, dur);

delay(20);

} else if (distance < 30){

tone(buzzer, freq, dur);

delay(25);

} else if (distance < 35){

tone(buzzer, freq, dur);

delay(30);

} else if (distance < 40){

tone(buzzer, freq, dur);

delay(35);

} else if (distance < 45){

tone(buzzer, freq, dur);

delay(40);

} else if (distance < 50){

tone(buzzer, freq, dur);

delay(45);

} else if (distance < 55){

tone(buzzer, freq, dur);

delay(50);

}

//Delay 50ms before next reading.

delay(50);

}

**//processing 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,"COM3", 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(220,10,10); // 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)),(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); // 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("EWUIAN ", 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)));

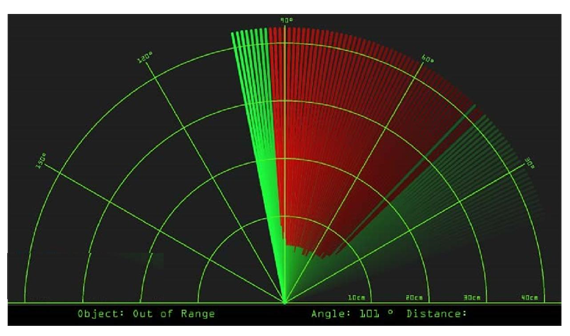
rotate(radians(-60));

text("150°",0,0);

popMatrix();

}

1. **Write and upload sketch to Processing**



The 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 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.

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.

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.

1. **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.

**Reference**

[1] Ultrasonic Radar and Its Applications ‘MANSOOR-UL-HASSAN SIDDIQUE’ Chief Executive Officer Digitronics Communication Pvt. Ltd DCPL Opp. Aladdin Amusement Park Pakistan

[2] Ultrasonic RADAR/Electronic Design Lab (EE-318) Submitted by:Praveen Tamhankar (06d07007),Piyush Mittal (06d07035), Ashutosh Singh (06d07034) Guide: Prof. Jayanta Mukhergee.

[3] https://en.wikipedia.org/wiki/Arduino

[4] http://www.elecfreaks.com/estore/

[5] Servo motor datasheet.

[6]https://en.wikipedia.org/wiki/Breadboard.

[7]<http://howtomechatronics.com/projects/arduino-radar-project/>