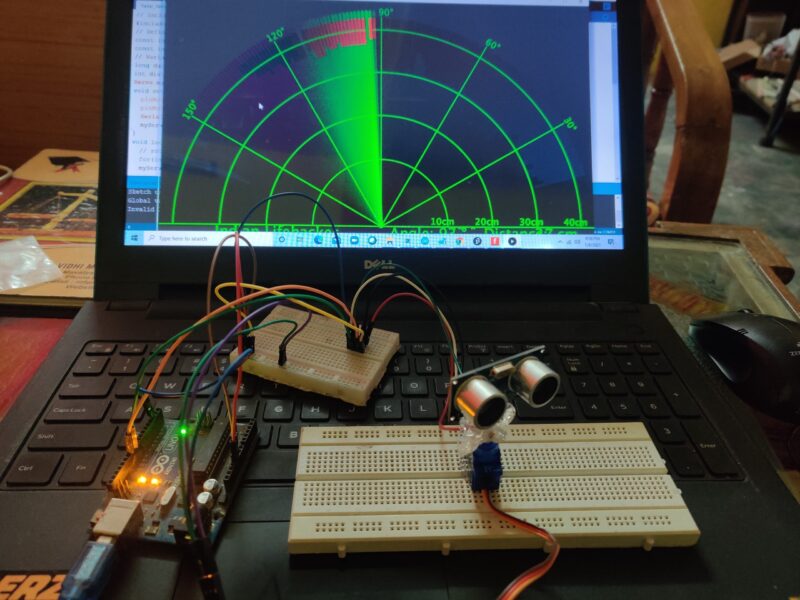
HOME SECURITY SYSTEM BY USING RADAR



Elsaphanne project

**Contents**

[1 Introduction](https://www.circuits-diy.com/ultrasonic-sensor-with-servo-motor-arduino-tutorial/#Introduction)

[2 Hardware Components](https://www.circuits-diy.com/ultrasonic-sensor-with-servo-motor-arduino-tutorial/#Hardware_Components)

[3 Ultrasonic Sensor Servo Motor with Arduino](https://www.circuits-diy.com/ultrasonic-sensor-with-servo-motor-arduino-tutorial/#Ultrasonic_Sensor_Servo_Motor_with_Arduino)

[3.1 Schematic](https://www.circuits-diy.com/ultrasonic-sensor-with-servo-motor-arduino-tutorial/#Schematic)

[3.1.1 Wiring / Connections](https://www.circuits-diy.com/ultrasonic-sensor-with-servo-motor-arduino-tutorial/#Wiring_Connections)

[3.2 Installing Arduino IDE](https://www.circuits-diy.com/ultrasonic-sensor-with-servo-motor-arduino-tutorial/#Installing_Arduino_IDE)

[3.3 Installing Libraries](https://www.circuits-diy.com/ultrasonic-sensor-with-servo-motor-arduino-tutorial/#Installing_Libraries)

[3.4 Code](https://www.circuits-diy.com/ultrasonic-sensor-with-servo-motor-arduino-tutorial/#Code)

[3.5 Working Explanation](https://www.circuits-diy.com/ultrasonic-sensor-with-servo-motor-arduino-tutorial/#Working_Explanation)

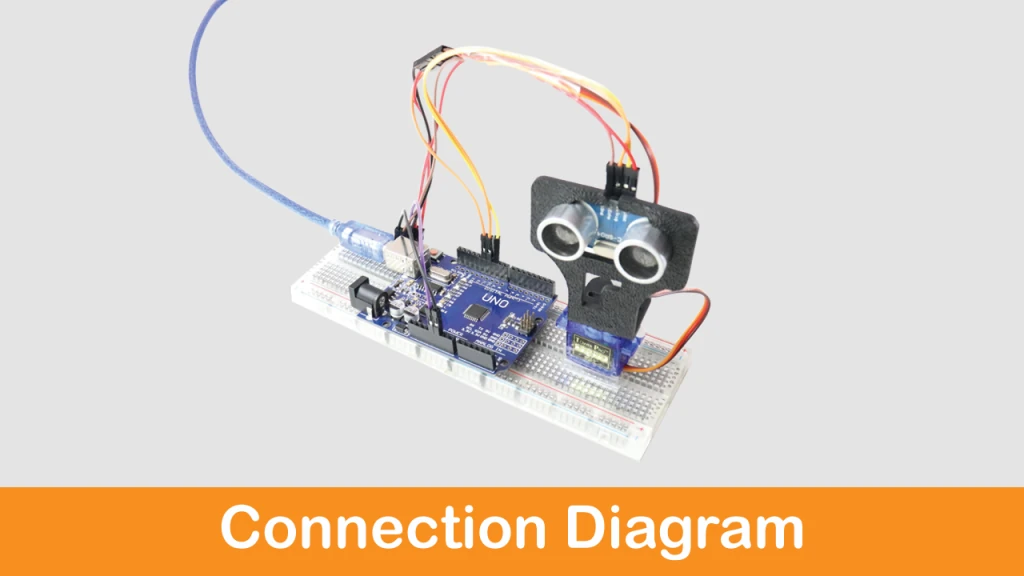
[4 Applications](https://www.circuits-diy.com/ultrasonic-sensor-with-servo-motor-arduino-tutorial/#Applications)

[5 Conclusion.](https://www.circuits-diy.com/ultrasonic-sensor-with-servo-motor-arduino-tutorial/#Conclusion)

**Introduction**

# Radar using Arduino and Ultrasonic sensor

Radar using Arduino and ultrasonic sensors is a very good project in the electronic branch. It can detect any object in the path with the help of an ultrasonic sensor. Ultrasonic sensors rotate over the servo motor and almost cover all directions in rotatory motion. There is processing software that makes it possible. The ultrasonic sensor is very useful in many electronics projects. it uses in distance measuring and object detection. The radar system uses the ultrasonic sensor to detect the object in front of the sensor. And servo motor help to cover the distance by rotating itself slowly.



Hey, welcome back to ElisaphaTronic. I’m Elsaphanne. Make sure to share all the knowledge on a project I have. My main motive is to teach you better. I hope you will learn everything about the project. And if you have any query you can ask in the comment section. In this **Arduino Radar** project, I will share all the instructions and code, a circuit by which you can make the project by itself. Arduino radar is a very attractive and famous project in electronics and can be made in major projects. Although it is only for the learning purpose we can’t use this project in the product as well.

## What is Arduino Radar it?

As I mentioned in the above paragraph the **Arduino Radar detector** project is popular nowadays and now we will understand what actually it is. It is just like mini radar it recognizes the object in the path and makes a red affected area near the object. So, this is called radar and it works on the Ultrasonic frequency. it consists ultrasonic sensor mounted over the servo motor and connected to software that shows the output on the computer screen. The interface of this software is pretty the same as the Radar interface.

## How Does Radar using Arduino Works?

The ultrasonic sensor rotates with the servo motor and transmitted the ultrasonic waves during this time. And the whole time a graph interface make in the simulation software. And if any object comes under the range of the ultrasonic sensor it starts to detect the object. At that time the graph interface inside the software becomes red in the object area. Ultrasonic sensor work as an object detector in this project. **Radar using ultrasonic sensor** works in software makes the reaction according to the waves received.

Ultrasonic sensors have two terminals one is a transmitter and another is the receiver. The transmitter terminal is known as the Trigger and the receiver terminal is known as the echo. Arduino continuously gives a command to the Servo motor to rotate. And the transmitter transmits the signal parallel likewise the software also makes the graph. The ultrasonic sensor gave a different signal to the Arduino if anything comes in the path. Then Arduino notifies the software for the affected region. The project depends on the [ultrasonic sensor working](https://techatronic.com/obstacle-avoiding-robot-using-arduino/). **Radar using Arduino**, ultrasonic sensor, and servo motor contents no other major components. I will also give you the processing software link below.

An HC-SR04 analog thresholds controlled SG51R servo motor using an Arduino UNO is an electronic system that uses a combination of an HC-SR04 ultrasonic distance sensor, an SG51R servo motor, and an Arduino UNO microcontroller to control the position of the servo motor based on the distance of an object in front of the sensor.

The HC-SR04 sensor uses ultrasonic waves to measure the distance of an object in front of it and returns the distance in centimeters. The Arduino UNO microcontroller receives this distance data and uses it to control the position of the SG51R servo motor.

***Hardware Components required:***

You will require the following hardware for Interfacing Ultrasonic Sensor Servo Motor with Arduino.

* Arduino Uno
* Servo motor
* Processing Software
* Breadboard
* Jumper wire
* Arduino cable
* computer

**DESCRIPTION OF COMPONENTS**

## *Arduino uno\_R3*

## C:\Users\KIIZA\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\45235128.tmp

## The Arduino UNO is the best board to get started with electronics and coding. If this is your first experience tinkering with the platform, the UNO is the most robust board you can start playing with. The UNO is the most used and documented board of the whole Arduino family.

## Arduino UNO is a microcontroller board based on the ****ATmega328P****. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. You can tinker with your UNO without worrying too much about doing something wrong, worst case scenario you can replace the chip for a few dollars and start over again.

## *Parts of Arduino uno R3*

## C:\Users\KIIZA\AppData\Local\Microsoft\Windows\INetCache\Content.MSO\AE545976.tmp

## 

## *Ultrasonic Sensor HC-SR04*

HC-SR04 is an ultrasonic distance sensor. This economical sensor provides a sensing range of 2 to 400cm of non-contact measurement functionality with an accuracy that can reach up to 3mm. Each HC-SR04 module includes an ultrasonic transmitter, a receiver, and a control circuit.

There are only 4 pins that you need to know about the HC-SR04 Sensor: VCC (Power), GND (Ground), Trig (Trigger), and Echo (Receive), and. You will find this sensor very simple and easy to set up and use for your next DIY projects & tutorials.



HC-SR04 Ultrasonic Features

* Operating voltage: +5V
* Theoretical  Measuring Distance: 2cm to 450cm
* Practical Measuring Distance: 2cm to 80cm
* Accuracy: 3mm
* Measuring angle covered: <15°
* Operating Current: <15mA
* Operating Frequency: 40Hz

**HC-SR04 Ultrasonic Pinout**

|  |  |  |
| --- | --- | --- |
| **Pin Number** | **Pin Name** | **Description** |
| 1 | Vcc | The Vcc pin powers the sensor, typically with +5V |
| 2 | Trigger | The trigger pin is an Input pin. This pin has to be kept high for 10us to initialize measurement by sending a US wave. |
| 3 | Echo | The echo pin is an Output pin. This pin goes high for a period of time which will be equal to the time taken for the US wave to return back to the sensor. |
| 4 | Ground | This pin is connected to the Ground of the system. |

**SERVO MOTOR**



# What Is A Servo Motor?

As our technology advances, the use of robots and other autonomous applications in our daily lives increases as well. While cheaper robots use stepper or brushed DC motors, more advanced robotics require the use of servo motors. But what is a servo motor and why are they used in most industrial applications?

A servo motor is a self-contained electrical device that moves parts of a machine with high efficiency and great precision. In simpler terms, a servo motor is a BLDC motor with a sensor for positional feedback. This allows the output shaft to be moved to a particular angle, position, and velocity that a regular motor cannot do. However, a servo motor is only one part of a closed-loop motion control system. A complete motion system includes an amplifier, control circuit, drive gears, potentiometer, shaft, and either an encoder or resolver as well as the servo motor.

# *Servo or Stepper?*

What makes a servo motor different from a stepper motor? While both kinds of motors can control speed and position, they are both designed for very different applications. Stepper motors have built-in steps allowing the controller to signal how many steps to make, however, this only works if the controller knows the position of the output shaft. Because of this, when a stepper motor is powered up the controller moves the output shaft to a known position or until it activates an end limit switch. A servo motor uses a sensor to know the position of its output shaft so that when it is powered on it can immediately go to the desired position.

The design of a stepper motor limits its performance due to the limited feedback sensors. If the system is overloaded a stepper motor could skip steps causing positional errors and having to recalibrate the motion system. With a servo motor, the encoder and controller track the position of the output shaft so any errors can be corrected.

Another differentiation between a servo and a stepper motor is their top speed and torque at speeds. A stepper motor provides the most torque when it is not moving, referred to as holding torque. As the speed of the output shaft increases, the torque is drastically reduced. A servo motor has a much higher top speed than a stepper motor and will be able to provide more torque at speed.

# *Advantages of Servo Motors*

Servo motors offer several advantages over other types of motors, including:

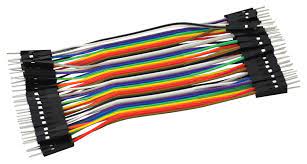
* **Precision Control**: Servo motors provide precise control of position, speed, and torque, making them ideal for applications where accuracy and repeatability are critical.
* **High Torque**: Servo motors are designed to provide high torque at all speeds, which makes them well-suited for applications that require high starting torque and move loads at high speeds.
* **Fast Response Time**: Servo motors have a very fast response time, which makes them ideal for applications that require rapid acceleration.
* **Wide Speed Range**: Servo motors are capable of operating at a wide range of speeds, from very slow to very fast, without losing accuracy or precision.

# *Servo Motor Applications*

Servo motors are used in a wide range of industrial and commercial applications that require precise control of motion, including:

* **Robotics**: Servo motors are commonly used in robotics for precise control of joint movements, as well as for grippers and end effectors.
* **CNC Machines**: Servo motors are widely used in computer numerical control (CNC) machines for precision positioning and motion control of cutting tools.
* **Packaging Machinery**: Servo motors are used in packaging machinery to control the movement of conveyor belts, as well as for the precise positioning and movement of packaging materials.
* **Aerospace**: Servo motors are used in aerospace applications for precise control of aircraft components, such as flaps and landing gear.
* **Autonomous Guided Vehicles**: Servo motors are used to power and steer the wheels in autonomous guided vehicles.
* **Medical Equipment**: Servo motors are used in medical equipment for precise control of medical devices, such as surgical robots, prosthetics, and other medical devices.
* **Printing and Paper Processing**: Servo motors are used in printing and paper processing equipment for precise control of paper feeding, cutting, and folding.
* **Industrial Automation**: Servo motors are used in a wide range of industrial automation applications, including conveyor systems, material handling, and assembly lines.

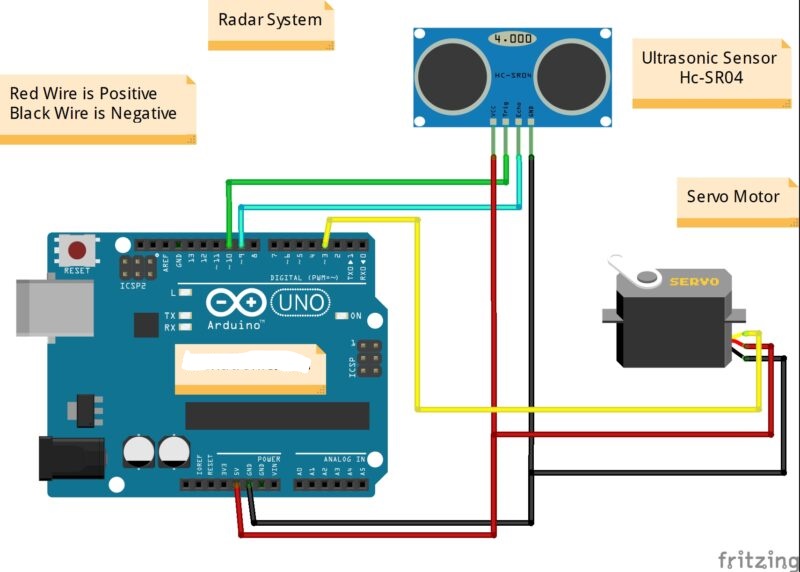
***JUMPER WIRES***



Jumper wires are simply wires that have connector pins at each end, allowing them to be used to connect two points to each other without soldering. Jumper wires are typically used with [breadboards](https://blog.sparkfuneducation.com/what-is-a-breadboard) and other prototyping tools in order to make it easy to change a circuit as needed. Fairly simple. In fact, it doesn’t get much more basic than jumper wires.

***Schematic Diagram/ Fritzing diagram***

Make connections according to the circuit diagram given below.



## 

|  |  |
| --- | --- |
| **Arduino UNO** | **Ultrasonic Sensor** |
| ( +5V ) VCC | VCC ( Positive + ) |
| GND ( Ground ) | GND ( Ground – ) |
| D10 Pin | Trig Pin |
| D9 Pin | Echo Pin |
| **Arduino UNO** | **Servo Motor** |
| D3 Pin | OUT Pin ( Orange Colour ) |
| ( +5V ) VCC | VCC        ( Red Colour ) |
| GND ( Ground ) | GND       ( Black Colour ) |

## Software Installation and Code

We need two software to complete this Arduino radar project. One is Arduino IDE, and the other is Processing IDE. Download both Software from the below links,

* [**Arduino IDE 1.8.13**](https://www.arduino.cc/en/software)
* [**Processing IDE**](https://processing.org/download/)

Processing application is visual arts-based software for learning to code. After downloading, extract the Zip file, and you will get the processing application (.exe file).

In this project, we are using two codes: Arduino UNO and the other for Processing.

* **Here is the Arduino source code**:

// Includes the Servo library

#include <Servo.h>.

// Defines Tirg and Echo pins of the Ultrasonic Sensor

const int trigPin = 10;

const int echoPin = 9;

// 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;

}

* **Here is the processing code for the Arduino radar project:**

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,"COM11", 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)),(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("Elsaphanne ", 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();

}

***Working Explanation***

The code starts by including the necessary libraries such as the Servo library and the NewPing library, which provide the necessary functions for controlling the servo motor and measuring the distance of an object using the HC-SR04 sensor. Next, the code declares the pin numbers for the HC-SR04 sensor, SG51R servo motor, and the Arduino UNO. The trig pin of the HC-SR04 is connected to pin 8, the echo pin to pin 9, and the control pin of the SG51R servo motor to pin 10.

In the setup() function, the code initializes the serial communication, attaches the servo motor to the control pin, creates a new instance of the NewPing library, and set the maximum distance threshold. In the loop() function, the code uses the ping() function of the NewPing library to measure the distance of an object in front of the HC-SR04 sensor in centimeters. The code then creates an if-else statement to check if the distance of the object is greater or less than a certain threshold that is set by the user. If the distance is less than the threshold, the code rotates the servo motor to a certain angle as an indication that the object is too close.

***Applications***

* Security systems and surveillance cameras
* Industrial automation for controlling the position of machine parts
* Automated testing equipment
* Medical equipment for precise positioning and movement
* Automatic vending machines
* Distance-based control in automated stages for theater and events
* Smart home appliances

***ADVANTAGES***

1. Radar procurable value is very low

2. Working and maintenance value is low.

3. Distance active resolution is high

4. Radar’s jam is troublesome

5. It can work in any place

6. NASA uses radio detection and ranging to map the world and alternative plants 7. Activity gets updated in conclusion

***ACKNOWLEDGEMENT***

I would like to thank my Lecture ISHIMWE Viviane for his expert advice and encouragement through this difficult project. I would also like

To thank my friends who always encouraged me throughout this period.

***Conclusion.***

In this paper a system radar system was designed with the help of Arduino, servomotor and ultrasonic sensor which can detect the position,

Distance of obstacle which comes in its way and converts it into visually representable form.

This system can be used in robotics for object detection and avoidance system or can also be used for intrusion detection for location sizes.

Range of the system depends upon type of ultra-sonic sensor used. We used HC-SR04 sensor which range from 2 to 40 cm.

We hope you have found this Arduino – Ultrasonic Sensor – Servo Motor Circuit very useful. If you feel any difficulty in making it feel free to ask anything in the comment section.

THANK YOU !!!