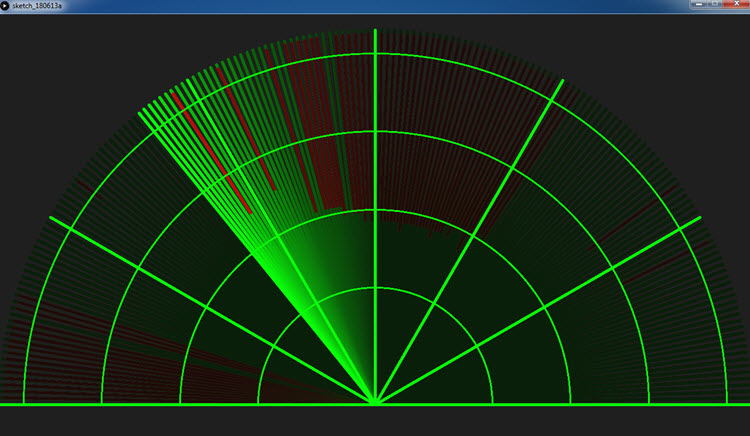
**R&D Project: Radar Based Security System**

Abstract:

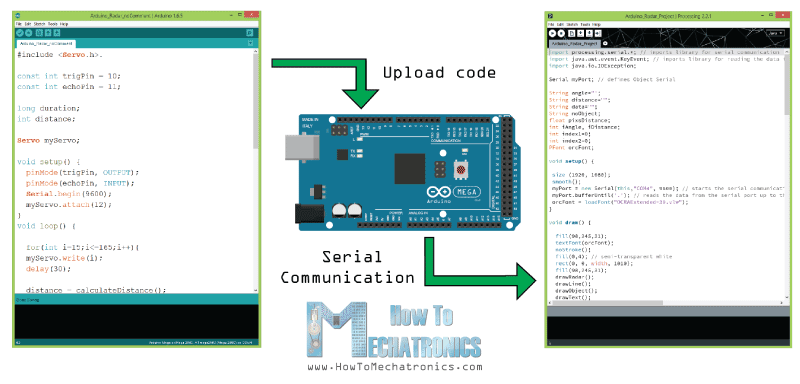
**Objective:**

Radar (Radio Detection and Ranging) is a detection system that uses [radio waves](https://en.wikipedia.org/wiki/Radio_wave) to determine the distance (range), angle, or velocity of objects. It can be used to detect [aircraft](https://en.wikipedia.org/wiki/Aircraft), [ships](https://en.wikipedia.org/wiki/Marine_radar), [spacecraft](https://en.wikipedia.org/wiki/Spacecraft), [guided missiles](https://en.wikipedia.org/wiki/Guided_missiles), [motor vehicles](https://en.wikipedia.org/wiki/Motor_vehicle), [weather formations](https://en.wikipedia.org/wiki/Weather_radar), and [terrain](https://en.wikipedia.org/wiki/Terrain-following_radar). A radar system consists of a [transmitter](https://en.wikipedia.org/wiki/Transmitter) producing [electromagnetic waves](https://en.wikipedia.org/wiki/Electromagnetic_wave) in the [radio](https://en.wikipedia.org/wiki/Radio_spectrum) or [microwaves](https://en.wikipedia.org/wiki/Microwave) domain, a transmitting [antenna](https://en.wikipedia.org/wiki/Antenna_(radio)), a receiving antenna (often the same antenna is used for transmitting and receiving) and a [receiver](https://en.wikipedia.org/wiki/Radio_receiver) and [processor](https://en.wikipedia.org/wiki/Data_processing_system) to determine properties of the object(s). Radio waves (pulsed or continuous) from the transmitter reflect off the object and return to the receiver, giving information about the object's location and speed.

In this project we have designed a Radar Based Security System using Arduino UNO. Main aim of the project is to detect enemies surrounded by security places like Ships in seas , aircrafts in the sky. But here we are doing it for the practical purpose so here we have used low config components like MG90 Servo motor, ultrasonic sensor. All you need for this [Arduino Project](https://howtomechatronics.com/arduino-projects/) is an Ultrasonic Sensor for detecting the objects, a small hobbyist Servo Motor for rotating the sensor and an Arduino Board for controlling them. You can watch the following video or read the written tutorial below.

**Working :**

Radar Based Security System using Arduino that is made up with some electronics like Servo motor, Ultrasonic sensor and buzzer. Servo motor will work as a stand for ultrasonic sensors. While ultrasonic is searching for any objects this servo motor will rotate 360 degrees that is scanning 360 degrees of surrounding for a particular object. If any object is present in that particular region we can find that object and also find the distance between them.We can visualize that information in processing applications.



We have to interface Arduino IDE and Processing IDE software, then only we can visualize our Arduino output into the processing software.

* Make sure to change port number in processing code which you get port number in the arduino.
* If you get any errors in Processing code at “**loadFont("OCRAExtended-30.vlw")**” make sure to which font selected in processing ide.
* Select that same font which you mentioned in the processing code at Tools tab.

**Components Required:**

* Arduino UNO
* Ultrasonic Sensor
* Servo
* Buzzer
* LED
* Jumper Wires
* 220 ohms Resistor
* Bread board
* USB Type A to type b

**Software’s:**

* Arduino IDE
* Processing IDE

**Ultrasonic Sensor**

An ultrasonic sensor is an electronic device that measures the distance of a target object by emitting ultrasonic sound waves, and converts the reflected sound into an electrical signal. Ultrasonic waves travel faster than the speed of audible sound (i.e. the sound that humans can hear). Ultrasonic sensors have two main components: the transmitter (which emits the sound using piezoelectric crystals) and the receiver (which encounters the sound after it has travelled to and from the target).With this technology we can detect an object using this device.

**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. If a motor is powered by a DC power supply then it is called a DC servo motor, and if it is AC-powered motor then it is called AC servo motor. For this tutorial, we will be discussing only about the DC servo motor working. Apart from these major classifications, there are many other types of servo motors based on the type of gear arrangement and operating characteristics. A servo motor usually comes with a gear arrangement that allows us to get a very high torque servo motor in small and lightweight packages. Due to these features, they are being used in many applications like toy cars, RC helicopters and planes, Robotics, etc.

**Buzzer:**

Active Buzzer Arduino module produces a single-tone sound when signal is high. To produce different tones use the Passive Buzzer module. The Active Buzzer module consists of a piezoelectric buzzer with a built-in oscillator. It generates a sound of approximately 2.5 kHz when the signal is high.

**LED:**

Light Emitting Diodes (LEDs) are all around us. They are in our homes, our cars, even our phones. LEDs come in a variety of shapes and sizes, this gives designers the ability to tailor them to their product. Any time something electronic lights up, there’s a good chance that an LED is behind it. Their low power and small sizes make them a great choice for many different products as they can be worked into the design more seamlessly to make it an overall better device. LED should list a ‘Forward Voltage’ that defines the amount of voltage required in order to conduct electricity and produce light. If you try and supply anything less than this amount the LED will remain open and non-conductive. Once the voltage dropped across an LED reaches the forward voltage, your LED will light up

**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(3);

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(3);

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;

}

**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 (1920, 1080); // \*\*\*CHANGE THIS TO YOUR SCREEN RESOLUTION\*\*\*

smooth();

myPort = new Serial(this,"COM6", 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.

orcFont = loadFont("OCRAExtended-30.vlw");

}

void draw() {

fill(98,245,31);

textFont(orcFont);

// 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("Object: " + noObject, 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();

}