DEPARTMENT OF
ELECTRONICS AND COMMUNICATION ENGINEERING
College of Engineering and Technology
SRM Institute of Science and Technology

MINI PROJECT REPORT

ODD Semester, 2023-2024

Lab code & Sub Name	:	18ECO108J Embedded System Design using Arduino
Year & Semester	:	3rd YEAR / 5th Semester
Project Title	:	RADAR SYSTEM USING ARDUINO AND ULTRASONIC SENSOR
Lab Supervisor	:	**Dr.V. Padmajothi** AP/ECE
Team Members	:	1. Nishanth.s (RA2111003010829) 2. Shivam Kumar Singh (RA2111003010871)

Particulars	**Max. Marks**	**Marks Obtained**
		Name:
		Register No :
Program and Execution	20	
Demo verification &viva	15	
Project Report	05	
Total	**40**	

Date :

Staff Name :

Signature :

RADAR SYSTEM USING ARDUINO AND ULTRASONIC SENSOR

OBJECTIVE:

The objective of the "Radar System Using Arduino and Ultrasonic Sensor" project is to design and implement a cost-effective radar system that utilizes an Arduino microcontroller and ultrasonic sensors to detect and track objects in real-time. This project aims to provide an affordable and accessible radar solution for various applications, such as security, surveillance, and obstacle detection, by combining the Arduino platform's versatility with the accuracy and reliability of ultrasonic sensors.

ABSTRACT:

This project presents an affordable radar system that combines Arduino and ultrasonic sensors to detect and track objects. The system emits ultrasonic waves, measures their return time, and processes the data in real-time. It offers a simplified radar solution for various applications, making radar technology more accessible to hobbyists, students, and professionals.

INTRODUCTION:

Radar systems have traditionally been associated with advanced military and aviation applications due to their precision in object detection and tracking. These systems often come with a hefty price tag and can be too complex for non-specialists to implement. In an effort to democratize radar technology, this project introduces a novel solution that leverages the versatility of the Arduino platform and the accuracy of ultrasonic sensors to create an affordable and accessible radar system. By doing so, it opens up possibilities for applications like security, surveillance, and obstacle detection in a wide range of settings.

HARDWARE/SOFTWARE REQUIREMENTS:

Hardware Requirements:

1. Arduino Board (e.g., Arduino Uno or Arduino Mega)
2. Ultrasonic Sensors (e.g., HC-SR04 or similar)
3. Servo Motors (for rotating the ultrasonic sensor, if needed)
4. Breadboard and jumper wires

Software Requirements:

1. Arduino Integrated Development Environment (IDE)
2. Arduino software libraries for ultrasonic sensor (e.g., NewPing or similar)
3. A computer with a USB connection for programming the Arduino board
4. Processing IDE for GUI implementation

CONCEPTS/WORKING PRINCIPLE

- The aim of this project is to calculate the distance position and speed of the object placed at some distance from the sensor.

- Ultrasonic sensor sends the ultrasonic wave in different directions by rotating with help of servo motor. This wave travels in air and gets reflected back after striking some object. This wave is again sensed by the sensor and its characteristics is analysed and output is displayed in screen.

- Arduino IDE is used to write code and upload coding in Arduino and helps us to sense position of servo motor and posting it to the serial port along with the distance of the nearest object in its path. The output of sensor is displayed with the help of processing software to give final output in display screen.

APPROACH/METHODOLOGY/PROGRAMS:

Methods:

The methods employed in this project included selecting appropriate hardware components, such as Arduino boards and ultrasonic sensors, and integrating them following manufacturer specifications. Code development involved programming the Arduino to control sensors, process data, and manage optional servo motors for object tracking. Extensive testing and calibration were conducted to fine-tune system accuracy. Comprehensive documentation, including wiring diagrams and code explanations, was created for knowledge sharing, and consideration for future enhancements and safety compliance were integrated into the design process.

Processing for Data Visualization:

Processing is a powerful and versatile open-source software platform and programming language primarily designed for visual arts, creative coding, and data visualization. To use Processing in your radar system Arduino project, you'll need to write code in both Arduino and Processing. You can establish a serial communication link between the two to transfer data in real time. Processing provides a convenient environment for creating graphical user interfaces and data visualizations that can enhance the functionality and user experience of your radar system

Techniques:

In this project, ultrasonic sensors were employed as the primary sensing technique. Ultrasonic sensors emit high-frequency sound waves and measure the time it takes for these waves to bounce off objects and return to the sensor. By analyzing the time delay, the distance between the sensor and the object can be calculated accurately. Additionally, servo motors were utilized to scan the ultrasonic sensor horizontally, expanding the system's detection range. These combined techniques enabled real-time object detection and tracking, making the radar system versatile and affordable for various applications.

Overview of the Code:

The provided code is for an Arduino-based radar system that utilizes an ultrasonic sensor to detect and measure the distance of objects in its vicinity. It employs a servo motor to rotate the ultrasonic sensor, allowing it to scan a specific range. The system continuously rotates the servo from 15 to 165 degrees and back, collecting distance data at various angles. The collected data is then sent to the Serial Port for monitoring.

Structure of the Main Code:

The code utilizes an Arduino-based radar system, incorporating an ultrasonic sensor and a servo motor. It starts by including the Servo library and defining pins for the ultrasonic sensor. In the setup, it initializes the pins, sets up serial communication for data output, and attaches the servo motor. The main loop rotates the servo motor from 15 to 165 degrees and back while measuring distances using the calculateDistance() function. The distance data is sent to the Serial Port, separated by commas and periods. The calculateDistance() function triggers the ultrasonic sensor, measures the duration of the echo pulse, and calculates the distance based on the speed of sound. This code creates a radar-like system that scans a defined angle range and reports distance measurements via the Serial Monitor.

Main Code:

```
// Includes the Servo library
#include <Servo.h>.
// Defines Tirg and Echo pins of the Ultrasonic
Sensor
const int trigPin = 8;
const int echoPin = 7;
// 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(6); // 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();
Serial.print(i); // Sends the current degree into the
Serial Port
Serial.print(",");
Serial.print(distance); // Sends the distance value
into the Serial Port
Serial.print(".");
}
// 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; }
```

Structure of GUI code:

The code is written in the Processing programming language and serves the purpose of visualizing and interacting with data received from a radar system through the Arduino board. The code establishes a serial connection with the Arduino, retrieves angle and distance data from the radar, and then displays this information on a graphical user interface. The radar's sweep is visualized, and detected objects are plotted on the screen, including their distances and angles. Additionally, it shows a range of distance markers, a line representing the radar beam, and textual information about the detected object's angle and distance. The code's primary structure includes setup() for initialization, draw() for rendering the visualization, and various functions for drawing the radar, objects, lines, and text, as well as processing the serial data.

Processing GUI code:

```
import processing.serial.*;
Serial myPort; // Define 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(1366, 768);
  smooth();
  myPort = new Serial(this, "COM5", 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);
```

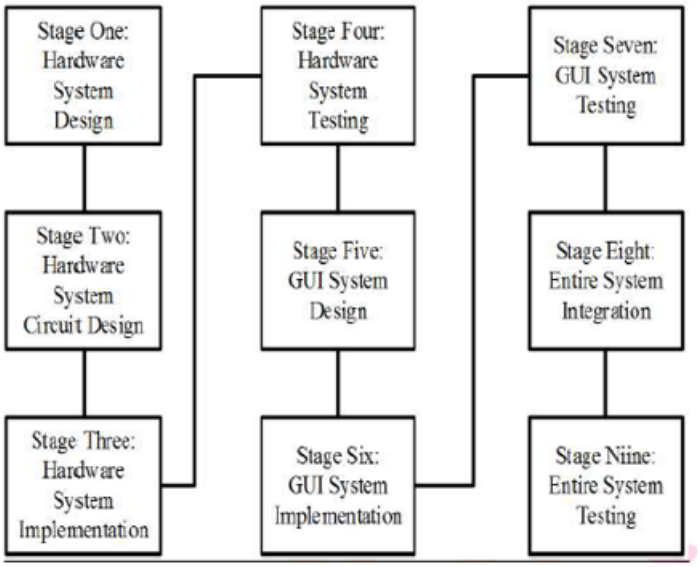
```
  drawRadar();
  drawLine();
  drawObject();
  drawText();
}

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("Shivam kumar singh", 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();
}
```

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

FLOWCHART:



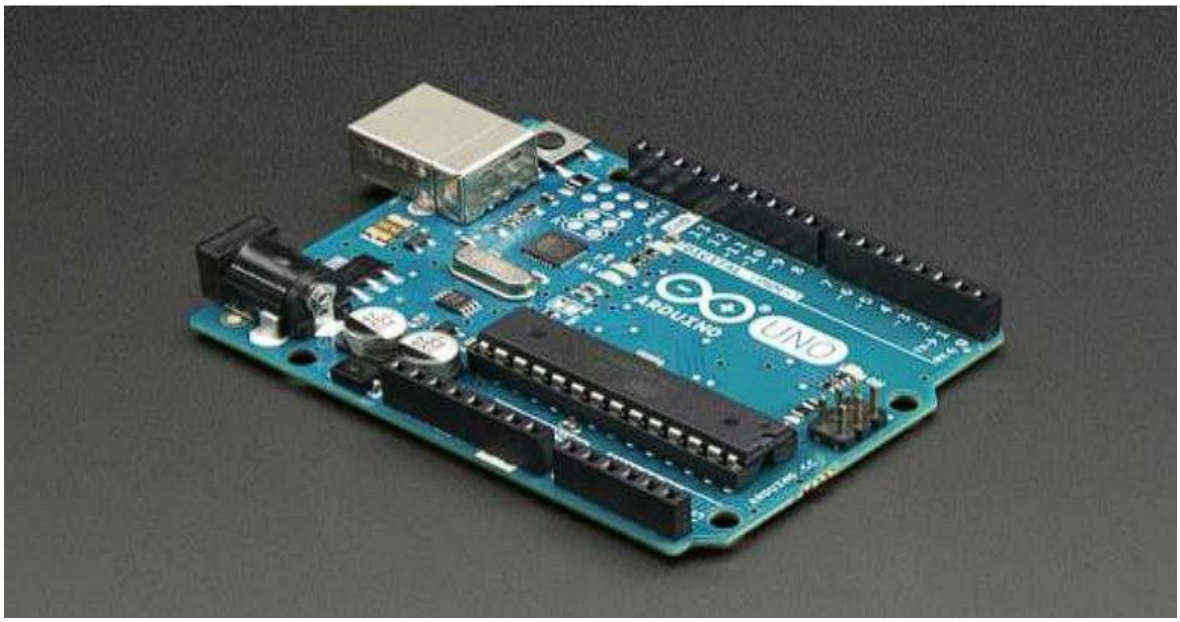
COMPONENTS:

Ultrasonic sensor:

An ultrasonic sensor works similar as of sonar. It can measure distance of object by sending sound waves. Sound waves are send at a specific frequency at a specific direction and listen for sound wave to come back. time taken by sound wave to come back helps us to determine distance of object.

Servo motor:

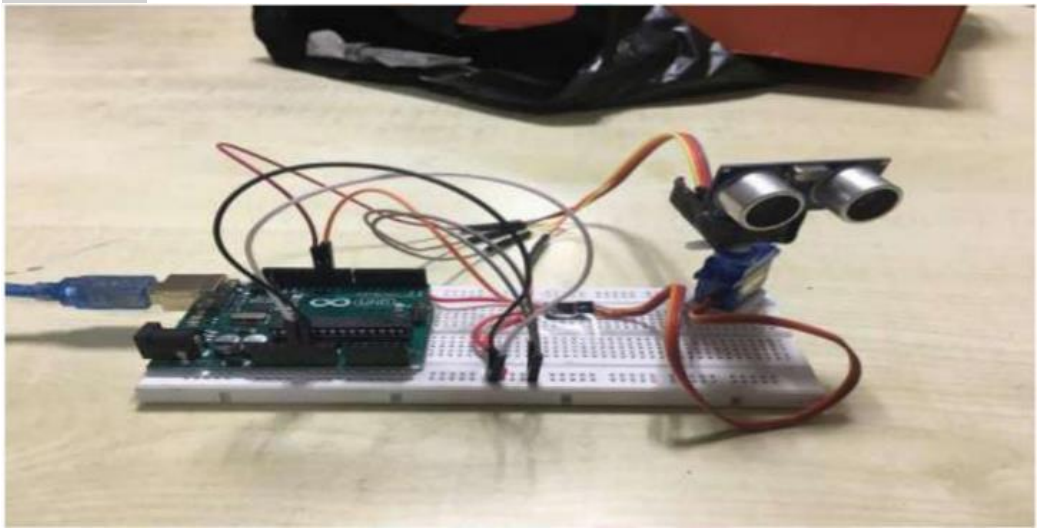
A servomotor is a rotary actuator that allows for precise control of angular position, velocity and acceleration. It consists of a suitable motor coupled to a sensor for position feedback. It also requires a relatively sophisticated controller, often a dedicated module designed specifically for use with servomotors. Servomotors are not a different class of motor, on the basis of fundamental operating principle, but uses servomechanism to achieve closed loop control with a generic open loop motor. Servomotors are used in applications such as robotics, CNC machinery or automated manufacturing.

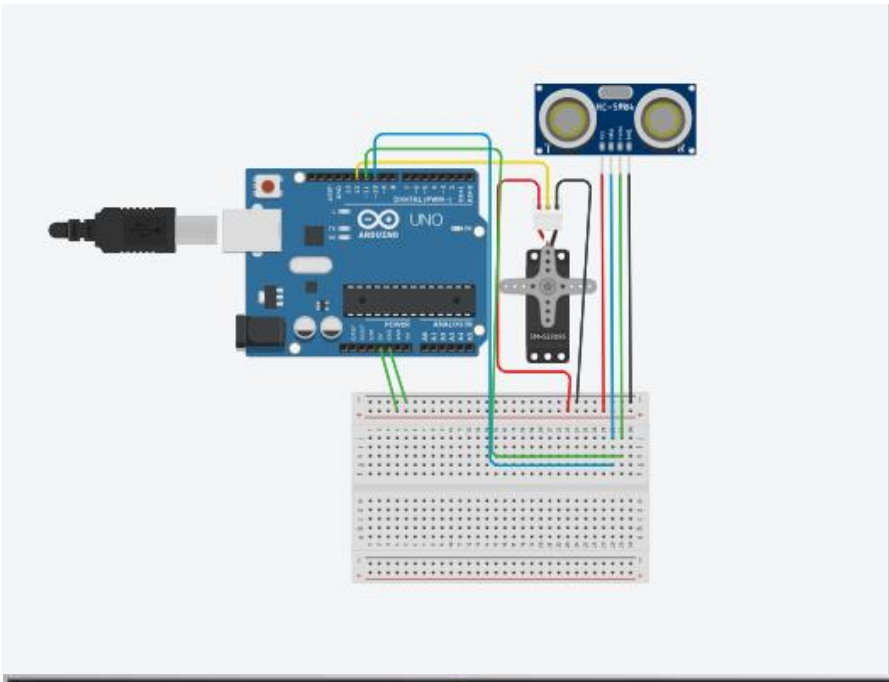


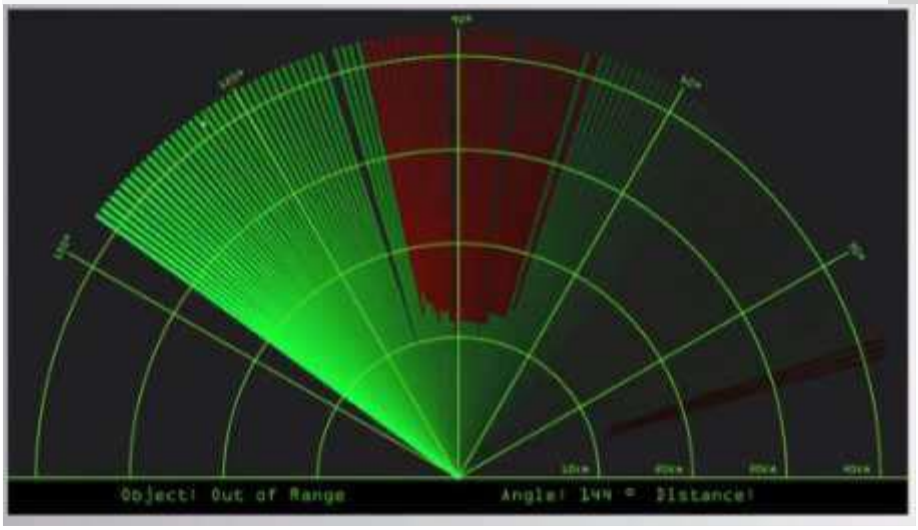
Arduino UNO:

The Arduino is an open source electronics platform based on easy to use hardware and software. The open source Arduino software 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. The Arduino software IDE contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common function.It connects to Arduino and Genuino hardware to upload programs and communicate with them. Program written using Arduino software are called sketches.

OUTPUT:







CONCLUSIONS:

By successfully implementing this radar system, we have demonstrated that complex technologies can be simplified and made more affordable without compromising functionality.

REFERENCES:

https://www.engineersgarage.com/arduino-based-ultrasonic-radar/