DE PBL REPORT

Introduction

In recent years, the need for robust security measures has become increasingly paramount, particularly in sensitive areas such as restricted facilities, critical infrastructure, and high-security zones. To address this demand, ultrasonic radar systems have emerged as a cutting-edge solution for unauthorized object detection, offering reliable detection capabilities that are essential for perimeter security, intrusion prevention, and asset protection. Our project delves into the application of ultrasonic radar technology for unauthorized object detection, exploring its underlying principles, key features, deployment scenarios, and potential benefits in enhancing security protocols across various domains. Through a comprehensive analysis, our report aims to provide insights into the functioning of ultrasonic radar systems as a formidable tool in safeguarding against unauthorized access and ensuring the integrity of secure environments.

Components List

- Positional Micro Servo
- Ultrasonic Distance Sensor (4 pin)
- Piezo
- Arduino Uno
- An LED
- 220 Ω Resistor

Component description

1. Positional Micro Servo

The Positional Micro Servo is a compact motorized device used in robotics and automation to control the angular position of components. It consists of a small DC motor, gears, and a feedback mechanism for precise positioning. Commonly employed in model airplanes, robotics, and other applications requiring precise angular control within a limited range.

1. Ultrasonic Distance Sensor (4 pin)

The Ultrasonic Distance Sensor is a sensor module used to measure distance based on the principle of ultrasonic waves. It typically consists of four pins: VCC (power supply), GND (ground), Trig (trigger), and Echo (echo). The sensor emits ultrasonic waves from the transmitter (Trig pin) and waits for the waves to bounce back from an object. The time taken for the waves to return is calculated by the module and converted into distance using the speed of sound. This distance information is then available through the Echo pin. The sensor is widely used in robotics, automation, and distance measurement applications due to its accuracy, reliability, and ease of integration.

1. Piezo Buzzer

An active piezo buzzer is a type of piezoelectric buzzer that includes an integrated oscillator circuit. This circuit generates the electrical signal required to drive the piezoelectric element, producing sound without the need for an external signal source. Active piezo buzzers are simple to use and commonly found in applications where a standalone sound-producing device is needed, such as alarms, timers, and electronic notification systems. They are compact, lightweight, and energy-efficient, making them suitable for a variety of electronic projects and products.

1. Arduino Uno

The Arduino Uno is an open-source microcontroller board based on the Microchip ATmega328P microcontroller. It can be integrated into a variety of electronic projects.

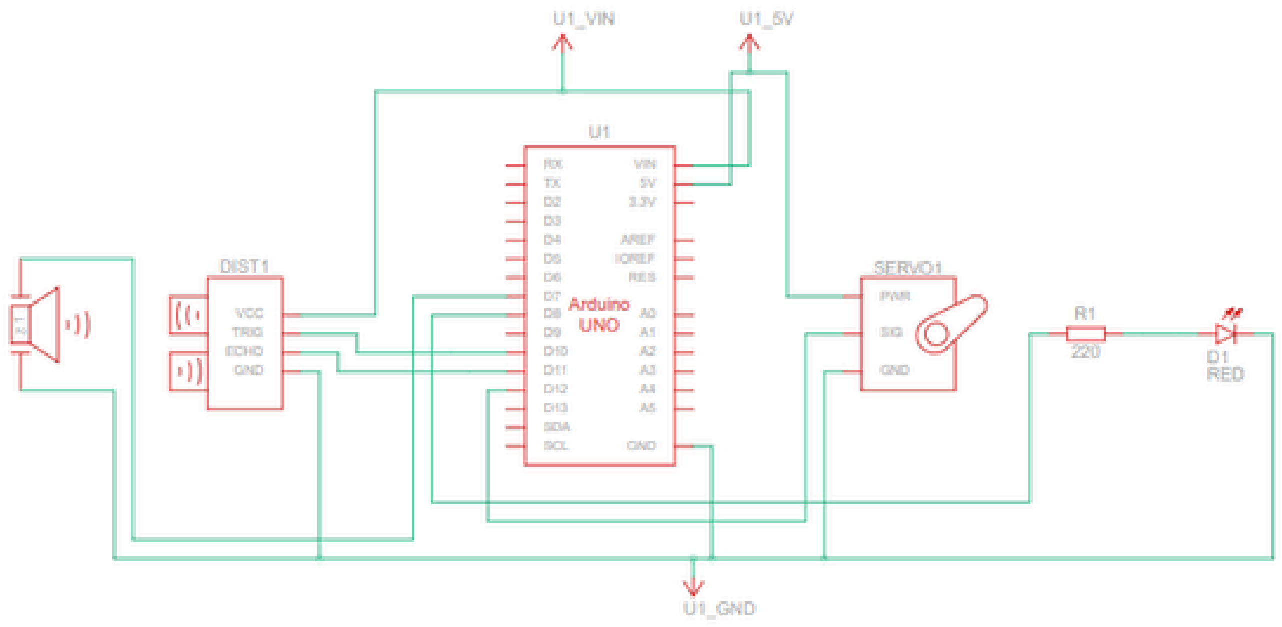
1. LED (Light Emitting Diode)

The major uses of LED (Light Emitting Diodes) are to illuminate objects and even places. Its application is everywhere due to its compact size, low consumption of energy, extended lifetime, and flexibility in terms of use in various applications.

1. Resistor (220 ohm)

A resistor is a type of passive electronic component that limits the flow of electric current in a circuit. It is characterized by its resistance value, which restricts the amount of current passing through it in proportion to the voltage applied. Resistor are commonly used in electronic circuits to protect components, set voltage levels, and control current flow. There compact size, reliability, and affordability make them a widely used component in various electronic devices and applications.

Circuit Diagram



Working

First of all, all the required connections of the resistor, LED, piezo buzzer and Arduino are made on the breadboard using jumper wires. Then below code is uploaded on the arduino which is connected to the computer for power supply. Now as per the uploaded code, the ultrasonic sensor emits high-frequency sound waves and measures the time taken for the waves to bounce back after hitting an object. By rotating the servo motor, the sensor covers a wide range of angles. When an object is detected within a certain range, the LEDs light up, and the buzzer sounds an alarm, alerting the user. In addition to this, we get the exact distance as well as the angle of the unauthorized object for precise tracking of its location.

Code

- Arduino

```
// Includes the Servo library

#include <Servo.h>
```

```
// Defines Trig and Echo pins of the Ultrasonic Sensor

const int trigPin = 10;

const int echoPin = 11;

const int ledPin = 7; // LED pin

const int buzzerPin = 8; // Buzzer pin


// Variables for the duration and the distance

long duration;

int distance;

Servo myServo; // Creates a servo object for controlling the servo motor

bool objectDetected = false; // Flag to track if an object is detected


void setup() {

  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output

  pinMode(echoPin, INPUT); // Sets the echoPin as an Input

  pinMode(ledPin, OUTPUT); // Sets the LED pin as an Output

  pinMode(buzzerPin, OUTPUT); // Sets the buzzer pin as an Output

  Serial.begin(9600);

  myServo.attach(12); // Defines on which pin is the servo motor attached

}


void loop() {

  // Rotates the servo motor from 15 to 180 degrees

  for (int i = 0; i <= 180; 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 additional 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 additional character right next to the previous value needed
later in the Processing IDE for indexing

    if (distance < 30) { // If an object is detected within 30 cm

      objectDetected = true; // Set the flag to true

      digitalWrite(ledPin, HIGH); // Turn on the LED

      digitalWrite(buzzerPin, HIGH); // Turn on the buzzer

      break; // Exit the loop

    }

  }


  if (objectDetected) { // If an object is detected

    while (distance < 30) { // While an object is within 30 cm

      delay(1000); // Wait for 1 second

      distance = calculateDistance(); // Check the distance again

    }

    objectDetected = false; // Reset the flag once the object is removed

    digitalWrite(ledPin, LOW); // Turn off the LED

    digitalWrite(buzzerPin, LOW); // Turn off the buzzer

  }

}


// 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 microseconds

  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

```
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 (1680, 900); // ***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.

}

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("Angle: " + iAngle +" °", width-width*0.52, height-height*0.0277);

  text("Distance: ", width-width*0.33, height-height*0.0277);

  if(iDistance<40) {

  text("        " + iDistance +" cm", width-width*0.27, 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();

}
```

Results

Our project successfully demonstrates the implementation of an ultrasonic radar system for unauthorized object detection. By integrating an ultrasonic distance sensor, a positional micro servo, LEDs, and a buzzer, the system effectively detects and alerts users to the presence of objects within a defined range. The radar sweeps through a 180-degree angle range, detecting objects and providing visual and auditory alerts when unauthorized objects are detected. This project showcases the feasibility and effectiveness of using ultrasonic technology for surveillance and security applications.

Applications

- Military surveillance: Utilizing the radar system to detect and track unauthorized personnel or vehicles in restricted military zones.
- Border security: Deploying the radar to monitor and detect intrusions across national borders or sensitive military installations.
- Perimeter defense: Establishing a radar perimeter around military bases or installations to detect and deter unauthorized entry.
- Asset protection: Safeguarding military assets, such as vehicles, equipment, and facilities, by employing the radar for intrusion detection.
- Force protection: Enhancing the security of military personnel by utilizing radar technology to detect potential threats and intrusions in operational environments.
- Reconnaissance: Supporting military reconnaissance missions by providing real-time information on the movement of objects or individuals in designated areas.
- Battlefield awareness: Integrating radar systems into military operations to enhance situational awareness and provide early warning of approaching threats or adversaries.
- Wildlife monitoring: Tracking the movement of animals or detecting intrusions in protected areas for conservation purposes.
- Monitoring equipment in hazardous environments: Detecting the presence of objects near sensitive machinery to prevent accidents.

References

- **https://nevonprojects.com**
- **https://chat.openai.com**
- **https://youtu.be/F3C1LtXdQ7Q**