**19z604 - EMBEDDED SYSTEMS PROJECT REPORT**

Envicheck System

21z304 - Reagan Benny

21z337 - Roshan Vasudev G

21z341 - Sai Sharan V

22z461 - Ajay Gowtham S

22z464 - Vijayarajan S

BACHELOR OF ENGINEERING

Branch : COMPUTER SCIENCE AND ENGINEERING



APRIL 2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PSG COLLEGE OF TECHNOLOGY

(Autonomous Institution)

COIMBATORE – 641 004

**Problem Statement:**

In various environments, especially those requiring enhanced security or surveillance, there is a need for reliable detection and alert systems to notify individuals of potential intrusions or unauthorized access. Traditional surveillance measures often lack the ability to detect subtle movements or presence in real-time, leading to potential security breaches. To address this challenge, the development of envicheck is proposed.

**Overview**

This project, the envicheck System, is an innovative embedded system designed to detect the proximity movements and disturbances in the environment . Utilizing an Arduino Uno as the microcontroller, this system integrates several sensors including an ultrasonic sensor and a sound sensor, alongside actuators such as a buzzer and a servo motor. This integration allows the system not only to detect objects but also to respond with audible alerts and physical movements, making it applicable for various practical applications such as automated parking assists, robotic alert systems, and interactive exhibits.

**Project Objectives**

**Accurate Object Detection:**

To detect objects within a range of up to 4 meters using an ultrasonic sensor.

**Audible and Mechanical Alerts:**

To provide alerts through a buzzer and indicate object direction via a servo motor based on proximity.

**Sound Triggered Responses:**

Utilize a sound sensor to activate or alter system behavior based on noise in the environment.

**Real-Time Interaction:**

Ensure the system reacts immediately to environmental changes.

**Modularity and Scalability:**

Design the system to be easily integrated with additional sensors or actuators.

**Temperature Monitoring:**

To detect and respond to abnormal temperature changes, enhancing safety and environmental awareness.

**Hardware Components**

**Core Components**

**Arduino Uno:**

The central controller for managing sensor inputs and actuating the buzzer and servo motor.

**HC-SR04 Ultrasonic Sensor:**

Measures the distance to an object by emitting ultrasonic waves and measuring their echoes.

**Temperature Sensor (e.g.,):**

Measures ambient temperature to detect abnormal temperature conditions.

**Sound Sensor:**

Detects ambient sound levels, potentially used for triggering system responses.

**Servo Motor:**

Used to rotate or point towards the direction of detected objects.

**Active Buzzer:**

Emits a sound to alert the presence and proximity of objects.

**Additional Components**

**Jumper Wires:**

For making connections between the Arduino and other components on the breadboard or a PCB.

**Breadboard/PCB:**

For prototyping and final circuit assembly.

**System Design and Architecture:**

**Ultrasonic Sensor:** Attached to digital pins for triggering and echo reception.

**Sound Sensor:** Linked to an analog pin to assess sound levels.

**Temperature Sensor:** Connected to another analog pin to monitor ambient temperatures.

**Servo Motor:** Connected to a PWM-capable digital pin for controlled movements.

**Buzzer:** Tied to a digital pin for operational control.

**Component Connectivity:**

**1. Arduino Uno:**

* Power (5V and GND): Supplies power to all components requiring 5V.

**2. HC-SR04 Ultrasonic Sensor:**

* ﻿﻿VCC to Arduino 5V
* ﻿﻿GND to Arduino GND
* ﻿﻿Trig to Arduino Digital Pin 10
* ﻿﻿Echo to Arduino Digital Pin 11

**3. Sound Sensor:**

* ﻿﻿VCC to Arduino 5V
* ﻿﻿GND to Arduino GND
* ﻿﻿Analog Output to Arduino Analog Pin AO

**4. TMP36 Temperature Sensor:**

* ﻿﻿VCC to Arduino 5V
* ﻿﻿GND to Arduino GND
* ﻿﻿Vout (Output) to Arduino Analog Pin A1

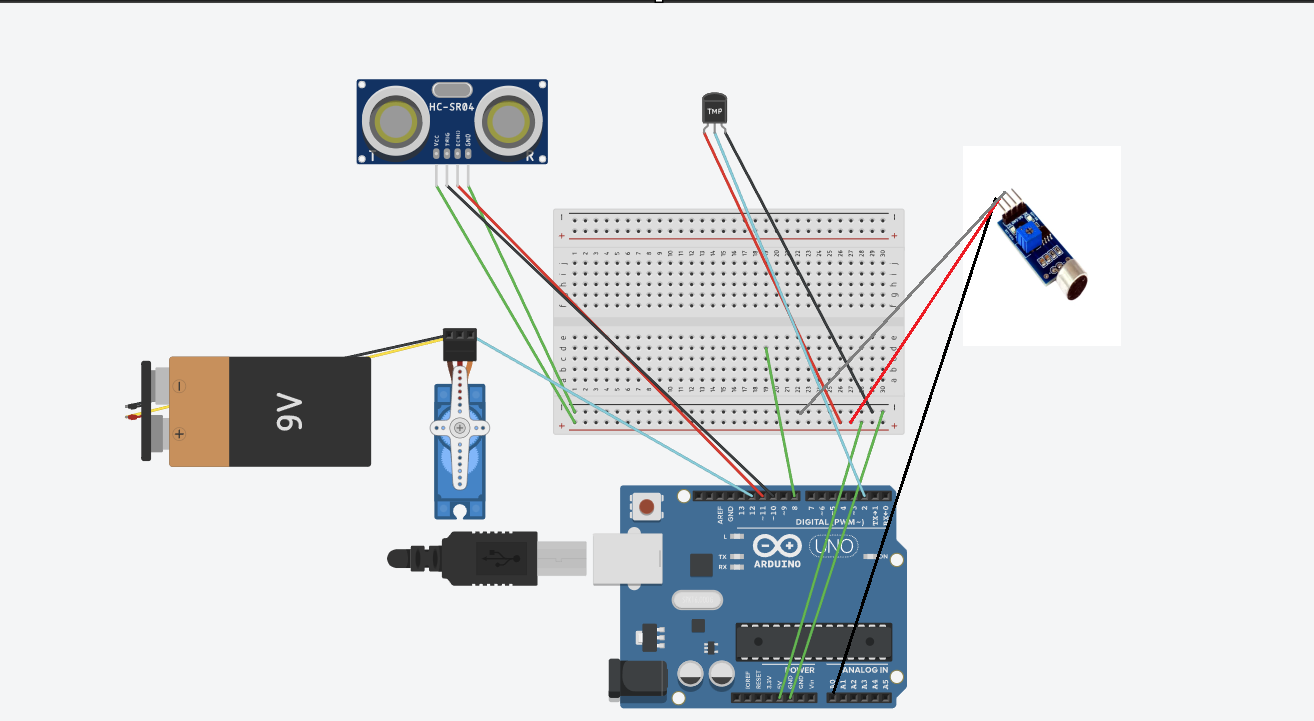
**5. Servo Motor:**

* ﻿﻿Power (Red or Orange) to Arduino 5V
* ﻿﻿Ground (Brown or Black) to Arduino GND
* ﻿﻿Control (Yellow or White) to Arduino Digital Pin 9

**6. Buzzer:**

* ﻿﻿VCC to Arduino Digital Pin 12
* ﻿﻿GND to Arduino GND

**Circuit Diagram:**

****

**Software Implementation:**

**ALGORITHM:**

1. Initialize all components.
2. Read Sensors: Continuously measure distance, sound, and temperature.
3. Process Data:
   * Trigger the buzzer and adjust the servo if an object is detected too close.
   * Activate the buzzer in different patterns or tones if abnormal temperature or loud noises are detected.
4. Actuate Responses: Based on the severity and type of detection.
5. Loop: Maintain continuous monitoring and responsiveness.

**Distance Measurement:** Utilizing the ultrasonic sensor to continuously measure the distance to the nearest object.

**Sound Detection:** Monitoring ambient sound levels via the sound sensor for potential system activation or mode changes.

**Servo Motor Control:** Adjusting the angle of the servo motor based on the location and distance of the detected object.

**Buzzer Control:** Activating the buzzer based on proximity thresholds set in the software.

**CODE:**

**Progressive (For GUI):**

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

// defines 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,"COM14", 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 coordinates 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 coordinates 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 coordinates 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\*0.48, 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();

}

**Arduino IDE :**

// Includes the Servo and lcd library

#include <Servo.h>

#include <dht11.h>

#define DHT11PIN 4

dht11 DHT11;

//Defines lcd pins

// Defines Tirg and Echo pins of the Ultrasonic Sensor

const int trigPin = 10;

const int echoPin = 11;

const int sound= A0;

//Defines piezo pin

const int piezoPin = 8;

// Variables for the duration and the distance

long duration;

int distance;

int notes[] = {262, 462, 862, 1662, 3262}; // Enter here the notes you like

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

pinMode(sound, INPUT);

Serial.begin(9600);

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

}

void loop() {

int soundValue = analogRead(soundSensorPin); // Read the analog value from the Sound Sensor

int threshold = 100;

// rotates the servo motor from 15 to 165 degrees

for(int i=15;i<=165;i++){

myServo.write(i);

if(soundValue > threshold){

tone(piezoPin, notes[1]);

delay(1000);

noTone(piezoPin);

delay(3000);

}

int chk = DHT11.read(DHT11PIN);

if (DHT11.temprature > tempthreshold){

tone(piezoPin, 2500);

delay(1000);

noTone(piezoPin);

delay(5000);

}

distance = calculateDistance();// Calls a function for calculating the distance measured by the Ultrasonic sensor for each degree

//beep sequence

if(distance > 40){

noTone(piezoPin);

delay(10);

noTone(piezoPin);

delay(30);

}

else if (distance <= 40 && distance > 30){

tone(piezoPin, notes[1]);

delay(1000);

noTone(piezoPin);

delay(30);

}

else if (distance <= 30 && distance > 20){

tone(piezoPin,notes[2]);

delay(500);

noTone(piezoPin);

delay(30);

}

else if (distance <= 20 && distance > 10){

tone(piezoPin,notes[3]);

delay(100);

noTone(piezoPin);

delay(30);

}

else {

tone(piezoPin,notes[4]);

delay(10);

noTone(piezoPin);

delay(30);

}

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);

distance = calculateDistance();

if(distance > 40){

noTone(piezoPin);

delay(10);

noTone(piezoPin);

delay(30);

}

else if (distance <= 40 && distance > 30){

tone(piezoPin, notes[4]);

delay(10);

noTone(piezoPin);

delay(30);

}

else if (distance <= 30 && distance > 20){

tone(piezoPin,notes[3]);

delay(10);

noTone(piezoPin);

delay(30);

}

else if (distance <= 20 && distance > 10){

tone(piezoPin,notes[2]);

delay(10);

noTone(piezoPin);

delay(30);

}

else {

tone(piezoPin,notes[1]);

delay(10);

noTone(piezoPin);

delay(30);

}

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

//U(m/s)=dX(m)/dT(s)

//in this case Duration(time)= 2\*Distance/SpeedOfSound=>

//Distance=SpeedOfSound\*Duration/2

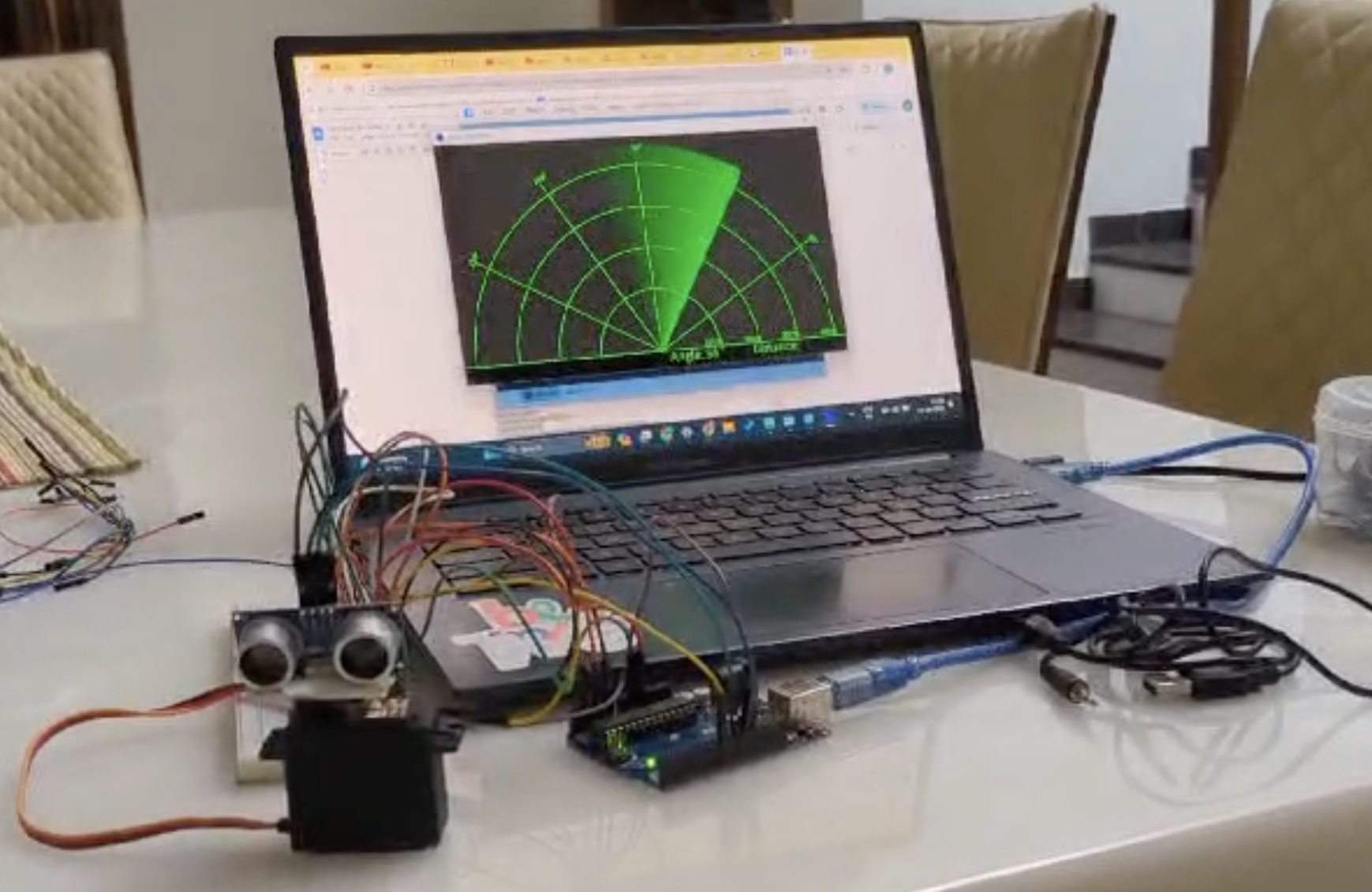
// In dry air at 20 °C, the speed of sound is 343.2 m/s or 0.003432 m/Microsecond or 0,03434 cm/Microseconds

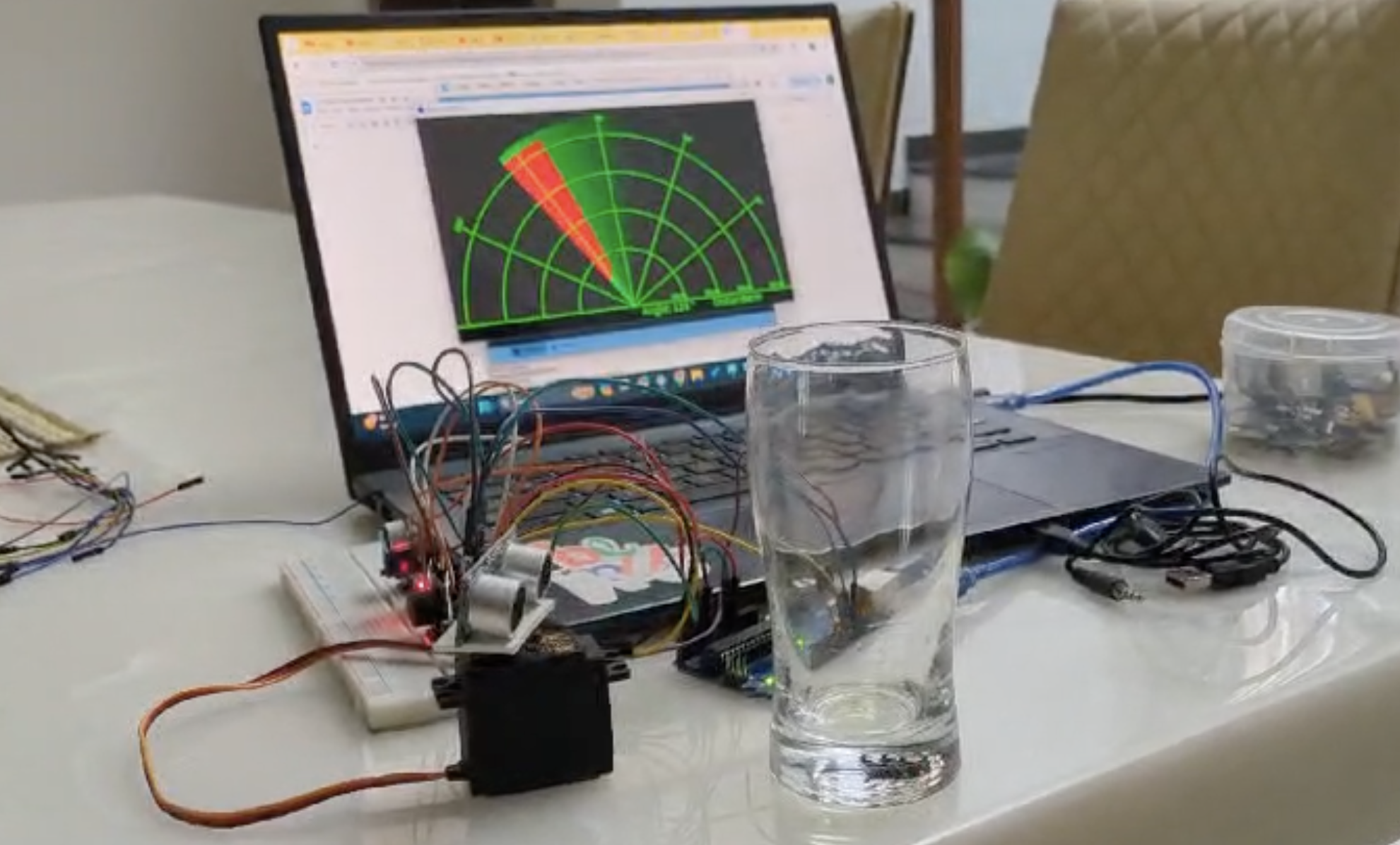
distance= duration\*0.034/2;

return distance;

}

**Calibration and Testing:**

****

****

**CONCLUSION:**

The envicheck System is a versatile, real-time interactive system that effectively combines detection capabilities with mechanical and auditory feedback. Utilizing an Arduino Uno, ultrasonic sensor, sound sensor, temperature sensor, servo motor, and buzzer, it serves both practical applications such as automated parking aids and home safety devices, as well as educational tools for learning about sensor integration and automated response systems.

Overall, the Envicheck System exemplifies the use of modern technology to create smart, responsive systems that enhance safety, efficiency, and interactivity across various environments, laying a solid foundation for future enhancements and demonstrating the potential of embedded systems in addressing real-world challenges.