MAVEN SILICON

Centre of Excellence in VLSI

Project Report for Obstacle detection for blind

Table of contents

1. Student Information

Student Name: UDAYON PAUL

University name and location: Vellore Institute of Technology , VELLORE, Vellore Campus, Tiruvalam Rd, Katpadi, Vellore, Tamil Nadu 632014

University Serial Number (USN) if any:

Submission Date: 11/07/2024

2. Introduction

Improving the mobility and safety of people with visual impairments is the primary goal of creating an "obstacle detection for the blind" system that uses an Arduino Uno, ultrasonic sensors, buzzers, and a power source. The following goals are intended to be attained by this system:

Boost Independence: The technology helps people with visual impairments travel more confidently and independently without substantially depending on others by giving them real-time feedback on barriers.

Boost Safety: Early obstacle detection provides a safer environment for users by reducing the risk of accidents and injuries.

Affordability: A wider range of people may access technology due to its inexpensive cost, which is achieved by using parts like ultrasonic sensors and Arduino Uno.

Simplicity and Reliability: The system's uncomplicated and dependable design makes it easy for individuals without technical experience to use and maintain.

Scalability and Customization: The system may be expanded or combined with other assistive technologies for improved usefulness, and it can be tailored to meet the needs of each particular user.

In conclusion, the goal of developing such a system is to enhance the quality of life for those who are visually impaired by offering an accessible, affordable, and useful obstacle detection solution.

3. Project Specifications

3.1 Function Requirements

- The System will sense the distance of any obstacle in 3 directions at 5 feet height.
- That is obstacles in the front, left and right direction.
- A buzzer is turned on when any obstacle is found at 3 meters distance.
- Buzzer frequency will increase if the distance is decreasing from 3 meters.
- Buzzer frequency will decrease when the obstacle distance increases up to 3 meters.
- Buzzer will stop if the obstacle moves out beyond 3 meters.

3.2 Non-function Requirements

• Performance requirements:
The buzzer should start buzzing within 30 seconds of obstacle detection at 3 meters
• Environment requirements:
The system should work in the temperature range 0 degree centigrade to 60 degrees centigrade
• Note:
This project can be used in vehicles for reverse parking assistance with a change in distance specification to 1 meter instead of 3 meters.

4. Hardware Architecture

Arduino Uno :

Microcontroller Capabilities: The Arduino Uno is based on the ATmega328P microcontroller, which provides a balance of performance and power efficiency suitable for real-time sensor data processing and control tasks required in obstacle detection.

GPIO Availability: The Uno offers 14 digital I/O pins (6 of which can be used as PWM outputs) and 6 analog inputs, providing ample connectivity options for integrating multiple ultrasonic sensors and buzzers necessary for detecting and signaling obstacles.

Processing Speed: Operating at 16 MHz, the ATmega328P provides sufficient processing speed for handling the ultrasonic sensor data, performing necessary calculations, and generating timely responses through the buzzer.

Power Supply Flexibility: The Arduino Uno can be powered via USB or an external power source, with a recommended range of 7 - 12 V, allowing flexibility in choosing power supply options for different deployment scenarios.

Integrated Development Environment (IDE): The Arduino IDE supports straightforward code development and debugging, utilizing a simplified version of C++ which is accessible yet powerful enough for embedded systems programming.

Serial Communication: The Uno features UART, SPI, and I2C communication protocols, facilitating easy communication with other devices or additional modules if needed for system expansion or integration with more complex assistive technologies.

Community and Libraries: A vast ecosystem of libraries exists for the Arduino platform, including libraries for interfacing with ultrasonic sensors (such as the HC-SR04) and controlling buzzers, which speeds up development and ensures reliable operation.

Bootloader: The Uno comes with a pre-installed bootloader, allowing for easy code uploading without needing external hardware programmers, streamlining the development process.

HC-SR04 ultrasonic sensor:

Measurement Range: The HC-SR04 provides a wide detection range from 2 cm to 400 cm, allowing it to detect obstacles at varying distances, which is crucial for providing timely alerts to visually impaired users.

Accuracy: The sensor offers a high level of accuracy with a resolution of approximately 3 mm, ensuring precise distance measurements necessary for reliable obstacle detection.

Operating Frequency: The HC-SR04 operates at 40 kHz, which is inaudible to humans, ensuring that the ultrasonic pulses do not cause any auditory discomfort to the user.

Simple Interface: The HC-SR04 uses a simple trigger and echo pin interface, making it straightforward to integrate with the Arduino Uno. The sensor sends an ultrasonic pulse and measures the time it takes for the echo to return, which can be easily processed by the microcontroller to calculate distance.

Low Power Consumption: The sensor operates with a low current consumption of around 15 mA during measurement, making it suitable for battery-operated systems and ensuring longer operational periods without frequent recharging.

Compatibility with Arduino: There is extensive library support and community knowledge available for integrating the HC-SR04 with Arduino platforms, which simplifies development and troubleshooting.

Power supply (9-volt battery):

Voltage Compatibility: The Arduino Uno operates optimally with an input voltage range of 7 - 12 volts. A 9-volt battery falls within this range, providing a stable and suitable power supply for the microcontroller and connected components.

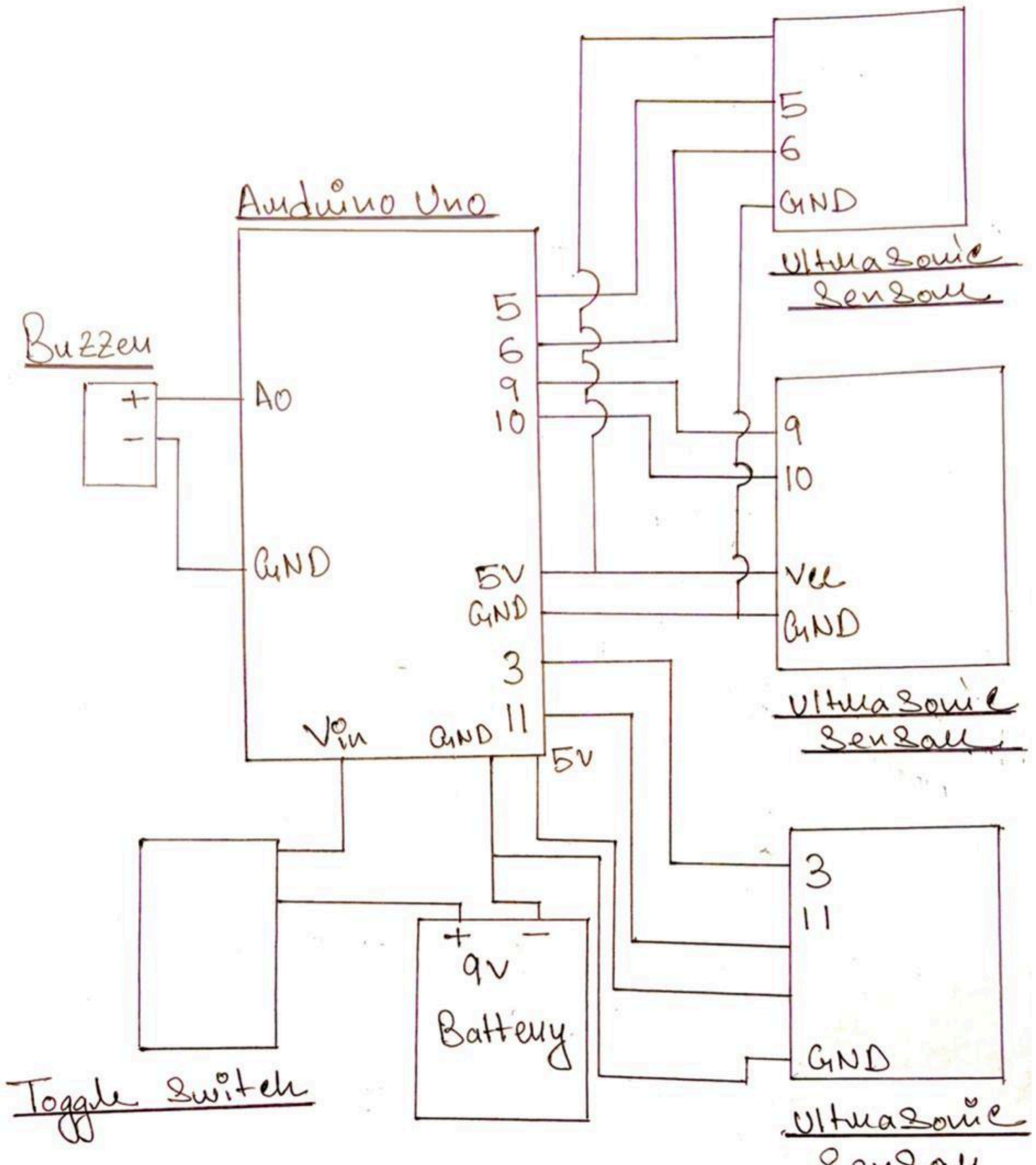
Portability: A 9-volt battery is compact and lightweight, making the entire system easily portable. This is crucial for a wearable or handheld device intended to assist visually impaired users in navigating their environment.

Current Capacity: A typical 9-volt battery can provide sufficient current to power the Arduino Uno and attached peripherals like ultrasonic sensors and buzzers. The Arduino Uno, combined with a few sensors and buzzers, typically requires less than 100 mA, which is well within the discharge capacity of standard 9-volt batteries.

Buzzer :

Audible Feedback: A buzzer provides clear and immediate audible feedback, which is crucial for visually impaired users who rely on auditory cues to perceive their environment. The sound emitted by the buzzer can effectively alert the user to the presence of obstacles.

Udayan Paul


Arduino Uno
Buzzer
+
-
A0
GND
5
6
9
10
5V
GND
3
11
Vin
GND
5V
5
6
GND
Ultrasonic Sensor
9
10
Vcc
GND
Ultrasonic Sensor
3
11
GND
Ultrasonic Sensor
+
-
9V Battery
Toggle Switch

4.1 List of hardware components used:

SERIAL NUMBER	Description of Hardware	Quantity	Remarks, if any
1	Arduino Uno	1	Versatile
2	HC-SR04 ultrasonic sensor:	3	Precise
3	Power supply (9-volt battery)	1	Portable
4	Buzzer	1	Alert

5. Software Architecture:

Main Function (setup and loop):

setup Function:

1. Purpose: This function is called once when the program starts. It is used to initialize variables, pin modes, start using libraries, etc.
2. Importance:It sets up the necessary configurations for the pins used in the program. It ensures that the program starts with the correct settings, which is essential for proper operation.

loop Function:

1. Purpose: This function runs continuously after the setup function has completed. It contains the core logic of the program that needs to run repeatedly.
2. Importance:It continuously measures distances using the ultrasonic sensors. It processes these measurements and controls the buzzer based on the distances. It keeps the system responsive and ensures real-time operation.

   Sub-Function (setFreq):

   setFreq Function:

   1. Purpose: This function sets the frequency and duration for the buzzer.
   2. Importance:It modularizes the code by separating the buzzer control logic from the main loop.It simplifies the loop function, making the code easier to read and maintain. It allows for reusability; the same function can be called with different parameters whenever needed.

pseudocode for the main function

```
Function setup:
    Begin Serial communication at 9600 baud
   Set pinMode of trigpin1 to OUTPUT
   Set pinMode of echopin1 to INPUT
   Set pinMode of trigpin2 to OUTPUT
   Set pinMode of echopin2 to INPUT
   Set pinMode of trigpin3 to OUTPUT
   Set pinMode of echopin3 to INPUT
   Set BUZZpin output to LOW
End Function
Function loop:
   Measure distance from sensor 1
   Measure distance from sensor 2
   Measure distance from sensor 3
   If any distance is between 0 and 30 cm:
      Determine the minimum distance
      Print the minimum distance

      If the minimum distance is between 0 and 5 cm:
         Call setFreq with 255, 50
      Else if the minimum distance is between 5 and 10 cm:
         Call setFreq with 230, 150
      Else if the minimum distance is between 10 and 15 cm:
         Call setFreq with 205, 250
      Else if the minimum distance is between 15 and 20 cm:
         Call setFreq with 180, 350
      Else if the minimum distance is between 20 and 25 cm:
         Call setFreq with 155, 450
   Else:
      Set BUZZpin output to 0
End Function
```

pseudocode for the sub-function

```
Function setFreq(freq, delay1):
    Set BUZZpin output to freq
    Wait for delay1 milliseconds

    Set BUZZpin output to 0
    Wait for delay1 milliseconds
End Function
```

Combined Pseudocode:

```
Function setup:
    Begin Serial communication at 9600 baud

    Set pinMode of trigpin1 to OUTPUT
    Set pinMode of echopin1 to INPUT
    Set pinMode of trigpin2 to OUTPUT
    Set pinMode of echopin2 to INPUT
    Set pinMode of trigpin3 to OUTPUT
    Set pinMode of echopin3 to INPUT
    Set BUZZpin output to LOW
End Function

Function loop:
    Measure distance from sensor 1
    Measure distance from sensor 2
    Measure distance from sensor 3

    If any distance is between 0 and 30 cm:
        Determine the minimum distance
        Print the minimum distance

        If the minimum distance is between 0 and 5 cm:
            Call setFreq with 255, 50
        Else if the minimum distance is between 5 and 10 cm:
            Call setFreq with 230, 150
        Else if the minimum distance is between 10 and 15 cm:
            Call setFreq with 205, 250
        Else if the minimum distance is between 15 and 20 cm:
            Call setFreq with 180, 350
        Else if the minimum distance is between 20 and 25 cm:
            Call setFreq with 155, 450
    Else:
        Set BUZZpin output to 0
End Function

Function setFreq(freq, delay1):
    Set BUZZpin output to freq
    Wait for delay1 milliseconds
    Set BUZZpin output to 0
    Wait for delay1 milliseconds
End Function
```

6. Code

```
int trigoin1 = 5;
int echopin1 = 6;

int trigoin1 = 9;
int echopin1 = 10;

int trigoin1 = 3;
int echopin1 = 11;

int BUZZpin = A0;

void setup()
 {
  Serial.begin (9600);
  pinMode(trigpin1,OUTPUT);
  pinMode(echopin1,INPUT);

  pinMode(trigpin2,OUTPUT);
  pinMode(echopin2,INPUT);

  pinMode(trigpin3,OUTPUT);
  pinMode(echopin3,INPUT);

  analogWrite(BUZZpin, LOW);
 }

   void setFreq(int freq, int delay1)
   {
    analogWrite(BUZZpin, freq);
    delay(delay1);
    analogWrite(BUZZpin, 0);
    delay(delay1);
```

```
}

void loop()
{
 long duration1, distance1;
 digitalWrite(trigpin1,LOW);
 delayMicroseconds(2);
 digitalWrite(trigpin1,HIGH);
 delayMicroseconds(10);
 digitalWrite(trigpin1,LOW);
 duration1 = puseIn(echopin1,HIGH);
 distance = (duration1/2) / 29.1;

 if((distance1 >= 0) && (distance1 <= 30) ||
    (distance2 >= 0) && (distance2 <= 30) ||
    (distance3 >= 0) && (distance3 <= 30))

 {
   long distance = min(min(distance1, distance2), distance3);
   Serial.print("distance:");
   Serial.println(distance);

   if((distance >=0) && (distance <=5))
   {
     setFreq(255, 50);
   }
   else if((distance >=5) && (distance <=10))
   {
     setFreq(230, 150);
   }
   else if((distance >=10) && (distance <=15))
   {
     setFreq(205, 250);
   }
   else if((distance >=15) && (distance <=20))
```

```
    {
      setFreq(180, 350);
    }
    else if((distance >=20) && (distance <=25))
    {
      setFreq(155, 450);
    }
  }
  else
  {
    analogWrite(BUZZpin, 0);
  }
}
```

7. Testing and results:

Testing the possible Conditions and the desired outcomes:

1. No Object Detected by Any Sensor:

Condition: distance1 > 30 and distance2 > 30 and distance3 > 30
Outcome: Buzzer is off.

2. Object Detected by Any One Sensor:

Condition:
0 <= distance1 <= 30 and distance2 > 30 and distance3 > 30
distance1 > 30 and 0 <= distance2 <= 30 and distance3 > 30
distance1 > 30 and distance2 > 30 and 0 <= distance3 <= 30
Outcome: Buzzer frequency and delay are set based on the distance detected by the sensor.

3. Object Detected by Any Two Sensors:

Condition:
0 <= distance1 <= 30 and 0 <= distance2 <= 30 and distance3 > 30
0 <= distance1 <= 30 and distance2 > 30 and 0 <= distance3 <= 30
distance1 > 30 and 0 <= distance2 <= 30 and 0 <= distance3 <= 30
Outcome: Buzzer frequency and delay are set based on the minimum distance detected by the sensors.

4. Object Detected by All Three Sensors:

Condition: 0 <= distance1 <= 30 and 0 <= distance2 <= 30 and 0 <= distance3 <= 30
Outcome: Buzzer frequency and delay are set based on the minimum distance detected by the sensors.

Results:

For each condition where an object is detected by at least one sensor, the specific outcome is determined by the minimum distance. The outcomes are as follows:

Distance Between 0 and 5 cm:

Condition: 0 <= distance <= 5
Outcome:
Buzzer frequency: 255
Delay: 50 milliseconds
Distance Between 5 and 10 cm:

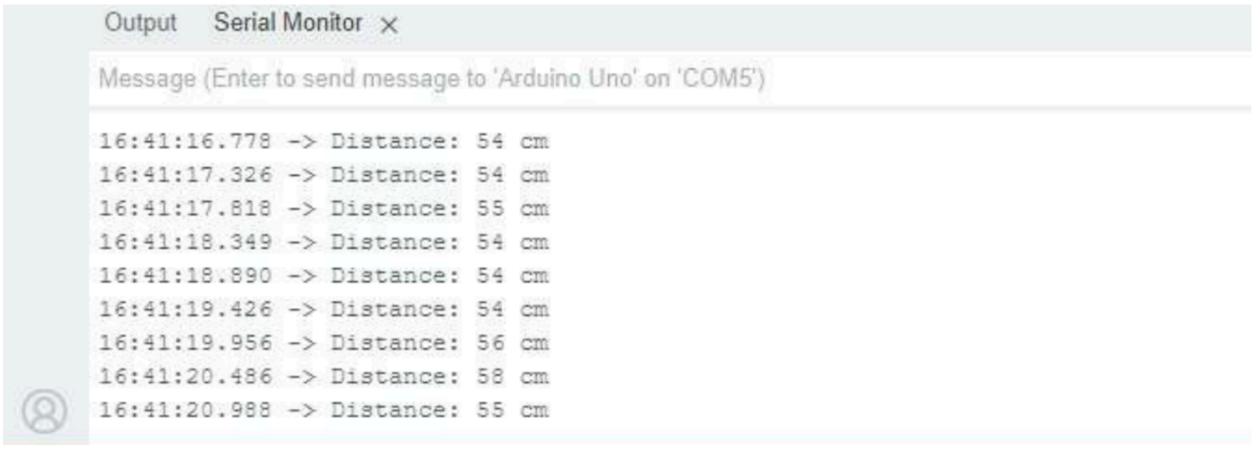
Condition: 5 < distance <= 10
Outcome:
Buzzer frequency: 230
Delay: 150 milliseconds
Distance Between 10 and 15 cm:

Condition: 10 < distance <= 15
Outcome:
Buzzer frequency: 205
Delay: 250 milliseconds
Distance Between 15 and 20 cm:

Condition: 15 < distance <= 20
Outcome:
Buzzer frequency: 180
Delay: 350 milliseconds
Distance Between 20 and 25 cm:

Condition: 20 < distance <= 25
Outcome:
Buzzer frequency: 155
Delay: 450 milliseconds
Distance Between 25 and 30 cm:

Condition: 25 < distance <= 30
Outcome:
Buzzer frequency: 0 (Buzzer off)



```
16:41:16.778 -> Distance: 54 cm
16:41:17.326 -> Distance: 54 cm
16:41:17.818 -> Distance: 55 cm
16:41:18.349 -> Distance: 54 cm
16:41:18.890 -> Distance: 54 cm
16:41:19.426 -> Distance: 54 cm
16:41:19.956 -> Distance: 56 cm
16:41:20.486 -> Distance: 58 cm
16:41:20.988 -> Distance: 55 cm
```

8. Future work:

Possible future improvements:

Hardware Improvements

Additional Sensors:
Integrate more ultrasonic sensors to cover a larger area or to provide more detailed distance measurements from different directions.

Different types of Sensors:
Use different types of sensors (e.g., infrared, LiDAR) to complement ultrasonic sensors and improve accuracy and reliability.

Enhanced Buzzer:
Replace the buzzer with a speaker and use more complex audio signals or tones to indicate distance more clearly.

Software Improvements

Improved Distance Calculation:
Implement more sophisticated algorithms to filter out noise and improve the accuracy of distance measurements.

Signal Processing:
Use signal processing techniques to better interpret sensor data and handle cases where multiple objects are detected.

Dynamic Frequency Adjustment:
Implement a more granular control of the buzzer frequency and delay to provide a smoother transition of tones as the distance changes.

Integration with Other Systems:

Connect the system to other devices or systems (e.g., via Bluetooth, Wi-Fi) to send distance data to a central controller or a smartphone app.

User Interface Improvements

Visual Feedback:
Add LEDs or a display screen to provide visual feedback of the distance measurements along with the auditory feedback.

User Configuration:
Allow users to configure the system parameters (e.g., distance thresholds, buzzer frequencies) via a user interface or a mobile app.

Various applications of the project Obstacle detection for blind:

1. Obstacle Detection for Robots

Modification:
Mount the sensors on a robot to detect obstacles in its path.
Integrate with the robot's control system to enable automatic navigation and collision avoidance.
Enhancement:
Use multiple sensors placed around the robot to provide 360-degree coverage.
Implement more sophisticated algorithms for path planning and obstacle avoidance.

2. Smart Parking Assistance

Modification:
Install sensors on the front and rear bumpers of a car to assist with parking.
Use the buzzer to provide auditory feedback to the driver about the proximity of obstacles.
Enhancement:
Add a visual display on the dashboard to show the distance to obstacles.

Integrate with the car’s infotainment system to provide more detailed alerts and recommendations.

3. Level Measurement in Tanks

Modification:
Place the sensor at the top of a tank to measure the level of liquid inside.
Use the buzzer to alert when the liquid reaches certain levels.
Enhancement:
Connect to a remote monitoring system to provide real-time data on liquid levels.
Implement a control system to automatically fill or empty the tank based on sensor readings.

4. Home Security System

Modification:
Install sensors at entry points (doors, windows) to detect intrusions.
Use the buzzer to sound an alarm when an intruder is detected.
Enhancement:
Integrate with a home automation system to send alerts to the homeowner’s smartphone.
Use additional sensors like infrared, to improve detection accuracy.

5. Proximity Sensing in Industrial Automation

Modification:
Place sensors on machinery to detect the presence of objects or personnel.
Use the buzzer to provide alerts for safety purposes or to indicate operational status.
Enhancement:
Connect to an industrial control system to automate responses based on sensor data.
Implement data logging to track and analyze machine performance over time.