DISTANCE SENSING
USING ULTRASONIC SENSOR

PROJECT REPORT

Submitted by

Gowtham Rajasekaran

Aim

To design and implement a system that detects the distance from an object using an Arduino Uno.

Objective

1. To sense the distance of objects from point using an ultrasonic sensor.
2. To trigger respective LED when an object is detected within given ranges.
3. To create an efficient and reliable proximity detection system.

Principle

It sends an ultrasonic pulse out at 40 kHz, which travels through the air, and if there is an obstacle or object, it will bounce back to the sensor. By calculating the travel time and the speed of sound, the distance can be calculated.

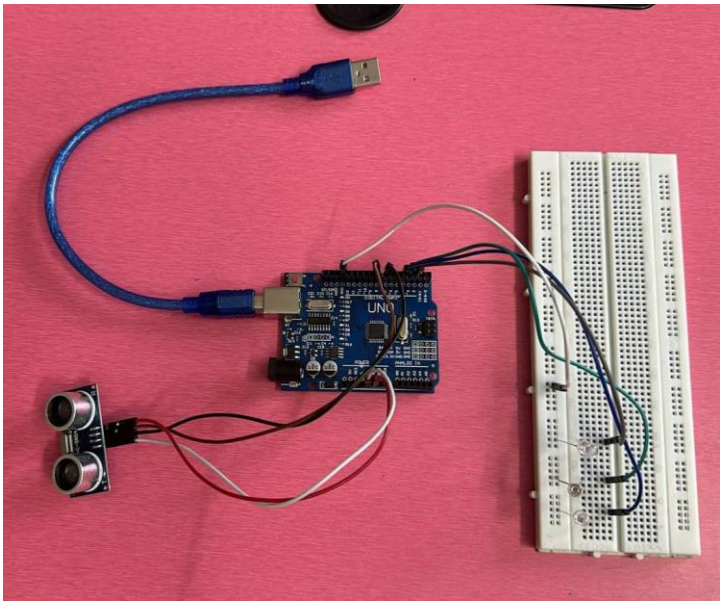
Apparatus Required

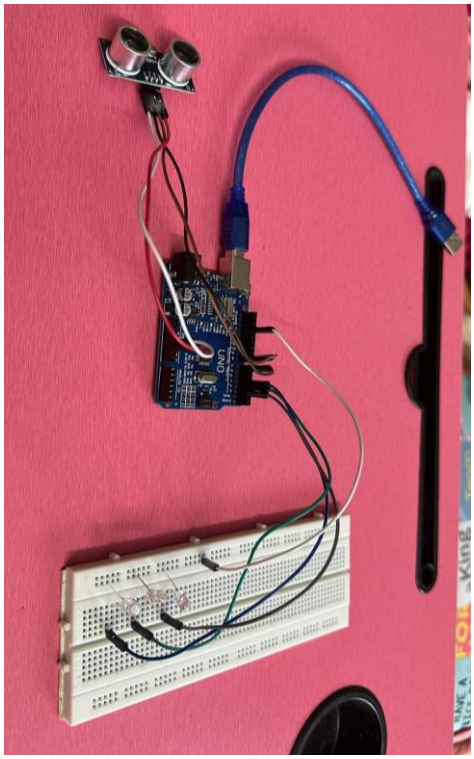
Apparatus	Quantity	Range
Resistor	3	150 - 330 Ω
Buzzer	1	
LED BULB	1,1,1	RED, YELLOW, GREEN
Arduino	1	
Ultrasonic Sensor	1	2-400 cm (Range),
Bread Board	1	
Connecting Wires	A bunch	

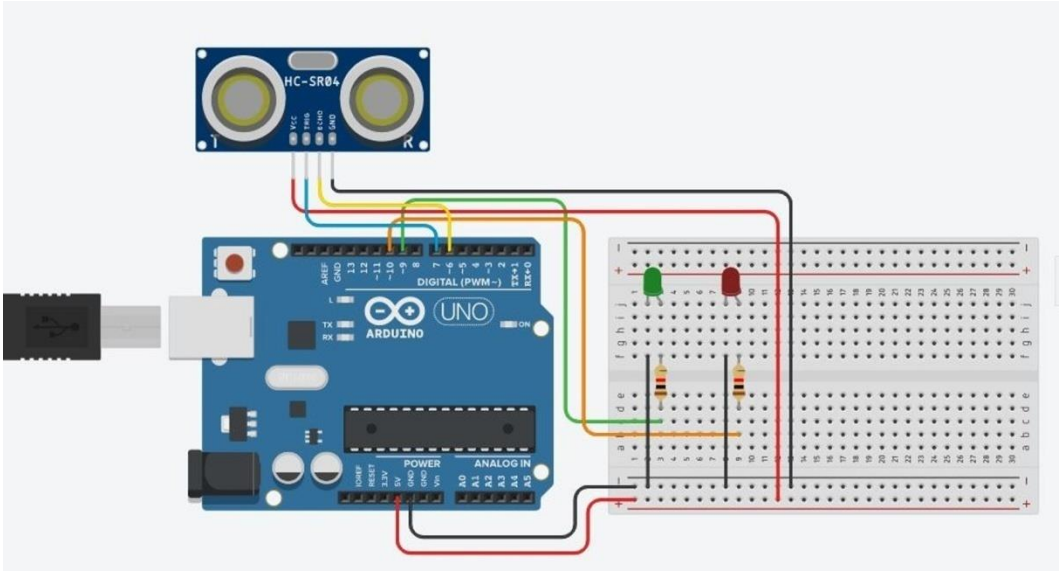
Abstract/Description

The Ultrasonic sensor emits an pulse which is then detected by a receiver in the ultrasonic sensor which returns the time taken for the pulse to after emission this time is then used to calculate the distance from the ultrasonic sensor which is then used to calculate the distance by using the formula:" ((Microseconds / 2) / 29)".

Circuit/System diagram







Working

Hardware Connections:

1. **Ultrasonic Sensor (HC-SR04):**
   - VCC to 5V on the Arduino.
   - GND to GND on the Arduino.
   - trigPin (pin 7) to the Trigger pin on the sensor.
   - echoPin (pin 8) to the Echo pin on the sensor.
2. **LEDs:**
   - Green LED anode to pin 4 through a resistor (typically 220 ohms).
   - Yellow LED anode to pin 2 through a resistor.
   - Red LED anode to pin 3 through a resistor.
   - All LED cathodes to GND.
3. **Buzzer:**
   - Positive terminal to pin 6.
   - Negative terminal to GND.

Steps to Assemble:

1. Connect the ultrasonic sensor to the Arduino as per the connections mentioned above.
2. Connect the LEDs to the Arduino with appropriate resistors to limit current.
3. Connect the buzzer to the Arduino.
4. Power up the Arduino and upload the provided code.

After uploading the code, the system will measure the distance and light up the appropriate LED and sound the buzzer based on the distance detected.

Code

```
int trigPin = 7;
int echoPin = 8;
int green = 4;
int yellow = 2;
int red = 3;
int buzzer = 6;

void setup() {
  Serial.begin(9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(green, OUTPUT);
  pinMode(yellow, OUTPUT);
  pinMode(red, OUTPUT);
  pinMode(buzzer, OUTPUT);
}

void loop() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  long duration, cm;
  duration = pulseIn(echoPin, HIGH);
  delayMicroseconds(100);
  cm = (duration / 29) / 2;
  Serial.println(cm);

  if (cm > 60) {
    noTone(buzzer);
    digitalWrite(green, HIGH);
    digitalWrite(yellow, LOW);
    digitalWrite(red, LOW);
  }
  if (cm <= 60 && cm >= 30) {
    noTone(buzzer);
    if (cm > 30 && cm <= 45) {
      tone(buzzer, 5000);
    }
    digitalWrite(yellow, HIGH);
    digitalWrite(red, LOW);
    digitalWrite(green, LOW);
  }
  if (cm < 30) {
    tone(buzzer, 1000);
    digitalWrite(red, HIGH);
    digitalWrite(yellow, LOW);
    digitalWrite(green, LOW);
  }
}
```

Explanation of the Code:

1. **Pin Definitions:**
    - trigPin: Trigger pin for the ultrasonic sensor.
    - echoPin: Echo pin for the ultrasonic sensor.
    - green, yellow, red: Pins for the green, yellow, and red LEDs, respectively.
    - buzzer: Pin for the buzzer.
2. **Setup Function:**
    - Initializes serial communication.
    - Sets pin modes for the trigger pin, echo pin, LEDs, and buzzer.
3. **Loop Function:**
    - Sends a pulse from the trigger pin.
    - Measures the time it takes for the echo pin to receive the pulse back.
    - Calculates the distance in centimeters.
    - Based on the distance, controls the LEDs and buzzer:
        - If the distance is greater than 60 cm, the green LED lights up.
        - If the distance is between 30 and 60 cm, the yellow LED lights up, and the buzzer sounds at 5000 Hz if the distance is between 30 and 45 cm.
        - If the distance is less than 30 cm, the red LED lights up, and the buzzer sounds at 1000 Hz.

Input and Output

Discussions

Testing the Setup

1. **Power on the Arduino:**
    - Ensure all connections are secure and the Arduino is connected to your computer.
2. **Observe the LEDs and Buzzer:**
    - Place an object at different distances from the ultrasonic sensor.
    - The LEDs and buzzer should respond according to the distance:
        - Green LED lights up if distance > 60 cm.
        - Yellow LED lights up if 30 cm <= distance <= 60 cm (with buzzer sounding at 5000 Hz if 30 cm < distance <= 45 cm).
        - Red LED lights up if distance < 30 cm, and the buzzer sounds at 1000 Hz.

Challenges and Considerations

When working on an Arduino distance sensing project, several challenges and considerations should be kept in mind to ensure a successful and efficient implementation. Here are some potential challenges and considerations:

Challenges:

1. **Sensor Accuracy and Precision:**
    - **Environmental Factors:** Ultrasonic sensors can be affected by temperature, humidity, and air pressure. These factors can cause variations in the speed of sound, affecting the accuracy of distance measurements.
    - **Surface Material:** The sensor might have difficulty accurately measuring distances to surfaces that absorb sound (like soft fabrics) or have irregular shapes.
2. **Interference and Noise:**
    - **Electrical Noise:** Other electronic components and devices can introduce noise, potentially affecting sensor readings and Arduino performance.
    - **Acoustic Interference:** Multiple ultrasonic sensors operating in the same environment can interfere with each other, leading to inaccurate readings.
3. **Power Supply Issues:**
    - **Inadequate Power:** Ensure the power supply is sufficient to run the Arduino and all connected components. Power fluctuations can cause unreliable operation.
    - **Power Consumption:** High power consumption from multiple LEDs and the buzzer can drain the power source quickly if using batteries.

4. **Component Placement and Wiring:**
    - **Wiring Length:** Long wires can introduce resistance and signal degradation. Keep wiring as short as possible.
    - **Secure Connections:** Loose connections can lead to intermittent functionality or failure of components.
5. **Code Efficiency:**
    - **Timing Issues:** The delay and pulseIn functions can introduce timing issues, especially when used with other time-sensitive tasks.
    - **Optimization:** Ensure the code is optimized to run efficiently, especially in real-time applications.

Considerations:

1. **Component Specifications:**
    - **LED Specifications:** Ensure the LEDs' forward voltage and current ratings match the resistors used.
    - **Buzzer Specifications:** Ensure the buzzer can operate at the frequencies specified in the code (5000 Hz and 1000 Hz).
2. **Heat Dissipation:**
    - **Resistors:** Resistors dissipate heat when current flows through them. Ensure they are rated to handle the power they dissipate without overheating.
3. **Sensor Placement:**
    - **Orientation:** Place the ultrasonic sensor in a way that it has a clear path to measure distances without obstructions.
    - **Height and Angle:** Adjust the sensor height and angle for optimal detection of objects at various distances.
4. **Calibration:**
    - **Initial Calibration:** Perform initial calibration to ensure accurate distance measurement.
    - **Regular Calibration:** Periodically recalibrate the sensor to account for any environmental changes or sensor drift over time.
5. **Debugging and Testing:**
    - **Serial Monitoring:** Use the serial monitor for real-time debugging and to check sensor readings.
    - **Unit Testing:** Test individual components (sensor, LEDs, buzzer) separately before integrating them into the final project.
6. **Expandability:**
    - **Future Enhancements:** Design the project with future enhancements in mind. For example, you might want to add more sensors or integrate it with other systems (e.g., IoT platforms).

Implementation Tips

- **Stable Power Supply:** Use a stable and sufficient power source to ensure consistent operation.
- **Noise Reduction:** Use capacitors to filter out noise and reduce interference in power lines.
- **Code Modularity:** Write modular code with functions for each task (e.g., reading the sensor, updating LEDs, controlling the buzzer) to improve readability and maintainability.
- **Physical Enclosure:** Consider an enclosure for your project to protect components from environmental factors and accidental damage.

Conclusion

Using an ultrasonic sensor and Arduino for distance sensing is an effective method for a variety of applications, offering simplicity, low cost, and ease of implementation. While there are challenges related to accuracy and environmental factors, these can often be mitigated through careful setup, programming, and calibration. This project provides a solid foundation for learning about sensor integration, data processing, and real-time measurement systems.

Result

Thus, implemented Distance Sensing with Arduino UNO successfully.

References

1. https://howtomechatronics.com/tutorials/arduino/ultrasonic-sensor-hc-sr04/
2. https://www.geeksforgeeks.org/distance-measurement-using-ultrasonic-sensor-and-arduino/
3. https://www.hackster.io/blackpanda856/controlling-led-s-using-ultrasonic-distance-sensor-19f38f