Ultrasonic Distance Sensing and Buzzer Alert System Using Raspberry Pi

Introduction:

The Ultrasonic Distance Sensing and Buzzer Alert System is a simple yet powerful project that helps in detecting objects within a specific range using an HC-SR04 Ultrasonic Sensor. If an object comes too close, the system triggers a buzzer as an alert.

This project is useful in various real-world applications, such as obstacle detection for robots, security alarm systems, and automatic distance measurement in smart devices. It provides a hands-on experience in working with sensors, GPIO interfacing, and real-time processing using Raspberry Pi.

Working:

1. The HC-SR04 Ultrasonic Sensor sends out high-frequency ultrasonic waves.
2. These waves hit an object and reflect back to the sensor.
3. The sensor records the time taken for the waves to return.
4. The Raspberry Pi processes this data and calculates the distance to the object.
5. If the object is closer than the set threshold (e.g., 10 cm), the buzzer turns ON as a warning.
6. If the object moves away, the buzzer turns OFF.

Components Required:

- Raspberry Pi (any model with GPIO pins)
- HC-SR04 Ultrasonic Sensor
- Buzzer (Piezo or Active)
- Resistors (1kΩ, 2kΩ for voltage divider)
- Breadboard and Jumper Wires
- Power Supply for Raspberry Pi

Fig 1. Ultrasonic Sensor

Circuit Diagram and Connections:

Component	Pin on HC-SR04 / Buzzer	Raspberry Pi Pin
HC-SR04 Ultrasonic Sensor	VCC	5V
HC-SR04 Ultrasonic Sensor	GND	GND
HC-SR04 Ultrasonic Sensor	Trigger (TRIG)	GPIO 23
HC-SR04 Ultrasonic Sensor	Echo (ECHO)	Voltage Divider → GPIO 24
Buzzer	Positive (VCC)	GPIO 18
Buzzer	Negative (GND)	GND

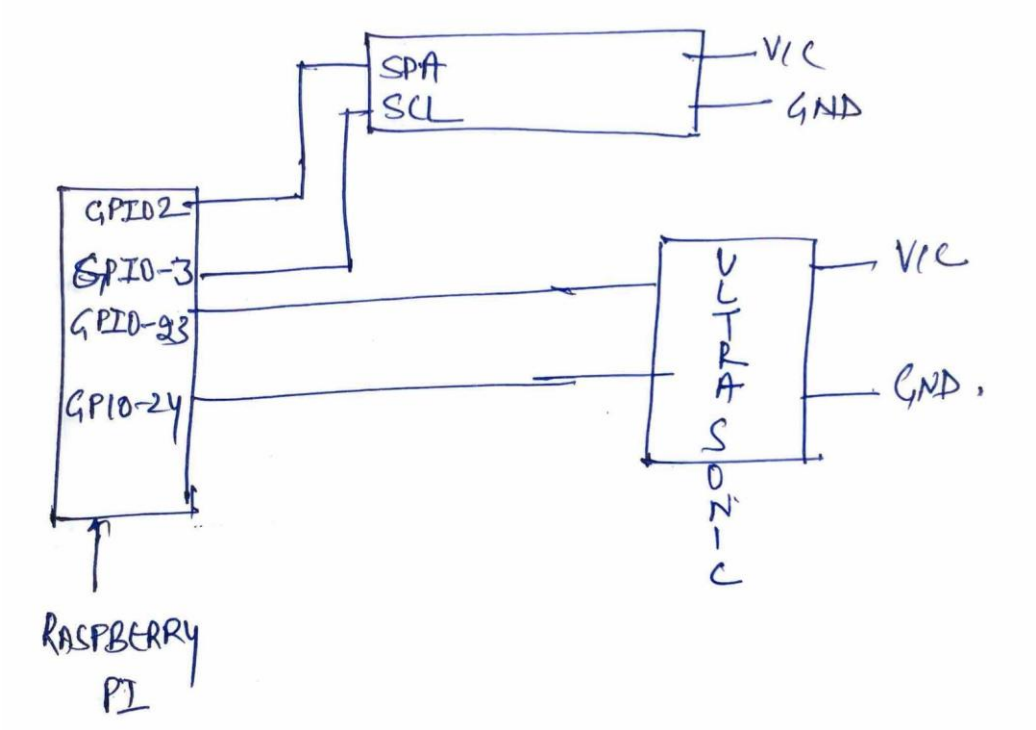

Fig 2. Circuit Connection with Raspberry Pi

Applications:

- Obstacle Detection in Robotics – Helps robots navigate by avoiding obstacles.
- Security Alarm Systems – Detects intruders and triggers alerts.
- Parking Assistance – Assists drivers by detecting nearby objects.
- Smart Waste Management – Detects bin levels for efficient waste collection.

Learnings:

- Understanding ultrasonic distance measurement and signal processing.
- Interfacing HC-SR04 and buzzer with Raspberry Pi.
- Writing Python scripts for real-time sensor data processing.
- Implementing a voltage divider to safely connect 5V sensors to 3.3V GPIO pins.
- Practical application of automation and IoT concepts.

Conclusion:

The **Ultrasonic Distance Sensing and Buzzer Alert System** using **Raspberry Pi** is a practical and efficient project that demonstrates the integration of sensors for real-time object detection. By using the **HC-SR04 ultrasonic sensor**, we can accurately measure distances and trigger a **buzzer alert** when an object comes too close.

Outcome:

Fig 3. Outcome on LED

Program:

```
import RPi.GPIO as GPIO
import time
from RPLCD.i2c import CharLCD
lcd = CharLCD('PCF8574', 0x27)

lcd.clear()

lcd.cursor_pos = (0, 0)  # First row
lcd.write_string(f'Unltasonic Distance')
lcd.cursor_pos = (1, 0)  # First row
lcd.write_string(f'Measurement')

GPIO.setwarnings(False)

# Define GPIO pins
TRIG = 23
ECHO = 24
LED_Pin = 26

# Setup GPIO
GPIO.setmode(GPIO.BCM)
```

```
GPIO.setup(TRIG, GPIO.OUT)
GPIO.setup(ECHO, GPIO.IN)
GPIO.setup(LED_Pin, GPIO.OUT)

def get_distance():
    # Send a short pulse to trigger the sensor
    GPIO.output(TRIG, True)
    time.sleep(0.00001)  # 10µs pulse
    GPIO.output(TRIG, False)

    # Wait for echo signal (start time)
    while GPIO.input(ECHO) == 0:
        start_time = time.time()

    # Wait for echo signal end (stop time)
    while GPIO.input(ECHO) == 1:
        stop_time = time.time()

    # Calculate time difference
    elapsed_time = stop_time - start_time

    # Distance calculation (Speed of sound = 34300 cm/s)
    distance = (elapsed_time * 34300) / 2  # Divide by 2 to get one-way distance
    return round(distance, 2)

try:
    while True:
        dist = get_distance()
        lcd.clear()

        lcd.cursor_pos = (0, 0)  # First row
```

```
        lcd.write_string(f'Ultrasonic Distance')
        lcd.cursor_pos = (1, 0)  # First row
        lcd.write_string(f'Measurement')

        lcd.cursor_pos = (3, 0)  # First row
        lcd.write_string(f'Distance: {dist} cm')

        if 10<dist and dist<35:
            GPIO.output(LED_Pin, GPIO.HIGH)

        else:
            GPIO.output(LED_Pin, GPIO.LOW)

        #print(f"Distance: {dist} cm")
        time.sleep(0.5)

except Exception as e:
    print(f"Error: {e}")

finally:
    GPIO.cleanup()
```