09c Building an Ultrasonic Sensor Alarm with a Keypad and Piezo Buzzer

Nicholas Bruzzese

January 8, 2025

Introduction

In this lesson, you'll learn how to build an alarm system using an ultrasonic sensor, a piezo buzzer, and a 4x4 keypad with a Raspberry Pi. The system monitors distance using the ultrasonic sensor and triggers an alarm when motion is detected. The alarm can only be deactivated by entering the correct PIN code on the keypad.

Objectives

By the end of this lesson, you will:

1. Understand how to use an ultrasonic sensor to measure distance.
2. Learn how to use a keypad for user input.
3. Control a piezo buzzer to simulate an alarm.
4. Combine components to create a functional alarm system.

Hardware Requirements

- Raspberry Pi (any model with GPIO pins).
- HC-SR04 Ultrasonic Sensor.
- 4x4 Keypad.
- Piezo buzzer.
- Jumper wires.

Hardware Setup

Component Connections

Component	Pin Name	GPIO Pin
Ultrasonic Sensor	**TRIG**	GPIO 6
	ECHO	GPIO 5
Keypad	**Row 1 (L1)**	GPIO 5
	Row 2 (L2)	GPIO 6
	Row 3 (L3)	GPIO 13
	Row 4 (L4)	GPIO 19
	Column 1 (C1)	GPIO 12
	Column 2 (C2)	GPIO 16
	Column 3 (C3)	GPIO 20
	Column 4 (C4)	GPIO 21
Piezo Buzzer	Positive	GPIO 18
	Negative	GND

Wiring Diagram

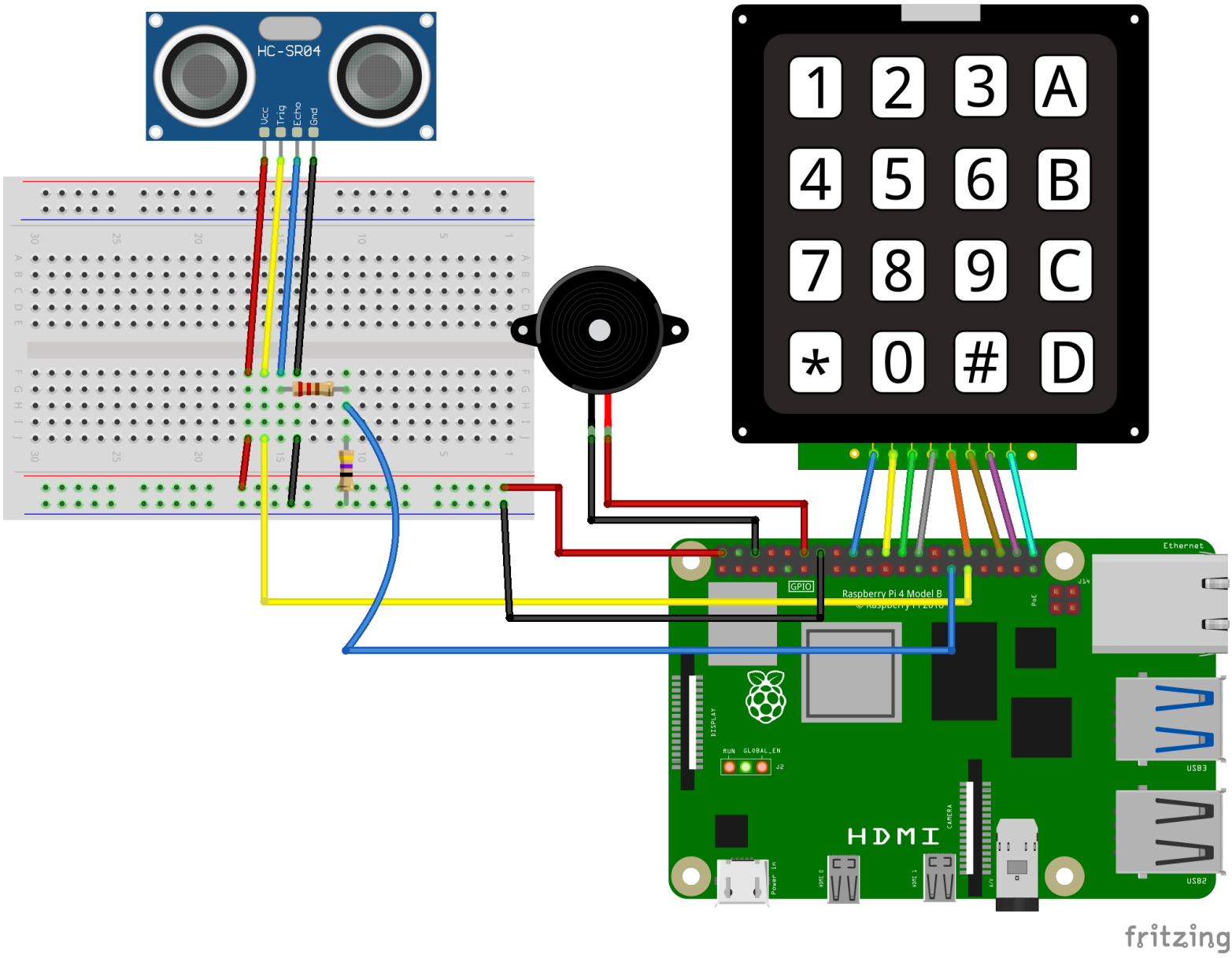

Figure 1: Wiring Diagram

Code Walkthrough

The Python code combines the functionality of the ultrasonic sensor, keypad, and piezo buzzer to create the alarm system.

1. GPIO Initialization

The code initializes the GPIO pins for the ultrasonic sensor, keypad, and buzzer. The ultrasonic sensor measures distance, the keypad accepts user input, and the buzzer serves as the alarm.

```
# Initialize GPIO
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

# Setup ultrasonic sensor GPIO pins
GPIO.setup(TRIG, GPIO.OUT)
GPIO.setup(ECHO, GPIO.IN)

# Setup keypad GPIO pins
GPIO.setup([L1, L2, L3, L4], GPIO.OUT)
GPIO.setup([C1, C2, C3, C4], GPIO.IN, pull_up_down=GPIO.PUD_DOWN)

# Setup buzzer GPIO pin
GPIO.setup(BUZZER, GPIO.OUT)
```

2. Measuring Distance

The `measure_distance()` function sends a pulse from the TRIG pin and calculates the time it takes for the echo to return to the ECHO pin. The function then converts this time into a distance in centimeters.

```
def measure_distance():
"""Measure distance using the ultrasonic sensor."""
GPIO.output(TRIG, True)
time.sleep(0.00001)
GPIO.output(TRIG, False)

while GPIO.input(ECHO) == 0:
pulse_start = time.time()

while GPIO.input(ECHO) == 1:
pulse_end = time.time()

pulse_duration = pulse_end - pulse_start
distance = pulse_duration * 17150 # Speed of sound: 34300 cm/s
return round(distance, 2)
```

3. Activating and Deactivating the Buzzer

The `buzz()` function controls the buzzer state. It turns the buzzer on or off based on the `state` parameter.

```
def buzz(state):
"""Turn the buzzer on or off."""
GPIO.output(BUZZER, state)
```

4. Keypad Input

The `read_line()` function scans the keypad for user input. If a key is pressed, it calls the `process_key()` function to handle the input.

```
def read_line(line, characters):
"""Read input from the keypad."""
global input_code, alarm_active

GPIO.output(line, GPIO.HIGH)
if GPIO.input(C1) == 1:
process_key(characters[0])
if GPIO.input(C2) == 1:
process_key(characters[1])
if GPIO.input(C3) == 1:
process_key(characters[2])
if GPIO.input(C4) == 1:
process_key(characters[3])
GPIO.output(line, GPIO.LOW)
```

5. Processing Keypad Input

The `process_key()` function checks the user input against the predefined PIN code. If the input matches, it deactivates the alarm and stops the buzzer.

```
def process_key(key):
"""Handle key press events."""
global input_code, alarm_active

if alarm_active:
print(f"Key pressed: {key}")
input_code += key

if len(input_code) >= len(PIN_CODE):
if input_code == PIN_CODE:
print("Correct PIN entered. Alarm deactivated.")
buzz(False)
alarm_active = False
else:
```

```
    print("Incorrect PIN. Try again.")
    input_code = ""
```

6. Main Loop

The main loop monitors the ultrasonic sensor for motion and activates the alarm when an object is detected within the threshold distance. It also continuously checks the keypad for PIN entry.

```
    try:
    print("System initialized. Monitoring for motion...")

    while True:
    distance = measure_distance()
    print(f"Distance: {distance} cm")

    if distance < DISTANCE_THRESHOLD and not alarm_active:
    print("Motion detected! Alarm triggered!")
    buzz(True)
    alarm_active = True

    if alarm_active:
    read_line(L1, ["1", "2", "3", "A"])
    read_line(L2, ["4", "5", "6", "B"])
    read_line(L3, ["7", "8", "9", "C"])
    read_line(L4, ["*", "0", "#", "D"])

    time.sleep(0.1)

    except KeyboardInterrupt:
    print("\nApplication stopped!")

    finally:
    GPIO.cleanup()
```

Testing

1. **Trigger the Alarm**: Place an object within 30 cm of the ultrasonic sensor. The buzzer should activate, indicating the alarm is triggered.
2. **Deactivate the Alarm**: Enter the PIN (1234) on the keypad. The buzzer should stop, and the system should return to monitoring mode.

Experiment Ideas

- **Adjust Distance Threshold**: Change the `DISTANCE_THRESHOLD` to make the system more or less sensitive.
- **Add LED Indicators**: Use LEDs to indicate system status (e.g., green for monitoring, red for alarm).
- **Custom Alarm Sound**: Modify the buzzer to play different tones or patterns when triggered.
- **Data Logging**: Record motion detection events to a file or database.

With this setup, you've created a versatile and functional alarm system that combines multiple Raspberry Pi components. Let me know if you have questions or want to expand this project!