CA Assignment - 1

Application : Home Security System

Description:

The "Smart Security System" is a versatile project that utilizes a combination of sensors and output devices to create an intelligent monitoring system. The system is designed to enhance security and provide insights into the environment it's placed in.

The core components of the system include an ultrasonic sensor, a flame sensor (PIR), and a flame sensor. The ultrasonic sensor detects proximity or distance of objects, the flame sensor identifies movement within its range, and the flame sensor measures the flame of moving objects.

When the system detects any unusual activity or specific conditions, it triggers two types of alerts: auditory and visual. An integrated buzzer emits a sound alert, drawing attention to the potential threat, while an LED bulb provides a clear visual indication of the triggered event. This immediate feedback helps users respond promptly to any detected anomalies.

The system can be customized and adapted for various scenarios. For instance, it can be set up at entrances, windows, or areas of interest in homes, offices, or other locations. Additionally, it offers the option of environmental monitoring, enabling users to keep track of changes and activities in the monitored area.

The project involves the use of an Arduino, Raspberry Pi and ESP8266 boards to control and process data from the sensors. The programming logic for the sensors and output devices is coded into the Arduino, allowing the system to autonomously respond to changes in its surroundings. This project not only enhances security but also provides a valuable tool for monitoring and managing the environment.

Components:

1. Hardware Platforms:
   a. Arduino Uno
   b. Raspberry Pi
   c. ESP8266 or Esp 8266

2. Sensors:
   a. Ultrasonic sensor
   b. Flame sensor (PIR)
   c. Flame Sensor

3. Actuators:
   a. Buzzer

b. Relay

Specification of Components:

Ultrasonic Sensor:
- Operating Range: 2 cm to 5 meters or more
- Accuracy: Millimeter-level precision
- Output: Analog voltage, digital pulse width, or serial communication
- Power Supply: Typically 3.3V to 5V
- Application: Robotics, automation, proximity detection

Flame Sensor (PIR - Passive Infrared):
- Detection Range: Typically 5 to 7 meters
- Sensitivity: Adjustable for different flame sizes
- Output: Digital (high/low)
- Delay Settings: Adjustable for controlling how long the output remains high after flame detection ● Application: Security systems, lighting control, occupancy sensing

Flame Sensor:
- Type: Various types like Hall Effect, Optical, Inductive
- Measurement Method: Detects rotational or linear flame
- Output: Typically a digital pulse or analog voltage proportional to flame
- Mounting: Attached to a rotating or moving part of a machine
- Application: Automotive (wheel flame sensors), industrial machinery, robotics

Buzzer:
- Type: Piezoelectric or electromagnetic
- Voltage: Typically 3V to 12V
- Sound Output: 70 dB to 120+ dB
- Frequency: 2 kHz to 4 kHz (audible range)
- Mounting: Through-hole or surface-mount
- Application: Alarms, notifications, signaling

Description of Components:

Arduino Uno:

The Arduino Uno is a microcontroller-based development board designed for building various electronic projects and prototypes. It features an ATmega328P microcontroller, digital and analog pins, and can be programmed using the Arduino programming language. Arduino Uno is known for its simplicity and ease of use, making it a great choice for beginners and hobbyists. It's commonly used for robotics, home automation, sensor interfacing, and more.

Raspberry Pi:

The Raspberry Pi is a series of single-board computers designed to provide affordable computing solutions for various applications. It runs on a variety of operating systems and offers more computational power compared to traditional microcontrollers. Raspberry Pi is equipped with GPIO pins that allow you to interface with sensors, motors, and other hardware. It's widely used for programming, web servers, media centers, educational purposes, and IoT projects.

ESP8266:

The ESP8266 is a powerful Wi-Fi and Bluetooth-enabled microcontroller module. It's part of the ESP8266/ESP8266 family and is known for its versatility and wireless capabilities. The ESP8266 comes with a dual-core processor, a variety of digital and analog pins, and built-in Wi-Fi and Bluetooth connectivity. It's a popular choice for Internet of Things (IoT) projects, allowing you to create devices that can communicate over the internet or with other devices via wireless protocols.

Ultrasonic sensor (HC-SR04):

The ultrasonic sensor (HC-SR04) is a distance measuring device that uses ultrasonic sound waves to determine the distance between the sensor and an object. It emits a sound pulse and measures the time it takes for the pulse to bounce back after hitting an object. This information is used to calculate the distance. It's commonly used in projects for detecting obstacles, measuring distances, and creating proximity-based systems.

Flame Sensor (Passive Infrared :

A **flame detector** is a sensor designed to detect and respond to the presence of a flame or fire. Responses to a detected flame depend on the installation but can include sounding an alarm, deactivating a fuel line (such as a propane or a natural gas line), and activating a fire suppression system. The IR Flame sensor used in this project is shown below, these sensors are also called **Fire sensor module** or **flame detector sensor** sometimes.

Passive Infrared

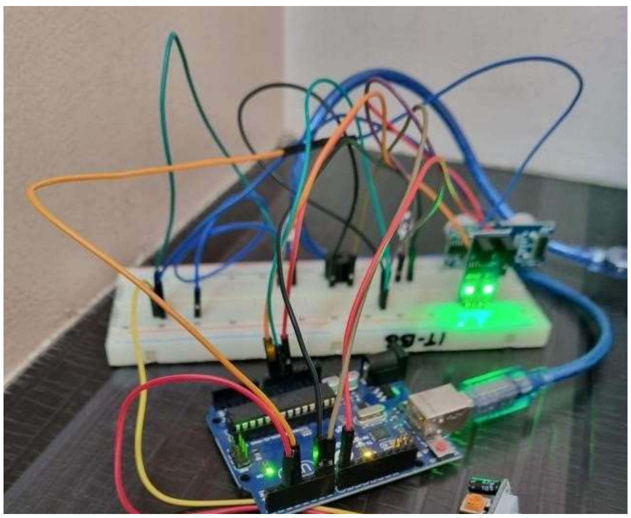
The flame sensor, also known as a Passive Infrared (PIR) sensor, detects changes in infrared radiation in its field of view. It's designed to sense the movement of heat-emitting objects, such as humans or animals. When flame is detected, the sensor generates an electrical signal that can be used to trigger events, such as turning on lights or sounding an alarm. PIR sensors are widely used for security systems, home

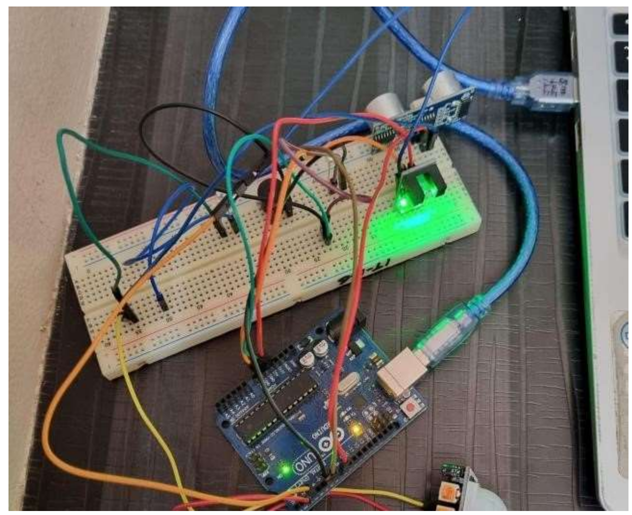
Buzzer:

A buzzer is an electroacoustic transducer that generates sound when an electrical signal is applied to it. It's a simple audio output device used to produce audible alerts or tones. In the project, the buzzer is used to provide an auditory alert when certain conditions are met, such as detecting flame or a change in the environment. Buzzer alerts are commonly found in alarm systems, timers, and notifications.

Interfacing of the components:

1. Arduino Uno





Interfacing of Sensors on Arduino uno board

Code:

```
// defines pins numbers
const int triggerPin = 9;
const int echoPin = 10;
const int buzzerPin = 11;
const int flamePin = 2; // Input from PIR sensor //
// defines variables long
duration;
int distance; int
safetyDistance; int
value = 0; int
pirState = LOW;

void setup() { pinMode(flamePin,
  INPUT); pinMode(triggerPin,
  OUTPUT); pinMode(echoPin,
  INPUT);
  pinMode(pinMode(buzzerPin,
  OUTPUT);

  Serial.begin(9600);
}

void loop() {
  int flameValue = digitalRead(flamePin);

  // Clears the trigPin
  digitalWrite(triggerPin, LOW);
  delayMicroseconds(2);

  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(triggerPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(triggerPin, LOW);

  // Reads the echoPin, returns the sound wave travel time in
  microseconds duration = pulseIn(echoPin, HIGH); distance = (duration
  * 0.0343) / 2;
```

```
// Reads the echoPin, returns the sound wave travel time in
microseconds duration = pulseIn(echoPin, HIGH); distance = (duration
  * 0.0343) / 2;

 safetyDistance = distance; if (
 safetyDistance <= 5)
 { digitalWrite(ledPin2,
 HIGH); tone( buzzer Pin,
 1000); delay(1000);
 }
 else{
   noTone(buzzerPin);
 }

 Serial.print("Distance: ");
 Serial.println(distance);
 delay(1000);
}
```

Code for flame sensor:

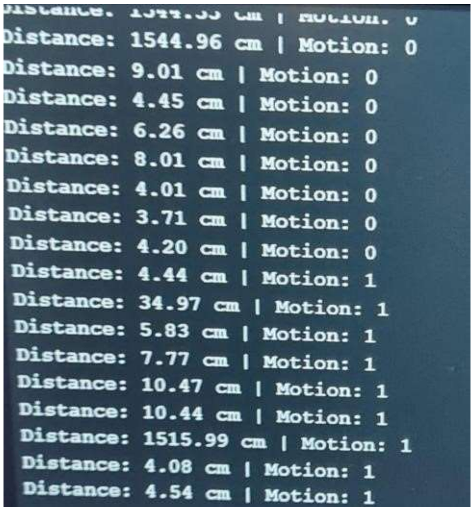
```
const int sensorPin = 2; // Pin that the sensor is connected
to volatile int rpm = 0; // Flame in RPM unsigned long
lastTime = 0;
unsigned int intervals = 0;

void setup() {
  Serial.begin(9600); // Start serial communication pinMode(sensorPin,
  INPUT); // Set the sensor pin as an input
  attachInterrupt(digitalPinToInterrupt(sensorPin), rpmCounter, RISING); // Attach interrupt to the RPM
counter function
}
```

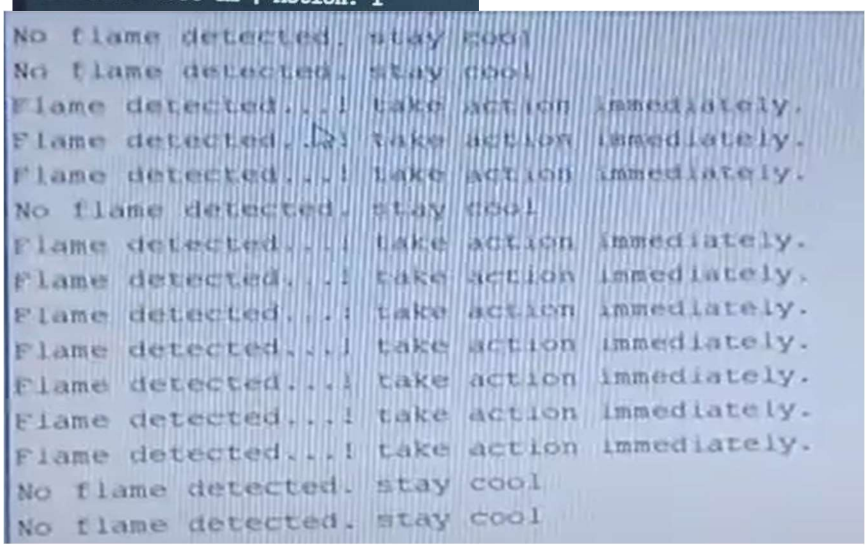
```
void loop() {
  unsigned long currentTime = millis();
  rpm = 60 * 1000 / (currentTime - lastTime) * intervals;
  lastTime = currentTime;
  intervals = 0;

  // Convert RPM to meters per second float wheelRadius =
  0.1; // Example wheel radius in meters float
  metersPerSecond
  = (2 * 3.14 * wheelRadius * rpm) / 60;
  Serial.print("Flame (m/s): ");
  Serial.println(metersPerSecond);

  // Convert meters per second to kilometers per hour
  float kilometersPerHour = metersPerSecond * 3.6;
  Serial.print("Flame (km/h): ");
  Serial.println(kilometersPerHour);
  delay(1000);
} void rpmCounter()
{
intervals++;
}
```

Output:

1) **When an object is detected within a range of 5cms, the blue LED bulb lights up and the buzzer beeps**

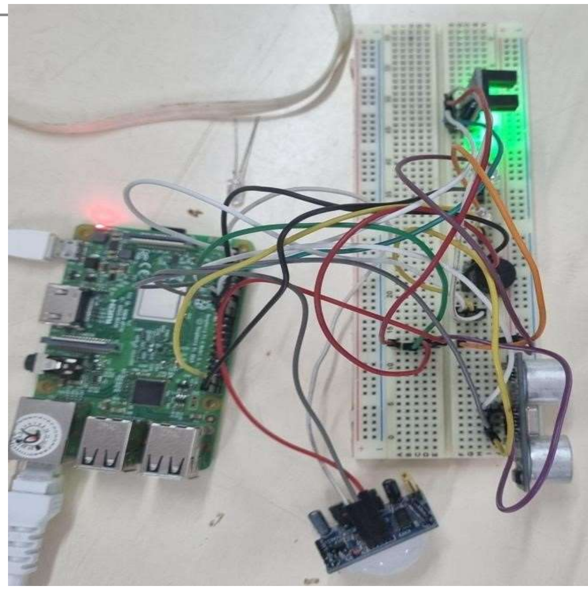
Console output:



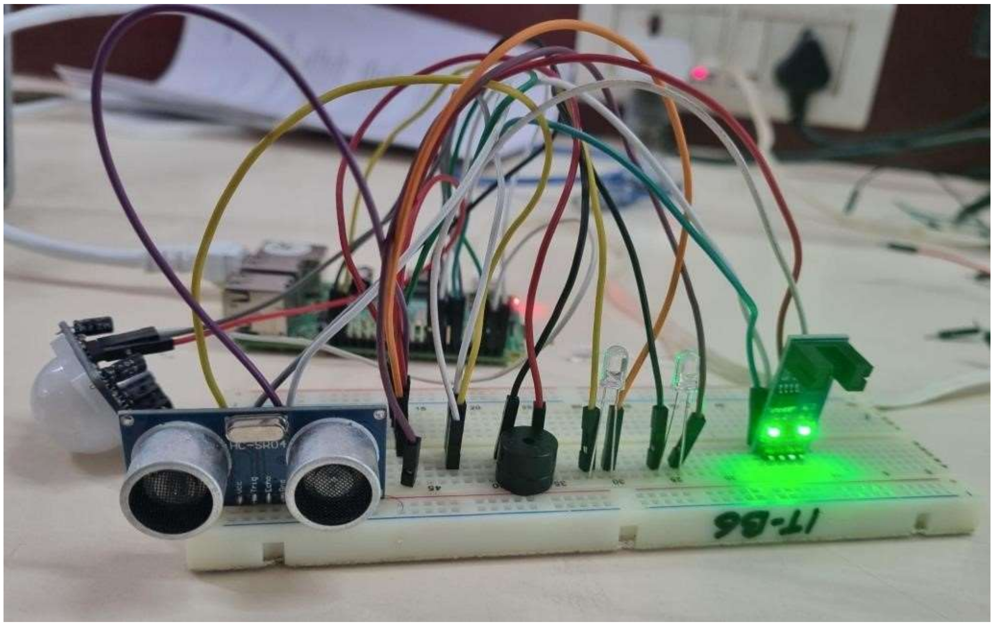


i) Ultrasonic and Flame sensor **ii) Flame sensor**

2. Raspberry Pi



b) Interfacing of Sensors on Raspberry Pi



A) Interfacing of Sensors on Raspberry Pi

Code: import RPi.GPIO as
GPIO import time

# Pin definitions
triggerPin = 14
echoPin = 27
buzzerPin = 26
ledPin1 = 25
ledPin2 = 33
flamePin = 12 # Input from PIR sensor flame_digital_pin
= 23 # Digital input from flame sensor

# Setup GPIO GPIO.setmode(GPIO.BCM)
GPIO.setup(flamePin, GPIO.IN)
GPIO.setup(triggerPin, GPIO.OUT)
GPIO.setup(echoPin, GPIO.IN)
GPIO.setup(ledPin1, GPIO.OUT)
GPIO.setup(ledPin2, GPIO.OUT)
GPIO.setup(buzzerPin, GPIO.OUT)
GPIO.setup(flame_digital_pin, GPIO.IN)
GPIO.setup(flame_analog_pin, GPIO.IN)

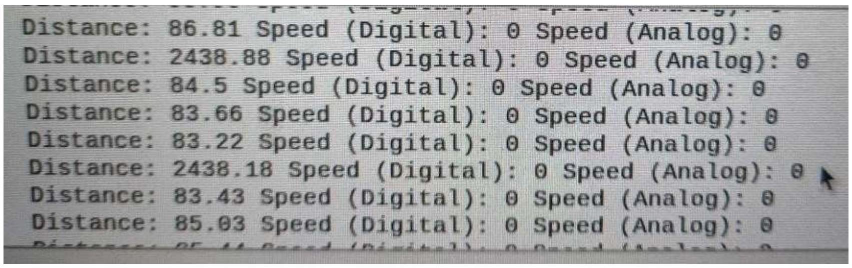
```
def ultrasonic_distance():
    GPIO.output(triggerPin, GPIO.LOW)


    time.sleep(0.2)
    GPIO.output(triggerPin, GPIO.HIGH)
    time.sleep(0.00001)
    GPIO.output(triggerPin, GPIO.LOW)
    while   GPIO.input(echoPin)   ==   0:
    pulse_start = time.time()
    while GPIO.input(echoPin) == 1:
        pulse_end = time.time()
    pulse_duration = pulse_end -
    pulse_start distance = pulse_duration
    * 17150 distance = round(distance, 2)
    return distance

def read_flame_digital():
    return GPIO.input(flame_digital_pin)

# Setup try:
    while True:
        flameValue = GPIO.input(flamePin)

        if flameValue == GPIO.HIGH:
            GPIO.output(ledPin1, GPIO.HIGH)
        else:
            GPIO.output(ledPin1, GPIO.LOW)

        distance = ultrasonic_distance()
        flame_digital_value = read_flame_digital()
        flame_analog_value = read_flame_analog()

        safetyDistance = distance
        if safetyDistance <= 5:
            GPIO.output(ledPin2, GPIO.HIGH)
            GPIO.output(buzzerPin, GPIO.HIGH)
            time.sleep(1)

        else:
            GPIO.output(ledPin2, GPIO.LOW)
            GPIO.output(buzzerPin, GPIO.LOW)
```

```
        print("Distance:", distance, "Flame (Digital):", flame_digital_value, "Flame (Analog):",
  flame_analog_value) time.sleep(1)
  except
     KeyboardInterrupt:
     GPIO.cleanup()
```

Code for flame sensor:

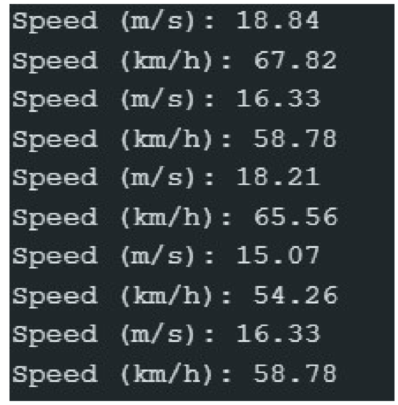
```
  import RPi.GPIO as GPIO import
  time

sensor_pin = 2 # GPIO pin that the sensor is connected to
wheel_radius = 0.1 # Example wheel radius in meters rpm
  = 0
  last_time = time.time()
  intervals = 0 def
  rpm_counter(channel):
     global intervals
  intervals += 1
GPIO.setmode(GPIO.BCM)
GPIO.setup(sensor_pin, GPIO.IN)
  GPIO.add_event_detect(sensor_pin, GPIO.RISING, callback=rpm_counter) try: while
     True: current_time =
        time.time()
        rpm = 60 * 1000 / (current_time - last_time) * intervals
        last_time = current_time
        intervals = 0
        # Convert RPM to meters per second
        meters_per_second = (2 * 3.14 * wheel_radius * rpm) /
        60 print("Flame (m/s):", meters_per_second) # Convert
        meters per second to kilometers per hour
        kilometers_per_hour = meters_per_second * 3.6
        print("Flame (km/h):", kilometers_per_hour)
        time.sleep(1)
  except KeyboardInterrupt:
     GPIO.cleanup() # Clean up GPIO on Ctrl+C
```

Console output:



```
Distance: 86.81 Speed (Digital): 0 Speed (Analog): 0
Distance: 2438.88 Speed (Digital): 0 Speed (Analog): 0
Distance: 84.5 Speed (Digital): 0 Speed (Analog): 0
Distance: 83.66 Speed (Digital): 0 Speed (Analog): 0
Distance: 83.22 Speed (Digital): 0 Speed (Analog): 0
Distance: 2438.18 Speed (Digital): 0 Speed (Analog): 0
Distance: 83.43 Speed (Digital): 0 Speed (Analog): 0
Distance: 85.03 Speed (Digital): 0 Speed (Analog): 0
```

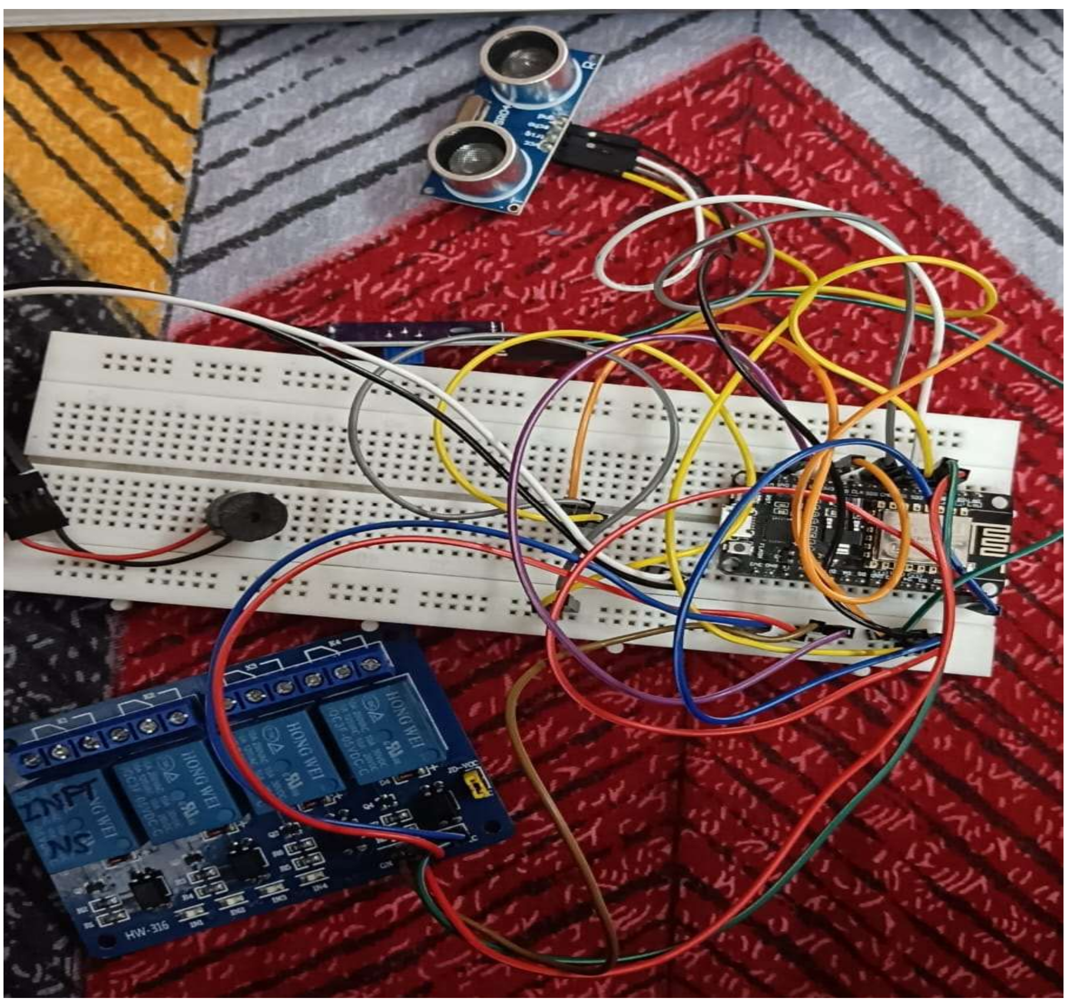
i) Ultrasonic Sensor



```
Speed (m/s): 18.84
Speed (km/h): 67.82
Speed (m/s): 16.33
Speed (km/h): 58.78
Speed (m/s): 18.21
Speed (km/h): 65.56
Speed (m/s): 15.07
Speed (km/h): 54.26
Speed (m/s): 16.33
Speed (km/h): 58.78
```

ii) Flame Sensor

3. ESP8266



A) Interfacing of Sensor on ESP8266

Code:

```
#define TRIG_PIN 12
#define ECHO_PIN 14
#define BUZZER_PIN 27
#define FLAME_PIN 26 #define
LED_PIN_1 25 #define
LED_PIN_2 33

void setup() {
  pinMode(TRIG_PIN, OUTPUT); pinMode(ECHO_PIN,
  INPUT);

  pinMode(LED_PIN_1, OUTPUT);
  pinMode(LED_PIN_2, OUTPUT);

  analogWrite(BUZZER_PIN, LOW);
  digitalWrite(LED_PIN_1, LOW);
  digitalWrite(LED_PIN_2, LOW);
  Serial.begin(9600);
}

void loop() { // Ultrasonic Sensor
  digitalWrite(TRIG_PIN, LOW);d
  elayMicroseconds(2);di
  gitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW); long
  duration = pulseIn(ECHO_PIN, HIGH);
  float distance = duration * 0.034 / 2;

  // Buzzer and LED based on ultrasonic sensor
  if (distance < 10) {
    tone(BUZZER_PIN, 1000);
    analog Write( BUZZER_PIN,
    128); digitalWrite(LED_PIN_1,
    HIGH);
  } else { analogWrite(BUZZER_PIN,
    0); digitalWrite(LED_PIN_1, LOW);
  }
```

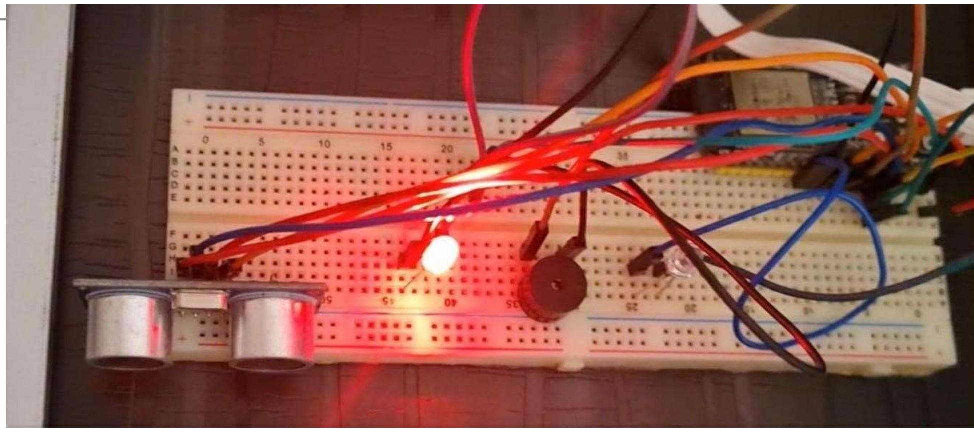
```
// Flame Sensor if (digitalRead(FLAME_PIN)
== HIGH) { digitalWrite(LED_PIN_2, HIGH);
    } else {
      digitalWrite(LED_PIN_2, LOW);
    }

    Serial.print("Distance: ");
    Serial.print(distance);
    Serial.print(" cm | Flame: ");

    Serial.println(digitalRead(FLAME_PIN)); delay(500); //

    Adjust this delay as needed

  }
```

Code for flame sensor:

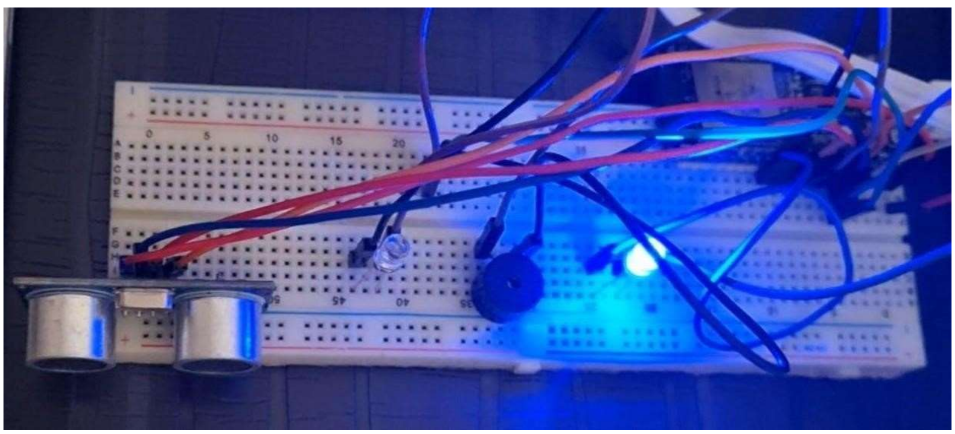
```
const int sensorPin = 26; // Pin that the sensor is connected to
volatile int rpm = 0; // Flame in RPM unsigned
long lastTime = 0;
unsigned int intervals = 0;

void setup() {
  Serial.begin(9600); // Start serial communication pinMode(sensorPin,
  INPUT); // Set the sensor pin as an input
  attachInterrupt(digitalPinToInterrupt(sensorPin), rpmCounter, RISING); // Attach interrupt to the RPM
counter function
}

void loop() {
  unsigned long currentTime = millis();
  rpm = 60 * 1000 / (currentTime - lastTime) * intervals;
  lastTime = currentTime;
  intervals = 0;
  float wheelRadius = 0.1; // Example wheel radius in meters
  float metersPerSecond = (2 * 3.14 * wheelRadius * rpm) / 60; // Convert RPM to meters per second
  Serial.print("Flame (m/s): "); Serial.println(metersPerSecond);
  float kilometersPerHour = metersPerSecond * 3.6; // Convert meters per second to kilometers per hour
  Serial.print("Flame (km/h): "); Serial.println(kilometersPerHour); delay(1000);
}
```

```
void rpmCounter() {
  intervals++;
```

Output:

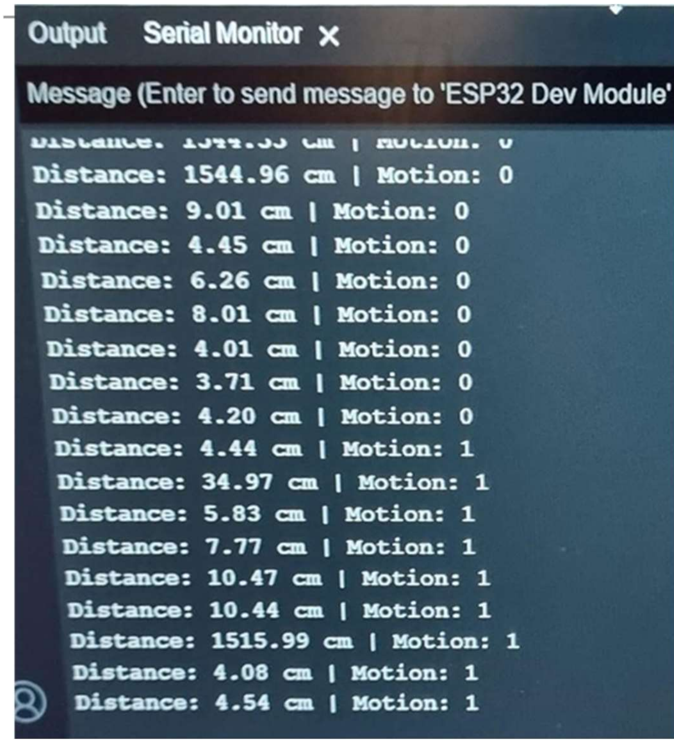


i) When a flame is detected, the buzzer beeps.

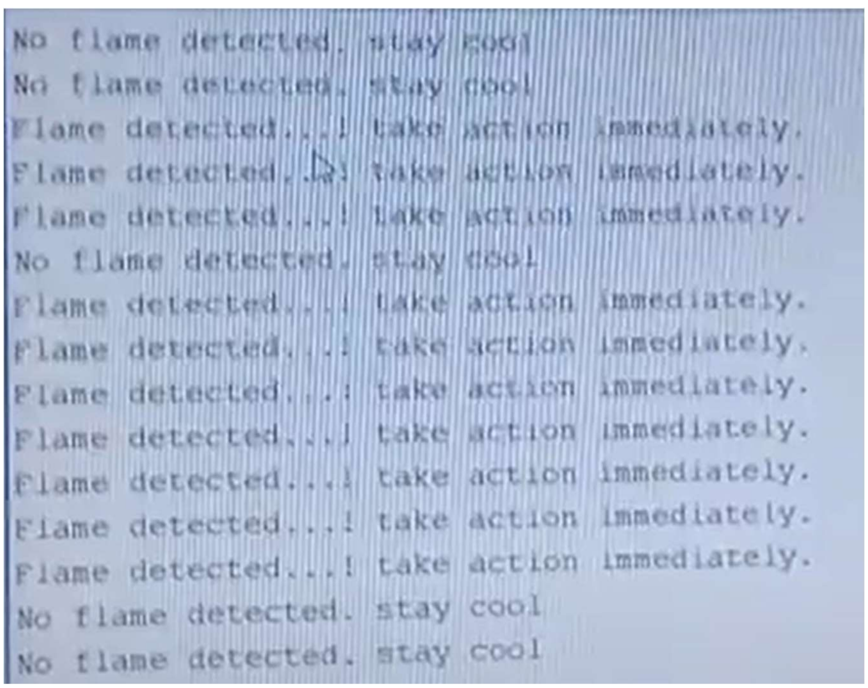


ii) When an object is detected within a range of 5cms, the blue LED bulb lights up and the buzzer beeps

Console Output:



i) Ultrasonic and Flame Sensor



Conclusion:

Hence, we have successfully completed the interfacing of three sensors and two actuators with three different hardware platforms.