

**19CSE213- OS CAPSTONE REPORT**



**Multi-Processing Fire Detection System using Raspberry Pi**

**1) Team Members**:

|  |  |  |
| --- | --- | --- |
| **S. No** | **Name** | **Roll no:** |
| 1. | Aman Kshetri | CH.EN.U4CSE21003 |
| 2. | Raj Sah Rauniyar | CH.EN.U4CSE21051 |
| 3. | Saroj Kumar Bhagat | CH.EN.U4CSE21060 |
| 4. | Tej Mahesh | CH.EN.U4CSE21041 |

**2) Problem Definition**:

The problem we are trying to solve with this project is to detect fires in real-time using a **Raspberry Pi-based system**. Fires can cause significant damage to property and loss of life if not detected and addressed quickly. Therefore, our project aims to detect fires as quickly as possible using sensors connected to a Raspberry Pi, which will alert users through a buzzer and notification messaging using API's, allowing them to take appropriate action.

**3) Methodology**:

The methodology for this project involves using a Raspberry Pi and a fire sensor to detect fires in real-time.

The fire sensor will detect the presence of flames and send a signal to the Raspberry Pi.

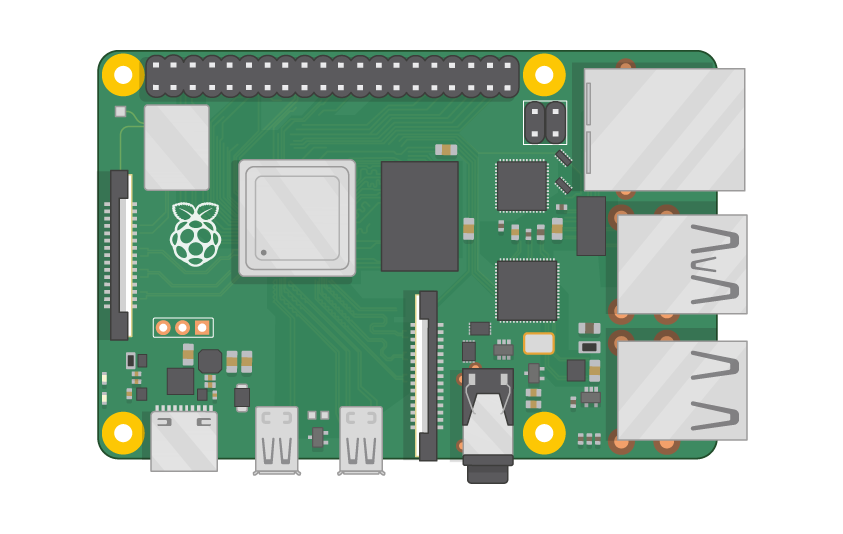
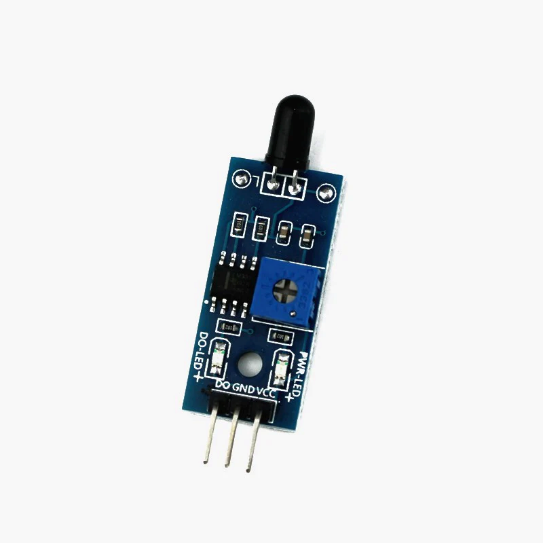
The Raspberry Pi will then process this signal and use a multi-processing approach to analyze the data and determine whether a fire is present.

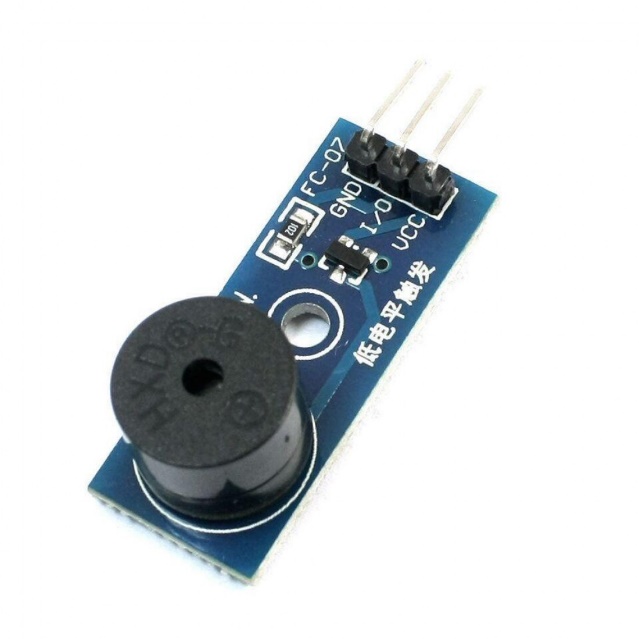
If a fire is detected, the Raspberry Pi will activate a buzzer to alert users to the danger and send a notification message to the user's devices.

In this methodology, API's will be used to connect the notification messaging system to the Raspberry Pi, which will ensure prompt delivery of notifications. This approach will enable the system to send notifications to the user's devices.

**4) Components**:

The components that we have used for this project are listed below:

**5) Code:**

The Source Code for our **Multi-Processing Fire Detection System using Raspberry Pi** is given below: The attachment contains Python Files executing the OS concepts.

import time

import RPi.GPIO as GPIO

from pushbullet import Pushbullet

from multiprocessing import Process, Value

flame\_sen = 18

buzzer = 29

fire\_detected = Value('i', 0)

GPIO.setmode(GPIO.BOARD)

GPIO.setwarnings(False)

GPIO.setup(flame\_sen, GPIO.IN)

GPIO.setup(buzzer, GPIO.OUT)

pb = Pushbullet("o.59hrJyB39l75SDxN4tnxWh9XoYtp1GoE")

print(pb.devices)

def detect\_flame(fire\_detected):

    while True:

        if GPIO.input(flame\_sen) == GPIO.LOW:

            print("Fire detected!")

            fire\_detected.value = 1

        else:

            fire\_detected.value = 0

        time.sleep(0.1)

def alert(pb):

    while True:

        if fire\_detected.value == 1:

            print("Sending push notification...")

            GPIO.output(buzzer, GPIO.LOW)

            dev = pb.get\_device('OPPO CPH1931')

            push = dev.push\_note("Alert!!", "Fire at your place")

        else:

            GPIO.output(buzzer, GPIO.HIGH)

        time.sleep(0.1)

if \_\_name\_\_ == '\_\_main\_\_':

    p1 = Process(target=detect\_flame, args=(fire\_detected,))

    p2 = Process(target=alert, args=(pb,))

    p1.start()

    p2.start()

    p1.join()

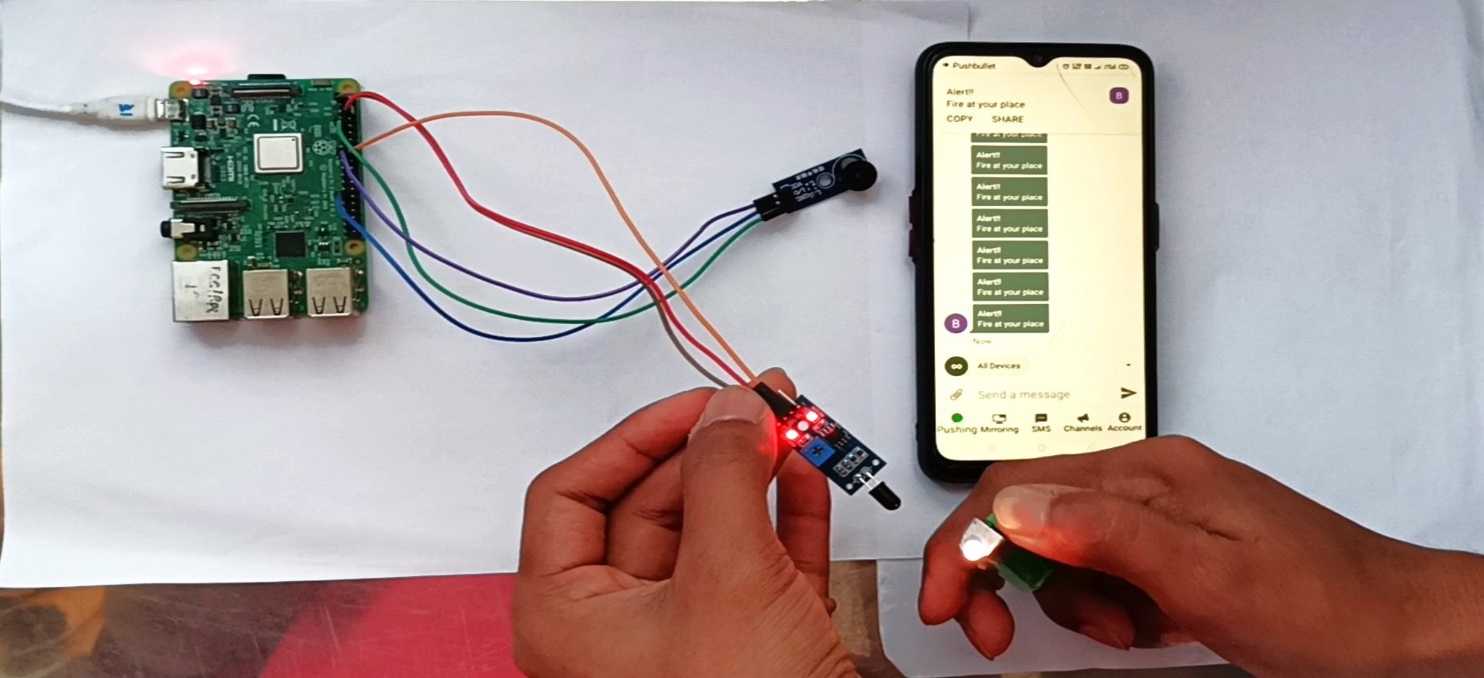
    p2.join()

**# Code Explanation:**

1. Importing Libraries:
   * The **time** library provides functions for time-related operations.
   * **RPi.GPIO** is a Python library used for accessing and controlling the Raspberry Pi's GPIO pins.
   * **Pushbullet** is a library used for sending push notifications to devices.
   * **multiprocessing** provides support for running multiple processes simultaneously.
2. GPIO Pin Setup:
   * Two variables, **flame\_sen** and **buzzer**, are defined to represent the GPIO pin numbers.
   * The **GPIO.setmode(GPIO.BOARD)** sets the pin numbering mode to use the physical pin numbers on the Raspberry Pi board.
   * **GPIO.setwarnings(False)** disables GPIO warnings.
   * **GPIO.setup(flame\_sen, GPIO.IN)** sets the flame sensor pin as an input pin.
   * **GPIO.setup(buzzer, GPIO.OUT)** sets the buzzer pin as an output pin.
3. Pushbullet Initialization:
   * The Pushbullet API key is provided to create a Pushbullet object called **pb**.
   * **pb.devices** retrieves a list of available devices associated with the Pushbullet account.
4. Flame Detection Function:
   * The **detect\_flame** function is defined to continuously monitor the flame sensor input.
   * It runs in an infinite loop.
   * If the flame sensor input is low (flame detected), it sets the **fire\_detected** value to 1.
   * Otherwise, it sets the **fire\_detected** value to 0.
   * It sleeps for 0.1 seconds before repeating the loop.
5. Alert Function:
   * The **alert** function continuously checks the **fire\_detected** value.
   * It runs in an infinite loop.
   * If the **fire\_detected** value is 1, indicating a fire, it sends a push notification using Pushbullet.
   * It also sets the buzzer pin to low (active state).
   * If the **fire\_detected** value is 0, it sets the buzzer pin to high (inactive state).
   * It sleeps for 0.1 seconds before repeating the loop.
6. Process Creation and Execution:
   * The **\_\_name\_\_ == '\_\_main\_\_'** condition ensures that the following code is only executed when the script is run directly, not when imported as a module.
   * Two processes, **p1** and **p2**, are created, representing the flame detection and alert functions respectively.
   * The **target** parameter specifies the function to be executed by each process, and the **args** parameter provides the arguments for the function.
   * Both processes are started using the **start()** method.
   * The **join()** method is called on both processes to wait for them to complete before exiting the program.

**6) Output**

The final output for our project is given below:

****

**7) Conclusion**:

In conclusion, the project successfully addresses the problem of fire detection using Raspberry Pi. By implementing multi-processing techniques, the system achieves faster and more efficient fire detection. The integration of fire sensors, buzzer, and notification messaging through APIs enables timely alerts and notifications, allowing for quick response to fire incidents. The project demonstrates the potential of using Raspberry Pi as a cost-effective and versatile platform for fire detection applications.

**THE-END**