AI BASED EARLY FIRE DETECTION USING RASPBERRY PI PICO

1. Introduction

One of the most common man made disaster is by fire. The project focuses on implementation in petrol pumps, and other fire zone and fire restricted areas.Today, we'll look at an unusual Pico project that provides a quick alarm response to the people in the particular zone when a fire source or flame is detected. The Advantage on real time applications is, it provides quick response before a smoke spread due to fire Whenever a fire/flame is detected it sends quick response to the peoples in the zone.

2. Components

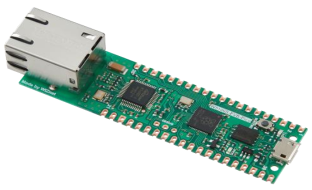
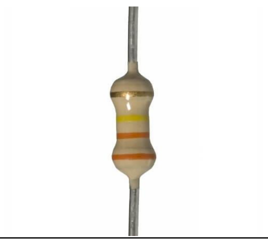
2.1 Hardware

- Jumper wires
- Wiznet Pico board
- LED Bulb
- 330k Resistor
- 16*2 LCD Display
- Buzzer
- I2C Module

2.2 Software

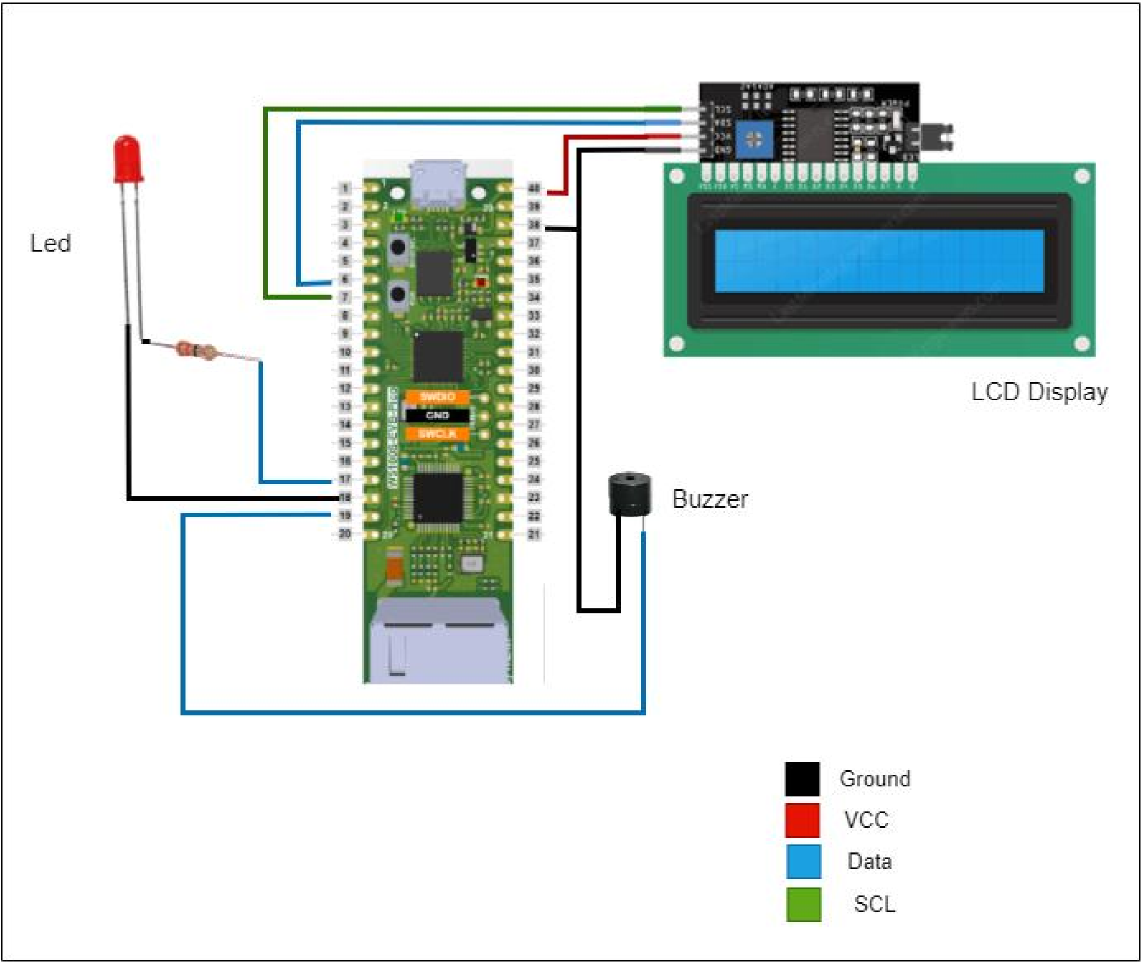
- Yolov5 Object detection
- Arduino IDE
- Google Colaboratory
- PyCharm - Python IDE

3. Component Description

Device	**Description**
Jumper Wire	The jumper wires are used to connect the components.
Wiznet Pico board 	W5100S-EVB-Pico is a microcontroller evaluation board based on the Raspberry Pi RP2040 and fully hardwired TCP/IP controller W5100S.
LED Bulb	The led is used to provide red light warning when flame or fire source is detected.
330k Resistor 	330K Ohm 0.25W High Quality Carbon Film Resistor (CFR) with ±5% Tolerance and Tin Plated Copper Leads.
16*2 LCD Display	To display the text when fire is detected.

Buzzer	To provide the alarm beep when fire is detected.
I2C Module	The I2C is a serial communication protocol, it has two terminals, one is of the clock and the other is for serial data communication.

4. Circuit diagram



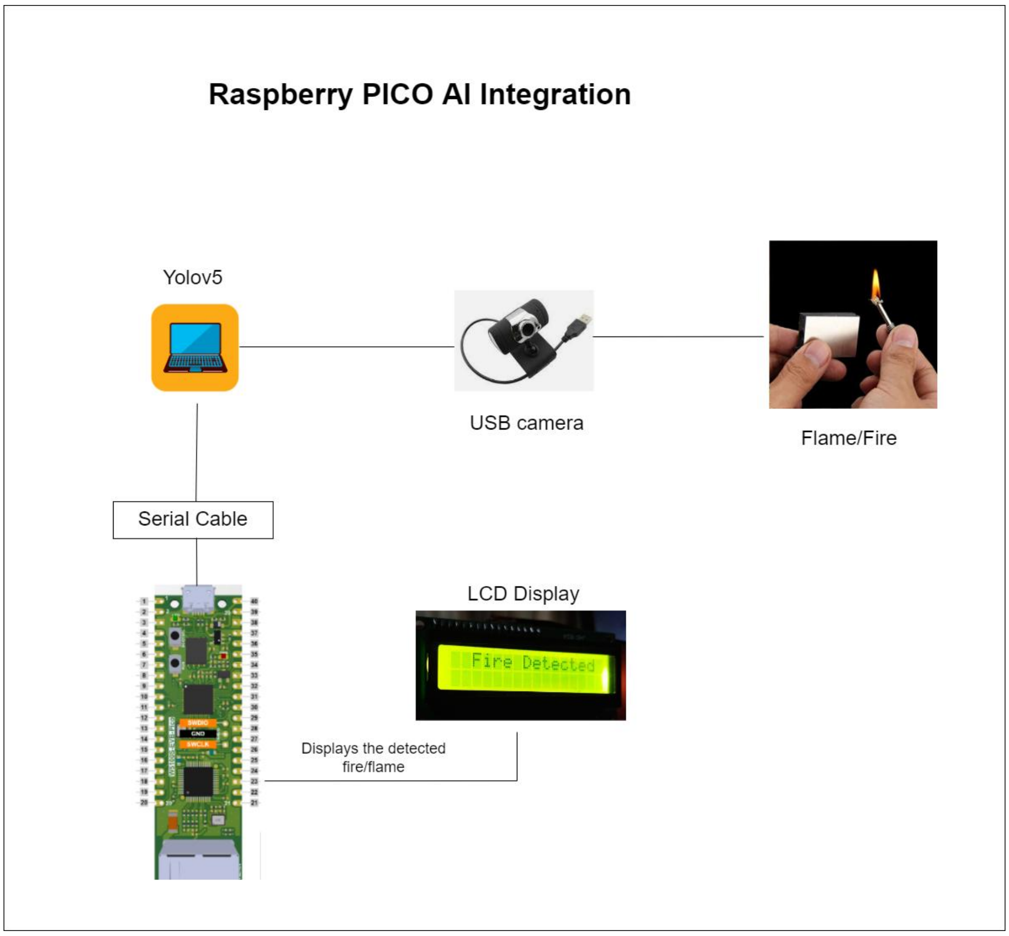
4.1. Pin Configuration

LED: Consist of 2 pins, positive pin which is connected to 330K resistor and GP13. and negative pin is connected to GND (ground).

BUZZER: Consist of 2 pins in which positive pin is connected to GP14. Negative pin is connected to GND.

16*2 LCD Display with I2C Module: Consist of 4 pins. VCC, GND, SDA, SCL. VCC connected to 5V. SDA pin connected to GP4. SCL pin connected to GP5. GND can be connected to any GND of pico board.

5. AI diagram



5.1 AI implementation

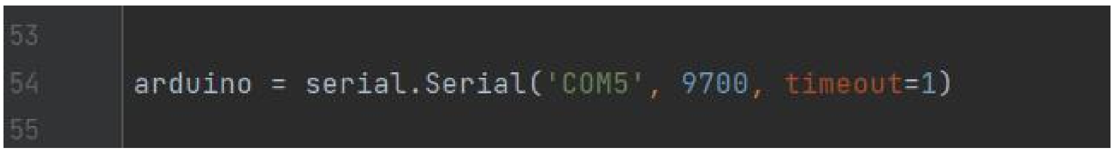
In this project we have used the YOLOV5 object detection model on a custom trained datasets which could detect when a flame or fire source is found. Training is done by various fire sources such as lamps,lighters and matchsticks. The annotation is done with the collected real time flames and fire sources. Then feed it to the training algorithm. Ensure that the present working directory is the YOLOV5 directory and all dependencies are installed. The coco128.yaml file has to be updated with the datasets directory and also the classes used. Train using the command:

```
!python train.py --img 416 --batch 16 --epochs 100 --data coco128.yaml --weights yolov5s.pt --cache
```

A pertained weight has been assigned to the training [yolov5s.pt] and is running at 100 epochs. Once the training is done check in the "runs" folder to check the accuracy of predictions in the .jpg files. The real time detection can be used with the command :

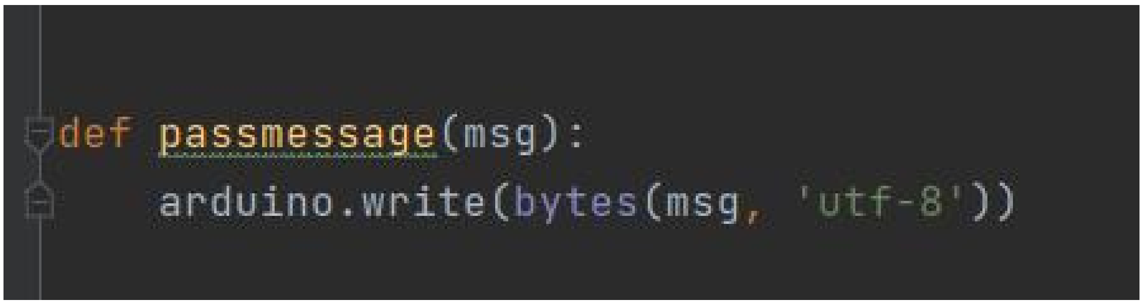
python test.py --weights best.pt --img 416 --conf 0.28 --source 1

While running this command the present working directory should be the root of the yolov5 folder. The weight file ending in ".pt" file has to be given in the - -weight argument. The source can be set to 1 or 0 depending on the webcam used as internal or external. On running this command a frame opens up with bounding box detection. Through serial communication it sends the data and displays the status in LCD display ,These status messages include "Fire detected",when a flame or fire source is found using camera, in case no fire/flame are detected "No fire detected" command. On receiving the status via serial communication to raspberry pico ,we check for keywords matching. A global variable is assigned for each status and this status will be displayed in the LCD display as well as the Serial Monitor.



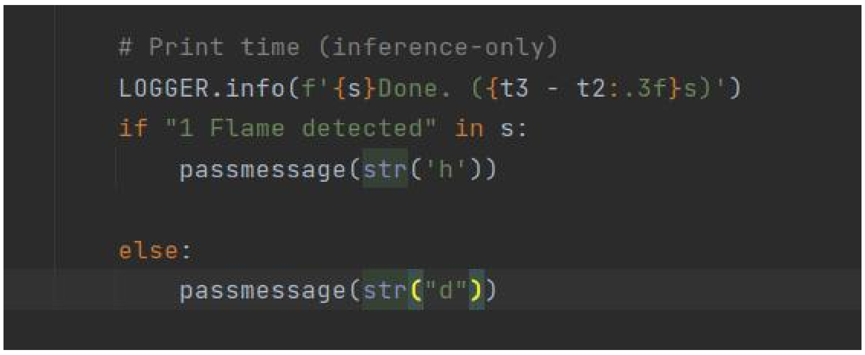
```
arduino = serial.Serial('COM5', 9700, timeout=1)
```

Port configuration for the serial communication.



```
def passmessage(msg):
    arduino.write(bytes(msg, 'utf-8'))
```

Function for the obtained keyword which is writing to Arduino IDE.



```
# Print time (inference-only)
LOGGER.info(f'{s}Done. ({t3 - t2:.3f}s)')
if "1 Flame detected" in s:
    passmessage(str('h'))

else:
    passmessage(str("d"))
```

Keyword conditions for the fire detection.

6. Code

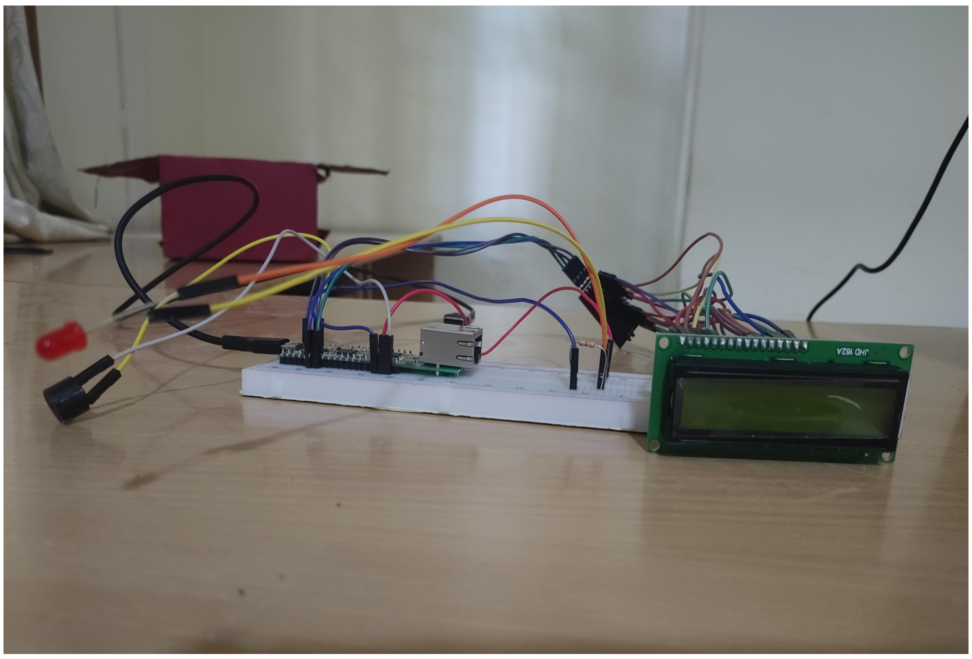
6.1 Arduino code

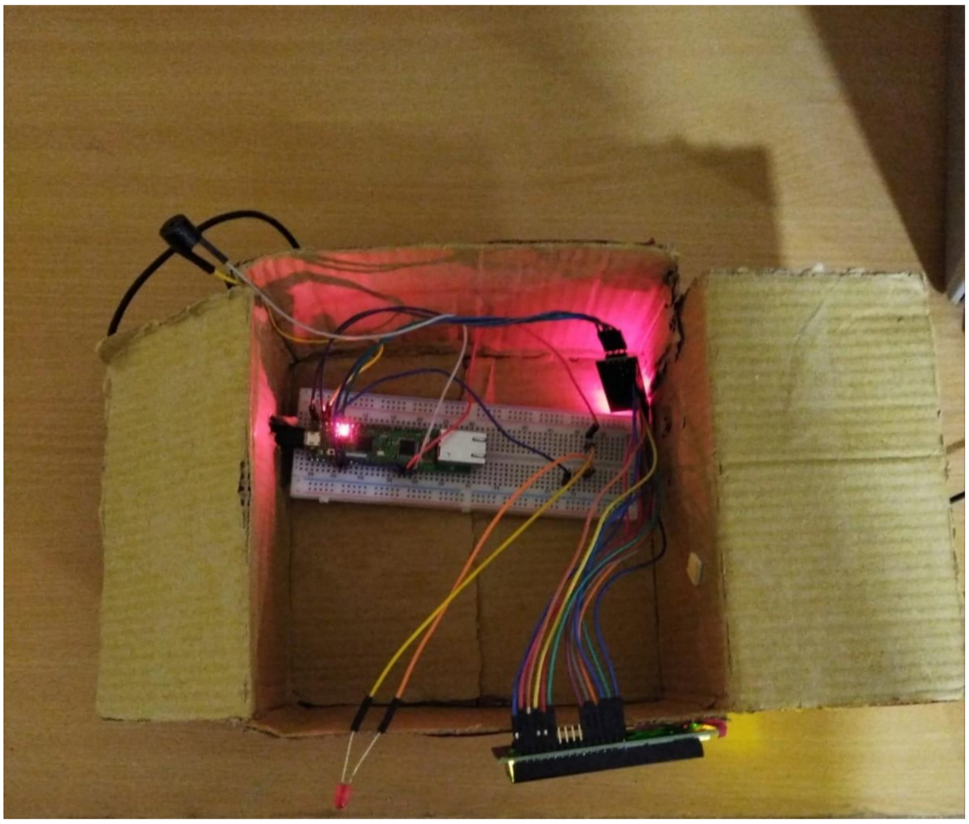
```
//libraries
#include <Wire.h>
#include <LiquidCrystal_I2C.h> //For LCD crystal display

LiquidCrystal_I2C lcd(0x27, 16, 2);

// Initializing Variables
int Le = 13; // led data pin
const int buzzer=14; // buzzer pin
char data;

void setup()
{
    Serial.begin(9700);
    pinMode(Le,OUTPUT);
    pinMode(buzzer, OUTPUT);
    // Setting the LED and buzzer Off when program is executed.
    digitalWrite(Le,LOW);
    digitalWrite(buzzer,LOW);
}

void loop()
{
  //Serial connection
  while(Serial.available())
  {
    data = Serial.read(); //Serial read
  }

  //Receiving data from camera through serial communication.
  if(data == 'h') //if recieved character is f then turning on the led and buzzer for early fire detection.
  {
    digitalWrite(Le,HIGH);        // turn the LED on (HIGH is the voltage level)
    digitalWrite(buzzer, HIGH);   // turn the LED on (HIGH is the voltage level)
    delay(1000);
    Serial.println("LED turned on");
    lcd.begin();

    lcd.backlight();
    lcd.clear();
    lcd.setCursor(2,0);
    lcd.print("Fire Detected "); // To print in LCD display

  }

46   else           //if recieved character is not f then turning off the led and 
buzzer.
47   {
48     digitalWrite(Le,LOW);         // turn the LED off (LOW is the voltage level)
49     digitalWrite(buzzer, LOW);     // turn the LED off by making the voltage LOW
50     delay(1000);
51     Serial.println("LED turned off");
52     lcd.begin();
53 
54     lcd.backlight();
55     lcd.clear();
56     lcd.setCursor(4,0);
57     lcd.print("No Fire");        // To print in LCD display
58 
59   }
60 }
```

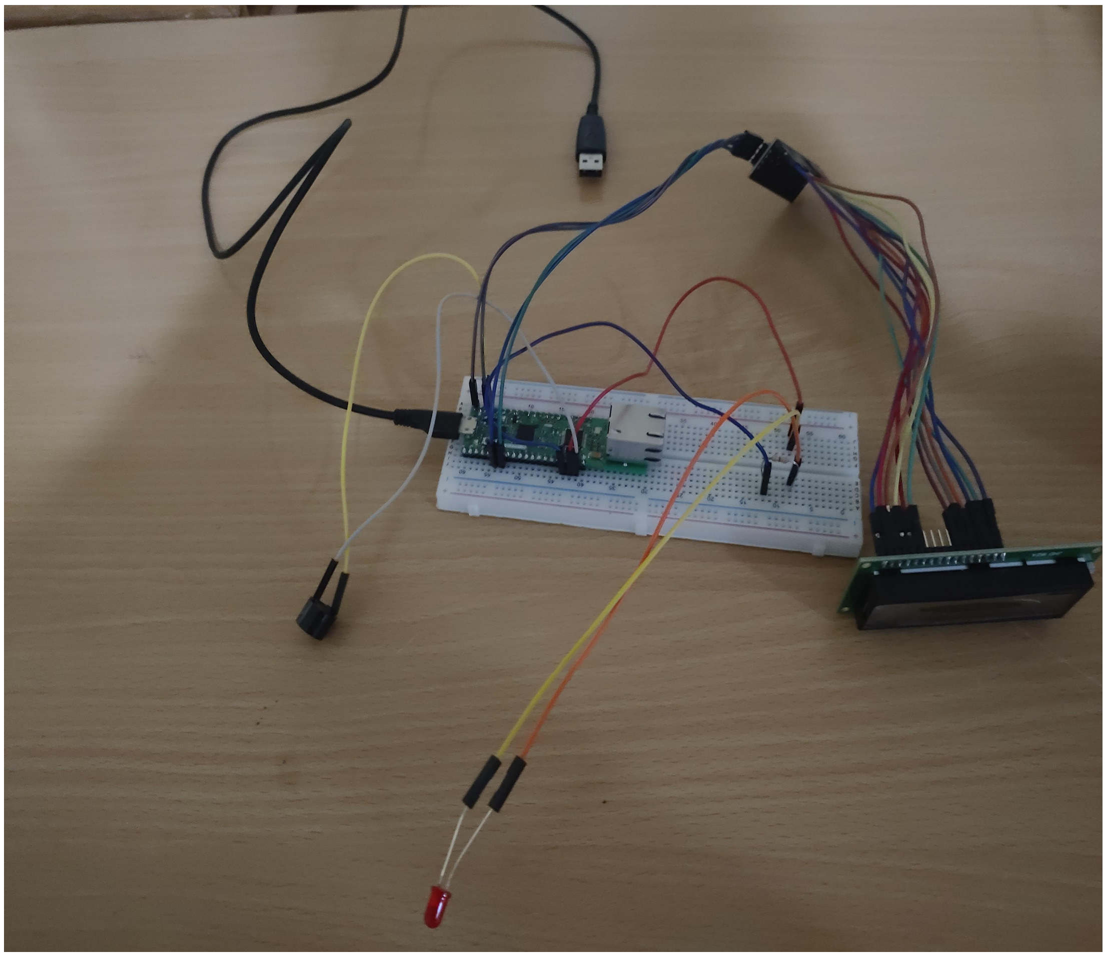
Code Explanation:

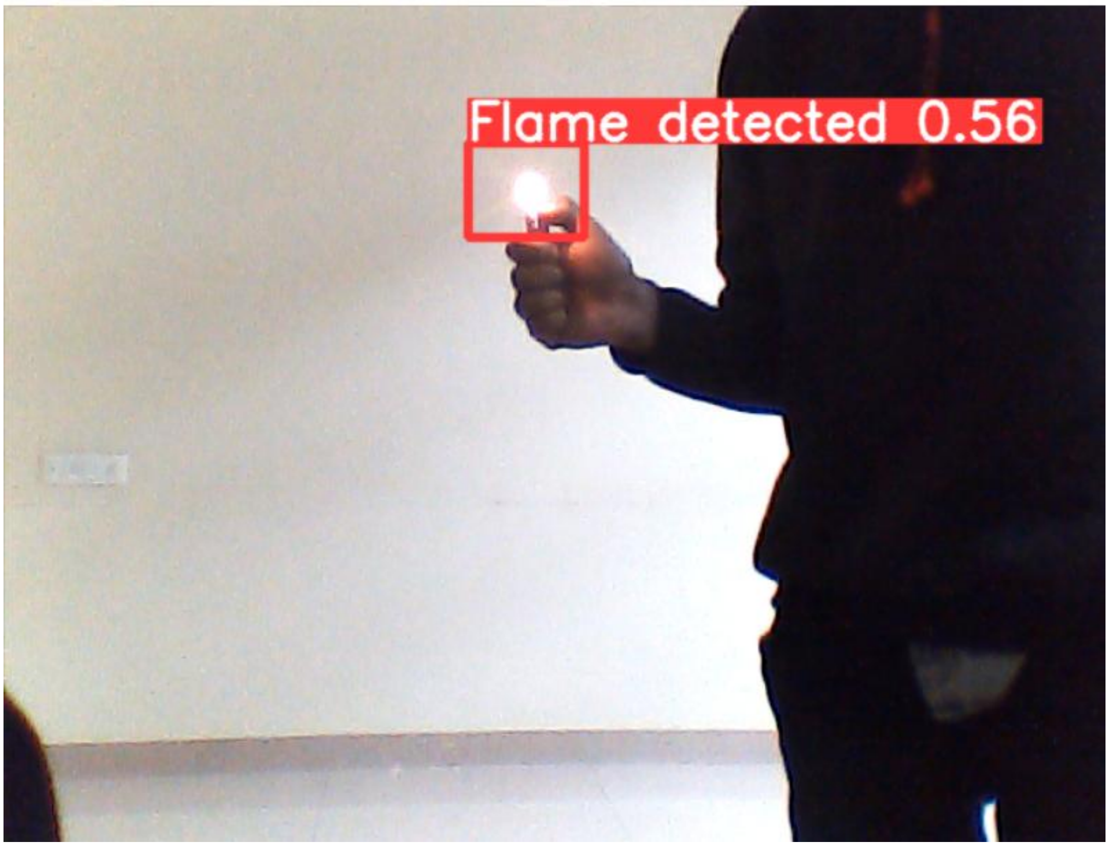
Liquidcrystal_I2C library has to be added to the arduino sketch. Line [2 - 3] contains the libraries required for LCD display which is wire.h and Liquidcrystal_I2C.h. Line [7 - 10] Initializing variables for led , buzzer and a global variable data. Line [12 - 20] void setup contains pinMode of led and buzzer. Setting led and buzzer voltage level to low. Line [22] void loop starts. Line [25 - 28] while loop to read the data through serial connection. Line [31 - 44] if condition with data = h , where h is a keyword passed through serial communication. If the data read is “h” then the buzzer and led gets activated to indicate the fire detected. In lcd display “Fire detected ” message been printed. Line [46 - 59] else condition when there is “No fire” detected. when no fire is detected led and buzzer voltage level will be low.

7. Images

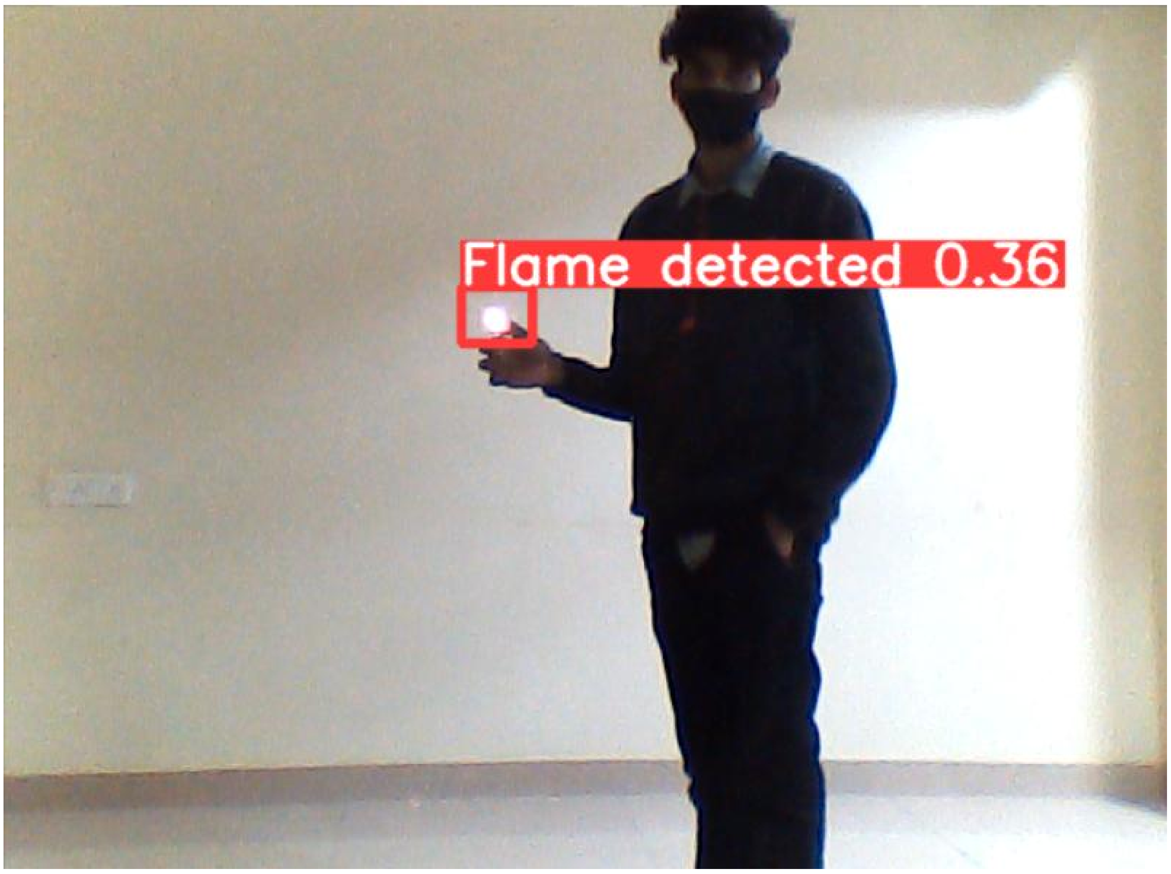




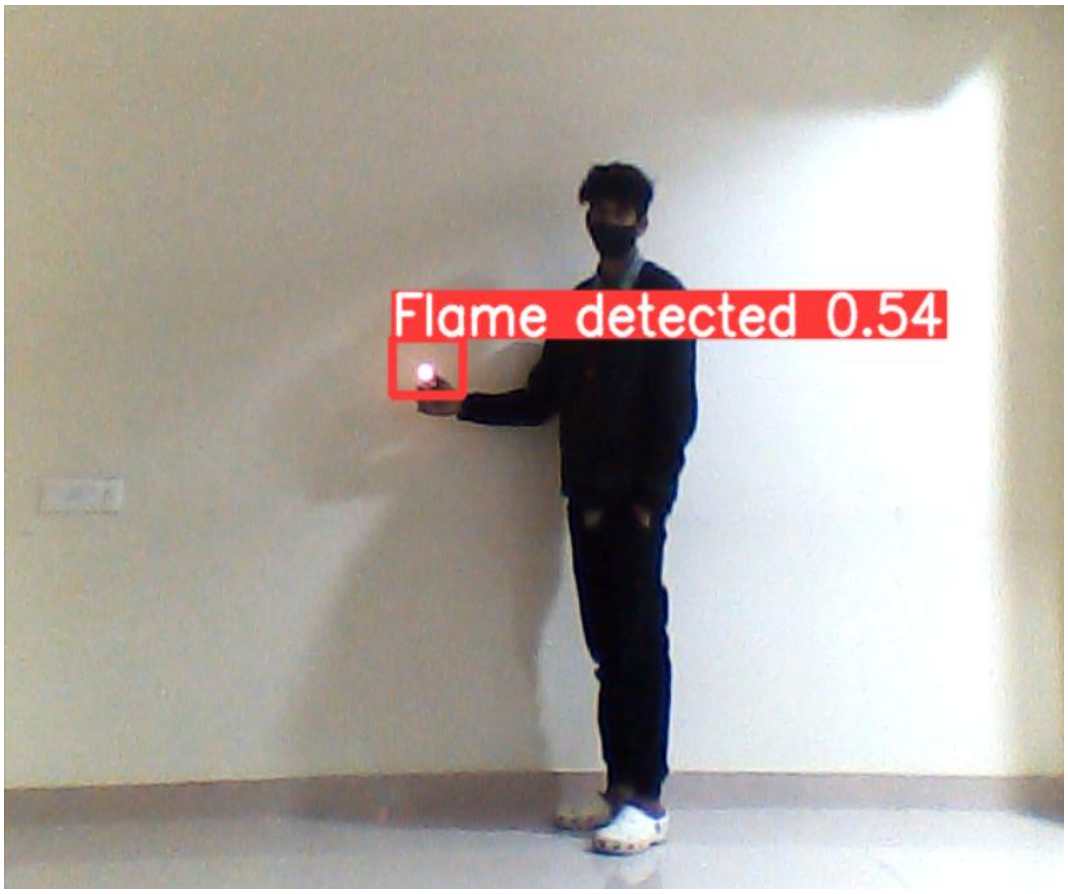




Short Range



Mid Range



Long Range

LCD Display of fire detection

LCD Display for No fire detection

8. References

[1]https://towardsdatascience.com/early-fire-detection-system-using-deep-learning-and-opencv-6cb60260d54a

[2 https://create.arduino.cc/projecthub/shubhamsuresh/image-processing-based-fire-detection-extinguisher-system-57ddc3

[3]https://www.c-sharpcorner.com/article/how-to-create-a-fire-detection-alarming-system-using-arduino-uno-r3/

[4]https://www.researchgate.net/publication/356171734_Design_of_a_Home_Fire_Detection_System_Using_Arduino_and_SMS_Gateway

[5]https://create.arduino.cc/projecthub/shubhamsuresh/image-processing-based-fire-detection-extinguisher-system-57ddc3