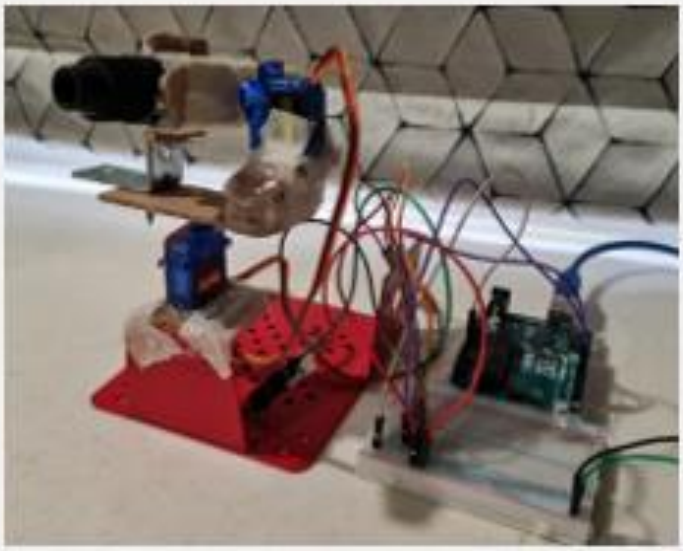
INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Autonomous Fire Extinguisher using OpenCV and Arduino



TEAM – 5

Section 13

Composition:

Vedant Agrawal - (20AG10042)

Ratan Raj Ojha - (20CE30021)

Monish Natarajan - (20CS30033)

Pranay Pandey - (20PH20026)

Acknowledgement

We learned a lot during our project, "Autonomous fire extinguisher using OpenCV and Arduino". Let it be Image Processing, Arduino programming, or the theoretical part of the project. It was our first time being creative with electronics. We faced many difficulties in the journey, like slow image processing algorithm, defective components, improper mapping of output data resulting in inaccurate servo movements. However, with the constant guidance & support of our professors Mr. Adway Mitra, Mrs. Priyanka, Mr. Ayan Roy, we were able to complete it successfully. This project is of great value in the future, and our efforts to build this were worth it.

Once again, thanks to DIY faculty and TAs to inculcate practical and necessary skills to work upon. Also, huge thanks to the IIT Kharagpur administration for introducing this course for the 2020 fresher batch.

Thanking you,

Members of Team 5.

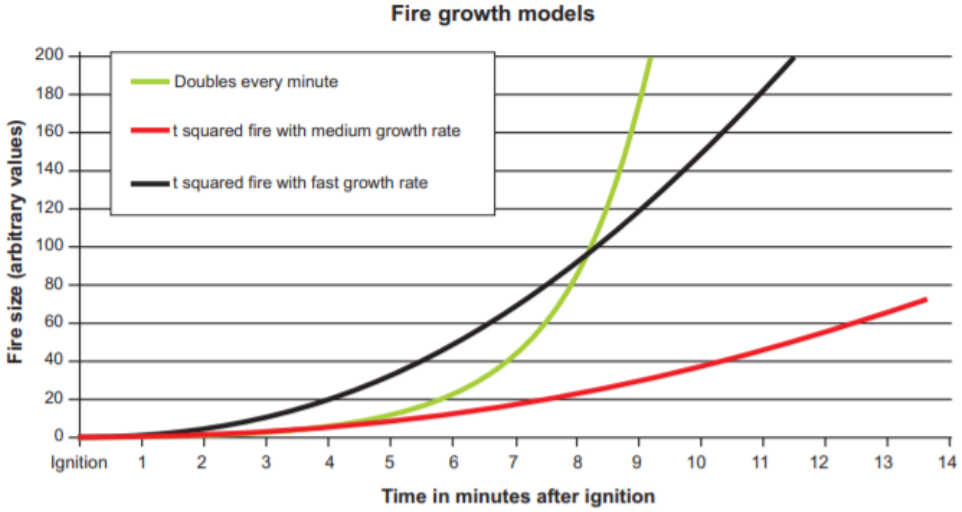
Contents

Introduction:

Quick and effective responses are extremely crucial in fire-fighting operations. The delay of a few minutes could cause the flames to grow rapidly and eventually mean the death of many. The fire-fighters too put their lives at risk in the course of such operations. From 2014 to 2018 the US reported an annual death of about 65 fire-fighters and many more injured while on duty. Automation of the rescue operations can play a crucial role in rendering faster response times and reducing risk on human lives.

The objective of this project is to develop a working model of an automated fire-extinguishing system that locates fire and automatically extinguishes it by releasing high pressure water.

With the advent of self-driving cars, we envision a future wherein a similar autonomous fire-extinguishing system is mounted on an autonomous emergency response vehicle making rescue operations much quicker and safer.

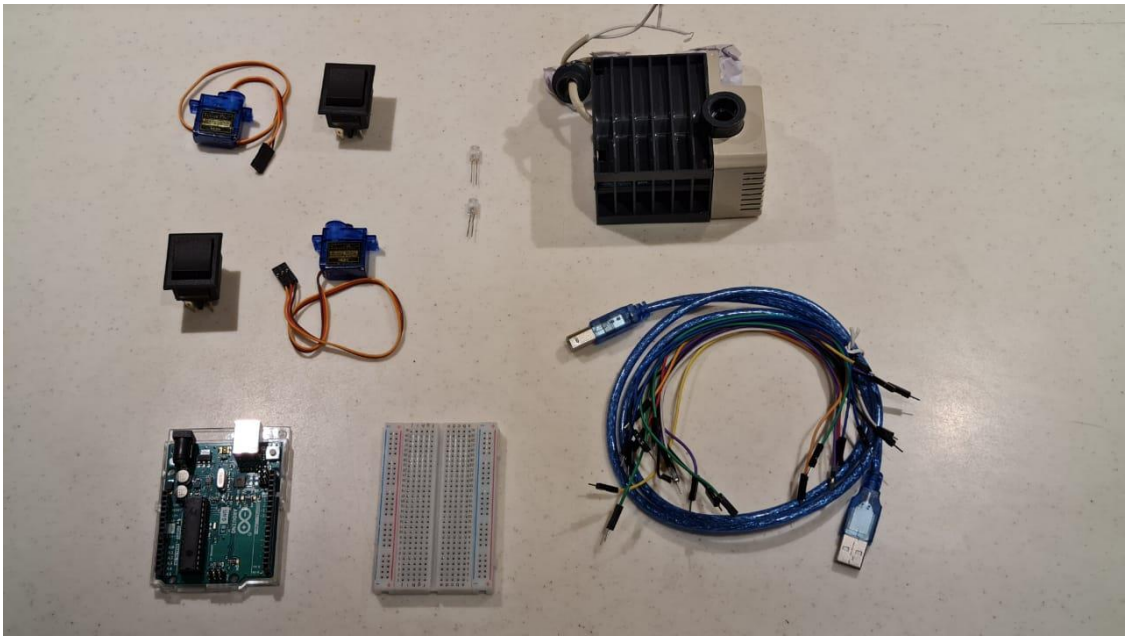


(Link : https://www.epsu.org/sites/default/files/article/files/6367-Its-about-time-LOW-RES2.pdf)

Challenges involved in this project:

- Flame Detection and return Location of Flame
- Arduino Motor Controls to adjust trajectory
- Control of Water Pump until fire is extinguished

Hardware Requirements:



1. **ARDUINO UNO R3 × 1**

   Cost: 450 Rs.

   Functionality: This is the microcontroller which will help to control the servos and the relay module for Water Pump.

2. **Breadboard × 1**

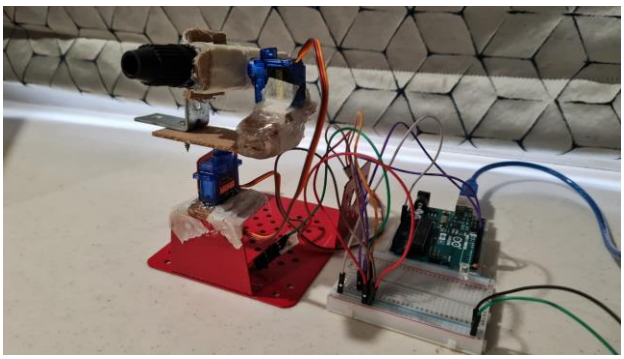
   Cost: Rs. 50

   Functionality: This is used to manage as well as make easy and clean connections for our project.

3. **Micro Servos (SG90) × 2**

   Cost: 200 × 2 = Rs. 400

   Functionality: These are used to actuate rotational mechanism of the water nozzle. 600 mA micro servos were sufficient for our project due to low torque requirements.



4. Water Pump (220 V AC) × 1

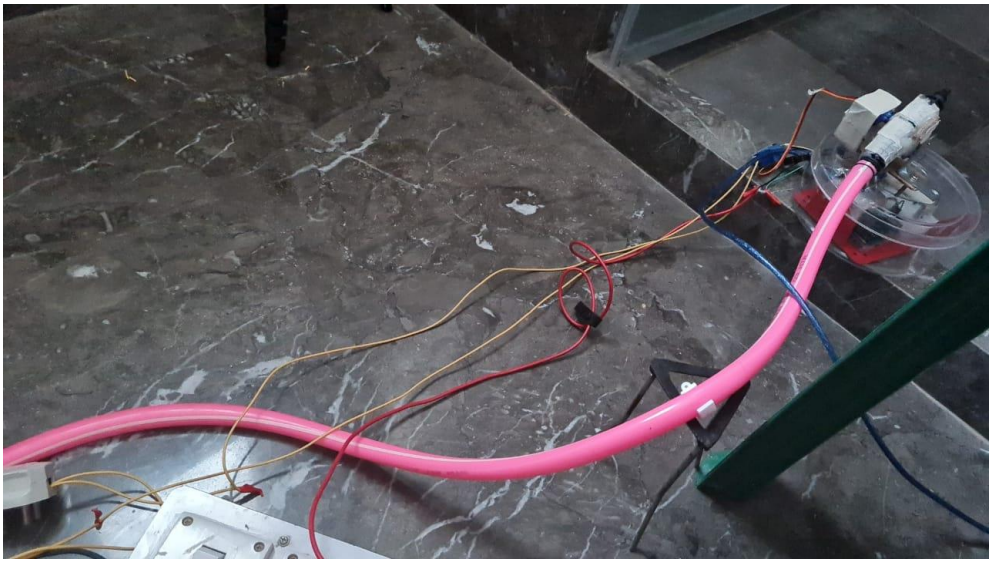
Cost: Rs. 160

Functionality: This is used to provide water to the nozzle at a high velocity in order to extinguish the fire at some height. Our project requirement was to obtain a range of the water stream to be about 2 - 3 m. Therefore, we chose this pump.

5. Water Nozzle and Hose

Cost: Rs. 50

Functionality: This is used to shoot a streamlined beam of water to the fire so that the projectile motion can be implemented.



6. Camera Module

Cost: Rs. 215

Functionality: This is used to capture the image of the flame so that it can be processed.

7. Relay Module

Cost: Rs. 100

Functionality: This is used to control the operation of a high power appliance with a LOW 5V signal from Arduino, in our project it is used to control a water pump.

8. Central Processing Unit

Cost: Rs. 1000

Functionality: It is used to process the image captured by the camera module. It determines the coordinates and condition of the wire and transmits the respective data to the Arduino.

9. 5V 1A DC Power Source

Cost: Rs.100

Functionality: It is used to power the 2 micro servos which require around 1 A of current for full functionality.

10. 220V AC Power Source

Cost: Rs. 6 / kWhr

Functionality: It is used to power the water pump.

11. Plastic Case for Water-Proofing

Cost: Rs. 10

Functionality: It safely waterproofs the hardware components from water splashes.

12. Jumper and Connecting Wires

Cost: Rs. 50

Functionality: It is used in the connections of various power sources to the components, from CPU to Arduino, from Arduino to different actuators etc.

13. Craft - Sticks

Cost: Rs. 10

Functionality: It is used in making the servo rotating mechanism.

14. Hot Glue

Cost: Rs. 5

Functionality: It is used in making the servo rotating mechanism.

15. Masking Tape

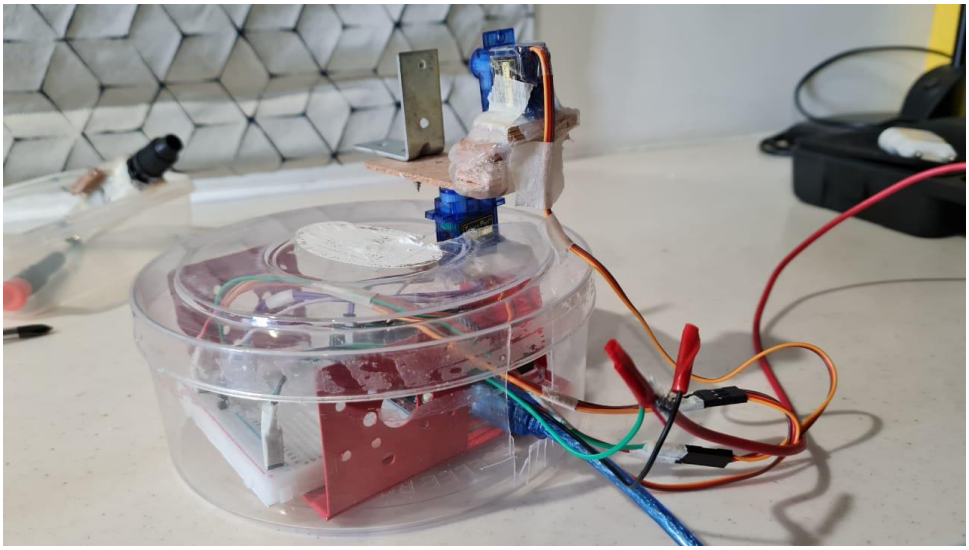
Cost: Rs. 15

Functionality: It is used in making the servo rotating mechanism.

16. L-Bracket :

Cost: Rs. 5

Functionality: It is used to hold the Water Nozzle for free rotation.



Total Cost: Rs. 2700/-

Flame Detection using OpenCV library in Python:

About OpenCV library in Python:

OpenCV is a Python open-source library, which is used for computer vision in Artificial intelligence, Machine Learning, face recognition, etc. The purpose of computer vision is to understand the content of the images. It extracts the description from the pictures, which may be an object, a text description, a three-dimensional model, and so on.

In this project, we use OpenCV to detect the location of a candle flame using a built-in laptop web-camera. The flame is placed at a distance of 1.5 metres away from the camera.

Approach for Flame Detection:

A brightness or intensity-based segmentation approach is used to detect the location of the candle flame. For any grey scale image, pixel intensity values range from 0-255 wherein 0 represents perfect black and 255 represents white. A source of light such as a candle produces a region or a cluster of very bright pixels.

With x set as the threshold value the following pseudocode was implemented.

```
if pixel_intensity >= x : set pixel as white

if pixel_intensity < x  : set pixel as black
```

As a result a binary image is produced. The mean of coordinates of the white pixels is obtained and returned as the centre of the region of interest.

This centre is used to adjust the servo motors and direct the trajectory of the water to extinguish the flame.

Fig. (a) Binary Image of Candle Flame (b) Final Region of Interest

Servo-Motor Setup for Alignment of Water Trajectory

The flame could be located anywhere on the 2-D plane of the screen located 1.5 metres away from the water-pump module. Hence to correctly set the trajectory of water, rotation of the water nozzle about two different axes was necessary.

This challenge was overcome by using two servo-motors aligned perpendicular to each other as shown in the diagram. One motor controls the orientation of the base plate on which the nozzle rests and the second motor controls the angle of the nozzle with respect to the base plate.

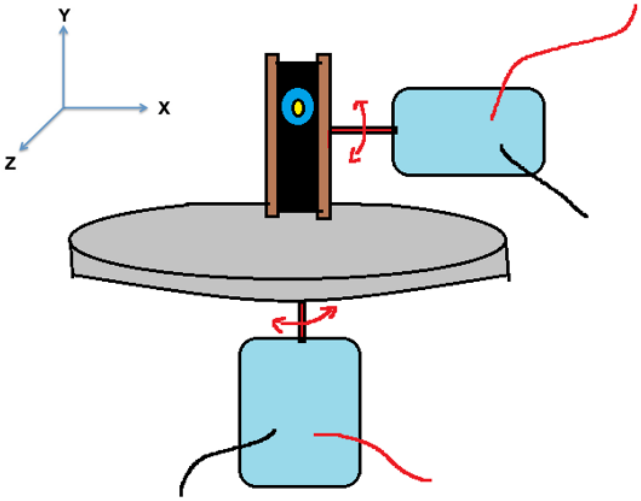


Fig. Perpendicular motors setup for alignment of nozzle

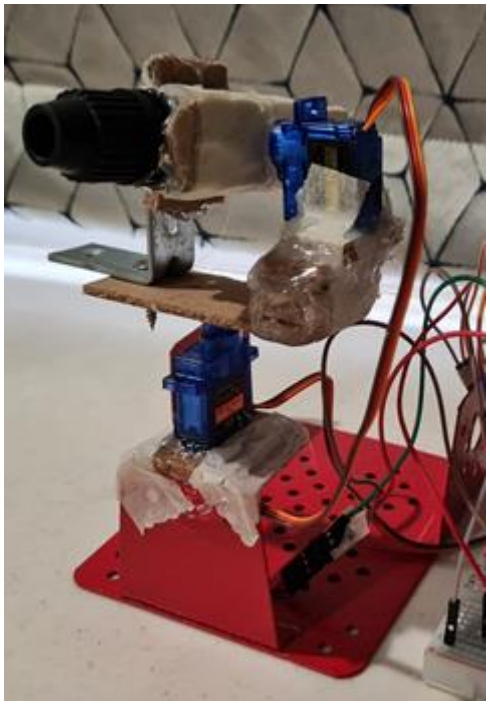
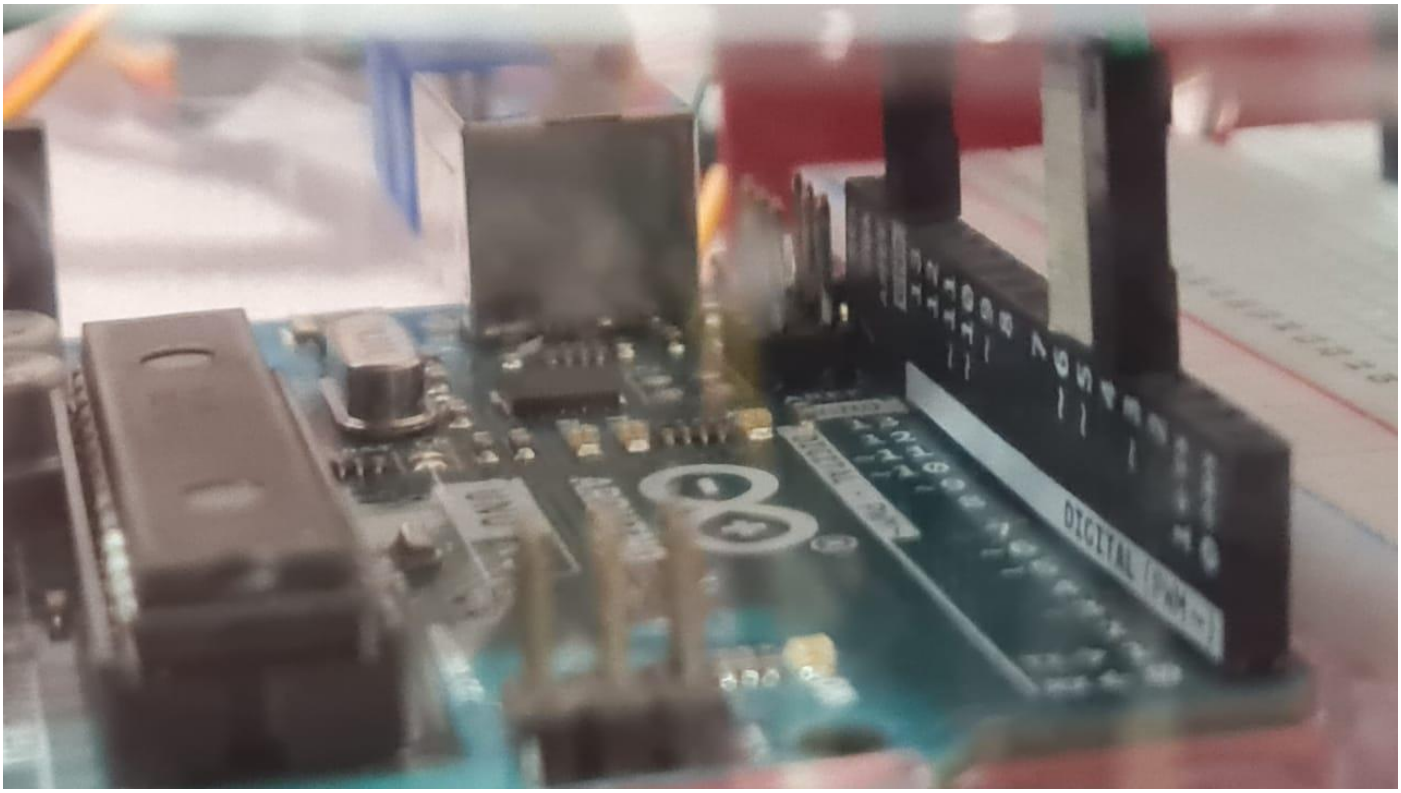


Fig. Nozzle mounted to the motor mechanism

Programming in Arduino and Integration with Python

We used PySerial, a Python API module to communicate with the Arduino UNO board. Python sends a string of format “X<z_rotation>Y<plane_theta>Z<status of flame>”. Using string operations .remove() we extracted each angle and stored them in the variables of the same name.

We first adjust the servo motors to correct position, by passing the extracted angles. Then if the status of flame is 1 the power is supplied to the water pump for 6 seconds and then the pump is turned off. Again angles and status are overwritten by new values from the image and this process continues.



Projectile Calculations of Water Stream and Distance Coordinates Mapping

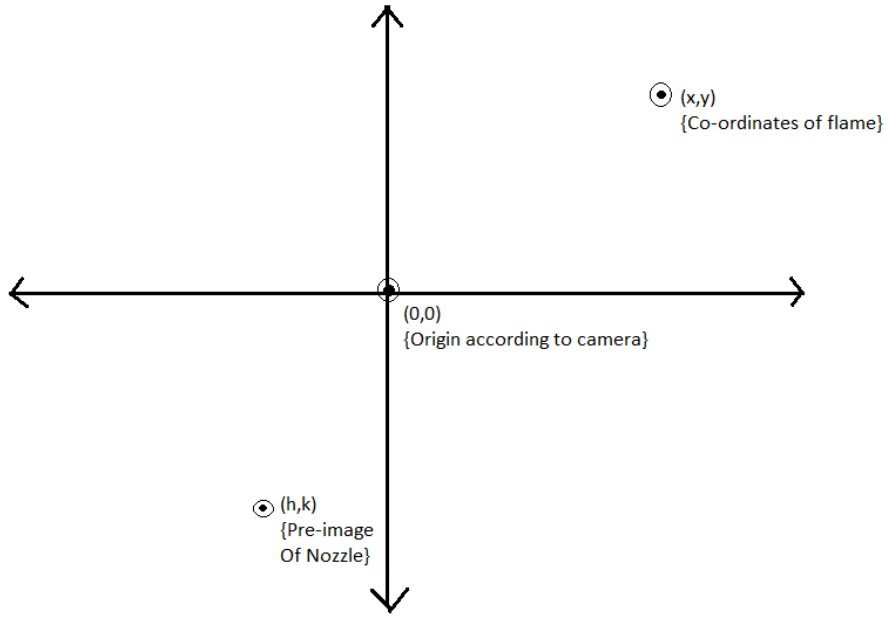


Fig: Shifting of origin.

As we can see, the coordinates of origin given by the camera are at the center of the screen, but the position of preimage of the nozzle is not same as the origin given by the camera so we are shifting our origin at the preimage of the nozzle.

So, after shifting our origin the new coordinates of the flame will be (x+h, y+k).

By taking distance of the plane of the candle flame 'D' we will calculate the angles θ and ϕ accordingly where: θ- vertical angle of rotation

ϕ- horizontal angle of rotation.

Calculations:

$\tan \phi = (x+h)/D$

$\phi = \arctan((x+h)/D)$

Now, θ will be calculated by the equation of trajectory of the projectile.

Taking:

$L = (D^2 + (x+h)^2)^{½}$

$y + k = Y$

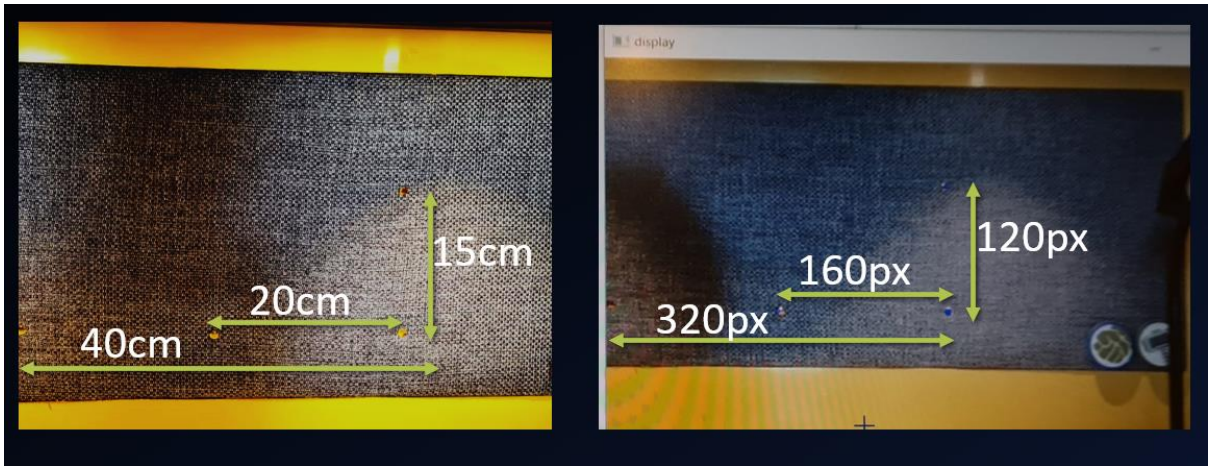
$Y = L(\tan\theta) - gL^2(1+\tan^2\theta)/2v^2$ [v= Speed of the water stream]

Now, this quadratic equation will be giving two values of θ. We will be taking the smaller value of θ to make our system more efficient so as to reduce the time to reach the fire.

$$\theta = \arctan\left(\frac{\left(v^2 - \sqrt{v^4 - g\left(2v^2 Y + gL^2\right)}\right)}{gL} \right)$$

As we know 'v' is the speed of the water stream which will be constant for our setup, it was calculated experimentally using equation of trajectory for different datasets and computing the averaged value.

Experiment For cm/Pixel Ratio:



Therefore this confirms the linearity of all the directions, for a fixed cm/pixel ratio = D/640 where D is the distance of the screen from the camera in cm. This data is for D= 80 cm.

Demonstration:

Check the following YouTube link:

Appendix:

- Python Code:

```python
import numpy as np
import serial
import time
import cv2
from math import *

## All params are in m.
## Distance of screen from camera = D
D=1.46

## Height of the camera = H
H= 0.70

## Height of the nozzle = h
h=0.21

## Distance of the nozzle from screen
d=0.79

## m/pixel ratio= D/640 for a distance D of the screen from camera
fact= D/640

##velocity = 4m/s
u=4

## gravity
g = 9.81

arduino = serial.Serial(port='COM7', baudrate=9600, timeout=0.1)

def find_centre2(white_loc):
    n = len(white_loc)
    if n<120:
        print("No Flame Detected")
        cv2.imshow('display', img)
        return 0
    arr = np.array(white_loc)
    x,y = np.sum(arr,axis=0)/n
    return (round(y),round(x))
```

```python
def operate(img):
    img_in = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    img_out = np.zeros(img_in.shape)
    img_out[img_in>230]=255
    return img_out

def find_white_loc(img_in):
    white_loc=[]
    m,n = img_in.shape
    for i in range(m):
        for j in range(n):
            if img_in[i,j]==255:
                white_loc.append([i,j])
    return white_loc

def plane_theta(centre):
        ## coordinates with respect to camera
        x = centre[0]*fact
        y = centre[1]*fact

        ## coordinates with respect to nozzle. Obtained by shifting of origin
        X = sqrt(d**2+(x)**2)
        Y = y + H - h
        print("X:",X, "Y: ", Y)

        Q1 = ( atan( ( (u**2) + sqrt( u**4 - 2*Y*g*u**2 - (g**2)*(X**2) ) ) / (g*X) ) ) * 180/3.1415
        Q2 = ( atan( ( (u**2) - sqrt( u**4 - 2*Y*g*u**2 - (g**2)*(X**2) ) ) / (g*X) ) ) * 180/3.1415

        if (abs(Q1)<abs(Q2)):
            angle = Q1
        else:
            angle = Q2
        print("Vertical Angle is : ",angle)
        return angle
```

```python
def z_rotation(centre):

    ## coordinates with respect to camera
        x = centre[0]*fact
        y = centre[1]*fact


        ## coordinates with respect to nozzle. Obtained by shifting of origin
        X = x
        Y = y + H - h

        angle=(atan(X/d))*180/3.1415

        print("Horizontal Angle is :", angle)
        return angle



cap = cv2.VideoCapture(0)
if not cap.isOpened():
    print("Cannot open camera")
    exit()
while True:
    ret, frame = cap.read()
    frame = cv2.flip(frame,1)
    if not ret:
        print("Can't receive frame . Exiting ...")
        break
    img = np.copy(frame)
    img_mat = operate(frame)
    white_loc = find_white_loc(img_mat)
    centre = find_centre2(white_loc)


    if centre!=0:
        adj_centre=(centre[0]-320,-centre[1]+240)
        if centre!= None:
            print(adj_centre)
            img = cv2.circle(img,centre,40,(255,0,0),3)
            img = cv2.circle(img,centre,0,(255,0,0),5)
            img = cv2.circle(img,(320,240),0,(255,0,0),5)

            img = cv2.circle(img,(160,240),0,(255,0,0),3)
            img = cv2.circle(img,(320,120),0,(255,0,0),3)
```

```
        cv2.imshow('display', img)
        arduino_out="X"+str(z_rotation(adj_centre))+"Y"+str(plane_theta(adj_centre))+"Z"+str(1)
        arduino.write(bytes(arduino_out, 'utf-8'))
        arduino.flush()
        time.sleep(0.5)
        print(arduino_out)

    else:
            arduino_out="X"+str(360)+"Y"+str(360)+"Z"+str(0)
            arduino.write(bytes(arduino_out, 'utf-8'))
            arduino.flush()
            time.sleep(0.5)
            print(arduino_out)
    if cv2.waitKey(1) == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

- Arduino code:
-

motor_mech

```
#include <Servo.h>
#include <Math.h>

int servoPin_y = 6;  //for rotation in vertical plane
int servoPin_x = 5;  //for rotation in horizontal plane
int pumpPin = 4;
Servo Servo_y;
Servo Servo_x;
void setup() {
  Serial.begin(9600);
  Serial.setTimeout(100);
  // put your setup code here, to run once:
   Servo_y.attach(servoPin_y);
   Servo_x.attach(servoPin_x);

   pinMode(pumpPin, OUTPUT);
}

void loop() {
  int counter = 0;
  digitalWrite(pumpPin, HIGH);

  float plane_theta=360,z_rotation=360;
  String serialData=Serial.readString();
  plane_theta=extractY(serialData);
  z_rotation=extractX(serialData);
  counter = (int)extractZ(serialData);
  //Serial.print("Got it");
  //Serial.print(plane_theta,z_rotation);
  delay(100);

  if(plane_theta < 180 && z_rotation < 180)
  {
    Servo_y.write(round(95+plane_theta));
    delay(500);
    Servo_x.write(round(100+z_rotation));
    delay(2000);

    if(counter!= 0)
    {
      digitalWrite(pumpPin, HIGH);
      delay(4000);
      digitalWrite(pumpPin, LOW);
      }
    }
}

//"X<z_rotation>Y<plane_theta>Z<flame_status>"

int extractX(String data){
  data.remove(data.indexOf("Y"));
  data.remove(data.indexOf("X"),1);
  return data.toFloat();
  }
int extractY(String data){
  data.remove(data.indexOf("Z"));
  data.remove(0,data.indexOf("Y")+1);
  return data.toFloat();
  }
int extractZ(String data){
data.remove(0,data.indexOf("Z")+1);
return data.toFloat();
```

References:

For Projectile Motion:

https://aapt.scitation.org/doi/full/10.1119/1.3639153

For Arduino:

https://www.arduino.cc/reference/en/

For OpenCV and Python:

OpenCV documentation: https://docs.opencv.org/4.5.2/d6/d00/tutorial_py_root.html

Colour based segmentation: https://realpython.com/python-opencv-color-spaces/

An Ending Note:

The report ends here, but the implementation of this project is the future itself. Imagine 15 years in the future, when we would be accustomed to autonomous vehicles. If we integrate this Image processing-based extinguishing mechanism with AI driving algorithms being developed at Google or Tesla, this will make fire services independent of human inputs, thus minimizing delays or unavailability of services. Thanks for reading.