**Department of Electrical and Computer Engineering**
**North South University**

**Senior Design Project – CSE499**

# A Machine Learning and Embedded Approach to Detect Fire, Identify the Cause of it and Extinguish Using the Best Possible Extinguisher

| | |
|---|---|
| **Md. Rashiqur Rahman** | **ID# 1812366042** |
| **Sabbir Ahmed** | **ID# 1813515042** |
| **Md. Shafin Islam Rudro** | **ID# 1821054642** |
| **Sadia Aktar** | **ID# 1821516642** |

**Faculty Advisor:**

**Md. Shahriar Hussain**

**Senior Lecturer**

**ECE Department**

# LETTER OF TRANSMITAL

February, 2022

To

Dr. Rajesh Palit

Chairman,

Department of Electrical and Computer Engineering

North South University, Dhaka

Subject: Submission of Capstone Project Report on "**A Machine Learning and Embedded Approach to Detect Fire, Identify the Cause of it and Extinguish Using the Best Possible Extinguisher**"

Dear Sir,

With due respect, we would like to submit our Capstone Project Report on "**A Machine Learning and Embedded Approach to Detect Fire, Identify the Cause of it and Extinguish Using the Best Possible Extinguisher**" as a part of our BSc program. This project was very much valuable to us as it helped us gain experience from practical field and apply in real life. We tried to the maximum competence to meet all the dimensions required from this report.

We will be highly obliged if you kindly receive this report and provide your valuable judgment. It would be our immense pleasure if you find this report useful and informative to have an apparent perspective on the issue.

Sincerely Yours,

Md Rashiqur Rahman

ECE Department

North South University, Bangladesh

Sabbir Ahmed

ECE Department

North South University, Bangladesh

MD Shafin Islam Rudro

ECE Department

North South University, Bangladesh

Sadia Aktar

ECE Department

North South University, Bangladesh

# APPROVAL

Md. Rashiqur Rahman (ID #1812366042), Sabbir Ahmed (ID #1813515042), Md. Shafin Islam Rudro (ID #1821054642), and Sadia Aktar (ID #1821516042) from Electrical and Computer Engineering Department of North South University, have worked on the Senior Design Project titled "**A Machine Learning and Embedded Approach to Detect Fire, Identify the Cause of it and Extinguish Using the Best Possible Extinguisher**" under the supervision of Md. Shahriar Hussain partial fulfillment of the requirement for the degree of Bachelors of Science in Engineering and has been accepted as satisfactory.

**Supervisor's Signature**

……………………………………

**Md. Shahriar Hussain**

**Senior Lecturer**

Department of Electrical and Computer Engineering

North South University

Dhaka, Bangladesh.

**Chairman's Signature**

……………………………………

**Dr. Rajesh Palit**

**Professor**

Department of Electrical and Computer Engineering

North South University

Dhaka, Bangladesh.

# DECLARATION

This is to certify that this Project is our original work. No part of this work has been submitted elsewhere partially or fully for the award of any other degree or diploma. Any material reproduced in this project has been properly acknowledged.

Students' names & Signatures

**1. Md. Rashiqur Rahman**

Rashiqur Rahman

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

**2. Sabbir Ahmed**

[illegible]

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

**3. Md. Shafin Islam Rudro**

[illegible]

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

**4. Sadia Aktar**

[illegible]

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

# ACKNOWLEDGEMENTS

We would like to extend our sincere gratitude to the Almighty for bestowing us with the strength and determination to complete this project successfully. Our capstone project as part of the BSc curriculum has provided us with a valuable opportunity to apply the theoretical knowledge we have gained and gain real-world experience. We are deeply thankful to our teachers who guided us and provided us with the technical skills and support needed to successfully complete this project. Our special thanks go to Md. Shahriar Hussain, an esteemed member of the department, for his unwavering support and invaluable assistance throughout the project. We are also grateful to the ECE department at North South University for offering us the CSE 499 course which allowed us to delve deep into this project and bring it to fruition. Lastly, we would like to express our appreciation to our friends and family for their constant encouragement and support. Their contributions have been instrumental in making this project a reality.

# ABSTRACT

## A Machine Learning and Embedded Approach to Detect Fire, Identify the Cause of it and Extinguish Using the Best Possible Extinguisher

Fire accidents have become a major concern in Bangladesh, resulting in numerous deaths and devastating situations. To address this issue, a dynamic and portable remote-controlled prototype has been developed to monitor and extinguish chemical fires. The prototype utilizes a machine-learning approach for fire detection and an embedded approach to determine the cause of the fire. The system consists of a dashboard camera, microprocessor, gas sensors, and a microcontroller to provide information to the user and automatically extinguish the fire. The system was developed using appropriate datasets and algorithms, resulting in a high level of accuracy in fire detection while maintaining its cost-effectiveness and user-friendliness. The user can access all the information they need through a web-based application, and the device can be connected to a remote application over a mobile phone. The system produced satisfactory results, successfully detecting and extinguishing chemical fires with a high level of accuracy.

## CONTENTS

## LIST OF FIGURES

# Chapter 1
# Overview

## 1.1 Introduction

One of the main causes of financial damages and human catastrophe worldwide is fire, which can strike in a variety of settings, including mines, forests, residential buildings, offices, and schools. It is crucial to stop the spread of fire as soon as possible in order to lessen the damage that a fire causes and the associated financial losses. As a way of providing a prompt warning to report the beginning of a fire, fire detection is an essential component of surveillance systems that monitor environments. To measure fire probabilities up close, the majority of fire detectors use point sensors. The detector must be able to distinguish between a fire and a non-fire incident in order to reduce the likelihood of a false fire alarm being set off by the fire detection system. False fire alarms may be very expensive to respond to and can also disrupt services and production. Therefore, in order to create a dependable and effective system and avoid the difficulties associated with false alarms from fire detection systems, it is imperative to assess the present state of fire detection technology.

## 1.2 Project Description

Selection of a wrong fire extinguisher can aggravate the fire even more and increase the harms with it. Building a device able to do this exact job is the goal of this project.

Once activated manually or through the mobile app, the device will be able to navigate itself to the fire location. The device will be able to detect fire with the help of a camera sensor and confirm the fire outbreak. A gas sensor will scan the air in the vicinity of the fire and detect the element possibly responsible for the fire. Users within or outside of the area will be informed about the fire and provided with the analyzed results.

Once detected the element responsible for the outbreak, the device will suggest all the best possible extinguisher for the fire. In addition, the device will be capable of notifying the near hospitals and fire services. The system is multi-leveled and different at the same time due to these qualities.

## 1.3 Purpose of the Project

Thousands of people yearly die for different incidents revolving fire. Extinguishing fire as easy as it might sound, sometimes the wrong extinguisher can add more fuel to the fire costing precious lives. Targeting this specific area, the purpose of this project is to build a device capable of detecting the cause of fire, eliminating the risk of losing precious lives.

This will result in an overall improvement of the user's quality of life and safety.

## 1.4 Report Outine

This Project Report will discuss about the project in detail elaborated over five chapters. The chapters are:

**Chapter 1**: This chapter is a brief introduction to the reason, goal, and description of the project.

**Chapter 2**: This chapter contains all the background studies done for this project.

**Chapter 3:** Within this chapter, all the theoretical aspects and details about the engineering tools is described.

**Chapter 4**: All diagrams and illustrations regarding the project is shown in this chapter.

**Chapter 5**: Chapter 5 wrap ups the project report with a summary of the work done, Project Development Phases, Ethical and Professional Responsibilities, and Social Contribution.

# Chapter 2
# Related Work

## 2.1 Introduction

There are many existing papers related to home security systems as individual devices or few limited features but none comes with a great combo in such low cost.

## 2.2 Similar Existing Systems

The authors in this paper [1] presented a real-time system for automatic fire detection using color video input. A system like this has numerous crucial military and commercial uses and has a lot of benefits over conventional UV and infrared fire detectors. The benefits include better detection, fewer false alarms, and more detailed data regarding the location, size, and growth rate of the fire. We derive algorithms for fire detection based on the spectral, spatial, and temporal properties of fire events from the physical properties of fire. These algorithms have been integrated into a system that has been tested successfully on a wide range of fire and false alarm stimuli. We present experimental results demonstrating system performance on a burning jet fuel fire.

More than two thousand vehicles are damaged by unwanted fire on a daily basis. A not-so-distant contributory factor is the lack of a sophisticated fire safety system in the automobile. This has been addressed by designing and implementing a fuzzy logic control system with feedback from an Arduino microcontroller system. A medium-sized physical ear was used to test the automated system, which consists of flame sensors, temperature sensors, smoke sensors, and a redesigned transportable carbon dioxide air-conditioning unit. Results demonstrate that the auto fire detection and control system identify and puts out fire in less than 20 seconds without any false alarms. Utilizing new algorithms and fuzzy logic, an inventive and extremely promising hardware implementation solution module for fire detection and control for automobiles has been created [2].

This project involved the design, production, and testing of a mobile robot platform with an early fire detection unit. The robot was specifically designed to follow virtually prescribed courses using obstacle avoidance features and motion planning units and to scan the surroundings using a fire detection unit to find the fire source. The system detected the fire occurrence using smoke, flame, and temperature monitoring sensors. In order to obtain more dependable and highly precise findings from the fire detection unit Partial Least Square, a sensor-based fusion system was also created and proposed (PLS). Based on 200 sample calibration and validation data sets, PLS was used to generate the fire detection equation from measurement values for temperature, smoke, and flame. SAS/STAT (Statistical Analysis System) software was used for the development of the calibration and validation data sets [3]. To evaluate the system's capacity to identify gasoline, five different distances were chosen, measuring 100, 500, 600, 800, and 1000 mm. According to test results, the built robot can locate the fire source up to 1000 millimeters away. It can also be understood from the test results that the proposed sensor fusion system provides more reliable detection than one sensor-based system with an accuracy value of 92%.

This paper [4] presents a novel fire detection device in a highly complicated urban environment, such as an office building, supermarket, school, etc. A robot for fire detecting and/or search and rescue needs to move fast while searching wide area. For this, they developed a thermal image camera module which is capable of selectively acquiring very large field of view (FOV) panoramic view images or narrow field of view images from a single viewpoint, at video rates. An image processing and device controlling algorithm to detect a fire fast and accurately was developed as well. The developed device was tested in the indoor environments and the feasibility of the device was evaluated.

Support Vector Regression (SVR) was used to build a model to forecast the potential occurrence of flashover [5]. The forecast model is developed to use an array of heat detectors temperature data including in adjacent spaces, to recover temperature data from the room of fire origin and predict potential for flashover. Two special treatments, sequence segmentation and learning from fitting, are proposed to overcome the temperature limitation of heat detectors in real-life fire scenarios and to enhance prediction capabilities to determine if the flashover condition is met.

The authors in [6] presented a novel fire detection device in the highly complicated urban environment, such as an office building, supermarket, school, etc. A thermal image camera module is developed which is capable of selectively acquiring very large field of view (FOV) - panoramic view - images or narrow field of view images from a single viewpoint, at video rates. An image processing and device-controlling algorithm to detect a fire quickly and accurately were developed also

The robot was specially developed to track virtually prescribed paths with obstacle avoidance function through obstacle avoidance and motion planning units and to scan the environment in order to detect the fire source via fire detection unit [7]. A sensor-based fusion system was also developed and proposed to get more reliable and highly accurate results from the fire detection unit. Partial Least Square (PLS) was used to derive the fire detection equation from the temperature, smoke, and flame measurement values based on calibration and validation. SAS/STAT (Statistical Analysis System) software was used for the development of the calibration and validation data sets.

# Chapter 3
# Theoretical Background

## 3.1 Introduction

We've learned several important methods for implementing this project, and we're currently pushing it to completion. Following are the requirements that have all been covered.

## 3.2 Techniques and Skills

To detect the fire, we needed an advanced algorithm that is efficient to use. So, we’ve used the YOLO V5 algorithm to detect the fire without any hesitation.

The object identification method YOLO, which stands for "You Only Look Once," organizes photos into a grid. In the grid, each cell oversees finding items within it. Due to its accuracy and speed, YOLO is one of the most well-known object detection algorithms. YOLO V5 is approximately 180% quicker (140 FPS vs 50 FPS) and 88% smaller than YOLOv4 (27 MB vs. 244 MB).

Using the relevant dataset, this algorithm does image processing. For our experiment, we employed a dataset that included full-fire images that could be compared to camera-generated images.

Prior to employing sensors, we employed an important hardware known as the ESP32 Devkit. The Esp32 Devkit is a development board designed to test the ESP-WROOM-32 module. It is built around the ESP32 microprocessor, which supports Wi-Fi, Bluetooth, Ethernet, and Low Power in a single chip. The ESP32 already includes an antenna and RF balun, as well as a power amplifier, low-noise amplifiers, filters, and a power management module. The entire solution occupies the smallest amount of printed circuit board space. This board is utilized with TSMC 40nm low power 2.4 GHz dual-mode Wi-Fi and Bluetooth chips, which have the best power and RF attributes and are safe, dependable, and adaptable to a wide range of applications.

The primary goal of this project is to determine the Chemical Reason for the Fire and then put it out. In this case, datasets are utilized to identify the fire. However, due to a lack of data, we used a few sensors and assigned a parameter to each one based on the presence of a fire in the surrounding region. This attribute can be used to determine the cause of the flames. The sensors we are using are MQ-2 Gas Sensor, MQ-4 gas sensor, MQ-5 gas sensor, MQ-135 gas sensor, humidity Sensor, temperature sensor, and flame sensor.

Then there are two extinguishers for putting out the flames. The reason for two extinguishers is that if the fire is not caused by a chemical, it will be extinguished with a single extinguisher. However, if the fire was caused by a chemical, it will be extinguished by the other extinguisher.

We are using an IoT platform to send notifications about the fire and its cause. The Blynk IoT platform is what we're using. Blynk is an IoT platform for iOS and Android smartphones that uses the Internet to control Arduino, Raspberry Pi, and NodeMCU. This application is used to compile and provide the required address on the available widgets to create a graphical interface or human-machine interface (HMI).

## 3.3 Equipment Lists

Here are some of the tools we'll need to put this project together.

### 3.3.1 ESP32 Devkit Module

We have used ESP32 Devkit Module (shown in Fig 3.1) in our project. The ESP32 Devkit module is an entry-level, low-footprint development board from the ESP32 series. The peripheral set on this board is extensive. The ESP32's internal pinout has been designed for hassle-free prototyping [7].

Figure 3.1: ESP32 Devkit Module [8].

### 3.3.2 ESP32 Cam Module

The ESP32 CAM module is a compact, low-power camera module built on the ESP32 platform. It features an inbuilt TF card slot and an OV2640 camera. Numerous clever IoT applications, including WiFi image upload, QR identification, wireless video monitoring, and others, can make use of the ESP32-CAM [9].

Figure 3.2: ESP32 Cam Module [9].

### 3.3.3 DHT11 Temperature and Humidity Sensor

The DHT11 is a simple and inexpensive digital temperature and humidity sensor. It measures the airflow with a capacitive humidity sensor and a thermistor and outputs a digital signal on the data pin. It is quite simple to operate, however, data collection requires precise timing [10].

Figure 3.3: DHT11 Temperature and Humidity Sensor [11].

### 3.3.4 Infrared Flame Detection Sensor Module

This sensor basically detects IR (Infrared) light wavelength between 760 nm – 1100 nm that is emitted from the fire flame or light source. The flame sensor comes with a YG1006 Phototransistor sensor which is a high speed and high sensitivity [12].

Figure 3.4: Infrared Flame Detection Sensor Module [12].

### 3.3.5 MQ-2 Gas Sensor

MQ2 Gas Sensor is a Metal Oxide Semiconductor (MOS) type Gas Sensor used to identify gases such as methane, butane, LPG, smoke, and others. Because gas detection is based on a change in the resistance of the sensing material as a result of contact with the gas, it is also known as chemiresistors [13].

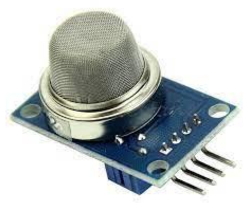


Figure 3.5: MQ-2 Gas Sensor [14].

### 3.3.6 MQ-4 Gas Sensor

The MQ4 Gas Sensor is mainly used to detect the methane (CNG) gas concentration in the air, either at home or in industry. This sensor has a detecting element made primarily of ceramic with an aluminum-oxide base and a tin dioxide coating that is encased in a stainless-steel mesh [15].

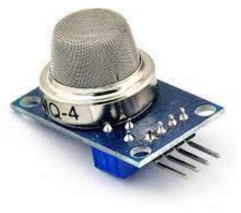


Figure 3.6: MQ-4 Gas Sensor [16].

### 3.3.7 MQ-5 Gas Sensor

The MQ5 is used in gas leak detection equipment in consumer and industrial applications. This sensor is suitable for detecting LPG, natural gas, and coal gas. Avoid the noise of alcohol, cooking fumes, and cigarette smoke. The sensitivity can be adjusted by the potentiometer [17].

Figure 3.7: MQ-5 Gas Sensor [17].

### 3.3.8 MQ-135 Gas Sensor

The MQ-135 Gas sensor can detect gases like Ammonia (NH3), sulfur (S), Benzene (C6H6), CO2, and other harmful gases and smoke. Like the others in the MQ series of gas sensors, this sensor has a pin for both digital and analog output. The digital pin turns high when the amount of these gases in the air exceeds a predetermined threshold [18].

Figure 3.8: MQ-135 Gas Sensor [18].

# Chapter 4
# Result and Analysis

## 4.1 Project Illustration.

The whole project was done for a prototype that is remotely controllable. It is also helpful to collect the chemical gases around the prototype, find the fire using sensors and a camera with a machine learning approach, and let the user know through a web-based application. After collecting and detecting the fire, the prototype will use the best extinguisher based on the surrounding situation. For example, if only gases are gathered around the prototype, the application will notify the user of the gases and recommend a suitable extinguisher for those chemical gases. In addition, if the prototype detects only fire, it will extinguish it with its reserve water. Also, after the prototype finds a fire and collects gas simultaneously, it will put out the fire with its CO2 reserve. Also, after detecting the fire through the machine-learning approach of the prototype, it will extinguish the fire with its reserve water.

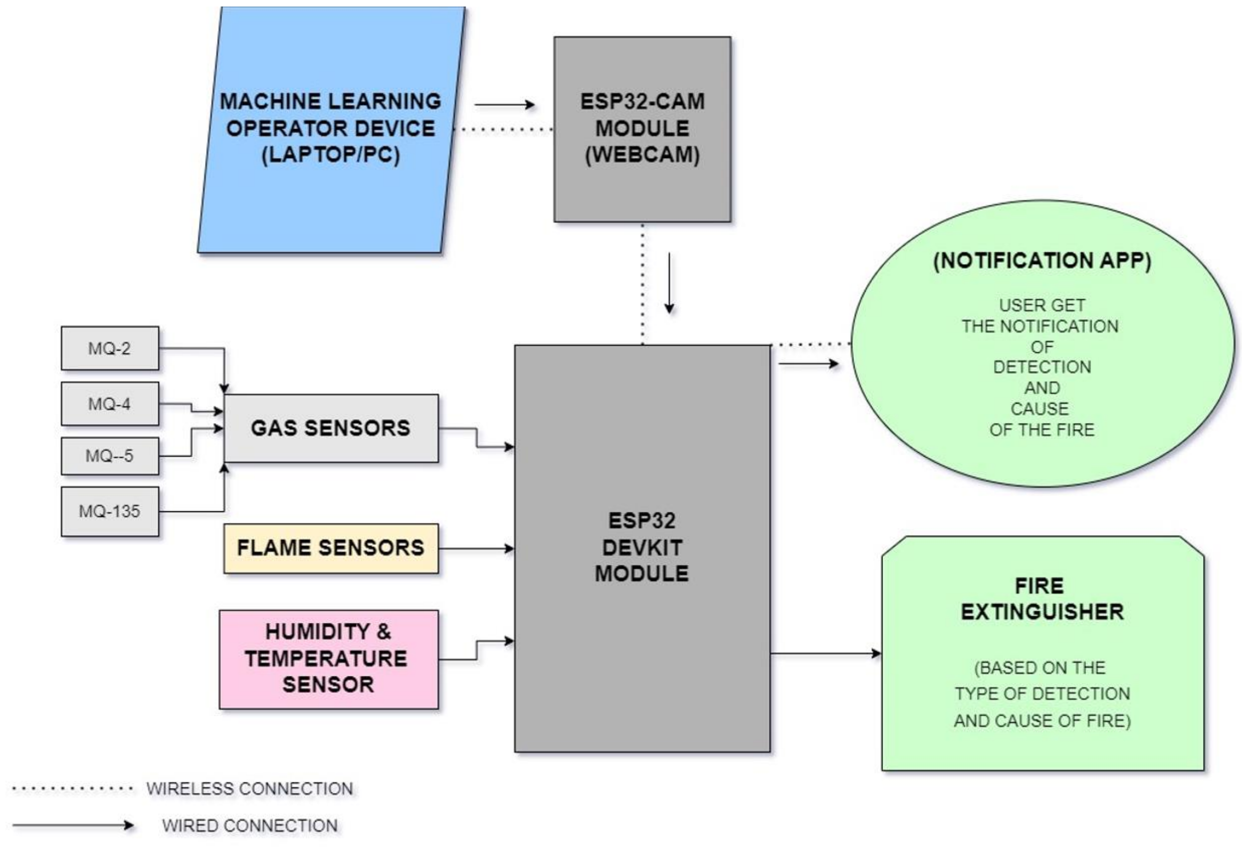
The embedded approach has been used for collecting the chemical gases around the prototype. Through this approach, many sensors and microcontrollers have been used. The ESP32 DevKit Module, in particular, has been used as a microcontroller. Two microcontrollers were used: one for collecting chemical gases, flames, and temperatures, and another for controlling the car via a mobile phone app. Four sensors have been used to collect chemical gases: MQ-2, MQ-4, MQ-5, and MQ-135. Also, to collect the environment's temperature and humidity and the flame of the fire, DHT11 and flame sensors have been used.

Another microcontroller, the ESP32 CAM, has been used as a wireless camera. A machine-learning approach was used to detect the fire. An algorithm called Yolo V5 has been used for the machine learning approach. Some datasets have been used to train the model to detect fire. After implementing this part of the project, the results show higher accuracy.

Overall, the prototype has been made to detect the cause of the fire and extinguish it with the best extinguisher based on the situation around the prototype. Nowadays, this kind of prototype is essential in our country.

## 4.2 System Diagram

Below, the entire system diagrams have been shown. This specifies the actual model of this project.



MACHINE LEARNING OPERATOR DEVICE (LAPTOP/PC)
ESP32-CAM MODULE (WEBCAM)
(NOTIFICATION APP)
USER GET THE NOTIFICATION OF DETECTION AND CAUSE OF THE FIRE
MQ-2
MQ-4
MQ--5
MQ-135
GAS SENSORS
FLAME SENSORS
HUMIDITY & TEMPERATURE SENSOR
ESP32 DEVKIT MODULE
FIRE EXTINGUISHER
(BASED ON THE TYPE OF DETECTION AND CAUSE OF FIRE)
WIRELESS CONNECTION
WIRED CONNECTION

Figure 4.1: System Diagram

Our fire detection system utilizes YOLOV5 to detect flames via webcam. In order to keep costs low, we have utilized an ESP32 microcontroller equipped with a camera module as the primary source of image capture. In addition to the webcam, the system is equipped with various other sensors including a flame sensor, gas sensor, smoke sensor, and temperature sensor. In the event of a fire, the system will trigger an alarm in the form of a buzzer and notify users through a web-based application. With the use of gas sensors such as the MQ-2 and MQ-5, the system is able to accurately determine the cause of the fire and extinguish it accordingly.

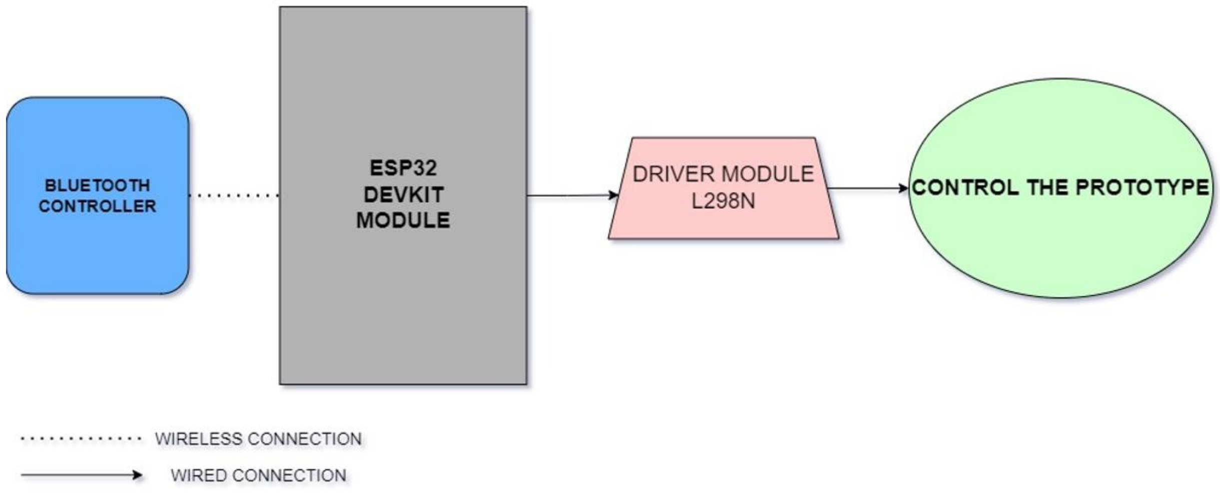


Figure 4.2: Prototype Control Diagram.

Figure 4.2 showcases the 4-wheeled vehicle we created to transport and manage our system. The car is powered by an ESP32 module and controlled through a smartphone using an android application called "Bluetooth RC Controller." The driver module used is L298N. The operator can easily control the car by viewing its surroundings through the attached webcam.

Through the system diagram, we've split our work in two parts. Hardware and Software.

## 4.2.1 Hardware Part

This hardware component contains the construction of the prototype's embedded equipment. We have divided the hardware component's completion into three sections. A car chassis, an L298N motor module, a Li-po battery, a DC switch, and an ESP32 DevKit Module microcontroller were used to construct the body of the remote-control prototype. With gas sensors (MQ-2, MQ-4, MQ-5, and MQ-135), a flame sensor, a DHT11 sensor, an ESP32 DevKit Module microcontroller, a buzzer, a 2-channel 5V relay module, two water pumps, two 9V batteries, two tanks (one for water, another for CO2), and two servo motors, the gas, flame, humidity, and temperature detection and extinguisher model were constructed. The webcam was then constructed using an ESP32 Cam microcontroller. Due to the combination of the three phases, the concept appeared to be an attractive prototype.

### 4.2.2 Software Part

Using the Arduino IDE, the entire embedded system code has been written. This code provides the whole sensor and microcontroller functionality of the prototype, including automobile control, fire detection, chemical gas collection, and situation resolution.

As webcam fire detection involves machine learning, the Python Integrated Development Environment was utilized to create the code. This facilitates the identification of the fire with efficiency and precision. Bluetooth RC Controller, an Android application, has been used to control the prototype. BLYNK, a web application, was utilized to receive notification of the prototype's alarming surrounding circumstances. In this application, users can quickly receive a notification on the fire's origin.

## 4.3 Results

From the pre-trained model, we get some result which has been shown below.

Figure 4.3: Input of Pretrained Model.

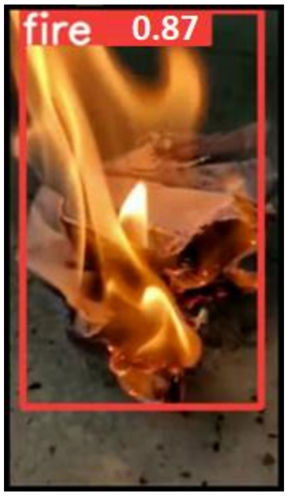


Figure 4.4: Output of Pre-trained Model.

As was already indicated, it is being utilized as a pretrained model before the project's practical appearance is created. So, here is what the pretrained model produced. An instructional video has first been made. In Fig. 4.4, it is clear that captured video was employed in the YOLO V5 algorithm's pretrained model procedure. The result of the pretrained model, Figure 4.5, illustrates how the fire was correctly and successfully recognized from the fire detection dataset. The precision has been demonstrated flawlessly in this outcome, as we had anticipated.

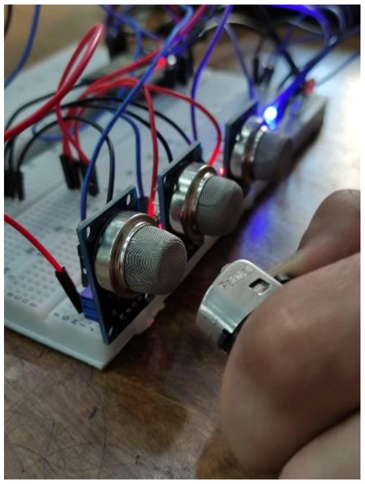


Figure 4.5: MQ-2 Sensor Test Result

Figure 4.5 shows the test result of MQ-2 gas sensor. Whenever the presence of gas crossed 500 ppm, a blue LED got turned on as a signal.

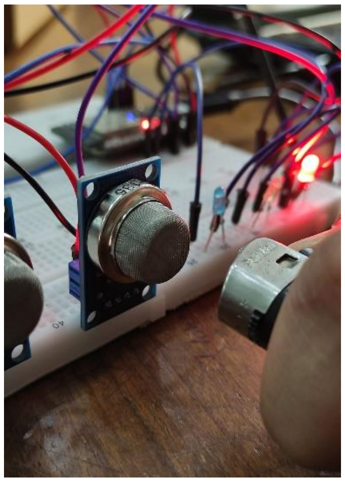


Figure 4.6: MQ-135 Sensor Test Result

Figure 4.6 shows the test result of MQ-135 gas sensor. Whenever the presence of gas crossed 500 ppm, a red LED got turned on as a signal.

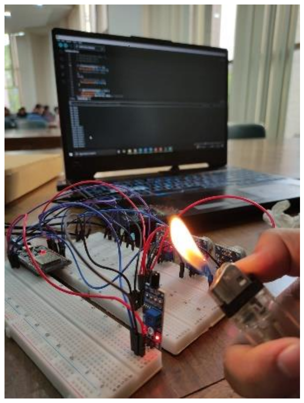
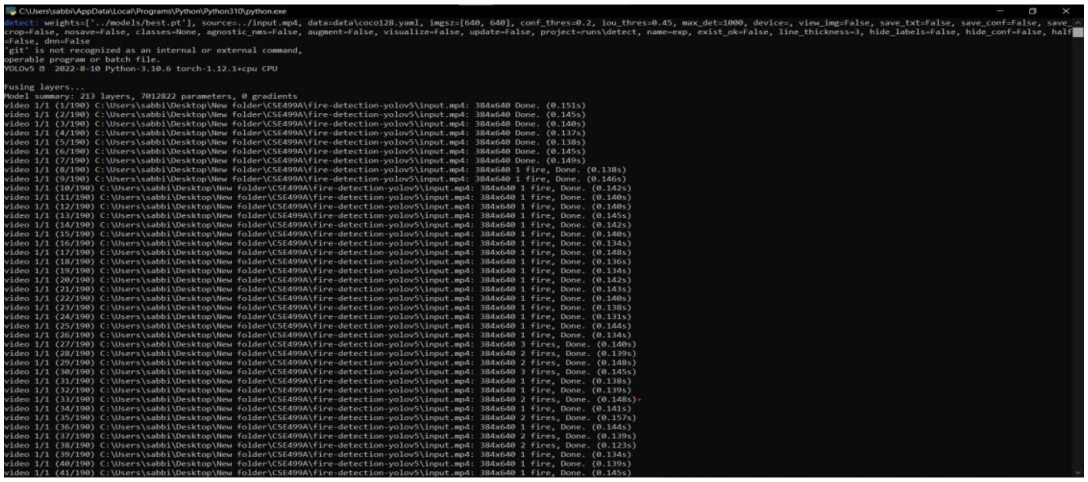


Figure 4.7: Flame Sensor Test Result

Figure 4.7 shows the test result of flame sensor. Whenever there was presence of fire a buzzer gets on to alert the surroundings.

**Code Execution:**



```
C:\Users\sabbi\AppData\Local\Programs\Python\Python310\python.exe
detect: weights=['../models/best.pt'], source=../input.mp4, data=data\coco128.yaml, imgsz=[640, 640], conf_thres=0.2, iou_thres=0.45, max_det=1000, device=, view_img=False, save_txt=False, save_conf=False, save_
crop=False, nosave=False, classes=None, agnostic_nms=False, augment=False, visualize=False, update=False, project=runs\detect, name=exp, exist_ok=False, line_thickness=3, hide_labels=False, hide_conf=False, half
=False, dnn=False
'git' is not recognized as an internal or external command,
operable program or batch file.
YOLOv5 ▯  2022-8-10 Python-3.10.6 torch-1.12.1+cpu CPU

Fusing layers...
Model summary: 213 layers, 7012822 parameters, 0 gradients
video 1/1 (1/190) C:\Users\sabbi\Desktop\New folder\CSE499A\fire-detection-yolov5\input.mp4: 384x640 Done. (0.151s)
video 1/1 (2/190) C:\Users\sabbi\Desktop\New folder\CSE499A\fire-detection-yolov5\input.mp4: 384x640 Done. (0.145s)
video 1/1 (3/190) C:\Users\sabbi\Desktop\New folder\CSE499A\fire-detection-yolov5\input.mp4: 384x640 Done. (0.140s)
video 1/1 (4/190) C:\Users\sabbi\Desktop\New folder\CSE499A\fire-detection-yolov5\input.mp4: 384x640 Done. (0.137s)
video 1/1 (5/190) C:\Users\sabbi\Desktop\New folder\CSE499A\fire-detection-yolov5\input.mp4: 384x640 Done. (0.138s)
video 1/1 (6/190) C:\Users\sabbi\Desktop\New folder\CSE499A\fire-detection-yolov5\input.mp4: 384x640 Done. (0.145s)
video 1/1 (7/190) C:\Users\sabbi\Desktop\New folder\CSE499A\fire-detection-yolov5\input.mp4: 384x640 Done. (0.149s)
video 1/1 (8/190) C:\Users\sabbi\Desktop\New folder\CSE499A\fire-detection-yolov5\input.mp4: 384x640 1 fire, Done. (0.138s)
video 1/1 (9/190) C:\Users\sabbi\Desktop\New folder\CSE499A\fire-detection-yolov5\input.mp4: 384x640 1 fire, Done. (0.146s)
video 1/1 (10/190) C:\Users\sabbi\Desktop\New folder\CSE499A\fire-detection-yolov5\input.mp4: 384x640 1 fire, Done. (0.142s)
video 1/1 (11/190) C:\Users\sabbi\Desktop\New folder\CSE499A\fire-detection-yolov5\input.mp4: 384x640 1 fire, Done. (0.140s)
video 1/1 (12/190) C:\Users\sabbi\Desktop\New folder\CSE499A\fire-detection-yolov5\input.mp4: 384x640 1 fire, Done. (0.140s)
video 1/1 (13/190) C:\Users\sabbi\Desktop\New folder\CSE499A\fire-detection-yolov5\input.mp4: 384x640 1 fire, Done. (0.145s)
video 1/1 (14/190) C:\Users\sabbi\Desktop\New folder\CSE499A\fire-detection-yolov5\input.mp4: 384x640 1 fire, Done. (0.142s)
video 1/1 (15/190) C:\Users\sabbi\Desktop\New folder\CSE499A\fire-detection-yolov5\input.mp4: 384x640 1 fire, Done. (0.140s)
video 1/1 (16/190) C:\Users\sabbi\Desktop\New folder\CSE499A\fire-detection-yolov5\input.mp4: 384x640 1 fire, Done. (0.134s)
video 1/1 (17/190) C:\Users\sabbi\Desktop\New folder\CSE499A\fire-detection-yolov5\input.mp4: 384x640 1 fire, Done. (0.148s)
video 1/1 (18/190) C:\Users\sabbi\Desktop\New folder\CSE499A\fire-detection-yolov5\input.mp4: 384x640 1 fire, Done. (0.136s)
video 1/1 (19/190) C:\Users\sabbi\Desktop\New folder\CSE499A\fire-detection-yolov5\input.mp4: 384x640 1 fire, Done. (0.134s)
video 1/1 (20/190) C:\Users\sabbi\Desktop\New folder\CSE499A\fire-detection-yolov5\input.mp4: 384x640 1 fire, Done. (0.142s)
video 1/1 (21/190) C:\Users\sabbi\Desktop\New folder\CSE499A\fire-detection-yolov5\input.mp4: 384x640 1 fire, Done. (0.143s)
video 1/1 (22/190) C:\Users\sabbi\Desktop\New folder\CSE499A\fire-detection-yolov5\input.mp4: 384x640 1 fire, Done. (0.140s)
video 1/1 (23/190) C:\Users\sabbi\Desktop\New folder\CSE499A\fire-detection-yolov5\input.mp4: 384x640 1 fire, Done. (0.138s)
video 1/1 (24/190) C:\Users\sabbi\Desktop\New folder\CSE499A\fire-detection-yolov5\input.mp4: 384x640 1 fire, Done. (0.131s)
video 1/1 (25/190) C:\Users\sabbi\Desktop\New folder\CSE499A\fire-detection-yolov5\input.mp4: 384x640 1 fire, Done. (0.144s)
video 1/1 (26/190) C:\Users\sabbi\Desktop\New folder\CSE499A\fire-detection-yolov5\input.mp4: 384x640 1 fire, Done. (0.134s)
video 1/1 (27/190) C:\Users\sabbi\Desktop\New folder\CSE499A\fire-detection-yolov5\input.mp4: 384x640 3 fires, Done. (0.140s)
video 1/1 (28/190) C:\Users\sabbi\Desktop\New folder\CSE499A\fire-detection-yolov5\input.mp4: 384x640 2 fires, Done. (0.139s)
video 1/1 (29/190) C:\Users\sabbi\Desktop\New folder\CSE499A\fire-detection-yolov5\input.mp4: 384x640 2 fires, Done. (0.148s)
video 1/1 (30/190) C:\Users\sabbi\Desktop\New folder\CSE499A\fire-detection-yolov5\input.mp4: 384x640 3 fires, Done. (0.145s)
video 1/1 (31/190) C:\Users\sabbi\Desktop\New folder\CSE499A\fire-detection-yolov5\input.mp4: 384x640 1 fire, Done. (0.138s)
video 1/1 (32/190) C:\Users\sabbi\Desktop\New folder\CSE499A\fire-detection-yolov5\input.mp4: 384x640 1 fire, Done. (0.139s)
video 1/1 (33/190) C:\Users\sabbi\Desktop\New folder\CSE499A\fire-detection-yolov5\input.mp4: 384x640 2 fires, Done. (0.148s)
video 1/1 (34/190) C:\Users\sabbi\Desktop\New folder\CSE499A\fire-detection-yolov5\input.mp4: 384x640 1 fire, Done. (0.141s)
video 1/1 (35/190) C:\Users\sabbi\Desktop\New folder\CSE499A\fire-detection-yolov5\input.mp4: 384x640 2 fires, Done. (0.157s)
video 1/1 (36/190) C:\Users\sabbi\Desktop\New folder\CSE499A\fire-detection-yolov5\input.mp4: 384x640 1 fire, Done. (0.144s)
video 1/1 (37/190) C:\Users\sabbi\Desktop\New folder\CSE499A\fire-detection-yolov5\input.mp4: 384x640 2 fires, Done. (0.139s)
video 1/1 (38/190) C:\Users\sabbi\Desktop\New folder\CSE499A\fire-detection-yolov5\input.mp4: 384x640 2 fires, Done. (0.123s)
video 1/1 (39/190) C:\Users\sabbi\Desktop\New folder\CSE499A\fire-detection-yolov5\input.mp4: 384x640 1 fire, Done. (0.134s)
video 1/1 (40/190) C:\Users\sabbi\Desktop\New folder\CSE499A\fire-detection-yolov5\input.mp4: 384x640 1 fire, Done. (0.139s)
video 1/1 (41/190) C:\Users\sabbi\Desktop\New folder\CSE499A\fire-detection-yolov5\input.mp4: 384x640 1 fire, Done. (0.145s)
```

Figure 4.8: Code Execution

Figure 4.8 shows the code execution of the YOLOV5. The code takes the image/video as input and matches layer by layer to determine if there is fire or not.

## 4.4 Simulation

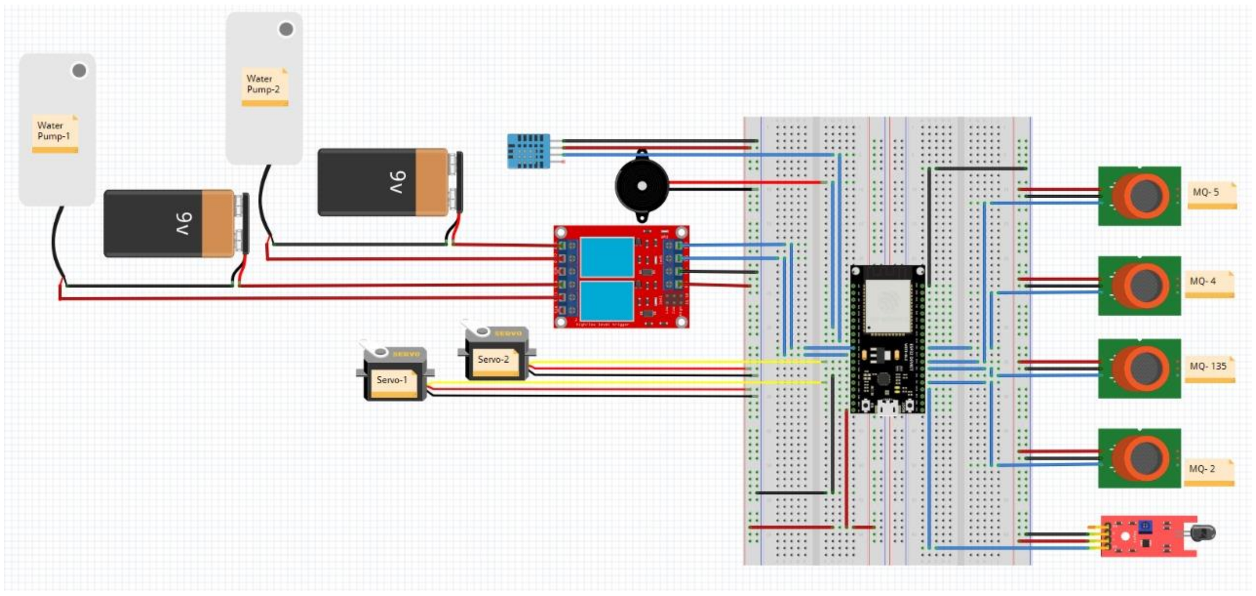


Figure 4.9: Circuit Simulation.

Figure 4.9 is the circuit simulation of our project. We have implemented it in Fritzing. We simulated our experiment first here in Fritzing then we implemented it in hardware.

## 4.5 Project Prototype:

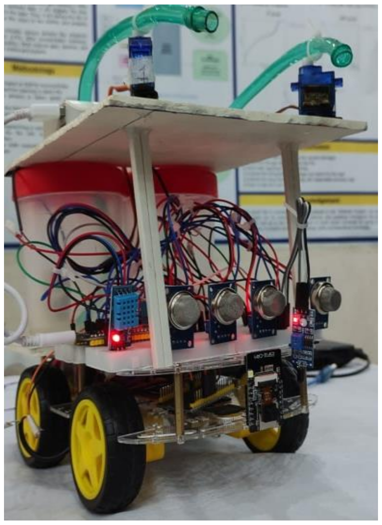


Figure 4.10: Final Project Prototype

The Fire Fighting Robot prototype is designed based on ESP32 microcontroller and it detects fire using Machine Learning. The robot is able to detect the flame, smoke and also gases in the emergency situations and suggest the best extinguisher to control the fire. Whenever fire/flame, smoke, gases is noticed it gives the buzzer as an alert to nearby people and sends notification to users. It is controlled through the Arduino programming. The robot is remotely controlled with an app connected via Bluetooth which can be used to extinguish fire where fire fighters may struggle to reach. The camera and flame sensor connected to the micro-controller detects the flame in its surroundings. The robot can turn 360 degrees to detect fire in all directions. The water pump and dry chemical pump is connected to the pipe so that it is able to reach the area to spray easily depending what caused the fire. Here the speed of motors can be controlled by Pulse Width

Modulation technique. Thus, our project is able to reduce the injury to victims and also property damage to certain extent.

## 4.6 Summary:

A portable and dynamic remote-controlled prototype has been developed to tackle the increasing number of fire accidents in Bangladesh. The prototype uses a machine learning approach and embedded system to detect the fire, determine its cause and provide the best solution to prevent further damage. The prototype consists of a dashboard camera, microprocessor, gas sensors and a web-based application for the user to access all the necessary information. The device has been developed with appropriate datasets and algorithms and has shown to be accurate in detecting fire. The remote-control feature allows the user to move the prototype and connect with the application through a mobile phone. The prototype is economical, portable, easy to use, and marketable in Bangladesh.

# Chapter 5
# Conclusion

.

## 5.1 Summery of Work

The main idea of our project is to build a device that can detect fire using either photo or video scanning and find the main cause of this fire. It will analyze the image or video and then if it detects any fire, it will give us signal. Then it will give us the actual cause of the fire and the best way to extinguish it.

## 5.2 Project Development Phases

We divided our project mainly into seven phases. But through our CSE499A, we have only done four phases. These four phases are the planning phase, literature phase, data collecting phase, and processing phase.

The planning phase: We began by gathering ideas and planning what we would do with our project. It took 2 weeks to plan our project.

Literature review phase: In this phase, we did our literature reviews. We read a total of fifteen papers similar to our projects. We did our literature review for 4 weeks.

Data collecting phase: In this phase, we started to collect data sets about detecting fire. Then we began looking for data sets to determine the cause of the fire. Although we found a dataset for detecting fire, we couldn’t find any dataset for finding the cause of the fire. It took 2 weeks to find the data set for our project.

Possessing phase: We implement and test the fire detection data set on our laptop after collecting it. This phase took 2 weeks to complete.

In CSE499B, we completed our final phase, which was the practical implementation of our project. It took around 1 month to finish this phase.

## 5.3 Ethical and Professional Responsibilities

In our project, there is no lack of professionalism and our work is completely ethical. We approached our work with seriousness and completed it responsibly. While working, we didn't damage the environment.

Our device will feature a camera and a few fire detection sensors. The device will activate the fire signal when it detects any fire using its camera. Sensors can also be used to detect a fire. In order to prevent further damage or a serious accident, it will then try to determine the origin of the fire and the best way to put it out. Additionally, it will send a signal to the closest fire station to alert them about the fire in the meantime. Social Contribution

Our project can have an impact in many social sectors. It can have an impact on economic, social, health and safety. In our society, we can see many accidents happening because of fire. Sometimes, because of late or failed attempts to extinguish fire, it can cause more disastrous accidents. So, our project can prevent those kinds of accidents. This way, it can improve our health and safety sector. If the accident count decreases because of our project, then it will be good for our society and economy.

# Bibliography

[1] G. Healey, D. Slater, T. Lin, B.Drda and A.D. Geodeke, “A system for eal-time fire detection”. CONFERENCE PAPER

[2] Jun Wang, Wai Cheong Tam, and Richard Peacock, “P-Flash - A Machine Learning-based Model for Flashover Prediction using Recovered Temperature Data”. SUBMITTED PAPER

[3] Jung-Hoon Hwang, Sewoong Jun, Kaehoon Jeon, and Jongbae Lee, “Fire Detection Device for Robotic Fire Fighting”. CONFERENCE PAPER.

[4] H. SayginSucuoglu, Ismail Bogrekci, and Pinar Demircioglu, “Development of Mobile Robot with Sensor Fusion Fire Detection Unit”. JOURNAL

[5] Robert Sowah, Kwame O. Ampadu, Abdul Ofoli, Koudjo Koumadi, Godfrey A. Mills, and Joseph Nortey, “Design and implementation of a fire detection and control system for automobiles using fuzzy logic”. CONFERENCE PAPER

[6] [Vyshani M B, Shikha Sureh, “IoT Technology Based Fire-Fighter Robot”. JOURNAL

[7] "ESP32-DevKitC." ESP32-DevKitC Board | Espressif. Accessed: February 12, 2023. [Online]. Available: https://www.espressif.com/en/products/devkits/esp32-devkitc

[8] "ESP32 Development Board - DEVKIT V1." grobotronics.com. Accessed: February 12, 2023. [Online]. Available: https://grobotronics.com/esp32-development-board-devkit-v1.html?sl=en

[9] "Camera Module Based on ESP32." grobotronics.com. Accessed: February 12, 2023. [Online]. Available: https://grobotronics.com/camera-module-based-on-esp32.html?sl=en

[10] "DHT11 temperature-humidity sensor." Evil Mad Scientist Shop. Accessed: February 12, 2023. [Online]. Available: https://shop.evilmadscientist.com/productsmenu/716

[11] "DHT11 Temperature and Humidity Sensor." Naba Tech Shop. Accessed: February 12, 2023. [Online]. Available: https://nabatechshop.com/product/dht11-temperature-and-humidity-sensor/

[12] "Flame Sensor Fire Detection Module IR Infrared 4 Wire Flame." Udvabony.com - Electronics, Sensors, Robotics Online Shop. Accessed: February 12, 2023. [Online]. Available: https://udvabony.com/product/flame-sensor-module/

[13] "In-Depth: How MQ2 Gas/Smoke Sensor Works? & Interface it with Arduino." Last Minute Engineers. Accessed: February 12, 2023. [Online]. Available: https://lastminuteengineers.com/mq2-gas-senser-arduino-tutorial/

[14] "MQ-2 Flammable Gas & Smoke Sensor Robotics Bangladesh." RoboticsBD. Accessed: February 12, 2023. [Online]. Available: https://store.roboticsbd.com/sensors/671-mq-2-flammable-gas-smoke-sensor-robotics-bangladesh.html

[15] "MQ-4 Methane Gas Sensor Module." QuartzComponents. Accessed: February 12, 2023. [Online]. Available: https://quartzcomponents.com/products/mq-4-gas-sensor-module

[16] "MQ-4 Methane Natural Gas Sensor Robotics Bangladesh." RoboticsBD. Accessed: February 12, 2023. [Online]. Available: https://store.roboticsbd.com/sensors/673-mq-4-methane-natural-gas-sensor-robotics-bangladesh.html

[17] "MQ-5 Smoke Gas Detector Sensor Robotics Bangladesh." RoboticsBD. Accessed: February 12, 2023. [Online]. Available: https://store.roboticsbd.com/sensors/674-mq-5-smoke-gas-detector-sensor-robotics-bangladesh.html

[18] "MQ-135 Gas Sensor Robotics Bangladesh." RoboticsBD. Accessed: February 12, 2023. [Online]. Available: https://store.roboticsbd.com/sensors/679-mq-135-gas-sensor-robotics-bangladesh.html

# Appendices

```
# YOLOv5   by Ultralytics, GPL-3.0 license
"""
Run inference on images, videos, directories, streams, etc.

Usage - sources:
    $ python path/to/detect.py --weights yolov5s.pt --source 0              # webcam
                                                             img.jpg        # image
                                                             vid.mp4        # video
                                                             path/          # directory
                                                             path/*.jpg     # glob

Usage - formats:
    $ python path/to/detect.py --weights yolov5s.pt                 # PyTorch
                                         yolov5s.torchscript        # TorchScript
                                         yolov5s.onnx               # ONNX Runtime or OpenCV DNN with --dnn
                                         yolov5s.xml                # OpenVINO
                                         yolov5s.engine             # TensorRT
                                         yolov5s.mlmodel            # CoreML (macOS-only)
                                         yolov5s_saved_model        # TensorFlow SavedModel
                                         yolov5s.pb                 # TensorFlow GraphDef
                                         yolov5s.tflite             # TensorFlow Lite
                                         yolov5s_edgetpu.tflite     # TensorFlow Edge TPU
"""
```

```
import argparse

import os

import platform

import sys

from pathlib import Path


import torch

import torch.backends.cudnn as cudnn


FILE = Path(__file__).resolve()

ROOT = FILE.parents[0]  # YOLOv5 root directory

if str(ROOT) not in sys.path:

    sys.path.append(str(ROOT))  # add ROOT to PATH

ROOT = Path(os.path.relpath(ROOT, Path.cwd()))  # relative


from models.common import DetectMultiBackend

from utils.dataloaders import IMG_FORMATS, VID_FORMATS, LoadImages, LoadStreams

from utils.general import (LOGGER, check_file, check_img_size, check_imshow,
check_requirements, colorstr, cv2,

                           increment_path, non_max_suppression, print_args, scale_coords,
strip_optimizer, xyxy2xywh)

from utils.plots import Annotator, colors, save_one_box

from utils.torch_utils import select_device, time_sync
```

```
@torch.no_grad()
def run(
        weights=ROOT / 'yolov5s.pt',  # model.pt path(s)
        source=ROOT / 'data/images',  # file/dir/URL/glob, 0 for webcam
        data=ROOT / 'data/coco128.yaml',  # dataset.yaml path
        imgsz=(640, 640),  # inference size (height, width)
        conf_thres=0.25,  # confidence threshold
        iou_thres=0.45,  # NMS IOU threshold
        max_det=1000,  # maximum detections per image
        device='',  # cuda device, i.e. 0 or 0,1,2,3 or cpu
        view_img=False,  # show results
        save_txt=False,  # save results to *.txt
        save_conf=False,  # save confidences in --save-txt labels
        save_crop=False,  # save cropped prediction boxes
        nosave=False,  # do not save images/videos
        classes=None,  # filter by class: --class 0, or --class 0 2 3
        agnostic_nms=False,  # class-agnostic NMS
        augment=False,  # augmented inference
        visualize=False,  # visualize features
        update=False,  # update all models
        project=ROOT / 'runs/detect',  # save results to project/name
        name='exp',  # save results to project/name
        exist_ok=False,  # existing project/name ok, do not increment
        line_thickness=3,  # bounding box thickness (pixels)
        hide_labels=False,  # hide labels
        hide_conf=False,  # hide confidences
        half=False,  # use FP16 half-precision inference
        dnn=False,  # use OpenCV DNN for ONNX inference
):
    source = str(source)
    save_img = not nosave and not source.endswith('.txt')  # save inference images
    is_file = Path(source).suffix[1:] in (IMG_FORMATS + VID_FORMATS)
    is_url = source.lower().startswith(('rtsp://', 'rtmp://', 'http://', 'https://'))
    webcam = source.isnumeric() or source.endswith('.txt') or (is_url and not is_file)
    if is_url and is_file:
        source = check_file(source)  # download

    # Directories
    save_dir = increment_path(Path(project) / name, exist_ok=exist_ok)  # increment run
    (save_dir / 'labels' if save_txt else save_dir).mkdir(parents=True, exist_ok=True)  # make dir

    # Load model
    device = select_device(device)
    model = DetectMultiBackend(weights, device=device, dnn=dnn, data=data, fp16=half)
    stride, names, pt = model.stride, model.names, model.pt
    imgsz = check_img_size(imgsz, s=stride)  # check image size
```

```
# Dataloader
if webcam:
    view_img = check_imshow()
    cudnn.benchmark = True  # set True to speed up constant image size inference
    dataset = LoadStreams(source, img_size=imgsz, stride=stride, auto=pt)
    bs = len(dataset)  # batch_size
else:
    dataset = LoadImages(source, img_size=imgsz, stride=stride, auto=pt)
    bs = 1  # batch_size
vid_path, vid_writer = [None] * bs, [None] * bs

# Run inference
model.warmup(imgsz=(1 if pt else bs, 3, *imgsz))  # warmup
seen, windows, dt = 0, [], [0.0, 0.0, 0.0]
for path, im, im0s, vid_cap, s in dataset:
    t1 = time_sync()
    im = torch.from_numpy(im).to(device)
    im = im.half() if model.fp16 else im.float()  # uint8 to fp16/32
    im /= 255  # 0 - 255 to 0.0 - 1.0
    if len(im.shape) == 3:
        im = im[None]  # expand for batch dim
    t2 = time_sync()
    dt[0] += t2 - t1
```

```
# Inference
visualize = increment_path(save_dir / Path(path).stem, mkdir=True) if visualize else False
pred = model(im, augment=augment, visualize=visualize)
t3 = time_sync()
dt[1] += t3 - t2

# NMS
pred = non_max_suppression(pred, conf_thres, iou_thres, classes, agnostic_nms,
max_det=max_det)
dt[2] += time_sync() - t3

# Second-stage classifier (optional)
# pred = utils.general.apply_classifier(pred, classifier_model, im, im0s)

# Process predictions
for i, det in enumerate(pred):  # per image
    seen += 1
    if webcam:  # batch_size >= 1
        p, im0, frame = path[i], im0s[i].copy(), dataset.count
        s += f'{i}: '
    else:
        p, im0, frame = path, im0s.copy(), getattr(dataset, 'frame', 0)
    p = Path(p)  # to Path
```

```
            save_path = str(save_dir / p.name)  # im.jpg
            txt_path = str(save_dir / 'labels' / p.stem) + ('' if dataset.mode == 'image' else f'_{frame}')
# im.txt
            s += '%gx%g ' % im.shape[2:]  # print string
            gn = torch.tensor(im0.shape)[[1, 0, 1, 0]]  # normalization gain whwh
            imc = im0.copy() if save_crop else im0  # for save_crop
            annotator = Annotator(im0, line_width=line_thickness, example=str(names))
            if len(det):
                # Rescale boxes from img_size to im0 size
                det[:, :4] = scale_coords(im.shape[2:], det[:, :4], im0.shape).round()

                # Print results
                for c in det[:, -1].unique():
                    n = (det[:, -1] == c).sum()  # detections per class
                    s += f"{n} {names[int(c)]}{'s' * (n > 1)}, "  # add to string

                # Write results
                for *xyxy, conf, cls in reversed(det):
                    if save_txt:  # Write to file
                        xywh = (xyxy2xywh(torch.tensor(xyxy).view(1, 4)) / gn).view(-1).tolist()  #
normalized xywh
                        line = (cls, *xywh, conf) if save_conf else (cls, *xywh)  # label format
                        with open(f'{txt_path}.txt', 'a') as f:
```

```
                    f.write(('%g ' * len(line)).rstrip() % line + '\n')

                if save_img or save_crop or view_img:  # Add bbox to image
                    c = int(cls)  # integer class
                    label = None if hide_labels else (names[c] if hide_conf else f'{names[c]}
{conf:.2f}')
                    annotator.box_label(xyxy, label, color=colors(c, True))
                if save_crop:
                    save_one_box(xyxy, imc, file=save_dir / 'crops' / names[c] / f'{p.stem}.jpg',
BGR=True)

        # Stream results
        im0 = annotator.result()
        if view_img:
            if platform.system() == 'Linux' and p not in windows:
                windows.append(p)
                cv2.namedWindow(str(p), cv2.WINDOW_NORMAL |
cv2.WINDOW_KEEPRATIO)  # allow window resize (Linux)
                cv2.resizeWindow(str(p), im0.shape[1], im0.shape[0])
            cv2.imshow(str(p), im0)
            cv2.waitKey(1)  # 1 millisecond

        # Save results (image with detections)
```

```
            if save_img:
                if dataset.mode == 'image':
                    cv2.imwrite(save_path, im0)
                else:  # 'video' or 'stream'
                    if vid_path[i] != save_path:  # new video
                        vid_path[i] = save_path
                        if isinstance(vid_writer[i], cv2.VideoWriter):
                            vid_writer[i].release()  # release previous video writer
                        if vid_cap:  # video
                            fps = vid_cap.get(cv2.CAP_PROP_FPS)
                            w = int(vid_cap.get(cv2.CAP_PROP_FRAME_WIDTH))
                            h = int(vid_cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
                        else:  # stream
                            fps, w, h = 30, im0.shape[1], im0.shape[0]
                        save_path = str(Path(save_path).with_suffix('.mp4'))  # force *.mp4 suffix on
results videos
                        vid_writer[i] = cv2.VideoWriter(save_path, cv2.VideoWriter_fourcc(*'mp4v'),
fps, (w, h))
                    vid_writer[i].write(im0)

        # Print time (inference-only)
        LOGGER.info(f'{s}Done. ({t3 - t2:.3f}s)')
```

```
    # Print results
    t = tuple(x / seen * 1E3 for x in dt)  # speeds per image
    LOGGER.info(f'Speed: %.1fms pre-process, %.1fms inference, %.1fms NMS per image at
shape {(1, 3, *imgsz)}' % t)
    if save_txt or save_img:
        s = f"\n{len(list(save_dir.glob('labels/*.txt')))} labels saved to {save_dir / 'labels'}" if save_txt
else ''
        LOGGER.info(f"Results saved to {colorstr('bold', save_dir)}{s}")
    if update:
        strip_optimizer(weights[0])  # update model (to fix SourceChangeWarning)


def parse_opt():
    parser = argparse.ArgumentParser()
    parser.add_argument('--weights', nargs='+', type=str, default=ROOT / 'yolov5s.pt', help='model
path(s)')
    parser.add_argument('--source', type=str, default=ROOT / 'data/images',
help='file/dir/URL/glob, 0 for webcam')
    parser.add_argument('--data', type=str, default=ROOT / 'data/coco128.yaml', help='(optional)
dataset.yaml path')
    parser.add_argument('--imgsz', '--img', '--img-size', nargs='+', type=int, default=[640],
help='inference size h,w')
    parser.add_argument('--conf-thres', type=float, default=0.25, help='confidence threshold')
    parser.add_argument('--iou-thres', type=float, default=0.45, help='NMS IoU threshold')
    parser.add_argument('--max-det', type=int, default=1000, help='maximum detections per
image')
    parser.add_argument('--device', default='', help='cuda device, i.e. 0 or 0,1,2,3 or cpu')
    parser.add_argument('--view-img', action='store_true', help='show results')
    parser.add_argument('--save-txt', action='store_true', help='save results to *.txt')
    parser.add_argument('--save-conf', action='store_true', help='save confidences in --save-txt
labels')
    parser.add_argument('--save-crop', action='store_true', help='save cropped prediction boxes')
    parser.add_argument('--nosave', action='store_true', help='do not save images/videos')
    parser.add_argument('--classes', nargs='+', type=int, help='filter by class: --classes 0, or --
classes 0 2 3')
    parser.add_argument('--agnostic-nms', action='store_true', help='class-agnostic NMS')
    parser.add_argument('--augment', action='store_true', help='augmented inference')
    parser.add_argument('--visualize', action='store_true', help='visualize features')
    parser.add_argument('--update', action='store_true', help='update all models')
    parser.add_argument('--project', default=ROOT / 'runs/detect', help='save results to
project/name')
    parser.add_argument('--name', default='exp', help='save results to project/name')
    parser.add_argument('--exist-ok', action='store_true', help='existing project/name ok, do not
increment')
    parser.add_argument('--line-thickness', default=3, type=int, help='bounding box thickness
(pixels)')
```

```
    parser.add_argument('--hide-labels', default=False, action='store_true', help='hide labels')
    parser.add_argument('--hide-conf', default=False, action='store_true', help='hide confidences')
    parser.add_argument('--half', action='store_true', help='use FP16 half-precision inference')
    parser.add_argument('--dnn', action='store_true', help='use OpenCV DNN for ONNX inference')
    opt = parser.parse_args()
    opt.imgsz *= 2 if len(opt.imgsz) == 1 else 1  # expand
    print_args(vars(opt))
    return opt


def main(opt):
    check_requirements(exclude=('tensorboard', 'thop'))
    run(**vars(opt))


if __name__ == "__main__":
    opt = parse_opt()
    main(opt)
```

## Hardware Code:

```
#include <ESP32Servo.h>
#include <DHT.h>
#define DHTTYPE DHT11
```

```
Servo myservo1;
Servo myservo2;

int x;

//humidity sensor
int humidity = 25;
DHT dht(humidity, DHTTYPE);


//int Buzzer135 = 27;
int Gas_analog135 = 4;

// for mq2
//int Buzzer2 = 23;
int Gas_analog2 = 2;

// for mq4
//int Buzzer4 = 32;
int Gas_analog4 = 5;

// for mq5
//int Buzzer5 = 26;
int Gas_analog5 = 16;

//flame sensor
int buzzerPin = 32;
int flamePin = 15;

int Flame = HIGH;


int pump1 = 27 ;
int pump2 = 26;
int pos1 = 0;
int pos2 = 0;
boolean fire1 = false;
boolean fire2 = false;


void setup() {
  Serial.begin(115200);
  pinMode(Gas_analog2, INPUT);
  pinMode(Gas_analog4, INPUT);
  pinMode(Gas_analog5, INPUT);
  pinMode(Gas_analog135, INPUT);
  pinMode(pump1, OUTPUT);
  pinMode(pump2, OUTPUT);
  digitalWrite(pump1, HIGH);
  digitalWrite(pump2, HIGH);
  myservo1.attach(13);
  myservo1.write(90);
  myservo2.attach(14);
  myservo2.write(90);
  Serial.println(F("DHTxx test!"));
  dht.begin();
  pinMode(buzzerPin, OUTPUT);
  pinMode(flamePin, INPUT);
  Serial.setTimeout(1);

}

void loop() {
  myservo1.write(90);
  myservo2.write(90);

  int gassensorAnalog2 = analogRead(Gas_analog2);
  int gassensorAnalog4 = analogRead(Gas_analog4);
  int gassensorAnalog5 = analogRead(Gas_analog5);
  int gassensorAnalog135 = analogRead(Gas_analog135);

  while (!Serial.available());
  x = Serial.readString().toInt();

  if(x>0){
    digitalWrite(buzzerPin, HIGH);
  }

  Flame = digitalRead(flamePin);
  if (Flame == LOW)
  {
    digitalWrite(buzzerPin, HIGH);
    Serial.println("Fire");


    // digitalWrite(redled, HIGH);
    // digitalWrite(greenled, LOW);
  }
  else
  {
    digitalWrite(buzzerPin, LOW);
    Serial.println("No Fire");

  //   digitalWrite(greenled, HIGH);
  //   digitalWrite(redled, LOW);
  }



  if (Flame == LOW && (gassensorAnalog2 > 50 || gassensorAnalog4 > 500 ||
gassensorAnalog5 > 500 || gassensorAnalog135 > 150))
  {
    // digitalWrite(buzzerPin, HIGH);
    // Serial.println("Fire");
    fire1 = true;
  }

  // else{
  //   digitalWrite(buzzerPin, LOW);

  // }

  if (Flame == LOW && (gassensorAnalog2 < 500 || gassensorAnalog4 < 500 ||
gassensorAnalog5 < 500 || gassensorAnalog135 < 150))
  {
    // digitalWrite(buzzerPin, HIGH);
    // Serial.println("Fire");
    fire2 = true;
  }

  if( gassensorAnalog2 > 100)
    {
     Serial.print("LPG Gas is Detected in MQ2 = ");
     Serial.print(gassensorAnalog2);
     Serial.println("");
    //  fire = true;
    }
  else
   {
    //  Serial.println("MQ2 CLear");
   }

  if( gassensorAnalog4 > 500)
    {
     Serial.print("C.Methane Gas is Detected in MQ4 = ");
     Serial.print(gassensorAnalog4);
     Serial.println("");
    //  fire = true;
    }
  else
   {

    //  Serial.println("MQ4 CLear");
   }

  if( gassensorAnalog5 > 500)
    {
     Serial.print("NG Gas is Detected in MQ5 = ");
     Serial.print(gassensorAnalog5);
     Serial.println("");
    //  fire = true;
    }
  else
   {
    //  Serial.println("MQ5 CLear");
   }

  if( gassensorAnalog135 > 150)
    {
     Serial.print("Bad Air Quaility Detected in MQ135 = ");
     Serial.print(gassensorAnalog135);
     Serial.println("");

    }
  else
   {
    //  Serial.println("Air Quality Stable");
   }



  float h = dht.readHumidity();
  float t = dht.readTemperature();
  float f = dht.readTemperature(true);

  if (isnan(h) || isnan(t) || isnan(f)) {
    Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }
```

```
  float hif = dht.computeHeatIndex(f, h);
  float hic = dht.computeHeatIndex(t, h, false);

  // Serial.print(F("Humidity: "));
  // Serial.print(h);
  // Serial.print(F("%  Temperature: "));
  // Serial.print(t);
  // Serial.print(F("°C "));
  // Serial.print(f);
  // Serial.print(F("°F  Heat index: "));
  // Serial.print(hic);
  // Serial.print(F("°C "));
  // Serial.print(hif);
  // Serial.println(F("°F"));

  while(fire1==true){
    put_off_fire();
    Serial.println("Fire Fire!");
  }

  while(fire2==true){
    put_off_fire_water();
    Serial.println("Fire Fire!");
  }
  delay(100);
}

void put_off_fire(){
  digitalWrite(pump1, LOW);
  delay(1000);
  for(pos1=50; pos1<=180; pos1+=1){
    myservo1.write(pos1);
    Serial.println(pos1);
    delay(10);
  }
  for(pos1=180; pos1>=50; pos1-=1){
    myservo1.write(pos1);
    delay(10);
  }


  digitalWrite(pump1, HIGH);
  myservo1.write(90);
  fire1 = false;
}
```

```
void put_off_fire_water(){
  digitalWrite(pump2, LOW);
  delay(1000);

  for(pos2=50; pos2<=180; pos2+=1){
    myservo2.write(pos2);
    Serial.println(pos2);
    delay(10);
  }
  for(pos2=180; pos2>=50; pos2-=1){
    myservo2.write(pos2);
    delay(10);
  }
  digitalWrite(pump2, HIGH);
  myservo2.write(90);
  fire2 = false;
}
```