



slington college
(इस्लिङ्टन कलेज)

CS5068NI– Cloud Computing & IoT

Automated Fire Chasing and Extinguishing Robot

Assessment Type

50% Group Report

Semester

2024 Spring

Group Members

London Met ID	Student Name
23047609	Aarzo Pandey
23047604	Tulasa Giri
23047592	Anupam Shrestha
23047579	Sittal Karki
23047593	Sulav Bhattarai

Assignment Due Date: May 15, 2025

Assignment Submission Date: May 14, 2025

Submitted to: Mr. Sugat Man Shakya

Word Count: 6391

I confirm that I understand my coursework needs to be submitted online via My Second Teacher under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a mark of zero will be awarded.

Document 3.docx

Islington College,Nepal

Document Details

Submission ID
trn:oid::3618:95892124

Submission Date
May 14, 2025, 9:34 PM GMT+5:45

Download Date
May 14, 2025, 9:39 PM GMT+5:45

File Name
Document 3.docx

File Size
41.7 KB

34 Pages
6,391 Words
34,338 Characters

8% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Match Groups

- 47 Not Cited or Quoted 7%
Matches with neither in-text citation nor quotation marks
- 8 Missing Quotations 1%
Matches that are still very similar to source material
- 0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation
- 0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

- 3% Internet sources
- 1% Publications
- 8% Submitted works (Student Papers)





Integrity Flags

0 Integrity Flags for Review

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Match Groups

-  **47 Not Cited or Quoted 7%**
Matches with neither in-text citation nor quotation marks
-  **8 Missing Quotations 1%**
Matches that are still very similar to source material
-  **0 Missing Citation 0%**
Matches that have quotation marks, but no in-text citation
-  **0 Cited and Quoted 0%**
Matches with in-text citation present, but no quotation marks

Top Sources

-  **3% Internet sources**
-  **1% Publications**
-  **8% Submitted works (Student Papers)**

Acknowledgment

We are highly thankful to our module leader Mr. Sugat Man Shakya and our class tutor Mr. Bishnu Pandey for allowing us to work on this project on Automated Fire Chasing and Extinguishing Robot. They have provided us with tremendous guidance, encouragement, and support throughout the course, which has been constantly helping us in exploring and understanding various aspects of IoT technology. Equally, we thank our friends whose collaboration shared ideas and technical knowledge provided the project much-needed momentum. Finally, we thank everyone who directly or indirectly supported us in bringing this project to full fruition.

Abstract

Fire hazards in sensitive environments such as server rooms can pose a significant threat to critical infrastructure, human safety, and business continuity. This project introduces an **Automated Fire Chasing and Extinguishing Robot**, designed to detect and suppress small fires autonomously using embedded sensors and actuators. The system leverages Internet of Things (IoT) principles, integrating flame detection via infrared (IR) sensors, autonomous movement through BO motors, and targeted extinguishing using a mini water pump controlled by a servo motor, all orchestrated by an Arduino Uno microcontroller. Upon detecting a fire, the robot determines the flame's location, navigates towards it, and activates the water pump to extinguish it. This low-cost, modular prototype is particularly suitable for deployment in confined, high-risk environments where early intervention is crucial. The project demonstrates the potential for IoT-driven robotics in enhancing fire safety and minimizing response time during critical incidents.

Table of contents

1.	Introduction.....	1
1.1.	Current Scenario	1
1.2.	Problem Statement and Project as Solution.....	2
1.3.	Aim and Objectives.....	4
2.	Background.....	5
2.1.	System Overview	5
2.2.	Design and Diagram	6
2.2.1.	Block Diagram	6
2.2.2.	System Architecture.....	7
2.2.3.	Circuit Diagram	8
2.2.4.	Schematic Diagram.....	10
2.2.5.	Flowchart	11
2.3.	Requirement Analysis.....	12
2.3.1.	Hardware Components.....	12
2.3.2.	Software Components	22
3.	Development	26
	Step 1: Design and Planning	26
	Step 2: Resource Collection.....	27
	Step 3: System Development	27
	Step 4: Pins and Connection	35
	Step 5: Writing Program	37
4.	Results and Findings	38
4.1.	Testing.....	39
5.	Future works	45

6.	Conclusion	46
7.	References	47
8.	Appendix	49
8.1.	Source code	49
8.2.	Individual Contribution Table.....	56
8.3.	Individual Contribution Structure	57

Table of Figures

Figure 1 Block diagram	6
Figure 2 System Architecture of Automated Fire Chasing and Extinguishing Robot.....	7
Figure 3 Circuit Diagram.....	8
Figure 4 Pin connection table	9
Figure 5 Schematic Diagram	10
Figure 6 Flowchart.....	11
Figure 7 IR Flame sensor.....	12
Figure 8 Arduino UNO	13
Figure 9 BO motor and wheel.....	14
Figure 10 Mini Servo	15
Figure 11 Water pump	16
Figure 12 Li-ion battery	17
Figure 13 Breadboard	18
Figure 14 Male-Female jumper wire	19
Figure 15 Male to Male jumper wire	19
Figure 16 Diode	20
Figure 17 TIP-122 Transistor.....	21
Figure 18 1K Ohm Resistor	22
Figure 19 Arduino IDE	22
Figure 20 Microsoft Word	23
Figure 21 Wraw.io	23
Figure 22 Cirkuit IDE	24
Figure 23 Fritzing	24
Figure 24 Skeleton of the robot	28
Figure 25 System of robot.....	28
Figure 26 Integrating components	29
Figure 27 Wiring Flame sensor.....	30
Figure 28 Flame sensor	31
Figure 29 Connecting BO motor to motor driver	32
Figure 30 Connecting motor driver to Arduino	32

Figure 31 Connecting mini servo.....	33
Figure 32 Uploading code through Arduino IDE	34
Figure 33 Writing program in Arduino IDE	37
Figure 34 Position before igniting fire.....	39
Figure 35 Position of robot after sensing fire.	40
Figure 36 Water pump was activated.....	41
Figure 37 Sensors struggled to sense fire in bright condition.....	42
Figure 38 Water pump shut off after extinguishing fire	43
Figure 39 Robot not navigating around obstacle.	44

Table of Table

Table 1 Test 1..... 39

Table 2 Test 2..... 40

Table 3 Test 3..... 41

Table 4 Test 4..... 42

Table 5 Test 5..... 43

1. Introduction

In high-risk environments such as server rooms, laboratories, and small-scale industrial facilities, fire outbreaks can cause severe operational and financial damage. Early detection and rapid suppression are essential to prevent such incidents from escalating. This project, titled **“Automated Fire Chasing and Extinguishing Robot”**, presents a functional prototype designed to detect and suppress fire using autonomous navigation and sensor-based decision-making.

The system leverages core principles of the Internet of Things (IoT), where interconnected devices are equipped with sensors and actuators operate with minimal human intervention. Using an Arduino microcontroller as the central control unit, the robot integrates infrared flame sensors to detect fire, DC motors for mobility, and a servo-controlled water pump system for targeted suppression. Sensor data is processed in real-time to determine flame direction, allowing the robot to automatically navigate toward the source and extinguish it using a mounted water jet.

The design emphasizes modularity, portability, and autonomous functionality, making it suitable for compact and critical environments such as server rooms. By combining embedded systems, controlling logic, and physical components into a single automated solution, this project offers a scalable foundation for future integration with building management systems and larger safety infrastructures.

This report outlines the complete development process, including circuit design, wiring, integration, system architecture, and the challenges encountered during assembly and testing.

1.1. Current Scenario

Server rooms and data centers are highly vulnerable to fire hazards due to their dense concentration of electrical equipment and continuous operation. The presence of high-voltage power supplies, complex wiring systems, and overheated hardware creates an environment where even a minor malfunction can quickly escalate into a dangerous fire. With increasing reliance on digital infrastructure, the risks and consequences of such fires are also growing globally. (Impact Fire, 2025)

- In August 2018, a major fire broke out in a Tokyo building suspected to be an AWS data center under construction. The fire burned for over eight hours, resulting in five deaths

and fifty injuries, with a significant portion of the building severely damaged. (Sensor, 2021)

- In March 2023, a fire at the Maxnod Datacenter in Saint-Trivier-sur-Moignans, Ain, France, destroyed an entire 800 m² facility. All equipment inside was lost, and the site required a complete reconstruction.
- Later that year, in September 2023, a fire at Windstream's data center in Lincoln, Nebraska caused approximately \$200,000 in damages. The incident disrupted emergency services, including 911 systems across multiple counties, exposing serious flaws in centralized communication infrastructure. (Zhang, Dgtl Infra, 2023)
- Locally, in 2019, a fire at Subisu's headquarters in Kathmandu caused by an electrical short circuit resulted in severe server damage. The disruption affected over 250,000 cable subscribers and led to significant service outages and financial loss. (Himalayan News Service, 2019)

These cases show the scale of destruction that can be caused by fires in server environments.

In many of these scenarios, fires spread quickly due to delayed human response or lack of on-site safety mechanisms. Automated systems offer a clear advantage in such environments. Fire detection, localization, and suppression can be carried out without human involvement, reducing damage and avoiding life-threatening situations. Autonomous systems like the one proposed in this project can respond in real time, providing a safer and more reliable solution to fire emergencies in high-risk areas.

1.2. Problem Statement and Project as Solution

Fire safety is a vital requirement in any modern organization, especially within environments like server rooms that contain sensitive hardware and critical digital infrastructure. These spaces typically contain a high density of electrical components, extensive wiring, and other flammable materials making them extremely vulnerable to electrical fires. Once a fire ignites, there is an immediate risk to both equipment and data, as well as service continuity. Below are some key fire hazards present in server rooms, along with how our proposed solution directly addresses them:

A. High Risk of Fire Due to Electrical Overload

Problem: Server rooms operate with high-voltage equipment, power-hungry servers, and complex wiring systems. These conditions raise the chances of short circuits, overheating, and electrical sparks, all of which can easily lead to a fire.

Solution: Our robot is equipped with multiple flame sensors that actively scan the surroundings. Upon detecting even the slightest indication of fire, the robot immediately navigates toward the source and suppresses it at the earliest stage, reducing the chances of escalation

B. Delay in Human Response

Problem: Fires in server rooms often break out during off-hours when no personnel are present. Any delay in spotting and responding to a fire increases the extent of damage significantly.

Solution: The autonomous design of our robot allows it to continuously monitor and respond to fire incidents in real time, regardless of human presence. This rapid, automated response minimizes delays and prevents fire from spreading or causing further harm.

C. High Cost of Conventional Fire Suppression Systems

Problem: Advanced fire suppression systems such as FM200, Novec 1230, or CO₂ flooding setups are costly to purchase, install, and maintain. This places them out of reach for smaller organizations, academic institutions, and budget-restricted IT setups.

Solution: Our firefighting robot offers a low-cost, accessible alternative. Built from affordable and widely available components such as microcontrollers, flame sensors, servo motors, and a basic water pump, it brings essential fire response functionality to resource-limited environments without compromising effectiveness

D. Safety Risks to Firefighting Personnel

Problem: In the event of a fire, human responders face extreme hazards, burns, toxic smoke, and even electrocution. These risks are amplified in confined, electronics-heavy spaces like server rooms.

Solution: Our robot removes the need for human intervention by autonomously navigating to the fire zone and extinguishing it. This ensures complete safety for personnel while maintaining an effective emergency response. It is especially beneficial for locations where human access is delayed or unsafe.

In conclusion, the system we propose not only addresses immediate fire threats in sensitive environments but also offers a reliable, scalable, and affordable solution to bridge the gap between modern safety demands and practical implementation.

1.3. Aim and Objectives

To build a cost-effective, autonomous fire detection and suppression system tailored for high-risk, enclosed environments such as server rooms, using Arduino and sensor-based automation.

Objectives:

- To design and assemble a mobile fire-fighting robot equipped with flame sensors, a water pump, and Arduino microcontroller for real-time fire detection.
- To implement motor-driven navigation that enables the robot to locate and move toward fire sources without external control.
- To test and evaluate the robot's performance specifically in a simulated server room setup with common heat sources and cable arrangements.
- To minimize human involvement in fire emergencies by ensuring the robot can operate independently during off-hours or in hard-to-access areas.
- To validate the system as a low-cost alternative to commercial suppression systems for small organizations, educational labs, and local server installations.

2. Background

2.1. System Overview

This project was developed with the main goal of minimizing the risk of losing important data stored in servers, and more importantly, helping to reduce the danger to human life in case of fire. Our system is built using components like the Arduino Uno, a driving motor, four BO motors, a battery, a breadboard, and various wiring connections. The battery is one of the most important parts, as it acts as the main power supply for the entire system. The breadboard helps us distribute the power to all components easily without needing to solder anything, which makes building and testing a lot more manageable.

In our prototype, once the sensor detects a flame, it sends a signal that activates the water pumping system, which is designed to extinguish the fire. While our current version works, it's still a prototype, so there are many improvements that can be made in the future especially in terms of efficiency, accuracy, and safety features. But as it stands, this project has laid a strong foundation for a system that can help protect server rooms and other sensitive environments.

2.2. Design and Diagram

2.2.1. Block Diagram

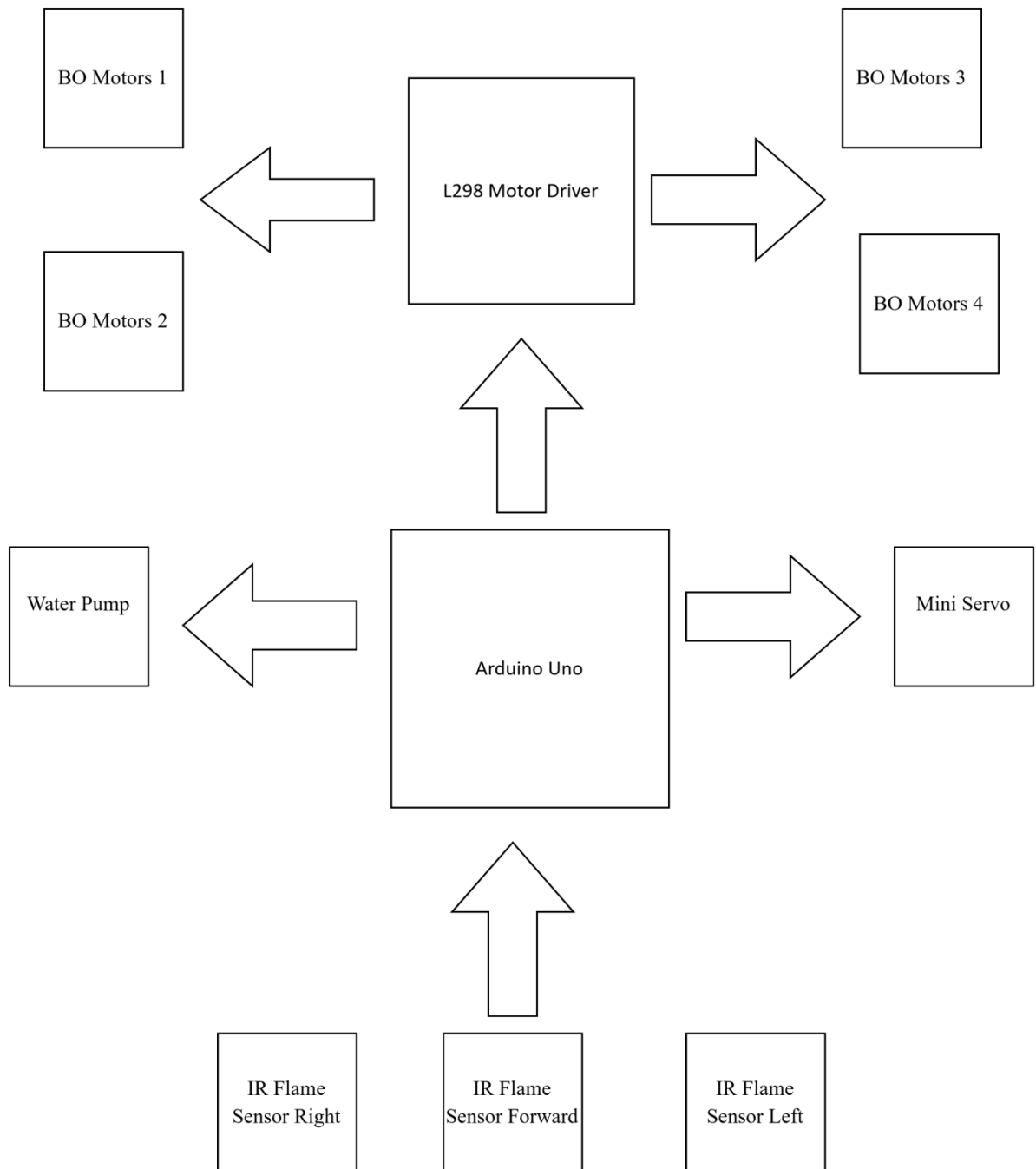


Figure 1 Block diagram

2.2.2. System Architecture

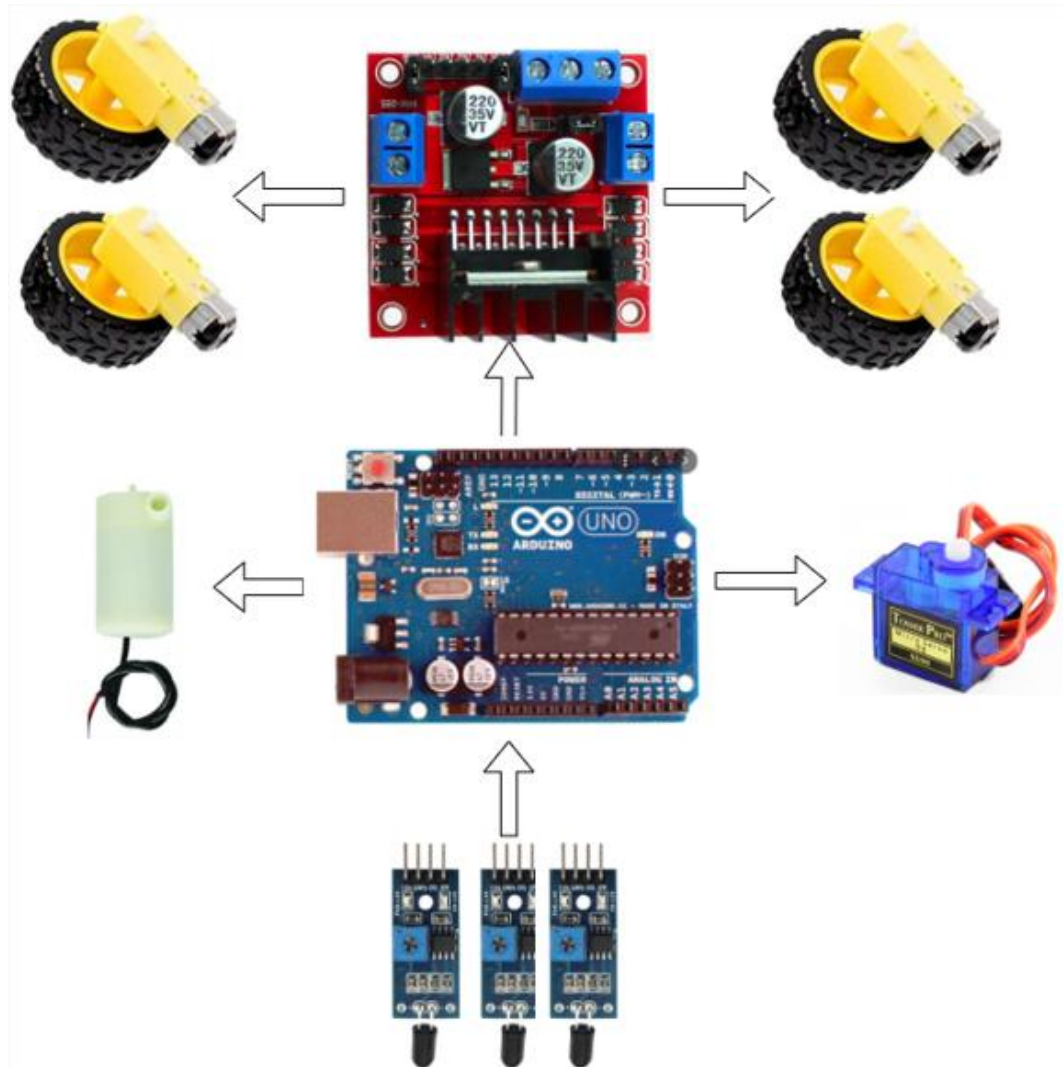


Figure 2 System Architecture of Automated Fire Chasing and Extinguishing Robot

The above diagram demonstrates the system architecture of Automated Fire Chasing and Extinguishing Robot. The Diagram helps to understand all the necessary components and shows how the input and output works in the system. As shown in architecture, IR flame sensors send an input signal to the microcontroller Arduino UNO which control mini servo, water pump and Motor driver with B.O motor accordingly.

2.2.3. Circuit Diagram

An circuit diagram is a simplified graphical representation of an electrical circuit using standardized symbols to show components like resistors, batteries, switches, LEDs, and capacitors, along with the wires that connect them.

The following circuit diagram shows the connection of our project Automated fire chasing and extinguishing robot.

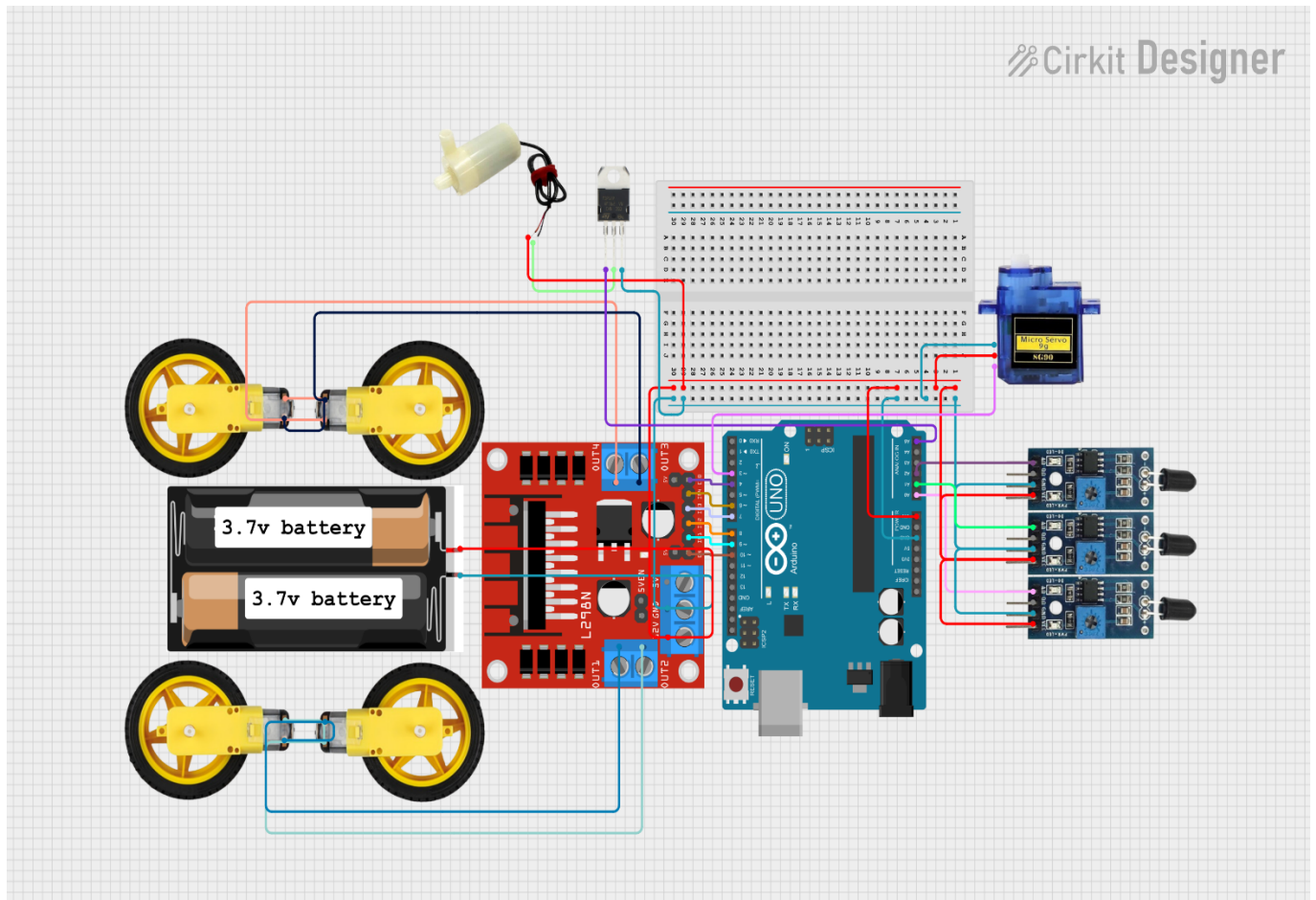


Figure 3 Circuit Diagram.

S no	Parts	Pins available	Motor driver	Arduino uno	Breadboard	Transistor
1	IR sensor ● Right sensor ● Forward sensor ● Left sensor	A0,Gnd, Vcc		Analog A2,A1,A0	Vcc: +12v Gnd:Gnd	
2	Bo moter ● Moter 1 ● Moter 2 ● Moter 3 ● Moter 4	Positive (+) Negative(-)	(+):Out 1 (-) :Out2 (+):out3 (-):Out4			
3	Mini servo	+5v/Gnd/ Pwm		Pwm:Digital (3)	Vcc:+12v Gnd:Gnd	
4	Water pump	Positive (+) Negative(-)			Positive(+): +12v	Negative(-) : Collector
5	Battery	Positive (+) Negative(-)	Vcc:12v Gnd:Gnd		Vcc:12v Gnd :Gnd	
6	Transistor	Collector Emitter Base			Emitter:Gnd(-)	
7	Motor driver	IN1, IN2, IN3, IN4, EN A, EN B, VCC, GND, OUT1, OUT2, OUT3, OUT4		Digital10:ENA Digital 9:IN1 Digital 8:IN2 Digital 7:IN3 Digital 6:IN4 Digital 4:ENB	Vcc:+12v Gnd:Gnd	

Figure 4 Pin connection table

Figure 5 Schematic Diagram

2.2.5. Flowchart

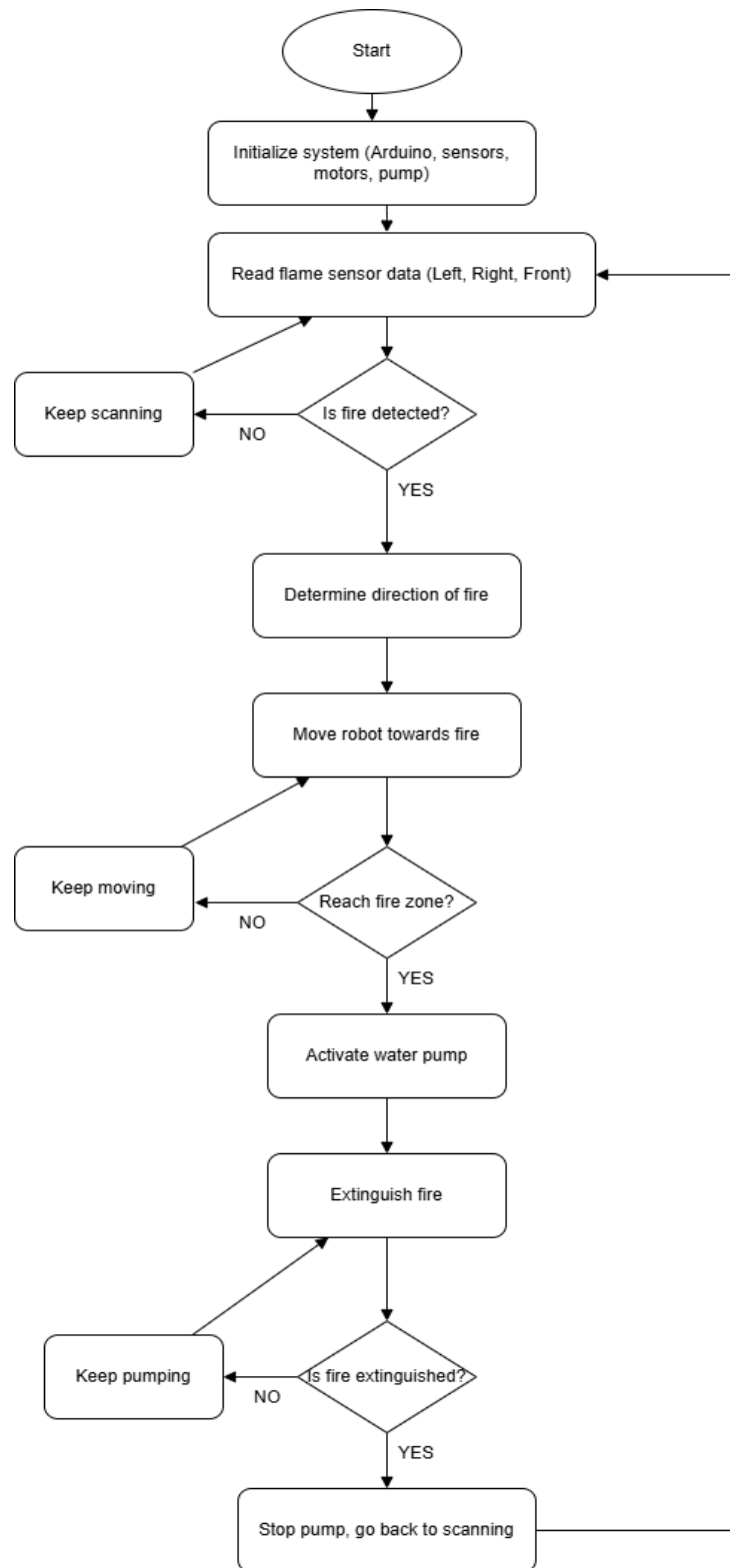


Figure 6 Flowchart

2.3. Requirement Analysis

2.3.1. Hardware Components

In this project, the hardware parts are mainly used to detect signals and perform physical tasks like opening or closing a gate. The main hardware used includes.

IR Flame Sensors

An IR flame sensor is a passive component that detects fire by sensing infrared radiation between 760 nm and 1100 nm, typically emitted by flames. The 4-pin IR Flame Sensor offers A0 for analog flame detection, D0 for digital on/off indication, and GND and VCC for power and ground. (Electroduino, 2025)

In this system, three 4-pin IR flame sensors is used to enable directional flame detection, the sensors are strategically labeled and positioned as right, left, and front.

- Type → Passive
- Function → Input (Infrared Radiation detector)
- Signal Type → Analog
- Quantity → 3

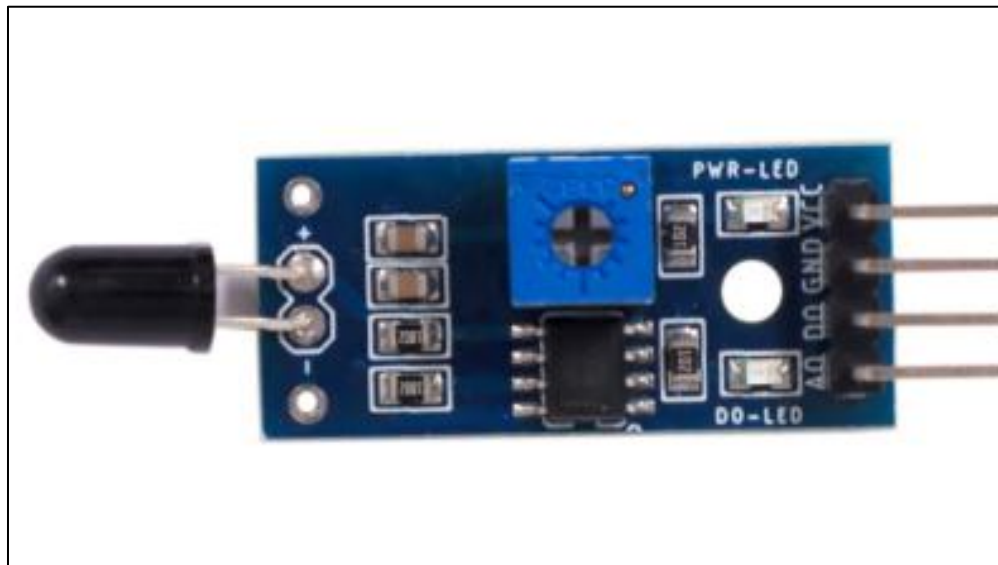


Figure 7 IR Flame sensor

Arduino UNO

An Arduino is an open-source microcontroller board based on the ATmega328P, capable of reading both analog and digital input signals from various sensors and using them to control outputs such as motors, servos, and pumps. It offers 14 digital I/O pins (6 of which provide

PWM output), 6 analog input pins, a USB connection for programming, a power jack, and a reset button.

In this system, the Arduino UNO IS the central processing unit. It receives input from the IR flame sensors, processes this input based on predefined logic and controls the movement of the robot via the motor driver and activates the mini servo and water pump to suppress the detected fire. The Arduino is programmed using the Arduino IDE and coded in C/C++.

Type → Active

Function → Processing and Control Unit

Signal Type → Digital (with analog input capability)

Quantity → 1

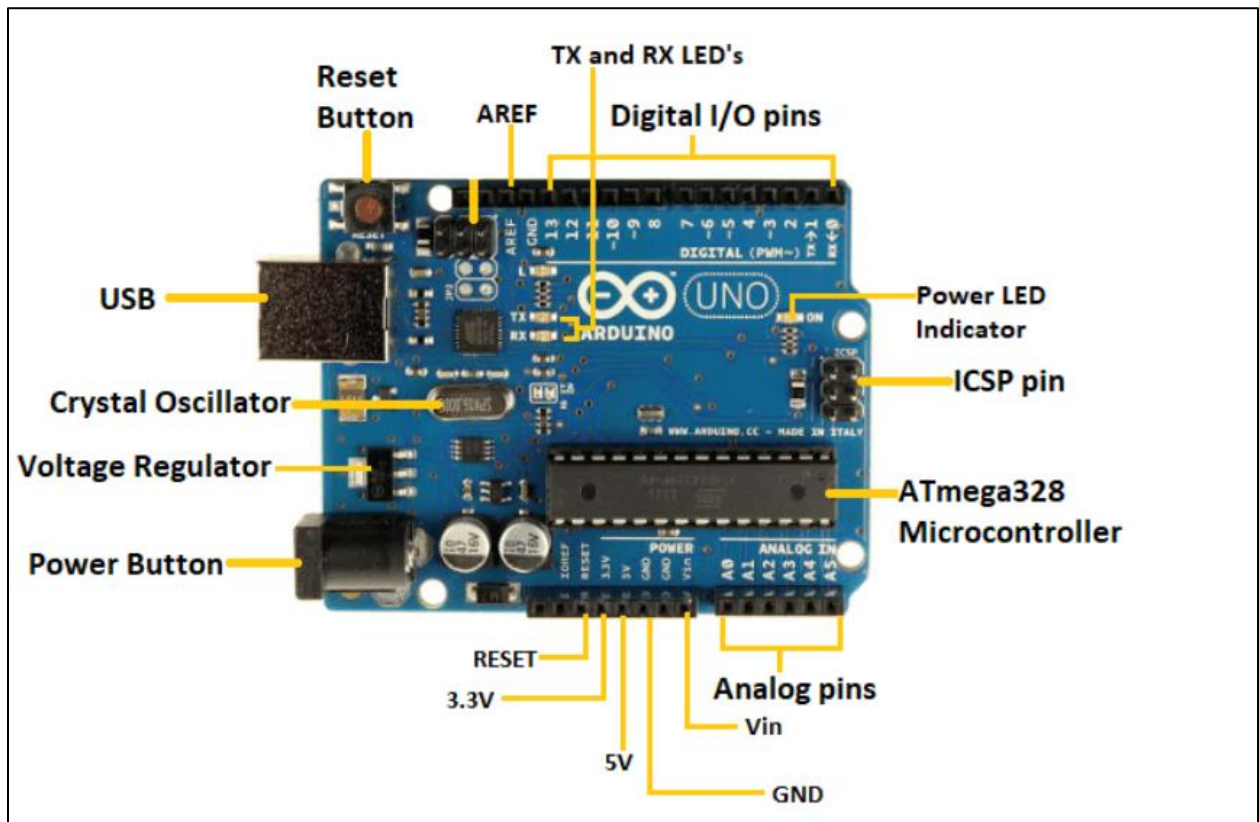


Figure 8 Arduino UNO

BO Motors + Wheels

BO (Battery Operated) motors are lightweight DC gear motors which has low voltage requirements, decent torque, and affordability. These motors typically operate within a voltage

range of 3V to 12V, making them compatible with most microcontroller-based systems like Arduino.

In this project, four BO motors are used each connected to one wheel of the robot in a 4-wheel drive configuration. The motors are controlled through an L298N motor driver, which enables the Arduino to send digital signals to drive the robot forward, backward, left, or right for real-time fire tracking and navigation toward the flame source.

Type → Active

Function → Output (Mechanical Movement)

Signal Type → Digital (PWM Control)

Quantity → 4 motors + 4 wheels



Figure 9 BO motor and wheel

Mini Servo

A mini servo motor is a high precision actuator commonly used to control angular movement. These miniature motor's range from 4.8 to 6 V and three connections power, ground, and signal which is based on Pulse Width Modulation (PWM) signals. (Arduino Docs, 2024)

In this system, a single mini servo motor is used to control the direction of the water nozzle, allowing the robot to aim the water spray toward the detected fire source. The servo is connected to one of Arduino's PWM capable digital pins and receives instructions to adjust its angle based on which sensor detects the flame.

Type → Active

Function → Output (Rotational Actuation)

Signal Type → Digital (PWM Control)

Quantity → 1



Figure 10 Mini Servo

Water Pump

A water pump is a mechanical which operates on a DC voltage (typically 3V–12V) device that is implemented to convey water from one place to another, through controlled electrical signals. (Sempa, 2024)

In this project, a mini submersible water pump is used to spray water through a nozzle mounted on a servo, which directs the spray accurately for extinguishing fires once a flame is detected. The pump is powered by the onboard battery. It is activated by the Arduino through a transistor

or motor driver circuit, allowing the microcontroller to switch the pump ON or OFF based on sensor inputs.

Type → Active

Function → Output (Water Dispensing)

Signal Type → Digital (On/Off Control via Transistor or Motor Driver)

Quantity → 1



Figure 11 Water pump

Battery (Lithium-ion)

Lithium-ion (Li-ion) batteries are rechargeable power sources known for high energy density and consistent voltage output. (Everything PE, 2023)

In this project, two 3.7V Li-ion batteries are used in combination to provide sufficient power supply for all components, including the Arduino, motors, servo, and water pump. When connected in series, these batteries deliver a combined output of 7.4V, which is suitable for powering the Arduino Uno and driving peripheral devices.

Type → Active

Function → Power Supply

Signal Type → DC Voltage

Quantity → 2



Figure 12 Li-ion battery

Breadboard

A solderless construction and testing device for electronic circuits is known as a breadboard. It consists of a grid of interconnected holes where electronic components are held securely and connected without soldering. It allows components to be interconnected using jumper wires without permanent connections. (Soledered, 2024)

In this project, the breadboard serves as the central platform for connecting various modules, including IR flame sensors, BO motors, mini servo, water pump, and the Arduino Uno. It enables a modular and flexible approach to circuit assembly and troubleshooting.

Type → Passive

Function → Circuit Prototyping Platform

Signal Type → Mixed (Analog & Digital)

Quantity → 1

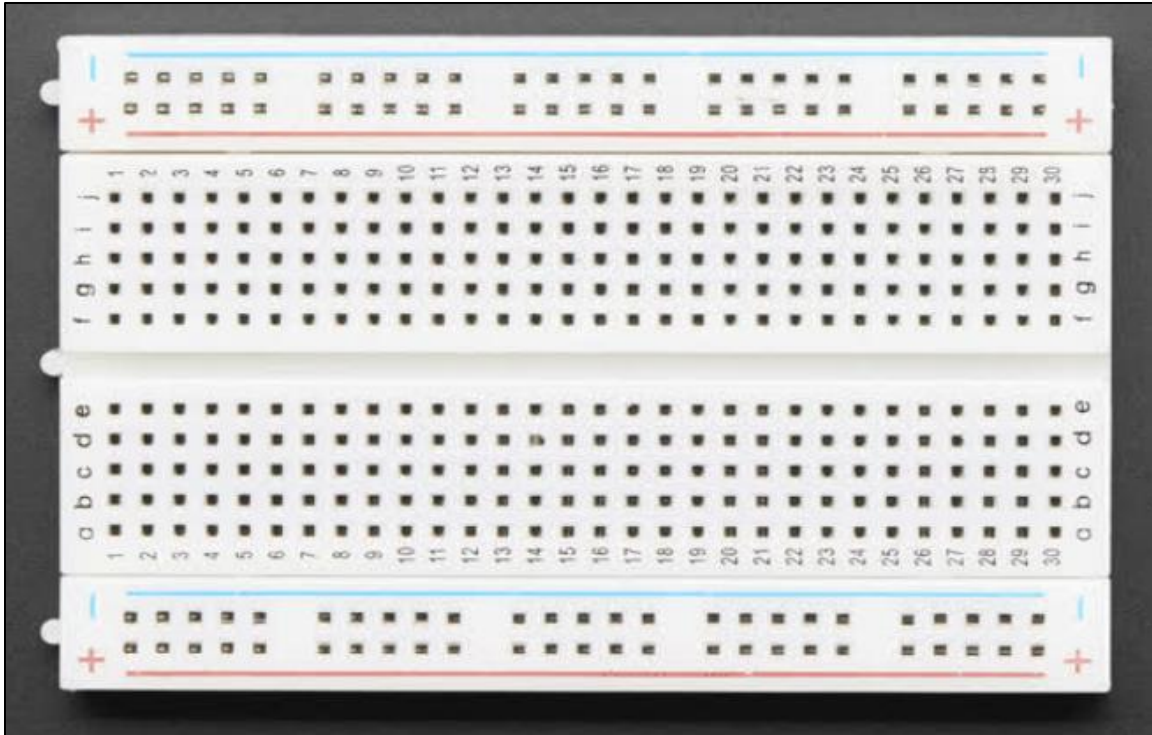


Figure 13 Breadboard

Male-female jumper Wire

Male-to-Female jumper wires are flexible connectors with a male pin on one end and a female socket on the other. They are critical in prototyping, especially when working with development boards and modules that have male or female headers. (Mblasd, 2022)

In this system, male-to-female jumper wires serve several essential purposes:

- Connecting IR flame sensors (which have male header pins) to the Arduino UNO.
- Linking the motor driver module to the Arduino for signal transmission.
- Connecting the Arduino power pins (VIN, 5V and GND) to the breadboard.

Type ➔ Passive

Function ➔ Electrical Interconnection

Signal Type ➔ Analog and Digital

Quantity ➔ Multiple (based on system connections)

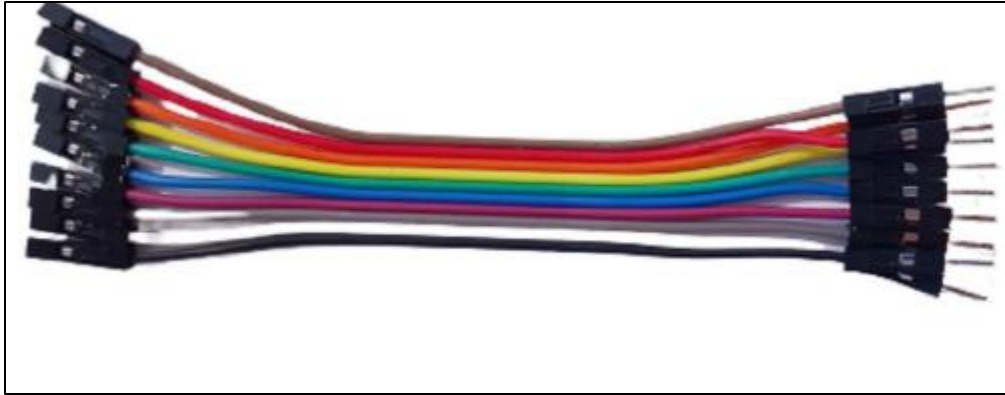


Figure 14 Male-Female jumper wire

Male-Male jumper Wire

A male-to-male jumper wire is a type of electrical wire having connector pins at both ends. It is used primarily for establishing temporary connections between different components on a breadboard or prototype circuitry. (mblasd, 2022)

In this firefighting robot system, male-to-male jumper wires are used in several key roles:

- Bridging breadboard rails, connecting the power rail to other components' rows for efficient power distribution.
- Connecting the servo motor to both the Arduino UNO and breadboard.

Type → Passive

Function → Electrical Interconnection

Signal Type → Analog and Digital

Quantity → Multiple (as per system requirements)

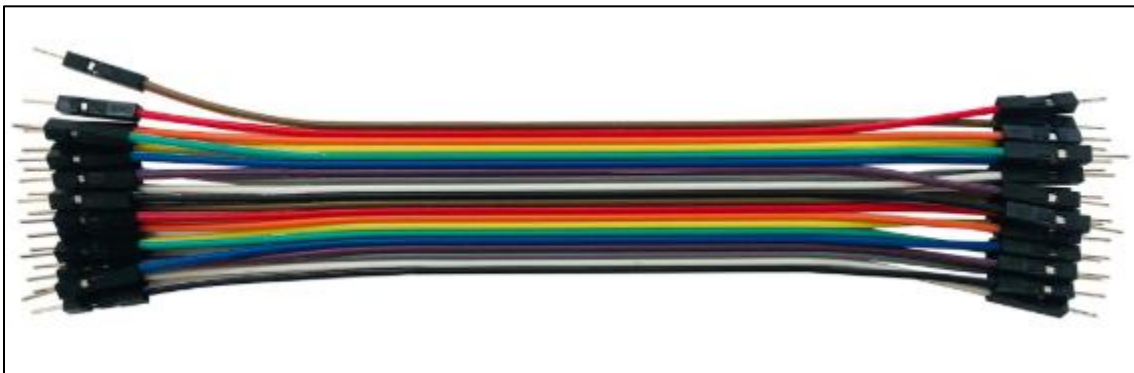


Figure 15 Male to Male jumper wire

Diode

An electronic component that allows current to flow in one direction (forward-biased) and blocks it in the opposite direction (reverse-biased), thus protecting sensitive components from voltage spikes and reverse current. (Jennifer Abella, 2025)

In this firefighting robot system, the **1N4148 diode** is primarily used to protect the water pump circuit from back electromotive force (back-EMF) generated when the motor inside the pump is suddenly switched off. Without this protection, the reverse current could damage the Arduino or the transistor controlling the pump.

Type → Passive

Function → Reverse Current Protection

Signal Type → DC

Quantity → 1 (used with the water pump)

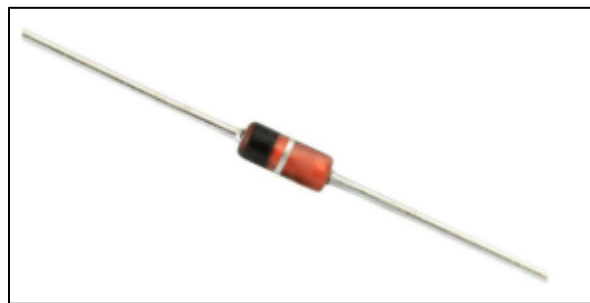


Figure 16 Diode

Transistor

Transistors are three-legged semiconductor devices-emitter, base, and collector-which are used to amplify or switch electronic signals. (Riordan, 2025)

In this firefighting robot system, the **TIP122 transistor** is used to control the water pump, allowing the Arduino to switch the pump ON or OFF using a digital signal. Since the pump requires more current than the Arduino can supply directly, the transistor acts as a switch and current amplifier, powered by an external battery.

Type → Active

Function → Switching and Current Amplification

Signal Type → DC (control signal from Arduino to switch high current load)

Quantity → 1 (used to control the water pump)

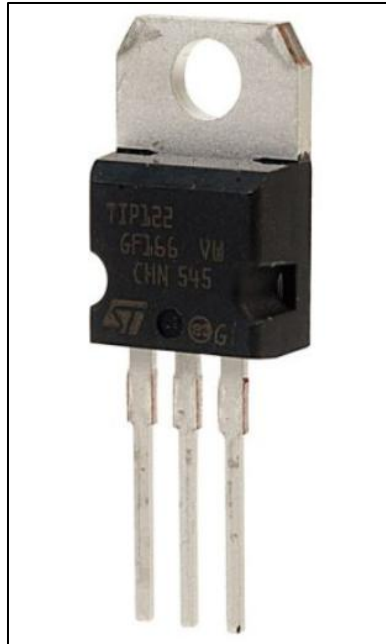


Figure 17 TIP-122 Transistor

Resistor

Resistor is a passive electronic component that limits the flow of electrical current. It is commonly used in circuits to protect sensitive components by reducing voltage or current to safe levels. (ETHW, 2024)

In this firefighting robot system, the **1k Ω** resistor is connected in series with the base of the TIP122 transistor. Its purpose is to limit the current flowing from the Arduino's digital output pin into the transistor's base, preventing damage to both the transistor and the microcontroller.

Type → Passive

Function → Current Limiting

Signal Type → DC

Quantity → 1 (used in transistor base circuit)



Figure 18 1K Ohm Resistor

2.3.2. Software Components

Arduino IDE



Figure 19 Arduino IDE

The Arduino Integrated Development Environment (IDE) is an open-source software platform used to write, compile, and upload code to Arduino-compatible microcontroller boards. (Arduino, 2025)

In this project, the Arduino IDE was used to write and test the logic that controls various hardware components of the fire-fighting robot, such as the IR sensors, servo motor, motor driver, and water pump. The code was developed to allow the robot to detect fire using the IR sensors, navigate toward the fire using the BO motors, and then activate the servo motor and water pump to extinguish the fire.

Microsoft Word



Figure 20 Microsoft Word

Microsoft is widely used processing tool developed by Microsoft. It makes user to create, edit format and shared text-based documents. (Microsoft, 2025)

In this project, we made effective use of Microsoft tools, especially OneDrive and Word, to manage collaboration and documentation. Using OneDrive, our team could work on the same document simultaneously, eliminating the need for constant emailing or version tracking. This allowed for real-time collaboration, edits, and feedback, making our workflow much smoother. Microsoft Word was used to write, format, and finalize this report. Its built-in features such as automatic spell check, referencing tools, image and diagram integration, and table of contents played an essential role in maintaining academic standards and keeping the report structured and professional.

Draw.io



Figure 21 Wraw.io

Draw.io is also known as diagram.net which is a free and open source for online diagramming tool used for creating flowcharts, network diagram and more. (diagram.net, 2025)

For this project it is used to create the system architecture, flowchart, block diagram which represent the components interact and work with firefighting robot.

Cirkit IDE

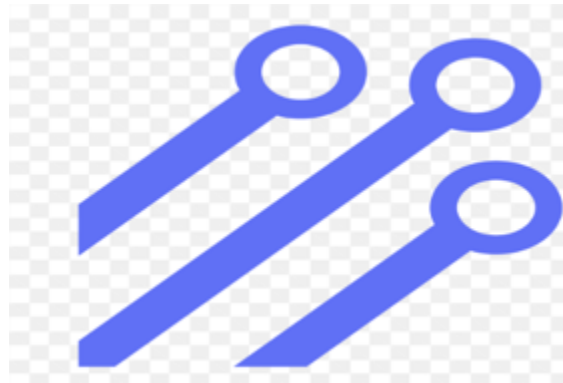


Figure 22 Cirkit IDE

Cirkit IDE is an online platform that helps to create and simulate Arduino projects. It allows to write code, wire together components and run simulations without any installation of new software on their computer. (Cirkit Designer, 2024)

For this project, Cirkit IDE was used to design the circuit diagram of the Automated Fire Chasing and Extinguishing Robot. It allowed the team to virtually place components such as Arduino Uno, flame sensors, servo motor, BO motors, pump, resistors, transistors, and power supply and observe how they interconnect.

Fritzing



Figure 23 Fritzing

Fritzing is an open-source electronics design tool that enables users to document, prototype, and share hardware projects. It provides a user-friendly interface for creating schematic diagrams,

breadboard layouts, and PCB designs, making it ideal for both beginners and professionals in electronics. (Soldered, 2025)

In this project, Fritzing was used to create the schematic diagram of the Automated Fire Chasing and Extinguishing Robot. This tool helped the team to visually represent the electrical wiring and connections among components such as the Arduino Uno, IR sensors, BO motors, mini servo, motor driver, and water pump.

3. Development

Step 1: Design and Planning

First, our team held a meeting where we discussed and shared various project ideas. Although we came from different perspectives, we decided to select a project grounded in real-life relevance and practical application. As students specializing in Networking and IT Security, we are well aware of how vital server rooms are to any organization. These rooms house critical infrastructure, and any fire-related incident can result in severe data loss, hardware damage, or even loss of life.

With this understanding, we chose to develop a project focused on fire prevention and response in server rooms. Our idea is to build an automated, Arduino-based fire extinguishing system. The system is powered by batteries and includes a flame detection sensor, a driving motor, and a breadboard for connectivity. Upon detecting fire or extreme heat, the sensor sends a signal that activates the extinguishing process.

A key feature of our system is its water flow mechanism, which is guided by servo motors. Once a fire is detected, the communication between the sensor, Arduino and the motors directs the flow of water precisely toward the source of the flame. We are using deionized water to avoid any potential damage to electrical components, which makes the solution both safe and effective for use in environments like data centers.

Real incidents inspired our project such as the OVHCloud data center fire in France, the explosion at Google's data center in Iowa, and the September 2024 fire at Alibaba Cloud's data center in Singapore. These events highlight the urgent need for early detection and automatic response systems. Our project aims to address this by offering a cost-effective and responsive solution to reduce the risk and impact of server room fires.

Step 2: Resource Collection

We submitted an application with the guidance of our module leader, Mr. Shishir Subedi Sir, to the college's resource department. The application clearly listed all the components we needed for our project, along with a brief explanation of their roles in the system. After a careful review and approval from Mr. Subedi, the application was forwarded to the department. This process ensured that we were well-prepared and that our request was justified based on the project's technical requirements.

Thanks to the college's support, we received several core components essential to our system. These included four BO motors, a driving motor, three IR flame sensors, an Arduino UNO board, and a breadboard. Additionally, we procured the remaining items ourselves, such as jumper wires, resistors, diodes, transistors, and most importantly, the battery, which serves as the primary power source for the entire setup. The successful collection of all required resources through both institutional assistance and our own initiative enabled us to confidently proceed with the development of our project.

Step 3: System Development

Phase 1: Skeleton and system of the robot

We began our system development by assembling the core mechanical structure of the project. One of our initial architectural design ideas was to build the system in the form of a compact mobile unit, similar to a car model. This design was chosen to ensure that the system could move or be repositioned within a server room environment, allowing flexible fire detection and extinguishing coverage. To bring this idea to life, we started by attaching four BO motors to the four corners of a rectangular wooden base. The wooden platform was selected for its durability and ability to support the overall weight of the system while keeping all the components stable and protected.

Then we gathered the elements to make sure the elements were working, and our prototype would get to the Skeleton we designed.



Figure 24 Skeleton of the robot

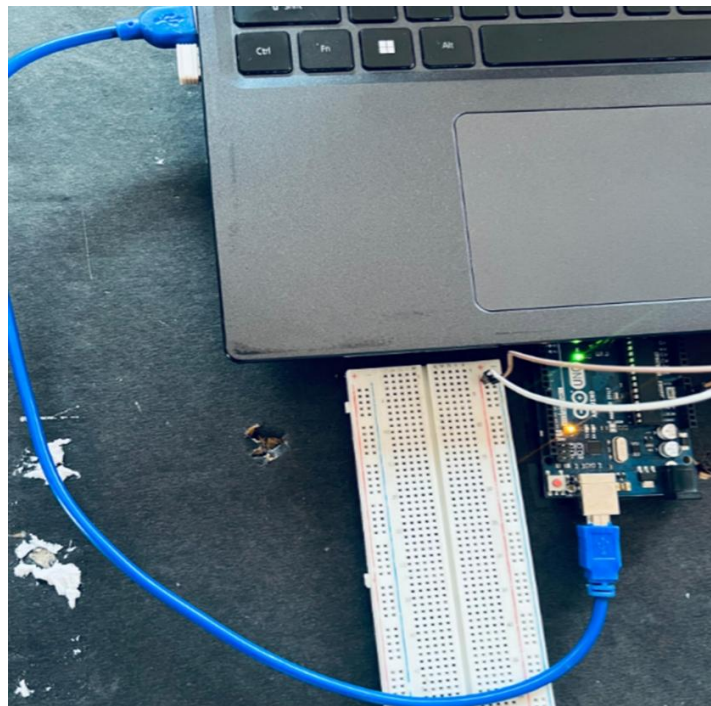


Figure 25 System of robot

Phase 2: Integrating Components (Input)

Once the base and motors were securely attached, we moved on to mounting the main electronic components. The Arduino UNO board and the driving motor were carefully installed on the upper side of the wooden platform to avoid any contact with potential water exposure. We then fixed the battery holder to the bottom side of the platform. This positioning helped in maintaining the weight balance of the system and also ensured that the power supply would remain secure and undisturbed during movement or operation.

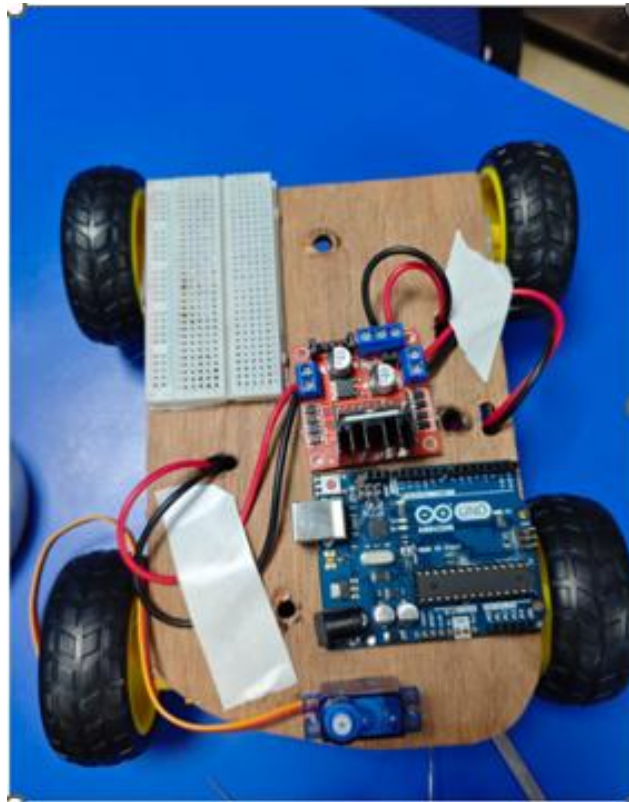


Figure 26 Integrating components

After securing the basic structure, we installed the breadboard next to the Arduino board. The breadboard plays a key role in distributing power to various parts of the system and enabling easy connections without soldering. We then connected three IR flame sensors in strategic positions to ensure maximum fire detection coverage. These sensors are designed to detect infrared radiation from flames, and once triggered, they send signals to the Arduino board for further processing. Proper calibration and testing of these sensors were necessary to avoid false triggers or missed detections.

After we were done setting up, we moved on to the IR flame sensors. We used three IR flame sensors, and each of them had four pins: VCC, GND, A0, and D0. Since we didn't use D0 at all, we just ignored that pin. To make the wiring cleaner and avoid a mess of six separate wires going to the breadboard, we combined the GND pins of all three sensors into one wire and did the same for the VCC pins. So, in the end, only two wires (one VCC and one GND) went from the sensors to the breadboard.

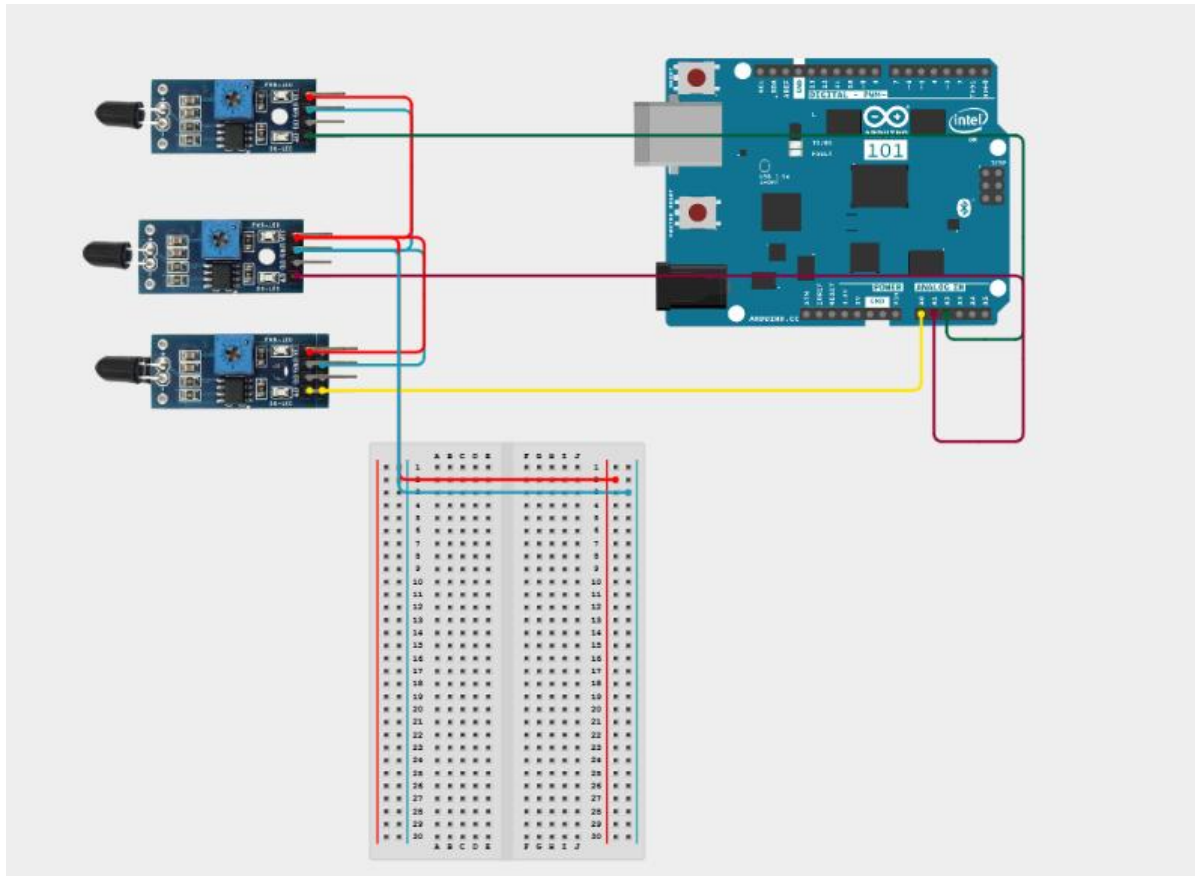


Figure 27 Wiring Flame sensor

Then, we connected the A0 pin of each sensor separately to the Arduino because those are the ones that send the signal when fire is detected. That part was important A0 is where the flame detection signal comes out from, so each A0 had to go into its own analog input pin on the Arduino.

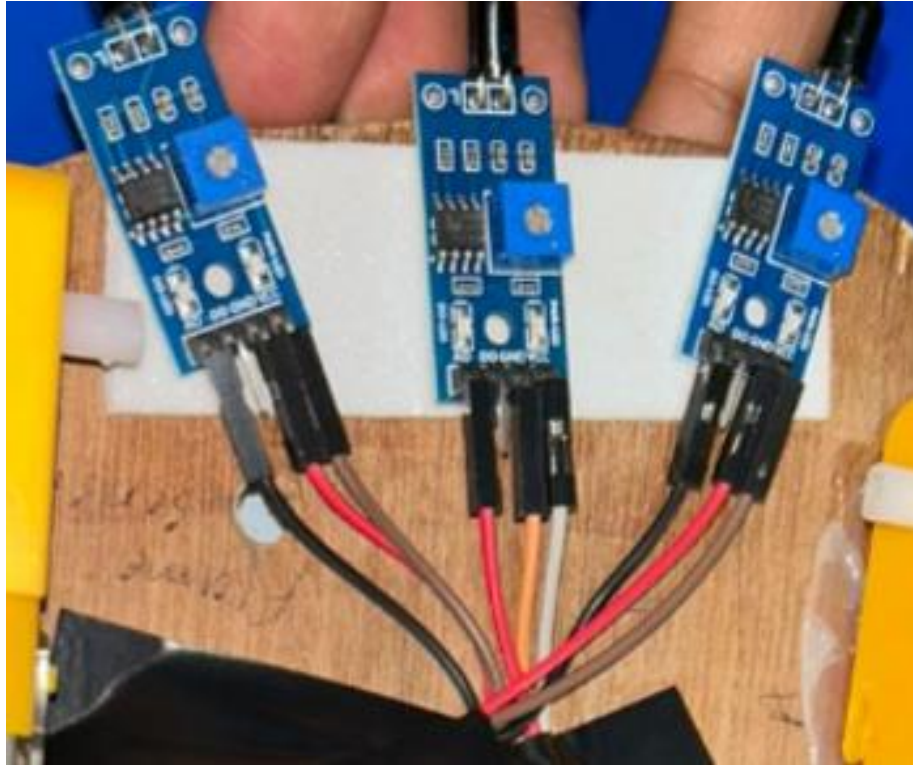


Figure 28 Flame sensor

We made sure the VCC and GND were plugged into the breadboard rails correctly VCC to the positive rail and GND to the negative rail, which were already powered through the connection from the Arduino and motor driver.

Phase 3: Output device and Wiring

Step 1: Connecting the BO Motors

Wiring process was started with the BO motors, which were responsible for the movement of the robot. These motors were first connected to the motor driver module. The motor driver acts like a bridge between the BO motors and the Arduino, allowing us to control direction and speed through Arduino's digital pins.

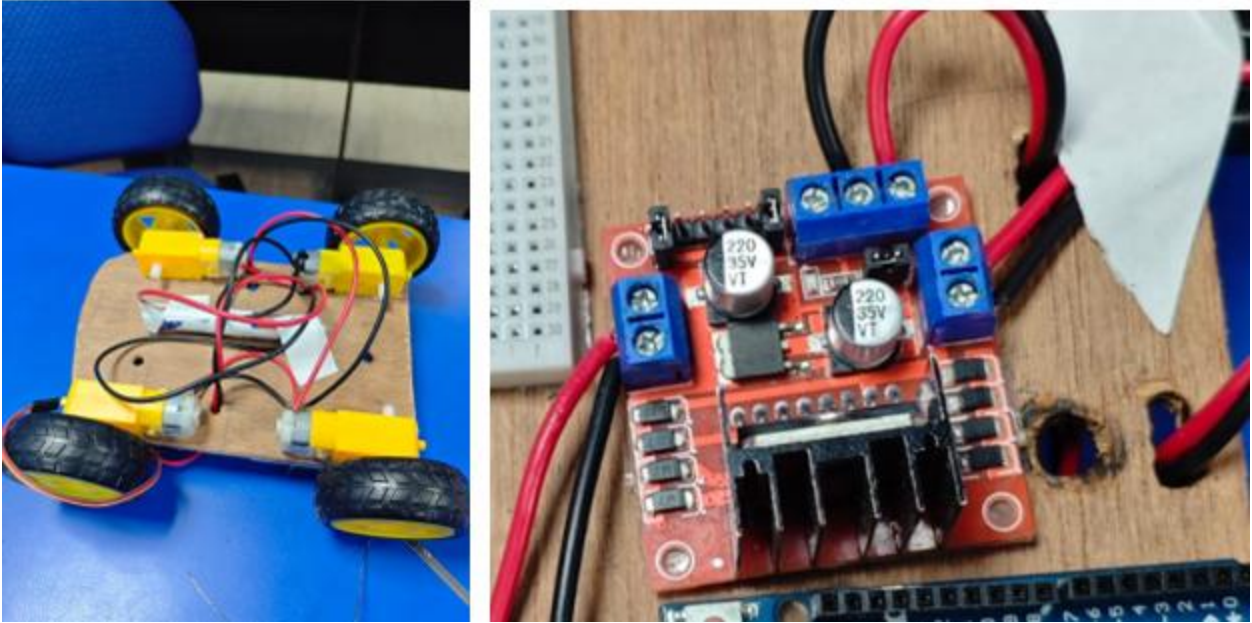


Figure 29 Connecting BO motor to motor driver

Step 2: Linking the Motor Driver to the Arduino

Once the BO motors were in place, we connected the motor driver to the Arduino. The digital pins of the Arduino were connected to the motor driver's input pins. This setup allowed the Arduino to send commands to the motors. We also made sure to connect the motor driver to the power source, which in our case was a battery pack.

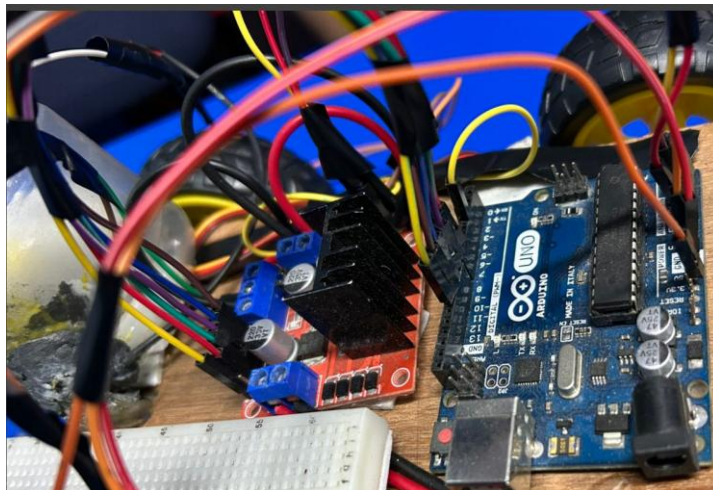


Figure 30 Connecting motor driver to Arduino

Step 3: Placing the Breadboard and Power Distribution

We then started working on the power distribution through the breadboard. Breadboards were mainly used to avoid any soldering and make connections more manageable. We connected the GND (ground) and VIN (voltage in) pins from the Arduino to the breadboard's power rails. This allowed us to distribute power from the Arduino to other components like sensors and the servo.

Step 4: Hooking Up the Servo Motor

The servo motor, which controls the direction of the water spray, was connected next. We wired its signal pin to a PWM pin on the Arduino and connected VCC and GND to the breadboard. The servo needs stable power, so double-checking the connections here was important to prevent shaking or glitchy movement.

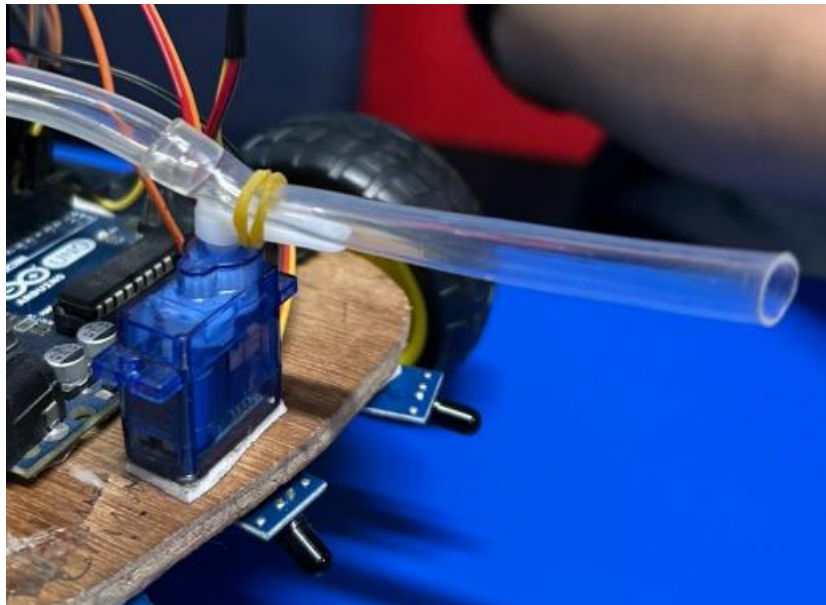


Figure 31 Connecting mini servo

Step 5: Final Battery and Power Checks

Finally, we completed the power circuit by connecting the battery pack to the motor driver module, which then powered both the breadboard and Arduino. This setup allowed all connected components motors, sensors, and the servo to draw power as needed.

After all connections were in place, we triple-checked each wiring point for loose connections or polarity mistakes. Only then did we power up the system and begin testing to make sure everything responded correctly to input.

Phase 4: Coding and Result

After completing the hardware setup, we moved on to the software part. We wrote the necessary code using the Arduino IDE and uploaded it to the Arduino UNO board via USB connection. The code handled all core logic, including sensor input, motor control, and the activation of the water pump and servo motor. Once the code was uploaded, we entered the troubleshooting phase. As expected in any hardware project, we faced several challenges during this stage ranging from motor coordination issues to incorrect sensor responses. However, through trial, error, and teamwork, we were able to debug and fine-tune the system until it functioned as intended.

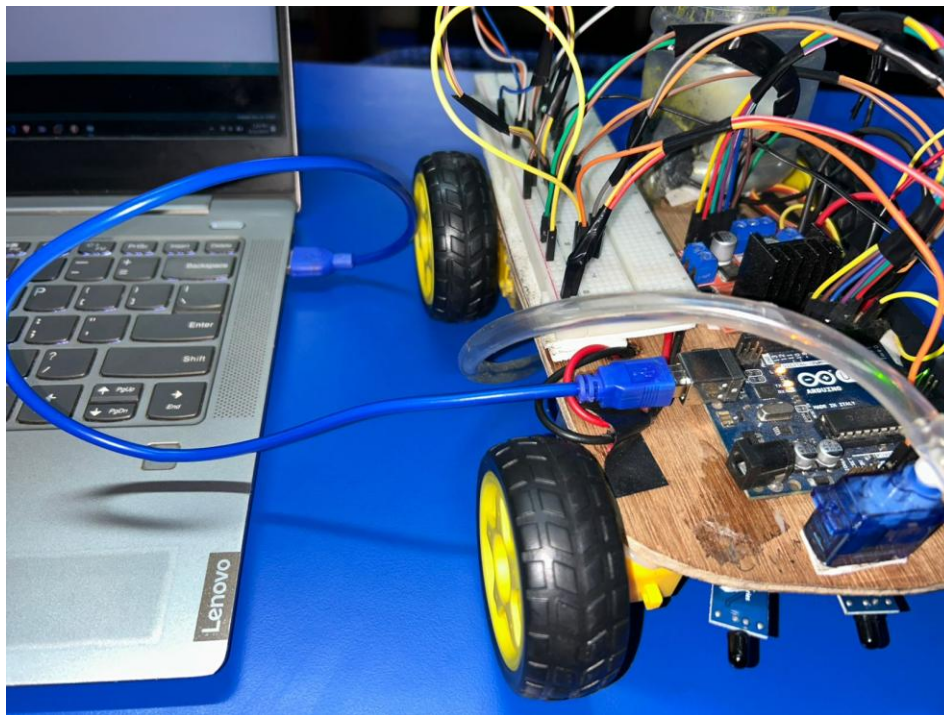


Figure 32 Uploading code through Arduino IDE

Overall, the development process was both challenging and rewarding. It involved a combination of planning, hardware assembly, programming, and continuous testing. Most

importantly, it taught us the value of collaboration, patience, and precision in building a working prototype from scratch.

Step 4: Pins and Connection

This step connects the Arduino, motor driver, sensors, mini servo, and water pump by linking their power, ground, and signal pins to ensure proper coordination. It ensures that each component receives the required voltage and communicates effectively with the Arduino for fire detection and extinguishing.

- Battery GND (-) → Motor driver's GND power pin → Bread board (-) power rail.
- Battery VCC (+) → Motor Driver's +12V → Bread board (+) power rail.
- Arduino UNO GND → bread board (-) power rail
- Arduino UNO VIN → bread board (+) power rail

Motor Driver and Arduino connection

- Motor Driver IN1 → Arduino D9
- Motor Driver IN2 → Arduino D8
- Motor Driver IN3 → Arduino D7
- Motor Driver IN4 → Arduino D6
- ENA A → Arduino D10
- ENA B → Arduino D5
- OUT1/OUT2 → Left BO motor
- OUT3/OUT4 → Right BO motor

Flame sensor connection

- Flame sensor GND → Breadboard power rail (-)
- Flame sensor VCC → Arduino 5v
- Right flame sensor AO → Arduino A0
- Left flame sensor AO → Arduino A2
- Front flame sensor AO → Arduino A1

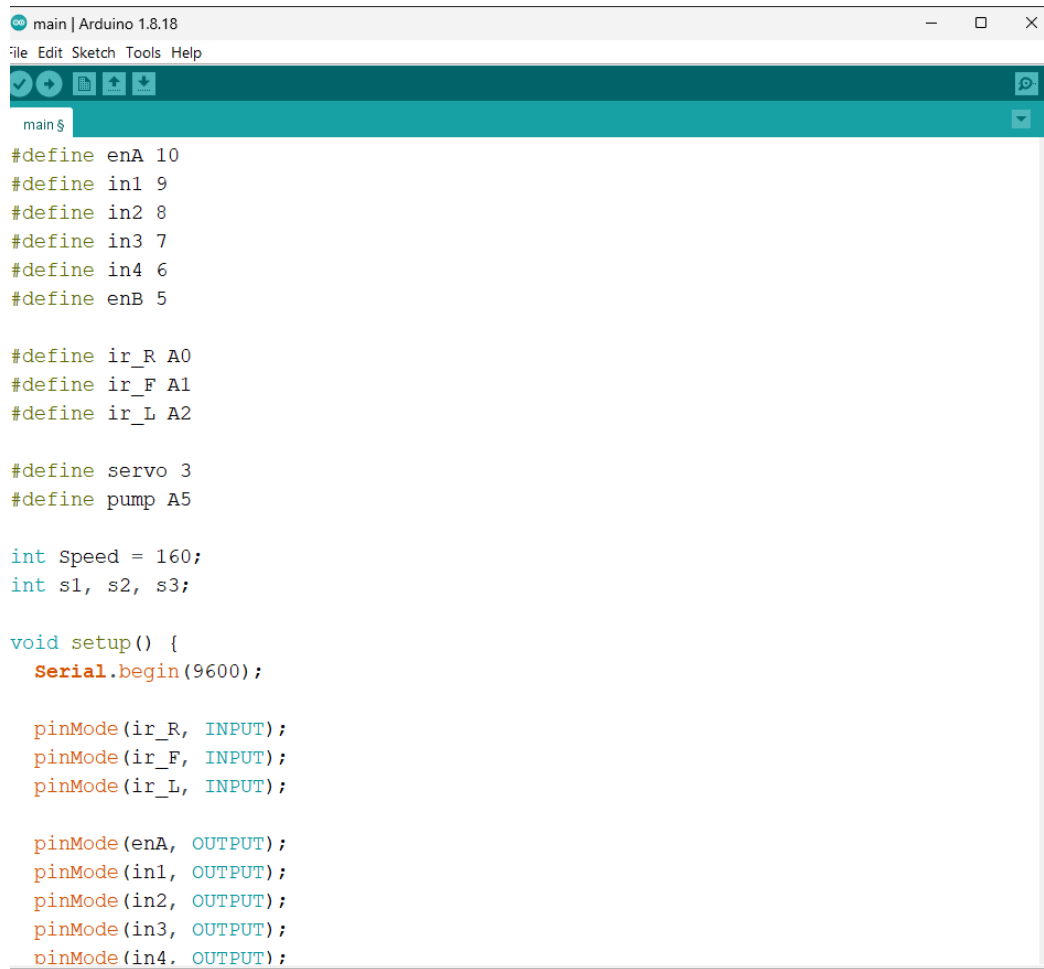
Mini servo

- Mini servo VCC → Breadboard power rail (-)
- Mini servo GND → Breadboard power rail (+)
- Mini servo PWM → Arduino D3

Water pump

- Pump (-) → Transistor Collector
- Pump (+) → Breadboard power rail (+)
- Transistor Emitter → GND
- Diode cathode → pump (+)
- Diode anode → pump (-)
- Transistor base → Resistor → Arduino A5

Step 5: Writing Program



The screenshot shows the Arduino IDE interface with a sketch named 'main'. The code defines several pins and variables, sets up the serial port, and configures the pins for input and output. The code is as follows:

```
main | Arduino 1.8.18
File Edit Sketch Tools Help

main $

#define enA 10
#define in1 9
#define in2 8
#define in3 7
#define in4 6
#define enB 5

#define ir_R A0
#define ir_F A1
#define ir_L A2

#define servo 3
#define pump A5

int Speed = 160;
int s1, s2, s3;

void setup() {
  Serial.begin(9600);

  pinMode(ir_R, INPUT);
  pinMode(ir_F, INPUT);
  pinMode(ir_L, INPUT);

  pinMode(enA, OUTPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);
```

Figure 33 Writing program in Arduino IDE

4. Results and Findings

After uploading our code to the Arduino board, we immediately encountered several issues that required in-depth troubleshooting. One of the first problems we faced was with the servo motor, which wasn't responding at all during the initial testing phase. At first, we assumed the issue might be with the code or wiring, but after careful checking, we discovered that the problem stemmed from the breadboard. We had to replace the breadboard twice, seeking support from our college's resource department.

Eventually, we identified that the real issue was caused by bridging problem on the breadboard. Specifically, the power connection from the (+) and (-) rails weren't properly extended to the central section of the board. This lack of proper bridging led to inconsistent power flow across the system. As a result, we took time to sit down and understand how internal breadboard rails work and how bridging is supposed to be done for full functionality. This learning process was crucial and allowed us to better handle similar issues in the future.

Another challenge was with the water pump, which failed to activate despite the motor and code appearing correct at first glance. We went through our entire Arduino code line by line, checking for logic errors, pin mismatches, or power issues. It was a tedious and frustrating process, but by working as a team and staying patient, we were finally able to resolve the issue. Once all systems were responding properly including the BO motor, flame sensor, servo motor, and water pump, we moved forward to the system testing phase, confident in the stability of our build.

4.1. Testing

Test 1

Test	1
Objective	To evaluate whether the robot is capable of detecting and navigating towards flame source.
Activity	A candle was lit in front of the robot and the system was allowed to run to observe flame detection and movement.
Expected Result	The robot should detect the flame and navigate towards it for fire suppression.
Actual Result	The robot successfully detected the flame and moved in its directions.
Conclusion	The test was successful. The robot responded appropriately to the presence of fire.

Table 1 Test 1

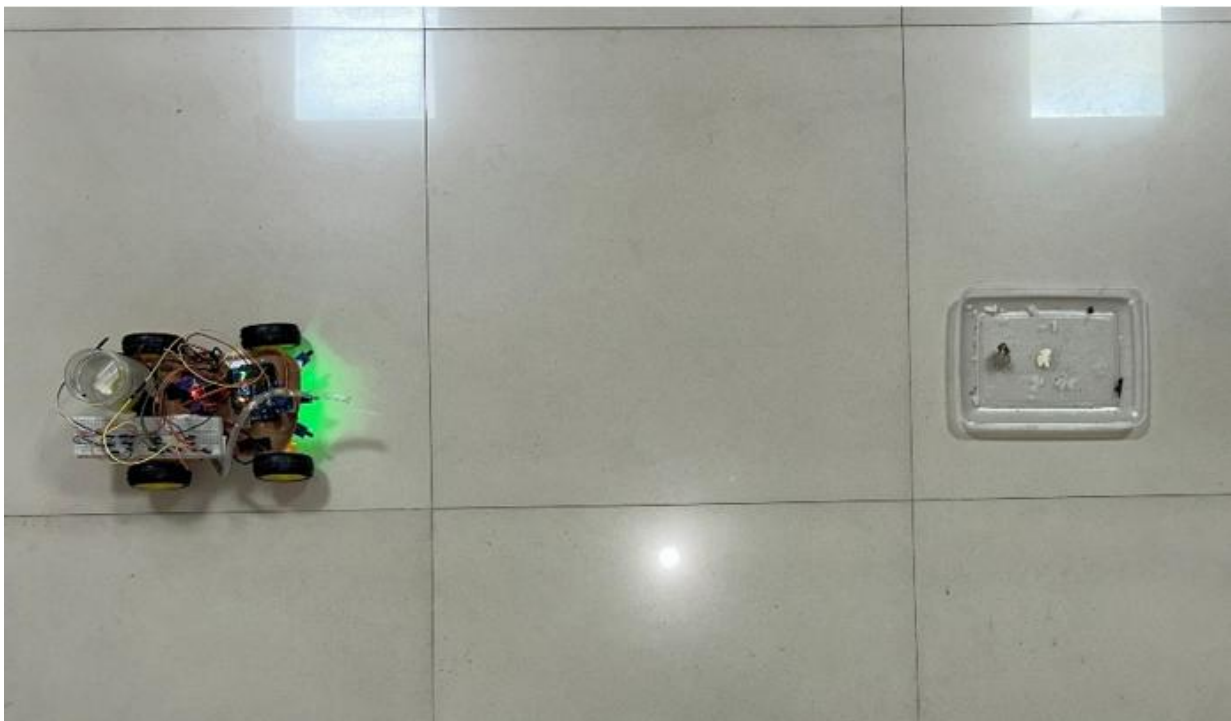


Figure 34 Position before igniting fire

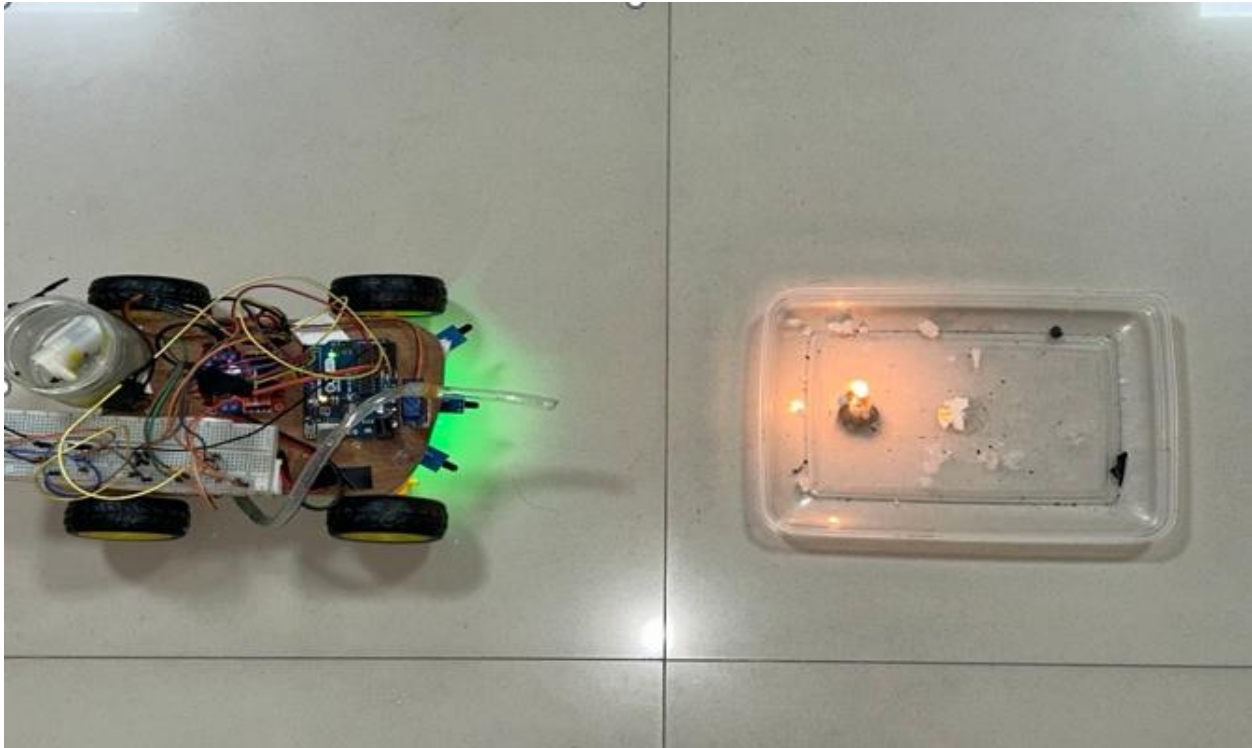


Figure 35 Position of robot after sensing fire.

Test 2

Test	2
Objective	To verify whether the water pump activates correctly after flame detection.
Activity	The candle was lit again, and the robot was observed to check if the pump activates after detection
Expected Result	The robot detects the flame and activates the water pump to extinguish it.
Actual Result	The robot detected the fire, and the water pump was activated as expected.
Conclusion	Test 2 was successful. The water pump functioned properly in response to the flame.

Table 2 Test 2

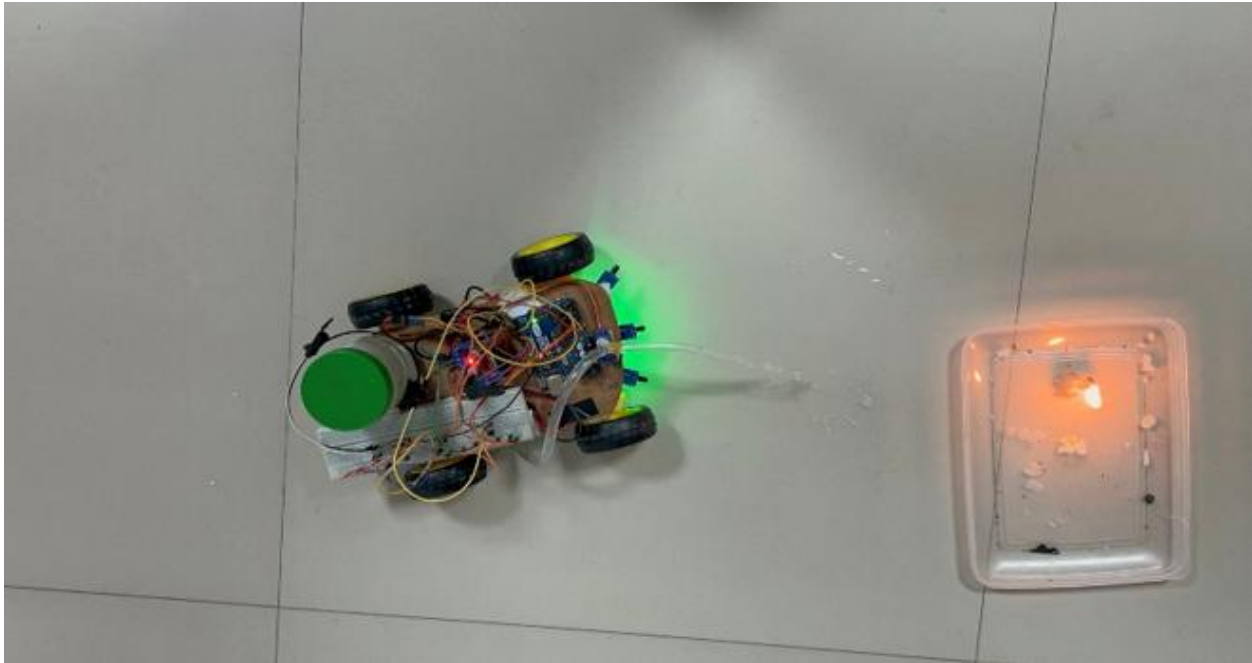


Figure 36 Water pump was activated

Test 3

Test	3
Objective	To observe the sensor performance and detection accuracy under bright sunlight.
Activity	The robot was tested outdoors on a sunny day with a lit candle as the flame source.
Expected Result	The robot will detect the flame and operate normally despite the sunlight.
Actual Result	The robot failed to consistently detect the flame, showing confusion due to ambient light interference.
Conclusion	Test 3 was unsuccessful. The sensors struggled in bright lighting conditions and may require improvement or shielding...

Table 3 Test 3



Figure 37 Sensors struggled to sense fire in bright condition

Test 4

Test	4
Objective	To check whether the servo motor and water pump shut off automatically once the fire is extinguished.
Activity	The system was allowed to detect and suppress the flame, then observed to see if components stopped functioning after extinguishment.
Expected Result	Both the servo motor and water pump should stop operating after the fire is out.
Actual Result	The components turned off as expected once the fire was extinguished.
Conclusion	Test 4 was successfully completed. Auto shutdown functionality works as intended....

Table 4 Test 4

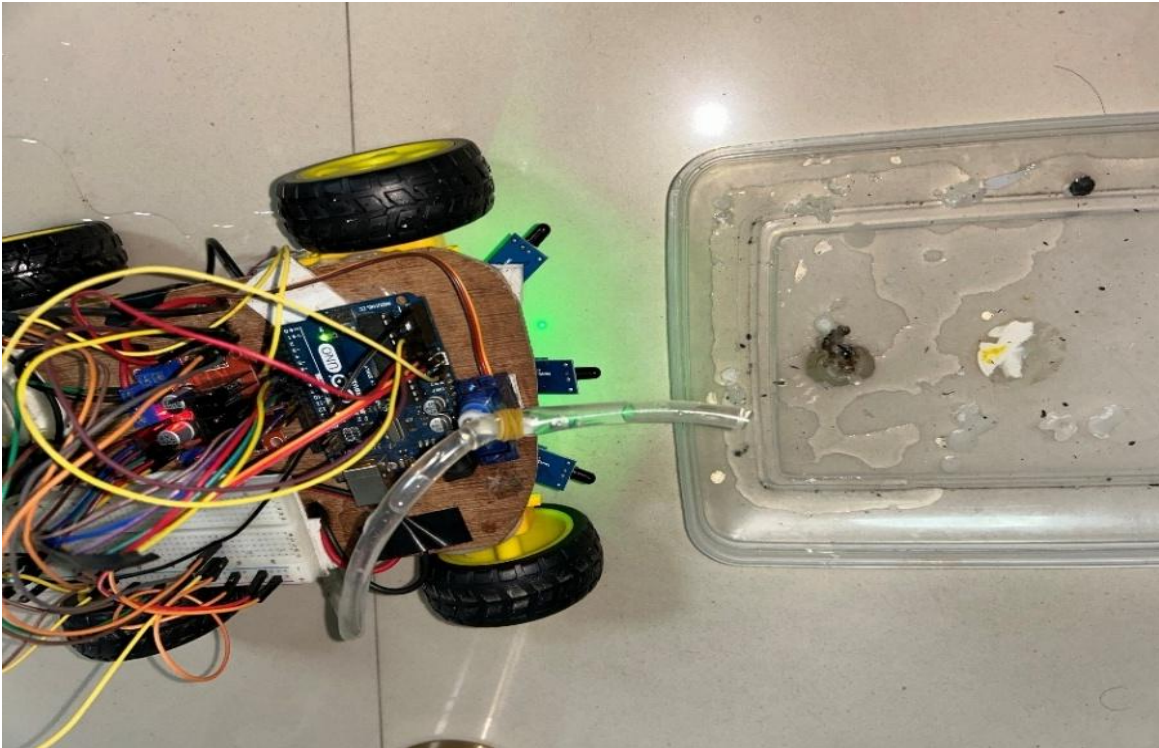


Figure 38 Water pump shut off after extinguishing fire

Test 5

Test	5
Objective	To evaluate the robot's ability to navigate around obstacles while detecting fire.
Activity	A physical barrier was placed between the robot and the flame to test navigation response.
Expected Result	The robot should identify an alternate path to reach the flame.
Actual Result	The robot was unable to effectively navigate around the obstacle.
Conclusion	Test 5 was unsuccessful, showed a limitation in obstacle handling. Further development is needed in this area.

Table 5 Test 5



Figure 39 Robot not navigating around obstacle.

5. Future works

While developing the project, we encountered a few issues particularly in the tests when the robot wasn't as efficient as desired. For instance, in sunlight or when an object is placed between the fire and the robot, detection was either late or missed. One of the key areas where we will refine our project is sensor sensitivity as well as the response of the robot in intricate surroundings. Rather than depending on IR flame detectors exclusively, in the future, we can integrate multiple types of sensors like incorporating smoke or temperature sensors to improve accuracy.

One other issue was handling obstacles. When we put something in its way, it couldn't navigate through the fire. This could be fixed with the use of ultrasonic or proximity sensors that aid in the detection of obstacles and change the course of movement accordingly. Additionally, employing a superior chassis or structuring the body in a more stable configuration could enhance balance as well as mobility, if we are planning on deploying this in real environments such as server rooms or small spaces.

We're also thinking of making the water system stronger. Right now, the water pressure and flow are just enough, but if the fire is larger or the angle is off, it might not work well. So, we could use a better water pump and maybe redesign the servo motor angle for better purposes.

Finally, we hope to make the system smarter by including IoT features like app control or real-time notifications. That way, the robot can send alerts, show its location, or even be controlled remotely if needed. These are ideas we really want to work on as we move forward.

6. Conclusion

Throughout this project, our focus was to build a practical solution for a real-world problem: fire hazards in server rooms. Our autonomous fire-fighting robot was developed with the goal of detecting and extinguishing flames as quickly and safely as possible without human intervention. Using an Arduino microcontroller, IR flame sensors, servo-controlled nozzles, and a compact water pump system, we created a working prototype capable of identifying fire sources, navigating toward them, and suppressing flames in the early stages.

During development and testing, we gained valuable experience in circuit design, hardware troubleshooting, embedded programming, and real-time problem-solving. The robot was tested in multiple rounds under different lighting and environmental conditions. While most tests showed successful detection and response, issues like interference from sunlight, limited obstacle avoidance, and occasional misalignment of the water spray were observed.

These limitations helped us identify key areas for future improvements. The robot can be enhanced with better obstacle detection (such as ultrasonic sensors), improved light filtering for flame sensors, and more precise actuation for targeting the fire. Additionally, integrating a basic camera module or thermal sensor could allow better flame localization and decision-making. A stronger chassis and enclosed wiring would also improve its durability for real deployment. Moreover, using GSM or Wi-Fi modules to send alerts or status reports in real-time could make it more effective in remote server environments.

Looking ahead, we aim to upgrade the prototype into a more advanced version suitable for real-world deployment in data centers, labs, and critical infrastructure sites. Each team member has played a vital role—from sensor integration and coding to frame building and documentation—and this project has strengthened our understanding of automation in emergency response and teamwork in technical problem-solving. We now have a solid foundation to develop this into a more reliable, scalable, and life-saving tool.

7. References

- Arduino Docs. (2024). *Arduino Docs*. Retrieved May 03, 2025, from <https://docs.arduino.cc/learn/electronics/servo-motors/>
- Circuit Designer. (2024). *Circuit Designer*. Retrieved May 03, 2025, from <https://docs.circuitdesigner.com/component/82493423-36f6-44d1-933e-c3d087cac576/motor-and-wheels>
- diagram.net. (2025, April 28). *Draw.io*. Retrieved from diagram.net: <https://www.drawio.com/>
- Electroduino. (2025). *ElectroDuino*. Retrieved april 22, 2025, from <https://www.electroduino.com/ir-infrared-flame-sensor-module/#:~:text=A%20Flame%20Sensor%20module%20or,fire%20flame%20or%20light%20source.>
- ETHW. (2024). *ETHW*. Retrieved May 03, 2025, from https://ethw.org/Resistor?gad_source=1&gad_campaignid=22141898068&gbraid=0AAA AADOXuoS8CHO_LtnYd35ocwe72rAPM&gclid=Cj0KCQjw_dbABhC5ARIsAAh2Z-QDhZe7TPpVV9jAbGkgQsaFi41UJBqTGt9qFycO4IUZRh7wFosZGewaAtP0EALw_wcB
- Everything PE. (2023). *Everything PE*. Retrieved May 03, 2025, from https://www.everythingpe.com/community/what-is-a-lithium-ion-battery?gad_source=1&gad_campaignid=22451877011&gbraid=0AAAAAodorec6neuYw8mArleyXAsy_kx94&gclid=Cj0KCQjw_dbABhC5ARIsAAh2Z-Qe4mq1qTe1HHjCZmIJsyilmL2xioZYxWfx5kBdE6sL6HPXK8tm9MaAj6DEALw_wcB
- Himalayan News Service. (2019). *Property worth millions destroyed in Subisu fire*. Kathmandu: Himalayan News Service.
- Impact Fire. (2025). *Impact Fire*. Retrieved april 29, 2025, from <https://resources.impactfireservices.com/which-fire-protection-system-is-best-for-server-rooms-and-data-centers>

- Jennifer Abella, S. A. (2025). *Britannica*. Retrieved May 03, 2025, from <https://www.britannica.com/technology/diode>
- mblasd. (2022). *WordPress.com*. Retrieved May 03, 2025, from <https://electronicssecret.wordpress.com/2022/07/14/jumper-wires-an-introduction-applications-and-working-principles/>
- Mblasd. (2022). *WordPress.com*. Retrieved may 03, 2025, from <https://electronicssecret.wordpress.com/2022/07/14/jumper-wires-an-introduction-applications-and-working-principles/>
- Microsoft. (2025, April 28). *Microsoft Word*. Retrieved from Microsoft: <https://www.microsoft.com/en-us/microsoft-365/word>
- Riordan, M. (2025). *Britannica*. Retrieved May 03, 2025, from <https://www.britannica.com/technology/transistor>
- Sempa. (2024). *SEMPA*. Retrieved May 03, 2025, from <https://sempapompa.com/en/blog-detay/yaningla-mucadele-sistemlerinde-su-pompalarının-rolü#:~:text=Jockey%20Pumps%3A%20Jockey%20pumps%20are,pressure%20at%20the%20desired%20level.>
- Sensor, R. (2021). *Rika Sensor*. Retrieved april 29, 2025, from <https://www.rikasensor.com/fire-incidents-occur-frequently-in-the-computer-room-is-the-computer-room-environmental-monitoring-system-useful.html>
- Soledered. (2024). *Soledered*. Retrieved may 03, 2025, from https://soldered.com/learn/what-is-a-breadboard-and-how-to-use-it/?srsltid=AfmBOopRcHBD4hsRh1ZVr2LFFKqiwXIBQK-Yy1Kkoj4lI3ggmveoOpxf_
- Zhang, M. (2023). *Dgtl Infra*. Retrieved april 30, 2025, from <https://dgtlinfra.com/data-center-fires/>

8. Appendix

8.1. Source code

```
#define enA 10
```

```
#define in1 9
```

```
#define in2 8
```

```
#define in3 7
```

```
#define in4 6
```

```
#define enB 5
```

```
#define ir_R A0
```

```
#define ir_F A1
```

```
#define ir_L A2
```

```
#define servo 3
```

```
#define pump A5
```

```
int Speed = 160;
```

```
int s1, s2, s3;
```

```
void setup() {
```

```
    Serial.begin(9600);
```

```
pinMode(ir_R, INPUT);
```

```
pinMode(ir_F, INPUT);
```

```
pinMode(ir_L, INPUT);
```

```
pinMode(enA, OUTPUT);
```

```
pinMode(in1, OUTPUT);
```

```
pinMode(in2, OUTPUT);
```

```
pinMode(in3, OUTPUT);
```

```
pinMode(in4, OUTPUT);
```

```
pinMode(enB, OUTPUT);
```

```
pinMode(servo, OUTPUT);
```

```
pinMode(pump, OUTPUT);
```

```
for (int angle = 90; angle <= 140; angle += 5) {
```

```
    servoPulse(servo, angle);
```

```
}
```

```
for (int angle = 140; angle >= 40; angle -= 5) {
```

```
    servoPulse(servo, angle);
```

```
}
```

```
for (int angle = 40; angle <= 95; angle += 5) {
```

```
    servoPulse(servo, angle);
```

```
}
```

```
    analogWrite(enA, Speed);  
    analogWrite(enB, Speed);  
    delay(500);  
}
```

```
void loop() {  
  
    s1 = analogRead(ir_R);  
    s2 = analogRead(ir_F);  
    s3 = analogRead(ir_L);  
  
    Serial.print(s1);  
    Serial.print("\t");  
    Serial.print(s2);  
    Serial.print("\t");  
    Serial.println(s3);  
    delay(50);  
  
    if (s1 < 250) {  
        Stop();  
        digitalWrite(pump, 1);  
        for (int angle = 90; angle >= 40; angle -= 3) {  
            servoPulse(servo, angle);  
        }  
    }  
}
```

```
    }  
  
    for (int angle = 40; angle <= 90; angle += 3) {  
  
        servoPulse(servo, angle);  
  
    }  
}  
  
else if (s2 < 350) {  
  
    Stop();  
  
    digitalWrite(pump, 1);  
  
    for (int angle = 90; angle <= 140; angle += 3) {  
  
        servoPulse(servo, angle);  
  
    }  
  
    for (int angle = 140; angle >= 40; angle -= 3) {  
  
        servoPulse(servo, angle);  
  
    }  
  
    for (int angle = 40; angle <= 90; angle += 3) {  
  
        servoPulse(servo, angle);  
  
    }  
}  
  
else if (s3 < 250) {  
  
    Stop();  
  
    digitalWrite(pump, 1);  
  
    for (int angle = 90; angle <= 140; angle += 3) {  
  
        servoPulse(servo, angle);
```

```
    }  
  
    for (int angle = 140; angle >= 90; angle -= 3) {  
  
        servoPulse(servo, angle);  
  
    }  
  
}  
  
else if (s1 >= 150 && s1 <= 700) {  
  
    digitalWrite(pump, 0);  
  
    backward();  
  
    delay(100);  
  
    turnRight();  
  
    delay(200);  
  
}  
  
else if (s2 >= 150 && s2 <= 800) {  
  
    digitalWrite(pump, 0);  
  
    forward();  
  
}  
  
else if (s3 >= 150 && s3 <= 700) {  
  
    digitalWrite(pump, 0);  
  
    backward();  
  
    delay(100);  
  
    turnLeft();  
  
    delay(200);  
  
}
```

```
else {  
    digitalWrite(pump, 0);  
    Stop();  
}
```

```
delay(10);  
}
```

```
void servoPulse(int pin, int angle) {  
    int pwm = (angle * 11) + 500;  
    digitalWrite(pin, HIGH);  
    delayMicroseconds(pwm);  
    digitalWrite(pin, LOW);  
    delay(20);  
}
```

```
void forward() {  
    digitalWrite(in1, LOW);  
    digitalWrite(in2, HIGH);  
    digitalWrite(in3, HIGH);  
    digitalWrite(in4, LOW);  
}
```

```
void backward() {  
    digitalWrite(in1, HIGH);
```



```
    digitalWrite(in2, LOW);  
    digitalWrite(in3, LOW);  
    digitalWrite(in4, HIGH);  
}  
  
void turnLeft() {  
    digitalWrite(in1, LOW);  
    digitalWrite(in2, HIGH);  
    digitalWrite(in3, LOW);  
    digitalWrite(in4, HIGH);  
}  
  
void turnRight() {  
    digitalWrite(in1, HIGH);  
    digitalWrite(in2, LOW);  
    digitalWrite(in3, HIGH);  
    digitalWrite(in4, LOW);  
}  
  
void Stop() {  
    digitalWrite(in1, LOW);  
    digitalWrite(in2, LOW);  
    digitalWrite(in3, LOW);  
    digitalWrite(in4, LOW);  
}
```

8.2. Individual Contribution Table

Student Name	Role	Contribution
Sulav Bhattarai	<ul style="list-style-type: none"> • System architecture and Flowchart • Proposal • Development • Device Integration • Coding • Documentation • Appendix • Hardware Components 	20%
Aarzoo Pandey	<ul style="list-style-type: none"> • System Overview and Flowchart • Development • Device Integration • Future works • Problem statement and Project as Solution • Proposal • Result and Findings • Conclusion 	20%
Sittal Karki	<ul style="list-style-type: none"> • Introduction • Overall help in design • Resource collection of projects • Software components • Aim and Objectives • Device integration 	20%
Anupam Shrestha	<ul style="list-style-type: none"> • Circuit diagram • Schematic diagram • Device Integration • Block diagram • Testing • Conclusion 	20%
Tulasa Giri	<ul style="list-style-type: none"> • Current scenario • Problem statement and Project as solution • Application Submission • Acknowledgement and Abstract • Hardware components • Testing 	20%

8.3. Individual Contribution Structure

