FIRE FIGHTING ROBOT USING ARDUINO UNO

COMPUTATIONAL PHYSICS

A PROJECT REPORT

Submitted by

CHAITANYA VARMA	**-**	**CB.SC.U4AIE24017**
HARSHITH REDDY	**-**	**CB.SC.U4AIE24018**
JISHNU TEJA DANDAMUDI	**-**	**CB.SC.U4AIE24019**
JIVITES DAMODAR	**-**	**CB.SC.U4AIE24020**

CENTER FOR COMPUTATIONAL ENGINEERING AND NETWORKING

AMRITA SCHOOL OF ARTIFICIAL INTELLIGENCE

COIMBATORE-641112

2024

AMRITA SCHOOL OF ARTIFICIAL INTELLIGENCE
AMRITA VISHWA VIDYAPEETHAM
COIMBATORE-641112

DECLARATION

We, **Chaitanya Varma – CB.SC.U4AIE24017**, **Harshith Reddy - CB.SC.U4AIE24018**, **Jishnu Teja Dandamudi - CB.SC.U4AIE24019** and **Jivites Damodar - CB.SC.U4AIE24020**, hereby declare that the project work entitled "**FIRE FIGTHING ROBOT USING ARDUINO UNO**" is the record of the original work done by us under the guidance of **Dr. Jithin E.V**, Assistant Professor, Department of Artificial Intelligence, Amrita Vishwa Vidyapeetham, Coimbatore.

Chaitanya Varma - CB.SC.U4AIE24017 **Harshith Reddy - CB.SC.U4AIE24018**

Jishnu Teja Dandamudi - CB.SC.U4AIE24019 **Jivites Damodar - CB.SC.U4AIE24020**

Place: Coimbatore – 641112
Date:

COUNTERSIGNED

Dr. Jithin E.V,
Assistant Professor,
Department of Artificial Intelligence

TABLE OF CONTENTS

ABSTRACT

A basic design of robot that can fight fires at an affordable cost could prove to be boon in fighting domestic fires, till help arrives. The robot developed consists of three elements which is the hardware, electronic interfacing circuits, and software program. This firefighting robotic system is capable of detecting and extinguishing fire. These robots can be made to roll into places where it is not safe for humans to enter. Time is of essence when it comes to fighting fires as even a few minutes' delay can turn small fires into raging inferno. This robot is designed as a first response unit so it can suppress the fire keeps it under control till help arrives. This firefighting robotic system is controlled by an Arduino Uno development board. It is also equipped with the fire flame sensor for detecting fires. It is equipped with a water tank and a pump. So, on detecting fires it sprays water extinguishing the fire. Water spraying nozzle is mounted on servo motor to cover maximum area. Although there is a lot of scope for improvement, this could be a first step in developing a complete fire-fighting robot that could also rescue victims. The main function of this robot is to become an unmanned support vehicle, developed to search and extinguish fire. By using such robots, fire identification and rescue activities can be done with greater accuracy and securely without exposing the fire fighters to dangerous conditions. In other words, robots can reduce the need to expose fire fighters to danger. The problem definition for a firefighting robot is to develop a robot which accurately tries to detect a flame or fire nearby and move to that place if an only if there is no obstacle for it to move. The robot must also be able to detect multiple fires at the same time. There is no controller for the robot. The robot moves as fast as it can move to the nearby detected fire and cuts off the fire. The objective of this project is to design and develop an autonomous firefighting robot capable of detecting, locating, and extinguishing fires in various environments. The robot should operate in a range of settings, including indoor spaces, outdoor areas, and potentially hazardous environments that are difficult for human firefighters to navigate.

INTRODUCTION

India recorded 1.6million fire accidents and 27,027 accidental fire deaths in the year 2021. The most common cause of house fires which also accounts for 42% of accidents is cooking [1]. Open flames from stoves can result in accidents when left unsupervised even for short duration of time. Annually 60 -70 Fire fighters die on duty, which is very high. Fighting various kinds of fire is part of every fire fighter's job which is very dangerous especially when they have to enter into buildings or tunnels filled with smoke [2]. If robots could be designed to detect and put out fires, humans will not have to put their lives at risk. The effects of fire are unpredictable, and they can occur in both young, recently constructed forests and existing natural backwoods. Fire has a direct effect on plant growth because it removes unwanted plants, allowing new species to arise [3]. Moreover, robots are not affected by poisonous fumes or carbon monoxide, hence they can enter places where it is not safe for humans to enter [4]. Although different fires require different kinds of fire retardants, here an attempt is made to detect fire before it rages out of control and becomes very difficult to extinguish [5]. A complete robot would have to detect the source of fire and also identify the type of fire and then use the right type of fire retardant. The idea being presented is around a firefighting robot. Robots are more suited to errands than humans since they are more efficient, practical, and precise [6]. As technology evolved, it filled in the gaps in domination, simplifying human labour. The firefighting robot has been upgraded to detect and extinguish fires in impacted regions [7]. Such a robot would prove to be a boon to the fire fighters and they could work with the firefighters putting out fires faster and efficiently. These robots could be used to reach the victims faster hence reducing the risk of injury to victims [8]. This paper presents a primitive Fire Fighting Robot. Robots could be made to make intelligent decision about the type of fires and the choice of retardants to use. But that would require various types of gas sensors and other key knowledge to make a decision about the type and source of fire [9]. The robot designed is similar to Fire Fox, a firefighting robot that carries its own water tank.

This study includes an ARDUINO C++ programming code for performing the movement of the robot and extinguishing the fire detected. The code allows the free movement of the robot until it extinguishes the fire and manually stopped by the user, it will try to detect the fires nearby. It even detects any flames nearby and sprays water.

KEY REQUIREMENTS

1. **Fire Detection:** The robot is equipped with smoke sensors to accurately identify the presence and intensity of fire.
2. **Autonomy:** The robot operates autonomously, making real-time decisions based on environmental conditions and fire locations
3. **Mobility:** The robot can navigate diverse terrains (e.g., stairs, uneven surfaces) using suitable locomotion methods (wheels, tracks, or legs).
4. **Power Supply:** The design incorporates a reliable and sufficient power source to ensure extended operation during emergencies
5. **Extinguishing Mechanism:** The robot can effectively fires using extinguisher.

GOALS

1. **Step 01:** Develop a prototype that successfully detects and extinguishes a fire in a controlled environment.
2. **Step 02:** Test and evaluate the robot’s performance in various scenarios to refine its capabilities.
3. **Step 03:** Provide documentation and recommendations for future improvements and potential real-world applications.

COMPONENTS REQUIRED

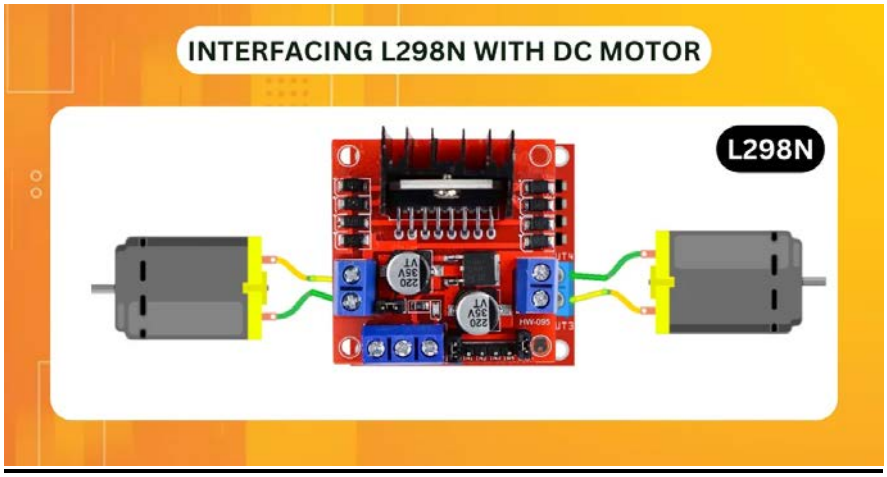
1. 5V 1 channel relay module compatible with 5V microcontroller (x 1)
2. DIY Mini Tubewell Kit (water Pump, Push Connector, Pipe, Battery Holder) starter Kit (x 1)
3. Single core wires (x 1)
4. Smart Motor Robot Car Battery Box Chassis Kit DIY Speed Encoder for Arduino(x1)
5. L298 Motor Driver Module (x 1)
6. Flame Sensor Module Detector for Temperature Detecting (x 3)
7. SG90 Micro Digital Servo Motor 9G for Robot Controls (x 1)

DESCRIPTION OF COMPONENTS

1. L298 MOTAR DRIVER MODULE

The L298N is a dual H-Bridge motor driver which allows speed and direction control of two DC motors at the same time.

This motor driver is one of the cheapest and the easiest way to control DC motors. One more advantage is that it can control a stepper motor as well.



2. SG90 MICRO DIGITAL SERVO MOTOR 9G

The SG90 servo is a small, affordable servo motor commonly used in robotics, DIY projects, and educational applications. It is lightweight and low-cost with plastic gears for applications where minimal torque is required.

The water pump is connected to the battery. The water pump's hose is attached to the servo motor which rotates it in the direction of the burning fire. The pump then sprays water filled in the tank with which it is connected to put out the fire.

3. FLAME SENSOR

A flame-sensor is one kind of detector which is mainly designed for detecting as well as responding to the occurrence of a fire or flame. We are using 3 sensors in our project.

Flame Sensor (3-Pin)

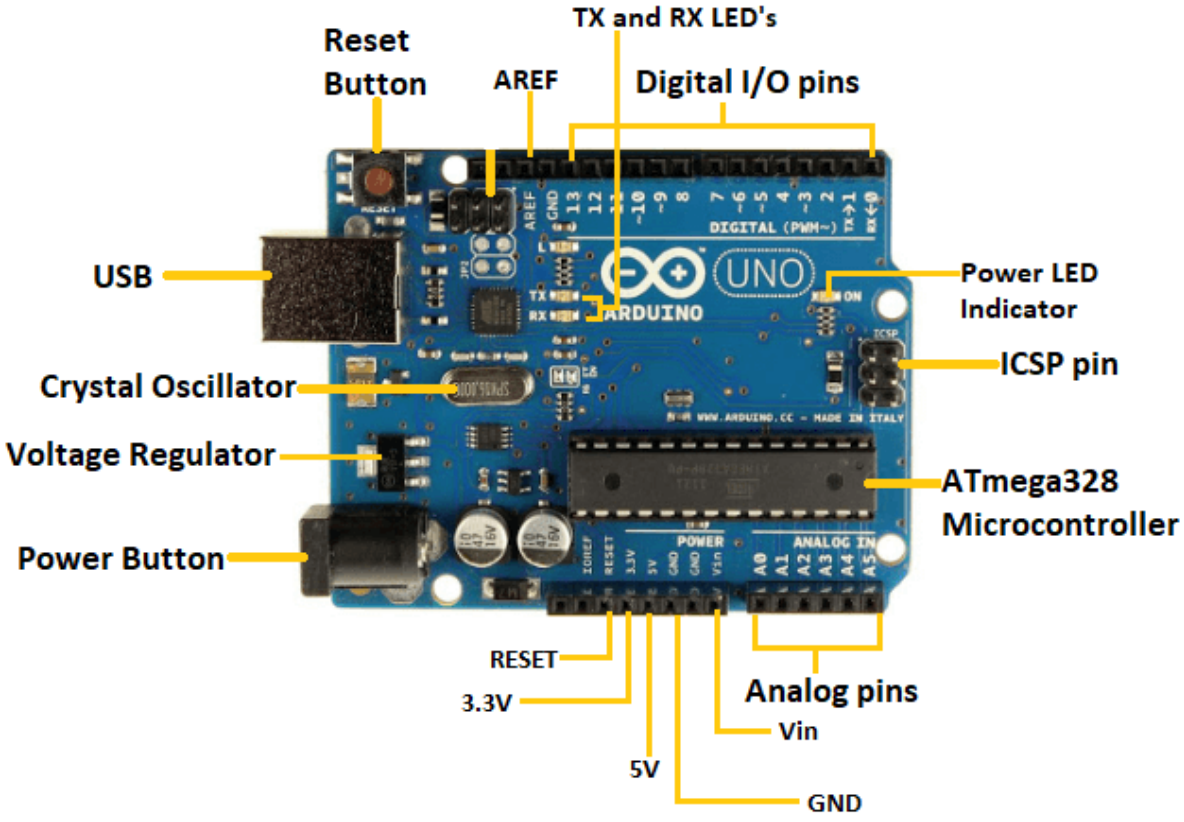
4. ARDUINO UNO

Arduino UNO is based on an ATmega328P microcontroller. It is easy to use compared to other boards, such as the Arduino Mega board, etc. The board consists of digital and analog Input/Output pins (I/O), shields, and other circuits.

The Arduino UNO includes 6 analog pin inputs, 14 digital pins, a USB connector, a power jack, and an ICSP (In-Circuit Serial Programming) header.

The USB port in the Arduino board is used to connect the board to the computer using the USB cable.

Arduino Uno is coded with Embedded c language

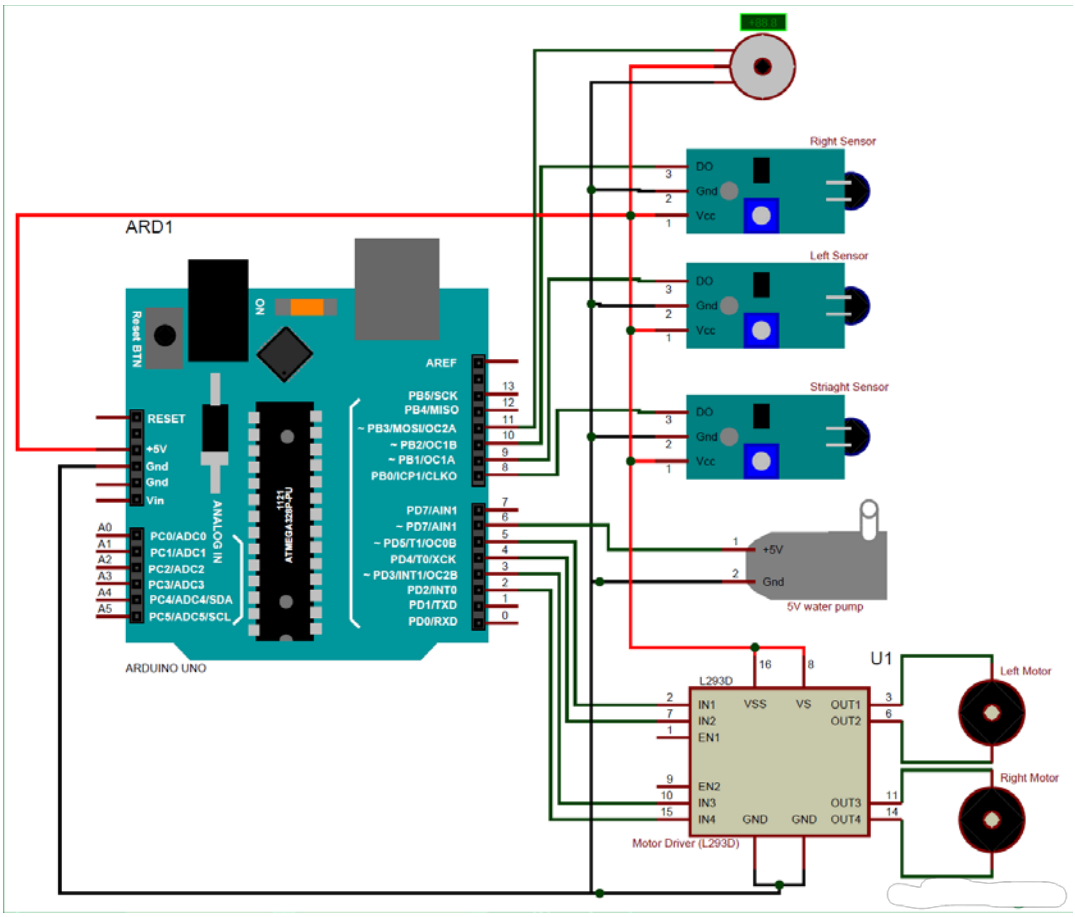


5. ROBOTIC CAR CHASIS

Package Includes:

1. 1 x Car chassis
2. 2 x Wheels
3. 1 x Castor wheel
4. 2 x BO Motors
5. 2 x Speed encoders
6. 1 x Battery holder
7. All necessary screw and nuts.

CIRCUIT DIAGRAM



EXPLANATION OF CIRCUIT DIAGRAM

1. **Power supply for micro controller:**
   - As micro controller (Arduino uno) will act as brain of the robot it needs a power supply
   - Power Source: The microcontroller will be powered by the battery from the DIY Mini Tubewell Kit. Ensure the voltage is within the microcontroller's operating range (e.g., 5V for Arduino).
2. **Flame Sensors connection:**
   - VCC: Connect the VCC pin of each flame sensor to the 5V pin of the microcontroller.
   - GND: Connect the GND pin of each flame sensor to the GND pin of the microcontroller.
   - Analog OUT: The flame sensor has a digital output pin that will give a HIGH signal when fire (infrared light) is detected. Connect each sensor's Analog OUT pin to a different Analog input pin on the microcontroller A1, A2, A4.
   - The microcontroller will monitor these input pins to know which sensor has detected fire and in which direction to move the robot.

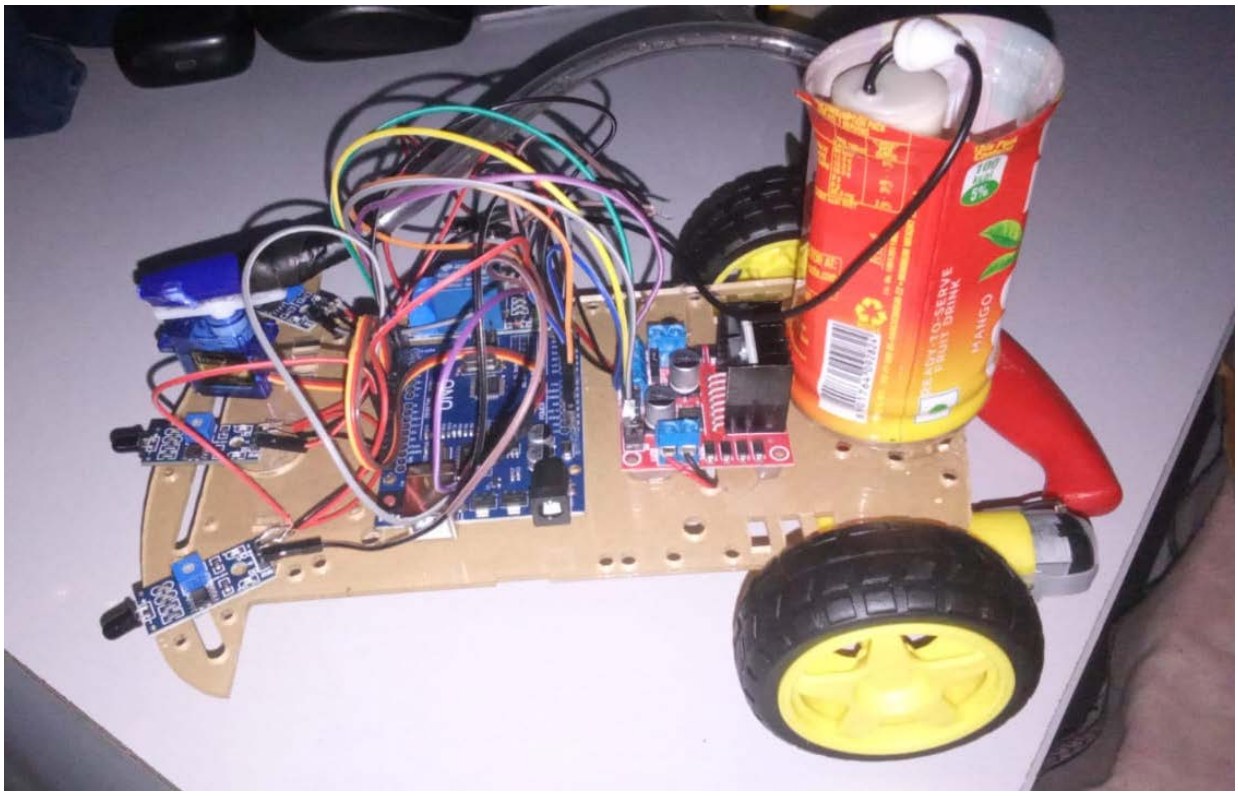
3. **L298 Motor Driver (for Robot Movement) Connections:**
    - Power Input: Connect the battery's positive terminal to the 12V pin on the L298 motor driver, and the negative terminal to the GND pin of the motor driver.
    - Motor Connections: The two motors from the Smart Motor Robot Car Battery Box Chassis Kit will be connected to the Motor Output A and Motor Output B terminals of the L298 motor driver. Motor 1 (left) connects to OUT1 and OUT2 of the L298 module.
    - Motor 2 (right) connects to OUT3 and OUT4 of the L298 module.
    - Control Pins:
        a. The L298 motor driver has input pins (IN1, IN2, IN3, and IN4) that will control the direction of each motor. Connect IN1 and IN2 to the microcontroller's digital pins (D2 and D3) to control Motor 1 (left).
        b. Connect IN3 and IN4 to other digital pins (D4 and D5) to control Motor 2 (right).
        c. The microcontroller will control these pins to move the robot forward, backward, or turn in the direction of the fire.

4. **SG90 Servo Motor (for Directing Water) Connections:**
    - VCC: Connect the servo motor's VCC pin to the 5V pin of the microcontroller.
    - GND: Connect the GND pin of the servo motor to the GND pin of the microcontroller.
    - Signal Pin: Connect the Signal pin of the servo motor to a PWM-enabled pin of the microcontroller (D11).
    - The servo motor will be controlled using PWM signals to change its angle. Based on where the fire is detected, the servo motor will adjust the direction of the water nozzle to aim the water pump.

5. **5V Relay Module (for Water Pump Control) Connections:**
    - VCC: Connect the relay’s VCC pin to the 5V pin of the microcontroller.
    - GND: Connect the GND pin of the relay to the microcontroller’s GND.
    - Control Pin: Connect the relay’s control (signal) pin to a digital output pin of the microcontroller (D6). This pin will activate the relay when fire is detected, allowing current to flow to the water pump.
    - Relay Output (NO and COM): The water pump from the DIY Mini Tubewell Kit is connected to the relay’s Normally Open (NO) and Common (COM) terminals. The relay will close the circuit and activate the water pump when it detects fire.

6. **DIY Mini Tubewell Kit (Water Pump) Power Source:**
   The water pump needs to be connected to the relay module, which controls when the water pump turns on. One terminal of the water pump will be connected to the NO terminal of the relay, and the other terminal will connect to the ground. When the relay closes, the water pump will receive power and start pumping water.

Component	Connections
Flame Sensors (3x)	VCC to 5V, GND to GND, OUT pins to microcontroller pins **A1, A2, A4**
L298 Motor Driver	Power from battery, OUT1/OUT2 and OUT3/OUT4 to motors, IN1-IN4 to **D2, D3, D4, D5**
Relay Module	VCC to 5V, GND to GND, Signal pin to **D6**, controls water pump through NO/COM
Water Pump	Connected to NO and COM of relay, activated when fire is detected
Servo Motor (SG90)	VCC to 5V, GND to GND, Signal pin to **D11 (PWM pin)**



WORKING PRINCIPLE

1. **Fire Detection**: The flame sensors continuously monitor the surroundings. When any sensor detects fire, it sends a signal to the microcontroller.
2. **Motor Control**: Based on which sensor detects the fire, the microcontroller controls the motors (via the L298 Motor Driver) to move the robot toward the fire.
3. **Fire Extinguishing**: Once close to the fire, the microcontroller activates the relay, turning on the water pump.
4. The servo motor adjusts the direction of the water stream towards the detected fire.
5. Autonomous Navigation: The robot can be programmed to stop when the fire is extinguished or continue to search for other flames.

CODE

```
#include <Servo.h>

Servo myservo;  // create servo object to control a servo
// twelve servo objects can be created on most boards

int pos = 0;

void setup() {
  // put your setup code here, to run once:
myservo.attach(11);
pinMode(2, OUTPUT);
pinMode(3, OUTPUT);
pinMode(4, OUTPUT);
pinMode(5, OUTPUT);
pinMode(6, OUTPUT);
pinMode(A1, INPUT);
pinMode(A2, INPUT);
pinMode(A4, INPUT);
Serial.begin(9600);
digitalWrite(6, LOW);
}

void loop() {
  // put your main code here, to run repeatedly:

int a = analogRead(A1);
int b = analogRead(A2);
int c = analogRead(A4);
Serial.print(a);
Serial.print("    ");
Serial.print(b);
Serial.print("    ");
Serial.print(c);
Serial.println("    ");
delay(50);

if (a<=250)

{
  digitalWrite(2, HIGH);
  digitalWrite(3, LOW);
  digitalWrite(4, HIGH);
```

```
digitalWrite(5, LOW);
delay(500);
digitalWrite(2, LOW);
digitalWrite(3, LOW);
digitalWrite(4, LOW);
digitalWrite(5, LOW);
digitalWrite(6, LOW);

for (pos = 60; pos <= 120; pos += 1) { // goes from 0 degrees to 180 degrees
  // in steps of 1 degree
  myservo.write(pos);              // tell servo to go to position in variable 'pos'
  delay(15);                       // waits 15 ms for the servo to reach the position
}
for (pos = 120; pos >= 60; pos -= 1) { // goes from 180 degrees to 0 degrees
  myservo.write(pos);              // tell servo to go to position in variable 'pos'
  delay(15);                       // waits 15 ms for the servo to reach the position
}
for (pos = 60; pos <= 120; pos += 1) { // goes from 0 degrees to 180 degrees
  // in steps of 1 degree
  myservo.write(pos);              // tell servo to go to position in variable 'pos'
  delay(15);                       // waits 15 ms for the servo to reach the position
}
for (pos = 120; pos >= 60; pos -= 1) { // goes from 180 degrees to 0 degrees
  myservo.write(pos);              // tell servo to go to position in variable 'pos'
  delay(15);                       // waits 15 ms for the servo to reach the position
}
for (pos = 60; pos <= 120; pos += 1) { // goes from 0 degrees to 180 degrees
  // in steps of 1 degree
  myservo.write(pos);              // tell servo to go to position in variable 'pos'
  delay(15);                       // waits 15 ms for the servo to reach the position
}
for (pos = 120; pos >= 60; pos -= 1) { // goes from 180 degrees to 0 degrees
  myservo.write(pos);              // tell servo to go to position in variable 'pos'
  delay(15);                       // waits 15 ms for the servo to reach the position
}
delay(500);

}


else if(b<=250)

{
```

```
digitalWrite(2, HIGH);
  digitalWrite(3, LOW);
  digitalWrite(4, HIGH);
  digitalWrite(5, LOW);
  delay(500);
  digitalWrite(2, LOW);
  digitalWrite(3, LOW);
  digitalWrite(4, LOW);
  digitalWrite(5, LOW);

  digitalWrite(6, LOW);
for (pos = 60; pos <= 120; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15 ms for the servo to reach the position
  }
  for (pos = 120; pos >= 60; pos -= 1) { // goes from 180 degrees to 0 degrees
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15 ms for the servo to reach the position
  }
  for (pos = 60; pos <= 120; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15 ms for the servo to reach the position
  }
  for (pos = 120; pos >= 60; pos -= 1) { // goes from 180 degrees to 0 degrees
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15 ms for the servo to reach the position
  }
  for (pos = 60; pos <= 120; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15 ms for the servo to reach the position
  }
  for (pos = 120; pos >= 60; pos -= 1) { // goes from 180 degrees to 0 degrees
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15 ms for the servo to reach the position
  }

  }
  else if(c<=250)

{
```

```
digitalWrite(2, HIGH);
  digitalWrite(3, LOW);
  digitalWrite(4, HIGH);
  digitalWrite(5, LOW);
  delay(500);
  digitalWrite(2, LOW);
  digitalWrite(3, LOW);
  digitalWrite(4, LOW);
  digitalWrite(5, LOW);
  digitalWrite(6, LOW);
for (pos = 60; pos <= 120; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15 ms for the servo to reach the position
  }
  for (pos = 120; pos >= 60; pos -= 1) { // goes from 180 degrees to 0 degrees
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15 ms for the servo to reach the position
  }
  for (pos = 60; pos <= 120; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15 ms for the servo to reach the position
  }
  for (pos = 120; pos >= 60; pos -= 1) { // goes from 180 degrees to 0 degrees
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15 ms for the servo to reach the position
  }
  for (pos = 60; pos <= 120; pos += 1) { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15 ms for the servo to reach the position
  }
  for (pos = 120; pos >= 60; pos -= 1) { // goes from 180 degrees to 0 degrees
    myservo.write(pos);              // tell servo to go to position in variable 'pos'
    delay(15);                       // waits 15 ms for the servo to reach the position
  }


  }



else if(a>=251 && a<=700)

{
```

```
  digitalWrite(6, HIGH);
  digitalWrite(2, HIGH);
  digitalWrite(3, LOW);
  digitalWrite(4, HIGH);
  digitalWrite(5, LOW);
  delay(500);
  digitalWrite(6, HIGH);
  digitalWrite(2, LOW);
  digitalWrite(3, LOW);
  digitalWrite(4, LOW);
  digitalWrite(5, LOW);
  delay(500);

  }



else if(b>=251 && b<=800)

{ digitalWrite(2, HIGH);
  digitalWrite(3, LOW);
  digitalWrite(4, HIGH);
  digitalWrite(5, LOW);

  digitalWrite(6, HIGH);




  }



else if(c>=251 && c<=800)

{



  digitalWrite(2, HIGH);
  digitalWrite(3, LOW);
  digitalWrite(4, HIGH);
  digitalWrite(5, LOW);
```

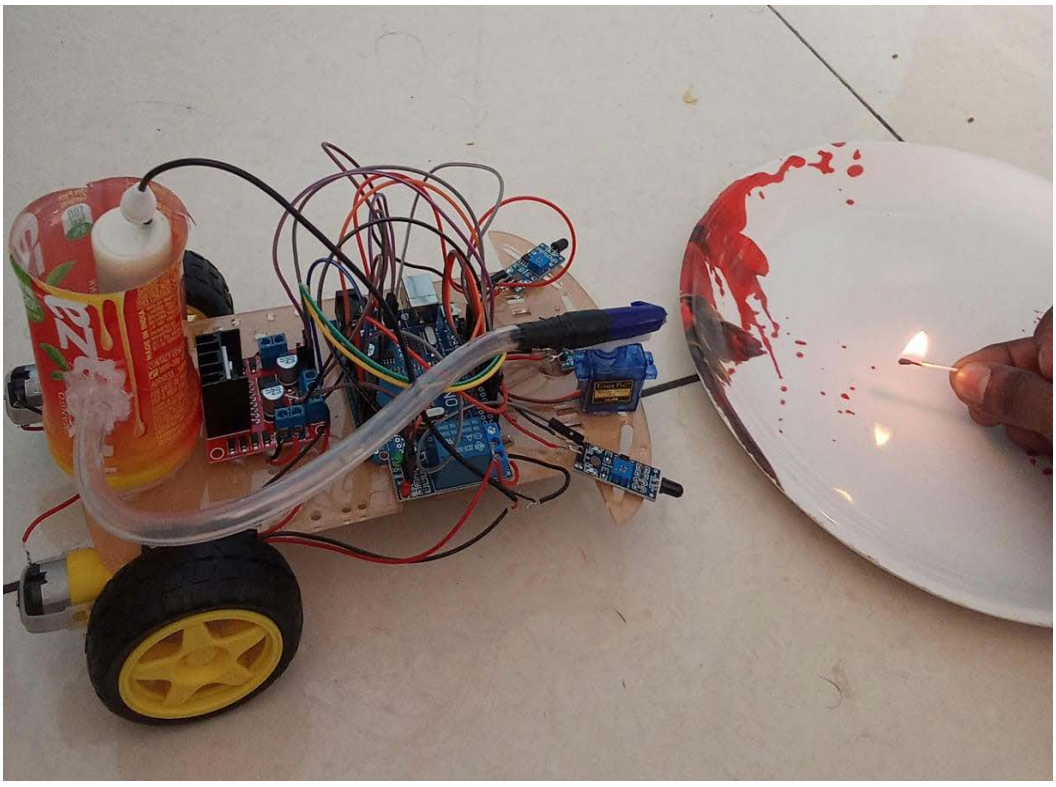
```
  digitalWrite(6, HIGH);
  delay(500);
  digitalWrite(6, HIGH);

  digitalWrite(2, HIGH);
  digitalWrite(3, LOW);
  digitalWrite(4, HIGH);
  digitalWrite(5, LOW);
  delay(500);

  }

else

{
  digitalWrite(6, HIGH);
  digitalWrite(2, LOW);
  digitalWrite(3, LOW);
  digitalWrite(4, LOW);
  digitalWrite(5, LOW);
  }
  }
```



CONCLUSION

In conclusion, the development of a firefighting robot for our computational physics project demonstrates the significant intersection of robotics, physics, and engineering principles. Through our simulations and designs, we have effectively modeled the robot's dynamics, optimizing its performance in challenging environments. The integration of sensors for smoke detection and temperature measurement, alongside precise movement algorithms, highlights the potential for real-world applications in emergency response.

Future work could involve enhancing the robot's autonomy and incorporating advanced AI for improved decision-making in complex scenarios. Ultimately, this project not only showcases the capabilities of computational physics in solving practical problems but also contributes to the ongoing efforts in improving fire safety and disaster response technologies. The insights gained here could pave the way for more sophisticated robotic systems in various fields, underscoring the importance of interdisciplinary collaboration in advancing technology for the greater good.

FUTURE WORK

In our model, it uses only water, but in future we can asses the type of fire and use CO2, etc. whichever is required at that moment for extinguishing the fire. It eventually reduces the delay time as it does not take time in deciding the fire type. And it also does not take time in deciding the type of resource to use to extinguish the fire. As our firefighting robot uses water with a submersible pump to extinguish the fire. It can also be made into a self-balancing robot where if the front wheels are damaged due to some unforeseen circumstances.

REFERENCE

[1] An Arduino Uno Controlled Fire Fighting Robot for Fires in Enclosed Spaces by Monica P Suresh; V R Vedha Rhythesh; J Dinesh; K Deepak; J Manikandan

[2] Fire Fighting Robot with More Accuracy and
Implementation Using Arduino Uno by Ayush Dubey; Bhartendu Singh; Narendra Kumar; Mohammed Ayad Alkhafaji; Mustafa Isam

[3] FIRE FIGHTING ROBOT by PRAMOD BN, HEMALATHA KN, Poornima BJ, Harshitha R, PRAMOD GN, October 2019

[4] AUTONOMOUS FIRE PROTECTION ROBOT WITH
NOTIFICATION by Faculty of Electrical & Electronic Engineering University Tun Hussein Onn Malaysia.

[5] A low-cost sensor based autonomous and semi-autonomous firefighting squad robot by Satya Veera Pavan Kumar Maddukuri; Uday Kishan Renduchintala; Aravinthan Visvakumar; Chengzong Pang; Sravan Kumar Mittapally

[6] Fire Fighting Robotic Vehicle using Arduino by S. Dina; M. Mahalekshmi; V. Kokila; R. JeyaPreetha; S. Sankarakumar.

[7] Nagesh M S, Deepika T V, Stafford Michael, and Dr. M
Shivakumar, Fire Extinguishing Robot, International Journal of Advanced Research in Computer and Communication Engineering, 5(12), December 2016.