Embedded Firefighter Robot Car

Embedded Systems Final Design Project, Spring 2024/2025

King Abdullah II School of Engineering

Princess Sumaya University for Technology

Supervisor: Dr. [Your Supervisor’s Name]

Team Members:

Ahmad Alhiary (20210074)

Mohammad Rowad (20200726)

Hamza Melhem (20200881)

# Table of Contents

* 1. Abstract
* 2. Introduction and Background
* 3. Design
* 3.1 Mechanical Design
* 3.2 Electrical Design
* 3.3 Software Design
* 4. Problems & Recommendations
* 5. Conclusion

# Abstract

In a world where technology continues to evolve to make our lives safer and more efficient, this project introduces the Embedded Firefighter Robot Car—an innovative solution aimed at enhancing fire safety. This autonomous robotic system is designed to detect and suppress small fires in environments that are potentially hazardous or inaccessible to humans. By combining embedded system technologies, flame detection sensors, infrared obstacle avoidance, and a functional water spray mechanism, this robot showcases a practical and impactful application of modern engineering. This report details the conceptualization, design, challenges, and outcomes of developing the Firefighter Robot Car, which can serve as a prototype for future emergency response systems in smart buildings, warehouses, or industrial plants.

# Introduction and Background

Fire emergencies remain one of the most dangerous situations in both industrial and domestic settings. Traditional methods of firefighting often involve significant human risk, especially in confined spaces or locations with limited visibility. With the aim of reducing these risks, we embarked on the development of an autonomous robot that could assist or operate independently during fire outbreaks. The concept revolves around enabling a mobile system, powered by an embedded microcontroller, to identify flames and navigate its surroundings while avoiding obstacles. Once a fire is detected, the robot initiates a targeted water-spraying action to suppress it. This project not only reflects our understanding of embedded systems but also illustrates the meaningful contribution of automation in safety-critical applications.

## 3.1 Mechanical Design

Mechanically, the Firefighter Robot Car is built on a robust four-wheel drive chassis, providing stability and smooth maneuverability. The choice of a compact and sturdy frame was essential to ensure that the robot could navigate through narrow pathways or cluttered areas without tipping or losing balance. A small water tank and an electrically operated pump are securely mounted on the base, with a nozzle fixed at the front to allow forward spraying. Flame sensors and IR sensors are carefully positioned at strategic points around the robot’s frame to maximize environmental awareness. The design is modular, allowing for easy upgrades and maintenance. Each component has been selected for its durability, ensuring that the robot can withstand heat, slight moisture, and mechanical stress during operation.

## 3.2 Electrical Design

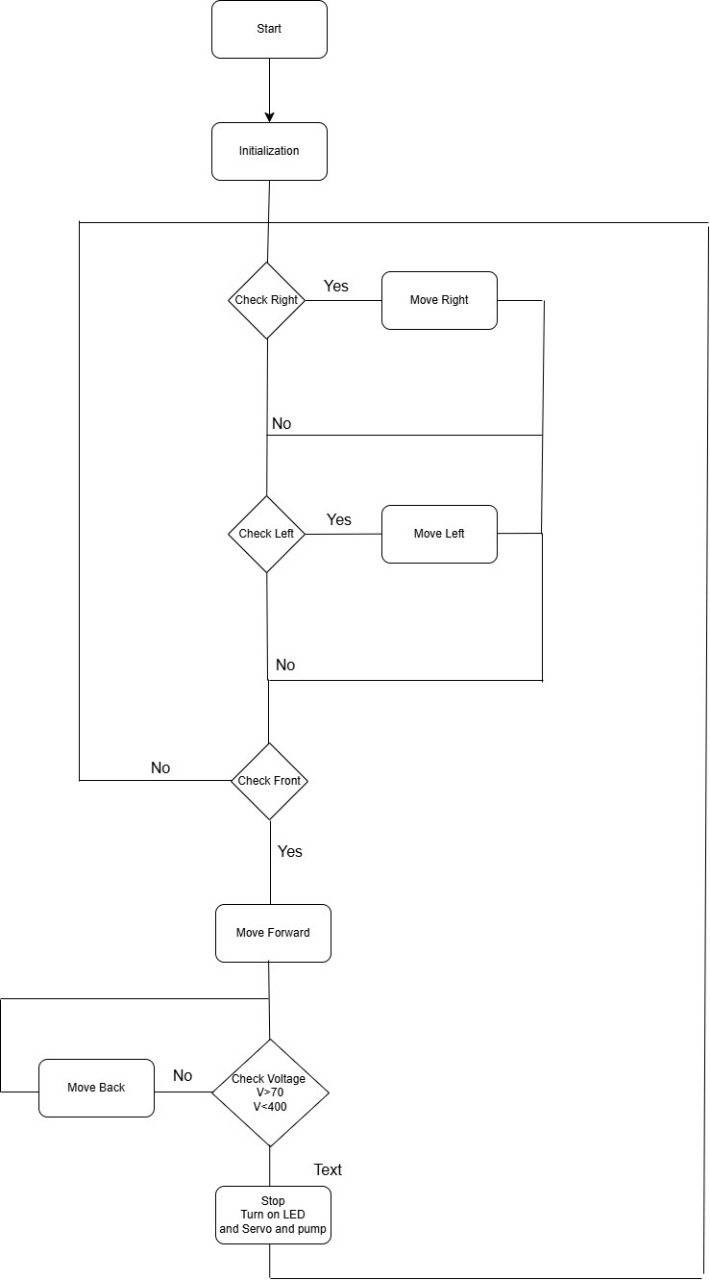
The electrical architecture of the robot forms the nervous system of the entire project. At the heart of the system lies an embedded microcontroller that handles input from various sensors and outputs control signals to the actuators. Flame sensors are used to detect the presence and intensity of fire by capturing infrared radiation. To ensure that the robot avoids collisions, infrared sensors are implemented to detect nearby obstacles. The DC motors, responsible for movement, are driven via motor drivers that receive PWM signals from the microcontroller to adjust speed and direction. Additionally, a small but powerful water pump is wired to activate when the system confirms fire detection. Power management was a critical part of the design; the system was divided into separate power rails to avoid voltage drops and performance fluctuations when all components operate simultaneously.

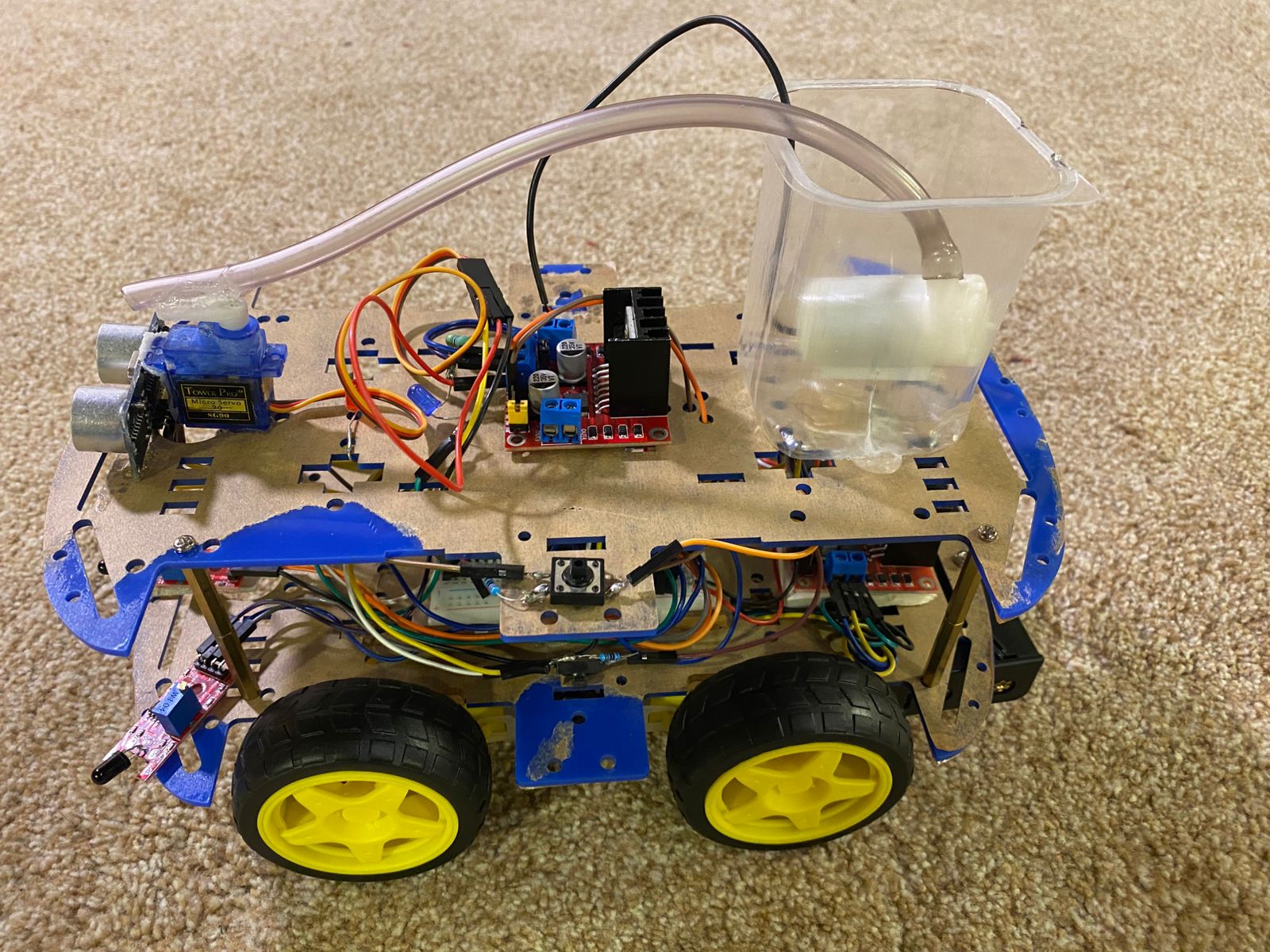
## 3.3 Software Design

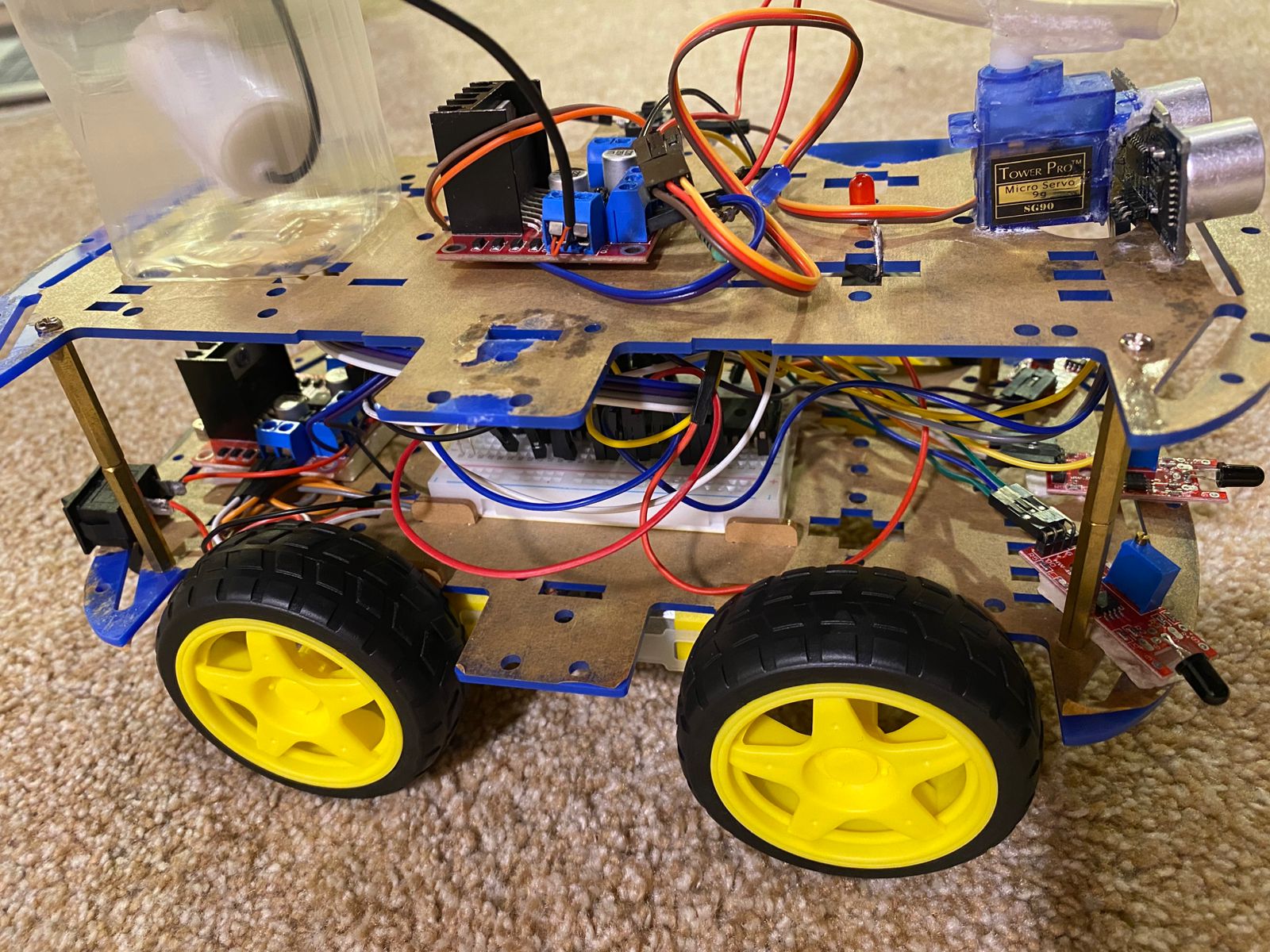
The embedded software serves as the intelligence of the robot. Developed using a structured programming approach, the code enables the robot to continuously scan its surroundings, interpret sensor data, and execute actions in real time. The microcontroller runs a main control loop that constantly reads values from the flame and obstacle sensors. Based on pre-defined thresholds and decision-making logic, it decides whether the robot should move forward, stop, turn, or activate the water pump. A state-machine design is used to ensure smooth transitions between different behaviors. Additional conditions are incorporated to minimize false alarms and ensure the robot does not spray water unnecessarily. The software is also optimized for responsiveness, ensuring the robot reacts quickly in emergency scenarios without lag or hesitation.

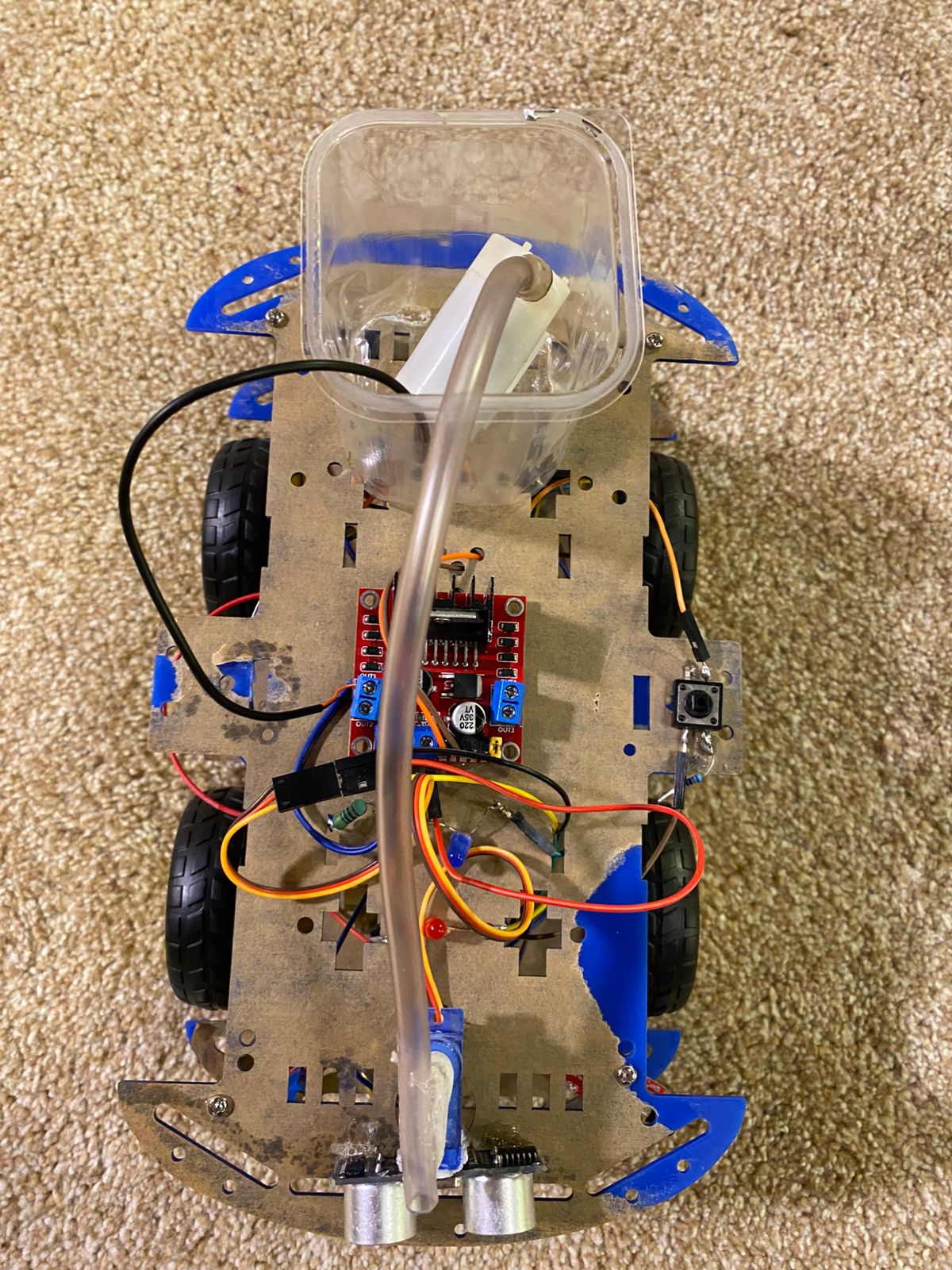
**Key Components**

* **Ultrasonic Sensor:** Acts like the robot's eyes—uses sound to detect obstacles and avoid bumps.
* **Servo Motor:** Turns parts (like the nozzle) to aim water accurately when fire is spotted.
* **H-Bridge Motor Driver:** Controls the direction and speed of the wheels—lets the robot move forward, backward, and turn.
* **Water Pump:** Sprays water when fire is detected—like the robot’s fire extinguisher.
* **Flame Sensor:** Detects fire by sensing light and heat—tells the robot where to aim.
* **LEDs:** Show what the robot is doing—like blinking when fire is found.
* **Push Button:** Used to start or stop the robot manually.
* **DC Motors:** Drive the wheels to move the robot around—controlled by the H-bridge.



****

****

****

**Source Code:**

**//PUMP PIN RB7**

**int tick;**

**int tick1;**

**int sensor\_voltage;**

**int H\_L;**

**int angle;**

**int distance;**

**int a;**

**int b;**

**int time;**

**int turning\_time\_delay = 5000;**

**void initialize();**

**void interrupt();**

**void stop\_moving();**

**void move\_left();**

**void move\_right();**

**void move\_forward();**

**void move\_backwards();**

**void water\_pump\_ON();**

**void water\_pump\_OFF();**

**void adjust\_position();**

**void check\_and\_extinguish\_fire();**

**int dist();**

**void delayy(int time\_ms) {**

**time=0; // Reset overflow counter**

**while (time < time\_ms);**

**}**

**void CCPPWM\_init(){ // Configure and CCP2 at 2ms period with 50% duty cycle**

**T2CON = 0x07; // Enable Timer2 at Fosc/4 with 1:16 prescaler (8 uS percount 2000uS to count 250 counts)**

**CCP2CON = 0x0C; // Enable PWM for CCP2**

**PR2 = 250; // 250 counts = 8uS \*250 = 2ms period**

**CCPR2L = 125; // Buffer where we are specifying the pulse width (duty cycle)**

**}**

**void Speed(int p){**

**CCPR2L = p; // PWM from RC1**

**}**

**void ATD\_init\_A0();**

**int ATD\_read\_A0();**

**void check\_front\_right();**

**void check\_front\_left();**

**void check\_front();**

**void check\_start();**

**void main() {**

**initialize();**

**ATD\_init\_A0();**

**CCPPWM\_init();**

**b=0;**

**a=1;**

**while(1){**

**PORTD=PORTD & 0B11011111;**

**check\_start();**

**while(b){**

**PORTD=PORTD | 0B00100000;**

**check\_front\_right();**

**check\_start();**

**check\_front\_left();**

**check\_start();**

**check\_front();**

**check\_start();**

**adjust\_position();**

**check\_start();**

**check\_and\_extinguish\_fire();**

**check\_start();**

**}**

**}**

**}**

**void initialize(){**

**TRISA=0X01;**

**TRISB=0X00;**

**TRISC=0B10000000;**

**TRISD=0B00001111;**

**PORTA=0X00;**

**PORTB=0X00;**

**PORTC=0X00;**

**PORTD=0X00;**

**//For Timer 0**

**OPTION\_REG = 0x87; // 32.8ms overflow**

**TMR0 = 0;**

**TMR1H = 0;**

**TMR1L = 0;**

**H\_L = 1; // start high**

**CCP1CON = 0x08; // Compare mode, set output on match**

**T1CON = 0x01;**

**INTCON = 0b11100000; // Enable GIE, peripheral interrupts and TMR0 Interrupt**

**PIE1 = PIE1|0x04; // Enable CCP1 interrupts**

**CCPR1H = 2000>>8; // Value preset in a program to compare the TMR1H value to - 1ms**

**CCPR1L = 2000; // Value preset in a program to compare the TMR1L value to**

**}**

**void interrupt(){**

**if(INTCON & 0x04){// TMR0 Overflow interrupt, will get here every 32ms**

**time++;**

**tick1++;**

**INTCON = INTCON & 0xFB;//Clear T0IF**

**}**

**if(PIR1 & 0x04){**

**if(a==1){ // CCP1 interrupt**

**if(H\_L){ // high**

**CCPR1H = angle >> 8;**

**CCPR1L = angle;**

**H\_L = 0; // next time low**

**CCP1CON = 0x09; // compare mode, clear output on match**

**TMR1H = 0;**

**TMR1L = 0;**

**}**

**else{ //low**

**CCPR1H = (40000 - angle) >> 8; // 40000 counts correspond to 20ms**

**CCPR1L = (40000 - angle);**

**CCP1CON = 0x08; // compare mode, set output on match**

**H\_L = 1; //next time High**

**TMR1H = 0;**

**TMR1L = 0;**

**} }else{**

**PIR1 = PIR1&0xFB;**

**}**

**PIR1 = PIR1&0xFB;**

**}**

**}**

**void check\_start(){**

**if(PORTD & 0B00001000){**

**b++;**

**}**

**if(b%2==0){**

**b=0;**

**}else{b=1;}**

**}**

**void ATD\_init\_A0(){**

**ADCON0=0x41; // ON, Channel 0, Fosc/16== 500KHz, Dont Go**

**ADCON1=0xCE; // RA0 Analog, others are Digital, Right Allignment,**

**TRISA=0x01;**

**}**

**int ATD\_read\_A0(){**

**ADCON0=ADCON0 | 0x04; // GO**

**while(ADCON0&0x04); // wait until DONE**

**return (ADRESH<<8)|ADRESL;**

**}**

**void check\_front\_right(){**

**// read port b4 : right flame sensor**

**if(PORTD & 0b00000001){**

**tick1 = 0;**

**// read port b7 : front sensor until detects fire**

**while(!(PORTD & 0b00000100)){**

**// if it turns more than the turning\_th stop (turning\_th is time)**

**if (tick1 >= turning\_time\_delay) break;**

**// move right**

**move\_right();**

**}**

**delay\_ms(100);**

**// stop moving**

**stop\_moving();**

**}**

**}**

**void check\_front\_left(){**

**// read port b5 : left flame sensor**

**if ((PORTD & 0b00000010)){**

**tick1 = 0;**

**// read port b7 : front sensor until detects fire**

**while(!(PORTD & 0b00000100)){**

**// move left**

**if (tick1 >= turning\_time\_delay) break;**

**move\_left();**

**}**

**// stop moving**

**stop\_moving();**

**}**

**}**

**void check\_front(){**

**while((PORTD & 0b00000100)){**

**sensor\_voltage = ATD\_read\_A0();**

**// while flame detected approach it untill specified flame strength**

**while(sensor\_voltage <= 70 || sensor\_voltage >= 400){**

**sensor\_voltage = ATD\_read\_A0();**

**move\_forward();**

**distance=dist();**

**if(distance<20){**

**move\_right();**

**delayy(2000);**

**move\_forward();**

**delayy(2000);**

**move\_left();**

**delayy(2000);**

**move\_forward();**

**}else{move\_forward();}**

**if(!(PORTD & 0b00000100)){break;}**

**}**

**if(sensor\_voltage <= 70) {break;}**

**if((PORTD & 0b00000100)){break;} }**

**delayy(100);**

**stop\_moving();**

**}**

**void stop\_moving(){**

**PORTB = PORTB & 0b11110000;**

**Speed(0);**

**}**

**void move\_left(){**

**stop\_moving();**

**Speed(90);**

**PORTB = PORTB | 0b00000001;**

**PORTB = PORTB | 0b00001000;**

**}**

**void move\_right(){**

**stop\_moving();**

**Speed(90);**

**PORTB = PORTB | 0b00000100;**

**PORTB = PORTB | 0b00000010;**

**}**

**void move\_forward(){**

**stop\_moving();**

**Speed(170);**

**PORTB = PORTB |0b00000001;**

**PORTB = PORTB |0b00000100;**

**}**

**void move\_backwards(){**

**stop\_moving();**

**Speed(90);**

**PORTB = PORTB | 0b00000010;**

**PORTB = PORTB | 0b00001000;**

**}**

**void adjust\_position(){**

**while((PORTD & 0b00000100)){**

**sensor\_voltage = ATD\_read\_A0();**

**while(sensor\_voltage < 70){**

**move\_backwards();**

**sensor\_voltage = ATD\_read\_A0();**

**}**

**if( sensor\_voltage>=70) {break;}**

**if(!(PORTD & 0b00000100)){break;}**

**}**

**delayy(100);**

**stop\_moving();**

**}**

**void check\_and\_extinguish\_fire(){**

**//pump on while there is fire:**

**sensor\_voltage = ATD\_read\_A0();**

**while ((PORTD & 0b00000100))**

**{ //check if fire is too close or fire is too far (we check if its too far because**

**//we read the digital signal from the front flame sensor so as not to pump water from too far distance)**

**if(sensor\_voltage <70 || sensor\_voltage > 400) break;**

**stop\_moving();**

**a=1;**

**PORTB=PORTB|0B10000000;**

**angle = 1000;**

**PORTD=PORTD | 0B11000000;**

**delayy(2000);**

**angle = 3500;**

**PORTD=PORTD & 0B00111111;**

**delayy(2000);**

**angle = 1000;**

**PORTD=PORTD | 0B11000000;**

**delayy(2000);**

**angle = 3500;**

**PORTD=PORTD & 0B00111111;**

**delayy(2000);**

**angle = 1000;**

**PORTD=PORTD | 0B11000000;**

**}**

**PORTD=PORTD & 0B00111111;**

**angle = 2250;**

**delayy(2000);**

**a=0;**

**PORTB=PORTB & 0B01111111;**

**}**

**int dist(){**

**int d = 0;**

**T1CON = 0x10; // Use internal clock, no prescaler**

**delayy(100);**

**TMR1H = 0; // Reset Timer1**

**TMR1L = 0;**

**PORTC = PORTC | 0b01000000; // Trigger HIGH**

**delayy(1); // 1 ms delay**

**PORTC = PORTC & 0b10111111; // Trigger LOW**

**while (!(PORTC & 0b10000000));**

**T1CON = T1CON | 0b00000001; // Start Timer**

**while (PORTC & 0b10000000);**

**T1CON = T1CON & 0b11111110; // Stop Timer**

**d = (TMR1L | (TMR1H << 8)); // Read Timer1 value**

**d = d / 58.82; // Convert time to distance (cm)**

**delayy(10);**

**T1CON = 0x01;**

**return d;**

**}**

# Problems & Recommendations

During development, several technical challenges were encountered that shaped our understanding and improved our design decisions:

1. False Fire Detection: Initially, the flame sensors misinterpreted strong ambient light as fire, triggering false alarms. We resolved this by calibrating the sensors and incorporating filtering algorithms to improve detection accuracy.

2. Water Pump Limitations: The first water pump selected lacked sufficient pressure to extinguish a flame effectively. After testing multiple options, we upgraded to a more powerful unit that could deliver consistent and targeted spraying.

3. Power Distribution: Using a single power source for both motors and sensors caused intermittent resets. We resolved this by redesigning the power circuit, introducing voltage regulators and separate power buses for high-load and logic-level components.

For future iterations, we recommend:

- Integrating thermal cameras or UV sensors for more precise fire detection.

- Adding remote control capabilities via Wi-Fi or Bluetooth for manual override.

- Using rechargeable lithium-ion battery packs for improved energy efficiency.

- Upgrading the chassis with heat-resistant materials to improve survivability in real fire environments.

# Conclusion

The Embedded Firefighter Robot Car stands as a testament to the power of embedded systems when combined with a clear, real-world purpose. By merging mechanical design, sensor integration, and intelligent programming, we created a mobile platform capable of acting independently in critical situations. This project not only enhanced our technical skills but also opened our eyes to the broader societal impact of engineering. The robot has potential applications in smart building safety, industrial fire prevention, and educational robotics. With continued development, we believe it can be transformed into a commercially viable product, contributing meaningfully to modern fire safety strategies.