1. Introduction

In the search for convenience and efficiency, IoT emerges as the perfect solution, blending technology seamlessly into daily life. The idea of Internet of Things originated in 1990s with the rise in integration of Information Technology with Operational Technology (Yousif, 2023).

With the rise in technology, it is important to know about IOT trends, their potential and wise implementation of innovation in our daily life. With this motive, we have been assigned an IOT project. Following the trend, we chose to explore one of the areas of IOT innovation i.e., the Smart Vehicle. By integrating the components and utilizing the IOT technology, this project aims to build a prototype of Obstacle Avoidance Car which has capability of obstacle detection, connectivity, and user-controlled navigation with prewritten programs.

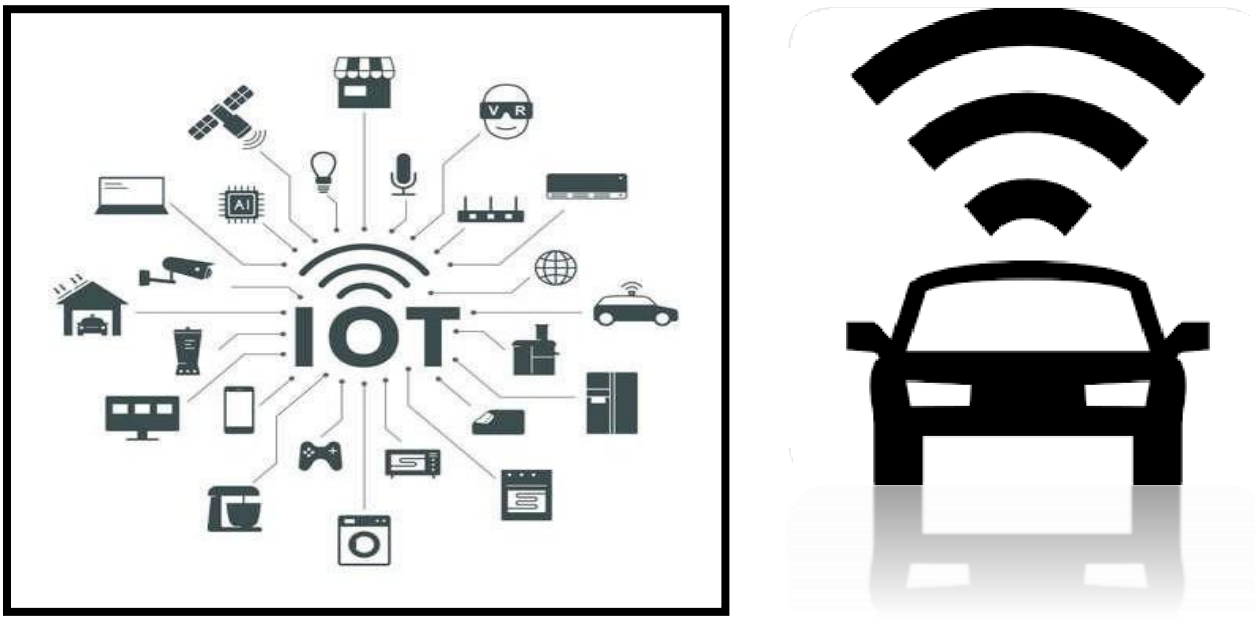

Figure 1: Application of IoT

1.1. Current scenario

Cities worldwide are suffering from urbanization trends and urban planning challenges. Private vehicles, what once was considered as the luxury has now become the necessity for personal mobility due to the lack of efficient public transport systems in Nepal. With the growing population of Nepal i.e., **29,164,578** as published by **National Statistics Office, GoN in 2021** and growing demands of vehicles, cities are facing difficulties like traffic jams, parking problem and inefficient road utilization leading to failure in reaching the destination.

Figure 2: Traffic Jam in Nepal

In the context of Nepal, it has extensive network of narrow roads and streets. Improper parking of bikes, different animals like cows and dogs moving unpredictably on the street and various obstacles adds chaos to the urban mobility in the narrow streets which results in accidents and physical damage to the property. To mitigate this, embracing technology is the only way.

1.2. Problem Statement and Project as a solution

According to the Nepal Demographic and Health Survey-2022, the country experiences a significant rate of fatalities due to road accidents, with an average of **14** out of every **100,000** people losing their lives in road accidents (The Kathmandu Post , 2024). Statistics conveys significant challenges, including frequent accidents, property damage, and chaotic traffic conditions, particularly in densely populated urban areas and narrow roads in Nepal.

The proposed solution can be the development of **Obstacle Avoidance Car**, equipped with advanced sensors, real- time data processing. The project aims to support road safety with precise obstacle detection and controlled navigation capabilities. IoT connectivity is great for processing real time data and controlling the car with the device we have in reach, with the help of user-friendly interface hosted. In situations where there is limited space for opening car doors in a parking area, the car can be parked remotely with no driver in.

1.3. Aims and Objectives

The main aim of the project is to build an Obstacle Avoidance Car integrated with advanced sensors to solve the problem of every vehicle owner struggling to reach destinations via urban roads and streets with safety issues.

Objectives

- To research about the components with its usage that can utilized to build the car
- To implement the advanced senor systems like IR Sensors to detect the things present on the path.
- To design a solid algorithm for analyzing sensor data and safe navigation.
- To research cloud platforms and implement real time data storage.
- To build a user-friendly interface for the proper navigation of the car.
- To conduct the testing to ensure the capability and performance of the car under various conditions.

2. Background

2.1. System Overview

The prototype of the Smart Car meets the aim of building a car which detects obstacles on path and stops accordingly which prevents accidents. Different components like micro- controller, sensors, moto driver, motors and other peripherals complete the prototype. In the project, two micro- controllers are used for the purpose of user-control over navigation and overall functioning of the car. The microcontroller acts as the brain of the prototype by collecting the data, processing it, and giving instructions to actuators. The technology is powered by batteries.

The microcontroller acts as the brain of the prototype by collecting the data, processing it, and giving instructions to actuators. The real-time data collected by the sensor is analyzed and according to prewritten algorithms, the car performs the action. The UI for controlling the car was hosted in the NodeMCU. In this way, the car navigates based on user-interaction and prewritten algorithms for obstacle detection and avoidance.

2.2. Hardware architecture of the system

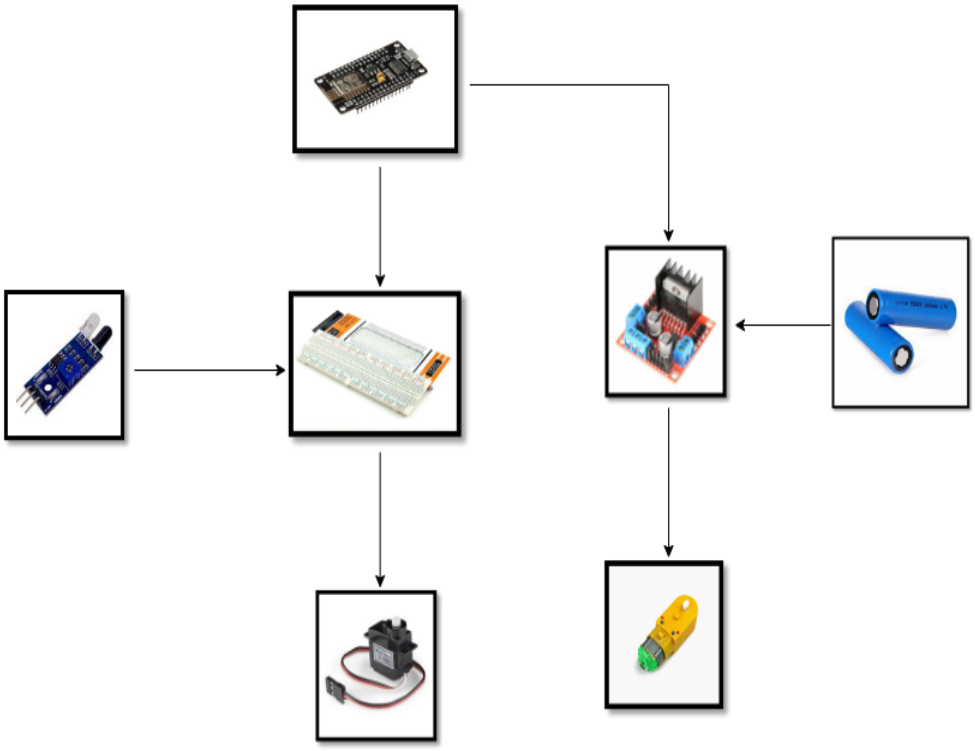

Figure 3: Hardware Architecture of the Prototype

2.3. Flowchart of the system

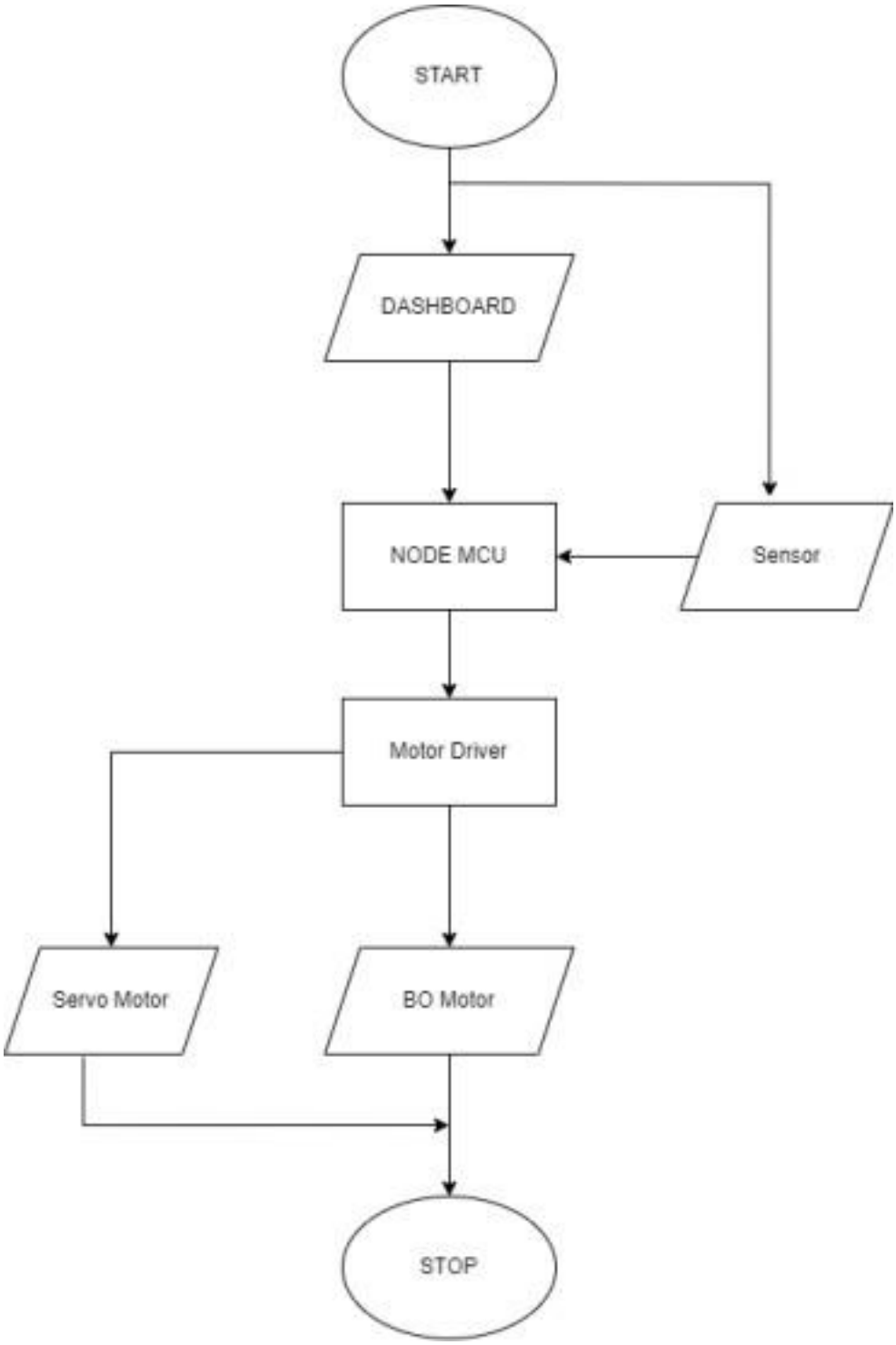

Figure 4: Flowchart for the system.

2.4. Circuit diagram of the system

- **Pictorial Diagram**

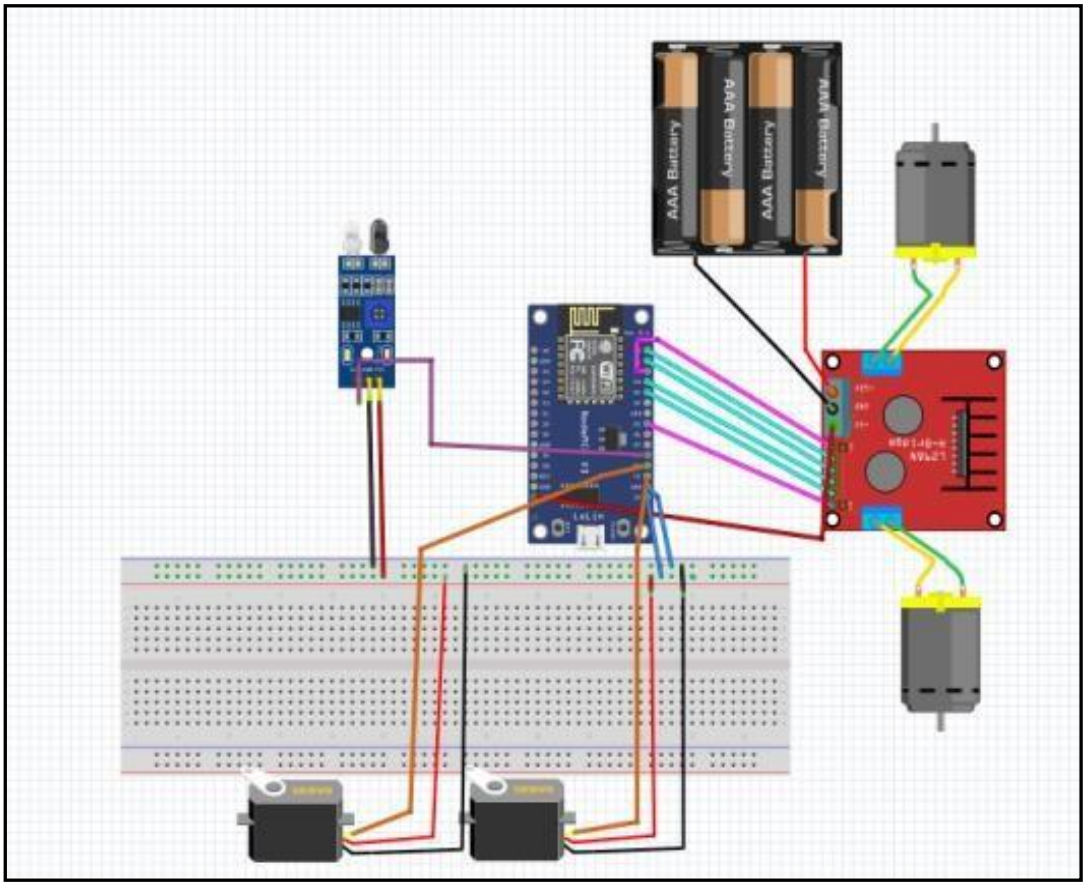

Figure 5: Pictorial Circuit Diagram of the System

- **Schematic Diagram**

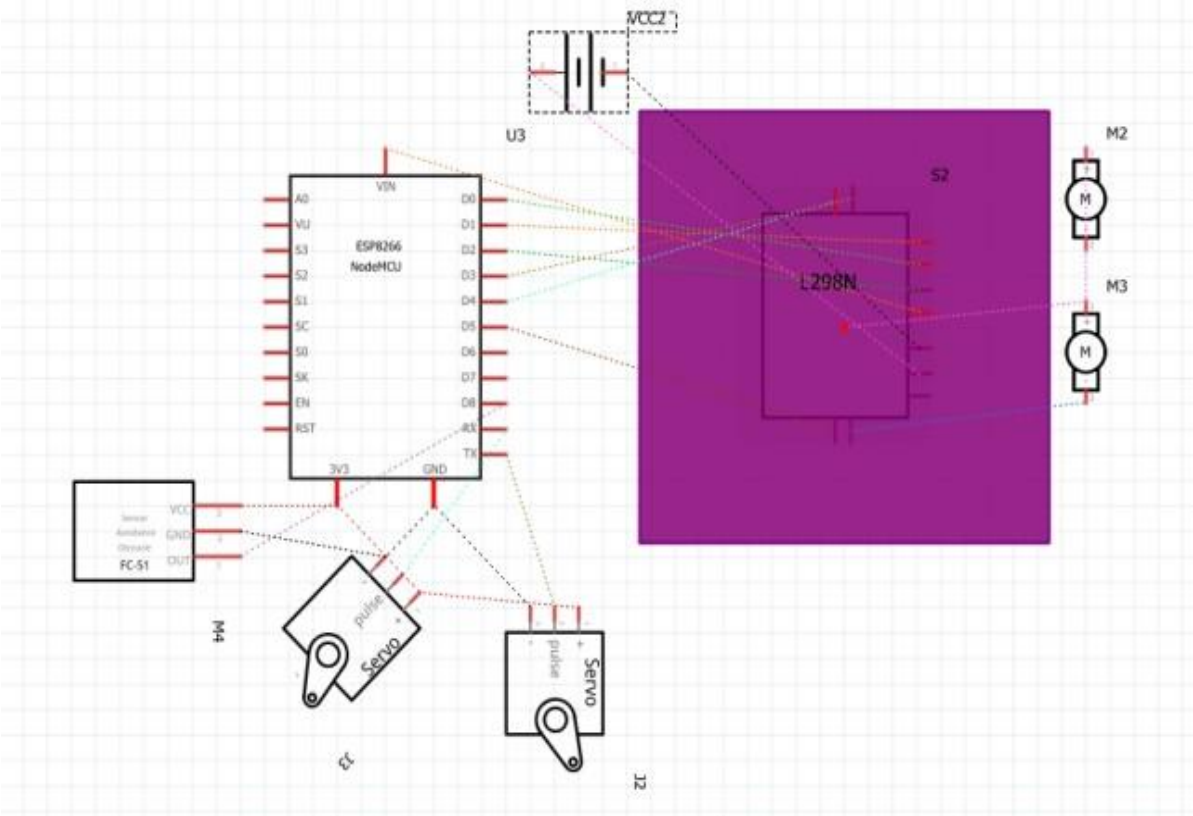

Figure 6: Schematic Circuit Diagram of System

2.5. Requirement Analysis

Microcontroller Vs Microprocessor

Feature	Microcontroller (µC)	Microprocessor (µP)
Purpose	Especially designed for specific embedded system	Especially designed for computing application
Architecture	It's single-chip computer system with I/O interfaces	It's CPU with I/O interfaces
Integration Level	Highly integrated	Less integrated
Development	IDE is provided by the manufacturer with programming and different tools.	Standard development tools and programming language like C, C++ is used.
Processing Power	Low	High
Instruction Set	Fixed	Flexible
On-board Memory	On-chip	-

Table 1: Differences between Micro-controller and Microprocessor (Geeks for Geeks, 2023)

The following hardware and software are selected for the prototype:

Device	Device Type (I/O)	Sub-Type	Measurement
PIR sensor	Input device	Active sensor	Proximity
Servo motor	Output device	Actuator	Position
BO motor	Output device	Actuator	Torque, Speed

Table 2: Type of Sensors and Actuators

Types of sensors and actuators are explained in detail in Appendix A.

Hardware Components

- **Node MCU**

  In the project, NodeMCU development board serves as the brain of the car, enabling real-time data processing and for executing the programs written.

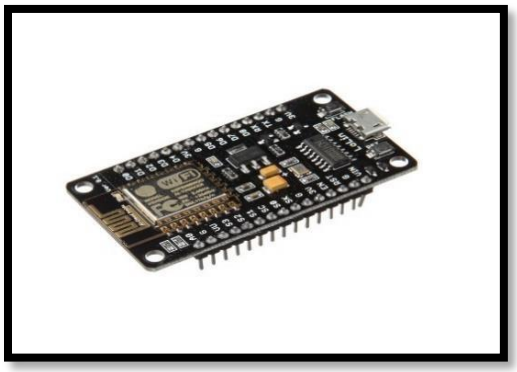

Figure 7: Node MCU (Daraz, 2024)

- **Bread Board**

  Bread Board is a handy tool to connect the components like sensors and motors. In the project, we are using a breadboard to build and evaluate our electronic circuits.

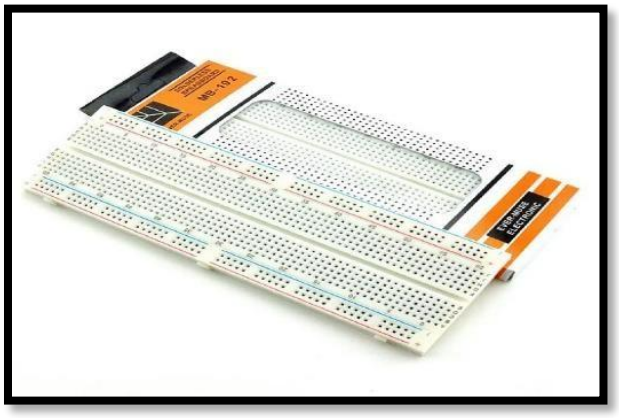

Figure 8: Breadboard (Daraz, 2024)

- **Jumper Wires**

  In our project, we are utilizing jumper wires to connect various electronic components and modules within the car's circuitry.

Figure 9: Jumper Wire (Sparkfun, 2024)

- **BO motors**

  For the battery powered prototype, BO is the excellent choice. So, we use the motor for propulsion (movement of the car in different direction).

Figure 10: BO motor (Quartz Components , 2024)

- **Servo Motors**

  In the project, Servo Motors acts as the mini robots that control the steering of the car. It will closely interact with sensors for getting the real time-data of the obstacles arising close to the car and helps in changing the directions accordingly.

Figure 11: Servo Motor (SparkFun, 2024)

- **Infrared Proximity Sensor**

  For accuracy and redundancy, infrared proximity sensor will be used. This dual-sensor approach will provide our car with robust obstacle detection capabilities, ensuring reliable navigation and collision avoidance in diverse environments and in case of sensor failures.

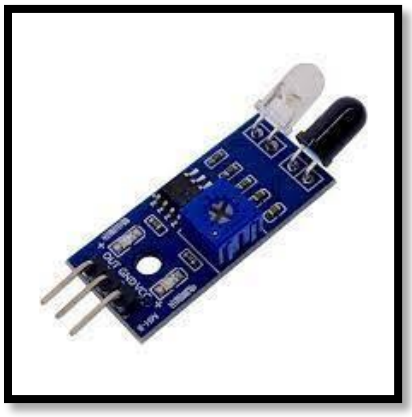

Figure 10: Infrared Proximity Sensor (ControllersTech, 2024)

- **Motor Driver**

  We are using the motor driver here for controlling the wheels as it takes signal from NODEMCU and translates into specific commands to regulate speed and direction.

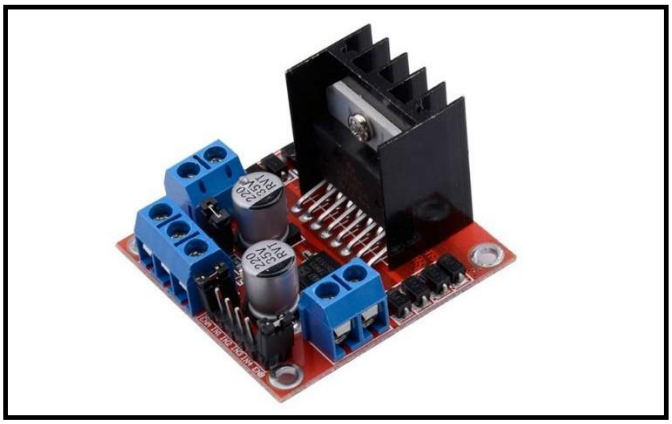

Figure 12: Motor Driver

- **Lithium Battery**

  Our project is battery powered and we have used 3.7V rechargeable lithium batteries for the project.

Figure 13: Lithium Battery

Software Components

- **Arduino IDE**

  Arduino IDE is the open-source tool which is used in our project to write the program in the micro controller for the desired functionality (Arduino , 2024).

Figure 14 : Arduino IDE (Arduino , 2024)

- **Tinkercad**

  Tinkercad is a free web app for the 3d designs, electronics, and coding. This is utilized in the project to design the circuits and simulations (Autodesk Tinkercad, 2024).

Figure 15: Tinkercad (Autodesk Tinkercad, 2024)

- **Fritzing**

  Fritzing is used in the project for creating the diagrams. It is a software application that is designed to facilitate the creation of digital charts and diagrams (Soldered team, 2023)

Figure 16: Fritzing.

3. Development

3.1. Planning and Design

Coursework released in April. After the teams were divided, we started brainstorming the whole idea of building a useful prototype. The documents released stating all the requirements for the project were taken into consideration for planning the entire project. The team wanted to design an Obstacle Avoidance Car.

After gaining some extent of technical knowledge, the team decided to work on connections between different components. For that software like Fritzing and Tinker Cad were used for creating the circuit diagram and simulating the whole connections. After the virtual experimentation of the components, we started researching about data processing and extra components for betterment.

3.2. Resource Collection

As per the planning and requirements, main components for the projects were listed. After the research, team members started searching for the required models of the components. The list was reviewed by the supervisors, and they suggested which to add and remove from the list.

The college provides the required components. The team collected all the components from the Resource Department. Since some of the components were out of stock and unavailable, the team decided to buy it from the external sources. After gathering all the materials, we started to learn how to utilize it in the best and most careful way. Now, it was time to build the prototype following pre-defined steps.

3.3. System Development

Phase I: Component Evaluation

Before starting the project, we just made sure that we had all the components required for the project. After researching the component application and features we selected the best component with required features. To know whether the components worked accurately and with their potential capacity, all the components were tested one after another, especially the sensors.

Phase II: Programming

The circuit diagram was made for reference. According to that, we started making the connections. Firstly, we connected our micro-controller, NodeMcu with the PC via USB Cable. It received power as well as the code written using Arduino IDE. Here, we coded for controlling the component as per the requirement. We spent our time debugging and finding a better way to control the components. The seamless communication between hardware was the main priority.

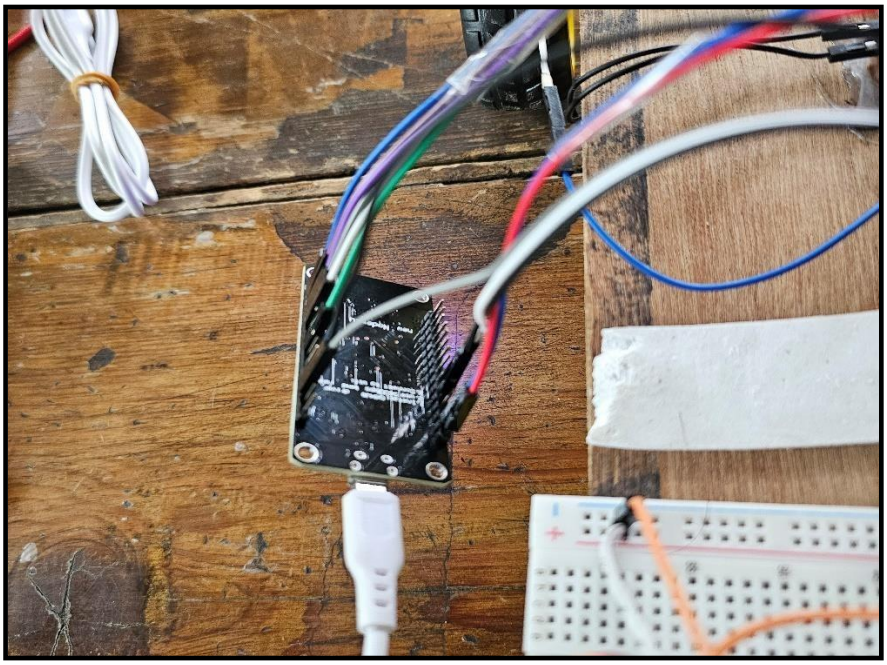

Figure 17: Node MCU programming

ArduinoConnection.ino

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <Servo.h>


#define ENA 4    // Enable/speed motors Right        GPIO4(D2)
#define ENB 14   // Enable/speed motors Left         GPIO12(D6)
#define IN_1 16  // L298N in1 motors Right           GPIO15(D8)
#define IN_2 5   // L298N in2 motors Right           GPIO13(D7)
#define IN_3 0   // L298N in3 motors Left            GPIO2(D4)
#define IN_4 2   // L298N in4 motors Left            GPIO0(D3)
#define steering 1
#define PIR 3

int pos;

Servo myservo;

String command;      //String to store app command state.
int speedCar = 800;  // 400 - 1023.
int speed_Coeff = 3;

```

Output

```
Writing at 0x00018000... (50 %)
Writing at 0x0001c000... (57 %)
Writing at 0x00020000... (64 %)
Writing at 0x00024000... (71 %)
Writing at 0x00028000... (78 %)
Writing at 0x0002c000... (85 %)
Writing at 0x00030000... (92 %)
Writing at 0x00034000... (100 %)
Wrote 307264 bytes (223667 compressed) at 0x00000000 in 19.7 seconds (effective 124.5 kbit/s)...
Hash of data verified.

Leaving...
Hard resetting via RTS pin...
```

Figure 18 : Programming

Phase III: Component Assembling

All the required components were integrated one after another in following ways:

Connection of BO motors to moto driver: In the beginning, we tested the gear motors/BO motors, then we connected each end of BO motor to motor driver in a manner that +ve and -ve terminal of a motor is connected to OUT1 and OUT2 of the motor driver and +ve and -ve of another motor is connected to OUT3 and OUT4 of the motor driver.

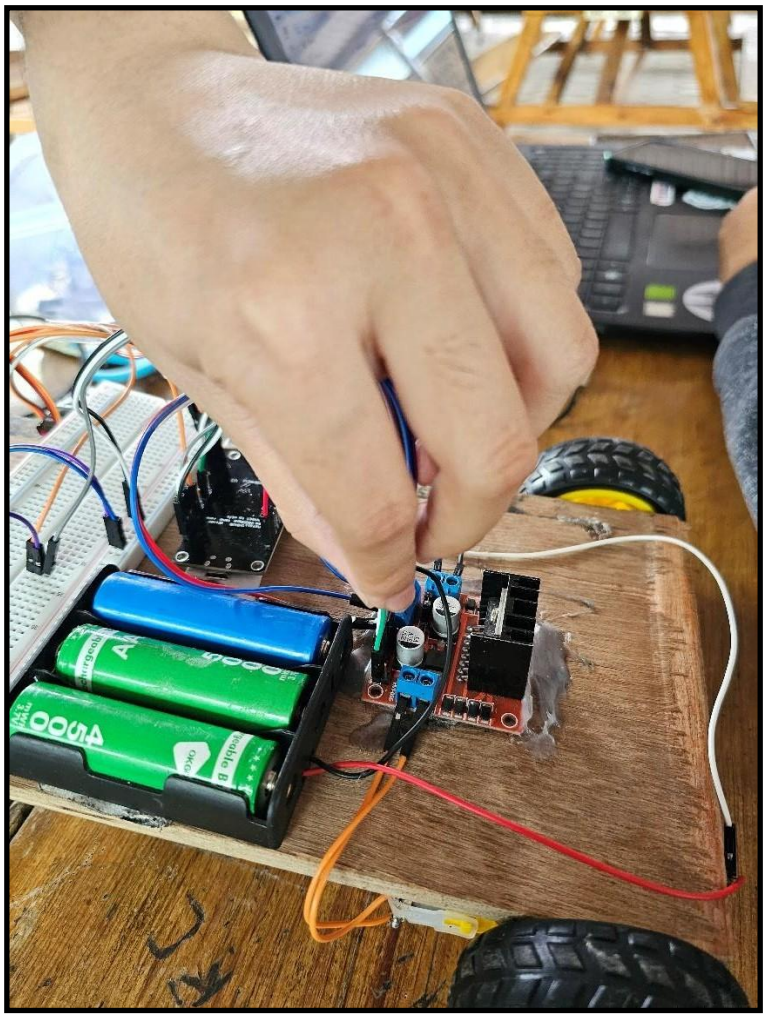

Figure 19- Connection of BO Motors to Motor Driver

Connection of NodeMCU to moto driver: For this connection, we connected the digital pins of NodeMCU to IN1, IN2, IN3 and IN4 of motor driver. Here, the analog pins of NodeMCU are also connected to the ENA and ENB of motor driver. GND pins of both were interconnected.

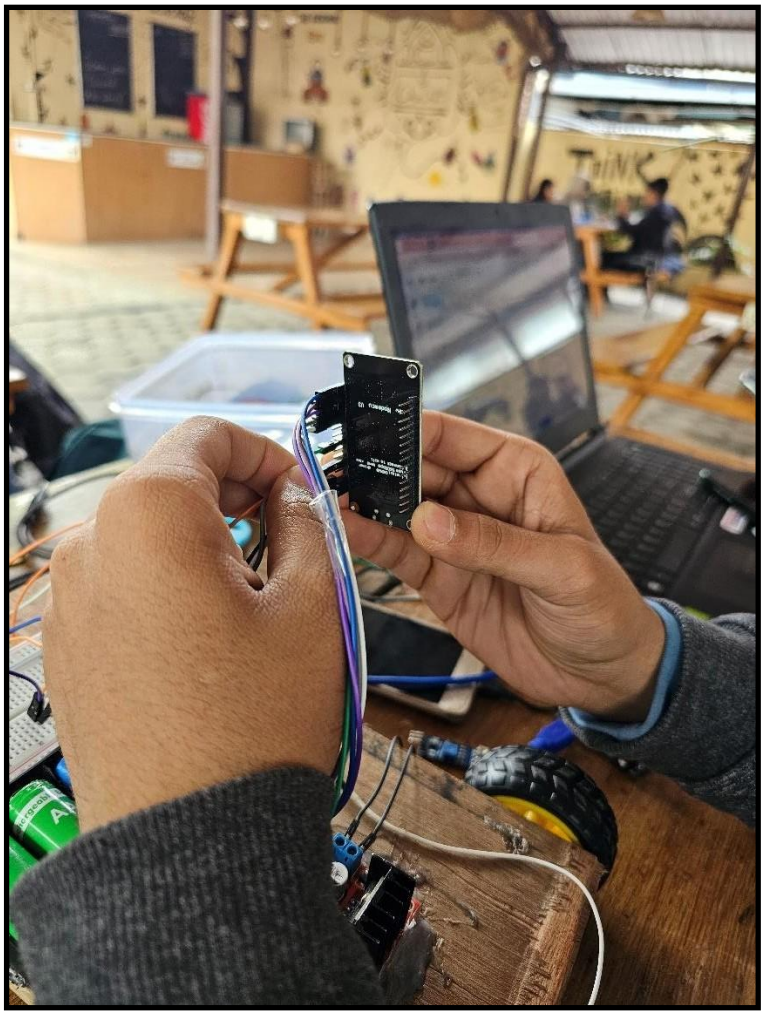

Figure 20- Connection of NodeMCU to moto driver.

Connection of Battery to motor driver: For this, we simply connected -ve terminal of battery to GND of motor driver and +ve terminal of the battery to 12V of the motor driver.

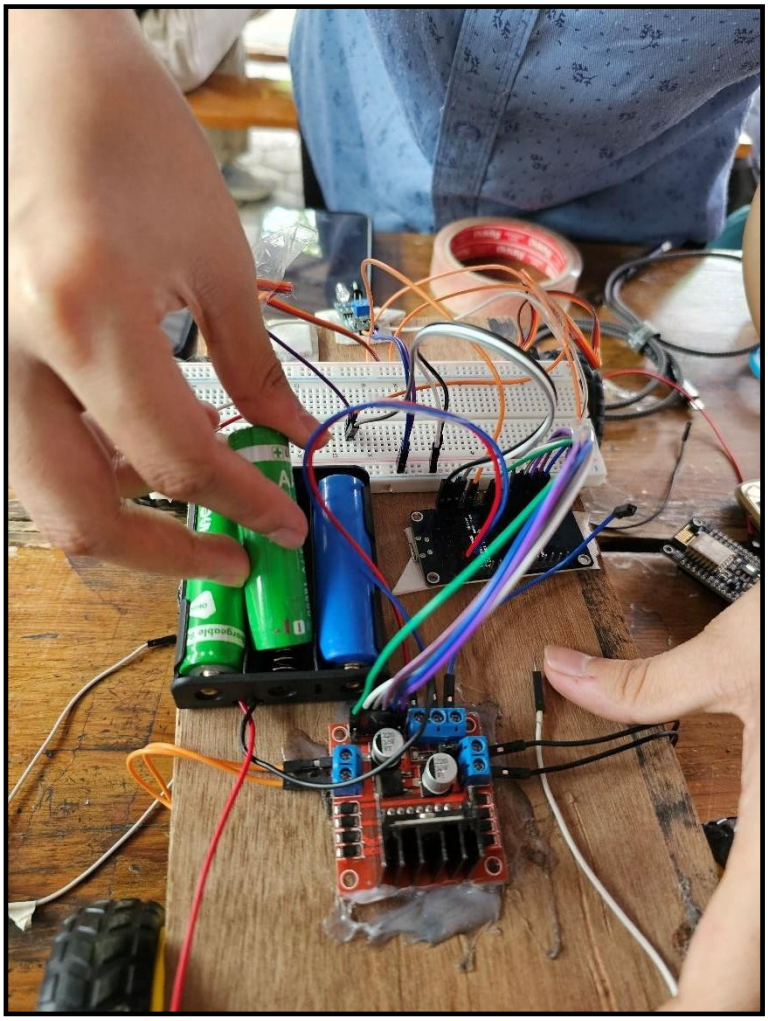

Figure 21- Connection of Battery to motor driver.

Connection of NodeMCU to servo motor: Both servomotors shared the +ve and -ve of the NodeMCU through 3V and GND pins of NodeMCU through breadboard and their data pin/input pin were connected to digital pins.

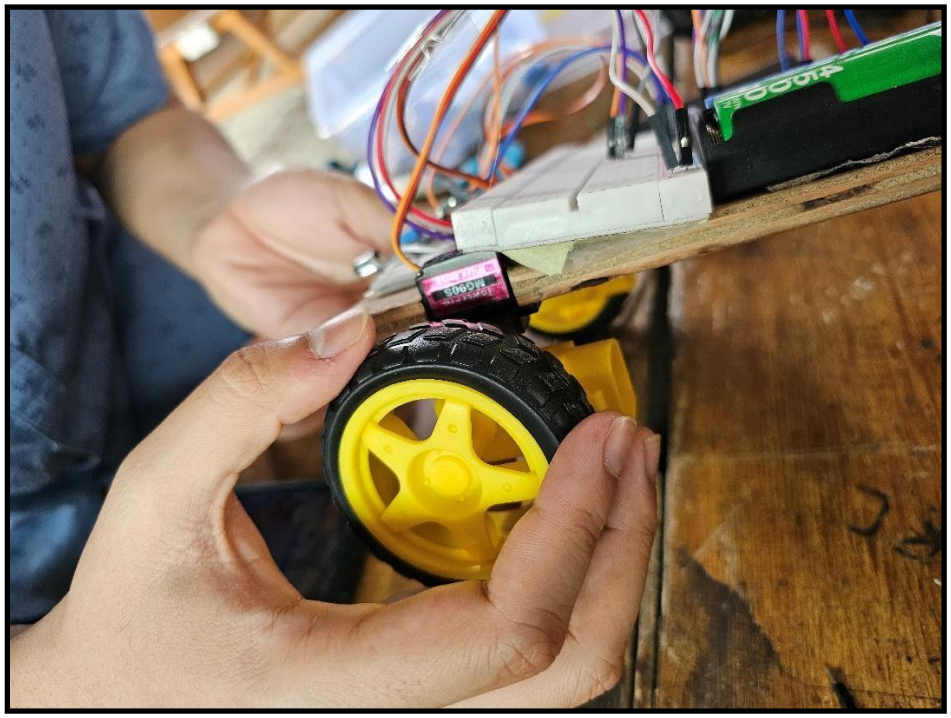

Figure 22- Connecting servo motor.

Connection of PIR to NodeMCU: The +ve and -ve of the PIR sensor was also shared with 3V and GND of NodeMCU through breadboard with servo motors and output pin of PIR sensor was connected to digital pin of NodeMCU.

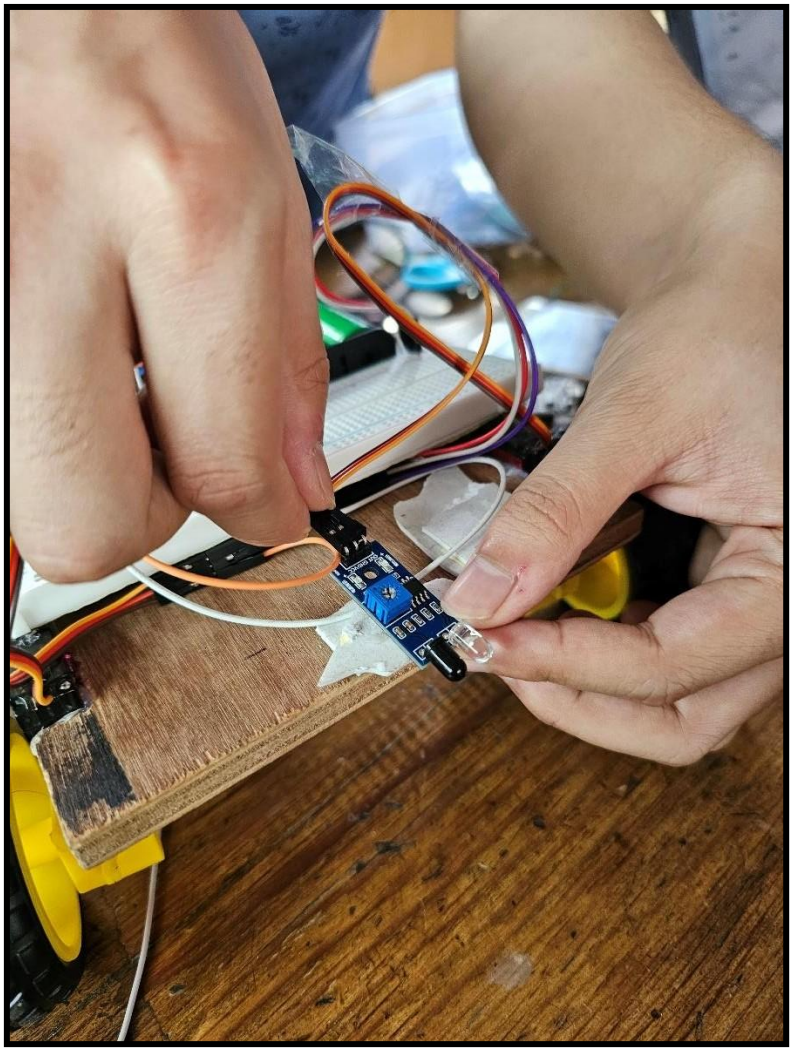

Figure 23- Connecting IR sensor.

Phase IV: Integrating Car Control Application

We have explored and used a mobile application named NodeMCU_Car. Using either the NodeMCU's Wi-Fi feature or its default IP address, we've managed to take full control over how our car moves around. The screenshot of the interface is attached below.

In this way, we have tested the overall functionality of the car.

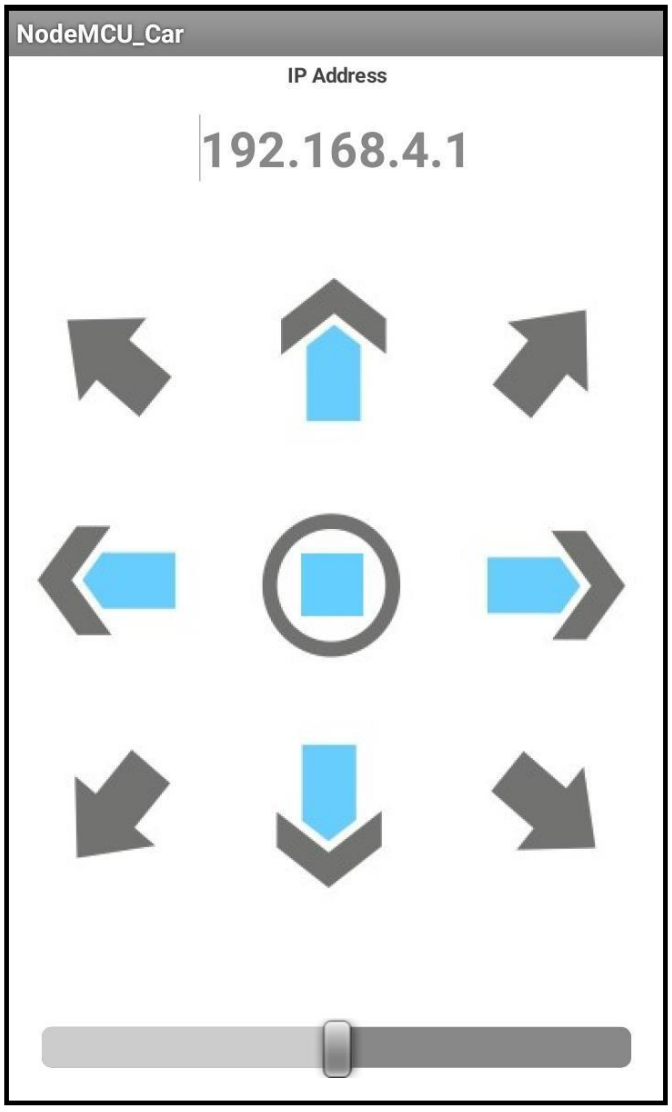

Figure 24: Smart Car controlling application.

3.3.1. PIN Connection of Component

Pins of NodeMCU	Pins of Motor Driver	Pins of Servo Motors	Pins of IR Sensors
D2	ENA		
D0	IN1		
D1	IN2		
D3	IN3		
D4	IN4		
D5	ENB		
TX		Signal pins of both Motors	
D8			OUT
VIN	7V		
GND	GND (+ve of Battery)	GND	GND
	12V (-ve of Battery)		
3V	Pins of Motor Driver	VCC	VCC

Table 3: PIN Connection of Components

4. Result and Findings

4.1. Result

The team is successful at creating the fully functional prototype of Obstacle Avoidance Car, meeting all the objectives mentioned. This obstacle avoidance car detects the obstacles and navigates safely in the urban area with the aim of enhancing the road safety and easy controlling of the car.

The system integrates sensors which capture the real-time data of the objects detected on the path, and uploads the recorded data (speed, direction) to the cloud system for further storage and analysis. Based on the algorithms designed in the onboard computer, the car's speed, direction, and movement is controlled to avoid collisions from any objects on path. The communication module integrated in the computer helps in remotely controlling the car with the help of application.

4.2. Findings

4.2.1. Test Case I

Test Case I	
Test No.	1
Objective	To check whether the NodeMCU is functional.
Test Steps Description	The codes were written for the functioning of Servo motors and uploaded it to Node MCU via Arduino IDE. After that, we connected the servo motor to Node MCU with power supply.
Expected Outcome	The code implemented on NodeMCU should enable the desired outcome through servo motor.
Actual Outcome	The code implemented on NodeMCU produced the desired outcome of the servo motor.
Conclusion	The test was successful.

Table 4: Test Case I

The proof of **Test Case I** is demonstrated using the video.

Video link: Both Servo Motor Function

4.2.2. Test Case II

Test Case II	
Test No.	2
Objective	To check whether the BO motor is functional through motor driver.
Test Steps Description	The code written for BO motor is uploaded in NodeMCU and connected to the motor driver. Now, the BO motor is connected to the moto driver.
Expected Outcome	BO motor must spin after getting the proper instruction and power supply.
Actual Outcome	BO motor rotated after getting the proper instruction and power supply.
Conclusion	The test was successful.

Table 5: Test Case II

The proof of **Test Case II** is demonstrated using the video.

Video link: Bo Motors both Forward and Backward

4.2.3. Test Case III

Test Case III	
Test No.	3
Objective	To validate if the IR sensor detects the obstacle within its range and angle and provide accurate data to NodeMCU for correct commands to motor driver.
Test Steps Description	The code for IR sensor was uploaded in Node MCU. We connected the IR to Node MCU. We observed whether the value changed when an object was placed in front of sensor.
Expected Outcome	The monitored value must change when the object is placed in front of it.
Actual Outcome	The monitored value changed when the object was placed in front of it.
Conclusion	The test was successful.

Table 6: Test Case III

The proof of **Test Case III** is demonstrated using the video.

Video link: Complete Smart Car with IR sensor working

4.2.4. Test Case IV

Test Case IV	
Test No.	4
Objective	To check whether the Smart Car meets the objectives defined for it, i.e. remote navigation, and obstacle avoidance.
Test Steps Description	With the help of the application for controlling the car, different buttons were pressed for the navigation of the car. The obstacles were placed in front of the car for it to react as per our desire.
Expected Outcome	The Smart Car must stop when the object is placed and be remotely controlled.
Actual Outcome	The Smart Car avoided the obstacle and functioned as per the instruction provided by the user.
Conclusion	The test was successful.

Table 7: Test Case IV

The proof of **Test Case IV** is demonstrated using the video.

Video link: Complete Smart Car with IR sensor working

Note: The demonstration of fully functional prototype is attached here:

Video Link: Smart_Car

5. Future Work

Our team has come a long way from not having any idea about the topic, to building a prototype of Smart Car. Now, we have potential enough to visualize the potential of our prototype to become a problem-solving Smart Car if some ideas are implemented. Some of the future work we really want to perform are listed below:

1. **Collection of enhanced sensors**: The prototype we built uses only one type of sensor which may not give accurate calculation but integration of variant of sensors can enhance the quality of Smart Car. The integration of other sensors like Ultrasonic sensor, motion sensors Cameras, etc. can make the car more versatile.
2. **Implementing advanced algorithms**: Sophisticated algorithms which help the car navigate autonomously can be built.
3. **Cloud Integration**: Cloud integration may help in better data storage and analysis. Maintenance of the software can be done remotely. Optimizing the performances can be great.
4. **Machine Learning**: Machine learning makes it easier to learn and improve the navigation capabilities which also helps in recognizing objects, predicting the movement pattern, and making good decisions for navigation.

6. Conclusion

In conclusion, the prototyping of Smart Car is a significant step forward in the field of IoT and technology. After rigorous plannings and challenging work, we are successful at creating the prototype. The prototype includes an array of technologies like IR sensors and motors. Throughout the development process we faced a lot of challenges but we troubleshooted it on our own.

Looking ahead, our project has potential to be a useful product if some changes according to trends like Machine Learning are integrated into it. Furthermore, the collaboration with industry experts was a valuable experience. The advice we received and the new things we learned from them while troubleshooting our own project was good. Furthermore, the development of prototypes not only contributed to technical advancement but also pushed our boundaries to think something creative and innovate.

Source Code:

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <Servo.h>



#define ENA 4     // Enable/speed motors Right        GPIO4(D2)
#define ENB 14    // Enable/speed motors Left         GPIO12(D6)
#define IN_1 16  // L298N in1 motors Right           GPIO15(D8)
#define IN_2 5   // L298N in2 motors Right           GPIO13(D7)
#define IN_3 0   // L298N in3 motors Left            GPIO2(D4)
#define IN_4 2   // L298N in4 motors Left            GPIO0(D3)
#define steering 1
#define PIR 15

int pos;

Servo myservo;

String command;      //String to store app command state.
int speedCar = 150;  // 400 - 1023.
int speed_Coeff = 3;

const char* ssid = "NodeMCU Car";
ESP8266WebServer server(80);

void setup() {

  Serial.begin(115200);
```

```
  pinMode(ENA, OUTPUT);
  pinMode(ENB, OUTPUT);
  pinMode(IN_1, OUTPUT);
  pinMode(IN_2, OUTPUT);
  pinMode(IN_3, OUTPUT);
  pinMode(IN_4, OUTPUT);
  myservo.attach(steering);
  pinMode(PIR, INPUT);
  myservo.write(0);

  // Connecting WiFi

  WiFi.mode(WIFI_AP);
  WiFi.softAP(ssid);

  IPAddress myIP = WiFi.softAPIP();
  Serial.print("AP IP address: ");
  Serial.println(myIP);

  // Starting WEB-server
  server.on("/", HTTP_handleRoot);
  server.onNotFound(HTTP_handleRoot);
  server.begin();
}

void goAhead() {

  digitalWrite(IN_1, HIGH);
  digitalWrite(IN_2, LOW);
  analogWrite(ENA, speedCar);
```

```
  digitalWrite(IN_3, LOW);
  digitalWrite(IN_4, HIGH);
  analogWrite(ENB, speedCar);

}

void goBack() {

  digitalWrite(IN_1, LOW);
  digitalWrite(IN_2, HIGH);
  analogWrite(ENA, speedCar);

  digitalWrite(IN_3, HIGH);
  digitalWrite(IN_4, LOW);
  analogWrite(ENB, speedCar);
}

void goRight() {
  pos = pos + 5;
  myservo.write(pos);

}

void goLeft() {
  pos = pos - 5;
  myservo.write(pos);
}


void stopRobot() {
```

```
  digitalWrite(IN_1, LOW);
  digitalWrite(IN_2, LOW);
  analogWrite(ENA, speedCar);

  digitalWrite(IN_3, LOW);
  digitalWrite(IN_4, LOW);
  analogWrite(ENB, speedCar);
}

void loop() {
  server.handleClient();
  command = server.arg("State");

  if (command == "B") goBack();
  else if (command == "L") goLeft();
  else if (command == "R") goRight();
  else if (command == "S") stopRobot();
  else if (digitalRead(PIR) == 1) {
    if (command == "F") goAhead();

  }
}

void HTTP_handleRoot(void) {

  if (server.hasArg("State")) {
    Serial.println(server.arg("State"));
  }
  server.send(200, "text/html", "");
  delay(1);
}
```