Design and Implementation of Robot Car Based on ATmega328P

by

Jamiul Kawsar
ID: CSE2103024006

Jihad Howlader
ID: CSE2001019164

Tahia Rifat
ID: CSE2103024086

Supervised by
Md. Ashfakur Rahman

Submitted in partial fulfillment of the requirements for the degree of
Bachelor of Science in Computer Science and Engineering



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SONARGAON UNIVERSITY (SU)

September 2025

APPROVAL

The project titled "**Design and Implementation of a Robot Car Based on Atmega328p**" submitted by Jamiul Kawsar (CSE2103024006), Jihad Howlader (CSE2001019164) and Tahia Rifat (CSE2103024086) to the Department of Computer Science and Engineering, Sonargaon University (SU), has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science and Engineering and approved as to its style and contents.

Board of Examiners

Md. Ashfakur Rahman
Lecturer,
Department of Computer Science and Engineering
Sonargaon University (SU)

Supervisor

(Examiner Name and Signature)
Department of Computer Science and Engineering
Sonargaon University (SU)

Examiner 1

(Examiner Name and Signature)
Department of Computer Science and Engineering
Sonargaon University (SU)

Examiner 2

(Examiner Name and Signature)
Department of Computer Science and Engineering
Sonargaon University (SU)

Examiner 3

DECLARATION

We, hereby, declare that the work presented in this report is the outcome of the investigation performed by us under the supervision of **Md. Ashfakur Rahman, Lecturer,** Department of Computer Science and Engineering, Sonargaon University, Dhaka, Bangladesh. We reaffirm that no part of this project has been or is being submitted elsewhere for the award of any degree or diploma.

Countersigned

(Md. Ashfakur Rahman)
Supervisor

Signature

Jamiul Kawsar
ID: CSE2103024006

Jihad Howlader
ID: CSE2001019164

Tahia Rifat
ID: CSE2103024086

ABSTRACT

In today's world, intelligent robotic cars play a significant role across various fields, including military operations, industrial manufacturing, warehouses, laboratories, and daily life services. These systems not only provide convenience and efficiency but also contribute significantly to economic development. With the widespread adoption of smart cars in human activities, there is an increasing demand for robotic systems capable of solving complex problems in dynamic environments. Consequently, higher performance and safety requirements are being set for intelligent vehicles, especially in hazardous scenarios such as handling explosive devices, managing radioactive substances, and conducting search-and-rescue operations. A robotic car is typically an integration of electronic, electrical, and mechanical engineering systems, designed to perform tasks autonomously or under human supervision. In such contexts, wireless remote-controlled robot cars can play a crucial role in reducing human risk while maintaining operational efficiency. This research focuses on the design and implementation of an intelligent robotic car based on the ATmega328P microcontroller. The system incorporates a Bluetooth communication module for smartphone-based control, and an ultrasonic ranging module for distance measurement. The robot combines both autonomous and manual control modes through integrated hardware, embedded control algorithms, and a four-wheel mechanical structure. Experimental results demonstrate that the smart car successfully performs functions such as Bluetooth-based smartphone-based control, real-time video streaming, obstacle avoidance, line following.

ACKNOWLEDGMENT

At the very beginning, we would like to express my deepest gratitude to the Almighty Allah for giving us the ability and the strength to finish the task successfully within the schedule time.

We are auspicious that we had the kind association as well as supervision of **Md. Ashfakur Rahman**, Lecturer, Department of Computer Science and Engineering, Sonargaon University whose hearted and valuable support with best concern and direction acted as necessary recourse to carry out our project.

We would like to convey our special gratitude to **Prof. Bulbul Ahamed**, Head of the Department, Computer Science and Engineering, Sonargaon University, for his kind concern and precious suggestions.

We are also thankful to all our teachers during our whole education, for exposing us to the beauty of learning.

Finally, our deepest gratitude and love to my parents for their support, encouragement, and endless love.

LIST OF ABBREVIATIONS

4WD	Four-Wheel Drive
AC	Alternating current
AI	Artificial Intelligence
AREF	Analog Reference
COBOL	Common Business-Oriented Language
COM	Communication
CPU	Central Processing Unit
DC	Direct Current
DIY	Do It Yourself
FORTRAN	Formula Translation
GND	Ground
GPS	Global Positioning System
IC	Integrated Circuit
ICSP	In-Circuit Serial Programmer
IDE	Integrated Development Environment
IDII	Interaction Design Institute Ivrea
IOREF	Input /Output Reference
LED	Light Emitting Diode
PC	Personal Computer
PWM	Pulse Width Modulation
RPM	Revolutions Per Minute
SCL	Serial Clock
SDA	Serial Data
UART	Universal Asynchronous Serial Receiver / Transmitter
USB	Universal Serial Bus

TABLE OF CONTENTS

Title			Page No.
DECLARATION			iii
ABSTRACT			iv
ACKNOWLEDGEMENT			v
LIST OF ABBREVIATION			vi
CHAPTER 1			1-4
INTRODUCTION			
1.1	Background and Significance		1
1.2	Research motivation		2
1.3	Research Problem		3
1.4	Design Proposal		3
1.5	Objectives		4
1.6	Structure and Content Arrangement		4
CHAPTER 2			5-30
LITERATURE REVIEW			
2.1	Background on Autonomous Robotics		5
2.2	Obstacle Avoidance Robots		5
2.3	Line Following Robots		6
2.4	Smart Cars and Advanced Research		8
2.5	Background Study of Hardware Components		9-28
	2.5.1	Arduino UNO R3	9
	2.5.2	Arduino IDE	12
	2.5.3	ESP32-CAM	15
	2.5.4	L298N Motor Driver	19
	2.5.5	Servo Motors (SG90)	25
2.6	Background on Robot Kinematics		29
2.7	Research Gap		29

CHAPTER 3 31-54
DESIGN AND IMPLEMENTATION
3.1 Overall Design Scheme and Objectives…….. 31
3.2 Smart Car Implementation……. 32-49
3.2.1 Ultrasonic Obstacle Avoidance……... 33
3.2.2 Infrared Obstacle Avoidance……... 37
3.2.3 Infrared Line Tracking……. 42
3.2.4 Bluetooth Control…… 48
3.3 Motion System Modeling …….. 49-54
3.3.1 Motion Model of the Smart Car……... 49
3.3.2 Wheel Differential Steering Model ……. 52
3.3.3 Movement Pose Analysis …… 53

CHAPTER 4 55-69
EXPERIMENTAL RESULTS AND ANALYSIS
4.1 Robot Car Assembly ……..... 55
4.2 Results and Analysis…….…… 67

CHAPTER 5 70-70
SUMMARY AND OUTLOOK
5.1 Concusion....…….…… 70
5.2 Future Works…….. 70

REFERENCES ……..…….. 71-72

LIST OF TABLES

Table No.	Title	Page No.
Table 2.1	Arduino UNO Digital Input/ Output Overview	10
Table 2.2	Arduino UNO Analog Inputs Overview	10
Table 2.3	Arduino UNO R3 Communication Overview	10
Table 2.4	Arduino Programming Overview	11
Table 2.5	Arduino IDE Toolbar Buttons	14
Table 2.6	ESP32-CAM features	16
Table 2.7	Camera Specifications	17
Table 2.8	microSD Card Reader Pin Connections	18
Table 2.9	ESP32-CAM Power Supply	19
Table 2.10	L298N Characteristics	20
Table 2.11	Direction Control illustration	22
Table 2.12	Servo Motor Specifications	28
Table 2.13	Servo Motor Advantages	28
Table 2.14	Servo Motor Applications	28
Table 3.1	Hardware Required for The Project	32
Table 3.2	Ultrasonic Avoidance Module Parameters	34
Table 3.3	Infrared Obstacle Avoidance Module Parameters	37
Table 3.4	Infrared Obstacle Avoidance Module Pin Connection	39
Table 3.5	Infrared Tracking Module Parameters	45
Table 3.6	Infrared Tracking Module Wire Connection	45
Table 3.7	Function of Bluetooth module	48
Table 3.8	Key information identification table of coordinate system	51

LIST OF FIGURES

Figure No.	Title	Page No.
Fig.2.1	SFR08 Beam and Dimensions	6
Fig.2.2	Line follower and obstacle avoidance bot using Arduino	7
Fig.2.3	Supervised Learning	8
Fig.2.4	Arduino UNO R3	9
Fig.2.5	Atmega328p Pin-out Diagram	11
Fig.2.6	Atmega328p internal architecture	12
Fig.2.7	Arduino IDE	13
Fig.2.8	Arduino IDE toolbar buttons	13
Fig.2.9	ESP32-CAM	16
Fig.2.10	ESP32-CAM features	17
Fig.2.11	ESP32-CAM Pin-out	18
Fig.2.12	Wheel	19
Fig.2.13	Arrangement of Chip Pins	20
Fig.2.14	L298N Motor Driver	21
Fig.2.15	Relation between Pulse and Voltage	21
Fig.2.16	Working of H-Bridge	22
Fig.2.17	L298N Motor Driver connection with Arduino	23
Fig.2.18	Anatomy of Servo Motor or Steering Gear	26
Fig.2.19	Servo Pins	26
Fig.2.20	Servo Motor Rotation	27
Fig.3.1	The Overall Design of Smart Car	31
Fig.3.2	Physical Diagram of the Ultrasonic Module	33
Fig.3.3	Ultrasonic working sequence diagram	35
Fig.3.4	Principle of the Distance Measurement of Ultrasonic Waves	35
Fig.3.5	Ultrasonic obstacle avoidance function program execution flow chart	36
Fig.3.6	Schematic diagram of infrared obstacle avoidance	38
Fig.3.7	detection distance adjustment	38
Fig.3.8	Data when there is no obstacle or ambient light	41
Fig.3.9	Obstacles and strong ambient light	42
Fig.3.10	Infrared tracking module physical map	43
Fig.3.11	Schematic diagram of the tracking module	44
Fig.3.12	Schematic diagram of the tracking module installation	44
Fig.3.13	When the infrared tracking module does not detect the black line data	47

Fig.3.14	Data when the infrared tracking module detects a black line	47
Fig.3.15	Bluetooth Connection	49
Fig.3.16	Schematic diagram of smart car coordinates in global coordinate system	50
Fig.3.17	Differential Steering Pose Model Diagram	53
Fig.4.1	Selecting Preference Option on Arduino IDE	55
Fig.4.2	Additional Boards Manager URLs	56
Fig.4.3	Selecting the Boards Manager Option	56
Fig.4.4	Searching for the esp32 library	57
Fig.4.5	Installing the esp32 library	57
Fig.4.6	ESP32-CAM Connection with Micro USB cable	58
Fig.4.7	Board Selection	58
Fig.4.8	WiFi Name and Password Setup	58
Fig.4.9	Code Uploading	59
Fig.4.10	Getting the IP Address	59
Fig.4.11	WiFi Connection	60
Fig.4.12	Accessing the Video Streaming Server	60
Fig.4.13	Welding wire on the motor	61
Fig.4.14	Diagram of Motor Installation	61
Fig.4.15	Mounting Wheel	62
Fig.4.16	Diagram of Connection Between Motors and Drive Board	62
Fig.4.17	Remove Short Pin from Driver	63
Fig.4.18	Mounting Motor Driver	63
Fig.4.19	The Back of Complete Tracing Module Installation	64
Fig.4.20	Infrared Obstacle Avoidance Module Mounting	64
Fig.4.21	Pan and Tilt Stand and Servo connection	65
Fig.4.22	ESP Connection with Servos	65
Fig.4.23	Ultrasonic Module Mounting with Servo	66
Fig.4.24	Mounting Arduino UNO R3	66
Fig.4.25	Complete Robot Car	67
Fig.4.26	The Robot Car after Powering Up	68

CHAPTER 1

INTRODUCTION

1.1 Background and Significance

Intelligent vehicles have become an essential component of modern technological applications, finding wide use in fields such as military operations, industrial manufacturing, logistics, life services, and research environments. This growing reliance on intelligent systems has led to increased attention from scientific and industrial communities toward the development of advanced autonomous technologies, particularly in the domain of automatic control for intelligent vehicles.

One of the critical aspects of autonomous mobility is obstacle avoidance, which enables intelligent vehicles to navigate safely in dynamic and unpredictable environments. Obstacle avoidance systems rely on various sensors to gather real-time environmental data, allowing the vehicle to perceive its surroundings and make adaptive decisions based on predefined algorithms and strategies. The integration of such systems forms the foundation for autonomous navigation and ensures safe and efficient movement in real-world applications.

As the applications of intelligent vehicles expand, so do the challenges they must address. The demand for robots capable of performing complex tasks has increased, requiring advanced algorithms, real-time processing, and robust mechanical designs. Consequently, research in obstacle avoidance technologies has intensified, leading to innovations that improve accuracy, reliability, and real-time responsiveness. Modern intelligent vehicles are no longer limited to basic automation; they now incorporate features such as multi-sensor fusion, adaptive control, and diverse operational modes to enhance flexibility and self-learning capabilities. This evolution marks the transition from traditional automated systems to intelligent, multifunctional robotic platforms capable of operating in diverse environments.

An autonomous robot is defined as a system capable of operating without human intervention in unstructured or partially unknown environments. These robots achieve autonomy through embedded software intelligence that allows them to sense their surroundings, perform path planning, detect obstacles, and make navigation decisions dynamically [1]. Common examples of autonomous robots include robotic cars, drones, humanoid robots, snake-like robots, and industrial service robots.

Among these, line-following and obstacle-avoiding robots represent fundamental systems widely used in research and industrial applications due to their practical utility and relatively simple design. Such robots typically incorporate infrared sensors for line tracking and ultrasonic sensors for obstacle detection and distance measurement. The ultrasonic sensor enables the robot to identify obstacles in its path, calculate the distance, and take appropriate action, such as stopping, turning, or rerouting.

In addition to navigation and safety, real-time video streaming has become an increasingly important feature in robotic systems. A camera-equipped robot can transmit live video from its

operational environment, enabling remote monitoring and control in hazardous or inaccessible areas. The integration of pan-tilt servo mechanisms further enhances the system's capability by allowing dynamic camera movement, thereby providing a broader and more flexible field of view for surveillance and decision-making.

The significance of such technologies extends beyond research and industrial use; they hold immense potential in the transportation sector. Autonomous driving systems equipped with obstacle avoidance and real-time vision capabilities can significantly reduce accidents, particularly on congested roads where rapid response is crucial. These systems can apply emergency brakes, navigate around obstacles, and choose safe alternative paths without human intervention. Moreover, robotic vehicles with manual and autonomous operation modes can be utilized for search-and-rescue missions, hazardous material handling, and disaster response operations, where human safety is a critical concern.

Therefore, the development of an intelligent robotic car that integrates autonomous navigation, obstacle avoidance, line following, Bluetooth-based manual control, and real-time video streaming is both timely and essential. Such a system not only demonstrates technological innovation but also provides a practical solution for a wide range of real-world applications, including automation, safety, and surveillance.

1.2 Research Motivation

The rapid progress in robotics, embedded systems, and artificial intelligence has created new opportunities for developing cost-effective autonomous vehicles. Modern self-driving technologies rely heavily on complex AI models and high-performance hardware, which, while powerful, are often too expensive for use in educational and small-scale research settings. On the other hand, low-cost robotic platforms exist but are typically limited to single functionalities, such as line following or simple obstacle avoidance, making them insufficient for demonstrating the full spectrum of autonomous navigation concepts.

This imbalance creates a strong motivation to design a system that is both affordable and multifunctional. By integrating ATmega328P-based control, ultrasonic and infrared sensors, Bluetooth wireless communication, and ESP32-CAM for video streaming, it is possible to build a robot car that not only performs line following and obstacle avoidance but also supports wireless smartphone control and real-time monitoring. Such a system would provide a bridge between basic educational robots and advanced autonomous vehicles, making robotics research more accessible to students, hobbyists, and researchers.

The motivation behind this work is therefore to create a scalable, versatile, and practical robotic platform that combines multiple autonomous features in a low-cost design. This not only demonstrates the feasibility of smart navigation systems on limited hardware but also contributes to the advancement of robotics education and affordable applied research.

1.3 Research Problem

In the past, many robotic cars were designed to perform only one specific task, such as line following or obstacle avoidance. However, this project aims to integrate a wide range of capabilities into a single, multifunctional robot car. The proposed system is a smart robot that can automatically detect and avoid obstacles, find the best path to continue its movement, and follow a target based on the selected task. It also includes real-time video streaming and camera movement to ensure accurate and detailed surveillance.

The robot can move freely and perform multiple operations based on commands received from a smartphone. While the car streams live video, the user can control its direction remotely using either Bluetooth or Wi-Fi. This design reduces the need for excessive hardware while increasing the robot's flexibility and functionality. The heart of the system is the Arduino UNO microcontroller, which connects and controls all other components, including sensors, motors, the camera module (ESP32-CAM), and wireless modules.

The user can switch between different operating modes such as line following, obstacle avoidance, and manual control by sending specific commands from the smartphone. In addition, the system offers live video surveillance to always monitor the robot's environment.

This type of multifunctional robot car can be highly useful in real-world applications. For instance, it can operate in hazardous environments where it is too risky for humans to enter such as areas with toxic gases, radiation, or explosive materials. The robot can access and monitor places that are difficult or dangerous for people, making it an ideal tool for surveillance, disaster response, and safety inspections.

1.4 Design Proposal

The control system of the intelligent robot car is primarily built around the ATmega328 microcontroller, which serves as the brain of the vehicle. It is supported by several key components that enable the car to perform multiple smart functions. An L298D motor driver is used to control the movement of the four-wheel drive system. An ESP32-CAM module is integrated into the system to provide real-time video streaming and camera control.

For wireless communication, a Bluetooth module is used, allowing the robot to receive commands from a smartphone or other compatible devices. The car is equipped with an infrared (IR) sensor for detecting objects in its path, as well as an ultrasonic sensor module (transmitter and receiver) to measure distances and avoid collisions.

Smart cars are designed to support both manual control and autonomous operation. Manual control is enabled through smartphone commands via Bluetooth, while autonomous behavior is achieved using onboard sensors and programmed logic. The entire system is mounted on a compact four-wheel chassis, and the software running on the microcontroller coordinates all the hardware components.

Experimental testing confirmed that the intelligent robot car successfully performs various tasks such as Bluetooth-based remote control, live video streaming, automatic obstacle avoidance, and automatic line tracking. These results demonstrate the effectiveness and versatility of the proposed multifunctional robotic car.

1.5 Objectives

The main objective of this project is to design and implement a Bluetooth-controlled intelligent robot car using the Atmega328 microcontroller (Arduino UNO). This robot can perform a wide range of functions, combining both autonomous and manual control. It can avoid obstacles using ultrasonic and infrared sensors, follow a path using IR line-tracking sensors, and be controlled remotely through either a Bluetooth-enabled smartphone or an infrared remote control. Additionally, the robot supports live video streaming through the ESP32-CAM module, allowing real-time monitoring of its surroundings. The system is also designed to be expandable, meaning external modules can be added to enhance its features in the future. These objectives aim to make robots a versatile and practical solution for educational, surveillance, and automation applications.

1.6 Structure and Content Arrangement

This thesis presents the complete hardware and software design of the intelligent robot car. It begins with an analysis of the importance of creating a system model for the robot, followed by an explanation of the kinematic and dynamic modeling processes. These models help in understanding how different performance factors affect the robot's behavior during operation. The entire thesis is organized into four chapters.

- Chapter 1: Introduces the intelligent robot car based on the Atmega328P microcontroller. It outlines the background, significance, research goals, and problems addressed, along with the proposed design and overall scope of the thesis.
- Chapter 2: Literature review and background studies of hardware and software components. Obstacle avoidance and line following robot’s previously work on that.
- Chapter 3: Focuses on the hardware components used in the project. It explains each component's specifications, how to set up the environment for uploading the code, the wiring connections, communication flow, and task-based experimental results, including algorithms and code explanations.
- Chapter 4: Demonstrates the physical assembly of the robot car and details the wire connections required for the system to work correctly.
- Chapter 5: Summarizes the work carried out in the project, highlights the key results and achievements, discusses current limitations, and suggests areas for improvement in future developments.

CHAPTER 2

LITERATURE REVIEW

2.1 Background on Autonomous Robotics

The rapid progress in Artificial Intelligence (AI) [1] and semiconductor technology has enabled the development of highly advanced and intelligent ground vehicles. Modern vehicles are now equipped with powerful processors that allow them to perform complex tasks, even in harsh and unpredictable environments. These advancements have made autonomous systems more reliable, accurate, and widely used.

An autonomous robot is a type of intelligent system that can make decisions and operate without direct human control. Such robots are programmed to reach specific destinations by automatically controlling the speed and direction of their wheels. Based on sensor data and decision-making algorithms, the robot determines whether to move straight, turn left, or turn right to reach the target safely.

Over the years, researchers have developed autonomous robotic systems using different technologies and methods, depending on the application. Some focused on obstacle avoidance and navigation, while others integrated features such as vision systems, line following, or wireless remote control. The main goal across these studies is to design systems that are accurate, efficient, and adaptable to different environments.

2.2 Obstacle Avoidance Robots

Obstacle avoidance is one of the fundamental capabilities of autonomous robots. Numerous researchers have focused on enhancing robot safety and efficiency in dynamic environments. For instance, Borenstein and Koren [2] provided a comprehensive classification of obstacle avoidance techniques, while Egersted [3][4] explained how different strategies can be applied depending on the task requirements.

The majority of entry-level autonomous robots rely on ultrasonic sensors for detecting obstacles, as they provide non-contact distance measurement and are inexpensive. The SFR08 ultrasonic module, shown in Fig. 2.1, is one such example widely used in robotics. Its beam pattern and sensing range demonstrate how ultrasonic technology can identify nearby objects, making it suitable for real-time obstacle detection.

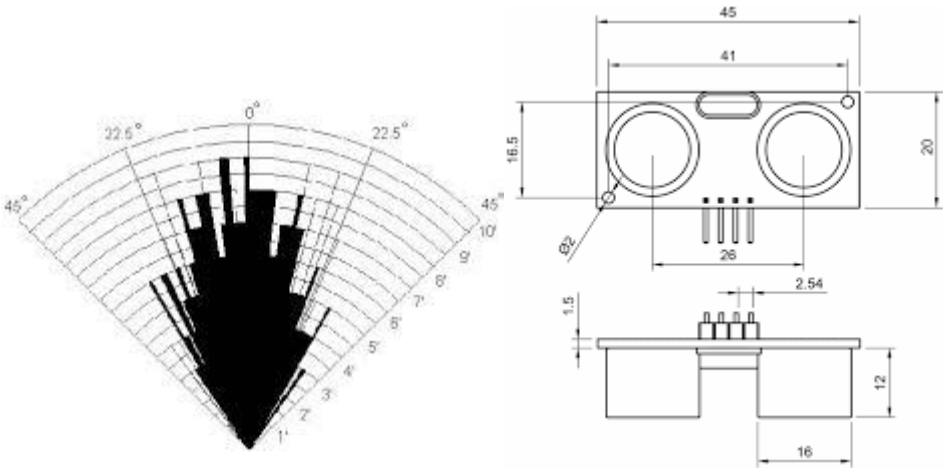


Fig 2.1: SFR08 Ultrasonic Sensor Beam Pattern and Dimensions

Beyond ultrasonic sensors, researchers have also explored alternative technologies. For example, Aniket D. Adhvaryu et al. [6] developed an obstacle-avoiding robot using infrared (IR) and passive infrared (PIR) sensors. Their work showed that PIR sensors are more effective in detecting human presence compared to IR sensors. Similarly, Vaghela et al. [7] integrated Android and Bluetooth-based control with ultrasonic sensing to create a flexible wireless obstacle avoidance robot.

2.3 Line Following Robots

mobile robotics. They are designed to follow a predetermined path, usually represented by a black or white line on a contrasting surface. These robots are widely used in industrial automation, education, and research because of their simplicity, low cost, and practical applications in material transport and service robotics.

The principle of line following is based on optical contrast detection. Typically, infrared (IR) sensors are used to distinguish between dark and light surfaces. When the sensor passes over a dark line on a light surface (or vice versa), it detects a change in reflected light intensity. This signal is processed by a microcontroller, which adjusts the wheel speeds accordingly to keep the robot aligned with the path.

The most basic line-following robots use two IR sensors, one on each side of the line. More advanced systems employ three, five, or even seven sensors to improve accuracy and responsiveness, particularly at curves and junctions. Control strategies such as proportional control, fuzzy logic, or PID (Proportional–Integral–Derivative) controllers are often integrated to minimize tracking errors [4].

Line-following robots have been successfully deployed in several domains:

- Industrial automation: Used in factories for automatic guided vehicles (AGVs) to transport materials along predefined tracks.
- Service sector: Deployed in hospitals and airports for autonomous delivery tasks.

- Education: Popular among students to learn about microcontrollers, control systems, and robotics.
- Research: Frequently combined with additional tasks such as obstacle avoidance and wireless communication [5], [6].

Aamir Attar et al. [5] proposed a robot that integrates line following and obstacle avoidance in one system. Their Arduino-based design (shown in Fig. 2.2) demonstrates how robots can intelligently decide whether to continue following a line or avoid an obstacle when one is detected.

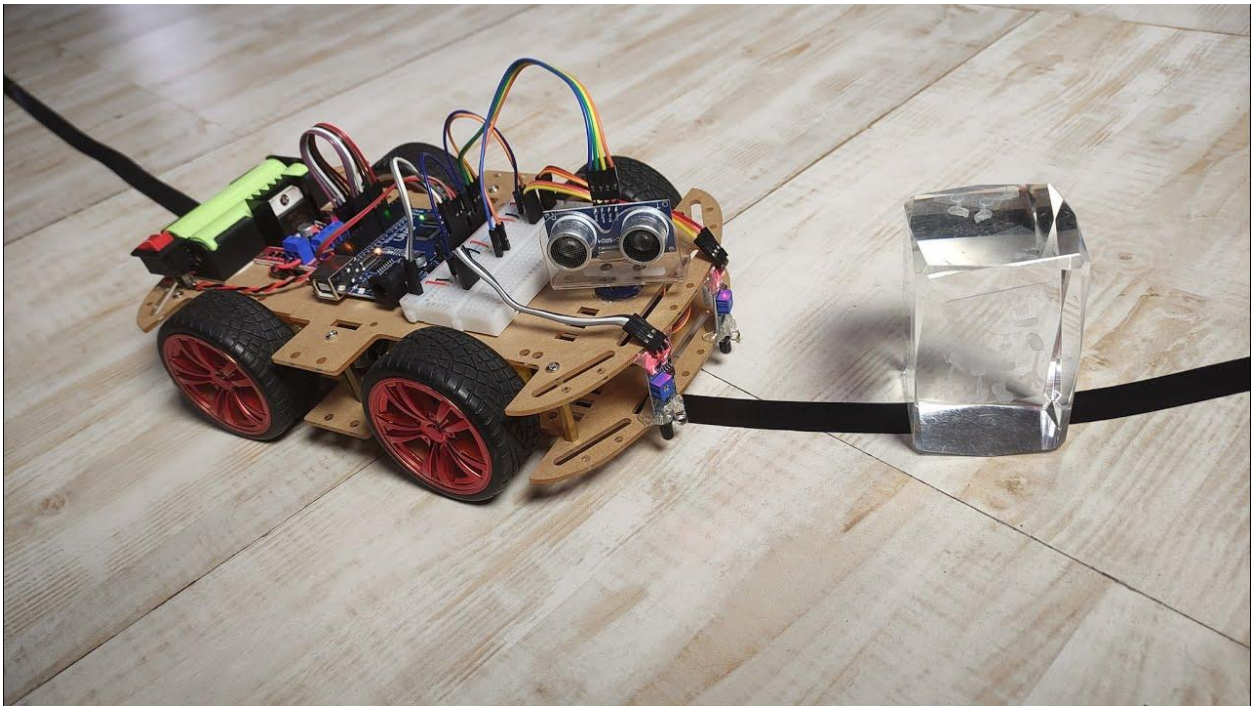


Fig. 2.2: Line follower and obstacle avoidance robot using Arduino

Aniket D. Adhvaryu et al. [6] developed a general-purpose wheeled platform that integrates line following with obstacle detection. Their work emphasized educational applications, allowing students to explore sensor integration, motor control, and microcontroller programming.

Other research has focused on improving control accuracy. For instance, some systems employed PID controllers to dynamically adjust wheel speeds and reduce oscillations [4]. Others integrated wireless control modules such as Bluetooth or Wi-Fi to enable remote monitoring and command input, extending line-following robots toward IoT-based robotics [7].

Line-following robots are a cornerstone of autonomous navigation research. While many designs achieve reliable tracking in controlled environments, they still face challenges of robustness and adaptability. Previous research [5]–[7] shows that integrating line following with obstacle avoidance, wireless communication, and vision systems can significantly improve performance. This motivates the development of more versatile platforms, such as the ATmega328P-based robot car proposed in this thesis.

2.4 Smart Cars and Advanced Research

The rapid advancement of Artificial Intelligence (AI), computer vision, and embedded systems has significantly accelerated the development of smart and autonomous cars. These vehicles extend beyond basic robotic functions such as line following and obstacle avoidance, incorporating advanced perception, decision-making, and communication capabilities. Modern self-driving cars are designed to improve safety, comfort, and efficiency, making them an essential part of future transportation systems [8].

The integration of AI into the automobile industry has enabled autonomous vehicles to make real-time decisions based on large volumes of sensor data. Techniques such as supervised learning, reinforcement learning, and deep learning are widely used for lane detection, object recognition, and path planning.

Chishti et al. [8] applied supervised learning in a self-driving car prototype. A human-driven car first collected more than 3,000 data points, which were later used to train a predictive model. Their system achieved an accuracy of 89% in predicting correct driving actions. Fig. 2.3 illustrates the process of supervised learning in this context.

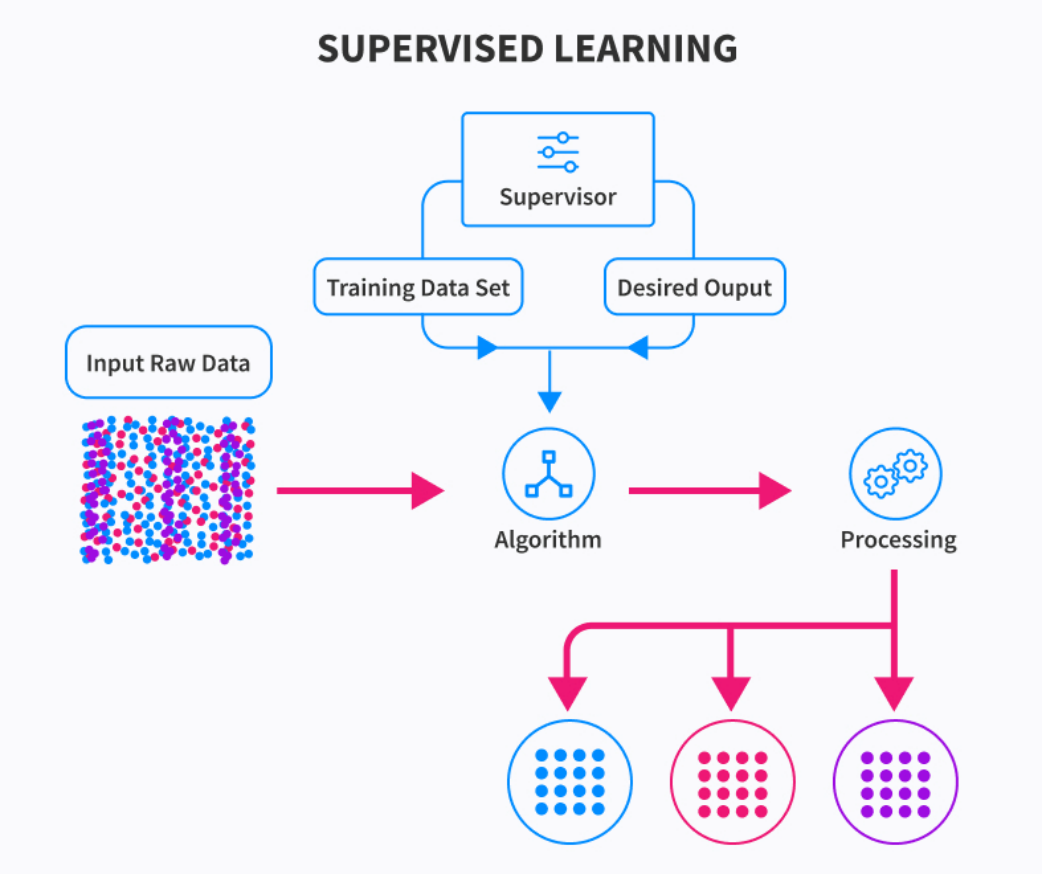


Fig. 2.3: Supervised learning approach for self-driving cars

Smart cars represent the convergence of robotics, AI, and communication technologies. While early robot cars were limited to line following or simple obstacle avoidance, recent works [8]–[10]

demonstrate the integration of machine learning, V2V communication, and hybrid sensor systems. These advancements bring autonomous vehicles closer to real-world deployment, but also highlight ongoing challenges in safety, cost, and adaptability.

The robot car developed in this thesis builds upon these research directions by integrating ATmega328P-based control, ultrasonic and IR sensing, Bluetooth wireless communication, and ESP32-CAM video streaming. This combination provides a cost-effective platform that captures the essence of smart vehicle technology at a smaller scale.

2.5 Background Study of Hardware Components

2.5.1 Arduino UNO R3

Arduino UNO R3 is a microcontroller board based on the ATmega328P. It has 14 digital input/output pins (6 can be used as PWM outputs), 6 analog inputs, a USB connection, a power jack and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started

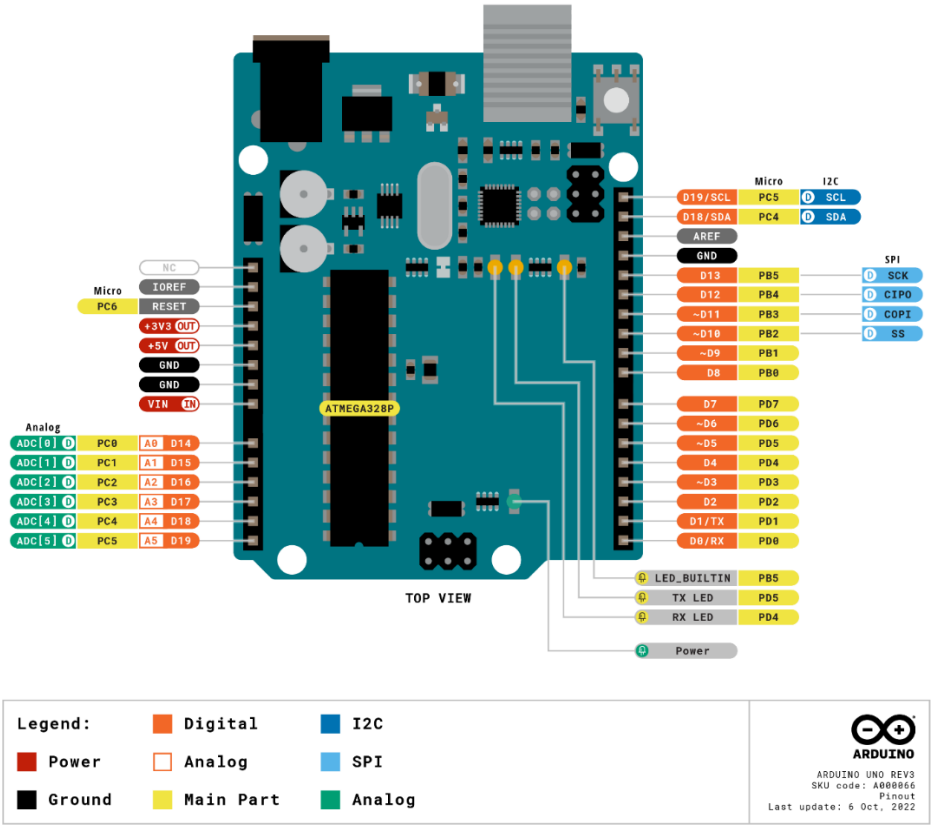


Fig.2.4: Arduino UNO R3

(1) Arduino UNO R3 Digital and Analog Pins Overview

No.	Digital Pin Overview
1	It has 14 Digital Inputs and Outputs
2	It has a logic level of 5V
3	Its Sink/Source is 20mA per pin
4	It utilizes Universal Asynchronous Serial Receiver/Transmitter (UART) for which pin 0 is for receiving and pin 1 is for transmitting
5	It has 8-bit PWM
6	It has a Serial Peripheral Interface

Table 2.1: Arduino UNO Digital Input/ Output Overview

No.	Analog Inputs Overview
1	Analog Input pins are A0, A1, A2, A3, A4, A5
2	Its sample input is from 0V to 5V, and it has a 10 bits resolution
3	The AREF pin can be used to adjust the upper limit of the input range
4	It has Two-wire Interface COM
5	Pin A4 is SDA pin for data
6	Pin A5 is SCL pin for clock

Table 2.2: Arduino UNO Analog Inputs Overview

No.	Communication Overview	
1	It can communicate with a computer, another Arduino, shields, sensors.	
2	Asynchronous communication (No clock):	
3	UART TTL (5V)	On board ATmegaU16 programmed to function as USB to serial chip
		ATmegaU16 uses standard USB COM drivers so that no external drivers are required for communication with the board
4	Synchronous communication (Clock) uses SPI Pins 10, 11, 12, 13, and TWI pins A4 andA5.	

Table 2.3: Arduino UNO R3 Communication Overview

No.	Programming Overview
1	The boot-loader firmware comes already preinstalled with the Arduino UNO
2	New sketches can be Uploaded via the USB
3	External programmer can be used to upload sketches via the In-circuit serial programmer pins (ICSP)
4	It has 32KB of Flash memory where the sketches are stored
5	It has 2KB of SRAM where the variables are stored until power cycled
6	It has 1KB of EEPROM which stores long term information.

Table 2.4: Arduino Programming Overview

(2) ATMEGA328P as Heart of Arduino

The Atmega328p is a popular microcontroller due to it being a major component in the Arduino board products. The Atmega328p is the 8-bit RISC heart of the Arduino Uno and Nano, with a maximum clock frequency of 20MHz, 32KBprogramFLASH, and 2KB of RAM. Atmega328p contains many on-board peripherals, including UART, SPI, timers, ADC, comparators, and a watchdog, and is housedina28-DIP package which enables designers to easily prototype their designs before committing to surface mount technology. With a temperature range of -40°Cto 105°Cand voltage range of 1.8V to 5.5V, the Atmega328p truly is a versatile, cost-effective microcontroller.

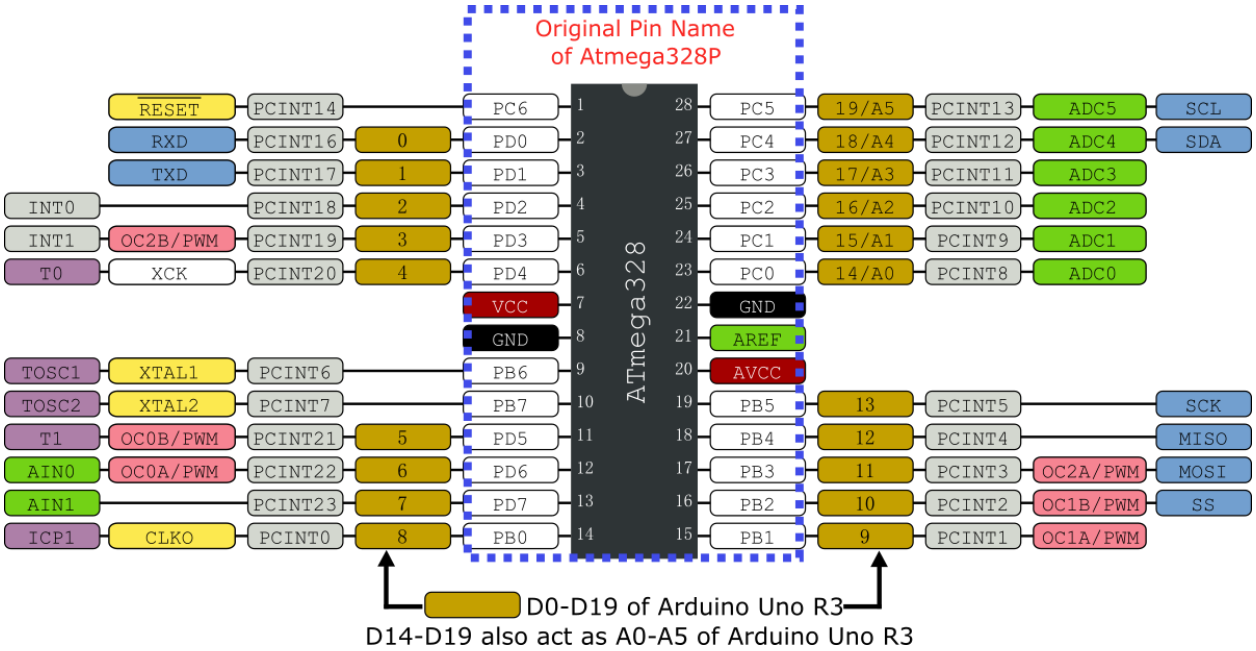


Fig. 2.5: Atmega328p Pin-out Diagram

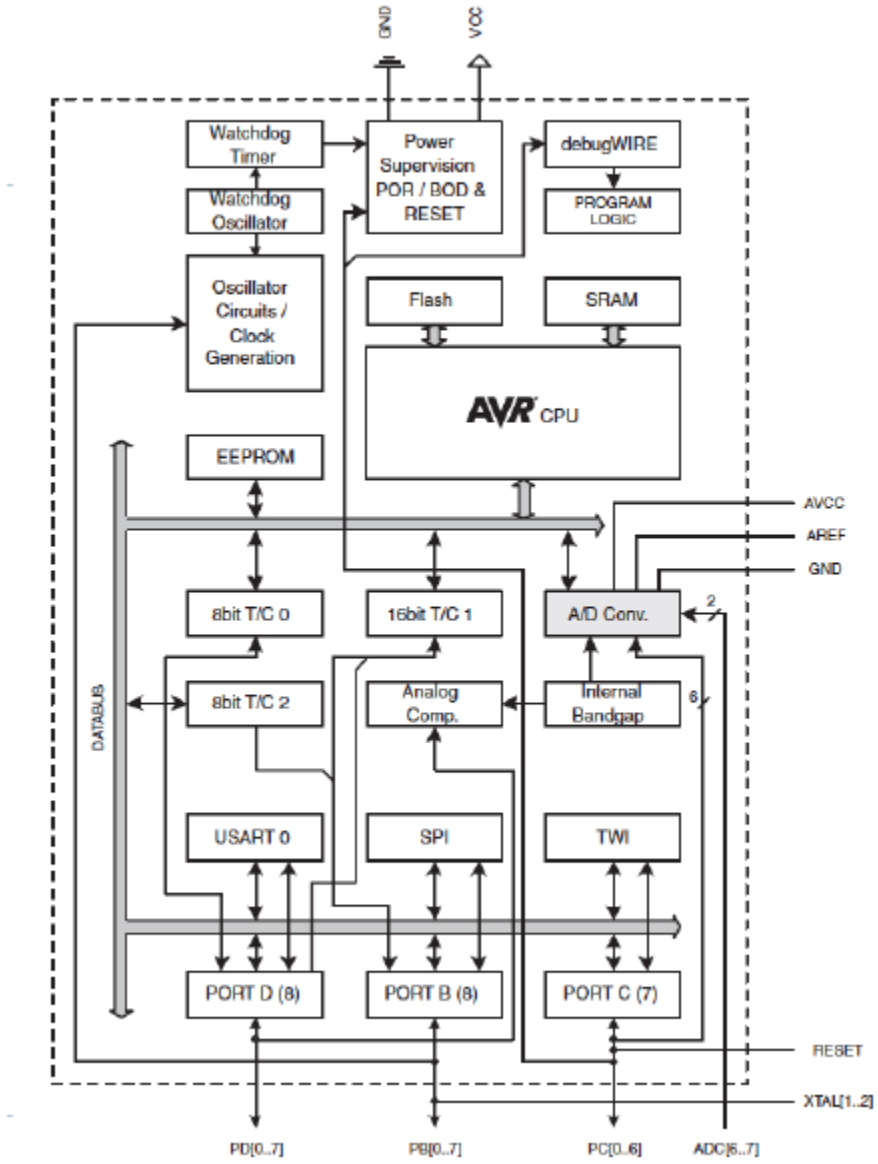


Fig.2.6: Atmega328p internal architecture

2.5.2 Arduino IDE

The Arduino Integrated Development Environment - or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them.

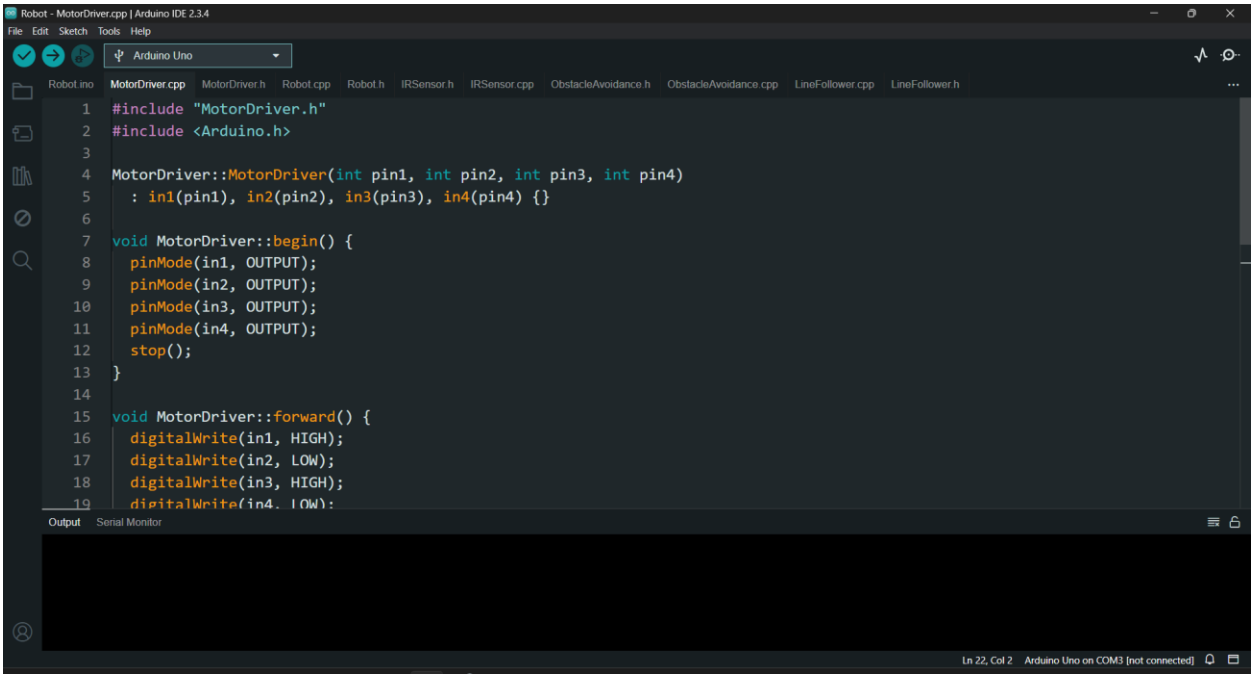


Fig.2.7: Arduino IDE

(1) Writing Sketches

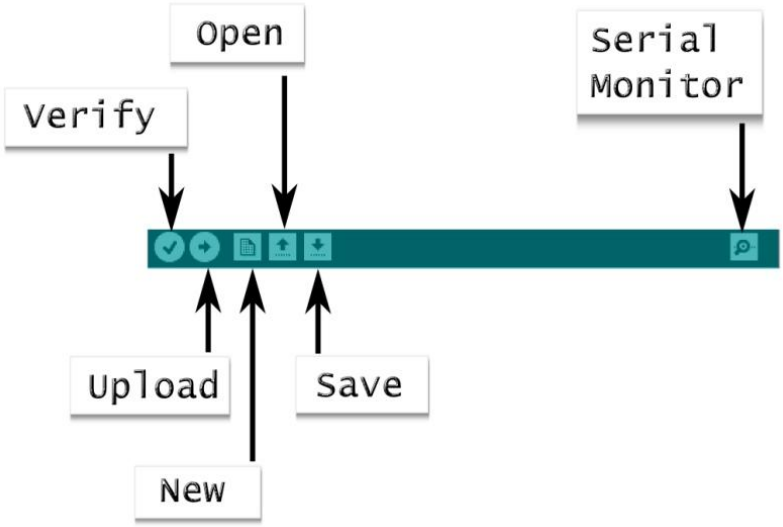


Fig.2.8: Arduino IDE toolbar buttons

Programs written using Arduino Software (IDE) are called sketches. These sketches are written in the text editor and are saved with the file extension (.ino). The editor has features for cutting/pasting and for searching/replacing text. The message area gives feedback while saving and exporting and displays errors. Console 14 displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom right-hand corner of the

window displays the configured board and serial port. The toolbar buttons allow us to verify and upload programs, create, open, and save sketches, and open the serial monitor.

No.	Toolbar
1	Verify Checks the code for errors compiling it.
2	Upload Compiles the code and uploads it to the configured board. See uploading below for details. Note: If we are using an external programmer with the board, we can hold down the "shift" key on the computer when using this icon. The text will change to "Upload using Programmer"
3	New Creates a new sketch.
4	Open Presents are a menu of all the sketches in the sketchbook. Clicking one will open it within the current window overwriting its content. Note: due to a bug in Java, this menu doesn't scroll; if we need to open a sketch late in the list, use the File | Sketchbook menu instead.
5	Save sketch.
6	Serial Monitor Opens the **serial monitor**. Additional commands are found within the five menus: File, Edit, Sketch, Tools, Help. The menus are context sensitive, which means only those items relevant to the work currently being carried out are available.

Table 2.5: Arduino IDE Toolbar Buttons

(2) Sketchbook

The Arduino Software (IDE) uses the concept of a sketchbook: a standard place to store programs (or sketches). The sketches in the sketchbook can be opened from the File > Sketchbook menu or from the Open button on the toolbar. The first time we run the Arduino software, it will automatically create a directory for the sketchbook. We can view or change the location of the sketchbook location from with the Preferences dialog.

(3) Tabs, Multiple Files, and Compilation

Allows us to manage sketches with more than one file (each of which appears inits own tab). These can be normal Arduino code files (no visible extension), C files (.cextension), C++ files (.cpp), or header files (.h). Before compiling the sketch, all the normal Arduino code files of the sketch (.ino, .pde) are concatenated into a single file following the order the tabs are shown in. The other file types are left as is.

(4) Uploading

Before uploading the sketch, we need to select the correct items from the Tools >Board and Tools > Port menus. The **boards** are described below. On the Mac, the serial port is probably something like /dev/tty.usbmodem241 (for an UNO or Mega2560 or Leonardo) or /dev/tty.usbserial-1B1 (for a Duemilanove or earlier USB board), or /dev/tty.USA19QW1b1P1.1 (for a serial board connected with a Keyspan USB-to-Serial adapter). On Windows, it's probably COM1 or COM2 (for a serial board) or COM4, COM5, COM7, or higher (for a USB board) - to find out, we look for USB serial device in the ports section of the Windows Device Manager. On Linux, it should be /dev/ttyACMx , /dev/ttyUSBx or similar. Once we've selected the correct serial port and board, press the upload button in the toolbar or select the Upload item from the Sketch menu. Current Arduino boards will reset automatically and begin the upload. With older boards (pre-Diecimila) that lack auto-reset, we'll need to press the reset button on the board just before starting the upload. On most boards, we'll see the RX and TX LEDs blink as the sketch is uploaded. The Arduino Software (IDE) will display a message when the upload is complete or shows an error.

When uploading a sketch, to use the Arduino bootloader, a small program that has been loaded on to the microcontroller on the board. It allows us to upload code without using any additional hardware. The bootloader is active for a few seconds when the board resets; then it starts whichever sketch was most recently uploaded to the microcontroller. The bootloader will blink the on-board (pin 13) LED when it starts (i.e. when the board resets).

(5) Serial Monitor

This displays serial sent from the Arduino board over USB or serial connector. To send data to the board, enter text and click on the "send" button or press enter. Choose the baud rate from the drop-down menu that matches the rate passed to Serial.begin in the sketch. Note that on Windows, Mac or Linux the board will reset (it will rerun the sketch) when we connect with the serial monitor. Please note that the Serial Monitor does not process control characters; if we sketch needs complete management of the serial communication with control characters, we can use an external terminal program and connect it to the COM port assigned to the Arduino board. We can also talk to the board from Processing, Flash, MaxMSP, etc.

2.5.3 ESP32-CAM

We are going to use ESP32-CAM module as the eye for the robot car. We will make use of the ESP32CAMs built-in WiFi capabilities to program a web-based user interface that will allow us to control flashlight, move the camera in different directions, and view the video camera output. ESP32-CAM has a very competitive small-sized camera module in the industry. It can work independently as the smallest system, with a size of only 27*40.5*4.5mm, and a deep sleep current as low as 6mA.ESP-32CAM can be widely used in various IoT applications, home smart devices, industrial wireless control, wireless monitoring, QR wireless identification, wireless positioning system signals, and other IoT applications.ESP-32CAMadopts DIP sealed package and can be

used directly by the plug-in, realizing the rapid production of products, providing customers with high-reliability connection methods, in support for various IoT hardware terminal occasions.

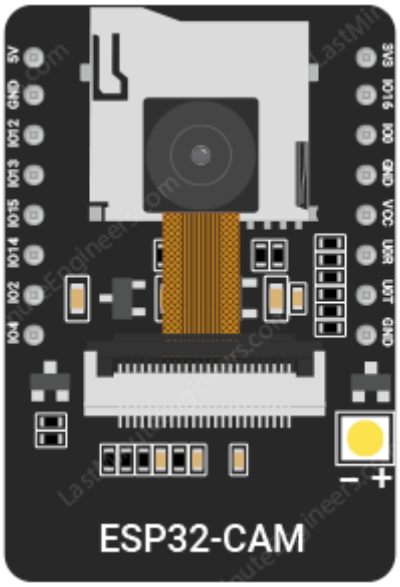


Fig.2.9: ESP32-CAM

No.	ESP32-CAM features
1	The smallest 802.11b/g/n Wi-Fi BT SoC module
2	Adopt a low-consumption, double core 32-bit CPU, which can be taken as an application processor.
3	The main frequency is up to 240MHz, and the computing power reaches 600 DMIPS.
4	Built-in 520 KB SRAM, external 8MB PSRAM.
5	Compatible with UART/SPI/I2C/PWM/ADC/DAC ports.
6	Support OV2640 and OV7670 cameras, with a built-in flashing light.
7	Support TF card, multiple sleep modes, WIFI upload, and STA/AP/STA+AP working mode.
8	Built-in Lwip and FreeRTOS.
9	Support Smart Config/AirKiss smart config
10	Support second development.
11	Support for serial port local and remote firmware upgrades (FOTA)

Table 2.6: ESP32-CAM features

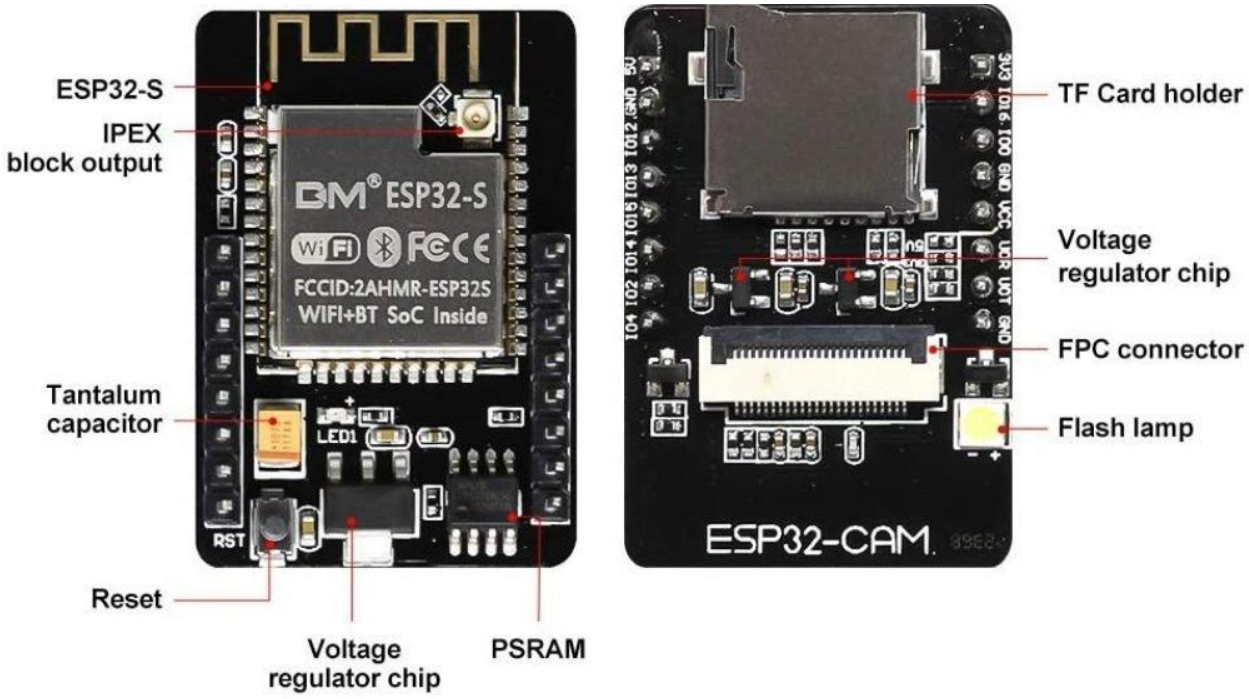


Fig.2.10: ESP32-CAM features

Camera Specifications:

The ESP32-CAM includes an OV2640 camera module. The device also supports OV7670 cameras. The OV2640 has the following specifications:

No.	Camera Specifications
1	2 Megapixel sensor
2	Array size UXGA 1622×1200
3	Output formats include YUV422, YUV420, RGB565, RGB555 and 8-bit compressed data
4	Image transfer rate of 15 to 60 fps

Table 2.7: Camera Specifications

ESP32-CAM Pin-out:

The following figure shows the ESP32-CAM pin-out (AI-Thinker module).



Fig.2.11: ESP32-CAM Pin-out

There are three **GND** pins and two pins for power: either **3.3V** or **5V**. **GPIO 1** and **GPIO** 3 are serial pins. We need these pins to upload code to the board. Additionally, GPIO 0 also plays an important role, since it determines whether theESP32 is in flashing mode or not. When GPIO 0 is connected to **GND**, the ESP32 is in flashing mode.

The following pins are internally connected to the microSD card reader:

No.	microSD card reader pin connections
1	GPIO 14: CLK
2	GPIO 15: CMD
3	GPIO 2: Data 0
4	GPIO 4: Data 1 (also connected to the on-board LED)
5	GPIO 12: Data 2
6	GPIO 13: Data 3

Table 2.8: microSD Card Reader Pin Connections

Power Supply:

Of course, our robot car isn't going anywhere without a source of power! Need to supply two voltages for the car:

No.	ESP32-CAM Supply Voltages
1	Power for the ESP32CAM – Although the module works on 3.3-volts it also has a 5-volt power input, and many experimenters have discovered that this is more reliable than the 3.3- volt one. So I suggest that we allow for 5-volts, with a decent amount of current capability as the WiFi connection requires occasional bursts of power.
2	Power for the Motors – This voltage level will depend upon the motors of course. The main factors here are that it (a) needs to be independent from the ESP32CAMpower supply and(b) needs to have sufficient current to power the motors.

Table 2.9: ESP32-CAM Power Supply

For my robot, I designed a small power supply using an adjustable buck converter and powered by a pair of 18650 batteries. The buck converter supplies the voltage for the ESP32CAM, and the motors are driven right from the batteries (7.4- volts total).

2.5.4 L298N Motor Driver



Fig.2.12: Wheel

Small and cheap DC motor with a gearbox ratio of 1: 120. The gearbox has a reduction in speed, which gives the engine much more power. These engines are ideal for robot platforms, such as robot cars. They fit well with the rubber wheel of 65mm. The motors have an input voltage between 3V and 7.5V, with a higher voltage giving more output. This motor has a return axis on which a rotary encoder can be installed in order to read how the motor moves with high resolution.

The motor drive is a necessary condition for the motor to play its superior performance. Its main function is to provide sufficient current and power for the motor. In the car, we choose the L298N as the motor driver chip because it is a high voltage and current full-bridge driver chip, the chip uses 15 pins package. It is a special motor driving integrated circuit (two H bridges) with a high voltage and current full-bridge driver. And it contains 4 channel logic drive circuit, which belongs to a kind of two-phase and four-phase special motor drive which contains two Hbridges of high voltage large current. The output current is 2A, the maximum current is 4A, and the maximum

working voltage is 50V, which can drive the load under 46V and 2A, such as high-power DC motor, stepper motor, solenoid valve, and so on. The chip with two enabled control terminals uses the standard logic level to control signals, allows or prohibits the device to work when the input signal does not interfere, it has a logic power input terminal that can enable the internal logic circuit to work under low voltage, and feedback the variation to the control circuit. Especially, the input can be connected directly with the MCU and easily controlled. When the DC motor is driven, the stepper motor can be directly controlled, and it can be turned forward and reversely, which only needs to change the logic level of the input. The pin arrangement is shown in Fig. 2.13, Pins 1 and 15 can separately connect to the current sampling resistor and form the current sensing signal.

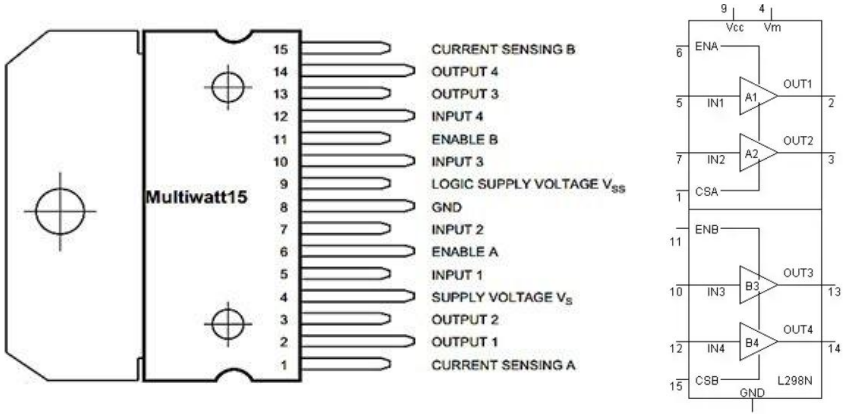


Fig.2.13: Arrangement of Chip Pins

L298N can drive 2 motors that are connected between OUT1, OUT2 and OUT3, OUT4. 5, 7, 10, and 12 pins are connected to the input control level for controlling the positive and negative rotation of the motor, ENA, ENB are connected to the control enable terminal for controlling the running and shutdown of the motor.

No.	Characteristics
1	Signal indicator
2	The speed is adjustable
3	The strong anti-interference ability with photoelectric isolation
4	Over-voltage and over-current protection
5	Controlling two motors separately
6	Controlling the stepper motor
7	The speed control with PWM pulse width modulation
8	Positive and negative rotation

Table 2.10: L298N Characteristics

(1) Controlling a DC Motor

To have complete control over the DC motor, we have to control its speed and rotation direction. This can be achieved by combining these two techniques.

(1) PWM – For controlling speed
(2) H-Bridge – For controlling rotation direction

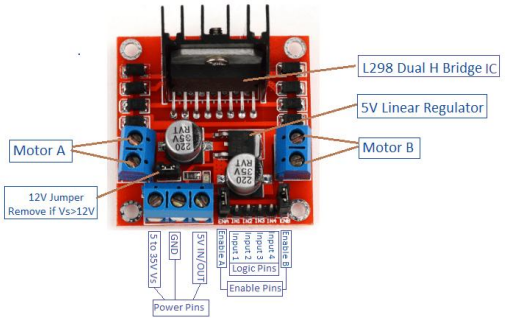


Fig.2.14: L298N Motor Driver

(2) PWM For controlling speed

The speed of a DC motor can be controlled by varying its input voltage. A common technique for doing this is to use PWM (Pulse Width Modulation). PWM is a technique where the average value of the input voltage is adjusted by sending series of ON-OFF pulses. The average voltage is proportional to the width of the pulses known as the Duty Cycle. The higher the duty cycle, the greater the average voltage being applied to the dc motor (High Speed), and the lower the duty cycle, the less the average voltage being applied to the dc motor (Low Speed).

The image below illustrates the PWM technique with various duty cycles and average voltages.

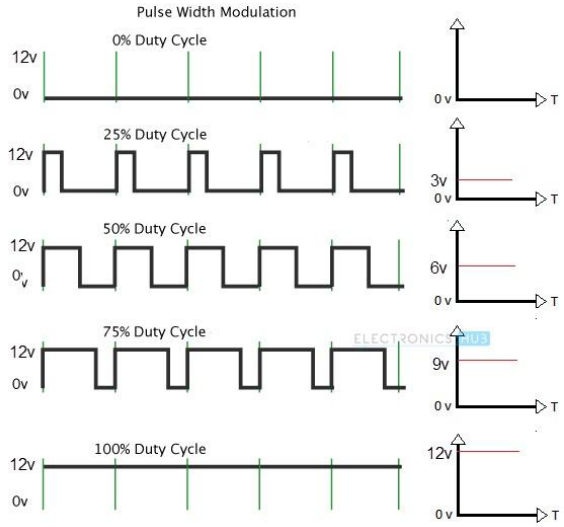


Fig. 2.15: Relation between Pulse and Voltage

(3) Control Pins

Using the direction control pins, we can control whether the motor spins forward or backward. These pins control the switches of the H-Bridge circuit inside L298N IC. The module has two direction control pins for each channel. The IN1 andIN2 pins control the spinning direction of motor A while IN3 and IN4 control motor B. The spinning direction of a motor can be controlled by applying either logic HIGH (5 Volts) or logic LOW(Ground) to these inputs. The chart below illustrates how this is done.

Input1	Input2	Spinning Direction
Low (0)	Low (0)	Motor OFF
High (1)	Low (0)	Forward
Low (0)	High (1)	Backward
High (1)	High (1)	Motor OFF

Table 2.11: Direction Control illustration

(4) H-Bridge – For controlling rotation direction

The DC motor's spinning direction can be controlled by changing the polarity of its input voltage. A common technique for doing this is to use an H-Bridge. An H-Bridge circuit contains four switches with the motor at the center forming an H-like arrangement. Closing two switches at the same time reverse the polarity of the voltage applied to the motor. This causes a change in the spinning direction of the motor.

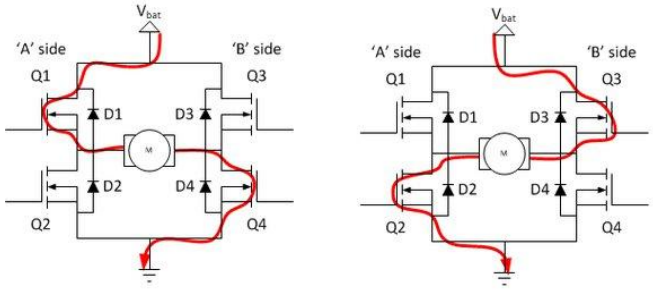


Fig. 2.16: Working of H-Bridge

(5) L298N connection with Arduino

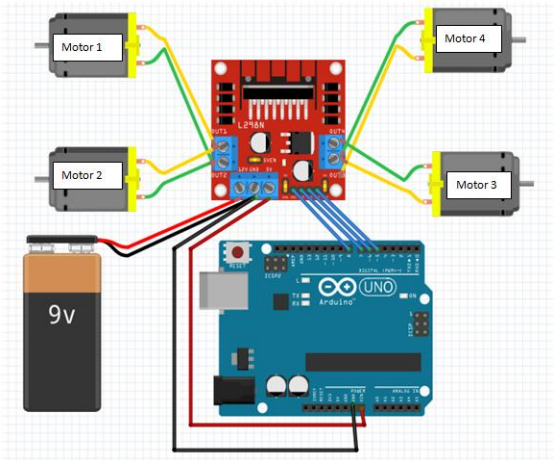


Fig. 2.17: L298N Motor Driver connection with Arduino

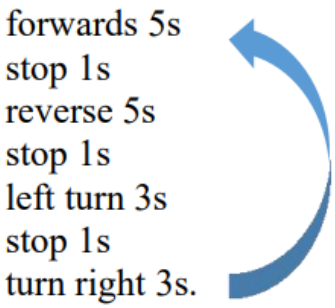
(6) Motor Direction Check

```
int IN1_PIN = 6;
int IN2_PIN = 10;
int IN3_PIN = 5;
int IN4_PIN = 9;
void setup() {
Serial.begin(9600);
pinMode(IN1_PIN, OUTPUT);
pinMode(IN2_PIN, OUTPUT);
pinMode(IN3_PIN, OUTPUT);
pinMode(IN4_PIN, OUTPUT);
}
void loop() {
digitalWrite(IN1_PIN, HIGH);
digitalWrite(IN2_PIN, LOW);
```

```
digitalWrite(IN3_PIN, LOW);
digitalWrite(IN4_PIN, HIGH);
delay(5000);
//**************************//Forward
digitalWrite(IN1_PIN, LOW);
digitalWrite(IN2_PIN, LOW);
digitalWrite(IN3_PIN, LOW);
digitalWrite(IN4_PIN, LOW);
delay(1000);
//*****************//Stop
digitalWrite(IN1_PIN, LOW);
digitalWrite(IN2_PIN, HIGH);
digitalWrite(IN3_PIN, HIGH);
digitalWrite(IN4_PIN, LOW);
delay(5000);//*****************//Back
digitalWrite(IN1_PIN, LOW);
25
digitalWrite(IN2_PIN, LOW);
digitalWrite(IN3_PIN, LOW);
digitalWrite(IN4_PIN, LOW);
delay(1000);
//**************************//Stop
digitalWrite(IN1_PIN, HIGH);
digitalWrite(IN2_PIN, LOW);
digitalWrite(IN3_PIN, HIGH);
digitalWrite(IN4_PIN, LOW);
delay(3000);
//**************************//Left
```

```
digitalWrite(IN1_PIN, LOW);
digitalWrite(IN2_PIN, LOW);
digitalWrite(IN3_PIN, LOW);
digitalWrite(IN4_PIN, LOW);
delay(1000); //****************//Stop
digitalWrite(IN1_PIN, LOW);
digitalWrite(IN2_PIN, HIGH);
digitalWrite(IN3_PIN, LOW);
digitalWrite(IN4_PIN, HIGH);
delay(3000);
//****************//Right
}
```

After uploading the code turn on the power to observe the wheel rotation, if it appears in a loop as below:



Caution: If the motor does not change the line to check if the contact is poor, the battery is fully charged (6.5V or more), If the car is not following the procedure, please turn off the L298N driver board.

2.5.5 Servo Motor (SG90)

The servo motor is also called steering gear. It was first used to realize its steering function on the ship. Servo motor is a position control rotary actuator. It mainly consists of housing, circuit board, core-less motor, gear, and position sensor. Its working principle is that the servo receives the signal sent by MCU or receiver and produces a reference signal with a period of 20ms and width of

1.5ms, then compares the acquired DC bias voltage to the voltage of the potentiometer and outputs a voltage difference.

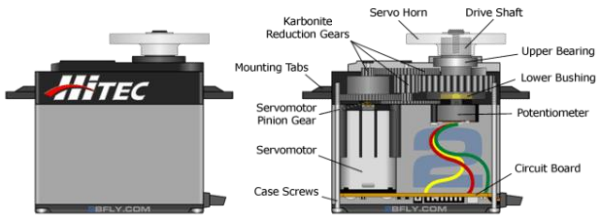


Fig. 2.18: Anatomy of Servo Motor or Steering Gear

Servo motor comes with many specifications. But all of them have three connection wires, distinguished by brown, red, and orange colors (different brands may have a different color). The Brown one is for GND, redone for power positive, and the orange one is for the signal line.



Fig. 2.19: Servo Pins

The rotation angle of the servo motor is controlled by regulating the duty cycle of the PWM (Pulse-Width Modulation) signal. The standard cycle of the PWM signal is 20ms (50Hz). Theoretically, the width is distributed between 1ms-2ms, but in fact, it's between 0.5ms-2.5ms. The width corresponds to the rotation angle from0° to 180°. But note that for different brand motors, the same signal may have a different rotation angle.

The control signal enters the signal modulation chip from the channel at the receiving end to obtain a DC bias voltage. It has a reference circuit inside, which generates a reference signal with a period of 20ms and a width of 1.5 milliseconds, and compares the obtained DC bias voltage with the voltage of the potentiometer to obtain a voltage difference output. Finally, the positive and negative voltage output of the voltage difference to the motor drive chip determines the forward and reverse of the motor. When the motor speed is constant, the potentiometer is rotated by the cascade reduction gear so that when the voltage difference is 0, the motor stops rotating.

When the control circuit board receives the control signal from the signal line, the motor is controlled to rotate, and the motor drives a series of gear sets to decelerate and then drives to the

output steering wheel. The output shaft of the steering gear and the position feedback potentiometer are connected. When the steering wheel rotates, the position feedback potentiometer is driven. The potentiometer will output a voltage signal to the control circuit board for feedback, and then the control circuit board determines the motor according to the position. The direction and speed of rotation to achieve the target stop.

The workflow is: control signal → control circuit board →motor rotation→gear set deceleration → steering wheel rotation → position feedback potentiometer →control circuit board feedback.

The control signal period of the servo is 20MS pulse width modulation (PWM) signal, wherein the pulse width is from 0.5-2.5MS, and the corresponding steering wheel position is 0-180 degrees, which varies linearly. Give him a certain pulse width, its output shaft will maintain a certain corresponding angle, no matter how the external torque changes, until it is given a pulse signal of another width, it will change the output angle to the new. The corresponding position is shown in Fig. 2.18. There is a reference circuit inside the steering gear to generate a reference signal with a period of 20MS and a width of 1.5MS. There is a comparator that compares the applied signal with the reference signal to determine the direction and size to produce the motor's rotation signal.

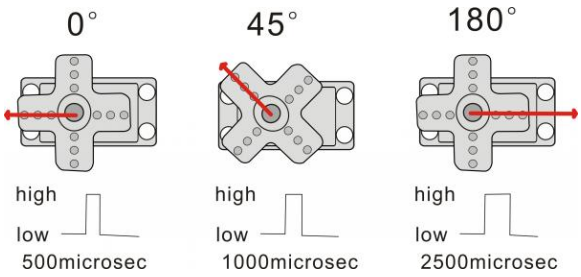


Fig. 2.20: Servo Motor Rotation

There are two ways to control a servo motor with Arduino. One is to use a common digital sensor port of Arduino to produce a square wave with different duty cycles to simulate the PWM signal and use that signal to control the positioning of the motor. Another way is to directly use the Servo function of the Arduino to control the motor. In this way, the program will be easier, but it can only control a two-contact motor because, for the servo function, only digital pins 9 and 10 can be used.

No.	Specifications
1	Operating angle range: about 180 ° (at 500 → 2500 μsec)
2	Pulse width range: 500 → 2500 μsec
3	No-load speed: 0.12 ± 0.01 sec / 60 (DC 4.8V) 0.1 ± 0.01 sec / 60 (DC 6V)
4	No-load current: 200 ± 20mA (DC 4.8V) 220 ± 20mA (DC 6V)
5	Stopping torque: 1.3 ± 0.01kg · cm (DC 4.8V) 1.5 ± 0.1kg · cm (DC 6V)
6	Stop current: ≦ 850mA (DC 4.8V) ≦ 1000mA (DC 6V)
7	Standby current: 3 ± 1mA (DC 4.8V) 4 ± 1mA (DC 6V)
8	Lead length: 250 ± 5 mm
9	Appearance size: 22.9 * 12.2 * 30mm
10	Weight: 9 ± 1 g (without servo horn)
11	Storage temperature: -20 ℃ ~ 60 ℃
12	Operating temperature: -10 ℃ ~ 50 ℃

Table 2.12: Servo Motor Specifications

No.	Advantages
1	Low Cost
2	They have a wide range of sizes and torque ratings
3	They are simple to control

Table 2.13: Servo Motor Advantages

No.	Applications
1	Robotics
2	Animatronics
3	Radio control Cars/Boats/Planes
4	Door control

Table 2.14: Servo Motor Applications

2.6 Background on Robot Kinematics

Mobile robots are often classified as holonomic or non-holonomic systems, depending on their motion constraints. A holonomic robot can move freely in all directions, whereas a non-holonomic robot has limited movement capabilities. For example, a car-like robot or a two-wheel differential drive robot cannot move sideways directly; it must rotate and then move forward or backward.

In robotics, one of the most widely used drive mechanisms is the differential drive system. A differential drive robot typically consists of two independently driven wheels mounted on either side of the chassis. By varying the angular velocities of the left and right wheels, the robot can achieve different types of movement:

- If both wheels rotate at the same speed, the robot moves in a straight line.
- If the wheels rotate at different speeds, the robot follows a curved trajectory.
- If one wheel rotates forward and the other backward, the robot performs an in-place rotation.

The study of kinematics in mobile robots focuses on the relationship between wheel velocities and the resulting robot trajectory. Researchers such as Campion et al. [1] and Siegwart et al. [2] have described mathematical models for non-holonomic robots, which form the foundation for robot navigation and control. These models are essential for designing algorithms related to path planning, posture stabilization, and trajectory tracking.

Understanding these fundamental principles provides the theoretical basis for analyzing and modeling the motion of the proposed robot car in this thesis. In Chapter 3, the specific motion model, differential steering equations, and pose analysis of the smart car are presented in detail.

2.7 Research Gap

Over the past few decades, significant progress has been made in the fields of obstacle avoidance, line following, and wireless robot control. Researchers have developed various approaches using ultrasonic sensors, infrared modules, Bluetooth communication, and even AI-driven vision systems [5]–[10]. However, most of these studies are limited to single-task robots, such as line-following or obstacle-avoiding platforms, and lack a unified design that integrates multiple navigation and control functions. In addition, while some works have explored smartphone-based wireless control, many of these systems do not include autonomous decision-making capabilities.

Another limitation of prior research is the lack of integration between sensing, wireless control, and vision systems in low-cost platforms. High-end smart car prototypes use cameras and AI for real-time navigation, but these designs are expensive, computationally intensive, and therefore unsuitable for educational or resource-constrained applications. Affordable platforms that are

designed for learning and small-scale experimentation often exclude real-time video feedback, which reduces their applicability in advanced robotics education and monitoring tasks.

To address these gaps, this thesis proposes the design and implementation of a robot car based on the ATmega328P microcontroller, integrating ultrasonic and infrared sensors for obstacle detection, IR modules for line following, Bluetooth for smartphone-based control, and ESP32-CAM for video streaming. By combining these features into a single system, the proposed design provides a cost-effective, multipurpose, and scalable robotic platform that bridges the gap between basic educational robots and advanced smart car prototypes, offering both autonomous and manual control capabilities.

CHAPTER 3

Design And Implementation

3.1 Overall Design Scheme and Objectives

In this paper, the core controller of the smart Car is selected as the Arduino control board, which controls the DC motor, and then controls the rotation direction and speed of the left and right wheels to control the driving condition of the car. The infrared ranging sensor and ultrasonic ranging sensor are used. It can realize the obstacle avoidance function of the car, and the action of the car can be controlled manually through the Bluetooth connection with the mobile phone. The second core controller of the car is selected as the ESP32-CAM module which acts as the robot car's eye and controls camera rotation in 0-180 degrees up-down and 0-180 degrees left-right for better surveillance. The overall design is shown in Fig. 3.1.

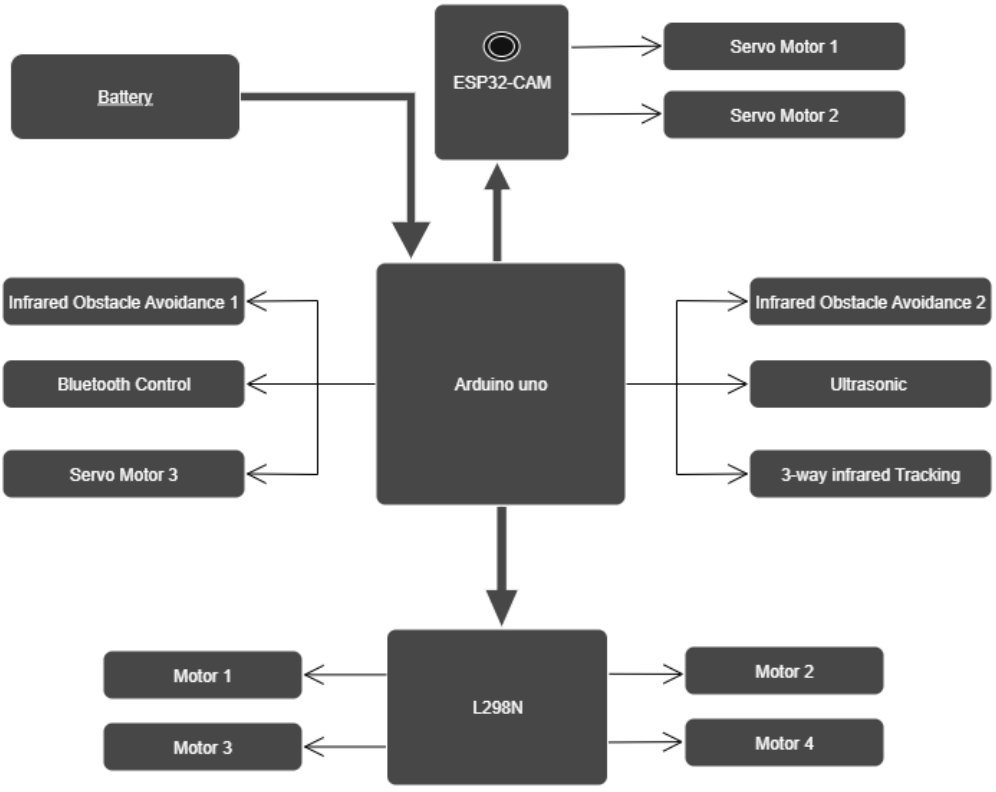


Fig. 3.1: The Overall Design of Smart Car

The functions of this smart car can achieve include: Infrared Obstacle Avoidance, Infrared Tracking, Ultrasonic Obstacle Avoidance, Mobile Phone Bluetooth Control, Video Streaming, Surveillance Camera Rotation. The overall goal is to accurately avoid obstacles at a set speed without touching the obstacles on the planned road section. Obstacles, smooth turns and smooth

arrival at the destination. Remote control functionality can be used when it's necessary to control the robot car manually. Camera can be used for better understanding of the unknown environment and can be rotated around 180-degree scale for surveillance of the environment.

3.2 Smart Car Implementation

The hardware implementation components of this project are shown in Table 3.1:

No.	Hardware	Quantity
1	Arduino UNO R3(Atmega328p)	1
2	ESP32-CAM	1
3	SG90s Servo	3
4	Line infrared tracing module	1
5	Infrared obstacle avoidance module	2
6	Ultrasonic obstacle avoidance	1
7	12580mwh, 18650 3.7V rechargeable lithium battery with longer endurance	2
8	4WD Chassis RC car	1
9	Jumper wires (male to male, female to female, male to female)	
10	DC Motors	4
11	L298N Motor Driver	1
12	Power switch	1
13	double 1850 battery holder	1
14	HC-SR04 holder	1
15	Copper Pillars	10 (M3+30mm), 4 (M3+6mm)
16	M2 and M3 Hex nuts	10-20
17	M3 round head screws	10-20
18	Shield Arduino Sensor V5	1

Table 3.1: Hardware Required for The Project

3.2.1 Ultrasonic Obstacle Avoidance

The Ultrasonic Obstacle Avoidance Module is an electronic sensor system that uses ultrasonic waves to detect and measure the distance between the robot car and surrounding objects. It works by transmitting ultrasonic signals at a specific frequency and then receiving the reflected signals from nearby obstacles. By calculating the time taken for the sound waves to return, the module determines the distance to the object. This feature enables the robot car to avoid collisions and navigate safely in real time.



Fig. 3.2: Physical Diagram of the Ultrasonic Module

(1) Types of Ultrasonic Sensors

Ultrasonic sensors are generally classified into two main categories: electroacoustic type and fluid dynamic type.

Electroacoustic type sensors mainly include piezoelectric sensors, magnetostrictive sensors, and electrostatic sensors. Among these, piezoelectric sensors are the most widely used due to their stability, reliability, and high precision.

Fluid dynamic type sensors include both gas and liquid whistles, but these are rarely used in robotic systems.

At the present stage, most ultrasonic modules are based on piezoelectric sensors, as they provide accurate, efficient, and robust performance for distance detection. The module used in the robot car can measure distances from 2 cm to 400 cm without physical contact, with a measurement accuracy of up to ±3 mm. The system compensates for errors by correcting the ranging results and uses GPIO communication with an internal watchdog, ensuring stable and reliable operation.

The ultrasonic module consists of three main parts: an ultrasonic transmitter, a receiver, and a control circuit. In smart car applications, such as distance measurement and steering control, ultrasonic sensors play a critical role in detecting obstacles ahead. This allows the intelligent car to make timely decisions, such as turning or stopping, to avoid collisions. Fig. 2.19 shows the physical structure of the Ultrasonic Obstacle Avoidance Module.

(2) Ultrasonic Obstacle Avoidance Module Parameters

No.	Ultrasonic Module Parameters
1	Power Consumption Current: Minimum 1 mA, Maximum 65 mA
2	Resonant Frequency: 40 kHz
3	Detection Range: 2 cm to 4 m (Error: 0.1 ± 1%); Closest detection: 4 mm; Maximum distance: 4 m; Continuous data output without additional settings
4	Operating Temperature Range: -40°C to +100°C
5	Dimensions: 46 mm × 21 mm × 20 mm (H)
6	Fixed Hole Size: 2 × Φ2 mm; Pitch: 42 × 16 mm

Table 3.2: Ultrasonic Avoidance Module Parameters

(3) Ultrasonic Obstacle Avoidance Module Works

The most commonly used ultrasonic sensor at this stage is the HC-SR04 with 4 pins: VCC, GND, Trig, and Echo. The method of ultrasonic ranging is echo detection, which means that the ultrasonic transmitter emits ultrasonic waves in a certain direction. At the same time as the emission, the timer starts counting. The ultrasonic waves propagate through the air, and when an obstacle is encountered, they are reflected back immediately. When the receiver detects the reflected ultrasonic waves, the timer stops.

The working sequence diagram is shown in Fig. 2.20. The propagation speed of ultrasonic waves in the air is approximately 340 m/s. According to the time t recorded by the timer, the distance s from the emission point to the obstacle can be calculated as:

$$s = 340.t/2$$

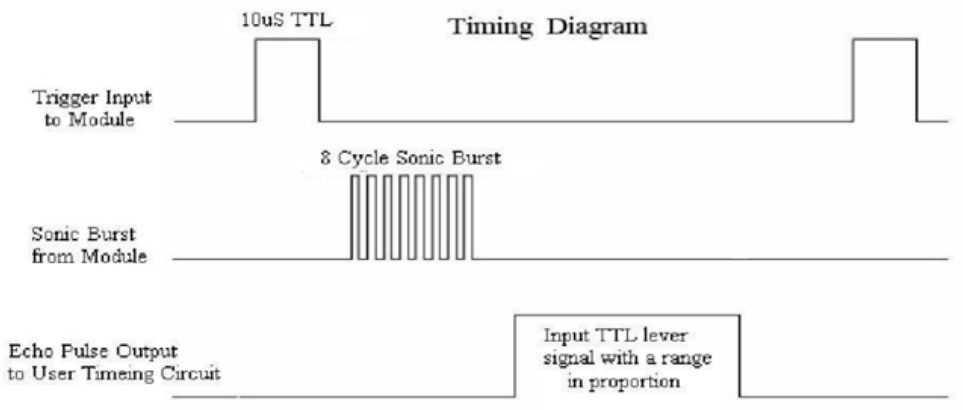


Fig. 3.3: Ultrasonic working sequence diagram

Let's analyze this timing diagram. First, the ultrasonic module is started by a trigger signal. The host must send a high-level pulse of at least 10 μs to trigger the module. After this, the module automatically emits ultrasonic waves. The output Echo signal is what we measure. The high-level duration of the Echo signal corresponds to the time taken for the ultrasonic wave to travel to the obstacle and back. By recording this time using a timer, the distance can be calculated. The result is divided by 2 because the total time includes both the transmission and reception.

Since ultrasonic waves are sound waves, their velocity depends on the air temperature. If the temperature remains roughly constant, the ultrasonic speed can be treated as constant. For high-accuracy measurements, temperature compensation can be applied. Once the speed of sound is determined, the distance can be accurately measured from the round-trip travel time. This is the basic principle of ultrasonic ranging.

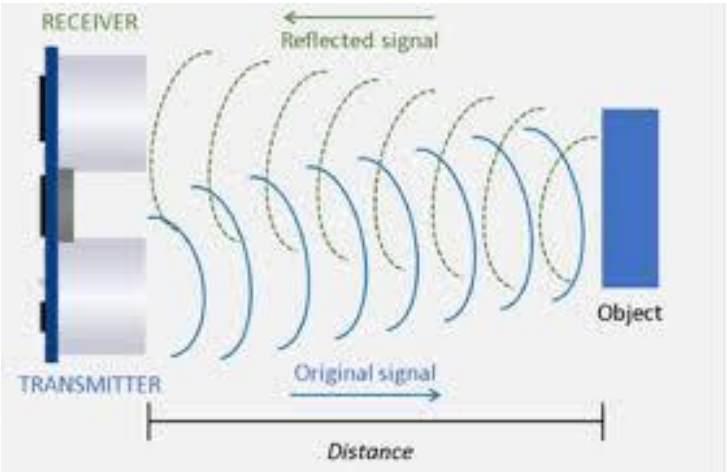


Fig. 3.4: Principle of the Distance Measurement of Ultrasonic Waves

(4) Ultrasonic Obstacle Block Schematic

The ultrasonic sensor is mainly divided into two parts: the transmitting circuit and the receiving circuit, as shown in Fig. 2.21. The transmitting circuit consists of the ultrasonic transmitting transducer, which emits ultrasonic waves. A square wave signal of 40 kHz from the Arduino digital output pin is sent to the ultrasonic transducer. This signal drives the transducer to generate ultrasonic waves that propagate through the air. The circuit is designed to provide sufficient power to the transducer to ensure strong ultrasonic emission.

The receiving circuit includes the ultrasonic sensor's receiving element, a two-stage amplifying circuit, and a frequency detection system. The ultrasonic waves reflected from an obstacle are received by the sensor, but the returned signal is very weak. The two-stage amplifier strengthens this signal so that it can be processed accurately. The amplified signal is then sent to a detection system, which identifies signals at the transmitted frequency (40 kHz) to avoid interference from other sound sources. When the frequency matches, a low-level interruption signal is sent to the microcontroller, indicating that an echo has been received. The microcontroller measures the time between sending and receiving the signal to calculate the distance to the obstacle, allowing the robot to respond and avoid collisions.

(5) Ultrasonic Obstacle Avoidance Function Experiment

Ultrasonic Obstacle Avoidance Function Flow chart:



Fig. 3.5: Ultrasonic obstacle avoidance function program execution flow chart

3.2.2 Infrared Obstacle Avoidance

The infrared obstacle avoidance and finder module integrates the infrared obstacle avoidance function and chasing function on one module, the infrared obstacle avoidance function is to emit an infrared signal through an infrared transmitting tube on the module , when the infrared signal encounters an obstacle being launched back, the infrared receiving tube receives the reflected infrared signal, thus judging that there are obstacles, to achieve the purpose of avoiding obstacles.

(1) Principle of infrared obstacle avoidance

The infrared obstacle avoidance and light-seeking module has a pair of infrared emitting and receiving tubes, and the transmitting tube emits infrared rays of a certain frequency. When the detecting direction encounters an obstacle (reflecting surface), the infrared light is reflected back and received by the receiving tube and passes through the comparator. After the circuit is processed, the green indicator light will be on, and the signal output interface will output a digital signal (a low-level signal). The detection effective range is 2~30cm, and the working voltage is 3.3V-5V. The detection distance of the sensor can be adjusted by the potentiometer. Because of the use of infrared rays, the anti-interference ability is very strong, and the measurement accuracy is high when the distance is moderate. In addition, the module is easy to assemble and easy to use and can be widely used in many situations such as robot obstacle avoidance, obstacle avoidance trolley, pipeline counting, and black and white line tracking.

(2) Infrared Obstacle Avoidance Module Parameters

No.	Ultrasonic Module Parameters
1	Working voltage: 5V
2	Infrared detection effective distance range: 2 ~ 30cm
3	Peak sensitivity wavelength: 590nm
4	Output infrared obstacle avoidance signal: digital signal

Table 3.3: Infrared Obstacle Avoidance Module Parameters

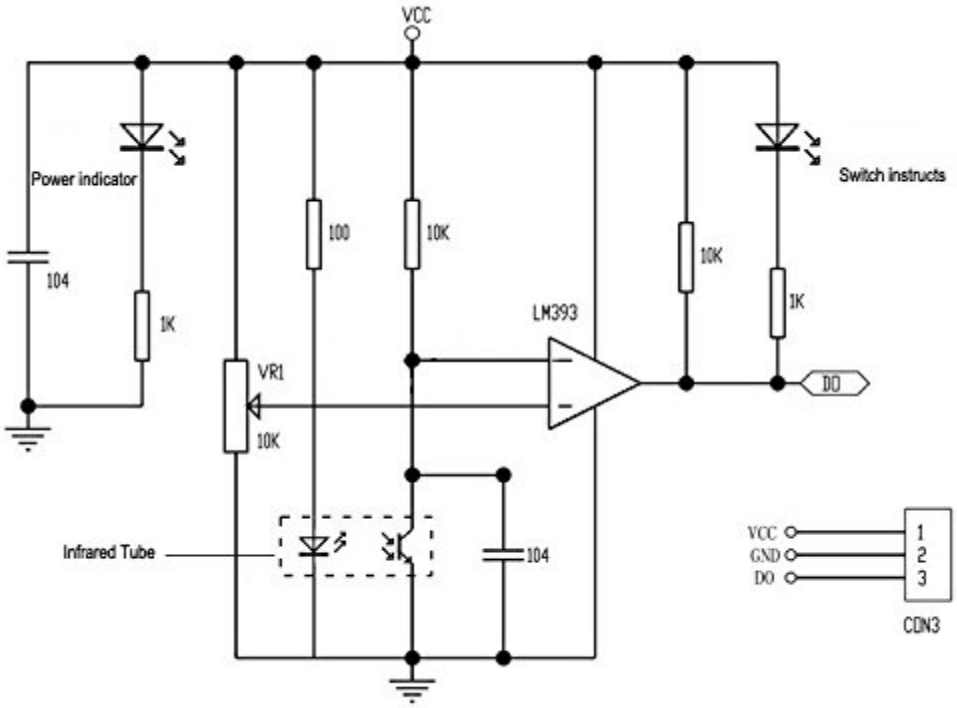


Fig. 3.6: Schematic diagram of infrared obstacle avoidance

Note: This module can adjust the distance of infrared obstacle avoidance detection by potentiometer. The detection distance is 2~30cm. If we find that the distance measurement is not very sensitive during use, we can use the trimmer potentiometer to achieve the desired result (clockwise potentiometer, The detection distance is increased; the counterclockwise potentiometer is used to reduce the detection distance), as shown in Fig. 2.23.

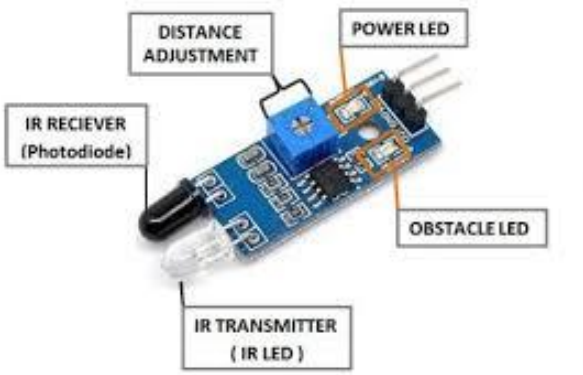


Fig. 3.7: detection distance adjustment

(3) Wire Connection with Arduino

Left Infrared obstacle avoidance module	Arduino Uno pin	Right Infrared obstacle avoidance module	Arduino Uno pin
V	VCC	V	VCC
G	GND	G	GND
D	4	D	A3

Table 3.4: Infrared Obstacle Avoidance Module Pin Connection

(4) IR Avoidance Test Code

```
const int LeftAvoidancePin = 4;   // Left IR sensor OUT pin
const int RightAvoidancePin = 3;  // Right IR sensor OUT pin

int dl, dr;

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

  pinMode(LeftAvoidancePin, INPUT);
  pinMode(RightAvoidancePin, INPUT);

  Serial.println("=== IR Obstacle Avoidance Test ===");
}
void loop() {
  // Read sensors
  dl = digitalRead(LeftAvoidancePin);
  dr = digitalRead(RightAvoidancePin);

  // Print status
```

```
  Serial.print("Left IR: ");
  Serial.print(dl);
  Serial.print(" | Right IR: ");
  Serial.println(dr);

  delay(500); // half-second delay for readability
}
```

---

(6) IR Avoidance Test

Step 1: Upload the code to the Car.

Step 2: Open the Serial Monitor.

Step 3: Place the obstacle 10cm in from the transmitter and receiver, and fine tune the potentiometer until output indicator lights up. Observed at the time that the output of the infrared obstacle avoidance signal and the serial monitor is 0. When the infrared obstacle avoidance signal output on the serial port monitor of the obstacle is removed, the infrared obstacle avoidance function is proved to be normal.

Step 4: Use a cover to block the light near the photo-resistor and then observe that the output value of the light-seeking signal on the serial monitor is relatively large and then remove the obstruction. It is found that the output value of the light-seeking signal on the serial monitor becomes smaller. We can also use the flashlight of the mobile phone to illuminate the photo-resistor to observe the change in the output value of the light-seeking signal on the serial monitor.

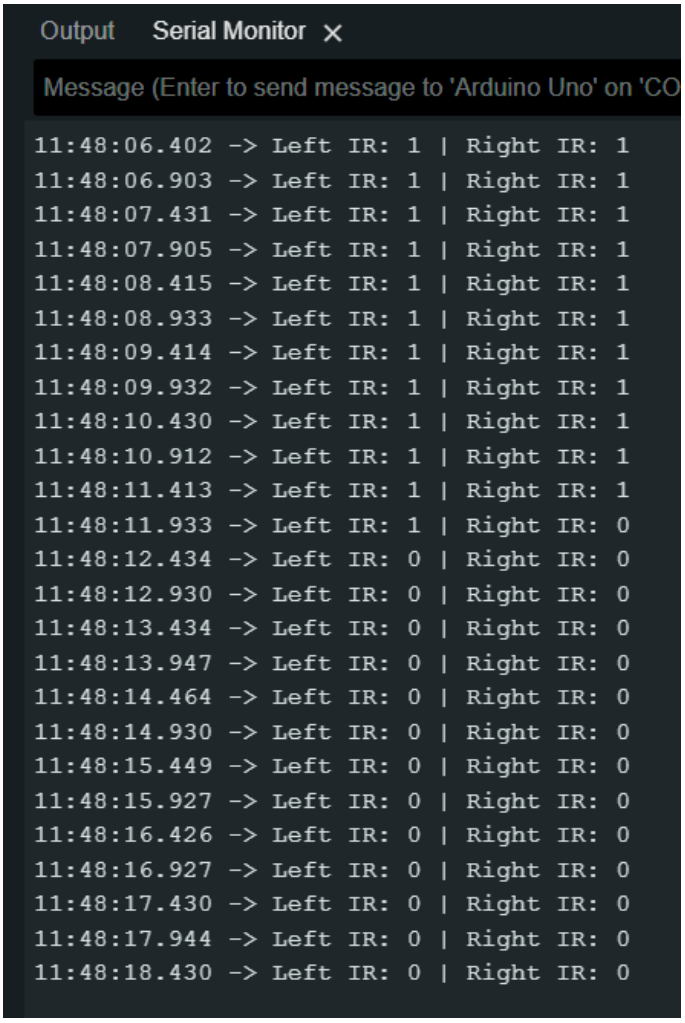


Fig. 3.8: Data when there is no obstacle or ambient light

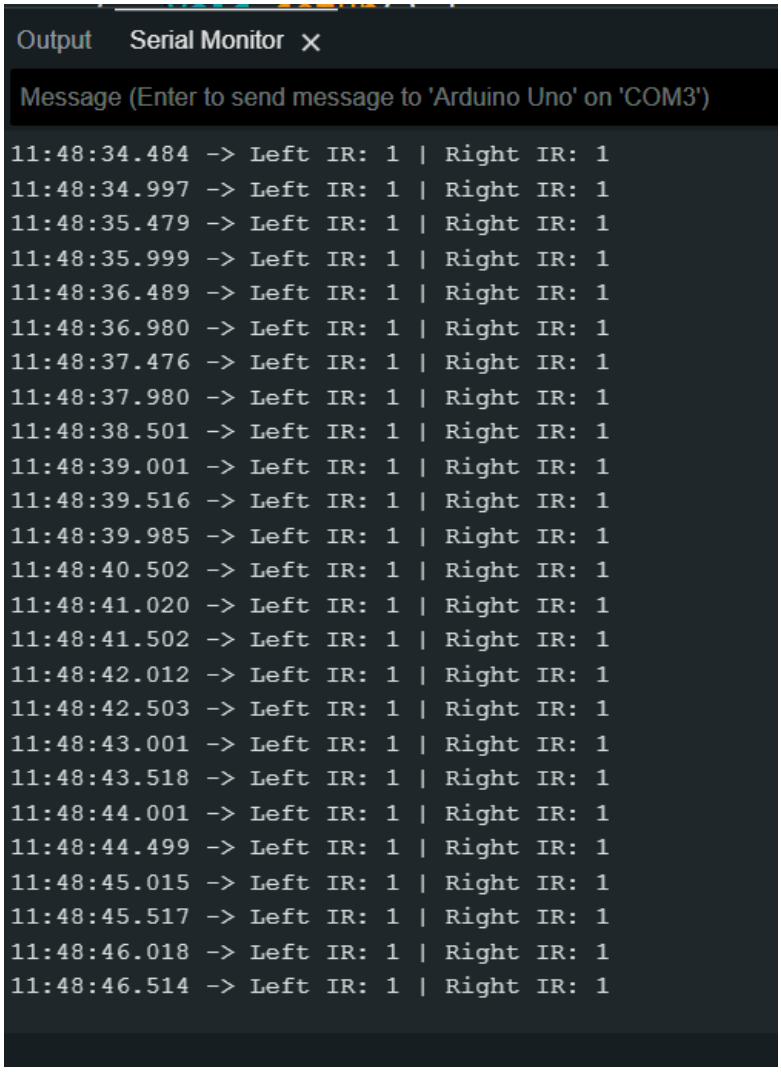


Fig. 3.9: Obstacles and strong ambient light

3.2.3 Infrared Line Tracking

The robot car uses a 3-way infrared tracking module that can follow either a black line or a white line. If the floor of the experiment is black, then should follow the white line (that is, put a white track on the floor); if the floor of the experiment is white, then we should follow the black line (ie, put a black on the floor) In other words, it is to make the track and other environments form a distinct color difference.

Black line tracking means that the car walks along the black line on the white floor. Because the black line and the white floor have different reflection coefficients of infrared rays, the "road" can be judged according to the strength of the received reflected infrared signal. White line tracking means that the car walks along the white line on the black floor. Because the white line and the black floor have different reflection coefficients to the infrared rays, the "road" can be judged according to the strength of the received reflected infrared signal. In the robot car, each tracking

module uses the ITR9909 sensor, the ITR9909 infrared reflection sensor, and a photoelectric sensor that combines emission and reception. It consists of an infrared light-emitting diode and an NPN infrared phototransistor. The detection reflection distance is 1mm-25mm. The sensor is specially equipped with an M3 fixed mounting hole. The adjustment direction and fixing are convenient and easy to use. The74HC14 Schmitt trigger inverter is used, the signal is clean, the wave form is good, and the driving ability is strong. It can be applied to robot obstacle avoidance, robot tracking of the white line or black line, black line in white background, white line in black bottom, and is a necessary sensor for tracking robot. Pipeline metering, metering pulse data sampling, fax machine shredder paper detection, and many other occasions. The small module physical map is shown in Fig. 3.10.

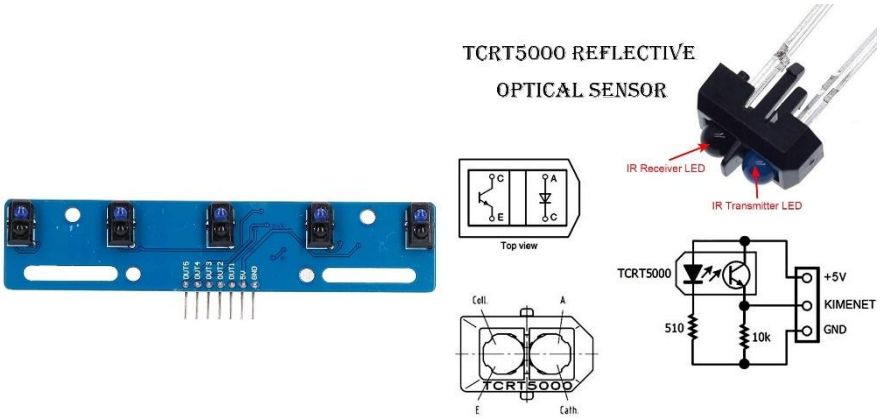


Fig. 3.10: Infrared tracking module physical map

(1) Working principle of the infrared tracking module

Infrared tracking module Whether it is black or white, the method we usually use is infrared detection. The infrared detection method, which uses infrared rays to have different reflection properties on the surface of objects of different colors, continuously emits infrared light to the ground during the driving of the car, and the infrared rays emitted are not reflected or reflected but are not strong enough. When the time is large, the infrared receiving tube is always in the off state. At this time, the output end of the module is low, indicating that the LED is always off; when the infrared light encounters the white paper floor, diffuse reflection occurs, and the reflected light is mounted on The receiving tube on the trolley receives; and the infrared light is reflected back and the intensity is large enough, the photo-transistor is saturated, and the output of the module is at a high level, indicating that the LEDis illuminated. The car is based on the reflected infrared light to determine the position of the black line and the walking path of the car. In the robot car, we use a three-way tracking module, one module has three sensors, one on each side, one in the middle, as shown in Fig. 3.9, the tracking sensors are all in a straight line.

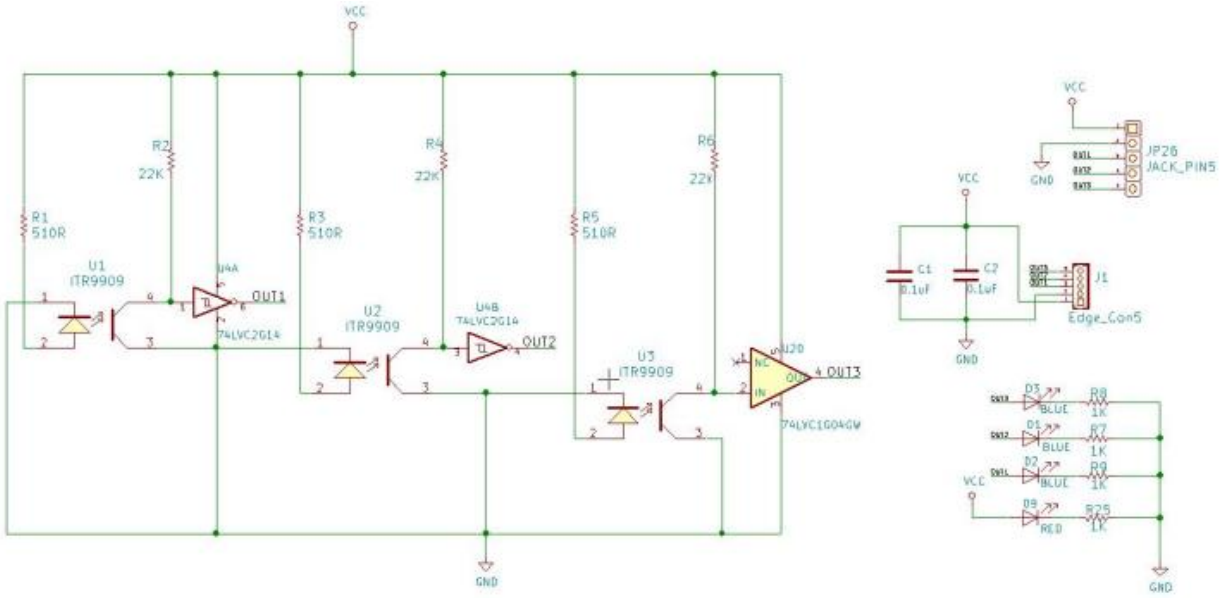


Fig. 3.11: Schematic diagram of the tracking module

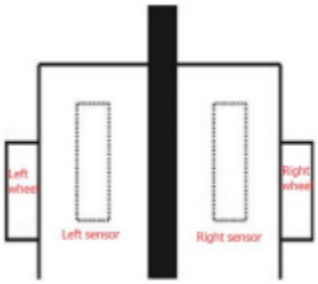


Fig. 3.12: Schematic diagram of the tracking module installation

When the car advances, it is always between the two first-level sensors. When the car deviates from the black line: If the left sensor detects the black line, the right and middle sensors will not detect the black line, indicating the car. Offset to the left, the cart will deflect slightly to the right, keeping the middle sensor always detecting the black line and walking along the black line; if the right sensor detects the blackline, the left, and middle sensors will not detect the black line, indicating that the car has been shifted to the right, then the car will be slightly deflected to the left, keeping the middle sensor always detecting the black line, and walking along the black line.

(2) Infrared Tracking Module Parameters

No.	Infrared Tracking Module Parameters
	Infrared Tracking Module Parameters
1	Using ITR9909 infrared reflection sensor.
2	Detection distance: 1mm~25mm, focus distance is 2.5mm.
3	Comparator output, clean signal, good waveform, strong driving capability, over 15mA.
4	Working voltage 3.3V-5V.
5	Use wide voltage 74LVC1G04 and 74LVC1G4 inverters, digital switching output (0 and 1).
6	Fixed bolt holes for easy installation

Table 3.5: Infrared Tracking Module Parameters

(3) Infrared Tracking Module Wire Connection

Infrared Tracking Module	Arduino UNOR3
VCC	VCC
GND	GND
OUT1	A0
OUT2	A1
OUT3	A2

Table 3.6: Infrared Tracking Module Wire Connection

(4) Infrared Tracking Module Test

Code:

```
void setup() {
  Serial.begin(9600);
  45
```

```
    pinMode(A0, INPUT);
    pinMode(A1, INPUT);
    pinMode(A2, INPUT);
}
void loop() {
    int left, center, right;
    left = digitalRead(A0);
    center = digitalRead(A1);
    right = digitalRead(A2);
    Serial.print("right:");
    Serial.print(right);
    Serial.print(" ");
    Serial.print("center:");
    Serial.print(center);
    Serial.print(" ");
    Serial.print("left:");
    Serial.print(left);
    Serial.println(" ");
}
```

Step 1: Upload the code.

Step 2: Open the Serial Monitor.

Step 3: First block the infrared tracking module sensor with the hand. At this point, the module's red indicator light will flash, monitor the contents printed on the serial monitor, and then block the infrared tracking module sensor with a black plate. At this point, red on the module. The indicator light will not be illuminated and will then observe what has been printed on the serial monitor. If the black line is detected, the module output prints 0, if no black line is detected, the module output prints 1.



Fig. 3.13: When the infrared tracking module does not detect the black line data

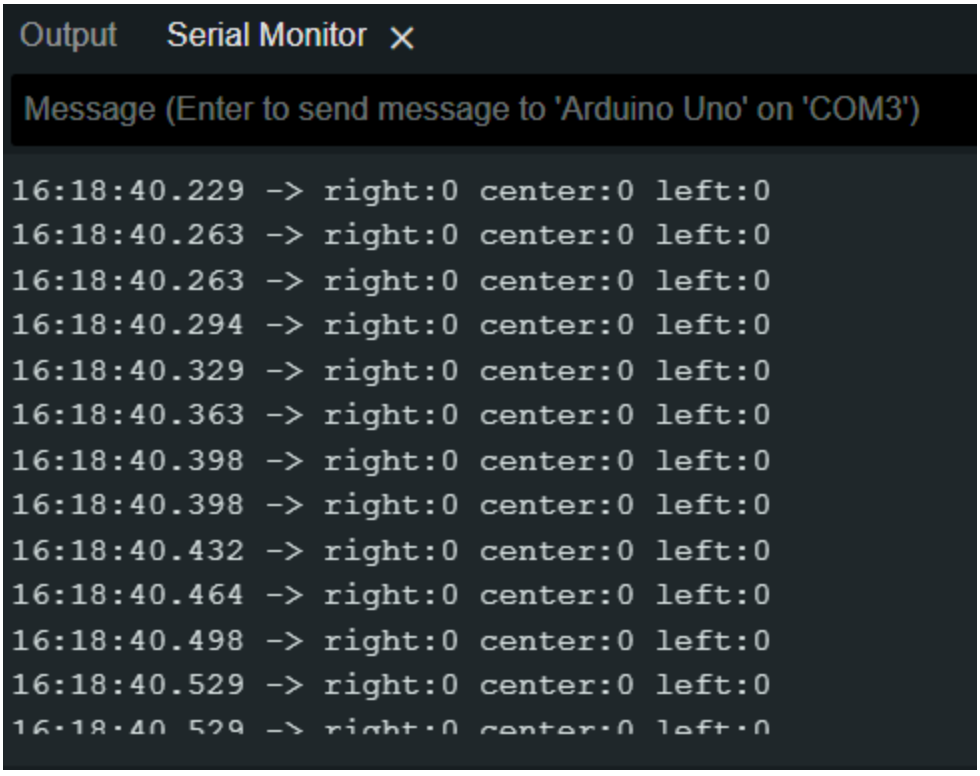


Fig. 3.14: Data when the infrared tracking module detects a black line

From Fig. 2.30 and Fig. 2.31, we can see that when the infrared tracking module does not detect the black line, it outputs a high level, and when the black line is detected, it outputs a low level because the analog port is used to acquire the sensor signal. Therefore, the value printed by the serial port is also analog, that is, it is close to 1024 when it is high level and 0 when it is low level. If the output is a digital signal, the serial port prints a value of 0 or 1.

3.2.4 Bluetooth Control

The HC-05 Bluetooth module is a wireless communication device based on the Bluetooth 2.0 standard. It is widely used in embedded systems and robotics projects to enable wireless serial communication between a microcontroller and external devices such as smartphones, tablets, or PCs. The module is commonly used in slave mode for receiving commands, but it can also be configured as a master device to initiate communication with other Bluetooth modules. The HC-05 communicates through the UART protocol (TX and RX pins) with a default baud rate of 9600 bps, which can be configured using AT commands.

No	Function	Description
1	Wireless Serial Communication	Provides UART-based communication between Arduino and external devices.
2	Slave Mode	Default mode; pairs with a master device (smartphone/PC) to receive commands.
3	Master Mode	Can be configured to act as a master device to initiate connection with other modules.
4	AT Command Support	Allows configuration of settings such as device name, password, and baud rate.
5	LED Status Indicator	Indicates pairing and connection status with different blinking patterns.
6	Short-Range Wireless Control	Provides communication up to 10 meters in open space, suitable for real-time control.

Table 3.7: Function of Bluetooth module

(1) Bluetooth Data Protocol

Using the Bluetooth remote control car, the Android App is used to send commands to the Arduino UNO serial port via Bluetooth to control the forward, reverse, and speed of the motor. Since it involves wireless communication, one of the indispensable issues is the communication problem between the two devices. But there is no common "language" between them, so we need to design a communication protocol to ensure data interaction between Android APP and Arduino UNO.

(2) Bluetooth Connection with Shield v5.0

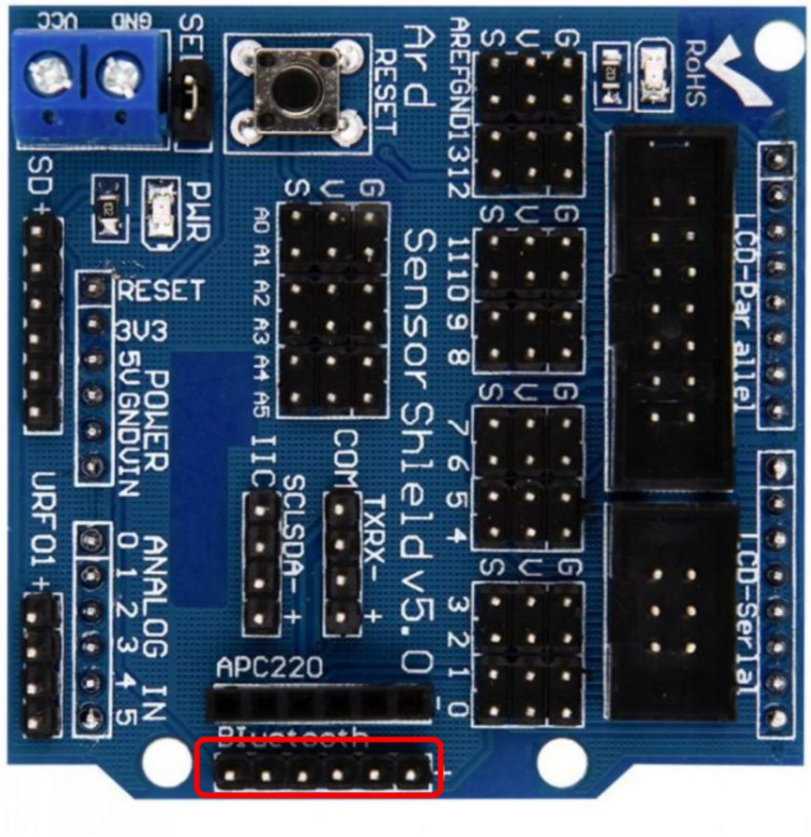


Fig. 3.15: Bluetooth Connection

3.3 Motion System Modeling

3.3.1 Motion Model of the Smart Car

Establishing a reasonable mathematical model is the most important step in the study of smart cars. After the establishment of the motion model, quantitative mathematical expression can be carried out. According to the mathematical model of the smart car, the mathematical relationship expression between the physical quantities of the smart car can be constructed. At the same time, the establishment of the mathematical model is also the basis for the positioning and path planning of the smart car. Establishing a mathematical model is the process of establishing a functional relationship. By looking for the relationship between the input and output of the function, the variable relationship of the smart car can be found. The relationship between the speed, position, and attitude of the smart car is the key to studying the positioning of the smart car, so the establishment of the relationship between the state variables of the smart car and the control independent variables is very important. In the experimental smart car in this paper, all four wheels

are directly driven by DC motors, and the steering is realized by the differential rotation of the wheels. According to the existing experimental conditions, the following assumptions are made:

(1) The experimental ground is a horizontal plane, the smart car has a rigid structure, and the body of the smart car is a rigid body on the wheels.
(2) The friction force of the contact points between the four tires and the test ground is large enough and will not slip.
(3) The thickness of the four tires of the smart car is equal, which only affects the vertical distance between the center of gravity of the smart car and the ground, and does not affect the angle of the car on the horizontal plane.
(4) The wheels of the smart car are absolutely symmetrically distributed, relying on differential steering.
(5) Ignore air resistance.

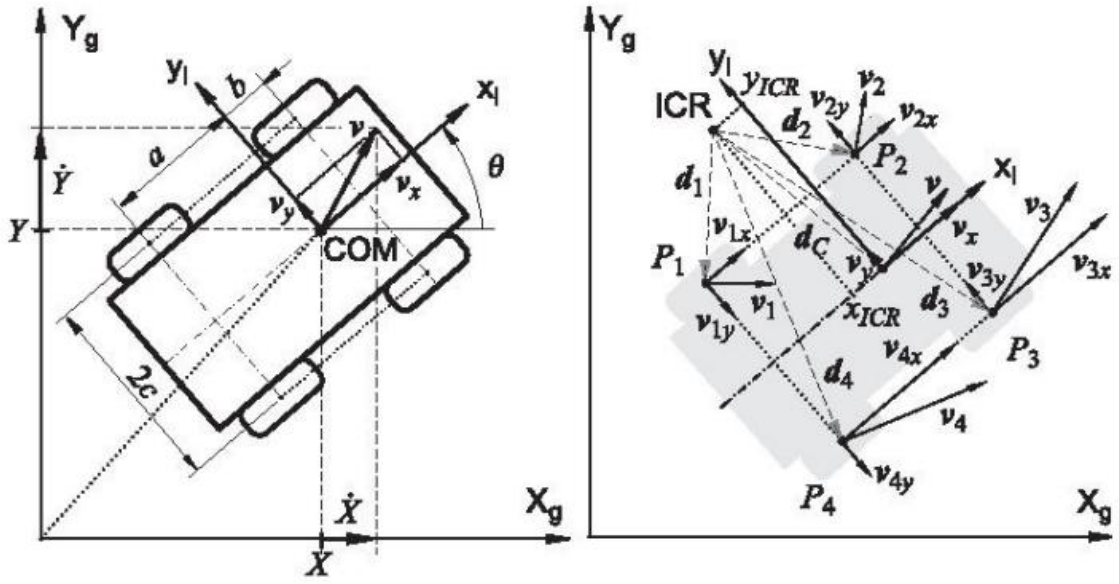


Fig. 3.16: Schematic diagram of smart car coordinates in global coordinate system

The smart car has three degrees of freedom, because the car can move forward, backward, turn left and right, and the forward and backward are in a straight line. This article will establish two Cartesian coordinate systems: the local Cartesian coordinate system and the global Cartesian coordinate system. The global Cartesian coordinate system is the driving environment range of smart cars. The establishment of the local Cartesian coordinate system can determine the location of the smart car and the correspondence of the information points in the environment. The schematic diagram of the coordinate system established in this paper is shown in Fig. 2.33. The information related to the establishment of the coordinate system is shown in Table 2.23. The angle between the x-axis in the local rectangular coordinate system and the X gin the global rectangular coordinate system is θ, and (x, y, θ) T is used to represent the position and direction of the smart car in the global environment.

Coordinate System	Origin	Abscissa	Ordinate
Local Rectangular	COM	x1	y1
Global Rectangular	0	Xg	Yg

Table 3.8: Key information identification table of coordinate system

In the figure, the geometric center point of the car is pointing G, b is the distance between the front axle and the center of mass COM, a is the distance between the rear axle and COM, 2c is the distance between the left and right wheels, and Ris the radius of the wheel. The linear speed of the left wheel of the smart car suck, and the linear speed of the right wheel is ignorant. The smart car in this paper is b=13.5cm, a=12cm, R=3.2cm, and c=13cm.

When building the motion model of the car, the starting point of the smart car movement is the origin 0 of the global coordinate, and the position of the real-time mass center of the car is the origin COM of the local Cartesian coordinate system. The smart car in the experimental study adopts a four-wheel drive mode. The speed of the wheels driving on the same side is the same during operation. The left and right wheels of the smart car and the front and rear wheels have the same rotational speeds respectively. The center of mass speed of the smart car is equal to the wheel speed. The relationship between is:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{v_L + v_R}{2} \\ \frac{-v_L + v_R}{2c} \end{bmatrix}$$

Among them, the linear speed of the intelligent car ν =[νx, νy, θ], angular velocity of the center of mass ω=[0,0,ω]T

Thus, the kinematic model of the smart car can be expressed as:

$$\begin{bmatrix} x \\ y \\ \theta \end{bmatrix} = \begin{bmatrix} con\theta & 0 \\ sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

And the above formula must satisfy the following non-integrity constraints:

$$xsin\theta - ycos\theta = 0$$

Therefore, controlling the position and attitude of the smart car can be achieved by adjusting the variable linear velocity v and angular velocity w in the function, or by adjusting the variable linear velocity VL and VR, that is, the left wheel speed K and the right wheel speed VR, to achieve intelligent Changes in the car's posture.

3.3.2 Wheel Differential Steering Model

When the differential steering kinematics model is established, the external forces, such as air resistance and acceleration resistance, are not considered due to the low speed of the trolley. The smart car uses a DC motor to drive the two-wheel differential rotation to achieve turning. The following assumptions are made for the smart car:

The car is regarded as a rigid body, and the chassis, body and driving wheels are not deformed;

When driving, the wheel of the trolley is in a pure rolling state, and no slippage occurs;

The longitudinal axis of the trolley is perpendicular to the traveling direction of the trolley, and no deformation occurs.

Assuming that the car body is running in the above ideal state, COM is the instantaneous center of speed, and the instantaneous speed Vc of the car at point C is:

$$v_c = \frac{v_L + v_R}{2}$$

Assuming that the angular velocity of the car body is the side, when the car travels clockwise, we can get:

$$\begin{cases} v_L = \theta(2c + c) \\ v_R = \theta(2c - c) \end{cases}$$

the angular velocity of the smart car can be obtained as:

$$\theta = \frac{v_L - v_R}{2c}$$

The instantaneous turning radius of the car is:

$$R = \frac{2c(v_L + v_R)}{2(v_L - v_R)}$$

Then the kinematics equation of the car when driving at differential speed is:

$$\begin{cases} v_c = \frac{v_L + v_R}{2} \\ \theta = \frac{v_L - v_R}{2c} \end{cases}$$

By adjusting the line speed of the left wheel and the line speed of the right wheel, the movement mode of the differential driving of the smart car can be controlled:

(1) If $v_L = -v_R, v_c = 0$, The car does a rotary motion.
(2) If $v_L = v_R, v_L = v_R = v_c, R += \infty$, The car drive in a straight line.

(3) If $v_L < v_R$ or $v_L > v_R, v_c = \frac{v_L + v_R}{2}$, The car is steering, and the instantaneous turning radius is at this time $R = \frac{2c(v_L + v_R)}{2(v_L - v_R)}$, then the turn of the intelligent car in this paper is half $R = \frac{2c(v_L + v_R)}{2(v_L - v_R)} = \frac{13(v_L + v_R)}{(v_L - v_R)} = 13cm;$

3.3.3 Movement pose analysis

The connection of the position points of the car at different time points is the trajectory of the smart car. When the car is driving, it needs to continuously obtain the position and attitude of the car body in the X_gOY_g coordinate system. Fig. 2.34 shows the change of its pose when turning at differential speed.

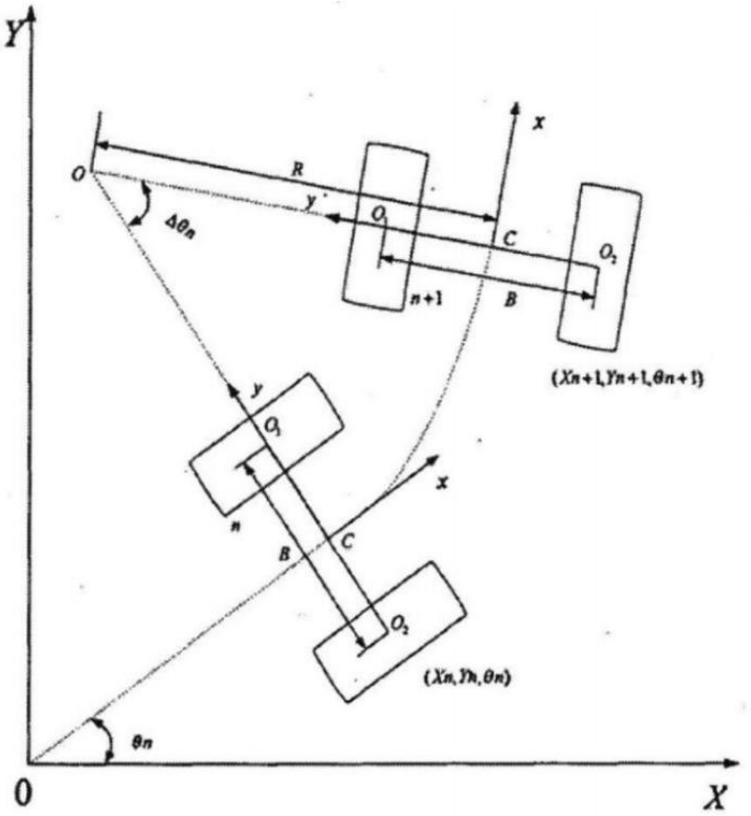


Fig. 3.17: Differential Steering Pose Model Diagram

The moving coordinate system xCOMy is established on the body of the car. After Δt time, the body of the car reaches the end pose coordinates $(X_{n+1}, Y_{n+1}, \theta_{n+1})$ from the initial starting point coordinates (X_n, Y_n, θ_n). If the driving trajectory of the car is an arc, there is $\Delta\theta_n = \theta_{n+1} - \theta_n$, R is the driving trajectory of the car, then it is known that the pose vector of the car at time n is (X_n, Y_n, θ_n), According to the below formulas, the pose vector at the time of (n+1) of the car can be obtained as $(X_{n+1}, Y_{n+1}, \theta_{n+1})$.

$$\begin{cases} \Delta X_n = \Delta x_n cos\theta_n - \Delta x_n sin\theta_n \\ \Delta Y_n = \Delta x_n sin\theta_n - \Delta y_n cos\theta_n \\ \Delta x_n = \omega \Delta t \end{cases}$$

$$\begin{cases} X_{n+1} = \Delta X_n + X_n \\ Y_{n+1} = \Delta Y_n + Y_n \\ \theta_{n+1} = \Delta \theta_n + \theta_n \end{cases}$$

From the above analysis of the driving position and attitude of the car, it can be concluded that the adjustment of the rotational speed of the left and right wheels of the trolley can change and determine the driving position and attitude of the trolley to achieve the ideal position and attitude. Digital control is used to realize the differential steering of the smart car.

According to the ultrasonic propagation speed of the ultrasonic obstacle avoidance sensor, it can be calculated that when it detects that the distance between the obstacle ahead and itself meets the turning distance, it takes 0.76 milliseconds. According to the driving speed of the smart car, the car will continue to Move forward1.27*10 -5 cm, combined with its obstacle avoidance accuracy, set the safe distance threshold between the car and the obstacle to 14 cm.

CHAPTER 4

EXPERIMENTAL RESULTS AND ANALYSIS

4.1 Robot Car Assembly

(1) Environment Set Up

If ESP32-CAM module can be found in the board type of Arduino IDEthen this section can be skipped.

Step 1: Open Arduino IDE.

Step 2: Go to File-->Preferences.

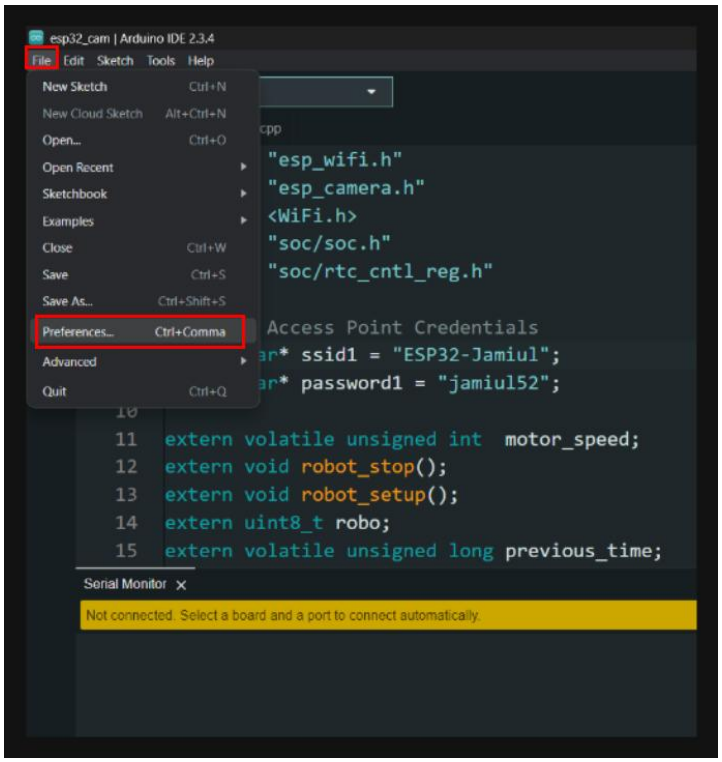


Fig. 4.1: Selecting Preference Option on Arduino IDE

Step 3: Paste *https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json* to the Additional Boards Manager URLS box.

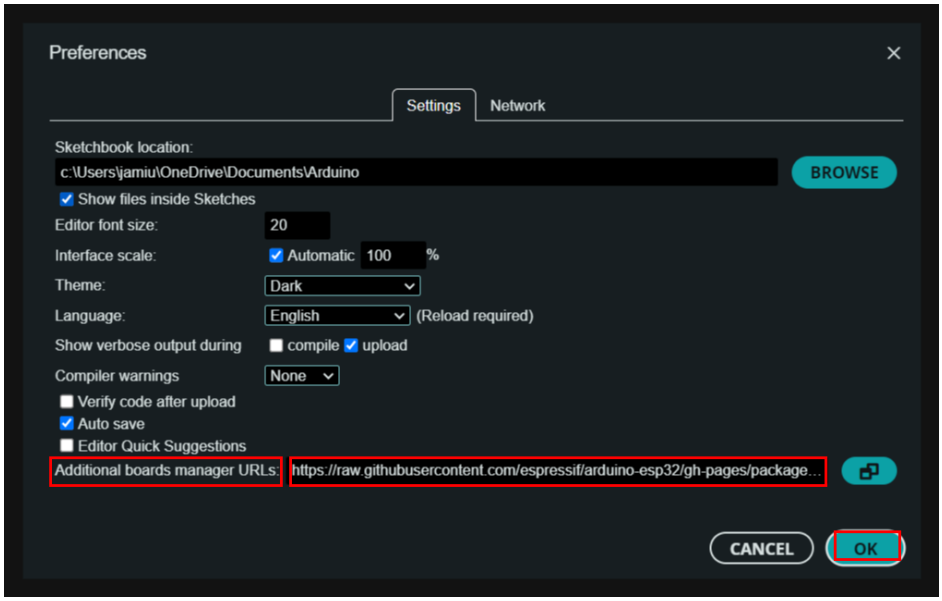


Fig. 4.2: Additional Boards Manager URLs

Step 4: Configure the development board: Open Tools-->Board: "Arduino UNO" -->Boards Manager. Search esp32 to select the version and install it.

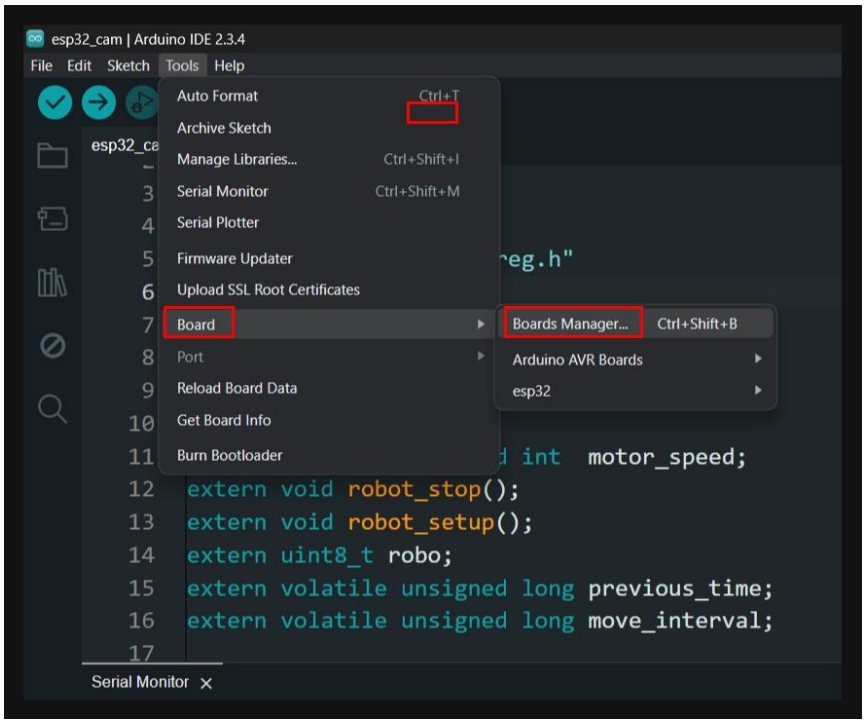


Fig. 4.3: Selecting the Boards Manager Option

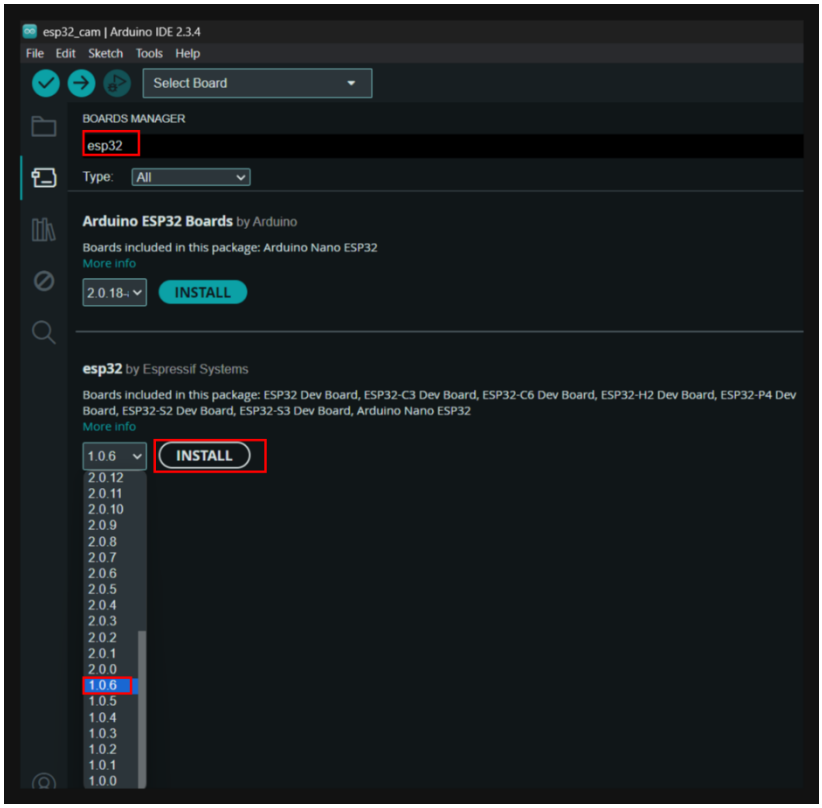


Fig. 4.4: Searching for the esp32 library

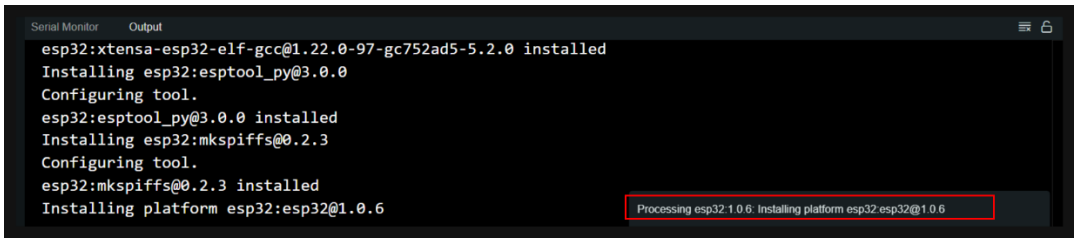


Fig. 4.5: Installing the esp32 library

Step 5: Wait until successful installation.

(2) Uploading the Code

Step 1: Connect the ESP32-CAM with the micro-USB cable and upload the code without any problem.



Fig. 4.6: ESP32-CAM Connection with Micro USB cable

Step 2 : make sure the parameters of serial port and board type are correct.

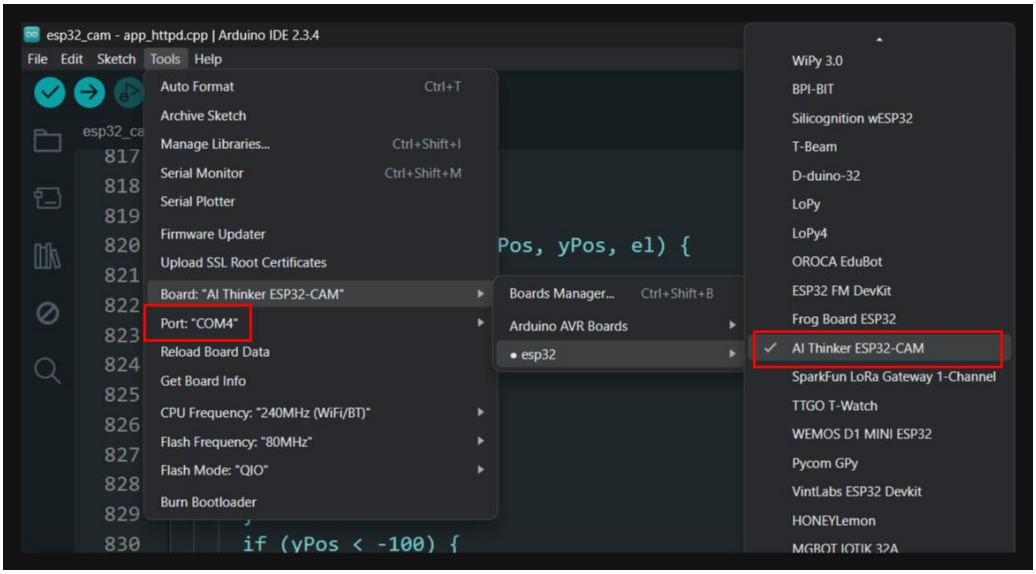


Fig. 4.7: Board Selection

Step 3: Configure WiFi name and password:

```
const char* ssid1 = "ESP32-Jamiul";
const char* password1 = "jamiul52";
```

Fig. 4.8: WiFi Name and Password Setup

Step 4: Upload the code and the IDE shows these dots on the debugging window as shown below, press the ESP32-CAM on-board RST button.

```
esptool.py v3.0-dev
Serial port COM4
Connecting....
```

Fig. 4.9: Code Uploading

After a few seconds, the code should be successfully uploaded to the board.

(3) Getting IP address

After uploading the code, Open the Serial Monitor at a baud rate of 115200. Press the ESP32-CAM on-board Reset button. The ESP32 IP address should be printed in the Serial Monitor.

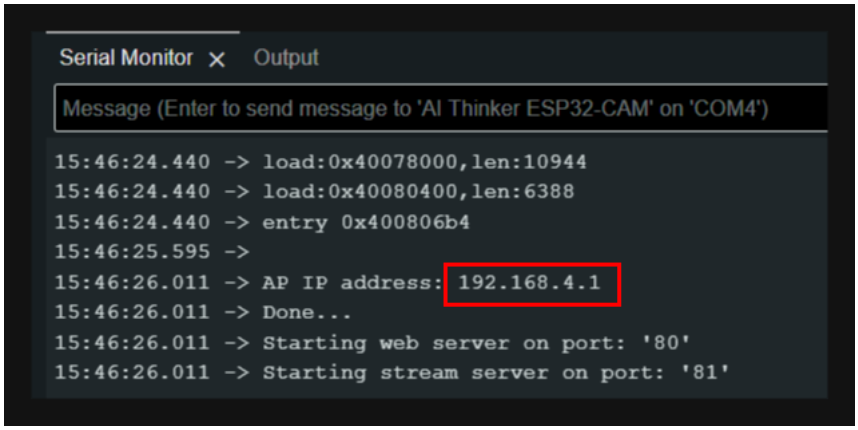


Fig. 4.10: Getting the IP Address

(4) Accessing the Video Streaming Server

Step 1: Connect the WiFi.

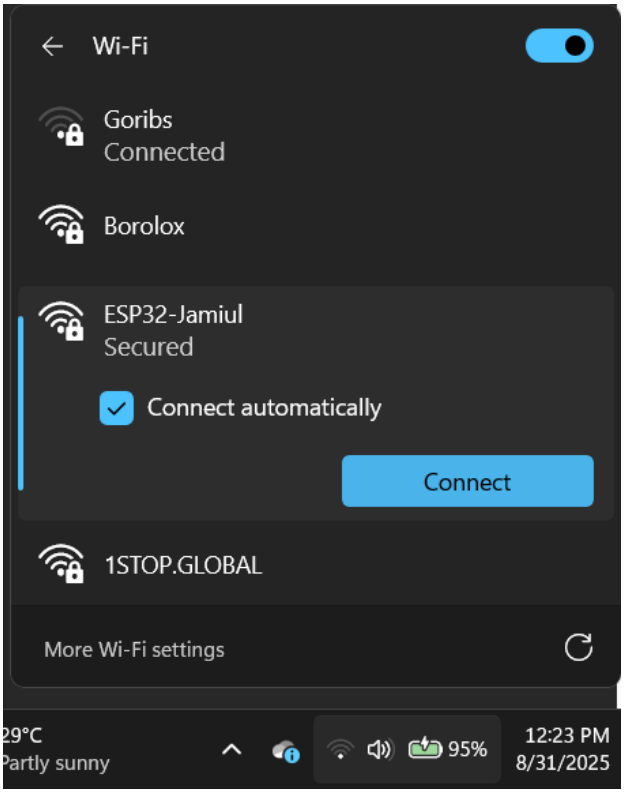


Fig. 4.11: WiFi Connection

Step 2: Open a browser and type the ESP32-CAM IP address A page withe the current Video streaming should load.

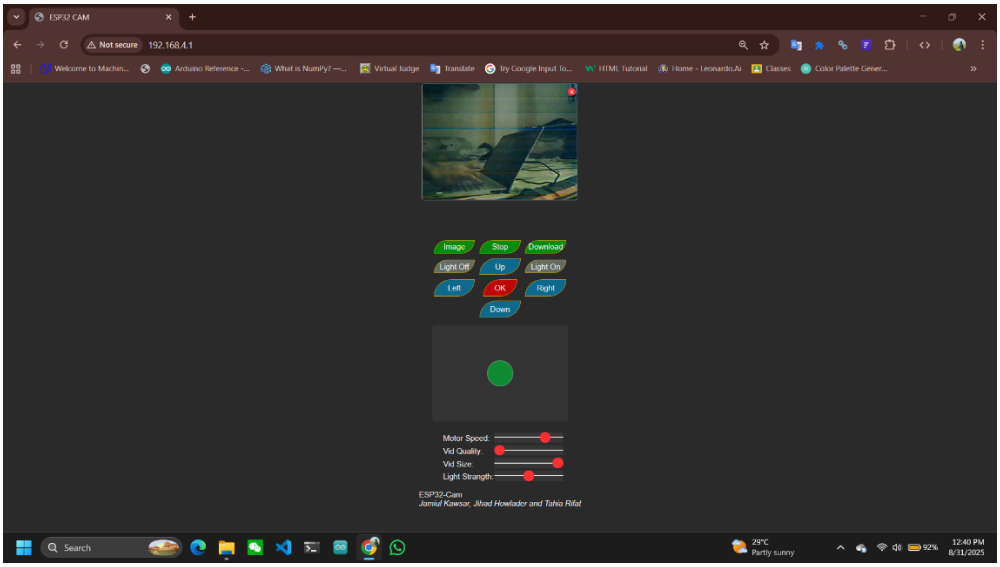


Fig. 4.12: Accessing the Video Streaming Server

(5) WD Car Chassis and Motor Mounting

First open all the boxes, take out the 4WD Chassis, screwdriver, screws, nuts, black and red welding wires, four motors and wheels etc.

Step 1: Welding wire on the motor

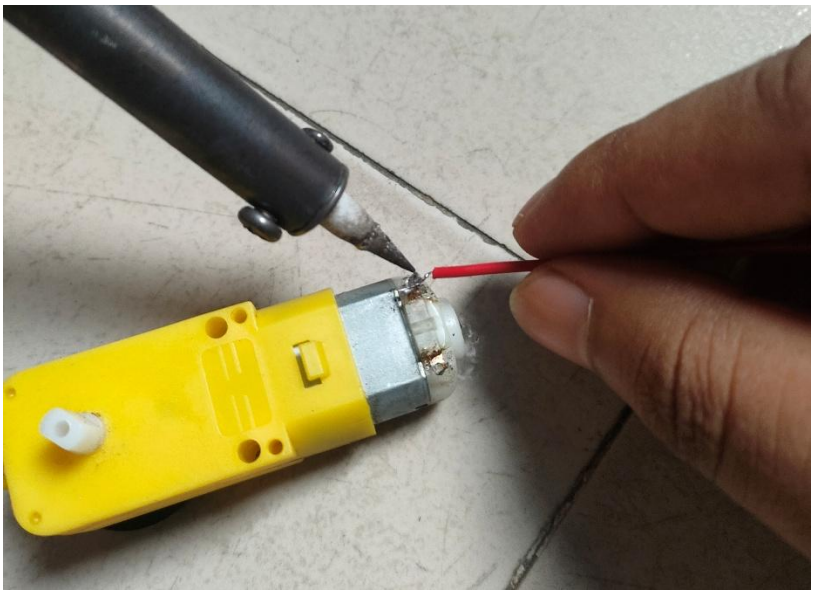


Fig. 4.13: Welding wire on the motor

Step 2: Mounting motor mount with Car Chassis

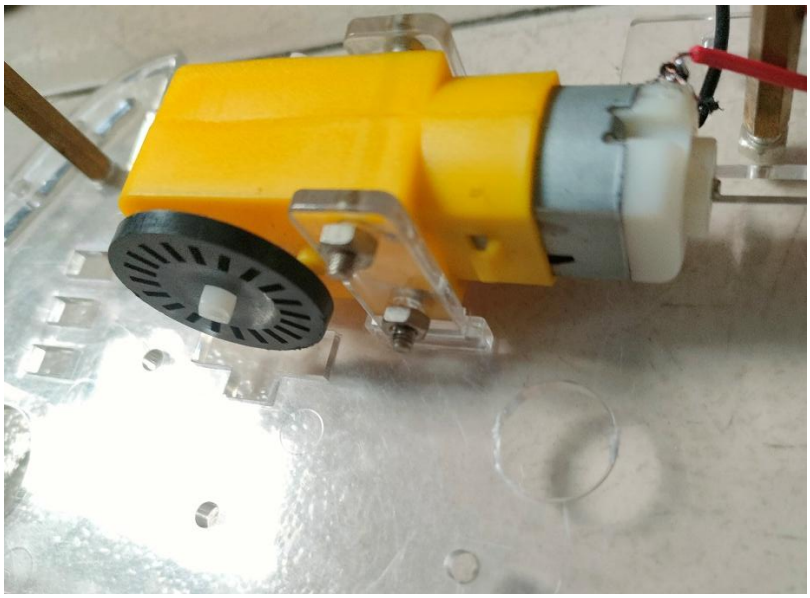


Fig. 4.14: Diagram of Motor Installation

Note: The screws should not be screwed too tightly, otherwise the wheel adjustment will be affected. The motor holder screws are always installed from the outside to the inside. If it is installed on the opposite side, the extended screw will affect the rotation of the wheel. Installation must ensure that the screw hole in the bracket is aligned with the screw hole in the acrylic plate.

Step 3: Mounting wheels

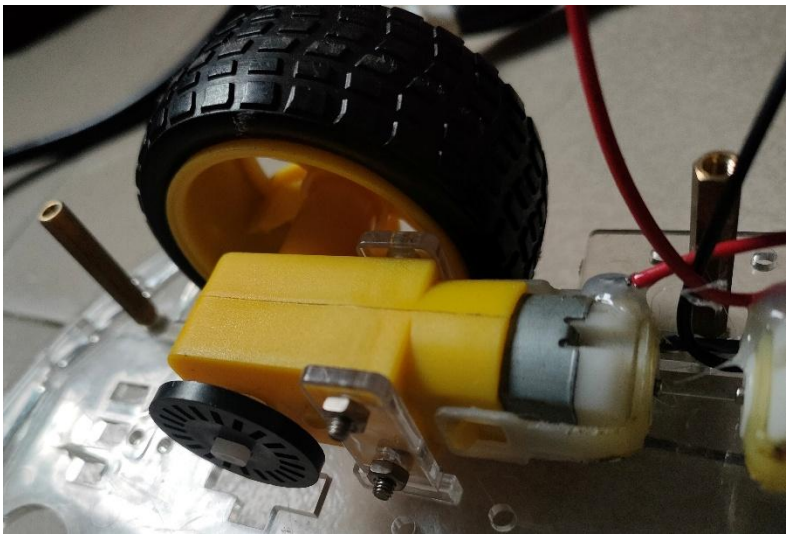


Fig. 4.15: Mounting wheel

(6) Motor Driver connection with the motors

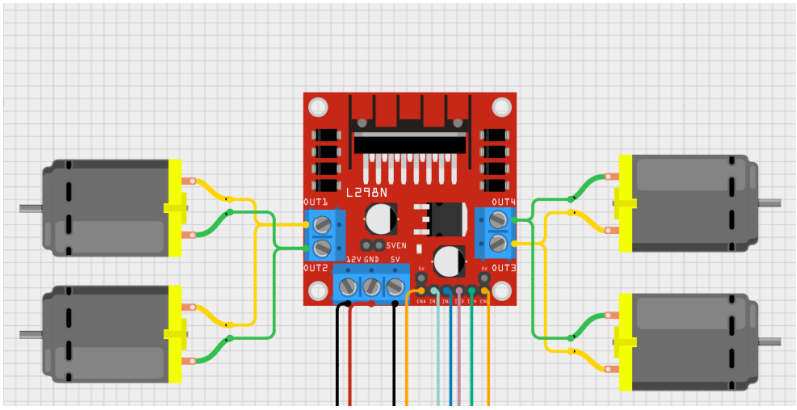


Fig. 4.16: Diagram of Connection Between Motors and Drive Board

Next connect the motor wires to the drive, remembering that the rotational directions of the two motors need to be identical, so we must first change and debug the program when wiring. Connecting the motor to any motor driver board, as shown in Fig. 3.17, then connecting the two wires of the battery box to the motor drive + 12v (red) and GND (black) and leading one wire from + 5V to “3” or “4” in the figure (IN1&IN2 one motor set, IN3 & IN4 another set). Observing the rotation directions of motors, if the rotation directions are not the same then only "1" and "2" need to be changed in the figure. Short enA and enB pin to control the motor speed. Removing short pin shown in fig. 3.16.

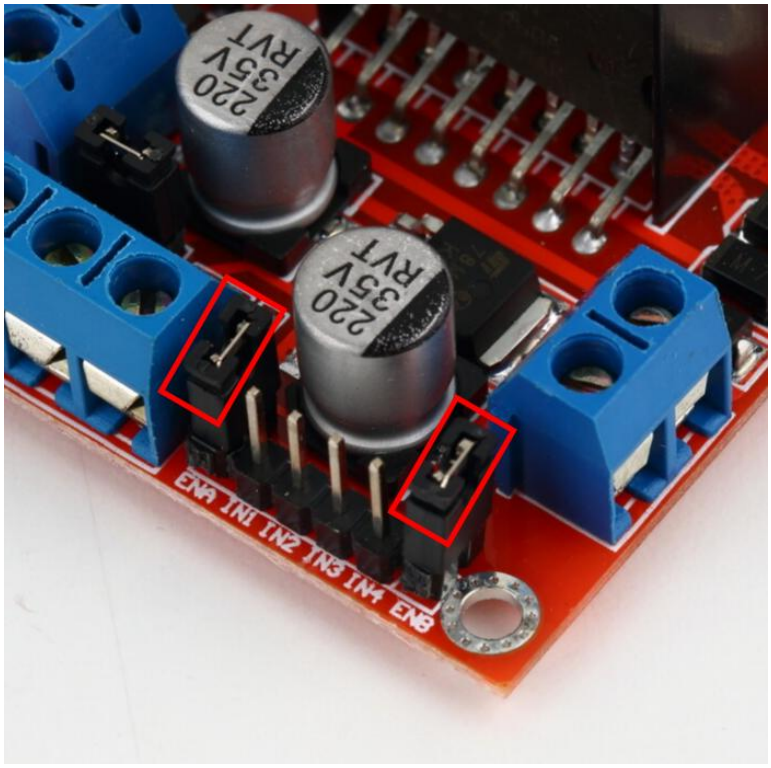


Fig. 4.17: remove short pin from driver

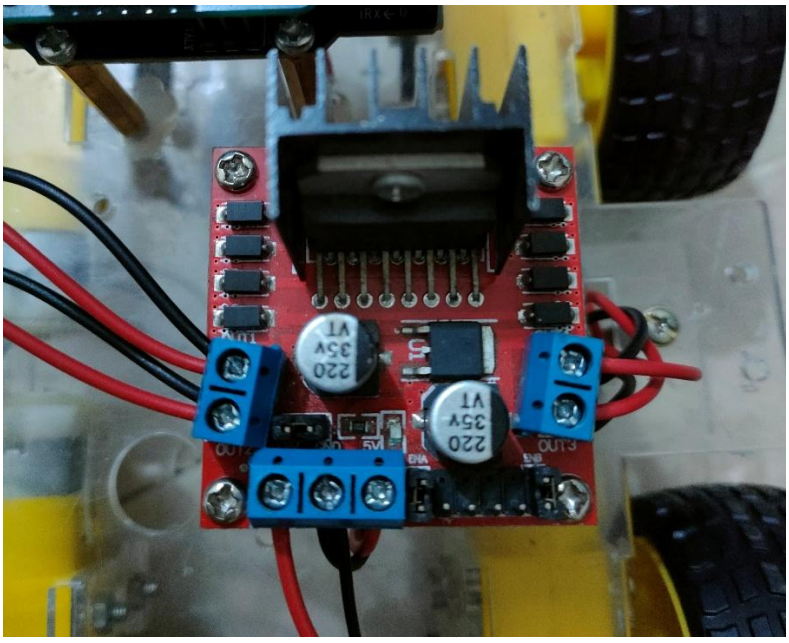


Fig. 4.18: Mounting Motor Driver

(7) Installing the Tracing Module

Attach the tracing module in front of the car chassis's lower part by glue gun, sensing part need to be downward as shown in the figure 3-21.

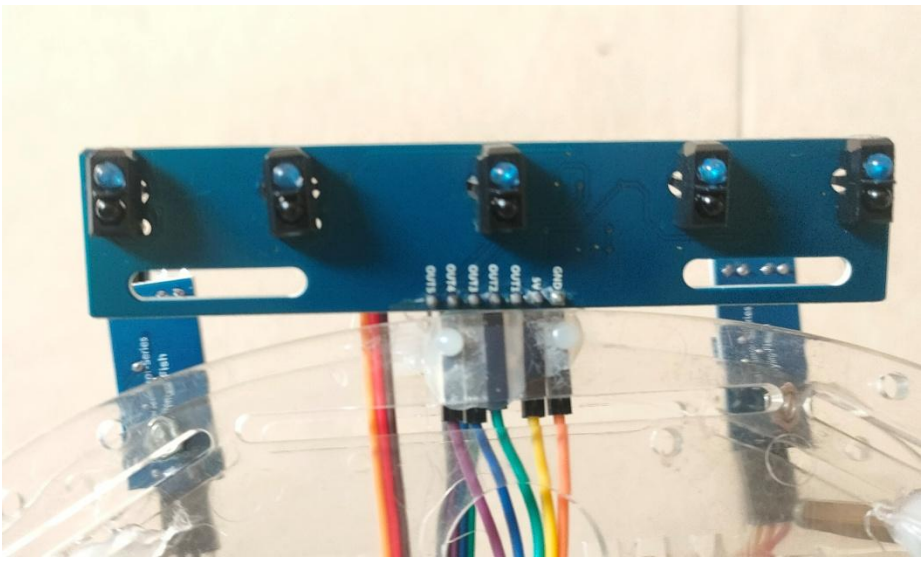


Fig. 4.19: The Back of Complete Tracing Module Installation

(8) Infrared obstacle avoidance module

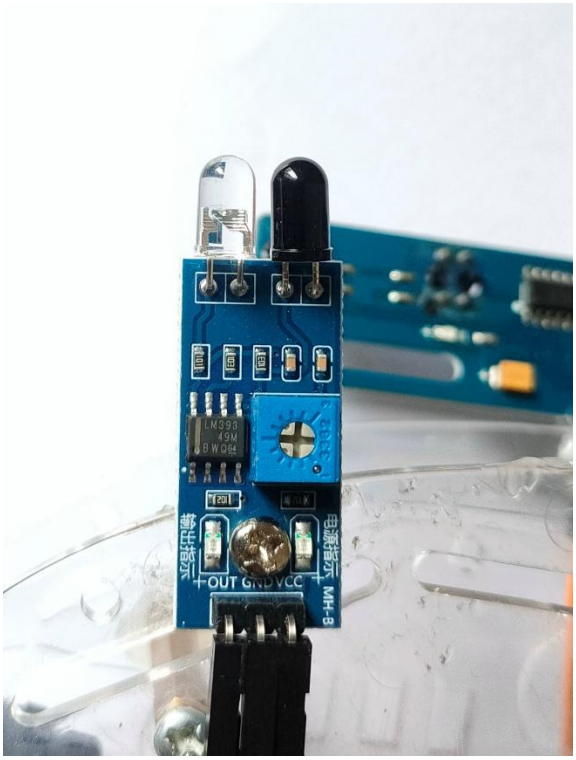


Fig. 4.20: Infrared Obstacle Avoidance Module Mounting

(9) Pan and Tilt Stand and Servo Motors Mount

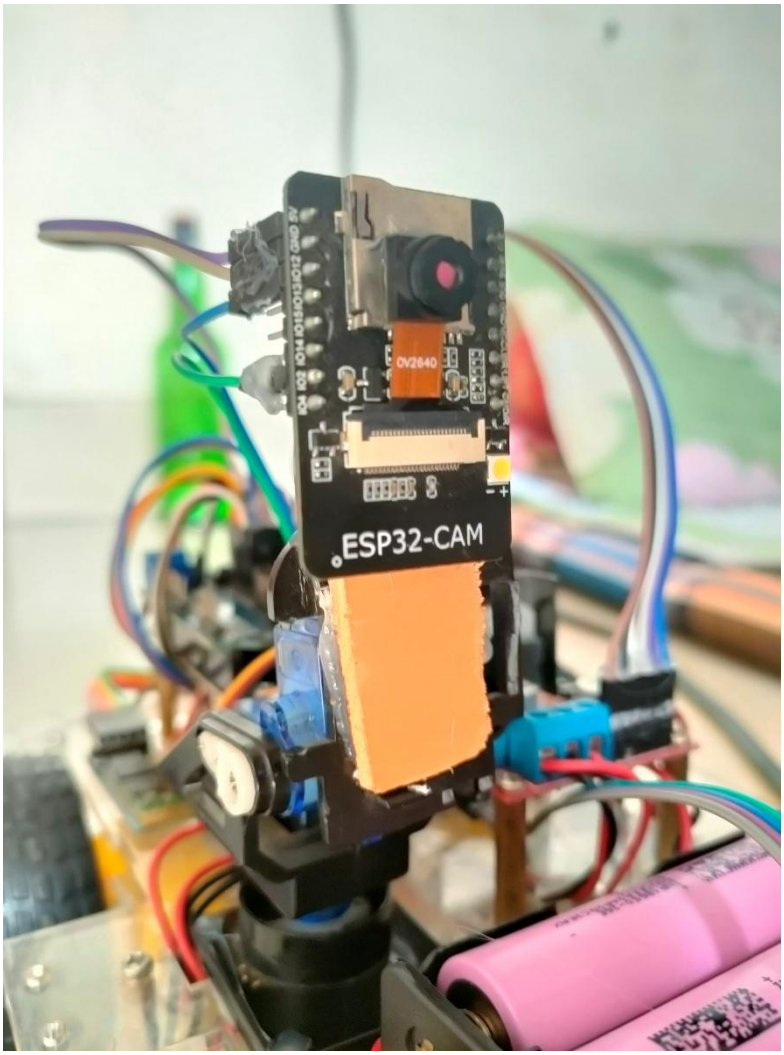


Fig. 4.21: Pan and Tilt Stand and Servo mount

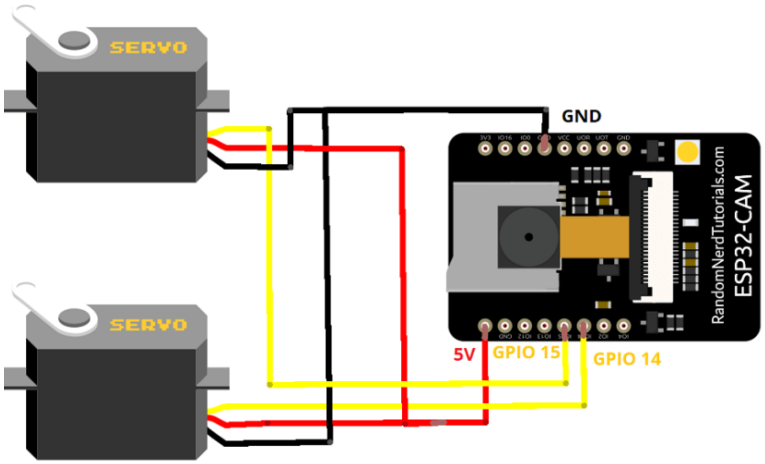


Fig. 4.22: ESP Connection with Servos

(10) Ultrasonic Module Mounting with Servo

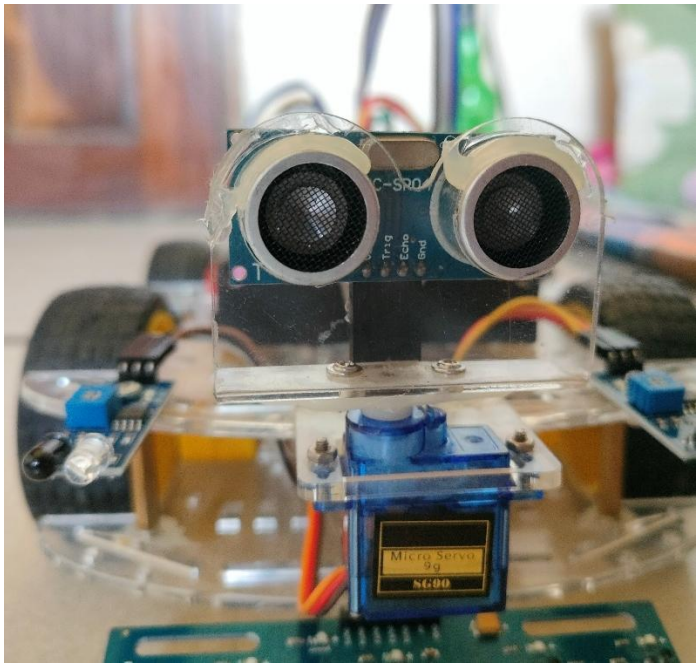


Fig. 4.23: Ultrasonic Module Mounting with Servo

(11) Arduino UNO R3 Mount

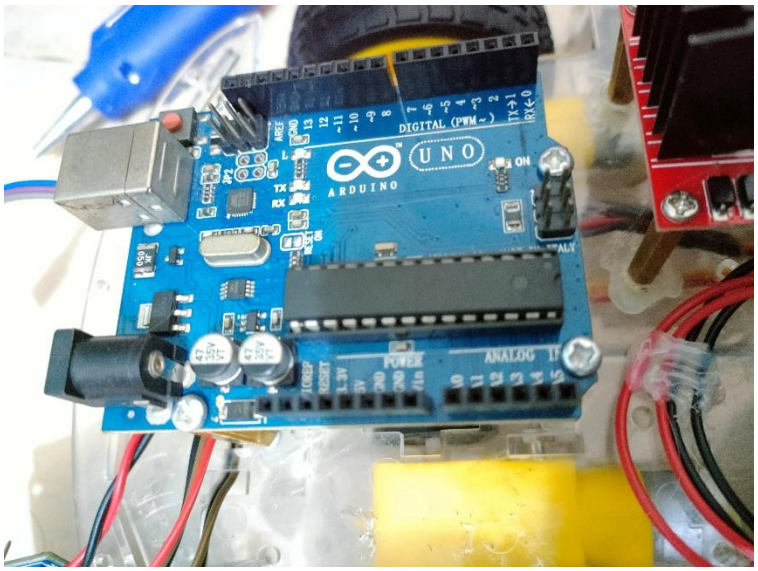


Fig. 4.24: Mounting Arduino UNO R3

4.2 Results and Analysis

Except for ultrasonic mapping, most of the objectives have been successfully achieved which still need further refinement as the test results show that it is still far from perfect. However, it did manage to draw the required shape. For more accurate results, the number of readings can be increased. It will have more scan points, and a higher resolution of the shape can be detected.

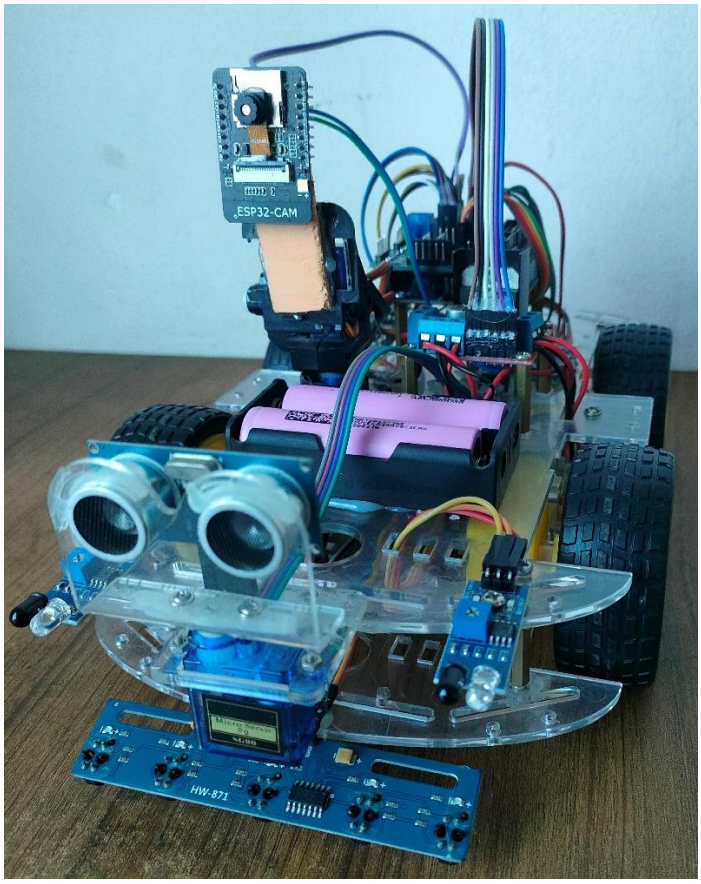


Fig. 4.25: Complete Robot Car

During the operation of the robot, the ultrasonic sensor transmits an ultrasound wave to the front position (85 degrees), right position (10 degrees), and left position (170 degrees). When the wave hits an obstacle, it recedes and accumulates distances for the front, right and left positions. Next, the microcontroller compares the values based on its algorithm and determines whether to proceed or change course. Tests performed on the final hardware revealed the limitations of the detection algorithm. The limitations were related to the inability to detect certain barriers, and this was the result of not being able to measure the sensor barriers outside the sensor measurement range. When an object is in the path of a vehicle and this object is not in the sight of the sensor, it cannot be detected which will lead to a collision. To avoid this, the test was performed in a more enclosed area where the wall was the only obstacle, and the car was able to move freely without collision. To implement a vehicle that will detect multiple obstacles and avoid them, more sensors will need to be used to cover a wider range of obstacle detection.

The continuous current that the shield can deliver is dependent on how the shield is kept cool. The printed circuit board is designed in such a way to draw heat out of the motor driver chips but performance can still be improved by the addition of heatsinks. As the motor driver chip temperature approaches its limit, less power is delivered to the motor which would result in motors not being able to rotate. The shield was tested at room temperature with no forced airflow or heat sinks, and it was able to deliver continuous current to the four motor channels before the thermal protection started reducing the current. The shield delivered enough power to move the car for about 20-30 minutes, after which the power was insufficient for the motors.

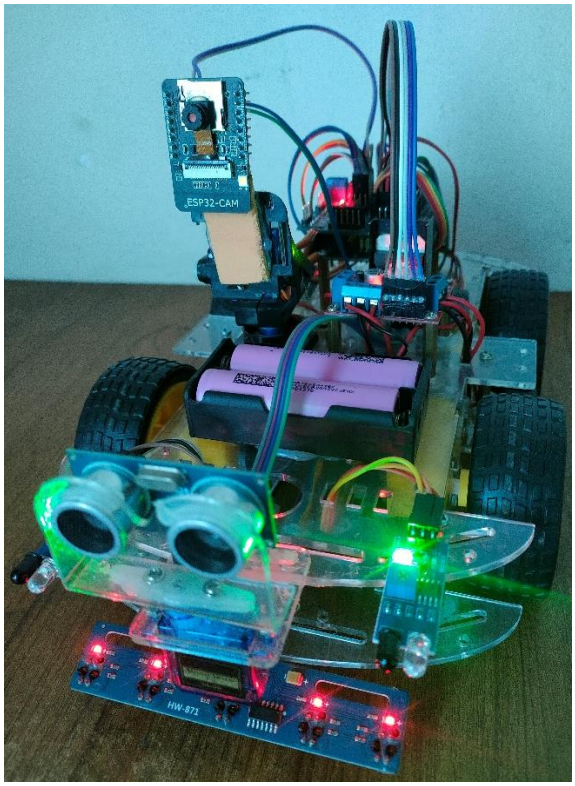


Fig. 4.26: The Robot Car after Powering Up

There is a lot of pressure on batteries coming from components, especially motors. Motors require a lot of power for their operation, especially the low efficiency of cheap motors. The motor driver's shield is driven separately from the Arduino. The Arduino needed a 9-12V supply and was powered by two (3.7 V * 2 = 7.4V) batteries while the motor driver shield required 5-36V and was supplied with the same battery connected in series. Motors, servo motors, and sensors each require 3-6V. The power source of this vehicle can also be changed to maintain long term use.

The reason for using Bluetooth here is that all smartphones are equipped with a Bluetooth module and the Bluetooth module may not be interrupted so easily as Bluetooth is set to only bond with the smartphone itself. However, for future improvement, it can be changed to a Wi-Fi connection.

Since this is a prototype, the car used is not perfect. To travel on a closed road like a cave or hilly area, the car must be modified according to such purpose. The current car works well under normal conditions. In the cave, it may not be able to move, and the car may not be equipped with suspension for traveling on bumpy roads.

The project uses a 2-megapixel camera, the camera's resolution decreases at night which can be changed with a high-resolution camera for a better understanding of unknown areas. A night-vision camera can be installed to view the area at night.

CHAPTER 5

CONCLUSION AND FUTURE WORKS

5.1 Conclusion

In this paper, an embedded robot car system based on Arduino control is designed, and the robot four-wheeled car is selected as a typical representative for research. The overall scheme of the robot car is designed, and the structure and driving mode of the car body are analyzed and compared. At the same time, the car detection system is studied, and it is determined that the infrared photoelectric sensor is used to realize the tracking function, and the ultrasonic and infrared distance measurement sensors are used. Comprehensive multiplexing is used to collect distance information to realize the automatic obstacle avoidance function of the car, and the opposite-beam infrared sensor is selected to complete the speed detection function. The software and hardware design of the robot car is completed. The necessity of establishing a system model of the robot car is analyzed, the modeling process of the kinematic model and the dynamic model is expounded, and the influence of these two models on various performance parameters of the car during operation is analyzed.

5.2 Future Works

Much can be done to improve the performance of this robot for further work. A fully autonomous robot should be able to adapt to the environment (land, underwater, underground, space) as this is very important in robotics. It should be able to operate for extended periods without external interference to avoid harmful situations and may move all or part of its own throughout the operating environment. Boosters can be installed to increase the speed of robotic vehicles. To further diversify the vehicle, machine-learning algorithms can be applied where robots will record video or take pictures, and those images can be analyzed for self-determination. Gas sensors, mine detectors, and radioactivity detectors can be added to detect hazardous environments. The car can be used to help in dangerous situations. In such situations, additional functions such as the robotic arm may be effective.

REFERENCES

[1] Kumar, N.S., Saravanan, M. and Jeevananthan, S., 2011. *Microprocessors and microcontrollers*. Oxford University Press, Inc..

[2] Borenstein, J. and Koren, Y., 1989. Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on systems, Man, and Cybernetics*, *19*(5), pp.1179-1187.

[3] Egerstedt, M., Johansson, K., Lygeros, J. and Sastry, S., 1999, December. Behavior based robotics using regularized hybrid automata. In *Proceedings of the 38th IEEE conference on decision and control (Cat. No. 99CH36304)* (Vol. 4, pp. 3400-3405). IEEE.

[4] De La Croix, J.P. and Egerstedt, M., 2014, June. Flipping the controls classroom around a MOOC. In *2014 American Control Conference* (pp. 2557-2562). IEEE.

[5] Amir, A., Aadil, A., Abhishek, D., Shahid, K. and Dip, A., 1987. Line follower and obstacle avoidance bot using Arduino. *International Journal of Advanced Computational Engineering and Networking*, 2(8), pp.740–741.

[6] Adhvaryu, A.D., 2005, April. Obstacle-avoiding robot with IR and PIR motion sensors. In *IOP Conference Series: Materials Science and Engineering* (Vol. 247, pp. 529-551).

[7] Ankit, V., Jigar, P. and Savan, V., 2016. Obstacle avoidance robotic vehicle using ultrasonic sensor, android and bluetooth for obstacle detection. *Int. Res. J. Eng. Technol*, *3*, pp.339-348.

[8] Li, C., Wang, J., Wang, X. and Zhang, Y., 2015, November. A model based path planning algorithm for self-driving cars in dynamic environment. In *2015 chinese automation congress (CAC)* (pp. 1123-1128). IEEE.

[9] Chishti, S.O., Riaz, S., BilalZaib, M. and Nauman, M., 2018, November. Self-driving cars using CNN and Q-learning. In *2018 IEEE 21st International Multi-Topic Conference (INMIC)* (pp. 1-7). IEEE.

[10] Memon, Q., Ahmed, M., Ali, S., Memon, A.R. and Shah, W., 2016, November. Self-driving and driver relaxing vehicle. In *2016 2nd international conference on robotics and artificial intelligence (ICRAI)* (pp. 170-174). IEEE.

[11] Mandal, S., Saw, S.K., Maji, S., Das, V., Ramakuri, S.K. and Kumar, S., 2016, February. Low cost arduino wifi bluetooth integrated path following robotic vehicle with wireless GUI remote control. In *2016 International Conference on Information Communication and Embedded Systems (ICICES)* (pp. 1-5). IEEE.

[12] Barua, V., Neloy, M.A.I., Rahat, S.U., Das, M., Joy, M.S.I., Nahar, N. and Pathak, A., 2020. An ultrasonic line follower robot to detect obstacles and edges for industrial and rescue operations. *International Journal of Computer Applications*, *176*(31), p.31.

[13] Tian, Y. and Du, G., 2019, October. Infrared line following and ultrasonic navigating robot with ATMEGA328 Pro. In *2019 IEEE 3rd Advanced Information Management,*

Communicates, Electronic and Automation Control Conference (IMCEC) (pp. 856-860). IEEE.

[14] Jo, W., Kim, J., Wang, R., Pan, J., Senthilkumaran, R.K. and Min, B.C., 2022. SMARTmBOT: A ROS2-based low-cost and open-source mobile robot platform. *arXiv preprint arXiv:2203.08903*.

[15] Mandal, P.K., Kushwaha, P., Goud, V., Gupta, R. and Dave, H., 2025. IOT Based Surveillance Robot. *International Journal on Advanced Electrical and Computer Engineering*, *14*(1), pp.69-75.

[16] Mandal, S., Saw, S.K., Maji, S., Das, V., Ramakuri, S.K. and Kumar, S., 2016, February. Low cost arduino wifi bluetooth integrated path following robotic vehicle with wireless GUI remote control. In *2016 International Conference on Information Communication and Embedded Systems (ICICES)* (pp. 1-5). IEEE.

[17] Vidyadhar, R.P., Mallik, D.S., Kumar, K.U. and Renuka, B., 2024, May. An IoT based obstacle detection and line follower robot using LabVIEW. In *2024 2nd International Conference on Advancement in Computation & Computer Technologies (InCACCT)* (pp. 852-857). IEEE.

[18] Ray, M.A., Sarkhel, M.D., Banerjee, M.S., Dutta, M.P. and Bauri, M.K., IOT Based Distance Measuring Robot.

[19] Tai, W.W., Ilias, B., Shukor, S.A., Rahim, N.A. and Markom, M.A., 2019, November. A study of ultrasonic sensor capability in human following robot system. In *IOP Conference Series: Materials Science and Engineering* (Vol. 705, No. 1, p. 012045). IOP Publishing.

[20] Liu, Y., Fan, R., Yu, B., Bocus, M.J., Liu, M., Ni, H., Fan, J. and Mao, S., 2018, December. Mobile robot localisation and navigation using lego nxt and ultrasonic sensor. In *2018 IEEE international conference on robotics and biomimetics (ROBIO)* (pp. 1088-1093). IEEE.