UESTC 3004:
Team Design Project and Skills

TDPS
Final Report

GLASGOW COLLEGE, UESTC

Team 43
UESTC RedFlag

Content

List of Figures

List of Tables

Abstract

Robots are sophisticated and programmable machines that have gained significant attention across various domains. They are defined as autonomous or semi-autonomous electromechanical devices capable of performing complex tasks. The concept of robots emerged in the mid-20th century, with their development rooted in the field of automation. Over time, robots have evolved from mechanical devices to intelligent systems that integrate sensors, actuators, and advanced control algorithms. The development of robots involves a multidisciplinary approach, drawing upon expertise from fields such as mechanical engineering, electrical engineering, computer science, artificial intelligence, and more. Robots have profound implications for society, with applications ranging from industrial automation to healthcare, exploration, and beyond, transforming the way we work, live, and interact with technology.

In order to better adapt to the rapid development of science and technology, we **electrical engineering students** are expected to possess a wide range of abilities, including strong analytical skills for problem-solving, proficiency in technical knowledge specific to their discipline, effective communication and teamwork skills, adaptability to changing technologies, creativity in finding innovative solutions, and a commitment to lifelong learning. These abilities enable us to tackle complex challenges, work collaboratively, adapt to evolving technologies, and contribute to the advancement of their field throughout their academic and professional careers.

The aim of the Team Design Project and Skills (TDPS) course is to foster teamwork and develop students' abilities through designing and constructing a smart robot. This multidisciplinary course combines theoretical knowledge with practical skills, enabling students to collaborate effectively in the creation of **intelligent robotic systems**. By integrating concepts from various fields, such as mechanical engineering, electrical engineering, computer science, and artificial intelligence, students gain a comprehensive understanding of the complex nature of robot design. Through hands-on projects, students engage in the entire lifecycle of robot development, from analyzing technical requirements and designing efficient algorithms to constructing physical prototypes and implementing advanced control mechanisms. The TDPS course cultivates not only technical proficiency but also essential **skills** in project management, communication, and collaboration, preparing students for real-world challenges in the field of robotics.

Our team, **Team 43 UESTC_RedFlag**, consisted of 10 members from diverse engineering disciplines, including Electrical and Electronic Engineering, Communication Engineering, and Microelectronic Engineering. Throughout the collaborative design process of the TDPS course, we undertook the development of a smart car system with eight interconnected modules: PCB & Power, Motion, Mechanical Arm, Ultrasound, OpenMV, Line Tracking, Detection, and Communication. The design encompassed a four-wheel flat structure, integrating these modules to achieve the desired functionality. The operational process involved the PCB & Power module providing the necessary power supply, while the Motion module executed the corresponding movements guided by the Line Tracking and Ultrasound modules. The Mechanical Arm module facilitated the release of objects, and the OpenMV and Detection modules enabled visual recognition and object detection. Finally, the Communication module established wireless communication with external devices. Although our project achieved relative success, further improvements were identified, particularly in enhancing the accuracy of visual recognition and ensuring stable performance during movement. Through effective collaboration and the application of our interdisciplinary expertise, we **successfully designed and developed the smart car system**, exemplifying the integration of engineering principles in the context of **an integrated electronic design project.**

Acknowledgement

We extend our utmost gratitude to the University of Glasgow (UoG) and the University of Electronic Science and Technology of China (UESTC) for providing us with the invaluable opportunity to undertake our smart car project as part of the Team Design Project and Skills (TDPS) course.

We would like to express our sincere appreciation to our esteemed course instructors, Dr. Abdullah Al-Khalidi, Dr. Wasim Ahmad, and Dr. Oluwakayode Ontreti, for their exceptional guidance, mentorship, and unwavering support throughout the project. Their profound expertise, insightful feedback, and dedication to our academic and professional development have been instrumental in shaping the direction and success of our project.

We would also like to acknowledge the contributions of the General Teaching Assistants (GTAs) who provided invaluable assistance and support during the course. Their prompt responsiveness, technical acumen, and willingness to aid us have significantly contributed to the smooth execution of the project.

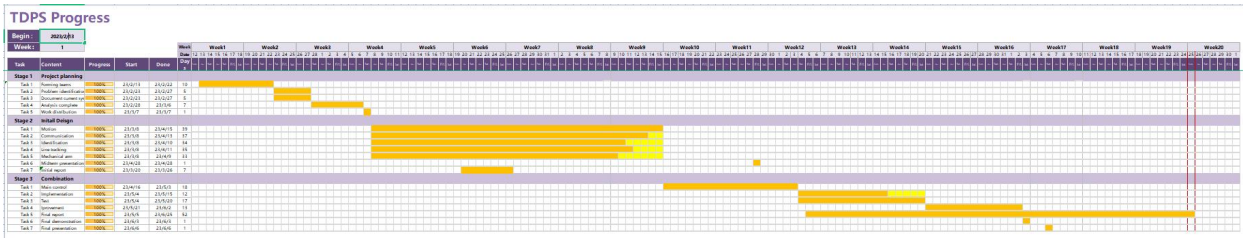
Furthermore, we would like to express our sincere gratitude to our dedicated team members from various engineering disciplines who played a pivotal role in the successful completion of this project. The collaborative efforts, expertise, and commitment of each team member were instrumental in overcoming challenges and achieving our project goals. Our collective contributions greatly enriched our learning experience and contributed to the overall success of the smart car project.

Work distribution

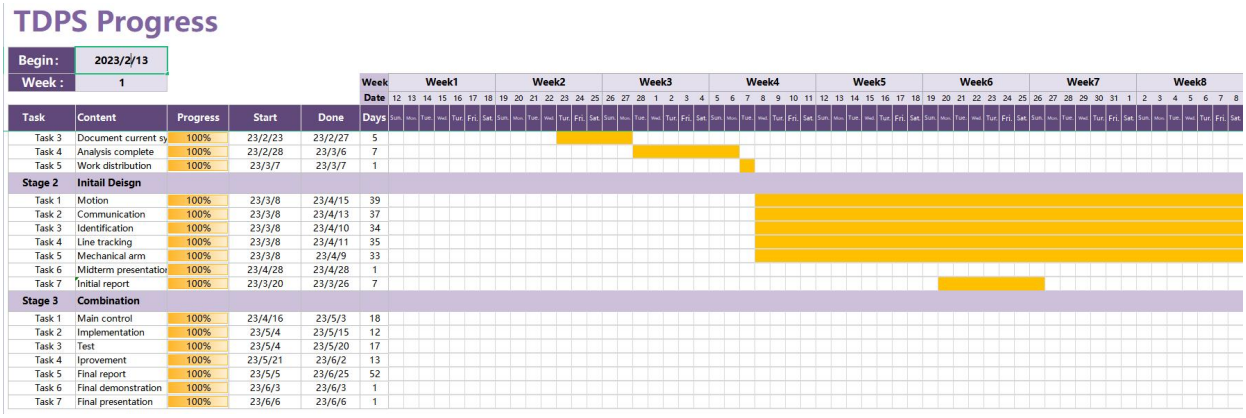
Work	Name	UoG
Main Control	Wang Huiyuan	2614012W
PCB&Power	Qu Zihan	2614274Q
Motion control	Luo Wenjun	2614226L
Mechanical Arm	Li Shanshan	2614182L
Ultrasound & OPenMV Hardware	Li Jingyuan	2614262L
OpenMV-Line Tracking	Liu Daichen	2614175L
Openmv-Shape Matching	Zeng Zihang	2614050Z
Wireless communication	Zhao Jiaxin	2614041Z
Patio 1 implementation	Huang Yicheng	2614272H
Patio 2 implementation	Wang Hongjing (Team leader)	2614026W

Gantt Chart

Total:



From week1 to week9:



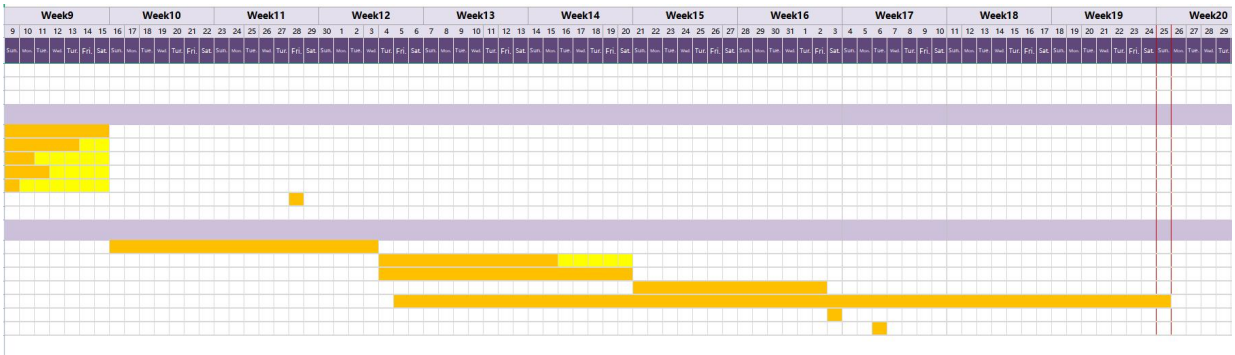
TDPS Progress

Begin: 2023/2/13

Week : 1

Task	Content	Progress	Start	Done	Days
Task 3	Document current sy	100%	23/2/23	23/2/27	5
Task 4	Analysis complete	100%	23/2/28	23/3/6	7
Task 5	Work distribution	100%	23/3/7	23/3/7	1
Stage 2	**Initail Deisgn**				
Task 1	Motion	100%	23/3/8	23/4/15	39
Task 2	Communication	100%	23/3/8	23/4/13	37
Task 3	Identification	100%	23/3/8	23/4/10	34
Task 4	Line tracking	100%	23/3/8	23/4/11	35
Task 5	Mechanical arm	100%	23/3/8	23/4/9	33
Task 6	Midterm presentatio	100%	23/4/28	23/4/28	1
Task 7	Initial report	100%	23/3/20	23/3/26	7
Stage 3	**Combination**				
Task 1	Main control	100%	23/4/16	23/5/3	18
Task 2	Implementation	100%	23/5/4	23/5/15	12
Task 3	Test	100%	23/5/4	23/5/20	17
Task 4	Iprovement	100%	23/5/21	23/6/2	13
Task 5	Final report	100%	23/5/5	23/6/25	52
Task 6	Final demonstration	100%	23/6/3	23/6/3	1
Task 7	Final presentation	100%	23/6/6	23/6/6	1

From week10 to week20:



1. Introduction

1.1 Course Aim:

The aim of the Team Design and Project Skills (TDPS) course is to develop students' academic and logical abilities in order to cultivate their teamwork, problem-solving, and project management skills within the context of electronic and communications engineering. During the course, ten students form three majors, which are EEE, CE, and ME, will form a team to build a smart car. By the end of the course, students should be equipped with the necessary knowledge and practical skills to effectively collaborate in teams, analyze technical requirements, design and construct electronic systems, manage project budgets and timelines, and produce concise and well-researched technical reports. Furthermore, the course aims to enhance students' command of the English language, including complex usage, and their ability to comprehend and critically evaluate scientific and engineering articles. The ultimate objective is to prepare students for the demands of the corporate environment and the rapidly evolving electronics industry, ensuring their readiness to tackle real-world challenges and contribute effectively to their respective fields.

1.2 Problem Analysis

The designed intelligent vehicle is required to finish tasks in two patios.

1.2.1 Patio 1

In patio 1, there are three specific tasks outlined in **Figure 1.2.1**. The initial task requires the car to autonomously follow a black line from the starting point (marked in green) on the left side to the bridge, represented by a blue line. The second task involves crossing the bridge, which is equipped with wire mesh on its surface and slopes on both sides, as depicted in **Figure 1.2.2**. The car should be able to detect the bridge, automatically turn right, and navigate across it. In the third task, after successfully crossing the bridge and making a left turn, there is a U-turn arch followed by a gate. The gate as shown in **Figure 1.2.3** is colored purple, and the car should be able to recognize it, pass through it, and ultimately come to a stop in front of the red box, indicating the completion of the course.

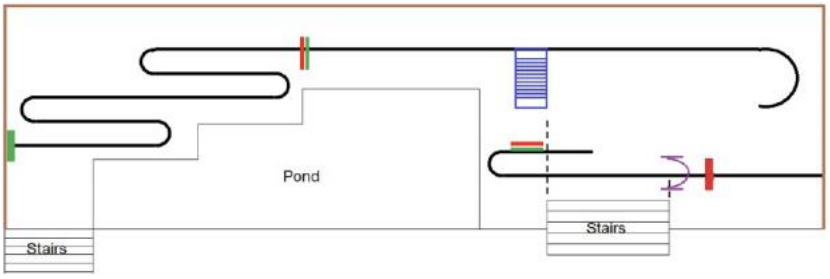


Figure 1.2.1. The schematic diagram of Patio 1

Figure 1.2.2. Shape and dimensions of the bridge for Task 2 in Patio 1

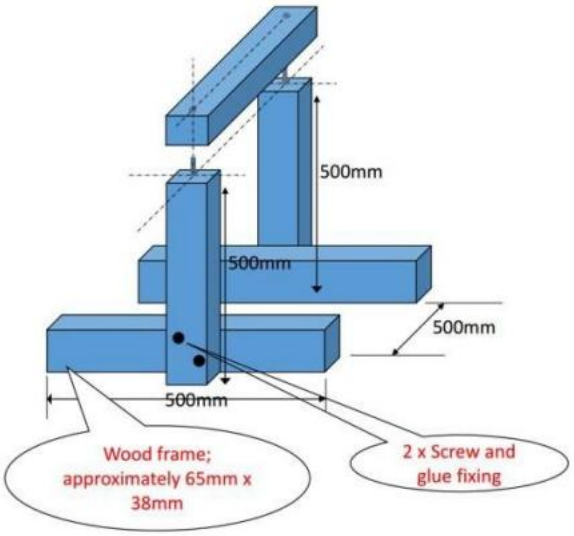


Figure 1.2.3. Shape and dimensions of the arch for task 3

1.2.2 Patio 2

According to **Figure 1.2.4**, track 2 is designed with three primary tasks. The first task requires the smart car to accurately recognize a predefined shape. Once identified, the car should automatically navigate in a specific direction designated for that shape. In the second task, the car is expected to autonomously follow a designated path, reach a specific position, identify a basket, and successfully throw a small ball into the basket. Finally, for the last task, the car needs to enter a planter area and transmit specific information to the computer system.

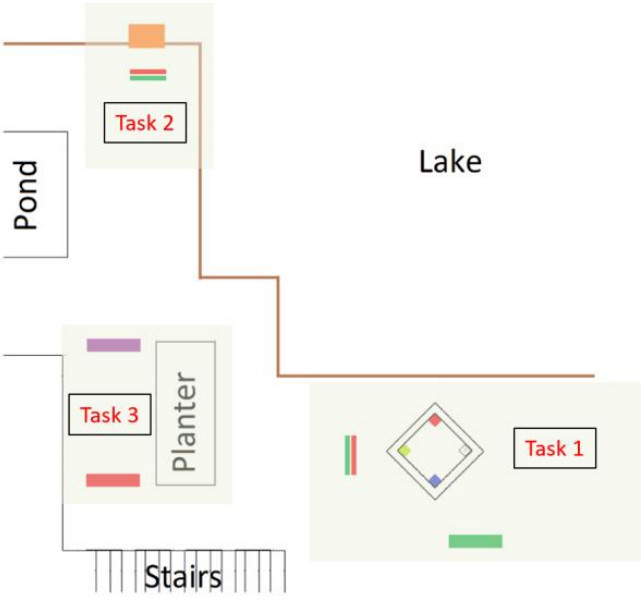


Figure 1.2.4. The schematic diagram of patio 2

1.3 Research Focus

The **research focus** of the TDPS course lies in the design and construction of a smart car system, which is an integrated electronic design project. The **objective** is to create a sophisticated and functional smart car that can perform specific tasks within given constraints. The project requires extensive research and development in various areas. The team's research encompasses the analysis of technical requirements, leading to the formulation of a comprehensive design plan for the smart car. The focus is on developing and implementing a robust and efficient hardware system, encompassing modules such as PCB & Power, Motion, Mechanical Arm, Ultrasound, OpenMV, Line Tracking, Detection, and Communication. The team explores cutting-edge technologies and techniques to enhance the accuracy of visual recognition, improve stability during movement, and optimize the overall performance of the smart car. Through a collaborative approach, the team coordinates the interconnection and communication between these modules, ensuring seamless integration and effective functionality. Additionally, the team conducts research on project budget management, utilizing project planning methodologies to define milestones and measure achievement. The **ultimate goal** is to create a fully functional smart car that can autonomously track lines, identify objects, drop a ball, and communicate with a controller.

1.4 Literature review

Smart car design relies heavily on visual perception, particularly through cameras, for safe and efficient driving. Two articles address the significance of cameras and the challenges of solar interference in image processing. Qingyao Tian's paper discusses grayscale image processing methods to mitigate solar interference[1]. Yawan Zhang's paper explores the use of the OpenMV vision system, enabling path planning and environment perception in smart cars[2].

Qingyao Tian's research focuses on reducing solar interference in grayscale images by applying image processing techniques like gray transform and contrast enhancement. This enhances path extraction accuracy and provides reliable input for smart car path planning. Yawan Zhang's article introduces the OpenMV vision system, an open-source system based on computer vision and machine learning. It enables smart cars to understand road conditions precisely, improving safety and efficiency.

These studies highlight the crucial role of cameras in smart car design and the need to address solar interference. The proposed techniques and the integration of the OpenMV vision system contribute to the advancement of visual perception capabilities in smart cars. Future research can explore advanced filtering methods and the integration of multiple sensors to enhance visual perception. Combining the OpenMV system with LiDAR and **radar** technologies could lead to comprehensive and robust smart car designs. Ultimately, these advancements aim to create safer and more efficient transportation systems.

2. Overall System Design Approach

2.1 Overall structure

The structure of the car, as shown in **Figure 2.1.1(left)**, consists of four layers from top to bottom.

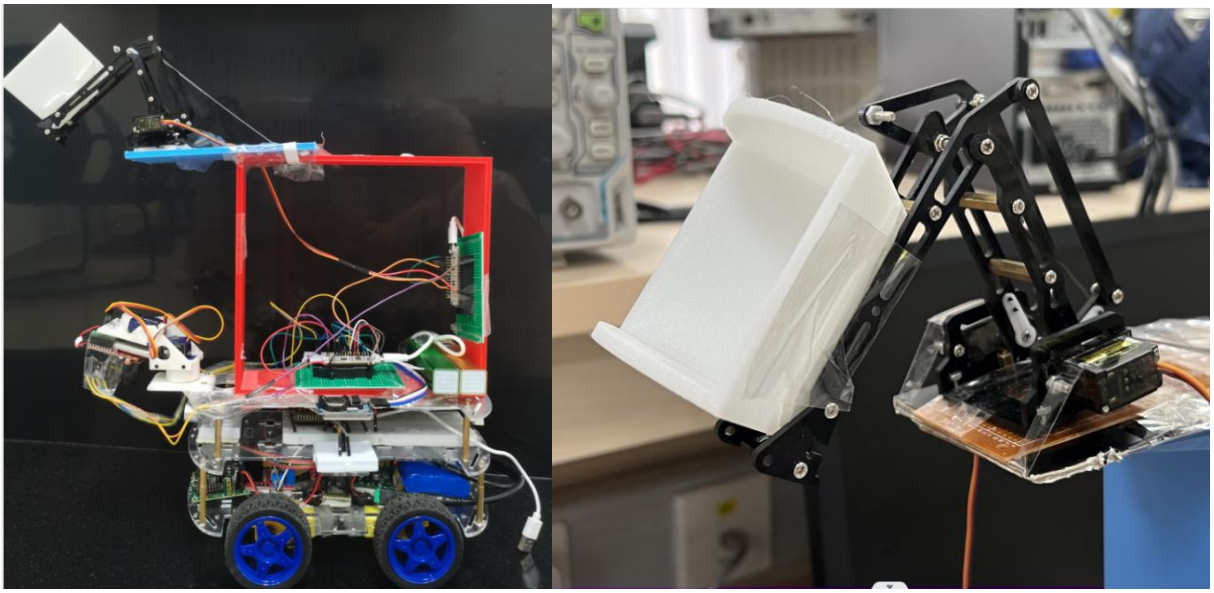


Figure 2.1.1. The structure of the car (left) and the 1st layer: mechanical arm (right)

1st layer (**Figure 2.1.1(right)**): Mechanical arm. This layer is responsible for picking up objects. The mechanical arm is designed to provide flexibility and precision in its movements.

2nd layer (**Figure 2.1.2(left)** and **Figure 2.1.3(right)**): Communication module and OpenMV module. The communication module enables the car to establish connections and communicate with thePC. The OpenMV module is equipped with a camera that provides visual perception capabilities to the car, allowing it to detect and recognize objects or patterns in its surroundings.

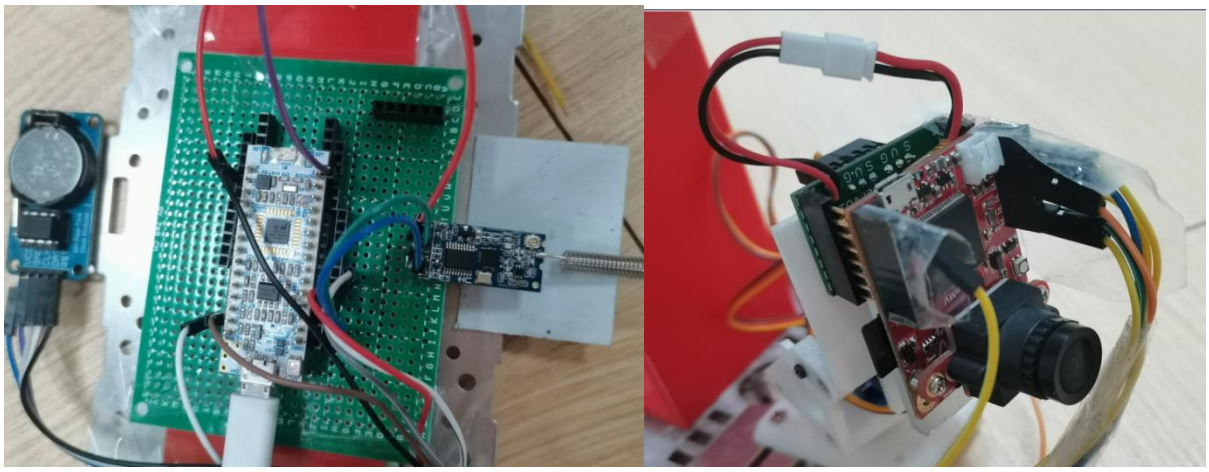


Figure 2.1.2. The 2nd layer: communication module and the 2nd layer: OpenMV module (right)

3rd layer (**Figure 2.1.3(left)**): Three ultrasound modules. These modules use ultrasonic waves to measure distances and detect obstacles. By utilizing the ultrasound technology, the car can navigate and avoid collisions with objects or obstacles in its path.

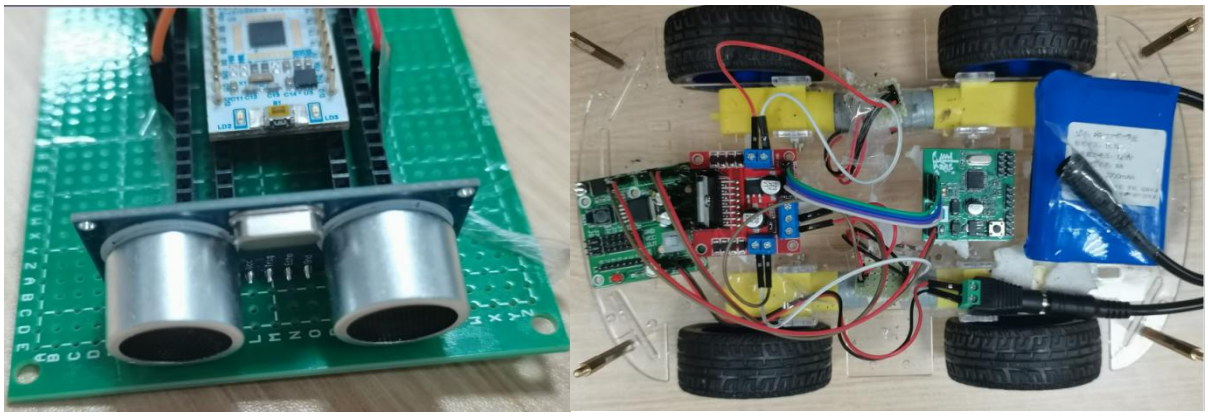


Figure 2.1.3. The 3rd layer: ultrasound modules (left) and the 4th layer: Motion control (right)

4th layer (**Figure 2.1.3(right)**): Motion control. This layer incorporates the necessary components for controlling the motion of the car, such as motors, wheels, and related circuitry. It enables the car to move in different directions, change speed, and execute predefined or user-defined motion sequences.

Overall, the car's multi-layered structure enables it to perform a wide range of tasks and functions. The mechanical arm provides dexterity, the communication offers communication capabilities, the and OpenMV modules enable the car to detect pattern, the ultrasound modules enhance navigation and obstacle avoidance, and the motion control layer ensures precise and controlled movement. This integrated design allows the car to operate effectively in various scenarios and applications.

2.2 Main control

Handled by: Wang Huiyuan
UoG: 2614012w

The vehicle adopts a four-wheel drive structure as shown in **Figure 2.2.1(left)**.The initial design of the smart car is a four-wheel flat car with a four-wheel drive structure, which includes five modules: motion module, mechanical arms module, visual recognition module, and wireless communication module as shown in **Figure 2.2.1(right)** However, during the practical implementation, we discovered the need for further subdivision of the recognition module, leading to the addition of the ultrasonic module. The final design consists of the vision module, communication module, mechanical arm module, ultrasonic module, and motion module, as shown in **Figure 2.2.2**.

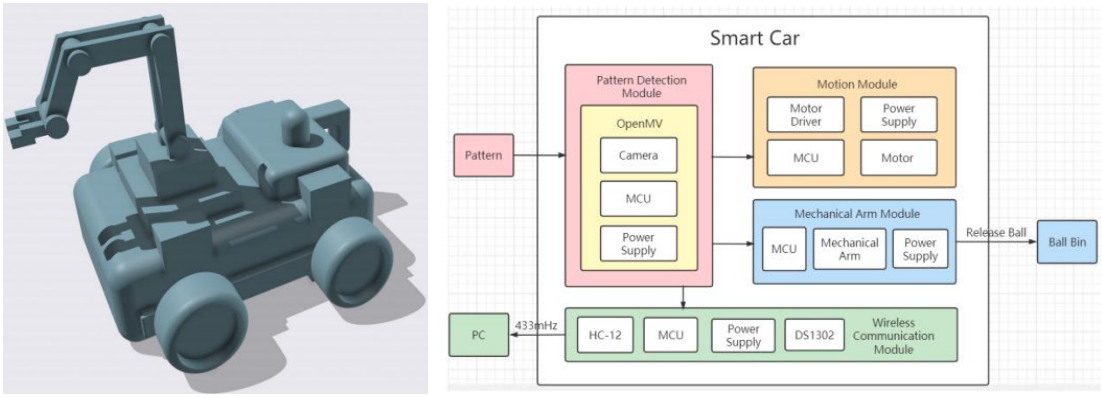


Figure 2.2.1. The concept of the car (left) and the initial structure of the smart car (right)

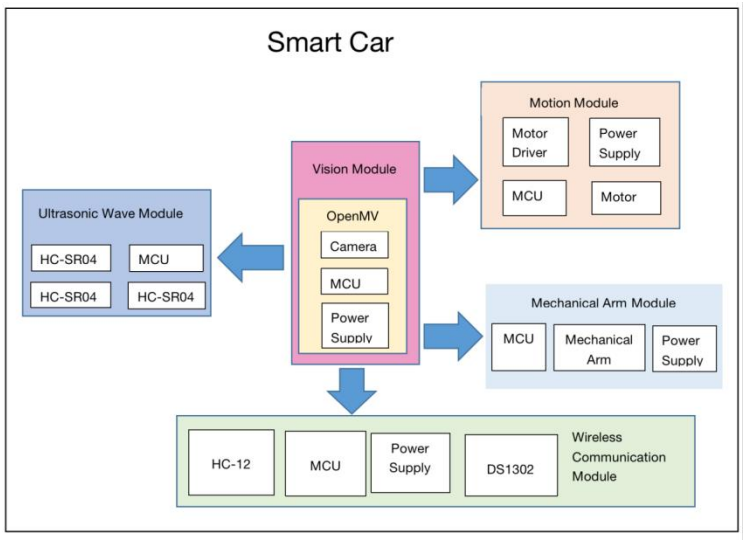


Figure 2.2.2. The final structure of the smart Car

The Motion Module is the driving module of smart car, which is responsible for controlling the whole car's power drive system, including the car's moving, steering and other actions. It receives instructions from other modules and controls the car's motors and wheels according to the instructions, achieving precise motion control.

The Vision Module plays a pivotal role in the functionality of smart cars, as it is responsible for route and arrow recognition. Utilizing advanced image processing algorithms, this module can accurately identify roads and

navigation signs to provide precise navigation information for smart cars.

The Communication Module plays an important role in smart cars, which is responsible for communicating with computers and transmitting information such as team and time. Through wireless communication with computers, smart cars can transmit data in real time and interact with external systems to achieve more advanced functions and tasks. This module is implemented combining the HC-12 and DS1302, which are employed on one MCU.

The Mechanical Arm Module is an additional function module of the smart car, which is responsible for the delivery of table tennis. The module can grasp the ping-pong ball accurately and put it into the designated position through high precision motion control. The design of the mechanical arm module enables the smart car to have more flexible and diverse functions, which can adapt to different use scenarios and task requirements.

The Ultrasonic Module is one of the sensing modules of the intelligent car, which is responsible for detecting the distance around the car body and the obstacles in the front, left and right directions. By transmitting and receiving ultrasonic waves, smart cars can obtain real-time information about their surroundings, enabling them to avoid collisions and drive safely.

Additionally, to ensure independent power supply for each module, each module of the smart car is equipped with a separate battery. The goal is to ensure that each module operates efficiently and reliably, avoiding interference between different modules. In addition, all modules of the vehicle are shared to ensure good electrical connection and signal transmission.

Communication between Modules

The Communication between Modules is realized through OpenMV camera and single chip microcomputer. Each module exchanges and controls information with OpenMV camera, as shown in **Figure 2.2.3**, and SCM. The basic logic of the whole system is that the measurement module (such as the OpenMV camera) sends the information to the MCU, which controls the drive and throwing module according to the received information to perform the corresponding actions.

Figure 2.2.3. OpenMV

The operational process of the system involves several interconnected steps, enabling seamless functioning and communication between its various components. The OpenMV acts as the controller, which is aimed to communicate with other functional modules and transmit the desired instructions to control the motion. The OpenMV has a set of SPI interfaces and is also equipped with an I2C bus, a CAN bus, and two asynchronous serial buses (TX/RX) for communication with other modules or sensors. Through these interfaces, the vision

module can realize the data exchange and control command transmission with other modules.

Communication with the motion module is achieved through the universal asynchronous transceiver (UART). OpenMV transmits the command signal through a set of UART transmit pins (TX) to the UART receive pins (RX) of the vehicle body module. In the car module, the microcontroller adjusts the duty ratio of PWM signal according to the received instruction signal. By changing the duty ratio of the PWM signal, the speed and direction of the motor on both sides of the car can be controlled, so as to realize the forward, backward, left and right turn of the vehicle, and adjust the speed as needed.

The communication between OpenMV and the mechanical arm module depends on the high and low level change of the GPIO pin. One of OpenMV's GPIO pins is normally at a low level, but when the arm is required to work, the output of this pin changes to a high level, and after a period of time it is pulled down again. This pattern of pin level change acts as a control signal to inform the manipulator module when to perform an action.

The communication with the ultrasonic module is also realized through UART. The UART sending pins of the ultrasonic module send the distance information to another set of UART receiving pins (RX) of OpenMV. Since OpenMV has only one TX bus and one RX bus, it is used to communicate with the ultrasonic module and the car body module respectively, so their communication does not interfere with each other.

In addition, the system incorporates a wireless communication module. This module serves as the communication link between the system and the PC. It utilizes a 433mHz signal to establish a reliable and efficient wireless connection. Through this communication channel, crucial data, feedback, and status updates are transmitted to the PC for monitoring, analysis, and further processing.

The interboard communication between OpenMV and the communication module is also achieved through the high and low level changes of the GPIO pins. One of OpenMV's GPIO pins is set to a low level output and changes to a high level when a command needs to be sent. The other GPIO pin is set as input, and when the communication module receives the signal, it sends a high level signal to OpenMV. After receiving the high level signal, OpenMV continues to perform the next step. When OpenMV detects that the vehicle has entered the planting area, the clock module reads the clock data and transmits it to the microcontroller. The microcontroller packages the data and sends it to a wireless transmitter, which transmits the information to a computer or other receiving device.

In conclusion, the communication methods between OpenMV and other modules can be summarized in **Table 2.2.1**.

Table 2.2.1. The communication between OpenMV and other modules

Module	OpenMV Pins that send signals	OpenMV Pins that receive signals
Mechanical Arms	GPIO（Low level→High Level）	/
Ultrasonic Wave	TX	RX
Motion	TX	RX
Wireless Communication	GPIO（Low level→High Level）	GPIO（low level→High Level）

Pins Configuration of OpenMV

Referring to [1], **Figure 2.2.4** illustrates the pin diagram of OpenMV.

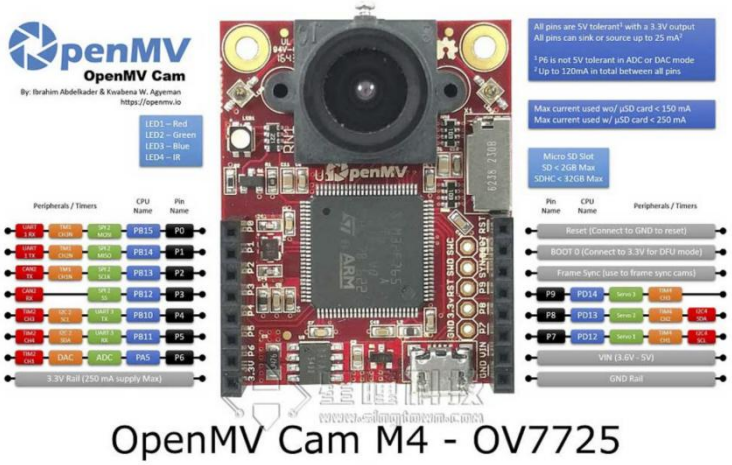


Figure 2.2.4. OpenMV Cam M4-OV7725

The Pin0 and P1 UART1 pins are used for communication of the ultrasonic sensor, they receive data from the sensor and transmit it to the OpenMV board for further processing and analysis. In this way, the OpenMV board can obtain the distance information of objects in the environment in real time, providing accurate data support for subsequent decision-making and control.

Next is the Pin2 pin, which acts as the GPIO pull-down input pin to receive the information returned by the communication module. Communication modules can be wireless modules, Bluetooth modules or other types of data transmission devices. OpenMV board can communicate bidirectional with external devices through this pin to realize data transmission and instruction interaction. This provides a wider range of possibilities for the application of OpenMV boards, which can be linked and collaborated with other devices or systems.

The Pin3 pin is used as the GPIO output pin to control the movement of the robot arm. By controlling the level state of this pin, the OpenMV board can send signals to the robotic arm to control the position and attitude of its end effector. This facilitates robot control and automation tasks, allowing OpenMV boards to play an important role in industrial automation, intelligent manufacturing and other fields.

The Pin4 and Pin5 pins are used to control the movement of the car through the UART3 communication protocol. UART3 is a common serial communication protocol, through the pin, OpenMV board can send control commands to the car, such as forward, backward, left, right turn, etc. This provides flexibility and scalability for applications in areas such as robot navigation and intelligent transportation.

In addition, the Pin6 pin is used as the GPIO output pin to control the start and stop of the communication module. By controlling the level state of the pin, the OpenMV board can control the working state of the communication module and realize the flexible control of communication. This is very important for wireless data transmission, remote monitoring and other application scenarios, you can decide when to start or stop the communication module according to the actual needs, thereby saving energy and improving the reliability of the system.

Finally, Pin7, Pin8 and Pin9 are used to control the motion of the steering gear. Steering gear is a commonly used position control device, which can precisely control the Angle and position. Through these three pins, the OpenMV board can send control signals to the steering gear, achieving precise control of its Angle and position. This provides a flexible mechanical control method for robot motion, camera PTZ control and other applications.

3. Submodules

3.1 Hardware circuit design

Handled by: Qu Zihan
UoG number: 2614274Q

In this section, the hardware circuit of the car was designed. In this design, some key components were selected to meet the voltage requirements and drive the motors effectively. The LM2576-ADJ chip was used to convert the battery's 12V output to a stable 5V voltage level. The LP2992 served as a low dropout voltage regulator for the 3.3V input microcontroller. The motor drive module was utilized the L298N chip, allowing for simultaneous control of a pair of DC motors with precise speed and direction control. The flow chart of the circuit was shown as **Figure 3.1.1**.

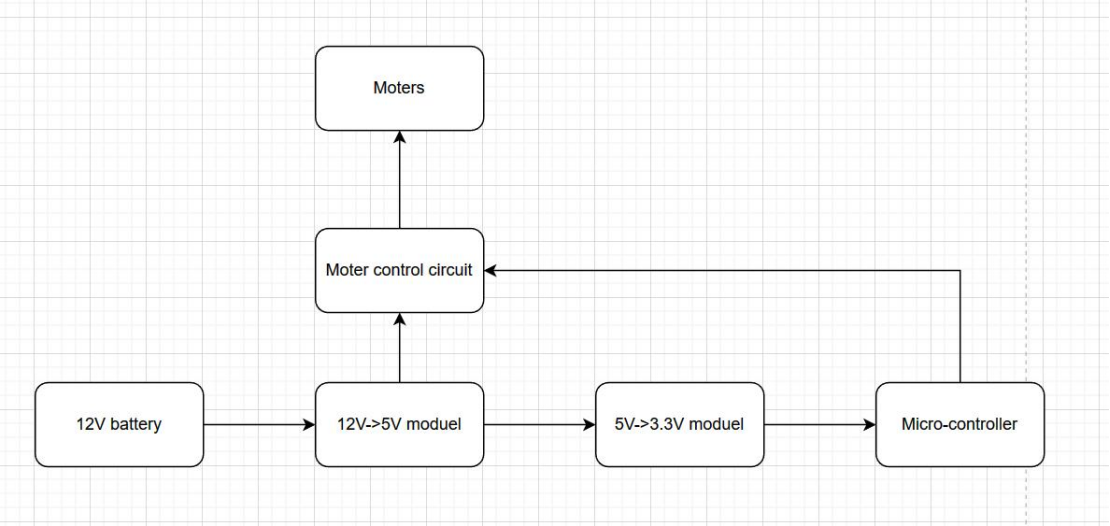


Figure 3.1.1. Overview of the circuit

3.1.1 Power Supply Circuit

To meet the requirements of the application, where a lithium-ion battery supplied 12V and the motors required a 5V voltage supply, a circuit was designed. Additionally, the microcontroller needed a 3.3V voltage supply. The issue of decreasing battery voltage over time needed to be addressed in the circuit design. In this regard, Switching Power Supply Design (Third Edition) was referred to, which provided instructions for designing the circuit, including the choice of capacitors and inductors[1].

3.1.1.1 Circuit for transformation from 12V to 5V

The LM2576-ADJ chip (shown in **Figure 3.1.2**) was selected for this design. It was used to convert the 12V output of the battery into 5V. The LM2576 chip is available in fixed voltage and adjustable voltage versions. In the fixed voltage version, the chip had integrated feedback resistors for voltage regulation, while the adjustable voltage version allowed users to set the output voltage by configuring external feedback resistors connected to the feedback pin[2]. Additionally, there was a high-voltage version of the LM2576 chip, denoted by the HV suffix, which could withstand higher input voltage levels.

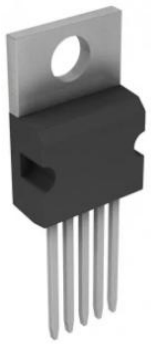


Figure 3.1.2. Real figure of LM2576

According to the datasheet, the circuit can be constructed as **Figure 3.1.3**.

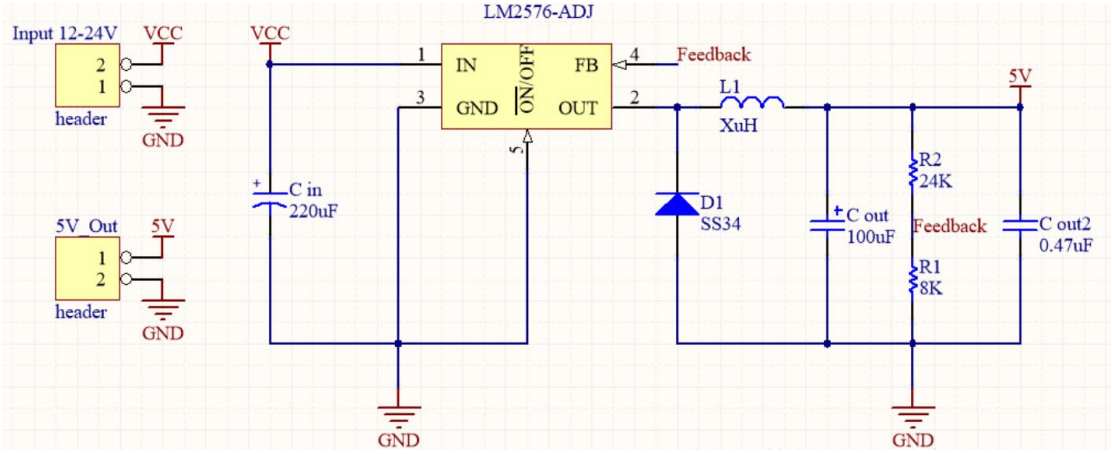


Figure 3.1. 3. Circuit for transformation from 12V to 5V

In the design using the LM2576 chip, the following key points should be considered:

1. Input Capacitor: To stabilize the input power supply and reduce input voltage ripple and noise, it is recommended to place an input capacitor near the LM2576 chip's input pin. According to the data sheet, we choose 100uF.
2. Output Capacitor: To provide stable output voltage and reduce output ripple, it is advised to place an output capacitor near the LM2576 chip's output pin.
3. Feedback Circuit: For the adjustable voltage version of the LM2576 chip, two voltage divider resistors should be configured near the feedback pin to set the desired output voltage. Select appropriate resistor values based on the design requirements and desired output voltage range.
4. Input and Output Power Filtering: To ensure the stability and reliability of the input and output power supply, it is recommended to add suitable power filtering capacitors and inductors (if required) near the LM2576 chip to filter out noise and interference.
5. Thermal Protection: The LM2576 chip typically includes thermal protection features. However, in high-temperature environments or specific applications, additional heat dissipation measures or external temperature sensors may be necessary to ensure the chip operates within a stable temperature range.

6. As for diode, we have chosen not to use common diodes like 1N4007 for two main reasons: 1. Schottky diodes have a lower forward voltage drop. 2. The LM2576 operates at a fixed switching frequency of 52kHz, which is too high for diodes like 1N4007 and may result in delayed turn-off, whereas Schottky diodes can transition from conducting to non-conducting state quickly, making them suitable for operation at high switching frequencies.

Considering these factors, the surface-mount diode SS34, capable of handling a current of 3A and with a voltage rating of 40V, was chosen. Its through-hole counterpart is 1N5822. (SS14 corresponds to 1N5819, with a current rating of 1A and a voltage rating of 40V.) By using the SS34 or 1N5822 diode, efficient and reliable performance in the LM2576 circuit could be ensured, especially at the specified switching frequency.

3.1.1.2 Circuit for transformation from 5V to 3.3V

AMS1117 (shown in **Figure 3.1.4**) is a low dropout (LDO) voltage regulator IC that can be used in the design to meet the voltage requirements of the 3.3V input microcontroller**[3]**.

Figure 3.1.4. Real figure of AMS1117

For the motor's 5V supply, the AMS1117 was used as a step-down voltage regulator. It had a low dropout voltage, ensuring a stable 3.3V output even when the input voltage from the battery decreased over time. According to the datasheet, the circuit could be constructed as shown in **Figure 3.1.5.**

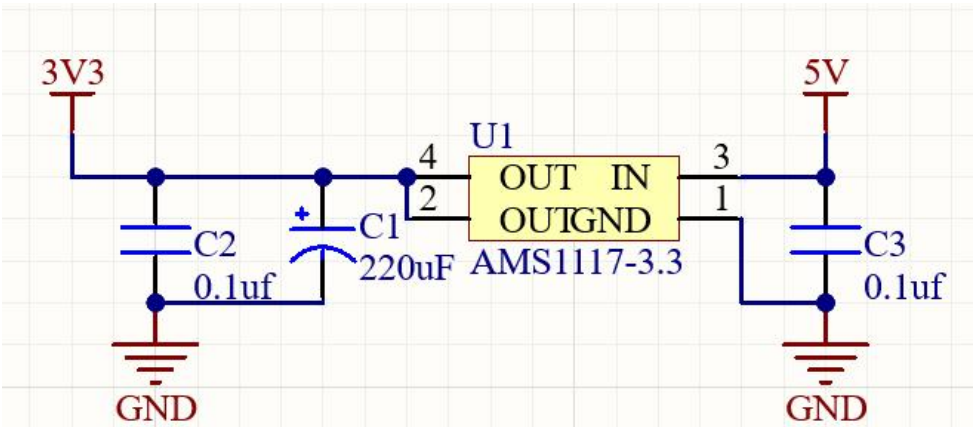


Figure 3.1.5. Circuit for transformation from 5V to 3.3V

With the input voltage of 5V at Vin, the output voltage of 3.3V can be obtained. By incorporating the AMS1117 voltage regulator IC into the circuit design, the required voltage levels for both the motor and microcontroller can be achieved. Furthermore, this design effectively addresses the issue of decreasing battery voltage over time.

3.1.2 Circuit for Motor Drive

The motor drive module utilizes the L298N (**Figure 3.1.6**) as its core component. The L298N chip features a dual-channel H-bridge structure, allowing for the simultaneous driving of a pair of DC motors. Additionally, the L298N supports PWM signal input, enabling precise control over the speed and direction of the motors**[4]**.

Figure 3.1.6. Real figure for L298N

As seen in F**igure 3.1.7**, the L298N contains two H bridge inside. The H-bridge circuit is a typical control circuit for DC motors. It is named after its shape, which resembles the letter "H." The H-bridge consists of four transistors arranged in a configuration that resembles the four legs of the letter H, while the motor represents the horizontal bar of the H. It's important to note that the following diagrams are only schematic representations and do not depict the complete circuit, including the driving circuits for the transistors.

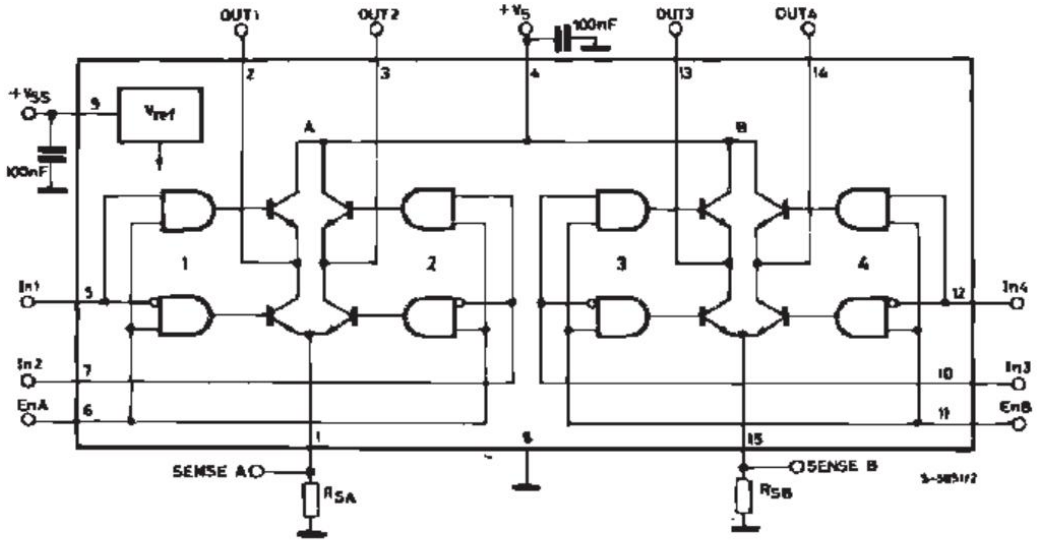


Figure 3.1.7. Internal structure of L298N

A H-bridge motor drive circuit consists of four transistors and a motor. To make the motor operate, it is necessary to turn on a pair of diagonally opposite transistors. Depending on which pair of transistors is turned on, the current may flow from left to right or from right to left through the motor, thereby controlling the motor's rotation direction.

As shown in **Figure 3.1.8(left)**, when transistor Q1 and Q4 are turned on, the current flows from the positive terminal of the power supply through Q1, passes through the motor from left to right, and then returns to the negative terminal of the power supply through Q4. Following the direction of the current arrows in the diagram, this current flow direction will drive the motor to rotate clockwise. As shown in **Figure 3.1.9(right)**, when

transistors Q2 and Q3 are turned on, the current will flow from right to left through the motor, thereby driving the motor to rotate in the opposite direction (indicated by the counterclockwise arrows around the motor in the diagram).

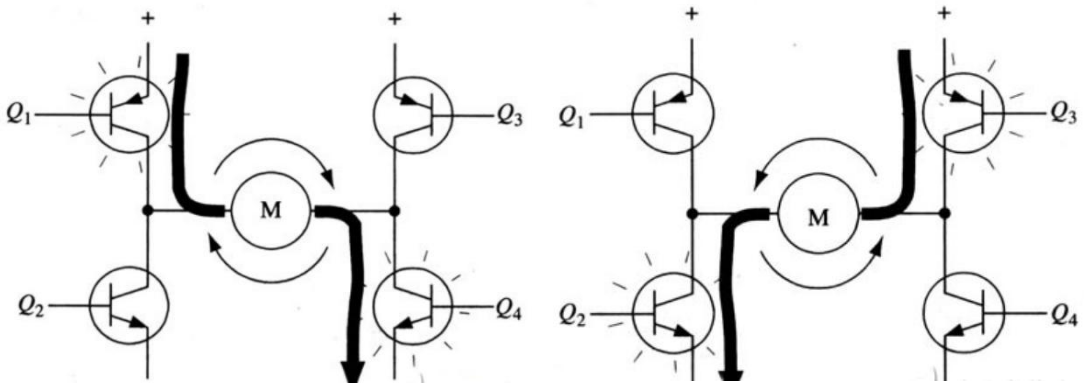


Figure 3.1.8.H bridge when Q1 Q4 are on (left) and H bridge when Q2 Q3 are on (right)

The L298N chip is an excellent fit for our design requirements. It has 15 pins, each serving a specific function. Pin 9 is a 5V logic level input used to power the logic circuit. Pin 4 is the voltage input for powering the motor. Pins 5, 7, 10, and 12 are the input terminals of the H-bridge switch. Pins 6 and 11 are the enable terminals, controlling the left and right sides of the H-bridge, respectively. PWM signals can also be input through these pins. Pins 2, 3, 13, and 14 serve as the output terminals, providing power to the left and right motors. Pins 1 and 15 are feedback pins for the H-bridge. Lastly, pin 8 is connected to ground.

Based on the L298N chip, the motor drive module was designed, as depicted in the schematic diagram shown in **Figure 3.1.9**. This module integrated the necessary functionality and pin configuration of the L298N chip to efficiently drive the motors and enable precise control over their direction and speed.

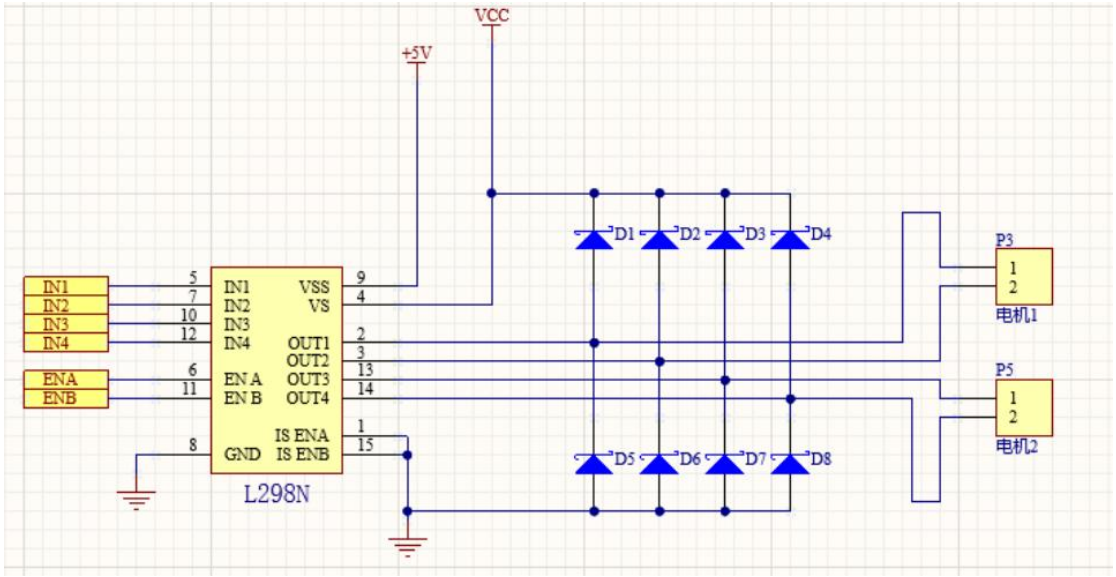


Figure 3.1.9. Circuit for motor drive

3.1.3 Test results for Hardware Circuit

In this section, the results obtained from the hardware circuit experiments will be analyzed and summarized. The experiments were conducted to evaluate the performance of the two power transfer circuits and the motor driving circuit.

3.1.3.1 Test result for 12V to 5V circuit

Here, the test of the circuit based on the LM256 (**Figure 3.1.10**) was conducted. Three sets of inputs were tested, including 12V (the standard output of the battery), 11.8V, and 11.6V. As observed in **Figure 3.1.11-3.1.13**, the output remained at 5.04V, even as the battery voltage decreased due to prolonged use.

Figure 3.1.10. Real circuit for 12V to 5V transformer

Figure 3.1.11. The 12V input (left) and output result (right) of LM256 circuit

Figure 3.1.12. The 11.8V input (left) and output result of LM256 circuit(right)

Figure 3.1.13. The 11.6V input (left) and output result of LM256 circuit(right)

Based on the experimental results, it can be concluded that the designed hardware circuit, which utilized the LM2576-ADJ chip, successfully met the desired requirements for voltage conversion. The circuit demonstrated

stable output voltage, even when the battery's output voltage was slightly less than 12V. These findings validate the suitability of the LM2576 chip and the selected components in the circuit design, confirming its capability to deliver consistent and accurate voltage regulation.

3.1.3.2 Test result for 5V to 3.3V circuit

In this section, the 5V to 3.3V circuit (**Figure 3.1.14**) utilizing the AMS1117 voltage regulator was subjected to testing to evaluate its performance and determine if it met the desired requirements. We gave a 5V input to the circuit and the output result is shown in **Figure 3.1.15**.

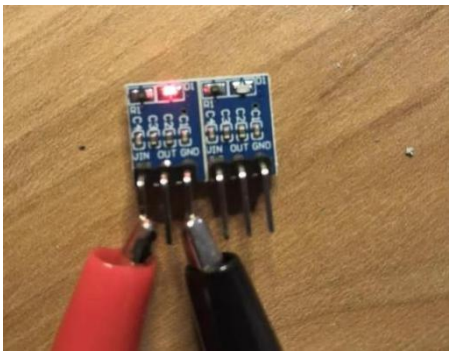


Figure 3.1.14. Real circuit for 12V to 5V transformer

Figure 3.1.15. The input (left) and output result of AMS1117 circuit(right)

Based on these experimental results, it can be concluded that the implemented circuit using the AMS1117 voltage regulator successfully met the desired requirements for voltage conversion, which is to convert the 5V voltage to 3.3V. It's worth to mention that a 0.03V deviation can be accepted.

3.1.3.3 Test result for motor driving circuit

To validate the functionality and performance of the motor drive module utilizing the L298N chip (**Figure 3.1.16**), we conducted experimental testing. The objective was to assess the module's ability to drive the motors efficiently, control their direction, and regulate their speed.

First, the basic operation of the module was verified by applying different input configurations and observing the corresponding motor response. The input-output relationship outlined in **Table 3.1.1** was followed, and the PWM signals and input pins were adjusted accordingly.

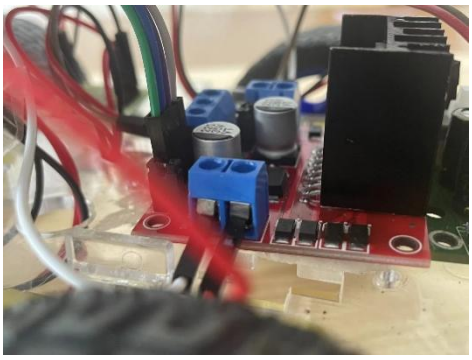


Figure 3.1.16. Real circuit for motor driving circuit

Table 3.1.1. The input-output relationship of the motor control module

ENA	ENB	IN1	IN2	IN3	IN4	Motor1	Moter2
0	0	/	/	/	/	Stop	Stop
PWM	PWM	0	1	0	1	Forward	Forward
PWM	PWM	0	1	1	0	Forward	Backward
PWM	PWM	1	0	0	1	Backward	Forward
PWM	PWM	1	0	1	0	Backward	Backward

The test results are shown in **Table 3.1.1.** During the testing, the following results were observed:

1. Stop Mode: When ENA and ENB were both set to 0, the motors remained stationary, as expected.
2. Forward Direction: By applying PWM signals to ENA and ENB and setting IN1 = 0, IN2 = 1, IN3 = 0, and IN4 = 1, the motors rotated in the forward direction. We varied the PWM signals to adjust the speed, and the module effectively controlled the motors' rotational speed.
3. Reverse Direction: To reverse the motor rotation, we set IN1 = 1, IN2 = 0, IN3 = 1, and IN4 = 0 while applying PWM signals to ENA and ENB. As anticipated, the motors reversed their rotation direction, and we could regulate the speed using the PWM signals.
4. Mixed Direction: By selectively setting the input pins, we were able to achieve mixed direction control. For example, setting IN1 = 0, IN2 = 1, IN3 = 1, and IN4 = 0 resulted in one motor rotating forward while the other rotated in reverse.

Overall, the experimental testing confirmed the successful operation and performance of the motor drive module using the L298N chip. The module effectively controlled the motors' direction and speed based on the applied input signals, aligning with the expected behavior outlined in the input-output relationship.

These results validate the suitability of the motor drive module for our design requirements and demonstrate its capability to drive DC motors efficiently and precisely control their motion.

3.2 Motion

Handled by: Luo Wenjun
UoG:2614226L

In order to make our car can finish the task as required, we need to control the behavior of the left and right motor based on the command from the OpenMV.

3.2.1 MCU PCB Design

To achieve the aforementioned functionality, a specific microcontroller board was designed, as shown in Figure 3.2.1, **Figure 3.2.2**, and **Figure 3.2.3**. The chosen microcontroller unit (MCU) for this design is the STM32F103C6T6 [1,2]. This MCU was selected due to its ability to meet the hardware requirements and its affordability, as it is one of the least expensive MCUs available in the market.

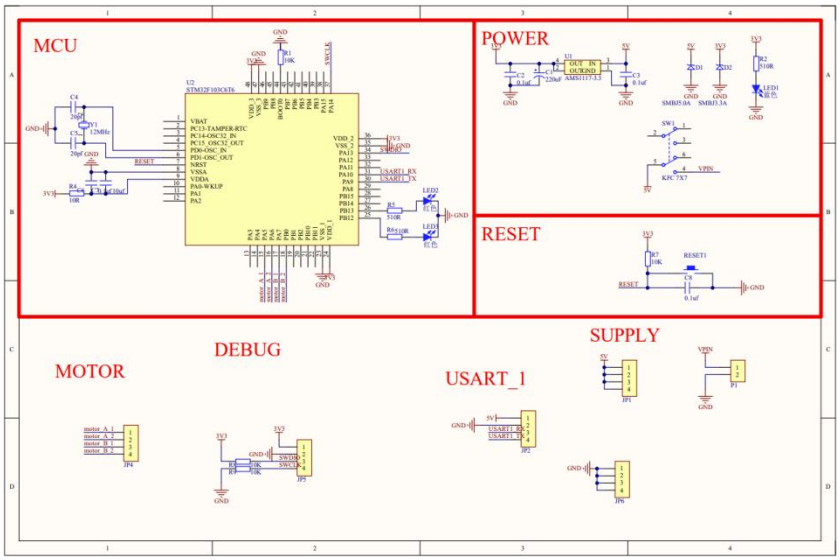


Figure 3.2.1. The schematic diagram of the PCB

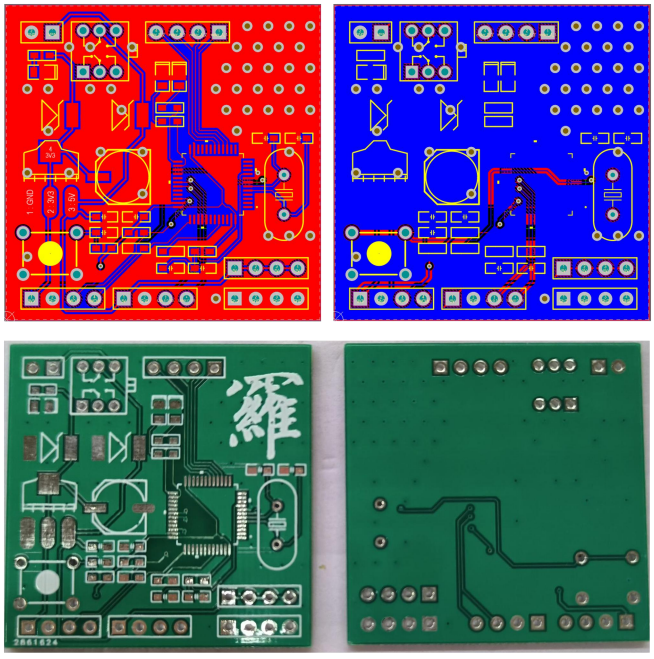


Figure 3.2.2. The diagram of the PCB and its real product

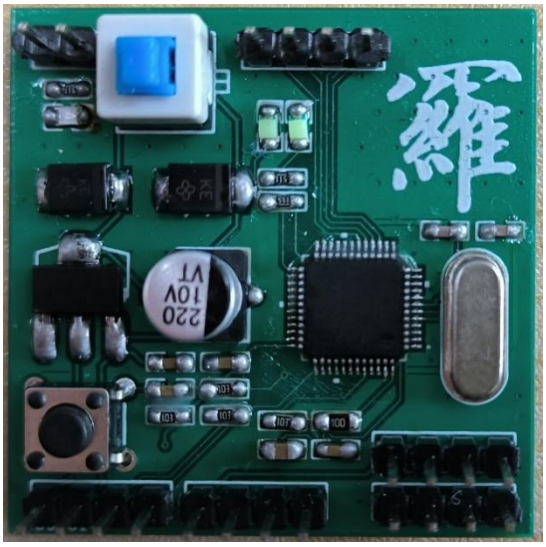


Figure 3.2.3. Final product

In the PCB board design, two Transient Voltage Suppression (TVS) diodes were used to protect sensitive circuits from transient or sudden voltage spikes. The TVS diodes acted as voltage clamps, providing a low-impedance path for excessive voltage transients. When a transient voltage exceeded a certain threshold, the TVS diodes were quickly conducted current, diverting the excess voltage away from the protected circuit. This helped prevent damage to components such as integrated circuits, transistors, or other sensitive electronic devices. Reliable protection against voltage surges was provided by the TVS diodes, ensuring the longevity and reliability of electronic systems. In this PCB board, a 3.3V and a 5V TVS diode were used.

Furthermore, a 1A voltage regulator chip AMS1117-3.3 [3] was integrated into the circuit to produce a 3.3V voltage from a 5V voltage, which was needed by the STMF103C6T6 microcontroller.

3.2.2 Configuration Detail

The pin configuration of the system is shown in **Figure 3.2.4**. PD0 and PD1 are configured as the input pin of 12M external crystal resonator which provide the accurate clock signal to the system. The detail of the clock configuration is shown in **Figure 3.2.5**. PA13 and PA14 are configured as the pin to download the code into the MCU. PA9 and PA10 were configured as the USART pins to communicate with OpenMV. PB12 and PB13 were configured as the GPIO_Output, which were connected with LED in the PCB. These two LEDs were used to indicate the situation of MCU when write codes. PA5, PA6, PA7 and PB0 are also configured as the GPIO_Output , which were used to control the state of left motor and right motor. More detail of this control principle can be found in **Section 3.2.3**. Internally, TIM1 was configured to control the duty cycle of the PWM signal. An update interruption is configured, and in the interruption handle function, the logic output of PA5, PA6, PA7 and PB0 could be changed based on the duty cycle wanted. Thus, the speed of the car can be controlled.

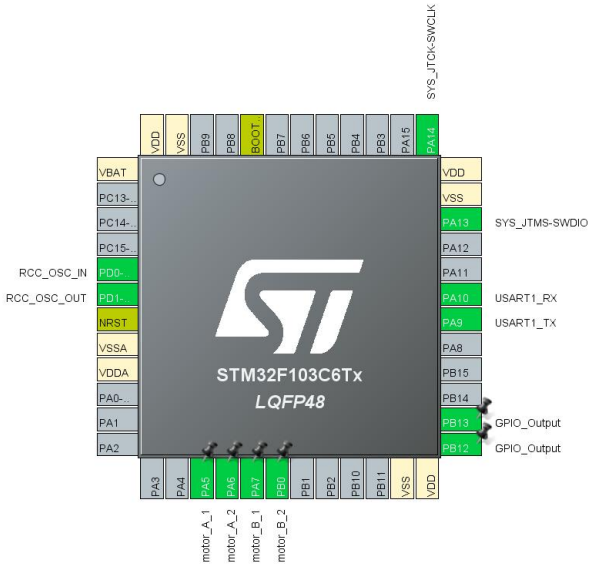


Figure 3.2.4. The pin configuration of MCU

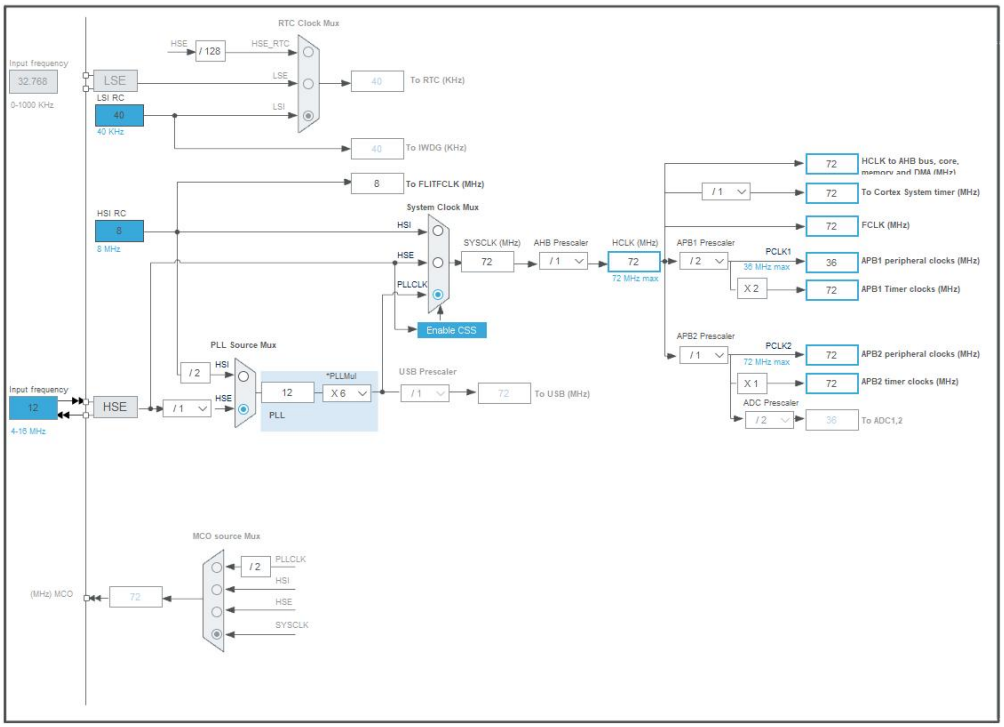


Figure 3.2.5. Clock configuration

3.2.3 Working Principle

As shown in **Figure 3.2.6**, when receive the command from OpenMV by USART, the self-designed PCB will modify the duty cycle of PWM wave it transmits to motor driver module. In this way, the speed and direction of the left and right motor can be controlled.

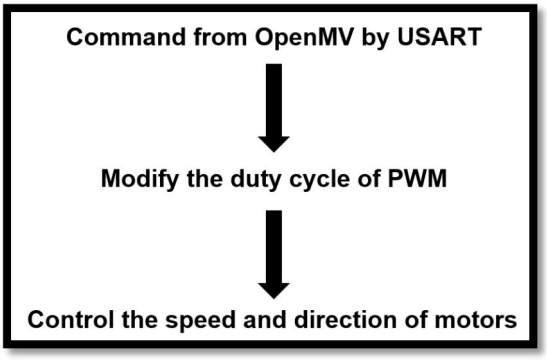


Figure 3.2.6. Working flow of the motion part

The example of commands being defined are shown in **Table 3.2.1**. The speed of the motor is divided into 30 parts for each direction. "+" represents moving forward, while "-" represents moving backward. Take "rotateL15R15" as an example, it means both motors should move forward with 50% of its full speed. The verification of this function is shown in **Figure 3.2.7**. It shows the output PWM signal given the input command is "rotateL15R15".

Table 3.2.1. Example of command from OpenMV

Command	Left motor	Right motor
rotateL15R15	+15	+15
leftmL15R15	-15	+15
rightmL15R15	+15	-15
bothmL15R15	-15	-15

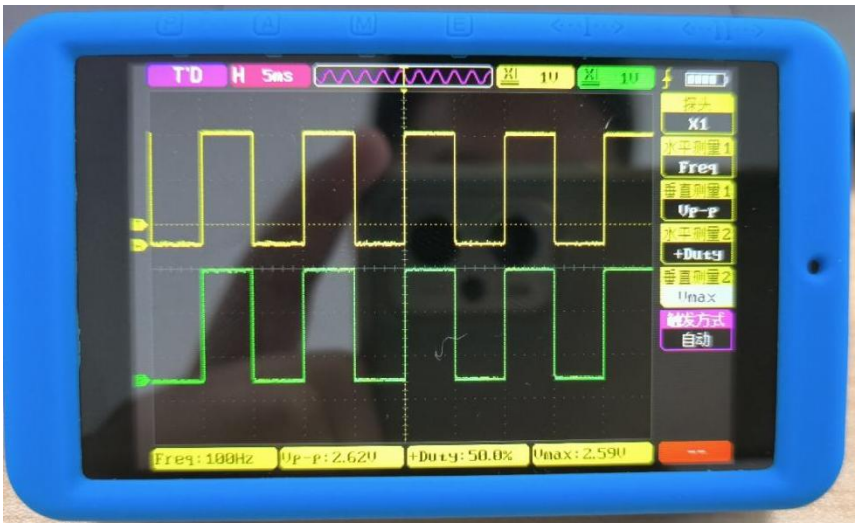


Figure 3.2.7. Verification of the functionality of command

The PWM signal is transmitted to motor driver module which react the signal as **Table 3.2.2**. Take left motor as an example. It is controlled by IN 1 and IN 2. When IN 1 and IN 2 are "0" and "1" respectively, it moves forward. While when IN 1 and IN 2 are "1" and "0" respectively, it moves backward. Besides, if the input of IN 1 and IN 2 are the same, it will stop.

Table 3.2.2. Control of motor

IN 1 / IN 3	IN 2 / IN 4	State
0	0	Stop
0	1	Forward
1	0	Backward
1	1	Stop

By control the PWM duty cycle, the percentage of on-time of motor can be controlled. Thus, the speed of motor can be controlled.

In summary, the overall structure of the motion system is shown as **Figure 3.2.8**

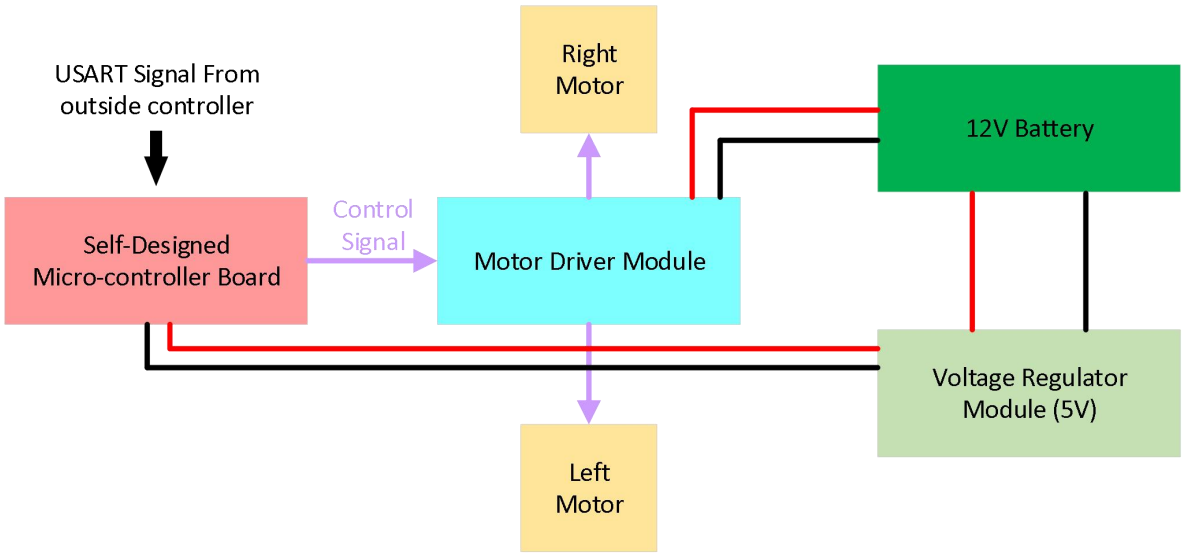


Figure 3.2.8. Overall structure of motion system

3.3 Mechanical Arm

Handled by: Li Shanshan
UoG: 2614182L

The Mechanical Arm submodule is an essential component of our smart car project, designed to achieve precise manipulation tasks. The designing process of mechanical arm is shown in **Figure 3.3.1.** The mechanical arm consists of acrylic version robotic arm components and is powered by a **servo motor SG90** for controlled movements. The submodule is seamlessly integrated into the system, with the STM32L432KC development board serving as the central control unit and Cube IDE software platform facilitating coding and integration. By utilizing the Mechanical Arm submodule, our smart car demonstrates the ability to accurately **release a table tennis ball** into a designated basket, showcasing the advanced capabilities of our design in fulfilling **Task 2 in Patio 2,** as shown in **Figure 3.3.2.**

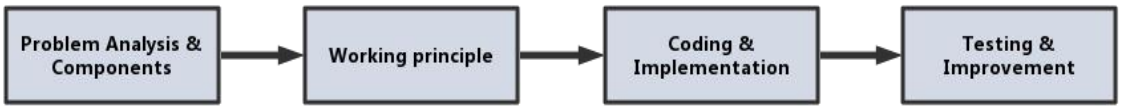


Figure 3.3.1. The designing process of mechanical arm

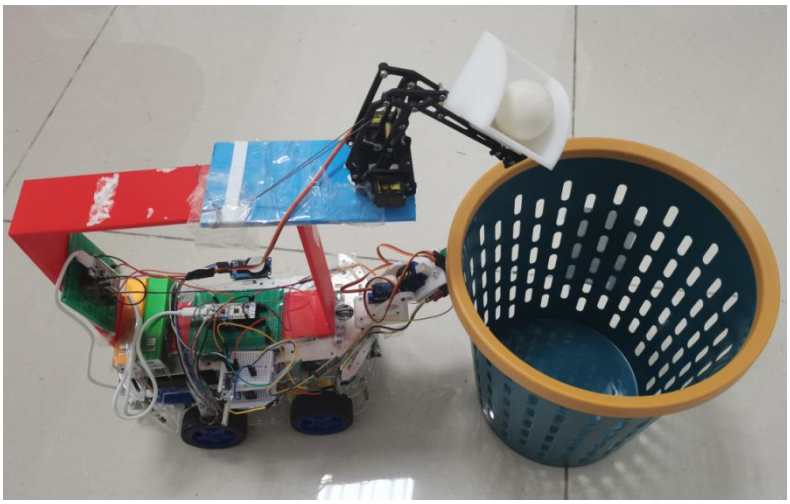


Figure 3.3.2. The task of mechanical arm

3.3.1 Requirement analysis:

In Patio 2, Task 2 of the TDPS project, the **objective** is to design a mechanism for the smart car to travel to the release point and accurately release a table tennis ball into the basket. To fulfill this requirement, a mechanical arm is chosen as the ideal solution. The mechanical arm should be capable of manipulating the tennis ball with precision and accuracy, considering the dimensions of the basket (25cm diameter) and the tennis ball (4cm diameter).

For the arm structure, we opted for **acrylic version robotic arm** components due to their lightweight, durability, and ease of assembly, as shown in **Figure 3.3.3**. The power for the mechanical arm is provided by **servo motor SG90,** known for its compact size and suitable torque capabilities.

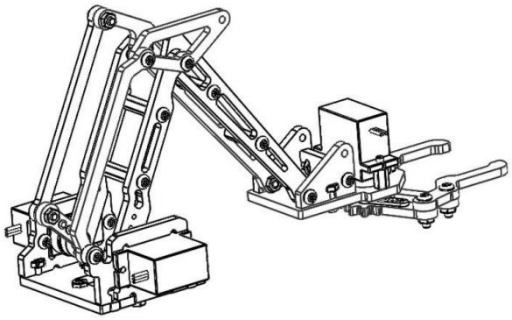


Figure 3.3.3. The sketch of acrylic version robotic arm

In terms of hardware, the development board used for controlling the mechanical arm is the **STM32L432KC[1]**, as shoen in **Figure 3.3.4,** known for its reliability and compatibility with various peripherals. The software development environment utilized for coding was **Cube IDE[2]**, offering a user-friendly interface and extensive libraries to facilitate programming and integration.

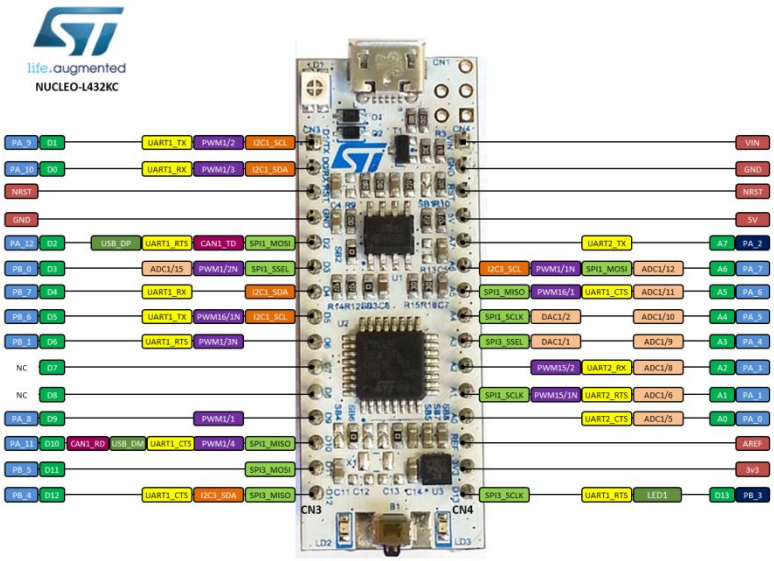


Figure 3.3.4. The structure of STM 32 L432KC [1]

Overall, the mechanical arm for Patio 2 Task 2 requires precise control, compatibility with the smart car system, and the ability to manipulate the tennis ball effectively within the given dimensions. The chosen components and software platforms were carefully selected to meet these requirements and ensure the successful execution of the task.

3.3.2 Working Principle

The **SG90 servo motor**, a commonly utilized component in the mechanical arm subsystem of our smart car project, operates based on the principles of electromagnetism and rotational motion, as shown in **Figure 3.3.5**. It belongs to the category of **DC servos**, which offer accurate and precise control over angular position. The working principle of the SG90 motor can be elucidated by drawing upon established theories and incorporating relevant formulas [3,4].

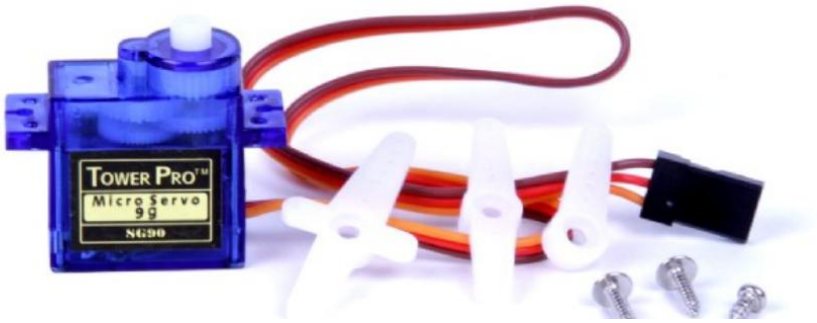


Figure 3.3.5. The picture of the motor SG90[4]

According to **Ampere's Law**, the magnetic field generated by the current flowing through the coils within the motor's stator interacts with the permanent magnet rotor, resulting in the production of a torque. This torque can be quantified using the formula:

$$\tau = F \times d$$

where τ represents the torque, F denotes the force exerted, and d signifies the distance. The force exerted by the motor is directly proportional to the current passing through the coils, as postulated by Ampere's Law. The distance parameter, on the other hand, is associated with the length of the motor's lever arm, influencing the moment arm in the torque equation [5].

To achieve controlled motion, the SG90 motor incorporates a feedback mechanism. This mechanism typically employs a potentiometer, optical encoder, or magnetic encoder to provide positional feedback to the motor control system. By comparing the actual position with the desired position, the control system adjusts the motor's operation to achieve accurate angular displacement. The resolution of the feedback mechanism determines the granularity of the motor's position control [3,5].

The TowerPro SG-90 servo motor offers a range of features that contribute to its popularity and effectiveness in motion control applications. Firstly, the motor operates at an optimal voltage of +5V, ensuring compatibility with common power sources in electronic systems, as shown in **Figure 3.3.6**. This voltage range facilitates seamless integration into a variety of projects. Secondly, the SG-90 motor boasts an impressive operating speed of 0.1s/60°, enabling precise and rapid movements essential for achieving accurate motion control. Lastly, the motor provides a rotation range from 0° to 180°, offering a wide angular span to accommodate various motion requirements [4,5].

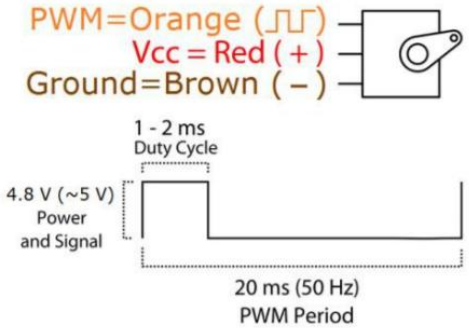


Figure 3.3.6. The PWM controlling of the motor

Controlling the TowerPro SG-90 motor requires an understanding of its specific control parameters. The motor

operates based on a time base pulse wave, typically with a duration of approximately 20ms. Within this time frame, the high-level pulse width determines the angle of motion. By varying the duration of the high voltage pulse, precise control over the motor's rotation angle can be achieved. Notably, the relationship between high voltage duration and rotation angle is shown in **Table 3.3.1.**

Table 3.3. 1. The relationship between High volatge duration & Rotation angle

High volatge duration	Rotation angle
0.5 ms	0°
1.0 ms	45°
1.5 ms	90°
2.0 ms	135°
2.5 ms	180°

This relationship allows for fine-grained control over the motor's motion and facilitates precise positioning within the desired range [4,5].

These **principles and formulas** find support in reputable references and research articles. One valuable reference is the textbook "Electric Machinery and Transformers" by Bhag S. Guru and Hüseyin R. Hiziroglu[3], which provides comprehensive coverage of electromechanical systems and serves as a valuable resource for understanding servo motors. Additionally, research papers published in renowned engineering journals such as IEEE Transactions on Industrial Electronics and IEEE Transactions on Robotics offer further insights into the working principles and control strategies of servo motors [5].

By harnessing the working principle of the SG90 motor, along with the theoretical foundations and formulas mentioned, the mechanical arm in our smart car project can achieve precise and controlled movements. The incorporation of position feedback enables accurate positioning of the arm, contributing to the successful execution of various tasks. This knowledge is derived from established theories and supported by authoritative references, highlighting the scientific rigor employed in the design and implementation of our smart car's mechanical arm subsystem.

3.3.3 Coding & Implementation

For the coding and implementation of the mechanical arm's PWM control, the STM32L432KC development board was used in conjunction with the **Cube IDE** development software. The STM32L432KC microcontroller, featuring a Cortex-M4 core, provided the necessary processing power and peripherals for the task.

Cube IDE, developed by STMicroelectronics, is an integrated development environment specifically designed for STM32 microcontrollers. It offers a comprehensive set of tools, including code generation, debugging, and peripheral configuration, to streamline the development process.

To **generate the PWM signal,** a **timer** within the STM32 microcontroller was configured. The chosen timer for PWM output was TIM1_CH1, as shown in **Figure 3.3.7.** The clock configuration was set to 4MHz, providing the necessary timing accuracy for the PWM generation, as shown in **Figure 3.3.8.**

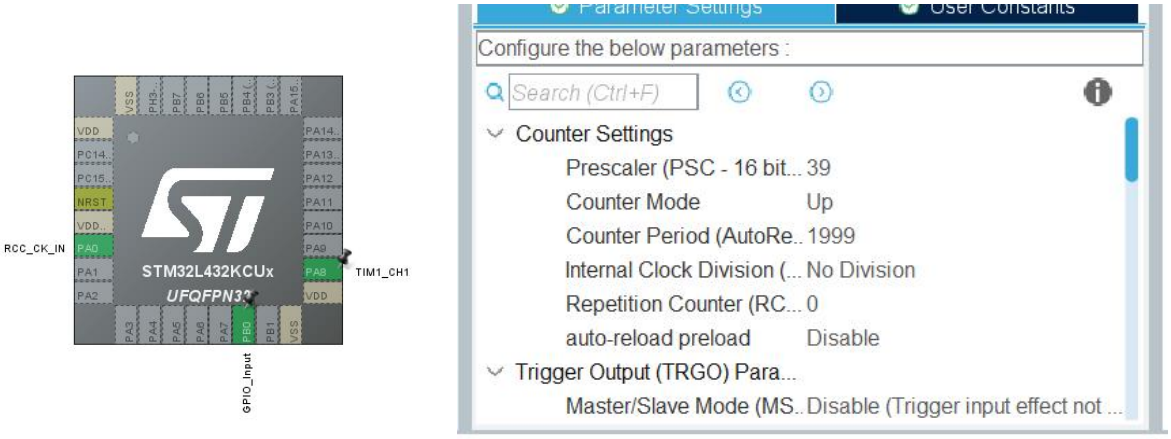


Figure 3.3.7. Pin configuration (left) and timer configuration (right)

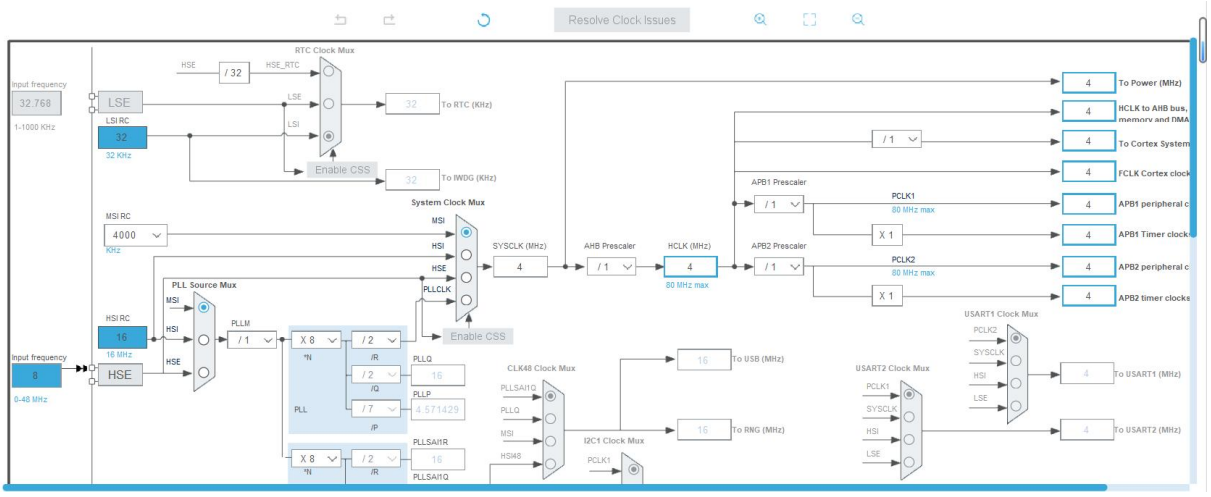


Figure 3.3.8. Clock configuration

To achieve the **desired frequency of 50Hz** for the PWM signal, the prescaler and counter period were carefully selected. By setting the prescaler to 39 and the counter period to 1999, the timer would count from 0 to 1999 and then reset, resulting in a repeating cycle, as shown in **Figure 3.3.7.** This configuration ensured that the PWM signal had a frequency of 50Hz, as calculated using the formula:

$$\text{PWM Frequency} = \frac{\text{Clock Frequency}}{(\text{Prescalar} + 1) \times (\text{Counter Period } + 1)}$$

Therefore, the **calculation steps** are listed as follows:

$$\text{PWM Frequency} = \frac{4\text{M}}{(39 + 1) \times (1999 + 1)} = 50\text{Hz}$$

$$\text{Preiod} = \frac{1}{\text{Frequency}} = \frac{1}{50\text{Hz}} = 20\text{ms}$$

$$\text{Duty cycle} = \frac{\text{Pulse}}{\text{Counter Period}} = \frac{100}{2000} = 0.05$$

$$\text{High voltage time} = 0.05 \times 20\text{ms} = 0.1\text{ms}$$

According to **Table 3.3.1,** when the high voltage duration is 0.1ms, the ratation angle of the motro is 45 [5,7].

The **duty cycle** of the PWM signal was controlled by setting a compare register value (CCR). The CCR value determined the threshold at which the PWM output transitioned from low to high. When the counter value (CNT) was less than the CCR value, the TIM1_CH1 output was set to low. Once the counter value exceeded the CCR value, the TIM1_CH1 output became high.

The **GPIO input** of the development board was utilized to control the generation of the PWM output. When the GPIO input was high, the PWM signal was generated. If the CNT value was less than the CCR value, the TIM1_CH1 output remained low. Once the CNT value exceeded the CCR value, the TIM1_CH1 output transitioned to high[4].

The precise control of the duty cycle and the generation of the PWM signal were crucial for driving the mechanical arm's movement. The configurations and calculations described above ensured accurate control of the PWM signal, allowing for the precise positioning and control of the mechanical arm.

3.3.4 Improvement of the Mechanical Arm Structure

In the **initial design** of the mechanical arm for the smart car project, a gripper mechanism was chosen to securely hold and release the ball, as shown in **Figure 3.3.9 (left).** However, it was observed that the engagement of the gears within the gripper was not optimal, resulting in uneven movement of the gripper's two sides. This inconsistency posed challenges in achieving precise control over the ball. Furthermore, the gripper mechanism did not provide reliable ball holding and dropping capabilities, compromising the effectiveness of the overall system. Three terms of tests were conducted and each term has 50 times of tests. The result of fail time was shown in **Table 3.3.2.**

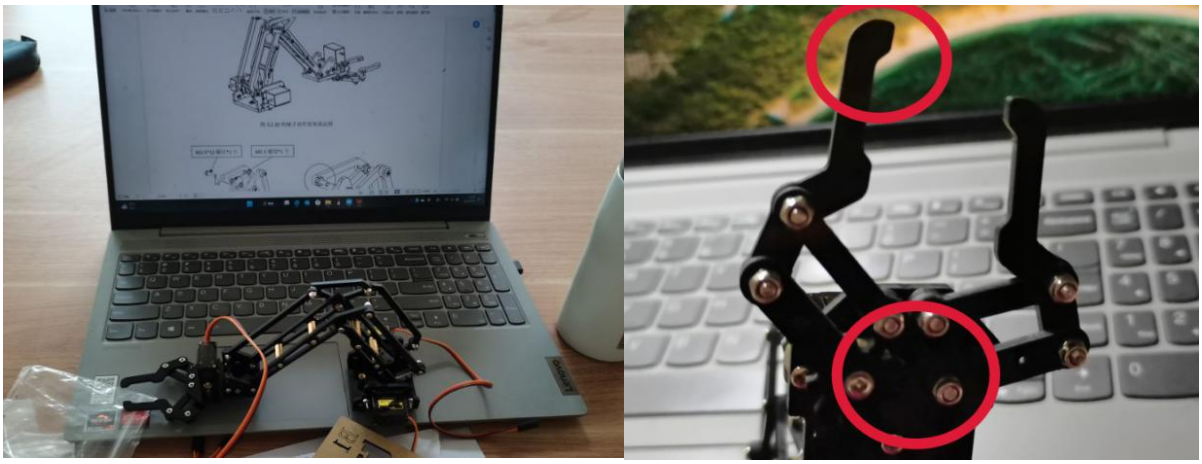


Figure 3.3.9. The structure of initial design (left) and he structure of gripper (right)

Table 3.3.2. Fail time for the initial design

Group of tests	Transportation	Drop
1	35	18
2	37	10
3	40	23

To overcome these challenges and ensure reliable ball handling and dropping, **a novel solution** was devised: the design of a specialized box for the mechanical arm using cardboard, as shown in **Figure 3.3.10 (left)**. The box structure effectively contains the ball within its boundaries, minimizing the risk of accidental displacement. The

mechanism for ball release was enhanced by implementing a swinging motion of the robotic arm. By oscillating the arm up and down, the ball is controlled to drop from a predetermined position within the box. This approach provides improved stability and control over the ball's release, facilitating the successful execution of various tasks.

The **new box structure** for the mechanical arm represents a significant improvement over the initial gripper mechanism and the subsequent cardboard holder. However, during transportation or sudden movements, the ball had the tendency to fall out from the cardboard holder, potentially leading to undesired consequences. Three terms of tests were conducted for the cardboard design and each term has 50 times of tests. The results of fail time was shown in **Table 3.3.3.**

Table 3.3.3. Fail time for the cardboard design

Group of tests	Transportation	Drop
1	16	10
2	14	8
3	18	11

To address this concern, a more refined and durable box structure was designed using computer-aided **design (CAD) software**, specifically **SolidWorks[8]**. The use of CAD software facilitated precise and detailed modeling of the box, ensuring optimal dimensions, shape, and structural integrity, as shown in **Figure 3.3.10**.(right). The design process involved considering factors such as ball containment, ease of integration with the mechanical arm, and compatibility with the smart car's overall architecture. Once the box design was finalized in the virtual environment, the next step involved fabricating a physical prototype using a 3D printer. The 3D printing technology allowed for the conversion of the digital design into a tangible object with the desired specifications. The utilization of a 3D printer enabled rapid and cost-effective production of the box structure, allowing for further testing and validation of its performance.

The same test was conducted and the results was shown in **Table 3.3.4.** The adoption of the improved box structure in the mechanical arm subsystem represents a significant advancement over the previous iterations. The transition from a cardboard prototype to a professionally designed and 3D printed box showcases the dedication to achieving a reliable and functional solution. This iterative design process highlights the importance of incorporating feedback, testing, and continuous refinement in the pursuit of engineering excellence.

Figure 3.3.10. The improved design (left) and the final design (right)

Table 3.3.4. Fail time for the 3D box design

Group of tests	Transportation	Drop
1	6	6
2	3	5
3	7	4

This improvement in the mechanical arm structure highlights the iterative nature of engineering design, where initial concepts are refined and optimized through continuous evaluation and experimentation. The adoption of the box design showcases the ingenuity and problem-solving skills employed in overcoming challenges and achieving desired outcomes within the smart car project [9].

3.3.5 Final configuration and test

Afterwards, tests were conducted to evaluate the mechanical arm's capacity under different circumstances for consistent ball transportation and precise ball placement into a predefined target area, as depicted in **Table 3.3.5.**

The mechanical arm's starting positions were configured to be at rest, at a 30-degree angle, and at a 45-degree angle. The rotational angle of the mechanical arm was set at 30°, 45°, or 60°. Each trial was repeated 50 times, and both the duration of ball transportation and the time taken for the ball to be dropped into the designated basket were meticulously recorded.

Table 3.3.5. The results of the tests

Trial	Initial Position	Drop time (Transportation)	Rotation angle	In time (Release)
1	resting position	3	30°	4
2	resting position	5	45°	3
3	resting position	7	60°	2
4	a 30-degree angle	2	30°	1
5	a 30-degree angle	1	45°	0
6	a 30-degree angle	1	60°	1
7	a 45-degree angle	7	30°	3
8	a 45-degree angle	8	45°	3
9	a 45-degree angle	8	60°	1

Upon comparison, it was observed that in Trial 5, with an initial position of the mechanical arm set at 30° and a rotation angle of 45°, the drop time during transportation was minimized to 1 unit, and the time taken for the ball to drop into the basket was minimized to 0 units. Consequently, Trial 5 was selected for implementation.

3.4 Distance Measurement System

Handled by: Li Jingyuan
GUID:2614262L

3.4.1 System Design

The robot car must possess the ability to perceive the distance between itself and surrounding objects to execute basic maneuvers such as acceleration, deceleration, and steering. Consequently, a distance measurement system was designed, and its diagram is shown in **Figure 3.4.1.**

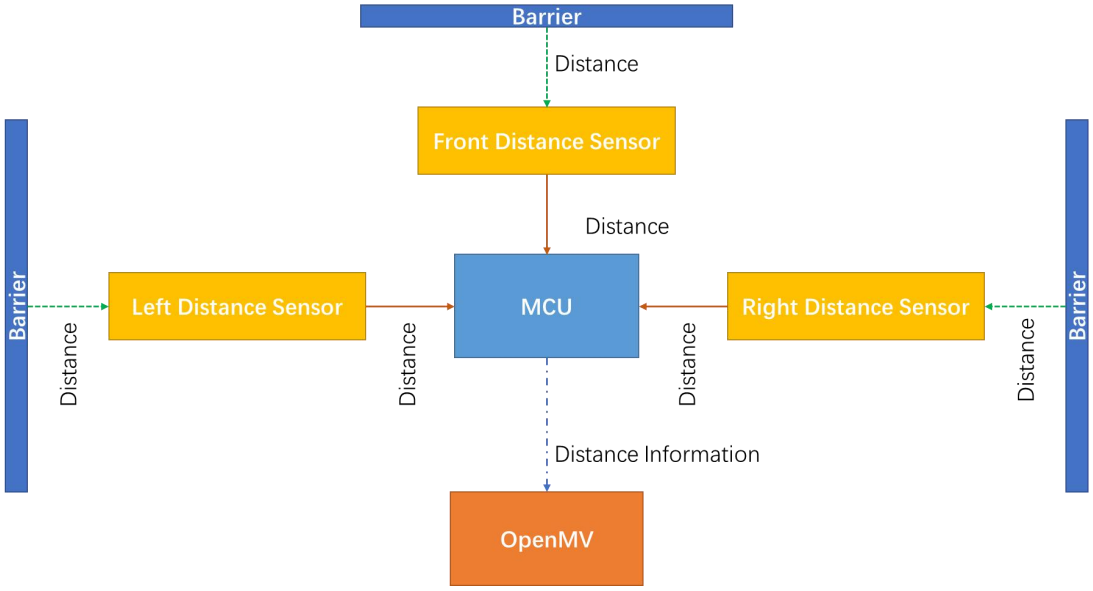


Figure 3.4.1. Diagram of the distance measurement system

Three distance measurement sensors are implemented in the front, left and right side to measure the distances between the robot car and barriers in those directions. Apart from this, an MCU is demanded to receive the measured distance from distance sensors, process them and sent the distance information to OpenMV H7 R2 camera module, the central control unit of the robot car.

3.4.2 Hardware Selection

3.4.2.1 MCU Selection

In terms of MCU, NUCLEO-L432KC was chosen. NUCLEO-L432KC is a development board from STMicroelectronics that is based on the ARM Cortex-M4 microcontroller. It features an STM32L432KCU6 MCU with a maximum clock speed of 80 MHz, 256 KB flash memory, and 64 KB SRAM. The board also includes an ST-LINK/V2-1 debugger which allows for easy programming and debugging. **Figure 3.4.2** is an actual object of NUCLEO-L432KC and its pin information **[1]**.

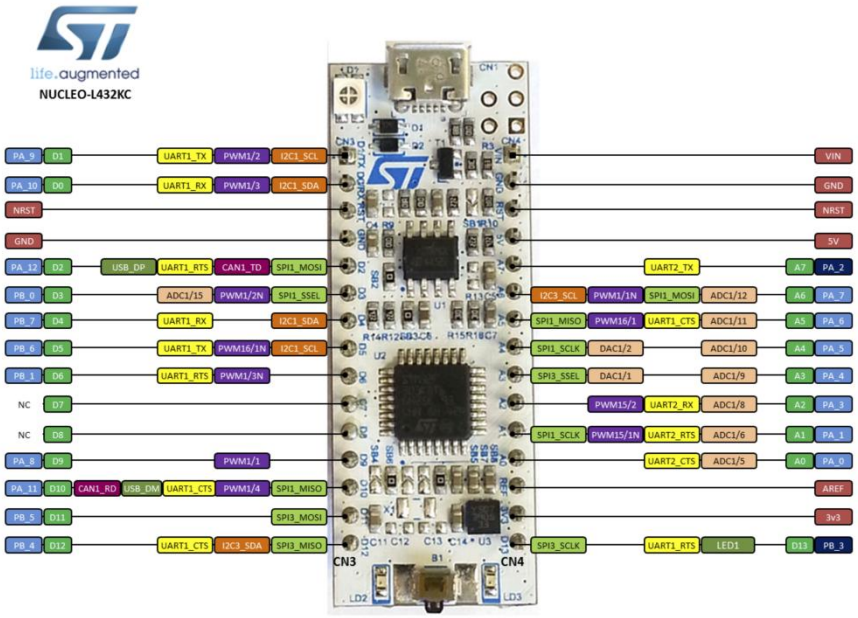


Figure 3.4.2. NUCLEO-L432KC and its pin information

The NUCLEO-L432KC board is fully compatible with the mbed online platform, which is a free development environment for embedded systems. The mbed platform provides an easy-to-use interface for prototyping and developing projects using ARM microcontrollers like the NUCLEO-L432KC.

With mbed, developers can leverage the power of C++ and a plethora of optimized libraries and APIs designed specifically for ARM architecture. Additionally, this platform offers access to an extensive array of resources such as sample code, tutorials, and documentation that facilitate seamless project initiation.

Apart from this, NUCLEO-L432KC board supports several communication interfaces that can be used for inter-board communication, including UART, SPI, I2C, and USB. These interfaces allow multiple boards to exchange data and signals with each other, enabling the creation of more complex systems.

3.4.2.2 Sensor Selection

For distance measurement, a selection of multiple sensors is available. Ultrasonic (HC-SR04 (**Figure 3.4.3.(left)**) [2] or laser (TOF050C (**Figure 3.4.3.(right)**) [3] devices can be utilized to accurately measure distances. In the first section, we provide a comparison between these two types of devices, which is illustrated in **Table 3.4.1.**

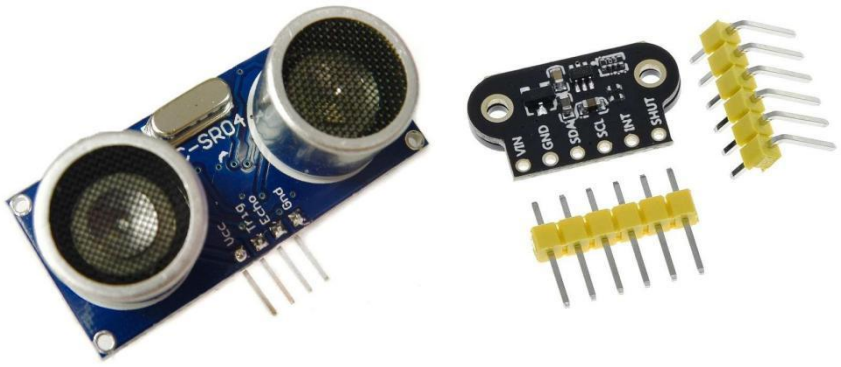


Figure 3.4.3. HC-SR04 Ultrasonic distance sensor(left) and TOF050C laser distance sensor (right)

Table 3.4.1. Comparison between 2 distance sensors

Device	Pins	Communication	Range	Unit cost/yuan
HC-SR04	4	GPIO	2-450cm	6.33
TOF050C	6	I2C	2-50cm	61.1

In **Table 3.4.1**, it is observed that the ultrasonic module HC-SR04 exhibits a wider measurement range, simpler wire connection, and communication method in addition to being more cost-effective. Apart from this. As a result, the ultrasonic module is chosen.

3.4.3 Device working principle

The HC-SR04 is an ultrasonic distance sensor that utilizes sonar principles to measure distances and detect objects in its surrounding environment. It is widely employed in robotics and automation projects for precise distance measurement and object detection [4]. The detailed properties of HC-SR04 are listed in **Table 3.4.2**.

Table 3.4.2. Hardware properties of HC-SR04

Pins	Vcc	5V
	Trig	Trig Pin
	Echo	Echo Pin
	GND	Grounded
Ultrasonic Wave Frequency	40 kHz	
Range	2-450 cm	
Max Measuring Frequency	40 Hz	

The operational principle of the HC-SR04 can be elucidated in a systematic manner. To initiate measurement, a trigger signal is transmitted to the sensor's Trigger pin, which is typically a brief high-level pulse lasting at least 10 microseconds. Upon receipt of the trigger signal, the HC-SR04 generates eight ultrasonic pulses by activating its ultrasonic transmitter that emits high-frequency sound waves at 40 kHz.

The ultrasonic pulse generated by the sensor's transmitter propagates through the air as a sound wave, expanding in a conical shape. If there is an object or obstacle obstructing its path, it will be struck and reflected as an echo.

The HC-SR04 is equipped with a receiver near the transmitter, which detects the reflected sound wave or echo upon reception. The time taken for the pulse to travel from the sensor to the object and back is precisely measured by this device. The workflow of HC-SR04 is shown in **Figure 3.4.4**.

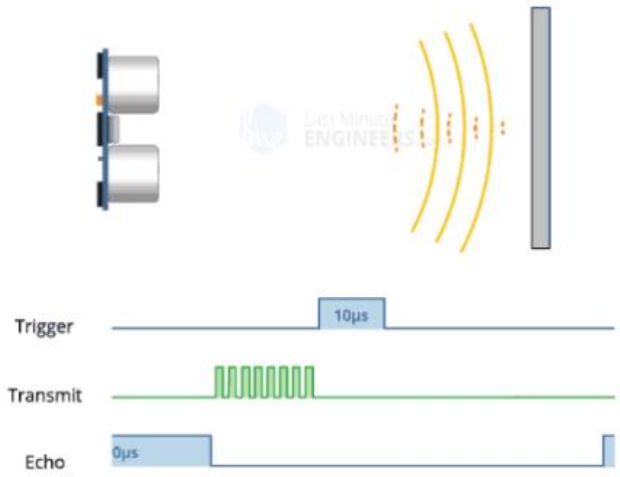


Figure 3.4.4. Working principle of HC-SR04

To determine the distance to an object, precise time measurement is crucial. The speed of sound in air (approximately 343 meters per second) is a known constant used in the calculation. By multiplying the measured time by the speed of sound and dividing by two, one can accurately calculate the distance between sensor and object. This division by two accounts for round-trip pulse travel time, and the calculation can be expressed as **Equation 3.4.1.**

$$Distance = 343 \times \frac{time}{2} \quad (\mathbf{3.4.1})$$

The HC-SR04 provides the calculated distance as a digital signal, typically through its Echo pin. A microcontroller or a device connected to the HC-SR04 reads the duration of the high-level signal on the Echo pin. This duration corresponds to the time taken for the pulse to travel, and it can be converted to distance using the formula mentioned above.

3.4.4 Communication

In this system, UART is utilized to transmit distance information from NUCLEO-L432KC to OpenMV. The utilization of UART for transmitting distance information offers several advantages. First, it is a simple and widely supported communication protocol, making it easier to implement and integrate into the system. Additionally, by utilizing UART, the NUCLEO-L432KC can efficiently transfer the distance information to OpenMV, the MCU of the overall system - ensuring real-time and reliable data transmission. Apart from this, the asynchronous nature of UART enables flexible communication, obviating the need for a shared clock signal between devices. Furthermore, its compatibility with diverse platforms and devices ensures seamless integration into existing system architecture.

Moreover, utilizing UART within the mbed framework proves to be a highly effective approach. This is due to the comprehensive and user-friendly development environment provided by mbed, which simplifies the implementation of UART communication. The built-in UART APIs and functions offered by the mbed library make configuring and using UART for data transmission between devices an effortless task. Additionally, the well-documented and supported functionality of UART in mbed enables developers to seamlessly integrate it into their projects [5].

3.4.5 Circuit Construction

The circuit of the distance measurement system is constructed as **Figure 3.4.5.**

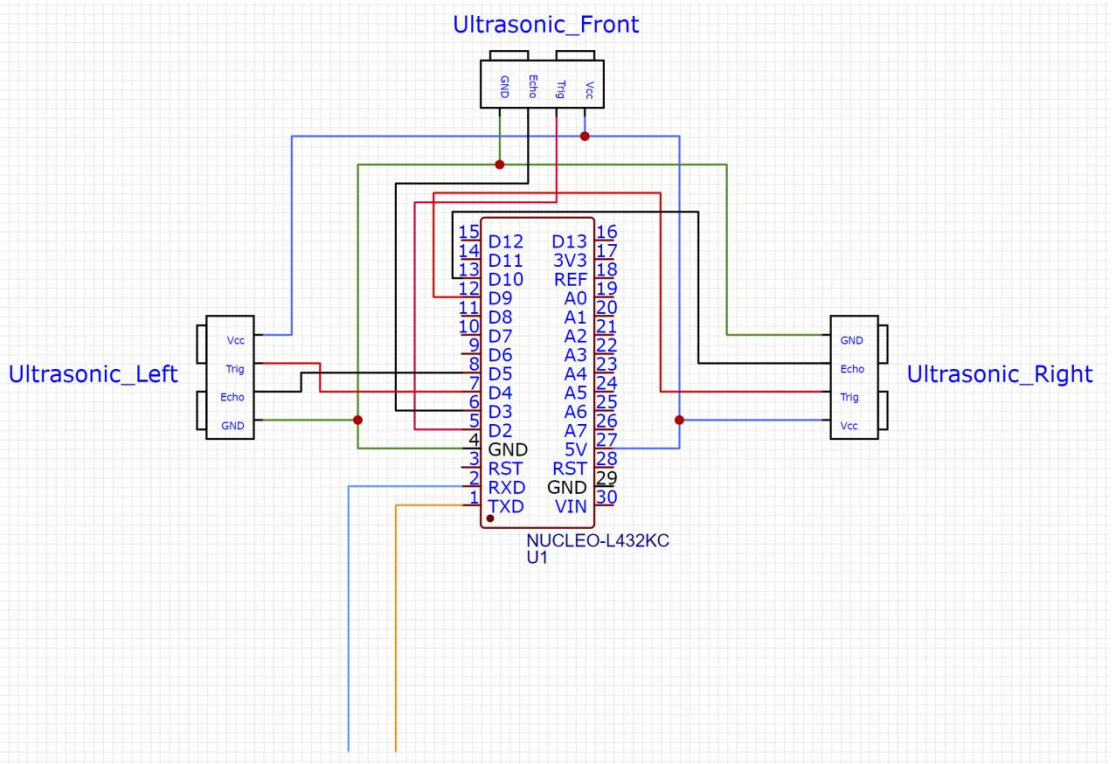


Figure 3.4. 5. Circuit diagram of the distance measurement system

As depicted in the diagram, this system employs three HC-SR04 ultrasonic sensors and a NUCLEO-L432KC microcontroller. Initially, these sensors were positioned at the front, left, and right sides of the robot car with their Vcc pins connected to the 5V pin of NUCLEO-L432KC. Subsequently, all four modules were grounded together for common use. In the next step, trigger pins of each sensor were linked respectively to D2, D4 and D9 ports on NUCLEO-L432KC while Echo pins were wired correspondingly to D3, D5 and D10.

3.4.6 Software Development

The distance measurement system's software program was developed using Mbed OS. In our software design, the NUCLEO-L432KC receives distance information from three sensors recorded in centimeters. This information is then integrated into a string (e.g., "1 2 3") and transmitted to OpenMV via UART at a maximum frequency of 40 Hz, which is the highest measurement frequency of the ultrasonic sensor.

3.4.7 Testing

The system was tested by revealing the distance information on both NUCLEO-L432KC and OpenMV as the robot car moved closer to barriers from a distance. The actual distance (measured by ruler), distance displayed from NUCLEO-L432KC and OpneMV were noted. The testing results are listed in **Table 3.4.3**.

Table 3.4. 3. Testing results of the distance measurement system

Actual Distance/cm	NUCLEO-L432KC Distance/cm	OpenMV Distance/cm
5.2, 10.1, 30.2	5, 10, 30	5, 10, 30
1.3, 9.5, 10.3	843, 11, 13	843, 11, 13
5.7, 0.5 20.7	6, 845, 21	6, 845, 21
10.2, 11.3, 20.8	10, 12, 21	None

13.5, 30.1, 20.2	15, 30, 21	15, 30, 21
15.3, 12.1, 14.7	15, 11, 15	15, 11, 15
6.8, 8.3, 11.2	7, 8, 12	None

In the test, we observed that the distance information would frequently exceed the upper measurement bound of HC-SR04 (450 cm) and surge to an unexpected high value, often surpassing 800 cm. Moreover, there were instances where OpenMV failed to read the transmitted data and displayed "None".

3.4.8 Improvement

The software code of NUCLEO-L432KC and OpenMV has been adjusted in response to the issues identified during testing in section **5.7**. Specifically, measurements that exceed the maximum threshold of 450 cm are now assigned a value of 2 cm due to our assurance that they are too close to barriers. Furthermore, in the event that OpenMV fails to read the information, a default value of "0" will be assigned by OpenMV. These values will be disregarded during distance measurement so as not to affect accuracy.

3.5 Machine Vision Hardware

Handled by: Li Jingyuan
GUID:2614262L

3.5.1 Introduction of OpenMV H7 R2 Camera Module

The OpenMV4 H7 R2 Camera (OpenMV) (**Figure 3.5.1**) is designed specifically for machine vision applications, and it offers a wide range of features and functions that are useful for developers and engineers working in this field. It is equipped with a powerful ARM Cortex-M7 processor that can run complex machine learning algorithms and computer vision tasks in real-time. This enables the camera to analyze images and video streams quickly and accurately, making it ideal for applications that require fast, real-time analysis[1].

Figure 3.5.1. OpenMV4 H7 R2 Camera Module

The camera features a wide range of sensors, including a 5-megapixel camera sensor, a microSD card slot, a temperature sensor, and an accelerometer. These sensors enable the camera to capture high-quality images and data, and to detect and respond to changes in its environment. Apart from this, the camera comes with a variety of built-in machine vision algorithms, including edge detection, face detection, object tracking, color detection, and more. These algorithms make it easy to get started with machine vision applications and can be customized to suit specific use cases. In addition, this device supports multiple communication options, including USB, I2C, SPI, UART, and CAN. This makes it easy to integrate the camera with other hardware and software systems, and to communicate with other devices in a network.

In the section of programming, OpenMV is based on the MicroPython programming language, which is a version of Python designed for use on microcontrollers and other embedded systems. This language provides a simple and intuitive syntax that is easy to utilize.

Overall, the OpenMV4 H7 R2 Camera is a powerful and versatile machine vision camera that offers a wide range of features and functions for developers and engineers working in this field. Whether you're working on a robotics project, an automation system, or an IoT device, this camera is a great choice for your machine vision needs.

3.5.2 Open MV Pan-Tilt Holder

The OpenMV pan-tilt holder is a crucial component for controlling the orientation of the OpenMV Camera series, providing a versatile solution for motorized camera positioning. Based on mechanical motion and servo motor control principles, this holder enables precise and programmable adjustments in both horizontal (pan) and vertical (tilt) axes.

The pan-tilt holder comprises several essential components, including a robust base that serves as the foundation for the entire assembly. This base is equipped with mounting holes or slots, facilitating seamless integration with other robotic or vision-based systems. Additionally, two servo motors are attached to the base - one for pan movement and another for tilt movement, which act as the driving force behind adjusting the camera's position.

To control the pan and tilt movements, the servo motors are connected to 3 steer control pins of OpenMV camera module which can generate Pulse Width Modulation (PWM) signals. PWM signals allow for precise control over the position of the servo motors by varying the duty cycle of the signal. By sending appropriate PWM signals to the servo motors, the pan-tilt holder can achieve accurate and incremental adjustments in both horizontal and vertical directions.

The OpenMV Cam module is securely fastened to the pan-tilt holder using a bracket or frame specifically designed for this purpose. The bracket ensures a stable and rigid connection between the camera module and the pan-tilt assembly, preventing any unwanted vibrations or movement during operation. This reliable mounting mechanism ensures that the camera maintains its position even during rapid movements or changes in orientation. The assembly of OpenMV and its pan-tilt holder is illustrated in **Figure 3.5.2**.

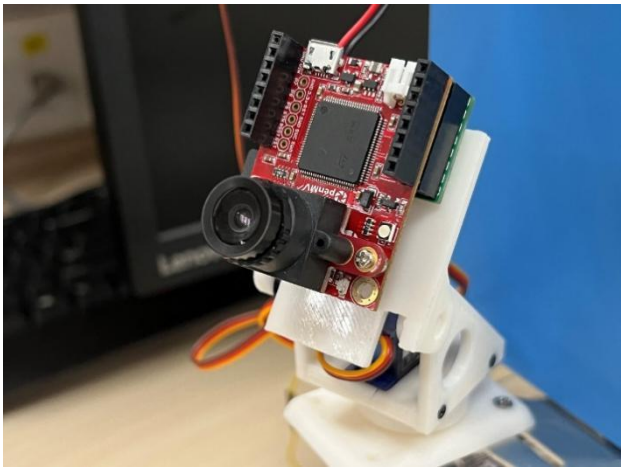


Figure 3.5.2. OpenMV and its pan-tilt holder

This pan-tilt holder (**Figure 3.5.3**) was modeled in SolidWorks, its overall size is 5.5 cm×4 cm×9.5 cm. Eventually, the holder was constructed by 3D-Printing with Polylactic acid (PLA). The filling rate of PLA was chosen to be 20%, due to the requirement of lightness.

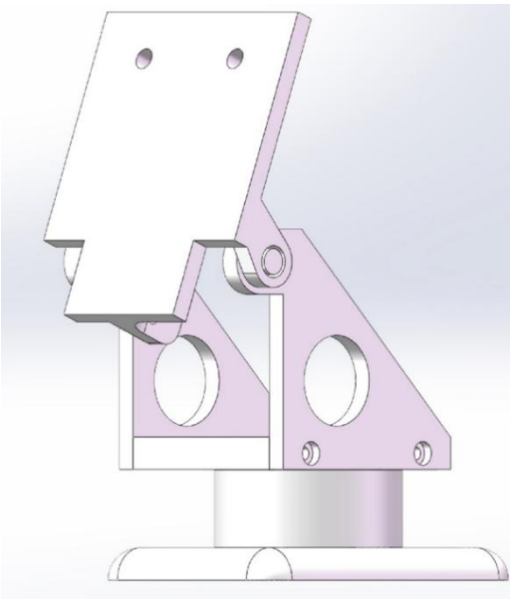


Figure 3.5.3. 3D model of OpenMV pan-tilt holder

3.6 Line Tracking

Handled by: Liu Daichen
GUID: 2614275L

To fulfill the tracking requirement in Patio 1, Task 1&3, the robot utilizes the OpenMV module which incorporates efficient and effective algorithms. Aimed at specific illustration of this task, OpenMV is used to capture the paths of Task 1 and Task 3 in Patio 1 as depicted in **Figure 3.6.1**. Targeting to address it, our design dissects this line tracking task into three segments: **Digital Image Processing**, **Line Recognition**, and **Motion Control**.

(a) Path in Patio 1, Task 1 (b) Path in Patio 1, Task 3

Figure 3.6.1. Paths Photos in Task 1 taken by OpenMV

In **Section 3.6.1**, effective techniques in **Digital Image Processing** are presented to address various environmental factors such as sunlight, weather conditions, and shading. The primary objective of this section is to extract the path from its surroundings. In **Section 3.6.2**, a kind of fast line recognition algorithm is designed to map out the route of this robot. In **Section 3.6.3**, Proportional–Integral–Derivative Controller is introduced to manipulate the motion of this robot according to the given route. Finally, **Section 3.6.4** provides the experimental results of this task, appended with the technical analysis and discussion.

3.6.1 Digital Image Processing (DIP)

Digital Image Processing is a method of manipulating digital images using mathematical algorithms performed on a digital computer. As an essential component of digital signal processing, it has numerous advantages over analog image processing, including the ability to apply a broader range of algorithms and avoid issues such as noise accumulation and distortion during processing. Widely recognized techniques in this field encompass image sampling and quantization, morphological image processing, feature extraction, recognition tasks, and image registration. In this project, we will focus on the **Morphological Image Processing** and **Edge Detection**.

- **Preliminary**

Before investigating the application of DIP, some basic knowledges are required to be introduced. An **image** is defined as a two-dimensional function $F(x,y)$, where x and y are two-dimensional spatial coordinates, and the amplitude of F at any pair of coordinates (x,y) is called the **pixel** of that image at that point. When x, y, and amplitude values of F are finite, we call it a **digital image**. In computer, an image can be stored in form of matrix. For example, a $N \times M$ digital image can be represented as:

$$F = \begin{bmatrix} F(0,0) & F(0,1) & F(0,2) & \cdots & F(0,M-1) \\ F(1,0) & F(1,1) & F(1,2) & \cdots & F(1,M-1) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ F(N-1,0) & F(N-1,1) & F(N-1,2) & \cdots & F(N-1,M-1) \end{bmatrix}$$

It should be noticed that the value of $F(x,y)$ actually represents the color at one pixel (x,y). If $F(x,y) = \{0,1\}$, the image should be a binary image, where 0 refers to black and 1 refers to white.

However, the complexity of the real world cannot be reduced to a simple binary distinction between black and white. **Color space** and **Channel** are introduced to represent a colorful image. The standard color spaces include RGB and HSV. OpenMV usually applies the RGB color space, which has three types of colors or attributes known as **R**ed, **G**reen and **B**lue, as shown in **Figure 3.6.2**. Each pixel in an RGB image is represented by three color values, which denote the intensity of red, green, and blue light required to produce that specific hue. The combination of the three primary colors can generate a vast array of hues and tones. **Figure 3.6.1** depicts the image captured by OpenMV in RGB Mode.

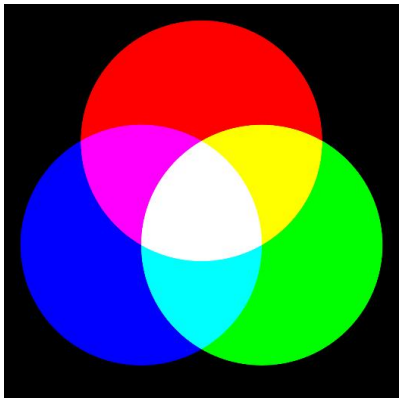
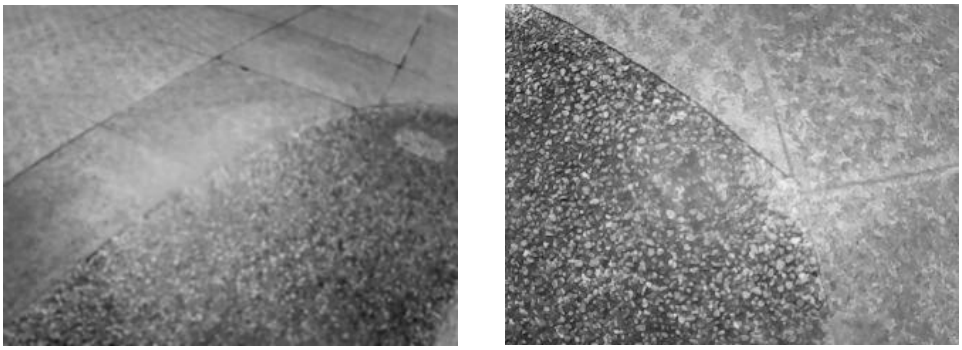


Figure 3.6.2. RGB Color Space

In this task, color is not a primary factor for distinguishing. Therefore, a grayscale image can be utilized, which is a simplified version of an image where each pixel solely represents the light intensity without any chromatic information. Grayscale images are represented using a single channel, and each pixel value indicates the brightness level of that pixel. In most grayscale image formats, pixel intensities are typically represented using 8 bits, allowing 256 levels of gray (ranging from 0 for black to 255 for white). The value 0 represents black, while 255 represents white, and the intermediate values represent shades of gray. **Figure 3.6.3** depicts two images in Patio 1 by OpenMV in Grayscale Mode. Obviously, the clarity of the grayscale is sufficient for discriminating the path from its surroundings.



(a) Right Turning (b) Left Turning

Figure 3.6.3. Grayscale Images in Patio 1

- **Morphological Image Processing**

Taking into consideration environmental factors such as sunlight and shading, Morphological Image Processing is employed to extract valuable information, enhance features, and execute various image processing tasks. In other words, it performs mathematical operations on the image with the aim of making the recognition module robust in most circumstances.

The two fundamental morphological operations are dilation and erosion. Referring to [1], **Dilation** expands the boundaries of regions or objects in an image, making them larger and more connected. It achieves this by considering each pixel in the image and replacing it with the maximum value within its neighborhood, resulting in a smoother and more cohesive structure. On the other hand, **erosion** shrinks the boundaries of objects, reducing their size and smoothing out irregularities by replacing each pixel with the minimum value within its neighborhood. **Figure 3.6.4** shows the effects of dilation and erosion operations.

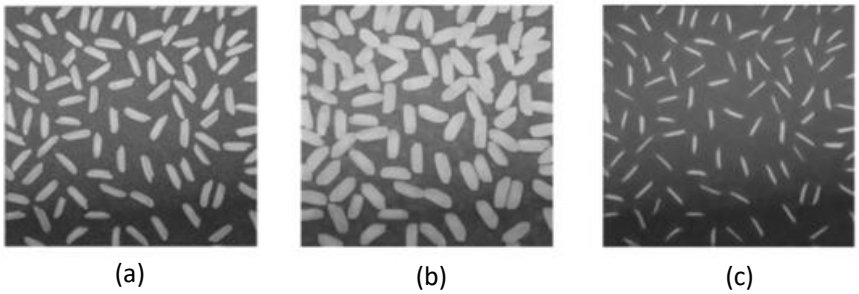


Figure 3.6.4. (a) Original Image (b) Dilated Image (c) Eroded Image

According to [2], these two basic operations can be combined and further extended to perform more complex operations such as opening and closing. **Opening** is the sequential application of erosion followed by dilation and is often used to remove noise, smooth edges, and separate overlapping objects. **Closing**, on the other hand, is the sequential application of dilation followed by erosion and is useful for filling gaps, closing holes, and joining broken or disconnected structures. **Figure 3.6.5** shows the effects of dilation and erosion operations on separating chromosomes.

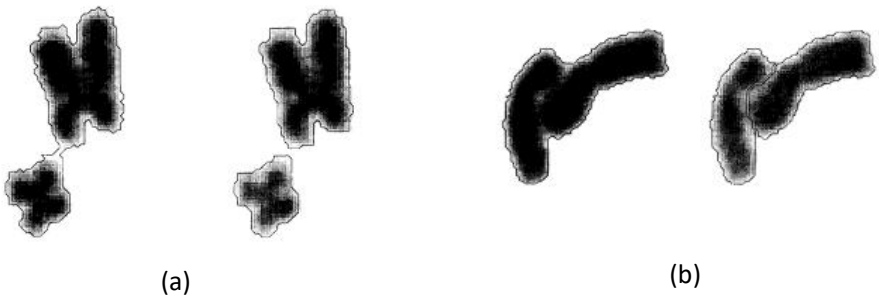


Figure 3.6.5. (a) Touching Chromosomes Separated by Opening Operation
(b) Distinguish Chromosomes by Closing Operation

Morphological Image Processing provides powerful tools for image analysis and manipulation, allowing researchers and practitioners to extract valuable information, enhance features, and solve complex image processing problems. Its versatility and effectiveness make it an essential tool in the field of Digital Image Processing. Its application in our project is clarified in **Section 3.6.4**.

- **Edge Detection using Canny Algorithm**

After performing the aforementioned operations, a clear image is obtained; however, it alone is insufficient for tracking purposes. To extract the path from the entire image, implementing Canny algorithm - initially proposed by Canny and John[3] - becomes essential. This algorithm accurately identifies and localizes edges present in an image while minimizing noise and spurious detections by following a series of steps to achieve its goal.

1. Gaussian Smoothing:

The initial step of the Canny algorithm involves applying Gaussian smoothing to the input image. This is performed in order to reduce noise and unwanted details that may interfere with the edge detection process. The application of Gaussian smoothing entails convolving the image with a Gaussian kernel, resulting in a blurred rendition of the original image. For example, a Gaussian kernel with the size of 3x3 and standard deviation 1 can be expressed as:

$$\begin{bmatrix} 0.0751136 & 0.1238414 & 0.0751136 \\ 0.1238414 & 0.2041799 & 0.1238414 \\ 0.0751136 & 0.1238414 & 0.0751136 \end{bmatrix}$$

Visually, the effects of this operator is to smoothing the image, as **Figure 3.6.6** shows.

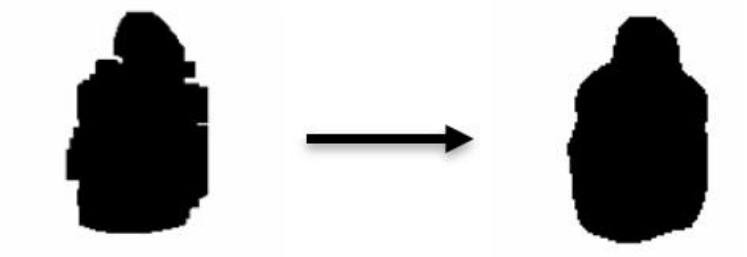


Figure 3.6.6. The effects of Gaussian Smoothing Operation

2. Gradient Calculation:

After smoothing the image, the next step involves calculating the gradient magnitude and orientation at each pixel by applying the Sobel operator to the smoothed image. Specifically, the Sobel operator computes gradients in both horizontal and vertical directions, with gradient magnitude calculated as the square root of squared gradients in both directions added together. This processed can be represented as the following expressions:

$$\mathbf{G_x} = \mathbf{S_x} * \mathbf{I} = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \boldsymbol{I}$$

$$\mathbf{G_y} = \mathbf{S_y} * \mathbf{I} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \boldsymbol{I}$$

$$g_{xy}(i,j) = \sqrt{g_x(i,j)^2 + g_y(i,j)^2}$$

in which, $\mathbf{I}$ denotes the grayscale image matrix, $\mathbf{G_x}, \mathbf{G_y}$ denotes the horizontal and vertical pixel gradient matrices, respectively. It should be noticed that gradient orientation is determined using arctangent of vertical-to-horizontal gradient ratio.

3. Non-Maximum Suppression:

In this step, the algorithm conducts non-maximum suppression to reduce the number of detected edges by examining the gradient magnitude and orientation at each pixel and suppressing gradients that are not local

maxima. Only local maxima indicate where edges are strongest, while non-maximum gradients are likely to be part of noise or weak edges. This step helps obtain a one-pixel wide representation of the edges.

4. Double Thresholding:

In this step, the algorithm conducts non-maximum suppression to reduce the number of detected edges by examining the gradient magnitude and orientation at each pixel and suppressing gradients that are not local maxima. Only local maxima indicate where edges are strongest, while non-maximum gradients are likely to be part of noise or weak edges. This step helps obtain a one-pixel wide representation of the edges.

5. Edge Tracking by Hysteresis:

To tackle the issue of noisy and weak edges, the Canny algorithm employs hysteresis-based edge tracking. It initiates from strong edges and follows along weak edges as long as they are connected to the former, thereby facilitating seamless connection of fragmented edges and obtaining continuous contours.

The result of the Canny algorithm is a binary image where the white pixels represent the detected edges. By adjusting the parameters such as the standard deviation of the Gaussian kernel, the threshold values, and the hysteresis parameters, the algorithm can be fine-tuned to suit specific applications and image characteristics. Detailed test and discussion on Edge Detection are provided in **Section 3.6.4**.

3.6.2 Line Recognition

The output of Canny Algorithm is a binary image. In order to achieve line tracking, the robot is required to know when to go straight and when to rotate. A kind of fast algorithm based on the middle lines is designed. The flowchart of this algorithm is shown as **Figure 3.6.7**.

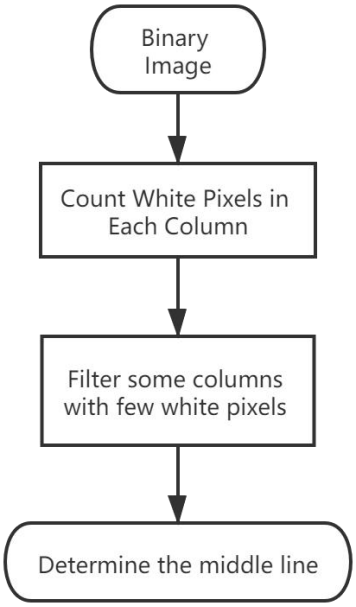


Figure 3.6.7. Self-Designed Algorithm Flowchart

As depicted in the aforementioned flowchart, the processed binary image is initially represented as a matrix of pixels. Subsequently, by traversing the entire matrix, the number of white pixels in each column is computed. To mitigate noise interference, columns with minimal white pixel count are filtered out. Finally, the remaining

columns are sorted based on their respective number of white pixels to obtain the central line of this binary image.

3.6.3 Motion Control

To reduce the distortion motion of our robot, PID (Proportional-Integral-Derivative) controller is adpted in our design. A PID controller is a widely utilized feedback control mechanism in robotics and automation systems, which continuously adjusts control inputs based on the error between the desired value and actual value to regulate and maintain a desired output. According to [4], a basic diagram of PID controller is shown as **Figure 3.6.8**.

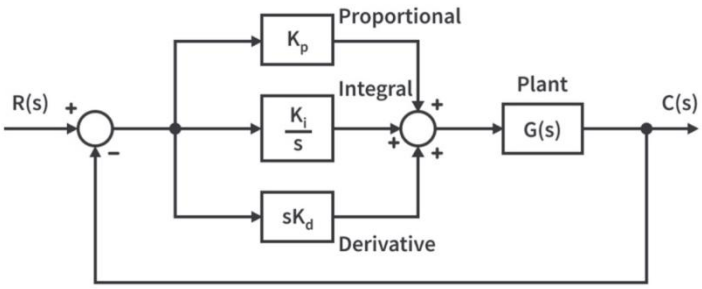


Figure 3.6.8. Block Diagram of a System with PID Control

The PID controller combines three components: proportional, integral, and derivative, each serving a specific purpose in the control loop.

Proportional (P) Component:

The proportional component of the PID controller generates an output proportional to the current error. It multiplies the error by a proportional gain factor (K_p), which determines the strength of the proportional response. The larger the error, the larger the control output. The proportional component helps in reducing steady-state errors, but it may cause overshoot and instability if used alone.

Integral (I) Component:

The integral component of the PID controller accounts for accumulated past errors. It sums up the error over time and multiplies it by the integral gain factor (K_i). The integral component helps in eliminating steady-state errors by continuously adjusting the control output. It is particularly useful when there are biases or disturbances in the system that cannot be addressed by the proportional component alone.

Derivative (D) Component:

The derivative component of the PID controller predicts the future error trend based on the rate of change of the error. It calculates the derivative of the error with respect to time and multiplies it by the derivative gain factor (K_d). The derivative component provides a damping effect and helps in reducing overshoot and stabilizing the system. It anticipates the system's behavior and reacts to sudden changes in error, thereby improving the system's response time.

The final control output of the PID controller is the sum of the contributions from the proportional, integral, and derivative components:

$$\text{Control output} = (K_p * \text{error}) + (K_i * \text{integral of error}) + (K_d * \text{derivative of error})$$

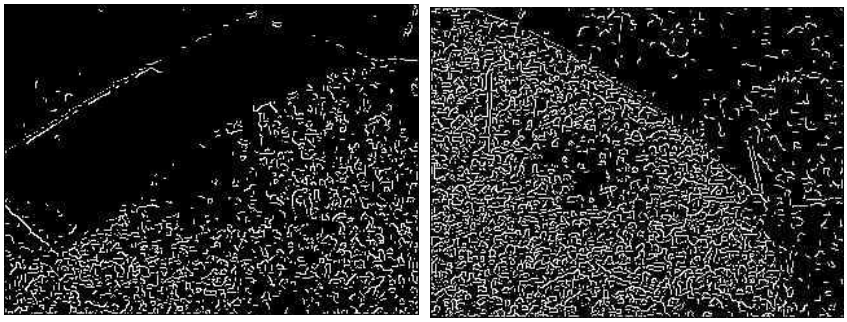
Tuning a PID controller involves adjusting the values of the proportional, integral, and derivative gain factors (K_p, K_i, K_d) to achieve the desired control performance. This is typically done through manual tuning or automated methods such as Ziegler-Nichols or optimization algorithms. Fortunately, OpenMV has introduced a library which integrates the whole algorithm, which can be directly used in our robot.

3.6.4 Experimental Results and Discussion

To evaluate the accuracy and efficiency of our algorithm, we conducted multiple rounds of testing using the OpenMV4 H7 R2 Camera. These tests primarily focused on edge detection and line recognition. As statistical analysis alone cannot fully capture the effects of PID, we adopted parameters that enabled the robot to move with maximum stability. All of the codes in this part is attached in **Appendix**.

Results and Discussion on Edge Detection

Using the Canny algorithm to process the grayscale images in **Figure 3.6.3**, the desired images are shown in **Figure 3.6.9**.



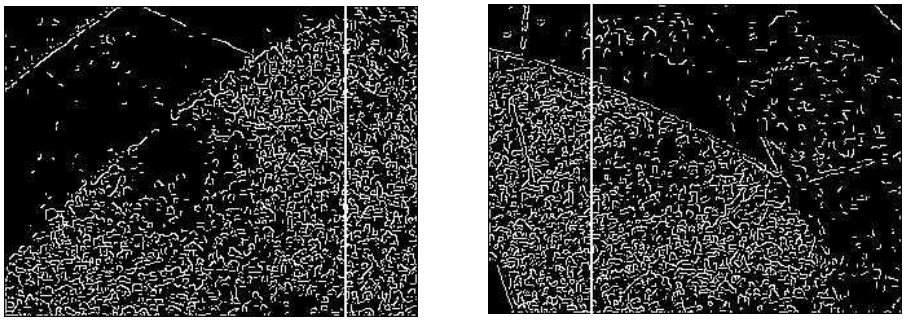
(a) (b)

Figure 3.6.9. The Images After Edge Detection of (a) Right Turning (b) Left Turning

Figure 3.6.9 demonstrates the effectiveness of the Canny algorithm in accurately determining the path that our robot must follow. Although some noise points exist, the major part of the path can be clearly distinguished from the whole image. In addition, multiple tests have been conducted under various weather conditions, including sunny, windy and even rainy. All experiments have confirmed the robustness of the canny algorithm as demonstrated in **Figure 3.6.9**.

Results and Discussion on Edge Detection

The algorithm in **Figure 3.6.7** was subjected to similar experiments conducted at the same location in Patio 1, with results presented in **Figure 3.6.10**.



(a) (b)

Figure 3.6.10. The Images After Line Recognition of (a) Right Turning (b) Left Turning

The success of our algorithm in determining the robot's movement direction can be demonstrated by these two images. It can be observed that as the middle line moves closer to the left (right), a larger angle of rotation is required for the robot. This property can be combined with the PID controller. When the middle line is very close to the left boundary or the right boundary for a continuous time, the PID controller manipulates the robot to rotate a larger angle. When the middle line is far away from the boundaries, the PID controller adjust the direction of the robot slowly.

Limitations and Future Work

The algorithms utilized in this project have been largely integrated into the OpenMV library, with the exception of the middle line detection algorithm. While its implementation is straightforward, it also imposes several limitations such as the time complexity and efficiency. At the same time, many other alternatives can be implemented such as Morph Transform to conduct edge detection, neutral network to recognize the path. Although these algorithms may require more time, they exhibit a slightly higher degree of accuracy compared to our own. Future research should prioritize the investigation of these algorithms in recognition tasks and conduct an efficiency and accuracy comparison among them.

3.7 Shape Matching

Handled by: Zeng Zihang

GUID: 2614050Z

3.7.1 Problem Analysis

According to Patio 2 Task 1, the robot is required to scan and recognize the shape on the stand of the square. The shapes to be detected are three arrows shown in **Figure 3.7.1**.

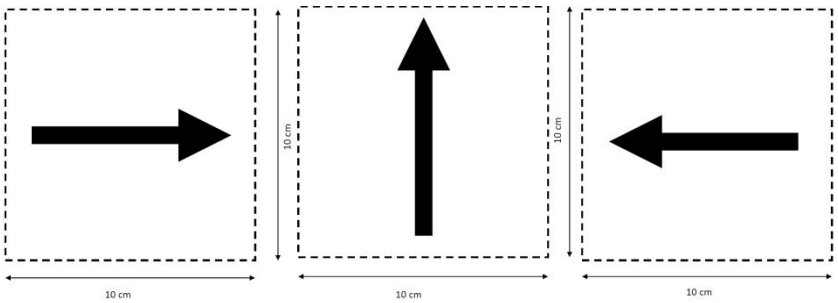


Figure 3.7.1. Three shapes for detection

There are two main challenges needed to overcome in this task. Considering that the car is moving in the recognition process, the recognition must be real-time and rapid, which means that the time complexity of the algorithm must be controlled at a low level. In addition, the task requires a high-accuracy algorithm, which can achieve a high recognition accuracy in the case of environmental interference.

To address these challenges, the following contributions were made in this work:

- It was analytically shown that the NCC, AGAST, and CNN models were not the most suitable and efficient models for Patio2 Task 1.

- A novel color-blob detection algorithm was proposed, which reduced the computational cost by accurately identifying the ROI of target shapes and calculating the pixel distribution.

- The cost-accuracy trade-off benefits of the proposed algorithm were demonstrated through experiments, and potential problems were reflected upon.

3.7.2 Related Work

3.7.2.1 Normalized Cross Correlation (NCC)

Normalized Cross Correlation (NCC) is a widely used technique in image recognition and computer vision applications. It is a measure of similarity between two images or image patches, which is particularly useful for tasks such as template matching and object detection. NCC measures the similarity between two signals by computing the normalized dot product between them [1]. In the context of image recognition, it allows us to compare the similarity between a template image and a target image or a specific region of interest within the target image. The Normalized Cross Correlation coefficient (NCC) is defined as:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Where x_i and y_i represent the intensity values of the corresponding pixels in the template image (x) and the target image (y), $\bar{x}$ and $\bar{y}$ denote the mean intensity values of the template image (x) and the target image (y).

3.7.2.2 Adaptive and Generic Accelerated Segment Test (AGAST)

The Adaptive and Generic Accelerated Segment Test (AGAST) is a corner detection algorithm used in computer vision and image processing. It is designed to efficiently and accurately identify corners or interest points, shown in **Figure 3.7.2** [2]. AGAST is an improved version of the traditional Segment Test (FAST) algorithm, which aims to provide faster and more robust corner detection capabilities.

Figure 3.7.2. Interest points in picture

AGAST can be employed in template matching, which involves finding instances of a given shape or object within an image. AGAST's ability to detect corners enables the creation of a template that represents the shape of interest. The template is then matched against the target image using corner-based matching algorithms, allowing for accurate and efficient shape matching.

3.7.2.3 Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNNs) are a class of deep learning models specifically designed for processing and analyzing visual data, such as images.

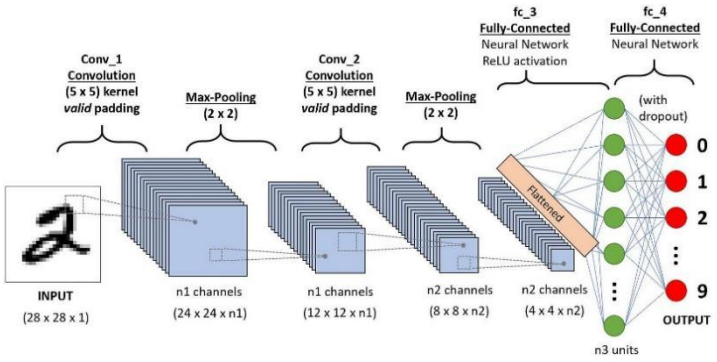


Figure 3.7.3. Architecture of CNN

CNNs have demonstrated exceptional performance in image classification tasks, surpassing traditional computer vision approaches. Their ability to learn hierarchical representations and capture spatial features through

convolutions and pooling enables them to extract intricate patterns and discriminative features from images. The deep and layered architecture of CNNs, shown in **Figure 3.7.3**, allows them to model complex relationships and generalize well to unseen data [3].

3.7.2.4 Discussion

Table 3.7.4 indicates the features of algorithms above. The Normalized Cross-Correlation (NCC), although exhibits strengths in handling variations in rotation and scale between images, as well as being less affected by changes in illumination, the effectiveness can be compromised by occlusions and noise present in images. Additionally, the high computational cost associated with NCC renders it unsuitable for real-time tasks [1]. Another algorithm, the Adaptive and Generic Accelerated Segment Test (AGAST), offers faster detection speed and robustness against noise. Nevertheless, AGAST encounters difficulties when dealing with simple shapes, as it struggles to find sufficient corners or keypoints for accurate detection [2]. On the other hand, the Convolutional Neural Network (CNN) has gained prominence due to its high accuracy in image recognition. However, the implementation of CNN requires substantial computational resources and large labeled datasets, making it impractical for this particular study [3]. Therefore, an alternative approach or technique needs to be explored to address the specific requirements of the problem at hand.

Table 3.7.1. Comparison of different algorithms

	NCC	AGAST	CNN
Accuracy	High	Low	High
Stability	Low	High	High
Computational cost	High	Low	High

3.7.3 Color Blob Detection

3.7.3.1 Image Processing in OpenMV

The OpenMV image processing library is a powerful and versatile tool that provides a range of functionalities for image analysis and computer vision applications. Developed specifically for the OpenMV Cam, a low-cost and open-source microcontroller board equipped with a camera module, this library offers a comprehensive set of image processing algorithms and functions [4].

Region of Interest (ROI)

The Region of Interest (ROI) plays a crucial role in many image processing and computer vision tasks, allowing users to focus on specific areas of an image rather than processing the entire image.

Once the ROI is defined, subsequent image processing operations and algorithms can be applied exclusively to the selected region. This enables efficient analysis and extraction of features, such as object detection, tracking, or specific image measurements, within the defined ROI. Users can perform tasks like thresholding, color filtering, edge detection, or even more advanced operations like template matching or machine learning-based classification, specifically within the region of interest [5].

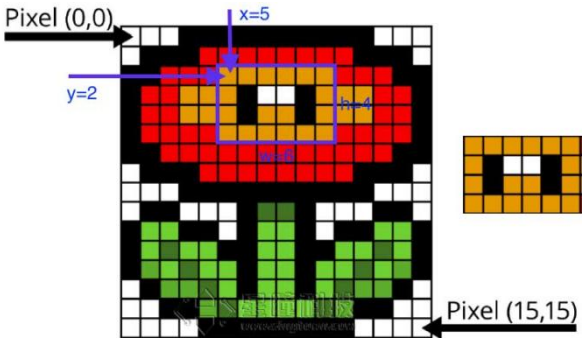


Figure 3.7.4. Region of interest(ROI)

Figure 3.7.4 shows the definition of ROI. The ROI is in the format of (x, y, w, h) tupple, where x and y are the X and Y coordinate in the top left corner of the ROI area, w and h is the width and height of ROI area.

LAB Color Space

The LAB color space is a perceptually uniform color model widely used in computer vision and image processing applications. It provides an effective representation of color information that is more consistent with human perception compared to other color models like RGB or HSV. In the OpenMV image processing library, the LAB color space is supported, allowing users to leverage its advantages for various color-based analysis tasks.

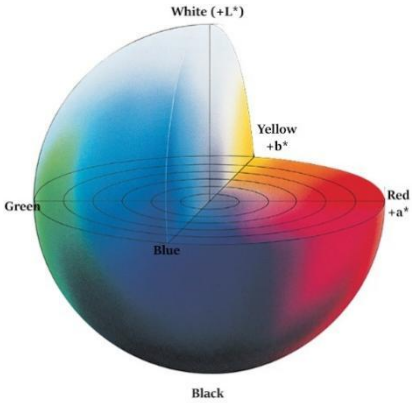


Figure 3.7.5. LAB color space

From **Figure 3.7.5**, the LAB color space separates color information into three channels: L (lightness), A (green-red axis), and B (blue-yellow axis). The L channel represents the brightness or intensity of the image, while the A and B channels capture the color components [6]. This separation enables decoupling of luminance and chrominance information, making it easier to distinguish color variations and perceive subtle differences in color.

The LAB color space is particularly useful in scenarios where accurate color discrimination is required, such as color-based object detection, tracking, or segmentation [6]. By converting an image into the LAB color space, users can perform operations like color thresholding or filtering more effectively, as they can target specific color ranges in the A and B channels.

To facilitate the process of setting color thresholds and filtering based on the LAB color space, OpenMV provides a LAB threshold editor. This graphical tool allows users to interactively define color ranges and thresholds within the LAB color space. Users can visually select a color of interest from an image and adjust the threshold sliders to specify the acceptable range for the A and B channels.

Relevant Functions

In the OpenMV image processing library, the image.find_blobs() function and the blob.pixels() function are valuable tools for blob detection and analysis in images.

The image.find_blobs() function allows users to detect and identify groups of connected pixel as blobs. By applying appropriate color thresholds, users can isolate specific objects or regions of interest based on their color characteristics [4]. The blob.pixels() function provides convenient access to the pixel-level information within a blob. It detects and identifies pixels based on color thresholds and returns the number of matched pixels [4].

3.7.3.2 Rationale

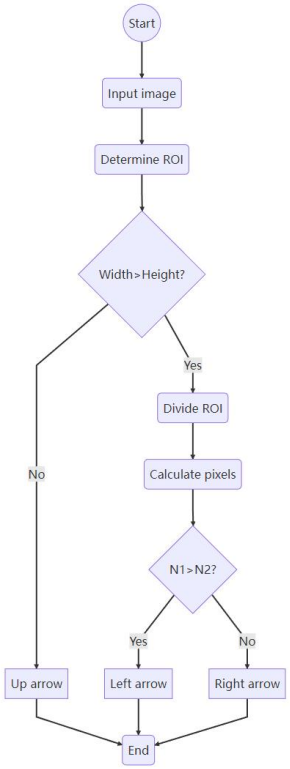


Figure 3.7.6. Algorithm flowchart

The rationale of color blob detection is shown in **Figure 3.7.6**. First, use the LAB threshold editor in OpenMV to find the threshold parameter corresponding to the arrow graph. After reading the image from OpenMV, use the image.find_blobs() function to find the ROI corresponding to the black arrow symbol and compare the width and height of the ROI. If the height is greater than the width, the shape is an up arrow, otherwise divide the ROI into left and right parts. Then use blob.pixels() function to calculate the number of pixels that meet the threshold in the left and right regions. If the number of pixels in the left half (N1) is greater than the number of pixels in the right half (N2), the shape is left arrow, otherwise is right arrow.

3.7.4 Test and Analysis

3.7.4.1 Experimental Setup

Table 3.7.2. Parameter settings

Parameter	Value

Pixel format	RGB565 (16bit)
Frame size	QQVGA (160x120)
Skip frames	2000

3.7.4.2 Accuracy

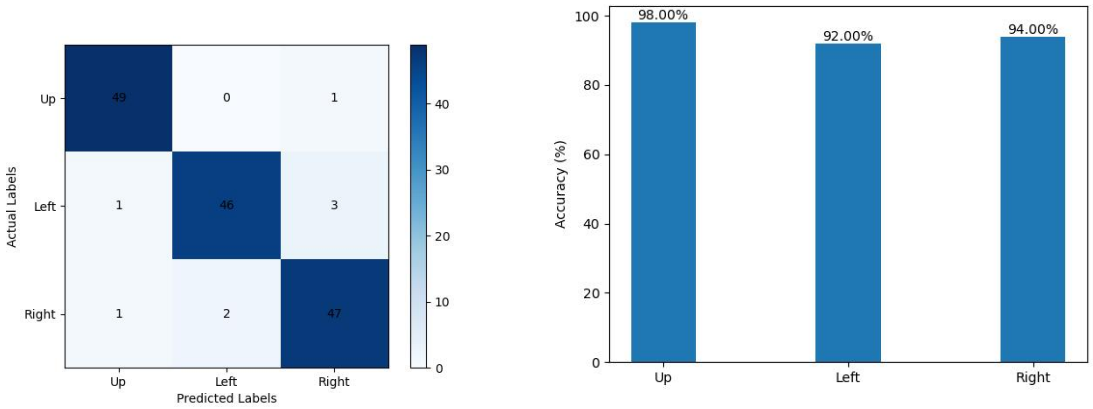


Figure 3.7.7. Confusion matrix (left) and the accuracy bar chart (right)

We presented the results using a confusion matrix in **Figure 3.7.7 (left)** and an accuracy bar chart in **Figure 3.7.7 (right)** to provide an intuitive visualization. The confusion matrix depicts the correspondence between the actual directions and the model's predicted directions [7]. Each cell in the matrix represents the number of correctly predicted samples for the corresponding direction. From the confusion matrix, it can be observed that Color Blob Detection model demonstrated good performance in the recognition task of arrow directions. To provide a clearer representation of the accuracy for each direction, we created an accuracy bar chart, as shown in Figure 2. From the bar chart, it can be observed that the model performed well in recognizing the "Up" directions, achieving accuracies of 98%. However, the accuracy for the "Left" and "Right" directions were comparatively lower, with 92% and 94% respectively.

3.7.4.2 Time Complexity Analysis

The **Table 3.7.4** shows the time complexity of different algorithms [8], where N represents the side length of the image, M represents the side length of the template, L, K and C represent the number of CNN layers, the size of the convolution kernel and the number of channels, respectively, and S is the number of sampling point.

Table 3.7.3. Time complexity of different algorithms

Algorithm	**Time complexity**
Color blob detection	$O(N^2)$
NCC	$O((N-M+1)^2 * M^2)$
CNN	$O(L * K^2 * C * N^2)$

AGAST	$O(S * N^2)$

From the table, it is obvious that compared with other commonly used algorithms, the Color blob detection method has lower time complexity, meaning lower computational cost.

3.7.4.3 Limitations and Future Improvements

Although our algorithm achieves a good balance between recognition accuracy and computation, there are still some problems with the algorithm.

First, algorithm performance is easily affected by light intensity. Specifically, the LAB threshold is different under different lighting conditions. Secondly, the accuracy rate is also affected by the recognition distance. If the recognition distance is small, the camera cannot capture the complete arrow shape; if the recognition distance is too far, the image will be disturbed by environmental color blocks.

Therefore, in **future work,** the following measures can be taken to ensure the performance of the algorithm:

1. Update the threshold parameter information before each test
2. Set an appropriate detection distance, and only detect when OpenMV is at this distance from the image to be recognized.

3.8 Wireless Communication

Handled by: Zhao Jiaxin

UoG: 2614041Z

According to the requirements of Patio2 Task3, the car needs to stop near the flower bed, send the team name and current time information, and then continue moving forward after receiving confirmation. This requires the construction of a communication system between the car and the computer, which can send and receive information. This section will mainly describe the specific deployment of this communication system, including the design of the transmitter and receiver, as well as time acquisition.

3.8.1 Communication System Design

Required materials:

·Two HC-12 modules

·One HC-USB-T adapter

·One DS1302 module

·One microcontroller

The wireless communication system was designed, as depicted in **Figure 3.8.1**. Two HC-12 modules were utilized to cater to the requirement of having both a transmitter and a receiver. To bridge the interface gap between the HC-12 modules (which have TTL interfaces) and the computer (which has USB interfaces), the HC-USB-T adapter was connected, as shown in **Figure 3.8.2.** The microcontroller served as the control center, managing the overall command input and output.

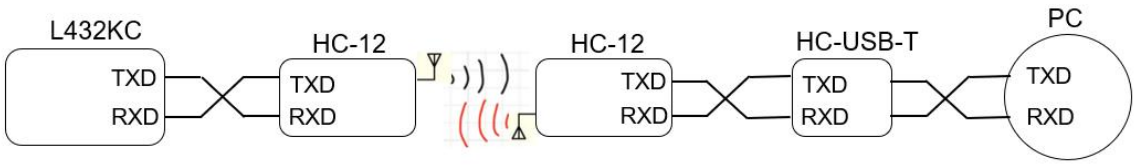


Figure 3.8.1. Wireless communication system

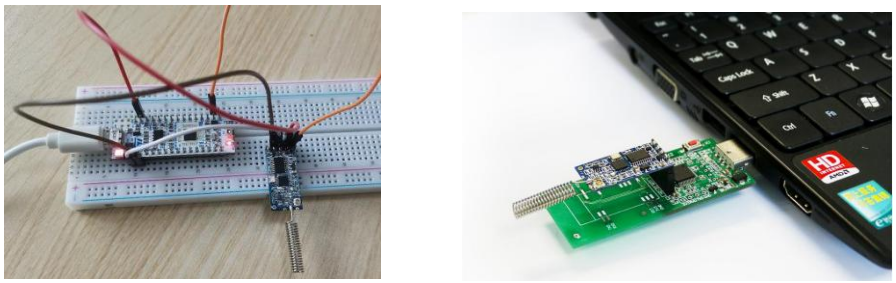


Figure 3.8. 2. Connection of MCU and HC-12(left) and connection of MCU and the computer(right)

3.8.2 Design of Transmitter and Receiver (HC-12)

The physical appearance and interface of the HC-12 module are shown in **Figure 3.8.3**, and the parameter table is shown in **Table 3.8.1**. The HC-12 module replaces traditional wired connections by transmitting

corresponding signals from the TX pin, wirelessly reaching the other HC-12 module [1]. The communication mode for HC-12 is **half-duplex**, allowing the signal to travel in both directions, but only permitting signal to travel in one direction on one channel at a time [1, 2]. This is sufficient for the task, because the task requires both computer and MCU to transmit signal but in different time. The interface is simple, requiring only the TX pin to be connected to the corresponding RX pin and the RX pin to be connected to the corresponding TX pin. However, the focus of the communication system design lies in the baud rate setting and wireless channel selection.

The HC-12 module provides reliable wireless channels from 001 to 100, corresponding to frequencies from 433.3 MHz to 473 MHz [2]. The baud rate of the serial interface affects the communication distance and the amount of information transmitted. Higher baud rates allow for larger information transfer but sacrifice communication distance, while lower baud rates allow for longer communication distances but can only guarantee transmission of smaller amounts of reliable information [2, 3]. Considering the requirements of the project, we need to use 433 MHz radio waves for transmission and select channel 1, which is closest to that frequency. Commonly used baud rates are 4800 bps, 9600 bps, 14400 bps, and 19200 bps. Since the communication distance is relatively short and the amount of information transmitted is not large, we choose a lower baud rate of 9600 bps.

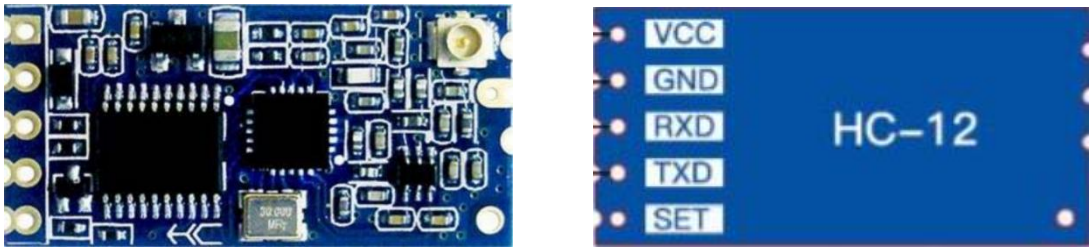


Figure 3.8.3. The physical appearance(left) and interface of HC-12(right)

Table 3.8.1. The parameter of HC-12

Parameter Name	Parameter	Parameter Name	Parameter Value
Model	HC-12	Module Size	27.4*13.2*4mm
Interface	UART 3.3V-5V TTL Level	Operating Frequency Band	433.4-473.0MHz
Operating Voltage	3.2-5.5V	Antenna Interface	Spring antenna
Communication Level	3.3V/5V level	Operating Humidity	10%-90%
Transmitting Power	UART 20dBm(MAX)	Operating Temperature	−25℃~＋75℃
Reference Distance	1000m		

3.8.3 Trigger Condition Analysis

According to the task, the car will transmit the corresponding information to the computer upon receiving the signal from the OpenMV camera. It is required to continue the car's movement after confirming receipt of the information. After analysis, there are two main methods to solve this issue.

1. The computer automatically send an acknowledgement to the smart car after receiving the information.
2. The computer send the confirmation message manually from the computer.

By careful analysis, method 1 is too complex to establish the corresponding mechanism on the computer to control the automatic transmission and may have issues with delayed or failed transmission. Method 2 may also introduce some delay since it is manipulated. Worse still, when sending information from the computer fails, the car is faced with a situation where it stops and the task fails. Therefore, we combine both methods.

After sending the information, the car starts a timer. If there is any information received within 10 seconds or when 10 seconds have elapsed, the car continues moving forward. This effectively avoids the failure of the car being unable to proceed due to delayed reception of information [2]. The flowchart for this approach is shown in **Figure 3.8.4**

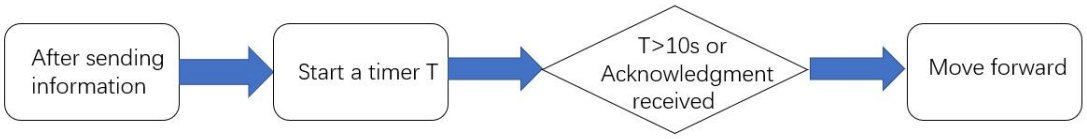


Figure 3.8.4. The flowchart of trigger condition

3.8.4 Time Acquisition Method Analysis

Transmitting the team name information is straightforward, requiring the corresponding content to be written in the programming platform. However, as time is constantly changing, it presents a challenge for the car to acquire accurate time. One direct method is to read the time from the microcontroller. It could be solved by giving the microcontroller an initial time and updating it every second, implementing a "counter" that increments and carries over different time units. However, if the power to the microcontroller is cut off, the counting will stop, and when the power is restored, the car will start counting from the initial time instead of the time at which it stopped. Therefore, we need a clock module that can continue counting even after power loss to provide accurate time.

The clock module should provide its own power internally and keeps counting time. When the module is connected to the MCU, the module can provide the current time to the MCU at any time. By using this module, it only requires a simple initialization of the clock module, and the code complexity is relatively low. Among many clock modules, DS1302 serial I/O communication mode saves ports, and the advantages of cheap price and low power consumption stand out among many clock modules, we choose it as a clock module.

3.8.5 Principle and Design of DS1302

The physical connection of DS1302 clock module is shown in **Figure 3.8.5(left).** The internal circuit is shown in **Figure 3.8.5(right),** where the module uses an external 32.768 kHz crystal oscillator (Y2) as the clock source, a backup battery (P2) to ensure continued operation after power loss, a diode (D1) for voltage rectification to prevent damage to the backup battery due to high voltage, and a pull-up resistor (R6) to enhance signal stability [4, 5, 6]. In terms of time acquisition, we mainly focus on the CLK, DAT, and RST pins of the DS1302 module

to read the clock.

Clock reading diagram is illustrated in **Figure 3.8.6**. When performing read/write operations on the DS1302 module, the clock needs to be pulled high while the RST pin is set to a high level. The address byte is read at the rising edge of the clock, and the data is output/input at the falling edge/rising edge of the clock [4]. This could achieve time initialization and obtain of time.

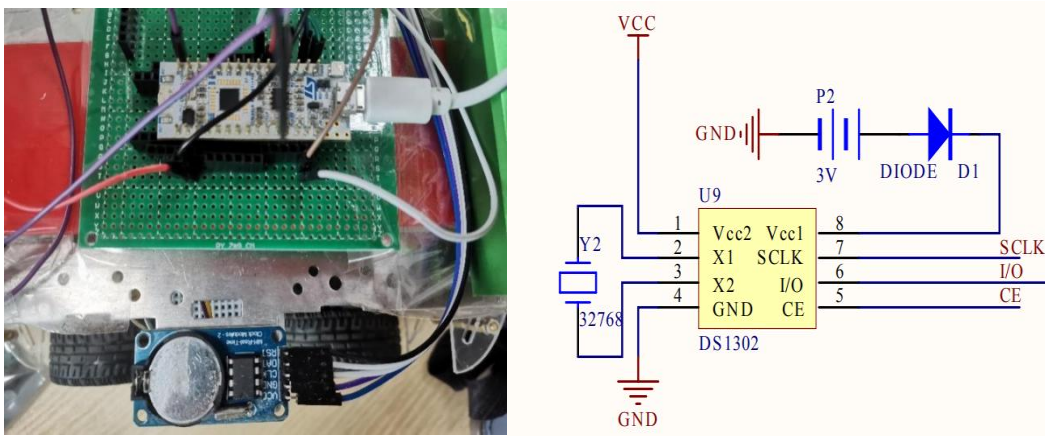


Figure 3.8.5. Physical connection of the clock module (left) and the internal circuit of DS1302 (right)

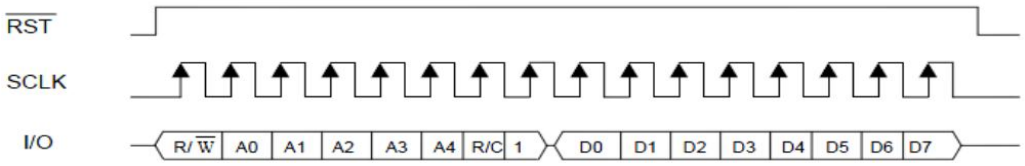


Figure 3.8.6. Diagram of "writing" data

3.8.6 Discussion and Test Results

Problems:

During our field tests, interference occurred as multiple groups were conducting wireless communication tests simultaneously. The received signals of the HC-12 modules were corrupted, resulting in garbled or receiving other groups' information, leading to task failures.

The initial design of HC-12 is shown in **Table 3.8.2**.

Table 3.8. 2. The initial parameter of HC-12

Serial baud rate	9600bps
Communication channel	Channel 001
Transmission mode	FU3

Solutions:

Based on the principles of communication, the signal is modulated by the carrier and sent to the wireless channel [7]. The we can avoid interference by changing the carrier frequency and serial baud rate used for transmission. Because the modulated signal contains infinitely many side frequency components, when some side frequency components fall into the band of the adjacent channel receiver, it will cause adjacent frequency interference [7]. Therefore, one possible method is to change the transmission channel and serial baud rate.

Considering the requirement of 433 MHz transmission and the frequency intervals of 400 kHz between each channel in the range of 001 to 100, we chose channel 003, which is close to 433 MHz, and set the baud rate to 19200 bps to minimize interference. Notably, FU3 transmission mode is retained since this mode ensures the synchronization of the air baud rate and the serial baud rate, making the module more flexible The final setting of HC-12 is shown in **Table 3.8.3**. In the final testing and evaluation stage, the car was able to transmit information accurately within a range of approximately 20 meters, exceeding the length of the testing area. From the theoretical basis and communication results so far, modifying antenna types and other factors can help extend the communication distance. The significant improvement in signal interference can be seen in the final results shown in **Figure 3.8.7**, ensuring a high success rate for the task.

Table 3.8.3. The final set of HC-12

Serial baud rate	19200bps
Communication channel	Channel 003
Transmission mode	FU3

Figure 3.8.7. Final information on the screen

4. Implementation and Analysis

4.1 Overall description

This section will provide an overview of the testing process, test results, and improvements for Patio 1 and Patio 2. In Patio 1, the emphasis was on route tracing, in-position turning, and crossing the bridge. Patio 2 involved addressing tasks such as shape matching, table tennis ball release, and communication. Patio 2 was characterized by a more complex environment, including a larger path distance and challenges posed by cobbled roads. The issues were addressed to ensure seamless transitions between tasks.

4.2 Patio 1

Handled by: Huang Yicheng
UoG: 2614272

4.2.1 Description

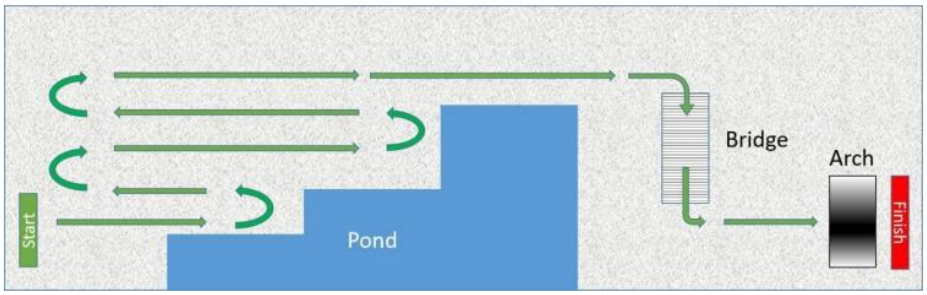


Figure 4.2.1. Route of the car[1]

In Patio 1, the overall task can be divided into three subtasks: route tracing, in-position turning, and crossing the bridge. The smart trolley's primary objective is to patrol the designated line, followed by executing a right turn to cross the bridge. Once the trolley successfully crosses the bridge, it must make an immediate left turn and come to a stop at the arch.

The real site in patio 1 is shown in **Figure.4.2.1**.

Figure 4.2.2. The real site in patio 1 (a) the bridge (b) the curved lines (c) the arch

4.2.2 Design and Integration

In this task, the MCU used is STM32. The module to be used includes: OpenMV module, motion module, ultrasound module.

The basic functions of integration include:

a. Turning control commands from the OpenMV module to the motion module through UART.
b. Communication between the ultrasound module and the OpenMV module through UART.
c. The motion module controls cornering and directional adjustment by adjusting the differential speed on both sides.

The whole task in patio 1 can be divided into 3 tasks, which are tracing the route, in-position turning and cross bridge. The design and integration of the system are explained in detail as follows:

Task 1 Tracing the Route

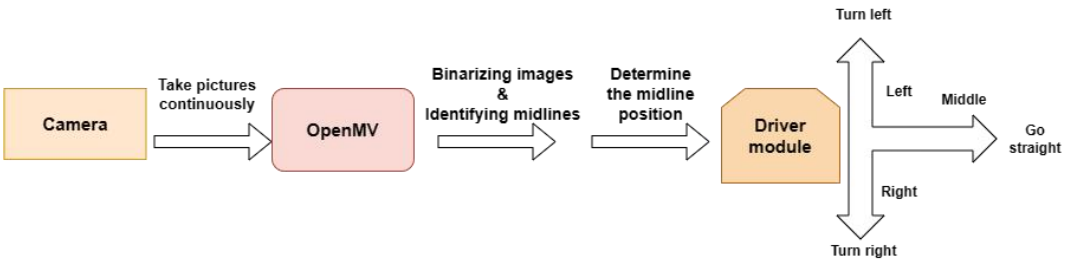


Figure 4.2.3. Workflow of tracing the route

The task of route tracing necessitates the utilization of the OpenMV module to detect the line and establish communication with the driver module for controlling the direction of the smart trolley.

During the initiation of the smart trolley's patrol, the OpenMV module continuously captures images of the surroundings, which are subsequently subjected to a binarization process [2]. By distinguishing the white pixel dots on the processed images, the location of the route's lines within the photos is determined. To facilitate control, the image is divided into four distinct parts. If the line is positioned within the central area, a command is transmitted to the driver module via UART communication to proceed straight. However, if the line falls within the other two parts, the command is issued to initiate a turn by adjusting the differential speed between the wheels on both sides.

This integration of the OpenMV module and driver module enables the smart trolley to effectively trace the route by accurately detecting the line and adjusting its direction accordingly. The image processing capabilities of the OpenMV module, combined with the communication with the driver module, facilitate precise control over the smart trolley's movement, ensuring it stays on track during its patrol.

Task 2 In-position Turning

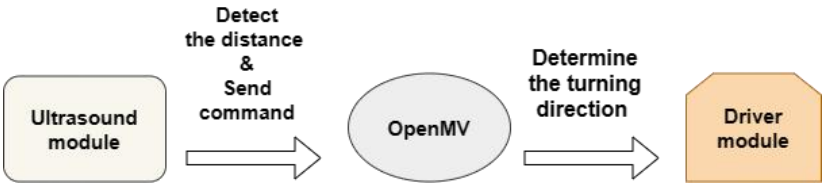


Figure 4.2.4. Workflow of in-position turning

In patio 1, there are two turns required, which are before and after crossing the bridge. communication between ultrasonic module, OpenMV module and motion module was used to let our trolley turn in position.

Before proceeding onto the bridge, precise control of the steering direction is crucial to ensure that the car can make a 90-degree turn to the right and face the bridge directly. Our team's solution involves using the ultrasonic module on the left side of the trolley to detect obstacles placed on the left side during the first turn. When an obstacle is detected, the OpenMV module sends steering commands to the drive module. To control the steering angle, we adjusted the differential speed of the wheels on both sides and fine-tuned them multiple times during field testing. We determined that a velocity difference of 30% duty cycle and 70% duty cycle, along with a remaining time of 1.5 seconds, resulted in a 90-degree rotation [3]. At the turn after crossing the bridge, another obstacle is placed on the right side. If the right ultrasonic module detects this obstacle, the trolley will make a left turn. Once the turn is completed, it will resume line tracking operations.

Task 3 Crossing Bridge

The completion of this task requires communication between the OpenMV module, the motion module, and the ultrasonic module [4,5]. The difficulty of the bridge crossing task is divided into two parts: going up and going down the bridge. Adequate power is needed to get the trolley onto the bridge, and to get off the bridge, the trolley needs to be stable without tipping over. We set a medium speed for the trolley to get on and off the bridge.

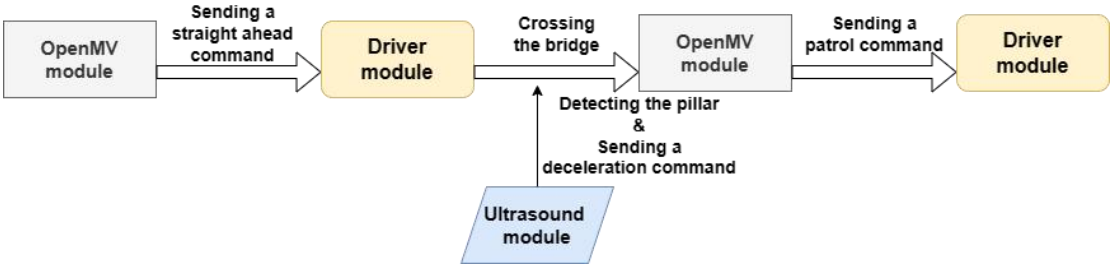


Figure 4.2.5. Workflow of crossing bridge

4.2.3 Results

Task 1 Tracing the Route

During the actual testing phase, challenges were encountered by our team in accurately defining the size of the region in the image, which determined when corrections were made to the trolley's trajectory. When the range for the trolley to proceed straight was set too wide, it led to sluggish responsiveness during line cruising. After conducting numerous experiments, it was eventually determined that a range of 25% on each side was suitable for the car to make effective course corrections, as depicted in **Figure 4.2.6.**

Figure 4.2.6. The successful experiment of patrolling the line

Task 2 In-position Turning

The function was tested by implementing the designs mentioned earlier. The successful turning of the trolley can be observed in **Figure 4.2.7.** During practical tests, we encountered instances where the precise angle of the turn could not be accurately determined. To address this issue, we incorporated the assistance of the OpenMV module to correct the direction of the trolley after each turn. Through these adjustments, we ultimately achieved the objective of the task.

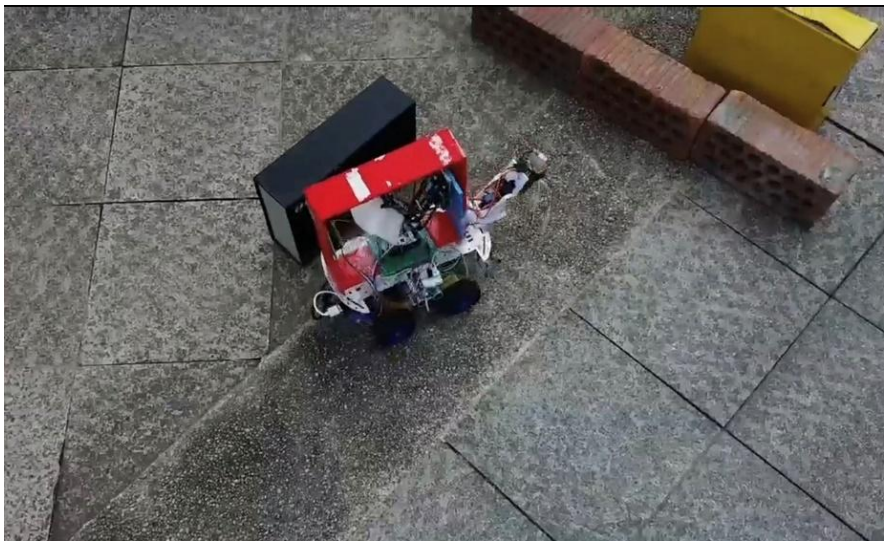


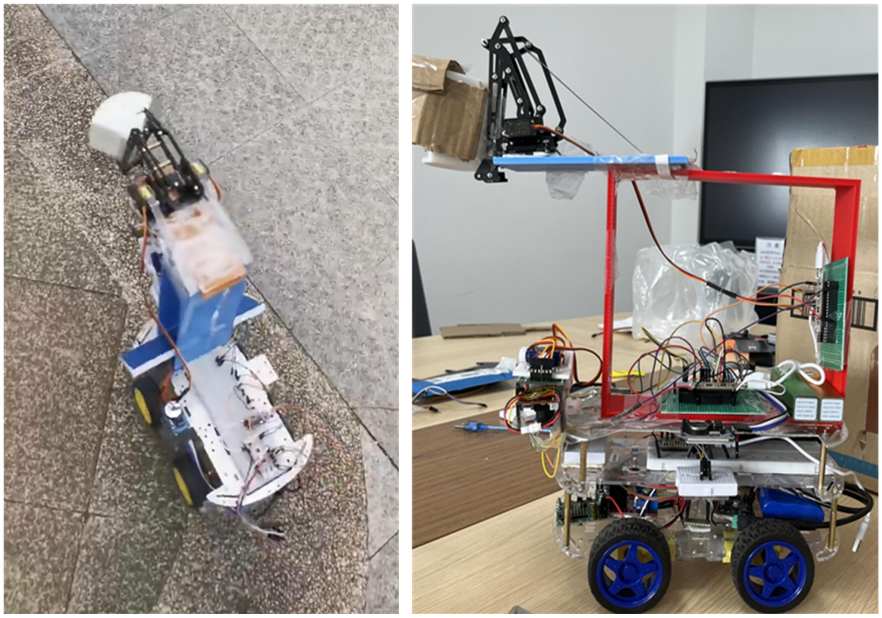
Figure 4.2.7. The successful experiment of in-position turning

Task 3 Crossing Bridge

In the previous tests, it was discovered that if the trolley was driven too fast, it would roll over when going off the bridge. Conversely, if it was driven too slowly, it would be unable to ascend onto the bridge. Two solutions

were devised by our team to address this problem.

The first solution involved changing the structure of the trolley. A comparison diagram showcasing the modifications made to the trolley's structure is shown in **Figure 4.2.8.** The new structure featured a more rearward center of gravity, ensuring that it was not inclined forward when coming off the bridge.



(a) (b)

Figure 4.2.8. Comparison of trolley construction (a)Before (b) After

The second measure involved adjusting the speed of the trolley while on the bridge. When the ultrasonic modules on both sides of the trolley simultaneously detected the columns of the bridge, a command was sent to the motion module, instructing the trolley to slow down. The principle of this process is illustrated in **Figure 4.2.9.**

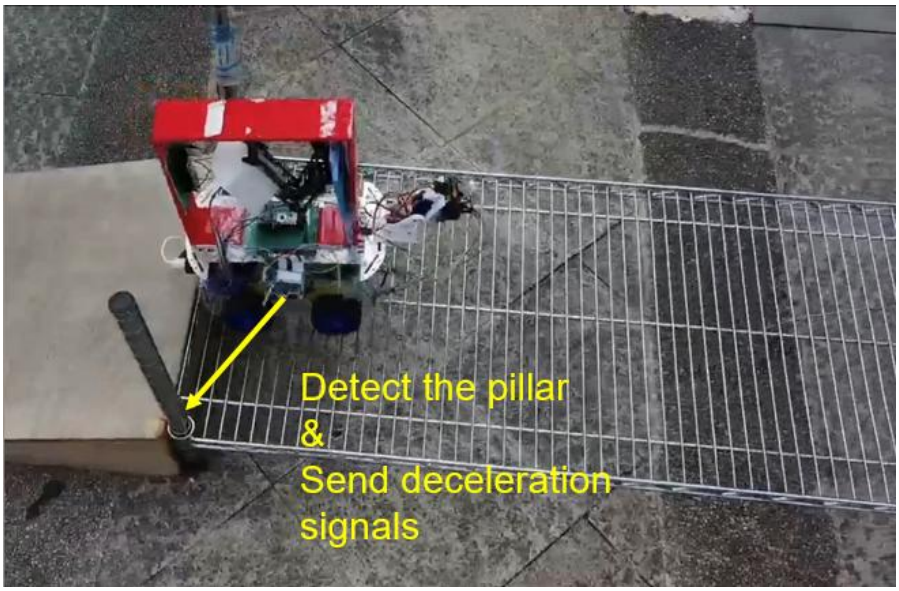


Figure 4.2.9. Trolley deceleration process

With the implementation of these two solutions, the requirements of Task 3 have been successfully met.

4.3 Patio2

Handled by: Wang Hongjing
UoG:2614026W

From previous part, many essential components have been introduced. In patio 2, there are three major tasks to complete. The first one is **shape matching**. In this part, the recognition algorithm, which is introduced in **appendix**, and openMV [1], which is introduced in **part 3.5**, are applied. The second one is **release table tennis ball**. In this part, the ultrasound, which is introduced in **part 3.4**, is applied. The third one is communication part. In this part, the wireless communication [2] and transceiver HC-12 [3], which is introduced in **part 3.8**, is applied.

Compared to patio 1, patio 2 has more complex environment. Firstly, the whole path distance of patio 2 is larger than that of patio 1. Secondly, sometimes the path is cobbled road, which is not flat and will affect the movement of robot car and the detection of ultrasound. Therefore, the path between different tasks is most important and there are many problems need to address.

4.3.1 Task analysis and problems to consider

The tasks in patio 2 and the required steps and methods are illustrated in **Table 4.3.1**.

Table 4.3.1. Tasks in patio 2 and corresponding requirements

Tasks	Shape matching	Release ball	Communication
Requirements 1	Recognize shape	Travel to the release point	Stop when entering planter area
Requirements 2	Turn to correct direction	Successfully release table tennis ball	Transmit message when stop
Requirements 3	Knockout matched shape	Carry ball from task 1 to task 2	Continue to the end of route and stop

For task 1, shape matching, it requires robot car to recognize shape and know out matched shape. From the starting point, the robot will go straight to the certain region where it will read/scan and recognize the shape on the stand of the square to determine direction, then it will turn to correct direction. For example, if it recognize the shape of straight arrow, the robot should rotate 90°in counterclockwise, or turn left. **Table 4.3.2** illustrates different conditions of recognizing different arrows. After rotation, the robot will go straight and knockout one of the matched shapes on the stand. Until know, the task 1 is completed.

Table 4.3.2. Operations or robot when recognizing different arrows

Arrow	**Rotation angle**	**Clockwise/ Counterclockwise**

Left	135°	counterclockwise
Straight	90°	counterclockwise
Right	45°	counterclockwise

For task 2, the operation of release table tennis ball is very easy and it only requires the rotation of steering engine and the ball will be released and fall into the basket. However, the trigger condition of release ball matters the most. In this task, the ultrasound is applied and the specific situation will be introduced later. In addition, one thing that can never be ignored is that the ball should not be bounced out when robot car moving.

For task 3, it has three major requirements. Firstly, it should stop when entering planter area. It depends on the detection of ultrasound. Secondly, when it stops, the message including team's name and the current time, using 24-hour clock, should be transmitted as a radio signal at 433 MHz at fixed data rate using a transceiver, namely Wavesens HC-12 wireless module. It should consider the influenced by other teams and it has been introduced in **part 3.8**. Thirdly, after the receipt of the transmission has been acknowledged, the robot should continue to the end of the route, which is the lower edge of the planter.

The site of patio 2 is composed of fence, planter, cobbled road and flat road so there are many problems to consider.
Table 4.3.3 lists several problems to consider.

Table 4.3.3. Problems to consider

Problems			
How many ultrasonic modules?	Front 1	Left 1	Right 1
Which method?	Line tracking	Ultrasonic ranging	Color block recognition
Environment	Fence	Cobbled road	Light
Precision	Recognition accuracy	Threshold value	Angle
Robustness	Moving	Rotation	
Ball's stability	Successfully be released?	Bounce out when moving?	

- **Ultrasonic module**: Three modules, one in front side, one in left side and one in right side.
- **Methods**: Ultrasonic ranging. Line tracking and clock recognition is possible but worse than ultrasound.
- **Environment**: Fence, or handrail may possible to be followed by robot car. Cobbled road can severely affect the movement of robot car. Light may not affect detection.
- **Precision**: The precision of recognition accuracy mainly depends on the distinguish of left arrow and right arrow. The threshold value of ultrasound is vitally important because it determines the robot car to rotate timely. Besides, the accuracy of rotation angle may fail to achieve task or not able to arrive right region.
- **Robustness and stability**: The usage of wheel instead of caterpillar band may less stable. The table tennis

ball may bounce out when moving through uneven cobbled road.

4.3.2 Route design:

After considering the tasks requirements and possible problems. The route should be designed in advance.

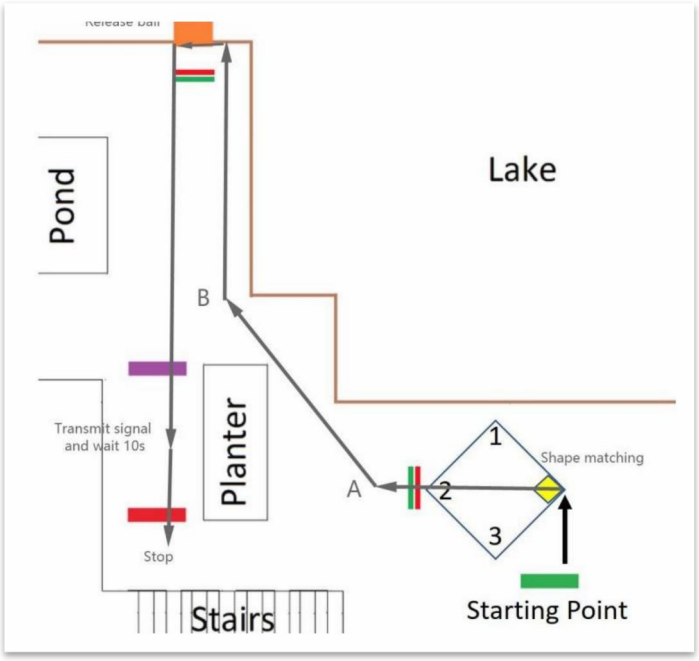


Figure 4.3.1. Initial route design

Above **Figure 4.3.1** illustrates the initial route design. The robot car should follow the blank line and move the paths between different tasks. This route is straightforward and easy to operate. Besides, it will use less time than other route.

However, it exists **many problems** and can easily be influenced by external factors.
Some possible difficulties are as follows:

1. The first step of travel from A to B is just reach point A. However, it is easy just for recognizing straight arrow. If the matched shape is left or right arrow, the robot car will firstly reach point 1 or 3 and it may not easy to find the location of A and towards the correct direction.
2. The path from point A to B is a straight line. However, the distance between planter and handrail, or fence, may narrow. It has high possibility that robot car will dash to planter with just a little deviation.
3. It is also a problem that how to determine the location of point A and B. Two beacons may be placed at these two points but **it has already been used in patio** 1. Another method may calculate fixed time and set the predefined logic or running time. However, it is easy be affected by external interference and it is not automatic, and not smart.
4. The path from point B to basket is uneven cobbled road. Therefore, when the robot car moves, its moving direction may be deviated so the bias should be considered.
5. The wheel may be abrased and the Dupont threads may loose and disconnected with pins.

Above all, the initial route design should be rejected.

To adjust the path from task 1 to task 2, here are two new routes. One is following the fence nearby the lake, and

another one is following the edge of planter and then detecting the basket. The path follows the fence is flat. Eventually, the former one was chosen and the final route design is shown like **Figure 4.3.2.**

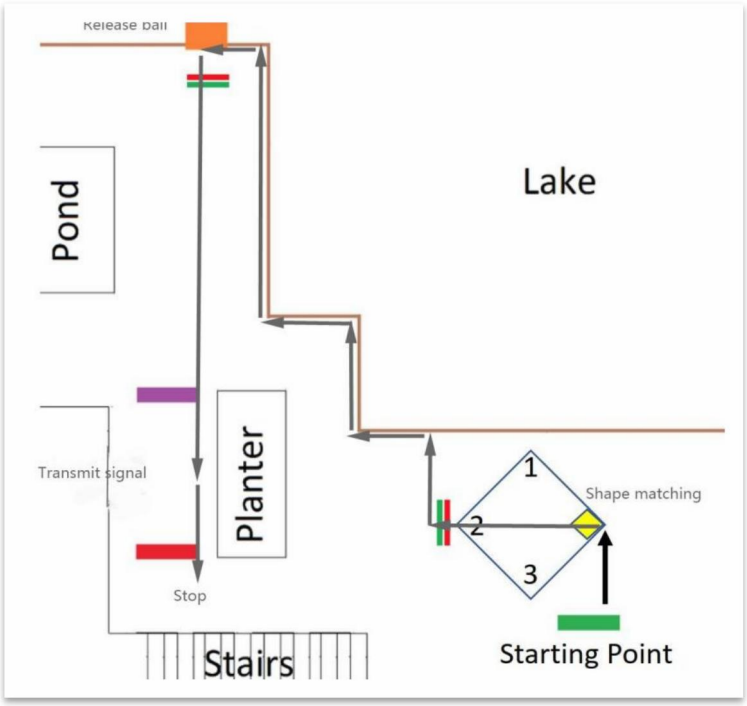


Figure 4.3.2. Final route design

The method of following fence can be line tracking or ultrasonic ranging. However, based on the test of different algorithm, the ultrasonic ranging is more useful. Front and right ultrasonic modules are utilized to judge the condition of rotation and release ball during the path from task 1 to task 2 and left ultrasonic module is utilized to judge the condition of stop in task 3. The basic logic will be introduced in the next part.

4.3.2 Logic flow chart:

In previous part, the essential components and final route design have been introduced. In the next step, the logic of whole patio should be explained in details.

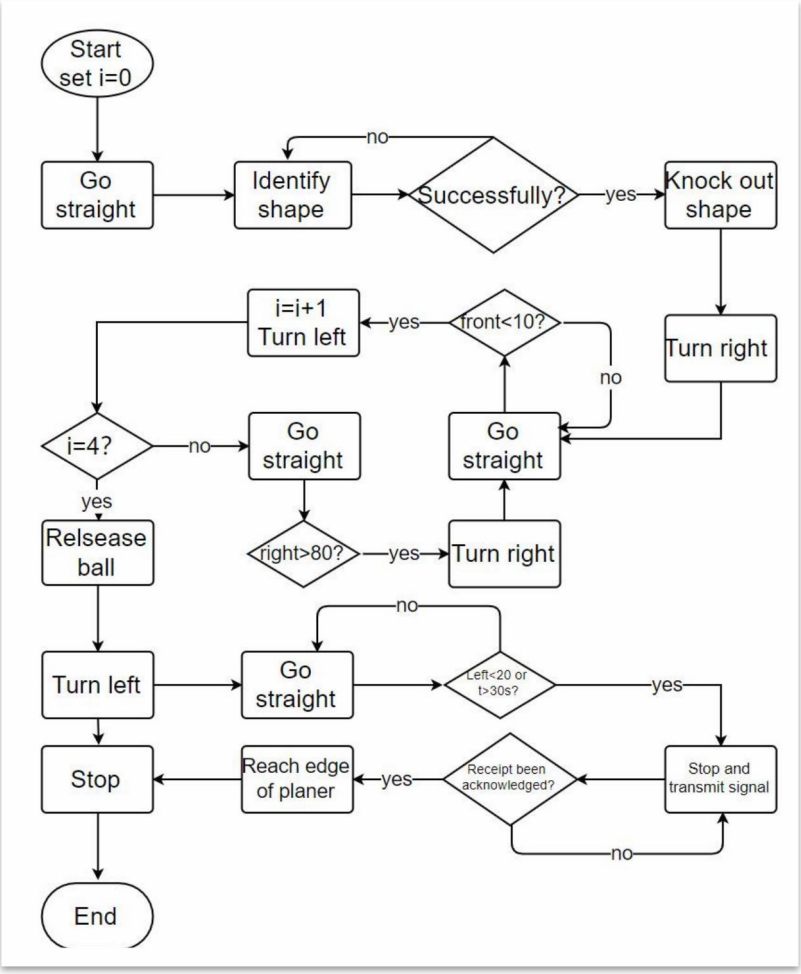


Figure 4.3.3. Flow chart of patio 2

Figure 4.3.3 describes each step of whole process in patio 2.

Firstly, the robot car starts to move and initialize the value of i and set it as 0. The value i is used to counter the times that the distance in front of car is less than 10 centimeters. Then, car will go straight and identify shape. If it identifies arrow successfully, it will go certain area and **knock out shape standing board**. Otherwise, it will recognize the shape again. After knocking out, it will turn right and follow the fence, where is nearby the lake. When the front ultrasound module detect that the distance is less than 10 centimeters, it will rotate 90°in counterclockwise, or turn left, and .let i=i+1. If I equal to 4, it means that the robot car already reaches the area where basket is located, and it will **release table tennis ball**. When the right ultrasound module detect that the distance is more than 80 centimeters, it will rotate 90°in clockwise, or turn right. After releasing ball, it will go task 3 planter. When the left ultrasound module detect that the distance is less than 20 centimeters or the time is more than 30 seconds, it will **stop and transmit signal**. After the receipt of the transmission has been acknowledged, the robot should continue to the end of the route, which is the lower edge of the planter.

4.3.3 Test and result:

In this part, the tests and results of patio 2 will be introduced.

For task 1, **Figure 4.3.4** shows that the robot car successfully achieves this task.

Figure 4.3.4. (a)Before shape matching (b)After shape matching

Initially, the robot was programmed to navigate towards the square located on the right-hand side of the diamond, where one of the three shapes would be positioned. Once the specific shape was identified, the robot proceeded to move towards the corresponding shape within the diamond.

Figure 4.3.5. (a)Before releasing ball (b)After releasing ball

For task 2, **Figure 4.3.5** shows that the robot car successfully achieves this task.
The robot came and reached the task 2 area and it detected that this is the fourth time front ultrasonic module detected the front distance is less than 10 centimeters. Then, it released the table tennis ball.

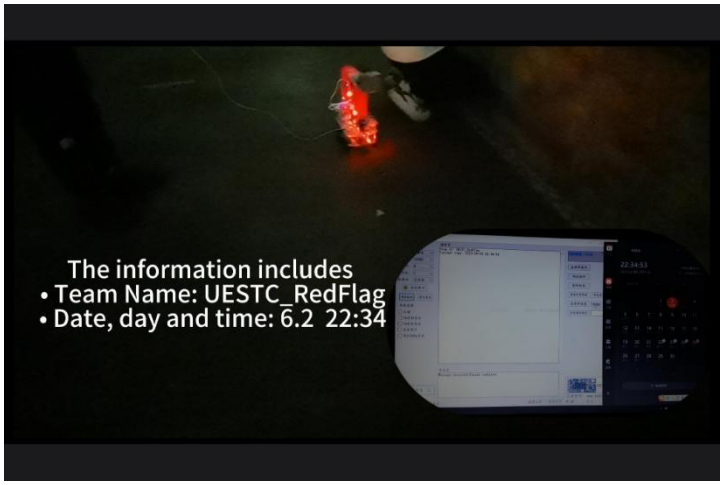


Figure 4.3.6. Stop and transmit signal

For task 3, **Figure 4.3.6** shows that the robot car successfully achieves this task.
When the left ultrasonic module detected that the distance is less than 20 centimeters or the time is more than 30secons, it stopped and transmitted information, including team name and current time.

Reference:

1.4 Literature review

[1] Q. Tian, “Discussion on Smart Car Routing Method Based on Gray Image.” IOP Conference Series: Earth and Environmental Science, vol. 440, no. 4, p. 042087, 2020, doi: 10.1088/1755-1315/440/4/042087.

[2] Y.Zhang, “Smart Car Based on Open MV Vision System.” IOP Conference Series: Earth and Environmental Science, vol. 310, no. 3, p. 032054, 2019, doi: 10.1088/1755-1315/310/3/032054.

2.2 Main control

[1]OpenMV, “OpenMV Cam H7 R2 Camera Board Devices”, openmv.io. (2022). Accessed: May 31st, 2023. [Online]. Available: https://book.openmv.cc/MCU/pyb.html

3.1 Hardware circuit design

[1] R. Prasad and M. Moray, "Switching Power Supply Design," 3rd ed. New York, NY, USA: McGraw-Hill, 2011.

[2] Texas Instruments. (2012). LM2576: SIMPLE SWITCHER® 3.0 A Step-Down Voltage Regulator Datasheet. [Online]. Available: https://www.ti.com/lit/ds/symlink/lm2576.pdf. [Accessed: Jun. 8, 2023].

[3] Advanced Monolithic Systems, Inc. (2013). AMS1117 Low Dropout Voltage Regulator Datasheet. [Online]. Available: https://www.advanced-monolithic.com/pdf/ds1117.pdf. [Accessed: Jun. 8, 2023].

[4] STMicroelectronics. (2003). L298N: Dual Full-Bridge Driver Datasheet. [Online]. Available: https://www.st.com/resource/en/datasheet/l298.pdf. [Accessed: Jun. 8, 2023].

3.2 Motion

[1] “STM32F103x4 STM32F103x6,” Low-density performance line, ARM-based 32-bit MCU with 16 or 32 KB Flash, USB, CAN, 6 timers, 2 ADCs, 6 com. interfaces, https://www.st.com/resource/en/datasheet/stm32f103c4.pdf [Accessed Jun. 4, 2023].

[2] “RM0008 Reference manual,” STM32F101xx, STM32F102xx, STM32F103xx, STM32F105xx and STM32F107xx advanced Arm®-based 32-bit MCUs, https://www.st.com/resource/en/reference_manual/rm0008-stm32f101xx-stm32f102 xx-stm32f103xx-stm32f105xx-and-stm32f107xx-advanced-armbased-32bit-mcus-stmicroelectronics.pdf [Accessed Jun. 4, 2023].
[3] Alldatasheet.com, “AMS1117 Datasheet (PDF) - advanced monolithic systems,” ALLDATASHEET.COM - Electronic Parts Datasheet Search, https://www.alldatasheet.com/datasheet-pdf/pdf/49118/ADMOS/AMS1117.html [Accessed Jun. 4, 2023].

3.3 Mechanical Arm

[1] STMicroelectronics. (2021). STM32L432KC Microcontroller Datasheet. Retrieved from https://www.st.com/resource/en/datasheet/stm32l432kc.pdf [Accessed May. 4, 2023].
[2] STMicroelectronics. (2021). STM32CubeIDE. Retrieved from https://www.st.com/en/development-tools/stm32cubeide.html[Accessed May. 4, 2023].
[3] Guru, B. S., & Hiziroglu, H. R. (2001). Electric Machinery and Transformers. Oxford University Press. [Accessed May. 10, 2023].
[4] Liu, S., Fu, H., & Luo, C. (2018). A survey of control methods for electric servo motor. IEEE Transactions on Industrial Electronics, 65(5), 4075-4086. [Accessed May. 10, 2023].
[5] Huang, Y., & Zhou, Y. (2016). An incremental encoder for position control of servo motor. IEEE Transactions on Robotics, 32(2), 457-463. [Accessed May. 10, 2023].

[6] TowerPro. (n.d.). SG90 Micro Servo. Retrieved from https://www.towerpro.com.tw/product/sg90-7/ [Accessed May. 11, 2023].
[7] Topcopter. (2019). SG90 TowerPro Servo Motor. Retrieved from https://www.topcopter.es/en/arduino/servomotores/sg90-towerpro/ [Accessed May. 11, 2023].
[8] Dassault Systèmes. (Year). "SolidWorks." [Online]. Available: http://www.solidworks.com/. [Accessed: May 12, 2023].
[9] Ferguson, S., & Sprunk, W. (2019). Engineering Fundamentals: An Introduction to Engineering (6th ed.). Boston, MA: Cengage Learning. [Accessed: May 12, 2023].

3.4 Distance Measurement System

[1] ARM Mbed, "NUCLEO-L432KC", os.mbed.com. Accessed: June 5th, 2023. [Online.] Available: https://os.mbed.com/platforms/ST-Nucleo-L432KC/
[2] SparkFun Electronics, "Ultrasonic Ranging Module HC-SR04", cdn.sparkfun.com. Accessed: June 1st, 2023. [Online.] Available: https://cdn.sparkfun.com/ datasheets/Sensors/Proximity/HCSR04.pdf
[3] KAMAMI, "TOF050C - module with distance sensor VL6180X (50cm)", kamami.pl. Accessed: June 3th, 2023. [Online.] Available: https://kamami.pl/en/distance-sensors/587518-tof050c-module-with-distance-sensor-vl6180x-50cm.html
[4] Random Nerd Tutorial, "Complete Guide for Ultrasonic Sensor HC-SR04 with Arduino", randomnerdtutorials.com. Accessed: May 17st, 2023. [Online]. Available: https://randomnerdtutorials.com/complete-guide-for-ultrasonic-sensor-hc-sr04/
[5] ARM Mbed, "Serial", os.mbed.com. Accessed: June 5th, 2023. [Online.] Available: https://os.mbed.com/docs/mbed-os/v5.15/apis/serial.html

3.5 Machine Vision Hardware

[1] OpenMV, "OpenMV Cam H7 R2 Camera", openmv.io. (2022). Accessed: May 31st, 2023. [Online]. Available: https://singtown.com/product/50908/openmv4-h7-r2/

3.6 Line Tracking

[1] Yuan, Suzhen et al. "Improved quantum dilation and erosion operations." *International Journal of Quantum Information* 14 (2016): 1650036.
[2] Ji, Liang et al. "Erosion and dilation of binary images by arbitrary structuring elements using interval coding." *Pattern Recognit. Lett.* 9 (1989): 201-209.
[3] Canny, John. "A computational approach to edge detection." *IEEE Transactions on pattern analysis and machine intelligence* 6, pp.679-698, Nov. 1986, doi: 10.1016/B978-0-08-051581-6.50024-6
[4] Kushal Gowda, "Proportional, Integral, and Derivative Control 4.2", CircuitBread, Accessed: Jun. 4, 2023. [Online]. Available: https://www.circuitbread.com/tutorials/proportional-integral-and-derivative-control-4-2

3.7 Shape Matching

[1] Feng Zhao, Qingming Huang, and W. Gao, "Image Matching by Normalized Cross-Correlation," in 2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings, Toulouse, France: IEEE, 2006, p. II–II. doi: 10.1109/ICASSP.2006.1660446.
[2] H. Zhang, J. Wohlfeil, and D. Grießbach, "Extension and evaluation of the AGAST Feature Detector," ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. III–4, pp. 133–137, 2016. doi:10.5194/isprsannals-iii-4-133-2016.
[3] R. Chauhan, K. K. Ghanshala, and R. C. Joshi, "Convolutional Neural Network (CNN) for Image Detection and Recognition," in 2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC), Jalandhar, India: IEEE, Dec. 2018, pp. 278–282.

doi: 10.1109/ICSCCC.2018.8703316.

[4]I. Abdelkader, Y. El-Sonbaty, and M. El-Habrouk, “OPENMV: A PYTHON POWERED, EXTENSIBLE MACHINE VISION CAMERA,” Data Mining and Computational Intelligence, 2017.

[5] A. Nieto-Castanon, S. S. Ghosh, J. A. Tourville, and F. H. Guenther, “Region of interest based analysis of functional imaging data,” NeuroImage, vol. 19, no. 4, pp. 1303–1316, 2003. doi:10.1016/s1053-8119(03)00188-5.

[6] S. Murali and V. K. Govindan, “Shadow detection and removal from a single image using lab color space,” Cybernetics and Information Technologies, vol. 13, no. 1, pp. 95–103, 2013. doi:10.2478/cait-2013-0009.

[7] R. Susmaga, “Confusion matrix visualization,” Intelligent Information Processing and Web Mining, pp. 107–116, 2004. doi:10.1007/978-3-540-39985-8_12.

[8] “Complete guide on complexity analysis - data structure and algorithms tutorial,” GeeksforGeeks, https://www.geeksforgeeks.org/complete-guide-on-complexity-analysis/ (accessed Jun. 10, 2023).

3.8 Wireless Communication

[1] Antoine et al., “Arduino and HC-12 Long Range Wireless Communication Module,” How To Mechatronics, https://howtomechatronics.com/tutorials/arduino/arduino-and-hc-12-long-range-wireless-communication-module/ (accessed Jun. 10, 2023).

[2] “STM32 ultra low power mcus,” STMicroelectronics, https://www.st.com/en/microcontrollers-microprocessors/stm32-ultra-low-power-mcus.html (accessed Jun. 10, 2023).

[3] “Make your own GPS transmitter with the HC-12 transceiver - projects,” All About Circuits, https://www.allaboutcircuits.com/projects/gps-transmission-with-the-hc-12-transmitter/ (accessed Jun. 10, 2023).

[4] R. Zhong, X. Yang and L. Diao, "A design of anti-interference electricity and integrated protection device for low-voltage three-phase asynchronous motor based-on DS1302," Power System Protection and Control, vol. 37, (23), pp. 100, 2009.

[5] DS1302 trickle-charge timekeeping chip - analog devices, https://www.analog.com/media/en/technical-documentation/data-sheets/DS1302.pdf (accessed Jun. 9, 2023).

[6] Microcontrollers Lab, “DS1302 real time clock (RTC) chip,” Microcontrollers Lab, https://microcontrollerslab.com/ds1302-rtc-chip-pinout-example-arduino-applications/ (accessed Jun. 10, 2023).

[7] D. HERCOG, COMMUNICATION PROTOCOLS: Principles, Methods and Specifications. 2020. DOI: 10.1007/978-3-030-50405-2.

4.3 Patio1

[1] “UofG Secure Login - Loading Session Information,” moodle.gla.ac.uk. https://moodle.gla.ac.uk/pluginfile.php/6274039/mod_resource/content/11/TDPS_ProjectHandbook_2022-2023.pdf (accessed Jun. 23, 2023).

[2] Y. Zhang, "Smart Car Based on Open MV Vision System," IOP Conference Series. Earth and Environmental Science, vol. 310, (3), pp. 32054, 2019.

[3] L. Zheng et al, "Dynamic Analysis and Path Planning of a Turtle-Inspired Amphibious Spherical Robot," Micromachines (Basel), vol. 13, (12), pp. 2130, 2022.

[4] F. Gong and Y. M. Zhang, "Design of intelligent throwing robot based on machine vision," Journal of Physics. Conference Series, vol. 1939, (1), pp. 12004, 2021.

[5] L. Liu et al, "Design of Intelligent Logistics Car based on STM32," Journal of Physics. Conference Series, vol. 1952, (4), pp. 42102, 2021.

4.3 Patio2

[1] SingTown technology, OpenMV. Accessed on: May 23, 2023. [Online]. Available: https://book.openmv.cc/

[2] Arm mbed, Serial. Accessed on: May 23, 2023. [Online]. Available: https://os.mbed.com/docs/mbed-os/v5.15/apis/serial.html

[3] HC TECH, HC-12. Accessed on: May 23, 2023. [Online]. Available: https://www.hc01.com/goods/640e91920be12d0114404c95

Appendix A (Bill)

Table A.1 documents the materials and their prices involved in this project.

Table A.1. Bill of Materials

Category	Element	Cost(RMB)
Vision	OpenMV Cam H7	436
Communication	HC SR-04	60
	DS1302	12.61
MCU	STM32 L432 KC x 3	240
Power	12 V Battery x 3	15.8 x 3 = 47.4
	Power Bank x 2	60
Mechanical Arm	Mechanical Arm	45
Others	Main Body	80.21
	Wires	9.98
Total		991.2

Appendix B (Core code)

Patio 1

```
import sensor, image, time, math
import pyb
from pyb import UART
import function
uart = UART(3,115200)
sonic = UART(1, 115200)
angle = pyb.Servo(2)
horizontal = pyb.Servo(1)
width = 320
height = 240


sensor.reset()
sensor.set_pixformat(sensor.GRAYSCALE)
sensor.set_framesize(sensor.QVGA)
sensor.set_gainceiling(8)
sensor.skip_frames(time = 2000)
sensor.set_hmirror(True)
sensor.set_vflip(True)

stage = 0

# First stage - Line tracking

angle.angle(45) # todo
horizontal.angle(0)

while(1):
    print("stage:", stage)

    img = sensor.snapshot().gaussian(1)
    img_matrix = img.find_edges(image.EDGE_CANNY, threshold=(70, 79))
    his = function.count_white_pixels(img_matrix)
    middle_line = (function.find_middle_line(his)+1) * 8
    img.draw_line((middle_line, 0, middle_line, img.height()-1), color = 255, thickness = 2)
    rho_err = abs(middle_line)-img.width()/2
    #print("rho_error:", rho_err)
    rho_output = (int)(rho_err / (img.width()/2) * 4);

    print("rho_output:", rho_output)
```

```
        left = "3"
        right = "3"
        command = "rotate"
        if rho_err > 40:
            right = str(2)
            left = str(4 + rho_output)
            command = "rightm"
        elif rho_err < -40:
            right = str(4 - rho_output)
            left = str(2)
            command = "leftm"
        #command = "rotate";
        command = command + left + right + "\r\n";
        print(command)
        uart.init(115200,8,None,1)
        uart.write(command)

        sonic.init(115200,8,None,1)
        d = function.read_distance(sonic)
        print(d)
        if d[0] > 0 and d[1] < 40:
            break

stage += 1
print("stage:", stage)


for i in range(1000):
    uart.write("stop\r\n")

for i in range(1000):
    uart.write("rightm63\r\n")

time.sleep_ms(1600)

for i in range(1000):
    uart.write("stop\r\n")

for i in range(1000):
    uart.write("rotate55\r\n")

while(1):
    sonic.init(115200, 8, None, 1)
    d = function.read_distance(sonic)
    print(d)
    if d[0] > 0 and d[1] < 40 and d[2] < 40:
```

```
            break;

for i in range(1000):
    uart.write("rotate33\r\n")


while(1):
    sonic.init(115200, 8, None, 1)
    d = function.read_distance(sonic)
    print(d)
    if d[0] > 0 and d[0] < 99 and d[2] < 40:
        break;


for i in range(1000):
    uart.write("leftm36\r\n")

time.sleep_ms(2000)

for i in range(1000):
    uart.write("stop\r\n")

stage += 1
print("stage:", stage)

angle.angle(60)

while(1):
    print("stage:", stage)

    img = sensor.snapshot().gaussian(1)
    img_matrix = img.find_edges(image.EDGE_CANNY, threshold=(70, 70))
    his = function.count_white_pixels(img_matrix)
    middle_line = (function.find_middle_line(his)+1) * 8
    img.draw_line((middle_line, 0, middle_line, img.height()-1), color = 255, thickness = 2)
    rho_err = abs(middle_line)-img.width()/2
    #print("rho_error:", rho_err)
    rho_output = (int)(rho_err / (img.width()/2) * 4);

    print("rho_output:", rho_output)
    left = "3"
    right = "3"
    command = "rotate"
    if rho_err > 40:
        right = str(2)
        left = str(4 + rho_output)
```

```
        command = "rightm"
    elif rho_err < -40:
        right = str(4 - rho_output)
        left = str(2)
        command = "leftm"

    command = command + left + right + "\r\n";
    print(command)
    uart.init(115200,8,None,1)
    uart.write(command)

    sonic.init(115200,8,None,1)
    d = function.read_distance(sonic)
    print(d)
    if d[0] > 0 and d[1] < 40 and d[2] < 40:
        break

for i in range(1000):
    uart.write("stop\r\n")
```

Line Tracking

```
import sensor, image, time, math
import pyb
from pyb import UART
uart = UART(3,115200)
width = 320
height = 240
angle2 = pyb.Servo(2)
angle2.angle(60) # Adjust to -30

sensor.reset() # Initialize the camera sensor.
sensor.set_pixformat(sensor.GRAYSCALE) # or sensor.RGB565
sensor.set_framesize(sensor.QVGA) # or sensor.QVGA (or others)
sensor.set_gainceiling(8)
sensor.skip_frames(time = 2000) # Let new settings take affect.
sensor.set_hmirror(True)
sensor.set_vflip(True)


flag = 1
stage = 1
while(flag == 1):
    if stage == 1: # first stage
```

```
        img = sensor.snapshot().gaussian(1)
        img_matrix = img.find_edges(image.EDGE_CANNY, threshold=(70, 79))
        his = count_white_pixels(img_matrix)
        middle_line = (find_middle_line(his)+1) * 8
        img.draw_line((middle_line, 0, middle_line, img.height()-1), color = 255, thickness = 2)
        rho_err = abs(middle_line)-img.width()/2
        print("rho_error:", rho_err)
        rho_output = (int)(rho_err / (img.width()/2) * 4);


        print("rho_output:", rho_output)
        left = "3"
        right = "3"
        command = "rotate";
        #uart.write("leftm23\r\n")
        if rho_err > 40:
            right = str(2)
            left = str(4 + rho_output)
            command = "rightm"
        elif rho_err < -40:
            right = str(4 - rho_output)
            left = str(2)
            command = "leftm"
        command = command + left + right + "\r\n";
        print(command)
        uart.write(command)
```

Function

```
import sensor, image, time
from pyb import Pin, UART


def count_white_pixels(b_array):
    his = list()
    for i in range(39):
        real_column = (i + 1) * 8 - 1
        his.append(0)
        index = real_column
        his[i] += b_array[index]

        for j in range(239):
            index = index + 320
            his[i] += b_array[index]
```

```
    return his

def find_middle_line(his):
    sorted_list, sorted_index = sort_with_index(his)
    for i in range(25):
        his[sorted_index[i]] = 0
    half_sum = int(sum(his) / 2)
    temp = 0
    for i in range(39):
        temp += his[i]
        if temp >= half_sum:
            if i < 20 and i != 0:
                return (i-1)
            else:
                return i

def sort_with_index(a):
    b = sorted(enumerate(a), key=lambda a:a[1])
    sorted_index = [a[0] for a in b]
    sorted_list = [a[1] for a in b]
    return sorted_list, sorted_index


def read_distance():
    time.sleep_ms(200)
    distance_read = str(sonic.readline())

    distance_read = distance_read[2:-3].split(' ')
    print(distance_read)
    flag = True
    if len(distance_read) < 3:
        flag = False
    for d in distance_read:
        if not d.isdigit():
            flag = False
    if flag:
        int_list = [int(i) for i in distance_read]
        for i in int_list:
            if i < 1 or i > 99:
                int_list = [0]
                break
    else:
        int_list = [0]

    return int_list
```

Patio 2

```
import sensor
import image

sensor.reset()
sensor.set_pixformat(sensor.RGB565)
sensor.set_framesize(sensor.QQVGA)
sensor.skip_frames(time = 2000)


while(1):

    img = sensor.snapshot()
    # LAB threshold
    threshold = (0, 37, -128, 127, -128, 127)
    blobs = img.find_blobs([red_threshold])

    # ROI parameters
    tx=ty=tw=th=0
    # largest blob area
    maxx=-1
    for b in blobs:
        (x,y,w,h)=b.rect()
        if(maxx < w*h):
            maxx=w*h;
            tx=x
            ty=y
            tw=w
            th=h

    img.draw_rectangle(tx,ty,tw,th)

    if(th>tw):
        print("up")
    else:
        left_roi = (tx,ty,int(tw/2),th)
        right_roi = (tx+int(tw/2),ty,int(tw/2),th)

        # color blob segmentation for the left and right parts
        left_blobs = img.find_blobs([red_threshold], roi=left_roi)
        right_blobs = img.find_blobs([red_threshold], roi=right_roi)
```

```
        img.draw_rectangle(left_roi)
        img.draw_rectangle(right_roi)

        #number of matched pixels in the left and right parts
        sum1=sum2=0
        for b in left_blobs:
            sum1=sum1+b.pixels()
        for b in right_blobs:
            sum2=sum2+b.pixels()
        if(sum1>sum2):
            print("left")
        else:
            print("right")
```

Ultrasonic

```
#include "mbed.h"
#include "platform/mbed_thread.h"
#include "string"
#include "HCSR04.h"

DigitalOut myled(LED1);
DigitalIn send_signal(D6,PullDown);
HCSR04 front_sonar(D2, D3);
HCSR04 left_sonar(D4, D5);
HCSR04 right_sonar(D9, D10);

int distance_front();
int distance_left();
int distance_right();
Serial sonic(D1, D0); // tx yellow, rx green,

int main() {
    int distance_F;
    int distance_L;
    int distance_R;
    myled=0;
    int count = 0;
    sonic.baud(115200);
    while(1){
        myled=1;
        wait_us(12500);
        distance_F = distance_front();
        distance_L = distance_left();
```

```
            distance_R = distance_right();
            //printf("Front distance is %d cm\n", distance_F);
            //printf("Left distance is %d cm\n", distance_L);
            //printf("Right distance is %d cm\n", distance_R);
            sonic.printf("%d %d %d\n",distance_F, distance_L, distance_R);
            myled=0;
            wait_us(12500);
            }

}

int distance_front(){
    int get_distance=0;
    int distance=0;
    get_distance=int(front_sonar.getCm());

    if(get_distance>99){
        distance=99;
    }
    else if(get_distance<2){
        distance=2;
    }
    else{
        distance=get_distance;
    }
    return distance;
}
int distance_left(){
    int get_distance=0;
    int distance=0;
     get_distance=int(left_sonar.getCm());

    if(get_distance>99){
        distance=99;
    }
    else if(get_distance<2){
        distance=2;
    }
    else{
        distance=get_distance;
    }
    return distance;
}
int distance_right(){
    int get_distance=0;
    int distance=0;
```

```
    get_distance=int(right_sonar.getCm());

    if(get_distance>99){
        distance=99;
    }
    else if(get_distance<2){
        distance=2;
    }
    else{
        distance=get_distance;
    }
    return distance;
}
```

Communication

```
#include "mbed.h"
#include "DS1302.h"

// DS1302 I/O pin definitions
DigitalOut ds1302_clk_pin(D2);
DigitalInOut ds1302_dat_pin(D3);
DigitalOut ds1302_rst_pin(D4);
Serial pc(D1, D0);
// DS1302 instance
DS1302 rtc(D2, D3, D4);
DigitalIn button(D12);
int main()
{
    // Initialize DS1302
    rtc.init();

    // Set initial time to 3:52:30 on January 1, 2023
    rtc.setTime(23, 6, 3, 9, 11, 0);
    pc.baud(9600);
    pc.format(8, Serial::None,1); //receive data and transport it to laptor
    while (1) {

        // Get current time
        uint8_t year, month, day, hour, minute, second;
        rtc.getTime(&year, &month, &day, &hour, &minute, &second);
        if (button) {
            // Output the current time via serial communication
            pc.printf("Team: Redflag\r\n");
```

```
            pc.printf("Current time: %02d-%02d-%04d %02d:%02d:%02d\r\n",
                      day, month, year, hour, minute, second);
        }
        // Print current time

        // Delay for 1 second
        ThisThread::sleep_for(1000);
    }
}
```

Drive Module

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  * @attention
  *
  * <h2><center>&copy; Copyright (c) 2021 STMicroelectronics.
  * All rights reserved.</center></h2>
  *
  * This software component is licensed by ST under BSD 3-Clause license,
  * the "License"; You may not use this file except in compliance with the
  * License. You may obtain a copy of the License at:
  *                        opensource.org/licenses/BSD-3-Clause
  *
  ******************************************************************************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "led.h"
#include "motor_driver.h"
#include "my_usart.h"
#include <string.h>

/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
```

```
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
TIM_HandleTypeDef htim1;

UART_HandleTypeDef huart1;

/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART1_UART_Init(void);
static void MX_TIM1_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */
    u16 times=0;
    u8 i=0;
    char* command;
```

```
    u8 speed_now = 0;
    u8 speed_left = 0;
    u8 speed_right = 0;


  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_USART1_UART_Init();
  MX_TIM1_Init();
  /* USER CODE BEGIN 2 */
  HAL_UART_Receive_IT(&huart1, (u8 *)aRxBuffer, RXBUFFERSIZE);//该函数会开启接收中断：
标志位 UART_IT_RXNE，并且设置接收缓冲以及接收缓冲接收最大数据量



  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
        if(USART_RX_STA&0x8000){


            command = (char*)USART_RX_BUF;
            //printf("%s", command);

            if(strstr(command, "rotate")) {
```

```
        int p = 0;
        for(int i = 0; i <=15; ++i)
        {
            if(command[i] == 'e') p = i;
        }
        speed_left = command[p + 1] - '0';
        speed_right = command[p + 2] - '0';
        motor_left_speed(speed_left);
        motor_right_speed(speed_right);
    } else if(strcmp(command, "stop")==0){
        motor_stop();
    } else if(strstr(command, "leftm")) {
      int p = 0;
        for(int i = 0; i <=15; ++i)
        {
            if(command[i] == 'm') p = i;
        }
        speed_left = command[p + 1] - '0';
        speed_right = command[p + 2] - '0';
        motor_left_back(speed_left);
        motor_right_speed(speed_right);
    } else if(strstr(command, "rightm")) {
      int p = 0;
        for(int i = 0; i <=15; ++i)
        {
            if(command[i] == 'm') p = i;
        }
        speed_left = command[p + 1] - '0';
        speed_right = command[p + 2] - '0';
        motor_left_speed(speed_left);
        motor_right_back(speed_right);
    } else if(strstr(command, "bothm")) {
      int p = 0;
        for(int i = 0; i <=15; ++i)
        {
            if(command[i] == 'm') p = i;
        }
        speed_left = command[p + 1] - '0';
        speed_right = command[p + 2] - '0';
        motor_left_back(speed_left);
        motor_right_back(speed_right);
    }


    for(i=0;i<=15;i++){
        command[i]=0;
```

```
            }



            USART_RX_STA=0;
        }else{
            HAL_Delay(100);
            //times++;
            //if(times%100==0)printf("\r\nALIENTEK\r\n");

//
//              if(times%30==0)LED1_Toggle;//闪烁 LED,提示系统正在运行.
        }


    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.HSEPredivValue = RCC_HSE_PREDIV_DIV1;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLMUL = RCC_PLL_MUL6;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
```

```
  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                                |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
  {
    Error_Handler();
  }
}

/**
  * @brief TIM1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_TIM1_Init(void)
{

  /* USER CODE BEGIN TIM1_Init 0 */

  /* USER CODE END TIM1_Init 0 */

  TIM_ClockConfigTypeDef sClockSourceConfig = {0};
  TIM_MasterConfigTypeDef sMasterConfig = {0};

  /* USER CODE BEGIN TIM1_Init 1 */

  /* USER CODE END TIM1_Init 1 */
  htim1.Instance = TIM1;
  htim1.Init.Prescaler = 71;
  htim1.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim1.Init.Period = 999;
  htim1.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim1.Init.RepetitionCounter = 0;
  htim1.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_ENABLE;
  if (HAL_TIM_Base_Init(&htim1) != HAL_OK)
  {
    Error_Handler();
  }
  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim1, &sClockSourceConfig) != HAL_OK)
```

```
  {
    Error_Handler();
  }
  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim1, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN TIM1_Init 2 */
  HAL_TIM_Base_Start_IT(&htim1);


  /* USER CODE END TIM1_Init 2 */

}

/**
  * @brief USART1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_USART1_UART_Init(void)
{

  /* USER CODE BEGIN USART1_Init 0 */

  /* USER CODE END USART1_Init 0 */

  /* USER CODE BEGIN USART1_Init 1 */

  /* USER CODE END USART1_Init 1 */
  huart1.Instance = USART1;
  huart1.Init.BaudRate = 115200;
  huart1.Init.WordLength = UART_WORDLENGTH_8B;
  huart1.Init.StopBits = UART_STOPBITS_1;
  huart1.Init.Parity = UART_PARITY_NONE;
  huart1.Init.Mode = UART_MODE_TX_RX;
  huart1.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart1.Init.OverSampling = UART_OVERSAMPLING_16;
  if (HAL_UART_Init(&huart1) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN USART1_Init 2 */
```

```
  /* USER CODE END USART1_Init 2 */

}

/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOD_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();
  __HAL_RCC_GPIOB_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOA, motor_A_1_Pin|motor_A_2_Pin|motor_B_1_Pin, GPIO_PIN_RESET);


  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOB, motor_B_2_Pin|GPIO_PIN_12|GPIO_PIN_13, GPIO_PIN_RESET);

  /*Configure GPIO pins : motor_A_1_Pin motor_A_2_Pin motor_B_1_Pin */
  GPIO_InitStruct.Pin = motor_A_1_Pin|motor_A_2_Pin|motor_B_1_Pin;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_PULLDOWN;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
  HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /*Configure GPIO pins : motor_B_2_Pin PB12 PB13 */
  GPIO_InitStruct.Pin = motor_B_2_Pin|GPIO_PIN_12|GPIO_PIN_13;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_PULLDOWN;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_HIGH;
  HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/**
```

```
  * @brief  This function is executed in case of error occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  /* User can add his own implementation to report the HAL error return state */
  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  /* User can add his own implementation to report the file name and line number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */

/************************ (C) COPYRIGHT STMicroelectronics *****END OF FILE****/
```