Smart Robot Design and Demonstration

Team Design Project and Skills Final Report

Team 41 "Furious"

Team Members

Li Ruixuan 2614435 2020190506032	Wang Ruiqi 2614425 2020190506022	Guo Yixiao 2614409 2020190506006	Wang Jiaqi 2614441 2020190506038	Liu Jingrui 2613972 2020190902029
Zeng Yu 2613964 2020190902021	Peng Yankai 2613970 2020190902027	Zhang Jinyang 2613975 2020190902032	Li Yusen 2613971 2020190902028	Luo Jiaqi 2613952 2020190902009

[In no particular order, equal contribution]

电子科技大学
格拉斯哥学院
Glasgow College, UESTC

Smart Robot Design And Demonstration

Team Design Project and Skills Final Report

FURIOUS



Team 41
“Furious”

Abstract

Smart robots have become an indispensable facet of present-day life, offering a convergence of technological advancements and advanced features. These robots can operate a diverse range of tasks and have substantially impacted people's daily work. The Team Design Project and Skills (TDPS) aim at working collaboratively on the design and construction of smart robots. The team's primary objective is to create a robot with integrated electronic systems that can perform specific functions. With ten students pooling their skills and efforts, the team aims to efficiently achieve its goal. The team has complete flexibility to choose appropriate hardware components that align with their blueprint while operating within a designated budget.

The design and anticipated outcomes of the robotic car are capable of performing distinct tasks on two separate patios. The robotic car incorporates three fundamental systems: the Sensor System, Executive Control System, and Wireless Communication System. The Sensor System combines a visual processor, ultrasonic sensors, and an accelerometer and gyroscope module to enable shape detection, distance measurement, and precise motion control. The Executive Control System integrates a microcontroller, a robotic arm with a servo actuator, and a motor drive circuit, enabling signal processingarmsrm control, and motor driving. The Wireless Communication System facilitates seamless transmission of essential data to a laptop using a wireless transceiver. Through the harmonious integration of these advanced systems, the robotic car is expected to demonstrate seamless operation, versatility, and accuracy in executing tasks on both patios, showcasing its effectiveness and contributing to the desired outcomes.

The successful implementation of the robotic system was a collective effort, with each team member contributing to the development of subsystems and integration of the overall robot control system. The project management was excellent, facilitated by a well-coordinated team schedule that ensured the timely completion of tasks. This project has provided invaluable teamwork experience for each team member, highlighting the importance of collaboration and coordination in achieving project goals.

Acknowledgement

We would like to express our sincere gratitude to the following individuals and organizations who have contributed to the successful completion of the Team Design Project and Skills - Smart Robot.

Our supervisor, Professor Oluwakayode Onireti and Abdullah Al-Khalidi, for their invaluable guidance and support throughout the duration of this project. Their expertise and advice helped us clarify the project requirements, provided valuable suggestions for manufacturing, and offered guidance on various details. Their input significantly contributed to the overall success of our Smart Robot.

The teaching assistants, Xu Dongchen, who provided assistance and feedback at various stages of the project. His availability, patience, and willingness to address our queries were immensely valuable in refining our understanding and improving the robot's functionality. Glasgow college, UESTC, for providing the necessary resources, facilities, and infrastructure that facilitated the smooth execution of the project. We are grateful for the support received from the institution.

The exceptional members of our team who demonstrated unwavering dedication, working day and night, sacrificing countless hours of rest, and showcasing true teamwork spirit. Their relentless efforts, commitment, and collaboration were instrumental in bringing the Smart Robot to life. Each team member brought their unique skills, expertise, and enthusiasm, contributing to different aspects of the project, from design and programming to testing and integration.

We extend our deepest appreciation to everyone mentioned above. Your support and contributions have been instrumental in our achievements, and we are truly grateful for your assistance.

Team 41 "*Furious*"

Chengdu, June 2023

Contents

Abstract III
Acknowledgement IV
Contents V
Abbreviation VII
List of Figures VIII
List of Tables X
1. Introduction 1
1.1. Background 1
1.2. Task Analysis 1
1.2.1. Patio 1 1
1.2.2. Patio 2 2
1.2.3. General Rules 3
2. Top-level Structure Design 3
2.1. Sensor System 3
2.2. Executive Control System 4
2.2.1. Wireless Communication System 5
3. Implementation 5
3.1. Sensor System Implementation: Detection for Patrol and Adjustments 5
3.1.1. Visual Processor 5
3.1.2. Ultrasonic Ranging Sensor 16
3.1.3. Accelerometer and Gyroscope 17
3.2. Executive Control System Implementation: Robot Control and Operation 23
3.2.1. MCU 23
3.2.2. Power Regulator Circuit 24
3.2.3. Battery Monitoring Circuit 26
3.2.4. Motor Drive Circuit 27
3.2.5. Motors 29
3.2.6. PCB Integration 30
3.2.7. Mechanical Arm 33
3.3. Wireless Communication System Implementation: Message Delivery 35
3.3.1. HC-12 36
3.3.2. UART 36
3.3.3. Wireless data transmission 37
3.3.4. DS1302 Clock Module 38
3.3.5. Programming Overview 39
3.4. Peripherals 40
3.4.1. OLED 40
3.5. Robot Mechanical Structure 40
3.6. System Integration 45
3.6.1. Patio 1 45
3.6.2. Patio 2 50

4. Results & Discussion 56
4.1. Arrow Identification 56
4.2. Line Patrol 56
4.2.1. Straight Line Movement 56
4.2.2. Patrolling along the railings 57
4.2.3. Crossing the stone road 57
4.2.4. Turning 58
4.3. Wireless Communication 58
4.3.1. How to start sending messages 58
4.3.2. Clock accuracy 58
5. Conclusion 60
References 61
Appendix 62
A. Gantt Chart 62
B. Programming 63
B.1. Visual Processor 63
B.1.1. Line Detection and Recognition 63
B.1.2. Arrow Detection and Recognition 64
B.2. Ultrasonic Sensors 67
B.2.1. Ultrasonic Sensors Definition 67
B.2.2. Ultrasonic Sensors Header File 68
B.3. Accelerometer and Gyroscope 71
B.3.1. Accelerometer and Gyroscope Definition 71
B.3.2. Accelerometer and Gyroscope Header File 73
B.4. Patio 1 Main Program 73
B.5. Patio 2 Main Program 80
C. Project Management 94

Abbreviation

Abbreviation	Full Name
AC	Alternating Current
ADC	Analog-to-Digital Converter
DMP	Digital Motion Processor
DC	Direct Current
LED	Light-Emitting Diode
LCD	Liquid-Crystal Display
MCU	Micro Controller Unit
NC	Normally Closed
OLED	Organic Light-Emitting Diode
PWM	Pulse Width Modulation
PCB	Printed Circuit Board
PVC	Polyvinyl Chloride
ROI	Region of Interest
TV	Television

List of Figures

Figure 1. Patio 1 tasks. ····2
Figure 2. Patio 2 tasks. ····3
Figure 3. Sensor system. ····4
Figure 4. Executive control and wireless communication system. ····5
Figure 5. The pinout of OpenMV [1]. ····6
Figure 6. Illustration of track extraction methods based on grayscale thresholds. ····7
Figure 7. Sobel kernels. ····7
Figure 8. Illustration of track extraction method based on Sobel Operator. ····8
Figure 9. Simple high pass kernels. ····8
Figure 10. Illustration of track extraction method based on simple high-pass filter. ····9
Figure 11. Illustration of track extraction method based on Canny Operator. ····9
Figure 12. Improved Erosion & Dilation kernels. ····9
Figure 13. Illustration of the Dilation & Erosion method. ····10
Figure 14. Comparison of the mean value and standard deviation methods. ····11
Figure 15. Partially selected ROIs. ····12
Figure 16. Shape and size of the three types of arrows. ····12
Figure 17. Example training set images. ····14
Figure 18. ROI cropping using color patch detection. ····14
Figure 19. Example training set images (binarization). ····15
Figure 20. Example training set images (fixed threshold binarization). ····16
Figure 21. HC-SR04 pins and pin names [2]. ····16
Figure 22. Results of successful measurements with the HC-SR04. ····17
Figure 23. Coordinate system transformation relationships. ····19
Figure 24. MPU6050 [4]. ····20
Figure 25. Nucleo-G474RE pin names and pinouts. ····23
Figure 26. Schematic diagram of the switching power supply circuit. ····25
Figure 27. Schematic diagram of the battery monitoring circuit. ····27
Figure 28. Schematic diagram of the motor drive circuit. ····28
Figure 29. Spatially-averaged motor noise measurements for motor operation at 12 V with different frequency ranges [10]. ····29
Figure 30. Layout design of the main board. ····31
Figure 31. The front and back 3D-rendered view of the main board. ····31
Figure 32. The extension board wiring. ····32
Figure 33. The front and back 3D-rendered view of the extension board. ····32
Figure 34. The front and back real shots of the soldered PCBs. ····33
Figure 35. PCB power-up test. ····33
Figure 36. 3D-printed claw. ····34
Figure 37. SG90 servo. ····34
Figure 38. 3D printed connector. ····35
Figure 39. Slider design drawing. ····35
Figure 40. The overall mechanical arm. ····35

Figure 41. Acrylic panel. ····· 41
Figure 42. Chassis of the robot. ····· 42
Figure 43. The schematic diagram of the structure of the first layer. ····· 42
Figure 44. The frame to hold the OLED and the switch. ····· 43
Figure 45. The schematic diagram of the structure of the second layer. ····· 43
Figure 46. The schematic diagram of the structure of the third layer. ····· 44
Figure 47. Triangular plate for connecting and fixing PVC pipes. ····· 44
Figure 48. The final assembled robot. ····· 45
Figure 49. Patio 1 scene layout and partial action track. ····· 46
Figure 50. Line tracking in phase 1 of patio 1. ····· 46
Figure 51. Ultrasonic recognition of obstacles and 90 degree turns. ····· 47
Figure 52. Traveling forward again for 1.2s and stopped, adjusting the forward direction. ····· 47
Figure 53. Getting on the bridge. ····· 48
Figure 54. Crossing the bridge. ····· 48
Figure 55. Line tracking after getting off the bridge. ····· 49
Figure 56. Passing through the gate. ····· 49
Figure 57. The patio 1 flow diagram. ····· 50
Figure 58. Patio 2 tracked route. ····· 51
Figure 59. Stop for arrow identification. ····· 51
Figure 60. Knock out the bracket at starting point (yellow square). ····· 52
Figure 61.Patrol along railing with ultrasonic sensors. ····· 52
Figure 62. The robot was facing the stone road. ····· 53
Figure 63. Release the ball from claw. ····· 53
Figure 64. The robot identified the beacon ····· 53
Figure 65. An example of message with timestamp ····· 54
Figure 66. The patio 2 flow diagram ····· 55
Figure 67. Example for image binarization. ····· 56
Figure 68. Adjusting the direction while moving forward. ····· 57
Figure 69. Patio 2 – the stone road. ····· 58
Figure 70. Gantt chart. ····· 62

List of Tables

Table 1. Training results for full picture recognition (grayscale). ····· 14
Table 2. Training results for full picture recognition (binary). ····· 15
Table 3. The data format using UART serial communication. ····· 22
Table 4. G474 Pin Definition. ····· 24
Table 5. Overall task division of the project teamwork ····· 94
Table 6. Task division in Stage 1 ····· 95
Table 7. Task division in Stage 2 ····· 95
Table 8. Task division in Stage 3 ····· 96
Table 9. Task division in Stage 4 ····· 96
Table 10. PCB bill of materials. ····· 96
Table 11. The financial record of the team by the end of the project ····· 97

1. Introduction

Person in charge: Luo Jiaqi

1.1. Background

Our college offers a unique collaborative course called Team Design Project and Skills. This chapter covers the teamwork project's goals, discusses the report's structure, and gives background information on intelligent robots.

Intelligent robots originated in the mid-nineteenth century and over the decades they have been refined in terms of function and form and are gradually being used in any area of society. A small robot may contain a lot of interdisciplinary knowledge and is an amalgamation of various technologies. Initially, intelligent robots were designed to perform industrial tasks that were more dangerous or hazardous to human health. For humans, the development of intelligent robots is an important expression of technological development and an inevitable trend. In the future, as various disciplines develop and innovate, the development of intelligent robots will inevitably move in the direction of cross-disciplinary development.

The goal of Team Design Projects and Skills (TDPS) is to develop team cooperation skills via the design and implementation of an integrated electronic system that can carry out certain activities. Ten students from the University of Electronic Science and Technology Glasgow College's EE, CE, and ME majors make up each team, which will work autonomously to finish the design project. The objective of the project is to develop and construct an intelligent robot that can carry out a variety of duties on two distinct courtyards, such as patrolling along a predetermined course, recognizing and responding to various color blocks, carrying and releasing things, and communicating data. At the same time, it's important to follow certain broad guidelines for working as a team, choosing gear, etc.

1.2. Task Analysis

Our design must accomplish two separate sets of tasks on two different patios. The robot should operate smoothly and correctly on both patios after the design job is finished. The next part provides a quick overview of the necessary tasks.

1.2.1. Patio 1

Figure 1 shows the target tasks which need to be accomplished on Patio 1. The designed robot must do the following on this patio:

• Start from the green baseline and follow the path of colored tiles to reach the blue bridge [Task 1];

• Recognize and cross on top of the blue bridge [Task 2];

• Find the purple gate and go through it and then stop at an inexact point at which the robot stops after passing through the gate is not critical. [Task 3].

The gate should be aligned with the strip of colorful tiles at the end of the right-most black line in the image below. The photo's leftmost black line denotes the alignment of the right-side of

the bridge with the left-side of the stairway.

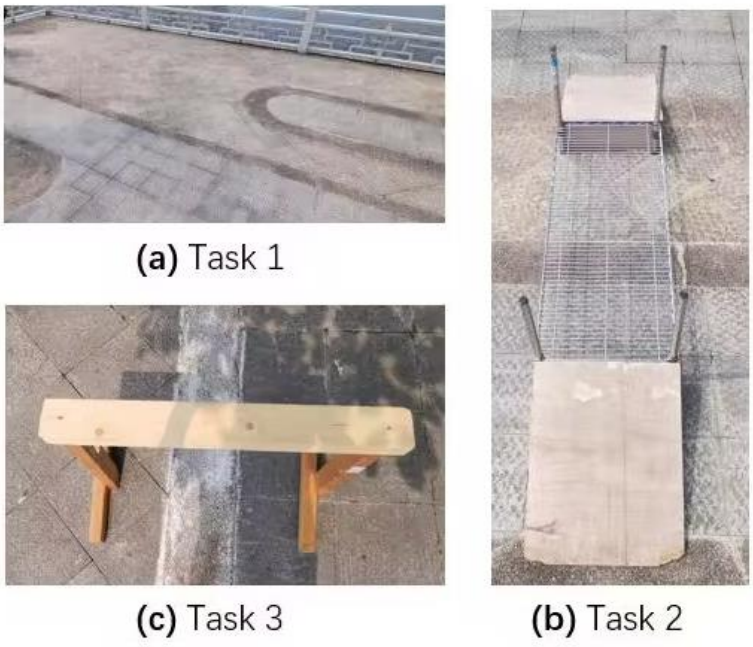


Figure 1. Patio 1 tasks.

1.2.2. Patio 2

Figure 2 shows the target tasks which need to be accomplished on Patio 2. The designed robot must do the following on this patio:

•Start from the green point, scanning and recognizing the shape on the stand when the robot moved to the yellow square in order to determine the direction. The robot will be placed at the edge of the green line as outlined. Before this task begins, a table tennis size ball should be carried on the robot till the second task. Then, it will be released into the basket. The robot should be programmed to move to the square at the right-hand side of the diamond where one of three shapes will be placed. After the shape has been determined, the robot will then move to the corresponding shape in the diamond. There are three situations here:

– A left arrow: The robot should move to the left-most tip of the diamond and knock out the stand with the left arrow shape.

– A straight arrow: The robot should move to the upper tip of the diamond and go over stand with the straight arrow shape.

– A right arrow: The robot should move to the lower tip of the diamond and knock out the stand with the right arrow shape. [Task 1];

• Release the tennis ball to the basket at the release point. The diameter of the basket is 25 cm and will be placed at the front of the lower rail of the fence. [Task 2];

• Transmit a message to a laptop including team name, date, day and time (24-hour clock) in radio signal at 433 MHz when the robot enters the planter area (the top edge of the planter is purple) [Task 3].



(a) Task 2 **(b)** Task 1 **(c)** Task 3

Figure 2. Patio 2 tasks.

1.2.3. General Rules

The robot must run a software that has been previously downloaded to the robot's microcontroller. Instructions cannot be sent to the robot in real time. The path that the robot must travel from the diamond to the basket is not governed by any regulations. Navigation along the side of the planter and pond, or along the fence, or any other method that does not infringe the basic guidelines.

A total of two beacons may be deployed on each patio to send signals for the robot to follow in order to aid it with navigation. So, for patio 1, two beacons can be utilized, or one beacon for patio 1 and one beacon for patio 2, or two beacons for patio 2.

The transition from one task to the next should be programmed (with no outside influence). All electrical systems, as well as all connections to circuit components and subsystems, must be tough and dependable. Between the battery and the remainder of the robot, a fuse and an ON/OFF switch should be installed. The fuse should be sufficiently designed such that it is not destroyed during normal robot operation but creates an open circuit if more current than predicted is pulled from the battery. Each team is asked to develop a motor driver circuit as well as the PCB that will house this circuit. The project's maximum cost is 1000 RMB.

2. Top-level Structure Design

People in charge: Luo Jiaqi, Zeng Yu

2.1. Sensor System

The key to the sensor system is composed of a module that includes a visual processor, ultrasonic sensor, accelerometer, and gyroscope. The main purpose of this module is to receive and process signals from the external environment, and then transmit the processed signals to the control system. In this regard, the visual processor is responsible for shape detection and line tracking. Due to the low color differentiation of the two types of tiles in the field test scene, we chose to use OpenMV4 H7 Plus to identify the boundary line between the two types of tiles to ensure stable driving of the robot for both Patio1 and Patio2. On the other hand, the ultrasonic sensor is used for distance measurement, beacon and fence recognition, and ensuring smooth and accurate movement of the robot. For this purpose, the HC-SR04 ultrasonic sensor was

selected. The sensor can report the range of objects up to 13 feet away, and provide excellent non-contact range detection between 2 cm and 400 cm with an accuracy of 3mm. We installed one and two HC-SR04s on the front and left side of the robot, respectively. In addition, the module containing the accelerometer and gyroscope is equipped with MPU6050, which can control the robot's movement direction. Therefore, the sensor system module is a key component of the entire robot system, enabling it to operate effectively and efficiently.

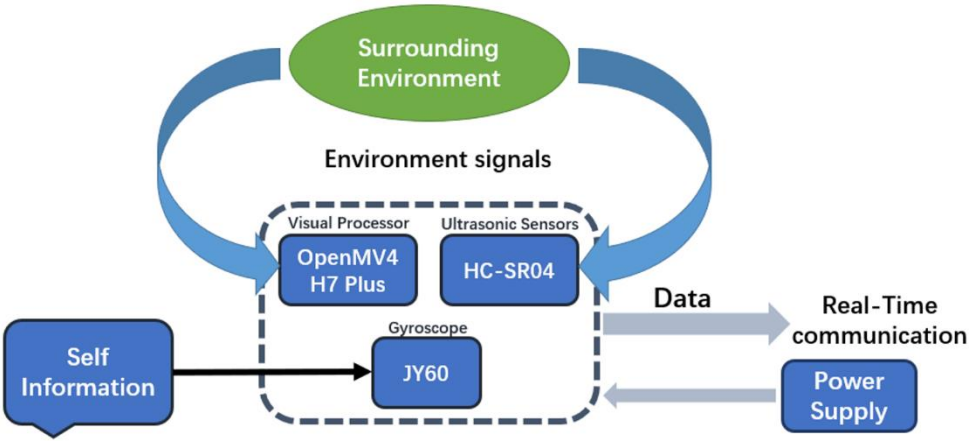


Figure 3. Sensor system.

2.2. Executive Control System

The primary components of the execution control system are a microcontroller, a mechanical arm, and a motor drive circuit. The MCU receives and processes signals from the sensor system to generate control signals, which are then transmitted to the robot's components. To optimize the speed and accuracy of signal processing, we have selected NUCLEO-G474RE, which provides rich peripheral interfaces that could be connected to a variety of sensors and communication modules. The robotic arm, which comprises a position servo actuator and a 3D printed paw, is responsible for holding and releasing the ball. We have opted to use SG90 for the actuator section to control the paw's movements, so that it can rotate up to 180 degrees in both directions with a high degree of accuracy and precision. The motor drive circuit receives control signals from the MCU and drives the motor to execute commands. The circuit uses the TB6612FN motor drive chip for motor control and the MG540 encoder motor for driving the robot. Additionally, the motor drive circuit is responsible for powering the entire system.

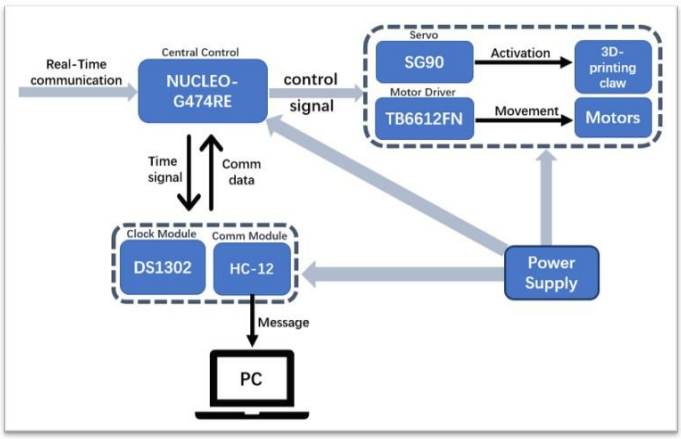


Figure 4. Executive control and wireless communication system.

2.2.1. Wireless Communication System

In the given scenario, the robot is responsible for transmitting a message to a designated laptop as soon as it enters the planter zone. The crucial data contained within this transmission includes the Team Name, Date, day and time, as well as the current time of day displayed in 24-hour clock format. This transmission will be accomplished using a Wavesens HC-12 wireless transceiver, which operates at 433MHz with a fixed data rate. Once the laptop acknowledges receipt of the message, the robot will continue on its pre-determined path until it reaches the endpoint indicated by a red box in the Patio 2 blueprint and a red arrow and line in the accompanying photograph. This procedure guarantees that the robot transmits updates effectively and executes assigned tasks within the specified parameters.

3. Implementation

3.1. Sensor System Implementation: Detection for Patrol and Adjustments

3.1.1. Visual Processor

People in charge: Wang Jiaqi, Li Ruixuan

OpenMV is an open-source, microcontroller-based machine vision platform that is designed for easy integration with a wide range of hardware and software applications [1]. As a recognition module, the OpenMV provides a comprehensive set of features for developing and prototyping embedded vision applications.

The module has an on-board image sensor that can capture images in a variety of formats, including grayscale, RGB565, and YUV422. Its powerful processor that allows for real-time image processing and analysis. The module has a set of programmable GPIO pins, which can be used for a wide range of applications, including controlling actuators and sensors.

The module comes with pre-trained models for object detection, face detection, and color recognition. It supports custom model training using TensorFlow Lite, allowing for the development of custom object detection and recognition models. Besides, the module is

supported by a comprehensive set of software tools and libraries, including the OpenMV IDE, which provides a user-friendly environment for developing and testing machine vision applications. The pinout of OpenMV is shown in Figure 5.

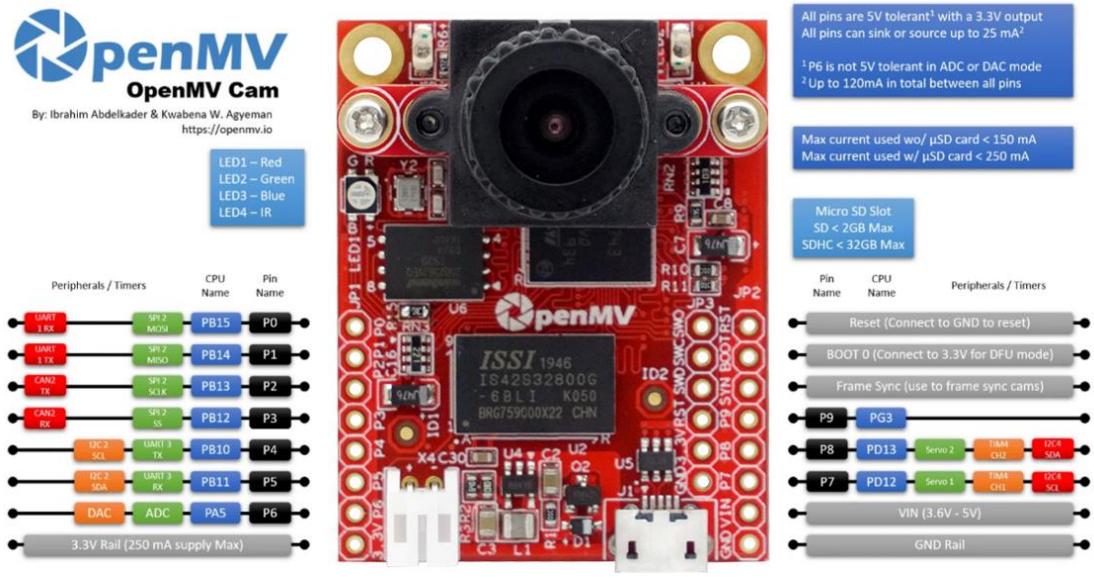


Figure 5. The pinout of OpenMV [1].

In this project, the NucleoG474 is the main control unit and the OpenMV is used as the vision sensor. During the patrol, the PID algorithm and image recognition are executed on the OpenMV, which transmits the data packets from the image recognition and PID algorithm to the G474 using the UART protocol, which includes the speed (corresponding duty cycle) and direction of rotation of the left and right wheels, for a total of four sets of data. The G474 uses the OpenMV data to directly control the two motors, ultimately achieving a patrol effect.

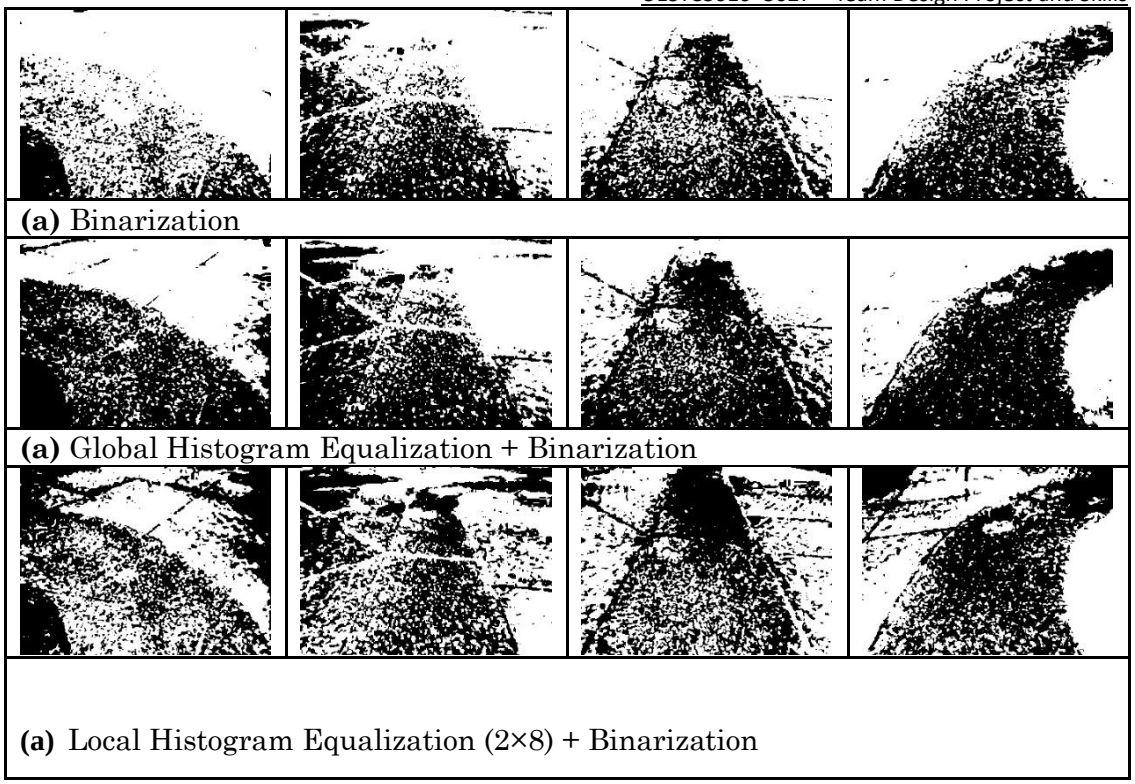
Line Detection and Recognition

(a) Track extraction method based on grayscale thresholds

In the initial attempt, pixel values, or grayscale values, were used as the basis for track extraction. As shown in Figure 6, direct binarization was not a good choice. Considering the enhancement of image contrast, both global and local histogram equalization were used before binarization to enhance trajectory features, as shown in Figure 6, but the results were still not ideal. After analysis, it was found that this was because the color of the track and the side road surface were not fixed, and the color of the track may be darker or lighter than the color of the side road surface.



(a) Original Images



(a) Binarization

(a) Global Histogram Equalization + Binarization

(a) Local Histogram Equalization (2×8) + Binarization

Figure 6. Illustration of track extraction methods based on grayscale thresholds.

(b) Track extraction method based on edge detection algorithms

Considering that the track usually has a different texture from the side road surface, edge detection methods would be a good choice. Two edge detection algorithms were adopted, one is the Sobel operator and the other is the Canny operator. The Sobel method is a simple high-pass filter implemented using two convolution kernels and summing the two resulting images, while the Canny method consists of 5 steps, which are Gaussian filtering, pixel gradient calculation, non-maximum suppression, hysteresis thresholding, and isolated weak edge suppression. The two kernels of Sobel method are shown in Figure 7. The result of Sobel method and the binarized version are shown in Figure 8.

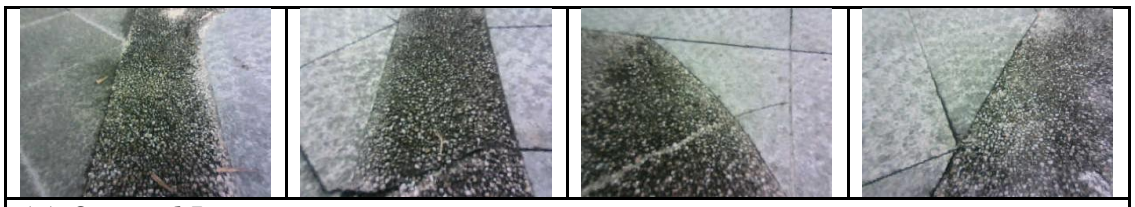
-1	0	1
-2	0	2
-1	0	1

(a) Horizontal

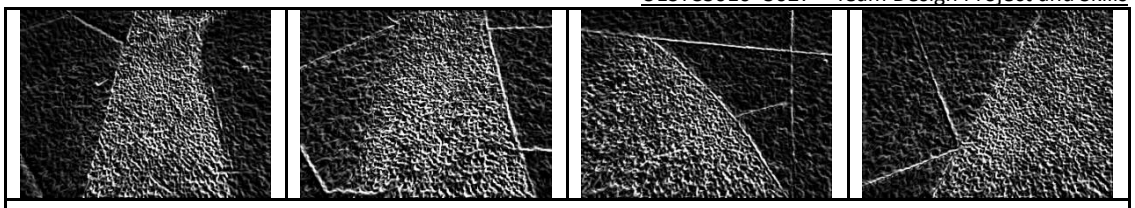
-1	-2	-1
0	0	0
1	2	1

(b) Vertical

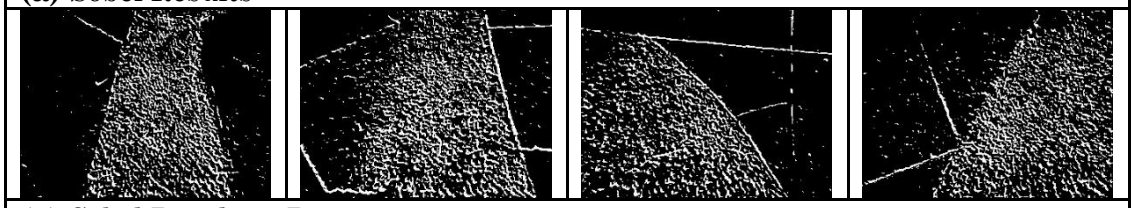
Figure 7. Sobel kernels.



(a) Original Images



(a) Sobel Results



(a) Sobel Results + Binarization

Figure 8. Illustration of track extraction method based on Sobel Operator.

Note that the example images used in this section all are the same as the original image in Figure 8. In addition to the Sobel algorithm, a simple high-pass filtering method was also attempted, with convolution kernels shown in Figure 9. The corresponding results and binary results are shown in Figure 10.

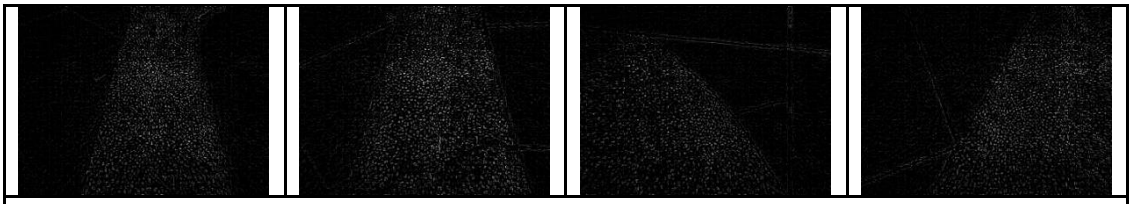
-1	-1	-1
-1	5	-1
-1	-1	-1

(a) Kernel K0

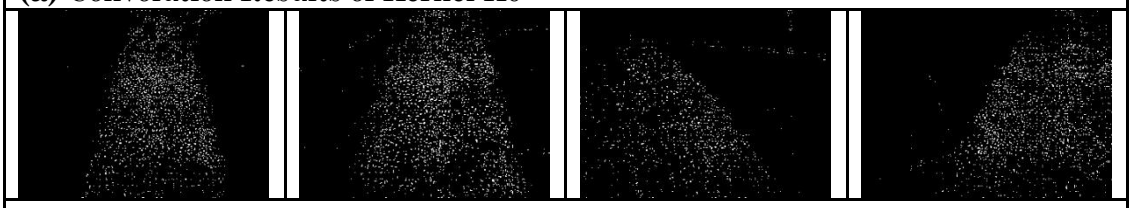
-1	-1	-1
-1	6	-1
-1	-1	-1

(b) Kernel K1

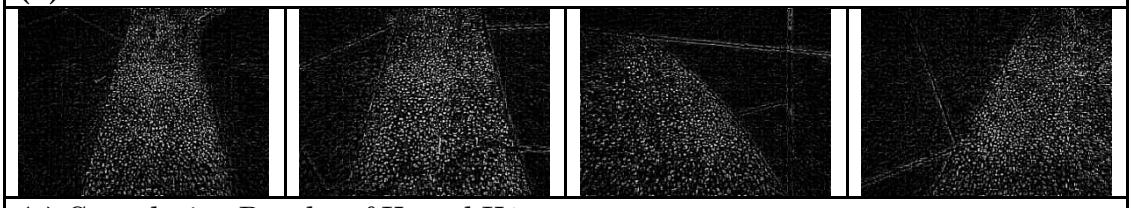
Figure 9. Simple high pass kernels.



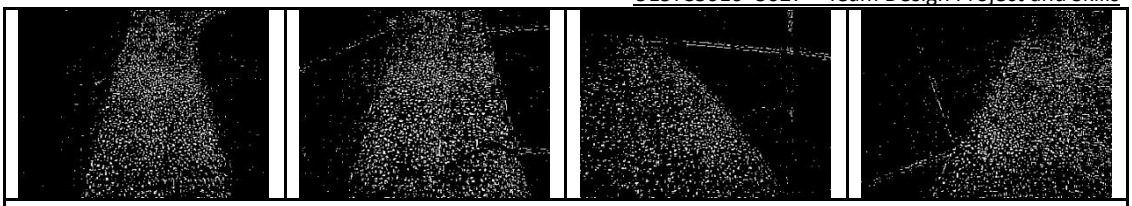
(a) Convolution Results of Kernel K0



(b) Convolution + Binarization Results of Kernel K0



(c) Convolution Results of Kernel K1



(d) Convolution + Binarization Results of Kernel K1

Figure 10. Illustration of track extraction method based on simple high-pass filter.

A Canny operator has two adjustable parameters, and in this study, optimal parameter settings obtained through brute force search were used for comparison with the results of the Sobel algorithm. The results of Canny method are shown in Figure 11.

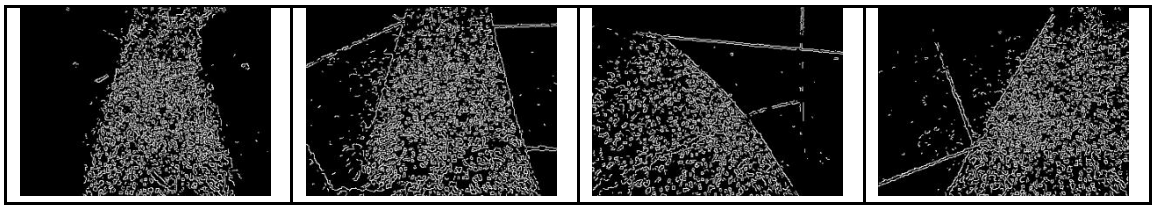


Figure 11. Illustration of track extraction method based on Canny Operator.

After comparing the above three methods, considering the signal strength of the target area and the noise intensity of the background area, the Canny algorithm was ultimately chosen.

(c) Post-processing of Canny images using Dilation & Erosion

Since the initial idea was to use a simple navigation algorithm, further processing was needed for the Canny image. Dilation & Erosion methods were used to eliminate noise and enhance target features. Considering the differences in the distribution of white pixels in the noise and target areas, the erosion and dilation kernels were improved as shown in Figure 12. The results of using the default filled erosion and dilation kernels are shown in Figure 13 (a), and the results of using the improved kernels are shown in Figure 13 (b). The improved erosion and dilation processing had ideal results, but it was not ultimately used due to the later use of the ROI region variance method.

0	0	1	1	1	0	0
0	1	1	1	1	1	0
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
0	1	1	1	1	1	0
0	0	1	1	1	0	0

(a) Erosion Kernel O7

1	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	1	0
0	0	1	0	0	0	1	0	0
0	0	0	1	0	1	0	0	0
0	0	0	0	1	0	0	0	0
0	0	0	1	0	1	0	0	0
0	0	1	0	0	0	1	0	0
0	1	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	1

(b) Dilation Kernel X9

Figure 12. Improved Erosion & Dilation kernels.

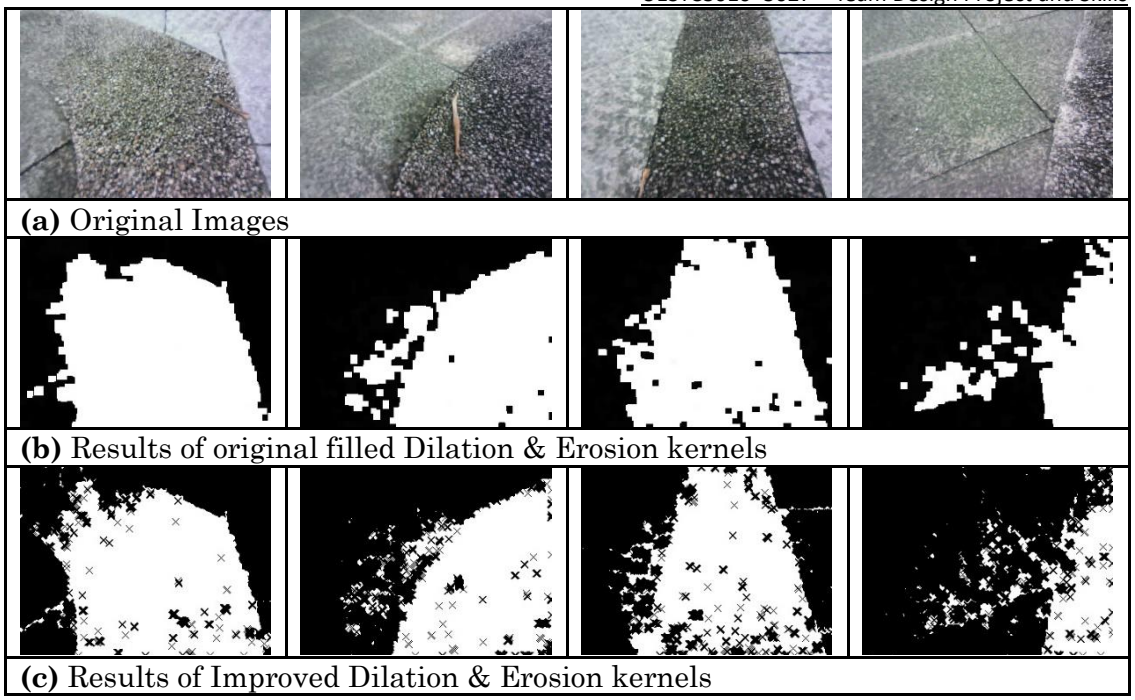


Figure 13. Illustration of the Dilation & Erosion method.

(d) Navigation algorithm with linear regression method

Initially linear regression was used as the navigation algorithm, but it incurs huge computational overhead when dealing with raw size images, resulting in frame rate degradation and vehicle speed slowdown. Therefore, an informative inductive simplification of the image is required to obtain a smaller pixel matrix. This could be achieved by computing statistics such as mean or standard deviation. However, even with the simplification, it is still necessary to ensure that the processed pixel matrix does not negatively affect the navigation accuracy, which also requires a relatively large computational overhead.

When linear regression was tested at the beginning, it was found that the computational overhead of the method was huge when dealing with the original size of the image, resulting in frame rate degradation and slow speeds of the car. Therefore, in order to reduce the computational complexity, an informative inductive reduction of the image is used. Specifically, in order to reduce the calculation time under the premise of ensuring the navigation accuracy, we process the image information to obtain the speed of the two wheels in the following basic steps:

1. Preset parameters: set the camera mode to grayscale and the image size to QVGA (240x320). Each region of Interest (ROI) was defined as a 20x20 pixel region. The lower and upper limits of the pixel values of the Canny algorithm were set to 95 and 135, respectively. The default or reference speed of both wheels is set to 40, and the maximum and minimum speed are set to 70 and 10, respectively.

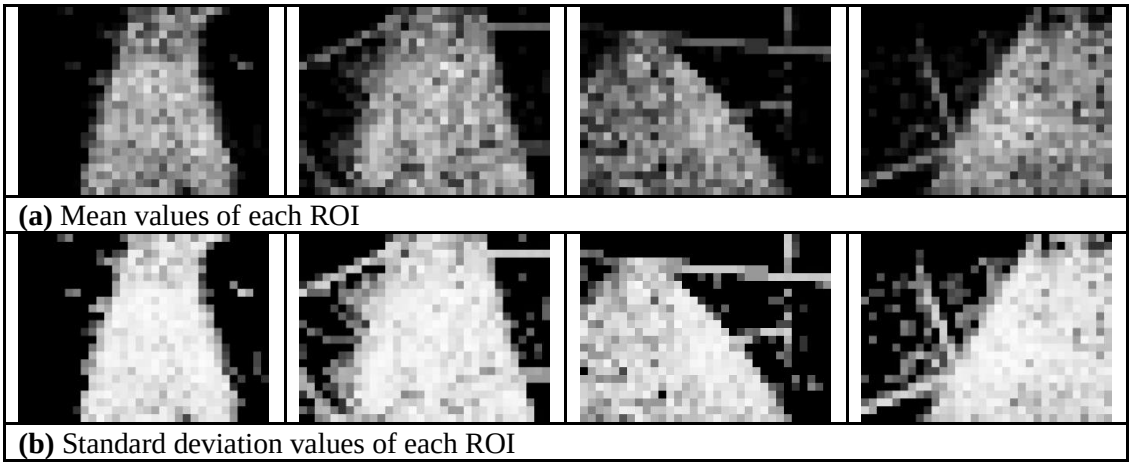
2. Processing a new image: The whole image is processed using the Canny operator to produce a binary image. A row of ROIs is selected from the horizontal mid-line of the binary image. The standard deviation of pixel values was calculated for each ROI to obtain a series of standard deviation values assigned to the coordinate position of

each ROI. The standard deviation value was taken as the mass and the centroid of the ROI of the series was calculated. Determine whether the vehicle needs to turn and, if so, the degree of turn.

3. Filtering mechanism: Since the centroid is easily affected by ROIs that are far from the vertical mid-line, a filtering mechanism is implemented. The first approach is to discard ROIs with standard deviations below 25.5 and compute the centroids of the remaining ROIs. Another approach is to increase the degree of standard deviation to 2 to suppress the weight of low standard deviation regions. Different variations of the two methods were tested to determine the final choice.

4. Speed control on a straight track: a column of ROIs is arranged vertically above the previously obtained centroid. Calculate and accumulate the standard deviation values for this list of ROIs. This sum indicates whether the road ahead is straight and clear. The accumulated values are normalized to obtain a coefficient, which is multiplied with the initial velocities of the two wheels to determine the final velocity.

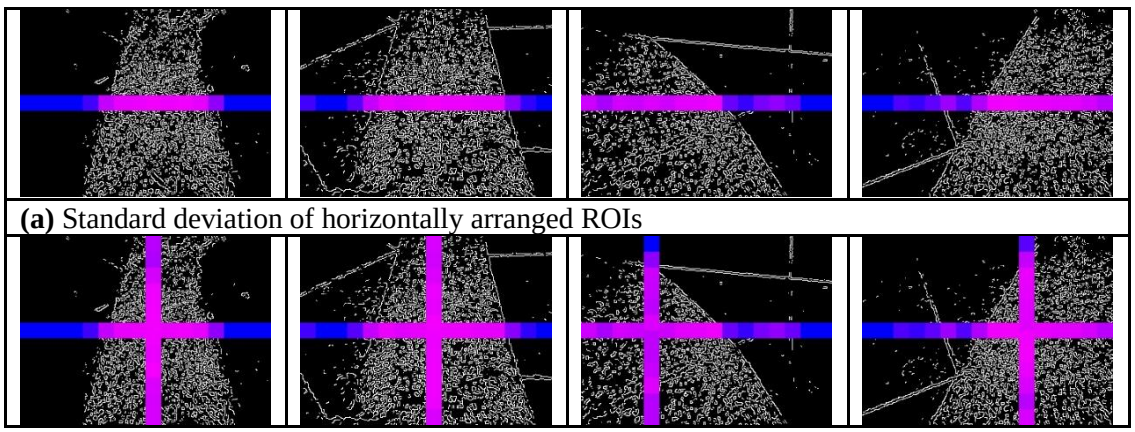
5. Application of PID algorithm: Application of PID algorithm to improve the response speed of the system and enhance anti-jamming ability. It is applied to the two-wheel speed difference and the speed normalization coefficient. The specific values of D (derivative) and I (integral) terms were determined through field tests.



(a) Mean values of each ROI

(b) Standard deviation values of each ROI

Figure 14. Comparison of the mean value and standard deviation methods.



(a) Standard deviation of horizontally arranged ROIs

(b) Standard deviation of horizontally arranged ROIs and vertically arranged ROIs

Figure 15. Partially selected ROIs.

(e) Selection of Algorithms

Regarding the reason for choosing the standard deviation instead of the mean, the proportion of Canny edges in dense areas is around 45%, resulting in a standard deviation close to the maximum value (127.5) and a mean close to half of the maximum value. Both measures can provide good discriminative power. However, the standard deviation is more suitable for regions with fewer edges, since large areas of the image where almost half of the pixels belong to Canny edges are not common in this line tracking task.

At the same time, in the tracking task of the robot, we realized scanning a line of Region of Interest (ROI) in the vertical direction for speed control. For each ROI, the standard deviation of the pixel values is calculated. The standard deviation can provide the degree of variation of pixel values in the ROI, which is used to judge the flatness of the road. The standard deviation values of each ROI were accumulated and normalized to range between 0 and 1. A linear or nonlinear function can be used to map the standard deviation accumulation to a range of wheel speeds, so that the resulting final speed can be used to control the car's vertical movement.

In addition, although the PID control algorithm can also be applied to speed control to improve the response speed and stability of the system, in the end, we did not use the PID control scheme, because other strategies or methods are enough to meet the needs of robot line inspection in the application.

Arrow Detection and Recognition

(a) General Task

For Patio 2 task 1, the objective is to identify an arrow indicating a left, center, or right direction and navigate towards the designated location indicated by the arrow. The arrow, with its shape and size shown in Figure 16, will be positioned on an acrylic standing card to facilitate its recognition.

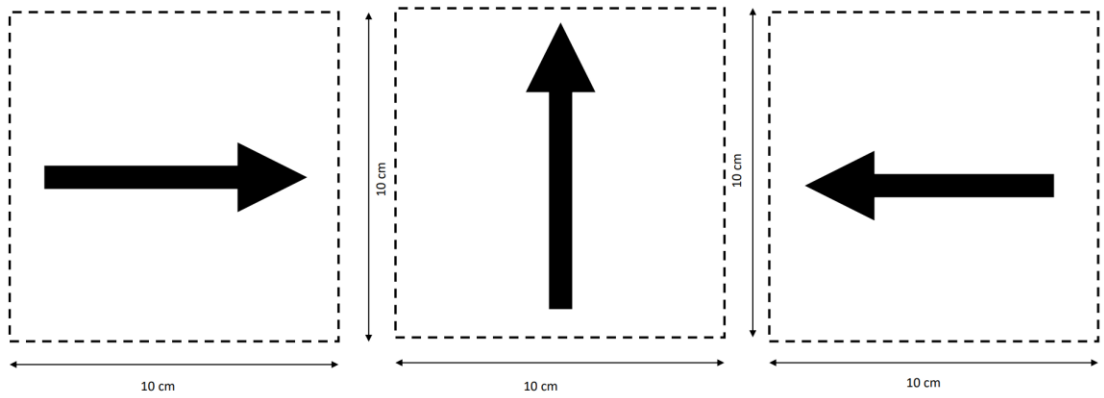


Figure 16. Shape and size of the three types of arrows.

(b) Difficulties and Limitations

Firstly, the camera was not positioned at the correct angle and position directly facing the arrow sign, which may result in some distortion in the shape of the arrow in the image, and the arrow may not strictly point in the horizontal or vertical direction. Secondly, the varying lighting conditions significantly affect the grayscale values of the arrow and the background in the image. Lastly, set the resolution of the OpenMV camera to QVGA (320*240) by default, in practical testing, when the algorithm's time and space complexity both reach $O(n^2)$, the frame rate of the recognition process is likely to drop to one or less, which imposes a high requirement on the stability and accuracy of the recognition process.

Considering the difficulties and limitations mentioned above, a lightweight neural network would be a good choice for completing the arrow recognition task, as it can avoid directly dealing with the complex image and has an acceptable computational load. In this project, the MobileNetV2 model was applied for image classification.

(c) MobileNetV2

MobileNetV2 is a deep convolutional neural network consisting of 53 layers that can categorize images into one of the 1000 object categories. Its architecture is optimized for mobile devices like OpenMV via the utilization of inverted residual blocks, bottleneck layers, and depthwise convolutions. Moreover, the approach results in a substantially low count of parameters.

During field tests, when MobileNetV2 was utilized for image classification, the default input image size was set to 160*160. Under these circumstances, the online running frame rate of OpenMV fluctuates between 0.8 and 1.2. Considering that the neural network is able to provide a stable and accurate recognition result after effective training, this frame rate is acceptable.

(d) Full picture recognition (implementation 1)

In the initial attempt, the whole picture taken by OpenMV was used for arrow identification. In order to reduce unnecessary feature information, grayscale images are used in the training and recognizing process. Some examples are shown in Figure 17. Initially, the entire image captured by the camera was utilized for arrow identification. To reduce unrelated feature information, the training images were first transformed into grayscale images. The labels for the right, top, and left arrows were designated as 1, 2, and 3, respectively. A total of 200 original data sets were collected for each label, and through horizontal flipping, data augmentation resulted in a total of 1200 images. During the training process, a 9:1 training to validation set ratio was utilized, and the trained model's recognition accuracy in field test is taken as the metric for evaluating its effectiveness and robustness.



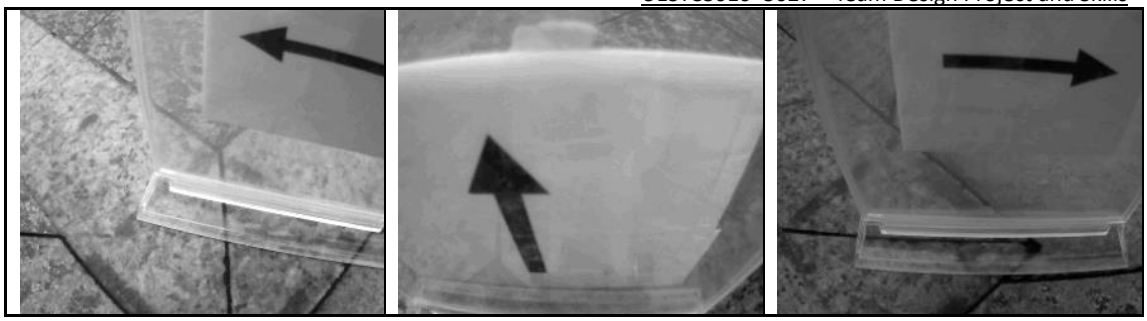


Figure 17. Example training set images.

Through several attempts, five rounds of training were determined to be an optimal balance between overfitting and underfitting. The training results are shown in Table 1. It could be seen that the theoretical performance of the current model should be satisfactory, however, in field tests, it was discovered that the neural network's ability to differentiate between the left and right arrows was significantly influenced by the lighting conditions. Considering the unpredictability of weather conditions on the day of the field demonstration, alternative methods must be implemented to attain a more consistent performance.

Table 1. Training results for full picture recognition (grayscale).

Train loss	Train accuracy	Validation loss	Validation accuracy
0.0082	0.9981	0.0330	0.9916

(e) ROI recognition (implementation 2)

Considering that complex backgrounds may have adverse effects on classification, a pre-cropping method based on white color block detection was introduced, as shown in Figure 18. By pre-setting a range of color close to white, the color block detection algorithm can locate the white region where the arrow resides and simplify the image information by cropping the concerned region. After the dataset was re-collected and the network model was re-trained, the validation set results indicated that the classification performance is still good. However, during field tests, this method exhibited severe vulnerability to changes in lighting conditions, and thus cannot be adopted as the final solution.

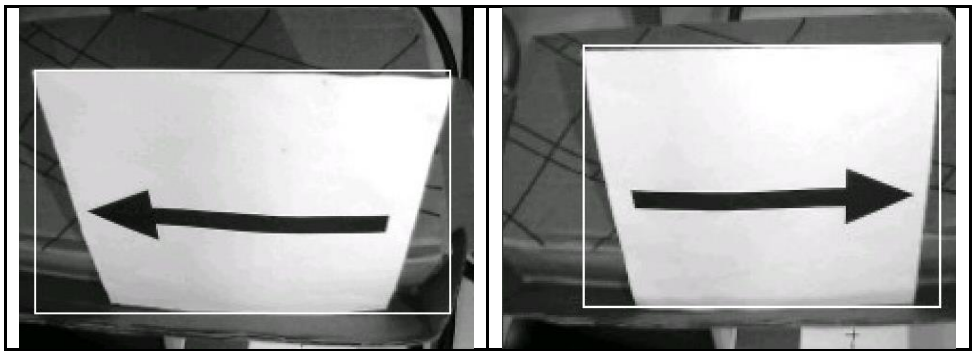


Figure 18. ROI cropping using color patch detection.

(f) Binary recognition (implementation 3)

As the ROI strategy failed to achieve the desired results, alternative techniques had to be

employed in order to further simplify the image features overall. Hence, the average gray level of every image was used as the threshold for a binarization process. The binary images resulting from the examples in Figure 17 are shown in Figure 18. The parameter settings during training remain the same as in grayscale image recognition.

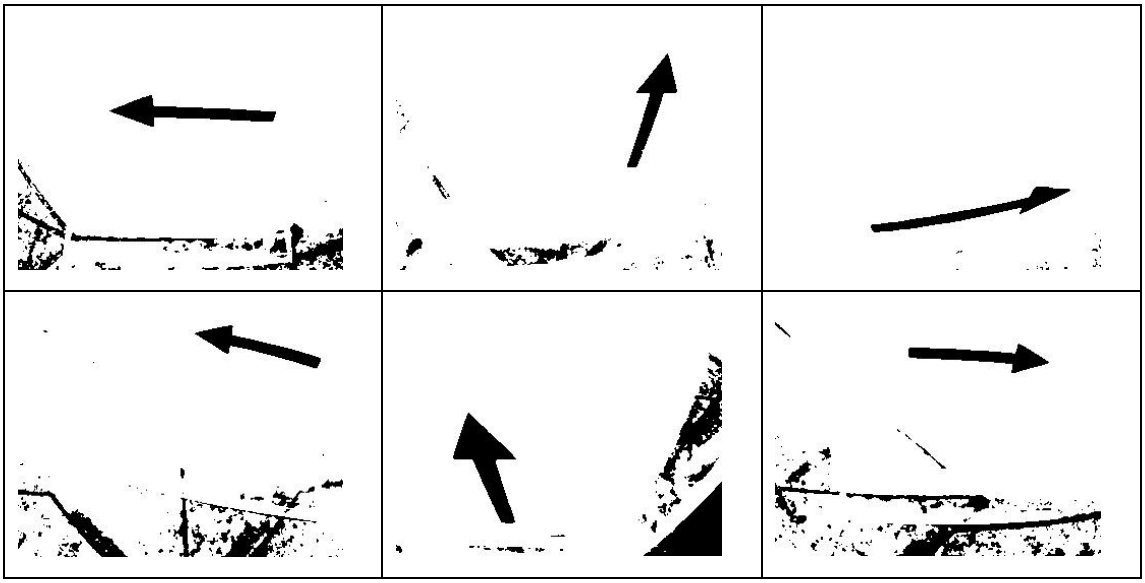


Figure 19. Example training set images (binarization).

Same as in part 1, five rounds of t raining were used. The training results are shown in Table 2. The theoretical performance of the model for binary images was also satisfactory as the model for grayscale images. In field tests, the performance was significantly improved and variations in lighting conditions no longer have a substantial impact on the recognition accuracy.

Table 2. Training results for full picture recognition (binary).

Train loss	Train accuracy	Validation loss	Validation accuracy
0.0098	0.9954	0.0107	1.0000

It should be noted that similar to the training set processing, a varying binary threshold is also used in the recognition process, to be specific, this variable threshold was set to be the average gray level of each image. To illustrate the necessity of this, a fixed threshold of 127 were applied, the binarization results of the examples in Figure 17 are shown in Figure 19. It could be seen that for some images the arrow feature decayed, and in severe conditions, the entire image could be even completely covered.



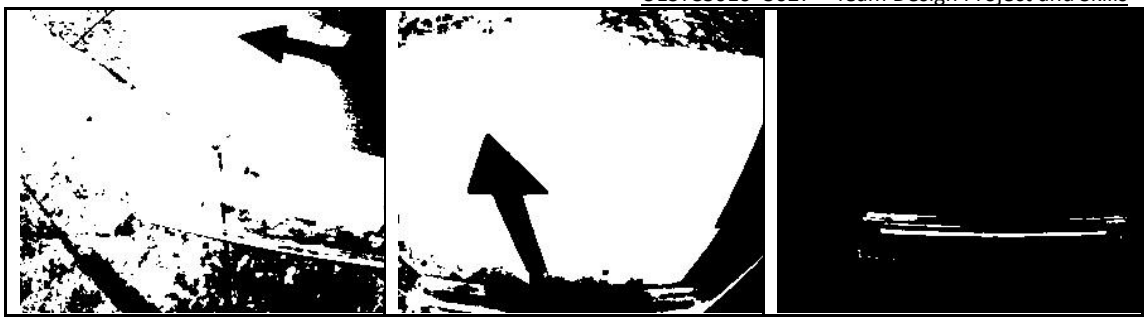


Figure 20. Example training set images (fixed threshold binarization).

3.1.2. Ultrasonic Ranging Sensor

People in charge: Wang Ruiqi, Zeng Yu

This project uses several ultrasonic distance sensors to measure the distance between the robot and the obstacles, intending to help the robot to decide the moment to turn and the moment to stop. We have used the HC-SR04 modules which are popular sensors used in robotics and electronic projects [2]. The HC-SR04 module was tested using a Nucleo-G474RE (a Mbed microcontroller) [3]. One of the three HC-SR04 modules was installed at the front of the robot and the others were installed on the right side of the robot. The front one was used to measure the distance between the robot and the obstacle ahead and the side two were used to measure the lateral distance.

The HC-SR04 pins and pin names are shown in Figure 21.



Figure 21. HC-SR04 pins and pin names [2].

To debug the module, A high-level signal of at least 10us was input to the Trig pin and the module would automatically send eight 40khz square waves and detect whether there was a signal. If a signal was returned, a high level would be output through the Echo pin. The duration of the high level was the time between the ultrasonic wave being emitted and the return. The test distance D could be expressed by the following equation:

$$D = \frac{T \cdot v}{2}$$

Where T represents the high-level time and v represents the speed of sound (340m/s).

The results of successful measurements with the HC-SR04 are shown in Figure 22.

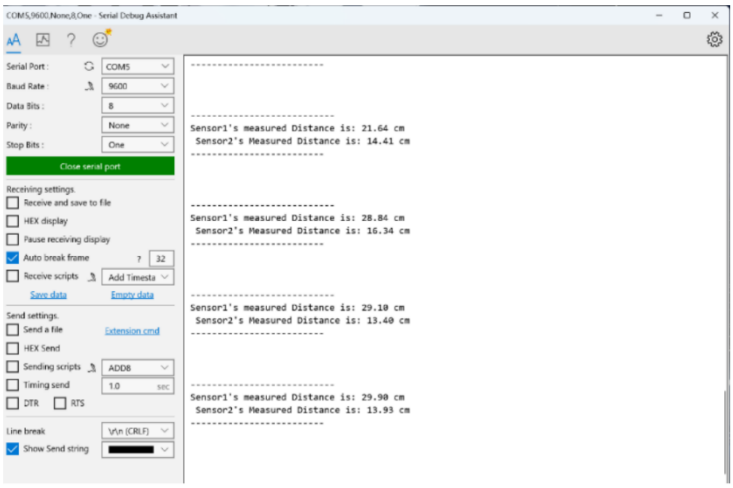


Figure 22. Results of successful measurements with the HC-SR04.

3.1.3. Accelerometer and Gyroscope

People in charge: Wang Ruiqi, Zeng Yu

Brief Introduction

The MPU6050 is a very popular 6-axis motion sensor (3-axis accelerometer + 3-axis gyroscope). It measures the motion and directional changes of mechanical devices.

Accelerometers are, in essence, force sensors. It is used to check how much force (including gravity) is applied to different planes (up and down, left and right, front and back) and then calculate the angle. A gyroscope is, in essence, an angular velocity detector. For example, if an object, with the X axis as its axis, turns to 45 degrees in one second, its angular velocity on the X axis is 45 degrees per second. The unit of angular velocity is degree per second (DPS) which reflects the speed of rotation. As for the resolution, three 16-bit ADCs are used for each of the 3-axis acceleration and 3-axis gyroscope, corresponding to one 16-bit ADC for each axis.

The MPU6050 is also expandable: A second I^2C interface is included for the connection of external magnetic sensors and for the output of the complete 9-axis fusion algorithm to the application via the main I^2C interface using its own DMP (Digital Motion Processor) hardware acceleration engine.

Attitude solving: The DMP from InvenSense makes it very easy to implement attitude solving and reduces the load on the operating system for motion processing operations.

Initialisation Operations

1. Initialising the I^2C interface and reseting the MPU6050 so that all its internal registers return to their default values.
2. Setting the full range of the angular velocity sensor (gyroscope) and the acceleration sensor. According to the above, a 16-bit ADC is used for each axis, so the data that can be output for each axis is also 2^{16} i.e. the corresponding range is -32768 to +32768. So how many degrees/sec does a gyroscope's 32768 refer to in terms of angular velocity,

for example? This is actually determined by the range set by the MPU6050, if the range is different and the 32768 will represent a different value. In terms of gyroscopes, the MPU6050 has four ranges to choose from: ±250, ±500, ±1000, ±2000 degrees/s. For instance, If ±250 degrees/sec is selected, then -32768 to +32768 stands for -250 to +250.

3. Setting other parameters.
4. Configuring the system clock source and enabling the angular velocity sensor and acceleration sensor.

DMP (Digital Motion Processor)

The DMP converts the raw angular velocity data from the acceleration and angular velocity sensors into a 4-tuple of data to complete the calculation of the Euler angles.

The quaternion output from the MPU6050's DMP is in q30 format, which means that the floating point number is magnified by a factor of 2^{30}. Before converting to Euler angles, it must be converted to floating point, i.e. divided by the 2^{30}, and then calculated using the formula:

```
q0 = quat[0] / q30; //q30 format to floating point

q1 = quat[1] / q30;

q2 = quat[2] / q30;

q3 = quat[3] / q30;

// calculate the Euler angle: pitch/roll/yaw angle
pitch = asin(-2 * q1 * q3 + 2 * q0* q2)* 57.3; // pitch angle
roll = atan2(2 * q2 * q3 + 2 * q0 * q1, -2 * q1 * q1 - 2 * q2* q2 + 1)* 57.3; //
traverse roll angle
yaw = atan2(2 * (q1 * q2 + q0 * q3),q0 * q0 + q1 * q1 - q2 * q2 - q3 * q3) * 57.3;
// yaw angle
```

Where quat[0] to quat[3] are quaternions, here the radians are converted to angles (57.3 degrees = 180/π) so that the result is obtained in degrees.

MPU6050 quaternion solving attitude algorithm

The rotation around the Zn axis is used as an example to derive how the coordinates are transformed during the rotation in space:

The projection of the space vector r in the $O - X_1Y_1Z_1$ (or coordinate system 1) is $[r_{x1}\ r_{y1}\ r_{z1}]^T$ and the projection in the $O - X_2Y_2Z_2$ (or coordinate system 2) is $[r_{x2}\ r_{y2}\ r_{z2}]^T$, and the relationship between the two sets of coordinate system values is to be deduced. Since the axis of rotation proceeds around the OZ_1 axis. Therefore, the Z coordinate remains unchanged, and $r_{z1} = r_{z2}$. From Figure 23:

$$r_{x2} = OA + AB + BC$$
$$= \mathrm{OD}\cos\mathrm{a} + \mathrm{BD}\sin\mathrm{a} + \mathrm{BF}\sin\mathrm{a}$$
$$= \mathrm{r_{x1}cosa} + \mathrm{r_{y1}sina}$$

$$r_{y2} = \mathrm{DE} - \mathrm{AD}$$
$$= \mathrm{DF}\cos\mathrm{a} - \mathrm{OD}\sin\mathrm{a}$$
$$= \mathrm{r_{y1}}\cos\mathrm{a} - \mathrm{OD}\sin\mathrm{a}$$
$$= \mathrm{r_{y1}}\cos\mathrm{a} - \mathrm{r_{x1}}\sin\mathrm{a}$$
$$r_{z2} = r_{z1}$$

Organised into matrix form:

$$\begin{bmatrix} r_{x2} \\ \mathrm{r_{y1}} \\ r_{z1} \end{bmatrix} = \begin{bmatrix} \cos\mathrm{a} & \sin\mathrm{a} & 0 \\ -\sin\mathrm{a} & \cos\mathrm{a} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{x1} \\ \mathrm{r_{y1}} \\ r_{z1} \end{bmatrix}$$

Therefore:

$$r_1 = \begin{bmatrix} r_{x1} \\ \mathrm{r_{y1}} \\ r_{z1} \end{bmatrix}$$

$$r_2 = \begin{bmatrix} r_{x2} \\ \mathrm{r_{y2}} \\ r_{z2} \end{bmatrix}$$

Rotation matrix:

$$C_1^2 = \begin{bmatrix} \cos\mathrm{a} & \sin\mathrm{a} & 0 \\ -\sin\mathrm{a} & \cos\mathrm{a} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Therefore, C_1^2 consisting of the Euler angles a can describe this rotation.

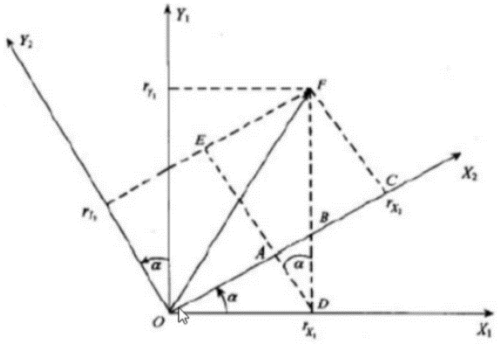


Figure 23. Coordinate system transformation relationships.

Similarly the rotation matrix for the remaining two axis directions can be introduced. These three rotation matrices are multiplied and thus combined into one rotation matrix:

$$C_n^b = C_2^b C_1^2 C_n^1 = \begin{bmatrix} \cos\gamma & 0 & -\sin\gamma \\ 0 & 1 & 0 \\ \sin\gamma & 0 & \cos\gamma \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
$$= \begin{bmatrix} \cos\gamma\cos\psi + \sin\gamma\sin\psi\sin\theta & -\cos\gamma\sin\psi + \sin\gamma\cos\psi\sin\theta & -\sin\gamma\cos\theta \\ \sin\psi\cos\theta & \cos\psi\cos\theta & \sin\theta \\ \sin\gamma\cos\psi - \cos\gamma\sin\psi\sin\theta & -\sin\gamma\sin\psi - \cos\gamma\cos\psi\sin\theta & \cos\gamma\cos\theta \end{bmatrix}$$

This C_n^b matrix is then the direction cosine matrix (also called the rotation matrix). Assuming the existence of a b-system and an n-system, this matrix also describes the rotation of the b-system with respect to the n-system. If we know the coordinates of any vector in the b-system, then we can use the direction cosine matrix to determine its coordinates in the n-system, and

vice versa.

The directional cosine matrix is a combination of the sines and cosines of the Euler angles θ, γ, ψ and the postures we need are these three angles, which we can solve by simply applying the differential equations for the Euler angles:

$$\begin{bmatrix}\dot{\gamma}\\ \dot{\theta}\\ \dot{\psi}\end{bmatrix}=\frac{1}{\cos\theta}\begin{bmatrix}\cos\theta & \sin\gamma\sin\theta & \cos\gamma\sin\theta\\ 0 & \cos\theta\cos\gamma & -\sin\gamma\cos\theta\\ 0 & \sin\gamma & \cos\gamma\cos\theta\end{bmatrix}^{-1}\cdot\begin{bmatrix}\omega_{E\Delta\Delta}^{b}\\ \omega_{Eb\bar{z}}^{b}\\ \omega_{E\Delta z}^{b}\end{bmatrix}$$

On the left-hand side of the above equation are the Euler angles after this update, corresponding to row, pitch and yaw. the three angles on the right-hand side are measured in the previous cycle, while ω represents the angular velocity. The three angular velocities are equal to the angle of rotation of the three-axis gyroscope during this cycle, in radians. By solving this differential equation the current Euler angle can be solved.

Why not use Euler angles to represent rotations instead of introducing quaternions? The reasons are listed below:

1. The Euler angular differential equation contains a large number of trigonometric operations, which makes real-time solving difficult.
2. The magic "GimbalLock" occurs when the pitch angle is $\mathbf{90^o}$. Therefore, the Euler angle method is only applicable when there is little change in horizontal attitude.
3. The quadratic method solves a system of linear differential equations with only four unknown quantities, which is computationally small and easy to operate.

Rotation in three dimensions can be described in terms of unit quaternions. Firstly, a quaternion is defined:

$$\mathbf{q}=a+\vec{u}=q_0+q_1i+q_2j+q_3k$$

The directional cosine matrix described by the Euler angles is described by this newly defined quaternion as:

$$C_n^b=\begin{bmatrix}q_1^2+q_0^2-q_3^2-q_2^2 & 2(q_1q_2-q_0q_3) & 2(q_1q_3+q_0q_2)\\ 2(q_1q_2+q_0q_3) & q_2^2-q_3^2+q_0^2-q_1^2 & 2(q_2q_3-q_0q_1)\\ 2(q_1q_3-q_0q_2) & 2(q_2q_3+q_0q_1) & q_3^2-q_2^2-q_1^2+q_0^2\end{bmatrix}$$

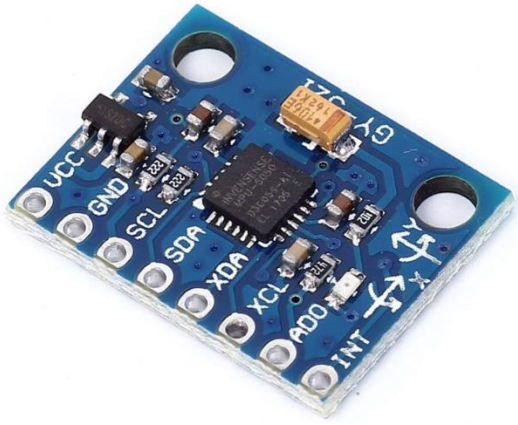


Figure 24. MPU6050 [4].

To recap the whole step:

```
// gx, gy, gz, ax, ay, az correspond to gyroscope x, y, z values and accelerometer x, y, z values respectively

void IMUupdate(float gx, float gy, float gz, float ax, float ay, float az)
{ float norm;
float vx, vy, vz; float ex, ey, ez;

// Converting the values collected by the accelerometer (3D vectors) into unit vectors
norm = sqrt(ax*ax + ay*ay + az*az);
ax = ax / norm;
ay = ay / norm;
az = az / norm;

// The quaternions are converted below into the three elements of the third row in the directional cosine. vx, vy, vz are also the unit vectors of gravity converted from the reference system of body coordinates of the Euler angles (quaternions) from the last time.
vx = 2*(q1*q3 - q0*q2);
vy = 2*(q0*q1 + q2*q3);
vz = q0*q0 - q1*q1 - q2*q2 + q3*q3;

// ax, ay, az are the measured gravity vectors and vx, vy, vz are the gyro-integrated attitudes to derive the gravity vectors, which are the gravity vectors on the body's coordinate reference system. Then the error vector between them, which is the error between the gyro-integrated attitude and the attitude measured by the adder, can be expressed as a vector fork product.
exInt = exInt + ex*Ki;
eyInt = eyInt + ey*Ki;
ezInt = ezInt + ez*Ki;

// adjusted gyroscope measurements
gx = gx + Kp*ex + exInt;
gy = gy + Kp*ey + eyInt;
gz = gz + Kp*ez + ezInt;

// The first order Longo Kuta was used to solve the quadratic differential equation, where T is the measurement period and g the gyroscope angular velocity:
q0 = q0 + (-q1*gx - q2*gy - q3*gz)*halfT;
q1 = q1 + (q0*gx + q2*gz - q3*gy)*halfT;
q2 = q2 + (q0*gy - q1*gz + q3*gx)*halfT;
q3 = q3 + (q0*gz + q1*gy - q2*gx)*halfT;

```

```
34. // Finally, the quaternion is converted into an Eulerian angle according to the
conversion relationship between the quaternion direction cosine array and the Eulerian
angle:
35. ANGLE.Yaw = atan2(2 * q1 * q2 + 2 * q0 * q3, -2 * q2*q2 - 2 * q3* q3 + 1)* 57.3; //
yaw
36. ANGLE.Pitch = asin(-2 * q1 * q3 + 2 * q0* q2)* 57.3; // pitch
37. ANGLE.Roll = atan2(2 * q2 * q3 + 2 * q0 * q1, -2 * q1 * q1 - 2 * q2* q2 + 1)* 57.3;
// roll
38. }
```

JY60

The JY-60 is a common inertial measurement unit (IMU) with an integrated MPU6050 chip and Kalman filter. The JY-60 communicates with the microcontroller via a standard UART interface and transmits the angles directly to the microcontroller. 2. The JY601 offers improved immunity to interference, lower noise and more stable measurement performance. The MPU6050 requires digital filtering to be implemented in the program. 3. The JY601 has a more detailed data sheet with clearer data register addresses for each axis, making it easier to read directly. This is why we finally chose to use the JY60.

Practical project application

The MPU6050 is used in our robots when the robot needs to turn at a specified angle. For example, in Patio 1 the robot needs to turn 90° to perform a task on the bridge after completing its patrol.2 The application of the MPU6050 is not limited to quarter-turns. With the help of the MPU6050, our robot can theoretically turn at any specified angle, which makes our robot more flexible and adaptable to different scenarios.

There are two difficulties in implementing this part: once the attitude determination and control algorithm of the MPU6050 has been implemented, the second difficulty is the communication between this module and the MCU.

The difficult part is the Data Unpacking between JY60 and MCU. JY60 uses WT61 protocol to transmit its data. The WT61 protocol is a serial communication protocol used for transmitting data from inertial measurement units (IMUs). The WT61 refers to a type of IMU module that outputs sensor data such as accelerometers, gyroscopes, and magnetometers using the WT61 protocol. The protocol uses UART serial communication to transfer data. The data format is shown in Table 3. The start byte and end byte are both fixed values of 0x55 and 0xAA, respectively. The flag byte of 0x53 indicates that the data packet is a sensor data packet. The data length varies depending on the specific implementation and the data field contains the sensor data. The checksum field is calculated by summing all bytes except the start and end bytes, taking the one's complement, and adding 1.

Table 3. The data format using UART serial communication.

Start	Flag	Data	Data	Checksu	End Byte

Byte		Length		m	
0x55	0x53	0xXX	Data	Check	0xAA

Corresponding code should be written to perform data unpacking on the data gain from the JY60 to make the data programmable for latter experiment.

3.2. Executive Control System Implementation: Robot Control and Operation

3.2.1. MCU

People in charge: Li Ruixuan

The Nucleo-G474RE module is a development board based on the STM32G474RE microcontroller, which is a member of the STM32G4 series of 32-bit ARM Cortex-M4 microcontrollers [3]. As a main controller module, it provides a comprehensive set of features for developing and prototyping embedded applications. It also has a rich set of peripherals, including multiple UART, SPI, and I2C interfaces, USB, CAN, and ADC.

Nucleo-G474RE pin names and pinouts are shown in Figure 25.

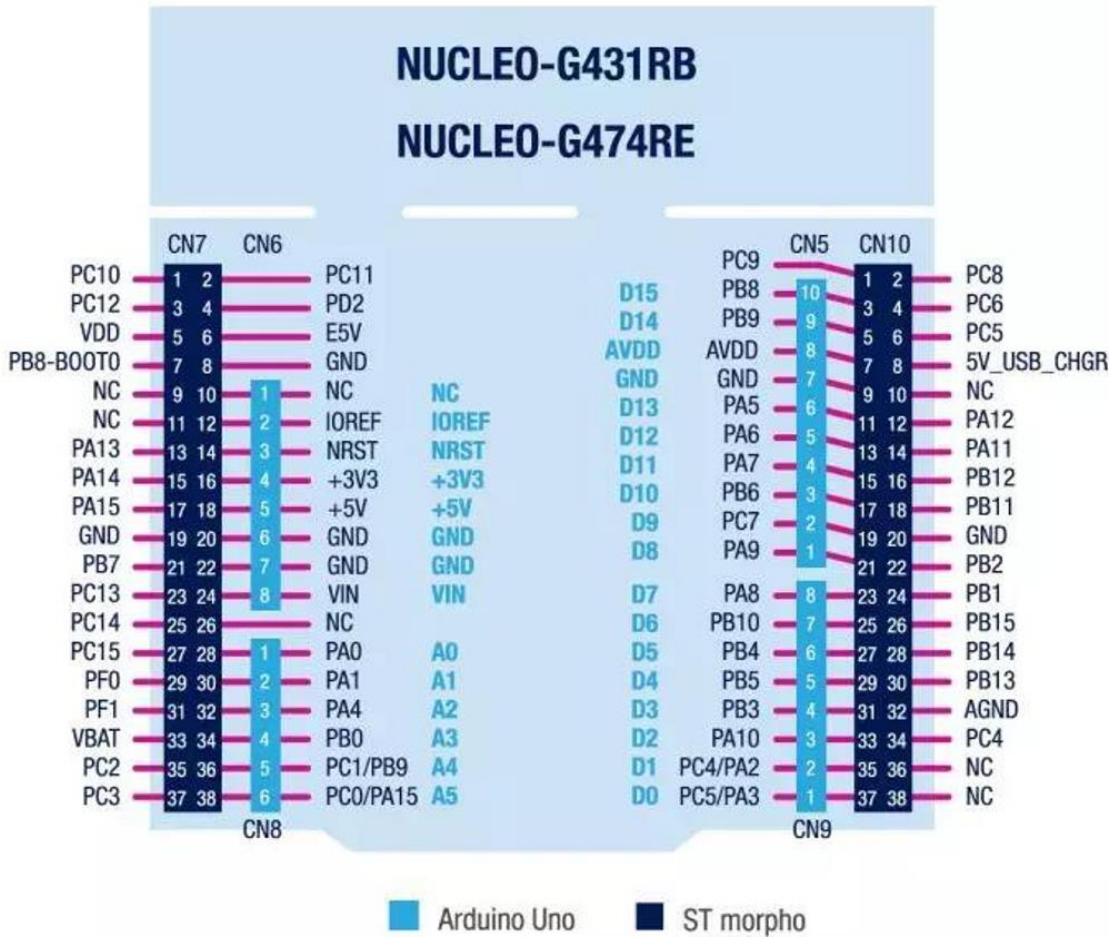


Figure 25. Nucleo-G474RE pin names and pinouts.

Table 4. G474 Pin Definition.

Pin	Definition
D1	BufferedSerial HC (Serial communication with OpenMV)
D2	DigitalOut in1
D3	DigitalOut in2
D4	DigitalOut in3
D5	DigitalOut in4
D6	PwmOut left
D7	PwmOut right
A0	PwmOut Claw
D8	HCSR04 sensor1 (front supersonic sensor1 trig)
D9	HCSR04 sensor1 (front supersonic sensor1 echo)
D10	HCSR04 sensor2 (left supersonic sensor 2 trig)
D11	HCSR04 sensor2 (left supersonic sensor 2 echo)
D12	HCSR04 sensor3 (left supersonic sensor 3 trig)
D13	HCSR04 sensor3 (left supersonic sensor 3 echo)

3.2.2. Power Regulator Circuit

Person in charge: Liu Jingrui

A 12V rechargeable lithium battery pack provides the power supply for all hardware circuits in this design. However, as the two processors and some other modules require a 5V or 3V3 power supply, a circuit for the power supply is vital. Our team has proposed two plans for the power regulator circuit and compared them with each other:

Plan A: Use TI's LM2596S DC-DC regulator to convert 12V to 5V and obtain 3.3V through LDO-LP2992. LM2596S is a monolithic IC that provides all the active functions for a step-down (buck) switching regulator, capable of driving a 3A load with excellent line and load regulation. However, the LM2596S is an obsolete chip with a low level of integration and a large footprint and requires relatively more complex peripheral circuitry. Also, LP2992 only has a maximum output current of 250mA, which may only partially fulfil the power requirement of the system.

Plan B: Use Silergy's synchronous step-down DC-DC regulator SY8368 to convert 12V to 5V and obtain 3V3 through LDO-ME6211. SY8368 could provide a highly stable and efficient 5V output, while its integrated multiple battery protection functions could protect the system. It is a new chip with a high level of integration and a tiny footprint, significantly reducing the complexity of the peripheral circuits. The ME6211 performs with high PSRR, low noise, low ripple, and low dropout 3v3 output power supply for digital voltage. It can also provide an accurate reference voltage for ADC.

Finally, our team decided to apply plan B since SY8368 [5] is a relatively advanced chip with high integration, a small packaging size, and only requires a simple peripheral circuit. With well-designed peripheral components and PCB layout, it can achieve good efficiency with less noise and higher stability.

The schematic diagram of the switching power supply circuit is shown in Figure 26. As shown in Figure 26, a fuse and a switch are also considered for system safety considerations. A 3A-break fuse protects the overall system, and two 1A-break fuses are specially designed to

protect OpenMV and G474. It should be noted that the switch and power cable has been specially selected to carry 3A current. Additionally, a blue LED serves as an indicator. If the 5V output functions well, the blue LED will turn on.

The spherical circuit was designed according to the datasheet. Especially the input and output capacitors should be designed carefully. According to the instruction, C7 is a ceramic capacitor to decouple pin IN to GND with at least 10uF. Decoupling capacitors and bypass capacitors are used in DC-DC buck converter circuits to filter out the noise and provide stable voltage to the load. C1, C2, C3, C4, C5, and C9 help to reduce the input and output voltage ripple caused by the switching action of the buck converter.

The values and types of the capacitors were determined according to experience design rules and the recommended value by the datasheet of SY8368 and ME6211 [6]. Different values can be combined in parallel to fit our operation frequency range. Therefore, the output capacitor is a parallel connection of two $22\mu F$ ceramic capacitors for high-frequency filtering. The rest of the components are calculated below.

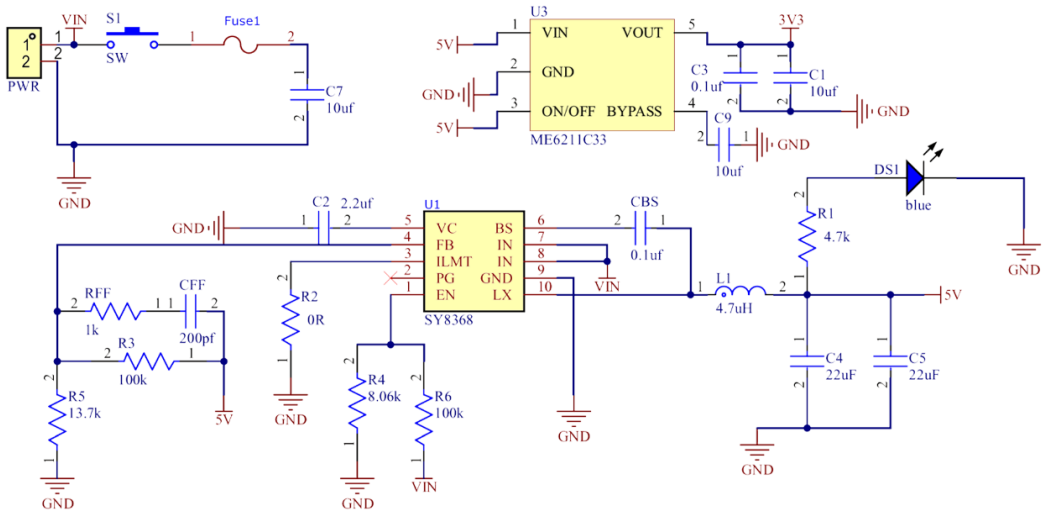


Figure 26. Schematic diagram of the switching power supply circuit.

The parameters of the devices in Figure 26 are calculated as follows:

(1) C_1

C_{in} is the C_1 in Figure 26. $fsw = 800kHz, I_{out-max} = 3A$. Assume that $V_{in-ripple} = 400mV, ESR_{max} = 100m\Omega$.

$$\because V_{in-ripple} = \frac{0.25I_{out-max}}{C_{in} \times fsw} + I_{out-max} \times ESR_{max}$$

$$\therefore C_{in} = 9.375\mu F \approx 10\mu F$$

(2) V_{out}

$$R_5 = R_3 \times \frac{V_{ref}}{V_{out} - V_{ref}} \Rightarrow V_{out} = V_{ref} \times (\frac{R_3}{R_5} + 1)$$

$$\because V_{ref} = 0.6V, V_{out} = 5V, \therefore set\ R_3 = 100k\Omega, R_5 = 13.7k\Omega$$

The value of V_{out} in reality, is equal to 4.98V.

(3) EN pin

$$V_{start} = 3.6 \times 3 = 10.8V, V_{stop} = 3.3 \times 3 = 9.9V$$

$$V_{ENH} = 0.8V, V_{ENL} = 0.4V$$

$$\because \frac{V_{in} - V_{EN}}{R_6} = \frac{V_{EN}}{R_4} \Rightarrow \frac{V_{start} - V_{ENH}}{R_6} = \frac{V_{ENH}}{R_4}, \frac{V_{stop} - V_{ENL}}{R_6} = \frac{V_{ENL}}{R_4}$$

$$\therefore R_6 = 100k\Omega, R_4 = 8.06k\Omega \Rightarrow V_{start} = 10.8V, V_{stop} = 10.4V$$

(4) L_1, considering the ripple rate is 40%, as an experienced value provided by the datasheet

$$L_1 = \frac{V_{out-max} \times (V_{in-max} - V_{out})}{V_{in-max} \times r \times I_{out} \times fsw}$$

$$V_{out-max} = 4.2 \times 3, set\ r = 0.4(ripple\ I)$$

$$\therefore L_1 = 3.14\mu H, choose\ L_1 = 4.7\mu H$$

$$I_{L-RMS} = \sqrt{I_{out-max}^2 + \frac{1}{12} \times \left(\frac{V_{out} \times (V_{in-max} - V_{out})}{V_{in-max} \times L_1 \times fsw \times 0.8}\right)^2} = 3.014A, set\ I_{out-max} > 3.5A$$

$$I_{L-CPK} = I_{out} + \frac{V_{out} \times (V_{in-max} - V_{out})}{1.6 \times V_{in-max} \times L_1 \times fsw} = 3.501A$$

(5) C_{out}

$$C_{out} > 2 \times \frac{\Delta I_{out}}{fsw \times \Delta V_{out}}$$

Assume that: $\Delta I_{out} = 75\%I = 2.25A, \Delta V_{out} = 5\%V_{out} = 0.25V$,

$$\therefore C_{out} > 22.5\mu F$$

At the same time: $C_{out} > \frac{1}{8fsw} \times \frac{1}{\frac{V_{out-ripple}}{I_{LPK} - I_{RMS}}}, set\ V_{out-ripple} = 1\%V_{out} = 50mV$

$$\therefore C_{out} > 1.521\mu F$$

Finally, since the available inductance is greater than the design value, two $22\mu F$ 25V ceramic capacitors are connected in parallel as the C_{out}.

3.2.3. Battery Monitoring Circuit

Person in charge: Liu Jingrui

Overcharging or over-discharging can both compromise the life span of a lithium battery pack. There is an optimum safe operating voltage range for a rechargeable 12V lithium battery, specifically between 10.5 V and 12.6 V. Also, receiving real-time information about the battery level is helpful since whether the robot has enough power could be checked.

Therefore, a battery monitoring circuit, including a potential divider and a buzzer alarmer, was designed. The core of this circuit is a potential divider that can divide 12V supply to be within the full-scale input range of the onboard ADC so that it can read the signal and be processed by the MCU. The buzzer is controlled by a triode SS8050 [7]. When a HIGH-level signal is given to the SS8050 by the controller, the triode will become conductive, pulling down to alarm the buzzer if the battery voltage is less than 10.5V.

Before making the final decision, three plans have been compared with each other:

Plan A: A coulometer could be used. The coulometer can provide power information with

the highest accuracy. However, this coulometer may need to work with a current sense chip to measure the power, resulting in a higher cost and more programming complexity.

Plan B: Monitoring the output voltage. Since the battery power can be converted into a corresponding electric potential, its voltage could be monitored to get rough information. This method is easy, but the accuracy is relatively low.

Plan C: Impedance tracking method: Calculating the impedance value of the object by applying a variable frequency signal to the measurement object and monitoring the amplitude and phase changes of the signal passing through the thing.

Finally, Plan B was chosen since this method could be easily achieved in hardware and software. Additionally, an ADC could be used, which our team is more familiar with. The relatively low accuracy could be corrected by programming.

The battery monitoring circuit is shown in Figure 27. KPEG242 is a buffer. The final experiment result showed that when the safe operating voltage range was exceeded, the buzzer alarmed, and the power level will be displayed on an OLED screen.

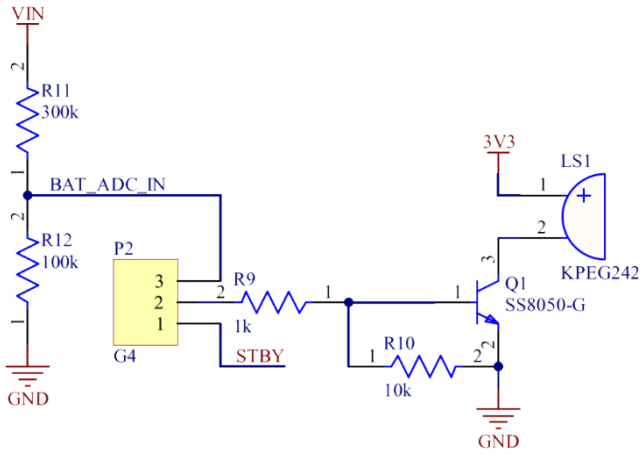


Figure 27. Schematic diagram of the battery monitoring circuit.

3.2.4. Motor Drive Circuit

Person in charge: Liu Jingrui

A motor drive circuit has been designed in this project for three reasons:

Firstly, it allows the microcontroller to control the speed, direction, and torque of the motor. As for ordinary motors, they could turn once they are powered, but when it comes to controlling the speed that they ride, how the motor speed could be controlled by controlling the voltage depends on this motor drive circuit.

Secondly, the motor drive circuit could contribute to meeting the power requirements of the motors. To meet the power requirements of the motors, the drive circuit acts as a current amplifier to make adjustments.

Moreover, it protects the rest blocks of components and devices, such as the microcontroller, from high voltage and current spikes generated by the motor during operation, providing isolation between the microcontroller and the motor, which may have different power

requirements and ground potentials.

To implement the motor drive circuit, a suitable chip should be chosen. There are two types of motor drive chips that are used to control low-power DC motors and stepper motors and are commonly seen in the market, L298N and TB6612FNG:

L298N [8]: This chip supports 7V to 35V operating voltage. The output current can reach 2A. However, L298N has relatively low efficiency and needs a heatsink to prevent overheating. Additionally, it requires a current-continuity diode to build the peripheral circuit.

TB6612FNG [9]: This chip operates motors from 2.5 to 13.5V on the motor voltage and boasts up to 95% efficiency1. It also has a low current standby mode. It has two digital inputs per H-bridge (one for each half of the bridge) and a PWM control frequency that reaches up to 100kHZ. It runs at 2.7V-5V logic and is suitable for motor voltages from 4.5V up to 13.5V.

The final decision was TB6612FNG, as compared with L298N, TB6612FNG has a higher level of integration, enabling a smaller package and simpler peripheral circuitry. It also has a higher efficiency, unlike the L298N, which requires a heatsink due to its vast heat generation.

Then, the schematic was designed according to the datasheet shown in Figure 28.

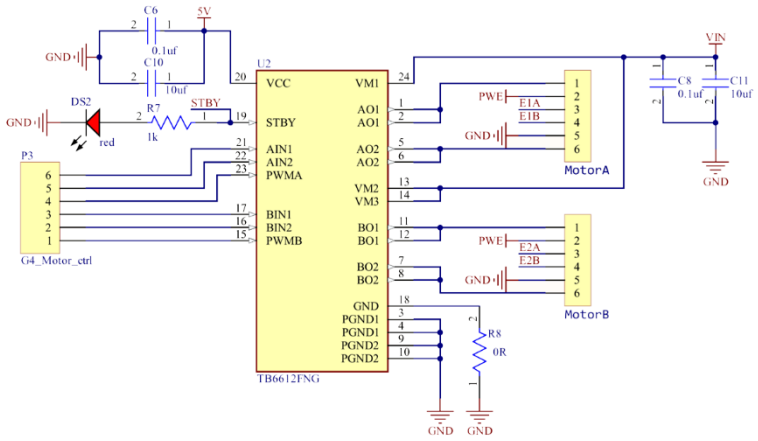


Figure 28. Schematic diagram of the motor drive circuit.

As shown in Figure 28, a single chip drives two DC step-down motors. The red LED connected to the STBY pin is an indicator to reflect the operating state of the motor. If the STBY is pulled up, the red LED will be turned on, indicating that the motors are ready to work. An NC resistor has been designed for commissioning the hall encoders.

Decoupling and bypass capacitors are also designed to improve power integrity and reduce noise. They are connected between the power supply and the ground but serve different purposes. Decoupling capacitors C6, C7, C8, C10, and C11 filter out low-frequency noise from the power supply and provide a stable voltage to the motor driver chip.

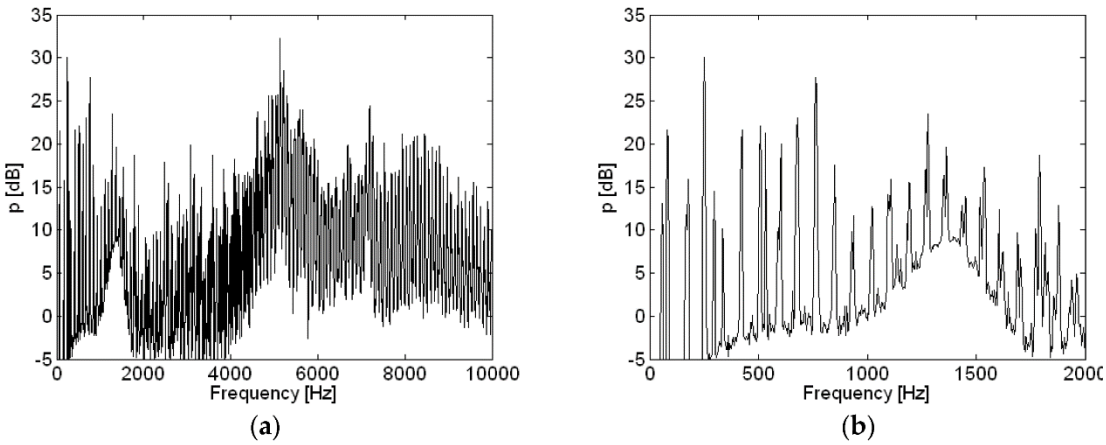


Figure 29. Spatially-averaged motor noise measurements for motor operation at 12 V with different frequency ranges [10].

One point that could be improved is the absence of ceramic bypass capacitors. Since the typical spectrum of DC motor electrical noise is a huge band and spurious pulse signal as shown in Figure 29, a spike voltage mainly generated by the disconnection of the brushes from the commutator contacts [10]. If the necessary suppression measures are not taken, the electrical interference level will often exceed the limit value. The ceramic bypass capacitor bypasses voltage spikes by providing a low-impedance path to the common side of the noise source. The capacitor can be connected between each lead of the motor and GND or between the two leads. Connecting a capacitor between the brushes and GND will have a great effect.

3.2.5. Motors

Person in charge: Wang Ruiqi

The MG513 is a DC geared motor with a Hall encoder, which provides precise and accurate control of the motor's speed and position [11]. The motor has a high torque output and smooth operation, which makes it suitable for applications that require high power output. Equipped with a Hall encoder, it provides precise feedback on the motor's speed and position.

The Hall encoder is a type of position sensor that provides precise and accurate feedback on the motor's speed and position. The encoder works by detecting changes in the magnetic field produced by the motor's magnets as they pass over the Hall sensors. This information is then used to calculate the motor's speed and position.

The MG513 motor can be controlled using a variety of motor control techniques, including PWM (Pulse Width Modulation) and PID (Proportional-Integral-Derivative) control. These control techniques allow for precise control of the motor's speed and position, which is essential for applications that require high precision and accuracy. In this project, we used PWM to control the motors.

When trying to get a robot to move in a straight line, we often think that we just need to make sure that the duty cycles of both motors are the same. This idea has been proven wrong in

practice.

Two motors of the same model will have different speeds during operation due to a number of factors. These factors include:

1. Different supply voltages: If two motors are connected to different supplies they may receive different voltages, which may result in different speeds.

2. Different loads: If two motors are under different loads, they may also have different speeds. For example, if one motor is installed in a light load application, it may run faster, while the other motor may have to carry a heavy load and run slower.

3. Mechanical faults: If a motor has mechanical faults or wears out, its speed may be reduced.

4. Magnetic field deviations: Two motors of the same model may have slight manufacturing differences, such as magnetic field deviations, which may result in different speeds.

3.2.6. PCB Integration

Person in charge: Liu Jingrui

We integrated the circuitry of the system power supply and motor driver into our independently designed PCB, firstly because it provides a stable, secure system power supply and secondly, because it simplifies the circuit structure to the maximum extent in the form of customizing the required circuit subblocks.

Additionally, as for the clock, MPU, and wireless communication modules, we also designed an additional expansion PCB for the expansion of the interface to ensure a stable, simple, and elegant connection between the above modules. The layout of the extension board is shown in Figure 32 and Figure 33.

The primary work was focused on the design of the main board. We can achieve the desired system stability with a reasonable and optimised layout design, reducing electromagnetic interference and temperature drift influence on the circuit. Therefore, we decided to choose suitable packaging, draw footprints of the critical components, and route manually.

Altium Designer is a professional PCB design software that integrates various PCB design processes, and it is a high-performance and easy-to-use tool. Our team applied Altium Designer for the entire PCB design process. The layout design of the main board with all layers visible is shown in Figure 30. The work focuses on the proper layout and routing of the components of the power and motor blocks.

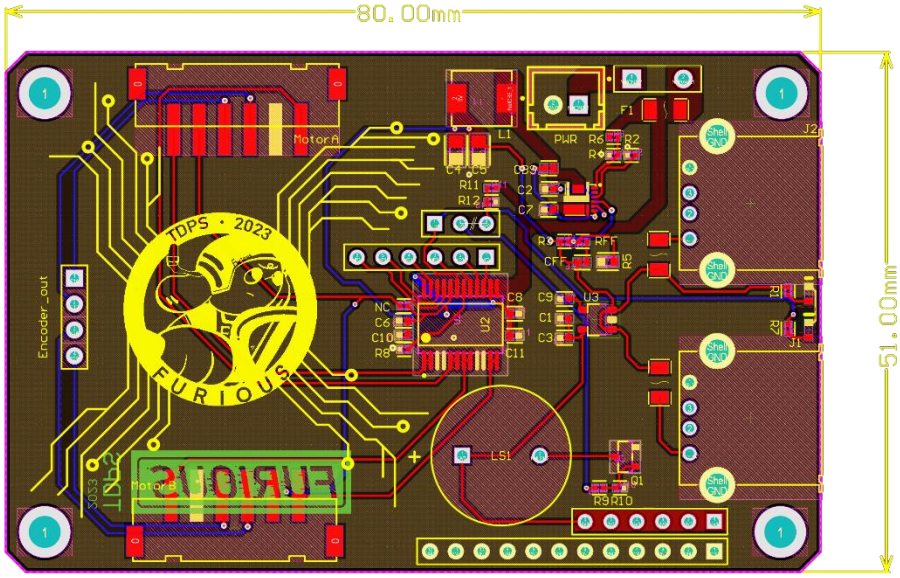


Figure 30. Layout design of the main board.

For the two blocks, the decoupling capacitors are used to filter out the noise and provide a stable voltage supply to the IC chips. The closer they are to the power pins, the less inductance and resistance they have, and the more effective they are at decoupling. If they are too far from the power pins of ICs, the trace between them and the power pins can act as an antenna and pick up noise, or create a large loop area that can radiate noise. Therefore, it is recommended to place decoupling capacitors as close as possible to the IC power pins, preferably on the same side of the board. Thus, the decoupling capacitors, which are C6, C7, C8, C10, and C11, are placed close to the power pins of SY8368 and TB6612, as shown in Figure 30.

From the same perspective, bypass capacitors are placed close to the output pin to reduce their parasitic inductance and resistance, and improve the efficiency they have at reducing ground bounce. Therefore, the decoupling capacitors C1, C2, C3, C4, C5, and C9 are placed close to the output pins of SY8368 and ME6211, as shown in Figure 30.

Other considerations include electromagnetic shielding for the inductor, the relative position between the inductor and the feedback capacitor, and minimising the trace lengths, loop areas, and parasitic elements that can degrade the performance of the capacitors. After the layout, a 3D view of the PCB could be rendered automatically. The front and back view of the main board is shown in Figure 31.

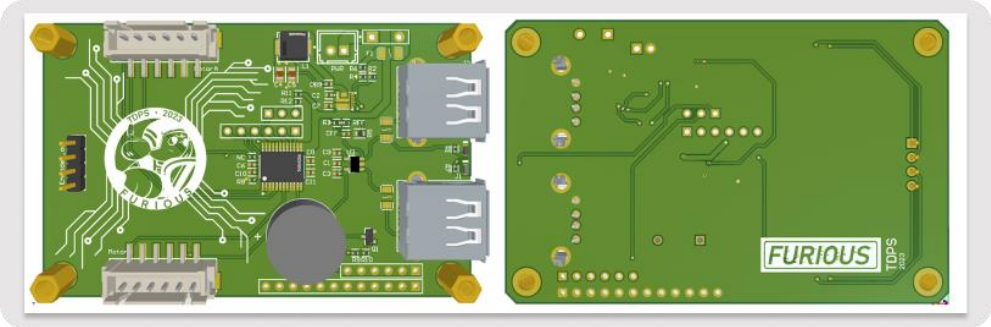


Figure 31. The front and back 3D-rendered view of the main board.

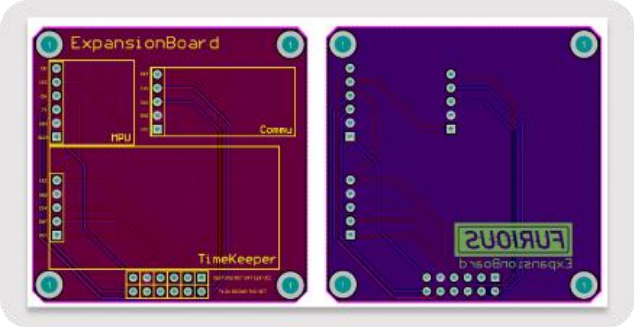


Figure 32. The extension board wiring.

The 3D-rendered front and back view of the extension board is shown in Figure 33.

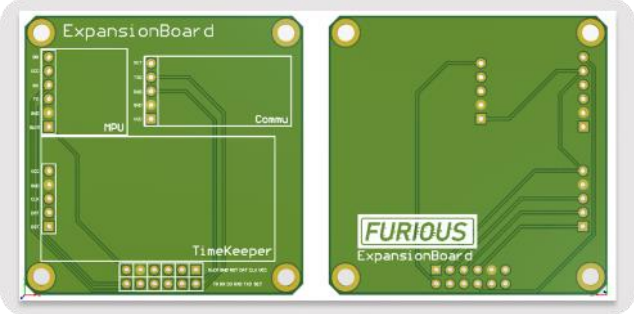


Figure 33. The front and back 3D-rendered view of the extension board.

The real shots of the soldered main board and extension board are shown in Figure 34. Figure 35 shows the power-up test result and the output accuracy of the system power supply without load.

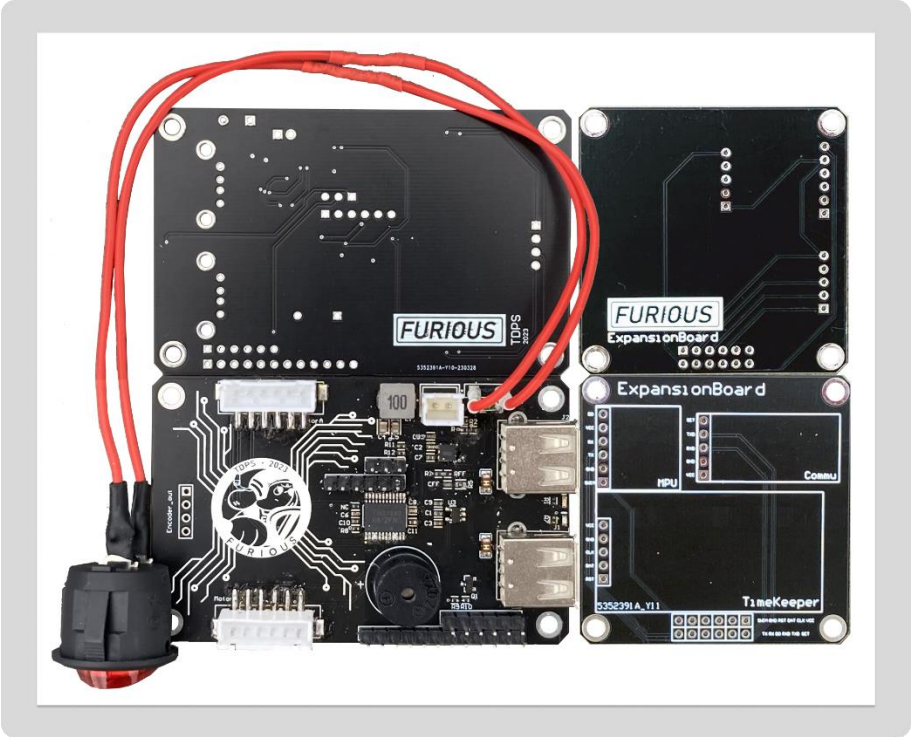


Figure 34. The front and back real shots of the soldered PCBs.

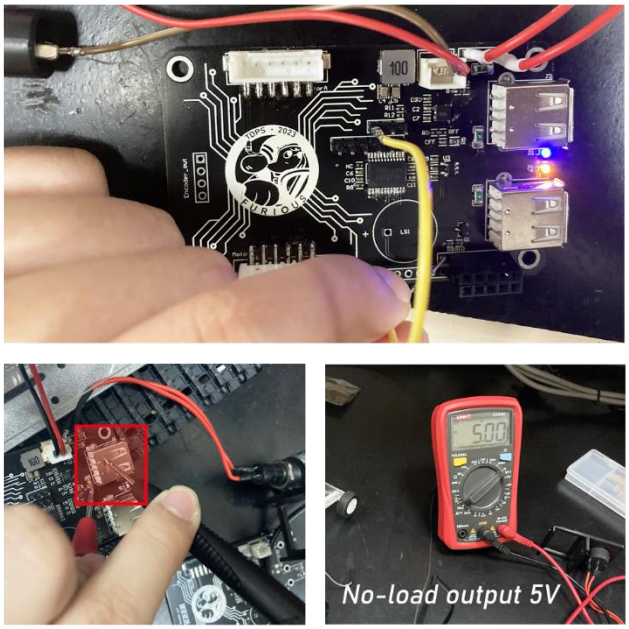


Figure 35. PCB power-up test.

3.2.7. Mechanical Arm

People in charge: Guo Yixiao, Zeng Yu

The mechanical arm system aims at completing task 2 of patio 2, which requires the robot to hold a table tennis ball at the beginning of patio 2 and place the ball in a basket after patrolling. Our team has proposed a solution that a 3D-printed claw to be installed on the trolley at enough height. After detecting the basket, a SG90 servo was used to open the claw and release the ball. The 3D-printed claw is shown in.



Figure 36. 3D-printed claw.

SG90 is a small-sized, low-cost servo motor that is commonly used in hobbyist and DIY projects. It is designed to be lightweight and compact, making it ideal for small-scale applications that require precise control of movement.

The SG90 servo motor operates using pulse-width modulation (PWM) signals from a microcontroller or other control board. It has a 3-wire interface that includes a power supply, ground, and signal line. The signal line receives PWM signals that dictate the position of the motor shaft, while the power supply and ground lines provide power to the motor.

The SG90 servo motor has a 180-degree rotation range and can operate at a speed of up to 60 degrees per second. It also has a torque output of up to 1.8 kg/cm, making it capable of lifting small loads or controlling the movement of small mechanical components.



Figure 37. SG90 servo.

Raising the height of the arm was to fix a PVC pipe at a 45-degree angle to the surface of the robot on the third layer. The fixing method was to cut a triangular plate using laser cutting and then fix the triangular plate to the third layer board through triangular brackets. The PVC pipe was fixed to the triangular plate using screws and nuts. The lead to the servo was connected to the MCU by passing through the PVC pipe. Additionally, we have also designed a 3D printed connector which is shown in Figure 38. The connector was fixed to a PVC pipe while holding the claw in a constant position.

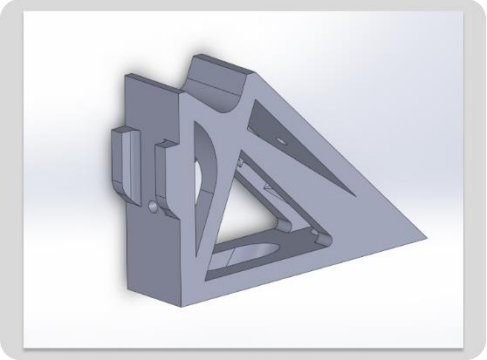


Figure 38. 3D printed connector.

In addition, the OpenMV was also loaded onto the mechanical arm via a slider which is shown in Figure 39. The position of this slider on the mechanical arm could be adjusted.

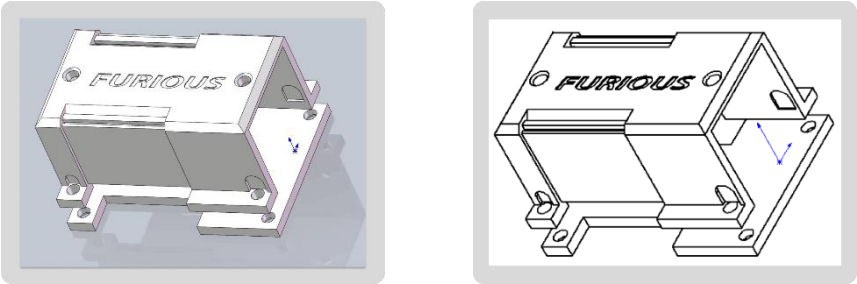


Figure 39. Slider design drawing.

The overall mechanical arm is shown in Figure 40.



Figure 40. The overall mechanical arm.

3.3. Wireless Communication System Implementation: Message Delivery

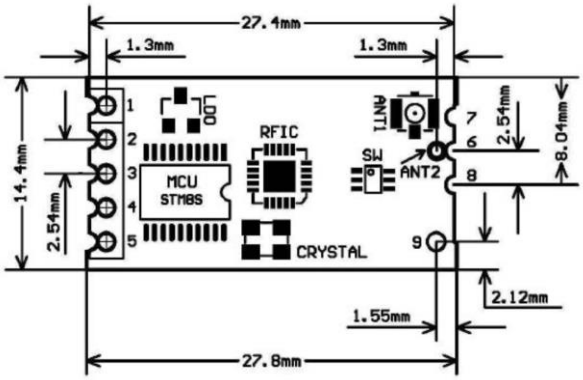
Person in charge: Zhang Jinyang, Peng Yankai, Li Yusen

During patio 2-task 3, our robotic unit is programmed to stop moving when it enters the designated planter region. At this point, it is tasked with transmitting a comprehensive message directly to a linked laptop. The message includes critical elements such as the team name, current date, day of the week, and timestamp of this operation. We will use a Wavesens HC-12 wireless transceiver operating at 433MHz to facilitate the transmission of this critical data and

ensure fast and seamless transmission without latency or interruptions. The robot is programmed to resume its journey to the end of its designated route when the laptop successfully receives the signal.

3.3.1. HC-12

The HC-12 [12] wireless serial communication module is a revolutionary addition to wireless data transmission technology. It operates within the 433.4-473.0MHz band and can transmit with a robust power of up to 100mW. With 400 KHz increments and support for up to 100 channels, this module adapts easily to different applications. The module's receiving sensitivity reaching -117dBm ensures errorless communication up to a distance of one kilometer in open areas. This module effortlessly integrates into application systems due to its stamp hole design, compact size, and patch welding support. It offers two distinct antenna options to its users. The first is an ANT1 PCB antenna base for an external 433M antenna, while the second is an ANT2 junction for spring antenna welding.





3.3.2. UART

UART, an acronym for Universal Asynchronous Receiver/Transmitter, is a widely used hardware device or protocol that manages asynchronous serial communication between computers, microcontrollers, and other peripheral devices. One of the main advantages of UART is its simplicity and ease of use. It only requires two wires for communication: one for transmitting data (TX) and another for receiving data (RX). This feature makes UART ideal for basic serial communications over relatively long distances. Unlike SPI and I2C, UART does not require a clock signal for data synchronization. Instead, data is transferred asynchronously. This means the transmitter and receiver merely need to agree on the data rate, also known as the baud rate, prior to communication. The data structure in UART communication is quite straightforward. It starts with a start bit, followed by data bits (usually 8), optionally a parity bit for error checking, and finally a stop bit.

In order to use the UART, we need to figure out how to use it, the specific steps are summarised below:

1. Configuration: The first step in using UART is to configure the settings for the communication. Both devices involved in the communication must agree on these settings, which

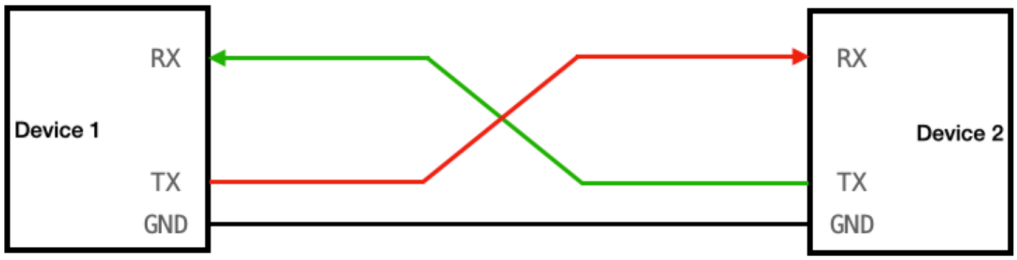
typically include the baud rate (speed of communication), data bits (commonly 8), stop bits, and parity (for error detection).

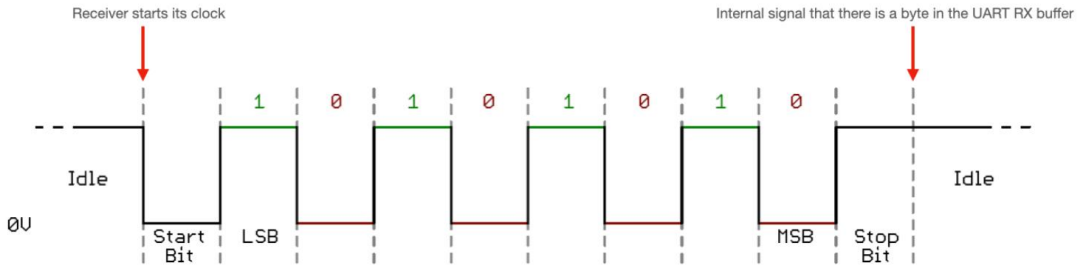
2. Data Transmission: Data transmission using UART follows a specific structure called a data frame. Each frame starts with a start bit, followed by the data byte, then an optional parity bit, and finally one or more stop bits. When the UART transmitter sends data, it converts the data from parallel to serial form, then sends it bit by bit.

3. Data Reception: On the receiving side, the UART receiver converts the received serial data back into parallel form. It checks for start and stop bits to determine the beginning and end of the data frame. If parity is used, the receiver also checks the parity bit for possible transmission errors.

4. Error Checking: When the parity bit is used, any error during the transmission can be detected. If an error is found, the receiving device must have an error-handling mechanism in place, typically requesting the transmitting device to resend the data.

5. Data Processing: Once data is received and any errors are addressed, the data can be processed as needed for the given application.





The HC-12 module applies UART as a crucial part of its data handling process. It uses UART for both initial configuration and ongoing communication with a connected device. By employing UART, the HC-12 receives data in a format that's ideal for wireless transmission. The module interprets incoming UART data, wirelessly transmits it using its built-in RF transceiver, and converts received RF signals back into UART data for the connected device. This underlines how central the UART protocol is to the HC-12's functionality in a wireless communication setup.

3.3.3. Wireless data transmission

The working principle of wireless data transmission using the HC-12 module involves two main components: the transmitter and the receiver. For the transmitter, the HC-12 module

encodes the serial data and modulates it using FSK modulation, which involves shifting the carrier frequency between two predefined frequencies to represent 0s and 1s in the data stream. Then, the encoded and modulated data is then transmitted wirelessly using the HC-12 module's built-in antenna. The module uses the 433MHz frequency band, and the transmitted data can reach a range of several hundred meters, depending on the environment and obstacles. For the receiver, it received the transmitted data through its antenna, which operates on the same frequency as the transmitter module. The received data is decoded and demodulated by the HC-12 module, converting it back into the original serial format. Then, the decoded data is then sent to the receiving device through UART. The receiving device can process the data according to its intended application.

3.3.4. DS1302 Clock Module

The DS1302 [13] is a trickle-charge timekeeping chip from Maxim Integrated that contains a real-time clock/calendar and 31 bytes of static RAM. It communicates with a microprocessor via a simple serial interface. The real-time clock provides seconds, minutes, hours, days, dates, months, and year information. The end-of-the-month date is automatically adjusted for months with fewer than 31 days, including corrections for leap years. The clock operates in either the 24-hour or 12-hour format with an AM/PM indicator. The DS1302 has a built-in power-sense circuit that detects power failures and automatically switches to the backup supply. Timekeeping operation continues while the part operates from the backup supply. The DS1302 is designed to operate on very low power and retain data and clock information on less than 1μW. The DS1302 is particularly useful in applications where low power consumption is critical, such as battery-backed devices. In our robot car, we connect the DS1302 chip to the L432KC microprocessor. To set up this device, the following steps are conducted:

Software Configuration:

1. Import or configure the necessary libraries for communication with the L432KC.
2. Configure the selected GPIO pins as per the L432KC specifications and assign the correct pin numbers.
3. Implement the software driver for the communication protocol with the DS1302 and we choose SPI as our protocol.

Initialization and Communication:

1. In the main program, initialize the selected GPIO pins, set their direction, and configure other necessary parameters.
2. Initialize the DS1302 module by sending control commands and data to configure its operating mode and time format.
3. Read or update the time and date by sending commands using the communication protocol with the DS1302.
4. Use the relevant features and libraries of the L432KC to perform desired operations based on the time information provided by the DS1302.

To apply the DS1302 module to the L432KC microcontroller, connect the VCC and GND pins

of the DS1302 to the 3.3V power pin and ground, respectively. Then, connect the DAT, CLK, and RST pins to three separate configurable GPIO pins of the L432KC microcontroller.

3.3.5. Programming Overview

To complete the required tasks, wireless communication modules are programmed in three steps, as outlined below:

1. Import the repository of DS1302 timekeeping module from the mbed website.
2. Program the DS1302 module for time data reading.
3. Program the HC-12 module to transmit the message with time data to the laptop.

In the library file of DS1302, several functions that enable the module to function properly were defined. Among these functions, two crucial ones for this project would be introduced in the remainder of this section.

For the first start of time recording, the time setting function is desired to initial the time data. The definition of this function was shown as below. The input of function should be a UNIX timestamp, which is the total seconds count from the January 1st, 1970. This timestamp was broken down into several components: year, month, day, week, hour, minute and second. These components were written into the corresponding register as binary-coded decimal (BCD) code, and the register address was predefined in the header file.

```
1. void DS1302::set_time(time_t t)
2. {
3.     struct tm *_t = localtime(&t);
4.     writeReg(Seconds, (_t->tm_sec % 10) + ((_t->tm_sec / 10) << 4));
5.     writeReg(Minutes, (_t->tm_min % 10) + ((_t->tm_min / 10) << 4));
6.     writeReg(Hours, (_t->tm_hour % 10) + ((_t->tm_hour / 10) << 4));
7.     writeReg(Dates, (_t->tm_mday % 10) + ((_t->tm_mday / 10) << 4));
8.     writeReg(Months, ((_t->tm_mon + 1) % 10) + (((_t->tm_mon + 1) / 10) << 4));
9.     writeReg(Days, _t->tm_wday + 1);
10.    writeReg(Years, ((_t->tm_year - 100) % 10) + (((_t->tm_year - 100) / 10) << 4));
11. }
```

The second significant function is **time** that is applied to read the current time and convert it to a timestamp. It is worth noting that the data read from the register need to be convert to an integer, since the stored data was in BCD format. The definition of time was shown as follows.

```
1. time_t DS1302::time(time_t  *t)
2. {
3.     char regs[7];
4.     _CE = 1;
5.     wait_us(4);
6.     writeByte(ClockBurst | 1);
7.     for (int i = 0; i<7; i++)
8.         regs[i] = readByte();
9.     _CE = 0;
```

```
10.
11.     struct tm _t;
12.     _t.tm_sec = (regs[0] & 0xF) + (regs[0] >> 4) * 10;
13.     _t.tm_min = (regs[1] & 0xF) + (regs[1] >> 4) * 10;
14.     _t.tm_hour = (regs[2] & 0xF) + (regs[2] >> 4) * 10;
15.     _t.tm_mday = (regs[3] & 0xF) + (regs[3] >> 4) * 10;
16.     _t.tm_mon = (regs[4] & 0xF) + (regs[4] >> 4) * 10 - 1;
17.     _t.tm_year = (regs[6] & 0xF) + (regs[6] >> 4) * 10 + 100;
18.
19.     // convert to timestamp
20.     return mktime(&_t);
21. }
```

3.4. Peripherals

Person in charge: Wang Ruiqi

3.4.1. OLED

In this project, the use of OLED was not directly related to the tasks and was only used to output the name and slogan of our team.

An OLED (organic light-emitting diode) is a diode which contains organic compounds that emit light when an electric current passes through them. OLED displays are known for their high contrast ratio, wide viewing angles and low energy consumption. In recent years they have become increasingly popular and are commonly used in smartphones, TVs and other electronic devices. The technology allows for thinner screens and sharper images than conventional LCDs.

A liquid crystal display (LCD) is a flat screen or other electronically moderated optical devices that uses the light modulation characteristics of liquid crystals in combination with a polariser. The reason for not using LCDs here is that they do not display well and displaying clear text is more complex.

The texts “WE ARE FURIOUS” and our team name “FURIOUS” were displayed successfully using an OLED.

3.5. Robot Mechanical Structure

People in charge: All team members

Having implemented the functions of the different subsystems, we needed to combine the different components to form a fully functional robot, which involved designing a rational overall mechanical structure.

The large trolley chassis we have chosen for our robot is relatively small and at the same

time we need to mount different kinds of sensors and mechanics on the chassis. Taking into account the limited space and the coordination of the robot, we designed different layers to mount the different parts of the subsystem.

After the layout of each layer has been determined, the structure of each layer was drawn on AutoLaser. AutoLaser software refers to the computer programs and algorithms that operate the laser distance measurement system. It controls the hardware components such as the laser transmitter and timer, calculates the distance based on the time of flight of the laser pulse and provides the user with the results of the measurement and any other information. After the layout was finished, the laptop was connected to the laser cutter and the acrylic panels were cut into the designed structure and layout.

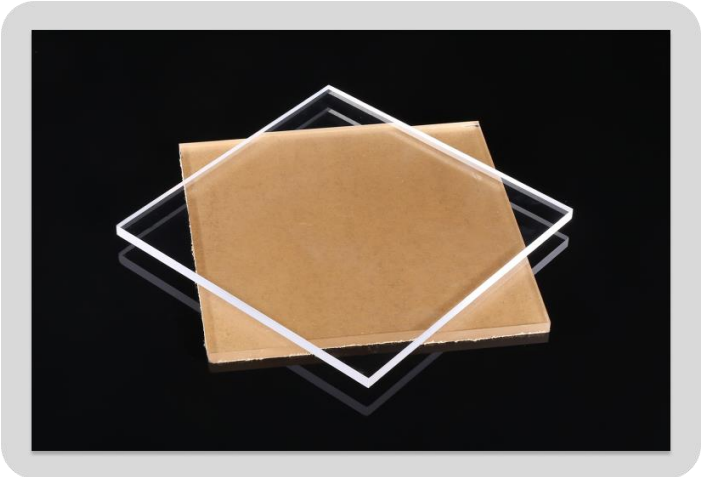


Figure 41. Acrylic panel.

The chassis of the robot and a front support frame is shown in Figure 42. The battery was fixed in the chassis of the robot rather than in one of the layers of the robot. This kind of fixation has two advantages: it can free up valuable space and it can lower the center of gravity of our robot, which is helpful for task2 in patio1 when the robot needs to cross the bridge. Additionally, probably due to the small chassis and the high body, the robot tends to roll over when going off the bridge. Therefore, we have added a front support frame to the base. As shown in Figure 42, a triangular structure made of wooden panels was also mounted on the front support frame which is used for knocking out the stands.

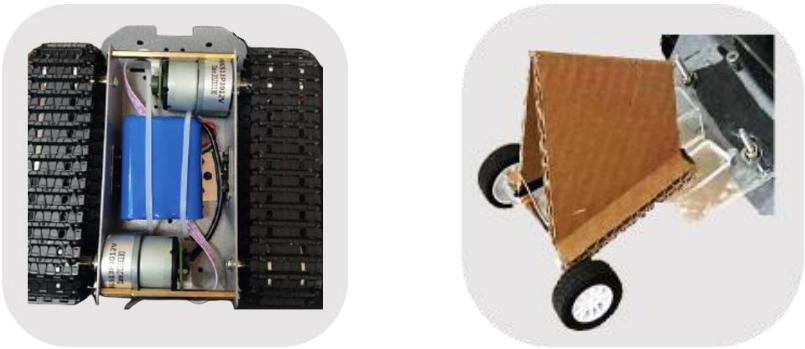


Figure 42. Chassis of the robot.

Apart from the base of the robot, we have designed a total of three layers of structure. From the bottom to the top, the first, the second and the third layers.

On the first layer, the motor driver PCB, and the extension PCB used to connect the angular velocity and accelerometer modules, wireless communication modules and clock modules (including these three modules) were placed. The schematic diagram of the structure of the first layer is shown in Figure 43.

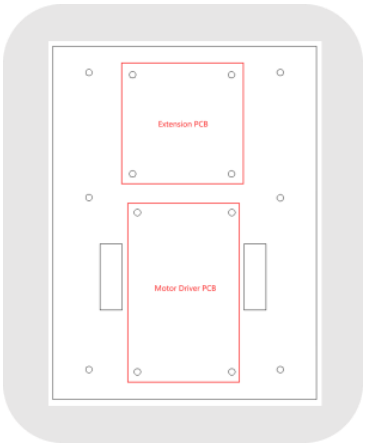


Figure 43. The schematic diagram of the structure of the first layer.

On the second layer, the three ultrasonic ranging sensors, the MCU, the OLED as well as the switch were placed. We have also designed a frame to hold the OLED and the switch, which is shown in Figure 44.

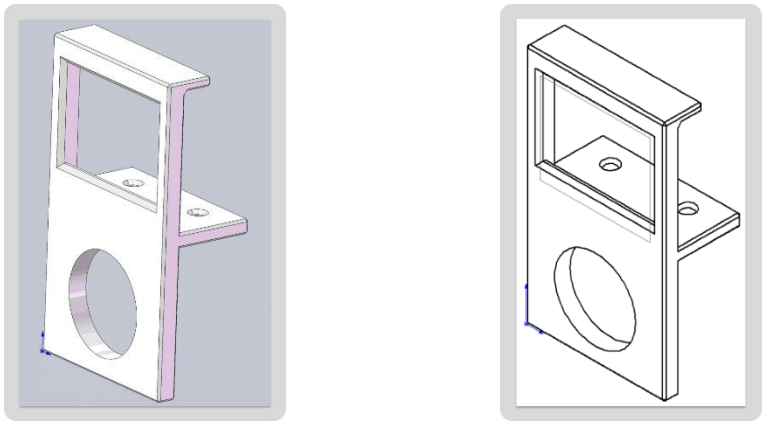


Figure 44. The frame to hold the OLED and the switch.

The schematic diagram of the structure of the second layer is shown in Figure 45. One of the ultrasonic waves was mounted directly in front of the body and the remaining two were mounted on the right side facing the forward direction.

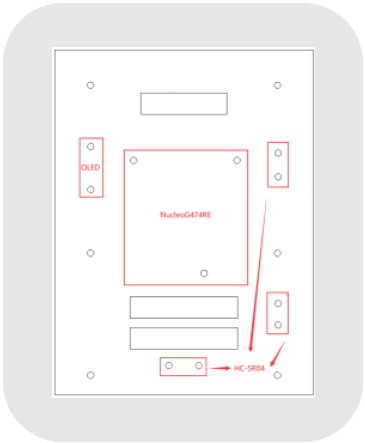


Figure 45. The schematic diagram of the structure of the second layer.

On the third layer, the PVC tube and the mechanical claw were installed. The OpenMV was fixed on the PVC tube. This kind of fixation makes it much more convenient for us to adjust the position and height of OpenMV. It also facilitated the collection of more information on the road surface by OpenMV. The schematic diagram of the structure of the third layer is shown in Figure 46.

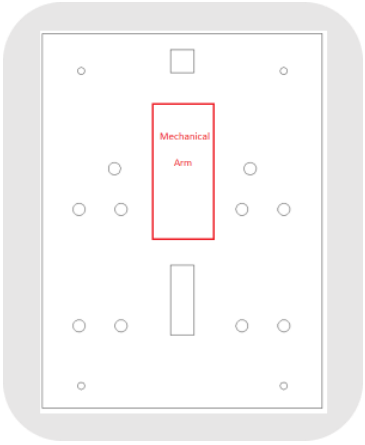


Figure 46. The schematic diagram of the structure of the third layer.

The triangular plate for connecting and fixing PVC pipes to the third layer is shown in Figure 47.

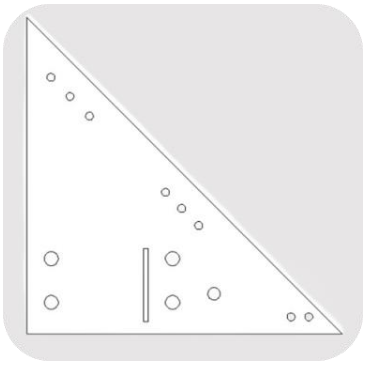


Figure 47. Triangular plate for connecting and fixing PVC pipes.

The final assembled robot is shown in Figure 48.



Figure 48. The final assembled robot.

3.6. System Integration

3.6.1. Patio 1

People in charge: Wang Ruiqi, Wang Jiaqi

The patio 1 scene layout and partial action track is shown in Figure 49. The patio 1 task can be divided into three general phases. As for the first phase, the robot will start from the starting point (green) and follow the trail to the bridge (blue) by identifying the coloured tiles. As for the second phase, the robot detects the obstacles (yellow) to its right, makes a turn and then goes up and down the bridge. As for the third phase, after coming down from the bridge, the robot recognises the coloured tiles (blue) and follows the path to the gate (purple). Once the gate has been detected, the robot will stop automatically after a short period of time.

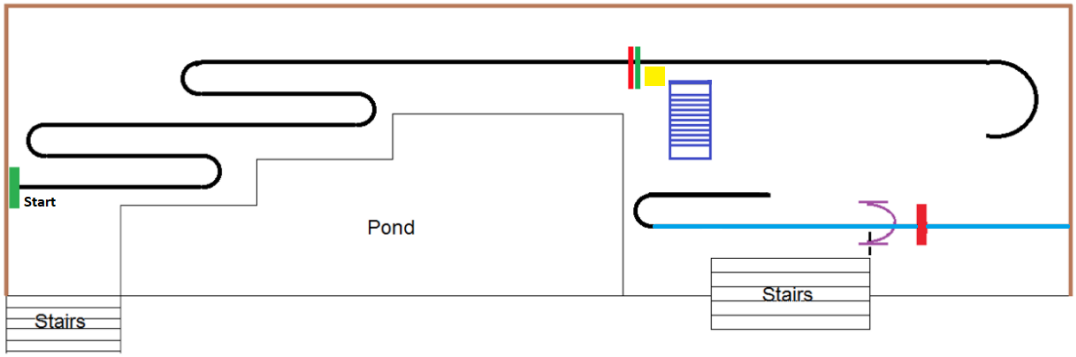


Figure 49. Patio 1 scene layout and partial action track.

The patio 1 flow diagram is shown in Figure 57, which corresponds to the control logic of the MCU code.

The first loop shown in the flowchart corresponds to the first stage of patio 1: OpenMV continuously identifies the coloured tiles and sends the processed information to the MCU, which in turn sends control signals to the motors for line tracking. During this loop, the two ultrasonic sensors on the right side of the robot are also constantly checking the distance to the obstacle. When the distance detected by both ultrasonic sensors is less than 20 cm at the same time, i.e. when the robot finishes the first phase of patrolling and comes to the yellow obstacle shown in Figure 49, the robot exits the first loop.



Figure 50. Line tracking in phase 1 of patio 1.

When the robot detects the obstacle, the robot will again travel straight ahead for 0.7 seconds to a position roughly on the horizontal median of the bridge. Then the robot itself turns 90 degrees to the right in place using the angular accelerometer and gyroscope.

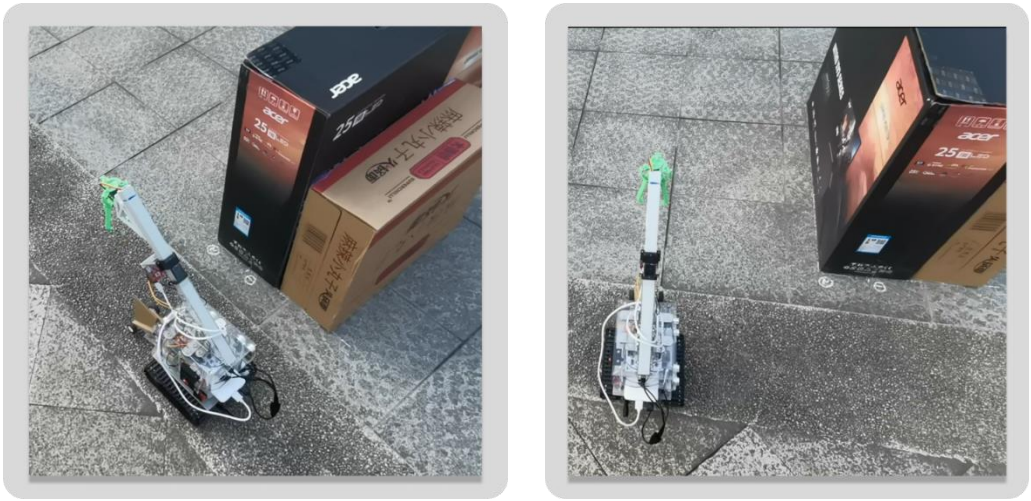


Figure 51. Ultrasonic recognition of obstacles and 90 degree turns.

The bridge is rectangular in shape and four vertical upward rods exist at the four corners of the rectangle. After a 90 degree turn, the direction of travel of the robot is not necessarily perpendicular to the plane formed by the two nearest poles to the upper bridge slope, and there may be a slight error. This error is magnified by the fact that the robot is positioned at a considerable distance from the upper bridge slope, and after a period of travel. If this error is not corrected, the robot may not be able to move up the upper bridge slope, or may travel in a bad direction of travel on the upper bridge slope.

Therefore, after the robot has turned 90 degrees, we let it travel forward again for 1.2s and stopped, adjusting the robot's forward direction in situ by means of the ultrasonic sensors on the right side (the plane of the obstacle is parallel to the plane of the right side of the bridge).



Figure 52. Traveling forward again for 1.2s and stopped, adjusting the forward direction.

After adjusting the forward direction, the robot enters the second loop shown in the flowchart. In this loop, the robot constantly adjusts itself via the JY60 (the angular accelerometer and gyroscope) so that it can move in a straight line. When either of the two ultrasonic sensors on the right side detects the rod on the bridge, the robot will jump out of this loop.



Figure 53. Getting on the bridge.

The bridge deck between the ramps will consist of barbed wire. Once the robot is on the bridge, the gaps between the tracks are partially embedded in the wire mesh, thus again correcting the robot's direction of travel. So, we can just keep the robot moving in a straight line over the bridge.



Figure 54. Crossing the bridge.

Once the robot is off the bridge and recognize the coloured tiles, it goes into the third loop shown in the flowchart. The third loop is similar to the first loop: OpenMV continuously identifies the coloured tiles and sends the processed information to the MCU, which in turn sends control signals to the motors for line tracking. During this loop, the two ultrasonic sensors on the right side of the robot are also constantly checking the distance to the gate frames. When the distance detected by one of the ultrasonic sensors is less than 30 cm, i.e. when the robot has crossed the gate (purple) shown in Figure 49, the robot exits the third loop.

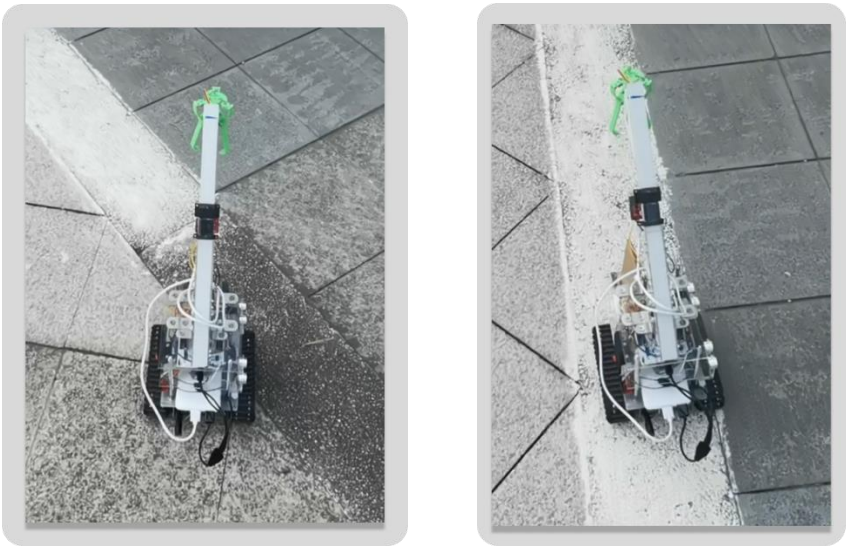


Figure 55. Line tracking after getting off the bridge.

Finally, after passing through the gate, the robot will keep travelling forward for 2s and stop.

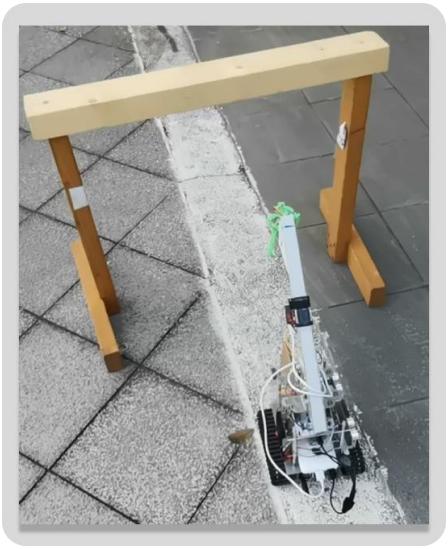


Figure 56. Passing through the gate.

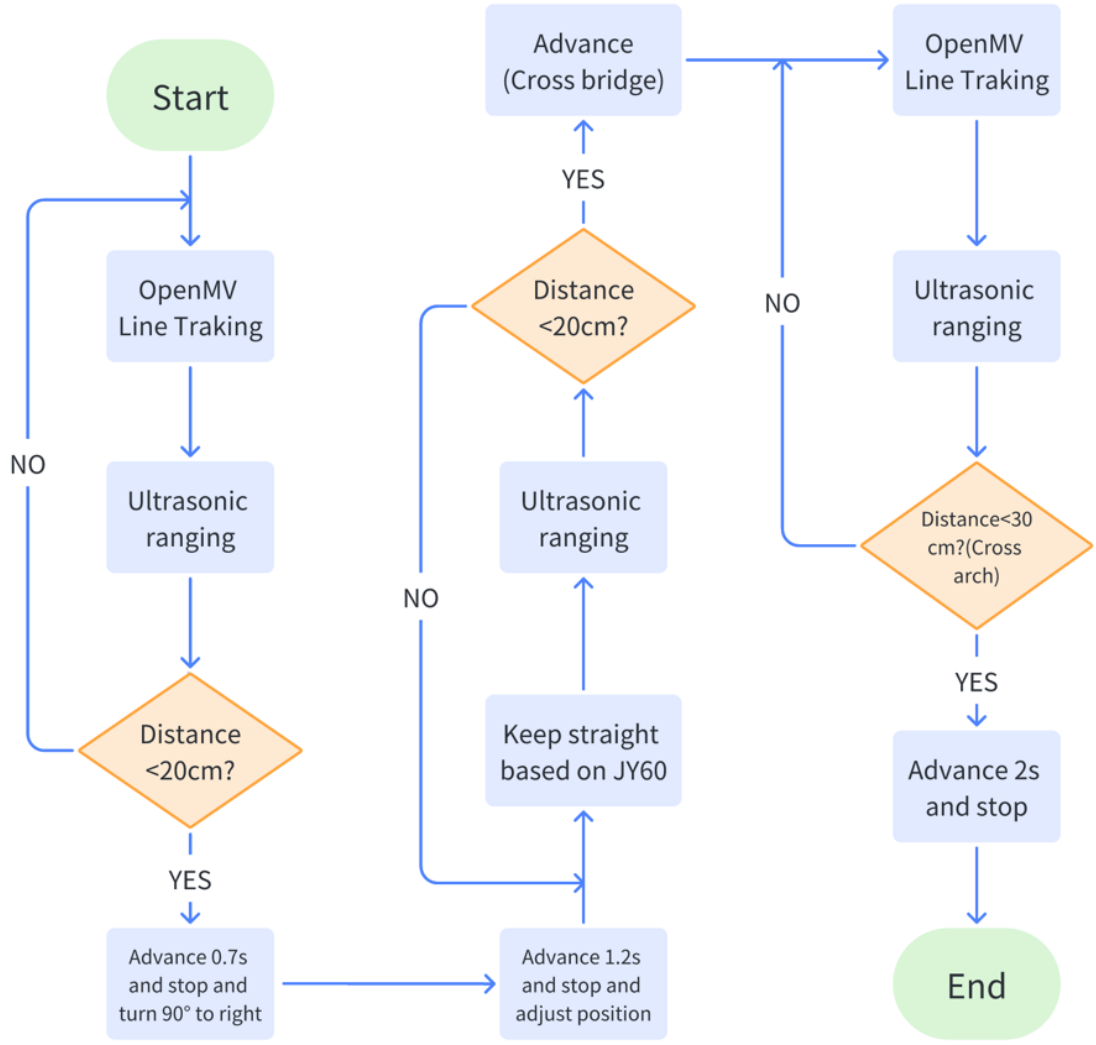


Figure 57. The patio 1 flow diagram.

3.6.2. Patio 2

People in charge: Li Yusen, Zhang Jinyang

The patio 2 tracked route is shown in Figure 58. Starting from the starting point (green), the robot will move to the yellow square where it would scan and identify the shapes on the squares to determine the direction and then knock out a bracket with matching shape in positions 1, 2 and 3. After knocking out the bracket, the robot will reach the railing (brown line near the lake). The robot will patrol along the railing with the aid of ultrasonic sensors until it reaches the ball release point (orange). At this point, the robot operated the servo to release the ball in the claw. The robot then follows a straight line through the stone road area (light blue) to the communication area. When the ultrasonic sensor recognized the beacon that we have placed, the robot will transmit a message to the laptop.

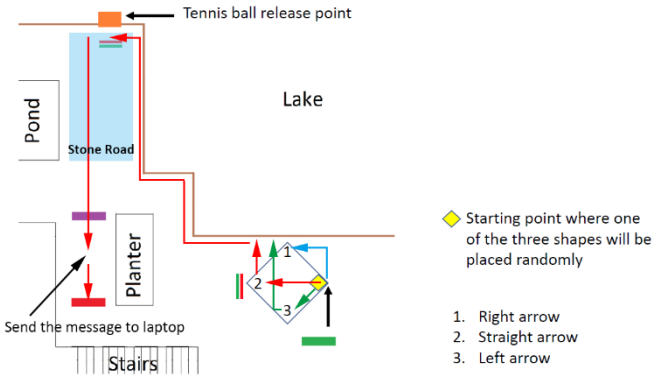


Figure 58. Patio 2 tracked route.

The patio 2 flow diagram is shown in Figure 66, which is also correspond to the control logic of the MCU code.

When the robot was started, it would keep straight movement to reach the yellow square. The stable algorithm for implementing this straight movement was shown in discussion part detailed. In the following of this section, all the mentioned straight movements utilize this algorithm. The Figure 59 shows that the robot would stop in front of the bracket at starting point (yellow square) for arrow identification.



Figure 59. Stop for arrow identification.

After the MCU got the identification result from the OpenMV, the robot would knock out the bracket by advancing for a short time. This behavior was shown in Figure 60.

Then, the robot would knock out the bracket at the other rips of diamond (position 1, 2, 3 in Figure 58) and walk near the railing. The tracks of this action were show in Figure 58 (blue: right arrow, red: straight arrow, green: left arrow). The detailed operations of robot were show in the patio 2 flow diagram (Figure 66), starting from arrow identification result judgement block.

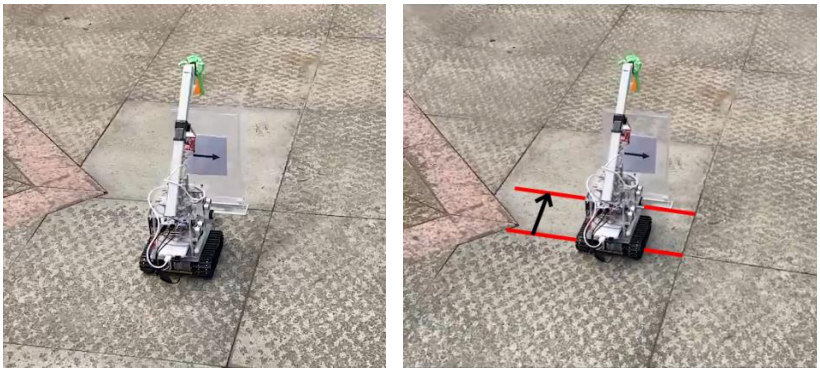


Figure 60. Knock out the bracket at starting point (yellow square).

When the robot was closed to the railing, it would start to patrol along the railing with the help of ultrasonic sensors. The corresponding patrolling algorithm was specific in the discussion section. In patrol, the robot would keep detecting if it reached the corner via the ultrasonic modules. Once this condition was met, the robot would turn to parallel with the next railing section.

The robot would patrol along the railing until it faced the stone road (light blue area in Figure 58). To pass though this region, the robot would keep straight movement directly. The robot would turn and keep forward when the front ultrasonic sensor detected the railing. After walking for a short while, the robot would stop in front of the basket.

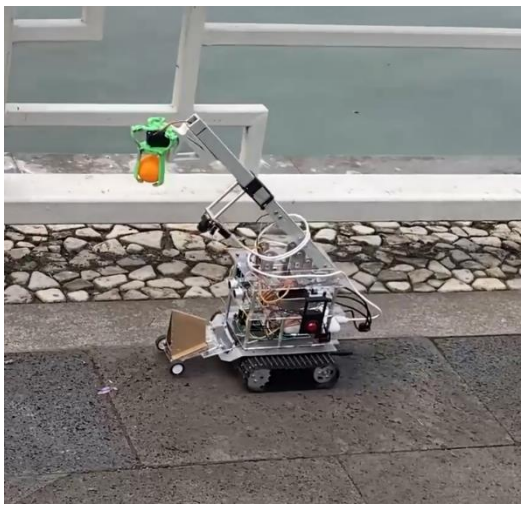


Figure 61.Patrol along railing with ultrasonic sensors.



Figure 62. The robot was facing the stone road.

Then, the robot would start to complete the task 2 of patio 2. The claw was operated to release the table tennis ball (shown in Figure 63).



Figure 63. Release the ball from claw.

After the ball was thrown into the basket, the robot would turn and walk through the stone road by utilising the keeping straight algorithm. The robot would remain in this moving state until it identified our beacon in the communication area. Completing this identification, the robot would stop and send the message with timestamp to the laptop. An example of this structured message was shown in Figure 65. Finally, the robot would way out the communication area and stop.



Figure 64. The robot identified the beacon

```
Team 41: Furious

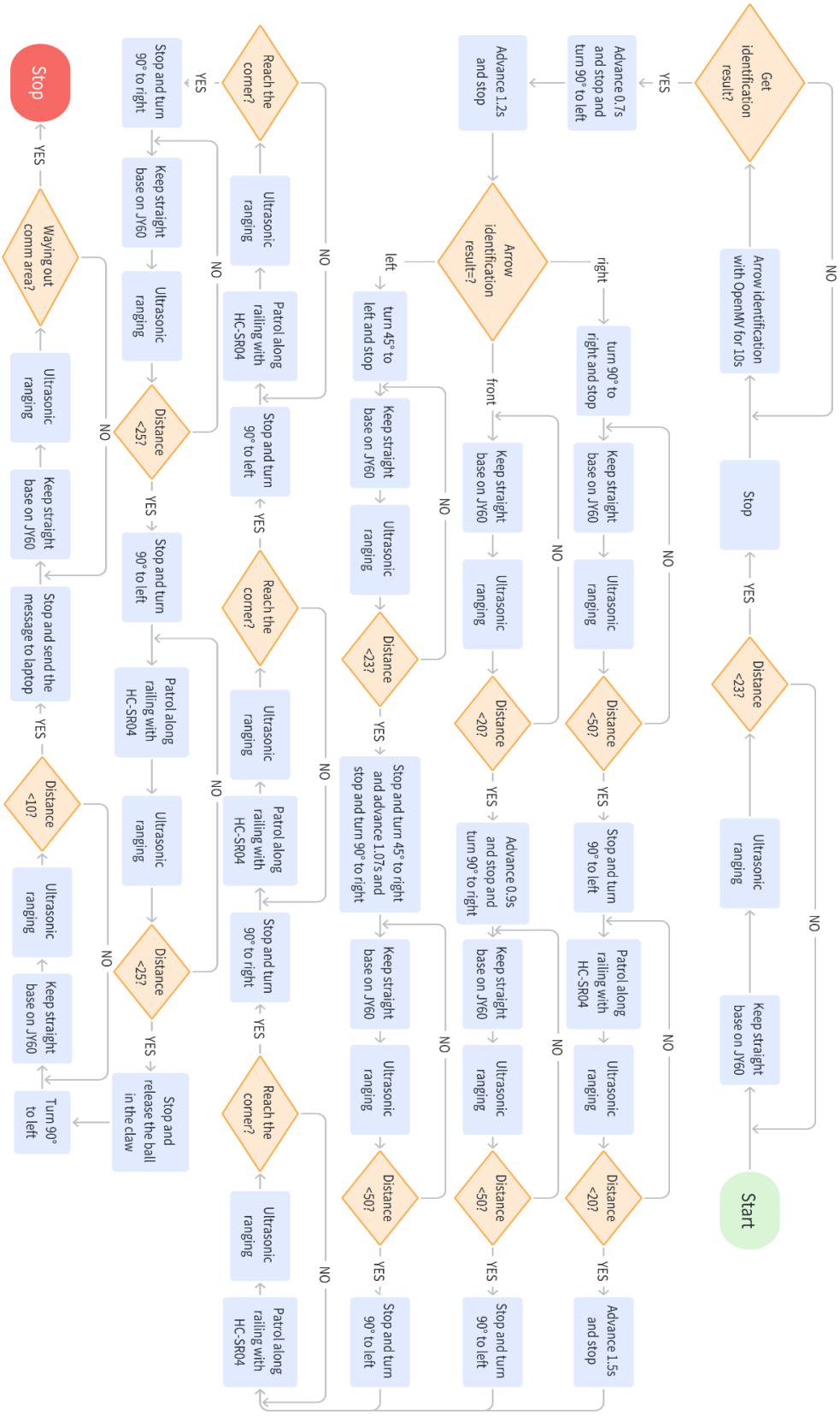
Li Ruixuan; Wang Ruiqi; Peng Yankai; Guo Yixiao; Luo Jiaqi; Li
Yusen; Liu Jingrui; Zeng Yu; Zhang Jinyang; Wang Jiaqi

Date:2023/3/19 (SUN)

Current time:22:36:52
```

Figure 65. An example of message with timestamp

Figure 66. The patio 2 flow diagram



4. Results & Discussion

4.1. Arrow Identification

In task1, the robot will stop in front of the sign and identify the arrow on the sign, but OpenMV may identify the arrow incorrectly. In order to ensure that OpenMV sends the correct arrow recognition result to the MCU, we extend the time that the robot stays in front of the sign. This allows OpenMV to perform multiple recognitions, of which the one with the highest number of occurrences is the final result. The robot then behaves accordingly to this result.

In addition, we have improved the recognition algorithm to increase the recognition accuracy. In the final solution, the previous grey-scale graphics interception algorithm was replaced as the binary algorithm for recognition. The algorithm binarizes the images captured by the camera, which significantly improves recognition accuracy. This is a common method of data enhancement. The Figure 67 shows the result of image binarization.

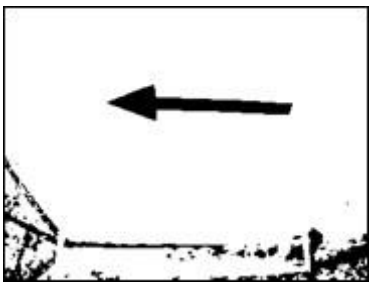


Figure 67. Example for image binarization.

4.2. Line Patrol

4.2.1. Straight Line Movement

In our field tests, we found that if the robot is only able to move in a straight line by controlling the rotation of the wheels, the robot will deviate from the straight direction after a certain period of time. To solve this problem, we chose to use the accelerometer JY60 to assist with this function: before starting to move in a straight line, the direction of the robot was read from the accelerometer as the initial direction. After this, the robot would adjust the wheels according to the difference between the real time direction and the initial direction to ensure that the robot could move in a straight line.

In Figure 68, this adjusting process was shown in detail. At the beginning, the robot is set to remain stationary for a short period to obtain the initial direction (denoted as ‘a’ in figure and code). While moving forward, robot will adjust the real-time orientation (defined as ‘yaw1’) to ‘a’. To achieve this, the robot adjusts the speed of its wheels based on the orientation deviation: increasing the speed of the wheel on the side of the deviation and decreasing the speed of the other wheel, both with the same amount of variation (defined as ‘Coeff’). For example, when the ‘yaw1’ is deviated to the right of the ‘a’, the robot increases the speed of the right wheel and decreases the speed of the left wheel. The ‘Coeff’ is calculated as follow:

$$Coeff = \begin{cases} 2\times 0.001\times |yaw1 - a| + 0.02, & Coeff < 0.1 \\ 0.1, & Coeff \geq 0.1 \end{cases}$$

Since the rotation speed of motor is control by the PWM signal, the speed was represented as the duty cycle of PWM. The setting of the two parameters of the last function is obtained by our group through a large number of field tests, which has good robustness in the field. At the same time, to ensure that the robot will not turn around in place because of the large speed difference between the tracks, the value of 'Coeff' should below 0.1.

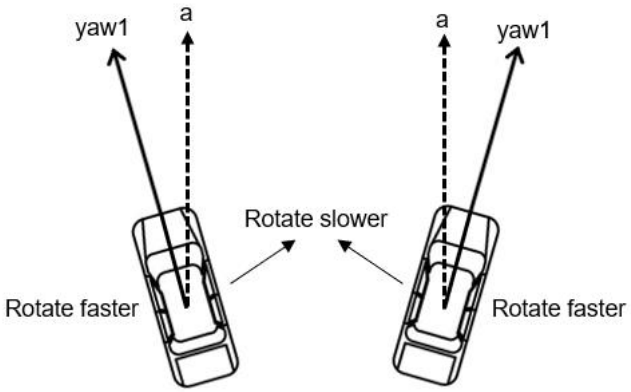


Figure 68. Adjusting the direction while moving forward.

4.2.2. Patrolling along the railings

As shown in Figure 58, in some stages, the robot will patrol the line along the railings, this is achieved by means of ultrasonic sensors mounted on the right side of the trolley. In the original solution, the robot would stop and reorient itself to be parallel to the railing before continuing on its way. However, after field tests this was found to be very time consuming. In the end we chose to reorient the robot as it travelled and to ensure that there was a fixed distance between the robot and the railing. With this improvement, the robot was able to do a good job of moving along the railings.

The algorithm to implement this function is similar to the one mentioned in the previous section. The main difference is that the orientation deviation is replaced by the difference in distance from two ultrasonic sensors to the railing, when calculating the amount of speed variation 'Coeff'. Furthermore, we expected that the robot could maintain a pre-determined distance from the railing as it patrols along it. The robot controls its wheel speeds based on the distance (measured by ultrasonic sensor) from the robot to the railing, slowing down the wheels on the side near the railing when the distance is greater than the set distance, and speeding them up when the distance is less. Conversely, the wheel speed on the opposite side is adjusted in the reverse manner: slowing down when the distance to the railing is less than the set distance and speeding up when the distance is greater.

4.2.3. Crossing the stone road

The robot needed to cross the stone road after releasing the tennis ball. The area of the stone road (blue) and the route of travel in this area (red arrow) is shown in Figure 69. When the robot is running on the stone road, large bumps occur. At this point the ultrasonic modules'

recognition of the distance to side objects can be very unstable and the robot has difficulty using this data to make adjustments to its direction of travel. As the robot only needs to travel in a straight line on the gravel road (except for turns), we have chosen to use the accelerometer directly to assist the robot to travel in a straight line through this area. The specific algorithm has mentioned in 2.2.1.

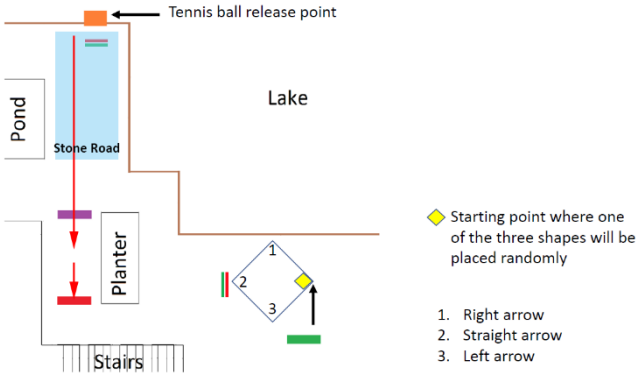


Figure 69. Patio 2 – the stone road.

4.2.4. Turning

At first, we fixed the parameters that control the robot's turns. However, we found that as the running time increased and the battery power decreased, the robot would lose power. In order to get it to turn in the right direction, these parameters need to be modified. This suggests that these parameters will be time dependent. To solve this problem, we chose to use the angle information from the accelerometer to assist the robot in turning.

4.3. Wireless Communication

4.3.1. How to start sending messages

We chose to place a marker in the communication area. As soon as this object is recognised by the ultrasonic module at the front of the robot, the robot sends a message to the laptop.

4.3.2. Clock accuracy

In the experiment, we noticed that the recorded time in DS1302 module would have a large gap between the actual time after a long-term use. The DS1302 module uses a crystal oscillator to generate clock signals, and the accuracy of the crystal oscillator can be affected by various factors such as temperature, voltage, humidity, and so on. As a result, the clock accuracy of the DS1302 module may deviate from the actual time due to the inherent instability of the crystal oscillator's accuracy.

Additionally, the first start time was set by the UNIX timestamp which is defined in the test program. Since this timestamp is likely to differ from the actual program execution time, the error between the stored time and the current time was hard to eliminate.

5. Conclusion

After undergoing initial design conceptualization, top-level structural design, modular implementation of various components, and final integration and assembly, we have completed the final project design after a period of three months. Following rigorous testing and confirmation, we are pleased to announce that our robot is capable of completing all tasks for both patios.

Among these, it also includes countless modifications we made to the initial design plan. This is because during the specific implementation process, we encountered various problems related to circuit design or code logic in an uninterrupted manner. Especially when each submodule was able to run normally, we once encountered many situations where various modules could not cooperate well with each other to achieve a specific function during the assembly process.

During the process of conducting on-site testing in designated areas, we encountered numerous issues that caused robots to malfunction due to factors related to the site or weather. Through continuous recording and analysis of the robots' operation and the working conditions of each subsystem, we made frequent modifications to our hardware design and program coding. Eventually, after countless days and nights of debugging, we successfully resolved all issues and ultimately met all design requirements.

Considering the certain limitation of total cost, we have taken into account the selection of relatively inexpensive devices in the choice of equipment for each module. Therefore, the stability of the overall robotic system still needs to be improved. Similarly, we focus more on practical considerations in our design, with some shortcomings in the appearance and internal wiring layout design.

References

[1] "OpenMV Cam H7," *OpenMV*. https://openmv.io/products/openmv-cam-h7 (accessed May 20, 2023).
[2] "HCSR04.pdf." Accessed: May 20, 2023. [Online]. Available: https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf
[3] "NUCLEO-G474RE - STM32 Nucleo-64 development board with STM32G474RE MCU, supports Arduino and ST morpho connectivity - STMicroelectronics." https://www.st.com/en/evaluation-tools/nucleo-g474re.html (accessed May 20, 2023).
[4] "MPU 6050 Module, 3 Axis Analog Gyro Sensors Accelerometer 3-5V – BigaMart." https://bigamart.com/product/3-5v-gy-521-gy521-gy-521-mpu-6050-mpu6050-mpu-6050-module-3-axis-analog-gyro-sensors-accelerometer-for-arduino/ (accessed Jun. 05, 2023).
[5] "Silergy-Corp-SY8368QNC_C125897.pdf." Accessed: Jun. 22, 2023. [Online]. Available: https://datasheet.lcsc.com/szlcsc/Silergy-Corp-SY8368QNC_C125897.pdf
[6] alldatasheet.com, "ME6211 Datasheet(PDF) - List of Unclassifed Manufacturers." https://www.alldatasheet.com/datasheet-pdf/pdf/1138839/ETC2/ME6211.html (accessed Jun. 22, 2023).
[7] "ss8050-d.pdf." Accessed: Apr. 25, 2023. [Online]. Available: https://www.onsemi.com/download/data-sheet/pdf/ss8050-d.pdf
[8] "L298N pdf, L298N Description, L298N Datasheet, L298N view ::: ALLDATASHEET :::" https://pdf1.alldatasheet.com/datasheet-pdf/view/22440/STMICROELECTRONICS/L298N.html (accessed Apr. 26, 2023).
[9] alldatasheet.com, "TB6612FNG Datasheet(PDF) - Toshiba Semiconductor." https://www.alldatasheet.com/datasheet-pdf/pdf/807693/TOSHIBA/TB6612FNG.html (accessed Apr. 11, 2023).
[10] Y. T. Cho, "Characterizing Sources of Small DC Motor Noise and Vibration," *Micromachines*, vol. 9, no. 2, Art. no. 2, Feb. 2018, doi: 10.3390/mi9020084.
[11] "MG513 Datasheet, PDF - Alldatasheet." https://www.alldatasheet.com/view.jsp?Searchword=MG513 (accessed May 20, 2023).
[12] "HC-12 Wireless Serial Port Communication Module.pdf." Accessed: Jun. 22, 2023. [Online]. Available: https://www.elecrow.com/download/HC-12.pdf
[13] alldatasheet.com, "DS1302 Datasheet(PDF) - Dallas Semiconductor." https://www.alldatasheet.com/datasheet-pdf/pdf/58478/DALLAS/DS1302.html (accessed Jun. 22, 2023).

Appendix

A. Gantt Chart

Person in charge: Luo Jiaqi

The team frequently monitors the overall progress of teamwork and makes adjustments when necessary by recording every important project event and visualizing it as a Gantt chart shown in Figure 70. The chart clearly illustrates the interdependencies of each task and their respective durations.

ID	Task Name	Start	Finish	Duration
1	Team building	2023/2/27	2023/3/1	3d
2	First team meeting & task allocation	2023/3/1	2023/3/3	3d
3	Chassis assembly	2023/3/6	2023/3/10	5d
4	OpenMV-Line Tracking	2023/3/7	2023/3/10	4d
5	Listing & purchasing the hardware components	2023/3/7	2023/3/14	6d
6	Design of PCB	2023/3/8	2023/3/10	3d
7	Mechanical arm conduction	2023/3/13	2023/3/14	2d
8	The second group online meeting	2023/3/14	2023/3/15	2d
9	Ultrasonic ranging	2023/3/15	2023/3/17	3d
10	Layout design	2023/3/15	2023/3/17	3d
11	OpenMV performance improvement	2023/3/16	2023/3/24	7d
12	Design of communication module	2023/3/17	2023/3/17	1d
13	PCB schematic	2023/3/20	2023/3/22	3d
14	The third group online meeting	2023/3/20	2023/3/20	1d
15	Programming of HC12	2023/3/21	2023/3/24	4d
16	Design of different layers of the robot	2023/3/21	2023/3/22	2d
17	Updating & Perfecting	2023/3/22	2023/3/23	2d
18	The forth group meeting	2023/3/24	2023/3/24	1d
19	Track & arrow image acquisition	2023/3/24	2023/3/27	2d
20	Subsequent processing of the Carry image	2023/3/27	2023/3/30	4d
21	PCB chip soldering & testing	2023/3/27	2023/3/30	4d
22	Field test of line tracking (different weathers)	2023/3/31	2023/4/3	2d
23	Distance measurement using HC-SR04 & G474	2023/4/4	2023/4/7	4d
24	Motor revolution measurement experiment	2023/4/7	2023/4/10	2d
25	Gyroscope angle measurement & robot turning	2023/4/10	2023/4/14	5d
26	Communication between G474 & OpenMV	2023/4/17	2023/4/25	7d
27	Robot assembly experiment	2023/4/25	2023/4/28	4d
28	Whole vehicle assembly & functional tests	2023/5/15	2023/5/17	3d
29	Patio 1 task tests	2023/5/18	2023/5/30	9d
30	Patio 2 task tests	2023/5/22	2023/5/30	7d
31	Fine-tune each part of the robot	2023/5/17	2023/5/25	7d

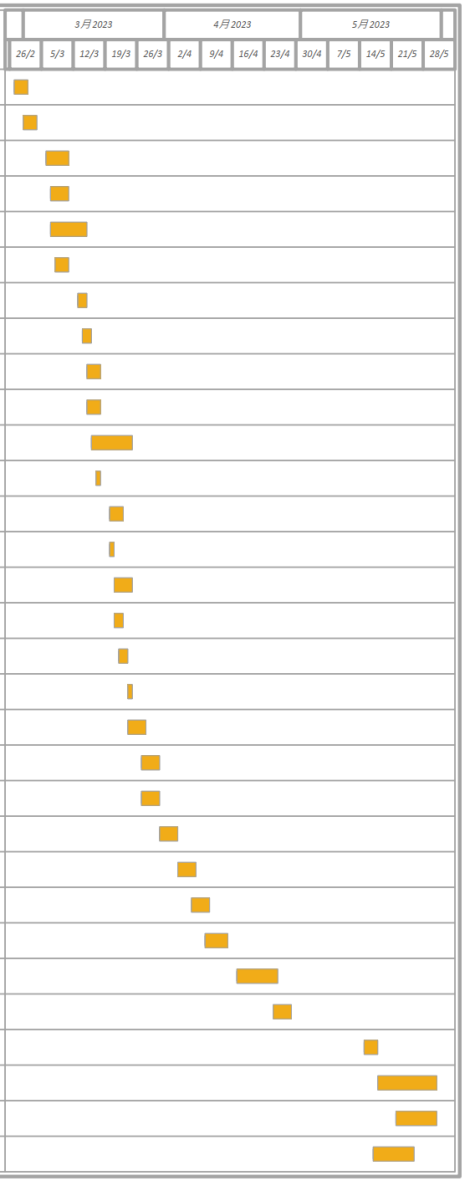


Figure 70. Gantt chart.

B. Programming

B.1. Visual Processor

B.1.1. Line Detection and Recognition

```
import sensor, image, time
import ustruct
from pyb import LED
from pyb import UART

LED(1).on()
LED(2).on()
LED(3).on()

x_ROIsize = 20
y_ROIsize = 20
lyr2Y = 90 #110 #230

sensor.reset() # Initialize the camera sensor.
sensor.set_pixformat(sensor.GRAYSCALE)
sensor.set_framesize(sensor.QVGA)
sensor.set_auto_exposure(False)
sensor.skip_frames(time = 2000) # Let new settings take effect.
clock = time.clock()

uart = UART(3,115200)
uart.init(115200, bits=8, parity=None, stop=1)

def sending_data(left_speed, right_speed, left_label, right_label):
    global uart
    data = ustruct.pack("<bbbb",
                        int(left_speed),
                        int(right_speed),
                        int(left_label),
                        int(right_label)
                        )
    uart.write(data)

while(True):
    clock.tick()
    img = sensor.snapshot()
    edges = img.find_edges(image.EDGE_CANNY, threshold = (79, 119))
    ctrd = 0
```

```
    totm = 1 # Prevent division by zero
    for i in range(16): #32
        statistics = edges.get_statistics(roi = (x_ROIsize * i, lyr2Y, x_ROIsize,
y_ROIsize))
        ctrd = ctrd + statistics.mean() * statistics.stdev() * 2 * (i - 7.5)
        totm = totm + statistics.mean()
    ctrd = int(ctrd / (0.5 * 127.5 * 0.5 * (totm + 1)))
    if ctrd >= 31:
        ctrd = 30
    if ctrd <= -31:
        ctrd = -30

    left_speed = int(40 + 1.7 * ctrd)
    right_speed = int(1.2 * (40 - 1.7 * ctrd)) # 1.2-->Motor damping difference

    if left_speed < 0:
        left_label = 0
    else:
        left_label = 1
    if right_speed < 0:
        right_label = 0
    else:
        right_label = 1

    print(f"left_speed={left_speed} right_speed={right_speed}
left_label={left_label} right_label={right_label}")
    sending_data(right_speed, left_speed, 1 - left_label, 1 - right_label)
```

B.1.2. Arrow Detection and Recognition

```
import sensor, image, time, os, tf, uos, gc
import ustruct
from pyb import LED
from pyb import UART

sensor.reset()
sensor.set_pixformat(sensor.RGB565)
sensor.set_framesize(sensor.QVGA)
sensor.set_windowing((320, 240))
sensor.skip_frames(time=2000)

LED(1).on()
LED(2).on()
```

```
LED(3).on()

uart = UART(3, 115200)
uart.init(115200, bits=8, parity=None, stop=1)

def sending_data(pk1, pk1, pk1, pk1): # ,right_speed,left_label,right_label
    global uart
    data = ustruct.pack("<bbbb",
                        int(pk1),
                        int(pk1),
                        int(pk1),
                        int(pk1)
                        )
    uart.write(data)

net = None
labels = None

try:
    # Load the pre-trained model
    net = tf.load("trained.tflite", load_to_fb=uos.stat('trained.tflite')[6] > (gc.mem_free() - (64*1024)))
except Exception as e:
    print(e)
    raise Exception('Failed to load "trained.tflite", did you copy the .tflite and labels.txt file onto the mass-storage device? (' + str(e) + ')')

try:
    # Load the labels file
    labels = [line.rstrip('\n') for line in open("labels.txt")]
except Exception as e:
    raise Exception('Failed to load "labels.txt", did you copy the .tflite and labels.txt file onto the mass-storage device? (' + str(e) + ')')

clock = time.clock()
TOT = 0

while(True):

    if TOT == 0:
        Ltot = 0
        Utot = 0
        Rtot = 0
```

```
54.
55.     clock.tick()
56.
57.     # Capture an image from the camera
58.     img = sensor.snapshot()
59.     img = img.to_grayscale()
60.
61.     # Calculate the histogram and find the threshold
62.     hst = img.get_histogram()
63.     ths = hst.get_threshold()
64.
65.     # Apply binary thresholding to the image
66.     bni = img.binary([(ths.value(), 255)])
67.     img = bni.to_rgb565() # Prepare for MobileNetV2
68.
69.     for obj in net.classify(img, min_scale=1.0, scale_mul=0.8, x_overlap=0.5,
y_overlap=0.5):
70.         # Retrieve the predicted labels and confidence values
71.         predictions_list = list(zip(labels, obj.output())) # probabilities recorded
72.
73.         # Accumulate the confidence values for each class
74.         Ltot = Ltot + predictions_list[0][1]
75.         Utot = Utot + predictions_list[1][1]
76.         Rtot = Rtot + predictions_list[2][1]
77.         TOT = TOT + 1
78.
79.         if TOT == 2:
80.             # Determine the class with the highest confidence
81.             if (Ltot > Utot) and (Ltot > Rtot):
82.                 maxC = 3
83.             if (Utot > Ltot) and (Utot > Rtot):
84.                 maxC = 2
85.             if (Rtot > Ltot) and (Rtot > Utot):
86.                 maxC = 1
87.
88.             # Send the classification results via UART
89.             for i in range(800):
90.                 sending_data(maxC, maxC, maxC, maxC)
91.             print(maxC)
92.             TOT = -1
93.
94.     print(clock.fps(), "fps")
```

B.2. Ultrasonic Sensors

The complete function of the ultrasonic sensor is defined in the following program.

B.2.1. Ultrasonic Sensors Definition

```
/**
 * @author Nestor Pereira-Neto Team: SalvadorEngenharia
 *
 * @section LICENSE
 *
 * Copyright (c) 2010 ARM Limited
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 * of this software and associated documentation files (the "Software"), to deal
 * in the Software without restriction, including without limitation the rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in
 * all copies or substantial portions of the Software.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
 * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 * THE SOFTWARE.
 *
 * @section DESCRIPTION
 *
 * Biblioteca para uso do sonar HC-SR04, possui rotinas para permitir medições
 * de distências retornado valores em centímetros, polegadas ou o tempo
 * percorrido pelo som (em microsegundos).
 * Esta biblioteca pode ser usada com qualquer pino do microcontrolador.
 */


#include "HCSR04.h"
#include "mbed.h"


```

```
39. HCSR04::HCSR04(PinName trigger, PinName echo) : _t(trigger), _e(echo) {
40.     thread_sleep_for(100);
41.     }
42.
43. float HCSR04::getCm(void){
44.     distcm = readEcho()/58;
45.     return distcm;
46.     }
47.
48. float HCSR04::getIn(void){
49.     distin = readEcho()/148;
50.     return distin;
51.     }
52.
53. float HCSR04::readEcho(void){
54.     _t=1;                       //Inicio do trigger
55.     wait_us(10);                //10us de pulso
56.     _t=0;                       //Fim do trigger
57.     while(!_e);
58.     _tempo.start();
59.     while(_e);
60.     tdist = _tempo.read_us();   //Leitura do tempo transcorrido
61.     _tempo.stop();              //Paro o temporizador
62.     _tempo.reset();             //Reset para o próximo ciclo
63.     return tdist;
64. }
```

B.2.2. Ultrasonic Sensors Header File

The following header file is written to extend the functions defined in this program for use in the main program.

```
 1. /**
 2.  * @author Nestor Pereira-Neto  Team: SalvadorEngenharia
 3.  *
 4.  * @section LICENSE
 5.  *
 6.  * Copyright (c) 2010 ARM Limited
 7.  *
 8.  * Permission is hereby granted, free of charge, to any person obtaining a copy
 9.  * of this software and associated documentation files (the "Software"), to deal
10.  * in the Software without restriction, including without limitation the rights
11.  * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
12.  * copies of the Software, and to permit persons to whom the Software is
```

```
13.  * furnished to do so, subject to the following conditions:
14.  *
15.  * The above copyright notice and this permission notice shall be included in
16.  * all copies or substantial portions of the Software.
17.  *
18.  * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
19.  * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
20.  * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
21.  * AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
22.  * LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
23.  * OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
24.  * THE SOFTWARE.
25.  *
26.  * @section DESCRIPTION
27.  *
28.  * Biblioteca para uso do sonar HC-SR04, possui rotinas para permitir medições
29.  * de distências retornado valores em centímetros, polegadas ou o tempo
30.  * percorrido pelo som (em microsegundos).
31.  * Esta biblioteca pode ser usada com qualquer pino do microcontrolador.
32.  */
33.
34.
35. #ifndef HCSR04_H
36. #define HCSR04_H
37.
38. #include "mbed.h"
39.
40.
41.
42. /**
43.  * Sonar HC-SR04 example.
44.  * @code
45.  * #include "mbed.h"
46.  * #include "HCSR04.h"
47.  *
48.  * DigitalOut myled(LED1);
49.  * Serial pc(USBTX,USBRX);
50.  *
51.  * HCSR04 sonar(PTD5, PTA13);
52.  *
53.  * int main() {
54.  *     while(1) {
```

```
 55.  *         printf("Distancia detectada pelo sensor Frente %.2f cm \n",
sonar.getCm());
 56.  *         wait_ms(1000);
 57.  *     }
 58.  * }
 59.  *
 60.  * @endcode
 61. */
 62.
 63. class HCSR04 {
 64.     public:
 65.         /** Constructor, create HCSR04 instance
 66.           *
 67.           * @param trigger TRIG pin
 68.           * @param echo ECHO pin
 69.           */
 70.         HCSR04(PinName trigger, PinName echo);
 71.
 72.         /** It make a reading of the sonar Faz uma leitura do sonar
 73.           *
 74.           * @returns Tempo do pulso echo em microsegundos*/
 75.         float readEcho(void);
 76.
 77.         /** It messures the distance in centimeter "cm"
 78.           *
 79.           *@returns Distance in centimeter
 80.           */
 81.         float getCm(void);
 82.
 83.         /** Mede a distência em polegadas "in"
 84.           *
 85.           *@returns Distência em in*/
 86.         float getIn(void);
 87.
 88.     private:
 89.         float tdist;        //Leitura do tempo transcorrido
 90.         float distcm;       //Guarda o valor da distanciância em centímetros
 91.         float distin;       //Guarda o valor da distência em polegadas
 92.
 93.         DigitalOut _t;      //Configuração do pino de Trigger
 94.         DigitalIn _e;       //Configuração do pino de Echo
 95.         Timer _tempo;       //Cria um objeto timer
 96.
```

```
97. };
98.
99. #endif
```

B.3. Accelerometer and Gyroscope

B.3.1. Accelerometer and Gyroscope Definition

The complete function of the accelerometer and gyroscope is defined in the following program.

```
1. #include "JY60.h"
2.
3. UnbufferedSerial device(A4,A5);  // tx, rx
4. unsigned char Re_buf[11],counter=0;
5. unsigned char sign=0;
6. float yaw1 = 0;
7. float yaw2 = 0;
8. int GetData1=0;
9. int GetData2=0;
10. float DeltaAngle=0;
11. float FinishFlag=0;
12.
13. void StartdeviceEvent() {
14.     //Re_buf[counter]=(unsigned char)device.getc();
15.     (unsigned char)device.read(&Re_buf[counter],sizeof(Re_buf[counter]));
16.     //if(counter==0&&Re_buf[0]!=0x55) return;       // Data No. 0 is not a frame
header
17.     if(counter!=0||Re_buf[0]==0x55){
18.     counter++;
19.     if(counter==11)             // Receive 11 data
20.     {
21.        counter=0;               // Reassignment in preparation for the next frame of
data to be received
22.        sign=1;
23.     }
24.     // Attitude resolution starts when a complete data frame is received
25.     if(sign)
26.    {
27.     sign=0;
28.     if(Re_buf[0]==0x55)      // Checking the frame header
29.     {
30.     static float lastYaw = 0;
31.     static float turnNum = 0;
32.     if(Re_buf [1]==0x53)
```

```
33.     {
34.         yaw1=0;
35.         double yaw_;
36.         yaw_ = Re_buf[7] << 8 | Re_buf[6];
37.         yaw_ = yaw_ / 32768.0 * 180;
38.         if(lastYaw < 90 && yaw_ > 270){turnNum -= 1;}
39.         else if(lastYaw > 270 && yaw_ < 90){turnNum += 1;}
40.         lastYaw = yaw_;
41.         yaw1 = yaw_ + 360*turnNum;
42.         GetData1=1;
43.     }
44.     else GetData1=0;
45.     }
46.   }
47. }
48. }
49.
50. void EnddeviceEvent() {
51.     //Re_buf[counter]=(unsigned char)device.getc();
52.     (unsigned char)device.read(&Re_buf[counter],sizeof(Re_buf[counter]));
53.     //if(counter==0&&Re_buf[0]!=0x55) return;       // Data No. 0 is not a frame
header
54.     if(counter!=0||Re_buf[0]==0x55){
55.     counter++;
56.     if(counter==11)             // Receive 11 data
57.     {
58.        counter=0;               // Reassignment in preparation for the next frame of
data to be received
59.        sign=1;
60.     }
61.     // Attitude resolution starts when a complete data frame is received
62.     if(sign)
63.    {
64.     sign=0;
65.     if(Re_buf[0]==0x55)      // Checking the frame header
66.     {
67.     static float lastYaw = 0;
68.     static float turnNum = 0;
69.     if(Re_buf [1]==0x53)
70.     {
71.         yaw2=0;
72.         double yaw_;
73.         yaw_ = Re_buf[7] << 8 | Re_buf[6];
```

```
        yaw_ = yaw_ / 32768.0 * 180;
        if(lastYaw < 90 && yaw_ > 270){turnNum -= 1;}
        else if(lastYaw > 270 && yaw_ < 90){turnNum += 1;}
        lastYaw = yaw_;
        yaw2 = yaw_ + 360*turnNum;
        DeltaAngle = abs(yaw2-yaw1);
        GetData2=1;
    }
    else GetData2=0;
    }
  }
}
}
```

B.3.2. Accelerometer and Gyroscope Header File

The following header file is written to extend the functions defined in this program for use in the main program.

```
#ifndef GYRO_H
#define GYRO_H
#include "mbed.h"

extern float yaw1;
extern float yaw2;
extern int GetData1;
extern int GetData2;
extern float DeltaAngle;
extern float FinishFlag;
//extern float a_x, a_y, a_z;
void StartdeviceEvent();
void EnddeviceEvent();
#endif
```

B.4. Patio 1 Main Program

```
/* mbed Microcontroller Library
 * Copyright (c) 2019 ARM Limited
 * SPDX-License-Identifier: Apache-2.0
 */
#include "mbed.h"
#include "platform/mbed_thread.h"
#include "HCSR04.h"
#include "JY60.h"

```

```
BufferedSerial HC(D1,D0,115200); // Serial communication with OpenMV

char buf[32]={0};


DigitalOut in1(D2);
DigitalOut in2(D3);
DigitalOut in3(D4);
DigitalOut in4(D5);
PwmOut left(D6);
PwmOut right(D7);

PwmOut Claw(A0);

HCSR04 sensor1(D8, D9);// front supersonic sensor, trig, echo
HCSR04 sensor2(D10, D11);// left supersonic sensor1
HCSR04 sensor3(D12, D13);// left supersonic sensor2
float diff;// Ultrasonic gap measurement

float distance1,distance2,distance3;
float TurnDistance=30;
float StopDistance=30;
char left_speed,right_speed;
char left_label,right_label;
int num;
float coeff;
float startangle=-90;

void init()
{

    left.write(0.0);
    right.write(0.0);
    in1=0;
    in2=0;
    in3=0;
    in4=0;
    left_speed=0;
    right_speed=0;
    left_label=2;
    right_label=2;
}
void run()
```

```
{
    left_speed=int(left_speed);
    right_speed=int(right_speed);
    left_label=int(left_label);
    right_label=int(right_label);
  if((left_label==0 || left_label==1) &&(right_label==0 ||right_label==1)){
    if (left_label==0){
        in3=0;
        in4=1;
        left_speed=left_speed-256;
        }else{
            in3=1;
            in4=0;
            }
    float left_data=float(abs(left_speed)/100.0);
    left.write(left_data);
    if (right_label==0){
        in1=1;
        in2=0;
        right_speed=right_speed-256;
        }else{
            in1=0;
            in2=1;
            }
    float right_data=float(abs(right_speed)/100.0);
    right.write(right_data);
  }else{
      left.write(0.0);
      right.write(0.0);
  }
    }
void getdata_pack()
{
    num=HC.read(buf,sizeof(buf));// read the data sent by OpenMV and store them in
buf
    // assign the data to different parameters
    left_speed=buf[0];
    right_speed=buf[1];
    left_label=buf[2];
    right_label=buf[3];
}
void angle()//JY60 angle measurement
{
```

```
    while (true)
    {
        StartdeviceEvent();
        if(GetData1)
        {
            break;
        }
    }
}
void forwardinline()
{
    angle();
    coeff = 0.001*abs(yaw1-startangle)+0.02;// Set a deflection factor that decreases with increasing distance difference of the deflection, but does not exceed 0.1 maximum
    if(coeff > 0.1){
        coeff = 0.1;
    }
    if(yaw1 > startangle){// Off to the left of a straight line
    left.write(0.36-2*coeff);
    right.write(0.30+2*coeff);
    }
    else if(yaw1 < startangle){// Off to the right of a straight line
        left.write(0.36+2*coeff);
        right.write(0.30-2*coeff);
    }
}
int main()
{
    init();
    while(true)// task1
    {
        if(sensor2.getCm()<TurnDistance&&sensor3.getCm()<TurnDistance)// Two ultrasonic modules on the side detect the markers simultaneously
        {
        thread_sleep_for(1500);// Continue on past the marker
        // Start of the turn
        in1=0;
        in2=0;
        in3=0;
        in4=0;
        thread_sleep_for(500);// Stop before the start of the turn to prevent acceleration effects
```

```
134.
135.          while(true)
136.          {
137.              StartdeviceEvent();
138.              if(GetData1)
139.              {
140.                  break;
141.              }
142.          }
143.          in2=1;
144.          in4=1;
145.          left.write(0.35);
146.          right.write(0.35);
147.          // Judging the moment to stop the turn
148.          while(true)
149.          {
150.              if(DeltaAngle>80||DeltaAngle==80)
151.              {
152.                  DeltaAngle=0;
153.                  break;
154.              }// Turn stops at 90 degrees, jump out of cycle, otherwise continue
turning
155.              else{
156.                  while (true)
157.                  {
158.                  EnddeviceEvent();
159.                  if(GetData2)// Jump out of the loop when data is received
160.                  {
161.                  break;
162.                  }
163.                  }
164.              }
165.           }// End of turn
166.
167.          in2=0;
168.          in4=0;
169.          thread_sleep_for(1000);// Stop for a while after the end of the turn
170.          break;// Jump out of the loop and end task1
171.          }
172.
173.          else// Conduct task1 rounds
174.          {
175.              getdata_pack();
```

```
176.             run();
177.         }
178.     }
179.     // Straight ahead before going onto the bridge
180.     // First level the robot by ultrasound, the distance is not important, but it must be level
181.     in1=1;
182.     in2=0;
183.     in3=0;
184.     in4=1;
185.     left.write(0.45);
186.     right.write(0.4);
187.     thread_sleep_for(2000);// A period of time forward at the end of the turn
188.
189.     in1=0;
190.     in2=0;
191.     in3=0;
192.     in4=0; // Stop to start position adjustment
193.     thread_sleep_for(1000);
194.
195.     // Leveling the robot using ultrasound
196.     while(true)
197.     {
198.         diff=sensor2.getCm()-sensor3.getCm();
199.         if(abs(diff)<0.5){
200.             break;// Consider the robot level and jump out of the loop
201.         }
202.         else if(diff<-0.5)
203.         {
204.             in1=1;
205.             in2=0;
206.             in3=1;
207.             in4=0;
208.             left.write(0.25);
209.             right.write(0.25);
210.             thread_sleep_for(100);
211.         }
212.         else if(diff>0.5)
213.         {
214.             in1=0;
215.             in2=1;
216.             in3=0;
217.             in4=1;
```

```
218.             left.write(0.25);
219.             right.write(0.25);
220.             thread_sleep_for(100);
221.         }
222.     }
223.     // Start on the bridge
224.
225.         in1=1;
226.         in2=0;
227.         in3=0;
228.         in4=1;
229.         while(true)
230.         {
231.             if(sensor2.getCm()<30||sensor3.getCm()<30)// Jump out of the loop and
start going straight when the robot recognise the railings on the bridge
232.             {
233.                 break;
234.             }
235.             else
236.             {// Adjust the robot's position if it doesn't recognise the railing on
the bridge
237.                forwardinline();
238.             }
239.         }
240.
241.         left.write(0.48);
242.         right.write(0.40);
243.         thread_sleep_for(17000);
244.         // The robot over bridge
245.         in1=1;
246.         in2=0;
247.         in3=1;
248.         in4=0;
249.         left.write(0.3);
250.         right.write(0.3);
251.         thread_sleep_for(1100);// Deflected at an angle to facilitate OpenMv
recognition
252.         while(true)
253.         {
254.
255.         if(sensor2.getCm()<StopDistance||sensor3.getCm()<StopDistance)
256.         {
257.             in1=1;
```

```
258.              in2=0;
259.              in3=0;
260.              in4=1;
261.              left.write(0.4);
262.              right.write(0.45);
263.              thread_sleep_for(2000);
264.              in1=0;
265.              in2=0;
266.              in3=0;
267.              in4=0;
268.              break;
269.           }
270.           else
271.           {
272.              getdata_pack();
273.              run();
274.           }
275.           }
276. }
```

B.5. Patio 2 Main Program

```
#include "mbed.h"
#include "HCSR04.h"
#include "platform/mbed_thread.h"
#include "JY60.h"

BufferedSerial PC(USBTX,USBRX,9600);// used to debug

BufferedSerial HC(D1,D0,115200); //Serial communication with OpenMV: UART1
char buf[32]={0};
char r1,r2;
char r3,r4;
int num;
float coeff,diff=0,a=0;

AnalogOut control(PA_4);                          //task3 on/off
DigitalOut DCout1(D2);
DigitalOut DCout2(D3);
DigitalOut DCout3(D4);
DigitalOut DCout4(D5);
PwmOut left(D6);
PwmOut right(D7);
HCSR04 sensor1(D8,D9); //front supersonic sensor, trig, echo
```

```
HCSR04 sensor2(D10,D11); //right supersonic sensor1 up
HCSR04 sensor3(D12,D13); //right supersonic sensor2 down
PwmOut myservo(PA_1);                              //the claw


void init(){
    myservo.period_ms(20);
    myservo.pulsewidth(0.0005);
    left.write(0.0);
    right.write(0.0);
    DCout1 = 0;
    DCout2 = 0;
    DCout3 = 0;
    DCout4 = 0;
    r1 = 0;
    r2 = 0;
    r3 = 0;
    r4 = 0;
    }

void forward(){// go forward
    DCout1=1;
    DCout2=0;
    DCout3=0;
    DCout4=1;
    left.write(0.47);
    right.write(0.37);
    }

void ifgetan(){// judge if the mpu6050 get the angle data
    while(true){
        StartdeviceEvent();
        if(GetData1){
           break;
        }
    }
    }

void forwardinline(){
    ifgetan();
    // Below is the deflection coefficient is set, which increases with the increase of
    // the distance difference of the offset, but the maximum is not more than 0.1.
    coeff = 0.001*abs(yaw1-a)+0.02;
```

```
    if(coeff > 0.1){
        coeff = 0.1;
    }
    if(yaw1 > a){// Veer to the left of the line
    left.write(0.36-2*coeff);
    right.write(0.30+2*coeff);
    }
    else if(yaw1 < a){// Veer to the right of the line
        left.write(0.36+2*coeff);
        right.write(0.30-2*coeff);
    }
    }

void backward(){// go backward

    DCout1=0;
    DCout2=1;
    DCout3=1;
    DCout4=0;
    left.write(0.45);
    right.write(0.38);
    }

void stop(){
    DCout1=0;
    DCout2=0;
    DCout3=0;
    DCout4=0;
    left.write(0.0);
    right.write(0.0);
    }

void alleft(){
    DCout1=1;
    DCout2=0;
    DCout3=1;
    DCout4=0;
    left.write(0.3);
    right.write(0.3);
    }

void alright(){
    DCout1=0;
```

```
    DCout2=1;
    DCout3=0;
    DCout4=1;
    left.write(0.3);
    right.write(0.3);
    }

void turnl(){
    while(true){
    alleft();
    thread_sleep_for(100);
    if(sensor2.getCm() > sensor3.getCm()){
        break;
        }
    }
    stop();
    }

void turnr(){
    while(true){
    alright();
    thread_sleep_for(100);
    if(sensor2.getCm() < sensor3.getCm()){
        break;
        }
    }
    stop();
    }

void getdata_pack(){
    HC.read(buf,sizeof(buf));
    r1=buf[0];
    r2=buf[1];
    r3=buf[2];
    r4=buf[3];
    }

void getparallel(){
    while(true){
        if((sensor2.getCm()-sensor3.getCm()) <= 0.3 && (sensor2.getCm()-
sensor3.getCm()) >= -0.3){
            stop();
            break;
```

```
        }
        else if(sensor2.getCm() > sensor3.getCm()){// The tail of the car is close to
the fence
            turnr();
        }
        else if(sensor2.getCm() < sensor3.getCm()){// The front of the car is close to
the fence
            turnl();
        }
    }
    }

void goinrange(){
    DCout1=1;
    DCout2=0;
    DCout3=0;
    DCout4=1;
    // Set the deflection coefficient, which increases with the distance
    // difference between the two ultrasounds, but the maximum value is not more than
0.1
    diff = sensor2.getCm()-sensor3.getCm();
    coeff = 0.001*abs(diff)+0.02;
    if(coeff > 0.1){
        coeff = 0.1;
    }
    if(sensor2.getCm() > 33 || diff > 0){
        left.write(0.35-2*coeff);
        right.write(0.30+2*coeff);
    }
    else if(sensor2.getCm() < 33 || diff < 0){
        left.write(0.35+2*coeff);
        right.write(0.30-2*coeff);
    }
    }

void goinrangelast(){
    DCout1=1;
    DCout2=0;
    DCout3=0;
    DCout4=1;
    diff = sensor2.getCm()-sensor3.getCm();
    coeff = 0.001*abs(diff)+0.02;
    if(coeff > 0.1){
```

```
        coeff = 0.1;
    }
    if(sensor2.getCm() > 14 || diff > 0){
        left.write(0.35-2*coeff);
        right.write(0.30+2*coeff);
    }
    else if(sensor2.getCm() < 14 || diff < 0){
        left.write(0.35+2*coeff);
        right.write(0.30-2*coeff);
    }
    }

int main(){
    init();
    // get to the recognition point
    ifgetan();
    a=yaw1;
    forward();
    while(true){
        forwardinline();
        if(sensor1.getCm() < 23){
            stop();
            break;
        }
    }
    thread_sleep_for(2500);

    // task1 start
    while(true){
        while(true){
            thread_sleep_for(10000);
            getdata_pack();
            if(r1 != 0){
                printf("%d\n",r1);
                break;
            }
        }
        forward();
        thread_sleep_for(700);
        stop();
        thread_sleep_for(900);
        alleft();// turn left 90 get to the corner
        while(true){
```

```
    ifgetan();
    if(yaw1 > 88){
        stop();
        break;
    }
}
thread_sleep_for(500);
forward();
thread_sleep_for(1200);
stop();
if(r1 == 1){// recognize the arrow point to the right, label 1
    alright();// turn right 90 1
    while(true){
        ifgetan();
        if(yaw1 < 7){
            stop();
            break;
        }
    }
    thread_sleep_for(500);
    stop();
    ifgetan();
    a=yaw1;
    forward();
    while(true){
        forwardinline();
        if(sensor1.getCm() < 50){
            stop();
            break;
        }
    }
    thread_sleep_for(2000);
    alleft();// turn left 90 1
    while(true){
        ifgetan();
        if(yaw1 > 85){
            stop();
            break;
        }
    }
    thread_sleep_for(500);
    stop();
```

```
        while(true){
            goinrange();
            if(sensor1.getCm() < 20){
                forward();
                thread_sleep_for(1500);
                stop();
                break;
            }
        }
        break;
    }

    else if(r1 == 2){// recognize the arrow point to front, label 2
        while(true){
            ifgetan();
            a=yaw1;
            forward();
            while(true){
                forwardinline();
                if(sensor1.getCm() < 20){
                    forward();
                    thread_sleep_for(900);
                    stop();
                    break;
                }
            }
            thread_sleep_for(1000);
            alright();// turn right 90 2
            while(true){
                ifgetan();
                if(yaw1 < 7){
                    stop();
                    break;
                }
            }
            thread_sleep_for(500);
            stop();

            ifgetan();
            a=yaw1;
            forward();
            while(true){
                forwardinline();
```

```
                if(sensor1.getCm() < 50){
                    stop();
                    break;
                }
            }

            thread_sleep_for(1000);
            alleft();// turn left 90 2
            while(true){
                ifgetan();
                if(yaw1 > 85){
                    stop();
                    break;
                }
            }
            thread_sleep_for(500);
            stop();
            }
            break;
}

    else if(r1 == 3){// recognize the arrow point to the left, label 3
        thread_sleep_for(2000);
        alleft();// turn left 135 3
        while(true){
            ifgetan();
            if(yaw1 > 132){
                stop();
                break;
            }
        }
        thread_sleep_for(1250);
        stop();

        ifgetan();
        a=yaw1;
        forward();
        while(true){
            forwardinline();
            if(sensor1.getCm() < 23){
                stop();
                break;
            }
```

```
}

thread_sleep_for(2000);
alright();// turn right 90 3
while(true){
    ifgetan();
    if(yaw1 < 100){
        stop();
        break;
    }
}
thread_sleep_for(1070);
stop();

ifgetan();
a=yaw1;
forward();
while(true){
    forwardinline();
    if(sensor1.getCm() < 23){
        forward();
        thread_sleep_for(1070);
        stop();
        break;
    }
}

thread_sleep_for(1000);
alright();// turn right 90 3
while(true){
    ifgetan();
    if(yaw1 < 7){
        stop();
        break;
    }
}
thread_sleep_for(500);
stop();

ifgetan();
a=yaw1;
forward();
while(true){
```

```
            forwardinline();
            if(sensor1.getCm() < 50){
                stop();
                break;
            }
        }

        thread_sleep_for(1000);
        alleft();// turn left 90 3
        while(true){
            ifgetan();
            if(yaw1 > 85){
                stop();
                break;
            }
        }
        thread_sleep_for(500);
        break;
    }
}

// task1 finished, waying to task2
thread_sleep_for(1000);
while(true){// first parallel
    goinrange();
    if(sensor2.getCm() > 60 && sensor3.getCm() > 60){
        stop();
        break;
    }
}

alright();// first turn, turn right 90 6
while(true){
    ifgetan();
    if(yaw1 < 7){
        stop();
        break;
    }
}
thread_sleep_for(500);
forward();
thread_sleep_for(1500);
stop();
```

```
while(true){// second parallel
    goinrange();
    if(sensor1.getCm() < 35){
        stop();
        break;
    }
}
thread_sleep_for(1000);

// second turn, turn left
alleft();// turn left 90 7
while(true){
    ifgetan();
    if(yaw1 > 85){
        stop();
        break;
    }
}
thread_sleep_for(500);

while(true){// third parallel
    goinrange();
    if(sensor2.getCm() > 100 && sensor3.getCm() > 100){
        stop();
        break;
    }
}
backward();
thread_sleep_for(400);
stop();
thread_sleep_for(1000);

// third turn, turn right to face the stone road
alright();// turn right 90 8
while(true){
    ifgetan();
    if(yaw1 < 11){
        stop();
        break;
    }
}
thread_sleep_for(500);
```

```
forward();
thread_sleep_for(2650);
stop();

ifgetan();
a=yaw1;
forward();
while(true){
    forwardinline();
    if(sensor1.getCm() < 25){
        stop();
        break;
    }
}
thread_sleep_for(500);

// fourth turn, turn left to face the bin and get parallel to the last fence
alleft();// turn left 90 9
while(true){
    ifgetan();
    if(yaw1 > 85){
        stop();
        break;
    }
}
thread_sleep_for(500);

while(true){// waying to the bin
    goinrangelast();
    if(sensor1.getCm() < 25){
        stop();// stop in front of the bin
        break;
    }
}

// task2 start
for(float offset=0; offset<0.0013; offset+=0.00005)
{
    myservo.pulsewidth(0.0005 + offset);
    thread_sleep_for(50);
}
thread_sleep_for(2000);
// myservo.pulsewidth(0.0005);
```

```
    // break;
    // task2 finished, waying to task3

    thread_sleep_for(1000);
    alleft();// turn left 90 10
    while(true){
        ifgetan();
        if(yaw1 > 181){
            stop();
            break;
        }
    }
    thread_sleep_for(1370);
    ifgetan();
    a=yaw1;
    forward();// first straight
    while(true){
        forwardinline();
        if(sensor2.getCm() < 10){
            stop();
            break;
        }
    }

    // task3 start
    thread_sleep_for(2000);
    control = 1;
    thread_sleep_for(6000);
    control = 0;
    // task3 finished, waying out

    ifgetan();
    a=yaw1;
    forward();
    thread_sleep_for(4000);
    while(true){// second straight, waying out the communication area
        forwardinline();
        if(sensor1.getCm() < 60 || sensor2.getCm() < 30){
            stop();
            break;
        }
    }
}
```

C. Project Management

Person in charge: Peng Yankai

In section, the overall division of work for the project as well as the workload division in four different stages are demonstrated. The overall workload of the whole project is divided into six classes: Senser System, Executive Control System, Wireless Communication system, Integration, Clerical Work and General Management, which represents the main work that team members are responsible for in the project. The Senser System class correlated to the modules required for the robot to complete the task. In order to complete this class of work, we need to have knowledge of a range of hardware and software such as embedded, deep learning. In addition, it requires a strong ability to learn on our own. The Executive Control System provides the power and control system for the vehicle. This requires knowledge of mechanical modelling, 3D printing and PCB design. The Wireless Communication system is primarily concerned with enabling wireless communication between the car and the computer. This requires knowledge of the basic theory of wireless communication, the use of the relevant modules, etc. The class of Integration is about the assembly of the robot and the field testing tasks. For the clerical work, this mainly included initial report and final report. The class of General Management is to ensure that the work of the group is carried out in an orderly manner. In addition to the project manager, we have appointed coordinators in each sub-module to manage the tasks of a particular piece

In addition, we adopt "*Situation-based task allocation*", which means the tasks were assigned to team members based on the specific circumstances, needs and different stages. Based on this, the project can be divided into *4* stages.

Stage 1 occurs between March 1 and March 19. In this period of time, we mainly focused on the preparations for the robot including the procurement of components, the commissioning of basic modules, the design of the vehicle architecture, the design of the circuitry, etc. Between 20 March and the end of March, we completed the second stage of our work. Between 20 March and the end of March, we completed the second phase of work, including the commissioning of essentially all components, the design of the PCB and so on. In addition, we have successfully completed the mid-term inspection. Then comes to Stage 3 between April 1 – April 30. At this stage, we mainly put the parts together to achieve the complete framework of the robot. At the same, the member responsible for clerical work has also started writing the final report, building on the work already done. During the last stage, we put considerable effort into commissioning the robot on site. In the meantime, we are still working on the final report and have also started the preparation of the final presentation. It was also at this stage that our group showed their spirit of sacrifice, hard work and determination to overcome all difficulties. In the end, with the joint efforts of all the team members, we successfully completed the whole project. In Table 2-6, we demonstrated the specific work in the overall project and four different stages.

Table 5. Overall task division of the project teamwork

Class	*Responsibility*	*Member Name(s)*
Sensor System	Visual Processor	*Wang Jiaqi, Li Ruixuan*

	Ultrasonic Ranging Sensor	*Wang Ruiqi*
	Accelerometer and Gyroscope	*Wang Ruiqi*
Executive Control System	PCB & Circuit Design	*Liu Jingrui*
	Mechanical Arm	*Guo Yixiao*
Wireless Communication System	Clock Module&HC-12	*Li Yusen, Zhang Jinyang, Peng Yankai*
Integration	Patio 1	*Wang Ruiqi, Wang Jiaqi, Liu Jingrui*
	Patio 2	*Li Yusen, Guo Yixiao*
Clerical Work	Initial Report	*Zeng Yu, Luo Jiaqi*
	Final report	*Zeng Yu, Peng Yankai, Li Ruixuan, Luo Jiaqi, Zhang Jinyang*
General Management	Project Manager	*Peng Yankai*
	Technical Leader	*Wang Ruiqi, Li Yusen*
	Notebook Coordinator	*Luo Jiaqi*
	Report Coordinator	*Zeng Yu*
	Presentation Coordinator	*Peng Yankai*
	Meeting Recorder	*Li Ruixuan*

Table 6. Task division in Stage 1

Task	*Team members*
Robot car frame construction	*Wang Ruiqi*
Visual processor(OpenMV)	*Wang Jiaqi, Li Ruixuan*
Motor driver & Power circuit	*Liu Jingrui, Zeng Yu, Luo Jiaqi*
Wireless communication module	*Li Yusen, Zhang Jinyang, Peng Yankai*
Mechanic arm	*Guo Yixiao*

Table 7. Task division in Stage 2

Task	*Team members*
Robot car frame construction & PID	*Wang Ruiqi*
Visual processor(OpenMV)	*Wang Jiaqi, Li Ruixuan*
Motor driver & Power circuit	*Liu Jingrui*

Wireless communication module	*Zhang Jinyang, Li Yusen*
Mechanic arm	*Yixiao Guo*
Initial report & Notebook & Presentation	*Zeng Yu, Luo Jiaqi, Peng Yankai*

Table 8. Task division in Stage 3

Task	*Team members*
Patio 1 Realization	*Wang Ruiqi*, *Wang Jiaqi*, *Liu Jingrui*
Patio 2 Realization	*Li Yusen*，*Guo Yixiao, Zhang Jinyang*
Final report	*Zeng Yu, Luo Jiaqi, Peng Yankai, Li Ruixuan*

Table 9. Task division in Stage 4

Task	*Team members*
Patio 1 field experiment	*Wang Ruiqi*, *Wang Jiaqi*, *Liu Jingrui*
Patio 2 field experiment	*Li Yusen*，*Guo Yixiao*
Final report & Final Presentation	*Zeng Yu, Luo Jiaqi, Peng Yankai, Li Ruixuan, Zhang Jinyang*

Table 10. PCB bill of materials.

Comment	*Description*	*Designator*	*Footprint*	*LibRef*	*Quantity*
10uf	General Purpose Ceramic Capacitor, 0402, 10uF	C1, C7, C9, C10, C11	FP-0402-L_1_0_1-W_0_5_0_1-IPC_B	CMP-1034-02125-3	5
2.2uf	General Purpose Ceramic Capacitor, 0402, 2.2uF	C2	FP-0402-L_1_0_1-W_0_5_0_1-IPC_B	CMP-1034-02125-3	1
0.1uf	General Purpose Ceramic Capacitor, 0402, 0.1uF	C3, C6, C8, CBS	FP-0402-L_1_0_1-W_0_5_0_1-IPC_B	CMP-1034-02125-3	4
22uF	Chip Multilayer Ceramic Capacitors for General Purpose, 0805, 22uF, X5R, 15%, 20%, 25V	C4, C5	FP-GRM21B-0_15-IPC_B	CMP-2007-03385-2	2
200pf	General Purpose Ceramic Capacitor, 0402, 200pF, 10%, X5R, 15%, 6.3V	CFF	FP-0402-L_1_0_1-W_0_5_0_1-IPC_B	CMP-1034-02125-3	1
blue	Blue LED, 10 mA, 3.9 V, -40 to 100 degC, 2-Pin SMD (0603), RoHS, Tape and Reel	DS1	VISH-TLMX1100-2_V	CMP-1666-00028-1	1
red	Red, LED, 2.1 V, -40 to 100 degC, 2-	DS2	VISH-TLMX1100_V	CMP-1666-00027-1	1

	Pin SMD (0603), RoHS, Tape and Reel				
HDR-1X4	Header, 4-Pin	Encoder_out	HDR2.54-LI-4P	Header 4	1
Fuse	25℃ 3A	F1	F1206	JFC1206-1300FS	1
Fuse	25℃ 0.7A	Fuse2, Fuse3	F1206		2
USB type-A	USB 2.0 Type A Receptacle WR-COM, Horizontal, THT	J1, J2	61400416021	CMP-1502-03227-1	2
4.7uH & 3.3uH	FIXED IND 4.7UH 4.1A 50.16 MOHM	L1	FP-PCC-M0530M-LP-MFG	CMP-05439-000015-1	1
Buzzer	Piezo Indicator, 4100 Hz, 3 to 16 V, -20 to 70 degC, 2-Pin THD, RoHS	LS1	KINGS-KPEG242-2	CMP-2000-06518-1	1
HDR-1X6	6P motor connector	MotorA, MotorB	XH2.54-LS-6P	Header 6	2
0R	RES Thick Film, 0Ω, 1%, 100ppm/°C, 0402	NC, R2, R8	FP-CRCW0402-e3-IPC_C	CMP-2002-00051-3	3
3V3_out	Header, 6-Pin	P1	HDR1X6	Header 6	1

As most of the components are purchased online, their expenses are recorded as disbursements to track the accumulating project expenses. The final project bill is listed in Table 8 for review, where each item and its financial cost are marked. Effective budget management throughout the project's duration is evident, as the financial record indicates that the total actual cost is controlled below the maximum supported budget.

Table 11. The financial record of the team by the end of the project

Item/Component	Unit Price ¥	Number	Price ¥
OpenMV	429	1	429
Ultrasonic wave module	5.53	3	16.59
Clock Module	3.44	1	3.44
Chassis	176.58	1	176.58
HC-12	11.8	2	23.6
Nucleo-G474RE	145	1	145
Aluminum Post	2.2	10	22
Acrylic Plate	1.15	15	17.25
Mechanical Claw	27.45	1	27.45
PCB	7.6	1	7.6
PVC Tube	49	1	49
JY60	4.8	5	24
Lithium Battery	39.8	1	39.8
Other consumables			12
Total			**993.31**