Master's Thesis

An Implementation of an Indoor Search and Rescue Robot for Disaster Rescue

MOU Yingzhou

Supervisor HAYASHI Yukio

Japan Advanced Institute of Science and Technology
Kanazawa University
Division of Transdisciplinary Sciences
(Transdisciplinary Sciences)

January 2023

An Implementation of an Indoor Search and Rescue Robot for Disaster Rescue

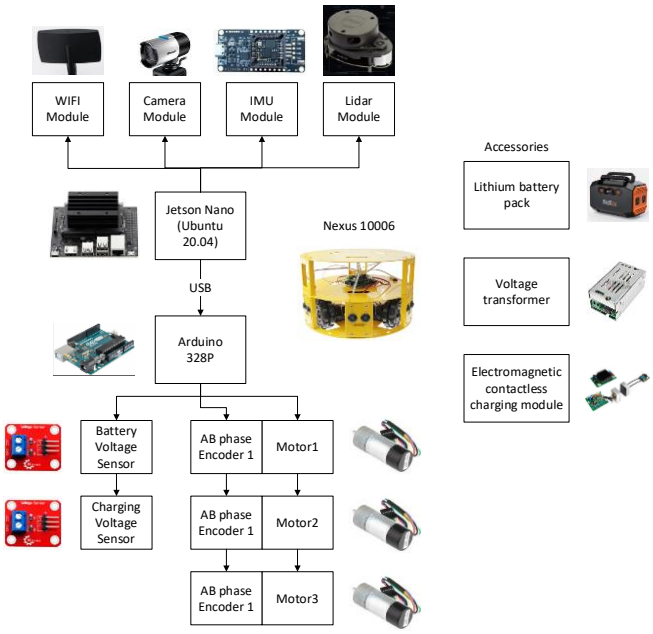
University Name: JAIST
Student Number: s2050004
Student's Name: MOU Yingzhou
Supervisors' Name: HAYASHI Yukio

1. Introduction

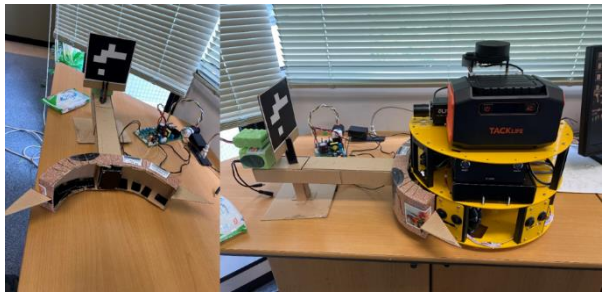
On average, a fire occurs every 23 seconds in the United States. In 2021, U.S. fire departments received a total of approximately 135,3500 fire calls, of which 486,500 fires occurred indoors, accounting for approximately 36% of the total calls. However, these fires resulted in a total of approximately 3,800 deaths, of which 2,880 were from indoor fires, a rate of 75.7% [1]. Therefore, when a fire occurs, rescuing trapped people in the fastest way will be the key to reducing the mortality rate of indoor fires. Considering the dangers of fire rescue, the use of robots instead of humans for rescue searching missions is seen as an increasing trend. However, professional rescue robots are very expensive and cannot be used as widely as fire extinguishers. Therefore, we proposed a lower-cost rescue search robot using inexpensive parts commonly found in household appliances, with the aim of making such robots more widely available.

2. Research Methods

In order to achieve the research goal, we first consider the hardware structure of the robot. The overall hardware structure diagram is shown in **Figure 1**. For the robot body, we used the Nexus 10006 chassis. It has three independent driving wheels, which are mounted 120 degrees apart on the edge of the chassis. Each wheel is equipped with an AB phases pulse encoder that outputs pulse signals as the wheel rotates. The three motors and corresponding encoders are controlled by an Arduino 328P microcontroller. In addition, we installed three drop detection sensors on the sides of the three wheels, which are also controlled by the 328P microcontroller. To monitor the remaining battery capacity and knowing if the charging base is properly connected, we use two voltage sensors. These two sensors are also connected to an Arduino 328P microcontroller. Due to the weak performance of Arduino microcontroller, in our proposal, the microcontroller is only responsible for the data acquisition of the above sensors and the control of the three motors. As for autonomous patrol function, we adopted Nvidia Jetson Nano as the central computer of the robot. The Nvidia Jetson Nano is an Arch64 single-board computer with 384 CUDA cores and 4GB RAM. In addition, a single-wire LiDAR, a 1080P fixed-focus fisheye camera, and a 9-axis inertial gyroscope are connected to the computer via a USB interface to provide the necessary information for autonomous navigation and movement of the robot. In order to make the robot self-charging, we used the BWS50-28-S1R5 electromagnetic contactless charging module produced by Bellnix. The module can provide a maximum charging power of 43W for the robot, which meets the needs of this study. In addition, we also installed a battery, voltage conversion modules, a wireless communication module and other necessary auxiliary equipment for the robot. By ourselves, we also made a charging base, and its docking schematic diagram is shown in **Figure 2**. There is an AR tag above the base, through which the robot can identify the position of the charging base and complete the docking process.



[Figure 1: Hardware structure]

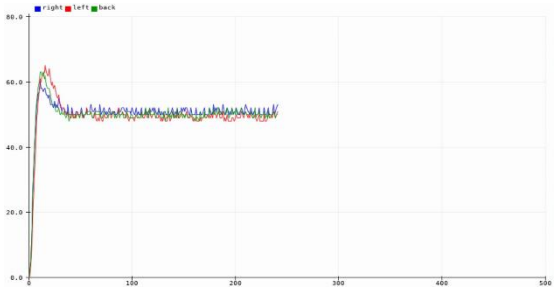


[Figure 2: Docking demo]

On the software part, we installed Ubuntu20.04 OS and associated hardware drivers for the Nvidia Jetson Nano. To simplify development, we used the ROS Noetic version of the software library and tools. The software library not only allows us to add new features to the robot car in a loosely coupled manner, but also provides some pre-packaged features. For example, the navigation module provides SLAM [2] mapping, path and other functions, and the IMU module provides the function of motor encoder and IMU sensor data fusion algorithm, etc. Based on the above conditions, we designed five functional modules for the robot, namely 1) navigation, 2) image recognition, 3) coordinate transformation, 4) hardware driver program, and 5) behavior control module to ensure the operation of the robot. It is worth mentioning that we used a machine learning model, YOLO version 5 [3], for our object detection [4] functions. This model is optimized for the ROS software library and allows for faster recognition with a lower processor load. After coding these functions, we performed various tests on these functions. For the chassis part of the robot, we performed motion tests and PID [5] parameter adjustments. Moreover, auto-charging and target recognition, we calibrated the camera using calibration charts, and to verify how the target recognition algorithm would be affected in the event of an indoor fire. The accuracy of the YOLO model is also tested in recognizing different human postures under different smoke concentrations by using dry ice to create smoke.

3. Results and Considerations

In the PID debugging test for the travel section, we specified different PID parameter values for the three motors, so that the motor motion was optimized. The results are shown in **Figure 3**. As can be seen, the three motors reached the same speed at the same time after starting. In addition, we have tested the robot's navigation and positioning, self-charging, and human body recognition functions, and achieved satisfactory results. In the smoke test, we found that the YOLO model has the highest success rate for human recognition in standing posture, while the recognition ability is weaker in squatting or lying posture. However, because we used a visible light camera that cannot penetrate smoke, the recognition success will be significantly reduced when smoke blocks the human body. The specific experimental results are shown in **Table 1**.



[Figure 3: The optimal PID output]

	No smoke		Half cover		Full cover	
Accuracy	Face	Back	Face	Back	Face	Back
Standing	0.9	0.93	0.9	X	0.86	X
Squatting	0.92	0.91	0.91	X	X	X
Lying	0.81	X	X	X	X	X

[Table 1: Smoke test result]

4. Conclusions

This study has proposed an implementation method of indoor search and rescue robot for disaster rescue. In addition, a machine learn model has been introduced to responsible for the target recognition. Unfortunately, the existing target recognition model has poor recognition ability especially for lying personnel. Meanwhile the ordinary camera does not have the ability to identify targets through smoke. Therefore, in the future, the model could be trained with human data in non-standing postures to enhance the recognition success rate. Besides, the robot's camera could be replaced with an infrared thermal one so that it can identify targets behind smoke.

Bibliography

[1] S. G. Badger, "Catastrophic Multiple-Death Fires and Explosions in the United States in 2021 (NFPA ®)," 2022.
[2] R. C. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," *Int J Rob Res*, vol. 5, no. 4, pp. 56–68, 1986.
[3] Ultralytics, "YOLOv5," *https://github.com/ultralytics/yolov5*, Apr. 21, 2021.
[4] Chinmoy Borah, "Evolution of Object Detection," *Analytics Vidhya*, Nov. 01, 2020.
[5] R. C. Panda, *Introduction to PID controllers: theory, tuning and application to frontier areas*. BoD–Books on Demand, 2012.

Contents

Chapter 1 Introduction 1
1.1 Background 1
1.2 Object 2
Chapter 2 Related Works 3
2.1 Previous Studies 3
2.1.1. Fire Rescue Robots 3
2.1.2. Navigation and Obstacle Avoidance Methods of Robot Vacuums 5
2.1.3. Self-charging Method 8
2.2 Affiliated Knowledge 11
2.2.1. PID Controls 11
2.2.2. Simultaneous Localization and Mapping 11
2.2.3. Inertial Measurement Unit 12
2.2.4. LiDAR 12
2.2.5. AR-Tags 13
2.2.6. Object Detection 13
Chapter 3 System Design 14
3.1 Hardware Selection 14
3.2 Software Development Environment 14
3.3 Hardware and Software Structures 15
3.3.1. Hardware Structure 15
3.3.2. Software Structure 16
Chapter 4 System Implementation 18
4.1 Experimental Method 18
4.2 Experimental Procedure 18
4.2.1. Moving Function Experiment 18
4.2.2. 3-D Modeling and Coordinate Transformation Setup 23
4.2.3. SLAM Mapping and Navigation 26
4.2.4. AR-Tag Based Self-charging Experiment 28
4.2.5. Object Detection Experiment 32
Chapter 5 Summary and Improvement 36
Bibliography 39

List of Figures

Figure 2.1 Scott Robot 4
Figure 2.2 SmokeBot 4
Figure 2.3 Hyper Soryu IV 5
Figure 2.4 Airship 5
Figure 2.5 Balloon 7
Figure 2.6 Docking devices by Minten et al. 9
Figure 2.7 Docking devices by Silverman et al. 10
Figure 2.8 Docking devices by Cassinis et al. 10
Figure 2.9 PID controller 11
Figure 2.10 AR Tags 13
Figure 2.11 Object detection 15
Figure 3.1 Hardware structure 17
Figure 3.2 Software structure 17
Figure 4.1 3WD robot kinematics 19
Figure 4.2 First experiment of PID 20
Figure 4.3 Reduce the value of P by half 21
Figure 4.4 Further reduce the left motor's P-value 21
Figure 4.5 First adjustment of D-values for all motors 22
Figure 4.6 An optimal parament set for three motors 23
Figure 4.7 Set the parameter I for rear motor 23
Figure 4.8 The real robot and the 3D model 24
Figure 4.9 Coordinate transformation setup 24
Figure 4.10 Coordinate transformation tree 25
Figure 4.11 Data flow 26
Figure 4.12 LiDAR map and real map 26
Figure 4.13 Create the global path 27
Figure 4.14 Obstacle avoidance and turning 28
Figure 4.15 Approaching the destination 28
Figure 4.16 Camera callibration 29
Figure 4.17 Contactless charging module on robot and charging base 29
Figure 4.18 Docking demonstration 30
Figure 4.19 Tag recognition test result 1 30

Figure 4.20 Tag recognition test result 2 31
Figure 4.21 In docking progress 31
Figure 4.22 Docking finished 32
Figure 4.23 Object detection while autonomous moving 33
Figure 4.24 Correctly operation 33
Figure 4.25 Saved picture 34
Figure 4.26 Object detection results 34
Figure 4.27 Smoke test 34

List of Tables

Table 2.1 Navigation and obstacle avoidance methods 6
Table 2.2 Comparation of sensors 8
Table 2.3 Comparation of charging methods 10
Table 3.1 Hardware list 16
Table 4.1 Smoke test 35

Chapter 1
Introduction

1.1 Background

On average, a fire occurs every 23 seconds in the United States. In 2021, U.S. fire departments received a total of approximately 135,3500 fire calls, of which 486,500 fires occurred indoors, accounting for approximately 36% of the total calls. However, these fires resulted in a total of approximately 3,800 deaths, of which 2,880 were from indoor fires, a rate of 75.7% [1]. Therefore, when a fire occurs, rescuing trapped people in the fastest way will be the key to reducing the mortality rate of indoor fires. However, the high temperature, toxic gases and collapsed interior decorations from fires will not only hinder the rescue search, but also threaten the safety of rescuers all the time. On the other hand, with the development of chip manufacturing technology and the rapid reduction of industrial manufacturing costs, automated all-terrain robots, drones and other devices no longer exist only in the imagination of science fiction, but are becoming more and more common in our lives. Considering the danger of fire scenes, the use of robots instead of humans in high-risk scenarios would be a good way to reduce the risks faced by rescuers.

In indoor fire rescue, because of the principle of life first, rescuing trapped people often has a higher priority than directly extinguishing the fire. The first step in extricating a trapped person is to determine the location of the trapped person. In the past, this was usually done based on phone calls, indoor surveillance video, and personal searches by rescuers. But there are certain limitations to such approaches. For example, it is difficult to cover all areas of a building with surveillance cameras, and the coverage area may be obscured by smoke or obstructions. The trapped people inside may lack effective channels of communication with the outside world, or they may be unconscious and unable to communicate with rescuers. Even if rescuers were to enter the building directly, they would be at great risk because they would not know the damage inside the building. If robots can be used to search for people, these problems may be solved.

In fact, there have been many robots used in fire rescue. For example, the U.S. Navy has developed a humanoid robot called THOR that can traverse the complex terrain on a ship and has the ability to open doors and extinguish fires. A robot called Thermite Robot, with

tracks and a water pump, was originally developed by the U.S. Army and will be used in mountain fire rescue missions in the U.S. Emicontrols has developed a robot called TAF20, fitted with a turbine that sprays water, and a bulldozer shovel, primarily for tunnel rescue. The turbine not only sprays water, but can also be used as an exhaust fan to clear smoke from the tunnel. Lockheed Martin made a robot called Fire Ox with a water tank that can tackle initial fires in areas that are out of reach of firemen and fire trucks. [2]

However, the use of these devices requires them to be carried and deployed to the fire by firefighters, making it difficult to be put into service immediately of a fire. In addition, fires inside buildings may block access to the interior from the outside, so the time to deploy the robots into the interior is further prolonged. If these robots were pre-deployed indoors as firefighting equipment, when a fire breaks out, the robots could be activated in the shortest possible time to automatically search and identify trapped people inside the building. This approach would save rescuers a great deal of time and ultimately save more lives. However, current indoor search robots are expensive and complex to operate, making it difficult to deploy them in advance inside buildings as part of indoor firefighting facilities, as is the case with fire hydrants. Therefore, if more inexpensive parts robots can be used and such robots are given similar functions as fire rescue robots, then the above problems will likely be solved. In addition, cheaper manufacturing costs would expand the use of robots, allowing more organizations and individuals who are constrained by financial resources to benefit from them.

1.2 Objective

In this thesis, we built a robot that can automatically patrol inside a building. In the process of patrolling, the robot will use the camera to search for the presence of people around it. When it recognizes people in the environment, it will automatically take a photo of the person and record the specific location coordinates of the person inside the building. Then, the picture with the location data will be sent to a computer used by rescue workers. After that, rescuers can refer to the location of the picture, knowing where the trapped people are. In addition, the robot has the ability of self-charging. It can recognize the position of the charging base in the experiment site using the camera and automatically complete the docking with the base. When charging is complete, it can continue to patrol the building.

Chapter 2
Related Works

2.1 Previous Studies

2.1.1. Fire Rescue Robots

In 2013, the Intelligent Mechanical Systems Engineering Laboratory of Aichi University of Technology and the Toyota City Fire Department developed a rescue search robot called Scott (Figure 2.1) through a collaboration with the Toyota City Fire Department Central Fire Department [3]. Since 2015, six rescue searches of fire scenes have been conducted using the robot Experiments. The robot has four angle-adjustable track wheels and can climb stairs. At the front of the robot, four cameras for driving were installed, three visible light cameras and one infrared camera. There is also a visible light camera for recording the situation of disaster site. In addition, the robot is equipped with carbon dioxide sensors, thermal imaging cameras, temperature sensors, and combustible gas concentration monitoring sensors for sensing environmental hazards. In order to communicate with trapped people, the robot also has microphones and speakers. In terms of navigation, the robot has a 2D Light Detection And Ranging (LiDAR) model UST-20LX, so that to generates a map of the environment. In terms of communication, the robot is equipped with both wired and wireless network devices, which can adapt to a variety of needs. The data of sensing and controlling of the robot is transmitted via the network. In 2018, a robot called SmokeBot (Figure 2.2) was developed at Örebro University, Sweden [4], which was equipped with 3D LiDAR, cameras, thermal imaging cameras, and hazardous gas sensors to work properly in a smoky environment. The rescuers can operate the robot from outside of the building, via wireless network, and view the image shooting by the robot in real time. With the 3D LiDAR and Simultaneous Localization and Mapping (SLAM) algorithm, it can automatically generate a map of the building interior and navigating with this map. If it lost the wireless network connection, it could return to the nearest communicable location automatically. Around 2007, Hyper Soryu IV (Figure 2.3), an indoor search robot with a 3-section body and track wheels, controlled by a communication cable equipped with a protective shell and hard joints, was jointly proposed by several Japanese universities. The robot has multiple cameras mounted on the front and side of the robot for remote operators to better understand the

indoor situation. In addition, it is equipped with laser distance sensors, ultrasonic sensors, and toxic gas sensors for better monitoring of fire scenes. In addition to track wheels indoor search robots, research institutions such as the University of Tokyo and Kobe University have proposed rescue search devices based on small airships (Figure 2.4) and balloons (Figure 2.5), which haven't been applied to the real fire rescue works yet.

Figure 2.1 Scott Robot

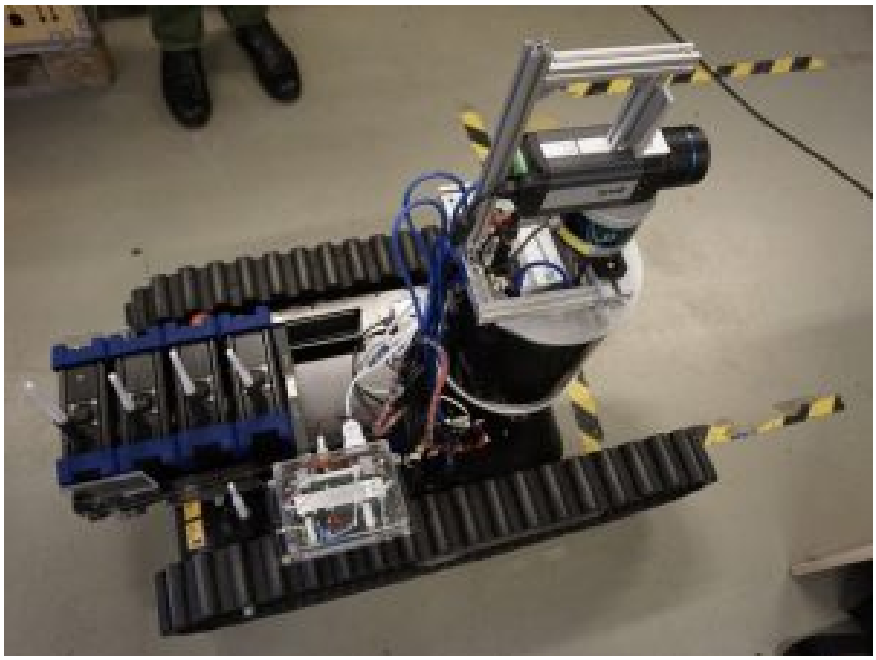


Figure 2.2 SmokeBot

Figure 2.3 Hyper Soryu IV

Figure 2.4 Airship

2.1.2. Navigation and Obstacle Avoidance Methods of Robot Vacuums

Through a web search, we have investigated 24 common brands of robot vacuums in the Chinese market and divided these 24 robots into five sections according to their selling prices. Then we have investigated the navigation methods and obstacle avoidance methods of each robot, which is summarized in the following table (Table 2.1).

Price	Mode	Navigation	Obstacle Avoidance
1000-2000CNY	JDJZ Z9	LDS LiDAR	IR collision sensor
	MIJIA 2C	VSLAM	Mechanical collision sensor
	MIJIA 2	dTof LiDAR	IR collision sensor
	ECOVACS N8	LDS LiDAR	Mechanical collision sensor
	ECOVACS N8 pro	LDS LiDAR	Structured Light
	360 X95	LDS LiDAR	mono camera
2000-3000CNY	RockBot T7S Plus	dTof LiDAR	Structured Light
	360 XIAOZHANGYU	dTof LiDAR	Mechanical collision sensor
	ECOVACS T9 Power	dTof LiDAR	Structured Light
	MI 2 Pro	LDS LiDAR	TOF
	360 X100 MAX	dTof LiDAR	IR collision sensor
3000-4000CNY	ZHUIMI W10	dTof LiDAR	Mechanical collision sensor
	YUNJING J1	dTof LiDAR	IR collision sensor
	RockBot T7 Pro	dTof LiDAR	Stereo Camera
	RockBot G10	dTof LiDAR	Mechanical collision sensor
	RUIMI EVA	dTof LiDAR	IR collision sensor
4000-5000CNY	ECOVACS T10 TURBO	LDS LiDAR	Structured Light
	YUNJING J2	dTof LiDAR	IR + Mechanical collision Sensor
	YUNJING J3	dTof LiDAR	IR + Mechanical collision Sensor
	RockBot G10S	dTof LiDAR	Structured Light+Camera
	ZHUIMI S10	LDS&VSLAM	Structured Light
Above 5000CNY	ECOVACS X1 OMNI	LDS&VSLAM	Structured Light
	RockBot G10S pro	dTof LiDAR	Structured Light+Camera
	ZHUIMI S10 pro	LDS&VSLAM	Structured Light

Table 2.1 Navigation and obstacle avoidance methods

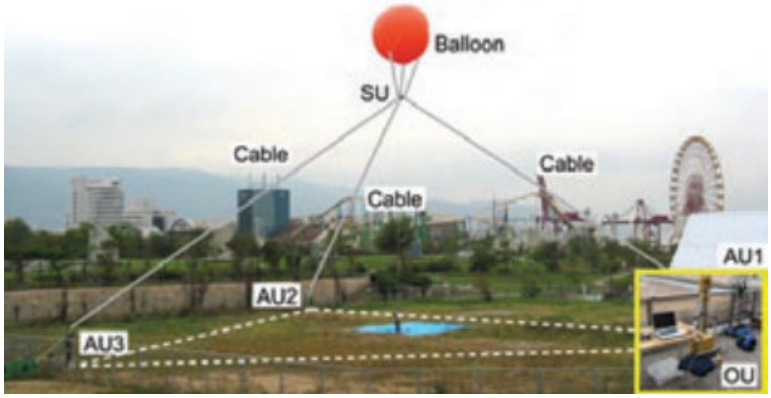


Figure 2.5 Balloon

In the navigation technology section, products at the 4,000CNY price level and below typically have only one of LiDAR or vision navigation, while more expensive models more commonly option for vision + LiDAR fusion navigation. Of all 24 robots, only one device used a pure vision-based navigation solution, 20 devices used pure LiDAR-based navigation methods, and three devices used LiDAR + vision fusion navigation technology. This indicates that LiDAR-based navigation technology is currently the more popular choice.

In terms of obstacle avoidance technology, the surveyed devices use methods including mechanical contact obstacle avoidance, infrared obstacle avoidance, structured light obstacle avoidance, camera obstacle avoidance, and camera and structured light fusion obstacle avoidance methods. Among them, infrared obstacle avoidance and structured light obstacle avoidance methods are the most common obstacle avoidance solutions. Products using these two technologies are available at all price levels. On the contrary, camera-based pure visual obstacle avoidance solutions are less common, which may be related to the lower accuracy of image recognition. One device uses TOF (Time of Flight) obstacle avoidance method, which is similar to laser distance measurement, and measures the distance of the target by calculating the time from the launch to the return of the laser. In addition, starting from the 4,000CNY price level, some devices use a solution that incorporates multiple obstacle avoidance technologies. According to the survey results, the advantages and disadvantages of IR and mechanical obstacle avoidance technology are comparable, with structured light having a slight advantage. (Table 2.2) For example, compared to structured light sensors, IR and mechanical collision sensors can only provide two states, collided and not collided, and are not as rich in data as structured light sensors. However, the price of these two kinds of sensors is lower, so they are more widely used in lower-priced products. On the other hand, as the advantage, structured light

sensors are more accurate in detection and provide 3D information about obstacles and can complement LiDAR, while they are more expensive and less durable than IR or mechanical collision sensors because of a more complex internal structure as the disadvantage. Therefore, structured light sensors are more commonly used in expensive products.

	Advantage	Disadvantage
IR collision sensor	Cheap, simple structure, not easily damaged	Not rich in data, only has collided and not collided states
Mechanical collision sensor	Cheapest, simple structure, not easily damaged	Not rich in data, only has collided and not collided states, not as durable as IR collision sensor
Structured Light sensor	Rich in data, provide 3D information about obstacles, can complement LiDAR	Expensive, more complex internal structure and not that much durable

Table 2.2 Comparation of sensors

2.1.3. Self-charging Method

Currently, self-charging solutions for robots include two categories, contact charging and non-contact charging. Among them, contact charging has the longest history of research and more studied results. For example, Roufas et al.[5] installed four infrared light-emitting diodes on the robot and two infrared receivers on the docking base to monitor the robot's motion in six degrees of freedom and calculated and transformed these pose parameters by least squares, finally achieved the automatic docking function of the robot to the base. Minten et al.[6] designed a docking based on the principle of machine vision method, as shown in Figure 2.6. This method takes advantage of the property that image pixels are linearly related to distance. The distance relationship between the robot and the base is determined by counting the percentage of color areas to the whole screen. In addition, because two colors are used, the effect of illumination on recognition accuracy is reduced. Silverman et al.[7] designed a conical mechanical guidance device (Figure 2.7) that guides the robot to accurately align with the base when the robot is close to the base. Cassinis et al.[8] used two light bulbs to guide the robot (Figure 2.8), and when the robot approaching the charging base, a bracket slightly wider than the robot is used to limit and adjust the robot's posture to complete the docking. The front of the robot has four metal strips that contact with an interface mounted inside the bracket to achieve the electrical connection.

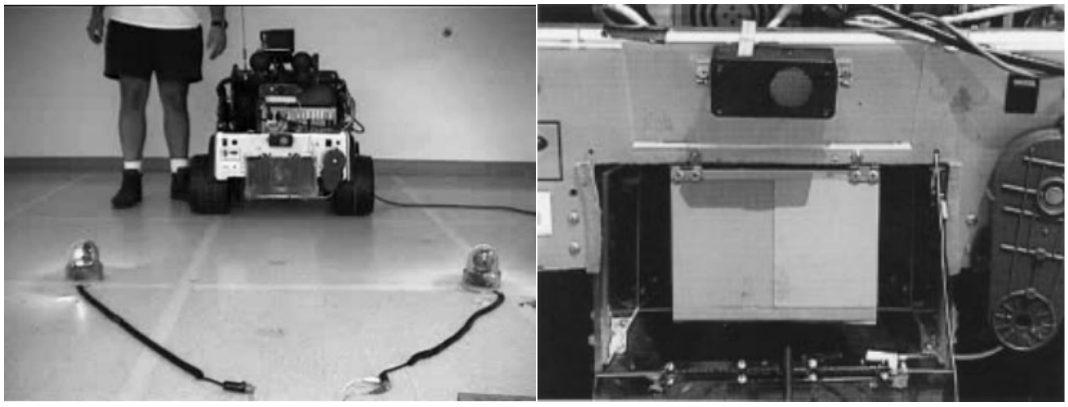


Figure 2.6 Docking devices by Minten et al.

Compared to contact charging, less research has been done on contactless charging. Common non-contact charging technologies include electromagnetic induction type, magnetic resonance coupling type and microwave transmission type. Among them, the electromagnetic induction charging principle is similar to a transformer without an iron core, where a metal coil is connected to the receiving side and the transmitting side respectively. When the transmission side outputs alternating current, the receiving side would also have voltage based on electromagnetic induction, thus realizing the transmission of current. Magnetic resonance coupling charging, on the other hand, makes use of the characteristic that the coil has a fixed frequency and amplifies the vibration amplitude of the receiving coil through electromagnetic changes, thus achieving energy conversion. Microwave transmission, however, takes advantage of the principle that radio waves can transmit energy.

In summary, contact and non-contact charging each have advantages and disadvantages. (Table 2.3) Contact charging method can support higher charging current while it has a simple structure, which not only shortens the charging time and reduced the price cost but also suitable for high-power robots. However, as a disadvantage, the contact interfaces are directly exposed to the air, and thus are prone to poor contact quality or failure due to dust accumulation and oxidation over a long period of time. Meanwhile, the exposed contact part in the air also increases the risk of electric shock. Although contactless charging can cope with the disadvantage of contact charging method, its higher cost and the disadvantage of less efficient charging than contact charging at this stage make the research in this area still relatively small. However, with the development of science and

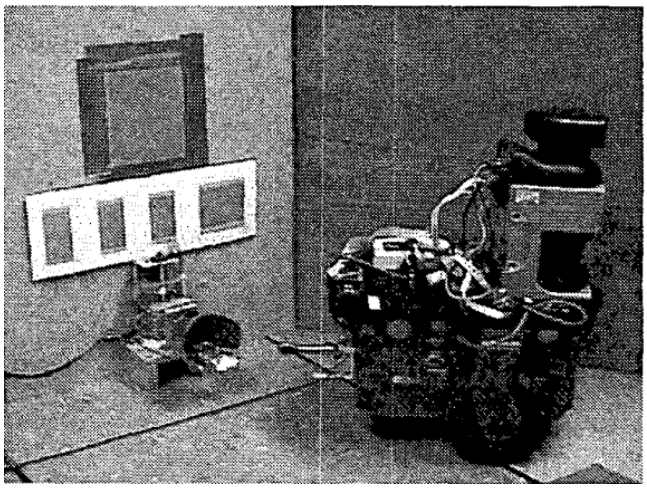


Figure 2.7 Docking devices by Silverman et al.

technology, the power consumption of robots is decreasing on the one hand, and the power of non-contact charging has been improved to a great extent on the other hand. All these changes make contactless charging seen a broader application prospect in the future.

Figure 2.8 Docking devices by Cassinis et al.

	Advantage	Disadvantage
Contact charging	Higher charging current, simple structure and cheap	Contact failure by dust accumulation and oxidation, risk of electric shock
Non-contact charging	No poor contact caused by dust and oxidation	Less efficient charging, higher price

Table 2.3 Comparation of charging methods

2.2 Affiliated Knowledge

2.2.1. PID Controls

PID control is a control method with a feedback mechanism that dynamically controls the output according to the feedback effect. The control method consists of three components, Proportional, Integral and Derivative, each of which is regulated by the parameters P, I, D (Figure 2.9) [9]. These three parameters correspond to the present error, the accumulated past error, and the future error, respectively. By setting the above three parameters reasonably, the computer can adjust the control amount according to each control result and finally achieve a constant output result. Take motor control as an example, using PID technology, it can make different motors maintain the same constant speed under different loads. It should be noted that due to the influence of manufacturing and assembly process, there may be individual differences in the same model of equipment, so the PID parameters between them usually cannot be directly followed.

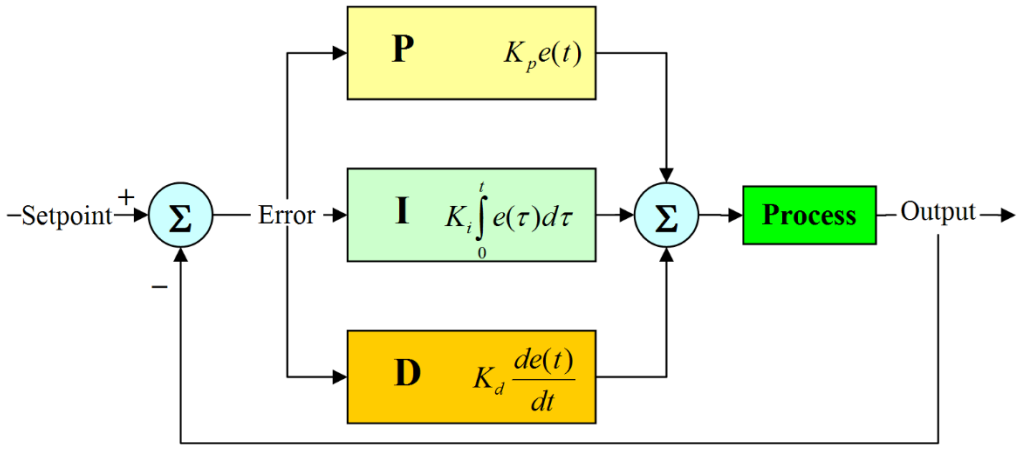


Figure 2.9 PID controller

2.2.2. Simultaneous Localization and Mapping

Simultaneous Localization and Mapping or SLAM, was first proposed by R.C. Smith and P. Cheeseman et al. in 1986 [10]. It is characterized by the fact that it enables the identification of one's own position while mapping the environment. To achieve this goal, SLAM techniques usually include several parts such as environment perception, filtering, localization, and map construction. Each part in turn includes a variety of possible technical routes. The devices currently used for SLAM environment sensing are usually of two types: LiDAR and binocular cameras. LiDAR is highly accurate, but expensive. Binocular cameras are cheaper, but less accurate than LiDAR and more susceptible to

ambient lighting. In order to improve recognition accuracy, SLAM based on multi-sensor fusion such as LiDAR, Camera, and IMU is more common in practical applications.

2.2.3. Inertial Measurement Unit

Inertial Measurement Unit or IMU, which is a device that measures the three-axis attitude angle or angular rate and acceleration of an object. If we measure the attitude, acceleration and angular rate of an object with a certain period, and accumulate these results, we can know the direction and moving distance of the object. Therefore, the reasonable use of IMU would be working to locate and track the object without relying on other devices. Inertial measurement unit is divided into two types: mechanical and electronic. Mechanical type is large in size, but is not susceptible to electromagnetic interference, and is commonly used in aviation and navigation. Electronic types are similar in appearance to ordinary silicon-based chips, small in size and low in energy consumption, and are suitable for use in portable devices.

2.2.4. LiDAR

LiDAR, or Laser Radar, is a device that uses a laser to scan the environment and map the environment based on the characteristic that the laser produces different reflections on different shaped objects. There are two categories of LiDAR: 3-D and 2-D. In principle, 3-D LiDAR is a superposition of several 2-D LiDARs. Therefore, the manufacturing cost of 3D LiDAR is much higher than that of 2D LiDAR. In terms of shape, there are solid-state LiDAR (vertical cavity, or VCSEL) and rotating LiDAR. Solid-state LiDAR is low cost and low power consumption, but short scanning range, while rotating LiDAR is expensive, but better in use. In terms of ranging principle, it is further divided into two categories: triangulation method (or laser direct structuring, LDS) and time-of-flight (TOF) method. In the triangulation method, the laser transmitter and CMOS receiver are separated by a certain distance, and the laser continuously emits laser light at a certain angle. Depending on the distance of the obstacle, the reflected light will fall on different positions of CMOS, according to which the light reflection angle can be calculated. Since the distance between the emitter and the receiver, as well as the emission angle and reflection angle of the light are known, the distance between the LiDAR and the obstacle can be calculated using the angle-edge-angle relationship. In the time-of-flight method, the laser sends a pulse laser at a certain time interval, and the receiver receives the laser. After calculating the time difference between the laser emission and reception, the time of laser flight in the air can be obtained, and the distance between the measurement target and the LiDAR can be obtained by combining the speed of light.

2.2.5. AR-Tags

AR-Tags are a kind of specially designed QR codes (Figure 2.10), and after attaching the tag to a target object, a monocular camera can be used to determine the pose and distance of the target object. The principle is that the same tag looks larger when it is close to the camera and looks smaller when it is farther away from the camera. When the pose of the tag changes, the appearance of the tag in the camera frame will be deformed. Because this change is linear, you can use this feature to determine the attitude and distance of the tag. It should be noted that the monocular camera itself does not have the ability to determine the distance of the target object, so when using AR-Tag, the camera should first be calibrated to ensure that the estimated size of the AR-tag image observed by the camera is the same as the real size.

Figure 2.10 AR Tags

2.2.6. Object Detection

Object detection (Figure 2.11) is a technology based on machine vision as well as image processing. This technique can recognize various objects in the input image, such as human body, building, vehicle, etc., in real time. Object detection methods are usually of two types, non-neural networks and neural networks. Non-neural network methods typically use support vector machines (SVMs) to classify objects, while methods using neural network concepts are typically based on convolutional neural networks (CNN) techniques that allow the algorithm to "automatically" learn to classify target object types without defining specific features of the target. Because of the self-improving and optimizing nature of algorithms based on neural network technology, it is the mainstream technology direction.[11]

Chapter 3
System Design

3.1 Hardware Selection

In Chapter 2, we have explained several researches related to firefighting robots and also analyzed what types of technologies are used in obstacle avoidance and navigation by the mainstream home robot vacuums in the market. Based on the analysis, we found that fire rescue robots are very similar to home cleaning robots in terms of navigation and obstacle avoidance technologies, for example, in the choice of navigation technologies, most of them use LiDAR-based SLAM solutions, while using cameras to assist in observing and sensing the environment. In addition, they all use some auxiliary sensors, such as wheel speed sensors and inertial gyroscopes, to monitor the robot's running speed and pose. Based on the above characteristics, we selected a set of suitable hardwares for this experiment, which are mostly from household appliances and personal electronic devices, as shown in Table 3.1 in detail.

3.2 Software Development Environment

In this system, we used the Arduino 328P that came with the Nexus10006 as the slave computer and introduced an ARM computer, the Nvidia Jetson nano, as the master computer. For these two devices, we used different software development tools and programming languages. For the Arduino part, we used the ArduinoCC IDE software provided by the manufacturer, as well as the C programming language required for that software. For the Nvidia Jetson nano part, we first installed the Ubuntu 20.04 operating system for it. On the Ubuntu operating system, we installed the ROS Noetic software library. Using the ROS Noetic library, we developed the robot's various functions in both C++ and Python. To facilitate the compilation of the program, we used another PC with ubuntu 20.04 and installed the Visual code cross-compiler tool for it.

3.3 Hardware and Software Structures

3.3.1. Hardware Structure

Figure 3.1 shows the connection relationships of all hardware used in our research. The robot body has three wheels, every wheel was driven by a motor, and an encoder was used for monitoring the speed of the wheel. These motors and encoders are connected by wire to the Arduino. Besides, two voltage sensors, responsible for monitoring the status of battery and charging connector respectively, are also connected to the Arduino by wire. The jetson nano was the central computer, responsible for data process and main program running. To achieve the navigation and target recognizing function, we installed a camera, an inertial measurement unit (IMU) and a rotating LiDAR for the robot. These devices are connected to the Jetson nano directly by USB cable. To achieve the automatic charging function, In terms of self-charging function, in order to avoid the poor contact due to contact oxidation, the electromagnetic contactless charging system has been introduced.

Figure 2.1 Object detection

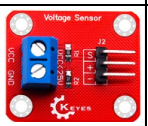
Robot body	Nexus 10006		3WD Omni wheels with 3 encoders and an Arduino 328P single chip computer
Mono camera	ELP manual variable focus FHD		USB web camera, with FHD CMOS and 2.8-12mm manual variable focus lens
Lidar	SLAMTEC A1M8		LDS type Lidar, 12m range, 360 degree omnidirectional, 5.5Hz scanning frequency
IMU	HIPNUC HI229DK		9-axis electronic gyroscope with magnetometer
Main computer	Nvidia Jetson Nano 4GB		Quad-core ARM A57 CPU@1.43GHz, 4GB LPDDR4 RAM, 128 core maxwell GPU
DC-DC Adaptor	HOUDEXINI DC-DC Adaptor		Input: DC 8-55V Output: DC 1-36V, 15A, 200W
Power charger	Bellnix BWS50-28s1R5		Input: AC85V-AC264V@50/60Hz, Output: DC28.7V, 1.5A
Voltage sensor	Keyes Voltage Sensor		Measuring range: DC 0-25V, compatible with Arduino
Battery	TACKLIFE P16		150W portable power station, 167Wh

Table 3.1 Hardware list

3.3.2. Software Structure

Figure 3.2 shows the software structure of the robot. Data related to some basic sensors, such as motors and encoders were transferred to the Arduino MCU at first, and then to the central computer, Jetson nano. Other sensors which have USB ports, such as camera and LiDAR, were connected directly to the central computer, and their data would be collected directly by central computer. On the central computer, the Ubuntu 20.04 and ROS library has been installed. By using ROS library, we create our function packages. Our robot mainly has five function packages, they were responsible for navigation, image recognition, sensors driving, robot coordinate transformation and other necessary functions respectively.

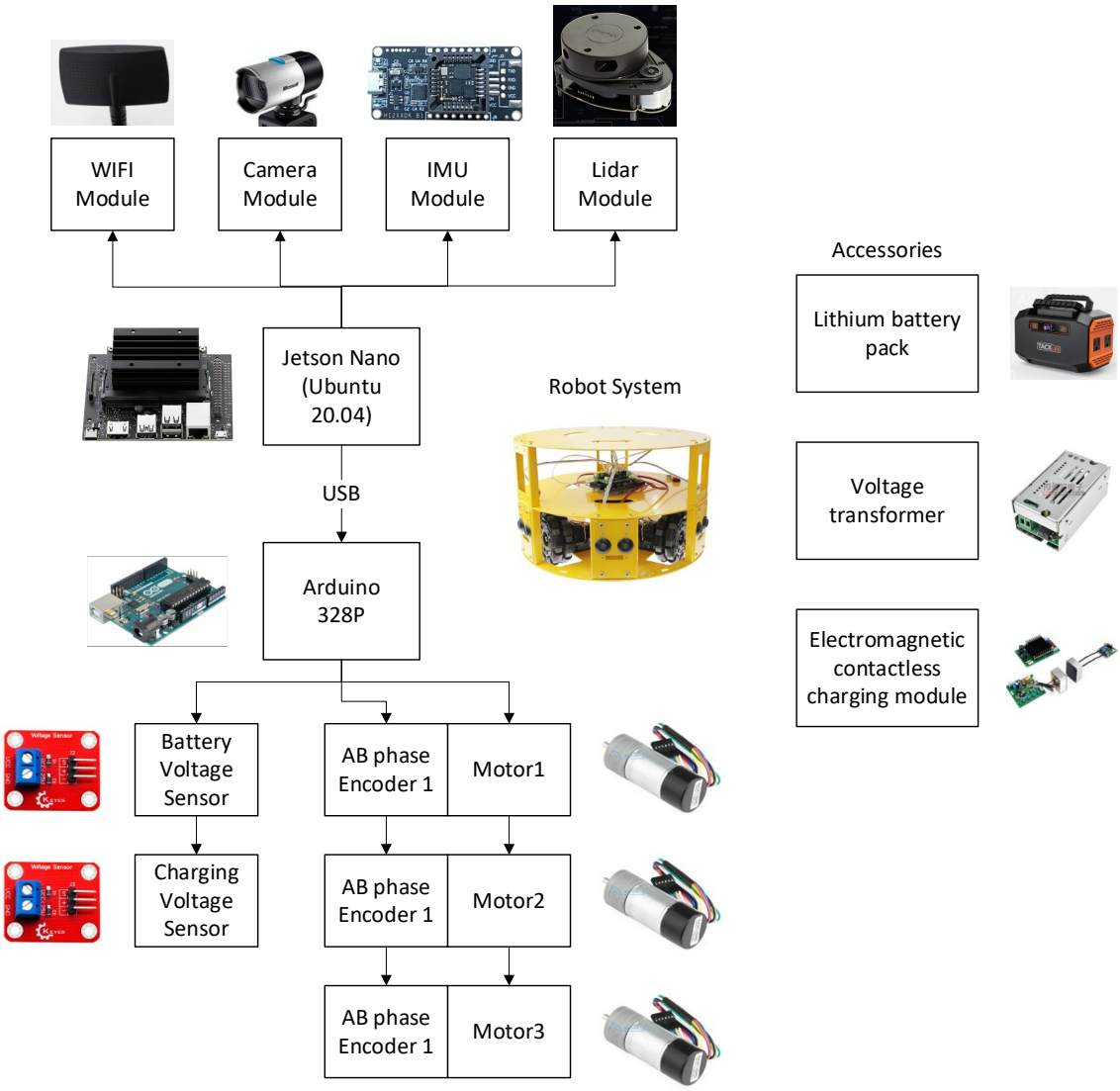


Figure 3.2 Hardware structure

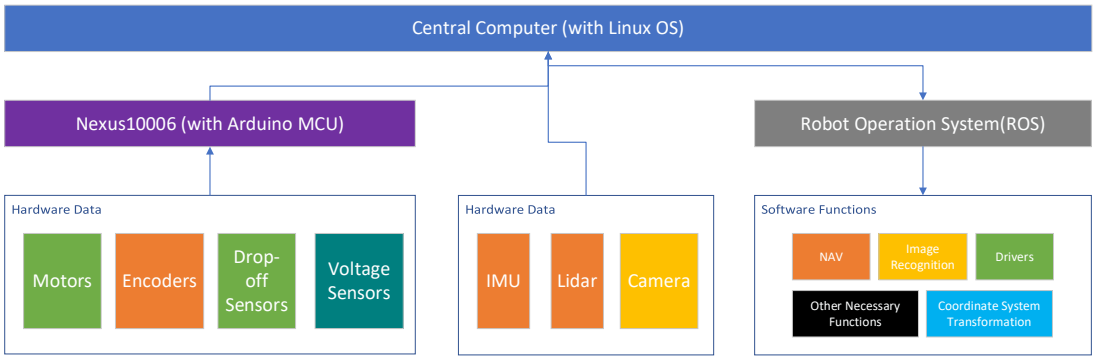


Figure 3.3 Software structure

Chapter 4
System Implementation

4.1 Experimental Method

In our experiments, we used the robot proposed in this paper with a DIY charging base and a Linux PC for monitoring the robot's running status. All experiments were conducted in the corridor of the 4th floor of the JAIST Knowledge Science Building III and in the K45 research room. The experiments included moving function experiment, self-charging experiments, SLAM mapping and navigation experiments and object detection experiments.

4.2 Experimental Procedure

4.2.1. Moving Function Experiment

In this study, since the ROS software library does not support 3WD robots, we need to develop a motor driver program for the 3WD robot. This driver program is divided into two parts, the first part is for the Arduino microcontroller that directly controls each motor, and the second part is for the Jetson nano that controls the moving direction and speed of the robot body. Therefore, the first step is to determine the speed relationship of each wheel when the robot is running according to the 3WD robot dynamics equation. The formula is shown as below.

$$\begin{bmatrix} v_A \\ v_B \\ v_C \end{bmatrix} = \begin{bmatrix} -\sin\left(\theta + \frac{\pi}{6}\right) & \cos\left(\theta + \frac{\pi}{6}\right) & R \\ -\sin\left(\theta - \frac{\pi}{6}\right) & -\cos\left(\theta - \frac{\pi}{6}\right) & R \\ \cos\theta & sin\theta & R \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ \omega \end{bmatrix}$$

Where v_A, v_B and v_C are the right wheel, left wheel and rear wheel speeds in the forward direction of the robot, respectively. v_x, v_y and ω are the longitudinal movement speed, lateral movement speed and rotation speed of the robot body, respectively. The specific correspondence is shown in Figure 4.1.

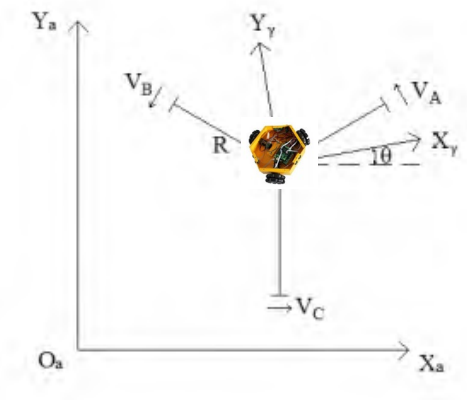


Figure 4.1 3WD robot kinematics

Using the above theory, we developed driver program for the 3WD robot so that it can be used with the ROS software library. In addition, to avoid unstable speed of the wheels due to the change of weight and resistance, we introduced a PID algorithm for the control of the wheels. Since the motors are not consistent enough and there are more significant differences in rotational resistance, we set different PID parameter values for each wheel in the code to compensate for this drawback of inconsistent physical characteristics of the motor. In this experiment, we used a PID setup with four parameters: the proportional parameter P, the integral parameter I, the differential parameter D and the adjustment multiplier O.

In the first experiment, we set the integral parameter I, the differential parameter D to 0, and the adjustment multiplier O to 50 by default, and only the parameter P was adjusted. Because we did not know what the appropriate P was, we randomly chose a value and set the parameter P to 20. At this point, the same PID parameters were temporarily set for all three motors. We send the control signal of speed 50 to the motors, and the result is shown in Figure 4.2. Where the horizontal axis represents the time and the vertical axis represents the speed.

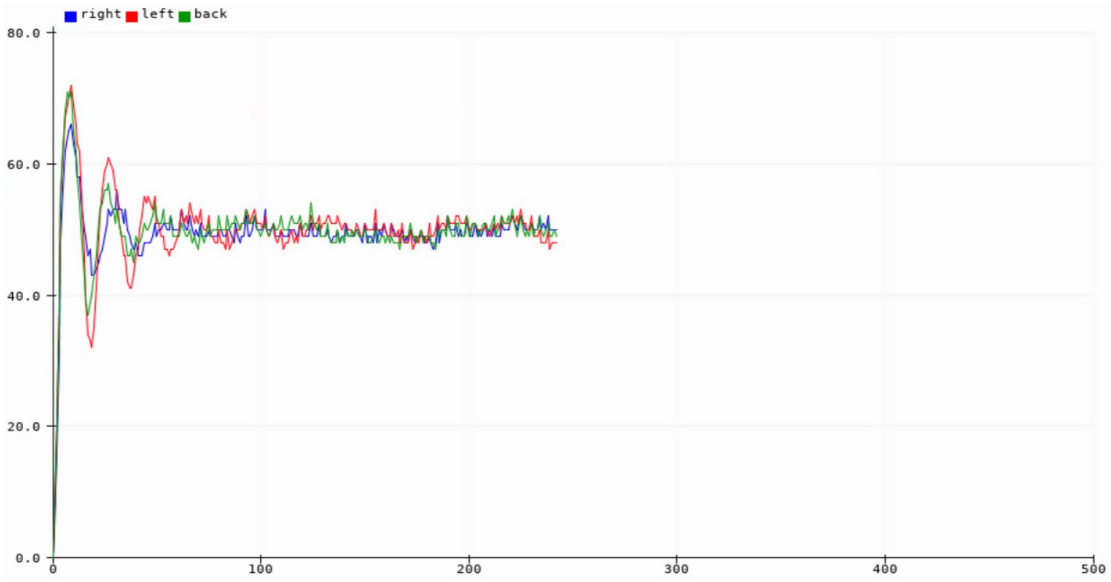


Figure 4.2 First experiment of PID

From Figure 4.2, it can be seen that the speed of the three motors undergoes a long oscillation at the beginning, indicating that the value of P is large and should be reduced further. Then we try to reduce the value of P by half, set to 10, and other parameters remain unchanged. The control command of speed 50 was still send to the motors, and the results were obtained as shown in Figure 4.3.

After the P-value is halved to 10, it can be seen that the speed oscillation of the left motor (red line) is still very obvious, the right motor (blue line) takes the shortest time to reach a stable speed, and the rear motor (green line) has a low-speed phenomenon at the beginning. For these phenomena, we believe that the P-value of the left motor is not yet optimal, while the low-speed of the rear motor can be compensated by the integral parameter I and the differential parameter D.

Therefore, we further adjust the P-value of the left motor and keep the parameters of the right motor and the rear motor unchanged, and the results are shown in Figure 4.4

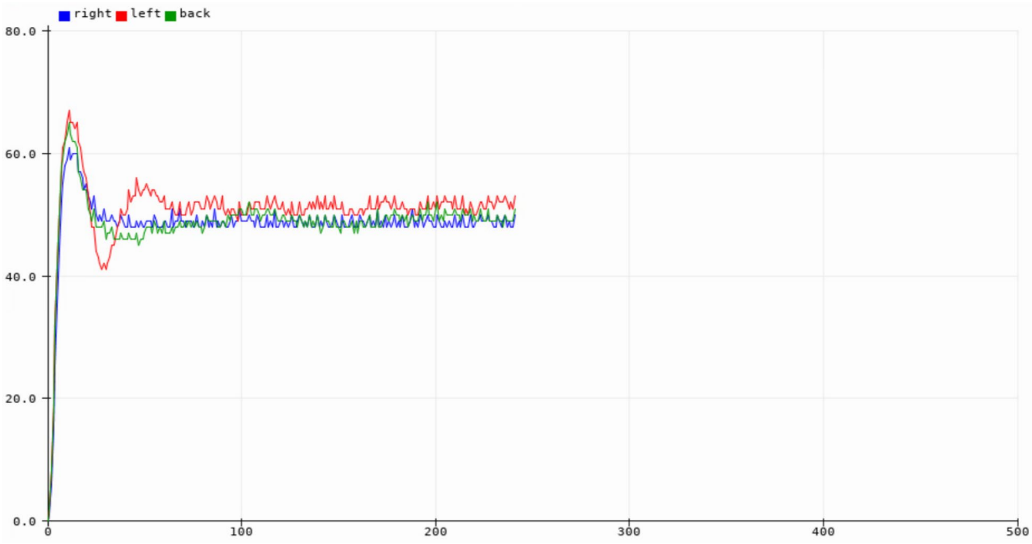


Figure 4.3 Reduce the value of P by half

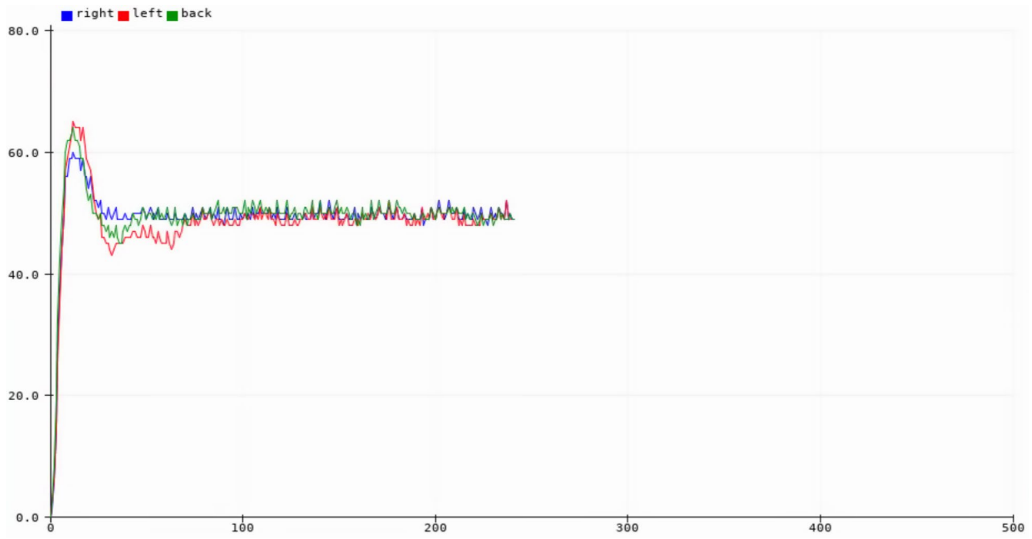


Figure 4.4 Further reduce the left motor's P-value

After further adjusting the P-value of the left motor, we can see that the oscillation of the left motor disappears, but similar to the case of the rear motor, the left motor also shows the phenomenon of low-speed. At this point, we believe that the adjustment of the P-values for the three motors is almost complete. In the next step, we will adjust the D-values for each motor in order to alleviate the under-speed phenomenon. We first set the same, smaller D value for all three motors, and then send a control signal to the motors for speed 50, and the observed results are shown in Figure 4.5.

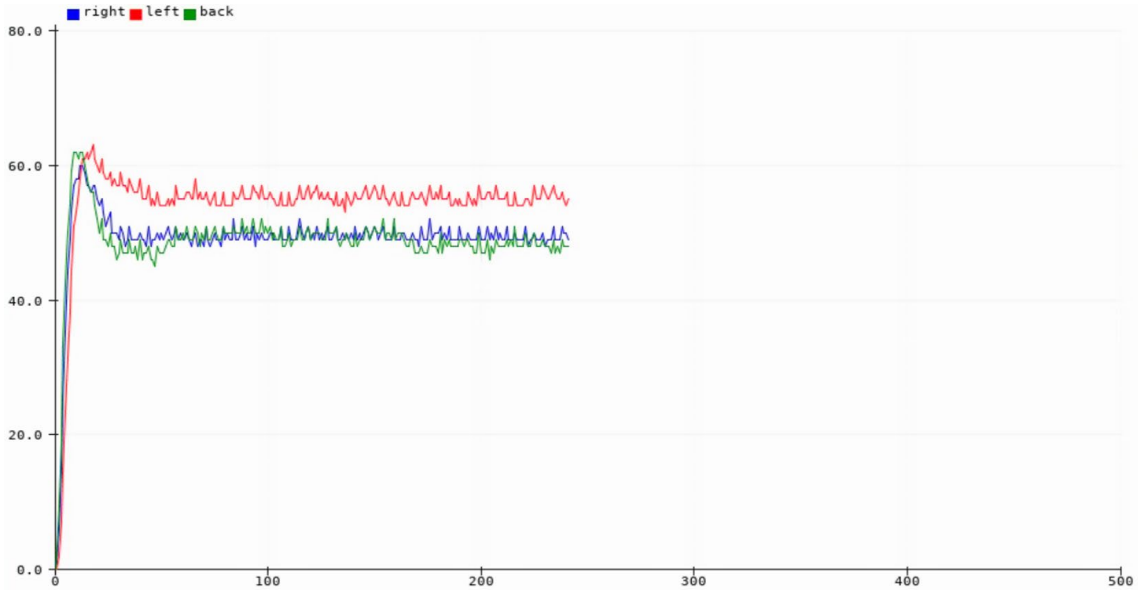


Figure 4.5 First adjustment of D-values for all motors

From the above graph we observe that although the low-speed of the left motor is solved, the constant speed of the motor is much higher than the target value of 50, so we think that this is most likely due to the still high P value of the left motor. Therefore, we further reduced the P-value of the left motor. In addition, we observed that the speed of the rear motor was not stable enough, which may also be due to the interaction between the P and D parameters. Therefore, in the next experiments, we tried hundreds of different combinations of P-parameters, D-parameters and O-parameters for the three motors, and finally found a relatively reasonable combination of parameters, and the running result are shown in Figure 4.6. It can be seen that after a short period of deceleration, the three motors reach the same speed at almost the same time and the speed remains relatively constant in the following time.

It is worth noting that at this time we have only adjusted the P, D, O parameters, and the parameter I will produce what effect we have not yet experimented. Therefore, we first try to set a value of I for the rear motor, and observe what phenomenon will occur. The result is shown in Figure 4.7.

It can be seen that when the integration parameter I is set for the rear motor, there is a long-time oscillation and the speed is never constant. Therefore, we believe that for the Nexus 10006 robot used in the experiment, only P, D and O parameters are needed.

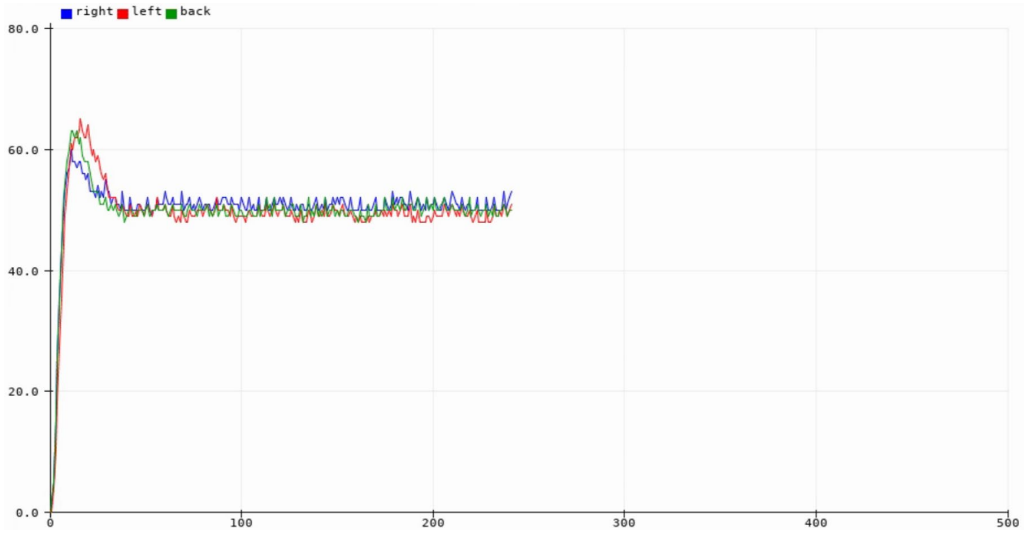


Figure 4.6 An optimal parament set for three motors

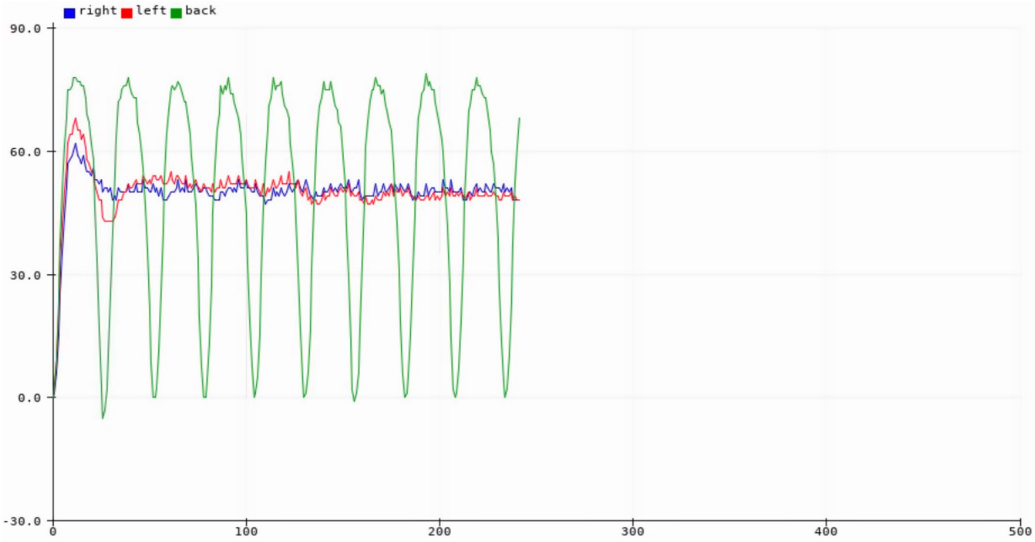


Figure 4.7 Set the parameter I for rear motor

4.2.2. 3-D Modeling and Coordinate Transformation Setup

According to the ROS development documentation, to develop a robot program with the ROS software library, it is necessary to set a uniform coordinate transformation relationship for all sensors installed on the robot, so that the data collected by all sensors will have the same reference point, or origin point. This is because the camera, odometer, gyroscope, and other devices are installed at different locations on the robot and have position deviations from each other, which will cause the collected data to have their respective locations as the origin. Therefore, these data cannot be directly used for the robot's navigation without coordination transformation.

To solve this problem, we built a three-dimensional coordinate transformation model for the robot and calibrated the positions of the sensors in the model according to the real dimensions and relative positions of the robot and the sensors. Using this model, together with the relevant functions of the ROS library, we can unify the data origin of all sensors with the position reference point of the robot. Figure 4.8 shows the comparison between the physical photograph of the robot and the model.

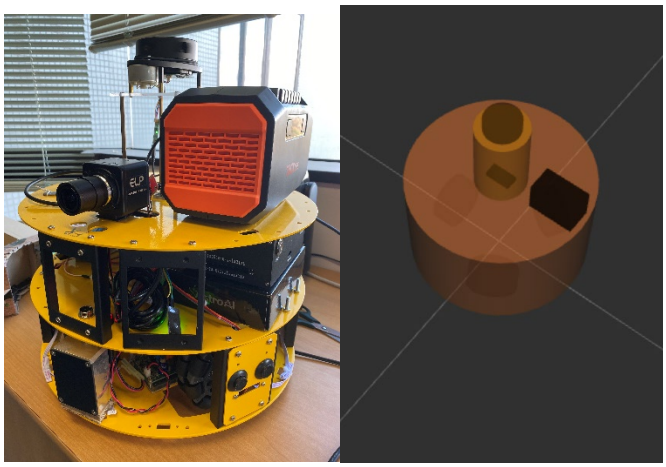


Figure 4.8 The real robot and the 3D model

To simplify the development, we use a large cylinder to represent the main structure of the robot and a small cylinder to represent the LIDAR mount. In the 3D model, the black rectangle located in front of the large cylinder is the camera, and the black rectangle at the bottom of the small cylinder is the IMU. The black pie-shaped structure at the top of the small cylinder is the LiDAR. Based on this model, we calibrated the position relationships for the three wheels, various sensors and the body structure of the robot. The results are shown in Figure 4.9.

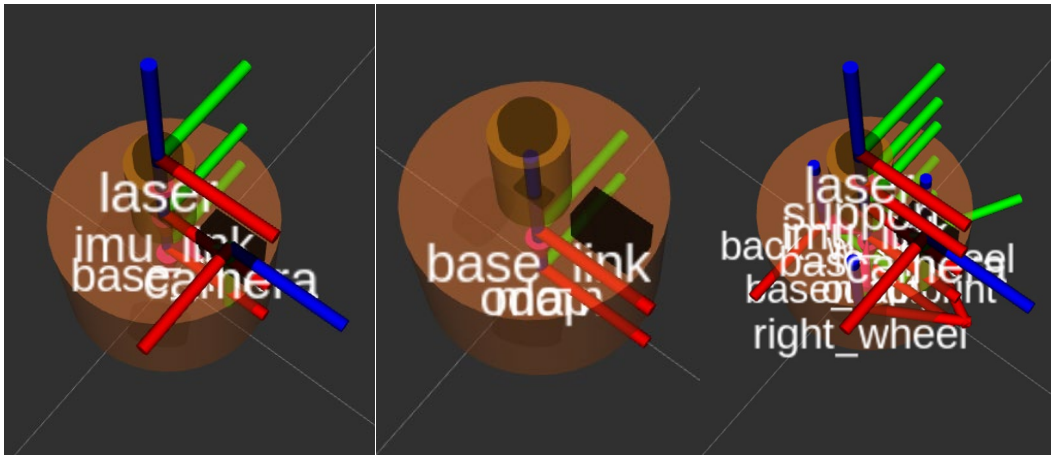


Figure 4.9 Coordinate transformation setup

Using this robot model, we unified the data of the robot body, each sensor, the odometer map and the navigation map, and the coordinate transformation tree has been setup. When the program running, the coordinate transformation of the data flow in the ROS software library is shown in Figure 4.10. Where Map is the coordinates of the global map generated by LiDAR and the odometer (odom) is the coordinates of the robot's positional data calculated with the wheel speed sensor as well as the IMU. From base_footprint downward, we can see that the data from the three wheels, the LiDAR, the IMU, and the camera are converted to base_link, which represents the center of gravity of the robot body, and also be seen as the reference point. The base_footprint is the projection of the robot's center of gravity on the ground, which coincides with the initial origin of the map and odometer coordinate systems. By using this coordinate transformation relationship, the data from each sensor of the robot can be used for detecting the posture and ranging, eventually for SLAM navigation, automatic docking and charging, and target recognition.

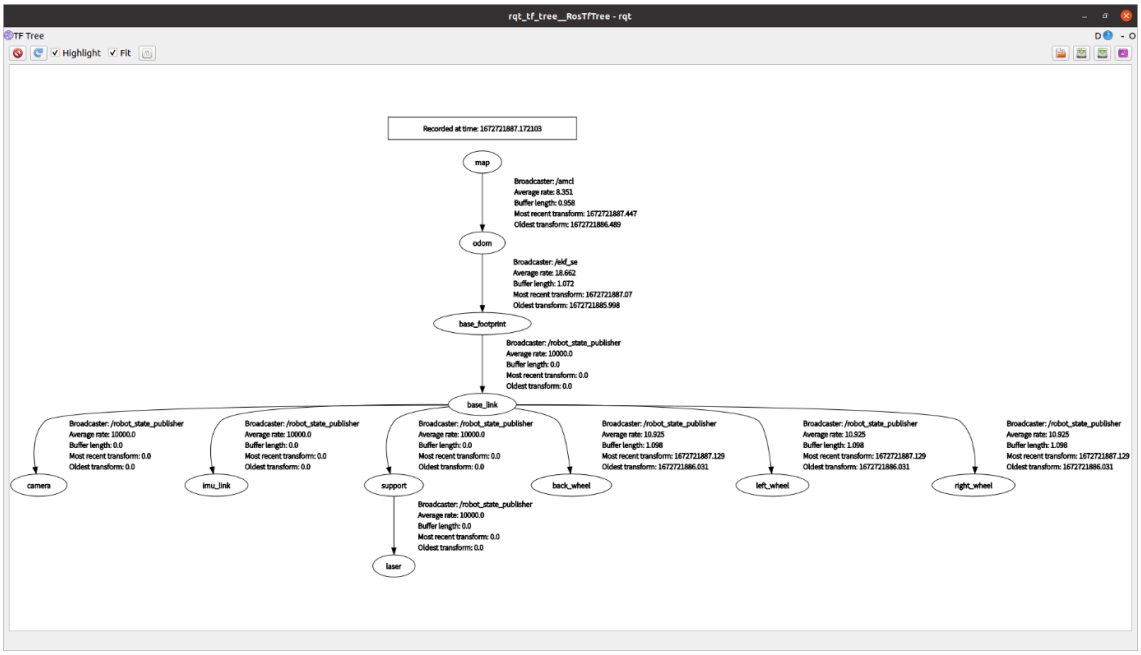


Figure 4.10 Coordinate transformation tree

It is worth mentioning that the ROS software library is not the same as the usual class-and-instance-based software library. In the ROS software library, to facilitate distributed computing, the passing of data does not rely on variables and values, but uses a medium called topics. The graphical tool that comes with ROS allows us to view the flow of data from various sensors while the robot is running, which was shown as Figure 4.11.

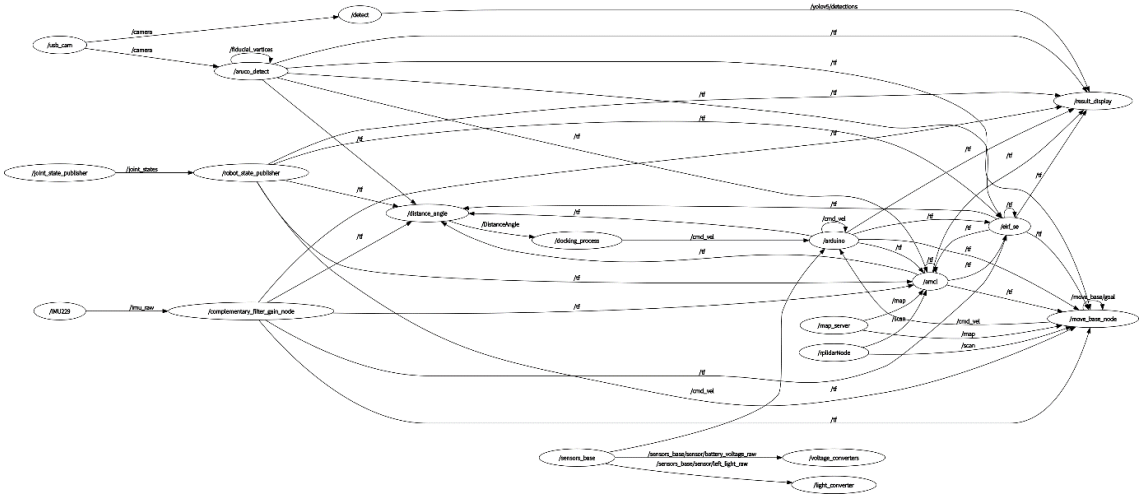


Figure 4.11 Data flow

4.2.3. SLAM Mapping and Navigation

Using LiDAR, wheel speed sensors and IMU, we can use the SLAM function of the ROS software library for map generation and navigation. It is important to note that the standard ROS software library usually only needs to obtain the pose and speed information from the wheel speed sensor and does not need to use the IMU sensor. However, in this study, we found that when the wheels skid, this kind of positional sensing method can present a large amount of erroneous data, which causes the robot to lose its direction while driving. Therefore, we introduced a fusion method of IMU and wheel speed sensors. Based on the sensor fusion method, we used LiDAR to build a map for the experimental site, as shown in Figure 4.12. In which, the LiDAR-generated map is shown on the left and the real map of the experimental environment is shown on the right. Obviously, the LiDAR-generated image is very close to the real situation.

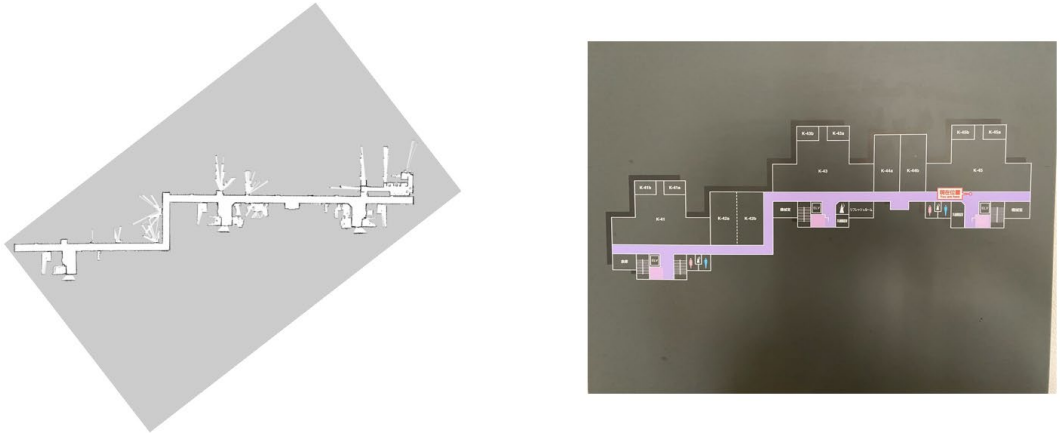


Figure 4.12 LiDAR map and real map

Using this map, we experimented with the autonomous navigation function of the robot. It should be noted that because of the weak processor performance of JetsonNano, the default transform_torlerance parameter does not allow the program to obtain valid coordinate transformation data within the specified time, so it is necessary to increase this parameter. For the costmap-related parameters, we set the inflation_radius parameter for the robot to avoid getting stuck at special locations such as walls and corners. More specifically, we set the global inflation_radius to be larger and the local inflation_radius to be smaller. In the path planning algorithm, we choose the Dijkstra algorithm for global path planning, which is a kind of shortest path search algorithm based on generalized traversal search, and can find the shortest path in the ideal state better. In the local path planning part, we use the DWA algorithm (dynamic window approach), which simulates different motion trajectories in space using multiple speeds and selects the optimal trajectory to drive the robot through the evaluation function. This function has a low computational complexity, requires less performance from the main computer, and has the property of real-time computation for obstacle avoidance. In addition, in our experiments, we found that because the LiDAR is located on the top of the robot and the rotation of the LiDAR generates vibration, which leads to a large ranging error when the robot is close to an obstacle such as a wall and cannot pass through the obstacle-free passage. To solve this problem, we improved the behaves when robot encountering obstacles, allowing it to “try to squeeze” through narrow spaces at a lower speed. Fig. 4-13 to Fig. 4-15 show how the robot automatically navigates to the target location.

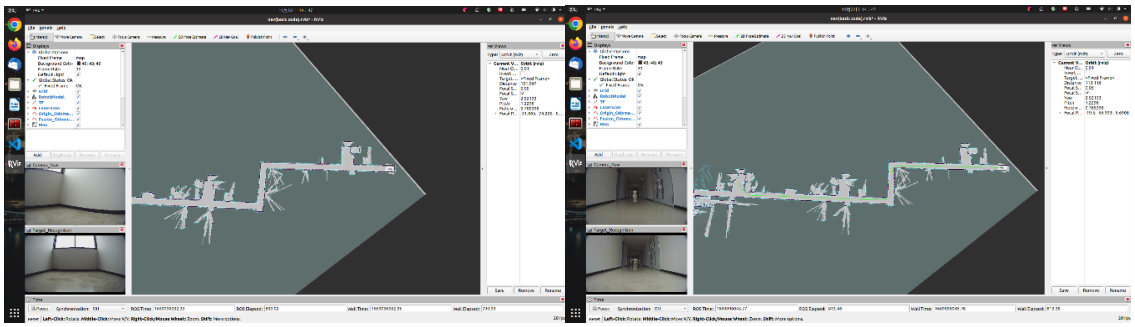


Figure 4.13 Create the global path

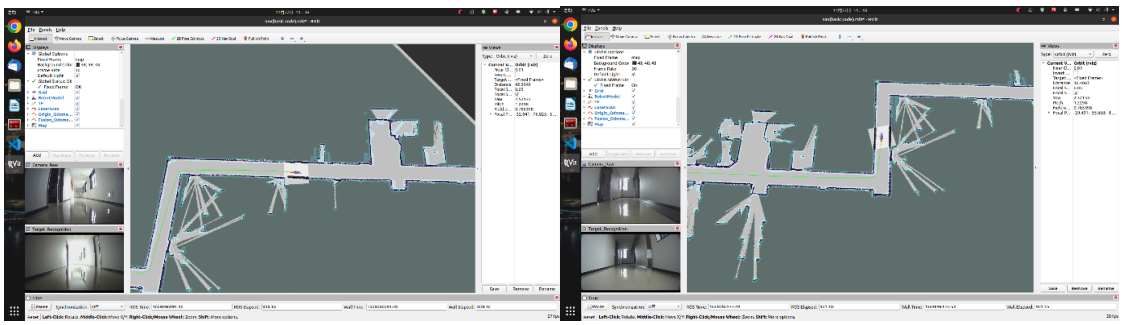


Figure 4.14 Obstacle avoidance and turning

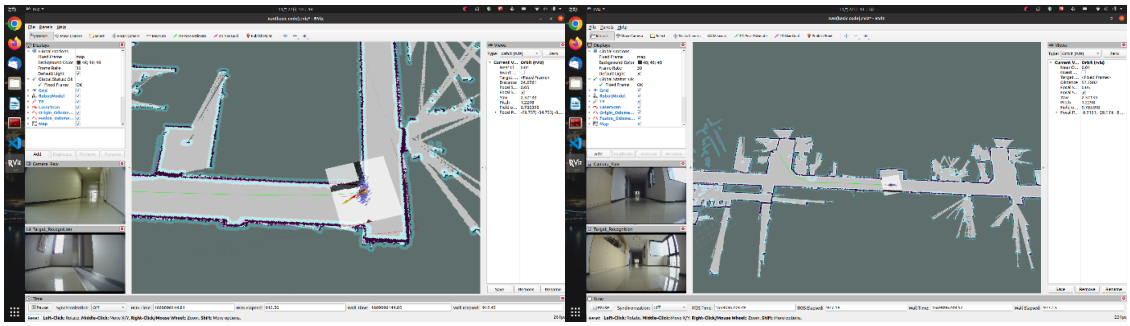


Figure 4.15 Approaching the destination

4.2.4. AR-Tag Based Self-charging Experiment

In our proposal for automatic docking charging, we use AR-Tag based on machine vision to locate the position and pose of the charging base. This is a method that uses a monocular camera to identify the distance as well as the pose of the tag using the size and the degree of deformation of the AR-Tag tag in the frame. Therefore, the camera needs to be calibrated before use. In this thesis, we used the camera_callibration program provided by the ROS library and a calibration plate with 7x9 black and white grids to calibrate the robot's camera. The calibration screen is shown in Figure 4.16. The camera calibration is done when the grid is moved horizontally, vertically and diagonally so that the progress bars under X, Y, Size and Skew are full.

After that, in order to achieve the automatic docking and charging function between robot and the base, we made an docking base in cardboard equipped with a electromagnetic contactless charging device. This charging module can provide a maximum charging power of 43W. The module mainly has two parts: the power supply side and the receiving side, where the power supply side is placed in the center of the charging base arm, and the receiving side is installed at the lower position directly in front of the robot, as shown in Figure 4.17 and Figure 4.18.

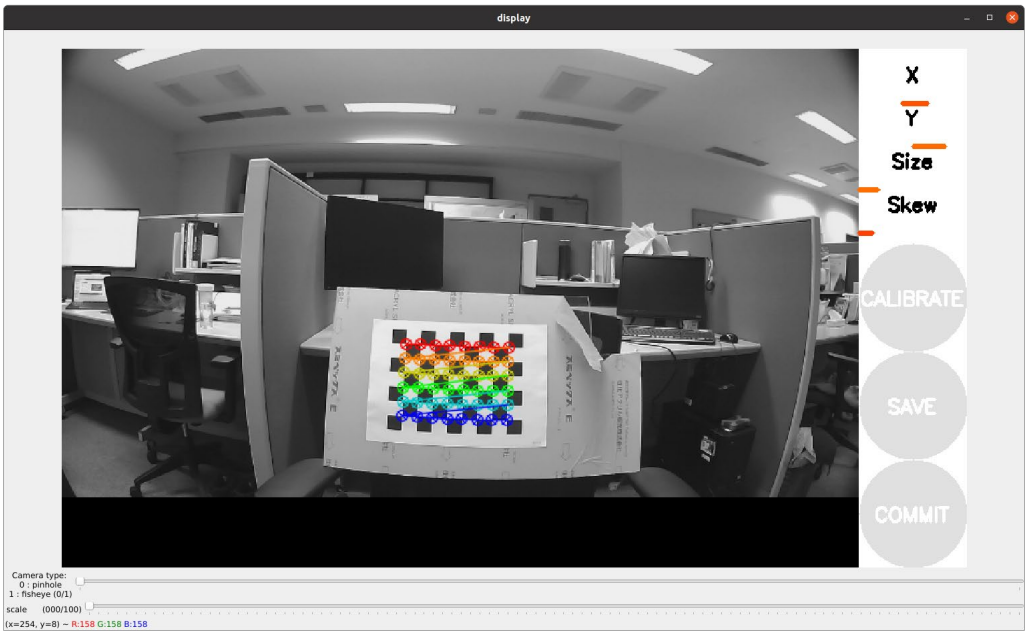


Figure 4.16 Camera callibration

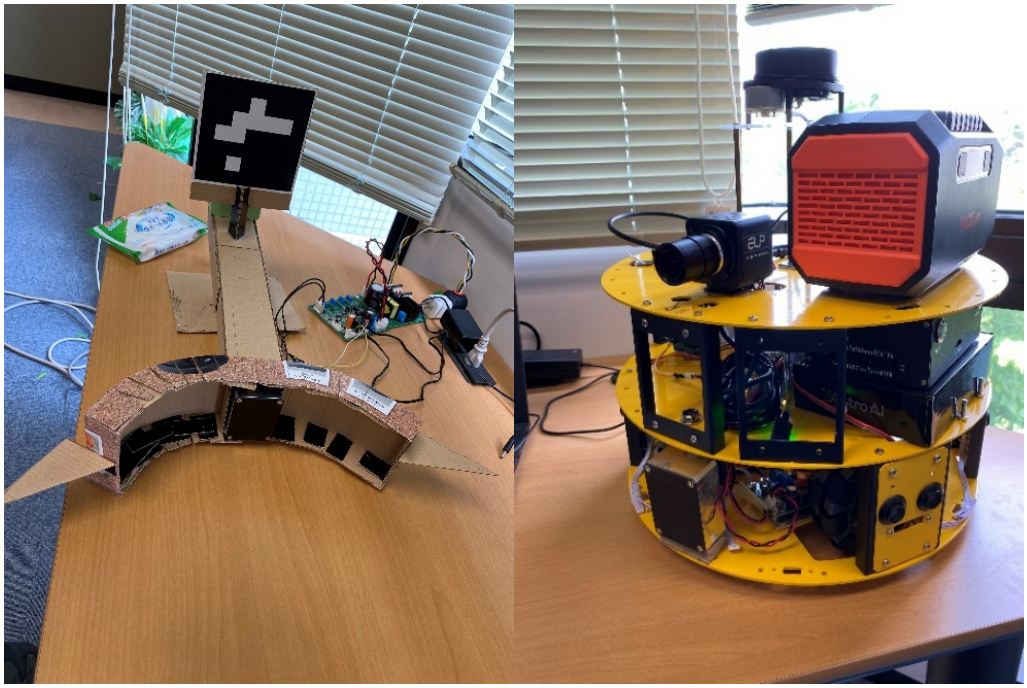


Figure 4.17 Contactless charging module on robot and charging base

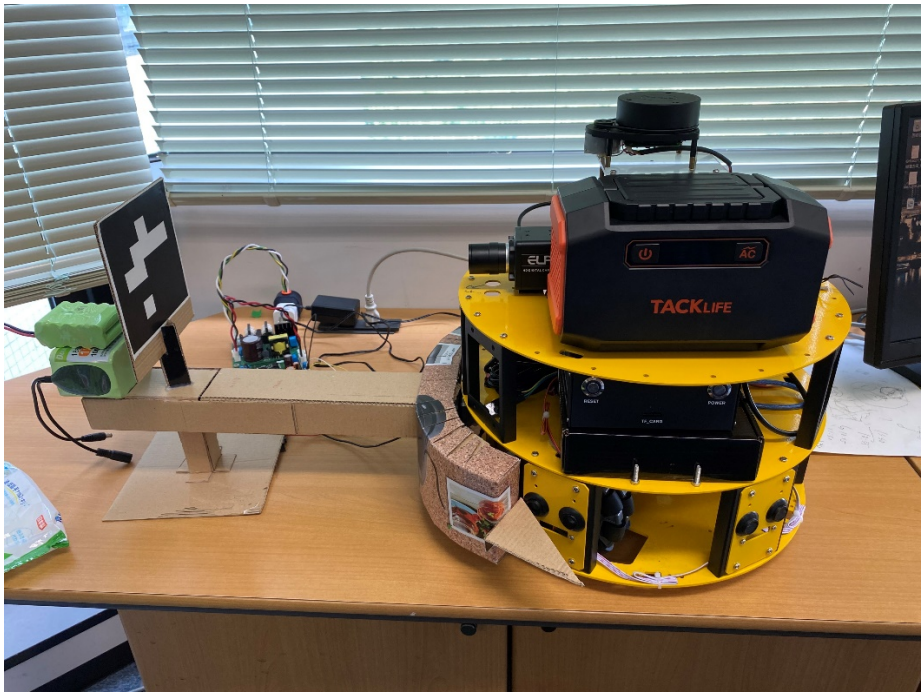


Figure 4.18 Docking demonstration

Using this docking device, we tested the robot's recognition of the pose and position of the charging base. We first placed the base with the AR-Tag in front of the robot camera and observed the robot's recognition of the AR-Tag. The results are shown in Figure 4.19.

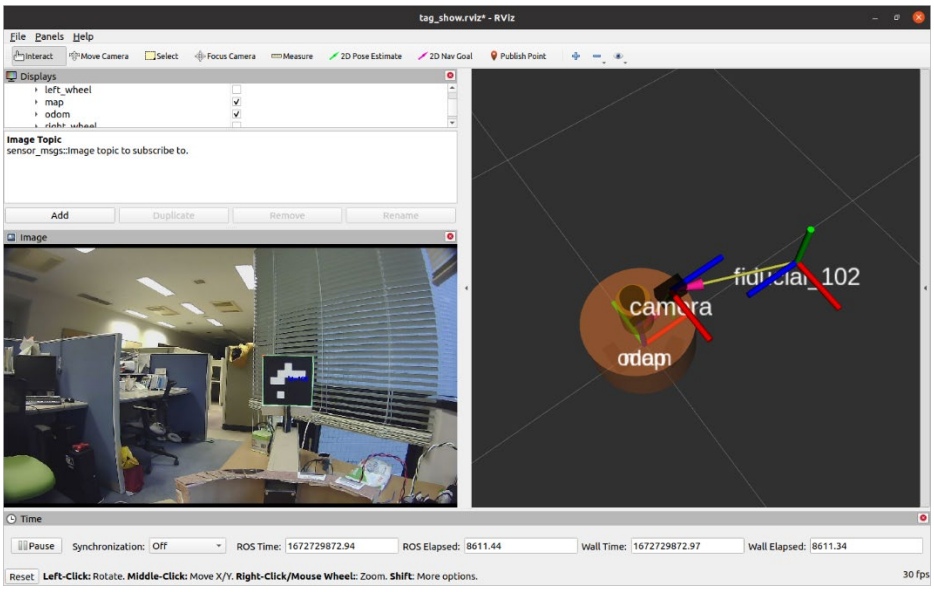


Figure 4.19 Tag recognition test result 1

As can be seen in the screenshot, in the camera screen on the left, the robot has shown the location of the tag in the screen with a green box, and the blue text in the green box shows the specific information about the tag. The distance and relative position of the

robot to the AR-Tag can also be seen in the 3D model of the robot in the right part of the screenshot, and this relative position is the same with what is shown in the left camera screen.

After that, we move the robot towards the other direction, but keep the tag still within the camera's view, and the result is shown in Figure 4.20. As can be seen, in the camera image on the left side of the screenshot, the robot is facing out of the window and the AR label is located on the left side of the robot. In the 3D model of the robot on the right side of the screenshot, the AR label is also located on the left side of the robot. This indicates that the robot's recognition of the label position pose is accurate.

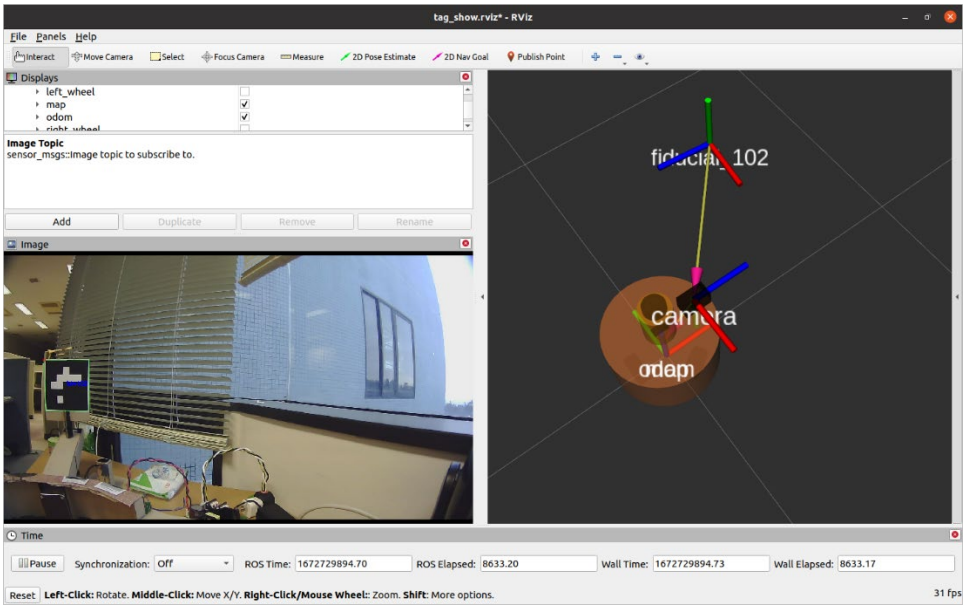


Figure 4.20 Tag recognition test result 2

Based on the above tests, we made a docking test. The experiment video record are shown as Figure 4.21 and Figure 4.22.

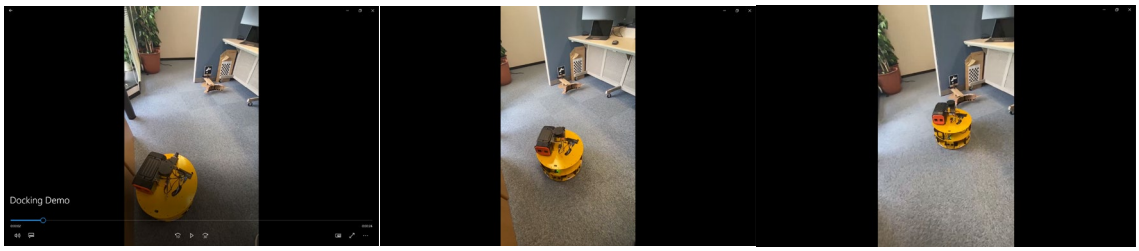


Figure 4.21 In docking progress

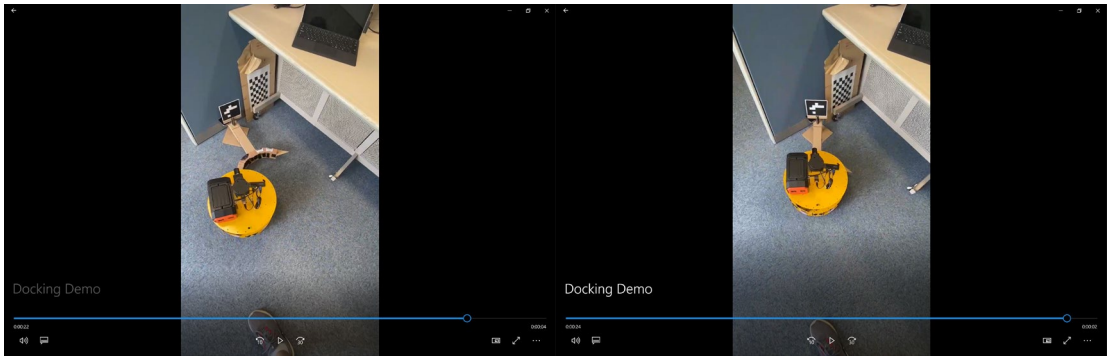


Figure 4.22 Docking finished

4.2.5. Object Detection Experiment

In this proposal, searching for trapped people in buildings and recording the location for the rescuers outside the building is a very important function. Therefore, for this function, we use a object detection model based on machine learning for identifying trapped people. For the robot developed based on the ROS library, the YOLO series object detection algorithm has many advantages such as high integration, perfect documentation, and low resource consumption, so we chose the YOLO version 5 [12], which was the mainstream in the ROS library at that time. Then, we used the automatic navigation function introduced in 4.2.2 to allow the robot to move autonomously in the experimental site, and enabled the object detection function. If an object marked as Person is recognized in the screen with an accuracy rate of 0.8 or higher, the recognition image will be automatically saved and the current location of the robot in the map will be recorded. Figure 4.23 shows a screenshot of the robot in running. At the bottom left of the screenshot, the object detection algorithm has detected the human object and its accuracy rate is 0.93. Since 0.93 is greater than 0.8, the screenshot is saved and the robot performed the correct operation as shown in Figure 4.24. The stored image is shown in Figure 4.25, and its file name is pic_-7.5_-19.8_raw.

During the autonomous moving of the robot, multiple identified objects may be encountered, so all the saved images are uploaded in real time to a folder on a remote server for use. All the recognition results in this folder are shown in Figure 4.26.

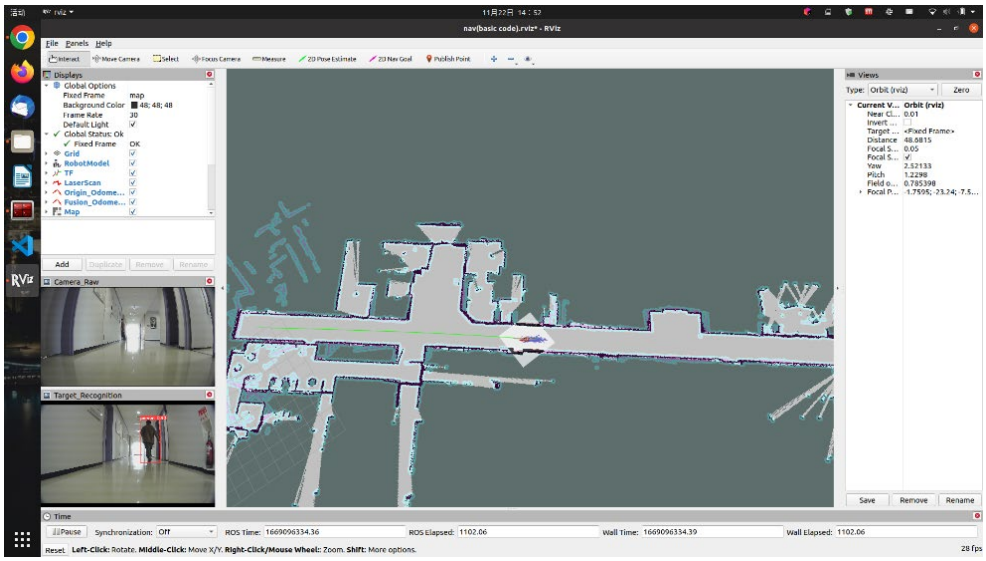


Figure 4.23 Object detection while autonomous moving

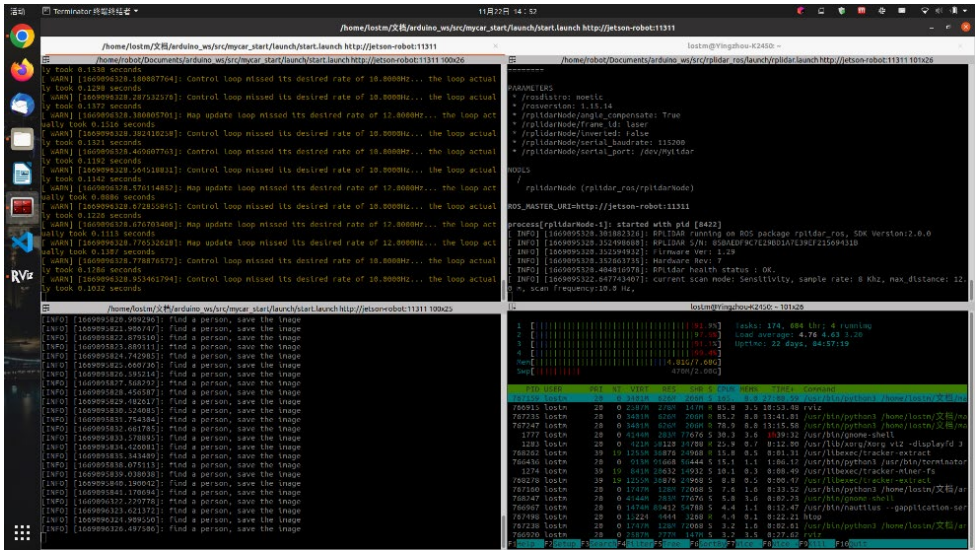


Figure 4.24 Correctly operation

Finally, considering smoke often presented in indoor fire scenes, which's may also be an important influence on the robot's recognition performance, we tested how smoke would affect the camera and the object detection algorithm by creating smoke with dry ice. In our test, we set three levels of smoke concentration, i.e., no smoke, half smoke, and full smoke states. Also, three poses were set for the test, and each pose was divided into face towards camera (face side) and back towards camera (back side). The object detection algorithm was then applied to see what would happen. Due to the limitations of the test site, the test was conducted in several rounds, and the screenshots shown in Figure 4.27 are the results of the last test. The sequence of experiments was from no smoke to full smoke, and from standing to lying postures.

Figure 4.25 Saved picture

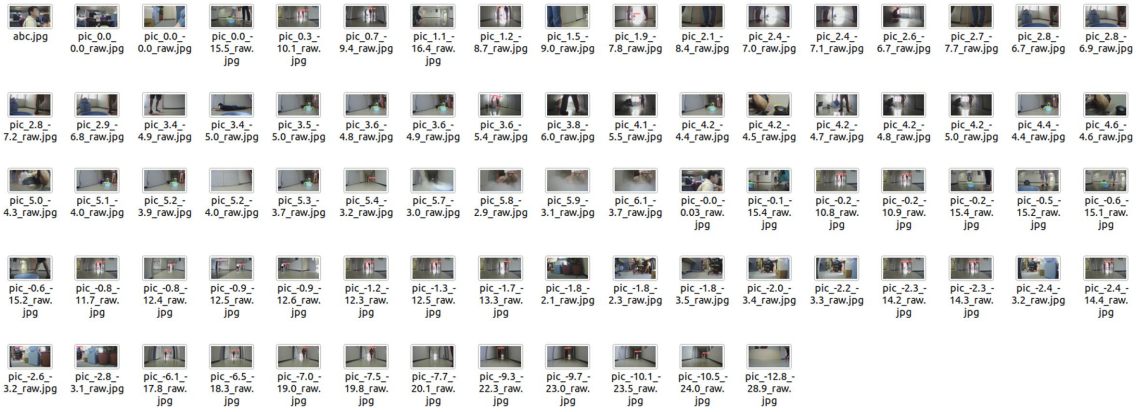


Figure 4.26 Object detection results

Figure 4.27 Smoke test

In our test, if a pose was not recognizable at lower levels of smoke, subsequent experiments were terminated. For example, if the half-smoke, standing posture, back-to-camera scenario failed, then the subsequent full-smoke, standing posture, back-to-camera scenario was not tested. In scenes where recognition fails, there is no red recognition box

in the screenshot. Scenes that were not experimented with are marked with a red circle.

	No smoke		Half cover		Full cover	
Accuracy	Face	Back	Face	Back	Face	Back
Standing	0.9	0.93	0.9	X	0.86	X
Squatting	0.92	0.91	0.91	X	X	X
Lying	0.81	X	X	X	X	X

Table 4.1 Smoke test

Test result based on Figure 4.27 and Table 4.1, Yolo_v5 has a higher recognition accurate for standing posture and a lower recognition accurate for squatting and lying posture. Meanwhile, the recognition accurate was higher on the face side (even when the person's face was not exposed) and lower on the back side. In addition, the concentration of smoke, to the extent of the human body coverage will also affect the accuracy of recognition. However, in the process of the experiment, it was found that even if the smoke covered the whole surface of the human body, if the concentration of the smoke was not thick enough and the outline of the human body could be seen, then the human body could still be recognized. The phenomenon was most pronounced when the person faced the camera.

Chapter 5
Summary and Improvement

In this thesis, we present a robotic implementation for searching trapped persons in indoor fires. Unlike conventional firefighting equipment, our robots use inexpensive parts commonly found in various household appliances, reducing costs and thus can be deployed inside buildings as part of a fixed firefighting facility before a fire occurs. As a result, these robots can be put to use more quickly when a fire breaks out, rather than having to wait to be deployed from distant locations, thus saving rescue time. In addition, this inexpensive robot design can be used in under developing areas where the cost of professional rescue robots is limited by the economic level.

For the robot proposed in this paper, we conducted experiments on PID parameter tuning, SLAM map building and navigation, AR-Tag-based self-charging, and YOLO machine-learning algorithm-based object detection. To perform the above experiments, we used the ROS software library as the running framework for the robot, and based on this, we developed software packages to achieve various functions. In addition, because the robot is equipped with multiple sensors, we designed a 3D model for the robot in order to allow these sensors to work with each other, and by calibrating the relative positions of each sensor in this model and setting up a coordinate transformation tree, we unified the data origin of each sensor on the robot backbone base_link. Finally, although the robot itself is equipped with an Arduino 328P microcontroller, its performance is not sufficient to run the various functions required in this proposal. Therefore, we installed an ARM architecture PC, Nvidia Jetson Nano, to it, and implemented the control system of the robot by using this PC as the host and the Arduino as the slave.

As the more novel part of this paper, the main points are as follows. First, the ROS software library is designed for 2-wheel differential robots and cannot be used for the 3WD robots in this paper. Therefore, we developed a motor driver for the ROS software library based on the kinematic formulation of the 3WD robot. Also, to address the problem that the use of cheap parts leads to poor consistency of the motors and the use of the same set of PID control parameters does not achieve optimal results, we set PID parameters for each set of motors separately when developing the driver, thus achieving a better control effect. In addition, unlike common drivers, the robot used in this paper

has both a JetsonNano host and an Arduino slave, so the drivers we developed are divided into host and slave, and the development languages include C, C++ and Python.

Secondly, we propose a new docking method for the automatic charging of robots. This method uses the concept of machine vision, using a monocular camera and an AR-Tag, to achieve recognition and pose confirmation of the automatic charging base. Subsequently, we use a 3D robot model with coordinate transformation function to unify the coordinate positions of the AR-Tag and the robot itself, so that the robot can clearly know the position and angle of the charging base, and finally this information allows the robot to achieve automatic docking with the charging base. On the other hand, for the power supply method, we chose the electromagnetic contactless charging module which is less common at present. The wireless charging module does not have an exposed metal contact surface, which not only avoids the risk of leakage, but also solves the problem of poor contact due to oxidation of the metal when exposed to air for a long time. This oxidation resistance is ideal for robots such as those used in this paper that require long-term deployment.

Finally, we introduced the YOLO version 5, a target recognition model based on machine learning principles, for detecting human bodies in the environment. Through smoke experiments, we found that the model works best for recognition of experimental targets in a smoke-free environment and in a standing position, but not well for experimental targets in a lying position. When smoke is present in the environment, the recognition success rate of the model decreases significantly. We speculate that, on the one hand, it is difficult for the ordinary camera we use to penetrate the smoke. On the other hand, the YOLO model may rarely use human data in lying posture for training.

Therefore, as the part that still needs to be improved, we think there are two points as follows. First, we use a 3WD chassis as the walking part of the robot, but this structure has a low ability to cross obstacles, and our robot may encounter difficulties when a fire occurs and the interior decoration material collapses causing obstacles in the path. Therefore, if this chassis can be replaced with a chassis that has a stronger ability to cross obstacles, such as a full-tracked chassis, 8X8 wheeled off-road chassis, then perhaps the situation will be greatly improved.

In addition, in the smoke experiment, we found that this proposal has a low recognition success rate for experimental scenes with smoke, non-standing posture, which is partly

due to the fact that the camera we used does not have the ability to penetrate smoke, and partly may be due to the lack of training of the YOLO version 5 model for non-standing human targets. If a thermal imaging camera that can penetrate smoke is used instead of a normal camera, and the model is trained using a dataset enhanced with non-standing human subjects, the results of the smoke experiments may be greatly improved.

Bibliography

[1] S. G. Badger, “Catastrophic Multiple-Death Fires and Explosions in the United States in 2021 (NFPA ®),” 2022.

[2] Eastern Kentucky University, “The Use of Robotics in Firefighting,” *EKU Online*, Jul. 22, 2020.

[3] A. Watanabe, H. Miura, M. Okugawa, and K. Hatanaka, “Scenario Verification for the Use of Robots in Fire-Fighting and Rescue Activities,” 2020.

[4] TECHABLE, “火災現場などで活躍するロボット「SmokeBot」はガスの有無も感知,” *TECHABLE*, Jun. 24, 2018.

[5] K. Roufas, Y. Zhang, D. Duff, and M. Yim, “Six Degree of Freedom Sensing For Docking Using IR LED Emitters and Receivers,” in *Experimental Robotics VII*, Springer Berlin Heidelberg, 2007, pp. 91–100. doi: 10.1007/3-540-45118-8_10.

[6] B. W. Minten, R. R. Murphy, J. Hyams, and M. Micire, “Low-order-complexity vision-based docking,” *IEEE Transactions on Robotics and Automation*, vol. 17, no. 6, pp. 922–930, 2001, doi: 10.1109/70.976026.

[7] M. C. Silverman, D. Nies, B. Jung, and G. S. Sukhatme, “Staying alive: A docking station for autonomous robot recharging,” *Proc IEEE Int Conf Robot Autom*, vol. 1, pp. 1050–1055, 2002, doi: 10.1109/ROBOT.2002.1013494.

[8] F. Tampalini, P. Bartolini, M. Automotive, R. Cassinis, and R. Fedrigotti, “Docking and charging system for autonomous mobile robots,” 2005. [Online]. Available: https://www.researchgate.net/publication/228800043

[9] Wikipedia contributors, “PID controller,” *Wikipedia*, Jan. 03, 2023. https://en.wikipedia.org/wiki/PID_controller (accessed Jan. 04, 2023).

[10] R. C. Smith and P. Cheeseman, “On the representation and estimation of spatial uncertainty,” *Int J Rob Res*, vol. 5, no. 4, pp. 56–68, 1986.

[11] Wikipedia contributors, “Object detection,” *Wikipedia*, Dec. 19, 2022. https://en.wikipedia.org/wiki/Object_detection (accessed Jan. 04, 2023).

[12] Ultralytics, “YOLOv5,” *https://github.com/ultralytics/yolov5*, Apr. 21, 2021.