UNIVERSITY OF TURKU

Design and implementation of a domestic disinfection robot based on 2D lidar

Smart Systems

Master's Degree Programme in Information and Communication Technology

Department of Computing, Faculty of Technology

Master of Science in Technology Thesis

Author:

Zhuoran Chao

Supervisors:

MSc (Tech) Jorge Peña Queralta

Assoc. Prof. Tomi Westerlund

May 2021

Master of Science in Technology Thesis
Department of Computing, Faculty of Technology
University of Turku

Subject: Smart Systems
Programme: Master's Degree Programme in Information and Communication Technology
Author: Zhuoran Chao
Title: A Design and Implementation of Domestic Disinfection Robot Based on 2D Lidar
Number of pages: 74 pages, 3 appendix pages
Date: May 2021

In the battle against the Covid-19, the demand for disinfection robots in China and other countries has increased rapidly. Manual disinfection is time-consuming, laborious, and has safety hazards. For large public areas, the deployment of human resources and the effectiveness of disinfection face significant challenges. Using robots for disinfection therefore becomes an ideal choice.

At present, most disinfection robots on the market use ultraviolet or disinfectant to disinfect, or both. They are mostly put into service in hospitals, airports, hotels, shopping malls, office buildings, or other places with daily high foot traffic. These robots are often built-in with automatic navigation and intelligent recognition, ensuring day-to-day operations. However, they usually are expensive and need regular maintenance. The sweeping robots and window-cleaning robots have been put into massive use, but the domestic disinfection robots have not gained much attention. The health and safety of a family are also critical in epidemic prevention. This thesis proposes a low-cost, 2D lidar-based domestic disinfection robot and implements it. The robot possesses dry fog disinfection, ultraviolet disinfection, and air cleaning. The thesis is mainly engaged in the following work:

The design and implementation of the control board of the robot chassis are elaborated in this thesis. The control board uses STM32F103ZET6 as the MCU. Infrared sensors are used in the robot to prevent from falling over and walk along the wall. The Ultrasonic sensor is installed in the front of the chassis to detect and avoid the path's obstacles. Photoelectric switches are used to record the information when the potential collisions happen in the early phase of mapping. The disinfection robot adopts a centrifugal fan and HEPA filter for air purification. The ceramic atomizer is used to break up the disinfectant's molecular structure to produce the dry fog. The UV germicidal lamp is installed at the bottom of the chassis to disinfect the ground. The robot uses an air pollution sensor to estimate the air quality. Motors are used to drive the chassis to move. The lidar transmits its data to the navigation board directly through the wires and the edge-board contact on the control board. The control board also manages the atmosphere LEDs, horn, press-buttons, battery, LDC, and temperature-humidity sensor. It exchanges data with and executes the command from the navigation board and manages all kinds of peripheral devices. Thus, it is the administrative unit of the disinfection robot. Moreover, the robot is designed in a way that reduces costs while ensuring quality.

The control board's embedded software is realized and analyzed in the thesis. The communication protocol that links the control board and the navigation board is implemented in software. Standard commands, specific commands, error handling, and the data packet format are detailed and processed in software. The software effectively drives and manages the peripheral devices. SLAMWARE CORE is used as the navigation board to complete the system design. System tests like disinfecting, mapping, navigating, and anti-falling were performed to polish and adjust the structure and functionalities of the robot. Raspberry Pi is also used with the control board to explore 2D Simultaneous Localization and Mapping (SLAM) algorithms, such as Hector, Karto, and Cartographer, in Robot Operating System (ROS) for the robot's further development.

The thesis is written from the perspective of engineering practice and proposes a feasible design for a domestic disinfection robot. Hardware, embedded software, and system tests are covered in the thesis.

Keywords: disinfection robot, lidar, ROS, SLAM.

Table of contents

1 Introduction

1.1 The Background and Significance of the Study

Over the past few years, China's total robot market has boomed, with manufacturers such as Ecovacs, Xiaomi, Haier, and Midea launching their sweeping robot products. In 2017, the domestic floor-sweeper market retail sales reached 5.6 billion yuan and 4.06 million units. Data from 2018 showed that auto-window-sweeping robots' sales increased fourfold, and auto-sweeping robots' sales increased 1092-fold[1]. The data shows that there is a huge demand for automated robots, especially sweeping robots.

Unlike the general vacuum cleaner, sweeping robots can conduct autonomous cleaning, and mobile navigation is the key technology. With the navigation system's continuous innovation and path planning technology, sweeping robots' intelligence and cleaning effects have significantly improved.

Automatic robots applied in different areas often involve distinct characteristics. However, we can extract some standard features that these robots should possess, for example, positioning, navigation, barrier avoidance, and path planning. Like ultrasonic sensors, lidar, or cameras, different sensors must be used to perceive the robot's environment to accomplish these tasks. Lidar has been widely used in research and achieved ideal outcomes indoors. We can also combine the use of cameras, and, in principle, we can extract more information from pictures and videos. Other projects also adopt multiple cameras and assign specific tasks to each camera, hoping to improve efficiency and accuracy. The final choice of implementation should depend on the actual needs. Figure 1-1 shows the structure of the robot's online market retail share according to the navigation method in 2019[2]. Furthermore, we can see that laser navigation is still the mainstream option.

In 2020, coronavirus hit the whole world, and the Chinese government has devoted massive resources to contain and then eradicate the epidemic. We have seen that many medical workers joined heading to Wuhan with great courage from television and online network. Moreover, we have also witnessed people dressed in protective clothing performing disinfection in public areas. There are multiple methods to disinfect public areas already, and we want to propose a feasible plan to disinfect the home space guarding the family's health and safety. We designed and implemented a disinfection robot possessing dry fog disinfection, ultraviolet disinfection, and air cleaning.

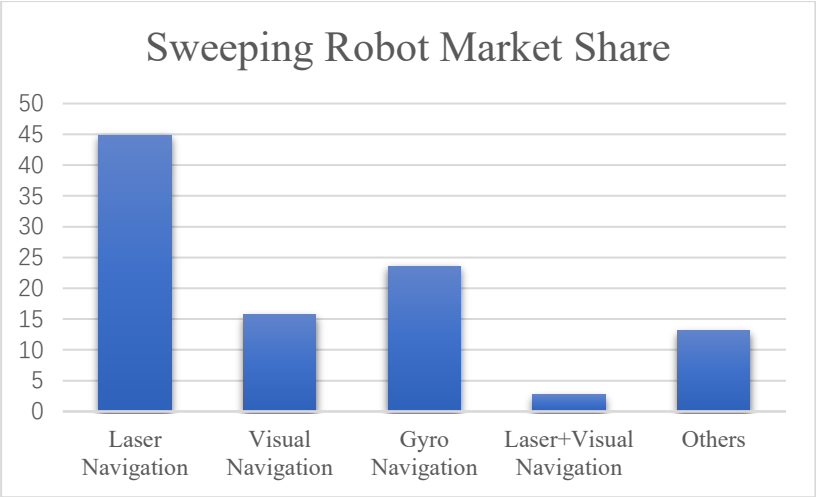


Figure 1-1 Sweeping Robot Market Share.

1.2 Research on Domestic Robots and SLAM

Research on sweeping robots started earlier abroad, and some representative products had been in mass-production and landed in the market. In 2001, Electrolux launched Trilobite[3], the first mass-produced sweeping robot. It is equipped with an ultrasonic sensor that allows it to avoid obstacles. When the cleaning task is completed or the battery is low, the robot will automatically return to the charging station. Trilobite's disadvantage is that it is insensitive to obstacles with sharp angles because the sensor's ultrasonic reflections are rare. Moreover, Trilobite stops at a short distance to the wall or other objects, causing the cleaning not to be thorough. MotionSense, the successor of Trilobite, was introduced in 2016[4]. It is highlighted in self-charging, scheduling, virtual wall, and remote control.

In 2003, Karcher introduced RC3000[3]. The robot has a random operating mode that gradually covers an area with the help of haptic sensors. One unique advantage is that it can examine air quality in the area being cleaned and determine whether this area is dirtier and needs more cleaning. Karcher released RC 3[5] in 2018. The robot can measure each room and then cleans it in a logical sequence.

In 2006, iRobot launched Roomba sweeping robot[3]. According to iRobot's official report, it sold more than 2 million units of Roomba[3]. The robot does not need to know the map of the area to be cleaned in advance. It uses simple algorithms such as spiral mode, random collisions, and walks along the wall to acquire the cleaning area's overall picture. Roomba's new generation can sense the cleanness of different areas and calculate how long it takes to clean an area. It can also walk along the longest route along which it would not collide with

any object. Roomba has expanded into multiple series by 2020, such as s series, e series, 500 series, 600 series, 800 series, and 900 series[6]. Roomba s9+ is the ninth-generation product that features optimized edge cleaning, an anti-allergen system, a self-emptying clean base, and linking technology[6].

In 2010, Neato Robotics began selling a robot that clean up the floor just by a simple press on the start button by users[3]. It adopts low-cost laser sensors and SLAM algorithms for 2D mapping while achieving acceptable real-time performance. The newest versions of Neato Robotics by 2020 include Neato D3s, D6, and D7[6]. The most recently announced Neato D10[7] could get into corners for a more complete clean, and it combines HEPA filters[8], three cleaning modes, self-charging, and autonomous navigation.

Table 1-1 shows the comparison of the technical parameters of the robot displayed in Figure 1-2.

Table 1-1
Parameters of robots.

MODEL	COMPANY	Features	PRICE(RMB)
DEEBOT T8	Ecovacs	Auto path planning, Anti-falling, Self-charging, High-resolution 3D mapping	2999
WINBOT WA50	Ecovacs	Auto path planning, Wireless design, Voice notification, Effective scrubbing	3199
AIRBOT ANDY	Ecovacs	Auto path planning, Air cleaning, ToF sensors, Anti-falling, Self-charging	7499
UNIBOT UF380	Ecovacs	Auto path planning, Self-charging, Remote control, Appliance managing, Household Security	5999

In recent years, the development of sweeping robots in china is speeding. Ecovacs, Xiaomi, and Haier have launched their series of products. Ecovacs, for example, has brought multiple kinds of robots to the market. DEEBOT is a floor-sweeping robot that has ToF[9] sensors to build precise maps and cover the entire room. It can measure the height and drop in case of being stuck or falling. The users can edit the maps on the mobile phone to dynamically adjust the cleaning strategy. WINBOT is a window-sweeping robot. AIRBOT is an air-cleaning robot. It can sense over 300 kinds of viruses or bacterial and cover an area of 120 square meters. The UNIBOT in Figure 1-2 is a butler robot. It can manage the household appliance, clean the floor, broadcast notifications, and patrol the fixed-point area. These robots are displayed in Figure 1-2.

The similarities between the disinfection robot, which is going to be discussed in

this thesis and those robots described above are the basic functionalities that mobile robots should all possess, for example, mapping and navigation. The distinction lies in that they each implement specific functions targeted at specific scenarios.

Figure 1-2 Ecovacs robots[10].

ROS is quite popular in developing robots in academia. It is a scalable framework developed to write programs for robots[11]. It integrates tools, libraries and defines a set of specifications for developing complex and reliable cross-platform robotic applications. Why does ROS need to exist? Why does it provide such a framework for developing robotic applications? It is a challenging task to develop generic robotic software. From a robot's perspective, tasks that are naïve to humans can be exceedingly hard in a robotic world. Dealing with such a challenge is inefficient for individuals, independent laboratories, or institutions. ROS was born out of this and created from scratch to encourage cooperative robotics software development. For example, there are experts in a lab specializing in building maps, and they could contribute to a state-of-the-art map-building system. There is another laboratory where some experts are good at identifying small objects from disorders using

computer vision. ROS is precisely designed to enable such groups to work together and produce win-win outcomes.

Here comes an overview of research in top laboratories at home and abroad in the direction of 3D vision and SLAM.

The Dyson Robotics Laboratory of Imperial College of London (ICL) in Europe was established in 2014, led by Professor Andrew Davison, and supported by Dyson Company with substantial funding and support. They collaborated to develop Dyson 360 eye vacuum cleaner. The laboratory's research direction is to develop computer vision programs that enable robots to go beyond the controlled environment, successfully navigate, and interact with the real world.

The main achievements of the laboratory are as follows. ElasticFusion[12]. It is a real-time dense visual SLAM system, which adopts an architecture that alternates between tracking and mapping[12-14], that can use RGB-D cameras[15, 16] to perform global and consistent three-dimensional dense reconstruction[17, 18] of a room. CodeSLAM[19], which is a large-scale photorealistic rendering system that generates indoor scene trajectories. SemanticFusion[20] is a real-time visual SLAM system that can use convolutional neural networks to semantically annotate dense 3D scenes.

The Autonomous System Lab at ETH Zurich is led by Professor Roland Siegwart and is part of the Institute for Robotics and Intelligent Systems (IRIS). The laboratory aims at creating robots and intelligent systems that can operate autonomously in complex and diverse environments. The laboratory also designs the electromechanical and control system to enable the robot to adapt to different situations autonomously and deal with the uncertain and dynamic daily environment. The robot can move on the ground and in the water and has autonomous navigation functionality in a complex environment. The achievements of the laboratory are as follows. Libpointmatcher, this is a modular library that implements the Iterative Closest Point(ICP)[21] algorithm for point cloud registration. Ethzasl_ptam, this is a framework for monocular SLAM[22, 23]. It was first applied on a micro helicopter[24]. The Micro Air Vehicles (MAVs) need to enter some areas with no GPS signals or pre-built maps. This algorithm enables the MAVs to complete such tasks with a single camera and inertial sensors[24].

Multiple Autonomous Robotic System Laboratory (MARS) is in Minnesota. The laboratory's goal is to promote basic research and education in robotics and computer vision, emphasizing the estimation and control of autonomous ground, aviation, and space exploration vehicles. The laboratory's research directions are as follows: 1) vision/laser assisted inertial navigation system[25]. 2) large-scale 3D positioning and mapping on mobile phones and wearable computers[26]. 3) multi-robot/sensor positioning, mapping, and navigation. 4) the optimal information selection and integration.

The research of the Aerial Robotics Group of the Hong Kong University of Science and Technology covers the entire aviation robot system. The group develops fundamental technologies, such as state estimation, trajectory planning, and mapping, to make the aviation robots autonomously move in complicated environments. The main achievements of this laboratory are VINS-Mono[27] and VINS-Fusion. The latter is an extension of the former and supports multiple types of visual-inertial sensors. The VINS estimator[27] reaches the standard of real deployment with a minimum sensor suite.

The Computer Vision group of the Technical University of Munich, Germany, mainly focuses on computer vision, image processing, and pattern recognition. More specifically, the group's research involves image-based reconstruction, optical flow estimation, robot vision, and visual SLAM. The achievements of the lab are not only algorithms, but open-source code based on these algorithms. DVO_SLAM[28] provides a realization scheme for rigid body motion estimation from RGB-D cameras' continuous images. LSD-SLAM, which is a technology to build maps directly from monocular SLAM. The novelties of the algorithm lie in that it directly aligns two keyframes and incorporate noise into dense maps tracking[29]. DSO[30], which is a new direct sparse mapping method for visual odometry. It adopts the merits of the direct method, such as robustness and full image formation, and eliminates the drawbacks of the sparse methods.

1.3 The Main Contents and Structure of the Thesis

Chapter 1 introduces mobile robots' applications in different scenarios, such as sweeping robots, window cleaning robots, and air purification robots. It also gives the history of the development of robot cleaners and lists some classic products. Essential technologies used in disinfection robots are discussed. Finally, the research and progress of SLAM in some famous laboratories are presented.

Chapter 2 introduces the hardware and structure of the disinfection robot. The circuit design of the chassis control board is elaborated. Components related to disinfection are discussed. The most critical hardware of the robot includes the control board and the navigation board. We designed and implemented the control board. The navigation board is more complicated than the control board. It is built based on a more powerful central unit with multiple peripheral devices. The embedded Linux operating system needs to be correctly ported and customized to drive the navigation board. Algorithm realization comes when hardware and system are ready. The disinfection robot is a long-term project, and we made a two-step plan. Firstly, we use SLAMWARE CORE, a modular robot localization and navigation system produced by the SLAMTEC company, to build a complete and functional system. Then we can perform a system-level validation. At the same time, we use Raspberry Pi to test different SLAM algorithms. Raspberry Pi gives developers more access and more operability. The second-step plan is to build our customized navigation board. Finally, we give the model and picture of the robot.

Chapter 3 introduces the embedded software design of the chassis control board. It starts with software programming conventions that are followed in the robot project. The communication protocol complied by the control board and navigation board are discussed, ranging from definitions to implementation. Transferring data in a consistent form is more efficient and robust. The handling of peripheral devices is illustrated with flow charts, making it more straightforward.

Chapter 4 introduces the development of ROS and the communication mechanisms provided by it. Several classic 2D SLAM algorithms, such as Gmapping, Karto, Hector, are briefly discussed. The software structure of a robot program based on ROS is given at the end of chapter 4. Features and tools provided by ROS can help to speed up the development of a robot program and accomplish system-level validation. The program runs on Raspberry Pi to exercise different SLAM algorithms, which would help develop our customized navigation board.

In chapter 5, the results of system-validation based on SLAMWARE CORE and Raspberry Pi are given and analyzed.

The disinfection robot project is still undergoing. In chapter 6, the subsequent development of the robot and the corresponding challenges are discussed.

2 Hardware Design of the Disinfection Robot

A complete robot system is composed of hardware and software. This chapter discusses the hardware design and implementation. The top panel of the robot interacts with the users. They can operate the pushbuttons to start and stop the robot. The chassis is the most critical part of the robot in that it controls the robot's movement, mapping, navigation, and disinfection. The control board, navigation board, lidar, and other modules are installed in the chassis.

In general, the control board transfers peripheral devices' information to and executes the commands from the navigation board. It is in the middle level of the hardware design and plays a crucial role in the whole system. The navigation board is responsible for mapping and navigation by processing the data from the lidar and the control board. This chapter starts with the general introduction to the disinfection robot and moves forward to cover the details of specific hardware designs.

2.1 The General Introduction to the Disinfection Robot

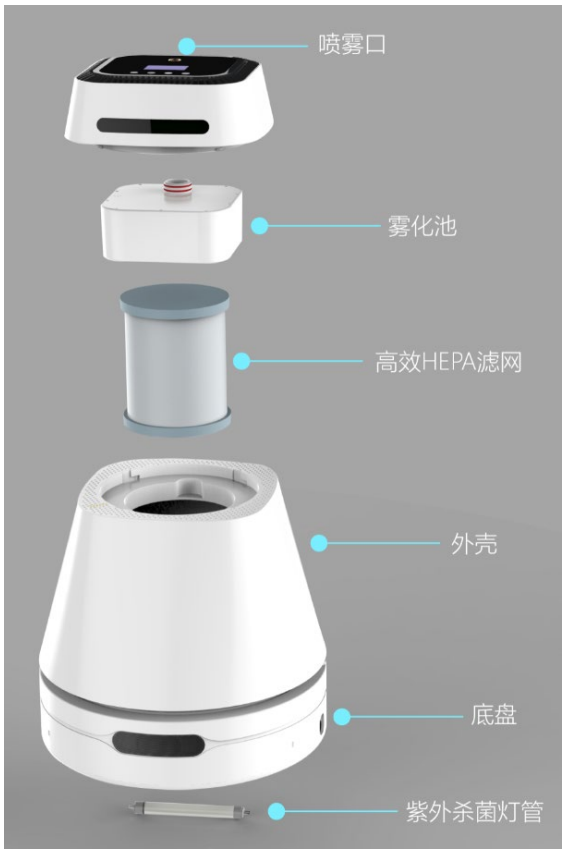


Figure 2-1 Overall structure of the robot.

The disinfection robot can divide into the following parts: top to down in Figure 2-1, top panel, water tank, HEPA filter, a shell, chassis, and the ultraviolet germicidal lamp. The top panel provides four push-bottoms for necessary and straightforward operations, such as start, emergency stop. A hole in the middle of the panel is where the disinfectant fog comes out. The LCD on the top panel displays some concise information: brand name, temperature, and battery status. The water tank holds the liquid disinfectant, which will be atomized by the ceramic atomizer. The HEPA filter is used with a centrifugal fan to clean the air while disinfecting. The water tank and the filter are installed inside the shell. The top panel is mounted to the shell by a simple twist, and the users can easily remove it and fill the disinfectant or replace the filter. The chassis is the most critical part of the robot. The control board and the navigation board are installed on the chassis. It is responsible for mapping, navigation, and disinfecting control. At the bottom of the chassis, the ultraviolet germicidal lamp sterilizes the floor while the robot moves. In all, the disinfection robot provides two mechanisms to sterilize: dry fog disinfection and ultraviolet disinfection. At the same time, air cleaning is provided.

2.2 A General Peek into the Control Board

The 3D model of the control board is given as Figure 2-2, and the PCB is shown in Figure 2-3. We can see from the two figures that the control board is not in a closed shape, and those openings along the edge of the control board match the design requirements of the inner structure of the disinfection robot. Table 2-1 displays the interfaces numbered in Figure 2-2, and the functions of the control board are listed in Table 2-2.

STM32F103ZET6 is the core of the control board. It drives the motors, manages peripheral devices, and exchanges data with the navigation board. The PCIe interface in the middle of the control board connects the navigation board responsible for calculating, mapping, and navigation. The data communication between the control board and the navigation board also works through this interface. The overall idea of hardware design covers control and navigation. The addition of ESP8266 is more like a backup scheme. The wireless module is connected to the MCU, leaving an alternative channel for data transmission between the control board and the navigation board. The robustness of the entire system improves with the proper redundancy of hardware design. It also makes it possible for the control board to communicate with other boards, making the navigation board replaceable in some sense.

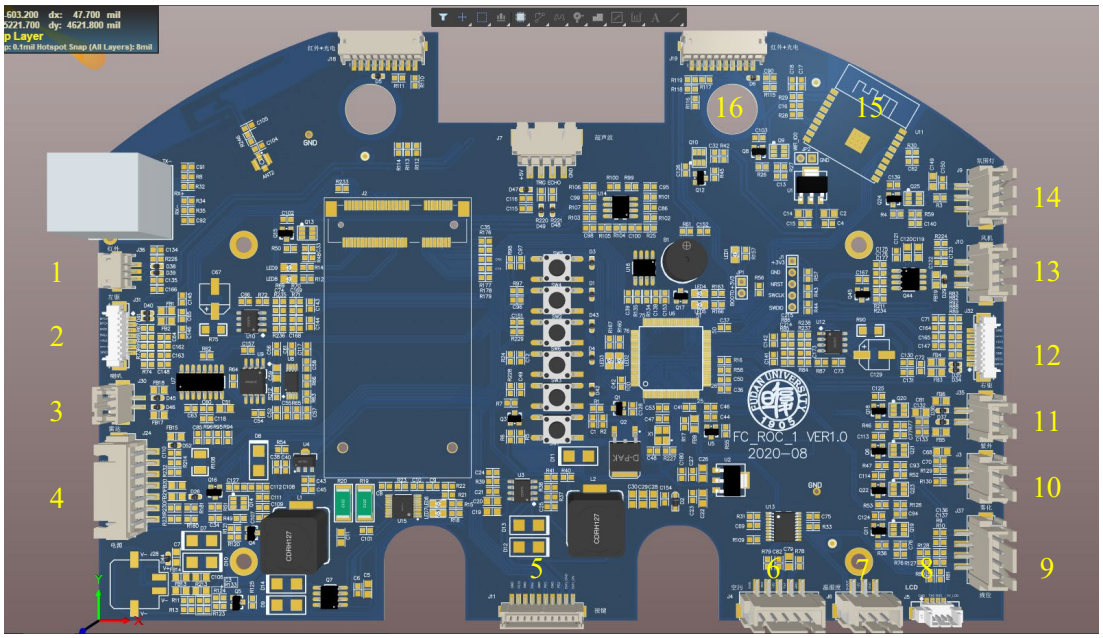


Figure 2-2 The 3D model of the control board.

Table 2-1

Interfaces of the control board.

Number	Interface	Number	Interface
1	Infrared sensor connector	9	Liquid-level detection module connector
2	Left motor connector	10	Ceramic atomizer connector
3	Speaker connector	11	UV germicidal lamp connector
4	2D Lidar connector	12	Right motor connector
5	Push-buttons connector	13	Centrifugal fan connector
6	Air detection module connector	14	Atmosphere LEDs connector
7	Temperature-humidity sensor connector	15	ESP8266
8	LCD connector	16	Infrared sensors and photoelectric switches connector

Table 2-2

Functions of the control board.

Function	Illustration
Movement control	The control board extracts and analyses the data packet sent from the navigation board and sets the PWM signals' parameters and rotating direction of the motors.

Function	Illustration
Sensors processing	The control board handles the sensors connected to it, like the infrared sensors, the ultrasonic sensor, and the temperature-humidity sensor.
Module's processing	The control board handles the air-detection module, the LCD, the fan, and other modules.
Power control	The power adapter and battery connector are on the control board. The control board is responsible for providing a clean and stable power source to all kinds of components in the disinfection robot.
Data communication	The control board sends the data packet to and receives the data packet from the navigation board.

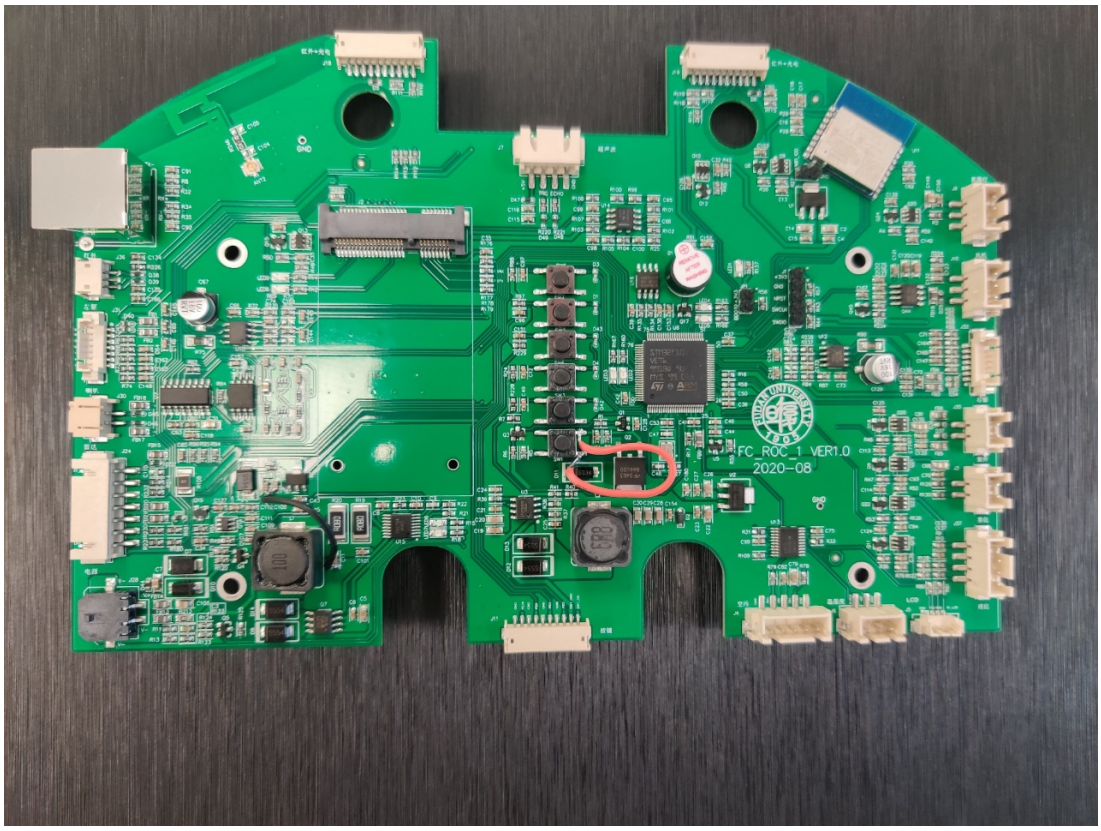


Figure 2-3 PCB of the control board.

2.3 Introduction to the Circuits of the Control Board

2.3.1 Power Management Circuits

The robot is powered by batteries in daily use. It is an unqualified design if users need to switch batteries from time to time. A rechargeable battery pack is equipped on the robot. The nominal voltage of the batteries is 14.8V, and the capacity is 10.4Ah. A power management circuit is designed to charge and discharge and batteries safely.

As presented in Figure 2-4, J28 is the charging interface, V+ connects to the power management chip, and VIN connects to ETA2822, which is shown in Figure 2-5. The other electronic components are used to improve the security and quality of the power source.

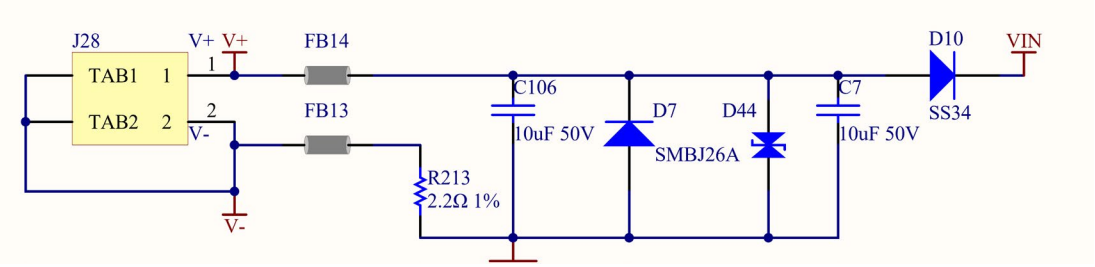


Figure 2-4 Charging interface.

ETA2822 is a DC/DC step-down switching regulator that can achieve efficiency up to 91% taking an input signal of 12V. It can deliver a 3.1A output current. In the disinfection robot, the motors and ultraviolet germicidal lamp consumes large current, making a regulator deliver constant and large current an inevitable design consideration. The 5V signal output by ETA2822 is the input to the three-terminal regulator AMS1117, which outputs a 3.3V signal. Both of the signals drive most of the circuits on the control board.

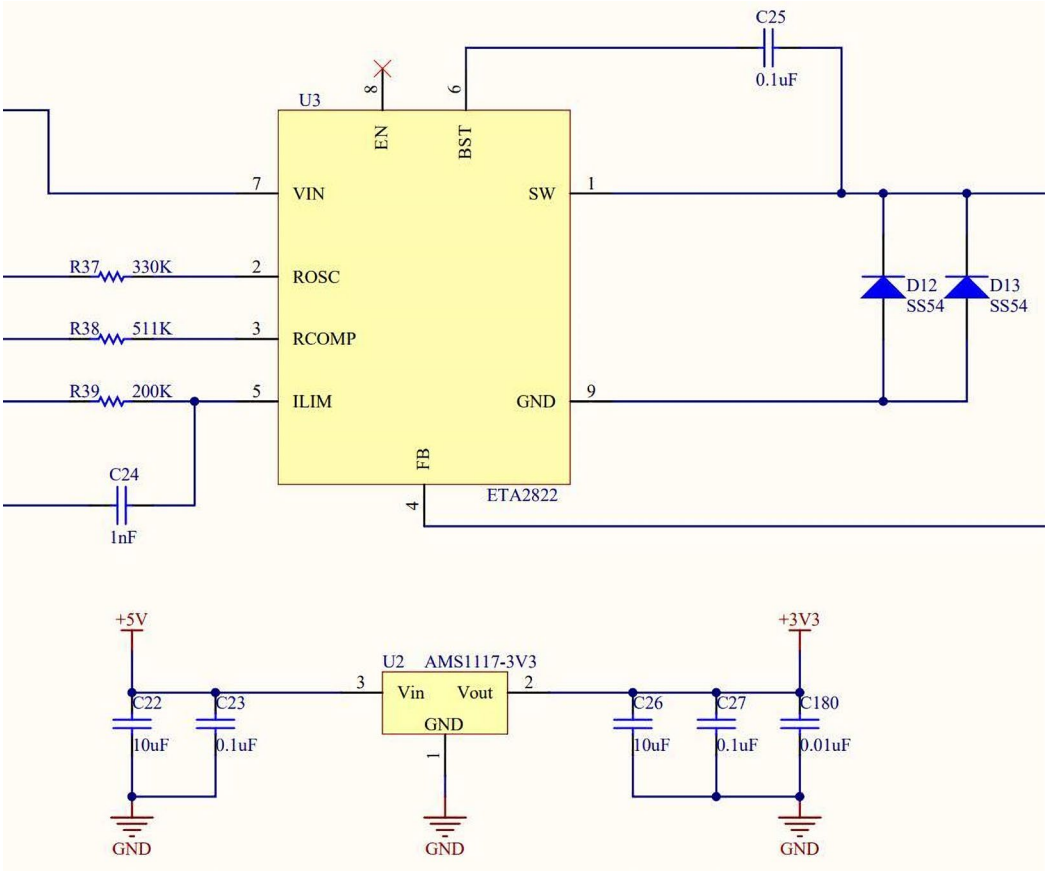


Figure 2-5 Power regulator.

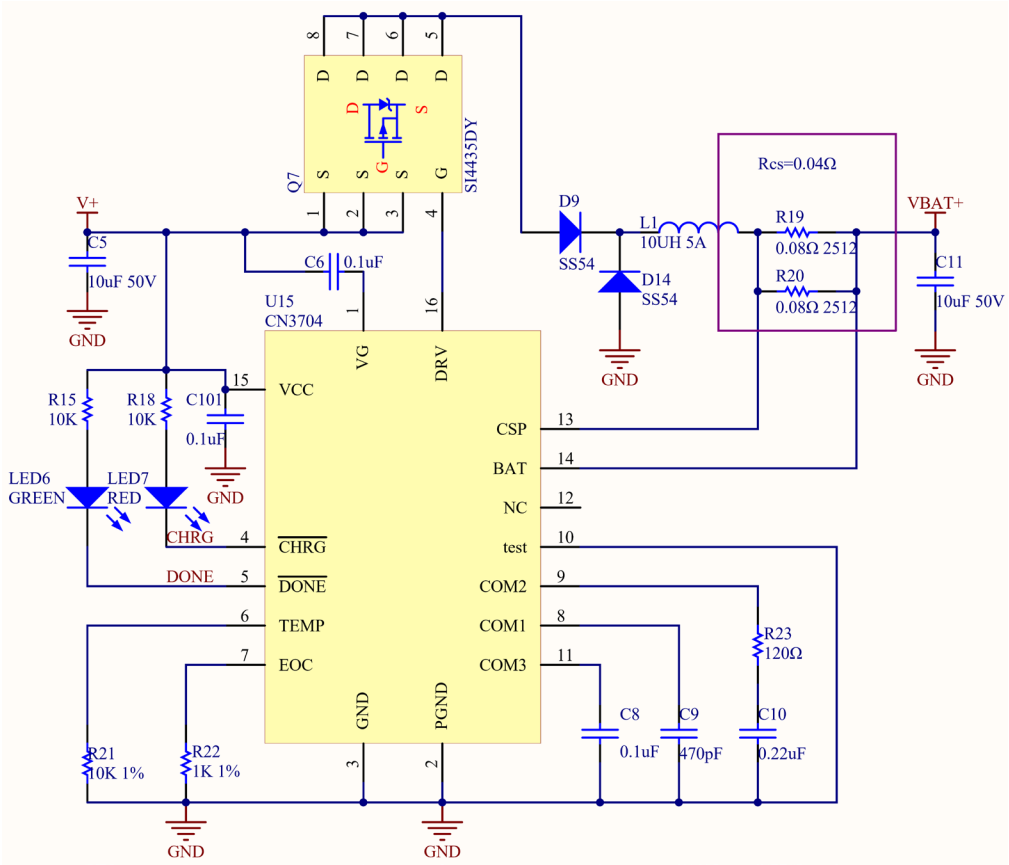


Figure 2-6 Charging circuits.

As displayed in Figure 2-6, the V+ from J28 connects to the VCC of CN3704. VCC is the power supply for the internal circuits of the CN3704, which is a battery charger controller specially designed for charging lithium batteries. We can charge the battery with a constant current by configuring the sensing resistance in the chip's peripheral circuit. In Figure 2-6, this is done by paralleling two sensing resistors of 0.08 ohm to meet the requirement. The sensing resistors are placed on the path of the output current. The voltage divided by the sensing resistors is proportional to the current flowing through them. The CSP pin and BAT pin of CN3704 measure the voltage drop on the sensing resistors to set the charging current. When charging an over-discharged battery, CN3704 will automatically trickle charge, and the current is 15% of the constant full-scale current mentioned before. The trickle charging mode will stop as the battery's voltage exceeds 11.2V, then the battery will be charged with a constant voltage and constant current. When the chip's input voltage is lower than the battery's voltage, CN3704 will automatically enter the sleep mode.

The CHRG pin will be pulled low by the switch inside CN3704 when in charging mode; otherwise, this pin is in a high impedance state. The DONE pin will pull low when the battery

is fully charged. It is also connected to STM32F103ZET6 so that the MCU can query the charging status of the battery.

As exhibited in Figure 2-7, a diode D11 connects between VBAT+ and VIN. One of the purposes of placing the diode is to unify the battery power supply and external power supply, ensuring the power input to ETA2822. The other purpose is to isolate the battery supply and external supply to avoid direct contact between the different power supply. The power supply to the control board must be “clean.”

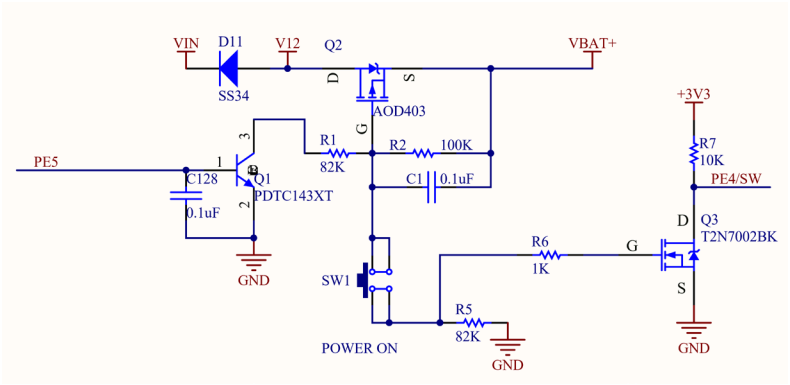


Figure 2-7 Power merging.

In daily operations, the robot powers by a battery. Some peripheral sensors, motors, and ultraviolet light tubes are power consuming. Power switch circuits are added on the power supply path to these components, as shown in Figure 2-8, to save power. The power switch circuit is composed of BJT and MOSFET. The MCU sends the control signal. Such a design makes it feasible to turn on or turn off the power supply to one specific power-consuming component. It elevates the operation resolution. The circuit in Figure 2-8 controls the power supply to the ceramic atomizer. Power switch circuits about other components are similar to Figure 2-8.

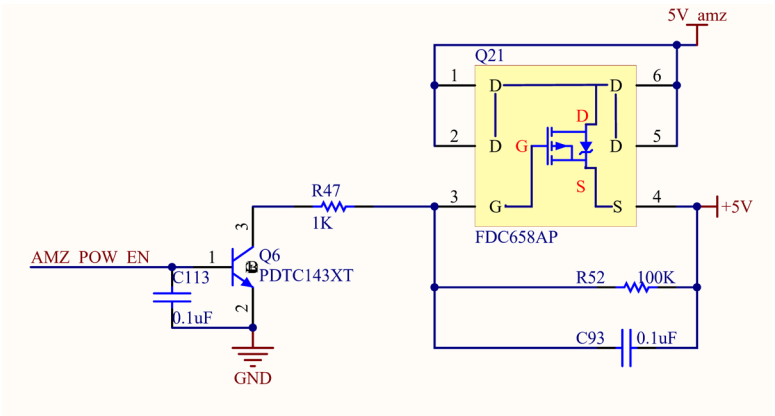


Figure 2-8 Power switch.

2.3.2 Level Shift Circuit

The MCU is driven by 3.3V, and the signals processed in both directions need to be compatible with the voltage reference. However, some peripheral sensors are of 5V reference. A level shifter needs to be there to handle this problem, and an automatic direction switch is better to be built inside the shifter. We choose TI's level shifter to meet these demands.

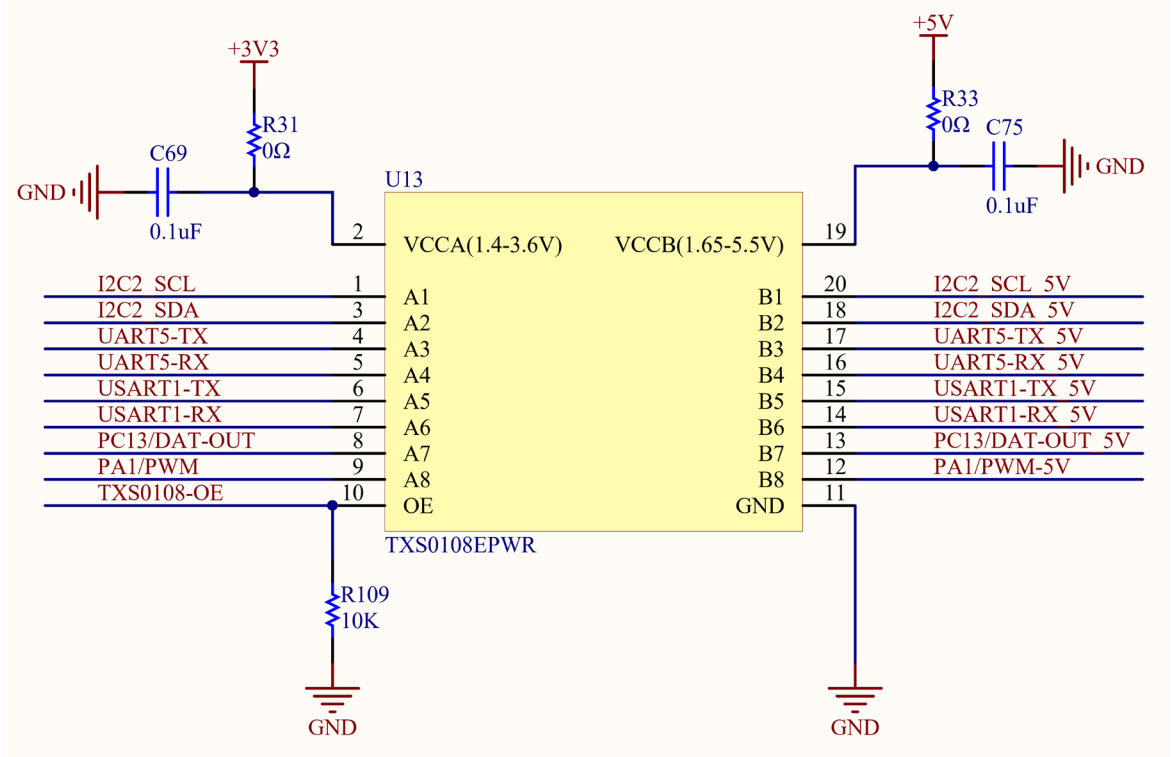


Figure 2-9 Level shifter.

As shown in Figure 2-9, TXS0108EPWR is a level shifter produced by TI. The maximum rate of data transmission supported reaches up to 110Mbps. In the open-drain operating mode, the data rate is up to 1.2Mbps. While the two-way transmission is supported, an excess direction switch is not needed. The reason that we choose this specific level shifter is that it has a flexible pull-up resistor setting. Each A-port IO has a pull-up resistor to VCCA, and each B-port IO has a pull-up resistor to VCCB. These pull-up resistors are integrated inside the chip, which simplifies the circuit design and lowers the cost. The pull-up resistor is 40K Ohm when the output is low and 4K Ohm when the output is high, which lowers the static power consumption. We can configure the OE pin to enable or disable the pull-up resistor setting. Such a design significantly avoids the complexity and clumsy of circuits if we add discrete resistors to the input signals and output signals.

2.3.3 Voice Playback Circuit

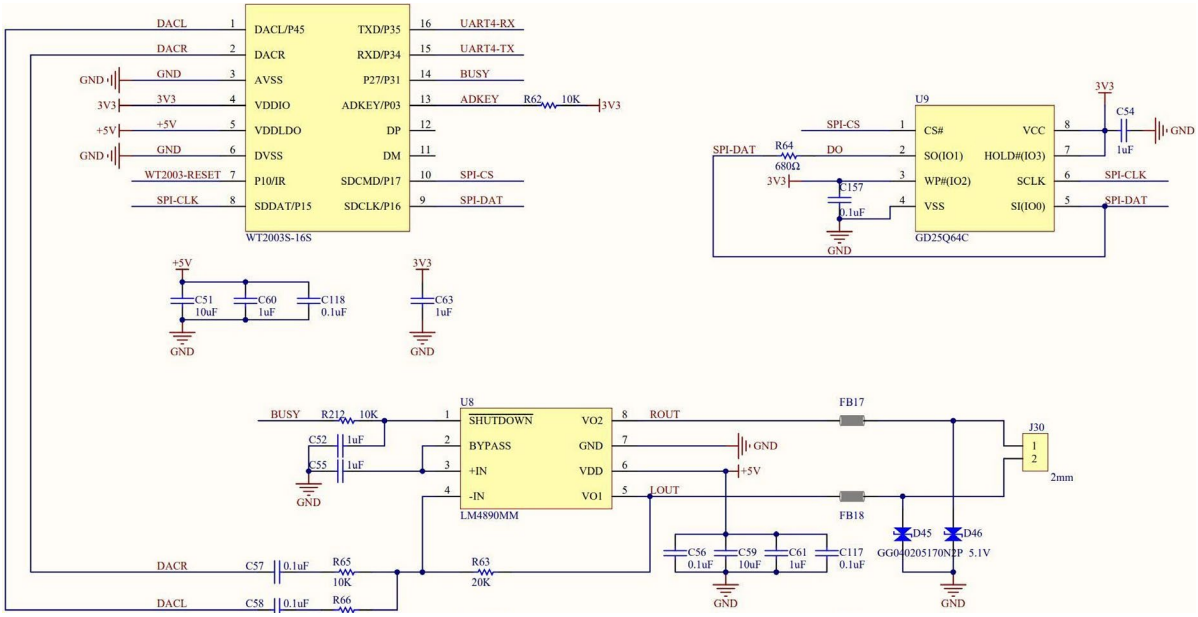


Figure 2-10 Voice playback circuits.

WT2003S-16S is a kind of MP3 chip that supports asynchronous serial communication with an adjustable baud rate. The chip also supports reading from and writing to the flash with the SPI. GD25Q64C in Figure 2-10 is used to store some static voice data for processing.

It has multiple play modes: file index play, inter-cur play, single-loop play, and loop playback. It supports flash with a maximum volume of up to 128Mbit. The LM4890 in Figure 2-10 is a power amplifier chip for audio signals. It only needs a small number of peripheral components to function, and the temperature detection mechanism builds inside the chip. The MCU connects the chip with UART. This part of the circuits increases user interactivity by playing some voice when the robot starts or stops working.

2.3.4 Motor Drive Circuit

The chassis adopts motor modules, which means the motor, wheel, and mechanical structure have been integrated. A4950 is the motor driver chip that is designed for pulse-width modulating DC motors.

The chip integrates a complete H-bridge control circuit, which controls the forward rotation, reverse rotation, and speed regulation of a single motor with two-channel PWM signals. The circuit is shown in Figure 2-11. The PWM signals are of 3.3V reference and the A4950 output 12V-referenced signals to drive the motor. By detecting and comparing input PWM signals, A4950 can also implement braking or enter a low-power mode. The R-IBUS signal inputs to

an amplifier circuit, and the MCU can monitor the motor current by checking the amplified I-R drop on the sensing register R90.

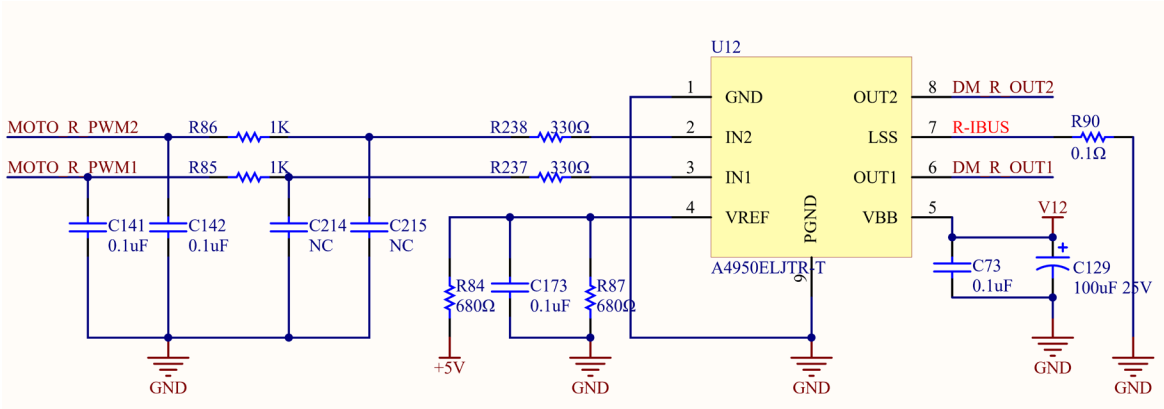


Figure 2-11 Motor driver.

2.3.5 Block Diagram of the Control Board

Based on the above introduction to the chassis control board, the overall block diagram is given as Figure 2-12. Most of the sensors are connected to and controlled by the STM32 MCU. The lidar is the exception. The output signals directly connect to the navigation board through wires and the hardware interface on the control board. The design of the power circuit is also a critical but easily ignored issue. It is the electrical basis for the regular operation of the entire hardware system.

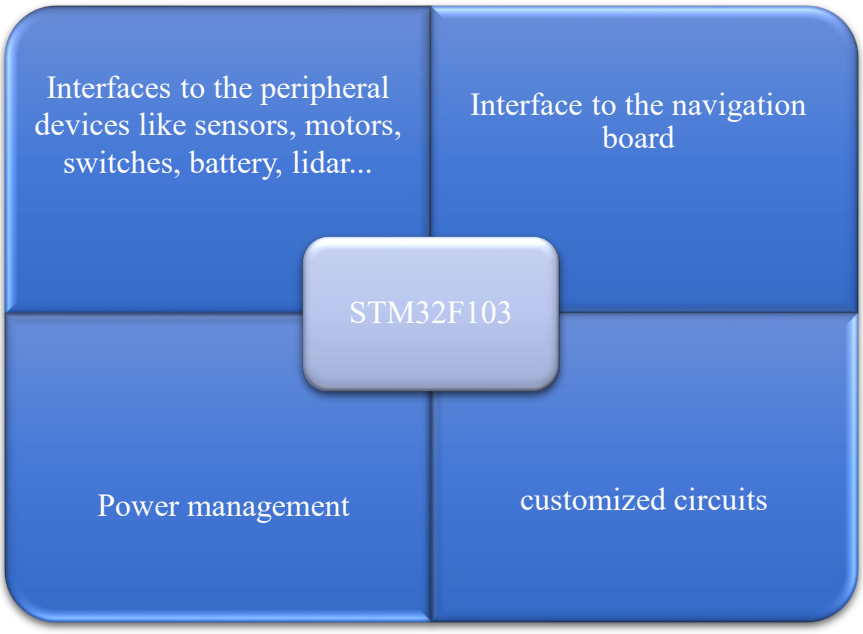


Figure 2-12 Block Diagram of the Control Board.

2.4 Introduction to the Critical Modules

This part mainly introduces some necessary modules used in the disinfection robot.

2.4.1 RPLIDAR-A1

Lidar is the most critical component for the disinfection robot to complete positioning, mapping, and path planning.

The robot equips with RPLIDAR-A1 produced by SLAMTEC. Figure 2-13[31] shows the components forming the lidar. It can scan 360 degrees, and the sufficient scanning distance can reach up to 12 meters. The generated point cloud data can be used for mapping, positioning, and environment modeling. And the A1 lidar collects 1450 points in one round rotation when the scanning frequency is set to 5.5Hz. The maximum scanning frequency can be configured to 10Hz.

The lidar consists of a motor, a digital signal processing unit, and a UART communication interface. It starts to rotate clockwise after power-on, and the users can get the data collected by the lidar from the UART. The lidar embeds a system that can detect rotating speed and adapt it. The scanning frequency of the lidar will be alternated in accordance with the detected speed.

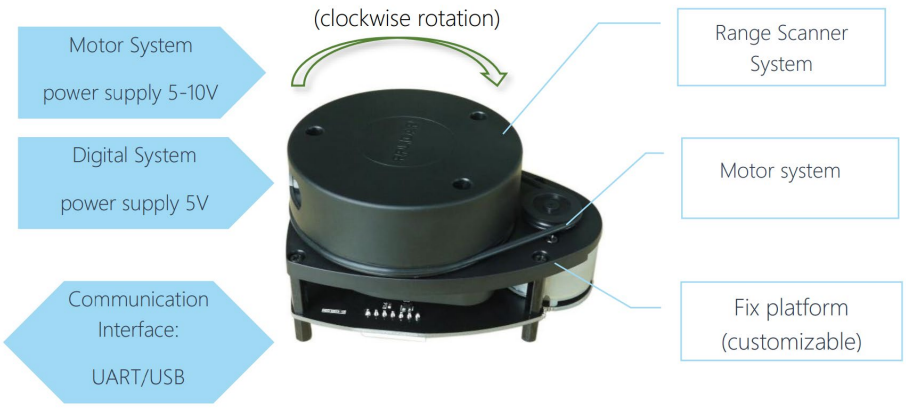


Figure 2-13 RPLIDAR A1.

Merely speaking, A1 lidar is a laser triangulation system. The lidar emits a modulated infrared signal and receives the signal reflected by the object. The reflected laser signal is sampled by the acquisition system and sent to the embedded DSP for processing. Then the distance and angle between the lidar and the object are sent out through UART. The triangulation system exhibits in Figure 2-14[31].

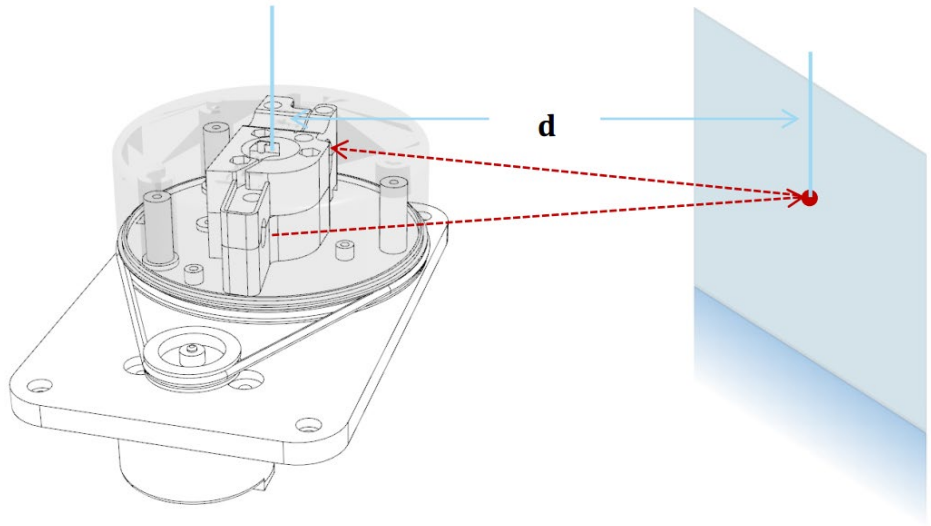


Figure 2-14 Triangular Ranging.

In ROS[11], there are tools that enable direct observation of data when the lidar connects to the PC. One of ROS's effective mechanisms is that robot software written by others can be packaged and spread to the community. SLAMTEC, the A1 lidar manufacturer, provides rplidar_ros, which can be used after being compiled and executed. The communication mechanism and related features of ROS will be described in chapter 4. Figure 2-15 shows how lidar displays the data output as a point cloud in the visualization tool rviz. The red borderline corresponds to the borders of a room. Figure 2-16 corresponds to the situation where the lidar is placed on the table. The point cloud looks rather messy and segmented, which is caused by the various items placed on the table.

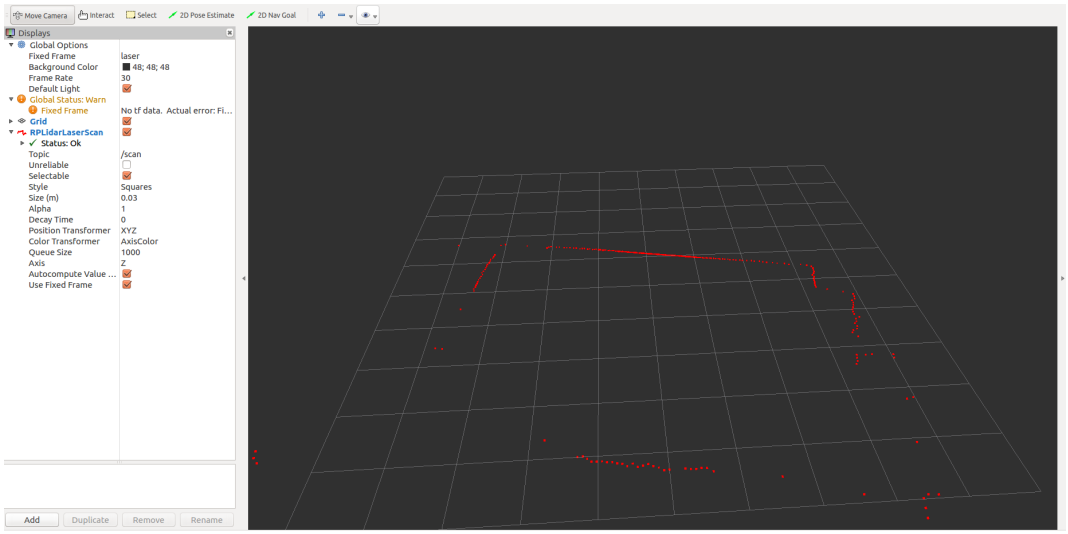


Figure 2-15 Point cloud of a room.

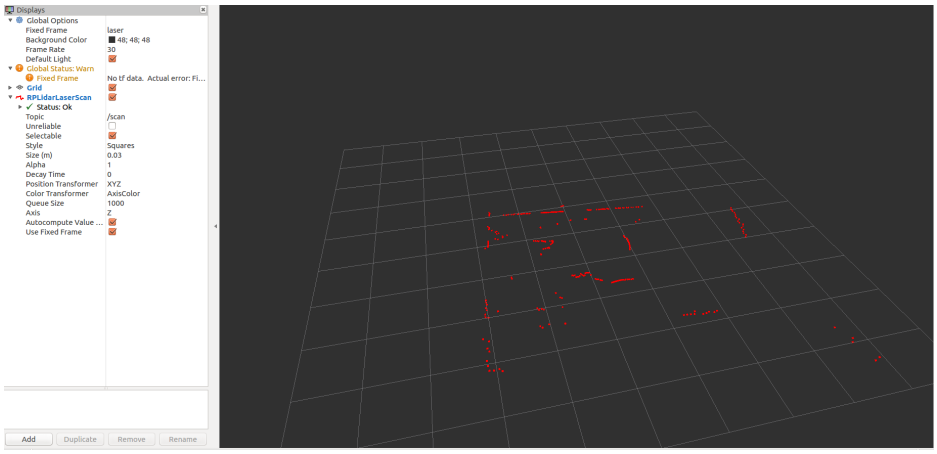


Figure 2-16 Point cloud of a desktop.

The following briefly introduces the representation of data from lidar in ROS. The left column in Figure 2-16 shows the configuration of rviz. Lidar publishes messages to /scan, and the other node subscribes to the topic. ROS provides commands which can inspect details of a topic. In Figure 2-17, the command shows that the publisher is a node abstracted from lidar, and the subscriber is rviz.

```
zhuoran@zhuoran-VirtualBox:~$ rostopic info scan
Type: sensor_msgs/LaserScan

Publishers:
 * /rplidarNode (http://zhuoran-VirtualBox:36471/)

Subscribers:
 * /rviz (http://zhuoran-VirtualBox:36785/)
```

Figure 2-17 Information of a topic.

There are commands in ROS that can inspect more details of a topic. In Figure 2-19, the command gives how frequently the message of the topic is updated. In Figure 2-20, the command peeks into the form of the message from lidar. In Figure 2-18, the command outputs the bandwidth that a topic consumes.

```
zhuoran@zhuoran-VirtualBox:~$ rostopic bw scan
subscribed to [/scan]
average: 24.54KB/s
        mean: 2.94KB min: 2.94KB max: 2.94KB window: 8
average: 23.99KB/s
        mean: 2.94KB min: 2.94KB max: 2.94KB window: 16
average: 22.83KB/s
        mean: 2.94KB min: 2.94KB max: 2.94KB window: 23
average: 22.99KB/s
        mean: 2.94KB min: 2.94KB max: 2.94KB window: 31
average: 23.09KB/s
        mean: 2.94KB min: 2.94KB max: 2.94KB window: 39
average: 22.66KB/s
        mean: 2.94KB min: 2.94KB max: 2.94KB window: 46
^Caverage: 20.98KB/s
        mean: 2.94KB min: 2.94KB max: 2.94KB window: 53
```

Figure 2-18 Bandwidth of a topic.

```
zhuoran@zhuoran-VirtualBox:~$ rostopic hz scan
subscribed to [/scan]
average rate: 7.609
        min: 0.128s max: 0.136s std dev: 0.00357s window: 8
average rate: 7.641
        min: 0.128s max: 0.136s std dev: 0.00340s window: 15
average rate: 7.632
        min: 0.125s max: 0.139s std dev: 0.00388s window: 23
average rate: 7.634
        min: 0.125s max: 0.139s std dev: 0.00385s window: 31
average rate: 7.631
        min: 0.125s max: 0.139s std dev: 0.00389s window: 38
average rate: 7.633
        min: 0.125s max: 0.139s std dev: 0.00377s window: 46
average rate: 7.630
        min: 0.125s max: 0.139s std dev: 0.00377s window: 54
average rate: 7.637
        min: 0.125s max: 0.139s std dev: 0.00379s window: 61
average rate: 7.635
        min: 0.125s max: 0.139s std dev: 0.00385s window: 69
^Caverage rate: 7.635
        min: 0.125s max: 0.139s std dev: 0.00383s window: 71
```

Figure 2-19 Update frequency of a topic.

```
zhuoran@zhuoran-VirtualBox:~$ rosmsg show sensor_msgs/LaserScan
std_msgs/Header header
  uint32 seq
  time stamp
  string frame_id
float32 angle_min
float32 angle_max
float32 angle_increment
float32 time_increment
float32 scan_time
float32 range_min
float32 range_max
float32[] ranges
float32[] intensities
```

Figure 2-20 Elements of a message.

2.4.2 Ultrasonic and Infrared Sensors

The US-100 ultrasonic module is used in the disinfection robot project, and it is installed on the front of the chassis, as shown in Figure 2-21. Ultrasonic distance measurement is primarily adopted in the reversing radar of automobiles and automatic motion planning and robot avoidance. The speed of ultrasonic waves propagating through the air is already known, and the time gap between launching and receiving the waves can be measured. With the speed and the time gap, distance can be calculated by a simple equation. The hardware interfaces of different ultrasonic modules are roughly the same: VCC, trig, echo, and GND. The working procedure of the module is as follows. The module initiates ranging when triggered by a signal from MCU. After that, the module automatically sends a series of square waves with a particular frequency and detects a returning signal. If there is, the echo will output a high-voltage signal. The duration of the high-voltage signal is the same as the time gap. By that, the distance can be calculated.

The disinfection robot is equipped with Sharp's infrared sensors GP2Y0A51SKOF. Its measurement range is from 2cm to 15cm. Based on the sensors' installing position, they can

be used as cliff sensors or wall-detecting sensors. The infrared sensors are used as cliff sensors if they emit infrared signals downwards. For example, when the robot moves to the edge of the stairs or steps, it may fall over. The primitive and critical design principle of hardware must be comprehensive against the cases it might encounter. Figure 2-22 points where those cliff infrared sensors are installed.

Figure 2-21 The place where the ultrasonic sensor is installed.

Infrared sensors are more accurate than ultrasonic sensors in estimating distances. They are more adopted in cases requiring high resolution. Ultrasonic sensors send waves with a specific beam angle. They appear more often in detecting whether there are obstacles in a particular direction.

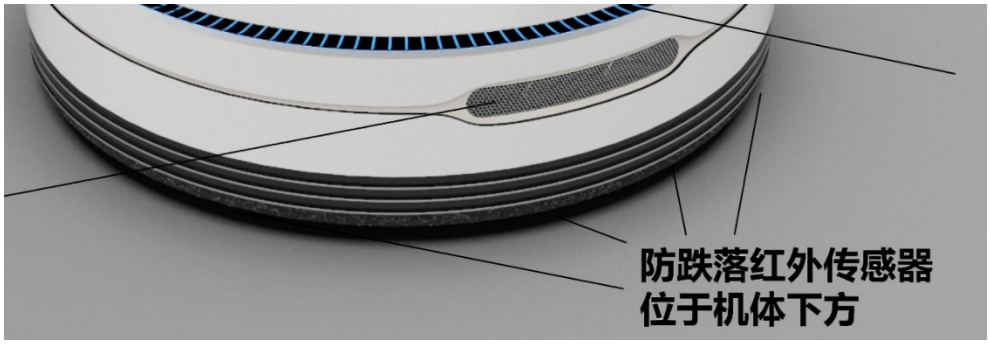


Figure 2-22 The place where the cliff infrared sensors are installed.

2.4.3 Photoelectric Switches

During the hardware design process, the designer must consider such an issue: the robot does not know the surrounding environment's details at first. It needs to sense the surroundings and gradually build a map. As a hardware designer, it is unwise to assume that the robot would not collide with other objects during the scanning process. The collision information should

not be wasted, and photoelectric switches are used to trigger the recording operation when the collision happens.

Figure 2-23 Photoelectric switch.

The photoelectric switch can divide into thru-scan type, retroreflective type, and diffuse-scan type according to detection mode. Photoelectric switches of thru-scan type are adopted in the disinfection robot project. In Figure 2-24, the outer boundary of the black shaft is wrapped by a plastic ring. When there is a collision, the shaft will rotate and embed it into the photoelectric switch slot. The emitted light beam is blocked, forming a switch signal. The signal will then be handled as a bit of information by MCU.

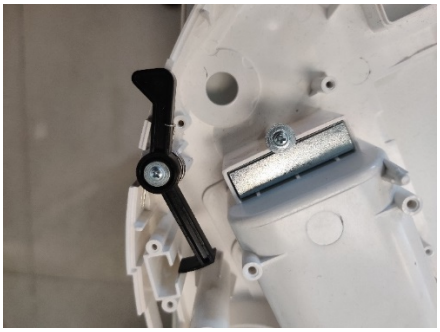


Figure 2-24 The slot.

2.4.4 Liquid Level Detection Module

The liquid level detection module uses TM601A as the controller. TM601A is a multi-channel capacitance-sensing chip that can be used for continuous liquid level detection. The corresponding height is calculated by detecting the changes in different channels' value when the liquid in the container reaches different height levels. The circuit design of the module is shown in Figure 2-25. TP0 and TP1 connect to the sensing capacitors. The host can set the liquid level reference value through the I2C interface and read the value of the liquid level's continuous change within the setting range. TM601A needs to obtain the upper and lower limits of the liquid level range when being operated on. Therefore, the data of critical point needs to be collected in advance and stored in a flash. There are three thresholds to be

collected. Firstly, collect and store the value when there is no liquid in the container. Secondly, repeat the process when the liquid level reaches the zero point (it does not represent zero when the container is empty). Finally, finish the operation when the liquid reaches the full level.

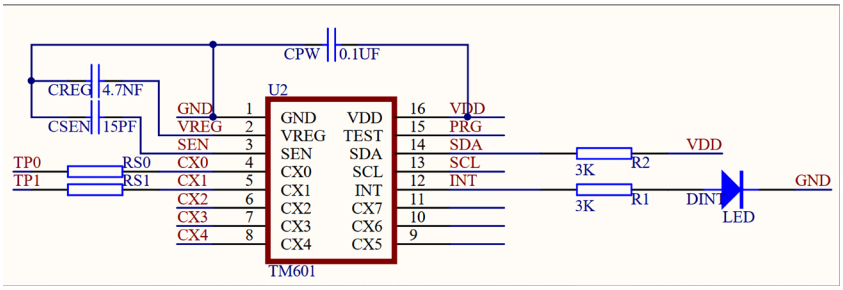


Figure 2-25 Liquid level detection chip-TM601A.

Typically used level measurement technologies include guided-wave radar measurement, radar level measurement, ultrasonic level measurement, and capacitive level measurement. In a disinfection robot, non-contact capacitive level measurement is applied. Capacitive level measurement acquires the liquid level by detecting the change in capacitance value caused by the liquid level change. Figure 2-26 helps to demonstrate the method more intuitively. The yellow film corresponds to the sensing capacitor, and the left piece is the other counterpart.

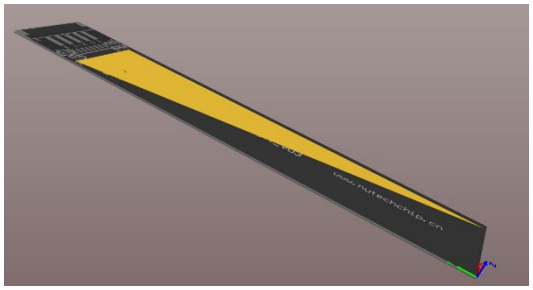


Figure 2-26 Liquid level detection module.

2.4.5 Disinfection-related Modules

The modules related to the disinfection functionality are a water tank, air filter, centrifugal fan, ultrasonic atomizer, and ultraviolet germicidal lamp. The water tank is used to hold the disinfectant. The sponge stick sucks up the disinfectant, and the ultrasonic atomizer will break up the molecular structure of the disinfectant through the high-frequency resonance. Then, the mist will spray out from the middle of the microporous atomizing sheet. This process is dry fog disinfection, and it includes some excellent features. When the average diameter of the droplets is less than 10 microns, as shown in Figure 2-27, the sprayed mist can be said to be ‘dry.’ The dry fog droplet will not settle and will move randomly according to the principle of

Brownian motion. The droplets will not aggregate together to produce large droplets. Dry fog droplets rebound after contact with the surface without breaking and wetting the surface, as displayed in Figure 2-28. Therefore, these properties of dry fog make the places that are difficult to reach also be well disinfected.

The ultraviolet germicidal lamp is installed at the bottom of the chassis to sterilize and disinfect the ground. The UV disinfection has no residual disinfectant or toxic by-products, thus avoiding secondary pollution and oxidative corrosion. The UV light in the range from 185nm to 400 nm has been proven to be antimicrobial. The UV germicidal lamp used in the disinfection robot emits the UVC[32] of 253.7nm, which lies in the range of optimum germicidal wavelength around 254nm. The centrifugal fan and the HEPA filter are installed below the water tank because the users need to fill the water tank more often than change the HEPA filter. The robot not only disinfects but also cleans the air.

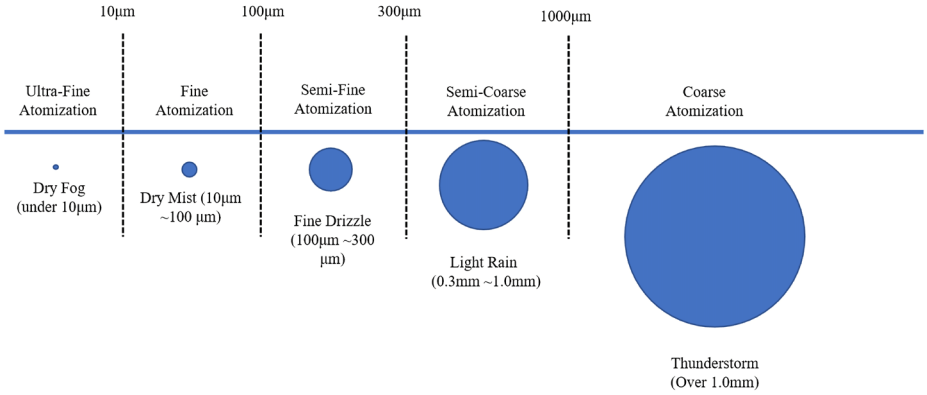


Figure 2-27 Atomization scale.

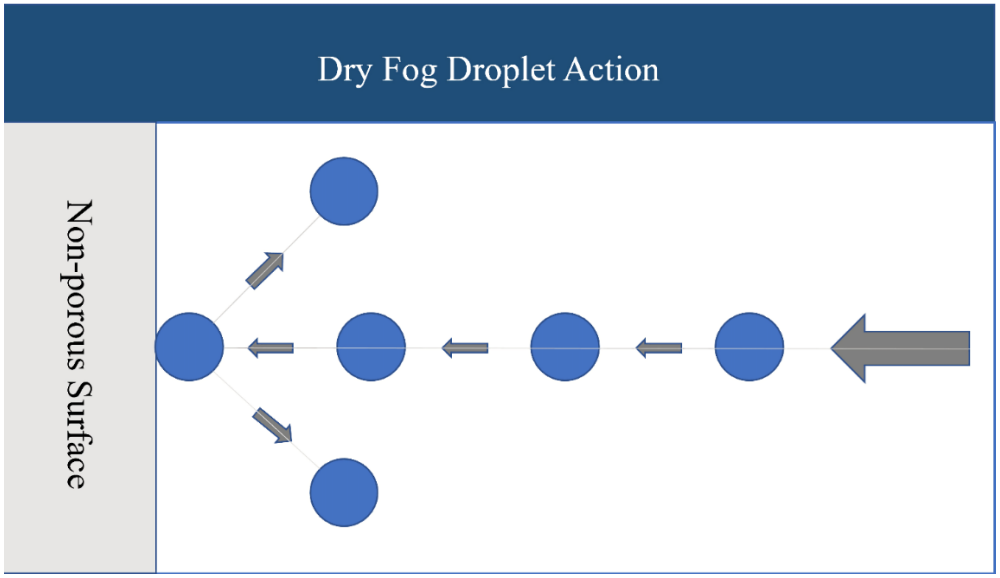


Figure 2-28 Dry fog droplet action.

2.5 The hardware of the Navigation Board

The control board not only collects data from its peripherals but also drives them to work. The way that how to control depends on the logical decision made by the navigation board. By now, our navigation board is still under development. For that, we adopt SLAMWARE CORE from SLAMTECH as the navigation board to perform a complete system-level validation. Besides that, we also exercise SLAM algorithms on Raspberry Pi to prepare for our customized navigation board development.

Figure 2-29 is a picture of SLAMWARE CORE. It consists of an ARM CPU, a flash, a memory, a built-in 9DOF IMU, and supports multiple sensor fusion. It connects to the control board through the edge-board contact. The lidar data is transmitted to it directly. We use this unit to complete the system design of the disinfection robot.

Figure 2-30 shows the inner structure of the chassis of the disinfection robot. The lidar is placed right in the middle of the chassis. It emits the modulated infrared signals through the space between the chassis and the robot's upper shell. SLAMWARE CORE connects to the control board through the PCIe interface. However, the communication between the two boards does not conform to the PCIe protocol. The details of the communication protocol will be discussed in chapter 3. Two pins of the PCIe interface are used, and the unoccupied pins can be used for expansion if needed.

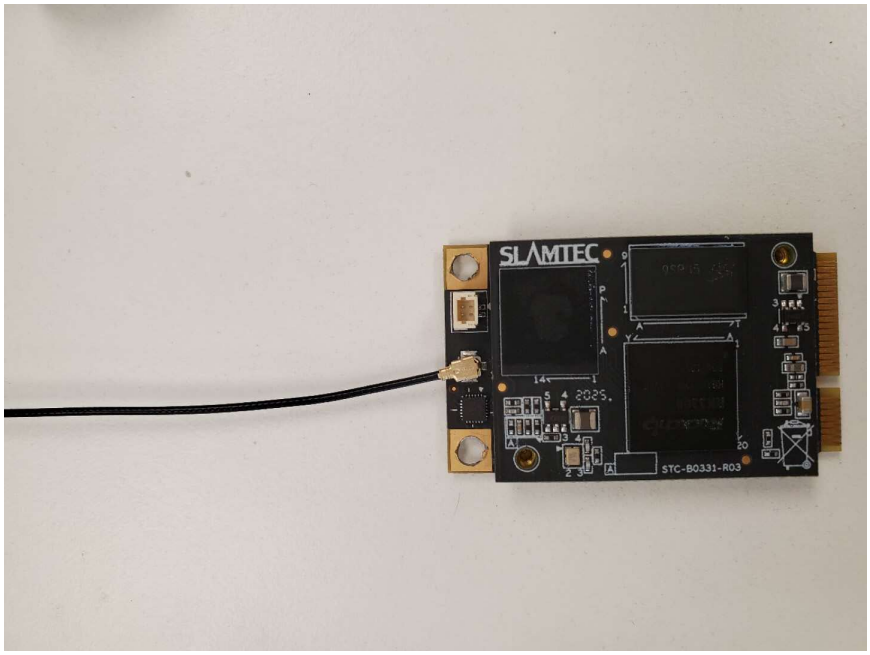


Figure 2-29 SLAMWARE CORE.

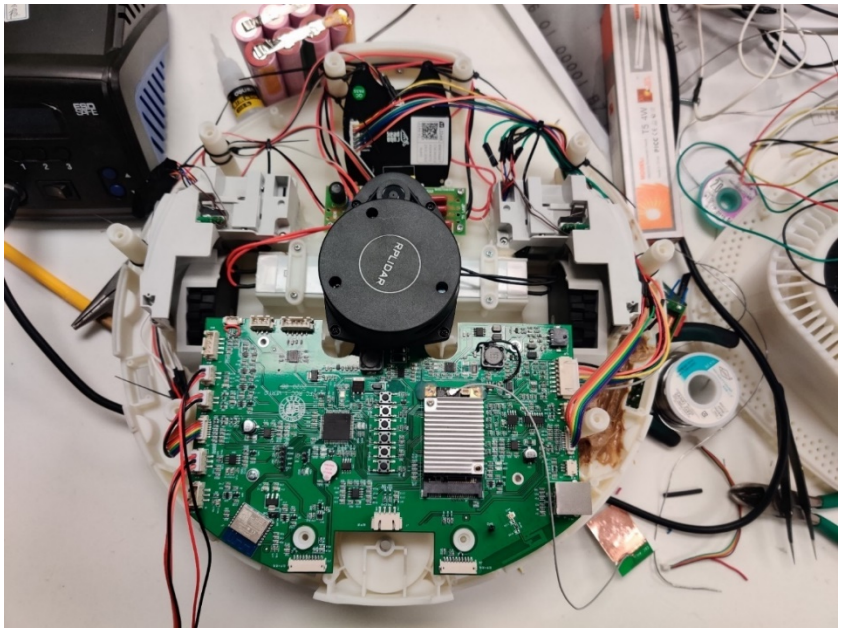


Figure 2-30 Inner structure of the chassis of the robot.

In chapter 4, the software architecture of using Raspberry Pi as the navigation board is given. The experiments are exercised under ROS[11], which is also discussed in chapter 4. SLAMWARE CORE completes the whole system enabling system-level validation. Experiments on Raspberry Pi generate experience for the following customized design of the navigation board.

2.6 Model and Physical Picture

Figure 2-31 shows the SolidWorks model of the disinfection robot. Figure 2-32 is the authentic test-version robot made by 3D printing.

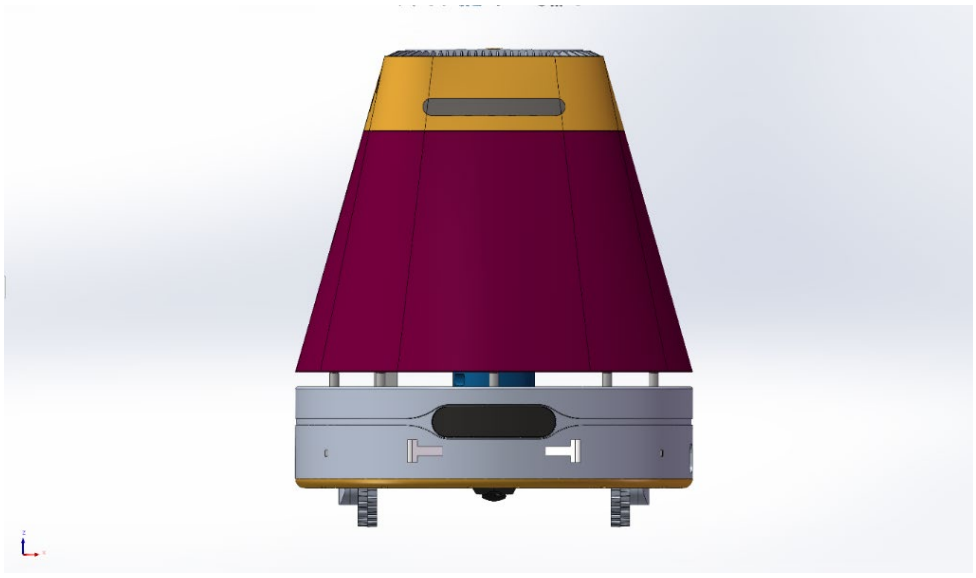


Figure 2-31 SolidWorks model of the robot.

Figure 2-32 3D-Printed robot.

Two wheels under the chassis are responsible for movement and direction control. A universal wheel is used to assist steering. The LCD on the top panel displays some parameters, such as brand name, liquid level, or temperature. The four buttons are used to perform some simple operations, such as start, emergent stop.

2.7 Summary

This chapter describes the hardware composition of the disinfection robot. It starts with the control board, which is responsible for collecting information, packaging, and responding to the request from the navigation board. The control board's circuits are elaborated, and the modules used in the robot are also described. The compulsory module is lidar, from which the data sent back are used to build maps, letting the robot know the environment it did not know. The multiple interfaces placed on the control board's perimeter are used to connect to sensors, motors, and switches. The control board collects the data from peripheral devices and controls them based on the parameters sent from the navigation board. The disinfection robot is a complete functioning system. It possesses the necessary functionalities to fulfil the requirements planned.

3 Embedded Software Design of the Control Board

The robot's hardware can divide into two parts: the control board and the navigation board. Each part is responsible for handling different tasks. The control board with MCU as the central unit directly interacts with peripheral devices to handle the first-hand information. MCU features in real-time operations and is reasonably fit for the scenarios that require immediate processing. Embedded software running on the control board does not have to consider complex task scheduling or concurrent processing. It needs to carefully arrange different tasks and interact correctly with the navigation board. The control board conforms to a communication protocol and exchanges data with the navigation board in a uniform format. A more powerful CPU is used on the navigation board to process the data from the control board and other tasks. From the development perspective, separating the control board and the navigation board can reduce instability and improve development efficiency. The control board communicates with the navigation board through the UART interface. Although they are connected physically through the PCIe interface, only a few edge-board contact pins are used. The control board reserves the wire connection between MCU and ESP8266, which leaves a backup communication channel.

This chapter presents the software design process of the MCU-based control board, including communication protocol, program flow, and detailed explanation.

3.1 Programming Conventions

Programming needs to follow specific rules to make the entire project structure clear. Good conventions make the code easy to modify and distribute. Following specific rules during the development is also conducive to the project's overall coherence, avoiding chaos.

The programming conventions followed by the robot project are as follows:

- Design file directories wisely. The code files should be arranged in different directories according to their characteristics.
- Try not to manipulate the underlying registers directly. Name macros and variables with self-annotation. Use macros to manipulate registers, obtain hardware mapping addresses, and manage the program's resources.
- Use the macros reasonably to improve the flexibility and robustness of the program.

- Classify the source files considering the places where they are applied. For example, the data exchange between the control board and navigation board uses the UART. However, the source files handing the data exchange should be separated from the source files that manipulate the hardware because the former involves data encapsulation and interpretation. The operations on the UART can be used as the bottom layer of those that rely on them.
- Pay attention to functions handling data transmission, especially memory management and the issue of data overflow.
- Functions or variables beginning with '_' indicate that they locate at a low level in the project's implementation, such as static variables in specific modules, function pointer variables, or buffers. The '_' symbol is more of a prompt to readers and does not affect the program's operations.
- Define a default value for those critical function pointers to prevent mis-operations.
- Pay attention to the annotation and self-annotation of the code.

3.2 The Overall Framework of the Embedded Software

This section discusses the embedded software's overall framework from power-on, initialization, and data processing to shut-down. The detailed explanations will be carried out in subsequent sections. Figure 3-1 shows the overall program flow.

The fault handling function is registered at the beginning of the program to deal with errors in a timely and effective manner. The fault handling function mainly completes related components' shutdown, such as unbinding the UART and DMA. It also designates default operations for interrupt-driven data transmission to secure the data buffer. The global interrupt is disabled when the software-level error handling is completed.

The program will wait for a short period when the registration of the error handling function is completed. When the power supply is stable, the board-level initialization will start. It completes the configuration of the clock tree, general-purpose input and output, priority-related hardware, ADC, and others.

When the board-level initialization completes, the initialization of all peripherals starts. It includes the initialization of debugging UART, battery detection and monitoring, infrared sensors, motor control, speed setting, collision detection, and ultrasonic sensors. The

initialization process jumps to execute the error handling function if any peripherals are not correctly configured.

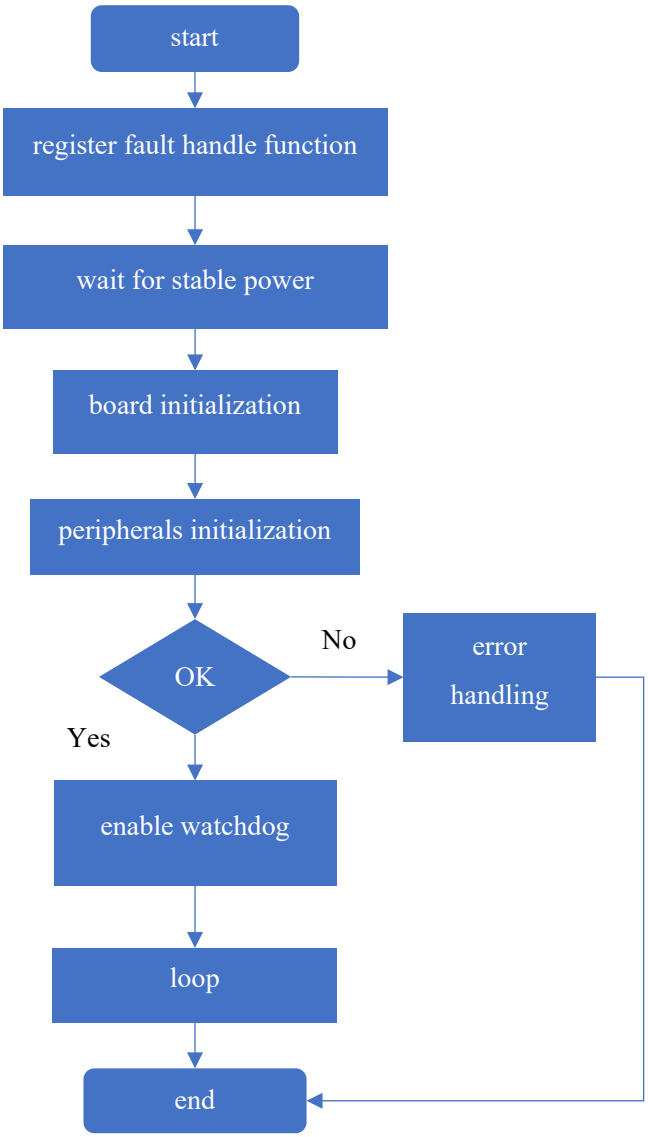


Figure 3-1 Overall program flow.

The watchdog is then enabled to prevent the program from being stalled due to unexpected errors. When the operations discussed above are finished correctly, the program will enter the loop function to execute the main tasks. The loop function will periodically check whether a data packet sent from the navigation board is received. The control board will respond to the navigation board according to the command decoded from the data packet. No matter a responding packet to the navigation board is needed or not, the heartbeat functions will be sequentially executed to handle the peripheral devices and update relevant variables.

3.3 Communication Protocol and the Software Implementation

3.3.1 Selection of the Hardware Transport Protocol

In an embedded project, the chip not only simply reads from or writes to GPIOs but also engages the transmission of a data stream. The MCU needs to communicate with peripherals using different hardware transport protocols, such as UART, SPI, or USB.

Figure 3-2 Format of the UART transmission.

The hardware transport protocol selection needs to consider the hardware consumption, rate of data transmission, and complexity of the data packet. UART is used to transmit data between the control board and navigation board, given the consideration of the robot's actual demands. The data format of UART is shown in Figure 3-2. The configurations are 115200 bps, 8 bits of data length, one stop bit, and no parity bit.

3.3.2 Specification of the Communication Protocol

UART is the carrier of data transmission. Specifically, two wires are needed to transmit data, transmission line (Tx) and reception line (Rx). The code writer can choose to transmit data with different UART configurations, but it is not efficient, and the code would be messy. Therefore, it is an inevitable choice to send data in a packet in uniform formats.

Table 3-1 shows the format of the response packet that the control board sends to the navigation board. In Table 3-1, u8 represents an 8-bit unsigned integer. u8[n] is an array that consists of u8 elements. The data packet is sent from the byte at a low address. The definition of the Flag field is shown in Table 3-2.

Table 3-1

Format of the response packet.

Data Type	u8	u8	u8	u8	u8[n]	u8
Field	Flag	Len	LenH	CMD	Payload[n]	CHKSUM
Sending Order	→					

Table 3-2

Format of the Flag field.

Storage Order	MSB							LSB
Flag	Reserved	LongFrame	Reserved	CheckSumEn	0	0	0	0

Currently, the response packet supports two formats, short data packet and long data packet. The two packets are distinguished by the value of the Flag field, respectively. In a short packet, the value of the Flag field is 0x10. The value is 0x50 in a long packet. The lower 4 bytes of the Flag field is set to zero by default and not used. When it is a long packet, the field Len, together with LenH, indicates the length of the packet, which is the sum of the length of the payload field and CMD field. The packet needs to attach the checksum to its end if the CheckSumEn bit of Flag is set to 1. In standard cases, this bit is also set to ensure the integrity and correctness of the packet.

In the disinfection robot's communication system, it is agreed that the endpoint that always initiates data transmission is the master and the endpoint that always receives a data packet before sending a response packet is the slave. The data packet sent by the master is called the request packet. The data packet sent by the slave is called the response packet. Request and response must appear in pairs, as shown in Figure 3-3. In a request packet, the Length field is equal to the sum of the length of the Payload field and one if the master wants the slave to execute a specific command, except for some special commands.

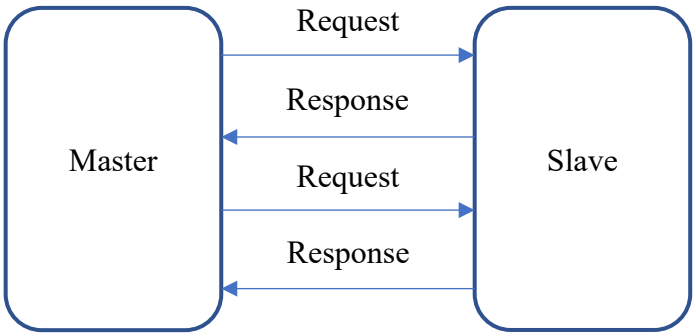


Figure 3-3 Master-Slave model.

The request packet format is slightly different from the response packet, which is reflected in the additional TYPE field in the request packet, indicating this is a packet sent by the master. Table 3-3 to Table 3-6 gives the formats of data packets involved in communications. Table 3-3 is the format of a long request packet, while Table 3-4 is the short counterpart. Table 3-5 is the format of a long response packet, while Table 3-6 is the short counterpart. The differences between the packets of the same category are just the Sync field and Length field.

Table 3-3

Format of a long request packet.

Flag	Length	TYPE	Request	Payload	CHKSUM
0x50	[15:0]	0xA8	CMD	TBD	u8

Table 3-4

Format of a short request packet.

Flag	Length	TYPE	Request	Payload	CHKSUM
0x10	[7:0]	0xA8	CMD	TBD	u8

Table 3-5

Format of a long response packet.

Flag	Length	Response	Payload	CHKSUM
0x50	[15:0]	Resp	u8[n]	u8

Table 3-6

Format of a short response packet.

Flag	Length	Response	Payload	CHKSUM
0x10	[7:0]	Resp	u8[n]	u8

3.3.3 Implementation of the Communication Protocol

The previous section discussed the packet format. This section explains the implementation of the communication protocol. The navigation board always initiates data transmission, and the control board receives and decodes the data packet. We start from data reception and gradually move to other topics.

The communication protocol stipulates that the data transmitted is in the byte stream format, and a whole chunk of byte array is allocated to manage the transmitting buffers of all the UARTs used in data transmission. Table 3-7 shows the structure of the array. Each UART used is numbered in the program, and this number is called the ID of the UART.

Table 3-7

Memory of the transmitting buffers.

ID [0] 512Bytes	ID [1] 512Bytes	…	ID [n-1] 512Bytes

The macro in the following is used to get the start address of the transmitting buffer of the UART numbered with ‘id.’ In the program, we add the offset of individual UART to the start address of the array to get the start address of the memory assigned to the corresponding UART.

*#define GET_TX_BUF (id) (tx_buf + getBufID(id) * USART_MAX_TX_COUNT);*

As manifested in Figure 3-4, the data reception is driven by the interrupt. The data will be processed according to the operations defined in the user-pointed interrupt handler. If the customized interrupt handler is not registered, the default interrupter handler will be executed. The default interrupt handler uses a ring buffer to store data to avoid buffer overflow. The UART that is used to carry the communication protocol is assigned an entire reception buffer of 512 bytes. When the buffer receives the first byte, we need to check its value to know if a legitimate packet transmission starts. If it is, the variable cached_state will activate, and the state machine that manages the status change of data reception starts. The cached_state variable is used to store the historical state, and the program checks the variable to determine the status of data reception. For example, when cached_state is equal to 1 or 2, it means that the buffer has received 0x10 or 0x50. When the cached_state is equal to 3, it means that the program is checking if the length of the payload field is correct or not. A data overflow will happen if the request packet’s payload length exceeds the reception buffer’s size. When the cached_state is equal to 4, it means that the buffer is waiting for the last byte, which is the checksum, to arrive.

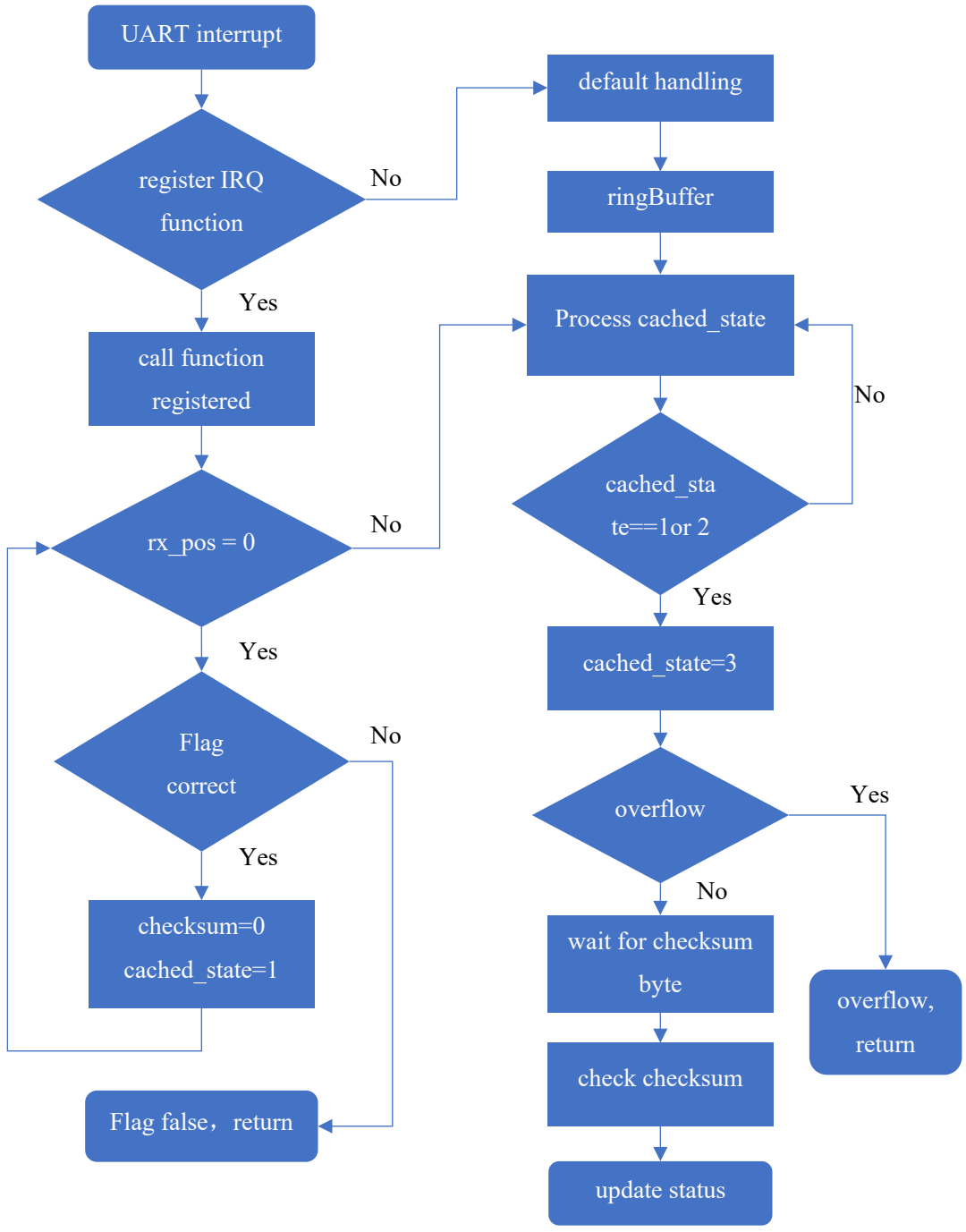


Figure 3-4 Interrupt-driven data reception.

For better comprehensions, the elements of the core structure of the communication protocol are explained as follows. There are ten elements in the structure. The first element defines the address of the buffer that stores the data received by UART. The second element stores the mapping address of the UART, which is used for communications in memory. The third element is a function pointer that registers the callback function executed when receiving interrupt happens. The fourth element and the fifth element operate the transmitting buffer. The sixth element stores the receiving buffer's size, while the seventh stores the size of the

transmitting buffer. The eighth element rx_state represents the state of the communication. The ninth element points out where to store the new data. The last element stores the checksum.

As mentioned before, the process of receiving data is driven by interrupts. The interrupt handler will complete the data filling, calculation, and comparison of the checksum and validate the packet. It does not care about what kind of request packet it is processing. The interrupt handler's responsibility is to ensure the legitimacy of the data packet, which means the packet is intact.

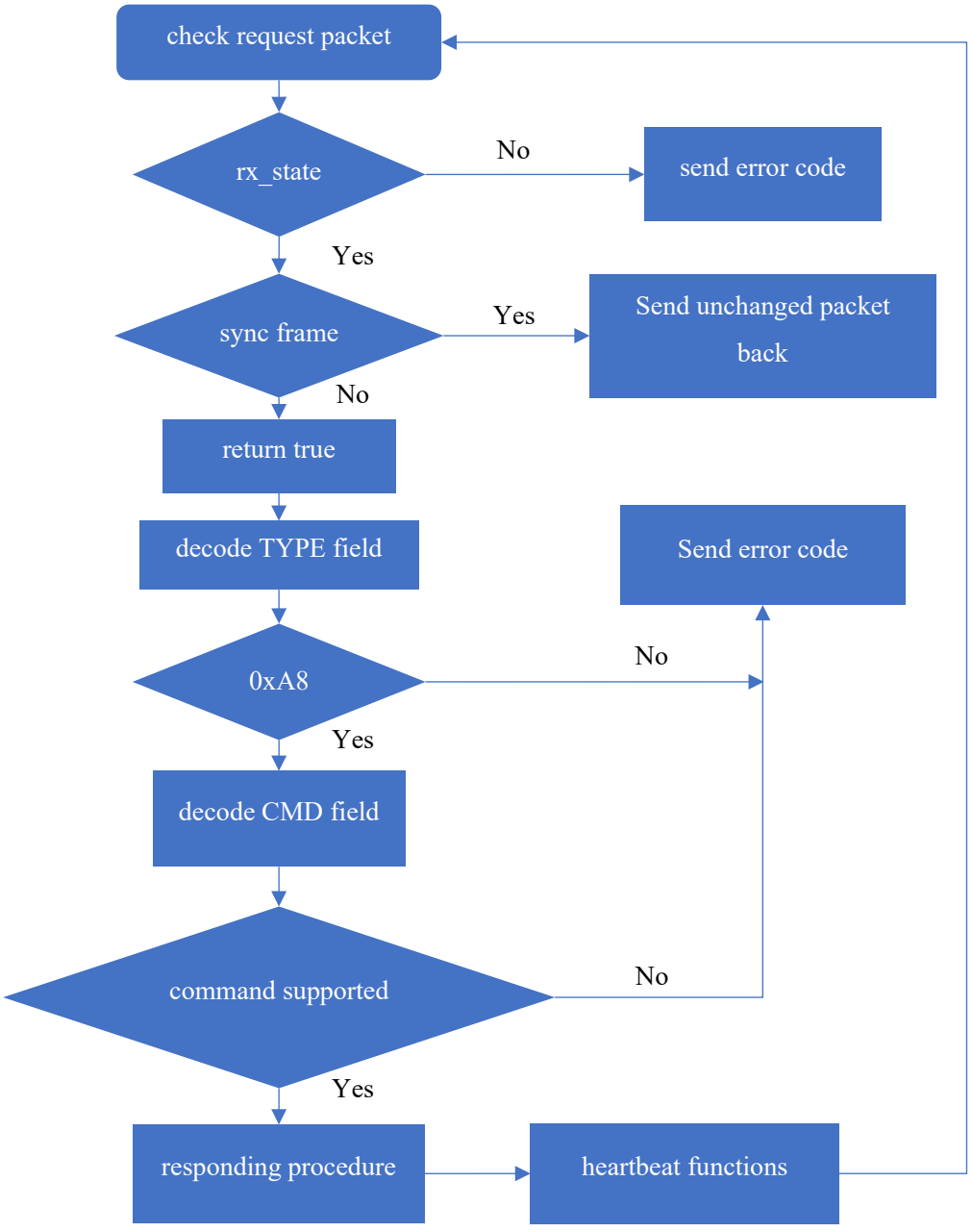


Figure 3-5 Decoding and responding.

When the slave has received the request packet, it needs to parse the master's command and initiate a response procedure according to the command, as shown in Figure 3-5. The slave needs to check the received packet before responding to the master. The program checks rx_state to know the state of the packet. If the slave and master are not correctly synchronized or the checksum comparison indicates that the data packet is compromised, the slave will send the error code to the master. The slave will send the unchanged data packet back to the master if the command field of the packet is SYNC byte. Before an official data transmission begins, synchronization between two parties needs to be performed to ensure each one is aware of the situation. The SYNC command is used to synchronize the master and slave, ensuring the connection is OK. Otherwise, we extract the value of the TYPE field and compare it with 0xA8. If they are equal, we can further decode the packet; otherwise, we need to send an error code to the master. When the decoding and responding are done, heartbeat functions will be executed to update the peripheral devices.

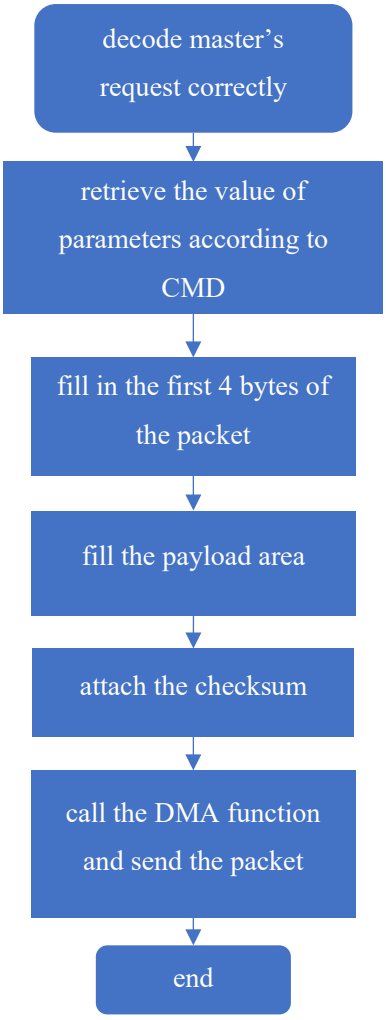


Figure 3-6 Packet assembling.

The preceding sections contain the contents regarding data reception through an interrupt, state transition, and concrete progress about how slave responds to the master's requests. Figure 3-6 walks along with detailed operations of the progress. For example, how to respond to the master's request that acquires the motor's speed or status of infrared sensors. They all conform to the operations demonstrated in Figure 3-6. Firstly, the slave decodes the request packet and extracts the value of the CMD field. Then the slave needs to match the value with a list of commands supported in the communication protocol. If the value is out, the slave will send an error code to the master. The slave will start assembling the packet if the value is on the list. The first four bytes will be filled in the response packet according to the length of the data to be transmitted. Then comes the payload area, and the checksum that is calculated is attached to the end of the response packet. Finally, DMA is initiated to transmit the response packet.

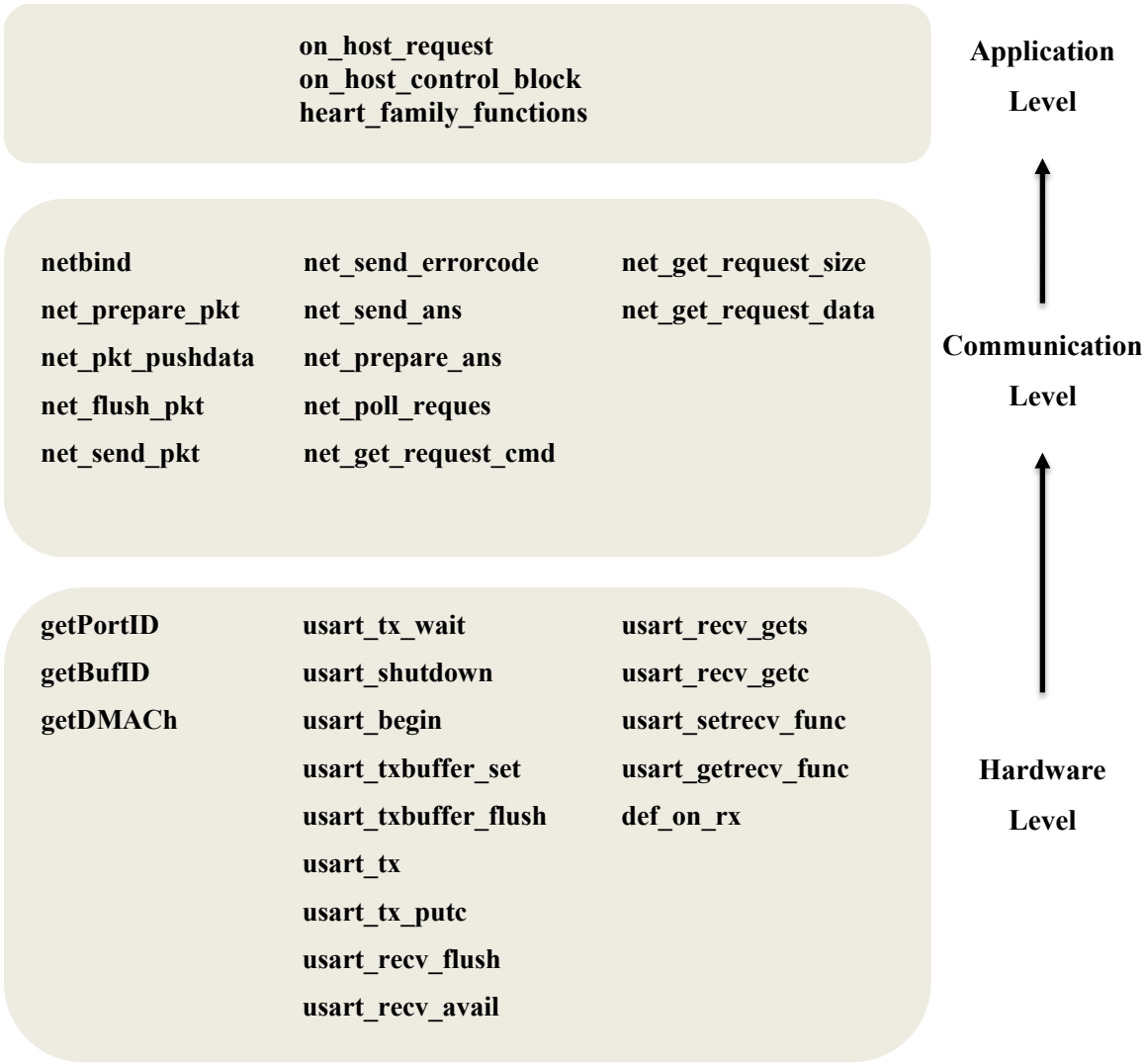


Figure 3-7 Software hierarchy of the communication protocol.

Figure 3-7 shows the hierarchical relations between functions in different layers. Three levels are displayed in Figure 3-7: hardware level, communication level, and application level.

The functions located at the hardware level directly manipulate hardware. Data reception starts from this level, while the data transmission will finally reach this level. At the communication level, functions complete the divided tasks of the communication protocol. The top-level runs general tasks by invoking low-level functions and realizing the specific task's logic.

3.4 Handling of Peripheral Devices

The communication protocol defines multiple commands that the slave should support. In this section, several representative procedures will be discussed.

3.4.1 Battery Monitoring

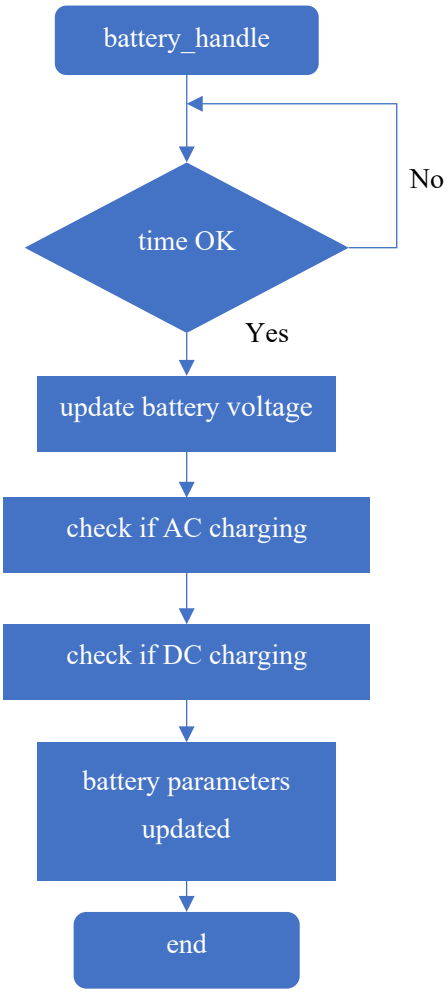


Figure 3-8 Battery monitoring flow.

The software handles the battery every 3 seconds because the battery is not rapidly changing in operations. When the battery is full, its voltage is 16.8V. If the voltage is below 12V, it means the robot needs to be charged. The robot can be charged with AC power or DC power. Two values will be returned. The first value stores the percentage of the battery ranging from 0 to 100. This value is calculated through the ADC, and the input battery voltage signal is divided to meet the voltage reference of the MCU. The second value stores the status indicating if the battery is being charged. The above two values will be packaged and sent to the master if the slave receives a corresponding command from the master.

3.4.2 Ultrasonic Sensors Monitoring

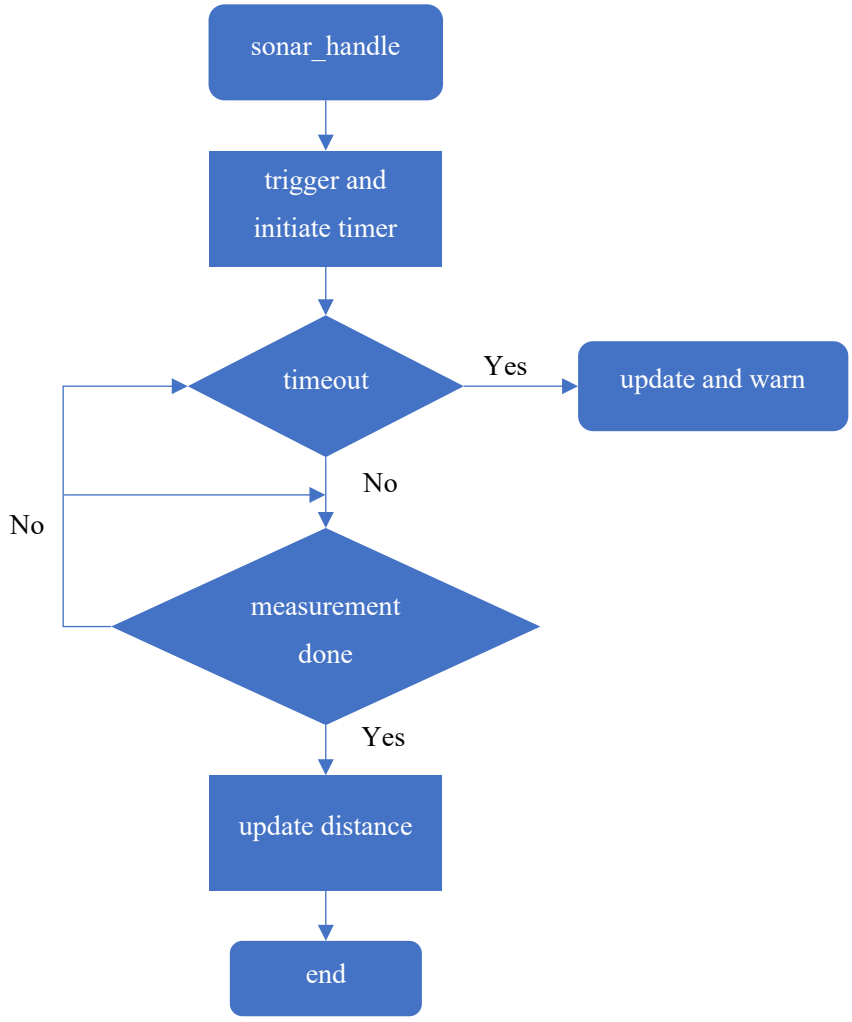


Figure 3-9 Ultrasonic sensors monitoring flow.

The number of ultrasonic sensors is configurable, and the slave's responding procedure discussed in section 3.3.3 will take care of the polling of all of them. When the sensor is triggered, a line interrupt is enabled at the same time. The sensor sends a series of 40KHz

square waves and detects the reflected waves. A timer will initiate to estimate the Echo pin's high voltage duration when the pin's voltage rises from low to high. Calculation of the distance starts when the measurement finishes; that is when the echo pin's voltage drops from high to low. The distances measured by ultrasonic sensors and infrared sensors are grouped if the master needs the original 'distance.'

3.4.3 Cliff Infrared Sensors and Bump Monitoring

As mentioned in 2.3.3, the robot detects collisions with the help of photoelectric switches. The feedbacks of the switches are organized in the form of a bitmap. It is simple to get the results, reading the values of GPIOs connected to the switches separately. The infrared sensor's output is a voltage signal, and there is a corresponding relationship between the voltage and the distance. MCU retrieves the values from ADC at 't' measurement and 't+1' measurement and then calculate the difference. After this, the difference will be compared with a threshold set already. The threshold corresponds to the value calculated when the maximum distance is reached and is pre-fixed in the data section. In typical cases, the ADC values can be considered constant within a specific range because the disinfection robot works on a flat floor. If these values exceed a threshold, the robot might have moved to the edge of a stair. The results from infrared sensors and photoelectric switches are organized as a bitmap. As displayed in Figure 3-11 , a 32-bit variable is segmented, and the results are stored as 1 or 0 in the appointed segment. We do not have to store the distance value; instead, we store the binary judgment. When more sensors are needed, the reserved segment can be activated.

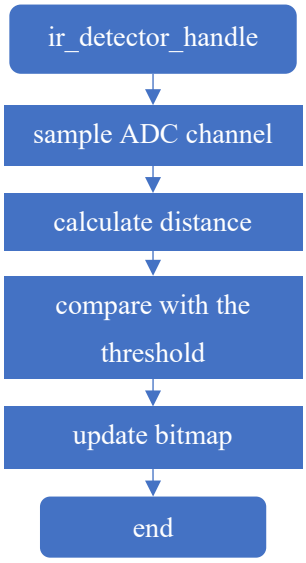


Figure 3-10 Infrared sensors and bump monitoring flow.

Figure 3-11 Bitmap format.

3.4.4 Link Monitoring

The control board and the navigation board are always physically connected when the robot is in working order. Correct data exchange needs a stable and deterministic link between the control board and the navigation board. Thus, the software needs to check the link often. A tick variable is used globally in the software and incremented every 1ms by a timer interrupt. Every time the slave decodes the data packet sent from the master; a variable will store a value that represents the duration since the board was powered-on. The other variable will update the value when the device loop starts. The robot will stop moving if the difference between the two variables' value is larger than a threshold. It is not reasonable to let the robot continue to move when the link between the administrative unit and the navigation unit is missing.

3.4.5 Atomization Control and Motor Control

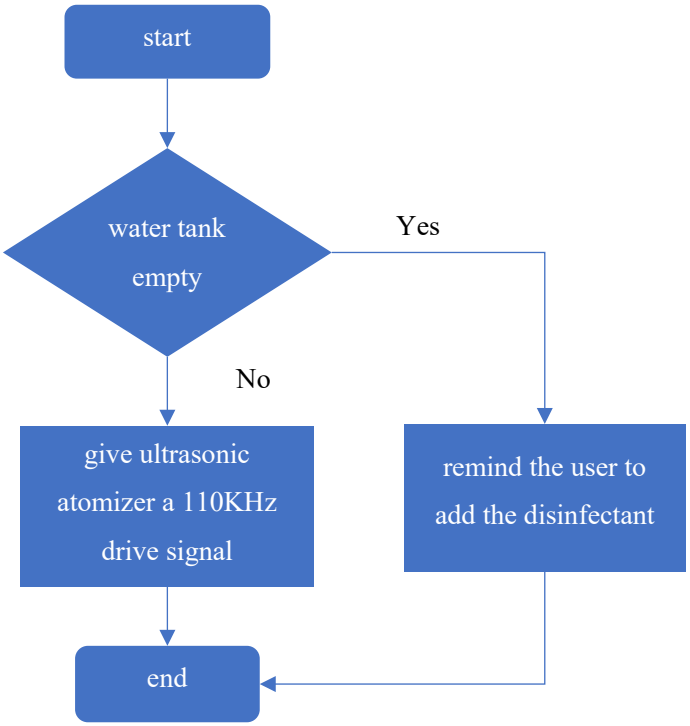


Figure 3-12 Atomization control flow.

The water tank holds the disinfectant, and the sponge stick touches the microporous atomizing sheet. The atomizer breaks up the liquid's molecular structure and produces a natural mist without heating or adding any chemical reagents. As shown in Figure 3-12, the program first checks if any disinfectant is left in the water tank by invoking the function to get the liquid level from the liquid-level detecting module. The robot should remind the user to add some disinfectant if the water tank is empty. In the disinfection robot project, it is simple to drive the ceramic atomizer. The atomizer connects to a drive board, and the drive board connects to the interface on the control board. There are two signals of the interface: +5V and GND. The atomizer needs a signal of a specific high frequency to produce the disinfectant mist. The MCU outputs a square wave of 110KHz, with 1KHz tolerance, to turn on and off the power switch, driving the atomizer.

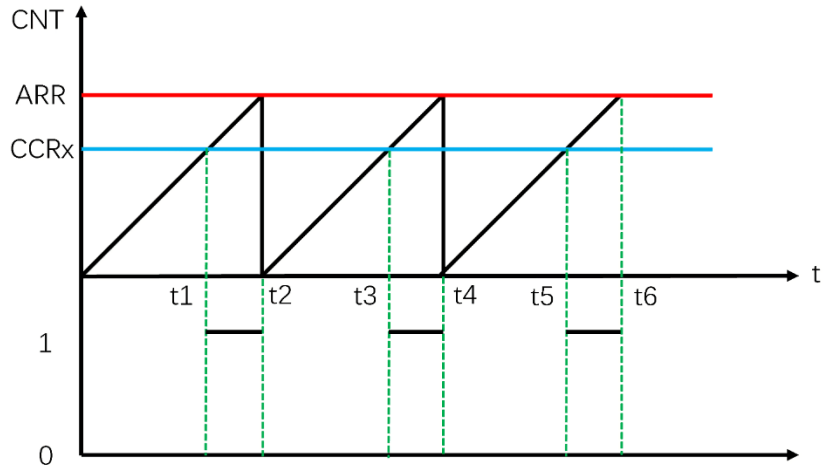


Figure 3-13 PWM setting.

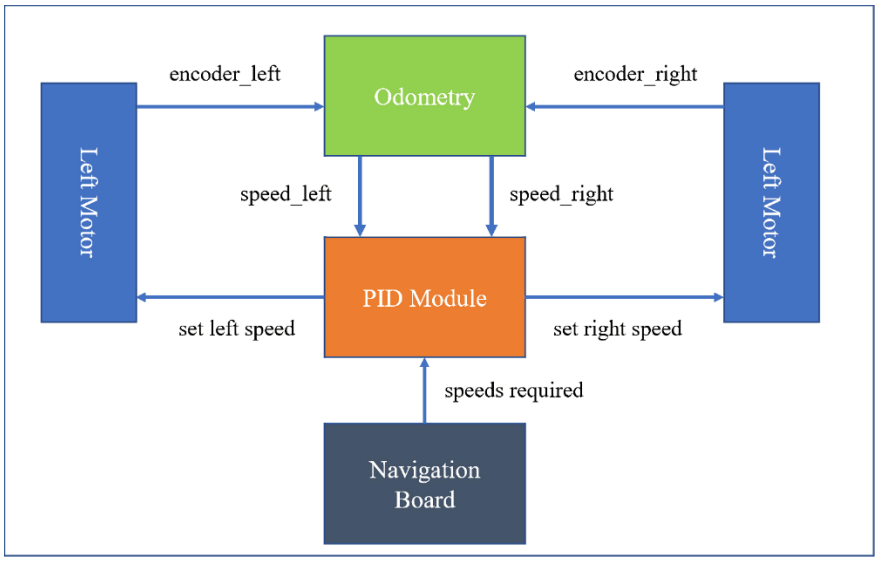


Figure 3-14 Speed adjustment.

The PWM signals drive the motors, and the speed of the motors can be adjusted by alternating the duty cycle of the PWM signals. We need to use the PWM generation function of the advanced timer in STM32 to fulfil it. As shown in Figure 3-13, when the timer's counter is set as an up-counter, the GPIO remapped to the timer channel outputs a low-level signal before the counter value reaches the value in the CCRX register. When the counter value is larger than the value in the CCRX register and less than the value in the ARR register, the GPIO outputs a high-level signal. The frequency of the PWM signal is determined by the ARR register, and the duty cycle is determined by the CCRX register. If we use TIM1 as the timer, the duty cycle can be calculated as (TIM1_CCRX / TIM1_ARR) * 100.

Speed adjustment of the robot is displayed in Figure 3-14. The current speed of the motor can be calculated through the encoder. The motor handling function is refreshed at a short time interval. The number of pulses generated by the encoder for one meter's shift can be found in the motor manufacturer's manual. The count of the pulses during the interval is stored in a variable, and a simple division can calculate the speed at the computing moment. After that, the variable will be set to 0 for the next computation. The calculated speed will be compared with the required speed sent from the navigation board. We use the incremental PID control algorithm to adjust the motors' actual speed to approximate the required speed. The mathematical expression of the incremental PID control algorithm is shown in equation 3-1. Before adjusting the PWM duty cycle, we need to pay attention to two exceptional cases. If the required speed is zero or the motors need to change the moving direction, the motors should first stop. We compute the difference between the desired speed and actual speed and store it into a variable, 'speed_current_error_p.' The second variable, 'speed_current_error_i,' accumulates the difference calculated every time so far. The third variable, 'speeed_current_error_d,' stores the difference between the difference at two consecutive moments.

$$u(k) = K_p e(k) + K_i \sum_{i=0}^{k} e(i) + K_d[e(k) - e(k-1)] \tag{3-1}$$

The value calculated by the incremental PID control algorithm is the duty cycle of the required PWM signal. However, we need to store the complement of the duty cycle to the CCRX register because it controls the duration of the low-level voltage signal.

3.5 Summary

This chapter discusses the embedded software design of the control board. It starts with conventions that the software should comply with. It is inefficient to process the data from sensors or motors and respond to the master, respectively. Thus, a communication protocol is needed to unify the format of data transmission. The details of the protocol, as well as its software realization, are covered. The explanation of a critical structure used in the software is given for better illustration. The peripheral devices are handled in a loop function. Firstly, the slave checks if there arrives a data packet from the master and decodes it. Then, the slave will respond to the master according to the meaning of the command.

At last, the device loop function will initiate to take turns to process the peripheral devices. It takes different methods to process these devices. For example, the program handles the cliff infrared sensor and wall-detecting infrared sensor in a distinctive manner. This procedure is executed in a loop until the robot is powered off or out of battery.

4 2D SLAM Algorithms and ROS

The disinfection robot's hardware can be divided into three main parts: control board, navigation board, and product structure. The board design has been elaborated in chapter 2, so is the embedded software design of the control board based on STM32F103 in chapter 3. This chapter will introduce the practice of 2D SLAM algorithms like Cartographer under the framework of ROS.

4.1 Introduction to ROS

ROS (Robot Operating System) is an operating system that is applied to robots. It is flexible, powerful， and particularly applicable to complicated scenarios such as robots with multiple nodes and multiple tasks. It provides a set of tools, libraries, and drivers to help develop robot applications with hardware abstraction[11]. ROS is popular in academia and industry and is widely used in robotic arms, mobile chassis, drones, and crewless vehicles.

Robotics is a system engineering, which involves many disciplines such as mechanics, electronics, control theory, communications, and software engineering. In the past, it took much effort to develop a robot. The developers need to design the robot's mechanical structure, draw circuit board, design drivers and communication protocols, assemble and package and write various perception and control algorithms. Each task costs massive time, and it is nearly impossible to complete a robot project merely by an individual.

However, with technological progress, the robot industry's labour division begins to move towards becoming multi-level and concrete. Motors, chassis, lidar, cameras, robotic arms, and other components are produced by a series of manufacturers specializing in certain areas. The social division of labour has accelerated the development of the robot industry. The integration of various components requires a unified software platform, which is the robot operating system ROS.

ROS is a framework for robot programming that binds loose components together, providing uniform communication architecture. Unlike Windows, Linux, or Mac, ROS is not a conventional operating system; it just connects the robot program to the OS. It is more like a middleware. Perception, decision policy, and control algorithm can be better organized and run in ROS.

There have some prominent features of ROS. ROS uses a distributed framework that allows the robot's process to run separately on different platforms. The point-to-point design makes it easy for module modification and custom design, thus improving the system's fault tolerance. ROS supports various programming languages such as C++, Python, Lisp, C#, and Java. Moreover, ROS has a vast community where the developers can find packages for different uses and search for parameters for stack or problems.

The disadvantages of ROS include limited real-time communication and stability for industry use. It is open source but supports Linux only.

4.2 Communication Architecture of ROS

4.2.1 Node & Master

Node is the smallest process unit in the world of ROS. There are multiple executable files in a software package. Each executable file corresponds to a node when it runs. Conventionally, each node takes care of one specific function of the robot. For example, we run node1 to drive a camera to capture images, node2 to drive lidar, and node3 to conduct route planning according to the sensors' information. In this way, we can lower the possibility of system failure. In a real robotic project, the robot is composed of numerous components, and it will initiate and run many nodes at the same time. A master node is needed to manage all those nodes. A node registers at the master node, and then it is included in the ROS program by the master node. The master node also links Point-to-Point communication between two nodes. This kind of relation is depicted in Figure 4-1.

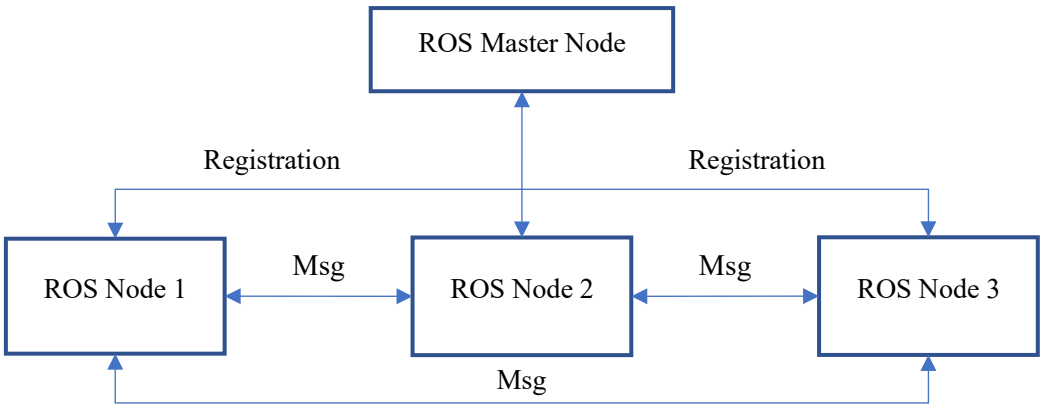


Figure 4-1 Master and node in ROS.

4.2.2 Topic

The topic is the most frequently used communication method in ROS. It is an ideal choice for transferring real-time, periodic messages. Suppose we have two nodes: publisher node and subscriber node. Both nodes have registered at the master node. The publisher node publishes particular messages to topic 1, and the subscriber node can then receive the message with the master node's dispatch. It is unilateral communication, as described in Figure 4-2.

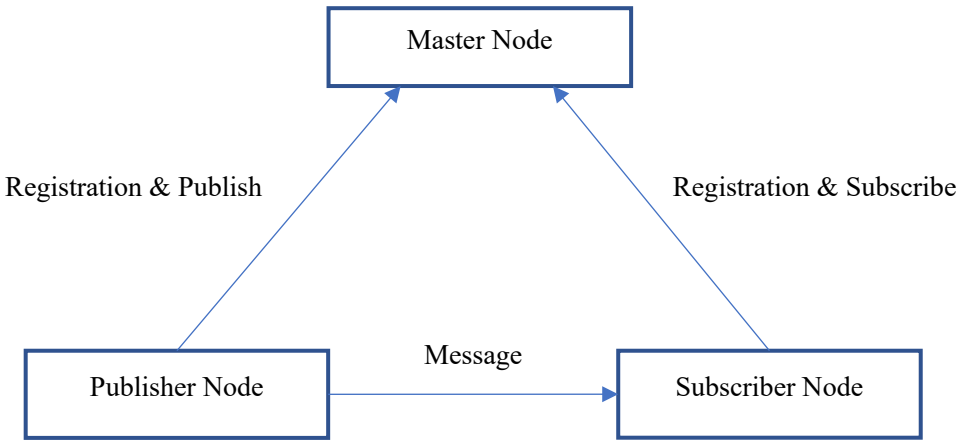


Figure 4-2 Topic in ROS.

4.2.3 Service

We have known that, in ROS, the topic is a mechanism for unilateral asynchronous communication. However, this mechanism cannot meet specific requirements sometimes. For example, some nodes temporarily need some non-periodic data, and it would cost massive unnecessary resources if using a topic that causes high power consumption. Service can handle this kind of problem correctly. In ROS, service is bidirectional. It can not only send messages but also accept the reply. There are two parties in service: the client and the other is the server. The client would send a request, waiting for a reply when the server handles the request. This mechanism is shown in Figure 4-3.

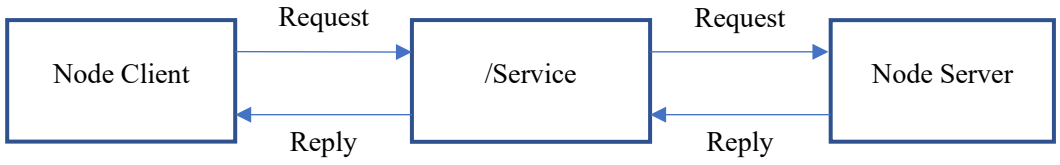


Figure 4-3 Service in ROS.

4.2.4 Parameter Server

The topic and service have been discussed above. Unlike those two mechanisms, the parameter server is a particular communication mechanism. The particularity lies in that the parameter server stores and configures the parameters and share those parameters globally. The distinction between parameter server and topic or service is that it is static. It maintains a data dictionary where stores all sort of parameters and configurations.

4.2.5 Action

The principle of action is working through client-server mode. It is a bilateral communication mechanism. There are two parties involved in communication exchange via Action Protocol in ROS. The concrete communication is implemented using an interface, which is shown in Fig 4-4[33].

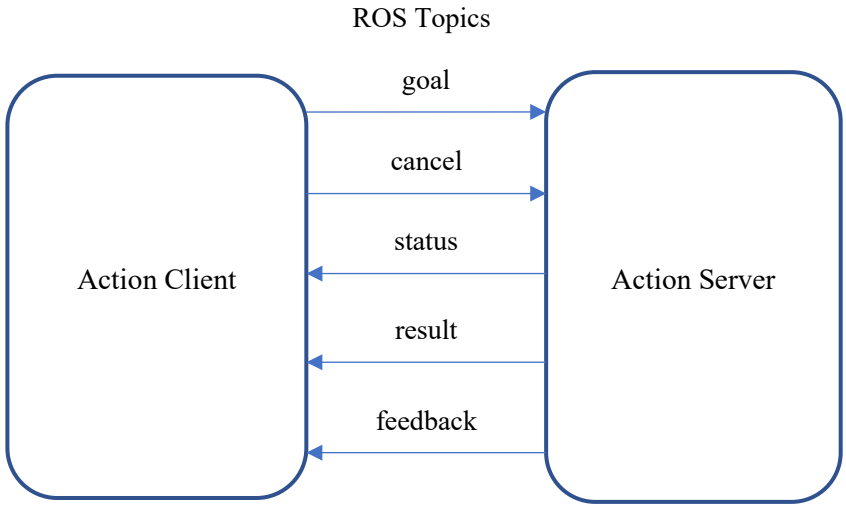


Figure 4-4 Action mechanism in ROS.

4.3 Classical 2D SLAM algorithms

(1) Gmapping

Gmapping is the most widely used laser-based 2D SLAM package in robots. It mainly uses Rao-Blackwellized Particle Filters[34], and it improves the accuracy of Gmapping by making two modifications on RBPF: adaptive resampling and improved proposal[34]. Gmapping can build indoor maps in real-time, requiring fewer computation resources while gaining higher precision. Compared with Hector SLAM, it does not need a lidar with high frequency and is more robust. In the scenario of creating small maps, Gmapping does not require too many

particles and loop closure detection[35], resulting in fewer computations compared with Cartographer. However, accuracy is still comparable to Cartographer. As the scale of the scene increases, the number of particles also required increases. The amount of memory and calculation correspondingly increases when building large maps. It has no loop closure detection, which would cause misalignment when the loop is closed. Although increasing the number of particles can close the map, it is at higher memory and calculation consumption. Gmapping is not an ideal option for building large maps, and it reaps fair results when mapping in long corridors and environments with standard features.

(2) HectorSLAM

Hector slam has relatively high demands for sensors; it mainly uses the Gauss-Newton method to solve the scan-matching problem. It combines a 2D SLAM system based on robust scan matching and a 3D navigation technique using an inertial sensing system[36]. The odometric information is not used, so it is applicable to aerial drones and ground vehicles to operate in uneven terrains[37]. The obtained map can be used to optimize the laser beam dot-matrix, estimate the representation of the laser beam point on the map, and calculate the probability occupying the grid. Gauss-Newton method can be used to obtain the rigid body transformation of the laser point set mapped to the existing map. To avoid the local minimum, we can use a multi-resolution map.

The disadvantage is that a lidar with high update frequency and low noise is needed in the system. Mapping results will be satisfied if the speed of the robot is low. On the other hand, it does not leverage the odometry data even though the data is reasonably accurate.

(3) KartoSLAM

KartoSLAM is based on graph optimization using a highly optimized and non-iterative Cholesky matrix to decouple sparse systems as its solver[38]. The graph optimization method uses the mean value of the graph to represent the map, and each node represents a pose of the robot along its trajectory. The connection of the arcs indicates the movement between successive poses[37]. When each new node is added, the map will be calculated and updated according to the node arcs' constraints in the space. In the ROS version of KartoSLAM, Spare Pose Adjustment(SPA)[39] is responsible for scan matching and loop closure. As the number of landmarks increases, the amount of memory soars. However, the graph-based method is better in maintaining large-scale maps compared with other methods.

(4) LagoSLAM

LagoSLAM is a method of linear approximation for graph optimization in which an initial guess is not needed. There are three options for optimizers: netwORK Optimizer (TORO), g2o[40], LAGO.

The basis of graph-based SLAM algorithms is the minimization of a nonlinear non-convex cost function[37, 41]. At each iteration, the initial problem of local convex approximation is solved to update the graph configuration. The process is iterated a certain number of times until the cost function reaches a local minimum.

(5) CoreSLAM

CoreSLAM is created to be simple and make the SLAM algorithm quickly understood with minimum loss of performance. It is a wrapper for a 200-line-code tinySLAM algorithm. It simplifies the algorithm[37, 42] into two steps. The first step calculates the distance based on a simple PF algorithm. The particle filter matcher is used to match the laser with the map. Each particle filter represents a possible pose of the robot and has an associated weight[37]. After selecting the best hypothetical distribution, the particles with low weight disappear, and new particles are generated. In the update step, scan lines are added to the map. When an obstacle is detected, the algorithm draws an adjustable set of points surrounding that obstacle.

4.4 Cartographer

In the disinfection robot project, we have designed the hardware and product structure. Our purpose is to implement a complete system and endow it with essential functions. We choose Cartographer, an algorithm released by Google in 2016, as our SLAM scheme. This section will analyze the algorithm and its software structure and demonstrate how the Cartographer is integrated into the disinfection robot's software structure in later sections. The main reason we choose Cartographer as our SLAM algorithm is that it does not have the critical demands of hardware, so that we can build the disinfection robot at a lower cost. Moreover, its code is standardized and engineering, which is fit for commercial use and secondary development.

The software structure of the Cartographer mainly contains two parts: cartographer and cartographer_ros. The cartographer handles data from the laser, IMU, odometry, and other sensors. It builds maps with these data. This part of the code is the core of the Cartographer. The cartographer_ros is a software package based on ROS that acquires data from sensors and converts them to the form conforming to cartographer regulations. Then the converted data is

transferred to the cartographer to be processed. The cartographer_ros is more like an upper-level application based on the cartographer. Figure 4-5 demonstrates such a relation.

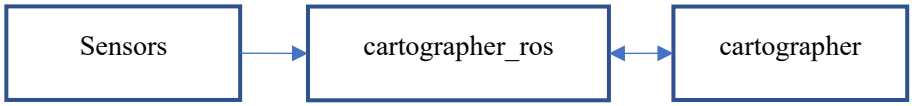


Figure 4-5 Relation between cartographer_ros and cartographer.

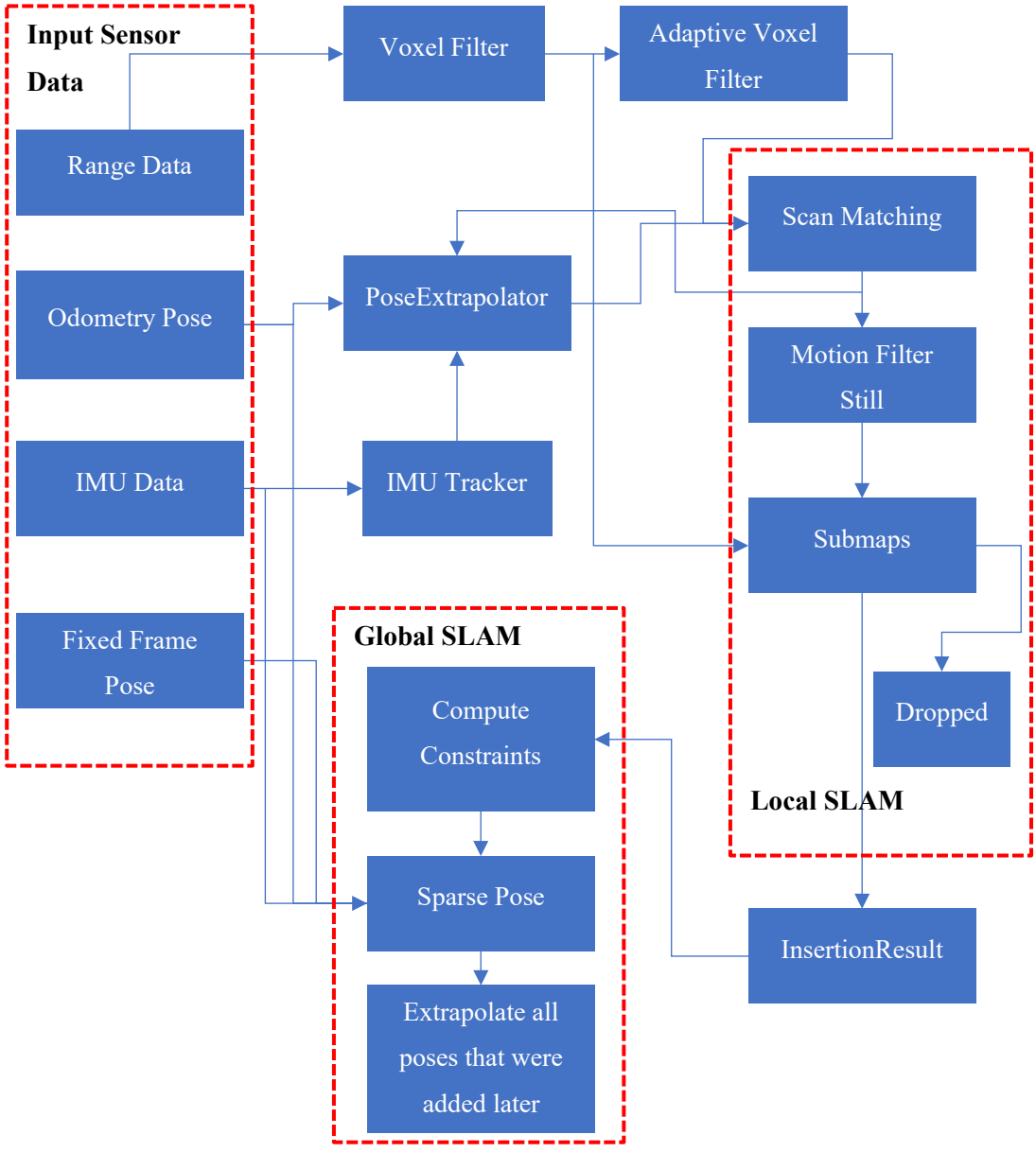


Figure 4-6 Software structure of cartographer.

Figure 4-6 shows that the cartographer can be seen as two separate but related subsystems: local SLAM and global SLAM.

Local SLAM is also called Frontend, and its main task is to collect data and build submaps. Scan-to-scan matching can be used to calculate relative pose changes, for example[39, 43-45]. However, it quickly accumulates errors. Scan-to-map matching, if a good initial estimate of the pose is provided, limits error accumulation. The Cartographer adopts the scan-to-submap matching. A submap is constructed by a few consecutive scans[35]. Before inserting a scan into a submap, we need to optimize the scan pose ξ relative to the local current map. Ceres[46] scan matcher is used to accomplish this. The matcher finds a scan pose that maximizes the probabilities at the scan points in the submap. This problem is treated as a nonlinear least-squares problem. Mathematically, it corresponds to equations 4-1 and 4-2[35]. Scan matching inevitably involves accumulating errors that will be resolved in global slam.

$$T_\xi p = \underbrace{\begin{pmatrix} cos\,\xi_\theta & -sin\,\xi_\theta \\ sin\,\xi_\theta & cos\,\xi_\theta \end{pmatrix}}_{R_\xi} p + \underbrace{\begin{pmatrix} \xi_x \\ \xi_y \end{pmatrix}}_{t_\xi} \tag{4-1}$$

$$\underset{\xi}{\operatorname{argmin}} \sum_{K=1}^{K} \left(1 - M_{smooth}\left(T_\xi h_k\right)\right)^2 \tag{4-2}$$

Global SLAM is also called Backend, and it is responsible for closing loops. It first checks whether there is a loop closure. This process applies ceres scan matching, and it avoids sinking into a local minimum that might happen in ICP[21] or PLICP[47], which performs brute-force search. The accompanying defect is immense calculations. However, the Cartographer can use branch-and-bound scan matching to speed up the process. Then it needs to handle the optimization problem of pose constraints after the loop is detected. The optimization problem is still formulated as a nonlinear least-squares problem.

Overall, local SLAM tries to build submaps with good quality, and global SLAM performs global optimization to tie a different set of submaps together with the best poses. Building a global map based on submaps can effectively avoid the interference of moving objects in the environment. From the software perspective, sensors send out data, and the data pass through two filters ending in the scan matching phase. These input data are used to build submaps. When a new laser scan comes, it will be inserted into the appropriate place in submaps being maintained. How the scan is inserted with the optimal poses depends on the strategy of ceres scan matching. This pose will be integrated with other sensors' data to estimate the next pose. CeresScanMatcher is the realization in the software of the strategy. The other crucial compartment of CeresScanMatcher is RealTimeCorrelativeScanMatcher, which conforms to

the strategy described in[48]. It can be enabled if there have no other sensors, or the developer does not trust them.

4.5 Software Architecture of the Disinfection Robot with Raspberry Pi

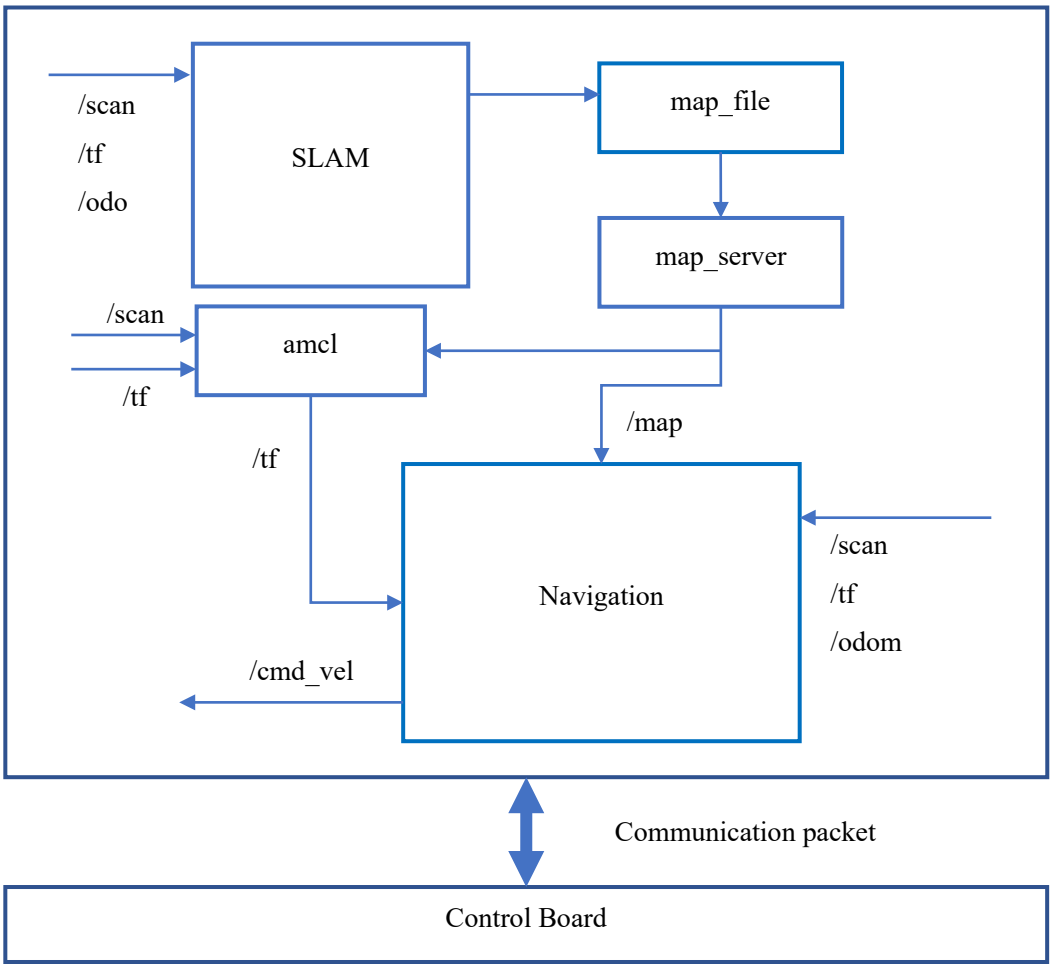


Figure 4-7 Software structure when using Raspberry Pi as the navigation board.

In chapter 2, we discussed the details of the control board. The control board groups all the sensors and motors together, transmitting data in a consistent form. The design's advantage lies in that there is only one node corresponding to the control board abstracted in ROS. We do not have to create multiple nodes in ROS, handling multiple data streams from different sensors. The same node can handle all the modules connected to the control board. The navigation board can extract the sensors' data if it complies with the communication protocol.

The SLAM block builds maps by processing the data from lidar, odometry, or other sensors. When the map is built entirely, we can invoke the service '/write_state,' provided by cartographer_ros, to save the map. However, the saved map is in '. pbstream' format. It needs

to be converted to the GridMap format that is universally applied in ROS. The converted map can be used by the map_server block directly.

One of the most powerful features of ROS is navigation[49]. The ‘move_base’ node is the final executive module of navigation control. The module takes charge of the scheduling of navigation’s behavior, such as initializing costmap[50] and planner and monitoring the navigation status. It subscribes to the user’s navigation goal. It publishes the real-time movement control signals to ‘/cmd_vel.’ All sorts of navigation algorithms in ‘move_base’ are invoked as plugins so that the algorithms can be replaced according to different scenarios. The ‘global_planner’ in the navigation block takes charge of the global path planning, while the ‘local_planner’ is responsible for local path planning[51]. The node ‘amcl’ implements global positioning with the particle filter algorithm, providing the robot with global location information. The node ‘map_server’ checks the map built by the SLAM block to provide the navigation block with the environment’s information.

Raspberry Pi is integrated with an embedded Linux operating system and a ROS of the kinetic version. It implements the communication protocol discussed in chapter 3. Besides Cartographer, Hector and Karto are used to build maps. The results will be shown and analyzed in chapter 5.

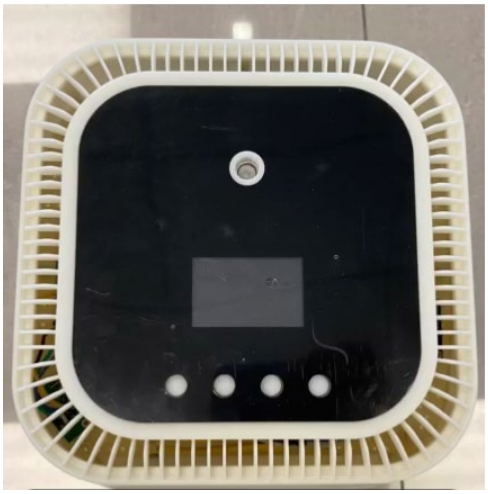
4.6 Summary

At the start, this chapter gives an essential introduction to ROS. The communication mechanisms provided by ROS can significantly enhance the development of robot programs. The distributive features of ROS perfectly conform to the characteristics of robot development. Before Cartographer, several classic 2D SLAM algorithms are briefly discussed and analyzed. The Cartographer can be divided into local SLAM and global SLAM both in algorithm and software realization. The cartographer_ros is more often used as it is integrated into ROS, leveraging the characteristics of ROS. The software architecture of the disinfection robot with Raspberry Pi is given at the end of the chapter.

5 System-Level Operations and Experiment Results

This chapter gives the results of the system-level operations of the disinfection robot. As discussed in previous chapters, two navigation boards are used. SLAMWARE CORE is a modular robot localization and navigation system. It is used to build a complete system. Raspberry Pi is used as a navigation board with the disinfection robot's control board to explore different SLAM algorithms.

5.1 System-Level Operations of the Disinfection Robot



Top panel.

Atomized disinfectant spray.

Figure 5-1 The top part of the robot.

The left part of Figure 5-1 shows the top panel of the disinfection robot, and the round hole on the upper half of the panel is where the dry fog comes out, as displayed in the right part. The intensity of the spray is adjustable.

Figure 5-2 Network connection.

As shown in Figure 5-2, the robot can connect to a mobile phone through a network. Users can use a mobile phone to control the robot to build maps, as shown in Figure 5-3. The left half screen shows the map in real-time. The other half shows the status of the robot and control buttons. Currently, the mobile application is not user-friendly, and it is for developers' use.

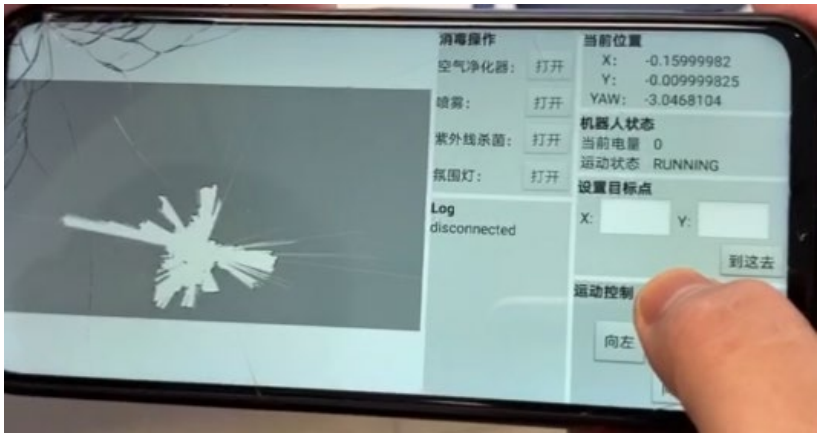


Figure 5-3 Mobile phone application.

Move forward. Move backward.

Move to the left. Move to the right.

Figure 5-4 Mapping with the control of a mobile phone.

The robot is moving, and it is not apparent in static figures. Viewed from a clockwise direction in Figure 5-4, the robot moves forward, backward, to the left, and to the right. At the time of writing, the robot has not supported autonomous mapping.

5.2 Experiment Results

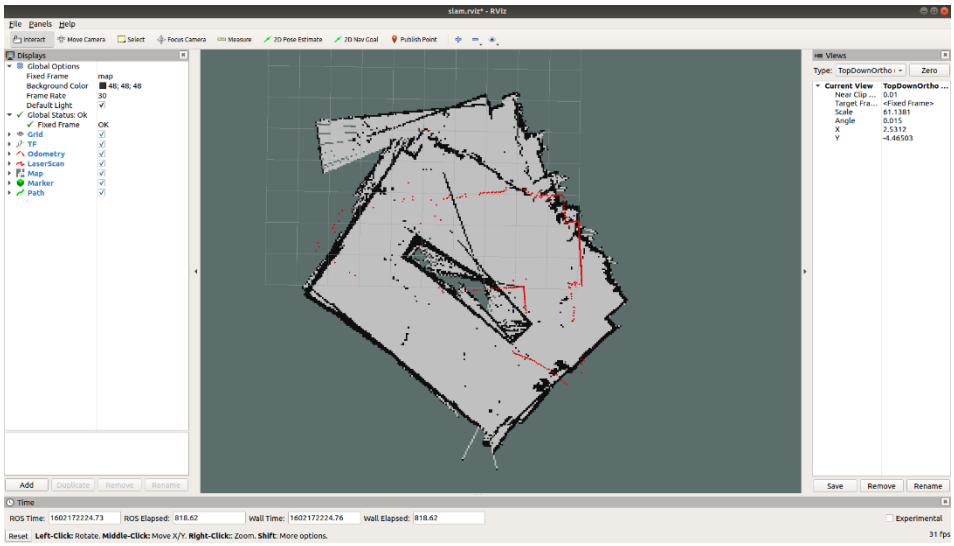


Figure 5-5 Hector mapping.

Figure 5-5 shows the map built by using the Hector SLAM algorithm. Hector does not leverage the feedback from the odometry, and it heavily relies on the lidar. The mapping result is not satisfying because the scanning frequency and the data resolution of the lidar are not high enough for the Hector algorithm.

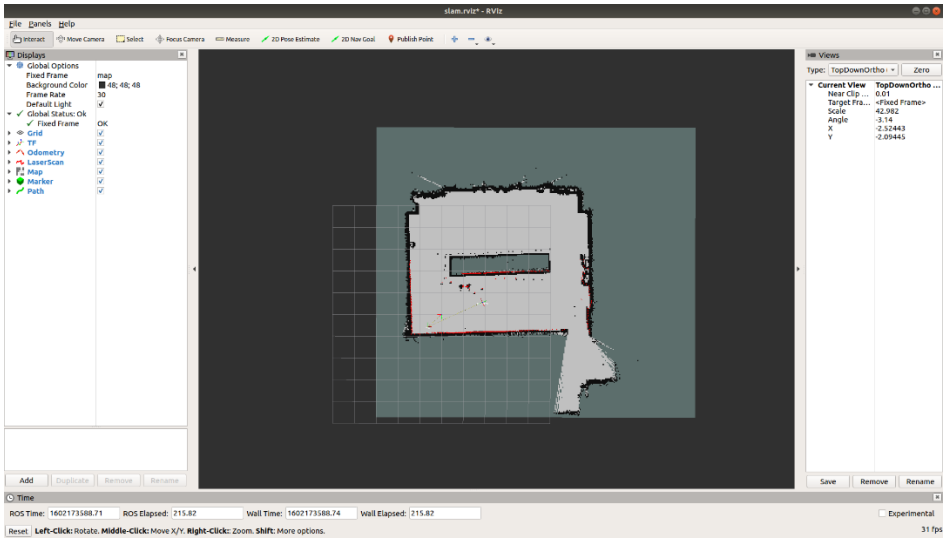


Figure 5-6 Karto mapping.

Figure 5-6 shows the map built by using the Karto SLAM algorithm. Karto SLAM is a graph-based SLAM algorithm, so as the cartographer. From Figure 5-7, we can see that the map

built by the cartographer is more detailed on the boundaries than that built by Karto. The conference room where the experiments were performed is shown in Figure 5-8. The downward right corner in the map corresponds to the door of the conference room, and the door is open during the mapping process. Thus, the map is not closed. The boundary of the conference table in the middle is clearly depicted.

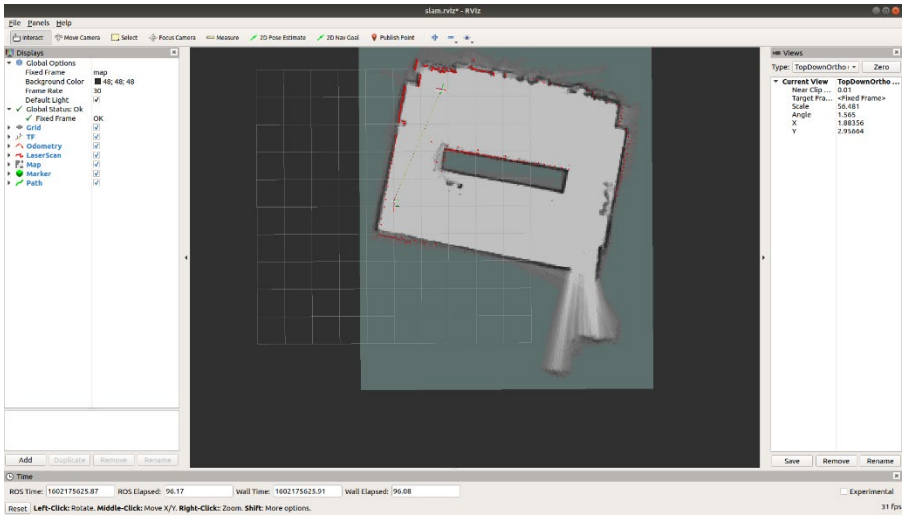


Figure 5-7 Cartographer mapping.

Figure 5-8 The conference room.

5.3 Summary

The disinfection robot works as expected as a system. We can connect to the robot with our mobile phone under the local network. The robot can be controlled to move around to build maps. SLAMWARE CORE is used to accomplish the system design so that the robot can

work nearly as a product. The Raspberry Pi is used with our control board to exercise different SLAM algorithms and observe the mapping results in ROS. The next development plan of the robot is discussed in chapter 6.

6 Conclusion and Prospect

6.1 Conclusion

In this thesis, we propose a feasible design for a domestic disinfection robot and demonstrate the process of implementing it. The main work accomplished in the thesis can divide into three parts.

The first part completes the design of the hardware of the disinfection robot. The control board act as a housekeeper, managing all the peripheral devices except the lidar. The signals output by lidar are connected to the navigation board through wires and edge-board contact. The control board packages the data read from sensors and sends it to the navigation board in a uniform format. We use SLAMWARE CORE as the navigation board to complete the system design. The SLAMWARE CORE is a modular robot localization and navigation system. We apply this module, hoping to get a fully functional robot and perform the system-level test.

The second part realizes the embedded software of the control board. The communication protocol is realized in software enabling robust data exchange between the control board and the navigation board. It analyzes the overall program flow and some critical process handling different tasks.

The third part integrates the control board, navigation board, and other components to form a complete robot. The results of system-level operations of the robot formed by SLAMWARE CORE and our control board are shown. The robot can build maps with a mobile application's control and show them in real-time on the phone screen. The results of mapping experiments using different 2D SLAM algorithms are listed. The experiments were conducted by adopting Raspberry Pi as the navigation board.

6.2 Prospect

We still have lots of improvements to make to perfect the disinfection robot. Control and navigation are the two core parts of the robot. We would build our customized navigation board, from hardware, OS integration to algorithms in the next step. ROS provides convenience for robot program developers. However, is it reliable enough to develop a product for commercial use? How to apply the SLAM algorithms, open-source, or a fresh start? These are issues to be handled in the future.

References

[1] ZHONGTAI SECURITIES. (2018). *Prepared Pathfinders in the Era of Robots*. Available: http://pdf.dfcfw.com/pdf/H3_AP201808021173396553_1.pdf

[2] Prospective Industry Research Institute. (2020). *Analysis of China's sweeping robot market and competitive landscape in 2020*. Available: http://app.cheaa.com/readnews.php?contentid=578914

[3] G. C. Mendoza, "Robot SLAM and navigation with multi-camera computer vision," 2012.

[4] Robotnyheter. (2016). *Electrolux lanserar robotdammsugaren MotionSense*. Available: https://robotnyheter.se/2016/09/18/electrolux-lanserar-robotdammsugaren-motionsense/

[5] *RC 3*. Available: https://www.kaercher.com/int/home-garden/robocleaner/rc-3-11982030.html

[6] Patrick Sinclair. (2020). *Best Robot Vacuum Cleaners for 2020 – The Top Choices This Year*. Available: https://www.allhomerobotics.com/best-robot-vacuums-top-choices-year/

[7] Paul Lamkin. (2020). *Neato D10 is the pick of a new robot vacuum cleaner trio*. Available: https://www.the-ambient.com/news/neato-d10-d9-d8-launch-price-specs-2309

[8] G. Cao, Y. J. r. I. C. o. B. Zhang, and B. Engineering, "Microbial Removal Efficiency in situ Measurement of Exhaust Air HEPA Filters Employed in Cleanrooms," pp. 1-4, 2009.

[9] S. Tsuji and T. Kohama, "Proximity and Contact Sensor for Human Cooperative Robot by Combining Time-of-Flight and Self-Capacitance Sensors," *IEEE Sensors Journal,* vol. 20, no. 10, pp. 5519-5526, 2020.

[10] *Four series of robots produced by Ecovacs Company*. Available: https://www.ecovacs.cn/

[11] M. Quigley, "ROS: an open-source Robot Operating System," in *ICRA 2009*, 2009.

[12] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. Davison, and S. J. T. I. J. o. R. R. Leutenegger, "ElasticFusion: Real-time dense SLAM and light source estimation," vol. 35, pp. 1697 - 1716, 2016.

[13] R. A. Newcombe *et al.*, "KinectFusion: Real-time dense surface mapping and tracking," in *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, 2011, pp. 127-136.

[14] R. A. Newcombe, S. Lovegrove, and A. J. I. C. o. C. V. Davison, "DTAM: Dense tracking and mapping in real-time," pp. 2320-2327, 2011.

[15] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. J. Leonard, and J. McDonald, "Real-time large-scale dense RGB-D SLAM with volumetric fusion," vol. 34, no. 4-5, pp. 598-626, 2015.

[16] P. Henry, D. Fox, A. Bhowmik, and R. Mongia, "Patch Volumes: Segmentation-Based Consistent Mapping with RGB-D Cameras," in *2013 International Conference on 3D Vision - 3DV 2013*, 2013, pp. 398-405.

[17] M. Keller, D. Lefloch, M. Lambers, S. Izadi, T. Weyrich, and A. J. I. C. o. D. V. Kolb, "Real-Time 3D Reconstruction in Dynamic Scenes Using Point-Based Fusion," pp. 1-8, 2013.

[18] J. Chen, D. Bautembach, and S. J. A. T. G. Izadi, "Scalable real-time volumetric surface reconstruction," vol. 32, pp. 113:1-113:16, 2013.

[19] M. Bloesch, J. Czarnowski, R. Clark, S. Leutenegger, A. J. I. C. C. o. C. V. Davison, and P. Recognition, "CodeSLAM - Learning a Compact, Optimisable Representation for Dense Visual SLAM," pp. 2560-2568, 2018.

[20] J. McCormac, A. Handa, A. Davison, and S. Leutenegger, "SemanticFusion: Dense 3D semantic mapping with convolutional neural networks," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 4628-4635.

[21] E. Mendes, P. Koch, and S. Lacroix, "ICP-based pose-graph SLAM," in *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2016, pp. 195-200.

[22] R. Munguía and A. J. M. P. i. E. Grau, "Monocular SLAM for Visual Odometry: A Full Approach to the Delayed Inverse-Depth Feature Initialization Method," vol. 2012, pp. 1-26, 2012.

[23] R. Munguia and A. Grau, "Monocular SLAM for Visual Odometry," in *2007 IEEE International Symposium on Intelligent Signal Processing*, 2007, pp. 1-6.

[24] S. Weiss, D. Scaramuzza, and R. J. J. F. R. Siegwart, "Monocular-SLAM-based navigation for autonomous micro helicopters in GPS-denied environments," vol. 28, pp. 854-874, 2011.

[25] C. X. Guo *et al.*, "Large-scale cooperative 3D visual-inertial mapping in a Manhattan world," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1071-1078.

[26] C. X. Guo *et al.*, "Resource-Aware Large-Scale Cooperative Three-Dimensional Mapping Using Multiple Mobile Devices," *IEEE Transactions on Robotics,* vol. 34, no. 5, pp. 1349-1369, 2018.

[27] T. Qin, P. Li, and S. Shen, "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator," *IEEE Transactions on Robotics,* vol. 34, no. 4, pp. 1004-1020, 2018.

[28] C. Kerl, J. Sturm, and D. Cremers, "Dense visual SLAM for RGB-D cameras," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 2100-2106.

[29] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-Scale Direct Monocular SLAM," in *ECCV*, 2014.

[30] J. Engel, V. Koltun, and D. Cremers, "Direct Sparse Odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 40, no. 3, pp. 611-625, 2018.

[31] SLAMTEC. (2020). *RPLIDAR A1 Low Cost 360 Degree Laser Range Scanner-Introduction and Datasheet*. Available: http://bucket.download.slamtec.com/7fe7e3656e811ab1a645753af40809f05fa7ddcd/LD108_SLAMTEC_rplidar_datasheet_A1M8_v2.4_en.pdf

[32] M. Buonanno, D. Welch, I. Shuryak, and D. J. Brenner, "Far-UVC light (222 nm) efficiently and safely inactivates airborne human coronaviruses," *Scientific Reports,* vol. 10, no. 1, p. 10285, 2020/06/24 2020.

[33] W. Wu, Changkun, Y. Wu, and Rocwang. *China University MOOC: Getting started with the robot operating system*. Available: https://github.com/DroidAITech/ROS-Academy-for-Beginners

[34] G. Grisetti, C. Stachniss, and W. Burgard, "Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters," *IEEE Transactions on Robotics,* vol. 23, no. 1, pp. 34-46, 2007.

[35] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2D LIDAR SLAM," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1271-1278.

[36] S. Kohlbrecher, O. v. Stryk, J. Meyer, and U. Klingauf, "A flexible and scalable SLAM system with full 3D motion estimation," in *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, 2011, pp. 155-160.

[37] J. M. Santos, D. Portugal, and R. P. Rocha, "An evaluation of 2D SLAM techniques available in Robot Operating System," in *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2013, pp. 1-6.

[38] R. Vincent, B. Limketkai, and M. Eriksen, "Comparison of indoor robot localization techniques in the absence of GPS," in *Defense + Commercial Sensing*, 2010.

[39] K. Konolige, G. Grisetti, R. Kümmerle, W. Burgard, B. Limketkai, and R. Vincent, "Efficient Sparse Pose Adjustment for 2D mapping," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010, pp. 22-29.

[40] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 3607-3613.

[41] L. Carlone, R. Aragues, J. Castellanos, and B. Bona, "A Linear Approximation for Graph-based Simultaneous Localization and Mapping," in *Robotics: Science and Systems*, 2011.

[42] B. Steux and O. E. Hamzaoui, "tinySLAM: A SLAM algorithm in less than 200 lines C-language program," in *2010 11th International Conference on Control Automation Robotics & Vision*, 2010, pp. 1975-1979.

[43] E. Olson, "M3RSM: Many-to-many multi-resolution scan matching," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 5815-5821.

[44] F. Lu and E. J. A. R. Milios, "Globally Consistent Range Scan Alignment for Environment Mapping," vol. 4, pp. 333-349, 1997.

[45] F. Martín, R. Triebel, L. Moreno, and R. Siegwart, "Two different tools for three-dimensional mapping: DE-based scan matching and feature-based loop detection," *Robotica,* vol. 32, no. 1, pp. 19-41, 2014.

[46] S. Agarwal and K. Mierle. *Ceres solver*. Available: http://ceres-solver.org

[47] X. Weijun, J. Rongxin, and C. Yaowu, "Map alignment based on PLICP algorithm for multi-robot SLAM," in *2012 IEEE International Symposium on Industrial Electronics*, 2012, pp. 926-930.

[48] E. B. Olson, "Real-time correlative scan matching," in *2009 IEEE International Conference on Robotics and Automation*, 2009, pp. 4387-4393.

[49] S. Gatesichapakorn, J. Takamatsu, and M. Ruchanurucks, "ROS based Autonomous Mobile Robot Navigation using 2D LiDAR and RGB-D Camera," in *2019 First International Symposium on Instrumentation, Control, Artificial Intelligence, and Robotics (ICA-SYMP)*, 2019, pp. 151-154.

[50] D. V. Lu, D. Hershberger, and W. D. Smart, "Layered costmaps for context-sensitive navigation," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 709-715.

[51] B. Cybulski, A. Wegierska, and G. Granosik, "Accuracy comparison of navigation local planners on ROS-based mobile robot," in *2019 12th International Workshop on Robot Motion and Control (RoMoCo)*, 2019, pp. 104-111.

Appendices

Appendix 1 A list of figures

Appendix 2 A list of tables