Build & Design Projecct of Group35

Group Leader of Hardware：Zhang Qihang
Group Member of Hardware：Gao Ziqi、Zhang Yiming、Li Yufeng、Shu Lei、Dai Haotian

Group Leader of Software：Cai Jiacheneg
Group Member of Software：Ye Fei、Zhang Xinpeng、Yang Xiaorui

CATALOGUE

01 Project Overview and Goals

Project Overview and Goals

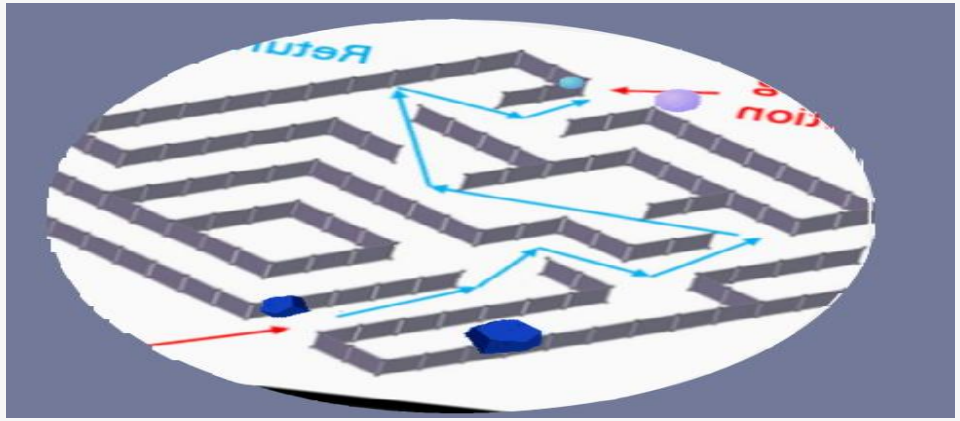
Overview

Development of a 2D LiDAR-Based Autonomous Mobile Robot for Maze Navigation and Real-Time Mapping using Integrated SLAM, Path Planning, and Control Technologies.

Core goal

Hardware Team:
Utilizing the STM32F446RE microcontroller, the hardware team will implement hardware interfaces, acquire sensor data, handle odometry and position estimation, execute real-time motor control, and manage Bluetooth communication.

Software Team：
Simulation environment construction:
Establish a specified map through simulation and develop a complete navigation program.
Path planning and navigation algorithm implementation
It is capable of constructing a complete maze map in real time based on simulation data



02

Project Division and Coordination

Members and division for hardware tasks　50%

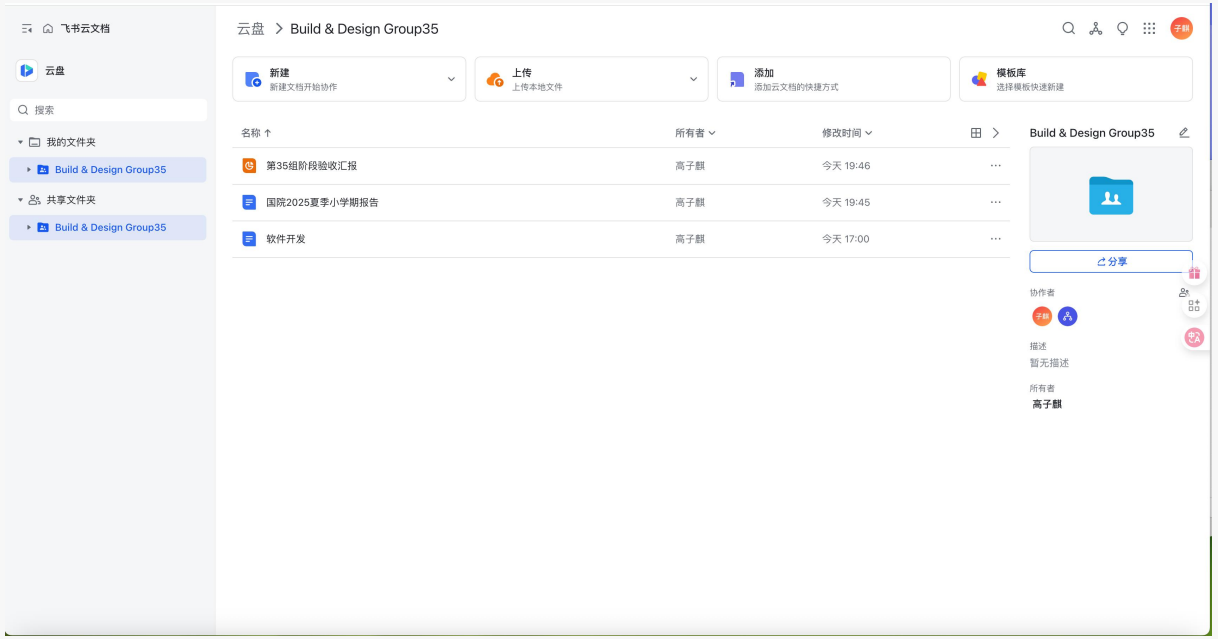
Name	Division
Qihang Zhang（8.3%）	Develop the motor driver.Read 3-axis gyroscope data from the configured MPU and use the attitude information to adjust the position PID loop.Support debugging via an LED display.Assist in acquiring data from the LiDAR sensor.
Ziqi Gao（8.3%）	Implementing the Bluetooth protocol, UART communication, encoder-based velocity measurement & data transmission, velocity PID tuning, and software function integration.
Lei Shu（8.3%）	Assembled the robot chassis, soldered components, managed internal cabling, designed the aesthetics, delivered the final presentation and report, and supported testing activities.
Yiming Zhang（8.3%）	To implement the LiDAR module's basic scanning capability and its data transmission link with the STM32, while also enabling the transmitted data to be monitored and checked from a mobile phone.
Yufeng Li（8.3%）	Define the data packet format for transmission and reception, transmit the data to a mobile app via Bluetooth for display, and debug all data communication.
Haotian Dai（8.3%）	Completed the physical assembly and aesthetic design of the robot, delivered the final report and presentation, gained a comprehensive understanding of the overall development workflow, and contributed to testing activities.

Members and division for software tasks　50%

Name	Division
Cai Jiacheng（12.5%）	Responsible for the integration of path planning and navigation algorithms, completed the coding implementation and integration debugging of the A* algorithm and Dynamic Window Approach (DWA) modules, and developed the Bluetooth communication module. Designed the communication protocol for interfacing with the hardware team and completed the coding of data transmission and reception logic.
YeFei（12.5%）	Responsible for the development of the simulation environment setup, developing the simulation generation module for LiDAR scanning data, implementing the core mapping function based on SLAM technology, processing simulation data and completing the basic logic for real-time map construction.
Zhang Xinpeng（12.5%）	Responsible for system integration and debugging, jointly undertaking the overall system integration testing with Yang Xiaoru, writing test cases to verify the compatibility of each module interface, coordinating and resolving coordination issues among modules, and optimizing the overall operational stability of the system.
Yang Xiaorui（12.5%）	Responsible for GUI development and data processing, developing GUI interfaces, implementing the real-time display function of the maze map, designing a clear visual interaction layout, and handling data processing. Worked with Zhang Xinpeng to handle the connection and debugging with the hardware team.

Coordination

We utilize Lark/Feishu cloud documents to create a knowledge base for sharing key project details and synchronizing daily progress updates.



03

Hardware module implementation

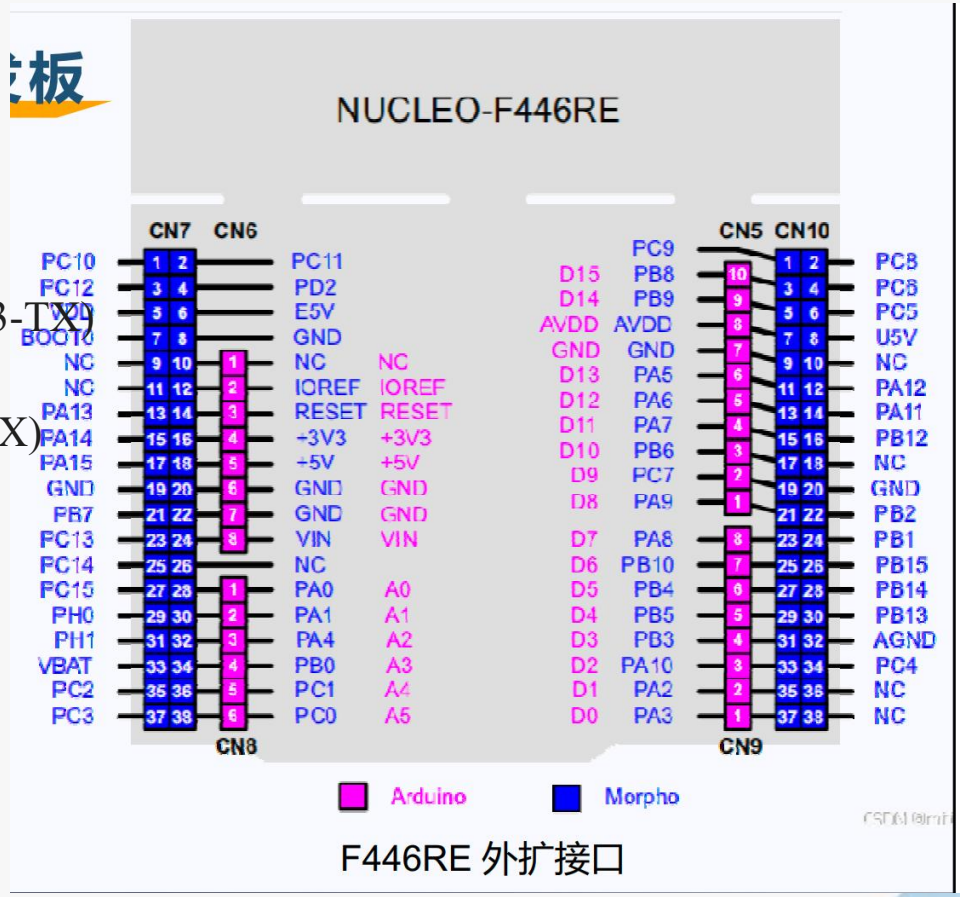
STM32-F446RE

main function

The STM32F446RE serves as the core controller for the robot's movement, integrating various peripherals including a Bluetooth module, encoders, LiDAR, and an MPU. It enables real-time control of the robot by processing movement commands received via the Bluetooth module. During the microcontroller software development process, initialization was carried out using STM32CubeMX to achieve custom pin configuration and functional allocation.

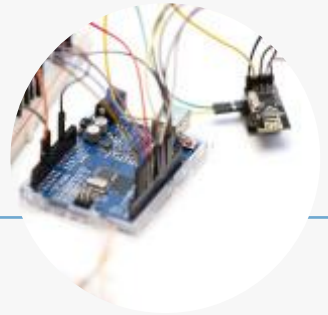
pin assignment

Motor drive：PA6(BIN2),PA7(BIN1),
PB0(AIN2),PB1(AIN1)
Bluetooth:PC5 (USART3-RX),PB10(USART3-TX)
Gyroscope: PB6,PB7
LiDAR：PC10(UART4_TX),PC11(UART4_RX)
Screen debugging interface : PA8,PC9
Encoder Interface ： PC6，PC7



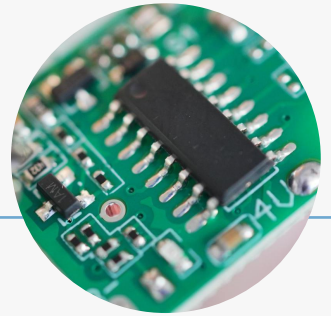
F446RE 外扩接口

Motor drive



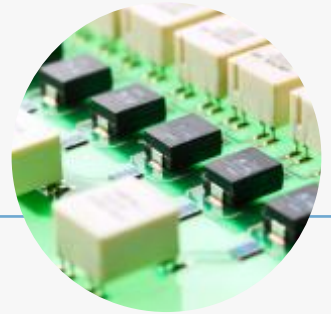
Function

The AT8236 motor driver module, when used with the MC520 motor, achieves precise motion control. It features simple wiring, a clear PWM speed regulation principle, and dual-channel control that supports accurate direction adjustment, providing strong power for applications such as smart cars.



Wiring implementation

The AT8236 module is closely connected to the STM32 pins. AIN1/AIN2 control the steering of the left motor, and BIN1/BIN2 control the steering of the right motor. The VM pin is connected to a 12V battery for power supply, ensuring stable and reliable operation of the motor.



PWM

Motor speed is controlled by regulating the PWM duty cycle. For example, by setting IN1 to output PWM and IN2 to ground, the motor rotates forward and gradually decelerates to a stop. Based on encoder feedback, the MC520 motor generates 390 pulses per revolution of its output shaft. Ultimately, a PID control algorithm, integrated with real-time data from a gyroscope, is employed to achieve precise angular movement.

Motor drive

```
// --- 小车动作函数的实现 ---
//前进
void goForward(uint16_t speed) {
    motorA_goForward(speed);
    motorB_goForward(speed);
}
//后退
void goBackward(uint16_t speed) {
//停止
void stopMotors() {
//左转
void turnLeft(uint16_t speed) {
//右转
void turnRight(uint16_t speed) {
void PID_TurnToLeft(void){
void PID_TurnToRight(void){
// --- 底层单个电机函数的实现 ---
//电机A
static void motorA_goForward(uint16_t speed)
{
    if (speed > PWM_MAX_PERIOD) speed = PWM_MAX_PERIOD;
    __HAL_TIM_SET_COMPARE(&MOTOR_TIM, MOTOR_A_IN1_CHANNEL, speed);
    __HAL_TIM_SET_COMPARE(&MOTOR_TIM, MOTOR_A_IN2_CHANNEL, 0);
}

static void motorA_goBackward(uint16_t speed)
{
static void motorA_stop(void)
{
//电机B
static void motorB_goForward(uint16_t speed)
{
static void motorB_goBackward(uint16_t speed)
{
static void motorB_stop(void)
{
```

通过调用封装好的电机驱动函数实现前进功能

底层利用pwm驱动电机转动

Motor drive

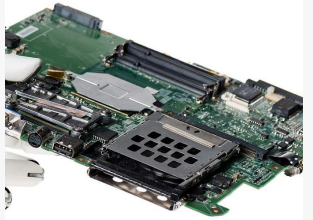
```
// --- 小车动作函数的实现 ---
//前进
void goForward(uint16_t speed) {
    motorA_goForward(speed);
    motorB_goForward(speed);
}
//后退
void goBackward(uint16_t speed) {
//停止
void stopMotors() {
//左转
void turnLeft(uint16_t speed) {
//右转
void turnRight(uint16_t speed) {
void PID_TurnToLeft(void){
void PID_TurnToRight(void){
// --- 底层单个电机函数的实现 ---
//电机A
static void motorA_goForward(uint16_t speed)
{
    if (speed > PWM_MAX_PERIOD) speed = PWM_MAX_PERIOD;
    __HAL_TIM_SET_COMPARE(&MOTOR_TIM, MOTOR_A_IN1_CHANNEL, speed);
    __HAL_TIM_SET_COMPARE(&MOTOR_TIM, MOTOR_A_IN2_CHANNEL, 0);
}

static void motorA_goBackward(uint16_t speed)
{
static void motorA_stop(void)
{
//电机B
static void motorB_goForward(uint16_t speed)
{
static void motorB_goBackward(uint16_t speed)
{
static void motorB_stop(void)
{
```

通过调用封装好的电机驱动函数实现前进功能

底层利用pwm驱动电机转动

Gyroscope

Function



The MPU6500 gyroscope module, integrating attitude tracking and odometer assistance, provides high-precision six-axis data reading capability, and is a key component in applications such as robot navigation and UAV flight control.

Key Implementation

Read the six-axis data of MPU6050, including accelerometer and gyroscope information, through the I2C interface (PB6-SCL, PB7-SDA). Use the official DMP driver library and calculate three angles using the quaternion method.□（PB6-SCL, PB7-SDA）

Data fusion

Combining encoder information, the precise displacement is calculated through a data fusion algorithm and applied to path deviation compensation. This method significantly improves the accuracy and reliability of navigation.

Gyroscope

Official driver library files

```
/*
 $License:
    Copyright (C) 2011-2012 InvenSense Corporation, All Rights Reserved.
    See included License.txt for License information.
 $
 */
/**
 *  @addtogroup  DRIVERS Sensor Driver Layer
 *  @brief       Hardware drivers to communicate with sensors via I2C.
 *
 *  @{
 *      @file       inv_mpu_dmp_motion_driver.c
 *      @brief      DMP image and interface functions.
 *      @details    All functions are preceded by the dmp_ prefix to
 *                  differentiate among MPL and general driver function calls.
 */
```

Obtain the corresponding angle information using quaternion processing

```
int MPU6050_DMP_Get_Date(float *pitch, float *roll, float *yaw)
{
    float q0 = 1.0f, q1 = 0.0f, q2 = 0.0f, q3 = 0.0f;
    short gyro[3];
    short accel[3];
    long quat[4];
    unsigned long timestamp;
    short sensors;
    unsigned char more;
    if(dmp_read_fifo(gyro, accel, quat, &timestamp, &sensors, &more))
    {
        return -1;
    }

    if(sensors & INV_WXYZ_QUAT)
    {
        q0 = quat[0] / Q30;
        q1 = quat[1] / Q30;
        q2 = quat[2] / Q30;
        q3 = quat[3] / Q30;

        *pitch = asin(-2 * q1 * q3 + 2 * q0 * q2) * 57.3; // pitch
        *roll = atan2(2 * q2 * q3 + 2 * q0 * q1, -2 * q1 * q1 - 2 * q2 * q2 + 1) * 57.3; // roll
        *yaw = atan2(2 * (q0 * q3 + q1 * q2), q0 * q0 + q1 * q1 - q2 * q2 - q3 * q3) * 57.3; // yaw
    }

    return 0;
}
```

Encoder

Read the encoder's pulse count per unit time in real-time and convert it to speed units (RPS - revolutions per second).

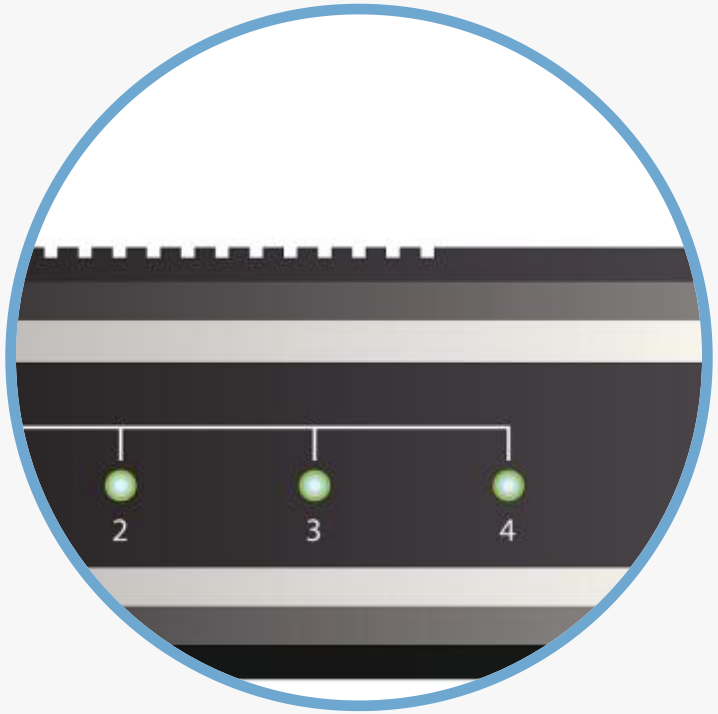
```
// 1. 从编码器模块获取原始脉冲增量
int16_t left_delta_pulses = Encoder_Read_Left();
int16_t right_delta_pulses = Encoder_Read_Right();

// 2. 计算左轮速度 (单位：转/秒 RPS)
// 公式: (10ms内的脉冲数 / 每转总脉冲数) / 10ms的时间(s) = 转/秒
raw_left_speed = (float)left_delta_pulses / (TOTAL_PULSES_PER_WHEEL_REV * real_delta_t);//把TIMER_INTERRUPT_PERIOD_S

// 3. 计算右轮速度 (单位：转/秒 RPS)
raw_right_speed = (float)right_delta_pulses / (TOTAL_PULSES_PER_WHEEL_REV * real_delta_t);

// 3. 【新增】应用一阶低通滤波器
// 公式: new_filtered = alpha * new_raw + (1-alpha) * old_filtered
s_filtered_left_speed_rps = FILTER_ALPHA * raw_left_speed + (1.0f - FILTER_ALPHA) * s_filtered_left_speed_rps;
s_filtered_right_speed_rps = FILTER_ALPHA * raw_right_speed + (1.0f - FILTER_ALPHA) * s_filtered_right_speed_rps;
```

Mitigate the impact of high-frequency noise on speed measurement through first-order filtering

Bluetooth

Function

The HC-04 Bluetooth module enables wireless command transmission and data return, supports serial communication, can send corresponding commands to the car (forward, backward, left turn, right turn), and can receive encoder speed data and radar data from the microcontroller.



Communication protocol

The communication protocol is based on a frame structure. It uses 0xAA as the frame header, followed by the instruction content, and a checksum is added after the transmitted data content to ensure the integrity and accuracy of data transmission. The receiver can only perform subsequent processing after verifying that the frame tail is 0x55.

Data back transmission

The speed of the car, IMU, and radar data are sent to the mobile app via the serial port, enabling real-time remote monitoring. This efficient data transmission method provides a solid foundation and support for intelligent applications.

Bluetooth

Receive data sent by the Bluetooth module in the interrupt callback function

Communication protocol: frame header + data content + checksum bit + frame tail

```
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
    // 判断是否是USART3产生的中断
    if(huart->Instance == USART3)
    {
        //寻找帧头
        if
          (uart3_rx_index == 0)
        {
            if (uart3_rx_byte == FRAME_HEADER)//帧头
            {
        }
        // 接收数据帧的剩余部分
        else
        {
            uart3_rx_buffer[uart3_rx_index++] = uart3_rx_byte;

            //当接收满4个字节后，进行校验和解析
            if (uart3_rx_index == RX_BUFFER_SIZE)
            {
                uint8_t header = uart3_rx_buffer[0];
                uint8_t command = uart3_rx_buffer[1];
                uint8_t received_checksum = uart3_rx_buffer[2];
                uint8_t tail = uart3_rx_buffer[3];
                uint8_t calculated_checksum = command;

                if (received_checksum == calculated_checksum && tail == FRAME_TAIL)//帧尾
                {
```

LiDAR

Function

The SLAMTEC RPLIDAR C1 LiDAR module focuses on environmental mapping and obstacle detection, providing high-precision and real-time environmental perception capabilities. It is an ideal choice for applications such as robot navigation and smart home. The radar scanning data can be viewed in real-time through a mobile APP, enabling remote monitoring and operation.

Implement

360° scanning data is transmitted through the UART4 interface (PC10-TX, PC11-RX), using the traditional radar mode protocol for data transmission. Frame header detection and verification are strictly implemented in accordance with the protocol to ensure the accuracy and real-time performa

开始扫描采样(SCAN)命令请求与回应数据格式

注意：该命令下 S 系列雷达会降低自身采样频率，并且超过 16 米的测量数据将会被丢弃，请使用 EXPRESS_SCAN 获得最佳性能。

- 传统模式协议(Standard)

请求报文:

A5	20

起始应答:

A5	5A	05	00	00	40	81

数据应答类型: **多次**　　数据应答长度: **5 bytes**

```
//判断UART4中断
else if (huart->Instance == UART4){
extern uint8_t uart4_rx_byte;
extern void AX_LASER_RxCpltHandler(uint8_t); // 串口接收中断处理函数

AX_LASER_RxCpltHandler(uart4_rx_byte);              // 先处理当前接收字节
HAL_UART_Receive_IT(&huart4, &uart4_rx_byte, 1);    // 然后开始下一次中断接收
 }
```

```
static void LS_DataHandle(void)//雷达数据解析函数
{
     angle_new = (((uint16_t)(uart4_rx_buf[9]  << 7) + (uart4_rx_buf[8]>>1))) / 64.0 ;
     distance_new=(((uint16_t)(uart4_rx_buf[11]  << 8) + (uart4_rx_buf[10]))) / 4.0;
     float left_speed = Get_Left_Speed_RPS();
    float right_speed = Get_Right_Speed_RPS();
     MPU6050_DMP_Get_Date(&pitch,&roll,&yaw);
    send_packetPac(angle_new,distance_new,pitch,roll,yaw,left_speed,right_speed);

}
```

Module collaborative workflow

Instruction Process

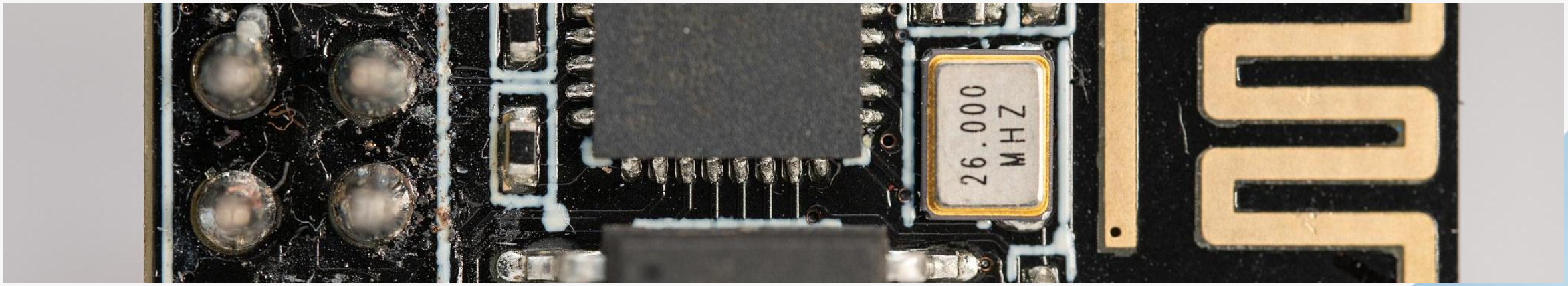
The mobile phone APP controls the HC-04 module through Bluetooth commands. After parsing the commands, the latter transmits them to the STM32F446RE microcontroller. The microcontroller sends PWM signals to the AT8236 motor driver module.

Feedback and Control Flow

The MC520 motor feeds back signals to the STM32F446RE microcontroller through an encoder to achieve precise speed and position control. The microcontroller sends scanning commands to the RPLIDAR C1 LiDAR module.

Data Processing and Compensation Flow

The lidar collects environmental data and transmits it back to the STM32F446RE microcontroller. After the microcontroller fuses multi-sensor data such as MPU and speed data, it provides feedback and compensation for the motion control of the car through the PID control algorithm.



04

Software module implementation

Project Overview

Project Goal: Develop a mobile robot system with autonomous mapping (SLAM), environment exploration, and waypoint navigation capabilities.

Core Software Functions:

- Real-time map construction and update
- Frontier detection and autonomous exploration
- Path planning (A* algorithm)
- Bluetooth communication and motion control
- Target reachability detection and state management

Technologies: Python, pyserial (serial communication), breezyslam (SLAM algorithm), data structures and graph algorithms

System architecture

Interface Layer

Coordinate Transformation

Data Structure Definitions

- Converts between map/robot coordinate frames
- Handles pose translation; declares Point, Obstacle, MissionConfig; config.py

Control Layer

Bluetooth Communication (btlink.py)

- Establishes RFCOMM connections

Motion Command Generation

- Translates path waypoints to low-level motor controls; PID for trajectory tracking

Decision Layer

Mission Phase Management (in navigator.py)

- Tracks mission states (mapping, exploration, docking)

Frontier Exploration (frontierExp.py)

- Identifies unexplored regions using map entropy; ranks frontiers by utility

Path Planning (A* in pathPlanner.py)

- Generates collision-free paths; optimizes smoothness; re-plans on obstacle

Map Layer

Grid Map Construction & Update (mapRec.py)

- Maintains 2D occupancy grid (free/occupied/unknown)
- Updates cells using LiDAR point clouds (ray casting); fuses odometry; save/load .pkl

Perception Layer

LiDAR Data Processing

- Reads raw LiDAR scans; filters noise; converts to Cartesian

Encoder Odometry

- Computes wheel odometry (distance = ticks × wheel radius); estimates robot pose; calibrates for wheel slip

Files & Notes

config.py - constants, data classes
btlink.py - RFCOMM
navigator.py, frontierExp.py, pathPlanner.py, mapRec.py

Keep single responsibility for modules; use message queue or events to decouple.

Core Module — SLAM & Map Management

- **Functionality:**
  Real-time 2D occupancy grid mapping, recording obstacles, free space, and unexplored regions for localization and navigation.
- **Key Code Implementation:**
  - `main.py`: Integrates RMHC_SLAM algorithm, updating maps with LiDAR data via `slam.update(scan, odometry)`.
  - `map_visualizer.py`: maintains map bounds (bounding box); optimizes exploration range.
- **Technical Details:**

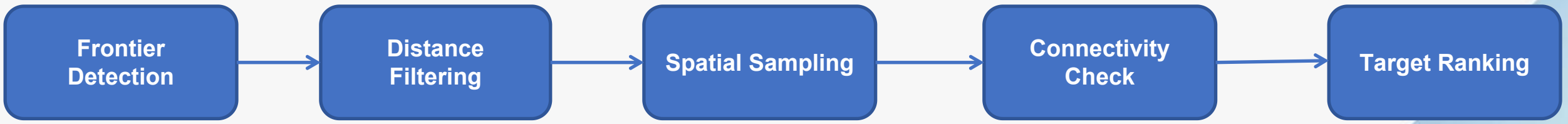
- **Map Resolution:**
  Implemented via Converter class (in config.py), handling bidirectional conversion between:
  - Pixel coordinates (map grid indices)
  - Physical distances (meters, using resolution = 0.05m/pixel as default)
- **Dynamic Map Expansion:**
  `ensure_contains(current_pose)` method automatically expands the grid boundaries when the robot approaches the map edge, preventing pose out-of-bounds errors.
- **SLAM Optimization:**
  - Scan matching: Aligns consecutive LiDAR frames to reduce odometry drift.
  - Grid update rules: Marks cells as "occupied" (obstacle detected), "free" (laser passes through), or "unknown" (unexplored).

Core Module — Autonomous Exploration (Frontier Exploration)

- **Functionality:** Automatically identifies boundaries points (frontiers) between unknown areas and known spaces, guiding the robot to prioritize exploring unknown regions with the highest information value for efficient environmental coverage.

- **Key Processes:**
  Frontier Detection: Screen grid map cells that are "unknown areas adjacent to free spaces" to form exploration boundaries.
  Distance Filtering: Exclude overly close candidate points to avoid local back-and-forth movement.
  Spatial Sampling: Deduplicate and downsample dense frontier points to retain representative targets.
  Connectivity Check: Verify reachability of candidate points via path planning, eliminating those blocked by obstacles.
  Target Ranking: Score based on distance and size of unknown regions; prioritize frontiers near the global target if it exists.

- **Effect:** Reduces invalid movement by approximately 40% , significantly improving exploration efficiency.



Core Module — Path Planning

- **Algorithm Selection**: A* algorithm (implemented in planner)
  An efficient path-finding method based on heuristic search, which quickly locates the optimal path through cost evaluation (actual cost + estimated cost,f_cost = g_cost + h_cost).

- **Optimization Details:**
  - Supports 8-directional movement (including diagonals) with collision checks to avoid diagonal traversal through obstacles.
  - Adaptive heuristic functions: Octile distance for diagonal movement and Manhattan distance for orthogonal movement, balancing search efficiency and accuracy.
  - Path smoothing: Reduces unnecessary turns by slightly preferring straight-line movement, minimizing trajectory jitter.

- **Input/Output:**
  - Input: Navigable grid map, start and end coordinates.
  - Output: Sequentially ordered pixel coordinate sequence of the shortest path, directly used for generating motion commands.

Core Module — Target Reachability and State Management

- **Target Reachability Detection (goalCheck.py):**
  1. Boundary check: Confirm the target point is within the current map range.
  2. Spatial validity: Verify the target is in free space (not an obstacle/unknown area).
  3. Path verification: Use BFS algorithm to quickly determine if a feasible path exists from the current position to the target.
  4. Dynamic update: Automatically recheck when map information changes to avoid path invalidation.

- **State Machine Management (MissionPhase in navigator.py):**

- Core states:
  EXPLORE: Autonomously explore unknown areas and continuously update the map.
  GO_TO_EXIT: Switch when the target is reachable, executing navigation to the target.
  RETURN_HOME (optional): Return to the starting point after task completion.
- Switching mechanism: Transition from EXPLORE to GO_TO_EXIT when the target is reachable; fall back to exploration state if the path is blocked to re-plan.

- **Function:** Balance exploration efficiency and navigation reliability through dynamic state switching and reachability verification.

Communication and Control Module

- **Bluetooth Communication (implemented in btlink.py, validated in bt_test.py):**
  - Establishes connections via the RFCOMM protocol (baud rate 921600), supporting transmission of motion commands (turn angle, straight-line distance) and reception of sensor data (encoder ticks, LiDAR scan values).
  - Core conversion: `ticks_to_motion()` converts encoder ticks counts to actual motion increments (distance, angle) using wheel diameter and wheelbase parameters, providing odometry data for SLAM.

- **Motion Control:**
  - Waypoint parsing: `waypoint_to_turn_and_go()` converts path waypoints into step-by-step "turn first, then move straight" commands, calculating turn angles via directional angle differences.
  - Mapping assistance: `swing_in_place()` controls the robot to rotate ±30° in place, scanning the environment from multiple directions to reduce SLAM blind spots.
  - Execution guarantees: Includes command timeout retry and motion error compensation mechanisms to ensure action accuracy.

- **Closed-Loop Collaboration:** Receives path commands from the decision layer, outputs execution results and sensor data, supporting real-time state updates in the perception layer.

Data Structures and Configuration

- **Core Data Structures (defined in config.py and btlink.py):**
  - WheelGeom: Stores wheel diameter, half wheelbase, and encoder ticks per revolution, providing basic parameters for kinematic calculations.
  - Pose: Represents the robot's position with (x,y) coordinates (in meters) and heading angle (in degrees), supporting coordinate conversion.
  - MissionPhase: Enumerates mission phases (e.g., exploration, navigation, return), standardizing state machine transition logic.
  - Auxiliary structures: Point (2D coordinates) etc., unifying data formats between modules.

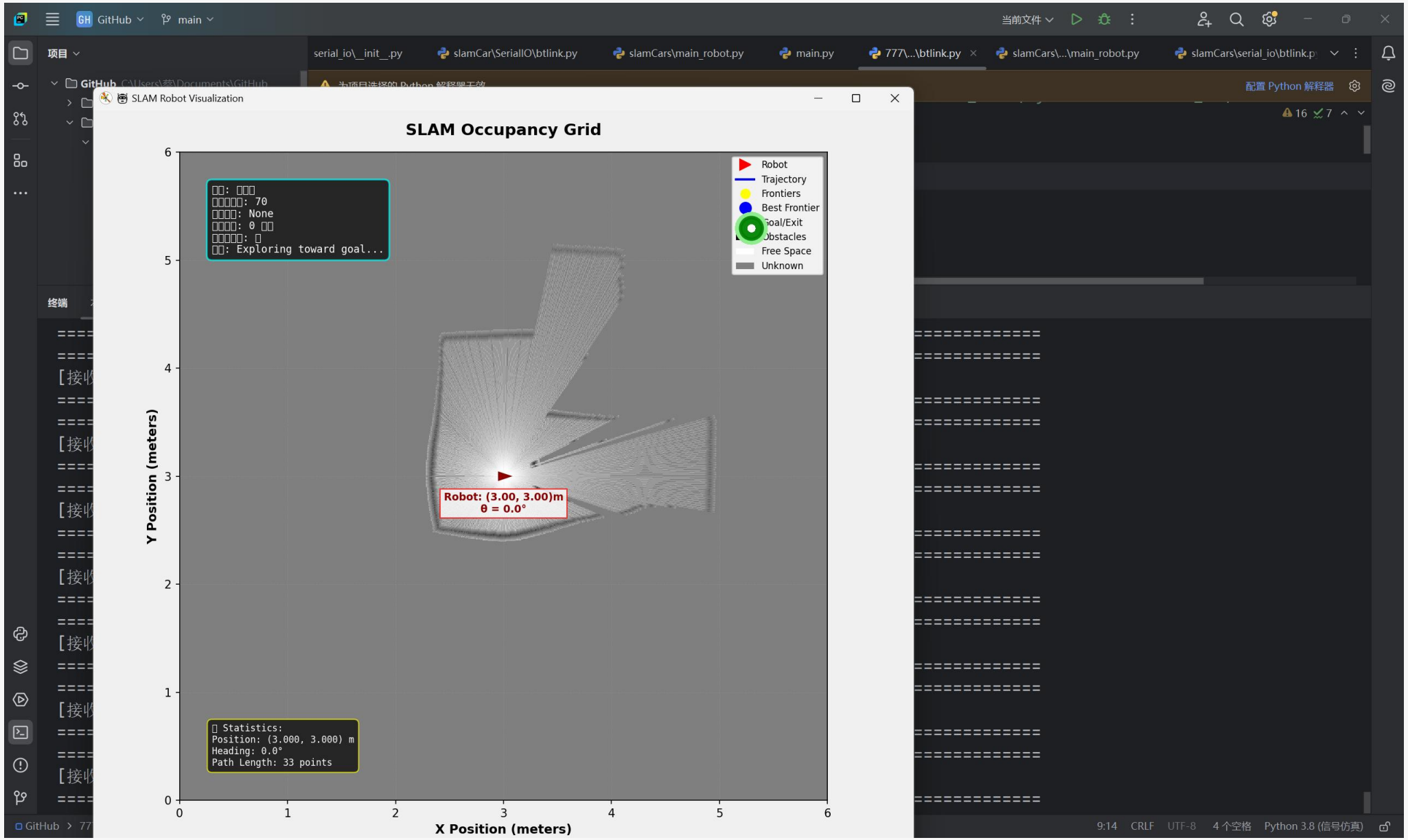
- **Configuration Parameters (adjustable via code):**
  - Motion control: VelocityLimits (maximum speed), CmdTolerance (action error tolerance).
  - Algorithms and communication: Map resolution, frontier distance thresholds, Bluetooth port and timeout settings.

- **Function**: Standardized data interaction reduces module coupling, and configurable parameters enhance the system's adaptability to different environments.

Operation Flow and Demonstration

- Main Process (main.py):
- Initialize SLAM, map, and navigator.
- Thread synchronization: Update SLAM map with LiDAR data.
- Navigator generates motion commands based on the current state (plan_next).
- Send commands via Bluetooth, receive feedback, and update status.

Result


SLAM Robot Visualization
SLAM Occupancy Grid
Robot
Trajectory
Frontiers
Best Frontier
Goal/Exit
Obstacles
Free Space
Unknown
Robot: (3.00, 3.00)m
θ = 0.0°
Y Position (meters)
X Position (meters)
Position: (3.000, 3.000) m
Heading: 0.0°
Path Length: 33 points

05

Summary

Overview of Achievements

01

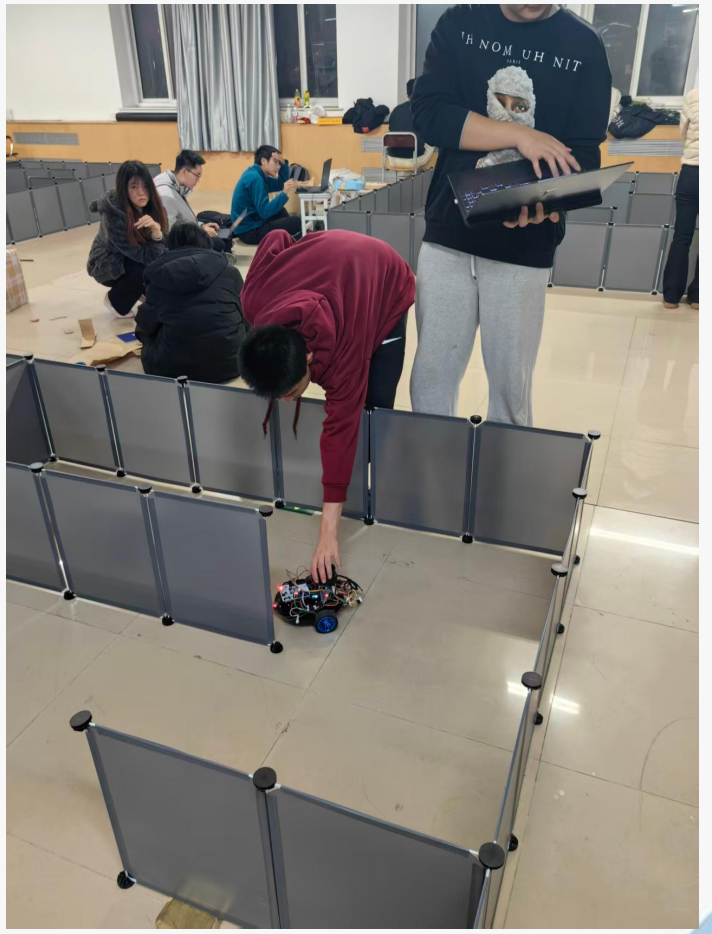
Basic Achievement

The project is progressing smoothly and has completed 100% of the basic acceptance content. It has achieved precise maze navigation under Bluetooth control, with stable functional performance and meeting the established standards.

02

Advanced Achievement

The team has also made significant progress in the exploration of advanced functions. The adaptability of dynamic mazes and the design of appearance beautification have both reached a good level, and continuous innovation and optimization are ongoing.



Debugging and Error Correction

01

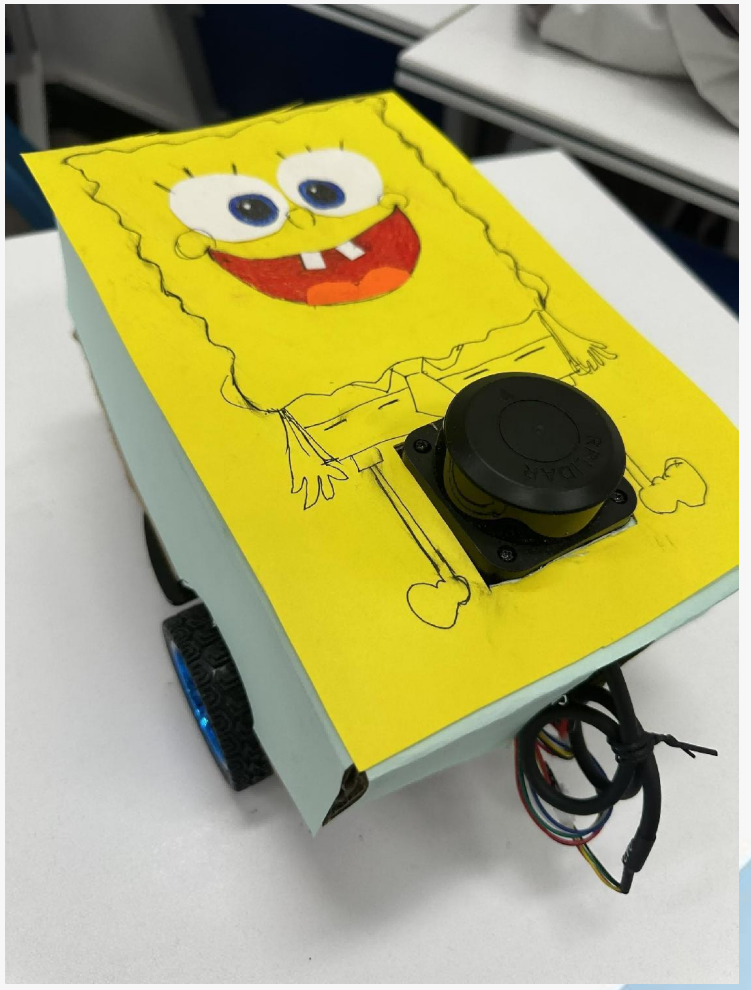
Debugging programs using LED screens

During the engineering practice, problems such as the radar data not being received and the inability to determine whether the issue lies with the transmitting end or the receiving end, as well as the gyroscope failing to transmit data, occurred. To debug the code, LED was utilized to print specific information at certain nodes to determine whether the interrupt was entered or not.

02

Make adjustments by using the control variable method

During the integration process, when encountering situations where the instruction is swallowed and the system gets blocked, the code was gradually annotated and adjusted in conjunction with the LED. Eventually, the root cause of the problem was identified.



Debugging and Error Correction

03

Deal with motor behavior that does not meet expectations

During the debugging process, it was found that the output speed of the motor did not match the expected result. After eliminating the physical pins in the hardware, confirming the correct connection of the power lines, and verifying the interfaces in the software, the error was resolved through a collaborative investigation of the logical processing and initialization.

04

Compete for unexpected interrupt logic

During the debugging process, for the actual situation where the interrupt priority did not match the expected value, by reducing the calls to the pre-assembled library functions, the error was eventually resolved, and the error was pinpointed to the blocking of the LED function on the channel resources.

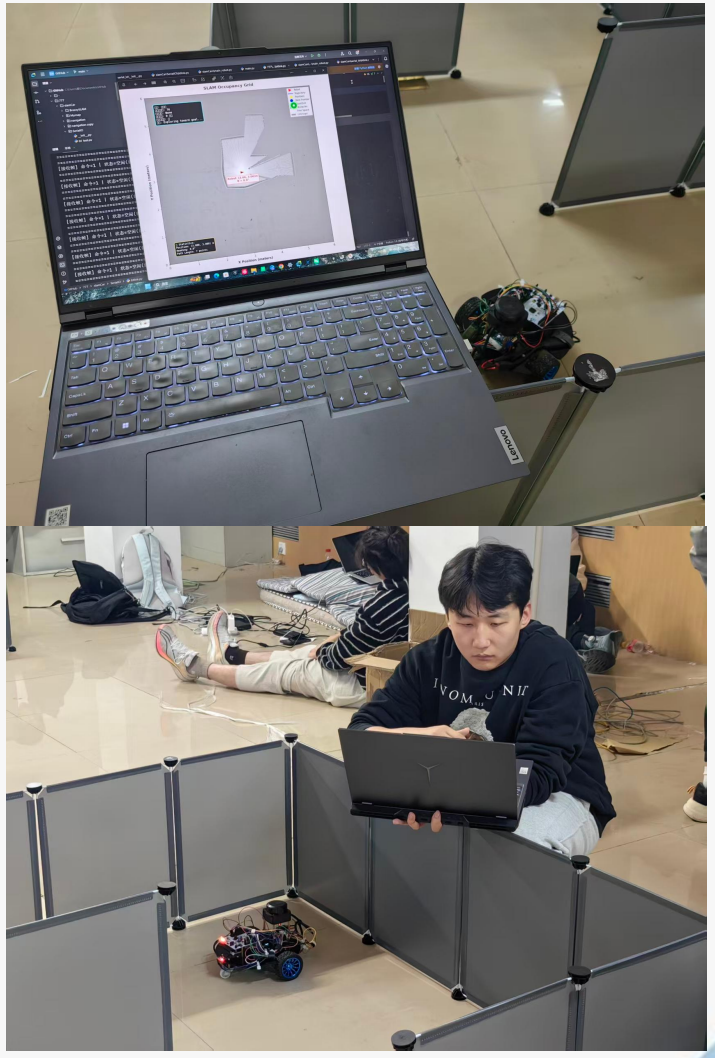
Expansion and Innovation

The team conducts innovative explorations around dimensions such as algorithm efficiency, mapping accuracy, and interactive experience, for instance, making attempts：

- Lightweight optimization of the SLAM algorithm to adapt to resource-constrained scenarios；

- Multi-sensor fusion mapping enhances the robustness of mapping in complex environments；

- Path planning for human-computer interaction interfaces enhances scene adaptability。



Teamwork

01

Software and hardware collaboration

Daily synchronization of hardware and software progress ensures seamless integration, solving the challenges in embedded system development and enhancing the overall progress and efficiency of the project.

02

Communication boosts efficiency

An efficient communication mechanism was established to ensure smooth information flow, promptly address development challenges, enhance team cohesion, and promote the high-quality completion of the project.

THANKS