Autonomous Mobile Robot for Maze Mapping and Navigation with 2D LiDAR

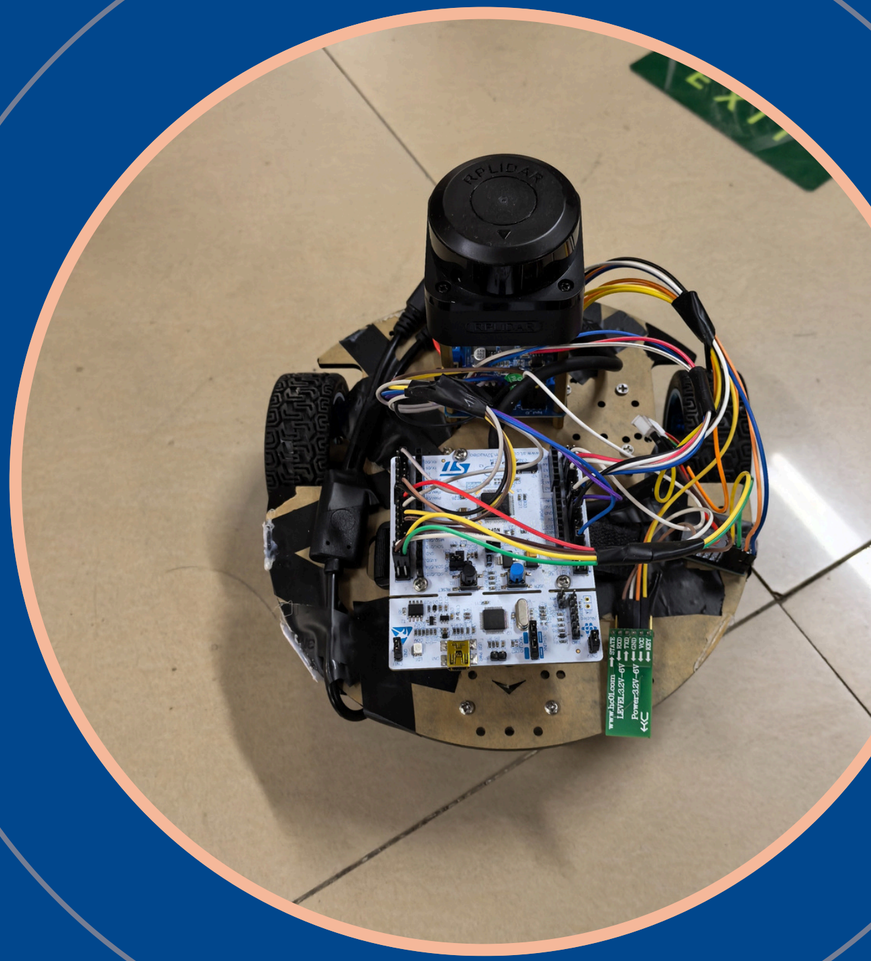
Reporting Team: 57

Reporting Date: 23rd October,2025

CONTENTS

PART 01

Project Background and Objectives



1.1 Project Background

This project, officially titled "Design and Build 2025 Autonomous Mobile Robot for Maze Mapping and Navigation with 2D LiDAR," is led by Jonathan Loo and Qihui Ye. Its core positioning is to develop an autonomous mobile robot equipped with 2D LiDAR (Light Detection and Ranging) sensing capabilities, aiming to solve the technical challenge of "real-time map construction, autonomous navigation, and return to the original path in an unknown complex maze environment." It is a comprehensive engineering practice project integrating hardware integration, firmware development, software algorithms, and system collaboration.

1.1.1 Core Project Objectives and Application Scenario Background

The core tasks of the project revolve around "autonomous maze operation," with its scenario settings and functional objectives having clear technical orientations:

- **Scenario Limitation:** The robot is required to operate in a complex maze environment of approximately 3m × 3m (as shown in Figure 1, which includes two key positions: the starting point and the exit). This scenario simulates the demand for "autonomous exploration in unknown enclosed spaces," serving as a simplified model for practical application scenarios such as warehouse inspection and indoor disaster rescue .

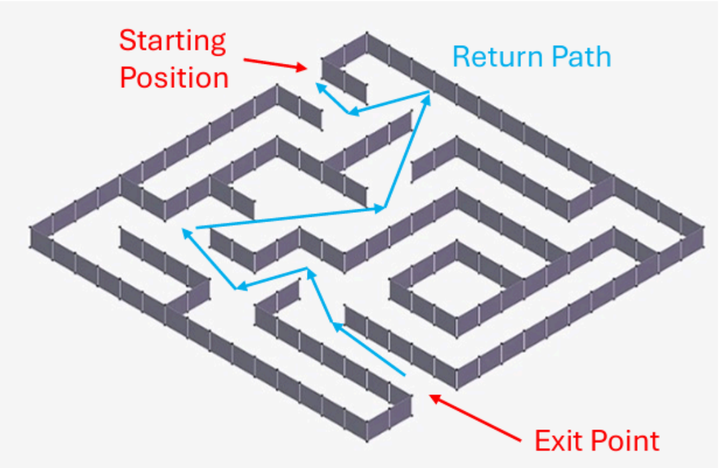


Figure 1. Example Maze with Starting Position and Exit Point

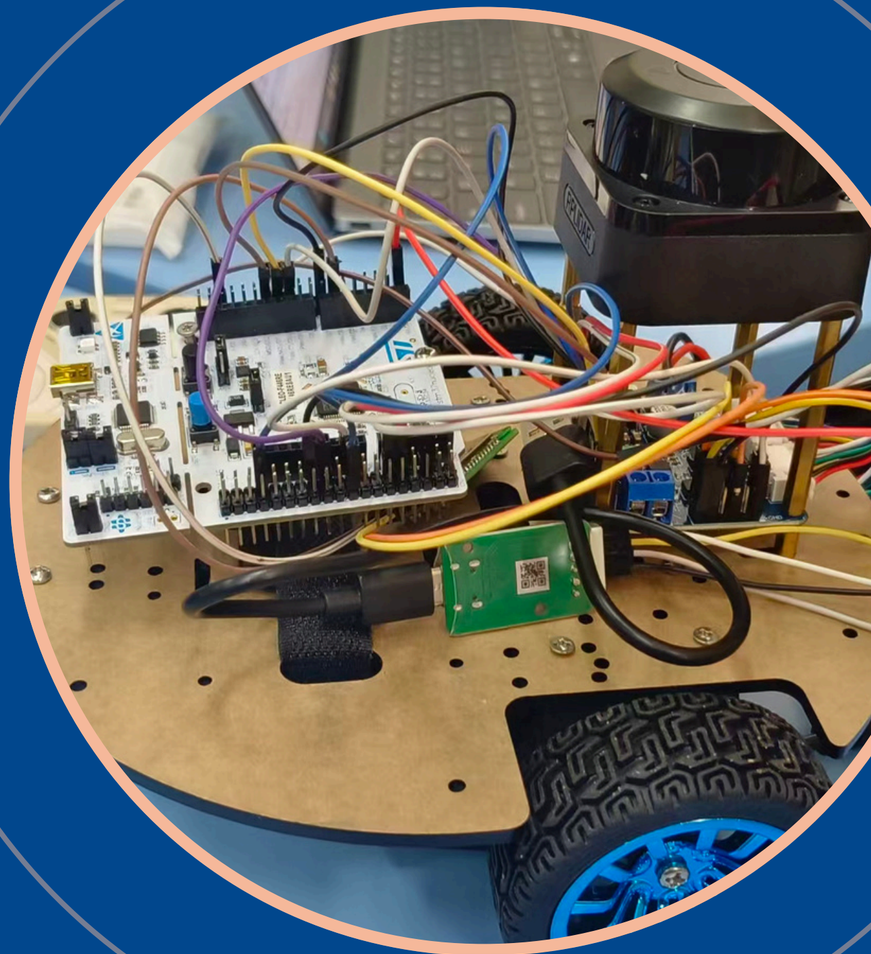
Project Background and Objectives

- **Functional Objectives:** The robot needs to sequentially complete three core tasks—First, the robot explores the maze.; second, accurately identify the exit point in the maze; and finally, return to the initial starting point from the exit point. Throughout the process, it relies on algorithms to make decisions and implement control .

- **Technical Nature:**The essence of this objective is to verify the integrity of the "perception-decision-control" closed loop—acquiring environmental data through 2D LiDAR, constructing maps and achieving localization through SLAM algorithms, generating motion trajectories through path planning algorithms, and implementing motor execution through PID control (control)—ultimately achieving operation .

PART 02

Team Division of Labor and Development Plan



2.1.1Team Division of Labor

CUSI:Chinese University Student ID

BUSI:British University Student ID

Hardware Team

Tianyu Zhao

CUSI:2023213299

BUSI:231221537

Assist in building the car hardware, configuring the development board pins, debugging the Bluetooth module, assisting in debugging the MPU6500 and motors to ensure they can correctly transmit data such as acceleration and left/right wheel speeds and execute motion commands, improving the code logic of the motor and main functions, implementing a customized odometer function, communicating with the software team to confirm the transmission content and format, and conducting final debugging and adjustments on the car.

Zhiyi Luo

CUSI:2023213113

BUSI:231224952

Be mainly responsible for writing the motor and encoder module, providing appropriate PWM waves for the car, enabling it to move forward, backward, turn left and turn right normally under Bluetooth control. By writing the PID algorithm, I ensured the car travels as straight as possible. Additionally, I calculated the mileage via the encoder to accurately feed back the distance the car traveled.Assisted in the repair and maintenance of the car's hardware.

Erfei Zha

CUSI:2023213126

BUSI:231222796

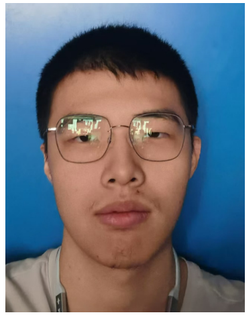
Be responsible for the six-axis data collection and angle measurement of MPU6500, having completed sensor initialization, data reading, and angle calculation algorithms. I realized the conversion of physical values for acceleration and angular velocity, and obtained real-time angles through gyroscope integration. Combined with encoder data, I achieved position estimation, providing a precise attitude perception and navigation foundation for the car.

2.1.1Team Division of Labor

CUSI:Chinese University Student ID

BUSI:British University Student ID

Hardware Team



Xiangyu Dai

CUSI:2023213559

BUSI:231224952

Set up Bluetooth reception interrupt, execute commands and provide feedback; be responsible for the construction and debugging of the radar section, receive radar data using a circular buffer to avoid overflow, and parse it into JSON for transmission.

Yangxinyue Zhou

CUSI:2023213233

BUSI:231222800

As the team leader, be responsible for assisting in building the car hardware, helping improve the code logic of the motor part, assigning tasks to team members, docking work items with the software team and initiating consultation meetings, welding hardware, purchasing and repairing car materials, maintaining and repairing the car body hardware, creating the car's appearance, and making the final presentation PPT.

Yue Wang

CUSI:2023213582

BUSI:231224446

Be responsible for the construction and debugging of the lidar section, including the mechanical installation of the radar module, power supply and signal connection, debugging of the data acquisition program, as well as data docking with the upper computer and control system. I completed the real-time display and accuracy calibration of radar ranging data, providing basic support for the system's map construction and path planning.

CUSI:Chinese University Student ID

BUSI:British University Student ID

2.1.1Team Division of Labor

Software Team

Xiaoxiao Ma

CUSI:2023213672

BUSI:231223715

In the software team, I was mainly responsible for implementing the path planning algorithm and developing the real-time monitoring algorithm for the robot car. I integrated the strengths of the A* algorithm and the DWA algorithm while also taking into account the specific conditions of the given maze. I proposed an innovative approach combining center-point planning with offset correction, which greatly simplified the pathfinding logic and reduced the implementation complexity of the robot car’s control code.

Hengwei Hu

CUSI:2023213609

BUSI:231223081

I am primarily responsible for the SLAM mapping module of the robot.My core task is to enable the robot to localize itself and simultaneously build a map of the unknown maze in real-time. This involves processing sensor data, selecting and implementing suitable algorithms, and ensuring the generated map is accurate and reliable for subsequent modules like path planning.

CUSI:Chinese University Student ID

BUSI:British University Student ID

2.1.1Team Division of Labor

Software Team

Yuruo Liang

CUSI:2023213442

BUSI:231226336

Mainly responsible for code parameter adjustment and optimization after each iteration. Through fine-tuning of key algorithm parameters, the efficient operation of path planning and SLAM algorithms is ensured. Meanwhile, targeted optimization is conducted based on the performance after each code update to improve system response speed and navigation accuracy, providing strong support for the robot's autonomous navigation in complex maze environments.

Yiliang Zhang

CUSI:2023213378

BUSI:231225661

In charge of the software-hardware coordination of the project, leading the data interaction and communication adaptation between firmware and host computer software. Specifically, it includes debugging of Bluetooth module data transmission, standardization of sensor (LiDAR/IMU/encoder) data formats, and ensuring real-time two-way communication between firmware and Python host software. It also unblocks the links for PID motor control command issuance and pose data return, ensuring the hardware implementation of functions like SLAM mapping and path planning, as well as the closed-loop operation of the system.

2.1.2Group contribution degree

Hardware Team

TianYu Zhao	ZhiYi Luo	ErFei Cha	XiangYu Dai	YangXinYue Zhou	Yue Wang
17.6%	17.6%	17.6%	17.6%	17.6%	12%

Software Team

XiaoXiao Ma	HengWei Hu	Yuruo Liang	Yiliang Zhang
25%	25%	25%	25%

2.2Four-Phase Development Plan

Phase 1: Training

- **Hardware Team:** To learn the basic knowledge of STM32, PID algorithm, CubeMX and Keil development software.
- **Software Team:** To learn the basics of SLAM algorithms, the implementation framework of A*/DWA algorithms.

Phase 2: Development

- **Hardware Team:** In the development of the MPU6500 six-axis sensor, we plan to complete the full driver configuration for the gyroscope and accelerometer; realize sensor initialization via the I2C bus, set the gyroscope range to ±250dps, accelerometer range to ±2g, and configure a 5Hz digital low-pass filter and 100Hz sampling rate; develop an angle calculation algorithm to calculate the yaw angle in real time through the integration of gyroscope Z-axis data, and establish a calibration mechanism to eliminate zero-offset error. For encoder speed measurement, use the encoder mode of TIM2 and TIM4 to collect left and right wheel pulses, plan to achieve accurate speed and distance calculation, and ensure real-time performance through a 10ms sampling period. For LiDAR data collection, configure USART6 for DMA communication at a baud rate of 460800, and use idle interrupt to realize data frame parsing to ensure complete collection of scanning data.

2.2Four-Phase Development Plan

- **Software Team:** To Combine and optimize the A* and DWA algorithms. Based on odometer data, the car's offset from the center point was determined, and a threshold was set. If the offset exceeds the threshold, the direction is automatically corrected to ensure the route is accurate and the car can reach the center point.

Phase 3: Integration

- **Hardware Team:** In the system integration phase, we will focus on solving the problem of multi-peripheral collaborative work. Through the command parsing of the Bluetooth module, the car's motion control functions will be realized, including forward, backward, steering and stopping. In terms of data fusion, the odometer data measured by the encoder will be combined with the angle information of the MPU6500, and the dead reckoning algorithm will be used to calculate the car's position coordinates (x_d, y_d) in real time. During the debugging process, the timing coordination of each module will be optimized to ensure that the 10ms cycle encoder data update and IMU data collection are synchronized. At the same time, a complete data transmission protocol will be established to upload position and attitude information via USART3 at a baud rate of 921600, supporting real-time monitoring and debugging output, and finally realizing a complete closed loop of sensor data collection, motion control and status feedback.

2.2Four-Phase Development Plan

- **Software Team:** The software team plans to first carry out coordinate construction for the 4x4 area, then perform scanning and mapping based on the data returned by the radar, and finally make a comprehensive judgment using the Euclidean distance between the coordinates of the nearest center point to the current position and the coordinates of the nearest center point to the exit to obtain the optimal moving point. After such iterations, the robot will move forward and eventually find the exit. This is an optimized version of DFS.

Phase 4: Acceptance

The software team and hardware team will work together to complete the production of the car's appearance, creation of the acceptance video, development of the PPT report, and packaging of the complete code package.

PART 03

Core Technology and System Design

Analysis of hardware and software technical details

Core Technology and System Design

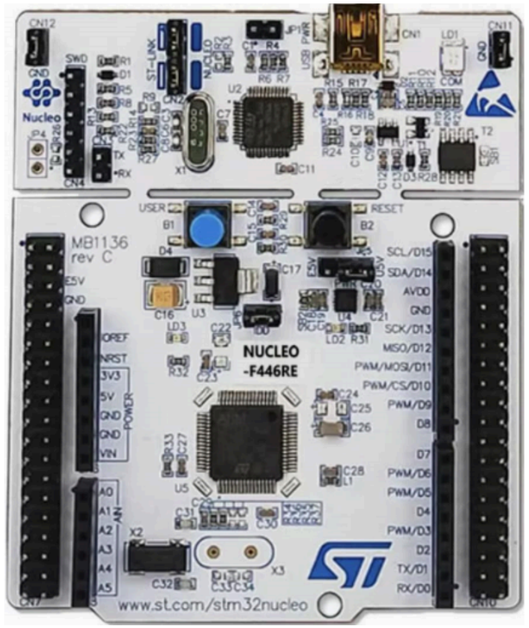
3.1 Hardware system design

Core hardware list

STM32F446RE，AT8236 Motor Driver，520 Geared Encoder DC Motor，SLAMTEC RPLIDAR C1，MPU6500 IMU，HC-04 Bluetooth Module

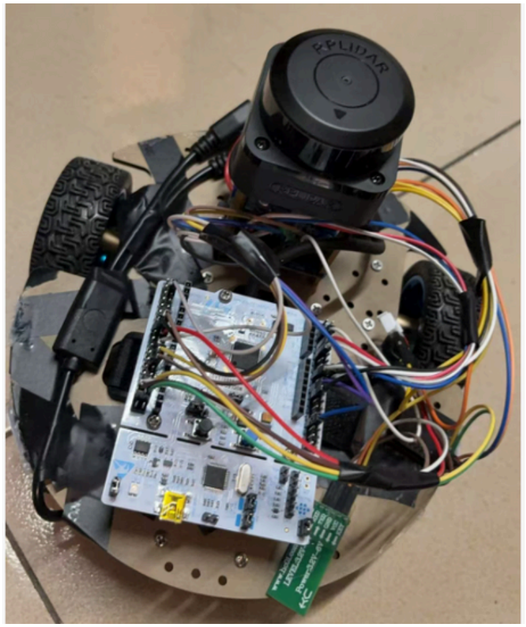
Hardware connection architecture

Taking the development board as the core, after reasonable pin configuration, each hardware is connected to the pins on the development board through DuPont wires, and each module transmits information to the upper computer via Bluetooth HC-04

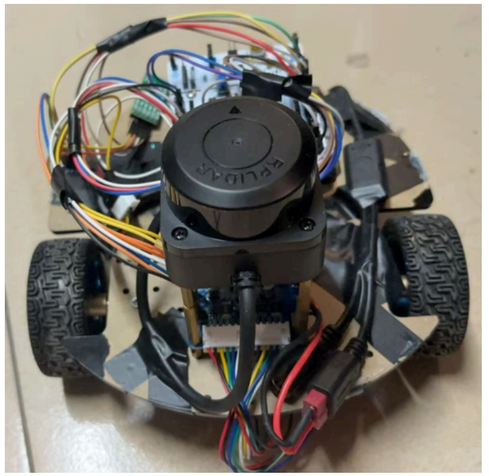


NUCLEO development board

Power drive module and radar



Hardware wiring diagram



Hardware wiring diagram

3.2 Software system design

3.2.1Layered Architecture

- Firmware layer (hardware control and data acquisition)
- Data processing layer (sensor data optimization)
- Algorithm layer (SLAM/path planning/exploration),
- UI layer (user interaction and visualization)

3.2.2Core module design

- Simulation environment module: Built on Python, it simulates the maze environment to generate 2D LiDAR simulation data and odometer simulation data
- SLAM mapping module: Output raster maps and ensure real-time performance through map updates
- Identifying unknown areas: By traversing the grid map to identify the boundaries of "unknown grids - idle grids", calculate the distance and size of the center points of each Frontier, select the optimal exploration target, and generate exploration sub-paths

3.2 Software system design

・Path planning module: The feasible advance position is determined by using the data returned from radar detection, that is, the center points of each block. The next local planning point is selected based on the Euclidean distance from the center point to the end point, and finally the path planning is achieved

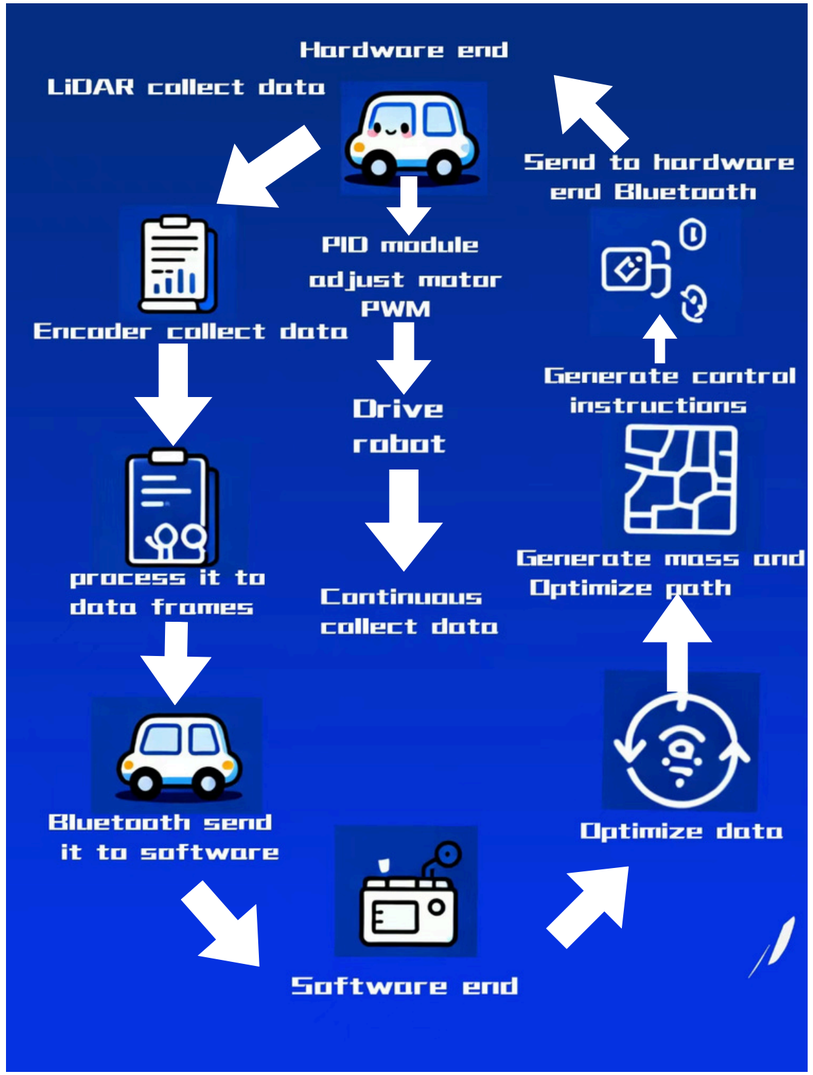
・User interface module: Developed based on Python, it includes: map visualization area (displaying the raster map generated by SLAM, the real-time position of the robot, and the exploration trajectory), navigation information area (displaying the coordinates of the target point, remaining distance, and current speed), control area (start/pause/back buttons, parameter adjustment sliders), radar information and odometer are returned at the terminal

3.2 Software system design

Hardware end: LiDAR collects environmental data → Encoder collects wheel speed data → Firmware layer processes it into standardized data frames → Bluetooth module sends it to the software end

Software end: Receive hardware data → Optimize LiDAR/ odometry data →SLAM module generates raster maps → Identify unknown areas →A* plans the global path →DWA optimizes the local path → Generate control instructions → Send them to the hardware end via Bluetooth

Hardware end: Receive control instructions →PID module adjusts motor PWM→ Drive robot movement → Continuously collect data

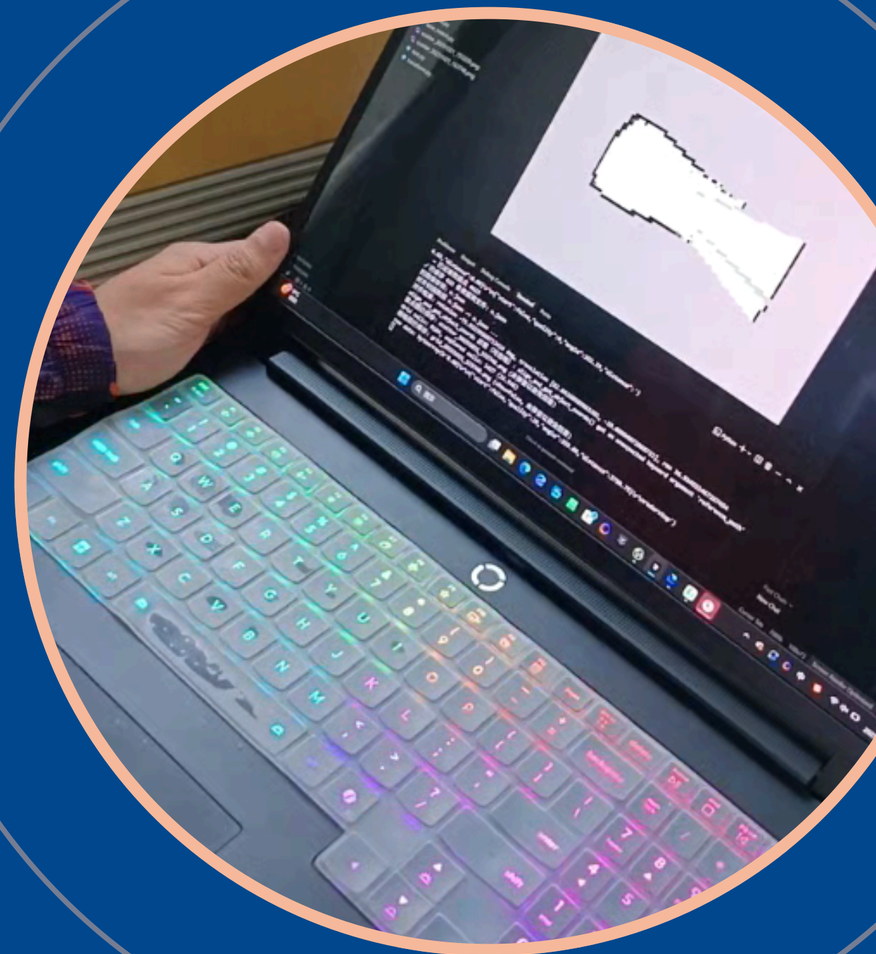


Data Flow Diagram

PART 04

Development process and key achievements

Present the development results



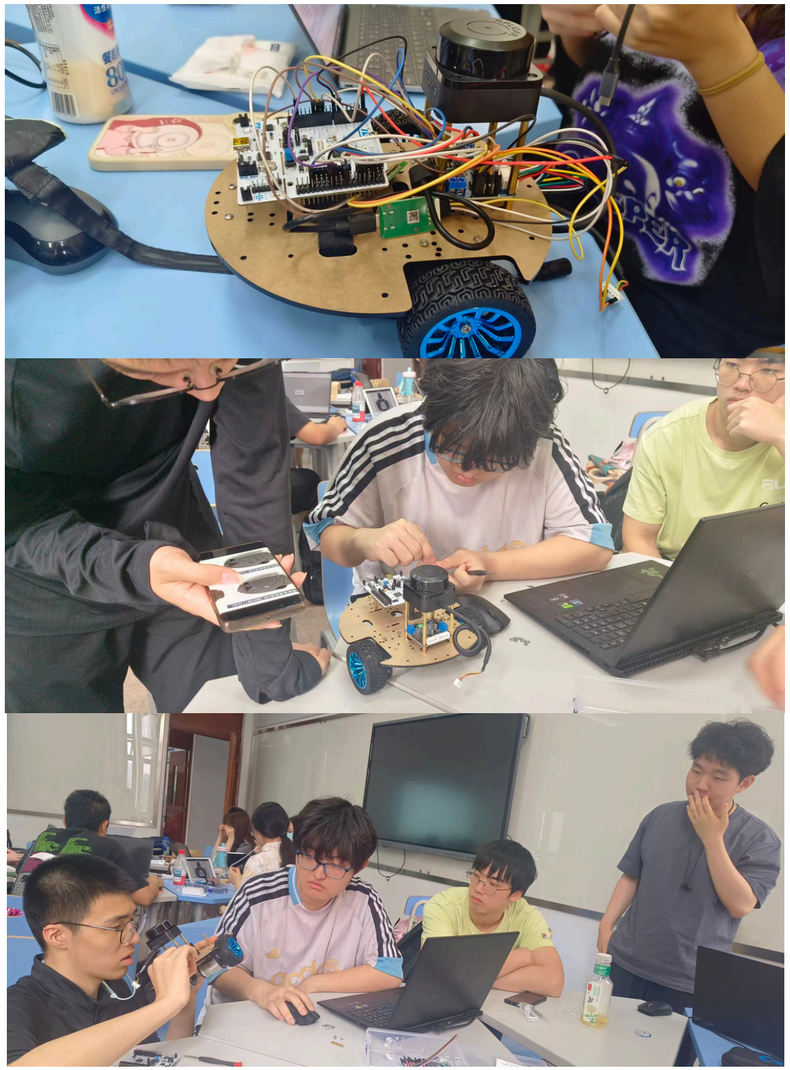
Development process and key achievements

4.1 Progressive achievement presentation

Phase 1

· The hardware team: Completed the basic architecture of the car, the correct transmission of radar gyroscope encoder information, and the command to move the car via Bluetooth

· The software team : Completed the simulation of the SLAM algorithm , the demo of the Frontier Detection and Selection algorithm . Determine the Bluetooth data transmission format



Phase 2

· The hardware team: Can transmit more complete information and more complex commands, complete the specially designed odometer, correct the radar transmission logic, and write and adjust the PID algorithm to ensure the stable movement of the car

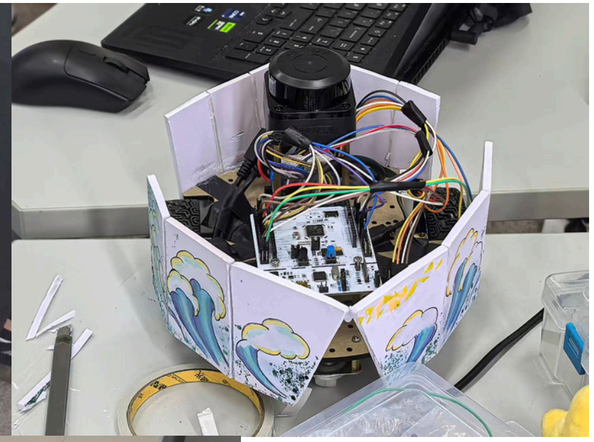
· The software team ：Combining and optimizing the A* and DWA algorithms, based on the odometer data, the offset of the car from the center point is known. A threshold is set. Ensure the correct route and reach the center point.

4.1 Progressive achievement presentation

Phase 3

· The hardware team:
The zero drift problem of the gyroscope is solved by filtering

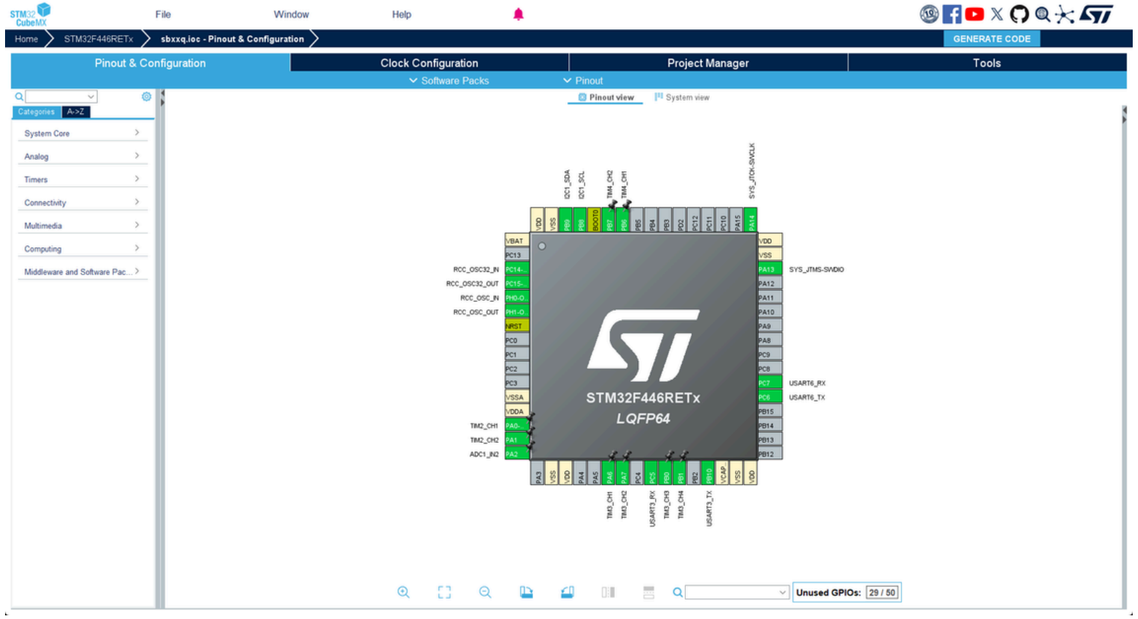
· The software team：
Firstly constructing the coordinates of a 4*4 area, then scanning and mapping based on the data sent back by the radar, finally making a comprehensive judgment based on the Euclidean distance of the coordinates of the center point closest to the current point and the center point closest to the exit to determine the optimal travel point.，they moved forward and finally found the exit to optimize the dfs



4.1 Progressive achievement presentation

Hardware functions

Motor forward and reverse rotation

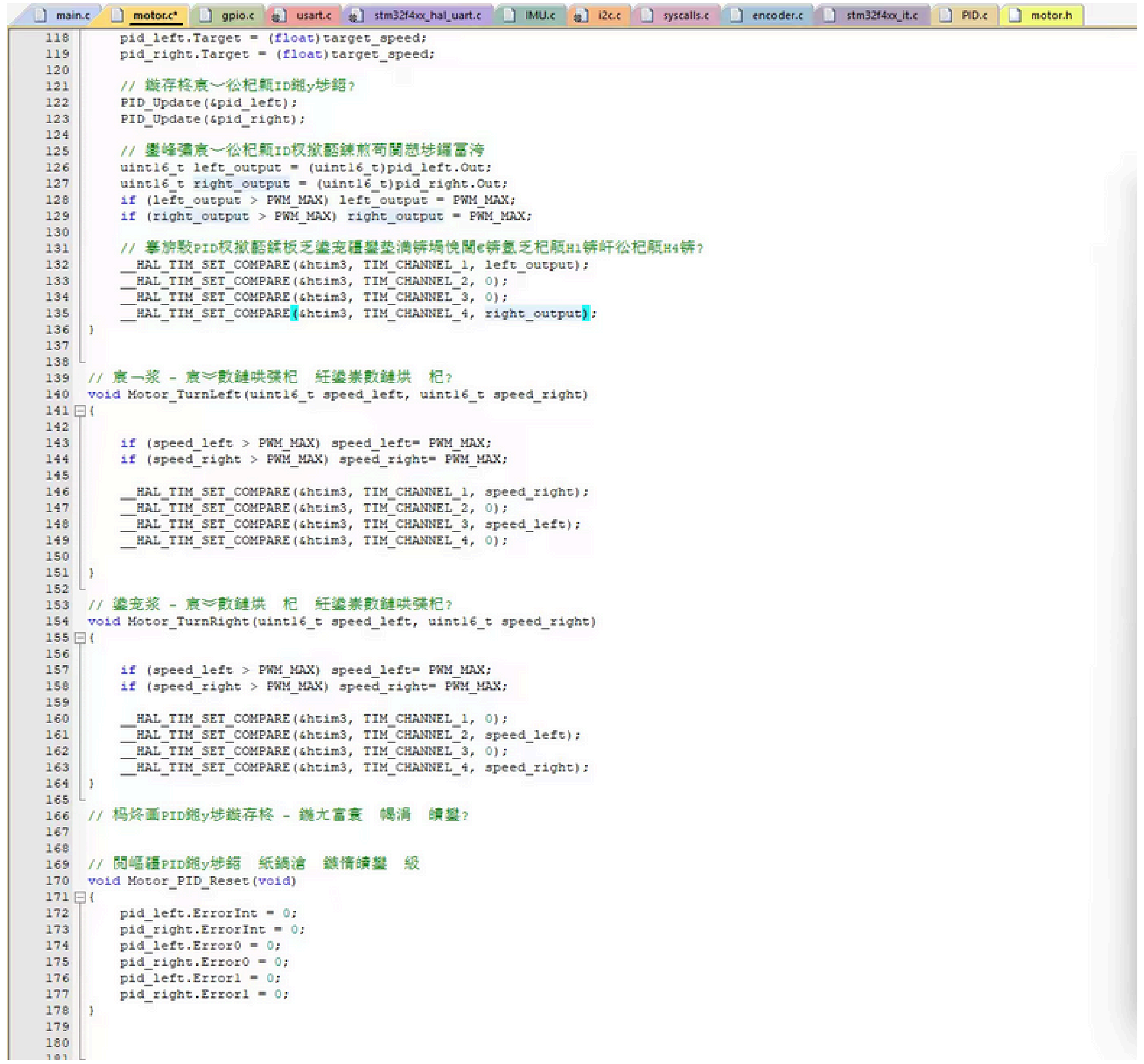


Pin configuration diagram of the development board

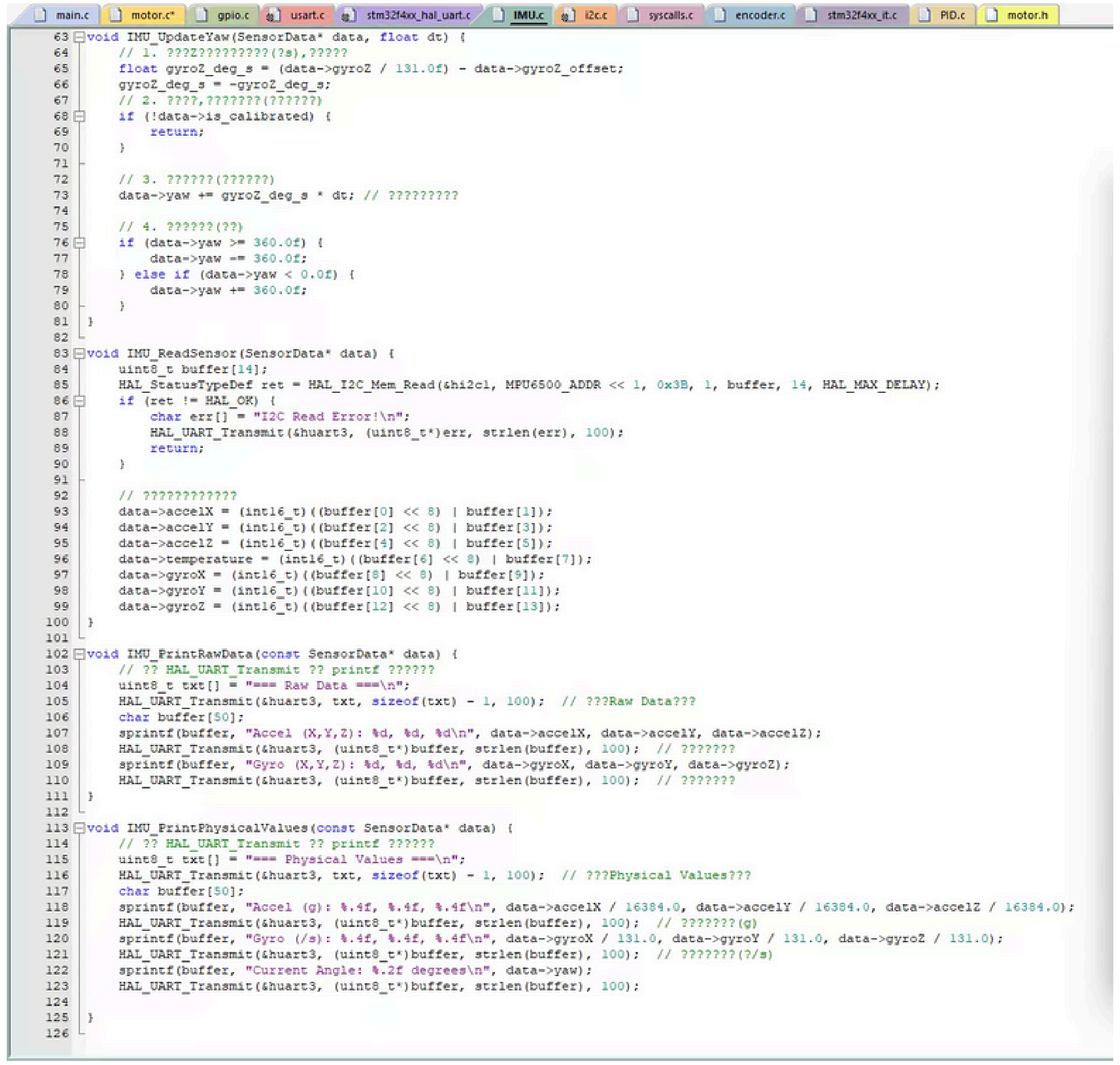
Development process and key achievements

4.2 Core Function Demonstration

Code implementation



Motor Code



MPU Code

4.2 Core Function Demonstration

Code implementation

```
uint8_t S  = data[0] & 0x01;
if ((data[1] & 0x01) != 1) return;
if (S==((data[0] >> 1) & 0x01)) return;

infolen = sprintf(rdinfobuf,"{\"start\":%s,\"quality\":%d,\"angle\":%.2f,\"distance\":%.2f}\r\n",
(S == 1) ? "true" : "false",
(data[0] >> 2),
(((uint16_t)data[2]) << 7) | (data[1] >> 1))/64.0f),
((((uint16_t)data[4] << 8) | data[3])/4.0f));

if (infolen > 0) {
  HAL_UART_Transmit_IT(&huart3, (uint8_t*)rdinfobuf, infolen);
}
}

void ParseData(void)
{
uint16_t i = 0;

while (i < parse_len)
{
  if (parse_len - i >= 7 && parse_buf[i] == 0xA5 && parse_buf[i+1] == 0x5A)
  {
    switch(parse_buf[i+2])
    {
      case 0x05:
        cur_data_len = parse_buf[i+2];
        sync_mode = 1;
        break;
      case 0x03:
        cur_data_len = parse_buf[i+2];
        sync_mode = 1;
      break;
      default:
        HAL_UART_Transmit_IT(&huart3,txtundentifiedmsg,sizeof(txtundentifiedmsg)-1);

    }
    //HAL_UART_Transmit(&huart3, &parse_buf[i], 7, 1000);
    i += 7;
  }
  else if (sync_mode && (parse_len - i) >= cur_data_len)
  {
    if (cur_data_len == 3)
    {
      if (parse_buf[i]==0x00)
      {
        HAL_UART_Transmit(&huart3, txtnice, sizeof(txtnice)-1, 1000);
      }
      else if (parse_buf[i]==0x01)
      {
        HAL_UART_Transmit(&huart3, txtwarn, sizeof(txtwarn)-1, 1000);
      }
      else if (parse_buf[i]==0x02)
      {
        HAL_UART_Transmit(&huart3, txterror, sizeof(txterror)-1, 1000);
        parse_buf[i+1]+=0x30;
        HAL_UART_Transmit(&huart3, parse_buf+i+1, 1, 1000);
      }
      else
      {
        HAL_UART_Transmit(&huart3, flag, sizeof(flag), 1000);
      }
      sync_mode = 0;
      cur_data_len = 0;
```

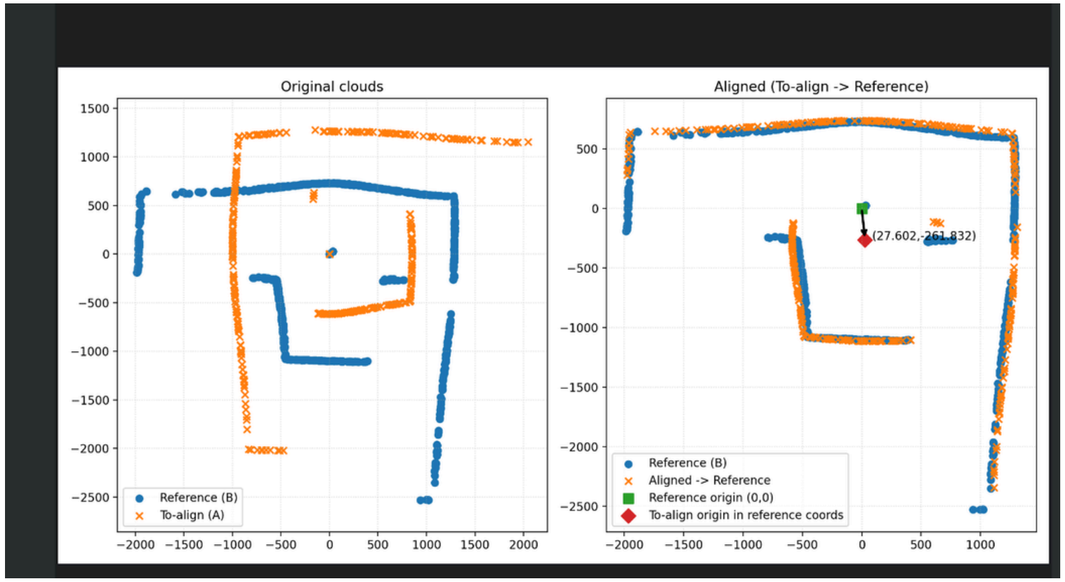
Radar Code

```
//if (!bluetooth_connected) bluetooth_connected = 1;
//HAL_UART_Transmit(&huart3, rx3buf,2,1000);
switch (rx3buf[0])

{
  case 0x00:
    switch(rx3buf[1])
    {
      case 0x00:
        HAL_UART_Transmit(&huart3,txtstop,sizeof(txtstop)-1,100);
        Motor_Stop();
        break;
      case 0x01:
        HAL_UART_Transmit(&huart3,txtforward,sizeof(txtforward)-1,100);
        Motor_Forward(200);
        break;
      case 0x02:
        HAL_UART_Transmit(&huart3,txtbackward,sizeof(txtbackward)-1,100);
        Motor_Backward(200);
        break;
      case 0x03:
        HAL_UART_Transmit(&huart3,txtleft,sizeof(txtleft)-1,100);
        Motor_TurnLeft(400,400);
        break;
      case 0x04:
        HAL_UART_Transmit(&huart3,txtright,sizeof(txtright)-1,100);
        Motor_TurnRight(400,400);
        break;
       case 0x05:
      // ???????
      Encoder_Data_Reset(&encoder_data);
      // ??????????
      x_d = 0.0f;
      y_d = 0.0f;
      prev_left_distance = 0.0f;
      prev_right_distance = 0.0f;
      HAL_UART_Transmit(&huart3, (uint8_t*)"Encoder Reset\r\n", 15, 1000);


      break;
        case 0x06:
          speedon=1-speedon;
      break;
      case 0x07:
          mpuon=1-mpuon;
      break;
      case 0x08: // ??:??????(????0x08 0x00???)
        if (rx3buf[1] == 0x00) {
            IMU_InitYawReference(&imuData); // ??????
            HAL_UART_Transmit(&huart3, (uint8_t*)"Angle reset to 0\n", 17, 100);
          }
      break;
      case 0x09: {
                  char msg[100];
                  float current_angle = imuData.yaw;
                  sprintf(msg, "Current Angle: %.2f degrees\n", current_angle);
                  HAL_UART_Transmit(&huart3, (uint8_t*)msg, strlen(msg), 100);
                  float diff;
                  if (is_first_call) {
                      diff = current_angle - 0.0f;  // ???:???? - 0

                      is_first_call = 0;            // ???????

                  } else {
                      diff = current_angle - last_angle;  // ????:?? - ???
```

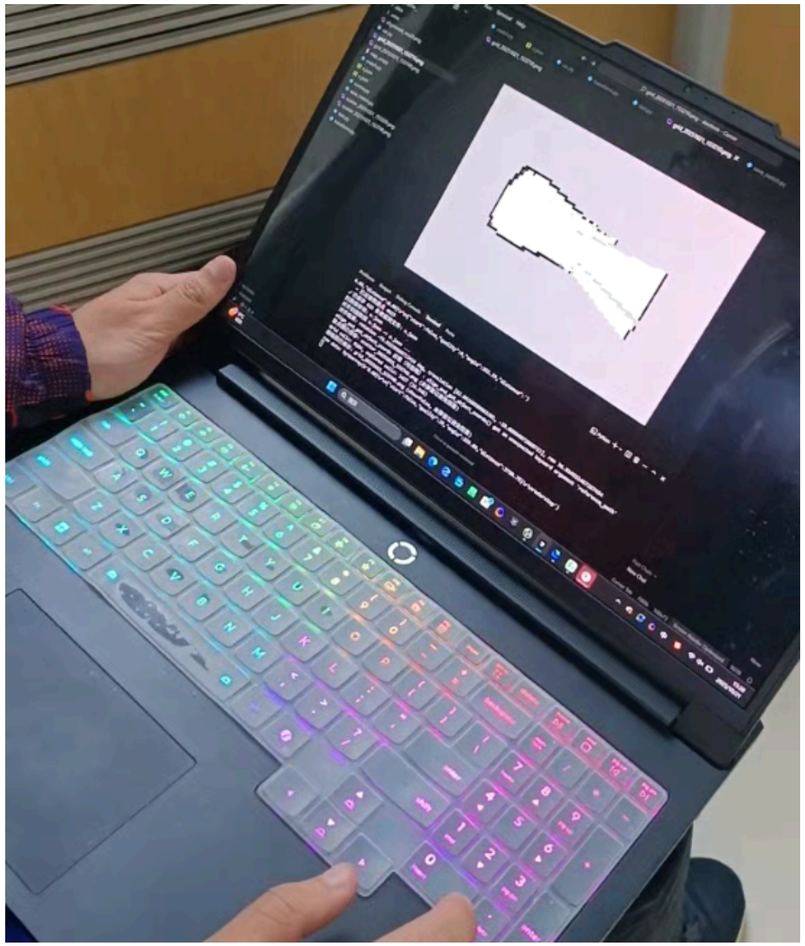
Main Code

4.2 Core Function Demonstration

Software functions



Dispersed points for slam mapping matching



The exploration process of the car

4.2 Core Function Demonstration

Software Codes

match.py

C: > Users > 马骁骁 > Desktop > Software > match.py > ...

```python
        return np.array(idxs, dtype=int), np.array(dists, dtype=float)


def best_fit_transform(A, B):
    assert A.shape == B.shape
    if A.shape[0] == 0:
        return np.eye(2), np.zeros(2)
    centroid_A = np.mean(A, axis=0)
    centroid_B = np.mean(B, axis=0)
    AA = A - centroid_A
    BB = B - centroid_B
    H = AA.T @ BB
    U, S, Vt = np.linalg.svd(H)
    R = Vt.T @ U.T
    if np.linalg.det(R) < 0:
        Vt[1, :] *= -1
        R = Vt.T @ U.T
    t = centroid_B - R @ centroid_A
    return R, t


def icp(A, B, max_iterations=1000, tolerance=1e-8):
    src = A.copy()
    prev_error = float('inf')
    errors = []
    total_R = np.eye(2)
    total_t = np.zeros(2)
    for i in range(max_iterations):
        idxs, dists = nearest_neighbors(src, B)
        B_corr = B[idxs]
        R, t = best_fit_transform(src, B_corr)
        src = (R @ src.T).T + t
        total_R = R @ total_R
        total_t = R @ total_t + t
        mean_error = np.mean(dists) if len(dists) > 0 else 0.0
        errors.append(mean_error)
        if abs(prev_error - mean_error) < tolerance:
            break
        prev_error = mean_error
    # 返回累计变换与最终对齐点以及迭代误差列表
    return total_R, total_t, src, errors


def rotation_matrix_to_deg(R):
    return math.degrees(math.atan2(R[1, 0], R[0, 0]))
```

match.py

C: > Users > 马骁骁 > Desktop > Software > match.py > ...

```python
def do_align(from_path, to_path, angle_unit='degree', max_iter=DEFAULT_MAX_ITER):
    # 如果 PCA 失败，退回到不做粗对齐的 ICP
    init_candidates = []
    if angle_A is None or angle_B is None:
        init_candidates.append((np.eye(2), np.zeros(2)))  # 无初始变换
    else:
        delta = angle_B - angle_A
        for add in [0.0, math.pi]:  # delta, delta+pi 两种情况
            theta = delta + add
            R_init = np.array([[math.cos(theta), -math.sin(theta)],
                               [math.sin(theta),  math.cos(theta)]], dtype=float)
            t_init = centroid_B - (R_init @ centroid_A)
            init_candidates.append((R_init, t_init))

    best = None
    # 遍历每个初始化，跑 ICP（从 A_init 开始），合并变换并计算 RMS，选择最优的
    for R_init, t_init in init_candidates:
        # apply initial transform to A to get A_init
        A_init = (R_init @ A.T).T + t_init
        # run ICP starting from A_init (icp will treat A_init as "source")
        try:
            R_icp, t_icp, A_aligned, errors = icp(A_init, B, max_iterations=max_iter)
        except Exception:
            continue

        # 合并变换：整体变换 R_final, t_final, 满足 R_final * A + t_final = A_aligned
        # 因为 A_init = R_init * A + t_init
        # ICP 返回 A_aligned = R_icp * A_init + t_icp = R_icp*(R_init*A + t_init) + t_icp
        # 所以 R_final = R_icp @ R_init ; t_final = R_icp @ t_init + t_icp
        R_final = R_icp @ R_init
        t_final = (R_icp @ t_init) + t_icp

        # 计算 A_aligned 到参考 B 的最近邻距离用于 RMS
        _, final_dists = nearest_neighbors(A_aligned, B)
        rms_error = float(np.sqrt(np.mean(np.array(final_dists) ** 2))) if len(final_dists) > 0 else float('inf')

        # 记录候选解
        cand = {
            'R_final': R_final,
            't_final': t_final,
            'A_aligned': A_aligned,
            'errors': errors,
            'rms': rms_error
        }
        if best is None or cand['rms'] < best['rms']:
            best = cand
```

Rotation Matching

4.2 Core Function Demonstration

Software Codes

car.py

```python
class RobotController:
            print(f"  从 数据解包组成: {e} ")

    # ----------------------
    # 发送指令封装（外部可直接调用这些方法）
    # ----------------------
    async def _write(self, data: bytearray):
        if not self.client or not self.client.is_connected:
            raise RuntimeError("Client not connected")
        await self.client.write_gatt_char(self.write_char, data)

    async def forward(self):
        await self._write(self.CMD_FORWARD)

    async def backward(self):
        await self._write(self.CMD_BACKWARD)

    async def left(self):
        await self._write(self.CMD_LEFT)

    async def right(self):
        await self._write(self.CMD_RIGHT)

    async def stop(self):
        await self._write(self.CMD_STOP)

    async def is_controll_1(self):
        await self._write(self.CMD_IS_STOP_CONTROLL_1)

    async def is_controll_0(self):
        await self._write(self.CMD_IS_STOP_CONTROLL_0)

    async def distance_controll(self):
        await self._write(self.CMD_DIS_CONTROLL)

    async def distance_ini_print(self):
        await self._write(self.CMD_INI_AND_PRINT_DIS)

    async def start_scan(self):
        """发送开始扫描指令（并可同时开启记录）"""
        await self._write(self.CMD_SCAN)

    async def stop_scan(self):
        await self._write(self.CMD_SCAN_STOP)

    # ----------------------
    # 数据记录管理
```

Car Control

transform.py

```python
# ------------------ 栅格化函数：射线更新（ray-cast）+ 点命中 ------------------

def gridify_json_to_grid(json_path,
                         res=50,
                         margin=1000,
                         angle_unit='degree',
                         use_logodds=True,
                         l0=0.0,
                         l_hit=0.9,
                         l_miss=-0.4,
                         l_min=-4.0,
                         l_max=4.0,
                         do_raycast=True,
                         save_npy=True,
                         out_npy_path=None,
                         return_observed_mask=True):
    """
    将 json_path 中的极坐标点栅格化；默认使用射线更新来区分 observed/unobserved。

    返回 (grid, meta, observed_mask) if return_observed_mask else (grid, meta).

    参数要点：
      - do_raycast: 如果 True，会对每个激光束从 sensor -> hit 进行 Bresenham 射线遍历，
                    对射线经过的格子应用 l_miss（表示 free observation），对命中格子应用 l_hit。
      - l_miss 应为负值，l_hit 为正值。

    注意：sensor 原点假定为 (0,0)（即极坐标转换的中心）。如果你的扫描中心不在 (0,0)，
    请传入适当的坐标变换并在调用前把点变换到期望参考系。
    """
    if not os.path.isfile(json_path):
        raise FileNotFoundError(json_path)

    polar = load_polar_json(json_path)
    pts = polar_to_cart(polar, angle_unit=angle_unit)

    if pts.shape[0] == 0:
        raise ValueError('No points to gridify')

    xs = pts[:, 0]
    ys = pts[:, 1]

    xmin = float(xs.min()) - margin
    xmax = float(xs.max()) + margin
    ymin = float(ys.min()) - margin
    ymax = float(ys.max()) + margin
```

Rasterization

PART 05

Summary

Summary

5.1 Project Summary

Outcome:
Successfully achieved the simultaneous realization of maze exploration, exit positioning, and return along the original path for autonomous mobile robots. Also, achieved environment mapping and path visualization. Implemented SLAM mapping, A* / DWA path planning, and Bluetooth command interaction functions.

The hardware team :
Has mastered valuable experience in stm32 and embedded development, gained a deeper understanding of hardware engineering, learned about the "back pressure awareness" of communication links and speed limit design, as well as the fundamentals of motion control and speed measurement closed-loop, and improved their coding capabilities

The software team :
Masters SLAM algorithms, Frontier Detection and Selection unknown area exploration, and path planning algorithms, enhancing the ability of user interface design

The team :
All members deepens multi-disciplinary technical collaboration and problem-solving capabilities

Summary

5.2 Shortcomings and Improvements

Hardware Team’s Current Deficiencies and Future Prospects

- Existing deficiencies:
no movement of the car are not being completely offset, in a long journey is still a small amount of deviation
- Future- function extension:
Calculate electronic fence to ensure the safety of the car - technology deepening: better adjust the speed closed loop, can make the car more straight forward and backward. Better encapsulation of code

Software Team’s Current Deficiencies and Future Prospects

- Existing deficiencies:
The mapping logic of BreezySLAM inherently leads to slow map construction and even occasional freezes.
- Future -function extention:
the mapping process can be optimized for greater speed and accuracy. Both the DWA and A algorithms can be further improved by deeply integrating and optimizing them with DFS, adopting a balanced strategy to make the robot’s decision-making more precise and efficient.

THANKS FOR WATCHING

Reporting Team: 57

Reporting Date: 23rd October,2025