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

2025/10/21

CONTENTS

»»» 01 «««

Project Introduction

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

Group Number: 12

Team Members: Jin Ye, Luo Zhengfei, Fan Zhijian, Xu Zihan, Gao Yinan, Zhang Zeyue, Li Shen, Liu Xiaoqin, Leng Yiheng, Zhou Dunyi, Wang Kerui

Institution: Beijing University of Posts and Telecommunications

Date: 2025/10/21

Team Members & Responsibilities

Hardware Team (6 Members)

- **JinYe**: STM32 MCU Programming & System Integration
- **Luo Zhengfei**: Motor Control & PID Tuning (Position, Speed, Angle)
- **Fan Zhijian**: Sensor Integration (LiDAR, IMU, Encoders)
- **Li Shen**: Power Management & Circuit Assembly
- **Liu Xiaoqin**: Bluetooth Communication Setup & Debugging
- **Zhou Dunyi**: Mechanical Structure & Chassis Assembly

Software Team (5 Members)

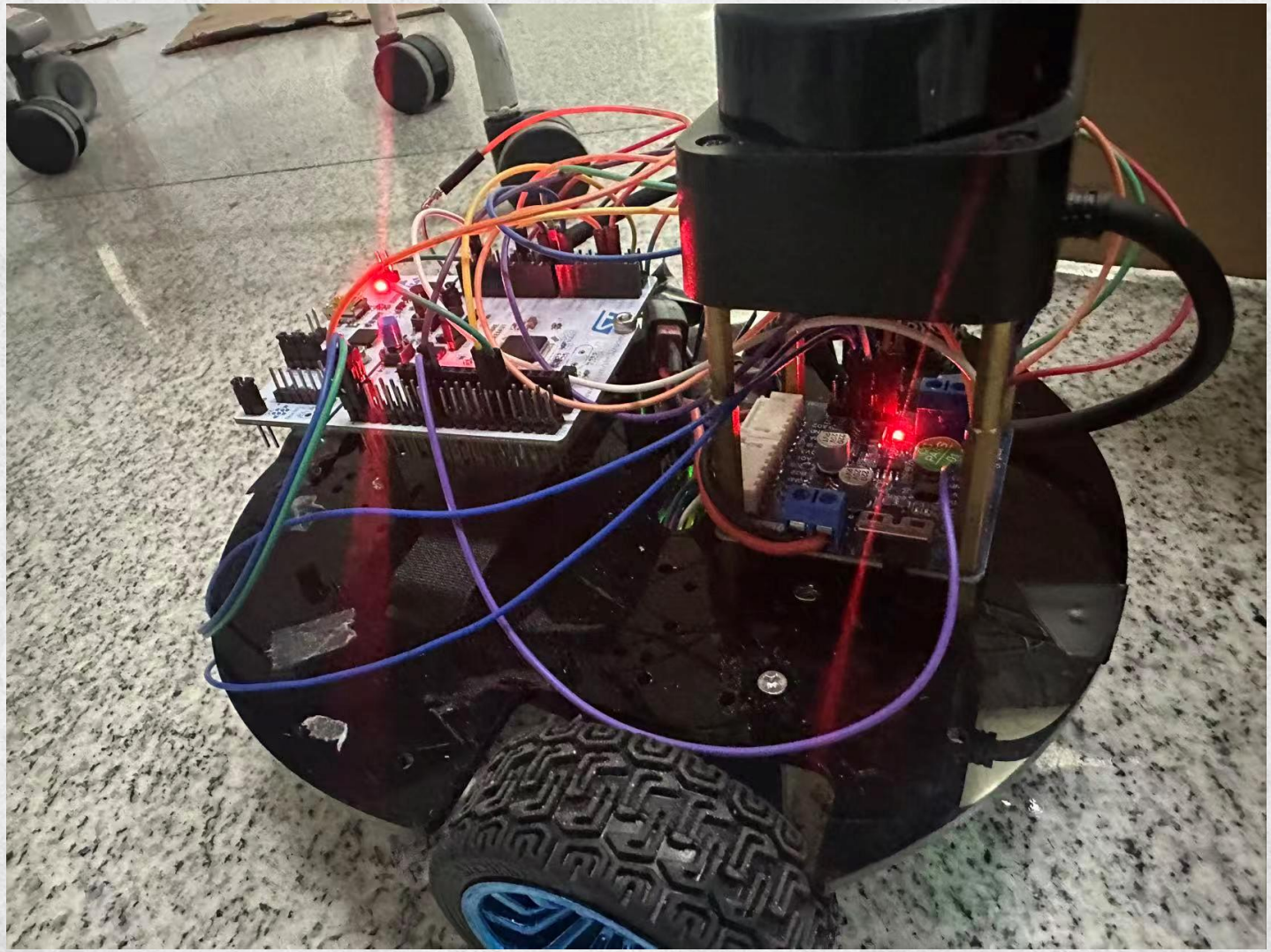
- **Xu Zihan**: SLAM Implementation (BreezySLAM)
- **Gao Yinan**: Global & Local Path Planning (A*, DWA)
- **Zhang Zeyue**: Frontier Detection & Autonomous Exploration Logic
- **Leng Yiheng, Wang Kerui**: Data Logging, Analysis & Bluetooth Communication (PySerial)

Project Overview & Design Philosophy

Mission Objective: To autonomously explore an unknown maze, identify the exit point, and return to the starting position.

Core Strategy

- Grid-Based Navigation: 70cm × 70cm cells
- Cell-by-Cell Movement: One cell at a time
- Precision Movement: 90° turns & straight lines
- Integrated SLAM: Real-time map updates



»»» 02 «««

System Architecture

System Architecture – High-Level Overview

Lower Layer: The "Nervous System"

(Firmware – STM32F446RE)

- Real-time sensor data acquisition
- Closed-loop motor control (PID)
- Odometry and pose estimation
- Bluetooth communication (HC-04)

Upper Layer: The "Brain"

(Host PC – Python)

- SLAM map construction (BreezySLAM)
- Frontier detection & path planning (A*, DWA)
- Real-time visualization (PyRoboViz)
- Navigation waypoint generation

Detailed Control Loop Architecture

Position Control Loop

01

The outer position control loop receives target positions from the host PC, calculates required movements, and outputs target speeds and angles for lower-level loops.

Input: Target (x, y, θ) → Process: Calculate motion → Output: Target speeds, angles

Angle Control Loop

02

The angle control loop uses IMU and encoder feedback to achieve precise turning. It adjusts differential PWM signals to maintain accurate heading.

Input: Target θ → Feedback: IMU + Encoders → Output: Differential PWM for turning

Speed Control Loop

03

The inner speed control loop regulates wheel speed using encoder feedback and PID control. It ensures consistent linear motion despite varying loads.

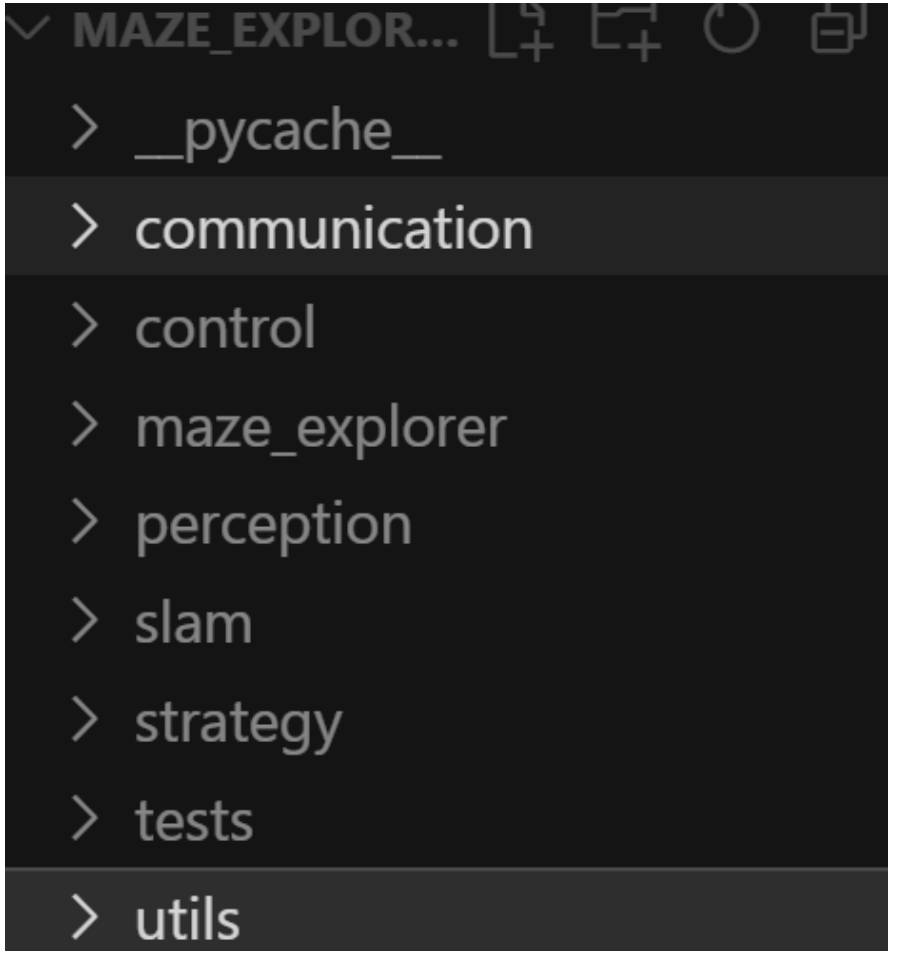
Input: Target wheel speed → Feedback: Encoder ticks → Output: Adjusted PWM via PID

»»» 03 «««

Hardware & Software Stack

Python Software Project Structure

Communication Module
Implements Bluetooth serial communication protocols
Manages data packet encoding, decoding, and transmission
Control Algorithm Module
Implements robot motion control and PID controllers
Generates motor control commands
Performs speed planning and trajectory tracking
Perception Processing Module
Fuses and preprocesses sensor data (LiDAR, IMU)
Filters IMU data and performs attitude resolution
Extracts environmental features and detects obstacles
SLAM Module
Implements laser SLAM algorithms
Manages occupancy grid map construction and updates
Handles pose estimation, map optimization, and loop closure detection
Navigation Strategy Module
Manages path planning and replanning
Handles task state machines and behavioral decisions



Software Implement ation

Path Planning

The software employs A* for global path planning and Dynamic Window Approach (DWA) for local obstacle avoidance, ensuring efficient navigation and collision prevention.

Communication Management

PySerial manages Bluetooth communication, ensuring robust data transfer between the STM32 microcontroller and the host PC for seamless control and monitoring.

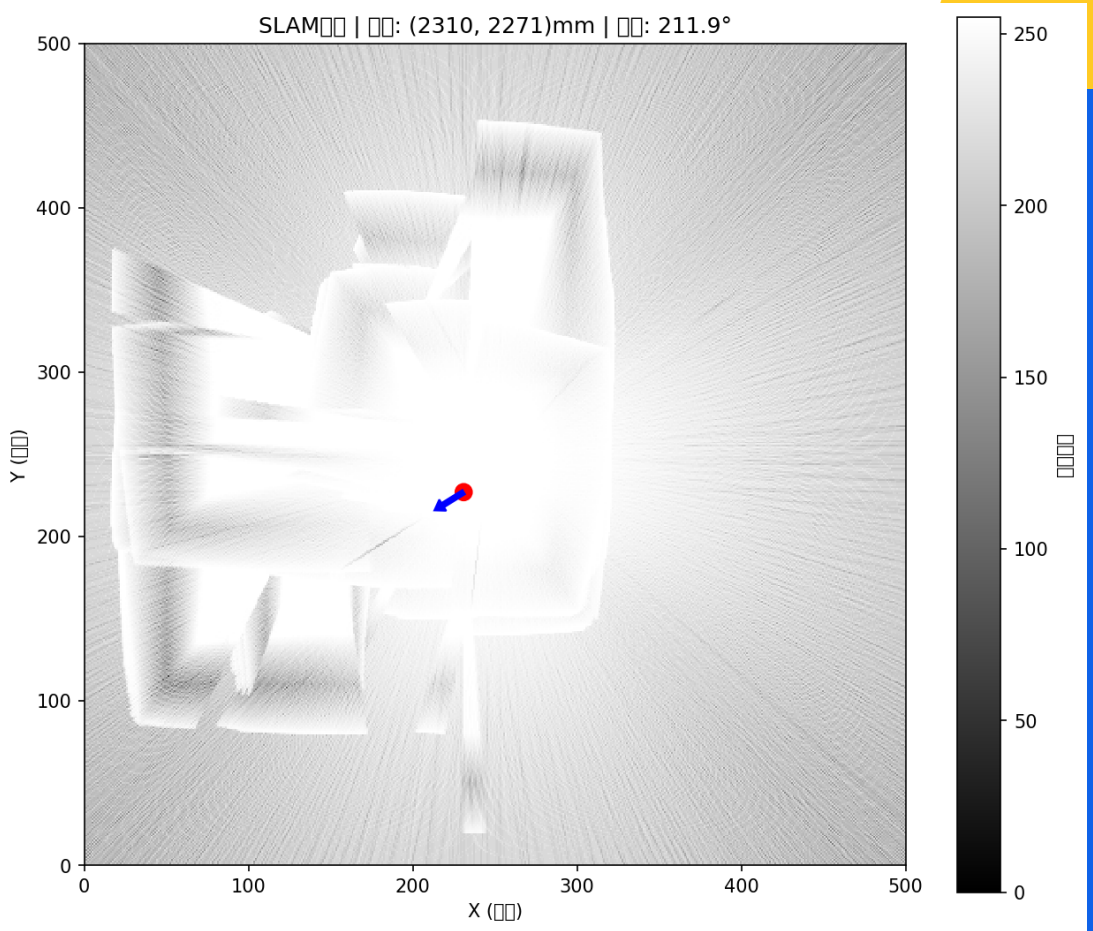
```python
def _process_buffer(self):
    while len(self.buffer) > 0:
        # 1. Detect LiDAR frame (header signature A5 5A 4C 44)
        if len(self.buffer) >= 4 and self.buffer[0:4] == b'\xA5\x5A\x4C\x44':
            break
        # 2. Text data (starting with alphabetic character)
        elif self.buffer[0:1].isalpha() or self.buffer[0:1] == b'[':
            idx = self.buffer.find(b'\n')

def _run_ble_thread(self):
    """Runs the BLE event loop in a separate thread"""
    self.loop = asyncio.new_event_loop()
    asyncio.set_event_loop(self.loop)
    self.loop.run_until_complete(asyncio.gather(*pending, return_exceptions=True))

def _parse_lidar_frame(self, frame: bytes):
    # Parse 1012-byte LiDAR frame
    timestamp = struct.unpack('<I', frame[4:8])[0]
    point_count = struct.unpack('<H', frame[8:10])[0]
    distances = struct.unpack('<500H', frame[10:1010])  # 500个uint16
    crc = struct.unpack('<H', frame[1010:1012])[0]

def send_command_with_seq(self, mode: int) -> int:
    self.command_seq = (self.command_seq + 1) % 65536
    cmd = f"CMD,{self.command_seq},{mode}\n"
```

ble_com.py

SLAM Implementation and Real-time Visualization

BreezySLAM is used for real-time 2D occupancy grid mapping, integrating LiDAR scans and odometry to create an accurate representation of the maze environment.

PyRoboViz provides real-time visualization of the map, robot pose, and planned paths. This tool enhances understanding of the robot's actions and decision-making.



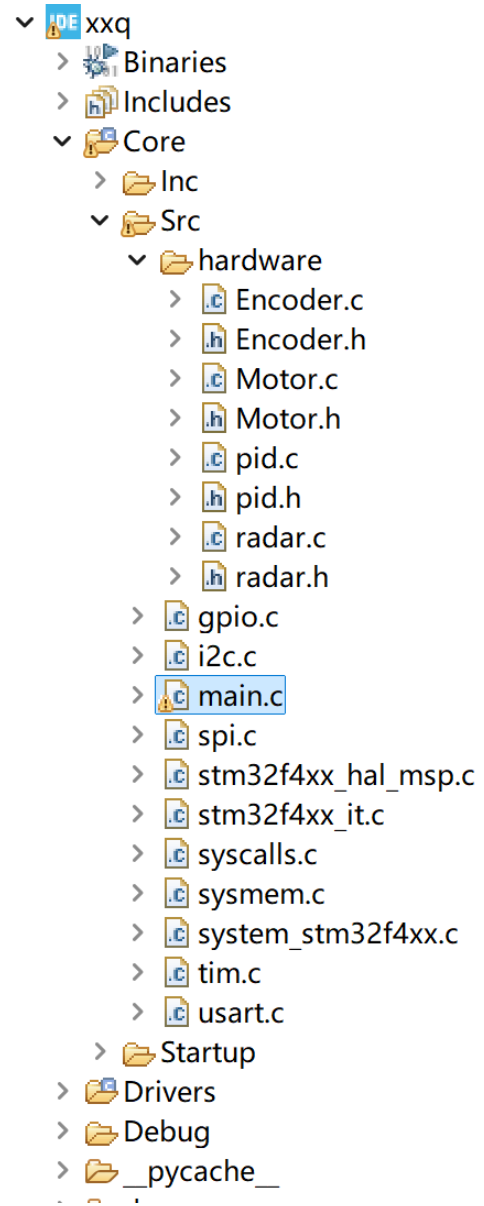
STM32 Firmware Project Structure

Core/Inc and Core/Src:
Store the core Hardware Abstraction Layer (HAL) code.

hardware:
The hardware driver layer, containing:
Encoder, Motor, PID control, radar driverVarious peripheral drivers: GPIO, I2C, SPI, USART, etc.

Startup:
STM32 startup files.

Drivers:
STM32 HAL library drivers.



Hardware Implementation

```c
while (1) {
    // Step 1: Process LiDAR data (state machine + sliding window parsing)
    LIDAR_ProcessPendingData();

    // Step 2: Send complete LiDAR frames to host PC
    if (n > 0 && LIDAR_ShouldSendFrame()) {
        LIDAR_PackAndSendRawFrame(points_buf, n);
    }

    // Step 3: System status update and sensor processing
    System_StatusUpdate();
    UART5_ProcessSensorData();

    // Step 4: PID control loop (20ms cycle)
    Motor_PID_Control(&htim1, &htim3, &htim2);

    // Step 5: Process Python commands
    Python_ParseCommand();

    // Step 6: Action completion detection and ACK response
    if (g_action_in_progress) {
        // Detect action completion and send ACK
    }
}
```

Main.c

STM32F446RE MCU

Serves as the core component responsible for real-time control, sensor data acquisition, and communication.

SLAMTEC RPLIDAR C1

A 2D LiDAR scanner used for 360° environmental perception, providing distance and angle data for SLAM.

Hardware Implementation

01

MPU6500 IMU

A 6-axis Inertial Measurement Unit that supplies orientation data (yaw angle) to supplement odometry and enhance angle control.

02

AT8236 Motor Driver & 520 Geared Encoder DC Motors

Deliver locomotion capability. The encoders provide critical feedback for speed control and odometry calculations.

03

HC-04 Bluetooth Module

Enables seamless wireless data/command exchange between the robot and the host PC.

Calibration & Error Mitigation

01

Encoder Calibration

Process: Command the robot to move a known distance and record the total encoder pulse count.

Result: Calculate the conversion factor (meters per pulse) for precise odometry.

Validation Method: Rotate the robot 10 full circles, measure the deviation from the theoretical 3600°, and perform error compensation.

02

IMU Calibration

Initial Calibration: Perform zero-offset calibration to define the initial heading (0°).

Continuous Optimization: Implement complementary filters to fuse gyroscope and accelerometer data, mitigating drift over time.

Performance Validation: Validate accumulated angular error through the 10-circle rotation test and apply compensation.

03

LiDAR Calibration

Position Calibration: Measure and correct the LiDAR's positional offset relative to the robot's rotation center.

Data Alignment: Ensure scan points accurately correspond to the robot's base coordinate frame.

Accuracy Verification: Test the alignment of scan data with maps in known environments.

»»» 04 «««

Results

Results & Demonstration

Basic Level

- Stable motor control (PID)
- Accurate odometry & pose
- LiDAR streaming & visualization

Intermediate Level

- Autonomous exploration
- Real-time SLAM with loop-closure
- Exit point identification

Advanced Level

- A* path planning (exit to start)
- Complete mission autonomy

»»» 05 «««

Summary & Outlook

Project Summary & Lessons Learned

Successes

The project achieved precise movement through PID tuning, seamless hardware-software integration, and stable Bluetooth communication, demonstrating robust autonomous navigation.

Challenges & Solutions

Challenges such as IMU drift and LiDAR noise were addressed through sensor fusion and data filtering. Bluetooth latency was mitigated by optimizing packet size and refresh rates.

Future Improvements

Future work includes adaptive PID gains, Extended Kalman Filter for enhanced sensor fusion, and a web-based interface for remote control and monitoring.

THANK YOU

Group 12

2025/10/21