Embedded System and Microcomputer Principle Project Report

Site Fan, Jiachen Xiao, Jiawei Fang, Ben Chen

ShixSU7: STM32-based smart vehicle with controller

December 23, 2024

Topic: Smart Vehicle
Group No.7

#	Name	SID	Coefficient
0	樊斯特	12111624	25%
1	肖佳辰	12112012	25%
2	方嘉玮	12110804	25%
3	陈贲	12212231	25%

Contents

1. System Function 4
   1.1. Task 1 4
   1.2. Task 2 4
   1.3. Task 3 4
   1.4. Task 4 4
   1.5. Task 5 4
2. System Design 4
   2.1. Working Principles 4
      2.1.1. Control Codes 5
      2.1.2. ACK1 Codes 5
      2.1.3. ACK2 Specific Codes 6
   2.2. System Frame Diagram 6
   2.3. Submodule Design 6
      2.3.1. SU7 6
      2.3.2. LeiJun 7
3. Results 7
4. Work Allocation 7
5. Issues and Solutions 7
   5.1. Bluetooth not working 7
   5.2. Oscillating reading of ultrasonic sensor 8
   5.3. Communication of commands misbehaved 8
   5.4. Bias in direction 8
6. Project Proposal 8
   6.1. STM32 Quadrotor Drone 8
   6.2. Requirements 9

1. System Function

1.1. Task 1

- We establish the connection between the controller and vehicle by bluetooth, and the former will start to display the distance reading from ultrasonic sensor. Otherwise, the screen will show error message when handshakes timeout.
- The controller is able to control the 9 kinds of motion in vehicle, which are forward, backward, rotate left and right, turn left and right, parallelly move left and right and stop.
- **Advanced.** We designed a communication protocol to ensure communication.

1.2. Task 2

- The vehicle can detect the obstacles with ultrasonic sensors in maximum speed.
- The distance reading is displayed on screen in real time.

1.3. Task 3

- We set the ground to $3.2m \times 3.2m$ with 4 by 4 squares. This is graphically shown on screen with text and color annotation.
- Besides manual mode, we have Waypoint and Auto mode. In Waypoint mode, user firstly sets the location of obstacles by clicking the squares. After that, user presses KEY1 and continues to click or swipe the screen for the routes, with arrows indicating the routes showing on screen as well. These points cannot be set on obstacles. The two points can be modify at anytime. With all points set, user sends command to vehicle to start moving by clicking the "Go" button.
- **Advanced** The vehicle can adjust its direction by measuring the angles between itself and the border line. This feature ensures the vehicle to run without any external physical adjustment, like kicking it.

1.4. Task 4

- In Auto mode, user only needs to set the start and end location. Then user click "Go" to starts the vehicle. Along the travelling, the vehicle will explore the ground for visible obstacles by ultrasonic sensor in a breath-first-search way. Next, the car will sort a possible route out (possibly the shortest) to the termination.
- In Auto mode, the vehicle will send the location of itself when entering a sqaure and location of a obstacle when detected, and information will displayed on screen, with red square indicating a obstacle and "()" indicating the vehicle.

1.5. Task 5

The vehicle can automatically track the guide lines with the grayscale sensor, simply by changing mode to "Race" on controller.

2. System Design

2.1. Working Principles

The communication protocol operates in a cyclic manner as follows:

1. **Control Code Transmission**: LeiJun initiates the process by sending a control code.
2. **ACK1 (Optional)**: SU7 responds with ACK1 if data transmission is required.
3. **Data Transmission (Optional)**: Either LeiJun or SU7 sends the data.
4. **ACK2**: SU7 sends ACK2 to acknowledge the receipt of the control code and any data transmitted.
5. The cycle repeats, starting with another control code.

- **No Data Transmission**: If no data is being sent, ACK1 and data transmission are skipped.
- **Role of Devices**: Control codes are always sent by LeiJun, while ACK1 and ACK2 are sent by SU7. Data can be sent by either device.

2.1.1. Control Codes

Control Code	SU7 <-> LeiJun	Data Length
0x00	Greeting, asking data	0byte
0x01	Set to control mode	0byte
0x02	Set to waypoint mode	0byte
0x03	Set to autopilot mode	0byte
0x04	Set to auto race mode	0byte
0x05	Set mode start	0byte
0x06	Set mode STOP	0byte
0x10	Set all wheel control	4byte
0x11	Set left1 wheel control	1byte
0x12	Set left2 wheel control	1byte
0x13	Set right1 wheel control	1byte
0x14	Set right2 wheel control	1byte
0x20	Set waypoint	n byte, `0xFF` for end
0x30	Set autopilot mode position	2byte
0x80	Get obstacle distance	4byte
0x81	Get obstacle position	1byte
0x90	Get SU7 position	1byte (2bit 0, leading 2bit 0, 4bit position)
0x9x	Get SU7 state	Reserved

2.1.2. ACK1 Codes

ACK1/2 Code	Function
0x00	ACK
0xF0~0xFF	Error codes

0xF0	Out of range
0xF1	Set mode failed, wait for end or STOP
0xFF	NACK

2.1.3. ACK2 Specific Codes

ACK2 Code	Function
0x80~0xEF	SU7 need send data, after LeiJun receive this, send this as control code to fetch data.

2.2. System Frame Diagram

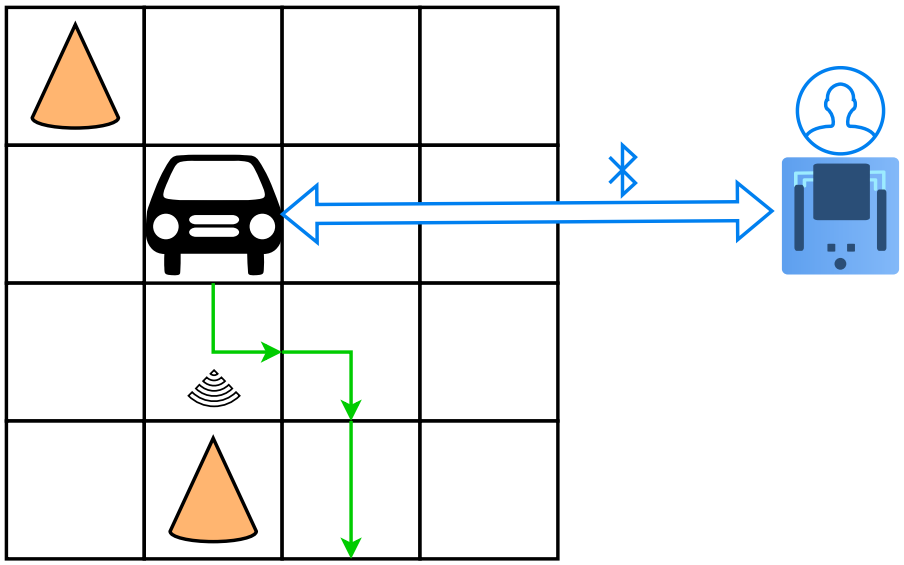

Figure 1: System Diagram of ShinxSU7

2.3. Submodule Design

The system can be divided into two submodules, SU7 and LeiJun.

2.3.1. SU7

Our vehicle is named SU7. In our protocol, SU7 is the slave component.

- In the lowest layer, the code of SU7 is a set of drivers that control the hardware by manipulating and reading the registers, for example, calculating the distance of obstacles from ultrasonic sensor by the timer interrupts and setting the motors input voltage by PWM.
- In the middle layer, SU7 will perform some optimization to improve the effects of each action. Because of the lack of sensors such as gyroscope, SU7 is unable to know if it actually rotate for a certain degrees. SU7 will calibrate the bias implicitly.
- The top layer controls the states of SU7 and communicates with LeiJun. For instance, in Waypoint mode, SU7 pushes the waypoint information to the queue and starts when receiving command. During jobs, SU7 will compute the running time elapsed for the motors to move to the next waypoint.

2.3.2. LeiJun

The controller "LeiJun" is the master in protocol. LeiJun interacts with user and sends user's commands to SU7, for example

- In Manual mode, LeiJun explicitly controls the motors by setting the speed respectively.
- In Waypoint mode, LeiJun won't tell the location of obstacles, instead, the location is only used for forbidding the user to send incorrect waypoints. LeiJun will only sends the waypoint information.
- In Auto mode, after sending departure and destination information, LeiJun continously fetches the location data of SU7 and detected obstacles, and displays them to the user.

3. Results

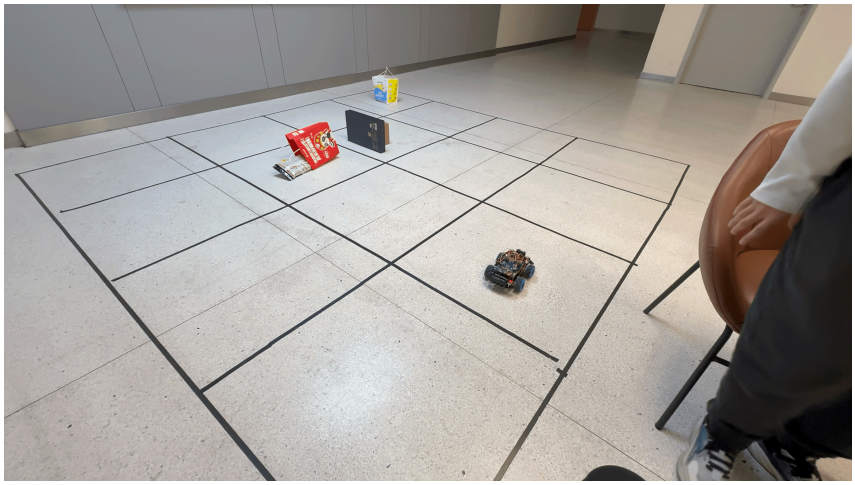

Figure 2: Showcase of working vehicle

4. Work Allocation

Name	Job
Site Fan	He is the main designer of LeiJun and assists in designing the protocol
Jiachen Xiao	He migrates the driver to CubeMXIDE, builds the framework of our design and defines the communication protocol.
Jiawei Fang	He develops the autopilot of vehicles.
Ben Chen	He debugs the Bluetooth module and assists in developing LeiJun and SU7.

5. Issues and Solutions

5.1. Bluetooth not working

Bluetooth is a wireless bridge of UART, which means ideally this can works with the code of UART. But the following problems came across

- Binding of incorrect pins. Bluetooth pins on vehicle is attached to PA9/PA10, but they are not responding. So we moved them to USART2 and PA2/PA3, and connected through dupont wire.

- Setting interrupts. Both components receive data by interrupt. The missing activation of UART interrupt blocks the data.

5.2. Oscillating reading of ultrasonic sensor

The distance measured by ultrasonic sensor was incorrect and fluctuated ridiculously. This was caused by the poor contact of pins.

5.3. Communication of commands misbehaved

The issue was due to three reasons

- Insufficient array size leads to buffer overflow, causing the incorrect control flow.
- The command is sent incorrectly in LeiJun, like `0x30` was wrongly written as `0x31`.
- Insufficient input voltage biases the motors output.

After a long period of debugging, we could solve the issue by fixing mistakes.

5.4. Bias in direction

Lack of sensors like gyroscope affects accurate rotation.

To solve the issue, we invented a algorithm to eliminate the bias by calibrating the configuration from the sensor data. We have 3 grayscale sensors aligned and if there's angles between them and the grid line, most likely one or two sensors have detected the black line, so we can infer the bias and adjusting the direction until all three sensors detect the line.

6. Project Proposal

6.1. STM32 Quadrotor Drone

Besides smart vehicles, we can also build a drone with STM32, for example, in blog post[1], a drone development suite is available on Taobao[2]. Through this project, students can gain a rich experience of in designing a self-calibration system with sensors and flight control algorithm.

Figure 3: Quadrotor Drone with STM32F103C8T6

[1] https://blog.csdn.net/black_sneak/article/details/130749146

[2] https://item.taobao.com/item.htm?id=640798672705

The suite also comes with a tiny rocker controller but lack of functions, so it can be further enhanced with a MiniSTM32 controller, to achieve some features like setting a autopilot, obstacles detection with infrared sensor and shown on screen.

6.2. Requirements

It's fully feasible since the demo code to fly the drone is available online. Similar to our project, we can improve the codes with the following tasks

- **Task 1** Controller with MiniSTM32 and rocker (20%).
  - The rocker is used to manually control. It communicates with board by wire, while board establishes the connection with drone by bluetooth or 2.4G.
  - Pedal lock. It's danger when accidentally trigger the flight, so you need to implement a simple lock with some gesture.
  - Control with touch screen, set flight route through screen and command to fly back to some place when losing connection.
- **Taks 2** Fly (25%)
  - Improve the flight control and adjust the posture of drone in the air.
  - Implement rolling along the X axis.
- **Task 3** Obstacle avoidance (20%)
  - With ultrasonic and infrared the drone can automatically detour.
- **Task 4** Spatial Modeling and Race (35%)
  - This requires a advanced motion of drone, like the FPV. In this task, the drone will run at maximum speed and pass through a more complicate situation.