**南京邮电大学工程训练中心**

**电装实习**

**创新项目实验报告**

|  |  |
| --- | --- |
| **项目名称：遥控小车** | |
| **项目组成员** | |
| **学号:** |  |
| **姓名：** |  |
| **分工：** |  |
| **所在学院：** |  |
| **所在专业：** |  |
| **学号：** |  |
| **姓名：** |  |
| **分工：** |  |
| **所在学院：** |  |
| **所在专业：** |  |
| **学号：** |  |
| **姓名：** |  |
| **分工：** |  |
| **所在学院：** |  |
| **所在专业：** |  |

## 一、项目的主要任务

利用ESP8266，实现基于UDP服务的远程控制小车，可以在电脑端控制小车前进、后退等操作。

## 二、团队成员分工

负责对项目的各个代码进行编写，主要的代码实现有控制系统模块，通信系统模块，以及输出模块等。

负责项目的原理图，以及PCB的设计，实物的组成

负责项目原件的购买，元器件焊接，以及实物的组装

## 三、项目的主要设计思路

小车的设计采取模块化设计，主要的模块有底盘模块、传感器模块、控制系统模块、通信模块、电源管理模块、接口与通信模块、输出模块，分别将各个模块完成调试好后进行组装，最后完成项目的小车设计

**3.1硬件选择与组装：**

选择轻便且强力的电机和可靠的驱动器，确保小车可以在不同地形上平稳行驶。采用ESP8266模块，因其成本效益高，且易于编程，能够有效地接入无线网络。设计一个紧凑且稳定的小车结构，考虑到电池的放置和电路的布局，确保所有部件都能在小车上稳定运行。

**3.2通信协议设计：**

设计一个简单而有效的通信协议，规定消息的格式和类型，确保数据传输的可靠性和效率。通信协议要考虑到错误检测和恢复机制，以应对可能的数据传输错误。

**3.3软件编程与逻辑控制：**

使用uPyCraft进行ESP8266的编程，开发控制小车运动的核心代码，包括速度控制、方向控制等。开发UDP服务端程序，运行在电脑上，接受用户的控制命令并发送给小车。在小车端，实现一个UDP客户端程序，能够接收并解析来自电脑端的指令，控制小车的行动。

## 四、项目关键元器件的选型

**4.1底盘模块：**

DC减速电机，例如金属齿轮减速电机，电压可根据需求选择，通常在6V到12V之间。

轮子：橡胶轮子，直径和轨迹要考虑小车的尺寸和预期地形。

编码器： 光电编码器，用于测量轮子的旋转，例如红外线编码器采用亚克力板，采用三轮式两个前驱动轮，加一个后万向轮，通过两前驱动轮差速进行转向。

**4.2控制系统模块：**

处理器：单片机或嵌入式计算机，例如Arduino、Raspberry Pi。

电机驱动器：驱动电机的电路板， L298N电机驱动模块。

通信模块： 通过通信代码的实现使用ESP8266来完成Wi-Fi模块。

电源管理模块：电池：UM3，开关采用 KCD11

输出模块：0.96寸OLED （IIC 模式）

## 五、项目的整体设计方案

**结构设计：**

小车框架：采用轻质亚克力板材料，结构以三轮设计为主，两个动力轮提供动力和转向，一个万向轮保持平衡。

电机安装：电机固定在小车底部，通过轴和轮子相连，确保动力传递。

**电路设计：**

主控制单元：ESP8266模块，负责处理来自远程控制端的信号，并控制小车的运行。

电机驱动：L298N驱动模块连接电机和ESP8266，控制电机的转速和方向。

电源管理：采用UM3型电池供电，KCD11开关控制电源的开关。

传感器接口：超声波传感器与ESP8266相连，实时反馈障碍物信息。

**软件逻辑设计：**

远程控制协议：设计一套UDP通信协议，实现电脑端软件与小车之间的实时通信。

控制逻辑：编写代码控制小车根据接收到的指令进行前进、后退、转向等操作。

避障算法：实现简单的避障逻辑，当传感器检测到障碍物时，小车自动停下或绕行。

**用户界面设计：**

远程控制端：开发一个图形用户界面，包含控制按钮和小车状态显示。

小车显示：0.96寸OLED显示屏显示小车状态或其他重要信息。

## 六、项目的主要工作原理

**通信机制：**

项目使用UDP协议实现远程控制功能。在电脑端，一个UDP服务端应用程序发送控制命令；在小车端，ESP8266模块作为UDP客户端接收这些命令。当ESP8266接收到命令时，它解析这些指令，并将相应的控制信号传递给电机驱动器和其他相关模块。

**驱动与运动控制：**

小车的移动由DC减速电机提供动力。电机通过L298N电机驱动模块接受来自ESP8266的控制信号。控制信号确定电机的转动方向（前进、后退）和速度。改变两个电机的速度或方向可以实现小车的转向。

**电源管理：**

小车通过UM3型电池供电，保证持续的电力输出。KCD11开关用于控制电源的开启和关闭。电源管理系统确保所有组件，包括ESP8266、电机驱动器和传感器，都有稳定的电力供应。

**用户交互：**

用户通过电脑端的图形界面发送控制指令，如前进、后退、转向等。用户界面还可以显示小车的状态信息，如速度、电量和环境感知数据。

**数据显示与反馈：**

小车装有OLED显示屏，实时显示工作状态、电量或其他重要信息。这提供了直观的反馈，帮助用户了解小车的实时状态和环境。

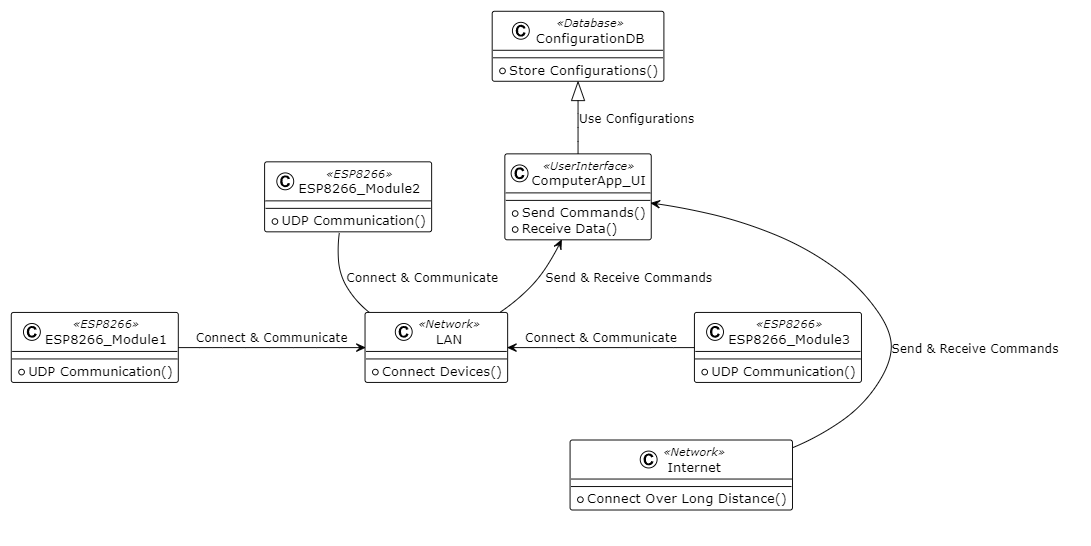


图1 工作原理

## **七、项目的硬件设计**（注：原理图和PCB设计及说明）

**原理图设计：**

ESP8266模块：作为核心控制单元，与其他所有模块进行通信和控制。

电机和驱动器：电机通过L298N驱动模块接入，该模块负责控制电机的转速和方向。

电源管理：电池连接到电源管理模块，该模块负责分配电源到ESP8266、电机驱动器以及其他传感器。

输出显示：OLED显示屏作为输出设备，显示状态信息。

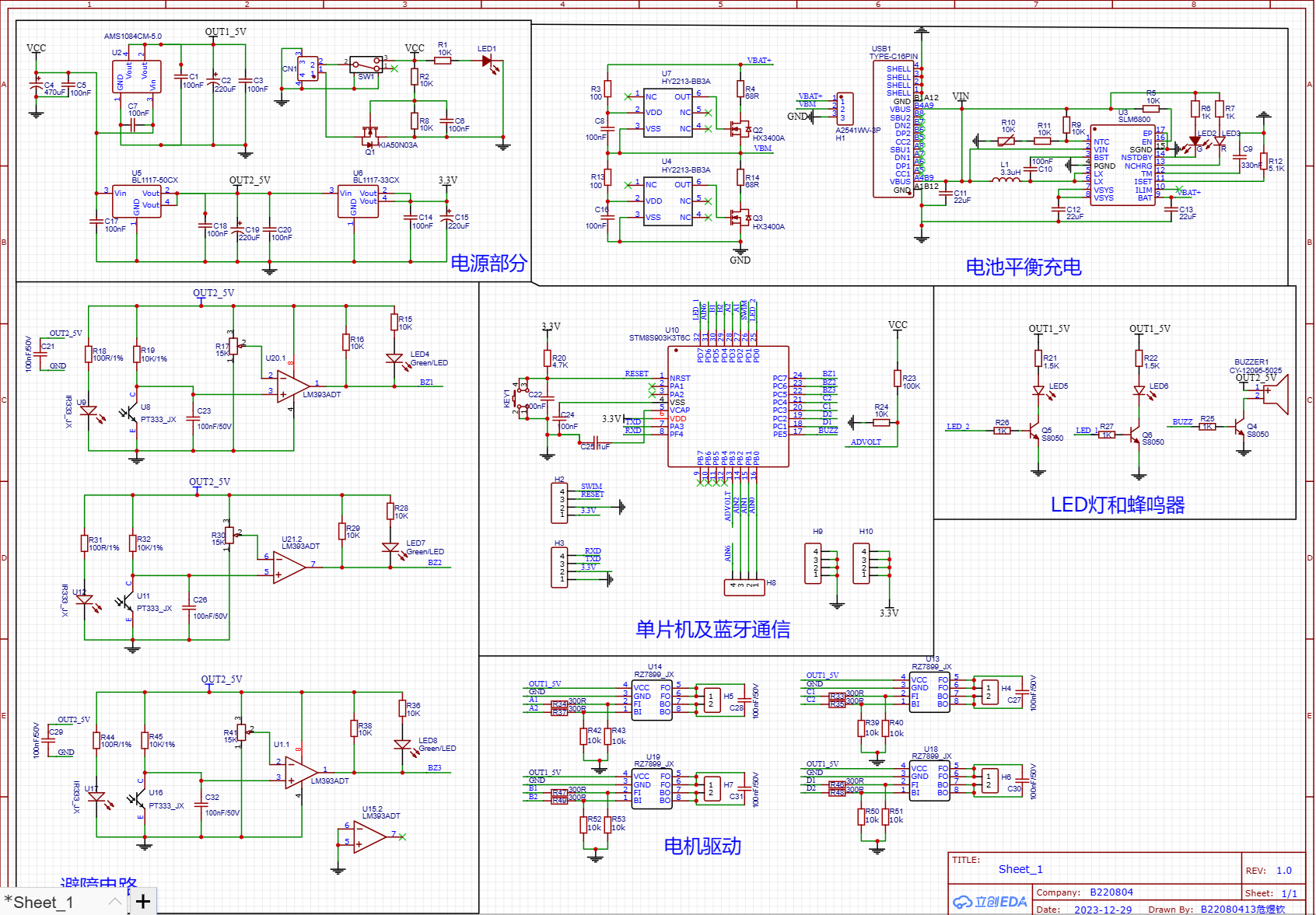


图2 原理图

**PCB设计：**

布局：设计PCB时，首先要考虑的是组件的物理布局。ESP8266、L298N、电源管理模块、传感器和显示屏需要被合理布局，以确保小车的紧凑性和稳定性。

线路设计：所有电子元件通过铜线路连接，设计时需要考虑信号完整性和电路的电磁兼容性。避免长线路和紧邻的高频线路，以减少噪声和干扰。

多层设计：根据需要，PCB可能是单层或多层设计。多层设计可以更有效地布局复杂的电路和提供更好的信号完整性。

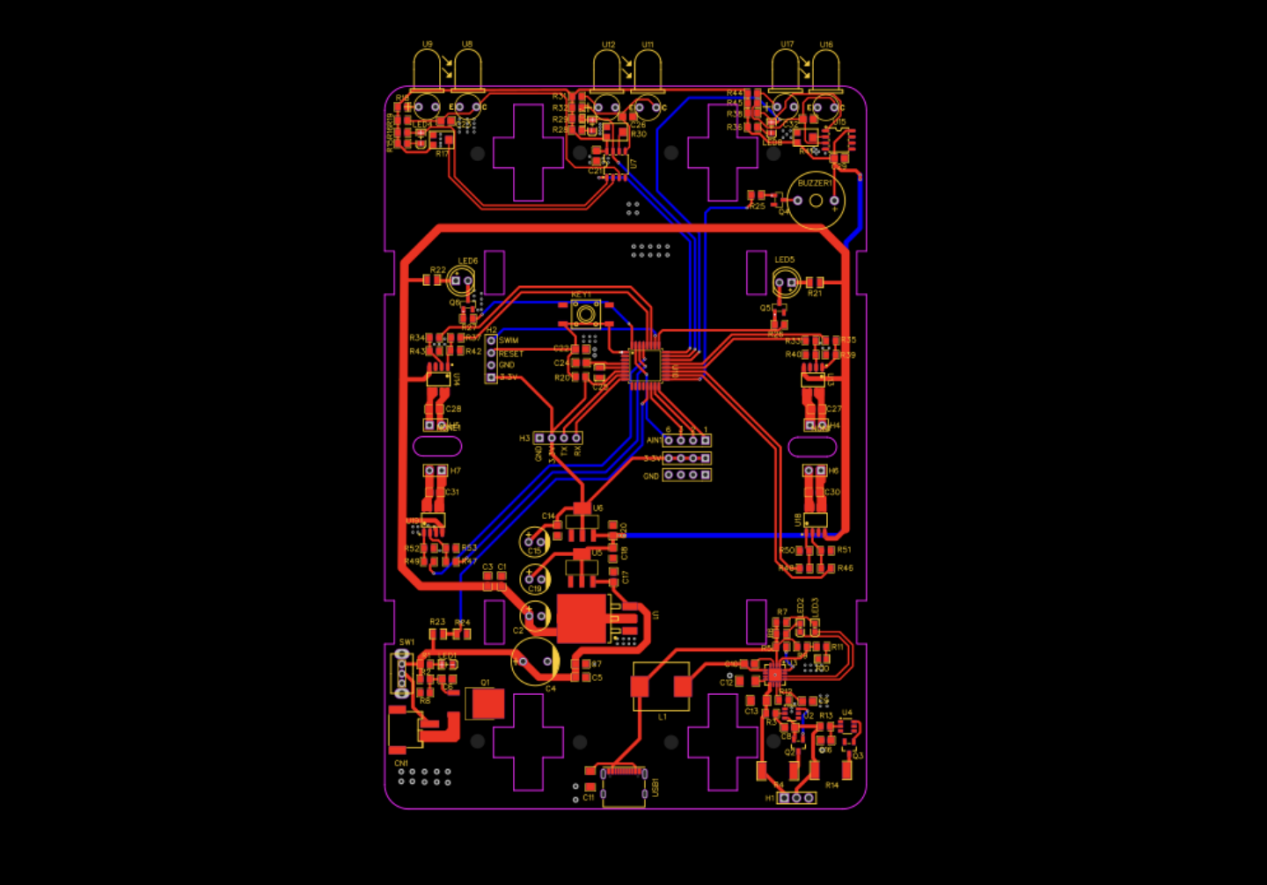
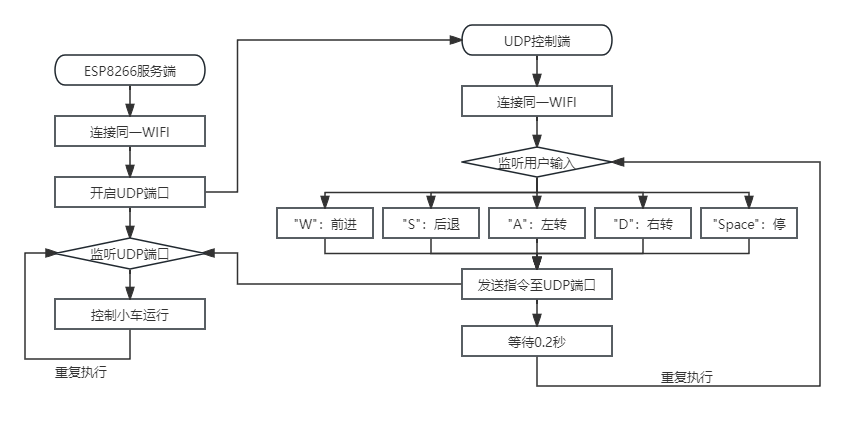


图3 PCB图

## **八、项目的软件设计**

图4 流程图

**开发环境：**

名称：uPyCraft

选择理由：uPyCraft是一种专为MicroPython设计的IDE，支持ESP8266等微控制器。它提供简单易用的图形界面，支持代码编写、运行和调试。对于使用MicroPython进行ESP8266开发，uPyCraft提供了便捷的开发体验。

**开发和测试：**

代码编写：在uPyCraft中编写控制逻辑和通信逻辑代码。

模块测试：逐个测试各个功能模块，如Wi-Fi连接、UDP通信、电机控制等。

集成测试：将所有模块集成在一起，进行全面测试，确保系统稳定运行。

实地测试：在实际环境中对小车进行测试，检查避障、远程控制等功能。

## **九、项目成品**

1. **Wi-Fi连接**

配置网络：在启动ESP8266模块时，它将尝试连接到预先配置的Wi-Fi网络。

连接状态：通过串口监视器查看Wi-Fi连接状态。

1. **远程控制**

启动控制软件：在电脑端启动UDP服务端程序。

发送控制命令：使用图形界面发送前进、后退、左转、右转等控制命令。

监控小车：通过软件界面监控小车的响应和状态。

1. **状态监测**

反馈信息：小车会通过UDP包将执行动作后的状态信息反馈给控制端。

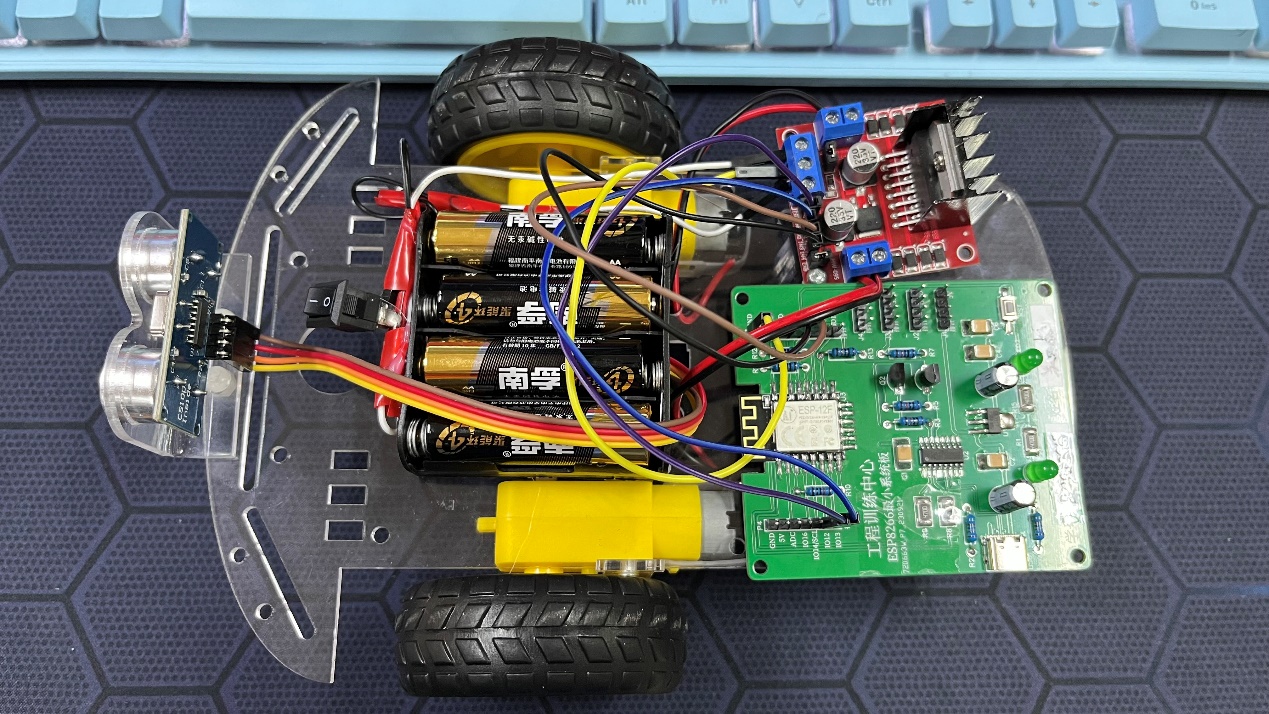


图5 小车成品

## 十、附录

**所需第三方依赖：**

keyboard==0.13.5

**服务端代码service.py：**

import machine

import time

import network

import socket

# 定义L298N连接到ESP-12F的引脚

IN1\_PIN = 12 # IO12

IN2\_PIN = 13 # IO13

IN3\_PIN = 4 # IO4

IN4\_PIN = 5 # IO5

# 初始化引脚

in1 = machine.Pin(IN1\_PIN, machine.Pin.OUT)

in2 = machine.Pin(IN2\_PIN, machine.Pin.OUT)

in3 = machine.Pin(IN3\_PIN, machine.Pin.OUT)

in4 = machine.Pin(IN4\_PIN, machine.Pin.OUT)

# 初始化网络

wifi\_ssid = "Sucial"

wifi\_password = "qwertyuiop"

udp\_port = 12345

sta = network.WLAN(network.STA\_IF)

sta.active(True)

sta.connect(wifi\_ssid, wifi\_password)

# 等待连接成功

while not sta.isconnected():

pass

# 打印ESP8266的IP地址

print("Connected to WiFi")

print("IP Address:", sta.ifconfig()[0])

# 设置UDP服务器

udp\_server = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

udp\_server.bind(('0.0.0.0', udp\_port))

# 接收并解析UDP命令

def process\_udp\_command(data):

if data == b'forward':

forward()

elif data == b'backward':

backward()

elif data == b'left':

left()

elif data == b'right':

right()

elif data == b'stop':

stop()

# 控制小车向前

def forward():

in1.off()

in2.on()

in3.on()

in4.off()

# 控制小车向后

def backward():

in1.on()

in2.off()

in3.off()

in4.on()

# 控制小车左转

def left():

in1.off()

in2.on()

in3.off()

in4.on()

# 控制小车右转

def right():

in1.on()

in2.off()

in3.on()

in4.off()

# 停止小车运动

def stop():

in1.off()

in2.off()

in3.off()

in4.off()

# 循环，等待UDP命令

try:

while True:

data, addr = udp\_server.recvfrom(1024)

process\_udp\_command(data)

except KeyboardInterrupt:

udp\_server.close()

**控制端代码control.py：**

import socket

import time

import keyboard

# ESP8266的IP地址和端口号

esp\_ip = "172.20.10.8"

esp\_port = 12345

# 创建UDP客户端

udp\_client = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

# 定义小车运动命令

commands = ["forward", "backward", "left", "right", "stop"]

print("通过'W':前进，'S':后退，'A':左转，'D':右转，'Space':刹车，来控制小车")

try:

while True:

if keyboard.is\_pressed('w'):

udp\_client.sendto(commands[0].encode(), (esp\_ip, esp\_port))

print("Send Command: Forward")

time.sleep(0.2)

udp\_client.sendto(commands[4].encode(), (esp\_ip, esp\_port))

elif keyboard.is\_pressed('s'):

udp\_client.sendto(commands[1].encode(), (esp\_ip, esp\_port))

print("Send Command: Backward")

time.sleep(0.2)

udp\_client.sendto(commands[4].encode(), (esp\_ip, esp\_port))

elif keyboard.is\_pressed('a'):

udp\_client.sendto(commands[2].encode(), (esp\_ip, esp\_port))

print("Send Command: Left")

time.sleep(0.2)

udp\_client.sendto(commands[4].encode(), (esp\_ip, esp\_port))

elif keyboard.is\_pressed('d'):

udp\_client.sendto(commands[3].encode(), (esp\_ip, esp\_port))

print("Send Command: Right")

time.sleep(0.2)

udp\_client.sendto(commands[4].encode(), (esp\_ip, esp\_port))

elif keyboard.is\_pressed('space'):

udp\_client.sendto(commands[4].encode(), (esp\_ip, esp\_port))

print("Send Command: Stop")

time.sleep(0.2)

except KeyboardInterrupt:

print("Exit")

udp\_client.close()