# Chapter15 TCP远程操控

## 实验目的

通过键盘按键来远程操控小车动作。

## 实验原理

引入了两个模块：socket和pygame。

socket：用来实现tcp通信。

pygame：用来实现按键捕捉（也可使用keyboard模块）。

将我们的pc作为客户端，通过按键捕获，将相应的命令字符串发送给服务端（树莓派），树莓派进行命令解析，执行相应的动作。

## 功能实现

我们设定一个命令格式`$cmd`。服务端发送命令字符串给树莓派，树莓派进行解析，执行相应功能。

- 开机

命令格式：`$poweron`

按下tab键开机，在不开机的情况下无法控制小车的任何功能。

- 关机

命令格式：`$poweroff`

在开机状态下，按下esc键小车关机。

- 行驶

命令格式：`$car_xxx`

提供前进（w或上箭头），后退（s或下箭头），左转（a或左箭头），右转（d或右箭头），原地左转（q），原地右转功能（e）,自动驾驶（0）

- LED

对LED首先设置一个开关命令`$turn_on_light`和`$turn_off-light`（空格键）

其次规范LED命令格式：`$LED_xxx`

## 代码实现

### 客户端

```python
import time
import pygame
import socket

recvStr=""
sendStr=""
cmd=""

light_flag = 0
#客户端
serverName="192.168.137.163"        #树莓派ip
serverPort=1113                     #端口号
clientSocket=socket.socket(socket.AF_INET,socket.SOCK_STREAM)

def tcp_client_connect():
    global clientSocket,serverPort,serverName
    clientSocket.connect((serverName, serverPort))
    print("connect successful")

def tcp_client_send(cmd):
    global clientSocket
    clientSocket.send(cmd.encode())
    #print("send successfully")

def tcp_client_send_close(cmd):
    global clientSocket
    clientSocket.send(cmd.encode())
    clientSocket.close()

def tcp_client_recv():
    global clientSocket
    recvStr=clientSocket.recv(1024)
    clientSocket.close()

def key2cmd(event):
    global cmd
    global light_flag
    ek = event.key
    if ek == pygame.K_UP or ek == pygame.K_w:
        cmd = "$car_up"
    elif ek == pygame.K_DOWN or ek == pygame.K_s:
        cmd = "$car_back"
    elif ek == pygame.K_LEFT or ek == pygame.K_a:
        cmd = "$car_left"
    elif ek == pygame.K_RIGHT or ek == pygame.K_d:
        cmd = "$car_right"
    elif ek == pygame.K_q:
        cmd = "$car_spinLeft"
    elif ek == pygame.K_e:
        cmd = "$car_spinRight"
    elif ek==pygame.K_0:
        cmd="$car_autodrive"
    elif ek==pygame.K_SPACE:
            cmd="$turn_off_light" if light_flag else "$turn_on_light"
            light_flag = not light_flag
        elif ek == pygame.K_1:
            cmd="$LED_red"
        elif ek==pygame.K_2:
            cmd="$LED_green"
        elif ek==pygame.K_3:
            cmd="$LED_blue"
        elif ek==pygame.K_ESCAPE:
            cmd="$poweroff"
        elif ek==pygame.K_TAB:
            cmd="$poweron"

        else:
            pass

pygame.init()
screen=pygame.display.set_mode((400,400))
try:
    tcp_client_connect()
    while 1:
        for event in pygame.event.get():
            if event.type==pygame.QUIT:
                exit(0)
            if event.type == pygame.KEYDOWN:
                key2cmd(event)
                tcp_client_send(cmd)
            #print(cmd)
            #print(time.ctime())
            cmd=""
except:
    print("error")
```

服务端

```
# 服务端
import socket
from time import *
from car_run import *
from led import *
from auto_drive import *

recvBuf=""
sendBuf=""
light_flag=0
power_flag=0

serverPort = 1113
serverSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
serverSocket.bind(("", serverPort))
serverSocket.listen(1)
```

```python
print("The server is already to receive!")
global connectionSocket
connectionSocket, addr = serverSocket.accept()



def tcp_server_recv():
    global connectionSocket, recvBuf
    recvBuf = connectionSocket.recv(50).decode()
    print("receive successful:"+recvBuf)

def tcp_server_recv_close():
    global connectionSocket,recvBuf
    connectionSocket, addr = serverSocket.accept()
    print(addr)
    recvBuf= connectionSocket.recv(50).decode()
    print(recvBuf)
    connectionSocket.close()
    print("receive successful")

def tcp_server_send(str):
    global connectionSocket
    serverSocket.send(str.encode())
    connectionSocket.close()

def tcp_server_parse(recvBuf):
    if recvBuf[0]=="$":
        global light_flag
        global power_flag
        if recvBuf.find("poweron")!=-1:
            power_flag=1
            car_run_init()
            led_init()

        if power_flag==0:
            print("小车未开机")
            return

        #小车已开机
        if recvBuf.find("car")!=-1:
            car_run_init()
            #行驶模块
            if recvBuf.find("up")!=-1:
                run(LOW_SPEED,LOW_SPEED)
                sleep(0.5)
            elif recvBuf.find("back")!=-1:
                back(LOW_SPEED,LOW_SPEED)
                sleep(0.5)
            elif recvBuf.find("left")!=-1:
                left(LOW_SPEED)
                sleep(0.5)
            elif recvBuf.find("right")!=-1:
                right(LOW_SPEED)
                sleep(0.5)
            elif recvBuf.find("spinLeft")!=-1:
                spin_left(SPIN_LOW_SPEED)
                sleep(0.5)
            elif recvBuf.find("spinRight")!=-1:
                spin_right(SPIN_LOW_SPEED)
                sleep(0.5)
            elif recvBuf.find("autodrive")!=-1:
                car_auto_drive()
            brake()
        #灯光模块
        elif recvBuf.find("turn_on_light")!=-1:
            led_init()
            print("LED init")
            light_flag=1
        elif recvBuf.find("turn_off_light")!=-1:
            led_clean()
            light_flag=0
        elif recvBuf.find("red")!=-1:
            if light_flag:
                LED_RED()
            else:
                print("LED未开，无法展示红色")
        elif recvBuf.find("green")!=-1:
            if light_flag:
                LED_GREEN()
            else:
                print("LED未开，无法展示绿色")
        elif recvBuf.find("blue")!=-1:
            if light_flag:
                LED_BLUE()
            else:
                print("LED未开，无法展示绿色")

        #小车关机
        elif recvBuf.find("poweroff")!=-1:
            power_flag=0
            brake()
            car_run_clean()
            led_clean()
        else:
            pass

    else:
        print("not a commond")

try:
    GPIO.setmode(GPIO.BCM)
    GPIO.setwarnings(False)
    #car_run_init()
    while 1:
        tcp_server_recv()
        tcp_server_parse(recvBuf)
        cmd=""
```

```
except:
    pass
```

其中用到的auto_drive模块：

```
# -*- coding:UTF-8 -*-
import RPi.GPIO as GPIO
from time import *

from car_run import *
from infrared_distance import *
#from infrared_light import *
from key import *
from led import *
from servo import *
from ultrasonic import *


def run_sides_space(run_speed, spin_time=0.2, spin_speed=SPIN_MID_SPEED):
    '''
    遇到障碍物,红外避障模块的指示灯亮,端口电平为LOW
    未遇到障碍物,红外避障模块的指示灯灭,端口电平为HIGH
    '''
    left_sensor = leftSensorValue_distance()
    right_sensor = rightSensorValue_distance()
    print("************", left_sensor)
    if left_sensor == GPIO.HIGH and right_sensor == GPIO.HIGH:
        print("左右都没障碍")  # 左右都没有
        run(run_speed, run_speed)
    elif left_sensor == GPIO.LOW and right_sensor == GPIO.HIGH:  # 左边有障碍
        print("左边有障碍")
        right(spin_speed)
        sleep(spin_time)
    elif left_sensor == GPIO.HIGH and right_sensor == GPIO.LOW:  # 右边有障碍
        print("右边有障碍")
        left(spin_speed)
        sleep(spin_time)
    else:
        print("左右都有障碍")  # 左右都有障碍
        spin_right(spin_speed)
        sleep(spin_time)


def runAndLight_according_distance():
    '''
    舵机旋转超声波测距避障,led根据车的状态显示相应的颜色并选择行驶模式
     '''
    # 品红色
    LED_RGB(GPIO.HIGH, GPIO.LOW, GPIO.HIGH)

    back(LOW_SPEED, LOW_SPEED)
    sleep(0.18)
    brake()

    # 舵机旋转到0度，即右侧，测距
    servo_appointed_detection(0)
    sleep(0.8)
    rightdistance = get_distance()

    # 舵机旋转到180度，即左侧，测距
    servo_appointed_detection(160)
    sleep(0.8)
    leftdistance = get_distance()

    # 舵机旋转到90度，即前方，测距
    servo_appointed_detection(75)
    sleep(0.8)
    frontdistance = get_distance()

    if leftdistance < DISTANCE_1 and rightdistance < DISTANCE_1 and frontdistance
< DISTANCE_1:
        # 亮红色，掉头
        LED_RGB(GPIO.HIGH, GPIO.LOW, GPIO.LOW)
        spin_right(SPIN_MID_SPEED)
        sleep(0.6)
    elif leftdistance >= rightdistance:
        # 亮蓝色
        LED_BLUE()
        spin_left(SPIN_MID_SPEED)
        sleep(0.15)
    elif leftdistance <= rightdistance:
        # 亮蓝色
        LED_BLUE()
        spin_right(SPIN_MAX_SPEED)
        sleep(0.15)


def car_auto_drive():
    on_off = 0
    GPIO.setmode(GPIO.BCM)
    GPIO.setwarnings(False)
    #car_run_init()
    infrared_distance_init()
    #infrared_light_init()
    key_init()
    led_init()
    servo_init()
    ultrasonic_init()


    # 按下开关开始启动，再按一下熄火
    def key_pressed_callback(pin):
        nonlocal on_off
        if not on_off:
            on_off = 1
            sleep(0.5)
        else:
            brake()
            #car_run_clean()
            servo_clean()
            #led_clean()
            exit(0)

    GPIO.add_event_detect(key, GPIO.RISING, key_pressed_callback, bouncetime=15)
    sleep(2)
    print("Press key2 to poweron!")
    while 1:
        if not on_off:
            brake()
            led_pause()
            servo_pause()
            while not on_off:
                pass

        else:
            distance = get_distance()
            if distance > DISTANCE_2:
                print("distance>50")
                run_sides_space(MID_SPEED)
                # run(MID_SPEED,MID_SPEED)
                LED_GREEN()
            elif distance > DISTANCE_1 and distance <= DISTANCE_2:
                print("30<distance<=50")
                LED_RGB(1, 0, 1)
                run_sides_space(LOW_SPEED)

            else:
                print("distance<30")
                runAndLight_according_distance()

    car_run_cleanup()
    servo_cleanup()
    led_cleanup()
```