

# 实验 5 自主移动机器人避障与运动控制

GROUP01小组成员

- 林永欣 SA18225225
- 刘皓冬 SA18225232
- 曹力月 SA18225018
- 夏军 SA18225406
- 徐涛 SA18225422

## 一.实验步骤

### <1> 树莓派的基本配置

#### 1.远程登陆

在windows打开终端cmd，然后输入 `ssh pi@192.168.0.1` ,密码是 123 (我们的树莓派的IP是192.168.0.1，密码已经自行设置为123)。

显示如下：（出现如下截图代码已经进入树莓派的linux系统）

```
pi@raspberrypi: ~  
Microsoft Windows [Version 10.0.17134.472]  
(c) 2018 Microsoft Corporation. All rights reserved.  
  
C:\Users\linyongxin>ssh pi@192.168.0.1  
pi@192.168.0.1's password:  
  
The programs included with the Debian GNU/Linux system are free software;  
the exact distribution terms for each program are described in the  
individual files in /usr/share/doc/*/copyright.  
  
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent  
permitted by applicable law.  
Last login: Sat Jan 20 19:03:36 2018 from 192.168.0.26  
pi@raspberrypi:~$
```

#### 2.配置树莓派模式

输入命令 `cd /etc/hostapd` ,进入hostapd文件夹。

```
pi@raspberrypi: /etc/hostapd  
pi@raspberrypi:/etc/hostapd $ cd /etc/hostapd/  
pi@raspberrypi:/etc/hostapd $ ls  
hostapd.conf  ifupdown.sh  
pi@raspberrypi:/etc/hostapd $
```

输入命令 `vim hostapd.conf` ,显示如下：

```
pi@raspberrypi: /etc/hostapd
interface=wlan0
driver=nl80211
ssid=111
hw_mode=g
channel=6
ieee80211n=1
wmm_enabled=1
ht_capab=[HT40][SHORT-GT-20][DSSS_CCK-40]
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_key_mgmt=WPA-PSK
wpa_passphrase=12345678
rsn_pairwise=CCMP
```

按下键盘的i键进入vim编辑器的插入模式，输入命令 `ssid = USTC_GROUP01` 和 `wpa_passphrase = 12345678` 修改账号密码，然后按ESC键进入操作模式，输入 `:wq!`，保存写文件修改并强制退出。

```
pi@raspberrypi: /etc/hostapd
interface=wlan0
driver=nl80211
ssid=USTC_Group01
hw_mode=g
channel=6
ieee80211n=1
wmm_enabled=1
ht_capab=[HT40][SHORT-GT-20][DSSS_CCK-40]
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_key_mgmt=WPA-PSK
wpa_passphrase=12345678
rsn_pairwise=CCMP
```

输入命令 `cd /etc`，进入etc文件夹，然后输入命令 `sudo vim rc.local` 进行编辑。

```
pi@raspberrypi: /etc
1 #!/bin/sh -e
2 #
3 # rc.local
4 #
5 # This script is executed at the end of each multiuser runlevel.
6 # Make sure that the script will "exit 0" on success or any other
7 # value on error.
8 #
9 # In order to enable or disable this script just change the execution
10 # bits.
11 #
12 # By default this script does nothing.
13 #
14 # Print the IP address
15 _IP=$(hostname -I) || true
16 if [ "$_IP" ]; then
17   printf "My IP address is %s\n" "$_IP"
18 fi
19
20 iptables-restore < /etc/iptables.ipv4.nat
21
22 cd /home/pi/mjpg-streamer-master/mjpg-streamer-experimental/
23 ./start.sh &
24
25 # cd /etc/hostapd/
26 # ./host.conf &
27
28
rc.local
"rc.local" 36L, 643C
```

在rc.local文件中加入代码 `cd /etc/hostapd` 和 `./host.conf &`，&代表在后台运行的意思，然后保存退出即可。

```
pi@raspberrypi: /etc
4 #
5 # This script is executed at the end of each multiuser runlevel.
6 # Make sure that the script will "exit 0" on success or any other
7 # value on error.
8 #
9 # In order to enable or disable this script just change the execution
10 # bits.
11 #
12 # By default this script does nothing.
13 #
14 # Print the IP address
15 _IP=$(hostname -I) || true
16 if [ "$_IP" ]; then
17     printf "My IP address is %s\n" "$_IP"
18 fi
19
20 iptables-restore < /etc/iptables.ipv4.nat
21
22 cd /home/pi/mjpg-streamer-master/mjpg-streamer-experimental/
23 ./start.sh &
24
25 cd /etc/hostapd/
26 ./host.conf &
27
28
29
30 #cd /home/pi/SmartCar/
31 #./TCP_control
rc.local [?] 26, 1 37%
-- INSERT --
```

最终重启系统即可。

## <2> GPIO编程练习

### 1.完整代码

```
# -*- coding:UTF-8 -*-

import RPi.GPIO as GPIO
import time

#RGB三色灯引脚定义
LED_R = 22
LED_G = 27
LED_B = 24

#设置RGB三色灯为BCM编码方式
GPIO.setmode(GPIO.BCM)

#RGB三色灯设置为输出模式
GPIO.setup(LED_R, GPIO.OUT)
GPIO.setup(LED_G, GPIO.OUT)
GPIO.setup(LED_B, GPIO.OUT)

#命令行控制灯亮灭
try:
    while True:
        status = input('请输入灯的状态: ')
        print(status)
        if status == 1:
            GPIO.output(LED_R, GPIO.HIGH)
            GPIO.output(LED_G, GPIO.LOW)
            GPIO.output(LED_B, GPIO.LOW)
```

```

        print('ok')
    elif status == 0:
        GPIO.output(LED_R,GPIO.LOW)
        GPIO.output(LED_G,GPIO.LOW)
        GPIO.output(LED_B,GPIO.LOW)
        print('off')
except:
    print "except"
    #使用try except语句，当CTRL+C结束进程时会触发异常后
    #会执行gpio.cleanup()语句清除GPIO管脚的状态
    GPIO.cleanup()

```

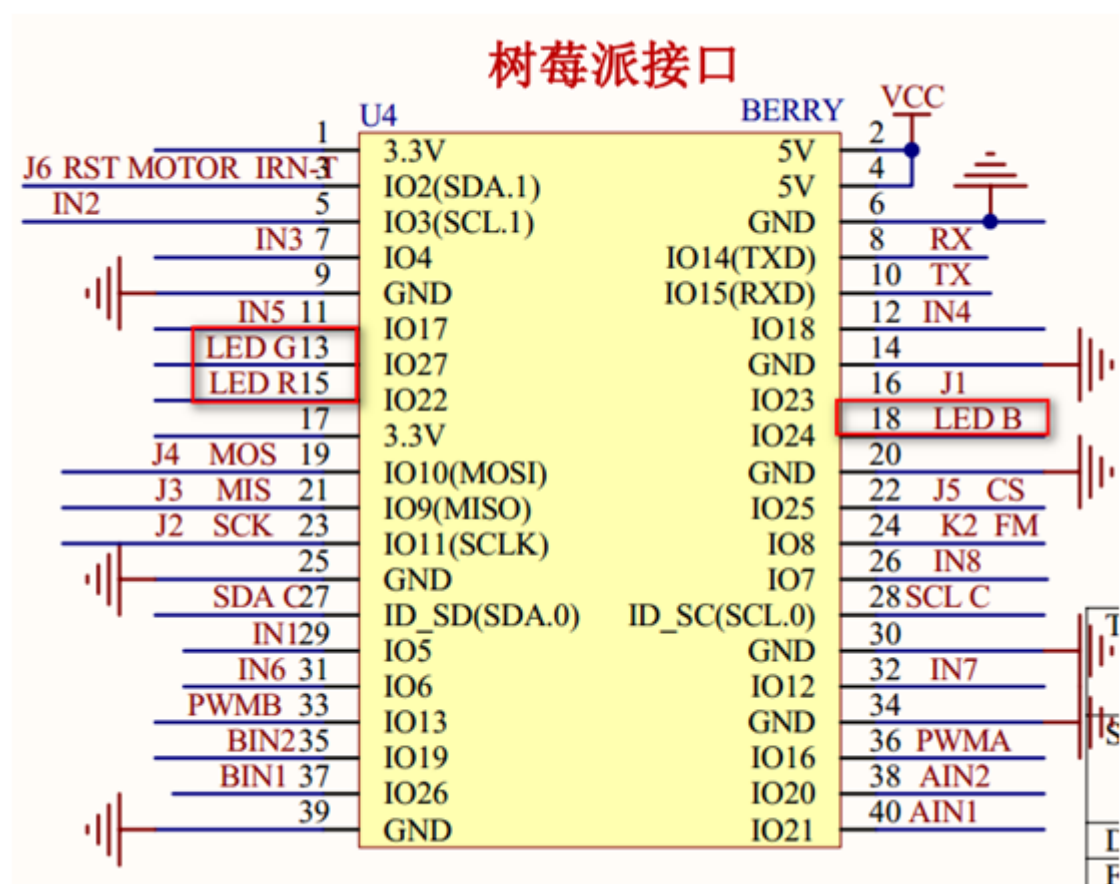
## 2.代码解读

#RGB三色灯引脚定义

```

LED_R = 22
LED_G = 27
LED_B = 24

```



wiringPi编码引脚		树莓派 40Pin 引脚对照表				树莓派物理引脚	
wiringPi 编码	BCM 编码	功能名	物理引脚 BOARD编码		功能名	BCM 编码	wiringPi 编码
		3.3V	1	2	5V		
8	2	SDA.1	3	4	5V		
9	3	SCL.1	5	6	GND		
7	4	GPIO.7	7	8	TXD	14	15
		GND	9	10	RXD	15	16
0	17	GPIO.0	11	12	GPIO.1	18	1
2	27	GPIO.2	13	14	GND		
3	22	GPIO.3	15	16	GPIO.4	23	4
		3.3V	17	18	GPIO.5	24	5
12	10	MOSI	19	20	GND		
13	9	MISO	21	22	GPIO.6	25	6
14	11	SCLK	23	24	CE0	8	10
		GND	25	26	CE1	7	11
30	0	SDA.0	27	28	SCL.0	1	31
21	5	GPIO.21	29	30	GND		
22	6	GPIO.22	31	32	GPIO.26	12	26
23	13	GPIO.23	33	34	GND		
24	19	GPIO.24	35	36	GPIO.27	16	27
25	26	GPIO.25	37	38	GPIO.28	20	28
		GND	39	40	GPIO.29	21	29

由上面两张图可以知道相应的管脚，LED\_R连接到主控板上的物理引脚15口对应的BCM编码的引脚为22，所以 `LED_R = 22`，同理LED\_G和LED\_B分别接在主控板上的物理引脚为13和18口，对应于BCM编码的引脚为27,24，所以程序为 `LED_G = 27` 和 `LED_B = 24`

```
#设置RGB三色灯为BCM编码方式
GPIO.setmode(GPIO.BCM)
```

这行代码时将GPIO的编码方式设置为BCM编码的方式。

在树莓派中有三种GPIO编码方式，分别是BOARD、BCM和wiringPi。

python版本中一般用BCM编码的方式，C语言版本中用wiringPi编码方式。

参考链接：

树莓派GPIO的三种方式：<https://blog.csdn.net/hu7850/article/details/51785594>

```
#RGB三色灯设置为输出模式
GPIO.setup(LED_R, GPIO.OUT)
GPIO.setup(LED_G, GPIO.OUT)
GPIO.setup(LED_B, GPIO.OUT)
```

将LED\_R、LED\_G和LED\_B分别设置为输出模式，从电路图中我们可以知道三个引脚都是共阴接地，所以输出高电平时等亮，输出低电平时等灭。

```

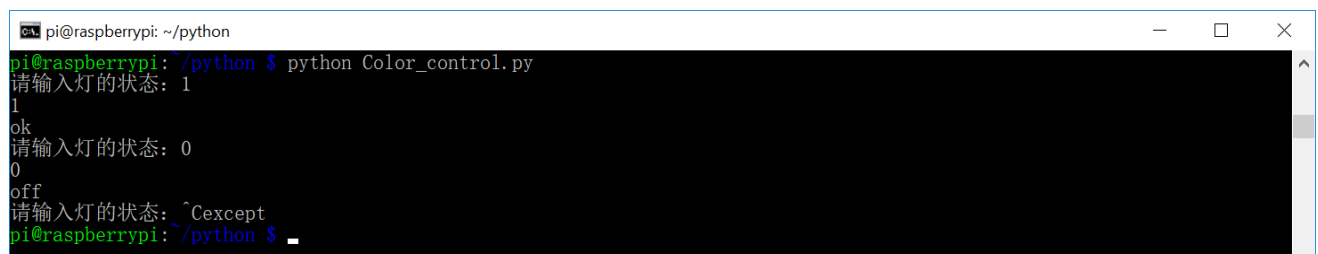
while True:
    status = input('请输入灯的状态: ')
    print(status)
    if status == 1:
        GPIO.output(LED_R, GPIO.HIGH)
        GPIO.output(LED_G, GPIO.LOW)
        GPIO.output(LED_B, GPIO.LOW)
        print('ok')
    elif status == 0:
        GPIO.output(LED_R, GPIO.LOW)
        GPIO.output(LED_G, GPIO.LOW)
        GPIO.output(LED_B, GPIO.LOW)
        print('off')

```

控制灯亮灭逻辑实现。

通过if ... elif ...语句来实现控制灯亮灭。当输入1时打开灯，当输入0时关掉灯。在这里只显示红灯，其他颜色关闭。

### 3.运行结果



```

pi@raspberrypi: ~/python
pi@raspberrypi: ~/python $ python Color_control.py
请输入灯的状态: 1
1
ok
请输入灯的状态: 0
0
off
请输入灯的状态: ^Cexcept
pi@raspberrypi: ~/python $

```

输入1可以打开红灯，输入0可以关闭灯，按 **Ctrl+c** 可以退出程序。

## <3> 电机驱动控制

### 1.完整代码

```

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

#小车电机引脚定义
IN1 = 20
IN2 = 21
IN3 = 19
IN4 = 26
ENA = 16
ENB = 13

#设置GPIO口为BCM编码方式
GPIO.setmode(GPIO.BCM)

#忽略警告信息
GPIO.setwarnings(False)

```

#电机引脚初始化操作

```
def motor_init():  
    global pwm_ENA  
    global pwm_ENB  
    global delaytime  
    GPIO.setup(ENA,GPIO.OUT,initial=GPIO.HIGH)  
    GPIO.setup(IN1,GPIO.OUT,initial=GPIO.LOW)  
    GPIO.setup(IN2,GPIO.OUT,initial=GPIO.LOW)  
    GPIO.setup(ENB,GPIO.OUT,initial=GPIO.HIGH)  
    GPIO.setup(IN3,GPIO.OUT,initial=GPIO.LOW)  
    GPIO.setup(IN4,GPIO.OUT,initial=GPIO.LOW)  
    #设置pwm引脚和频率为2000hz  
    pwm_ENA = GPIO.PWM(ENA, 2000)  
    pwm_ENB = GPIO.PWM(ENB, 2000)  
    pwm_ENA.start(0)  
    pwm_ENB.start(0)
```

#小车前进

```
def run(delaytime):  
    GPIO.output(IN1, GPIO.HIGH)  
    GPIO.output(IN2, GPIO.LOW)  
    GPIO.output(IN3, GPIO.HIGH)  
    GPIO.output(IN4, GPIO.LOW)  
    pwm_ENA.ChangeDutyCycle(80)  
    pwm_ENB.ChangeDutyCycle(80)  
    time.sleep(delaytime)
```

#小车后退

```
def back(delaytime):  
    GPIO.output(IN1, GPIO.LOW)  
    GPIO.output(IN2, GPIO.HIGH)  
    GPIO.output(IN3, GPIO.LOW)  
    GPIO.output(IN4, GPIO.HIGH)  
    pwm_ENA.ChangeDutyCycle(80)  
    pwm_ENB.ChangeDutyCycle(80)  
    time.sleep(delaytime)
```

#小车左转

```
def left(delaytime):  
    GPIO.output(IN1, GPIO.LOW)  
    GPIO.output(IN2, GPIO.LOW)  
    GPIO.output(IN3, GPIO.HIGH)  
    GPIO.output(IN4, GPIO.LOW)  
    pwm_ENA.ChangeDutyCycle(80)  
    pwm_ENB.ChangeDutyCycle(80)  
    time.sleep(delaytime)
```

#小车右转

```
def right(delaytime):  
    GPIO.output(IN1, GPIO.HIGH)  
    GPIO.output(IN2, GPIO.LOW)  
    GPIO.output(IN3, GPIO.LOW)  
    GPIO.output(IN4, GPIO.LOW)
```

```
pwm_ENA.ChangeDutyCycle(80)
pwm_ENB.ChangeDutyCycle(80)
time.sleep(delaytime)
```

#小车原地左转

```
def spin_left(delaytime):
    GPIO.output(IN1, GPIO.LOW)
    GPIO.output(IN2, GPIO.HIGH)
    GPIO.output(IN3, GPIO.HIGH)
    GPIO.output(IN4, GPIO.LOW)
    pwm_ENA.ChangeDutyCycle(80)
    pwm_ENB.ChangeDutyCycle(80)
    time.sleep(delaytime)
```

#小车原地右转

```
def spin_right(delaytime):
    GPIO.output(IN1, GPIO.HIGH)
    GPIO.output(IN2, GPIO.LOW)
    GPIO.output(IN3, GPIO.LOW)
    GPIO.output(IN4, GPIO.HIGH)
    pwm_ENA.ChangeDutyCycle(80)
    pwm_ENB.ChangeDutyCycle(80)
    time.sleep(delaytime)
```

#小车停止

```
def brake(delaytime):
    GPIO.output(IN1, GPIO.LOW)
    GPIO.output(IN2, GPIO.LOW)
    GPIO.output(IN3, GPIO.LOW)
    GPIO.output(IN4, GPIO.LOW)
    pwm_ENA.ChangeDutyCycle(80)
    pwm_ENB.ChangeDutyCycle(80)
    time.sleep(delaytime)
```

#延时2s

```
time.sleep(2)
```

#try/except语句用来检测try语句块中的错误，

#从而让except语句捕获异常信息并处理。

```
try:
    motor_init()
    while True:
        print("0代表前进\n 1代表后退\n 2代表左转\n 3代表右转\n
              4代表原地左转\n 5代表原地右转\n 6代表停止\n")
        status = input("请输入控制模式: ")
        if status == 0:
            run(2)
        elif status == 1:
            back(2)
        elif status == 2:
            left(2)
        elif status == 3:
            right(2)
```



```

elif status == 4:
    spin_left(2)
elif status == 5:
    spin_right(2)
elif status == 6:
    brake(1)
except KeyboardInterrupt:
    pass
pwm_ENA.stop()
pwm_ENB.stop()
GPIO.cleanup()

```

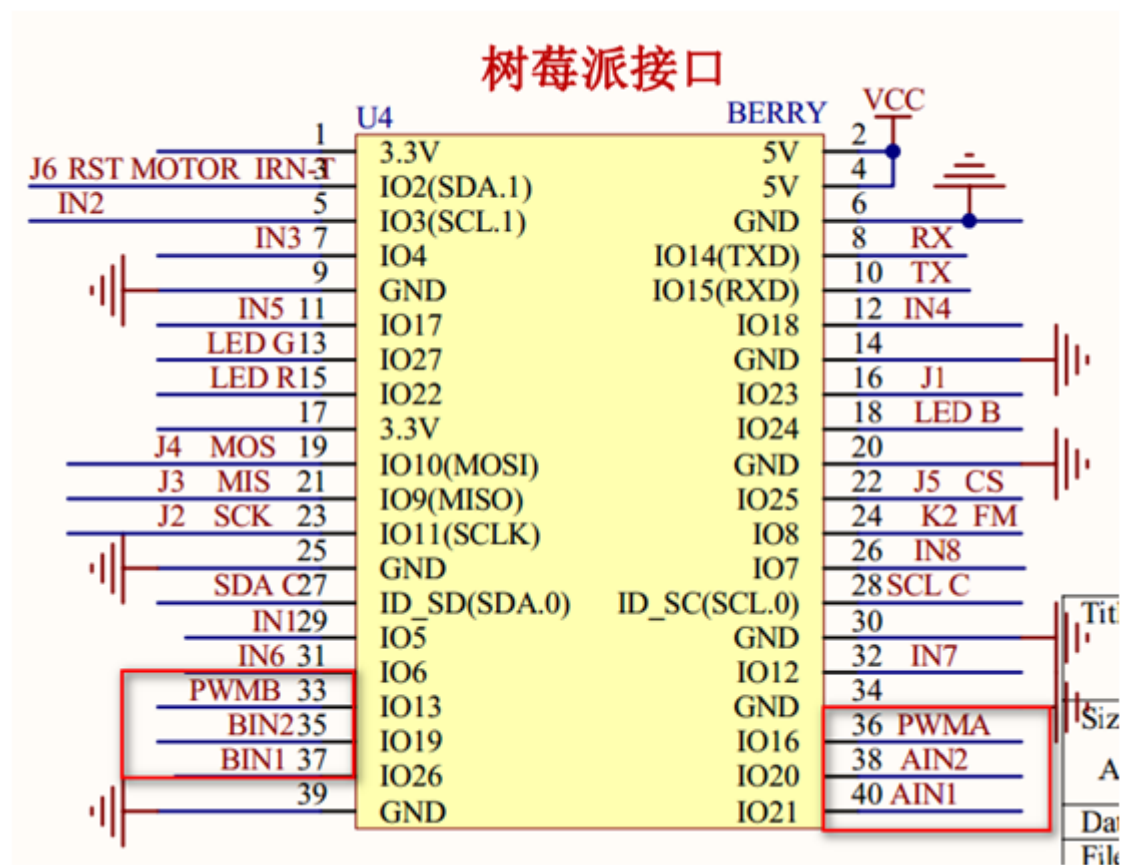
## 2.代码解读

#小车电机引脚定义

```

IN1 = 20
IN2 = 21
IN3 = 19
IN4 = 26
ENA = 16
ENB = 13

```



wiringPi编码引脚		树莓派 40Pin 引脚对照表				树莓派物理引脚	
wiringPi 编码	BCM 编码	功能名	物理引脚 BOARD 编码		功能名	BCM 编码	wiringPi 编码
		3.3V	1	2	5V		
8	2	SDA.1	3	4	5V		
9	3	SCL.1	5	6	GND		
7	4	GPIO.7	7	8	TXD	14	15
		GND	9	10	RXD	15	16
0	17	GPIO.0	11	12	GPIO.1	18	1
2	27	GPIO.2	13	14	GND		
3	22	GPIO.3	15	16	GPIO.4	23	4
		3.3V	17	18	GPIO.5	24	5
12	10	MOSI	19	20	GND		
13	9	MISO	21	22	GPIO.6	25	6
14	11	SCLK	23	24	CE0	8	10
		GND	25	26	CE1	7	11
30	0	SDA.0	27	28	SCL.0	1	31
21	5	GPIO.21	29	30	GND		
22	6	GPIO.22	31	32	GPIO.26	12	26
23	13	GPIO.23	33	34	GND		
24	19	GPIO.24	35	36	GPIO.27	16	27
25	26	GPIO.25	37	38	GPIO.28	20	28
		GND	39	40	GPIO.29	21	29

由电路原理图可知AIN1,AIN2,PWMA,BIN1,BIN2,PWMB分别接在树莓派主控板上的40,38,36,37,35,33物理引脚上,对应的BCM编码是21,20,16,26,19,13。

引脚对应我们的代码如下：

IN1 = 20 # IN1对应AIN2

IN2 = 21 # IN2对应AIN1

IN3 = 19 # IN3对应BIN2

IN4 = 26 # IN4对应BIN1

ENA = 16 # ENA对应PWMA

ENB = 13 # ENB对应PWMB

```
#电机引脚初始化操作
def motor_init():
    global pwm_ENA
    global pwm_ENB
    global delaytime
    GPIO.setup(ENA,GPIO.OUT,initial=GPIO.HIGH)
    GPIO.setup(IN1,GPIO.OUT,initial=GPIO.LOW)
    GPIO.setup(IN2,GPIO.OUT,initial=GPIO.LOW)
    GPIO.setup(ENB,GPIO.OUT,initial=GPIO.HIGH)
    GPIO.setup(IN3,GPIO.OUT,initial=GPIO.LOW)
    GPIO.setup(IN4,GPIO.OUT,initial=GPIO.LOW)
    #设置pwm引脚和频率为2000hz
    pwm_ENA = GPIO.PWM(ENA, 2000)
    pwm_ENB = GPIO.PWM(ENB, 2000)
    pwm_ENA.start(0)
    pwm_ENB.start(0)
```

初始化代码中，将ENA和ENB两个引脚设置成高电平，使能。四个电机的引脚设置成低电平，让轮子静止。之后设置两个enable引脚的PWM频率为2000Hz，启动两个PWM，初始化占空比为0，也就是说，都是电压为0。

之后设置小车的动作，分别为前进，后退，左转，右转，原地左转，原地右转，停止，共七种状态。

```
#小车前进
def run(delaytime):
    GPIO.output(IN1, GPIO.HIGH)
    GPIO.output(IN2, GPIO.LOW)
    GPIO.output(IN3, GPIO.HIGH)
    GPIO.output(IN4, GPIO.LOW)
    pwm_ENA.ChangeDutyCycle(80)
    pwm_ENB.ChangeDutyCycle(80)
    time.sleep(delaytime)
```

代码中，设置小车运行的时间，设置占空比为80，比值越大，速度越快。前进的时候，分别设置四个轮子是前进还是后退，每两个管脚控制两个轮子，同在一侧的两个轮子的控制是同步的。IN1和IN2控制左轮，IN3和IN4控制右轮。前进的时候，两边设置成前高后低，轮子就是向前的，设置成前低后高就是向后，在向左转的时候，左边的轮子不动，都设置成低，后边轮子设置成向前，就是前高后低。具体状态见下表。

管脚	IN1	IN2	左轮胎状态
电位	high	Low	forward
电位	low	high	back
管脚	IN3	IN4	右轮胎状态
电位	high	Low	forward
电位	low	high	back

左转和右转的操作就是通过控制左右轮胎的状态决定的。

```
#延时2s
time.sleep(2)

#try/except语句用来检测try语句块中的错误，
#从而让except语句捕获异常信息并处理。
try:
    motor_init()
    while True:
        print("0代表前进\n 1代表后退\n 2代表左转\n 3代表右转\n\n 4代表原地左转\n 5代表原地右转\n 6代表停止\n")
        status = input("请输入控制模式: ")
        if status == 0:
            run(2)
        elif status == 1:
            back(2)
        elif status == 2:
            left(2)
```

```

        elif status == 3:
            right(2)
        elif status == 4:
            spin_left(2)
        elif status == 5:
            spin_right(2)
        elif status == 6:
            brake(1)
    except KeyboardInterrupt:
        pass
    pwm_ENA.stop()
    pwm_ENB.stop()
    GPIO.cleanup()

```

主函数中,根据你的输入来决定小车的运动状态, 用if-else循环嵌套实现。在程序结束后, 将PWM设置为停止。在整个程序的运行过程中, 通过PWM占空比来设置驱动小车轮子的电机的电压, 从而改变小车的行进速度。

小车实现左右转的机制: 左转是左轮不动, 右轮向前, 这里可以使用两边轮子的速度之差去调整小车的转向半径和转向角度。

## <4> 红外光电传感器应用

### 1.初始化代码分析

```

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

#小车电机引脚定义
IN1 = 20
IN2 = 21
IN3 = 19
IN4 = 26
ENA = 16
ENB = 13

#小车按键定义
key = 8

#红外避障引脚定义
AvoidSensorLeft = 12
AvoidSensorRight = 17

#设置GPIO口为BCM编码方式
GPIO.setmode(GPIO.BCM)

#忽略警告信息
GPIO.setwarnings(False)

#电机引脚初始化为输出模式
#按键引脚初始化为输入模式
#红外避障引脚初始化为输入模式

```

```
def init():
    global pwm_ENA
    global pwm_ENB
    GPIO.setup(ENA,GPIO.OUT,initial=GPIO.HIGH)
    GPIO.setup(IN1,GPIO.OUT,initial=GPIO.LOW)
    GPIO.setup(IN2,GPIO.OUT,initial=GPIO.LOW)
    GPIO.setup(ENB,GPIO.OUT,initial=GPIO.HIGH)
    GPIO.setup(IN3,GPIO.OUT,initial=GPIO.LOW)
    GPIO.setup(IN4,GPIO.OUT,initial=GPIO.LOW)
    GPIO.setup(key,GPIO.IN)
    GPIO.setup(AvoidSensorLeft,GPIO.IN)
    GPIO.setup(AvoidSensorRight,GPIO.IN)
    #设置pwm引脚和频率为2000hz
    pwm_ENA = GPIO.PWM(ENA, 2000)
    pwm_ENB = GPIO.PWM(ENB, 2000)
    pwm_ENA.start(0)
    pwm_ENB.start(0)
```

在这个实验中，用到了红外感应器，两个红外感应器的引脚的BCM编码分别是12,17（左，右），对应树莓派的32,11引脚，当红外感应器检测到有物体的时候，感应器上的蓝灯会亮，引脚会返回低电平，程序可以通过 `AvoidSensorLeft（right）` 的值判断是否有物体。在初始化代码中，将pwm设置为2000Hz。

## <5> 舵机和超声波传感器的应用

### 1.工作原理

#### 1.向量场直方图

实验中，控制舵机在0度到180度的范围，每次转动15度，这样，当舵机转动到0、15、30、45等角度的时候，可以用超声波模块测量小车与当前角度正前方障碍物的距离，然后用一个长度为13的List依次保留这些距离，然后求出这些距离的最大值，找到这个最大值所在的索引，乘以15，就是小车前进方向所在的角度。

以0、15、30等为横坐标，距离为纵坐标，就构成了向量场直方图。

这里采用List这样一个数据结构，理由如下：

在一个for循环中，可以一边通过比较记录下当前的最大值，一边记录下当前的最大值的索引，而这个索引乘以15就是我们要求的角度，这样操作起来非常的方便，而Python里面的其他数据结构，如Dictionary、Tuple都不具备这个优点。

以15度为一个间距，是因为间距不能太小，否则测完180度方向内的距离耗时太多，而太大会导致结果不够精确。

#### 2.人工势场法

人工势场法是局部路径规划的一种比较常用的方法。这种方法假设机器人在一种虚拟力场下运动。

工作原理是：将机器人在环境中的运动视为一种机器人在虚拟的人工受力场的运动。障碍物对机器人产生斥力，目标点对机器人产生引力，引力和斥力的合力作为机器人的加速力，来控制机器人的运动方向和计算机器人的位置。

**引力场（attraction）**：随机器人与目标点的距离增加而单调递增，且方向指向目标点；

**斥力场（repulsion）**：在机器人处在障碍物位置时有一极大值，并随机器人与障碍物距离的增大而单调减小，方向指向远离障碍物方向。



图 2.1 引力势场

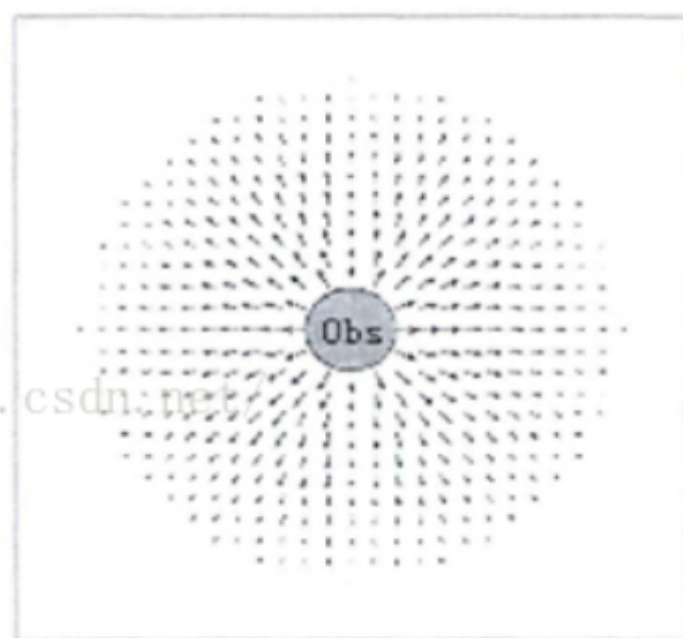


图 2.2 斥力势场



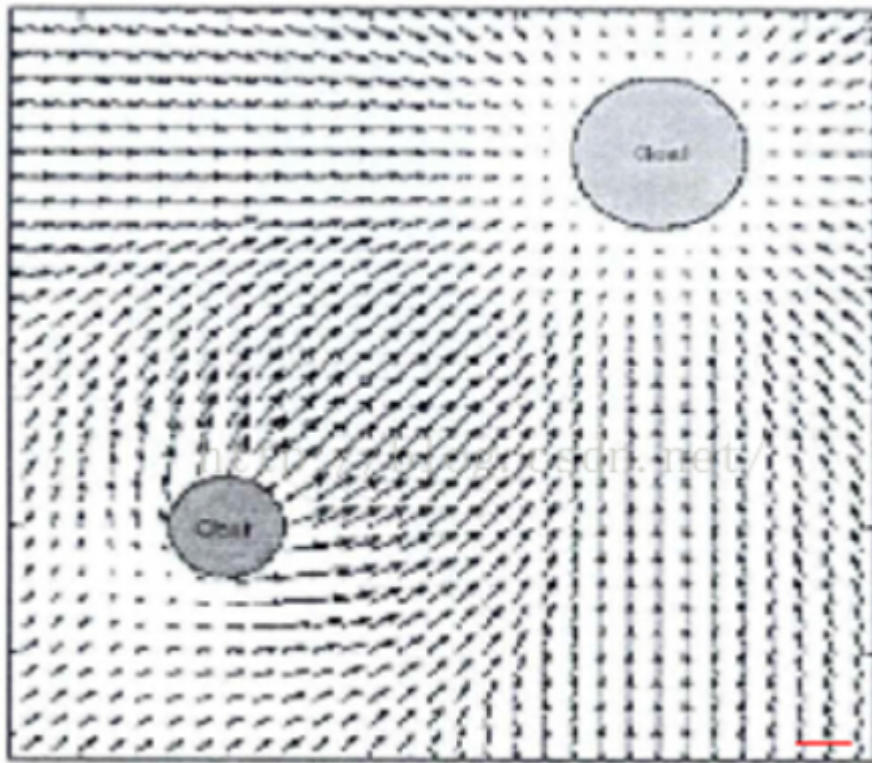


图 2.3 引力和斥力的合力

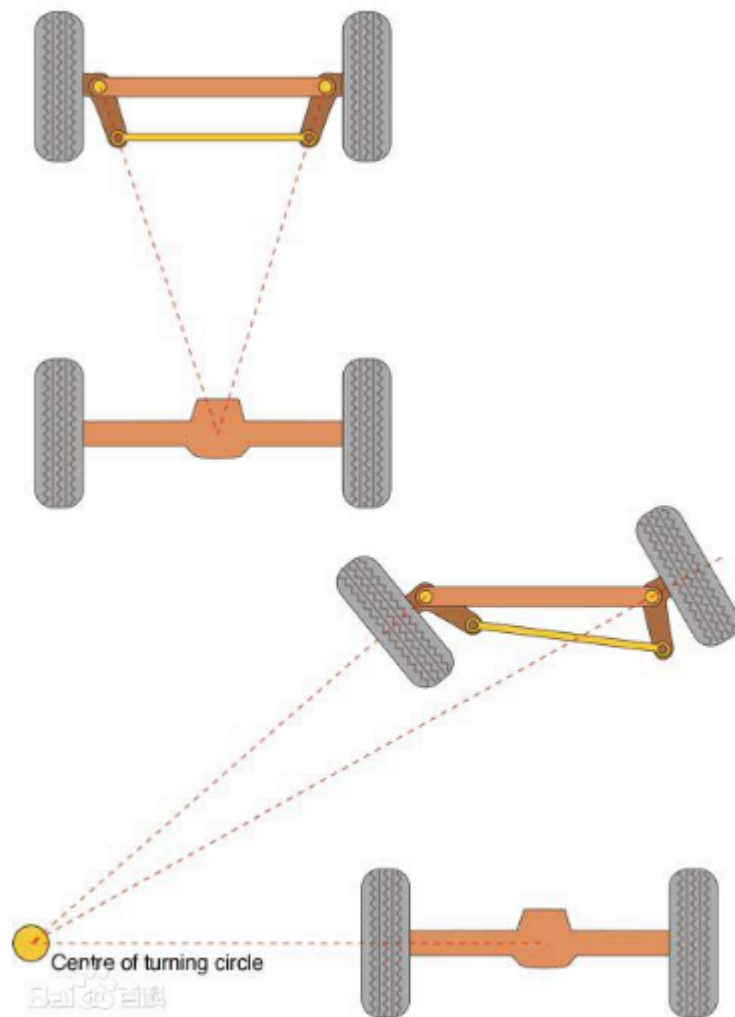
存在的缺点：

- 存在陷阱区域
- 在相近的障碍物群中不能识别路径
- 在障碍物前震荡
- 在狭窄通道中摆动
- 障碍物附近目标不可达

### 3. 小车转向操作

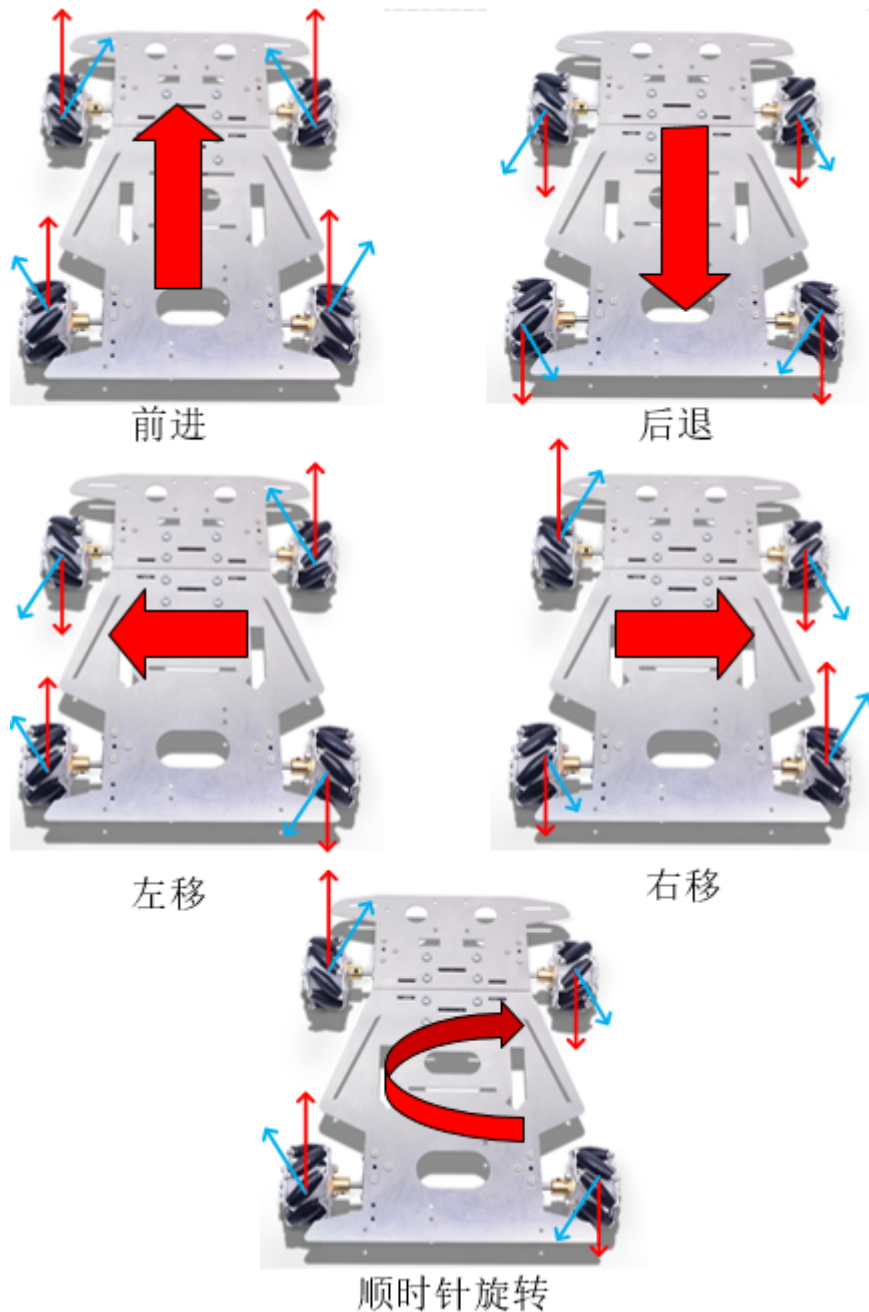
车的转向目前主要分为以下几种。

**阿克曼方式**，这个是当前使用最多的方式，四轮汽车都采用这种方式，通过改变前轮的方向，可以改变车子的转向半径，同时前面左右两个轮子的转向角度不一样，通常会相差 $2^\circ$ 到 $4^\circ$ ，使四个轮子路径的圆心大致交汇于后轴的延长上瞬间转向中心。



还有一种是瑞典轮，这种轮子通过在轮子上加装一些小的滚动轮，可以让下车沿任意方向运动，只需要改变各个轮子的转动方向，另外对角线上的轮子上的小滚轮的方向是一致的。这样四个轮子不同的转动方式可以组合成各种运动方式。如下图左上角第一个所示，前轮向前滚动，左右两边还会有分别向右和向左的力，这两个力的水平分量相互抵消，合成一个向前的力，后轮也是同样如此，这时小车即可沿直线向前运动。





另外一个实验中用到的\***差速四轮驱动**，通过改变四个轮子的速度差来控制小车的运动路径，在实验中，我们通过超声波平台使用直方图法找到了可行驶的一个方向，这个方向和小车的前进方向有一个夹角，通过这个夹角的值，我们可以计算出四个轮子的速度配比，使得小车在这样的控制下小车的转向半径，另外，实验用到的小车的转向角度还有一个时间的控制变量，时间的长短决定了小车转向轨迹的长短。我们这里将时间固定，通过改变两边轮子的速度差来控制小车转向的转弯半径，速度差越大，转向半径越小。也就是说，转弯半径越小相同时间下转向的角度更大。如果我们已知希望转向的方向和小车前进方向的夹角大小，就可以控制小车向预定方向转弯。同时，由于小车和前方障碍物之间的空间是30 cm，所以为了使小车避免和前方障碍物接触，要控制小车的轮速不能过大，如果过大，小车会直接撞向障碍物。也不能过小，小车无法在规定时间内转向对应角度。通过测试，粗略的估算了小车轮速的范围是[35, 60],也就是说，在这个范围内通过转向角度来确定轮速差。就可以转出一个预期角度的近似值。

## 2.代码分析

```
#超声波函数
def Distance_test():
```

```

GPIO.output(TrigPin,GPIO.HIGH)
time.sleep(0.000015)
GPIO.output(TrigPin,GPIO.LOW)
while not GPIO.input(EchoPin):
    pass
t1 = time.time()
while GPIO.input(EchoPin):
    pass
t2 = time.time()
print "distance is %d " % (((t2 - t1)* 340 / 2) * 100)
time.sleep(0.01)
return ((t2 - t1)* 340 / 2) * 100

```

#舵机旋转到指定角度

```

def servo_appointed_detection(pos):
    for i in range(18):
        pwm_servo.ChangeDutyCycle(2.5 + 10 * pos/180)

```

#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

```

def servo_color_carstate():

```

#开红灯

```

GPIO.output(LED_R, GPIO.HIGH)
GPIO.output(LED_G, GPIO.LOW)
GPIO.output(LED_B, GPIO.LOW)
back(20, 20)
time.sleep(0.08)
brake()

```

# 舵机从0度到180度扫描，建立直方图

```

disList = []
maxdistance = 0
for i in range(13):
    servo_appointed_detection(15 * i)
    time.sleep(0.1)
    value = Distance_test()
    if value > 0 and value < 200:
        disList.append(value) # 只有测量的距离在(0,200)的范围内，才加入列表中
    else:
        disList.append(0) # 为了之后可以通过索引来计算角度，不符合条件的角度用0来占位
    if value > maxdistance:
        maxdistance = value
        maxangle = i * 15
maxaverage = 0

```

```

print('maxdistance:',maxdistance)

```

```

print('maxangle:',maxangle)

```

```

time.sleep(0.08)

```

```

servo_appointed_detection(90)

```

```

time.sleep(0.5)

```

```

"""

```

轮子的速度控制在35~60这个范围内

```

"""

```

```

if maxdistance > 30:

```

```

    if maxangle > 0 and maxangle < 90: # 右边没有障碍物

```

```

        turn_ratio = (90-maxangle)/90 # 计算小车预计转向角度
        right(35+turn_ratio*(60-35),35) # 控制小车转向希望的角度
        time.sleep(0.2)
    elif maxangle == 90: # 前面没有障碍物
        run(30,30)
    elif maxangle > 90 : # 左边没有障碍物
        turn_ratio = -(90-maxangle)/90
        left(35,35+turn_ratio*(60-35))
        time.sleep(0.2)
elif maxdistance <= 30:
    back(30,30)
    time.sleep(0.1)

```

代码中用转向角度的占90的比率作为小车两边轮子的速度差。

disList列表用来存放超声波测量得出的距离，通过一个if语句过滤一些非法值，用else来填充零值。让滞后利用索引决定角度的工作可以顺利完成。