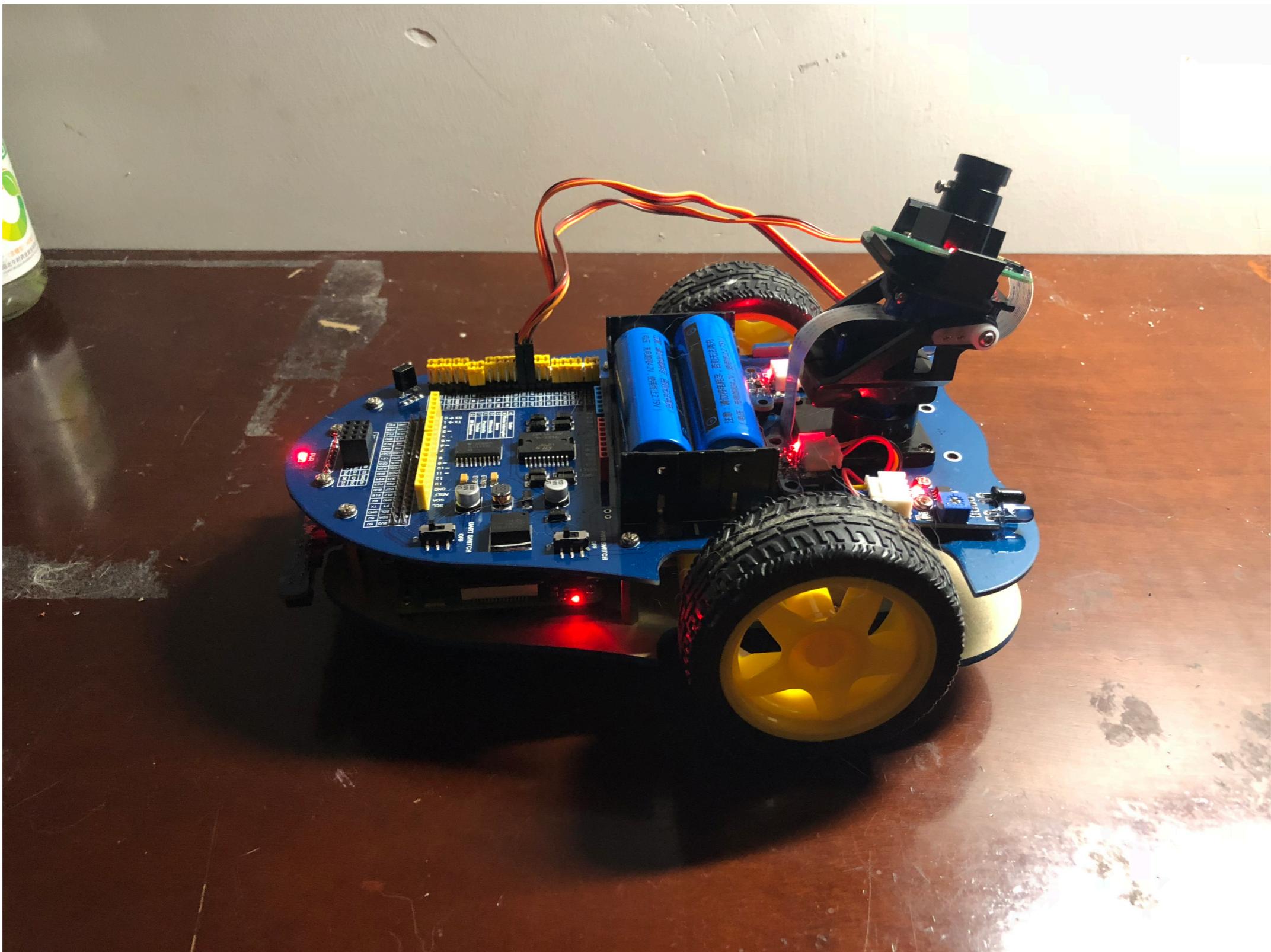


# **避障、跟踪、循迹监控小车**

**物联网1502 叶玲见**



小车整体图

# 完成进度

- 完成：避障、跟踪、循迹、监控模块
- 未完成：SLAM模块

# 实验环境

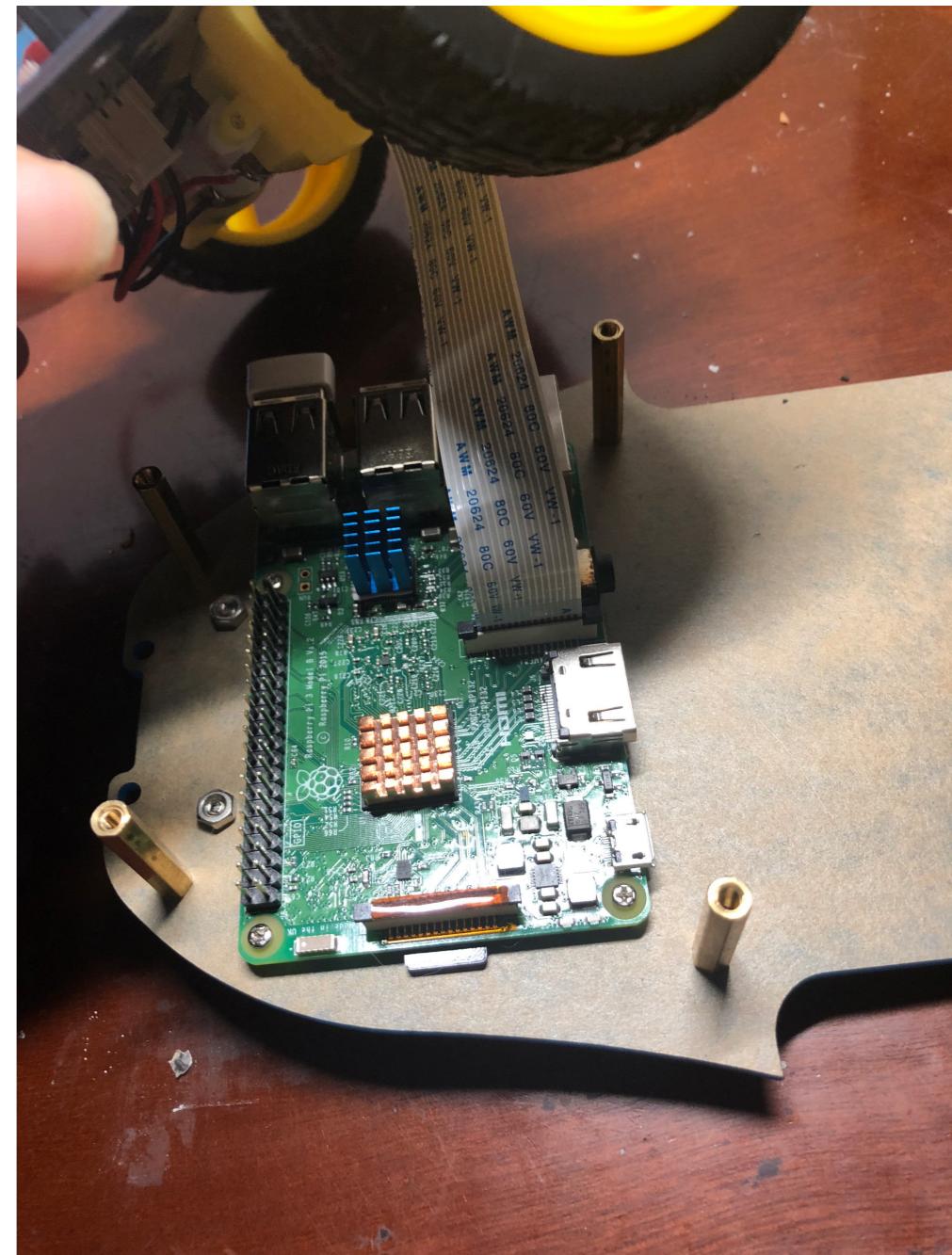
- 主控板: RASPBIAN 4.14
- 辅助库或软件: samba, python2.7, RPI.GPIO,vim, MJPG-streamer
- 上位机: Windows10 + Xshell6 + putty

# 小车组成

- 主控板：树莓派3B
- 基板：AlphaBot
- 使用模块：电机驱动模块，红外线避障模块，红外线跟踪模块，五路红外避障模块、测速模块，树莓派摄像头模块，供电模块

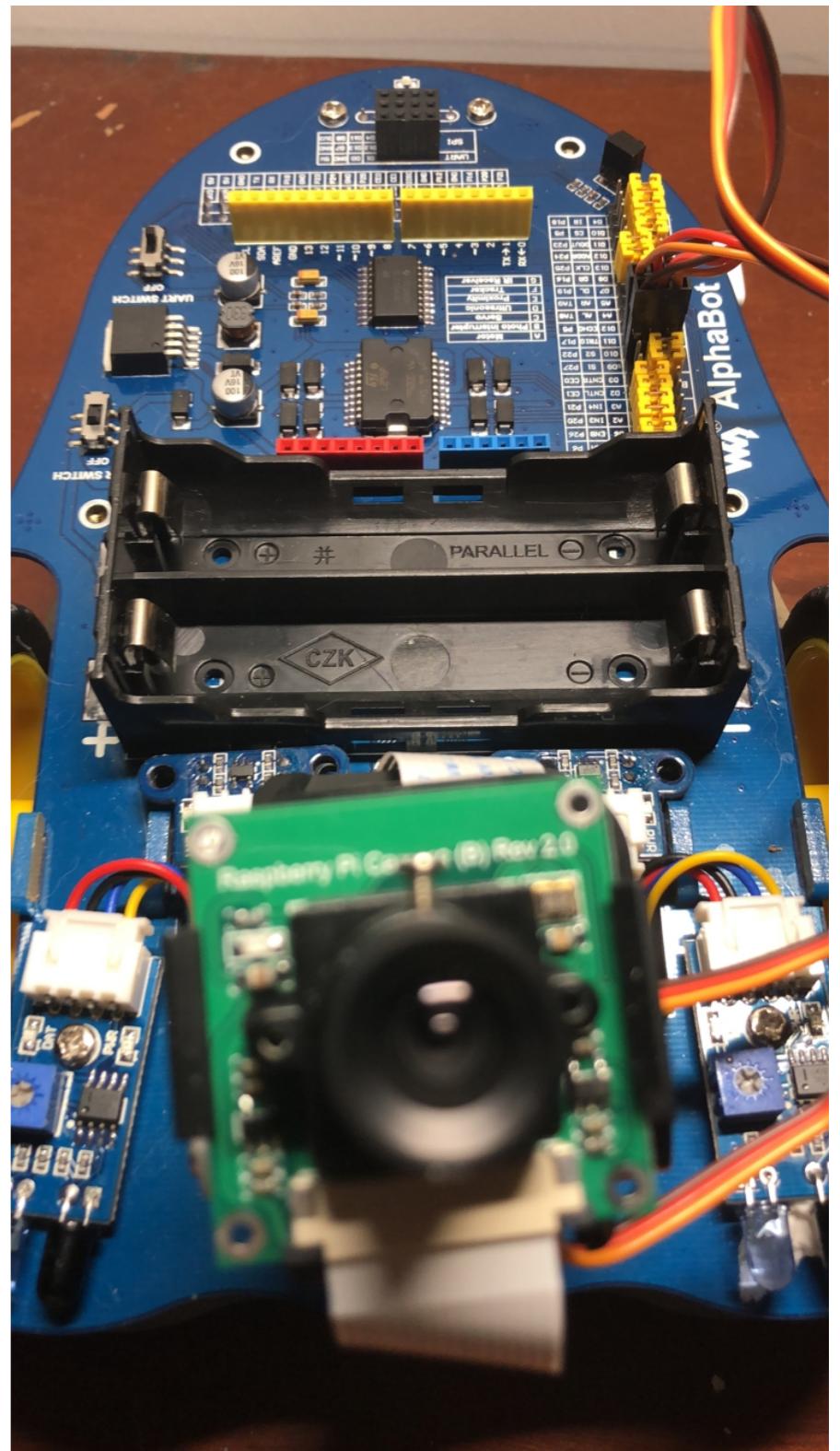
# 树莓派

- SD卡中存储RASPBIAN系统
- CSI接口连接树莓派摄像头模块
- 在系统中主要安装python2.7依赖库，用于支持主控程序的运行，以及mjpg-stream使我们可以通过web观察摄像头获得的图像信息
- 红外线传感器模块、测速模块以及巡线模块（没完成）通过GPIO口接入



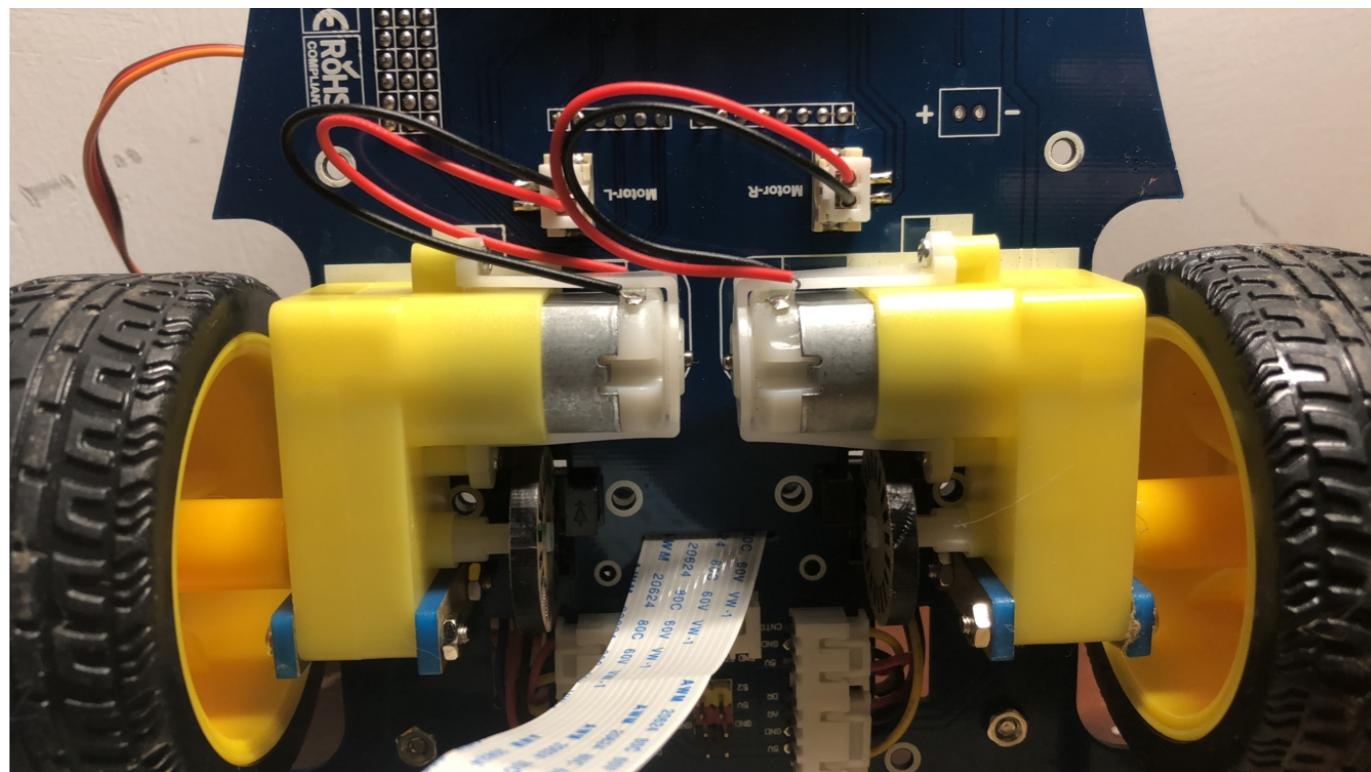
# AlphaBot

- 基板连接主控板的全部GPIO口
- 红外线传感器模块、测速模块以及五路红外循迹模块直接与基板连接，通过基板针脚连接主控板的GPIO口
- 直流电机直接连接基板供电
- 摄像头的舵机控制没有完成



# 电机驱动模块

- PIN12、PIN13接小车左电机，  
PIN20、PIN21接小车右电机，  
PIN6、PIN26分别为左右电机  
的输出使能管脚
- PIN12、PIN13、PIN20、  
PIN21输出PWM脉冲实现小车  
调节速度



# 运动控制代码

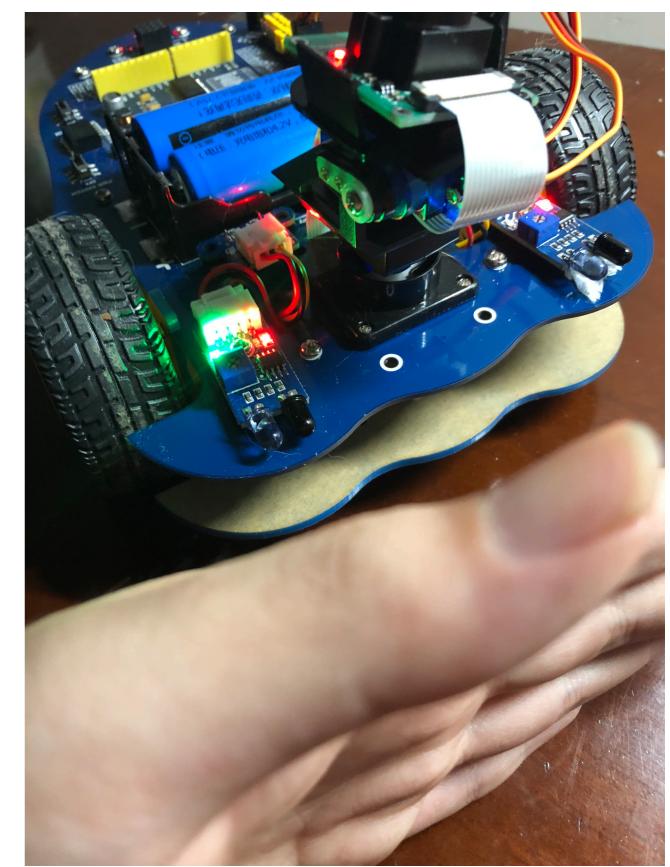
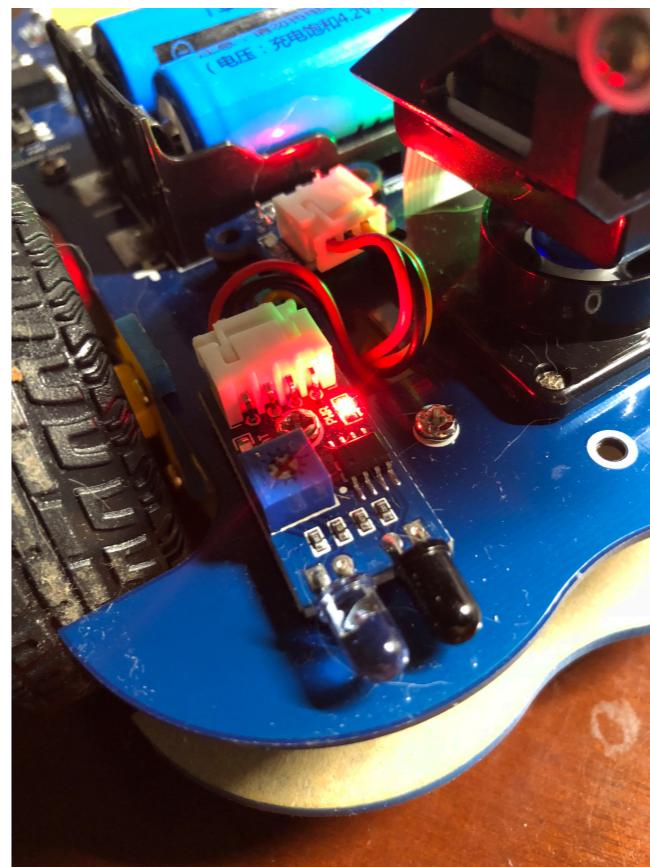
```
1 import RPi.GPIO as GPIO
2 import time
3
4 class MotionBase(object):
5
6     def __init__(self,in1=12,in2=13,ena=6,in3=20,in4=21,enb=26):
7         self.IN1 = in1
8         self.IN2 = in2
9         self.IN3 = in3
10        self.IN4 = in4
11        self.ENA = ena
12        self.ENB = enb
13
14        GPIO.setmode(GPIO.BCM)
15        GPIO.setwarnings(False)
16        GPIO.setup(self.IN1,GPIO.OUT)
17        GPIO.setup(self.IN2,GPIO.OUT)
18        GPIO.setup(self.IN3,GPIO.OUT)
19        GPIO.setup(self.IN4,GPIO.OUT)
20        GPIO.setup(self.ENA,GPIO.OUT)
21        GPIO.setup(self.ENB,GPIO.OUT)
22        self.forward()
23        self.PWMA = GPIO.PWM(self.ENA,600)
24        self.PWMB = GPIO.PWM(self.ENB,600)
25        self.PWMA.start(60)
26        self.PWMB.start(60)
27
28    def forward(self):
29        GPIO.output(self.IN1,GPIO.HIGH)
30        GPIO.output(self.IN2,GPIO.LOW)
31        GPIO.output(self.IN3,GPIO.LOW)
32        GPIO.output(self.IN4,GPIO.HIGH)
33
34    def stop(self):
35        GPIO.output(self.IN1,GPIO.LOW)
36        GPIO.output(self.IN2,GPIO.LOW)
37        GPIO.output(self.IN3,GPIO.LOW)
38        GPIO.output(self.IN4,GPIO.LOW)
39
40    def backward(self):
41        GPIO.output(self.IN1,GPIO.LOW)
```

"MotionBase.py" [dos] 82L, 2203C

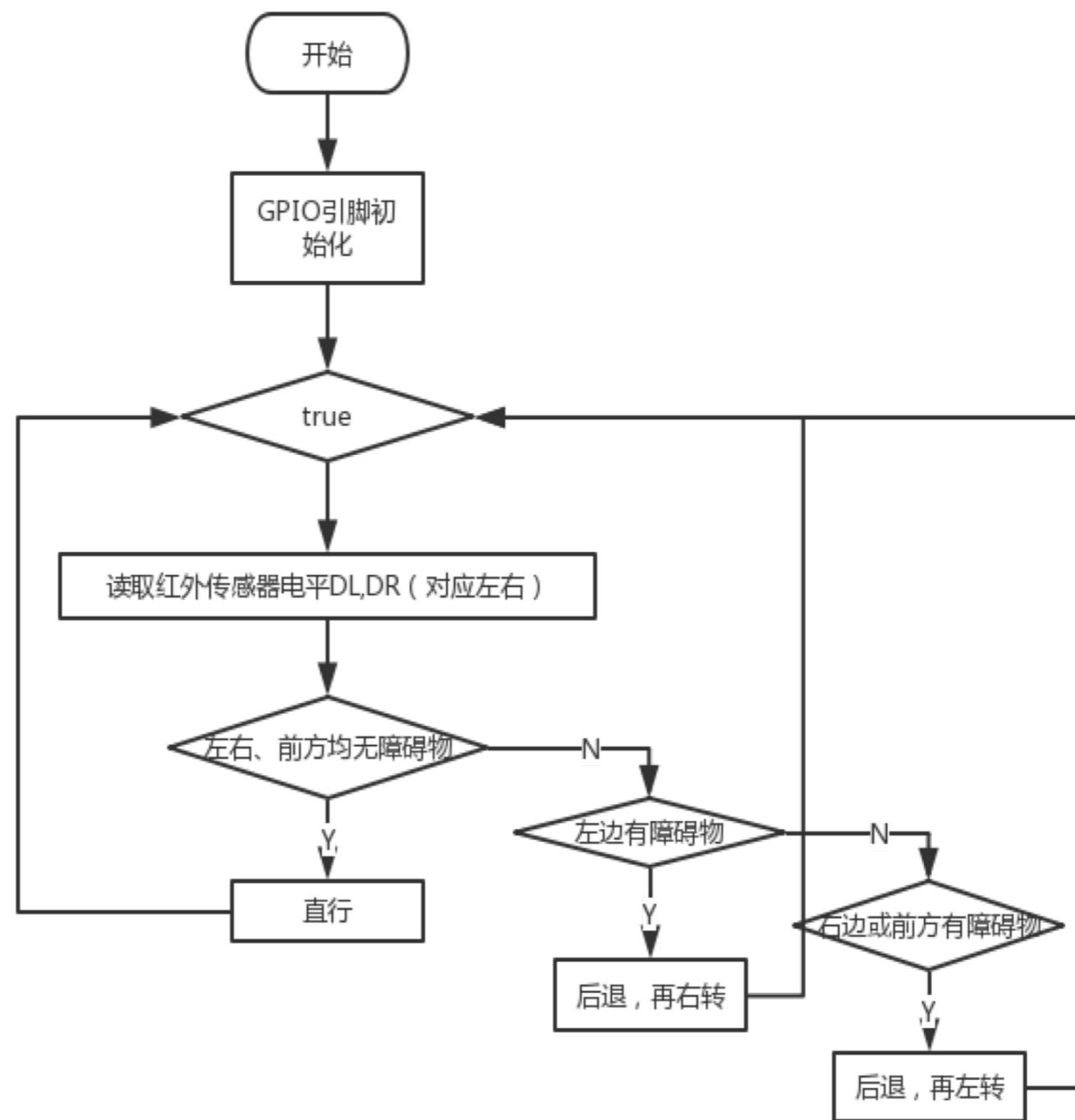
```
42        GPIO.output(self.IN2,GPIO.HIGH)
43        GPIO.output(self.IN3,GPIO.HIGH)
44        GPIO.output(self.IN4,GPIO.LOW)
45
46    def left(self):
47        GPIO.output(self.IN1,GPIO.LOW)
48        GPIO.output(self.IN2,GPIO.LOW)
49        GPIO.output(self.IN3,GPIO.LOW)
50        GPIO.output(self.IN4,GPIO.HIGH)
51
52    def right(self):
53        GPIO.output(self.IN1,GPIO.HIGH)
54        GPIO.output(self.IN2,GPIO.LOW)
55        GPIO.output(self.IN3,GPIO.LOW)
56        GPIO.output(self.IN4,GPIO.LOW)
57
58    def setPWMA(self,value):
59        self.PWMA.ChangeDutyCycle(value)
60
61    def setPWMB(self,value):
62        self.PWMB.ChangeDutyCycle(value)
63
64    def setMotor(self, left, right):
65        if((right >= 0) and (right <= 100)):
66            GPIO.output(self.IN1,GPIO.HIGH)
67            GPIO.output(self.IN2,GPIO.LOW)
68            self.PWMA.ChangeDutyCycle(right)
69        elif((right < 0) and (right >= -100)):
70            GPIO.output(self.IN1,GPIO.LOW)
71            GPIO.output(self.IN2,GPIO.HIGH)
72            self.PWMA.ChangeDutyCycle(0 - right)
73        if((left >= 0) and (left <= 100)):
74            GPIO.output(self.IN3,GPIO.HIGH)
75            GPIO.output(self.IN4,GPIO.LOW)
76            self.PWMB.ChangeDutyCycle(left)
77        elif((left < 0) and (left >= -100)):
78            GPIO.output(self.IN3,GPIO.LOW)
79            GPIO.output(self.IN4,GPIO.HIGH)
80            self.PWMB.ChangeDutyCycle(0 - left)
81
82    █
```

# 红外传感器

- 红外传感器由红外发射管和红外接收管组成
- 当没有障碍物时，接收管断开，对应代码中读到接收管的状态为1
- 当前方出现障碍物时，接收管导通，对应代码中读到的接收管状态为0
- 红外避障模块和红外跟踪模块都是基于红外传感器完成的



# 避障工作流程图

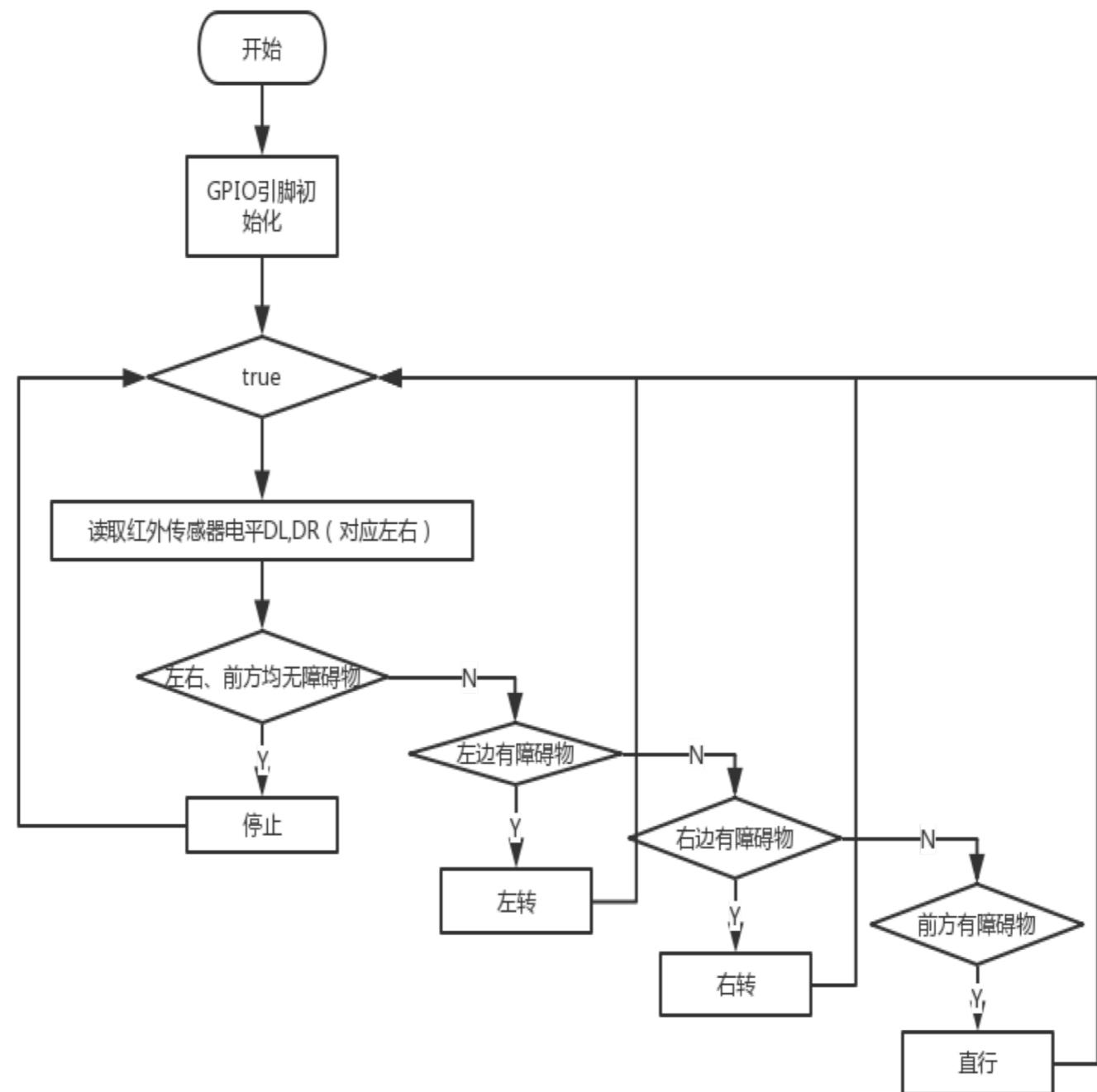


# 红外避障代码

```
1 import RPi.GPIO as GPIO
2 import time
3 from MotionBase import MotionBase
4
5 motion = MotionBase()
6
7 DR = 16
8 DL = 19
9
10 GPIO.setmode(GPIO.BCM)
11 GPIO.setwarnings(False)
12 GPIO.setup(DR,GPIO.IN,GPIO.PUD_UP)
13 GPIO.setup(DL,GPIO.IN,GPIO.PUD_UP)
14
15 try:
16     while True:
17         DR_status = GPIO.input(DR)
18         DL_status = GPIO.input(DL)
19         if((DL_status == 1) and (DR_status == 1)):
20             motion.forward()
21             print("forward")
22         elif((DL_status == 1) and (DR_status == 0)):
23             motion.left()
24             print("left")
25         elif((DL_status == 0) and (DR_status == 1)):
26             motion.right()
27             print("right")
28         else:
29             motion.backward()
30             time.sleep(0.2)
31             motion.left()
32             time.sleep(0.2)
33             motion.stop()
34             print("backward")
35
36 except KeyboardInterrupt:
37     GPIO.cleanup();
38
~
~
~
```

"Infrared\_Obstacle\_Avoidance.py" [dos] 38L, 759C

# 跟踪工作流程图

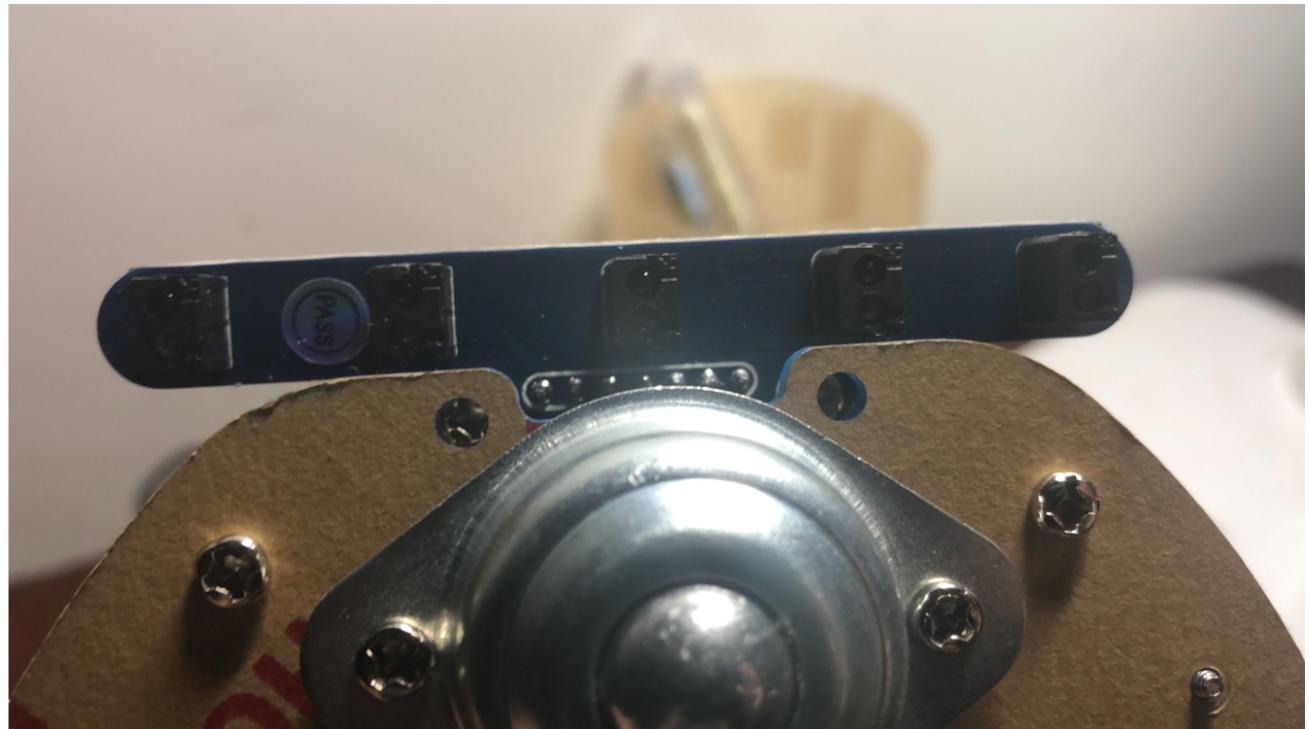


# 红外跟踪代码

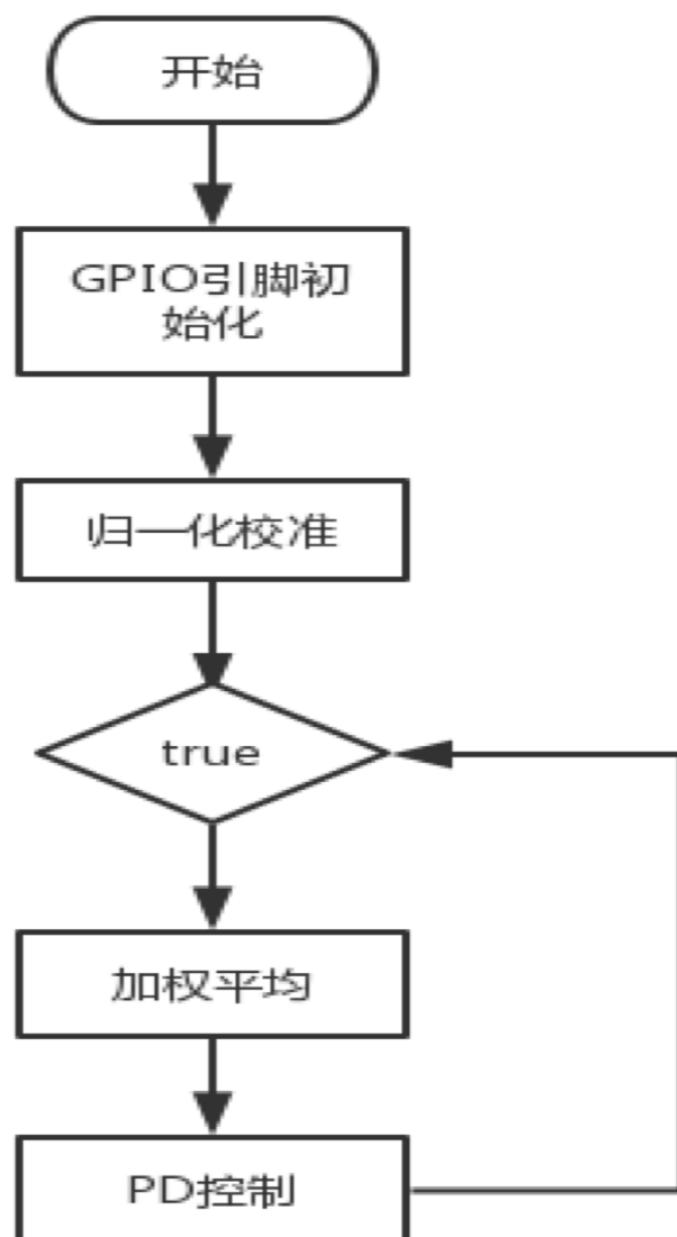
```
1 import RPi.GPIO as GPIO
2 import time
3 from MotionBase import MotionBase
4 import smbus
5
6 motion = MotionBase()
7
8 DR = 16
9 DL = 19
10
11
12 GPIO.setmode(GPIO.BCM)
13 GPIO.setwarnings(False)
14 GPIO.setup(DR,GPIO.IN,GPIO.PUD_UP)
15 GPIO.setup(DL,GPIO.IN,GPIO.PUD_UP)
16
17 motion.stop()
18 try:
19     while True:
20         DR_status = GPIO.input(DR)
21         DL_status = GPIO.input(DL)
22         if((DL_status == 0) and (DR_status == 0)):
23             motion.forward()
24             print("forward")
25         elif((DL_status == 1) and (DR_status == 0)):
26             motion.right()
27             print("right")
28         elif((DL_status == 0) and (DR_status == 1)):
29             motion.left()
30             print("left")
31         else:
32             motion.stop()
33             print("stop")
34
35 except KeyboardInterrupt:
36     GPIO.cleanup();
37
```

# 五路红外循迹模块

- 循迹模块和红外避障模块类似
- 红外发射二极管不断发射红外线，当发射出的红外线被物体反射时，被红外接收器接收，并输出模拟值。输出模拟值和物体距离以及物体颜色有关
- 循迹模块采用五路红外传感器通过测量五路红外传感器，判断黑线的位置，从而控制小车运动



# 循迹工作流程图



# 循迹代码

```
1 #!/usr/bin/python
2 # -*- coding:utf-8 -*-
3 import RPi.GPIO as GPIO
4 import time
5
6 CS = 5
7 Clock = 25
8 Address = 24
9 DataOut = 23
10
11 class TRSensor(object):
12     def __init__(self,numSensors = 5):
13         self.numSensors = numSensors
14         self.calibratedMin = [0] * self.numSensors
15         self.calibratedMax = [1023] * self.numSensors
16         self.last_value = 0
17
18     def AnalogRead(self):
19         value = [0,0,0,0,0,0]
20         #Read Channel0~channel4 AD value
21         for j in range(0,6):
22             GPIO.output(CS, GPIO.LOW)
23             for i in range(0,4):
24                 #sent 4-bit Address
25                 if(((j) >> (3 - i)) & 0x01):
26                     GPIO.output(Address,GPIO.HIGH)
27                 else:
28                     GPIO.output(Address,GPIO.LOW)
29         #read MSB 4-bit data
30         value[j] <= 1
31         if(GPIO.input(DataOut)):
32             value[j] |= 0x01
33         GPIO.output(Clock,GPIO.HIGH)
34         GPIO.output(Clock,GPIO.LOW)
35         for i in range(0,6):
36             #read LSB 8-bit data
37             value[j] <= 1
38             if(GPIO.input(DataOut)):
39                 value[j] |= 0x01
40             GPIO.output(Clock,GPIO.HIGH)
41             GPIO.output(Clock,GPIO.LOW)
42             #no mean ,just delay
43             for i in range(0,6):
44                 GPIO.output(Clock,GPIO.HIGH)
45                 GPIO.output(Clock,GPIO.LOW)
46             #
47             time.sleep(0.0001)
48             GPIO.output(CS,GPIO.HIGH)
49             return value[1:]
50
51     def calibrate(self):
52         max_sensor_values = [0]*self.numSensors
53         min_sensor_values = [0]*self.numSensors
54         for j in range(0,10):
55             sensor_values = self.AnalogRead();
56             for i in range(0,self.numSensors):
57
58                 # set the max we found THIS time
59                 if((j == 0) or max_sensor_values[i] < sensor_values[i]):
60                     max_sensor_values[i] = sensor_values[i]
61
62                 # set the min we found THIS time
63                 if((j == 0) or min_sensor_values[i] > sensor_values[i]):
64                     min_sensor_values[i] = sensor_values[i]
65
66         # record the min and max calibration values
67         for i in range(0,self.numSensors):
68             if(min_sensor_values[i] > self.calibratedMin[i]):
69                 self.calibratedMin[i] = min_sensor_values[i]
70             if(max_sensor_values[i] < self.calibratedMax[i]):
71                 self.calibratedMax[i] = max_sensor_values[i]
72
73     def readCalibrated(self):
74         value = 0
75         #read the needed values
76         sensor_values = self.AnalogRead();
77         for i in range (0,self.numSensors):
78
79             denominator = self.calibratedMax[i] - self.calibratedMin[i]
```

# 循迹代码

```
83     if(denominator != 0):
84         value = (sensor_values[i] - self.calibratedMin[i])* 1000 / denominator
85
86     if(value < 0):
87         value = 0
88     elif(value > 1000):
89         value = 1000
90
91     sensor_values[i] = value
92
93     print("readCalibrated",sensor_values)
94     return sensor_values
95
96 """
97
98     0*value0 + 1000*value1 + 2000*value2 + ...
99     -----
100     value0 + value1 + value2 + ...
101 """
102
103 def readLine(self, white_line = 0):
104
105     sensor_values = self.readCalibrated()
106     avg = 0
107     sum = 0
108     on_line = 0
109     for i in range(0,self.numSensors):
110         value = sensor_values[i]
111         if(white_line):
112             value = 1000-value
113         # keep track of whether we see the line at all
114         if(value > 200):
115             on_line = 1
116
117         # only average in values that are above a noise threshold
118         if(value > 50):
119             avg += value * (i * 1000); # this is for the weighted total,
120             sum += value;           #this is for the denominator
121
122     if(on_line != 1):
123         # If it last read to the left of center, return 0.
```

```
if(self.last_value < (self.numSensors - 1)*1000/2):
    #print("left")
    return 0;

# If it last read to the right of center, return the max.
else:
    #print("right")
    return (self.numSensors - 1)*1000

self.last_value = avg/sum

return self.last_value

GPIO0.setmode(GPIO0.BCM)
GPIO0.setwarnings(False)
GPIO0.setup(Clock,GPIO0.OUT)
GPIO0.setup(Address,GPIO0.OUT)
GPIO0.setup(CS,GPIO0.OUT)
GPIO0.setup(DataOut,GPIO0.IN,GPIO0.PUD_UP)

if __name__ == '__main__':
    from MotionBase import MotionBase

    maximum = 35;
    integral = 0;
    last_proportional = 0

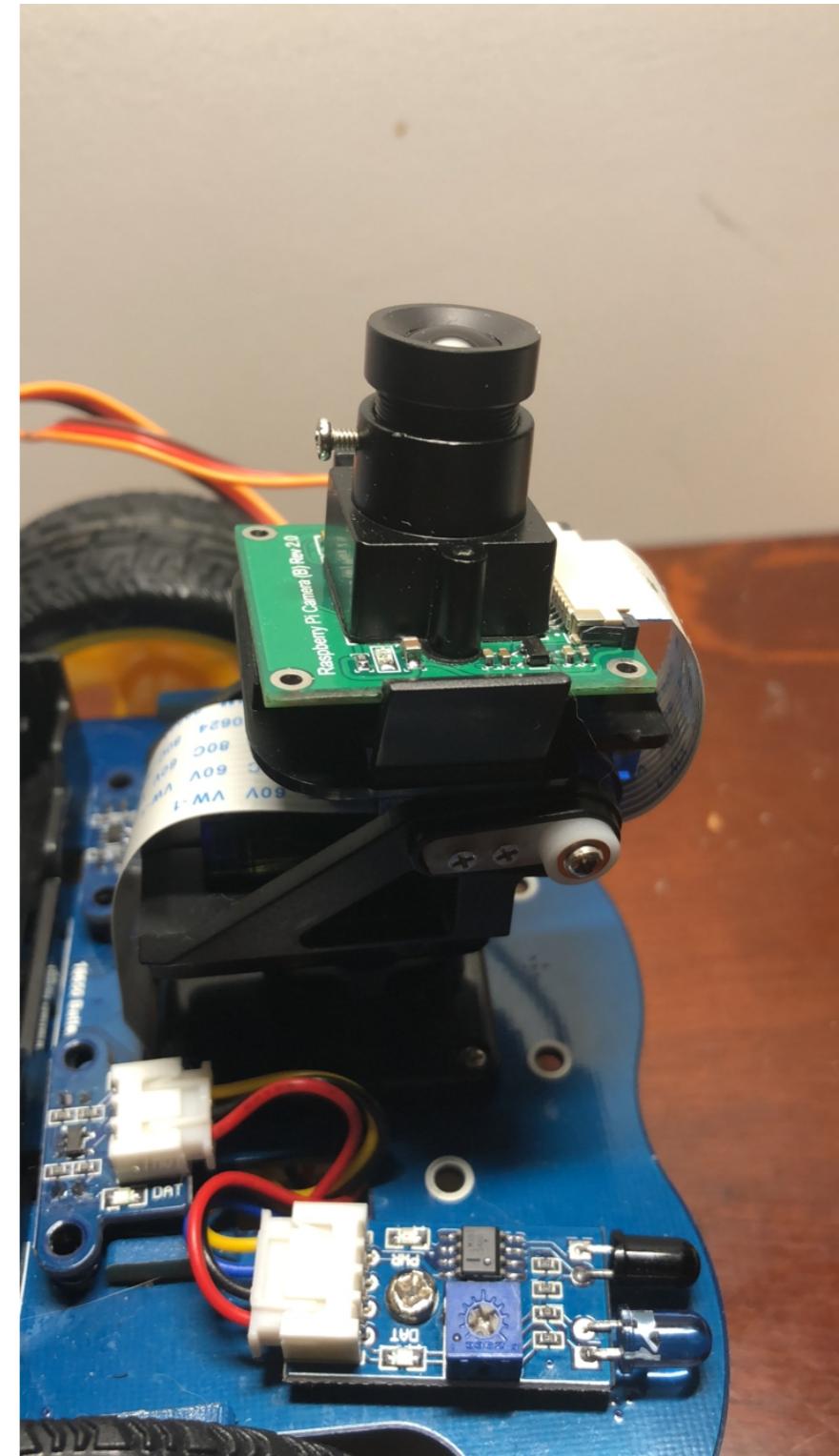
    TR = TRSensor()
    motion = MotionBase()
    motion.stop()
    print("Line follow Example")
    time.sleep(0.5)
    for i in range(0,400):
        TR.calibrate()
        print i
    print(TR.calibratedMin)
    print(TR.calibratedMax)
    time.sleep(0.5)
    motion.backward()
    while True:
```

# 循迹代码

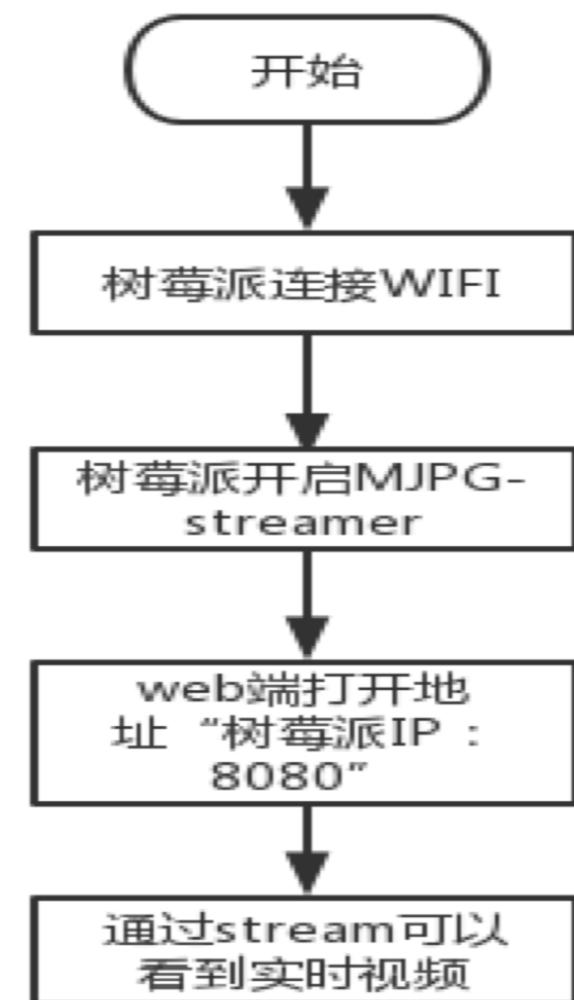
```
165     position = TR.readLine()
166     #print(position)
167
168     # The "proportional" term should be 0 when we are on the line.
169     proportional = position - 2000
170
171     # Compute the derivative (change) and integral (sum) of the position.
172     derivative = proportional - last_proportional
173     integral += proportional
174
175     # Remember the last position.
176     last_proportional = proportional
177
178     power_difference = proportional/25 + derivative/100 #+ integral/1000;
179
180     if (power_difference > maximum):
181         power_difference = maximum
182     if (power_difference < - maximum):
183         power_difference = - maximum
184     print(position,power_difference)
185     if (power_difference < 0):
186         motion.setPWMB(maximum + power_difference)
187         motion.setPWMA(maximum);
188     else:
189         motion.setPWMB(maximum);
190         motion.setPWMA(maximum - power_difference)
191
```

# 树莓派摄像头

- 采用的是树莓派官方摄像头，500W像素
- 通过CSI接口线连接树莓派
- 组装了一个舵机云台，但是底部控制水平方向旋转的舵机在安装过程中损坏，所以在这里舵机没有使用
- 通过mjpg-stream将视频流信息放在8080端口，在局域网内可以通过上位机web获得视频信息



# 监控工作流程图



# mjpg-stream开机自启动配置

在终端输入命令： sudo vim /etc/rc.local

```
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
    printf "My IP address is %s\n" "$_IP"
fi
cd /home/pi/mybot/mjpg-AlphaBot/mjpg-streamer
sudo ./start.sh
exit 0
```

# web端获得的实时影像

**Demo Pages**  
a ressource friendly  
streaming application

Home  
Static  
Stream  
Java  
Javascript  
VideoLAN  
Control

**Version info:**  
v0.1 (Okt 22, 2007)

**Hints**

This example shows a stream. It works with a few browsers like Firefox for example. To see a simple example click [here](#). You may have to reload this page by pressing F5 one or more times.

**Source snippet**

```

```

© The MJPG-streamer team | Design by Andrew Vilk

# 系统评价

该小车基于物联网的三层架构和模块化的设计思想完成，在掌握树莓派和AlphaBot基板的电路原理图及其二者的连接理论上，完成了各个功能模块的设计和测试工作。本系统综合利用嵌入式编程、python编程、电路、前向运动学、闭环控制等知识，实现了小车在一个地形不复杂、环境不恶劣的陌生环境中的自主避障、跟踪、巡线以及视频监控功能。系统的供电也是一个非常值得研究考虑的问题，本系统采用两节电池串接的7.4V电压输出，在电量消耗的过程中，输入的电压会不断降低，如何保证系统的正常运行或是及时的给出预警是一个未来需要解决的问题。

# 系统特点

小车系统的设计是基于模块化的设计思想的，这可以在日后小车的研究过程中方便的进行代码复用；相关执行脚本的编码是基于python2.7he RPI.GPIO库的，代码编写逻辑十分清晰，不需要考虑最底层的具体实现，同时也可以非常方便地移植到其他平台；同时，本系统实现了小车在一个地形不复杂、环境不恶劣的陌生环境中的自主避障、跟踪、巡线以及视频监控功能，但是精度不是很高。

# 系统不足

- 小车在通过红外传感器进行避障、跟踪的时候，轮子的运动控制存在一定的滞后性，也就是说主控程序得到的结果不能第一时间反应在运动上，导致避障测试时有时候在撞上障碍物后才开始表现出避障，导致跟踪测试时等待一会小车才开始跟踪物体。这是直流电机本身的特性导致的不足之处，在未来会更换直流电机和控制芯片以期望达到更加灵敏、精确的控制。
- 在避障的实现中，由于只有两个前置的红外传感器，所以只考虑了前方、左边、右边有障碍物的情况。
- 在避障的实现中，由于小车车体的缘故，导致在避障转弯的过程中有时候会因为空间不足而出现小车卡在障碍物上一直只有一个轮子转动，使小车一直无法脱离避开障碍物。在目前的代码修改中，只增加了后退运动的时间，但是由于滞后性，有时候依然无法解决这个问题。
- 小车开始循迹前，在黑线上方左右晃动小车幅度大时，小车在循迹时运动的左右摇晃幅度也是比较大的；当晃动幅度小时。小车在循迹时运动的左右摇晃幅度会比较小，但会出现偶尔偏离黑线的情况。这就导致了小车行进姿态的稳定性和循迹的稳定性的矛盾。因为在这方面算法了解的不透彻，导致执行效果不尽如人意。同时也没有考虑到在循迹过程中如果脱离了黑线的补救措施，导致其在脱离黑线后会进行胡乱的运动。

# 系统不足

- 在视频监控的实现中，一开始做了一个可以通过web控制小车转向并获得实时视频的网页信息，但是通过ip+8000端口号（在webiopi的config文件中配置的8000端口）无法访问我写的网页界面，解决问题无果，所以使用了MJPG-streamer这个现成的开源库来完成了web端的视频监控，并没有控制小车转向的功能，这是很大的一点不足之处。
- 在设置视频监控模块和避障等其他模块的开机自启动的时候，出现了只能设置一个脚本自启动的情况，所以目前只做了视频监控模块的自启动，其他模块的启动需要通过远程登录终端进行手动启动。
- 摄像头云台的舵机损坏，没有完成摄像头方向的控制功能。
- 由于树莓派的主频限制及发热问题，没有完成小车的SLAM功能。

# 收获

在本次基于树莓派的避障、跟踪、循迹、监控小车的设计过程中，对此前学习的类似嵌入式系统设计、机器人的知识有了更深的理解，尤其是对于一个轮式机器人的体系架构及必要硬件组成有了非常深刻的认识，并真正的将这些知识组合完成了一个轮式机器人的开发。虽然本次是在易于实现的树莓派平台上完成的，同时基板也定义了所要用的模块的接口，这也给开发大大降低了难度，但是这并不妨碍对于机器人相关接口和算法基本思想的学习。在前期花费大量时间组装机器人的过程中，引脚的跳线连接以及各个GPIO口的定义对我造成了很大的困扰，此前在实验室中做的开发很少需要做整体性的连接；同时在连接元器件的时候还需要考虑短路安全性问题，为此在很多地方都需要加上绝缘螺母。小车是一个脱离直连电源的系统，而上面连接的各个模块以及树莓派本身的功耗又比较大，这让小车只能在电池的供电下运行1小时左右就会罢工（因为电压不足，无法驱动树莓派）。记得之前在机器人论坛上有一个大牛讲过：简单机器人设计最大的问题不是运动控制以及各种算法的实现，而是供电和续航问题，这回算是真正体会了。

同时通过本次小车的避障功能的实现过程，我体会到了要想实现一个真正可以在陌生环境中进行自主稳定避障的小车的困难性。其困难性来源于两方面：一是直流电机驱动的问题，如果直流电机中没有减速箱，就会出现我在测试中小车直接装上障碍物的情况，而如果加入了减速箱，就会减慢小车轮子的转速，使其转向变得比较困难，这也会加大小车的功耗；二是在避障过程中需要考虑很多情况，不单单是红外传感器指向方向的障碍物情况，比如还应该考虑万一遇到了在转弯过程中小车转弯的空间不够导致下车卡死在障碍物上的情况。这个问题有在本次小车的实现中的解决方案是先后退一定的距离，但是这个距离需要如何把控，在具体实现中采用了执行后退运动固定的时间，但是因为滞后性移动的距离是不确定的。一种考虑的方案是根据轮编码器来后退固定的距离，通过小车的当前姿态以及小车的形状构造来决定后退距离，但是在轮式小车上获得当前小车姿态是比较困难的。

**THE END**