**自动驾驶树莓派小车（含全部代码）**

**作 者：**Mark\_Zhao\_9527

**时 间：**2016.08.02

**参考作者：1** www.sentdex.com 引用作者 观澜小站（优酷）

2 **[number007cool](http://blog.chinaunix.net/uid/21658993.html)** http://blog.chinaunix.net/uid-21658993-id-4694734.html

**声 明：**

1、本系列文档原始资料，能查到原始作者外都已标明原作者，否则无法标明。

2、欢迎各位朋友分享。

3、本系列提倡动手实践，含有一些PYTHON语言讲解，适合初学者，大牛请绕道！

4、本系列重新实践了网上的一些树莓派教程同时加入一些新的设计，力求系统全面、全开源、易理解。但由于能力有限，若文章中有问题，欢迎朋友们指正。

**目录：**

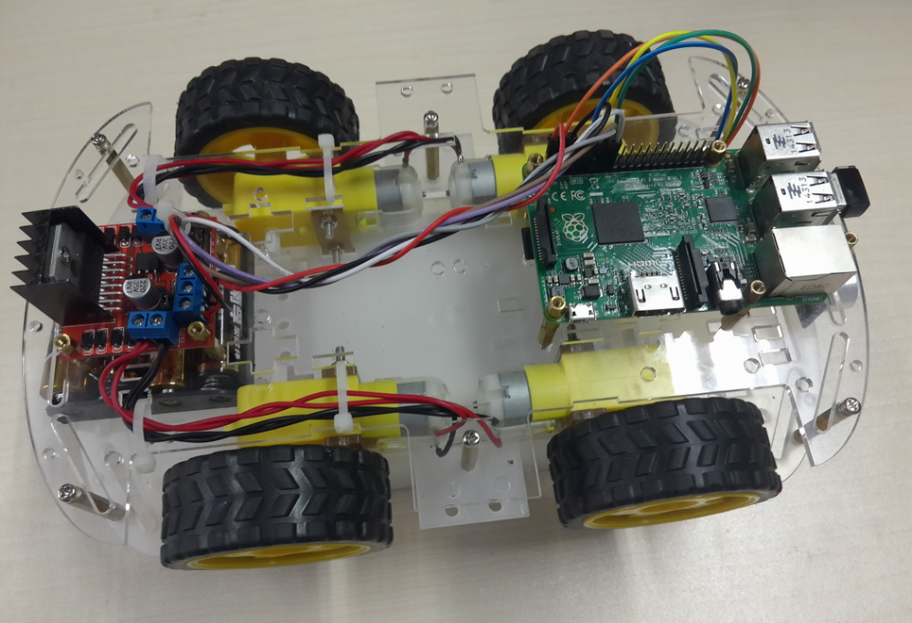
一、树莓派自驾驶小车的准备工作

二、单个轮子的控制——前进、后退

三、4个轮子的联动——前进、后退、左转、右转、左旋转、右旋转

四、超声传感器的连接与编程

五、树莓派自动驾驶小车的编程与实践



**一、树莓派自驾驶小车的准备工作**

**物料准备：**

1、树莓派一个

2、HC-SR04超声波测距传感器模块

3、无线网卡一个

4、L298N电机驱动板一个

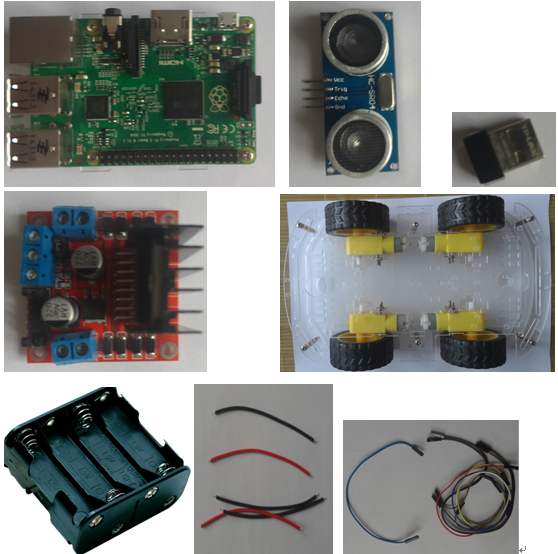
5、移动电源一个（输出至少2A以上）

6、电源小盒一个（用以通过L298N板电机供电）

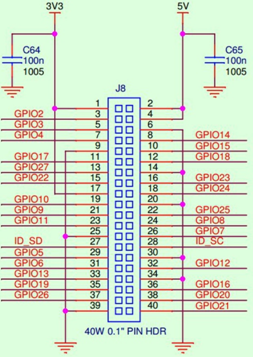
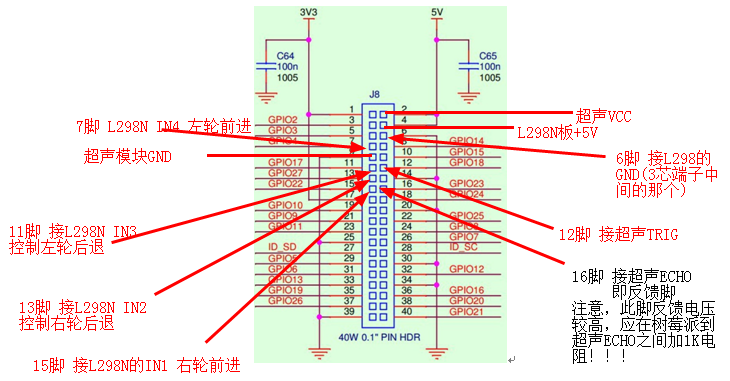
7、1K电阻一个（用以连接超声传感器与树莓派）

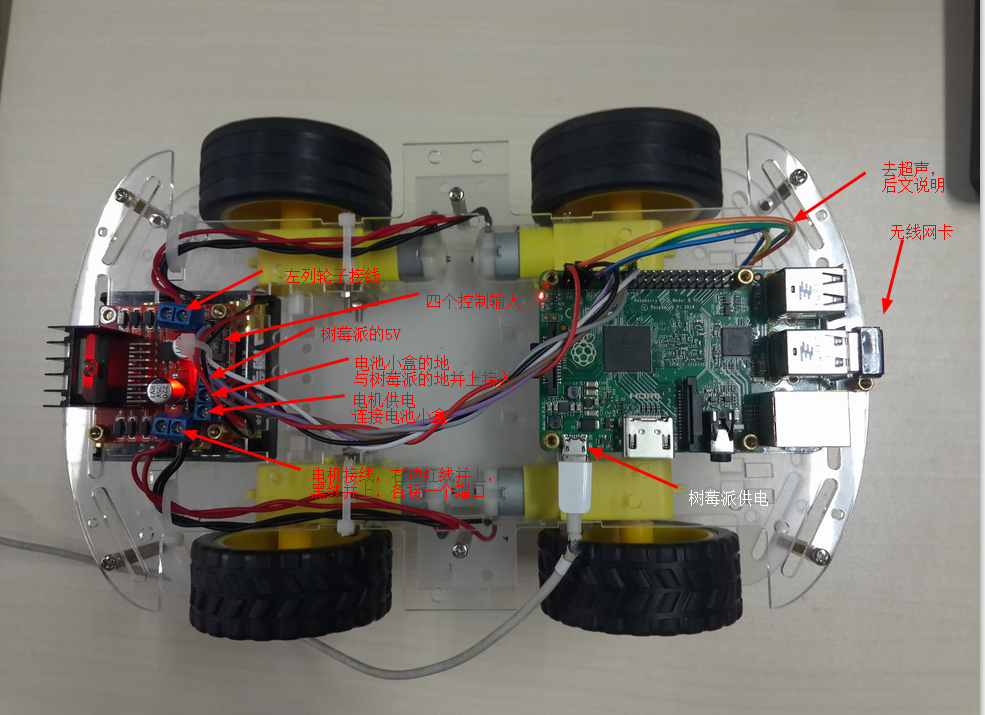
8、小车一个

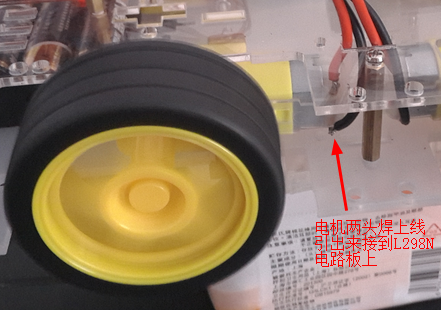
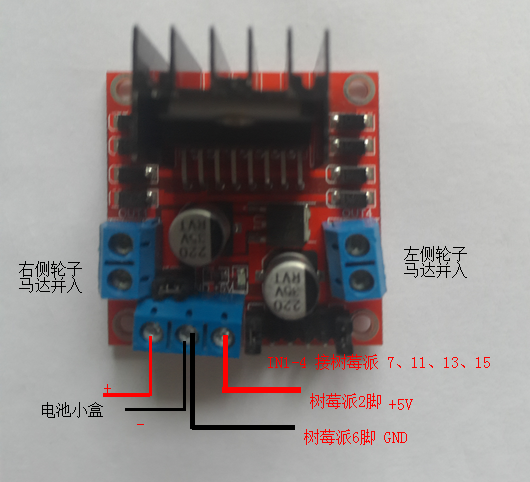
9、杜邦线若干



**线路连线：**



**二、一个轮子的控制——前进、后退**

由于我现在小车所有的轮子都已装好。左侧两个轮子是并入L298N一侧端口的。所以现在我用两个轮子的转动来演示本小节。（原理和程序都一模一样）

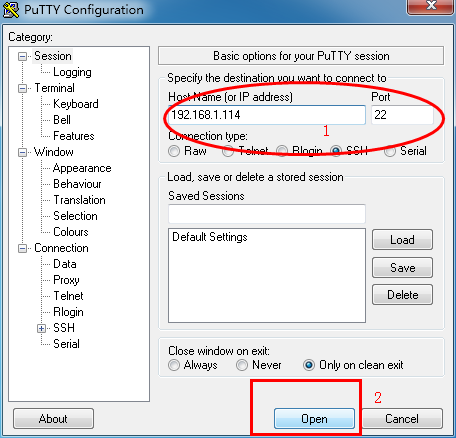
由于我手上没上HDMI线，所以我要远程控制我的小派。我先从路由器查寻到我的IP，然后用putty软件登入。

（如果，你不会上面一步，不要紧，看下我的**第二课教程：网络配置及远程控制**）

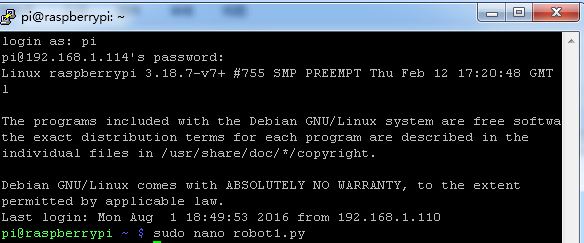
第一步：putty远程登入树莓派

输入帐户名：pi

输入密码：raspberry 注意密码不可见



第二步：输入 sudo nano robot1.py 来创建一个程序



第三步 ：编写如下图的代码

建议朋友自行输入，此段程序我没有拷贝TXT文档。后面复杂一些的程序我都贴出TXT文档。

L298N是一块电机驱动板。

接好电机电源及控制电源后。然后就是IN1/IN2/IN3/IN4了。

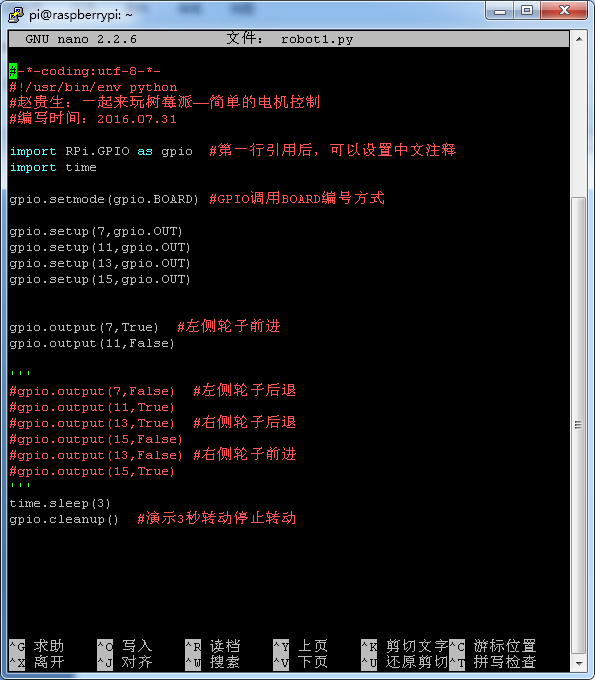
IN1为高电平（树莓派有输出就是高电平）控制一个轮子的前进，低电平（没输出）就不转

IN2则高电平（树莓派有输出就是高电平）控制上面的轮子的后退，低电平就不转。

IN3同理IN1，控制另一侧轮子的方向前进。

IN4同理IN2，控制另一侧轮子的方向后退。

为方便阅读程序，我加上一些注释，供参考！



注意：1、要在程序中使用中文字符，要加上第一行。

2、程序可以’’’ ‘’’ 注释一段，也可用#注释掉一行。注释完后该段/句不会被执行。

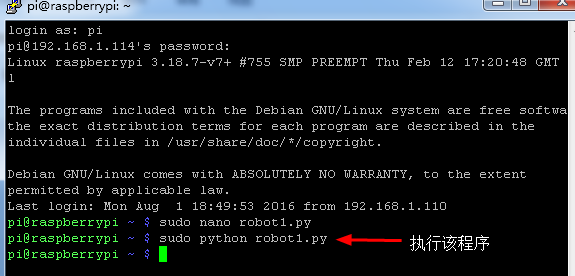
3、本人是在文中用#来注释掉其他的来演示的，你也可以尝试注释其他语句来看你的小车执行情况。

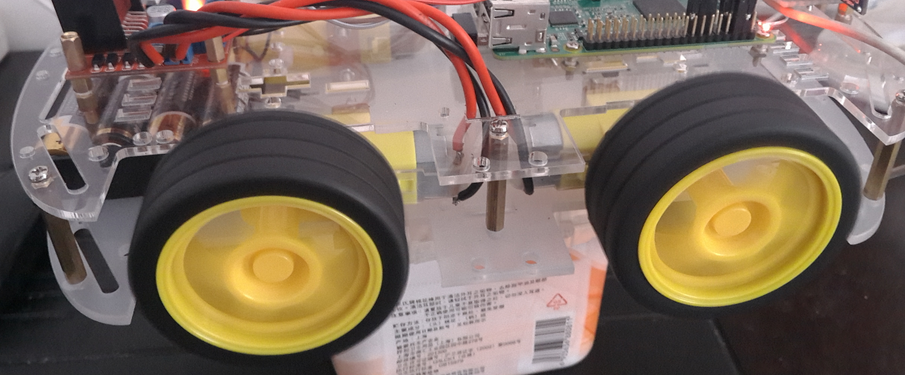
4、如果车轮方向不对，你可以倒一下L298N的IN1与IN2的位置。（IN3/IN4同理）

5、按CTRL+X 退出，点Y保存，按ENTER确认。

输入 sudo python robot1.py 执行程序

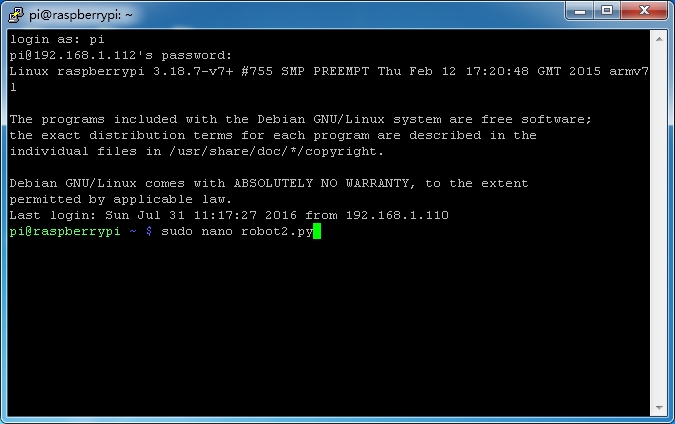
**看看效果：车轮按我想要的方向动了。**





**三、4个轮子的联动——前进、后退、左转、右转、左旋转、右旋转**

第一步：同上，输入 sudo nano robot2.py 创建一个新的程序



**第二步：编写robot2 程序**

本程序是联动4个轮子。

若4个轮子都前进，则小车前进

若左侧轮子不动，右侧轮子前进，则为左转。

若左侧轮子前进，右侧轮子不动，则为右转。

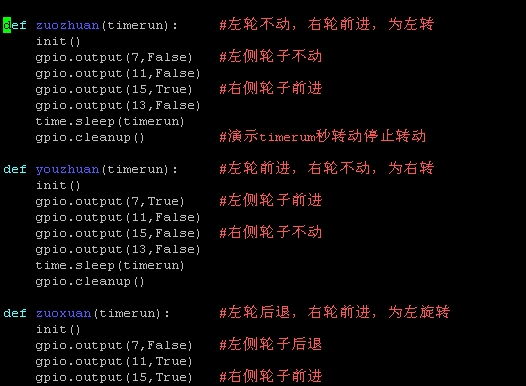
若左侧轮子前进，右侧轮子后退，则为右旋转。

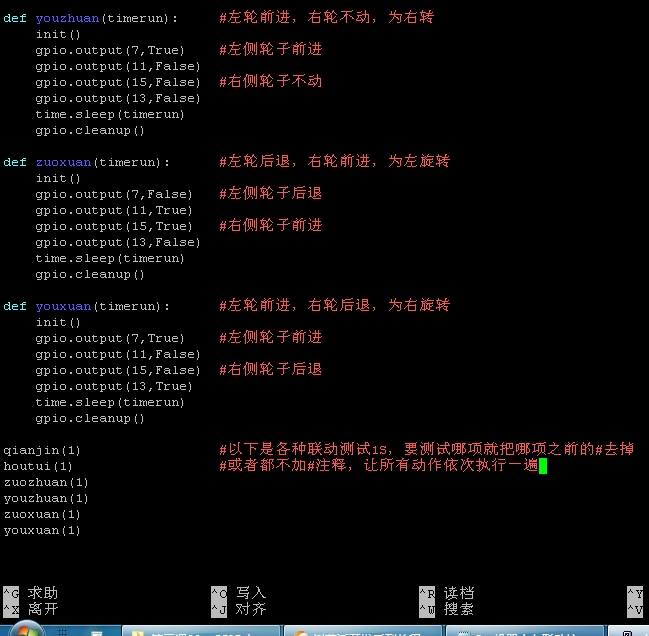
若左侧轮子后退，右侧轮子前进，则为左旋转。

下面是截图，后附代码TXT。如果你的程序有问题，请对比图片检查。

温馨提示：如果程序有问题，很有可能是排版没对齐。







#-\*-coding:utf-8-\*-

#!/usr/bin/env python

#赵贵生：一起来玩树莓派5——机器人小车联动控制：前进、后退、左转、右转、旋转

#编写时间：2016.07.31

import RPi.GPIO as gpio #第一行引用后，可以设置中文注释

import time

gpio.setwarnings(False) #去掉一些不必要的警告

zuoqian = 7 #7脚为控制左侧轮前进，11脚控制左侧轮子后退

zuohou = 11

youqian = 15 #15脚为控制右侧轮前进，13脚控制右侧轮子后退

youhou = 13

def init():

gpio.setmode(gpio.BOARD) #GPIO调用BOARD编号方式

gpio.setup(7,gpio.OUT)

gpio.setup(11,gpio.OUT)

gpio.setup(13,gpio.OUT)

gpio.setup(15,gpio.OUT)

def qianjin(timerun): #左右都往前，为前进

init() #初始化

gpio.output(7,True) #左侧轮子前进

gpio.output(11,False)

gpio.output(15,True) #右侧轮子前进

gpio.output(13,False)

time.sleep(timerun)

gpio.cleanup() #演示timerum秒转动停止转动

def houtui(timerun): #左右都往后，为前后

init()

gpio.output(7,False) #左侧轮子后退

gpio.output(11,True)

gpio.output(15,False) #右侧轮子后退

gpio.output(13,True)

time.sleep(timerun)

gpio.cleanup()

def zuozhuan(timerun): #左轮不动，右轮前进，为左转

init()

gpio.output(7,False) #左侧轮子不动

gpio.output(11,False)

gpio.output(15,True) #右侧轮子前进

gpio.output(13,False)

time.sleep(timerun)

gpio.cleanup() #演示timerum秒转动停止转动

def youzhuan(timerun): #左轮前进，右轮不动，为右转

init()

gpio.output(7,True) #左侧轮子前进

gpio.output(11,False)

gpio.output(15,False) #右侧轮子不动

gpio.output(13,False)

time.sleep(timerun)

gpio.cleanup()

def zuoxuan(timerun): #左轮后退，右轮前进，为左旋转

init()

gpio.output(7,False) #左侧轮子后退

gpio.output(11,True)

gpio.output(15,True) #右侧轮子前进

gpio.output(13,False)

time.sleep(timerun)

gpio.cleanup()

def youxuan(timerun): #左轮前进，右轮后退，为右旋转

init()

gpio.output(7,True) #左侧轮子前进

gpio.output(11,False)

gpio.output(15,False) #右侧轮子后退

gpio.output(13,True)

time.sleep(timerun)

gpio.cleanup()

qianjin(1) #以下是各种联动测试1S，要测试哪项就把哪项之前的#去掉

#houtui(1)

#zuozhuan(1)

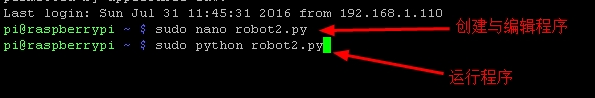
#youzhuan(1)

#zuoxuan(1)

#youxuan(1)

#这是终止行，以下返回正文==============================================

第三步： 按ctrl+ x退出，按Y确认，按ENTER写入。 输入 sudo robot2.py 执行



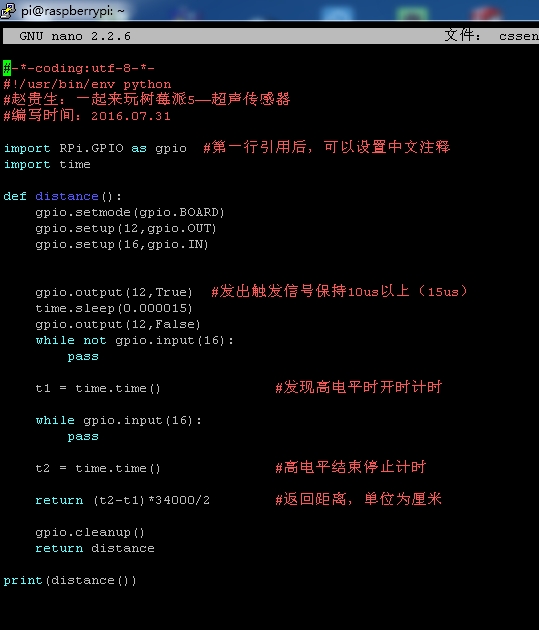
**OK,没有问题！**

**四、超声传感器的连接与编程**

首先，看一下HC-SR04超声波测距传感器模块的原理，给TRIG一个不低说10uS的触发，接收反馈的ECHO数据。从开始有信号开始计时，到信号没有再计一次时，算出一个时间差，剩以一个系数34000/2就是距离了！

第一步 输入sudo nano cscensor.py 创建超声程序

第二步 编写超声程序



#-\*-coding:utf-8-\*-

#!/usr/bin/env python

#赵贵生：一起来玩树莓派5——超声传感器

#编写时间：2016.07.31

import RPi.GPIO as gpio #第一行引用后，可以设置中文注释

import time

def distance():

gpio.setmode(gpio.BOARD)

gpio.setup(12,gpio.OUT)

gpio.setup(16,gpio.IN)

gpio.output(12,True) #发出触发信号保持10us以上（15us）

time.sleep(0.000015)

gpio.output(12,False)

while not gpio.input(16):

pass

t1 = time.time() #发现高电平时开时计时

while gpio.input(16):

pass

t2 = time.time() #高电平结束停止计时

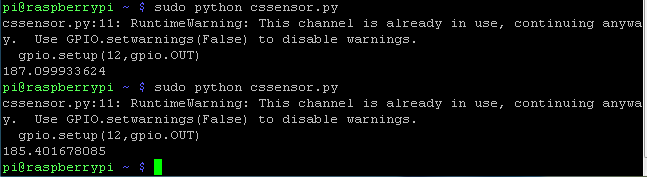
return (t2-t1)\*34000/2 #返回距离，单位为厘米

gpio.cleanup()

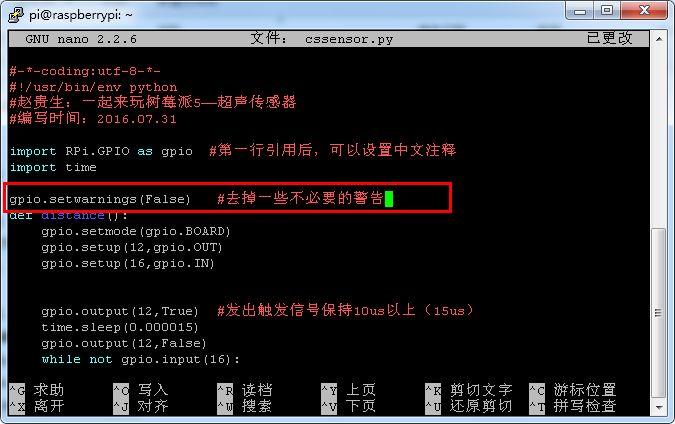
return distance

print(distance())

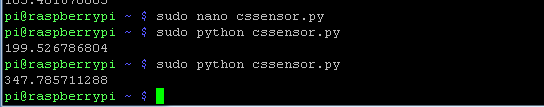
第三步 输入sudo cssensor.py 测试程序



这时，你发现程序报错，这里看错误就知道，要在程序中加点东西



按CTRL+X保存退出，再次运行程序，不再报错！



如果你要输出一些提示信息，同时给输出加上单位CM，你可以在打印输出那一句改成：



输出会变成下面的样子：



当然这是执行一次输出一个距离结果，如果你要程序自己不断的输出距离参数，以便自己改变距离来测试超声器件，你可以这么做，在主程序（最后一行）作点改变就行。

try:

while True:

print 'Distance: %0.2f m' %distance()

time.sleep(3)

except KeyboardInterrupt:

gpio.cleanup()

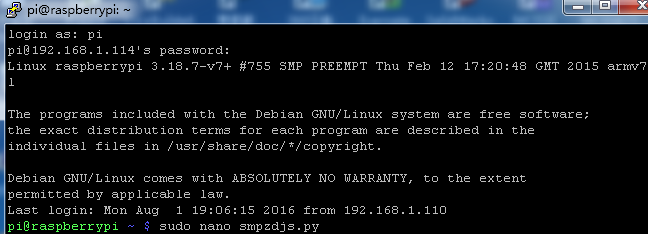
上面这段替换掉来原程序的最后一行，来形成一个循环程序，替换后，在执行程序时按CTRL+C随时可以退出程序。效果怎么样，你自己实践一下。

如果你要变成中文显示，只要做一些小的改变就行了，你自己摸索吧！

**五、树莓派自动驾驶小车的编程与实践**

自动驾驶程序怎么编写？嗯，没错，只要把第三个程序和第四个程序结合起来就欧了！

第一步 输入sudo nano smpzdjs.py 创建一个程序



第二步 编写程序

#-\*-coding:utf-8-\*-

#!/usr/bin/env python

#赵贵生：一起来玩树莓派5——树莓派机器人自动驾驶

#编写时间：2016.07.31

#引入autonomy函数，产生0-4之间随机数。随时数为0时测试前方距离。前过TF秒，1为原地左转。2为右转。3为左旋

#通过distance子程序测试距离，小于15cm倒退1S。如果前方还太近，原地左转3S，还太近，停止

import RPi.GPIO as gpio #第一行引用后，可以设置中文注释

import time

import sys

import random

gpio.setwarnings(False) #去掉一些不必要的警告

zuoqian = 7 #7脚为控制左侧轮前进，11脚控制左侧轮子后退

zuohou = 11

youqian = 15 #15脚为控制右侧轮前进，13脚控制右侧轮子后退

youhou = 13

def init():

gpio.setmode(gpio.BOARD) #GPIO调用BOARD编号方式

gpio.setup(7,gpio.OUT)

gpio.setup(11,gpio.OUT)

gpio.setup(13,gpio.OUT)

gpio.setup(15,gpio.OUT)

gpio.setup(12,gpio.OUT)

gpio.setup(16,gpio.IN)

def qianjin(timerun): #左右都往前，为前进

gpio.output(7,True) #左侧轮子前进

gpio.output(11,False)

gpio.output(15,True) #右侧轮子前进

gpio.output(13,False)

time.sleep(timerun)

gpio.cleanup() #演示timerum秒转动停止转动

def houtui(timerun): #左右都往后，为前后

gpio.output(7,False) #左侧轮子后退

gpio.output(11,True)

gpio.output(15,False) #右侧轮子后退

gpio.output(13,True)

time.sleep(timerun)

gpio.cleanup()

def zuozhuan(timerun): #左轮不动，右轮前进，为左转

gpio.output(7,False) #左侧轮子不动

gpio.output(11,False)

gpio.output(15,True) #右侧轮子前进

gpio.output(13,False)

time.sleep(timerun)

gpio.cleanup() #演示timerum秒转动停止转动

def youzhuan(timerun): #左轮前进，右轮不动，为右转

gpio.output(7,True) #左侧轮子前进

gpio.output(11,False)

gpio.output(15,False) #右侧轮子不动

gpio.output(13,False)

time.sleep(timerun)

gpio.cleanup()

def zuoxuan(timerun): #左轮后退，右轮前进，为左旋转

gpio.output(7,False) #左侧轮子后退

gpio.output(11,True)

gpio.output(15,True) #右侧轮子前进

gpio.output(13,False)

time.sleep(timerun)

gpio.cleanup()

def youxuan(timerun): #左轮前进，右轮后退，为右旋转

gpio.output(7,True) #左侧轮子前进

gpio.output(11,False)

gpio.output(15,False) #右侧轮子后退

gpio.output(13,True)

time.sleep(timerun)

gpio.cleanup()

def tingzhi(timerun): #刹车不走了

gpio.output(7,False)

gpio.output(11,False)

gpio.output(15,False)

gpio.output(13,False)

time.sleep(timerun)

gpio.cleanup()

def distance():

init()

gpio.output(12,True) #发出触发信号保持10us以上（15us）

time.sleep(0.000015)

gpio.output(12,False)

while not gpio.input(16):

pass

t1 = time.time() #发现高电平时开时计时

while gpio.input(16):

pass

t2 = time.time() #高电平结束停止计时

return (t2-t1)\*34000/2 #返回距离，单位为厘米

gpio.cleanup()

return distance

def check\_front():

init()

dist = distance()

if dist < 15:

print('close',dist)

init()

houtui(1) #速度大快后退1S足够

init()

dist = distance()

if dist < 15:

print('too close',dist)

init()

zuoxuan(3)

init()

houtui(1)

dist = distance()

if dist<15:

print('too too close',dist)

tingzhi(2)

def autonomy():

tf = 0.1

x = random.randrange(0,5)

if x==0:

check\_front()

init()

qianjin(0.2) #前进的时间稍微长一些，效果好一点

print '前进程序被执行。现在地距离是: %0.2f cm' %distance()

elif x==1:

check\_front()

init()

zuoxuan(0.2)

print '左旋程序被执行。现在地距离是: %0.2f cm' %distance()

elif x==2:

check\_front()

init()

youxuan(tf)

print '右旋程序被执行。现在地距离是: %0.2f cm' %distance()

elif x==3:

check\_front()

init()

zuozhuan(tf)

print '左转程序被执行。现在地距离是: %0.2f cm' %distance()

elif x==4:

check\_front()

init()

youzhuan(tf)

print '右转程序被执行。现在地距离是: %0.2f cm' %distance()

try:

while True:

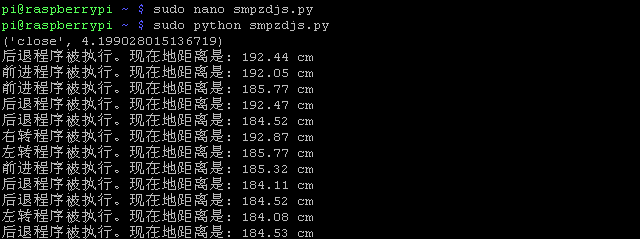
autonomy()

except KeyboardInterrupt:

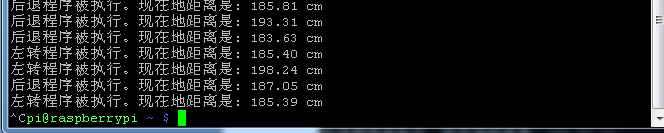
gpio.cleanup()

第三步 输入 sudo python smpzdjs.py 执行程序，以下是执行结果：

我在程序的每个子步骤中设定了标识点，即提示什么程序在执行，同时实时把距离反馈到远程操作端。



根据程序的设定，按CTRL+C退出程序，树莓派小车不再运动。



然后在实践中发现，整个小车会出现晃动的情况，那是因为我的随机数设的太多了，有太多选择可以被执行，所在要最后的小车的自主运动完美，我这边是去掉了几组运动。

当然，你可以自己随便设定。只须如下做这么一点修改。

def autonomy():

tf = 0.2

x = random.randrange(0,3)

if x==0:

check\_front()

init()

qianjin(0.3) #前进的时间稍微长一些，效果好一点

print '前进程序被执行。现在地距离是: %0.2f cm' %distance()

elif x==1:

check\_front()

init()

zuoxuan(0.3)

print '左旋程序被执行。现在地距离是: %0.2f cm' %distance()

elif x==2:

check\_front()

init()

youxuan(0.15)

print '右旋程序被执行。现在地距离是: %0.2f cm' %distance()

# elif x==3:

# check\_front()

# init()

# youxuan(tf)

# print '右旋程序被执行。现在地距离是: %0.2f cm' %distance()

# elif x==4:

# check\_front()

# init()

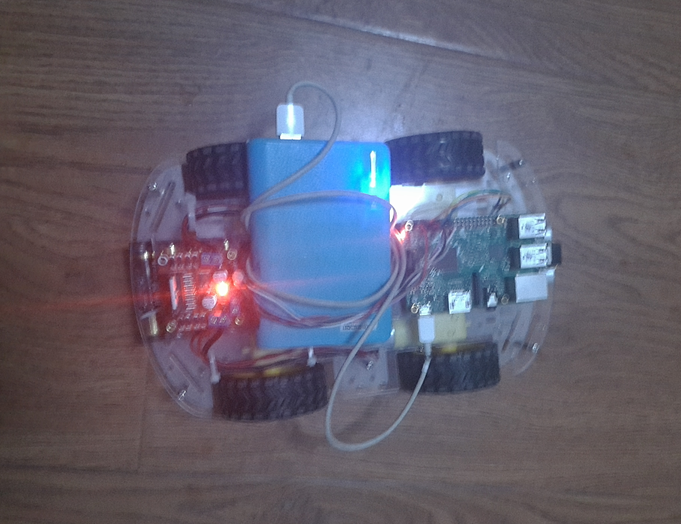
# youzhuan(tf)

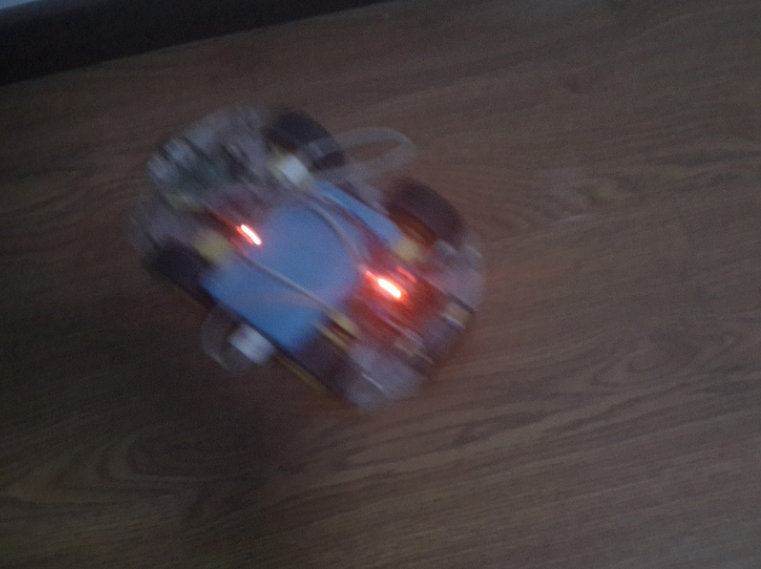
# print '右转程序被执行。现在地距离是: %0.2f cm' %distance()

这样车子就不晃动了，当距离近了，自己左旋找出路。

当然，这个随机数任由你编程，小车的完美运行还要靠根据实际情况调整呢！

So动手干吧！





嗖的一声，就跑了！