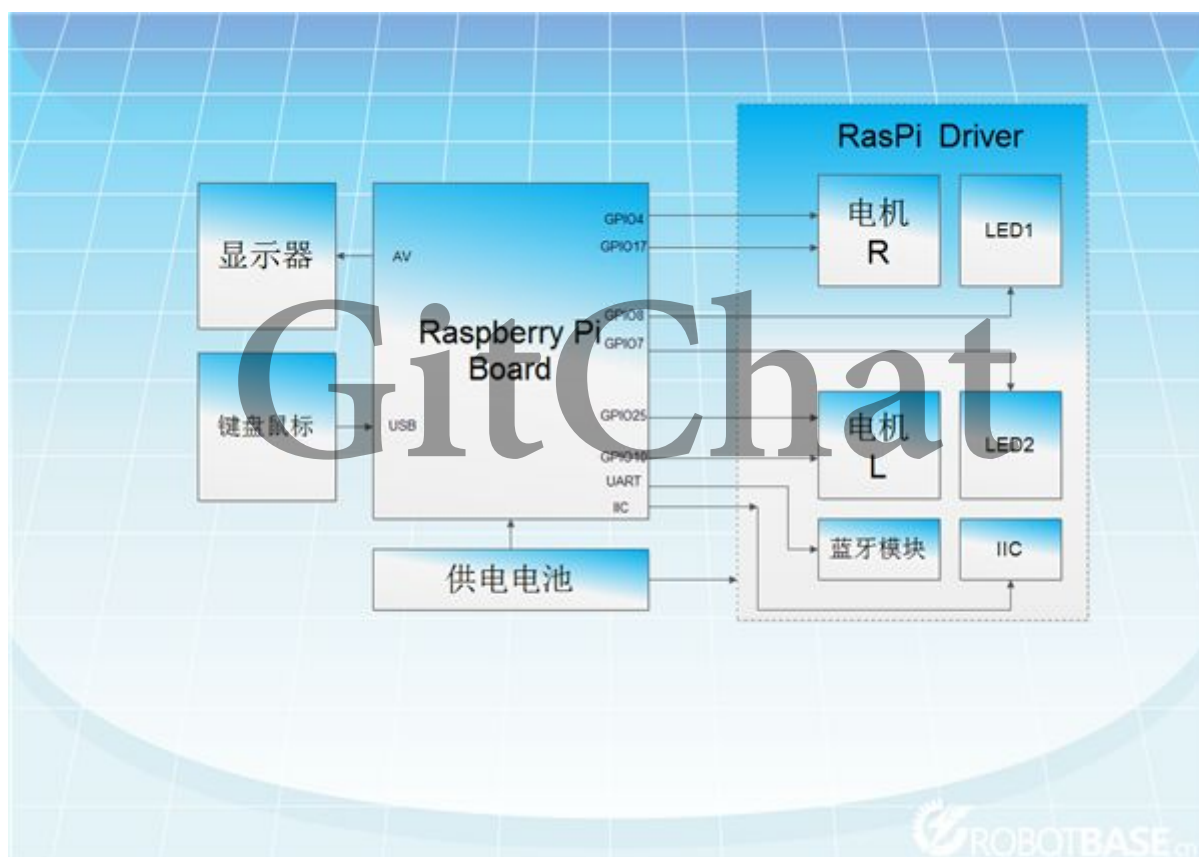
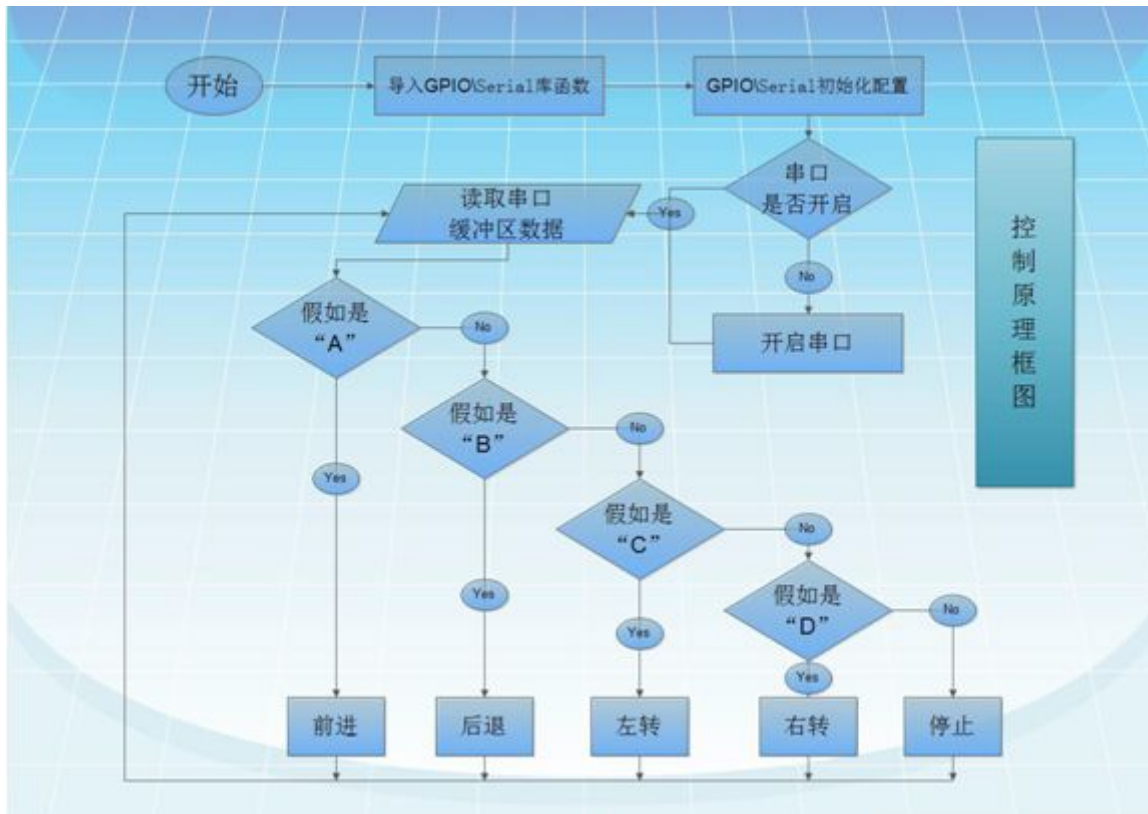


# 如何利用树莓派从无到有打造一款树莓派机器人

树莓派是为学习计算机编程教育而设计，只有信用卡大小的微型电脑，最早的系统基于Linux，随着Win10 IOT的发布，现在树莓派也可以运行Windows。树莓派虽然只有信用卡大小，但是内心却非常的强大，视频，音频等功能都是有的，现在树莓派3版本有1G内存，1.2GHZ频率，拥有操作系统的树莓派预留了40个可以驱动各种传感器和驱动器的I/O接口，所以我们使用树莓派作为机器人的控制器，将控制软件写入树莓派的TF卡中，然后这个软件通过GPIO接口就能控制驱动和传感器了。





树莓派和其他机器人的控制器有着本质的不同，因为树莓派有完善的操作系统（其他的只有控制系统）并且对Python支持的非常的好。所以使用Python语言可以快速的在树莓派上开发软件去控制机器人的传感器，树莓派还有另外一个优势就是它能够运行人工智能相关的算法，比如可以在上面运行SVM，能简单的对数据进行分类；

将树莓派作为机器人的大脑是未来的趋势，本场Chat围绕如何使用树莓派开发智能机器人控制系统展开，包括以下内容：

## 1. Raspbian操作系统介绍

树莓派的操作系统Raspbian的开发者是麦克·汤姆森和彼得·格林。该系统为官方推荐的操作系统。它巧妙地将Paspberry和Debian的名字合并成了一个单词。Raspbian是基于Debian的专为Cortex-A系列开发的能运行在树莓派上的操作系统。Debian是另一种发行版Ubuntu的基础，而Ubuntu是最为流行的Linux桌面系统，本系统拥有良好的社区支持。Raspbian系统附带着35000多个软件包，并集成了轻量级的图形界面LXDE。Raspbian提供了完善的功能，并且组织的非常好，能够支持最新的硬件和软件。

随着树莓派在极客社区中的广泛使用（目前树莓派总共卖出去了1250万台），树莓派已经成为世界第三大计算平台（前两个是windows和mac），所以树莓派基金会（备注：树莓派是由树莓派基金会开发的一款微型电脑，基金会只是设计电路图，设计操作系统和维护社区，树莓派的生产现在有RS和element14两家工厂进行）为桌面电脑打造了一款操作系统:Raspbian Pixel。

## 2. 使用python开发传感器驱动库GPIO库

前面我说过，树莓派打造的机器人主要是利用GPIO接口去控制机器人的驱动和各种传感器，因为树莓派有操作系统，所以使用Python可以开发控制软件，现在有很多的库文件支持，比如：<https://github.com/RPi-Distro/python-gpiozero>，这个库文件就对树莓派的GPIO支持的非常好。

树莓派的多个可编程的GPIO（General Purpose Input/Output）接口，可以用来驱动各种外设（如传感器，步进电机等）。目前在树莓派上流行的GPIO开发环境主要有两种，Python GPIO和基于C语言的wiringPi。我们推荐Python GPIO，因为Python不仅上手简单，而且其解释语言的特性使得程序不用编译，对代码做了任何修改之后就能直接运行，极大方便了调试。

所以我罗列一些GPIO的库文件供大家参考：

- <https://github.com/projectweekend/Pi-GPIO-Server>
- [https://github.com/adafruit/Adafruit\\_Python\\_GPIO](https://github.com/adafruit/Adafruit_Python_GPIO)
- <https://github.com/alaudet/hcsr04sensor>
- <https://github.com/vitiral/gpio>

### 3. 步进电机和超声波传感器

机器人分成四大部件：

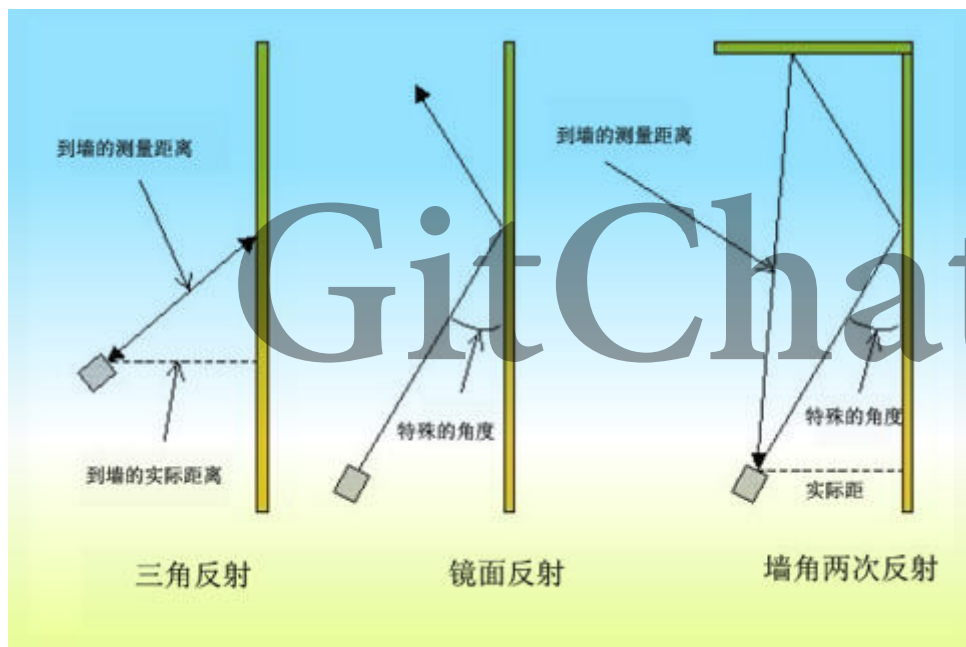
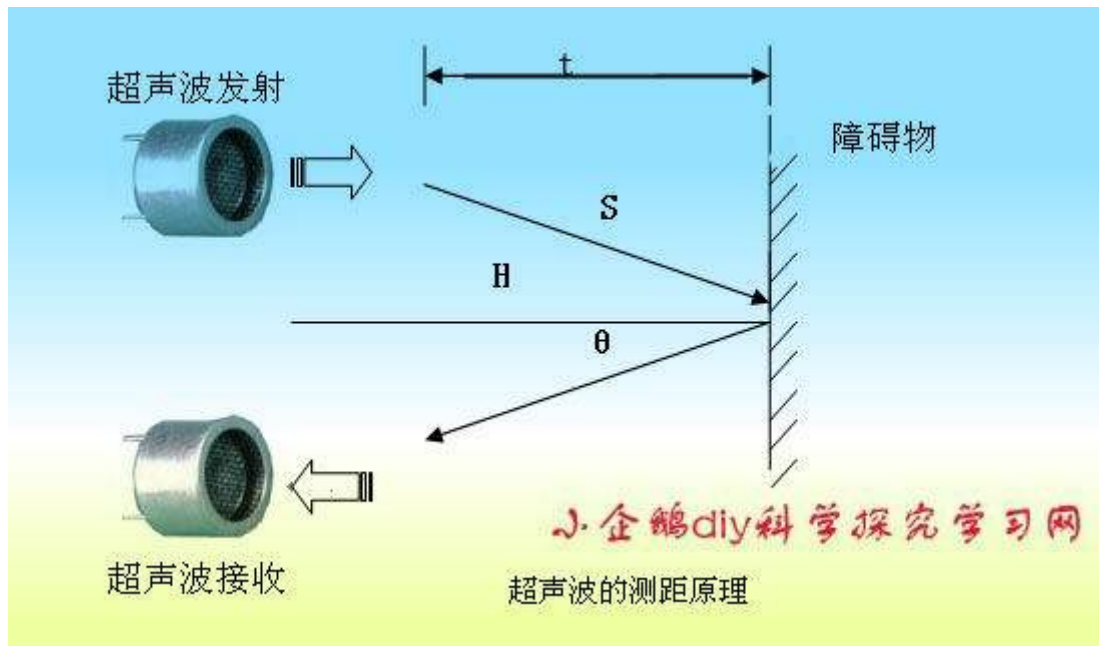
- 机器人控制系统，这部分有人称为是机器人“大脑”部分。
- 机械部分，机器人都是要执行一定的任务的，这部分任务就是机械部分完成的。
- 感知部分，机器人要不断的感知周围环境，并且通过感知进行信息或者数据的传输。
- 驱动部分，机器人通过驱动部件链接机械和传感器，从而驱动机械部件执行一定的任务。

步进电机顾名思义就是一步一走的一种驱动器，目前市面上对于小型服务级机器人有很多步进电机，大家可以通过搜索引擎或者电商网站找到相关的步进电机。

步进电机和超声波传感器都是协助树莓派实现功能的重要外设。通过Python编写相应程序后（编写的这些程序通过一定的工具烧进树莓派的TF卡中），我们可以驱动步进电机在不同时间正转和反转，从而带动相关连接部件的转动，例如：智能小车的车轮作向前或向后的运动；机械手的不同关节的摆动及夹取物品；无人机多轴螺旋桨的同步旋转实现升空。超声波传感器是利用超声波的特性研制而成的传感器。而安装了超声波传感器的智能机器人，在树莓派的控制下可以识别放在周围的障碍物，前后移动超声波测距仪，将收集的信号传输回到数据处理中心，数据处理中心会显示测得的距离，完全可以实现小车避障的功能。

因为我们想打造一款能够自动避障的智能小车，所以我们要使用超声波传感器作为能识别物体的传感器，目前市面上有很多的超声波传感器，传感器主要的技术指标是精度和

性价比。



我们以一个实例看看如何将步进电机和超声波传感器融合起来，这里我们将使用树莓派组建自导航寻迹小车，并且在下一个章节中我们讲解一下如何利用CNN对着部分代码改写，从而实现自动驾驶的功能。

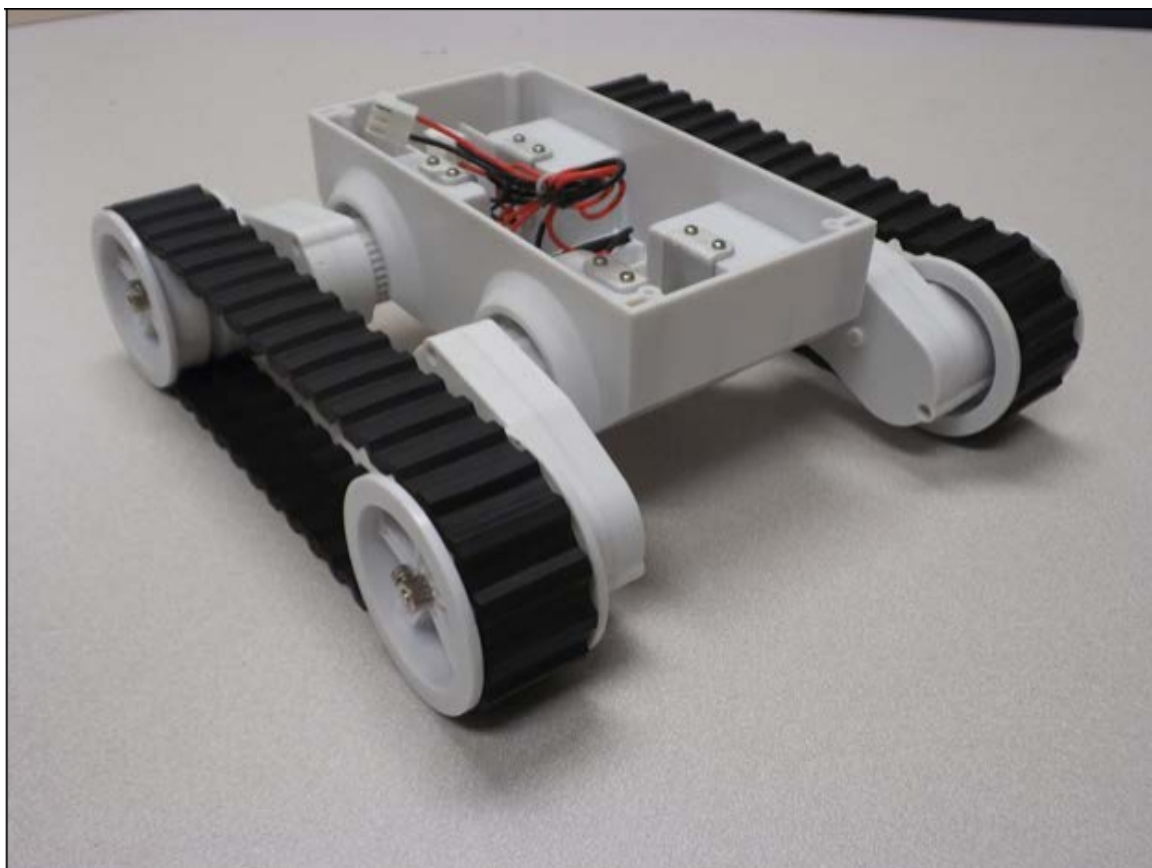
通过阅读本章内容你将学到如下内容：

- 如何使用GPIO接口控制直流电机的速度
- 如何使用树莓派编程控制移动平台
- 如何为寻迹小车规划路线

为了完成这个项目，你必须准备如下的硬件：

- 一个树莓派
- 一个至少8G的是class10的TF卡
- 一个如下图的小车地盘



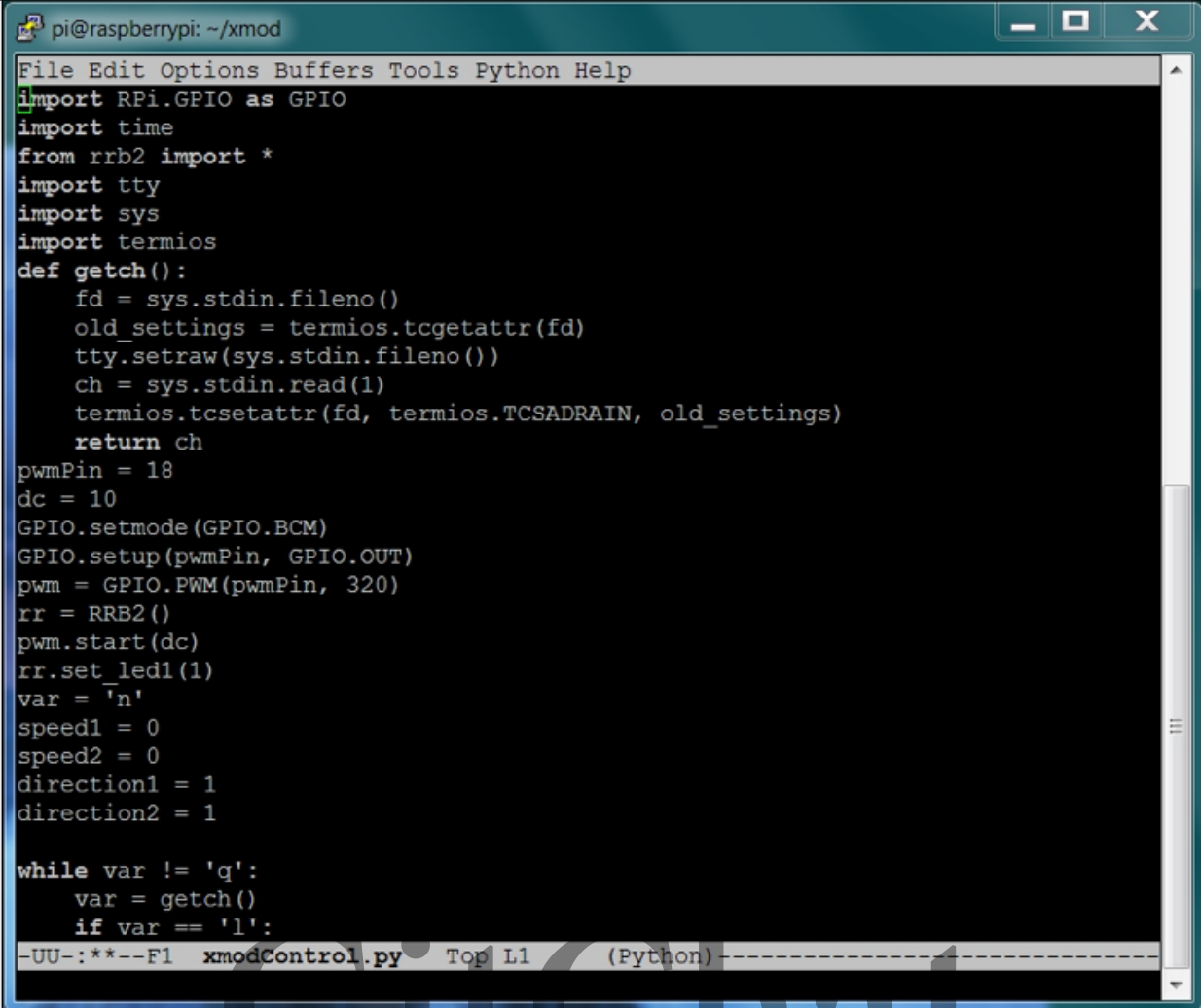


连接电机驱动板的方法如下：

- 将电机驱动板安装在树梅派上。
- 将电源上的电源线连接到驱动板上的电源输入端，电压为6-7v。也可以使用4节AA电池或者2SLiPo锂电池，将地线和电源线连接到点击驱动板上。
- 接下来，将其中一个驱动信号连接到驱动板上电机1的驱动端口上。将电机1连接到右侧电机，电机2连接到左侧。
- 最后将第二个驱动信号链接到驱动板上电机2的驱动端口上。



在树莓派中增加相应的python代码用于驱动电机和超声波传感器。



```
pi@raspberrypi: ~/xmod
File Edit Options Buffers Tools Python Help
import RPi.GPIO as GPIO
import time
from rrb2 import *
import tty
import sys
import termios

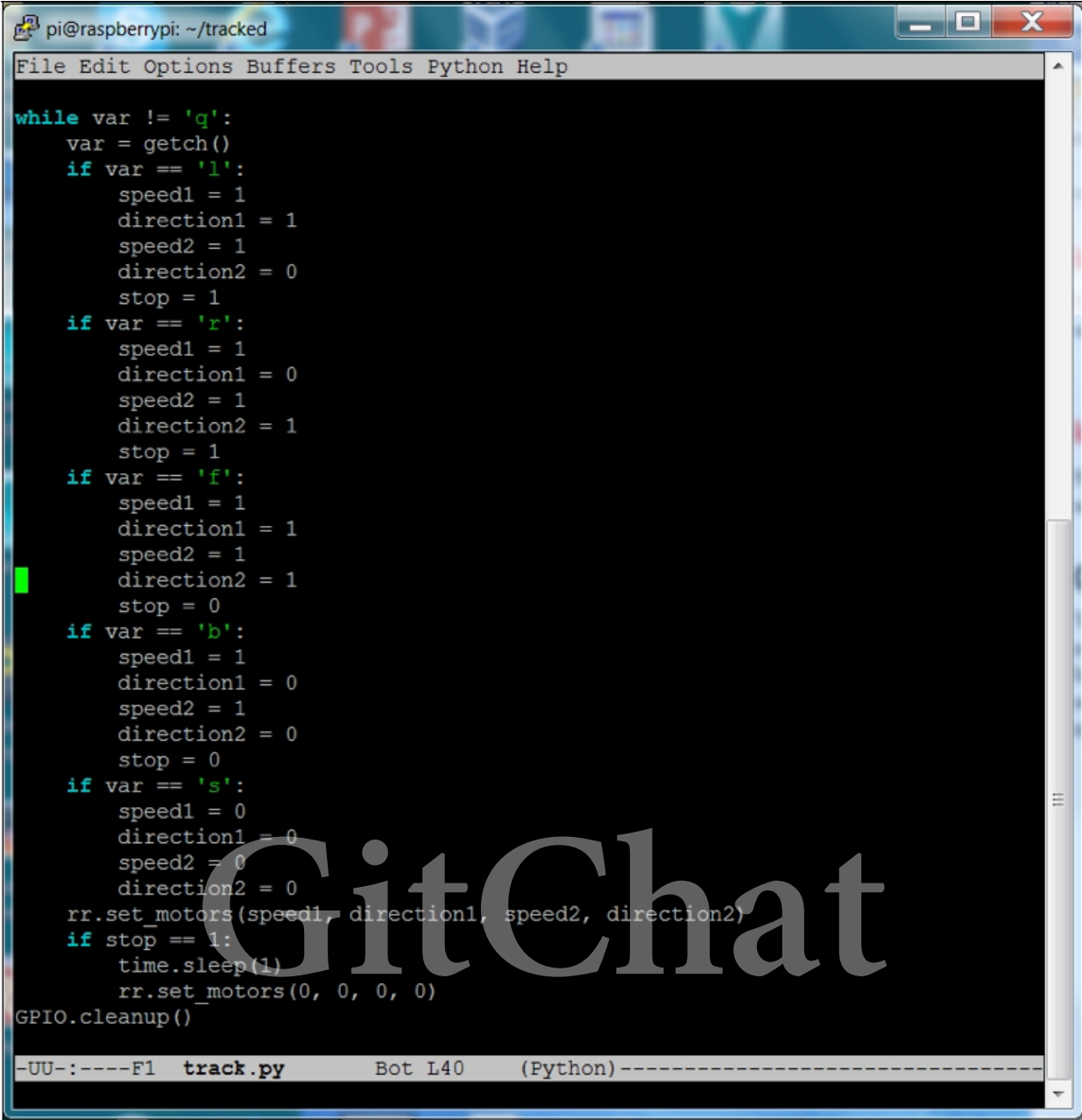
def getch():
    fd = sys.stdin.fileno()
    old_settings = termios.tcgetattr(fd)
    tty.setraw(sys.stdin.fileno())
    ch = sys.stdin.read(1)
    termios.tcsetattr(fd, termios.TCSADRAIN, old_settings)
    return ch

pwmPin = 18
dc = 10
GPIO.setmode(GPIO.BCM)
GPIO.setup(pwmPin, GPIO.OUT)
pwm = GPIO.PWM(pwmPin, 320)
rr = RRB2()
pwm.start(dc)
rr.set_led1(1)
var = 'n'
speed1 = 0
speed2 = 0
direction1 = 1
direction2 = 1

while var != 'q':
    var = getch()
    if var == 'l':
        speed1 = 10
        speed2 = 10
        direction1 = 1
        direction2 = 1
    elif var == 'r':
        speed1 = 10
        speed2 = 10
        direction1 = -1
        direction2 = -1
    elif var == 'f':
        speed1 = 10
        speed2 = 10
        direction1 = 1
        direction2 = 1
    elif var == 'b':
        speed1 = 10
        speed2 = 10
        direction1 = -1
        direction2 = -1
    elif var == 's':
        speed1 = 0
        speed2 = 0
        direction1 = 1
        direction2 = 1
    elif var == 'd':
        speed1 = 0
        speed2 = 0
        direction1 = -1
        direction2 = -1
    elif var == 'q':
        break
    else:
        speed1 = 0
        speed2 = 0
        direction1 = 1
        direction2 = 1

pwm.stop()
rr.shutdown()
GPIO.cleanup()
```

代码的第二部分驱动两个电机，实现对寻迹小车前后进行和转弯的控制，代码如下图所示。



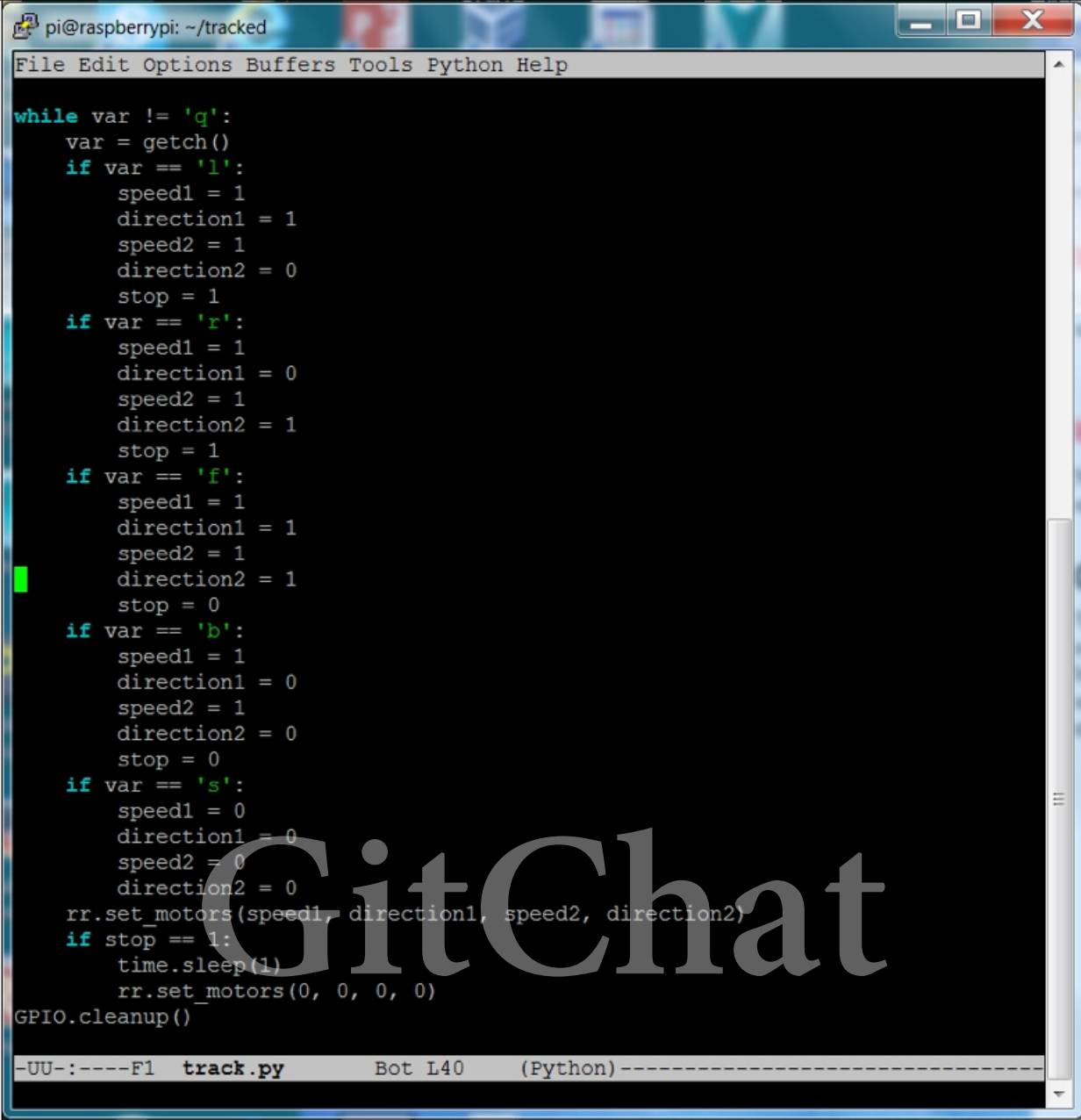
```
pi@raspberrypi: ~/tracked
File Edit Options Buffers Tools Python Help

while var != 'q':
    var = getch()
    if var == 'l':
        speed1 = 1
        direction1 = 1
        speed2 = 1
        direction2 = 0
        stop = 1
    if var == 'r':
        speed1 = 1
        direction1 = 0
        speed2 = 1
        direction2 = 1
        stop = 1
    if var == 'f':
        speed1 = 1
        direction1 = 1
        speed2 = 1
        direction2 = 1
        stop = 0
    if var == 'b':
        speed1 = 1
        direction1 = 0
        speed2 = 1
        direction2 = 0
        stop = 0
    if var == 's':
        speed1 = 0
        direction1 = 0
        speed2 = 0
        direction2 = 0
    rr.set_motors(speed1, direction1, speed2, direction2)
    if stop == 1:
        time.sleep(1)
        rr.set_motors(0, 0, 0, 0)
GPIO.cleanup()

-UU-:----F1 track.py Bot L40 (Python)-----
```

我们通过上述的代码基本清楚了我们需要的代码如何才能驱动相应的电机完成这个工作。



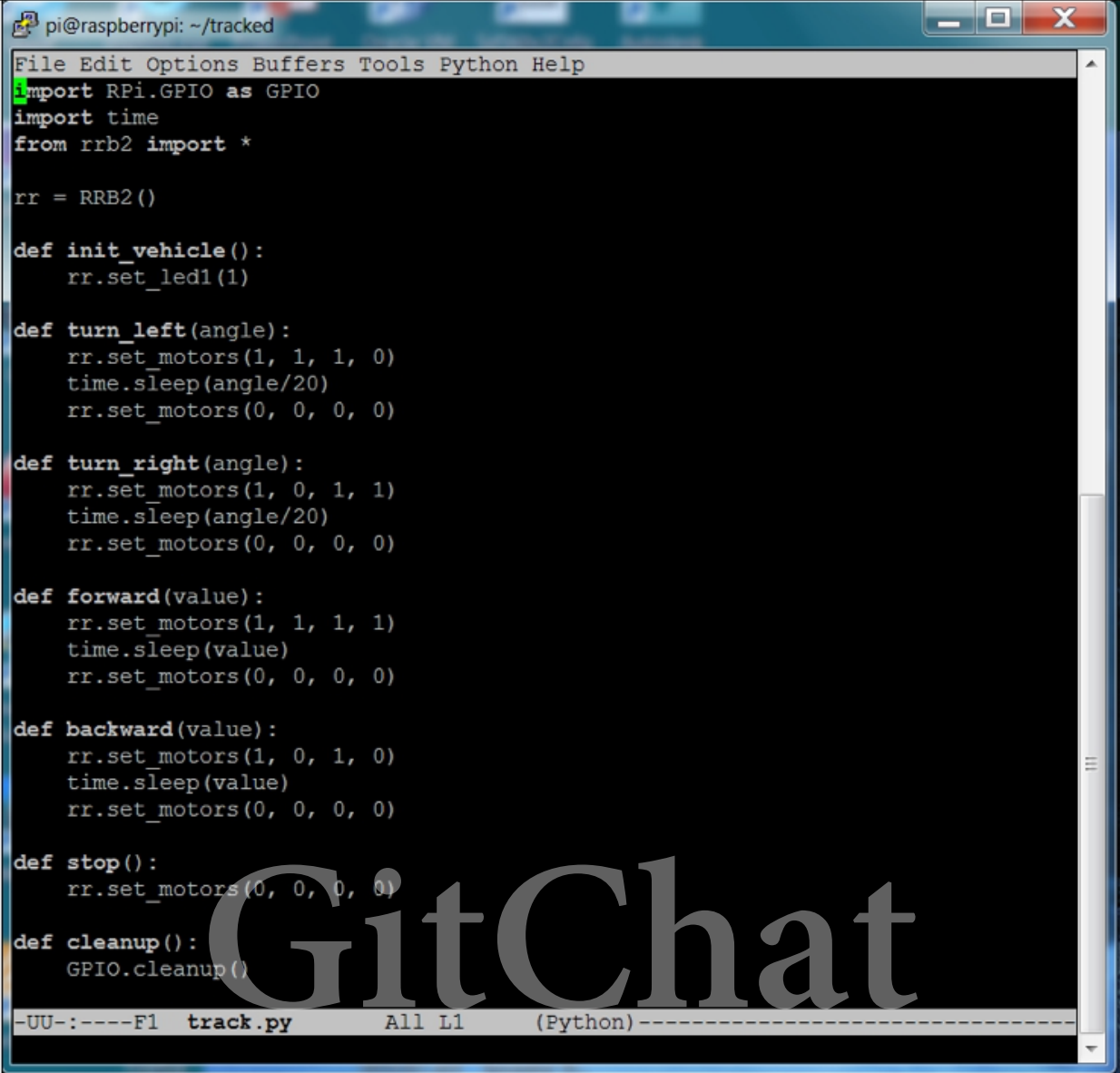


```
pi@raspberrypi: ~/tracked
File Edit Options Buffers Tools Python Help

while var != 'q':
    var = getch()
    if var == 'l':
        speed1 = 1
        direction1 = 1
        speed2 = 1
        direction2 = 0
        stop = 1
    if var == 'r':
        speed1 = 1
        direction1 = 0
        speed2 = 1
        direction2 = 1
        stop = 1
    if var == 'f':
        speed1 = 1
        direction1 = 1
        speed2 = 1
        direction2 = 1
        stop = 0
    if var == 'b':
        speed1 = 1
        direction1 = 0
        speed2 = 1
        direction2 = 0
        stop = 0
    if var == 's':
        speed1 = 0
        direction1 = 0
        speed2 = 0
        direction2 = 0
    rr.set_motors(speed1, direction1, speed2, direction2)
    if stop == 1:
        time.sleep(1)
        rr.set_motors(0, 0, 0, 0)
GPIO.cleanup()

-UU-:----F1 track.py Bot L40 (Python)-----
```

根据之前的说明，`rr.set_motors()`可以实现单独指定每个电机的速度和方向。现在已经有基本的代码实现对寻迹小车的驱动，还需要进一步修改这些代码来实现在其他Python程序中调用这些函数。还需要增加一些标准位移使寻迹小车能够按照指定角度转向或移动一定距离。代码如下图所示：



```
pi@raspberrypi: ~/tracked
File Edit Options Buffers Tools Python Help
import RPi.GPIO as GPIO
import time
from rrb2 import *

rr = RRB2()

def init_vehicle():
    rr.set_led1(1)

def turn_left(angle):
    rr.set_motors(1, 1, 1, 0)
    time.sleep(angle/20)
    rr.set_motors(0, 0, 0, 0)

def turn_right(angle):
    rr.set_motors(1, 0, 1, 1)
    time.sleep(angle/20)
    rr.set_motors(0, 0, 0, 0)

def forward(value):
    rr.set_motors(1, 1, 1, 1)
    time.sleep(value)
    rr.set_motors(0, 0, 0, 0)

def backward(value):
    rr.set_motors(1, 0, 1, 0)
    time.sleep(value)
    rr.set_motors(0, 0, 0, 0)

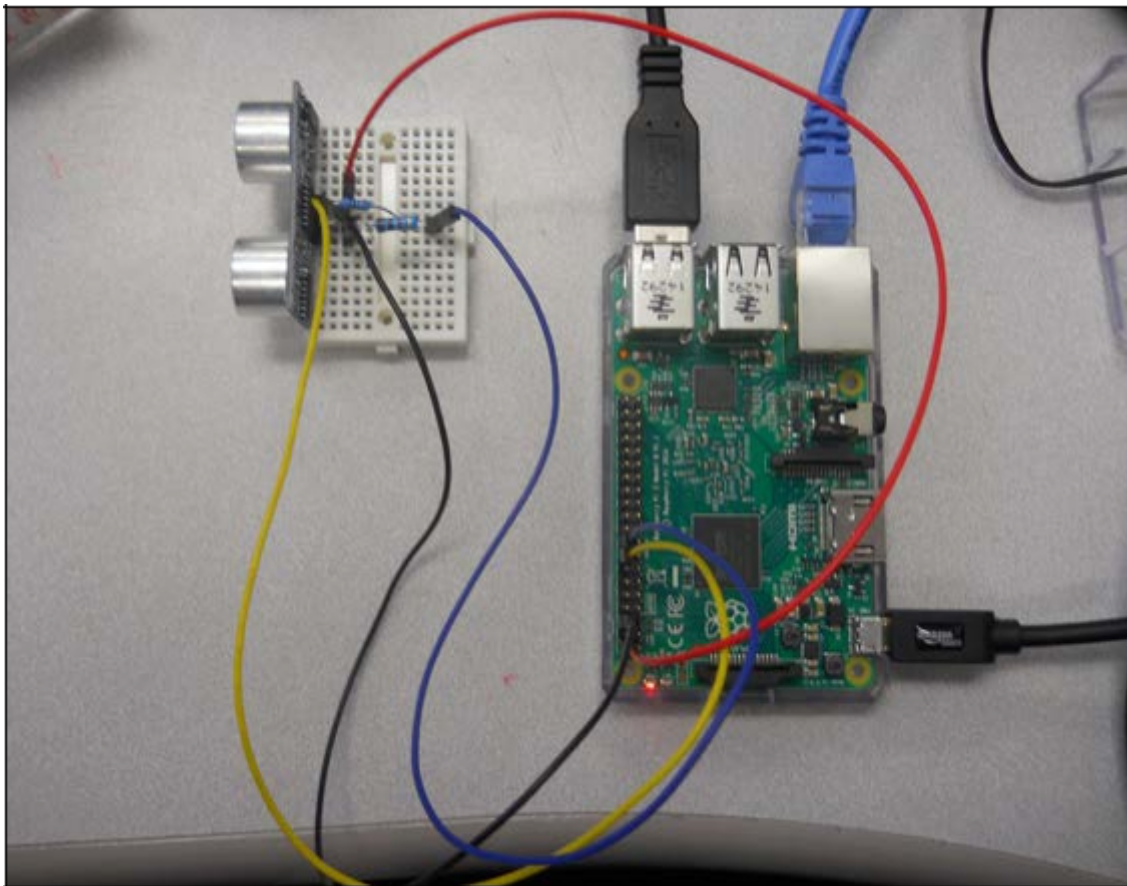
def stop():
    rr.set_motors(0, 0, 0, 0)

def cleanup():
    GPIO.cleanup()

--UU-:----F1  track.py  All L1  (Python)-----
```

在函数 `turn_right(angle)` 和 `turn_left(angle)` 中指令 `time.sleep(angle/20)` 使寻迹小车按照转向角度大小运动一定的时间达到指定的转向角度。根据实际情况可能需要修改一下分母20的大小来实现准确的转向。指令 `time.sleep(value)` 使寻迹小车根据 `value` 值大小运动一定的时间来实现运动指定距离。接下来为了实现寻迹小车的运动，需要将传感器连接到寻迹小车上以便让小车知道其附近的情况。

最后，将传感器连接到树莓派上，通过面包板对树莓派的软件进行测试，电路图如下：



当传感器连接完成后，需要一段代码来读取传感器返回的数值，先将传感器固定（在静态测试情况下），然后程序转换成距离，下图是程序的Python代码。

```
pi@raspberrypi: ~  
File Edit Options Buffers Tools Python Help  
import RPi.GPIO as GPIO  
import time  
GPIO.setmode(GPIO.BCM)  
  
trig_pin = 23  
echo_pin = 24  
GPIO.setup(trig_pin, GPIO.OUT)  
GPIO.setup(echo_pin, GPIO.IN)  
  
GPIO.output(trig_pin, False)  
print "Waiting to settle"  
time.sleep(1)  
GPIO.output(trig_pin, True)  
time.sleep(0.00001)  
GPIO.output(trig_pin, False)  
  
while GPIO.input(echo_pin)==0:  
    start = time.time()  
  
while GPIO.input(echo_pin)==1:  
    end = time.time()  
  
duration = end - start  
distance = duration * 17150  
distance = round(distance, 2)  
print "Distance:", distance, "cm"  
GPIO.cleanup()  
  
--UU--:----F1 sonar sensor.py All L1 (Python)-----  
For information about GNU Emacs and the GNU system, type C-h C-a.
```

然后运行这部分代码，我们就看到了超声波传感器测距的结果了。

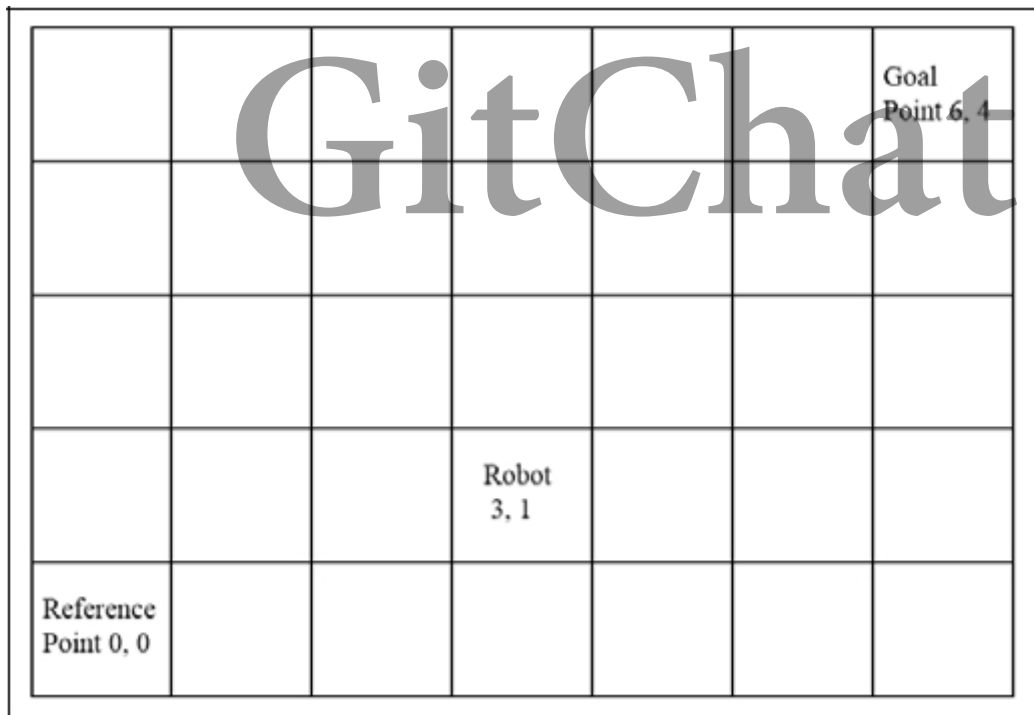
通过上述的代码我们基本上清楚了如何通过树莓派控制电机、也知道了我们如何控制树莓派，通过超声波传感器的数据反馈给树莓派控制程序，然后树莓派做出决策，并将这个决策再次反馈给驱动电机，电机做出正转或者反转的相应，此时小车是前进或者后退。

通过上面的介绍我们了解到我们要使用手动去控制小车的前进或者后退，那么小车如何选择路劲呢？下面就重点讲讲设置动态规划路劲。

自动规划路线意味着在遇到障碍之前，不可能完全预知障碍的存在。你的设备需要自行决定运行过程中如何前进。这是一个复杂的问题，但是如果想要你的设备自如地在环境中运行，有几个基本的概念需要了解和应用。让我们先来解决当你知道要让设备运行到哪里时的线路规划问题，然后再在线路上加一些障碍物。

## 基本线路规划

为了学习动态线路规划，也就是在预先不知道障碍物存在时的线路规划问题，需要通过下面这个框架来了解设备的位置以及设备要运行到的目标位置。

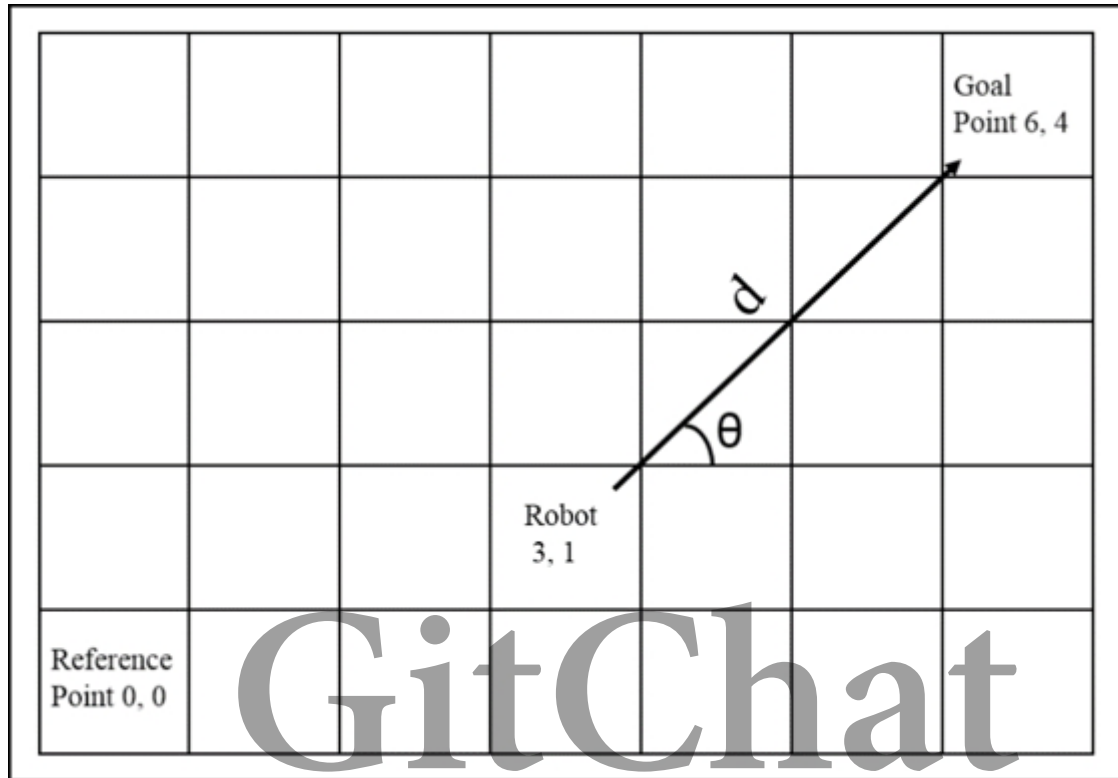


**这个网格有三个关键位置，具体解释如下：**

- 左下角的点是一个固定参照位置，x轴和y轴的方向也是固定的，其他点的位置都可以根据该参照点和x轴、y轴确定。
- 第二个关键位置是机器人的起点。寻迹小车会根据x坐标和y坐标保持自身相对于x轴、y轴上直到目标位置为止的一些固定参考点位置的轨迹。它会使用指南针来跟踪这些方向。

- 第三个关键位置是目标位置，目标位置也会用X轴和Y轴上相对于固定参照位置的坐标来表示。如果知道起点以及起点和目标位置之间的角度，就可以规划到达目标位置的最优的（距离最短）路线。可以根据目标位置、寻迹小车的位置和一些简单的数学公式来计算寻迹小车和目标位置之间的距离和角度。

对于路径规划有一个公式处理，大家查阅几何书籍就能找到相关的数学公式，我们用一个示意图进行表示。



我们可以用一段很简单的python代码来实现以上操作，让轨迹小车向前运动和转向，我们将这个文件命名为robotLib.py的文件，它包括了所有伺服初始化设定的使寻迹小车向前运动或转向的程序。然后使用 `from compass import *` 这行代码调用路径规划的程序和管理，我们同样通过这行代码将指南针程序导入其中，完整的代码如下：



```

pi@raspberrypi: ~/tracked
File Edit Options Buffers Tools Python Help
#!/usr/bin/python
import time
from track import *
import math

xpos_robot = int(raw_input("Robot X Position: "))
ypos_robot = int(raw_input("Robot Y Position: "))
xpos_goal = int(raw_input("Goal X Position: "))
ypos_goal = int(raw_input("Goal Y Position: "))

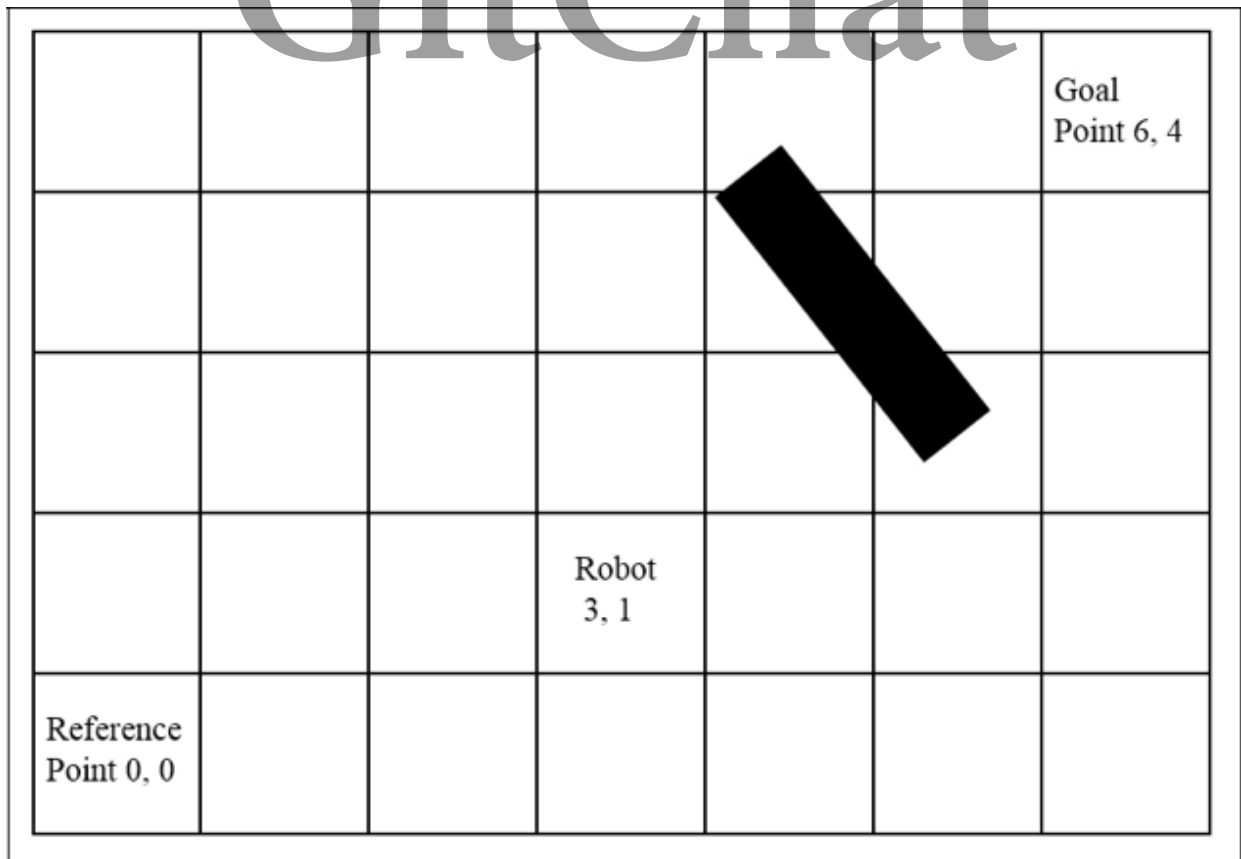
distance = math.sqrt((xpos_goal - ypos_robot)**2 + (ypos_goal - ypos_robot)**2)
angle = round(math.degrees(math.atan2((ypos_goal - ypos_robot), (xpos_goal - xpos_robot))))
if angle < 0:
    angle = angle + 360
print (angle)
# Turn to the right bearing
if (angle) < 180:
    turn_right(angle)
else:
    turn_left(angle)
print (distance)
forward(distance)

--UU-:----F1  robotGoal.py  All L1  (Python)-----
For information about GNU Emacs and the GNU system, type C-h C-a.

```

## 壁障

上面我们讲述了无障碍路径规划，是比较简单的，但是当寻迹小车需要绕过障碍物时，规划路径就具有更大的挑战性，假如一个障碍物位于之前所规划的路径之上，具体如图所示。



仍然可以使用同样的计算方法获得起始的行进角度；但是，现在需要使用声纳传感器去检测障碍物。当声纳传感器检测到障碍物，寻迹小车就需要停下来并且重新计算避开障碍物的路径和到达目标位置的路径。一个很简单的方法是，当寻迹小车发现障碍物，它

向右转90度，向前行进一段距离，然后再计算最优的路径。当再次转到向目标位置运动的方向后，如果没有障碍物，寻迹小车将会沿着最优路线前进。

为了检测障碍物，需要调用传感器的库函数。可以使用指南针来更加精确的确定前进角度，使用`from compass import *`命令可以导入指南针的库函数来使用指南针。也可以使用时间的库函数和`time.sleep`命令来控制不同指令执行的时长。需要修改`track.py`的库函数，使整个命令组不会有一个固定的结束时间，如下图所示：



```
pi@raspberrypi: ~/tracked
File Edit Options Buffers Tools Python Help

xpos_robot = int(raw_input("Robot X Position: "))
ypos_robot = int(raw_input("Robot Y Position: "))
xpos_goal = int(raw_input("Goal X Position: "))
ypos_goal = int(raw_input("Goal Y Position: "))

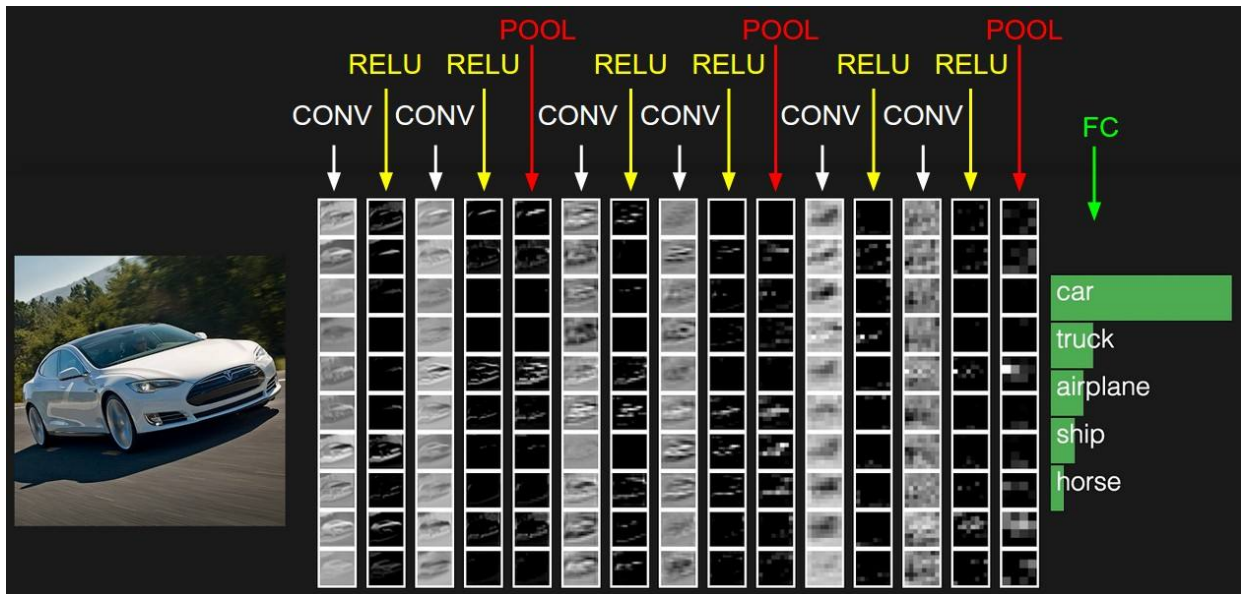
distance, angle = positionRobot(xpos_robot, ypos_robot, xpos_goal, ypos_goal)

start_time = time.time()
forward()
barrier = rr.get_distance()
elapsed_time = 0

while barrier > 10 and elapsed_time < distance:
    elapsed_time = time.time() - start_time
    barrier = rr.get_distance()
    if barrier > 0 and barrier < 10:
        print "barrier", barrier
        distance_traveled = elapsed_time
        new_distance = 1
        ypos_robot = ypos_robot + distance_traveled * math.sin(math.radians(angle))
        ypos_goal_barrier = ypos_robot + new_distance * math.sin(math.radians(angle + 90))
        xpos_robot = xpos_robot + distance_traveled * math.cos(math.radians(angle))
        xpos_goal_barrier = xpos_robot + new_distance * math.cos(math.radians(angle + 90))
        distance = positionRobot(xpos_robot, ypos_robot, xpos_goal_barrier, ypos_goal_barrier)
        start_time = time.time()
        forward()
        elapsed_time = 0
        while elapsed_time < new_distance:
            elapsed_time = time.time() - start_time
            print "Done moving around barrier"
            ypos_robot = ypos_goal_barrier
            xpos_robot = xpos_goal_barrier
            distance = positionRobot(xpos_robot, ypos_robot, xpos_goal, ypos_goal)
            start_time = time.time()
            forward()
            barrier = rr.get_distance()
            elapsed_time = 0
        stop()
    print "Goal Reached"
-UU-:----F1 robotBarrier.py 40% L57 (Python)-----
```

通过以上的讲解我们了解了如何进行路径规划，但是我们想要一个能够自动驾驶的智能小车改如何做呢？在第四章我们就重点强调如何通过卷积神经网络去训练一个自动驾驶的模型。

## 4. 使用CNN作为智能小车自动驾驶系统



卷积神经网络 (Convolutional Neural Network, CNN) 是一种适合使用在连续值输入信号上的深度神经网络, 比如声音、图像和视频。CNN在无人驾驶3D感知与物体检测中的有着广泛的应用。

2016年3月, Mohammad Rastegari 等人在论文中首次提出了 XNOR-Net 的概念。这篇论文旨在利用二值化操作寻找到最优的简化网络, 并分别介绍了两种有效的网络: Binary-Weight-Networks 和 XNOR-Networks。Binary-Weight-Networks 是对 CNN 中所有的权重做近似二值化, 可以节省 32 倍的存储空间。而且, 由于权重被二值化, 卷积过程只剩加减算法, 不再包括乘法运算, 可以提高约两倍的运算速度, 这促使 CNN 可以在不牺牲准确率的情况下在小存储设备上使用, 包括便携式设备。

XNOR-Networks 算法则是对 CNN 中所有的权重和输入同时做近似二值化, 如果卷积运算中的所有操作数都是二进制的, 那么两个二进制向量的点乘就可以等同于同或运算和位运算。而这些操作天然就被 CPU 等通用计算设备支持, 所以二值化神经网络能够跑在普通的 CPU 和更便宜的 ARM 芯片甚至是树莓派等设备上。这就为 CNN 移植到树莓派系统提供了理论基础。

树莓派智能小车感知检测起点与终点之间的地理环境和周围变化, 进而选择最优路线, 躲避危险, 就可以真正实现智能小车的自动驾驶。拥有更广阔的发展前景。