目录

南 京 大 学
实 验 报 告

学生姓名：丛进　　指导教师：方元

实验地点：仙林校区基础实验楼嵌入式教室

实验时间：第 15-17 周周四 18：30-20：30,1 月 5 日，1 月 6 日

实验学时: 2

实验名称：避障循迹遥控小车

一、 实验目的

1） 拼装小车
2） 自动避障
3） 循迹
4） 遥控器遥控

二、 实验器材

2.1 树莓派

树莓派这类计算机结构简单、体积小、耗电低, 却拥有与普通计算机几乎相同的功能和性能, 可以很方便地植入各种应用系统中。这类单板计算机也是典型的嵌入式系统的基础。本次实验用到的树莓派 3B/3B+，带有一个以太网接口、4 个 USB host、1 个无线网接口和蓝牙接口。片内大量 I/O 接口通过一组 2*20 引脚引出, 作为扩展设备控制接口。引脚功能见图 1。

2.2 Alphabot2 小车

拼装采用底部超声波，顶部安放树莓派的方式。最终成果见图 8图 9图 10

Item	GPIO	Item	GPIO
Buzzer	GPIO4	Joystick	GPIO7–11
IR sensors (Bottom)	GPIO5(CS)	Wheel A	GPIO12–13
(TLC1543)	GPIO25(CLK)		GPIO6 (CTRL)
	GPIO24(ADDR)	Wheel B	GPIO20–21
	GPIO23(DOUT)		GPIO26(CTRL)
IR remoter recv.	GPIO17	Servo (I2C)	GPIO2(SDA)
Color LEDs (Bottom)	GPIO18	(PCA9685)	GPIO3(SCL)
UltraSonic	GPIO22(Trig)	IR (Front-R)	GPIO19
	GPIO27(Echo)	IR (Front-L)	GPIO16

图 1: Alphabot2 资源分配表

三、 实验原理

3.1 传感器

3.1.1 超声波传感器——避障原理

超声波硬件见图 3，测距模块原理见图 2。Trig 端口产生不短于 10ms 的正脉冲, 模块会自动发出 8 个周期的超声脉冲信号 (40kHz), 并在 Echo 端输出高电平。当检测到回波时将 Echo 回置到低电平。通过测量 Echo 高电平维持周期, 再根据声波速度, 就可以算出发射器到障碍物之间的距离。

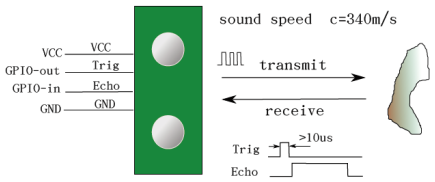

图 2: 超声波测距原理

图 3: 超声波硬件

3.1.2 蜂鸣器

蜂鸣器是一种一体化结构的电子发声设备，主要有压电式和电磁式两种类型。多谐振荡器通电源后输出 1.5 2.5kHz 的音频信号，推动压电蜂鸣片发声。

蜂鸣器的 GPIO 口输入为高电平后，开始发声音。修改为低电平后，停止发声。

3.1.3 前端红外传感器——避障原理

硬件见图 4，小车前部左右各一个。红外传感器由一个红外发射管和一个红外接收管组成。发射管和接收管之间有隔离板。遇到障碍物时，发射管发出的红外光，被物体反射后由接收管接收, 有障碍时输出低电平，没有时输出高电平。

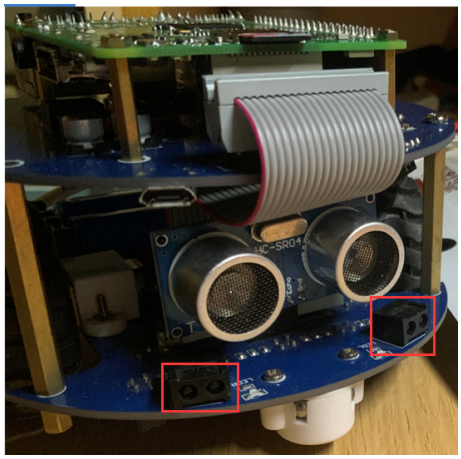

图 4: 2 个光照传感器

3.1.4 底部红外传感器——循迹原理

硬件见图 5，小车底部有 5 个。与前端的红外传感器只能输出 0-1 不同，底部传感器经过模数转换器得到量化的红外强度值，即输出 0-999 的具体强度值。接收强度与传播距离和反射面对红外光的吸收性质有关。在低速模式下前进。高频检测 5 个探测器的光强，如果有一侧光强和其他差距太多，则进行超微小角度微调方向。

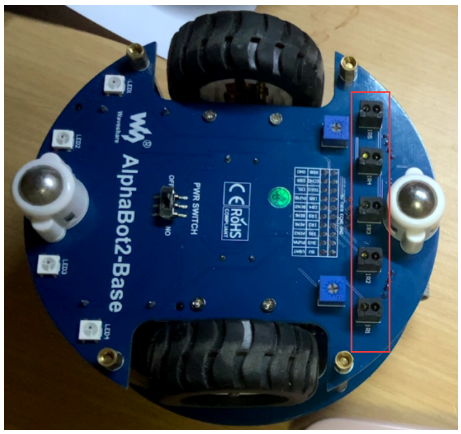

图 5: 底部 5 个光照传感器

3.2 彩灯

每一个彩灯灯珠是一个三色 LED, 通过红、绿、蓝三个发光二极管产生的不同亮度形成不同的颜色。每个发光二极管由一个 8 位的数字信号控制, 实现 256 级亮度。数字信号以串行方式输入, 多个灯珠串接, 形成一个灯带。灯带的数字信号控制时间在微秒以下, 对时钟要求比较严格, 普通应用程序从软件上很难满足要求, 必须通过硬件实现。rpi_ws281x 模块提供了 Python 接口。

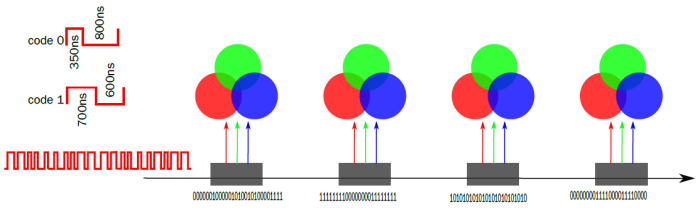

图 6: 彩灯控制原理

3.3 遥控器

红外遥控器发射的信号使用 38kHz 左右的载波对基带进行调制。接收端对信号监测、放大、滤波、解调等等一系列处理, 然后输出基带信号。收发方采用约定的协议进行通信。图 7是 NEC 红外通信协议发送信号的波形。发送端首先发送一个 9ms 低电平接 4.5ms 高电平的引导码, 接收方检测到引导码后开始识别后面的数据：0.56ms 低 +0.56ms 高表示“0”, 0.56ms 低 +1.69ms. 高表示“1”。一组 0、1 序列构成一个按键特征字。接收方根据收到的特征字产生一定的动作。

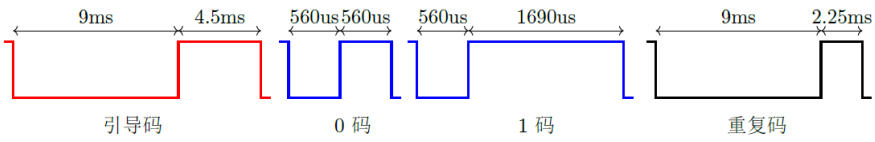

图 7: NEC 红外遥控器协议

3.4 直流电机车轮——避障原理

小车拥有两个独立的直流电机，分别控制左侧和右侧的两个车轮。可以分别设置顺时针或者逆时针转动。不采用后退方式。

3.4.1 前进

没有障碍物，一直直线前进。速度可通过命令行在运行中调节。

3.4.2 转向

1. 采用原地转圈方案。每次转向 30° 左右。使用该方案的原因是，无法保证小车周围是否有足够空间进行后退转向，或者非对称轴偏移转向。在且低速状态下，一个轮子不转，另一个轮子有可能转不动。

2. 超声波和红外共同检测障碍物。仅使用红外会因为材质原因撞上部分物体，比如黑色的门。仅使用超声波，会因为只能探测直线，而导致测着撞到障碍物。两者结合可以保证几乎 100% 不撞到障碍物。

当红外探测到左边有障碍物 → 右转 (左侧轮子前转，右侧轮子后转)。

当红外探测到右边有障碍物 → 右转 (左侧轮子后转，右侧轮子前转)。

当红外同时探测到障碍物，或者只有超声波探测到障碍物，随机左右转。

随机的转圈是为了进入几乎环形的死角有机会出来。

3.5 交互控制方式

树莓派的有静态 Ip 地址 192.168.208.132，电脑与树莓派之间通过 ssh 链接控制。通过命令 ssh root@192.168.208.132 进行链接，链接完成后进入树莓派命令行页面

传输代码文件使用 scp 命令。文件夹上传至树莓派：scp -r car root@192.168.208.132: 。-r 表示递归文件夹其中 car 时本项目控制小车所有代码文件所在的文件夹。

四、 实验内容

4.1 模块化编程

超声波探测、红外探测、彩灯、蜂鸣器等功能分离成独立的、可相互改变的“模块”，使得每个模块都包含着执行预期功能的一个唯一方面。

由于 python 没有指针概念，所以没法实现跨文件变量共享。这给多线程操作全局变量带来麻烦。于是在 global_var.py 维护了一个字典 _global_dict。其他文件在使用时 import global_var. 对 _global_dict 进行 get 和 set 操作。

代码 1: 全局字典

```
1 '''
2 没有指针机制，没法实现多线程需要的跨文件全局变量。维护一个字典
3 '''
```

```
4
5  global _global_dict
6  _global_dict = {'d':100,
7                  'speed':10,
8                  'buzzer_switch':0,
9                  'led_switch':0
10                 }
11
12
13 def  set(key, value):
14     _global_dict[key] = value
15
16
17 def get(key):
18     return _global_dict.get(key,None)
19
20 def get_items():
21     return _global_dict.items()
```

4.2 超声波探测

用超声波硬件探测直线精确距离。

代码 2: 超声波探测

```
1  import RPi.GPIO as GPIO
2  import time
3  import global_var
4
5  GPIO.setmode(GPIO.BCM)  # GPIO.BOARD
6  Trig=22  # tirg links out
7  Echo=27  # echo links in
8  GPIO.setup(Trig,GPIO.OUT)
9  GPIO.setup(Echo,GPIO.IN)
10 def distance():
11     while 1:
12         GPIO.output(Trig,GPIO.HIGH)
13         time.sleep(0.08) # sleep时间太短，d一直是同一个数字
14         GPIO.output(Trig,GPIO.LOW)
15         while GPIO. input(Echo)==GPIO.LOW:
16             pass
17         t1=time.time()
18         while GPIO. input(Echo) == GPIO.HIGH:
19             pass
20         t2=time.time()
21         global_var. set('d',(t2-t1)*34000/2)
```

4.3 前端红外探测

主要用于探测左右有无障碍物。灵敏度调节到 10cm 附近有障碍物时改变电平 (手掌测试结果，具体距离视障碍物材质)

代码 3: 红外探测障碍物

```
import RPi.GPIO as GPIO
import time
import global_var

'''
灵敏度：往中间旋转会变得灵敏，远距离就能探测到
'''
R =19  # 右侧感光二极管探测器,有东西就变成0
L=16
GPIO.setmode(GPIO.BCM)
GPIO.setup(R,GPIO.IN,pull_up_down=GPIO.PUD_UP)
GPIO.setup(L,GPIO.IN,pull_up_down=GPIO.PUD_UP)

def detect():
    while 1:
        global_var. set('Rd',GPIO. input(R))
        global_var. set('Ld', GPIO. input(L))
        time.sleep(0.08)
```

4.4 底部红外探测

没有黑胶布，所以只有例论代码

代码 4: 底部红外探测

```
'''
底部5个光照强度探测仪
'''
import RPi.GPIO as GPIO
import time
import move

CS = 5
CLOCK = 25
ADDRESS = 24
DATAOUT = 23
GPIO.setmode(GPIO.BCM)
GPIO.setup ((CS , CLOCK , ADDRESS), GPIO.OUT)
GPIO.setup(DATAOUT , GPIO.IN , GPIO.PUD_UP)
def AnalogRead ():
     value = [0]*(6)
     #Read Channel0~channel5 AD value
     for j in  range(0, 6):
          GPIO.output(CS , GPIO.LOW)
          for i in  range(0, 10):
               #sent 4-bit Address
               if (i < 4):
                    bit = (((j) >> (3 - i)) & 0x01)
                    GPIO.output(ADDRESS , bit)

               #read 10-bit data
               value[j] <<= 1
               value[j] |= GPIO. input(DATAOUT)
               GPIO.output(CLOCK , GPIO.HIGH)
               GPIO.output(CLOCK , GPIO.LOW)

```

```
32          GPIO.output(CS , GPIO.HIGH)
33          time.sleep (0.0001)
34      return value [1:] # invalid address for channel 0
35
36  # pwr是数组后面
37  while True:
38      d=AnalogRead ()
39      left= sum(d[:2])
40      right= sum(d[3:])
41
42      if(left<right):
43          move.turnright()
44          print('left')
45      else:
46          move.right()
47          print('right')
48      time.sleep(0.5)
```

4.5 移动

速度可以在小车移动时通过命令行修改

移动策略见实验原理直流电机部分

代码 5: 移动方法

```
1   import RPi.GPIO as GPIO
2   import time
3   import random
4   import global_var
5   l1 = 12
6   l2 = 13
7   l_control = 6
8
9   r1=20# pwr指示灯那边,右侧
10  r2=21
11  r_control=26
12  GPIO.setmode(GPIO.BCM)
13  GPIO.setup((l1, l2, l_control), GPIO.OUT)
14  GPIO.setup((r1, r2, r_control), GPIO.OUT)
15  l_speed = GPIO.PWM(l_control, 1000)
16  r_speed = GPIO.PWM(r_control, 1000)
17
18  def right(speed,reverse=0):
19      # 右轮子转
20      GPIO.output((r1, r2), (0,1))
21      if reverse:
22          GPIO.output((r1, r2), (1,0))
23      l_speed.start(speed)
24
25
26  def left(speed,reverse=0):
27      GPIO.output((l1, l2), (0,1))
28      if reverse:
29          GPIO.output((l1, l2), (1,0))
30      r_speed.start(speed)
31
```

```
32
33
34  def forward(speed):
35      # 右边轮子动力好像差一点
36      right(speed)
37      left(speed)
38
39
40
41  def backward(speed,t):
42      l_speed.stop()
43      r_speed.stop()
44
45      right(speed,1)
46      left(speed,1)
47      if t==-1:
48          t=999999
49      time.sleep(t)
50      l_speed.stop()
51      r_speed.stop()
52
53
54  def turnRight(speed=10):
55      right(speed,1)
56      left(speed)
57
58      time.sleep(0.3)
59      l_speed.stop()
60      r_speed.stop()
61
62
63  def turnLeft(speed=10):
64      right(speed)
65      left(speed,1)
66
67      time.sleep(0.35)
68      l_speed.stop()
69      r_speed.stop()
70
71  def stop():
72      # 立刻停下
73      l_speed.stop()
74      r_speed.stop()
75
76  def AIturn(speed,d,Rd,Ld,danger_d):
77      if d<danger_d or (Rd==0 and Ld==0):
78          choice = random.randint(0, 1)
79          if choice:
80              turnLeft(speed)
81          else:
82              turnRight(speed)
83      if d<danger_d or (Rd==1 and Ld==0):
84          # 左侧有东西
85          turnRight(speed)
86          print('turnRight!')
87      if d<danger_d or (Rd==0 and Ld==1):
88          turnLeft(speed)
89          print('turnLeft!')
90
```

```
def readSpeed():
    # 用于运行时修改车速
    while 1:
        s= int( input())
        if 0<=s<=100:
            global_var. set('speed',s)
```

4.6 LED 变色灯光

彩灯可以显示全白，也可以五颜六色。具体代码比较长，限于文章篇幅，可在仓库中查看源代码 https://github.com/Ironstarboy/RP3b/blob/master/car/LED.py。展示效果可以看视频。

4.7 蜂鸣器

遇到障碍物时，设置了警报。

代码 6: 移动方法

```
'''
蜂鸣器
'''
import RPi.GPIO as GPIO
import time
import global_var

GPIO.setmode(GPIO.BCM)
buzzer=4
GPIO.setup(buzzer,GPIO.OUT)

def ring(danger_d=15,t=0.1):
    while 1:
        if global_var.get('d')<danger_d:
            GPIO.output(buzzer,1) #GPIO.HIGH,高电压响
            time.sleep(t)
            GPIO.output(buzzer,0)

def slience():
    GPIO.output(buzzer, 0)
    print('buzzer off')
```

4.8 命令行操作

可以初始速度设置，决定是否打开蜂鸣器功能、彩灯功能

参数解释

1） -s –speed，设置小车速度。并检验是否是 0-100 的合法值。运行时可以改变速度。

2） -b –buzzer 控制蜂鸣器开关。1 为开，0 为关

3） -l –led 控制彩灯开关。1 为开，0 为关

代码 7: 运行时命令行操作

```
# https://geek-docs.com/python/python-tutorial/python-argparse.html
import run
import global_var
import argparse
import time
import threading
import buzzer



parser = argparse.ArgumentParser(description='car control')
parser.add_argument('-s','--speed', type= int,default=10, help="set inital car speed")
parser.add_argument('-l','--led',default=0, help='set led on or off')
parser.add_argument('-b','--buzzer',default=0, help='set buzzer on or off')
args = parser.parse_args()

if args.led:
    global_var. set('led_switch',1)
    print('led state = {}'. format(args.led))

if args.buzzer:
    global_var. set('buzzer_switch',1)
    print('buzzer state = {}'. format(args.buzzer))

if args.speed:
    global_var. set('speed',args.speed)
    r = threading.Thread(target=run.run)
    r.start()
    print('current speed = {}'. format(global_var.get('speed')))
```

4.9 多线程

采用多线程技术，以实现距离探测、蜂鸣器警报、LED 灯效、前进并行运算。

代码 8: 多线程运行

```
import move
import buzzer
import time
import threading
import LED
import RPi.GPIO as GPIO
import sonic
import global_var
import light
import argparse



danger_d=15 # 单位厘米，距离内蜂鸣+转弯

threading.Thread(target=sonic.distance).start()
buz=threading.Thread(target=buzzer.ring) # args=(danger_d,0.5)
```

```
threading.Thread(target=light.detect).start()
led=threading.Thread(target=LED.run)
threading.Thread(target=move.readSpeed).start()  # 运行中改变次小车速度
# GPIO.output(4,0)#ctrl c退出，会导致ring没来得及执行变成低电平代码

def run():
    if global_var.get('buzzer_switch'):
        buz.start()
        print('buzzer on!')
    if global_var.get('led_switch'):
        led.start()
        print('led on!')
    while 1:
        d = global_var.get('d')  # 不停读取正前方距离d
        R_detected = global_var.get('Rd')  # Rd=0即右前方有东西
        L_detected = global_var.get('Ld')
        move.forward(global_var.get('speed'))
        move.AIturn(10,d,R_detected,L_detected,danger_d)
```

4.10 遥控器

由于拿到的盒子里并没有遥控器，仅在第 17 周借用同学遥控器实现了简单的代码

代码 9: 遥控器

```
import RPi.GPIO as GPIO
import time
ir=17
GPIO.setmode(GPIO.BCM)
GPIO.setup(ir,GPIO.IN,GPIO.PUD_UP)
keymap={
  0x45:'CH-',0x46:'CH ',0x47:'CH+',
  0x44:'<<<',0x40:'>>>',0x43:'>>|',
  0x47:' - ',0x15:' + ',0x09:'EQ ',
  0x07:' 0 ',0x19:'100+',0x0D:'200+',
  0x0C:' 1 ',0x18:' 2 ',0x5E:' 3 ',
  0x08:' 4 ',0x1C:' 5 ',0x5A:' 6 ',
  0x42:' 7 ',0x52:' 8 ',0x4A:' 9 '}

def getkey():
  if GPIO. input(ir) == GPIO.HIGH:
    return
  channel = GPIO.wait_for_edge(ir,GPIO.RISING,timeout=8)
  GPIO.remove_event_detect(ir)
  if channel is not None:
    return

  time.sleep(0.0045)

  data = 0
  for shift in  range(0,32):
    while GPIO. input(ir) == GPIO.LOW:
      time.sleep(0.0001)

    count = 0
```

```
    while GPIO. input(ir) == GPIO.HIGH and count < 10:
      count += 1
      time.sleep(0.0005)

    if(count > 1):
      data |=1<<shift

  a1 = (data >> 24) & 0xff
  a2 = (data >> 16) & 0xff
  a3 = (data >> 8) & 0xff
  a4 = (data) & 0xff
  if ((a1+a2) == 0xff) and ((a3+a4) == 0xff):
    return a2
  else: print("repeat key")

print('irremote test start ...')

while True:
  key = getkey()
  if(key != None):
    print('key = ',keymap[key])



A=13
B=12
CONTROL=6
GPIO.setmode(GPIO.BCM)
GPIO.setup((A,B,CONTROL),GPIO.OUT)
speed=GPIO.PWM(CONTROL,1000)
GPIO.output((A,B),(1,0))

f= open('/dev/input/event0','rb')

while True:
  d=f.read(48)
  key=d. hex()[40:42]
  print(key)
  if(key == 18):
    speed.start(90)
    time.sleep(1)
    speed.stop()
    GPIO.output((A,B),(0,1))
    speed.ChangeDutyCycle(20)
    time.sleep(1)
    speed.stop()
    GPIO.cleanup()
```

五、 实验结果

1） 完整小车

见图 8图 9图 10

2） 避障功能

本次实验实现了基于红外检测的自动避障小车应用，小车可以利用红外传感

图 8: 小车顶部

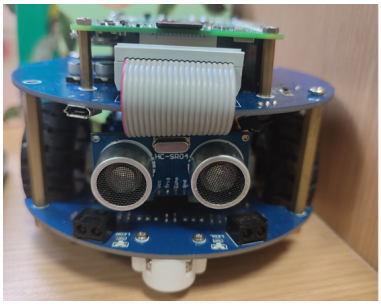

图 9: 小车前部

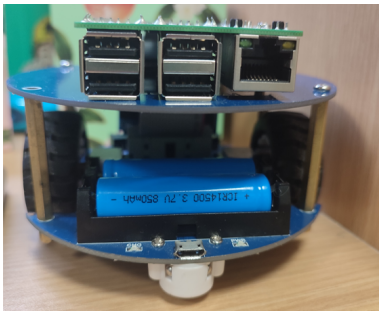

图 10: 小车后部

组件实时探测前、左、右三个方向。障碍物，并根据检测结果实时操纵直流电机来调整行进的速度与方向，实现避障功能。

避障效果展示视频见附件。

全部代码地址：https://github.com/Ironstarboy/RP3b/tree/master/car

视频对应控制台输出结果见图 11. 解释：led stated=1 buzzer state=1 表示蜂鸣器和彩灯都开着。turnLeft,turnRight 记录了每次转向的方向。出现的数字 0，10，20 表示在运行中输入的速度数据。

3） 遥控功能

本次实验完成了遥控器代码的撰写。

4） 循迹功能

本次实验在理论上完成了循迹的代码。并用手掌成功测试且可以输出数据。

```
RaspberryPI:~/car # python3 parse.py --speed 10 -b 1 -l 1
led state = 1
buzzer state = 1
current speed = 10
buzzer on!
led on!
turnLeft!
turnLeft!
turnLeft!
turnLeft!
turnRight!
turnLeft!
20turnRight!
turnRight!
turnLeft!
0
turnRight!
turnLeft!
10
20
```

图 11: 命令行运行输出与控制

六、 实验展望

本次实验实现了小车避障的所有功能，但仍然存在些许不足的地方，可以作为后续实验文进的研究方向。

1） 由于超声波和红外探测其距离地盘比较高。所以小车无法探测低于探测平面的物体，比如地上的绳子

2） 红外探测器灵敏度比较难调，很难调到左右一致

3） 由于宿舍无黑色胶布，没有条件完成循迹的实操

七、 实验感悟

作为一位来自商学院的本科生，本专业内几乎接触不到代码和硬件，但我从小一直对电子科学有着极大的兴趣。这门智能系统嵌入式应用课程，成功让我体验到电子科学的无比魅力！当自己一个个实现功能的时候，很有成就感。我想哪些奋斗在科研或者工业一线的电子人才，也一定很享受亲手改变世界，造出让生活更美好、更便利的电子产品的过程！

非常感谢老师一学期以来的指导！