**智能开门报警器**

**-**基于python实现

姓名: 陈李锋  学号: 2014081025 14通信

**一．项目目的：**

1、使用超声波传感器和其他传感器实现一个智能开门报警器

2、熟悉树莓派上python的使用

**二．需要硬件：**

1、HC-SR04超声波传感器

2、有源蜂鸣器

3、LED和电阻

4、杜邦线和面包板等

5、 树莓派

**三．项目介绍**

对于一个人的宿舍，住在一楼比较担心住安问题，所以这次结合的是老师提供的一些传感器和树莓派，做了一个简易的智能开关报警器。

其主要原理是通过超声波传感器，每隔一段时间检测其和门之间的距离，若发现和之前值不一样，就会通过蜂鸣器警报和LED闪烁。

另外对于传感器的介绍，网上都可以找到，这里就不做搬运工作了。

对于HC-SR04超声波传感器

<http://wenku.baidu.com/view/97031272f46527d3240ce0f2.html>

这次项目是通过树莓派的Python编程，使用了课程教的GPIO等知识，端口选择是BCM模式。

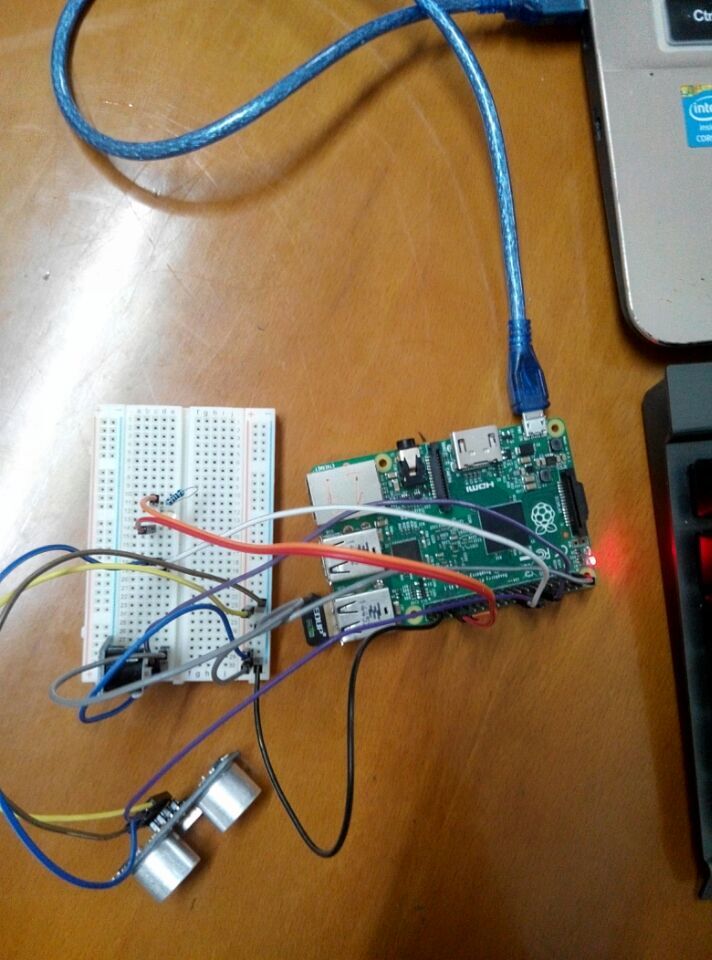
**四．接线和结果：**

1.接线：

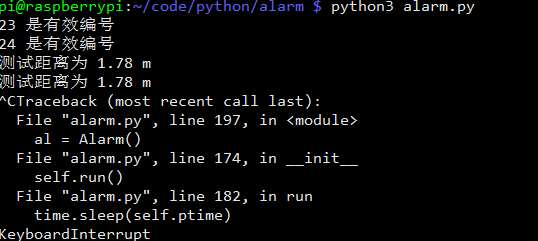
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 传感器 | 超声波Echo | 超声波Trip | 蜂鸣器I/O | LED使能 |
| 树莓派（物理） | 18 | 16 | 12 | 32 |

传感器的其他电源脚和地脚分别接在树莓派的VCC和GND上

2. 接线图片和结果



3.代码主要显示显示结果（没有对着门）



**五．项目体会**

这一次本来使用功能性比Arduino更强的树莓派来做大项目的，但因为树莓派连接上去使用的VI等工具都不尽人意，所以只好在Windows上面编程再移植到板上去，效率实在有点低，所以只是选择做了一个小项目。

这一次做的项目做的是把之前的传感器都尽量使用起来，编程工作也不算太难。网上找的大多数都是对各传感器单独使用的程序，没有把所有的都结合起来。这次搜集了几个小程序，理解后直接移植过来。因为使用的是Python，作为一门glue language,用别人的轮子来做这样的工作也算比较简单。因为是电子系之前的接线等经验也算丰富，对于做起来也算得心应手，所以总的来说整个项目下来都没有遇到什么大的坑 。

**六．程序**

#!/usr/bin/env python

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

import RPi.GPIO as GPIO

import time

import sys

import string

#### 定义Bell类

class Bell(object):

#### Bell类的构造函数

def \_\_init\_\_(self,pin):

self.pin = pin

self.pins = [5,6,12,13,16,17,18,19,20,21,22,23,24,25,26,27]

self.up\_time = 1.5

self.down\_time = 0.5

self.check\_pin(pin)

self.run()

def run(self):

self.setup(self.pin)

try:

self.loop(self.pin)

except KeyboardInterrupt:

self.destroy(self.pin)

#### 检查输入的端口是否合法

def check\_pin(self,pin):

if pin in self.pins:

print("%d 是有效编号"%pin)

else:

print("只能输入以下有效的pin编号")

for i in self.pins:

print (i),

exit()

#### 初始化GPIO口

def setup(self,pin):

#采用BCM编号

GPIO.setmode(GPIO.BCM)

#设置GPIO为输出状态，输入低电平

GPIO.setup(pin,GPIO.OUT)

GPIO.output(pin,GPIO.LOW)

def loop(self,pin):

for i in range(1,10):

GPIO.output(pin,GPIO.HIGH)

print("bell up")

time.sleep(self.up\_time)

GPIO.output(pin,GPIO.LOW)

print("bell down")

time.sleep(self.down\_time)

def destroy(self,pin):

#### 恢复GPIO口状态

GPIO.output(pin,GPIO.LOW)

GPIO.setup(pin,GPIO.IN)

#### 定义了Light类

class Light(object):

#### Light类的构造函数

def \_\_init\_\_(self,pin):

self.pin = pin

self.pins = [5,6,12,13,16,17,18,19,20,21,22,23,24,25,26,27]

self.up\_time = 0.5

self.down\_time = 0.5

self.check\_pin(pin)

self.run()

def run(self):

self.setup(self.pin)

try:

self.loop(self.pin)

except KeyboardInterrupt:

self.destroy(self.pin)

def check\_pin(self,pin):

if pin in self.pins:

print("%d 是有效编号"%pin)

else:

print("只能输入以下有效的pin编号")

for i in self.pins:

print (i),

exit()

#### 初始化GPIO口

def setup(self,pin):

#### 采用BCM编号

GPIO.setmode(GPIO.BCM)

#### 设置GPIO为输出状态，输入低电平

GPIO.setup(pin,GPIO.OUT)

GPIO.output(pin,GPIO.LOW)

def loop(self,pin):

for i in range(1,10):

GPIO.output(pin,GPIO.HIGH)

print("light up")

time.sleep(self.up\_time)

GPIO.output(pin,GPIO.LOW)

print("light down")

time.sleep(self.down\_time)

def destroy(self,pin):

#### 恢复GPIO口状态

GPIO.output(pin,GPIO.LOW)

GPIO.setup(pin,GPIO.IN)

#### 定义Ultrasonic类

class Ultrasonic(object):

def \_\_init\_\_(self,trig\_pin,echo\_pin):

self.trig\_pin = trig\_pin

self.echo\_pin = echo\_pin

self.pins = [5,6,12,13,16,17,18,19,20,21,22,23,24,25,26,27]

self.check\_pin(self.trig\_pin,self.echo\_pin)

self.run(self.trig\_pin,self.echo\_pin)

def check\_pin(self,trig\_pin,echo\_pin):

if trig\_pin in self.pins:

print("%d 是有效编号"%trig\_pin)

else:

print("只能输入以下有效的pin编号")

for i in self.pins:

print (i),

exit()

if echo\_pin in self.pins:

print("%d 是有效编号"%echo\_pin)

else:

print("只能输入以下有效的pin编号")

for i in self.pins:

print (i),

exit()

#### 初始化GPIO口

def setup(self,trig\_pin,echo\_pin):

#### 采用BCM编号

GPIO.setmode(GPIO.BCM)

GPIO.setwarnings(False)

#### 设置GPIO为输出状态，输入低电平

GPIO.setup(self.trig\_pin,GPIO.OUT,initial=GPIO.LOW)

GPIO.setup(self.echo\_pin,GPIO.IN)

def run(self,trig\_pin,echo\_pin):

self.setup(self.trig\_pin,self.echo\_pin)

#### 发出触发信号

GPIO.output(self.trig\_pin,GPIO.HIGH)

#### 保持15us

time.sleep(0.000015)

GPIO.output(self.trig\_pin,GPIO.LOW)

while not GPIO.input(self.echo\_pin):

pass

#### echo端口发现高电平，开始计时

t1 = time.time()

while GPIO.input(self.echo\_pin):

pass

#### echo端口高电平停止，结束计时

t2 = time.time()

length = (t2-t1)\*340/2

print("测试距离为 %0.2f m"%length)

return length

#### 定义Alarm类

class Alarm(object):

def \_\_init\_\_(self):

self.ptime = 5

self.tolerance = 0.05

self.run()

#### 通过超生波模块2次测距，如果测距的距离在可容忍的误差内则Pass

#### 如果明显测距距离不一样，则说明门被打开,点亮报警灯，打开蜂鸣器报警

def run(self):

while True:

ul = Ultrasonic(23,24)

len1 = ul.run(23,24)

time.sleep(self.ptime)

len2 = ul.run(23,24)

if len1 > (len2 - 0.5) or len1 < (len2 + 0.5) :

self.echo()

else:

pass

def echo(self):

#### 点亮报警灯

light = Light(12)

#### 打开报警器

bell = Bell(18)

if \_\_name\_\_ == '\_\_main\_\_':

al = Alarm()