Chapter11 四路巡线

实验目的

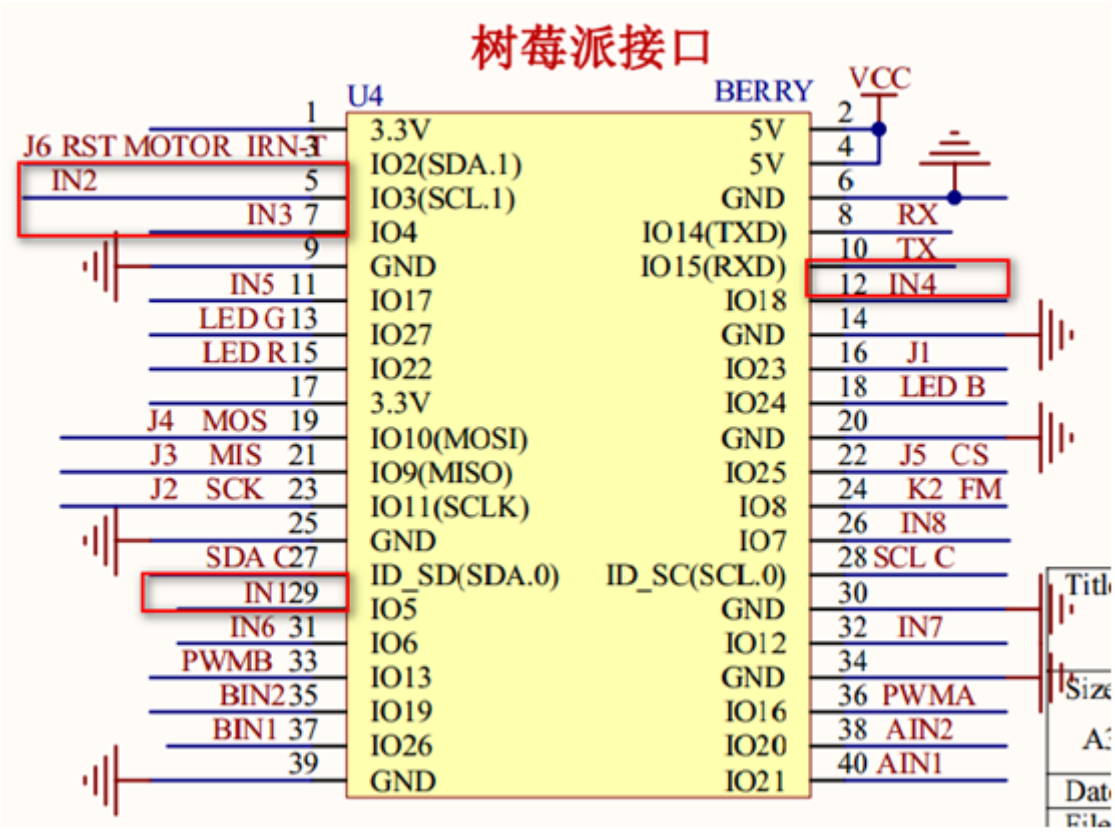
远程登录树莓派系统之后，运行程序，按下按键K2，启动红外巡线功能，小车会自动的巡黑线行走。

实验原理

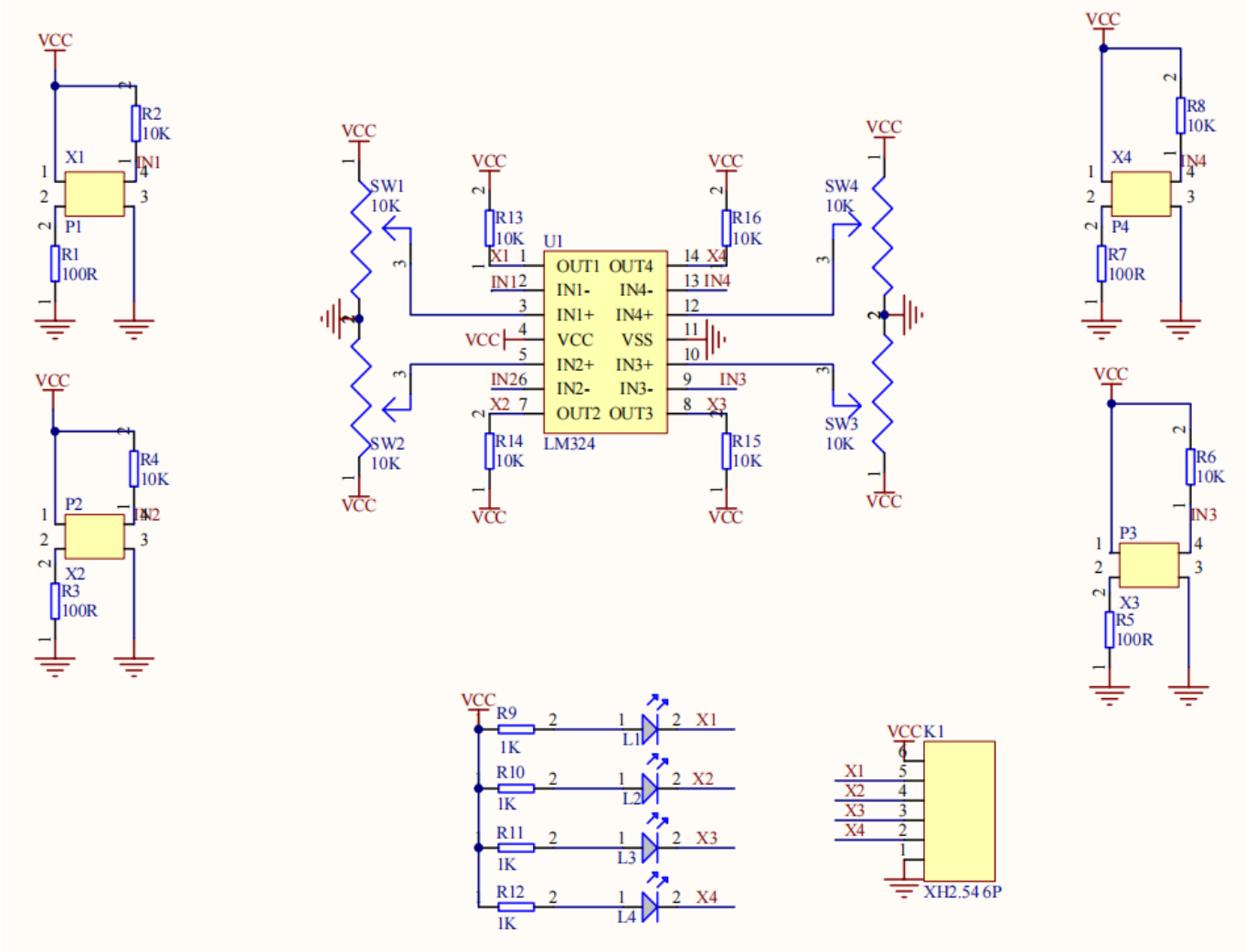
巡线原理

红外传感器巡线的基本原理是利用物体的反射性质，当红外线发射到黑线上时会被黑线吸收掉，而发射到其他的颜色的材料上的红外会有一部分反射回红外接受管上。根据是否能够接收到返回红外，可以编写代码实现小车巡线行驶。

硬件原理



图一 树莓派接口



图二 巡线模块接口及电路

本次实验可以调节四路红外循迹模块的电位器使得巡线的灵敏度达到最佳。

代码实现

```
#-*- coding:UTF-8 -*-
import RPi.GPIO as GPIO
import time

#循迹红外引脚定义
#TrackSensorLeftPin1 TrackSensorLeftPin2 TrackSensorRightPin1 TrackSensorRightPin2
#      3                  5                  4                  18
TrackSensorLeftPin1  =  3   #定义左边第一个循迹红外传感器引脚为3口
TrackSensorLeftPin2  =  5   #定义左边第二个循迹红外传感器引脚为5口
TrackSensorRightPin1 =  4   #定义右边第一个循迹红外传感器引脚为4口
TrackSensorRightPin2 =  18  #定义右边第二个循迹红外传感器引脚为18口

#设置GPIO口为BCM编码方式
GPIO.setmode(GPIO.BCM)

#忽略警告信息
GPIO.setwarnings(False)
```

```
#电机引脚初始化为输出模式
#按键引脚初始化为输入模式
#寻迹引脚初始化为输入模式
def init():
    global pwm_PWMA
    global pwm_PWMB

    GPIO.setup(PWMA,GPIO.OUT,initial=GPIO.HIGH)
    GPIO.setup(AIN2,GPIO.OUT,initial=GPIO.LOW)
    GPIO.setup(AIN1,GPIO.OUT,initial=GPIO.LOW)

    GPIO.setup(PWMB,GPIO.OUT,initial=GPIO.HIGH)
    GPIO.setup(BIN2,GPIO.OUT,initial=GPIO.LOW)
    GPIO.setup(BIN1,GPIO.OUT,initial=GPIO.LOW)

    GPIO.setup(key,GPIO.IN)

    GPIO.setup(TrackSensorLeftPin1,GPIO.IN)
    GPIO.setup(TrackSensorLeftPin2,GPIO.IN)
    GPIO.setup(TrackSensorRightPin1,GPIO.IN)
    GPIO.setup(TrackSensorRightPin2,GPIO.IN)
    #设置pwm引脚和频率为2000hz
    pwm_PWMA = GPIO.PWM(PWMA, 2000)
    pwm_PWMB = GPIO.PWM(PWMB, 2000)
    pwm_PWMA.start(0)
    pwm_PWMB.start(0)
try:
    init()
    key_scan()
    while True:
        #检测到黑线时循迹模块相应的指示灯亮，端口电平为LOW
        #未检测到黑线时循迹模块相应的指示灯灭，端口电平为HIGH
        TrackSensorLeftValue1  = GPIO.input(TrackSensorLeftPin1)
        TrackSensorLeftValue2  = GPIO.input(TrackSensorLeftPin2)
        TrackSensorRightValue1 = GPIO.input(TrackSensorRightPin1)
        TrackSensorRightValue2 = GPIO.input(TrackSensorRightPin2)

        #四路循迹引脚电平状态
        # 0 0 X 0
        # 1 0 X 0
        # 0 1 X 0
        #以上6种电平状态时小车原地右转
        #处理右锐角和右直角的转动
        if (TrackSensorLeftValue1 == False or TrackSensorLeftValue2 == False) and
TrackSensorRightValue2 == False:
            spin_right(1)
            time.sleep(0.08)

        #四路循迹引脚电平状态
        # 0 X 0 0
        # 0 X 0 1
        # 0 X 1 0
        #处理左锐角和左直角的转动
        elif TrackSensorLeftValue1 == False and (TrackSensorRightValue1 == False
```

```
or  TrackSensorRightValue2 == False):
            spin_left(1)
            time.sleep(0.08)

        # 0 X X X
        #最左边检测到
        elif TrackSensorLeftValue1 == False:
            spin_left(1)

        # X X X 0
        #最右边检测到
        elif TrackSensorRightValue2 == False:
            spin_right(1)

        #四路循迹引脚电平状态
        # X 0 1 X
        #处理左小弯
        elif TrackSensorLeftValue2 == False and TrackSensorRightValue1 == True:
            left(1)

        #四路循迹引脚电平状态
        # X 1 0 X
        #处理右小弯
        elif TrackSensorLeftValue2 == True and TrackSensorRightValue1 == False:
            right(1)

        #四路循迹引脚电平状态
        # X 0 0 X
        #处理直线
        elif TrackSensorLeftValue2 == False and TrackSensorRightValue1 == False:
            run(1)

        #当为1 1 1 1时小车保持上一个小车运行状态

except KeyboardInterrupt:
    pass
pwm_PWMA.stop()
pwm_PWMB.stop()
GPIO.cleanup()
```