Chapter13 多方向超声波避障

实验目的

远程登录树莓派之后，运行程序。当前方距离大于50cm时，小车以高速行驶，并通过红外模块判断左右是否有障碍来修正方向；当前方距离大于30cm但小于等于50cm时，小车低速行驶，并通过红外模块判断左右是否有障碍来修正方向；当前方距离当小于等于30cm时，七彩灯模块亮红色，接着转动舵机到0度的位置，超声波测距并记录，转动舵机到180度的位置，超声波测距并记录，舵机归位，并记录所测距离，比较左右的距离来决定向左还是向右避障。当前方，左侧，右侧的距离均小于30cm时应该掉头避障。

实验原理

见之前内容

代码实现

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

on_off=0

#速度常量
ZERO_SPEED          =0
LOW_SPEED           =10
MID_SPEED           =60
MAX_SPEED           =100
SPIN_LOW_SPEED      =10
SPIN_MID_SPEED      =40
SPIN_MAX_SPEED      =70

#小车电机引脚定义
AIN2 = 20
AIN1 = 21
BIN2 = 19
BIN1 = 26
PWMA = 16
PWMB = 13

#小车按键定义
key = 8

#超声波引脚定义
EchoPin = 0
TrigPin = 1

#RGB三色灯引脚定义
LED_R = 22
LED_G = 27
```

```
LED_B = 24

#舵机引脚定义
ServoPin = 23

#红外避障引脚定义
AvoidSensorLeft = 12
AvoidSensorRight = 17

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

#忽略警告信息
GPIO.setwarnings(False)

#按键回调函数
def key_pressed_callback(pin):
    global on_off
    on_off=0 if on_off else 1

#电机引脚初始化为输出模式
#按键引脚初始化为输入模式
#超声波,RGB三色灯,舵机引脚初始化
#红外避障引脚初始化
def init():
    global pwm_PWMA
    global pwm_PWMB
    global pwm_servo
    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(EchoPin,GPIO.IN)
    GPIO.setup(TrigPin,GPIO.OUT)
    GPIO.setup(LED_R, GPIO.OUT)
    GPIO.setup(LED_G, GPIO.OUT)
    GPIO.setup(LED_B, GPIO.OUT)
    GPIO.setup(ServoPin, GPIO.OUT)
    GPIO.setup(AvoidSensorLeft,GPIO.IN)
    GPIO.setup(AvoidSensorRight,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)
    #设置舵机的频率和起始占空比
    pwm_servo = GPIO.PWM(ServoPin, 50)
    pwm_servo.start(0)

    GPIO.add_event_detect(key,GPIO.RISING,key_pressed_callback,bouncetime=15)
```

```
#超声波函数
def get_distance():
    GPIO.output(TrigPin,GPIO.HIGH)
    time.sleep(0.000015)
    GPIO.output(TrigPin,GPIO.LOW)
    while not GPIO.input(EchoPin):
        pass
    t1 = time.time()
    while GPIO.input(EchoPin):
        pass
    t2 = time.time()
    distance=((t2 - t1)* 340 / 2) * 100
    print ("distance is {}".format(distance))
    time.sleep(0.01)
    return distance

#舵机旋转到指定角度
def servo_appointed_detection(pos):
    for i in range(18):
        pwm_servo.ChangeDutyCycle(2.5 + 10 * pos/180)


#舵机旋转超声波测距避障，led根据车的状态显示相应的颜色
def servo_color_carstate():
    #品红色
    LED_RGB(GPIO.HIGH,GPIO.LOW,GPIO.HIGH)

    back(LOW_SPEED, LOW_SPEED)
    time.sleep(0.08)
    brake()

    #舵机旋转到0度，即右侧，测距
    servo_appointed_detection(0)
    time.sleep(0.8)
    rightdistance = get_distance()

    #舵机旋转到180度，即左侧，测距
    servo_appointed_detection(180)
    time.sleep(0.8)
    leftdistance = get_distance()

    #舵机旋转到90度，即前方，测距
    servo_appointed_detection(90)
    time.sleep(0.8)
    frontdistance = get_distance()

    if leftdistance < 30 and rightdistance < 30 and frontdistance < 30:
        #亮红色，掉头
        LED_RGB(GPIO.HIGH,GPIO.LOW,GPIO.LOW)
        spin_right(SPIN_MID_SPEED)
        time.sleep(0.58)
    elif leftdistance >= rightdistance:
```

```
            #亮蓝色
            LED_BLUE()
            spin_left(SPIN_MID_SPEED)
            time.sleep(0.28)
        elif leftdistance <= rightdistance:
            #亮品红色，向右转
            LED_RGB(GPIO.HIGH,GPIO.LOW,GPIO.HIGH)
            spin_right(SPIN_MAX_SPEED)
            time.sleep(0.28)

    #延时2s
    time.sleep(2)

    #try/except语句用来检测try语句块中的错误，
    #从而让except语句捕获异常信息并处理。
    try:
        init()
        while True:
            if on_off:
                distance = get_distance()
                if distance > 50:
                    #遇到障碍物,红外避障模块的指示灯亮,端口电平为LOW
                    #未遇到障碍物,红外避障模块的指示灯灭,端口电平为HIGH
                    LeftSensorValue  = GPIO.input(AvoidSensorLeft)
                    RightSensorValue = GPIO.input(AvoidSensorRight)

                    if LeftSensorValue == True and RightSensorValue == True :
                        run(MID_SPEED, MID_SPEED)          #当两侧均未检测到障碍物时调用
    前进函数
                    elif LeftSensorValue == True and RightSensorValue == False :
                        spin_left(SPIN_MID_SPEED)      #右边探测到有障碍物，有信号返回，原
    地向左转
                        time.sleep(0.002)
                    elif RightSensorValue == True and LeftSensorValue == False:
                        spin_right(SPIN_MID_SPEED)     #左边探测到有障碍物，有信号返回，原
    地向右转
                        time.sleep(0.002)
                    elif RightSensorValue == False and LeftSensorValue == False :
                        spin_right(SPIN_MID_SPEED)     #当两侧均检测到障碍物时调用固定方向
    的避障(原地右转)
                        time.sleep(0.002)

                    run(MID_SPEED, MID_SPEED)
                    LED_GREEN()

                elif 30 < distance <= 50:
                    #遇到障碍物,红外避障模块的指示灯亮,端口电平为LOW
                    #未遇到障碍物,红外避障模块的指示灯灭,端口电平为HIGH
                    LeftSensorValue  = GPIO.input(AvoidSensorLeft)
                    RightSensorValue = GPIO.input(AvoidSensorRight)

                    if LeftSensorValue == True and RightSensorValue == True :
                        run(LOW_SPEED, LOW_SPEED)          #当两侧均未检测到障碍物时调用
    前进函数
```

```
                elif LeftSensorValue == True and RightSensorValue == False :
                    spin_left(SPIN_MID_SPEED)     #右边探测到有障碍物，有信号返回，原
地向左转
                    time.sleep(0.002)
                elif RightSensorValue == True and LeftSensorValue == False:
                    spin_right(SPIN_MID_SPEED)    #左边探测到有障碍物，有信号返回，原
地向右转
                    time.sleep(0.002)
                elif RightSensorValue == False and LeftSensorValue == False :
                    spin_right(SPIN_MID_SPEED)    #当两侧均检测到障碍物时调用固定方向
的避障(原地右转)
                    time.sleep(0.002)
                run(LOW_SPEED, LOW_SPEED)
                LED_GREEN()

            elif distance <= 30:
                servo_color_carstate()
        else:
            brake(0)
            while not on_off:
                pass
except KeyboardInterrupt:
    pass
pwm_PWMA.stop()
pwm_PWMB.stop()
GPIO.cleanup()
```