Chapter14 自动驾驶

实验目的

调用前面实现好的各种模块来实现简易的自动驾驶功能。

实验步骤

模块化

将我们每一个chapter实现的功能封装成模块，需要使用相应模块时直接引入相应的模块。

行驶模块

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

#速度常量
ZERO_SPEED          =0
LOW_SPEED           =10
MID_SPEED           =30
MAX_SPEED           =100
SPIN_LOW_SPEED      =10
SPIN_MID_SPEED      =40
SPIN_MAX_SPEED      =70

#小车电机引脚定义，A控制左侧，B控制右侧。
AIN1 = 21
AIN2 = 20

BIN1 = 26
BIN2 = 19

PWMA = 16
PWMB = 13

global pwm_PWMA
global pwm_PWMB

#电机引脚初始化操作
def car_run_init():
    global pwm_PWMA,pwm_PWMB
    #初始化引脚
    GPIO.setup(PWMA,GPIO.OUT,initial=GPIO.LOW)
    GPIO.setup(AIN2,GPIO.OUT,initial=GPIO.LOW)
    GPIO.setup(AIN1,GPIO.OUT,initial=GPIO.LOW)

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


def car_run_clean():
    global pwm_PWMA, pwm_PWMB
    pwm_PWMA.stop()
    pwm_PWMB.stop()
    cleanList=[AIN1,AIN2,BIN1,BIN2,PWMA,PWMB]
    GPIO.cleanup(cleanList)

#小车前进
def run(leftSpeed,rightSpeed):
    global pwm_PWMA, pwm_PWMB
    GPIO.output(AIN2, GPIO.HIGH)
    GPIO.output(AIN1, GPIO.LOW)
    GPIO.output(BIN2, GPIO.HIGH)
    GPIO.output(BIN1, GPIO.LOW)
    pwm_PWMA.ChangeDutyCycle(leftSpeed)
    pwm_PWMB.ChangeDutyCycle(rightSpeed)

#小车后退
def back(leftSpeed,rightSpeed):
    global pwm_PWMA, pwm_PWMB
    GPIO.output(AIN2, GPIO.LOW)
    GPIO.output(AIN1, GPIO.HIGH)
    GPIO.output(BIN2, GPIO.LOW)
    GPIO.output(BIN1, GPIO.HIGH)
    pwm_PWMA.ChangeDutyCycle(leftSpeed)
    pwm_PWMB.ChangeDutyCycle(rightSpeed)

#小车左转
def left(speed):
    global pwm_PWMA, pwm_PWMB
    GPIO.output(AIN2, GPIO.LOW)
    GPIO.output(AIN1, GPIO.LOW)
    GPIO.output(BIN2, GPIO.HIGH)
    GPIO.output(BIN1, GPIO.LOW)
    pwm_PWMA.ChangeDutyCycle(ZERO_SPEED)
    pwm_PWMB.ChangeDutyCycle(speed)

#小车右转
def right(speed):
    global pwm_PWMA, pwm_PWMB
    GPIO.output(AIN2, GPIO.HIGH)
    GPIO.output(AIN1, GPIO.LOW)
    GPIO.output(BIN2, GPIO.LOW)
    GPIO.output(BIN1, GPIO.LOW)
    pwm_PWMA.ChangeDutyCycle(speed)
    pwm_PWMB.ChangeDutyCycle(ZERO_SPEED)
```

```
#小车原地左转
def spin_left(spinSpeed):
    global pwm_PWMA, pwm_PWMB
    GPIO.output(AIN2, GPIO.LOW)
    GPIO.output(AIN1, GPIO.HIGH)
    GPIO.output(BIN2, GPIO.HIGH)
    GPIO.output(BIN1, GPIO.LOW)
    pwm_PWMA.ChangeDutyCycle(spinSpeed)
    pwm_PWMB.ChangeDutyCycle(spinSpeed)

#小车原地右转
def spin_right(spinSpeed):
    global pwm_PWMA, pwm_PWMB
    GPIO.output(AIN2, GPIO.HIGH)
    GPIO.output(AIN1, GPIO.LOW)
    GPIO.output(BIN2, GPIO.LOW)
    GPIO.output(BIN1, GPIO.HIGH)
    pwm_PWMA.ChangeDutyCycle(spinSpeed)
    pwm_PWMB.ChangeDutyCycle(spinSpeed)


#小车停止
def brake():
    GPIO.output(AIN2, GPIO.LOW)
    GPIO.output(AIN1, GPIO.LOW)
    GPIO.output(BIN2, GPIO.LOW)
    GPIO.output(BIN1, GPIO.LOW)
```

LED模块

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

#RGB三色灯引脚定义
LED_R = 22
LED_G = 27
LED_B = 24

def led_init():
    GPIO.setup(LED_R, GPIO.OUT)
    GPIO.setup(LED_G, GPIO.OUT)
    GPIO.setup(LED_B, GPIO.OUT)

def led_clean():
    led_list=[LED_R,LED_G,LED_B]
    GPIO.cleanup(led_list)

def LED_RGB(R,G,B):
    GPIO.output(LED_R,R)
    GPIO.output(LED_G,G)
    GPIO.output(LED_B,B)

def LED_RED():
    GPIO.output(LED_R,GPIO.HIGH)
    GPIO.output(LED_G,GPIO.LOW)
    GPIO.output(LED_B,GPIO.LOW)

def LED_GREEN():
    GPIO.output(LED_R,GPIO.LOW)
    GPIO.output(LED_G,GPIO.HIGH)
    GPIO.output(LED_B,GPIO.LOW)

def LED_BLUE():
    GPIO.output(LED_R,GPIO.LOW)
    GPIO.output(LED_G,GPIO.LOW)
    GPIO.output(LED_B,GPIO.HIGH)

def led_pause():
    LED_RGB(GPIO.LOW,GPIO.LOW,GPIO.LOW)
```

红外测距模块

```
"""
遇到光线,寻光模块的指示灯灭,端口电平为HIGH
未遇光线,寻光模块的指示灯亮,端口电平为LOW
"""
#-*- coding:UTF-8 -*-
import RPi.GPIO as GPIO

#光敏电阻引脚定义
ldrSensorLeft = 7
ldrSensorRight = 6

def infrared_light_init():
    GPIO.setup(ldrSensorLeft,GPIO.IN)
    GPIO.setup(ldrSensorRight,GPIO.IN)

def leftSensorValue_light():
    return GPIO.input(ldrSensorRight)

def rightSensorValue_light():
    return GPIO.input(ldrSensorRight)
```

超声波测距模块

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

DISTANCE_1= 30
DISTANCE_2= 50


#超声波引脚定义
echoPin = 0
trigPin = 1


def ultrasonic_init():
    GPIO.setup(echoPin,GPIO.IN)
    GPIO.setup(trigPin,GPIO.OUT)

#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):
        #print("等待接受返回信号")
        pass
    t1 = time.time()
    while GPIO.input(echoPin):
        #print("正在接受返回信号")
        pass
    t2 = time.time()
    distance=((t2 - t1)* 340 / 2) * 100
    print ("distance is {}".format(distance))
    time.sleep(0.01)
    #print(distance)
    return distance
```

按键模块

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

#小车按键定义
key = 8

def key_init():
    GPIO.setup(key,GPIO.IN)
```

舵机模块

```python
#-*- coding:UTF-8 -*-
"""
本次舵机转动控制七彩灯控制舵机采用的是系统的pwm库
"""
import RPi.GPIO as GPIO
import time

#开关
on_off=0

#舵机引脚定义
servoPin = 23

global pwm_servo

def servo_init():
    global pwm_servo
    GPIO.setup(servoPin, GPIO.OUT)
    pwm_servo = GPIO.PWM(servoPin, 50)
    pwm_servo.start(0)

    #设置pwm引脚和频率为50hz
def servo_pause():
    GPIO.output(servoPin,GPIO.LOW)


def servo_clean():
    global pwm_servo
    pwm_servo.stop()
    GPIO.cleanup(servoPin)


#舵机旋转到指定角度
def servo_appointed_detection(pos):
    global pwm_servo
    pwm_servo.ChangeDutyCycle(2.5 + 10 * pos/180)
    time.sleep(0.01)


if __name__=="__main__":
    while 1:
        #for i in range(0,160):
        #    servo_appointed_detection(i)
        #    time.sleep(0.01)
        #for i in range(160,0,-1):
        #    servo_appointed_detection(i)
        #    time.sleep(0.01)
        #time.sleep(1)

        servo_appointed_detection(160)
        time.sleep(0.8)
        servo_appointed_detection(75)
        time.sleep(0.8)
        servo_appointed_detection(0)
        time.sleep(0.8)
```

光敏循光模块

```
"""
遇到光线,寻光模块的指示灯灭,端口电平为HIGH
未遇光线,寻光模块的指示灯亮,端口电平为LOW
"""
#-*- coding:UTF-8 -*-
import RPi.GPIO as GPIO

#光敏电阻引脚定义
ldrSensorLeft = 7
ldrSensorRight = 6

def infrared_light_init():
    GPIO.setup(ldrSensorLeft,GPIO.IN)
    GPIO.setup(ldrSensorRight,GPIO.IN)

def leftSensorValue_light():
    return GPIO.input(ldrSensorRight)

def rightSensorValue_light():
    return GPIO.input(ldrSensorRight)
```

按键控制开关

之前的按键控制不好用，所以我们换一个控制方式，按下key2小车开始运动，再按一次key2小车关机。

我们使用事件监测函数来实现，若不清楚请参见**Chapter3**中的必备知识（一猜就知道都忘的差不多了）。

```
#按下开关开始启动，再按一下熄火
on_off=0

def key_pressed_callback(pin):
    global on_off
    if not on_off:
        on_off=1
    else:
        brake()
        car_run_clean()
        servo_clean()
        led_clean()
        exit(0)

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

自动驾驶主函数

要实现的功能：

1.时刻调用超声波模块进行测距，当离前方障碍物的距离大于设定值时，通过红外避障模块时刻修正方向，亮绿灯。当距离小于设定值时，停车，探测小车左侧和右侧的距离，根据距离进行相应操作。（描述不太明确，看代码就懂了哈哈）

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

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)

from car_run import *
from infrared_distance import *
from infrared_light import *
from key import *
from led import *
from servo import *
from ultrasonic import *

#按下开关开始启动，再按一下熄火
on_off=0

def key_pressed_callback(pin):
    global on_off
    if not on_off:
        on_off=1
    else:
        brake()
        car_run_clean()
        servo_clean()
        led_clean()
        exit(0)



def run_sides_space(run_speed,spin_time=0.2,spin_speed=SPIN_MID_SPEED):
    '''
    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
    '''
    left_sensor = leftSensorValue_distance()
    right_sensor=rightSensorValue_distance()
    print("************",left_sensor)
    if left_sensor==GPIO.HIGH and right_sensor==GPIO.HIGH:
        print("左右都没障碍")                              #左右都没有
        run(run_speed,run_speed)
    elif left_sensor==GPIO.LOW and right_sensor==GPIO.HIGH:     #左边有障碍
        print("左边有障碍")
```

```
            right(spin_speed)
            time.sleep(spin_time)
        elif left_sensor==GPIO.HIGH and right_sensor==GPIO.LOW:     #右边有障碍
            print("右边有障碍")
            left(spin_speed)
            time.sleep(spin_time)
        else:
            print("左右都有障碍")                                #左右都有障碍
            spin_right(spin_speed)
            time.sleep(spin_time)


def runAndLight_according_distance():
    '''
    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并选择行驶模式
     '''
    #品红色
    LED_RGB(GPIO.HIGH,GPIO.LOW,GPIO.HIGH)

    back(LOW_SPEED, LOW_SPEED)
    time.sleep(0.18)
    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(160)
    time.sleep(0.8)
    leftdistance = get_distance()

    #舵机旋转到90度，即前方，测距
    servo_appointed_detection(75)
    time.sleep(0.8)
    frontdistance = get_distance()

    if leftdistance < DISTANCE_1 and rightdistance < DISTANCE_1 and frontdistance
< DISTANCE_1:
        #亮红色，掉头
        LED_RGB(GPIO.HIGH,GPIO.LOW,GPIO.LOW)
        spin_right(SPIN_MID_SPEED)
        time.sleep(0.6)
    elif leftdistance >= rightdistance:
        #亮蓝色
        LED_BLUE()
        spin_left(SPIN_MID_SPEED)
        time.sleep(0.15)
    elif leftdistance <= rightdistance:
        #亮蓝色
        LED_BLUE()
        spin_right(SPIN_MAX_SPEED)
        time.sleep(0.15)
```

```python
if __name__=="__main__":
    GPIO.setmode(GPIO.BCM)
    GPIO.setwarnings(False)

    car_run_init()
    infrared_distance_init()
    #infrared_light_init()
    key_init()
    led_init()
    servo_init()
    ultrasonic_init()

    GPIO.add_event_detect(key,GPIO.RISING,key_pressed_callback,bouncetime=15)
    time.sleep(2)
    print("Press key2 to poweron!")
    while 1:
        if not on_off:
            brake()
            led_pause()
            servo_pause()
            while not on_off:
                pass

        else:
            distance=get_distance()
            if distance>DISTANCE_2:
                print("distance>50")
                run_sides_space(MID_SPEED)
                #run(MID_SPEED,MID_SPEED)
                LED_GREEN()
            elif distance>DISTANCE_1 and distance<=DISTANCE_2:
                print("30<distance<=50")
                LED_RGB(1,0,1)
                run_sides_space(LOW_SPEED)

            else:
                print("distance<30")
                runAndLight_according_distance()

    car_run_cleanup()
    servo_cleanup()
    led_cleanup()
```