电子系统导论实验报告

实验 10 【八字避障】

指导教师： 邹卓

学生姓名： 李晓婉 学生姓名： 陈筠 学生姓名： 贺祺

学 号： 22307130364 学 号 22307130508 学 号： 22307130442

专 业： 技术科学试验班 专 业： 技术科学试验班 专 业： 技术科学试验班

日 期： 2023.6.6

一、 实验目的：

1. 锻炼学生应用所学知识解决实际问题的能力，通过自行组装小车、在绕障代码基础上修改代码、多次调试寻找最快速度与最佳稳定性，使其顺利完成八字绕行的进阶任务。
2. 锻炼学生面对综合的项目，独立思考并利用网络查询相关拓展知识的能力。
3. 锻炼学生自行连接代码和器件安装的能力，贯通对软硬件的知识体系。
4. 培养学生基于突发的问题独立找出问题关键与本质，分析问题原因，从而解决问题的科学精神与品质。
5. 培养组员间充分发挥各自优势，分工与团队合作结合地完成任务的能力。

二、 实验原理：

1.自动控制：

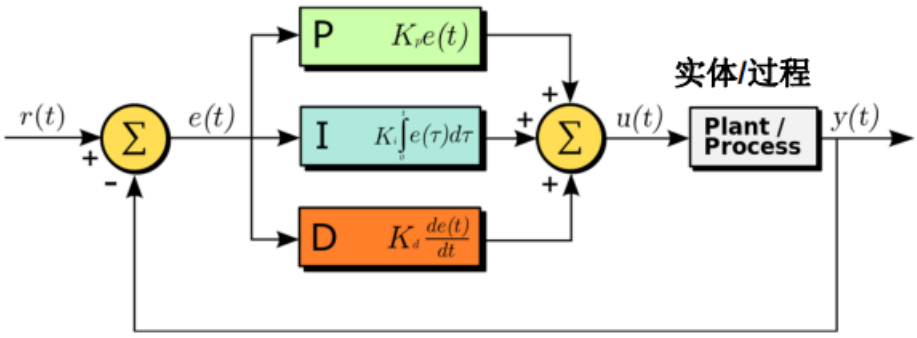
在没有人直接参与的情况下，利用外加的设备或装置，使机器、设备或生产过程的某个工作状态或参数自动地按照预定的规律运行。

2.PID 控制：

（1）简介：

当今的闭环自动控制技术都是基于反馈的概念以减少不确定性。反馈理论的要素包括三个部分：测量、比较和执行。测量关键的是被控变量的实际值，与期望值相比较，用这个偏差来纠正系统的响应，执行调节控制。在工程实际中，应用最为广泛的调节器控制规律为比例(proportion)、积分(integral)、微分(derivative)控制，简称 PID 控制，又称 PID 调节。

（2）控制原理：



如上图所示

r(t)是所需的设定值；

y(t)是测量的过程输出值；

e(t) 即偏差是设定值与过程值的差（r(t)-y(t)）；

u(t) 是经过计算得到的控制变量，输入到控制器中。

P ： proportion （非负比例系数），就是偏差乘以一个常数。

I ： integral （非负积分系数），就是对偏差进行积分运算。

D ： derivative （非负微分系数），对偏差进行微分运算 。

以控制小车为例：

将小车速度设定值 r(t)为 3m/s，由光电码盘测得速度为 3.2m/s，即过程输出值 y(t)；偏差 e(t)为-0.2m/s。

P：将-0.2m/s 乘以一个系数（正）输入到控制器中，以减小输出的占空比，则车轮转速将降低，向设定值靠近。KP 越大则调节的灵敏度越大，但过大可能会使实际速度低于 3m/s（超调）。

I：只经过比例调节的小车，可能稳定后的速度为 3.1m/s，存在稳态误差-0.1m/s；虽然误差很小，但是因为积分项也会随着时间的增加而加大，它推动控制器的输出增大，从而使稳态误差进一步减小，直到等于 0。

D：小车中有些组件存在较大惯性或者滞后性，其变化总是落后于误差的变化。假设经比例调节后实际速度为 3.1m/s，则设定速度与实际速度的差值由-0.2m/s 变为-0.1m/s，e(t)的差分为 0.1m/s2，将此差分乘以系数（正）加到控制变量中，相比只有比例环节减缓了速度降低的趋势（减小超调量）。

（3）数学表达：

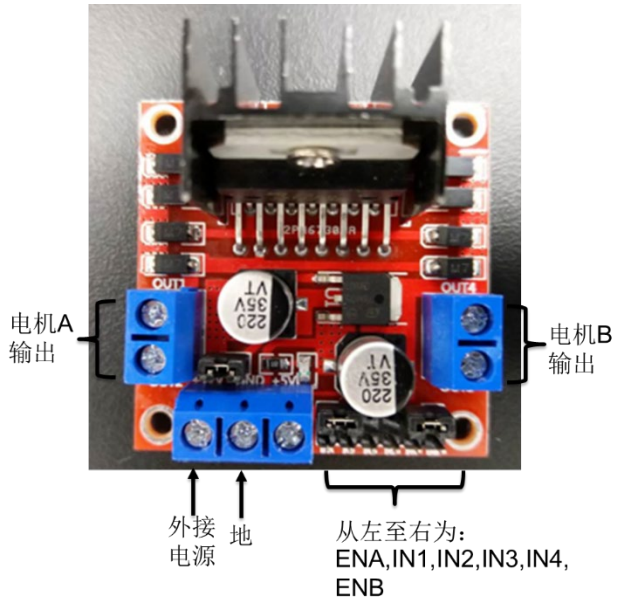
$$u(t) = K_p e(t) + K_i \int_0^t e(t')\,dt' + K_d \frac{de(t)}{dt}$$

Kp、Ki、Kd 都是非负的，分别表示比例项，积分项和微分项的系数。

3. 电机驱动板

这块驱动板的核心是 L298N 芯片，该芯片包含了两个 H 桥结构。

树莓派和驱动板的逻辑部分需要共地，否则树莓派的逻辑就无法被驱动板识别。如果树莓派未使用锂电池供电：树莓派的 GND 需要和电池负极一同接到 GND 端以共地；如果树莓派也使用锂电池供电，则本身就共地，GND 端接入电池负极即可。



4. L298N

Inputs		Function
V_{en} = H	C = H ; D = L	Forward
	C = L ; D = H	Reverse
	C = D	Fast Motor Stop
V_{en} = L	C = X ; D = X	Free Running Motor Stop

L = Low　　H = High　　X = Don't care

树莓派控制驱动板通常有两种方法。

方法一：C 和 D 端输入固定，Ven 端输入 PWM 波。当 PWM 在高电平时，电机加速；PWM 在低电平时，电机自由旋转。

方法二：Ven 端和 C（或 D）端输入固定，D（或 C）端输入 PWM 波。当 PWM 在高电平时，电机加速；PWM 在低电平时，电机制动。

5. Python-OpenCV 图像处理——图像阈值处理

图像阈值处理是一种广泛应用的分割技术，利用图像中要提取的目标物与其背景在灰度特性上的差异，把图像视为具有不同灰度级的两类区域（目标和背景）的组合，选取一个合适的阈值，以确定图像中每个像素点应该属于目标还是背景区域，从而产生相应的二值图像。

实验中对图像进行了多次处理：

a) 将图像从 RGB 色彩空间转换到 YCrCb 色彩空间。YCrCb 色彩空间将 RGB 图像表示为亮度（Y）和两个色度信号（Cr 和 Cb）的组合；

b) 提取 Cr 通道。这使得亮度的变化不会影响实验；

c) 二值化。使得跑道和图像的其余部分完全区分开来：跑道是黑色，其余部分是白色。这么做方便了确定方向偏差的逻辑。

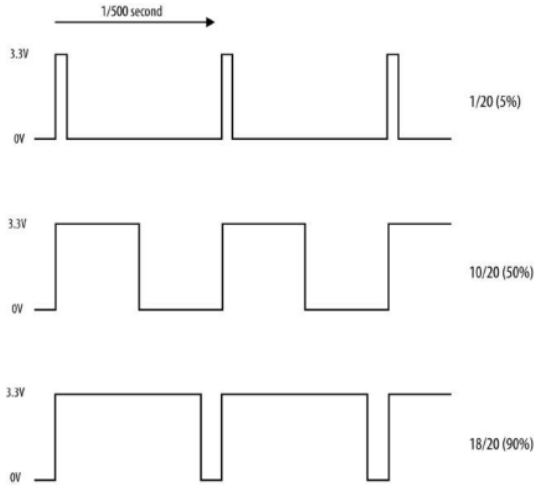
6.脉冲宽度调制（PWM, Pulse Width Modulation）

在实际应用中，PWM 波的占空比是 PWM 的主要特性。

（1）占空比 = 脉宽 / 周期

（2）调节占空比：可以固定脉宽，改变周期；也可以固定周期，改变脉宽。我们通常采用后者。

（3）占空比越大，从整个周期来看，平均电压越高；占空比越小，则平均电压越低。



（4）如何产生 PWM 波

a)软件 PWM

先在目标 GPIO 上输出一个电平（高电平或低电平），持续一段时间，然后把电平取反，再持续一段时间，再取反，循环往复。这种方式的精度一般较低，受到定时器精度、操作系统调度等影响，并且一般依赖于 CPU 中断，会占用少量 CPU 资源。

b)硬件 PWM

有些 CPU 自带 PWM 硬件，只要给出期望的频率和占空比，这些硬件就会独立产生 PWM 波，依赖于内部的硬件定时电路，精度通常较高，而且不需要占用额外的 CPU 资源。

注：使用 RPi.GPIO 模块的 PWM 类产生 PWM 波，这个模块产生的 PWM 全部是软件 PWM。

6.直流电机

能将直流电能转换成机械能的旋转电机。两端电压差越大，转得越快。如果给它加上间歇性通断的电压，电机就会

“加速-减速-加速-减速-加速……”；如果这个电压“通断”足够快，就会使电机转速较稳定地维持在某一数值，这个“数值”取决于输入电压的平均值。

7. 用 PWM 给直流电机调速

通过改变 PWM 波的占空比给直流电机调速。一般小型处理器的引脚不能直接驱动直流电机。因为直流电机将电能转换为机械能，需要比较大的电流提供电能，我们的小车上的直流电机所需的电流可能是安培级的，而树莓派的 GPIO 只能提供毫安级的电流。所以需要有个驱动电路，它有输出大电流的能力，能接收树莓派的控制信号，并按照树莓派的意思让电机转或不转。对于我们所使用的小车上的电机，用 50Hz 到一两百 Hz 的频率已经不会产生明显的抖动，而且频率不算高，即便是软件 PWM 也能胜任。

三、实验内容

1.正确设置todesk，保证todesk可远程访问，并使小车连接上手机热点。

2.完成小车的组装，特别是寻找最合适的摄像头角度并使其固定,我们在这次实验中选择与绕桩竞速同样的三角固定方式，但是将摄像头的位置压低，确保不让小车提前看到交叉口产生位置的错误判断。

3.写入代码，选择PID控制，通过调节占空比，PID参数，在拐弯处的边界速度control值等来减少小车震荡，提高小车运行速度完成八字避障。

四、实验分析

1.本次实验中，由于赛道设置有交叉口，我们小组发现，如果小车在直道上面震荡过于明显，摄像头位置过高提前看到了交叉口，会有转错弯的情况发生。根据这个情况我们选择让小车加速减少震荡冲过交叉口。

2.增量式PID相较标准PID在本次实验中的优势：

a)更快的响应速度：增量式PID控制器使用误差变化量作为控制量，相比标准PID控制器的误差值更能反映系统的动态变化，因此响应速度更快。
b)更精准的控制：增量式PID控制器对误差的变化进行计算，可以更加准确地对小车行驶偏差进行调整，有效提高了控制精度。
c)更好的鲁棒性：在小车巡线中，由于路面、光照等环境因素的变化，控制系统可能会遭遇噪声、干扰等问题，增量式PID控制器通过计算误差变化量，可以更好地消除噪声、干扰等因素的影响，提高了系统的鲁棒性。

综上所述，增量式PID控制器在小车巡线中具有更快的响应速度、更精准的控制和更好的鲁棒性，可以更好地适应小车行驶过程中的动态变化和环境干扰，因此在小车巡线控制中具有一定的优势。

3.路径完全偏离时的补救:

当小车完全偏离赛道时，程序获得的二值化图像是全白的，这时的处理是直接跳过本次循环继续读下一帧图像，直到再次“看到”跑道恢复正常的控制。这样做使得小车在“看不到”跑道时沿用最后一次 “看到”跑道时使用的占空比，进而做出正确的转向。

4.实验中的障碍物箱子也会对小车的行进造成影响，有一个箱子是带有红色边缘的，如果小车摄像头扫到了这个红色边缘区域将会影响direction的判断，导致提前转弯撞箱子（我们第二次测试的时候就出现了这种情况）。

5.代码分析：

代码一：

```
# 这是一个示例 Python 脚本。
# coding:utf-8
import RPi.GPIO as GPIO
import time
import cv2
import numpy as np
#前期准备部分

#限制较大值
def sign(x):
    if x > 0:
        return 1.0
    else:
        return -1.0

# 定义引脚
EA, I2, I1, EB, I4, I3 = (13, 19, 26, 16, 20, 21)
FREQUENCY = 50

# 设置 GPIO 口
GPIO.setmode(GPIO.BCM)

# 设置 GPIO 口为输出
GPIO.setup([EA, I2, I1, EB, I4, I3], GPIO.OUT)
GPIO.output([EA, I2, EB, I3], GPIO.LOW)
GPIO.output([I1, I4], GPIO.HIGH)

pwma = GPIO.PWM(EA, FREQUENCY)
pwmb = GPIO.PWM(EB, FREQUENCY)

# pwm 波初始化
pwma.start(0)
pwmb.start(0)

# center 定义
center_now = 320
```

```
# 打开摄像头，图像尺寸 640*480（长*高），opencv 存储值为 480*640（行*列）
cap = cv2.VideoCapture(0)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)

# PID 定义和初始化三个 error 和 adjust 数据
error = [0.0] * 3
adjust = [0.0] * 3

#PID 参数配置、目标值、左右轮基准占空比和占空比偏差范围（根据实际情况调整）
kp = 2.5
ki = 0.45
kd = 0.2
target = 320
lspeed = 55
rspeed = 55
control = 30
ret, frame = cap.read()

#前期准备完毕，节省时间
print("准备完毕！按下 Enter 启动！")
input()



try:
    while True:
        ret, frame = cap.read()

        # RGB 图像转换到 YCrCb 空间
        YCrCb_orange = cv2.cvtColor(frame, cv2.COLOR_BGR2YCrCb)
        # 提取 Cr 通道并高斯去噪，（3,3）表示高斯核大小
        Cr_orange = cv2.GaussianBlur(YCrCb_orange[:, :, 1], (3, 3), 0)
        # 二值化，160 为阈值
        ret, imgbinary = cv2.threshold(Cr_orange, 140, 255,
cv2.THRESH_BINARY)
        cv2.imshow('display', imgbinary)

        # 单看第 400 行的像素值 s
        color = imgbinary[400]
        # 找到 black 色的像素点个数
        black_count = np.sum(color == 255)
```

```
        # 防止 black_count=0 的报错
        if black_count == 0:
            continue
        else:
            orange_index = np.where(color == 255)
        # 找到黑色像素的中心点位置
        center_now = (orange_index[0][black_count - 1] +
orange_index[0][0]) / 2

        # 计算出 center_now 与标准中心点的偏移量
        direction = center_now - 320

        print("偏差值：", direction)

        # 更新 PID 误差
        error[0] = error[1]
        error[1] = error[2]
        error[2] = center_now - target

        # 更新 PID 输出（增量式 PID 表达式）
        adjust[0] = adjust[1]
        adjust[1] = adjust[2]
        adjust[2] = adjust[1] \
                    + kp * (error[2] - error[1]) \
                    + ki * error[2] \
                    + kd * (error[2] - 2 * error[1] + error[0]);

        print(adjust[2])
        # 饱和输出限制在 control 绝对值之内
        if abs(adjust[2]) > control:
            adjust[2] = sign(adjust[2]) * control
        # print(adjust[2])

        # 执行 PID
        print(adjust[2])
        # you 转
        if adjust[2] >= 20:
            pwma.ChangeDutyCycle((lspeed + adjust[2]*1.1))
            pwmb.ChangeDutyCycle(rspeed - adjust[2] * 0.83)
            print('turn right')

        # zuo 转
        elif adjust[2] <= -20:
            pwma.ChangeDutyCycle((lspeed+ adjust[2] * 0.81))
```

```
            pwmb.ChangeDutyCycle(rspeed - adjust[2]*1.15)
            print('turn left')

        elif adjust[2] < 20 and adjust[2] > -20:
            pwma.ChangeDutyCycle(90)
            pwmb.ChangeDutyCycle(90)
            print('go straight')




        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
except KeyboardInterrupt:
    print("结束！")
    pass
# 释放清理

cap.release()
cv2.destroyAllWindows()
pwma.stop()
pwmb.stop()
GPIO.cleanup()
# 按 Shift+F10 执行或将其替换为您的代码。
# 按 双击 Shift 在所有地方搜索类、文件、工具窗口、操作和设置。


def print_hi(name):
    # 在下面的代码行中使用断点来调试脚本。
    print(f'Hi, {name}')  # 按 Ctrl+F8 切换断点。


# 按间距中的绿色按钮以运行脚本。
if __name__ == '__main__':
    print_hi('PyCharm')
```

这一段代码具有较高的稳定性，在绕圈时偏离轨道的几率很小。但有以下两种问题

（1）在起跑阶段，极小的偏移量也会触发左右转，因此小车直行代码很少被使用导致小车运行速度较慢

（2）启动时，小车会因为微小偏移量而出发左转或者右转代码，因此从静止状态突然转弯导致起步慢，而且增加小车震荡的可能性。

代码二：
```
# 这是一个示例 Python 脚本。
# coding:utf-8
import RPi.GPIO as GPIO
import time
import cv2
import numpy as np
#前期准备部分

#限制较大值
def sign(x):
    if x > 0:
        return 1.0
    else:
        return -1.0

# 定义引脚
EA, I2, I1, EB, I4, I3 = (13, 19, 26, 16, 20, 21)
FREQUENCY = 50

# 设置 GPIO 口
GPIO.setmode(GPIO.BCM)

# 设置 GPIO 口为输出
GPIO.setup([EA, I2, I1, EB, I4, I3], GPIO.OUT)
GPIO.output([EA, I2, EB, I3], GPIO.LOW)
GPIO.output([I1, I4], GPIO.HIGH)

pwma = GPIO.PWM(EA, FREQUENCY)
pwmb = GPIO.PWM(EB, FREQUENCY)

# pwm 波初始化
pwma.start(0)
pwmb.start(0)

# center 定义
center_now = 320

# 打开摄像头，图像尺寸 640*480（长*高），opencv 存储值为 480*640（行*列）
cap = cv2.VideoCapture(0)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
```

```
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)

# PID 定义和初始化三个 error 和 adjust 数据
error = [0.0] * 3
adjust = [0.0] * 3

#PID 参数配置、目标值、左右轮基准占空比和占空比偏差范围（根据实际情况调整）
kp = 2.5
ki = 0.45
kd = 0.2
target = 320
lspeed = 55
rspeed = 55
control = 30
ret, frame = cap.read()

#前期准备完毕，节省时间
print("准备完毕！按下 Enter 启动！")
input()



try:
    while True:
        ret, frame = cap.read()

        # RGB 图像转换到 YCrCb 空间
        YCrCb_orange = cv2.cvtColor(frame, cv2.COLOR_BGR2YCrCb)
        # 提取 Cr 通道并高斯去噪，（3,3）表示高斯核大小
        Cr_orange = cv2.GaussianBlur(YCrCb_orange[:, :, 1], (3, 3), 0)
        # 二值化，160 为阈值
        ret, imgbinary = cv2.threshold(Cr_orange, 140, 255,
cv2.THRESH_BINARY)
        cv2.imshow('display', imgbinary)

        # 单看第 400 行的像素值 s
        color = imgbinary[400]
        # 找到 black 色的像素点个数
        black_count = np.sum(color == 255)

        # 防止 black_count=0 的报错
        if black_count == 0:
            continue
```

```
        else:
            orange_index = np.where(color == 255)
        # 找到黑色像素的中心点位置
        center_now = (orange_index[0][black_count - 1] +
orange_index[0][0]) / 2

        # 计算出 center_now 与标准中心点的偏移量
        direction = center_now - 320

        print("偏差值：", direction)

        # 更新 PID 误差
        error[0] = error[1]
        error[1] = error[2]
        error[2] = center_now - target

        # 更新 PID 输出（增量式 PID 表达式）
        adjust[0] = adjust[1]
        adjust[1] = adjust[2]
        adjust[2] = adjust[1] \
                    + kp * (error[2] - error[1]) \
                    + ki * error[2] \
                    + kd * (error[2] - 2 * error[1] + error[0]);

        print(adjust[2])
        # 饱和输出限制在 control 绝对值之内
        # print(adjust[2])

        # 执行 PID
        print(adjust[2])
        # you 转
        if adjust[2] >= 70:
            if abs(adjust[2]) > control:
                adjust[2] = sign(adjust[2]) * control
            pwma.ChangeDutyCycle((lspeed + adjust[2]*1.1))
            pwmb.ChangeDutyCycle(rspeed - adjust[2] * 0.83)
            print('turn right')

        elif adjust[2] >= 20 and adjust[2] < 70:
            if abs(adjust[2]) > control:
                adjust[2] = sign(adjust[2]) * control
            pwma.ChangeDutyCycle(99)
            pwmb.ChangeDutyCycle(99 - adjust[2] * 0.5)
            print('turn right')
```

```
        # zuo 转
        elif adjust[2] <= -70:
            if abs(adjust[2]) > control:
                adjust[2] = sign(adjust[2]) * control
            pwma.ChangeDutyCycle((lspeed+ adjust[2] * 0.81))
            pwmb.ChangeDutyCycle(rspeed - adjust[2]*1.15)
            print('turn left')

        elif adjust[2] <= -20 and adjust[2] > -70:
            if abs(adjust[2]) > control:
                adjust[2] = sign(adjust[2]) * control
            pwma.ChangeDutyCycle((99+ adjust[2] * 0.5))
            pwmb.ChangeDutyCycle(99)
            print('turn left')

        elif adjust[2] < 20 and adjust[2] > -20:
            pwma.ChangeDutyCycle(99)
            pwmb.ChangeDutyCycle(99)
            print('go straight')




        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
except KeyboardInterrupt:
    print("结束！")
    pass
# 释放清理

cap.release()
cv2.destroyAllWindows()
pwma.stop()
pwmb.stop()
GPIO.cleanup()
# 按 Shift+F10 执行或将其替换为您的代码。
# 按 双击 Shift 在所有地方搜索类、文件、工具窗口、操作和设置。


def print_hi(name):
    # 在下面的代码行中使用断点来调试脚本。
    print(f'Hi, {name}')  # 按 Ctrl+F8 切换断点。
```

```
# 按间距中的绿色按钮以运行脚本。
if __name__ == '__main__':
    print_hi('PyCharm')
```

对于小车的直行识别做了一定的优化：

（1）增大偏置量上限，让小车仅在偏置量较大的区域（拐弯处）运行左右转代码

（2）增设偏置量区间，让小车尽可能地运行直行代码，同时做一定的微调。

在本实验中，Kp Ki Kd三个参数对于小车运行影响并不大，因此有限调整占空比内的参数，使得小车运行更适应电机性能和实际赛道。

五、总结与思考

本次实验我们选择了与上次竞速实验不同的 PID 自动控制引导小车前进。此外实验中多次出现过硬件设备的问题，一步步排查各部分零件的状态以确定哪部分零件需要更换，加深了我们对硬件系统的认知。通过调节各项参数优化小车行驶以取得更好成绩。本次实验大大加深了我们的团队合作能力和细节排查能力，我们需要在各种不确定因素中尽量使小车的运行稳定下来，让每一次的运行都收敛于一个最佳结果。最后衷心对这门课的老师和助教们表示感谢！