电子系统导论实验报告

实验 【定时与计数】

指导教师： 王海鹏

学生姓名： 蔡亦扬 学生姓名： 周训哲 学生姓名： 孔恩燊

学号：23307130258 学号：20307110315 学号：23307130021

专业：技术科学试验班 专业：计算机 专业：技术科学试验班

日 期： 2024.5.6

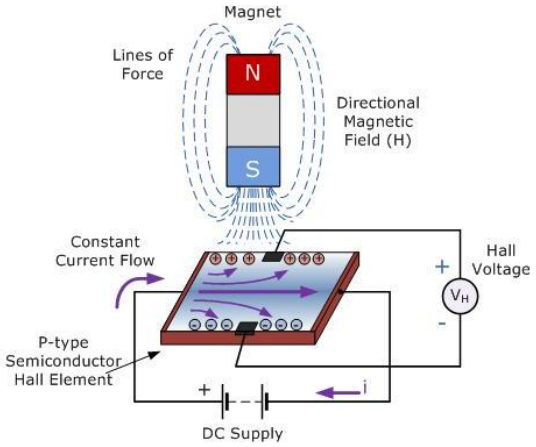
一、 实验目的：

1. 了解霍尔码盘的基本原理
2. 掌握树莓派定时计数的方法

二、 实验原理：

(一) 霍尔效应

1. 霍尔效应是电磁效应的一种，是美国物理学家霍尔于 1879 年在研究金属的导电机制时发现的。
2. 当电流垂直于外磁场通过半导体时，载流子发生偏转，垂直于电流和磁场的方向会产生一附加电场，从而在半导体的两端产生电势差，这一现象就是霍尔效应，这个电势差也被称为霍尔电势差。打个比方：好比一条路，本来大家是均匀的分布在路面上，往前移动。当有磁场时，大家可能会被推到靠路的右边行走。故路(导体)的两侧，就会产生电压差。
3. 霍尔效应产生的电子流动方向使用左手定则判断。



(二) 霍尔编码器

1. 霍尔编码器电机参数：
   额定电压：DC 6V
   工作电压：DC 5-13V
   工作电流：390mA
   传感器类型：霍尔式
   减速比：1:45（电机转 45 圈，车轮转 1 圈）
   分辨率：585 脉冲/车轮转 1 圈

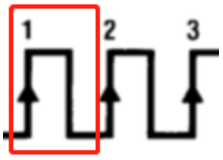
2. 霍尔编码器测速原理

1) 当电机转动时，电机后部的磁体跟随电机一起转动，根据霍尔效应，磁场变化将引起霍尔传感器电压的高低变化。

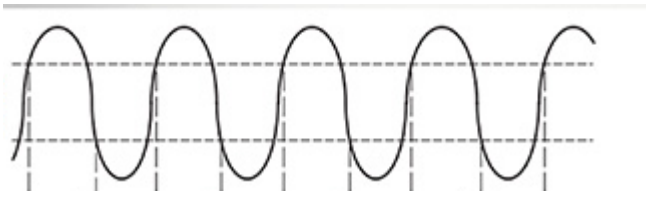
2) 该电压变化经过整形后，变成连续的高低电平变化——即方波信号。

3) 车轮旋转一圈，将产生 585 个高低交替的脉冲（下图红框为一个脉冲）。**可以用手动转一圈来大致确认脉冲数量。**

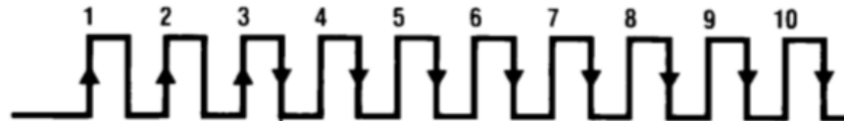
4) 通过计算时间 t 内监测到的上升沿数 n，可以算出车速：$v = \frac{pi*d*n}{585*t}$（d为车轮直径）

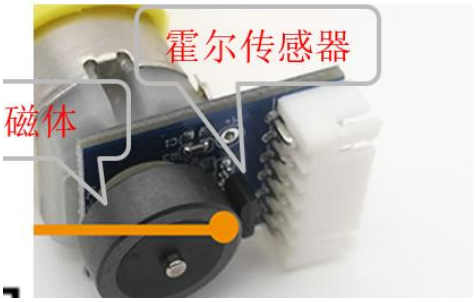


5) 霍尔编码器信号：



整形后信号：

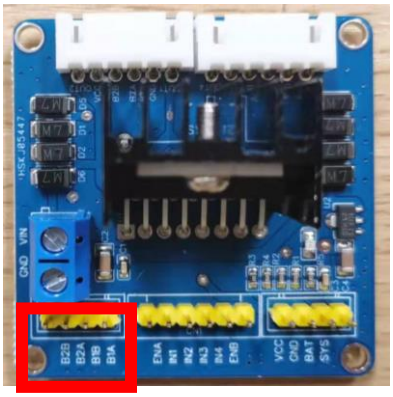




3. 树莓派与霍尔编码器连接

1) 引脚介绍

和编码器相关的引脚如下图红框所示：



从左至右为：B2B，B2A，B1B，B1A。

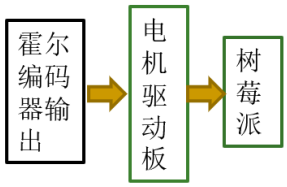
其中 B1A、B1B 是一个电机的两相输出，B2A、B2B 是另一个电机的两相输出，每组中任选一个就可以获取转速。

注意：B2A、B2B 对应被 ENA、IN1、IN2 三个管脚控制的电机；而 B1A、B1B 是对应 ENB、IN3、IN4 三个管脚控制的电机速度数据的输出口；明确对应关系，才能自由调节代码和硬件接线的对应关系。

2) 操作步骤

a) 连接时，先利用下发的白色排线，将电机和电机驱动板连接。

b) 连接 B2A、B1A 引脚和树莓派 GPIO6、GPIO12 引脚。



三、 实验内容：

通过转接板，将霍尔编码器和树莓派连接，并通过定时计数方式测量转速与 PWM 占空比的关系。

- 先连接单侧轮子的霍尔编码器同树莓派连线，测速绘图；
- 再连接双侧轮子的霍尔编码器同树莓派连线，测速绘图；

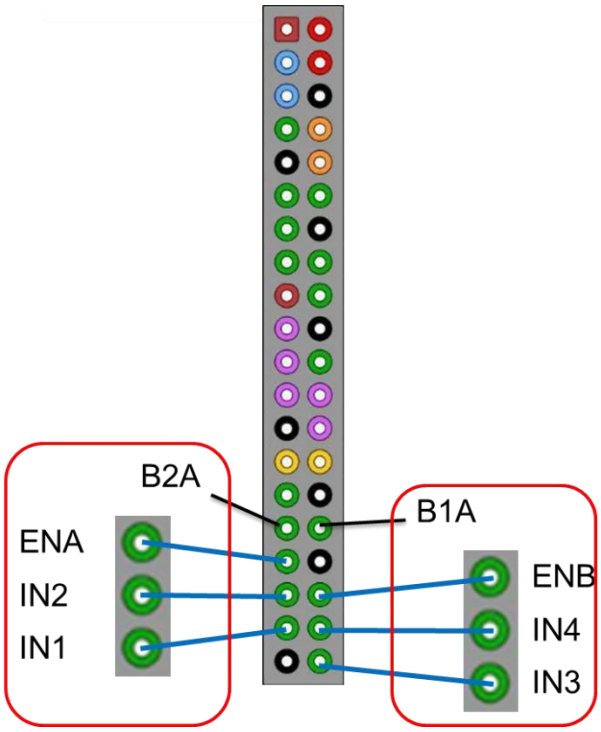
1. 电机连接：

本实验需要霍尔编码器测电机速度，因此需要能够控制电机。

电机驱动板的连接与第 9 节相同。

GPIO 连接效果如图，B2A 测 A 电机的速度，B1A 测 B 电机的速度。

实验中，A 电机连接右轮，B 电机连接左轮。



2. 代码执行：

1) 设置各 GPIO 与 pwm

```
import RPi.GPIO as GPIO
import time
import threading
import numpy
import matplotlib.pyplot as plt

EA, I2, I1, EB, I4, I3, LS, RS = (13, 19, 26, 16, 20, 21, 6, 12)
FREQUENCY = 50
GPIO.setmode(GPIO.BCM)
GPIO.setup([EA, I2, I1, EB, I4, I3], GPIO.OUT)
GPIO.setup([LS, RS],GPIO.IN)
GPIO.output([EA, I2, EB, I3], GPIO.LOW)
GPIO.output([I1, I4], GPIO.HIGH)

pwma = GPIO.PWM(EA, FREQUENCY)
pwmb = GPIO.PWM(EB, FREQUENCY)
pwma.start(0)
pwmb.start(0)
```

导入用于计时、多线程与绘图的库。

定义并初始化 GPIO 管脚。

设置用于控制两轮的 pwm，初始化为：频率：50Hz；占空比：0。

2) event_detected() 函数

功能：检测信号的上升沿和下降沿，并在检测到边缘时执行线程回调函数

使用方法：

```
def my_callback(channel)          //定义一个回调函数。
  global lcounter                 //引入全局变量lcounter与rcounter。
  global rcounter
  if (channel == LS):             //判断是哪个通道触发了调回，并给相应的
    lcounter += 1                 //counter加一。
  elif(channel==RS):
    rcounter += 1
GPIO.add_event_detect(LS,GPIO.RISING,callback=my_callback)
GPIO.add_event_detect(RS,GPIO.RISING,callback=my_callback)
//添加两个边沿检测，并调回my_callback
```

其中，GPIO.RISING 也可以使用 GPIO.FALLING、GPIO.BOTH 对边缘进行检测。

3) 测速函数

```
lspeed = 0
rspeed = 0
```

//设置全局变量 lspeed、rspeed，用于向主函数传递电机速度。

```
lcounter = 0
rcounter = 0
```

//lcounter 与 rcounter 用于记录从上一次被清零开始，两个霍尔传感器收到了多少个方波。

```
def my_callback(channel)
  global lcounter
  global rcounter
  if (channel == LS):
    lcounter += 1
  elif(channel==RS):
    rcounter += 1
```

//每当 LS 或 RS 收到一个上升沿，就会调用一次 my_callback ，使相应的 counter 增加一。

```
def getspeed():
    global rspeed
    global lspeed
    global lcounter
    global rcounter
    GPIO.add_event_detect(LS,GPIO.RISING,callback=my_callback)
    GPIO.add_event_detect(RS,GPIO.RISING,callback=my_callback)
    while True:
        rspeed=(rcounter/585.0) # Pulses gennerated per circle
        lspeed=(lcounter/585.0)
        rcounter = 0
        lcounter = 0
        time.sleep(1)
```

//getspeed 函数每隔一秒读取一次 counter 值并转换成速度传递给相应的 speed，然后将 counter 清零。

//注：“/585.0”是因为轮子转一圈会有 585 个脉冲，用“.0”是为了防止 speed 被自动取整。

4) 使用 Threading 模块创建线程

```
import threading
```

//导入 Threading 模块

```
thread1=threading.Thread(target=getspeed)
```

//创建新线程

```
thread1.start()
```

//开启线程

5) 测量 pwm 与车轮速度的关系

```
thread1=threading.Thread(target=getspeed)
thread1.start()
```

//开启 getspeed 函数为一个线程，它会不停的统计光电门输入的上升沿，并每隔一秒把全局变量更新为前一秒的速度。单位：圈/秒

```
i=0
x=[]
y1=[]
y2=[]
while i<=20:
    x.append(5*i)
    pwma.ChangeDutyCycle(5*i)
    pwmb.ChangeDutyCycle(5*i)
    time.sleep(3)
    y1.append(lspeed)
    y2.append(rspeed)
    i=i+1
```

//主函数每隔 3 秒增加一次 pwm 的占空比（本例中步长为 5%）。并读取一次新占空比下的两个 speed，存入两个数组中。

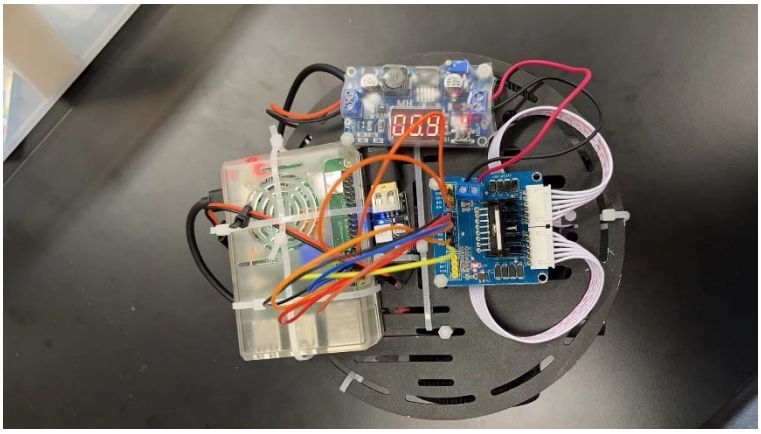
```
plt.plot(x,y1,'-o')
plt.plot(x,y2,'-*')
pwma.stop()
pwmb.stop()
GPIO.cleanup()
plt.show()
```

//显示出 lspeed 与 rspeed 关于 pwm 的关系图像。

//注：threading 没有提供停止线程的方法，关闭图像后可以使用^+z 结束程序。

四、 实验分析：

1. 电机连接：



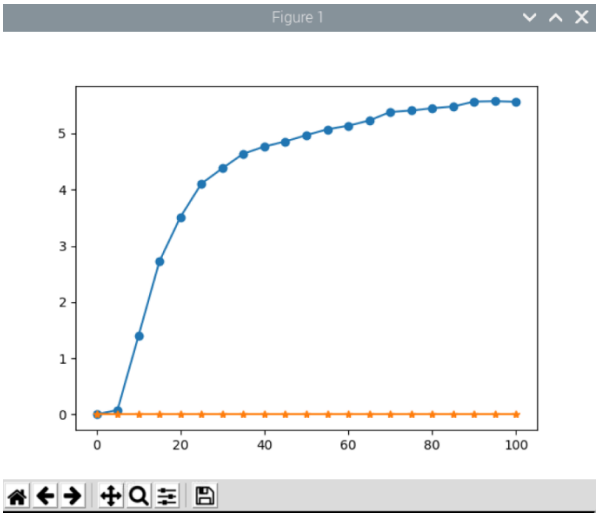
2. 绘制速度-pwm 图像

X 轴：pwm/%　　蓝色：左轮

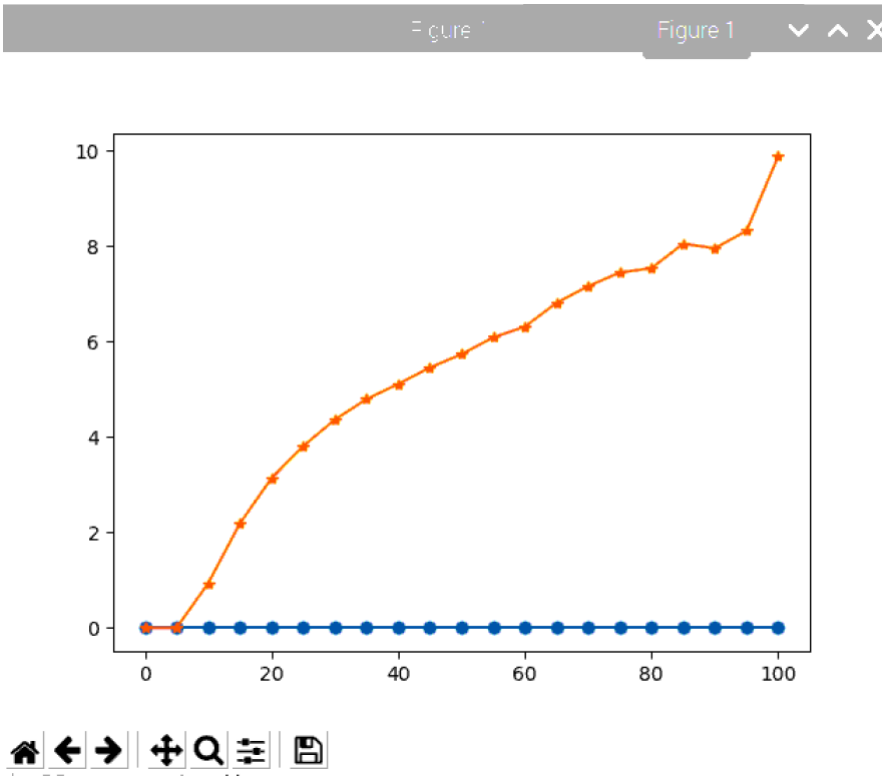
Y 轴：速度/（圈/秒）　　黄色：右轮

1) 实验一：连接单侧轮子的霍尔编码器同树莓派连线，测速绘图。

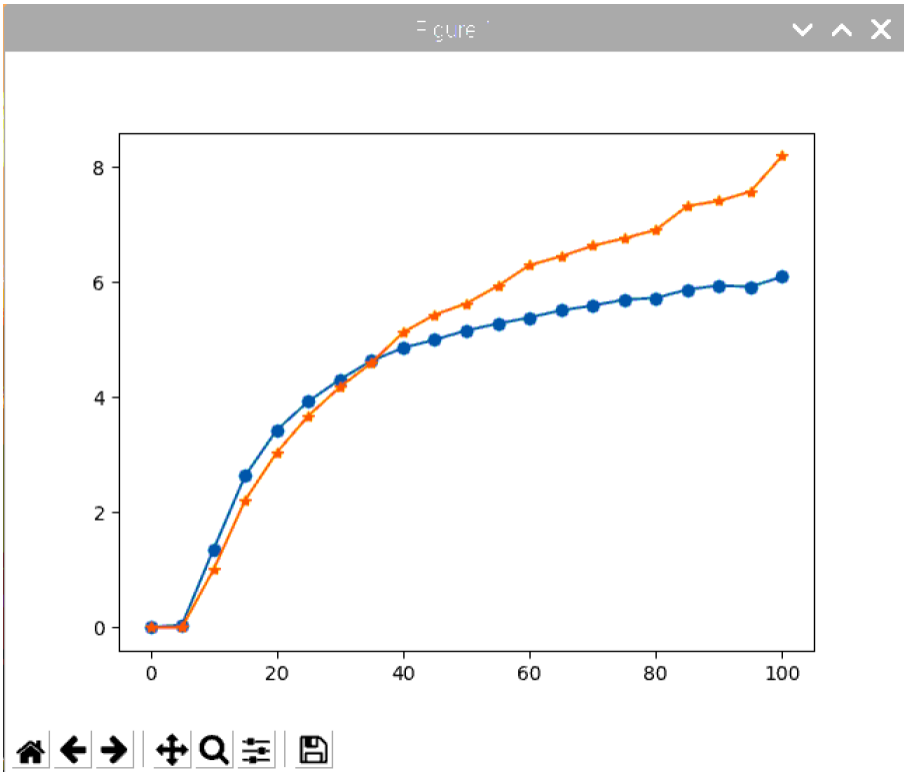
先连接左轮，不连接右轮，得到图像：



再连接右轮，不连接左轮，得到图像：



2) 实验二：连接双侧轮子的霍尔编码器同树莓派连线，测速绘图。得到图像如下：



说明右轮转速大于左轮。

五、 总结与思考：

总结：

通过用定时计数方式测量转速与 PWM 占空比的关系的实验，了解了霍尔码盘的基本原理，掌握了树莓派定时计数的方法。学会了根据具体情况调整接线、调整电机与轮子的对应关系，对后续的学习有很大的帮助。

思考：

1. **请问本次实验为什么需要采用多线程？能否只用单线程完成？**

   需要采用多线程的原因是需要同时进行 PWM 信号的输出和定时器的计数，这两个操作需要独立进行并且在同一时间内完成。如果使用单线程，我们只能在一段时间内执行一种操作，因此不能同时完成这两个任务，无法准确测量转速和 PWM 占空比之间的关系。使用多线程可以将这两个操作同时进行，提高测量的准确性和精度，因此采用多线程是必要的，且无法使用单线程进行。

2. **如果发现轮子不转，如何分步排查故障？**

1) 检查电源连接是否正确，电源电压是否正常。
2) 检查电机连接是否正确，电机线路是否断开或短路。
3) 检查程序是否正确，是否正确配置参数。
4) 检查编译是否成功，是否正确上传到树莓派。
5) 检查 PWM 输出是否正确，例如 PWM 频率、占空比等是否符合要求。
6) 检查电机驱动模块是否正常工作。
7) 检查电机是否正常工作，例如电机是否损坏或卡住。

3. **如果发现两个轮子转速差异很大，如何分步确定原因？**

1) 检查电机连接是否正确，电机线路是否断开或短路。
2) 检查程序是否正确，是否正确配置参数。
3) 检查硬件是否一致，例如两个电机是否型号一致、电机驱动模块是否一致等。
4) 检查电机功率是否足够，是否需要更高功率的电机来保证转速一致。
5) 检查两个电机的机械负载是否一致，例如是否因为机械负载不同导致转速差异。
6) 检查电机电压是否一致，例如是否因为电压不同导致转速差异。
7) 检查电机本身是否有损坏或老化，电机是否需要更换。