7.定时与计数

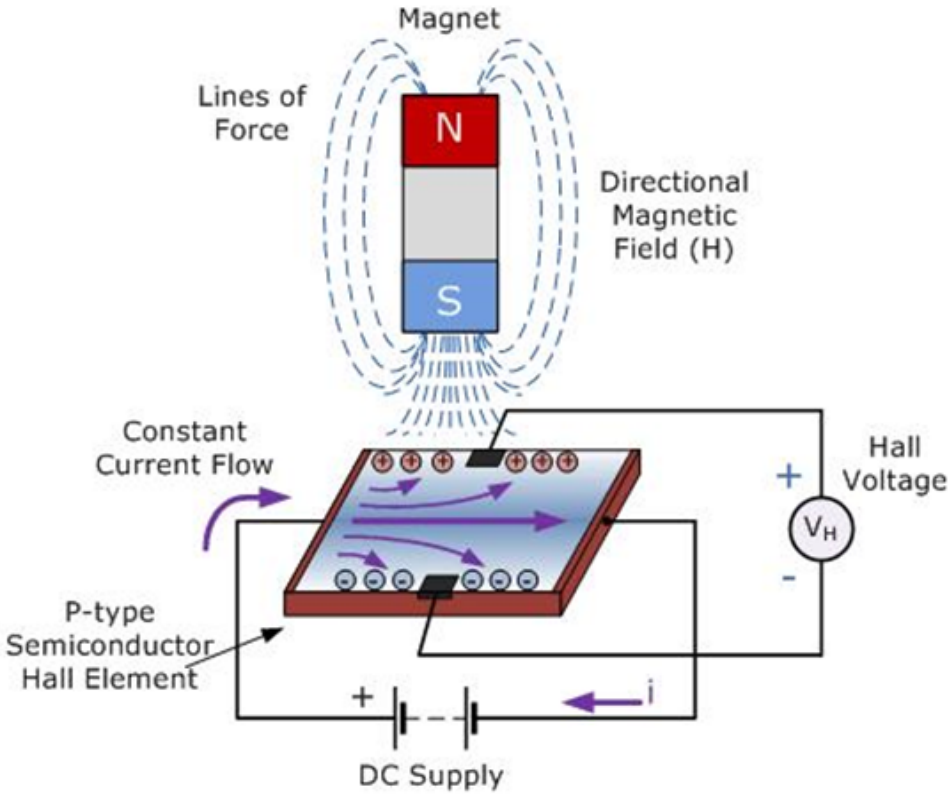
实验目的

- 了解霍尔码盘的基本原理
- 掌握树莓派定时计数的方法

reference

霍尔效应

- 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。
- 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。
- 霍尔效应产生的电子流动方向使用左手定则判断。



霍尔编码器电机

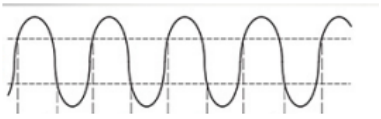
- 参数
  - 额定电压：DC 6V

- 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

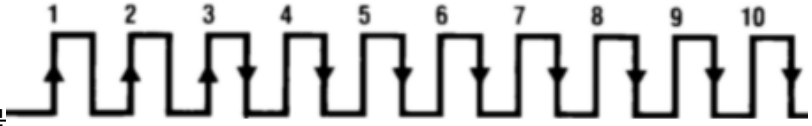
霍尔编码器测速原理

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

霍尔编码器信号



整形后信号

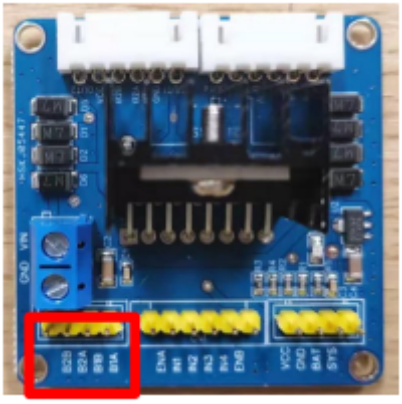


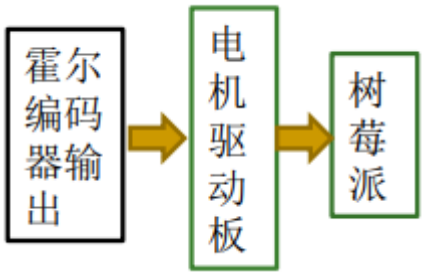
实验内容

树莓派与霍尔编码器连接

- 引脚介绍
  - 和编码器相关的引脚如图红框所示
  - 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则对应被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。

从左至右为：B2B，B2A，B1B，B1A

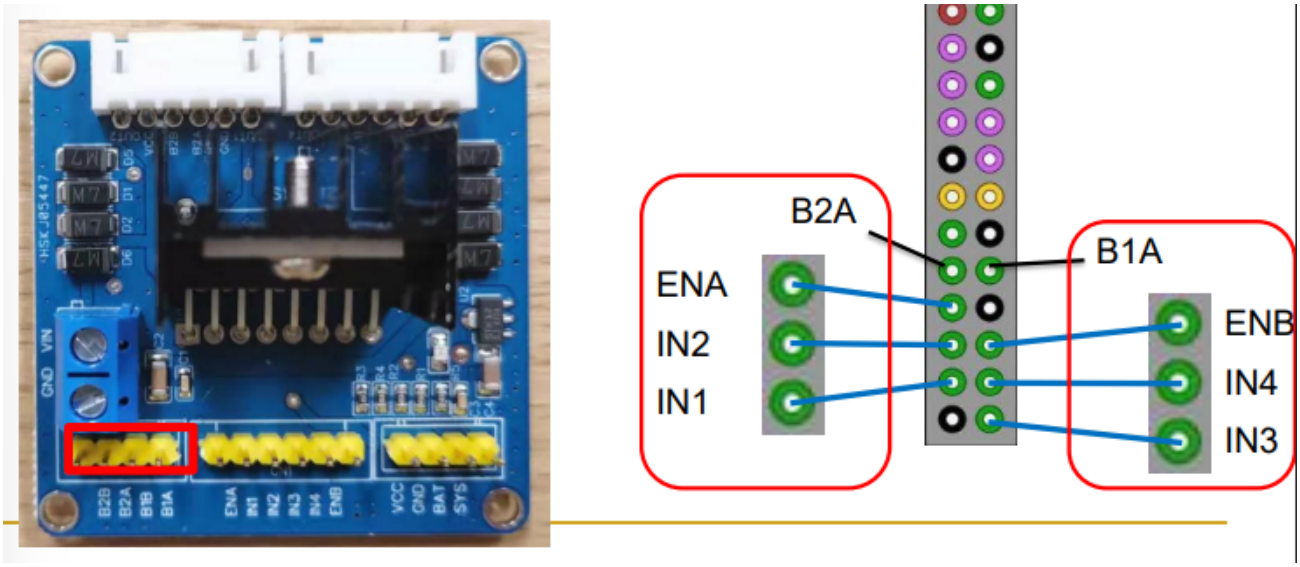




- 操作步骤
  - 连接时，先利用下发的白色排线，将电机和电机驱动板连接
  - 连接B2A、B1A引脚和树莓派GPIO6、GPIO12引脚

电机连接（一种参考接法）

- 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



A电机与B电机到底连接哪个轮子根据需求自行决定！

正式实验

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；

代码分析

```
# 库引入
import RPi.GPIO as GPIO
import time # 引入时间库
import threading # 引入线程库
import numpy
import matplotlib.pyplot as plt
# 接口定义与初始化
# 设置各个GPIO口与pwm
EA, I2, I1, EB, I4, I3, LS, RS = (13, 19, 26, 16, 20, 21, 6, 12)
FREQUENCY = 50 # 50Hz
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) # 占空比为0
pwmb.start(0)

lspeed = 0
rspeed = 0
lcounter = 0
rcounter = 0

# event detect函数
# 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
def my_callback(channel): # 定义回调函数
        global lcounter # 引入全局变量
        global rcounter
        if (channel==LS): # 判断是哪个通道触发了回调函数
            lcounter+=1 # 计数器加1
        elif(channel==RS):
            rcounter+=1

# 测速函数
def getspeed():
    global rspeed #设置全局变量lspeed、rspeed，用于向主函数传递电机速度
    global lspeed
    #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
    global lcounter
    global rcounter
    # 添加两个边沿检测，并调回my_callback
    # GPIO.RISING 也可以使用GPIO.FALLING、GPIO.BOTH 对边缘进行检测
    GPIO.add_event_detect(LS,GPIO.RISING,callback=my_callback)
    GPIO.add_event_detect(RS,GPIO.RISING,callback=my_callback)
    while True:
    #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。
        rspeed=(rcounter/585.0) # “/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
        lspeed=(lcounter/585.0)
```

```
        rcounter = 0
        lcounter = 0
        time.sleep(1)

thread1=threading.Thread(target=getspeed) # 创建新线程
thread1.start() # 启动线程
# 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
# 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

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

# 显示出lspeed与rspeed关于pwm的关系图像。
plt.plot(x,y1,'-o')
plt.plot(x,y2,'-*')
pwma.stop()
pwmb.stop()
GPIO.cleanup()
plt.show()
```

实验报告中需要回答的问题

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？

2. 如果发现轮子不转，如何分步排查故障？

3. 如果发现两个轮子转速差异很大，如何分步确定原因？