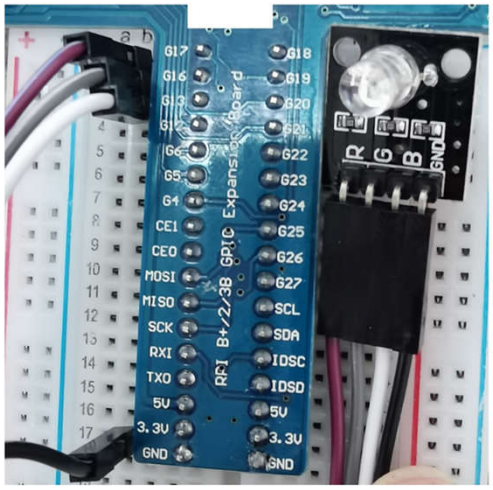
智能系统与控制

树莓派：PWM-RGB-LED

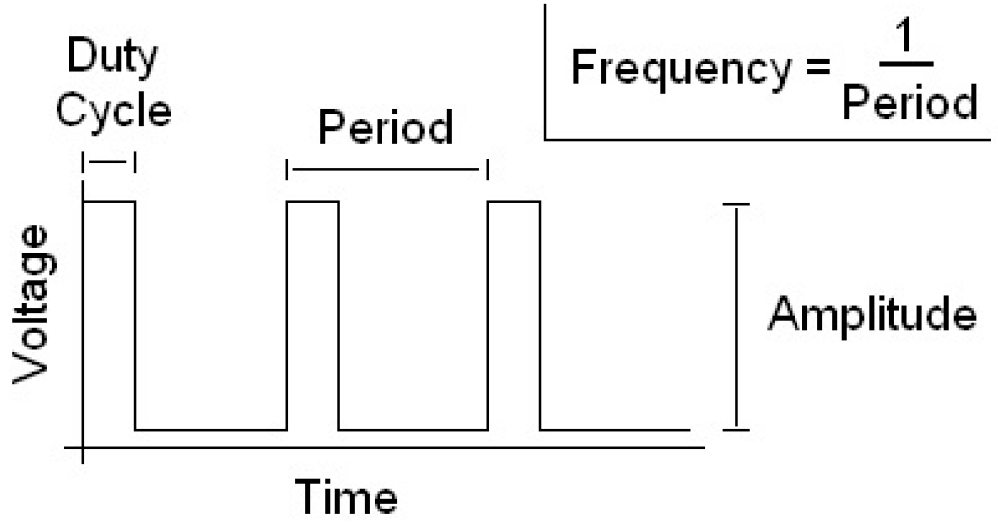
(Pulse-width modulation,脉冲宽度调制)



于泓
鲁东大学
信息与电气工程学院
2021.10.19

PWM的基本原理

- 脉冲宽度调制（PWM），是用脉冲信号对模拟信号进行近似的一种技术，一般变换后脉，冲的周期固定，但脉冲的工作周期（**Duty Cycle，即一个周期内高电平的比例**）会依所需模拟信号的大小而改变，高电平所占比例越高，拟合的模拟信号的幅度越大。



一个 PWM 信号是一系列的周期脉冲信号。

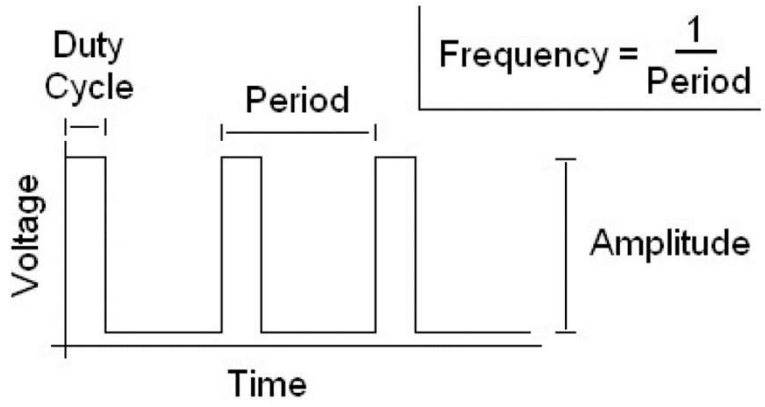
PWM 信号主要有两个参数：

（1）信号的周期长度（Period），通常用频率（Frequency）来描述。

Frequency = 1/ Period

（2）工作周期（Duty Cycle），该参数通常用占空比（Duty Ratio）来描述。

占空比 = **100*** Duty-Cycle / Period

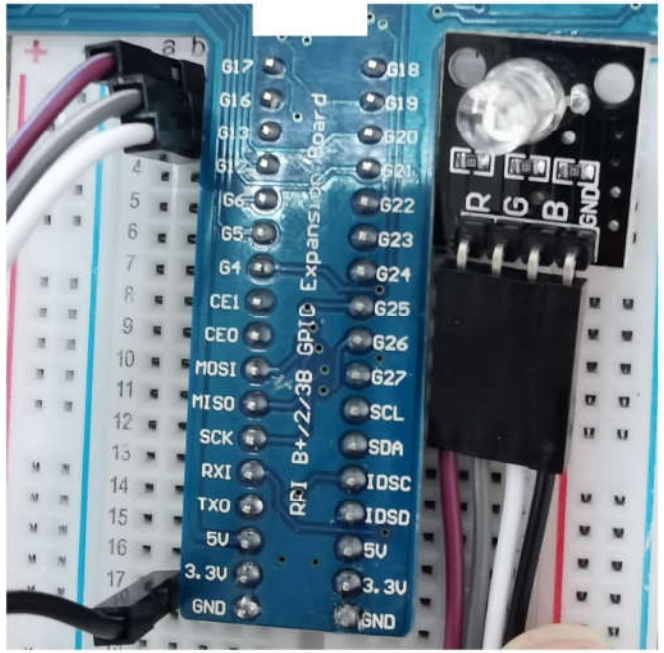


- PWM 技术通过使用高分辨率计数器（调制频率）调制方波的占空比，从而实现对一个模拟信号的电平进行编码。
- 最大的优点是从处理器到被控对象之间的所有信号都是数字形式的，无需再进行数模转换过程；而且对噪声的抗干扰能力也大大增强，这使得 PWM 在通讯等信号传输行业得到大量应用的主要原因。
- 模拟信号能否使用 PWM 进行编码调制，仅依赖带宽，这即意味着只要有足够的带宽，任何模拟信号值均可以采用 PWM 技术进行调制编码，一般而言，负载需要的调制频率要高于 **10Hz**，在实际应用中，频率约在 **1kHz 到 200kHz** 之间

PWM 控制 RGB-LED

三个引脚
电压的高低
决定了颜色
的成分

电压的高低
由PWM进行控制



树莓派引脚	RGB LED 模块
GPIO17	R
GPIO16	G
GPIO13	B
GND	GND

树莓派的任意引脚都可以被设置为PWM工作方式，普通引脚精度较低
专用引脚精度较高

保证代码支持中文

```
#-*- coding: utf-8 -*
import RPi.GPIO as GPIO
import time
from pin_dic import pin_dic

class RGB_LED(object):
    def __init__(self,pin_R,pin_G,pin_B):
        self.pins = [pin_R,pin_G,pin_B]
        # 设置为输出引脚，初始化第电平，灯灭
        for pin in self.pins:
            GPIO.setup(pin, GPIO.OUT)
            GPIO.output(pin, GPIO.LOW)

        # 设置三个引脚为pwm对象，频率2000
        self.pwm_R = GPIO.PWM(pin_R, 2000)
        self.pwm_G = GPIO.PWM(pin_G, 2000)
        self.pwm_B = GPIO.PWM(pin_B, 2000)

        # 初始占空比为0
        self.pwm_R.start(0)
        self.pwm_G.start(0)
        self.pwm_B.start(0)
```

设置为输出引脚

构造PWM对象
频率为2000hz

设置初始占空比吧

颜色转为占空比

```
def color2ratio(self,x,min_color,max_color,min_ratio,max_ratio):
    return (x - min_color) * (max_ratio - min_ratio) / (max_color - min_color) + min_ratio
```

颜色设置

```
def setColor(self,col):
    R_val,G_val,B_val = col

    R =self.color2ratio(R_val, 0, 255, 0, 100)
    G =self.color2ratio(G_val, 0, 255, 0, 100)
    B =self.color2ratio(B_val, 0, 255, 0, 100)
    # 改变占空比
    self.pwm_R.ChangeDutyCycle(R)
    self.pwm_G.ChangeDutyCycle(G)
    self.pwm_B.ChangeDutyCycle(B)

def destroy(self):
    self.pwm_R.stop()
    self.pwm_G.stop()
    self.pwm_B.stop()
    for pin in self.pins:
        GPIO.output(pin, GPIO.HIGH)
    GPIO.cleanup()
```

对象销毁

```python
if __name__ == "__main__":

    # 设置引脚编号模式
    GPIO.setmode(GPIO.BOARD)

    # 定义三个引脚
    pin_R = pin_dic['G17']
    pin_G = pin_dic['G16']
    pin_B = pin_dic['G13']

    # 定义 RGB_LED 对象
    m_RGB_LED = RGB_LED(pin_R,pin_G,pin_B)

    # 定义显示的颜色（R，G，B）
    colors = [(255,0,0),(0,255,0),(0,0,255),(255,255,0),(0,197,204),(192,255,62),(148,0,211),(118,238,00)];

    # 循环显示各种颜色
    try:
        while True:
            for col in colors:
                # 打印颜色
                print(col)
                # 设置颜色
                m_RGB_LED.setColor(col)
                # 延时
                time.sleep(3)
    except KeyboardInterrupt:
        print('\n Ctrl + C QUIT')
    finally:
        m_RGB_LED.destroy()
```