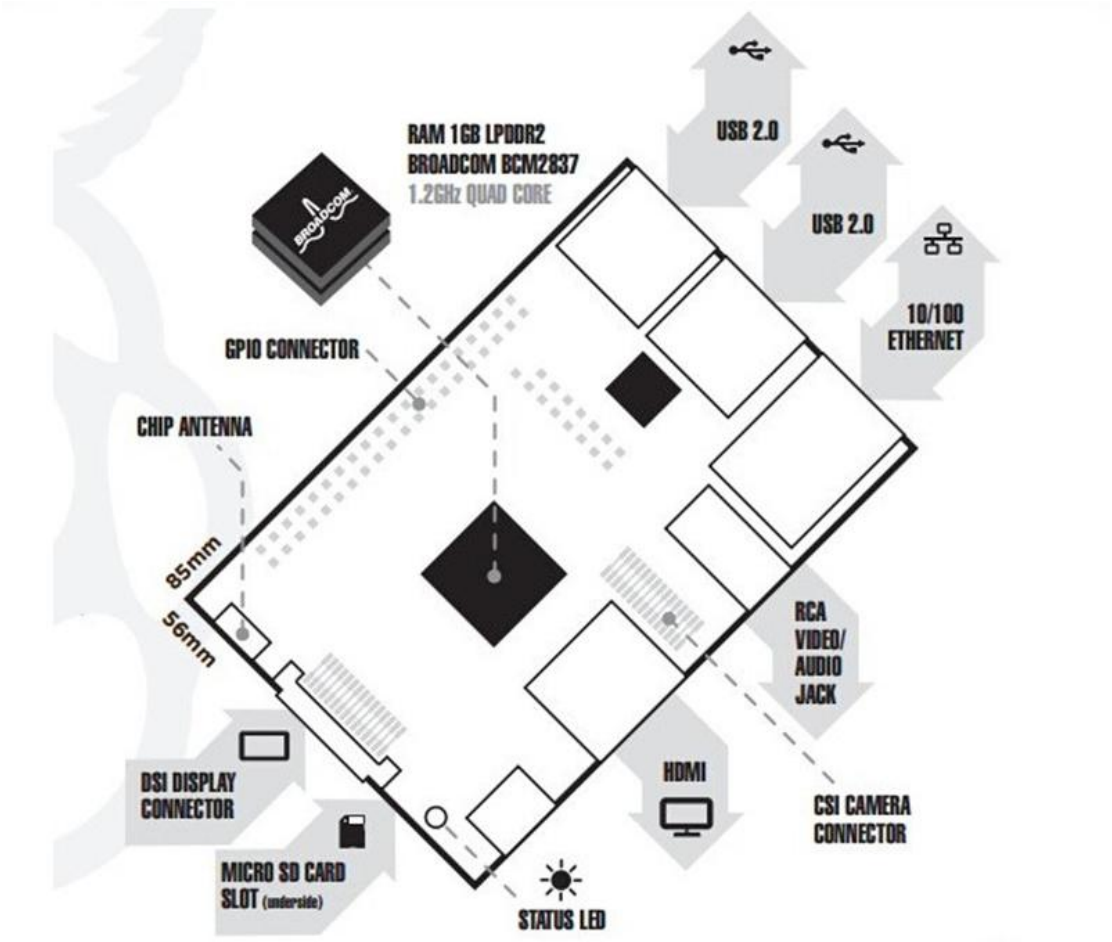
RPi.GPIO 使用手册

硬件：Raspberry Pi 3 Model B
系统：Raspbian
使用语言：Python3
RPi.GPIO 版本：0.6.2
RPi.GPIO 主页：https://sourceforge.net/p/raspberry-gpio-python/wiki/Home/
作者：汪志军
E-mail：gin0101@126.com
博客：https://my.oschina.net/ginnywzj/blog
编写时间：2016 年 10 月 27 日



1. 安装

在树莓派官方推荐的 Raspbian 系统中 RPi.GPIO 已经默认装好了。你可以输入下面指令更新 RPi.GPIO。

```
$ sudo apt-get update
$ sudo apt-get install python-rpi.gpio python3-rpi.gpio
```

测试 RPi.GPIO 是否安装成功。

```
$ python3
>>> import RPi.GPIO as GPIO
```

如果弹出 ImportError 错误则说明安装失败或者没有安装。可以通过指令（# pip install RPi.GPIO）安装 RPi.GPIO。

如果没有显示任何错误，则可以通过 print(RPi.GPIO) 打印 RPi.GPIO 模块文件路径。

2. 简介

RPi.GPIO 是 python 调用包，提供了一些方法来操作树莓派上 GPIO 引脚。使用 python 程序可以很方便的调用这些方法。目前 RPi.GPIO 提供了 GPIO 输入、输出和软件模拟 PWM 方法，可惜的是暂不提供 SPI、I2C、UART 和硬件 PWM 方法。python2 和 python3 都能调用 RPi.GPIO，在调用时请注意 python2 和 python3 语法区别，本手册以 python3 为例。

在调用输入、输出、PWM 功能前必须先导入封装库（import RPi.GPIO as GPIO）和设定 RPi.GPIO 引脚编号系统。

有两种编号系统可供选择：GPIO.BOARD 和 GPIO.BCM。

GPIO.BOARD：树莓派电路板上的编号，从 1 到 40，图 1 中 Pin#列对应的编号。

GPIO.BCM：BCM 芯片上的 GPIO 引脚编号，图 1 中 NAME 列对应 GPIO 的编号。

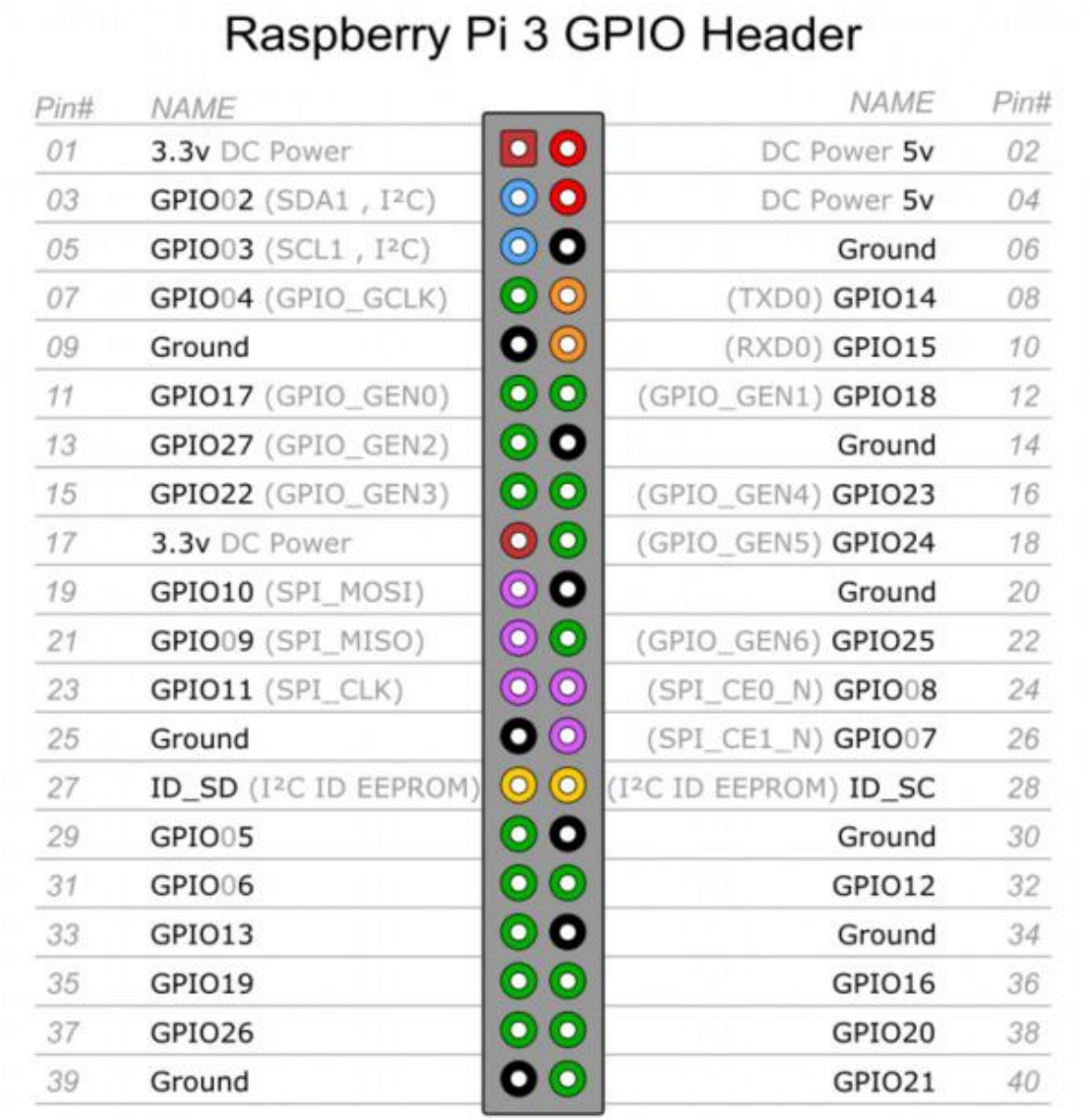


图 1

设置编号系统：

```
# GPIO.BOARD 编号系统
GPIO.setmode(GPIO.BOARD)

# GPIO.BCM 编号系统
GPIO.setmode(GPIO.BCM)
```

可以用：mode = GPIO.getmode() 查看编号系统。mode 的值只可能为 GPIO.BOARD、GPIO.BCM、None 三种情况中的一种。如果没有设定编号系统 mode 的值为 None。用 print 打印 GPIO.BOARD、GPIO.BCM、None 三个值，对应的显示是 10、11、None。

设置引脚输出输入模式：

setup 可以引脚的输入输出模式。

#输入模式

GPIO.setup(channel, GPIO.IN)

#输出模式

GPIO.setup(channel, GPIO.OUT)

注意：当为 BOARD 编号系统时，设置引脚模式只能对 GPIO 口设置不能是 DC Power 和 Ground 引脚，否则会弹出 ValueError: The channel sent is invalid on a Raspberry Pi 错误。设置时请参考图 1 引脚编号。

取消引脚设置模式警告：

当一个引脚已经被设定输入或输出模式时，当其他文件再次对它设定模式时就可能会弹出 RuntimeWarning: This channel is already in use,continuing anyway.Use GPIO.setwarnings(False) to disable warnings 警告。提示我们用 GPIO.setwarnings(False)消除警告。GPIO.setwarnings(False)应放在引脚模式设置前，否则无效。虽然不设置 GPIO.setwarnings(False)会有警告，但引脚的模式还是会被改变。

清除引脚设置：

把配置了的引脚模式清空初始化到默认状态。Raspberry Pi 3 Model B 引脚默认状态是输入模式，无上拉下拉电阻。在退出程序时应该清除引脚设置，让树莓派引脚恢复初始状态。

GPIO.clearup()　　#初始化所有引脚

#初始化指定引脚

GPIO.cleanup(channel)

GPIO.cleanup((channel1, channel2))

GPIO.cleanup([channel1, channel2])

GPIO.RPI_INFO： 获取树莓派的信息，下面是 Raspberry Pi 3 Model B 退回的信息。

{'MANUFACTURER': 'Sony', 'TYPE': 'Pi 3 Model B', 'P1_REVISION': 3, 'PROCESSOR': 'BCM2837', 'RAM': '1024M', 'REVISION': 'a02082'}

GPIO.VERSION：获取正在使用的 RPi.GPIO 版本（截止 2016 年 10 月 27 日最新版本为 0.6.2）。

3. 输入

配置方法：

```
GPIO.setup(channel, GPIO.IN)
# GPIO.IN 值为 1，所以也可以写成
GPIO.setup(channel, 1)
```

channel 可以是单个引脚（如 37），也可以是 list（如[35,37]）或者是 tuple（如(35,37)）。

获取输入值：

```
GPIO.input(channel)
```

channel 只能为单个引脚（如 37）。

上拉、下拉电阻配置：

```
GPIO.setup(channel, GPIO.IN, pull_up_down = GPIO.PUD_UP)
```

GPIO.PUD_UP 为上拉，GPIO_PUD_DOWN 为下拉。上拉下拉电阻为 10KΩ 。

提示：在设置边沿触发时可以预先设置好引脚电平的初始状态，上升沿触发设置引脚为下拉，下降沿触发设置引脚为上拉。

边沿触发：

等待边沿触发：

```
GPIO.wait_for_edge(channel, GPIO.RISING)
```

GPIO.RISING（上升沿）可以换成 GPIO.FALLING（下降沿）或者 GPIO.BOTH（上升沿和下降沿）。

wait_for_edge 方法会阻止它下面的程序执行，系统会一直等待它被上升沿或下降沿触发。为了防止程序长时间等待，我们可以给它一个超时时间 timeout（单位 ms）。

```
channel = GPIO.wait_for_edge(channel, GPIO_RISING, timeout=5000)
```

如果在 timeout 时间内没有触发则退回的 channel 值为 None，如果有触发则退回被触发的引脚编号。

边沿事件检测：

```
GPIO.add_event_detect(channel, GPIO.RISING)
do_something()
if GPIO.event_detected(channel):
        print('Button pressed')
```

event_detected()需要和 add_event_detect()联合使用。只有当上升沿或下降沿发生在 GPIO.event_detected(channel)执行前，才能被检测到。

回调函数：

```
GPIO.add_event_detect(channel, GPIO.RISING, callback=my_callback)
```

my_callback 就是我们的回调函数。

```
def my_callback_one(channel):
    print('Callback one')
def my_callback_two(channel):
    print('Callback two')
GPIO.add_event_detect(channel, GPIO.RISING)
GPIO.add_event_callback(channel, my_callback_one)
GPIO.add_event_callback(channel, my_callback_two)
```

每次检测到触发时，my_callback_one 和 my_callback_two 会按添加的顺序执行，上面代码先 my_callback_one 后 my_callback_two，而不是同时执行。回调函数可以实时被触发，一旦有上升沿或下降沿就会被执行，前提是整个程序还在运行。

消抖处理：

当检测上升沿或下降沿时，检测函数可能会被触发多次，这是因为电流波形有抖动。我们可以在检测函数中加入防抖延时时间 bouncetime（单位 ms），类似于开关按钮硬件设计中的软件防抖。

```
GPIO.add_event_detect(channel, GPIO.RISING, callback=my_callback, bouncetime=200)
```

移除检测：

```
GPIO.remove_event_detect(channel)
```

4. 输出

配置方法：

```
GPIO.setup(channel, GPIO.OUT)
# GPIO.OUT 值为 0，所以也可以写成
GPIO.setup(channel, 0)
```

channel 可以是单个引脚（如 37），也可以是 list（如[35,37]）或者是 tuple（如(35,37)）。

控制引脚输出值：

```
#输出高电平（3.3V）
GPIO.output(channel,GPIO.HIGH)# GPIO.HIGH 值为 1
#输出低电平（0V）
GPIO.output(channel,GPIO.LOW)# GPIO.LOW 值为 0
```

channel 可以是单个引脚（如 37），也可以是 list（如[35,37]）或者是 tuple（如(35,37)）。也可以同时给多个引脚赋值

GPIO.output([channel0,channel1],[GPIO.LOW,GPIO.HIGH])

GPIO.input()可以读取引脚输出值，所以可以使用

GPIO.output(channel, not GPIO.input(channel))

来翻转引脚电平值。

5. PWM

设置 PWM 前需要把对应引脚设置为输出模式。

设置 PWM 频率：p = GPIO.PWM(channel, frequency)

开启 PWM 并设置占空比（高电平占的比重）：p.start(dc)　#0.0 <= dc <= 100.0

修改 PWM 频率：p.ChangeFrequency(freq)

修改占空比：p.ChangeDutyCycle(dc)　#0.0 <= dc <= 100.0

关闭 PWM：p.stop()

```
p = GPIO.PWM(channel, 2)
p.start(60)
```

说明：PWM 频率为 2，则一秒引脚电平值刷新 2 次。占空比为 60，则高电平比重为 60%（占空比越大高电平占的比重越大）。如果在引脚上接一个 LED（要串联一个电阻）时，则 LED 每秒钟闪烁 2 次。图 2 是 PWM 在引脚上的电平变化时序图。

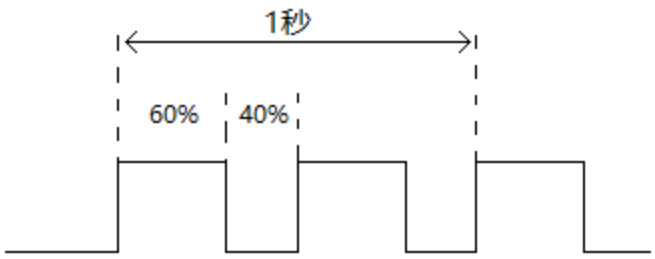


图 2

6. 举例

A. 设置引脚编号方式为 BOARD，并把 35 引脚设置为输入上拉模式，打印出 35 引脚的输入值。

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setup(35, GPIO.IN, pull_up_down = GPIO.PUD_UP)
value = GPIO.input(35)
print(value)
GPIO.cleanup()
```

B. 设置引脚编号方式为 BCM，并把 GPIO19 与 GPIO26 引脚设置为输出模式，GPIO19 输出 0，GPIO26 输出 1。

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup([19, 26], GPIO.OUT)
GPIO.output([19, 26], [GPIO.LOW,GPIO.HIGH])
```

C. 设置引脚编号方式为 BOARD，并检测 37 引脚下降沿，当有下降沿触发时翻转 35 引脚的电平。（Ctrl+c 停止程序运行）

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
GPIO.setup(37, GPIO.IN, GPIO.PUD_UP)
GPIO.setup(35, GPIO.OUT)

def callback_fun(channel):
    GPIO.output(35, not GPIO.input(35))
GPIO.add_event_detect(37, GPIO.FALLING, callback_fun, bouncetime=200)

while True:
    time.sleep(1)

GPIO.cleanup()
```

D. 设置引脚编号方式为 BOARD，在 35 引脚配置一个呼吸灯。（Ctrl+c 停止程序运行）

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD)
GPIO.setup(35, GPIO.OUT)
p = GPIO.PWM(35, 50)
p.start(0)
try:
    while 1:
        for dc in range(0, 101, 5):
            p.ChangeDutyCycle(dc)
            time.sleep(0.1)
        for dc in range(100, -1, -5):
            p.ChangeDutyCycle(dc)
            time.sleep(0.1)
except KeyboardInterrupt:
    pass
p.stop()
GPIO.cleanup()
```