树莓派开发

14 树莓派GPIO接口体验

一、开关信号与功率

- 在计算机眼里，所有信号只有1（通）和0（断），这就是所谓的数字信号。电源开关可以直接对应数字信号。
- 相对的，像温度值这样的叫做模拟信号， 它比数字信号要复杂，需要额外的设备（A/D、D/A数模转换)来处理。
- 树莓派的GPIO接口只处理数字信号，如果是模拟信号，需要先用A/D转换，才能接受，反之用D/A。
- Raspberry Pi有一套叫做GPIO的接口，即通用接口，你可以将它们用于任何你希望的目的。
- 这些GPIO可以用作对外部世界的input和ouput。
- 需要注意的是 Raspberry Pi上的GPIO是低功率（电压是固定的3.3V，体现为电流大小）的，所以有时你可能需要一块扩展板，借助额外的供电，来满足需要高功率的场合。

二、实验准备

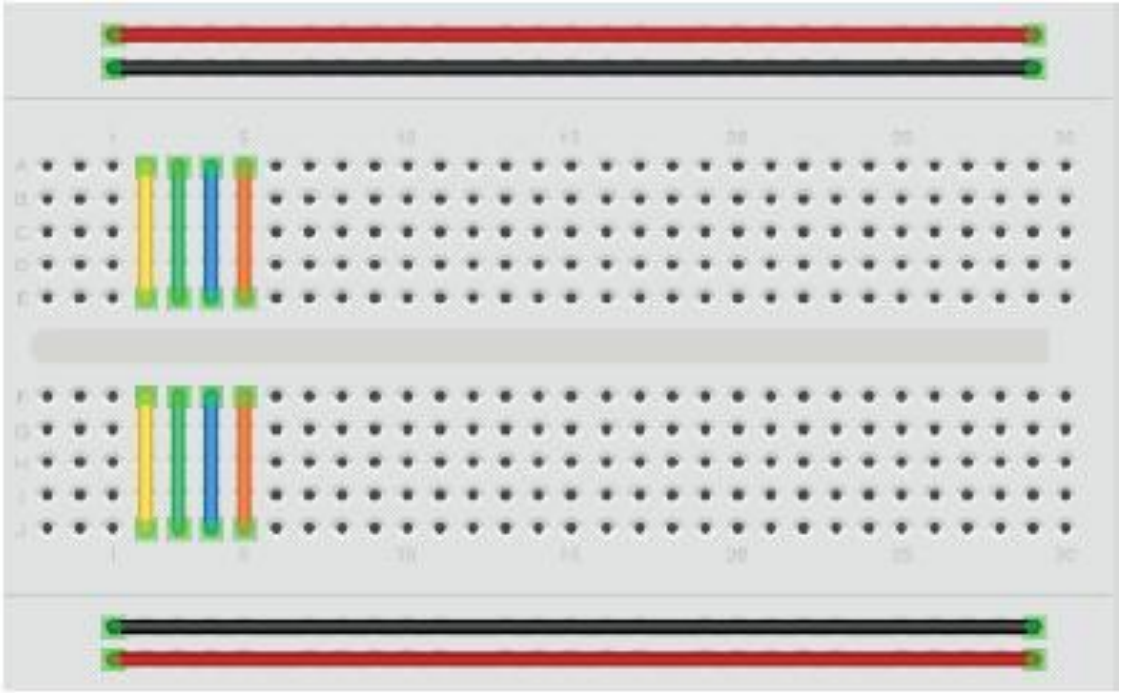
- 下面来进行一个简单的实验，这个实验仅需要一个开关和LED灯。
- 开关：我们需要一个像下图的按键开关。一般开关可以分为两种，闭锁开关（latching switch）和瞬时开关（momentary switch）。
- 前者可以保持开或关的状态，就像电灯开关那样；后者则只有你按下它的时候才接通，比如键盘按键。这里我们用的是一个普通的轻触开关（后者）。
- LED：发光二级管非常常见，它们通常被用作指示灯。这里是一支普通的3mm LED灯

二、实验准备

- 为方便实验，还需要一块面包板和一些导线。
- 它们可以使你非常方便的连接和修改电路。
- 还需要10千欧、1千欧和47欧的电阻三支。
- 最后，还需要三条母对公杜邦线，用来连接Raspberry Pi和面包板。

二、实验准备-面包板

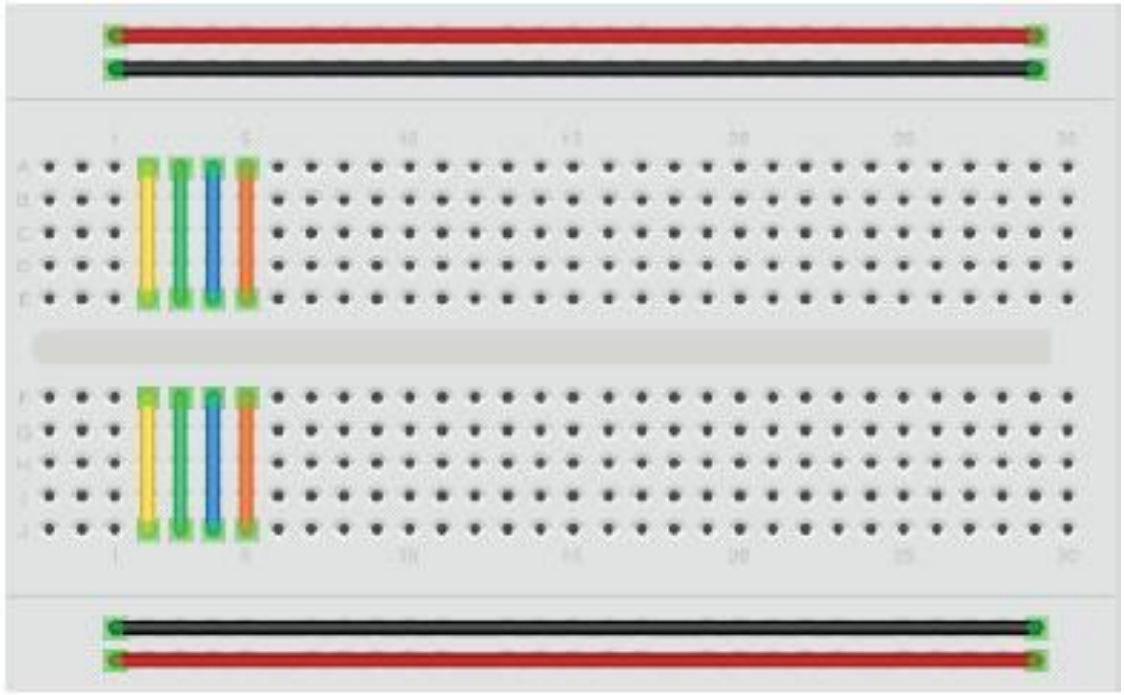
- 我们经常用面包板进行电子电路的原型系统构建与测试工作。
- 从表面看，面包板是排列为栅格（grid）的许多插孔，而在面包板的底部，这些插孔通过一条条的导电金属条连接到一起。
- 金属条典型的排列方式如下图所示：



二、实验准备-面包板

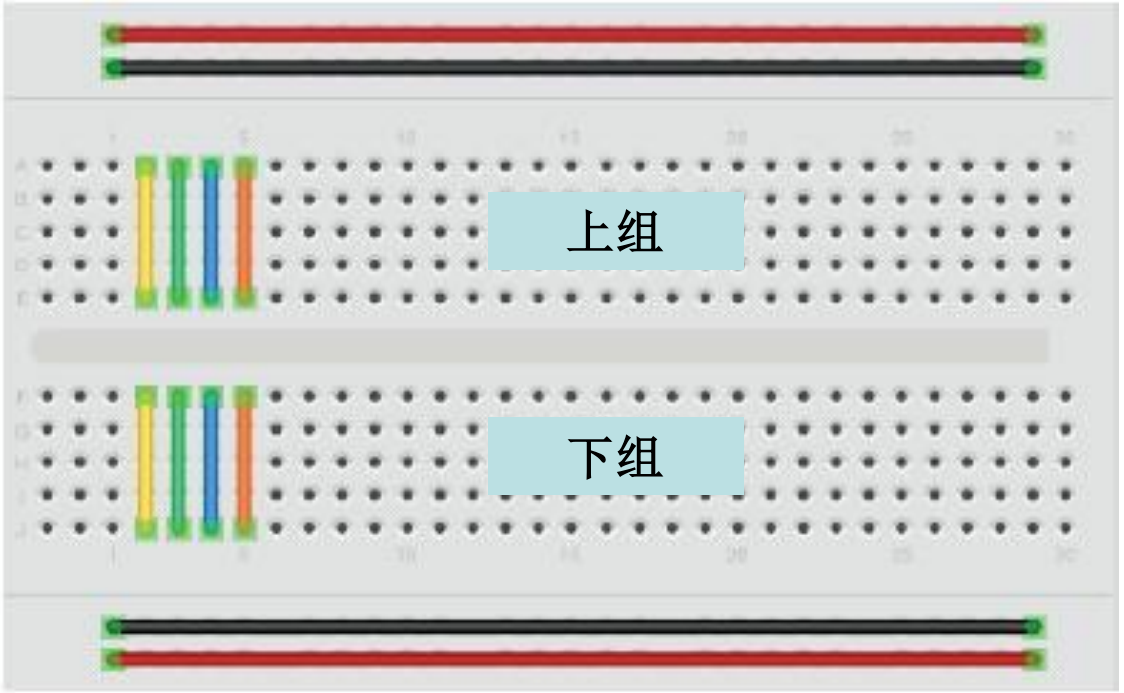
- 在面包板的最上部和最下部分别有两条平行走向的金属条，通常我们将它们分别设置为电源线（红色）和地线（黑色）。
- 上部或下部的电源线是联通的，也就是说，在上部或下部一行的任何一个孔插入，都可以接到电源正或地线上。

- 这样，面包板中部的电子器件可以方便的“就近”连接到电源（5V等）或者地。



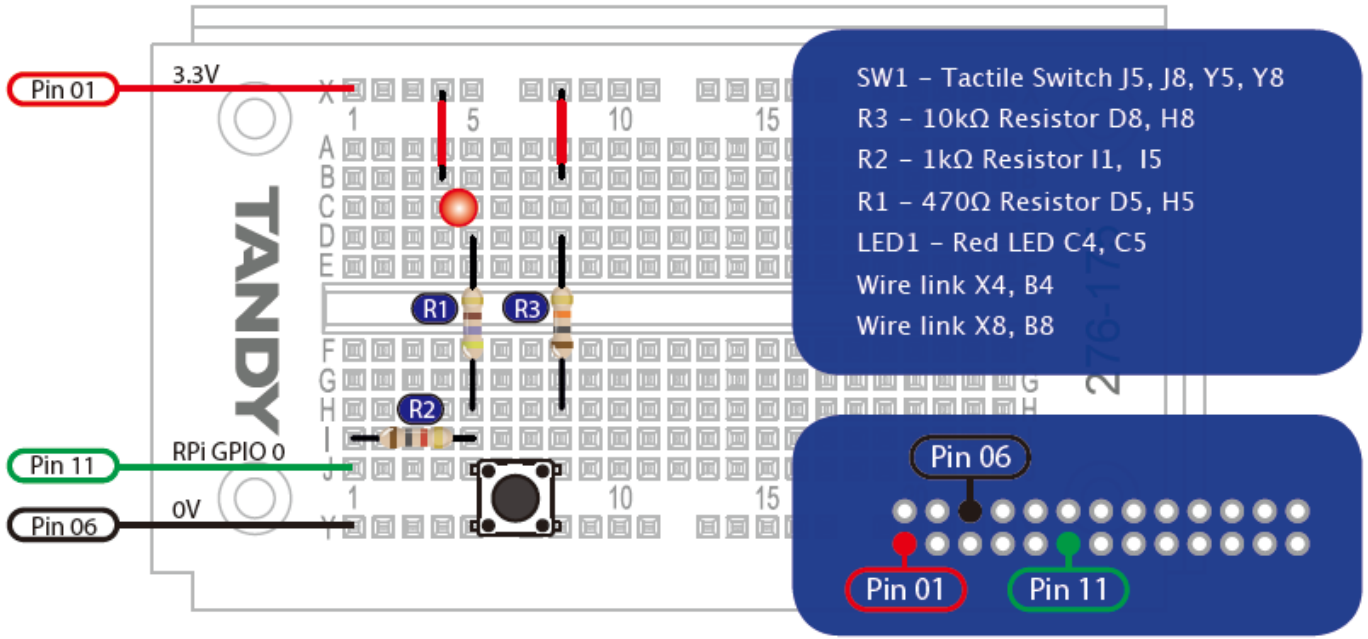
二、实验准备-面包板

- 在面包板二组（上下）电源条之间的部分，是放置电子元器件的“中间地带”。
- 中间地带分为上下二组，上组和下组的同列5个插孔是联通的，而同行是不通的（如图彩色连线）。
- 所以，元器件放置或者是“跨列”，或者是同列“跨组”。
- 否则是无效的。
- 大的面包板则还会分为更多的“组”



二、实验准备-面包板

- 以下面要做的项目为例：
  - R1、R3是同列跨组，而R2是跨列，LED则是跨行跨列
  - 两根电源线（正）从电源引向元器件，而开关则跨在地和元器件之间。
  - 另外3根则连接面包板和Pi



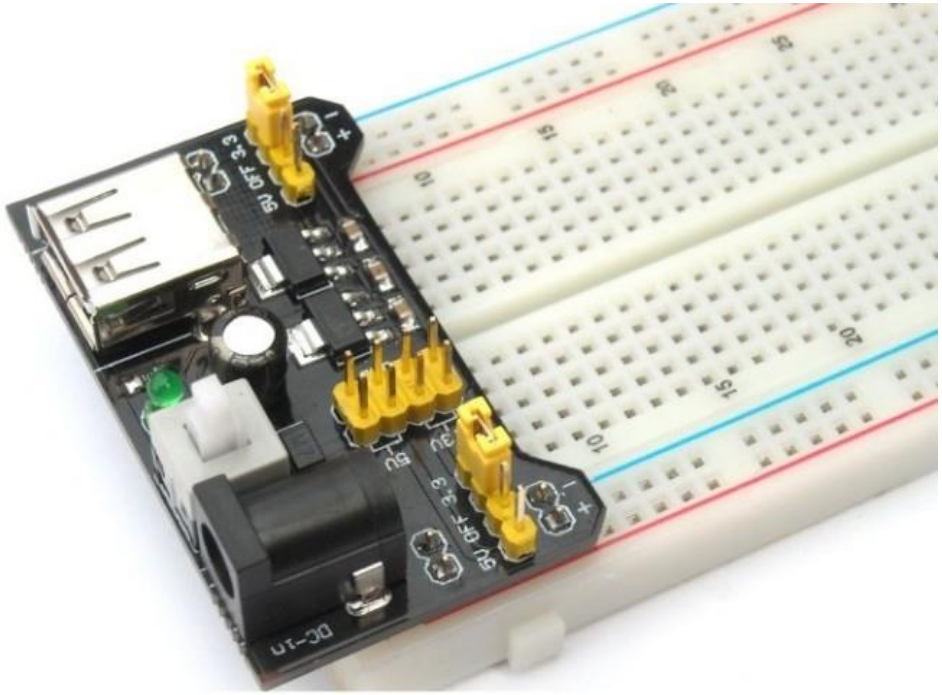
二、实验准备-面包板电源

- 面包板的外接电源：
- 面包板也可以不用树莓派的电源，而是用一个专用的电源模块，这是兼容树莓派、可提供5V、3.3V的电源

- 这个是单独给面包板供电，不用再从树莓派引出电源，最大的好处：减少烧坏树莓派的风险

二、实验准备-面包板电源

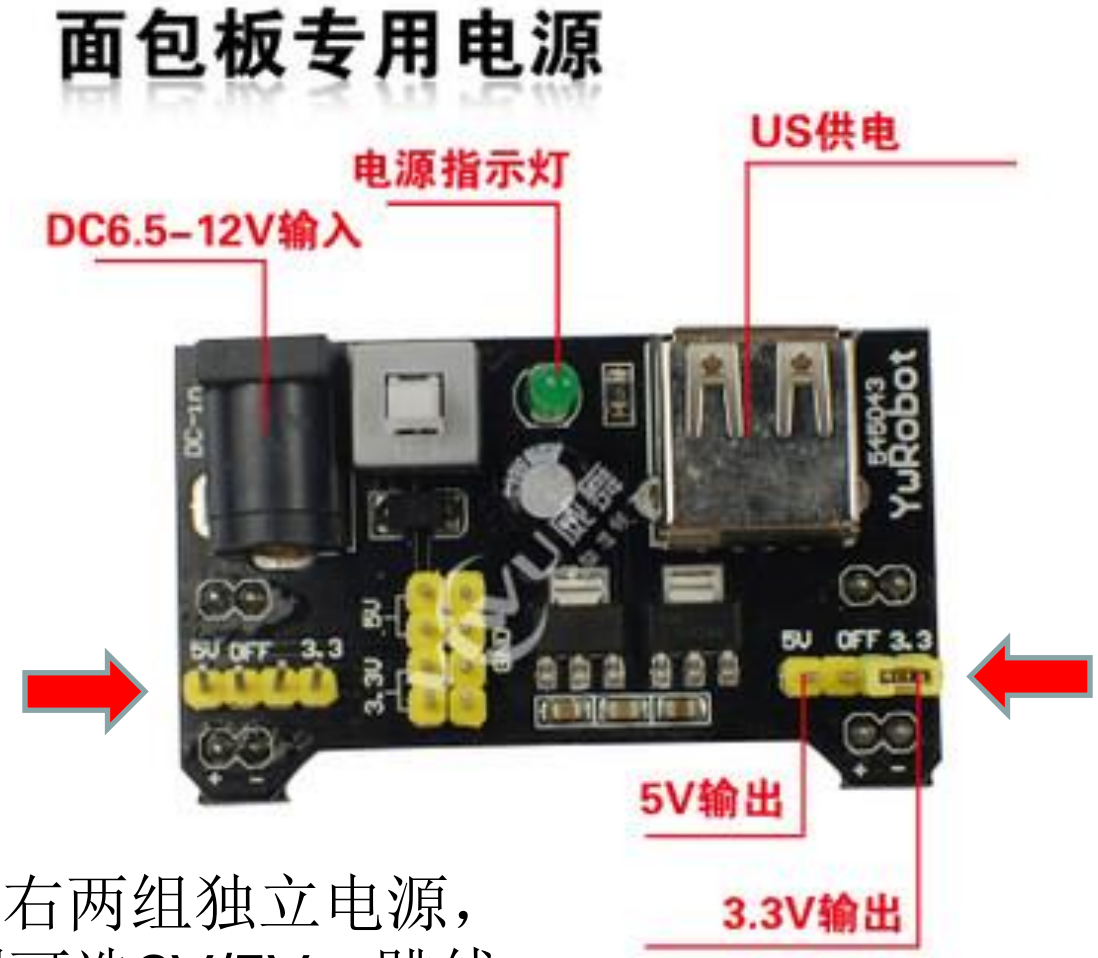
- 电源参数：
  - 输入电压：6.5-12V（直流）或USB供电
  - 输出电压：3.3V、5V可切换
  - 最大输出电流：<700ma
  - 上下两路两路独立控制，可切换为0V、3.3V、5V
  - 板载两组3.3V、5V直流输出插针



面包板与独立电源的连接

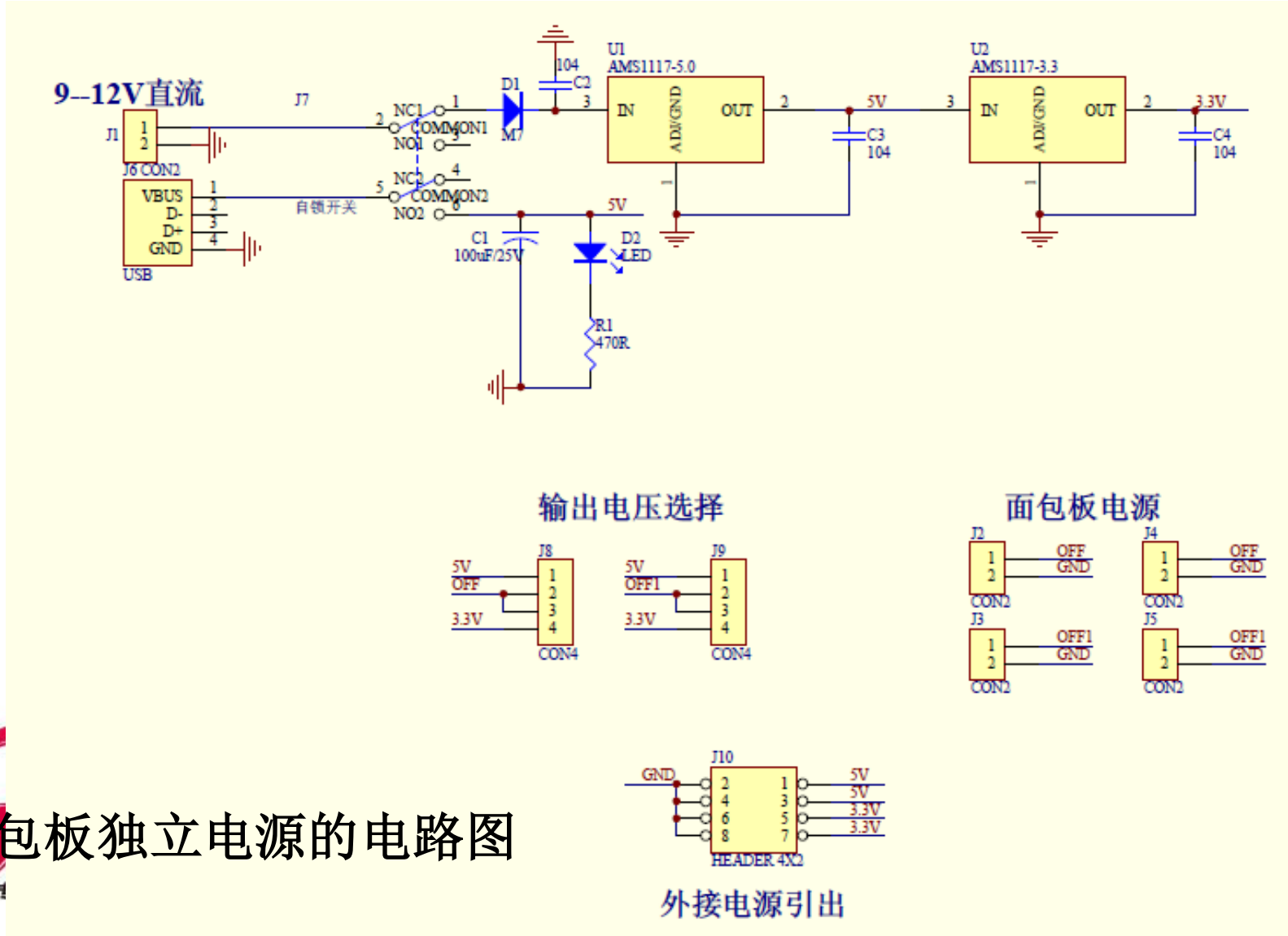
二、实验准备-面包板电源

□接线说明：



□左右两组独立电源，分别可选3V/5V，跳线连接为输出电压

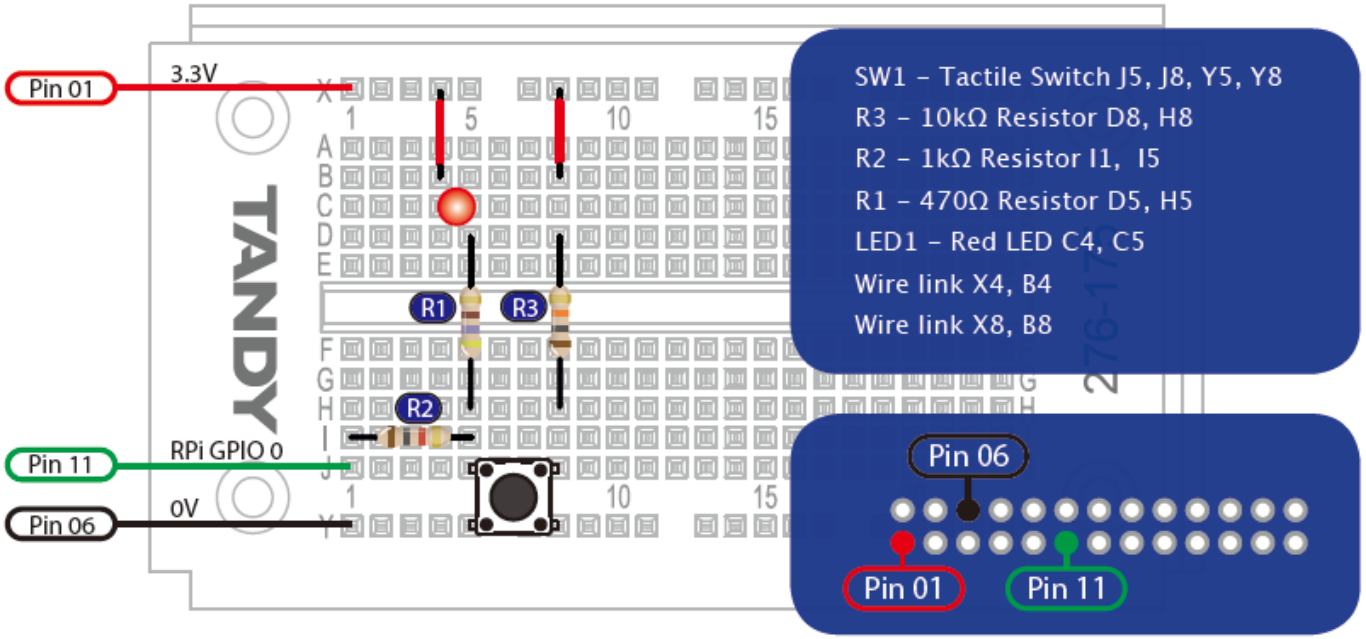
二、实验准备-面包板电源



面包板独立电源的电路图

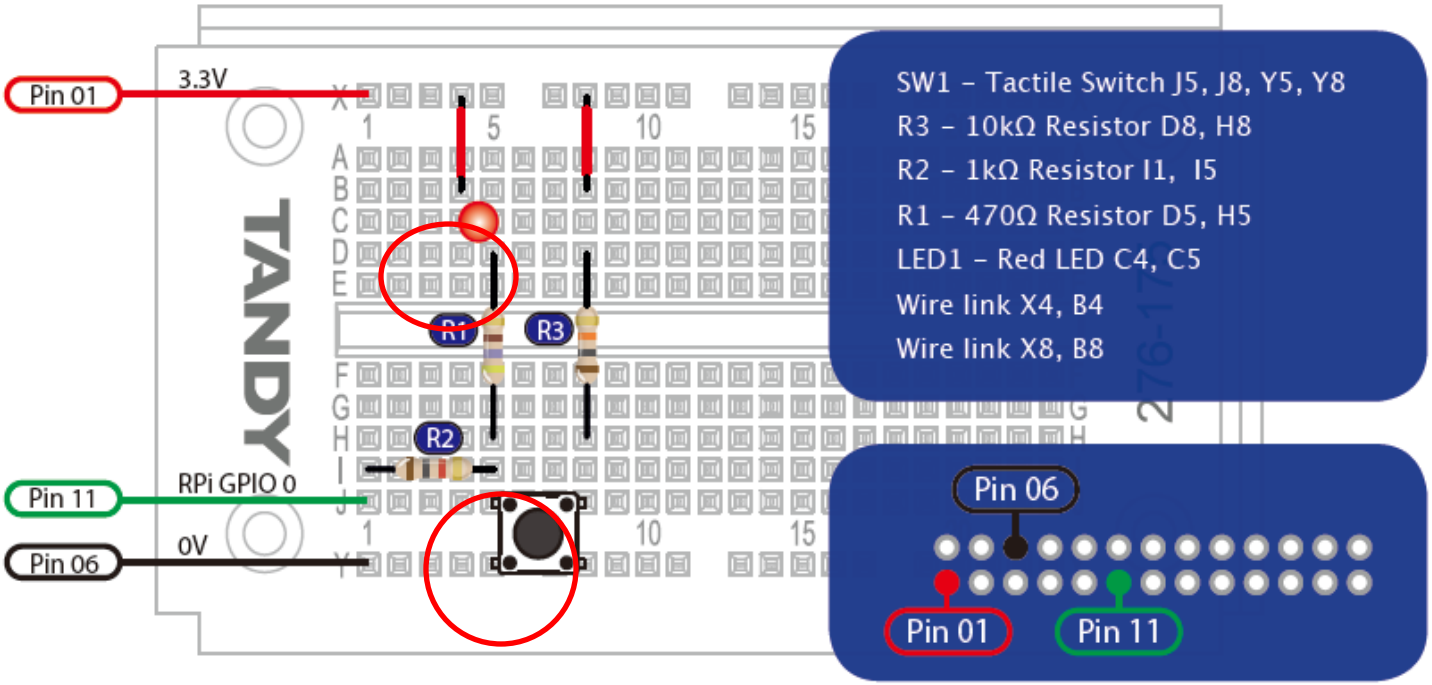
二、实验准备

- 按照下图，把所有元件插上面包板：
  - 1个LED灯、1个开关
  - 3个电阻：R1、R2、R3
  - 2根板上连接导线
  - 3根外接导线（树莓派一头暂不管）



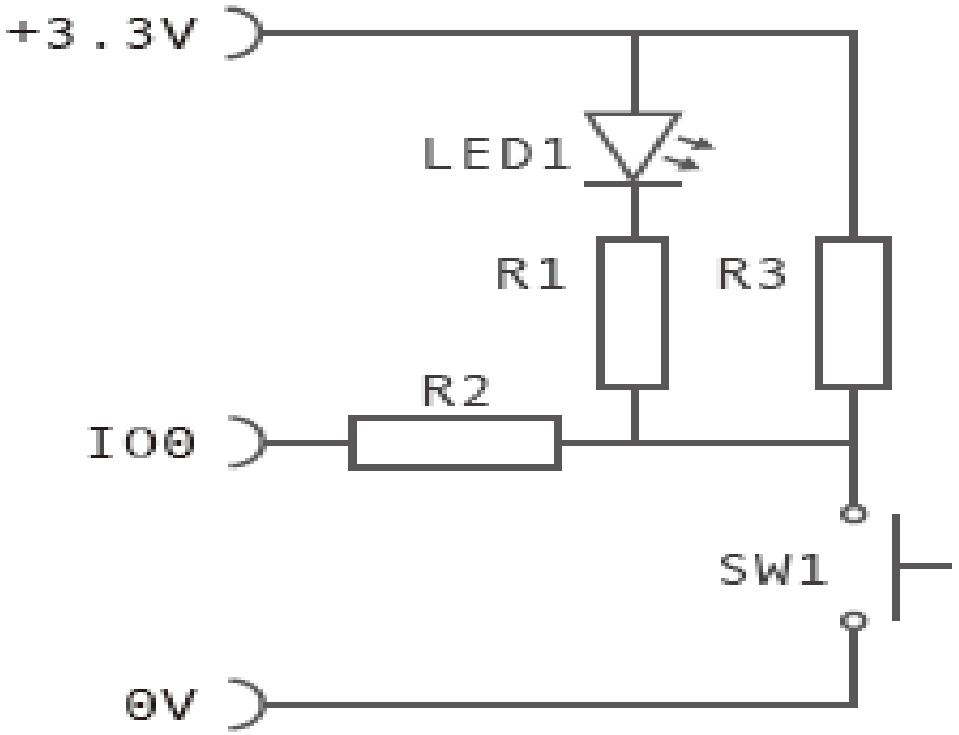
二、实验准备

- 注意，除LED外，其他零件没有正负极：
  - 开关两脚必需跨在电源槽和空插槽之间
  - LED正负极不要接错，一般而言，正极引脚比较长，插在C4孔，负极引脚端，插在C5。



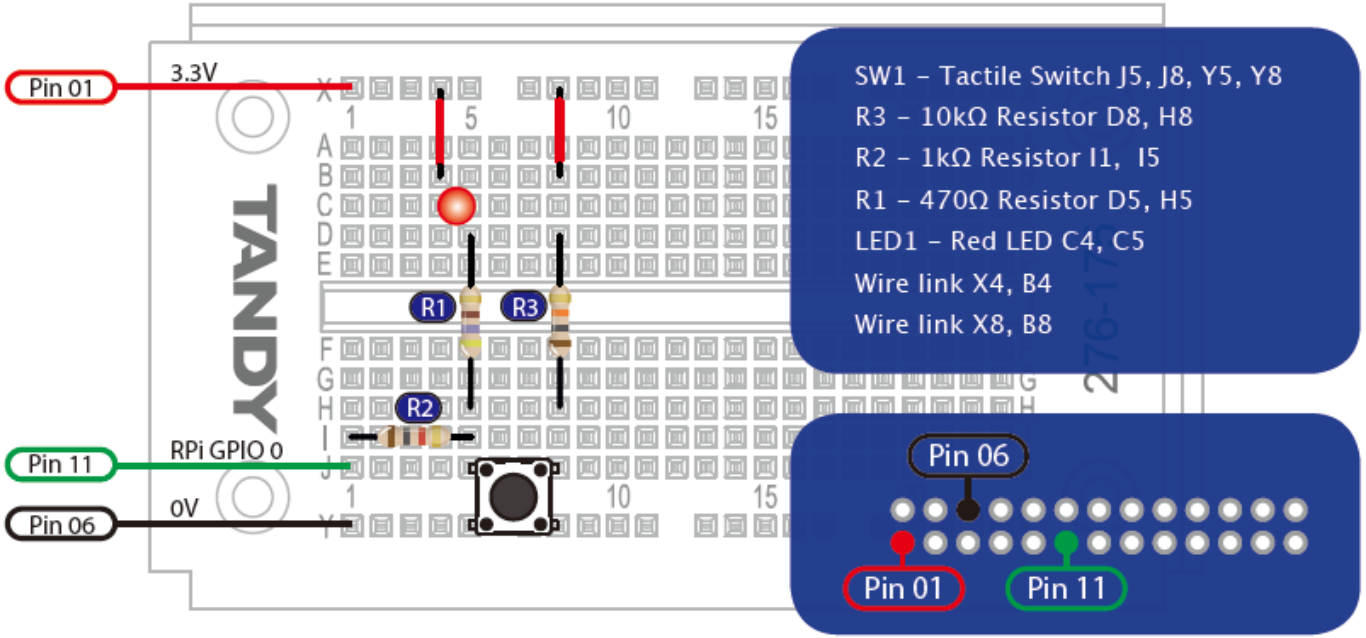
三、实验的电路图

- 实验的电路图如下：



四、完成接线

- 三条杜邦线接Raspberry Pi的pin1、pin6和pin11（如下图，把Raspberry Pi接口管脚摆在左上方。



- 如果没有开关，可以用导线把开关的位置接上，即：连接Y5与J5、Y8与J8。
- 用什么当“开关”？

五、启动树莓派

- 连接好电路后，打开Raspberry Pi，按下开关，LED应该亮起,否则请检查电路。
- 如果LED在开关放开的情况下仍然发光，可以把开关取下旋转90度插回。
- 因为Raspberry Pi的接口不带buffer保护，所以我们接入R2作为保护。
- 而10千欧的R3作为上拉电阻，确保pin11始终处在高电位，而当我们按下开关，pin11就被拉到低电位。

六、安装python GPIO包

- 本例使用的是python语言。我们用的是Raspberry Pi官网上的debian系统的话，python已经包含在里面了。
- **为了保险起见，还是先安装python-dev，输入以下指令。**

  sudo apt-get install python-dev
- 再安装python程序包来赋予Raspberry Pi控制GPIO的能力。从http://pypi.python.org/pypi/RPi.GPIO下载最新的版本
- 使用终端来安装程序包。打开PuTTY，进入系统。
- 执行如下命令：

gunzip RPi.GPIO-0.2.0.tar.gz（版本号可能不同）

tar -xvf RPi.GPIO-0.2.0.tar

cd RPi.GPIO-0.2.0

sudo python setup.py install

- 至此，安装过程结束。

七、编写编程

- 现在开始写一个程序，来监视开关状态并在屏幕上显示些东西。

```
cd ..
nano mybutton.py
```

- 输入（注意空格，否则报错。中文可能成为乱码！）

```
#！/usr/bin/python
import time  //引入time和RPi.GPIO以使用其中的函数
import RPi.GPIO as GPIO//设置模式
GPIO.setmode(GPIO.BOARD)
GPIO.setup(11, GPIO.IN) //把pin11配置为input模式以接收开关状态
while True: // while True是一个死循环。
    mybutton = GPIO.input(11) //读开关状态是否按下
  If mybutton == False: // if语句判断pin11取回的状态，当它变成低电位，即我们按下开关时，就在屏幕上输出一个OK
      print “OK”
      time.sleep(.2) //让程序休息0.2秒。
      GPIO.cleanup() //退出时清理通道
```

八、运行程序

- 输入

sudo python mybutton.py

- 来启动程序，按下开关，应该可以看见屏幕上出现的OK。按ctrl+c（死循环）可以终止程序运行。

九、让LED闪烁起来

- 把程序改一下：

```
GPIO.setup(11,GPIO.OUT)
while True：
 GPIO.output(11, GPIO.HIGH)
 time.sleep(1)
 GPIO.output(11, GPIO.LOW)
 time.sleep(1)
```

- 常亮？——把地线拔掉——好了，闪了！
- 让它在后台运行吧

```
python mybutton.py&
```

- 系统返回一个进程号 ****，程序开始在后台运行
- 掉此进程：Kill **** （进程号）

十、添加计数器的功能

- 把死循环改成计数器，并每按一次开关，就报数一次。
- 请同学们自己完成。开关可以用拔插电源的方式代替。

十一、 用wiringPi GPIO实现对LED的控制

- 用wiringPi GPIO的好处是实现更简单

十一、wiringPi GPIO说明

- WiringPi是应用于树莓派平台的GPIO控制库函数，WiringPi遵守GUN Lv3。
- wiringPi使用C或者C++开发，并且可以被其他语言包使用，例如python、ruby或者PHP等。
- wiringPi包括一套gpio控制命令，使用gpio命令可以控制树莓派GPIO管脚，用户可以利用gpio命令通过shell脚本控制或查询GPIO管脚。
- wiringPi是可以扩展的，可以利用wiringPi的内部模块扩展模拟量输入芯片，可以使用MCP23x17/MCP23x08（I2C 或者SPI）扩展GPIO接口。

十一、wiringPi GPIO说明

- 另外，用户可以自己编写扩展模块并把自定义的扩展模块集成到wiringPi中。
- WiringPi支持模拟量的读取和设置功能，不过在树莓派上并没有模拟量设备。但是使用WiringPi中的软件模块却可以轻松地应用AD或DA芯片。

十二、wiringPi GPIO安装

1)方案A——使用GIT工具

通过GIT获得wiringPi的源代码

git clone git://git.drogon.net/wiringPi

cd wiringPi

./build

build脚本会帮助你编译和安装wiringPi

2)方案B——直接下载

可以在https://git.drogon.net/?p=wiringPi;a=summary网站上直接下载最新版本编译使用

tar xfz wiringPi-xx.tar.gz

cd wiringPi-xx

./build

十三、wiringPi GPIO测试

- wiringPi包括一套gpio命令，使用gpio命令可以控制树莓派上的各种接口，通过以下指令可以测试wiringPi是否安装成功。

$gpio -v

$gpio readall

即可出现右面的io图

```
root@raspberrypi:~/wiringPi# gpio -v
gpio version: 2.21
Copyright (c) 2012-2014 Gordon Henderson
This is free software with ABSOLUTELY NO WARRANTY.
For details type: gpio -warranty

Raspberry Pi Details:
  Type: Model B+, Revision: 1.2, Memory: 512MB, Maker: Sony
root@raspberrypi:~/wiringPi# gpio readall
 +-----+-----+---------+------+---+--B Plus--+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 |     |     |    3.3v |      |   |  1 || 2  |   |      | 5v      |     |     |
 |   2 |   8 |   SDA.1 |   IN | 1 |  3 || 4  |   |      | 5V      |     |     |
 |   3 |   9 |   SCL.1 |   IN | 1 |  5 || 6  |   |      | 0v      |     |     |
 |   4 |   7 | GPIO. 7 |   IN | 0 |  7 || 8  | 1 | ALT0 | TxD     | 15  | 14  |
 |     |     |      0v |      |   |  9 || 10 | 1 | ALT0 | RxD     | 16  | 15  |
 |  17 |   0 | GPIO. 0 |   IN | 1 | 11 || 12 | 0 | IN   | GPIO. 1 | 1   | 18  |
```

十四、wiringPi GPIO程序例子

程序：

```
#include <wiringPi.h>
int main(void)
{
  wiringPiSetup() ;
  pinMode (0, OUTPUT) ;
  for(;;)
  {
    digitalWrite(0, HIGH) ; delay (500) ;
    digitalWrite(0,  LOW) ; delay (500) ;
  }
}
```

十五、wiringPi GPIO程序运行

5、编译运行：

在树莓派上:

gcc -Wall -o test test.c -lwiringPi

sudo ./test

在虚拟机中：

am-linux-gcc -Wall -o test test.c -lwiringPi

sudo ./test

6、注意事项：

1）IO的编号方式略有不同，采用wiring编码方式。

2）-lwiringPi表示动态加载wiringPi共享库。

十六、使用WiringPi的实现点亮LED

□有多种实现点亮LED的方法（不同的包），下面再体验一下wiringPi的实现方法，最后你会发现——大同小异！

①首先确保你已经安装了make，如果没装，很简单：

```
pacman -S make
```

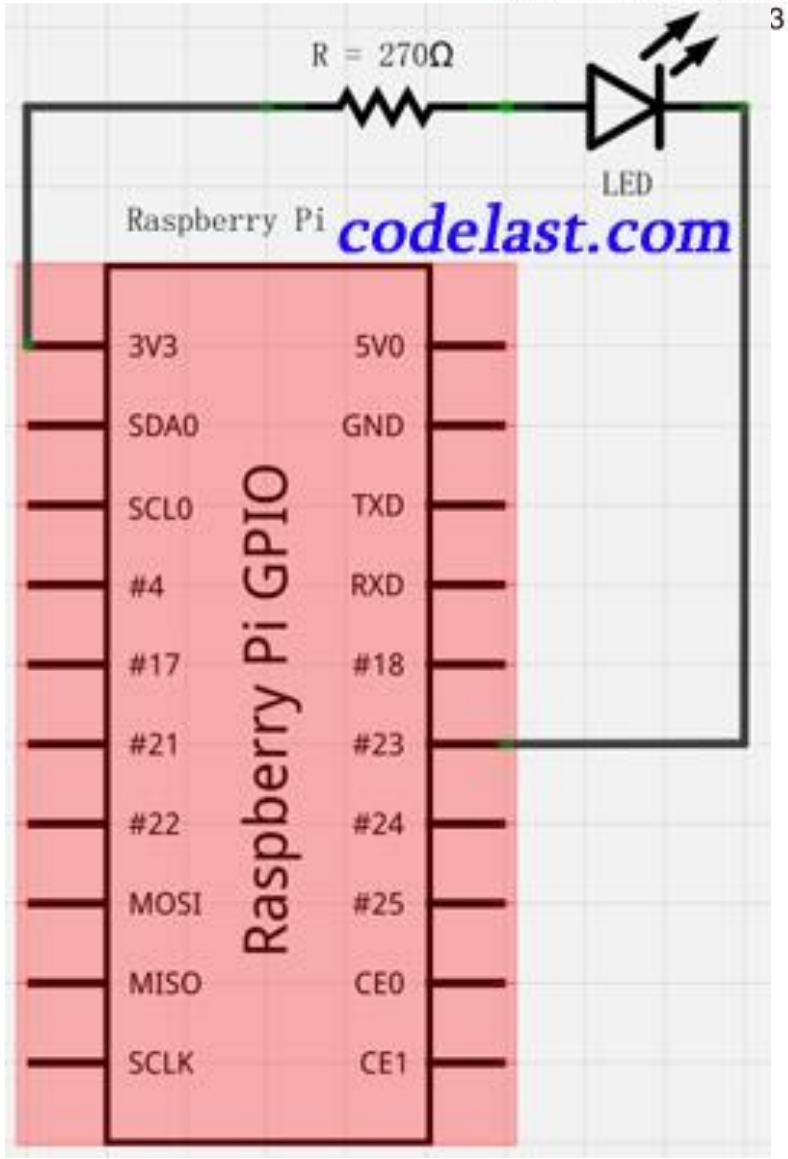
②下载安装包，解压，编译，安装：

```
mkdir temp
cd temp
wget http://project-downloads.drogon.net/files/wiringPi.tgz
tar xf wiringPi.tgz
cd wiringPi/wiringPi/
make
make install
```

这样wiringPi就算安装完了。

十七、电路图

- 在本例中，随意选了一个GPIO口——GPIO 23（即端口16）来接LED。
- 电路图如下：



十八、控制程序1


```
// led.c
#include <wiringPi.h>
#include <stdio.h>
#include <stdlib.h>
int main (int argc,char* argv[])
{
  if (argc < 2) {
    printf("Usage example:  ./led 4 \n");
    return 1;
  }
  int pinNumber = atoi(argv[1]);
  if (-1 == wiringPiSetup()) {
    printf("Setup wiringPi failed!");
    return 1;
  }
```

- 需要注意的是，一旦你用digitalWrite()函数置了高电平，那么只要你不把它置为低电平，它将一直维持在高电平。

十八、控制程序2


```
  pinMode(pinNumber, OUTPUT); // 设置为output模式，设置方法不同
  while(1) {
    digitalWrite(pinNumber, 1); // 置为高电平
    delay(800); //暂停
    digitalWrite(pinNumber, 0); // 置为低电平
    delay(800);
  }

  return 0;
}
```

- 编译程序：

  gcc led.c -o led –lwiringPi

- 运行程序：

  ./led 4

十九、显示结果

- 现在可以看到LED开始闪烁了
- 这里向程序传入了一个参数4，它代表你要置高电平的是GPIO的哪个脚。
- 为什么是4呢？因为以前已经说了，树莓派的GPIO口有两种命名方式，一种是树莓派的方式，另一种是Broadcom的方式，当使用WiringPi时，应参考前者。
- 因此，对应到Broadcom方式的GPIO 23上，那就应该是GPIO 4，所以应该向程序输入参数4。

二十、作业

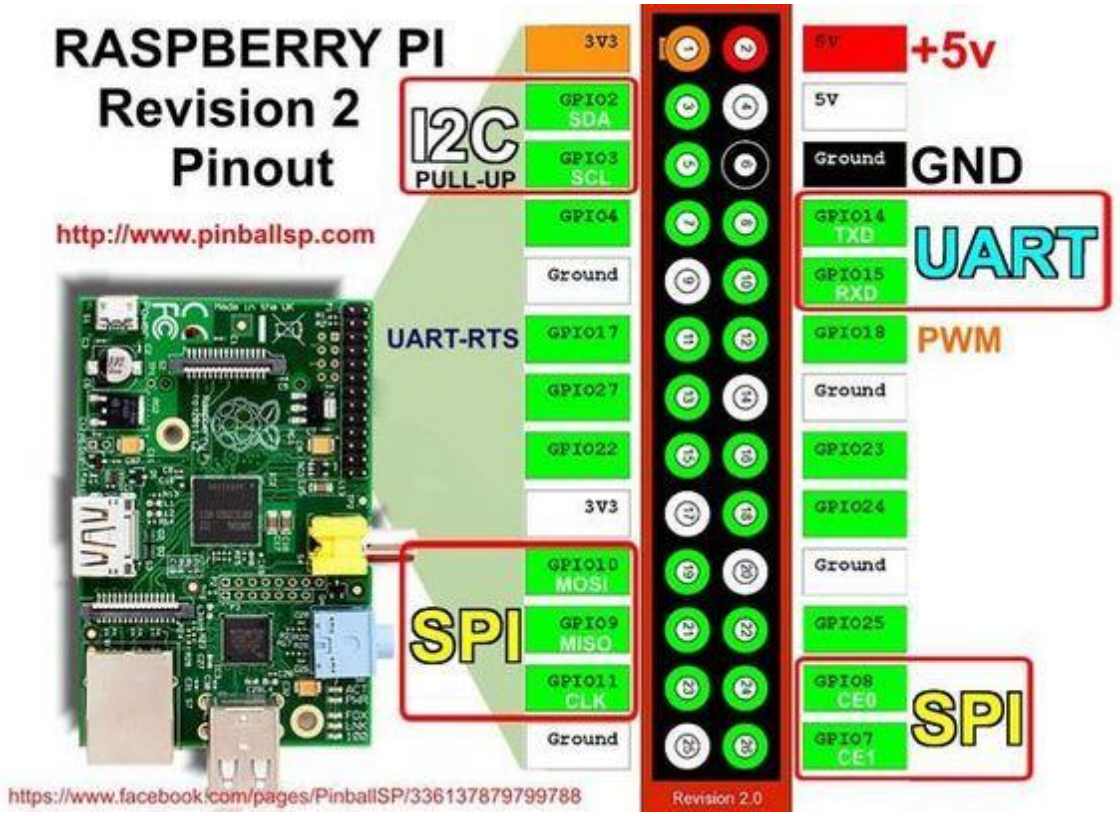
请同学们自己实现：点亮多个LED，让它们轮流闪烁！

提示：

（1）参考十七的电路图，将控制一个LED“复制”为控制多个LED

（2）需要改变什么？为什么？

（3）如果用WiringPi GPIO直接控制LED的话，至少可以接多少LED？



十八、BCM2835 C Library

1、下载:

$ wget http://www.airspayce.com/mikem/bcm2835/bcm2835-1.35.tar.gz

2、解压缩: $tar xvzf bcm2835-1.35.tar.gz

3、进入压缩之后的目录:$cd bcm2835-1.35

4、 配置: $./configure

5、从源代码生成安装包:$make

6、执行检查: $sudo make check

7、安装 **bcm2835**库: $sudo make install

十八、BCM2835 C Library

8、例子

```
#include <bcm2835.h>
// P1插座第11脚
#define PIN RPI_GPIO_P1_11
int main(int argc, char **argv)
{
  if (!bcm2835_init())
  return 1;
  // 输出方式
  bcm2835_gpio_fsel(PIN, BCM2835_GPIO_FSEL_OUTP);
  while (1)
  {
    bcm2835_gpio_write(PIN, HIGH);
    bcm2835_delay(100);
    bcm2835_gpio_write(PIN, LOW);
    bcm2835_delay(100);
  }
  bcm2835_close();
  return 0;
}
```

十八、BCM2835 C Library

9、注意事项：

1）IO的编号方式略有不同，采用wiring编码方式。

2）-lwiringPi表示动态加载wiringPi共享库。

实验结束

总结一下三种方法的异同