一. wiringOP 简介

wiringOP 是基于 wiringPi v2.46 移植修改的，可以在 OrangePi 开发板中使用。它是一个 C 语言写成的类库，除了 GPIO 库，还包含了 I2C 库、SPI 库 UART 库和 PWM 库等，并且还提供了丰富的测试示例。wiringOP 库还包含了一个命令行工具——gpio 命令，通过这个命令可以很直观的看到所有 gpio 引脚当前的状态，还可用来读写设置 gpio 引脚，以及一些其他的功能。

二. wiringOP 下载安装

2.1 安装 git 工具

apt-get update

apt-get install git

2.2 使用 git 下载 wiringOP

git clone https://github.com/orangepi-xunlong/wiringOP.git

2.3 使用 ./build 命令编译安装

cd wiringOP

chmod 777 build

./build

build 编译脚本将会编译和安装 wiringOP 到系统中，您不需要再进行其他设置

2.4 测试安装是否成功

打开命令终端，可以通过 gpio -v 命令来检查 wiringOP 是否安装成功，如果出现下图中类似的输出结果，说明 wiringOP 已经安装成功。

```
root@OrangePi:~/wiringPi# gpio -v
gpio version: 2.46
Copyright (c) 2012-2018 Gordon Henderson
This is free software with ABSOLUTELY NO WARRANTY.
For details type: gpio -warranty
```

三. wiringOP 使用说明

3.1 gpio 命令

通过 gpio 命令我们可以查看所有 gpio 引脚当前详细的状态，并进行读、写和修改模式等操作，下面详解讲解每个命令选项的使用方法

A. 如果只输入不带参数 **gpio** 命令，会提示一些使用帮助信息

```
root@OrangePi:~# gpio
gpio: At your service!
  Type: gpio -h for full details and
        gpio readall for a quick printout of your connector details
root@OrangePi:~#
```

B. **gpio -h** 命令会列出 gpio 命令所有可以使用的选项

```
root@OrangePi:~# gpio -h
gpio: Usage: gpio -v
       gpio -h
       gpio [-g|-1] ...
       gpio <mode/qmode/read/write/pwm> ...
       gpio <toggle/blink> <pin>
       gpio readall
       gpio unexportall/exports
       gpio export/edge/unexport ...
       gpio pwm-bal/pwm-ms
       gpio pwmr <range>
       gpio pwmc <divider>
       gpio i2cd/i2cdetect port
       gpio serial port
       gpio rbx/rbd
       gpio wb <value>

root@OrangePi:~#
```

C. 我们可以通过 **gpio readall** 命令查看所有的 GPIO 引脚当前的详细信息

- **GPIO** 一栏表示 GPIO 口对应的引脚编号，为空表示此引脚为电源或 GND
  具体的计算方法为：(字母在字母表中的位置 - 1) * 32 + 引脚序号
  如 PC04 引脚编号为：(3 - 1) * 32 + 4 = 68
- **wPi** 一栏为 wiringOP 对所有的 GPIO 口重新进行的编号，从 0 开始依次递增，并且不包括电源和 GND 这些物理引脚，所以总数会比物理引脚少
- **Name** 一栏表示每个物理引脚的名字
- **Mode** 一栏可以直观的看到每个引脚的当前功能配置
- **V** 一栏即 value，表示引脚当前为高电平还是低电平
- **Physical** 一栏即物理引脚，也就是我们在开发板上能看到的排针对应的编号

```
root@OrangePi:~/wiringPi# gpio readall
 +------+-----+----------+------+---+OrangePiH3+---+------+----------+-----+------+
 | GPIO | wPi |   Name   | Mode | V | Physical | V | Mode | Name     | wPi | GPIO |
 +------+-----+----------+------+---+----++----+---+------+----------+-----+------+
 |      |     |     3.3v |      |   |  1 || 2  |   |      | 5v       |     |      |
 |   12 |   0 |   SDA.0  | ALT2 | 0 |  3 || 4  |   |      | 5V       |     |      |
 |   11 |   1 |   SCL.0  | ALT2 | 0 |  5 || 6  |   |      | 0v       |     |      |
 |    6 |   2 |      PA6 |  OFF | 0 |  7 || 8  | 0 | ALT3 | TxD3     | 3   | 13   |
 |      |     |       0v |      |   |  9 || 10 | 0 | ALT3 | RxD3     | 4   | 14   |
 |    1 |   5 |     RxD2 | ALT2 | 0 | 11 || 12 | 0 | OFF  | PD14     | 6   | 110  |
 |    0 |   7 |     TxD2 | ALT2 | 0 | 13 || 14 |   |      | 0v       |     |      |
 |    3 |   8 |     CTS2 |  OFF | 0 | 15 || 16 | 0 | OFF  | PC04     | 9   | 68   |
 |      |     |     3.3v |      |   | 17 || 18 | 0 | OFF  | PC07     | 10  | 71   |
 |   64 |  11 |     MOSI | ALT3 | 0 | 19 || 20 |   |      | 0v       |     |      |
 |   65 |  12 |     MISO | ALT3 | 0 | 21 || 22 | 0 | OFF  | RTS2     | 13  | 2    |
 |   66 |  14 |     SCLK | ALT3 | 0 | 23 || 24 | 0 | ALT3 | CE0      | 15  | 67   |
 |      |     |       0v |      |   | 25 || 26 | 0 | OFF  | PA21     | 16  | 21   |
 |   19 |  17 |   SDA.1  | ALT3 | 0 | 27 || 28 | 0 | ALT3 | SCL.1    | 18  | 18   |
 |    7 |  19 |     PA07 |  OFF | 0 | 29 || 30 |   |      | 0v       |     |      |
 |    8 |  20 |     PA08 |  OFF | 0 | 31 || 32 | 0 | OFF  | RTS1     | 21  | 200  |
 |    9 |  22 |     PA09 |  OFF | 0 | 33 || 34 |   |      | 0v       |     |      |
 |   10 |  23 |     PA10 |  OFF | 0 | 35 || 36 | 0 | OFF  | CTS1     | 24  | 201  |
 |   20 |  25 |     PA20 |  OFF | 0 | 37 || 38 | 0 | ALT2 | TxD1     | 26  | 198  |
 |      |     |       0v |      |   | 39 || 40 | 0 | ALT2 | RxD1     | 27  | 199  |
 |    4 |  28 |     PA04 | ALT2 | 0 | 41 || 42 | 0 | ALT2 | PA05     | 29  | 5    |
 +------+-----+----------+------+---+----++----+---+------+----------+-----+------+
 | GPIO | wPi |   Name   | Mode | V | Physical | V | Mode | Name     | wPi | GPIO |
 +------+-----+----------+------+---+OrangePiH3+---+------+----------+-----+------+
```

D. **gpio mode** 命令可以配置某个 GPIO 口为输入或者输出模式

- **命令格式：gpio mode <pin> <mode>**

  pin: 默认情况下为 wPi 的引脚编号

  mode: 可以为 in 或者 out

```
root@OrangePi:~# gpio mode 0 in
root@OrangePi:~# gpio readall
 +------+-----+----------+------+---+OrangePiH3+---+------+----------+-----+------+
 | GPIO | wPi |   Name   | Mode | V | Physical | V | Mode | Name     | wPi | GPIO |
 +------+-----+----------+------+---+----++----+---+------+----------+-----+------+
 |      |     |     3.3v |      |   |  1 || 2  |   |      | 5v       |     |      |
 |   12 |   0 |   SDA.0  |   IN | 1 |  3 || 4  |   |      | 5V       |     |      |
 |   11 |   1 |   SCL.0  |  OUT | 0 |  5 || 6  |   |      | 0v       |     |      |
```

E. **gpio qmode** 命令可以查询某个引脚的当前模式

- **命令格式：gpio qmode <pin>**

  pin: 默认情况下为 wPi 的引脚编号

```
root@OrangePi:~# gpio qmode 0
IN
root@OrangePi:~# gpio qmode 1
OUT
root@OrangePi:~#
```

F. **gpio read** 命令可以查看某个引脚当前的电平状态

- **命令格式：gpio read <pin>**

  pin: 默认情况下为 wPi 的引脚编号

```
root@OrangePi:~# gpio read 0
1
root@OrangePi:~# gpio read 1
0
root@OrangePi:~#
```

G. **gpio write** 命令可以设置某个引脚当前的电平状态

- **命令格式：gpio read <pin> <value>**

  pin: 默认情况下为 wPi 的引脚编号

  value: 为 0 或者 1

```
root@OrangePi:~# gpio mode 0 out
root@OrangePi:~# gpio write 0 1
root@OrangePi:~# gpio readall
 +------+-----+----------+------+---+OrangePiH3+-
 | GPIO | wPi |   Name   | Mode | V | Physical |
 +------+-----+----------+------+---+----++----+-
 |      |     |     3.3v |      |   |  1 || 2  |
 |   12 |   0 |   SDA.0  |  OUT | 1 |  3 || 4  |
 |   11 |   1 |   SCL.0  |  OUT | 0 |  5 || 6  |
```

H. **gpio pwm** 可以设置某个具有 PWM 复用功能的 GPIO 口为 pwm 模式。

此命令只能设置那些硬件上具有 pwm 复用功能的引脚，否则会提示错误。H3 芯片具有 pwm 复用功能的 GPIO 引脚为 PA05，对应的 wpi 编号为 **42**。

- **命令格式：gpio pwm <pin> <value>**

  pin: 为具有 pwm 功能引脚对应的 wPi 引脚编号，H3 为 42

  value: 为脉冲宽度的大小，范围必须在 0 ~ 1024 之间

```
root@OrangePi:~/wiringPi# gpio pwm 20 100
the pin you choose doesn't support hardware PWM
you can select wiringPi pin 42 for PWM pin
or you can use it in softPwm mode
please use soft pwmmode or choose PWM pin
root@OrangePi:~/wiringPi# gpio pwm 29 2000
val pwmWrite 0 <= X <= 1024
Or you can set new range by yourself by pwmSetRange(range
root@OrangePi:~/wiringPi#
```

这两个是错误示例，第一个是 pin 值不对，第二个是 value 的值不对，分别会提示不同的错误信息。正确示例如下图所示：

```
root@OrangePi:~# gpio pwm 29 500
root@OrangePi:~#
```

另外需要注意的是 H3 的 PA05 正好是系统的调试串口的 RX 引脚，所以当使能了 pwm 模式会导致系统的调试串口能不用，此时可以通过 ssh 远程登录来控制系统。

I. **gpio toggle** 命令会反转当前引脚的电平值，如果当前为高电平，运行这条命令后引脚的值就会变成低电平

- **命令格式：gpio toggle <pin>**

  pin: 默认情况下为 wPi 的引脚编号

J. **gpio blink** 命令会持续的反转当前引脚的电平值，如果此引脚接了一个 LED 灯，可以看到 LED 会不停的闪烁

- **命令格式：gpio blink <pin>**

  pin: 默认情况下为 wPi 的引脚编号

K. **gpio export** 命令用来导出某个引脚的 /sys/class/gpio 接口，这样就可以通过 /sys/class/gpio 接口来控制 GPIO 引脚，我们也可以在程序或者 shell 脚本中调用它。

- **命令格式：gpio export <pin> <mode>**

  pin: 默认情况下为 wPi 的引脚编号

  mode: 可以为 in 或者 out

```
root@OrangePi:/sys/class/gpio# gpio export 2 out
root@OrangePi:/sys/class/gpio# ls
export  gpio6  gpiochip0  unexport
root@OrangePi:/sys/class/gpio# cat gpio6/direction
out
root@OrangePi:/sys/class/gpio#
```

L. **gpio unexport** 命令用来关闭已导出的某个引脚的 /sys/class/gpio 接口

- **命令格式：gpio unexport <pin>**

  pin: 默认情况下为 wPi 的引脚编号

```
root@OrangePi:/sys/class/gpio# ls
export  gpio6  gpiochip0  unexport
root@OrangePi:/sys/class/gpio# gpio unexport 2
root@OrangePi:/sys/class/gpio# ls
export  gpiochip0  unexport
root@OrangePi:/sys/class/gpio#
```

M. **gpio unexportall** 命令用来关闭所以已导出引脚的 /sys/class/gpio 接口

- **命令格式：gpio unexportall**

```
root@OrangePi:/sys/class/gpio# ls
export  gpio11  gpio12  gpio13  gpio14  gpio6  gpiochip0  unexport
root@OrangePi:/sys/class/gpio# gpio unexportall
root@OrangePi:/sys/class/gpio# ls
export  gpiochip0  unexport
root@OrangePi:/sys/class/gpio#
```

N. **gpio exports** 命令用来查看所以已导出引脚的 /sys/class/gpio 接口的状态

- **命令格式：gpio exports**

  第一列表示引脚的 wPi 编号

  第二列表示引脚对应的 GPIO 编号

  第三列表示引脚的方向

  第四列表示引脚当前的电平值

  第五列表示引脚当前的触发方式

```
root@OrangePi:/sys/class/gpio# gpio exports
GPIO Pins exported:
   0(12): out  0  none
   1(11): out  0  none
   2(6): out  0  none
   3(13): out  0  none
   4(14): out  0  none
root@OrangePi:/sys/class/gpio#
```

O. **gpio i2cd/i2cdetect port** 用于检测挂载在 i2c 总线上的器件

其中 i2cd 和 i2cdetect 选项的作用是一样的

- **命令格式：gpio i2cd/i2cdetect <port>**

  port: 表示 i2c 总线的序号，如 0 代表检测 i2c-0 总线上的器件

```
root@OrangePi:~# gpio i2cd 0
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: 50 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
root@OrangePi:~#
```

P. **gpio serial port** 用于通过回环测试来确定串口功能是否正常

- **命令格式：gpio serial <port>**

  port: 表示具体串口设备节点，如 /dev/ttyS2

使用这条命令前必须先用杜邦线短接要测试的串口设备的 RX 和 TX 引脚

```
root@OrangePi:~# gpio serial /dev/ttyS2

Out:    0:   ->   0
Out:    1:   ->   1
Out:    2:   ->   2
Out:    3:   ->   3
Out:    4:   ->   4
Out:    5:   ->   5
Out:    6:   ->   6
```

Q. **gpio rbx** 用于读取前 wPi 编号前八位(即 0-7) GPIO 的值，并用十六进制显示结果

- **命令格式：gpio rbx**

```
root@OrangePi:~# gpio rbx
FF
root@OrangePi:~#
```

R. **gpio rbx** 用于读取前 wPi 编号前八位(即 0-7) GPIO 的值，并用十进制显示结果

- **命令格式：gpio rbx**

```
root@OrangePi:~# gpio rbd
255
root@OrangePi:~#
```

S. **gpio wb** 用于一次性写 wPi 编号前八位(即 0-7) GPIO 的值

- **命令格式：gpio wb <value>**

  value: 为八位 GPIO 的值，可以用十进制也可以用十六进制表示，用十六进制需要在数值的前面加 0x，如 0xFF

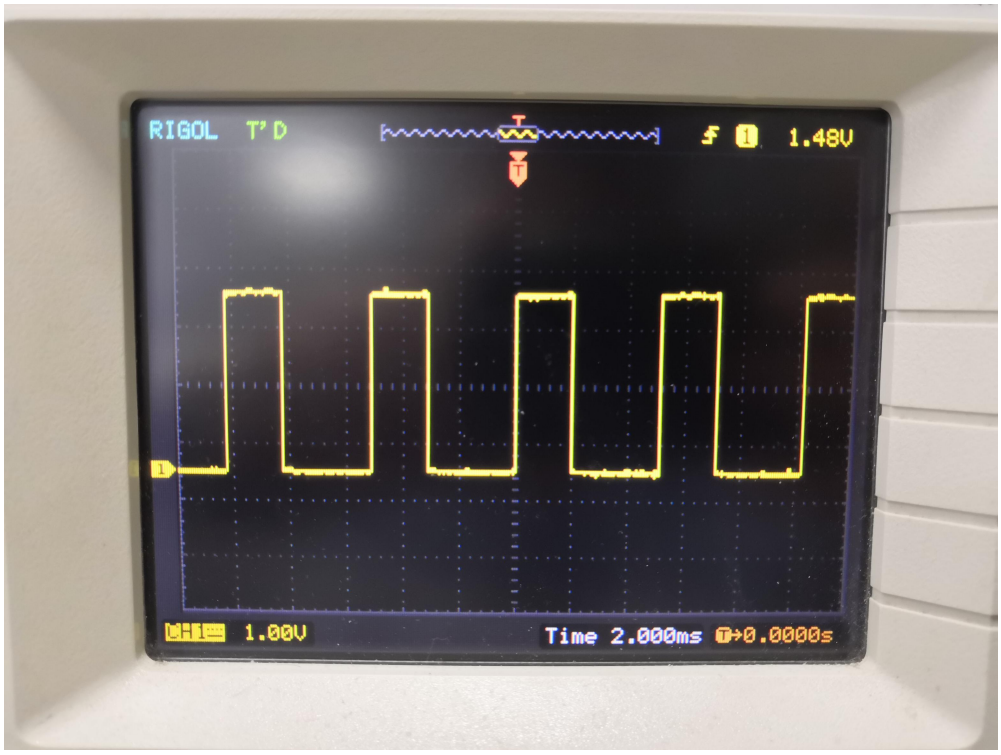
```
root@OrangePi:~# gpio wb 0xff
root@OrangePi:~# gpio wb 255
root@OrangePi:~# gpio readall
 +------+-----+----------+------+---+OrangePiH3+---+------+----------+-----+------+
 | GPIO | wPi |   Name   | Mode | V | Physical | V | Mode | Name     | wPi | GPIO |
 +------+-----+----------+------+---+----++----+---+------+----------+-----+------+
 |      |     |     3.3v |      |   |  1 || 2  |   |      | 5v       |     |      |
 |   12 |   0 |    SDA.0 | ALT2 | 1 |  3 || 4  |   |      | 5V       |     |      |
 |   11 |   1 |    SCL.0 | ALT2 | 1 |  5 || 6  |   |      | 0v       |     |      |
 |    6 |   2 |      PA6 |  OFF | 1 |  7 || 8  | 1 | ALT3 | TxD3     | 3   | 13   |
 |      |     |       0v |      |   |  9 || 10 | 1 | ALT3 | RxD3     | 4   | 14   |
 |    1 |   5 |     RxD2 | ALT2 | 1 | 11 || 12 | 1 | OFF  | PD14     | 6   | 110  |
 |    0 |   7 |     TxD2 | ALT2 | 1 | 13 || 14 |   |      | 0v       |     |      |
SPI
```

T. gpio pwm-bal 用于设置 PWM 模式为脉冲模式

- **命令格式：gpio pwm-bal**

U. gpio pwm-ms 用于设置 PWM 模式为循环模式

- **命令格式：gpio pwm-ms**



3.2 wiringPi 设置函数

有三个函数来对 wiringOP 进行初始化，它们是：

- int wiringPiSetup(void);

  该函数初始化 wiringOP，假定程序将使用 wiringOP 的管脚定义图。具体管脚映射，可以通过 gpio readall 命令来查看。该函数需要 root 权限。

- int wiringPiSetupPhys(void);

  该函数和 wiringPiSetup 函数类似，区别在于允许程序使用物理管脚定义。该函数需要 root 权限。

- int wiringPiSetupSys(void);

  该函数初始化 wiringOP，使用/sys/class/gpio 接口，而不是直接通过操作硬件来实现。该函数可以使用非 root 权限用户，在此种模式下的管脚号是对应的 GPIO 管脚号，

  在此种模式下，在运行程序前，您需要通过/sys/class/gpio 接口导出要使用的管脚。你可以在一个独立的 shell 脚本中来导出将要使用的管脚，或者使用系统的 system() 函数来调用 gpio 命令。

在程序开始时需要调用上面的一个函数来进行初始化，否则，程序可能不能正常工作。

3.3 GPIO 测试示例程序

在 wiringOP/examples 目录下有一些 wiringOP 库的使用示例

```
#include <stdio.h>
#include <wiringPi.h>

int main (void)
{
  int i = 0;

  wiringPiSetup () ;

  for (i=0; i<28; i++)
        pinMode (i, OUTPUT) ;

  for (;;)
  {
    for (i=0; i<28; i++)
        digitalWrite (i, HIGH) ;         // On
    delay (500) ;                        // 500mS

    for (i=0; i<28; i++)
        digitalWrite (i, LOW) ;         // Off
    delay (500) ;
  }
  return 0 ;
}
```

编译测试方法：

```
# cd wiringOP/examples
# make blink
# ./blink
```

这个测试程序的路径为 wiringOP/examples/blink.c，最开始通过 wiringPiSetup() 函数进行初始化，然后调用 pinMode() 设置所有的 GPIO 口为输出模式，然后通过一个 for 死循环不断的反转所有的 GPIO 口引脚的电平，如果此时我们给 GPIO 口接上小灯，可以看到小灯会不停的闪烁。这个程序可以用来测试板子上所有的普通 GPIO 口的功能是否正常。

3.4 串口测试程序

```
#include <stdio.h>
#include <string.h>
#include <errno.h>

#include <wiringPi.h>
#include <wiringSerial.h>

int main ()
{
  int fd ;
  int count ;
  unsigned int nextTime ;

/*
        串口的设备节点名和波特率需要根据具体的情况设置，这里默认是
ttyS2/115200
*/
  if ((fd = serialOpen ("/dev/ttyS2", 115200)) < 0)
  {
    fprintf (stderr, "Unable to open serial device: %s\n", strerror
(errno)) ;
    return 1 ;
  }

  if (wiringPiSetup () == -1)
  {
    fprintf (stdout, "Unable to start wiringPi: %s\n", strerror
(errno)) ;
    return 1 ;
  }

  nextTime = millis () + 300 ;
```

```
  for (count = 0 ; count < 256 ; )
  {
    if (millis () > nextTime)
    {
      printf ("\nOut: %3d: ", count) ;
      fflush (stdout) ;
      serialPutchar (fd, count) ;
      nextTime += 300 ;
      ++count ;
    }

    delay (3) ;

    while (serialDataAvail (fd))
    {
      printf (" -> %3d", serialGetchar (fd)) ;
      fflush (stdout) ;
    }
  }

  printf ("\n") ;
  return 0 ;
}
```

这是通过调用 wiringOP 串口库的串口函数来测试串口的回环功能。运行程序前需要先短接对应串口 RX 和 TX 两个引脚。

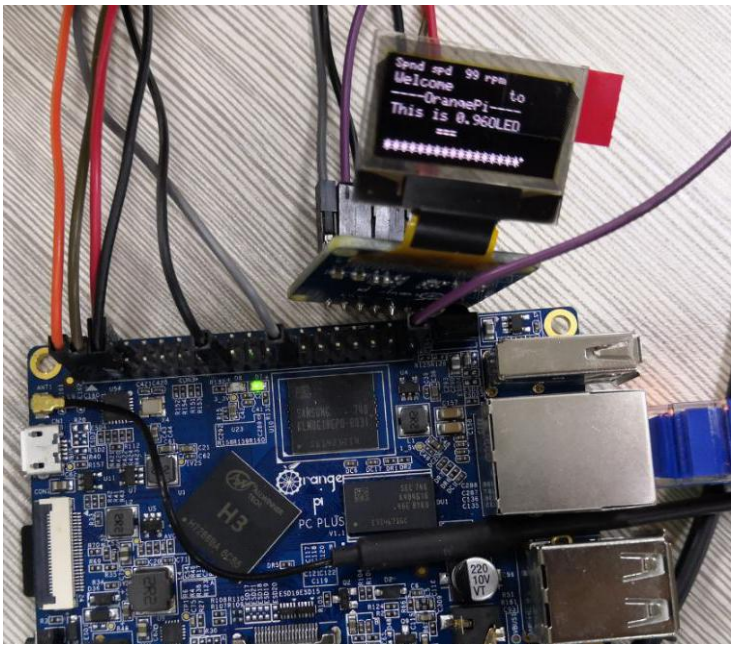
编译测试方法：

```
# cd wiringOP/examples
# make serialTest
# ./serialTest
Out:   0:  ->   0
Out:   1:  ->   1
Out:   2:  ->   2
Out:   3:  ->   3
Out:   4:  ->   4
Out:   5:  ->   5
Out:   6:  ->   6
Out:   7:  ->   7
Out:   8:  ->   8
Out:   9:  ->   9
```

3.5 I2C 测试程序

wiringOP/examples 中移植了一个 oled_demo.c 测试程序，可以使用 OrangePi 的 0.96 寸 OLED 模块测试 I2C 接口的功能

接线图如下图所示：



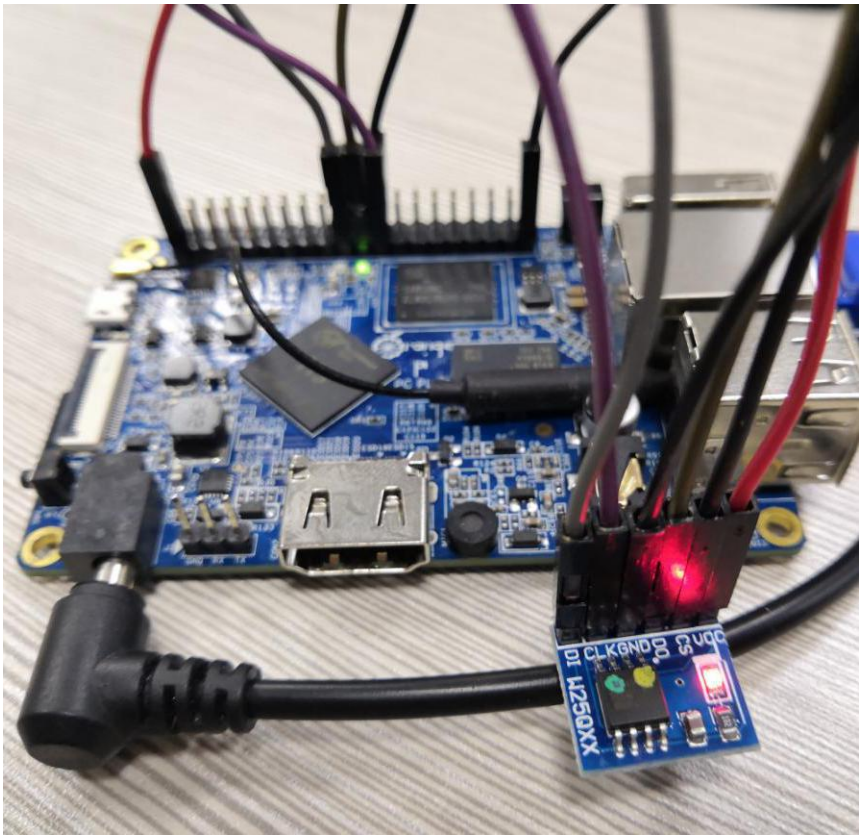
测试方法如下：

```
# cd wiringOP/examples
# make oled_demo
# ./oled_demo  /dev/i2c-0   //设备节点号需要根据具体情况修改
---------start--------
----------end---------
#
```

3.6 SPI 测试程序

wiringOP/examples 中移植了一个 w25q64_test.c 测试程序，可以使用 W25QXX 模块测试 SPI 接口的功能

接线图如下图所示：



测试方法如下：

```
# make w25q64_test
# ./w25q64_test
JEDEC ID : ef 40 18    //能正确读出 ID 说明 SPI 通信正常
Unique ID : d2 66 64 b8 e3 3e 39
Read Data: n=256
------------------------------------------------------------
00000: 30 31 32 33 34 35 36 37 38 39 41 42 43 44 45 46 |a2
00010: 47 48 49 4a 4b 4c 4d 4e 4f 50 51 52 53 54 55 56 |e8
00020: 57 58 59 5a ff ff ff ff ff ff ff ff ff ff ff ff |56
00030: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |f0
00040: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |f0
00050: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |f0
00060: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |f0
00070: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |f0
00080: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |f0
00090: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |f0
000a0: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |f0
000b0: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |f0
000c0: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |f0
```

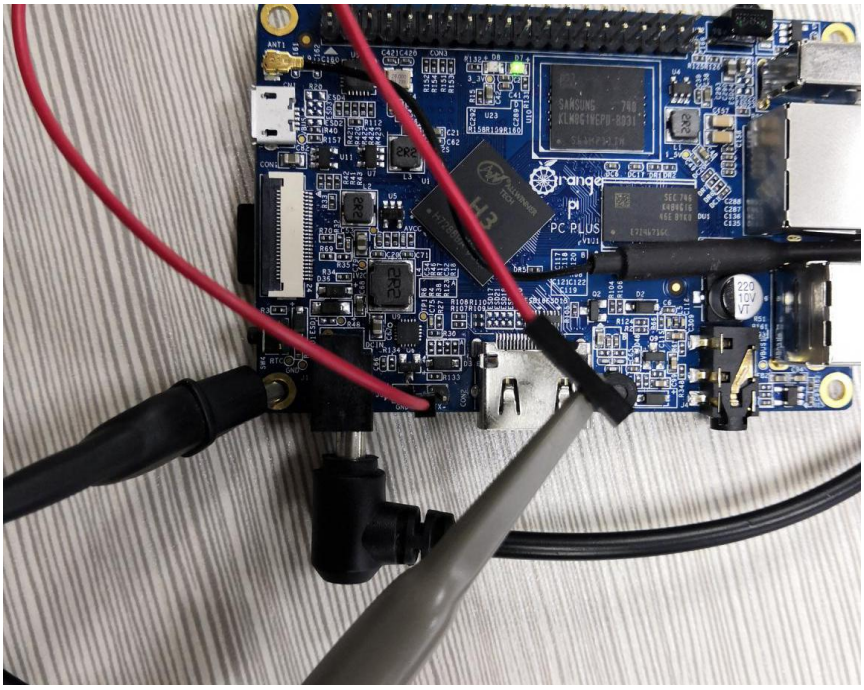
```
000d0: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |f0
000e0: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |f0
000f0: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |f0
---------------------------------------------------------
       c1 c4 c7 ca 71 73 75 77 79 7b 84 86 88 8a 8c 8e |10

Fast Read Data: n=256
---------------------------------------------------------
00000: 30 31 32 33 34 35 36 37 38 39 41 42 43 44 45 46 |a2
00010: 47 48 49 4a 4b 4c 4d 4e 4f 50 51 52 53 54 55 56 |e8
00020: 57 58 59 5a ff ff ff ff ff ff ff ff ff ff ff ff |56
00030: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |f0
00040: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |f0
00050: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |f0
00060: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |f0
00070: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |f0
00080: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |f0
00090: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |f0
000a0: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |f0
000b0: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |f0
000c0: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |f0
000d0: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |f0
000e0: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |f0
000f0: ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff |f0
---------------------------------------------------------
       c1 c4 c7 ca 71 73 75 77 79 7b 84 86 88 8a 8c 8e |10

......
......
......

Status Register-1: 0
Status Register-2: 0
```

3.6 PWM 测试程序

wiringOP/examples 中的 pwm.c 程序可以用来测试 pwm 的功能。

```
#include <wiringPi.h>

#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>

#define PWM_PIN 29 //H3 必须设置为 29

int main (void)
{
  printf ("OrangePi Pi wiringPi PWM test program\n") ;

  if (wiringPiSetup () == -1)
    exit (1) ;

  pinMode (PWM_PIN, PWM_OUTPUT) ; //设置 GPIO 口为 PWM 模式
  pwmWrite (PWM_PIN, 500);        //设置占空比

  return 0 ;
}
```

接线图如下图所示，我们可以通过示波器来查看 PWM 的波形：



测试方法如下：

```
# cd wiringOP/examples
# make pwm
# ./pwm
```

示波器显示波形如下：