OrangePi 4 硬件接口相关说明。

首先安装下 wiringOP，以下操作可能会用到
下载 wiringOP

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

编辑及安装

```
root@OrangePi:~/wiringOP# ./build
All available boards:
   0. OrangePi_PC2
   1. OrangePi_A64
   2. OrangePi_ZERO
   3. OrangePi_H3
   4. OrangePi_LITE2
   5. OrangePi_H3_ZEROPLUS2
   6. OrangePi_3
   7. OrangePi_RK3399
   8. OrangePi_ONEPLUS
   9. OrangePi_4
Choice: 4
```

安装完成

```
All Done.

NOTE: To compile programs with wiringPi, you need to add:
    -lwiringPi
  to your compile line(s) To use the Gertboard, MaxDetect, etc.
  code (the devLib), you need to also add:
    -lwiringPiDev
  to your compile line(s).
```

1、串口的使用

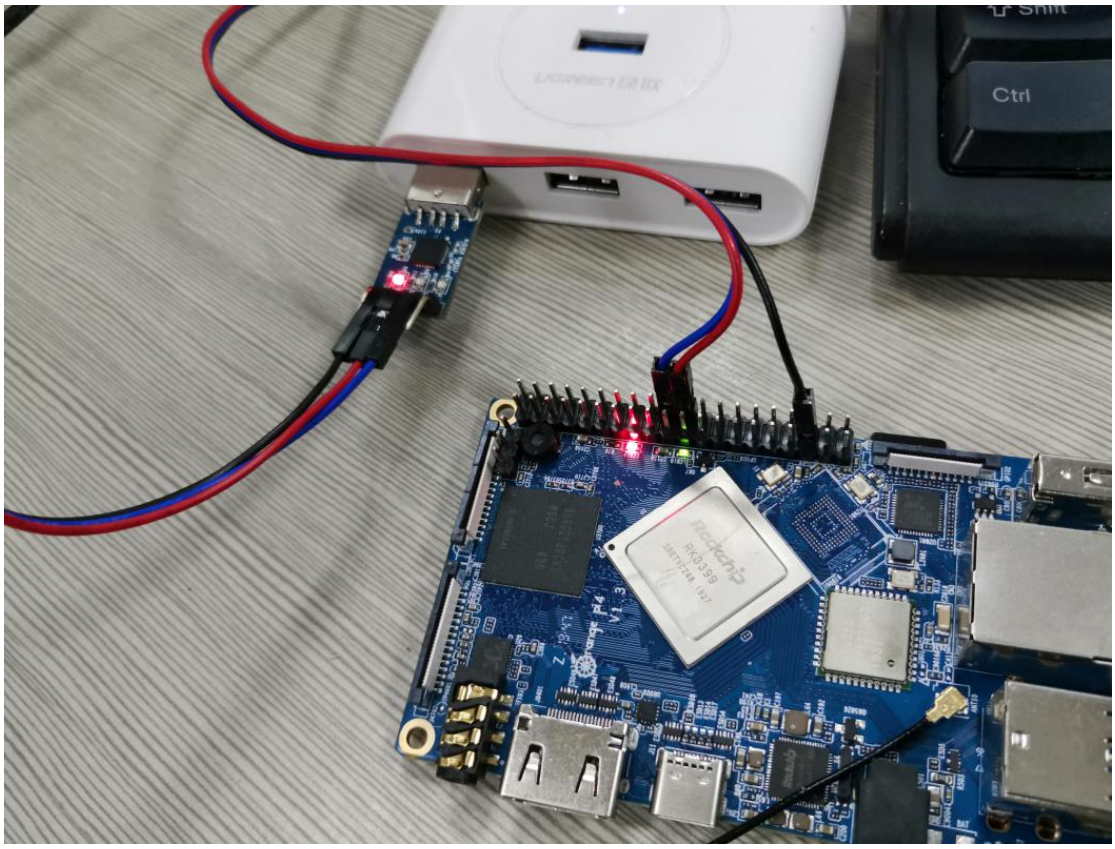
OrangePi 4 只有 uart4 可供应用程序开发使用
在 wiringOP 的 example 目录下有个 serialRead.c 可用于测试串口。
打开 serialRead.c，修改 ttyS2 为 ttyS4。

```
35    if ((fd = serialOpen ("/dev/ttyS4", 115200)) < 0)
```

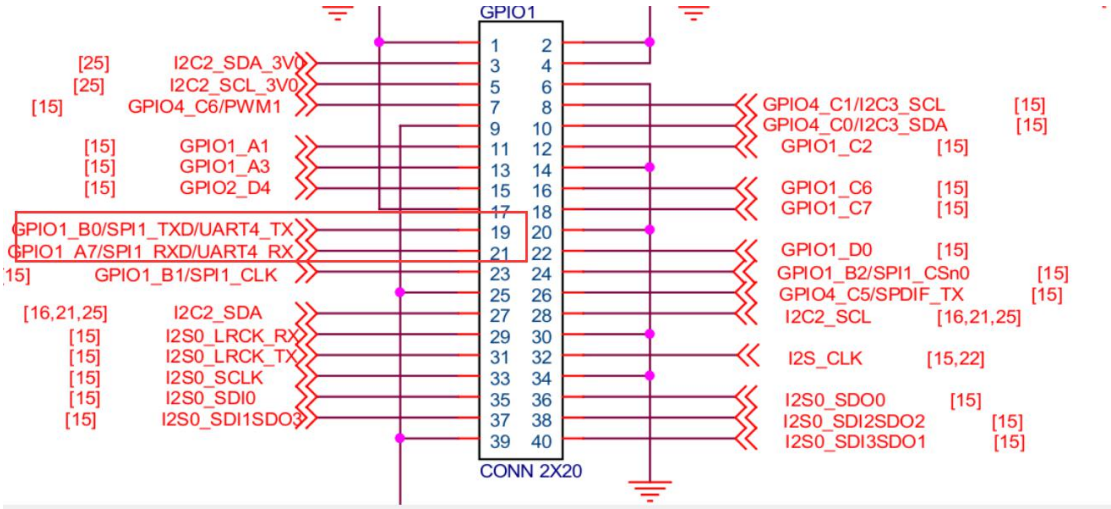
编译 serialRead.c

```
cd example
make serialRead
```

通过 usb 转 ttl 串口连接板子与 PC。



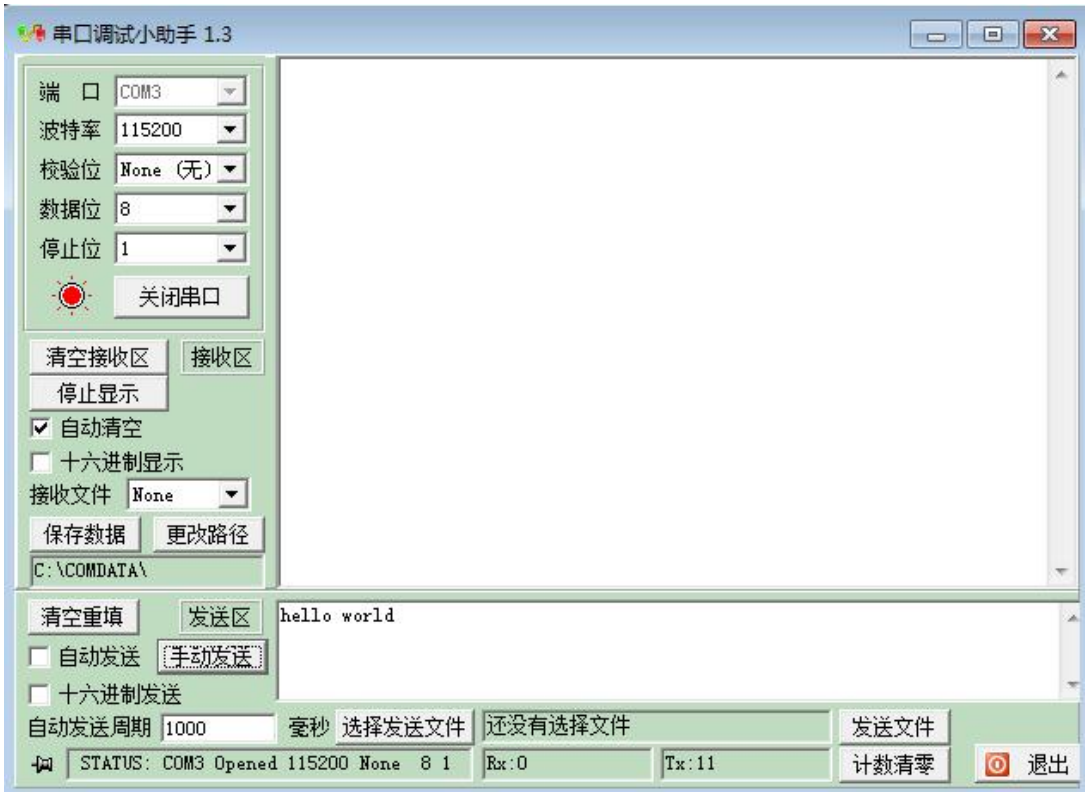
19 脚是 TXD，21 脚是 RXD。



在板子上运行测试程序

```
root@OrangePi:~/wiringOP/examples# ./serialRead
```

在 PC 端使用串口调试助手之类的软件，波特率设为 115200，发送一个字符串“hello world”。

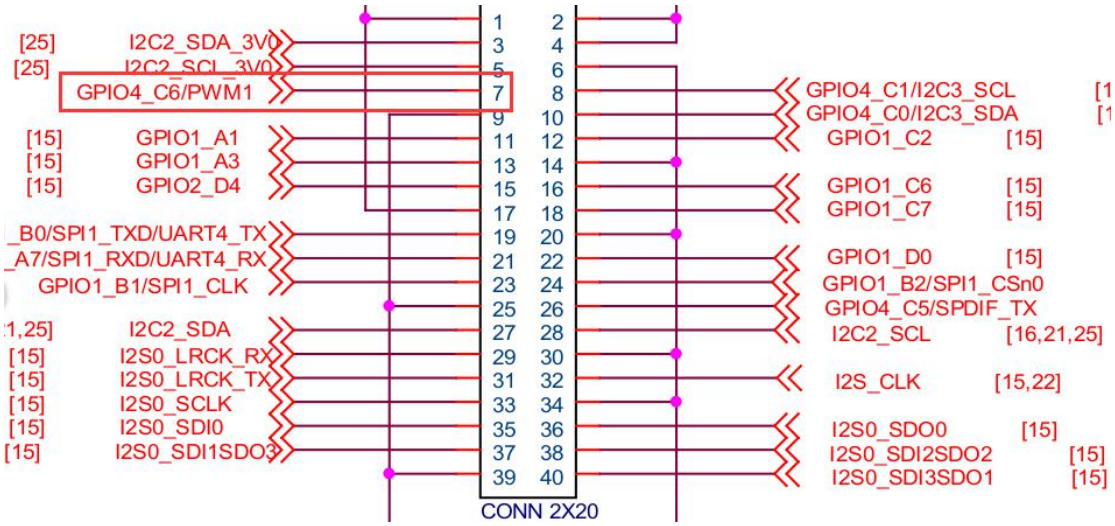


板子收到“hello world”字符串。

```
root@OrangePi:~/wiringOP/examples# ./serialRead
hello world
```

2、PWM 使用说明

Pwm 提供了用户层的接口，40pin 的第 7 引脚为 PWM1



官方释放的镜像 dts 已经打开了 pwm1

```
&pwm1 {
        status = "okay";
};
```

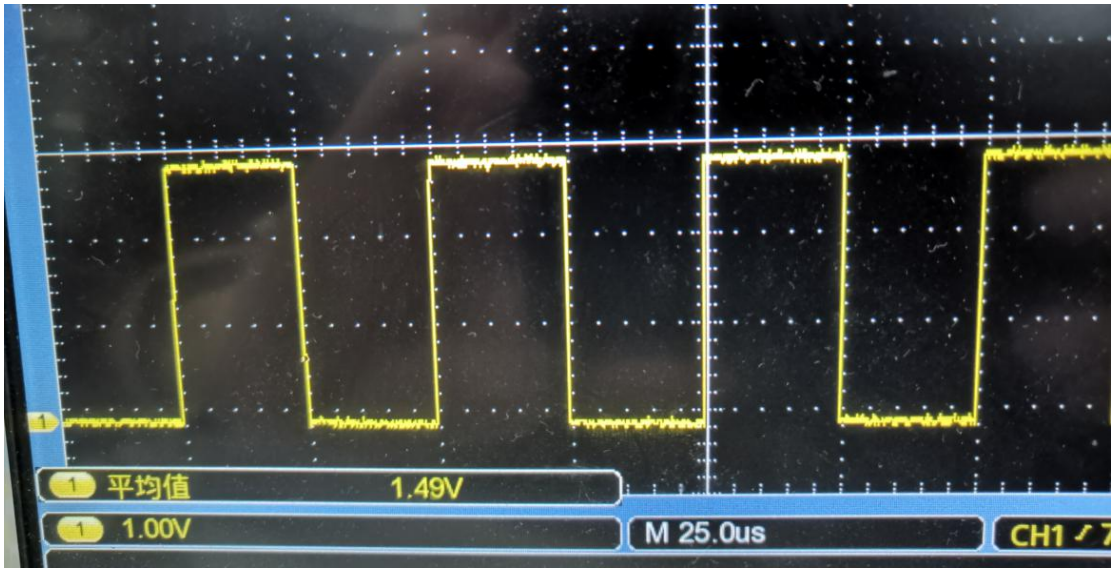
pwm 驱动加载成功后，在/sys/class/pwm/下会产生 pwmchip1 目录，向 export 文件写入 0，就会打开 pwm 定时器，会产生一个 pwm0 目录。相反往 unexport 文件写入 0，就会关闭 pwm 定时器，同时 pwm0 目录会被删除。
该目录有以下几个文件：

```
enable ： 写入 1 使能 pwm，写入 0 关闭 pwm
polarity  ： 有 normal 和 inversed 两个参数选择，表示输出引脚电平翻转。
duty_cycle ： 单位纳秒，在 normal 模式下，表示高电平持续的时间
              在 inversed 模式下，表示低电平持续时间。
period  ： 单位纳秒，表示 pwm 波持续周期
```

举例：让 pwm1 输出占空比为 50%，周期为 50 微秒的方波。

```
cd /sys/class/pwm/pwmchip1
echo 0 > export
echo 50000 > pwm0/period
echo 25000 > pwm0/duty_cycle
echo 1 > pwm0/enable
```

用示波器可看到 pwm1 输出的波形。



3、SPI 的使用说明

SPI 和 UART4 共享相同的引脚。官网镜像不支持 SPI，需要修改内核的 DTS 文件以启用 SPI。

首先按照用户手册方法下载 Linux 源代码。

1、修改 dts 以启用 SPI

```
cd OrangePiRK3399_Pi4/
vi kernel/arch/arm64/boot/dts/rockchip/rk3399-orangepi.dts
```

找到 spi 的定义

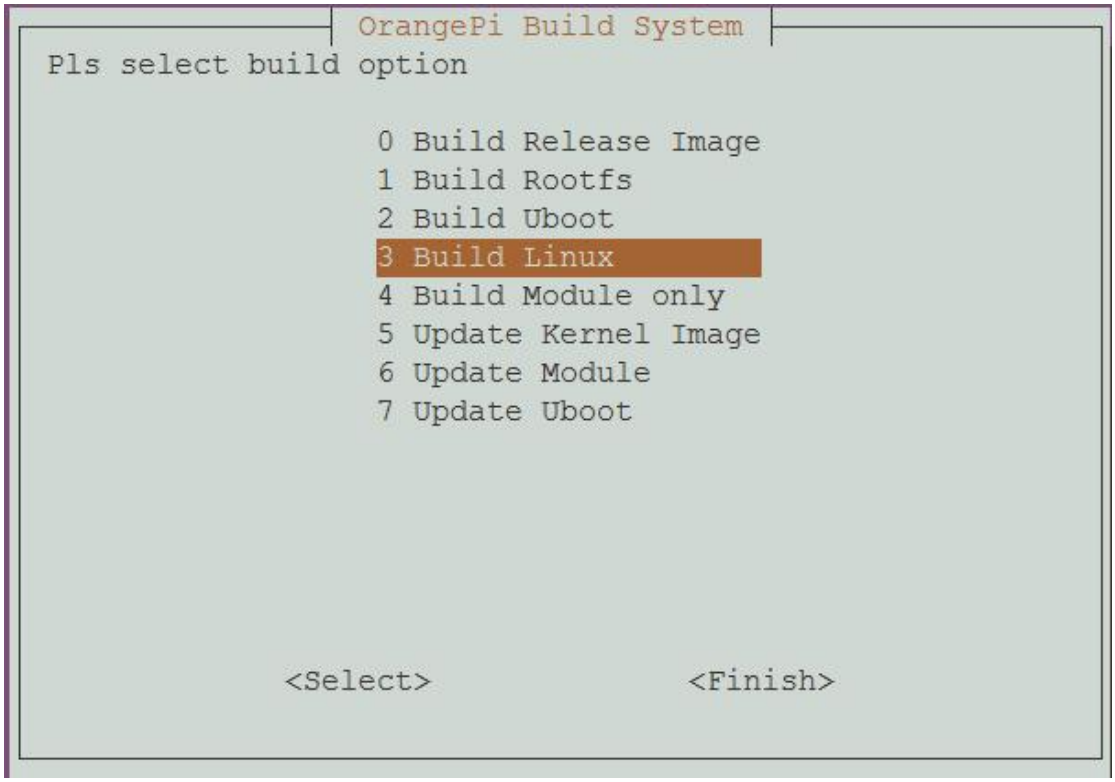
```
&spi1 {
        status = "disable";         //将 disable 改为 okay
```

找到找到 uart4 的定义

```
&uart4 {
        status = "okay";            //将 okay 改为 disable
```

编译内核

```
./build.sh
```



编译完成后，需要将内核更新到 SD 卡中。

2、内核替换
准备一张烧录有 OrangePi 4 Linux 镜像的 SD 卡。通过读卡器插到 PC 的 usb 接口。

首先确定 SD 卡的设备节点。
拔出 SD 卡，执行 ls /dev/sd*

```
csy@ubuntu:~$ ls /dev/sd*
/dev/sda  /dev/sda1  /dev/sda2  /dev/sda5
```

插上 SD 卡，执行 ls /dev/sd*

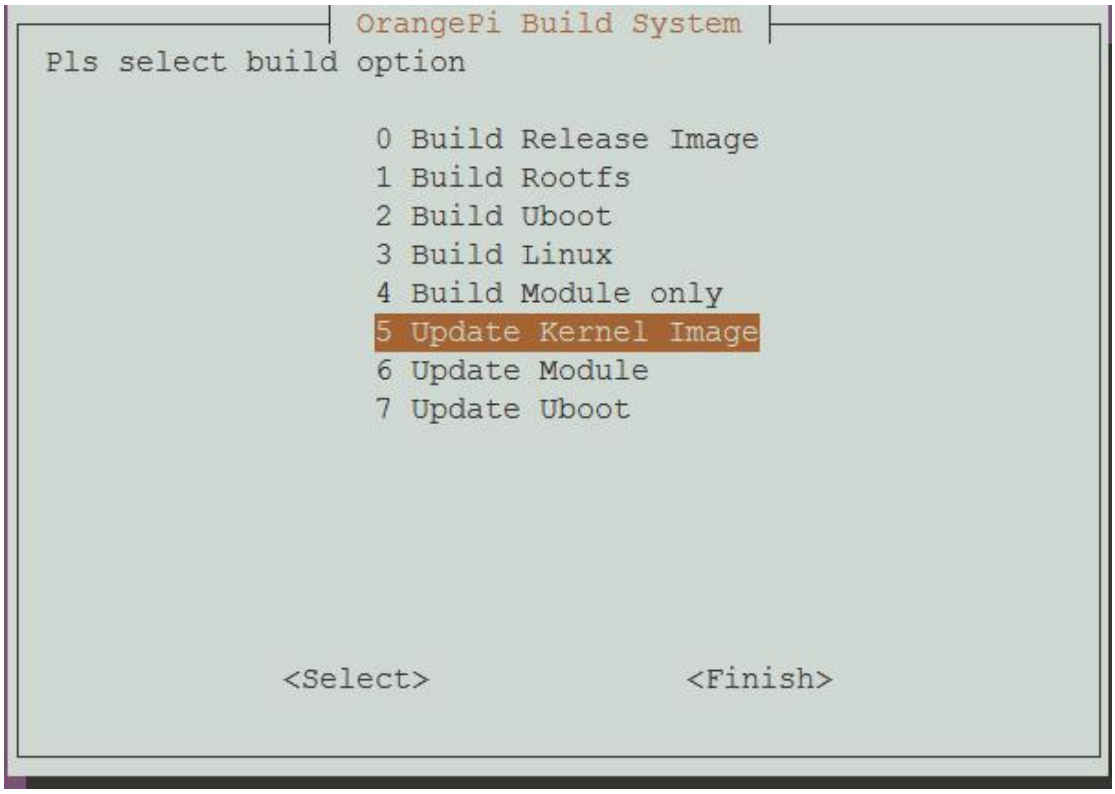
```
csy@ubuntu:~$ ls /dev/sd*
/dev/sda   /dev/sda2  /dev/sdb   /dev/sdb2  /dev/sdb4
/dev/sda1  /dev/sda5  /dev/sdb1  /dev/sdb3
```

可知 SD 卡对应的设备节点是/dev/sdb

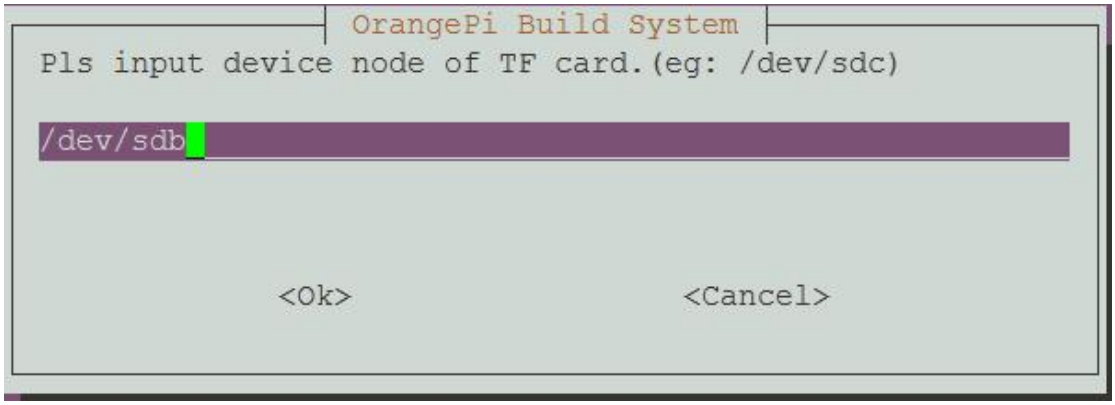
执行 build.sh

```
./build.sh
```

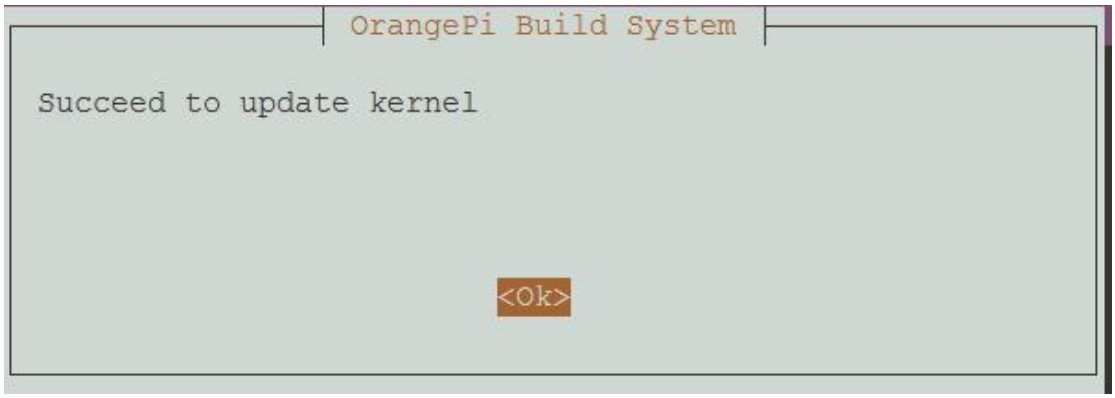
选择 5 Update Kernel Image



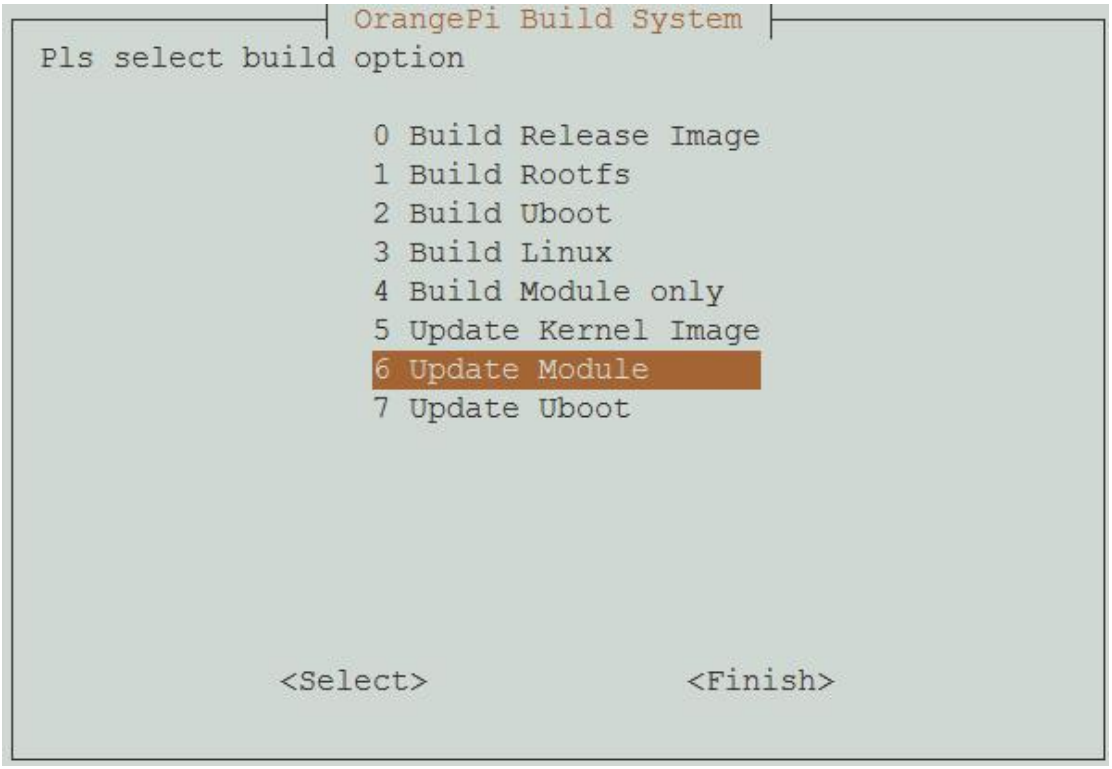
输入/dev/sdb



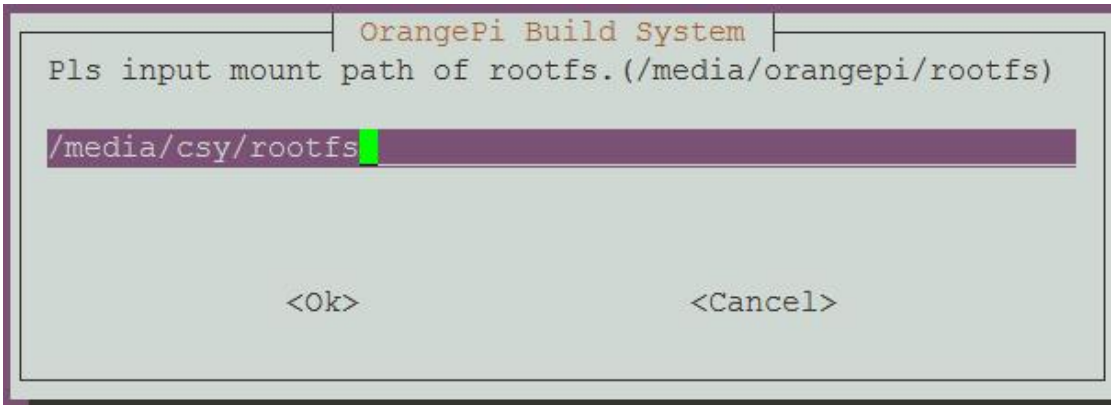
更新完成。



内核模块最好也更新下。选择 6 Update Module



SD 卡的 rootfs 分区挂载点是/media/csy/rootfs，每个人的环境会有所不同



回车。

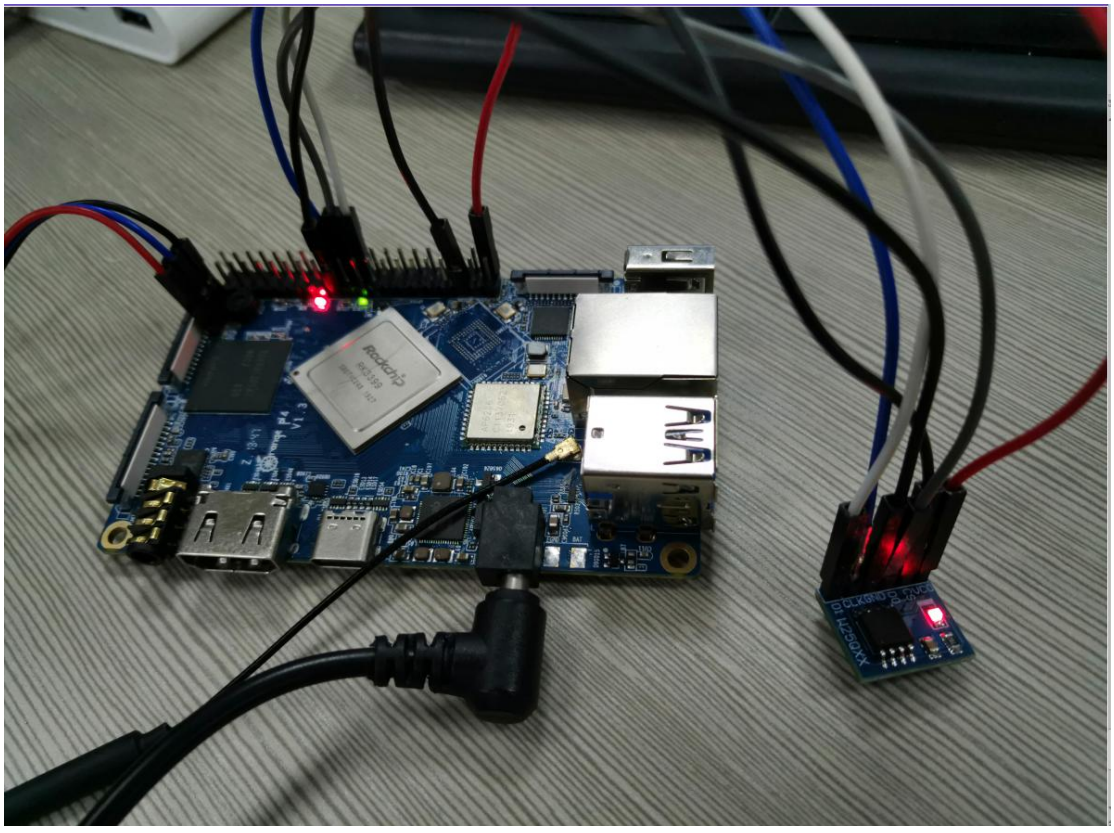
SD 卡插入 OrangePi 4，启动系统。Spi 已经启用

```
root@OrangePi:~# ls /dev/spidev1.0
/dev/spidev1.0
```

wiringOP 提供了一个 w25q64_test.c 用于测试 SPI。
需要用到 W25QXX 模块。
硬件接线如下，具体引脚定义可查看 OrangePi 4 的原理图

VCC - 1
CS - 24
DO - 21
GND - 6
CLK - 23
DI - 19

注：左边为 W25QXX 模块的引脚，右边为 40pin 的物理编号。



编译测试程序

```
cd wiringOP/examples/
make w25q64_test
```

运行测试程序。

```
root@OrangePi:~/wiringOP/examples# ./w25q64_test
Unable to open SPI device: No such file or directory
root@OrangePi:~/wiringOP/examples#
```

出现以上报错的原因是 wiringOP 对于 spi 通道 0 访问的设备节点是 /dev/spidev0.0。

```
static const char       *spiDev0  = "/dev/spidev0.0" ;
static const char       *spiDev1  = "/dev/spidev0.1" ;
```

而 RK3399 平台名称是/dev/spidev1.0.
所以创建一个软链接即可。

```
ln -s /dev/spidev1.0 /dev/spidev0.0
```

运行测试程序。

```
root@OrangePi:~/wiringOP/examples# ./w25q64_test
JEDEC ID : ef 40 17
Unique ID : df 66 80 12 83 67 33
Read Data: n=256
```

能读取 ef 40 17 说明 SPI 通信正常。

4、I2C 的使用说明

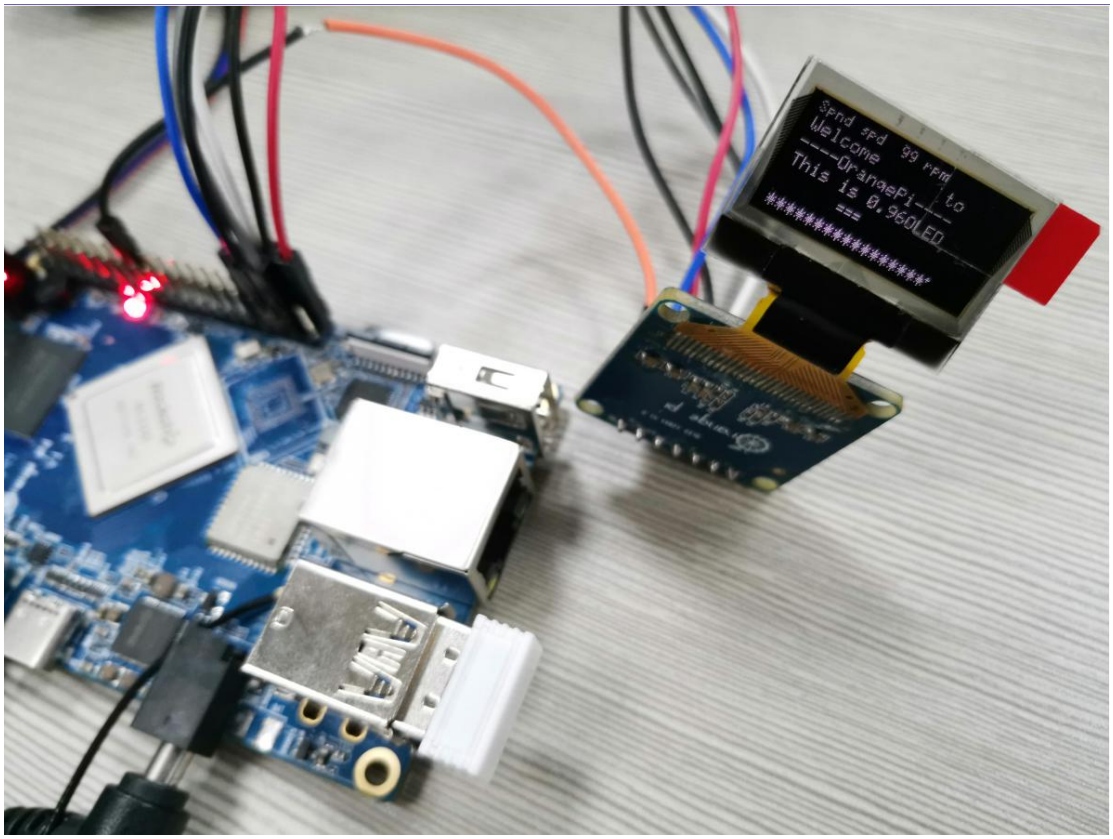
wiringOP/examples 中移植了一个 oled_demo.c 测试程序，可以使用 OrangePi 的 0.96
寸 OLED 模块测试 I2C 接口的功能。
OrangePi4 40pin 上有两个 i2c 通道，分别是 i2c2，i2c3。对应的引脚
3 - I2C2_SDA
5 - I2C2_SCL
8 - I2C3_SCL
10 - I2C3_SDA
具体可以查看 OrangePi 4 的原理图。

I2C2 的测试
接线：GND-6、VCC-4，SCL-5，SDA-3，RST-1，DC-9，CS-25
注：左边为 oled 模块的引脚，右边为 40pin 的物理编号。

```
cd wiringOP/examples
make oled_demo
```

执行测试程序

```
./oled_demo /dev/i2c-2
```



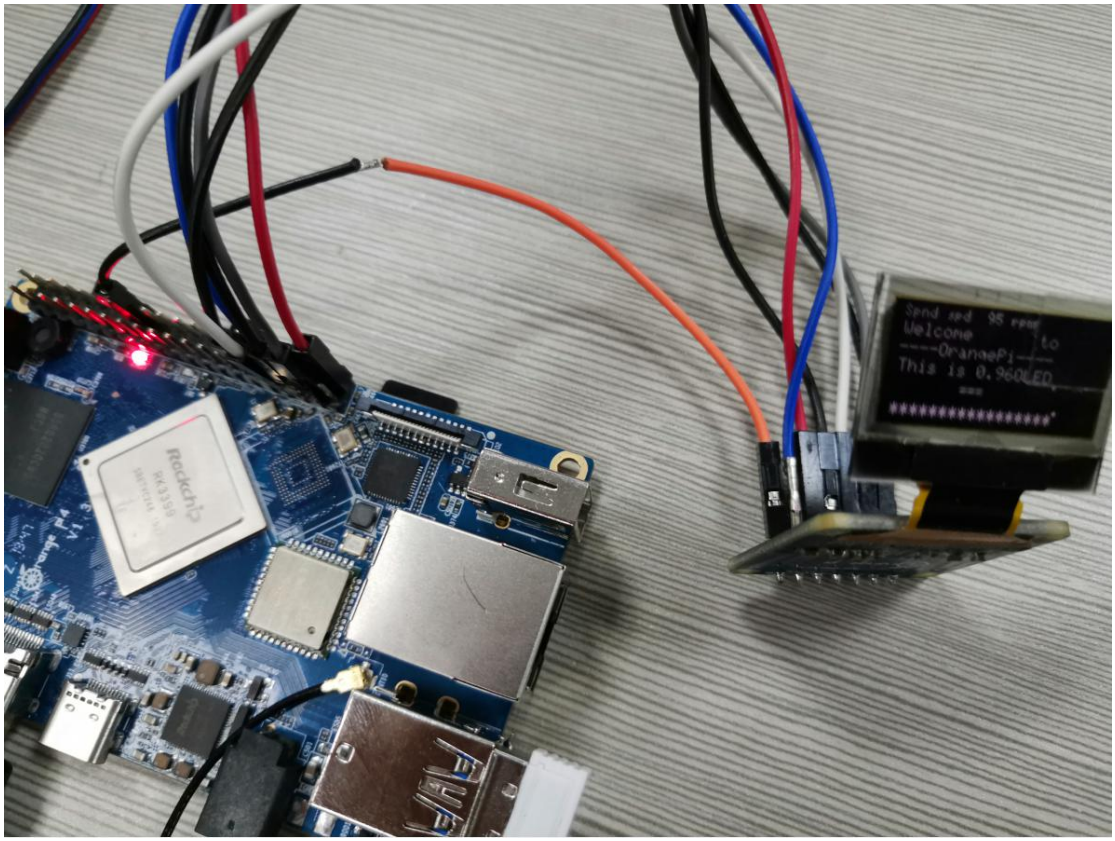
I2C 3 的测试

接线只有 SDA 和 SCL 需要改，其他不需要动。

SCL-8，SDA-10

运行测试程序

```
./oled_demo /dev/i2c-3
```


Welcone to
-----OrangePi-----
This is 0.96OLED
rockchip

5、看门狗测试代码

```
#include <stdio.h>
#include <errno.h>
#include <fcntl.h>
#include <linux/watchdog.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <sys/ioctl.h>

#define DEV_NAME "/dev/watchdog"

int main(int argc, char *argv[])
{
        int interval = 10;
    if (argc >= 2) interval = atoi(argv[1]);
    int margin = 10;
    if (argc >= 3) margin = atoi(argv[2]);

        printf("watchdogd started interval %d, margin %d !\n", interval,
margin);
        int fd = open(DEV_NAME, O_RDWR|O_CLOEXEC);
    if (fd == -1) {
        printf("Failed to open %s.\n", DEV_NAME) ;
        return 1;
    }

        int timeout = interval + margin;
        int ret = ioctl(fd, WDIOC_SETTIMEOUT, &timeout);

    if (ret) {
        printf("Failed to set timeout to %d\n", timeout);
        ret = ioctl(fd, WDIOC_GETTIMEOUT, &timeout);
        if (ret) {
            printf("Failed to get timeout\n");
        } else {
            if (timeout > margin) {
                interval = timeout - margin;
            } else {
                interval = 1;
            }
            printf("Adjusted interval to timeout returned by driver: \
```

```
                        timeout %d, interval %d, margin %d.\n", timeout,
interval, margin);

        }
    }
        ret = ioctl(fd, WDIOC_GETTIMEOUT, &timeout);
        printf("get timeout: %d.\n)", timeout);

        while (1) {
         write(fd, "", 1);
         sleep(interval);
    }
}
```

编译

```
gcc watchdog.c -o watchdog
```

执行

```
./watchdog
```

按 ctrl+C 退出后，会显示超时时间，时间到后会重启系统。