

Lite Version Tutorial

1. Environment preparation

Write the official horizon image to the motherboard, and install the WiringPi tool on the RDK-X3 after boot

```
sudo apt update
sudo apt install git-core
```

Get the WiringPi source code online via git and execute the following command

```
git clone https://gitee.com/study-dp/wiringPi.git
```

Go to the WiringPi directory to install WiringPi. Run the build.sh script to automatically compile and install the WiringPi library

```
cd wiringPi
./build
```

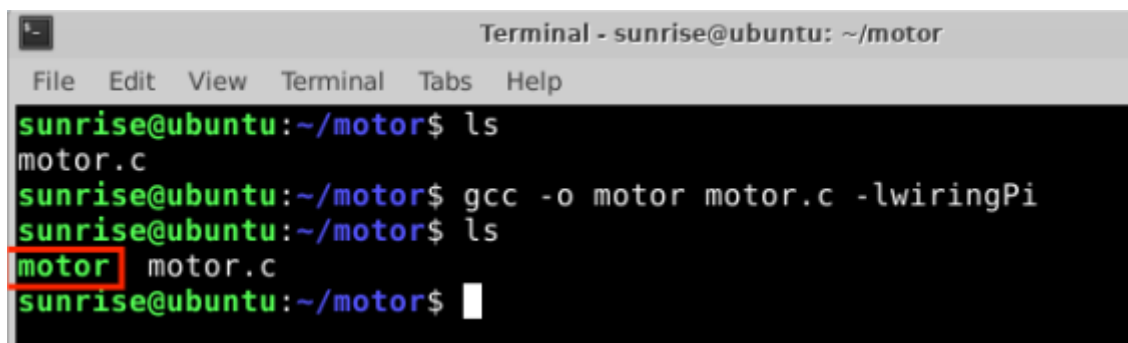
Create a working folder on the motherboard and paste the motor.c file in the file

```
cd && mkdir motor && cd motor
```

Compile the motor.c file. If you do not install the gcc compiler, please install the gcc compiler by yourself

```
gcc -o motor motor.c -lwiringPi
```

You can see that the terminal displays a green executable file

A terminal window titled "Terminal - sunrise@ubuntu: ~/motor" with a menu bar (File, Edit, View, Terminal, Tabs, Help). The terminal shows the following commands and output:

```
sunrise@ubuntu:~/motor$ ls
motor.c
sunrise@ubuntu:~/motor$ gcc -o motor motor.c -lwiringPi
sunrise@ubuntu:~/motor$ ls
motor  motor.c
sunrise@ubuntu:~/motor$
```

The word "motor" in the final 'ls' output is highlighted with a red box.

Refer to the cable preparation below, after the cable is connected, you can directly run the file

```
sudo ./motor
```

2. Prepare cable connections

Connect the AT8236 thin module to the RDK-X3 according to the wiring diagram. The battery is connected to the power input port of the module. (**VM and GND are the power input ports of the module.**), the input voltage range is 5~12V, which is determined according to the working voltage of the motor.

Where M+ and M- are the two power inputs of the motor respectively, which are determined according to their own motor pins

RDK-X3(40pin)	AT8236 lite module		AT8236 lite module	motor
GPIO6	AIN1		AOUT1	M+
GPIO5	AIN2		AOUT2	M-
GPIO30	BIN1		BOUT1	M+
GPIO27	BIN2		BOUT2	M-
3V3	VCC			

***Note: The line sequence of the motor interface must correspond to the motor pin! No match will burn the drive board!! ***

3. Main code display

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

/* GPIO config define */
#define AIN1 6
#define AIN2 5
#define BIN1 30
#define BIN2 27

/* Parameter define */
static int high_edge_time = 0; //初始化设定值 initialValue
static int cycle_time = 500; //pwm范围0~500 pwmRange

void forward(void)
{
    softPwmWrite(AIN1, 500); // duty = 500/cycle_time
    softPwmWrite(AIN2, 0);
    softPwmWrite(BIN1, 500);
    softPwmWrite(BIN2, 0);
}

void back(void)
{
    softPwmWrite(AIN1, 0);
    softPwmWrite(AIN2, 500); // duty = 500/cycle_time
    softPwmWrite(BIN1, 0);
    softPwmWrite(BIN2, 500);
}
```

```

void stop(void)
{
    softPwmWrite(AIN1, 0);          // duty = 0/cycle_time
    softPwmWrite(AIN2, 0);
    softPwmWrite(BIN1, 0);
    softPwmWrite(BIN2, 0);
}

int main (void)
{
    /* Initial gpio with chip pin ,if use mapping pin -> wiringPiSetup*/
    wiringPiSetupGpio () ;

    printf("cycle_time:%d\n", cycle_time);
    pinMode(AIN1, OUTPUT);
    pinMode(AIN2, OUTPUT);
    pinMode(BIN1, OUTPUT);
    pinMode(BIN2, OUTPUT);
    softPwmCreate(AIN1, high_edge_time, cycle_time);
    softPwmCreate(AIN2, high_edge_time, cycle_time);
    softPwmCreate(BIN1, high_edge_time, cycle_time);
    softPwmCreate(BIN2, high_edge_time, cycle_time);

    /* Control pwm */
    forward();
    printf("forward\n");
    delay (5000);
    stop();
    delay (1000);
    back();
    printf("back\n");
    delay (5000);
    stop();
    delay (1000);
    printf("over\n");
}

```

Horizon official wiringpi usage reference: <https://gitee.com/study-dp/WiringPi>

wiringpi official documentation: <http://wiringpi.com/reference/software-pwm-library/>

4. Experimental results

Run the program, drive board drive motor forward 5s, then reverse 5s.