

Remi Pi_Linux Software Evaluation Guide



File Status: [] Draft [√] Release	FILE ID:	MYIR-REMI PI-SW-DG-ZH-L5.10.83
	VERSION:	V1.0[Document]
	AUTHOR:	Gali
	CREATED:	2024-01-11
	UPDATED:	2024-01-13

Copyright © MYIR Electronics Limited 2010-2024 all rights reserved

Revision History

Revision	AUTHOR	PARTICIPANT	DATE	DESCRIPTION
V1.0[document]	Gali		20240111	Initial version, for Remi Pi

CONTENT

Revision History	2
CONTENT	3
1. Overview	9
1.1. Hardware Resources	9
1.2. Software Resources	9
1.3. Document resource	10
1.4. Preparation	10
2. Core Components	11
2.1. CPU	11
2.1.1. View the CPU Information	11
2.1.2. View CPU Utilization	12
2.1.3. Gets CPU Temperature	13
2.1.4. CPU Stress Test	13
2.2. Coprocessor Cortex M33	15
2.2.1. Development environment construction	15
2.2.2. Installing FSP	22
2.2.3. Heterogeneous communication	23
2.3. Graphics	28
2.4. Memory	29
2.4.1. Viewing memory information	29
2.4.2. Get memory usage	29
2.4.3. Memory Stress Test	30
2.5. eMMC	32
2.5.1. Check eMMC Capacity	32
2.5.2. View the eMMC Partition Information	32
2.5.3. eMMC Performance Test	33
2.5.4. eMMC Life Test	34
2.6. RTC	36
2.7. Watchdog	38

2.7.1. Stop the Watchdog	38
2.7.2. Watchdog testing in user space	38
2.7.3. Application test watchdog	39
2.8. EEPROM	41
2.9. Power Management.....	43
2.9.1. View the supported modes	43
2.9.2. Set the wake source	43
2.9.3. Sleep to memory	43
2.9.4. Wake up using the debug serial port.....	43
3. Peripheral Interface	45
3.1. GPIO	45
3.1.1. Export GPIO to User Space	45
3.1.2. Set/View the GPIO Direction	45
3.1.3. Set/View the GPIO Value	45
3.2. LED Lamp	47
3.2.1. View the LED0	47
3.2.2. Heartbeat indicator LED0 is used as an example	47
3.3. Key	48
3.3.1. Test Keys	48
3.4. USB HOST	50
3.4.1. View the printed information about the usb	50
3.4.2. The USB flash drive is mounted with read and write	51
3.4.3. Uninstall USB flash drive	51
3.5. USB OTG	52
3.5.1. Enumerate different peripherals	52
3.5.2. Read and Write functions	52
3.6. Micro SD card	53
3.6.1. Check TF Card Capacity	53
3.6.2. TF card Performance Test	53
3.7. Display	56
3.7.1. LVDS display test	56
3.8. Touch Panel	58
3.8.1. Touch screen connection	58

3.8.2. Touch test with evtest command	58
3.9. Backlight	61
3.9.1. View the current backlight level	61
3.9.2. View the maximum backlight level	61
3.9.3. Backlight adjustment	61
3.10. MIPI-CSI	62
3.10.1. i2cdetect Check whether the driver is loaded	62
3.10.2. Viewing device nodes	62
3.10.3. View the formats supported by the device	62
3.10.4. preview	65
3.10.5. Take pictures	66
4. Network interface	67
4.1. Ethernet	67
4.1.1. Manually configure a temporary static IP address	67
4.1.2. Automatically permanently configure a static IP address	68
4.2. WIFI	71
4.2.1. STA mode manually connects to the WIFI hotspot	71
4.2.2. STA mode script manually connects to WIFI hotspot	76
4.2.3. The STA automatically connects to the WIFI hotspot upon startup	76
4.2.4. Manually configure WIFI hotspots in AP mode	78
4.2.5. The AP mode script configures the WIFI hotspot	81
4.3. BLUETOOTH	82
5. Expand peripheral interfaces	85
5.1. MY-WiredCom Module	85
5.1.1. RS232	85
5.1.2. RS485	87
5.1.3. CAN	90
5.1.4. SPI	93
6. Network application	95
6.1. PING	95
6.1.1. Network Connection	95

6.1.2. PING Public Network	95
6.2. SSH	96
6.3. SCP	98
6.3.1. Copy Files from Remote to Local	98
6.3.2. Copy Files form Local to Remote	98
6.4. FTP	99
6.4.1. Login from Host PC to Target Device with FTP	99
6.4.2. Create test files on the development board	99
6.4.3. Development host FTP View the current directory	100
6.4.4. Download file	100
6.4.5. Upload file	100
6.5. TFTP	102
6.5.1. Install TFTP Server Application	102
6.5.2. Configure TFTP Server	102
6.5.3. Restart TFTP Service	103
6.6. DHCP	104
6.7. Iptables	105
6.7.1. Configure development board iptables	105
6.7.2. Ping Test	106
6.7.3. Delete Rules for iptables	106
6.7.4. Ping the Target Device Again	107
6.8. Ethtool	108
6.9. iperf3	110
6.9.1. Test Performance under TCP Mode	110
6.9.2. Test Performance under UDP Mode	113
7. Graphic system	116
7.1. GPU	116
7.1.1. Setting up the test environment	116
7.1.2. Start the kmscube test	116
7.1.3. Start the glmark2 test	117
7.2. QT	119
7.2.1. Check Qt information	119
7.2.2. Introduction of QT Running Environment	119

7.2.3. Start Qt Applications	121
8. Multimedia applications	123
8.1. VPU	123
8.1.1. View video format	123
8.1.2. Play a Video	124
8.2. Audio	125
8.2.1. HMI2.0 music playback	125
8.2.2. Play music manually	125
8.3. Video	127
8.3.1. HMI2.0 video play	127
8.3.2. Manual video playback	127
9. System Tools	128
9.1. Compress and Decompress Tools	128
9.1.1. tar tool	128
9.1.2. gzip tool	129
9.2. File System Tools	131
9.2.1. mount tool	131
9.2.2. mkfs tool	132
9.2.3. fsck tool	133
9.2.4. dumpe2fs	133
9.3. Disk Management Utils	136
9.3.1. fdisk tool	136
9.3.2. dd tool	137
9.3.3. du tool	137
9.3.4. df tool	138
9.4. Process Management Utils	140
9.4.1. ps tool	140
9.4.2. top tool	142
9.4.3. vmstat tool	142
9.4.4. kill tool	145
10. Development support	147

10.1. Development Language	147
10.1.1. SHELL	147
10.1.2. C/C++	148
10.1.3. Python	149
10.2. Database	151
11. Reference	152
Appendix A	153
Warranty & Technical Support Services	153

1. Overview

Remi Pi is an industrial-grade processor, based on Renesas RZ/G2L industrial CPU, compatible with all accessories of Raspberry Pi, rich software resources, support Debian/Ubuntu/Linux and other operating systems, convenient product prototype construction, can take into account development, learning and practical application. 《Remi Pi_Linux Software Evaluation Guide》 is used to introduce the test steps and evaluation methods of running the core resources and peripheral resources under the open source Linux system on the development board of Mier. This article can be used as a pre-evaluation guide, and can also be used as a test guide for general system development. In this document, The myir-image-full system will be explained as an example.

1.1. Hardware Resources

The Remi Pi of Mill Electronics is composed of the core plate MYC-YG2LX and the bottom plate MYB-YG2LX-REMI. The core plate and the bottom plate are welded by stamp hole. In addition, MYIR provides a wealth of software resources and documentation. Also, the user will use some accessories during the evaluation test, see the list below.

Table 1-1. Optional Modules

Accessories	Interface	Description
Camera	MIPI CSI	MY-CAM003M(5 megapixel) https://www.myirtech.com/list.asp?id=611
LVDS	LVDS	MY-LVDS070C (7 "touch screen with capacitive) https://www.myirtech.com/list.asp?id=634
RS232/RS485/ CAN Module	Raspberry PI	MY-WiredCom https://www.myirtech.com/list.asp?id=665

1.2. Software Resources

The BSP of Remi Pi series development board is based on Renesas official open source community edition Linux BSP transplantation and modification, and the system image is built by Yocto project. Flash-Write, Bootloader, Kernel and all parts of the file system software resources are open in the form of source code. For details, please see the 《Remi Pi_SDK Release Notes》. The development board has burned the system image according to the core board model when it leaves the factory. You only need to power on to use it.

1.3. Document resource

According to the different stages of the user's use of the development board, the SDK contains different types of documents and manuals such as documents for each stage, release instructions, evaluation guidelines, development guidelines, application notes and so on. For a list of documentation, see the 《Remi Pi_SDK Release Notes》.

1.4. Preparation

Before you start to evaluate the development board software, you need to make some necessary preparations for the development board and configure some basic environment, including the correct hardware wiring, configuring the debugging serial port, setting up the boot and other steps. Detailed steps can be found in the 《Remi Pi_Quick Start Guide》.

The following sections focus on how to evaluate and test the hardware resources and interfaces of the system as well as the software functions. It mainly uses some commonly used tools and commands under Linux, as well as self-developed applications to test. Software evaluation guide is divided into several parts to describe, including: core resources, peripheral resources, network applications, multimedia applications, development support applications, system tools and other major categories. The following chapters will give a comprehensive explanation of each part, and describe the specific evaluation methods and steps of each part of the resources in detail.

2. Core Components

In Linux system, the proc virtual file system is provided to query the parameters of various core resources and some general tools are provided to evaluate the performance of resources. The following will read and test the parameters of core resources such as CPU, memory, eMMC, and RTC.

2.1. CPU

Remi Pi uses Renesas RZ/G2L as the core processor, which is equipped with dual-core Cortex-A55@1.2GHz+Cortex-M33@200MHz processor. It integrates Mail-G31 GPU(500MHz), a high performance 3D acceleration engine, video processing unit (H.264 hardware codec support), and 16-bit DDR4-1600 / DDR3L-1333 memory controller, Gigabit Ethernet controller, USB, CAN, SD card, MIPI-CSI and other peripheral interface, in industry, medical, power and other industries have been widely used.

In this section, we will use Mill's Remi Pi development board as an example to illustrate the test.

2.1.1. View the CPU Information

Read the CPU provider and parameter information in the system, which can be obtained through the /proc/cpuinfo file.

```
root@myir-remi-1g:/# cat /proc/cpuinfo
processor       : 0
BogoMIPS      : 48.00
Features       : fp asimd evtstrm crc32 atomics fphp asimdhp cpuid asimdrr
m lrcpc dcpop asimddp
CPU implementer : 0x41
CPU architecture: 8
CPU variant    : 0x2
CPU part       : 0xd05
CPU revision   : 0
```

```
processor      : 1
BogoMIPS      : 48.00
Features      : fp asimd evtstrm crc32 atomics fphp asimdhp cpuid asimdrrm lrcpc dcpop asimddp
CPU implementer : 0x41
CPU architecture: 8
CPU variant    : 0x2
CPU part       : 0xd05
CPU revision   : 0
```

- Processor: The number of logical processing cores in a system can be either a physical core for multi-core processors or a virtual logical core using hyperthreading technology.
- BogoMIPS : A rough measure of the number of millions of instructions the CPU runs per second at kernel startup (Million Instructions Per Second).

2.1.2. View CPU Utilization

```
root@myir-remi-1g:/# top
top - 00:53:19 up 33 min,  1 user,  load average: 0.06, 0.02, 0.00
Tasks: 101 total,  1 running, 100 sleeping,   0 stopped,   0 zombie
%Cpu(s):  0.5 us,  0.2 sy,  0.0 ni, 99.1 id,  0.0 wa,  0.2 hi,  0.2 si,  0.0 st
MiB Mem :  855.5 total,  273.2 free,  339.6 used,  242.6 buff/cache
MiB Swap:   0.0 total,   0.0 free,   0.0 used.  440.3 avail Mem

   PID USER      PR  NI   VIRT   RES   SHR S  %CPU  %MEM    TIME+  COMMAND
  364 root        20   0 365092  68680 36732 S   0.6   7.8   0:15.23 mxa
pp2
  392 root        20   0 1328628 45364 19252 S   0.6   5.2   0:08.34 con
tainerd
  555 root        20   0   4744   2484  1972 R   0.6   0.3   0:00.18 top
  383 root        20   0 1432752 76644 34208 S   0.3   8.7   0:09.94 doc
kerd
  540 root        20   0     0     0     0 I   0.3   0.0   0:00.10 kworke
r/u4:0-flush-179:0
```

```
1 root      20   0  92300   7708   5124 S    0.0   0.9   0:07.31 syste
md
```

- %us: Represents the CPU utilization of the user space program.
- %sy: Represents the cpu usage of system space, mainly kernel programs.
- %ni: Represents the cpu usage of user space and the programs scheduled by nice.
- %id: Idle cpu.
- %wa: Percentage of CPU time waiting for input and output.
- %hi: The number of hard interrupts processed by the cpu.
- %si: Number of soft interrupts processed by the cpu.
- %st: real-time.

2.1.3. Gets CPU Temperature

The built-in temperature sensor of the CPU is used to collect the temperature of the CPU.

```
root@myir-remi-1g:/# cat /sys/class/thermal/thermal_zone0/temp
50500
```

The number shown above is in the unit of one thousandth of degree centigrade, the result divided by 1000 is the current temperature.

2.1.4. CPU Stress Test

There are many ways to test the CPU pressure. For example, "bc" command can be used to calculate the PI, and we can test the stability of the CPU in the process of operation by this means.

```
root@myir-remi-1g:/# echo "scale=5000; 4*a(1)" | bc -l -q &
[1] 565
```

The above command will calculate the PI in the background and accurate to 5000 decimal places. The calculation process needs a period of time. At this time, we can check the variation of CPU utilization by the top command, as follows:

```
root@myir-remi-1g:~# top
top - 00:59:56 up 39 min,  1 user,  load average: 0.24, 0.16, 0.08
Tasks: 103 total,   2 running, 101 sleeping,   0 stopped,   0 zombie
```

```
%Cpu(s): 54.3 us,  5.7 sy,  0.0 ni, 40.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   855.5 total,   272.4 free,   340.1 used,   243.0 buff/cache
MiB Swap:    0.0 total,    0.0 free,    0.0 used.  439.8 avail Mem
  PID USER      PR  NI   VIRT   RES   SHR S  %CPU  %MEM    TIME+
COMMAND
    565 root        20   0   2760   1764   1424 R   88.2   0.2   0:17.06 bc
    383 root        20   0 1432752  76644  34208 S   11.8   8.7   0:11.88 doc
kerd
erd
. . .
```

After about 3 minutes, the result of PI was calculated. The CPU usage was very high and there were no exceptions, indicating that the CPU stress test had passed. By increasing the accuracy requirement, the test pressure can be further improved.

```
root@myir-remi-1g:/# 3.1415926535897932384626433832795028841971693993
75105820974944592307\
8164062862089986280348253421170679821480865132823066470938446095505
8\
.....
```

2.2. Coprocessor Cortex M33

The RZ/G2L is a multi-core heterogeneous processor consisting of a dual-core Cortex-A55 and a single-core Cortex-M33. It can run both soft real-time Linux and hard real-time RTOS. This chapter mainly introduces the development environment construction of coprocessor CM33 and the communication between heterogeneous examples.

2.2.1. Development environment construction

The M33 nuclear development mainly used Renesas e2 studio tool, which can be downloaded from Renesas official website, https://www.renesas.cn/cn/zh/software-tool/e2studio-information-rz-family?dow_secondary=visible#download, You can also obtain Cortex M33-example.zip from 04_sources in the resource package on MYiR, It contains e2 studio tools, FSP packages, and heterogeneous communication examples. First, unzip the e2 studio tool in Cortex M33-example.zip, double-click it to open it, and select [All Users], as shown in Figure 2-1:

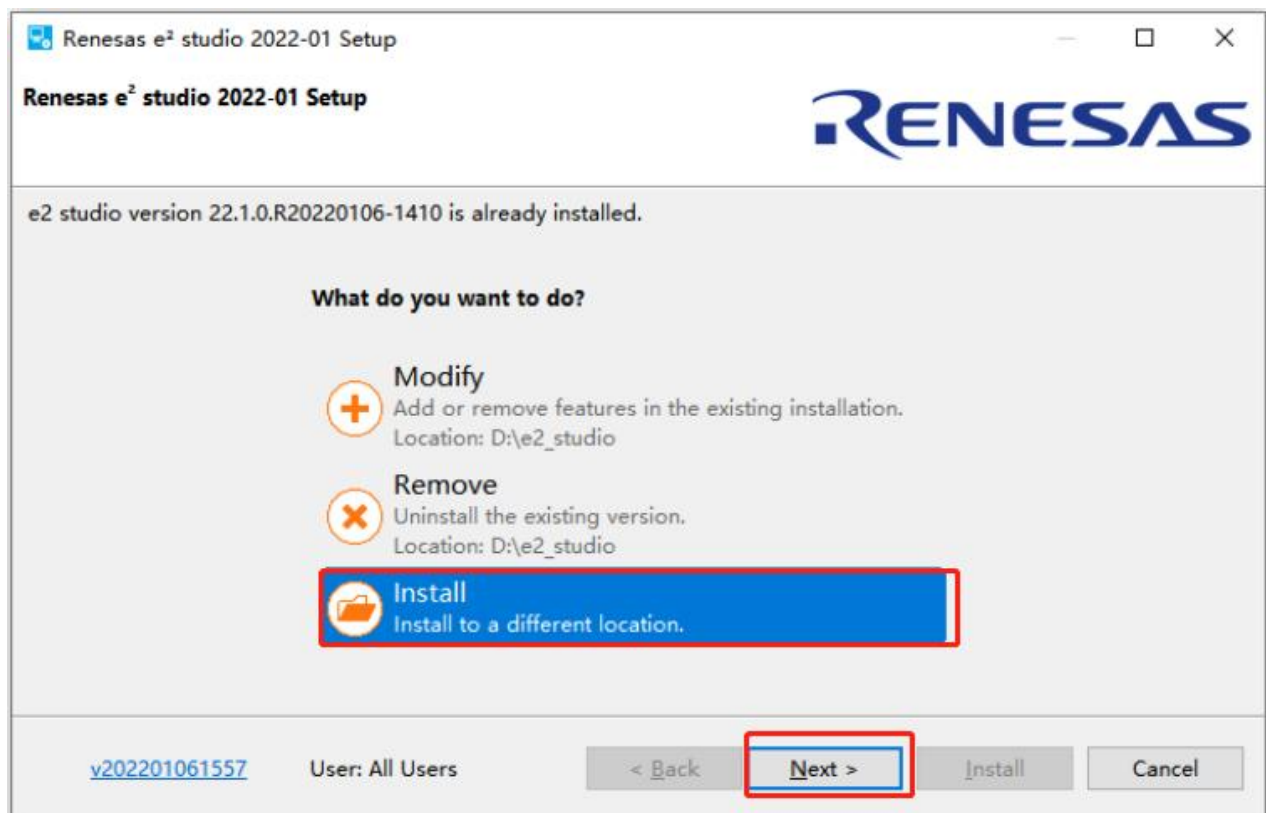


Figure 2-1. Installation selection

Select [Install]->[Next], shown in Figure 2-2:

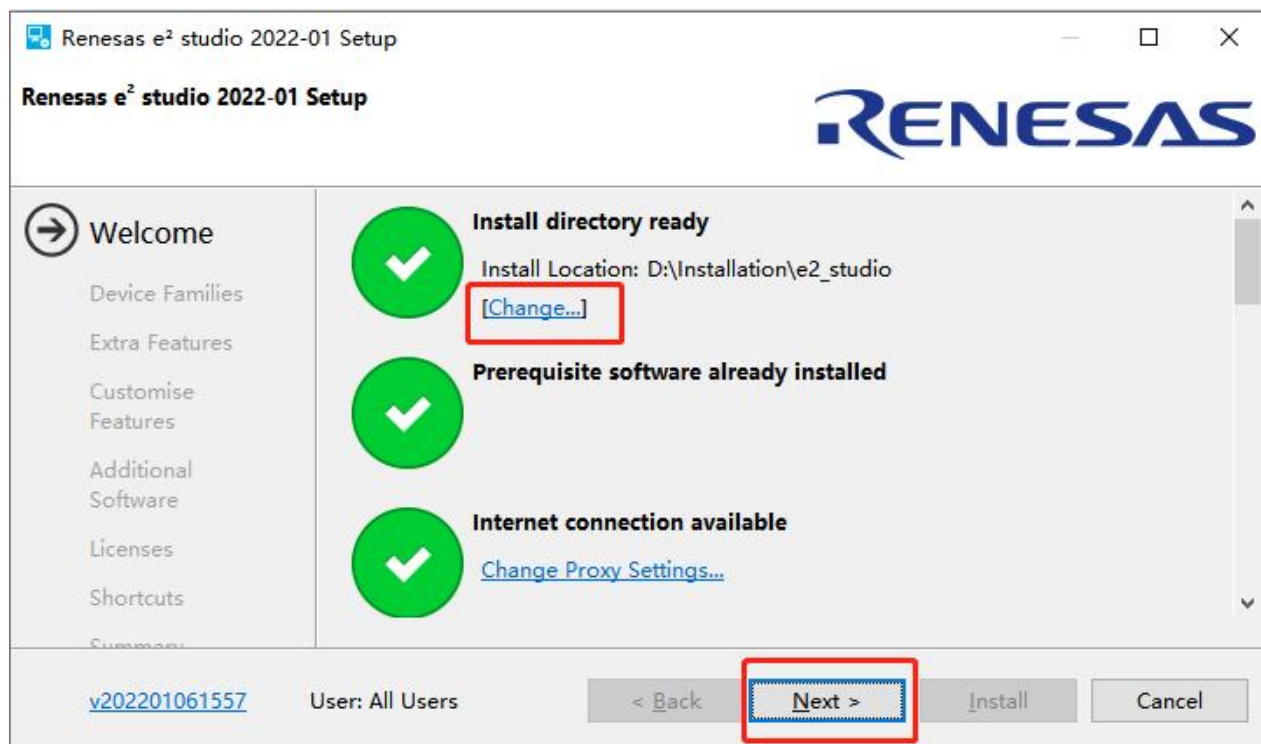


Figure 2-2. Path assignment

Choose [Change...] To specify the installation path. After specifying the installation path, click OK to continue, as shown in Figure 2-3:

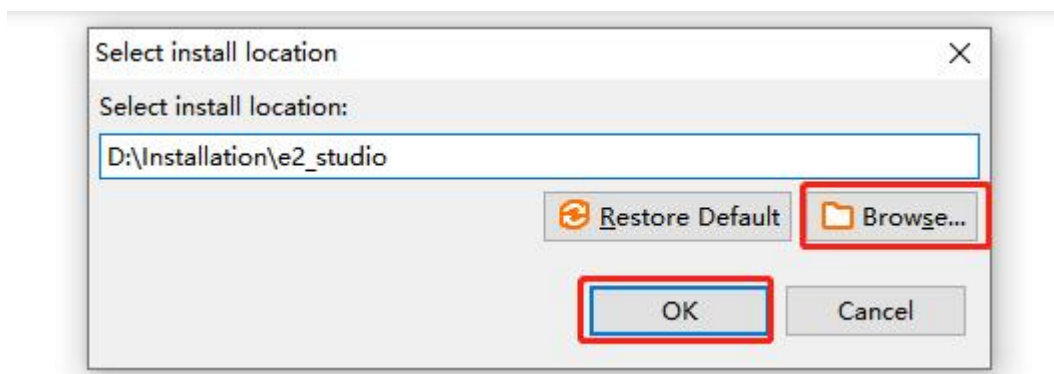


Figure 2-3. Path selection

Select the device family to be installed and click the [Next] button to continue, as shown in Figure 2-4:

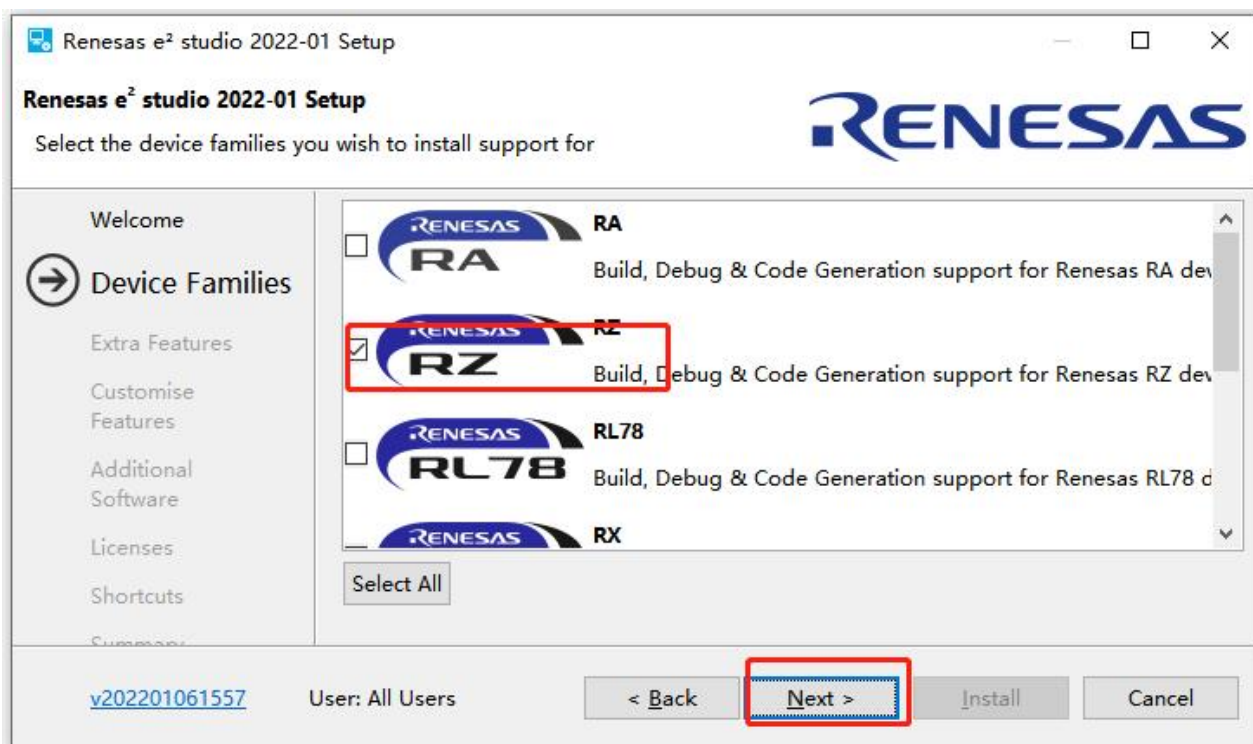


Figure 2-4. Device family selection

Additional Features Select the "additional features" to be installed (language pack, Git support, RTOS support...) Click next to continue, as shown in Figure 2-5:

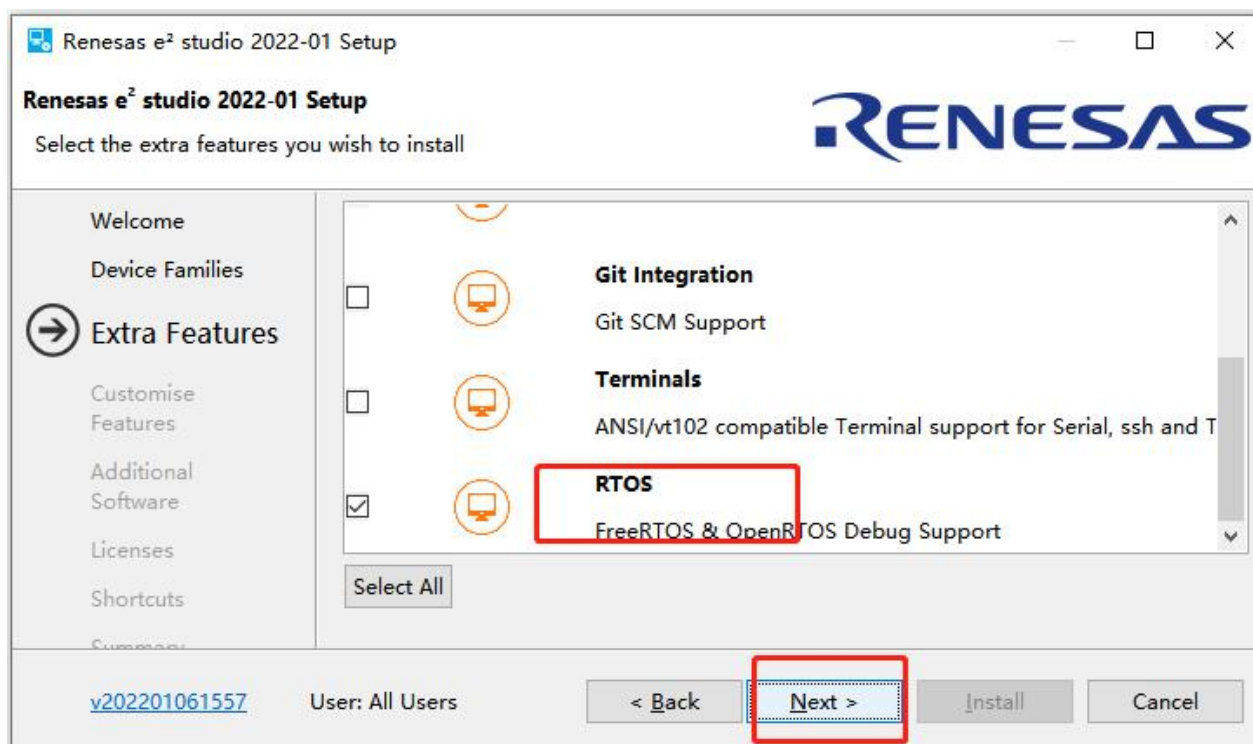


Figure 2-5. RTOS selection

Select the components you want to install and click [Next] to continue. Check that Renesas FSP Smartl Configurator Core is indeed selected. Otherwise, FSP cannot be built on e2 studio, as shown in Figure 2-6:

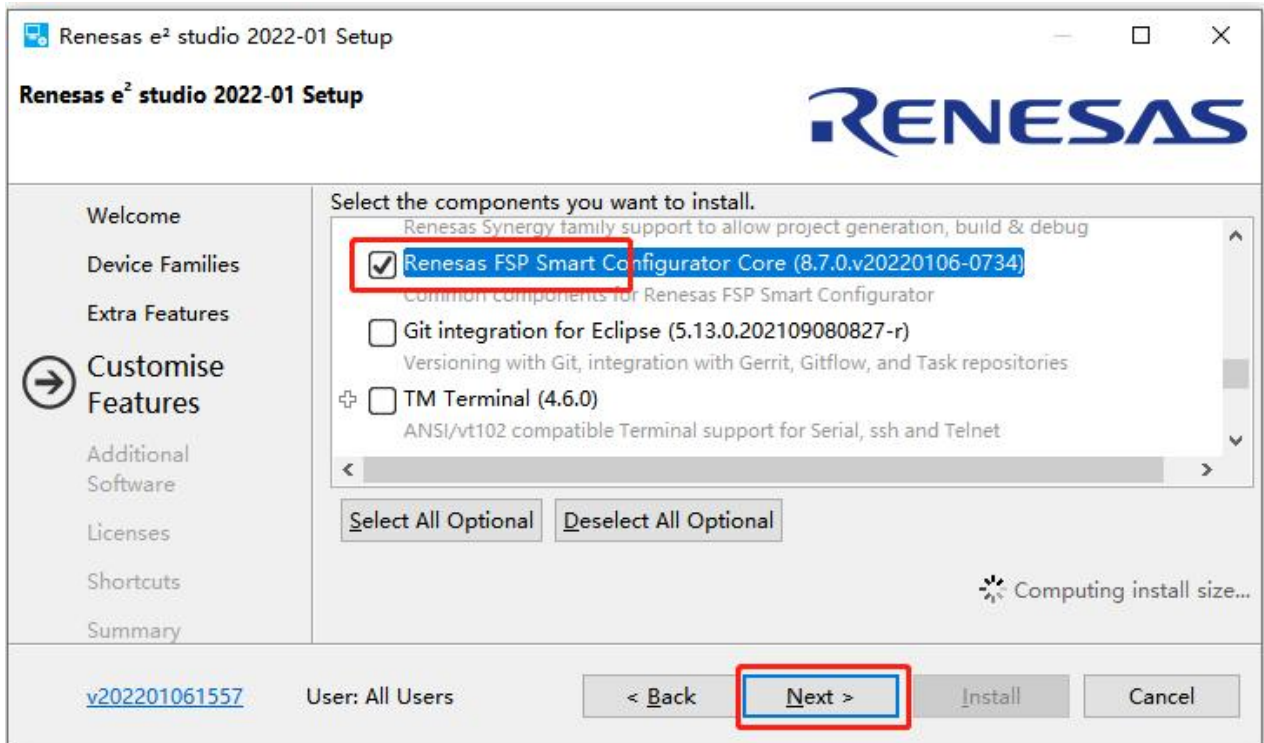


Figure 2-6. FSP selection

Select additional software to install ,as shown in Figure 2-7:

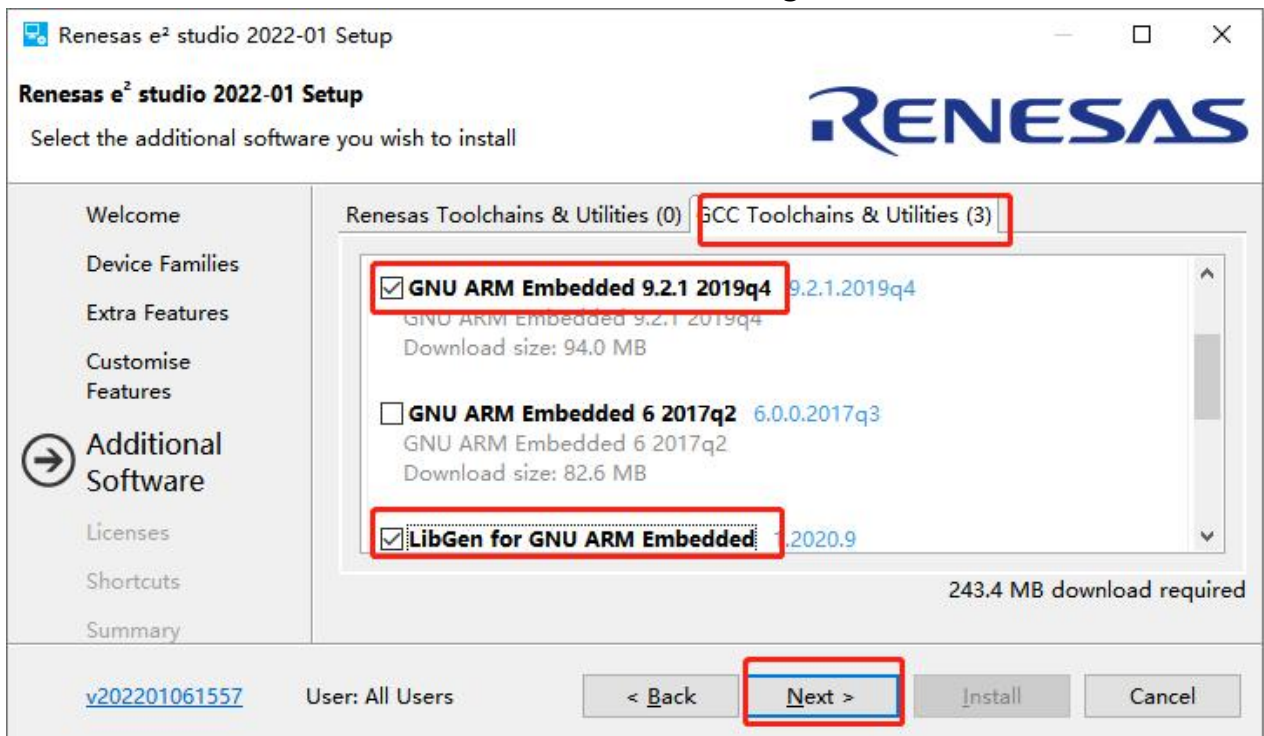


Figure 2-7. Compiler selection

The license agreement is agreed, as shown in Figure 2-8:

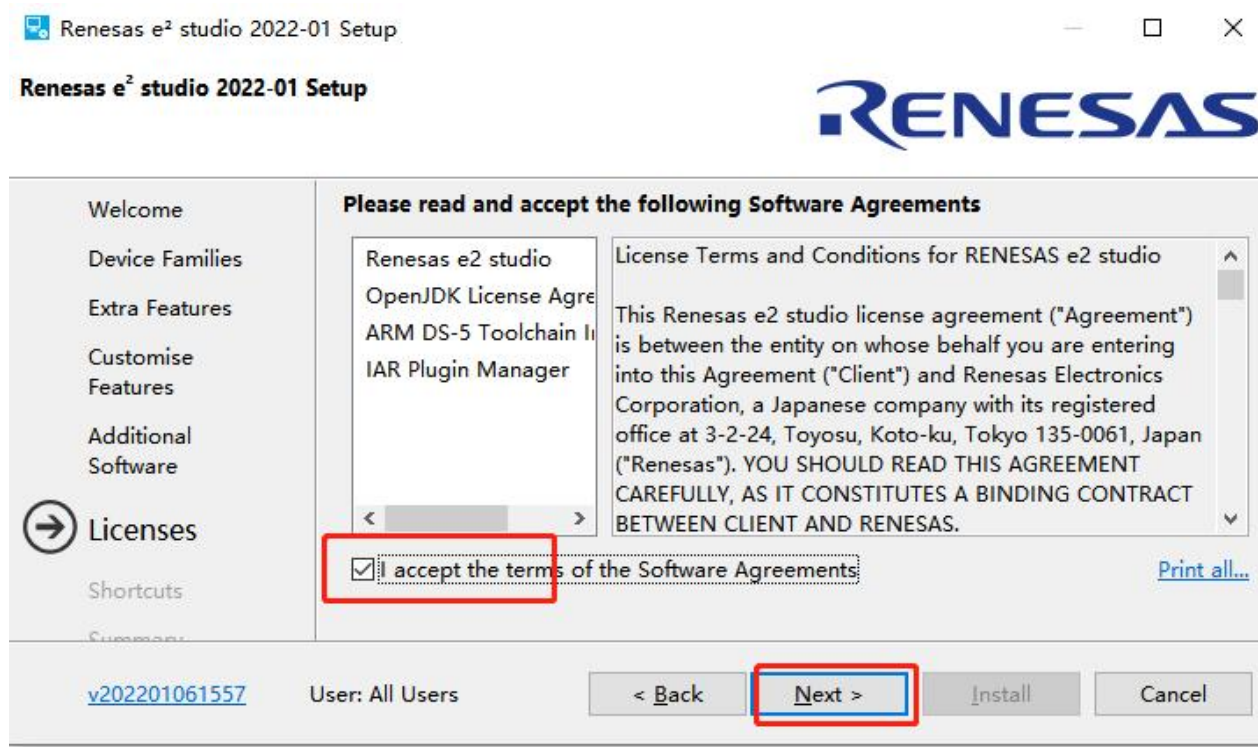


Figure 2-8. License option

Confirm the content and click Install, as shown in Figure 2-9:

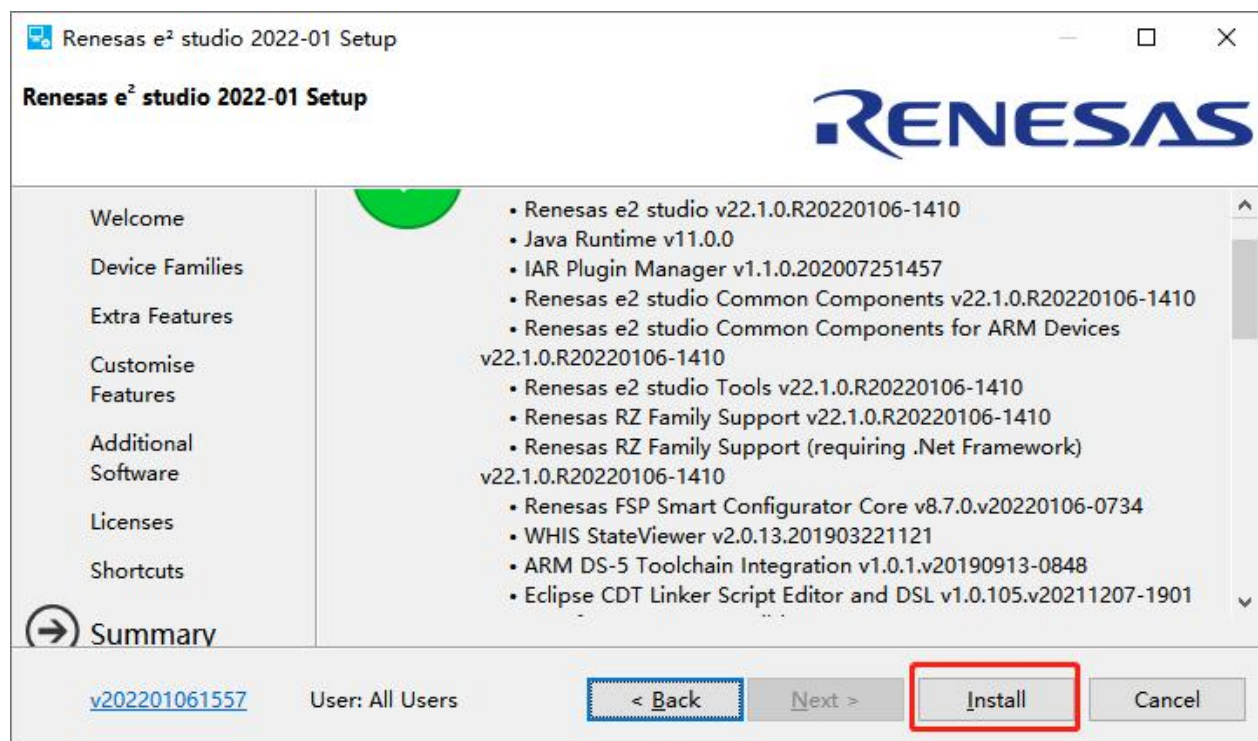


Figure 2-9. Confirm installation selection

The page for installing GNU ARM Embedded Toolchain 9.2.1 2019q4 is displayed, as shown in Figure 2-10:

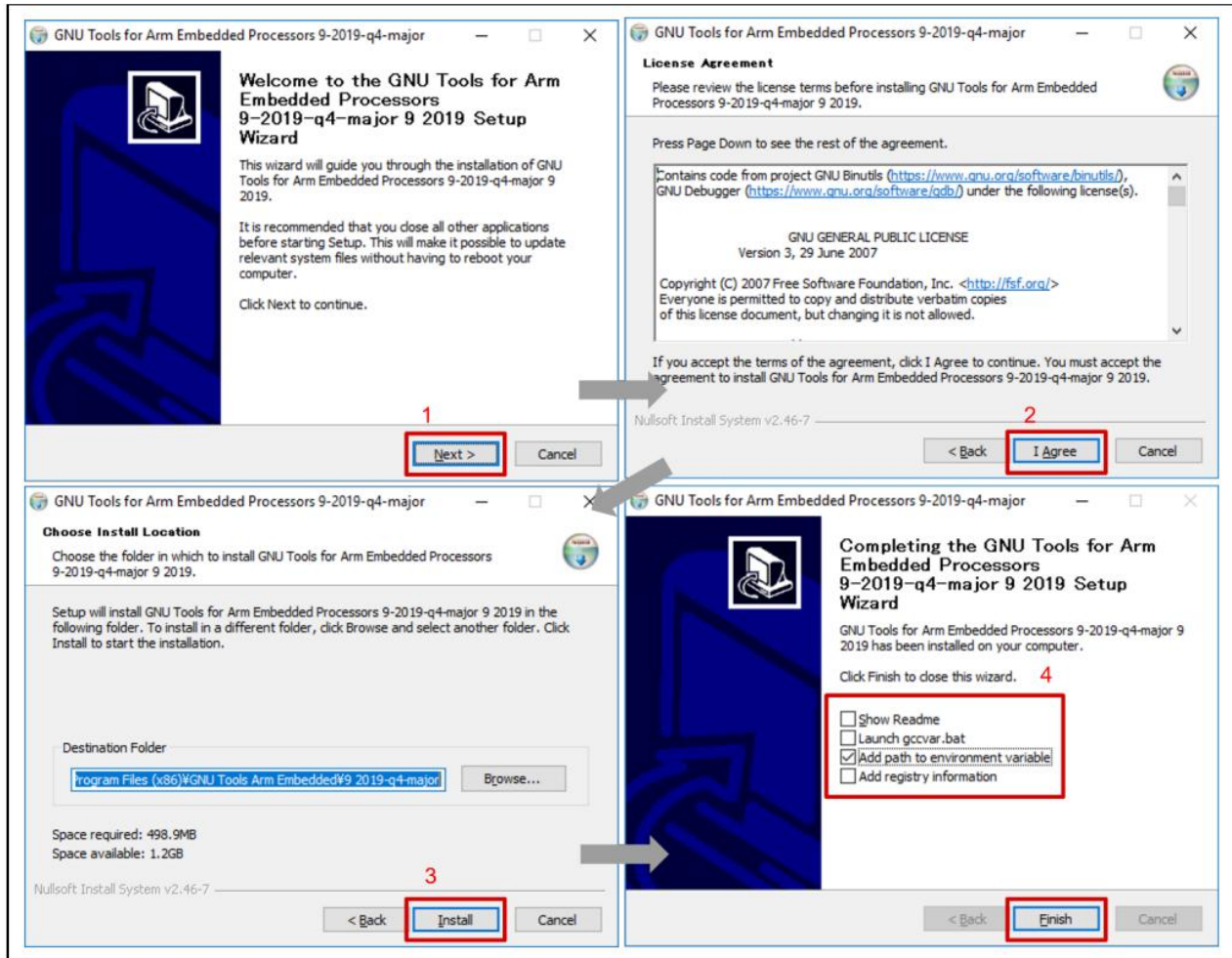


Figure 2-10. Tool chain selection

Libgen updates the GNU ARM Embedded Toolchains. If it is selected in the Add-on Software pane of e2 studio, during installation you will see the Libgen Update for GNU ARM Embedded Toolchains installation wizard, as shown in Figure 2-11:

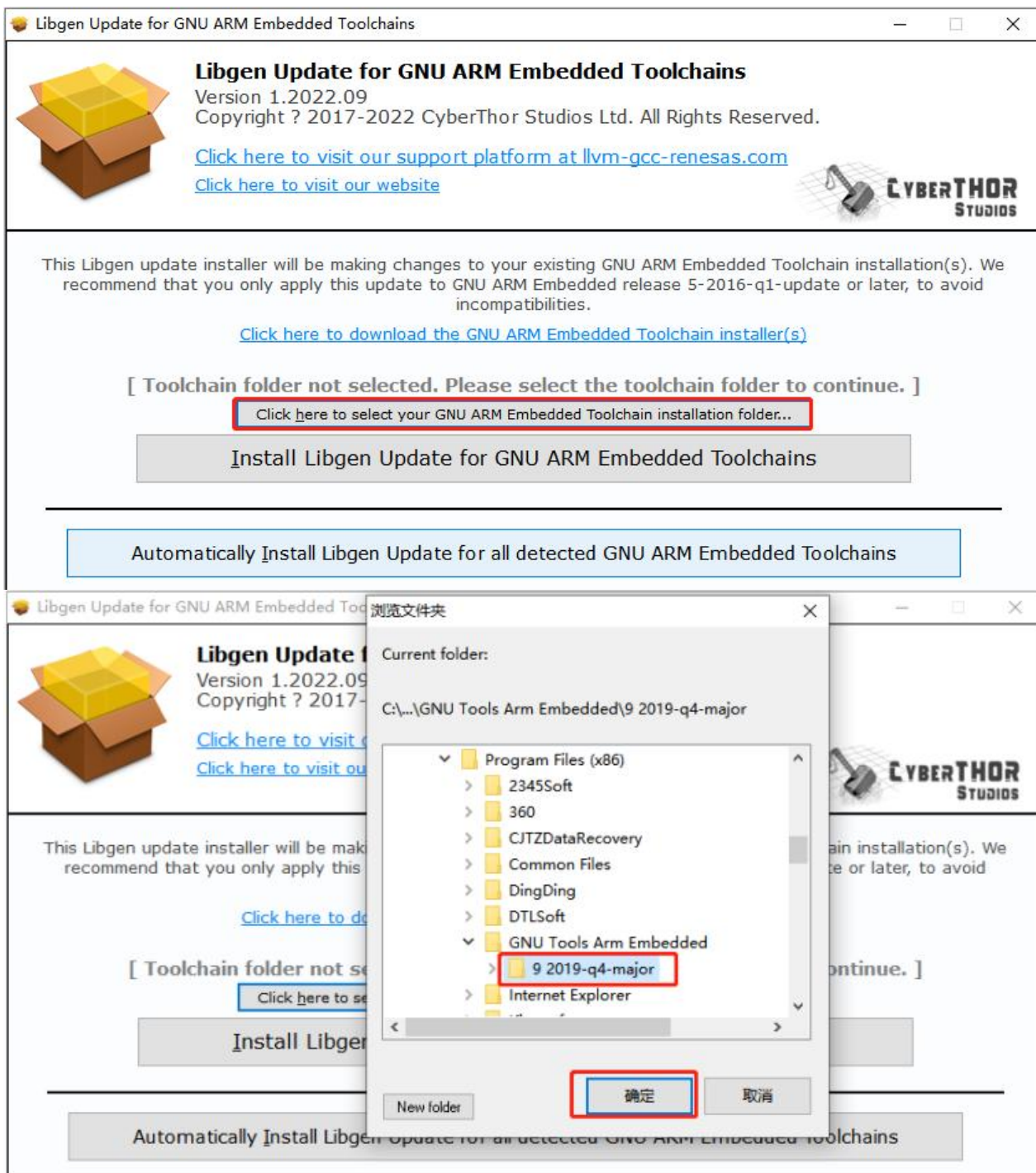


Figure 2-11. The tool chain installation directory is selected

Next, click Libgen Update to install the GNU ARM embedded Toolchain, as shown in Figure 2-12:

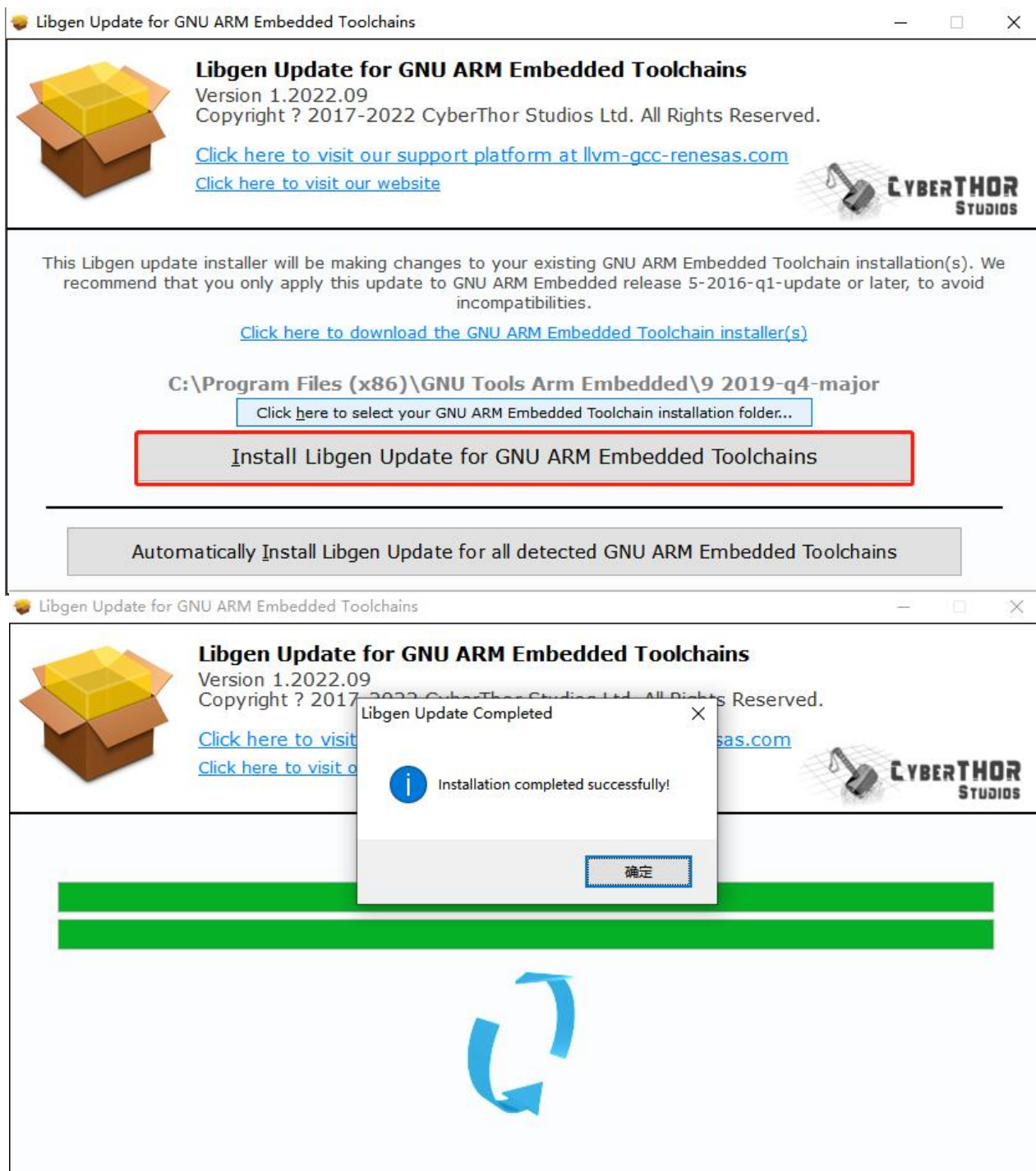


Figure 2-12. Libgen update

2.2.2. Installing FSP

Decompress the Cortex M33-example.zip package in the 04_sources directory and double-click RZG_FSP_Packs_v1.2.0.exe to install the fsp package, as shown in Figure 2-13:

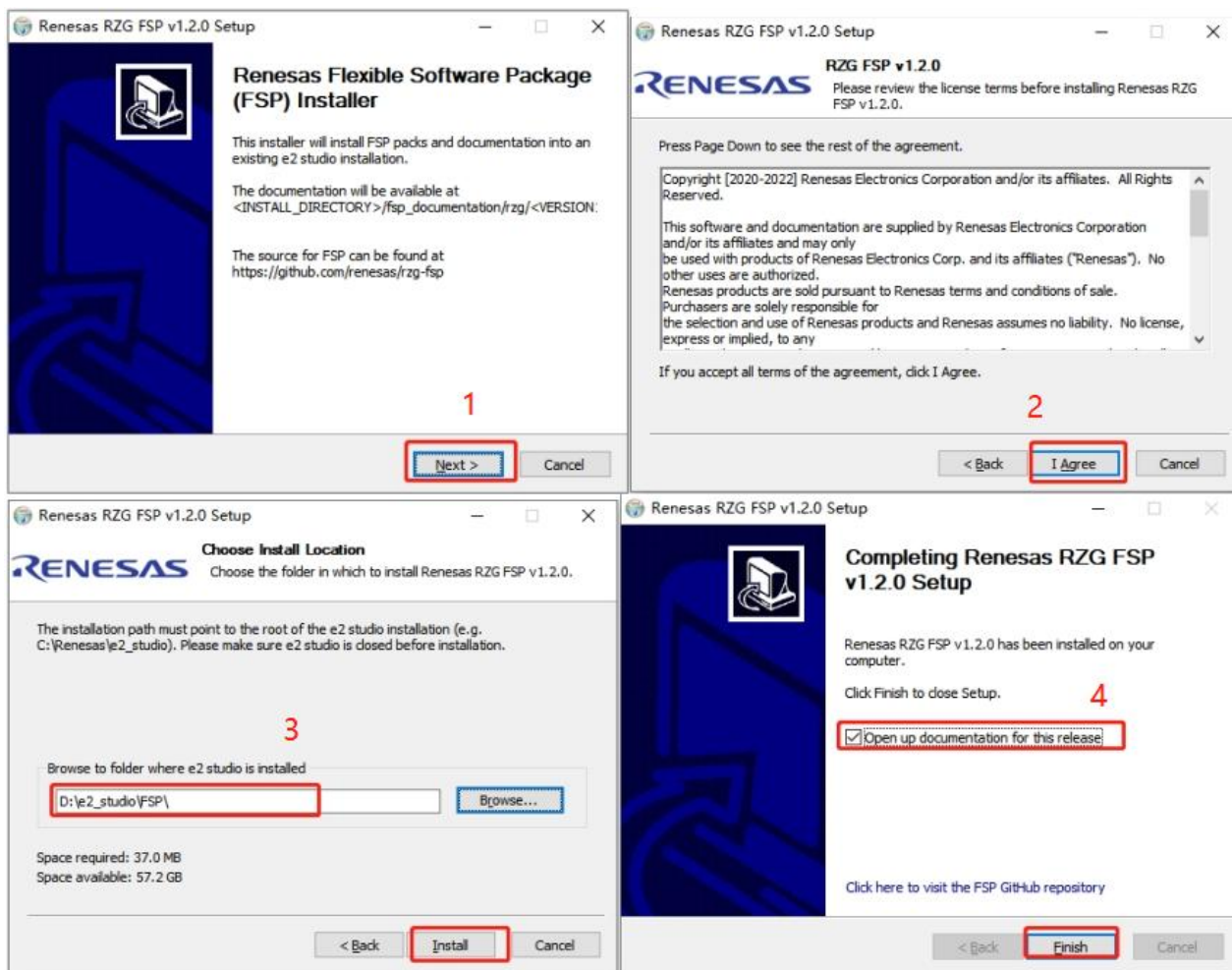


Figure 2-13. fsp installation

2.2.3. Heterogeneous communication

Decompress the rzg2l-cm33-multi-os-pgg. zip file in the M33.zip package in 04_sources and use e2 studio to import the rzg2l_cm33_rpmsg_demo project. Open e2 studio and select the directory where the new project will be created. Note that there is no Chinese path; otherwise, the project compilation may fail, as shown in Figure 2-14:

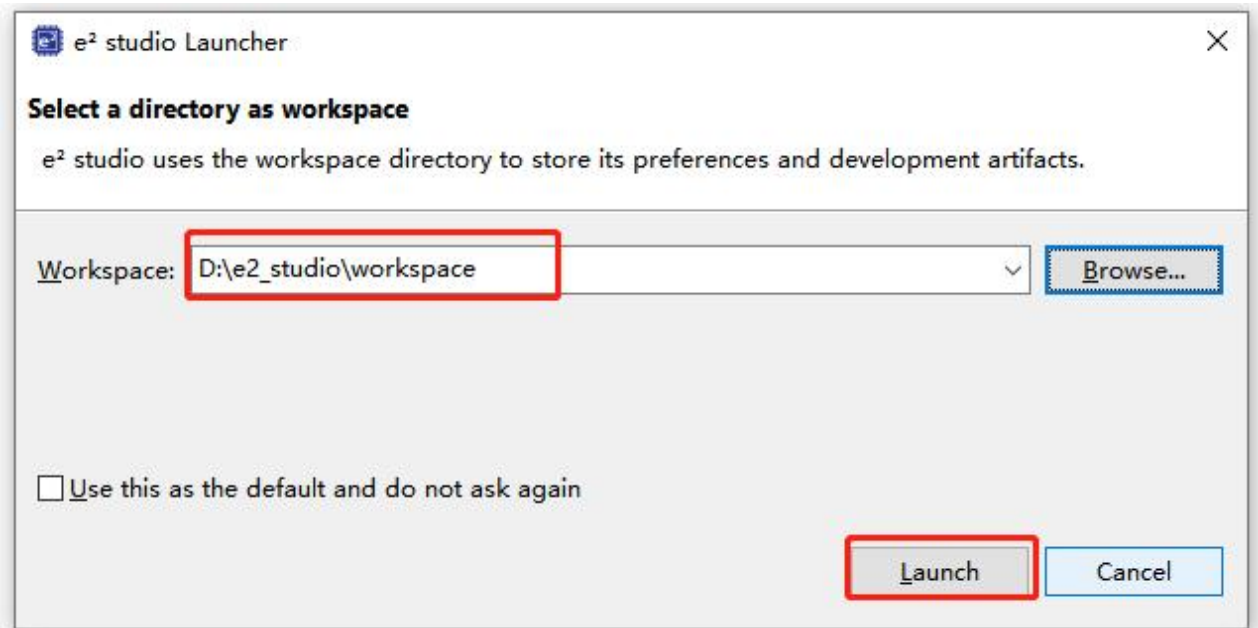


Figure 2-14. Engineering route selection

Import the project and click one by one[File]->[Import]->[General]->[Existing Projects into Workspace]->[Next],Select rzg2l_cm33_rpmsg_demo project,as shown in Figure:

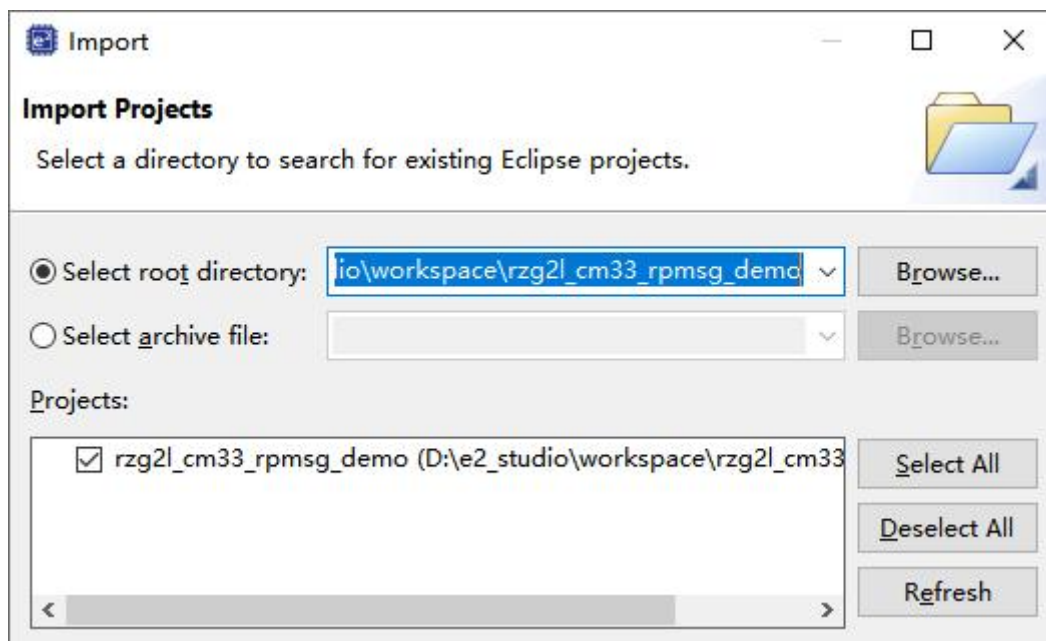


Figure 2-15. Project import

The import succeeds, for example, 2-16:

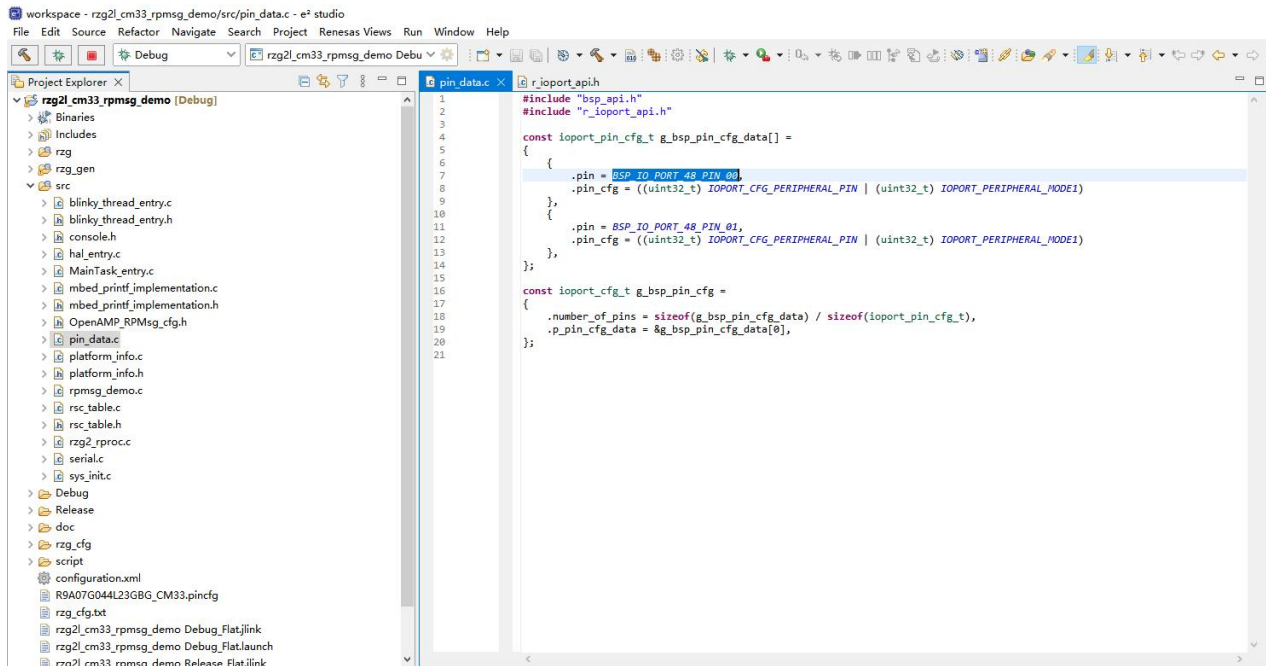


Figure 2-16. Project import success

Go to Project->Build Project, compile the project, compile successfully will generate a debug directory, which is stored in the compiled binary file, Copy the following files generated by debug to the sd card for CM33 Project call in uboot.

```
rzg2l_cm33_rpmsg_demo_secure_vector.bin
rzg2l_cm33_rpmsg_demo_secure_code.bin
rzg2l_cm33_rpmsg_demo_non_secure_vector.bin
rzg2l_cm33_rpmsg_demo_non_secure_code.bin
```

Insert the sd card into the sd card slot on the development board and perform the following call during the uboot phase.

```
=> switch_sdhi1 sdcard
=> ls mmc 1:1
```

```
System Volume Information/
400  rzg2l_cm33_rpmsg_demo_secure_code.bin
64  rzg2l_cm33_rpmsg_demo_secure_vector.bin
42160 rzg2l_cm33_rpmsg_demo_non_secure_code.bin
1984 rzg2l_cm33_rpmsg_demo_non_secure_vector.bin
0  LED_secure_code.bin
0  LED_secure_vector.bin
84  LED_non_secure_code.bin
0  LED_non_secure_vector.bin
```

```
8 file(s), 1 dir(s)
=> dcache off
=> mmc dev 1
switch to partitions #0, OK
mmc1 is current device
=> fatload mmc 1:1 0x0001FF80 rzg2l_cm33_rpmsg_demo_secure_vector.bin
64 bytes read in 12 ms (4.9 KiB/s)
=> fatload mmc 1:1 0x42EFF440 rzg2l_cm33_rpmsg_demo_secure_code.bin
400 bytes read in 13 ms (29.3 KiB/s)
=> fatload mmc 1:1 0x00010000 rzg2l_cm33_rpmsg_demo_non_secure_vector.b
in
1984 bytes read in 13 ms (148.4 KiB/s)
=> fatload mmc 1:1 0x40010000 rzg2l_cm33_rpmsg_demo_non_secure_code.bi
n
42160 bytes read in 18 ms (2.2 MiB/s)
=> cm33 start_debug 0x1001FF80 0x00010000
=> dcache on
=> run bootcmd
```

After the development board starts successfully and enters the root file system, then execute the sample program as follows:

```
root@myir-remi-1g:~# rpmsg_sample_client 1
metal: warning: metal_linux_irq_handling: Failed to set scheduler: -1.
Successfully probed IPI device
metal: info: metal_uio_dev_open: No IRQ for device 42f00000.rsctbl.
Successfully open uio device: 42f00000.rsctbl.
Successfully added memory device 42f00000.rsctbl.
metal: info: metal_uio_dev_open: No IRQ for device 43100000.vring-ctl1.
Successfully open uio device: 43100000.vring-ctl1.
Successfully added memory device 43100000.vring-ctl1.
metal: info: metal_uio_dev_open: No IRQ for device 43500000.vring-shm1.
Successfully open uio device: 43500000.vring-shm1.
Successfully added memory device 43500000.vring-shm1.
metal: info: metal_uio_dev_open: No IRQ for device 42f01000.mhu-shm.
```

```
Successfully open uio device: 42f01000.mhu-shm.
Successfully added memory device 42f01000.mhu-shm.
Initialize remoteproc successfully.
creating remoteproc virtio
initializing rpmsg shared buffer pool
initializing rpmsg vdev
  1 - Send data to remote core, retrieve the echo and validate its integrity ..
Remote proc init.
RPMSG endpoint has created.
RPMSG service has created.
sending payload number 0 of size 17
echo test: sent : 17
  received payload number 0 of size 17
.....
sending payload number 471 of size 488
echo test: sent : 488
  received payload number 471 of size 488
*****
Test Results: Error count = 0
*****
Quitting application .. Echo test end
Stopping application..
```

As you can see from the above sample program, after waiting for a communication channel to be established between CA55 and CM33, the CA55 sample program starts sending messages to CM33 and increases the size of the messages from a minimum of 17 to a maximum of 488. When CM33 receives the message sent by CA55, it sends it back to CA55.

2.3. Graphics

Graphics Processing Unit (GPU), also known as display core, vision processor, display chip, is a kind of microprocessor dedicated to do image and graphics-related computing work on personal computers, workstations, game consoles and some mobile devices (such as tablets, smart phones, etc.). As the core of the display system, graphics processing unit (GPU) has powerful data computing ability, and implements the functions of Two Dimension /Three Dimension (2D/3D) graphics processing, image processing and display control in the form of hardware accelerator.

Remi Pi core chip integrates a high performance 3D acceleration engine Mail-G31 GPU. As a 3D graphics engine (3DGE), the GPU supports 2D graphics, 3D graphics and GPGPU. OpenCL 2.0, OpenGL ES 1.1, 2.0, 3.0, 3.1, and 3.2 are supported. For detailed evaluation and testing, refer to Section 7.1, Graphics and Image Processing.

2.4. Memory

Remi Pi core chip provides an external SDRAM interface to support external memory up to 4 Gbytes, 16 bit DDR3L-1333 / DDR4-1600, the development board memory uses DDR4, the following is the test of 1GByte DDR memory.

2.4.1. Viewing memory information

The parameter information of memory in the system can be obtained by reading the “/proc/meminfo” file.

```
root@myir-remi-1g:/# cat /proc/meminfo
MemTotal:      876000 kB
MemFree:       279408 kB
MemAvailable:  450880 kB
Buffers:       6988 kB
Cached:        229348 kB
SwapCached:    0 kB
Active:        38796 kB
Inactive:      330740 kB. . .
```

- MemTotal : All available RAM size.
- MemFree : LowFree + HighFree.
- Buffers : The size used to cache the block device.
- Cached : The buffer size of the file.
- SwapCached : Memory that has been swapped out. Associated with I/O.
- Active : Frequently (recently) used memory.
- Inactive : Memory that has not been used much recently.

2.4.2. Get memory usage

You can run the free command to read the memory usage. The -m parameter indicates that the unit is MByte.

```
root@myir-remi-1g:/# free -m
              total          used          free      shared  buff/cache       available
```

Mem:	855	271	440	10	143
507					
Swap:	0	0	0		

- total : Total Memory.
- used : The amount of memory used.
- free : The amount of memory available.

2.4.3. Memory Stress Test

By giving the size and times of the test memory, the existing memory of the system can be tested on the pressure. The system tool memtester can be used for testing, such as specifying the memory size of 300MB, the number of tests is 1, and the test command is "memtester 300M 1".

The following is an example of a single test using 300MB of memory:

```
root@myir-remi-1g:/# memtester 300M 1
memtester version 4.3.0 (64-bit)
Copyright (C) 2001-2012 Charles Cazabon.
Licensed under the GNU General Public License version 2 (only).
pagesize is 4096
pagesizemask is 0xfffffffffff000
want 300MB (314572800 bytes)
got 300MB (314572800 bytes), trying mlock ...locked.
Loop 1/1:
  Stuck Address      : ok
  Random Value       : ok
  Compare XOR        : ok
  Compare SUB        : ok
  Compare MUL        : ok
  Compare DIV        : ok
  Compare OR         : ok
  Compare AND        : ok
  Sequential Increment: ok
  Solid Bits         : ok
  Block Sequential   : ok
  Checkerboard       : ok
```

Bit Spread : ok
Bit Flip : ok
Walking Ones : ok
Walking Zeroes : ok

Done.

2.5. eMMC

eMMC is a data storage device, including a MultiMediaCard (MMC) interface, a NAND Flash component. Its cost, small size, Flash technology independence, and high data throughput make it ideal for embedded products. Remi Pi is equipped with an eMMC with 8G capacity. This section will explain the steps and methods of viewing and operating eMMC under Linux system. The test methods of eMMC in this section are suitable for development boards equipped with eMMC memory.

2.5.1. Check eMMC Capacity

The eMMC partition information and capacity can be queried through the “fdisk -l” command as follows:

```
root@myir-remi-1g:/# fdisk -l
Disk /dev/mmcblk0: 7.29 GiB, 7818182656 bytes, 15269888 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x0c2032a9

Device            Boot  Start      End  Sectors  Size Id Type
/dev/mmcblk0p1          20480   122879   102400    50M  c W95 FAT32 (LBA)
/dev/mmcblk0p2       122880 15269887 15147008    7.2G 83 Linux
```

- /dev/mmcblk0p1 : Used to store kernel and dtb files
- /dev/mmcblk0p2 : Used to store the file system

Here /dev/mmcblk0p1 starts at 20480 blocks, with the bootloader and partition table information stored in front.

2.5.2. View the eMMC Partition Information

By “df” command, you can query eMMC partition information, usage, mount directory and other information.

```
root@myir-remi-1g:/# mount /dev/mmcblk0p1 /mnt/
```

```
root@myir-remi-1g:/# df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        6.8G  1.4G  5.2G  21% /
devtmpfs         235M    0  235M   0% /dev
tmpfs            429M    0  429M   0% /dev/shm
tmpfs            429M  9.8M  419M   3% /run
tmpfs            429M    0  429M   0% /sys/fs/cgroup
tmpfs            429M    0  429M   0% /tmp
tmpfs            429M   36K  428M   1% /var/volatile
tmpfs            86M     0   86M   0% /run/user/0
/dev/mmcblk0p1   50M   23M   27M  46% /mnt
```

- /dev/root : Root file system, mounted to the root directory.
- tmpfs: Memory virtual file systems mounted to different directories.
- devtmpfs : Used to create dev for the system.
- /dev/mmcblk0p1: It is used to store kernel and dtb files. If it is not mounted by default, you can manually mount it to view the files.

2.5.3. eMMC Performance Test

The performance test mainly tests the reading and writing speed of files by eMMC in Linux system. The test is generally combined with the dual command of "time" and "dd" .

● Write Performance

```
root@myir-remi-1g:/# time dd if=/dev/zero of=tempfile bs=1M count=100 co
nv=fdatasync
100+0 records in
100+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 4.12802 s, 25.4 MB/s
real    0m4.137s
user    0m0.001s
sys     0m0.585s
```

When you use the dd command to write a file, you need to add the "conv = fdatasync" parameter, which means that after using the dd command to write

multiple times, it will be synchronized to the disk(flush-cached). Because the disk is usually written to the cache first and then returned before it is written to the disk. Here we test that the write disk speed is 25.4MB/s.

- **Read Performance**

In embedded system, it is often necessary to test the system's performance of reading. If you want to test reading files directly from disk, you need to ignore the impact of cache. In this case, you can specify the parameter "iflag = direct, nonblock" .

```
root@myir-remi-1g:/# time dd if=tempfile of=/dev/null bs=1M count=100 iflag=direct,nonblock
100+0 records in
100+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 2.78449 s, 37.7 MB/s
real    0m2.789s
user    0m0.004s
sys     0m0.143s
```

The test results show that reading speed without cache is 37.7MB/s.

2.5.4. eMMC Life Test

emmc chip has a service life, and the number of emmc erasable times on this platform is about 3000. Too frequent erasable emmc may result in the end of the service life of emmc. It is recommended not to erase emmc too frequently and write a large amount of data.

- **View the emmc lifetime**

```
root@myir-remi-1g:~# mmc extcsd read /dev/mmcblk0 | grep LIFE
eMMC Life Time Estimation A [EXT_CSD_DEVICE_LIFE_TIME_EST_TYP_A]: 0x01
eMMC Life Time Estimation B [EXT_CSD_DEVICE_LIFE_TIME_EST_TYP_B]: 0x01
```

- **Type B standard**

Table 2-1 lists the type B standards.

Table 2-1. Estimated device life Type B Value

Value	Description
0x00	Not Defined
0x01	0%~10% device life time used
0x02	10%~20% device life time used
0x03	20%~30% device life time used
0x04	30%~40% device life time used
0x05	40%~50% device life time used
0x06	50%~60% device life time used
0x07	60%~70% device life time used
0x08	70%~80% device life time used
0x09	80%~90% device life time used
0x0A	90%~100% device life time used
0x0B	Exceeded its maximum estimated device life time
Others	Reserved

2.6. RTC

RTC (Real-time clock) itself is a clock that is used to record the Real time. When the software system is shut down, the system time is retained and the time is kept. Remi Pi uses an external RTC (RX8025T), and the test of RTC usually uses the `hwclock` and `date` commands commonly used in Linux system. The following test writes the system time to the RTC, reads the RTC time and sets it as the system time and carries out the test of time power-down maintenance.

- **View RTC Devices in Linux System**

```
root@myir-remi-1g:/# ls /dev/rt* -al
lrwxrwxrwx 1 root root      4 Nov 25  1929 /dev/rtc -> rtc0
crw----- 1 root root 252, 0 Nov 25  1929 /dev/rtc0
root@myir-remi-1g:~# cat /sys/class/rtc/rtc0/name
rtc-rx8025t 1-0032
```

`/dev/rtc0` is the default RTC device driver node for applications.

Set System Time

Set the system time to Mon Jan 16 17:53:30 UTC 2023:

```
root@myir-remi-1g:/# date 011617532023.30
Mon Jan 16 17:53:30 UTC 2023
```

- **Write System Time into RTC**

Write the system time set by the `date` command in the previous step to the RTC device:

```
root@myir-remi-1g:/# hwclock -w
```

- **Read Data from RTC and Set to System Time**

```
root@myir-remi-1g:/# hwclock -r
2023-01-16 17:55:19.544461+00:00
```

- **Keep RTC Time When Power is Off**

Shut down the development board and disconnect the power supply. After about 2 minutes, power on and start up again. View the RTC time and systemtime:

```
root@myir:~# hwclock -r
```

2023-01-16 17:57:19.544461+00:00

The above information indicates that the RTC time and system time viewed after rebooting have increased by about 2 minutes compared with the previous setting, indicating that the RTC is working normally. If the precision of the RTC needs to be tested in detail, the outage time can be extended by such as 24 hours to test the difference between the RTC time and the standard time.

- **Make the System Time and the RTC Time Synchronization**

```
root@myir-remi-1g:/# hwclock -s  
root@myir-remi-1g:/# date  
Mon Jan 16 17:58:17 UTC 2023
```

If you add the “hwclock -s” command to the startup script of Linux System, you can ensure that system time and RTC time are synchronized with each startup.

2.7. Watchdog

Linux kernel contains Watchdog subsystem. In the process of hardware design, the Watchdog function can be realized by using internal watchdog timer or external watchdog chip to monitor the operation of the system. The system will be able to automatically reset when the system fails to feed the dog. There are three watchdog dogs inside the Remi Pi chip. This chapter will explain how to test watchdog dogs under linux.

2.7.1. Stop the Watchdog

Write "V" to the watchdog node to try to disable the watchdog. By default, the watchdog is in use and cannot be disabled.

```
root@myir-remi-1g:/# echo V > /dev/watchdog0
-sh: /dev/watchdog0: Device or resource busy
```

2.7.2. Watchdog testing in user space

- **Modify the watchdog timeout**

Watchdog timeout configuration files in the directory/lib/systemd/system/watchdog. Service

```
[Unit]
Description=WatchdogTimer

[Service]
Type=simple
ExecStart=/usr/bin/watchdog-test -d -t 60 -e
ExecStop=/usr/bin/killall -s INT watchdog-test

[Install]
WantedBy=multi-user.target
```

- **Intentional kernel crash**

Simulate kernel crash, test watchdog reset function, default 60s system restart.

```
root@myir-remi-1g:/# echo c > /proc/sysrq-trigger
```

```
[ 190.855961] sysrq: Trigger a crash
[ 190.859368] Kernel panic - not syncing: sysrq triggered crash
[ 190.865090] CPU: 1 PID: 346 Comm: sh Not tainted 5.10.83-cip1-yocto-standard #1
[ 190.872360] Hardware name: Renesas SMARC EVK based on r9a07g044l2 (DT)
[ 190.878852] Call trace:
[ 190.881298] dump_backtrace+0x0/0x1b0
[ 190.884943] show_stack+0x18/0x68
[ 190.888244] dump_stack+0xd8/0x134
[ 190.891630] panic+0x174/0x33c
[ 190.894671] sysrq_handle_crash+0x1c/0x20
[ 190.898659] __handle_sysrq+0x8c/0x160
[ 190.902387] write_sysrq_trigger+0x94/0x138
[ 190.906551] proc_reg_write+0xac/0xf8
[ 190.910194] vfs_write+0xbc/0x390
[ 190.913490] ksys_write+0x68/0xf0
[ 190.916785] __arm64_sys_write+0x1c/0x28
[ 190.920689] el0_svc_common.constprop.4+0x8c/0x180
[ 190.925460] do_el0_svc+0x24/0x90
[ 190.928763] el0_svc+0x14/0x20
[ 190.931802] el0_sync_handler+0x90/0xb8
[ 190.935619] el0_sync+0x160/0x180
[ 190.938923] SMP: stopping secondary CPUs
[ 190.942837] Kernel Offset: disabled
[ 190.946309] CPU features: 0x0240026,2800a238
[ 190.950554] Memory Limit: none
[ 190.953601] ---[ end Kernel panic - not syncing: sysrq triggered crash ]. . .
```

2.7.3. Application test watchdog

The guard dog is one way. Either to systemd or to a third-party program. When you use an application to control the watchdog, you need to let the systemd process release the watchdog. You only need to stop the watchdog service.

- **Release the watchdog**

Use systemctl to stop the watchdog service

```
root@myir-remi-1g:/# systemctl stop watchdog.service
```

- **Set the Timeout Settings of the Watchdog**

To implement the timeout through IOCTL command:WDIOC_SETTIMEOUT, and you also need a parameter, that is:timeout. Examples of use:

```
ioctl(fd, WDIOC_SETTIMEOUT, &timeout);
```

The above is the reference code for setting the current timeout of the watchdog. "fd" is the file handle of the watchdog device.

- **Demo Application of the Watchdog**

Compile the watchdog file and copy it to the development board. Run the following command:

```
root@myir-remi-1g:/# watchdog
Usage: wdt_driver_test <timeout> <sleep> <test>
    timeout: value in seconds to cause wdt timeout/reset
    sleep: value in seconds to service the wdt
```

Run the watchdog app for a timeout of 4s and feed the dog every 1s:

```
root@myir-remi-1g:/# watchdog 4 1
Starting wdt_driver (timeout: 4, sleep: 1)
Trying to set timeout value=4 seconds
The actual timeout was set to 4 seconds
```

If the 1s above is changed to greater than 4s, the required 4s feeding time is exceeded, and the development board will restart. If you press Ctrl+C at this point to end the program, ending the program means you have stopped feeding the dog, and the watchdog timer has not stopped, which will cause the timer to overflow and a reset to restart.

2.8. EEPROM

EEPROM refers to charged erasable programmable read-only memory. It is a kind of storage chip that does not lose data after power failure. It can erase existing information and reprogram on a computer or special equipment. It is generally used in plug and play. The Remi Pi core module is equipped with a 32KB capacity EEPROM. This section will explain the steps and methods of viewing and operating EEPROM under Linux system.

- **View the eeprom model number**

```
root@myir-remi-1g:/# cat sys/bus/i2c/devices/3-0050/name
24c256
```

- **Read and write files to eeprom**

The low level indicates read/write, while the high level indicates read/write. Set the level of P4_0 to 0 before reading/writing.

```
root@myir-remi-1g:/# echo 152 > /sys/class/gpio/export
root@myir-remi-1g:/# echo out > /sys/class/gpio/P4_0/direction
root@myir-remi-1g:/# echo 0 > /sys/class/gpio/P4_0/value
root@myir-remi-1g:/# cat sys/class/gpio/P4_0/value
0
root@myir-remi-1g:/# echo test > /sys/bus/i2c/devices/3-0050/eeprom
root@myir-remi-1g:/# cat /sys/bus/i2c/devices/3-0050/eeprom
test
```

- **Power off for 2 minutes and then read data again**

```
root@myir-remi-1g:/# cat sys/bus/i2c/devices/3-0050/eeprom
test
```

- **Write protection**

If the value is set to high, the system enters the protected state and cannot write files. If the value is set to low, the system recovers.

```
root@myir-remi-1g:/# echo 1 > /sys/class/gpio/P4_0/value
root@myir-remi-1g:/# cat sys/class/gpio/P4_0/value
```

1

```
root@myir-remi-1g:/# echo hello > /sys/bus/i2c/devices/3-0050/eeprom
-sh: echo: write error: Connection timed out
root@myir-remi-1g:/# echo 0 > /sys/class/gpio/P4_0/value
root@myir-remi-1g:/# echo hello > /sys/bus/i2c/devices/3-0050/eeprom
root@myir-remi-1g:/# cat /sys/bus/i2c/devices/3-0050/eeprom
hello
```

- **eeeprom application test**

```
root@myir-remi-1g:/# eeprom_test -d /dev/i2c-3 -a 0x50 -s 0x10 -w test

WRITE:test
WRITE SUCCESS!
root@myir-remi-1g:/# eeprom_test -d /dev/i2c-3 -a 0x50 -s 0x10 -r 16

READ:test
TOTAL 16 BYTES.
root@myir-remi-1g:/# eeprom_test -d /dev/i2c-3 -a 0x50 -s 0x10 -e 16

ERASE:test
ERASE SUCCESS!
root@myir-remi-1g:/# eeprom_test -d /dev/i2c-3 -a 0x50 -s 0x10 -r 16

READ:
TOTAL 16 BYTES.
```

2.9. Power Management

This section demonstrates the Suspend functionality of Linux power management, which allows the board to sleep and wake up by external events. The Linux kernel generally provides three kinds of Suspend: Freeze, Standby, and STR(Suspend to RAM) are triggered by writing "freeze", "standby", and "mem" to the "/sys/power/state" file, respectively, in userspace. Remi Pi supports freeze and mem, but disk does not support it yet. The following will take mem as an example for testing.

2.9.1. View the supported modes

```
root@myir-remi-1g:/# cat /sys/power/state
freeze mem disk
```

2.9.2. Set the wake source

To test sleep wake up, set the wake up source first. Set the Debug serial port (ttySC0) as the wake up source.

```
root@myir-remi-1g:/# echo enabled > /sys/devices/platform/soc/.1004b800.
serial/tty/ttySC0/power/wakeup
```

2.9.3. Sleep to memory

Write a string to /sys/power/state in the user space to enter the corresponding power management mode. You can sleep the string to the memory in the following ways:

```
root@myir-remi-1g:/# echo mem > /sys/power/state
[ 861.411904] PM: suspend entry (s2idle)
[ 861.417399] Filesystems sync: 0.001 seconds
[ 861.422545] Freezing user space processes ... (elapsed 0.001 seconds) done.
[ 861.431486] OOM killer disabled.
[ 861.434697] Freezing remaining freezable tasks ... (elapsed 0.000 seconds)
done.
[ 861.443022] printk: Suspending console(s) (use no_console_suspend to debug)
```

2.9.4. Wake up using the debug serial port

After entering the sleep command, the development board will hibernate, save the running status data to the memory, close the peripheral, and enter the waiting mode. At this time, because the setting is debug wake up, the system can be successfully woken up by randomly typing characters on the keyboard, as shown below:

```
[ 861.477495] rzg2l-mipi-dsi 10860000.dsi: failed to stop HS clock
[ 861.579486] rcar-du 10890000.display: vertical blanking timeout
[ 862.280737] ravb 11c20000.ethernet eth0: Link is Down
[ 1020.824193] YT8531S Ethernet 11c20000.ethernet-ffffffff:04: attached PHY driver [YT8531S Ethernet] (mii_bus:phy_addr=11c20000.ethernet-ffffffff:04, irq=POLLD)
[ 1020.960206] YT8531S Ethernet 11c30000.ethernet-ffffffff:06: attached PHY driver [YT8531S Ethernet] (mii_bus:phy_addr=11c30000.ethernet-ffffffff:06, irq=POLLD)
[ 1021.132507] lt8912 0-0048: "1920x1080": 60 148500 1920 2008 2052 2200 1080 1116 1121 1125 0x8 0x0
[ 1021.619375] OOM killer enabled.
[ 1021.622509] Restarting tasks ... done.
[ 1021.643845] PM: suspend exit
root@myir-remi-1g:/#
```

3. Peripheral Interface

3.1. GPIO

GPIO is tested using the sysfs interface of the file system. The following uses P40_2 as an example to illustrate how to use GPIO. The GPIO pin of MYD-YG2LX is defined in the form PX_Y. Since PX_Y is converted to the pin numbering formula: $(X*8)+120+Y$, P40_2 is numbered: $(40*8)+120+2=442$.

3.1.1. Export GPIO to User Space

```
root@myir-remi-1g:/# echo 442 > /sys/class/gpio/export
root@myir-remi-1g:/# ls /sys/class/gpio/
P40_2 P4_0 export gpiochip120 unexport
```

The P40_2 directory is generated in /sys/class/gpio/.

3.1.2. Set/View the GPIO Direction

- Set as Input

```
root@myir-remi-1g:/# echo "in" > /sys/class/gpio/P40_2/direction
```

- Set as Output

```
root@myir-remi-1g:/# echo "out" > /sys/class/gpio/P40_2/direction
```

- View the Direction of GPIO

```
root@myir-remi-1g:/# cat /sys/class/gpio/P40_2/direction
out
```

Return "in" for input and "out" for output.

3.1.3. Set/View the GPIO Value

- Set Output Low

```
root@myir-remi-1g:/# echo 0 > /sys/class/gpio/P40_2/value
```

- Set Output High

```
root@myir-remi-1g:/# echo 1 > /sys/class/gpio/P40_2/value
```

- **View the GPIO Value**

```
root@myir-remi-1g:/# cat /sys/class/gpio/P40_2/value  
1
```

You can see that the P40_2 output is high. You can measure pin 11 of the J20 extension IO on the development board with a multimeter and see that the voltage is around 3.3V.

3.2. LED Lamp

The Linux system provides a separate subsystem to facilitate the operation of LED devices from user space. The subsystem provides an interface for LED devices in the form of files. These interfaces are located in the /sys/class/leds directory. In the list of hardware resources, we have listed all the LEDs on the device. Let's test a LED by reading and writing sysfs from the command. The following commands are generic commands and are general methods of controlling LEDs .

3.2.1. View the LED0

```
root@myir-remi-1g:/# ls sys/class/leds/  
LED0
```

3.2.2. Heartbeat indicator LED0 is used as an example

- **Read the LED0 status. 0 indicates that LED0 is disabled, and 1 to 255 indicates that LED0 is enabled**

```
root@myir-remi-1g:/# cat sys/class/leds/LED0/brightness  
0
```

- **Turn off the LED0 heartbeat indicator**

```
root@myir-remi-1g:/# echo "none" > /sys/class/leds/LED0/trigger
```

- **Light up LED0**

```
root@myir-remi-1g:/# echo 1 > sys/class/leds/LED0/brightness
```

- **Extinguish LED0**

```
root@myir-remi-1g:/# echo 0 > sys/class/leds/LED0/brightness
```

- **Enable the LED0 heartbeat indicator**

```
root@myir-remi-1g:/# echo "heartbeat" > /sys/class/leds/LED0/trigger
```

3.3. Key

Linux's /dev/input/eventx device can be used to easily debug input devices such as the mouse, keyboard, and trackpad. This section is devoted to testing key. Use the evtest command to see if the keystrokes are responsive. Remi Pi has two keys, Reset is the system reset button and User is the user button.

3.3.1. Test Keys

- **View Information of the Input Device Event**

```
root@myir-remi-1g:/# evtest
No device specified, trying to scan all of /dev/input/event*
Available devices:
/dev/input/event0:      keyboard
Select the device event number [0-0]:
```

The device event of the keyboard is event0.

- **evtest Tests key information**

Run the following command and press User. The following information is displayed on the serial port terminal.

```
root@myir-remi-1g:/# evtest
No device specified, trying to scan all of /dev/input/event*
Available devices:
/dev/input/event0:      keyboard
Select the device event number [0-0]: 0
Input driver version is 1.0.1
Input device ID: bus 0x19 vendor 0x1 product 0x1 version 0x100
Input device name: "keyboard"
Supported events:
  Event type 0 (EV_SYN)
  Event type 1 (EV_KEY)
    Event code 2 (KEY_1)
Properties:
Testing ... (interrupt to exit)
```

```
Event: time 3029547809.3029547809, type 1 (EV_KEY), code 2 (KEY_1), value 1
Event: time 3029547809.3029547809, ----- SYN_REPORT -----
Event: time 3029547809.3029547809, type 1 (EV_KEY), code 2 (KEY_1), value 0
Event: time 3029547809.3029547809, ----- SYN_REPORT -----
Event: time 3029547810.3029547810, type 1 (EV_KEY), code 2 (KEY_1), value 1
Event: time 3029547810.3029547810, ----- SYN_REPORT -----
Event: time 3029547810.3029547810, type 1 (EV_KEY), code 2 (KEY_1), value 0
Event: time 3029547810.3029547810, ----- SYN_REPORT -----
Event: time 3029547810.3029547810, type 1 (EV_KEY), code 2 (KEY_1), value 1
Event: time 3029547810.3029547810, ----- SYN_REPORT -----
Event: time 3029547810.3029547810, type 1 (EV_KEY), code 2 (KEY_1), value 0
Event: time 3029547810.3029547810, ----- SYN_REPORT -----
```

Every time you press User, the current terminal will print out the current event code value, that is, the key is normal.

3.4. USB HOST

Remi Pi has two USB2.0 interfaces, one is TYPE-C interface, and the other USB2.0 is converted into 2-way USB interface through HUB expansion chip. This section verifies the feasibility of USB Host driver through relevant commands or hotplug and USB HUB. Realize the function of reading and writing U disk, usb enumeration function.

3.4.1. View the printed information about the usb

- View USB device information

Connect the USB flash drive to the USB Host port (J4) on the development board. The following message is displayed:

```
root@myir-remi-1g:/# [ 2647.135771] usb 1-1.1: new high-speed USB device n
umber 3 using ehci-platform
[ 2647.370403] usb-storage 1-1.1:1.0: USB Mass Storage device detected
[ 2647.377480] scsi host0: usb-storage 1-1.1:1.0
[ 2648.392821] scsi 0:0:0:0: Direct-Access      VendorCo ProductCode      2.00
PQ: 0 ANSI: 4
[ 2648.403301] sd 0:0:0:0: [sda] 61440000 512-byte logical blocks: (31.5 GB/29.
3 GiB)
[ 2648.415925] sd 0:0:0:0: [sda] Write Protect is off
[ 2648.421417] sd 0:0:0:0: [sda] No Caching mode page found
[ 2648.427856] sd 0:0:0:0: [sda] Assuming drive cache: write through
[ 2648.453037] sda: sda1
[ 2648.460076] sd 0:0:0:0: [sda] Attached SCSI removable disk
[ 2660.727380] usb 1-1.1: USB disconnect, device number 3
[ 2665.215901] usb 1-1: reset high-speed USB device number 2 using ehci-pla
[ 2665.795769] usb 1-1.2: new high-speed USB device number 4 using ehci-pl
atform
[ 2666.034208] usb-storage 1-1.2:1.0: USB Mass Storage device detected
[ 2666.041159] scsi host0: usb-storage 1-1.2:1.0
[ 2667.080626] scsi 0:0:0:0: Direct-Access      VendorCo ProductCode      2.00
PQ: 0 ANSI: 4
```

```
[ 2667.090121] sd 0:0:0:0: [sda] 61440000 512-byte logical blocks: (31.5 GB/29.3 GiB)
[ 2667.102755] sd 0:0:0:0: [sda] Write Protect is off
[ 2667.109859] sd 0:0:0:0: [sda] No Caching mode page found
[ 2667.117653] sd 0:0:0:0: [sda] Assuming drive cache: write through
[ 2667.141085] sda: sda1
[ 2667.148123] sd 0:0:0:0: [sda] Attached SCSI removable disk
[ 2669.104176] usb 1-1.2: USB disconnect, device number 4
```

Based on the preceding information, you can see that the device to be mounted is sda1.

3.4.2. The USB flash drive is mounted with read and write

- **Mount the USB flash drive**

```
root@myir-remi-1g:/# mount /dev/sda1 /mnt/
```

- **Read**

You need to create a test.txt file on the USB flash drive in advance.

```
root@myir-remi-1g:/# ls /mnt/
1.txt  System Volume Information
root@myir-remi-1g:/# cat /mnt/1.txt
helloworld!
```

- **Write**

```
root@myir-remi-1g:/# touch /mnt/2.txt
root@myir-remi-1g:/# echo "hello" > /mnt/2.txt
root@myir-remi-1g:/# cat /mnt/2.txt
hello
```

After the file is written, run the sync command to ensure that all data is written into the USB flash drive before uninstalling the USB flash drive.

3.4.3. Uninstall USB flash drive

- **Unloading operation**

```
root@myir-remi-1g:/# umount /mnt
```

3.5. USB OTG

Remi Pi has a USB OTG interface, and an OTG adapter with TYPE-C interface should be prepared before OTG mount test. This chapter enumerates different usb peripherals to test OTG interface.

3.5.1. Enumerate different peripherals

```
root@myir-remi-1g:/# [ 566.532399] usb 1-1.1: USB disconnect, device number 3
[ 680.223969] usb 2-1: new full-speed USB device number 2 using ohci-platform
[ 680.463023] usb 2-1: not running at top speed; connect to a high speed hub
[ 680.490384] usb-storage 2-1:1.0: USB Mass Storage device detected
[ 680.497187] scsi host0: usb-storage 2-1:1.0
[ 681.521096] scsi 0:0:0:0: Direct-Access Generic MassStorageClass 1536 PQ: 0 ANSI: 6
[ 681.804064] sd 0:0:0:0: [sda] 30703616 512-byte logical blocks: (15.7 GB/14.6 GiB)
[ 681.823096] sd 0:0:0:0: [sda] Write Protect is off
[ 681.840100] sd 0:0:0:0: [sda] Write cache: disabled, read cache: enabled, doesn't support DPO or FUA
[ 681.931072] sda: sda1
[ 1218.292142] input: PixArt Dell MS116 USB Optical Mouse as /devices/platform/soc/11c50000.usb/usb2/2-1/2-1:1.0/0003:413C:301A.0001/input/input2
[ 1218.305223] hid-generic 0003:413C:301A.0001: input: USB HID v1.11 Mouse [PixArt Dell MS116 USB Optical Mouse] on usb-11c50000.usb-1/input0
[ 1218.348897] mousedev: PS/2 mouse device common for all mice
[ 1237.677020] usb 2-1: USB disconnect, device number 3
```

3.5.2. Read and Write functions

After the usb peripheral is mounted using the otg port, see 3.4 for read and write operations.

3.6. Micro SD card

Micro SD card, formerly known as trans flash card (TF Card), is a very small flash memory card. Compared with the standard SD card, micro SD card is smaller in shape, and it is the smallest SD card in the type of SD card. Although the size and interface shape of micro SD card are different from the original SD card, the interface specification remains unchanged to ensure compatibility. If the micro SD card is inserted into a specific adapter card, it can be used as a standard SD card. The SD card has become the most widely used memory card in consumer digital devices. It has the characteristics of large capacity, high performance and security. There are usually 9 pins on the back of micro SD card, including 4 data lines, supporting 1 bit / 4 bit data transmission width.

Remi Pi supports two SDHI interfaces. The Remi Pi development board uses SDHI1 to connect to the Micro SD card.

3.6.1. Check TF Card Capacity

The TF Card partition information and capacity can be queried through the "fdisk -l" command as follows:

```
root@myir-remi-1g:/# fdisk -l
.....
Disk /dev/mmcblk1: 14.66 GiB, 15720251392 bytes, 30703616 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x00000000

Device            Boot Start      End  Sectors  Size Id Type
/dev/mmcblk1p1  *           63 30703615 30703553 14.7G  c W95 FAT32 (LBA)
➤ /dev/mmcblk1p1: Partition of an sd card
```

3.6.2. TF card Performance Test

The performance test mainly tests the file reading and writing speed of an SD card in linux. The test is usually performed using the time and dd commands. Mount the TF card partition to be tested. The last partition /dev/mmcblk1p1 is used as an example. The directory to be mounted is /mnt/.

- **Write Performance**

```
root@myir-remi-1g:/# time dd if=/dev/zero of=tempfile bs=1M count=100 co
nv=fdatasync
100+0 records in
100+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 3.98196 s, 26.3 MB/s

real    0m3.991s
user    0m0.001s
sys     0m0.573s
root@myir-remi-1g:/# time dd if=tempfile of=/mnt/1.txt bs=1M count=100 co
nv=fdatasync
100+0 records in
100+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 7.77278 s, 13.5 MB/s

real    0m7.778s
user    0m0.001s
sys     0m0.914s
```

When you use the dd command to write files, you need to add conv=fdatasync, which means that after the dd command is written for N times, flush cache is synchronized to disks. Because writes to disk are usually written to cache first and then returned before they're written to disk. The disk write speed tested here is 13.5MB/s.

- **Read Performance**

Ignore cache effects when reading files. It is possible to specify the parameter iflag=direct, nonblock.

```
root@myir-remi-1g:/# time dd if=/mnt/1.txt of=/dev/null bs=1M count=100 if
lag=direct,nonblock
100+0 records in
100+0 records out
104857600 bytes (105 MB, 100 MiB) copied, 6.63845 s, 15.8 MB/s

real    0m6.643s
user    0m0.001s
sys     0m0.065s
```

The data read speed from the SD card is 15.8MB/s.

3.7. Display

Remi Pi adopts LT8912 chip to convert into 1 LVDS output and 1 HDMI output for output display.

3.7.1. LVDS display test

Remi Pi ADAPTS to the LVDS capacitive screen of Mill, model MY-LVDS070C. This screen is a 7-inch capacitive touch screen and a color active matrix thin film transistor (TFT) liquid crystal display (LCD). The screen is first connected to the J6 LVDS interface of the bottom plate through a 40pin facet connection cable. Start the development board, enter the /dev/dri/directory, ls -l can see the interface to operate the gpu, send commands and other operations are carried out through this device node.

- **Gets the current connection status**

```
root@myir-remi-1g:/# ls -l /dev/dri/
root@myir-remi-1g:/# cat /sys/kernel/debug/dri/0/state
plane[32]: plane-0
    crtc=crtc-0
    fb=46
        allocated by = [fbcon]
        refcount=2
        format=XR24 little-endian (0x34325258)
        modifier=0x0
        size=800x480
        layers:
            size[0]=800x480
            pitch[0]=3200
            offset[0]=0
            obj[0]:
                name=0
                refcount=3
                start=00100000
```

```

size=1536000
imported=no
paddr=0x0000000058200000
vaddr=0000000033b34ab8

crtc-pos=800x480+0+0
src-pos=800.000000x480.000000+0.000000+0.000000
rotation=1
normalized-zpos=0
color-encoding=ITU-R BT.601 YCbCr
color-range=YCbCr limited range

```

- **Close the current HMI application and run `./framebuffer_test`**

```

root@myir-remi-1g:/# killall mxapp2
root@myir-remi-1g:/# ./framebuffer_test
The framebuffer device was opened successfully.
vinfo.xres=800
vinfo.yres=480
vinfo.bits_per_bits=32
vinfo.xoffset=0
vinfo.yoffset=0
red.offset=16
green.offset=8
finfo.line_length=3200
finfo.type = PACKED_PIXELS
The framebuffer device was mapped to memory successfully.
color: red   rgb_val: 00FF0000
color: green rgb_val: 0000FF00
color: blue  rgb_val: 000000FF
color: r & g  rgb_val: 00FFFF00
color: g & b  rgb_val: 0000FFFF
color: r & b  rgb_val: 00FF00FF
color: white rgb_val: 00FFFFFF

```

You can see the color change on the screen.

3.8. Touch Panel

Touch includes capacitive touch and resistive touch. Remi Pi series development board hardware does not support resistive touch at present, but supports capacitive touch. MYIR Technology provides 7 "touch LCD accessories, see Table 1-1. Accessories can be purchased according to actual needs. Capacitive screen in use is more sensitive, few problems. In addition, the capacitor screen does not need to be accurate. Because according to the principle of the capacitor screen, the capacitor screen can accurately identify the position of the finger contact with the screen in use, with high sensitivity. If we click on the software in use, there is generally only one situation: there is a problem on the screen. Here's a simple test to test the capacitive screen's touch capabilities using the evtest command.

3.8.1. Touch screen connection

Connect the MY-LVDS070C touch LCD to the J6 interface of the development board in accordance with Section 3.7.1. Note that the connection should be made with the 40pin pin of the different plane line.

3.8.2. Touch test with evtest command

The terminal executes "evtest" to enter the test interface. Select the test peripheral as the touch screen, where the default is input interrupt 0. Select "0" in the test interface and press enter to start the test:

```
root@myir-remi-1g:~# evtest
No device specified, trying to scan all of /dev/input/event*
Available devices:
/dev/input/event0:      generic ft5x06 (79)
/dev/input/event1:      keyboard
Select the device event number [0-1]: 0
Input driver version is 1.0.1
Input device ID: bus 0x18 vendor 0x0 product 0x0 version 0x0
Input device name: "generic ft5x06 (79)"
Supported events:
  Event type 0 (EV_SYN)
```

Event type 1 (EV_KEY)

Event code 330 (BTN_TOUCH)

Event type 3 (EV_ABS)

Event code 0 (ABS_X)

Value 0

Min 0

Max 799

Event code 1 (ABS_Y)

Value 0

Min 0

Max 479

Event code 47 (ABS_MT_SLOT)

Value 0

Min 0

Max 4

Event code 53 (ABS_MT_POSITION_X)

Value 0

Min 0

Max 799

Event code 54 (ABS_MT_POSITION_Y)

Value 0

Min 0

Max 479

Event code 57 (ABS_MT_TRACKING_ID)

Value 0

Min 0

Max 65535

Properties:

Property type 1 (INPUT_PROP_DIRECT)

Testing ... (interrupt to exit)

Event: time 3029529688.3029529688, type 3 (EV_ABS), code 57 (ABS_MT_TRACKING_ID), value 0

```
Event: time 3029529688.3029529688, type 3 (EV_ABS), code 53 (ABS_MT_POSITION_X), value 716
Event: time 3029529688.3029529688, type 3 (EV_ABS), code 54 (ABS_MT_POSITION_Y), value 225
Event: time 3029529688.3029529688, type 1 (EV_KEY), code 330 (BTN_TOUCH), value 1
Event: time 3029529688.3029529688, type 3 (EV_ABS), code 0 (ABS_X), value 716
Event: time 3029529688.3029529688, type 3 (EV_ABS), code 1 (ABS_Y), value 225
Event: time 3029529688.3029529688, ----- SYN_REPORT -----
Event: time 3029529688.3029529688, type 3 (EV_ABS), code 53 (ABS_MT_POSITION_X), value 717
Event: time 3029529688.3029529688, type 3 (EV_ABS), code 1 (ABS_Y), value 226
Event: time 3029529688.3029529688, ----- SYN_REPORT -----
Event: time 3029529688.3029529688, type 3 (EV_ABS), code 53 (ABS_MT_POSITION_X), value 713
. . .
```

As can be seen from the above, coordinate values and key values are mainly displayed. Specific information is as follows:

- EV_SYN: Synchronous event.
- EV_KEY: Key events, such as BTN_TOUCH, indicate touch keys.
- EV_ABS: Absolute coordinates.
- BTN_TOUCH: Touch button.
- ABS_MT_SLOT: The corresponding value in the event.
- ABS_MT_TRACKING_ID: Indicates the start of information collection.
- ABS_X: Relative to the screen's absolute coordinate X.
- ABS_Y: Relative to the screen's absolute coordinate Y.
- ABS_MT_POSITION_X: The x coordinate position of the center point of the screen contact surface.
- ABS_MT_POSITION_Y: The Y coordinate position of the center point of the screen contact surface.

3.9. Backlight

This routine is mainly to test the brightness of the backlight. Taking the lvds screen as an example, we can query and adjust the brightness of the backlight by operating the corresponding file of the LVDS screen in the /sys directory.

3.9.1. View the current backlight level

```
root@myir-remi-1g:/# cd /sys/class/backlight/backlight
root@myir-remi-1g:/sys/class/backlight/backlight# ls
actual_brightness  bl_power  brightness  device  max_brightness  power  scale  subsystem  type  uevent
root@myir-remi-1g:/sys/class/backlight/backlight# cat brightness
2
```

3.9.2. View the maximum backlight level

```
root@myir-remi-1g:/sys/class/backlight/backlight# cat max_brightness
7
```

The maximum backlight level is 7 and ranges from 0 to 7.

3.9.3. Backlight adjustment

If you need to change the backlight level, use the echo command to write the brightness level to the Brightness Settings, but did not exceed the maximum.

- **Turn off the backlight**

Writing 0 to the brightness parameter will turn off the backlight:

```
root@myir-remi-1g:/sys/class/backlight/backlight# echo 0 > brightness
```

- **Turn on the backlight**

Writing 7 to the brightness parameter will turn on the backlight:

```
root@myir-remi-1g:/sys/class/backlight/backlight# echo 7 > brightness
```

3.10. MIPI-CSI

Remi Pi development board has an independent MIPI-CSI interface, which is suitable for MY-CAM003M camera with OV5640 lens and supports up to 500W pixels. This camera is used here for test instructions. Note that the connection should use the same plane line of 24pin and connect the J2 interface of the development board..

3.10.1. i2cdetect Check whether the driver is loaded

On the mainboard schematic diagram, you can find the I2C bus to which the device is mounted. You can find the corresponding address in the corresponding I2C schematic diagram or device tree file. The address of OV5640 is 0x3C.

```
root@myir-remi-1g:~# i2cdetect -y -r 0
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
10:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
20:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
30:  -- -- -- -- -- -- -- -- -- -- UU UU -- --
40:  -- -- -- -- -- -- -- UU UU UU 4b -- -- -- --
50:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
60:  -- -- -- -- -- -- -- -- -- -- -- -- -- --
70:  -- -- -- -- -- -- -- --
```

3.10.2. Viewing device nodes

```
root@myir-remi-1g:~# ls /dev/video0
/dev/video0
```

3.10.3. View the formats supported by the device

```
root@myir-remi-1g:~# v4l2-ctl -D -d /dev/video0 --list-formats-ext
Driver Info:
    Driver name      : rzg2l_cru
    Card type        : RZG2L_CRU
    Bus info         : platform:10830000.video
```

```
Driver version   : 5.10.83
Capabilities     : 0xa5200001
                  Video Capture
                  Read/Write
                  Streaming
                  Extended Pix Format
                  Device Capabilities
Device Caps      : 0x25200001
                  Video Capture
                  Read/Write
                  Streaming
                  Extended Pix Format
```

Media Driver Info:

```
Driver name      : rzg2l_cru
Model           : renesas,cru-r9a07g044
Serial          :
Bus info        : platform:10830000.video
Media version    : 5.10.83
Hardware revision: 0x00000000 (0)
Driver version   : 5.10.83
```

Interface Info:

```
ID              : 0x03000011
Type            : V4L Video
```

Entity Info:

```
ID              : 0x0000000f (15)
Name            : CRU output
Function        : V4L2 I/O
Pad 0x01000010 : 0: Sink
```

```
Link 0x02000013: from remote pad 0x10000003 of entity 'rzg2l_csi2 1
0830400.csi2': Data
```

```
Link 0x02000015: from remote pad 0x10000004 of entity 'rzg2l_csi2 1
0830400.csi2': Data
```

Link 0x02000017: from remote pad 0x1000005 of entity 'rzg2l_csi2 10830400.csi2': Data

Link 0x02000019: from remote pad 0x1000006 of entity 'rzg2l_csi2 10830400.csi2': Data

ioctl: VIDIOC_ENUM_FMT

Type: Video Capture

- [0]: 'NV16' (Y/CbCr 4:2:2)
- [1]: 'GREY' (8-bit Greyscale)
- [2]: 'YUYV' (YUYV 4:2:2)
- [3]: 'UYVY' (UYVY 4:2:2)
- [4]: 'BGR3' (24-bit BGR 8-8-8)
- [5]: 'XR24' (32-bit BGRX 8-8-8-8)
- [6]: 'AR24' (32-bit BGRA 8-8-8-8)
- [7]: 'BA24' (32-bit ARGB 8-8-8-8)
- [8]: 'RGGG' (8-bit Bayer RGRG/GBGB)
- [9]: 'BA81' (8-bit Bayer BGBG/GRGR)
- [10]: 'GRBG' (8-bit Bayer GRGR/BGBG)
- [11]: 'GBRG' (8-bit Bayer GBGB/RGRG)
- [12]: 'RG10' (10-bit Bayer RGRG/GBGB)
- [13]: 'BG10' (10-bit Bayer BGBG/GRGR)
- [14]: 'BA10' (10-bit Bayer GRGR/BGBG)
- [15]: 'GB10' (10-bit Bayer GBGB/RGRG)
- [16]: 'RG12' (12-bit Bayer RGRG/GBGB)
- [17]: 'BG12' (12-bit Bayer BGBG/GRGR)
- [18]: 'BA12' (12-bit Bayer GRGR/BGBG)
- [19]: 'GB12' (12-bit Bayer GBGB/RGRG)
- [20]: 'pREE' (14-bit Bayer RGRG/GBGB Packed)
- [21]: 'pBEE' (14-bit Bayer BGBG/GRGR Packed)
- [22]: 'pgEE' (14-bit Bayer GRGR/BGBG Packed)
- [23]: 'pGEE' (14-bit Bayer GBGB/RGRG Packed)
- [24]: 'RG16' (16-bit Bayer RGRG/GBGB)
- [25]: 'BYR2' (16-bit Bayer BGBG/GRGR)

[26]: 'GR16' (16-bit Bayer GRGR/BGBG)

[27]: 'GB16' (16-bit Bayer GBGB/RGRG)

3.10.4. preview

Run the v4l2-init.sh script to set and enable weston, as follows:

```
root@myir-remi-1g:~# vi v4l2-init.sh
root@myir-remi-1g:~# cat v4l2-init.sh
#!/bin/sh

cru=$(cat /sys/class/video4linux/video*/name | grep "CRU")
csi2=$(cat /sys/class/video4linux/v4l-subdev*/name | grep "csi2")

# Available resolution of OV5645.
# Please choose one of a following resolution then comment out the rest.
#ov5645_res=1280x960
ov5640_res=640x480
#ov5645_res=2592x1944

if [ -z "$cru" ]
then
    echo "No CRU video device founds"
else
    media-ctl -d /dev/media0 -r
    if [ -z "$csi2" ]
    then
        echo "No MIPI CSI2 sub video device founds"
    else
        media-ctl -d /dev/media0 -l "'rzg2l_csi2 10830400.csi2':1 -> 'CRU output':0 [1]"
        media-ctl -d /dev/media0 -V "'rzg2l_csi2 10830400.csi2':1 [fmt:UYVY8_2X8/$ov5640_res field:none]"
        media-ctl -d /dev/media0 -V "'ov5640 0-003c':0 [fmt:UYVY8_2X8/$ov5640_res field:none]"
    fi
fi
```

```
echo "Link CRU/CSI2 to ov5640 0-003c with format UYVY8_2X
8 and resolution $ov5640_res"
fi
fi
```

Execute the script and start weston to preview as follows:

```
root@myir-remi-1g:~# ./v4l2-init.sh
Link CRU/CSI2 to ov5640 0-003c with format UYVY8_2X8 and resolution 640x
480
root@myir-remi-1g:~# weston-start
root@myir-remi-1g:~# gst-launch-1.0 v4l2src device=/dev/video0 ! video/x-ra
w,format=YUY2,width=640,height=480 ! waylandsink
Setting pipeline to PAUSED ...
Pipeline is live and does not need PREROLL ...
Setting pipeline to PLAYING ...
New clock: GstSystemClock
```

At this time, the camera interface is displayed. The detailed information is as follows:

- device: Is a specified device node
- video/x-raw: Display video format
- width: Display width
- height: Display height
- waylandsink: Display terminal

3.10.5. Take pictures

```
root@myir-remi-1g:~# gst-launch-1.0 v4l2src device=/dev/video0 num-buffers
=3 ! jpegenc ! filesink location=/home/root/test112.jpg
Setting pipeline to PAUSED ...
Pipeline is live and does not need PREROLL ...
Setting pipeline to PLAYING ...
New clock: GstSystemClock
Got EOS from element "pipeline0".
Execution ended after 0:00:00.515239250
Total time: 0.515244 seconds
```

Freeing pipeline ...

4. Network interface

The Remi Pi development board contains two gigabit Ethernet interfaces and one Wi-Fi interface. The following will briefly introduce and test the configuration of these network devices.

4.1. Ethernet

There are many network configuration tools in Linux, such as net-tools, iproute2, systemd-networkd, network manager, connman, etc., which can be selected according to actual needs during system customization. This section describes several common methods of manual temporary configuration and automatic permanent configuration on Ethernet.

4.1.1. Manually configure a temporary static IP address.

- Use ifconfig to Configure the Network Manually

Firstly, check the network device information through ifconfig command .

```
root@myir-remi-1g:~# ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500  metric 1
    inet 192.168.1.230  netmask 255.255.255.0  broadcast 192.168.1.255
    ether ae:30:dc:eb:e1:05  txqueuelen 1000  (Ethernet)
    RX packets 14909  bytes 2065857 (1.9 MiB)
    RX errors 0  dropped 2883  overruns 0  frame 0
    TX packets 88  bytes 8062 (7.8 KiB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
    device interrupt 180
```

The following describes how to manually configure IP address 192.168.1.100 for eth0:

```
# ifconfig eth0 192.168.1.100 netmask 255.255.255.0 up
```

The above command is to manually set the IP address of eth0 to 192.168.1.100, subnet mask to 255.255.255.0, and the default broadcast address to 192.168.1.255, and use the up parameter to activate them, as shown below:

```
root@myir-remi-1g:~# ifconfig eth0 192.168.1.100 netmask 255.255.255.0 up
root@myir-remi-1g:~# ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500 metric 1
    inet 192.168.1.100 netmask 255.255.255.0 broadcast 192.168.1.255
    ether ae:30:dc:eb:e1:05 txqueuelen 1000 (Ethernet)
    RX packets 3191 bytes 475107 (463.9 KiB)
    RX errors 0 dropped 687 overruns 0 frame 0
    TX packets 66 bytes 6000 (5.8 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
    device interrupt 180
```

- **Configure the Network Manually by iproute2**

The ifconfig command that sets the IP address manually can also be substituted using “ip addr” and “ip link” , for more information about iproute2, please refer to <https://wiki.linuxfoundation.org/networking/iproute2> for detail.

```
root@myir-remi-1g:~# ip addr flush dev eth0
root@myir-remi-1g:~# ip addr add 192.168.1.101/24 brd + dev eth0
root@myir-remi-1g:~# ip link set eth0
```

If an IP address has been previously configured, the IP address configured using “ip addr add” will become the secondary address, so use “ip addr flush” to clear the previous address before configuring and activating it. Once configured, view the eth0 information from the IP Addr Show command as follows:

```
root@myir-remi-1g:~# ip addr show eth0
3: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state U
P group default qlen 1000
    link/ether ae:30:dc:eb:e1:05 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.101/24 brd 192.168.1.255 scope global eth0
        valid_lft forever preferred_lft forever
```

4.1.2. Automatically permanently configure a static IP address

IP addresses configured through the “ifconfig” and “ip” commands are lost when power is lost, and if you need to make IP addresses permanent, you need to modify the corresponding configuration files of the network management tool.

- **Configure IP Addresses by systemd-networkd Dynamically**

The default image of Remi Pi uses systemd-networkd for NIC configuration. For details, see the configuration file 80-wired.network in the /lib/systemd/network/ directory.

```
root@myir-remi-1g:/# cat /lib/systemd/network/80-wired.network
[Match]
Name=en* eth*
KernelCommandLine=!nfsroot
KernelCommandLine=ip

[Network]
DHCP=yes

[DHCP]
RouteMetric=10
ClientIdentifier=mac
```

The preceding configuration file will configure the NIC that matches en* and eth*. If the kernel startup parameter does not contain the nfsroot parameter, DHCP will automatically configure the IP address, gateway, and DNS for the NIC that matches. See the following links for detailed configuration parameters:
<https://www.freedesktop.org/software/systemd/man/systemd.network.html>.

- **systemd-networkd permanently configures a static IP address**

If you need to configure a permanent IP address for eth0, you can compile another 10-eth0-static.network file and place it in the /etc/systemd/network/ directory:

```
root@myir-remi-1g:/# vi /etc/systemd/network/10-eth0-static.network
[Match]
Name=eth0

[Network]
```

Address=192.168.1.100/24

Gateway=192.168.1.1

After the configuration is complete, restart systemd-networkd.service. You can find that the IP address of eth0 is 192.168.1.100, as shown in the following figure:

```
root@myir-remi-1g:/# systemctl restart systemd-networkd.service
```

```
root@myir-remi-1g:/# ifconfig eth0
```

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500  metric 1
    inet 192.168.1.100  netmask 255.255.255.0  broadcast 192.168.1.255
    inet6 fe80::ac30:dcff:feeb:e105  prefixlen 64  scopeid 0x20<link>
    ether ae:30:dc:eb:e1:05  txqueuelen 1000  (Ethernet)
    RX packets 3514  bytes 504284 (492.4 KiB)
    RX errors 0  dropped 20  overruns 0  frame 0
    TX packets 118  bytes 12680 (12.3 KiB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

4.2. WIFI

This section mainly introduces the configuration and use of Wi-Fi under Linux. Usually, Wi-Fi module can support two working modes, namely STA mode and AP mode, and some devices also support STA and AP mode to work at the same time. STA mode allows the device to connect to external Wi-Fi hotspots, and AP mode turns the device into a Wi-Fi hotspot for other devices to connect to. The Remi Pi onboard WIFI module is not included in the device tree started by default. The device tree name needs to be changed in the uboot phase as follows:

```
r=> ls mmc 0:1
23665152  Image
    39850  mys-rzg2l-sdcard.dtb
    39703  mys-rzg2l-wifi.dtb
3 file(s), 0 dir(s)
=> setenv fdt_file mys-rzg2l-wifi.dtb
=> setenv sdhi_sw "if test MYS-RZG2L = MYS-RZG2L ; then switch_sdhi1 wifi;
fi"
=> saveenv
Saving Environment to MMC... Writing to MMC(0)... OK
```

After restart by pressing the reset key, the network node wlan0 of the Wi-Fi device is generated as follows:

```
root@myir-remi-1g:/# ifconfig wlan0
wlan0: flags=4098<BROADCAST,MULTICAST>  mtu 1500  metric 1
    ether 68:4e:05:b3:fe:d1  txqueuelen 1000  (Ethernet)
    RX packets 0  bytes 0 (0.0 B)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 0  bytes 0 (0.0 B)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

4.2.1. STA mode manually connects to the WIFI hotspot

Let's try manually connecting to a nearby Wi-Fi hotspot called "myir007", which is a WPA2-encrypted Wi-Fi hotspot with the password 12345678.

- **Ensure that the wlan0 network device is active**

```
root@myir-remi-1g:/# ifconfig wlan0 up
[ 1827.639470] RTW: rtw_ndev_notifier_call(wlan0) state:14
[ 1827.644831] RTW: _netdev_open(wlan0) , bup=0
[ 1827.651159] RTW: rtl8733b_hal_init fw source from array
[ 1827.668406] RTW: SetHwReg: bMacPwrCtrlOn=1
[ 1827.770287] RTW: _rtw_hal_set_fw_rsvd_page(wlan0) Get [ NOR ] RsvdPageN
Um ==>
[ 1827.777528] RTW: LocPsPoll: 4
[ 1827.781447] RTW: LocNullData: 5
[ 1827.785200] RTW: LocQosNull: 6
[ 1827.788747] RTW: LocBTQosNull: 7
[ 1827.792575] RTW: _rtw_hal_set_fw_rsvd_page(wlan0) Get [ NOR ] RsvdPageN
Um <==
.....
```

- **Scan nearby WIFI hotspots**

```
root@myir-remi-1g:/# iw dev wlan0 scan | grep SSID
.....
[ 1957.037944] RTW: rtw_indicate_scan_done(wlan0)
    SSID: myir007
    SSID: myir_hys_2.4G
    SSID: myir0000
    SSID: Myir-caigou_Wi-Fi5
    SSID: Myir-caigou
    SSID: myir_hong
    SSID: myir007
.....
```

- **wpa_passphrase Sets the wifi name and password**

```
root@myir-remi-1g:/# wpa_passphrase myir007 12345678 >> /etc/wpa_supplic
ant.conf
root@myir-remi-1g:/# cat /etc/wpa_supplicant.conf
```

```
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=0
update_config=1

network={
    key_mgmt=NONE
}
network={
    ssid="myir007"
    #psk="12345678"
    psk=a1b5192bfbe5bd3b942bc83213d6c0e0bd281194844dbec97f4ece71
6946a2ed
}
```

- **Close the wpa_supplicant process**

Before using wpa_supplicant to connect and configure WIFI, check whether the wpa_supplicant process is open and then close the process.

```
root@myir-remi-1g:/# killall wpa_supplicant
```

- **Initialize wpa_supplicant**

wpa_supplicant is a tool for connecting and configuring WIFI. It interacts with the driver through the socket and reports data to the user layer. The user layer can also send commands to wpa_supplicant to mobilize the driver through the socket to operate the WiFi chip. It usually runs in the background, as shown below:

```
root@myir-remi-1g:/# wpa_supplicant -B -Dnl80211 -c /etc/wpa_supplicant.conf -i wlan0
[ 2763.442086] RTW: _rtw_pwr_wakeup call ips_leave....
[ 2763.447964] RTW: ==>ips_leave cnts:2
[ 2763.451539] RTW: ==> rtw_ips_pwr_up.....
[ 2763.456490] RTW: ==> ips_netdrv_open.....
[ 2763.460979] RTW: rtl8733b_hal_init fw source from array
[ 2763.477028] RTW: SetHwReg: bMacPwrCtrlOn=1
[ 2763.578537] RTW: _rtw_hal_set_fw_rsvd_page(wlan0) Get [ NOR ] RsvdPageN
Um ==>
```

```
[ 2763.585789] RTW: LocPsPoll: 4
[ 2763.588786] RTW: LocNullData: 5
[ 2763.591978] RTW: LocQosNull: 6
[ 2763.595026] RTW: LocBTQosNull: 7
[ 2763.598465] RTW: _rtw_hal_set_fw_rsvd_page(wlan0) Get [ NOR ] RsvdPageN
Um <==
[ 2764.031156] RTW: rtl8733b_hal_init Download Firmware from array success
[ 2764.037790] RTW: NIC FW Version:1 SubVersion:10 FW size:116576
[ 2764.052602] RTW: is_valid_id_status: HALMAC_FEATURE_DUMP_PHYSICAL_EF
USE
[ 2764.059295] RTW: is_valid_id_status: HALMAC_FEATURE_DUMP_PHYSICAL_EF
USE
[ 2764.066969] RTW: is_valid_id_status: HALMAC_FEATURE_DUMP_PHYSICAL_EF
USE
[ 2764.074347] RTW: is_valid_id_status: HALMAC_FEATURE_DUMP_PHYSICAL_EF
USE
[ 2764.081647] RTW: is_valid_id_status: HALMAC_FEATURE_DUMP_PHYSICAL_EF
USE
[ 2764.089025] RTW: is_valid_id_status: HALMAC_FEATURE_DUMP_PHYSICAL_EF
USE
[ 2764.096401] RTW: is_valid_id_status: HALMAC_FEATURE_DUMP_PHYSICAL_EF
USE
[ 2764.103720] RTW: is_valid_id_status: HALMAC_FEATURE_DUMP_PHYSICAL_EF
USE
[ 2764.111043] RTW: is_valid_id_status: HALMAC_FEATURE_DUMP_PHYSICAL_EF
USE
[ 2764.118502] RTW: is_valid_id_status: HALMAC_FEATURE_DUMP_PHYSICAL_EF
USE
[ 2764.125871] RTW: is_valid_id_status: HALMAC_FEATURE_DUMP_PHYSICAL_EF
USE
.....
```

➤-B: Background daemon.

➤-D: Driver name.

- -c: Configuration information path.
- -i:wifi interface for listening.

- **Obtain the WIFI ip address**

```
root@myir-remi-1g:/# udhcpc -i wlan0
udhcpc: started, v1.31.1
udhcpc: sending discover
[ 2894.112035] RTW: rtw_set_ps_mode(wlan0) Leave 802.11 power save - WIFI-
LPS_CTRL_SPECIAL_PACKET
udhcpc: sending select for 192.168.30.100
udhcpc: lease of 192.168.30.100 obtained, lease time 7200
/etc/udhcpc.d/50default: Adding DNS 202.96.128.166
/etc/udhcpc.d/50default: Adding DNS 202.96.134.133
root@myir-remi-1g:/# [ 2894.968151] RTW: OnAction_back
[ 2894.971252] RTW: OnAction_back, action=0
[ 2894.975661] RTW: issue_addba_rsp_wait_ack(wlan0) ra=3c:06:a7:7b:59:bd stat
us:=0 tid=0 size:64, acked, 1/3 in 0 ms
[ 2898.567903] RTW: rtw_set_ps_mode(wlan0) Enter 802.11 power save - WIFI-
TRAFFIC_IDLE

root@myir-remi-1g:/# ifconfig wlan0
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500  metric
1
    inet 192.168.30.100  netmask 255.255.255.0  broadcast 192.168.30.255
    inet6 fe80::6a4e:5ff:feb3:fed1  prefixlen 64  scopeid 0x20<link>
    ether 68:4e:05:b3:fe:d1  txqueuelen 1000  (Ethernet)
    RX packets 51  bytes 17225 (16.8 KiB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 38  bytes 6703 (6.5 KiB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

- **Ping Baidu**

```
root@myir-remi-1g:/# udhcpc -i wlan0
root@myir-remi-1g:/# ping www.baidu.com -I wlan0
```

```
PING www.baidu.com (14.215.177.3[ 3263.551223] RTW: rtw_set_ps_mode(wlan0) Leave 802.11 power save - WIFI-LPS_CTRL_LEAVE
9): 56 data bytes
64 bytes from 14.215.177.39: seq=0 ttl=55 time=10.578 ms
64 bytes from 14.215.177.39: seq=1 ttl=55 time=8.370 ms
[ 3265.550269] RTW: rtw_set_ps_mode(wlan0) Leave 802.11 power save - WIFI-LPS_CTRL_LEAVE
64 bytes from 14.215.177.39: seq=2 ttl=55 time=9.018 ms
64 bytes from 14.215.177.39: seq=3 ttl=55 time=8.810 ms
[ 3267.495882] RTW: rtw_set_ps_mode(wlan0) Enter 802.11 power save - WIFI-TRAFFIC_IDLE
[ 3267.505153] RTW: Turbo EDCA =0x5ea42b
[ 3267.550986] RTW: rtw_set_ps_mode(wlan0) Leave 802.11 power save - WIFI-LPS_CTRL_LEAVE
64 bytes from 14.215.177.39: seq=4 ttl=55 time=9.029 ms
```

If you cannot ping the public network, add it to /etc/resolv.conf:

```
# Generated by Connection Manager
nameserver 8.8.8.8
nameserver 8.8.4.4
```

4.2.2. STA mode script manually connects to WIFI hotspot

Section 4.2.1 describes the wifi connection in detail, which is easy to understand. If it is troublesome to run the following command, the connection can be successfully connected. For details about how to obtain the ip address after the connection is successful, see Section 4.2.1

```
root@myir-remi-1g:/# ifup_wifi_sta -ssid myir007 -passwd 12345678
```

4.2.3. The STA automatically connects to the WIFI hotspot upon startup

It is very inconvenient to manually connect the wifi hotspot every time you start up or execute the script manually. Here, I will introduce how to automatically connect the wifi hotspot at startup. After connecting the wifi hotspot by using

ifup_wifi_sta above, A new file is generated in the /etc/wpa_supplicant/ directory, as shown below:

```
root@myir-remi-1g:/# ls etc/wpa_supplicant
wpa_supplicant-wlan0.conf
root@myir-remi-1g:/# cat etc/wpa_supplicant/wpa_supplicant-wlan0.conf
ctrl_interface=/var/run/wpa_supplicant
ctrl_interface_group=0
update_config=1

network={
    ssid="myir007"
    #psk="12345678"
    psk=a1b5192bfbe5bd3b942bc83213d6c0e0bd281194844dbec97f4ece71
6946a2ed
}
```

As long as the corresponding service is enabled, you can boot and connect to WIFI.

```
root@myir-remi-1g:/# systemctl enable wpa_supplicant@wlan0.service
Created symlink /etc/systemd/system/multi-user.target.wants/wpa_supplicant@wlan0.service -> /lib/systemd/system/wpa_supplicant@.service.
```

After the restart, you can see that the WIFI is connected, but there is no ip address.

```
root@myir-remi-1g:~# wpa_cli -iwlan0 -p/var/run/wpa_supplicant status
bssid=3c:06:a7:7b:59:bd
freq=5785
ssid=myir007
id=0
mode=station
wifi_generation=4
pairwise_cipher=CCMP
group_cipher=CCMP
key_mgmt=WPA2-PSK
wpa_state=COMPLETED
address=68:4e:05:b3:fe:d1
```

```
uuid=b979bff3-1d2a-53f5-82d4-6ef67e178b30
```

According to the static ip address configuration of the Ethernet card, you only need to set wlan0 to automatically obtain the ip address after the system starts up. For details, see /lib/systemd/network/80-wired.network to create 11-wlan0.network.

```
root@myir-remi-1g:~# vi /etc/systemd/network/11-wlan0.network
root@myir-remi-1g:~# cat /etc/systemd/network/11-wlan0.network
[Match]
Name=wlan0
[Network]
DHCP=yes
[DHCP]
RouteMetric=20
```

Restart the development board again, and you can see that WIFI has obtained the IP.

```
root@myir-remi-1g:~# ifconfig wlan0
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500  metric
1
    inet 192.168.30.100  netmask 255.255.255.0  broadcast 192.168.30.255
    inet6 fe80::6a4e:5ff:feb3:fed1  prefixlen 64  scopeid 0x20<link>
    ether 68:4e:05:b3:fe:d1  txqueuelen 1000  (Ethernet)
```

4.2.4. Manually configure WIFI hotspots in AP mode

Hostapd is a wireless access point program with encryption function. It is a convenient tool for building wireless access points on Linux operating system. Support the IEEE 802.11 protocol and IEEE802.1 X/WPA/connected/EAP/RADIUS encryption. The board serves as a WiFi hotspot and needs to assign network parameters such as IP and route to each terminal (such as a mobile phone) that accesses the hotspot. For example, create a wireless wifi hotspot with SSID as MYIR_TEST and PASSWD as myir2020. The following is a manual configuration:

- **Configure the address of wlan0**

When the AP mode is used, you need to activate wlan0 and configure a static IP address. In this case, set the default IP address 192.168.10.1

```
root@myir-remi-1g:/# ifconfig wlan0 192.168.10.1 up
[ 2776.252128] RTW: rtw_ndev_notifier_call(wlan0) state:14
[ 2776.257828] RTW: _netdev_open(wlan0) , bup=1
[ 2776.262208] RTW: -871x_drv - drv_open, bup=1
[ 2776.267079] RTW: cfg80211_rtw_set_power_mgmt(wlan0) enabled:1, timeout:
.....
```

We explained earlier that the current Wi-Fi module does not support the simultaneous operation of STA and AP modes, so the configuration of STA mode needs to be cleared here, as follows:

```
root@myir-remi-1g:/# killall udhcpd
root@myir-remi-1g:/# killall wpa_supplicant
```

- **Run the DHCP service using the wlan0 device**

wlan0 works in AP mode. When other external devices connect to the AP hotspot, wlan0 dynamically assigns IP addresses to other external devices. Therefore, wlan0 is used to run the DHCP service program udhcpd. The udhcpd configuration file is /etc/udhcpd1.conf.

```
root@myir-remi-1g:/# vi /etc/udhcpd1.conf
root@myir-remi-1g:/# cat /etc/udhcpd1.conf
#the start and end of the IP lease block
start 192.168.10.10
end 192.168.10.254
# the interface that udhcpd will use
interface wlan0
opt dns 8.8.8.8
option subnet 255.255.255.0
opt router 192.168.10.1
option domain local
option lease 864000
```

When the AP mode is configured and the external device connects to the hotspot, it obtains the IP address in the address pool through wlan0. The IP address range is 192.168.10.10 to 192.168.10.254, and the subnet mask is 255.255.255.0. The default gateway is 192.168.10.1, the DNS is 8.8.8.8, and the DHCP lease is 864000

seconds. After the udhcpd configuration file is ready, run the following command to start the udhcpd service:

```
root@myir-remi-1g:/# udhcpd /etc/udhcpd1.conf
```

- **Run hostapd service using wlan0**

The most critical step in configuring the AP mode is to start the hostapd service. Before starting services, you need to run /etc/hostapd.conf to configure the AP ssid, password, encryption algorithm, driver type, and working mode. For details about parameter configurations, see:

<http://w1.fi/cgiit/hostap/plain/hostapd/hostapd.conf>, Here is the configuration of "/etc/hostapd1.conf" for the current hardware, as follows:

```
root@myir-remi-1g:/# vi /etc/hostapd1.conf
root@myir-remi-1g:/# cat /etc/hostapd1.conf
# File: /etc/hostapd.conf
interface=wlan0
driver=nl80211
# mode Wi-Fi (a = IEEE 802.11a, b = IEEE 802.11b, g = IEEE 802.11g)
hw_mode=g
ssid=MYIR_TEST
channel=7
wmm_enabled=0
macaddr_acl=0
# Wi-Fi closed, need an authentication
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=myir2020
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

After the configuration file is ready, run the following command to start the hostapd service.

```
root@myir-remi-1g:/# hostapd -B /etc/hostapd1.conf
Configuration file: /etc/hostapd1.conf
```

```
[ 3658.365208] RTW: cfg80211_rtw_get_txpower(wlan0) total max: -10000 mbm
[ 3658.371976] RTW: cfg80211_rtw_get_txpower(wlan0) total max: -10000 mbm
[ 3658.379155] RTW: cfg80211_rtw_get_txpower(wlan0) total max: -10000 mbm
.....
wlan0: interface state UNINITIALIZED->ENABLED
w[ 3660.362565] RTW: rtw_ac_parm_cmd_hdl(wlan0) AC_BE = 0x00006425
lan0: AP-ENABLED[ 3660.369424] RTW: rtw_ac_parm_cmd_hdl(wlan0) AC_BK =
0x0000a4
```

● Connect to WIFI

After connecting to the "MYIR_TEST" wifi hotspot using a mobile phone, the IP address 192.168.10.10 is displayed on the mobile phone, and the development board can ping the mobile phone. But the phone can't connect to the Internet.

```
root@myir-remi-1g:/# ping 192.168.10.10
PING 192.168.10.10 (192.168.10.10): 56 data bytes
64 bytes from 192.168.10.10: seq=0 ttl=64 time=2.388 ms
64 bytes from 192.168.10.10: seq=1 ttl=64 time=2.758 ms
64 bytes from 192.168.10.10: seq=2 ttl=64 time=4.701 ms
64 bytes from 192.168.10.10: seq=3 ttl=64 time=289.663 ms
```

● Set network forwarding

Currently, only the wifi hotspot is available. You need to set the wireless NIC to share eth0 to access the Internet. The development board is used to connect to the router, and the ip address and dns can be obtained normally, and then the forwarding and camouflage forwarding can be set. At this time, you can see that the mobile phone can access the Internet.

```
root@myir-remi-1g:/# echo "1" > /proc/sys/net/ipv4/ip_forward
root@myir-remi-1g:/# iptables -t nat -A POSTROUTING -s 192.168.10.1/24 -o
eth0 -j MASQUERADE
```

4.2.5. The AP mode script configures the WIFI hotspot

```
root@myir-remi-1g:~# ifup_wifi_ap
killall: udhcpd: no process killed
```

4.3. BLUETOOTH

BlueZ (<http://www.bluez.org/>) is commonly used to configure and manage Bluetooth devices on Linux platforms. BlueZ is a relatively complete toolset and protocol stack for the configuration and management of Bluetooth. These tools are used to configure and use Bluetooth

The Remi Pi development board automatically loads Bluetooth devices upon startup. The following information is displayed:

```
myir-remi-1g login: root (automatic login)
Last login: Fri Jan  1 00:00:24 UTC 2066
[ 40.105483] Bluetooth: h5_open
[ 40.109183] Bluetooth: hci_uart_register_dev
[ 40.128827] rtk_btcoex: Open BTCOEX
[ 40.151383] rtk_btcoex: BTCOEX hci_rev 0x05a8
[ 40.156956] rtk_btcoex: BTCOEX Imp_subver 0xcbcd
[ 42.247284] Bluetooth: __hci_uart_flush: hdev 00000000e90a925d tty 000000
0013eebac5
[ 42.255416] Bluetooth: hci_uart_close: hdev 00000000e90a925d
[ 42.261316] Bluetooth: __hci_uart_flush: hdev 00000000e90a925d tty 000000
0013eebac5
[ 42.269195] rtk_btcoex: Close BTCOEX
[ 42.272911] rtk_btcoex: -x
```

After successful loading, the hci0 device node is generated

```
root@myir-remi-1g:~# rfkill list
0: hci0: Bluetooth
    Soft blocked: no
    Hard blocked: no
```

The following details how to configure and connect nearby Bluetooth devices using the BlueZ toolset:

● Activate Bluetooth device hci0

```
root@myir-remi-1g:~# hciconfig hci0 up
```

```
[ 194.484340] rtk_btcoex: Open BTCOEX  
[ 194.736818] Bluetooth: hu 00000000d6f0a907 retransmitting 1 pkts  
[ 194.744463] rtk_btcoex: BTCOEX hci_rev 0x05a8  
[ 194.748899] rtk_btcoex: BTCOEX Imp_subver 0xcbcd
```

● **bluetoothctl Enters the Bluetooth control interface**

```
root@myir-remi-1g:~# bluetoothctl  
Agent registered  
[CHG] Controller 98:03:CF:6C:E8:C3 Pairable: yes  
[bluetooth]#
```

● **The Bluetooth control power was enabled**

```
[bluetooth]# power on  
Changing power on succeeded
```

● **The Bluetooth proxy was enabled**

```
[bluetooth]# agent on  
Agent is already registered  
[bluetooth]# default-agent  
Default agent request successful
```

● **Scan for nearby Bluetooth devices that can connect**

```
[bluetooth]# scan on  
Device 49:E7:6F:9E:C0:C3 49-E7-6F-9E-C0-C3  
Device 64:00:D3:46:C0:9A 64-00-D3-46-C0-9A  
Device 54:33:A6:CB:2E:0D 54-33-A6-CB-2E-0D  
Device 6B:AC:F3:B7:FB:F3 6B-AC-F3-B7-FB-F3  
Device 79:71:6C:5E:9F:C6 79-71-6C-5E-9F-C6  
Device 5D:80:04:00:3F:1D 5D-80-04-00-3F-1D  
Device F0:9E:4A:5E:61:94 LAPTOP-R5OA69F7  
Device D0:B4:5D:7F:F5:61 Tony929 的 P50 Pro  
Device 77:F7:E4:9E:25:0F 77-F7-E4-9E-25-0F  
Device 6E:48:26:20:FF:44 6E-48-26-20-FF-44  
Device 44:D6:4B:25:56:35 44-D6-4B-25-56-35  
Device C0:3C:59:46:38:49 USER-WU  
Device 4C:C4:38:AA:DA:6B 4C-C4-38-AA-DA-6B  
Device CC:42:10:3D:82:A7 ZRY
```

```
Device 60:84:2E:20:F8:4A 60-84-2E-20-F8-4A
Device F0:A6:54:99:AA:F6 QDXCXY
Device 4B:9E:34:66:F3:98 4B-9E-34-66-F3-98
Device AC:50:DE:C1:42:CE DUXY
Device 1C:D1:07:7B:33:88 ...
Device 6F:7F:D0:61:EE:53 6F-7F-D0-61-EE-53
Device 84:5C:F3:24:9C:89 DESKTOP-LJRC636
Device DC:21:48:97:6F:C0 YZQ
Device F4:60:E2:A3:B3:3D 希灵小米 mix Ω
Device 5A:4A:13:38:3C:91 5A-4A-13-38-3C-91
Device 98:B3:79:0F:8E:B3 PENG
```

● Pair equipment

```
[bluetooth]# pair 98:B3:79:0F:8E:B3
Attempting to pair with 9C:92:4F:6D:31:7[ 117.163760] rtk_btcoex: hci (periodi
c)inq cancel/exit
7
[ 117.175670] rtk_btcoex: hci create connection, start paging
[bluetooth]# [ 120.377464] rtk_btcoex: connected, handle 0001, status 0x00
[ 120.383186] rtk_btcoex: Page success
[ 120.689437] rtk_btcoex: io capability request
[CHG] Device 98:B3:79:0F:8E:B3 Connected: yes
Request confirmation
[agent] Confirm passkey 563182 (yes/no): yes
[ 133.011567] rtk_btcoex: l2cap op 6, len 16, out 1
[ 133.016618] rtk_btcoex: TX l2cap disconn req, hndl 0x0001, dcid 0x0108, sc
id 0x0040
[ 133.024417] rtk_btcoex: handle_l2cap_discon_req: handle 0x0001, dcid 0x010
8, scid 0x0040, dir 1
```

● Connecting device

```
[bluetooth]# connect 98:B3:79:0F:8E:B3
Attempting to connect to 98:B3:79:0F:8E:B3
[CHG] Device 98:B3:79:0F:8E:B3 Connected: yes
Connection successful
```

5. Expand peripheral interfaces

Remi Pi development board provides a wealth of peripheral interface, in addition to the basic peripheral interface, but also external to a variety of expansion modules. It makes the user's development more flexible and convenient. The following describes the test steps of several optional modules introduced by Mill. Users buy them on demand.

5.1. MY-WiredCom Module

MY-WiredCom module is a form of peripheral interface for Raspberry PI introduced by Mill, including RS232/RS485/CAN/SPI/I2C and other peripheral interfaces. This module needs to be purchased by users according to their needs. For module details, please refer to Table 1-2 for the list of optional modules. Users need to connect the module to the J13 interface of the development board before testing.

5.1.1. RS232

This section uses the Linux API to configure the sending and receiving functions of the development board RS232. The serial port device file in Linux is usually named `/dev/ttySCn(n=0,1,2,3.....)`. `n` indicates the serial port device number in Linux. `ttySC` is the serial port device name defined by the kernel. This section takes `scif2` as an example to test, and the corresponding device node is `ttySC2`. Its test configuration is shown in the following table:

Table 5-1.RS232 port configurations

Interface	Remi Pi	windows 10
Hardware interface	RS232	USB-RS232 Module
Device node	ttySC2	com4
Tools	uart_test	sscom

Connect the USB-RS232 converter to the USB Host port of the PC and connect the RXD and TXD of J2 to the TXD and RXD of the USB-RS232 converter, respectively.

- **Test Receiving Data from RS232 of Development Board**

In windows, the sscom sends data:

```
[11:31:59.945]Send→◇123456789
[11:32:00.859]Send→◇123456789
[11:32:01.627]Send→◇123456789
[11:32:02.485]Send→◇123456789
[11:32:03.987]Send→◇123456789
[11:32:04.660]Send→◇123456789
[11:32:05.443]Send→◇123456789
[11:32:06.308]Send→◇123456789
[11:32:07.303]Send→◇123456789
[11:32:08.722]Send→◇123456789
```

Execute the following command on the development board to receive the data. After the execution of the command, the interrupt will enter a blocking state, waiting to receive data sent from the computer serial port:

```
root@myir-remi-1g:/# uart_test /dev/ttySC2 -b 115200 -r -asc -l
/dev/ttySC2 RECV=[9] : 123456789
/dev/ttySC2 RECV=[9] : 123456789
/dev/ttySC2 RECV=[9] : 123456789
/dev/ttySC2 RECV=[9] : 123456789
/dev/ttySC2 RECV=[9] : 123456789
/dev/ttySC2 RECV=[9] : 123456789
/dev/ttySC2 RECV=[9] : 123456789
/dev/ttySC2 RECV=[9] : 123456789
/dev/ttySC2 RECV=[9] : 123456789
/dev/ttySC2 RECV=[9] : 123456789
```

The data sent in windows is consistent with the data received by the development board, that is, the RS232 of the development board receives data normally.

- **Test Sending Data to RS232 from the Development Board**

The development board sends data:

```
root@myir-remi-1g:/# uart_test /dev/ttySC2 -b 115200 -w 123456789 -asc -l
/dev/ttySC2 write=[9] :123456789
/dev/ttySC2 write=[9] :123456789
/dev/ttySC2 write=[9] :123456789
```

```
/dev/ttySC2 write=[9] :123456789
/dev/ttySC2 write=[9] :123456789
/dev/ttySC2 write=[9] :123456789
/dev/ttySC2 write=[9] :123456789
/dev/ttySC2 write=[9] :123456789
/dev/ttySC2 write=[9] :123456789
/dev/ttySC2 write=[9] :123456789
/dev/ttySC2 write=[9] :123456789
/dev/ttySC2 write=[9] :123456789
```

After the preceding command is executed, the corresponding data is automatically sent.

On windows, the sscom starts to receive data:

```
[11:39:43.330]Receive←◆123456789
[11:39:43.860]Receive←◆123456789
[11:39:44.391]Receive←◆123456789
[11:39:44.922]Receive←◆123456789
[11:39:45.454]Receive←◆123456789
[11:39:45.984]Receive←◆123456789
[11:39:46.520]Receive←◆123456789
[11:39:47.046]Receive←◆123456789
[11:39:47.577]Receive←◆123456789
[11:39:48.110]Receive←◆123456789
[11:39:48.639]Receive←◆123456789
[11:39:49.170]Receive←◆123456789
```

In windows, the data received by sscom is consistent with the data sent by the development board, that is, the RS232 data sent by the development board is normal.

5.1.2. RS485

This routine demonstrates how to test the development board's RS485 sending and receiving functions using Linux apis. The serial port for the RS485 device is scif1 and the device node is ttySC1. The following table describes the hardware configurations of the RS485 port on the J2 module:

Table 5-2.RS485bInterface configuration

Interface	Remi Pi	windows 10
Hardware Interface	RS485	USB-RS485
Device Node	ttySC1	com4
Tools	rs485_test	sscom

Connect the USB-RS485 converter to the USB Host port of the PC and connect the 485A and 485B of J2 to the 485A and 485B of the USB-RS485 converter, respectively.

- **The RS485 on the development board sends data**

When the development board sends data, P32_0 should be set to high level first.

```
root@myir-remi-1g:~# echo 376 > /sys/class/gpio/export
root@myir-remi-1g:~# echo out > /sys/class/gpio/P32_0/direction
root@myir-remi-1g:~# echo 1 > /sys/class/gpio/P32_0/value
root@myir-remi-1g:~# uart_test /dev/ttySC1 -b 115200 -w 123456789 -asc -l
/dev/ttySC1 write=[9] :123456789
/dev/ttySC1 write=[9] :123456789
/dev/ttySC1 write=[9] :123456789
/dev/ttySC1 write=[9] :123456789
/dev/ttySC1 write=[9] :123456789
/dev/ttySC1 write=[9] :123456789
/dev/ttySC1 write=[9] :123456789
/dev/ttySC1 write=[9] :123456789
/dev/ttySC1 write=[9] :123456789
/dev/ttySC1 write=[9] :123456789
/dev/ttySC1 write=[9] :123456789
/dev/ttySC1 write=[9] :123456789
/dev/ttySC1 write=[9] :123456789
/dev/ttySC1 write=[9] :123456789
/dev/ttySC1 write=[9] :123456789
/dev/ttySC1 write=[9] :123456789
/dev/ttySC1 write=[9] :123456789
/dev/ttySC1 write=[9] :123456789
/dev/ttySC1 write=[9] :123456789
/dev/ttySC1 write=[9] :123456789
```

On windows, the sscom starts to receive data.

```
[20:02:09.968]收←◆123456789
```

```
[20:02:10.500]收←◆123456789
[20:02:11.030]收←◆123456789
[20:02:11.564]收←◆123456789
[20:02:12.092]收←◆123456789
[20:02:12.623]收←◆123456789
[20:02:13.154]收←◆123456789
[20:02:13.684]收←◆123456789
[20:02:14.215]收←◆123456789
[20:02:14.747]收←◆123456789
[20:02:15.278]收←◆123456789
[20:02:15.809]收←◆123456789
[20:02:16.340]收←◆123456789
[20:02:16.869]收←◆123456789
[20:02:17.401]收←◆123456789
[20:02:17.932]收←◆123456789
[20:02:18.463]收←◆123456789
```

The data received in windows is consistent with the data sent by the development board, that is, the RS485 data sent by the development board is normal.

- **RS485 receiving data under the development board**

The development board receives data and sets P32_0 to low level.

```
root@myir-remi-1g:~# echo 0 > /sys/class/gpio/P32_0/value
root@myir-remi-1g:~# uart_test /dev/ttySC1 -b 115200 -r -asc -l
/dev/ttySC1 RECV=[10] : 1234567890
/dev/ttySC1 RECV=[10] : 1234567890
/dev/ttySC1 RECV=[10] : 1234567890
/dev/ttySC1 RECV=[10] : 1234567890
/dev/ttySC1 RECV=[10] : 1234567890
/dev/ttySC1 RECV=[10] : 1234567890
/dev/ttySC1 RECV=[10] : 1234567890
/dev/ttySC1 RECV=[10] : 1234567890
/dev/ttySC1 RECV=[10] : 1234567890
/dev/ttySC1 RECV=[10] : 1234567890
/dev/ttySC1 RECV=[10] : 1234567890
```

```
/dev/ttySC1 RECV=[10] : 1234567890  
/dev/ttySC1 RECV=[10] : 1234567890  
/dev/ttySC1 RECV=[10] : 1234567890  
/dev/ttySC1 RECV=[10] : 1234567890  
/dev/ttySC1 RECV=[10] : 1234567890  
/dev/ttySC1 RECV=[10] : 1234567890  
/dev/ttySC1 RECV=[10] : 1234567890
```

After the preceding command is executed, the application program automatically receives data.

In windows, the sscom starts to send data.

```
[20:07:02.062]发→◇1234567890  
[20:07:05.406]发→◇1234567890  
[20:07:07.549]发→◇1234567890  
[20:07:09.576]发→◇1234567890  
[20:07:10.583]发→◇1234567890  
[20:07:11.559]发→◇1234567890  
[20:07:12.743]发→◇1234567890  
[20:07:14.222]发→◇1234567890  
[20:07:15.324]发→◇1234567890  
[20:07:16.679]发→◇1234567890  
[20:07:18.198]发→◇1234567890  
[20:07:19.462]发→◇1234567890  
[20:07:20.878]发→◇1234567890  
[20:07:21.952]发→◇1234567890
```

The data sent in windows is consistent with the data received by the development board, that is, the RS485 data received by the development board is normal.

5.1.3.CAN

This section uses “cansend” and “candump” commands in Linux system for SocketCAN communication test. The test here uses two development boards for docking tests.

Here, CANH and CANL pins of J2 seat son of MY-WiredCom module are used to connect with CANH and CANL boards of the same type.

- **Initializes the CAN network interface**

Set the CAN baud rate. To use canfd, set the baud rate and enable the CAN network interface. Refer to the following command to set the arbitration baud rate of the two development boards to 500Kbit and data baud rate to 4M respectively, and enable the canfd function.

```
root@myir-remi-1g:/# ifconfig can0 down
root@myir-remi-1g:/# ip link set can0 up type can bitrate 500000 sample-point 0.75 dbitrate 4000000 dsample-point 0.8 fd on
[ 565.913022] rcar_canfd 10050000.can can0: bitrate error 3.8%
[ 565.926675] IPv6: ADDRCONF(NETDEV_CHANGE): can0: link becomes ready
root@myir-remi-1g:/# ifconfig can0 up
```

At this point, the canfd function is enabled.

- **Send Data**

Set one of the boards to send and use cansend to send data:

```
root@myir-remi-1g:~# cansend can0 100#11.22.33.44
root@myir-remi-1g:~# cansend can0 100#11.22.33.44
root@myir-remi-1g:~# cansend can0 100#11.22.33.44
root@myir-remi-1g:~# cansend can0 100#11.22.33.44
root@myir-remi-1g:~# cansend can0 100#11.22.33.44
root@myir-remi-1g:~# cansend can0 100#11.22.33.44
root@myir-remi-1g:~# cansend can0 100#11.22.33.44
root@myir-remi-1g:~# cansend can0 100#11.22.33.44
root@myir-remi-1g:~# cansend can0 100#11.22.33.44
root@myir-remi-1g:~# cansend can0 100#11.22.33.44
```

- **Receive Data**

To set the other board to receive, you CAN use candump to view the CAN receive data:

```
root@myir-remi-1g:~# candump can0 -L
(1581092023.326369) can0 100#11223344
(1581092025.003865) can0 100#11223344
(1581092025.283862) can0 100#11223344
```

```
(1581092025.499923) can0 100#11223344
(1581092025.683966) can0 100#11223344
(1581092025.852050) can0 100#11223344
(1581092026.028037) can0 100#11223344
(1581092026.171930) can0 100#11223344
(1581092026.435972) can0 100#11223344
```

● Statistics of can0 information

After receiving and sending CAN data, it displays the details and statistics of CAN device. The value of "clock" stands for CAN clock, "drop" stands for lost packets, and "overrun" stands for bus overrun, "error" stands for bus error.

```
root@myir-remi-1g:~# ip -details -statistics link show can0
4: can0: <NOARP,UP,LOWER_UP,ECHO> mtu 72 qdisc pfifo_fast state UP mode
DEFAULT group default qlen 10
    link/can  promiscuity 0 minmtu 0 maxmtu 0
    can <FD> state ERROR-ACTIVE (berr-counter tx 0 rx 0) restart-ms 0
        bitrate 500000 sample-point 0.750
        tq 20 prop-seg 37 phase-seg1 37 phase-seg2 25 sjw 1
        rcar_canfd: tseg1 2..128 tseg2 2..32 sjw 1..32 brp 1..1024 brp-inc 1
        dbitrate 3846153 dsample-point 0.769
        dtq 20 dprop-seg 4 dphase-seg1 5 dphase-seg2 3 dsjw 1
        rcar_canfd: dtseg1 2..16 dtseg2 2..8 dsjw 1..8 dbrp 1..256 dbrp-inc 1
        clock 50000000
        re-started bus-errors arbit-lost error-warn error-pass bus-off
            0          0          0          0          0          0          nu
mtxqueues 1 numrxqueues 1 gso_max_size 65536 gso_max_segs 65535
    RX: bytes  packets  errors  dropped overrun mcast
    44         11       0      0      0      0
```

5.1.4. SPI

This section demonstrates how to use the Linux API for SPI testing. Linux SPI bus device names are usually in the form of `"/dev/spidevn.p"` . (n=0, 1, 2... , p=0, 1, 2...) , where "n" represents which SPI bus line, and "p" represents which CS signal line is used in this SPI bus line. The device file node for test is `/dev/spidev0.0`

During the test, the jumper cap first needs to be connected with PZ2(SI) and PZ1(SO) of J7 SPI5 on MY-WiredCom module.

- **Application Programming for Test**

To perform a test on the SPI, you need to write an application that first calls the `open()` function to open the device file and get the file descriptor:

```
static const char *device = "/dev/spidev0.0";
.....
fd = open(device, O_RDWR);
if (fd < 0)
    perror("can't open device");
.....
```

- **SPI Bus Configuration**

Set SPI mode:

```
ret = ioctl(fd, SPI_IOC_WR_MODE32, &mode);
```

Set SPI bus data bit length to 8 bits:

```
static uint8_t bits = 8;
ret = ioctl(fd, SPI_IOC_WR_BITS_PER_WORD, &bits);
```

Set SPI bus speed to 500Kbps:

```
static uint32_t speed = 500000;
ret = ioctl(fd, SPI_IOC_WR_MAX_SPEED_HZ, &speed);
```

Transfer data:

```
transfer(fd);
```

When finished, call the `close()` function to close:

```
close(fd);
```

- **Test sending and receiving data**

Compile the SPI test program and copy the generated spidev_test application execution file to the development board to run: MOSI and MISO were short-circuited for self-loop test.

```
root@myir-remi-1g:/# ./spidev_test
spi mode: 0x0
bits per word: 8
max speed: 500000 Hz (500 KHz)

FF FF FF FF FF FF
40 00 00 00 00 95
FF FF FF FF FF FF
FF FF FF FF FF FF
FF FF FF FF FF FF
DE AD BE EF BA AD
F0 0D
```

The above data indicates that the SPI is working properly.

6. Network application

Remi Pi development board factory burning image contains some common network applications by default, convenient for users to develop or debug, this section will introduce the use of some common network tools.

6.1. PING

PING is used to test network connectivity, network delay, and packet loss rate. Once the Ethernet connection is configured as described in 4.1.1, you can simply test the network connection using PING.

6.1.1. Network Connection

When you connect a device to a switch or router via a CAT6 cable, the console displays the connection information that the kernel outputs, as follows:

```
root@myir-remi-1g:/# [ 3177.064555] ravb 11c20000.ethernet eth0: Link is Down  
[ 3180.136781] ravb 11c20000.ethernet eth0: Link is Up - 1Gbps/Full - flow control off
```

6.1.2. PING Public Network

```
root@myir-remi-1g:/# ping www.baidu.com -I eth0  
PING www.baidu.com (14.215.177.39): 56 data bytes  
64 bytes from 14.215.177.39: seq=0 ttl=56 time=7.987 ms  
64 bytes from 14.215.177.39: seq=1 ttl=56 time=8.030 ms  
64 bytes from 14.215.177.39: seq=2 ttl=56 time=7.250 ms  
64 bytes from 14.215.177.39: seq=3 ttl=56 time=7.507 ms  
64 bytes from 14.215.177.39: seq=4 ttl=56 time=7.488 ms
```

NOTE: Ping public network needs to ensure that DNS is working properly.

The above result shows that the IP address of www.baidu.com after domain name resolution is 14.215.177.39. time represents the latency of the response, and the shorter the better. In addition to testing Ethernet, the ping command can also be used to test Wi-Fi.

6.2. SSH

SSH, short for Secure Shell, is developed by the Network Working Group of the IETF. SSH is a secure protocol based on the application layer. It is reliable and provides security for remote login sessions and other network services. Generally, dropbear or OpenSSH is used on Linux to implement SSH servers and clients. Let's test the use of the SSH client and server on an Ethernet connection. The current factory defaults include clients and services provided by Dropbear v2019.78. Configure the connection between the Ethernet port on the development board and the SSH server. The IP address of the Ethernet card is as follows:

```
root@myir-remi-1g:/# ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500  metric 1
    inet 192.168.1.181  netmask 255.255.255.0  broadcast 192.168.1.255
    inet6 fe80::c884:3eff:fe29:8f38  prefixlen 64  scopeid 0x20<link>
    ether ca:84:3e:29:8f:38  txqueuelen 1000  (Ethernet)
    RX packets 37895  bytes 6003655 (5.7 MiB)
    RX errors 0  dropped 131  overruns 0  frame 0
    TX packets 647  bytes 69762 (68.1 KiB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
    device interrupt 180
```

The IP address of the SSH server is 192.168.1.181. Run the ping command to check whether the connection between the development board and the SSH server is normal.

● SSH Client Test

The device connects to the SSH server as an SSH client. The SSH command is used to log in the SSH server on the device. The commands and results are as follows:

```
root@myir-remi-1g:/# ssh hjx@192.168.1.168

Host '192.168.1.168' is not in the trusted hosts file.
(ecdsa-sha2-nistp256 fingerprint sha1!! d7:94:a8:4e:04:88:48:15:c7:94:75:40:83:c3:
b6:ec:58:aa:f0:11)
Do you want to continue connecting? (y/n) y
```

```
hjk@192.168.1.168's password:
Welcome to Ubuntu 16.04.7 LTS (GNU/Linux 4.15.0-142-generic x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:        https://ubuntu.com/advantage

42 packages can be updated.
2 updates are security updates.

hjk@ubuntu:~$ ls
```

hjk indicates the user name on the server.

After successful login, the SSH server automatically accesses the console console, and users can perform hjk user permissions on the remote server from the client. To exit, run the "exit" command on the current console.

● SSH Server Test

The device acts as an SSH server, and other devices are connected to the device remotely. Since SSH service is also started on the device side by default, we can also use SSH commands on other devices with SSH clients to log in to the current device, with the following commands and results:

```
hjk@ubuntu:~$ ssh root@192.168.1.181
root@myir-remi-1g:~# ls
mxapp2
```

In the above example, we log in to the development board remotely as root and access the console console, which allows us to control the development board as root. To exit, run the "exit" command on the console.

```
root@myir-remi-1g:~# exit
Connection to 192.168.1.181 closed.
```

6.3. SCP

SCP is short for Secure Copy. It is a secure remote file copy command based on the SSH protocol in linux. It is very useful in system debugging.

The example of remote login using the SSH protocol and SSH client and server has been introduced. Here is an example of remote file copy using the SCP command:

6.3.1. Copy Files from Remote to Local

```
hjk@ubuntu:~$ scp hello.txt root@192.168.1.181:/home  
hello.txt
```

Go to the home directory of the development board and you can see the file as follows:

```
root@myir-remi-1g:/# cd home/  
root@myir-remi-1g:/home# ls  
ftp hello.txt root  
root@myir-remi-1g:/home#
```

6.3.2. Copy Files form Local to Remote

```
root@myir-remi-1g:/# scp read.txt hjk@192.168.1.168:/home/hjk  
hjk@192.168.1.168's password:  
read.txt 100% 16 0.0KB/s 00:00
```

During the copying process, please follow the prompts to input. After successful verification, the file is copied from the device to the \$HOME directory of the specified account on the server.

You can also copy directories by adding the "-r" parameter, as the SCP command helps.

6.4. FTP

FTP(File Transfer Protocol) is a network protocol for file transfer. The protocol follows the client-server communication model. To transfer files using FTP, the user needs to run an FTP client program and initiate a connection to a remote computer running the FTP server software. Once the connection is established, the client can choose to send and/or receive a copy of the file. The FTP server listens on TCP port 21 for connection requests from FTP clients. When a request is received, the server uses this port to control the connection and opens a separate port to transfer the file data.

The default factory myir-image-full image includes the FTP client program ftp and the server program vsftpd(for the specific migration process, see the Remi Pi Software Development Guide). Start the development board, and the FTP root directory of the development board is located at /var/lib/ftp. The following test is an example of anonymously logging in to the target development board using the ftp command in the LAN and transferring files from the development host (IP address 192.168.1.38) to the target development board (IP address 192.168.1.112).

6.4.1. Login from Host PC to Target Device with FTP

The development host is a PC with Windows10 system, open the cmd command window, and log in to the development board with FTP, the user name is root, and the password is not.

```
C:\Users\Administrator>ftp 192.168.1.112
Connect to 192.168.1.112.
220 (vsFTPd 3.0.3)
200 Always in UTF8 mode.
user(192.168.1.112:(none)): root
331 Please specify the password.
password:
230 Login successful.
```

6.4.2. Create test files on the development board

```
root@myir-remi-1g:~# touch aaa bbb ccc
```

```
root@myir-remi-1g:~# ls
MEasyTest-DEV MYD-YG2L23_factoryconfig.json aaa bbb ccc v4l2-init.sh
```

6.4.3. Development host FTP View the current directory

```
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
MEasyTest-DEV MYD-YG2L23_factoryconfig.json aaa bbb ccc v4l2-init.sh22
226 Directory send OK.
```

6.4.4. Download file

The FTP command line supports downloading multiple files using the mget subcommand, but requires user confirmation in the middle of each file, as shown below:

```
ftp> mget aaa bbb
200 Switching to ASCII mode.
mget aaa? y
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for aaa (0 bytes).
226 Transfer complete.
mget test? y
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for test (0 bytes).
226 Transfer complete.
ftp>bye
```

6.4.5. Upload file

FTP need ordinary user or root user login to upload files, here use root user login, user name is "root", no password, directly press "Enter" key, if you need password users can directly set the root password in the target development board.

- **Settings root User Password**

```
root@myir-remi-1g:/# passwd root
New password:1
```

```
Retype new password:1  
passwd: password updated successfully
```

- **FTP connects to the target board**

```
C:\Users\Administrator>ftp 192.168.1.112  
Connect to 192.168.1.112.  
220 (vsFTPd 3.0.3)  
200 Always in UTF8 mode.  
user(192.168.1.112:(none)): root  
331 Please specify the password.  
password:1  
230 Login successful.  
ftp>
```

- **Upload file**

FTP file transfer supports the ASCII and Binary modes. The default mode is ASCII. If you want to transfer binary files, such as executable programs on the development board, use the Binary mode. As follows:

```
ftp> binary  
200 Switching to Binary mode.  
ftp> put aaa  
200 PORT command successful. Consider using PASV.  
150 Ok to send data.  
226 Transfer complete.  
ftp>bye
```

FTP dates back to the early 1970s and was written without any security concerns. It does not use encryption for anything. Login credentials, such as username and password, and the data you downloaded or uploaded are transmitted in clear text. Therefore, it is not suitable for the transmission of sensitive information over the Internet, but considering its good compatibility and stability, it is very convenient to use in the LAN.

6.5. TFTP

Like FTP, TFTP uses client and server software to connect and transfer files between two devices, but the difference is that TFTP uses UDP protocol, which does not have the login function. It is very simple, especially suitable for transferring and backing up firmware, configuration files and other information on the device and server side. For example, TFTP protocol is supported in the common U-boot, which can load the server-side Linux system through the network and realize the function of network startup.

The default image file contains the TFTP client program provided by BusyBox, with the following command syntax:

```
root@myir-remi-1g:/# tftp --help
BusyBox v1.31.1 () multi-call binary.
```

```
Usage: tftp [OPTIONS] HOST [PORT]
```

The detailed parameters are described below:

- -g : Obtain file
- -p : Upload/Put file
- -l : Local files
- -r : Remote files
- HOST: Remote host name or IP address
- [PORT]: Optional remote host port, default is 69

Users can choose tftp-hpa on Linux, can also choose tftpd32/64

(http://tftpd32.jounin.net/tftpd32_download.html) on Windows as the TFTP server application. Now, we choose “tftpd-hpa” on Ubuntu as an example to show the TFTP server configuration.

6.5.1. Install TFTP Server Application

```
$ sudo apt-get install tftp-hpa tftpd-hpa
```

6.5.2. Configure TFTP Server

Create the TFTP server working directory and open the TFTP service configuration file as follows:

```
$ mkdir -p <WORKDIR>/tftpboot  
$ chmod -R 777 <WORKDIR>/tftpboot  
$ sudo vi /etc/default/tftpd-hpa
```

Modify or add the following fields:

```
TFTP_DIRECTORY="<WORKDIR>/tftpboot"  
TFTP_OPTIONS="-l -c -s"
```

6.5.3. Restart TFTP Service

```
$ sudo service tftpd-hpa restart
```

Once the TFTP server is configured, place a test file "Image" in the configured "<WORKDIR>/tftpboot/" directory, and you can download and upload files using the TFTP client on the target device.

```
root@myir-remi-1g:/# tftp -g -r Image -l Image 192.168.1.168
```

The above command downloads the "Image" from the TFTP server "<WORKDIR>/tftpboot/" directory to the current directory of the target device.

```
root@myir-remi-1g:/# tftp -p -l config -r config_01 192.168.1.168
```

The above command will upload the "config" file from the current directory on the target device to the directory previously configured on the TFTP server, under "<WORKDIR>/tftpboot/" and rename it to "config_01".

6.6. DHCP

DHCP (Dynamic Host Configuration Protocol) is a network protocol for local area networks. The IP address range is controlled by the server. Clients can automatically obtain the IP address and subnet mask assigned by the server when they log in to the server.

DHCP also includes both server and client roles. In 4.1.1, we tested using DHCP client mode to automatically obtain IP addresses. This section describes how to use the `udhcpc` command to manually obtain the IP address for network debugging.

Connect the development board to the router using the CAT6 network cable, manually assign an IP address to the `eth0` network adapter, and observe the ip acquisition process by `dhcpc`.

- **Run the `udhcpc` command to configure the IP address**

```
root@myir:~# udhcpc -i eth0
root@myir-remi-1g:/# udhcpc -i eth0
udhcpc: started, v1.31.1
udhcpc: sending discover
udhcpc: sending select for 192.168.1.181
udhcpc: lease of 192.168.1.181 obtained, lease time 1800
root@myir-remi-1g:/# ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500  metric 1
    inet 192.168.1.181  netmask 255.255.255.0  broadcast 192.168.1.255
    inet6 fe80::c884:3eff:fe29:8f38  prefixlen 64  scopeid 0x20<link>
    ether ca:84:3e:29:8f:38  txqueuelen 1000  (Ethernet)
    RX packets 78870  bytes 30839989 (29.4 MiB)
    RX errors 0  dropped 162  overruns 0  frame 0
    TX packets 46165  bytes 2827556 (2.6 MiB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
    device interrupt 180
```

6.7. Iptables

Iptables is an administrative tool for IPv4 packet filtering and NAT. It is used to set up, maintain, and check the IP packet filtering rule table in the Linux kernel. Several different tables can be defined. Each table contains many built-in chains and can also contain user-defined chains. Each chain is a list of rules that can match a set of packets. Each rule specifies how to handle matching packets. Devices using Linux systems typically use the “iptables” tool to configure firewalls. “iptables” handles packets based on methods defined by packet filtering rules, such as accept, reject, and drop. Let's use “iptables” to test intercepting ICMP packets, preventing other devices on the network from ping them. Specific commands to use see: <https://linux.die.net/man/8/iptables>.

6.7.1. Configure development board iptables

Use the “iptables” configuration on the target device to discard the input ICMP package and not respond to ping probes from other hosts, with the following command:

```
root@myir-remi-1g:/# iptables -A INPUT -p icmp --icmp-type 8 -j DROP
[ 6036.476710] kauditd_printk_skb: 29 callbacks suppressed
[ 6036.476719] audit: type=1325 audit(3029536928.544:42): table=filter family=
2 entries=23 op=xt_replace pid=815 comm="iptables"
root@myir-remi-1g:/# iptables -S
-P INPUT ACCEPT
-P FORWARD DROP
-P OUTPUT ACCEPT
-N DOCKER
-N DOCKER-ISOLATION-STAGE-1
-N DOCKER-ISOLATION-STAGE-2
-N DOCKER-USER
-A INPUT -p icmp -m icmp --icmp-type 8 -j DROP
-A FORWARD -j DOCKER-USER
-A FORWARD -j DOCKER-ISOLATION-STAGE-1
```

```
-A FORWARD -o docker0 -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -o docker0 -j DOCKER
-A FORWARD -i docker0 ! -o docker0 -j ACCEPT
-A FORWARD -i docker0 -o docker0 -j ACCEPT
-A DOCKER-ISOLATION-STAGE-1 -i docker0 ! -o docker0 -j DOCKER-ISOLATION-STAGE-2
-A DOCKER-ISOLATION-STAGE-1 -j RETURN
-A DOCKER-ISOLATION-STAGE-2 -o docker0 -j DROP
-A DOCKER-ISOLATION-STAGE-2 -j RETURN
-A DOCKER-USER -j RETURN
```

6.7.2. Ping Test

Ping the target device on the development host and specifying a deadline of 10, the result is shown as follows:

```
hjk@ubuntu:~$ ping 192.168.1.181 -w 10
PING 192.168.1.181 (192.168.1.181) 56(84) bytes of data.

--- 192.168.1.181 ping statistics ---
10 packets transmitted, 0 received, 100% packet loss, time 9207ms
```

The above results show that the development host cannot ping through the development board after the firewall is set.

6.7.3. Delete Rules for iptables

```
root@myir-remi-1g:/# iptables -F
[ 6198.948807] audit: type=1325 audit(3029537091.016:43): table=filter family=2 entries=24 op=xt_replace pid=817 comm="iptables"
root@myir-remi-1g:/# iptables -S
-P INPUT ACCEPT
-P FORWARD DROP
-P OUTPUT ACCEPT
-N DOCKER
-N DOCKER-ISOLATION-STAGE-1
-N DOCKER-ISOLATION-STAGE-2
```

-N DOCKER-USER

6.7.4. Ping the Target Device Again

```
hix@ubuntu:~$ ping 192.168.1.181 -w 5
PING 192.168.1.181 (192.168.1.181) 56(84) bytes of data.
64 bytes from 192.168.1.181: icmp_seq=1 ttl=64 time=0.410 ms
64 bytes from 192.168.1.181: icmp_seq=2 ttl=64 time=0.373 ms
64 bytes from 192.168.1.181: icmp_seq=3 ttl=64 time=0.357 ms
64 bytes from 192.168.1.181: icmp_seq=4 ttl=64 time=0.721 ms
64 bytes from 192.168.1.181: icmp_seq=5 ttl=64 time=0.423 ms

--- 192.168.1.181 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4080ms
rtt min/avg/max/mdev = 0.357/0.456/0.721/0.136 ms
```

Once the “iptables” rules are cleared, ping the target device again from the development host is ready to ping. The above example is just a simple demonstration, but iptables with various rules can be very powerful, and I won't go into the details here.

6.8. Ethtool

ethtool is a tool used to view and modify Ethernet device parameters. It is useful in network debugging. Run the ethtool command to view Ethernet card information and modify its parameters. First, we use ethtool -h to view the help information for the command:

```
root@myir-remi-1g:/# ethtool -h
ethtool version 5.4
Usage:
    ethtool DEVNAME Display standard information about device
    ethtool -s|--change DEVNAME      Change generic options
        [ speed %d ]
        [ duplex half|full ]
        [ port tp|aui|bnc|mii|fibre ]
        [ mdix auto|on|off ]
        [ autoneg on|off ]
        [ advertise %x ]
        [ phyad %d ]
        [ xcvr internal|external ]
        [ wol p|u|m|b|a|g|s|f|d... ]
        [ sopass %x:%x:%x:%x:%x:%x ]
        [ msglvl %d | msglvl type on|off ... ]
    ethtool -a|--show-pause DEVNAME Show pause options
    ethtool -A|--pause DEVNAME      Set pause options
        [ autoneg on|off ]
```

View the basic information of the current device's Ethernet as follows:

```
root@myir-remi-1g:/# ethtool eth0
Settings for eth0:
    Supported ports: [ TP MII ]
    Supported link modes:   10baseT/Half 10baseT/Full
                           100baseT/Half 100baseT/Full
                           1000baseT/Full
```

Supported pause frame use: Symmetric Receive-only

Supports auto-negotiation: Yes

Supported FEC modes: Not reported

Advertised link modes: 10baseT/Half 10baseT/Full
100baseT/Half 100baseT/Full
1000baseT/Full

Advertised pause frame use: No

Advertised auto-negotiation: Yes

Advertised FEC modes: Not reported

Speed: 1000Mb/s

Duplex: Full

Port: Twisted Pair

PHYAD: 4

Transceiver: internal

Auto-negotiation: on

MDI-X: Unknown

Supports Wake-on: g

Wake-on: d

Current message level: 0x000000cc (204)
link timer rx_err tx_err

Link detected: yes

You can run the `ethtool` command to view that the Ethernet card supports six connection modes: 10 Mbit/s, 100 Mbit/s, and Gigabit half duplex and full duplex. The current connection status is negotiated gigabit, full duplex, MII interface, and PHY address 6.

We can also use `ethtool` to set Ethernet parameters, which are useful in Ethernet debugging and diagnosis. For example, we can force the Ethernet to be set to 100 Mbit full duplex and disable auto-negotiation. The commands are as follows:

```
root@myir-remi-1g:/# ethtool -s eth0 speed 100 duplex full autoneg off
```

More information about `ethtool` can be found here: <http://man7.org/linux/man-pages/man8/ethtool.8.html>.

6.9. iperf3

IPerf3 is a tool that actively measures the maximum realized bandwidth over an IP network. It supports tuning various parameters such as test time, buffer size, and protocols (TCP, UDP, SCTP with IPv4 and IPV6). The iPerf3 can be divided into server mode or client mode by role, and we can use it to test and view network bandwidth in TCP mode, TCP window values, probability of retransmission, etc., as well as to test packet loss rates, latency, and jitter under specified UDP bandwidth. Start by installing iperf3 on your server, which can be downloaded at <https://iperf.fr/iperf-download.php>, connect the server and development board directly through the CAT6 network cable, and configure their respective IP addresses. For example, set the ip address of the server to 192.168.1.188 and the IP address of the development board to 192.168.1.177, and use the ping command to test that they are connected.

Note: Do not connect routers or switches to prevent transmission and forwarding of intermediate devices from affecting test results.

6.9.1. Test Performance under TCP Mode

- **Server Side (192.168.1.38)**

iperf3 on the server uses the -s parameter to indicate that it works in server-side mode.

```
C:\Users\Administrator>iperf3 -s
```

```
-----  
Server listening on 5201  
-----
```

- **Client Side (192.168.1.112)**

The iperf3 program running on the development board works in client, TCP mode, where the parameters are described below:

- -c 192.168.1.38 : Server ip address
- -i 1 : The interval for reporting test results is 1 second
- -t 30 : The total test duration was 30 seconds

```
root@myir-remi-1g:~# iperf3 -c 192.168.1.38 -i 1 -t 30
```

Connecting to host 192.168.1.38, port 5201

[5] local 192.168.1.74 port 35236 connected to 192.168.1.38 port 5201

[ID]	Interval		Transfer	Bitrate	Retr	Cwnd	
[5]	0.00-1.00	sec	95.7 MBytes	802 Mb/s	0	221 KBytes	
[5]	1.00-2.00	sec	93.8 MBytes	787 Mb/s	0	221 KBytes	
[5]	2.00-3.01	sec	67.5 MBytes	561 Mb/s	0	221 KBytes	
[5]	3.01-4.00	sec	61.2 MBytes	517 Mb/s	0	221 KBytes	
[5]	4.00-5.00	sec	61.2 MBytes	515 Mb/s	0	221 KBytes	
[5]	5.00-6.02	sec	62.5 MBytes	516 Mb/s	0	221 KBytes	
[5]	6.02-7.01	sec	61.2 MBytes	515 Mb/s	0	221 KBytes	
[5]	7.01-8.01	sec	61.2 MBytes	516 Mb/s	0	221 KBytes	
[5]	8.01-9.01	sec	61.2 MBytes	515 Mb/s	0	221 KBytes	
[5]	9.01-10.00	sec	78.3 MBytes	661 Mb/s	0	221 KBytes	
[5]	10.00-11.01	sec	91.0 MBytes	760 Mb/s	0	221 KBytes	
[5]	11.01-12.00	sec	94.8 MBytes	796 Mb/s	0	221 KBytes	
[5]	12.00-13.00	sec	93.4 MBytes	786 Mb/s	0	221 KBytes	
[5]	13.00-14.00	sec	89.6 MBytes	753 Mb/s	0	221 KBytes	
[5]	14.00-15.00	sec	89.8 MBytes	750 Mb/s	0	221 KBytes	
[5]	15.00-16.00	sec	86.4 MBytes	727 Mb/s	0	221 KBytes	
[5]	16.00-17.00	sec	80.9 MBytes	679 Mb/s	0	221 KBytes	
[5]	17.00-18.00	sec	92.9 MBytes	780 Mb/s	0	221 KBytes	
[5]	18.00-19.00	sec	94.8 MBytes	794 Mb/s	0	221 KBytes	
[5]	19.00-20.00	sec	92.0 MBytes	771 Mb/s	0	221 KBytes	
[5]	20.00-21.00	sec	96.2 MBytes	807 Mb/s	0	221 KBytes	
[5]	21.00-22.00	sec	94.3 MBytes	790 Mb/s	0	221 KBytes	
[5]	22.00-23.01	sec	88.2 MBytes	739 Mb/s	0	221 KBytes	
[5]	23.01-24.00	sec	93.1 MBytes	781 Mb/s	0	221 KBytes	
[5]	24.00-25.01	sec	95.6 MBytes	797 Mb/s	0	221 KBytes	
^C[5]	25.01-25.96	sec	90.8 MBytes	807 Mb/s	0	221 KBytes	

[ID]	Interval		Transfer	Bitrate	Retr		
[5]	0.00-25.96	sec	2.12 GBytes	701 Mb/s	0		sender
[5]	0.00-25.96	sec	0.00 Bytes	0.00 bits/sec			receiver

After 10 seconds, the test is complete on the client and the above test result is displayed, indicating that the TCP bandwidth is about 819Mbps without retransmission.

At the same time, the server also displays the following test result and continues to listen on port 5201 to wait for the client connection:

```
C:\Users\Administrator>iperf3 -s

-----
Server listening on 5201
-----
Accepted connection from 192.168.1.112, port 51882
[ 5] local 192.168.1.188 port 5201 connected to 192.168.1.177 port 51884
[ ID] Interval           Transfer    Bandwidth
[ 5]  0.00-1.00   sec    90.8 MBytes    762 Mbits/sec
[ 5]  1.00-2.00   sec    99.6 MBytes    835 Mbits/sec
[ 5]  2.00-3.00   sec   101 MBytes    847 Mbits/sec
[ 5]  3.00-4.00   sec   96.2 MBytes    807 Mbits/sec
[ 5]  4.00-5.00   sec   102 MBytes    852 Mbits/sec
[ 5]  5.00-6.00   sec   95.4 MBytes    801 Mbits/sec
[ 5]  6.00-7.00   sec   101 MBytes    851 Mbits/sec
[ 5]  7.00-8.00   sec   100 MBytes    840 Mbits/sec
[ 5]  8.00-9.00   sec   96.7 MBytes    811 Mbits/sec
[ 5]  9.00-10.00  sec   99.2 MBytes    832 Mbits/sec
[ 5] 10.00-11.00  sec   98.4 MBytes    825 Mbits/sec
[ 5] 11.00-12.00  sec   101 MBytes    848 Mbits/sec
[ 5] 12.00-13.00  sec   102 MBytes    855 Mbits/sec
[ 5] 13.00-14.00  sec   96.9 MBytes    813 Mbits/sec
[ 5] 14.00-15.00  sec   99.1 MBytes    831 Mbits/sec
[ 5] 15.00-16.01  sec   98.6 MBytes    820 Mbits/sec
[ 5] 16.01-17.00  sec   100 MBytes    849 Mbits/sec
[ 5] 17.00-18.00  sec   98.5 MBytes    827 Mbits/sec
[ 5] 18.00-19.00  sec   96.9 MBytes    813 Mbits/sec
[ 5] 19.00-20.00  sec   102 MBytes    854 Mbits/sec
```

[5]	20.00-21.00	sec	93.4 MBytes	784 Mbits/sec	
[5]	21.00-22.00	sec	99.8 MBytes	837 Mbits/sec	
[5]	22.00-23.00	sec	96.0 MBytes	806 Mbits/sec	
[5]	23.00-24.00	sec	95.0 MBytes	797 Mbits/sec	
[5]	24.00-25.00	sec	101 MBytes	850 Mbits/sec	
[5]	25.00-26.00	sec	98.7 MBytes	828 Mbits/sec	
[5]	26.00-27.00	sec	98.7 MBytes	828 Mbits/sec	
[5]	27.00-28.00	sec	97.2 MBytes	816 Mbits/sec	
[5]	28.00-29.00	sec	96.5 MBytes	806 Mbits/sec	
[5]	29.00-30.00	sec	101 MBytes	848 Mbits/sec	
[5]	30.00-30.00	sec	413 KBytes	936 Mbits/sec	

[ID]	Interval		Transfer	Bandwidth	
[5]	0.00-30.00	sec	0.00 Bytes	0.00 bits/sec	sender
[5]	0.00-30.00	sec	2.88 GBytes	826 Mbits/sec	receiver

Server listening on 5201					

6.9.2. Test Performance under UDP Mode

● Server Side (192.168.1.38)

Continue running iPerf3 on the server and use the -s parameter to indicate that you are working in the server mode.

```
PC $ $ iperf3 -s
```

```
-----
Server listening on 5201
-----
```

● Client Side (192.168.1.112)

Iperf3 works on the device as client, under UDP mode, where the parameters are described below:

- -u : Works in UDP mode
- -c 192.168.1.38 : Server IP address

- -i 1 : The interval for reporting test results is 1 second
- -t 10 : The total test duration was 10 seconds
- -b 100M : Set the UDP transmission bandwidth to 100Mbps.

```
root@myir-remi-1g:~# iperf3 -u -c 192.168.1.38 -i 1 -t 10 -b 100M
Connecting to host 192.168.1.38, port 5201
[ 5] local 192.168.1.112 port 47296 connected to 192.168.1.38 port 5201
[ ID] Interval            Transfer        Bitrate          Total Datagrams
[ 5]  0.00-1.00    sec   11.9 MBytes    99.9 Mb/s       8555
[ 5]  1.00-2.00    sec   11.9 MBytes    100 Mb/s       8562
[ 5]  2.00-3.00    sec   11.9 MBytes    100 Mb/s       8562
[ 5]  3.00-4.00    sec   11.9 MBytes    100 Mb/s       8561
[ 5]  4.00-5.00    sec   11.9 MBytes    100 Mb/s       8562
[ 5]  5.00-6.00    sec   11.9 MBytes    100 Mb/s       8562
[ 5]  6.00-7.00    sec   11.9 MBytes    100 Mb/s       8562
[ 5]  7.00-8.00    sec   11.9 MBytes    100 Mb/s       8561
[ 5]  8.00-9.00    sec   11.9 MBytes    100 Mb/s       8562
[ 5]  9.00-10.00   sec   11.9 MBytes    100 Mb/s       8562
-----
[ ID] Interval            Transfer        Bitrate          Jitter    Lost/Total Datagrams
[ 5]  0.00-10.00   sec   119 MBytes     100 Mb/s        0.000 ms  0/85611
(0%) sender
[ 5]  0.00-10.00   sec   119 MBytes     99.5 Mb/s        0.055 ms  397/85611
(0.46%) receiver
```

After 10 seconds, the client completes the test and displays the above test result. At the same time, the server also displays the following test result and continues to listen on port 5201 to wait for the client connection:

```
C:\Users\Administrator>iperf3 -s
Accepted connection from 192.168.1.112, port 51886
[ 5] local 192.168.1.188 port 5201 connected to 192.168.1.112 port 47296
[ ID] Interval            Transfer        Bandwidth        Jitter    Lost/Total Datagrams
[ 5]  0.00-1.00    sec   11.9 MBytes    99.9 Mb/s         0.032 ms  0/8553 (0%)
```

[5]	1.00-2.00	sec	11.7 MBytes	98.4 Mbits/sec	0.046 ms	137/8559 (1.6%)
[5]	2.00-3.00	sec	11.9 MBytes	99.5 Mbits/sec	0.031 ms	43/8564 (0.5%)
[5]	3.00-4.00	sec	11.6 MBytes	97.5 Mbits/sec	0.043 ms	210/8560 (2.5%)
[5]	4.00-5.00	sec	11.9 MBytes	100 Mbits/sec	0.038 ms	0/8561 (0%)
[5]	5.00-6.00	sec	11.9 MBytes	100 Mbits/sec	0.041 ms	0/8560 (0%)
[5]	6.00-7.00	sec	11.9 MBytes	100 Mbits/sec	0.027 ms	0/8562 (0%)
[5]	7.00-8.00	sec	11.9 MBytes	100 Mbits/sec	0.037 ms	0/8564 (0%)
[5]	8.00-9.00	sec	11.9 MBytes	100 Mbits/sec	0.044 ms	0/8560 (0%)
[5]	9.00-10.00	sec	11.9 MBytes	99.9 Mbits/sec	0.038 ms	7/8563 (0.082%)
[5]	10.00-10.00	sec	7.13 KBytes	71.8 Mbits/sec	0.055 ms	0/5 (0%)

[ID]	Interval		Transfer	Bandwidth	Jitter	Lost/Total Datagrams
[5]	0.00-10.00	sec	0.00 Bytes	0.00 bits/sec	0.055 ms	397/85611 (0.46%)

The client modifies parameter -b to increase the specified UDP bandwidth. When packet loss occurs, the measured UDP bandwidth is the actual UDP bandwidth. There are many parameters that can be configured during the iperf3 test, so users can adjust the test according to the actual application requirements. For example, you can increase the value of the -t parameter for a long time stress test, or specify the -P parameter for a multiple connection concurrent stress test. Please refer to the iperf3 test for more information: <https://iperf.fr/iperf-doc.php#3doc>.

7. Graphic system

7.1. GPU

As mentioned earlier, the Remi Pi CPU integrates the 3D acceleration engine Max-G31 GPU, which supports OpenCL 2.0, OpenGL ES 1.1, 2.0, 3.0, 3.1, and 3.2. In this section, we do some evaluation of the GPU performance using KMcube and glmark2. glmark2 is an open source OpenGL 2.0 and ES 2.0 benchmark used to benchmark Gpus. glmark2 provides a rich set of tests covering different aspects of graphics unit performance (buffering, architecture, lighting, texture, etc.), allowing for more comprehensive and meaningful testing, with each test being run for 10 seconds and frame rates calculated separately.

7.1.1. Setting up the test environment

Set up the test environment, such as connecting the LVDS screen or HDMI, then power on and start the development board to enter the MEasy HMI display interface.

7.1.2. Start the kmscube test

```
root@myir-remi-1g:~# kmscube
Using display 0xaaab02d388a0 with EGL version 1.4
=====
EGL information:
  version: "1.4"
  vendor: "Mesa Project"
client extensions: "EGL_EXT_client_extensions EGL_EXT_device_base EGL_EXT_dev
ice_enumeration EGL_EXT_device_query EGL_EXT_platform_base EGL_KHR_client_
get_all_proc_addresses EGL_KHR_debug EGL_EXT_platform_device EGL_EXT_platf
orm_wayland EGL_KHR_platform_wayland EGL_MESA_platform_gbm EGL_KHR_pl
atform_gbm EGL_MESA_platform_surfaceless"
  display extensions: "EGL_ANDROID_blob_cache EGL_EXT_buffer_age EGL_EXT_i
mage_dma_buf_import EGL_EXT_image_dma_buf_import_modifiers EGL_KHR_cl_e
```

```
vent2 EGL_KHR_config_attribs EGL_KHR_create_context EGL_KHR_create_context_
no_error EGL_KHR_fence_sync EGL_KHR_get_all_proc_addresses EGL_KHR_gl_colo
rspace EGL_KHR_gl_renderbuffer_image EGL_KHR_gl_texture_2D_image EGL_KHR
_gl_texture_3D_image EGL_KHR_gl_texture_cubemap_image EGL_KHR_image EGL
_KHR_image_base EGL_KHR_image_pixmap EGL_KHR_no_config_context EGL_KH
R_partial_update EGL_KHR_reusable_sync EGL_KHR_surfaceless_context EGL_EXT_
pixel_format_float EGL_KHR_wait_sync EGL_MESA_configless_context EGL_MESA_
drm_image EGL_MESA_image_dma_buf_export EGL_MESA_query_driver EGL_WL_
bind_wayland_display "
```

```
=====
```

OpenGL ES 2.x information:

version: "OpenGL ES 3.1 Mesa 21.3.7"

shading language version: "OpenGL ES GLSL ES 3.10"

vendor: "Panfrost"

renderer: "Mali-G31 (Panfrost)"

extensions: "GL_EXT_blend_minmax GL_EXT_multi_draw_arrays GL_EXT_texture_fil
ter_anisotropic GL_EXT_texture_compression_rgtc GL_EXT_texture_format_BGRA8
888 GL_OES_compressed_ETC1_RGB8_texture GL_OES_depth24 GL_OES_element
index_uint GL_OES_fbo_render_mipmap GL_OES_mapbuffer GL_OES_rgb8_rgba8
GL_OES_standard_derivatives

```
.....
```

7.1.3. Start the glmark2 test

Before starting glmark2, you need to close the HMI display.

```
root@myir-remi-1g:~# killall mxapp2
```

```
root@myir-remi-1g:~# export PAN_MESA_DEBUG="noafbc"
```

```
root@myir-remi-1g:~# weston-start
```

```
root@myir-remi-1g:~# glmark2-es2-wayland
```

```
=====
```

```
glmark2 2017.07
```

```
=====
```

```
OpenGL Information
```

```
[ 432.835988] panfrost 11840000.gpu: js fault, js=0, status=DATA_INVALID_F  
AULT, head=0x63c8140, tail=0x63c8140
```

```
GL_VENDOR:   [ 432.846851] panfrost 11840000.gpu: gpu sched timeout, j
s=0, config=0x7300, status=0x58, head=0x63c8140, tail=0x63c8140, sched_job
=000000007111daa7
```

```
Panfrost
```

```
GL_RENDERER: Mali-G31 (Panfrost)
```

```
GL_VERSION:  OpenGL ES 3.1 Mesa 21.3.7
```

```
=====
==
```

```
[build] use-vbo=false: FPS: 136 FrameTime: 7.353 ms
```

```
[build] use-vbo=true: ^C FPS: 143 FrameTime: 6.993 ms
```

A description of some parameters:

- FPS: Frame rate is how many frames there are in 1s
- FrameTime: Represents the time between frames, and the smoother this parameter is, the smoother the overall picture. During the test, you can see the corresponding frame rate for each render scene and give a comprehensive score at the end.

7.2. QT

QT is a cross-platform C++ graphical user interface application development framework. It can be used to develop both GUI and non-GUI programs, such as console tools and servers. Qt is an object-oriented framework that uses special code generation extensions and macros. Qt is easy to extend and allows true component programming.

The development board burns systems with Qt runtime libraries at delivery and provides a rich HMI demo system.

7.2.1. Check Qt information

First check the QT versions supported by the current system, as follows:

```
root@myir-remi-1g:~# ls /usr/lib64/libQt5Core*  
/usr/lib64/libQt5Core.so    /usr/lib64/libQt5Core.so.5  
/usr/lib64/libQt5Core.so.5.15  /usr/lib64/libQt5Core.so.5.15.0
```

7.2.2. Introduction of QT Running Environment

When running Qt applications, the Qt operating environment, such as platform plug-ins, display parameters, input devices and cursor Pointers, can be properly configured according to different software and hardware requirements.

- **Platform Plugins Configuration**

On embedded Linux systems, multiple platform plug-ins can be used: EGLFS, LinuxFB, DirectFB, or Wayland. However, the availability of these plug-ins depends on the nature of the actual hardware platform and how Qt is configured. EGLFS is the default plug-in on many motherboards. If not, use the QT_QPA_PLATFORM environment variable to request another plug-in. Also, for quick tests, use -platform with command-line arguments that have the same syntax. For example, in the user console, you can configure it using the following command.

```
root@myir-remi-1g:~# export QT_QPA_PLATFORM=linuxfb  
root@myir-remi-1g:~# ./mxapp2
```

For modern embedded Linux devices that include Gpus, the EGLFS plug-in is recommended to use Gpus for rendering. If the embedded Linux device does not

contain a GPU or does not support OpenGL ES, the linuxfb platform plug-in can only be specified to render the software through the CPU.

NOTE: Starting with Qt 5.0, Qt no longer has its own Windows System (QWS) implementation and therefore no longer supports the -QWS parameter used on earlier versions of Qt.

● Display Parameters Configuration

QT applications can write applications that match the screen by using the QScreen class or QDesktopWidget to get the parameters associated with the screen display. Getting the screen resolution from a QScreen or QDesktopWidget is generally fine, but sometimes the physical size is not necessarily correct due to the display drive. At this point, you can configure and adjust the following parameters to make the elements displayed on the actual interface fit the size of the display screen. The plug-in writes directly to the frame buffer through Linux's fbdev subsystem. Only software rendered content is supported. Please note that display performance may be limited in some Settings. However, since fbdev is deprecated in the Linux kernel, DRM dumb caching support is also available starting with Qt 5.9. To use it, set the QT_QPA_FB_DRM environment variable to a non-zero value. Once set, /dev/fb0 does not access the old-style frame cache device as long as the system supports dummy caching. Instead, the rendering is set through the DRM API, similar to the backend in eglfs_kmsEGLFS. The output is double buffered and page flipped, also providing proper vertical synchronization for the content rendered by the software. Note: None of the options described below apply when using dummy buffers because properties such as physical and logical screen sizes are automatically queried. The linuxfb plug-in allows you to specify additional Settings through the QT_QPA_PLATFORM environment variable or the -platform command-line option. For example, QT_QPA_PLATFORM=linuxfb:fb=/dev/fb1 specifies that /dev/fb1 must use a framebuffer device instead of default fb0. To specify more than one setting, separate it with a colon (:).

Table 7-1. QT fb plug-in

Environment Variable	Environment Variable
fb=/dev/fbN	Specifies the frame buffer device. On multiple monitor

	Settings, this setting enables you to run the application on different monitors. Currently, multiple frame buffers cannot be used from a single Qt application
size= <width>x<height>	Specify the screen size in pixels. The plug-in attempts to query the physical and logical display size from the frame buffer device. However, this query may not always lead to the correct result. You may want to specify these values explicitly
mmsize= <width>x<height>	Specify the physical width and height in millimeters
offset= <width>x<height>	Specifies the upper-left offset of the screen in pixels. Default position (0,0)
nographicsmodeswitch	Specifies that the virtual terminal will not be switched to graphics mode (KD_GRAPHICS). Typically, enabling graphic mode disables the blinking cursor and screen whitespace. However, when this parameter is set, these two functions are skipped as well
tty=/dev/ttyN	Overwrites the virtual console. Only used if nographicsmodeswitch is not set

In general, the default configuration is fine, but if a display element does not match the actual screen, you can adjust the parameters as described above.

● Input Devices Configuration

If you want to enable tslib support, you need to set the QT_QPA_EGLFS_TSLIB (for eglfs) or QT_QPA_FB_TSLIB (for linuxfb) environment variable to 1. Refer to the specific use method of

tslib: <https://github.com/libts/tslib/blob/master/README.md>.

NOTE: The TSLIB input handler, often used for resistive touches, generates mouse events and supports only single touch, and requires screen calibration for initial use.

7.2.3. Start Qt Applications

Typically, devices that use touch do not need to display the mouse pointer. Users can hide the mouse pointer by setting environment variables before executing the application.

If you are using the “eglfs” platform plug-in, set it as follows:

```
export QT_QPA_EGLFS_HIDE_CURSOR=1
```

If you are using a “linuxfb” platform plug-in, the Settings are as follows:

```
export QT_QPA_FB_HIDE_CURSOR=1
```

The myir-image-full image produced by MYD-YG2LX development board comes with MEasy HMI demo program, which uses linuxfb platform plug-in and EvdevTouch Touch processor, corresponding startup command program is as follows:

```
root@myir-remi-1g:~# export QT_QPA_FB_HIDECURSOR=1
root@myir-remi-1g:~# ./mxapp2 -platform linuxfb &
qt.qpa.input: xkbcommon not available, not performing key mapping
qml: index=0
qml: currentIndex=0
qml: index=0
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
```

mxapp2 is started by default. For details, see the /usr/bin/start.sh script. If you want to run your own Qt application, you need to terminate mxapp2 before starting other applications.

You can directly exit a process by using the kill command

```
root@myir-remi-1g:~# killall mxapp2
```

8. Multimedia applications

8.1. VPU

The common encoding formats supported by VPU are 1080P HEVC, H.265, VP9, VP8, H.264, etc. Remi Pi has one VPU, which can be used for hardware encoding and decoding of video. The supported encoding and decoding format is H.264. In this section, we will use the gstreamer tool to view the file video information and play it to the screen to illustrate the decoding function of VPU.

8.1.1. View video format

Here, gst-discoverer-1.0 is used to view the video format.

```
root@myir-remi-1g:~# cd /usr/share/myir/Video/
root@myir-remi-1g:/usr/share/myir/Video# ls
video1.mp4  video2.mp4
root@myir-remi-1g:/usr/share/myir/Video# gst-discoverer-1.0 video1.mp4
Analyzing file:///usr/share/myir/Video/video1.mp4
Done discovering file:///usr/share/myir/Video/video1.mp4
```

Topology:

- container: Quicktime
- audio: MPEG-4 AAC
- video: H.264 (High Profile)

Properties:

Duration: 0:01:59.993333333

Seekable: yes

Live: no

Tags:

video codec: H.264 / AVC

language code: en

bitrate: 1947605

```
datetime: 2022-12-16T05:42:43Z
container format: ISO MP4/M4A
audio codec: MPEG-4 AAC audio
maximum bitrate: 192000
```

8.1.2. Play a Video

Run the `gst-play-1.0` command on the development board terminal to play the video.

```
root@myir-remi-1g:/usr/share/myir/Video# weston-start
root@myir-remi-1g:/usr/share/myir/Video# gst-play-1.0 video1.mp4
Press 'k' to see a list of keyboard shortcuts.
Now playing /usr/share/myir/Video/video1.mp4
Redistribute latency...
Redistribute latency...
Redistribute latency
```

8.2. Audio

This section is to test audio playback. There are two main ways: one is direct playback by HMI2.0 application; the other is manual playback by gstream tool.

8.2.1. HMI2.0 music playback

HMI multimedia audio playback has been integrated into the mxapp2 program in the myir-image-full system. The program has been automatically run by default upon startup, and the test only needs to be done by clicking the music button in the multimedia.

8.2.2. Play music manually

- **Volume control**

alsamixer is a graphical interface configuration tool used to configure various parameters of audio. You can set the output volume of the headset by using this command.

```
root@myir-remi-1g:~# alsamixer
```

- **Play music in wav format**

```
oot@myir-remi-1g:/usr/share/sounds/alsa# gst-play-1.0 Front_Center.wav
Press 'k' to see a list of keyboard shortcuts.
Now playing /usr/share/sounds/alsa/Front_Center.wav
Redistribute latency...
0:00:01.4 / 0:00:01.4
Reached end of play list.
```

- **Play mp3 format music**

In Linux, the MP3 music demo directory is /usr/share/myir/Music. Go to the directory to view the corresponding audio file and run the following command on the terminal to play the MP3 music:

```
root@myir-remi-1g:/usr/share/myir/Music# gst-play-1.0 The\ promise.mp3
Press 'k' to see a list of keyboard shortcuts.
Now playing /usr/share/myir/Music/The promise.mp3
Redistribute latency...
```

0:00:02.6 / 0:03:15.8

- **Recording**

```
root@myir-remi-1g:/mnt# arecord -f cd test.wav
Recording WAVE 'test.wav' : Signed 16 bit Little Endian, Rate 44100 Hz, Stereo
^CAborted by signal Interrupt..
```

- **Play the recording**

```
root@myir-remi-1g:/mnt# gst-play-1.0 test.wav
Press 'k' to see a list of keyboard shortcuts.
Now playing /mnt/test.wav
Redistribute latency...
0:00:37.4 / 0:00:37.4
```

8.3. Video

This section mainly tests two ways to play a video, one is to play a video by HMI click directly, the other is to play a video by command.

8.3.1. HMI2.0 video play

HMI multimedia video playback has been integrated into the mxapp2 program in the myir-image-full system. The program has been automatically run by default upon startup, and the test only needs to be done by clicking the player button in the multimedia

8.3.2. Manual video playback

In Linux, the video demo directory is /usr/share/myir/Video. Go to the directory to view the corresponding video file:

```
root@myir-remi-1g:~# cd /usr/share/myir/Video/  
root@myir-remi-1g:/usr/share/myir/Video# ls  
video1.mp4  video2.mp4  
root@myir-remi-1g:/usr/share/myir/Video#
```

Run the following command on the development board terminal to play the video:

```
root@myir-remi-1g:/usr/share/myir/Video# weston-start  
root@myir-remi-1g:/usr/share/myir/Video# gst-play-1.0 video1.mp4  
Press 'k' to see a list of keyboard shortcuts.  
Now playing /usr/share/myir/Video/video1.mp4  
Redistribute latency...  
Redistribute latency...  
Redistribute latency
```

9. System Tools

Some common system tools are included in the default image to make it easy for users to view and manage the various resources of the system during system debugging or actual deployment of the product, and can also be invoked in SHELL scripts or other applications. These tools may not fully meet users' system customization requirements, and system developers need to make appropriate adjustments based on the actual situation.

9.1. Compress and Decompress Tools

This section mainly tests the compression and decompression tools of the system. Compression is to compress multiple files into a compression package , then it will be convenient for file transfer. And decompression will make the compressed files back to the original size for easy use. This section illustrates the file system with tools such as "tar" , "gzip" , "gunzip" , and so on.

9.1.1. tar tool

The "tar" tool, now commonly used in Linux, not only packages files, but also compresses, views, adds, and unzips them. Here is the package operation.

- **Syntax Format**

Run the following command to view the tar syntax:

```
root@myir-remi-1g:/# tar --help
BusyBox v1.31.1 () multi-call binary.
```

```
Usage: tar c|x|t [-ZzJjahmvokO] [-f TARFILE] [-C DIR] [-T FILE] [-X FILE] [FILE]..
```

The detailed parameters are described below:

- -c : Create a compressed file parameter instruction
- -x : Unpacking a compressed file parameter instruction

- -t : Check out the files in the tarfile! In particular, only one c/x/t can exist in the sequence of parameters! Do not exist at the same time! Since it is impossible to compress and uncompress at the same time
- -z : Do you have gzip properties at the same time? Is gzip compression required?
- -j : Do you have a bzip2 attribute at the same time? Is bzip2 compression necessary?
- -v : Show files during compression! This is often used, but is not recommended when running in the background
- -f : Use file name, please note, the file name must be followed -f parameter immediately
- -p : Use the original properties of the original file (properties do not change depending on the user)
- -P : Can use absolute path to compress
- --exclude FILE: During compression, do not pack a FILE!

● Use tar to Compress

Create a new test.txt file and run the following command to package the file in.gz format:

```
root@myir-remi-1g:/mnt# touch test.txt
root@myir-remi-1g:/mnt# tar -zcvf test.tar.gz test.txt
root@myir-remi-1g:/mnt# ls
test.tar.gz  test.txt
```

With the z parameter, the ".tar.gz" or ".tgz" represents a gzip-compressed tar file.

● Use tar to Decompress

Unpack the file which has been compressed as tar.gz:

```
root@myir-remi-1g:/mnt# tar -xvf test.tar.gz
test.txt
root@myir-remi-1g:/mnt# ls
test.tar.gz  test.txt
```

9.1.2. gzip tool

"gzip" is a file compression and decompression command frequently used in Linux systems, both convenient and easy to use

- **Syntax Format**

Enter the following command on the development board terminal to view the gzip syntax:

```
root@myir-remi-1g:/mnt# gzip -help
gzip: invalid option -- 'h'
BusyBox v1.31.1 () multi-call binary.

Usage: gzip [-cfkdt] [FILE]...
```

- **Use gzip to Compress**

```
root@myir-remi-1g:/mnt# gzip test.txt
root@myir-remi-1g:/mnt# ls
test.tar.gz  test.txt.gz
```

- **Use gzip to Decompress**

```
root@myir-remi-1g:/mnt# gzip test.txt
root@myir-remi-1g:/mnt# ls
test.tar.gz  test.txt.gz
root@myir-remi-1g:/mnt# gunzip test.txt.gz
root@myir-remi-1g:/mnt# ls
test.tar.gz  test.txt
```

9.2. File System Tools

Test the file system tool of the system. This section describes several common file system management tools. The built-in file system tools mount, mkfs, fsck, and dumpe2fs.

9.2.1. mount tool

“mount” is a command under Linux that connects a partition to a folder on Linux, associating the partition with that directory, so simply accessing the folder is equivalent to accessing the partition, in the following syntax format:

```
root@myir-remi-1g:/# mount -h
```

Usage:

```
mount [-lhV]
```

```
mount -a [options]
```

```
mount [options] [--source] <source> | [--target] <directory>
```

```
mount [options] <source> <directory>
```

```
mount <operation> <mountpoint> [<target>]
```

Mount a filesystem.

Options:

-a, --all	mount all filesystems mentioned in fstab
-c, --no-canonicalize	don't canonicalize paths
-f, --fake	dry run; skip the mount(2) syscall
-F, --fork	fork off for each device (use with -a)
-T, --fstab <path>	alternative file to /etc/fstab
-i, --internal-only	don't call the mount.<type> helpers
-l, --show-labels	show also filesystem labels
-n, --no-mtab	don't write to /etc/mtab
--options-mode <mode>	what to do with options loaded from fstab
--options-source <source>	

.....

The following is an example of mounting the fourth partition of an sd card:

```
root@myir-remi-1g:/# mount /dev/mmcblk1p4 /mnt/
[ 1469.550504] EXT4-fs (mmcblk1p4): mounted filesystem with ordered data
mode. Opts: (null)
[ 1469.559617] ext4 filesystem being mounted at /mnt supports timestamps u
ntil 2038 (0x7fffffff)
root@myir-remi-1g:/# ls /mnt/
boot.scr.uimg  splash.bmp                                ulimage
lost+found     uinitrd  mmc0_extlinux  mmc1_extlinux
```

9.2.2. mkfs tool

After partitioning the hard disk, the next step is to set up the Linux file system. Similar to a formatted hard disk in Windows. Setting up a file system on a hard disk partition washes out the data on the partition and is not recoverable, so make sure that the data on the partition is no longer used before setting up the file system. The command to set up the file system is mkfs, in the following syntax format:

```
root@myir-remi-1g:/# mkfs -h
```

Usage:

```
mkfs [options] [-t <type>] [fs-options] <device> [<size>]
```

Make a Linux filesystem.

Options:

```
-t, --type=<type>  filesystem type; when unspecified, ext2 is used
fs-options         parameters for the real filesystem builder
<device>           path to the device to be used
<size>             number of blocks to be used on the device
-V, --verbose      explain what is being done;
                   specifying -V more than once will cause a dry-run
-h, --help         display this help
```

```
-V, --version      display version
```

For more details see mkfs(8).

Format the fourth partition of the sd card as follows:

```
root@myir-remi-1g:/# umount /mnt/
root@myir-remi-1g:/# mkfs -t ext3 -V -c /dev/mmcblk1p4
mkfs from util-linux 2.35.1
mkfs.ext3 -c /dev/mmcblk1p4
mke2fs 1.45.7 (28-Jan-2021)
/dev/mmcblk1p4 contains a ext4 file system labelled 'bootfs'
      last mounted on Fri Jan  1 00:24:26 2066
Proceed anyway? (y,N) y
Discarding device blocks: done
Creating filesystem with 65536 1k blocks and 16384 inodes
Filesystem UUID: 6f100745-0288-4aa2-8fcf-e476806485d7
Superblock backups stored on blocks:
    8193, 24577, 40961, 57345

Checking for bad blocks (read-only test): done

Allocating group tables: done
Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: done
```

9.2.3. fsck tool

The “fsck” command is mainly used to check the correctness of the file system. When the file system fails, the fsck instruction can be used to try to fix it. And fix the Linux disk. For example:

```
root@myir-remi-1g:/# fsck -a /dev/mmcblk1p4
fsck from util-linux 2.35.1
/dev/mmcblk1p4: clean, 11/16384 files, 7477/65536 blocks
```

9.2.4. dumpe2fs

Prints information about super blocks and blocks groups of existing file systems on a particular device. The development board enters the following command to see the application syntax:

```
root@myir-remi-1g:/# dumpe2fs -h
dumpe2fs 1.45.7 (28-Jan-2021)
Usage: dumpe2fs [-bfghimxV] [-o superblock=<num>] [-o blocksizes=<num>]
device
```

View detailed properties of the file system formatted, for example, by entering a command to view the details of a disk:

```
root@myir-remi-1g:/# dumpe2fs /dev/mmcb1k1p4
dumpe2fs 1.45.7 (28-Jan-2021)
Filesystem volume name:   <none>
Last mounted on:         <not available>
Filesystem UUID:         6f100745-0288-4aa2-8fcf-e476806485d7
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal ext_attr resize_inode dir_index filetype sp
arse_super large_file
Filesystem flags:         unsigned_directory_hash
Default mount options:    user_xattr acl
Filesystem state:         clean
Errors behavior:          Continue
Filesystem OS type:       Linux
.....
```

Check the number of inodes on a disk. Inodes also consume disk space. Therefore, when the disk is formatted, the operating system automatically divides the disk into two areas. One is the data area, store file data; The other is the inode table, which stores the information contained in the inode.

```
root@myir-remi-1g:/# dumpe2fs /dev/mmcb1k1p4 | grep -i "inode size"
dumpe2fs 1.45.7 (28-Jan-2021)
Inode size:                128
```

Look at the number of blocks on a disk. When the operating system reads the hard disk, it will not read sectors one by one, which is too inefficient. Instead, it

will read multiple sectors in a row at one time, that is, read a "block" at one time. This "block" of multiple sectors is the smallest unit of file access.

```
root@myir-remi-1g:/# dumpe2fs /dev/mmcblk1p4 | grep -i "block size"
dumpe2fs 1.45.7 (28-Jan-2021)
Block size:                1024
```

9.3. Disk Management Utils

This section describes several common disk management tools used to test the system. The system provides built-in disk management tools such as fdisk, dd, mkfs, du, and df. You can use these commands to monitor the disk usage.

9.3.1. fdisk tool

The fdisk disk partitioning tool has corresponding applications for DOS, Windows, and Linux. On Linux, fdisk is a menu-based command. To partition a hard disk using fdisk, you can directly add the hard disk to be partitioned after the fdisk command. The syntax is as follows:

```
root@myir:~# fdisk -h
root@myir-remi-1g:/# fdisk -h
```

Usage:

```
fdisk [options] <disk>      change partition table
fdisk [options] -l [<disk>] list partition table(s)
```

Display or manipulate a disk partition table.

Options:

```
-b, --sector-size <size>      physical and logical sector size
-B, --protect-boot            don't erase bootbits when creating a new label
-c, --compatibility[=<mode>]  mode is 'dos' or 'nondos' (default)
-L, --color[=<when>]          colorize output (auto, always or never)
```

.....

Partition the eMMC:

```
root@myir-remi-1g:/# fdisk /dev/mmcbk0
```

Welcome to fdisk (util-linux 2.35.1).

Changes will remain in memory only, until you decide to write them.

Be careful before using the write command.

Command (m for help)

9.3.2. dd tool

The “dd” command is used to copy the specified input file to the specified output file. And the format can be converted during replication. The difference between the “dd” command and the “cp” command is that the “dd” command can be done on a floppy disk that has not been created, and the data copied to the floppy disk is actually a mirror file. Similar to the “diskcopy” command in DOS. The format of the “dd” command is:

The following is an example of creating a 2M file:

```
root@myir-remi-1g:/# time dd if=/dev/zero of=ffmpeg1 bs=2M count=1 conv=fsync
1+0 records in
1+0 records out
2097152 bytes (2.1 MB, 2.0 MiB) copied, 0.0932943 s, 22.5 MB/s

real    0m0.103s
user    0m0.004s
sys     0m0.017s
```

9.3.3. du tool

The du command is used to display disk space usage. This command shows, level by level, how each level of subdirectory of the specified directory occupies a block of file system data. du is generally used in the following syntax:

```
root@myir-remi-1g:/# du --help
```

```
Usage: du [OPTION]... [FILE]...
```

```
or: du [OPTION]... --files0-from=F
```

Summarize disk usage of the set of FILEs, recursively for directories.

Mandatory arguments to long options are mandatory for short options too.

-0, --null	end each output line with NUL, not newline
-a, --all	write counts for all files, not just directories
--apparent-size	print apparent sizes, rather than disk usage; although the apparent size is usually smaller, it may be

larger due to holes in ('sparse') files, internal fragmentation, indirect blocks, and the like

Partial parameter are as follows:

- -a: Displays the size of all directories or files
- -h: Take K, M and G Bytes as units to improve the readability of information.
- -k: Output in Kilo Bytes
- -m: Output in Mega Bytes

Calculates the size of the file generated by the dd command:

```
root@myir-remi-1g:/# du ffmpeg1
2048    ffmpeg1
root@myir-remi-1g:/# du -h ffmpeg1
2.0M    ffmpeg1
```

9.3.4. df tool

Used to display disk usage statistics for the current file system on a Linux system, commonly used as follows:

```
root@myir-remi-1g:/# df --help
```

Usage: df [OPTION]... [FILE]...

Show information about the file system on which each FILE resides, or all file systems by default.

Mandatory arguments to long options are mandatory for short options too.

- a, --all include pseudo, duplicate, inaccessible file systems
- B, --block-size=SIZE scale sizes by SIZE before printing them; e.g., '-BM' prints sizes in units of 1,048,576 bytes; see SIZE format below
- h, --human-readable print sizes in powers of 1024 (e.g., 1023M)
- H, --si print sizes in powers of 1000 (e.g., 1.1G)
- i, --inodes list inode information instead of block usage
- k like --block-size=1K
- l, --local limit listing to local file systems

```
--no-sync          do not invoke sync before getting usage info (default)
--output[=FIELD_LIST] use the output format defined by FIELD_LIST,
                    or print all fields if FIELD_LIST is omitted.
-P, --portability   use the POSIX output format
.....
```

Partial parameter are as follows:

- -h: The appropriate units can be displayed based on the size used
- -i: View the number of inodes and inode usage under the partition
- -T: Print out the file system type

To see the number of inodes and inode usage under the partition, use the following command:

```
root@myir-remi-1g:/# df -i
Filesystem      Inodes IUsed   IFree IUse% Mounted on
/dev/root        1783616 28652 1754964    2% /
devtmpfs         188685  1449  187236    1% /dev
tmpfs            238141    1  238140    1% /dev/shm
tmpfs            238141  1428  236713    1% /run
tmpfs            238141   17  238124    1% /sys/fs/cgroup
tmpfs            238141   10  238131    1% /tmp
tmpfs            238141   20  238121    1% /var/volatile
tmpfs            238141    5  238136    1% /run/user/0
```

Inodes are allocated by the system when we format them. Inodes are related to disk partition size. When our inode usage has reached 100 percent, we cannot write data to disk even if we still have disk space left.

9.4. Process Management Utils

Process is also an important concept in the operating system, it is a process of the execution of a program, the program is a static description of the process, the system run every program is in its process run. All processes in Linux are interconnected, and all processes have a parent except for initializing the process. Instead of being created, new processes are copied or copied from previous processes. All processes in Linux are derived from an "init" process with process number 1. Linux consists of three different types of processes, each with its own characteristics and attributes:

- The Interaction Process: A process started by a Shell can run both in the foreground and in the background.
- The Batch Process: This process has no connection to the terminal and is a sequence of processes. This process is committed to a process waiting for execution in queue order.
- Monitor Processes (daemons): Daemons are always active and usually run in the background, and daemons are usually started automatically by the system at the beginning through script activation or root.
- For Linux system, process management is an important link, and process management is usually achieved through process management tools. There are several commonly used process management commands in Linux system: "ps" , "top" , "vmstat" , "kill" and so on.

9.4.1. ps tool

"ps" is a command to show the status of current processes. The general syntax is as follows:

```
root@myir-remi-1g:/# ps --help
```

Usage:

```
ps [options]
```

```
Try 'ps --help <simple|list|output|threads|misc|all>'
or 'ps --help <s|l|o|t|m|a>'
```

for additional help text.

For more details see `ps(1)`.

Partial parameter are as follows:

- -u: Organize a user-centric display of process status information.
- -a: A process that is not terminal dependent.
- -x: The process associated with the terminal.

Usually the above commands are combined with: `aux`

- -e: Show all processes; The equivalent of `ax`.
- -f: Displays full format program information.

Usually the above commands are combined with: `ef`

- -H: Shows the number of processes at the process level
- -F: Display more program information

Usually the above commands are combined with: `eHF`

An example of displaying information for all processes is shown below:

```
root@myir-remi-1g:/# ps aux
```

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.6	1.1	23804	4820	?	Ss	06:57	0:07	/sbin/init
root	2	0.0	0.0	0	0	?	S	06:57	0:00	[kthreadd]
root	3	0.0	0.0	0	0	?	I<	06:57	0:00	[rcu_gp]
root	4	0.0	0.0	0	0	?	I<	06:57	0:00	[rcu_par_gp]
root	8	0.0	0.0	0	0	?	I<	06:57	0:00	[mm_percpu_wq]
root	9	0.0	0.0	0	0	?	S	06:57	0:00	[ksoftirqd/0]
root	10	0.0	0.0	0	0	?	I	06:57	0:00	[rcu_preempt]
root	11	0.0	0.0	0	0	?	S	06:57	0:00	[migration/0]
root	12	0.0	0.0	0	0	?	S	06:57	0:00	[cpuhp/0]
root	13	0.0	0.0	0	0	?	S	06:57	0:00	[cpuhp/1]

A brief explanation of some of the taskbars above:

- VSZ: Virtual memory size
- RSS: Resident size
- STAT: There are several process states

- R: Running
- S: Interruptable sleeping
- D: Uninterruptable sleeping
- T: Stopped
- Z: Zombie process
- +: Foreground process
- N: Low priority process
- I: Multithreaded process

9.4.2. top tool

The “top” command puts quite a bit of overall system performance information on one screen. The display can also be changed in an interactive manner. Dynamic continuous monitoring of the running state of the process, the “top” syntax is generally as follows:

```
root@myir-remi-1g:/# top -help
```

```
procs-ng 3.3.16
```

Usage:

```
top -hv | -bcEHiOSs1 -d secs -n max -u|U user -p pid(s) -o field -w [cols]
```

The following is an example of dynamically viewing system processes:

```
root@myir-remi-1g:/#
```

```
top - 07:19:53 up 22 min, 1 user, load average: 0.00, 0.03, 0.05
```

```
Tasks: 109 total, 1 running, 108 sleeping, 0 stopped, 0 zombie
```

```
%Cpu(s): 0.0 us, 1.3 sy, 0.0 ni, 98.7 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
```

```
MiB Mem : 425.8 total, 255.1 free, 56.6 used, 114.1 buff/cache
```

```
MiB Swap: 0.0 total, 0.0 free, 0.0 used. 351.6 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
593	root	20	0	2348	1596	1056	S	1.3	0.4	0:25.27	apps.plugin
553	root	20	0	129684	14196	2076	S	0.7	3.3	0:13.86	netdata
594	root	20	0	2692	2140	1608	S	0.7	0.5	0:07.95	bash

9.4.3. vmstat tool

This command looks at the usage status of the memory space and gives you a snapshot of the performance of the entire system. “vmstat” runs in two modes: sample mode and average mode. If no parameters are specified, “vmstat”

statistics run in average mode and “vmstat” displays the average of all statistics since system startup. The common syntax and parameters are as follows:

```
root@myir-remi-1g:/# vmstat -h
```

Usage:

```
vmstat [options] [delay [count]]
```

Options:

```
-a, --active          active/inactive memory
-f, --forks           number of forks since boot
-m, --slabs           slabinfo
-n, --one-header      do not redisplay header
-s, --stats           event counter statistics
-d, --disk            disk statistics
-D, --disk-sum        summarize disk statistics
-p, --partition <dev> partition specific statistics
-S, --unit <char>    define display unit
-w, --wide            wide output
-t, --timestamp       show timestamp

-h, --help           display this help and exit
-V, --version        output version information and exit
```

For more details see vmstat(8).

“vmstat” runs in average mode, showing the average of all statistics since system startup:

```
root@myir-remi-1g:/# vmstat
```

```
procs -----memory----- ---swap-- -----io---- -system-- -----cpu-----
 r  b  swpd  free buff  cache   si   so    bi    bo    in   cs us sy id wa st
 0  0   0 200640 13556 131852  0   0  39   11  148  227  6  3 90  0  0
```

A brief explanation of some of the taskbars above:

- r: Number of processes currently running. Instead of waiting for I/O, these processes have ready numbers to run. Ideally, the number of processes that can run is equal to the number of CPU available
- b: The number of blocked processes waiting for I/O to complete
- forks: The number of times a new process was created
- in: The number of system interrupts
- cs: The number of times a context switch occurred in the system
- us: Percentage of total CPU time consumed by user processes
- sy: The percentage of total CPU time consumed by the system code, which includes time consumed in the System, IRQ, and Softirq states
- wa: The percentage of total CPU consumed waiting for I/O
- id: The percentage of total CPU time consumed by the system idle

Statistical system various data details are as follows:

```
root@myir-remi-1g:/# vmstat -s
1905128 K total memory
195628 K used memory
27992 K active memory
303284 K inactive memory
1483588 K free memory
6612 K buffer memory
219300 K swap cache
0 K total swap
0 K used swap
0 K free swap
261868 non-nice user cpu ticks
0 nice user cpu ticks
4107 system cpu ticks
236501 idle cpu ticks
6913 IO-wait cpu ticks
1494 IRQ cpu ticks
10184 softirq cpu ticks
0 stolen cpu ticks
2356789 pages paged in
```

```
14321 pages paged out
    0 pages swapped in
    0 pages swapped out
1730612 interrupts
1618189 CPU context switches
3029529597 boot time
    753 forks
```

A brief explanation of some of the information above:

- total memory: Total system memory
- used memory: Used memory
- CPU ticks: This shows the CPU time since the system started, where "tick" is a unit of time
- forks: Roughly speaking, this represents the number of new processes that have been created since system startup

vmstat provides a great deal of information about the performance of Linux systems. It is one of the core tools for investigating system problems.

9.4.4. kill tool

Sends the specified signal to the appropriate process. Not specifying the model sends SIGTERM (15) to terminate the specified process. If the program cannot be terminated with the available "-KILL" parameter, the signal it sends is SIGKILL(9), which forces the process to be terminated. The process number can be viewed using the "ps" command or the "jobs" command. A root user will affect the user's process, and a non-root user can only affect his or her own process. The general syntax of the kill command is as follows:

```
kill [ -s signal | -p ] [ -a ] pid ...
kill -l [ signal ]
```

Partial parameter are as follows:

- -s: Specifies the signal to send
- -p: simulate transmitting signal
- -l: Specifies the name list of signals
- pid: The ID number of the process to abort

➤ Signal: According to the signal

First use “ps -ef” and the pipe command to determine the PID to kill the process.

```
root@myir-remi-1g:/# ps -ef | grep mxapp2
root      349      1 99 00:00 ?        00:44:14 /home/root/mxapp2 -plat
form linuxfb
root      754      348  0 00:44 ttySC0   00:00:00 grep mxapp2
```

Then type the following command to terminate the process:

```
root@myir-remi-1g:/# kill -9 349
```

The killall command terminates all processes in the same process group, allowing you to specify the name of the process to be killed instead of the PID process number:

```
root@myir-remi-1g:/# killall mxapp2
```

10. Development support

This chapter mainly introduces some basic information about secondary development for the current SDK, which provides an image of myir-image-full, its application is the QT runtime library and the reference HMI interface. Users need to select the appropriate section for evaluation testing according to the CPU model of the development board. For information about this image, please refer to the 《Remi PI_SDK Release Notes》.

10.1. Development Language

10.1.1. SHELL

Shell is a program written in C language, which is a bridge for users to use Linux. Shell is both a command language and a programming language. There are many types of common Linux shells:

- Bourne Shell (/usr/bin/sh or /bin/sh)
- Bourne Again Shell (/bin/bash)
- C Shell (/usr/bin/csh)
- K Shell (/usr/bin/ksh)
- Shell for Root (/sbin/sh)

View the supported categories as follows:

```
root@myir-remi-1g:/# cat etc/shells
# /etc/shells: valid login shells
/bin/sh
/bin/bash
```

Write a shell script as follows:

```
root@myir-remi-1g:/# cat shell_demo.sh
#!/bin/bash
echo "myir test"
root@myir-remi-1g:/# sh shell_demo.sh
myir test
root@myir-remi-1g:/# bash shell_demo.sh
```

myir test

10.1.2. C/C++

C/C++ is the most commonly used programming language for low-level application development under Linux platform, and it is also the most efficient language next to assembly. Using C/C++ for development is usually the way of cross development, that is, the development is carried out on the development host side, compiled to generate the binary execution file running on the target machine, and then deployed to the target machine to run.

In this way, you first need to install the SDK built based on Yocto. Please refer to the "Remi Pi_ Software Development Guide" for installation steps. After installation, you need to configure the SDK environment, which can be imported according to your actual installation path, as follows:

```
hjk@myir-server:~$ source /opt/yg2lx_sdk/environment-setup-aarch64-poky-linux
hjk@myir-server:~$ $CC -v
Using built-in specs.
COLLECT_GCC=aarch64-poky-linux-gcc
.....
gcc version 8.3.0 (GCC)
```

This section demonstrates application development by writing a simple Hello World instance, and the following demo program "hello.c" is written on the development host side:

```
//file: hello.c
#include<stdio.h>
int main(int argc,char *argv[])
{
    printf("hello world!\n");
    return 0;
}
```

C++ prepared demo program hello.cpp

```
//file: hello.cpp
#include <iostream>
```

```
using namespace std;
int main(int argc,char *argv[])
{
    cout << "hello world!";
    return 0;
}
```

Then compile the application program, c language is compiled with \$CC, C ++ is compiled with \$CXX, because the corresponding header file and link are required during compilation, \$CC/\$CXX contains the corresponding system library and configuration information, if directly compile with aarch64-poky-linux-gcc, it will not find the header file. At this point, you can add the -v parameter to see the link process in detail.

```
$CC -v hello.c -o hello
or
$CXX -v hello.cpp -o hello-CXX
.....
```

Then copy the generated execution file to the target machine through the scp command. The result is as follows:

```
root@myir-remi-1g:/# ./hello
hello world!

root@myir-remi-1g:/# ./hello-CXX
hello world!
```

10.1.3. Python

Python is a high-level programming language with interpreted, object-oriented, dynamic data types. Python was invented by Guido van Rossum in late 1989, and the first public release was released in 1991. Like the Perl language, the Python source code is also licensed under the GPL(GNU General Public License). This section focuses on testing the use of Python, both from the Python command line and from the script.

- **Check the Supported Version of Python**

```
root@myir-remi-1g:~# python3 -V
```

Python 3.8.14

- **Python Interactive Mode**

Launch python and Enter the following text at the python prompt, then press Enter to see what happens:

```
root@myir-remi-1g:~# python3
Python 3.8.14 (default, Sep  6 2022, 21:10:26)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print ("Hello, Python!")
Hello, Python!
```

In Python 3.8.14, the output of the above example is as follows:

```
Hello, Python!
>>>
```

To exit the Python command line, run `exit()` to exit Python:

```
>>> exit()
root@myir-remi-1g:~#
```

- **Write a script to test Python**

Write a simple Python script, and all Python files will have the `.py` extension.

```
root@myir-remi-1g:~# vi test.py
root@myir-remi-1g:~# cat test.py
#!/usr/bin/python3
print ("Hello, Python!")
```

To execute the script file, the Python3 interpreter is in `/usr/bin` and executes the script using the following command

```
root@myir-remi-1g:~# chmod +x test.py
root@myir-remi-1g:~# ./test.py
Hello, Python!
```

Invoke the Python3 interpreter with the script parameters to start executing the script until the script completes. When the script is finished, the interpreter is no longer valid. The current system does not support the `pip` command and the `pip` installation python package. If the customer needs the `pip` tool, they need to port it by themselves.

10.2. Database

A Database is a warehouse that organizes, stores, and manages data in terms of data structures. There are many types of databases, commonly used databases are Access, Oracle, Mysql, SQL Server, SQLite and so on.

- **System SQLite**

SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes normal disk files directly. A complete SQL database containing multiple tables, indexes, triggers, and views is contained in a single disk file. This is a lightweight database, is ACID compliant associative database management system, it is designed for embedded, and has been used in many embedded products, it is very low resource consumption, in embedded systems, may only need a few hundred K of memory is enough. It runs faster than Mysql and PostgreSQL, and MYD-YG2LX uses SQLite3. Let's do a simple test.

Launch sqlite3 and create a new database. <testDB.db> , enter the following command on the terminal interface to enter the operation interface.

```
root@myir-remi-1g:~# sqlite3 testDB.db
SQLite version 3.31.1 2020-01-27 19:55:54
Enter ".help" for usage hints.
sqlite>
```

The above command will create a file testDB.db in the current directory, and then use SQLite's .databases command to check if it is in the database list, as shown below:

```
sqlite> .databases
main: /home/root/testDB.db
```

Use the .quit command to exit the sqlite prompt, as shown below:

```
sqlite> .quit
root@myir-remi-1g:~#
```

For more information about SQLite, please refer to the official website <https://www.sqlite.org/docs.html>.

11. Reference

- **Renesas development community**

<https://www.renesas.com/us/en>

- **Yoto Project BSP Development Guide**

<https://www.yoctoproject.org/docs/3.1.1/bsp-guide/bsp-guide.html>

- **Yocto Project Linux Kernel Development Guide**

<https://www.yoctoproject.org/docs/3.1.1/kernel-dev/kernel-dev.html>

- **Qt for Embedded Linux**

<https://doc.qt.io/qt-5/embedded-linux.html>

- **Wayland Architecture**

<https://wayland.freedesktop.org/architecture.html>

- **Systemd Network Configuration**

<https://www.freedesktop.org/software/systemd/man/systemd.network.html>

Appendix A

Warranty & Technical Support Services

MYIR Electronics Limited is a global provider of ARM hardware and software tools, design solutions for embedded applications. We support our customers in a wide range of services to accelerate your time to market.

MYIR is an ARM Connected Community Member and work closely with ARM and many semiconductor vendors. We sell products ranging from board level products such as development boards, single board computers and CPU modules to help with your evaluation, prototype, and system integration or creating your own applications. Our products are used widely in industrial control, medical devices, consumer electronic, telecommunication systems, Human Machine Interface (HMI) and more other embedded applications. MYIR has an experienced team and provides custom design services based on ARM processors to help customers make your idea a reality.

The contents below introduce to customers the warranty and technical support services provided by MYIR as well as the matters needing attention in using MYIR' s products.

Service Guarantee

MYIR regards the product quality as the life of an enterprise. We strictly check and control the core board design, the procurement of components, production control, product testing, packaging, shipping and other aspects and strive to provide products with best quality to customers. We believe that only quality products and excellent services can ensure the long-term cooperation and mutual benefit.

Price

MYIR insists on providing customers with the most valuable products. We do not pursue excess profits which we think only for short-time cooperation. Instead, we hope to establish long-term cooperation and win-win business with customers. So we will offer reasonable prices in the hope of making the business greater with the customers together hand in hand.

Delivery Time

MYIR will always keep a certain stock for its regular products. If your order quantity is less than the amount of inventory, the delivery time would be within three days; if your order quantity is greater than the number of inventory, the delivery time would be always four to six weeks. If for any urgent delivery, we can negotiate with customer and try to supply the goods in advance.

Technical Support

MYIR has a professional technical support team. Customer can contact us by email (support@myirtech.com), we will try to reply you within 48 hours. For mass production and customized products, we will specify person to follow the case and ensure the smooth production.

After-sale Service

MYIR offers one year free technical support and after-sales maintenance service from the purchase date. The service covers:

Technical support service

MYIR offers technical support for the hardware and software materials which have provided to customers;

- To help customers compile and run the source code we offer;
- To help customers solve problems occurred during operations if users follow the user manual documents;
- To judge whether the failure exists;
- To provide free software upgrading service.

However, the following situations are not included in the scope of our free technical support service:

- Hardware or software problems occurred during customers' own development;
- Problems occurred when customers compile or run the OS which is tailored by themselves;
- Problems occurred during customers' own applications development;
- Problems occurred during the modification of MYiR's software source code.

After-sales maintenance service

The products except LCD, which are not used properly, will take the twelve months free maintenance service since the purchase date. But following situations are not included in the scope of our free maintenance service:

- The warranty period is expired;
- The customer cannot provide proof-of-purchase or the product has no serial number;
- The customer has not followed the instruction of the manual which has caused the damage the product;
- Due to the natural disasters (unexpected matters), or natural attrition of the components, or unexpected matters leads the defects of appearance/function;
- Due to the power supply, bump, leaking of the roof, pets, moist, impurities into the boards, all those reasons which have caused the damage of the products or defects of appearance;
- Due to unauthorized weld or dismantle parts or repair the products which has caused the damage of the products or defects of appearance;
- Due to unauthorized installation of the software, system or incorrect configuration or computer virus which has caused the damage of products.

Warm tips

- MYiR does not supply maintenance service to LCD. We suggest the customer first check the LCD when receiving the goods. In case the LCD cannot run or no display, customer should contact MYiR within 7 business days from the moment get the goods.

- Please do not use finger nails or hard sharp object to touch the surface of the LCD.
- MYIR suggests user purchasing a piece of special wiper to wipe the LCD after long time use, please avoid clean the surface with fingers or hands to leave fingerprint.
- Do not clean the surface of the screen with chemicals.
- Please read through the product user manual before you using MYIR' s products.
- For any maintenance service, customers should communicate with MYIR to confirm the issue first. MYIR' s support team will judge the failure to see if the goods need to be returned for repair service, we will issue you RMA number for return maintenance service after confirmation.

Maintenance period and charges

- MYIR will test the products within three days after receipt of the returned goods and inform customer the testing result. Then we will arrange shipment within one week for the repaired goods to the customer. For any special failure, we will negotiate with customers to confirm the maintenance period.
- For products within warranty period and caused by quality problem, MYIR offers free maintenance service; for products within warranty period but out of free maintenance service scope, MYIR provides maintenance service but shall charge some basic material cost; for products out of warranty period, MYIR provides maintenance service but shall charge some basic material cost and handling fee.

Shipping cost

During the warranty period, the shipping cost which delivered to MYIR should be responsible by user; MYIR will pay for the return shipping cost to users when the product is repaired. If the warranty period is expired, all the shipping cost will be responsible by users.

Products Life Cycle

MYIR will always select mainstream chips for our design, thus to ensure at least ten years continuous supply; if meeting some main chip stopping production, we will inform customers in time and assist customers with products updating and upgrading.

Value-added Services

- MYIR provides services of driver development base on MYIR' s products, like serial port, USB, Ethernet, LCD, etc.
- MYIR provides the services of OS porting, BSP drivers ' development, API software development, etc.
- MYIR provides other products supporting services like power adapter, LCD panel, etc.
- ODM/OEM services.

MYIR Electronics Limited

Room 04, 6th Floor, Building No.2, Fada Road,
Yunli Inteiligent Park, Bantian, Longgang District.

Support Email: support@myirtech.com

Sales Email: sales@myirtech.com

Phone: +86-755-22984836

Fax: +86-755-25532724

Website: www.myirtech.com