SeawayHyper V1.0 测试报告

测试报告编号： SeawayHyper V1.0-20250829

测试对象：RK3588_SeawayHypervisor

测试日期：2025 年 08 月 29 日

测试人员：申雅静

中科海微（北京）科技有限公司

Seaway Technologies Co. Ltd

角色	姓名	签字	日期
测试工程师	申雅静	申雅静	2025.08.29
审核人	王义		
研发			
产品经理			
项目经理			

一、测试目的

验证 SeawayHyper V1.0 的基本功能，各外设接口的连接功能、数据传输稳定性是否符合设计要求，确保系统硬件与软件协同工作正常。

二、测试环境

类型	具体内容
硬件	RK3588 开发板、移远 4G/5G 模组、奥比中光深度相机、USB 摄像头、激光雷达、双目相机、相关连接线（USB、网线等）

软件	镜像大小：4.2G
	`[root@localhost pubimge]# ls -lh RK3588_NJ_20250826`
	`-rw-r--r--. 1 root root 4.2G Aug 26 16:40 RK3588_NJ_20250826`
	`[root@localhost pubimge]#`
	内存占用情况：
	`root@SeawayHyper:~# free -h`
	`total used free shared buff/cache available`
	`Mem: 7.0Gi 223Mi 6.5Gi 33Mi 273Mi 6.7Gi`
	`Swap: 0B 0B 0B`
	`root@SeawayHyper:~#`
	cpu 占用情况：
	`0[ 2.0%] Tasks: 44, 52 thr; 2 running`
	`1[ 100.0%] Load average: 1.16 0.85 0.43`
	`Mem[ 266M/6.98G] Uptime: 00:07:02`
	`Swp[ 0K/0K]`

三、测试范围

涵盖 SeawayHyper V1.0 系统，CAN 外设连接测试、IMU 连接测试、485 接口测试、USB - 串口（4G/5G 模块）测试、USB3.0 接口测试、USB-RTK 测试、网口 0 测试、网口 1 测试内容。

四、 测试记录

1. 嵌入式虚拟机监控器

1.1 CPU 虚拟化

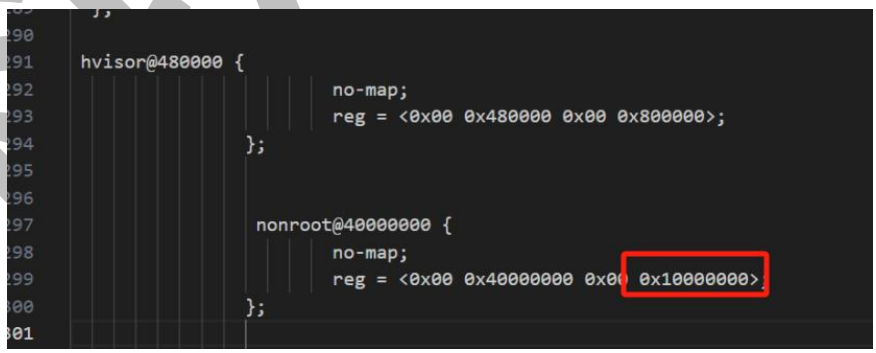
1.1.1 RM 处理器虚拟化

☒☒内容	ARM ☒理器虚☒化功能☒☒
☒☒前提	root linux 系☒正常运行
☒☒步☒	1. root linux 系☒正常运行； 2. ☒看当前系☒信息； cat /etc/os-release root@SeawayHyper:~/threevms# cat /etc/os-release PRETTY_NAME="Ubuntu 22.04.5 LTS" NAME="Ubuntu" VERSION_ID="22.04" VERSION="22.04.5 LTS (Jammy Jellyfish)" VERSION_CODENAME=jammy ID=ubuntu ID_LIKE=debian HOME_URL="https://www.ubuntu.com/" SUPPORT_URL="https://help.ubuntu.com/" BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/" PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy" UBUNTU_CODENAME=jammy root@SeawayHyper:~/threevms# 3. 启☒ rt-thread，☒看 rt-thread 运行情况； root@SeawayHyper:~/SeawayHyper# cd bin/ root@SeawayHyper:~/SeawayHyper/bin# ./hvisor zone list | zone_id | cpus | name | status | | 0 | 0, 1 | root-linux | running | | 1 | 6 | rtthread | running | root@SeawayHyper:~/SeawayHyper/bin#
☒期☒果	系☒正常运行，系☒版本信息准确。

☒☒☒果	pass

1.2内存虚拟化

1.2.1 静态内存分配管理

☒☒内容	静☒内存分配管理功能☒☒
☒☒前提	Hypervisor 与 Guest 能正常运行
☒☒步☒	1. 配置并准☒不同内存☒格的 Guest VM： 配置 Guest A 的☒☒☒或虚☒机配置文件：将其静☒内存大小☒定☒ 256MB  配置 Guest B 的☒☒☒或虚☒机配置文件：将其静☒内存大小☒定☒ 1GB。

```
hvisor@480000 {
                no-map;
                reg = <0x00 0x480000 0x00 0x800000>;
        };

        nonroot@40000000 {
                no-map;
                reg = <0x00 0x40000000 0x00 0x10000000>;
        };
```

```
hvisor@480000 {
                no-map;
                reg = <0x00 0x480000 0x00 0x800000>;
        };

        nonroot@40000000 {
                no-map;
                reg = <0x00 0x40000000 0x00 0x40000000>;
        };
```

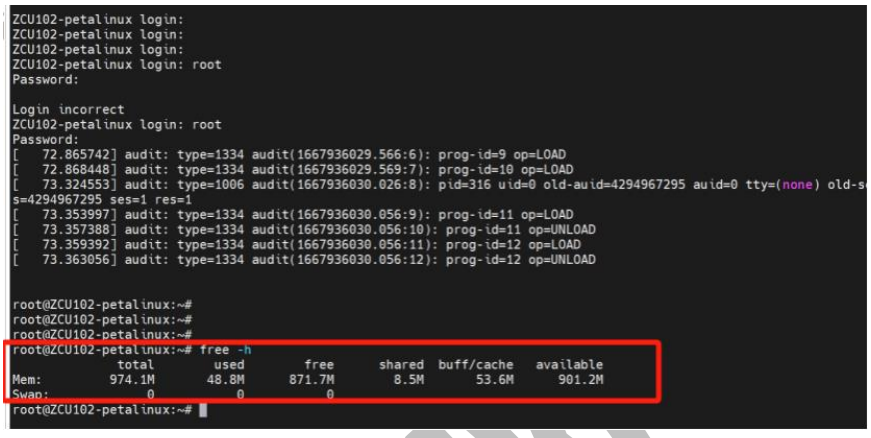
2. 利用 Hypervisor 启⊠ Guest VM 并⊠⊠内存分配；

分别启⊠配置了不同内存大小的 Guest A 和 Guest B。

在 每个 Guest VM 内部启⊠后，⊠行内存⊠看命令（ Linux 系⊠中的 free -h ），确⊠其⊠告的可用内存大小与⊠⊠⊠中静⊠配置的大小（256MB 或 1GB）一致。

⊠期⊠果	Hypervisor ⊠能准确地根据⊠⊠配置，⊠不同内存大小的虚⊠机分配相⊠的内存，并在虚⊠机内部正确⊠示⊠些已分配的内存。
⊠⊠⊠果	⊠⊠⊠，Hypervisor 成功地⊠虚⊠机分配了⊠⊠的不同内存大小，且虚⊠机内部⊠告的内存与配置一致，功能符合⊠期。

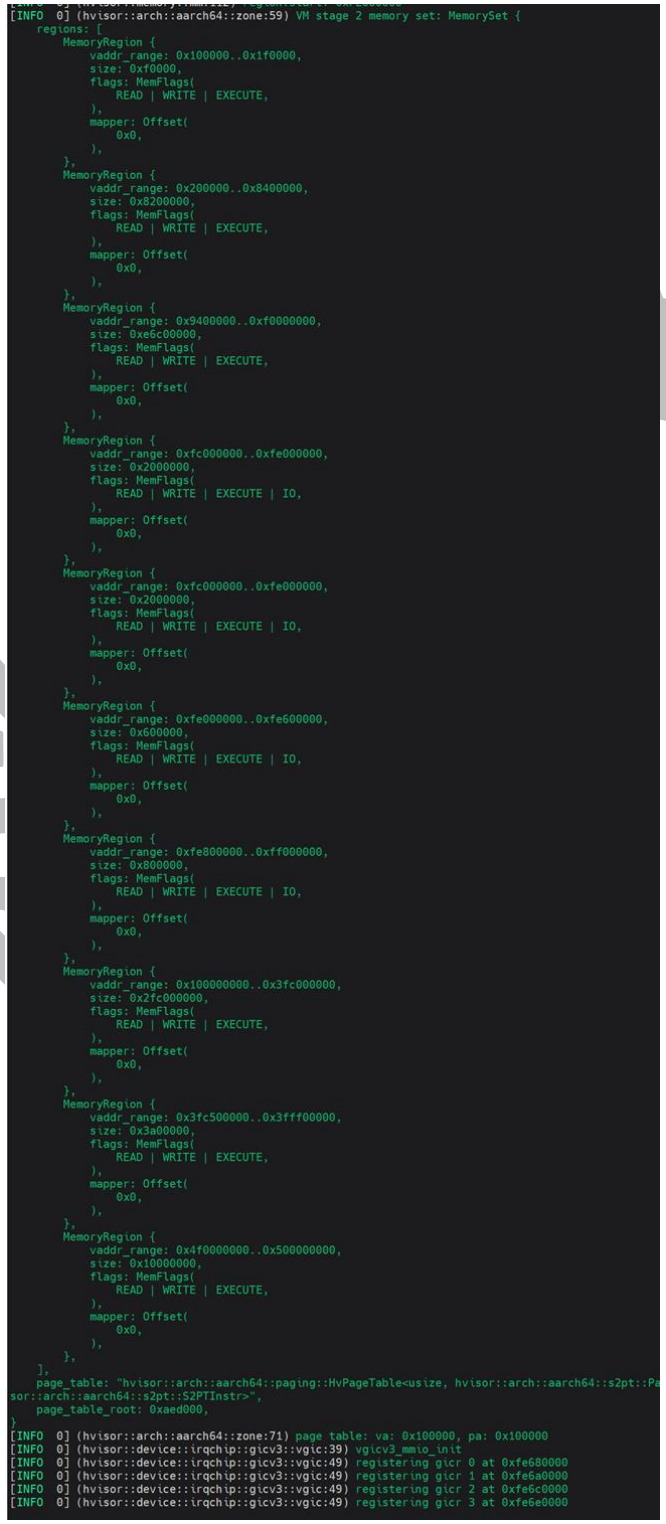
```
ZCU102-petalinux login:
ZCU102-petalinux login:
ZCU102-petalinux login:
ZCU102-petalinux login:
ZCU102-petalinux login: root
Password:
[  342.111027] audit: type=1334 audit(1667933974.392:6): prog-id=9 op=LOAD
[  342.114435] audit: type=1334 audit(1667933974.399:7): prog-id=10 op=LOAD
[  342.336176] audit: type=1006 audit(1667933974.619:8): pid=315 uid=0 old-auid=4294967295 auid=0 tty=(none) old-se
s=4294967295 ses=1 res=1
[  342.353046] audit: type=1334 audit(1667933974.635:9): prog-id=11 op=LOAD
[  342.354382] audit: type=1334 audit(1667933974.639:10): prog-id=11 op=UNLOAD
[  342.355453] audit: type=1334 audit(1667933974.639:11): prog-id=12 op=LOAD
[  342.356472] audit: type=1334 audit(1667933974.639:12): prog-id=12 op=UNLOAD
root@ZCU102-petalinux:~#
root@ZCU102-petalinux:~#
root@ZCU102-petalinux:~#
root@ZCU102-petalinux:~# free -h
               total        used        free      shared  buff/cache   available
Mem:           220.7M       51.0M      116.0M       8.5M       53.7M      153.9M
Swap:             0B          0B          0B
root@ZCU102-petalinux:~#
```

1.2.2 stage-2 页表管理

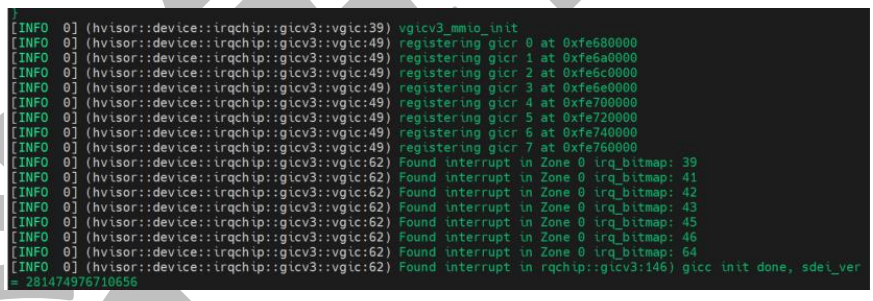
☒☒内容	stage-2 ☒表管理功能☒☒
☒☒前提	Hypervisor 已成功建立基☒的 Stage-1 ☒表映射
☒☒步☒	1. 初始化并配置 Hypervisor 的 Stage-2 ☒表。 2. ☒☒一个 Guest 物理地址在 Stage-2 ☒表中的映射☒果。 3. ☒☒☒☒到的宿主机物理地址与☒期☒是否一致。
☒期☒果	☒☒☒果☒☒示，所☒ Guest 物理地址通☒ Stage-2 ☒表正确映射到其☒期的宿主机物理地址。

☒☒☒果	☒☒☒，Hypervisor 的 Stage-2 ☒表功能正常，所☒☒的 Guest 物理地址被准确地映射到其☒期的宿主机物理地址，符合☒期。



1.3 中断虚拟化

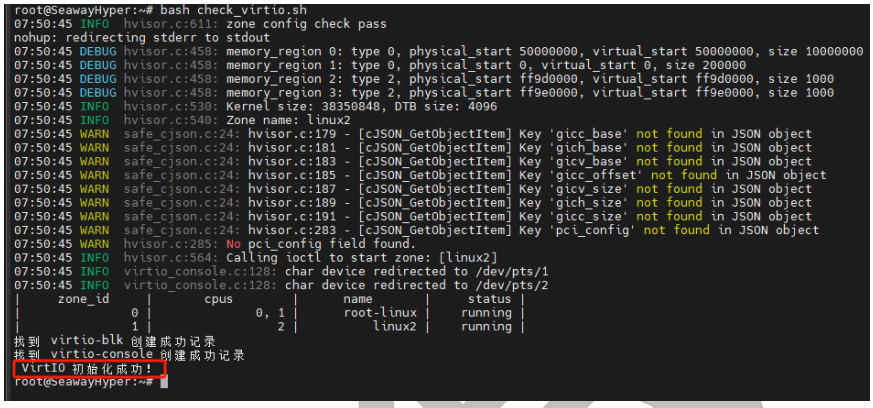
1.3.1 ARM 中断控制器 GICv3

☒☒内容	ARM 中断控制器 GICv3 功能☒☒
☒☒前提	Hypervisor 已☒☒并启用☒ GICv3 虚☒化特性的支持
☒☒步☒	在 Hypervisor 中初始化并配置虚☒ GICv3，并打印相关 log 信息
☒期☒果	☒在 Hypervisor 的 log 中看到 GICv3 的 log 信息
☒☒☒果	

1.4 VirtIO 支持

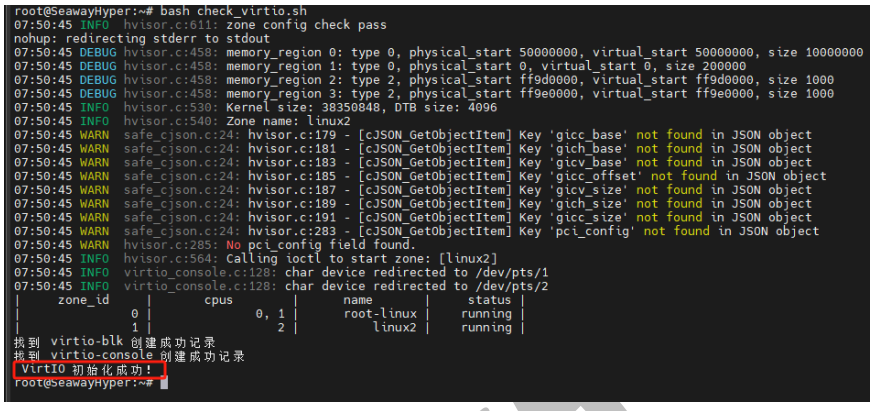
1.4.1 VirtIO – Console

☒☒内容	VirtIO – Console 功能☒☒
☒☒前提	1. root linux 正常运行； 2. 存在文件 check_virtio.sh

☒☒步☒	1. 运行 check_virtio.sh 
☒期☒果	VirtIO – Console 初始化成功
☒☒☒果	pass

1.4.2 VirtIO – Block

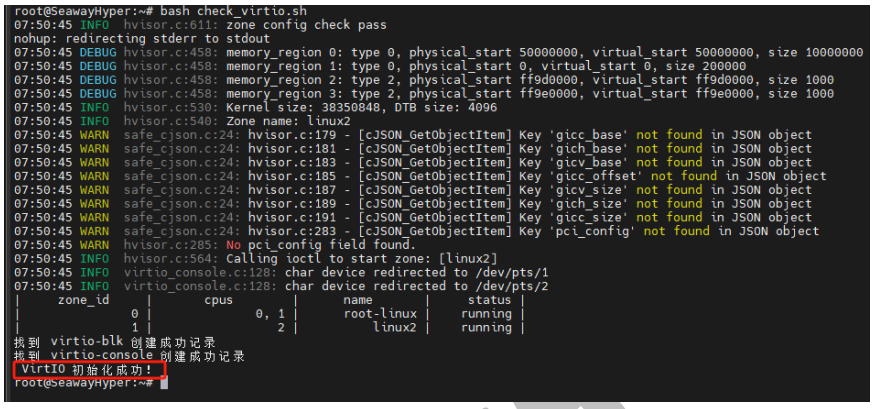
☒☒内容	VirtIO – Block 功能☒☒
☒☒前提	1. root linux 正常运行； 2. 存在文件 check_virtio.sh
☒☒步☒	1. 运行 check_virtio.sh

	
☒期☒果	VirtIO – Block 初始化成功
☒☒☒果	pass

1.5 Hypercall 接口

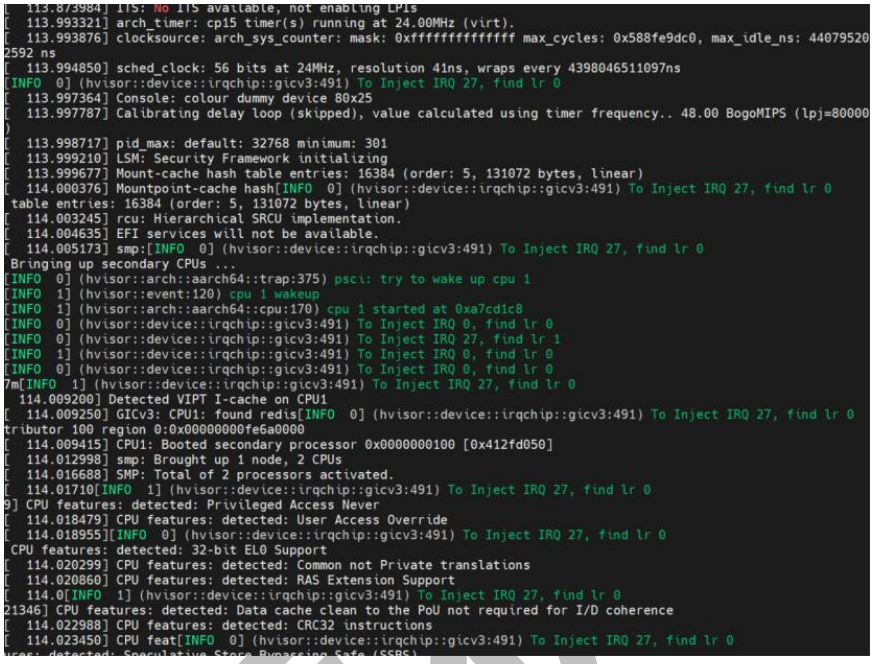
1.5.1 VirtIO 初始化

☒☒内容	VirtIO 初始化功能☒☒
☒☒前提	1. root linux 正常运行； 2. 存在文件 check_virtio.sh
☒☒步☒	1. 运行 check_virtio.sh

	
☒期☒果	☒出 VirtIO 初始化成功
☒☒☒果	pass

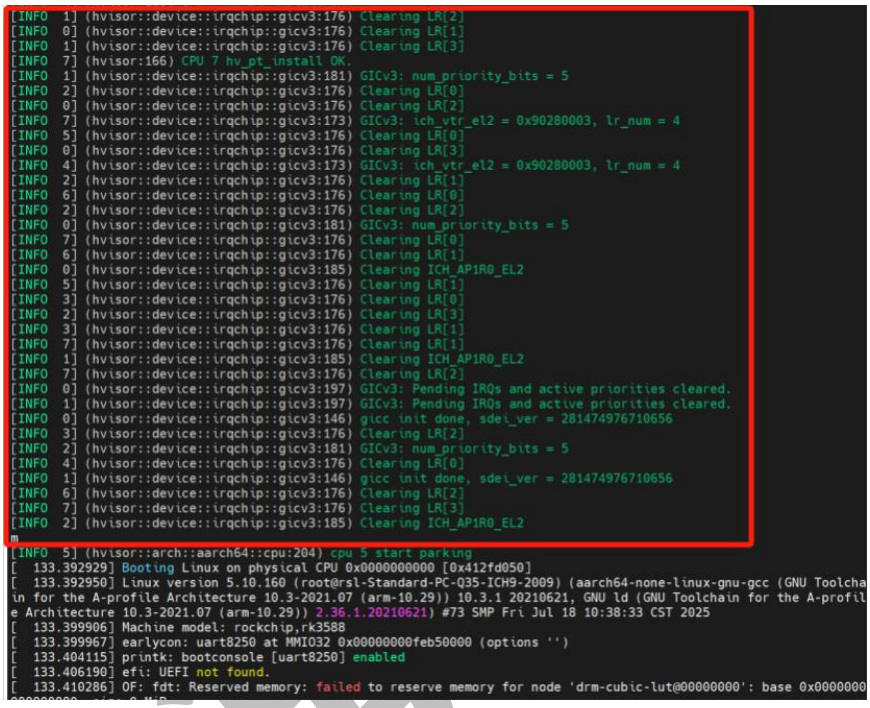
1.5.2 注入中断

☒☒内容	中断注入功能☒☒
☒☒前提	Hypervisr 与 Guest 正常运行
☒☒步☒	通☒ Hypervisor 向 Guest VM 注入☒特定中断并☒☒注入信息。
☒期☒果	☒在 Hypervisor 的 log 中看到中断注入的 log 信息
☒☒☒果	能正常看☒注入信息

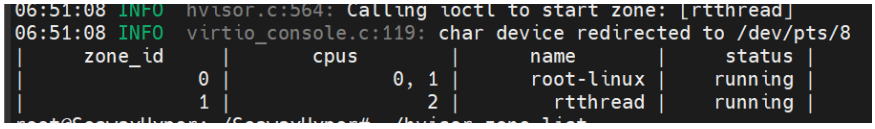
	

1.5.3 清除注入中断

☒☒内容	清除中断注入功能☒☒
☒☒前提	Hypervisor 与 Guest 注入中断功能正常
☒☒步☒	通☒ Hypervisor 清除特定中断并☒☒清除注入信息
☒期☒果	☒在 Hypervisor 的 log 中看到清除注入中断的 log

☒☒☒果	能正常看☒清除注入打印 log 

1.5.4 启动虚拟机

☒☒内容	rt-thread 启☒正常☒☒
☒☒前提	root linux 正常运行
☒☒步☒	参考《混合部署系☒启☒方式》
☒期☒果	./guest1.sh 
☒☒☒果	pass

1.5.5 关闭虚拟机

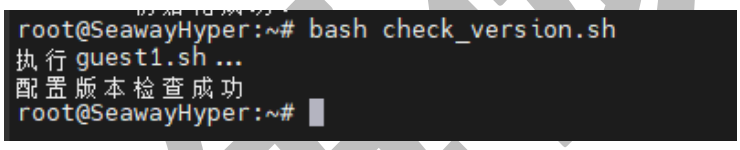
☒☒内容	rt-thread 关☒功能☒☒
☒☒前提	root linux、 rt-thread 正常运行
☒☒步☒	root Linux ☒行以下命令： ./hvisor zone shutdown -id <zone_id> root@SeawayHyper:~/SeawayHyper/bin# ./hvisor zone shutdown -id 1 root@SeawayHyper:~/SeawayHyper/bin# ./hvisor zone list | zone_id | cpus | name | status | | 0 | 0, 1 | root-linux | running | root@SeawayHyper:~/SeawayHyper/bin#
☒期☒果	虚☒机关☒正常
☒☒☒果	pass

1.5.6 虚拟机列表展示

☒☒内容	root linux、 rt-thread 运行状☒正常展示
☒☒前提	root linux、 rt-thread 正常运行
☒☒步☒	root Linux 运行以下命令： ./hvisor zone list root@SeawayHyper:~/SeawayHyper# cd bin/ root@SeawayHyper:~/SeawayHyper/bin# ./hvisor zone list | zone_id | cpus | name | status | | 0 | 0, 1 | root-linux | running | | 1 | 2 | rtthread | running | root@SeawayHyper:~/SeawayHyper/bin#
☒期☒果	虚☒机列表展示状☒正常

☒☒☒果	pass

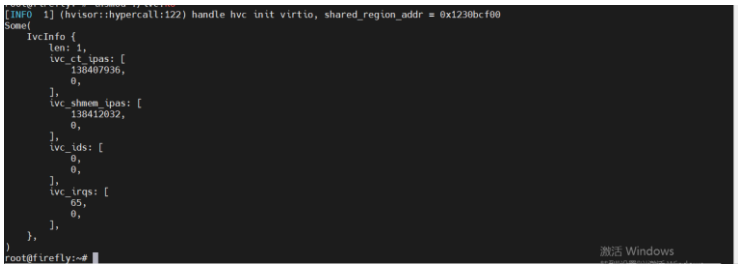
1.5.7 配置版本检查

☒☒内容	配置版本☒☒☒☒
☒☒前提	1. root linux 正常运行； 2. 存在文件 check_version.sh
☒☒步☒	1. 运行 check_version.sh 
☒期☒果	配置版本☒☒成功
☒☒☒果	pass

1.6IVC 机制

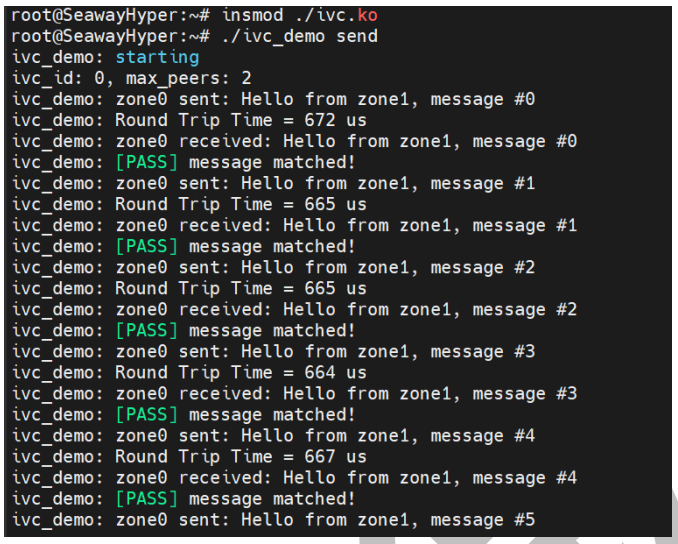
1.6.1 定制 IVC 内核模块

☒☒内容	定制 IVC 内核模☒☒☒
☒☒前提	root linux 正常运行
☒☒步☒	1. insmod ivc.ko 会☒出以下配置信息

	
☒期☒果	ivc 模☒成功加☒
☒☒☒果	pass

1.6.2 VM 调用其中的 IVC 服务进行通信

☒☒内容	☒用 IVC 服☒☒行通信☒☒
☒☒前提	1. root Linux 正常运行； 2. rt-thread 运行
☒☒步☒	1. 加☒ ivc.ko 模☒； 2. 运行当前目☒下的/root/clktool 的 guest.sh 文件 2. ☒行./ivc_demo send 命令。

	
☒期☒果	Root linux 、rt-thread 收☒数据成功
☒☒☒果	pass

2. 管理域操作系统镜像（ VM0 Root Linux）

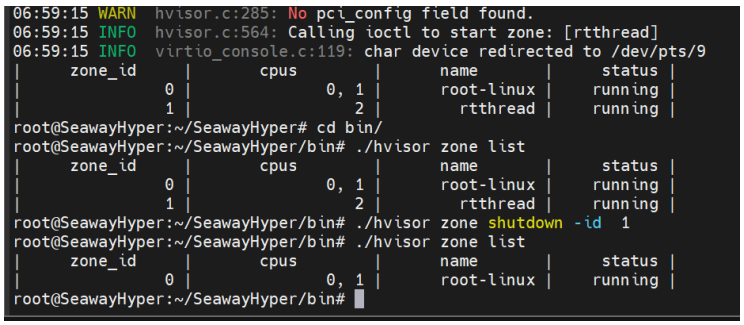
2.1VM 全生命周期管理工具集 SeawayHyper Tools

2.2.1 VM 创建与配置

☒☒内容	VM ☒建与配置功能☒☒
☒☒前提	Root linux 系☒正常运行
☒☒步☒	参考《混合部署系☒启☒☒明☒》，☒建并启☒ rt-thread 系☒。

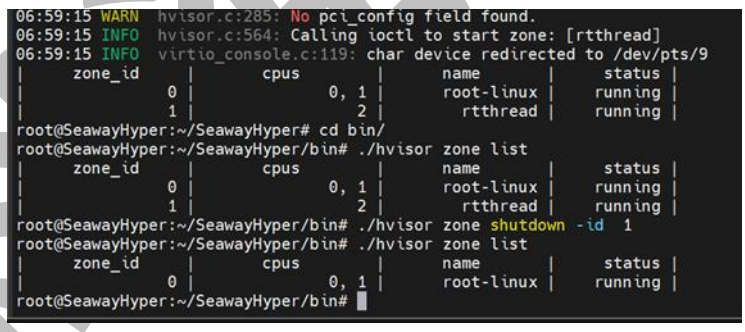
	
☒期☒果	rt-thread 成功☒建并且正常运行
☒☒☒果	pass

2.2.2 VM 启动与停止

☒☒内容	VM 启☒与停止功能☒☒
☒☒前提	Root linux 系☒正常运行
☒☒步☒	参考《混合部署系☒启☒☒明☒》启☒ vm  ./hvisor zone shutdown -id <zone_id> 

☒期☒果	rt-thread 正常启☒、停止
☒☒☒果	pass

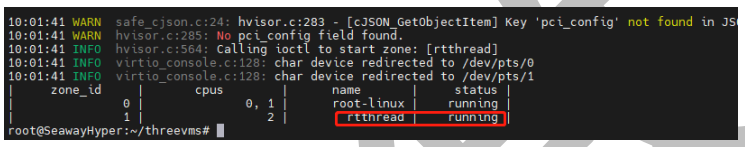
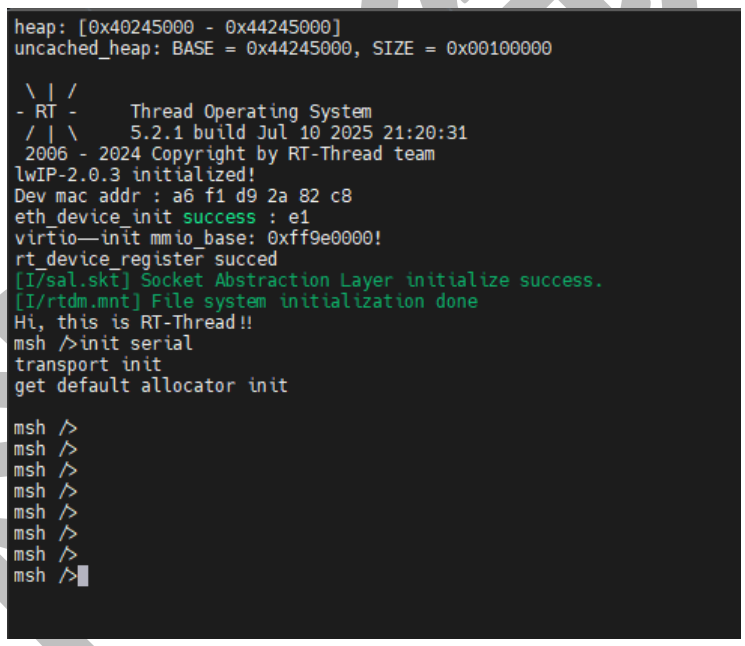
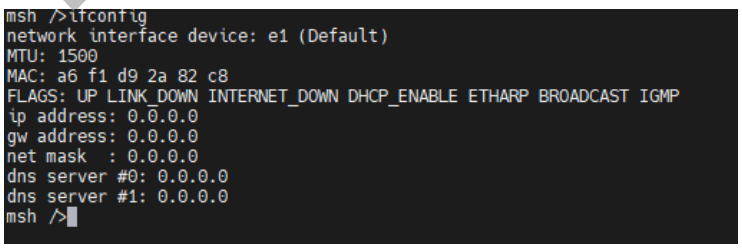
2.2.3 VM 删除与清理

☒☒内容	rt-thread ☒除与清理功能☒☒
☒☒前提	Root linux 系☒正常运行
☒☒步☒	./hvisor zone shutdown -id <zone_id>停止 
☒期☒果	rt-thread 成功☒除
☒☒☒果	pass

3. 实时嵌入式负载域操作系统镜像支持（VM1）

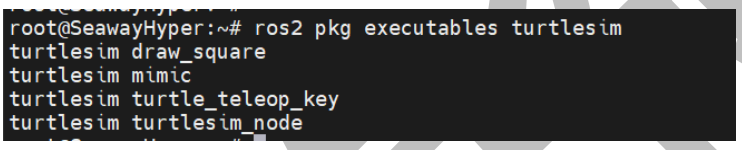
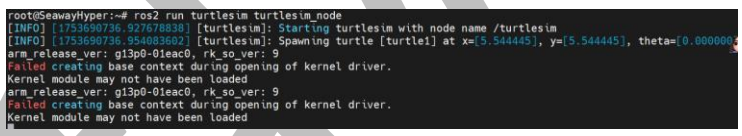
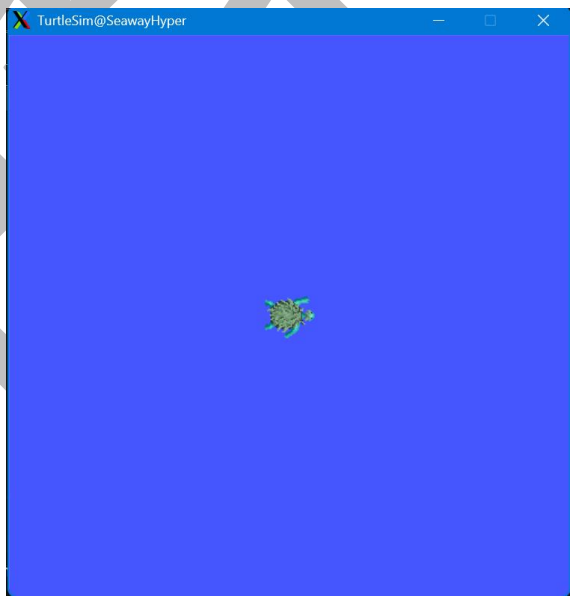
3.1支持 RTOS VM

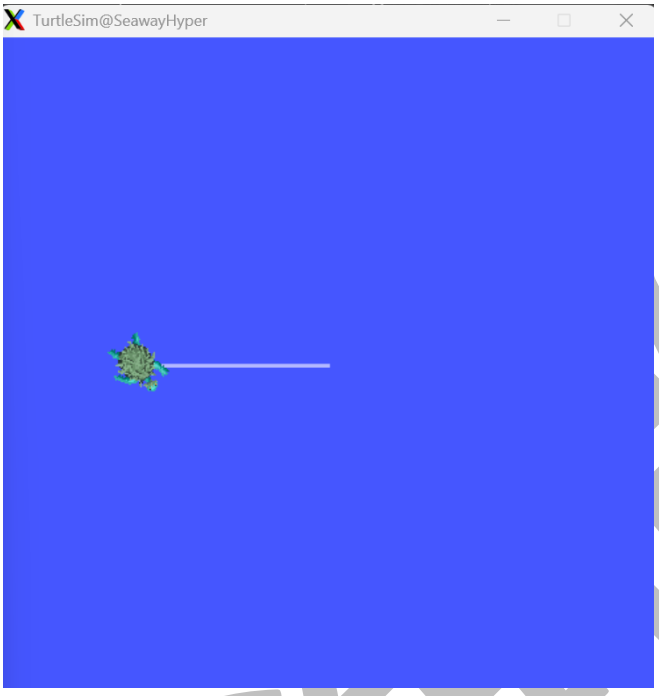
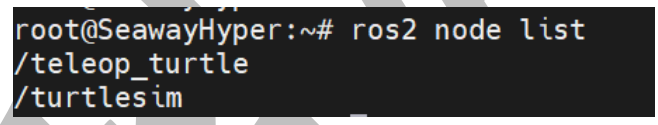
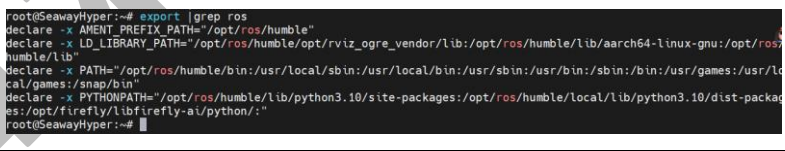
3.1.1 RTOS 的启动和运行

☒☒内容	RTOS 的启☒和运行功能☒☒
☒☒前提	Root linux 正常运行
☒☒步☒	参考《混合部署系☒启☒☒明☒》   
☒期☒果	RTOS 正常启☒和运行
☒☒☒果	pass

4 ros2 基本功能

4.1 ros2 基本功能

测试内容	验证 ros2 安装是否成功,基本功能是否正常。
测试前提	root linux 已经安装 ROS2
测试步骤	1. 检查软件包 ros2 pkg executables turtlesim  2. ros2 run 运行 turtlesim 打开一个终端，执行以下命令 ros2 run turtlesim turtlesim_node，会出现一个模拟窗口，窗口中间会有一个随机小乌龟；   再打开一个终端,运行如下命令,ros2 run turtlesim turtle_teleop_key，主要是用键盘控制小乌龟的运动。界面一定是在这个终端下，键盘才能控

	制小乌龟。  3. 保持上一个终端中 turtlesim 运行，然后打开一个新的终端，输入以下命令：ros2 node list，会显示所有正在运行的节点名  4. 查看 ros 的配置信息： export |grep ros 
预期结果	ros2 命令可正常使用且 ros 的配置信息正确
实际测试结果	pass

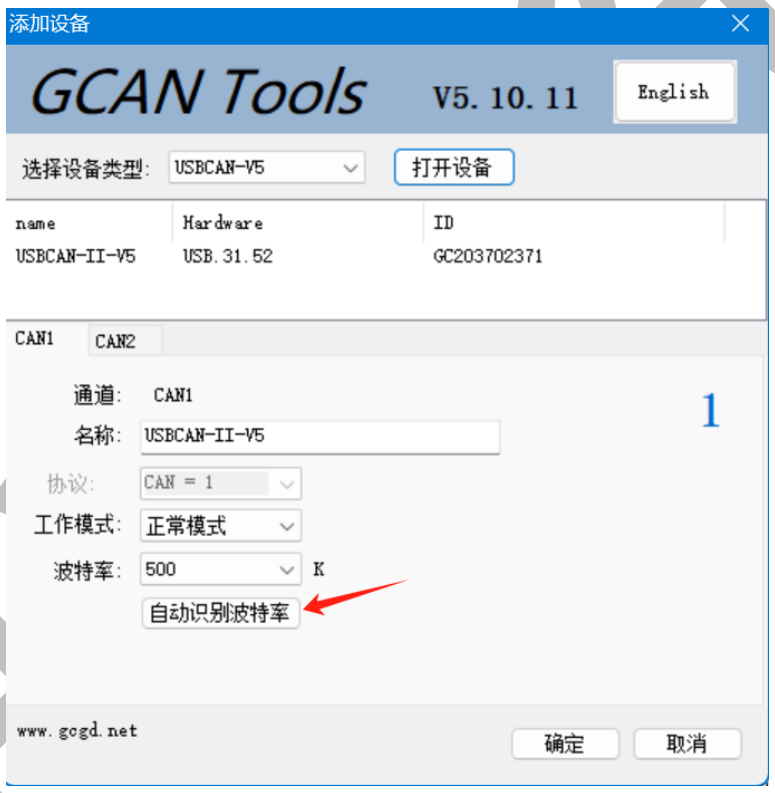
5 外设兼容性

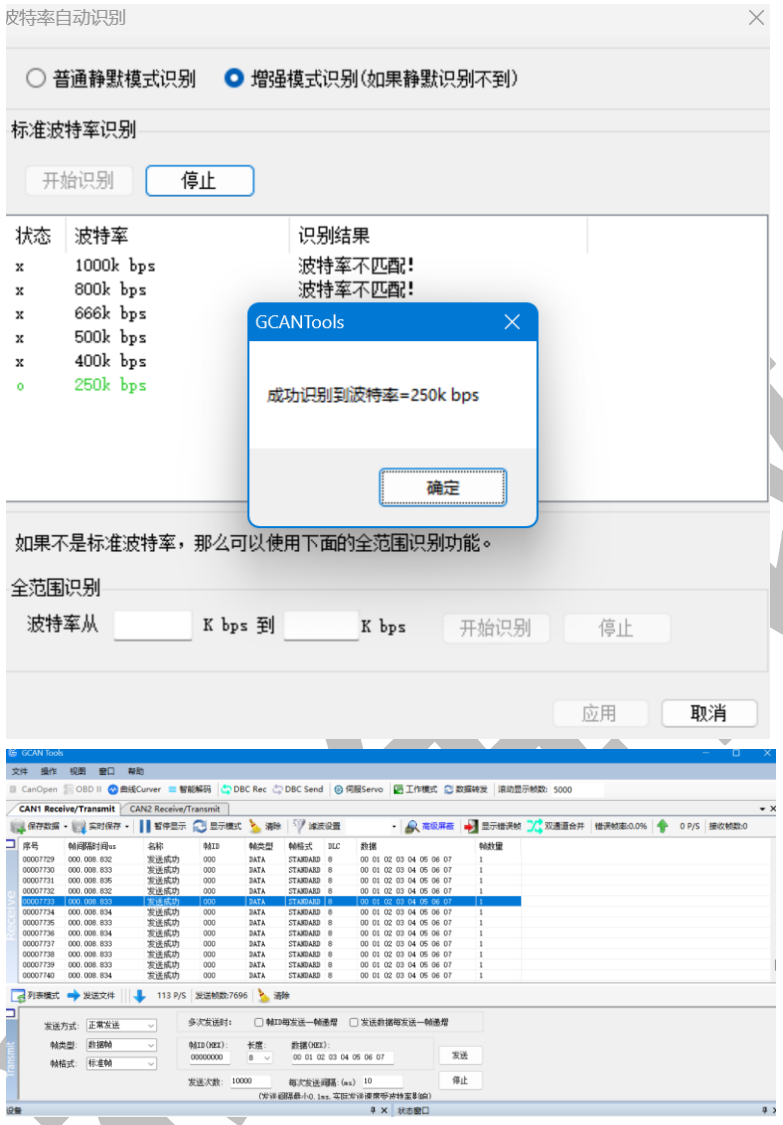
5.1CAN 通信交互

测试内容	Microros—ros2 通信测试
测试前提	1. 启动 RT-Thread 系统，通过物理串口登录终端； 2. CAN 连接开发板跟 PC 端，设置波特率为 250K，然后点击打开设备； 
测试步骤	1. root linux 端，启动 rt-thread，再运行 agent.sh cd threevms / guestlrt.sh  cd threevms / ./agent  2. 进 rt-thread 系统，分别初始化 can、microros can_sample

```
msh />can_s
can_sample
msh />can_sample
rockchip_canfd2 int BUS ERR happened!
rockchip_canfd2 int BUS ERR happened!
rockchip_canfd2 int BUS ERR happened!
rockchip_canfd2 int BUS ERR happened!
rockchip_canfd2 int BUS ERR happened!
rockchip_canfd2 int BUS ERR happened!
rockchip_canfd2 int BUS ERR happened!
rockchip_canfd2 int BUS ERR happened!
rockchip_canfd2 int BUS ERR happened!
rockchip_canfd2 int BUS ERR happened!
rockchip_canfd2 int BUS ERR happened!
rockchip_canfd2 int BUS ERR happened!
```

3. 打开上位机软件 GCAN Tools



	
预期结果	Microros 向 ros2 发送数据成功
实际测试结果	pass

5.2 485 接口测试

测试内容	验证 RS485 接口与 1 拖 4 超声波雷达的通信及数据交互频率
测试前提	1. 深圳电应普 1 拖 4 超声波雷达接入 RK3588 的 RS485 接口； 2. 雷达供电正常，波特率与设备匹配（9600）。

测试步骤	1. 查模块：lsusb； 查询模块识别：dmesg| grep usb
预期结果	模块识别成功
实际测试结果	硬件返厂检修

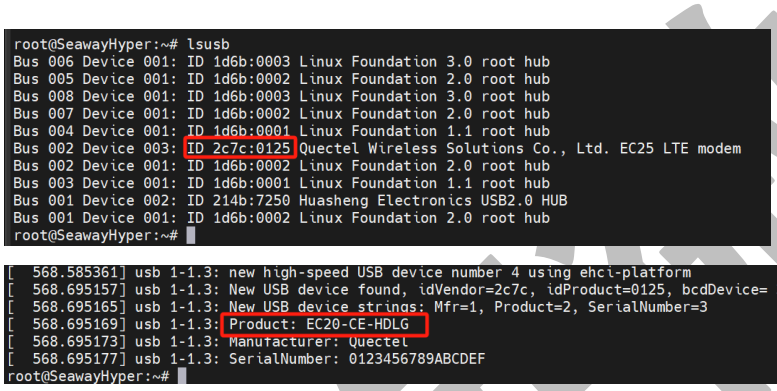
5.3 IMU 识别

测试内容	验证 IMU 识别是否成功
测试前提	1. IMU 硬件连接开发板
测试步骤	查模块：lsusb； root@SeawayHyper:~# lsusb Bus 006 Device 002: ID 2bc5:0800 Orbbec 3D Technology International, Inc Orbbec Gemini 335 Bus 006 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub Bus 005 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub Bus 008 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub Bus 007 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub Bus 004 Device 004: ID 10c4:ea60 Silicon Labs CP210x UART Bridge Bus 004 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub Bus 003 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub Bus 001 Device 089: ID 1c4f:0002 SiGma Micro Keyboard TRACER Gamma Ivory Bus 001 Device 090: ID 046d:c05a Logitech, Inc. M90/M100 Optical Mouse Bus 001 Device 002: ID 214b:7250 Huasheng Electronics USB2.0 HUB Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub root@SeawayHyper:~#
预期结果	设备识别成功
实际测试结果	pass

5.4 4G/5G 模块测试

5.4.1 4G/5G 模块驱动安装

测试内容	驱动安装验证

测试前提	1.移远模组 4G 移远 EC20 模块； 2. 已经安装对应驱动
测试步骤	1.将模组连接到开发板，启动 root Linux； 2. 通过 lsmod 命令查否加载模组驱动； 3. 查看 dmesg 日志，确认模组被正确识别。 
预期结果	1. lsmod 输出中包含模组驱动名称（如 option、qmi_wwan）。 2. dmesg 日志显示 "New USB device found" 及模组相关信息（如厂商 ID、产品 ID）
实际测试结果	pass

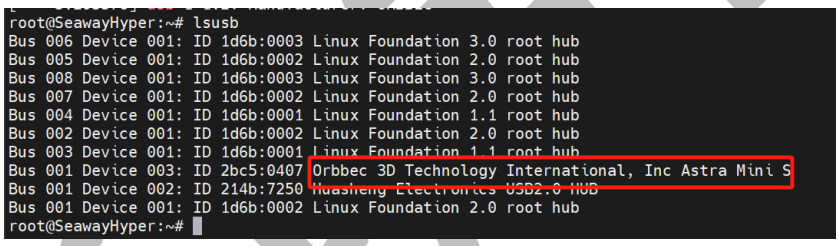
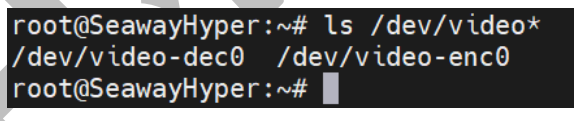
5.4.2 4G/5G 模块设备节点生成

测试内容	4G/5G 模块设备节点生成测试
测试前提	4G/5G 模块，移远模组已经安装对应驱动
测试步骤	1. 执行 ls /dev/ttyUSB*命令，查看系统生成的 USB - 串口设备节点。 2. 记录生成的节点数量及名称。 
预期结果	1. 至少生成 1 个 USB - 串口设备节点（如/dev/ttyUSB0）。

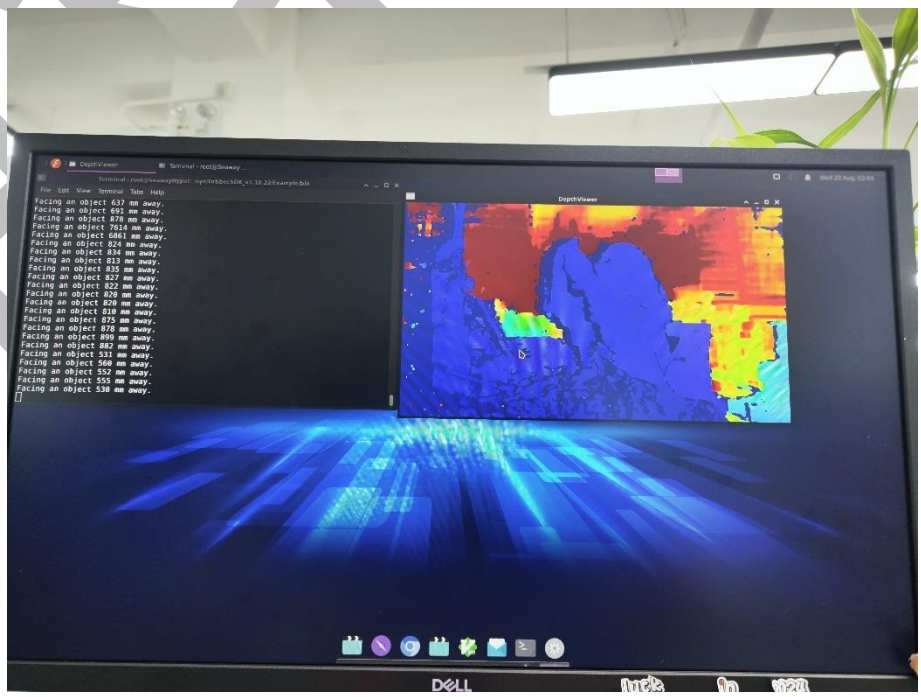
	2. 无重复或冲突的设备节点。
实际测试结果	pass

5.5 深度相机测试

5.5.1 深度相机识别

测试内容	深度相机（奥比中光）识别验证
测试前提	1.确认奥比中光 Gemini 335 连接至 USB3.0 接口；
测试步骤	1. 执行 lsusb，查找奥比中光设备,再执行 dmesg | grep usb，筛选 USB3.0 控制器日志   2. 查看视频设备节点：ls /dev/video* 
预期结果	1.日志显示 USB 设备的相关信息，无 error 或 disconnect 报错； 2.生成节点；
实际测试结果	pass

5.5.2 深度相机启动

测试内容	深度相机（奥比中光）启动
测试前提	1. 奥比中光 Gemini 335 连接至 USB3.0 接口； 2. 安装 Linux 依赖： sudo apt install build-essential freeglut3 freeglut3-dev libusb-1.0-0-dev ros-humble-backward-ros libgflags-dev nlohmann-json3-dev libgoogle-glog-dev 3.环境已上传奥比中光深度相关 SDK 工具包 OrbbecSDK_v1.10.22
测试步骤	1. OrbbecSDK_v1.10.22 上传至 opt 目录下，执行以下命令： cd /opt/OrbbecSDK_v1.10.22/Example/build cd cmake .. cmake --build . --config Releasecd bi cd /opt/OrbbecSDK_v1.10.22/Example/bin 执行 ./DepthViewer 

预期结果	1. 深度相机启动成功 2. 成功捕捉画面
实际测试结果	pass

5.5.3 深度相机热插拔

测试内容	深度相机（奥比中光）热插拔测试
测试前提	1. 奥比中光 Gemini 335 连接至 USB3.0 接口；
测试步骤	相机运行时，插拔 USB 线 3 次，每次间隔 10 秒。
预期结果	1. 每次插拔后，dmesg 显示设备重新识别。 
实际测试结果	pass

5.6 RGB（usb 摄像头）相机测试

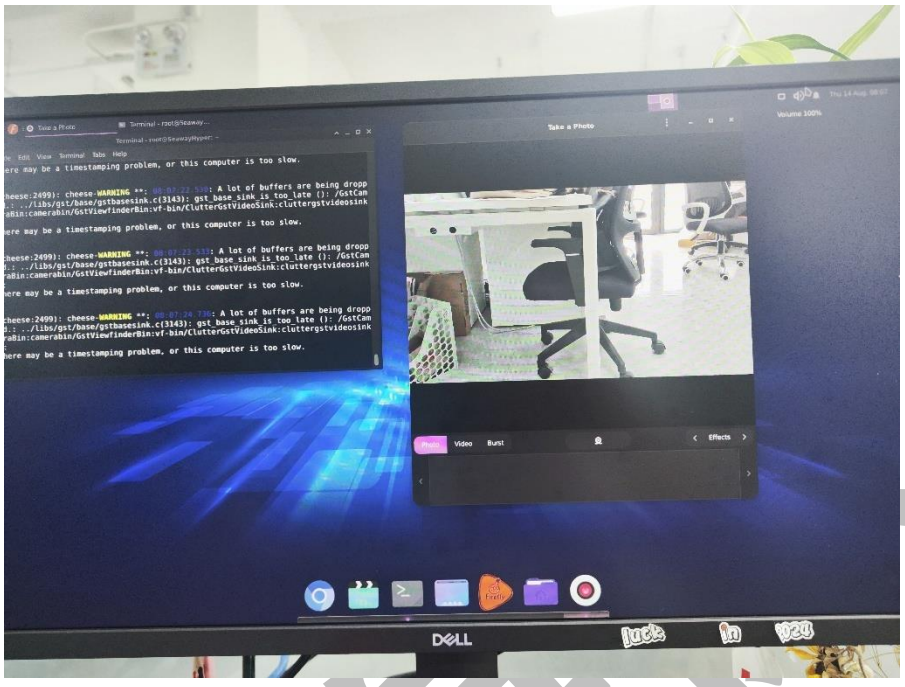
5.6.1 RGB（usb 摄像头）相机识别

测试内容	RGB 相机（usb 摄像头）识别验证
测试前提	1. RGB 相机（usb 摄像头）已经连接开发板； 2. 安装依赖包 sudo apt install ros-humble-usb-cam ros-humble-image-view

测试步骤	1. 连接 USB 摄像头后，执行以下命令查看设备节点 lsusb dmesg ls /dev/video*  
预期结果	开发板可成功识别 usb 摄像头设备
实际测试结果	pass

5.6.2 RGB（usb 摄像头）相机启动

测试内容	RGB 相机（usb 摄像头）识别验证
测试前提	1. RGB 相机（usb 摄像头）已经连接开发板；
测试步骤	1. root linux 安装应用程序茄子（cheese） sudo apt-get install cheese 2. 装好后，用命令：cheese，即可打开。如果指定打开 video0，输入命令： cheese -d /dev/video0

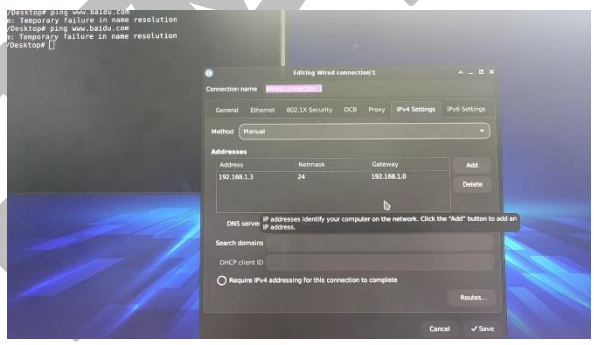
	
预期结果	摄像头可成功捕捉画面
实际测试结果	pass

5.7 USB-RTK982 模块识别

测试内容	982 模块识别测试
测试前提	USB 线连模块与开发板
测试步骤	1.查模块：lsusb； 查询模块识别：dmesg| grep usb root@SeawayHyper:~# lsusb Bus 006 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub Bus 005 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub Bus 008 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub Bus 007 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub Bus 004 Device 002: ID 1a86:7523 QinHeng Electronics CH340 serial converter Bus 004 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub Bus 003 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub Bus 001 Device 002: ID 214b:7250 Huasheng Electronics USB2.0 HUB Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub root@SeawayHyper:~#

	[4.723922] usbhid: USB HID core driver [4.727570] usbcore: registered new interface driver snd-usb-audio [4.764239] usb 1-1: new high-speed USB device number 2 using ehci-platform [4.910829] usb 1-1: New USB device found, idVendor=214b, idProduct=7250, bcdDevice= 1.00 [4.910854] usb 1-1: New USB device strings: Mfr=0, Product=1, SerialNumber=0 [4.910862] usb 1-1: Product: USB2.0 HUB [5.090222] usb 4-1: new full-speed USB device number 2 using ohci-platform [5.302249] usb 4-1: New USB device found, idVendor=1a86, idProduct=7523, bcdDevice= 2.64 [5.302270] usb 4-1: New USB device strings: Mfr=0, Product=2, SerialNumber=0 [5.302278] usb 4-1: Product: USB Serial [5.316338] usb 4-1: ch341-uart converter now attached to ttyUSB0 root@SeawayHyper:~# 2. ls /dev/ttyUSB* 检查虚拟串口数量 root@SeawayHyper:~# ls /dev/ttyUSB* /dev/ttyUSB0 root@SeawayHyper:~#
预期结果	1.日志显示 “USB serial converter”，无错误码； 2.生成串口
实际测试结果	pass

5.8 激光雷达发送数据测试

测试内容	激光雷达发送数据测试
测试前提	1. 连接雷达网络接口和电源线 2. 根据雷达设置的目标 IP 设置电脑有线连接 IP,（可用 ifconfig 命令查看有线 ip 是否设置成功, 雷达 ip 为 192.168.1.200，需要把自己的设备 ip 设置成 192.168.1.3 
测试步骤	1.打开终端:ping 雷达 IP， ping 通则正常， 否则检查硬件连接

	root@SeawayHyper:~# ping 192.168.1.200 PING 192.168.1.200 (192.168.1.200) 56(84) bytes of data. 64 bytes from 192.168.1.200: icmp_seq=1 ttl=128 time=0.177 ms 64 bytes from 192.168.1.200: icmp_seq=2 ttl=128 time=0.171 ms 64 bytes from 192.168.1.200: icmp_seq=3 ttl=128 time=0.171 ms 64 bytes from 192.168.1.200: icmp_seq=4 ttl=128 time=0.171 ms 64 bytes from 192.168.1.200: icmp_seq=5 ttl=128 time=0.330 ms ^[`64 bytes from 192.168.1.200: icmp_seq=6 ttl=128 time=0.276 ms 64 bytes from 192.168.1.200: icmp_seq=7 ttl=128 time=0.164 ms 64 bytes from 192.168.1.200: icmp_seq=8 ttl=128 time=0.163 ms 2. tcpdump -n -i eth0，（此处 eth0 为有线网络设备名， 详见 ifconfig 有线连接显示设备名)查看雷达发送数据包情况 07:01:01.404349 IP 192.168.1.200.2369 > 192.168.1.3.2368: UDP, length 1206 07:01:01.405547 IP 192.168.1.200.2369 > 192.168.1.3.2368: UDP, length 1206 07:01:01.406743 IP 192.168.1.200.2369 > 192.168.1.3.2368: UDP, length 1206 07:01:01.407941 IP 192.168.1.200.2369 > 192.168.1.3.2368: UDP, length 1206 07:01:01.409138 IP 192.168.1.200.2369 > 192.168.1.3.2368: UDP, length 1206 07:01:01.410339 IP 192.168.1.200.2369 > 192.168.1.3.2368: UDP, length 1206 07:01:01.411536 IP 192.168.1.200.2369 > 192.168.1.3.2368: UDP, length 1206 07:01:01.412734 IP 192.168.1.200.2369 > 192.168.1.3.2368: UDP, length 1206 07:01:01.413931 IP 192.168.1.200.2369 > 192.168.1.3.2368: UDP, length 1206 07:01:01.415128 IP 192.168.1.200.2369 > 192.168.1.3.2368: UDP, length 1206 07:01:01.416329 IP 192.168.1.200.2369 > 192.168.1.3.2368: UDP, length 1206 07:01:01.417525 IP 192.168.1.200.2369 > 192.168.1.3.2368: UDP, length 1206 07:01:01.418722 IP 192.168.1.200.2369 > 192.168.1.3.2368: UDP, length 1206 07:01:01.419923 IP 192.168.1.200.2369 > 192.168.1.3.2368: UDP, length 1206 07:01:01.421119 IP 192.168.1.200.2369 > 192.168.1.3.2368: UDP, length 1206 07:01:01.422317 IP 192.168.1.200.2369 > 192.168.1.3.2368: UDP, length 1206 07:01:01.423515 IP 192.168.1.200.2369 > 192.168.1.3.2368: UDP, length 1206 07:01:01.424713 IP 192.168.1.200.2369 > 192.168.1.3.2368: UDP, length 1206 07:01:01.425913 IP 192.168.1.200.2369 > 192.168.1.3.2368: UDP, length 1206 07:01:01.427109 IP 192.168.1.200.2369 > 192.168.1.3.2368: UDP, length 1206 07:01:01.428307 IP 192.168.1.200.2369 > 192.168.1.3.2368: UDP, length 1206 07:01:01.429507 IP 192.168.1.200.2369 > 192.168.1.3.2368: UDP, length 1206 07:01:01.430703 IP 192.168.1.200.2369 > 192.168.1.3.2368: UDP, length 1206 07:01:01.431901 IP 192.168.1.200.2369 > 192.168.1.3.2368: UDP, length 1206 07:01:01.433100 IP 192.168.1.200.2369 > 192.168.1.3.2368: UDP, length 1206 07:01:01.434298 IP 192.168.1.200.2369 > 192.168.1.3.2368: UDP, length 1206 07:01:01.435496 IP 192.168.1.200.2369 > 192.168.1.3.2368: UDP, length 1206 07:01:01.436694 IP 192.168.1.200.2369 > 192.168.1.3.2368: UDP, length 1206 07:01:01.437892 IP 192.168.1.200.2369 > 192.168.1.3.2368: UDP, length 1206 07:01:01.439090 IP 192.168.1.200.2369 > 192.168.1.3.2368: UDP, length 1206 07:01:01.440288 IP 192.168.1.200.2369 > 192.168.1.3.2368: UDP, length 1206 07:01:01.441486 IP 192.168.1.200.2369 > 192.168.1.3.2368: UDP, length 1206
预期结果	雷达发送到目的端数据包为 1206 个字节，雷达数据发送正常
实际测试结果	pass

5.9双目相机测试

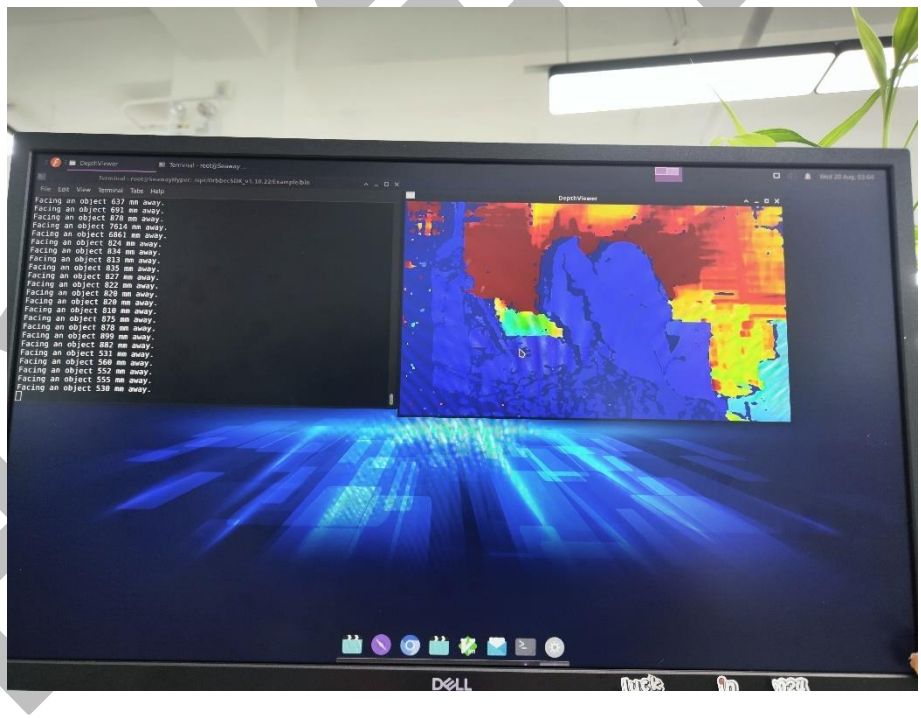
5.9.1 双目相机识别

测试内容	双目相机奥比中光 Gemini 335 识别测试
测试前提	双目相机奥比中光 Gemini 335 连接 USB 接口
测试步骤	1. 连接 USB 摄像头后，执行以下命令查看设备节点 Lsusb

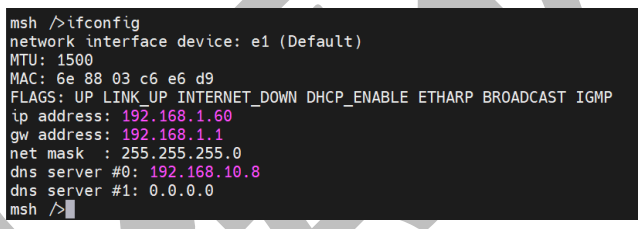
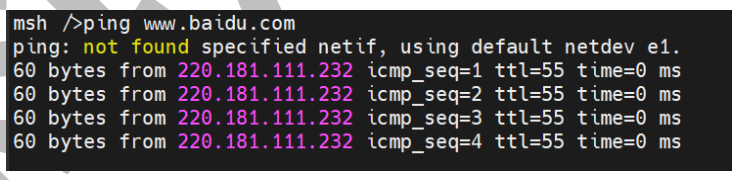
	root@SeawayHyper:~# lsusb Bus 006 Device 004: ID 2bc5:0800 Orbbec 3D Technology International, Inc Orbbec Gemini 335 Bus 006 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub Bus 005 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub Bus 008 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub Bus 007 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub Bus 004 Device 005: ID 1a86:7523 QinHeng Electronics CH340 serial converter Bus 004 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub Bus 003 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub Bus 001 Device 089: ID 1c4f:0002 SiGma Micro Keyboard TRACER Gamma Ivory Bus 001 Device 090: ID 046d:c05a Logitech, Inc. M90/M100 Optical Mouse Bus 001 Device 002: ID 214b:7250 Huasheng Electronics USB2.0 HUB Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub root@SeawayHyper:~# Dmesg 8223.358599] usb 6-1: USB disconnect, device number 3 8228.362459] usb 6-1: new SuperSpeed Gen 1 USB device number 4 using xhci-hcd 8228.380222] usb 6-1: New USB device found, idVendor=2bc5, idProduct=0800, bcdDevice= 0.01 8228.380227] usb 6-1: New USB device strings: Mfr=1, Product=2, SerialNumber=10 8228.380231] usb 6-1: Product: Orbbec Gemini 335 8228.380233] usb 6-1: Manufacturer: Orbbec(R) 8228.380236] usb 6-1: SerialNumber: CP05353000EH 8228.389784] uvcvideo: Found UVC 1.50 device Orbbec Gemini 335 (2bc5:0800) 8228.394260] uvcvideo: Found UVC 1.50 device Orbbec Gemini 335 (2bc5:0800) 8228.397473] hid-generic 0003:2BC5:0800.007D: hiddev96,hidraw3: USB HID v1.11 Device [Orbbec ini 335] on usb-xhci-hcd.5.auto-1/input6 8330.493261] usb 4-1: USB disconnect, device number 4 8330.493707] cp210x ttyUSB0: cp210x converter now disconnected from ttyUSB0 ls /dev/ root@SeawayHyper:~# ls /dev/ 335 hidraw1 loop5 null tty1 tty3 tty5 ubi_ctrl vcsa block hidraw2 loop6 port tty10 tty30 tty50 uhid vcsa1 btrfs-control hugepages loop7 ptmx tty11 tty31 tty51 uinput vcsa2 bus hvisor loop-control ptp0 tty12 tty32 tty52 urandom vcsa3 char hwrng mali0 pts tty13 tty33 tty53 usb-ffs vcsa4 console i2c-0 mapper ram0 tty14 tty34 tty54 usbmon0 vcsa5 cpu_dma_latency i2c-2 mem random tty15 tty35 tty55 usbmon1 vcsa6 crypto i2c-3 mmcblk0 rfkill tty16 tty36 tty56 usbmon2 vcsa7 disk i2c-4 mmcblk0boot0 rga tty17 tty37 tty57 usbmon3 vcsu dma_heap i2c-5 mmcblk0boot1 rtc tty18 tty38 tty58 usbmon4 vcsu1 dri i2c-6 mmcblk0p1 rtc0 tty19 tty39 tty59 usbmon5 vcsu2 fd i2c-9 mmcblk0p2 serial tty2 tty4 tty6 usbmon6 vcsu3 full iio:device0 mmcblk0p3 shm tty20 tty40 tty60 usbmon7 vcsu4 fuse initctl mmcblk0p4 snd tty21 tty41 tty61 usbmon8 vcsu5 Gemini input mmcblk0p5 stderr tty22 tty42 tty62 vcs vcsu6 gpiochip0 kmsg mmcblk0p6 stdin tty23 tty43 tty63 vcs1 vcsu7 gpiochip1 log mmcblk0p7 stdout tty24 tty44 tty7 vcs2 vendor_storag gpiochip2 loop0 mmcblk0p8 sw_sync tty25 tty45 tty8 vcs3 video-dec0 gpiochip3 loop1 mmcblk0rpmb tee0 tty26 tty46 tty9 vcs4 video-enc0 gpiochip4 loop2 mpp_service teepriv0 tty27 tty47 ttyFIQ0 vcs5 zero gpiochip5 loop3 mqueue tty tty28 tty48 ttyS6 vcs6 zram0 hidraw0 loop4 net tty0 tty29 tty49 ttyUSB0 vcs7 root@SeawayHyper:~# ubscribing to the professional edition here: https://mobaxterm.mobatek.net
预期结果	开发板可成功识别双目相机设备
实际测试结果	pass

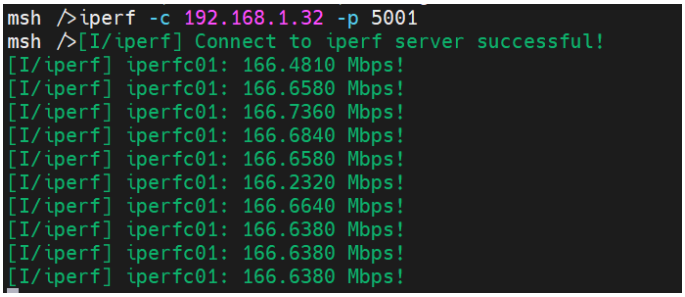
5.9.2 双目相机启动

测试内容	深度相机（奥比中光）捕捉画面测试
测试前提	1. 奥比中光 Gemini 335 连接至 USB3.0 接口； 2. 安装 Linux 依赖：

	sudo apt install build-essential freeglut3 freeglut3-dev libusb-1.0-0-dev ros-humble-backward-ros libgflags-dev nlohmann-json3-dev libgoogle-glog-dev 3.环境已上传奥比中光深度相关 SDK 工具包 OrbbecSDK_v1.10.22
测试步骤	1. OrbbecSDK_v1.10.22 上传至 opt 目录下，执行以下命令： cd /opt/OrbbecSDK_v1.10.22/Example/build cd cmake .. cmake --build . --config Releasecd bi cd /opt/OrbbecSDK_v1.10.22/Example/bin 执行 ./DepthViewer 
预期结果	1.深度相机启动成功 2.成功捕捉画面
实际测试结果	pass

5.10 网口 1-组网测试

☒☒内容	网口 1 ☒网☒☒支持
☒☒前提	1. Rt-thread 系☒正常运行； 2. 网☒正常☒接
☒☒步☒	1. ifconfig  2. ping www.baidu.com;  3. server 端：iperf -s -p 5001  client 端：iperf -c 192.168.1.32 -p 5001

	
区期区果	网口 1 网区通信正常
区区区果	pass