



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 <pre>[root@localhost pubimage]# ls -lh RK3588_NJ_20250826 -rw-r--r--. 1 root root 4.2G Aug 26 16:40 RK3588_NJ_20250826 [root@localhost pubimage]#</pre> 内存占用情况： <pre>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:~#</pre> cpu 占用情况： <pre>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]</pre>

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

测试内容	ARM 处理器虚拟化功能测试												
测试前提	root linux 系统正常运行												
测试步骤	1. root linux 系统正常运行； 2. 查看当前系统信息； cat /etc/os-release <pre>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#</pre> 3. 启动 rt-thread，查看 rt-thread 运行情况； <pre>root@SeawayHyper:~/SeawayHyper# cd bin/ root@SeawayHyper:~/SeawayHyper/bin# ./hvisor zone list</pre><table><tr><th>zone_id</th><th>cpus</th><th>name</th><th>status</th></tr><tr><td>0</td><td>0, 1</td><td>root-linux</td><td>running</td></tr><tr><td>1</td><td>6</td><td>rtthread</td><td>running</td></tr></table><pre>root@SeawayHyper:~/SeawayHyper/bin#</pre>	zone_id	cpus	name	status	0	0, 1	root-linux	running	1	6	rtthread	running
zone_id	cpus	name	status										
0	0, 1	root-linux	running										
1	6	rtthread	running										
测试结果	系统正常运行，系统版本信息准确。												

实验结果	pass
------	------

1.2 内存虚拟化

1.2.1 静态内存分配管理

实验内容	静态内存分配管理功能实验
实验前提	Hypervisor 与 Guest 能正常运行
实验步骤	1. 配置并验证不同内存规格的 Guest VM： 配置 Guest A 的虚拟机或虚拟机配置文件：将其静态内存大小设定为 256MB  配置 Guest B 的虚拟机或虚拟机配置文件：将其静态内存大小设定为 1GB。

	<pre>hvisor@480000 { no-map; reg = <0x00 0x480000 0x00 0x800000>; }; nonroot@40000000 { no-map; reg = <0x00 0x40000000 0x00 0x40000000>; };</pre> 2. 利用 Hypervisor 启 Guest VM 并内存分配； 分别配置了不同内存大小的 Guest A 和 Guest B。 在 每个 Guest VM 内部后，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，不同内存大小的虚拟机分配相的内存，并在虚拟机内部正确示些已分配的内存。
果	，Hypervisor 成功地虚拟机分配了的不同内存大小，且虚拟机内部告的内存与配置一致，功能符合期。 <pre>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-ses=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:~#</pre>

	<pre>ZCU102-petalinux login: ZCU102-petalinux login: ZCU102-petalinux login: root Password: Login incorrect ZCU102-petalinux login: root Password: [72.865742] audit: type=1334 audit(1667936029.566:6): prog-id=9 op=LOAD [72.868448] audit: type=1334 audit(1667936029.569:7): prog-id=10 op=LOAD [73.324553] audit: type=1006 audit(1667936030.026:8): pid=316 uid=0 old-auid=4294967295 auid=0 tty=(none) old-s s=4294967295 ses=1 res=1 [73.353997] audit: type=1334 audit(1667936030.056:9): prog-id=11 op=LOAD [73.357388] audit: type=1334 audit(1667936030.056:10): prog-id=11 op=UNLOAD [73.359392] audit: type=1334 audit(1667936030.056:11): prog-id=12 op=LOAD [73.363056] audit: type=1334 audit(1667936030.056:12): prog-id=12 op=UNLOAD root@ZCU102-petalinux:~# root@ZCU102-petalinux:~# root@ZCU102-petalinux:~# free -h total used free shared buff/cache available Mem: 974.1M 48.8M 871.7M 8.5M 53.6M 901.2M Swap: 0 0 0</pre>
--	---

1.2.2 stage-2 页表管理

实验内容	stage-2 页表管理功能实验
实验前提	Hypervisor 已成功建立基址的 Stage-1 页表映射
实验步骤	1. 初始化并配置 Hypervisor 的 Stage-2 页表。 2. 验证一个 Guest 物理地址在 Stage-2 页表中的映射结果。 3. 验证验证到的宿主机物理地址与预期是否一致。
实验结果	实验结果验证显示，所给 Guest 物理地址通过 Stage-2 页表正确映射到其预期的宿主机物理地址。

测试结果

预期, Hypervisor 的 Stage-2 表功能正常, 所预期的 Guest 物理地址被准确地映射到其预期的宿主机物理地址, 符合预期。

```
[INFO 0] (hvisor::arch::aarch64::zone:59) VM stage 2 memory set: MemorySet {
  regions: [
    MemoryRegion {
      vaddr_range: 0x100000..0x1f0000,
      size: 0xf0000,
      flags: MemFlags(
        READ | WRITE | EXECUTE,
      ),
      mapper: Offset(
        0x0,
      ),
    },
    MemoryRegion {
      vaddr_range: 0x200000..0x840000,
      size: 0x640000,
      flags: MemFlags(
        READ | WRITE | EXECUTE,
      ),
      mapper: Offset(
        0x0,
      ),
    },
    MemoryRegion {
      vaddr_range: 0x940000..0xf0000000,
      size: 0xe6c00000,
      flags: MemFlags(
        READ | WRITE | EXECUTE,
      ),
      mapper: Offset(
        0x0,
      ),
    },
    MemoryRegion {
      vaddr_range: 0xfc000000..0xfe000000,
      size: 0x2000000,
      flags: MemFlags(
        READ | WRITE | EXECUTE | IO,
      ),
      mapper: Offset(
        0x0,
      ),
    },
    MemoryRegion {
      vaddr_range: 0xfc000000..0xfe000000,
      size: 0x2000000,
      flags: MemFlags(
        READ | WRITE | EXECUTE | IO,
      ),
      mapper: Offset(
        0x0,
      ),
    },
    MemoryRegion {
      vaddr_range: 0xfe000000..0xfe600000,
      size: 0x600000,
      flags: MemFlags(
        READ | WRITE | EXECUTE | IO,
      ),
      mapper: Offset(
        0x0,
      ),
    },
    MemoryRegion {
      vaddr_range: 0xfe800000..0xff000000,
      size: 0x800000,
      flags: MemFlags(
        READ | WRITE | EXECUTE | IO,
      ),
      mapper: Offset(
        0x0,
      ),
    },
    MemoryRegion {
      vaddr_range: 0x100000000..0x3fc000000,
      size: 0x2fc000000,
      flags: MemFlags(
        READ | WRITE | EXECUTE,
      ),
      mapper: Offset(
        0x0,
      ),
    },
    MemoryRegion {
      vaddr_range: 0x3fc500000..0x3fff00000,
      size: 0x3a00000,
      flags: MemFlags(
        READ | WRITE | EXECUTE,
      ),
      mapper: Offset(
        0x0,
      ),
    },
    MemoryRegion {
      vaddr_range: 0x4f0000000..0x500000000,
      size: 0x10000000,
      flags: MemFlags(
        READ | WRITE | EXECUTE,
      ),
      mapper: Offset(
        0x0,
      ),
    },
  ],
  page_table: "hvisor::arch::aarch64::paging::HvPageTable<size, hvisor::arch::aarch64::s2pt::Pa
sor::arch::aarch64::s2pt::S2PTInstr>",
  page_table_root: 0xae000,
}
[INFO 0] (hvisor::arch::aarch64::zone:71) page table: va: 0x100000, pa: 0x100000
[INFO 0] (hvisor::device::irqchip:gicv3::vgic:39) vgicv3_mmio_init
[INFO 0] (hvisor::device::irqchip:gicv3::vgic:49) registering gicr 0 at 0xfe680000
[INFO 0] (hvisor::device::irqchip:gicv3::vgic:49) registering gicr 1 at 0xfe6a0000
[INFO 0] (hvisor::device::irqchip:gicv3::vgic:49) registering gicr 2 at 0xfe6c0000
[INFO 0] (hvisor::device::irqchip:gicv3::vgic:49) registering gicr 3 at 0xfe6e0000
```


☐☐步☐	1. 运行 check_virtio.sh <pre>root@seawayHyper:~# bash check_virtio.sh 07:50:45 INFO hvisor.c:611: zone config check pass nohup: redirecting stderr to stdout 07:50:45 DEBUG hvisor.c:458: memory_region 0: type 0, physical_start 50000000, virtual_start 50000000, size 10000000 07:50:45 DEBUG hvisor.c:458: memory_region 1: type 0, physical_start 0, virtual_start 0, size 200000 07:50:45 DEBUG hvisor.c:458: memory_region 2: type 2, physical_start ff9d0000, virtual_start ff9d0000, size 1000 07:50:45 DEBUG hvisor.c:458: memory_region 3: type 2, physical_start ff9e0000, virtual_start ff9e0000, size 1000 07:50:45 INFO hvisor.c:530: Kernel size: 38350848, DTB size: 4096 07:50:45 INFO hvisor.c:540: Zone name: linux2 07:50:45 WARN safe_cjson.c:24: hvisor.c:179 - [cJSON_GetObjectItem] Key 'gicc_base' not found in JSON object 07:50:45 WARN safe_cjson.c:24: hvisor.c:181 - [cJSON_GetObjectItem] Key 'gich_base' not found in JSON object 07:50:45 WARN safe_cjson.c:24: hvisor.c:183 - [cJSON_GetObjectItem] Key 'gicv_base' not found in JSON object 07:50:45 WARN safe_cjson.c:24: hvisor.c:185 - [cJSON_GetObjectItem] Key 'gicc_offset' not found in JSON object 07:50:45 WARN safe_cjson.c:24: hvisor.c:187 - [cJSON_GetObjectItem] Key 'gicv_size' not found in JSON object 07:50:45 WARN safe_cjson.c:24: hvisor.c:189 - [cJSON_GetObjectItem] Key 'gich_size' not found in JSON object 07:50:45 WARN safe_cjson.c:24: hvisor.c:191 - [cJSON_GetObjectItem] Key 'gicc_size' not found in JSON object 07:50:45 WARN safe_cjson.c:24: hvisor.c:283 - [cJSON_GetObjectItem] Key 'pci_config' not found in JSON object 07:50:45 WARN hvisor.c:285: No pci_config field found. 07:50:45 INFO hvisor.c:564: Calling ioctl to start zone: [linux2] 07:50:45 INFO virtio_console.c:128: char device redirected to /dev/pts/1 07:50:45 INFO virtio_console.c:128: char device redirected to /dev/pts/2 zone_id cpus name status 0 0, 1 root-linux running 1 2 linux2 running 找到 virtio-blk 创建成功记录 找到 virtio-console 创建成功记录 VirtIO 初始化成功! root@seawayHyper:~#</pre>
☐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

	<pre> root@seawayHyper:~# bash check_virtio.sh 07:50:45 INFO hvvisor.c:611: zone config check pass nohup: redirecting stderr to stdout 07:50:45 DEBUG hvvisor.c:458: memory_region 0: type 0, physical_start 50000000, virtual_start 50000000, size 10000000 07:50:45 DEBUG hvvisor.c:458: memory_region 1: type 0, physical_start 0, virtual_start 0, size 2000000 07:50:45 DEBUG hvvisor.c:458: memory_region 2: type 2, physical_start ff9d0000, virtual_start ff9d0000, size 1000 07:50:45 DEBUG hvvisor.c:458: memory_region 3: type 2, physical_start ff9e0000, virtual_start ff9e0000, size 1000 07:50:45 INFO hvvisor.c:530: Kernel size: 38350848, DTB size: 4096 07:50:45 INFO hvvisor.c:540: Zone name: linux2 07:50:45 WARN safe_cjson.c:24: hvvisor.c:179 - [cJSON_GetObjectItem] Key 'gicc_base' not found in JSON object 07:50:45 WARN safe_cjson.c:24: hvvisor.c:181 - [cJSON_GetObjectItem] Key 'gicv_base' not found in JSON object 07:50:45 WARN safe_cjson.c:24: hvvisor.c:182 - [cJSON_GetObjectItem] Key 'gicv_base' not found in JSON object 07:50:45 WARN safe_cjson.c:24: hvvisor.c:185 - [cJSON_GetObjectItem] Key 'gicc_offset' not found in JSON object 07:50:45 WARN safe_cjson.c:24: hvvisor.c:187 - [cJSON_GetObjectItem] Key 'gicv_size' not found in JSON object 07:50:45 WARN safe_cjson.c:24: hvvisor.c:189 - [cJSON_GetObjectItem] Key 'gicv_size' not found in JSON object 07:50:45 WARN safe_cjson.c:24: hvvisor.c:191 - [cJSON_GetObjectItem] Key 'gicc_size' not found in JSON object 07:50:45 WARN safe_cjson.c:24: hvvisor.c:283 - [cJSON_GetObjectItem] Key 'pci_config' not found in JSON object 07:50:45 WARN hvvisor.c:285: No pci_config field found. 07:50:45 INFO virtio_console.c:128: char device redirected to /dev/pts/1 07:50:45 INFO virtio_console.c:128: char device redirected to /dev/pts/2 +-----+-----+-----+-----+ zone_id cpus name status +-----+-----+-----+-----+ 0 1 root-linux running 1 2 linux2 running +-----+-----+-----+-----+ 找到 virtio-blk 创建成功记录 找到 virtio-console 创建成功记录 VirtIO 初始化成功! root@seawayHyper:~# </pre>
☑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

	<pre> root@seawayHyper:~# bash check_virtio.sh 07:50:45 INFO hvvisor.c:611: zone config check pass nohup: redirecting stderr to stdout 07:50:45 DEBUG hvvisor.c:458: memory_region 0: type 0, physical_start 50000000, virtual_start 50000000, size 10000000 07:50:45 DEBUG hvvisor.c:458: memory_region 1: type 0, physical_start 0, virtual_start 0, size 2000000 07:50:45 DEBUG hvvisor.c:458: memory_region 2: type 2, physical_start ff9d0000, virtual_start ff9d0000, size 1000 07:50:45 DEBUG hvvisor.c:458: memory_region 3: type 2, physical_start ff9e0000, virtual_start ff9e0000, size 1000 07:50:45 INFO hvvisor.c:530: Kernel size: 38350848, DTB size: 4096 07:50:45 INFO hvvisor.c:540: Zone name: linux2 07:50:45 WARN safe_cjson.c:24: hvvisor.c:179 - [cJSON_GetObjectItem] Key 'gicc_base' not found in JSON object 07:50:45 WARN safe_cjson.c:24: hvvisor.c:181 - [cJSON_GetObjectItem] Key 'gicv_base' not found in JSON object 07:50:45 WARN safe_cjson.c:24: hvvisor.c:182 - [cJSON_GetObjectItem] Key 'gicc_offset' not found in JSON object 07:50:45 WARN safe_cjson.c:24: hvvisor.c:185 - [cJSON_GetObjectItem] Key 'gicv_offset' not found in JSON object 07:50:45 WARN safe_cjson.c:24: hvvisor.c:187 - [cJSON_GetObjectItem] Key 'gicv_size' not found in JSON object 07:50:45 WARN safe_cjson.c:24: hvvisor.c:189 - [cJSON_GetObjectItem] Key 'gicc_size' not found in JSON object 07:50:45 WARN safe_cjson.c:24: hvvisor.c:191 - [cJSON_GetObjectItem] Key 'gicc_size' not found in JSON object 07:50:45 WARN safe_cjson.c:24: hvvisor.c:283 - [cJSON_GetObjectItem] Key 'pci_config' not found in JSON object 07:50:45 WARN hvvisor.c:285: No pci_config field found. 07:50:45 INFO hvvisor.c:564: Calling ioctl to start zone: [linux2] 07:50:45 INFO virtio_console.c:128: char device redirected to /dev/pts/1 07:50:45 INFO virtio_console.c:128: char device redirected to /dev/pts/2 +-----+-----+-----+-----+ zone_id cpus name status +-----+-----+-----+-----+ 0 1 root-linux running 1 2 linux2 running +-----+-----+-----+-----+ 抓到 virtio-blk 创建成功记录 抓到 virtio-console 创建成功记录 VirtIO 初始化成功! root@seawayHyper:~# </pre>
预期结果	输出 VirtIO 初始化成功
测试结果	pass

1.5.2 注入中断

内容	中断注入功能
前提	Hypervisor 与 Guest 正常运行
步骤	通过 Hypervisor 向 Guest VM 注入特定中断并注入信息。
预期结果	在 Hypervisor 的 log 中看到中断注入的 log 信息
测试结果	能正常看到注入信息

	<pre>[113.873984] IIS: No IIS available, not enabling LPis [113.993321] arch_timer: cp15 timer(s) running at 24.00MHz (virt). [113.993876] clocksource: arch_sys_counter: mask: 0xffffffffffffff max_cycles: 0x588fe9dc0, max_idle_ns: 44079520 2592 ns [113.994850] sched_clock: 56 bits at 24MHz, resolution 41ns, wraps every 4398046511097ns [INFO 0] (hvisor::device::irqchip::gicv3:491) To Inject IRQ 27, find lr 0 [113.997364] Console: colour dummy device 80x25 [113.997787] Calibrating delay loop (skipped), value calculated using timer frequency.. 48.00 BogoMIPS (lpj=80000) [113.998717] pid_max: default: 32768 minimum: 301 [113.999210] LSM: Security Framework initializing [113.999677] Mount-cache hash table entries: 16384 (order: 5, 131072 bytes, linear) [114.000376] Mountpoint-cache hash[INFO 0] (hvisor::device::irqchip::gicv3:491) To Inject IRQ 27, find lr 0 table entries: 16384 (order: 5, 131072 bytes, linear) [114.003245] rcu: Hierarchical SRCU implementation. [114.004635] EFI services will not be available. [114.005173] smp:[INFO 0] (hvisor::device::irqchip::gicv3:491) To Inject IRQ 27, find lr 0 Bringing up secondary CPUs ... [INFO 0] (hvisor::arch::aarch64::trap:375) psci: try to wake up cpu 1 [INFO 1] (hvisor::event:120) cpu 1 wakeup [INFO 1] (hvisor::arch::aarch64::cpu:170) cpu 1 started at 0xa7cdic8 [INFO 0] (hvisor::device::irqchip::gicv3:491) To Inject IRQ 0, find lr 0 [INFO 0] (hvisor::device::irqchip::gicv3:491) To Inject IRQ 27, find lr 1 [INFO 1] (hvisor::device::irqchip::gicv3:491) To Inject IRQ 0, find lr 0 [INFO 0] (hvisor::device::irqchip::gicv3:491) To Inject IRQ 0, find lr 0 7m[INFO 1] (hvisor::device::irqchip::gicv3:491) To Inject IRQ 27, find lr 0 [114.009200] Detected VIPT I-cache on CPU1 [114.009250] GICv3: CPU1: found redis[INFO 0] (hvisor::device::irqchip::gicv3:491) To Inject IRQ 27, find lr 0 tributor 100 region 0:0x00000000fe6a0000 [114.009415] CPU1: Booted secondary processor 0x00000000100 [0x412fd050] [114.012998] smp: Brought up 1 node, 2 CPUs [114.016688] SMP: Total of 2 processors activated. [114.01710][INFO 1] (hvisor::device::irqchip::gicv3:491) To Inject IRQ 27, find lr 0 9] CPU features: detected: Privileged Access Never [114.018479] CPU features: detected: User Access Override [114.018955][INFO 0] (hvisor::device::irqchip::gicv3:491) To Inject IRQ 27, find lr 0 CPU features: detected: 32-bit EL0 Support [114.020299] CPU features: detected: Common not Private translations [114.020860] CPU features: detected: RAS Extension Support [114.0[INFO 1] (hvisor::device::irqchip::gicv3:491) To Inject IRQ 27, find lr 0 21346] CPU features: detected: Data cache clean to the PoU not required for I/D coherence [114.022988] CPU features: detected: CRC32 instructions [114.023450] CPU feat[INFO 0] (hvisor::device::irqchip::gicv3:491) To Inject IRQ 27, find lr 0 ures: detected: Speculative Store Bypassing Safe (SSBS)</pre>
--	--

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
步骤	通过 Hypervisor 清除特定中断并清除注入信息
预期结果	在 Hypervisor 的 log 中看到清除注入中断的 log

预期结果	能正常看清除注入打印 log
------	----------------

```
[INFO 1] (hvisor::device::irqchip::gicv3:176) Clearing LR[2]
[INFO 0] (hvisor::device::irqchip::gicv3:176) Clearing LR[1]
[INFO 1] (hvisor::device::irqchip::gicv3:176) Clearing LR[3]
[INFO 7] (hvisor:166) CPU 7 hv_pt_install OK.
[INFO 1] (hvisor::device::irqchip::gicv3:181) GICv3: num_priority_bits = 5
[INFO 2] (hvisor::device::irqchip::gicv3:176) Clearing LR[0]
[INFO 0] (hvisor::device::irqchip::gicv3:176) Clearing LR[2]
[INFO 7] (hvisor::device::irqchip::gicv3:173) GICv3: ich_vtr_el2 = 0x90280003, lr_num = 4
[INFO 5] (hvisor::device::irqchip::gicv3:176) Clearing LR[0]
[INFO 0] (hvisor::device::irqchip::gicv3:176) Clearing LR[3]
[INFO 4] (hvisor::device::irqchip::gicv3:173) GICv3: ich_vtr_el2 = 0x90280003, lr_num = 4
[INFO 2] (hvisor::device::irqchip::gicv3:176) Clearing LR[1]
[INFO 6] (hvisor::device::irqchip::gicv3:176) Clearing LR[0]
[INFO 2] (hvisor::device::irqchip::gicv3:176) Clearing LR[2]
[INFO 0] (hvisor::device::irqchip::gicv3:181) GICv3: num_priority_bits = 5
[INFO 7] (hvisor::device::irqchip::gicv3:176) Clearing LR[0]
[INFO 6] (hvisor::device::irqchip::gicv3:176) Clearing LR[1]
[INFO 0] (hvisor::device::irqchip::gicv3:185) Clearing ICH_AP1R0_EL2
[INFO 5] (hvisor::device::irqchip::gicv3:176) Clearing LR[1]
[INFO 3] (hvisor::device::irqchip::gicv3:176) Clearing LR[0]
[INFO 2] (hvisor::device::irqchip::gicv3:176) Clearing LR[3]
[INFO 3] (hvisor::device::irqchip::gicv3:176) Clearing LR[1]
[INFO 7] (hvisor::device::irqchip::gicv3:176) Clearing LR[1]
[INFO 1] (hvisor::device::irqchip::gicv3:185) Clearing ICH_AP1R0_EL2
[INFO 7] (hvisor::device::irqchip::gicv3:176) Clearing LR[2]
[INFO 0] (hvisor::device::irqchip::gicv3:187) GICv3: Pending IRQs and active priorities cleared.
[INFO 1] (hvisor::device::irqchip::gicv3:187) GICv3: Pending IRQs and active priorities cleared.
[INFO 0] (hvisor::device::irqchip::gicv3:146) gicc init done, sdei_ver = 281474976710656
[INFO 3] (hvisor::device::irqchip::gicv3:176) Clearing LR[2]
[INFO 2] (hvisor::device::irqchip::gicv3:181) GICv3: num_priority_bits = 5
[INFO 4] (hvisor::device::irqchip::gicv3:176) Clearing LR[0]
[INFO 1] (hvisor::device::irqchip::gicv3:146) gicc init done, sdei_ver = 281474976710656
[INFO 6] (hvisor::device::irqchip::gicv3:176) Clearing LR[2]
[INFO 7] (hvisor::device::irqchip::gicv3:176) Clearing LR[3]
[INFO 2] (hvisor::device::irqchip::gicv3:185) Clearing ICH_AP1R0_EL2
[INFO 5] (hvisor::arch::aarch64::cpu:204) cpu 5 start parking
[133.392929] Booting Linux on physical CPU 0x0000000000 [0x412fd050]
[133.392950] Linux version 5.10.160 (root@sl-Standard-PC-Q35-ICH9-2009) (aarch64-none-linux-gnu-gcc (GNU Toolchain for the A-profile Architecture 10.3-2021.07 (arm-10.29)) 10.3.1 20210621, GNU ld (GNU Toolchain for the A-profile Architecture 10.3-2021.07 (arm-10.29)) 2.36.1.20210621) #73 SMP Fri Jul 18 10:38:33 CST 2025
[133.399906] Machine model: rockchip,rk3588
[133.399967] earlycon: uart8250 at MMIO32 0x00000000feb50000 (options '')
[133.404115] printk: bootconsole [uart8250] enabled
[133.406190] efi: UEFI not found.
[133.410286] OF: fdt: Reserved memory: failed to reserve memory for node 'drm-cubic-lut@00000000': base 0x00000000
```

1.5.4 启动虚拟机

预期内容	rt-thread 启动正常												
预期前提	root linux 正常运行												
预期步骤	参考《混合部署系统启动方式》												
预期结果	<pre>./guest1.sh 06:51:08 INFO hvisor.c:564: Calling ioctl to start zone: [rtthread] 06:51:08 INFO virtio_console.c:119: char device redirected to /dev/pts/8</pre> <table><tr><th>zone_id</th><th>cpus</th><th>name</th><th>status</th></tr><tr><td>0</td><td>0, 1</td><td>root-linux</td><td>running</td></tr><tr><td>1</td><td>2</td><td>rtthread</td><td>running</td></tr></table> <pre>root@G5000000000: /G5000000000# ./hvisor-zone-list</pre>	zone_id	cpus	name	status	0	0, 1	root-linux	running	1	2	rtthread	running
zone_id	cpus	name	status										
0	0, 1	root-linux	running										
1	2	rtthread	running										
预期结果	pass												

1.5.5 关闭虚拟机

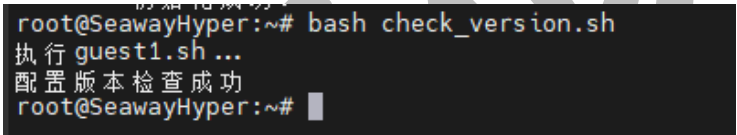
实验内容	rt-thread 关机功能实验
实验前提	root linux、rt-thread 正常运行
实验步骤	root Linux 执行以下命令： <pre>./hvisor zone shutdown -id <zone_id></pre> <pre>root@SeawayHyper:~/SeawayHyper/bin# ./hvisor zone shutdown -id 1 root@SeawayHyper:~/SeawayHyper/bin# ./hvisor zone list zone_id cpus name status 0 0, 1 root-linux running root@SeawayHyper:~/SeawayHyper/bin#</pre>
实验结果	虚拟机关机正常
实验结论	pass

1.5.6 虚拟机列表展示

实验内容	root linux、rt-thread 运行状态正常展示
实验前提	root linux、rt-thread 正常运行
实验步骤	root Linux 运行以下命令： <pre>./hvisor zone list</pre> <pre>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#</pre>
实验结果	虚拟机列表展示状态正常

预期结果	pass
------	------

1.5.7 配置版本检查

测试内容	配置版本检查
测试前提	1. root linux 正常运行； 2. 存在文件 check_version.sh
测试步骤	1. 运行 check_version.sh 
预期结果	配置版本检查成功
测试结果	pass

1.6 IVC 机制

1.6.1 定制 IVC 内核模块

测试内容	定制 IVC 内核模块
测试前提	root linux 正常运行
测试步骤	1. insmod ivc.ko 会输出以下配置信息

	<pre> [INFO 1] (hvisor:hypercall:122) handle hvc unit virtio, shared_region_addr = 0x1230bcf00 Send IvcInfo { len: 1, lvc_ct_upas: [138407936, 0,], lvc_shmem_upas: [138412632, 0,], lvc_ids: [0, 0,], lvc_irqs: [65, 0,], }, } root@irefly:~# </pre>
预期结果	ivc 模式成功加载
测试结果	pass

1.6.2 VM 调用其中的 IVC 服务进行通信

内容	用 IVC 服务进行通信
前提	1. root Linux 正常运行； 2. rt-thread 运行
步骤	1. 加载 ivc.ko 模块； 2. 运行当前目录下的/root/clktool 的 guest.sh 文件 2. 执行./ivc_demo send 命令。

	<pre> root@SeawayHyper:~# insmod ./ivc.ko root@SeawayHyper:~# ./ivc_demo send ivc_demo: starting ivc_id: 0, max_peers: 2 ivc_demo: zone0 sent: Hello from zone1, message #0 ivc_demo: Round Trip Time = 672 us ivc_demo: zone0 received: Hello from zone1, message #0 ivc_demo: [PASS] message matched! ivc_demo: zone0 sent: Hello from zone1, message #1 ivc_demo: Round Trip Time = 665 us ivc_demo: zone0 received: Hello from zone1, message #1 ivc_demo: [PASS] message matched! ivc_demo: zone0 sent: Hello from zone1, message #2 ivc_demo: Round Trip Time = 665 us ivc_demo: zone0 received: Hello from zone1, message #2 ivc_demo: [PASS] message matched! ivc_demo: zone0 sent: Hello from zone1, message #3 ivc_demo: Round Trip Time = 664 us ivc_demo: zone0 received: Hello from zone1, message #3 ivc_demo: [PASS] message matched! ivc_demo: zone0 sent: Hello from zone1, message #4 ivc_demo: Round Trip Time = 667 us ivc_demo: zone0 received: Hello from zone1, message #4 ivc_demo: [PASS] message matched! ivc_demo: zone0 sent: Hello from zone1, message #5 </pre>
预期结果	Root linux 、rt-thread 收数据成功
测试结果	pass

2. 管理域操作系统镜像（ VM0 Root Linux）

2.1 VM 全生命周期管理工具集 SeawayHyper Tools

2.2.1 VM 创建与配置

内容	VM 创建与配置功能
前提	Root linux 系正常运行
步骤	参考《混合部署系统后说明》，创建并启 rt-thread 系。

	<pre>06:59:15 WARN safe_cjson.c:24: hvisor.c:197 - [cJSON_GetObjectItem] Key 'gits_size' not found in JSON object 06:59:15 WARN hvisor.c:232: No gits fields in arch_config. 06:59:15 WARN safe_cjson.c:24: hvisor.c:283 - [cJSON_GetObjectItem] Key 'pci_config' not found in JSON object 06:59:15 WARN hvisor.c:285: No pci_config field found. 06:59:15 INFO hvisor.c:564: Calling ioctl to start zone: [rtthread] 06:59:15 INFO virtio_console.c:119: char device redirected to /dev/pts/9 zone_id cpus name status 0 0, 1 root-linux running 1 2 rtthread running 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#</pre>
预期结果	rt-thread 成功创建并且正常运行
测试结果	pass

2.2.2 VM 启动与停止

测试内容	VM 启动与停止功能测试
测试前提	Root linux 系统正常运行
测试步骤	参考《混合部署系统启动说明》启动 vm <pre>06:59:15 WARN safe_cjson.c:24: hvisor.c:197 - [cJSON_GetObjectItem] Key 'gits_size' not found in JSON object 06:59:15 WARN hvisor.c:232: No gits fields in arch_config. 06:59:15 WARN safe_cjson.c:24: hvisor.c:283 - [cJSON_GetObjectItem] Key 'pci_config' not found in JSON object 06:59:15 WARN hvisor.c:285: No pci_config field found. 06:59:15 INFO hvisor.c:564: Calling ioctl to start zone: [rtthread] 06:59:15 INFO virtio_console.c:119: char device redirected to /dev/pts/9 zone_id cpus name status 0 0, 1 root-linux running 1 2 rtthread running 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#</pre> ./hvisor zone shutdown -id <zone_id> <pre>06:59:15 WARN hvisor.c:285: No pci_config field found. 06:59:15 INFO hvisor.c:564: Calling ioctl to start zone: [rtthread] 06:59:15 INFO virtio_console.c:119: char device redirected to /dev/pts/9 zone_id cpus name status 0 0, 1 root-linux running 1 2 rtthread running 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# ./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#</pre>

预期结果	rt-thread 正常启动、停止
测试结果	pass

2.2.3 VM 删除与清理

测试内容	rt-thread 删除与清理功能
测试前提	Root linux 系统正常运行
测试步骤	./hvisor zone shutdown -id <zone_id> 停止 <pre> 06:59:15 WARN hvisor.c:285: No pci_config field found. 06:59:15 INFO hvisor.c:564: Calling ioctl to start zone: [rtthread] 06:59:15 INFO virtio_console.c:119: char device redirected to /dev/pts/9 zone_id cpus name status 0 1 root-linux running 1 2 rtthread running root@SeawayHyper:~/SeawayHyper# cd bin/ root@SeawayHyper:~/SeawayHyper/bin# ./hvisor zone list zone_id cpus name status 0 1 root-linux running 1 2 rtthread running root@SeawayHyper:~/SeawayHyper/bin# ./hvisor zone shutdown -id 1 root@SeawayHyper:~/SeawayHyper/bin# ./hvisor zone list zone_id cpus name status 0 1 root-linux running root@SeawayHyper:~/SeawayHyper/bin# </pre>
预期结果	rt-thread 成功删除
测试结果	pass

3. 实时嵌入式负载域操作系统镜像支持 (VM1)

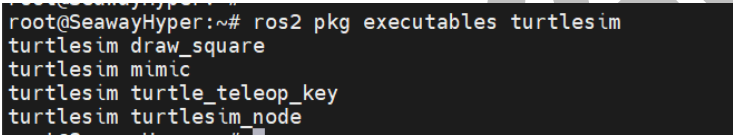
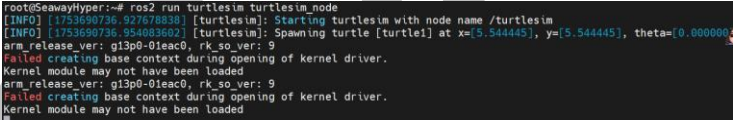
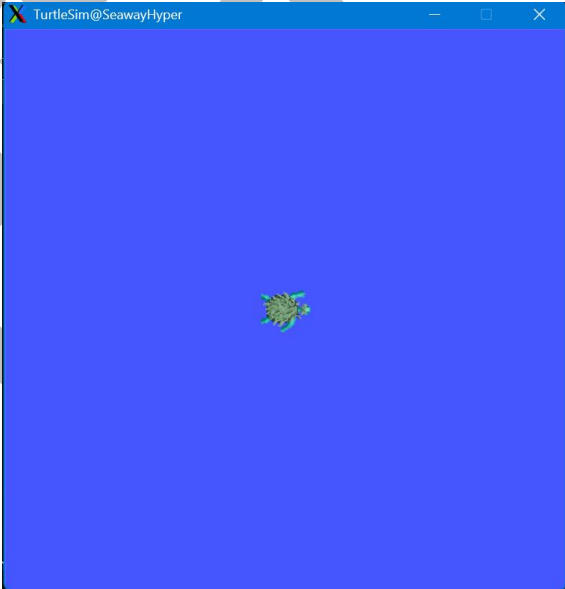
3.1 支持 RTOS VM

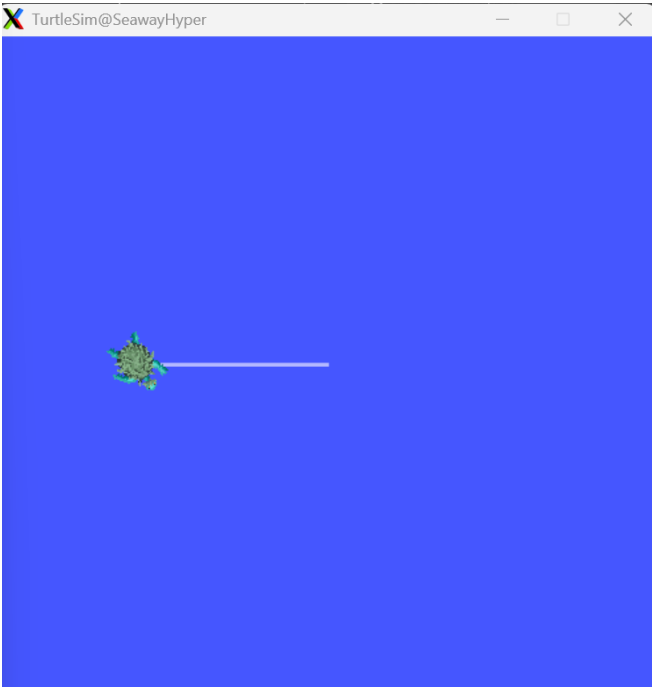
3.1.1 RTOS 的启动和运行

实验内容	RTOS 的启动和运行功能实验												
实验前提	Root linux 正常运行												
实验步骤	参考《混合部署系统启动说明》 <pre>10:01:41 WARN safe_cjson.c:24: hvisor.c:283 - [cJSON_GetObjectItem] Key 'pci_config' not found in JS 10:01:41 WARN hvisor.c:285: No pci_config field found. 10:01:41 INFO hvisor.c:564: Calling ioctl to start zone: [rtthread] 10:01:41 INFO virtio_console.c:128: char device redirected to /dev/pts/0 10:01:41 INFO virtio_console.c:128: char device redirected to /dev/pts/1</pre> <table><tr><th>zone_id</th><th>cpus</th><th>name</th><th>status</th></tr><tr><td>0</td><td>0, 1</td><td>root-linux</td><td>running</td></tr><tr><td>1</td><td>2</td><td>rtthread</td><td>running</td></tr></table> <pre>root@SeawayHyper:~/threevms#</pre> <pre>heap: [0x40245000 - 0x44245000] uncached_heap: BASE = 0x44245000, SIZE = 0x00100000 \ / - RT - Thread Operating System / \ 5.2.1 build Jul 10 2025 21:20:31 2006 - 2024 Copyright by RT-Thread team lwIP-2.0.3 initialized! Dev mac addr : a6 f1 d9 2a 82 c8 eth device init success : e1 virtio-init mmio_base: 0xff9e0000! rt_device_register succed [I/sal.skt] Socket Abstraction Layer initialize success. [I/rtdm.mnt] File system initialization done Hi, this is RT-Thread!! msh />init serial transport init get default allocator init msh /> msh /> msh /> msh /> msh /> msh /> msh /> msh /></pre> <pre>msh />ifconfig network interface device: e1 (Default) MTU: 1500 MAC: a6 f1 d9 2a 82 c8 FLAGS: UP LINK_DOWN INTERNET_DOWN DHCP_ENABLE ETHARP BROADCAST IGMP ip address: 0.0.0.0 gw address: 0.0.0.0 net mask : 0.0.0.0 dns server #0: 0.0.0.0 dns server #1: 0.0.0.0 msh /></pre>	zone_id	cpus	name	status	0	0, 1	root-linux	running	1	2	rtthread	running
zone_id	cpus	name	status										
0	0, 1	root-linux	running										
1	2	rtthread	running										
实验结果	RTOS 正常启动和运行												
实验结论	pass												

4 ros2 基本功能

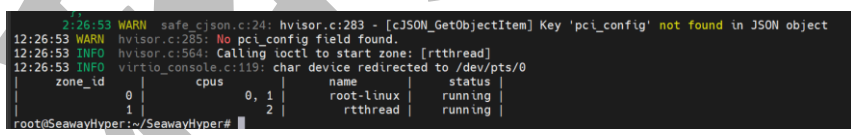
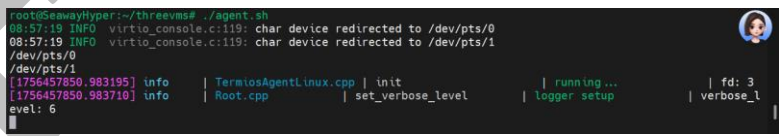
4.1 ros2 基本功能

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 <pre>ros2 pkg executables turtlesim</pre>  2. ros2 run 运行 turtlesim 打开一个终端，执行以下命令 <code>ros2 run turtlesim turtlesim_node</code>，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，<code>ros2 run turtlesim turtle_teleop_key</code>，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 <pre>root@SeawayHyper:~# ros2 node list /teleop_turtle /turtlesim</pre> 4. 查看 ros 的配置信息： <pre>export grep ros</pre> <pre>root@SeawayHyper:~# export grep ros declare -x AMENT_PREFIX_PATH="/opt/ros/humble" declare -x LD_LIBRARY_PATH="/opt/ros/humble/opt/rviz_ogre_vendor/lib:/opt/ros/humble/lib/aarch64-linux-gnu:/opt/ros/humble/lib" declare -x PATH="/opt/ros/humble/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin" declare -x PYTHONPATH="/opt/ros/humble/lib/python3.10/site-packages:/opt/ros/humble/local/lib/python3.10/dist-packages:/opt/firefly/libfirefly-ai/python/:" root@SeawayHyper:~#</pre>
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

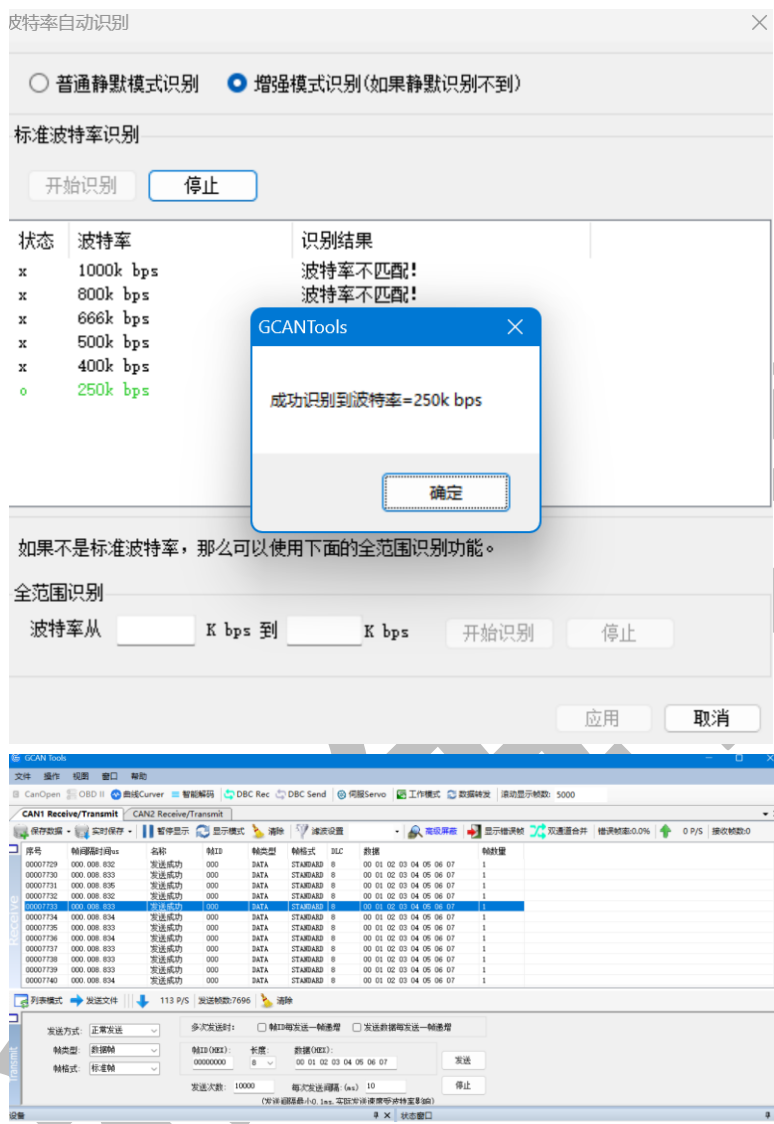
5.1 CAN 通信交互

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 / guest1rt.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!
```

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; <pre> 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/Mi00 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:~# </pre>
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。 <pre> 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 001: ID 1d6b:0001 Linux Foundation 1.1 root hub Bus 002 Device 003: ID 2c7c:0125 Quectel Wireless Solutions Co., Ltd. EC25 LTE modem 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:~# [568.585361] usb 1-1.3: new high-speed USB device number 4 using ehci-platform [568.695157] usb 1-1.3: New USB device found, idVendor=2c7c, idProduct=0125, bcdDevice= 3.18 [568.695165] usb 1-1.3: New USB device strings: Mfr=1, Product=2, SerialNumber=3 [568.695169] usb 1-1.3: Product: EC20-CE-HDLG [568.695173] usb 1-1.3: Manufacturer: Quectel [568.695177] usb 1-1.3: SerialNumber: 0123456789ABCDEF root@SeawayHyper:~# </pre>
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。 <pre> root@SeawayHyper:~# ls /dev/ttyUSB* /dev/ttyUSB0 /dev/ttyUSB1 /dev/ttyUSB2 /dev/ttyUSB3 root@SeawayHyper:~# </pre>
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

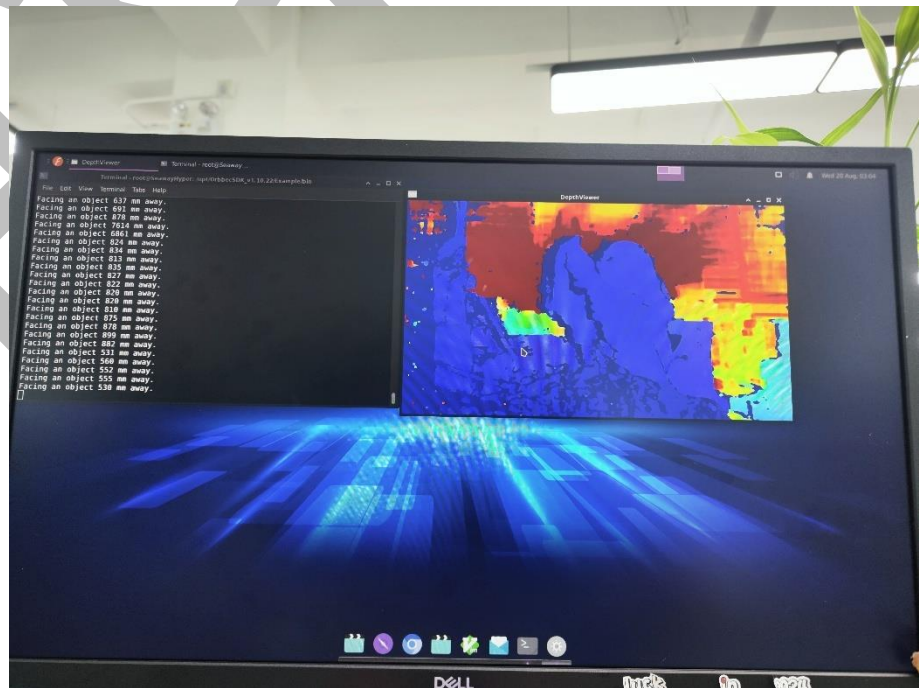
5.5 深度相机测试

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 <code>lsusb</code>，查找奥比中光设备，再执行 <code>dmesg grep usb</code>，筛选 USB3.0 控制器日志 <pre> 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 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 003: ID 2bc5:0407 Orbbec 3D Technology International, Inc Astra Mini S 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:~# </pre> <pre> [5.195935] usb 1-1.2: new high-speed USB device number 3 us [5.293343] usb 1-1.2: New USB device found, idVendor=2bc5, i [5.293361] usb 1-1.2: New USB device strings: Mfr=1, Product [5.293369] usb 1-1.2: Product: ASTRA [5.293376] usb 1-1.2: Manufacturer: ORBBEC root@SeawayHyper:~# </pre> 2. 查看视频设备节点：<code>ls /dev/video*</code> <pre> root@SeawayHyper:~# ls /dev/video* /dev/video-dec0 /dev/video-enc0 root@SeawayHyper:~# </pre>
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

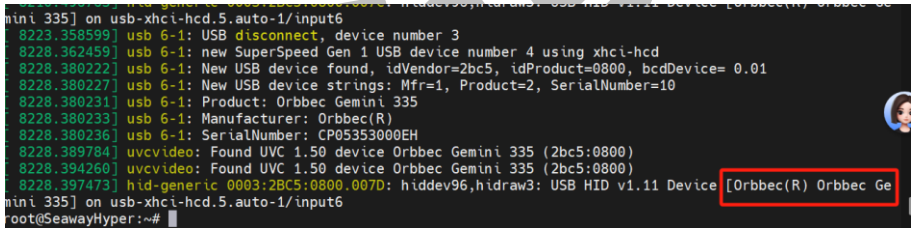
5.5.2 深度相机启动

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: <code>sudo apt install build-essential freeglut3 freeglut3-dev libusb-1.0-0-dev ros-humble-backward-ros libglfw3-dev nlohmann-json3-dev libgoogle-glog-dev</code> 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: <code>cd /opt/OrbbecSDK_v1.10.22/Example/build</code> <code>cd</code> <code>cmake ..</code> <code>cmake --build . --config Release</code> <code>cd /opt/OrbbecSDK_v1.10.22/Example/bin</code> 执行 <code>./DepthViewer</code>



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

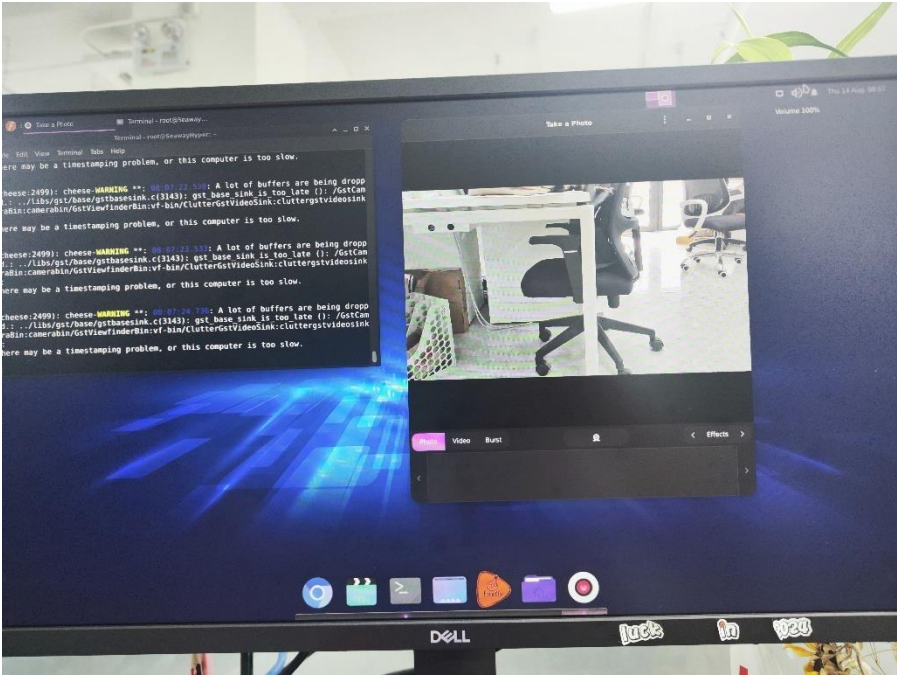
5.6.1 RGB（usb 摄像头）相机识别

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 <code>sudo apt install ros-humble-usb-cam ros-humble-image-view</code>

测试步骤	1. 连接 USB 摄像头后，执行以下命令查看设备节点 <pre>lsusb</pre> <pre>dmesg</pre> <pre>ls /dev/video*</pre>  <pre>root@SeawayHyper:~# ls /dev/video* /dev/video0 /dev/video1 /dev/video-dec0 /dev/video-enc0 root@SeawayHyper:~#</pre>
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） <pre>sudo apt-get install cheese</pre> 2. 装好后，用命令：cheese，即可打开。如果指定打开 video0，输入命令： <pre>cheese -d /dev/video0</pre>

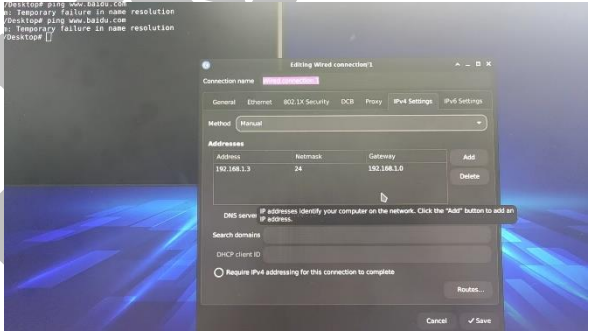
	
预期结果	摄像头可成功捕捉画面
实际测试结果	pass

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 <pre> 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:7259 Huasheng Electronics USB2.0 HUB Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub root@SeawayHyper:~# </pre>

	<pre>4.723919] usbcore: registered new interface driver usbhid [4.723922] usbhid: USB HID core driver [4.727370] 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 [5.302278] 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:~#</pre> 2. ls /dev/ttyUSB* 检查虚拟串口数量 <pre>root@SeawayHyper:~# ls /dev/ttyUSB* /dev/ttyUSB0 root@SeawayHyper:~#</pre>
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  The screenshot shows a terminal window with the following commands and output: <pre>root@SeawayHyper:~# ifconfig root@SeawayHyper:~# ifconfig eth0 192.168.1.3 netmask 255.255.255.0 root@SeawayHyper:~# ping -c 1 192.168.1.200 PING 192.168.1.200: 56 data bytes 64 bytes from 192.168.1.200: icmp_seq=0 ttl=64 time=0.045 ms 64 bytes from 192.168.1.200: icmp_seq=1 ttl=64 time=0.045 ms 64 bytes from 192.168.1.200: icmp_seq=2 ttl=64 time=0.045 ms 64 bytes from 192.168.1.200: icmp_seq=3 ttl=64 time=0.045 ms 64 bytes from 192.168.1.200: icmp_seq=4 ttl=64 time=0.045 ms 64 bytes from 192.168.1.200: icmp_seq=5 ttl=64 time=0.045 ms 64 bytes from 192.168.1.200: icmp_seq=6 ttl=64 time=0.045 ms 64 bytes from 192.168.1.200: icmp_seq=7 ttl=64 time=0.045 ms 64 bytes from 192.168.1.200: icmp_seq=8 ttl=64 time=0.045 ms 64 bytes from 192.168.1.200: icmp_seq=9 ttl=64 time=0.045 ms ^C root@SeawayHyper:~#</pre> The network settings window shows the following configuration: - Connection name: eth0 - Method: Manual - Addresses: 192.168.1.3/24 - Gateway: 192.168.1.1 - Search domains: (empty) - Require IPv4 addressing for this connection to complete: (checked)
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

```
07:01:01.403149 IP 192.168.1.200.2369 > 192.168.1.3.2368: UDP, length 1206
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

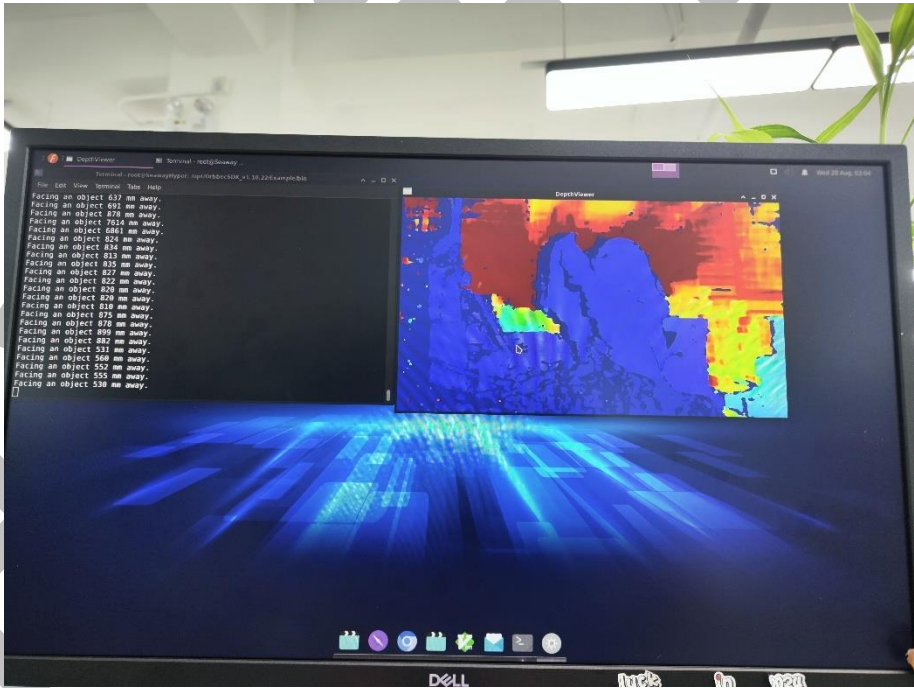
5.9.1 双目相机识别

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

	<pre>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 SiGamma 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:~#</pre> Dmesg <pre> [822.358599] usb 6-1: USB disconnect, device number 3 [822.362459] usb 6-1: new SuperSpeed Gen 1 USB device number 4 using xhci-hcd [822.380222] usb 6-1: New USB device found, idVendor=2bc5, idProduct=0800, bcdDevice= 0.01 [822.380227] usb 6-1: New USB device strings: Mfr=1, Product=2, SerialNumber=10 [822.380231] usb 6-1: Product: Orbbec Gemini 335 [822.380233] usb 6-1: Manufacturer: Orbbec(R) [822.380236] usb 6-1: SerialNumber: CP05353000EH [822.389784] uvcvideo: Found UVC 1.50 device Orbbec Gemini 335 (2bc5:0800) [822.394260] uvcvideo: Found UVC 1.50 device Orbbec Gemini 335 (2bc5:0800) [822.397473] hid-generic 0003:2BC5:0800.007D: hiddev96,hidraw3: USB HID v1.11 Device [Orbbec [830.493261] usb 4-1: USB disconnect, device number 4 [830.493707] cp210x ttyUSB0: cp210x converter now disconnected from ttyUSB0 root@SeawayHyper:~#</pre> ls /dev/ <pre> 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 hvsior 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 mmcblk0p9 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:~#</pre> Subscribing 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：

	<pre>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</pre>
测试步骤	1. OrbbecSDK_v1.10.22 上传至 opt 目录下，执行以下命令： <pre>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</pre> 
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

测试内容	网口 1 网口支持
测试前提	1. Rt-thread 系统正常运行； 2. 网口正常连接
测试步骤	1. ifconfig <pre>msh />ifconfig network interface device: e1 (Default) MTU: 1500 MAC: 6e 88 03 c6 e6 d9 FLAGS: UP LINK_UP INTERNET_DOWN DHCP_ENABLE ETHARP BROADCAST IGMP ip address: 192.168.1.60 gw address: 192.168.1.1 net mask : 255.255.255.0 dns server #0: 192.168.10.8 dns server #1: 0.0.0.0 msh /></pre> 2. ping www.baidu.com; <pre>msh />ping www.baidu.com ping: not found specified netif, using default netdev e1. 60 bytes from 220.181.111.232 icmp_seq=1 ttl=55 time=0 ms 60 bytes from 220.181.111.232 icmp_seq=2 ttl=55 time=0 ms 60 bytes from 220.181.111.232 icmp_seq=3 ttl=55 time=0 ms 60 bytes from 220.181.111.232 icmp_seq=4 ttl=55 time=0 ms</pre> 3. server 端：iperf -s -p 5001 <pre>root@SeawayHyper:~/SeawayHyper# iperf -s -p 5001 ----- Server listening on TCP port 5001 TCP window size: 128 KByte (default) ----- [1] local 192.168.1.32 port 5001 connected with 192.168.1.60 port 49154</pre> client 端：iperf -c 192.168.1.32 -p 5001

	<pre> msh />iperf -c 192.168.1.32 -p 5001 msh />[I/iperf] Connect to iperf server successful! [I/iperf] iperfc01: 166.4810 Mbps! [I/iperf] iperfc01: 166.6580 Mbps! [I/iperf] iperfc01: 166.7360 Mbps! [I/iperf] iperfc01: 166.6840 Mbps! [I/iperf] iperfc01: 166.6580 Mbps! [I/iperf] iperfc01: 166.2320 Mbps! [I/iperf] iperfc01: 166.6640 Mbps! [I/iperf] iperfc01: 166.6380 Mbps! [I/iperf] iperfc01: 166.6380 Mbps! [I/iperf] iperfc01: 166.6380 Mbps! </pre>
☑期☑果	网口 1 网☑通信正常
☑☑☑果	pass