reComputer RK3576

User Manual

Revision History

Version	Date	Description
1.0	2026-06-18	

Contents

C1. Introduction

The reComputer RK3576 is powered by the high-performance RK3576 processor (4x Cortex-A72 @2.2GHz + 4x Cortex-A53 @2.0GHz) and an ARM Mali-G52 MC3 GPU. This device offers 4GB, 8GB, or 16GB of LPDDR5 RAM and features an NPU with 6 TOPS of computing power, supporting INT4/INT8/INT16/FP16 operations. In terms of multimedia, it supports video encoding up to 4K@60fps and video decoding up to 8K@30fps.

Regarding interface requirements, it is equipped with two Gigabit Ethernet ports (one of which supports PoE), along with onboard Wi-Fi 6 and Bluetooth 5.4. The USB interfaces include 1x USB 3.0, 3x USB 2.0, and 1x Type-C port supporting OTG and DP output. It features three display outputs: Display, Type-C (DP 1.4), and a 4-lane MIPI DSI, and comes with an M.2 Key M 2280 slot (PCIe 2.1 x1) for an NVMe SSD or AI accelerator. Thanks to its dual Gigabit Ethernet, onboard wireless communication, and multiple display interfaces, this device is highly suitable for deployment in smart retail terminals, lightweight edge gateways, and building automation scenarios.

1.1 Features

Robust Hardware Foundation & Scalable Computing Power

- **Core Processor:** Powered by the Rockchip **RK3576**, featuring an Octa-core architecture (4x Cortex-A72@2.2GHz + 4x Cortex-A53@2.0GHz) alongside an Arm Mali-GPU.
- **High-Speed & Large Capacity Memory:** Supports up to **16GB LPDDR5**, ensuring exceptional memory throughput for on-device Large Language Models (LLMs) and demanding AI workloads.
- **Built-in & Expandable NPU:** Delivers **6 TOPS** of native on-device AI performance. Features a PCIe 2.1 (M.2 M Key) slot for hardware acceleration expansion, scaling up to **26 TOPS**.

Comprehensive AI Model Support & Multimodal Capabilities

- **Covers Three Major AI Scenarios:** Purpose-built to handle Perception AI, Generative AI, and Physical AI applications.
- **Seamless On-Device Execution of Mainstream Models:**
  - **Visual Interaction:** Achieves 77.9FPS running YOLOv11 (640x640), and fully supports vision-language models (VLMs) like Qwen2.5-VL 3B, as well as MobileNet, RetinaFace, and CLIP.
  - **On-Device LLMs:** Natively supports the deployment of **DeepSeek-R1-7b**.
  - **Real-time Voice Interaction:** Features low-latency Speech-to-Text (Whisper-base 2.0s) and Text-to-Speech real-time processing capabilities.

- **Robust Underlying Frameworks:** Broad compatibility with mainstream frameworks including ONNX, PyTorch, TensorFlow, and Caffe, supported by the mature RKNN-Toolkit2 toolchain.

- **Versatile Network Connectivity & Peripheral Expansion**
  - **Comprehensive Network Communication:** Equipped with 2x Gigabit Ethernet ports, Wi-Fi 6, and Bluetooth 5.4. Additionally supports PoE, 4G, LoRaWAN, Wi-Fi Halow, and SIM card expansion for diverse IoT deployments.
  - **Exceptional Hardware Expansibility:** Integrates a rich array of I/O interfaces, including MIPI-CSI, MIPI DSI, DP1.4, USB 3.0, PCIe 2.1, CAN FD, UART, PWM, I2C, and SPI.
- **Flagship Multimedia & Display Output**
  - **Versatile Display Interfaces:** Provides four types of display connectivity options, including HDMI 2.1, DP 1.4, MIPI-DSI, and USB Type-C, offering flexible interface selections for independent or duplicated multi-screen configurations.
  - **Hardware-Accelerated Codec:**Features an H.265 / H.264 / AV1 / AVS2 multivideo decoder supporting up to 8K@30fps, paired with an H.264 / H.265 multivideo encoder supporting up to 4K@60fps for efficient media streaming.
- **Frictionless Software Ecosystem & One-Click Deployment**
  - **Out-of-the-Box OS:** Comes pre-installed with **Armbian** and offers Multiple OS Support.
  - **reComputer AI Lab Ecosystem:** Engineered for rapid deployment with minimal complexity. Utilize simple Docker commands to instantly pull and run containers (e.g., launching `deepseek-r1-distill-qwen:1.5b` in seconds). Explore the full range of supported pre-configured environments on the https://sensecraft.seeed.cc/ai-lab/models .
  - **Extensive Developer Resources:** Grants access to over 100 ready-to-use AI models, 100+ open-source project libraries, and comprehensive, step-by-step tutorials.

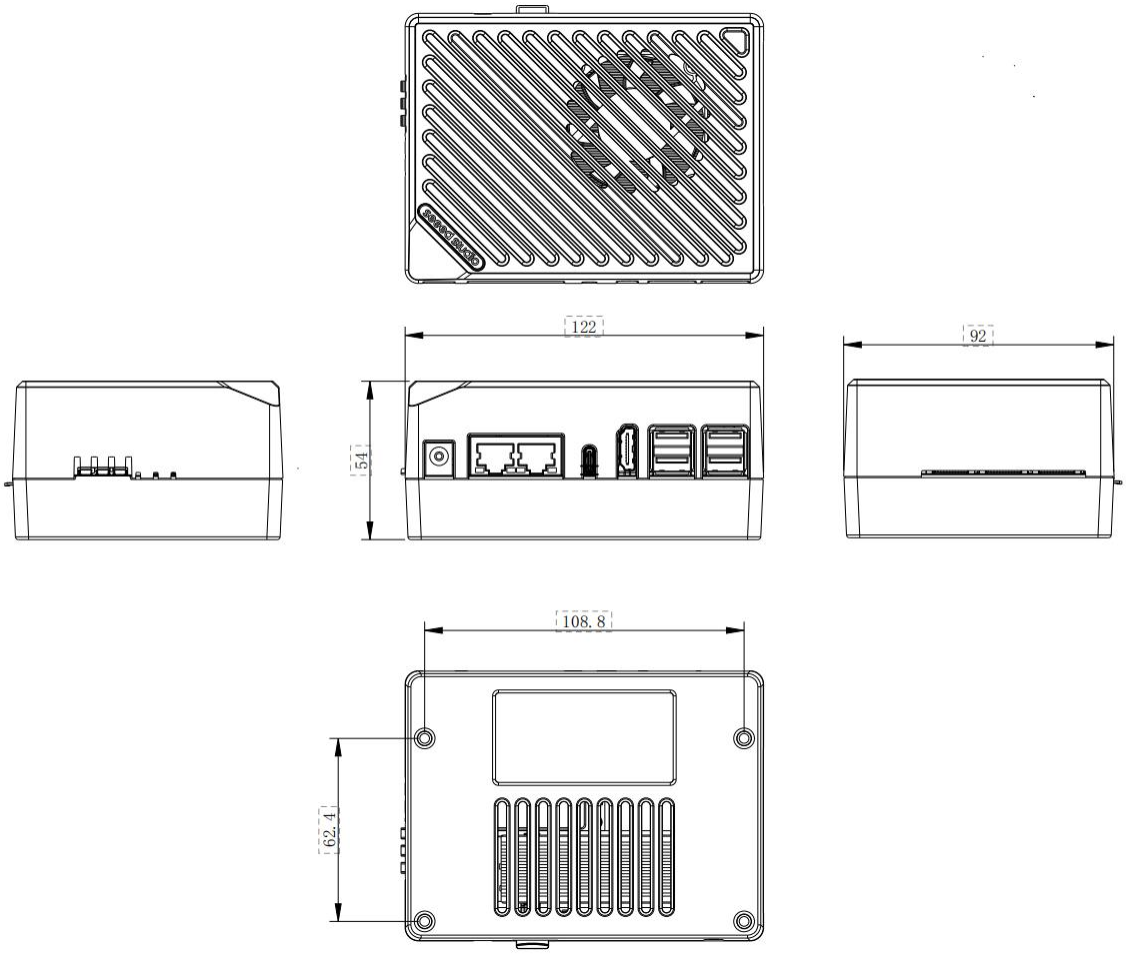
1.2 Specifications

Parameter	Description	
Hardware Specification		
Product Series	RK3576-20	RK3576-30
SOC	Rockchip RK3576	

CPU	4x Cortex-A72@2.2GHz + 4x Cortex-A53@2.0GHz	
GPU	ARM Mali-G52 MC3	
NPU	INT8@6TOPS; Supporting INT4/8/16/FP16/BF16/TF32 mixed operations	
Operating System	ArmBian OS	
RAM	4GB	8GB
System Specification		
Power Input	DC 9V~19V	
PoE	1 x PoE PD*	
Power	Yes	
Recover	Yes	
MaskROM	Yes	
Interface		
Ethernet	1x Gigabit Ethernet;	
	1x Gigabit Ethernet with PoE support(Additional PoE Module Required)	
USB	1x Type A USB 3.0; 3x Type A USB 2.0;	
	1x Type C for OTG & DP	
Display	1x Display; 1x DP1.4 via USB Type C port; 1x 4-lane MIPI DSI(22pin)	
GPIO	1x 40-Pin Expansion header	
SIM Card	1x Nano SIM Card Slot	

SD Card	1x Micro SD Card Slot
M.2 Slot	1x M.2 Key M 2280 slot (PCIe 2.1x 1), supporting either an NVMe SSD or an AI accelerator
Mini-PCIe	1x miniPCIe for 4G/LoRaWAN/Wi-Fi Halow
LED	1x Power; 1x Status; 1x User
Button	1x Power; 1x Recovery; 1x MaskROM
Audio	1x 3.5mm Headphone Jack with Microphone Input
CSI Camera	2x 4-lane MIPI CSI(22pin)
FAN	1x 4-PIN Header
RTC	1x 2-PIN Header
Wireless Communication	
Wi-Fi 2.4/5.0 GHz	Onboard WiFi6 with FPC Antenna
BLE 5.0	BT5.4 with FPC Antenna
LoRa®	USB LoRa®*/SPI LoRa®*
4G Cellular	4G LTE*
Standards	
Certification	CE, FCC
	TELEC
	RoHS
Ambient Conditions	
Operating Temperature	0 ~ 60°C

Others	
Heat Dissipation	Fan
Warranty	1 years
Production Lifetime	Until January 2036
Statement	Options marked with * require additional purchase according to the accessories list.



C2. Getting Start With reComputer RK3576

2.1 Hardware Connection

This page describes the required cables and basic connection order. Before starting with your reComputer RK3576, please follow the steps below to connect your peripherals:

Insert the SD Card

Insert the factory-included SD card into the microSD card slot on the device.

Connect Input Devices

Plug your keyboard and mouse into the USB 2.0 or USB 3.0 ports on the device.

Connect a Display

Connect your monitor to the HDMI port using an HDMI cable.

Connect to the Network

Insert a network cable into the Gigabit Ethernet port for a stable internet connection.

Connect Power

Finally, plug the power adapter into the DC Jack. The device will power on automatically.

Interface Overview

Please refer to the image below to identify the location of each port:

2.2 Configuration on First Boot

This guide will walk you through the initial setup of the Armbian system based on the RK3576 chip. After flashing the image and powering it on for the first time, the system will enter an interactive configuration interface.

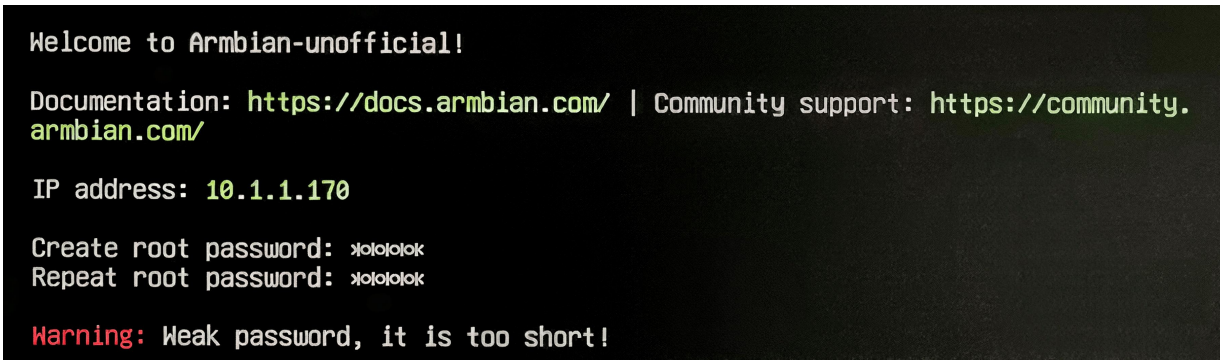
Set the Root Password

The primary task upon system startup is to secure the root account.

Action: Enter the desired new root password twice as prompted.

Note: Characters will not appear on the screen while typing (this is standard Linux security behavior).

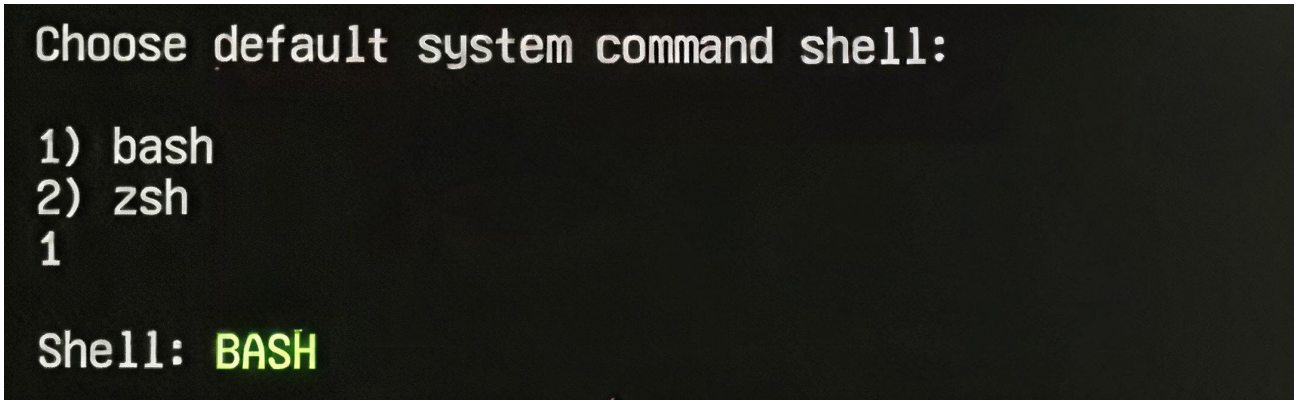
Warning: If the password is too short, the system will display a "Warning: Weak password" message. It is highly recommended to use a more complex combination.



Select the Default Shell

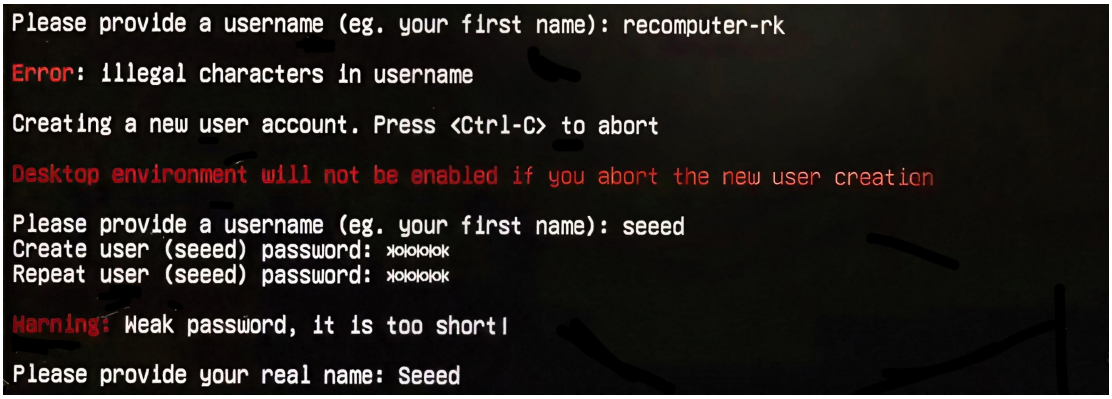
The system will ask you to specify your preferred command-line interpreter (Shell). Options: 1) bash or 2) zsh. Recommendation: Select 1) bash for better compatibility and its status as the industry standard for scripts and tutorials.

Action: Type 1 and press Enter.



Create a Regular User Account

For enhanced system security, Armbian requires the creation of a non-root user for daily operations. Username Rules: Use lowercase letters and numbers only. Do not include hyphens - or underscores _ (for example, recomputer-rk would be flagged as an invalid character). Action: Enter your desired username (e.g., seeed). Set and confirm the password for this user. Enter your Full Name (you can simply press Enter to skip this). Privileges: This user will be automatically granted sudo privileges, allowing you to run administrative commands safely.



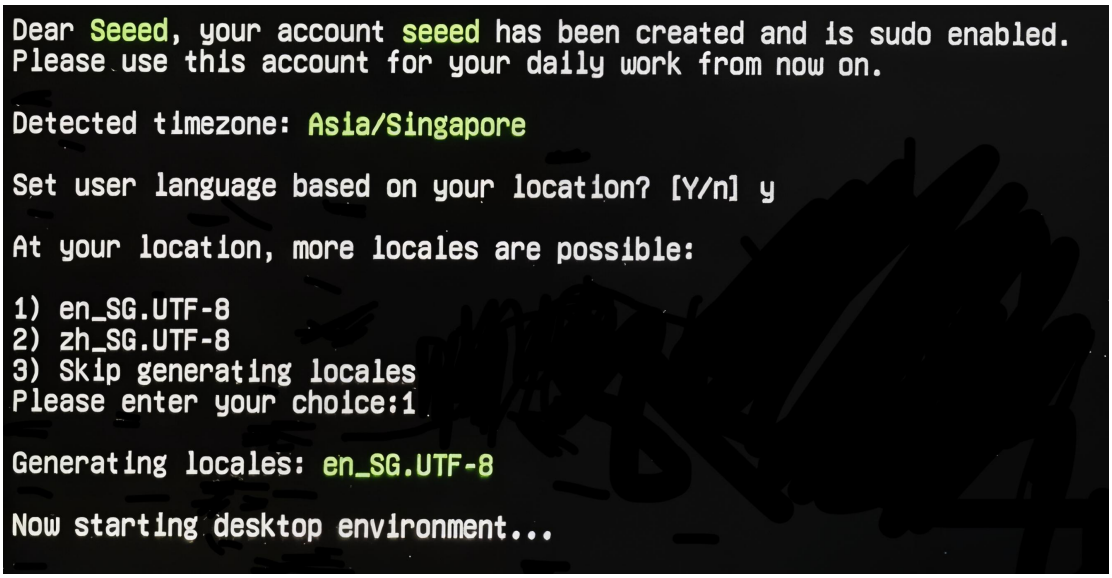
Timezone and Locale Settings

The system will attempt to automatically detect your location based on your network connection.

Timezone: Confirm whether the detected timezone (e.g., Asia/Singapore) is correct.

Locales (Language): * When asked Set user language based on your location? [Y/n], type y.

Select the specific language code from the provided list (e.g., selecting 1 for en_SG.UTF-8).



Please wait a moment. You will be directed to the login interface. Enter your configured username and password to access the desktop.

Ready to boot up your reComputer RK3576/RK3588? The system image is already pre-flashed, so you only need to make a few simple physical connections. Click the video below to watch how to properly connect the power and peripherals, and boot to the desktop for the first time:https://www.youtube.com/watch?v=M7u0K6UIsP4

2.3 Resource Download Summary

2.3.1 System Image Download

Development Tools

Tool Name	Description	Download Link
MobaXterm	Terminal tool for Serial/SSH (Portable v25.2)	Download Home Edition
balenaEtcher	An application for flashing OS images to SD cards	Official Site
DriverAssitant	Rockchip USB Driver Assistant (v5.1.1)	(Link TBD)

C3. Flash OS

3.1 Install the system on the MicroSD Card

File Download

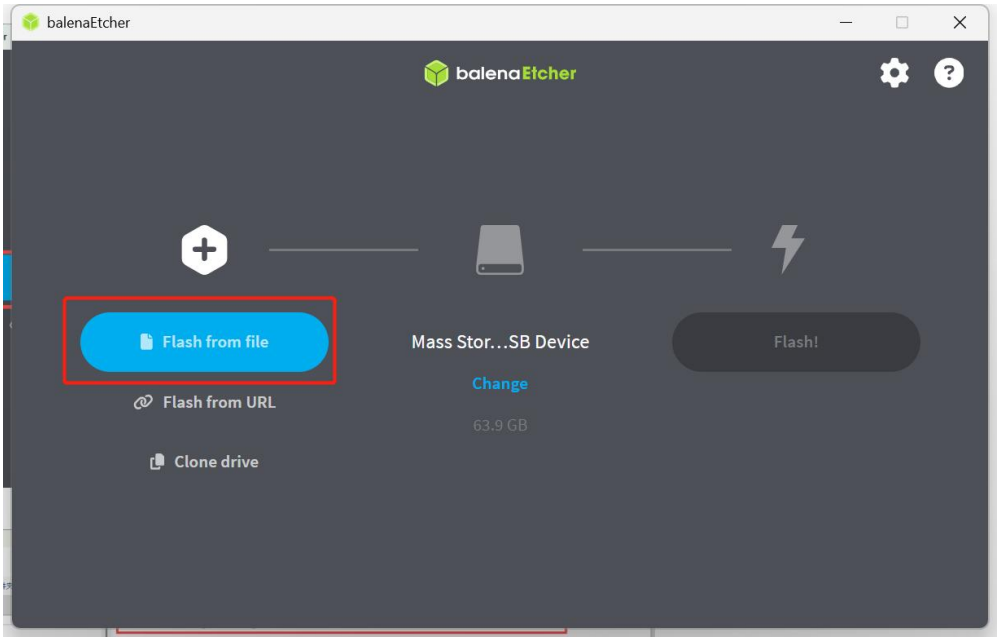
Download the system image from the official resource download page.

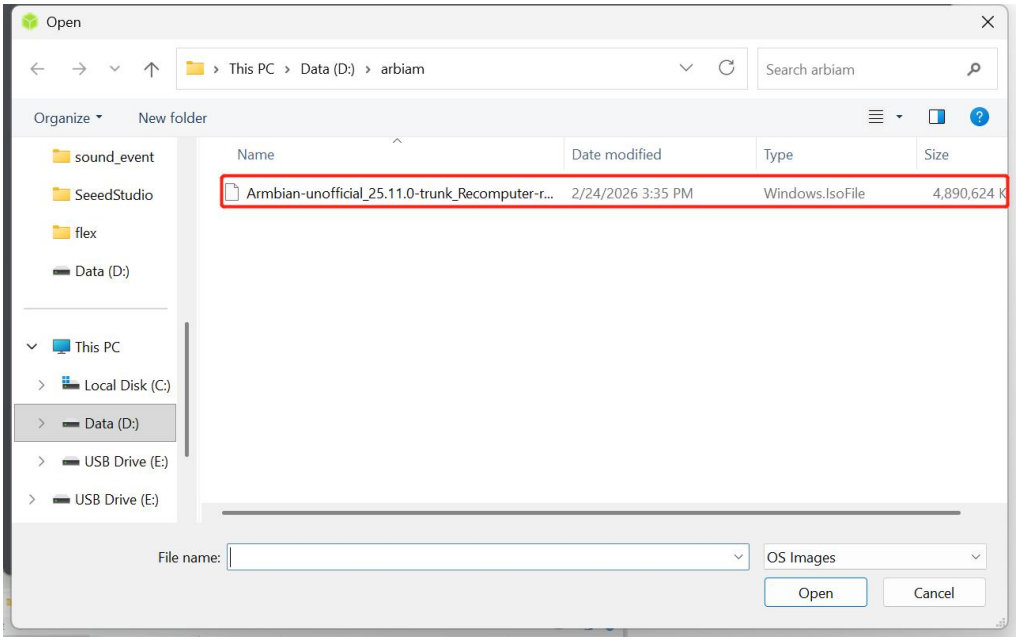
Prepare MicroSD Card

Insert the MicroSD card into an SD card reader, then connect the reader to your computer's USB port.

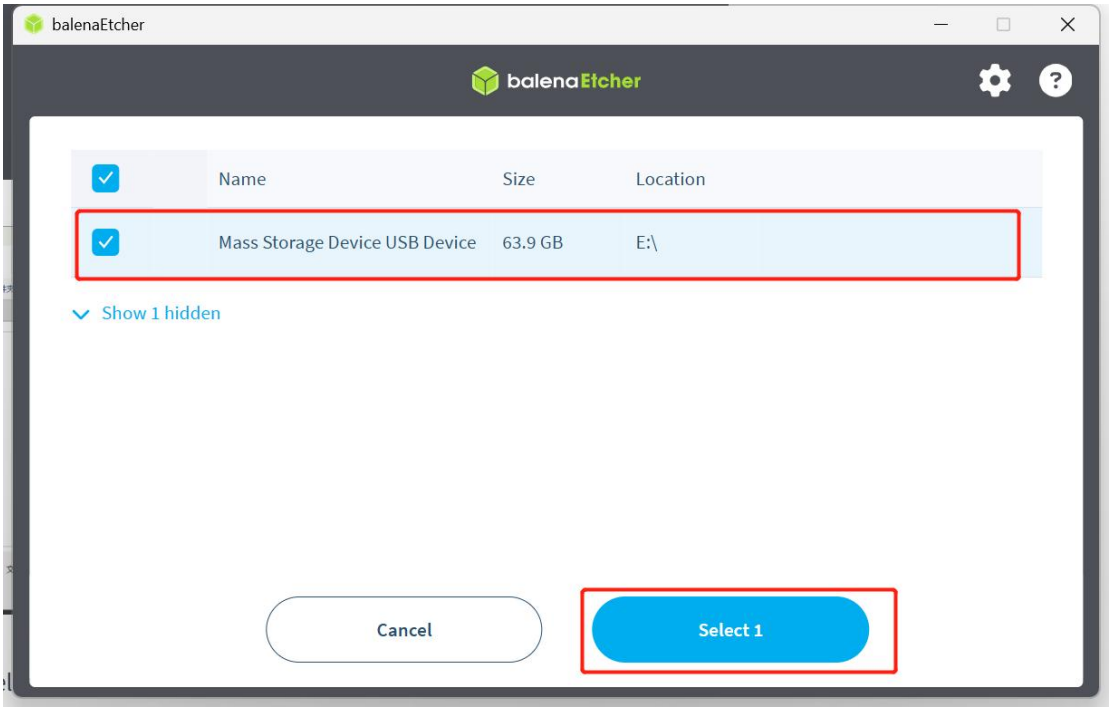
Flash the Image

It is recommended to use balenaEtcher to complete the flashing process. Click "Flash from file" and locate the downloaded .img file.

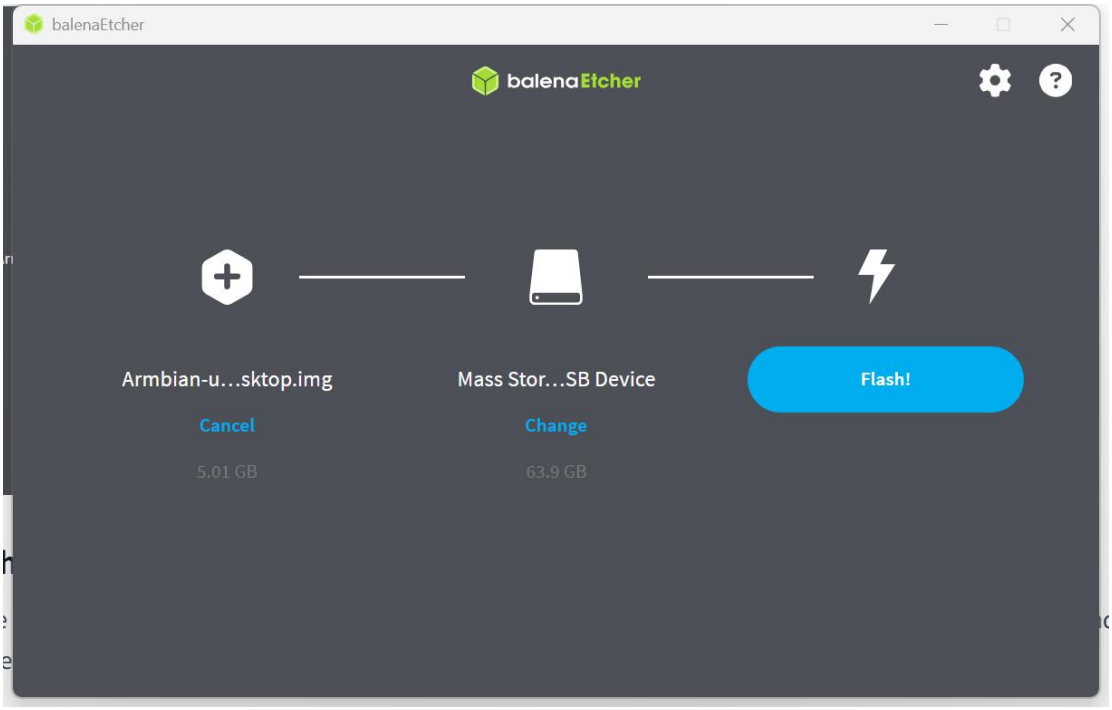




Click "Select target" to choose your MicroSD card.

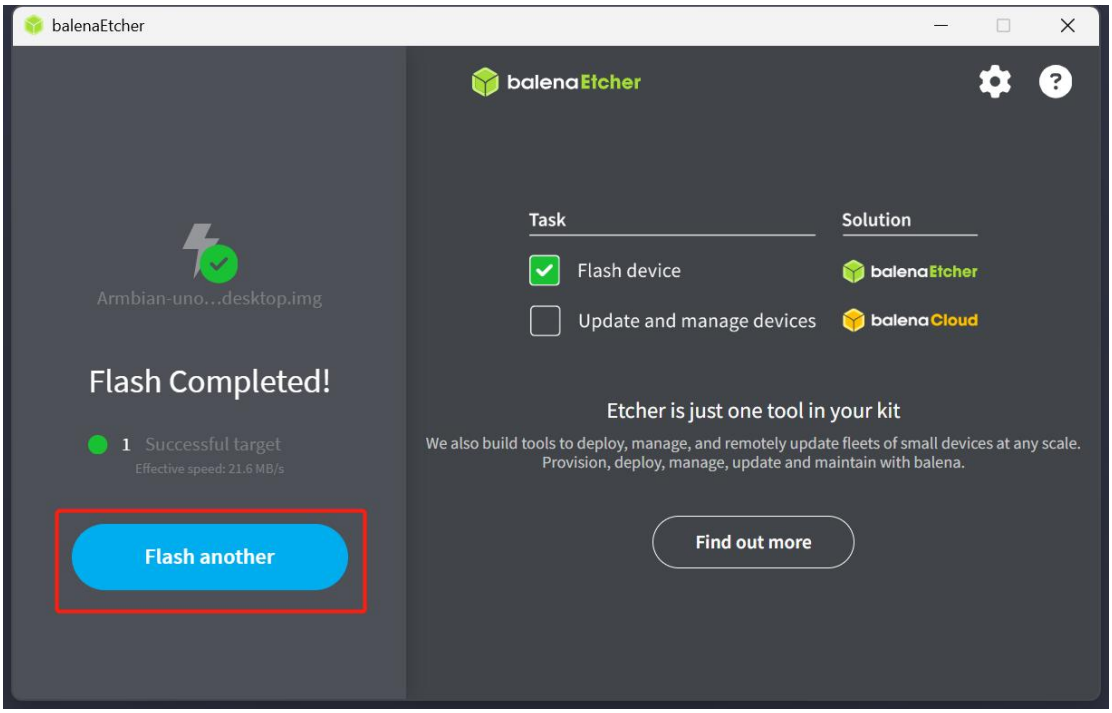


Click "Flash!" and wait for the process to complete.



Boot the System

Once the image is successfully written to the MicroSD card, insert it into the MicroSD card slot (as shown below) and power on the device. The system will begin to boot, and the desktop will be displayed on your HDMI monitor.



Log in to the System

Refer to the First Boot tutorial to complete the initial system startup and configuration settings.

3.2 Install the system on the SSD

The underlying SPI Flash firmware of the reComputer Industrial (RK3576) natively supports NVMe booting. On hardware models without onboard eMMC, there is no need to keep a MicroSD card inserted for assistance. The MicroSD card acts purely as a **one-time flashing medium**. Once the OS is deployed, the device can achieve a 100% independent native cold boot directly from the onboard M.2 NVMe SSD, delivering optimal industrial-grade reliability and performance.

Prerequisites

Hardware: reComputer RK3576 devkit with an NVMe SSD properly installed in the M.2 slot.

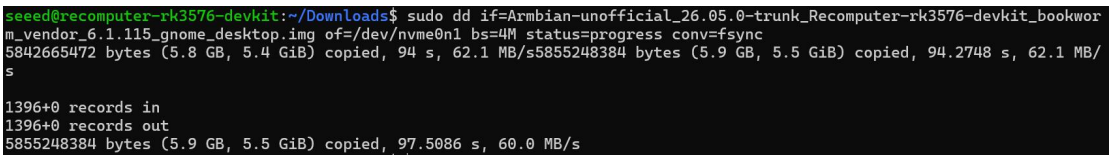
Preparation: A temporary MicroSD card flashed with an interim Armbian OS to serve as the initial deployment environment.

Core Deployment Steps

Step 1: Flash the Image onto the Target NVMe Drive

Boot into the temporary SD card system, and use the native `dd` tool to clone the system image directly into the physical NVMe drive · eg:

```
Plain Text
cd ~/Downloads
sudo dd if=Armbian-unofficial_26.05.0-trunk_Recomputer-rk3576-devkit_bookworm_vendor_6.1.115_gnome_desktop.img of=/dev/nvme0n1 bs=4M status=progress conv=fsync
```



Step 2: Repair GPT Partition Table & Resize Partition

Once flashed, the root partition must be extended to the physical boundaries of the SSD using `parted`:

```
Bash
# 1. Fix GPT Partition Table (Type 'F' and hit Enter when the Fix/Ignore prompt appears)
sudo parted /dev/nvme0n1 print Fix

# 2. Expand the 1st partition (rootfs) to 100% of the available space
sudo parted /dev/nvme0n1 resizepart 1 100%
```

```
# 3. Force the kernel to re-read the updated partition layout
sudo partprobe /dev/nvme0n1
```

Step 3: Expand the Ext4 Filesystem

Resize the Ext4 file system online so that the OS recognizes the full hardware capacity:

```
Bash
sudo resize2fs /dev/nvme0n1p1
```

Hardware Post-Execution & Native Verification

Step 1: Power Off and Remove the SD Card

After completing the steps above, no modification to any `/boot/armbianEnv.txt` is required. Power off the device immediately:

```
Plain Text
sudo poweroff
```

CRITICAL ACTION: After power-off, **completely remove the MicroSD card from its slot**, leaving only the M.2 NVMe SSD on the board.

Step 2: Final Verification

Power on the device again. The underlying U-Boot will automatically scan and bootstrap the system straight from the NVMe drive into the fresh Armbian configuration wizard. Run the following after setup:

```
Plain Text
df -h /
```

Successful Verification Output:

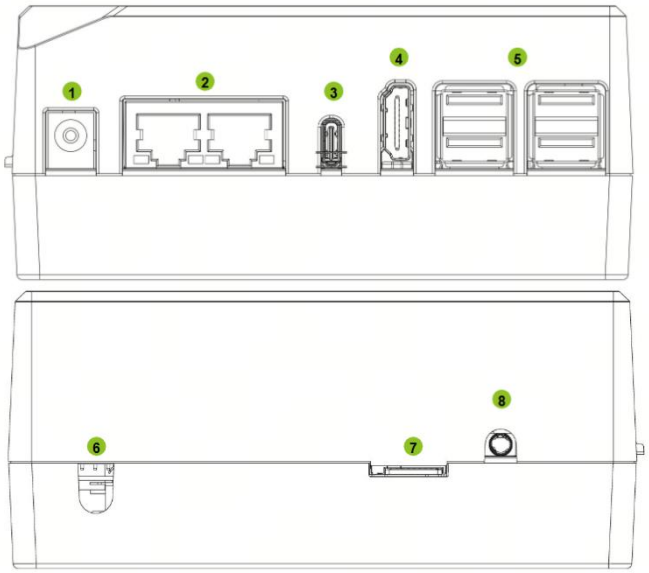
```
seeed@recomputer-rk3576-devkit:~$  df -h /
Filesystem      Size  Used Avail Use% Mounted on
/dev/nvme0n1p1  118G  4.4G  112G   4% /
```

The root directory `/` is natively mounted on `/dev/nvme0n1p1` with **100G+** available space, running flawlessly without an SD card.

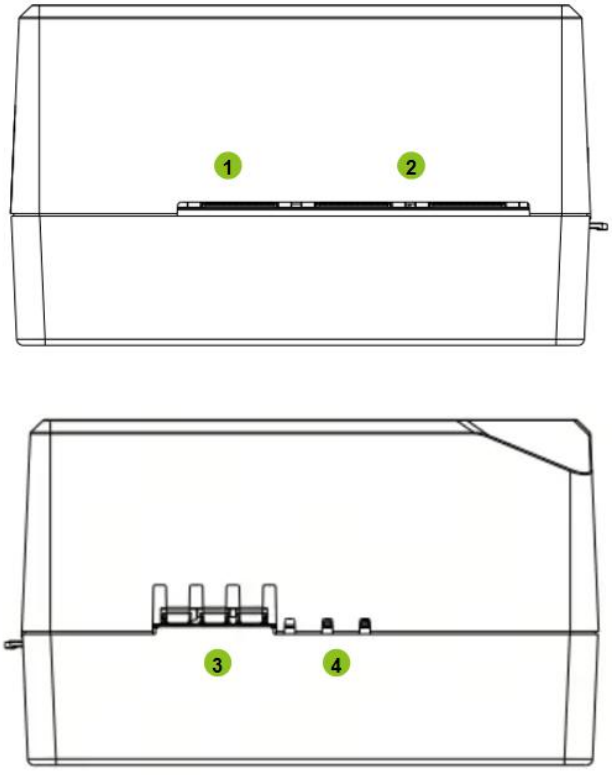
C4. Interface Usage Guidelines

4.1 System Overview

4.1.1 Interface Overview



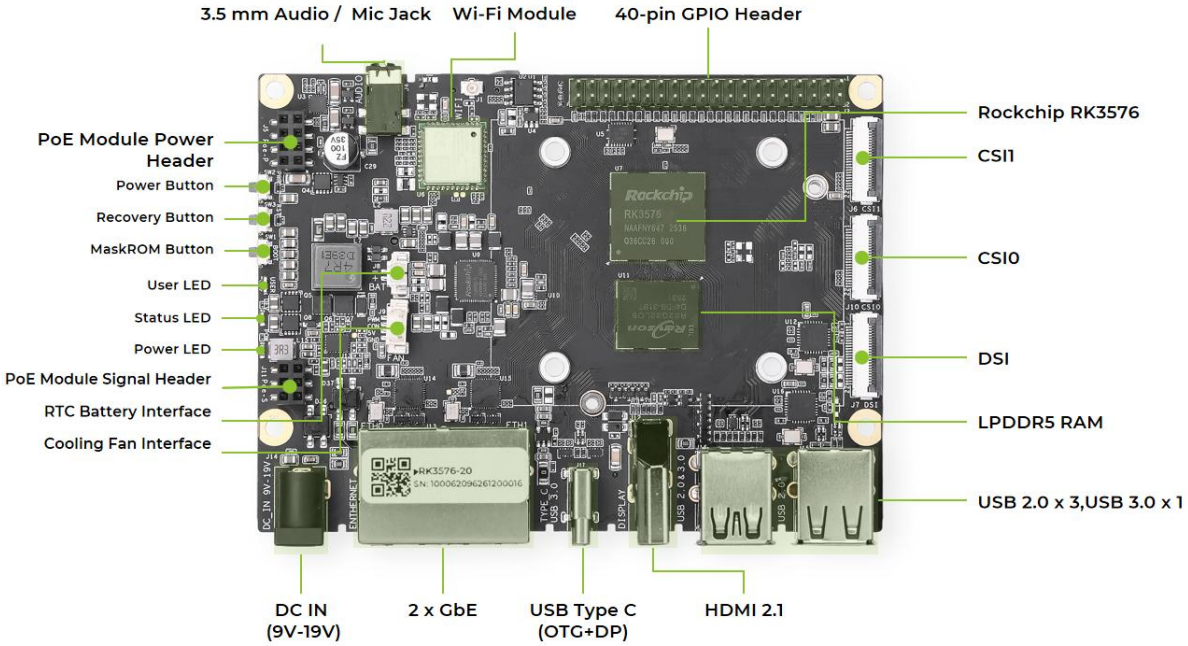
1. DC 9V~19V
2. 1x Gigabit Ethernet
   1x Gigabit Ethernet with PoE support
   (Additional PoE Module Required)
3. USB-C 2.0 (For flashing OS & Debug)
4. Display
5. 1x USB 3.0
   3x USB 2.0
6. 1 x Reserved Antenna Ports for Wireless
7. SD Card Slot
8. 1x 3.5mm Audio



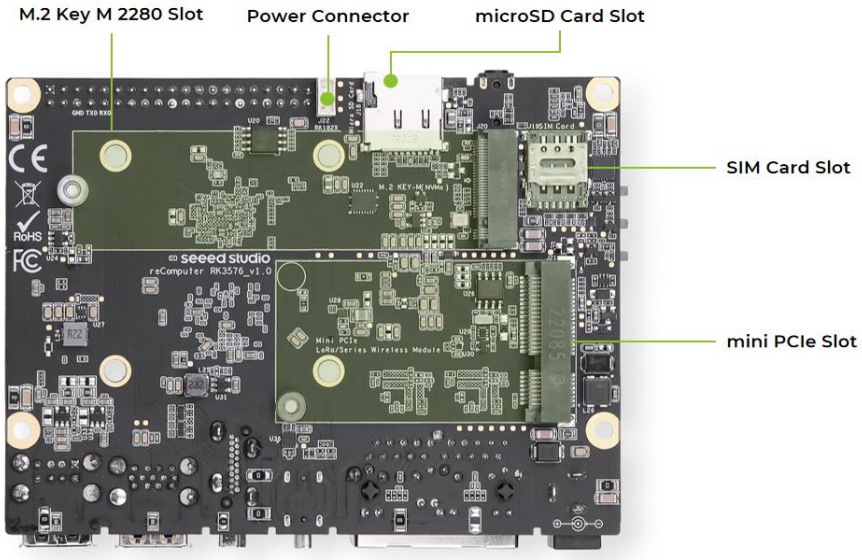
1. 1x 4-lane MIPI DSI(22pin)
2. 2x 4-lane MIPI CSI(22pin)
3. 1x Power Button
   1x Recovery Button
   1x MaskROM Button
4. 1x Power LED
   1x Status LED
   1x User LED

4.1.2 Mainboard Overview

Top View



Bottom View



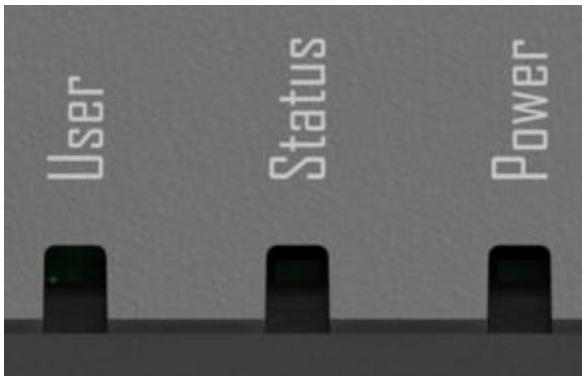
4.2 Interface Description

4.2.1 FAN

The reComputer RK3576 exposes fan control through the Linux hwmon subsystem. Fan speed can be read in real time, and the PWM duty cycle can be set manually or left to the automatic thermal governor.

4.2.2 LED Indicator Status

This device features three on-board LED indicators (Power, Status, and User) on the front panel. These LEDs allow users to monitor the hardware power-on sequence, observe system runtime behavior, and utilize a fully programmable interface for custom software interactions.



1. **LED States & Default Hardware Behavior**

Under the default system firmware, the LEDs exhibit the following standard behaviors during boot-up and post-boot operations:

LED Name	Default Color	Boot-up Phase Behavior	Post-Boot Behavior (System Ready)	Function & Status Description
User	Blue (Default)	Off	Heartbeat Blinking	User Custom Indicator: Utilizes an RGB tri-color LED at the hardware level. Upon successful system boot, the Blue LED is automatically claimed by the kernel to show a heartbeat pattern, signaling that the system is ready. The red and green channels remain

				off by default.
Status	Green	Off / Loading	Solid On	System Status Indicator: Represents the status of the Linux kernel and critical system services. It stays solid green once the system successfully boots and functions normally.
Power	Red	Solid On	Solid On	Power Indicator: Indicates that the carrier board is receiving valid power. It remains solid red as long as the device is connected to a stable power source (DC or PoE).

2. **Developer Essential: Control Boundaries**

Before starting secondary development, it is critical to understand the underlying hardware routing of each LED to avoid unnecessary troubleshooting:

- ✕ **Power LED (Red): Pure Hardware Control.** This LED is hardwired directly to the main power rail. There is no GPIO or kernel interface connected to it. It **cannot be controlled via software or commands**.
- ✕ **Status LED (Green): System-level Hardware Control.** This LED is physically managed by the underlying hardware power management logic or PMIC firmware to reflect core system stability. It does not expose any interface to the Linux user space and **cannot be controlled via software or commands**.
- **User LED (RGB Tri-color): Fully Developer Controllable!** This is an RGB LED capsule wired to the on-board GPIO expander (`gpiochip6`). It is the **only indicator on the device open for custom programming and software automation**. By default, the system claims the blue channel (USER_LED_B) as a heartbeat indicator, leaving the red and green channels idle.

3. **User RGB LED Control Guide**

In the Linux environment (e.g., Armbian), it is recommended to use the modern `gpiod` toolchain (`gpioinfo` / `gpioset`) to manipulate the idle Red and Green channels, and the standard Linux `sysfs` interface to control the Blue channel.

Locate the LED

In the system, LEDs are managed via the sysfs interface. Run the following command to verify the LED node:

```
Bash
ls /sys/class/leds/user-led
```

Expected Output :

Manual Control (ON/OFF)

Before controlling the LED manually, you need to set its trigger mode to none.

```
Bash
echo none | sudo tee /sys/class/leds/user-led/trigger
```

Turn ON/OFF the LED

```
Bash
# Turn ON
echo 1 | sudo tee /sys/class/leds/user-led/brightness
# Turn OFF
echo 0 | sudo tee /sys/class/leds/user-led/brightness
```

Automated Trigger Modes

The kernel provides various triggers to automate LED behavior based on system events.

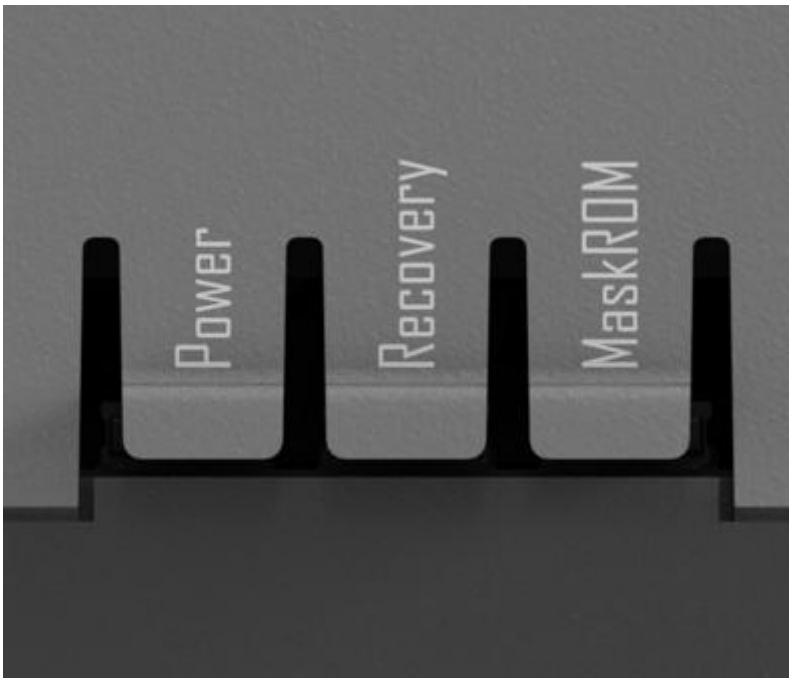
```
Bash
# Heartbeat Mode
echo heartbeat | sudo tee /sys/class/leds/user-led/trigger
# Custom Timer Mode (Blinking)
# Enable timer
echo timer | sudo tee /sys/class/leds/user-led/trigger
# Set intervals
echo 500 | sudo tee /sys/class/leds/user-led/delay_on
echo 500 | sudo tee /sys/class/leds/user-led/delay_off
```

Persistent Configuration on Boot

Since changes in /sys are volatile and reset after reboot, you can make them permanent by adding a cron job or using rc.local.

4.2.3 Button

The reComputer RK3576 series motherboards are equipped with 3 onboard physical buttons, primarily used for power control and low-level firmware flashing.

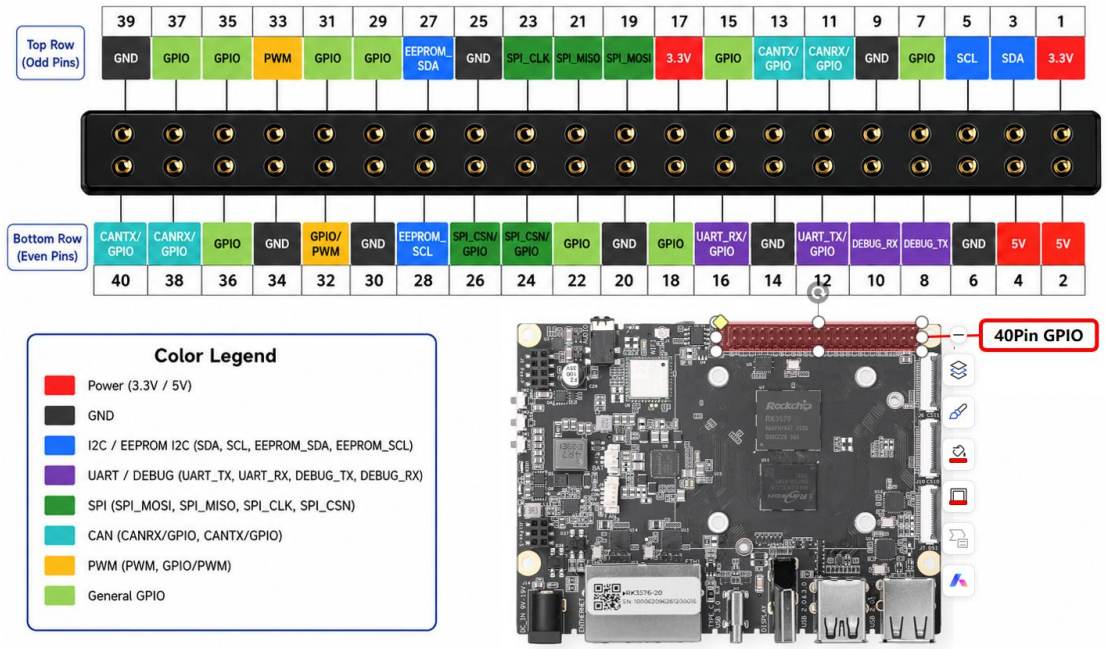


Button Name	Core Function	Operation & Trigger Mechanism	Developer Use Cases
Power	Power and system state control	**Short Press**: Power on / Trigger standard soft shutdown in Armbian OS (ACPI event) **Long Press (5-8s)**: Hard power off (force power cut)	Routine power on/off, forced reboot during system freezes, or capturing input events in the OS for custom actions.
Recovery	Enter Loader (Flashing) Mode	**Press and hold while powered off** -> Connect power -> Hold for 2-3 seconds, then release	Routine firmware upgrades or partition flashing (requires a functioning Bootloader).
MaskROM	Enter MaskROM (Unbricking) Mode	**Press and hold while powered off** -> Connect power -> Hold for 3 seconds, then release	Unbricking devices with corrupted Bootloaders, unbootable systems, or empty storage by forcing a low-level boot and full image flashing.

> **Developer Tip**: When using Recovery or MaskROM for flashing, ensure the device's **Type-C OTG port** is connected to the host PC via a data cable. You will need to use official Rockchip tools (such as RKDevTool for Windows or upgrade_tool for Linux) to execute the operations.

4.2.4 GPIO

The motherboard provides a standard 40-pin expansion header at the top, offering developers a comprehensive set of low-level hardware interfaces. These include power, debugging, and common industrial communication buses (I2C, UART, SPI, CAN, etc.), making it easy to connect external sensors, driver boards, or perform system-level debugging.



40-Pin Pinout Table

Function (Top Row / Odd Pins)	Pin	Pin	Function (Bottom Row / Even Pins)
3.3V (Power Output)	1	2	5V (Power Output)
SDA (General I2C Data)	3	4	5V (Power Output)
SCL (General I2C Clock)	5	6	GND (Ground)
GPIO (General Purpose IO)	7	8	DEBUG_TX (System Debug Serial TX)
GND (Ground)	9	10	DEBUG_RX (System Debug Serial RX)
CANRX / GPIO (CAN 1 Receive)	11	12	UART_TX / GPIO (General UART TX)
CANTX / GPIO (CAN 1 Transmit)	13	14	GND (Ground)

GPIO (General Purpose IO)	15	16	UART_RX / GPIO (General UART RX)
3.3V (Power Output)	17	18	GPIO (General Purpose IO)
SPI_MOSI (SPI Data Output)	19	20	GND (Ground)
SPI_MISO (SPI Data Input)	21	22	GPIO (General Purpose IO)
SPI_CLK (SPI Clock)	23	24	SPI_CSN / GPIO (SPI Chip Select 1)
GND (Ground)	25	26	SPI_CSN / GPIO (SPI Chip Select 2)
EEPROM_SDA (Dedicated I2C Data)	27	28	EEPROM_SCL (Dedicated I2C Clock)
GPIO (General Purpose IO)	29	30	GND (Ground)
GPIO (General Purpose IO)	31	32	GPIO / PWM (Pulse Width Modulation)
PWM (Pulse Width Modulation)	33	34	GND (Ground)
GPIO (General Purpose IO)	35	36	GPIO (General Purpose IO)
GPIO (General Purpose IO)	37	38	CANRX / GPIO (CAN 2 Receive)
GND (Ground)	39	40	CANTX / GPIO (CAN 2 Transmit)
Function (Top Row / Odd Pins)	Pin	Pin	Function (Bottom Row / Even Pins)
3.3V (Power Output)	1	2	5V (Power Output)
SDA (General I2C Data)	3	4	5V (Power Output)
SCL (General I2C Clock)	5	6	GND (Ground)
GPIO (General Purpose IO)	7	8	DEBUG_TX (System Debug Serial TX)
GND (Ground)	9	10	DEBUG_RX (System Debug Serial RX)
CANRX / GPIO (CAN 1 Receive)	11	12	UART_TX / GPIO (General UART TX)
CANTX / GPIO (CAN 1 Transmit)	13	14	GND (Ground)
GPIO (General Purpose IO)	15	16	UART_RX / GPIO (General UART RX)
3.3V (Power Output)	17	18	GPIO (General Purpose IO)
SPI_MOSI (SPI Data Output)	19	20	GND (Ground)
SPI_MISO (SPI Data Input)	21	22	GPIO (General Purpose IO)
SPI_CLK (SPI Clock)	23	24	SPI_CSN / GPIO (SPI Chip Select 1)
GND (Ground)	25	26	SPI_CSN / GPIO (SPI Chip Select 2)

EEPROM_SDA (Dedicated I2C Data)	27	28	EEPROM_SCL (Dedicated I2C Clock)
GPIO (General Purpose IO)	29	30	GND (Ground)
GPIO (General Purpose IO)	31	32	GPIO / PWM (Pulse Width Modulation)
PWM (Pulse Width Modulation)	33	34	GND (Ground)
GPIO (General Purpose IO)	35	36	GPIO (General Purpose IO)
GPIO (General Purpose IO)	37	38	CANRX / GPIO (CAN 2 Receive)
GND (Ground)	39	40	CANTX / GPIO (CAN 2 Transmit)

GPIO Usage

By default, most pins on the 40-Pin header function as **standard GPIOs**, unless pre-allocated by system drivers.The standard **gpiod** toolset (`gpiodetect`, `gpioinfo`, `gpioset`, `gpioget`) is recommended for shell-level manipulation.

To manipulate **Physical Pin 21** (which is specified as a General GPIO and currently free):

1. **Locate its mapped controller and line number:**

```
Plain Text
sudo gpioinfo | grep -i PIN_21
```

Output: `line 14: "PIN_21" unused input active-high`This indicates it belongs to `gpiochip1`, Line `14`.

2. **Set the pin to High (e.g., Turn on an LED):**

```
Plain Text
gpioset gpiochip1 14=1
```

3. **Set the pin to low:**

```
Plain Text
gpioset gpiochip1 14=0
```

4. **Read the input status of Physical Pin 27:**Querying `sudo gpioinfo | grep -i PIN_27` shows it maps to `gpiochip4`, Line `23`:

```
Plain Text
gpioget gpiochip4 23
```


To unlock the specialized peripheral hardware (I2C, UART, CAN, PWM) labeled on the pinout, **you must explicitly enable its corresponding Device Tree Overlay (DTBO)**. Otherwise, they will continue to function merely as generic GPIOs.

All 40-Pin specific overlay configurations reside in `/boot/dtb/rockchip/overlay/` and follow the strict naming prefix **`recomputer-rk3576-devkit-40pin-`**:

Target Function	Physical Pins	Required Overlay Filename
I2C3	Pin 3 (SDA), Pin 5 (SCL)	recomputer-rk3576-devkit-40pin-i2c3.dtbo
UART3	Pin 8 (TX), Pin 10 (RX)	recomputer-rk3576-devkit-40pin-uart3.dtbo
UART5	Pin 12 (TX), Pin 16 (RX)	recomputer-rk3576-devkit-40pin-uart5.dtbo
PWM2	Pin 32 (PWM), Pin 33 (PWM)	recomputer-rk3576-devkit-40pin-pwm2-ch6.dtbo
CAN0	Pin 11 (RX), Pin 13 (TX)	recomputer-rk3576-devkit-40pin-can0.dtbo
CAN1	Pin 38 (RX), Pin 40 (TX)	recomputer-rk3576-devkit-40pin-can1.dtbo

Configuration Procedure:

1. Open the system environment configuration file:

```
Plain Text
sudo nano /boot/armbianEnv.txt
```

1. Locate the `overlays=` line (or append it to the end of the file if missing), then insert the desired overlay name **without the `.dtbo` extension**. Separate multiple overlays with a space:

```
Plain Text
overlays=recomputer-rk3576-devkit-40pin-i2c3 recomputer-rk3576-devkit-40pin-can0
```

1. Save and exit (`Ctrl+O`, Enter, `Ctrl+X`), then restart the hardware to apply the device tree modification:

```
Plain Text
sudo reboot
```

Note

1. **Active by Default:** Please note that the **SPI Bus pins (Pins 19, 21, 23, 24, 26) are enabled by default** at boot time. You do NOT need to assign any custom DTBO for

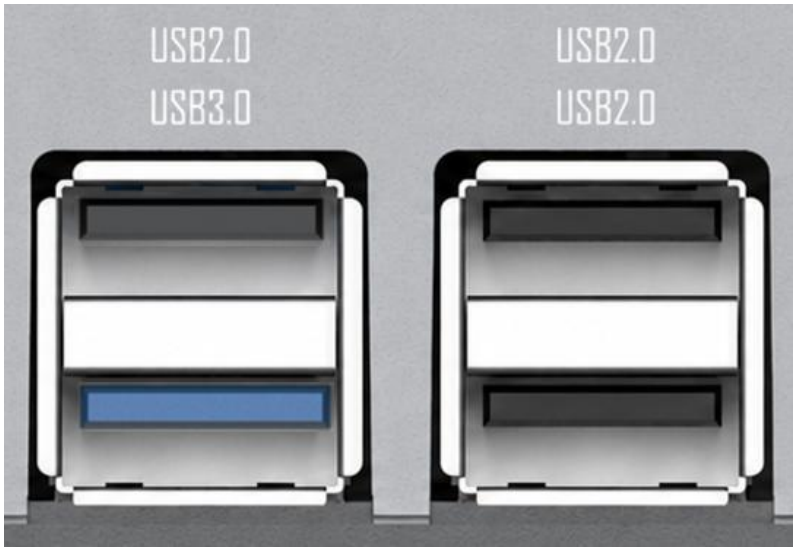
SPI in armbianEnv.txt.

2. **Hardware Architecture Sharing:** The physical SPI signals on the 40-Pin header share the exact same hardware bus domain with the onboard **4G LTE / LoRaWAN / Hailo Wi-Fi expansion slots**. To facilitate an out-of-the-box experience for these communication modules, the base operating system keeps this SPI channel permanently active.

3. **Developer Recommendation:** If you connect custom third-party SPI breakout boards onto the 40-Pin expansion header, ensure you handle Chip Select (CS) and addressing carefully to circumvent communication bus deadlocks with the built-in modules.

4.2.5 USB

This device provides multiple physical USB interfaces mapped to the reComputer RK3576 series. These interfaces handle external peripheral attachment, system firmware flashing (OTG), and secondary display output.



The USB resource allocation is defined as follows:

Interface Location / Label	Physical Form Factor	Bus Protocol & Maximum Bandwidth	Hardware Specification & Controller Routing
USB 3.0 Host	USB Type-A	**USB 3.2 Gen 1x1** (5 Gbps)	Routed via the internal USB 3.0 Controller. Provides high-bandwidth data channels optimized for USB 3.0 Industrial Cameras,

			NVMe storage expansion, or high-speed DAQ modules.
USB 2.0 Host	USB Type-A (Multiple)	**USB 2.0** (480 Mbps)	Managed by the USB 2.0 Host Controller. Used for standard peripherals such as keyboards, mice, 4G/5G cellular modems, hardware encryption dongles, or USB-to-UART bridge boards.
USB Type-C	USB Type-C	**USB 3.0 / 2.0 OTG + DP 1.4 Alt Mode**	A multi-protocol composite port routed to the RK3576 USB 3.0 OTG controller and the DisplayPort (DP) TX PHY. Supports bidirectional host/device roles and hardware video streaming.

Using `lsusb` allows you to view information about the USB devices connected to the system:

```
seeed@recomputer-rk3576-devkit:~$ lsusb
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 004: ID 046d:c077 Logitech, Inc. Mouse
Bus 001 Device 005: ID 046d:c34b Logitech, Inc. USB Keyboard
Bus 001 Device 003: ID 05e3:0610 Genesys Logic, Inc. Hub
Bus 001 Device 002: ID 05e3:0610 Genesys Logic, Inc. Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

Note

USB Type-C Interface Architecture

A. System Flashing & Maintenance (USB OTG Device Mode)

Triggered via Recovery / MaskROM keys, the port boots into device mode for low-level firmware flashing using `RKDevTool`.

B. DisplayPort Output (DP 1.4 Alt Mode)

Multiplexes hardware video lanes to drive a monitor directly via Type-C cables or adapters, delivering standard DP 1.4 displays.

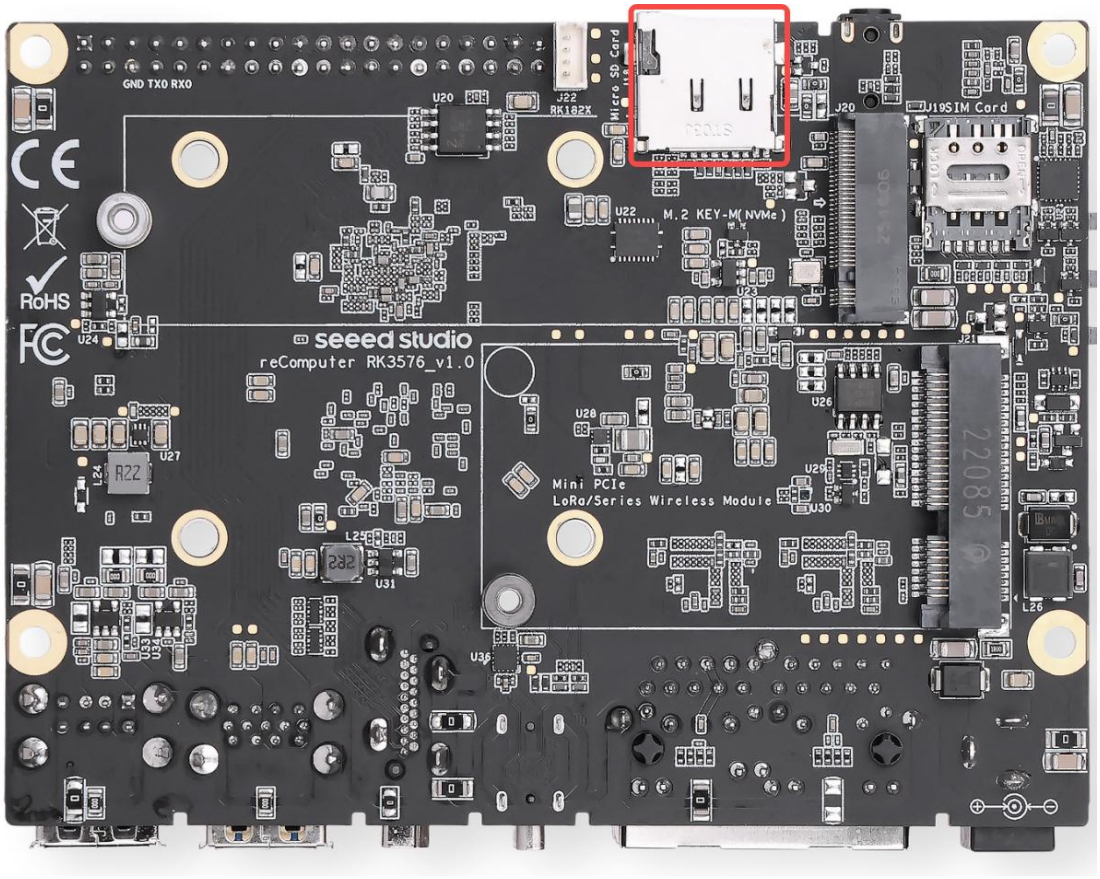
C. Standard Native Host Mode

Defaults to a standard USB 3.0 Host during normal OS runtime to interface directly with Type-C storage, network adapters, or hubs.

4.2.6 SD Card Slot

The motherboard is equipped with one standard **Micro SD (TF) card slot**, routed directly to the **SDMMC0** bus of the RK3576 SOC. This interface is primarily utilized for

system booting, running the operating system (such as Armbian OS), and local data storage.



Out-of-the-Box Experience

The device comes standard with an included **32GB Class 10 Micro SD card**. This card has passed rigorous compatibility testing by Seeed Studio and is perfectly capable of handling daily Armbian OS operations, basic network configurations, and lightweight edge AI algorithm validations, allowing developers to get started immediately.

Note

1. **Boot Priority Mechanism (MaskROM Stage)** The firmware boot sequence on the RK3576 is strictly controlled by the low-level MaskROM. If bootable firmware coexists on both the Micro SD card and the onboard eMMC, the **MaskROM will prioritize loading the SPL (Secondary Program Loader) from the Micro SD card**. Please pay close attention to this physical characteristic during firmware upgrades or multi-OS switching.

2. **NVMe Boot Limitation (U-Boot Stage)** It is critical to note that **M.2 NVMe SSDs are NOT within the direct MaskROM boot path**. The system cannot perform the initial stage-1 boot directly from a raw NVMe drive. To run the OS from an NVMe SSD, the system must first execute the bootloader on either the Micro SD card or the eMMC to enter the **U-Boot stage**, where U-Boot will then initialize, load, and hand

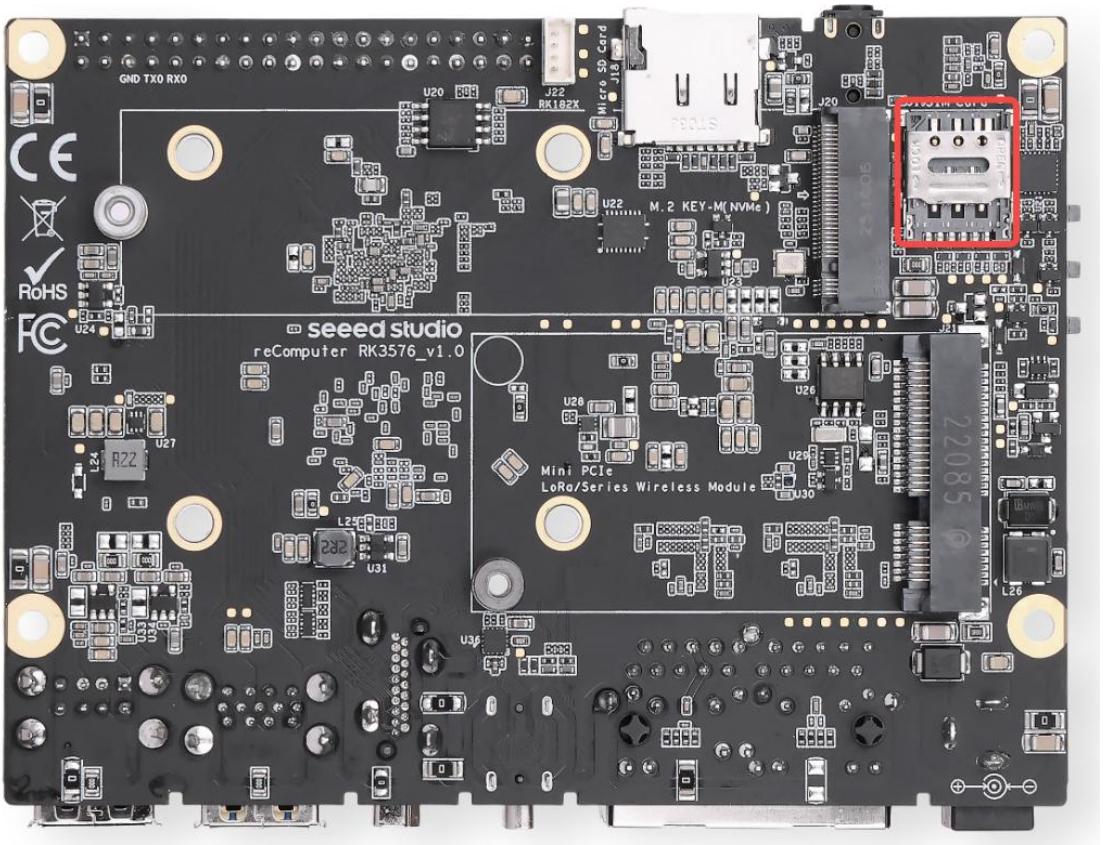
over control to the main OS residing on the NVMe storage.

3. **Heavy Workload Upgrade Recommendations** If you plan to deploy applications with high-frequency I/O read/write demands at the edge (such as running large databases with continuous local logging, or managing complex Docker container clusters), the massive volume of 4K random read/writes may hit the performance bottleneck of a standard Class 10 card. For such heavy-duty industrial scenarios, we recommend migrating the core OS rootfs to an **M.2 NVMe SSD** (via U-Boot chain-loading), or upgrading to a high-performance Micro SD card explicitly rated for **A1 or A2 (Application Performance Class)** to ensure optimal system responsiveness.

4. **Mount Point** In Armbian OS, the Micro SD card is typically enumerated as `/dev/mmcblk0`.

4.2.7 SIM Card Slot

The motherboard features one onboard card slot, explicitly designed to work with 4G LTE cellular modules installed in the Mini-PCIe slot. By inserting a standard carrier SIM card, the device can achieve edge-to-cloud cellular connectivity in environments without wired Ethernet.



Hardware Architecture & Mechanism

The signal lines of the Nano SIM slot are routed directly to the specific pins (such as UIM_PWR, UIM_DATA, UIM_CLK, UIM_RESET) of the **Mini-PCIe slot**.

- **Non-SOC Direct Connection**: Note that the SIM card signals do not interface directly with the RK3576 SOC. Instead, they are **fully managed and driven by the 4G module** installed in the Mini-PCIe slot. Consequently, the SIM card will only function when a compatible 4G module is active.

Note

Developer Notes & Operating Standards

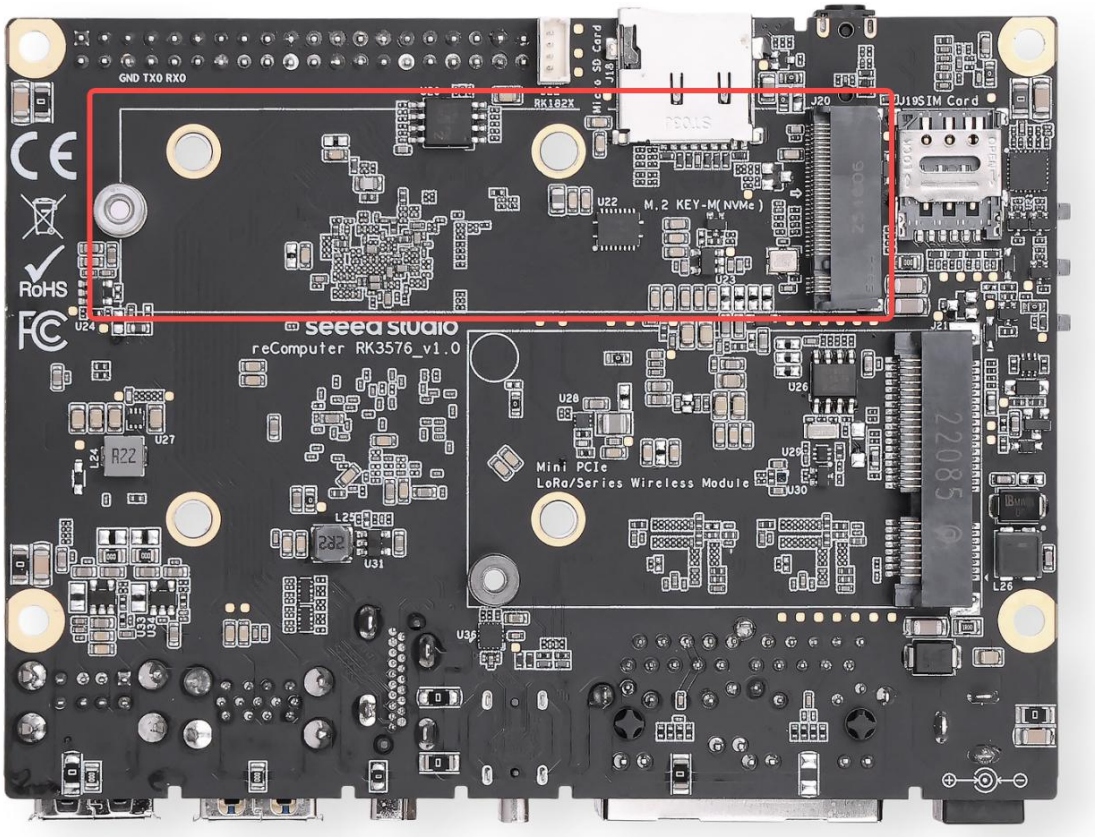
1. **Strictly No Hot-Plugging** Always insert or remove the SIM card **while the device is completely powered off** (DC power or PoE disconnected). Hot-plugging the SIM card while the system is live can permanently damage the SIM interface on the 4G module or cause network-related system crashes.

2. **Form Factor Specifications** This slot **strictly supports Nano SIM cards (4FF)**, which is the smallest standard size used in modern smartphones. Avoid using Micro or Standard SIM cards with cheap plastic adapters, as they can easily get stuck or bend the internal pins of the slot.

3. **Insertion Orientation** Please check the silk-screen icon or structural notch near the slot before insertion. Typically, the SIM card should be inserted with the **gold contacts facing down** and the notched corner going in first (please follow the physical alignment marking on the board).

4.2.8 M.2 Key M 2280 Slot

The motherboard is equipped with one standard M.2 Key M 2280 slot, driven by a **PCIe 2.1 x1** bus. It is designed for high-speed storage expansion or edge AI computing acceleration.



Supported Devices

- **NVMe SSD**: Supports standard 2280 NVMe Solid State Drives for system booting, high-capacity data storage, or housing large AI models.
- **AI Accelerator**: Compatible with M.2 Key M based AI accelerator modules to enhance edge inference capabilities.

Tested & Verified Devices

To ensure optimal compatibility and system stability, we highly recommend using the M.2 extension modules that have been fully tested and verified by **Seeed Studio**:

- **Storage**
  - Official Seeed Studio 2280 NVMe SSDs:https://www.seeedstudio.com/M-2-2280-SSD-128GB-p-5332.html
- **AI Accelerators**
  - **Rockchip Series**: RK1820 / RK1828
  - **Hailo Series**: Hailo-8
  - **DeepX Series**: DX-M1

Notes

1. **Protocol Compatibility**: This slot **strictly supports PCIe NVMe** devices. M.2 SATA SSDs are **NOT supported** and will not be recognized

by the OS.

2. **Bandwidth limitation**: The slot operates on a PCIe 2.1 x1 lane, providing a theoretical maximum bandwidth of approximately 500 MB/s. When purchasing an NVMe SSD, standard cost-effective Gen3/Gen4 drives are sufficient; ultra-high-speed Gen4 SSDs will be bottlenecked by the PCIe 2.1 x1 interface.

3. **Form Factor**: The mounting standoff is designed specifically for **2280** (22mm x 80mm) modules.

SSD Use Guide

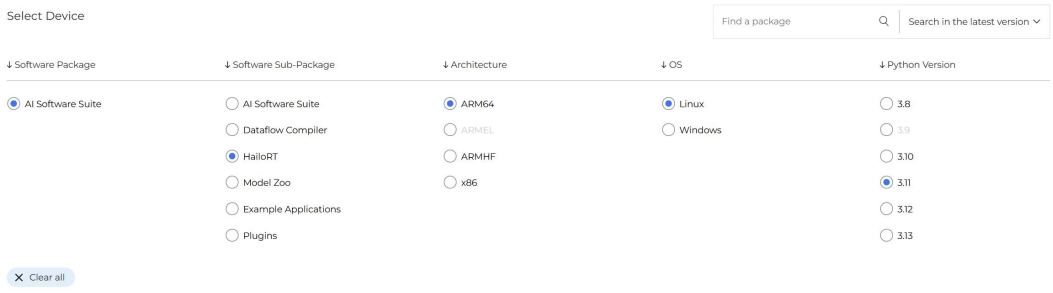
```bash
lsblk
```

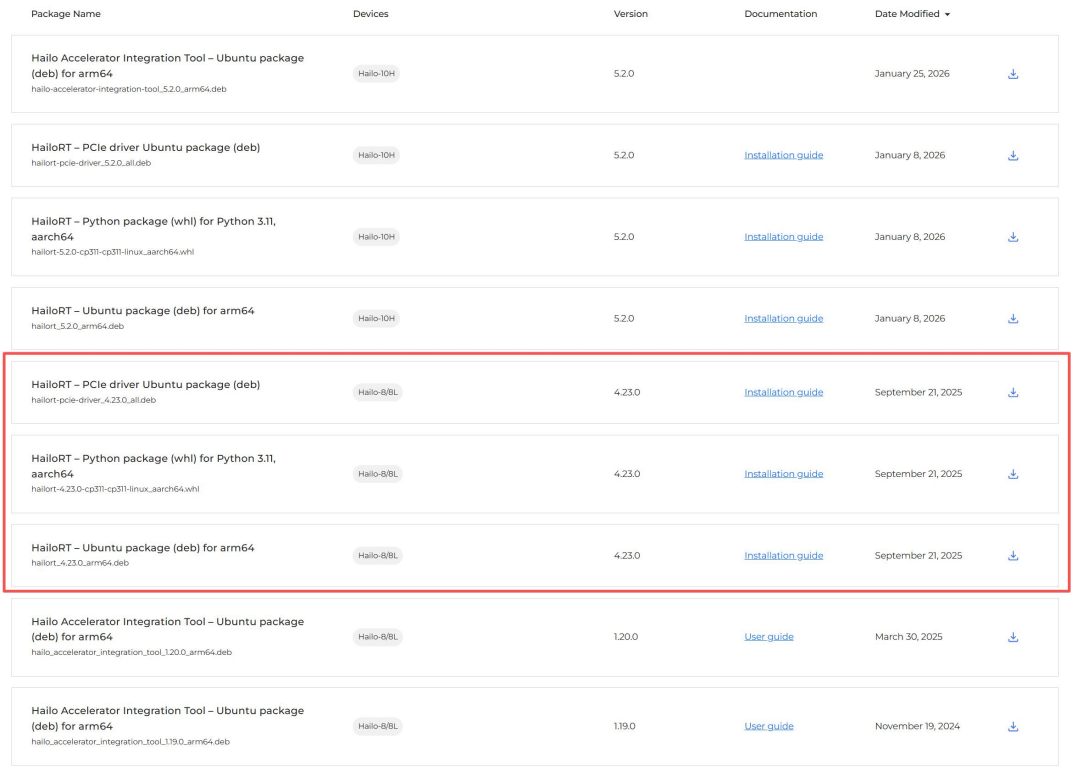
```
seeed@recomputer-rk3576-devkit:~$ lsblk
NAME        MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
mtdblock0    31:0    0    16M  0 disk
mmcblk1     179:0    0  29.7G  0 disk
└─mmcblk1p1 179:1    0  29.4G  0 part /var/log.hdd
                                    /
zram0       252:0    0   3.9G  0 disk [SWAP]
zram1       252:1    0    50M  0 disk /var/log
zram2       252:2    0     0B  0 disk
nvme0n1     259:0    0 119.2G  0 disk
```

Hailo YOLOv11 Deployment & Inference Guide

Package Installation

After installing the base PCIe driver, a system reboot is required for changes to take effect.





Package Name	Devices	Version	Documentation	Date Modified
Hailo Accelerator Integration Tool – Ubuntu package (deb) for arm64 hailo-accelerator-integration-tool_5.2.0_arm64.deb	Hailo-10H	5.2.0		January 25, 2026
HailoRT – PCIe driver Ubuntu package (deb) hailort-pcie-driver_5.2.0_all.deb	Hailo-10H	5.2.0	Installation guide	January 8, 2026
HailoRT – Python package (whl) for Python 3.11, aarch64 hailort-5.2.0-cp311-cp311-linux_aarch64.whl	Hailo-10H	5.2.0	Installation guide	January 8, 2026
HailoRT – Ubuntu package (deb) for arm64 hailort_5.2.0_arm64.deb	Hailo-10H	5.2.0	Installation guide	January 8, 2026
HailoRT – PCIe driver Ubuntu package (deb) hailort-pcie-driver_4.23.0_all.deb	Hailo-8/8L	4.23.0	Installation guide	September 21, 2025
HailoRT – Python package (whl) for Python 3.11, aarch64 hailort-4.23.0-cp311-cp311-linux_aarch64.whl	Hailo-8/8L	4.23.0	Installation guide	September 21, 2025
HailoRT – Ubuntu package (deb) for arm64 hailort_4.23.0_arm64.deb	Hailo-8/8L	4.23.0	Installation guide	September 21, 2025
Hailo Accelerator Integration Tool – Ubuntu package (deb) for arm64 hailo_accelerator_integration_tool_1.20.0_arm64.deb	Hailo-8/8L	1.20.0	User guide	March 30, 2025
Hailo Accelerator Integration Tool – Ubuntu package (deb) for arm64 hailo_accelerator_integration_tool_1.19.0_arm64.deb	Hailo-8/8L	1.19.0	User guide	November 19, 2024

Plain Text

```
# Install the PCIe driver
sudo dpkg -i hailort-pcie-driver_4.23.0_all.deb

# Reboot the system
sudo reboot

# After reboot, verify the driver is loaded
lsmod | grep hailo

# Install HailoRT
sudo dpkg -i hailort_4.23.0_arm64.deb

# Scan and verify device status
hailortcli scan

# Create and activate a virtual environment
python3 -m venv hailo_env
source hailo_env/bin/activate

# Install HailoRT Python library
pip install hailort-4.23.0-cp311-cp311-linux_aarch64.whl
```

```
# Verify installation and device connection
python3 -c "from hailo_platform import VDevice; vdev = VDevice();
print('Successfully connected via VDevice! Device info:', vdev)"
```

Install Hailo Model Zoo

To run official pre-trained models, you need to install the Hailo Model Zoo and its system dependencies.

```
Plain Text
# 1. Install required system libraries
sudo apt update
sudo apt install -y git libglib2.0-0 libgl1-mesa-glx

# 2. Clone the official repository (latest branch recommended)
git clone https://github.com/hailo-ai/hailo_model_zoo.git
cd hailo_model_zoo
pip install -e .
```

Running the YOLOv11 Model

- **Check camera devices:** Identify your webcam mount point.

```
Plain Text
v4l2-ctl --list-devices
```

- **Download model:** Ensure the `yolov11n.hef` model file is downloaded and placed in your working directory.

Create a file named `webcam_yolo11.py` and paste the code below. Adjust the `HEF_PATH` and `DEVICE_ID` under the configuration section to match your setup.

Python

```
Plain Text
import numpy as np
import cv2
```



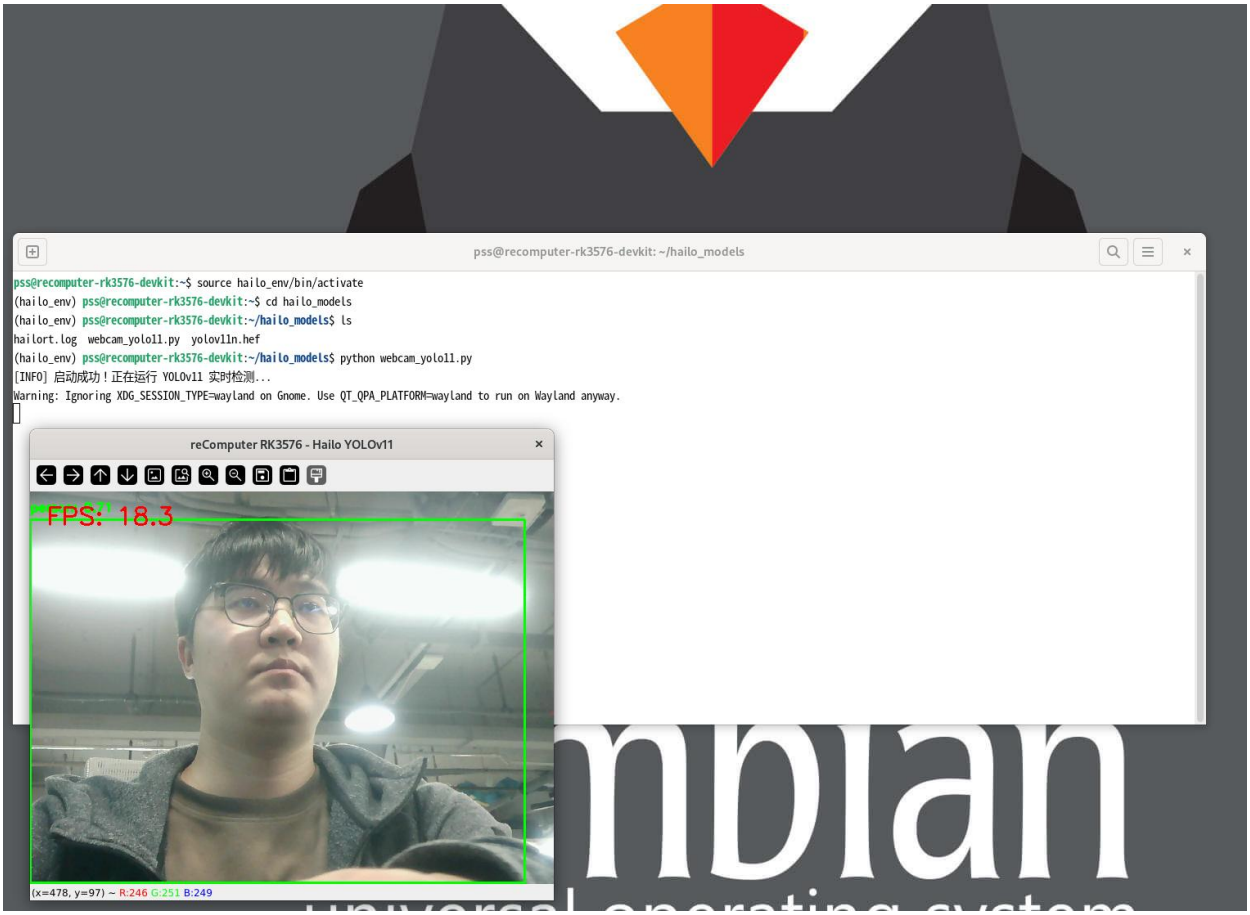
```
import time
from hailo_platform import (VDevice, HEF, InferVStreams,
ConfigureParams,
                            HailoStreamInterface, InputVStreamParams,
OutputVStreamParams)

# ================= Configuration =================
HEF_PATH = 'yolov11n.hef'
DEVICE_ID = "/dev/video40"  # Update based on v4l2-ctl output
CONF_THRESHOLD = 0.45# COCO Dataset 80 Class Labels
COCO_CLASSES = [
    "person", "bicycle", "car", "motorcycle", "airplane", "bus", "train", "truck",
"boat", "traffic light",
    "fire hydrant", "stop sign", "parking meter", "bench", "bird", "cat",
"dog", "horse", "sheep", "cow",
    "elephant", "bear", "zebra", "giraffe", "backpack", "umbrella",
"handbag", "tie", "suitcase", "frisbee",
    "skis", "snowboard", "sports ball", "kite", "baseball bat", "baseball
glove", "skateboard", "surfboard",
    "tennis racket", "bottle", "wine glass", "cup", "fork", "knife", "spoon",
"bowl", "banana", "apple",
    "sandwich", "orange", "broccoli", "carrot", "hot dog", "pizza", "donut",
"cake", "chair", "couch",
    "potted plant", "bed", "dining table", "toilet", "tv", "laptop", "mouse",
"remote", "keyboard", "cell phone",
    "microwave", "oven", "toaster", "sink", "refrigerator", "book", "clock",
"vase", "scissors", "teddy bear",
    "hair drier", "toothbrush"
]
# ================================================def main():#
1. Initialize Hailo Hardware
    hef = HEF(HEF_PATH)
    input_vstream_info = hef.get_input_vstream_infos()[0]
    input_h, input_w = input_vstream_info.shape[:2]

    cap = cv2.VideoCapture(DEVICE_ID)
    if not cap.isOpened():
        print("Cannot open webcam")
        return# Setup inference variables
    prev_time = 0with VDevice() as target:
        config_params_dict = ConfigureParams.create_from_hef(hef,
HailoStreamInterface.PCIe)
        network_group = target.configure(hef, config_params_dict)[0]
```



```
        with network_group.activate():
            vstream_params =
(InputVStreamParams.make_from_network_group(network_group),

OutputVStreamParams.make_from_network_group(network_group))
            with InferVStreams(network_group, vstream_params[0],
vstream_params[1]) as vstreams:
                print("[INFO] Initialization successful! Running YOLOv11 real-
time detection...")
                while True:
                    start_time = time.time()  # Record start time for FPS
                    ret, frame = cap.read()
                    if not ret: break# Preprocessing (Convert to RGB based on
previous validation)
                    frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
                    resized = cv2.resize(frame_rgb, (input_w, input_h))
                    input_tensor = np.expand_dims(resized, axis=0)

                    # Inference
                    outputs = vstreams.infer(input_tensor)

                    # Parsing and Drawing
                    h, w, _ = frame.shape
                    for name, class_list in outputs.items():
                        # Iterate through 80 classesfor class_id, detections in
enumerate(class_list[0]):
                            if len(detections) > 0:
                                for det in detections:
                                    if len(det) >= 5:
                                        ymin, xmin, ymax, xmax, confidence = det[:5]
                                        if confidence > CONF_THRESHOLD:
                                            # Coordinate Mapping
                                            left, top = int(xmin * w), int(ymin * h)
                                            right, bottom = int(xmax * w), int(ymax * h)

                                            # Get class name, display ID if out of bounds
                                            class_name = COCO_CLASSES[class_id] if
class_id < len(COCO_CLASSES) else f"ID {class_id}"# Draw bounding box
                                            cv2.rectangle(frame, (left, top), (right, bottom),
(0, 255, 0), 2)
                                            # Draw background and label text
                                            label = f"{class_name}: {confidence:.2f}"
                                            cv2.putText(frame, label, (left, top - 10),
```



```
                                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255,
0), 2)

            # Calculate and display real-time FPS
            curr_time = time.time()
            fps = 1 / (curr_time - start_time)
            # Print in the top left corner
            cv2.putText(frame, f"FPS: {fps:.1f}", (20, 40),
                    cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)

            # Display window
            cv2.imshow('reComputer RK3576 - Hailo YOLOv11', frame)
            if cv2.waitKey(1) & 0xFF == ord('q'):
                break

    cap.release()
    cv2.destroyAllWindows()

if __name__ == "__main__":
    main()
```



RK182x Deployment & Inference Guide

RK182x supports the coprocessor mode, where the host SoC (e.g., RK3588/RK3576) acts as the system core, responsible for task scheduling, resource allocation, and overall control(). It connects to the RK1820/RK1828 acceleration unit via high-speed PCIe (typically trained to Gen2 x1 at 5 GT/s, providing ~400 MB/s unidirectional bandwidth) or USB 3.0 interfaces. Both RK1820 and RK1828 share the same PCI Device ID (1d87:182a). The development framework consists of the PC-side model conversion tool (RKNN3 Toolkit) and the board-side runtime environment (RKNN3 Runtime).

Option A: Rapid Automated Deployment (Recommended)

Using the precompiled automated installer package from the SDK allows for a one-click deployment of the kernel driver, device firmware, runtime libraries, debugging tools, and system auto-start services.

```
Markdown
# Push the package via ADB from PC to Board
# Optional packages include:
#   - rknn3_rk182x_m2_installer_arm64.tgz (M.2 Module) [cite: 194]
adb push rknn3_rk182x_m2_installer_arm64.tgz /tmp/installer.tgz
adb shell "cd /tmp && tar xzf installer.tgz && ./install.sh"

# Note: A hard power cycle (physical power off) is MANDATORY to ensure proper hardware and firmware loading.
sudo poweroff
```

If manual control over driver binding is required during development and debugging, or when operating in an environment without the automated package installed, execute the following system bus commands in order.

Note: **RK1820 requires explicit manual activation of BusMaster after driver probing.**

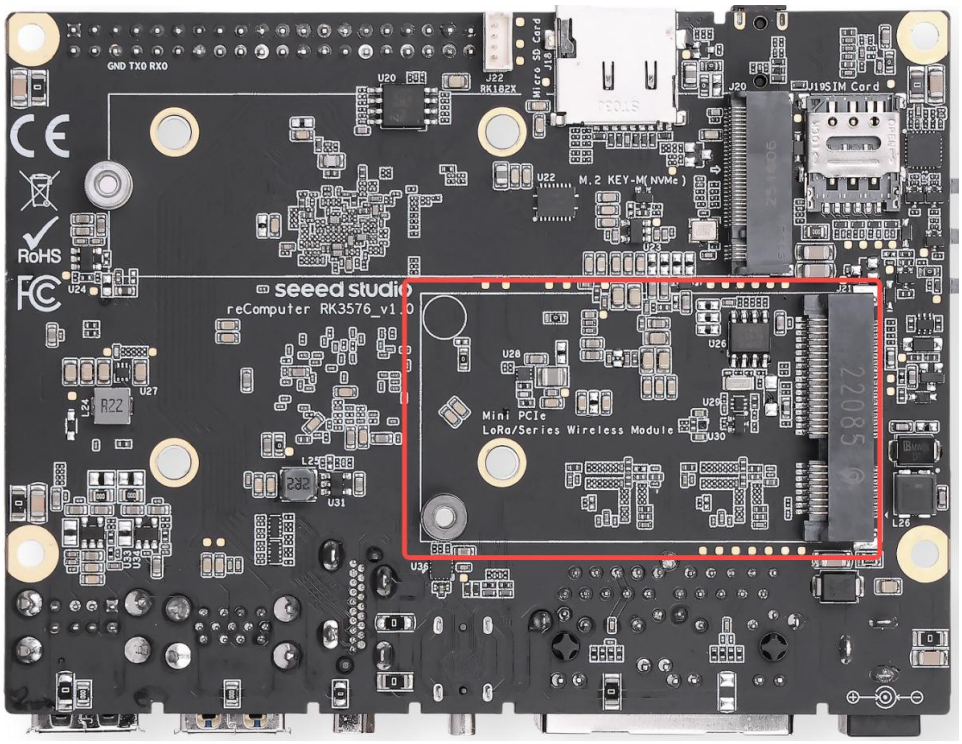
```
Markdown
# Force driver override and trigger binding
echo pcie-rkep | sudo tee /sys/bus/pci/devices/0000:01:00.0/driver_override
echo 0000:01:00.0 | sudo tee /sys/bus/pci/drivers/pcie-rkep/bind

# Enable the BusMaster register to activate accelerator card bus mastering
sudo setpci -s 01:00.0 COMMAND=0x0406
```


The host system runs Python 3.11 by default. Before deployment, ensure you have converted your models on an x86 machine using the RKNN3 Toolkit into the dedicated `.rknn` format (for both LLMs and CNNs) and uploaded the resulting files (e.g., `qwen2_5_1_5b_rk1820.rknn`) to the host.

```
Markdown
# 1. Create workspace
mkdir -p ~/rk_182x_work && cd ~/rk_182x_work

# 2. Clone Model Zoo and Toolkit repositories [cite: 458, 462]
git clone --recursive https://github.com/airockchip/rknn3-model-zoo.git
git clone https://github.com/airockchip/rknn3-toolkit.git

# 3. Install rknn3-toolkit-lite [cite: 460]
cd rknn3-toolkit/rknn3-toolkit-lite/packages
pip3 install ./rknn3_toolkit_lite-1.0.0-cp311-cp311-linux_aarch64.whl

# 4. Install dependencies
pip3 install -r requirements.txt
```

4.2.9 Mini-PCIe Slot

The motherboard features one standard Mini-PCIe (Mini PCI Express) slot, primarily designed for expanding industrial-grade wireless communication modules such as 4G LTE, LoRaWAN, or Wi-Fi HaLow.



Signal & Bus Architecture

This Mini-PCIe slot routes both **PCIe** and **USB** signals internally, ensuring excellent compatibility with the vast majority of mainstream wireless communication modules on the market.

- **Cellular Expansion**: In conjunction with the onboard **Nano SIM card slot**, it allows direct insertion of 4G LTE modules to enable cellular network connectivity.

Tested & Verified Devices

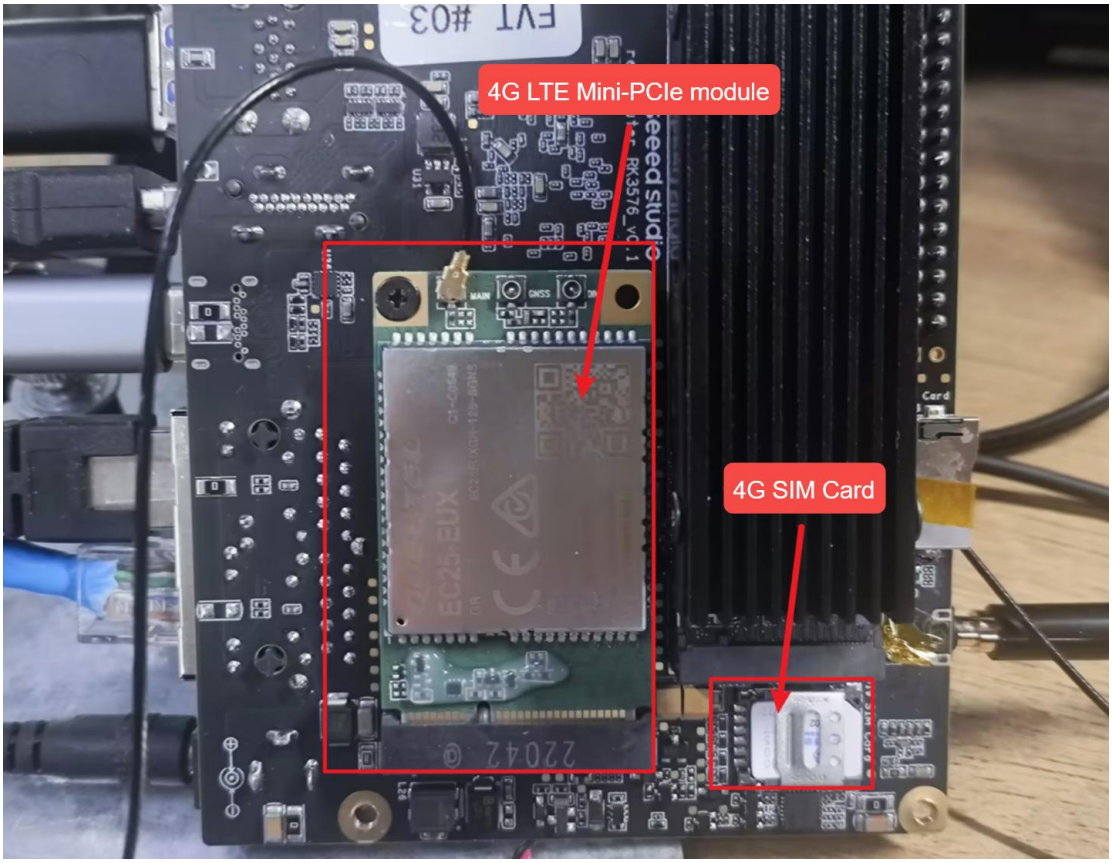
To ensure optimal compatibility and system stability, we highly recommend using the Mini-PCIe modules that have been fully tested and verified by **Seeed Studio**:

- **4G Cellular Network**
  - Official recommended 4G LTE Mini-PCIe module
- **LoRaWAN & IoT Wireless**
  - USB-based / SPI-based LoRaWAN gateway modules
  - Wi-Fi HaLow (802.11ah) long-range, low-power wireless modules

Notes

1. **SIM Card Integration**: When using a 4G LTE module, please insert the Nano SIM card into the onboard slot **while the device is powered off**. Hot-plugging the SIM card may cause module recognition failure or permanent damage.
2. **Antenna Routing**: Since industrial enclosures can shield wireless signals, always connect IPEX-to-SMA pigtails to route the antennas outside the chassis when deploying 4G or LoRaWAN modules.
3. **Driver & Network Configuration**: Most industrial 4G modules require specific USB serial drivers (such as the `option` driver) in Armbian OS. Developers can use `NetworkManager` or `pppd` to establish cellular connections.

4G Module (EC25) Configuration & Testing Guide



(1) Cellular Setup via AT Commands

Plug in the 4G module with the IoT SIM card, boot up the system, and verify the USB serial devices:

```
Bash
lsusb
ls /dev/ttyUSB*
```

Install and launch `minicom` to communicate with the module:

```
Bash
sudo apt install minicom
sudo minicom -D /dev/ttyUSB2
```

(2) Core AT Commands Quick Reference

Function	Command
Communication Test	AT
Send SMS	AT+CMGF=1 $\rightarrow$ AT+CMGS="Phone_Number"

Signal Quality (RSSI)	AT+CSQ
Network Registration	AT+CGREG? or AT+CREG?
Query/Activate Data	AT+QIACT? / AT+QIACT=1
Module Ping Test	AT+QPING=1,"www.baidu.com",1,4

(3) Detailed Status Inspection & Dialing Workflow

SIM Card Status Check: Send `AT+CPIN?` to verify the SIM status.

```
Plain Text
Command:  AT+CPIN?
Response: +CPIN: READY
Meaning:  The SIM card is present and ready (no PIN lock applied).
```

Signal Quality (RSSI) Check: Send `AT+CSQ` to evaluate the wireless environment.

```
Plain Text
Command:  AT+CSQ
Response: +CSQ: 28,99
Meaning:  Excellent signal. The value 28 maps to approx. -57 dBm (range 0-31, >20 is excellent). 99 indicates unknown Rx error rate (normal).
```

Network Registration Check: Send `AT+CGREG?` and `AT+CREG?` to check cellular attachment.

```
Plain Text
Command:  AT+CGREG?  and  AT+CREG?
Response: +CGREG: 0,1  /  +CREG: 0,1
Meaning:  The trailing 1 indicates "Registered, home network", meaning the module has successfully attached to the tower.
```

Current Carrier Query: Send `AT+COPS?` to check the operator and network mode.

```
Plain Text
Command:  AT+COPS?
Response: +COPS: 0,0,"T-Mobile",7
Meaning:  Currently attached to the specific carrier (e.g., T-Mobile). The trailing 7 indicates the connection is in LTE (4G) mode.
```

Data Context Activation: Query via `AT+QIACT?`, then activate the context using `AT+QIACT=1`.

```
Plain Text
Command:  AT+QIACT?   -> Response: OK (If empty, no context is currently active)
Command:  AT+QIACT=1  -> Response: OK
Meaning:  Successfully activates Context ID 1. The module will fetch a private IP from the carrier and enable cellular routing.
```

> Note: Voice calls can be initiated via ATD<number>; if concurrent voice/data is supported by the hardware and carrier plan.

Embedded Ping Test: Run AT+QPING to verify IP-level connectivity directly from the module.

```
Plain Text
Command:  AT+QPING=1,"www.google.com",1,4
Response: +QPING: 0["142.250.190.46",32,45,255]
          +QPING: 0,4,4,0,40,52,45
Meaning:  Successfully pinged the target domain. 4 packets sent, 4 received, 0% loss, a
```

(3) Troubleshooting

Symptoms / Error Codes	Potential Causes	Solutions (AT Commands / Actions)
+CME ERROR: 10 or 13	SIM card improperly inserted / Hot-plug not supported	Power off, verify slot orientation, re-insert, and reboot.
	SIM card is locked by a PIN code	Unlock via: AT+CPIN="Your_PIN_Code".
0,0 or 0,2 from AT+CGREG?	Antenna disconnected or no cellular signal	Ensure the antenna is securely connected to the MAIN RF port.
Registered to network but fails to get IP	Missing APN setting (Common for overseas carriers)	Configure APN: AT+CGDCONT=1,"IP","your_carrier_apn".
/dev/ttyUSB* nodes do not exist	Missing option driver in Linux kernel	Verify driver modules or switch to an officially supported system image.

Lora Module Configuration & Testing Guide

USB

Verify Device Recognition


```
Bash
ls /dev/ttyACM*
udevadm info /dev/ttyACM0 | grep -E "ID_VENDOR|ID_MODEL"
```

Expected output includes:

```
Bash
E: ID_MODEL=STM32_Virtual_ComPort
E: ID_VENDOR=STMicroelectronics
```

TX Test Command

```
Bash
cd ~/sx1302_hal/libloragw
sudo ./test_loragw_hal_tx \
 -u \
 -d /dev/ttyACM0 \
 -r 1250 \
 -m LORA \
 -f 867.1 \
 -s 12 \
 -b 125 \
 -n 1000 \
 -z 100 \
 --dig 3 \
 --pa 0 \
 --pwid 13
```

Expected Output

```
Bash
Opening USB communication interface
INFO: Configuring TTY
INFO: Connect to MCU
INFO: Concentrator MCU version is V01.00.00
INFO: MCU status: sys_time:197172 temperature:37.2oC
Note: chip version is 0x10 (v1.0)
TX done
TX done
```

```
seeed@recomputer-rk3576-devkit:~/sx1302_hal/libloragw$ sudo ./test_loragw_hal_tx -u -r 1250 -m LORA -f 867.1 -s 12
 -b 125 -n 1000 -z 100 --dig 3 --pa 0 --pwid 13 -d /dev/ttyACM0
Opening USB communication interface
INFO: Configuring TTY
INFO: Flushing TTY
INFO: Setting TTY in blocking mode
INFO: Connect to MCU
INFO: Concentrator MCU version is V01.00.00
INFO: MCU status: sys_time:197172 temperature:37.2oC
Note: chip version is 0x10 (v1.0)
INFO: using legacy timestamp
ARB: dual demodulation disabled for all SF
TX done
TX done
TX done
```

Common Errors & Fixes

Error	Cause	Fix
`chip version is 0xFF`	SPI mode used on a USB module	Add `-u` flag
`failed to open COM port ... No such file or directory`	Device disconnected (e.g. after GPIO reset)	Replug or reboot; do **not** run `reset_lgw.sh`
`USB disconnect` in dmesg	`reset_lgw.sh` toggled GPIO and disconnected USB	Skip the reset script for USB modules

Note:

Do not run `reset_lgw.sh` before testing USB-mode modules. The reset script toggles GPIO pins that power-cycle the module and cause USB disconnection. The STM32 MCU handles SX1302 reset internally.

SPI

Verify Device Recognition

```
Bash
ls /dev/ttyACM*
udevadm info /dev/ttyACM0 | grep -E "ID_VENDOR|ID_MODEL"
```

Expected output includes:

```
Bash
E: ID_MODEL=STM32_Virtual_ComPort
```



```
E: ID_VENDOR=STMicroelectronics
```

TX Test Command

```
Bash
cd ~/sx1302_hal/libloragw
sudo ./test_loragw_hal_tx \
 -u \
 -d /dev/ttyACM0 \
 -r 1250 -m LORA -f 867.1 -s 12 -b 125 \
 -n 1000 -z 100 --dig 3 --pa 0 --pwid 13
```

Expected Output

```
Bash
Opening USB communication interface
INFO: Configuring TTY
INFO: Connect to MCU
INFO: Concentrator MCU version is V01.00.00
INFO: MCU status: sys_time:197172 temperature:37.2oC
Note: chip version is 0x10 (v1.0)
TX done
TX done
...
```

HaLow WiFi Module Configuration & Testing Guide

HaLow WiFi requires a device tree overlay to expose the SPI bus and GPIO configuration for the MM6108 chip.

```
Markdown
echo "overlays=recomputer-rk3576-devkit-halow-wifi" | sudo tee -a
/boot/armbianEnv.txt
sudo reboot
```

Verify the file before rebooting：


```
Markdown
sudo reboot
```

After reboot, confirm the driver loaded and the interface appeared:

```
Markdown
tail -2 /boot/armbianEnv.txt
```

Expected output:

```
seeed@recomputer-rk3576-devkit:~$ tail -2 /boot/armbianEnv.txt
overlays=recomputer-rk3576-devkit-halow-wifi
usbstoragequirks=0x2537:0x1066:u,0x2537:0x1068:u
```

The morse-hostapd and morse-wpa_supplicant binaries internally call `morse_cli`, but the installed binary is named `morsectrl`. A symlink is required.

```
Markdown
sudo ln -s /usr/bin/morsectrl /usr/local/bin/morse_cli
```

This only needs to be done once.

Verify Hardware Initialization：

```
Markdown
dmesg | grep -i morse | grep -E "found|Loaded|initialized|MAC"
```

Expected output:

```
Markdown
morse_spi spi3.1: Morse Micro SPI device found, chip ID=0x0306
morse_spi spi3.1: Loaded firmware from morse/mm6108.bin, size 459124,
crc32 0x51d355b9
morse_spi spi3.1: Loaded BCF from morse/bcf_default.bin, size 1251,
crc32 0x941b2a82
morse_spi spi3.1: Firmware initialized
morse_spi spi3.1: Firmware Manifest MAC: 90:03:71:52:9d:8e
```

Check interfaces:

```
Markdown
ip link show | grep -E "wlan0|wlan1|morse0"
```

AP Mode

Create the config


```
Markdown
sudo nano /etc/morse-hostapd.conf
```

Minimal working configuration:

```
Markdown
interface=wlan0
driver=nl80211
ssid=HaLow_Test
country_code=AU        # Change to your region
hw_mode=a
channel=42             # AU channel, see table below
op_class=69
beacon_int=100
dtim_period=2
ieee80211ah=1
s1g_prim_chwidth=1
s1g_prim_1mhz_chan_index=0
s1g_capab=[SHORT-GI-ALL]
wpa=2
wpa_passphrase=12345678
wpa_key_mgmt=WPA-PSK
rsn_pairwise=CCMP
ctrl_interface=/var/run/hostapd
```

Bring interface down and start AP

```
Markdown
sudo ip link set wlan0 down
sudo morse-hostapd /etc/morse-hostapd.conf -B
```

Expected output (success)

```
Markdown
s1g mapped ht channel 159
Full Channel Information
      Operating Frequency: 923000 kHz
      Operating BW: 2 MHz
wlan0: interface state COUNTRY_UPDATE->ENABLED
wlan0: AP-ENABLED
```


Note: RAW Settings: ... Unable to set RAW warnings are non-fatal. They appear because the installed driver version does not support RAW (Restricted Access Window). The AP starts normally.

Check AP status

```
Markdown
sudo morse-hostapd_cli -i wlan0 status
sudo morse-hostapd_cli -i wlan0 all_sta   # list connected clients
```

Station Mode

Create the config:

```
Markdown
sudo nano /etc/morse-wpa_supplicant.conf
```

Minimal configuration:

```
Markdown
ctrl_interface=/var/run/wpa_supplicant
country=AU

network={
   ssid="HaLow_Test"
   psk="12345678"
   key_mgmt=WPA-PSK
}
```

Connect：

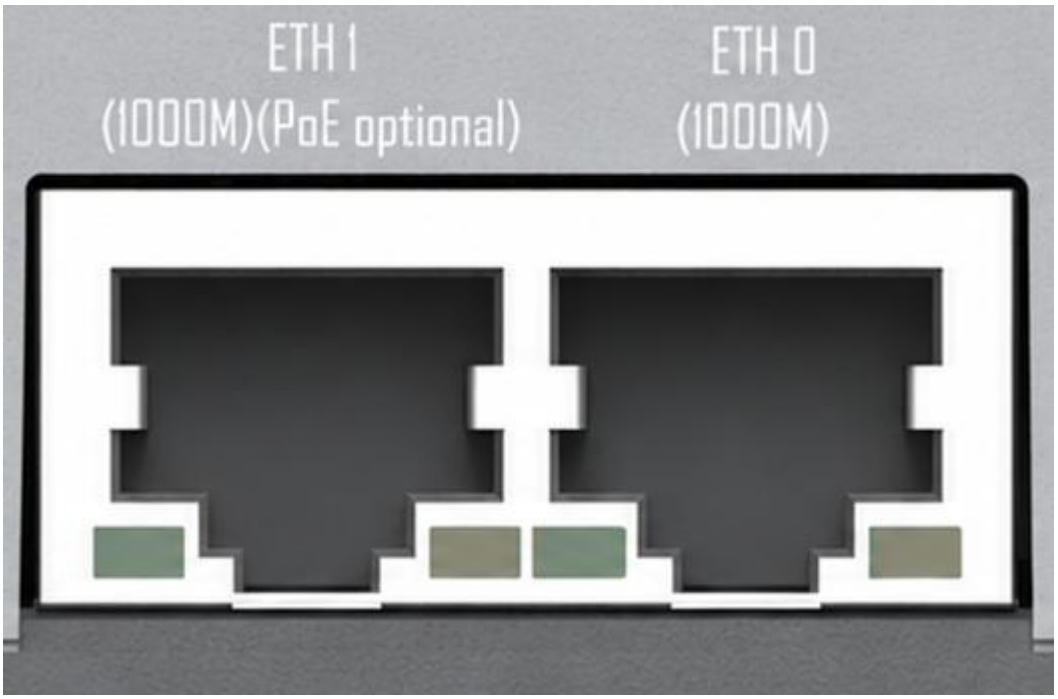
```
Bash
sudo ip link set wlan0 down
sudo morse-wpa_supplicant -i wlan0 -c /etc/morse-
wpa_supplicant.conf -D nl80211 -B
sudo dhclient wlan0
```

Check connection

```
Bash
sudo morse-wpa_cli -i wlan0 status
ip addr show wlan0
```

4.2.10 Ethernet RJ45

The motherboard features two independent **Gigabit Ethernet (RJ45) ports**. The dual-LAN design makes it ideal for building industrial edge gateways, implementing physical network isolation (intranet/extranet), or configuring complex network routing.



Port Definitions & Features :

- **1x Standard Gigabit Ethernet (GbE)**: Supports 10/100/1000 Mbps auto-negotiation Ethernet connection.
- *1x PoE-Enabled Gigabit Ethernet (GbE with PoE PD) : In addition to standard Gigabit networking, this port supports the **PoE PD (Powered Device)** protocol.

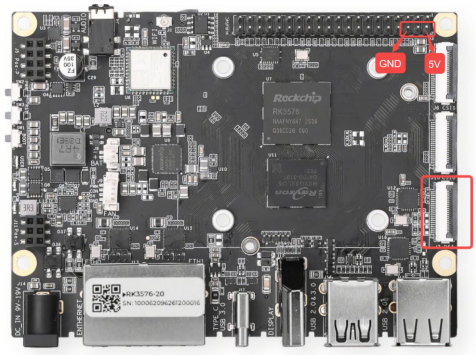
> - **Note:** As a PoE **Powered Device (PD)**, this RK3576 motherboard can **receive power directly through the Ethernet cable** from a PoE switch (PSE), eliminating the need for a separate DC power adapter.

Notes

1. **PoE Module Requirement**: The PoE power-receiving feature is not built directly into the baseboard. It **requires an additional, separately purchased PoE add-on module**. Without this module installed, the port functions solely as a standard Gigabit Ethernet port.
2. **PD vs. PSE Clarification**: Please note that this device acts as a PoE **PD (Power Receiver)**. It **DOES NOT support** PoE output (PSE), meaning it cannot supply power to external devices like PoE IP cameras.

3. **System Network Configuration**: In Armbian OS, these two physical ports are typically enumerated as eth0 and eth1. We recommend using the standard Linux NetworkManager tool (via nmtui or nmcli) or systemd-networkd to configure static IP addresses, network bridging, or link aggregation (bonding).

4.2.11 DSI



The motherboard is equipped with one **4-lane MIPI DSI (22-pin)** display interface, specifically designed for connecting high-resolution embedded LCDs or industrial touch panels.

Interface Features

- **High-Bandwidth Transmission**: Utilizes a 4-lane physical link design, offering significantly higher data throughput compared to traditional 2-lane interfaces. It can smoothly drive 1080P or even 2K resolution HD displays.
- **Physical Form Factor**: Uses a **22-pin 0.5mm pitch** FPC (Flexible Printed Circuit) connector with a flip-lock mechanism.
- **Raspberry Pi Ecosystem Compatibility**: The interface is physically backward-compatible, **supporting direct connections to standard Raspberry Pi DSI screens**, significantly reducing the cost and effort of sourcing accessories for developers.

DSI Configuration & Testing Guide

On the reComputer-RK3576, the MIPI DSI interface is managed via **Device Tree Overlays (DTBO)**. Users must manually load the required overlay file based on the connected display peripheral.

Open the terminal and edit the Armbian environment configuration file:

```
Plain Text
sudo nano /boot/armbianEnv.txt
```

Add the following line to the very end of the file to specify the DSI screen overlay:

```
Plain Text
overlays=recomputer-rk3576-devkit-raspi-7inch-touchscreen
```

Save and exit (Press Ctrl + O, then Enter to save in nano, and Ctrl + X to exit).

Update the package lists and ensure the required multimedia and display plugins are installed:

```
Plain Text
sudo apt-get update
sudo apt install v4l-utils -y
sudo apt-get install gstreamer1.0-plugins-base gstreamer1.0-plugins-good gstreamer1.0-plugins-bad gstreamer1.0-x -y
```

Reboot the system to apply the changes:

```
Plain Text
sudo reboot
```

After rebooting, verify whether the system successfully initialized the graphical desktop environment:

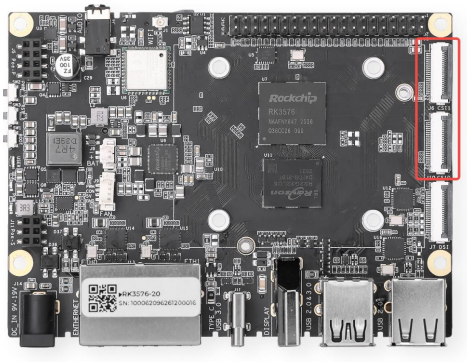
```
Plain Text
echo $XDG_SESSION_TYPE
```

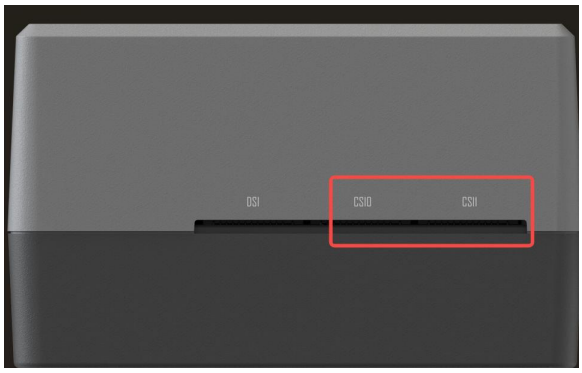
Notes

1. If it outputs x11 or wayland, the DSI display is driven properly and has entered the graphical interface.
2. **FPC Cable Orientation**: When inserting the FPC cable, pay strict attention to the **orientation of the gold contacts**. They must face the contact pins inside the connector. Inserting the cable backward may cause a short circuit or prevent the screen from turning on. Ensure the flip-lock is securely fastened after insertion.
3. **Drivers & Device Tree**: MIPI screens are not "plug-and-play." After connecting a display, you must enable the corresponding panel driver and backlight control nodes in Armbian OS by applying the appropriate Device Tree Overlays (e.g., using armbian-add-overlay) or modifying the DTB files in /boot.
4. **Touch Support**: This 22-pin interface typically integrates I2C signal pins for touch feedback. If you are using a MIPI touchscreen, ensure that both the display driver and the I2C touch IC driver (e.g.,

GT911) are loaded in the operating system.

4.2.12 CSI





The motherboard features two independent **4-lane MIPI CSI (22-pin)** camera interfaces. Powered by the RK3576's robust ISP (Image Signal Processor) and built-in NPU, the dual CSI design is perfect for building stereo vision systems, machine vision inspection stations, or streaming high-framerate video directly for edge AI model inference.

Interface Features

- **Dual 4-Lane Architecture**: Provides two physical ports (`CSI_0` and `CSI_1`), each equipped with a full 4-lane data channel, capable of handling two high-resolution or high-framerate industrial camera modules simultaneously.
- **Physical Form Factor**: Uses a **22-pin 0.5mm pitch** FPC connector.
- **Raspberry Pi Ecosystem Compatibility**: The 22-pin connector is physically and largely pin-compatible, **supporting direct connections to standard Raspberry Pi CSI cameras** (such as official or third-party modules based on IMX219 / IMX477 sensors). This allows developers to quickly validate their vision algorithms.

Notes

1. **Compatibility & Selection**: We strongly recommend using camera modules officially verified by Seeed Studio. Unverified or non-standard cameras may require developers to manually port Linux V4L2 (Video for Linux 2) drivers.
2. **Cable Connection Standards**: Pay close attention to the orientation of the FPC cable's gold contacts. Furthermore, during industrial deployment or robotics framework integration, keep in mind that MIPI signals are highly sensitive to Electromagnetic Interference (EMI). It is recommended to keep the camera ribbon cable length **under 30 cm (12 inches)**.
3. **Concurrent Multi-Camera Capture**: In Linux, the two cameras are

typically enumerated as /dev/video0 and /dev/video1. If your AI application or multimedia framework (like GStreamer or OpenCV) needs to capture both streams simultaneously, ensure proper memory bandwidth allocation and utilize hardware-accelerated plugins for optimal performance.

CSI Configuration & Testing Guide

This section guides you through enabling dual MIPI CSI cameras simultaneously (using the Raspberry Pi Camera V3 as an example).

Edit the environment configuration file:

```
Plain Text
sudo nano /boot/armbianEnv.txt
```

Add the following line to the end of the file to enable the V3 camera drivers for both CAM0 and CAM1:

```
Plain Text
overlays=recomputer-rk3576-devkit-cam0-rpi-v3 recomputer-rk3576-devkit-cam1-rpi-v3
```

Note: Multiple overlays must be separated by a **single space**.

Save the file and reboot the device:

```
Plain Text
sudo reboot
```

After rebooting, use the v4l-utils toolchain to verify and test the camera status.

List available camera devices and nodes:

```
Plain Text
v4l2-ctl --list-devices
```

View supported pixel formats and resolutions for a specific camera (e.g., /dev/video22):

```
Plain Text
v4l2-ctl --list-formats-ext --device=/dev/video22
```

Benchmark camera frame rate: Test the streaming performance under a specific resolution and pixel format (e.g., 3280x2464 MJPG):

```
Plain Text
```



```
v4l2-ctl -d /dev/video22 --set-fmt-video=width=3280,height=2464,pixelformat='MJPG' --stream-mmap=4 --set-selection=target=crop,flags=0,top=0,left=0,width=3280,height=2464 --stream-count=500
```

Install the GStreamer command-line tools:

```
Plain Text
sudo apt install gstreamer1.0-tools -y
```

Run the following pipeline to preview the real-time camera feed (replace `device=/dev/video11` with the actual video node index obtained from `v4l2-ctl --list-devices`):

```
Plain Text
gst-launch-1.0 v4l2src device=/dev/video11 ! video/x-raw,format=NV12,width=3280,height=2464
```

4.2.13 HDMI

The motherboard features one standard **HDMI (Type-A) port**, primarily designed for connecting external monitors, televisions, or industrial control displays, providing high-definition, synchronized audio and video output.

Interface Features

- **Ultra-HD Resolution**: Powered by the robust multimedia capabilities of the RK3576, this interface supports up to 4K ultra-high-definition video output, ideal for rendering sharp GUI dashboards or multiple surveillance video streams.
- **Audio & Video Sync**: The HDMI port transmits both video and digital audio signals simultaneously. If your connected monitor has built-in speakers, system audio will play directly through the HDMI connection without requiring an additional 3.5mm audio cable.
- **Multi-Display Support**: This HDMI port can work concurrently with the onboard Type-C (DP 1.4) and MIPI DSI interfaces. Developers can configure cloned displays or extended desktop modes across up to three screens within the Linux OS.

Notes

1. **Cable Standards**: To ensure stability and prevent screen flickering at 4K resolutions, please strictly use high-quality cables that meet **HDMI 2.0 or higher specifications**.
2. **EDID & Resolution Auto-Detection**: The operating system (e.g., Armbian / Ubuntu) automatically reads the monitor's EDID information upon boot to apply the optimal resolution. If you encounter a "No Signal" issue, you can log in via SSH and use the `xrandr` command to debug the display output status.
3. **Headless Mode Operations**: If you are deploying the device purely as an edge computing node without a physical monitor, the OS may suspend desktop UI rendering to save resources. If you still need VNC remote desktop access in a headless setup, we recommend plugging a "Dummy HDMI Plug (Display Emulator)" into the port.

C5. Configure

This guide focuses on Network connectivity, Remote Access (SSH), and File Management to help you quickly set up your reComputer RK series for development.

5.1 Network connection

5.1.1 Wired network connection

The easiest and most stable way to connect. Operation: Use a network cable to connect the network port of the development board to the router or switch. Configuration: DHCP is enabled by default, plug and play, no additional configuration is required.

5.1.2 Wireless WiFi connection

Graphical desktop (GUI)

You can choose to configure WiFi through a graphical desktop or command-line tools. If you have a monitor connected and are in a desktop environment, this is the most intuitive way to configure it. Open the network menu: Tap the network icon in the top right corner or taskbar (usually a fan-shaped WiFi logo). Select Network: Tap your WiFi name in the list. Enter Password: Enter your WiFi password in the pop-up dialog box and tap "Connect."

Command Line (nmcli)

If you operate via SSH or serial terminal, you can configure it using NetworkManager's command-line tools.

Commonly used operation cheat sheet

Function	Command	Description
Scan WiFi	nmcli device wifi list	List all available hotspots nearby
Connect to WiFi	nmcli device wifi connect "SSID" password "PWD"	Replace SSID and PWD with actuals
Check status	nmcli connection show	Displays all current network connections
Disconnect	nmcli connection down <connection_name>	Disconnect the specified network

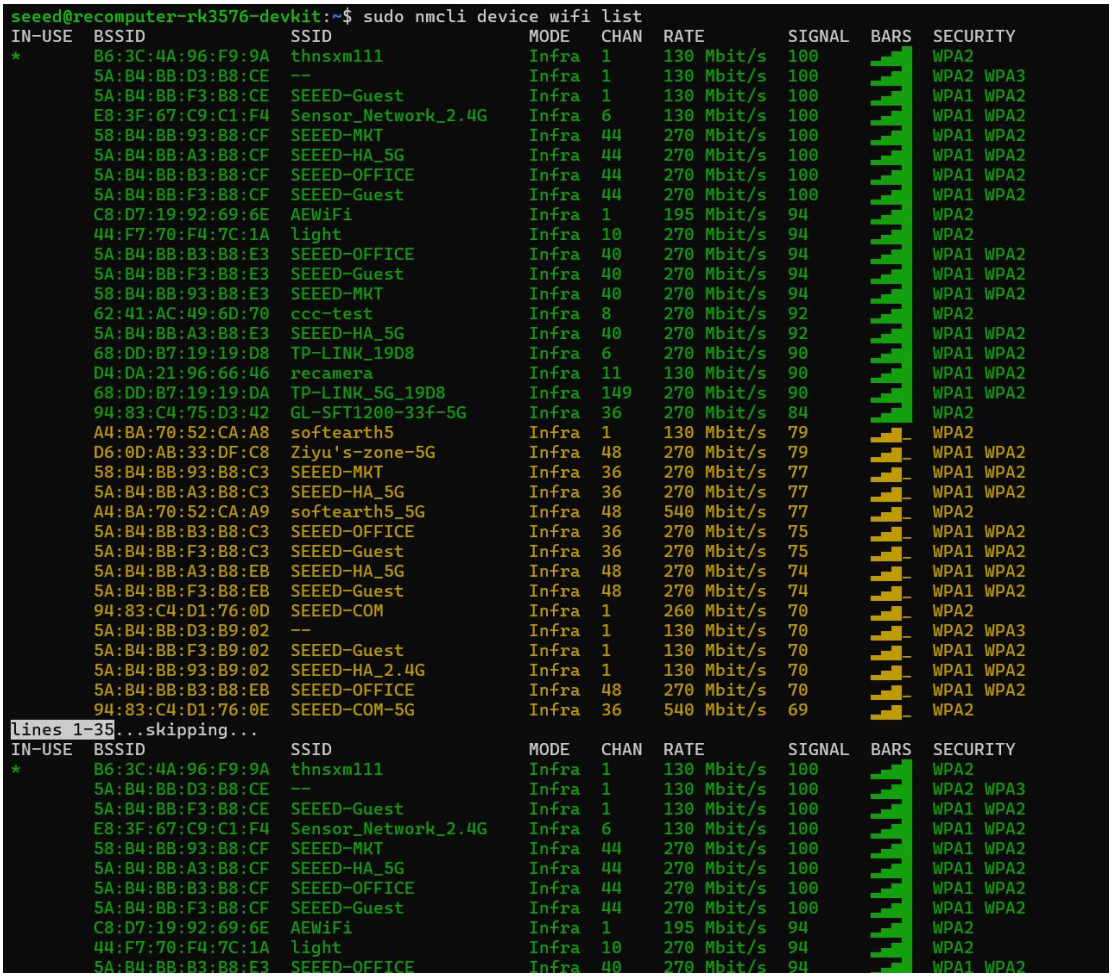
Example of operation

Step 1: Scan for available WiFi

bashCopy

```
Plain Text
sudo nmcli device wifi list
```

Example output:



bashCopy

```
Plain Text
IN-USE  BSSID              SSID                MODE   CHAN  RATE       SIGNAL
BARS  SECURITY
*       B6:3C:4A:96:F9:9A  thnsxm111           Infra  1     130 Mbit/s  100
▂▄▆█  WPA2
        5A:B4:BB:D3:B8:CE  --                  Infra  1     130 Mbit/s  100  ▂▄▆█
WPA2 WPA3
        5A:B4:BB:F3:B8:CE  SEEED-Guest         Infra  1     130 Mbit/s  100
▂▄▆█  WPA1 WPA2
        E8:3F:67:C9:C1:F4  Sensor_Network_2.4G  Infra  6     130 Mbit/s  100
▂▄▆█  WPA1 WPA2
```

Step 2: Connect to WiFi Let's say we want to connect to a network called , and the password is: 123 123askxxx

bashCopy

```
Plain Text
sudo nmcli device wifi connect "123" password "123askxxx"
```

After a successful connection, you'll be prompted: Device 'wlan0' successfully activated with '...'

5.1.3 Advanced configuration

In some development scenarios, such as as a server, you may need to have a fixed IP address.

bashCopy

```
Plain Text
# 1. Modify connection configuration (Assuming connection name is "Wired connection 1")
nmcli con mod "Wired connection 1" ipv4.addresses 192.168.1.100/24
nmcli con mod "Wired connection 1" ipv4.gateway 192.168.1.1
nmcli con mod "Wired connection 1" ipv4.dns "8.8.8.8 114.114.114.114"
nmcli con mod "Wired connection 1" ipv4.method manual

# 2. Restart the connection to apply settings
nmcli con up "Wired connection 1"
```

5.2 Remote login guide (SSH)

Hardware: reComputer RK3576 / RK3588 Operating System: Debian 12 This guide provides instructions on how to access your reComputer remotely using the Secure Shell (SSH) protocol.

5.2.1 Prerequisites

- The reComputer must be powered on and connected to the same Local Area Network (LAN) as your primary computer.
- An Ethernet connection is recommended for initial setup to ensure stability.

5.2.2 Identify the Device IP Address

You need the IP address of the reComputer to establish a connection.

- Via Local Terminal: If you have a monitor and keyboard connected to the reComputer, run:

bashCopy

```
Plain Text
hostname -I
```

```
seeed@recomputer-rk3576-devkit:~$ hostname -I
10.1.1.171 10.142.30.205 fd85:8202:f62a::269 fd85:8202:f62a:0:c1e5:f08a:2216:67af
```

- Via Router Dashboard: Access your router's web interface and look for a device named recomputer or similar in the DHCP client list.

5.2.3 Connection Methods

Option A: Windows (PowerShell or Command Prompt) Modern Windows versions include a built-in SSH client.

1. Press Win + R, type cmd, and hit Enter.
2. Input the following command (replace 10.0.x.x with your actual IP):

bashCopy

```
Plain Text
ssh seeed@10.1.x.x
```

```
C:\Users\seeed>ssh seeed@10.1.1.171
The authenticity of host '10.1.1.171 (10.1.1.171)' can't be established.
ED25519 key fingerprint is SHA256:TPdhAXgoH2aV+ww7yMyGrv5qTpz4VLWzjczx9v/o/WA.
This host key is known by the following other names/addresses:
    C:\Users\seeed/.ssh/known_hosts:119: 10.1.1.170
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.1.1.171' (ED25519) to the list of known hosts.
seeed@10.1.1.171's password:
Permission denied, please try again.
seeed@10.1.1.171's password:

 v25.11 rolling for recomputer rk3576 devkit running Armbian Linux 6.1.115-vendor-rk35xx

 Packages:     Debian stable (bookworm), possible distro upgrade (trixie)
 Updates:      Kernel upgrade enabled and 17 packages available for upgrade
 Support:      DIY (custom image)
 IPv4:         (LAN) 10.1.1.171, 10.142.30.205 (WAN) 113.87.161.60
 IPv6:         fd85:8202:f62a::269, fd85:8202:f62a:0:c1e5:f08a:2216:67af

 Performance:

 Load:         8%                  Uptime:       2 min
 Memory usage: 5% of 7.74G
 CPU temp:     28°C                Usage of /:   4% of 116G

 Commands:

 Configuration : armbian-config
 Upgrade       : armbian-upgrade
 Monitoring    : htop

seeed@recomputer-rk3576-devkit:~$ source ~/rknn_env/bin/activate
```

3. Type yes when asked to confirm the host's fingerprint.

4. Enter the password seeed (characters will not appear as you type).

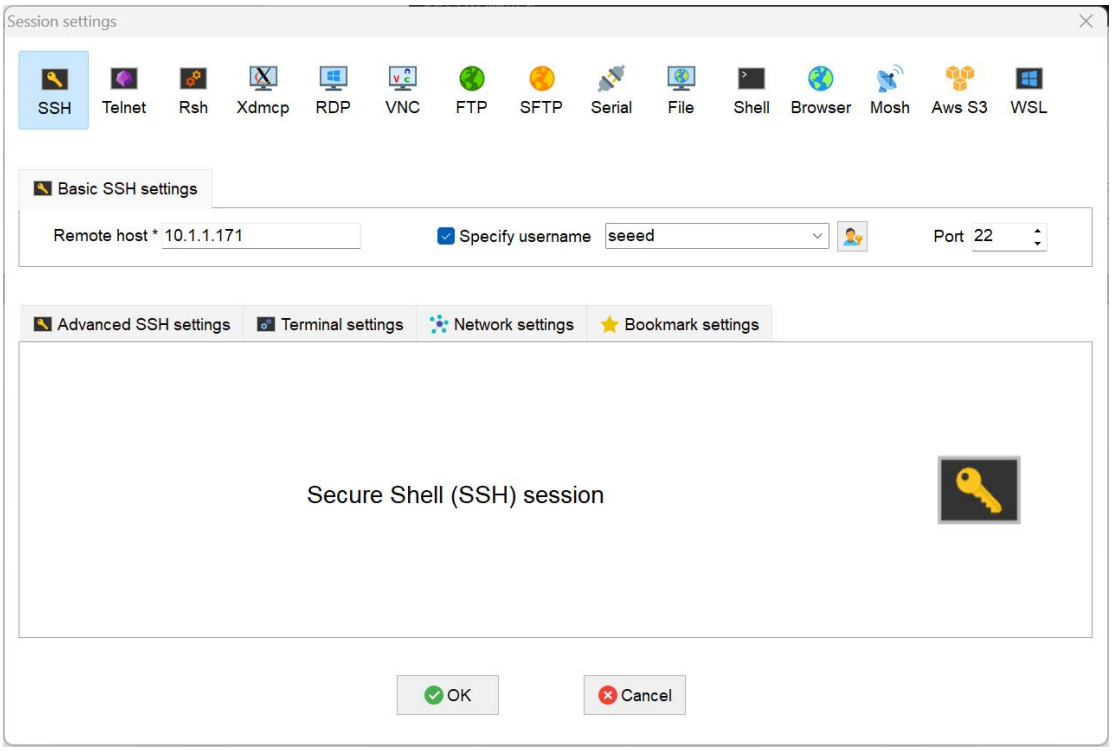
Option B: Linux / macOS Terminal Open your native terminal and run:

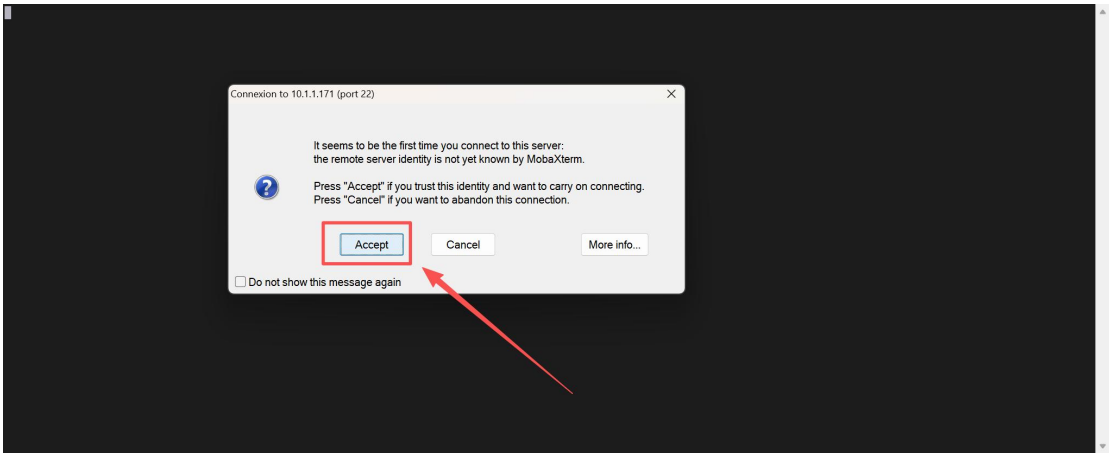
bashCopy

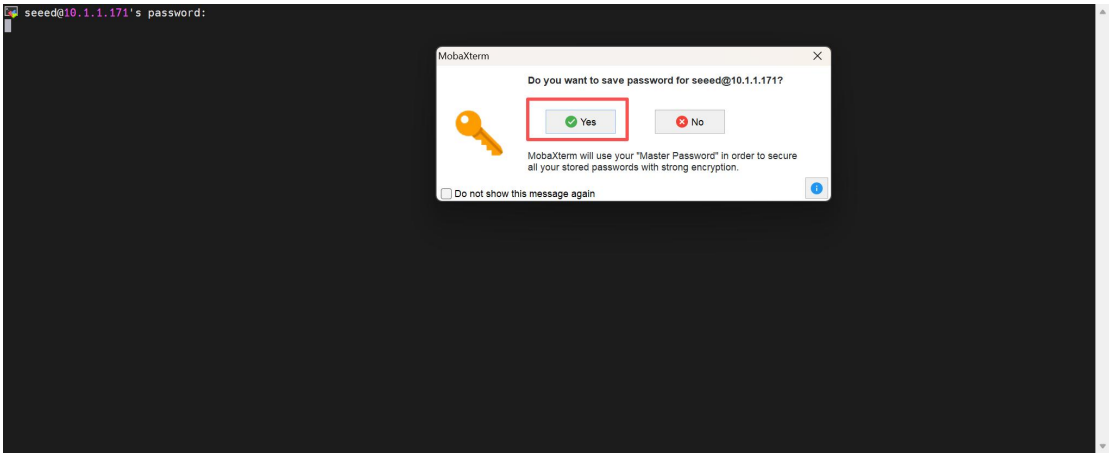
```
Plain Text
ssh seeed@<Your_Device_IP>
```

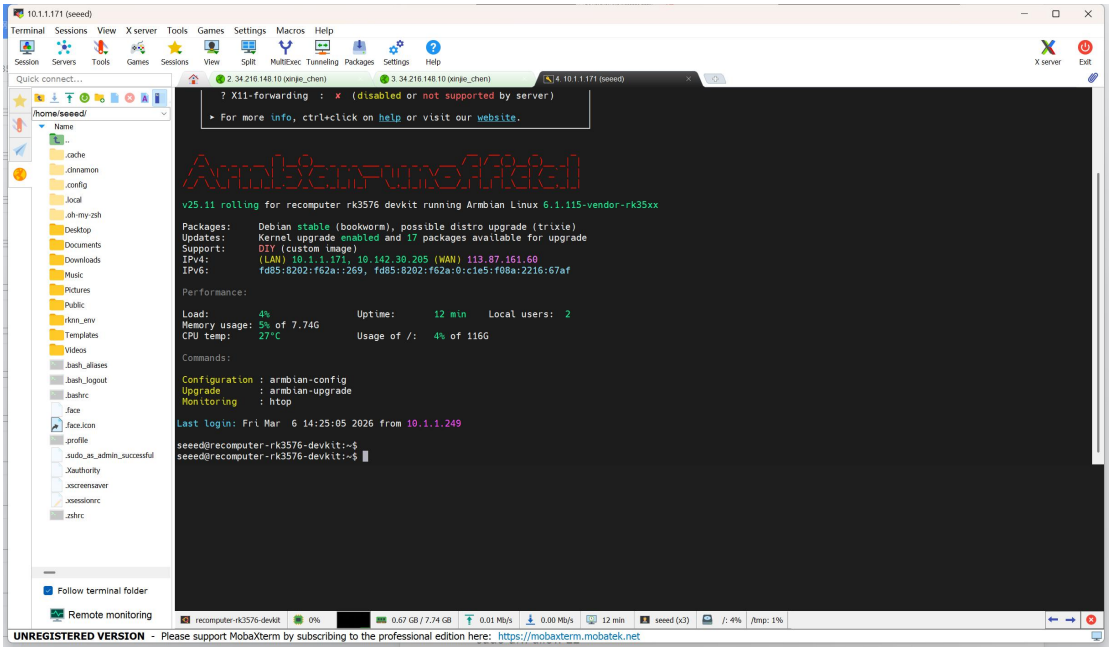
Option C: Professional Tools (Recommended) For a more robust development environment, consider:

MobaXterm: Features a built-in SFTP sidebar for easy file dragging.









5.2.4 Troubleshooting

. Connection Refused: Ensure the SSH service is active on the reComputer. You can check this locally by running:

bashCopy


```
Plain Text
sudo systemctl status ssh
```

- Firewall Issues: If the service is running but blocked, allow port 22 through the firewall:

bashCopy

```
Plain Text
sudo ufw allow 22
```

- Host Key Verification Failed: If you recently reflashed the OS on the same IP address, clear the old key on your PC using:

bashCopy

```
Plain Text
ssh-keygen -R <Your_Device_IP>
```

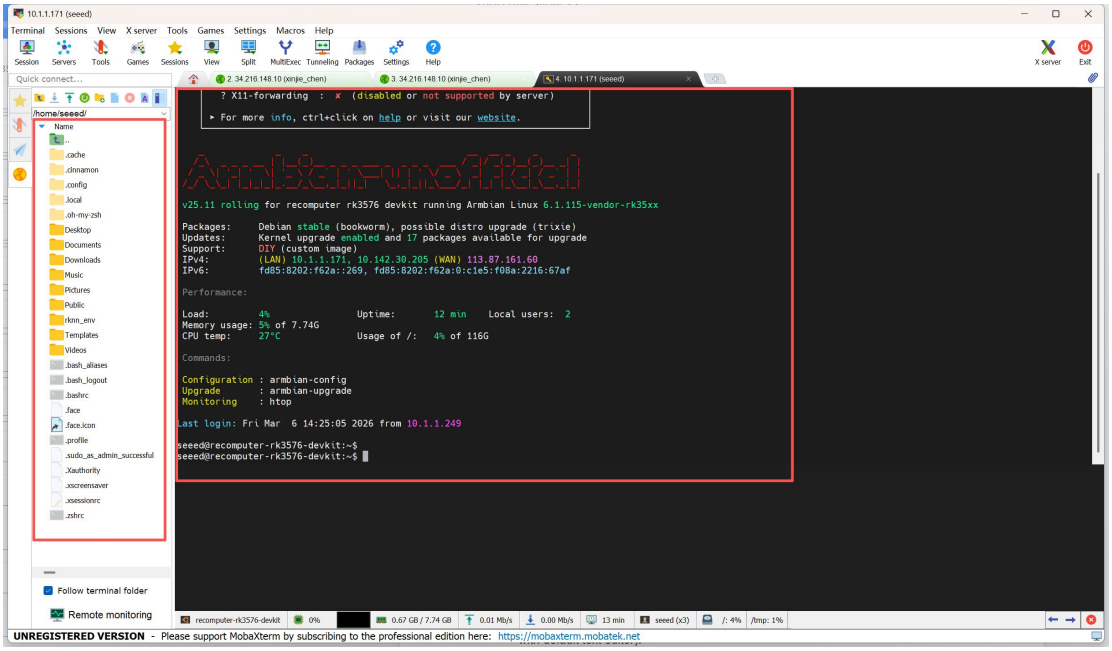
5.3 File Transfer Guide (MobaXterm SFTP)

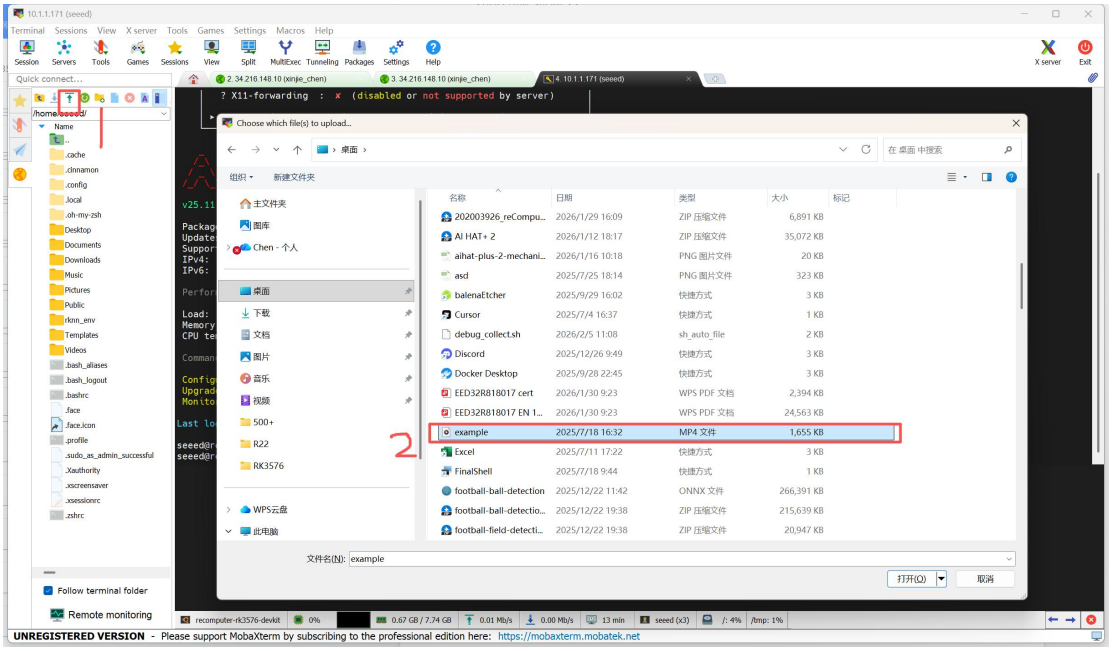
- Device: reComputer RK3576 / RK3588
- OS: Debian 12
- Core Tool: MobaXterm(Built-in SFTP Sidebar)

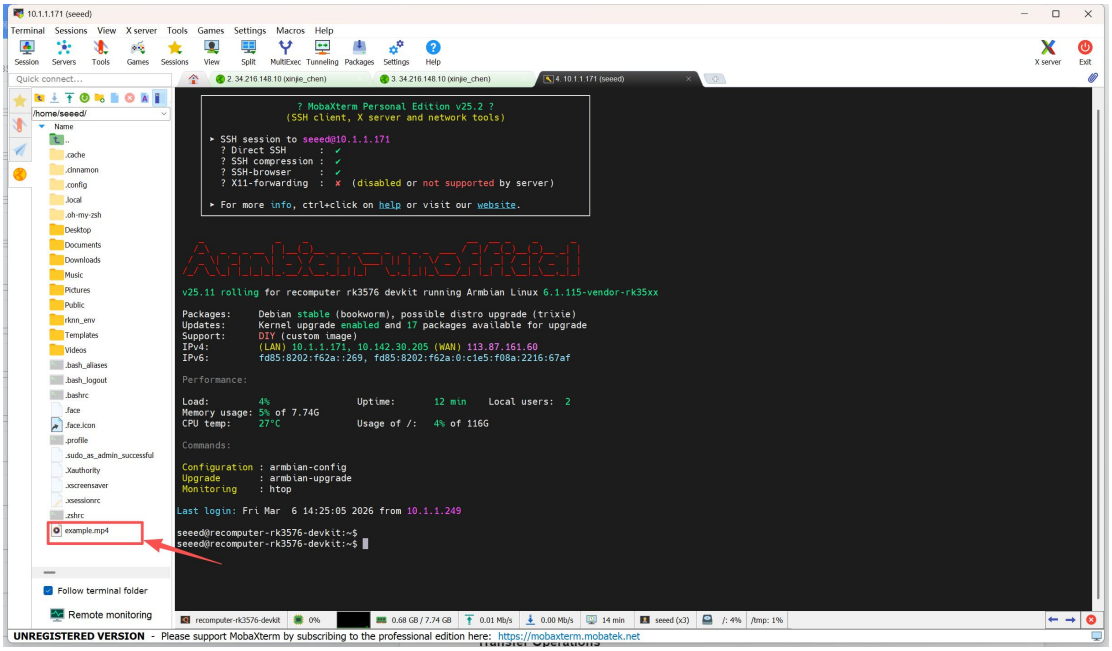
5.3.1 Interface Overview

Once you have successfully logged in via SSH, MobaXterm will automatically open an SFTP tab on the left side of the interface.

- Left Panel: Displays the file system on the reComputer (Remote).
- Right Panel: Your terminal command-line window.







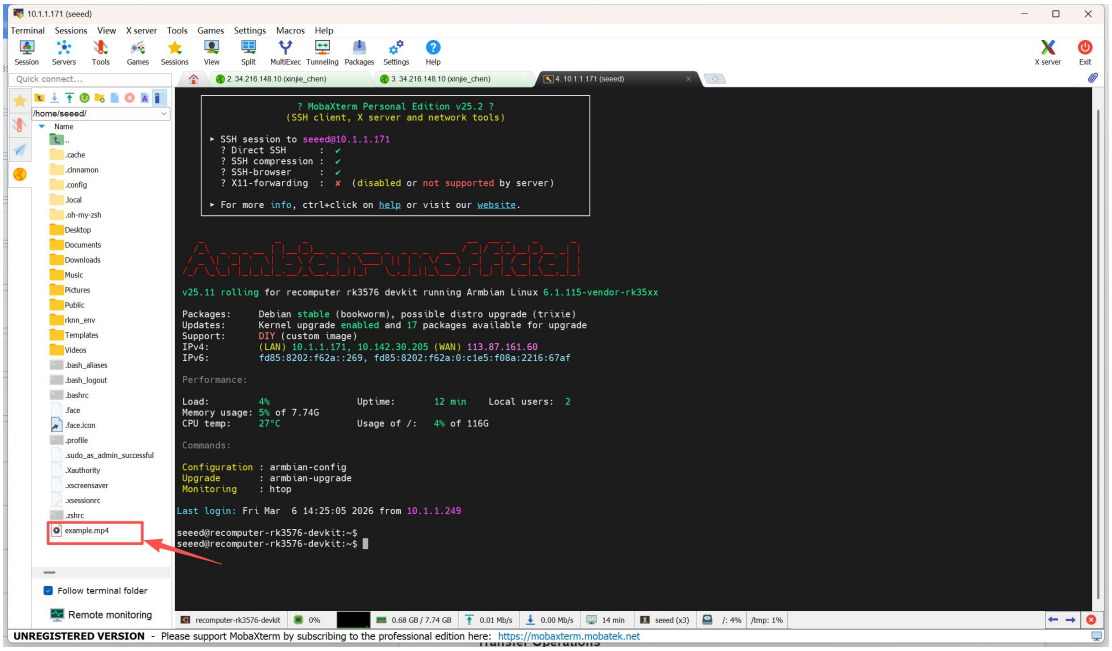
5.3.2 Transfer Operations

Upload

- Drag & Drop: Simply select files from your Windows folder and drag them into the SFTP panel on the left.
- Upload Button: Click the blue up-arrow icon above the left panel to select and upload files from your PC.

Download

- Drag & Drop: Select files in the left panel and drag them out to your Windows desktop or folder.
- Right-click, or select the "Download" button at the top: Right-click the file and select Download.



5.3.3 Advanced Features

- Live Edit
  - Operation: Double-click a text file in the left panel (or right-click and select Open with default text editor).
  - Sync: Edit and save; MobaXterm will prompt you to sync changes back to the reComputer. Click Yes.
- Quick Navigation
  - Click the "Follow terminal path" icon (yellow square) to sync the SFTP view with your terminal's current directory (cd).

5.3.4 Important Notes

- Permissions: The seeed user only has write permissions in /home/seeed by default.
- System Files: To edit system files (e.g., in /etc/), upload to /home/seeed first, then use sudo mv in the terminal.
- Reconnection: If the SFTP panel becomes unresponsive, click the Refresh button to resync.

C6. F&Q

6.1 Troubleshooting Remote SSH Connection Failure

If you encounter the error `ssh: connect to host 192.xxx.xxx.xxx port 22: Connection refused` when trying to connect remotely, and you have direct access to the device via a serial console or an external display, please run the following commands in the local terminal to check and enable the service:

```bash
# Check the status of the SSH service
sudo systemctl status ssh

# If it shows "inactive" or is not running, try to start and enable it
sudo systemctl start ssh
sudo systemctl enable ssh
```

C7. Warranty & Support

7.1 Warranty

1. From the date of sale, the company provides 12 months free warranty for the products.

2. Warranty coverage is limited to products purchased from the official Seeed Studio website or authorized distributors. Customers need to keep receipts and purchase vouchers.

3. The products to be repaired shall be properly packaged and transported, and the customer shall be responsible for any loss or damage during transportation.

4. During the warranty period, the freight and maintenance costs arising from product quality failures shall be borne by Seeed Studio. If the warranty period exceeds 24 months, Seeed Studio will charge the fee for replacing parts according to the product failure, and the freight is borne by the user.

5. During the free warranty period, in case of any of the following events, Seeed Studio has the right to refuse service or charge materials and service fees at its discretion.

Product failure or damage caused by improper use by users.

The product label is damaged and the product information cannot be identified.

Even within the warranty period, if the product has functional issues or is difficult to repair due to improper customer use, unaut- horized disassembly or modification, poor operating environment, improper maintenance, accidents, or other reasons. Seeed Studio reserves the right to make judgments on the above situations and collect maintenance fees.

Other unavoidable external factors cause product failure and damage.

The above warranty regulations are only applicable to the above Seeed Studio reComputer RK3576 series, other products are not applicable!

7.2 Support

Quick start guide:

reComputer AI Lab | Edge AI Models & Tools

reComputer AI Lab Ecosystem:

Engineered for rapid deployment with minimal complexity. Utilize simple Docker commands to instantly pull and run containers (e.g., launching `deepseek-r1-distill-qwen:1.5b` in seconds). Explore the full range of supported pre-configured environments on the https://sensecraft.seeed.cc/ai-lab/models

Tech support email:

If you encounter any issues while deploying or testing, please don't hesitate to contact our technical support team at techsupport@seeed.io, or refer to our online knowledge base, https://wiki.seeedstudio.com.

Customized service email:

For further information about customizations, welcome you to directly reach out at edge@seeed.cc, we will provide prompt reply.

Discord:

Discord community:

Welcome to join our official community, where you can exchange product-related questions and get relevant support.

https://discord.seeed.cc