

NanoPi R4S

Contents

- 1 Introduction
- 2 Hardware Spec
- 3 Diagram, Layout and Dimension
 - 3.1 Layout
 - 3.2 Differences Between R4S Standard Version & R4S Enterprise Version
- 4 Get Started
 - 4.1 Essentials You Need
 - 4.2 Install OS
 - 4.2.1 Download Image Files
 - 4.2.2 Flash to TF
- 5 Work with FriendlyWrt
 - 5.1 Introduction to FriendlyWrt
 - 5.2 First boot
 - 5.3 Account & Password
 - 5.4 Login FriendlyWrt
 - 5.5 Recommended security settings
 - 5.6 Change LAN IP in LuCI
 - 5.7 Safe shutdown operation
 - 5.8 Soft Factory Reset
 - 5.9 Install Software Packages
 - 5.9.1 Set up openwrt official opkg source
 - 5.9.2 Update Package List
 - 5.9.3 List Available Packages
 - 5.9.4 List Installed Packages
 - 5.9.5 Install Packages
 - 5.9.6 Remove Packages
 - 5.10 Disable IPv6
 - 5.11 Configure the function of the user button
 - 5.12 Configuring Quectel EC20 (4G module) dial-up networking
 - 5.13 Some common issues of FriendlyWrt
 - 5.14 Use USB2LCD to view IP and temperature
 - 5.15 How to Control Fan Speed for Cooling
 - 5.16 How to use USB WiFi
 - 5.16.1 Check USB WiFi Device with Command Line Utility
 - 5.16.2 Configure a USB WiFi Device as AP
 - 5.16.3 Common USB WiFi issues
 - 5.16.4 Change the default WiFi hotspot configuration
 - 5.17 Work with Docker Applications
 - 5.17.1 Work with Docker: Install JellyFin
 - 5.17.2 Work with Docker: Install Personal Nextcloud
 - 5.17.3 Expand Docker Storage
 - 5.17.4 Docker FAQ and solutions
 - 5.17.4.1 Unable to access the network services provided by the Docker container
 - 5.18 Mount smbfs
 - 5.19 Use sdk to compile the package
 - 5.19.1 Install the compilation environment
 - 5.19.2 Download and decompress sdk from the network disk
 - 5.19.3 Compile the package
 - 5.19.4 Install the ipk to NanoPi
 - 5.20 Build FriendlyWrt using GitHub Actions
- 6 Work with FriendlyCore
 - 6.1 FriendlyCore User Account
 - 6.2 Update Software Packages
 - 6.3 Setup Network Configurations
 - 6.3.1 Set static IP address
 - 6.3.2 Set a DNS
 - 6.3.3 Set up to use another network interface
 - 6.4 Setup Wi-Fi
 - 6.4.1 WiFi models supported
 - 6.4.1.1 M.2 WiFi Module
 - 6.4.1.2 Usb Dongle
 - 6.5 Install the kernel-header package
 - 6.6 Build kernel-header deb package
 - 6.7 Config status LEDs
 - 6.8 Delete Qt5 and related files
 - 6.9 Configure parameters for serial port
- 7 Work with Debian11 Desktop
 - 7.1 Introduction to Debian11 Desktop
 - 7.2 Account & Password
 - 7.3 View IP address
 - 7.4 Connect to Debian via SSH
 - 7.5 Update Software Packages
 - 7.6 Install x11vnc Server on Debian for Remote Access
 - 7.6.1 Install x11vnc server

- 7.6.2 Set your password
 - 7.6.3 Setup x11vnc server with systemd auto start up
 - 7.6.4 Testing remote access
- 7.7 Install the kernel-header package
- 7.8 Change time zone
 - 7.8.1 Check the current time zone
 - 7.8.2 List all available time zones
 - 7.8.3 Set the time zone (e.g. Shanghai)
- 7.9 Change startup LOGO and Wallpaper
 - 7.9.1 Change startup LOGO
 - 7.9.2 Change Wallpaper
- 7.10 Soft Factory Reset
- 7.11 Start the program automatically at startup(For example Kodi)
- 7.12 Disable auto-mounting
- 7.13 Setup Chinese language and Input method
 - 7.13.1 Setup Chinese language
 - 7.13.2 Installing Chinese input method
- 7.14 Installing Plex Multimedia Server
- 7.15 Install Docker on Debian
- 7.16 How to test NPU
- 8 Work with Debian10 Desktop
- 9 Buildroot Linux
- 10 How to Compile
 - 10.1 Setup Development Environment
 - 10.1.1 Method 1: Using docker to cross-compile
 - 10.1.2 Method 2: Setup build environment on the host machine
 - 10.1.2.1 Install required packages
 - 10.1.2.2 Setting the compiler path
 - 10.2 Build Openwrt/Friendlywrt
 - 10.2.1 Download Code
 - 10.2.1.1 FriendlyWrt 24.10
 - 10.2.1.2 FriendlyWrt 23.05
 - 10.2.2 First compilation step
 - 10.2.3 Secondary compilation steps
 - 10.2.4 Build u-boot only
 - 10.2.5 Build kernel only
 - 10.2.6 Build friendlywrt only
 - 10.3 Build Buildroot
 - 10.4 Build Other Linux
 - 10.4.1 Kernel and u-boot versions
 - 10.4.2 Build kernel linux-v4.4.y
 - 10.4.3 Build u-boot v2014.10
 - 10.4.4 Build kernel linux-v4.19.y
 - 10.4.5 Build kernel linux-v6.1.y
 - 10.4.6 Build u-boot v2017.09
 - 10.4.7 Running the build
 - 10.4.7.1 Install to target board
 - 10.4.7.1.1 MBR partition
 - 10.4.7.1.2 GPT partition
 - 10.4.7.2 Packaging and creating an SD image
 - 10.4.7.3 USB flashing
 - 10.4.7.3.1 Linux
 - 10.5 Build the code using scripts
 - 10.5.1 Download scripts and image files
 - 10.5.2 Compile the kernel
 - 10.5.3 Compile the kernel headers
 - 10.5.4 Compile the uboot
 - 10.5.5 Generate new image
 - 10.6 Building AOSP from source
 - 10.6.1 Hardware and Software Requirements
 - 10.6.2 Compile Android10
 - 10.6.2.1 Download Android10 Source Code
 - 10.6.2.2 Generate Image File
 - 10.6.2.3 Make OTA Packages
 - 10.6.2.4 Update System with New Image
 - 10.6.3 Compile Android8.1
 - 10.6.3.1 Download Android8.1 Source Code
 - 10.6.3.2 Generate Image File
 - 10.6.3.3 Update System with New Image
 - 10.6.4 Compile Android7
 - 10.6.4.1 Download Android7 Source Code
 - 10.6.4.2 Generate Image File
 - 10.6.4.3 Update System with New Image
- 11 Using On-Board Hardware Resources
 - 11.1 Access Serial Interface
 - 11.2 DTS files
- 12 Backup rootfs and create custom SD image (to burn your application into other boards)
 - 12.1 Backup rootfs
 - 12.2 Making a bootable SD card from a root filesystem
- 13 Configuring kernel command line parameters (only support for kernel4.4)
 - 13.1 eMMC Boot
 - 13.2 SD Boot
- 14 More OS Support
 - 14.1 DietPi
- 15 Link to Rockchip Resources
- 16 Schematic, PCB CAD File
- 17 Known Issues List

- 18 Update Log
 - 18.1 2023-12-01
 - 18.1.1 FriendlyWrt
 - 18.2 2023-05-26
 - 18.2.1 FriendlyWrt
 - 18.3 2023-04-26
 - 18.3.1 FriendlyWrt:
 - 18.4 2023-02-10
 - 18.4.1 Added Debian11
 - 18.5 2023-01-09
 - 18.5.1 FriendlyCore:
 - 18.6 2022-12-04
 - 18.6.1 FriendlyWrt:
 - 18.7 2022-09-06
 - 18.7.1 FriendlyWrt:
 - 18.8 2022-08-03
 - 18.8.1 FriendlyWrt:
 - 18.9 2022-07-27
 - 18.9.1 FriendlyWrt:
 - 18.10 2021-10-29
 - 18.10.1 FriendlyWrt:
 - 18.11 2021-08-31
 - 18.11.1 FriendlyWrt:
 - 18.12 2020-12-23

1 Introduction

- The NanoPi R4S(as "R4S") is an open source platform with dual-Gbps Ethernet ports designed and developed by FriendlyElec for IoT applications.
- The NanoPi R4S uses the RK3399 SoC. It has two Gbps Ethernet ports and 1G/4G DDR4 RAM. FriendlyElec ported an OpenWrt system for it. It works with Docker CE. It is a good platform for developing IoT applications, NAS applications etc.

2 Hardware Spec

- SoC: Rockchip RK3399
 - CPU: big.LITTLE, Dual-Core Cortex-A72(up to 2.0GHz) + Quad-Core Cortex-A53(up to 1.5GHz)
 - GPU: Mali-T864 GPU, supports OpenGL ES1.1/2.0/3.0/3.1, OpenCL, DX11, and AFBC
 - VPU: 4K VP9 and 4K 10bits H265/H264 60fps decoding, Dual VOP, etc
- PMU: RK808-D PMIC, cooperated with independent DC/DC, enabling DVFS, software power-down, RTC wake-up, system sleep mode
- RAM: 1GB DDR3/4GB LPDDR4
- Flash: no Onboard eMMC
- Ethernet: one Native Gigabit Ethernet, and one PCIe Gigabit Ethernet
- USB: two USB 3.0 Type-A ports
- Pin header extension interface
 - 2x5-pin header: SPI x 1, I2C x 1
 - 4-pin header: USB 2.0
- microSD Slot x 1
- Debug: one Debug UART, 3 Pin 2.54mm header, 3V level, 1500000bps
- LEDs: 1 x power LED and 3 x GPIO Controlled LED (SYS, LAN, WAN)
- others:
 - 2 Pin 1.27/1.25mm RTC battery input connector
 - one User Button
 - one 5V Fan connector
- Power supply: DC 5V/3A, via USB-C connector or Pin header
- PCB: 8 Layer, 66 mm x 66 mm
- Temperature measuring range: 0°C to 80°C

3 Diagram, Layout and Dimension

3.1 Layout

- 2x5-pin header

Pin#	Assignment	Pin#	Assignment
1	VDD_5V	2	VDD_3.3V
3	VDD_5V	4	GPIO4_C0/I2C3_SDA(3V)
5	GND	6	GPIO4_C1/I2C3_SCL(3V)
7	GPIO1_B1/SPI1_CLK	8	GPIO1_B0/SPI1_TXD/UART4-TX
9	GPIO1_B2/SPI1_CSn	10	GPIO1_A7/SPI1_RXD/UART4-RX

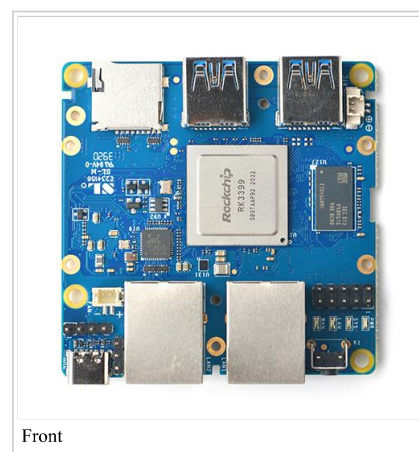
- 4-pin header

1	2	3	4
VDD_5V	USB_DM	USB_DP	GND

- Debug UART Pin Spec



Overview



Front

3V level signals, 1500000bps

Pin#	Assignment	Description
1	GND	0V
2	UART2DBG_TX	output
3	UART2DBG_RX	input

■ USB Port

Each USB 3.0 port has 2A overcurrent protection.

■ RTC

RTC backup current is 27uA.

Connector P/N: Molex 53398-0271

Notes

1. Power Input : 5V/3A, via USB Type-C(USB PD Specification is not supported) or Pin1&Pin2 of the 2x5-pin header

3.2 Differences Between R4S Standard Version & R4S Enterprise Version

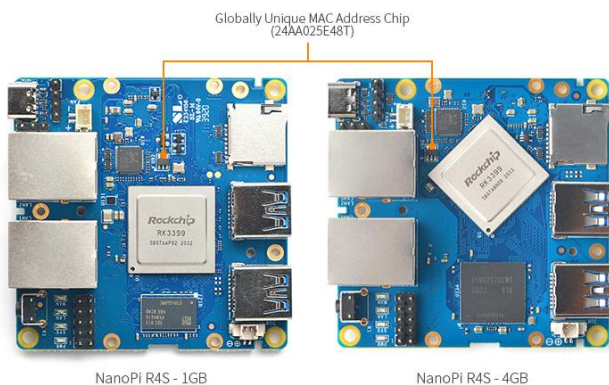
The R4S **Enterprise version** has a built-in EEPROM chip (Model: 24AA025E48T) which has a globally unique MAC address. This is a permanent and temper-proof address.

The R4S **Standard version** doesn't have this chip and has a MAC address that is generated by a software utility. Apart from this difference, the two versions have the same hardware configuration.

The Standard version doesn't have a MAC-address built-in EEPROM chip while the Enterprise version does. Both the Standard version and the Enterprise version have the same network chips (RealTek RTL8211E and R8111H). For more details please refer to the following screenshot. Retail users are recommended to choose a Standard version and enterprise users are recommended to choose an Enterprise version.

Tips: most of the existing embedded ARM boards such as RPi 3B have MAC addresses that are generated via software utilities. This generally doesn't have impact on network communication. Globally unique MAC addresses lead to better network performance in complicated network situations and are better for large-scale enterprise applications that manage multiple network devices and operations such as IP binding.

Position of the EEPROM Chip:

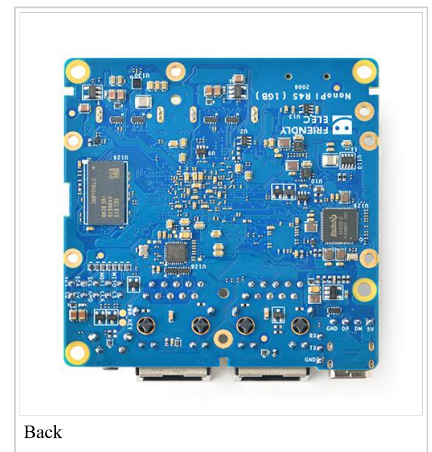


The NanoPi R4S Enterprise version has a globally unique MAC address which is by default allocated to the CPU's internal Ethernet (rtl8211e) and the name of the device is eth0. This device is named "LAN2" on the PCB and "WAN" on the board's case. This has been configured by default in FriendlyWrt.

Check MAC address on FriendlyWrt's website:



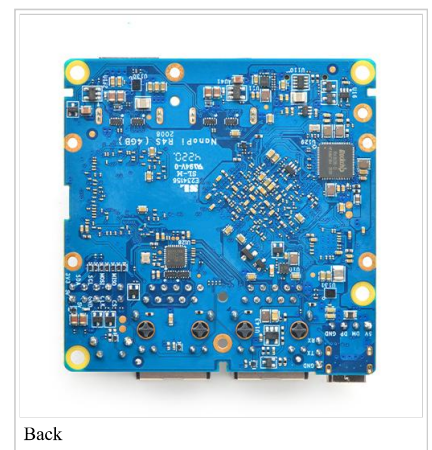
Check it in a command-line utility:



Back



Front



Back



Case

Image Files	
rk3399-sd-friendlywrt-24.10-YYYYMMDD.img.gz	FriendlyWrt image file, based on OpenWrt 24.10, kernel version 6.6.y
rk3399-sd-friendlywrt-24.10-docker-YYYYMMDD.img.gz	FriendlyWrt image file, built-in docker, based on OpenWrt 24.10, kernel version 6.6.y
rk3399-sd-friendlywrt-23.05-YYYYMMDD.img.gz	FriendlyWrt image file, based on OpenWrt 23.05, kernel version 6.6.y
rk3399-sd-friendlywrt-23.05-docker-YYYYMMDD.img.gz	FriendlyWrt image file, built-in docker, based on OpenWrt 23.05, kernel version 6.6.y
rk3399-sd-ubuntu-noble-core-4.19-arm64-YYYYMMDD.img.zip	Ubuntu 24.04 Core No desktop environment, command line only Kernel version 4.19.y
rk3399-sd-debian-bookworm-core-4.19-arm64-YYYYMMDD.img.gz	Debian 12 Core No desktop environment, command line only Kernel version 4.19.y
rk3399-sd-debian-bullseye-minimal-4.19-arm64-YYYYMMDD.img.gz	Debian 11(Bullseye) Desktop Uses LXDE as default desktop No pre-installed recommended software Supports hardware acceleration Kernel version 4.19.y
rk3399-sd-debian-bullseye-desktop-4.19-arm64-YYYYMMDD.img.gz	Debian 11(Bullseye) Desktop Uses LXDE as default desktop Pre-installed mpv, smplayer and chromium browser Supports hardware acceleration Kernel version 4.19.y
Other Image	
FriendlyWrt (Github Actions)	FriendlyWrt (https://github.com/friendlyarm/Actions-FriendlyWrt/releases)
Alpine-Linux (Github Actions)	Alpine-Linux (https://github.com/friendlyarm/Actions-Alpine-Linux/releases)
Flash Utility:	
win32diskimager.rar	Windows utility. Under Linux users can use "dd"

The detailed steps are as follows:

- Get an 8G SDHC card and backup its data if necessary;
- Download and extract the xxx.img.gz and win32diskimager;
- Run the win32diskimager utility under Windows as administrator. On the utility's main window select your SD card's drive, the wanted image file and click on "write" to start flashing the SD card. Under Linux run "dd" to flash the rkXXXX-sd-OSNAME-YYYYMMDD.img file to your SD card;
- Take out the SD and insert it to NanoPi-R4S's microSD card slot;
- Power on NanoPi-R4S and it will be booted from your TF card;

5 Work with FriendlyWrt

5.1 Introduction to FriendlyWrt

FriendlyWrt is a customized system made by FriendlyElec based on an OpenWrt distribution. It is open source and well suitable for developing IoT applications, NAS applications etc.

5.2 First boot

For the first boot, the system needs to do the following initialization work :

- 1) Extended root file system
- 2) Initial setup (will execute /root/setup.sh)

So you need to wait for a while (about 2~3 minutes) to boot up for the first time, and then set FriendlyWrt, you can enter the ttyd terminal on the openwrt webpage, when the prompt is displayed as root@FriendlyWrt, it means the system has been initialized.

```
root@FriendlyWrt
```

5.3 Account & Password

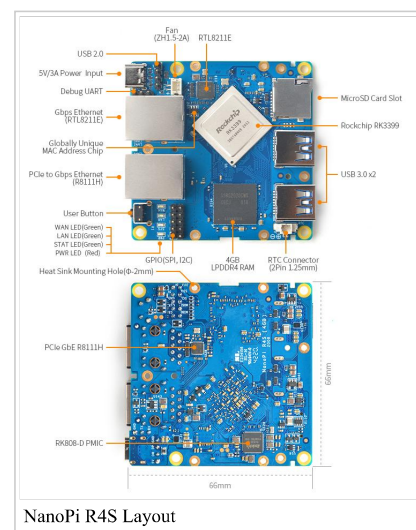
The default password is password (empty password in some versions). Please set or change a safer password for web login and ssh login. It is recommended to complete this setting before connecting NanoPi-R4S to the Internet.

5.4 Login FriendlyWrt

Connect the PC to the LAN port of NanoPi-R4S. If your PC without a built-in ethernet port, connect the LAN port of the wireless AP to the LAN port of NanoPi-R4S, and then connect your PC to the wireless AP via WiFi , Enter the following URL on your PC's browser to access the admin page:

- <http://friendlywrt/>
- <http://192.168.2.1/>
- [http://\[fd00:ab:cd::1\]](http://[fd00:ab:cd::1])

The above is the LAN port address of NanoPi-R4S. The IP address of the WAN port will be dynamically obtained from your main router through DHCP.



NanoPi R4S Layout

5.5 Recommended security settings

The following settings are highly recommended to complete before connecting NanoPi-R4S to the Internet.

- Set a secure password
- Only allow access to ssh from lan, change the port
- Check the firewall settings

Set up as you wish.

5.6 Change LAN IP in LuCI

- 1) Click on Network → Interfaces, then click on the Edit button of the LAN Network;
- 2) In General Setup tab, input new IP address (for example: 192.168.11.1), click "Save" and then click "Save & Apply";
- 3) On the pop-up window with the title “Connectivity change”, click "Apply and revert on connectivity loss";
- 4) Wait a moment, enter the new address in your computer's browser and login to FriendlyWrt;

5.7 Safe shutdown operation

Enter the "Services" -> "Terminal", enter the "poweroff" command and hit enter, wait until the led light is off, and then unplug the power supply.

5.8 Soft Factory Reset

Enter "System"->"Backup/Flash firmware", Click “Perform reset“ Button, Your device's settings will be reset to defaults like when FriendlyWrt was first installed. You can also do this in the terminal:

```
firstboot && reboot
```

5.9 Install Software Packages

5.9.1 Set up openwrt official opkg source

```
sed -i -e 's/mirrors.cloud.tencent.com/downloads.openwrt.org/g' /etc/opkg/distfeeds.conf
opkg update
```

5.9.2 Update Package List

Before install software packages update the package list:

```
$ opkg update
```

5.9.3 List Available Packages

```
$ opkg list
```

5.9.4 List Installed Packages

```
$ opkg list-installed
```

5.9.5 Install Packages

```
$ opkg install <package names>
```

5.9.6 Remove Packages

```
$ opkg remove <package names>
```

5.10 Disable IPv6

```
./root/setup.sh
disable_ipv6
reboot
```

5.11 Configure the function of the user button

By default, the user button is configured to reboot the device, as shown below:

```
echo 'BTN_1 1 /sbin/reboot' >> /etc/triggerhappy/triggers.d/example.conf
```

You can change its behavior by changing the configuration file above.

5.12 Configuring Quectel EC20 (4G module) dial-up networking

- Go to "Network" -> "Interfaces"

- Click "Delete" next to "WAN6", then click "Save & Apply"
- Click "Edit" next to "WAN", in the "Device" drop-down menu, select "Ethernet Adapter: wwan0", in the "Protocol" drop-down menu, select "QMI Cellular" and click "Switch Protocol"
- Click the "Modem Device" drop-down menu, select "/dev/cdc-wdm0", fill in the APN information (e.g. for China Mobile, enter "cmnet")
- Click "Save" to close the dialog. Finally, click "Save & Apply" at the bottom of the page to initiate the dial-up process
- Devices connected to LAN will have access to the Internet. If your device has a WiFi module, you can enable wireless AP functionality on the "Wireless" page and connect to the Internet via devices connected wirelessly

5.13 Some common issues of FriendlyWrt

- Unable to dial up
 - Go to "Network" -> "Firewall" and set "Inbound Data", "Outbound Data" and "Forwarding" in "WAN Zone" to "Accept";
 - If you still cannot access the Internet, you can try to turn off IPV6;
- Dial-up successful, but no outgoing traffic
 - Go to "Services" -> "Terminal" and type "fw4 reload" to try to reload the firewall settings again;
- Unable to power on
 - Try to replace the power adapter and cable. It is recommended to use a power supply with specifications above 5V/2A;
 - Note that some fast chargers with Type-C interface will have a delay, it may take a few seconds to start providing power;
- When doing secondary routing, the computer cannot connect to the Internet
 - If your main network is IPv4, and NanoPi-R4S works in IPv6, the computer may not be able to connect to the Internet. It is recommended to turn off IPv6 (the method is described later in this Wiki), or switch the main route to IPv6;
- If you have questions or have better suggestions, please send an email to techsupport@friendlyarm.com;

5.14 Use USB2LCD to view IP and temperature

Plug the USB2LCD module to the USB interface of NanoPi-R4S and power on, the IP address and CPU temperature will be displayed on the LCD:



5.15 How to Control Fan Speed for Cooling

(Note: The contents of this section are based on firmware released after 2021/08/31, kernel version kernel 5.10.xyz)

- The default behavior of the current PWM fan is: after a short wait (about 20 seconds) for power on, the fan will first work automatically for about 5 seconds, after which the behavior is driven by the kernel, which decides the fan on/off and the speed according to the CPU temperature.
- The behavior of the fan can be changed by modifying the following script: /usr/bin/fa-fancontrol.sh. For example, to change the CPU temperature when the fan starts working, you can change the following two lines:

```
echo 50000 > trip_point_3_temp # Indicates that the fan starts working at the lowest speed when the CPU temperature reaches 50 degrees
echo 55000 > trip_point_4_temp # Indicates that when the CPU temperature reaches 55 degrees, the fan rises to the second gear and above and automatically adjusts to the highest gear
```

- If you need to adjust the speed of each gear, you can modify the kernel dts file and recompile the kernel to achieve the purpose, the specific dts and modified location can be referred to the following commit: <https://github.com/friendlyarm/kernel-rockchip/commit/f74ac319f02e2d22cdd33227e7f167e4232809f9>

As shown below, the cooling-levels define 4 levels, with 0 being off and the highest level being 255:

```
fan: pwm-fan {
    compatible = "pwm-fan";
    /* FIXME: adjust leveles for the connected fan */
    cooling-levels = <0 12 18 255>;
    cooling-levels = <0 18 102 170 255>;
}
```

- If you are using kernel version 4.19.xyz, the fan is operated by PWM at the application level to achieve temperature control, the above content is not applicable, you need to modify this script:

```
/usr/bin/fa-fancontrol-direct.sh
```

5.16 How to use USB WiFi

5.16.1 Check USB WiFi Device with Command Line Utility

(1) Click on "services>ttyd" to start the command line utility

(2) Make sure no USB devices are connected to your board and run the following command to check if any USB devices are connected or not

```
lsusb
```

(3) Connect a USB WiFi device to the board and run the command again


```
lsusb
```

You will see a new device is detected. In our test the device's ID was 0BDA:C811

(4) Type your device's ID (in our case it was "0BDA:C811" or "VID_0BDA&PID_C811") in a search engine and you may find a device that matches the ID. In our case the device we got was Realtek 8811CU.

5.16.2 Configure a USB WiFi Device as AP

(1) Connect a USB WiFi device to the NanoPi-R4S. We recommend you to use the following devices:

WiFi Chipset \ OS	Distro Support		AP Mode
	FriendlyWrt OpenWrt 19.07.5	Ubuntu Core Ubuntu 20.04 64-bit	
RTL8188CUS/8188EU 802.11n WLAN Adapter	Preinstalled driver	Yes	✗
RT2070 Wireless Adapter	Preinstalled driver	Yes	✗
RT2870/RT3070 Wireless Adapter	Preinstalled driver	Yes	✗
RTL8192CU Wireless Adapter	Preinstalled driver	Yes	✗
Ralink MT7601/MT7601U	Preinstalled driver	Yes	✗
5G USB WIFI RTL8821CU/RTL8811CU (VID_0BDA & PID_C811)	Plug and play, Access Point mode by default	Yes	✓
5G USB WIFI RTL8812BU (VID_0BDA & PID_B812)	Plug and play, Access Point mode by default	Yes	✓
5G USB WiFi RTL8812AU (VID_0BDA & PID_8812)	Plug and play, Access Point mode by default	Yes	✓
 5G USB WiFi MediaTek MT7662 (VID_0E8D & PID_7612)	Plug and play, Access Point mode by default	No	✓

Note: devices that match these VID&PIDs would most likely work.

- (2) Click on "System>Reboot" and reboot your NanoPi-R4S
- (3) Click on "Network>Wireless" to enter the WiFi configuration page
- (4) Click on "Edit" to edit the configuration
- (5) On the "Interface Configuration" page you can set the WiFi mode and SSID, and then go to "Wireless Security" to change the password. By default the password is "password". After you make your changes click on "Save" to save
- (6) After you change the settings you can use a smartphone or PC to search for WiFi

5.16.3 Common USB WiFi issues

- 1) It is recommended to plug in the usb wifi in the off state, then power it on, FriendlyWrt will automatically generate the configuration file /etc/config/wireless, if not, see if there is wlan0 by ifconfig -a, if there is no wlan0, usually there is no driver.
- 2) If ifconfig -a sees wlan0, but the hotspot is not working properly, try changing the channel and country code, an inappropriate country code can also cause the WiFi to not work.
- 3) Some USB WiFis (e.g. MTK MT7662) work in CD-ROM mode by default and need to be switched by usb_modeswitch, you can try to add usb_modeswitch configuration to the following directory: /etc/usb_modeswitch.d.

5.16.4 Change the default WiFi hotspot configuration

FriendlyWrt sets the country, hotspot name and other parameters for USB WiFi by default, with the aim of being as plug-and-play as possible, but this does not guarantee that all modules will be compatible with this setting, you can change these behaviors by modifying the following file :

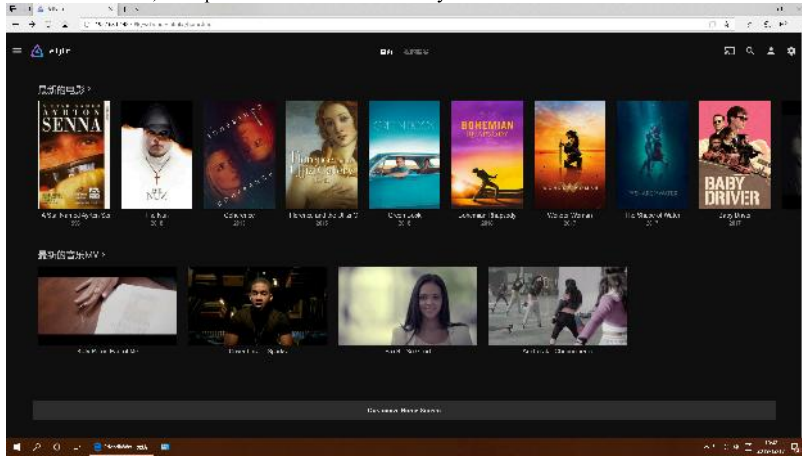
```
/lib/wifi/mac80211.sh
```

5.17 Work with Docker Applications

5.17.1 Work with Docker: Install JellyFin

```
mkdir -p /jellyfin/config
mkdir -p /jellyfin/videos
docker run --restart=always -d -p 8096:8096 -v /jellyfin/config:/config -v /jellyfin/videos:/videos jellyfin/jellyfin:10.1.0-arm64 --name myjellyfin
```

After installation, visit port 8096 and here is what you would find:



5.17.2 Work with Docker: Install Personal Nextcloud

```
mkdir /nextcloud -p
docker run -d -p 8888:80 --name nextcloud -v /nextcloud:/var/www/html/ --restart=always --privileged=true arm64v8/nextcloud
```

After installation, visit port 8888.

5.17.3 Expand Docker Storage

- Stop docker service first:

```
/etc/init.d/dockerd stop
```

- Rename the original /opt directory, create an empty /opt directory:

```
mv /opt /opt-old && mkdir /opt
```

- Format your drive as ext4, and mount it to the /opt directory:

A screenshot of the 'Mount Points - Mount Entry' dialog box. The 'Advanced Settings' tab is selected. The 'Enabled' checkbox is checked. The 'UUID' field contains the value 'c632a1d7-0898-45a4-9e7b-469bebbe9507 (/dev/nvme0n1p1, 238)'. The 'Mount point' dropdown menu is set to '/opt'. There are 'Dismiss' and 'Save' buttons at the bottom right.

- Enter the command "mount | grep /opt" to check the mount status:

```
root@FriendlyWrt:~# mount | grep /opt
/dev/nvme0n1p1 on /opt type ext4 (rw,relatime)
root@FriendlyWrt:~#
```

- Copy the files from the original /opt directory to the new /opt directory:

```
cp -af /opt-old/* /opt/ && rm -rf /opt-old
```

- Reboot the device

```
reboot
```

- After reboot, go to the "Docker" -> "Overview" page, check the information in the "Docker Root Dir" line, you can see that the Docker space has been expanded:

Docker - Overview

An overview with the relevant data is displayed here with which the LuCI docker client is connected.

Info	
Docker Version	20.10.12
Api Version	1.41
CPUs	4
Total Memory	1.91 GB
Docker Root Dir	/opt/docker (220.71 GB Available)
Index Server Address	https://index.docker.io/v1/

5.17.4 Docker FAQ and solutions

5.17.4.1 Unable to access the network services provided by the Docker container

Solution:

- Go to the "Firewall" settings and set "Forwarding" to "Accept";
- Turn off "Software Offload";

5.18 Mount smbfs

```
mount -t cifs //192.168.1.10/shared /movie -o username=xxx,password=yyy,file_mode=0644
```

5.19 Use sdk to compile the package

5.19.1 Install the compilation environment

Download and run the following script on 64-bit Ubuntu (version 18.04+): How to setup the Compiling Environment on Ubuntu bionic (<https://github.com/friendlyarm/build-env-on-ubuntu-bionic>)

5.19.2 Download and decompress sdk from the network disk

The sdk is located in the toolchain directory of the network disk:

```
tar xvf openwrt-sdk-*-rockchip-armv8_gcc-11.2.0_musl.Linux-x86_64.tar.xz
# If the path is too Long, it will cause some package compilation errors, so change the directory name here
mv openwrt-sdk-*-rockchip-armv8_gcc-11.2.0_musl.Linux-x86_64 sdk
cd sdk
./scripts/feeds update -a
./scripts/feeds install -a
```

5.19.3 Compile the package

download the source code of the example (a total of 3 examples are example1, example2, example3), and copy to the package directory:

```
git clone https://github.com/mwarning/openwrt-examples.git
cp -rf openwrt-examples/example* package/
rm -rf openwrt-examples/
```

Then enter the configuration menu through the following command:

```
make menuconfig
```

In the menu, select the following packages we want to compile (actually selected by default):

```
"Utilities" => "example1"
"Utilities" => "example3"
"Network" => "VPN" => "example2"
```

execute the following commands to compile the three software packages:

```
make package/example1/compile V=99
make package/example2/compile V=99
make package/example3/compile V=99
```

After the compilation is successful, you can find the ipk file in the bin directory, as shown below:

```
$ find ./bin -name example*.ipk
./bin/packages/aarch64_generic/base/example3_1.0.0-220420.38257_aarch64_generic.ipk
./bin/packages/aarch64_generic/base/example1_1.0.0-220420.38257_aarch64_generic.ipk
./bin/packages/aarch64_generic/base/example2_1.0.0-220420.38257_aarch64_generic.ipk
```

5.19.4 Install the ipk to NanoPi

You can use the scp command to upload the ipk file to NanoPi:

```
cd ./bin/packages/aarch64_generic/base/
scp example*.ipk root@192.168.2.1:/root/
```

Then use the opkg command to install them:

```
cd /root/
opkg install example3_1.0.0-220420.38257_aarch64_generic.ipk
opkg install example1_1.0.0-220420.38257_aarch64_generic.ipk
opkg install example2_1.0.0-220420.38257_aarch64_generic.ipk
```

5.20 Build FriendlyWrt using GitHub Actions

Please refre this link: <https://github.com/friendlyarm/Actions-FriendlyWrt>

6 Work with FriendlyCore

6.1 FriendlyCore User Account

- Non-root User:

```
User Name: pi
Password: pi
```

- Root:

```
User Name: root
Password: fa
```

6.2 Update Software Packages

```
$ sudo apt-get update
```

6.3 Setup Network Configurations

6.3.1 Set static IP address

By default "eth0" is assigned an IP address obtained via dhcp. If you want to change the setting you need to change the following file:

```
vi /etc/network/interfaces.d/eth0
```

For example if you want to assign a static IP to it you can run the following commands:

```
auto eth0
iface eth0 inet static
    address 192.168.1.231
    netmask 255.255.255.0
    gateway 192.168.1.1
```

6.3.2 Set a DNS

You also need to modify the following file to add the DNS configuration:

```
vi /etc/systemd/resolved.conf
```

For example, set to 192.168.1.1:

```
[Resolve]
DNS=192.168.1.1
```

Restart the systemd-resolved service with the following command:

```
sudo systemctl restart systemd-resolved.service
sudo systemctl enable systemd-resolved.service
```

6.3.3 Set up to use another network interface

To change the setting of "eth1" you can add a new file similar to eth0's configuration file under the /etc/network/interfaces.d/ directory.

6.4 Setup Wi-Fi

First, use the following command to check if Network-Manager is installed on your system:

```
which nmcli
```

If you have installed it, refer to this link to connect to WiFi: Use NetworkManager to configure network settings, If you do not have Network-Manager installed on your system, please refer to the following method to configure WiFi, By default the WiFi device is "wlan0". You need to create a configuration file under "/etc/network/interfaces.d/" for WiFi:

```
vi /etc/network/interfaces.d/wlan0
```

Here is a sample wlan0 file:

```
auto lo
iface lo inet loopback
auto wlan0
iface wlan0 inet dhcp
wpa-driver wext
wpa-ssid YourWiFiESSID
wpa-ap-scan 1
wpa-proto RSN
wpa-pairwise CCMP
wpa-group CCMP
wpa-key-mgmt WPA-PSK
wpa-psk YourWiFiPassword
```

Please replace "YourWiFiESSID" and "YourWiFiPassword" with your WiFiESSID and password. After save and close the file you can connect to your WiFi source by running the following command:

```
sudo systemctl daemon-reload
sudo systemctl restart networking
```

After you power on your board it will automatically connect to your WiFi source.

Please note that if you use one TF card to boot multiple boards the WiFi device name will likely be named to "wlan1", "wlan2" and etc. You can reset it to "wlan0" by deleting the contents of the following file and reboot your board: /etc/udev/rules.d/70-persistent-net.rules

6.4.1 WiFi models supported

6.4.1.1 M.2 WiFi Module

- RTL8822CE

6.4.1.2 Usb Dongle

- RTL8821CU (Vid: 0BDA, Pid: C811) (Test sample:TP-Link TL-WDN5200H)
- RTL8812AU (Vid: 0BDA, Pid: 8812)
- MediaTek MT7662 (Vid: 0E8D, Pid: 7612) (Test sample:COMFAST CF-WU782AC V2)

6.5 Install the kernel-header package

```
sudo dpkg -i /opt/archives/linux-headers-*.deb
```

6.6 Build kernel-header deb package

Please refre to: https://github.com/friendlyarm/sd-fuse_rk3399/blob/kernel-5.15.y/test/test-build-kernel-header-deb.sh

6.7 Config status LEDs

First determine whether the system already exists the leds initialization service:

```
sudo systemctl status leds
```

If the leds service already exists, change the default behavior of the LEDs by editing the following file:

```
/etc/init.d/leds.sh
```

Since there is no leds service in the early firmware, you need to refer to the following guide to manually configure the LEDs. First, set the following kernel modules to be automatically loaded at boot:

```
modprobe ledtrig-netdev
echo ledtrig-netdev > /etc/modules-load.d/ledtrig-netdev.conf
```

Put the following into the autorun script to associate the status leds with the ethernet interface, and you can configure it to behave in other ways by referring to these content:

```
echo netdev > /sys/class/leds/wan_led/trigger
echo eth0 > /sys/class/leds/wan_led/device_name
echo 1 > /sys/class/leds/wan_led/link

echo netdev > /sys/class/leds/lan_led/trigger
echo eth1 > /sys/class/leds/lan_led/device_name
echo 1 > /sys/class/leds/lan_led/link
```

6.8 Delete Qt5 and related files

Execute the following commands:

```

su root
cd /
rm -rf usr/local/Trolltech/Qt-5.10.0-rk64one usr/local/Trolltech/Qt-5.10.0-rk64one-sdk usr/bin/setqt5env* usr/bin/qt5demo etc/qt5
rm -rf opt/{qt5-browser,Qt5_CinematicExperience,qt5-multi-screen-demo,qt5-nmapper,qt5-player,qt5-smarthome,Qt5-Demo,qt5-qml-image-viewer,dual-camera}

```

6.9 Configure parameters for serial port

Use the following serial parameters:

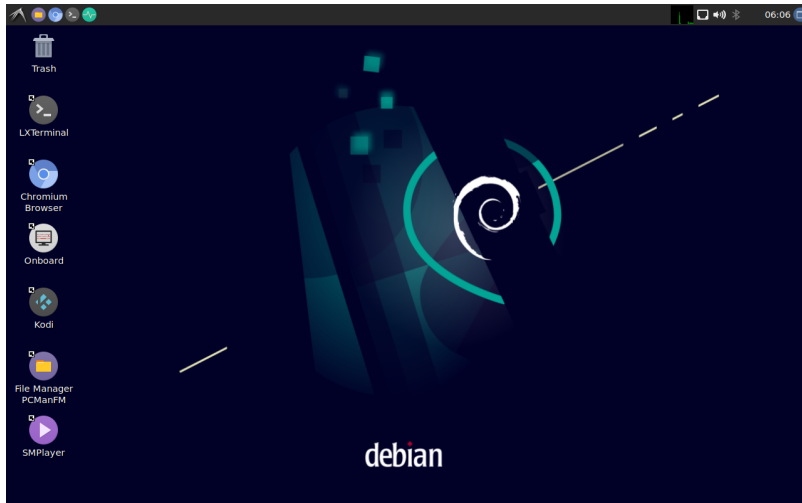
Baud rate	1500000
Data bit	8
Parity check	None
Stop bit	1
Flow control	None

7 Work with Debian11 Desktop

7.1 Introduction to Debian11 Desktop

Debian11 Desktop is a light-weighted debian desktop system, it has the following features:

- Uses LXDE as default desktop;
- Mali GPU-based OpenGL support;
- Support Rockchip MPP video hard coding and hard decoding;
- Pre-installed mpv and smplayer, both support 4K video hardware decoding;
- Pre-installed Chromium browser, support vpu/gpu hardware acceleration (video hard decoding limited to h264/mp4 format);
- Compatible with Plex Server and Docker;



7.2 Account & Password

Regular Account:

User Name: pi

Password: pi

Root:

the root user account is disabled by default, you may configure the root password through the 'sudo passwd root' command.

7.3 View IP address

Since the Debian Bullseye hostname is the hardware model by default, you can use the ping command to get the IP address: ping NanoPi-R4S

7.4 Connect to Debian via SSH

Run the following command: ssh pi@NanoPi-R4S

The default password is: pi

7.5 Update Software Packages

```

$ sudo apt-get update

```

7.6 Install x11vnc Server on Debian for Remote Access

7.6.1 Install x11vnc server

The following command to install x11vnc server:

```

sudo apt-get install x11vnc

```

```
sudo x11vnc -storepasswd /etc/x11vnc.pwd
```

```
sudo vi /lib/systemd/system/x11vnc.service
```

```
[Unit]
Description=Start x11vnc at startup.
Requires=display-manager.service
After=syslog.target network-online.target
Wants=syslog.target network-online.target

[Service]
Type=simple
ExecStart=/usr/bin/x11vnc -display :0 -forever -loop -noxdamage -repeat -rfbauth /etc/x11vnc.pwd -rfbport 5900 -shared -capslock -nomodtweak
ExecStop=/usr/bin/x11vnc -R stop
Restart=on-failure

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

```
sudo systemctl daemon-reload
sudo systemctl enable x11vnc.service
sudo systemctl start x11vnc
```

```
sudo dpkg -i /opt/archives/linux-headers-*.deb
```

```

$ sudo apt update
$ sudo apt install git gcc make bc
$ git clone https://github.com/RinCat/RTL88x2BU-Linux-Driver.git
$ cd RTL88x2BU-Linux-Driver
$ make -j$(nproc)
$ sudo make install
$ sudo modprobe 88x2bu

```

```
timedatectl
```

```
timedatectl list-timezones
```

```
sudo timedatectl set-timezone Asia/Shanghai
```

7.9 Change startup LOGO and Wallpaper

7.9.1 Change startup LOGO

Replace the following two files in the kernel source code directory and recompile the kernel :

kernel/logo.bmp

kernel/logo_kernel.bmp

Or use the script to operate, as shown below :

- Download scripts:

```
git clone https://github.com/friendlyarm/sd-fuse_rk3399.git -b kernel-4.19 --single-branch
cd sd-fuse_rk3399
```

- Compile kernel and repack firmware

```
convert files/logo.jpg -type truecolor /tmp/logo.bmp
convert files/logo.jpg -type truecolor /tmp/logo_kernel.bmp
sudo LOGO=/tmp/logo.bmp KERNEL_LOGO=/tmp/logo_kernel.bmp ./build-kernel.sh debian-bullseye-desktop-arm64
sudo ./mk-sd-image.sh debian-bullseye-desktop-arm64
sudo ./mk-emmc-image.sh debian-bullseye-desktop-arm64
```

Note: If your system is not debian-bullseye-desktop-arm64, please specify according to the actual situation

7.9.2 Change Wallpaper

Modify the following configuration file:

```
/home/pi/.config/pcmanfm/LXDE/desktop-items-0.conf
```

7.10 Soft Factory Reset

Execute the following command in a terminal:

```
sudo firstboot && sudo reboot
```

7.11 Start the program automatically at startup(For example Kodi)

Put the desktop file in the ~/.config/autostart/ directory, for example :

```
mkdir ~/.config/autostart/
cp /usr/share/applications/kodi.desktop ~/.config/autostart/
```

7.12 Disable auto-mounting

```
sudo systemctl mask udisks2
sudo reboot
```

7.13 Setup Chinese language and Input method

7.13.1 Setup Chinese language

Enter the following command and select 'zh_CN.UTF-8':

```
sudo dpkg-reconfigure locales
```

Add environment variables to .bashrc:

```
echo "export LC_ALL=zh_CN.UTF-8" >> ~/.bashrc
echo "export LANG=zh_CN.UTF-8" >> ~/.bashrc
echo "export LANGUAGE=zh_CN.UTF-8" >> ~/.bashrc
```

Reboot device:

```
sudo reboot
```

7.13.2 Installing Chinese input method

Enter the following command to install fcitx and Pinyin input method:

```
sudo apt update
sudo apt-get install fcitx fcitx-pinyin
sudo apt-get install im-config
sudo apt-get install fcitx-table*
sudo apt-get install fcitx-ui-classic fcitx-ui-light
sudo apt-get install fcitx-frontends-gtk2 fcitx-frontends-gtk3 fcitx-frontends-qt4
sudo apt-get remove --purge scim* ibus*
sudo reboot
```

After reboot, press Ctrl+Space to switch between Chinese and English input methods, and the input method icon will appear in the upper right corner, right-click the input method icon in the upper right corner to switch input methods in the pop-up menu, as shown below:



7.14 Installing Plex Multimedia Server

Visit the Plex website: <https://www.plex.tv/media-server-downloads/>
On the download page, select the category "Plex Media Server", choose "Linux" for the platform and "Ubuntu(16.04+)/Debian(8+) - ARMv8" for the version, After downloading the deb package, use the dpkg command to install the package:

```
sudo dpkg -i plexmediaserver_1.31.0.6654-02189b09f_arm64.deb
```

After installation, login to the Plex server by typing the following URL into your computer browser: <http://IP地址:32400/web/>

7.15 Install Docker on Debian

Please refer to: [How to Install Docker on Debian](#)

7.16 How to test NPU

Please refer to: [NPU](#)

8 Work with Debian10 Desktop

- Refer to:
 - Debian Buster

9 Buildroot Linux

Buildroot is a simple, efficient and easy-to-use tool to generate embedded Linux systems through cross-compilation. It contains a boot-loader, kernel, rootfs, various libraries and utilities(e.g. qt, gstreamer, busybox etc).

FriendlyELEC's Buildroot is based on Rockchip's version which is made with linux-sdk and maintained with git. FriendlyELEC's version is synced with Rockchip's version;

- Rockchip's Buildroot: <https://github.com/rockchip-linux/buildroot>
- Buildroot's official site: <https://buildroot.org>

For a more detailed description of the Buildroot system, please refer to: [Buildroot](#)

10 How to Compile

10.1 Setup Development Environment

10.1.1 Method 1: Using docker to cross-compile

Please refre to [docker-cross-compiler-novnc \(https://github.com/friendlyarm/docker-cross-compiler-novnc\)](https://github.com/friendlyarm/docker-cross-compiler-novnc)

10.1.2 Method 2: Setup build environment on the host machine

10.1.2.1 Install required packages

Install and run requirements ubuntu 20.04, install required packages using the following commands:

```
sudo apt-get -y update
sudo apt-get install -y sudo curl
sudo bash -c \"
$(curl -fsSL https://raw.githubusercontent.com/friendlyarm/build-env-on-ubuntu-bionic/master/install.sh)\"
```

The following cross-compilers will be installed:

Version	Architecture	Compiler path	Purpose
4.9.3	armhf	/opt/FriendlyARM/toolchain/4.9.3	Can be used to build 32-bit ARM applications
6.4	aarch64	/opt/FriendlyARM/toolchain/6.4-aarch64	Can be used to build kernel 4.4
11.3	aarch64	/opt/FriendlyARM/toolchain/11.3-aarch64	Can be used to build kernel 4.19 or higher and U-Boot

10.1.2.2 Setting the compiler path

Based on the table in the previous section, select the appropriate version of the compiler and add the compiler's path to PATH. For example, if you want to use the 11.3 cross-compiler, edit ~/.bashrc using vi and add the following content to the end:

```
export PATH=/opt/FriendlyARM/toolchain/11.3-aarch64/bin:$PATH
export GCC_COLORS=auto
```

Run the ~/.bashrc script to make it effective in the current commandline. Note: there is a space after ".":

```
./ ./.bashrc
```

To verify if the installation was successful:

```
$ aarch64-linux-gcc -v
Using built-in specs.
COLLECT_GCC=aarch64-linux-gcc
COLLECT_LTO_WRAPPER=/opt/FriendlyARM/toolchain/11.3-aarch64/libexec/gcc/aarch64-cortexa53-linux-gnu/11.3.0/lto-wrapper
Target: aarch64-cortexa53-linux-gnu
Configured with: /home/cross/arm64/src/gcc/configure --build=x86_64-build_pc-linux-gnu --host=x86_64-build_pc-linux-gnu --target=aarch64-cortexa53-linux-gnu --prefix=/opt/FriendlyARM/t
Thread model: posix
Supported LTO compression algorithms: zlib
gcc version 11.3.0 (ctng-1.25.0-119g-FA)
```

10.2 Build Openwrt/Friendlywrt

10.2.1 Download Code

Two versions are available, please choose as required:

10.2.1.1 FriendlyWrt 24.10

```
mkdir friendlywrt24-rk3399
cd friendlywrt24-rk3399
git clone https://github.com/friendlyarm/repo --depth 1 tools
tools/repo init -u https://github.com/friendlyarm/friendlywrt_manifests -b master-v24.10 \
-m rk3399.xml --repo-url=https://github.com/friendlyarm/repo --no-clone-bundle
tools/repo sync -c --no-clone-bundle
```

10.2.1.2 FriendlyWrt 23.05

```
mkdir friendlywrt23-rk3399
cd friendlywrt23-rk3399
git clone https://github.com/friendlyarm/repo --depth 1 tools
tools/repo init -u https://github.com/friendlyarm/friendlywrt_manifests -b master-v23.05 \
-m rk3399.xml --repo-url=https://github.com/friendlyarm/repo --no-clone-bundle
tools/repo sync -c --no-clone-bundle
```

10.2.2 First compilation step

```
./build.sh rk3399.mk # or rk3399-docker.mk
```

All the components (including u-boot, kernel, and friendlywrt) are compiled and the sd card image will be generated, then execute the following command to generate the image file for installing the system into the emmc:

```
./build.sh emmc-img
```

After making changes to the project, the sd card image needs to be repackaged by running the following command:

```
./build.sh sd-img
```

10.2.3 Secondary compilation steps

```
cd friendlywrt
make menuconfig
rm -rf ./tmp
make -j$(nproc)
cd ../
./build.sh sd-img
./build.sh emmc-img
```

10.2.4 Build u-boot only

```
./build.sh uboot
```


10.2.5 Build kernel only

```
./build.sh kernel
```

10.2.6 Build friendlywrt only

```
./build.sh friendlywrt
```

Or go to the friendlywrt directory and follow the standard openwrt commands. If you get an error with the above command, try using the following command to compile in a single thread:

```
cd friendlywrt
make -j1 V=s
```

10.3 Build Buildroot

please refer to: Buildroot

10.4 Build Other Linux

10.4.1 Kernel and u-boot versions

Operating System	Kernel Version	U-boot version	Cross-compiler	Partition type	Packaging Tool	Kernel branch
lubuntu	linux v4.4.y	u-boot v2014.10	6.4-aarch64	MBR (https://github.com/friendlyarm/sd-fuse_rk3399/blob/master/prebuilt/parameter.template)	sd-fuse (https://github.com/friendlyarm/sd-fuse_rk3399/tree/master)	nanopi4-linux-v4.4.y (https://github.com/friendlyrockchip/tree/nanopi4-linux-v4.4.y)
friendlycore-arm64						
friendlydesktop-arm64						
eflasher						
buildroot	linux v4.19.y	u-boot v2017.09	11.3-aarch64	GPT (https://github.com/friendlyarm/sd-fuse_rk3399/blob/kernel-4.19/prebuilt/parameter.template)	sd-fuse (https://github.com/friendlyarm/sd-fuse_rk3399/tree/kernel-4.19)	nanopi4-v4.19.y (https://github.com/friendlyrockchip/tree/nanopi4-v4.19.y)
ubuntu-focal-desktop-arm64						
debian-bullseye-desktop-arm64						
debian-bullseye-minimal-arm64						
friendlycore-focal-arm64						
debian-bookworm-core-arm64						
ubuntu-noble-core-arm64						
openmediavault-arm64	linux v6.1.y	u-boot v2017.09	11.3-aarch64	GPT (https://github.com/friendlyarm/sd-fuse_rk3399/blob/kernel-6.1.y/prebuilt/parameter-ext4.txt)	sd-fuse (https://github.com/friendlyarm/sd-fuse_rk3399/tree/kernel-6.1.y)	nanopi-r2-v6.1.y (https://github.com/friendlyrockchip/tree/nanopi-r2-v6.1.y)
friendlywrt21				GPT (https://github.com/friendlyarm/sd-fuse_rk3399/blob/kernel-6.1.y/prebuilt/parameter.txt)		
friendlywrt21-docker						
friendlywrt23						
friendlywrt23-docker						

- Kernel git repo: <https://github.com/friendlyarm/kernel-rockchip>
- U-boot git repo: <https://github.com/friendlyarm/u-boot-rockchip>
- The cross-compile toolchain is located in the path: /opt/FriendlyARM/toolchain/
- The SD-Fuse is a helper script to make bootable SD card image.
- Click on MBR and GPT in the table to view the partition layout (configuration file) for each system.

10.4.2 Build kernel linux-v4.4.y

This section applies to the following operating systems:

lubuntu	eflasher	friendlydesktop-arm64	friendlycore-arm64
---------	----------	-----------------------	--------------------

Clone the repository to your local drive then build:

```
git clone https://github.com/friendlyarm/kernel-rockchip --single-branch --depth 1 -b nanopi4-linux-v4.4.y kernel-rockchip
cd kernel-rockchip
export PATH=/opt/FriendlyARM/toolchain/6.4-aarch64/bin/:$PATH
touch .scmversion
# Load configuration
make ARCH=arm64 CROSS_COMPILE=aarch64-linux- nanopi4_linux_defconfig
# Optionally, if you want to change the default kernel config
# make ARCH=arm64 CROSS_COMPILE=aarch64-linux- menuconfig
# Start building kernel
```

```
make ARCH=arm64 CROSS_COMPILE=aarch64-linux- nanopi4-images -j$(nproc)
# Start building kernel modules
mkdir -p out-modules
make ARCH=arm64 CROSS_COMPILE=aarch64-linux- INSTALL_MOD_PATH="$PWD/out-modules" modules -j$(nproc)
make ARCH=arm64 CROSS_COMPILE=aarch64-linux- INSTALL_MOD_PATH="$PWD/out-modules" modules_install
KERNEL_VER=$(make CROSS_COMPILE=aarch64-linux-gnu- ARCH=arm64 kernelrelease)
rm -rf $PWD/out-modules/lib/modules/${KERNEL_VER}/kernel/drivers/gpu/arm/mali400/
[ ! -f "$PWD/out-modules/lib/modules/${KERNEL_VER}/modules.dep" ] && depmod -b $PWD/out-modules -E Module.symvers -F System.map -w ${KERNEL_VER}
(cd $PWD/out-modules && find . -name '*.ko' | xargs aarch64-linux-strip --strip-unneeded)
```

After the compilation, the following files will be generated:

kernel.img	resource.img	The kernel modules are located in the out-modules directory
------------	--------------	---

Run your build:
Please refre to #Running the build

10.4.3 Build u-boot v2014.10

This section applies to the following operating systems:

lubuntu	eflasher	friendlydesktop-arm64	friendlycore-arm64
---------	----------	-----------------------	--------------------

Clone the repository to your local drive then build:

```
git clone https://github.com/friendlyarm/uboot-rockchip --single-branch --depth 1 -b nanopi4-v2014.10_oreo
cd uboot-rockchip
export PATH=/opt/FriendlyARM/toolchain/6.4-aarch64/bin/:$PATH
make CROSS_COMPILE=aarch64-linux- rk3399_defconfig
make CROSS_COMPILE=aarch64-linux-
```

After the compilation, the following files will be generated:

uboot.img	trust.img	rk3399_loader_v1.22.119.bin (aka MiniLoaderAll.bin)
-----------	-----------	---

Installing the u-boot:
Please refre to #Running the build

10.4.4 Build kernel linux-v4.19.y

This section applies to the following operating systems:

ubuntu-focal-desktop-arm64	debian-bullseye-desktop-arm64	debian-bullseye-minimal-arm64	friendlycore-focal-arm64	ubuntu-noble-core-arm64	debian-bookworm-core-arm64	buildroot
----------------------------	-------------------------------	-------------------------------	--------------------------	-------------------------	----------------------------	-----------

Clone the repository to your local drive then build:

```
git clone https://github.com/friendlyarm/kernel-rockchip --single-branch --depth 1 -b nanopi4-v4.19.y kernel-rockchip
cd kernel-rockchip
export PATH=/opt/FriendlyARM/toolchain/11.3-aarch64/bin/:$PATH
touch .scmversion
# Configuring the Kernel
# Load default configuration
make ARCH=arm64 CROSS_COMPILE=aarch64-linux- nanopi4_linux_defconfig
# Optionally, Load configuration for FriendlyWrt
# make ARCH=arm64 CROSS_COMPILE=aarch64-linux- nanopi4_linux_defconfig friendlywrt.config
# Optionally, if you want to change the default kernel config
# make ARCH=arm64 CROSS_COMPILE=aarch64-linux- menuconfig
# Start building kernel
make ARCH=arm64 CROSS_COMPILE=aarch64-linux- nanopi4-images -j$(nproc)
# Start building kernel modules
mkdir -p out-modules
make ARCH=arm64 CROSS_COMPILE=aarch64-linux- INSTALL_MOD_PATH="$PWD/out-modules" modules -j$(nproc)
make ARCH=arm64 CROSS_COMPILE=aarch64-linux- INSTALL_MOD_PATH="$PWD/out-modules" modules_install
KERNEL_VER=$(make CROSS_COMPILE=aarch64-linux-gnu- ARCH=arm64 kernelrelease)
rm -rf $PWD/out-modules/lib/modules/${KERNEL_VER}/kernel/drivers/gpu/arm/mali400/
[ ! -f "$PWD/out-modules/lib/modules/${KERNEL_VER}/modules.dep" ] && depmod -b $PWD/out-modules -E Module.symvers -F System.map -w ${KERNEL_VER}
(cd $PWD/out-modules && find . -name '*.ko' | xargs aarch64-linux-strip --strip-unneeded)
```

After the compilation, the following files will be generated:

kernel.img	resource.img	The kernel modules are located in the out-modules directory
------------	--------------	---

Run your build:
Please refre to #Running the build

10.4.5 Build kernel linux-v6.1.y

This section applies to the following operating systems:

friendlywrt21	friendlywrt21-docker	friendlywrt23	friendlywrt23-docker	openmediavault-arm64
---------------	----------------------	---------------	----------------------	----------------------

Clone the repository to your local drive then build:

```
git clone https://github.com/friendlyarm/kernel-rockchip --single-branch --depth 1 -b nanopi-r2-v6.1.y kernel-rockchip
cd kernel-rockchip
export PATH=/opt/FriendlyARM/toolchain/11.3-aarch64/bin/:$PATH
touch .scmversion
# Configuring the Kernel
# Load default configuration
make CROSS_COMPILE=aarch64-linux-gnu- ARCH=arm64 nanopi4_linux_defconfig
# Optionally, Load configuration for FriendlyWrt
```

```
# make CROSS_COMPILE=aarch64-linux-gnu- ARCH=arm64 nanopi4_linux_defconfig friendlywrt.config
# Optionally, if you want to change the default kernel config
# make CROSS_COMPILE=aarch64-linux-gnu- ARCH=arm64 menuconfig
# Start building kernel
make CROSS_COMPILE=aarch64-linux-gnu- ARCH=arm64 -j$(nproc)
# Start building kernel modules
mkdir -p out-modules && rm -rf out-modules/*
make CROSS_COMPILE=aarch64-linux-gnu- ARCH=arm64 INSTALL_MOD_PATH="$PWD/out-modules" modules -j$(nproc)
make CROSS_COMPILE=aarch64-linux-gnu- ARCH=arm64 INSTALL_MOD_PATH="$PWD/out-modules" modules_install
KERNEL_VER=$(make CROSS_COMPILE=aarch64-linux-gnu- ARCH=arm64 kernelrelease)
[ ! -f "$PWD/out-modules/lib/modules/${KERNEL_VER}/modules.dep" ] && depmod -b $PWD/out-modules -E Module.symvers -F System.map -w ${KERNEL_VER}
(cd $PWD/out-modules && find . -name \*.ko | xargs aarch64-linux-strip --strip-unneeded)
```

Pack the kernel.img and resource.img:

```
wget https://raw.githubusercontent.com/friendlyarm/sd-fuse_rk3399/kernel-6.1.y/tools/mkkrnlimg && chmod 755 mkkrnlimg
wget https://raw.githubusercontent.com/friendlyarm/sd-fuse_rk3399/kernel-6.1.y/tools/resource_tool && chmod 755 resource_tool
wget https://raw.githubusercontent.com/friendlyarm/sd-fuse_rk3399/kernel-6.1.y/prebuilt/boot/logo.bmp
wget https://raw.githubusercontent.com/friendlyarm/sd-fuse_rk3399/kernel-6.1.y/prebuilt/boot/logo_kernel.bmp
./mkkrnlimg arch/arm64/boot/Image kernel.img
mkdir kernel-dtbs
cp -f arch/arm64/boot/dts/rockchip/rk3399-nanopi-r4s.dtb kernel-dtbs/rk3399-nanopi4-rev09.dtb
cp -f arch/arm64/boot/dts/rockchip/rk3399-nanopi-r4s.dtb kernel-dtbs/rk3399-nanopi4-rev0a.dtb
cp -f arch/arm64/boot/dts/rockchip/rk3399-nanopi-r4se.dtb kernel-dtbs/rk3399-nanopi4-rev0b.dtb
cp -f arch/arm64/boot/dts/rockchip/rk3399-nanopc-t4.dtb kernel-dtbs/rk3399-nanopi4-rev00.dtb
./resource_tool --dtbname kernel-dtbs/*.dtb logo.bmp logo_kernel.bmp
```

After the compilation, the following files will be generated:

kernel.img	resource.img	The kernel modules are located in the out-modules directory
------------	--------------	---

Run your build:
Please refre to #Running the build

10.4.6 Build u-boot v2017.09

This section applies to the following operating systems:

ubuntu-focal-desktop-arm64	debian-bullseye-desktop-arm64	debian-bullseye-minimal-arm64	friendlycore-focal-arm64	ubuntu-noble-core-arm64	debian-bookworm-core-arm64	buildroot
----------------------------	-------------------------------	-------------------------------	--------------------------	-------------------------	----------------------------	-----------

Clone the repository to your local drive then build:

```
git clone https://github.com/friendlyarm/rkbin --single-branch --depth 1 -b friendlyelec
git clone https://github.com/friendlyarm/uboot-rockchip --single-branch --depth 1 -b nanopi4-v2017.09
export PATH=/opt/FriendlyARM/toolchain/11.3-aarch64/bin/:$PATH
cd uboot-rockchip/
./make.sh nanopi4
```

After the compilation, the following files will be generated:

uboot.img	trust.img	rk3399_loader_v1.24.126.bin (aka MiniLoaderAll.bin)
-----------	-----------	---

Run your build:
Please refre to #Running the build

10.4.7 Running the build

10.4.7.1 Install to target board

10.4.7.1.1 MBR partition

This section applies to the following operating systems:

lubuntu	cflasher	friendlydesktop-arm64	friendlycore-arm64
---------	----------	-----------------------	--------------------

The MBR partitioning is only used by the Linux v4.4 kernel. You can check the partition layout by clicking on this link: partmap (https://github.com/friendlyarm/sd-fuse_rk3399/blob/master/prebuilt/parameter.template). To write an image file, you can use the dd command. For example, in the parameter.template file, "0x00014000@0x00014000(kernel)" specifies that the kernel partition starts at 0x00014000, which is equivalent to 81920 in decimal. Therefore, the dd command should be as follows:

```
dd if=kernel.img of=/dev/mmcblk0 seek=81920
```

10.4.7.1.2 GPT partition

This section applies to the following operating systems:

ubuntu-focal-desktop-arm64	debian-bookworm-core-arm64	debian-bullseye-desktop-arm64	debian-bullseye-minimal-arm64
friendlycore-focal-arm64	ubuntu-noble-core-arm64	friendlywrt21-kernel4	buildroot
friendlywrt21	friendlywrt21-docker	friendlywrt23	friendlywrt23-docker

The OS uses GPT partitions by default which is using the Linux v4.19 and Linux v5.15 kernel, you can use the dd command, but be careful to choose the right output device:

- The SD/TF Card device node: /dev/mmcblk0
- The eMMC device node: /dev/mmcblk2

The following is an example of how to update the kernel to eMMC:
Use the 'parted' command to view the partition layout:

```
parted /dev/mmcblk2 print
```

Sample outputs:

```
Model: MMC BJTDA4R (sd/mmc)
Disk /dev/mmcblk2: 31.3GB
Sector size (logical/physical): 512B/512B
Partition Table: gpt
Disk Flags:

Number  Start   End     Size    File system  Name      Flags
  1      8389kB  12.6MB  4194kB                uboot
  2      12.6MB  16.8MB  4194kB                trust
  3      16.8MB  21.0MB  4194kB                misc
  4      21.0MB  25.2MB  4194kB                dtbo
  5      25.2MB  41.9MB  16.8MB                resource
  6      41.9MB  83.9MB  41.9MB                kernel
  7      83.9MB  134MB   50.3MB                boot
  8     134MB  2500MB  2366MB  ext4          rootfs
  9     2500MB  31.3GB  28.8GB  ext4          userdata
```

as shown above, the resource partition is located at 5 and the kernel partition is located at 6. Use the dd command to write the resource.img and kernel.img files to these partitions, the commands are as follows:

```
dd if=resource.img of=/dev/mmcblk2p5 bs=1M
dd if=kernel.img of=/dev/mmcblk2p6 bs=1M
```

If you want to update u-boot:

```
dd if=uboot.img of=/dev/mmcblk2p1 bs=1M
```

To update new driver modules, copy the newly compiled driver modules to the appropriate directory under /lib/modules.

10.4.7.2 Packaging and creating an SD image

To create a new OS image file, you need to use the "sd-fuse" packaging tool.

"sd-fuse" is a collection of scripts that can be used to create bootable SD card images for FriendlyElec boards. Its main features include:

- Creation of root filesystem images from a directory
- Building of bootable SD card images
- Simple compilation of kernel, U-Boot, and third-party drivers

Please click on the following link to find out more:

Kernel version	Packaging Tool
linux v4.4.y	sd-fuse (https://github.com/friendlyarm/sd-fuse_rk3399/tree/master)
linux v4.19.y	sd-fuse_rk3399/kernel-4.19 (https://github.com/friendlyarm/sd-fuse_rk3399/tree/kernel-4.19)
linux v6.1.y	sd-fuse_rk3399/kernel-6.1.y (https://github.com/friendlyarm/sd-fuse_rk3399/tree/kernel-6.1.y)

10.4.7.3 USB flashing

Note: kernel v4.4.y is not supported

10.4.7.3.1 Linux

Reboot the board and enter loader mode with the following command:

```
sudo reboot loader
```

To flash U-Boot and kernel using the "upgrade_tool_v2.17_for_linux" tool, please use the following command:

```
sudo upgrade_tool di -k kernel.img
sudo upgrade_tool di -re resource.img
sudo upgrade_tool di -u uboot.img
sudo upgrade_tool RD
```

Note: "upgrade_tool" is a command-line tool provided by Rockchip for Linux operating systems (Linux_Upgrade_Tool).

10.5 Build the code using scripts

10.5.1 Download scripts and image files

```
git clone https://github.com/friendlyarm/sd-fuse_rk3399.git -b kernel-4.19
cd sd-fuse_rk3399
wget http://112.124.9.243/dvdfiles/RK3399/images-for-eflasher/friendlycore-focal-arm64-images.tgz
tar xvfz friendlycore-focal-arm64-images.tgz
```

10.5.2 Compile the kernel

Download the kernel source code and compile it. the relevant image files in the friendlycore-focal-arm64 directory will be automatically updated, including the kernel modules in the file system:

```
git clone https://github.com/friendlyarm/kernel-rockchip --depth 1 -b nanopi4-v4.19.y kernel-rk3399
KERNEL_SRC=$PWD/kernel-rk3399 ./build-kernel.sh friendlycore-focal-arm64
```

10.5.3 Compile the kernel headers

```
git clone https://github.com/friendlyarm/kernel-rockchip --depth 1 -b nanopi4-v4.19.y kernel-rk3399
MK_HEADERS_DEB=1 BUILD_THIRD_PARTY_DRIVER=0 KERNEL_SRC=$PWD/kernel-rk3399 ./build-kernel.sh friendlycore-focal-arm64
```

10.5.4 Compile the uboot

Download the uboot source code and compile it. the relevant image files in the friendlycore-focal-arm64 directory will be automatically updated:

```
git clone https://github.com/friendlyarm/uboot-rockchip --depth 1 -b nanopi4-v2017.09
UBOOT_SRC=$PWD/uboot-rockchip ./build-uboot.sh friendlycore-focal-arm64
```

10.5.5 Generate new image

Repackage the image file in the friendlycore-focal-arm64 directory into sd card image:

```
./mk-sd-image.sh friendlycore-focal-arm64
```

After the command is completed, the image is in the out directory, you can use the dd command to make the SD boot card, for example:

```
dd if=out/rk3399-sd-friendlycore-focal-4.19-arm64-YYYYMMDD.img of=/dev/sdX bs=1M
```

10.6 Building AOSP from source

10.6.1 Hardware and Software Requirements

- Your computer should have at least 16GB of RAM and 300GB of disk space. We recommend using a machine with 32GB of RAM and a large-capacity, high-speed SSD, and we do not recommend using virtual machines.
- If you encounter compilation errors, they may be caused by problems with the compilation environment. We recommend using the following Docker container for compilation: docker-cross-compiler-novnc (<https://github.com/friendlyarm/docker-cross-compiler-novnc>).

10.6.2 Compile Android10

10.6.2.1 Download Android10 Source Code

There are two ways to download the source code:

- **repo archive file on netdisk**

Netdisk URL: Click here (<http://download.friendlyelec.com/NanoPiR4S>)

File location on netdisk: "07_Source codes/rk3399-android-10.git-YYYYMMDD.tar.xz" (YYYYMMDD means the date of packaging)

After extracting the repo package from the network disk, you need to execute the sync.sh script, which will pull the latest code from gitlab:

```
tar xf "/path/to/netdisk/07_Source codes/rk3399-android-10.git-YYYYMMDD.tar.xz"
cd rk3399-android-10
./sync.sh
```

- **git clone from gitlab**

NanoPi-R4S source code is maintained in gitlab, You can download it by running the following command:

```
git clone --recursive https://gitlab.com/friendlyelec/rk3399-android-10.git -b main
```

Note: If the following error "error: unknown option `recurse-submodules'" appears, please upgrade git to v2.0.0 or above.

10.6.2.2 Generate Image File

You can compile an Android source code and generate an image file (non-root user is recommended):

```
cd rk3399-android-10
./build-nanopc-t4.sh -F -M
```

If you need to include google apps, you need to set an environment variable and then compile, as shown below:

```
cd rk3399-android-10
export INSTALL_GAPPS_FOR_TESTING=yes
./build-nanopc-t4.sh -F -M
```

10.6.2.3 Make OTA Packages

If you need the support of A/B (Seamless) System Updates, you need to do the following:

- Build your own update server for http download of update files;
- Customize the Updater application, the code is located in packages/apps/Updater, let it connect and download file from your server;
- Use the quick compilation script parameter -O or --ota to compile OTA Packages, as shown below:

```
cd rk3399-android-10
./build-nanopc-t4.sh -F -O -M
```

After the compilation is successfully completed, the OTA update related packages are located in the directory: rockdev/otapackage/, Please do not delete this directory.

After you have made some changes, compiling again with the parameter -O will generate ota-update-XXXXXXX.zip, which is an incremental update package.

OTA Packages decides whether to generate incremental update package according to BUILD_NUMBER, for details, please refer to build-nanopc-t4.sh.

To disable the A/B feature, you can refer to the following to modify device/rockchip/rk3399/nanopc-t4/BoardConfig.mk, and then recompile uboot and android:

```
BOARD_USES_AB_IMAGE := false
```

10.6.2.4 Update System with New Image

After compilation is done a new image file will be generated in the "rockdev/Image-nanopc_t4/" directory under Android 10's source code directory. You can follow the steps below to update the OS in NanoPi-R4S:

1) Insert an SD card which is processed with EFlasher to an SD card reader and insert this reader to a PC running Ubuntu. The SD card's partitions will be automatically mounted;

2) Copy all the files under the "rockdev/Image-nanopc_t4/" directory to the SD card's android10 directory in the "FRIENDLYARM" partition;

3) Insert this SD card to NanoPi-R4S and reflash Android

When flashing Android 10, EFlasher requires v1.3 or above. When flashing with Type-C, please use the tool AndroidTool v2.71 or Linux_Upgrade_Tool v1.49 provided by Rockchip.

10.6.3 Compile Android8.1

10.6.3.1 Download Android8.1 Source Code

There are two ways to download the source code:

■ repo archive file on netdisk

Netdisk URL: Click here (<http://download.friendlyelec.com/NanoPiR4S>)

File location on netdisk: sources/rk3399-android-8.1.git-YYYYMMDD.tgz (YYYYMMDD means the date of packaging)

After extracting the repo package from the network disk, you need to execute the sync.sh script, which will pull the latest code from gitlab:

```
tar xvf /path/to/netdisk/sources/rk3399-android-8.1.git-YYYYMMDD.tgz
cd rk3399-android-8.1
./sync.sh
```

■ git clone from gitlab

NanoPi-R4S source code is maintained in gitlab, You can download it by running the following command:

```
git clone https://gitlab.com/friendlyelec/rk3399-android-8.1 --depth 1 -b master
```

10.6.3.2 Generate Image File

You can compile an Android source code and generate an image file:

```
cd rk3399-android-8.1
./build-nanopc-t4.sh -F -M
```

10.6.3.3 Update System with New Image

After compilation is done a new image file will be generated in the "rockdev/Image-nanopc_t4/" directory under Android 8.1's source code directory. You can follow the steps below to update the OS in NanoPi-R4S:

1) Insert an SD card which is processed with EFlasher to an SD card reader and insert this reader to a PC running Ubuntu. The SD card's partitions will be automatically mounted;

2) Copy all the files under the "rockdev/Image-nanopc_t4/" directory to the SD card's android8 directory in the "FRIENDLYARM" partition;

3) Insert this SD card to NanoPi-R4S and reflash Android

Here is an alternative guide to update OS: sd-fuse_rk3399 (https://github.com/friendlyarm/sd-fuse_rk3399)

10.6.4 Compile Android7

10.6.4.1 Download Android7 Source Code

There are two ways to download the source code:

■ repo archive file on netdisk

Netdisk URL: Click here (<http://download.friendlyelec.com/NanoPiR4S>)
File location on netdisk: sources/rk3399-android-7.git-YYYYMMDD.tgz (YYYYMMDD means the date of packaging)
After extracting the repo package from the network disk, you need to execute the sync.sh script, which will pull the latest code from gitlab:

```
tar xvf /path/to/netdisk/sources/rk3399-android-7.git-YYYYMMDD.tgz
cd rk3399-nougat
./sync.sh
```

■ git clone from gitlab

NanoPi-R4S source code is maintained in gitlab, You can download it by running the following command:

```
git clone https://gitlab.com/friendlyelec/rk3399-nougat --depth 1 -b nanopc-t4-nougat
```

10.6.4.2 Generate Image File

You can compile an Android7 source code and generate an image file:

```
cd rk3399-nougat
./build-nanopc-t4.sh -F -M
```

10.6.4.3 Update System with New Image

After compilation is done a new image file will be generated in the "rockdev/Image-nanopc_t4/" directory under Android7's source code directory. You can follow the steps below to update the OS in NanoPi-R4S:
1) Insert an SD card which is processed with EFlasher to an SD card reader and insert this reader to a PC running Ubuntu. The SD card's partitions will be automatically mounted;
2) Copy all the files under the "rockdev/Image-nanopc_t4/" directory to the SD card's android8 directory in the "FRIENDLYARM" partition;
3) Insert this SD card to NanoPi-R4S and reflash Android
Here is an alternative guide to update OS: sd-fuse_rk3399 (https://github.com/friendlyarm/sd-fuse_rk3399)

11 Using On-Board Hardware Resources

11.1 Access Serial Interface

For now only UART4 is available for users:

Serial Interface	Serial Device
UART0	Used by Bluetooth
UART1	Used by Gbps Ethernet
UART2	Used by Serial Debug Port
UART3	Used by Gbps Ethernet
UART4	Available, device name is /dev/ttyS4 (note: this is only applicable for ROM released after 20180618)

11.2 DTS files

Please refer to DTS files

12 Backup rootfs and create custom SD image (to burn your application into other boards)

12.1 Backup rootfs

Run the following commands on your target board. These commands will back up the entire root partition:

```
sudo passwd root
su root
cd /
tar --warning=no-file-changed -cvzf /rootfs.tar.gz \
--exclude=/rootfs.tar.gz --exclude=/var/lib/docker/runtimes \
--exclude=/etc/firstuser --exclude=/etc/friendlyelec-release \
--exclude=/usr/local/first_boot_flag --one-file-system /
```

Note: if there is a mounted directory on the system, an error message will appear at the end, which can be ignored.

12.2 Making a bootable SD card from a root filesystem

Run the following script on your Linux PC host, we'll only mention "debian-bullseye-desktop-arm64 os" for brevity, but you can apply the same process for every linux OS.

```
su root
git clone https://github.com/friendlyarm/sd-fuse_rk3399 --single-branch -b kernel-4.19
cd sd-fuse_rk3399
tar xvf /path/to/netdrive/03_Partition\ image\ files/debian-bullseye-desktop-arm64-images.tgz
tar xvf /path/to/netdrive/03_Partition\ image\ files/emmc-eflasher-images.tgz
scp pi@BOARDIP:/rootfs.tar.gz /rootfs.tar.gz
mkdir rootfs
tar xvfz rootfs.tar.gz -C rootfs --numeric-owner --same-owner
./build-rootfs-img.sh rootfs debian-bullseye-desktop-arm64
./mk-sd-image.sh debian-bullseye-desktop-arm64
./mk-emmc-image.sh debian-bullseye-desktop-arm64 autostart=yes
```


13 Configuring kernel command line parameters (only support for kernel4.4)

13.1 eMMC Boot

Here are the steps:

Make an eflahser bootable SD card (use the firmware file starting with rk3xxx-eflasher-),

Insert the SD card into your computer, go to the SD card's OS-related directory, and edit the file parameter.txt, which is a text file containing command-line parameters,

Then boot from the SD card and burn the system to the eMMC.

13.2 SD Boot

To modify the command line parameters of the SD card, you need to repackage the SD card image file, you can use the sd-fuse script we provide to assist packaging:

```
git clone https://github.com/friendlyarm/sd-fuse_rk3399.git -b master --single-branch
cd sd-fuse_rk3399
tar xvfz /path/to/netdrive/03_Partition\ image\ files/friendlydesktop-arm64-images.tgz
tar xvfz /path/to/netdrive/03_Partition\ image\ files/emmc-flasher-images.tgz
vim friendlydesktop-arm64/parameter.txt # Edit command-line parameters
./mk-sd-image.sh friendlydesktop-arm64 # Repackage sd image file
./mk-emmc-image.sh friendlydesktop-arm64 # Repackage sd-to-emmc image file
```

14 More OS Support

14.1 DietPi



DietPi is a highly optimised & minimal Debian-based Linux distribution. DietPi is extremely lightweight at its core, and also extremely easy to install and use. Setting up a single board computer (SBC) or even a computer, for both regular or server use, takes time and skill. DietPi provides an easy way to install and run favourite software you choose.

For more information, please visit this link <https://dietpi.com/docs/>.

DietPi supports many of the NanoPi board series, you may download the image file from here:

- <https://dietpi.com/docs/hardware/#nanopi-series-friendlyarm>

15 Link to Rockchip Resources

- RK3399 datasheet V2.1 (http://opensource.rock-chips.com/images/d/d7/Rockchip_RK3399_Datasheet_V2.1-20200323.pdf)
- RK3399TRM V1.4 (http://opensource.rock-chips.com/images/e/ee/Rockchip_RK3399TRM_V1.4_Part1-20170408.pdf)

16 Schematic, PCB CAD File

- Schematic: NanoPi-R4S-1GB-2008-Schematic.pdf (<https://wiki.friendlyelec.com/wiki/images/0/06/NanoPi-R4S-1GB-2008-Schematic.pdf>) NanoPi-R4S-4GB-2008-Schematic.pdf (<https://wiki.friendlyelec.com/wiki/images/c/c2/NanoPi-R4S-4GB-2008-Schematic.pdf>)
- PCB CAD File: NanoPi_R4S_1GB_2008_dxf.zip (https://wiki.friendlyelec.com/wiki/images/a/ab/NanoPi_R4S_1GB_2008_dxf.zip) NanoPi_R4S_1GB_2008_dxf.zip (https://wiki.friendlyelec.com/wiki/images/1/18/NanoPi_R4S_4GB_2008_dxf.zip)

17 Known Issues List

- Q: UGREEN 18W QC power adapter cannot power R4S?
 - A: It needs to wait for a few seconds to work normally.

18 Update Log

18.1 2023-12-01

18.1.1 FriendlyWrt

- Update to kernel 6.1.63
- Update to openwrt-23.05.2

18.2 2023-05-26

18.2.1 FriendlyWrt

- Updated v22.03 to openwrt-22.03.5
- Updated v21.02 to openwrt-21.02.7

18.3 2023-04-26

18.3.1 FriendlyWrt:

- Upgrade v22.03 to openwrt-22.03.4
- Upgrade v21.02 to openwrt-21.02.6

18.4 2023-02-10

18.4.1 Added Debian11

There are three versions:

- Debian11 Core: Command-line only
- Debian11 Minimal: With Xfce desktop, lite version
- Debian11 Desktop: With Xfce desktop, full version

18.5 2023-01-09

18.5.1 FriendlyCore:

- optimized the systemd service

18.6 2022-12-04

18.6.1 FriendlyWrt:

- Fix the issue that the storage space cannot be expanded
- Improve stability of the eMMC Tools

18.7 2022-09-06

18.7.1 FriendlyWrt:

- Improved eMMC read performance of NanoPi-R4SE
- Added Fullcone NAT support
- upgrade to 22.03.0
- Fix NanoPC-T4 eMMC stability issue

18.8 2022-08-03

18.8.1 FriendlyWrt:

- Upgrade FriendlyWrt to the latest version 22.03-rc6
- Fixed the problem that the R4S/R4SE may not recognize the pcie device (lan port) after a soft reboot (small probability)
- Fixed the issue where the R4SE status led did not reflect the burn progress when burning the system to eMMC
- Firewall settings adjustment: single-port devices (e.g. NanoPi-T4/NanoPi-M4) are set to allow WAN inbound traffic by default for easy web configuration, while multi-port devices are still denied WAN inbound traffic by default
- Updated FriendlyWrt firmware with 4.19 kernel to match FriendlyWrt 21.02 docker with 5.15 kernel

18.9 2022-07-27

18.9.1 FriendlyWrt:

- Beta version 22.03-rc3 is available, you can choose according to your package requirements, stable version 21.02.3 is recommended.
- Both docker and non-docker versions are available, all features are the same except for docker.
- Improved compatibility issues with third-party packages
- Added support for "Soft Factory Reset" function
- Added web-based tool eMMC-Tools, support install FriendlyElec and some third party firmware to eMMC, besides raw-image also support rockchip package format firmware
- Other details: default timezone setting to Shanghai, new NAS category menu, remove lcd2usb, improve security settings, tune sysctl parameters, fix docker firewall settings, etc.
- Add support for new hardware model: NanoPi-R4SE

18.10 2021-10-29

18.10.1 FriendlyWrt:

- FriendlyWrt has been updated to the official stable version 21.02.1, features are basically the same as 19.07.5, support docker, usb wifi, etc.

18.11 2021-08-31

18.11.1 FriendlyWrt:

- Upgraded kernel to 5.10.60
- Add a high-speed 5G USB WiFi support, the network card model is Comfast CF-WU782AC V2, the chip model is MediaTek MT7662
- Improved USB WiFi compatibility
- Improved PWM fan support, fan controlled by kernel drive, temperature control support (Please search for "PWM fans" on the R4S WiKi page for details)
- Improved stability on first boot (previous version, bpfiler error occurred in some cases on first boot)

18.12 2020-12-23

- FriendlyWrt has been updated to the official stable version 19.07.5

Retrieved from "https://wiki.friendlyelec.com/wiki/index.php?title=NanoPi_R4S&oldid=25852"

-
- This page was last modified on 20 May 2024, at 03:41.