

Template:Networking rutos software development kit software development kit instructions

[Main Page](#) > [FAQ](#) > [Other Topics](#) > **Template:Networking rutos software development kit software development kit instructions**

A **software development kit (SDK)** is a set of software development tools that provides the possibility to create applications for a certain software package, software framework, computer system or similar development platform.

Note: The information in this page is in accordance to using SDK version **R_00.07.03**. The included Readme file in SDK archive file contains information for each separate SDK version and might differ from the older version or information provided in this wiki page. Please always follow the provided Readme file.

□

Contents

- [1 Introduction](#)
- [2 Prerequisites](#)
- [3 Ubuntu Installation and Pre-requisite](#)
- [4 Compiling a standard firmware](#)
- [5 Changing default settings](#)
- [6 Selecting packages](#)
- [7 Compiling a single package](#)

Introduction

This article contains instructions on how to utilize Teltonika SDK packages for RUTx series routers. The resources used in the provided examples are:

- Teltonika Router running RutOS
- [Ubuntu 22.04.3 LTS OS](#)
- SDK version **R_00.07.05.x**

The examples apply to various RUTOS routers, SDK versions and operating systems. In most cases the difference will only be that of a visual nature, The availability of SDK Firmware branding begins from version 7.05, although some of these modifications may be accessible on earlier firmware versions.

Most examples provided in this page are independent from each other. But it is highly recommended to acquaint yourself with the basics by reading the "[Prerequisites](#)" and "[Compiling a standard firmware](#)" sections first as they contain information that will be necessary in order to understand

some aspects of the other examples.

It should also be noted early that the first time you compile a firmware it may take up to a few hours to complete. Once the initial compilation is complete, packages will not be re-compiled next time, therefore the duration will decrease significantly. Please be aware that this also depends on your machine.

Prerequisites



You will need:

- A PC, laptop or virtual machine running Linux OS (We recommend using Ubuntu 22.04.3 LTS)
- A RUTOS supported router (version R_00.07.00 and up)
- An SDK intended for your router, which can be downloaded here: [Software Development Kit](#)

Ubuntu Installation and Pre-requisite

Compiling the firmware should be achievable on any Linux-based machine. Alternatively, options like Oracle's VirtualBox or Windows' WSL can be used, though the setup process may vary slightly. For this example, I will be using VMWare.

You can download the **VMWare** here: [VMWare](#)

1. You must download latest **Ubuntu** version can be found here: [Ubuntu](#) 
2. After initiating VMware, click Create a New Virtual Machine 
3. Choose Typical -> choose the downloaded Ubuntu .iso file




then click Next.

4. Input Name, username, and password. Then input the Virtual Machine Name and the save path.

Note: allocate at least 30-50GB GB for disk size and click Store virtual disk as a single file.




Then click Next.

5. Select Customize Hardware. Allow for 4GB or more of memory and 2 or more processors *more is preferable* depending on your machine. 

Finally, click Finish.

Ubuntu will initialize. Choose your language. The example will be in English.

6. Choose Minimal installation and check Install third-party software for graphics and Wi-Fi hardware and additional media formats. 

Click Continue

7. Once installation is done, click on Restart Now.

8. Once Ubuntu is now running. Open a browser and download the SDK file from Teltonika Networks wiki. You can find it out here: [Software Development Kit](#)

Compiling a standard firmware


First, you must install the packages required for the SDK to work. Open the Terminal application (**Ctrl + ALT + T**) and execute the following commands:

```
sudo apt update
```

```
sudo apt install nodejs
```

```
sudo apt install npm
```

```
sudo apt install build-essential ccache ecj fastjar file flex g++ gawk \  
gettext git java-propose-classpath java-wrappers jq libelf-dev \  
libffi-dev libncurses5-dev libncursesw5-dev libssl-dev libtool \  
python python2.7-dev python3 python3-dev python3-distutils \  
python3-setuptools rsync subversion swig time u-boot-tools \  
unzip wget xsltproc zlib1g-dev bison
```

Create a new folder (for this example I'll be using a directory called *RUTX_R* found at my **home** directory) and extract the SDK archive inside it. 

You can achieve this with drag and drop or by executing this command via Terminal:

Navigate to the "Downloads" folder:

```
cd ~/Downloads
```

Extract the tar.gz file:

```
tar -xzvf RUTX_R_GPL_00.07.05.X.tar.gz
```

This command will extract the contents of the *RUTX_R_GPL_00.07.05.X.tar.gz* file into the current directory, After running this command, you should find the extracted files in the same directory where you ran the command.

Note: don't forget to replace the file name and path in accordance with your own circumstances


Open a Terminal inside the SDK directory. You can change the directory in your current terminal (*cd ~/RUTX_R/openwrt-gpl-ipq40xx.Linux-x86_64*)

Once you open the Terminal, update the feeds:

```
./scripts/feeds update -a
```

Now you can compile a standard firmware by executing this command in the terminal:

```
make
```

If all is in order, the output should look something like this: 

Note: the first time you compile a firmware file it may take up to **30 minutes** before it is complete. Don't close the Terminal window up until then. Once it is finished, you will find the firmware in the `./bin/targets/ipq40xx/generic` directory. It should contain a file `openwrt-ipq40xx-qcom-ipq4018-rutx-squashfs-apps.bin`, it can be used to upgrade your router's firmware via its web interface.

You can speed up this process by passing an extra argument **-j**. This argument then compiles the necessary packages in parallel. To know how much jobs you can pass to the compiling process execute this command in the terminal:

```
nproc
```

This commands output should be a number. This number tells how much processing units are available to the current process. Now execute this command in the terminal:

```
make -j<nproc_output>
```



Troubleshooting tips:

1. If you get error on first 'make', do `./scripts/feeds update -a` once more.
2. Once you extract the SDK files, Rename `rutos-ipq40xx-rutx-gpl` to a different file name and move it to home directory.
3. Try running `make distclean`` if you believe that your SDK installation is corrupt. This will recompile the firmware from scratch.
4. Allowing us to run command. "make -j<nproc_output> V=sc" for verbose output. This way you will see more logs that may help to troubleshoot compilation issues.

Changing default settings

To create a firmware with different default settings, you must change the default in the config files, which are contained in `/openwrt-gpl-ipq40xx.Linux-x86_64/package/`. However, there is no unifying system regarding where one should look for config files related to specific services. Therefore, it is very important to acquaint yourself with the **UCI system** (RutOS configuration file system) in order to successfully navigate through the files:

- [Click here](#) for information on the configuration hierarchy
 - [Click here](#) to find what configs are related to which services
-

In many cases you will find that service names as they are displayed in the router's WebUI are similar to the names of the programs responsible for these services. For instance, [Firewall](#) settings can be changed in the `/openwrt-gpl-ipq40xx.Linux-x86_64/package/network/config/firewall/files` directory. However, if you open the aforementioned directory, you will find that it contains four files:



As you can see from the image above, only one of the files holds the Firewall configuration, while other files contain scripts related to the service. This will be different for each case. For example, among Mobile Utilities files, which can be found in /openwrt-gpl-ipq40xx.Linux-x86_64/package/mobutils/src/, you will find even more items, including different config files which serve a different purpose:



Selecting packages

To select which packages should be compiled into the firmware is through a special menu called **Kconfig**. To enter the **Kconfig**, standing in your projects root directory execute this command in the terminal:

```
make menuconfig
```

After executing this command you should see a view like in the picture below. 

To move around in the **Kconfig** use the arrows on the keyboard, space bar to select or to unselect the package, and enter key to enter sub directories of packages.

Note: Do not unselect any package if you do not know what you are doing.

1. Packages have three markings:

1. <*> - this symbol says that the package will be compiled and added to the firmware
2. <M> - this symbol says that the package will be compiled but will not be added to the firmware
3. < > - this symbol says that the package will not be compiled and will not be added to the firmware



By default all packages that were marked as to be compiled in the process, can be found separately from the firmware. Packages can be found in directory **~/RUT_X/openwrt-gpl-ipq40xx.Linux-x86_64/bin/packages/arm_cortex-a7_neon-vfpv4**.

This directory also has five sub directories. Packages are categorized in these directories by their type.

Compiling a single package

If you ever need to compile a single package you do not need to compile the whole firmware. As an example we will try to compile a text editor called **Nano**.

2. Now we need to do this steps:

1. Open the **Kconfig** by executing this command in the terminal:

```
make menuconfig
```

2. Navigate to the menu section of **Nano** text editor. It can be found under “Utilities→Editors”
3. Select **nano** package
4. Three times press the “**Exit**” entry
5. Save the configuration by selecting “**Yes**” in the prompted window



3. When the package is selected we need to compile and install it:

1. While standing in root directory of the project execute this command in the terminal:

```
make package/nano/compile
```



2. The compiled package can be found in the directory `~/RUT_X/openwrt-gpl-ipq40xx.Linux-x86_64/bin/packages/arm_cortex-a7_neon-vfpv4/packages`. Upload the **nano** install file into the router by executing this command in the terminal:

```
scp bin/packages/arm_cortex-a7_neon-  
vfpv4/packages/nano_4.3-1_arm_cortex-a7_neon-vfpv4.ipk  
root@<router_ip_addr>:/tmp
```

3. Connect to your routers command line interface by executing this command in the terminal:

```
ssh root@<router_ip_addr>
```

4. Change your directory to **/tmp** directory
5. Install the **nano** text editor by executing this command in the terminal:

```
opkg install nano_4.3-1_arm_cortex-a7_neon-vfpv4.ipk
```

6. Try out the nano text editor by executing this command:

```
nano example
```



Note: If the package that you want to compile isn't selected in the **Kconfig**, the install file of the package will not be created