# **OpenWrt Custom Firmware with Python Version Checker**

## **Overview**

This documentation provides a thorough explanation of how I built a custom OpenWrt firmware image for Raspberry Pi 4 Model B. The image includes a lightweight C application (check\_python) that verifies the presence and version of Python 3. It does so by executing python3 --version, printing the result to the terminal, and logging it to /tmp/python\_ver.log.

All build processes are encapsulated in Docker, making the environment reproducible and portable. Additionally, we package our application as a custom .ipk and integrate it via OpenWrt feed mechanisms.

## **Goals**

* Build a portable OpenWrt development environment using Docker.
* Create a custom OpenWrt feed repository with a C program.
* Use OpenWrt SDK source tree inside Docker to integrate the .ipk package.
* Flash the resulting firmware image to an SD card and run it on Raspberry Pi 4.
* Verify the application's functionality in real hardware.

## **Repository Structure**

I maintain two separate repositories for clarity and modularity:

### **1. Main Repository: MockProject\_Dev**

Handles orchestration of firmware build and testing:

MockProject\_Dev/

├── Dockerfile # Builds OpenWrt firmware using SDK source

├── Makefile # Orchestrates build, export, flash, and test

├── .config # Configuration for OpenWrt build (enables custom package)

└── feeds.conf.default # Declares all OpenWrt + custom feeds

### **2. Custom Feed Repository: c-application-feed**

Contains my application in OpenWrt feed format:

c-application-feed/

├── Makefile # Feed description (optional)

└── python\_checking/

├── Makefile # Defines package metadata, build, install steps

└── src/

├── Makefile # Compiles `check\_python.c`

└── check\_python.c # Application source

Feed is tracked using:

* **Branch**: feature/python-version-check
* **Tag**: v1.0-python-check

## **Explanation of Key Files**

### **MockProject\_Dev**

#### **Dockerfile**

* Based on ubuntu:22.04 for compatibility
* Installs build essentials, Python, OpenWrt build tools
* Clones full OpenWrt source
* Copy in feeds.conf.default
* Updates all feeds (including custom)
* Copy in .config
* Runs make defconfig to finalize the configuration
* Builds image using make -j$(nproc)

#### **Makefile**

Automates every task:

* build: Builds image using Docker
* copy\_to\_host: Extracts built image from container
* flash: Flash .img to SD card (uses dd)
* clean: Cleanup image and Docker resources

#### **.config**

Minimal OpenWrt configuration:

* Target is bcm27xx/bcm2711 for Raspberry Pi 4
* Enables:  
  + CONFIG\_PACKAGE\_python3=y
  + CONFIG\_PACKAGE\_python\_checking=y

#### **feeds.conf.default**

Declares official and custom feeds:

src-git packages https://git.openwrt.org/feed/packages.git

src-git python\_checking\_feed https://github.com/ntt1912/c-application-feed.git^feature/python-version-check

### **c-application-feed**

#### **Root Makefile**

Provides a description of feed for clarity.

#### **Package Makefile: python\_checking/Makefile**

* Metadata: package name, version, dependencies (+python3)
* Build/Prepare: Copies src/ into build directory
* Build/Compile: Compiles C source using host cross-compiler
* Package/install: Installs binary into /usr/bin/check\_python

#### **src/Makefile**

Simple:

check\_python: check\_python.c

$(CC) $(CFLAGS) -o check\_python check\_python.c

#### **src/check\_python.c**

int main() {

FILE \*fp = popen("python3 --version 2>&1", "r");

...

FILE \*log = fopen("/tmp/python\_ver.log", "w");

...

}

Uses popen() to get version, logs it to /tmp/, handles error cases.

## **Step-by-Step Process to Build and Test**

### **1. Build Docker Image & Firmware**

make build

### **2. Extract Firmware Image**

make copy\_to\_host

### **3. Flash to SD Card**

make flash DEV=/dev/sdX

### **4. Boot Raspberry Pi**

* Insert SD card
* Power on device
* Connect Ethernet to PC and connect via SSH or UART

### **5. Verify Application**

which check\_python

check\_python

cat /tmp/python\_ver.log

## **Results**

* check\_python is present on Raspberry Pi after boot
* It executes correctly and logs output to /tmp/python\_ver.log
* When Python is not available, the tool prints an error

Screenshot of Result:



