MOCK PROJECT DOCUMENT

1.1. Objective:

This project aims to simulate the development and deployment of a simple userspace utility for OpenWRT running on a Raspberry Pi 4B. The system checks for the presence of Python 3.9+ in the target environment and logs the result. It includes full automation using Docker, C programming, and OpenWRT packaging (.ipk).

Key goals include:

- Build a customized OpenWRT firmware image for Raspberry Pi 4.
- Integrate Python 3.11+ into the firmware at build time (no need for opkg).
- Develop a C application check_python that verifies the installed Python version.
- Use Docker and a Makefile to streamline the entire build, test, and packaging workflow.

1.2. Project structure

```
openwrt-python-vercheck/
├── Dockerfile              # Defines containerized build environment
├── Makefile                # Automates build, test, and packaging
├── check_python/           # OpenWRT package definition
│   ├── Makefile
│   └── src/
│       └── check_python.c  # The C application source code
├── output/                 # Output firmware and .ipk packages
├── openwrt_config.seed     # Optional predefined OpenWRT .config
└── README.md               # Project documentation
```

1.3. Technologies & Tools

Tool	Purpose
OpenWRT	Embedded Linux OS for routers and IoT devices
Docker	Íolated build and test environment
Python 3.11	Programming language embedded in the firmware
C language	Used to implement check_python utility
Raspberry Pi 4	Target hardware platform
Makefile	Automtes common build tasks
Git	Source control and version tagging

About the check_python App

Purpose**:** To check for Python 3.9+ presence and version at runtime.

Implementation:

- Uses open("python3 --version") to retrieve the version string.
- Prints the result to terminal:
  Detected Python Version: 3.11.x
- Writes the output to /tmp/python_ver.log.
- Prints an error if Python is not found.
- Allocated in openwrt/package/check_python/, which s a local (internal) package added manually to the OpenWRT build system, not external repositories.

In my Dockerfile:

COPY --chown=builder:builder check_python/ ./openwrt/package/check_python/

This copies your local package directly into the OpenWRT build tree. Then you enabled it with:

echo "CONFIG_PACKAGE_check_python=y" >> .config

About the Makefile

This Makefile automates the process of building OpenWRT inside a Docker container, running the container, extracting output files (firmware and .ipk), and cleaning up. It simplifies repetitive tasks in your OpenWRT image development workflow.

- Build the Docker image

```
# Build the Docker image
build:
    sudo docker build -t $(IMAGE_NAME) .
```

Builds a Docker image from the current Dockerfile.

Tags the image as openwrt-builder.

```
dai@dai:~/openwrt-python-check$ make build
sudo docker build -t openwrt-builder .
```

- Run the Docker container

```
# Run the container interactively with volume mounted
run:
    sudo docker run --rm -it \
        -v $(OUTPUT_DIR):/home/builder/openwrt/bin \
        --name $(CONTAINER_NAME) \
        $(IMAGE_NAME)
```

Runs the built Docker image as a container interactively (-it).

Mounts the local output/ directory to /home/builder/openwrt/bin inside the container.

Automatically removes the container after it exits (--rm).

- Extract .ipk and Firmware to host

```
# Extract .ipk and firmware images to host output/
package:
    @echo "Copying .ipk and firmware files to output/..."
    mkdir -p $(OUTPUT_DIR)
    sudo docker run --rm \
        -v $(OUTPUT_DIR):/home/builder/openwrt/bin \
        --entrypoint /bin/bash \
        $(IMAGE_NAME) -c 'cp -r /home/builder/openwrt/bin/targets/*/*/*.img /home/builder/openwrt/bin/ 2>/dev/null || true && \
          find /home/builder/openwrt/bin/packages/ -name "*.ipk" -exec cp {} /home/builder/openwrt/bin/ \; 2>/dev/null || true'
```

Runs a temporary container to copy all firmware .img files and .ipk packages from inside the container to the mounted host output/ directory.

Uses || true to ignore errors if some files are missing.

Useful for collecting build artifacts without needing to enter the container.

- Clean up Docker Image and Output

```
# Clean up Docker image and output folder
clean:
    sudo docker image rm $(IMAGE_NAME) || true
    sudo rm -rf $(OUTPUT_DIR) || true
```

Removes the Docker image openwrt-builder (if exists).

Deletes the output/ directory and its contents.

1.4. Execution workflow

1.4.1. C App Development (check_python.c)

- Written in C, uses popen() to run python3 --version.
- Parses and logs the version string.
- Handles the absence of Python gracefully.

1.4.2. OpenWRT Package Creation

- The app is wrapped in an OpenWRT package named check_python.
- Installs the binary to /usr/bin/ on the target device.

1.4.3. Dockerfile for Building OpenWRT

- Uses ubuntu:22.04 as the base image.
- Installs all OpenWRT dependencies (gcc, make, Python, etc.).
- Clones OpenWRT v23.05.3.
- Adds check_python to package/ directory.
- Enables required packages via .config:
  - CONFIG_PACKAGE_check_python=y
  - CONFIG_PACKAGE_python3=y
  - CONFIG_PACKAGE_libpython3, zlib, libbz2, etc.
- Builds a fully integrated .img image.

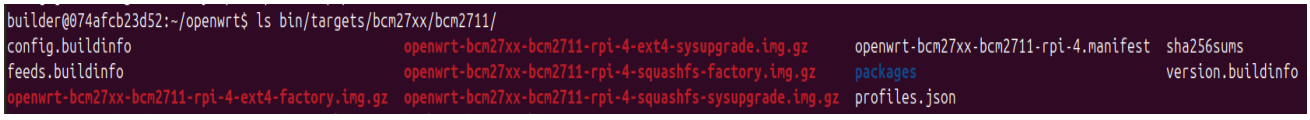
1.4.4. Building via Docker

```
sudo docker build -t openwrt-builder .
```

1.4.5. Flashing the image

Extract .img.gz and flash to SD card using Raspberry Pi Imager

Inside container, img.gz is allocated as the below picture



Because in Makefile of project, i have a make package to copy .ipk and .img files to the output/ dir of host computer.

```
dai@dai:~/openwrt-python-check/output/targets/bcm27xx/bcm2711$ ls
config.buildinfo
feeds.buildinfo
openwrt-bcm27xx-bcm2711-rpi-4-ext4-factory.img
openwrt-bcm27xx-bcm2711-rpi-4-ext4-sysupgrade.img.gz
openwrt-bcm27xx-bcm2711-rpi-4.manifest
openwrt-bcm27xx-bcm2711-rpi-4-squashfs-factory.img.gz
openwrt-bcm27xx-bcm2711-rpi-4-squashfs-sysupgrade.img.gz
packages
profiles.json
sha256sums
```

Flash openwrt-bcm27xx-bcm2711-rpi-4-ext4-factory.img

1.4.6. Testing on Raspberry Pi 4

Connect via Ethernet:

ssh root@192.168.1.1

Expected output:

```
root@OpenWrt:/# check_python
Detected Python Version: Python 3.11.7
root@OpenWrt:/# cat /tmp/python_ver.log
Detected Python Version: Python 3.11.7
```

1.5. Git integration

1.5.1. Initialize a Git repository

cd openwrt-python-check

git init

git add .

git commit -m “Done mock”

1.5.2. Create a new branch named feature/python-version-check

git checkout -b feature/python-version-check

1.5.3. Connect to the remote repository

git remote add origin

https://github.com/Dinh-Dai/Mock.git

1.5.4. Push code to the new branch

git push -u origin feature/python-version-check

1.5.5. Tag the version

git tag v1.0-python-check

git push origin v1.0-python-check

1.5.6. Check Git tags

git tag

git show v1.0-python-check