1. Download OpenWrt SDK

Goal: Download Openwrt source code and everything needed for building custom firmware images

To download Openwrt gitgub repository on workstation, go to desired folder and execute (1).

```
1. git clone https://git.openwrt.org/openwrt/openwrt.git
```

2. Configure and compile for the QEMU target

Goal: Prepare build system for build process, select build options for custom firmware image and run build system to compile it.

Switch to Openwrt 23.05 branch from master (1), update feeds and install them (2,3). Start menuconfig (4), select build options, save and exit menuconfig:

- Target system: *x86*
- Subtarget: *x86_64*
- Target profile: *Generic x86/64*
- Base system: *rpcd*
- Target images: *Build LiveCD image (ISO)*
- Utilities/prosps-ng: *procps-ng-ps*

The modified selection has been extracted from .config (5) into the diffconfig_x86 file. The folder *files* contain network configuration (90-network.sh), an rpcd script (pid) and password configuration (passwd, shadow *root:root*) if needed, which is copied into the Openwrt structure. Now we are ready for compile (6) (download necessary files and compile using 24 cores).

```
1. git checkout openwrt-23.05
2. ./scripts/feeds update -a
3. ./scripts/feeds install -a
4. make menuconfig
5. ./scripts/diffconfig.sh > diffconfig_x86
6. make download && make -j25
```

3. Boot the image with QEMU

Goal: Install QEMU, boot builded Openwrt image

Install QEMU (1), boot image (2).

```
1. sudo apt install qemu-system
2. qemu-system-x86_64 -boot d -cdrom \
~/openwrt/bin/targets/x86/64/openwrt-x86-64-generic-image.iso -m 1024 \
-nographic
```

4. Configure a working network access

Goal: Configure networking for QEMU and Openwrt

QEMU was initialized to include network connection *br0* within image boot (1). According to *https://wiki.debian.org/QEMU#Networking* bridge br0 was configured in /etc/network/interfaces (*interfaces.config*). Since IP address has been configured through script (90-network.sh) in *files* no need for manual setup with UCI commands over SSH connection. For WAN connection default route was added with *ip* command (2).

```
1. qemu-system-x86_64 -boot d -cdrom myimages/x86.iso \
-m 1024 -nographic -net nic -net bridge,br=br0
2. ip route add default via 192.168.8.1 dev br-lan
```

5. Create a simple rpcd script: Develop a simple script that returns a JSON list of running processes (PID, service)

Goal: Utilize *rpcd* which runs on *ubus* to list running processes.

Script is named *pid* and saved in *./files/usr/libexec/rpcd/*. Processes informations is retreived with *ps c*ommand which reads PID and COMMAND, stores it in *ps_output* variable separated with space. Afterwards *ps_output* is is parsed line by line with command *read* which stores data in variables *pid* and *service*. Finnaly variables *pid* and *service* are written to stdout as JSON objects with *printf* command.

6. Write Robot Framework test suite

Goal: Utilize Robot Fraamework to test Openwrt image booting, has functional networking and rpcd script returns relevant data.

Test suite file is named qemu_test.robot and initialized with log level set to trace (1). The *Start Process* test case initiates QEMU process in the background. After 25 seconds sleep (time to boot Openwrt) the *Ping Guest* test case utilizes *ping* to test working network connection to the guest, checking for received packets. RPCD script is tested through SSH connection with *List Processes* test case. Command *ubus call pid service* has been sent to guest over SSH and response in form of existing service name was checked (*procd)*. Both log and report files are included in repository.

```
1. robot -L trace qemu_test.robot
```