Getting Started with i.MX 8M Mini and Matter

CONTENTS

1 INTRODUCTION

Matter (previously known as Project CHIP) is a new single, unified, application-layer connectivity standard designed to enable developers to connect and build reliable, secure IoT (Internet of Things) ecosystems and increase compatibility among Smart Home and Building devices.

For enabling Matter devices, NXP offers scalable, flexible and secure platforms to enable the variety of use cases Matter addresses – from end nodes to gateways – so device manufacturers can focus on product innovation and accelerating time to market.

This document focuses on NXP's solution for Matter controller and OpenThread Border Router (OTBR) using the i.MX8M Mini Processor family.

The i.MX 8M Mini is NXP's first embedded multicore applications processor built using advanced 14LPC FinFET process technology, providing more speed and improved power efficiency. With commercial and industrial level qualification and backed by NXP's product longevity program, the i.MX 8M Mini family may be used in any general purpose industrial and IoT application.

https://www.nxp.com/part/i.MX8MMini#/

2 HARDWARE SETUP

There are two possible hardware solutions for the i.MX8M Mini matter enablement, that will be detailed below

- i.MX8MM + 88W8987(WiFi-BT combo Module) + K32W(OpenThread RCP module)
- i.MX8MM + IW612 (WiFi-BT-Thread tri-radio single-chip module)

2.1 Hardware Requirements

- i.MX 8M Mini Evaluation Kit (which contains the required materials below)
  - i.MX 8M Mini EVK
  - USB Type C power supply
  - USB A to micro USB cable
  - USB A to USB C cable
- K32W061 DK006
  - K32W061 DK006 + OM15082
  - K32W USB Dongle or K32W061 DK006
  - USB A to MINI USB

- USB type C to A
- IW612 A1 RD Board
  - Micro-SD to SD adapter
  - SDIO adapter with ribbon cable
  - RD-IW612-CSP-IPA-2A-V2 evaluation board
- Linux host computer
- Wi-Fi Access Point which supports IPv6 and IPv6 DHCP server

2.2 Software Requirements

- Universal Update Utility (UUU)
- A serial COM terminal for the host computer o For instructions on how to set up a serial COM terminal software, please refer to page 14 in the i.MX 8M Mini LPDDR4 EVKB Hardware User's Guide.

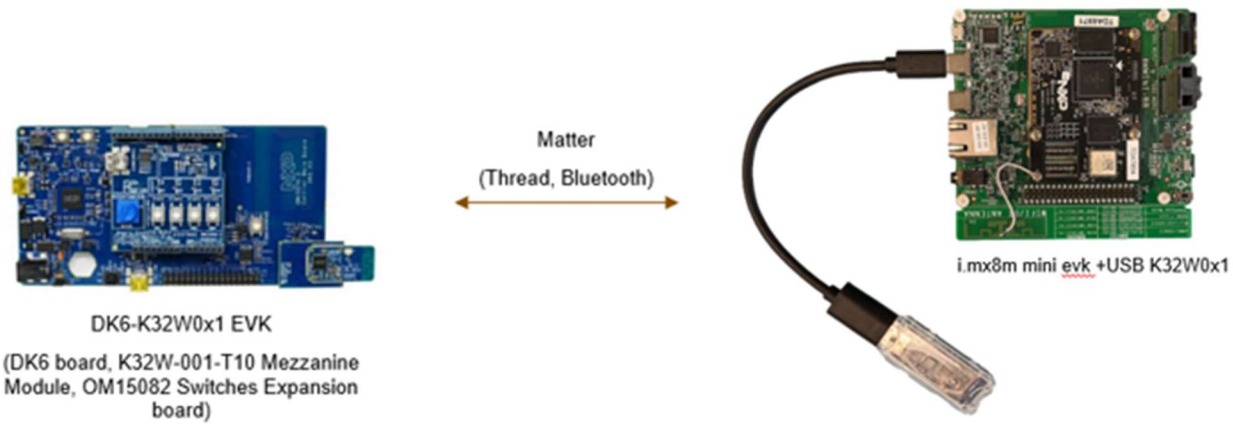
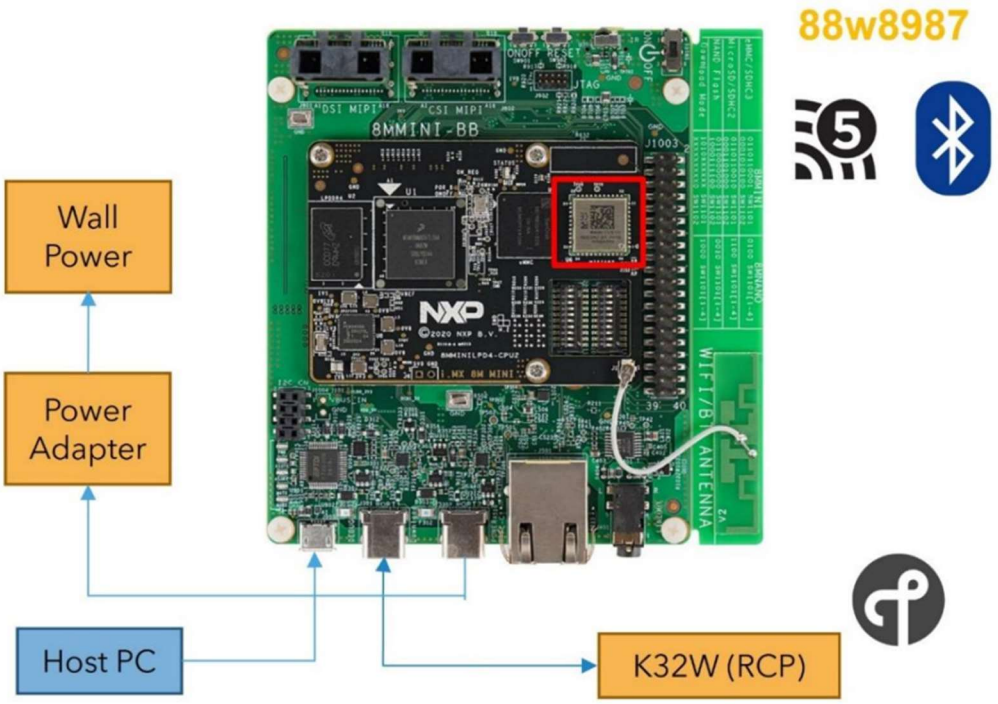
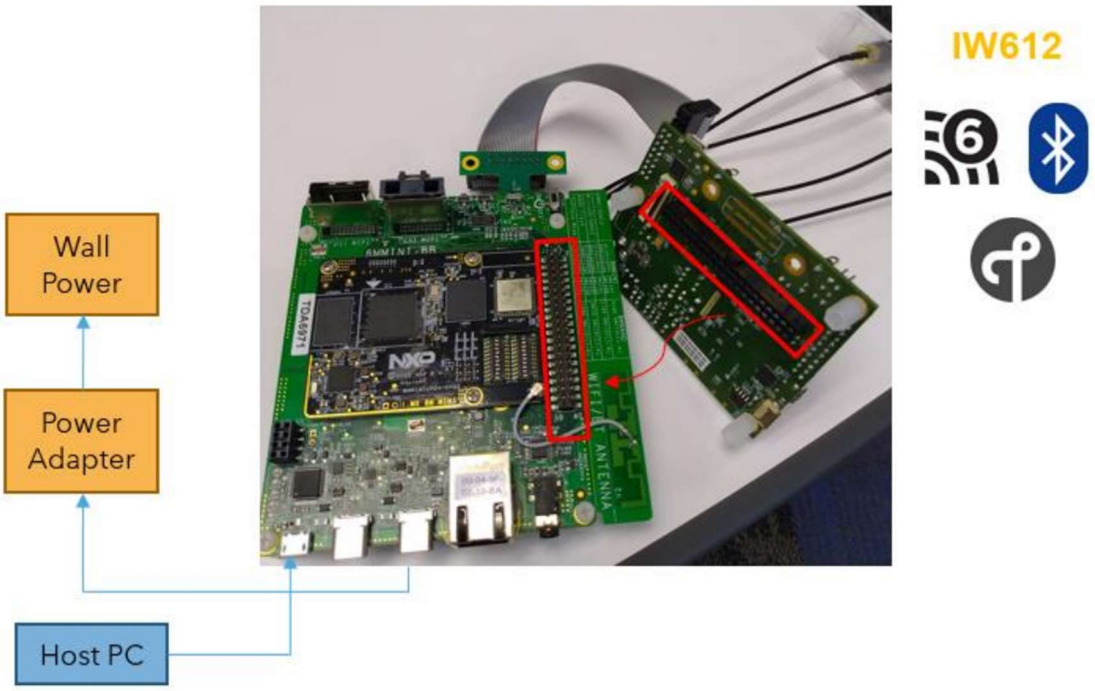


Figure 2-1 i.MX 8M Mini + K32W Matter setup

2.3 Hardware connections



OTBR i.MX8MM + 88W8987 + K32W



OTBR i.MX8MM + IW612

Note: Although L5.15.52 BSP contains the IW612 driver, it only support Wi-Fi/BT, it doesn't support thread. So for Matter enablement, you need to build the IW612 OTBR driver and firmware separately.

3 MATTER ENVIRONMENT SETUP

Matter development relies on open-source resources, leveraging Linux based operating systems like Ubuntu and other tools like git, gcc or python. This also includes GN, a meta build system that generates makefiles and Ninja, a build system meant to replace Make tool.

Matter is available as an open-source SDK (Software Development Kit) containing all the necessary components from scripts to install required tools to stack source code and vendor provided applications.

First step in developing a Matter application is to have Linux support for the build. The recommendation is to have a native Linux machine. If Windows is preferred operating system, support for the build can be set by using either:

- Windows Subsystem for Linux (WSL),
- Linux Virtual Machine.

3.1 WSL Ubuntu 20.04 LTS

The Windows Subsystem for Linux (WSL) provides a GNU/Linux environment directly to Windows.

Use the following steps to install the WSL Ubunut 20.04 LTS (Long Term Support):

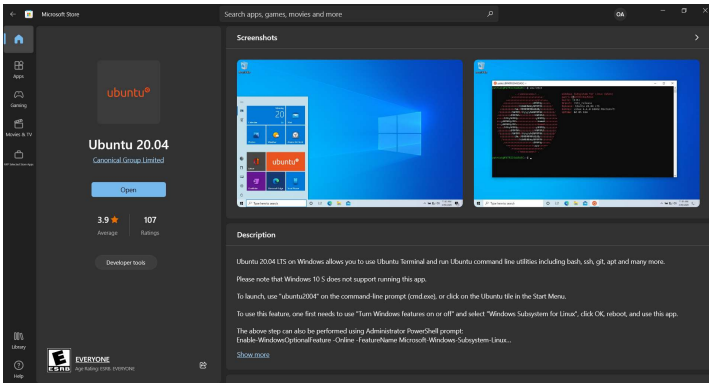
1. On Windows 10, open PowerShell as administrator and run the following commands:
   - Enable the Windows Subsystem for Linux:

```
dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart
```

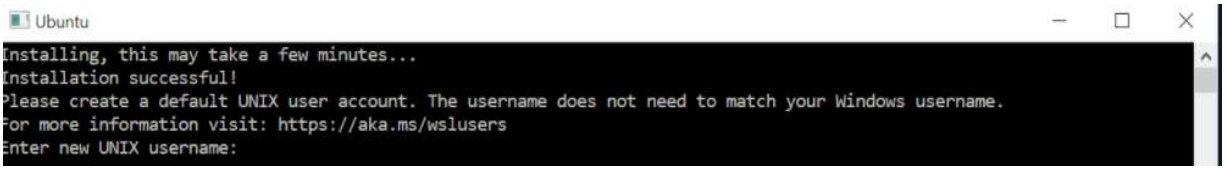
   - Enable virtual machine feature:

```
dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart
```

   - Restart your machine to complete the WSL installation
2. Install the Ubuntu 20.04 from Microsoft Store

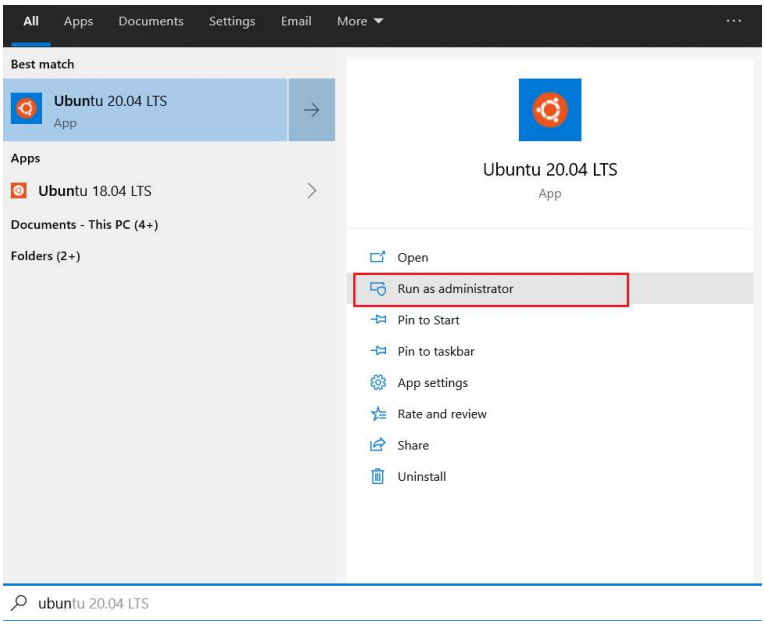


   - Create a user account and password for your new Linux distribution



- Next you will have to check the WSL version and activate WSL 1, if 2 is used. Run PowerShell as administrator:
  The following command is used to check the wsl version:
  `wsl --list -verbose`
  If wsl version 2 is used, then activate wsl 1 using:
  `wsl --set-version Ubuntu-20.04 1`

- After installation is complete, run Ubuntu 20.04 LTS as administrator.



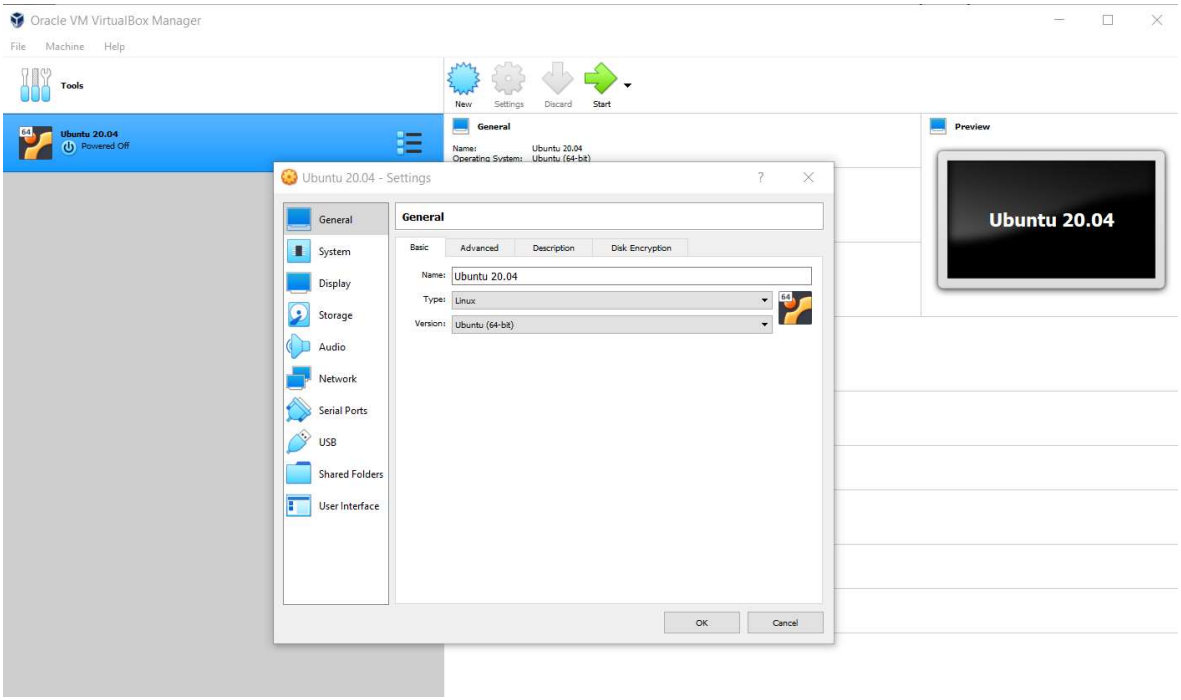
3.2 Linux Virtual Machine

The following steps guide you through virtualbox machine installation steps:

- Download and install virtual machine: https://www.virtualbox.org/wiki/Downloads
- Please consider that based on your operating system some extra steps will be required for enabling the Virtualization support.
- Download the desktop image for Ubuntu 20.04 Focal and create virtual machine using the downloaded ISO.
- Configure the virtual machine:
  - Linux Ubuntu 20.04 (64 bit);
  - VM (Virtual Machine) disk size – more than 20GB;

- Enable USB controller -> USB 1.1(OHCI) Controller (support for USB 2.0 is recommended, if possible, this being available via the Oracle VM VirtualBox Extension Pack);

- Enable network adapter-> Adapter 1 -> Attached to Bridged Adapter;



3.3 Setting up Matter Environment

The following steps guide you through creating Matter build environment for K32W0x1 MCU.

1. Matter Dependencies:
   - Check for updates and install dos2unix (useful for WSL):

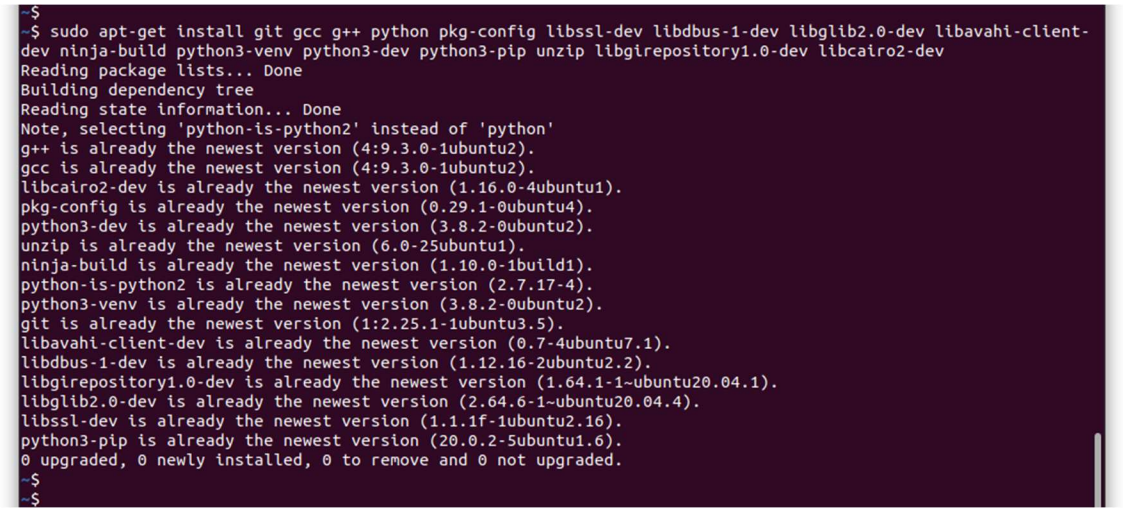
```
$ sudo apt update
```

```
$ sudo apt upgrade --y
```

```
$ sudo apt-get install dos2unix
```

   - Install Matter dependencies:

```
$ sudo apt-get install git gcc g++ python pkg-config libssl-dev libdbus-1-dev libglib2.0-dev libavahi-client-dev ninja-build python3-venv python3-dev python3-pip unzip libgirepository1.0-dev libcairo2-dev gcc-arm-none-eabi
```



   - Restart the Linux machine.

2. Generating the i.MX8MM Yocto image:

   The Yocto Project (YP) is an open source collaboration project that helps developers create custom Linux-based systems regardless of the hardware architecture. The project provides a flexible set of tools and a space where embedded developers worldwide can share technologies, software stacks, configurations, and best practices that can be used to create tailored Linux images for embedded and IOT (Internet of Things) devices, or anywhere a customized Linux OS (Operating System) is needed.

- To build the Yocto Project, some packages need to be installed. The list of packages required are:

```
$ sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc-multilib \build-essential chrpath socat cpio python3 python3-pip python3-pexpect \xz-utils debianutils iputils-ping python3-git python3-jinja2 libegl1-mesa libsdl1.2-dev \pylint3 xterm npm zstd build-essential libpython3-dev libdbus-1-dev python3.8-venv
```

- Then you need to install the repo tool:

```
$ mkdir ~/bin
$ curl https://storage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
$ chmod a+x ~/bin/repo
$ export PATH=~/bin:$PATH
```

- To build the Yocto Project, some python dependency packages need to be installed.

```
$wget https://raw.githubusercontent.com/project-chip/connectedhomeip/master/scripts/constraints.txt
$ pip3 install -r constraints.txt
$ pip3 install build mypy==0.910 types-setuptools pylint==2.9.3
$ pip install dbus-python
```

- If you have not configured the Git environment on your machine, you should to this now:

```
$ git config --global user.name "Your Name"
```

```
$ git config --global user.email "Your Email"
```

```
$ git config --list
```

- Then, Yocto build environment must be setup.
  (Note: The Yocto source code is maintained with a repo manifest, the tool repo is used to download the source code.)

```
$ mkdir ~/bin
$ curl http://commondatastorage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
$ chmod a+x ~/bin/repo
$ export PATH=${PATH}:~/bin
```

```
$ mkdir ${MY_YOCTO}
$ cd ${MY_YOCTO}
$ repo init -u https://github.com/nxp-imx/imx-manifest-b imx-linux-kirkstone -m imx-5.15.52-2.1.0.xml
$ repo sync
```

```
yocto$repo sync
Fetching:  0% (0/13) warming up
Receiving objects: 100% (158102/158102), 14.93 MiB | 31.72 MiB/s, done.2)
Resolving deltas: 100% (103740/103740), done.
Fetching: 100% (13/13), done in 1m45.717s
Garbage collecting: 100% (13/13), done in 0.093s
Checking out: 100% (13/13), done in 0.741s
repo sync has finished successfully.
yocto$
```

- Then integrate the meta-matter recipe into the Yocto code base

```
$ cd ${MY_YOCTO}/sources/
$ git clone https://github.com/NXPmicro/meta-matter.git
```

```
yocto$cd sources/
sources$git clone https://github.com/NXPmicro/meta-matter.git
Cloning into 'meta-matter'...
remote: Enumerating objects: 465, done.
remote: Counting objects: 100% (286/286), done.
remote: Compressing objects: 100% (126/126), done.
remote: Total 465 (delta 118), reused 274 (delta 111), pack-reused 179
Receiving objects: 100% (465/465), 3.24 MiB | 9.36 MiB/s, done.
Resolving deltas: 100% (175/175), done.
sources$
```

Note: If you use the IW612 as RCP you will need to add SPI config in ot-br-posix project.

Add “-DOT_POSIX_CONFIG_RCP_BUS=SPI” to sources/meta-matter/recipes-otbr/otbr/otbr.bb

EXTRA_OECMAKE

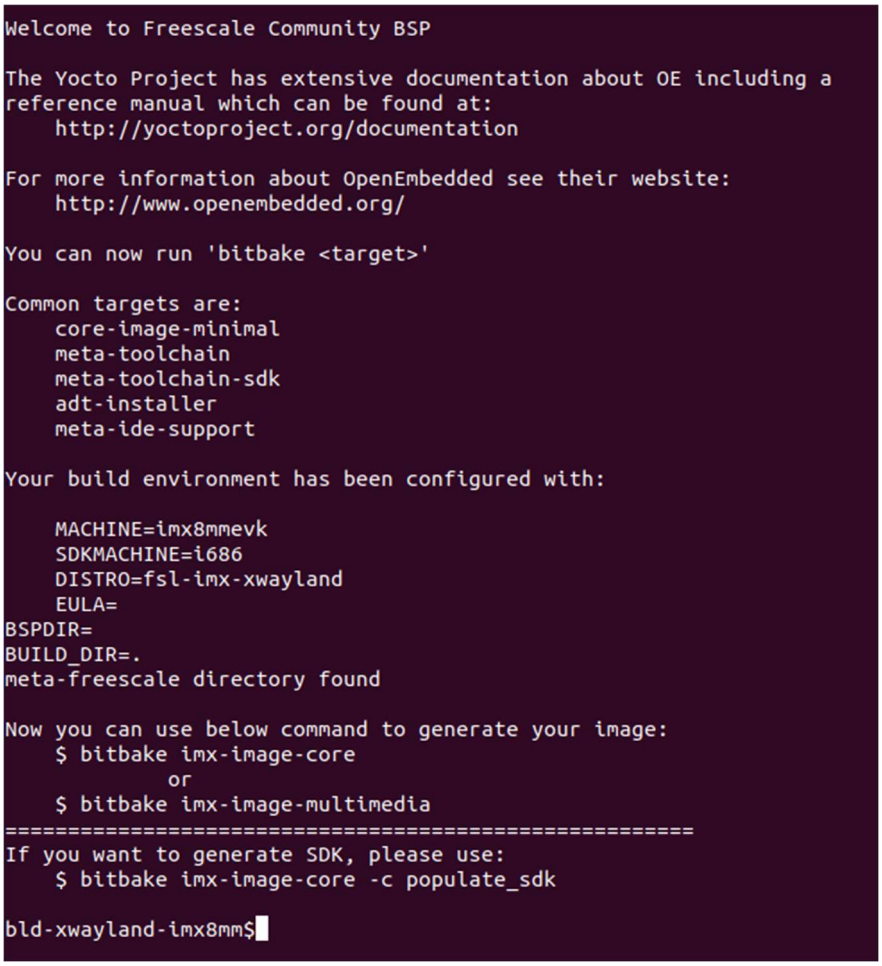
More information about the downloaded Yocto release can be found in the corresponding i.MX Yocto Project User’s Guide which can be found at NXP official website.

- Make sure your default Python of the Linux host is Python2.

```
$ python -version
Python 2.7.x
```

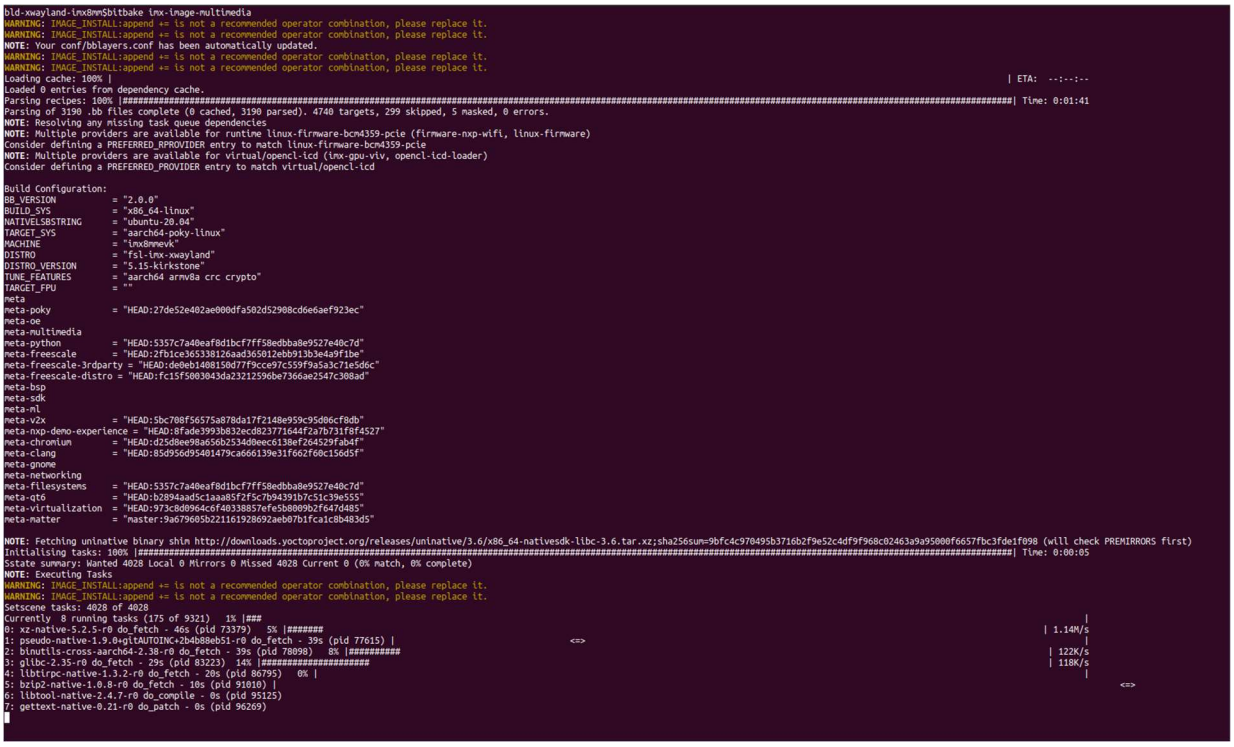
- Change the current directory to the top directory of the Yocto source code and execute the command below.

```
$MACHINE=imx8mmevk DISTRO=fsl-imx-xwayland source sources/meta-matter/tools/imx-iot-setup.sh bld-xwayland-imx8mm
```



- The system will create a directory bld-xwayland-imx8mm/ and enter this directory automatically, execute the command below under these directory to generate the Yocto images.

```
$ Bitbake imx-image-multimedia
```



After execution of previous commands, the Yocto image will be generated under ${MY_YOCTO}/bld-xwayland-imx8mm/tmp/deploy/images/imx8mmevk/imx-image-multimedia-imx8mmevk.wic.bz2 .

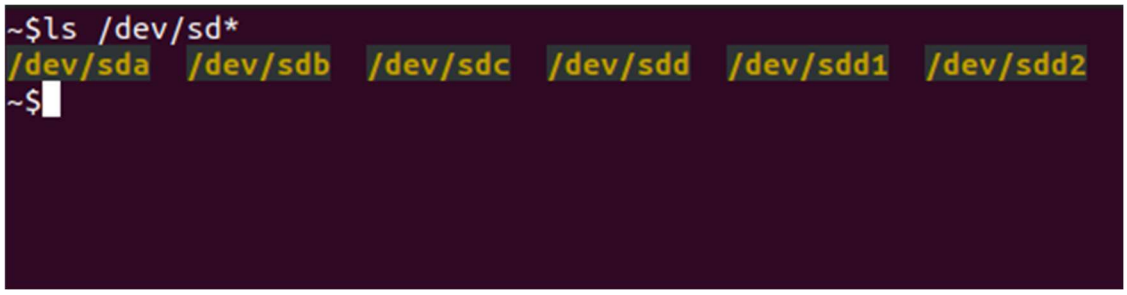
- The bz2 image is a symbolic link file, it is better to copy it to another folder ${MY_images} before unziping it.

```
$ cp ${MY_YOCTO}/bld-xwayland-imx8mm/tmp/deploy/images/imx8mmevk/imx-image-multimedia-imx8mmevk.wic.bz2 ${MY_images}
```

- The bzip2 command is used to unzip this file then the dd command is used to program the output file to a microSD card by running the command sequence below.

(Be cautious when executing the dd command below, make sure that you have the microSD card device connected! /dev/sdc in the command below represents a microSD card connected to the host machine with a USB adapter, however the output device name may vary. Use the command "ls /dev/sd*" to verify the name of the SD card device!)

```
$ cd ${MY_images}
$ bzip2 -d imx-image-multimedia-imx8mmevk.wic.bz2
$ sudo dd if=imx-image-multimedia-imx8mmevk.wic of=/dev/sdc bs=4M conv=fsync
```



- Or use the uuu tool to flash the image on the internal eMMC/SDHC 3 storage
  - connect the board via the Download port.
  - Setup the “Download” boot mode, see printed table onboard
  - Power on the board and use the following command in the same folder with the image

```
$ uuu -b emmc_all imx-image-multimedia-imx8mmevk.wic.bz2
```

3. Generating the i.MX8MM Yocto SDK:

- The Yocto SDK must be generated by the below command:

```
$ cd ${MY_YOCTO}/bld-xwayland-imx8mm
$ bitbake imx-image-multimedia -c populate_sdk
```

- Install the NXP Yocto SDK and set toolchain environment variables. Run the SDK installation script with root permission.

```
$ sudo tmp/deploy/sdk/fsl-imx-xwayland-glibc-x86_64-imx-image-multimedia-armv8a-imx8mmevk-toolchain-5.15-kirkstone.sh
```

- The default target directory for SDK will be prompted during SDK installation, as shown below, enter a new target directory:

NXP i.MX Release Distro SDK installer version 5.15-kirkstone
==========
Enter target directory for SDK (default: /opt/fsl-imx-xwayland/5.15-kirkstone):

- After the Yocto SDK is installed on the host machine, an environment setup script is also generated, and there are prompt lines telling the user to source the script each time when using the SDK in a new shell, for example:

```
$ . /opt/fsl-imx-xwayland/5.15-kirkstone-imx8mm/environment-setup-armv8a-poky-linux
```

- After the SDK package installed in the build machine, the Yocto build environment can be imported using the command:

```
$ source ${iMX8MM_SDK_INSTALLED_PATH}/environment-setup-armv8a-poky-linux
```

4. Matter Building Setup instructions:

   - Clone the Matter SDK using the public repo:

```
$ mkdir ${MY_Matter_Apps}#thisistopleveldirectoryofthisproject
$ cd ${MY_Matter_Apps}
$ git clone https://github.com/project-chip/connectedhomeip.git
$ cd connectedhomeip
$ git checkout -t origin/v1.0-branch-imx
$ git submodule update --init$source scripts/activate.sh
```

4 BUILDING AND RUNNING OPENTHREAD BOARDER ROUTER BINARIES WITH YOCTO SDK

- Fetch latest otbr source codes and execute the build:

```
$ git clone https://github.com/openthread/ot-br-posix
$ cd ot-br-posix
$ git checkout -t origin/main
$ git submodule update –init
$ source ${yocto_matter_sdk}/environment-setup-cortexa53-crypto-poky-linux
```

- For the i.MX8MM + K32W configuration:

```
$./script/cmake-build -DOTBR_BORDER_ROUTING=ON -DOTBR_WEB=ON \-DBUILD_TESTING=OFF -
DOTBR_DBUS=ON -DOTBR_DNSSD_DISCOVERY_PROXY=ON \-DOTBR_SRP_ADVERTISING_PROXY=ON -
DOT_THREAD_VERSION=1.3\-DOTBR_INFRA_IF_NAME=mlan0 \-DOTBR_BACKBONE_ROUTER=ON -
DOTBR_BACKBONE_ROUTER_MULTICAST_ROUTING=ON \-DOTBR_MDNS=mDNSResponder \-
DCMAKE_TOOLCHAIN_FILE=./examples/platforms/nxp/linux-imx/aarch64.cmake
```

- For the i.MX8MM + IW612 configuration

```
$ ./script/cmake-build -DOTBR_BORDER_ROUTING=ON -DOTBR_WEB=ON \ -DBUILD_TESTING=OFF -DOTBR_DBUS=ON -
DOTBR_DNSSD_DISCOVERY_PROXY=ON \ -DOTBR_SRP_ADVERTISING_PROXY=ON -DOT_THREAD_VERSION=1.3 \ -
DOTBR_INFRA_IF_NAME=mlan0 -DOT_POSIX_CONFIG_RCP_BUS=SPI \ -DOTBR_BACKBONE_ROUTER=ON -
DOTBR_BACKBONE_ROUTER_MULTICAST_ROUTING=ON \ -DOTBR_MDNS=mDNSResponder \ -
DCMAKE_TOOLCHAIN_FILE=./examples/platforms/nxp/linux-imx/aarch64.cmake
```

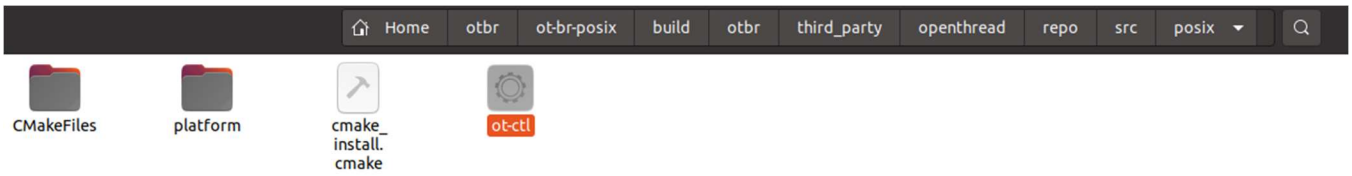
```
ot-br-posix$./script/cmake-build -DOTBR_BORDER_ROUTING=ON -DOTBR_REST=ON -DOTBR_WEB=ON -DOTBR_BACKBONE_ROUTER=ON \
>  -DOT_BACKBONE_ROUTER_MULTICAST_ROUTING=ON -DBUILD_TESTING=OFF -DOTBR_DBUS=ON -DOTBR_DNSSD_DISCOVERY_PROXY=ON \
>  -DOTBR_SRP_ADVERTISING_PROXY=ON -DOT_THREAD_VERSION=1.2 -DOTBR_INFRA_IF_NAME=mlan0 \
>  -DCMAKE_TOOLCHAIN_FILE=./examples/platforms/nxp/linux-imx/aarch64.cmake
+++ dirname ./script/cmake-build
++ cd ./script/..
++ [[ ! -n '' ]]
++ grep -s 'BeagleBone Black' /sys/firmware/devicetree/base/model
++ case "${OSTYPE}" in
++ have_or_die lsb_release
++ have lsb_release
++ command -v lsb_release
+++ lsb_release -i
+++ cut -c17-
+++ tr '[:upper:]' '[:lower:]'
```

```
changed 7 packages, and audited 8 packages in 9s

2 moderate severity vulnerabilities

To address issues that do not require attention, run:
  npm audit fix

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
npm notice
npm notice New minor version of npm available! 8.12.1 -> 8.19.2
npm notice Changelog: <https://github.com/npm/cli/releases/tag/v8.19.2>
npm notice Run `npm install -g npm@8.19.2` to update!
npm notice
[451/451] Linking CXX executable src/web/otbr-web
ot-br-posix$
```

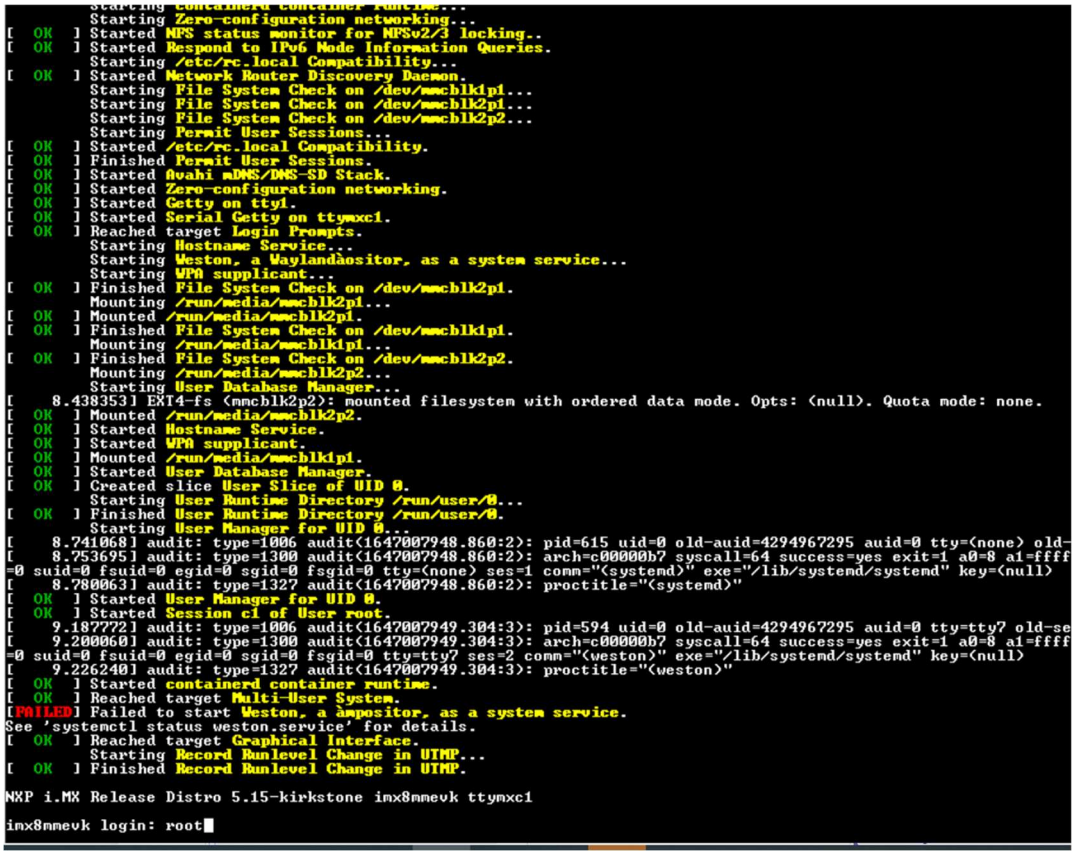
- The otbr-agent is built in ${MY_OTBR}/build/otbr/src/agent/otbr-agent.

- The otbr-web is built in ${MY_OTBR}/build/otbr/src/web/otbr-web.

- The ot-ctl is built in ${MY_OTBR}/build/otbr/third_party/openthread/repo/src/posix/ot-ctl.



- These files must be copied into Yocto's /usr/sbin/ folder.
- Insert the uSD card flashed with the Yocto image in the uSD card port on the board (From Chapter 3.3 step 2)
- Connect a micro USB cable from the DEBUG port on the board in the PC and open a serial connection with the following parameters:
    - Baud rate: 115200
    - 8 data bits
    - 1 stop bit
    - No parity
    - No flow control
- Put the power switch on the ON position
- After the boot sequence when you are prompted to type in an username use “root” (no password required)



4.1 Setup OTBR on IMX8MMEVK + 88W8987+K32W

- Flash k32w RCP binary to K32W061DK6 or USB dongle. For the DK6Programmer installation you can refer to K32W SDK documentation, The ot-rcp-check-usart-idle-not-txifo-dmacore-interface.bin is a designed binary for IMX8MM and supports Matter V1.0 firmware.

```
$ DK6Programmer.exe -s COM${X}-e FLASH -p ot-rcp-check-usart-idle-not-txifo-dmacore-interface.bin
```

- Set /etc/radvd.conf to enable IPv6 RA functions

```
$ vi /etc/radvd.conf

interface mlan0

{

    AdvSendAdvert on;

}
```

- Sync with current time

```
$ date -s "2022-11-20 17:14"
```

- Connect the OTBR to the target Wi-Fi AP network

```
$/usr/libexec/bluetooth/Bluetoothd&
$modprobe moal mod_para=nxp/wifi_mod_para.conf
$ sysctl -w net.ipv6.conf.mlan0.accept_ra_rt_info_max_plen=64
$ sysctl -w net.ipv6.conf.mlan0.accept_ra_rt_info_min_plen=48
$ wpa_passphrase ${SSID} ${PASSWORD} > imxrouter.conf
$ wpa_supplicant -d -B -i mlan0 -c ./imxrouter.conf
$ udhcpc -i mlan0
$ systemctl restart radvd
$ sleep 4
$ hciattach /dev/ttymxc0 any 115200 flow
$ hciconfig hci0 up
$ echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
$ echo 1 > /proc/sys/net/ipv4/ip_forward
$ echo 2 > /proc/sys/net/ipv6/conf/all/accept_ra
$ ln -sf /usr/sbin/xtables-nft-multi /usr/sbin/ip6tables
$ ipset create -exist otbr-ingress-deny-src hash:net family inet6
$ ipset create -exist otbr-ingress-deny-src-swap hash:net family inet6
$ ipset create -exist otbr-ingress-allow-dst hash:net family inet6
$ ipset create -exist otbr-ingress-allow-dst-swap hash:net family inet6
```

- Setup Openthread

```
$ ./otbr-agent -I wpan0 -B mlan0 'spinel+hdlc+uart:///dev/ttyUSB0?uart-
baudrate=1000000' -v -d 5 &
$ iptables -A FORWARD -i mlan0 -o wpan0 -j ACCEPT
$ iptables -A FORWARD -i wpan0 -o mlan0 -j ACCEPT
$ ip6tables -A FORWARD -i mlan0 -o wpan0 -j ACCEPT
$ ip6tables -A FORWARD -i wpan0 -o mlan0 -j ACCEPT
$ hciconfig hci0 up
$ otbr-web &
```

4.2 Setup OTBR on IMX8MMEVK + IW612

IW612 driver and Firmware are not integrated in i.MX BSP.

Using the following link, you can get the necessary package for IW612: [IW612-18.99.1.p140.16-18.99.1.p140.16-MXM5X18354.p7_V1-MGPL]

Change the i.MX8MM dtb files to enable SPI communication with the IW612 A1 RD board

- On PC

```
$ scp dts/imx8mm-evk-iw612-evk.dtb root@${imx8mm_ip}:/run/media/mmcblk2p1
```

- On i.MX8MM in uboot stage

```
Uboot=> setenv fdtfile imx8mm-evk-iw612-evk.dtb
Uboot=> saveenv
Uboot=> boot
```

After this operation you should be able to see **"/dev/spidev1.0"** in the filesystem after a system reboot.

- Install IW612 driver and firmware on IMX8MM
  - On PC

```
$ scp fw/* root@${imx8mm_ip}:/lib/firmware/nxp
$ scp -r iw612 root@${imx8mm_ip}:/home/root
```

- Connect the OTBR to the AP network

```
$ cd /home/root/iw612

$ date -s "2022-11-01 09:22"

$ /usr/libexec/bluetooth/Bluetoothd &

$ ./fw_loader /dev/ttymxc2 115200 0 /lib/firmware/nxp/uartspi_n61x_v1.bin.se 3000000

$ hciattach /dev/ttymxc2 any -s 3000000 3000000 flow dtron

$ hciconfig hci0 up


$ insmod mlan.ko

$ insmod sdxxx.ko mod_para=nxp/wifi_mod_para.conf

$ sysctl -w net.ipv6.conf.mlan0.accept_ra_rt_info_max_plen=64

$ sysctl -w net.ipv6.conf.mlan0.accept_ra_rt_info_min_plen=48

$ wpa_passphrase ${SSID} ${PASSWORD} > imxrouter.conf

$ wpa_supplicant -d -B -i mlan0 -c ./imxrouter.conf

$ udhcpc -i mlan0

$ systemctl restart radvd

$ sleep 4
```

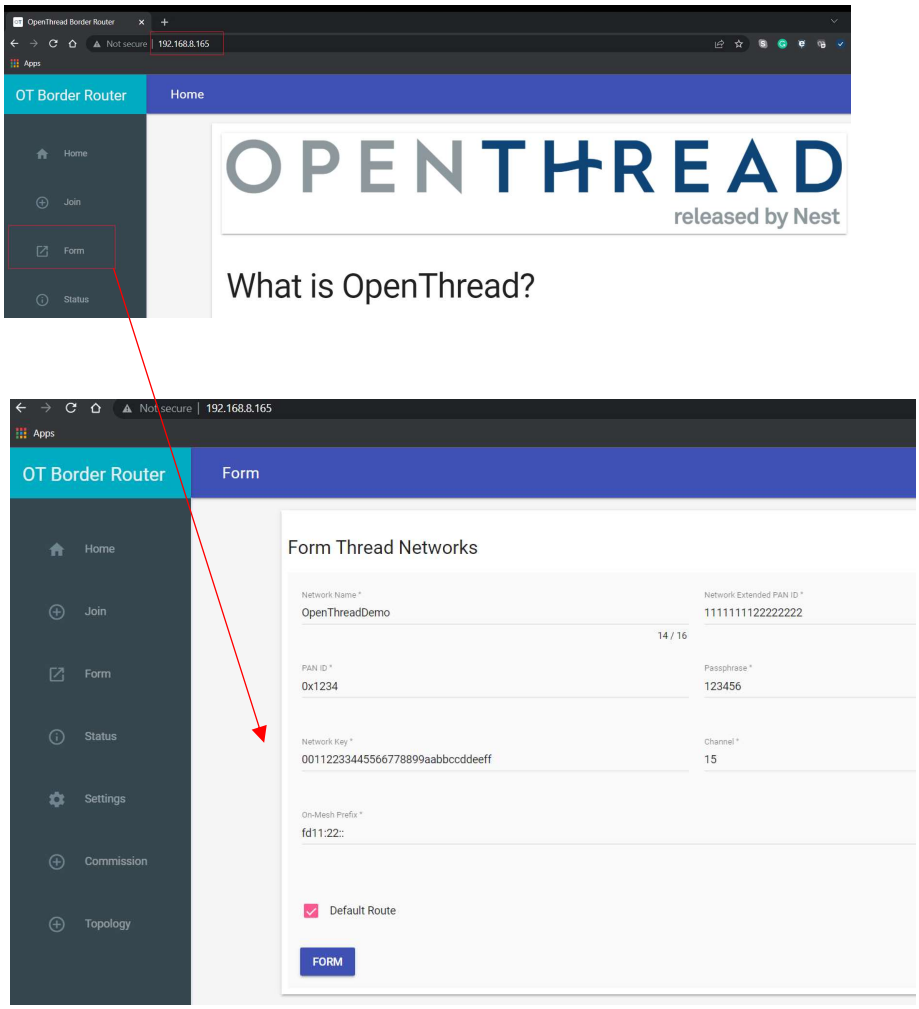
```
$ echo 1 > /proc/sys/net/ipv6/conf/all/forwarding
$ echo 1 > /proc/sys/net/ipv4/ip_forward
$ echo 2 > /proc/sys/net/ipv6/conf/all/accept_ra
$ ln -sf /usr/sbin/xtables-nft-multi /usr/sbin/ip6tables
$ ipset create -exist otbr-ingress-deny-src hash:net family inet6
$ ipset create -exist otbr-ingress-deny-src-swap hash:net family inet6
$ ipset create -exist otbr-ingress-allow-dst hash:net family inet6
$ ipset create -exist otbr-ingress-allow-dst-swap hash:net family inet6
```

- Setup OTBR

```
$ ./otbr-agent-spi -I wpan0 -B mlan0 'spinel+spi:///dev/spidev1.0?gpio-
resetdevice=/dev/gpiochip5&gpio-int-device=/dev/gpiochip5&gpio-int-line=12&gpio-
resetline=13&spi-mode=0&spi-speed=1000000&spi-reset-delay=0&skip-rcp-compatibility-
check=1' &
$ iptables -A FORWARD -i mlan0 -o wpan0 -j ACCEPT
$ iptables -A FORWARD -i wpan0 -o mlan0 -j ACCEPT
$ ip6tables -A FORWARD -i mlan0 -o wpan0 -j ACCEPT
$ ip6tables -A FORWARD -i wpan0 -o mlan0 -j ACCEPT
$ hciconfig hci0 up
$ other-web &
```

4.3 Form OTBR network

- To form the Thread Network – open the web interface Select the Form Window, configure the Thread Network and Press the ***Form*** Button:



- The alternative for the web interface is to set the Thread Network using command line interface:

```
$ ./ot-ctl dataset init new
$ ./ot-ctl dataset networkkey 00112233445566778899aabbccddeeff
$ ./ot-ctl dataset channel 15
$ ./ot-ctl dataset panid 0x1234
$ ./ot-ctl dataset extpanid 1111111122222222
$ ./ot-ctl dataset networkname Matter-NXP-1
$ ./ot-ctl prefix add fd08:b89:78:f372::/64 paos med
$ ./ot-ctl ifconfig up
$ ./ot-ctl thread start
$ ./ot-ctl state
```

5 I.MX8MM MATTER EXAMPLES

In the current Matter SDK for i.MX8MM platform we are providing reference examples for controller type application (CHIP-tool), OTA-provider app as well as end nodes, WiFi End Device type applications.

The examples are listed in the matter-> examples:

- CHIP Linux All-clusters Example
- CHIP Linux Lighting Example
- CHIP Linux Thermostat Example
- CHIP Linux CHIP-tool Example
- CHIP Linux OTA-provider Example

5.1 Matter application building instruction

The following build example will be based on the CHIP-tool app example:

- Set SDK environment variables:

```
$ export IMX_SDK_ROOT=/opt/fsl-imx-xwayland/5.15-kirkstone
```

- Start build environment by running the activate script:

```
$ source ./scripts/activate.sh
```

- Build the Chip-Tool by running the following command:

```
$./scripts/build/build_examples.py --target imx-chip-tool build
```

- Or you can build an end-node type app (e.g. Thermostat app):

```
$./scripts/build/imxlinux_examples.shexamples/nxp-thermostat/linux/ out/nxp-thermosta debug
```

```
connectedhomeip$  ./scripts/build/build_examples.py  --target imx-chip-tool build
2022-09-28 16:24:01 INFO    Building targets: imx-chip-tool
2022-09-28 16:24:01 INFO    Generating /home/razvan/Documents/EAR8/connectedhomeip/out/imx-chip-tool
2022-09-28 16:24:01 INFO    Generating imx-chip-tool
2022-09-28 16:24:02 INFO    Generating compile_commands took 4ms
2022-09-28 16:24:02 INFO    Done. Made 99 targets from 106 files in 206ms
```

```
2022-09-28 16:25:21 INFO    [478/484] stamp obj/third_party/connectedhomeip/third_party/jsoncpp/jsoncpp.stamp
2022-09-28 16:25:22 INFO    [479/484] ar libCHIP.a
2022-09-28 16:25:22 INFO    [480/484] stamp obj/third_party/connectedhomeip/examples/common/tracing/trace_handlers.stamp
2022-09-28 16:25:22 INFO    [481/484] ar chip-tool-utils.a
2022-09-28 16:25:23 INFO    [482/484] c++ obj/chip-tool.main.cpp.o
2022-09-28 16:25:28 INFO    [483/484] ld ./chip-tool
2022-09-28 16:25:28 INFO    [484/484] stamp obj/default.stamp
2022-09-28 16:25:28 INFO    Command ['ninja', '-C', '/home/razvan/Documents/EAR8/connectedhomeip/out/imx-chip-tool'] completed
connectedhomeip$
```

- After build is complete, the results can be found in

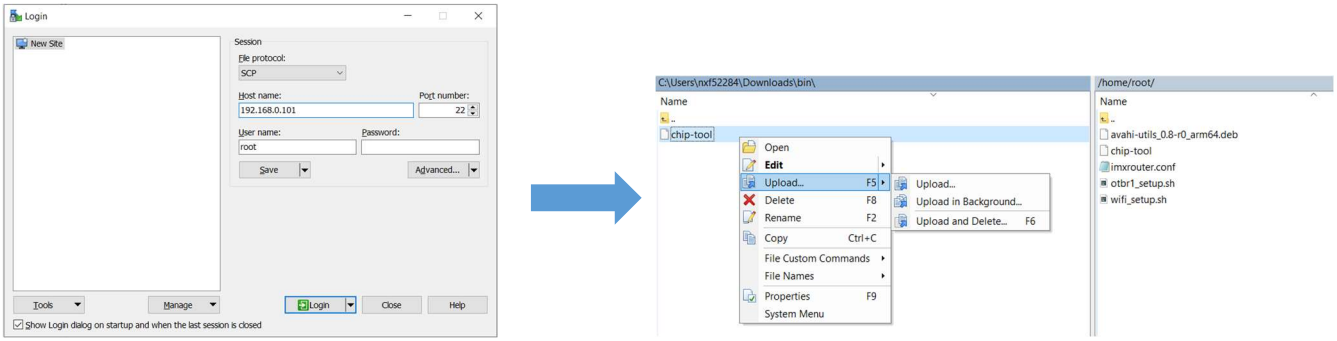
/connectedhomeip/out/imx-chip-tool/ folder

5.2 Deploying Matter application to the i.MX8MM board

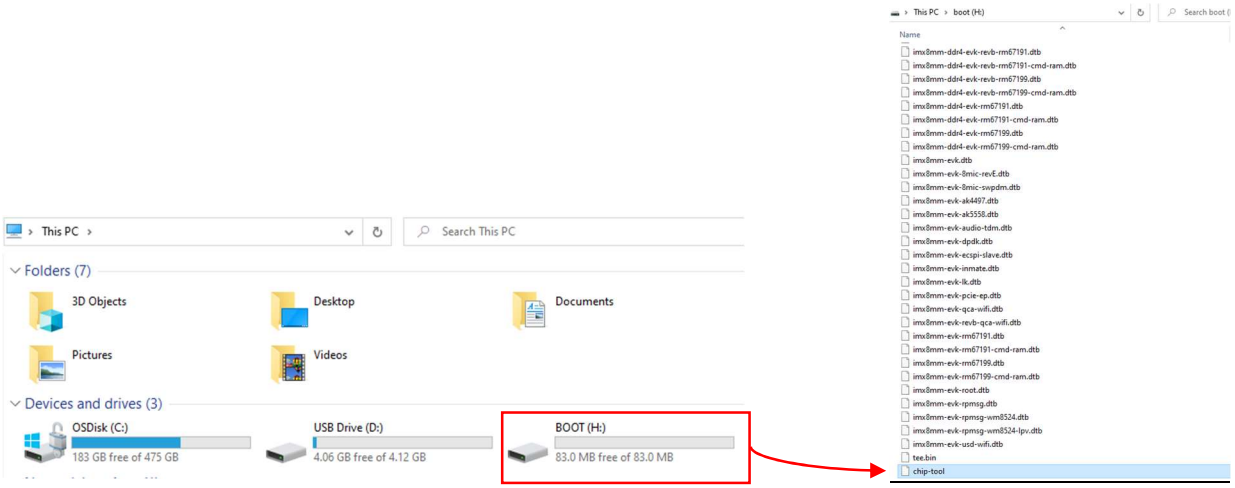
- The chip-tool application needs to be transferred to the i.MX8 Mini, this ca be either done by transferring it through a SCP transfer software

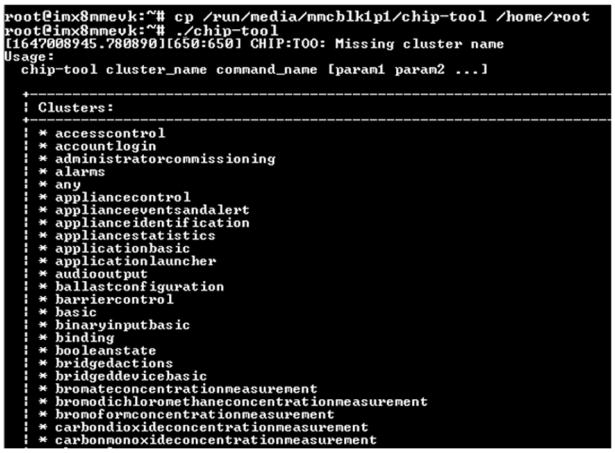
(e.g. WinSCP : https://winscp.net/eng/download.php) or by copying it in the SD card used to flash the i.MX8 Mini image)

- Transfer via SCP (for this example we will use Windows machine):
  - Connect the board to the same WiFi network with the PC that has chip-tool application
  - Discover the i.MX IPv4 (IP version 4) address using the command: $ ifconfig
  - Look for the IPv4 address in the **mlan0** interface
  - Open a new connection on the WinSCP application using the IPv4 address and the



- Transfer via uSD card:
  - Connect the uSD card to the PC using a card reader
  - You will see two new volumes added “USB Drive” and “BOOT”
  - Open “BOOT” and copy the chip-tool application there
  - Plug the uSD card back into the i.MX8 Mini board and boot it up
  - Inside the /home/root folder copy the chip-tool file from /run/media/mmcblk1p1/





6 MATTER NETWORK – CHIP TOOL COMISSIONGING AND CONTROL

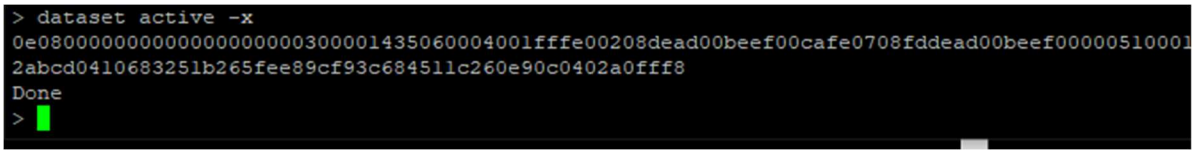
The prerequisite step for this part is to have the K32W0x1 setup as Matter light app. The steps for this setup can be found in "*Getting started with K32W0X1 in Matter*".

The following instructions will provide information on how to commission and run basic Matter Cluster commands from the CHIP-tool application built on the i.MX8MM platform:

Commissioning K32W0x1 over Bluetooth LE (Low Energy):

- Get the operational active dataset from the Thread Network. On the i.mx8m mini run the following command:

```
$ ot-ctl dataset active -x
```



- Activate the BLE (Bluetooth Low Energy) advertising on the K32W0x platform by pressing User Interface button;
- Run the following command to commission the device to the Thread Network:

```
$ ./chip-tool pairing ble-thread {NODE_ID_TO_ASSIGN} {ACTIVE_DATASET}
{SETUP_PIN_CODE}{DISCRIMINATOR}
```

Where:

- {NODE_ID_TO_ASSIGN} - the matter node id;
- {ACTIVE_DATASET} - Thread operational active dataset. The format is hex:xxxxxx).
- {SETUP_PIN_CODE} – pin code, default value is 20202021;

- {DISCRIMINATOR} – discriminator, default value is 3840;

Example:

```
$ ./chip-tool pairing ble-thread 1
hex:0e080000000000010000000300000f35060004001fffe00208111111112222222220708fd8e93c50a
ce6eae05100011223344556677889aabbccddeeff030e47265616444656d6f01021234041061e1206d2
c2b46e079eb775f41fc72190c0402a0fff8 20202021 3840
```

Sending cluster commands to K32W0x1:

These commands are sent from the imx8m mini shell terminal.

The format of the cluster command is:

```
$ ./chip-tool <cluster_name> <command_name> <param1, param2 ..>
```

Example of usage:

- send OnOff cluster ->Toggle command on endpoint 1

```
$ ./chip-tool onoff toggle 1 1
```

- send OnOff cluster ->On command on endpoint 1

```
$ ./chip-tool onoff on 1 1
```

- send OnOff cluster ->Off command on endpoint 1

```
$ ./chip-tool onoff off 1 1
```

- Send Attribute reporting configuration for on-off attribute using min/max interval = 300 seconds, cluster is on the endpoint 1

```
$ ./chip-tool onoff report on-off 300 301 1 1
```

- Read Channel Attribute from the Thread Diagnostic cluster

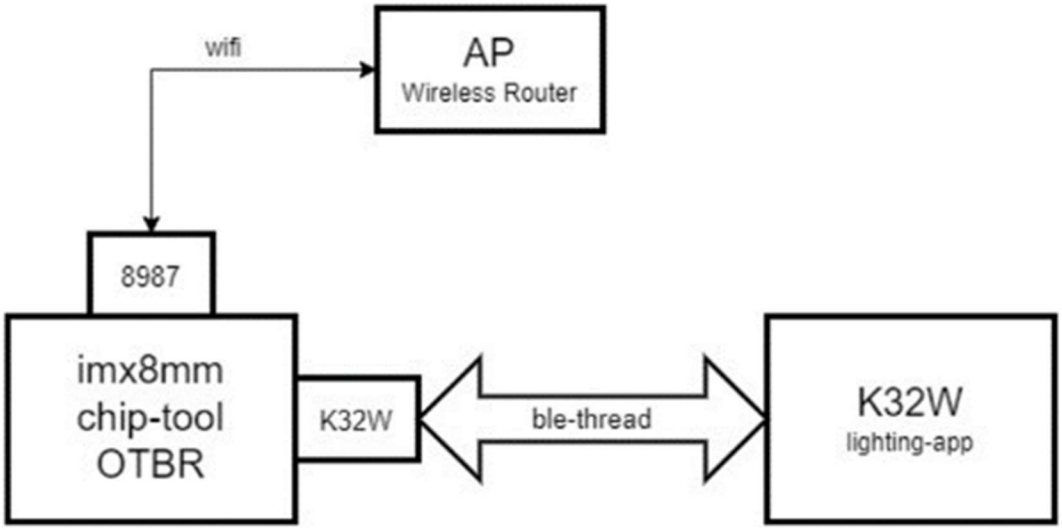
```
$ ./chip-tool threadnetworkdiagnostics read channel 1 0
```

- Read fabrics list based on the basic cluster

```
$ ./chip-tool operationalcredentials read fabrics-list 1 0
```

7 MATTER DEMO

7.1 Add a K32W end device in Matter Network



7.1.1 Pairing commissionee(Lighting node)tocommissioner(OTBR)

- Flash the K32W061 board with the lighting app (see K32W documntations for details)
- Press the reset button to power-cycle the board
- Press the SW2 button on OM15802, After 6 seconds will cause the device to reset it's persistent configuration and initiate a reboot.
- Press USERINTERFACE button to start BLE Advertising which is mandatory for device commissioning

NOTE: If the K32W061 device had been previously paired into a Matter network, it is recommended to first erase the external flash before pressing SW2 to perform the factory reset, some of the content from the external flash might impact the commissioning process. Run : DK6Programmer.exe-s COMxx -p k32w061dk6_EraseExtFlash.bin then input "1"on the UARTconsole to start "Erase Chip..." and wait untill you receive the "Chip Erased !!!" log printed.

- Pair the board to the i.MX8M MINI controller using the ble-thread commissioning option:

```
$ ./chip-tool pairing ble-thread 1
hex:0e080000000000010000000300000f35060004001fffe00208111111112222222220708fd8e93c50a
ce6eae05100011223344556677889aabbccddeeff030e47265616444656d6f01021234041061e1206d2
c2b46e079eb775f41fc72190c0402a0fff8 20202021 3840
```

Note: Normally the whole commissioning process will take approximately 20s before it complets. If there is “Device commissioning completed with success” message prompt on the chip-tool log, it means that the Matter device successfully joined the network, otherwise no further operation can be undertaken.

7.1.2 Controll the device

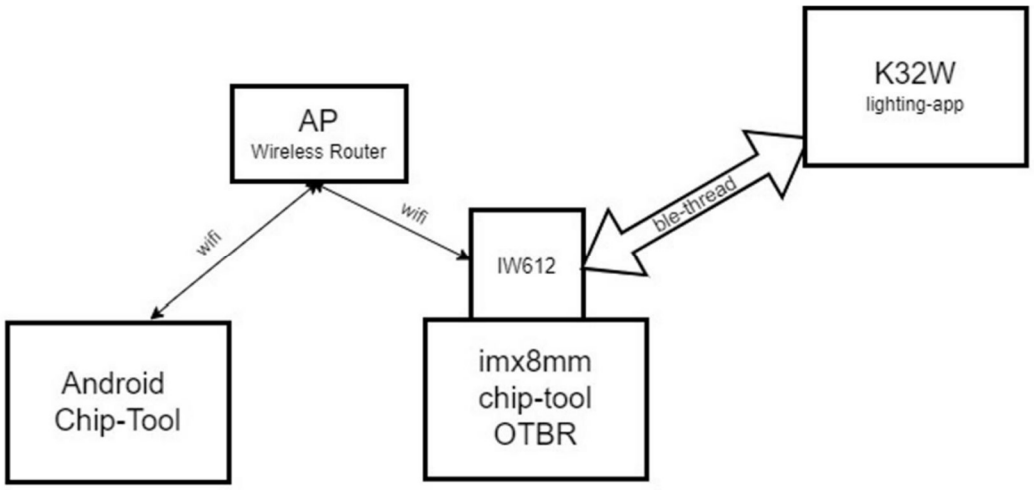
- Send an on/off command to the board:

```
$ chip-tool onoff toggle 1 1
```

- You will see the LED3 switching state between ON/OFF

7.2 Android CHIP-TOOL

In this demo an android smartphone is used as a controller, the i.MX8M MINI will act as OTBR and we will add the K32W as an end device in the Matter Network. (Android chip-tool build guide)



Perform the following checks before starting the demo:

- Check that your AP ha IPv6 capabilities and has the DHCPv6 server enabled
- Check if the WAN supports IPv6
- Check if the i.MX received an IPv6 address other than the one starting with **inet6 fe80::xxx:xxx:...**

```
$ ifconfig
mlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.3.169  netmask 255.255.254.0  broadcast 192.168.3.255
        inet6 2001:470:11e:f:c295:daff:fe00:e31f  prefixlen 64  scopeid
0x0<global>
        inet6 fe80::c295:daff:fe00:e31f  prefixlen 64  scopeid 0x20<link>
        ether c0:95:da:00:e3:1f  txqueuelen 1000  (Ethernet)
        RX packets 23  bytes 2888 (2.8 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 41  bytes 7853 (7.6 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

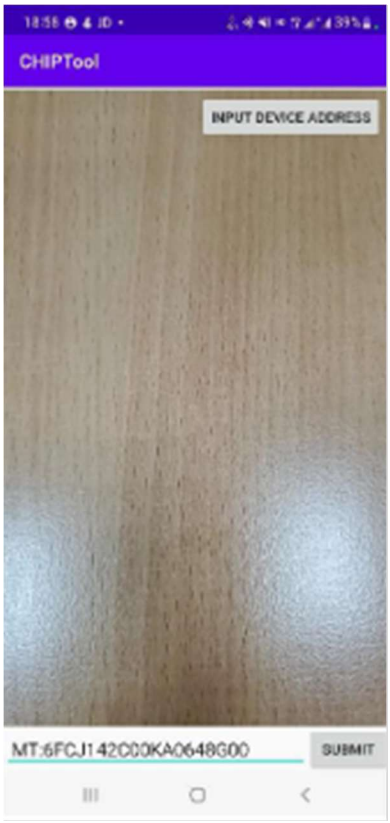
- Connect the smartphone to your PC and Install the chip-tool apk\

```
adb.exe install -r app-debug-nxp-v1.apk
```

- Factoryreset the K32W
- Choose the "**Provision chip device with thread**" in the app than scan the QR code
- The light app gives through the logs a link to the pairing QR code to be scanned, e.g.

https://project-chip.github.io/connectedhomeip/qrcode.html?data=MT%3A6FCJ142C00KA0648G00

- Another option is tu input the manual pairing code in the text box from the bottom of the screen



- Click submit, then input OTBR network information that were previously setup when starting the OTBR
- If the pairing succeeds you will be able to control the ON/OFF and Level Control clusters