Getting started with MW320 in 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 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 WiFi enabled platforms using the 88MW32x Micocontroller family.

The Wi-Fi® Microcontroller Platform provides a highly cost-effective, flexible, and easy-to-use hardware/software platform to build a new generation of smart connected devices delivering a broad-range of services to consumers including thermostats, appliances, lighting, home automation and remote access.

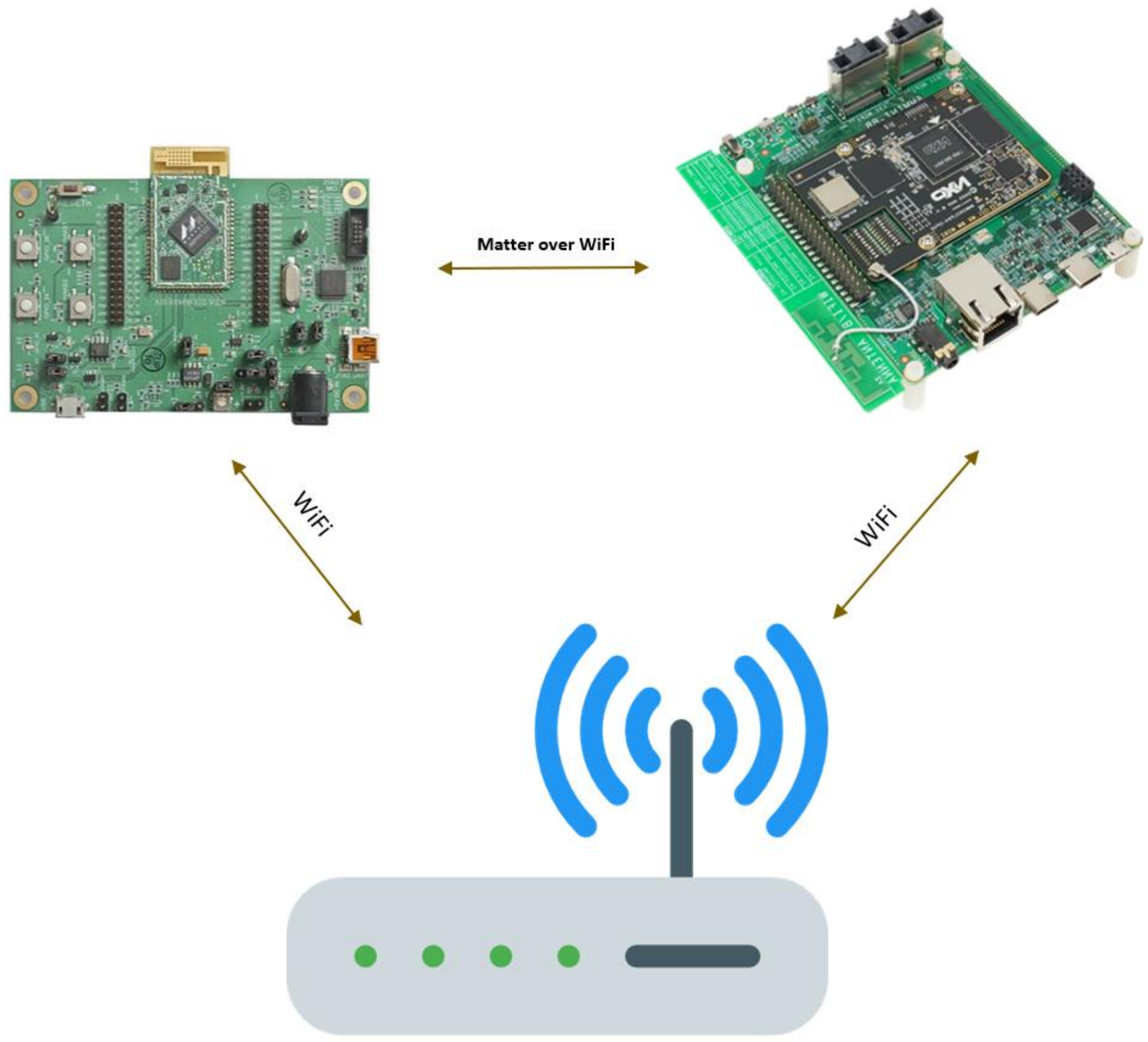
The platform is built using our high-performance 88MW32x Cortex-M4F Microcontroller and low power 802.11 b/g/n Wi-Fi® SoCs. It is powered by proven, field-tested EZ-Connect™ Software Development Kit (SDK.) This simplifies development and enables OEMs to focus on value-added features, and delivering applications and services to their customers.

https://www.nxp.com/part/MW320#/

2 HARDWARE SETUP

The minimum hardware required to create and run an end to end Matter setup with i.MX8MM is listed below:

- i.mx8m mini evk - acting as a Thread Border Router and Chip Controller: i.MX8MM-EVKB
- MW320 EVK – acting as Matter Accessory Device: MW320-EVK



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 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:

- 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:

1. On Windows 10, open PowerShell as administrator and run the following commands:

- Enable the Windows Subsystem for Linux:

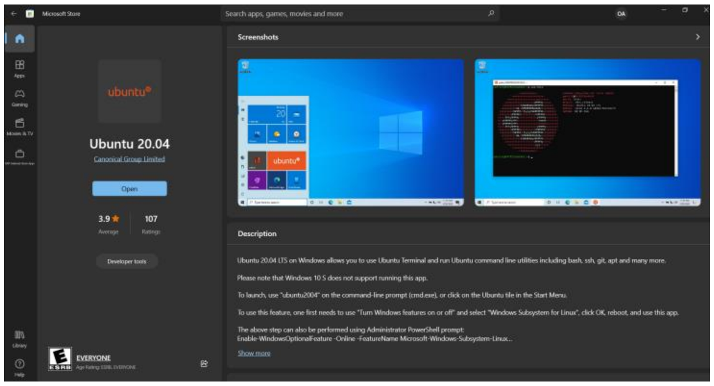
dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart

- 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



Ubuntu 20.04 LTS on Windows allows you to use Ubuntu Terminal and run Ubuntu command line utilities including bash, ssh, git, apt and many more.

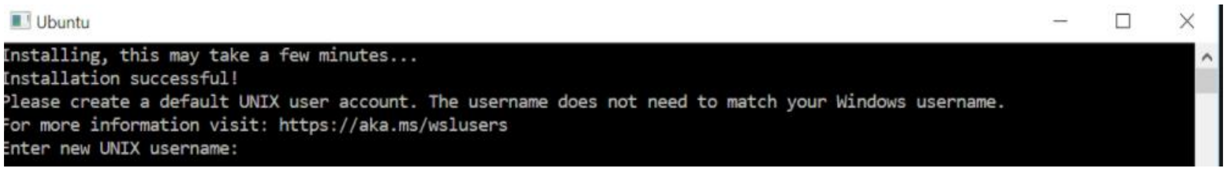
Please note that Windows 10 S does not support running this app.

To launch, use "ubuntu2004" on the command-line prompt (cmd.exe), or click on the Ubuntu tile in the Start Menu.

To use this feature, one first needs to use "Turn Windows features on or off" and select "Windows Subsystem for Linux", click OK, reboot, and use this app.

The above step can also be performed using Administrator PowerShell prompt:
Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Windows-Subsystem-Linux...

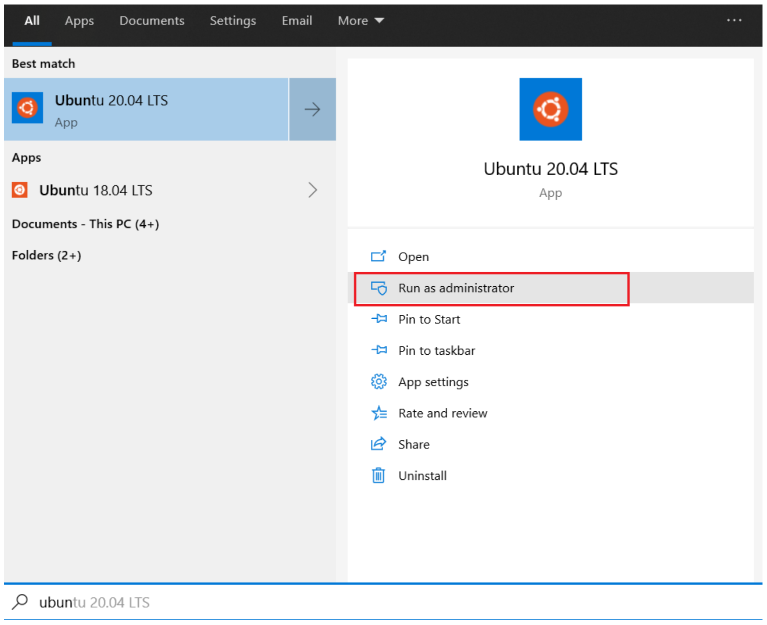
- Create a user account and password for your new Linux distribution



```
Installing, this may take a few minutes...
Installation successful!
Please create a default UNIX user account. The username does not need to match your Windows username.
For more information visit: https://aka.ms/wslusers
Enter new UNIX username:
```

- 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.

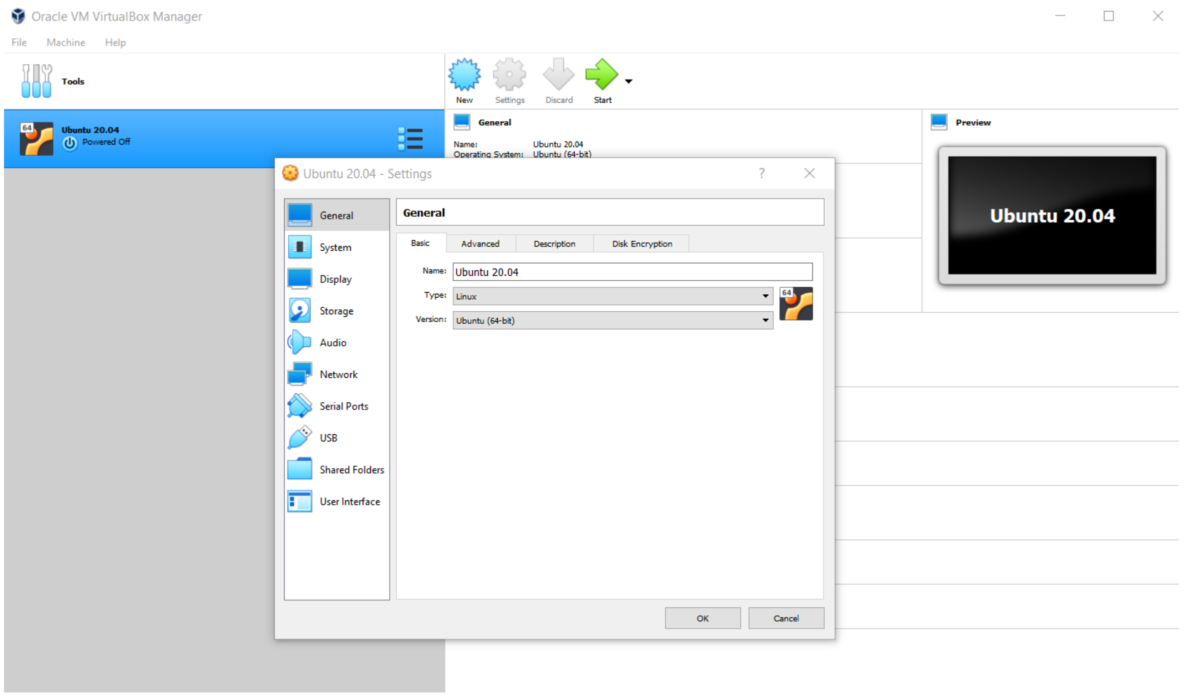


3.2 Linux Virtual Machine

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 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;



Name: Ubuntu 20.04
Operating System: Ubuntu (64-bit)

General

Name: Ubuntu 20.04
Type: Linux
Version: Ubuntu (64-bit)

3.3 Setting up Matter Environment

The following steps guide you through creating Matter build environment.

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

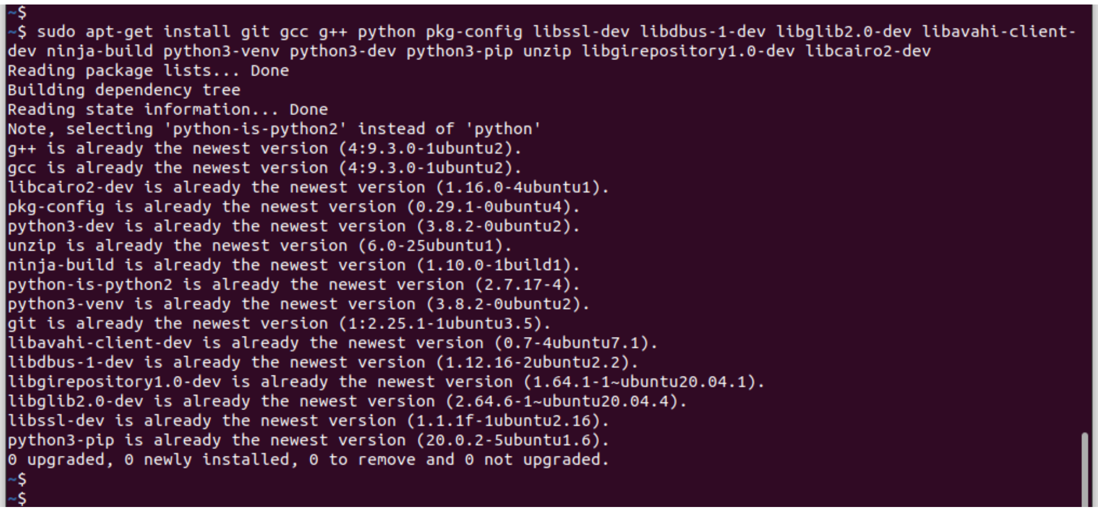
$ sudo apt update

$ 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



```
~$
~$ 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
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'python-is-python2' instead of 'python'
g++ is already the newest version (4:9.3.0-1ubuntu2).
gcc is already the newest version (4:9.3.0-1ubuntu2).
libcairo2-dev is already the newest version (1.16.0-4ubuntu1).
pkg-config is already the newest version (0.29.1-0ubuntu4).
python3-dev is already the newest version (3.8.2-0ubuntu2).
unzip is already the newest version (6.0-25ubuntu1).
ninja-build is already the newest version (1.10.0-1build1).
python-is-python2 is already the newest version (2.7.17-4).
python3-venv is already the newest version (3.8.2-0ubuntu2).
git is already the newest version (1:2.25.1-1ubuntu3.5).
libavahi-client-dev is already the newest version (0.7-4ubuntu7.1).
libdbus-1-dev is already the newest version (1.12.16-2ubuntu2.2).
libgirepository1.0-dev is already the newest version (1.64.1-1~ubuntu20.04.1).
libglib2.0-dev is already the newest version (2.64.6-1~ubuntu20.04.4).
libssl-dev is already the newest version (1.1.1f-1ubuntu2.16).
python3-pip is already the newest version (20.0.2-5ubuntu1.6).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
~$
~$
```

   - Restart the Linux machine.

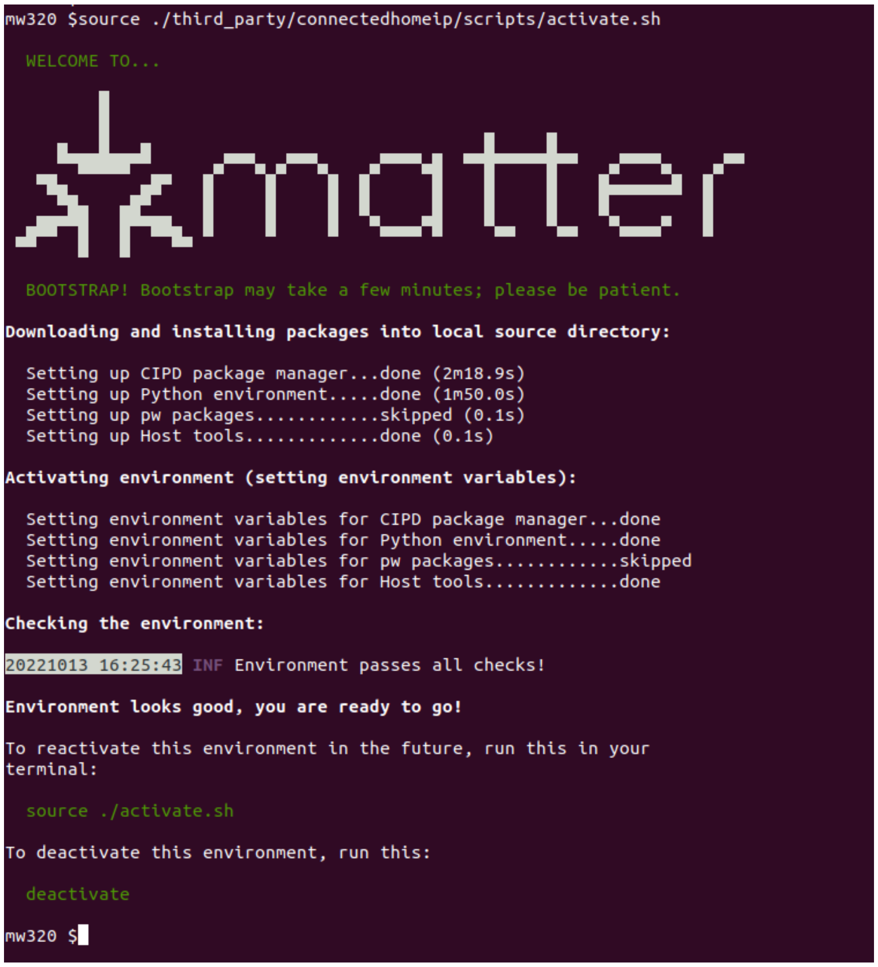
2. Matter Building Setup instructions:

- Clone the Matter SDK using the public repo:

$ git clone https://github.com/nxptest/connectedhomeip
$ cd connectedhomeip
$ git checkout tags/Ref_kit_v0.92

- Start build environment by running the activate script:

$ source ./scripts/activate.sh



```
mw320 $source ./third_party/connectedhomeip/scripts/activate.sh

  WELCOME TO...

  BOOTSTRAP! Bootstrap may take a few minutes; please be patient.

Downloading and installing packages into local source directory:

  Setting up CIPD package manager...done (2m18.9s)
  Setting up Python environment.....done (1m50.0s)
  Setting up pw packages...........skipped (0.1s)
  Setting up Host tools............done (0.1s)

Activating environment (setting environment variables):

  Setting environment variables for CIPD package manager...done
  Setting environment variables for Python environment.....done
  Setting environment variables for pw packages...........skipped
  Setting environment variables for Host tools............done

Checking the environment:

20221013 16:25:43 INF Environment passes all checks!

Environment looks good, you are ready to go!

To reactivate this environment in the future, run this in your
terminal:

  source ./activate.sh

To deactivate this environment, run this:

  deactivate

mw320 $
```

4 MW320 MATTER EXAMPLES

In the current Matter SDK for MW320 platform we are providing an implementation for the all-clusters example that was intended to work in different setups as either light node, light switch node or both.

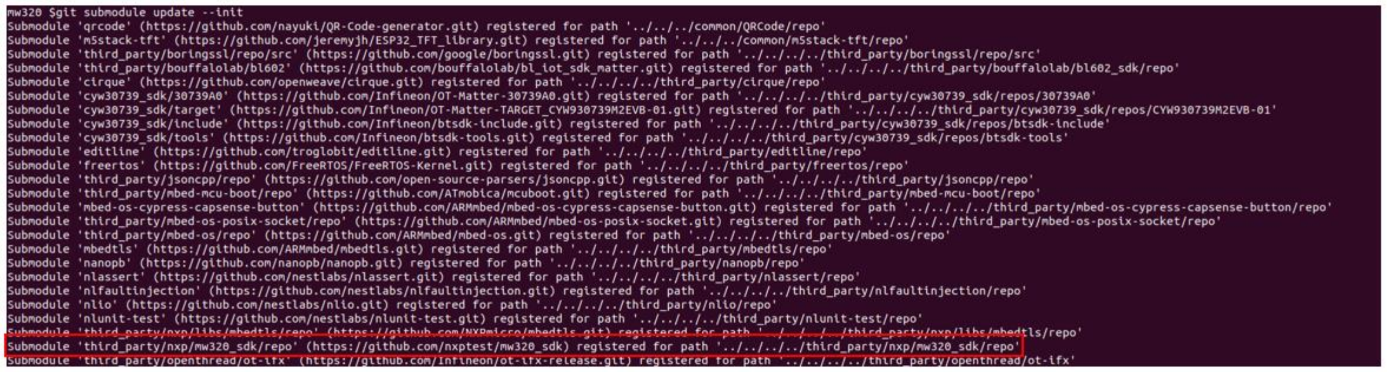
4.1 Matter application building instruction

The following build example will be based on the all-clusters app example:

- Change the directory to the all-clusters application base directory and run the following command in order to download the MW320 SDK as a submodule:

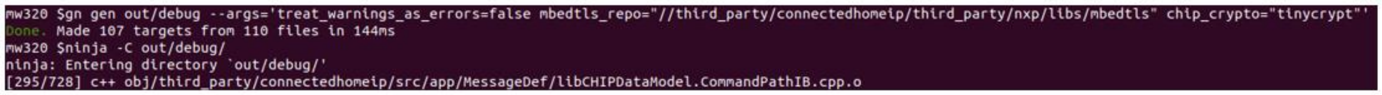
$ cd examples/all-clusters-app/nxp/mw320

$ git submodule update --init

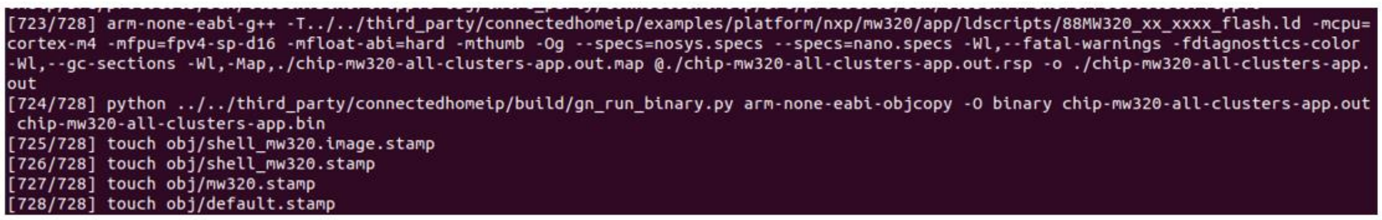


```
mw320 $git submodule update --init
Submodule 'qrcode' (https://github.com/nayuki/QR-Code-generator.git) registered for path '../../../common/QRCode/repo'
Submodule 'm5stack-tft' (https://github.com/jeremyjh/ESP32_TFT_library.git) registered for path '../../../common/m5stack-tft/repo'
Submodule 'third_party/boringssl/repo/src' (https://github.com/google/boringssl.git) registered for path '../../../../third_party/boringssl/repo/src'
Submodule 'third_party/bouffalolab/bl602' (https://github.com/bouffalolab/bl_iot_sdk_matter.git) registered for path '../../../../third_party/bouffalolab/bl602_sdk/repo'
Submodule 'cirque' (https://github.com/openweave/cirque.git) registered for path '../../../../third_party/cirque/repo'
Submodule 'cyw30739_sdk/30739A0' (https://github.com/Infineon/OT-Matter-30739A0.git) registered for path '../../../../third_party/cyw30739_sdk/repos/30739A0'
Submodule 'cyw30739_sdk/target' (https://github.com/Infineon/OT-Matter-TARGET_CYW930739M2EVB-01.git) registered for path '../../../../third_party/cyw30739_sdk/repos/CYW930739M2EVB-01'
Submodule 'cyw30739_sdk/include' (https://github.com/Infineon/btsdk-include.git) registered for path '../../../../third_party/cyw30739_sdk/repos/btsdk-include'
Submodule 'cyw30739_sdk/tools' (https://github.com/Infineon/btsdk-tools.git) registered for path '../../../../third_party/cyw30739_sdk/repos/btsdk-tools'
Submodule 'editline' (https://github.com/troglobit/editline.git) registered for path '../../../../third_party/editline/repo'
Submodule 'freertos' (https://github.com/FreeRTOS/FreeRTOS-Kernel.git) registered for path '../../../../third_party/freertos/repo'
Submodule 'third_party/jsoncpp/repo' (https://github.com/open-source-parsers/jsoncpp.git) registered for path '../../../../third_party/jsoncpp/repo'
Submodule 'third_party/mbed-mcu-boot/repo' (https://github.com/ATmobica/mcuboot.git) registered for path '../../../../third_party/mbed-mcu-boot/repo'
Submodule 'mbed-os-cypress-capsense-button' (https://github.com/ARMmbed/mbed-os-cypress-capsense-button.git) registered for path '../../../../third_party/mbed-os-cypress-capsense-button/repo'
Submodule 'third_party/mbed-os-posix-socket/repo' (https://github.com/ARMmbed/mbed-os-posix-socket.git) registered for path '../../../../third_party/mbed-os-posix-socket/repo'
Submodule 'third_party/mbed-os/repo' (https://github.com/ARMmbed/mbed-os.git) registered for path '../../../../third_party/mbed-os/repo'
Submodule 'mbedtls' (https://github.com/ARMmbed/mbedtls.git) registered for path '../../../../third_party/mbedtls/repo'
Submodule 'nanopb' (https://github.com/nanopb/nanopb.git) registered for path '../../../../third_party/nanopb/repo'
Submodule 'nlassert' (https://github.com/nestlabs/nlassert.git) registered for path '../../../../third_party/nlassert/repo'
Submodule 'nlfaultinjection' (https://github.com/nestlabs/nlfaultinjection.git) registered for path '../../../../third_party/nlfaultinjection/repo'
Submodule 'nlio' (https://github.com/nestlabs/nlio.git) registered for path '../../../../third_party/nlio/repo'
Submodule 'nlunit-test' (https://github.com/nestlabs/nlunit-test.git) registered for path '../../../../third_party/nlunit-test/repo'
Submodule 'third_party/nxp/libs/mbedtls/repo' (https://github.com/NXPmicro/mbedtls.git) registered for path '../../../../third_party/nxp/libs/mbedtls/repo'
Submodule 'third_party/nxp/mw320_sdk/repo' (https://github.com/nxptest/mw320_sdk) registered for path '../../../../third_party/nxp/mw320_sdk/repo'
Submodule 'third_party/openthread/ot-ifx' (https://github.com/Infineon/ot-ifx-release.git) registered for path '../../../../third_party/openthread/ot-ifx'
```

- Generate the build files using GN and start building the application image with the Ninja tool



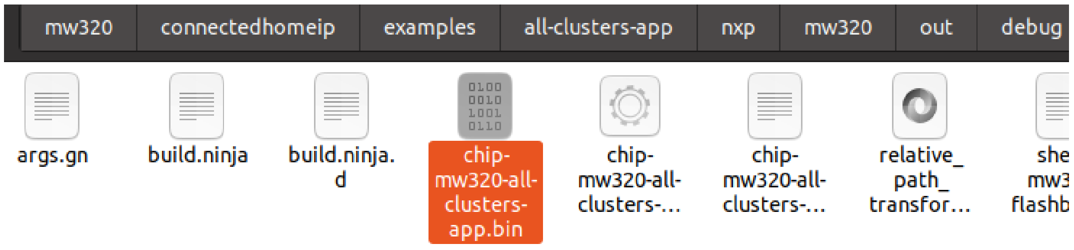
```
mw320 $gn gen out/debug --args='treat_warnings_as_errors=false mbedtls_repo="//third_party/connectedhomeip/third_party/nxp/libs/mbedtls" chip_crypto="tinycrypt"'
Done. Made 107 targets from 110 files in 144ms
mw320 $ninja -C out/debug/
ninja: Entering directory `out/debug/'
[295/728] c++ obj/third_party/connectedhomeip/src/app/MessageDef/libCHIPDataModel.CommandPathIB.cpp.o
```



```
[723/728] arm-none-eabi-g++ -T../../third_party/connectedhomeip/examples/platform/nxp/mw320/app/ldscripts/88MW320_xx_xxxx_flash.ld -mcpu=
cortex-m4 -mfpu=fpv4-sp-d16 -mfloat-abi=hard -mthumb -Og --specs=nosys.specs --specs=nano.specs -Wl,--fatal-warnings -fdiagnostics-color
-Wl,--gc-sections -Wl,-Map,./chip-mw320-all-clusters-app.out.map @./chip-mw320-all-clusters-app.out.rsp -o ./chip-mw320-all-clusters-app.
out
[724/728] python ../../third_party/connectedhomeip/build/gn_run_binary.py arm-none-eabi-objcopy -O binary chip-mw320-all-clusters-app.out
 chip-mw320-all-clusters-app.bin
[725/728] touch obj/shell_mw320.image.stamp
[726/728] touch obj/shell_mw320.stamp
[727/728] touch obj/mw320.stamp
[728/728] touch obj/default.stamp
```

- The built application can be found in the

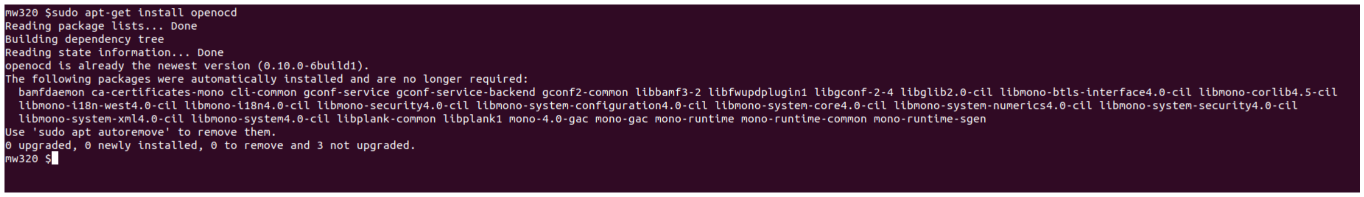
/out/debug/ folder


mw320
connectedhomeip
examples
all-clusters-app
nxp
mw320
out
debug
args.gn
build.ninja
build.ninja.d
chip-mw320-all-clusters-app.bin
chip-mw320-all-clusters-...
chip-mw320-all-clusters-...
relative_path_transfor...
she mw3 flashb

4.2 Matter application building instruction

- In order to flash the board the OpenOCD (Open On-Chip Debugger) package

$ sudo apt-get install openocd



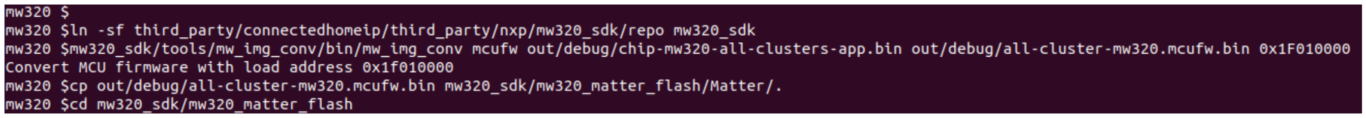
```
mw320 $sudo apt-get install openocd
Reading package lists... Done
Building dependency tree
Reading state information... Done
openocd is already the newest version (0.10.0-6build1).
The following packages were automatically installed and are no longer required:
  bamfdaemon ca-certificates-mono cli-common gconf-service gconf-service-backend gconf2-common libbamf3-2 libfwupdplugin1 libgconf-2-4 libglib2.0-cil libmono-btls-interface4.0-cil libmono-corlib4.5-cil
  libmono-i18n-west4.0-cil libmono-i18n4.0-cil libmono-security4.0-cil libmono-system-configuration4.0-cil libmono-system-core4.0-cil libmono-system-numerics4.0-cil libmono-system-security4.0-cil
  libmono-system-xml4.0-cil libmono-system4.0-cil libplank-common libplank1 mono-4.0-gac mono-gac mono-runtime mono-runtime-common mono-runtime-sgen
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 3 not upgraded.
mw320 $
```

- After that the MW320 firmware download image must be prepared:

$ ln -sf third_party/connectedhomeip/third_party/nxp/mw320_sdk/repo mw320_sdk

$ mw320_sdk/tools/mw_img_conv/bin/mw_img_conv mcufw out/debug/chip-mw320-all-clusters-app.bin out/debug/all-cluster-mw320.mcufw.bin 0x1F010000

$ cp out/debug/all-cluster-mw320.mcufw.bin mw320_sdk/mw320_matter_flash/Matter/.

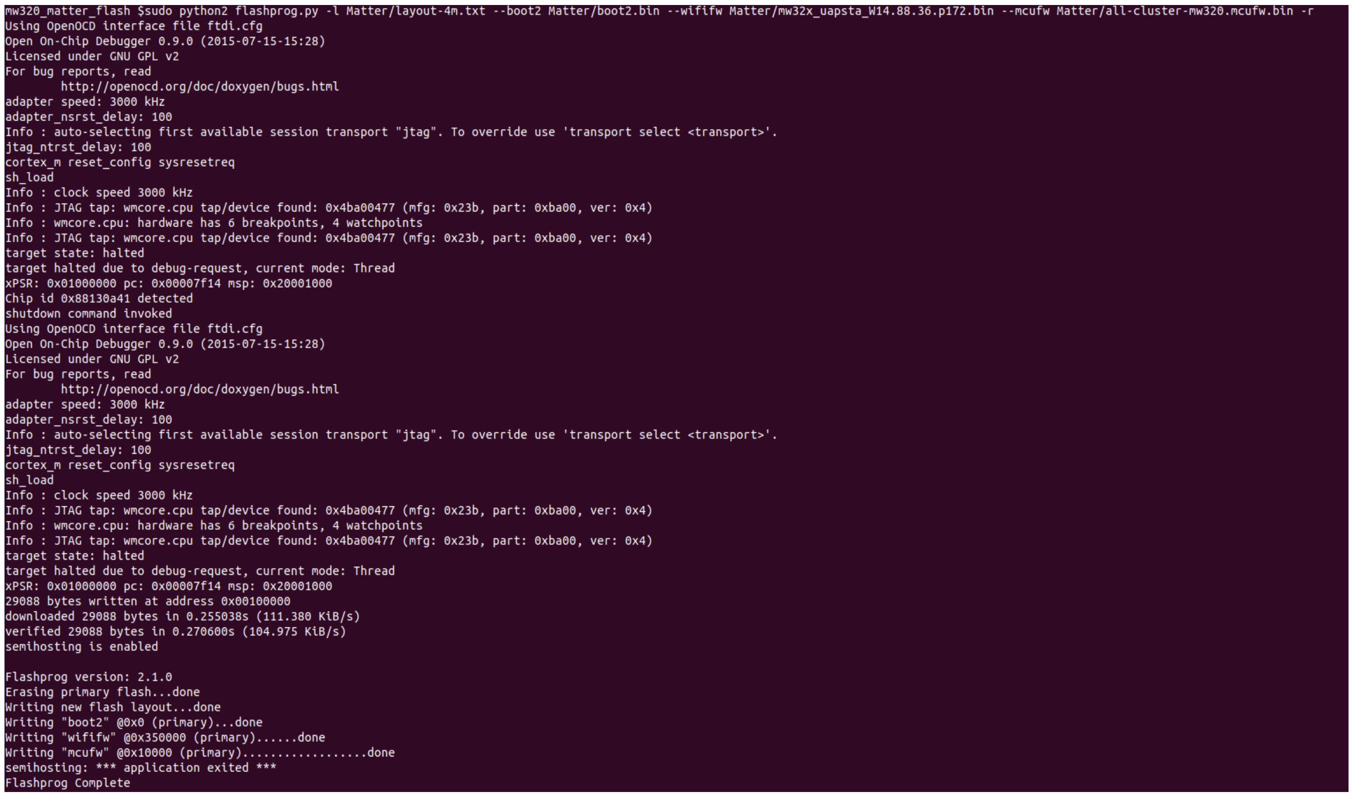


```
mw320 $
mw320 $ln -sf third_party/connectedhomeip/third_party/nxp/mw320_sdk/repo mw320_sdk
mw320 $mw320_sdk/tools/mw_img_conv/bin/mw_img_conv mcufw out/debug/chip-mw320-all-clusters-app.bin out/debug/all-cluster-mw320.mcufw.bin 0x1F010000
Convert MCU firmware with load address 0x1f010000
mw320 $cp out/debug/all-cluster-mw320.mcufw.bin mw320_sdk/mw320_matter_flash/Matter/.
mw320 $cd mw320_sdk/mw320_matter_flash
```

- Flashing the firmware image to MW320:

$ cd mw320_sdk/mw320_matter_flash

$ sudo python2 flashprog.py -l Matter/layout-4m.txt --boot2 Matter/boot2.bin --wififw Matter/mw32x_uapsta_W14.88.36.p172.bin --mcufw Matter/all-cluster-mw320.mcufw.bin -r



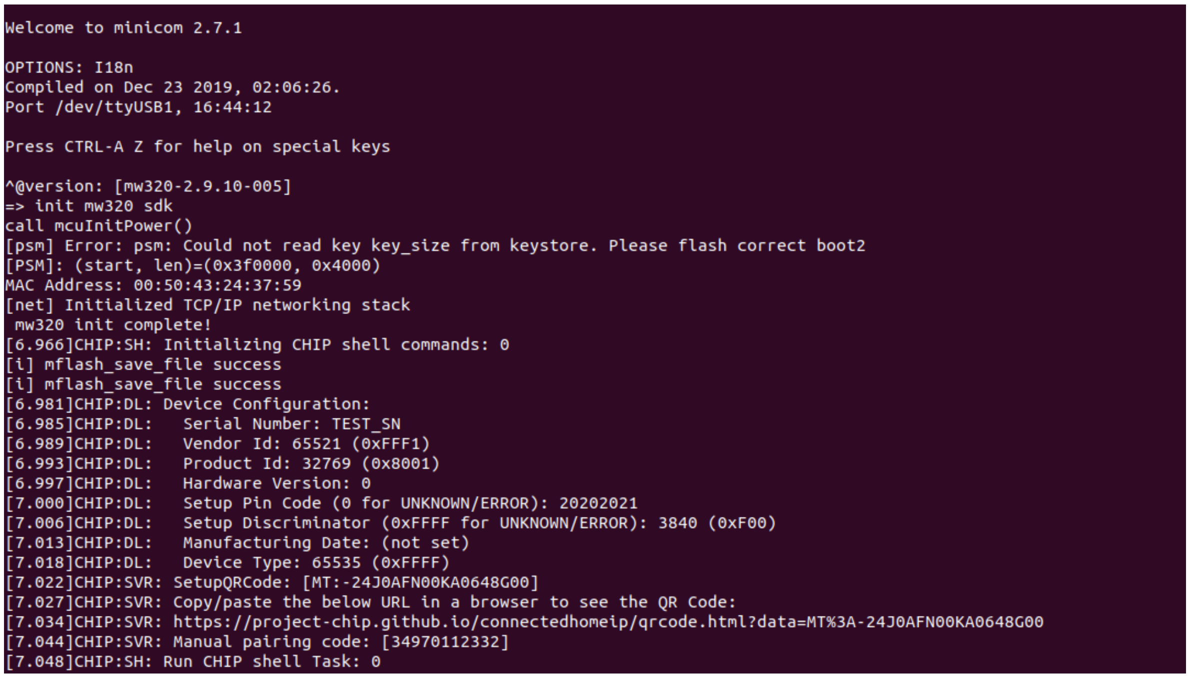
```
mw320_matter_flash $sudo python2 flashprog.py -l Matter/layout-4m.txt --boot2 Matter/boot2.bin --wififw Matter/mw32x_uapsta_W14.88.36.p172.bin --mcufw Matter/all-cluster-mw320.mcufw.bin -r
Using OpenOCD interface file ftdi.cfg
Open On-Chip Debugger 0.9.0 (2015-07-15-15:28)
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.org/doc/doxygen/bugs.html
adapter speed: 3000 kHz
adapter_nsrst_delay: 100
Info : auto-selecting first available session transport "jtag". To override use 'transport select <transport>'.
jtag_ntrst_delay: 100
cortex_m reset_config sysresetreq
sh_load
Info : clock speed 3000 kHz
Info : JTAG tap: wmcore.cpu tap/device found: 0x4ba00477 (mfg: 0x23b, part: 0xba00, ver: 0x4)
Info : wmcore.cpu: hardware has 6 breakpoints, 4 watchpoints
Info : JTAG tap: wmcore.cpu tap/device found: 0x4ba00477 (mfg: 0x23b, part: 0xba00, ver: 0x4)
target state: halted
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x00007f14 msp: 0x20001000
Chip id 0x88130a41 detected
shutdown command invoked
Using OpenOCD interface file ftdi.cfg
Open On-Chip Debugger 0.9.0 (2015-07-15-15:28)
Licensed under GNU GPL v2
For bug reports, read
        http://openocd.org/doc/doxygen/bugs.html
adapter speed: 3000 kHz
adapter_nsrst_delay: 100
Info : auto-selecting first available session transport "jtag". To override use 'transport select <transport>'.
jtag_ntrst_delay: 100
cortex_m reset_config sysresetreq
sh_load
Info : clock speed 3000 kHz
Info : JTAG tap: wmcore.cpu tap/device found: 0x4ba00477 (mfg: 0x23b, part: 0xba00, ver: 0x4)
Info : wmcore.cpu: hardware has 6 breakpoints, 4 watchpoints
Info : JTAG tap: wmcore.cpu tap/device found: 0x4ba00477 (mfg: 0x23b, part: 0xba00, ver: 0x4)
target state: halted
target halted due to debug-request, current mode: Thread
xPSR: 0x01000000 pc: 0x00007f14 msp: 0x20001000
29088 bytes written at address 0x00100000
downloaded 29088 bytes in 0.255038s (111.380 KiB/s)
verified 29088 bytes in 0.270600s (104.975 KiB/s)
semihosting is enabled

Flashprog version: 2.1.0
Erasing primary flash...done
Writing new flash layout...done
Writing "boot2" @0x0 (primary)...done
Writing "wififw" @0x350000 (primary)......done
Writing "mcufw" @0x10000 (primary).................done
semihosting: *** application exited ***
Flashprog Complete
```

- To catch the application logs a terminal connection needs to be opened to the board:

$ sudo apt-get install minicom

$ TERM=linux minicom -D /dev/ttyUSB1 -b 115200



```
Welcome to minicom 2.7.1

OPTIONS: I18n
Compiled on Dec 23 2019, 02:06:26.
Port /dev/ttyUSB1, 16:44:12

Press CTRL-A Z for help on special keys

^@version: [mw320-2.9.10-005]
=> init mw320 sdk
call mcuInitPower()
[psm] Error: psm: Could not read key key_size from keystore. Please flash correct boot2
[PSM]: (start, len)=(0x3f0000, 0x4000)
MAC Address: 00:50:43:24:37:59
[net] Initialized TCP/IP networking stack
 mw320 init complete!
[6.966]CHIP:SH: Initializing CHIP shell commands: 0
[i] mflash_save_file success
[i] mflash_save_file success
[6.981]CHIP:DL: Device Configuration:
[6.985]CHIP:DL:   Serial Number: TEST_SN
[6.989]CHIP:DL:   Vendor Id: 65521 (0xFFF1)
[6.993]CHIP:DL:   Product Id: 32769 (0x8001)
[6.997]CHIP:DL:   Hardware Version: 0
[7.000]CHIP:DL:   Setup Pin Code (0 for UNKNOWN/ERROR): 20202021
[7.006]CHIP:DL:   Setup Discriminator (0xFFFF for UNKNOWN/ERROR): 3840 (0xF00)
[7.013]CHIP:DL:   Manufacturing Date: (not set)
[7.018]CHIP:DL:   Device Type: 65535 (0xFFFF)
[7.022]CHIP:SVR: SetupQRCode: [MT:-24J0AFN00KA0648G00]
[7.027]CHIP:SVR: Copy/paste the below URL in a browser to see the QR Code:
[7.034]CHIP:SVR: https://project-chip.github.io/connectedhomeip/qrcode.html?data=MT%3A-24J0AFN00KA0648G00
[7.044]CHIP:SVR: Manual pairing code: [34970112332]
[7.048]CHIP:SH: Run CHIP shell Task: 0
```

5 MATTER NETWORK – TEST SETUP INSTRUCTIONS

The prerequisite step for this part is to have the i.MX8MM as Matter controller. The steps for this setup can be found in "*Getting started with i.MX8MM in Matter*".

The following instructions will provide information on how to for a matter test setup containing one controller an one end-nodes: light node

i.MX8 Mini Matter Controller Setup

- Connect the I.mx8 mini to the Wi-Fi AP network

$modprobe moal mod_para=nxp/wifi_mod_para.conf

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

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

$udhcpc -i mlan0

$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

Commission the MW320 board over WiFi

- Connect the MW320 board to the AP

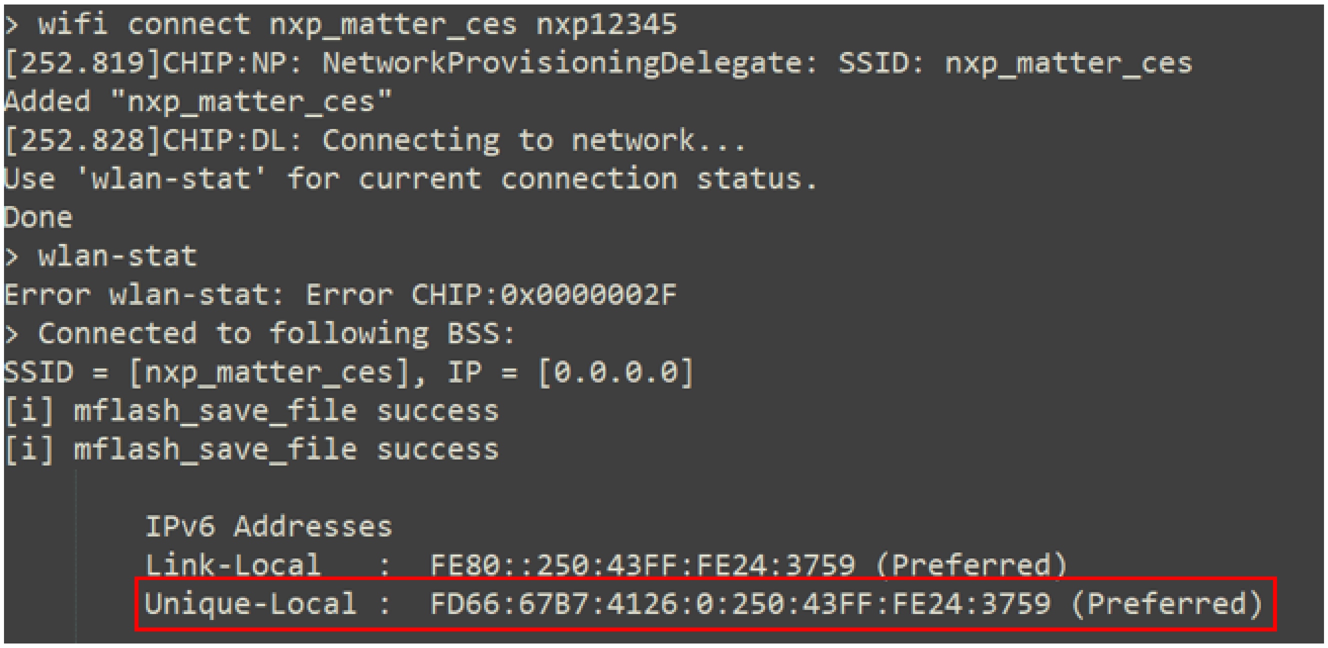
>wifi connect SSID PASSWORD

- Run the following command to commission the MW320 app over WiFi using the IPv6 global address:

$ chip-tool pairing ethernet {NODE_ID_TO_ASSIGN}{SETUP_PIN_CODE}{DISCRIMINATOR} {IPv6_ADDR}{PORT}

Where:

- ${NODE_ID_TO_ASSIGN} - the matter node id;
- ${SETUP_PIN_CODE} – pin code, default value is 20202021;
- ${DISCRIMINATOR} – discriminator, default value is 3840;
- ${IPv6_ADDR} – The global IPv6 address assigned to the MW320 board by the AP's DHCPv6 server
- ${PORT} – Web port where the all-clusters app server is opened, default value is 5540


```
> wifi connect nxp_matter_ces nxp12345
[252.819]CHIP:NP: NetworkProvisioningDelegate: SSID: nxp_matter_ces
Added "nxp_matter_ces"
[252.828]CHIP:DL: Connecting to network...
Use 'wlan-stat' for current connection status.
Done
> wlan-stat
Error wlan-stat: Error CHIP:0x0000002F
> Connected to following BSS:
SSID = [nxp_matter_ces], IP = [0.0.0.0]
[i] mflash_save_file success
[i] mflash_save_file success

        IPv6 Addresses
        Link-Local   :  FE80::250:43FF:FE24:3759 (Preferred)
        Unique-Local :  FD66:67B7:4126:0:250:43FF:FE24:3759 (Preferred)
```

The MW320 all-clusters application is intended to work as either a lighting device or a light-switch device. For this setup we will use it as a light end node

- You can send On/Off cluster commands to control the GPIO41 LED by the following commands

To send OnOff cluster ->Toggle command on endpoint 1 :

$ ./chip-tool onoff toggle {NODE_ID} 1

To send OnOff cluster ->On command on endpoint 1 :

$ ./chip-tool onoff on {NODE_ID} 1

To send OnOff cluster ->Off command on endpoint 1 :

$ ./chip-tool onoff off {NODE_ID} 1

To 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 {NODE_ID} 1