

How to setup and work with ESP-IDF and ESP-Matter

URL: <https://github.com/mozolin/matter-thread>

Windows Powershell (with Administrator rights):

~~~

`systeminfo`

~~~

OS Name: Microsoft Windows 10 Pro

OS Version: 10.0.19045 N/A Build 19045

Works successfully in this version of Windows!

~~~

`systeminfo`

~~~

OS Name: Microsoft Windows 10 Enterprise

OS Version: 10.0.19045 N/A Build 19045

Works without sharing COM-ports in this version of Windows!

1. How to install Ubuntu on Windows with WSL

Recommended Ubuntu 22.04...

<https://learn.microsoft.com/en-us/windows/wsl/install>

Check Ubuntu version:

~~~

`lsb_release -a`

~~~

If asked to add a new user, just do it, but after that we should switch to root.

Windows Powershell:

~~~

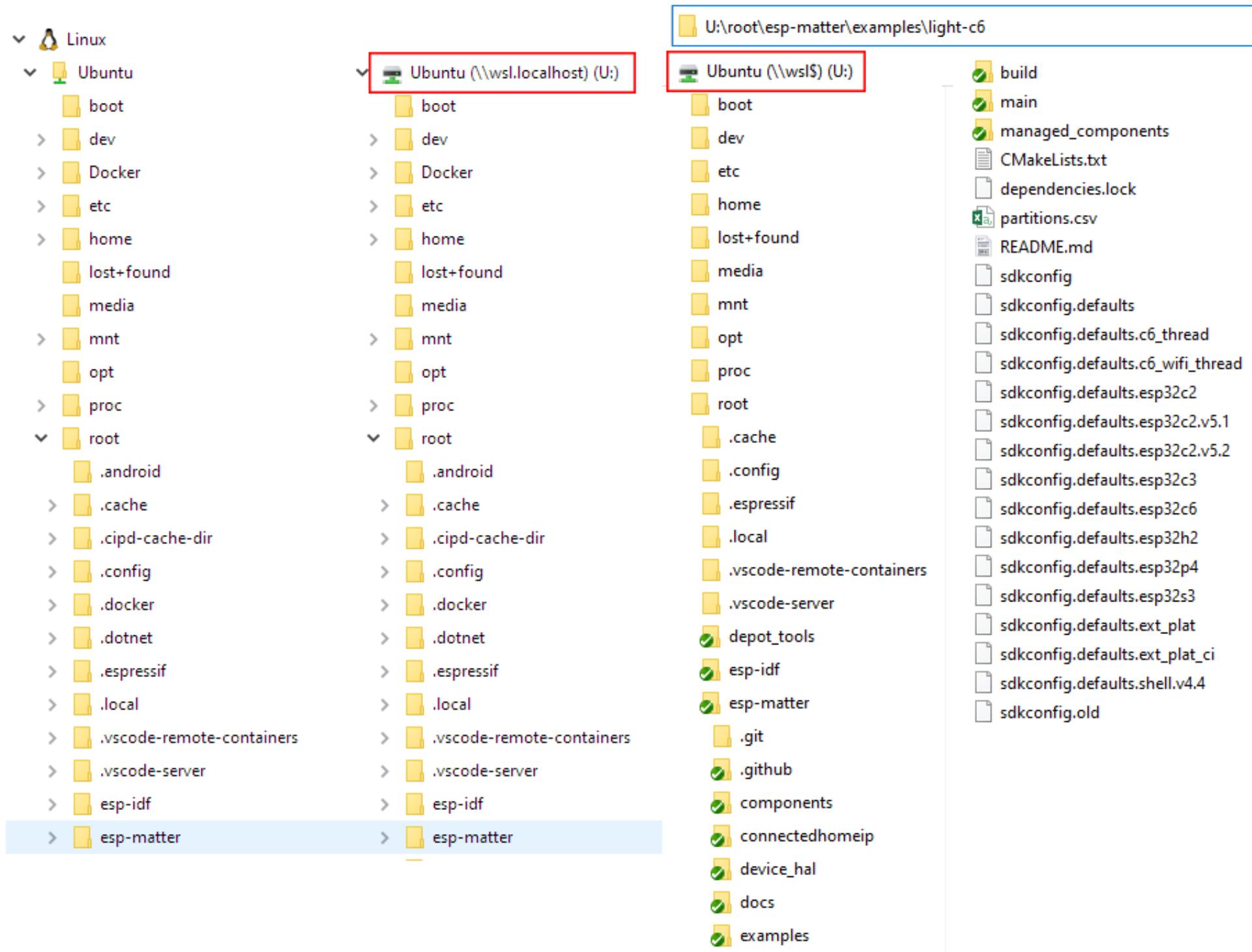
```
ubuntu config --default-user root      #-- default instance  
ubuntu2204 config --default-user root  #-- another instance
```

~~~

2. Ubuntu on Windows App Store

<https://apps.microsoft.com/search?query=ubuntu&hl=en-us&gl=US>

Map network resource `\wsl.localhost\Ubuntu` or `\wsl$\Ubuntu` on disk **U** (for instance). So, the project examples can be found in `U:\root\esp-matter\examples` folder. If we are using another WSL instance, then the path should be something like `\wsl$\Ubuntu-22.04`



3. ESP-IDF Prerequisites (Ubuntu only)

<https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/linux-macos-setup.html>

https://wiki.seeedstudio.com/Xiao_IDF/

~~~

```
sudo apt update
sudo apt upgrade
sudo apt-get install git wget flex bison gperf python3 python3-pip python3-venv cmake ninja-build ccache libffi-dev libssl-dev dfu-util libusb-1.0-0
```

~~~

4. ESP-IDF Setup (Windows and Ubuntu)

<https://docs.espressif.com/projects/esp-matter/en/latest/esp32/developing.html>

Windows Powershell (with Administrator rights):

~~~

```
D:
cd /
mkdir Espressif
cd Espressif
```

~~~

Both, Windows and Ubuntu (esp-idf v5.2.4 is recommended):

~~~

```
git clone --recursive https://github.com/espressif/esp-idf.git
cd esp-idf
git checkout v5.2.4
git submodule update --init --recursive
./install.sh
source ./export.sh
#./install.bat      --- Windows instance
#./export.bat       --- Windows instance
```

~~~

To /root/.bashrc add (Ubuntu only):

~~~

```
--- Alias for setting up the ESP-IDF environment
alias get_idf=' . ~/esp-idf/export.sh'
```

~~~

Than run:

~~~

```
source ~/.bashrc
```

~~~

Now it is possible to run `get_idf` to set up or refresh the esp-idf environment in any terminal session.

5. Matter Prerequisites (Ubuntu only)

<https://github.com/project-chip/connectedhomeip/blob/master/docs/guides/BUILDING.md#prerequisites>

https://wiki.seeedstudio.com/xiao_esp32_matter_env/

~~~  
sudo apt-get install git gcc g++ 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 libreadline-dev default-jre  
~~~

6. ESP Matter Setup (Ubuntu only)

<https://docs.espressif.com/projects/esp-matter/en/latest/esp32/developing.html#esp-matter-setup>

https://wiki.seeedstudio.com/xiao_esp32_matter_env/

~~~

```
cd esp-idf  
source ./export.sh  
cd ..
```

```
git clone --depth 1 https://github.com/espressif/esp-matter.git  
cd esp-matter  
git submodule update --init --depth 1  
cd ./connectedhomeip/connectedhomeip  
../scripts/checkout_submodules.py --platform esp32 linux --shallow  
cd ../../  
../install.sh  
cd ..
```

```
cd esp-idf; source ./export.sh; cd ..  
cd esp-matter; source ./export.sh; cd ..
```

```
export IDF_CCACHE_ENABLE=1
```

~~~

To /root/.bashrc add:

~~~

```
##-- Alias for setting up the ESP-Matter environment  
alias get_matter='~/esp-matter/export.sh'
```

```
##-- Enable ccache to speed up compilation  
alias set_cache='export IDF_CCACHE_ENABLE=1'
```

~~~

Than run:

~~~

```
source ~/.bashrc
```

~~~

Now it is possible to run `get_matter` and `set_cache` to set up or refresh the esp-matter environment in any terminal session.

Note:

"A complete installation of Ubuntu, ESP-IDF and ESP-Matter takes up about 20 GB of disk space on drive C."

7. Install Visual Studio Code

<https://code.visualstudio.com/>

8. Install Remote WSL extension in Visual Studio Code

<https://docs.espressif.com/projects/vscode-esp-idf-extension/en/latest/additionalfeatures/wsl.html#install-remote-wsl-extension-in-visual-studio-code>

9. Install usbipd-win

<https://github.com/dorsel/usbipd-win/releases>

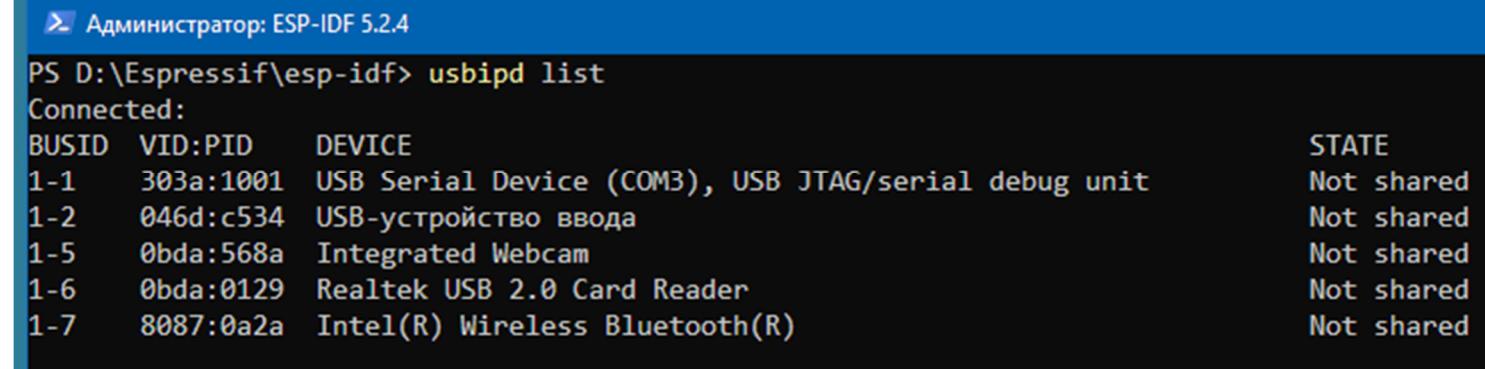
<https://docs.espressif.com/projects/vscode-esp-idf-extension/en/latest/additionalfeatures/wsl.html#usbipd-win-in-wsl>

Windows Powershell:

~~~

```
usbipd list  
usbipd list -u  
usbipd bind --busid 1-3  
usbipd attach --wsl --busid 1-3
```

~~~



BUSID	VID:PID	DEVICE	STATE
1-1	303a:1001	USB Serial Device (COM3), USB JTAG/serial debug unit	Not shared
1-2	046d:c534	USB-устройство ввода	Not shared
1-5	0bda:568a	Integrated Webcam	Not shared
1-6	0bda:0129	Realtek USB 2.0 Card Reader	Not shared
1-7	8087:0a2a	Intel(R) Wireless Bluetooth(R)	Not shared

usbipd_01.png

Administrator: ESP-IDF 5.2.4

```
PS D:\Espressif\esp-idf> usbidp list -u
```

Connected:

BUSID	VID:PID	DEVICE	STATE
1-1	303a:1001	Unknown device	Not shared
1-2	046d:c534	Logitech, Inc., Nano Receiver	Not shared
1-5	0bda:568a	Realtek Semiconductor Corp., Unknown device	Not shared
1-6	0bda:0129	Realtek Semiconductor Corp., RTS5129 Card Reader Controller	Not shared
1-7	8087:0a2a	Intel Corp., Bluetooth wireless interface	Not shared

usbidp_02.png

Administrator: ESP-IDF 5.2.4

```
PS D:\Espressif\esp-idf> usbidp bind --busid 1-1
```

```
PS D:\Espressif\esp-idf> usbidp list
```

Connected:

BUSID	VID:PID	DEVICE	STATE
1-1	303a:1001	USB Serial Device (COM3), USB JTAG/serial debug unit	Shared
1-2	046d:c534	USB-устройство ввода	Not shared
1-5	0bda:568a	Integrated Webcam	Not shared
1-6	0bda:0129	Realtek USB 2.0 Card Reader	Not shared
1-7	8087:0a2a	Intel(R) Wireless Bluetooth(R)	Not shared

usbidp_03.png

Administrator: ESP-IDF 5.2.4

```
PS D:\Espressif\esp-idf> usbidp attach --wsl --busid 1-1
```

```
usbidp: info: Using WSL distribution 'Ubuntu' to attach; the device will be available in all WSL 2 distributions.
```

```
usbidp: info: Detected networking mode 'nat'.
```

```
usbidp: info: Using IP address 172.18.160.1 to reach the host.
```

```
PS D:\Espressif\esp-idf> usbidp list
```

Connected:

BUSID	VID:PID	DEVICE	STATE
1-1	303a:1001	USB Serial Device (COM3), USB JTAG/serial debug unit	Attached
1-2	046d:c534	USB-устройство ввода	Not shared
1-5	0bda:568a	Integrated Webcam	Not shared
1-6	0bda:0129	Realtek USB 2.0 Card Reader	Not shared
1-7	8087:0a2a	Intel(R) Wireless Bluetooth(R)	Not shared

usbipd_04.png

Ubuntu:

~~~

```
dmesg | tail  
lsusb
```

~~~

root@Mike: ~

```
root@Mike:~# dmesg | tail  
[ 4407.060350] vhci_hcd: vhci_device speed not set  
[ 4407.130371] usb 1-1: new full-speed USB device number 2 using vhci_hcd  
[ 4407.210415] vhci_hcd: vhci_device speed not set  
[ 4407.280358] usb 1-1: SetAddress Request (2) to port 0  
[ 4407.323175] usb 1-1: New USB device found, idVendor=303a, idProduct=1001, bcdDevice= 1.02  
[ 4407.323183] usb 1-1: New USB device strings: Mfr=1, Product=2, SerialNumber=3  
[ 4407.323185] usb 1-1: Product: USB JTAG/serial debug unit  
[ 4407.323186] usb 1-1: Manufacturer: Espressif  
[ 4407.323188] usb 1-1: SerialNumber: 40:4C:CA:5E:17:E0  
[ 4407.326209] cdc_acm 1-1:1.0: ttyACM0: USB ACM device
```

usbipd_05.png

root@Mike: ~

```
root@Mike:~# lsusb  
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub  
Bus 001 Device 002: ID 303a:1001 Espressif USB JTAG/serial debug unit  
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

usbipd_06.png

```
root@Mike: ~
```

```
root@Mike:~# esptool.py -p /dev/ttyACM0 flash_id
esptool.py v4.8.1
Serial port /dev/ttyACM0
Connecting...
Detecting chip type... ESP32-C6
Chip is ESP32-C6 (QFN40) (revision v0.1)
Features: WiFi 6, BT 5, IEEE802.15.4
Crystal is 40MHz
MAC: 40:4c:ca:ff:fe:5e:17:e0
BASE MAC: 40:4c:ca:5e:17:e0
MAC_EXT: ff:fe
Uploading stub...
Running stub...
Stub running...
Manufacturer: 85
Device: 2017
Detected flash size: 8MB
Hard resetting via RTS pin...
```

usbipd_07.png

There may be issues with shared access to COM ports in Windows 10 Enterprise. Therefore, it will not be possible to use Ubuntu ports for flashing and monitoring the firmware. In this case, we should make a build in VSCode using a remote WSL or in Ubuntu, and then use a shared drive (U). Here are a couple of examples of BAT files for flashing and monitoring: [/D/Espressif/idf_build_c6.cmd](#) and [/D/Espressif/idf_build_h2.cmd](#)

Administrator: PowerShell Ubuntu

PS C:\Users\mozolin_ml> usbipd list

Connected:

BUSID	VID:PID	DEVICE	STATE
4-1	303a:1001	Устройство с последовательным интерфейсом USB (COM4), USB...	Not shared
4-2	303a:1001	Устройство с последовательным интерфейсом USB (COM5), USB...	Not shared
4-6	03f0:354a	USB-устройство ввода	Not shared
5-3	046d:c05b	USB-устройство ввода	Not shared

Persisted:

GUID	DEVICE
a5e34ae4-9416-4c23-963a-0b0c53b911b0	Устройство с последовательным интерфейсом USB (COM6), USB...A

usbipd: warning: Unknown USB filter 'klfltd.KES-21-17' may be incompatible with this software; 'bind --force' may be required.

usbipd: warning: USB filter 'USBPcap' is known to be incompatible with this software; 'bind --force' will be required.

usdipd_error_01.png

Administrator: PowerShell Ubuntu

PS C:\Users\mozolin_ml> usbipd bind --busid 4-2

usbipd: info: Device with busid '4-2' was already shared.

usbipd: warning: Unknown USB filter 'klfltd.KES-21-17' may be incompatible with this software; 'bind --force' may be required.

usbipd: warning: USB filter 'USBPcap' is known to be incompatible with this software; 'bind --force' will be required.

usdipd_error_02.png

Administrator: PowerShell Ubuntu

PS C:\Users\mozolin_ml> usbipd list

Connected:

BUSID	VID:PID	DEVICE	STATE
4-1	303a:1001	Устройство с последовательным интерфейсом USB (COM4), USB...	Not shared
4-2	303a:1001	Устройство с последовательным интерфейсом USB (COM5), USB...	Shared (forced)
4-6	03f0:354a	USB-устройство ввода	Not shared
5-3	046d:c05b	USB-устройство ввода	Not shared

usdipd_error_03.png

Administrator: PowerShell Ubuntu

```
PS C:\Users\mozolin_ml> usbipd attach --wsl --busid 4-2
usbipd: info: Using WSL distribution 'Ubuntu' to attach; the device will be available in all WSL 2 distributions.
usbipd: error: Unable to run 'usbip' client tool. Please report this at https://github.com/dorssel/usbipd-win/issues.
usbipd_error_04.png
```

10. Make an example project (all the settings are made for ESP32-C6 development board, Ubuntu only)

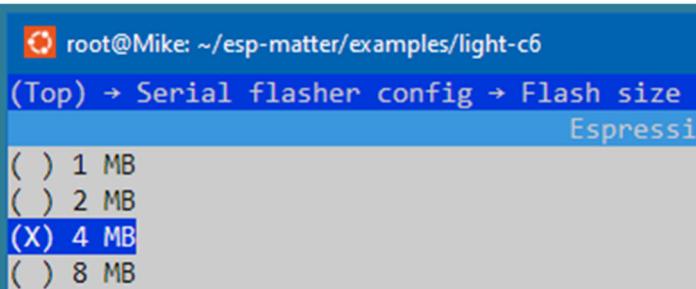
- Make a copy of /root/esp-matter/examples/light folder to .../light-c6
- Open this folder in VSCode using a remote WSL or in Ubuntu

~~~

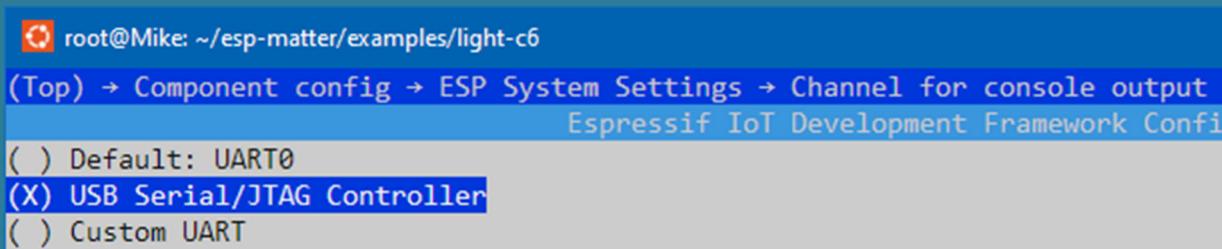
```
cd /root/esp-matter/examples/light-c6      #-- Navigate to the light example directory
rm -rf build/                            #-- Clean previous build files
idf.py set-target esp32c6                 #-- Set the build target to ESP32-C6
idf.py menuconfig                         #-- Enter the configuration menu
```

~~~

```
CONFIG_OPENTHREAD_ENABLED=y
CONFIG_ENABLE_WIFI_STATION=n
CONFIG_USE_MINIMAL_MDNS=n
```



esp_menuconfig_01.png



esp_menuconfig_02.png

```
root@Mike: ~/esp-matter/examples/light-c6
(Top) → Component config → OpenThread
Espressif IoT Development Framework Configuration
[*] OpenThread
[*]   Enable dynamic log level control
[*]   OpenThread console type (OpenThread)
[*]   Thread Operational Dataset --->
Thread Operational Dataset --->
```

esp_menuconfig_03.png

```
root@Mike: ~/esp-matter/examples/light-c6
(Top) → Component config → CHIP Device Layer → Device Identification Options
Espressif IoT Development Framework Configuration
(0xFFFF1) Device Vendor Id
(0x8000) Device Product Id
(0) Default Device Hardware Version
(0) Device Software Version Number
(0) Default Device type
```

esp_menuconfig_04.png

```
root@Mike: ~/esp-matter/examples/light-c6
(Top) → Component config → CHIP Core → General Options
Espressif IoT Development Framework Configuration
(8) Max CHIP Exchange Contexts
(5) Max Fabrics
(8) Max Unsolicited Message Handlers
[ ] Enable Pigweed RPC library
[ ] Use the minimal mDNS implementation shipped in the CHIP library
[*] Use the CHIP shell library
(256) Maximum command line buffer length of the chip shell
```

esp_menuconfig_05.png

```
root@Mike: ~/esp-matter/examples/light-c6
(Top) → Component config → CHIP Device Layer → WiFi Station Options
Espressif IoT Development Framework Configuration
[ ] Enable CHIP WIFI STATION
```

esp_menuconfig_06.png

```
~~~  
idf.py -p /tty/ACM0 build flash monitor  --- Building, flashing and monitoring  
~~~
```

JOIN THE THREAD NETWORK VIA NETWORKKEY

OpenThread Border Router (see: "How_to_setup_and_work_with_OpenThread_Border_Router.docx"):

```
~~~  
dataset active -x  
~~~
```

```
0e080000000000001000000030000154a0300001735060004001ffffe00208def5e21b6165cc560708fde61aeab4004131051000112233445566778899aabbccddeeff030f4  
f70656e5468726561642d32326339010222c90410a5e0c5822c1e723956af6b1ee43f084e0c0402a0f7f8
```

Done

```
~~~  
networkkey  
~~~
```

```
00112233445566778899aabbccddeeff
```

Done

Thread End Device:

```
~~~  
matter esp ot_cli dataset set active  
0e080000000000001000000030000154a0300001735060004001ffffe00208def5e21b6165cc560708fde61aeab4004131051000112233445566778899aa  
bbccddeeff030f4f70656e5468726561642d32326339010222c90410a5e0c5822c1e723956af6b1ee43f084e0c0402a0f7f8
```

Error 7: InvalidArgs

Why so? There is the difference between results of this command:

Here:

```
> matter esp ot_cli dataset set active  
0e080000000000001000000030000154a0300001735060004001ffffe00208def5e21b6165cc560708fde61aeab4004131051000112233445566778899aa  
bbccddeeff030f4f70656e5468726561642d32326339010222c90410a5e0c5822c1e723956af6b1ee43f084e0c0402a0f7f8
```

Example "Join the OTBR network" (<https://openthread.io/codelabs/openthread-border-router>):

```
> matter esp ot_cli dataset set active  
0e0800000000000010000000300001235060004001ffffe002083d3818dc1c8db63f0708fda85ce9df1e662005101d81689e4c0a32f3b4aa112994d29692  
030f4f70656e5468726561642d35326532010252e204103f23f6b8875d4b05541eeb4f9718d2f40c0302a0ff
```

The second one is shorter for 12 symbols!

```
~~~  
matter esp ot_cli dataset networkkey 00112233445566778899aabbccddeeff  
matter esp ot_cli dataset commit active  
matter esp ot_cli ifconfig up  
matter esp ot_cli thread start  
matter esp ot_cli state  
~~~
```

```
matter esp ot_cli dataset networkkey 00112233445566778899aabbccddeeff
```

Done

esp_join_openthread_01.png

```
matter esp ot_cli dataset commit active
```

Done

esp_join_openthread_02.png

```
matter esp ot_cli ifconfig up
```

Done

esp_join_openthread_03.png

```
matter esp ot_cli thread start
```

```
I(82652) OPENTHREAD:[N] Mle-----: Role router -> detached  
I(82652) OPENTHREAD:[I] Mle-----: Attempt to become router  
I(82652) OPENTHREAD:[I] Mle-----: Send Link Request (ff02:0:0:0:0:0:0:2)
```

Done

esp_join_openthread_04.png

```
> matter esp ot_cli state
```

router

Done

esp_join_openthread_05.png

```
> matter esp ot_cli dataset networkkey 00112233445566778899aabbccddeeff
Done
Done
> matter esp ot_cli dataset commit active

I(281397) OPENTHREAD:[I] Settings-----: Saved ActiveDataset
I(281397) OPENTHREAD:[I] DatasetManager: Active dataset set
Done
I(281397) OPENTHREAD:[I] Notifier-----: StateChanged (0x10040100) [KeySeqCntr NetworkKey ActDset]
Done
> matter esp ot_cli ifconfig up

Done
I(294547) OPENTHREAD:[I] Notifier-----: StateChanged (0x01001009) [Ip6+ LLAddr Ip6Mult+ NetifState]
I (294547) OT_STATE: netif up
I (294547) chip[DL]: Posting ESPSystemEvent: IP Event with eventId : 3
I (294547) chip[DL]: Posting ESPSystemEvent: IP Event with eventId : 3
Done
> matter esp ot_cli thread start
```

matter_esp_ot_cli.png

11. ??? Test Setup (CHIP Tool)

<https://docs.espressif.com/projects/esp-matter/en/latest/esp32c6/developing.html#test-setup-chip-tool>

12. ??? Working with the CHIP Tool

https://github.com/project-chip/connectedhomeip/blob/master/docs/development_controllers/chip-tool/chip_tool_guide.md

13. Matter Shell Reference

https://project-chip.github.io/connectedhomeip-doc/examples/chef/README_SHELL.html

<https://docs.espressif.com/projects/esp-matter/en/latest/esp32/developing.html#device-console>

```
matter config

VendorId:      65521 (0xFFFF)
ProductId:     32768 (0x8000)
HardwareVersion: 0 (0x0)
PinCode:       20202021
Discriminator: f00
Done
```

esp_matter_01.png

```
matter onboardingcodes none
```

```
QRCode: MT:Y.K90-Q000KA0648G00
```

```
QRCodeUrl: https://project-chip.github.io/connectedhomeip/qrcode.html?data=MT%3AY.K90-Q000KA0648G00
```

```
ManualPairingCode: 34970112332
```

```
Done
```

esp_matter_02.png

CHIP: QR Code

project-chip.github.io/connectedhomeip/qrcode.html?data=MT%3AW0GU2OTB00KA0648G00

Please scan with your CHIPTool app.



Payload: MT:W0GU2OTB00KA0648G00

This QR code is unique for your device. You may print a copy of this for subsequent use.

[Print QR Code](#)

qrcode_01.png

```
matter dataset set active 0e08000000000001000000030000154a0300001735060004001ffffe00208def5e21b6165cc560708fde61aeab4004131  
051000112233445566778899aabbcdddeeff030f4f70656e5468726561642d32326339010222c90410a5e0c5822c1e723956af6b1ee43f084e0c0402a0  
f7f8
```

Error: 47

esp_matter_03.png

14. How to generate Matter Onboarding Codes (QR Code and Manual Pairing Code)

<https://docs.espressif.com/projects/esp-matter/en/latest/esp32/faq.html#a1-9-how-to-generate-matter-onboarding-codes-qr-code-and-manual-pairing-code>

~~~

--- Generate the QR Code

```
chip-tool payload generate-qrcode --discriminator 3131 --setup-pin-code 20201111 --vendor-id 0xFFFF1 --product-id 0x8004 --version 0 --commissioning-mode 0 --rendezvous 2
```

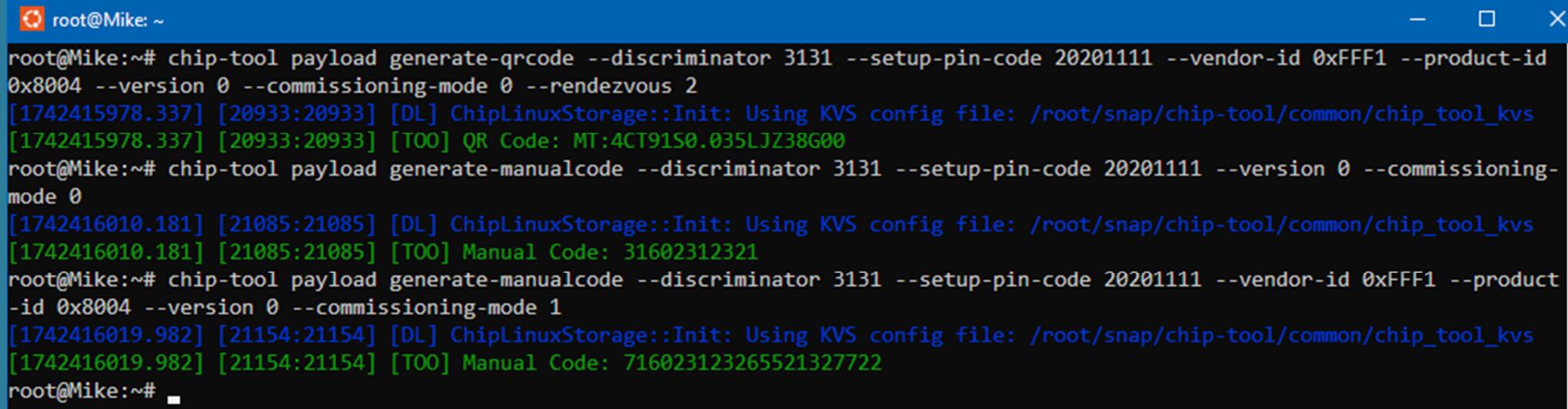
--- Generates the short manual pairing code (11-digit)

```
chip-tool payload generate-manualcode --discriminator 3131 --setup-pin-code 20201111 --version 0 --commissioning-mode 0
```

--- To generate a long manual pairing code (21-digit) that includes both the vendor ID and product ID, --commissioning-mode parameter must be set to either 1 or 2, indicating a non-standard commissioning flow

```
chip-tool payload generate-manualcode --discriminator 3131 --setup-pin-code 20201111 --vendor-id 0xFFFF1 --product-id 0x8004 --version 0 --commissioning-mode 1
```

~~~



The screenshot shows a terminal window with a blue header bar containing the text "root@Mike: ~". The main area of the terminal displays the command-line interface for generating Matter onboarding codes. The user runs three commands:

- The first command generates a QR code: `chip-tool payload generate-qrcode --discriminator 3131 --setup-pin-code 20201111 --vendor-id 0xFFFF1 --product-id 0x8004 --version 0 --commissioning-mode 0 --rendezvous 2`. The output shows log messages: "[1742415978.337] [20933:20933] [DL] ChipLinuxStorage::Init: Using KVS config file: /root/snap/chip-tool/common/chip_tool_kvs" and "[1742415978.337] [20933:20933] [TOO] QR Code: MT:4CT9150.035LJZ38G00".
- The second command generates a short manual pairing code: `chip-tool payload generate-manualcode --discriminator 3131 --setup-pin-code 20201111 --version 0 --commissioning-mode 0`. The output shows log messages: "[1742416010.181] [21085:21085] [DL] ChipLinuxStorage::Init: Using KVS config file: /root/snap/chip-tool/common/chip_tool_kvs" and "[1742416010.181] [21085:21085] [TOO] Manual Code: 31602312321".
- The third command generates a long manual pairing code: `chip-tool payload generate-manualcode --discriminator 3131 --setup-pin-code 20201111 --vendor-id 0xFFFF1 --product-id 0x8004 --version 0 --commissioning-mode 1`. The output shows log messages: "[1742416019.982] [21154:21154] [DL] ChipLinuxStorage::Init: Using KVS config file: /root/snap/chip-tool/common/chip_tool_kvs" and "[1742416019.982] [21154:21154] [TOO] Manual Code: 716023123265521327722".

esp_chiptool_qrcode.png