

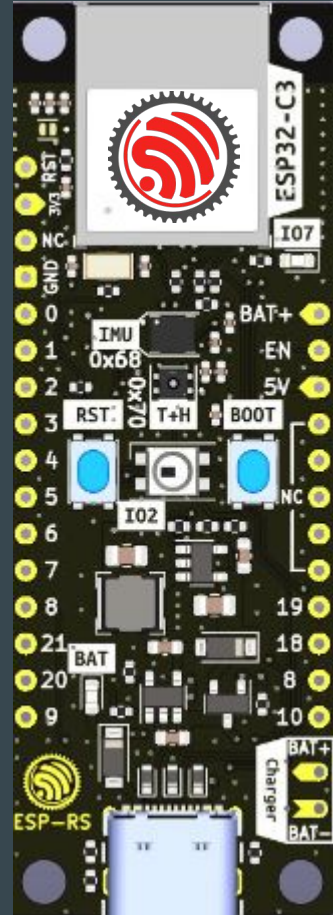
Matter and Home Automation with ESP32



2023-04-20

Digital Futures meetup - Brno - Clubco

Juraj Michálek - Espressif Systems



Espressif System

Espressif Systems (688018.SH) is a public multinational, fabless semiconductor company established in 2008, with offices in China, the Czech Republic, India, Singapore and Brazil.

ESP8266 - WiFi

ESP32 - dual-core

ESP32-S, ESP32-C and ESP32-H series

Open source: <https://github.com/espressif>



Espressif in Brno

Just across the street to Clubco ;-)

Espressif Systems (Czech) s.r.o.


Přízova 3, 602 00 Brno

Czechia, Europe





Many chips, many boards - quick help


<https://products.espressif.com/>



ESP32-S3

 Product Brief

 Docs & Certs

 DevKits


ESP32-S3 is a low-power MCU-based SoC that supports 2.4 GHz Wi-Fi and Bluetooth® Low Energy (Bluetooth LE).

ESP32-S3 has a complete Wi-Fi subsystem and a Bluetooth LE subsystem, State-of-the-art power and RF performance. S3 provides a rich set of peripheral interfaces, and supports ultra-low-power applications. Different security features allow the device to meet stringent security requirements.


Features:


- **Core:** Xtensa® single-dual 32-bit LX7 CPU, frequency up to 240MHz
- **Memories:**


Block Diagram:




List: 203 items

 IC/Module

 Development Board

 Comparison

 Export

<input type="checkbox"/>	Index	Name	MPN	Marketing Status	Type	Wi-Fi
<input type="checkbox"/>	1	ESP32-S3	ESP32-S3	Mass Production	SoC	IEEE 802.11 b/g/n; 2.4 ...
<input type="checkbox"/>	2	ESP32-S3	ESP32-S3R2	Mass Production	SoC	IEEE 802.11 b/g/n; 2.4 ...
<input type="checkbox"/>	3	ESP32-S3	ESP32-S3R8	Mass Production	SoC	IEEE 802.11 b/g/n; 2.4 ...
<input type="checkbox"/>	4	ESP32-S3	ESP32-S3R...	Mass Production	SoC	IEEE 802.11 b/g/n; 2.4 ...
<input type="checkbox"/>	5	ESP32-S3	ESP32-S3F...	Mass Production	SoC	IEEE 802.11 b/g/n; 2.4 ...

OSes and integration



ESP-IDF (OS based on FreeRTOS) - <https://github.com/espressif/esp-idf>

no_std a.k.a. bare metal with Rust - <https://github.com/esp-rs/esp-hal> (minimalistic)



Zephyr - <https://zephyrproject.org/>

- EDC22 Day 1 Talk 10: Applications of Asymmetric Multiprocessing with ESP32 Devices - including Rust on one core - <https://youtu.be/oble9ObAqxM>



NuttX - <https://nuttx.apache.org/> (as app, Linux-like OS)

SVD files: <https://github.com/espressif/svd>

Programming languages

Active support by Espressif teams

- C/C++
 - most common choice - <https://github.com/espressif/esp-idf>
- Rust
 - recommended for new design evaluation - <https://github.com/esp-rs>
 - security and memory guaranties of Rust
 - multi-target Xtensa, RISC-V, plus WASM, desktops or mobile
- Arduino - Maker choice
 - Arduino IDE 2.x
 - note: check the license for production



esp-rs

Libraries, crates and examples for using Rust on Espressif SoC's

Languages and frameworks

New and noteworthy:

- [Ada/Spark](#) - from AdaCore
- [Embedded Wizard](#) - DSL and C
- [Zig](#)

VM based:

- [CircuitPython](#) and [MicroPython](#) - Python-like language
- [Toit](#) - Toit language
- [Nanoframework](#) - C# language
- [Mongoose OS](#)
- downside: bigger VM
- upside: more robust, comes with OTA and monitoring

IDE

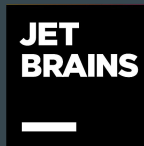
Supported by Espressif:

- VS Code with Espressif Extension - <https://docs.espressif.com/projects/esp-idf/en/stable/esp32/get-started/vscode-setup.html>
- Espressif IDE - <https://dl.espressif.com/dl/esp-idf/>



Supported by JetBrains:

- CLion - <https://www.jetbrains.com/clion/>



Supported by SysProgs

- Visual Studio with VisualGDB - <https://visualgdb.com/>



Supported by TARA Systems

- Embedded Wizard - <https://www.embedded-wizard.de/>



GUI Solutions by TARA Systems

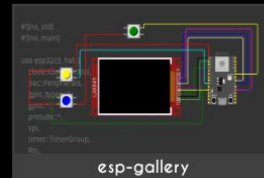
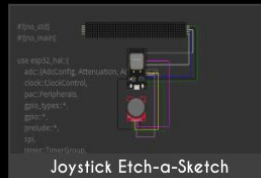
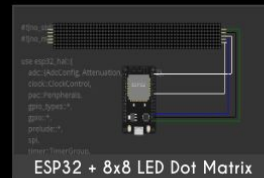
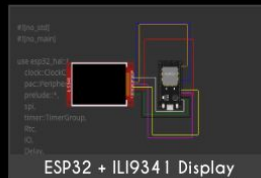
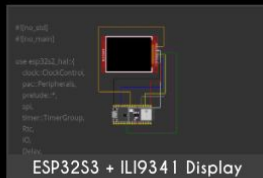
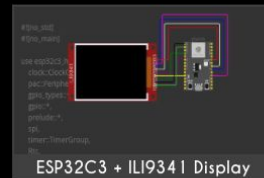
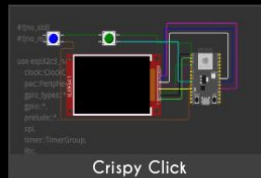
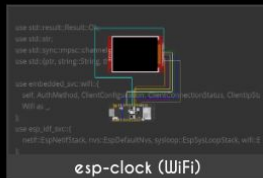
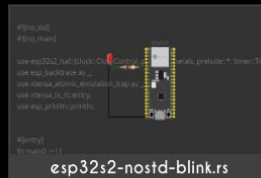
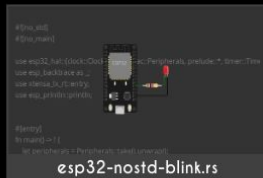
wokwi.com/rust

Contribute: <https://github.com/wokwi>

EDC22 Day 1 Talk 9: Your browser is
ESP32 - Wokwi -
<https://youtu.be/TKe4MgD6O8o>

Rust project examples

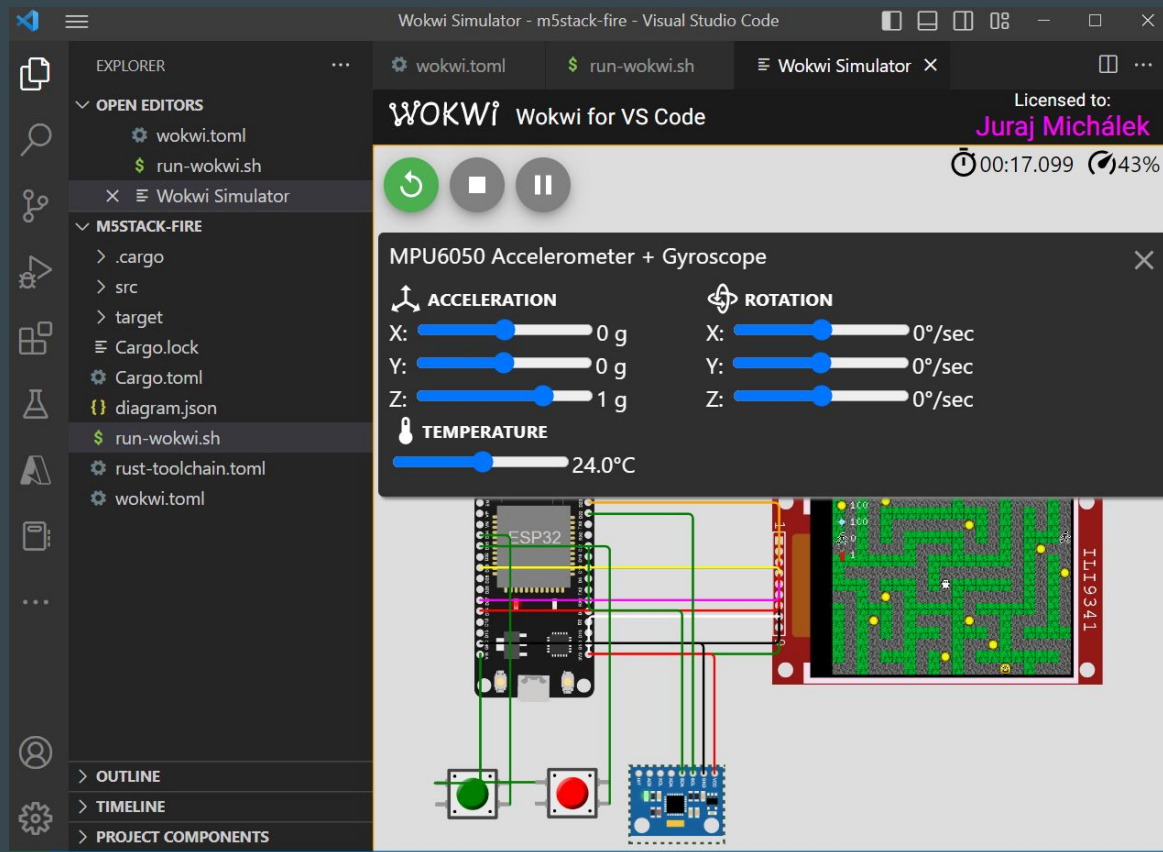
+ NEW PROJECT



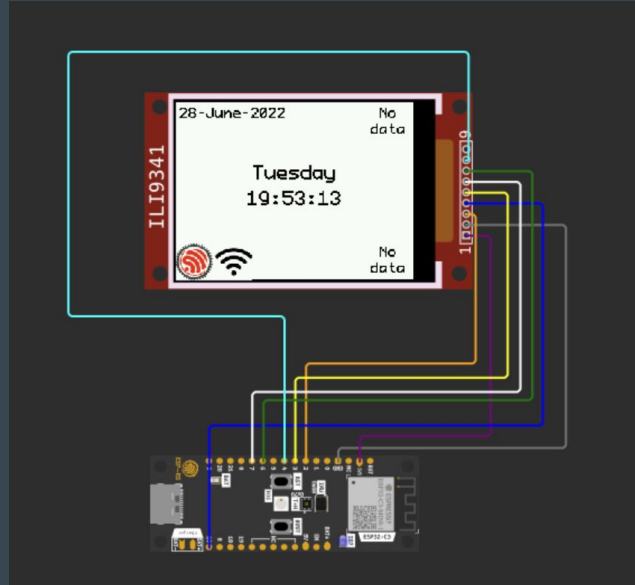
Wokwi - VS Code Plugin

Add wokwi.toml and diagram.json
to your project

CTRL+Shift+P - Wokwi: Start Simulator



Development containers and Wokwi



<https://github.com/playfulFence/esp-clock#dev-containers>

GUI

- LVGL - with SquareLine Studio
- Embedded Wizard
- QT for MCU
- Slint
- Rust bare metal with Embedded Graphics

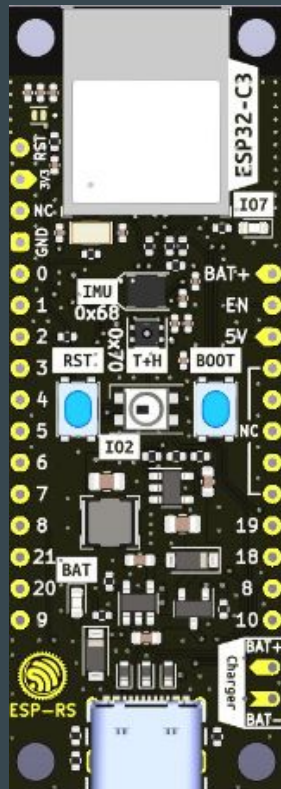
Designing Open Hardware - esp-rust-board

KiCad templates

<https://github.com/esp-rs/esp-rust-board>

ESP32-C3-DevKit-RUST-1 (available at Mouser, AliExpress)

<https://www.espressif.com/en/products/devkits>



Some examples of ESP32 based projects

CTAG-TBD

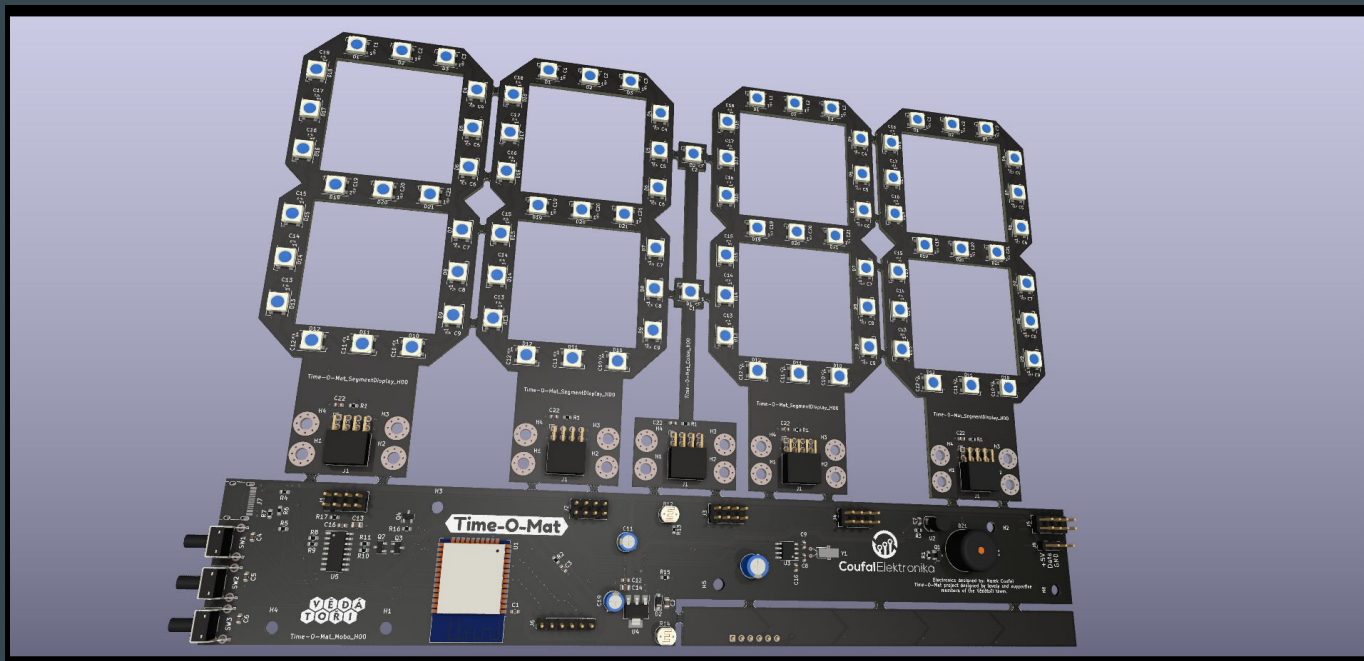
CTAG TBD >>to be determined<< an extendible
open source Eurorack sound module

<https://github.com/ctag-fh-kiel/ctag-tbd>



Time-O-Mat - built at summer camp

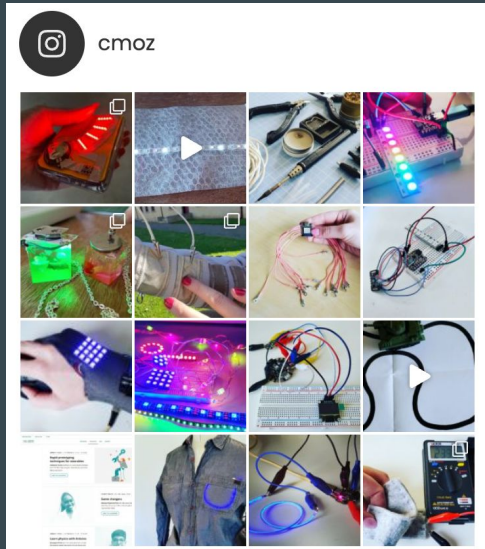
<https://github.com/vedatori/Time-O-Mat>



Wearables

The Ultimate Guide to Informed Wearable Technology

- book: <https://packt.link/01VBv>



Grafana



<https://grafana.com/blog/2020/06/17/how-to-monitor-a-sourdough-starter-with-grafana/>

<https://github.com/grafana/diy-iot> - Arduino now. We're not Rust yet :)



Many Problems of Home Automation and IoT

Silos

Fragmentation

Security

UX

Development

Certification

...

Matter 1.0

Repo: <https://github.com/project-chip/connectedhomeip>

Specs: <https://csa-iot.org/developer-resource/specifications-download-request/>

Matter for end user

No more silos

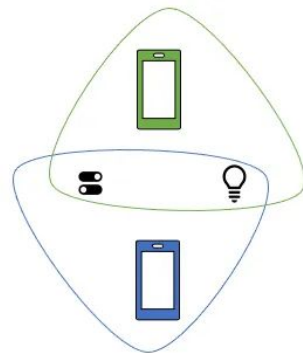
More automation - device-to-device communication

No more transport silos

More ecosystems

Better security

<https://blog.espressif.com/what-does-matter-mean-to-you-fa3bb53a7793>



Matter for device makers

Ease of development

Power of open

Device-to-device automation

Matter ecosystems

Manufacturer-specific innovations

Espressif and Matter

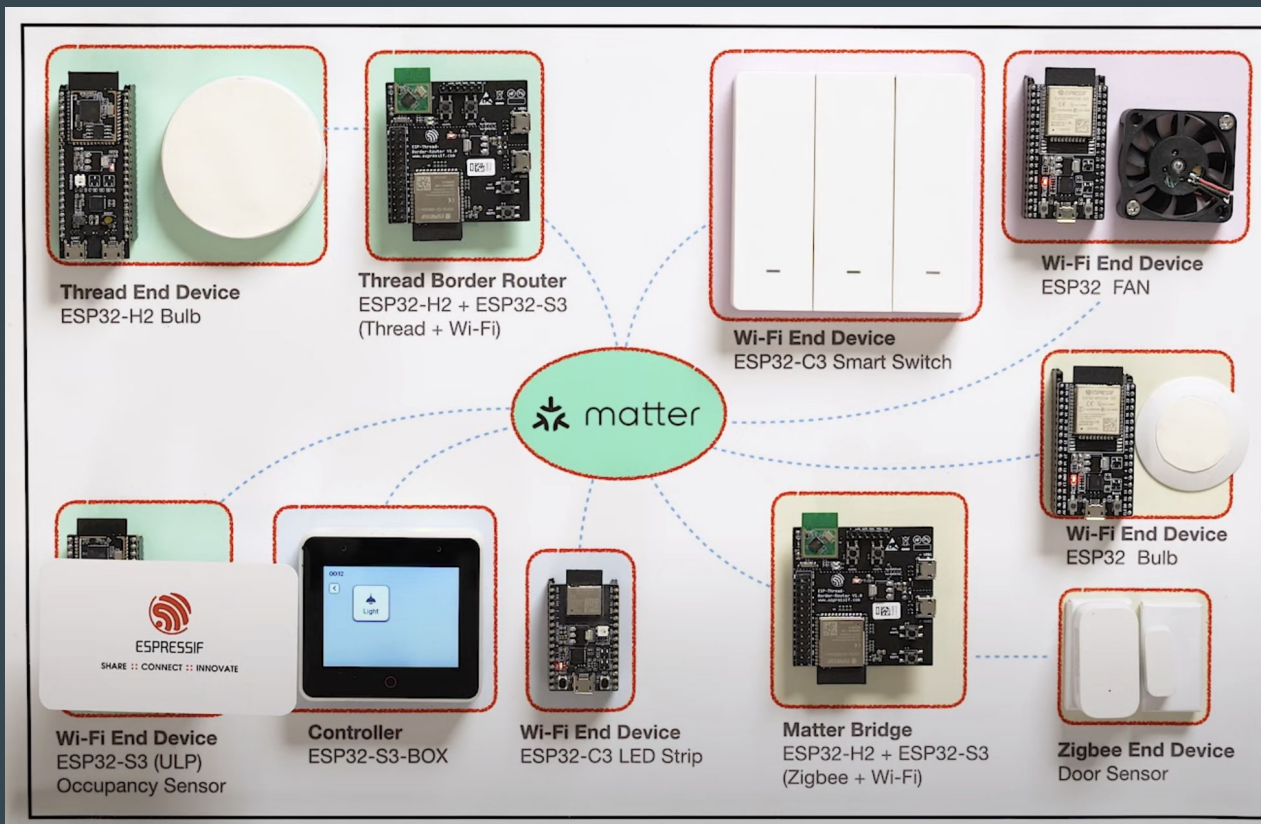


Support for ESP32 is upstream

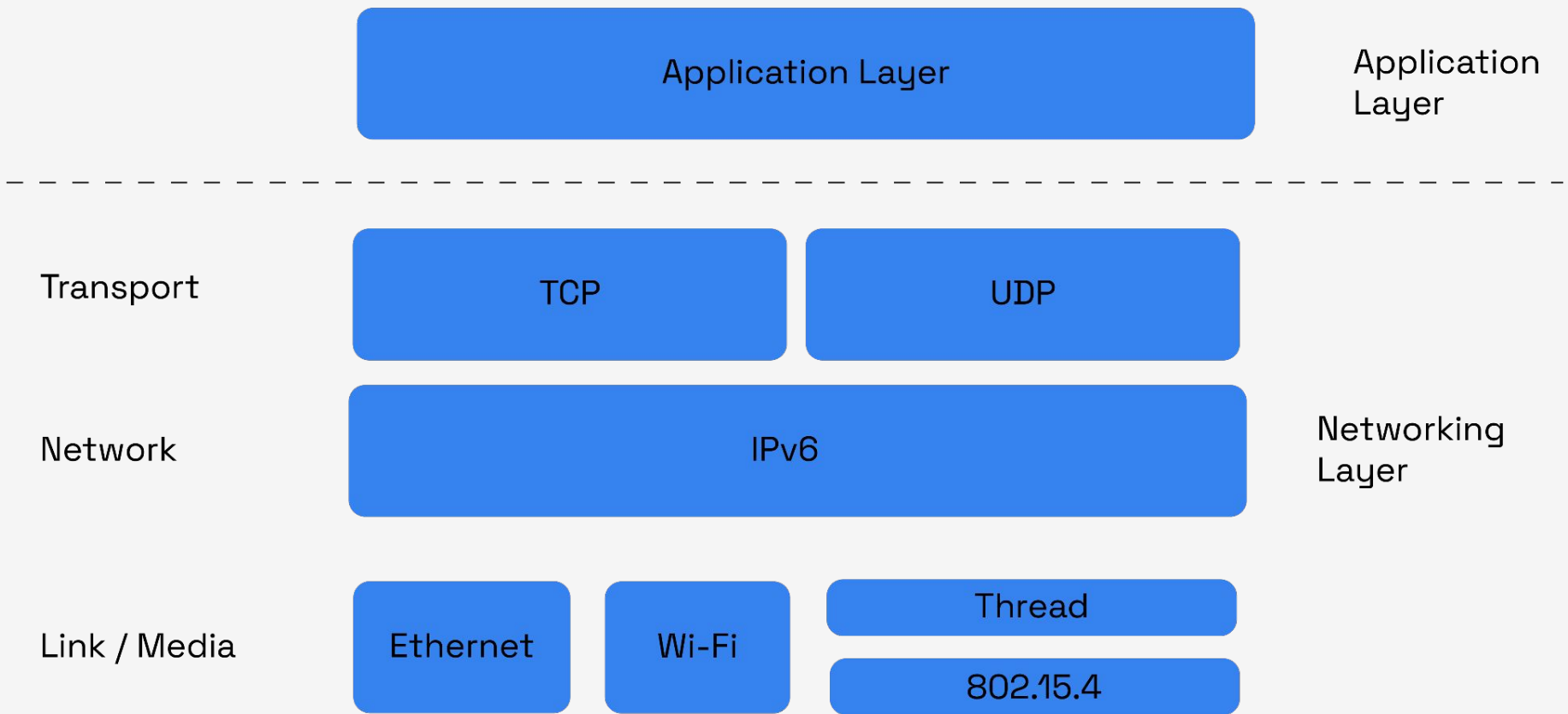
<https://github.com/project-chip/connectedhomeip/tree/master/examples/all-clusters-app/esp32>

esp-matter project - <https://github.com/espressif/esp-matter>

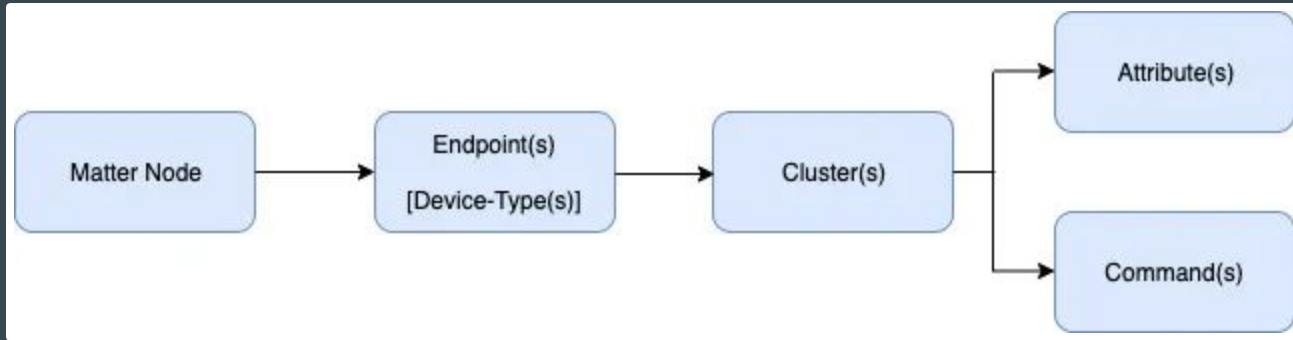
matter-rs - Rust - <https://github.com/project-chip/matter-rs>



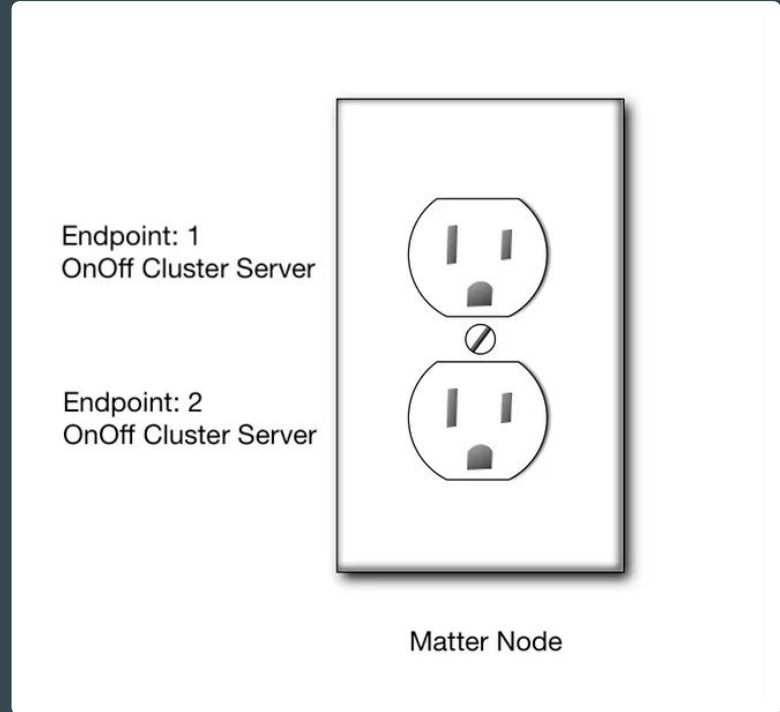
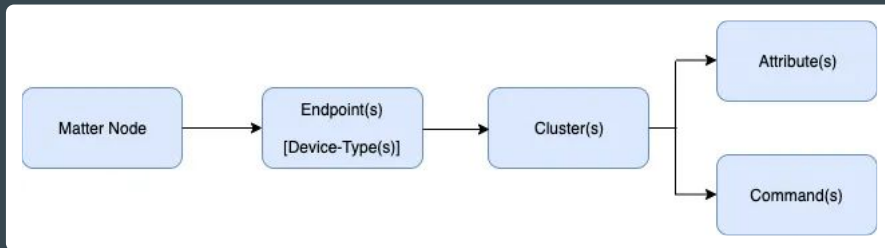
Espressif's Matter Demo - [https://youtu.be/Jr4Lut NgqA](https://youtu.be/Jr4LutNgqA)



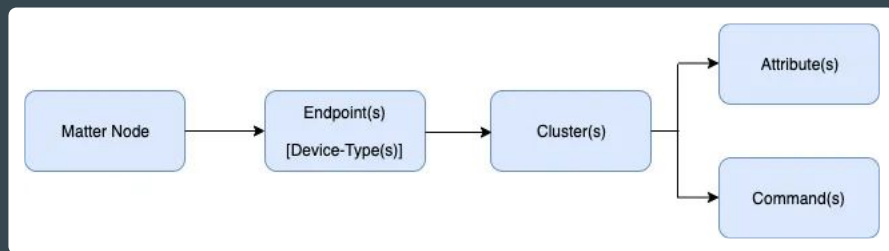
Simplistic Data Model



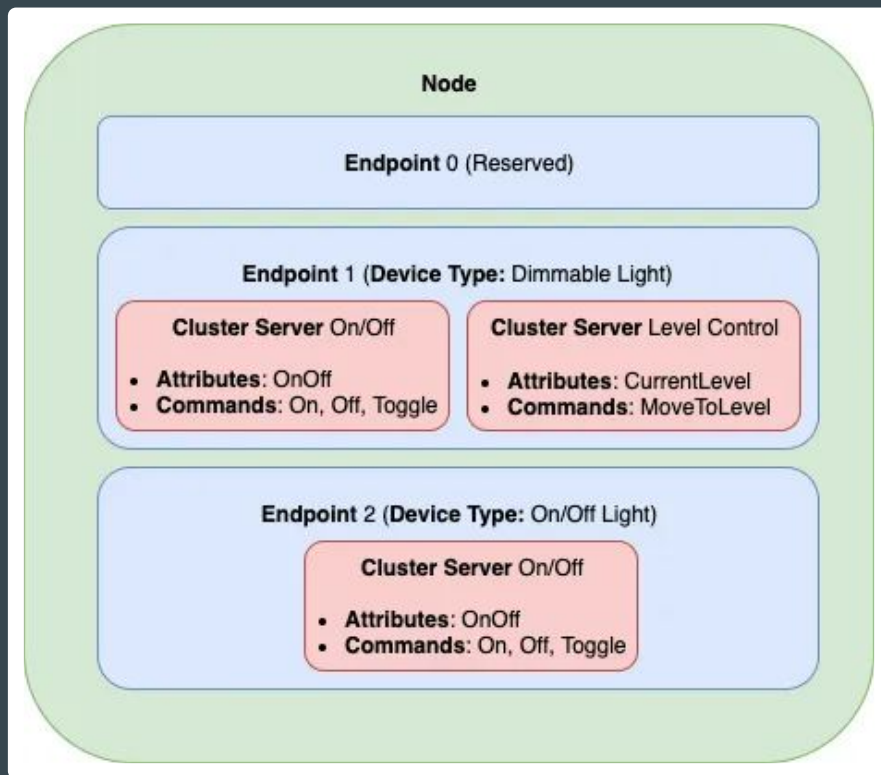
Matter node



Matter Node - OnOff Cluster Client - 4 endpoints



Simplistic Representation of a Matter Data Model



Cluster list

ID; Cluster Name

0x0003 - Identify

0x0004 - Groups

0x0005 - Scenes

0x0006 - On/Off

0x0008 - Level controlling

0x0045 - Boolean state

0x0050 - Mode Select

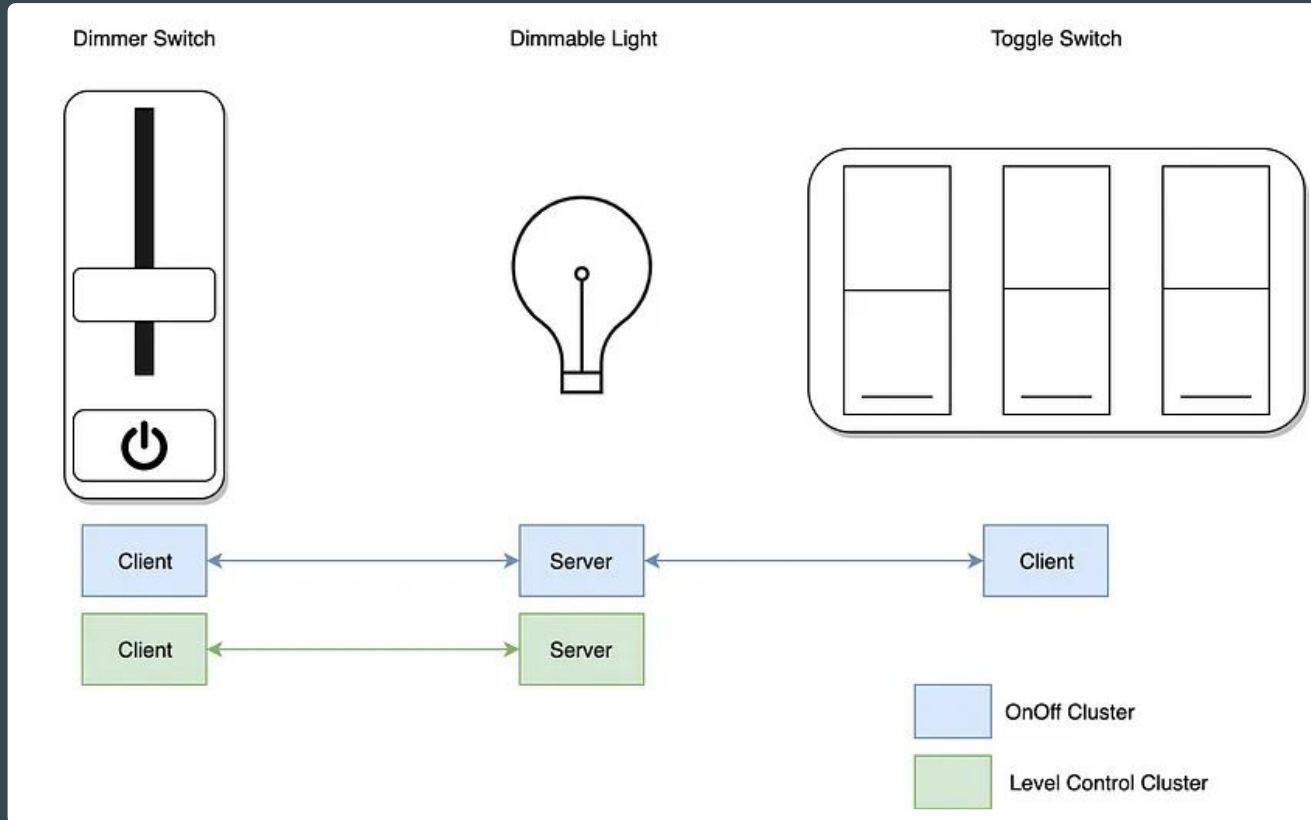
0x0508 - Low Power

0x0503 - Wake On LAN

0x003b - Switch

Specification: [Matter Application Clusters v1.0](#)

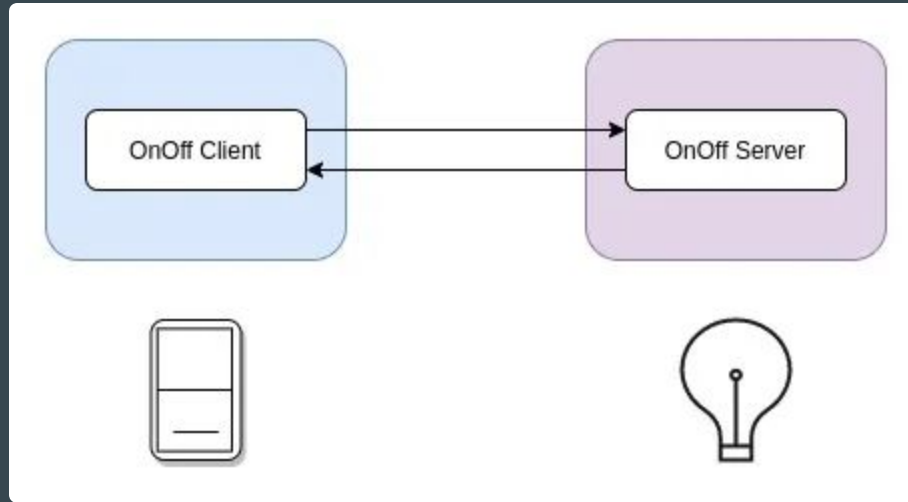
Matter - Client - Server



Endpoint 0 (reserved)

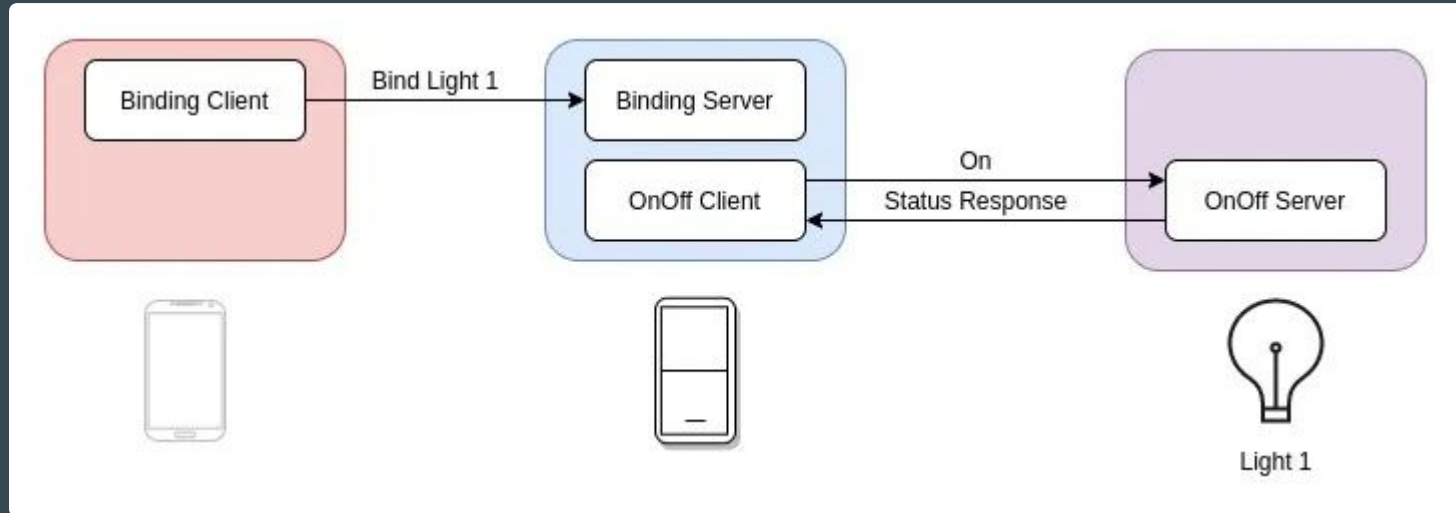
- Basic Information Cluster Server: Provides basic information about the node, like firmware version, manufacturer etc
- ACL Cluster Server: Allows configuration of the Access Control Lists for this node.
- Network Commissioning Cluster Server: Allows configuration of a network (Wi-Fi, Ethernet, Thread) on the node.

Device-to-device automation

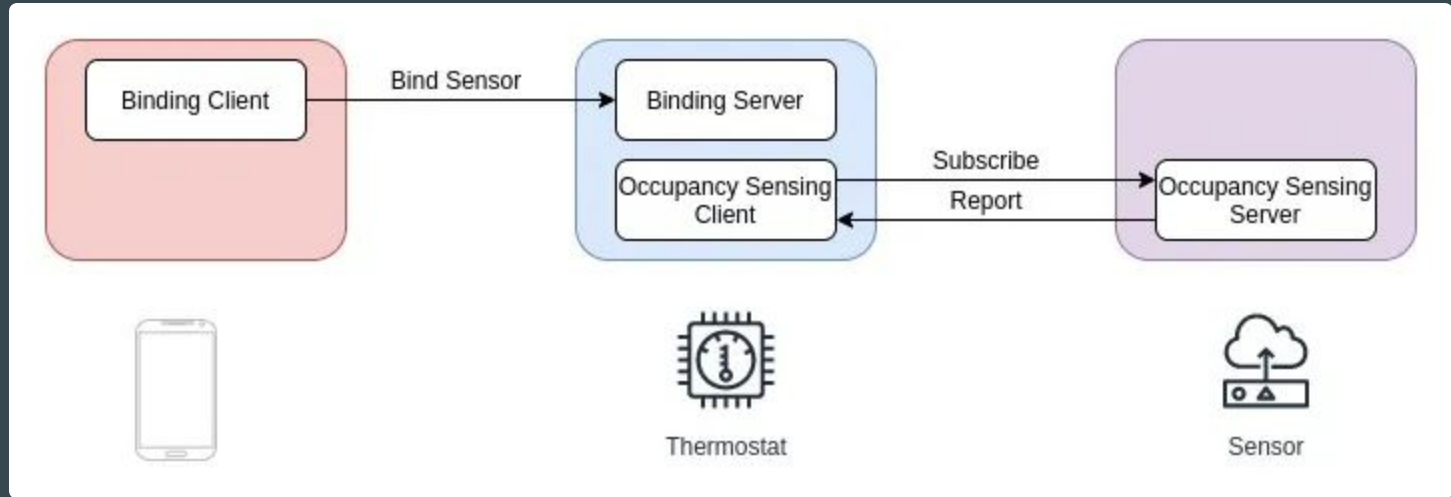


Synchronous communication

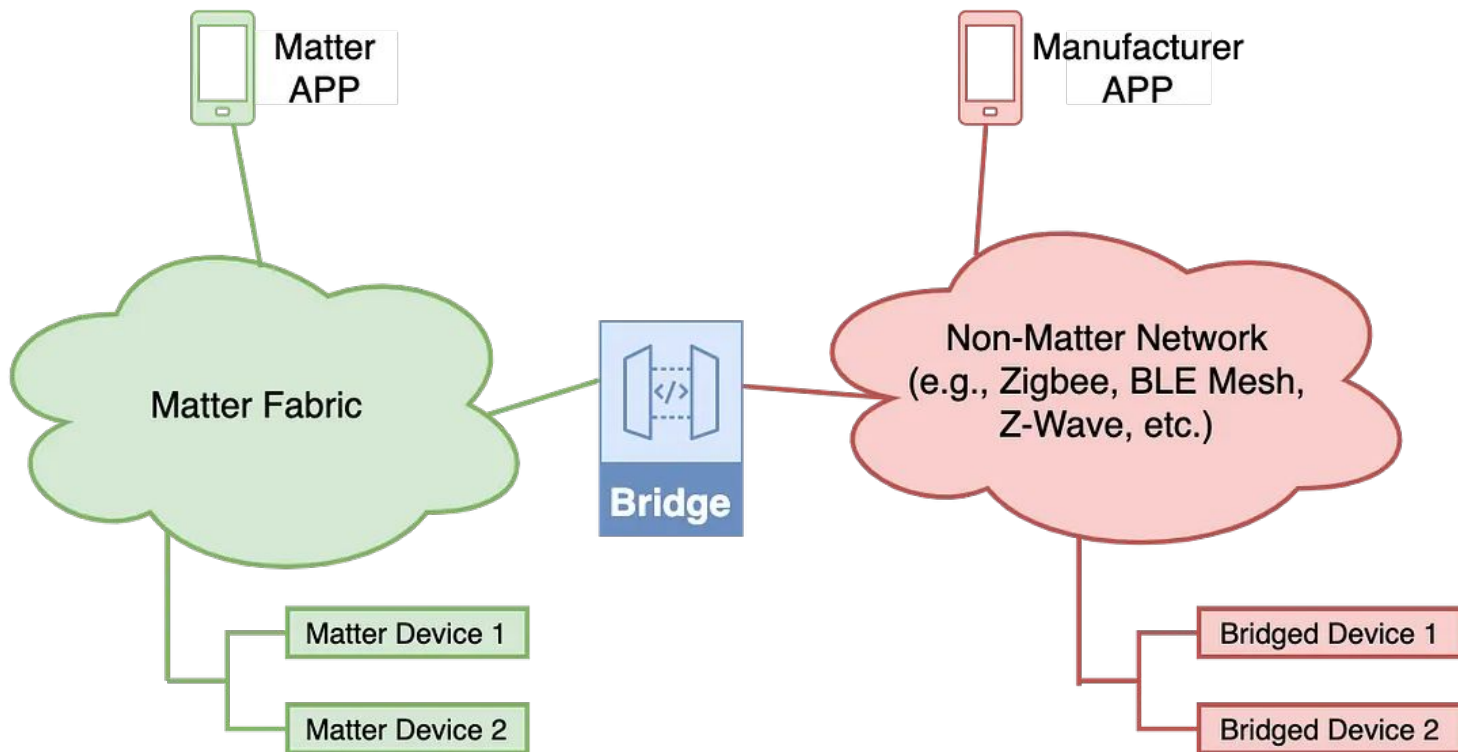
Requires Binding server



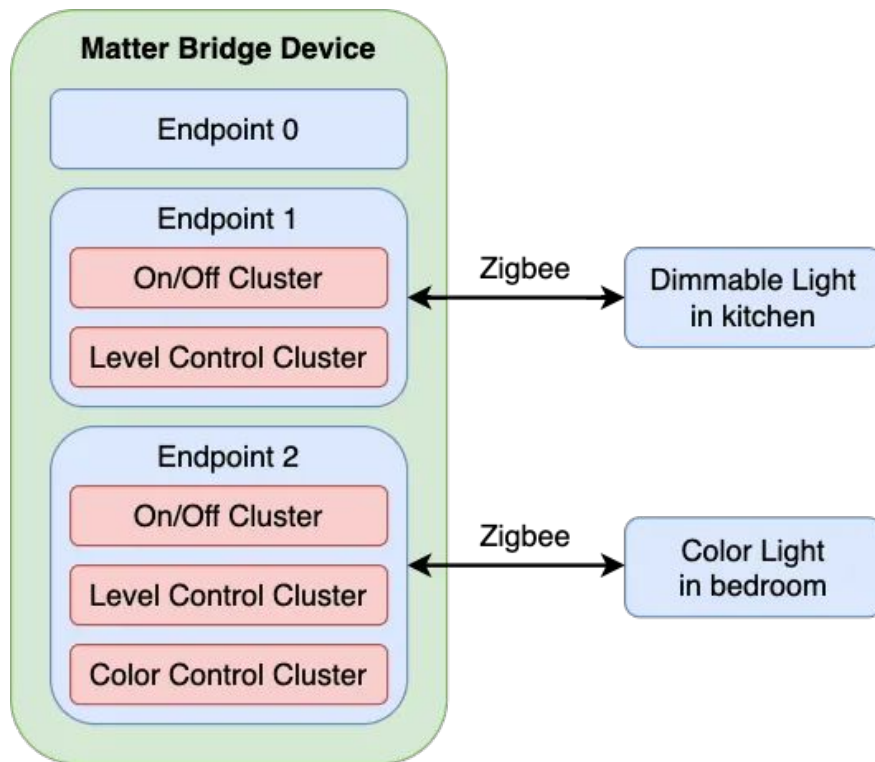
Asynchronous Notification (Subscribe-Report)



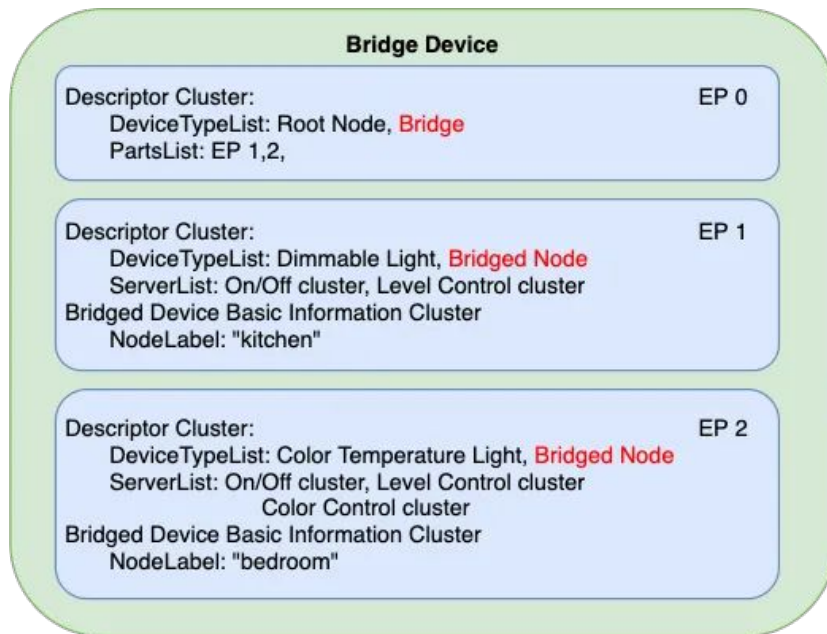
Matter Bridge



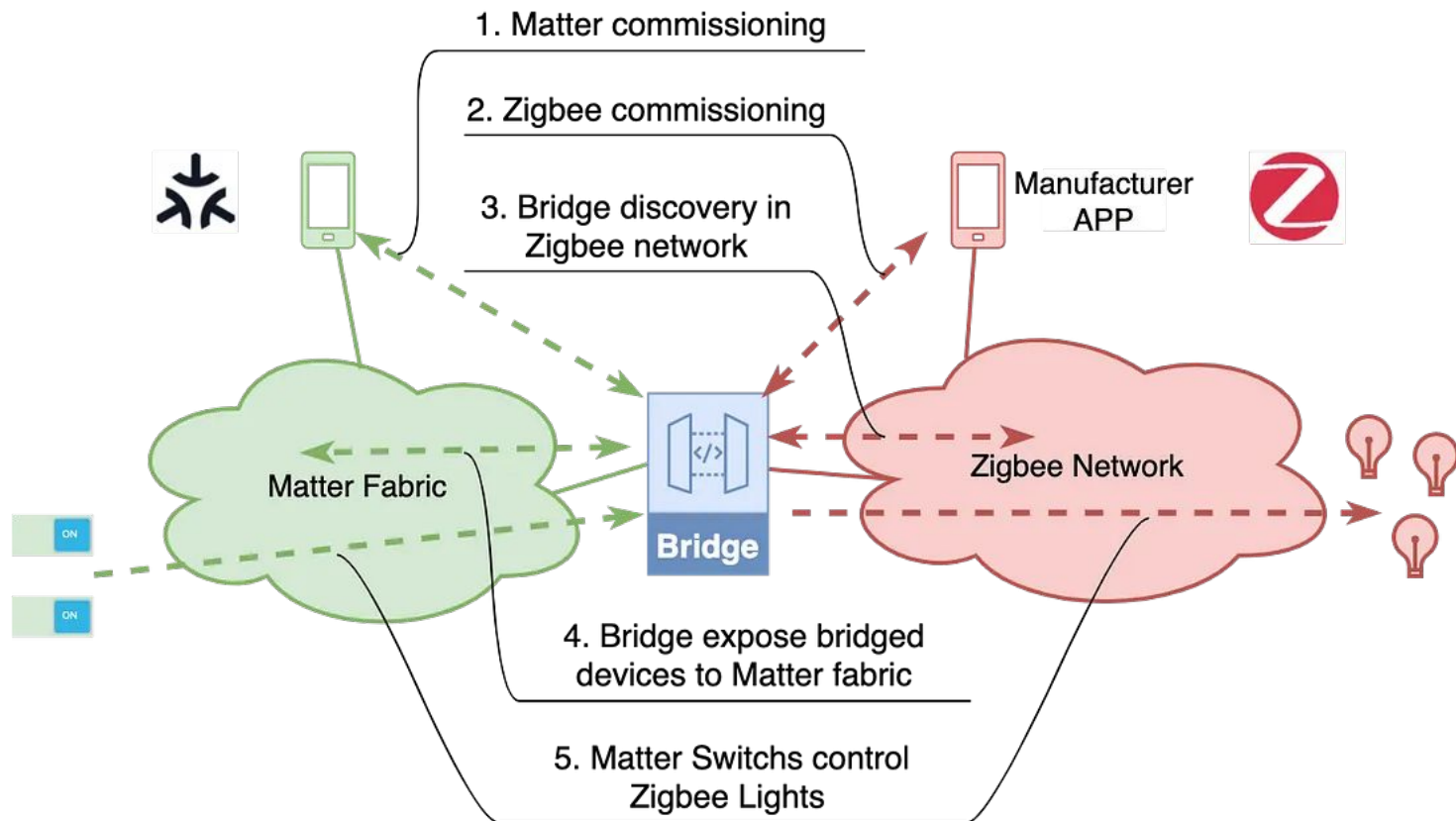
Matter Bridge Device



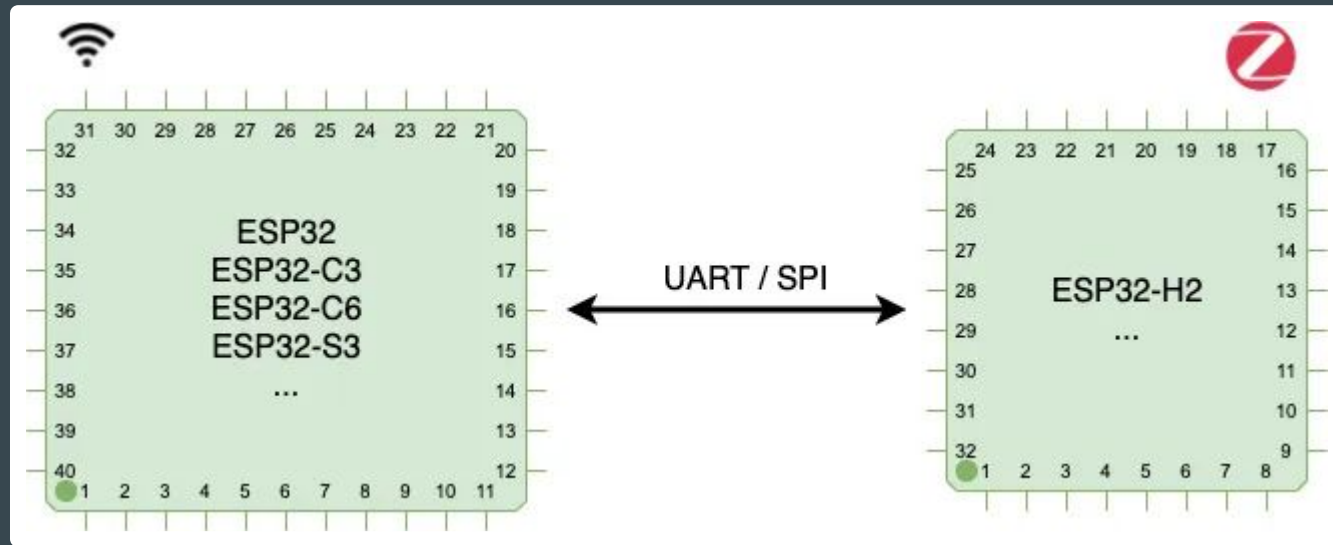
Matter Bridge data Model



Matter-Zigbee Bridge Workflow



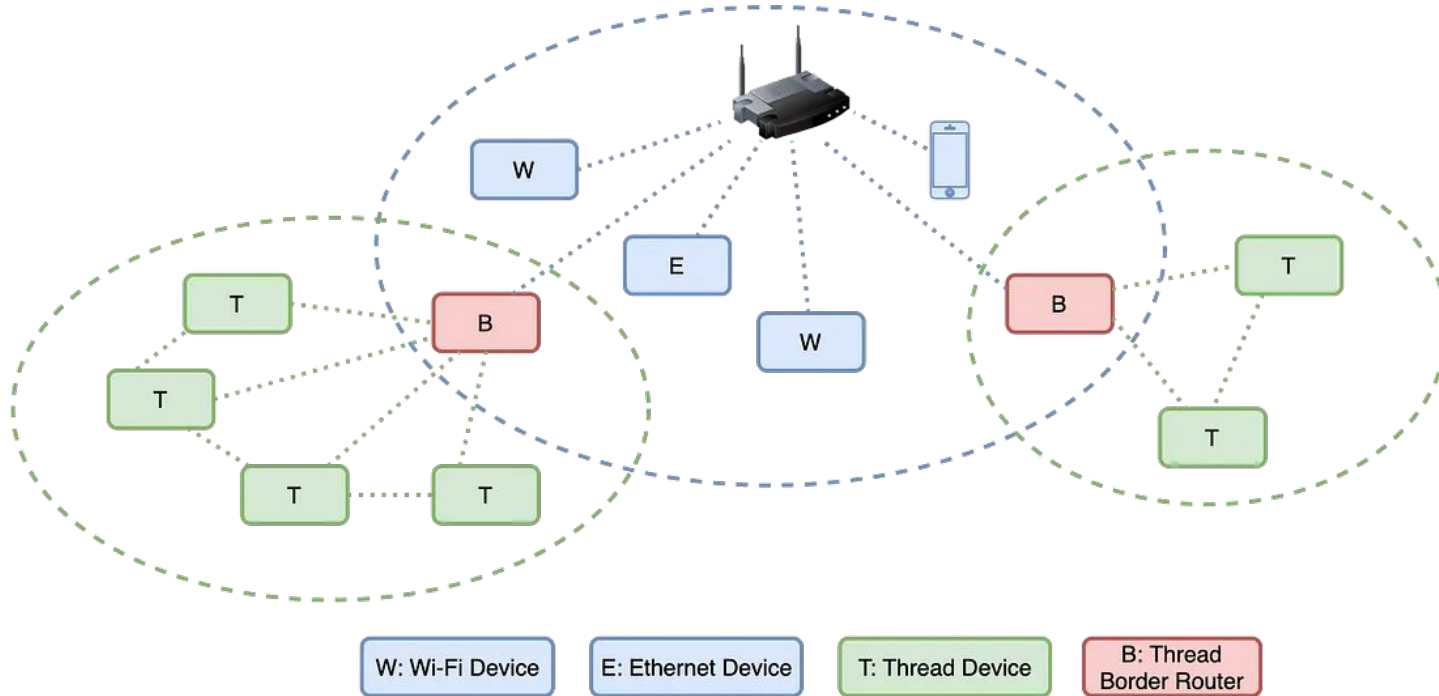
Matter-Zigbee bridge solution



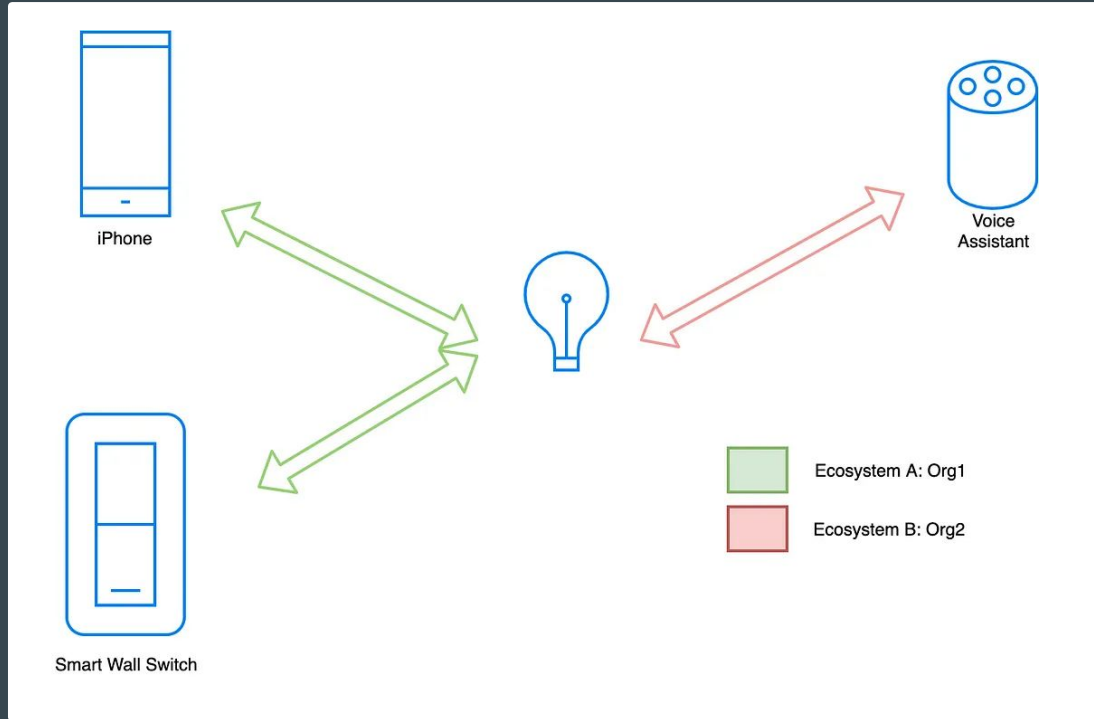
Matter-BLE mesh bridge solution

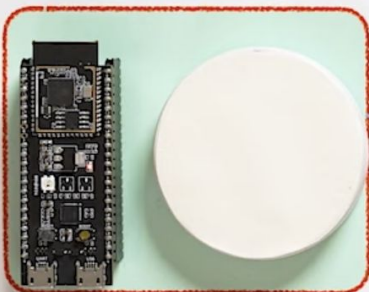


Matter topology



Multi-Admin





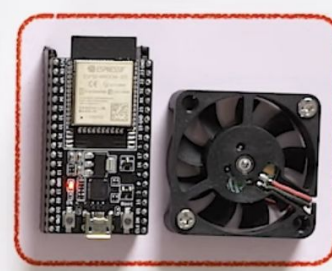
Thread End Device
ESP32-H2 Bulb



Thread Border Router
ESP32-H2 + ESP32-S3
(Thread + Wi-Fi)



Wi-Fi End Device
ESP32-C3 Smart Switch



Wi-Fi End Device
ESP32 FAN



Wi-Fi End Device
ESP32-S3 (ULP)
Occupancy Sensor



Controller
ESP32-S3-BOX



Wi-Fi End Device
ESP32-C3 LED Strip



Matter Bridge
ESP32-H2 + ESP32-S3
(Zigbee + Wi-Fi)

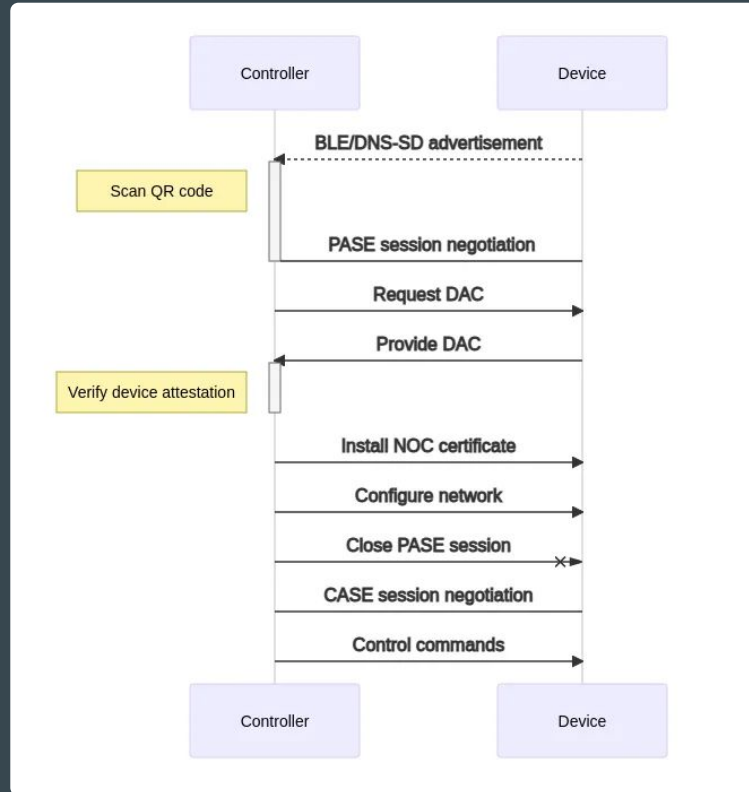


Wi-Fi End Device
ESP32 Bulb



Zigbee End Device
Door Sensor

Pairing Matter device



ESP Launchpad - <https://espressif.github.io/esp-launchpad/>



 Quick Start

 DIY

 Connect

 Console

 Settings

 About

ESP Launchpad helps you to flash the selected firmware image onto your device.

Ensure you have connected your device to the serial USB port. Click on the 'Connect' button in the top menu option, to connect to your attached device.

Note : Once you flash your device, any earlier firmware would be overwritten.

Choose from some of ESP's pre-built, out-of-the-box examples to flash and play

Select Application

Matter-AllClustersApp-TE9

ESP Chipset Type

Matter-AllClustersApp-TE9

Matter-AllClustersApp-M5Stack-TE9

Matter-LightingApp-TE9

RainMaker-Fan

RainMaker-GPIO

RainMaker-HomekitSwitch

RainMaker-LedLight

RainMaker-MultiDevice

RainMaker-Switch

RainMaker-TemperatureSensor

Flash

Quick start

esp-matter: <https://github.com/espressif/esp-matter>

Launchpad for flashing light and light_switch:

- <https://espressif.github.io/esp-launchpad/?flashConfigURL=https://espressif.github.io/esp-matter/launchpad.toml>

Commissioning

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

Options: chip-tool, Apple Home or Google Home with proper device in local net

chip-tool interactive start

pairing ble-wifi 0x7283 SSID PASS 20202021 3840

Note: Some older Youtube videos are referring to QR code from console. The feature was removed. Please, use one from documentation.



Setting up development environment

Please, read programming guide: <https://docs.espressif.com/projects/esp-matter/en/latest/esp32/>

Warning: Follow steps exactly, small deviation may result into non-working env.

Recommended OS: macOS, Ubuntu 22 LTS

```
apt install git ninja-build clang pkg-config libglib2.0-dev libssl-dev gcc g++ cmake
```

Install GN <https://gn.googlesource.com/gn/+refs/heads/main/README.md#getting-a-binary>

... or use Development Container

Matter development env for Windows - WSL2

```
winget install usbipd
```

<https://github.com/dorssel/usbipd-win/wiki/WSL-support>

```
sudo apt install linux-tools-virtual hwddata
```

```
sudo update-alternatives --install /usr/local/bin/usbip usbip `ls /usr/lib/linux-tools/*/usbip` | tail -n1` 20
```

```
usbipd wsl attach -b 1-5 -d Ubuntu-22.04
```

Connect from VM: `usbip attach --remote 192.168.32.137 --busid 1-4`

Limitations: mDNS, BLE layer

Python Matter Server

Only Ubuntu 20 LTS

<https://github.com/home-assistant-lib/python-matter-server>

<https://nabucasa.github.io/matter-example-apps/>

matter.js - NodeJS Matter implementation

<https://github.com/project-chip/matter.js>

- matter.js: the core Matter implementation in typescript which is JavaScript only and has no native dependencies.
- matter-node.js: a node.js implementation of a Matter Device and Controller

chip-tool

Recommended option

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

Android app (devel only)

https://github.com/project-chip/connectedhomeip/blob/master/docs/guides/android_building.md

Source code

<https://github.com/project-chip/connectedhomeip/tree/master/examples/android>

Recommended approach: use chip-tool or Google Home app

Observing Matter network

```
dns-sd -B _matter._tcp
```

```
avahi-browse _matter._tcp -r
```

_matterc._udp - device waiting to get commissioned

_matter._tcp - lists the commissioned device

Observing Matter network - output

Browsing for _matter._tcp

DATE: ---Mon 17 Apr 2023---

9:28:46.609 ...STARTING...

Timestamp	A/R	Flags	if Domain	Service Type	Instance Name
9:45:28.368	Add	2	11 local.	_matter._tcp.	494022C0EA7CC12B-0000000000001B669
9:45:37.609	Add	2	11 local.	_matter._tcp.	494022C0EA7CC12B-00000000000007283
9:46:42.917	Rmv	0	11 local.	_matter._tcp.	494022C0EA7CC12B-00000000000007283
9:47:29.678	Rmv	0	11 local.	_matter._tcp.	494022C0EA7CC12B-0000000000001B669

idf.py monitor

```
> matter config
```

```
VendorId:    65521 (0xFF1)
```

```
ProductId:   32768 (0x8000)
```

```
HardwareVersion: 0 (0x0)
```

```
PinCode:     20202021
```

```
Discriminator: f00
```

```
Done
```

```
> matter esp attribute get 0x1 0x6 0x0
```

```
I (377514) esp_matter_attribute: ***** Endpoint 0x0001's Cluster 0x0006's Attribute 0x0000 is 1 *****
```

```
I (377514) esp_matter_attribute: ***** Endpoint 0x0001's Cluster 0x0006's Attribute 0x0000 is 1 *****
```

```
Done
```

Bind light and light_switch

commission light: chip-tool pairing ble-wifi 0x7283 <SSID> <passphrase> setup-pin-code discriminator

commission light_switch: chip-tool pairing ble-wifi 0x7283 <SSID> <passphrase> setup-pin-code discriminator

bind light and switch: https://github.com/espressif/esp-matter/tree/main/examples/light_switch#21-bind-light-to-switch

update ACL of light: accesscontrol write acl '[{"privilege": 5, "authMode": 2, "subjects": [112233, 29315], "targets": null}]' 0x5164 0x0

bind light_switch: binding write binding '[{"node":20836, "endpoint":1, "cluster":6}]' 0x7283 0x1



Working configuration

switch: 0x7283 (29315)

binding write binding '[{"node":20836, "endpoint":1, "cluster":6}]' 0x7283 0x1

light: 0x5164 (20836)

accesscontrol write acl '[{"privilege": 5, "authMode": 2, "subjects": [112233, 29315], "targets": null}]' 0x5164 0x0

onoff toggle 0x5164 1

Working with console

```
matter esp attribute get <endpoint_id> <cluster_id> <attribute_id>
```

```
matter esp attribute get 0x1 0x6 0x0
```

```
matter esp attribute set <endpoint_id> <cluster_id> <attribute_id> <attribute value>
```

```
matter esp attribute set 0x1 0x6 0x0 1
```

```
matter esp controller read-event <node_id> <endpoint_id> <cluster_id> <event_id>
```

Fade In/Out - using Level Control

levelcontrol step 1 100 10 0 0 0x5164 1

levelcontrol step 0 100 10 0 0 0x5164 1

Multi-Admin scenario 1.

Devices can be a part of multiple Matter networks, called as fabrics, at the same time.

Apple Home being one and chip-tool being the other in this case. You can add a device into the chip-tool fabric through the Apple Home app using the multi-admin feature of Matter.

Commission the light (OnOff server) using Apple Home. You won't be able to commission the light_switch (OnOff client) using it as it isn't supported

From the Home app for the light, there should be an option like "Add pairing". Choose that and use the payload that is displayed there to commission the light into the chip-tool fabric. Command:

```
./chip-tool pairing code 0x5164 <payload-displayed-by-Home-app>
```

Note: The host machine running the chip-tool should be connected to the same Wi-Fi network as the iPhone and light. Node id is taken as 0x5164. You can choose a different one too.

Multi-Admin scenario 2.

Commission the light_switch directly using the chip-tool. Command:

```
./chip-tool pairing ble-wifi 0x7283 <SSID> <passphrase> setup-pin-code discriminator
```

Note: The same Wi-Fi network should be used. Default value of setup-pin-code is 20202021 and discriminator is 3840. Node id is taken as 0x7283. You can choose a different one too.

Now, both light and light_switch are on the chip-tool fabric. While light is additionally on the Apple fabric.

Follow the steps here to bind them: https://github.com/espressif/esp-matter/tree/main/examples/light_switch#21-bind-light-to-switch

You can now toggle the state of light using the light_switch. We have provided CLI commands for that. Command:

```
matter esp client invoke 0x1 0x5164 0x1 0x6 0x2
```

More commands: https://github.com/espressif/esp-matter/tree/main/examples/light_switch#23-device-console

As the light is also a part of the Apple fabric, any change made to it from the light-switch would be reflected in the Home app too.

Developer documentation

Main source: <https://csa-iot.org/developer-resource/specifications-download-request/>

Simplified documentation:

<https://blog.espressif.com/matter-38ccf1d60bcd>

<https://blog.espressif.com/matter-security-model-37f806d3b0b2>

Google - Creating project for testing

<https://developers.home.google.com/matter/project/create>

Google Nest Hub - supported devices

<https://developers.home.google.com/matter/supported-devices#hubs>

matter-rs

Early development

<https://github.com/project-chip/matter-rs>

ESP chips not supported yet

Join Rust ESP32 Community Meeting to see the progress:

<https://github.com/esp-rs/rust/discussions>



Async with Embassy

Embassy: <https://github.com/embassy-rs/embassy>

Examples of Embassy on ESP32:

- https://github.com/esp-rs/esp-hal/blob/main/esp32-hal/examples/embassy_hello_world.rs
- https://github.com/esp-rs/esp-hal/blob/main/esp32-hal/examples/embassy_spi.rs
- https://github.com/esp-rs/esp-hal/blob/main/esp32-hal/examples/embassy_wait.rs

Async topic author: Juraj Sadel

Busy Looping (not using async)

- Very inefficient, blocking

```
fn main() {  
    let mut button = Button::new();  
    let mut led = Led::new();  
    loop {  
        if button.is_pressed() {  
            led.on();  
        } else {  
            led.off();  
        }  
    }  
}
```

Interrupts (not using async)

- Driven by hardware
- Pretty complex

```
static BUTTON: Mutex<Option<Button>> = Mutex::new(None);
static LED: Mutex<Option<Led>> = Mutex::new(None);
fn main() {
    LED.lock().replace(Led::new());
    BUTTON.lock().replace(Button::new());
    setup_irq(button_event); // Magic
}

fn button_event() {
    if BUTTON.lock().as_mut().unwrap().is_pressed() {
        LED.lock().as_mut().unwrap().on();
    } else {
        LED.lock().as_mut().unwrap().off();
    }
}
```

Using async

- Don't have to setting and waiting for interrupt resuming the program
- Async executor can do that for us instead
- Power efficient

```
async fn main() {  
    let mut button = Button::new();  
    let mut led = Led::new();  
    loop {  
        button.wait_changed().await;  
        if button.is_pressed() {  
            led.on();  
        } else {  
            led.off();  
        }  
    }  
}
```

Embassy (EMBedded ASync)

- We need an EXECUTOR to be able to use async
- Controlling which task should run
- Embassy consists of multiple crates (Executor, HALs, Networking,...)
- no_std
- Can be (relatively) easily extendable/configurable with other public crates
- <https://embassy.dev>

Multi-target project (PoC) - ESP32 Spooky Maze

<https://github.com/georgik/esp32-spooky-maze-game>

Idea: sharing business logic in Rust between
multiple targets

Targets: ESP32, ESP32-S2, ESP32-S3, ESP32-C3, M5Stack, Wasm and Desktop

Article: <https://georgik.rocks/rust-bare-metal-application-for-esp32-desktop-android-and-ios/>



<https://github.com/espressif/esp-box>

Rust language support

Talk: Embedded Rust on ESP32 - Juraj Michálek - Rust Linz November 2022

<https://youtu.be/OPPPdgoDBQs>



Rust Training Embedded for ESP32-C3 by Ferrous Systems

Training: <https://ferrous-systems.com/training/#package-espessif-beginner-training>

Material: <https://esp-rs.github.io/espessif-trainings/>

GitHub: <https://github.com/esp-rs/espessif-trainings>



Espressif Developer Conference 2022 - recording



https://www.youtube.com/playlist?list=PLOzvoM7_Knrc6o-n25jYuXRB2T8UKk1NU

Developer Conference 2023 - scheduled for 12-13 September - <https://devcon.espressif.com/>

Embedded World 2024

Meet us in Nuremberg, Germany



embeddedworld

Exhibition&Conference

Visit us in Brno

Espressif Systems (Czech) s.r.o.

Přízova 3, 602 00 Brno

Czechia, Europe

