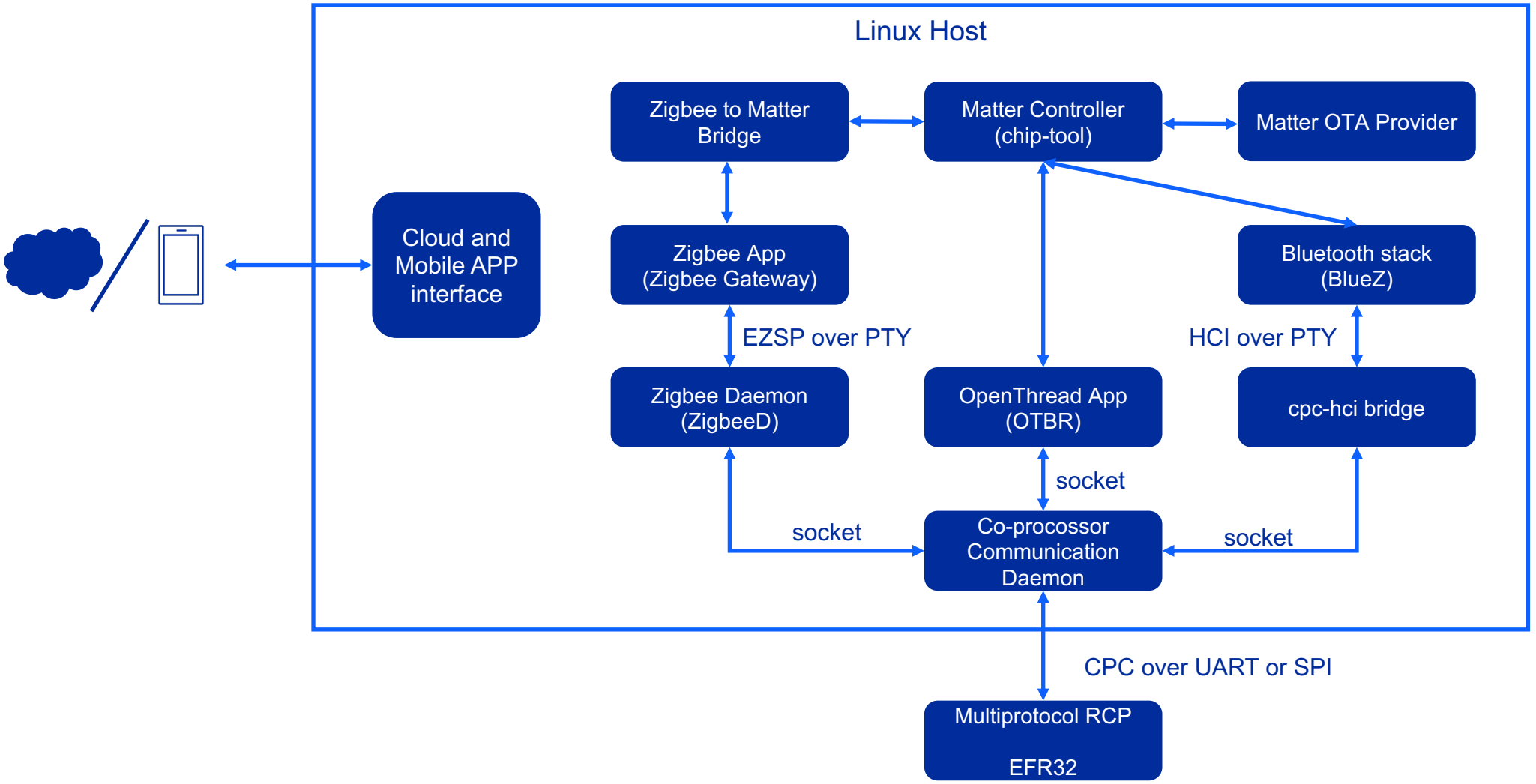
SILICON LABS

基于Linux系统和Silicon Labs的RCP架构的Matter网关

IoT Developer Services

Feb 2023

系统架构简介



Linux Host平台的开发环境

- **基于Debian Linux的版本**
  - 官方建议Ubuntu Linux平台22.04
  - 如果使用树莓派，也需要在Pi上面安装Ubuntu Server 22.04
  - ZigBee Daemon里面的ZigDee堆栈库没有开源，GSDK提供的是预编译好的arm32v7 和 arm64v8
- **Simplicity Studio 5 + GSDK**
- **安装Linux 开发所需要的依赖库：**

```
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 libreadline-dev
```

- **如果大家用树莓派，有一下几点请注意：**
  - 需要以下两个库

```
sudo apt-get install pi-bluetooth avahi-utils
```

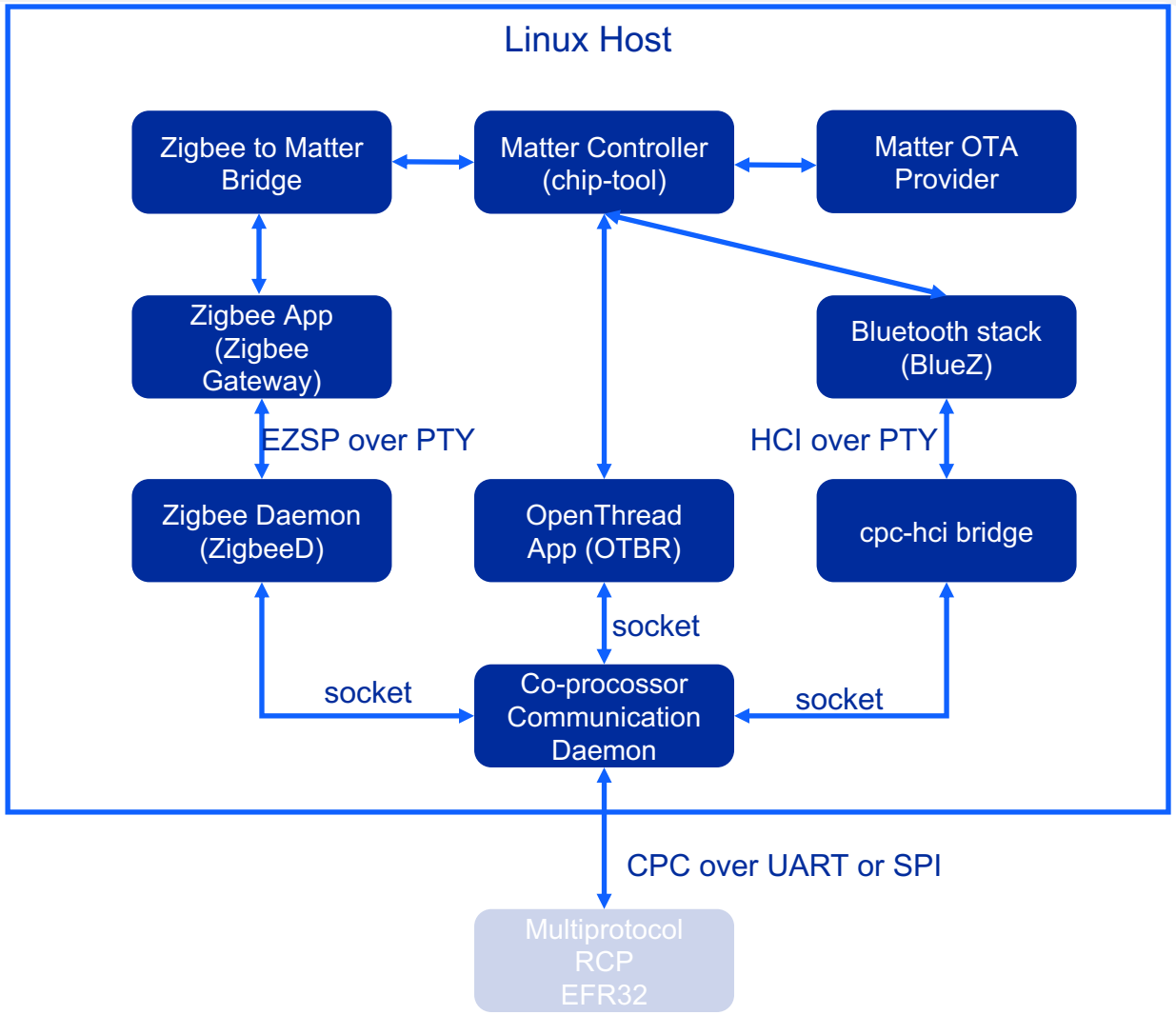
  - 停用swapfile功能

```
sudo dphys-swapfile swapoff
sudo dphys-swapfile uninstall
sudo apt remove dphys-swapfile
```

  - 如果要跑OTBR，需要加载ip6table_filter module

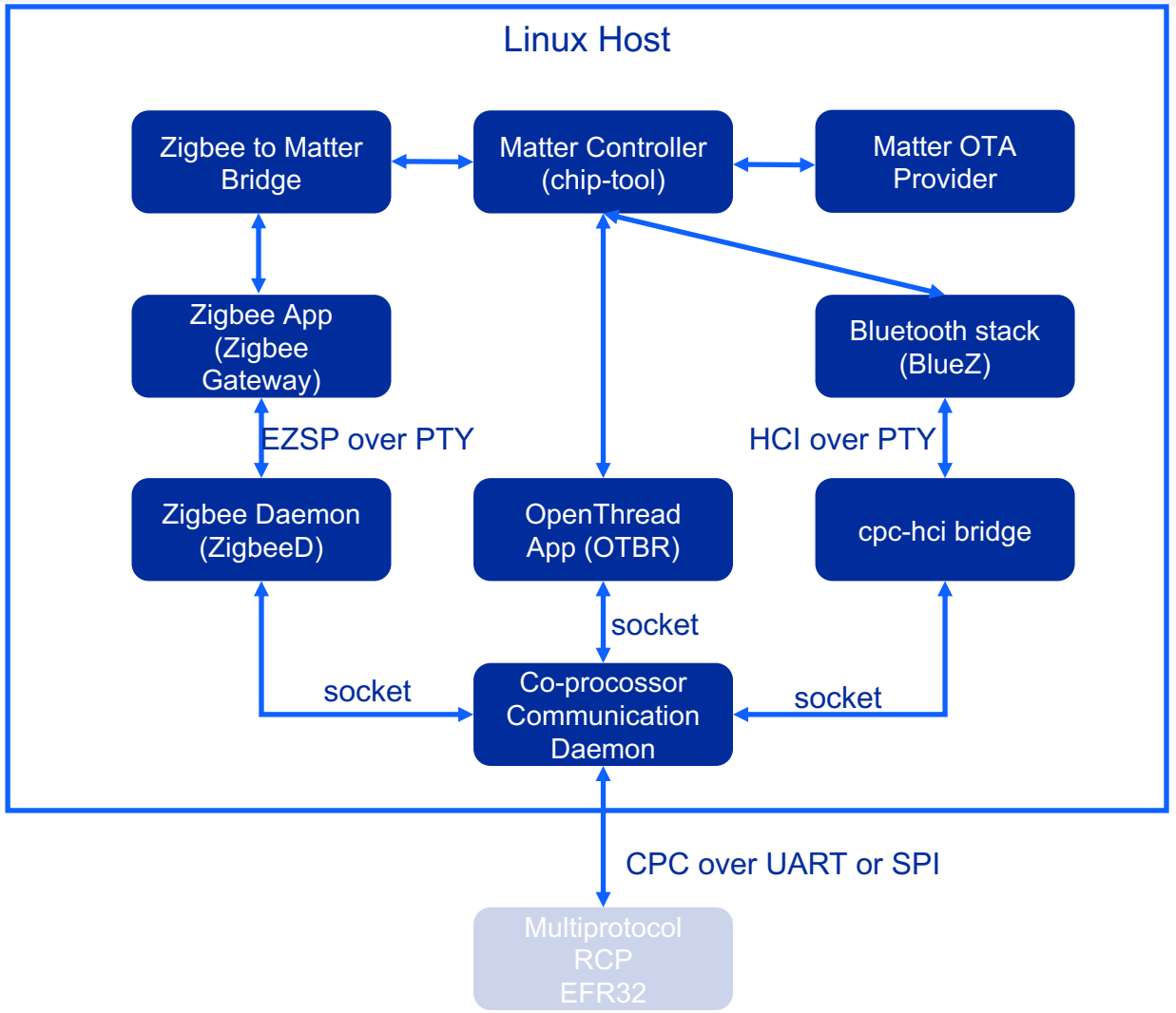
```
sudo modprobe ip6table_filter
```

或者把ip6table_filter 加到 */etc/modules*



Linux Host平台的开发环境

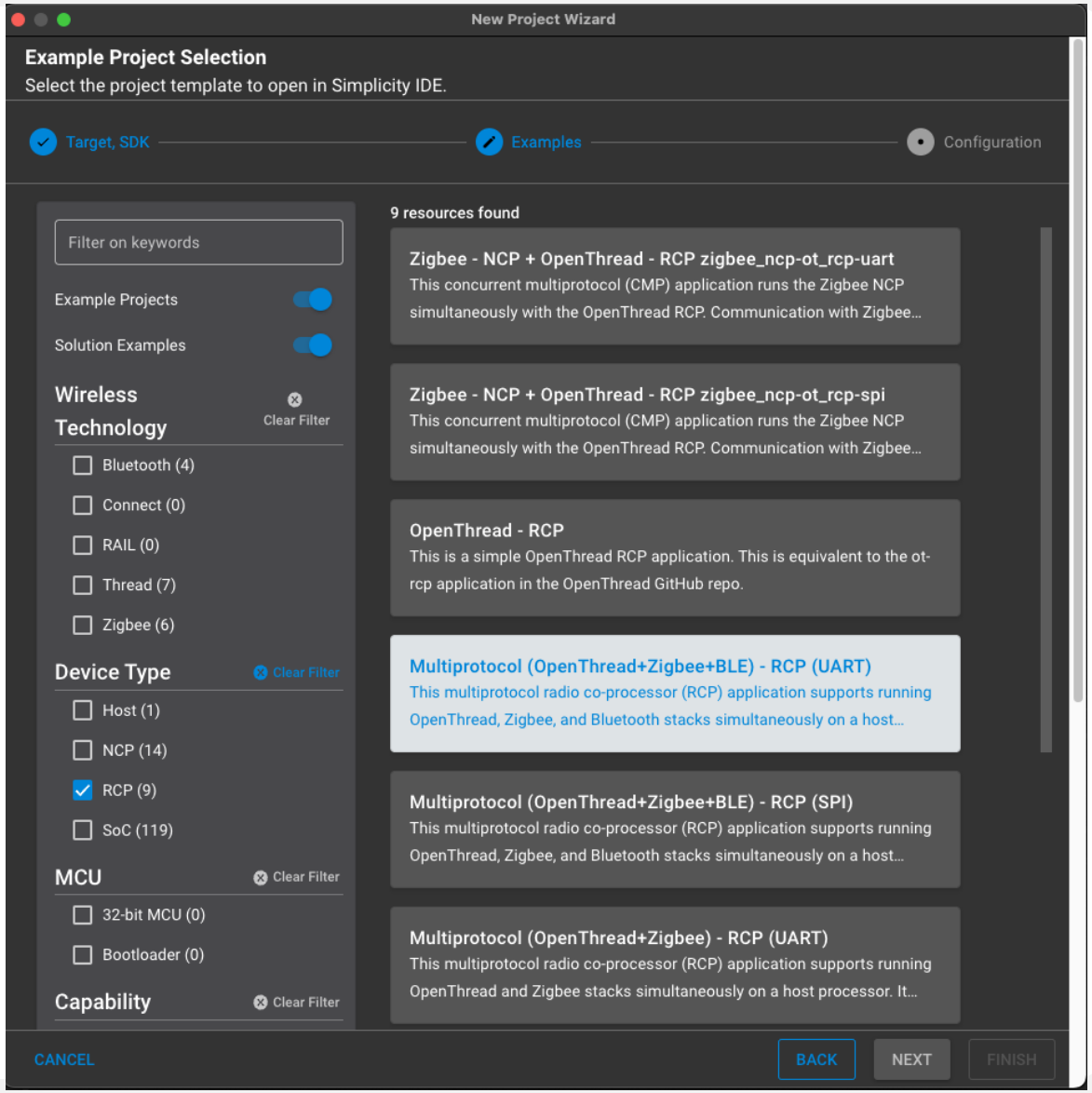
- **安装初始化Matter SDK**
  - git clone https://github.com/project-chip/connectedhomeip
  - cd connectedhomeip
  - git checkout <commit hash> // check the Matter Getting started guide page.
  - git submodule update –init
  - cd connectedhomeip
  - source scripts/activate.sh
  - 详情参考： https://github.com/project-chip/connectedhomeip/blob/master/docs/guides/BUILDING.md



EFR32的多协议RCP固件

- 建基于Silicon Labs RCP加上的多协议支援
- 支援Silicon Labs的CPC(Co-Processor Communication)通信协议
- 同时支援蓝牙的多协议运行
  - FreeRTOS
  - 把蓝牙的HCI信息通过CPC透传到HOST端
  - 固件大小~250K
  - 在Simplicity Studio 5下面相关的项目档是rcp-uart-802154-blehci.slcp 或 rcp-spi-802154-blehci.slcp
- 具体操作
  - File > New > Silicon Labs New Project Wizard
  - 选NEXT，到Example Project Selection界面
  - 如果要编译RCP固件，勾选RCP，然后适合的项目，例如：

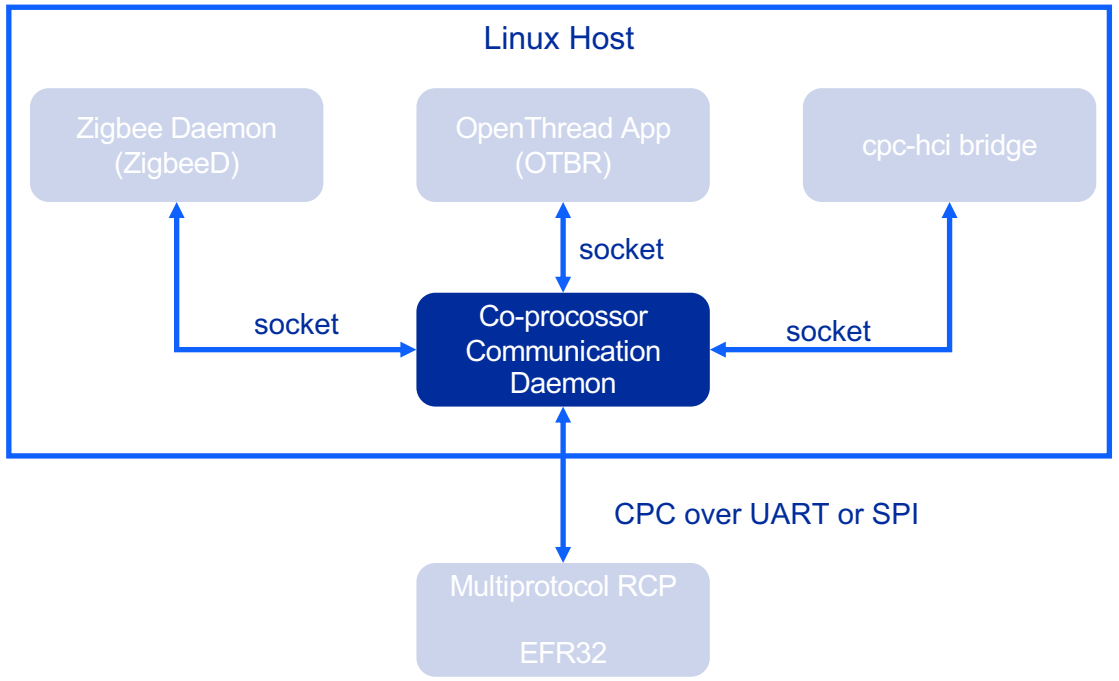
    Multiprotocol(Open Thread + Zigbee + BLE) - RCP(UART)



Co-Processor Communication Daemon

- **后台运行**
- **负责与RCP或者NCP硬件进行通信**
  - 可以同时和多个堆栈协议通过socket进行通信
  - 纠错和保证信息包的时序
- **CPCd 包括以下三部分**
  - **Cpcd执行档;**
  - **libcpc.so库，可以让其他C语言的应用来调用**
  - 配置文档 (cpcd.conf)
- **开源**
  - https://github.com/SiliconLabs/cpc_daemon
- **Building CPCd**
  - 根据 https://github.com/Sili-conLabs/cpc-daemon/blob/main/readme.md 来编译CPCd
  - 在执行make install后，再执行ldconfig来更新系统数据库
  - CPCd需要以下依赖`libmbedtls`

    ```
    sudo apt-get install libmbedtls-dev
    ```
  - 以树莓派为例，make install会把编出来的libcpc.so 放到 */usr/local/lib/*arm-linux-gnueabihf，而sl_cpc.h在 */usr/local/include*.

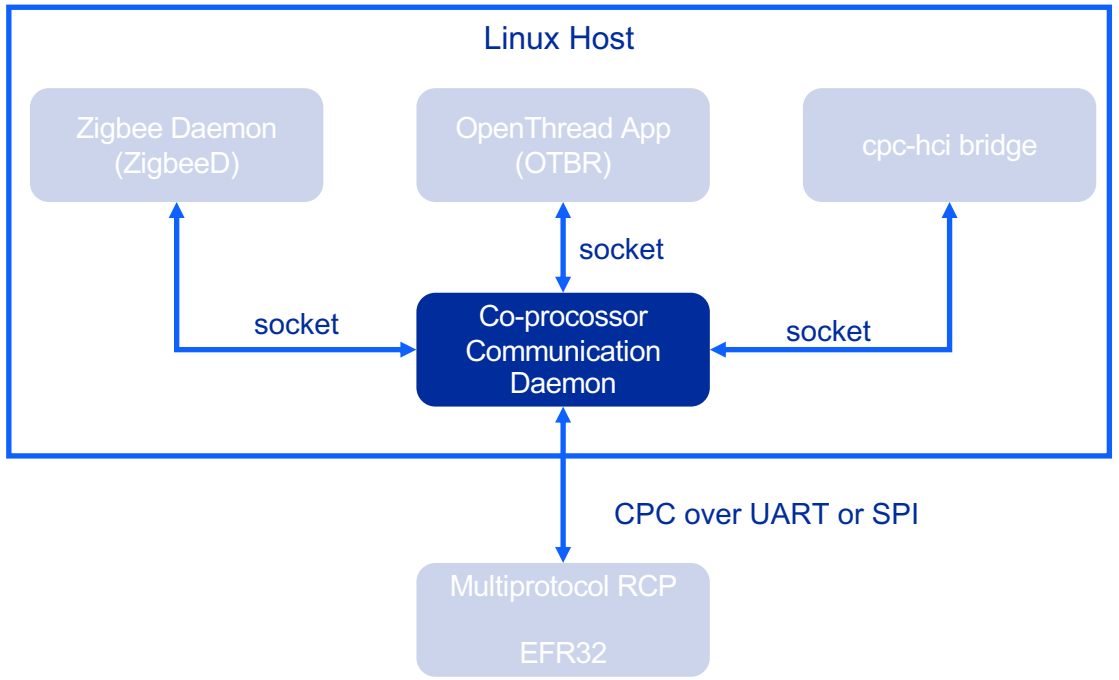


*参考：an1333-concurrent-protocols-with-802-15-4-rcp

*参考：an1351-using-co-processor communication_daemon

Co-Processor Communication Daemon

- **CPCd 设定**
  - 修改*cpcd.conf* 设定档（*/usr/local/etc/cpcd.conf* ）
  - 选择UART或SPI
  - 设定STDOUT_TRACE
- **CPC安全设定**
  - 在cpcd.conf里面默认开启
  - CPCd和RCP之间的通信是加密了的
  - 系统初始设置的时候，就需要有一个绑定的过程
  - 绑定可以通过脚本来完成
  - See https://github.com/SiliconLabs/cpc-daemon/blob/main/readme.md for instructions.
- **启动CPCd: `/usr/local/bin/cpcd`.**



*参考：an1333-concurrent-protocols-with-802-15-4-rcp

*参考：an1351-using-co-processor communication_daemon

OpenThread 边际路由（Border Router）

- **如何编译一个适用于Silicon Labs多协议架构的OTBR**
  - 在Simplicity Studio 5里的GSDK，包括了合用的OpenThread和OTBR版本
  - gecko_sdk/util/third_party/ot-br-posix
  - gecko_sdk/util/third_party/openthread
- **用cmake在GSDK的util/third_party/ot-br-posix目录下编译otbr-agent**
  - `OTBR_OPTIONS="-DOT_MULTIPAN_RCP=ON -DOT_POSIX_CONFIG_RCP_BUS=CPC" ./script/setup`
- **基于CPCd的OTBR需要有以下的定义**

```
OPENTHREAD_CONFIG_MULTIPAN_RCP_ENABLE=1
OPENTHREAD_CONFIG_RCP_BUS=OT_POSIX_RCP_BUS_CPC
OPENTHREAD_SPINEL_CONFIG_RCP_RESTORATION_MAX_COUNT=10
OPENTHREAD_POSIX_CONFIG_MAX_POWER_TABLE_ENABLE=0
OPENTHREAD_CONFIG_CHANNEL_MANAGER_ENABLE=0
OPENTHREAD_CONFIG_CHANNEL_MONITOR_ENABLE=0
```

- **如果出现以下报错：**

```
-- Looking for sl_cpc.h
-- Looking for sl_cpc.h - not found
-- CPC is not installed.
CMake Error at third_party/openthread/repo/src/posix/platform/CMakeLists.txt:146 (message):
Could not locate CPC daemon sources.
Please define the CMake variable 'CPCD_SOURCE_DIR'.
'CPCD_SOURCE_DIR' is an absolute path to the CPC Daemon sources
-- Configuring incomplete, errors occurred
```

  - sl_cpc.h has not been installed in */usr/local/include*
  - 把`-DCPCD_SOURCE_DIR=<path to cpc-daemon>`加到`OTBR_OPTIONS`.
- **编译ot-cli**

```
./bootstrap && make -f src/posix/Makefile-posix RCP_BUS=cpc MULTIPAN_RCP=1 RCP_RESTORATION_MAX_COUNT=10
MAX_POWER_TABLE=0 CHANNEL_MANAGER=0 CHANNEL_MONITOR=0
```

*参考：an1333-concurrent-protocols-with-802-15-4-rcp

蓝牙支援

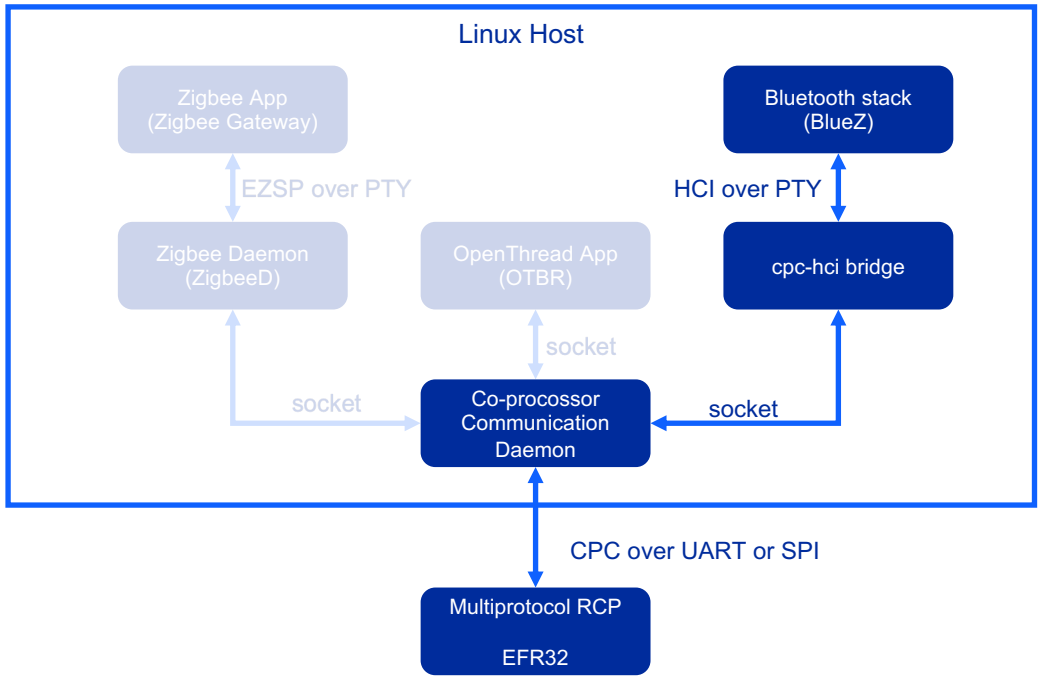
- **桥接程序cpc-hci-bridge**
  - 代码在GSDK里面的位置
  - gecko_sdk/app/bluetooth/example/example_host/bt_host_cpc_hci_bridge
- **在Zigbee+OpenThread+BLE RCP 架构上运用BlueZ堆栈**
  - 一般Linux系统都有BlueZ堆栈

```
sudo apt-get install bluetooth bluez bluez-tools rfkill libbluetooth-dev
```

- **用以下命令开始蓝牙服务**

```
service bluetooth start.
```

- **启动CPCd**
  - 却认CPCd已经成功连接RCP的EFR32，如果成功连接的话会出现“Daemon connected successfully”的打印信息.



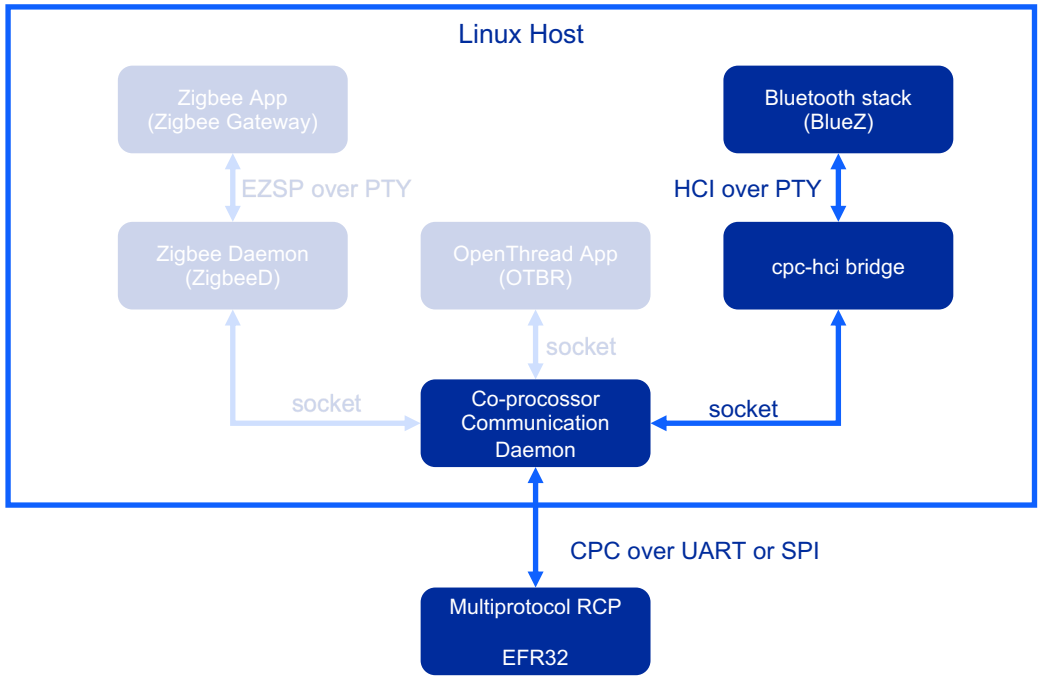
*参考：an1333-concurrent-protocols-with-802-15-4-rcp

蓝牙支援

- **启动 cpc-hci-bridge**
  - 用cpcd_0为参数启动cpc-hci-bridge
- **启动后cpc-hci-bridge会**
  - Multiprotocol RCP的EFR固件里打开一个CPC endpoint
  - 在Linux系统里面生成一个虚拟设备，例如 /dev/pts/2.
  - cpc-hci-bridge还会在现在的工作目录下建立一个 pts_hci 的连接（symlink）到这个虚拟设备
- **用hciattach把蓝牙堆栈挂上刚刚生成的虚拟设备**

```
sudo hciattach <device> any
```

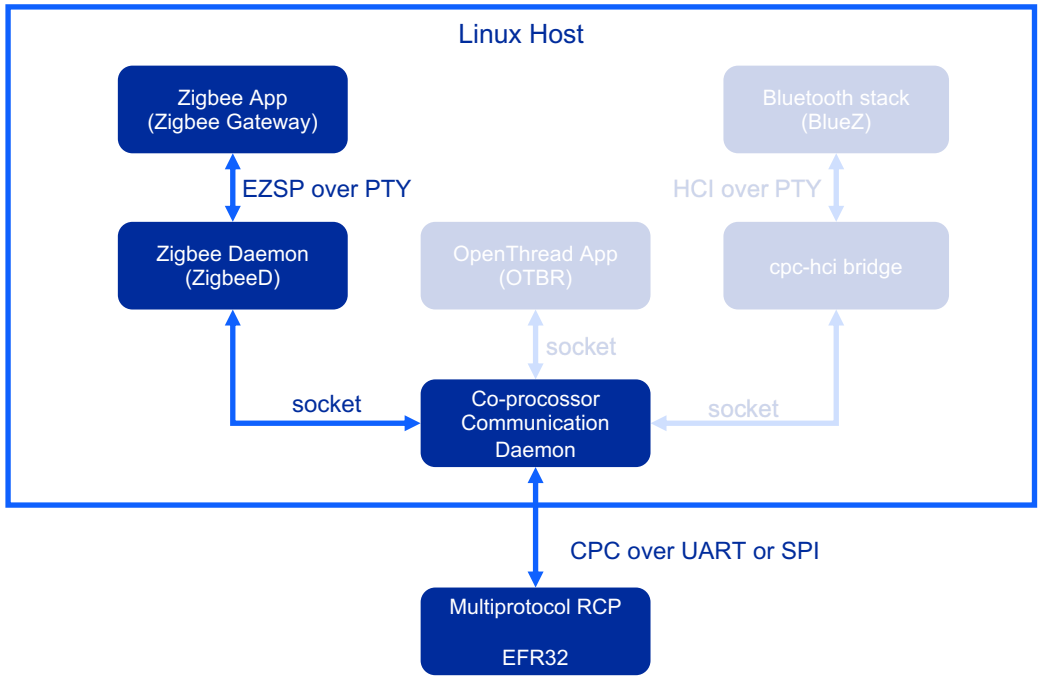
- **最后用`sudo bluetoothctl` 来启动蓝牙CLI工具**
- **另外还有一个比较有用的蓝牙工具叫`btmon`，可以监控来往蓝牙设备的数据**



*参考：an1333-concurrent-protocols-with-802-15-4-rcp

在CPCd+RCP架构上的ZigBee网关

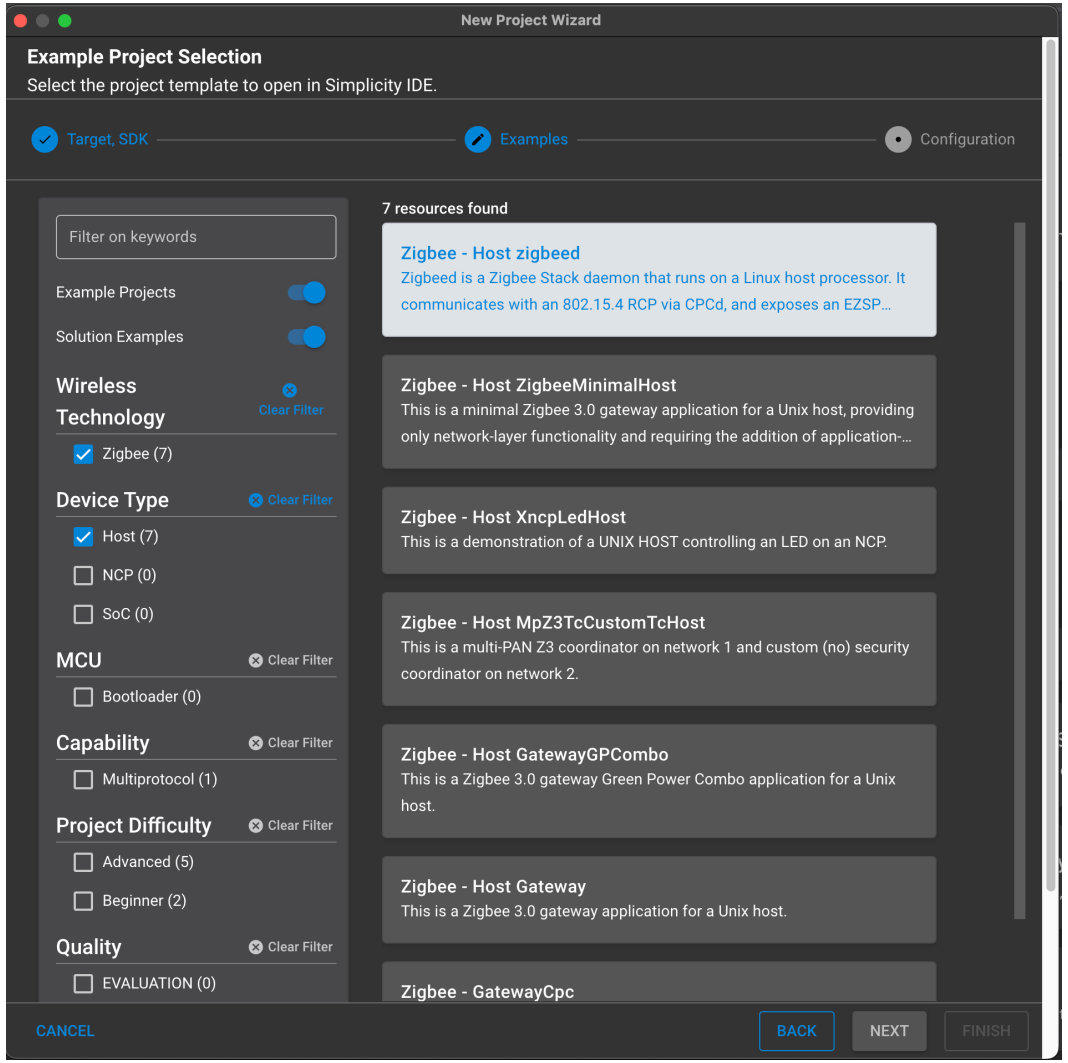
- **ZigbeeD可以从GSDK里面编译**
- **ZigbeeD依赖Silicon Labs ZigBee堆栈库**
  - Silicon Labs的ZigBee堆栈库目前是不开源的，GSDK有适用于ARM32v7和ARM64v8版本的堆栈库
  - 如果需要X86或其他架构请联络我们的销售团队
- **ZigbeeD在后台运行**
- **通过虚拟串口和上层的ZigBee网关通信，往下和CPCd连接**
- **在网关看来，它和NCP一样**



SILICON LABS

ZigbeeD的编译

- **编译需求**
  - Gecko SDK, the suite of Silicon Labs SDKs, including Zigbee, OpenThread, and Bluetooth
  - CPCd要编好并且安装好。“https://github.com/SiliconLabs/cpc-daemon”
- **通过Simplicity Studio 的用户界面编译**
  - File > New > Silicon Labs New Project Wizard
  - 选择Target Device，Linux 32 bit 或 Linux 64 bit > NEXT
  - zigbeed.slcp项目档在GSDK里面的路径： <GSDK Installation>/protocol/zigbee/app/zigbeed/zigbeed.slcp.



ZigbeeD的编译

- **用SLC-CLI工具在命令行生成项目和编译**
  - 安装Silicon Labs Configurator CLI tool (SLC-CLI)。如果客户希望订制自己的ZigbeeD项目， 这个工具可以帮助生成项目档 “*UG520: Software Project Generation and Configuration with SLC-CLI* for more information.”
  - 假设我们在PC上生成Makefile，然后在另外一个目标平台上编译
    - 预设架构为linux_arch_64 也就是arm64v8. 如果客户希望build一个arm32v7的架构，请选linux_arch_32。
    - 生成Makefile：

      ```
      slc generate -cp -s=../.. -p=app/zigbeed/zigbeed.slcp -d=app/zigbeed/output
      ```
      (arm64v8)

      ```
      slc generate -cp -s=../.. -with=linux_arch_32 -p=app/zigbeed/zigbeed.slcp -d=app/zigbeed/output
      ```
      (arm32v7)
    - 把整个生成的目录拷贝目标平台上去，然后就可以编译

      ```
      make -f zigbeed.Makefile
      ```

ZigbeeD设定

- **Zigbee网关要和ZigbeeD连接，就需要socat工具**

```
sudo apt-get install socat
```

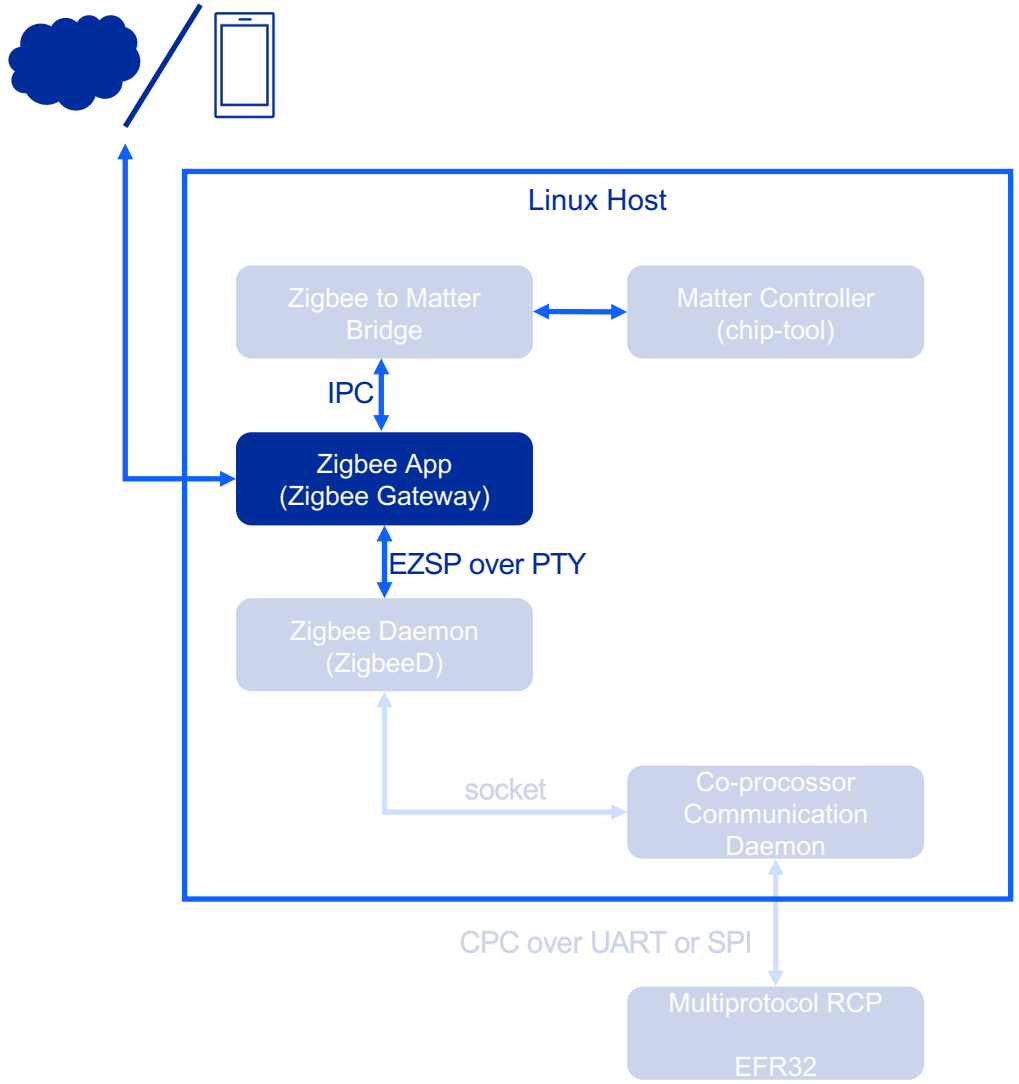
- **用socat生成两个相连的PTY设备，一个给ZigbeeD自己用，一个给Zigbee网关用**

```
socat -x -v pty,link=/dev/ttyZigbeeNCP pty,link=/tmp/ttyZigbeeNCP
```

- **启动ZigbeeD: `/usr/local/bin/zigbeed`**

ZigBee网关的开发

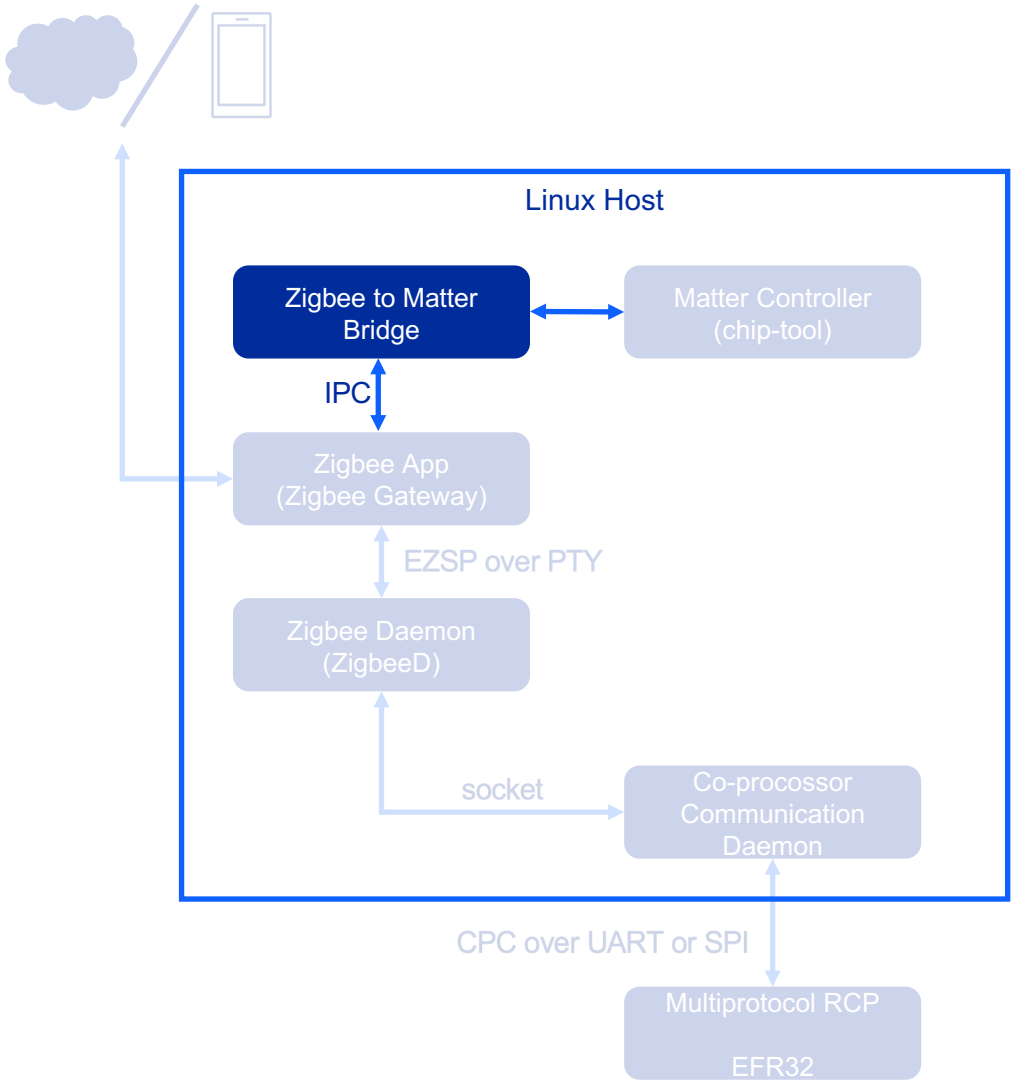
- **ZigBee网关应该具备的功能**
  - 网络生成
  - 设备入网
  - 设备离网
  - 设备状态上报和控制等等
  - 和云端，用户手机APP通信
- **ZigBee网关通过 `/dev/ttyZigbeeNCP`和`ZigbeeD`进行通信**
- **ZigBee网关可以通过IPC和Matter的桥接程序通信**
  - 定义通信的内容



Matter Bridge的开发

- **Matter Github里面的Dynamic Bridge App**
  - 预先定义好的Endpoints，用来模拟桥接的非Matter设备

- **基于Dynamic Bridge App， 需要什么改动才能够把一个真正的ZigBee 网络桥接进来？**
  - 和ZigBee网关的通信机制
  - 新的ZigBee设备入网
  - 更新Bridge App的Endpoints
  - 让用户或者控制中心知道有新的非Matter设备加了进来，需要从新获取Bridge App的Endpoints
  - 处理绑定关系



在Matter SDK上编译Chip-tool和 OTA-provider

- **CHIP-TOOL**
  - cd ~/connectedhomeip
  - ./scripts/examples/gn_build_example.sh examples/chip-tool out/standalone
  - 完成以后一个chip-tool的可执行档就会生成在 **connectedhomeip/out/standalone**
  - 参考https://github.com/project-chip/connectedhomeip/tree/master/examples/chip-tool

- **OTA-PROVIDER**
  - cd ~/connectedhomeip
  - scripts/examples/gn_build_example.sh examples/ota-provider-app/linux out/debug chip_config_network_layer_ble=false
  - 参考 https://github.com/project-chip/connectedhomeip/tree/master/examples/ota-provider-app/linux