zbhci

Release 0.1.0

lewin

Feb 10, 2023

CONTENTS

CHAPTER

ONE

GETTING STARTED

1.1 Requirement

In order to use Zigbee2MQTT, we need the following hardware:

1. LilyGo Zigbee2MQTT
2. A server that can run MQTT broker and Home Assistant (e.g. Raspberry Pi)
3. One or more Zigbee devices that will be paired with Zigbee2MQTT.

1.2 Set up and start Zigbee2MQTT

1.2.1 Connect to MQTT server

See *zigbee2mqtt*.

1.3 Connect the device

Search for supported devices for your device and follow the instructions to pair.

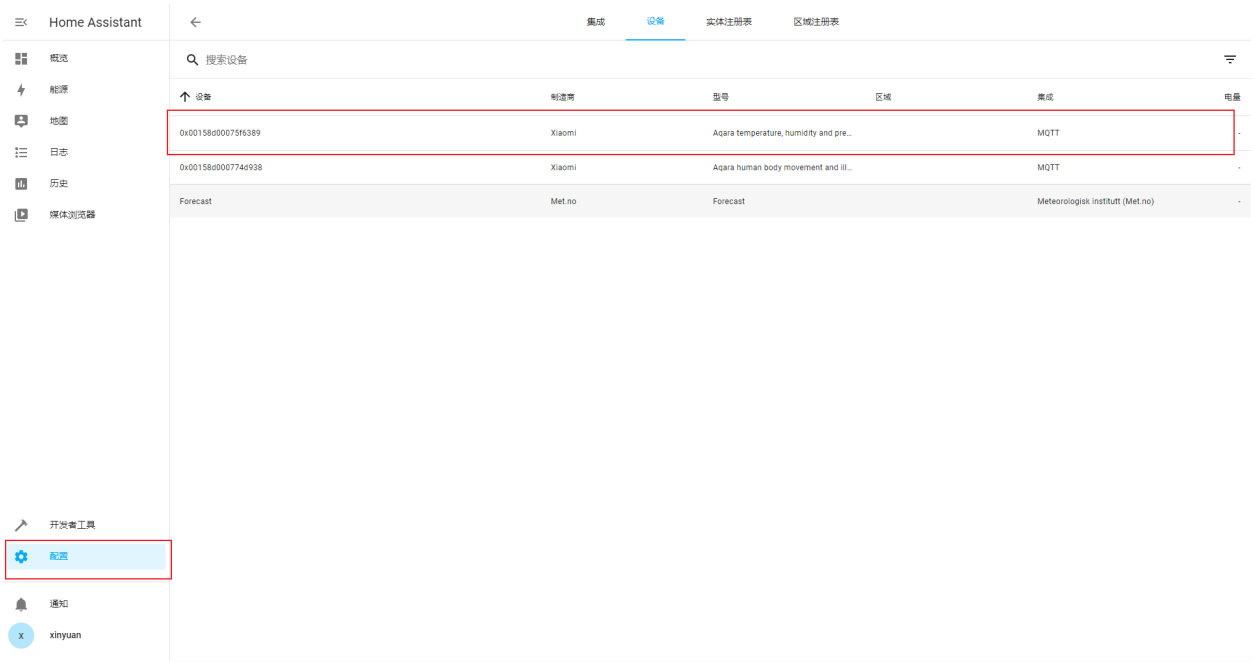
Once you see something similar to the following in the log, your device is paired and you can start to control it using the front end and MQTT messages.

```
I (276254) Zigbee2MQTT: Successfully interviewed '0x00158d000774d938', device has␣
↪successfully been paired
```

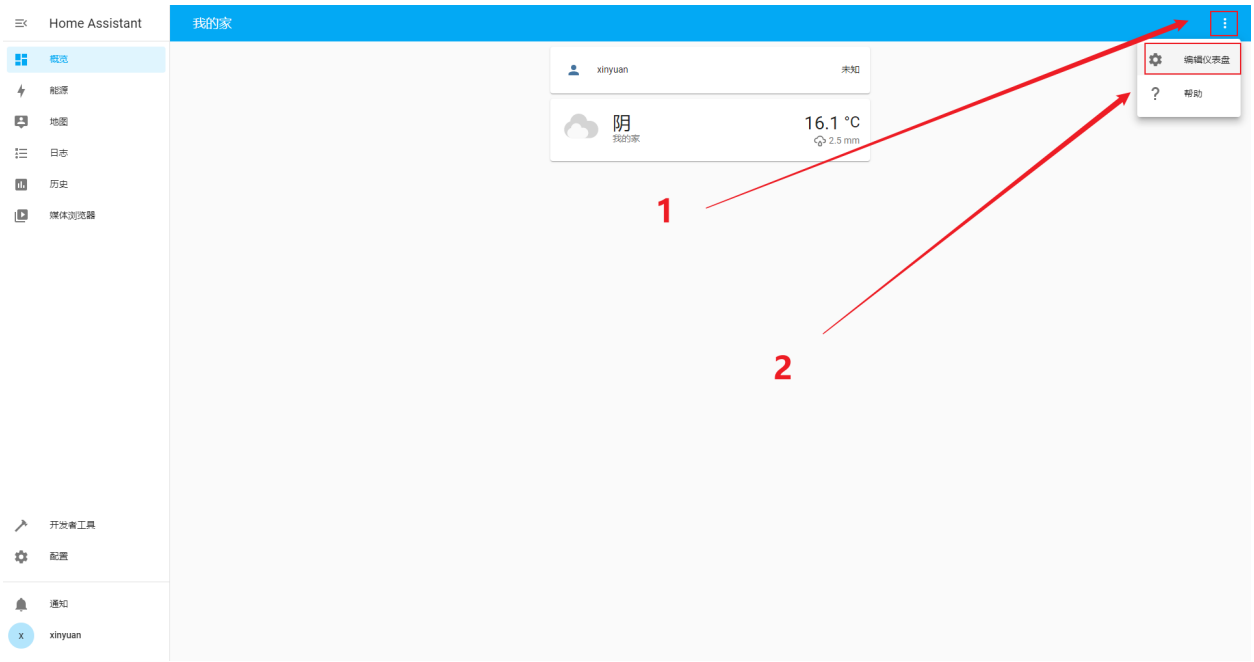
1.4 Home Assistant

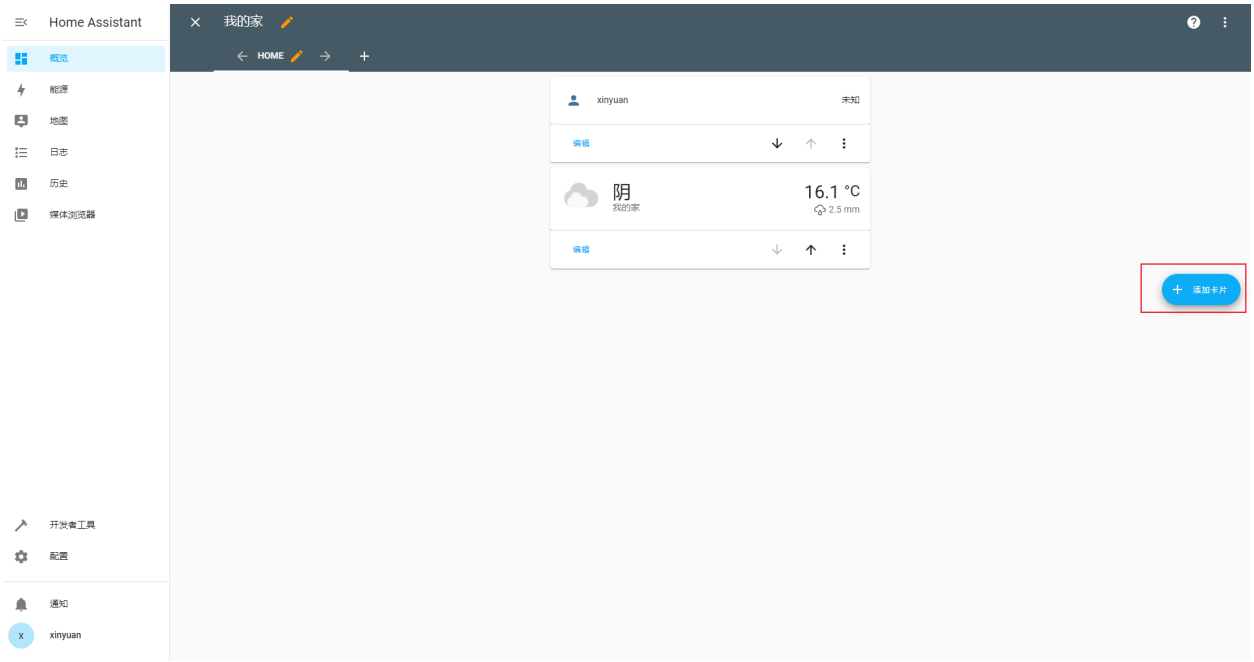
Note: Home Assistant integrates MQTT, please refer to *Home Assistant*

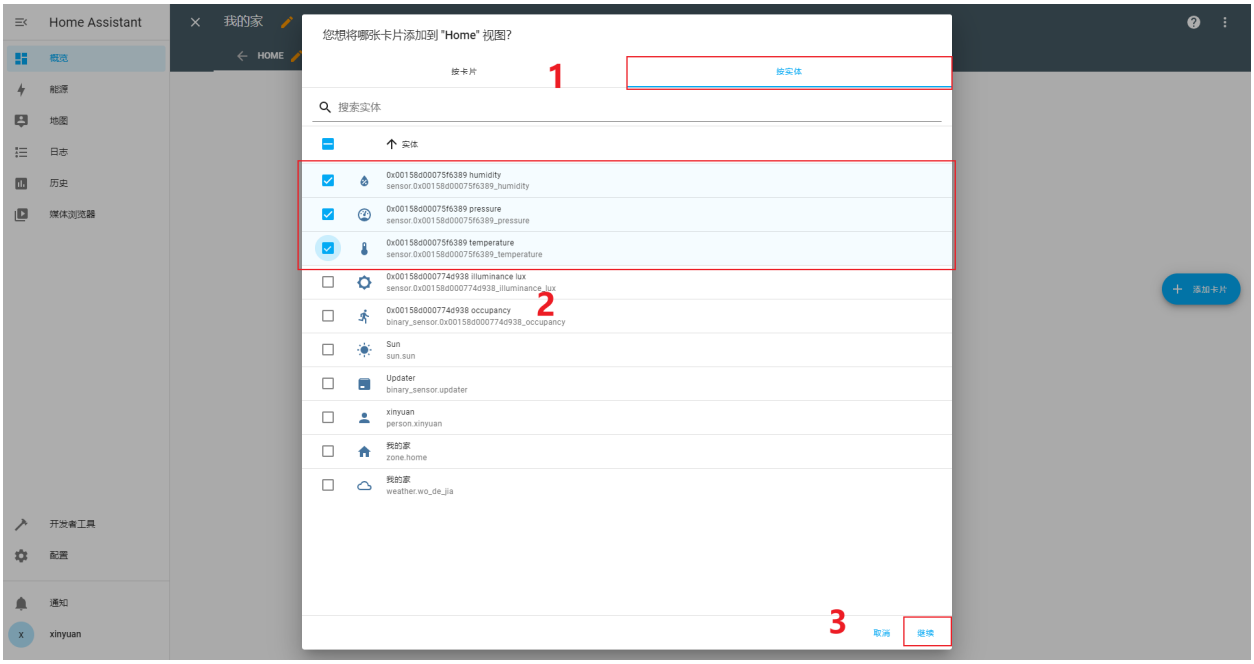
After Home Assistant integrates MQTT, the device information will be reported to Home Assistant

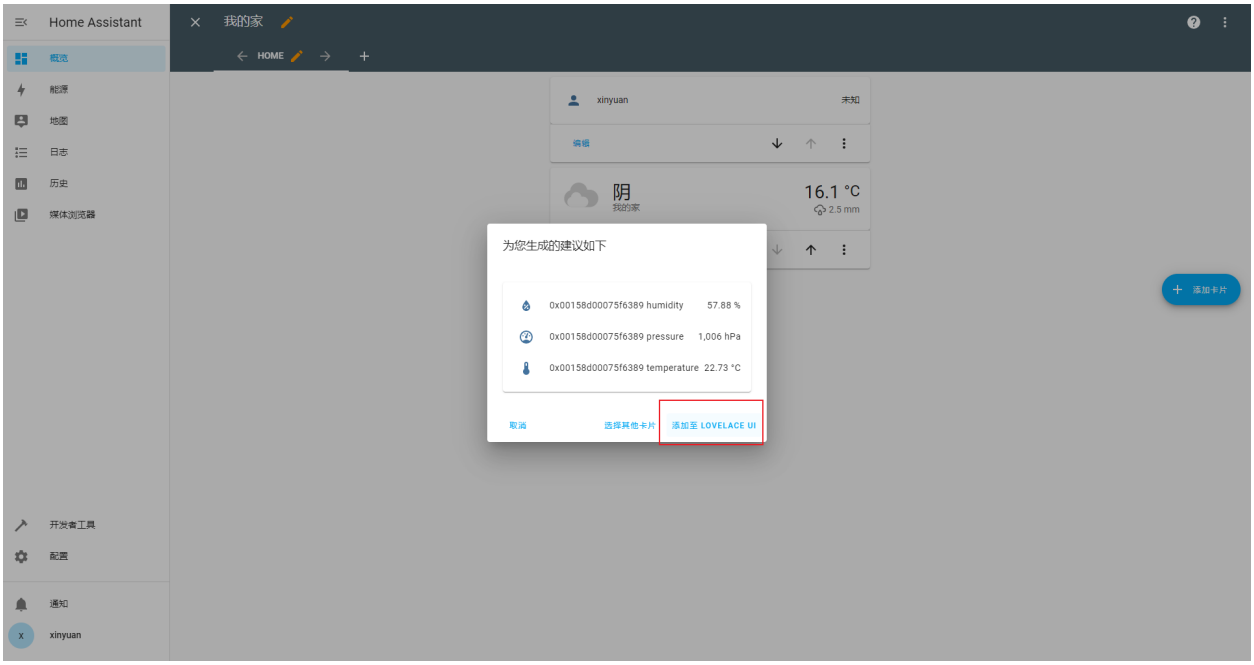


Follow the steps below to add devices to the homepage

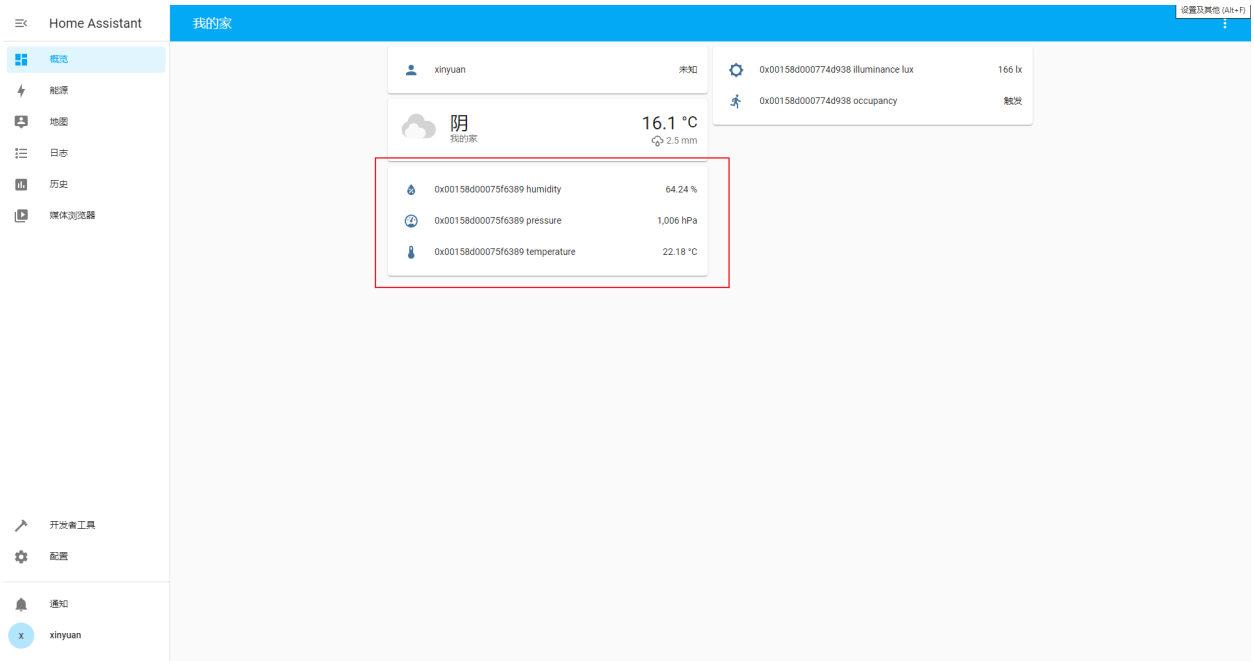



Home Assistant
我的家
HOME
概览
能源
地图
日志
历史
媒体浏览器
xinyuan
未知
编辑
阴
我的家
16.1 °C
2.5 mm
编辑
添加卡片
开发者工具
配置
通知
xinyuan


Home Assistant
我的家
HOME
概览
能源
地图
日志
历史
媒体浏览器
您想将哪张卡片添加到 "Home" 视图?
按卡片
1
按实体
搜索实体
实体
0x00158d00075f6389 humidity
sensor.0x00158d00075f6389_humidity
0x00158d00075f6389 pressure
sensor.0x00158d00075f6389_pressure
0x00158d00075f6389 temperature
sensor.0x00158d00075f6389_temperature
0x00158d000774d938 illuminance lux
sensor.0x00158d000774d938_illuminance_lux
0x00158d000774d938 occupancy
binary_sensor.0x00158d000774d938_occupancy
2
Sun
sun.sun
Updater
binary_sensor.updater
xinyuan
person.xinyuan
我的家
zone.home
我的家
weather.wo_de_jia
3
取消
继续
添加卡片
开发者工具
配置
通知
xinyuan



The addition is successful, you can see the changes in the measurement data in real time



CHAPTER

TWO

USER GUIDE

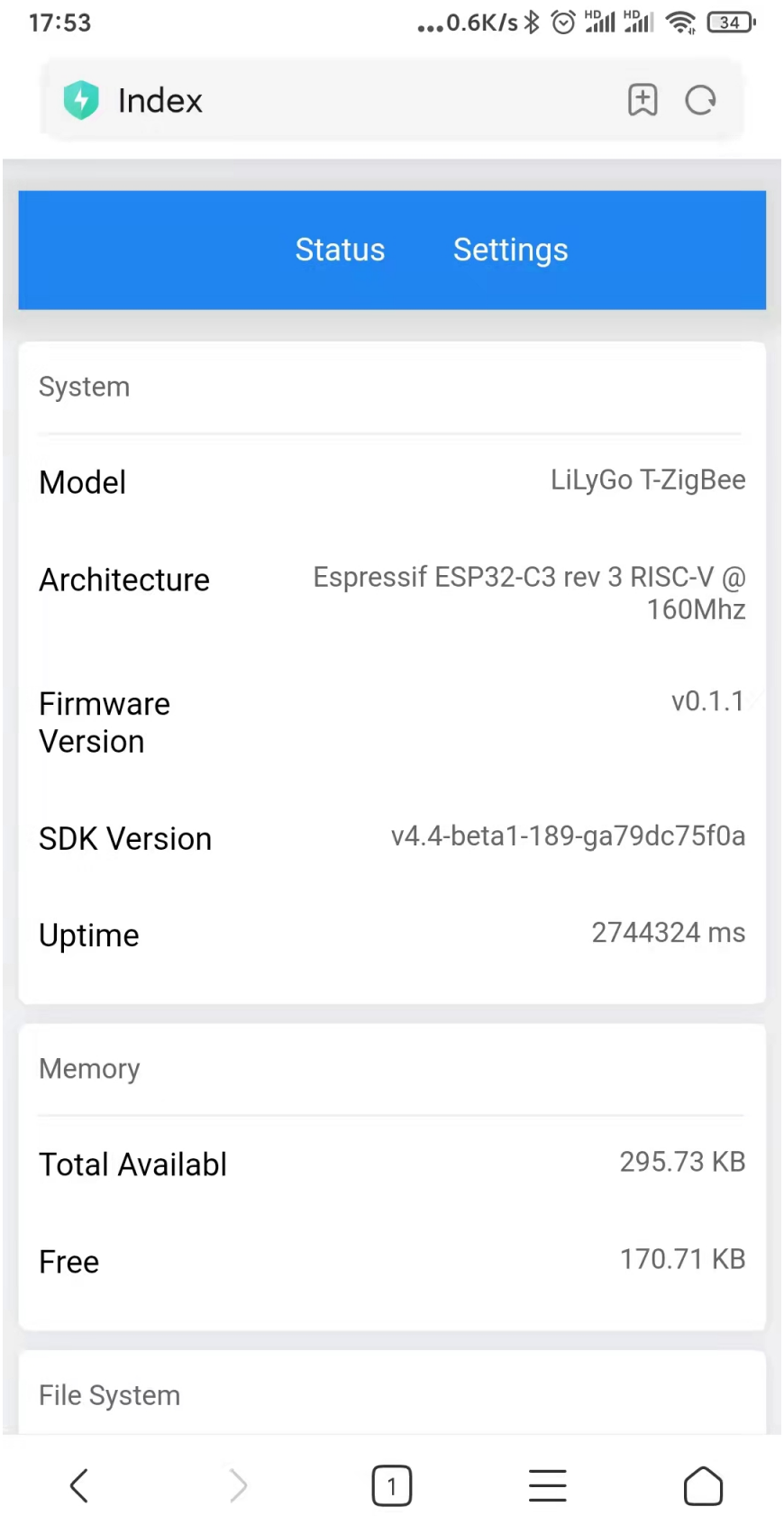
2.1 zigbee2mqtt

2.1.1 Configure

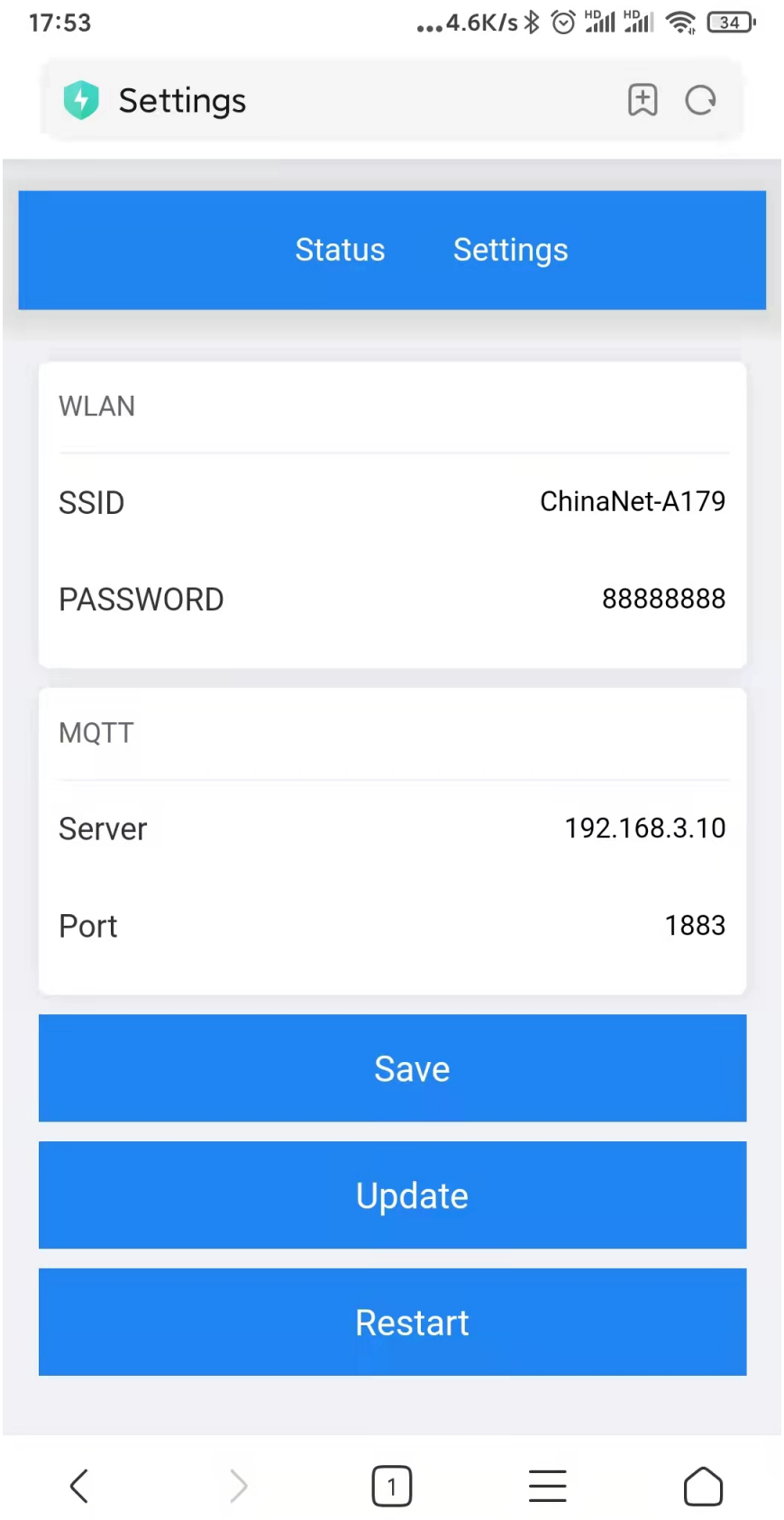
- Connect to the hotspot sent by `LilyGo Zigbee2MQTT`, for example `LilyGo-5090`


17:53
...0.3K/s
34
WLAN
开启WLAN
WLAN助理
LilyGo-5090
已连接，但无法访问互联网
选取附近的WLAN
ChinaNet-A179
ChinaNet-A179-5G
5G
xinyuan-2
DIRECT-D3F62F7A
ysw-office
ysw-office3
2.4G/5G

- Visit `192.168.4.1` in the browser to enter the configuration page


17:53
...0.6K/s
34
Index
Status
Settings
System
Model
LiLyGo T-ZigBee
Architecture
Espressif ESP32-C3 rev 3 RISC-V @ 160Mhz
Firmware Version
v0.1.1
SDK Version
v4.4-beta1-189-ga79dc75f0a
Uptime
2744324 ms
Memory
Total Availabl
295.73 KB
Free
170.71 KB
File System
1

- Configure the WiFi and mqtt server that need to be connected


17:53
4.6K/s
Settings
Status
Settings
WLAN
SSID
ChinaNet-A179
PASSWORD
88888888
MQTT
Server
192.168.3.10
Port
1883
Save
Update
Restart

After the wifi and mqtt server are successfully connected, the blue indicator on `LilyGo Zigbee2MQTT` will be always on.

2.1.2 Button Behavior

click	Prohibit zigbee sub-devices from joining the gateway
double click	Allow zigbee sub-devices to join the gateway

2.1.3 indicator light

red light

extinguish	zigbee is off
Bright	zigbee is running

green light

extinguish	Prohibit zigbee sub-devices from joining the gateway
Bright	Allow zigbee sub-devices to join the gateway

blue light

extinguish	run error
Bright	normal operation
flash	Not connected to WiFi
slow flash	Connected to WiFi, but not connected to mqtt server

2.2 mosquitto

2.2.1 apt

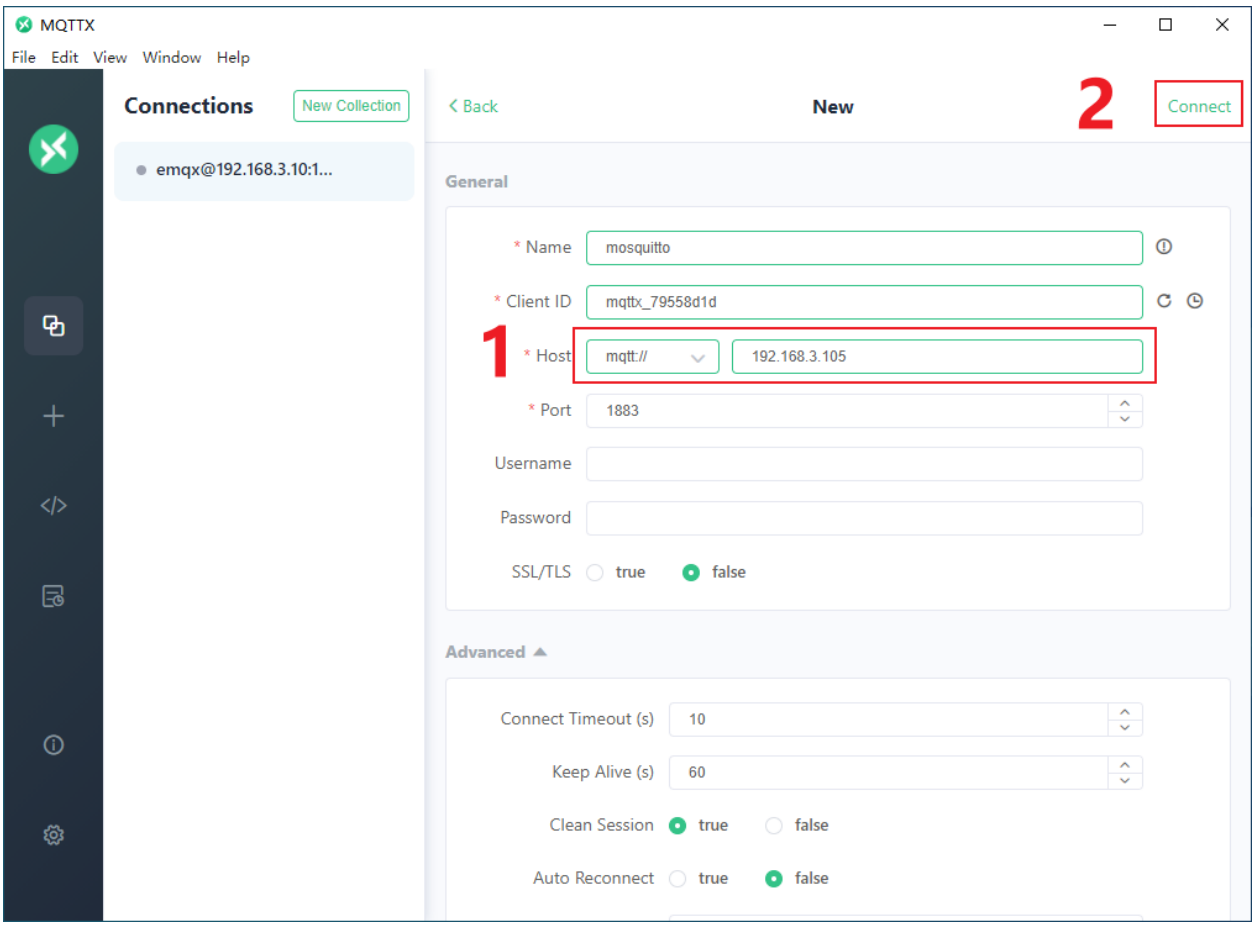
```
sudo apt-get update
sudo apt-get install mosquitto
```

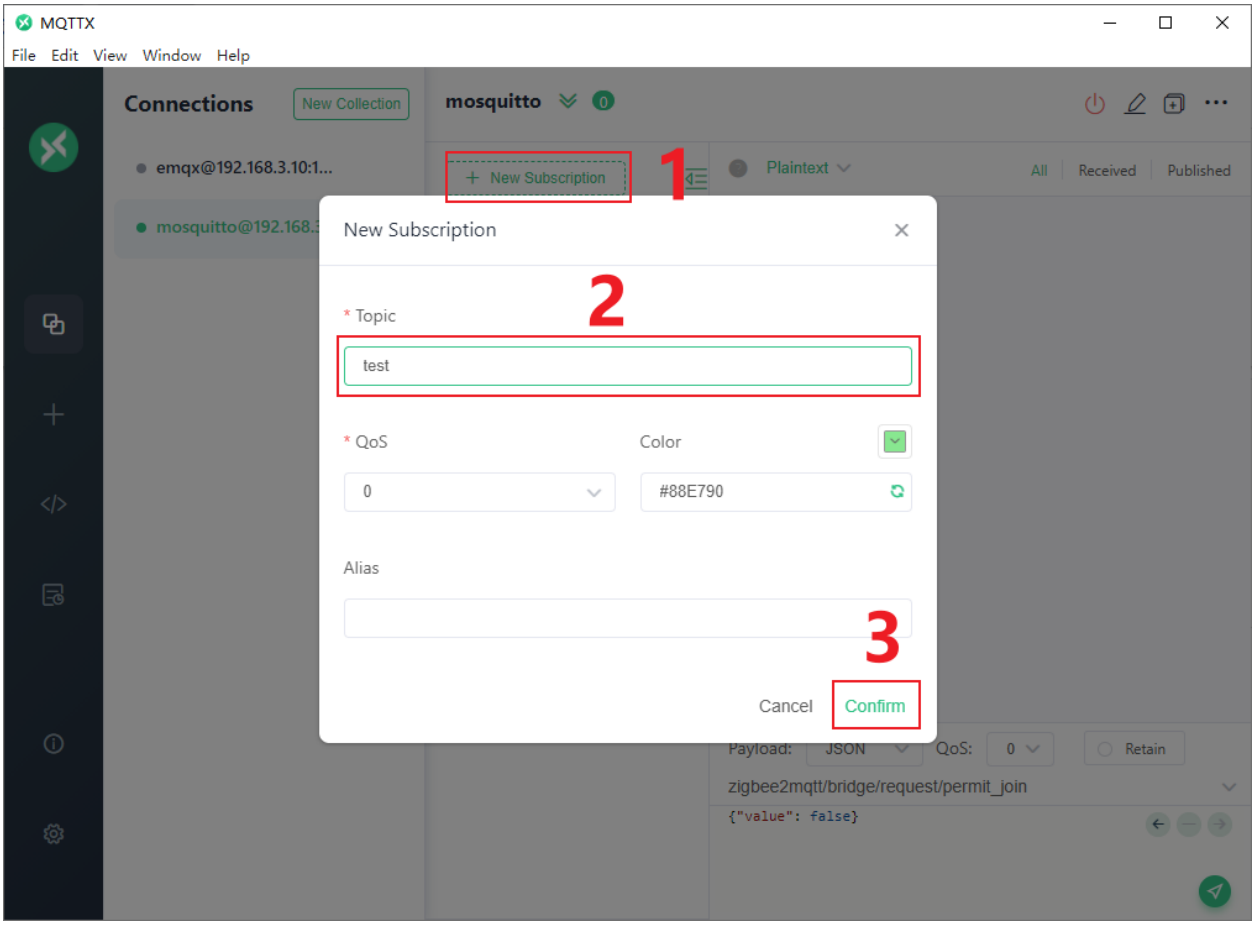
2.2.2 Configure

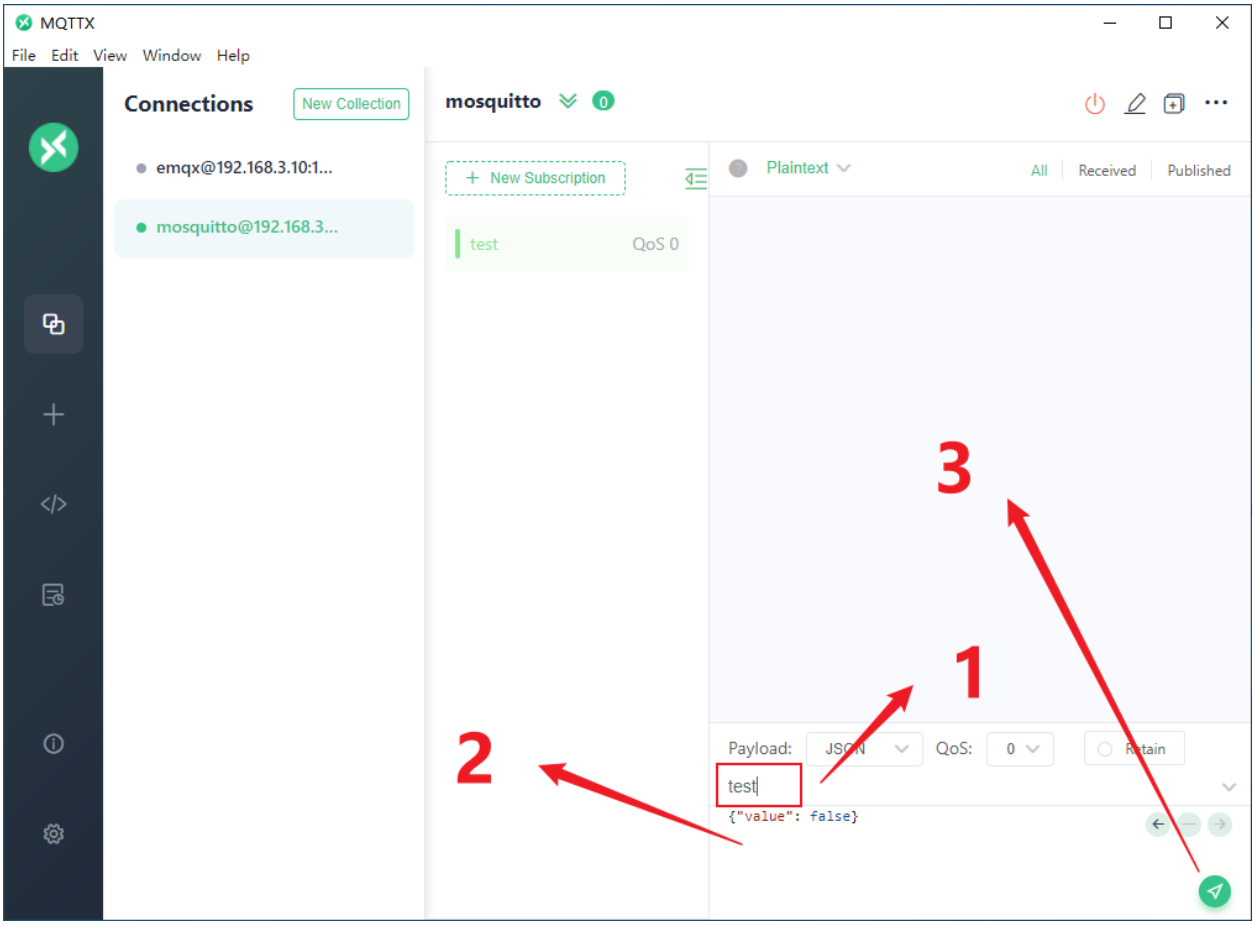
Modify the /etc/mosquitto/mosquitto.conf file, the content is as follows

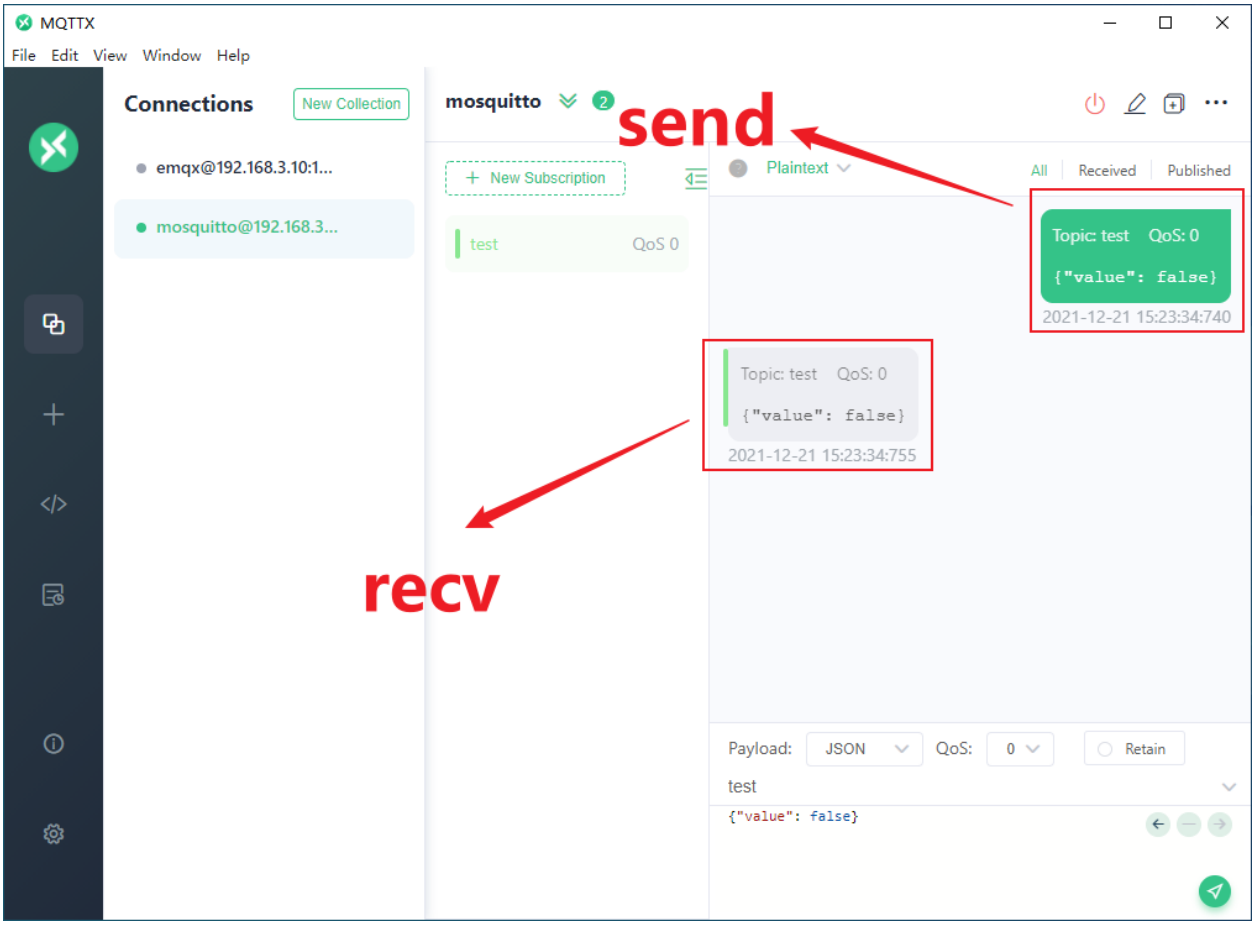
```
# Place your local configuration in /etc/mosquitto/conf.d/
#
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example

pid_file /run/mosquitto/mosquitto.pid

persistence true
persistence_location /var/lib/mosquitto/

log_dest file /var/log/mosquitto/mosquitto.log

include_dir /etc/mosquitto/conf.d

allow_anonymous true

port 1883
```

2.2.3 Test




MQTTX
File Edit View Window Help
Connections
New Collection
emqx@192.168.3.10:1...
mosquitto@192.168.3...
mosquitto
New Subscription
1
Plaintext
All
Received
Published
New Subscription
* Topic
2
test
* QoS
Color
0
#88E790
Alias
3
Cancel
Confirm
Payload:
JSON
QoS:
0
Retain
zigbee2mqtt/bridge/request/permit_join
{"value": false}


MQTTX
File Edit View Window Help
Connections
New Collection
emqx@192.168.3.10:1...
mosquitto@192.168.3...
mosquitto
0
New Subscription
test
QoS 0
Plaintext
All
Received
Published
3
1
2
Payload:
JSON
QoS:
0
Retain
test
{"value": false}



2.3 Home Assistant

2.3.1 Install docker

```
curl -fsSL https://get.docker.com | bash -s docker --mirror Aliyun
```

```
$ pwd
/home/pi
$ mkdir homeassistant
$ sudo docker run -d \
--name homeassistant \
--privileged \
--restart=unless-stopped \
-v /home/pi/homeassistant:/config \
--network=host \
ghcr.io/home-assistant/home-assistant:stable
```

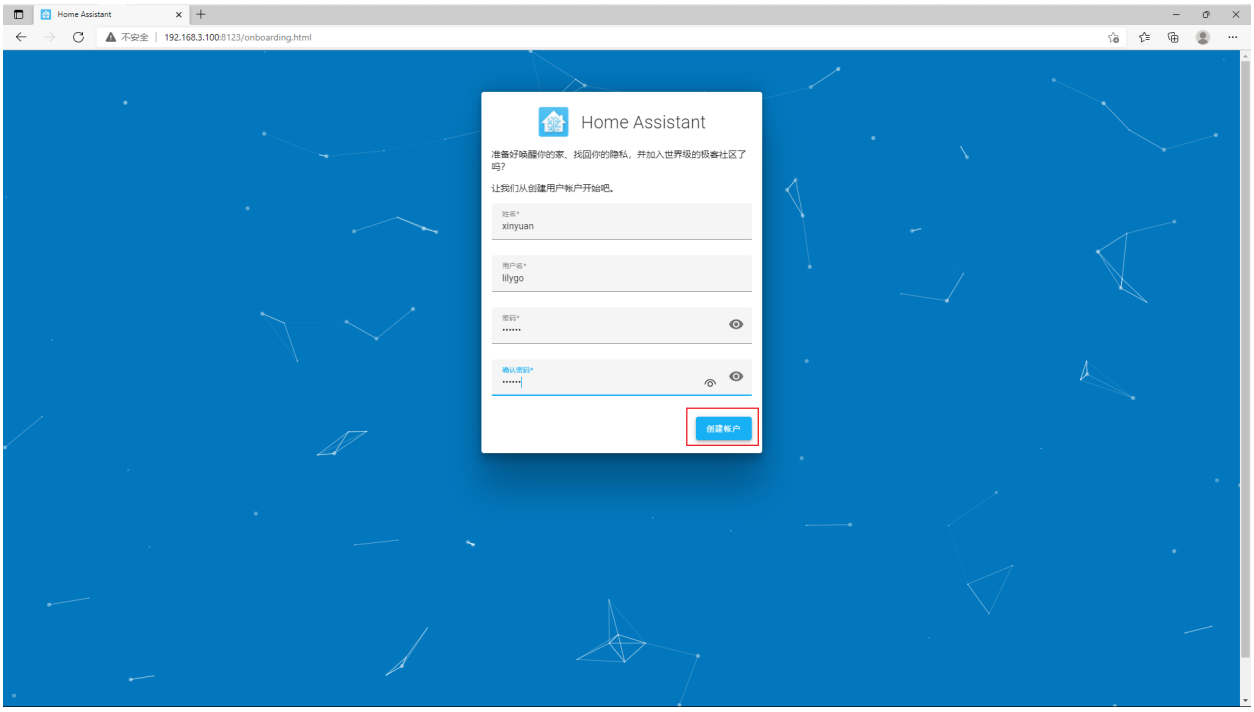
For more information, please refer to https://www.home-assistant.io/installation/raspberrypi

2.3.2 Confirm

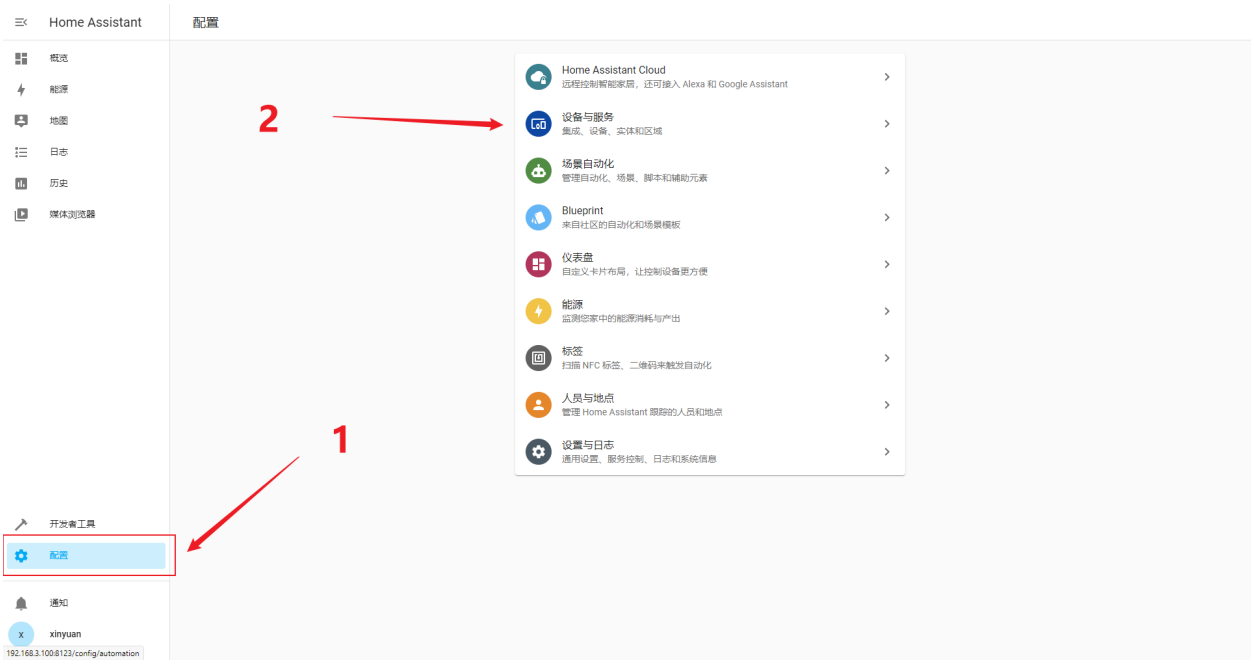
```
$ sudo docker ps -a
CONTAINER ID   IMAGE                                          COMMAND   CREATED       ␣
↪STATUS         PORTS     NAMES
f43aa176c789   ghcr.io/home-assistant/home-assistant:stable   "/init"   15 hours ago  ␣
↪Up 15 hours              homeassistant
```

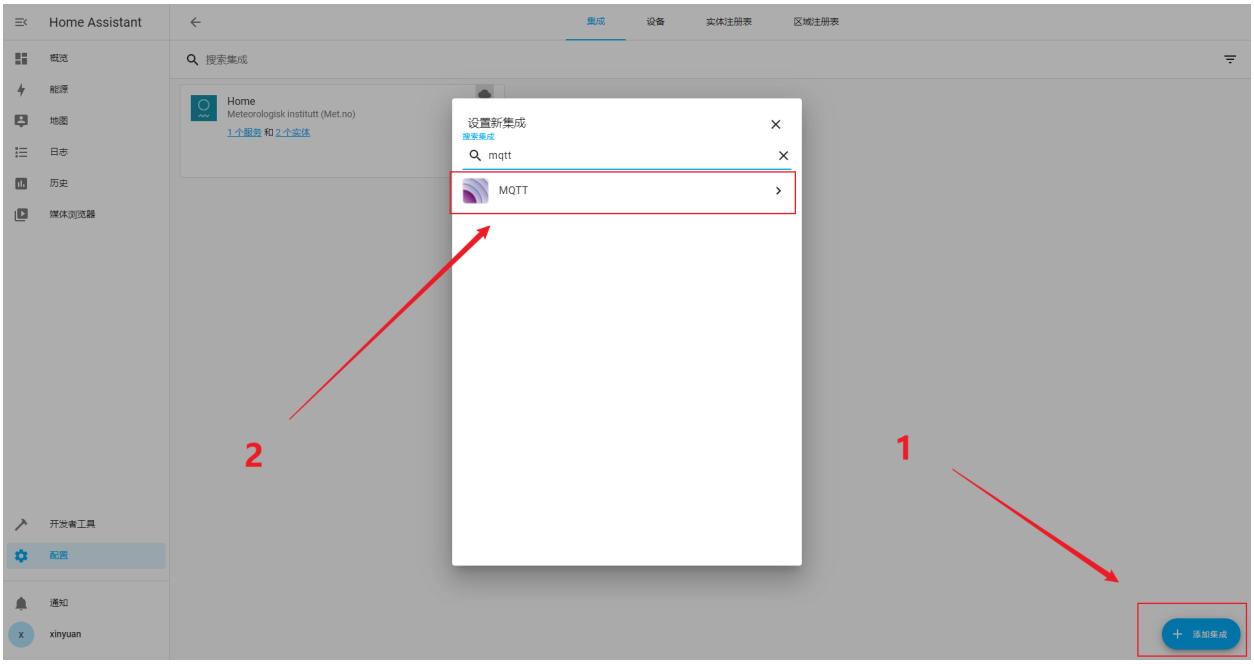
2.3.3 Configuration

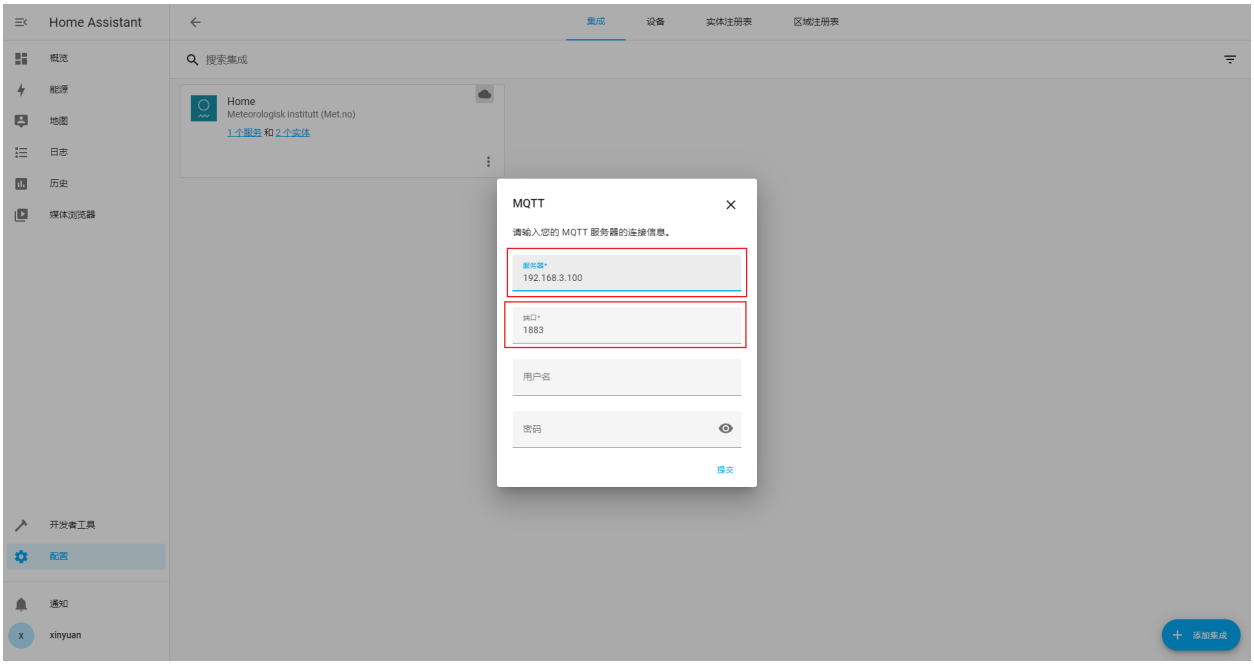
Once the Home Assistant Container is running Home Assistant should be accessible using `http://<host>:8123` (replace with the hostname or IP of the system)

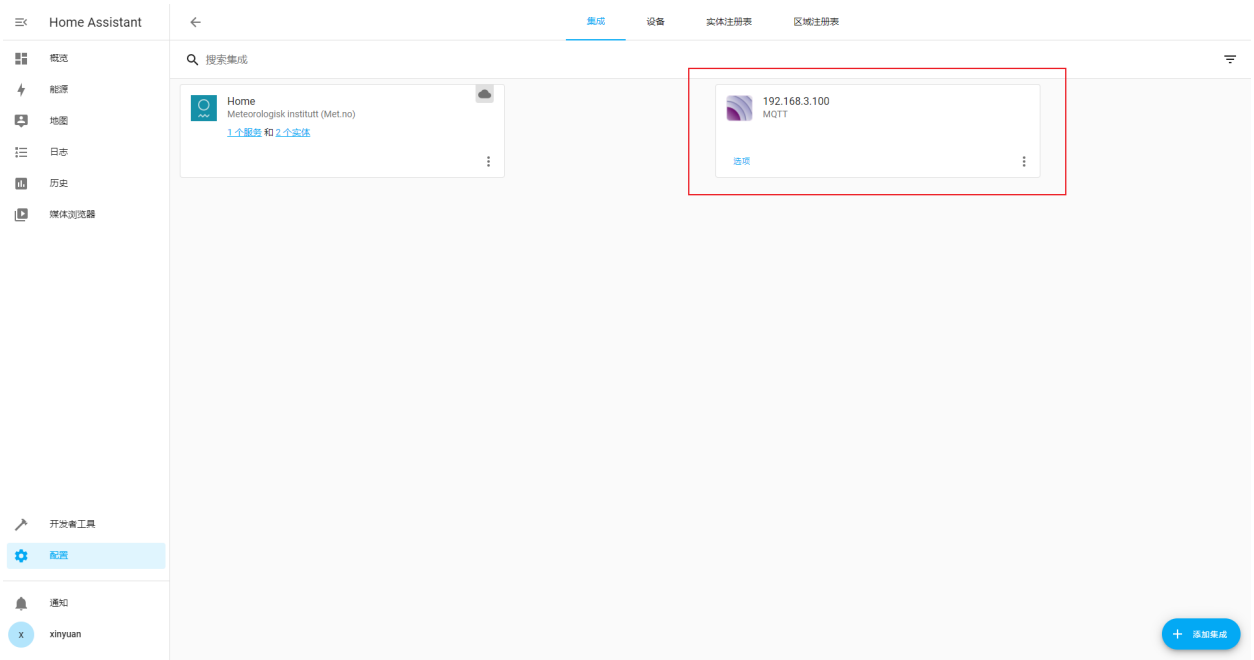


2.3.4 Integrated MQTT









2.4 Burning Guide

2.4.1 Introduction

`T-ZigBee` has two onboard SOCs, `ESP32-C3` of `espressif` and `TLSR8258` of `telink`.

Among them, `ESP32-C3` integrates WiFI and BLE, and `TLSR8258` integrates ZigBee and BLE. `TLSR8258` uses its ZigBee functionality only.

To program firmware for ESP32-C3 and TLSR8258, you need to use `T-U2T`. Through the DIP switch, choose to burn

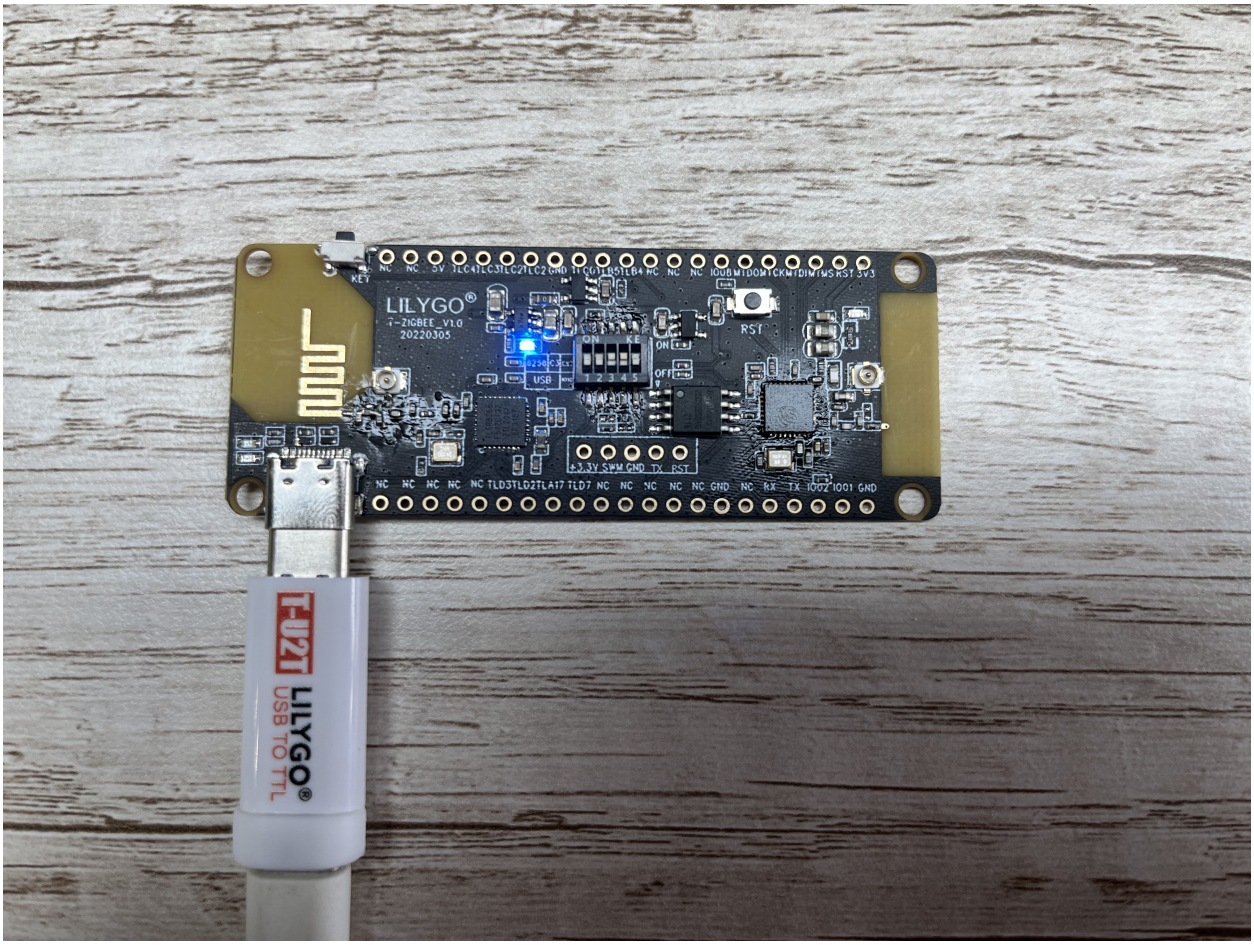
the firmware for different SOCs.

Although ESP32-C3 supports using USB to download firmware, in some development environments, you cannot directly use USB to view debugging information, which is inconvenient.Therefore, we use UART0 of ESP32-C3 to download and debug the firmware.

Note: Because the power supply of TLSR8258 is controlled by GPIO of `ESP32-C3`, before programming `TLSR8258`, please program examples/factory_test This program to power on `TLSR8258`.

2.4.2 ESP32-C3

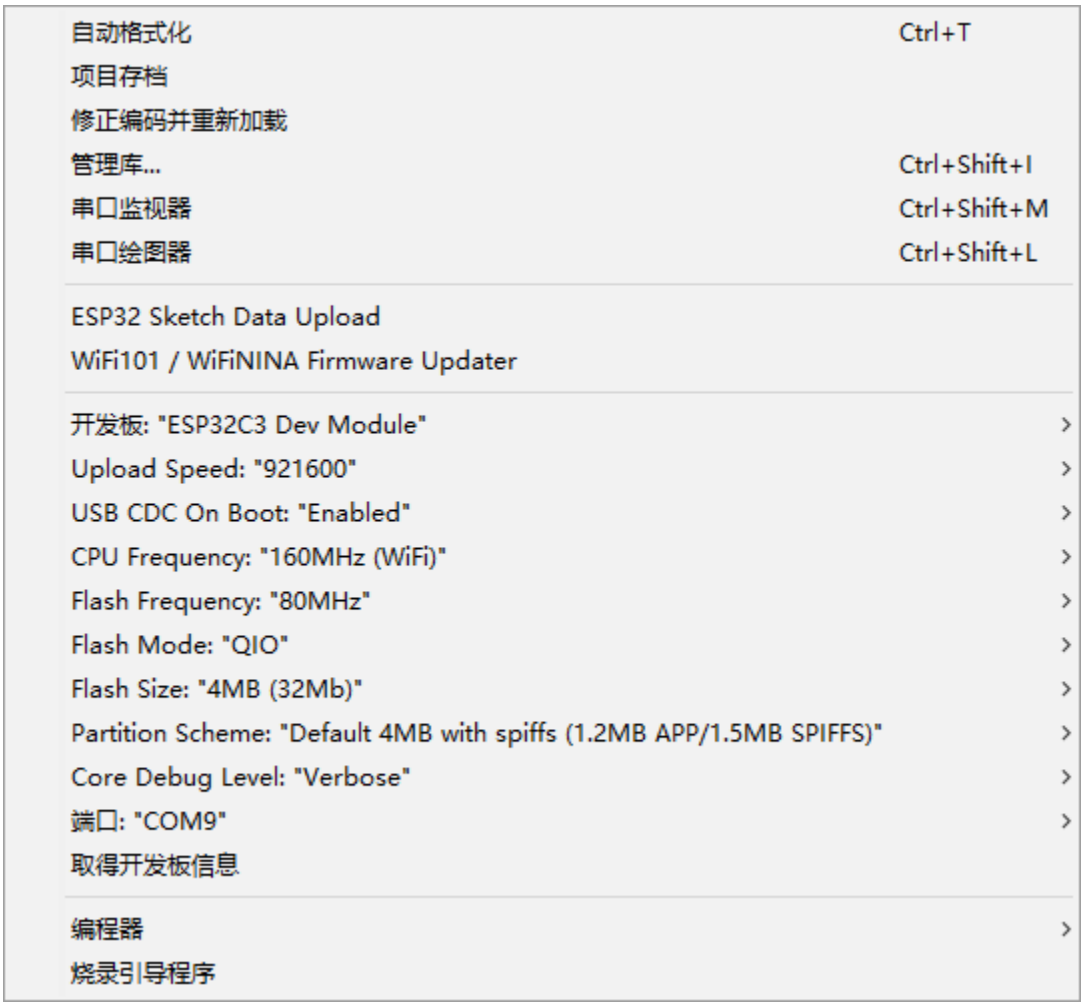
1. connect `T-U2T` connect to `T-ZigBee`



2. Set the DIP switch


ESP32-C3模式
ON
1 2 3 4 5

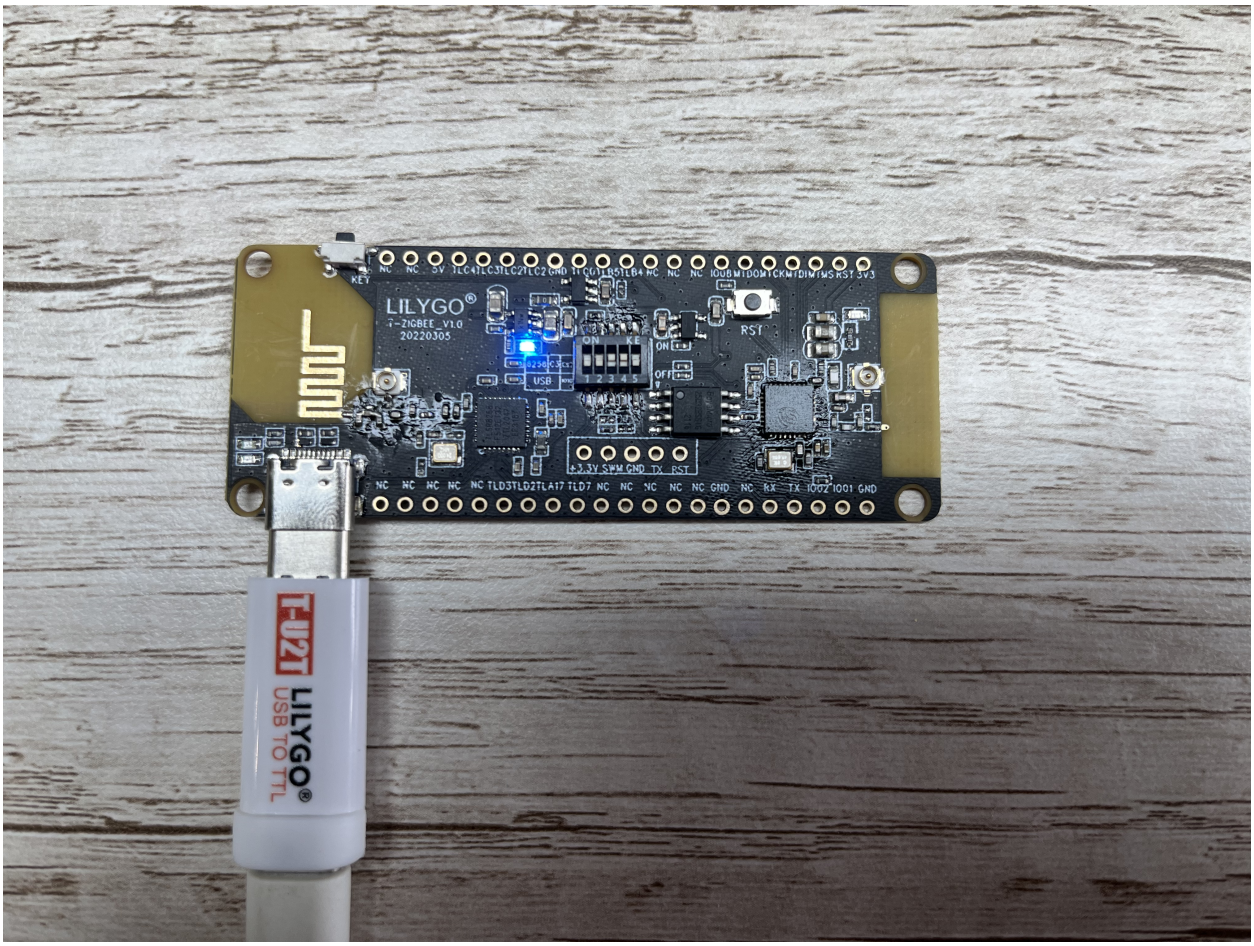
3. Burning


自动格式化 Ctrl+T
项目存档
修正编码并重新加载
管理库... Ctrl+Shift+I
串口监视器 Ctrl+Shift+M
串口绘图器 Ctrl+Shift+L
ESP32 Sketch Data Upload
WiFi101 / WiFiNINA Firmware Updater
开发板: "ESP32C3 Dev Module"
Upload Speed: "921600"
USB CDC On Boot: "Enabled"
CPU Frequency: "160MHz (WiFi)"
Flash Frequency: "80MHz"
Flash Mode: "QIO"
Flash Size: "4MB (32Mb)"
Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)"
Core Debug Level: "Verbose"
端口: "COM9"
取得开发板信息
编程器
烧录引导程序

2.4.3 TLSR8258

Note: `T-ZigBee` has been programmed with sampleGW_8258_20220302.bin, if not correct :code:` TLSR8258` makes other functional changes, it is not recommended to burn the firmware of `TLSR8258` at will.

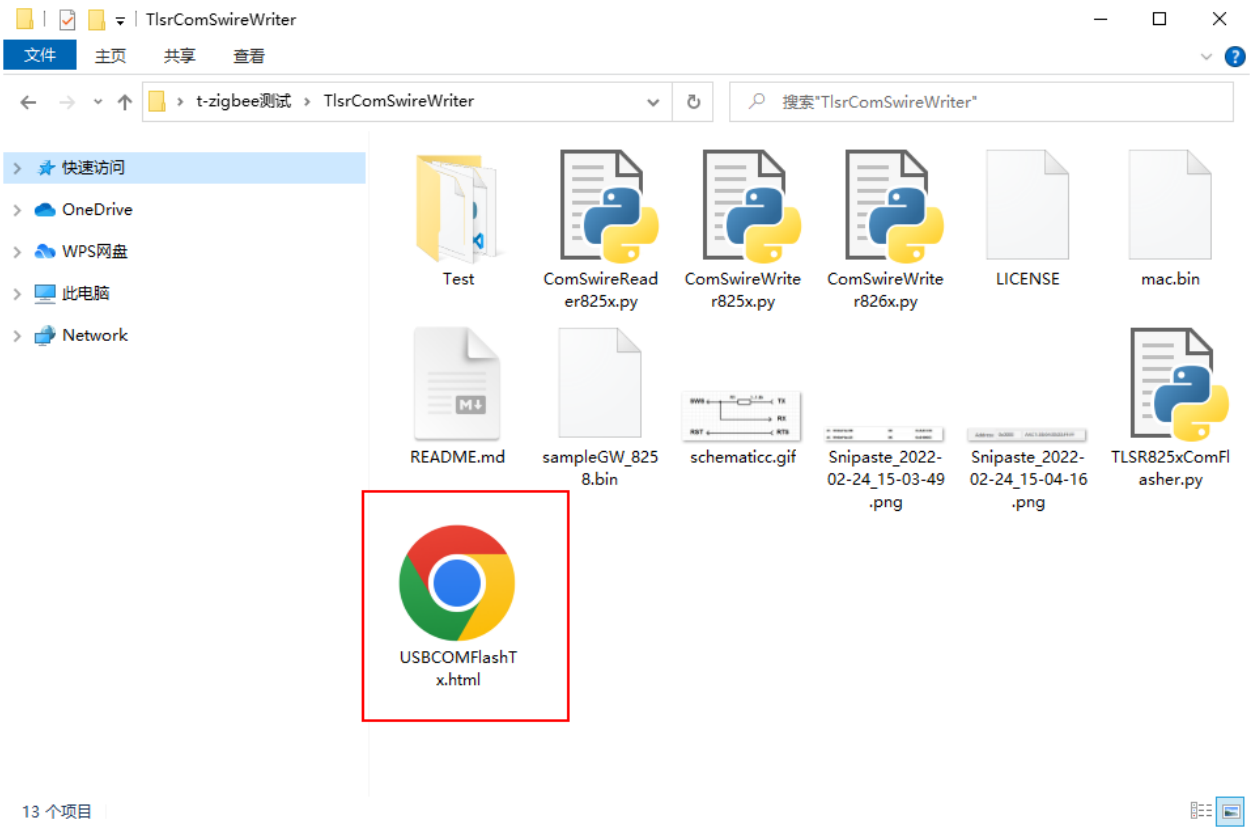
1. connect `T-U2T` connect to `T-ZigBee`

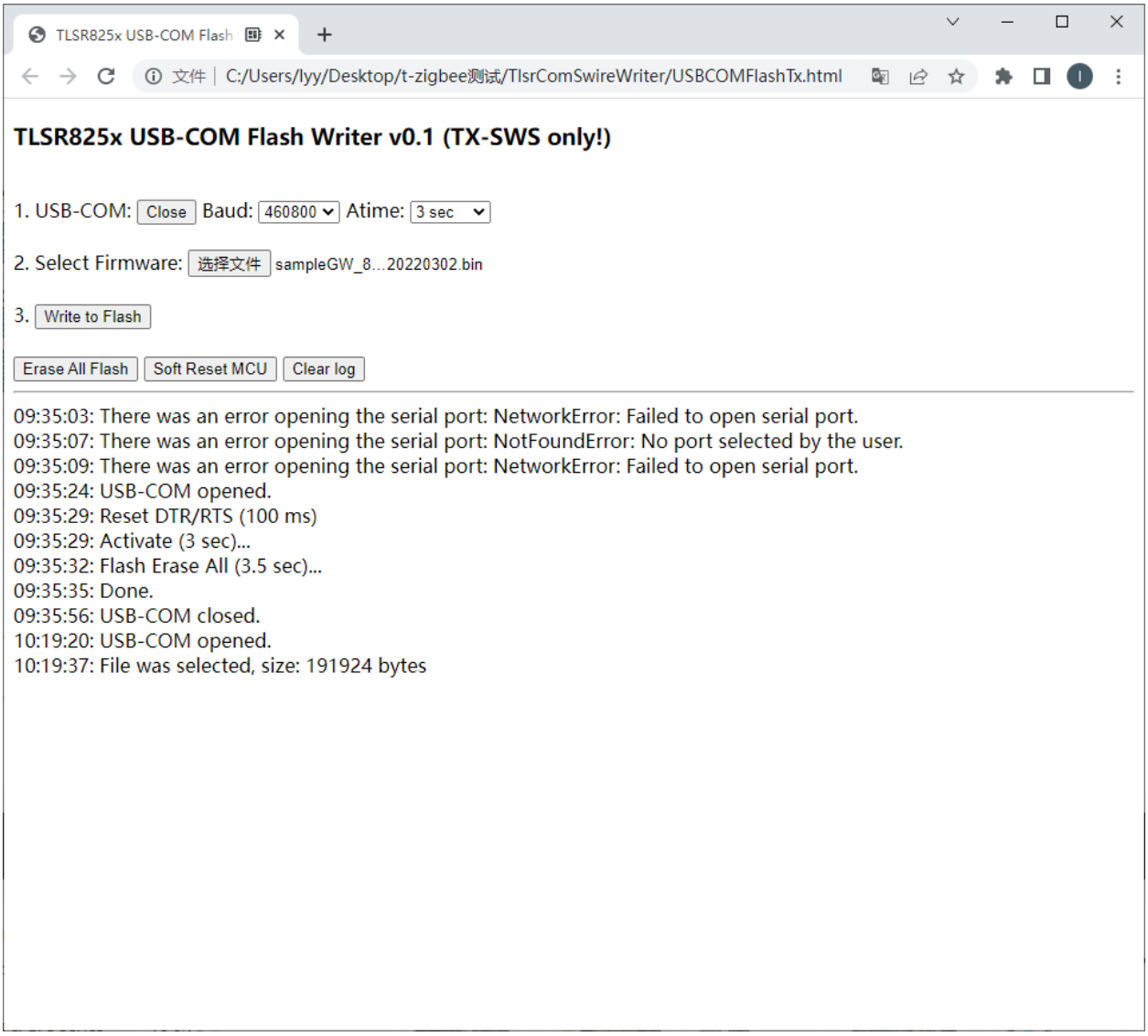


2. Set the DIP switch

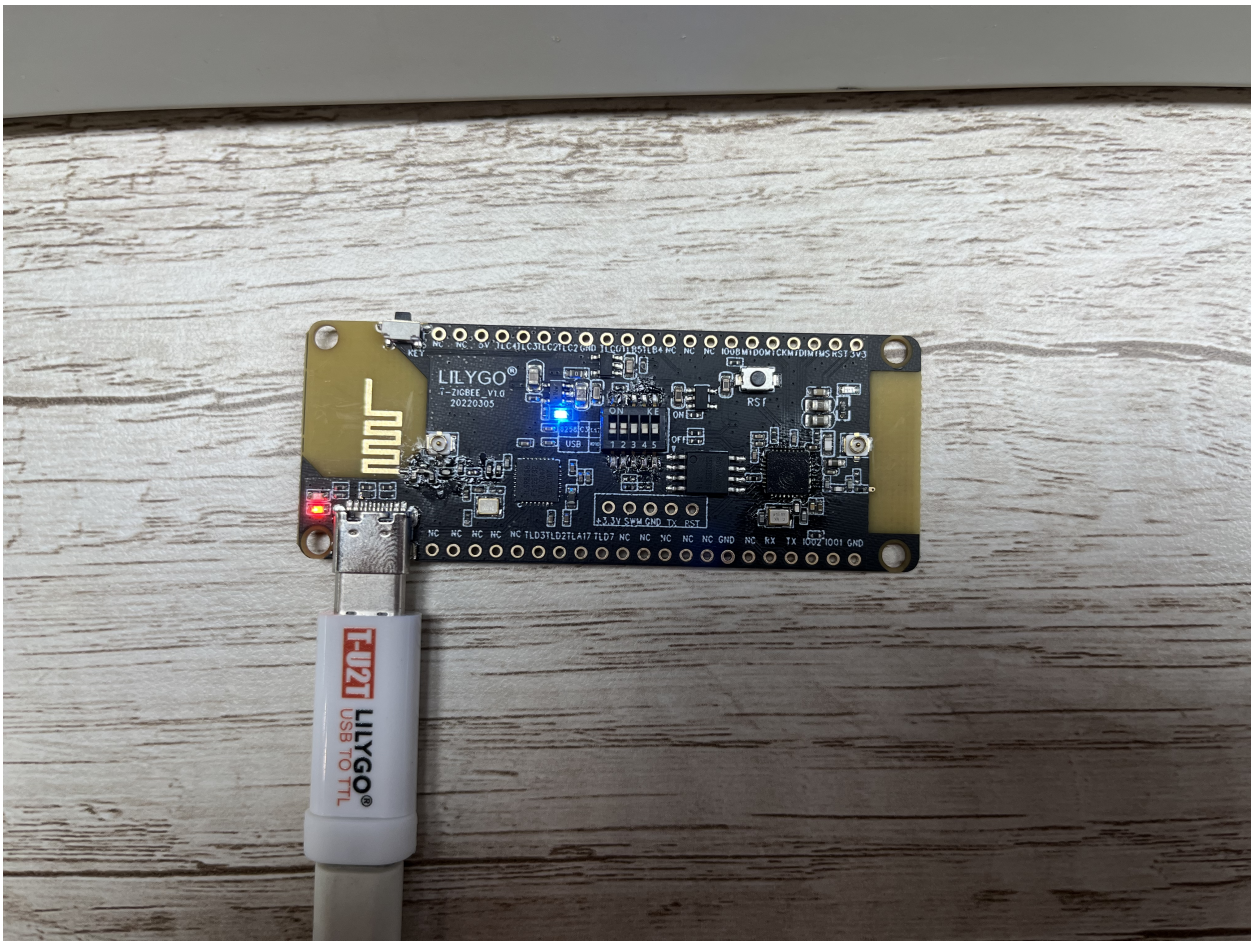
3. Use :code:TlsrComSwireWriter` to burn firmware

Please download TlsrComSwireWriter in advance.


TlsrComSwireWriter
文件
主页
共享
查看
t-zigbee测试 › TlsrComSwireWriter
搜索"TlsrComSwireWriter"
快速访问
OneDrive
WPS网盘
此电脑
Network
Test
ComSwireReader825x.py
ComSwireWriter825x.py
ComSwireWriter826x.py
LICENSE
mac.bin
README.md
sampleGW_8258.bin
schematicc.gif
Snipaste_2022-02-24_15-03-49.png
Snipaste_2022-02-24_15-04-16.png
TLSR825xComFlasher.py
USBCOMFlashTx.html
13 个项目



4. The burning is successful, the red indicator on the board will be on



2.4.4 Ordering Information

Product	Order channel		
T-ZigBee	t-zigbee_AliExpress	t-zigbee_TaoBao	t-zigbee_Amazon
T-U2T	t-u2t_AliExpress	t-u2t_TaoBao	t-u2t_Amazon

CHAPTER

THREE

DEVELOPER GUIDE

Functions

void **zbhci_Init**(QueueHandle_t queue)

void **zbhci_Deinit**(void)

void **zbhci_BdbCommissionFormation**(void)

start network formation.

void **zbhci_BdbCommissionSteer**(void)

classic join start.

Note: Only Router and EndDevice are valid

---

void **zbhci_BdbCommissionTouchlink**(*te_MsgBdbCommissionTouchlinkRole* eRole)

touch link network formation or touch link join start.

See also:

te_MsgBdbCommissionTouchlinkRole

Parameters

- **eRole** – **[in]** initiator or target

void **zbhci_BdbCommissionFindbind**(*te_MsgBdbCommissionFindbindRole* eRole)

find and bind touch link node.

See also:

te_MsgBdbCommissionTouchlinkRole

Parameters

- **eRole** – **[in]** initiator or target

void **zbhci_BdbFactoryReset**(void)

Reset device to factory new.

Note:

This interface could be called by Coordinator, Router or End-Device.

If coordinator:

It will erase all NV information and reset all layer settings.

if router or end device:

If it is a not factory new device, it will broadcast a Leave Command before factory new reset.

---

void **zbhci_BdbPreInstallCode**(uint64_t u64DevAddr, uint8_t *pu8UniqueLinkKey)

Add pre-install code to NV.

Parameters

- **u64DevAddr** – **[in]** the ieee address of the device using unique link key join
- **pu8UniqueLinkKey** – **[in]** the pointer of install code

void **zbhci_BdbChannelSet**(uint8_t u8ChannelIdx)

Set channel mask to APS_IB.

Parameters

- **u8ChannelIdx** – **[in]** channel: 11 - 26

void **zbhci_BdbDongleWorkingModeSet**(*te_MsgBdbDongleWorkingMode* eMode)

Set the working mode of the coordinator.

See also:

te_MsgBdbDongleWorkingMode

Parameters

- **eMode** – **[in]** work mode

void **zbhci_BdbNodeDelete**(uint64_t u64DevAddr)

Interface for forgetting all information about the specified device.

Parameters

- **u64DevAddr** – **[in]** ieee address of the specified device

void **zbhci_BdbTxPowerSet**(*te_BdbTxPowerLevel* eLevel)

Set specified transmitted power.

See also:

te_BdbTxPowerLevel.

Parameters

- **eLevel** – **[in]** Specified power

void **zbhci_NetworkStateReq**(void)

Get the network information of the coordinator node.

void **zbhci_DiscoveryNwkAddrReq**(uint16_t u16DstAddr, uint64_t u64IEEEAddr, uint8_t u8ReqType, uint8_t u8StartIdx)

Send address request to target device for short address.

Parameters

- **u16DstAddr** – **[in]** The destination address to which this command will be sent, shall be unicast or broadcast to all devices for which macRxOnWhenIdle = TRUE.
- **u64IEEEAddr** – **[in]** The IEEE address to be matched by the remote device.
- **u8ReqType** – **[in]** Request type for this command:
  - 0x00 – Single device response;
  - 0x01 – Extended response;
  - 0x02 ~ 0xFF – Reserved.
- **u8StartIdx** – **[in]** If the request type for this command is extended response, the startIdx provides the starting index for the requested elements of the associated device list.

void **zbhci_DiscoveryIeeeAddrReq**(uint16_t u16DstAddr, uint16_t u16NwkAddrOfInterest, uint8_t u8ReqType, uint8_t u8StartIdx)

Send address request to target device for IEEE address.

Parameters

- **u16DstAddr** – **[in]** The destination address to which this command will be sent, shall be unicast or broadcast to all devices for which macRxOnWhenIdle = TRUE.
- **u16NwkAddrOfInterest** – **[in]** The NWK address that is used for IEEE address mapping.
- **u8ReqType** – **[in]** Request type for this command:
  - 0x00 – Single device response;
  - 0x01 – Extended response;
  - 0x02 ~ 0xFF – Reserved.
- **u8StartIdx** – **[in]** If the request type for this command is extended response, the startIdx provides the starting index for the requested elements of the associated device list.

void **zbhci_DiscoveryNodeDescReq**(uint16_t u16DstAddr, uint16_t u16NwkAddrOfInterest)

Send node descriptor request.

Parameters

- **u16DstAddr** – **[in]** The destination address to which this command will be sent, shall be unicast or broadcast to all devices for which macRxOnWhenIdle = TRUE.
- **u16NwkAddrOfInterest** – **[in]** NWK address of the target.

void **zbhci_DiscoverySimpleDescReq**(uint16_t u16DstAddr, uint16_t u16NwkAddrOfInterest, uint8_t u8Endpoint)

Send simple descriptor request.

Parameters

- **u16DstAddr** – **[in]** The destination address to which this command will be sent, shall be unicast or broadcast to all devices for which macRxOnWhenIdle = TRUE.

- **u16NwkAddrOfInterest** – **[in]** NWK address of the target.
- **u8Endpoint** – **[in]** The endpoint on the destination.

void **zbhci_DiscoveryMatchDescReq**(uint16_t u16DstAddr, uint16_t u16NwkAddrOfInterest, uint16_t u16ProfileID, uint8_t u8NumInClusters, uint8_t u8NumOutClusters, uint16_t *pu16ClusterList)

Send match descriptor request.

Parameters

- **u16DstAddr** – **[in]** The destination address to which this command will be sent, shall be unicast or broadcast to all devices for which macRxOnWhenIdle = TRUE.
- **u16NwkAddrOfInterest** – **[in]** NWK address of the target.
- **u16ProfileID** – **[in]** Profile ID to be matched at the destination.
- **u8NumInClusters** – **[in]** Profile ID to be matched at the destination.
- **u8NumOutClusters** – **[in]** Profile ID to be matched at the destination.
- **pu16ClusterList** – **[in]** inClusterList + outClusterList.
  - List of input cluster IDs to be used for matching, the inClusterList is the desired list to be matched by the Remote Device (the elements of the inClusterList are the supported output clusters of the Local Device).
  - List of output cluster IDs to be used for matching, the outClusterList is the desired list to be matched by the Remote Device (the elements of the outClusterList are the supported input clusters of the Local Device)

void **zbhci_DiscoveryActiveEpReq**(uint16_t u16DstAddr, uint16_t u16NwkAddrOfInterest)

Send active endpoint request.

Parameters

- **u16DstAddr** – **[in]** The destination address to which this command will be sent, shall be unicast or broadcast to all devices for which macRxOnWhenIdle = TRUE.
- **u16NwkAddrOfInterest** – **[in]** NWK address of the target.

void **zbhci_DiscoveryLeaveReq**(uint64_t u64DevAddr, uint8_t u8Rejoin, uint8_t u8RemoveChildren)

Send the command of mgmt_leave_req.

Parameters

- **u64DevAddr** – **[in]** The IEEE address of the device to be removed or NULL if the device removes itself.
- **u8Rejoin** – **[in]** TRUE if the device is being asked to leave from the current parent and requested to rejoin the network. Otherwise, the parameter has a value of FALSE.
- **u8RemoveChildren** – **[in]** TRUE if the device being asked to leave the network is also being asked to remove its child device, if any. Otherwise, it has a value of FALSE.

void **zbhci_BindingReq**(uint64_t u64SrcIEEEAddr, uint8_t u8SrcEndpoint, uint16_t u16ClusterID, uint8_t u8DstAddrMode, *ts_DstAddr* sDstAddr, uint8_t u8DstEndpoint)

Send bind request, destination address is based on the srcAddr in the request command.

Parameters

- **u64SrcIEEEAddr** – **[in]** The IEEE address for the source.
- **u8SrcEndpoint** – **[in]** The source endpoint for the binding entry.

- **u16ClusterID** – **[in]** The identifier of the cluster on the source device that is bound to the destination.
- **u8DstAddrMode** – **[in]** The addressing mode for the destination address used in this command.
  - 0x00 – Reserved.
  - 0x01 – with group address;
  - 0x02 – with destination short address and endpoint;
  - 0x03 – with destination IEEE address and endpoint.
  - 0x04 ~ 0xFF – Reserved.
- **sDstAddr** – **[in]** The destination address for the binding entry.
- **u8DstEndpoint** – **[in]** Shall be present only if the dstAddrMode field has a value of 0x03 and, if present, shall be the destination endpoint for the binding entry.

void **zbhci_UnbindingReq**(uint64_t u64SrcIEEEAddr, uint8_t u8SrcEndpoint, uint16_t u16ClusterID, uint8_t u8DstAddrMode, *ts_DstAddr* sDstAddr, uint8_t u8DstEndpoint)

Send unbind request, destination address is based on the srcAddr in the request command.

Parameters

- **u64SrcIEEEAddr** – **[in]** The IEEE address for the source.
- **u8SrcEndpoint** – **[in]** The source endpoint for the binding entry.
- **u16ClusterID** – **[in]** The identifier of the cluster on the source device that is bound to the destination.
- **u8DstAddrMode** – **[in]** The addressing mode for the destination address used in this command.
  - 0x00 – Reserved.
  - 0x01 – with group address;
  - 0x02 – with destination short address and endpoint;
  - 0x03 – with destination IEEE address and endpoint.
  - 0x04 ~ 0xFF – Reserved.
- **sDstAddr** – **[in]** The destination address for the binding entry.
- **u8DstEndpoint** – **[in]** Shall be present only if the dstAddrMode field has a value of 0x03 and, if present, shall be the destination endpoint for the binding entry.

void **zbhci_MgmtLqiReq**(uint16_t u16DstAddr, uint8_t u8StartIdx)

Send the command of mgmt_lqi_req.

Parameters

- **u16DstAddr** – **[in]** The destination address to which this command will be sent, shall be unicast.
- **u8StartIdx** – **[in]** Starting index for the requested elements of the neighbor table.

void **zbhci_MgmtBindReq**(uint16_t u16DstAddr, uint8_t u8StartIdx)

Send the command mgmt_bind_req.

Parameters

- **u16DstAddr** – **[in]** The destination address to which this command will be sent, shall be unicast.
- **u8StartIdx** – **[in]** Starting index for the requested elements of the binding table.

void **zbhci_MgmtLeaveReq**(uint16_t u16DstAddr, uint64_t u64DevAddr, uint8_t u8Rejoin, uint8_t u8RemoveChildren)

Send the command of mgmt_leave_req.

Parameters

- **u16DstAddr** – **[in]** The destination address to which this command will be sent, shall be unicast.
- **u64DevAddr** – **[in]** The IEEE address of the device to be removed or NULL if the device removes itself.
- **u8Rejoin** – **[in]** TRUE if the device is being asked to leave from the current parent and requested to rejoin the network. Otherwise, the parameter has a value of FALSE.
- **u8RemoveChildren** – **[in]** TRUE if the device being asked to leave the network is also being asked to remove its child device, if any. Otherwise, it has a value of FALSE.

void **zbhci_MgmtDirectJoinReq**(void)

None.

void **zbhci_MgmtPermitJoinReq**(uint16_t u16DstAddr, uint8_t u8PermitDuration, uint8_t u8TCSignificance)

Send the command of mgmt_permitJoin_req.

Parameters

- **u16DstAddr** – **[in]** The destination address to which this command will be sent.
- **u8PermitDuration** – **[in]** The length of time in seconds during which the coordinator or router will allow associations. The value 0x00 and 0xff indicate that permission is disabled or enabled, respectively, without a specified limit.
- **u8TCSignificance** – **[in]** This field shall always have a value of 1, indicating a request to change the Trust Center policy. If a frame is received with a value of 0, it shall be treated as having a value of 1.

void **zbhci_MgmtNwkUpdateReq**(uint16_t u16DstAddr, uint16_t u16NwkManagerAddr, uint32_t u32ScanChannels, uint8_t u8ScanDuration, uint8_t u8ScanCount, uint8_t u8NwkUpdateId)

Send the command of mgmt_network_update_req.

Parameters

- **u16DstAddr** – **[in]** The destination address to which this command will be sent.
- **u16NwkManagerAddr** – **[in]** This field shall be present only if the scanDuration is set to 0xff, and, when present, it indicates the NWK address for the device with the Network Manager bit set in its Node Descriptor.
- **u32ScanChannels** – **[in]** Channel bit mask, 32-bit field structure.
- **u8ScanDuration** – **[in]** A value used to calculate the length of time to spend on scanning each channel. 0x00 ~ 0x05 or 0xfe or 0xff.
- **u8ScanCount** – **[in]** This field represents the number of energy scans to be conducted and reported. This field shall be present only if the scanDuration is within the range of 0x00 to 0x05.

- **u8NwkUpdateId** – **[in]** The value of the nwkUpdateId contained in this request. This value is set by the Network Channel Manager prior to sending the message. This field shall only be present if the scanDuration is 0xfe or 0xff. If the scanDuration is 0xff, then the value in the nwkUpdateId shall be ignored.

void **zbhci_NodesJoinedGetReq**(uint16_t u16StartIdx)

get the MAC address form key pair table so as to get the information of the node which have joined the network

Parameters

- **u16StartIdx** – **[in]** Starting index for the requested elements of the joined node list.

void **zbhci_NodesTogleTestReq**(uint8_t u8OnOff, uint8_t u8TimerInterval)

send ON or OFF command in ONOFF cluster

Parameters

- **u8OnOff** – **[in]** On/off command.
- **u8TimerInterval** – **[in]** The timer interval of the data transmission. Unit: millisecond.

void **zbhci_TxRxPerformanceTestReq**(uint16_t u16DstAddr, uint8_t u8SrcEp, uint8_t u8DstEp, uint16_t u16SendCnt, uint8_t u8Interval, uint8_t u8TxPowerSet, uint8_t *pu8Payload)

send Tx Rx Performance test Req.

Parameters

- **u16DstAddr** – The destination address to which this command will be sent.
- **u8SrcEp** – The source endpoint for the binding entry.
- **u8DstEp** – Destination endpoint
- **u16SendCnt** – Unknown
- **u8Interval** – Unknown
- **u8TxPowerSet** – Unused
- **pu8Payload** – Unused

void **zbhci_AfDataSendTestReq**(uint16_t u16DstAddr, uint8_t u8SrcEp, uint8_t u8DstEp, uint16_t u16ClusterId, uint16_t u16DataLen, uint8_t *pu8Payload)

Send AF data.

Parameters

- **u16DstAddr** – The destination address to which this command will be sent.
- **u8SrcEp** – The source endpoint for the binding entry.
- **u8DstEp** – Destination endpoint
- **u16ClusterId** – cluster identifier
- **u16DataLen** – data length
- **pu8Payload** – data payload

void **zbhci_ZclAttrRead**(uint8_t u8DstAddrMode, *ts_DstAddr* sDstAddr, uint8_t u8SrcEp, uint8_t u8DstEp, uint8_t u8Direction, uint16_t u16ClusterID, uint8_t u8AttrNum, uint16_t *pu16AttrList)

Send a Read Attribute command.

Parameters

- **u8DstAddrMode** – **[in]** Destination address mode:
  - 0x00 – Reserved.
  - 0x01 – with group address;
  - 0x02 – with destination short address and endpoint;
  - 0x03 – with destination IEEE address and endpoint.
  - 0x04 ~ 0xFF – Reserved.
- **sDstAddr** – **[in]** Destination address for this command.
- **u8SrcEp** – **[in]** Source endpoint.
- **u8DstEp** – **[in]** Destination endpoint if dstAddrMode is 2 or 3.
- **u8Direction** – **[in]** direction of the command:
  - 0 – Client to server;
  - 1 – Server to client.
- **u16ClusterID** – **[in]** Cluster identifier.
- **u8AttrNum** – **[in]** The number of attributes to be read.
- **pu16AttrList** – **[in]** The list of the attribute IDs to be read.

void **zbhci_ZclAttrWrite**(uint8_t u8DstAddrMode, *ts_DstAddr* sDstAddr, uint8_t u8SrcEp, uint8_t u8DstEp, uint8_t u8Direction, uint16_t u16ClusterID, uint8_t u8AttrNum, *ts_AttrList* *psAttrList)

Send a Write Attribute command.

Parameters

- **u8DstAddrMode** – **[in]** Destination address mode:
  - 0x00 – Reserved.
  - 0x01 – with group address;
  - 0x02 – with destination short address and endpoint;
  - 0x03 – with destination IEEE address and endpoint.
  - 0x04 ~ 0xFF – Reserved.
- **sDstAddr** – **[in]** Destination address for this command.
- **u8SrcEp** – **[in]** Source endpoint.
- **u8DstEp** – **[in]** Destination endpoint if dstAddrMode is 2 or 3.
- **u8Direction** – **[in]** direction of the command:
  - 0 – Client to server;
  - 1 – Server to client.
- **u16ClusterID** – **[in]** Cluster identifier.
- **u8AttrNum** – **[in]** The number of attributes to be written.
- **pu16AttrList** – **[in]** The list of the attributes to be written.

void **zbhci_ZclConfigReport**(uint8_t u8DstAddrMode, *ts_DstAddr* sDstAddr, uint8_t u8SrcEp, uint8_t u8DstEp, uint8_t u8Direction, uint16_t u16ClusterID, uint8_t u8AttrNum, *ts_AttrList* *psAttrList)

API to send ZCL configure report command.

Parameters

- **u8DstAddrMode** – **[in]** Destination address mode:
  - 0x00 – Reserved.
  - 0x01 – with group address;
  - 0x02 – with destination short address and endpoint;
  - 0x03 – with destination IEEE address and endpoint.
  - 0x04 ~ 0xFF – Reserved.
- **sDstAddr** – **[in]** Destination address for this command.
- **u8SrcEp** – **[in]** Source endpoint.
- **u8DstEp** – **[in]** Destination endpoint if dstAddrMode is 2 or 3.
- **u8Direction** – **[in]** specified the command direction:
  - 0 – Client to server;
  - 1 – Server to client.
- **u16ClusterID** – **[in]** Cluster identifier.
- **u8AttrNum** – **[in]** The number of attributes to be configured.
- **pu16AttrList** – **[in]** The list of the attributes to be configured.

void **zbhci_ZclReadReportCfg**(uint8_t u8DstAddrMode, *ts_DstAddr* sDstAddr, uint8_t u8SrcEp, uint8_t u8DstEp, uint8_t u8Direction, uint16_t u16ClusterID, uint8_t u8AttrNum, uint16_t *pu16AttrList)

This function can be used on a cluster client to send a 'read reporting configuration' command to a cluster server, in order to request the attribute reporting configuration data for a set of attributes.

Parameters

- **u8DstAddrMode** – **[in]** Destination address mode:
  - 0x00 – Reserved.
  - 0x01 – with group address;
  - 0x02 – with destination short address and endpoint;
  - 0x03 – with destination IEEE address and endpoint.
  - 0x04 ~ 0xFF – Reserved.
- **sDstAddr** – **[in]** Destination address for this command.
- **u8SrcEp** – **[in]** Source endpoint.
- **u8DstEp** – **[in]** Destination endpoint if dstAddrMode is 2 or 3.
- **u8Direction** – **[in]** specified the command direction:
  - 0 – Client to server;
  - 1 – Server to client.

- **u16ClusterID** – **[in]** Cluster identifier.
- **u8AttrNum** – **[in]** The number of attributes' configuration to be read.
- **pu16AttrList** – **[in]** The list of the attributes to be read.

void **zbhci_ZclLocalAttrWrite**(uint8_t u8Endpoint, uint16_t u16ClusterId, uint16_t u16AttrId, uint8_t u8DataLen, uint8_t *pu8Data)

void **zbhci_ZclSendReportCmd**(uint8_t u8DstAddrMode, *ts_DstAddr* sDstAddr, uint8_t u8SrcEp, uint8_t u8DstEp, uint8_t u8DisableDefaultRsp, uint8_t u8Direction, uint16_t u16ClusterID, uint16_t u16AttrID, uint8_t u8DataType, uint8_t u8DataLen, uint8_t *pu8Data)

void **zbhci_ZclBasicReset**(uint8_t u8DstAddrMode, *ts_DstAddr* sDstAddr, uint8_t u8SrcEp, uint8_t u8DstEp)

API to send Basic Reset to factory default command.

Parameters

- **u8DstAddrMode** – **[in]** Destination address mode:
  - 0x00 – Reserved.
  - 0x01 – with group address;
  - 0x02 – with destination short address and endpoint;
  - 0x03 – with destination IEEE address and endpoint.
  - 0x04 ~ 0xFF – Reserved.
- **sDstAddr** – **[in]** Destination address for this command.
- **u8SrcEp** – **[in]** Source endpoint.
- **u8DstEp** – **[in]** Destination endpoint if dstAddrMode is 2 or 3.

void **zbhci_ZclGroupAdd**(uint8_t u8DstAddrMode, *ts_DstAddr* sDstAddr, uint8_t u8SrcEp, uint8_t u8DstEp, uint16_t u16GroupId, uint8_t *pu8GroupName)

API to send Add command in Group cluster.

Parameters

- **u8DstAddrMode** – **[in]** Destination address mode:
  - 0x00 – Reserved.
  - 0x01 – with group address;
  - 0x02 – with destination short address and endpoint;
  - 0x03 – with destination IEEE address and endpoint.
  - 0x04 ~ 0xFF – Reserved.
- **sDstAddr** – **[in]** Destination address for this command.
- **u8SrcEp** – **[in]** Source endpoint.
- **u8DstEp** – **[in]** Destination endpoint if dstAddrMode is 2 or 3.
- **u16GroupId** – **[in]** Group identifier.
- **pu8GroupName** – **[in]** Group name, character string.

void **zbhci_ZclGroupView**(uint8_t u8DstAddrMode, *ts_DstAddr* sDstAddr, uint8_t u8SrcEp, uint8_t u8DstEp, uint16_t u16GroupId)

API to send View command in Group cluster.

Parameters

- **u8DstAddrMode** – **[in]** Destination address mode:
  - 0x00 – Reserved.
  - 0x01 – with group address;
  - 0x02 – with destination short address and endpoint;
  - 0x03 – with destination IEEE address and endpoint.
  - 0x04 ~ 0xFF – Reserved.
- **sDstAddr** – **[in]** Destination address for this command.
- **u8SrcEp** – **[in]** Source endpoint.
- **u8DstEp** – **[in]** Destination endpoint if dstAddrMode is 2 or 3.
- **u16GroupId** – **[in]** Group identifier.

void **zbhci_ZclGroupGetMembership**(uint8_t u8DstAddrMode, *ts_DstAddr* sDstAddr, uint8_t u8SrcEp, uint8_t u8DstEp, uint8_t u8GroupCount, uint16_t *pu16GroupList)

API to send Get Membership command in Group cluster.

Parameters

- **u8DstAddrMode** – **[in]** Destination address mode:
  - 0x00 – Reserved.
  - 0x01 – with group address;
  - 0x02 – with destination short address and endpoint;
  - 0x03 – with destination IEEE address and endpoint.
  - 0x04 ~ 0xFF – Reserved.
- **sDstAddr** – **[in]** Destination address for this command.
- **u8SrcEp** – **[in]** Source endpoint.
- **u8DstEp** – **[in]** Destination endpoint if dstAddrMode is 2 or 3.
- **u16GroupCount** – **[in]** Group count.
- **pu16GroupList** – **[in]** Group list.

void **zbhci_ZclGroupRemove**(uint8_t u8DstAddrMode, *ts_DstAddr* sDstAddr, uint8_t u8SrcEp, uint8_t u8DstEp, uint16_t u16GroupId)

API to send Remove command in Group cluster.

Parameters

- **u8DstAddrMode** – **[in]** Destination address mode:
  - 0x00 – Reserved.
  - 0x01 – with group address;
  - 0x02 – with destination short address and endpoint;

  - 0x03 – with destination IEEE address and endpoint.
  - 0x04 ~ 0xFF – Reserved.
- **sDstAddr** – **[in]** Destination address for this command.
- **u8SrcEp** – **[in]** Source endpoint.
- **u8DstEp** – **[in]** Destination endpoint if dstAddrMode is 2 or 3.
- **u16GroupId** – **[in]** Group identifier.

void **zbhci_ZclGroupRemoveAll**(uint8_t u8DstAddrMode, *ts_DstAddr* sDstAddr, uint8_t u8SrcEp, uint8_t u8DstEp)

API to send Remove All command in Group cluster.

Parameters

- **u8DstAddrMode** – **[in]** Destination address mode:
  - 0x00 – Reserved.
  - 0x01 – with group address;
  - 0x02 – with destination short address and endpoint;
  - 0x03 – with destination IEEE address and endpoint.
  - 0x04 ~ 0xFF – Reserved.
- **sDstAddr** – **[in]** Destination address for this command.
- **u8SrcEp** – **[in]** Source endpoint.
- **u8DstEp** – **[in]** Destination endpoint if dstAddrMode is 2 or 3.

void **zbhci_ZclGroupAddIfIdentify**(uint8_t u8DstAddrMode, *ts_DstAddr* sDstAddr, uint8_t u8SrcEp, uint8_t u8DstEp, uint16_t u16GroupId, uint8_t *pu8GroupName)

API to send View command in Group cluster.

Parameters

- **u8DstAddrMode** – **[in]** Destination address mode:
  - 0x00 – Reserved.
  - 0x01 – with group address;
  - 0x02 – with destination short address and endpoint;
  - 0x03 – with destination IEEE address and endpoint.
  - 0x04 ~ 0xFF – Reserved.
- **sDstAddr** – **[in]** Destination address for this command.
- **u8SrcEp** – **[in]** Source endpoint.
- **u8DstEp** – **[in]** Destination endpoint if dstAddrMode is 2 or 3.
- **u16GroupId** – **[in]** Group identifier.
- **pu8GroupName** – **[in]** Group name, character string.

void **zbhci_ZclIdentifyQuery**(uint8_t u8DstAddrMode, *ts_DstAddr* sDstAddr, uint8_t u8SrcEp, uint8_t u8DstEp, uint16_t u16IdentifyTime)

API to send Identify Query command in IDENTIFY cluster.

Parameters

- **u8DstAddrMode** – **[in]** Destination address mode:
  - 0x00 – Reserved.
  - 0x01 – with group address;
  - 0x02 – with destination short address and endpoint;
  - 0x03 – with destination IEEE address and endpoint.
  - 0x04 ~ 0xFF – Reserved.
- **sDstAddr** – **[in]** Destination address for this command.
- **u8SrcEp** – **[in]** Source endpoint.
- **u8DstEp** – **[in]** Destination endpoint if dstAddrMode is 2 or 3.
- **u16IdentifyTime** – **[in]** Unsigned 16-bit integer.

void **zbhci_ZclOnoffOn**(uint8_t u8DstAddrMode, *ts_DstAddr* sDstAddr, uint8_t u8SrcEp, uint8_t u8DstEp)

API to send ON command in ONOFF cluster.

Parameters

- **u8DstAddrMode** – **[in]** Destination address mode:
  - 0x00 – Reserved.
  - 0x01 – with group address;
  - 0x02 – with destination short address and endpoint;
  - 0x03 – with destination IEEE address and endpoint.
  - 0x04 ~ 0xFF – Reserved.
- **sDstAddr** – **[in]** Destination address for this command.
- **u8SrcEp** – **[in]** Source endpoint.
- **u8DstEp** – **[in]** Destination endpoint if dstAddrMode is 2 or 3.

void **zbhci_ZclOnoffOff**(uint8_t u8DstAddrMode, *ts_DstAddr* sDstAddr, uint8_t u8SrcEp, uint8_t u8DstEp)

API to send OFF command in ONOFF cluster.

Parameters

- **u8DstAddrMode** – **[in]** Destination address mode:
  - 0x00 – Reserved.
  - 0x01 – with group address;
  - 0x02 – with destination short address and endpoint;
  - 0x03 – with destination IEEE address and endpoint.
  - 0x04 ~ 0xFF – Reserved.
- **sDstAddr** – **[in]** Destination address for this command.
- **u8SrcEp** – **[in]** Source endpoint.
- **u8DstEp** – **[in]** Destination endpoint if dstAddrMode is 2 or 3.

void **zbhci_ZclOnoffToggle**(uint8_t u8DstAddrMode, *ts_DstAddr* sDstAddr, uint8_t u8SrcEp, uint8_t u8DstEp)

API to send TOGGLE command in ONOFF cluster.

Parameters

- **u8DstAddrMode** – **[in]** Destination address mode:
  - 0x00 – Reserved.
  - 0x01 – with group address;
  - 0x02 – with destination short address and endpoint;
  - 0x03 – with destination IEEE address and endpoint.
  - 0x04 ~ 0xFF – Reserved.
- **sDstAddr** – **[in]** Destination address for this command.
- **u8SrcEp** – **[in]** Source endpoint.
- **u8DstEp** – **[in]** Destination endpoint if dstAddrMode is 2 or 3.

void **zbhci_ZclLevelMove2level**(uint8_t u8DstAddrMode, *ts_DstAddr* sDstAddr, uint8_t u8SrcEp, uint8_t u8DstEp, uint8_t u8Level, uint16_t u16TransTime)

API to send Move To Level command in Level cluster.

Parameters

- **u8DstAddrMode** – **[in]** Destination address mode:
  - 0x00 – Reserved.
  - 0x01 – with group address;
  - 0x02 – with destination short address and endpoint;
  - 0x03 – with destination IEEE address and endpoint.
  - 0x04 ~ 0xFF – Reserved.
- **sDstAddr** – **[in]** Destination address for this command.
- **u8SrcEp** – **[in]** Source endpoint.
- **u8DstEp** – **[in]** Destination endpoint if dstAddrMode is 2 or 3.
- **u8Level** – **[in]** Level.
- **u16TransTime** – **[in]** Transition time, 1/10ths of a second.

void **zbhci_ZclLevelMove**(uint8_t u8DstAddrMode, *ts_DstAddr* sDstAddr, uint8_t u8SrcEp, uint8_t u8DstEp, uint8_t u8Mode, uint8_t u8Rate)

API to send Move command in Level cluster.

Parameters

- **u8DstAddrMode** – **[in]** Destination address mode:
  - 0x00 – Reserved.
  - 0x01 – with group address;
  - 0x02 – with destination short address and endpoint;
  - 0x03 – with destination IEEE address and endpoint.
  - 0x04 ~ 0xFF – Reserved.

- **sDstAddr** – **[in]** Destination address for this command.
- **u8SrcEp** – **[in]** Source endpoint.
- **u8DstEp** – **[in]** Destination endpoint if dstAddrMode is 2 or 3.
- **u8Mode** – **[in]** Move mode.
  - 0x00 – Up;
  - 0x01 – Down.
- **u8Rate** – **[in]** The rate field specifies the rate of movement in units per second.

void **zbhci_ZclLevelStep**(uint8_t u8DstAddrMode, *ts_DstAddr* sDstAddr, uint8_t u8SrcEp, uint8_t u8DstEp, uint8_t u8Mode, uint8_t u8StepSize, uint16_t u16TransTime)

API to send Step command in LEVEL cluster.

Parameters

- **u8DstAddrMode** – **[in]** Destination address mode:
  - 0x00 – Reserved.
  - 0x01 – with group address;
  - 0x02 – with destination short address and endpoint;
  - 0x03 – with destination IEEE address and endpoint.
  - 0x04 ~ 0xFF – Reserved.
- **sDstAddr** – **[in]** Destination address for this command.
- **u8SrcEp** – **[in]** Source endpoint.
- **u8DstEp** – **[in]** Destination endpoint if dstAddrMode is 2 or 3.
- **u8Mode** – **[in]** Move mode.
  - 0x00 – Up;
  - 0x01 – Down.
- **u8StepSize** – **[in]** A step is a change in the current level of 'step size' units.
- **u16TransTime** – **[in]** The transition time field specifies the time that shall be taken to perform the step, in 1/10ths of a second.

void **zbhci_ZclLevelStop**(uint8_t u8DstAddrMode, *ts_DstAddr* sDstAddr, uint8_t u8SrcEp, uint8_t u8DstEp)

API to send Stop command in Level cluster.

Parameters

- **u8DstAddrMode** – **[in]** Destination address mode:
  - 0x00 – Reserved.
  - 0x01 – with group address;
  - 0x02 – with destination short address and endpoint;
  - 0x03 – with destination IEEE address and endpoint.
  - 0x04 ~ 0xFF – Reserved.
- **sDstAddr** – **[in]** Destination address for this command.
- **u8SrcEp** – **[in]** Source endpoint.

- **u8DstEp** – **[in]** Destination endpoint if dstAddrMode is 2 or 3.

void **zbhci_ZclLevelMove2levelWithonoff**(uint8_t u8DstAddrMode, *ts_DstAddr* sDstAddr, uint8_t u8SrcEp, uint8_t u8DstEp, uint8_t u8Level, uint16_t u16TransTime)

API to send Move To Level command in Level cluster.

Parameters

- **u8DstAddrMode** – **[in]** Destination address mode:
  - 0x00 – Reserved.
  - 0x01 – with group address;
  - 0x02 – with destination short address and endpoint;
  - 0x03 – with destination IEEE address and endpoint.
  - 0x04 ~ 0xFF – Reserved.
- **sDstAddr** – **[in]** Destination address for this command.
- **u8SrcEp** – **[in]** Source endpoint.
- **u8DstEp** – **[in]** Destination endpoint if dstAddrMode is 2 or 3.
- **u8Level** – **[in]** Level.
- **u16TransTime** – **[in]** Transition time, 1/10ths of a second.

void **zbhci_ZclLevelMoveWithonoff**(uint8_t u8DstAddrMode, *ts_DstAddr* sDstAddr, uint8_t u8SrcEp, uint8_t u8DstEp, uint8_t u8Mode, uint8_t u8Rate)

API to send Move command in Level cluster.

Parameters

- **u8DstAddrMode** – **[in]** Destination address mode:
  - 0x00 – Reserved.
  - 0x01 – with group address;
  - 0x02 – with destination short address and endpoint;
  - 0x03 – with destination IEEE address and endpoint.
  - 0x04 ~ 0xFF – Reserved.
- **sDstAddr** – **[in]** Destination address for this command.
- **u8SrcEp** – **[in]** Source endpoint.
- **u8DstEp** – **[in]** Destination endpoint if dstAddrMode is 2 or 3.
- **u8Mode** – **[in]** Move mode.
  - 0x00 – Up;
  - 0x01 – Down.
- **u8Rate** – **[in]** The rate field specifies the rate of movement in units per second.

void **zbhci_ZclLevelStepWithonoff**(uint8_t u8DstAddrMode, *ts_DstAddr* sDstAddr, uint8_t u8SrcEp, uint8_t u8DstEp, uint8_t u8Mode, uint8_t u8StepSize, uint16_t u16TransTime)

API to send Step command in LEVEL cluster.

Parameters

- **u8DstAddrMode** – **[in]** Destination address mode:
  - 0x00 – Reserved.
  - 0x01 – with group address;
  - 0x02 – with destination short address and endpoint;
  - 0x03 – with destination IEEE address and endpoint.
  - 0x04 ~ 0xFF – Reserved.
- **sDstAddr** – **[in]** Destination address for this command.
- **u8SrcEp** – **[in]** Source endpoint.
- **u8DstEp** – **[in]** Destination endpoint if dstAddrMode is 2 or 3.
- **u8Mode** – **[in]** Move mode.
  - 0x00 – Up;
  - 0x01 – Down.
- **u8StepSize** – **[in]** A step is a change in the current level of 'step size' units.
- **u16TransTime** – **[in]** The transition time field specifies the time that shall be taken to perform the step, in 1/10ths of a second.

void **zbhci_ZclLevelStopWithonoff**(uint8_t u8DstAddrMode, *ts_DstAddr* sDstAddr, uint8_t u8SrcEp, uint8_t u8DstEp)

API to send Stop command in Level cluster.

Parameters

- **u8DstAddrMode** – **[in]** Destination address mode:
  - 0x00 – Reserved.
  - 0x01 – with group address;
  - 0x02 – with destination short address and endpoint;
  - 0x03 – with destination IEEE address and endpoint.
  - 0x04 ~ 0xFF – Reserved.
- **sDstAddr** – **[in]** Destination address for this command.
- **u8SrcEp** – **[in]** Source endpoint.
- **u8DstEp** – **[in]** Destination endpoint if dstAddrMode is 2 or 3.

void **zbhci_ZclSceneAdd**(uint8_t u8DstAddrMode, *ts_DstAddr* sDstAddr, uint8_t u8SrcEp, uint8_t u8DstEp, uint16_t u16GroupId, uint8_t u8SceneId, uint16_t u16TransTime, uint8_t u8SceneNameLen, uint8_t *pu8SceneName, uint8_t u8ExtFieldLen, uint8_t *pu8ExtFieldSets)

API to send add scene command in SCENE cluster.

Parameters

- **u8DstAddrMode** – **[in]** Destination address mode:
  - 0x00 – Reserved.
  - 0x01 – with group address;
  - 0x02 – with destination short address and endpoint;

  - 0x03 – with destination IEEE address and endpoint.
  - 0x04 ~ 0xFF – Reserved.
- **sDstAddr** – **[in]** Destination address for this command.
- **u8SrcEp** – **[in]** Source endpoint.
- **u8DstEp** – **[in]** Destination endpoint if dstAddrMode is 2 or 3.
- **u16GroupId** – **[in]** The group ID for which this scene applies.
- **u8SceneId** – **[in]** The identifier, unique within this group, which is used to identify this scene.
- **u16TransTime** – **[in]** The amount of time, in seconds, it will take for the device to change from its current state to the requested scene.
- **u8SceneNameLen** – **[in]** Length of scene name.
- **pu8SceneName** – **[in]** Scene name, char string.
- **u8ExtFieldLen** – **[in]** Length of extFieldSets field.
- **pu8ExtFieldSets** – **[in]** The sum of all such defines a scene.

void **zbhci_ZclSceneView**(uint8_t u8DstAddrMode, *ts_DstAddr* sDstAddr, uint8_t u8SrcEp, uint8_t u8DstEp, uint16_t u16GroupId, uint8_t u8SceneId)

API to send view scene command in SCENE cluster.

Parameters

- **u8DstAddrMode** – **[in]** Destination address mode:
  - 0x00 – without destination address and endpoint, for binding;
  - 0x01 – with group address;
  - 0x02 – with destination short address and endpoint;
  - 0x03 – with destination IEEE address and endpoint.
  - 0x04 ~ 0xFF – Reserved.
- **sDstAddr** – **[in]** Destination address for this command.
- **u8SrcEp** – **[in]** Source endpoint.
- **u8DstEp** – **[in]** Destination endpoint if dstAddrMode is 2 or 3.
- **u16GroupId** – **[in]** The group ID for which this scene applies.
- **u8SceneId** – **[in]** The identifier, unique within this group, which is used to identify this scene.

void **zbhci_ZclSceneRemove**(uint8_t u8DstAddrMode, *ts_DstAddr* sDstAddr, uint8_t u8SrcEp, uint8_t u8DstEp, uint16_t u16GroupId, uint8_t u8SceneId)

API to send remove scene command in SCENE cluster.

Parameters

- **u8DstAddrMode** – **[in]** Destination address mode:
  - 0x00 – Reserved.
  - 0x01 – with group address;
  - 0x02 – with destination short address and endpoint;

  - 0x03 – with destination IEEE address and endpoint.
  - 0x04 ~ 0xFF – Reserved.
- **sDstAddr** – **[in]** Destination address for this command.
- **u8SrcEp** – **[in]** Source endpoint.
- **u8DstEp** – **[in]** Destination endpoint if dstAddrMode is 2 or 3.
- **u16GroupId** – **[in]** The group ID for which this scene applies.
- **u8SceneId** – **[in]** The identifier, unique within this group, which is used to identify this scene.

void **zbhci_ZclSceneRemoveAll**(uint8_t u8DstAddrMode, *ts_DstAddr* sDstAddr, uint8_t u8SrcEp, uint8_t u8DstEp, uint16_t u16GroupId)

API to send remove all scene command in SCENE cluster.

Parameters

- **u8DstAddrMode** – **[in]** Destination address mode:
  - 0x00 – Reserved.
  - 0x01 – with group address;
  - 0x02 – with destination short address and endpoint;
  - 0x03 – with destination IEEE address and endpoint.
  - 0x04 ~ 0xFF – Reserved.
- **sDstAddr** – **[in]** Destination address for this command.
- **u8SrcEp** – **[in]** Source endpoint.
- **u8DstEp** – **[in]** Destination endpoint if dstAddrMode is 2 or 3.
- **u16GroupId** – **[in]** The group ID for which this scene applies.

void **zbhci_ZclSceneStore**(uint8_t u8DstAddrMode, *ts_DstAddr* sDstAddr, uint8_t u8SrcEp, uint8_t u8DstEp, uint16_t u16GroupId, uint8_t u8SceneId)

API to send store scene command in SCENE cluster.

Parameters

- **u8DstAddrMode** – **[in]** Destination address mode:
  - 0x00 – Reserved.
  - 0x01 – with group address;
  - 0x02 – with destination short address and endpoint;
  - 0x03 – with destination IEEE address and endpoint.
  - 0x04 ~ 0xFF – Reserved.
- **sDstAddr** – **[in]** Destination address for this command.
- **u8SrcEp** – **[in]** Source endpoint.
- **u8DstEp** – **[in]** Destination endpoint if dstAddrMode is 2 or 3.
- **u16GroupId** – **[in]** The group ID for which this scene applies.
- **u8SceneId** – **[in]** The identifier, unique within this group, which is used to identify this scene.

void **zbhci_ZclSceneRecall**(uint8_t u8DstAddrMode, *ts_DstAddr* sDstAddr, uint8_t u8SrcEp, uint8_t u8DstEp, uint16_t u16GroupId, uint8_t u8SceneId)

API to send recall scene command in SCENE cluster.

Parameters

- **u8DstAddrMode** – **[in]** Destination address mode:
  - 0x00 – Reserved.
  - 0x01 – with group address;
  - 0x02 – with destination short address and endpoint;
  - 0x03 – with destination IEEE address and endpoint.
  - 0x04 ~ 0xFF – Reserved.
- **sDstAddr** – **[in]** Destination address for this command.
- **u8SrcEp** – **[in]** Source endpoint.
- **u8DstEp** – **[in]** Destination endpoint if dstAddrMode is 2 or 3.
- **u16GroupId** – **[in]** The group ID for which this scene applies.
- **u8SceneId** – **[in]** The identifier, unique within this group, which is used to identify this scene.

void **zbhci_ZclSceneGetMembership**(uint8_t u8DstAddrMode, *ts_DstAddr* sDstAddr, uint8_t u8SrcEp, uint8_t u8DstEp, uint16_t u16GroupId)

API to send get scene membership command in SCENE cluster.

Parameters

- **u8DstAddrMode** – **[in]** Destination address mode:
  - 0x00 – Reserved.
  - 0x01 – with group address;
  - 0x02 – with destination short address and endpoint;
  - 0x03 – with destination IEEE address and endpoint.
  - 0x04 ~ 0xFF – Reserved.
- **sDstAddr** – **[in]** Destination address for this command.
- **u8SrcEp** – **[in]** Source endpoint.
- **u8DstEp** – **[in]** Destination endpoint if dstAddrMode is 2 or 3.
- **u16GroupId** – **[in]** The group ID for which this scene applies.

void **zbhci_ZclColorMove2hue**(uint8_t u8DstAddrMode, *ts_DstAddr* sDstAddr, uint8_t u8SrcEp, uint8_t u8DstEp, uint8_t u8Hue, uint8_t u8Direction, uint16_t u16TransTime)

This function sends a Move to Hue command to instruct a device to move its 'current hue' attribute to a target hue value in a continuous manner within a given time.

Parameters

- **u8DstAddrMode** – **[in]** Destination address mode:
  - 0x00 – Reserved.
  - 0x01 – with group address;

  - 0x02 – with destination short address and endpoint;
  - 0x03 – with destination IEEE address and endpoint.
  - 0x04 ~ 0xFF – Reserved.
- **sDstAddr** – **[in]** Destination address for this command.
- **u8SrcEp** – **[in]** Source endpoint.
- **u8DstEp** – **[in]** Destination endpoint if dstAddrMode is 2 or 3.
- **u8Hue** – **[in]** the target hue value.
- **u8Direction** – **[in]** the direction/path of the change in hue
- **u16TransTime** – **[in]** the time period, in tenths of a second, over which the change in hue should be implemented.

void **zbhci_ZclColorMove2Color**(uint8_t u8DstAddrMode, *ts_DstAddr* sDstAddr, uint8_t u8SrcEp, uint8_t u8DstEp, uint16_t u16ColorX, uint16_t u16ColorY, uint16_t u16TransTime)

This function sends a Move to Colour command to instruct a device to move its 'current x' and 'current y' attributes to target values in a continuous manner within a given time (where x and y are the chromaticities from the CIE xyY colour space).

Parameters

- **u8DstAddrMode** – **[in]** Destination address mode:
  - 0x00 – Reserved.
  - 0x01 – with group address;
  - 0x02 – with destination short address and endpoint;
  - 0x03 – with destination IEEE address and endpoint.
  - 0x04 ~ 0xFF – Reserved.
- **sDstAddr** – **[in]** Destination address for this command.
- **u8SrcEp** – **[in]** Source endpoint.
- **u8DstEp** – **[in]** Destination endpoint if dstAddrMode is 2 or 3.
- **u16ColorX** – **[in]** the target x-chromaticity in the CIE xyY colour space
- **u16ColorY** – **[in]** the target y-chromaticity in the CIE xyY colour space
- **u16TransTime** – **[in]** the time period, in tenths of a second, over which the colour change should be implemented.

void **zbhci_ZclColorMove2sat**(uint8_t u8DstAddrMode, *ts_DstAddr* sDstAddr, uint8_t u8SrcEp, uint8_t u8DstEp, uint8_t u8Saturation, uint16_t u16TransTime)

This function sends a Move to Saturation command to instruct a device to move its 'current saturation' attribute to a target saturation value in a continuous manner within a given time.

Parameters

- **u8DstAddrMode** – **[in]** Destination address mode:
  - 0x00 – Reserved.
  - 0x01 – with group address;
  - 0x02 – with destination short address and endpoint;
  - 0x03 – with destination IEEE address and endpoint.

  - 0x04 ~ 0xFF – Reserved.
- **sDstAddr** – **[in]** Destination address for this command.
- **u8SrcEp** – **[in]** Source endpoint.
- **u8DstEp** – **[in]** Destination endpoint if dstAddrMode is 2 or 3.
- **u8Saturation** – **[in]** the target saturation value.
- **u16TransTime** – **[in]** the time period, in tenths of a second, over which the change in saturation should be implemented.

void **zbhci_ZclColorMove2temp**(uint8_t u8DstAddrMode, *ts_DstAddr* sDstAddr, uint8_t u8SrcEp, uint8_t u8DstEp, uint16_t u16ColorTemperature, uint16_t u16TransTime)

This function sends a Move to Colour Temperature command to instruct a device to move its 'mired colour temperature' attribute to a target value in a continuous manner within a given time.

Parameters

- **u8DstAddrMode** – **[in]** Destination address mode:
  - 0x00 – Reserved.
  - 0x01 – with group address;
  - 0x02 – with destination short address and endpoint;
  - 0x03 – with destination IEEE address and endpoint.
  - 0x04 ~ 0xFF – Reserved.
- **sDstAddr** – **[in]** Destination address for this command.
- **u8SrcEp** – **[in]** Source endpoint.
- **u8DstEp** – **[in]** Destination endpoint if dstAddrMode is 2 or 3.
- **u16ColorTemperature** – **[in]** the target value of colour temperature
- **u16TransTime** – **[in]** the time period, in tenths of a second, over which the change in colour temperature should be implemented.

void **zbhci_ZclOtaImageNotify**(uint8_t u8DstAddrMode, *ts_DstAddr* sDstAddr, uint8_t u8SrcEp, uint8_t u8DstEp, uint8_t u8PayloadType, uint8_t u8QueryJitter)

This function issues an Image Notify message to one or more clients to indicate that a new application image is available for download.

Parameters

- **u8DstAddrMode** – **[in]** Destination address mode:
  - 0x00 – Reserved.
  - 0x01 – with group address;
  - 0x02 – with destination short address and endpoint;
  - 0x03 – with destination IEEE address and endpoint.
  - 0x04 ~ 0xFF – Reserved.
- **sDstAddr** – **[in]** Destination address for this command.
- **u8SrcEp** – **[in]** Source endpoint.
- **u8DstEp** – **[in]** Destination endpoint if dstAddrMode is 2 or 3.

- **u8PayloadType** – **[in]**
  - 0x00 – Query jitter;
  - 0x01 – Query jitter and manufacturer code;
  - 0x02 – Query jitter, manufacturer code, and image type;
  - 0x03 - Query jitter, manufacturer code, image type, and new file version.
- **queryJitter** – **[in]** By using the parameter, it prevents a single notification of a new OTA upgrade image from flooding the upgrade server with requests from clients.

Defines

ZCL_DATA_TYPE_NO_DATA

ZCL_DATA_TYPE_DATA8

ZCL_DATA_TYPE_DATA16

ZCL_DATA_TYPE_DATA24

ZCL_DATA_TYPE_DATA32

ZCL_DATA_TYPE_DATA40

ZCL_DATA_TYPE_DATA48

ZCL_DATA_TYPE_DATA56

ZCL_DATA_TYPE_DATA64

ZCL_DATA_TYPE_BOOLEAN

ZCL_DATA_TYPE_BITMAP8

ZCL_DATA_TYPE_BITMAP16

ZCL_DATA_TYPE_BITMAP24

ZCL_DATA_TYPE_BITMAP32

ZCL_DATA_TYPE_BITMAP40

ZCL_DATA_TYPE_BITMAP48

ZCL_DATA_TYPE_BITMAP56

ZCL_DATA_TYPE_BITMAP64

ZCL_DATA_TYPE_UINT8

ZCL_DATA_TYPE_UINT16

ZCL_DATA_TYPE_UINT24

ZCL_DATA_TYPE_UINT32

ZCL_DATA_TYPE_UINT40

ZCL_DATA_TYPE_UINT48

ZCL_DATA_TYPE_UINT56

ZCL_DATA_TYPE_UINT64

ZCL_DATA_TYPE_INT8

ZCL_DATA_TYPE_INT16

ZCL_DATA_TYPE_INT24

ZCL_DATA_TYPE_INT32

ZCL_DATA_TYPE_INT40

ZCL_DATA_TYPE_INT48

ZCL_DATA_TYPE_INT56

ZCL_DATA_TYPE_INT64

ZCL_DATA_TYPE_ENUM8

ZCL_DATA_TYPE_ENUM16

ZCL_DATA_TYPE_SEMI_PREC

ZCL_DATA_TYPE_SINGLE_PREC

ZCL_DATA_TYPE_DOUBLE_PREC

ZCL_DATA_TYPE_OCTET_STR

ZCL_DATA_TYPE_CHAR_STR

ZCL_DATA_TYPE_LONG_OCTET_STR

ZCL_DATA_TYPE_LONG_CHAR_STR

ZCL_DATA_TYPE_ARRAY

ZCL_DATA_TYPE_STRUCT

ZCL_DATA_TYPE_SET

ZCL_DATA_TYPE_BAG

ZCL_DATA_TYPE_TOD

ZCL_DATA_TYPE_DATE

ZCL_DATA_TYPE_UTC

ZCL_DATA_TYPE_CLUSTER_ID

ZCL_DATA_TYPE_ATTR_ID

ZCL_DATA_TYPE_BAC_OID

ZCL_DATA_TYPE_IEEE_ADDR

ZCL_DATA_TYPE_128_BIT_SEC_KEY

ZCL_DATA_TYPE_UNKNOWN

Typedefs

typedef enum *te_HCIMsgType* **te_HCIMsgType**

typedef enum *te_AddrMode* **te_AddrMode**

typedef enum *te_MsgBdbCommissionTouchlinkRole* **te_MsgBdbCommissionTouchlinkRole**

typedef enum *te_MsgBdbCommissionFindbindRole* **te_MsgBdbCommissionFindbindRole**

typedef enum *te_MsgBdbDongleWorkingMode* **te_MsgBdbDongleWorkingMode**

typedef enum *te_BdbTxPowerLevel* **te_BdbTxPowerLevel**

typedef union *ts_DstAddr* **ts_DstAddr**

typedef union *t_AttrData* **t_AttrData**

typedef struct *ts_AttrList* **ts_AttrList**

typedef enum *te_MsgAckStatus* **te_MsgAckStatus**

typedef struct *ts_MsgAckPayload* **ts_MsgAckPayload**

typedef struct *ts_MsgBdbCommissionFormationRspPayload* **ts_MsgBdbCommissionFormationRspPayload**

typedef struct *ts_MsgNetworkStateRspPayload* **ts_MsgNetworkStateRspPayload**

typedef struct *ts_MsgDiscoveryNwkAddrRspPayload* **ts_MsgDiscoveryNwkAddrRspPayload**

typedef struct *ts_MsgDiscoveryNwkAddrRspPayload* **ts_MsgDiscoveryIEEEAddrRspPayload**

typedef struct *ts_MsgDiscoveryNodeDescRspPayload* **ts_MsgDiscoveryNodeDescRspPayload**

typedef struct *ts_MsgDiscoverySimpleDescRspPayload* **ts_MsgDiscoverySimpleDescRspPayload**

typedef struct *ts_MsgDiscoveryMatchDescRspPayload* **ts_MsgDiscoveryMatchDescRspPayload**

typedef struct *ts_MsgDiscoveryActiveEpRspPayload* **ts_MsgDiscoveryActiveEpRspPayload**

typedef struct *ts_MsgBindRspPayload* **ts_MsgBindRspPayload**

typedef struct *ts_MsgBindRspPayload* **ts_MsgUnbindRspPayload**

typedef struct *ts_NeighborTable* **ts_NeighborTable**

typedef struct *ts_MsgMgmtLqiRspPayload* **ts_MsgMgmtLqiRspPayload**

typedef struct *ts_BindTabList* **ts_BindTabList**

typedef struct *ts_MsgMgmtBindRspPayload* **ts_MsgMgmtBindRspPayload**

typedef struct *ts_MsgMgmtLeaveRspPayload* **ts_MsgMgmtLeaveRspPayload**

typedef struct *ts_MsgMgmtPermitJoinRspPayload* **ts_MsgMgmtPermitJoinRspPayload**

typedef struct *ts_MsgNodesJoinedGetRspPayload* **ts_MsgNodesJoinedGetRspPayload**

typedef struct *ts_MsgTxRxPerformceTestRspPayload* **ts_MsgTxRxPerformceTestRspPayload**

typedef struct *ts_MsgNodesDevAnnceRspPayload* **ts_MsgNodesDevAnnceRspPayload**

typedef struct *ts_MsgAfDataSendTestRspPayload* **ts_MsgAfDataSendTestRspPayload**

typedef struct *ts_MsgLeaveIndicationPayload* **ts_MsgLeaveIndicationPayload**

typedef struct *ts_AttrRead* **ts_AttrRead**

typedef struct *ts_MsgZclAttrReadRspPayload* **ts_MsgZclAttrReadRspPayload**

typedef struct *ts_AttrWrite* **ts_AttrWrite**

typedef struct *ts_MsgZclAttrWriteRspPayload* **ts_MsgZclAttrWriteRspPayload**

typedef struct *ts_AttrConfigReport* **ts_AttrConfigReport**

typedef struct *ts_MsgZclConfigReportRspPayload* **ts_MsgZclConfigReportRspPayload**

typedef struct *ts_AttrReadConfigReport* **ts_AttrReadConfigReport**

typedef struct *ts_MsgZclReadReportCfgRspPayload* **ts_MsgZclReadReportCfgRspPayload**

typedef struct *ts_MsgZclReportMsgRcvPayload* **ts_MsgZclReportMsgRcvPayload**

typedef struct *ts_MsgZclGroupAddRspPayload* **ts_MsgZclGroupAddRspPayload**

typedef struct *ts_MsgZclGroupViewRspPayload* **ts_MsgZclGroupViewRspPayload**

typedef struct *ts_MsgZclGroupGetMembershipRspPayload* **ts_MsgZclGroupGetMembershipRspPayload**

typedef struct *ts_MsgZclGroupRemoveRspPayload* **ts_MsgZclGroupRemoveRspPayload**

typedef struct *ts_MsgZclIdentifyQueryRspPayload* **ts_MsgZclIdentifyQueryRspPayload**

typedef struct *ts_MsgZclOnOffCmdRcvPayload* **ts_MsgZclOnOffCmdRcvPayload**

typedef struct *ts_MsgZclSceneAddRspPayload* **ts_MsgZclSceneAddRspPayload**

typedef struct *ts_MsgZclSceneViewRspPayload* **ts_MsgZclSceneViewRspPayload**

typedef struct *ts_MsgZclSceneRemoveRspPayload* **ts_MsgZclSceneRemoveRspPayload**

typedef struct *ts_MsgZclSceneRemoveAllRspPayload* **ts_MsgZclSceneRemoveAllRspPayload**

typedef struct *ts_MsgZclSceneStoreRspPayload* **ts_MsgZclSceneStoreRspPayload**

typedef struct *ts_MsgZclSceneGetMenbershipRspPayload* **ts_MsgZclSceneGetMenbershipRspPayload**

typedef struct *ts_MsgDataConfirmPayload* **ts_MsgDataConfirmPayload**

typedef struct *ts_MsgMacAddrIndPayload* **ts_MsgMacAddrIndPayload**

typedef struct *ts_MsgNodeLeaveIndPayload* **ts_MsgNodeLeaveIndPayload**

typedef struct *ts_HciMsg* **ts_HciMsg**

Enums

enum **te_HCIMsgType**

ZigBee HCI message type

Values:

enumerator **ZBHCI_CMD_BDB_COMMISSION_FORMATION**

enumerator **ZBHCI_CMD_BDB_COMMISSION_STEER**

enumerator **ZBHCI_CMD_BDB_COMMISSION_TOUCHLINK**

enumerator **ZBHCI_CMD_BDB_COMMISSION_FINDBIND**

enumerator **ZBHCI_CMD_BDB_FACTORY_RESET**

enumerator **ZBHCI_CMD_BDB_PRE_INSTALL_CODE**

enumerator **ZBHCI_CMD_BDB_CHANNEL_SET**

enumerator **ZBHCI_CMD_BDB_DONGLE_WORKING_MODE_SET**

enumerator **ZBHCI_CMD_BDB_NODE_DELETE**

enumerator **ZBHCI_CMD_BDB_TX_POWER_SET**

enumerator **ZBHCI_CMD_ACKNOWLEDGE**

enumerator **ZBHCI_CMD_BDB_COMMISSION_FORMATION_RSP**

enumerator **ZBHCI_CMD_NETWORK_STATE_REQ**

enumerator **ZBHCI_CMD_NETWORK_STATE_RSP**

enumerator **ZBHCI_CMD_NETWORK_STATE_REPORT**

enumerator **ZBHCI_CMD_DISCOVERY_NWK_ADDR_REQ**

enumerator **ZBHCI_CMD_DISCOVERY_IEEE_ADDR_REQ**

enumerator **ZBHCI_CMD_DISCOVERY_NODE_DESC_REQ**

enumerator **ZBHCI_CMD_DISCOVERY_SIMPLE_DESC_REQ**

enumerator **ZBHCI_CMD_DISCOVERY_MATCH_DESC_REQ**

enumerator **ZBHCI_CMD_DISCOVERY_ACTIVE_EP_REQ**

enumerator **ZBHCI_CMD_DISCOVERY_LEAVE_REQ**

enumerator **ZBHCI_CMD_DISCOVERY_NWK_ADDR_RSP**

enumerator **ZBHCI_CMD_DISCOVERY_IEEE_ADDR_RSP**

enumerator **ZBHCI_CMD_DISCOVERY_NODE_DESC_RSP**

enumerator **ZBHCI_CMD_DISCOVERY_SIMPLE_DESC_RSP**

enumerator **ZBHCI_CMD_DISCOVERY_MATCH_DESC_RSP**

enumerator **ZBHCI_CMD_DISCOVERY_ACTIVE_EP_RSP**

enumerator **ZBHCI_CMD_BINDING_REQ**

enumerator **ZBHCI_CMD_UNBINDING_REQ**

enumerator **ZBHCI_CMD_BINDING_RSP**

enumerator **ZBHCI_CMD_UNBINDING_RSP**

enumerator **ZBHCI_CMD_MGMT_LQI_REQ**

enumerator **ZBHCI_CMD_MGMT_BIND_REQ**

enumerator **ZBHCI_CMD_MGMT_LEAVE_REQ**

enumerator **ZBHCI_CMD_MGMT_DIRECT_JOIN_REQ**

enumerator **ZBHCI_CMD_MGMT_PERMIT_JOIN_REQ**

enumerator **ZBHCI_CMD_MGMT_NWK_UPDATE_REQ**

enumerator **ZBHCI_CMD_MGMT_LQI_RSP**

enumerator **ZBHCI_CMD_MGMT_BIND_RSP**

enumerator **ZBHCI_CMD_MGMT_LEAVE_RSP**

enumerator **ZBHCI_CMD_MGMT_DIRECT_JOIN_RSP**

enumerator **ZBHCI_CMD_MGMT_PERMIT_JOIN_RSP**

enumerator **ZBHCI_CMD_MGMT_NWK_UPDATE_RSP**

enumerator **ZBHCI_CMD_NODES_JOINED_GET_REQ**

enumerator **ZBHCI_CMD_NODES_TOGLE_TEST_REQ**

enumerator **ZBHCI_CMD_TXRX_PERFORMANCE_TEST_REQ**

enumerator **ZBHCI_CMD_AF_DATA_SEND_TEST_REQ**

enumerator **ZBHCI_CMD_NODES_JOINED_GET_RSP**

enumerator **ZBHCI_CMD_NODES_TOGLE_TEST_RSP**

enumerator **ZBHCI_CMD_TXRX_PERFORMANCE_TEST_RSP**

enumerator **ZBHCI_CMD_NODES_DEV_ANNCE_IND**

enumerator **ZBHCI_CMD_AF_DATA_SEND_TEST_RSP**

enumerator **ZBHCI_CMD_LEAVE_INDICATION**

enumerator **ZBHCI_CMD_ZCL_ATTR_READ**

enumerator **ZBHCI_CMD_ZCL_ATTR_WRITE**

enumerator **ZBHCI_CMD_ZCL_CONFIG_REPORT**

enumerator **ZBHCI_CMD_ZCL_READ_REPORT_CFG**

enumerator **ZBHCI_CMD_ZCL_LOCAL_ATTR_READ**

enumerator **ZBHCI_CMD_ZCL_LOCAL_ATTR_WRITE**

enumerator **ZBHCI_CMD_ZCL_SEND_REPORT_CMD**

enumerator **ZBHCI_CMD_ZCL_ATTR_READ_RSP**

enumerator **ZBHCI_CMD_ZCL_ATTR_WRITE_RSP**

enumerator **ZBHCI_CMD_ZCL_CONFIG_REPORT_RSP**

enumerator **ZBHCI_CMD_ZCL_READ_REPORT_CFG_RSP**

enumerator **ZBHCI_CMD_ZCL_REPORT_MSG_RCV**

enumerator **ZBHCI_CMD_ZCL_LOCAL_ATTR_READ_RSP**

enumerator **ZBHCI_CMD_ZCL_LOCAL_ATTR_WRITE_RSP**

enumerator **ZBHCI_CMD_ZCL_ATTR_WRITE_RCV**

enumerator **ZBHCI_CMD_ZCL_BASIC**

enumerator **ZBHCI_CMD_ZCL_BASIC_RESET**

enumerator **ZBHCI_CMD_ZCL_GROUP**

enumerator **ZBHCI_CMD_ZCL_GROUP_ADD**

enumerator **ZBHCI_CMD_ZCL_GROUP_VIEW**

enumerator **ZBHCI_CMD_ZCL_GROUP_GET_MEMBERSHIP**

enumerator **ZBHCI_CMD_ZCL_GROUP_REMOVE**

enumerator **ZBHCI_CMD_ZCL_GROUP_REMOVE_ALL**

enumerator **ZBHCI_CMD_ZCL_GROUP_ADD_IF_IDENTIFY**

enumerator **ZBHCI_CMD_ZCL_GROUP_ADD_RSP**

enumerator **ZBHCI_CMD_ZCL_GROUP_VIEW_RSP**

enumerator **ZBHCI_CMD_ZCL_GROUP_GET_MEMBERSHIP_RSP**

enumerator **ZBHCI_CMD_ZCL_GROUP_REMOVE_RSP**

enumerator **ZBHCI_CMD_ZCL_IDENTIFY**

enumerator **ZBHCI_CMD_ZCL_IDENTIFY_QUERY**

enumerator **ZBHCI_CMD_ZCL_IDENTIFY_QUERY_RSP**

enumerator **ZBHCI_CMD_ZCL_ONOFF**

enumerator **ZBHCI_CMD_ZCL_ONOFF_ON**

enumerator **ZBHCI_CMD_ZCL_ONOFF_OFF**

enumerator **ZBHCI_CMD_ZCL_ONOFF_TOGGLE**

enumerator **ZBHCI_CMD_ZCL_ONOFF_CMD_RCV**

enumerator **ZBHCI_CMD_ZCL_LEVEL**

enumerator **ZBHCI_CMD_ZCL_LEVEL_MOVE2LEVEL**

enumerator **ZBHCI_CMD_ZCL_LEVEL_MOVE**

enumerator **ZBHCI_CMD_ZCL_LEVEL_STEP**

enumerator **ZBHCI_CMD_ZCL_LEVEL_STOP**

enumerator **ZBHCI_CMD_ZCL_LEVEL_MOVE2LEVEL_WITHONOFF**

enumerator **ZBHCI_CMD_ZCL_LEVEL_MOVE_WITHONOFF**

enumerator **ZBHCI_CMD_ZCL_LEVEL_STEP_WITHONOFF**

enumerator **ZBHCI_CMD_ZCL_LEVEL_STOP_WITHONOFF**

enumerator **ZBHCI_CMD_ZCL_SCENE**

enumerator **ZBHCI_CMD_ZCL_SCENE_ADD**

enumerator **ZBHCI_CMD_ZCL_SCENE_VIEW**

enumerator **ZBHCI_CMD_ZCL_SCENE_REMOVE**

enumerator **ZBHCI_CMD_ZCL_SCENE_REMOVE_ALL**

enumerator **ZBHCI_CMD_ZCL_SCENE_STORE**

enumerator **ZBHCI_CMD_ZCL_SCENE_RECALL**

enumerator **ZBHCI_CMD_ZCL_SCENE_GET_MEMBERSHIP**

enumerator **ZBHCI_CMD_ZCL_SCENE_ADD_RSP**

enumerator **ZBHCI_CMD_ZCL_SCENE_VIEW_RSP**

enumerator **ZBHCI_CMD_ZCL_SCENE_REMOVE_RSP**

enumerator **ZBHCI_CMD_ZCL_SCENE_REMOVE_ALL_RSP**

enumerator **ZBHCI_CMD_ZCL_SCENE_STORE_RSP**

enumerator **ZBHCI_CMD_ZCL_SCENE_GET_MEMBERSHIP_RSP**

enumerator **ZBHCI_CMD_ZCL_COLOR**

enumerator **ZBHCI_CMD_ZCL_COLOR_MOVE2HUE**

enumerator **ZBHCI_CMD_ZCL_COLOR_MOVE2COLOR**

enumerator **ZBHCI_CMD_ZCL_COLOR_MOVE2SAT**

enumerator **ZBHCI_CMD_ZCL_COLOR_MOVE2TEMP**

enumerator **ZBHCI_CMD_ZCL_IAS_ZONE**

enumerator **ZBHCI_CMD_ZCL_OTA_IMAGE_NOTIFY**

enumerator **ZBHCI_CMD_DATA_CONFIRM**

enumerator **ZBHCI_CMD_MAC_ADDR_IND**

enumerator **ZBHCI_CMD_NODE_LEAVE_IND**

enumerator **ZBHCI_CMD_AF_DATA_SEND**

enum **te_AddrMode**

Values:

enumerator **E_ADDR_MODE_BIND_ADDR**

for bind, without address and Endpoint

Note: Use with caution, it is invalid

---

enumerator **E_ADDR_MODE_GROUP_ADDR**

for group-casting: only need group address

enumerator **E_ADDR_MODE_SHORT_ADDR**

for unicasting with nwk address, with Endpoint

enumerator **E_ADDR_MODE_IEEE_ADDR**

for unicasting with ieee address, with Endpoint

enum **te_MsgBdbCommissionTouchlinkRole**

Values:

enumerator **E_BDB_COMMISSION_TOUCHLINK_ROLE_INITIATOR**

Touch link initiator

enumerator **E_BDB_COMMISSION_TOUCHLINK_ROLE_TARGET**

Touch link target

enum **te_MsgBdbCommissionFindbindRole**

Values:

enumerator **E_BDB_COMMISSION_FINDBIND_ROLE_INITIATOR**

Find & Bind initiator

enumerator **E_BDB_COMMISSION_FINDBIND_ROLE_TARGET**

Find & Bind target

enum **te_MsgBdbDongleWorkingMode**

Values:

enumerator **E_BDB_DONGLE_WORKING_MODE_GET_MAC_ADDR_MODE**

enumerator **E_BDB_DONGLE_WORKING_MODE_NORMAL_MODE**

enum **te_BdbTxPowerLevel**

Values:

enumerator **E_BDB_TX_POWER_LEVEL_P11p26dBm**

enumerator **E_BDB_TX_POWER_LEVEL_P11p09dBm**

enumerator **E_BDB_TX_POWER_LEVEL_P10p83dBm**

enumerator **E_BDB_TX_POWER_LEVEL_P10p62dBm**

enumerator **E_BDB_TX_POWER_LEVEL_P10p30dBm**

enumerator **E_BDB_TX_POWER_LEVEL_P10p05dBm**

enumerator **E_BDB_TX_POWER_LEVEL_P9p79dBm**

enumerator **E_BDB_TX_POWER_LEVEL_P9p54dBm**

enumerator **E_BDB_TX_POWER_LEVEL_P9p23dBm**

enumerator **E_BDB_TX_POWER_LEVEL_P8p92dBm**

enumerator **E_BDB_TX_POWER_LEVEL_P8p57dBm**

enumerator **E_BDB_TX_POWER_LEVEL_P8p20dBm**

enumerator **E_BDB_TX_POWER_LEVEL_P7p80dBm**

enumerator **E_BDB_TX_POWER_LEVEL_P7p37dBm**

enumerator **E_BDB_TX_POWER_LEVEL_P6p91dBm**

enumerator **E_BDB_TX_POWER_LEVEL_P6p45dBm**

enumerator **E_BDB_TX_POWER_LEVEL_P5p92dBm**

enumerator **E_BDB_TX_POWER_LEVEL_P5p33dBm**

enumerator **E_BDB_TX_POWER_LEVEL_P4p69dBm**

enumerator **E_BDB_TX_POWER_LEVEL_P3p99dBm**

enumerator **E_BDB_TX_POWER_LEVEL_P3p50dBm**

enumerator **E_BDB_TX_POWER_LEVEL_P3p33dBm**

enumerator **E_BDB_TX_POWER_LEVEL_P3p13dBm**

enumerator **E_BDB_TX_POWER_LEVEL_P2p93dBm**

enumerator **E_BDB_TX_POWER_LEVEL_P2p60dBm**

enumerator **E_BDB_TX_POWER_LEVEL_P2p36dBm**

enumerator **E_BDB_TX_POWER_LEVEL_P2p10dBm**

enumerator **E_BDB_TX_POWER_LEVEL_P1p83dBm**

enumerator **E_BDB_TX_POWER_LEVEL_P1p56dBm**

enumerator **E_BDB_TX_POWER_LEVEL_P1p25dBm**

enumerator **E_BDB_TX_POWER_LEVEL_P0p71dBm**

enumerator **E_BDB_TX_POWER_LEVEL_P0p52dBm**

enumerator **E_BDB_TX_POWER_LEVEL_N0p28dBm**

enumerator **E_BDB_TX_POWER_LEVEL_N0p51dBm**

enumerator **E_BDB_TX_POWER_LEVEL_N0p74dBm**

enumerator **E_BDB_TX_POWER_LEVEL_N1p21dBm**

enumerator **E_BDB_TX_POWER_LEVEL_N1p69dBm**

enumerator **E_BDB_TX_POWER_LEVEL_N2p23dBm**

enumerator **E_BDB_TX_POWER_LEVEL_N2p84dBm**

enumerator **E_BDB_TX_POWER_LEVEL_N3p48dBm**

enumerator **E_BDB_TX_POWER_LEVEL_N4p18dBm**

enumerator **E_BDB_TX_POWER_LEVEL_N4p97dBm**

enumerator **E_BDB_TX_POWER_LEVEL_N5p85dBm**

enumerator **E_BDB_TX_POWER_LEVEL_N6p83dBm**

enumerator **E_BDB_TX_POWER_LEVEL_N7p88dBm**

enumerator **E_BDB_TX_POWER_LEVEL_N9p14dBm**

enumerator **E_BDB_TX_POWER_LEVEL_N10p70dBm**

enumerator **E_BDB_TX_POWER_LEVEL_N12p57dBm**

enumerator **E_BDB_TX_POWER_LEVEL_N15p01dBm**

enumerator **E_BDB_TX_POWER_LEVEL_N18p40dBm**

enumerator **E_BDB_TX_POWER_LEVEL_N24p28dBm**

enum **te_MsgAckStatus**

Status message.

Values:

enumerator **ZBHCI_MSG_STATUS_SUCCESS**

Success

enumerator **ZBHCI_MSG_STATUS_INCORRECT_PARAMETERS**

Incorrect parameters

enumerator **ZBHCI_MSG_STATUS_UNHANDLED_COMMAND**

Unhandled command

enumerator **ZBHCI_MSG_STATUS_BUSY**

Busy

enumerator **ZBHCI_MSG_STATUS_NO_MEMORY**

No memory

enumerator **ZBHCI_MSG_STATUS_STACK_ALREADY_STARTED**

Stack already started

union **ts_DstAddr**

#include <zbhci_commom.h>

Public Members

uint16_t **u16DstAddr**

uint64_t **u64DstAddr**

union **t_AttrData**

#include <zbhci_commom.h>

Public Members

uint8_t **u8AttrData**

uint16_t **u16AttrData**

uint32_t **u32AttrData**

uint64_t **u64AttrData**

uint8_t **au8AttrData**[128]

struct **ts_AttrList**

#include <zbhci_commom.h>

Public Members

uint16_t **u16AttrID**

uint8_t **u8DataType**

uint16_t **u16DataLen**

t_AttrData **uAttrData**

struct **ts_MsgAckPayload**

#include <zbhci_commom.h>

Public Members

uint16_t **u16MsgType**

HCI command message type.

te_MsgAckStatus **eStatus**

0 = Success; 1 = Wrong parameter; 2 = Unsupported command; 3 = Busy; 4 = No memory.

uint8_t **u8Reserved**

Reserved

struct **ts_MsgBdbCommissionFormationRspPayload**

#include <zbhci_commom.h>

Public Members

uint8_t **u8Status**

struct **ts_MsgNetworkStateRspPayload**

#include <zbhci_commom.h>

Public Members

uint16_t **u16NwkAddr**

uint64_t **u64IeeeAddr**

uint16_t **u16PanId**

uint64_t **u64extPanId**

uint8_t **u8Channel**

struct **ts_MsgDiscoveryNwkAddrRspPayload**

#include <zbhci_commom.h>

Public Members

uint8_t **u8SeqNum**

ZDP transaction sequence number.

uint8_t **u8Status**

The status of the request command.

uint64_t **u64IEEEAddr**

64-bit address for the Remote Device.

uint64_t **u16NwkAddr**

16-bit address for the Remote Device.

uint8_t **u8NumAssocDev**

Count of the number of 16-bit short address to follow.

uint8_t **u8StartIdx**

Starting index into the list of associated devices for this report.

uint16_t **au16AssocDevList**[256]

The list of associated devices.

struct **ts_MsgDiscoveryNodeDescRspPayload**

#include <zbhci_commom.h>

Public Members

uint8_t **u8SeqNum**

ZDP transaction sequence number.

uint8_t **u8Status**

The status of the request command.

uint16_t **u16NwkAddrOfInterest**

NWK address for the request.

uint8_t **au8NodeDesc**[256]

This field shall only be included in the frame if the status field is SUCCESS.

struct **ts_MsgDiscoverySimpleDescRspPayload**

#include <zbhci_commom.h>

Public Members

uint8_t **u8SeqNum**

ZDP transaction sequence number.

uint8_t **u8Status**

The status of the request command.

uint16_t **u16NwkAddrOfInterest**

NWK address for the request.

uint8_t **u8Length**

The length of simple description.

uint8_t **au8SimpleDesc**[256]

This field shall only be included in the frame if the status field is SUCCESS.

struct **ts_MsgDiscoveryMatchDescRspPayload**

#include <zbhci_commom.h>

Public Members

uint8_t **u8SeqNum**

ZDP transaction sequence number.

uint8_t **u8Status**

The status of the request command.

uint16_t **u16NwkAddrOfInterest**

NWK address for the request.

uint8_t **u8MatchLen**

The count of endpoints on the Remote Device that match the request criteria.

uint8_t **au8MatchList**[256]

List of bytes each of which represents an 8-bit endpoint.

struct **ts_MsgDiscoveryActiveEpRspPayload**

#include <zbhci_commom.h>

Public Members

uint8_t **u8SeqNum**

ZDP transaction sequence number.

uint8_t **u8Status**

The status of the request command.

uint16_t **u16NwkAddrOfInterest**

NWK address for the request.

uint8_t **u8ActiveEpCount**

The count of active endpoints.

uint8_t **au8EpList**[256]

List of active endpoints.

struct **ts_MsgBindRspPayload**

#include <zbhci_commom.h>

Public Members

uint8_t **u8SeqNum**

ZDP transaction sequence number.

uint8_t **u8Status**

The status of the request command.

struct **ts_NeighborTable**

#include <zbhci_commom.h>

Public Members

uint64_t **ext_pan_id**

uint64_t **ext_addr**

uint16_t **network_addr**

uint8_t **deviceType**

uint8_t **rxOnWhenIdle**

uint8_t **relationship**

uint8_t **reserved1**

uint8_t **permitJoining**

uint8_t **reserved2**

uint8_t **depth**

uint8_t **lqi**

struct **ts_MsgMgmtLqiRspPayload**

#include <zbhci_commom.h>

Public Members

uint8_t **u8SeqNum**

ZDP transaction sequence number.

uint8_t **u8Status**

The status of the request command.

uint8_t **u8NeighborTabEntries**

Total number of Neighbor Table entries within the Remote Device.

uint8_t **u8StartIdx**

Starting index within the Neighbor Table to begin reporting for the neighborTabList.

uint8_t **u8NeighborTabListCount**

Number of Neighbor Table entries included within neighborTabList.

ts_NeighborTable **asNeighborTable**[8]

A list of descriptors, beginning with the startIdx element and continuing for neighborTabListCount.

struct **ts_BindTabList**

#include <zbhci_commom.h>

Public Members

uint64_t **u64SrcAddr**

the device who build the binding table

uint8_t **u8SrcEndpoint**

The source endpoint for the binding entry

uint8_t **u16ClusterId**

The identifier of the cluster on the source device that is bound to the destination

uint8_t **u8DstAddrMode**

destination address mode 0x01 - 16-bit group address for dstAddr and dstEp not present 0x03 - 64-bit extAddr for dstAddr and estEp present

uint64_t **u64DstExtAddr**

uint8_t **u8DstEndpoint**

struct *ts_BindTabList*.[anonymous].[anonymous] **sDstExtAddr**

uint16_t **u16DstGroupAddr**

union *ts_BindTabList*.[anonymous] **uDstAddr**

The destination address for the binding entry

struct **ts_MsgMgmtBindRspPayload**

#include <zbhci_commom.h>

Public Members

uint8_t **u8SeqNum**

ZDP transaction sequence number.

uint8_t **u8Status**

The status of the request command.

uint8_t **u8BindingTabEntries**

Total number of Binding Table entries within the Remote Device.

uint8_t **u8StartIdx**

Starting index within the Binding Table to begin reporting for the bindingTabList.

uint8_t **u8BindingTabListCount**

Number of Binding Table entries included within bindingTabList.

ts_BindTabList **asBindingTabList**[4]

A list of descriptors, beginning with the startIdx element and continuing for bindingTabListCount.

struct **ts_MsgMgmtLeaveRspPayload**

#include <zbhci_commom.h>

Public Members

uint8_t **u8SeqNum**

ZDP transaction sequence number.

uint8_t **u8Status**

The status of the request command.

struct **ts_MsgMgmtPermitJoinRspPayload**

#include <zbhci_commom.h>

Public Members

uint8_t **u8SeqNum**

ZDP transaction sequence number.

uint8_t **u8Status**

The status of the request command.

struct **ts_MsgNodesJoinedGetRspPayload**

#include <zbhci_commom.h>

Public Members

uint16_t **u16TotalCnt**

The total count of the joined nodes.

uint16_t **u16StartIdx**

Starting index within the mac address list.

uint8_t **u8ListCount**

The count of the MAC address list in the current packet.

uint8_t **u8Status**

The status of the request command.

uint64_t **au64MacAddrList**[6]

The MAC address list in the current packet.

uint16_t **au16ShortAddrList**[6]

The MAC address list in the current packet.

struct **ts_MsgTxRxPerformceTestRspPayload**

#include <zbhci_commom.h>

Public Members

uint16_t **u16DstAddr**

uint16_t **u16SendCnt**

uint16_t **u16AckCnt**

struct **ts_MsgNodesDevAnnceRspPayload**

#include <zbhci_commom.h>

Public Members

uint16_t **u16NwkAddr**

NWK address of the joined device.

uint64_t **u64IEEEAddr**

IEEE address of the joined device.

uint8_t **u8Capability**

Capability of the joined device.

struct **ts_MsgAfDataSendTestRspPayload**

#include <zbhci_commom.h>

Public Members

uint16_t **u16SrcAddr**

uint8_t **u8SrcEp**

uint8_t **u8DstEp**

uint16_t **u16ClusterId**

uint16_t **u16DataLen**

uint8_t **au8Payload**[256]

struct **ts_MsgLeaveIndicationPayload**

#include <zbhci_commom.h>

Public Members

uint64_t **u64MacAddr**

uint8_t **u8Rejoin**

struct **ts_AttrRead**

#include <zbhci_commom.h>

Public Members

uint16_t **u16AttrID**

uint8_t **u8Status**

uint8_t **u8DataType**

uint16_t **u16DataLen**

t_AttrData **uAttrData**

struct **ts_MsgZclAttrReadRspPayload**

#include <zbhci_commom.h>

Public Members

uint8_t **u8SeqNum**

uint16_t **u16SrcAddr**

uint8_t **u8SrcEp**

uint16_t **u16ClusterId**

uint8_t **u8AttrNum**

The number of attributes to be read.

ts_AttrRead **asAttrReadList**[8]

The list of the attributes to be read.

struct **ts_AttrWrite**

#include <zbhci_commom.h>

Public Members

uint8_t **u8Status**

uint16_t **u16AttrID**

struct **ts_MsgZclAttrWriteRspPayload**

#include <zbhci_commom.h>

Public Members

uint8_t **u8SeqNum**

uint16_t **u16SrcAddr**

uint8_t **u8SrcEp**

uint16_t **u16ClusterId**

uint8_t **u8AttrNum**

The number of attributes to be written.

ts_AttrWrite **asAttrWriteList**[256]

The list of the attributes to be written.

struct **ts_AttrConfigReport**

#include <zbhci_commom.h>

Public Members

uint8_t **u8Status**

uint8_t **u8ReportDirection**

uint16_t **u16AttrID**

struct **ts_MsgZclConfigReportRspPayload**

#include <zbhci_commom.h>

Public Members

uint8_t **u8SeqNum**

uint16_t **u16SrcAddr**

uint8_t **u8SrcEp**

uint16_t **u16ClusterId**

uint8_t **u8AttrNum**

TThe number of attributes' reporting to be configured.

ts_AttrConfigReport **asAttrConfigReportList**[16]

The list of the attributes to be configured.

struct **ts_AttrReadConfigReport**

#include <zbhci_commom.h>

Public Members

uint8_t **u8Status**

uint8_t **u8ReportDirection**

uint16_t **u16AttrID**

uint8_t **u8DataType**

uint16_t **u16MinRepInterval**

uint16_t **u16MaxRepInterval**

uint8_t **au8ReportableChange**[128]

uint16_t **u16TimeoutPeriod**

struct **ts_MsgZclReadReportCfgRspPayload**

#include <zbhci_commom.h>

Public Members

uint8_t **u8SeqNum**

uint16_t **u16SrcAddr**

The source address of the reporting message.

uint8_t **u8SrcEp**

The source endpoint of the reporting message.

uint16_t **u16ClusterId**

uint8_t **u8AttrNum**

The number of attributes' reporting to be read.

ts_AttrReadConfigReport **asAttrList**[8]

The list of the attributes to be read.

struct **ts_MsgZclReportMsgRcvPayload**

#include <zbhci_commom.h>

Public Members

uint8_t **u8SeqNum**

uint16_t **u16SrcAddr**

The source address of the reporting message.

uint8_t **u8SrcEp**

The source endpoint of the reporting message.

uint16_t **u16ClusterId**

uint8_t **u8AttrNum**

The number of attributes' reporting message to be received.

ts_AttrList **asAttrList**[4]

The list of the attributes' reporting message to be received.

struct **ts_MsgZclGroupAddRspPayload**

#include <zbhci_commom.h>

Public Members

uint8_t **u8Status**

The status field is set to SUCCESS, DUPLICATE_EXISTS, or INSUFFICIENT_SPACE as appropriate.

uint16_t **u16GroupId**

Group identifier.

struct **ts_MsgZclGroupViewRspPayload**

#include <zbhci_commom.h>

Public Members

uint8_t **u8Status**

The status field is set to SUCCESS, DUPLICATE_EXISTS, or INSUFFICIENT_SPACE as appropriate.

uint16_t **u16GroupId**

Group identifier.

uint8_t **u8GroupNameLength**

uint8_t **au8GroupName**[256]

Group name, character string.

struct **ts_MsgZclGroupGetMembershipRspPayload**

#include <zbhci_commom.h>

Public Members

uint8_t **u8Capability**

The remaining capability of the group table of the device.

uint8_t **u8GroupCount**

The number of groups contained in the group list field.

uint16_t **au16GroupId**[32]

The list of groupId in the group list field.

struct **ts_MsgZclGroupRemoveRspPayload**

#include <zbhci_commom.h>

Public Members

uint8_t **u8Status**

The status field is set to SUCCESS, DUPLICATE_EXISTS, or INSUFFICIENT_SPACE as appropriate.

uint16_t **u16GroupId**

Group identifier.

struct **ts_MsgZclIdentifyQueryRspPayload**

#include <zbhci_commom.h>

Public Members

uint16_t **u16ShortAddr**

The short address of the device.

uint8_t **u8SrcEp**

The source endpoint of the device.

uint16_t **u16Timeout**

The remaining time.

struct **ts_MsgZclOnOffCmdRcvPayload**

#include <zbhci_commom.h>

Public Members

uint8_t **u8SrcEp**

uint8_t **u8DstEp**

uint16_t **u16ClusterId**

uint8_t **u8CmdId**

struct **ts_MsgZclSceneAddRspPayload**

#include <zbhci_commom.h>

Public Members

uint8_t **u8Status**

SUCCESS, INSUFFICIENT_SPACE or INVALID_FIELD (the group is not present in the group table).

uint16_t **u16GroupId**

The group ID for which this scene applies.

uint8_t **u8SceneId**

The identifier, unique within this group, which is used to identify this scene.

struct **ts_MsgZclSceneViewRspPayload**

#include <zbhci_commom.h>

Public Members

uint8_t **u8Status**

SUCCESS, INSUFFICIENT_SPACE or INVALID_FIELD (the group is not present in the group table).

uint16_t **u16GroupId**

The group ID for which this scene applies.

uint8_t **u8SceneId**

The identifier, unique within this group, which is used to identify this scene.

uint16_t **u16TransTime**

Transition time copied from scene table entry.

uint8_t **u8SceneNameLength**

uint8_t **au8SceneName**[32]

Scene name copied from scene table entry. First byte is the length of the scene name.

uint8_t **extFieldLength**

uint8_t **au8ExtFieldSets**[32]

Extension field sets copied from scene table entry. First byte is the length of the extension field sets.

struct **ts_MsgZclSceneRemoveRspPayload**

#include <zbhci_commom.h>

Public Members

uint8_t **u8Status**

SUCCESS, INSUFFICIENT_SPACE or INVALID_FIELD (the group is not present in the group table).

uint16_t **u16GroupId**

The group ID for which this scene applies.

uint8_t **u8SceneId**

The identifier, unique within this group, which is used to identify this scene.

struct **ts_MsgZclSceneRemoveAllRspPayload**

#include <zbhci_commom.h>

Public Members

uint8_t **u8Status**

SUCCESS, INSUFFICIENT_SPACE or INVALID_FIELD (the group is not present in the group table).

uint16_t **u16GroupId**

The group ID for which this scene applies.

struct **ts_MsgZclSceneStoreRspPayload**

#include <zbhci_commom.h>

Public Members

uint8_t **u8Status**

SUCCESS, INSUFFICIENT_SPACE or INVALID_FIELD (the group is not present in the group table).

uint16_t **u16GroupId**

The group ID for which this scene applies.

uint8_t **u8SceneId**

The identifier, unique within this group, which is used to identify this scene.

struct **ts_MsgZclSceneGetMenbershipRspPayload**

#include <zbhci_commom.h>

Public Members

uint8_t **u8Status**

SUCCESS or INVALID_FIELD (the group is not present in the group table).

uint8_t **u8Capability**

Contain the remaining capacity of the scene table of the device.

uint16_t **u16GroupId**

The group ID for which this scene applies.

uint8_t **u8SceneCnt**

The number of scenes contained in the scene list field.

uint8_t **au8SceneList**[256]

Contain the identifiers of all the scenes in the scene table with the corresponding Group ID.

struct **ts_MsgDataConfirmPayload**

#include <zbhci_commom.h>

Public Members

uint8_t **u8SrcEndpoint**

uint8_t **u8Status**

uint8_t **u8ApsCnt**

struct **ts_MsgMacAddrIndPayload**

#include <zbhci_commom.h>

Public Members

uint64_t **u64DeviceExtAddr**

struct **ts_MsgNodeLeaveIndPayload**

#include <zbhci_commom.h>

Public Members

uint16_t **u16TotalCnt**

uint64_t **u64MacAddr**

struct **ts_HciMsg**

#include <zbhci_commom.h>

Public Members

uint16_t **u16MsgType**

uint16_t **u16MsgLength**

ts_MsgAckPayload **sAckPayload**

ts_MsgBdbCommissionFormationRspPayload **sBdbCommissionFormationRspPayload**

ts_MsgNetworkStateRspPayload **sNetworkStateRspPayloasd**

ts_MsgDiscoveryNwkAddrRspPayload **sDiscoveryNwkAddrRspPayload**

ts_MsgDiscoveryIEEEAddrRspPayload **sDiscoveryIEEEAddrRspPayload**

ts_MsgDiscoveryNodeDescRspPayload **sDiscoveryNodeDescRspPayload**

ts_MsgDiscoverySimpleDescRspPayload **sDiscoverySimpleDescRspPayload**

ts_MsgDiscoveryMatchDescRspPayload **sDiscoveryMatchDescRspPayload**

ts_MsgDiscoveryActiveEpRspPayload **sDiscoveryActiveEpRspPayload**

ts_MsgBindRspPayload **sBindRspPayload**

ts_MsgUnbindRspPayload **sUnbindRspPayload**

ts_NeighborTable **sghborTable**

ts_MsgMgmtLqiRspPayload **sMgmtLqiRspPayload**

ts_MsgMgmtBindRspPayload **sMgmtBindRspPayload**

ts_MsgMgmtLeaveRspPayload **sMgmtLeaveRspPayload**

ts_MsgMgmtPermitJoinRspPayload **sMgmtPermitJoinRspPayload**

ts_MsgNodesJoinedGetRspPayload **sNodesJoinedGetRspPayload**

ts_MsgTxRxPerformceTestRspPayload **sTxRxPerformceTestRspPayload**

ts_MsgNodesDevAnnceRspPayload **sNodesDevAnnceRspPayload**

ts_MsgAfDataSendTestRspPayload **sAfDataSendTestRspPayload**

ts_MsgLeaveIndicationPayload **sLeaveIndicationPayload**

ts_MsgZclAttrReadRspPayload **sZclAttrReadRspPayload**

ts_MsgZclAttrWriteRspPayload **sZclAttrWriteRspPayload**

ts_MsgZclConfigReportRspPayload **sZclConfigReportRspPayload**

ts_MsgZclReadReportCfgRspPayload **sZclReadReportCfgRspPayload**

ts_MsgZclReportMsgRcvPayload **sZclReportMsgRcvPayload**

ts_MsgZclGroupAddRspPayload **sZclGroupAddRspPayload**

ts_MsgZclGroupViewRspPayload **sZclGroupViewRspPayload**

ts_MsgZclGroupGetMembershipRspPayload **sZclGroupGetMembershipRspPayload**

ts_MsgZclGroupRemoveRspPayload **sZclGroupRemoveRspPayload**

ts_MsgZclIdentifyQueryRspPayload **sZclIdentifyQueryRspPayload**

ts_MsgZclOnOffCmdRcvPayload **sZclOnOffCmdRcvPayload**

ts_MsgZclSceneAddRspPayload **sZclSceneAddRspPayload**

ts_MsgZclSceneViewRspPayload **sZclSceneViewRspPayload**

ts_MsgZclSceneRemoveRspPayload **sZclSceneRemoveRspPayload**

ts_MsgZclSceneRemoveAllRspPayload **sZclSceneRemoveAllRspPayload**

ts_MsgZclSceneStoreRspPayload **sZclSceneStoreRspPayload**

ts_MsgZclSceneGetMenbershipRspPayload **sZclSceneGetMenbershipRspPayload**

ts_MsgDataConfirmPayload **sDataConfirmPayload**

ts_MsgMacAddrIndPayload **sMacAddrIndPayload**

ts_MsgNodeLeaveIndPayload **sNodeLeaveIndPayload**

union *ts_HciMsg*.[anonymous] **uPayload**

CHAPTER

FOUR

DEVICES

Currently 2 devices are supported from 1 different vendors. In case you own a Zigbee device which is NOT listed here, please see How to support new devices.

4.1 Xiaom

4.1.1 Xiaomi RTCGQ11LM

Model	RTCGQ11LM
Vendor	Xiaomi
Description	Aqara human body movement and illuminance sensor
Exposes	occupancy, illuminance_lux
Picture	

Pairing

Press and hold the reset button on the device for +- 5 seconds (until the blue light starts blinking). After this the device will automatically join. If this doesn't work, try with a single short button press.

4.1.2 Xiaomi WSDCGQ11LM

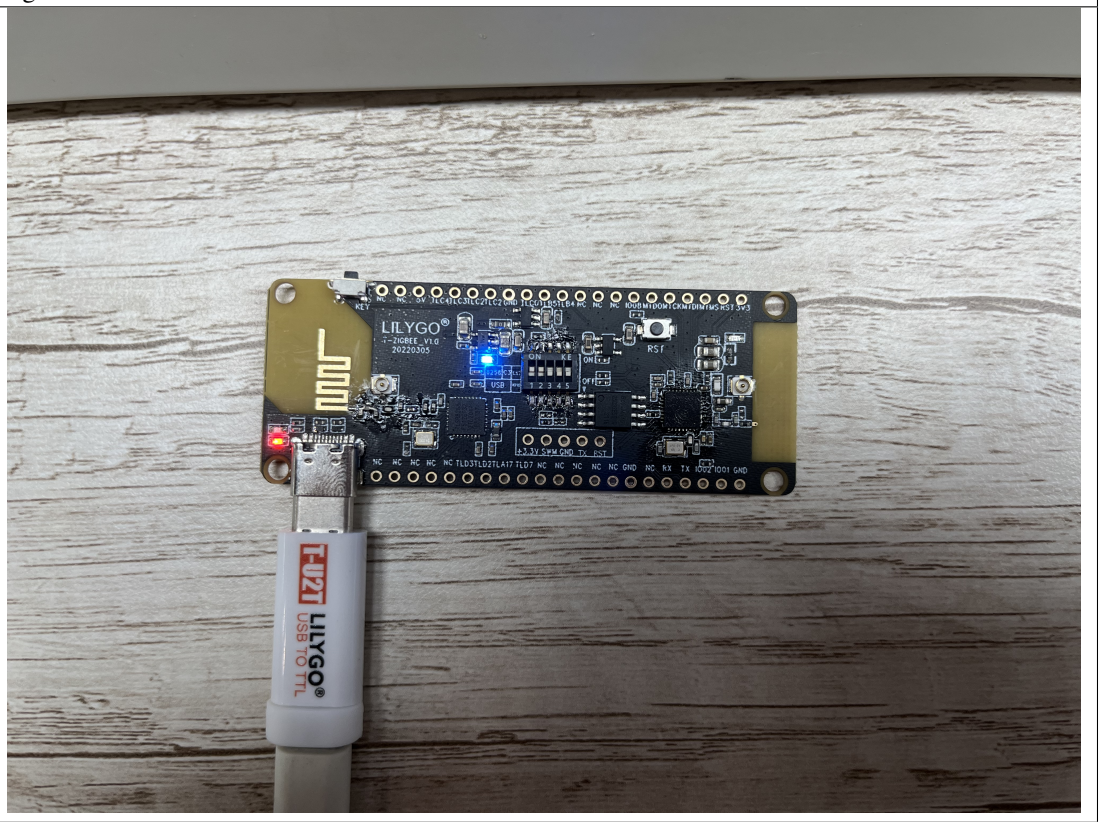
Mode	WSDCGQ11LM
Vendor	Xiaomi
Description	Aqara temperature, humidity and pressure sensor
Exposes	temperature, humidity, pressure
Picture	

Pairing

Press and hold the reset button on the device for +- 5 seconds (until the blue light starts blinking). After this the device will automatically join. If this doesn't work, try with a single short button press.

4.2 LILYGO

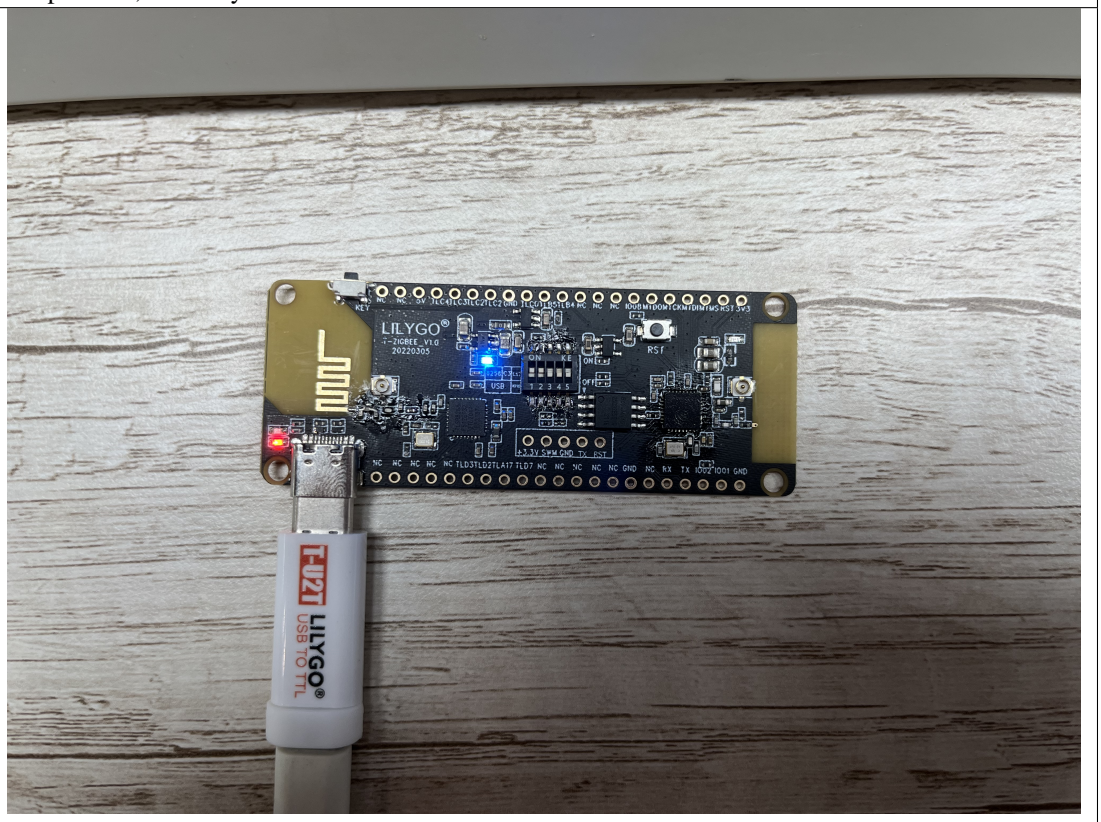
4.2.1 LILYGO light_demo

Model	light_demo
Vendor	LILYGO
Description	a simple lamp
Exposes	light
Picture	

Pairing

Press and hold the button on the top of the node device for 3 seconds to start pairing. The green light is always on, indicating that the pairing is complete.

4.2.2 LILYGO sensor_demo

Model	sensor_demo
Vendor	LILYGO
Description	A simple temperature and humidity sensor
Exposes	temperature, humidity
Picture	

Pairing

Press and hold the button on the top of the node device for 3 seconds to start pairing. The green light is always on, indicating that the pairing is complete.

Thank

The source code of this device is provided by swkim01.

INDEX

T

Z