# Module Gateway H2 Arduino 使用チュートリアル

本チュートリアルでは、Module Gateway H2 を使用して Zigbee および Thread Arduino サンプルプログラムを実行し、ネットワーク通信を実現する方法を紹介します。

**ボードマネージャーバージョン要件**

Zigbee および Thread の機能は、新しい ESP32 Arduino ボードマネージャーバージョンの使用と、いくつかのパーティションテーブル設定操作が必要です。本チュートリアルは v3.1.1 バージョンに基づいて実装されます。以下のインストールチュートリアルを参照して設定してください。

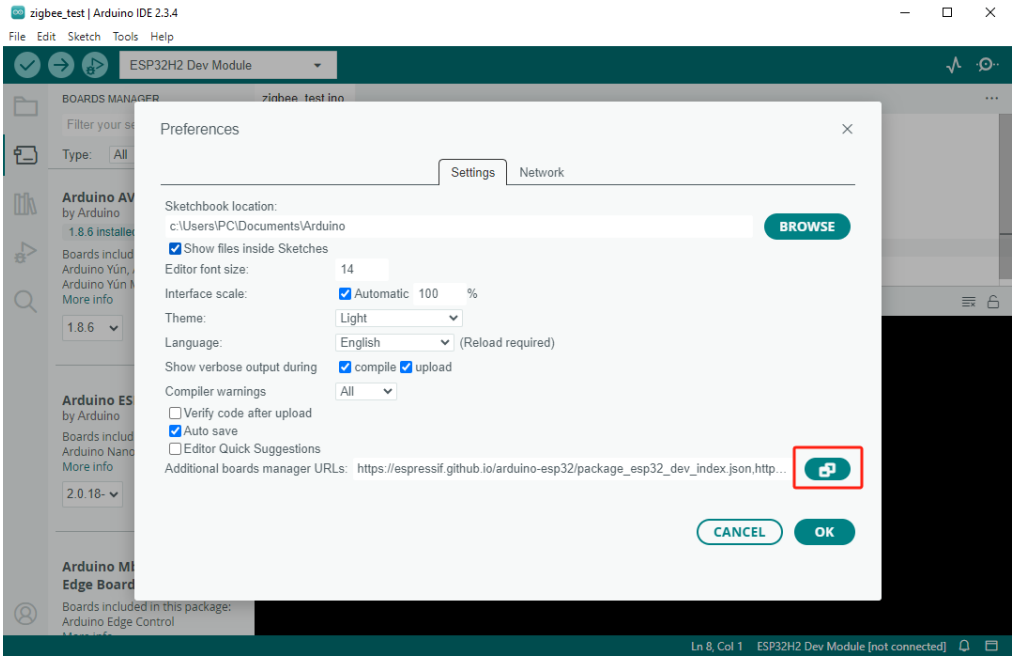
## 1. 準備作業

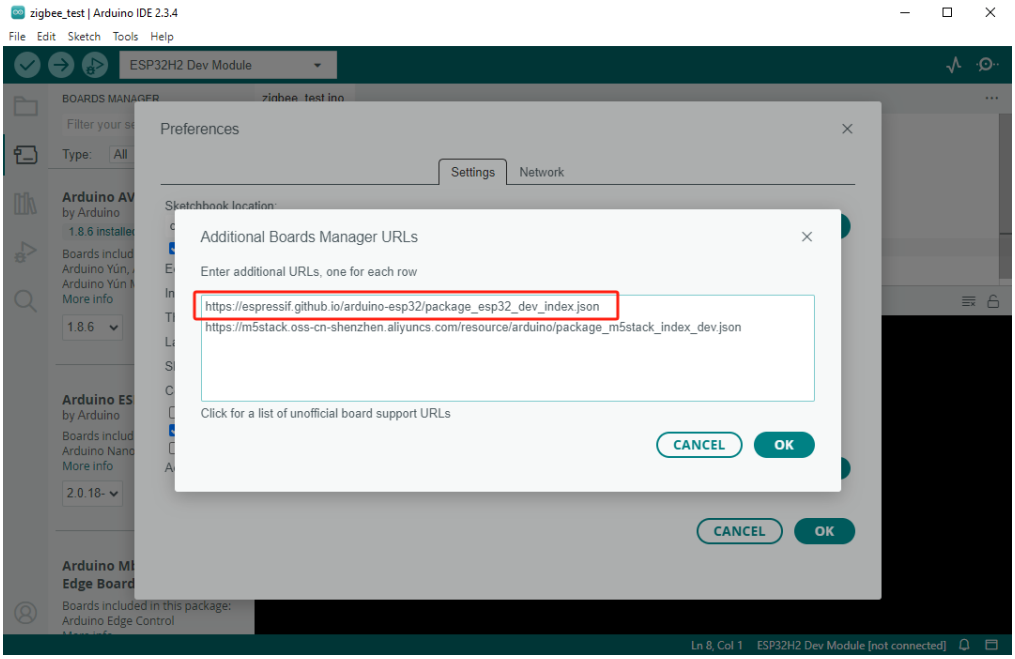
- 1.**環境設定:** Arduino IDE 入門チュートリアル を参照して IDE のインストールを完了してください。

  設定画面の「追加のボードマネージャー URL」の入力欄を探し、以下の URL を追加してください:

```
https://espressif.github.io/arduino-esp32/package_esp32_dev_index.json
```

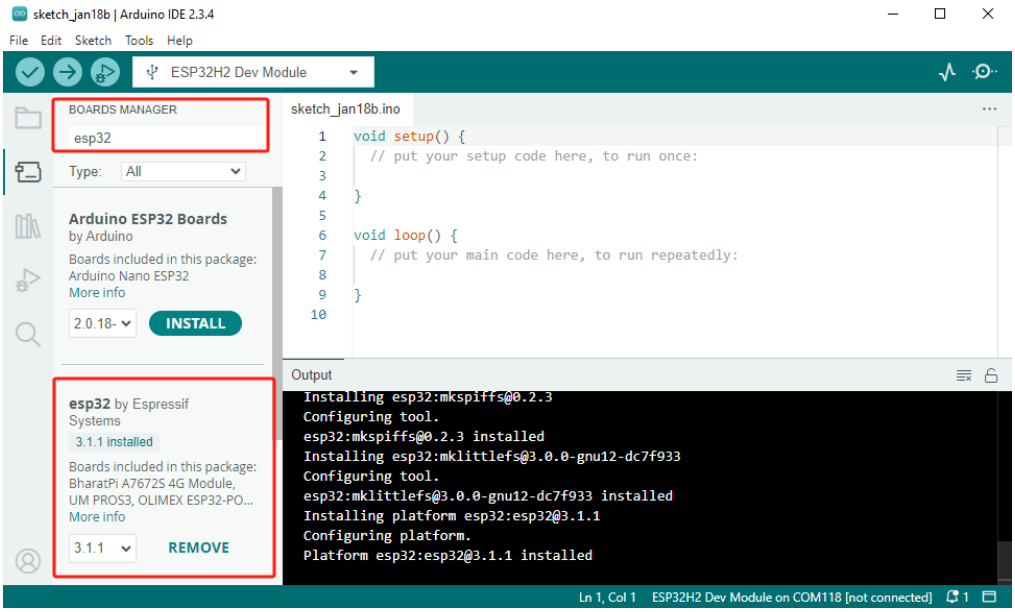

zigbee_test | Arduino IDE 2.3.4
File Edit Sketch Tools Help
New Sketch Ctrl+N
New Cloud Sketch Alt+Ctrl+N
Open... Ctrl+O
Open Recent
Sketchbook
Examples
Close Ctrl+W
Save Ctrl+S
Save As... Ctrl+Shift+S
Preferences... Ctrl+Comma
Advanced
Quit Ctrl+Q
zigbee_test.ino
void setup() {
// put your setup code here, to run once:
}
void loop() {
// put your main code here, to run repeatedly:
}
Output
1.8.6
REMOVE
Arduino ESP32 Boards


zigbee_test | Arduino IDE 2.3.4
File Edit Sketch Tools Help
ESP32H2 Dev Module
Preferences
Settings
Network
Sketchbook location:
c:\Users\PC\Documents\Arduino
BROWSE
Show files inside Sketches
Editor font size: 14
Interface scale: Automatic 100 %
Theme: Light
Language: English (Reload required)
Show verbose output during compile upload
Compiler warnings All
Verify code after upload
Auto save
Editor Quick Suggestions
Additional boards manager URLs: https://espressif.github.io/arduino-esp32/package_esp32_dev_index.json,http...
CANCEL
OK
Ln 8, Col 1 ESP32H2 Dev Module [not connected]

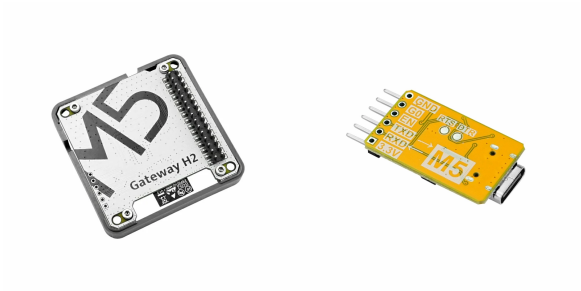

zigbee_test | Arduino IDE 2.3.4
File Edit Sketch Tools Help
ESP32H2 Dev Module
Preferences
Settings
Network
Additional Boards Manager URLs
Enter additional URLs, one for each row
https://espressif.github.io/arduino-esp32/package_esp32_dev_index.json
https://m5stack.oss-cn-shenzhen.aliyuncs.com/resource/arduino/package_m5stack_index_dev.json
Click for a list of unofficial board support URLs
CANCEL
OK
CANCEL
OK
Ln 8, Col 1 ESP32H2 Dev Module [not connected]

ボードマネージャーで　**ESP32**　を検索し、ボードのインストールを完了してください。

**注:**　この手順では多数のツールチェーンをダウンロードする必要があり、ダウンロードに失敗する場合は、ネットワーク環境を変更するかプロキシの設定を試みてください。

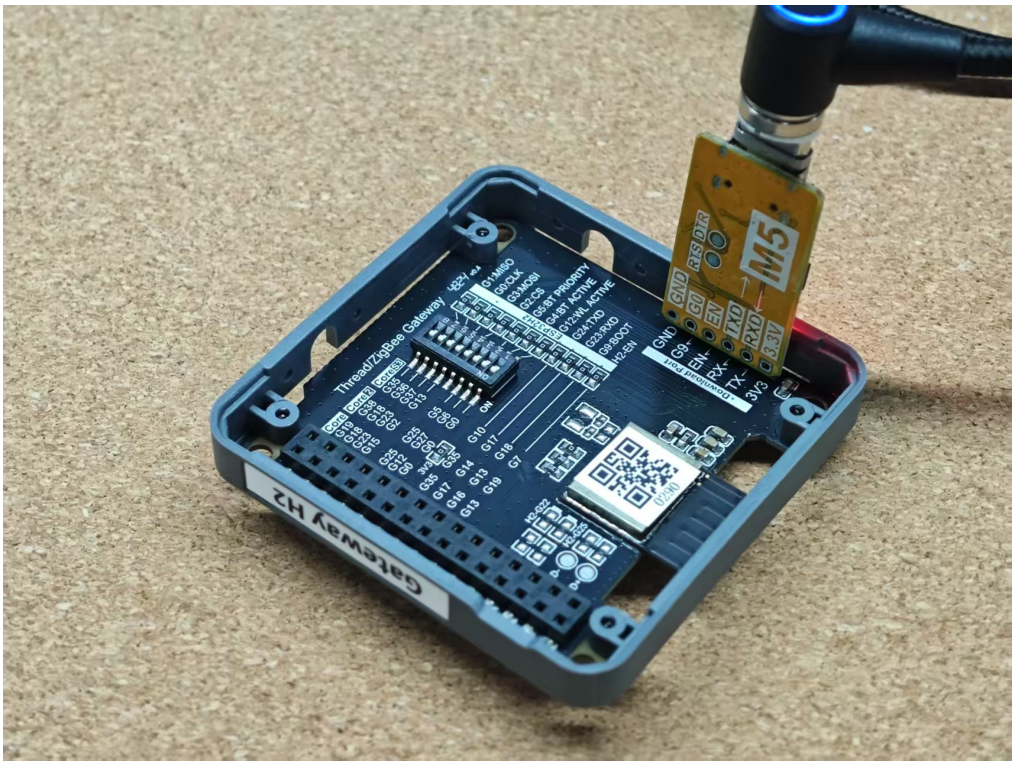



- 2.**使用するドライバライブラリ:**
  - arduino-esp32@3.1.1
- 3.**使用するハードウェア製品:**
  - ESP32 Downloader
  - Module Gateway H2




**Module Gateway H2**

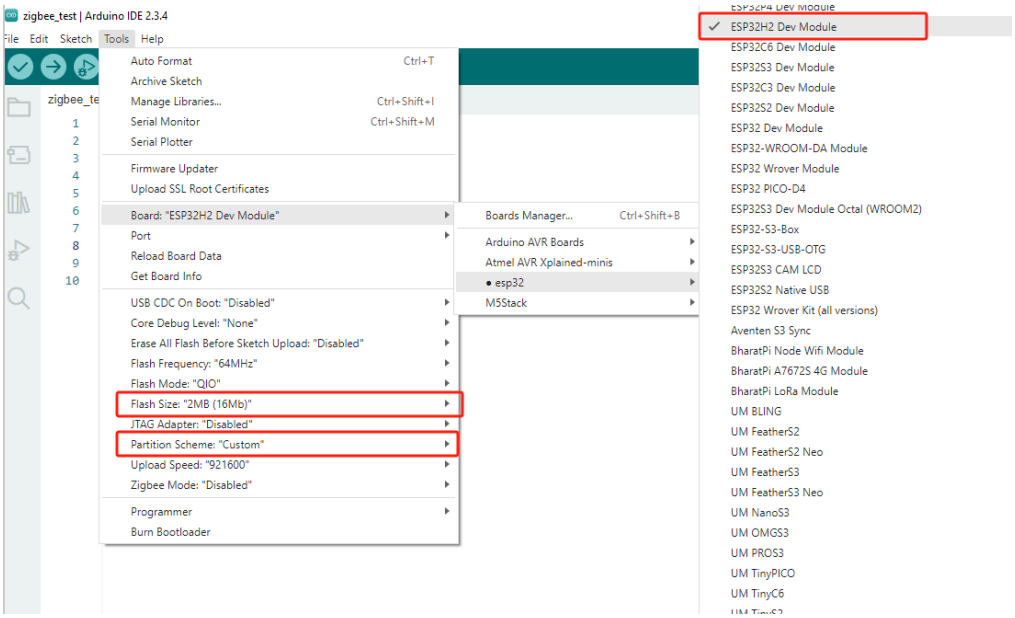
本チュートリアルでは、`Module Gateway H2` と `ESP32 Downloader` を使用してデモを行います。Module Gateway H2 のコアモジュールは `ESP32-H2-MINI-1-N2` を採用しており、独立して動作する能力を持っています。本チュートリアルでは、ESP32 Downloader を介して直接サンプルプログラムを Module Gateway H2 に書き込み、独立したデバイスとして使用します。

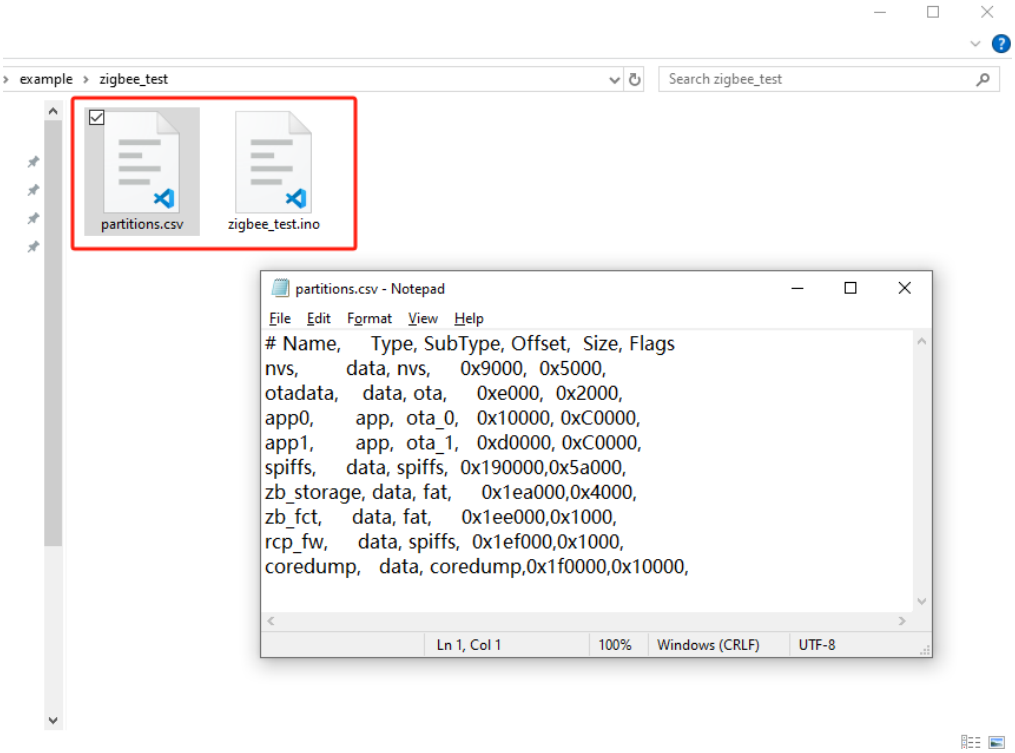



## 2. カスタムパーティションテーブル

**Partitions**

Module Gateway H2 のコアモジュールは 2MB フラッシュ版の `ESP32-H2-MINI-1-N2` を使用しているため、プログラムをコンパイルする前に使用するパーティションテーブルを調整する必要があります。
デフォルトのパーティションテーブルオプションには 2MB 版の設定が提供されていないため、`custom` オプションを使用します。このオプションを有効にする際は、カスタムパーティションテーブルファイルを用意し、プロジェクトファイル (.ino) と同じディレクトリに配置し、`partitions.csv` と命名してください。以下の各サンプルプログラムでは異なるパーティションテーブルが使用される場合があるので、各サンプルの具体的な説明を参照してください。



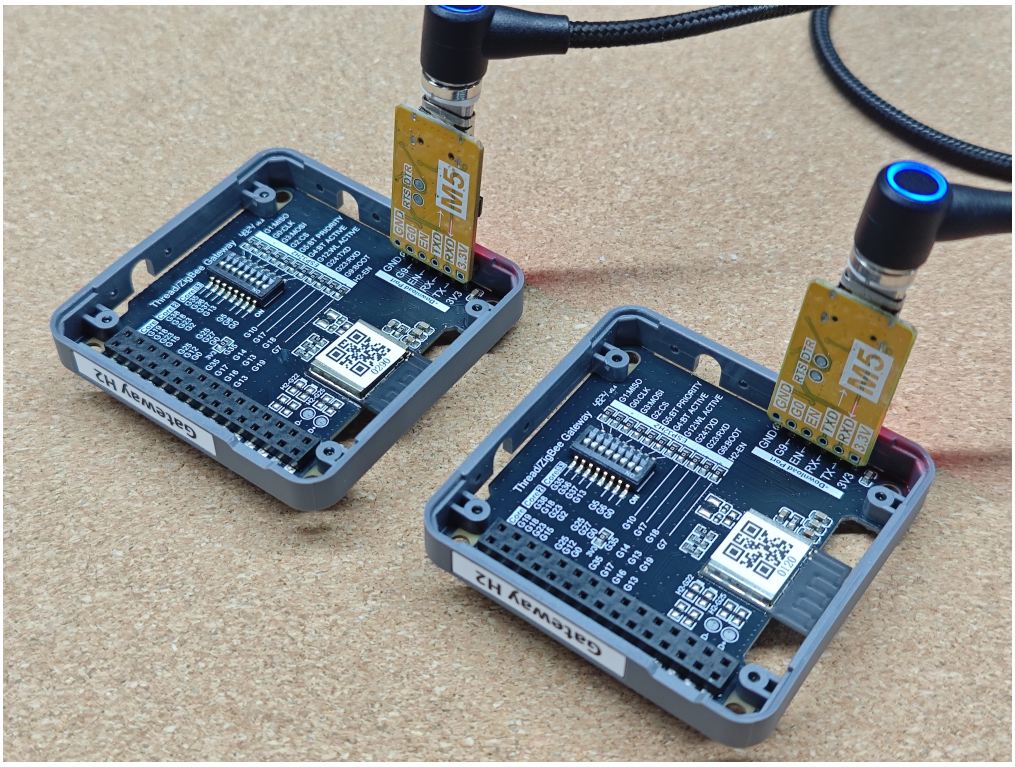

# 3. Zigbee

**Coordinator & End Device**

本サンプルでは、2 台の Module Gateway H2 を使用し、それぞれに Zigbee OnOff Switch (Coordinator) と Zigbee OnOff Light (End Device) のプログラムを書き込みます。

コーディネーター (Coordinator) は起動後、自動的にネットワークを作成し、デバイスの参加を待ち、デバイスが参加してバインディングが完了すると、間隔をおいてライトのオン / オフ指令を送信します。

エンドデバイス (End Device) は起動後、Coordinator からの制御指令を受け取り、現在の LED 状態を表示します。




## Zigbee OnOff Light (End Device)

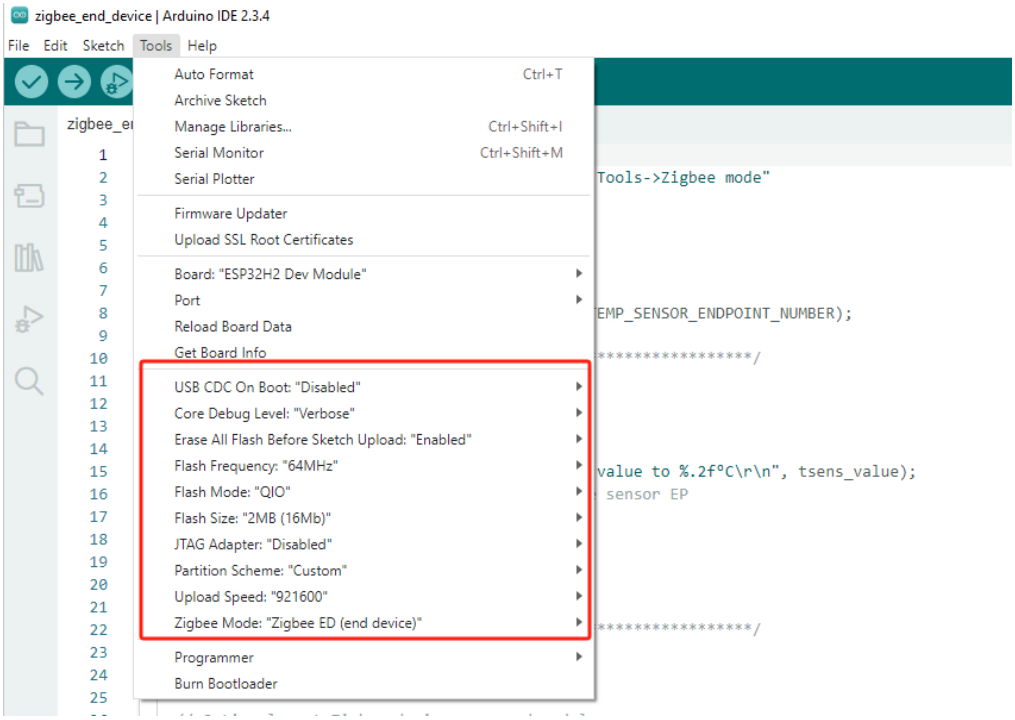
**Arduino IDE ツールメニューの設定:**

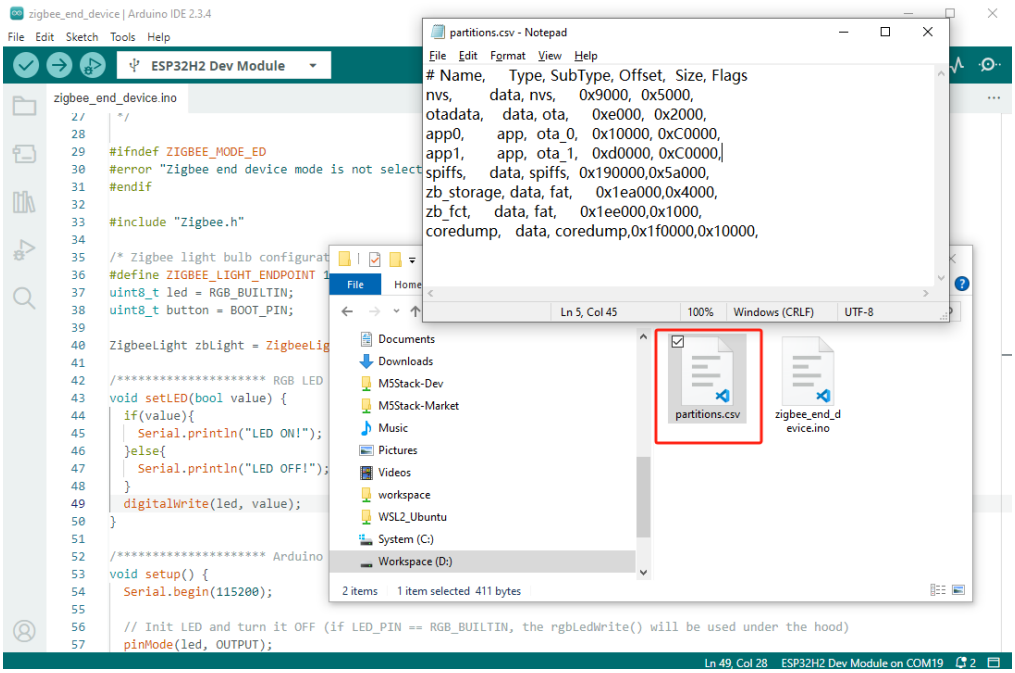
- 正しいボードを選択: `Tools -> Board: ESP32H2 Dev Module`
- フラッシュ消去を有効にする: `Tools -> Erase All Flash Before Sketch Upload: Enable` (無効にすると接続に失敗する可能性があります)
- フラッシュサイズを選択: `Tools -> Flash Size: 2MB`
- エンドデバイスモードを選択: `Tools -> Zigbee mode: Zigbee ED (end device)`
- Zigbee パーティションスキームを選択: `Tools -> Partition Scheme: custom`

> **Zigbee End Device パーティションテーブル設定**
>
> 以下の Zigbee サンプルプログラムはこのパーティションテーブル設定を使用してコンパイルされます。プログラムをコンパイルする前に必ず `custom` パーティションテーブルオプションを有効にし、以下のパーティションテーブルをコピーして `partitions.csv` という名前でプロジェクトファイル (.ino) と同じディレクトリに保存してください。

- Zigbee 2MB with spiffs

```
# Name,     Type, SubType, Offset,  Size, Flags
nvs,        data, nvs,     0x9000,  0x5000,
otadata,    data, ota,     0xe000,  0x2000,
app0,       app,  ota_0,   0x10000, 0xC0000,
app1,       app,  ota_1,   0xd0000, 0xC0000,
spiffs,     data, spiffs,  0x190000,0x5a000,
zb_storage, data, fat,     0x1ea000,0x4000,
zb_fct,     data, fat,     0x1ee000,0x1000,
coredump,   data, coredump,0x1f0000,0x10000,
```


zigbee_end_device | Arduino IDE 2.3.4
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter
Firmware Updater
Upload SSL Root Certificates
Board: "ESP32H2 Dev Module"
Port
Reload Board Data
Get Board Info
USB CDC On Boot: "Disabled"
Core Debug Level: "Verbose"
Erase All Flash Before Sketch Upload: "Enabled"
Flash Frequency: "64MHz"
Flash Mode: "QIO"
Flash Size: "2MB (16Mb)"
JTAG Adapter: "Disabled"
Partition Scheme: "Custom"
Upload Speed: "921600"
Zigbee Mode: "Zigbee ED (end device)"
Programmer
Burn Bootloader
Tools->Zigbee mode"
EMP_SENSOR_ENDPOINT_NUMBER);
*****************/
value to %.2f°C\r\n", tsens_value);
sensor EP
*****************/


zigbee_end_device | Arduino IDE 2.3.4
ESP32H2 Dev Module
zigbee_end_device.ino
partitions.csv - Notepad
File Edit Format View Help
# Name, Type, SubType, Offset, Size, Flags
nvs, data, nvs, 0x9000, 0x5000,
otadata, data, ota, 0xe000, 0x2000,
app0, app, ota_0, 0x10000, 0xC0000,
app1, app, ota_1, 0xd0000, 0xC0000,
spiffs, data, spiffs, 0x190000,0x5a000,
zb_storage, data, fat, 0x1ea000,0x4000,
zb_fct, data, fat, 0x1ee000,0x1000,
coredump, data, coredump,0x1f0000,0x10000,
Ln 5, Col 45 100% Windows (CRLF) UTF-8
Documents
Downloads
M5Stack-Dev
M5Stack-Market
Music
Pictures
Videos
workspace
WSL2_Ubuntu
System (C:)
Workspace (D:)
partitions.csv
zigbee_end_device.ino
2 items 1 item selected 411 bytes
Ln 49, Col 28 ESP32H2 Dev Module on COM19

cpp

```cpp
#ifndef ZIGBEE_MODE_ED
#error "Tools->Zigbee mode で Zigbee エンドデバイスモードが選択されていません"
#endif

#include "Zigbee.h"

/* Zigbee 電球の設定 */
#define ZIGBEE_LIGHT_ENDPOINT 10
uint8_t led = RGB_BUILTIN;
uint8_t button = BOOT_PIN;

ZigbeeLight zbLight = ZigbeeLight(ZIGBEE_LIGHT_ENDPOINT);

/********************* RGB LED 関数 **************************/
void setLED(bool value) {
  if(value){
    Serial.println("LED ON!");
  } else {
    Serial.println("LED OFF!");
  }
  digitalWrite(led, value);
}

/********************* Arduino 関数 **************************/
void setup() {
  Serial.begin(115200);

  // LED の初期化と消灯（もし LED_PIN が RGB_BUILTIN なら、内部で rgbLedWrite() が使用されます）
  pinMode(led, OUTPUT);
  digitalWrite(led, LOW);

  // ファクトリーリセット用ボタンの初期化
  pinMode(button, INPUT_PULLUP);

  // オプション: Zigbee デバイスの名前とモデルを設定
  zbLight.setManufacturerAndModel("Espressif", "ZBLightBulb");

  // ライト変更時のコールバック関数を設定
  zbLight.onLightChange(setLED);

  // Zigbee Core にエンドポイントを追加
  Serial.println("Zigbee Core に ZigbeeLight エンドポイントを追加中");
  Zigbee.addEndpoint(&zbLight);

  // すべてのエンドポイントが登録されたら、Zigbee を開始。デフォルトでは ZIGBEE_END_DEVICE として動作します
  if (!Zigbee.begin()) {
    Serial.println("Zigbee の起動に失敗しました!");
    Serial.println("再起動します...");
    ESP.restart();
  }
```

```
  Serial.println("ネットワークに接続中");
  while (!Zigbee.connected()) {
    Serial.print(".");
    delay(100);
  }
  Serial.println();
}

void loop() {
  // ファクトリーリセット用ボタンのチェック
  if (digitalRead(button) == LOW) {  // ボタンが押された場合
    // チャタリング防止処理
    delay(100);
    int startTime = millis();
    while (digitalRead(button) == LOW) {
      delay(50);
      if ((millis() - startTime) > 3000) {
        // 3 秒以上ボタンが押された場合、Zigbee を工場出荷状態にリセットし再起動
        Serial.println("Zigbee を工場出荷状態にリセットし、1 秒後に再起動します。");
        delay(1000);
        Zigbee.factoryReset();
      }
    }
    // ボタン押下でライトの状態を切り替え
    zbLight.setLight(!zbLight.getLightState());
  }
  delay(100);
}
```

## Zigbee OnOff Switch (Coordinator)

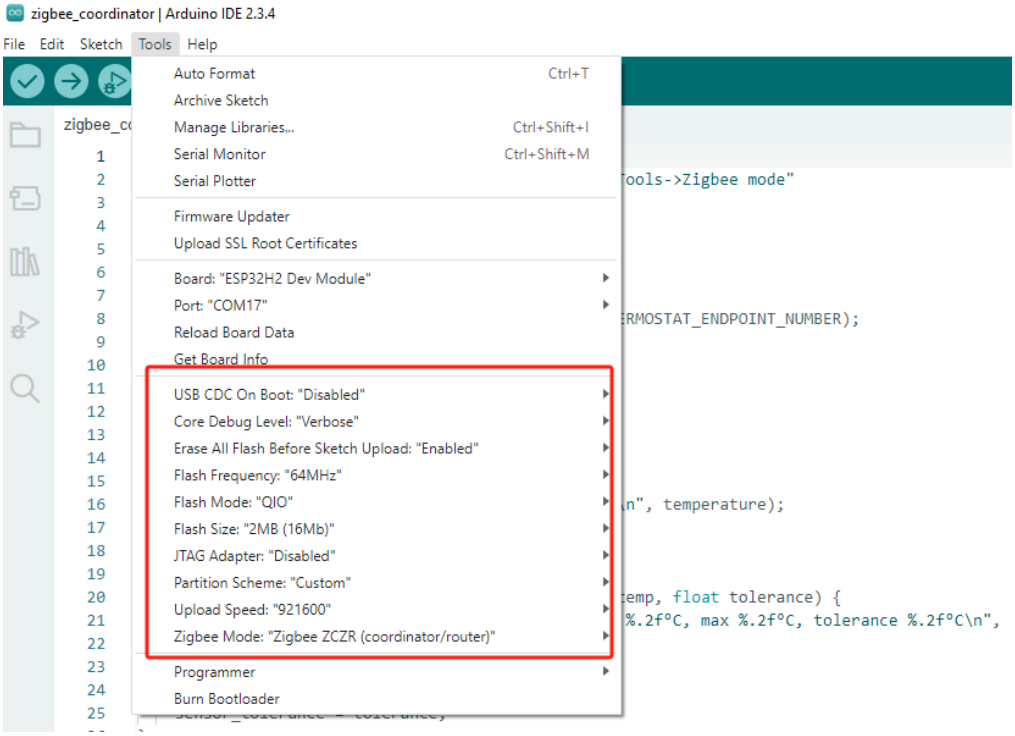
**Arduino IDE ツールメニューの設定:**

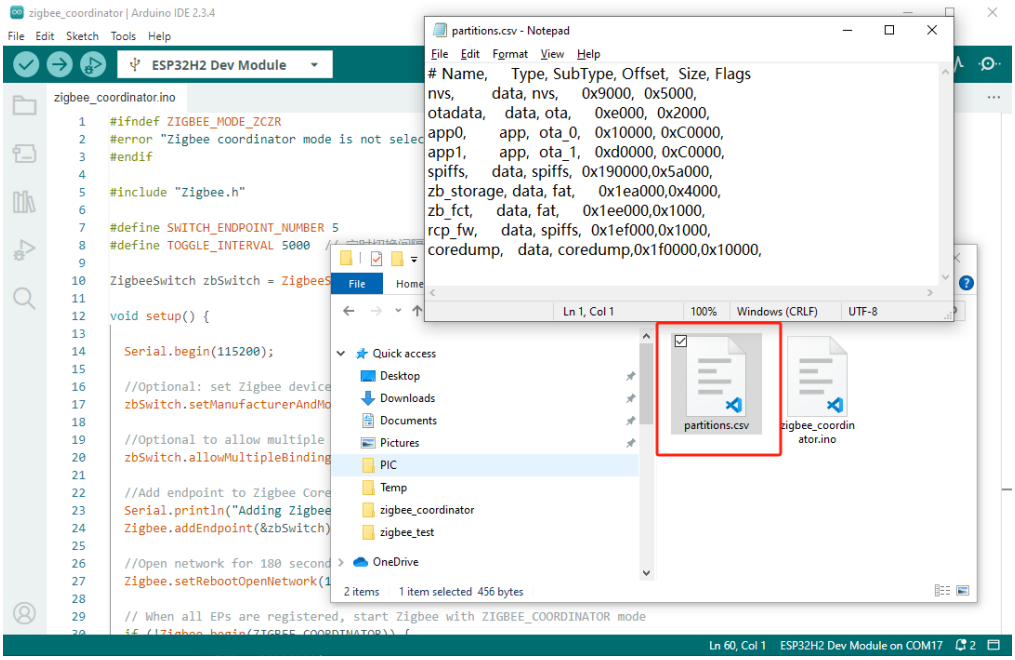
- 正しいボードを選択: `Tools -> Board: ESP32H2 Dev Module`
- フラッシュ消去を有効にする: `Tools -> Erase All Flash Before Sketch Upload: Enable` (無効にすると接続に失敗する可能性があります)
- フラッシュサイズを選択: `Tools -> Flash Size: 2MB`
- コーディネーターモードを選択: `Tools -> Zigbee mode: Zigbee ZCZR (coordinator/router)`
- Zigbee パーティションスキームを選択: `Tools -> Partition Scheme: custom`

> **Zigbee Coordinator パーティションテーブル設定**
>
> 以下の Zigbee サンプルプログラムはこのパーティションテーブル設定を使用してコンパイルされます。プログラムをコンパイルする前に必ず `custom` パーティションテーブルオプションを有効にし、以下のパーティションテーブルをコピーして `partitions.csv` という名前でプロジェクトファイル (.ino) と同じディレクトリに保存してください。

- Zigbee ZCZR 2MB with spiffs

```
# Name,     Type, SubType, Offset,  Size, Flags
nvs,        data, nvs,     0x9000,  0x5000,
otadata,    data, ota,     0xe000,  0x2000,
app0,       app,  ota_0,   0x10000, 0xC0000,
app1,       app,  ota_1,   0xd0000, 0xC0000,
spiffs,     data, spiffs,  0x190000,0x5a000,
zb_storage, data, fat,     0x1ea000,0x4000,
zb_fct,     data, fat,     0x1ee000,0x1000,
rcp_fw,     data, spiffs,  0x1ef000,0x1000,
coredump,   data, coredump,0x1f0000,0x10000,
```



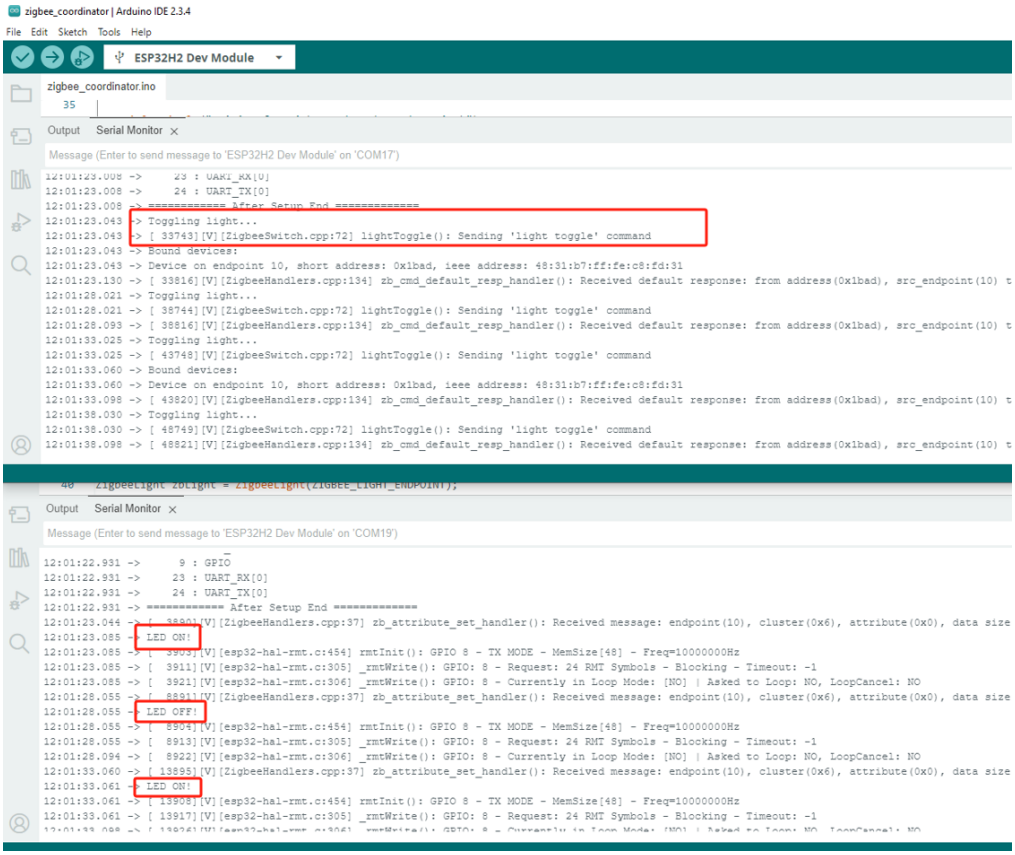
cpp

```cpp
#ifndef ZIGBEE_MODE_ZCZR
#error "Tools->Zigbee mode で Zigbee コーディネーターモードが選択されていません"
#endif

#include "Zigbee.h"

#define SWITCH_ENDPOINT_NUMBER 5
#define TOGGLE_INTERVAL 5000  // 切り替え間隔（ミリ秒）

ZigbeeSwitch zbSwitch = ZigbeeSwitch(SWITCH_ENDPOINT_NUMBER);

void setup() {

  Serial.begin(115200);

  // オプション: Zigbee デバイスの名前とモデルを設定
  zbSwitch.setManufacturerAndModel("Espressif", "ZigbeeSwitch");

  // オプション: 複数のライトがスイッチにバインドできるようにする
  zbSwitch.allowMultipleBinding(true);

  // Zigbee Core にエンドポイントを追加
  Serial.println("Zigbee Core に ZigbeeSwitch エンドポイントを追加中");
  Zigbee.addEndpoint(&zbSwitch);

  // 起動後 180 秒間、ネットワークをオープンにする
  Zigbee.setRebootOpenNetwork(180);

  // すべてのエンドポイントが登録されたら、ZIGBEE_COORDINATOR モードで Zigbee を開始
  if (!Zigbee.begin(ZIGBEE_COORDINATOR)) {
    Serial.println("Zigbee の起動に失敗しました!");
    Serial.println("再起動します...");
    ESP.restart();
  }

  Serial.println("スイッチにバインドされるライトを待機中");
  // スイッチがライトにバインドされるのを待つ:
  while (!zbSwitch.bound()) {
    Serial.printf(".");
    delay(500);
  }

  // オプション: バインドされたデバイス一覧を表示し、メーカーとモデル名を読み取る
  std::list<zb_device_params_t *> boundLights = zbSwitch.getBoundDevices();
  for (const auto &device : boundLights) {
    Serial.printf("エンドポイント %d のデバイス、ショートアドレス: 0x%x\r\n", device->endpoint, device->shor
    Serial.printf(
      "IEEE アドレス: %02X:%02X:%02X:%02X:%02X:%02X:%02X:%02X\r\n", device->ieee_addr[7], device->ieee
      device->ieee_addr[3], device->ieee_addr[2], device->ieee_addr[1], device->ieee_addr[0]
    );
```

```
    char *manufacturer = zbSwitch.readManufacturer(device->endpoint, device->short_addr, device->iee
    char *model = zbSwitch.readModel(device->endpoint, device->short_addr, device->ieee_addr);
    if (manufacturer != nullptr) {
      Serial.printf("ライトのメーカー: %s\r\n", manufacturer);
    }
    if (model != nullptr) {
      Serial.printf("ライトのモデル: %s\r\n", model);
    }
  }

  Serial.println();

}

void loop() {
  static unsigned long lastToggleTime = 0;

  // 一定間隔でライトの状態を切り替える
  if (millis() - lastToggleTime > TOGGLE_INTERVAL) {
    lastToggleTime = millis();
    Serial.println("ライトの状態を切り替えます...");
    zbSwitch.lightToggle();
  }

  // 10 秒ごとにバインドされたライトを表示
  static uint32_t lastPrint = 0;
  if (millis() - lastPrint > 10000) {
    lastPrint = millis();
    zbSwitch.printBoundDevices(Serial);
  }
}
```

## 使用手順

- 1. コーディネーターが既に動作しネットワークを作成していることを確認し、OnOff Light のコードをエンドデバイスに書き込みます。
- 2. デバイス起動後、自動的にネットワークを検索して参加し、OnOff Switch は定期的にライトの切り替え指令を送信します。




# 4. Zigbee ネットワークスキャン

**Arduino IDE ツールメニューの設定:**

- 正しいボードを選択: `Tools -> Board: ESP32H2 Dev Module`
- フラッシュ消去を有効にする: `Tools -> Erase All Flash Before Sketch Upload: Enable` (無効にすると接続に失敗する可能性があります)
- フラッシュサイズを選択: `Tools -> Flash Size: 2MB`
- コーディネーターモードを選択: `Tools -> Zigbee mode: Zigbee ZCZR (coordinator/router)`
- Zigbee パーティションスキームを選択: `Tools -> Partition Scheme: custom`

> **Zigbee サンプルパーティションテーブル設定**
>
> 以下の Zigbee サンプルプログラムはこのパーティションテーブル設定を使用してコンパイルされます。プログラムをコンパイルする前に必ず `custom` パーティションテーブルオプションを有効にし、以下のパーティションテーブルを `partitions.csv` という名前でプロジェクトファイル (.ino) と同じディレクトリに保存してください。

- Zigbee ZCZR 2MB with spiffs

```
# Name,     Type, SubType, Offset,  Size, Flags
nvs,        data, nvs,     0x9000,  0x5000,
otadata,    data, ota,     0xe000,  0x2000,
app0,       app,  ota_0,   0x10000, 0xC0000,
app1,       app,  ota_1,   0xd0000, 0xC0000,
spiffs,     data, spiffs,  0x190000,0x5a000,
zb_storage, data, fat,     0x1ea000,0x4000,
zb_fct,     data, fat,     0x1ee000,0x1000,
rcp_fw,     data, spiffs,  0x1ef000,0x1000,
coredump,   data, coredump,0x1f0000,0x10000,
```

**Zigbee ネットワークスキャン**

本サンプルでは、Module Gateway H2 を使用して Zigbee ネットワークのスキャンを実行し、ネットワーク情報をシリアルポートに出力します。

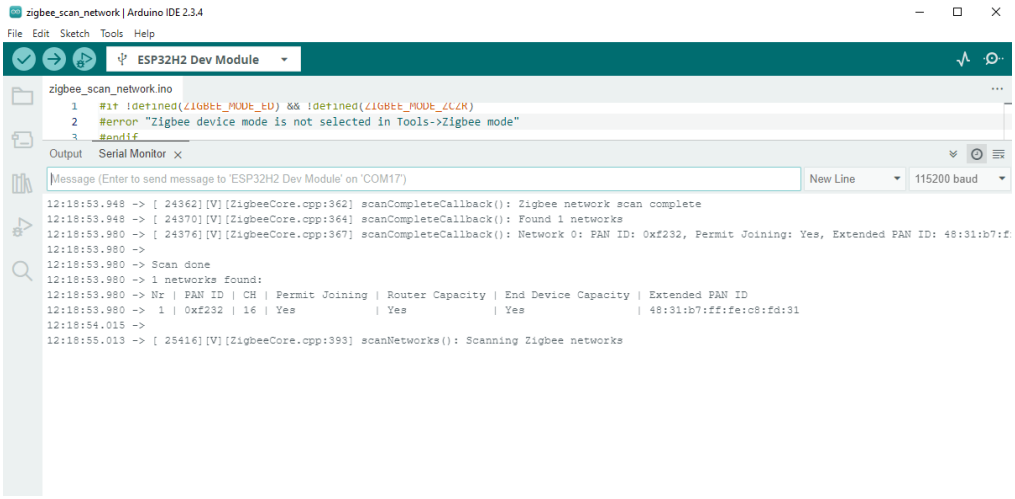
cpp

```cpp
#if !defined(ZIGBEE_MODE_ED) && !defined(ZIGBEE_MODE_ZCZR)
#error "Tools->Zigbee mode で Zigbee デバイスモードが選択されていません"
#endif

#include "Zigbee.h"

#ifdef ZIGBEE_MODE_ZCZR
zigbee_role_t role = ZIGBEE_ROUTER;  // または ZIGBEE_COORDINATOR に設定可能ですが、自身はスキャンしません
#else
zigbee_role_t role = ZIGBEE_END_DEVICE;
#endif

void printScannedNetworks(uint16_t networksFound) {
    if (networksFound == 0) {
        Serial.println("ネットワークが見つかりませんでした");
    } else {
        zigbee_scan_result_t *scan_result = Zigbee.getScanResult();
        Serial.println("\nスキャン完了");
        Serial.print(networksFound);
        Serial.println(" 件のネットワークが見つかりました:");
        Serial.println("Nr | PAN ID | CH | Permit Joining | Router Capacity | End Device Capacity | I
        for (int i = 0; i < networksFound; ++i) {
        // 各ネットワークの全情報を出力
        Serial.printf("%2d", i + 1);
        Serial.print(" | ");
        Serial.printf("0x%04hx", scan_result[i].short_pan_id);
        Serial.print(" | ");
        Serial.printf("%2d", scan_result[i].logic_channel);
        Serial.print(" | ");
        Serial.printf("%-14.14s", scan_result[i].permit_joining ? "Yes" : "No");
        Serial.print(" | ");
        Serial.printf("%-15.15s", scan_result[i].router_capacity ? "Yes" : "No");
        Serial.print(" | ");
        Serial.printf("%-19.19s", scan_result[i].end_device_capacity ? "Yes" : "No");
        Serial.print(" | ");
        Serial.printf(
            "%02x:%02x:%02x:%02x:%02x:%02x:%02x:%02x",
            scan_result[i].extended_pan_id[7],
            scan_result[i].extended_pan_id[6],
            scan_result[i].extended_pan_id[5],
            scan_result[i].extended_pan_id[4],
            scan_result[i].extended_pan_id[3],
            scan_result[i].extended_pan_id[2],
            scan_result[i].extended_pan_id[1],
            scan_result[i].extended_pan_id[0]
        );
        Serial.println();
        delay(10);
        }
        Serial.println("");
```

```
        // 以下のコードのためにメモリを解放するため、スキャン結果を削除
        Zigbee.scanDelete();
    }
}

void setup() {
    Serial.begin(115200);

    // スキャン専用に、エンドポイントなしで Zigbee スタックを初期化
    if (!Zigbee.begin(role)) {
        Serial.println("Zigbee の起動に失敗しました!");
        Serial.println("再起動します...");
        ESP.restart();
    }

    Serial.println("セットアップ完了、Zigbee ネットワークスキャンを開始します...");
    // デフォルトパラメータ（全チャンネル、スキャン時間 5）で Zigbee ネットワークスキャンを開始
    Zigbee.scanNetworks();
}

void loop() {
    // Zigbee ネットワークスキャンの進行状況を確認
    int16_t ZigbeeScanStatus = Zigbee.scanComplete();
    if (ZigbeeScanStatus < 0) {  // スキャン中またはエラーが発生
        if (ZigbeeScanStatus == ZB_SCAN_FAILED) {
        Serial.println("Zigbee スキャンに失敗しました。再度開始します。");
        delay(1000);
        Zigbee.scanNetworks();
        }
        delay(100);
        // もうひとつの選択肢はステータス ZB_SCAN_RUNNING - 待機します。
    } else {  // 0 件以上の無線ネットワークが見つかった
        printScannedNetworks(ZigbeeScanStatus);
        delay(1000);
        Zigbee.scanNetworks();  // 再度スキャン開始...
    }
    // ループ内で他の処理も可能です...
}
```

## 使用手順

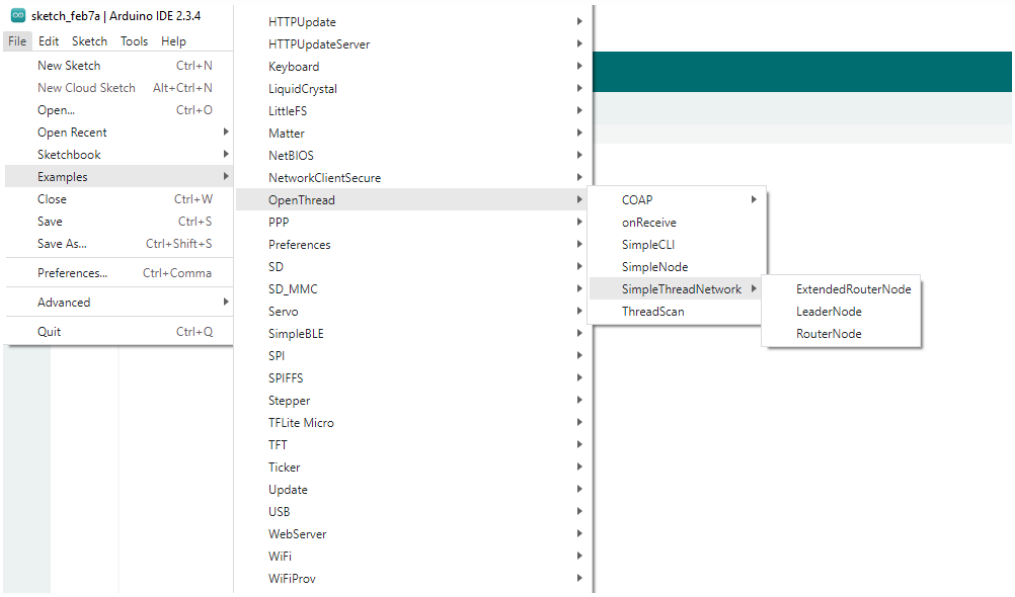
- 1.デバイス起動後、自動的にスキャンが開始され、周辺にアクティブな Zigbee ネットワークが存在する場合、各スキャン完了後に結果が表示され、次のスキャンが自動的に開始されます。




# 5. OpenThread

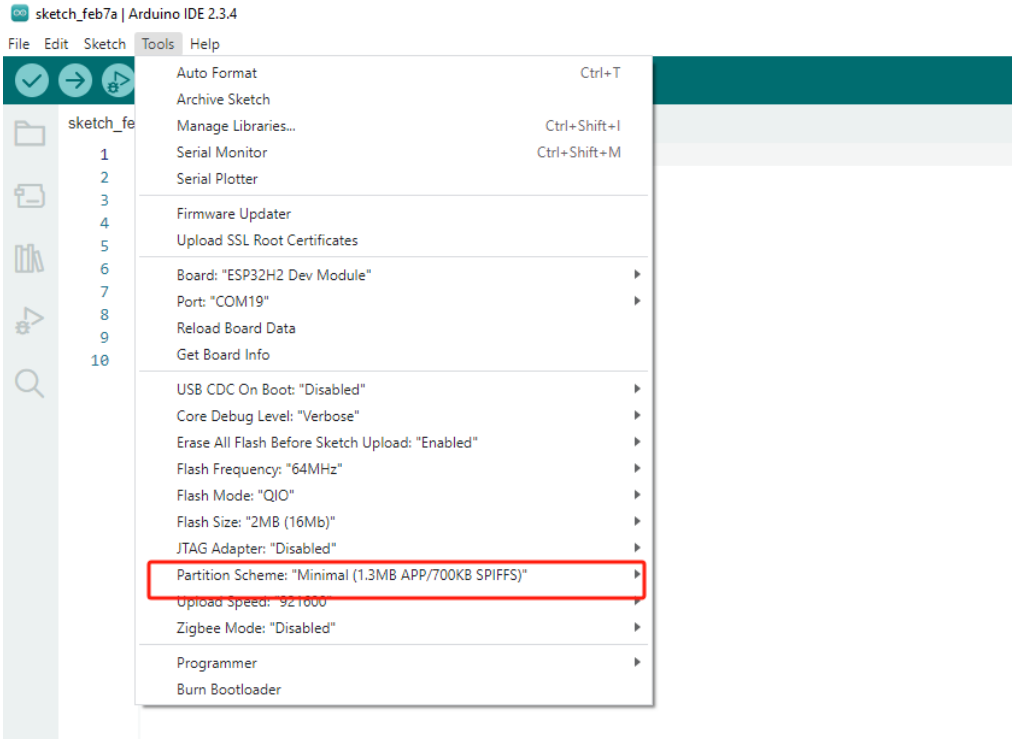
**サンプルプログラム**

メニューパス `File -> Examples -> OpenThread` で OpenThread 関連のサンプルプログラムを確認できます。プログラムをコンパイルする前に、以下のパーティションテーブル設定等の情報を参照してください。




**Arduino IDE ツールメニューの設定:**

- 正しいボードを選択: `Tools -> Board: ESP32H2 Dev Module`
- フラッシュ消去を有効にする: `Tools -> Erase All Flash Before Sketch Upload: Enable` (無効にすると接続に失敗する可能性があります)
- フラッシュサイズを選択: `Tools -> Flash Size: 2MB`
- Zigbee パーティションスキームを選択: `Tools -> Partition Scheme: Minimal SPIFFS (1.3MB APP/700K SPIFFS)`


sketch_feb7a | Arduino IDE 2.3.4
File Edit Sketch Tools Help
sketch_fe
1
2
3
4
5
6
7
8
9
10
Auto Format Ctrl+T
Archive Sketch
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter
Firmware Updater
Upload SSL Root Certificates
Board: "ESP32H2 Dev Module"
Port: "COM19"
Reload Board Data
Get Board Info
USB CDC On Boot: "Disabled"
Core Debug Level: "Verbose"
Erase All Flash Before Sketch Upload: "Enabled"
Flash Frequency: "64MHz"
Flash Mode: "QIO"
Flash Size: "2MB (16Mb)"
JTAG Adapter: "Disabled"
Partition Scheme: "Minimal (1.3MB APP/700KB SPIFFS)"
Upload Speed: "921600"
Zigbee Mode: "Disabled"
Programmer
Burn Bootloader