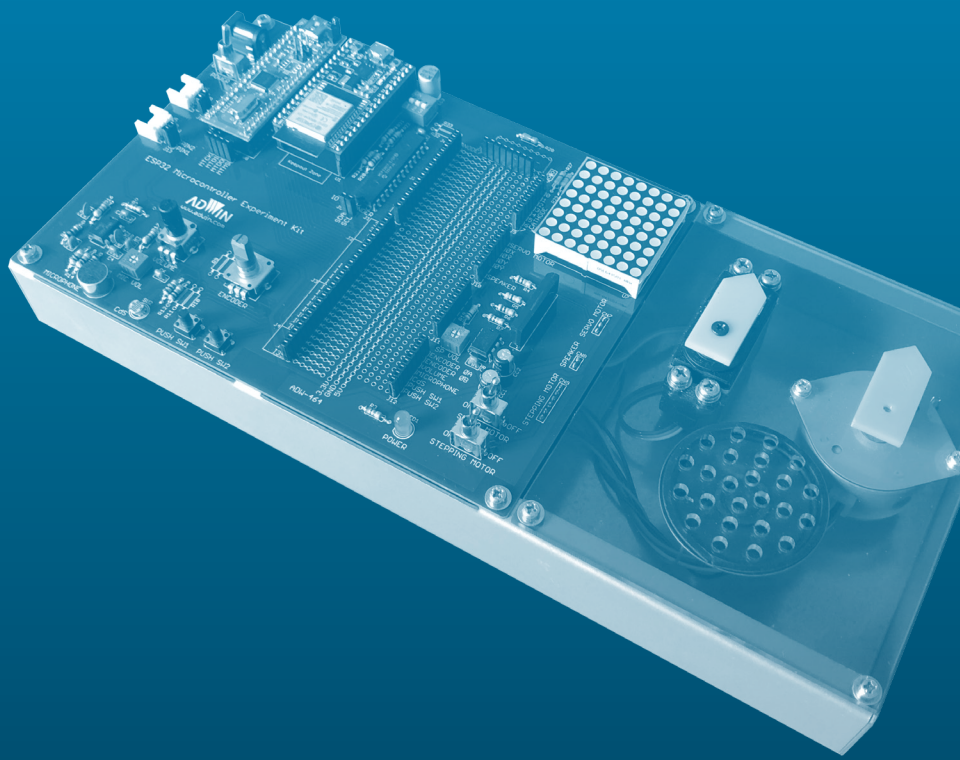
C言語で制御する

ESP32マイコン入門



ADWIN

開発環境構築

もくじ

注意事項

開発用ソフトウェアはすべてインターネットから入手してインストールします。
スタンドアロンもしくはセキュアな環境でインターネットからインスールできない場合は，ネットワークにつなげることのできる別の PC で必要なファイルをダウンロードし，開発用 PC にコピーしてお使いください。

なお，㈱アドウィンでは，お客様からソフトウェアの欠陥・障害や不具合が報告された場合，速やかにそれらを回避するための情報を提供するよう努めますが，修繕やバージョンアップ等の対応は必ずしもお約束できません。万が一，お客様がソフトウェアの欠陥に起因する直接的・間接的損害を被った場合においても，㈱アドウィンは一切の補償の義務を負いません。本キットの使用に必要な個々のソフトウェアは，お客様ご自身の責任の下で運用されるものといたします。
お客様はこの趣旨を十分に理解し，同意した上でソフトウェアをご使用願います。もし，同意いただけない場合，ソフトウェアのインストールや使用はご遠慮ください。本規定は，お客様に対してコンパイラ製品を含む本キットに付属する開発環境の使用を強要するものではありません。従いまして，お客様が上記に同意いただけない場合，別途，各社提供のソフトウェア開発環境製品をお買い求め頂きますよう，よろしくお願いいたします。なお，本記載内容は㈱アドウィンおよびお客様の有する法的権利を制限するものではありません。

01 開発環境のインストール

1-1. 拡張子の表示

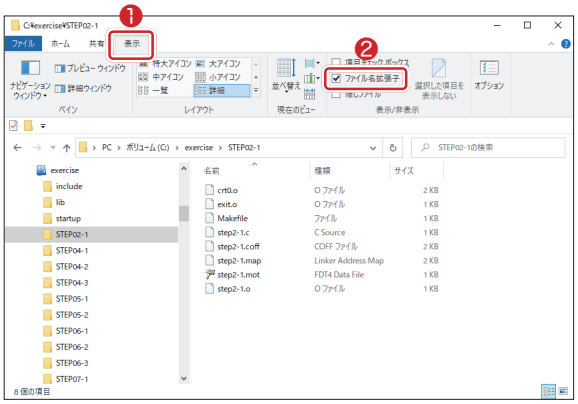
開発環境構築および実習では，拡張子を表示させた状態を図示したり，拡張子を含めたファイル名で説明しています。お手持ちのパソコンを，拡張子を表示させる設定にしておいてください。

> **拡張子**
> 拡張子とは，ファイルの種類を表す文字のことを言います。
> テキストだと「.txt」，C言語だと「.c」など，「．」に続いて表示される文字。

以下のいずれかの方法で設定できます。

❶ エクスプローラーの「表示」タブを開く。

❷「ファイル名拡張子」という項目にチェックを入れる。

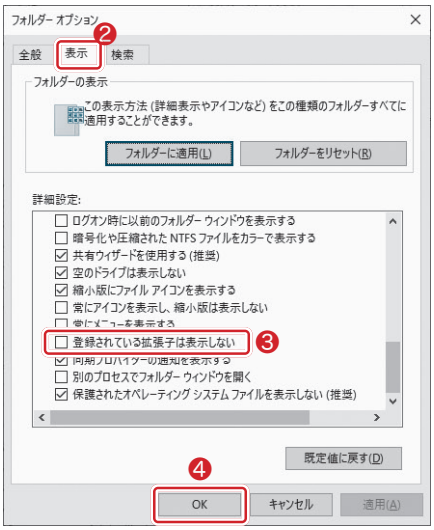


❶ エクスプローラーの「ファイル」メニューで［オプション］をクリック。

❷「フォルダーオプション」の「表示」タブをクリック。

❸ 詳細設定欄にある「登録されている拡張子は表示しない」という項目のチェックを外す。

❹「OK」をクリック。

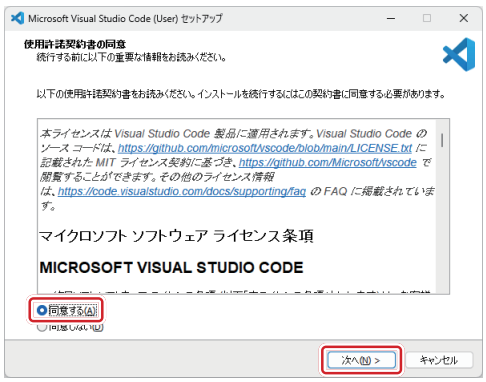


1-2. VSCode のインストール

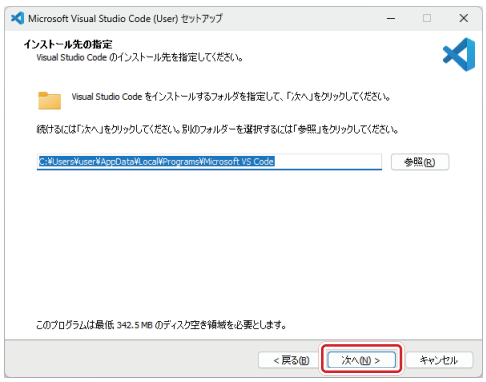
Visual Studio Code（以後，VSCode）の WEB サイトから，VSCode のインストーラをダウンロードしてください（お使いの環境によって Linux，macOS ではそれぞれの版を選択してください）。

https://azure.microsoft.com/ja-jp/products/visual-studio-code

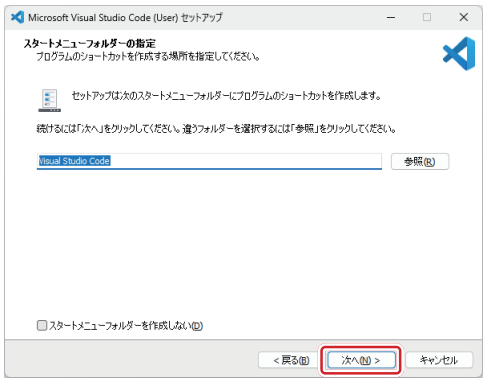
ダウンロード終了後，インストーラを起動し，指示にしたがってインストールしてください。例えば，Windows をお使いなら，ダウンロードファイル「VSCodeUserSetup-***-***.exe 」をクリックして実行します。



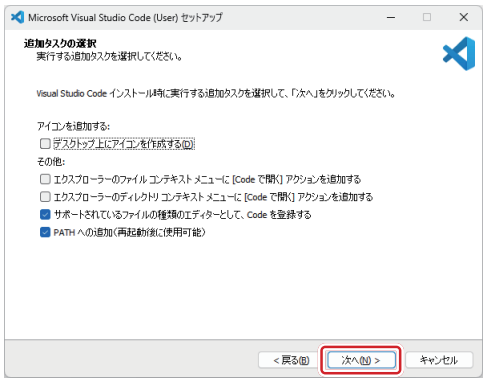
❶「使用許諾契約書の同意」で，契約書に同意した上で「同意する」を選択し，「次へ」をクリック。



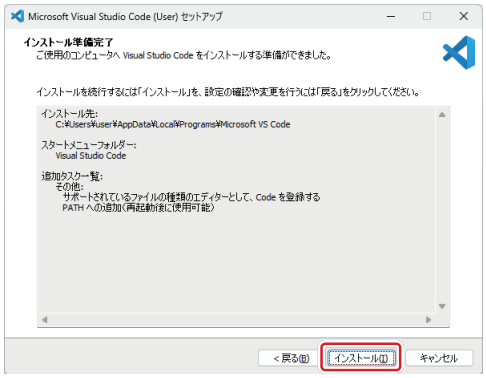
❷「インストール先の指定」で，VSCode のインストール先を指定する。特に事情が無ければ初期値のまま「次へ」をクリック。VSCode が起動する。



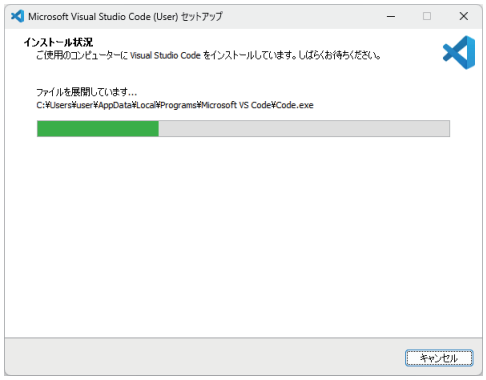
❸「スタートメニューフォルダ」で，プログラムのショートカットを作成する場所を指定する。特に事情が無ければ初期値のまま「次へ」をクリック。



❹「追加タスクの選択」で，インストール時に実行する追加タスクを選択する。特に事情が無ければ初期値のままで「次へ」をクリック。



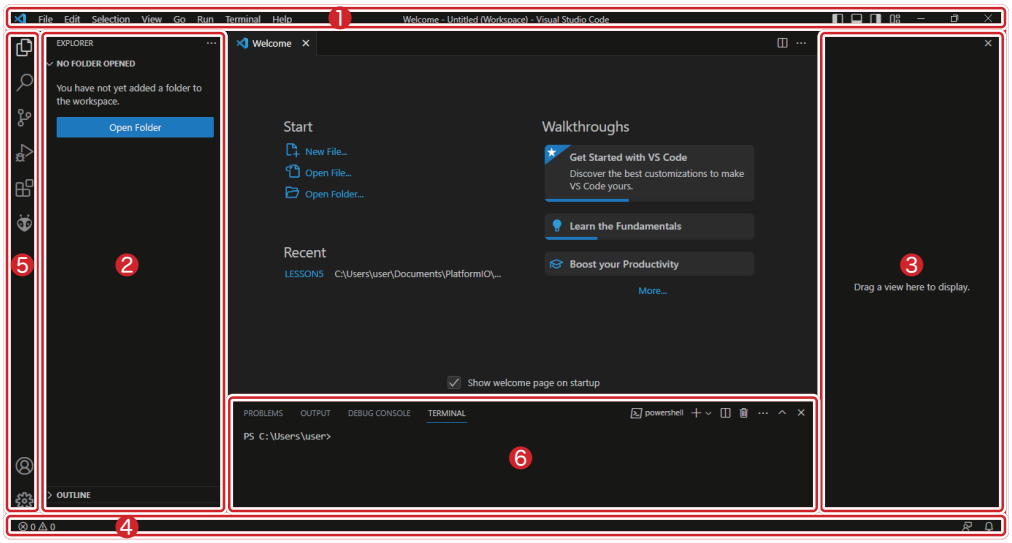
❺「インストール準備完了」で，インストールの内容を確認し，「インストール」をクリック。



❻ インストール中

❼「Visual Studio Code セットアップウィザードの完了」が表示されたら，「完了」をクリック。「Visual Studio Code を実行する」にチェックを入れておくと，VSCode が起動する。

■ VScode 画面各部名称



各部名称は日本語化後のものです。
VScode の日本語化は次ページを参照してください。

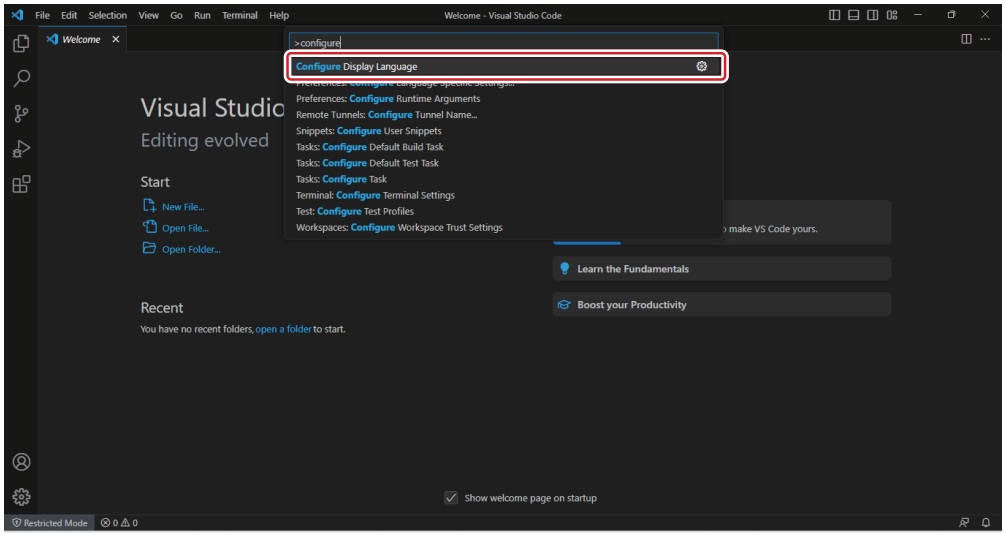
メニューバーの表示＞外観＞で各バーやパネルの表示／非表示を切り替えられます。

❶ メニューバー
❷ プライマリ サイドバー
❸ セカンダリ サイドバー
❹ ステータスバー
❺ アクティビティバー
❻ パネル

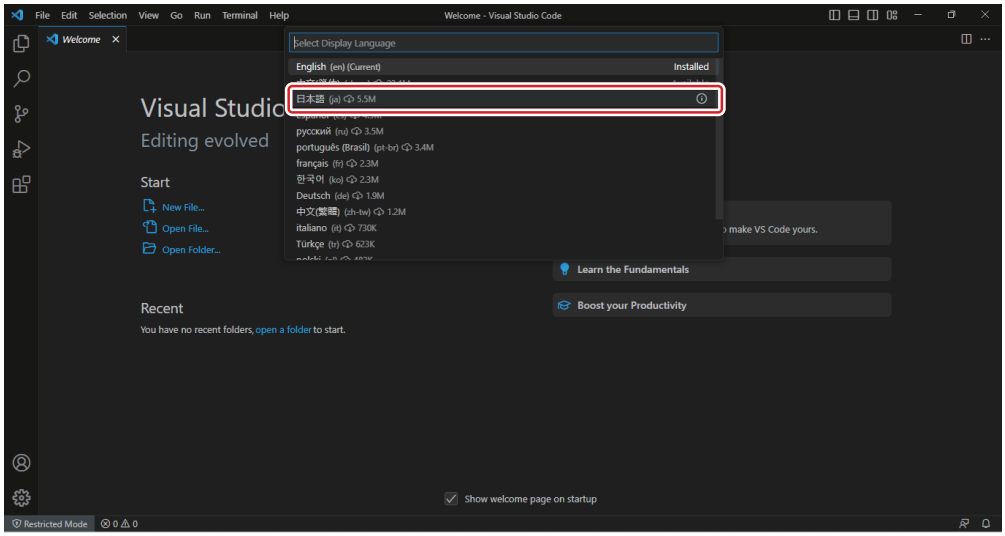
■ VScode の日本語化

メニュー表示が英語になっている場合は，以下の手順で日本語化することができます。
次ページ以降の解説や画面サンプルは，日本語化済のものを掲載します。

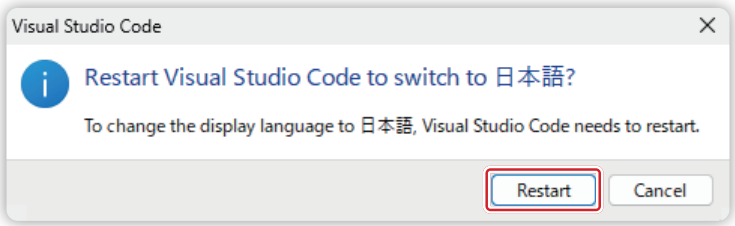
❶ VSCode を起動し，「F1」キーをクリック。検索窓が出てきたら「configure」と入力し，
表示された候補の中から「Configure Display Language」を選択。



❷ 表示する言語の候補が出てくるので「日本語」を選択。
すると，日本語言語パッケージのインストールが自動的に行われる。



❸インストールが完了すると，表示言語を変更するために再起動を促すメッセージが表示されるので「restart」をクリック。

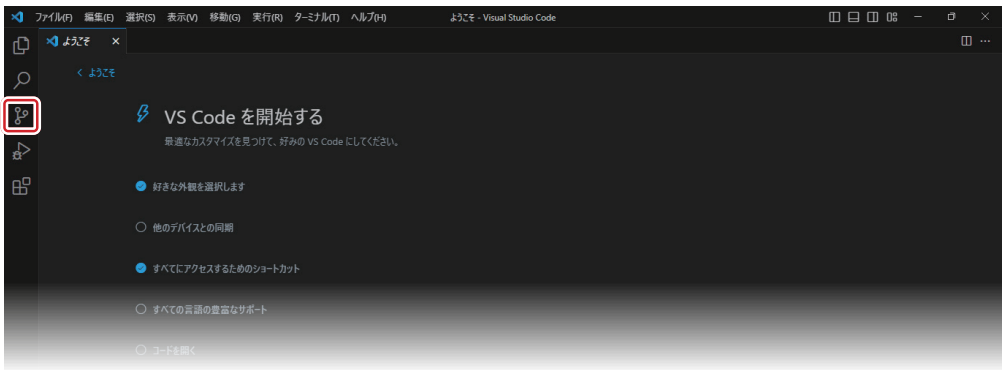


❹ VSCode が再起動し，メニュー等が日本語化された状態になる。

1-3. PlatformIO のインストール

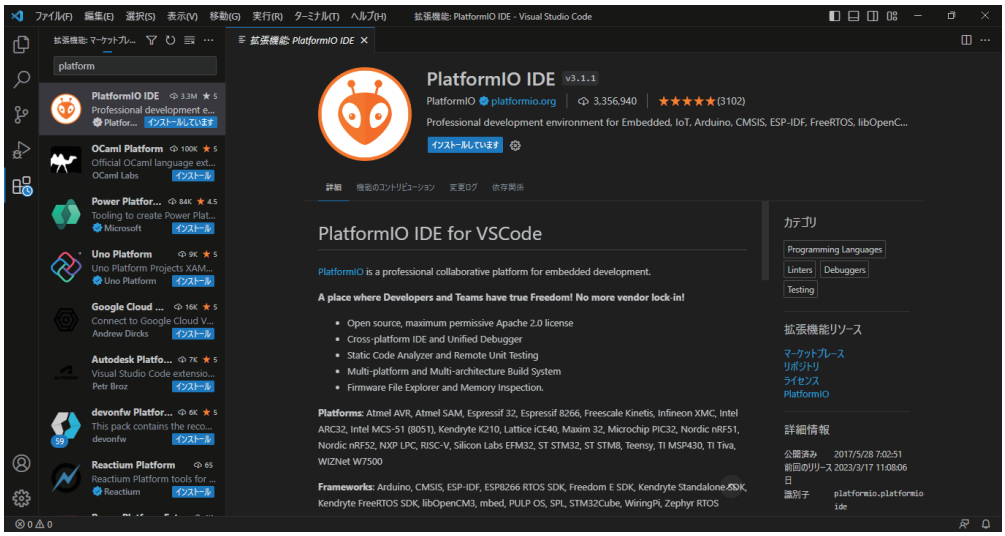
PlatformIO（プラットフォームアイオー）はオープンソースの統合開発環境（IDE）で，組み込みシステムの開発を容易にするためのツールセットです。PlatformIO は，プログラミング言語として C や C++ を使用し，ESP32 の他に Arduino や STM32 などのプロジェクト開発をサポートしています。
では，VScode に PlatformIO の機能拡張をインストールしていきましょう。

❶ VSCode のアクティビティバーで「拡張機能」をクリック。

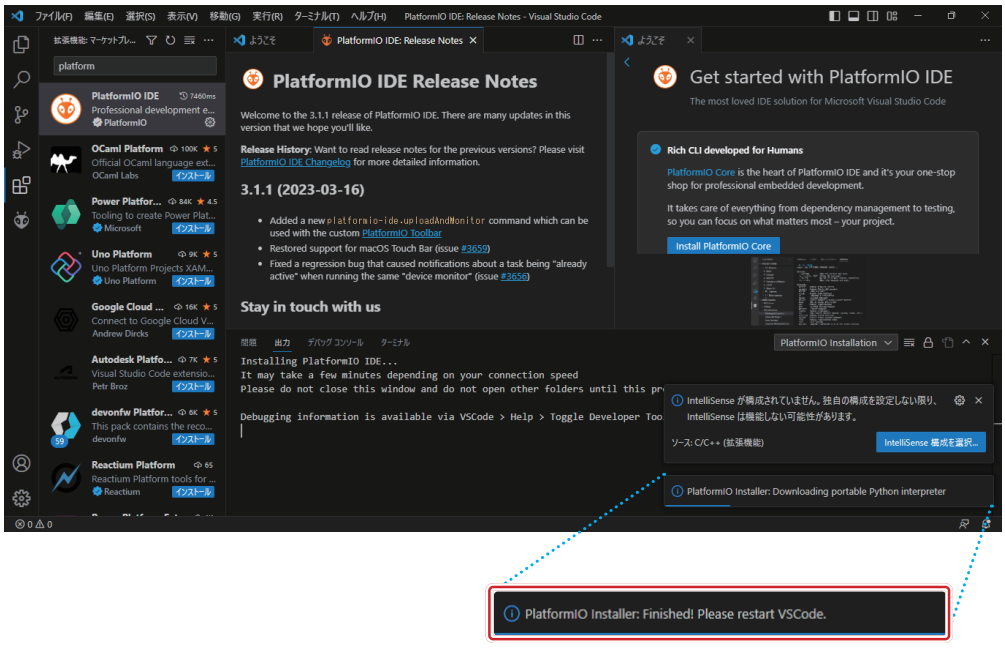


❷検索窓が出てきたら「platformio」と入力し，表示された候補の中から「PlatformIO IDE」の「install」をクリック。

❸ インストール中



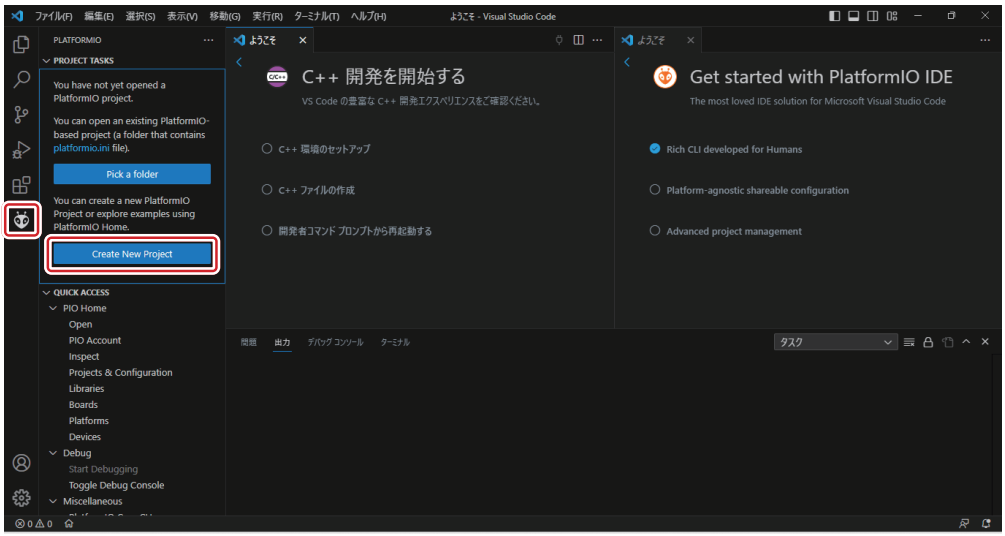
❹ 画面右下に「PlatformIO installer：Finished!」のメッセージが出て，インストールが完了したら VSCode をいったん終了する。再度起動後，PlatformIO が使えるようになる。



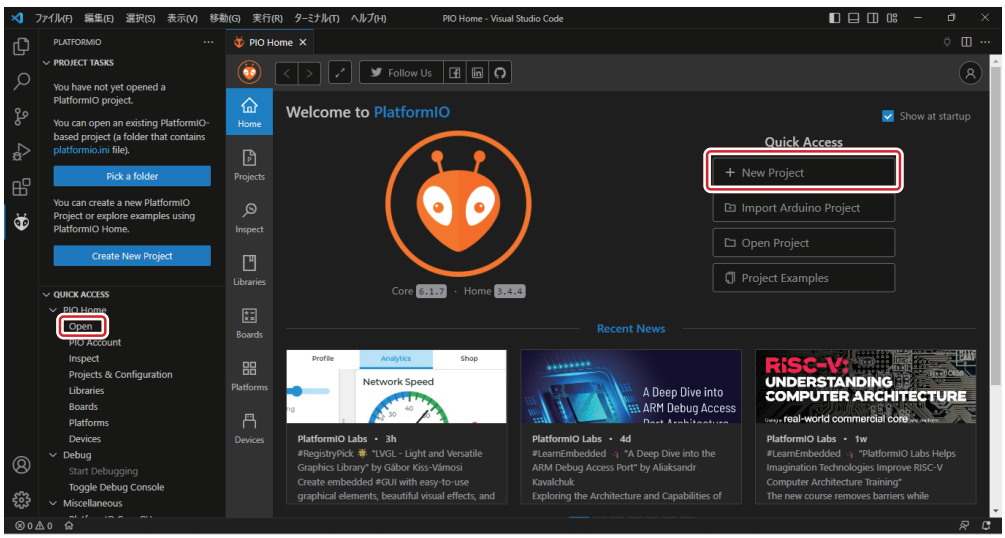
02 プロジェクトの新規作成

2-1. プロジェクトの新規作成

❶ VSCode のアクティビティバーで「PlatformIO」をクリック。
しばらく「Initializing PlatformIO Core...」と表示されるので完了するまで待つ（初回はやや時間がかかる）。
プロジェクト作成メニューが出てきたら「Create New Project」を選択する。



もしくは「QUICK ACCESS」→「PIO Home」→「Open」で「PIO Home」を開き,「Quick Access」から「New Project」を選択する。



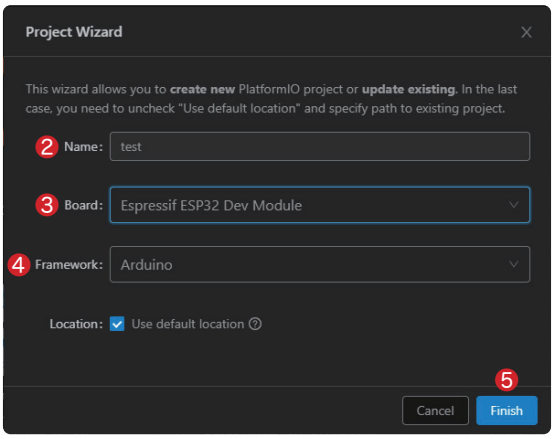
❷「Project Wizard」「Name」にプロジェクト名を入力する。

> ⚠ 使える文字は 半角英数字　と　_（アンダーバー）と　-（ハイフン）のみ
> 文字と小文字は区別されないので注意。

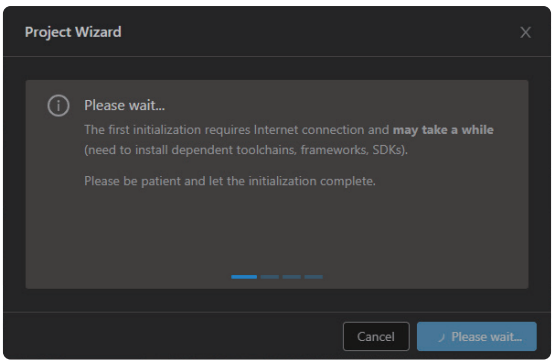
❸「Board」に「esp32 dev module」と入力し，表示された候補の中から「Espressif ESP32 Dev Module」を選択する。

❹「Framework」は「Arduino」のままにしておく。

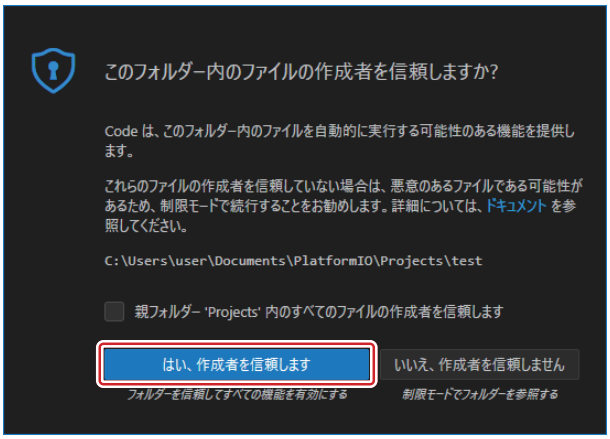
❺ 以上が完了したら「Finish」をクリック。



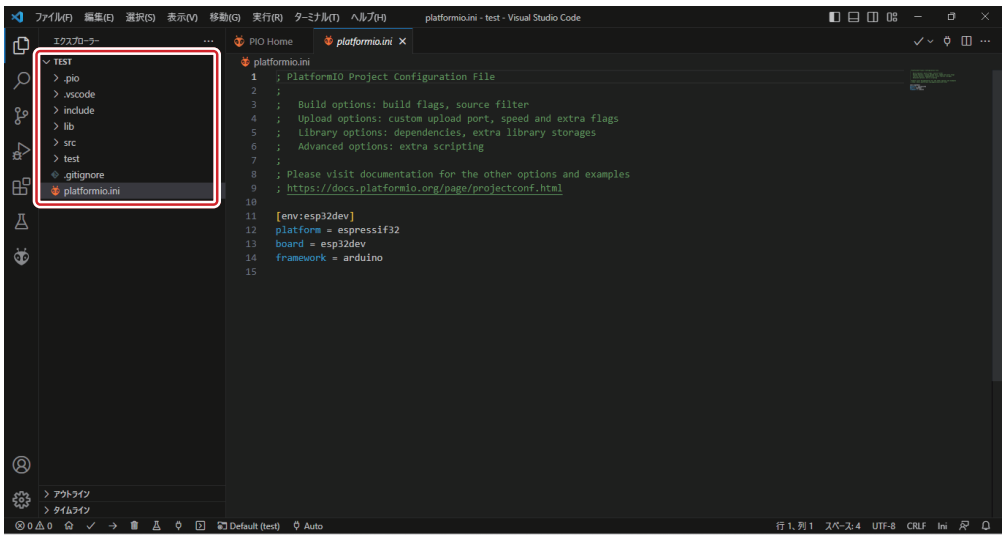
❻「Please wait...」と表示されるので，しばらく待つ（5~20 分程度時間がかかる。時間がかかるのは PlatformIO インストール後の初回のみ）。



❼「このフォルダー内のファイルの作成者を信頼しますか？」と表示されたら，「はい，作成者を信頼します」を選択する。



❽ VScode のエクスプローラーに作成したプロジェクトが表示されれば作成完了。



2-2. プロジェクトのディレクトリ構造

VSCode のエクスプローラーを開くと，PlatformIO のプロジェクトのディレクトリ構造を確認することができます。

ディレクトリ構造（および自動生成されるファイル）は以下のようになっています。

.pio	PlatformIO が使うディレクトリです。この中身を直接触ることは原則としてありません。
.vscode	VSCode の設定等に関するディレクトリです。
include	プログラムのヘッダファイルを置くことを意図したディレクトリです。初期状態では，このディレクトリの使い方を説明した「README」ファイルが置かれています。
lib	プログラムのライブラリを置くことを意図したディレクトリです。初期状態では，このディレクトリの使い方を説明した「README」ファイルが置かれています。
src	プログラムのソースコードを置くことを意図したディレクトリです。初期状態では，スケルトン（骨組み）のコードが書かれた「main.cpp」ファイルが置かれています。
test	プログラムのテスト用に用いることを意図したディレクトリです。初期状態では，テストに関する説明の書かれた「README」ファイルが置かれています。
.gitignore	プロジェクトを「git」で管理する際に，追跡しないディレクトリやファイルを指定する設定ファイルです。
platformio.ini	PlatformIO のプロジェクトに関する設定ファイルです。

プログラムを作成する場合は，「src」内の「main.cpp」ファイルからプログラムを書いていきます。また，必要に応じて「platformio.ini」ファイルの設定や，他のファイルの追加等を行います。

03 最初のプログラム

まずは簡単なプログラムを作成してみましょう。ここでは，最初のプログラムとして「シリアル通信」のプログラムを作ることにします。シリアル通信についてのより詳しい説明は「実習テキスト」の「STEP04　シリアル通信」で行いますが，とりあえずは「マイコンからPC に文字列を送るプログラム」と考えておきましょう。以下は，1 秒おきにシリアルモニタに「Hello, world!」と出力するプログラムを作成する例です。

3-1. PlatformIO の設定

プロジェクトを新規作成したら，「platformio.ini」に「monitor_speed = 115200」を追記します。この設定で，デフォルトの通信速度 9600bps を 115200bps に変更します。この設定が無い場合，シリアルモニタ上で文字化けが起こることがあります。

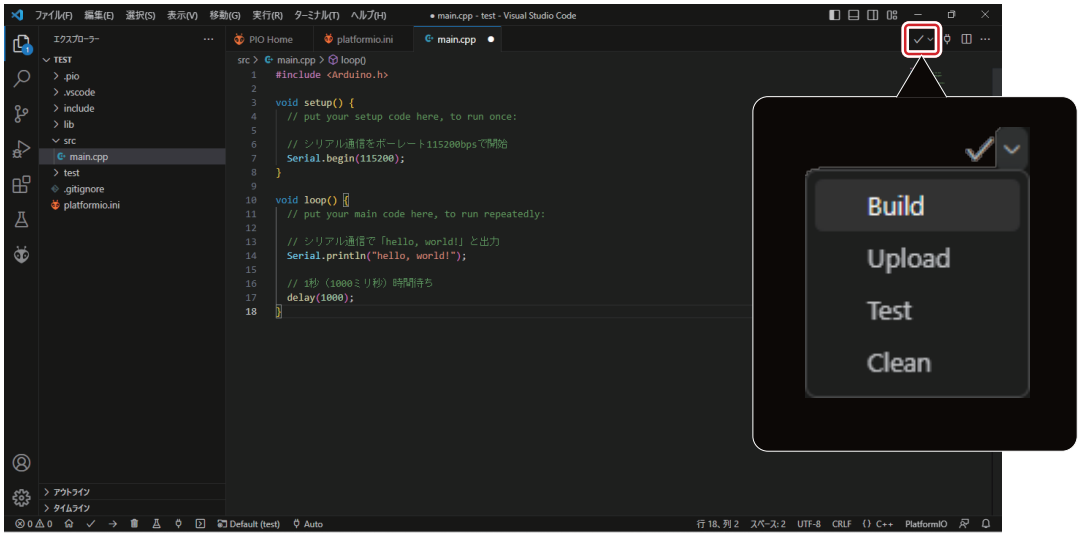
```
11  [env:esp32dev]
12  platform = espressif32
13  board = esp32dev
14  framework = arduino
15  monitor_speed = 115200
```

3-2. 最小限のプログラム

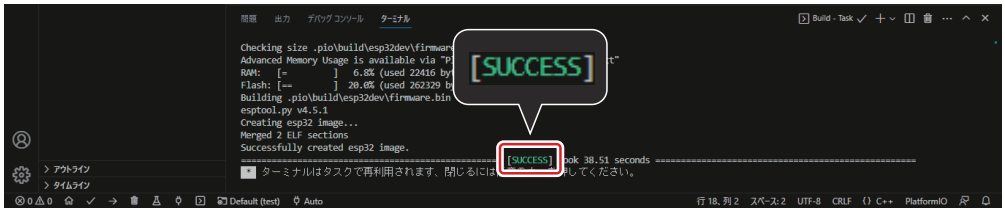
VScode のエクスプローラで「src」→「main.cpp」を開いてみましょう。これがメインのプログラムファイルです。プロジェクト作成直後には，main.cpp にはプログラムの雛形となるコードが書かれています。以下は初期状態での main.cpp のコード例です（実際のコードは PlatformIO のバージョン等により異なる場合があります）。

```
1  #include <Arduino.h>
2
3  // put function declarations here:
4  int myFunction(int, int);
5
6  void setup() {
7    // put your setup code here, to run once:
8    int result = myFunction(2,3);
9  }
10
11 void loop() {
12   // put your main code here, to run repeatedly:
13 }
14
15 // put function definitions here:
16 int myFunction(int x, int y) { return x + y; }
```

画面右上のチェックマークをクリック，もしくは ∨ をクリックして表示されるメニューから「Build」をクリックし，プロジェクトのビルド（コンパイルおよび実行ファイルの作成）を行います。



ターミナルに「SUCCESS」のメッセージが表示されればビルド成功です。



なお，上記コードには適正なプログラムとして最小限必要なコードの他に，雛形として便利なコードがいくらか含まれています。最小限必要なコード以外を除去すると，以下のようになります。

```
#include <Arduino.h>

void setup() {
}

void loop() {
}
```

このコードもビルドできることを確認しておきましょう。

3-3. シリアル通信のプログラム

それでは，「1 秒おきにシリアルモニタに「Hello, world!」と出力するプログラム」を作ってみましょう。
「main.cpp」を以下のようにコーディングしましょう。詳しい解説は後の STEP で行いますので，まずはこの通り正確に書いてみましょう。

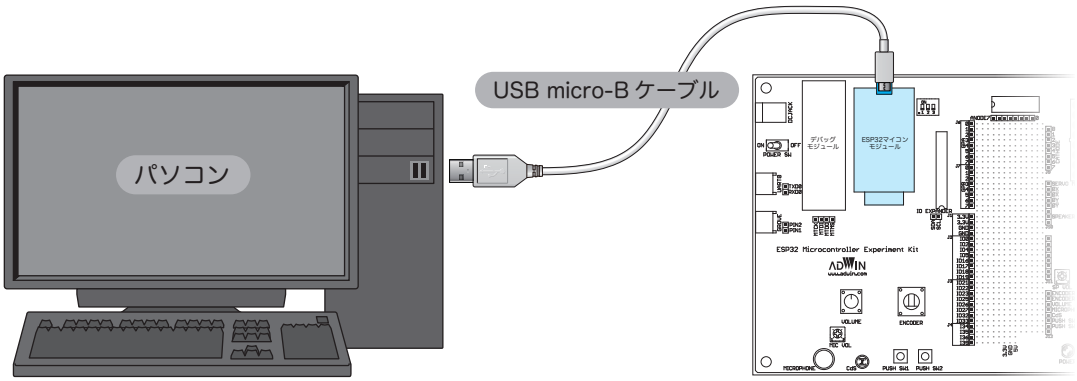
```
1  #include <Arduino.h>
2
3  void setup() {
4    Serial.begin(115200);
5  }
6
7  void loop() {
8    Serial.println("hello, world!");
9    delay(1000);
10 }
```

> ⚠ プログラムのコードは，原則として「半角英数字」「半角記号」「半角スペース」「改行」のみが使用可能です。

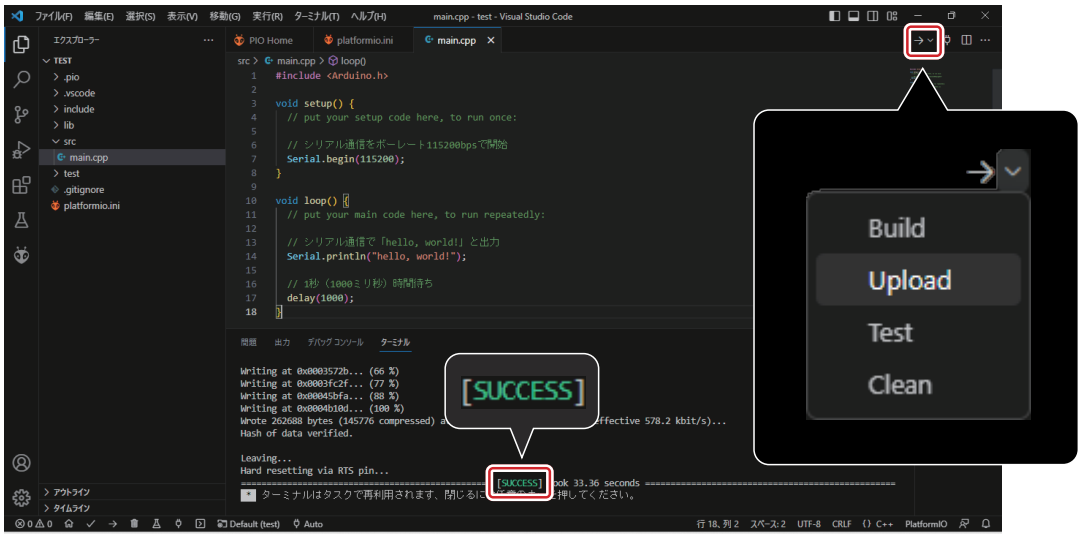
コーディングが完了したら，プロジェクトのビルド（コンパイルおよび実行ファイルの作成）を行います。ターミナルに「SUCCESS」とメッセージが表示されればビルド成功です。

3-4. プログラムのマイコンへの書き込み

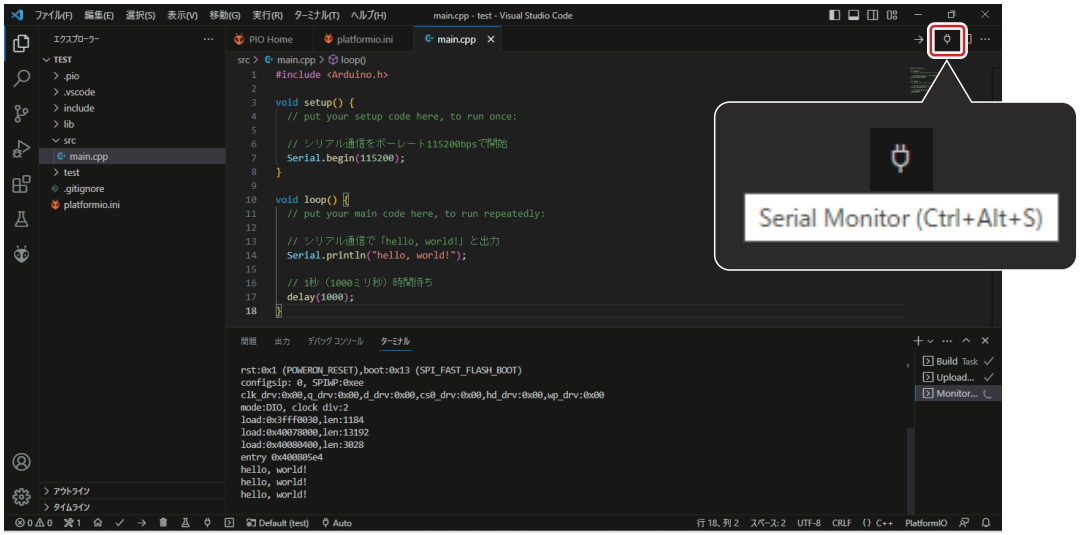
プロジェクトのビルドに成功したら，マイコンにプログラムを書き込みましょう。
まずは，パソコンとマイコンモジュールを USB Micro-B ケーブルで接続します。



画面右上の右矢印マークをクリック，もしくは ∨ をクリックして表示されるメニューから「Upload」をクリックし，マイコンにプログラムをアップロード（書き込み）します。ターミナルに「SUCCESS」とメッセージが表示されればアップロード成功です。



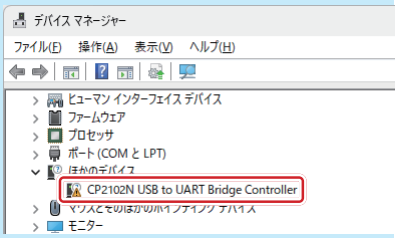
「Serial Monitor」をクリックし，シリアルモニターを表示します。ターミナルに，マイコンに関する情報の出力に続いて 1 秒おきに「Hello, world!」が出力されるでしょうか。



upload できない

upload が失敗 [FAILED] する場合は，ESP32 用の USB - UART ブリッジ仮想 COM ポート（VCP）ドライバ「CP2102N」がインストールされていないかもしれません。

Windows なら，ドライバがインストールされていないことは OS のデバイスマネージャで [CP2102N] に警告マークが付いていることで分かります。その場合は以下の手順でドライバをインストールしてください。



❶ 以下のリンクから，お使いの OS に対応したドライバをダウンロードする。

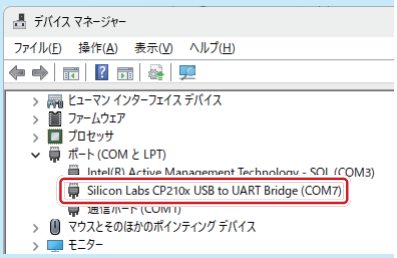
https://jp.silabs.com/developers/usb-to-uart-bridge-vcp-drivers?tab=downloads

❷ ダウンロードした zip ファイルを解凍し，ご使用の PC プロセッサに対応したインストーラを起動する。

64bit プロセッサ用 →
32bit プロセッサ用 →

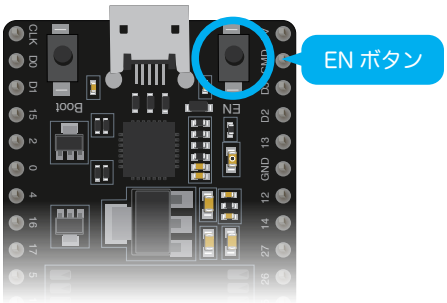
名前	更新日時	種類	サイズ
x64	2018/06/16 0:13	ファイル フォルダー	
x86	2018/06/16 0:13	ファイル フォルダー	
CP210xVCPInstaller_x64.exe	2017/09/28 2:58	アプリケーション	1,026 KB
CP210xVCPInstaller_x86.exe	2017/09/28 2:58	アプリケーション	903 KB
SLAB_License_Agreement_VCP_Windows....	2017/09/28 2:46	Text Document	9 KB
slabvcp.cat	2018/06/02 5:35	セキュリティ カタログ	11 KB
slabvcp.inf	2018/06/02 5:35	セットアップ情報	8 KB
v6-7-6-driver-release-notes.txt	2018/06/16 3:51	Text Document	16 KB

❸ ドライバのインストール終了後，デバイスマネージャで下図のように表示されれば OK。



3-5. マイコンのリセット

3-4 まで確認できたら，マイコンモジュールの EN ボタンを押してマイコンをリセットしてみましょう。



リセット後，ターミナルにまずはマイコンに関する情報が出力され，その後「Hello, world!」が 1 秒おきに出力されることを確認しましょう。

```
問題　出力　デバッグ コンソール　ターミナル

Hello, world!
ets Jul 29 2019 12:21:46

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:1184
load:0x40078000,len:13104
load:0x40080400,len:3036
entry 0x400805e4
Hello, world!
Hello, world!
Hello, world!
```

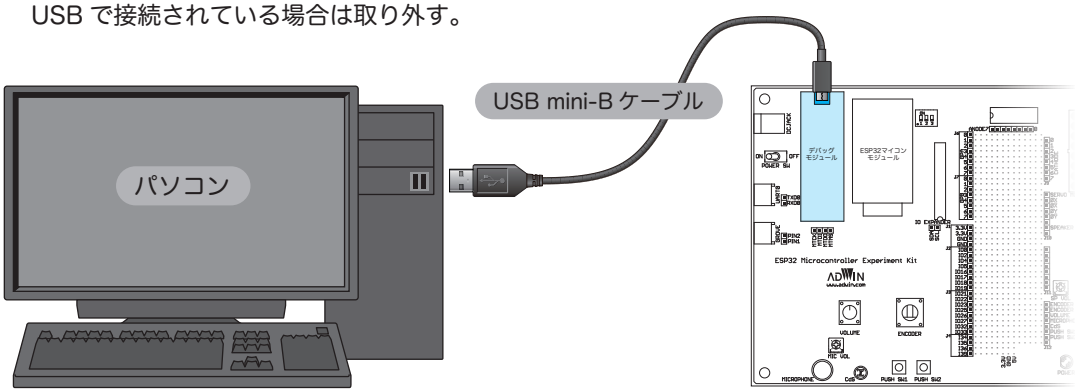
04 デバッグ環境の構築

4-1. FT2232D のドライバ書き換え

本キットは，デバッグモジュールとして FT2232D を搭載した AE-FT2232 を採用しています。Windows の場合，FT2232D のドライバを書き換える必要があります（Linux, MacOS では不要です）。

> ⚠ JTAG でデバッグを行う場合，JTAG 用のピン（IO12 ～ IO15）は原則として他の用途には使えません。

❶ AE-FT2232 を USB Mini-B ケーブルで PC と接続する。また，ESP32 モジュールが USB で接続されている場合は取り外す。



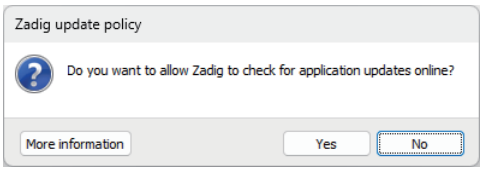
❷ デバイスマネージャーで二つの USB Serial Port が認識されていることを確認する。

❸ Web ブラウザで以下サイトにアクセスし，Zadig をダウンロードする。

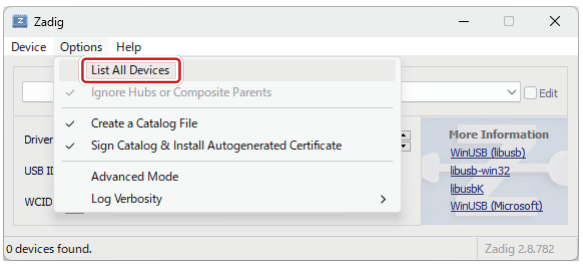
https://zadig.akeo.ie/

❹ ダウンロードした Zadig.exe を実行する。

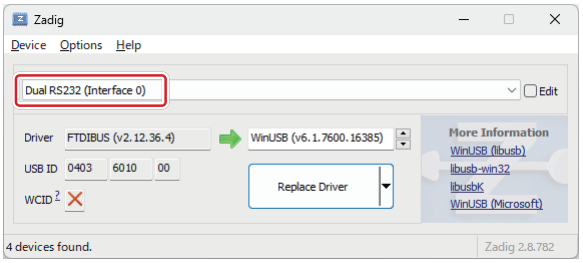
❺「Zadig update policy」は，「Yes」または「No」のどちらでも構わないのでクリック。



❻「Options」をクリックし，「List All Devices」にチェックを入れる。

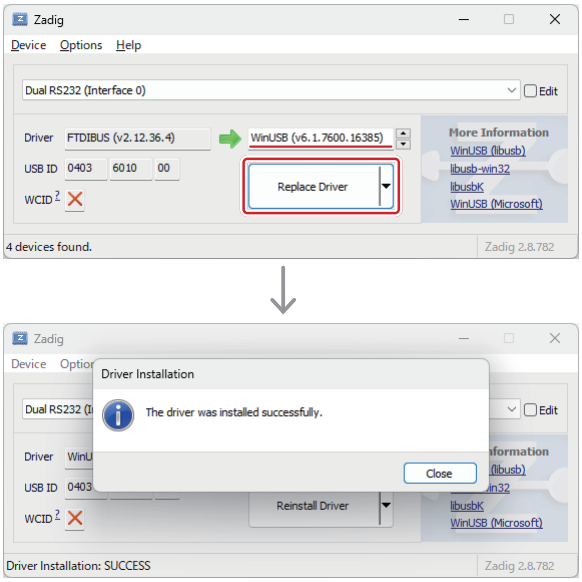


❼「Dual RS232 (Interface 0)」を選択し，デバイス名（「Dual RS232」）をメモしておく。なお，環境等により別の名前（「FT2232H MiniModule (Interface 0)」等）になっている可能性があるが，「(Interface 0)」となっているものを選択する。



以下，デバイス名「Dual RS232」の例で解説を進めるが，別の名前だった場合はその部分を読み替えること。

❽「Driver」の書き換え先（矢印の右側）が「WinUSB」になっていることを確認し，「Replace Driver」をクリックしてドライバを書き換える（しばらく時間がかかる）。
なお，書き換え済みの場合は「Reinstall Driver」と表示される。



❾ 書き換えが完了したら，デバイスマネージャーで USB Serial Port がひとつになっていることを確認する。また，「ユニバーサルシリアルバスデバイス」に「Dual RS232」というデバイスがあることを確認する。
なお，元に戻したい場合はデバイスマネージャーからドライバの再インストールを行う。

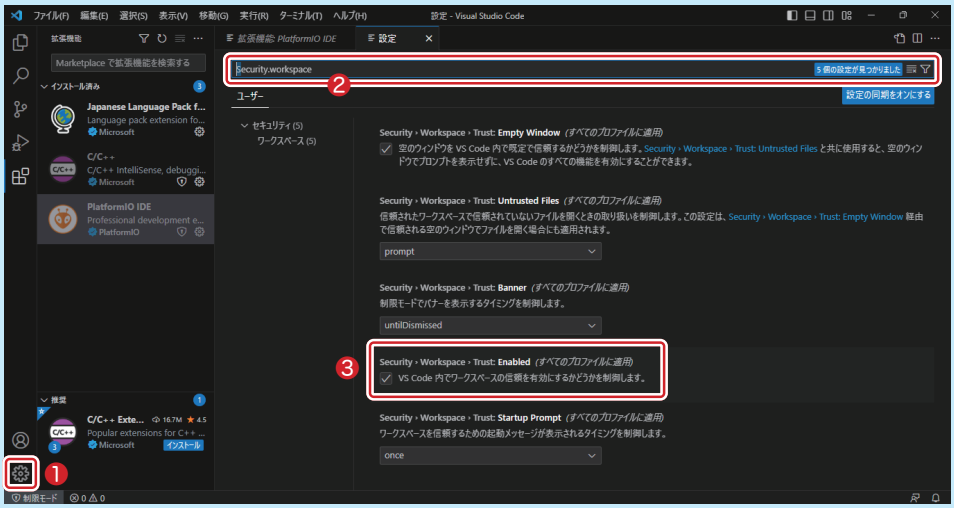
VScode の制限モード

制限モードによりプロジェクトが開けなくなってしまった場合，回避する方法はいくつかありますが，一時的に制限モードを解除する簡便な方法をご紹介しておきます。

❶ アクティビティバーの管理ボタン（歯車アイコン）> 設定 をクリック

❷ 検索エリアに security.workspace と入力

❸ Security > Workspace > Trust：Enabled のチェックを外す。

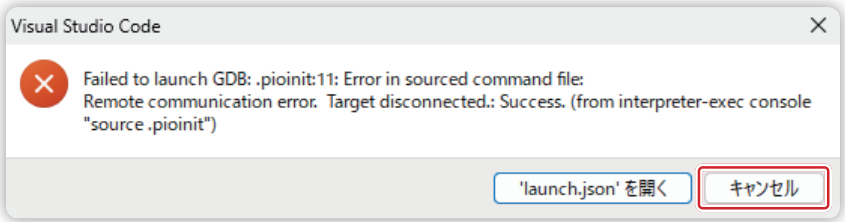


4-2. PlatformIO の設定

以下は Windows の場合の設定方法です（Linux，macOS ではパスを読み替えてください。デバイス名は dmesg コマンドで確認できます）。

❶ minimodule.cfg を編集する必要があるが，デバッグ実行前はファイルが存在しない。
一度「F5」キー（= メニュー 実行 > デバッグの開始）を押して，minimodule.cfg を含むデバッグ用ファイルを生成する。

この時点でデバッグモード移行できなくて問題なし。「キャンセル」をクリック。



❷ OS のエクスプローラーで %homepath%\.platformio\packages\tool-openocd-esp32\share\openocd\scripts\interface\ftdi の中のファイル minimodule.cfg を開く。

%homepath% は，Windows であれば，C:\Users\（ログインユーザーアカウント）\

❸「ftdi device_desc "FT2232H MiniModule"」の「FT2232H MiniModule」を，「FT2232D ドライバの書き換え」でメモしたデバイス名の「Dual RS232」に書き換える。既に一致している場合は書き換え不要。以下は書き換え例。

```
 9 adapter driver ftdi
10 # ftdi device_desc "FT2232H MiniModule"  ← 元コードをコメントアウト　削除しても OK
11 ftdi device_desc "Dual RS232"
12 ftdi vid_pid 0x0403 0x6010
```

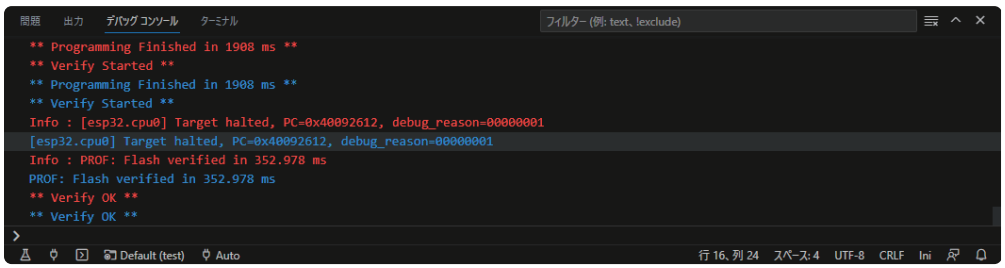
4-3. デバッグモードでのプログラム実行

❶ プロジェクトの「platform.ini」を開き，末尾に「debug_tool = minimodule」を追加する。以下は設定例。

```
[env:esp32dev]
platform = espressif32
board = esp32dev
framework = arduino
monitor_speed = 115200
debug_tool = minimodule
```

❷ メインボードの電源を ON にする。

❸「F5」キーを押すと，デバッグに向けた処理（現在のプログラムのコンパイルおよび書き込みを含む）が開始される。なお，デバッグモードでの操作ができるようになるにはやや時間がかかる。「デバッグコンソール」を開くと進捗を確認することができる。

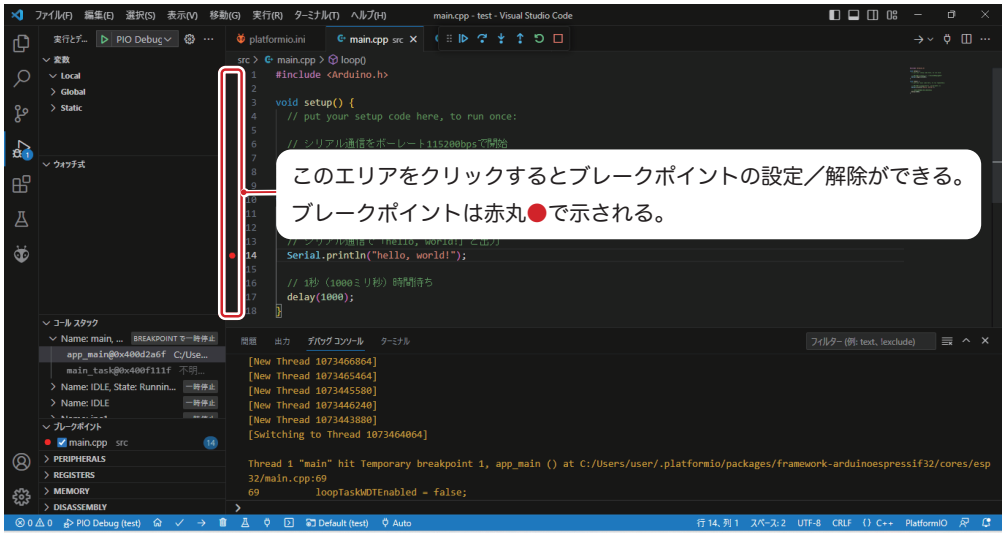


❹ デバッグツールバーが表示されれば準備完了。

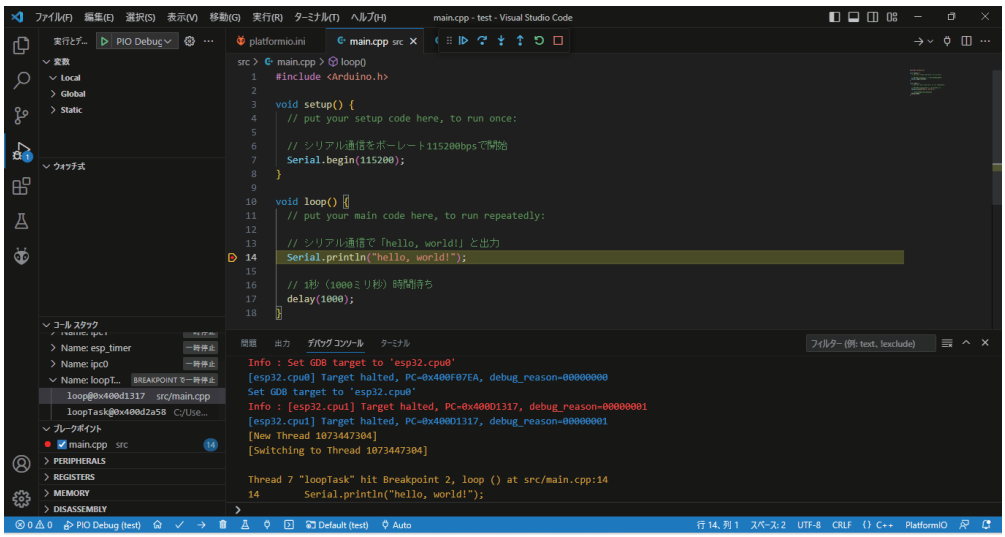
ボタン	操作名	キーボード	内容
	続行	F5	プログラムを次のブレークポイントまで実行する
	一時停止	F6	実行中のプログラムを一時停止する
	ステップ オーバー	F10	次の処理に進む（処理が関数の場合，関数の中には入らない
	ステップ イン	F11	関数の中に入る（処理が関数ではない場合，次の処理に進む）
	ステップ アウト	Shift+F11	現在の関数から外に出る
	再起動	Ctrl+Shift+F5	デバッガを再起動する
	停止	Shift+F5	デバッガを停止して終了する

4-4. デバッグモードの使い方

VScode のエクスプローラーで「main.cpp」を開き，loop 関数内の適当な場所にブレークポイントを入れてみましょう。その場所でプログラムが止まります。

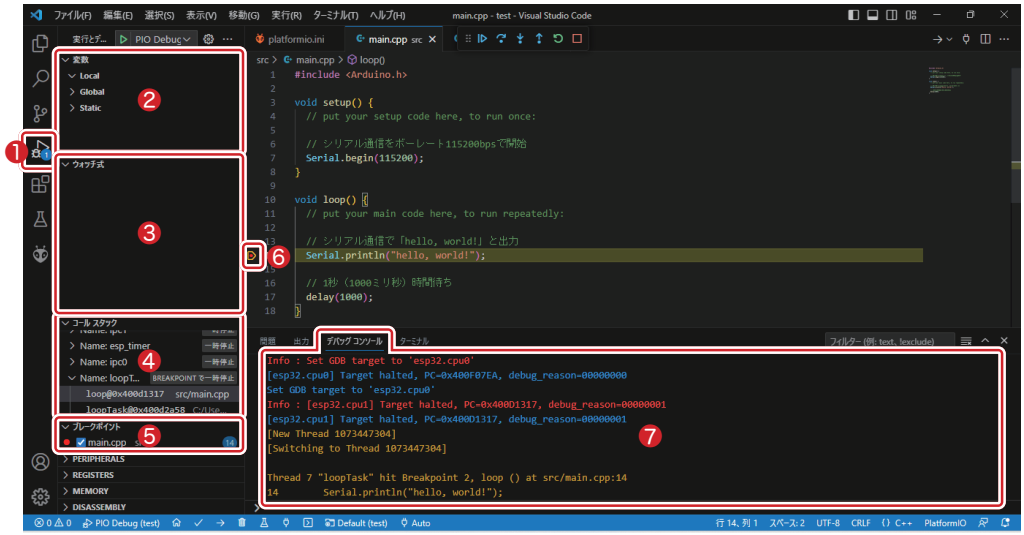


「続行」をクリックするとプログラムが動き出し，ブレークポイントの場所で再度止まります。



ブレークポイントを解除して「続行」すると，プログラムが連続して動くようになります。
デバッグを終了するには，デバッグツールバーの「停止」をクリックします。

■ デバッグモード画面各部名称



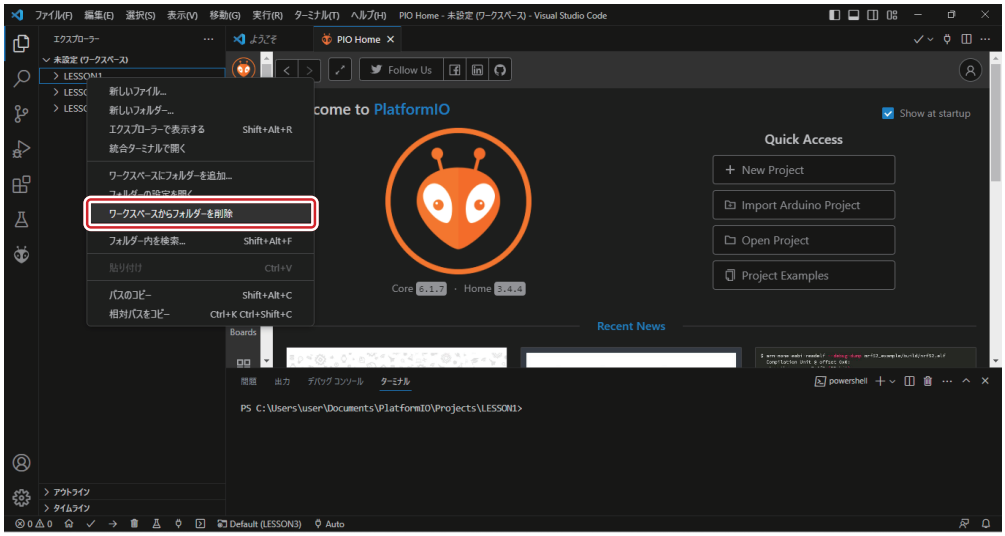
❶「実行とデバッグ」ボタン
❷ 変数　　：変数の値を確認できる
❸ ウォッチ式　　：任意の追加した式の評価結果を表示
❹ コールスタック　　：呼び出し経路を追跡できる
❺ ブレークポイント　　：ブレークポイントの設置リスト　クリックで設置点にジャンプ
❻ 実行ポイント　　：示された行で一時停止していることを表す
❼ デバッグコンソール　　：デバッグに関する情報を表示

05 PlatformIO のプロジェクト管理

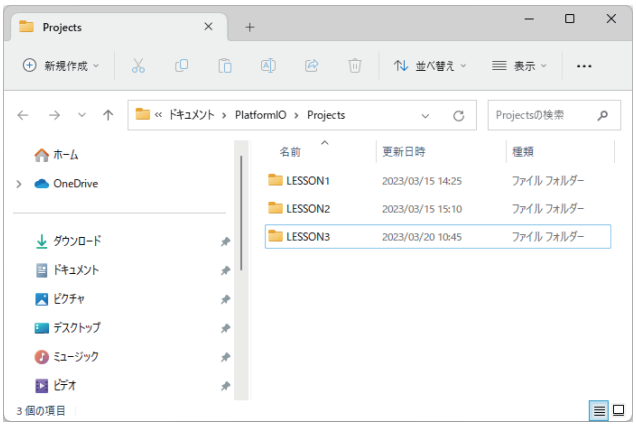
「プロジェクトの作成」については 2-1 で紹介しましたが，その他の操作も掲載しておきます。

■プロジェクトの削除

❶ VScode のワークスペースから削除したいプロジェクトを右クリックし，「ワークスペースからフォルダーを削除」をクリックする。

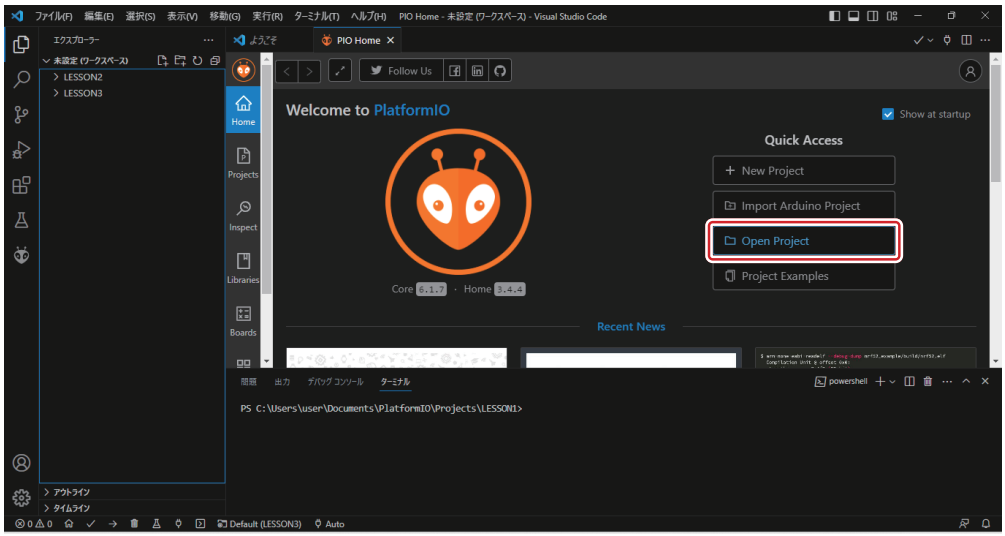


❷ ワークスペースから削除しても，元ファイルは削除されない。
元ファイルを削除したい場合は，VScode を終了し OS のエクスプローラーで削除する必要がある。

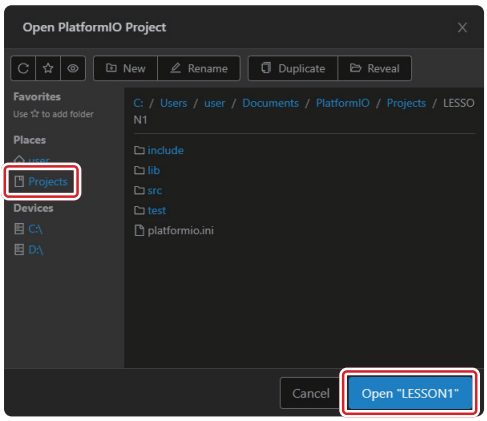


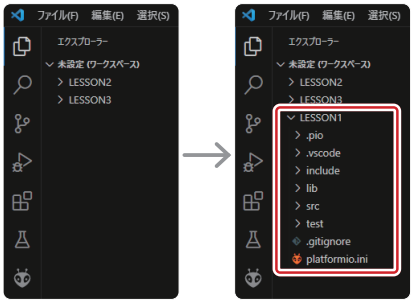
■プロジェクトの登録

❶ 元ファイルを残しておけば，ワークスペースへの再登録が可能。
PIO Home で「Open Project」をクリックする。



❷ Projects フォルダ内から登録したいプロジェクトのディレクトリを開き「Open プロジェクト名」ボタンをクリックすると，VScode のワークスペースにプロジェクトが現れる。

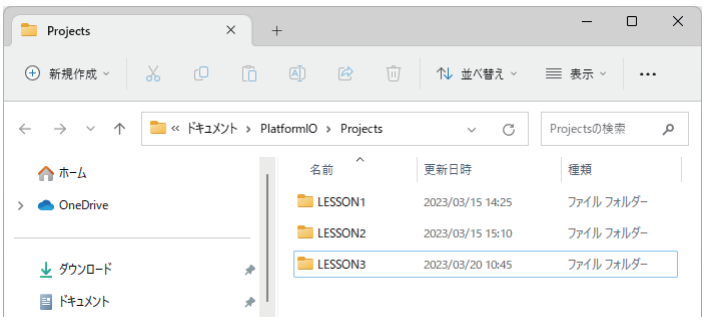




■プロジェクトの複製

❶ OS のエクスプローラーでプロジェクトフォルダを複製する。
フォルダ名がプロジェクト名となるので，命名ルールに従って変更する。

> ⚠ 使える文字は 半角英数字　と　_（アンダーバー）と　-（ハイフン）のみ
> 文字と小文字は区別されないので注意。



❷ 前ページの「プロジェクトの登録」の手順で VSCode に登録する。

C 言語で制御する
ESP32 マイコン入門　開発環境構築

２０２３年　10 月　１日　　初版　　発行

著　者　ESP32 マイコン実験キット研究会
発行者　答島　一成
発行所　株式会社アドウィン
　　広島市西区楠木町 3-10-13
　　TEL：082-537-2460（代表）
　　FAX：082-238-3920
　　E-mail：hanbai@adwin.com