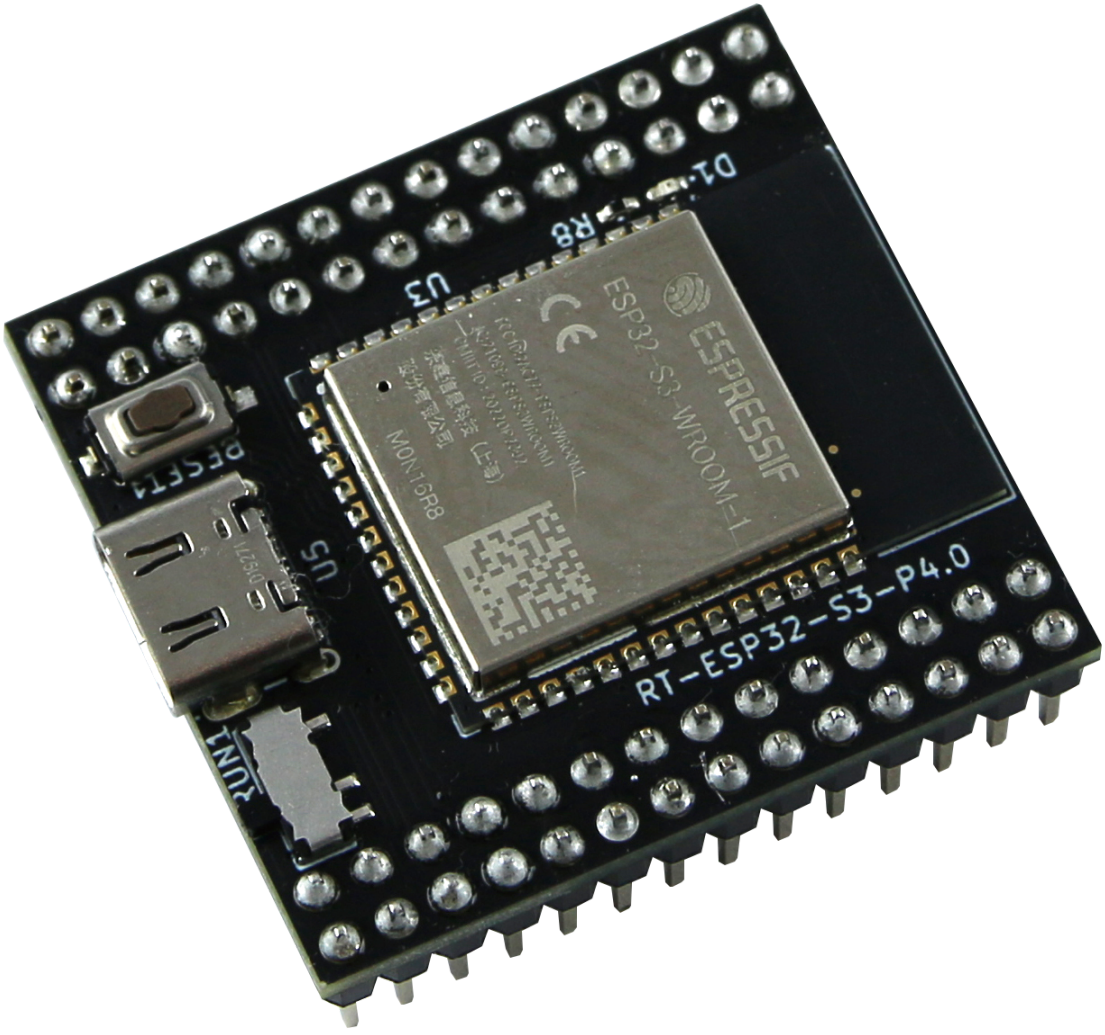
ESP32-S3マイコンボード
インストールマニュアル



1.0版

株式会社アールティ

RT CORPORATION

目次

目次 **1**
ご使用になる前に **2**
製品に対する注意事項 2
安全に関する注意事項 3
内容物一覧 4
主な仕様 **5**
本体の仕様 5
本製品の端子仕様 7
コンピュータの動作要件 8
組立 **9**
本製品を単体で使用する場合 9
環境設定 **10**
コマンドとArduinoスケッチの表記について 10
はじめに 11
Arduino IDEのインストール 11
Arduino IDEの初期設定 18
サンプルスケッチのビルドと書き込み 25
micro-ROSの環境設定 33
PCとPi:Co Classic3間の通信確認 38
使用しているツール、OSSのバージョン 40
製品保証 **41**
保証の内容 41
保証者の名称、所在地および電話番号 41
保証期間 41
保証の適用 41
保証の除外事項 42
保証の態様 42
お客様の費用負担 43
保証を受けるための手続き 43
お問い合わせ **44**
改訂履歴 **44**
Copyright・知的財産権について **44**

ご使用になる前に

　この度は、弊社の「ESP32-S3マイコンボード（以下「本製品」といいます）」をお買い上げいただき、誠にありがとうございます。本製品をご使用になる前に、本書をお読みいただきますようお願いいたします。

製品に対する注意事項

- 初めてロボットを使用される方は、経験者と一緒に作業することをお勧めします。
- 製品到着後、本書の内容物一覧と照らして必ず本体および付属品のご確認をお願いします。
- 製品の品質、検品および発送には万全を期していますが、万一お届けした製品に不良、破損（輸送中の本体破損も含みます）、付属品の不足がありましたら、製品到着後**7**日以内に弊社営業サポート（sales@rt-net.jp）までご連絡いただきますようお願いいたします。部品または本体の交換、不足品の発送等を行い、無償にて対応いたします。製品到着後8日以上が経過した場合、無償対応はいたしかねますのでご了承いただきますようお願いいたします。
- 製品の仕様および外観ならびにweb上で公開しているデータおよび情報は、改良のため予告なく変更することがあります。改良版は、ご購入時点の製品、データおよび情報と異なる可能性がありますが、異なる点について交換、返金、返品、改変等はいたしかねますのでご了承いただきますようお願いいたします。
- 本製品は、製造工程の特性により、フレームや部品に傷が付くことがあります。これらの傷は、本体の運転に支障を来さないため、保証の適用外とさせていただきます。
- 本書は、Windows、macOSまたはLinuxの基本的な使用方法を理解している方を対象として構成されています。
- 本書は、本製品を対象としたmicro-ROSサンプルソフトウェアについて解説いたしません。
- 本書において記載されております画像、その他記載されている会社名、製品名等の固有名詞は各社の登録商標または商標です。なお、本文中では、TM、(R)マークは省略しております。

安全に関する注意事項

警告表示について

マーク	マークの定義
⚠ 危険	「**危険**」を表します。 取り扱いを誤った場合に、死亡または重傷を負う状態が生じることが想定され、かつ火災発生等の財物に重大な損害が生じる緊急性の高い事項を表します。
⚠ 注意	「**注意**」を表します。 取り扱いを誤った場合に、軽傷または物理的損害が生じることが想定される事項を表します。

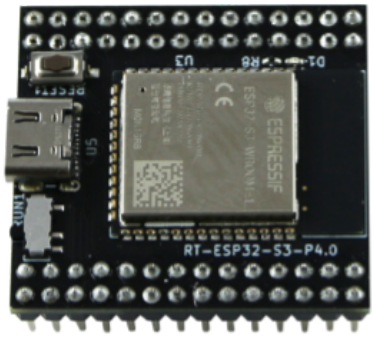
危険内容および対応方法

	危険内容（行為、現象）	マーク	対応方法
作業時	Pi:Co Classic3に本製品を搭載して走行した時、周辺の金属に触れてショートする。または、本製品の操作時に金属物(指輪やネックレス等)が触れてショートする。	⚠ 危険	運用時、周辺に金属物がないことを確認してください。また、操作時に金属製の指輪等を外した状態で操作してください。
組立時	Pi:Co Classic3にバッテリを繋げた状態で本製品を交換する。	⚠ 危険	Pi:Co Classic3の電源スイッチをOFFにし、バッテリを外した状態で本製品を交換してください。
組立時	本製品のJP1パッドをショートした状態で本製品をPi:Co Classic3に取り付け、Pi:Co Classic3の回路が破壊される。	⚠ 注意	JP1パッドをショートした場合は、本製品をPi:Co Classic3に取り付けないでください。

内容物一覧

本製品セットに含まれている物品は次のとおりです。

番号	物品	数量
1	ESP32-S3マイコンボード	1
2	USB Type-A - Type-C ケーブル	1
3	技適マークシール	1
4	入門テキストのダウンロードカード **※テキスト同梱版（RT-ESP32-S3-P3.0）のみに付属します**	1



①ESP32-S3マイコンボード

②USB Type-A - Type-C ケーブル

③技適マークシール

④ダウンロードカード

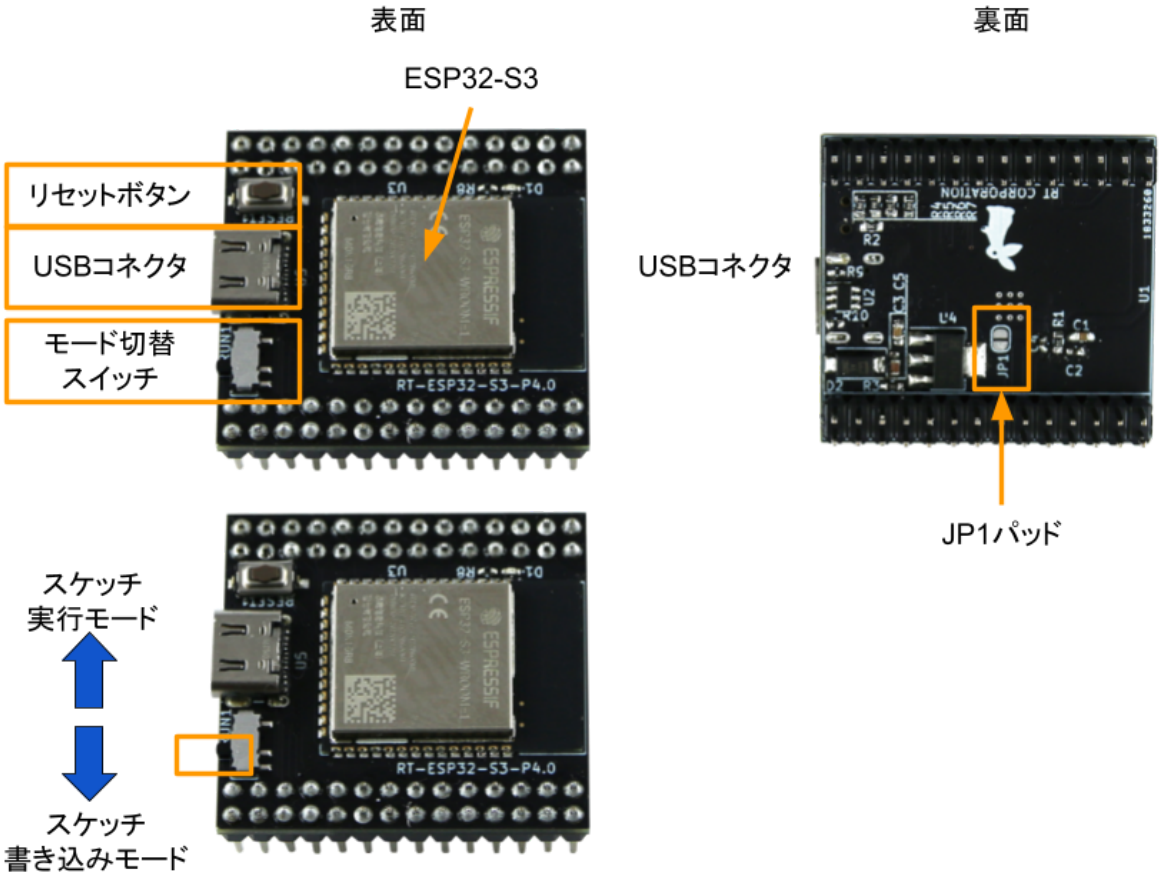
主な仕様

本体の仕様

本体の仕様は次のとおりです。

本製品の本体仕様一覧

項目	仕様
製品名	ESP32-S3マイコンボード
型番	RT-ESP32-S3-P3.0、RT-ESP32-S3-P3.0_NT
入力電源（定格電圧）	ピンヘッダからの供給：3.3V±10% 200mA以上 USBコネクタからの供給：5V±10%
CPU	Espressif Systems ESP32-S3-WROOM-1-N16R8
動作周波数	240MHz
ROM	384kByte
SRAM	512kByte
SPI Flashメモリ	16MByte
PSRAM	8MByte
書き込み	USB
寸法	34x37x14mm
重量	10g
モード切替スイッチ	ESP32-S3の起動モードを切り替えるスイッチです。RUN1のシルク側にスライドするとスケッチ実行モードで、反対側にスライドするとスケッチ書き込みモードで起動します。
リセットボタン	ESP32-S3のプログラムを再起動するボタンです。モード切替スイッチ操作後にこのボタンを押すと、切り替え後のモードで起動します。Pi:Co Classic3の電源をOFF->ONしてもプログラムを再起動できます。
JP1パッド	本製品をUSB電源で駆動する際に使用します。 JP1パッドをショートするとUSB電源から生成した3.3VをESP32-S3に供給します。 **JP1パッドをショートした後は、本製品をPi:Co Classic3に取り付けないでください。**


表面
裏面
ESP32-S3
リセットボタン
USBコネクタ
モード切替
スイッチ
USBコネクタ
JP1パッド
スケッチ
実行モード
スケッチ
書き込みモード
RT-ESP32-S3-P4.0

本製品の外観

本製品の端子仕様

ESP32-S3の端子仕様に対して本製品で追加した機能の注意点を記載します。下記の接続先一覧表を参照してください。接続先の括弧内にESP32-S3のピン名を記載しています。NCはNo Connectの略称です。

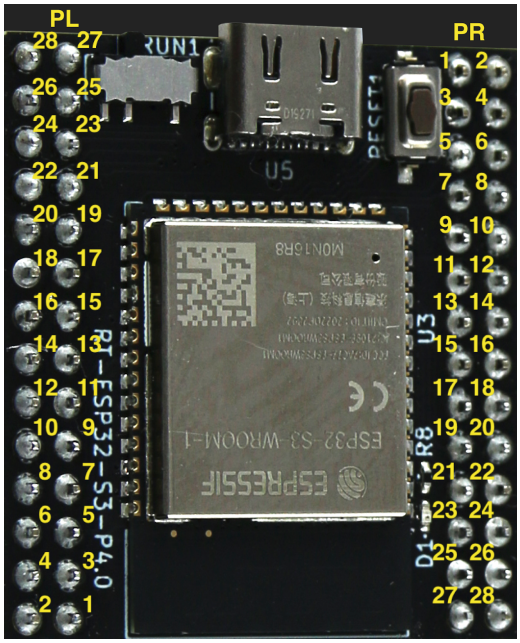
IO19（PR12）、IO20（PR9）はUSBにつながっています。USBを使わないときにIO19とIO20を使用できます。

IO0（PR6）は、書き込みモード、ユーザブートモードの切り替えピンです。ブート時にGNDと接続されるため、GPIOとして使用することは推奨しません。

IO45（PL15）とIO46（PR7）はGNDに対して10kΩでプルダウンしてあります。IO46はPi:Co Classic3の機能と接続していないためGPIOとして使用できます。GPIOとして使用する際はアクティブHIGHの出力として使用することを推奨します。

IO4（PL22）、IO5（PL23）、IO6（PL24）、IO7（PL25）は3.3Vに対して1MΩでプルアップしてあります。

ピンからPi:Co Classic3への接続先一覧（括弧内にはESP32-S3のピン名を記載）

外側のピン	内側のピン		内側のピン	外側のピン
NC	NC		SW_R(IO12)	SW_C(IO11)
NC	AD4(IO7)		SW_L(IO10)	MOTOR_EN(IO9)
AD3(IO6)	AD2(IO5)		NC	MD(IO0)
AD1(IO4)	AD0(IO8)		(IO46)	NC
(IO48)	GND		(IO20,USB D+)	(IO3)
(IO47)	NC		RESET(EN)	(IO19,USB D-)
NC	PWM_L(IO45)		NC	NC
NC	CW_L(IO21)		NC	RXD0
CW_R(IO14)	PWM_R(IO13)		TXD0	NC
AD0(IO8)	buzzer(IO38)		SLED_L(IO17)	SLED_R(IO18)
BLED1(IO39)	BLED0(IO40)		SLED_FL(IO15)	SLED_FR(IO16)
LED3(IO41)	LED2(IO42)		NC	NC
LED1(IO2)	LED0(IO1)		NC	NC
3.3V	3.3V		NC	NC

コンピュータの動作要件

本製品を使用するに当たり、下記の要件を満たすPCをご用意ください。

Linux

項目	仕様
OS	Ubuntu 22.04（推奨）
CPU	1GHz デュアルコア以上
メモリ	4GB以上
ストレージ	128GB以上
接続	USB、Wi-Fi

Windows

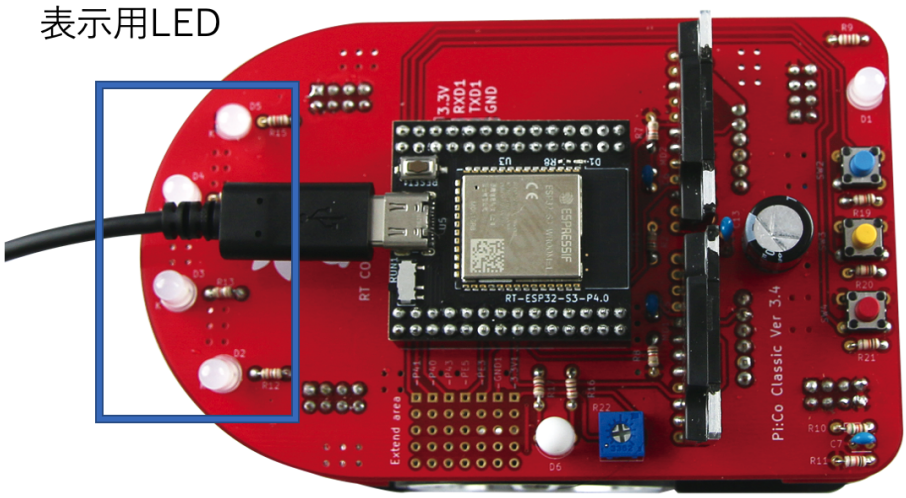
項目	仕様
OS	Windows 11
CPU	1GHz デュアルコア以上
メモリ	4GB以上
ストレージ	128GB以上
接続	USB、Wi-Fi

macOS

項目	仕様
OS	Monterey 12.6　or Ventura 13.0
CPU	Intel Core i5 1.6GHz以上
メモリ	4GB以上
ストレージ	128GB以上
接続	USB、Wi-Fi

組立

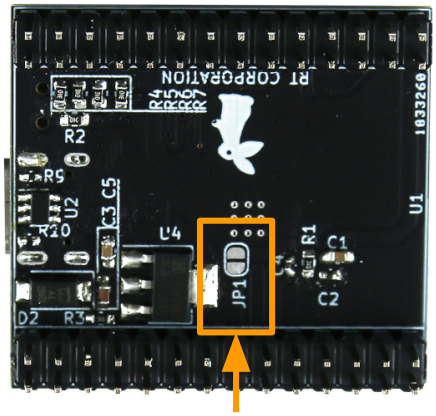
Pi:Co Classic3に取り付けられているCPU基板（RX631搭載CPUボード）を取り外して、本製品に置き換えます。このとき、下記写真のとおり本製品のUSBコネクタが表示用LEDの方に向くように本製品のピンをソケットに挿してください。取り付け後、ピンが曲がっていないか、ずれていないか確認してください。



本製品をPi:Co Classic3に正しく取り付けた状態

本製品を単体で使用する場合

本製品は、製品裏面にあるJP1パッドをショートすることで、Pi:Co Classic3から取り外して開発することが可能です。この場合、USBのバスパワーで本製品のCPUを駆動できます。ただし、**JP1パッドをショートした後は、本製品をPi:Co Classic3に取り付けないでください。**Pi:Co Classic3の電源回路を破壊する恐れがあります。



JP1パッド

製品出荷時のJP1パッドの状態

環境設定

ここでは本製品の開発環境の構築手順について説明します。

コマンドとArduinoスケッチの表記について

本書では、Ubuntuのターミナルで実行するコマンドを下記のように表記します。”Unset”より下の行がコマンドを表します。ただし、”#”で始まる行はコメントを表すため実行不要です。

```Unset
$ echo "コマンド"
# この行はコメントです
```

Arduinoスケッチは下記のように表記します。”C/C++”より下の行がスケッチの内容です。

```C/C++
void setup(){
 // ピンモードを出力に設定します
 pinMode(LED0, OUTPUT);
}

void loop(){
 // LEDを点滅させます
 digitalWrite(LED0, HIGH);
 delay(500);
 digitalWrite(LED0, LOW);
 delay(500);
}
```

はじめに

本製品はArduino Foundationが提供するArduino IDEを開発環境として使用します。Arduino IDEはマルチプラットフォームに対応しています。そのため、本製品は3種のOS（Windows、macOS、Linux）で開発できます。

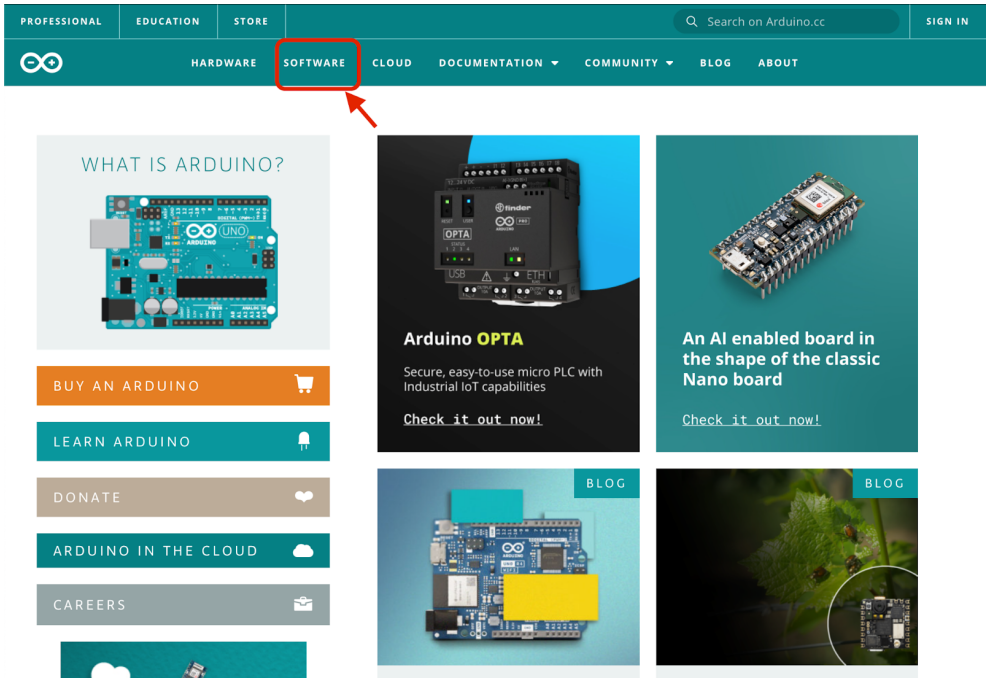
また、本製品はROS 2（Robot Operating System 2）というオープンソースソフトウェアにも対応しています。ROS 2を用いることで他の開発者が提供するソフトウェアを活用でき、3Dビジュアライザや、ゲームパッドによる遠隔操作等の機能をPi:Co Classic3に取り入れられます。本製品はmicro-ROSというマイコン向けのROS 2ソフトウェアを活用し、Pi:Co Classic3とPC間でのデータ通信を実現しています。

本章の前半ではArduino IDEのインストール手順を説明し、後半ではROS 2を使用する場合の開発環境の構築手順を説明します。ROS 2を使用する場合はLinux OSの一つである**Ubuntu**で開発することを推奨します。

Arduino IDEのインストール

2023年4月時点で最新版のArduino IDE 2.1.0をインストールする手順を示します。Arduino IDEが更新された場合は、本書内のバージョン番号を適宜読み替えてください。

webブラウザでhttps://www.arduino.ccにアクセスしSOFTWAREのタブをクリックします。

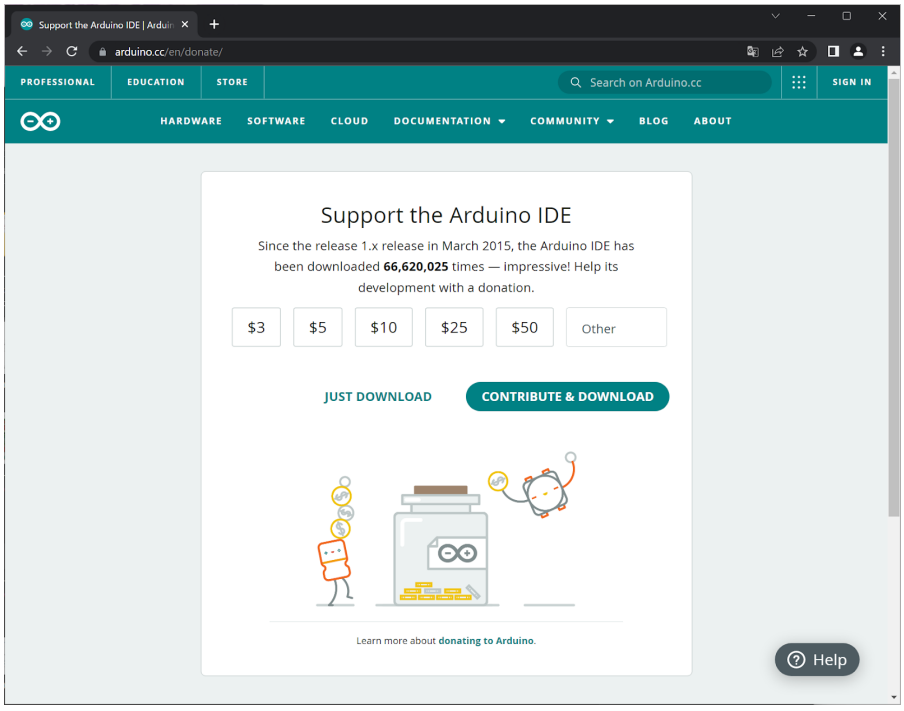


https://www.arduino.cc のトップページ

使用するOSに合うソフトウェアをDOWNLOAD OPTIONSからクリックします。本書では、Windows MSI installer、macOS Intel, 10.14: “Mojave” or newer, 64 bits、およびLinux AppImage 64 bits (X86-64)を選択した前提でインストール手順を説明します。

Arduino IDE 選択ページ

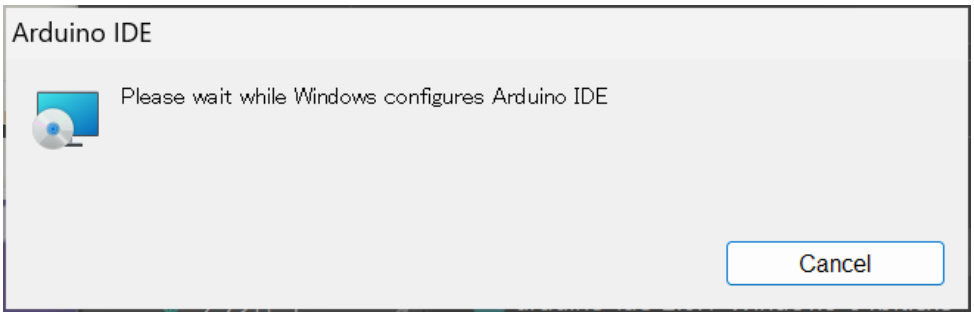
以下のSupport the Arduino IDEページが表示された場合は、寄付をしてソフトウェアをダウンロードする”CONTRIBUTE & DOWNLOAD”と、ダウンロードのみを実施する”JUST DOWNLOAD”の**どちらかを選択します。**



Arduino IDE ダウンロード方法選択ページ

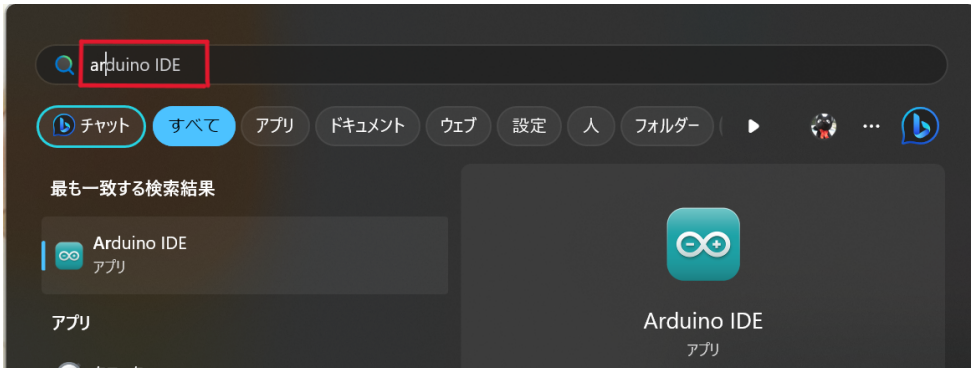
Windowsの場合

　ダウンロードしたMSI installer（例：arduino-ide_2.1.0_Windows_64bit_msi）をダブルクリックしてインストールします。インストールが終わると下記の画面が消えます。



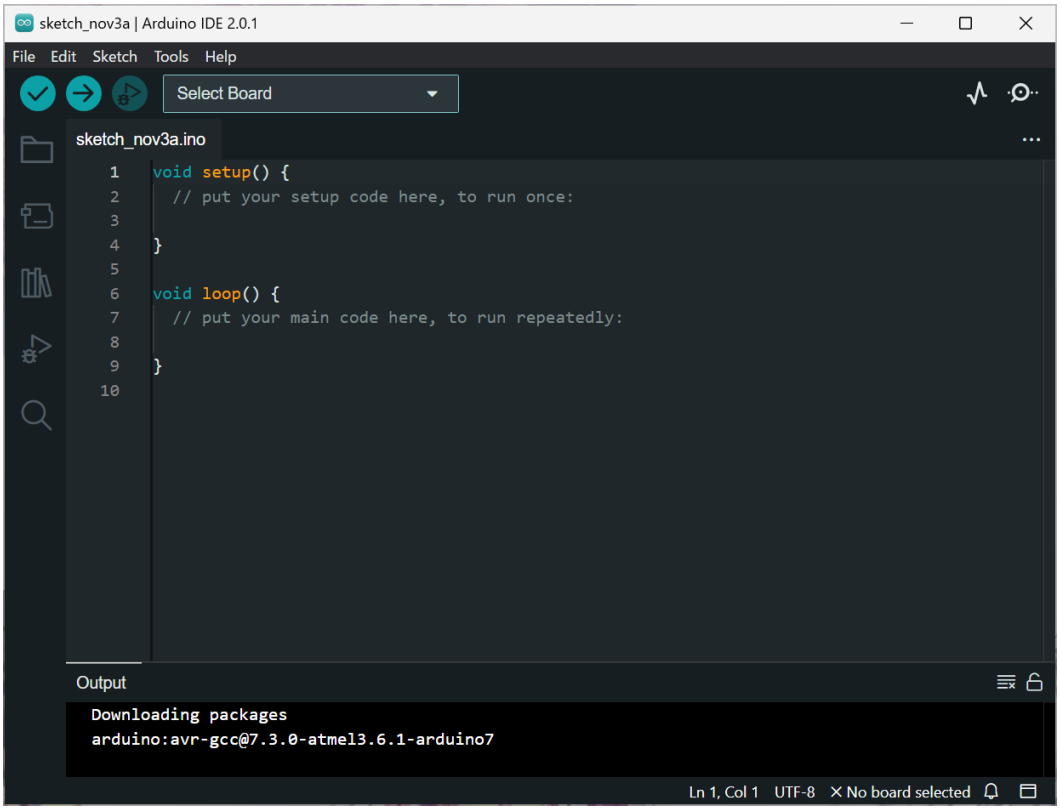
インストール中の画面

　Arduino IDEが正常にインストールできたことを確認するためArduino IDEを起動します。Windowsのスタートボタン->検索ボックスに”arduino”というキーワードを入力するとArduino IDEが検索結果として表示されます。



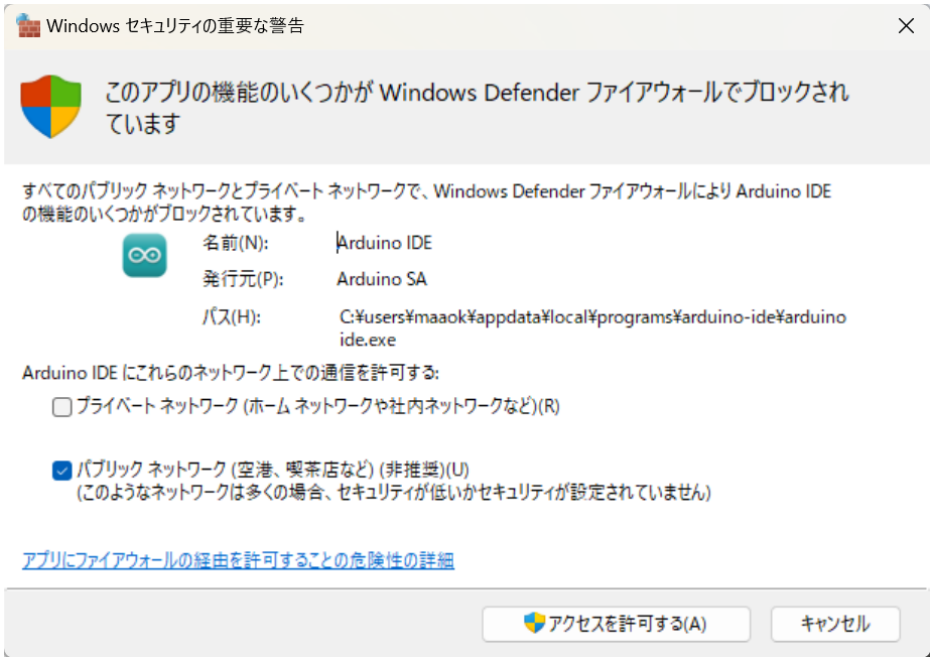
検索ボックスに”arduino”を入力した状態

表示されたArduino IDEをクリックし、Arduino IDEが起動すると以下のような画面が表示されます。以上でインストール完了です。



起動したArduino IDE

Windowsセキュリティの重要な警告メッセージが表示されたときは、アクセスを許可してください。アクセスをブロックしてしまうとArduino IDEのバージョンアップや追加ライブラリのダウンロードができなくなります。



Windowsセキュリティの重要な警告メッセージ

macOSの場合

　ダウンロードしたファイル（例：arduino-ide_2.1.0_macOS_64bit.dmg）をダブルクリックすると以下の画面が表示されます。画面左にあるArduino IDEのアイコンを右のフォルダに移動するとインストールが完了します。インストール後のArduino IDEはLaunchpadから起動できます。

dmgファイルを開いた状態

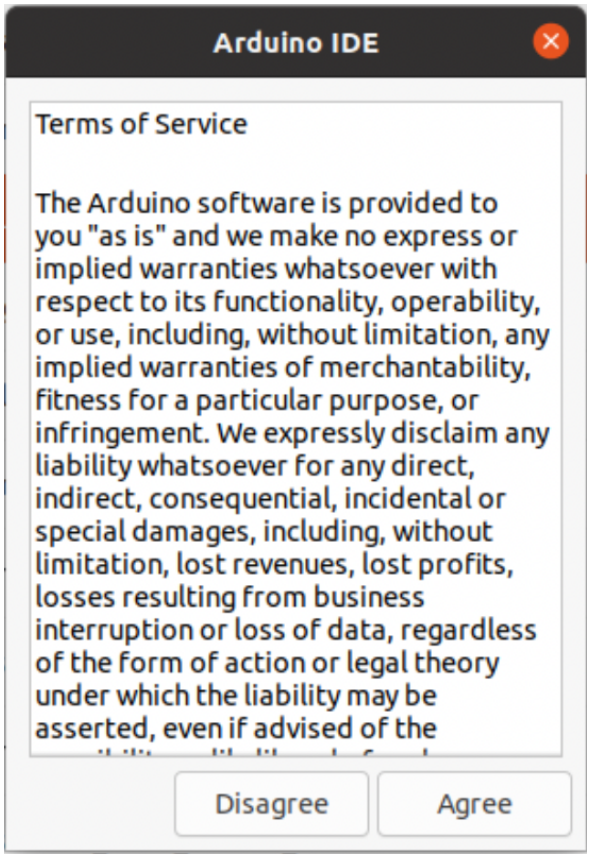
Linux(Ubuntu)の場合

　ダウンロードしたファイル（例：arduino-ide_2.1.0_Linux_64bit.AppImage）を右クリックし、メニューからプロパティを選択します。プロパティ内のアクセス権タブを開き、”プログラムとして実行可能”にチェックを入れます。

AppImageファイルを右クリックした状態

プロパティのアクセス権タブを開いた状態

その後、AppImageファイルをダブルクリックします。Terms of Serviceが表示された場合は、中身を確認し”Disagree/Agree” を選択してください。以上でインストール完了です。



Terms of Service画面

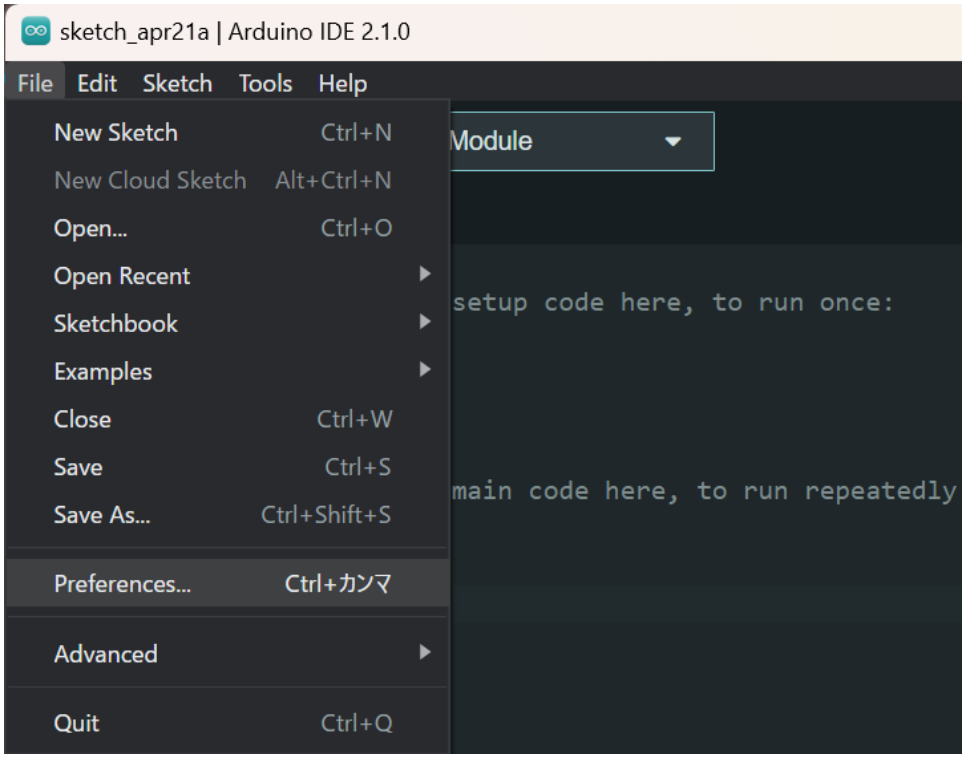
AppImageファイルをダブルクリックしてもArduino IDEが実行されない場合は、AppImageファイルのコンテンツを展開し実行するツール（FUSE）がインストールされていない可能性があります。https://github.com/AppImage/AppImageKit/wiki/FUSE 参照し、FUSEをインストールしてください。

Arduino IDEの初期設定

　ここではArduino IDEの言語や外観を設定します。また、本製品に搭載されているESP32-S3用のスケッチファイルをビルドするために、ライブラリやコンパイラを追加します。

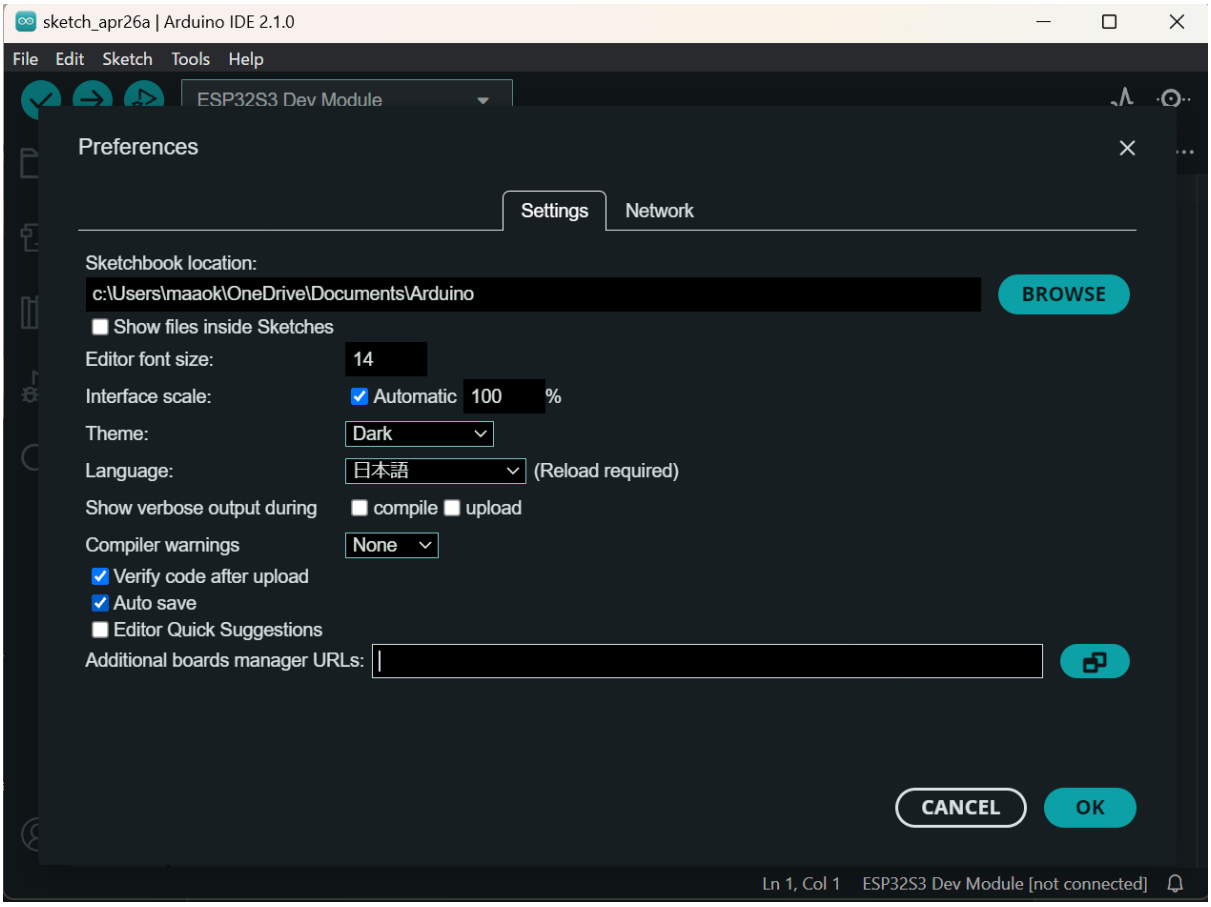
言語設定

　Arduino IDEの言語を日本語に設定します。メニューバーにあるFile->Preferences…を選択します。



メニューバーのFile->Preferences…を選択した状態

Languageを”English”から”日本語”に変更して右下にある “OK” をクリックします。



Languageを日本語に設定した状態

”OK”をクリックしたら自動的にArduino IDEが再起動します。再起動後、メニューバーの左端にあるボタンが”File”から”ファイル”になっていれば言語設定完了です。

外観設定

　OSの設定によっては、Arduino IDEの外観が暗い色調で表示されます。コントラストの弱いPC画面では明るい色調に設定するとスケッチ編集画面のカーソルをはっきり表示できる場合があります。下記手順に従ってお好みの外観を設定してください。

　メニューバーのファイル->基本設定...を選択します。”配色テーマ”を”Dark”から”Light”に変更し、”OK”をクリックします。これによりArduino IDEの外観が明るい色調に変更されます。以降のページで登場するArduino IDE画面は”Light”テーマを設定したものです。

配色テーマでLightを選択した状態

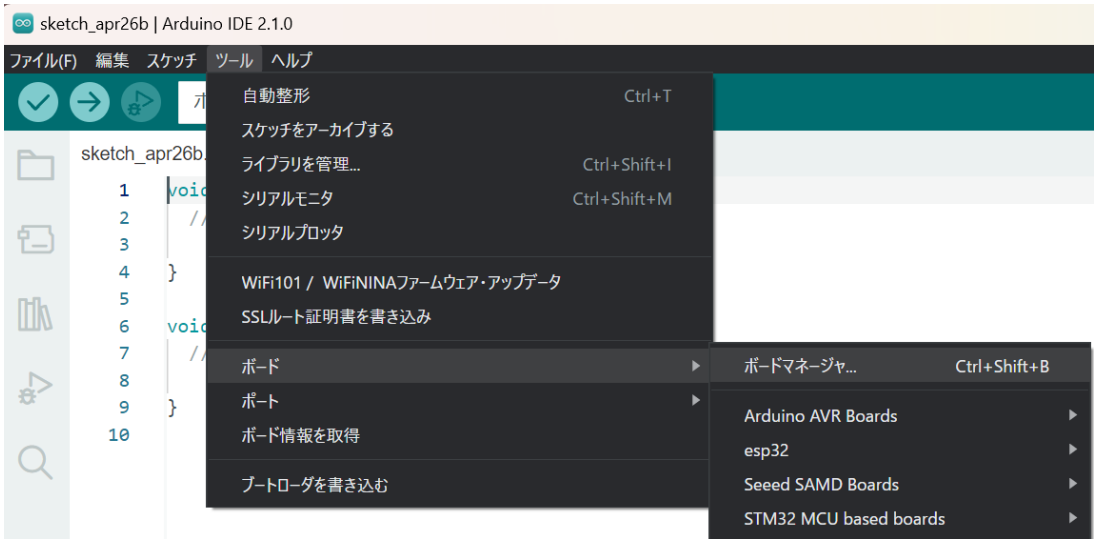
ESP32ボード情報の追加

スケッチを本製品に書き込めるように、ESP32のボード情報をArduino IDEに追加します。メニューバーのファイル->基本設定...を選択します。画面下部の”追加のボードマネージャのURL”に

https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json

を追加し”OK”をクリックします。

URLを追加したら、ESP32ライブラリやツールをダウンロードしてインストールします。メニューバーのツール->ボード->ボードマネージャ...を選択します。



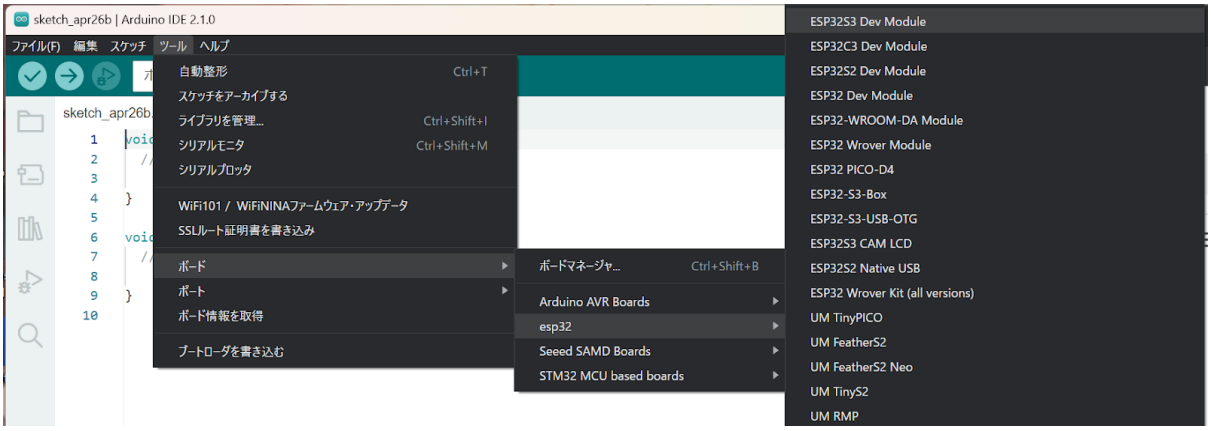
ボードマネージャ...を選択した状態

左側にあるボードマネージャの検索欄に”esp32”と入力すると”esp32 by Espressif Systems”が表示されます。表示されているバージョンが**2.0.2**以上であることを確認してください。この後インストールするmicro-ROS for Arduinoライブラリがバージョン2.0.2に対応しているためです。

バージョンが2.0.2以上であることを確認したら”インストール” をクリックします。2023年4月時点では2.0.8が最新です。

ボードマネージャ画面の検索欄に”esp32”を入力した状態

Arduino IDEはESP32以外のマイコンにも対応しているため、スケッチを書き込む際はESP32のボードを選択します。メニューバーのツール->ボード->esp32->ESP32S3 Dev Moduleを選択します。”ボード”と”ポート”が同じような文字で上下にあるため間違えないように選択してください。



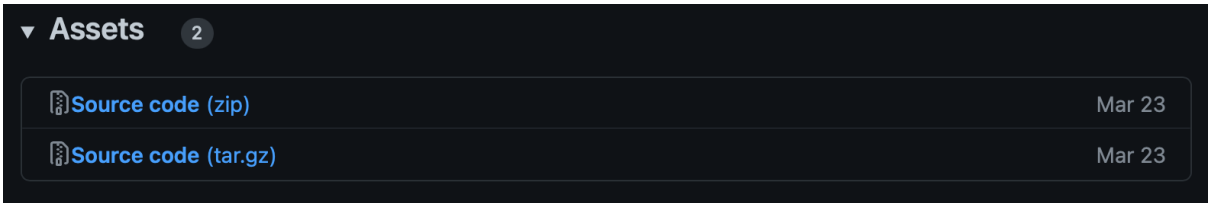
ツール->ボード->esp32->ESP32S3 Dev Modulesを選択

以上で、ESP32-S3のビルド環境設定完了です。

Arduino用のmmicro-ROSのライブラリをインストール

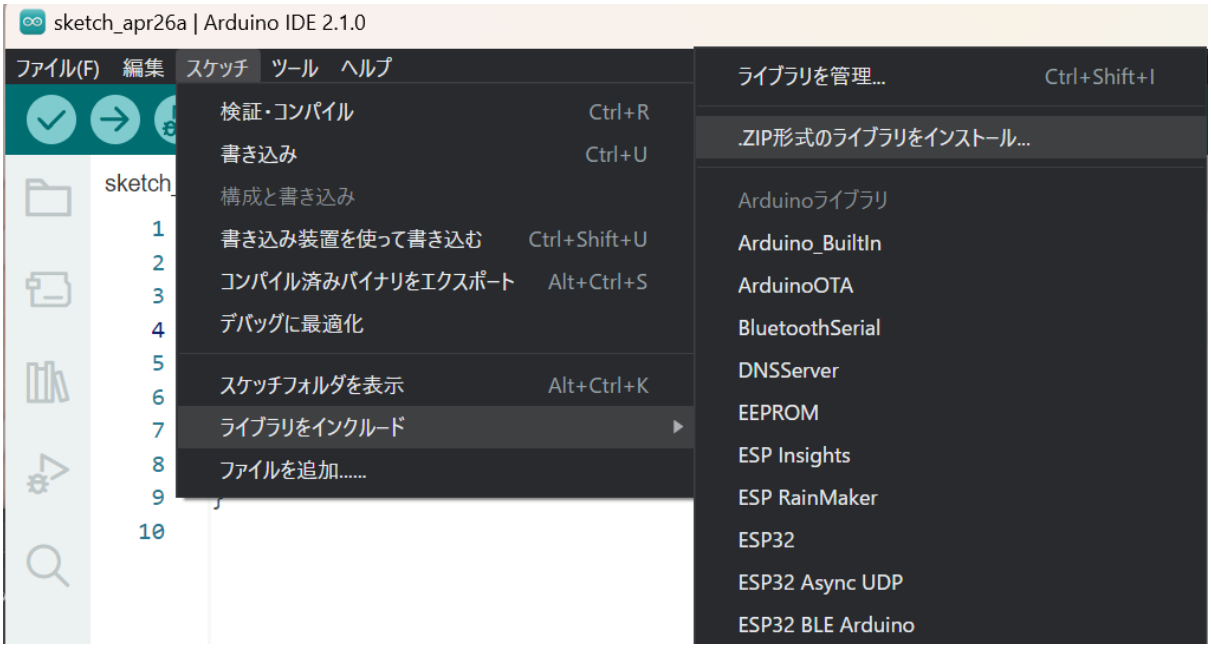
ここでは、Arduino向けのmmicro-ROSライブラリである**micro-ROS for Arduino**のインストール方法を説明します。

webブラウザでhttps://github.com/micro-ROS/micro_ros_arduino/releases にアクセスし、最新のライブラリを選択します。2023年4月時点ではv2.0.5-humbleが最新です。Assetsの中からsource code(zip)をダウンロードします。ダウンロードしたzipファイルは**展開不要です**。



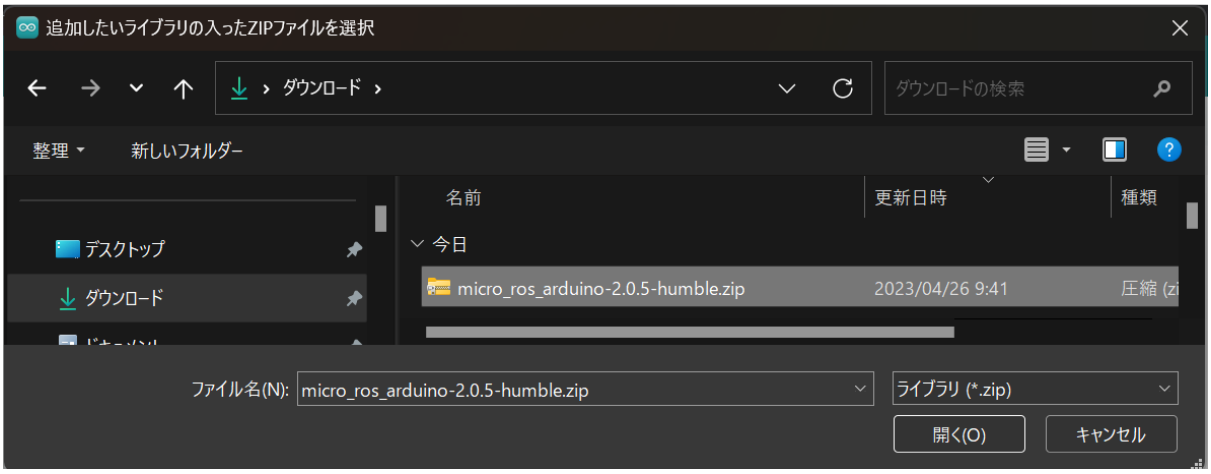
Source code (zip)をダウンロード

Arduino IDEを起動し、メニューバーのスケッチ->ライブラリをインクルード->".ZIP形式のライブラリをインストール..."を選択します。



スケッチ->ライブラリをインクルード->.ZIP形式のライブラリをインストール...を選択

先ほどダウンロードしたファイルを選択し、”開く”をクリックします。

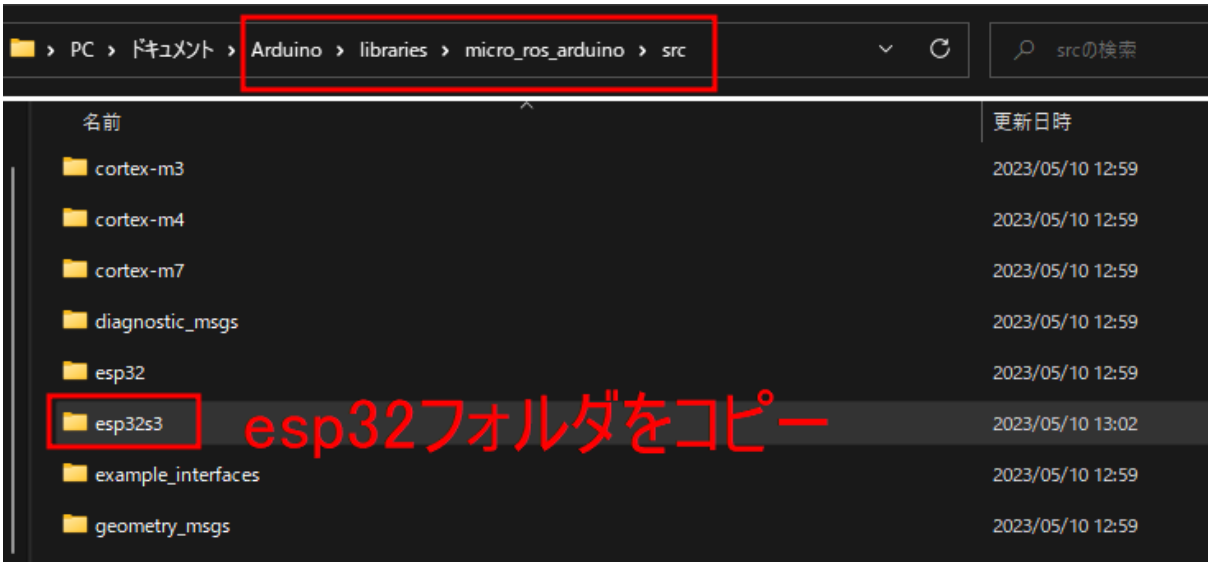


ダウンロードしたzipファイルを選択

インストールしたライブラリはESP32には対応していますがESP32-S3のビルド環境に対応していません。そのため、ライブラリ内のファイルを加工して対応します。

先ほどインストールしたライブラリは“スケッチブックの場所”¥libraries¥micro_ros_arduino”にあります。スケッチブックの場所はメニューバーのファイル->基本設定...で確認できます。

”スケッチブックの場所”¥libraries¥micro_ros_arduino¥srcにある”esp32”フォルダをコピーし、フォルダ名を”esp32s3”に変更します。”src”フォルダ内に”esp32”と”esp32s3”の2つのフォルダがあれば作業完了です。

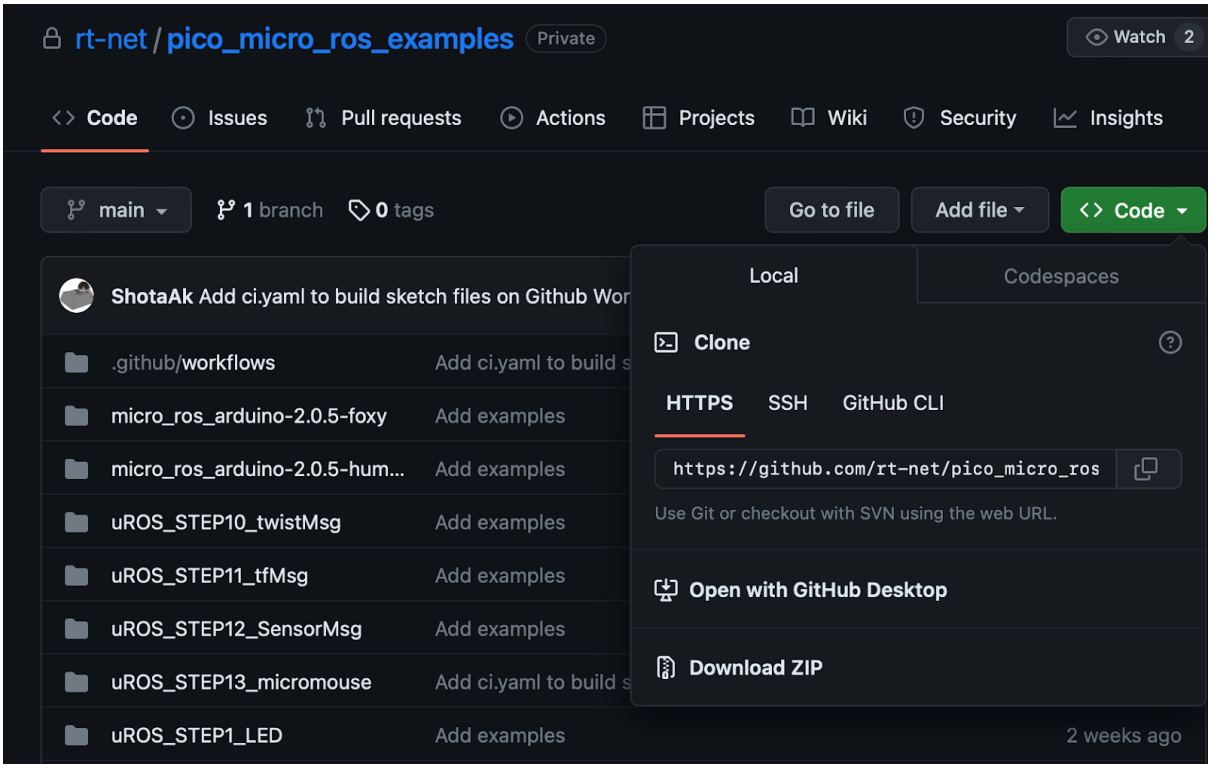


esp32フォルダをesp32s3としてコピー

サンプルスケッチのビルドと書き込み

webブラウザでhttps://github.com/rt-net/pico_micro_ros_arduino_examples にアクセスし、micro-ROS用のPi:Co Classic3のサンプルスケッチファイルをダウンロードします。

ページ内の”Code”から”Download ZIP”をクリックすることでダウンロードできます。ダウンロードしたzipファイルは**展開してください**。

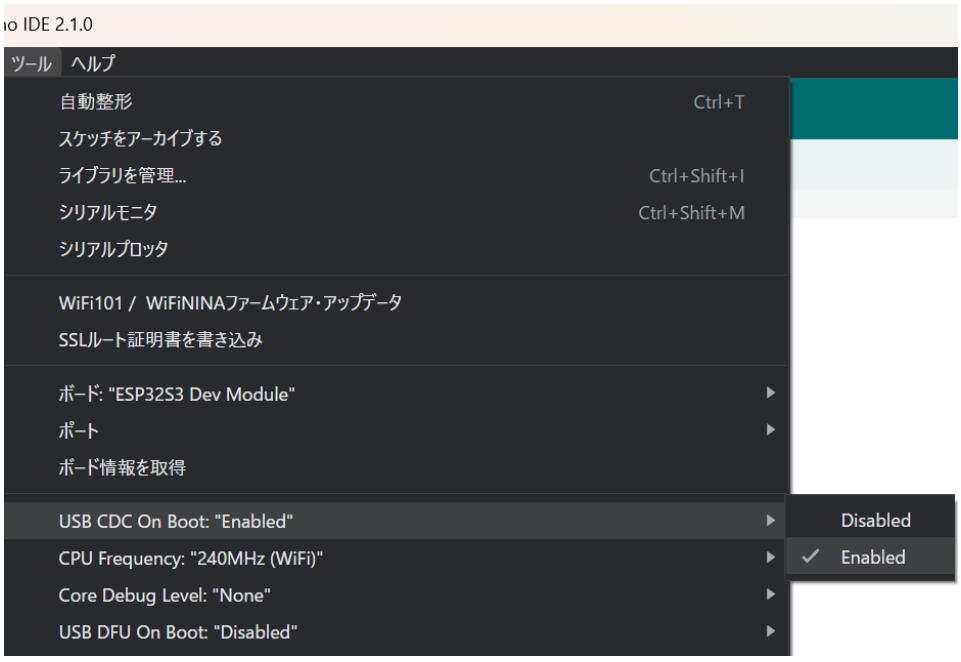


Code->Download ZIPを選択してサンプルスケッチをダウンロード

ここではPi:Co Classic3のLEDを点灯させるサンプルスケッチ（uROS_STEP1_LED）を例にとってスケッチのビルドと書き込み方法を説明します。

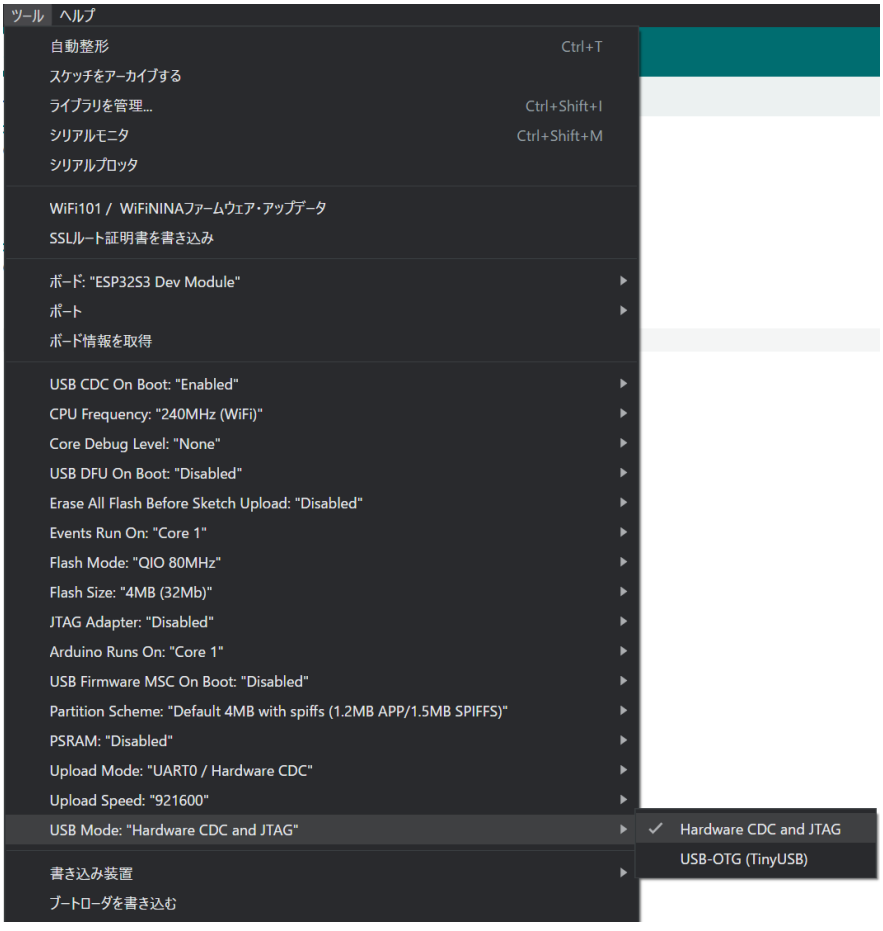
本製品にはUSBを使ってスケッチを書き込みます。デフォルトではUSBからの書き込みができないため、書き込みができるようにArduino IDEのオプションを2箇所変更します。

まず、メニューバーのツール->ボード->esp32->ESP32S3 Dev Moduleを選択します。次に、メニューバーのツール->USB CDC On Bootを開き、”Enabled”にします。



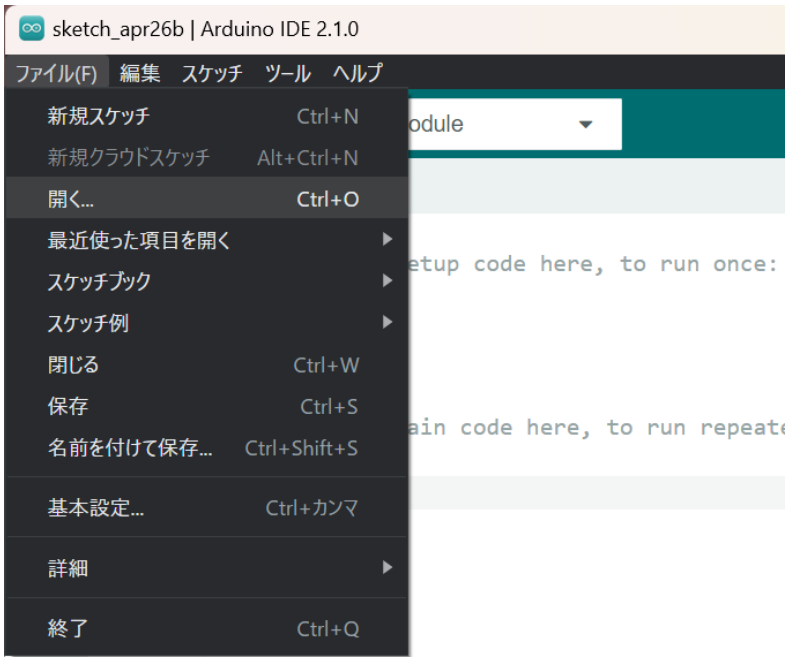
USB CDC On BootをEnabledに設定

メニューバーのツール->USB Modeを開き、”Hardware CDC and JTAG”にします。



USB ModeをHardware CDC and JTAGに設定

ダウンロードしたサンプルスケッチ内のuROS_STEP1/uROS_STEP1.inoファイルをダブルクリックで開きます。または、Arduino IDEを起動し、メニューバーのファイル->開く...をクリックしてuROS_STEP1.inoを開きます。



ファイル->開く...を選択する

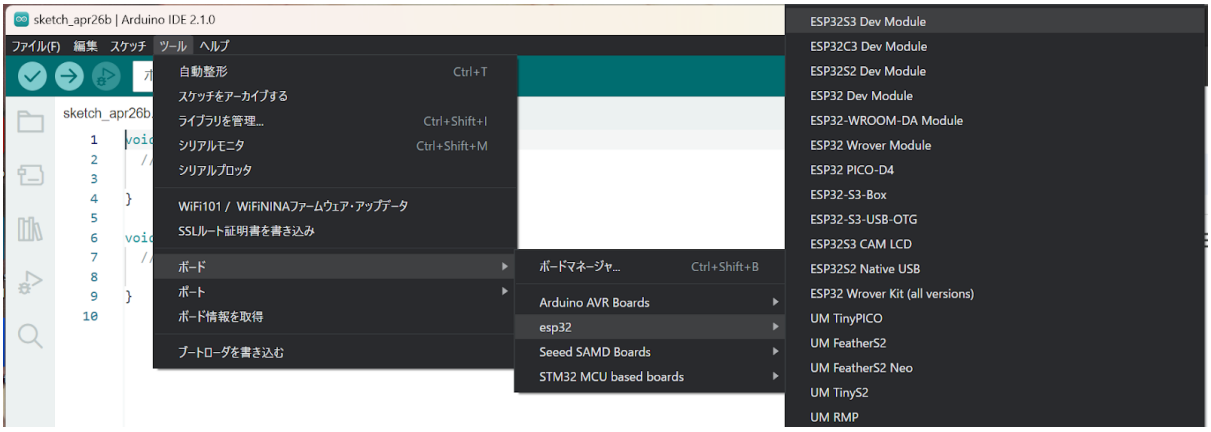
スケッチファイルを開く時に以下のメッセージが出た場合は、スケッチファイル名とフォルダ名が一致していない、またはフォルダが無いため新しくフォルダを作って良いか、ということを確認しています。Arduino IDEの仕様上、メインとなる”ファイル名”と”フォルダ名”が同じでないときにこのメッセージが表示されます。ファイル名やフォルダ名を変更する場合はそれぞれ同じ名前にしてください。

ファイル名とフォルダ名が一致しない場合の確認画面

サンプルスケッチを開くと以下のような画面が表示されます。

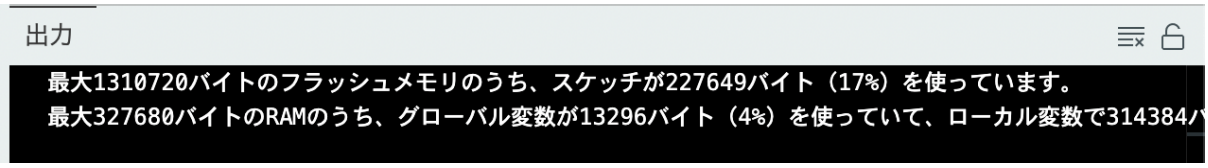
uROS_STEP1_LED.inoを開いた状態

ボードが”ESP32S3 Dev Module”になっていないときは、メニューバーのツール->ボード->esp32->ESP32S3 dev Moduleを選択してください。



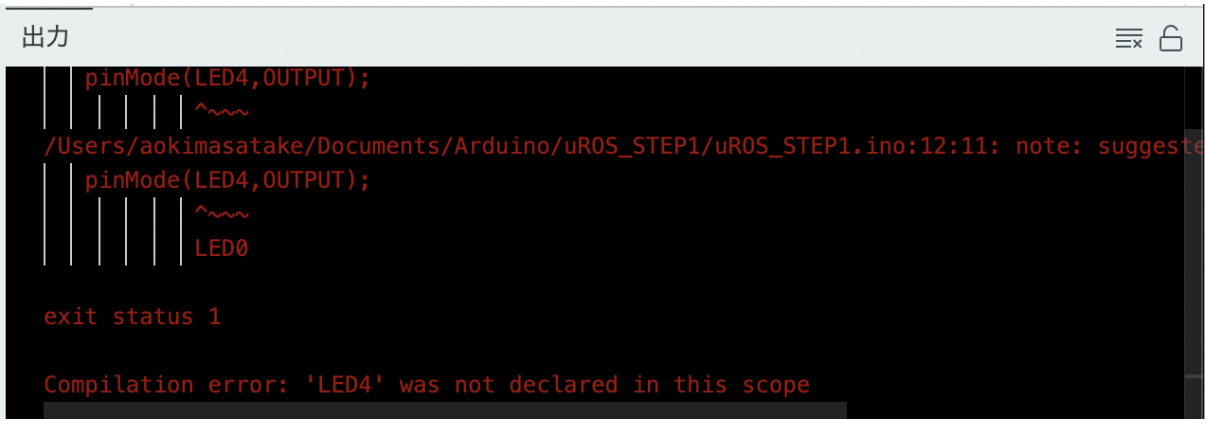
ツール->ボード->esp32->ESP32S3 dev Moduleを選択

左上にあるアイコン  で”検証”を実行しスケッチをビルドします。スケッチの記述に問題がなければ以下のようなメッセージが表示されます。



検証に問題が無いときのメッセージ

スケッチの記述にエラーがあった場合は、赤文字でエラー箇所が表示されます。



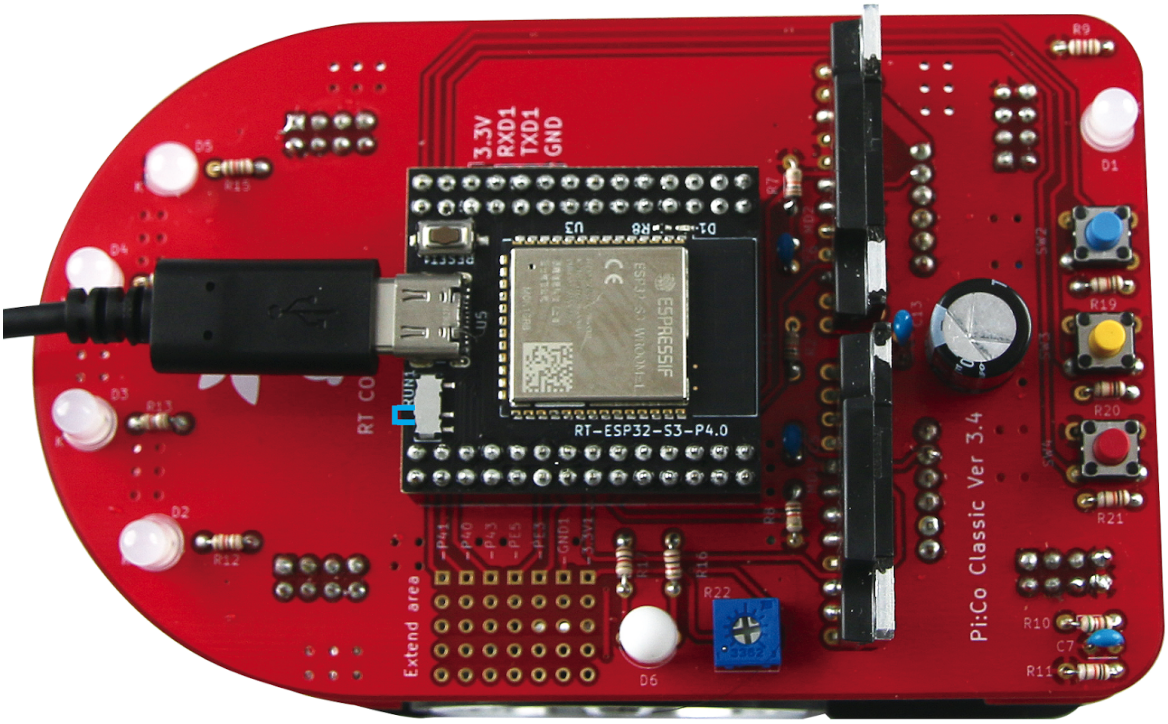
検証エラー時のメッセージ

Arduino IDEの右下にあるアイコン  をクリックすればメッセージを非表示にできます。もう一度クリックすると表示されます。画面を大きくしてスケッチ全体を少しでも多く見たい時や、このメッセージ画面が不要なとき等に活用してください。

Ubuntuの場合、ModuleNotFoundError: No module named 'serial'のメッセージが出力されたときはビルドができていません。下記のコマンドをターミナルで実行し、ビルドに必要なアプリをインストールしてください。

```
Unset
$ sudo apt-get install python-pip
$ pip install pyserial
$ pip install esptool
```

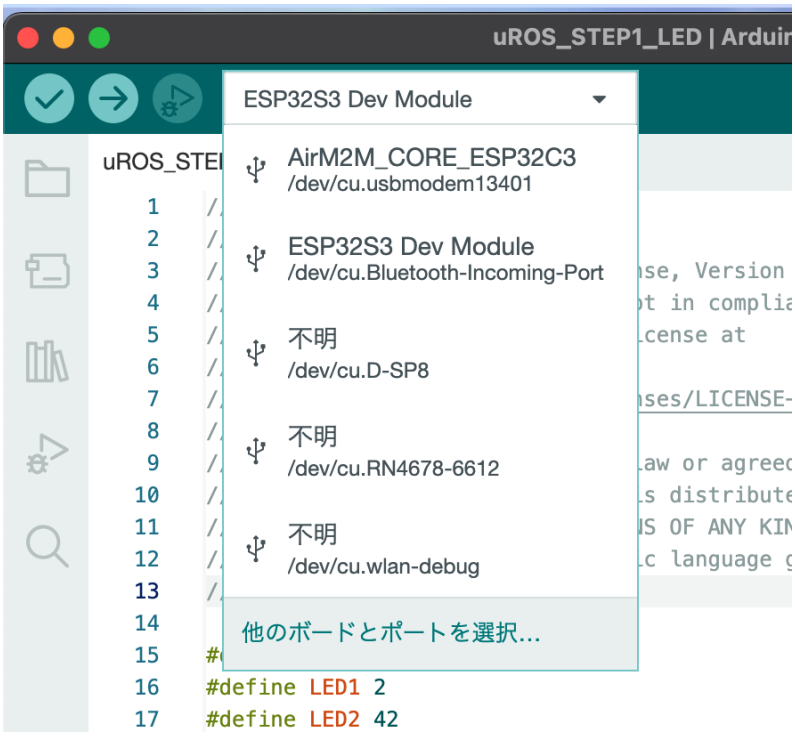
スケッチにエラーが無いことを確認したら、PCと本製品をUSBケーブルで接続し、本製品のモード切替スイッチをRUN1と書れていない無い方向（USBコネクタ側でない方向）にスライドし、Pi:Co Classic3の電源を入れます。Pi:Co Classic3の電源を入れた状態でPCと接続した場合は、本製品のリセットボタンを押します。



モード切替スイッチを操作し、USBケーブルを接続した状態

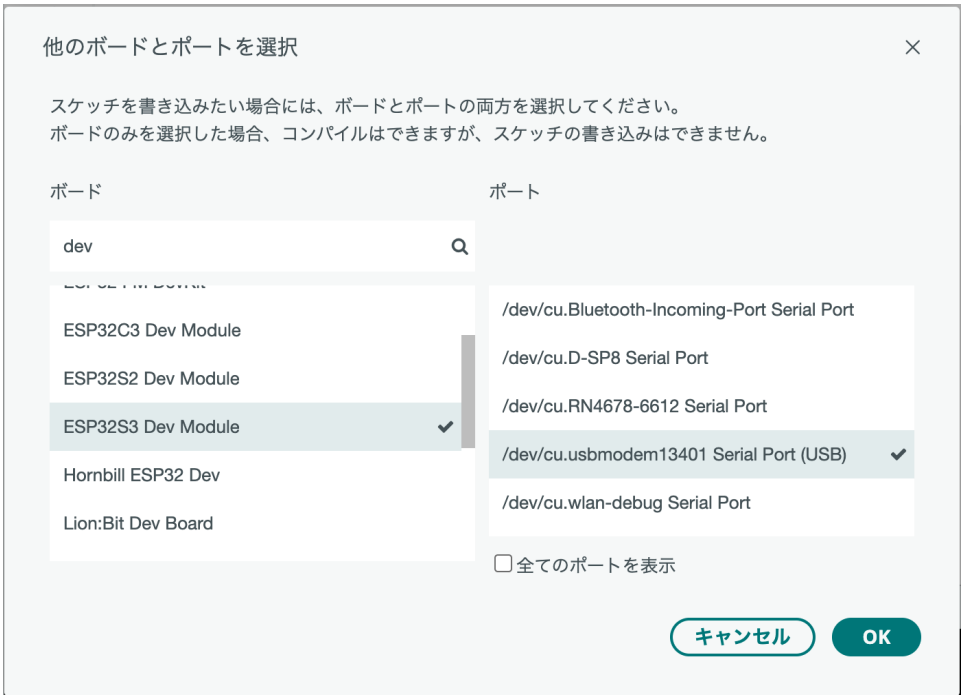
本製品をPCと接続した後、以下の手順でボードとポートを設定します。Pi:Co Classic3の電源をOFFした後や、本製品のリセットボタンを押した後にも再度設定してください。

メニューバーの下に表示されているボード名（ESP32S3 Dev Module）をクリックし、一番下の”他のボードとポートを選択...”を開きます。



ボード名->他のボードとポートを選択...を開く

ボードの検索欄に”dev”を入力し、”ESP32S3 Dev Module”にチェックが入っていることを確認します。チェックが入っていない場合は、ESP32S3 Dev Moduleをクリックします。ポート選択欄では本製品を接続したUSBポートを選択します。Windowsの場合はCOMX、macOSの場合はusbmodemXXXXX Serial Port(USB) 、Ubuntuの場合は/dev/ttyUSBXという名称です。選択したら”OK”をクリックします。

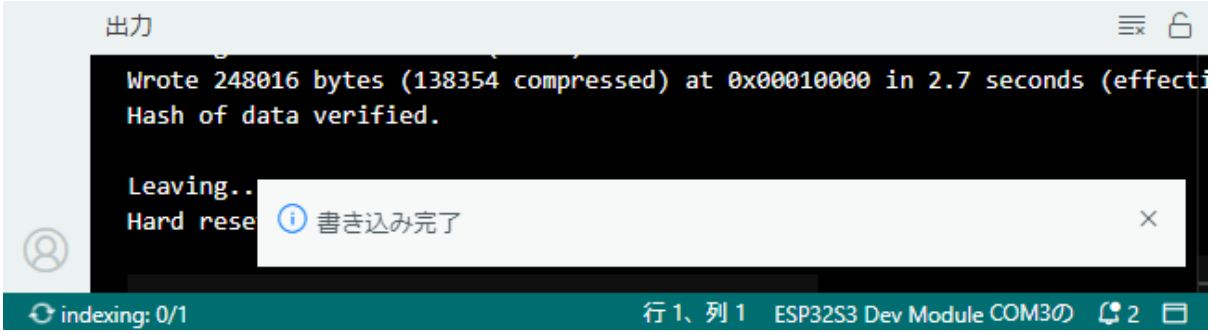


ボードとポートの設定

設定完了後、ボート名の左にUSBのマーク 

が追加されます。スケッチの書き込み以外に、PCとハードウェアシリアル通信をするときも、ポート設定が必要です。

ポート設定後、  をクリックしてスケッチを書き込みます。下記のように表示されたら書き込み完了です。



書き込み完了時の画面表示

Ubuntu PCで書き込みに失敗する時は、書き込み前にターミナルで次のコマンドを実行し、USBポートの使用権限を変更します。

```
Unset
$ sudo chmod 777 /dev/ttyACM0
```

永続的にUSBポートの使用権限を変更する場合は、次のコマンドを実行し、PCを再起動します。

```
Unset
# 下記を実行した後にPCを再起動すること
$ sudo usermod -aG dialout $USER
```

スケッチの書き込み完了後、本製品のモード切替スイッチをRUN1と書かれた方向にスライドしてリセットを押すか、電源をOFF->ONするとPi:Co Classic3前方のLEDが0.5秒点滅、0.5秒消灯を繰り返します。

micro-ROSの環境設定

　ここではmicro-ROSのインストール方法と、Pi:Co Classic3とPC間で通信する方法について説明します。PCのOSはUbuntuを推奨します。また、PCにはあらかじめROS 2 Humbleをインストールしてください。インストール方法は https://docs.ros.org/en/humble/Installation/Ubuntu-Install-Debians.html を参照してください。
　ROS 2 Humbleインストール後に下記のコマンドを実行し、ROS 2パッケージをビルドするために必要なソフトウェアをインストールしてください。

```
Unset
$ sudo apt install python3-colcon-common-extensions
$ sudo apt install python3-rosdep2
$ rosdep update
```

micro-ROS-Agentのインストール

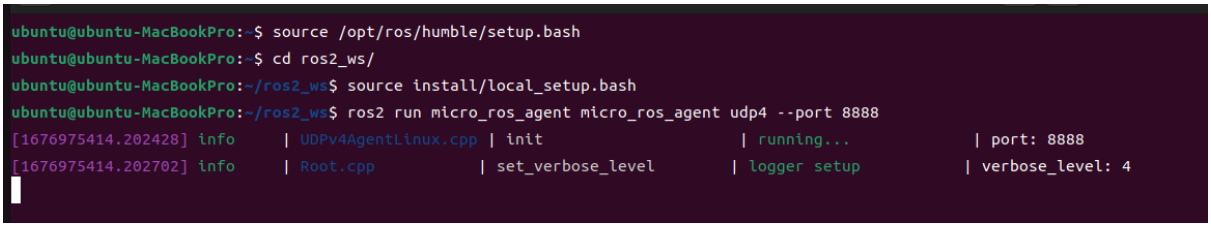
　Ubuntu PCにmicro-ROS-Agentのソースコードをダウンロードしてビルドします。次のコマンドを実行します。ROS 2のワークスペースはros2_wsとしています。

```
Unset
# ワークスペースを作成します
$ source /opt/ros/humble/setup.bash
$ mkdir -p ~/ros2_ws/src
$ cd ~/ros2_ws/src

# micro-ROSセットアップパッケージをダウンロードしてビルドします
$ git clone -b humble
https://github.com/micro-ROS/micro_ros_setup
src/micro-ros-build
$ sudo apt update && rosdep update
$ cd ~/ros2_ws
$ rosdep install --from-paths src --ignore-src -y
$ colcon build
$ source install/local_setup.bash

# micro-ROS-Agentパッケージをダウンロードします
$ ros2 run micro_ros_setup create_agent_ws.sh

# micro-ROS-Agnetパッケージをビルドしてインストールします
$ colcon build
$ source install/local_setup.bash
```

インストールができたら次のコマンドを実行しmicro-ROS-Agentを起動します。

```
Unset
$ ros2 run micro_ros_agent micro_ros_agent udp4 --port 8888
```

micro-ROS-Agentを起動すると次のように表示されます。終了時はCtrl+Cを入力します。



micro-ROS-Agentの実行画面

micro-ROSライブラリにカスタムメッセージをインストールする

Arduino用のmicro-ROSライブラリ（micro_ros_arduino）には、ロボットを操作する時に使用するtfやtwistなどの基本的なROSメッセージが用意されています。本製品のサンプルスケッチでもそれらのメッセージを使用しています。

ROSメッセージはユーザが自由に作成できます（これをカスタムメッセージと呼びます）。サンプルスケッチでは、Pi:Co Classic3のセンサ値を表現するカスタムメッセージを使用しています。micro_ros_arduinoはバイナリファイルで提供されているため、カスタムメッセージを追加するにはmicro_ros_arduinoをビルドします。

ここでは、micro_ros_arduinoのソースファイルをダウンロードし、カスタムメッセージを追加してビルドする方法を説明します。追加するカスタムメッセージは http://github.com/rt-net/pico_msgs に公開されています。

micro_ros_arduinoを作成する際にDockerを使用します。Dockerをインストールしていない場合はhttps://docs.docker.com/engine/install/ubuntu/を参照してインストールしてください。下記に示すコマンドは、先ほどのリンクにあるインストール方法の一つです。

```
Unset
$ sudo apt-get update
$ sudo apt-get install ca-certificates curl gnupg
$ sudo install -m 0755 -d /etc/apt/keyrings
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | \
sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg
$ sudo chmod a+r /etc/apt/keyrings/docker.gpg
$ echo "deb [arch="$(dpkg --print-architecture)"
signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu \
"$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
$ sudo apt-get update
$ sudo apt-get install docker-ce docker-ce-cli containerd.io
docker-buildx-plugin docker-compose-plugin
```

次にmicro_ros_arduinoのソースファイルをダウンロードします。

```
Unset
$ cd ~
$ git clone http://github.com/micro-ROS/micro_ros_arduino
```

ダウンロードしたmicro_ros_arduinoフォルダにあるextras/library_generation/extra_packages/extra_packages.reposにPi:Co Classic3のセンサのパッケージのURLを追加します。

```
Unset
  repositories:
     control_msgs:
       type: git
       url: https://github.com/ros-controls/control_msgs
       version: galactic-devel
     pico_msgs: # 追加
       type: git # 追加
      url: http://github.com/rt-net/pico_msgs # 追加
```

次にビルド用のdockerイメージでmicro_ros_arduinoをビルドします。オプションなしでコマンドを実行すると、micro_ros_arduinoがサポートしているすべてのマイコンに対してライブラリをビルドするため時間がかかります。ここでは、ESP32のみビルドするようにオプション(-p esp32)をつけます。

```
Unset
$ cd ~/micro_ros_arduino
$ docker pull microros/micro_ros_static_library_builder:humble
$ docker run -it --rm -v $(pwd):/project --env \
 MICROROS_LIBRARY_FOLDER=extras \
 microros/micro_ros_static_library_builder:humble -p esp32
```

ビルドが成功するとmicro_ros_arduino/srcにpico_msgsフォルダができます。このpico_msgsフォルダとesp32/libmicroros.aファイルをArduino IDEの環境にコピーします。
pico_msgsフォルダは、Arduinoライブラリがあるlibraries/micro_ros/ros_arduino-2.0.5-humble/srcにコピーし、libmicroros.aはlibraries/micro_ros/ros_arduino-2.0.5-humble/src/esp32s3にコピーします。

```
Unset
# ~/Arduino/librariesにライブラリがある場合
$ cd ~/micro_ros_arduino/src
$ cp -r pico_msgs
~/Arduino/libraries/micro_ros_arduino-2.0.5-humble/src
$ cp esp32/libmicroros.a
~/Arduino/libraries/micro_ros_arduino-2.0.5-humble/src/esp32s3
```

以上でカスタムメッセージのインストール完了です。

PCのROS 2環境にカスタムメッセージをインストールする

Pi:Co Classic3が送信するカスタムメッセージを受け取るために、PCのROS 2環境へカスタムメッセージをインストールします。次のコマンドを実行してください。

```
Unset
$ cd ~/ros2_ws/src
$ git clone https://github.com/rt-net/pico_msgs.git
$ cd ~/ros2_ws
$ rosdep install --from-paths src --ignore-src -y
$ colcon build
$ source install/local_setup.bash
```

以上でインストール完了です。

PCとPi:Co Classic3間の通信確認

ここでは、micro_ros_arduinoに追加したカスタムメッセージを使用し、Pi:Co Classic3からPCへメッセージを送信できることを確認します。

Pi:Co Classic3側

カスタムメッセージを使ったサンプルスケッチ（uROS_STEP11_SensorMsg）を使用します。PCとPi:Co Classic3との間で通信を行うためWi-Fiの設定が必要です。まず、PCをWi-Fiに接続してPCのIPアドレスを取得します（例 : 192.168.1.89）。その後、スケッチの下記の行を使用するWi-FiのAP（アクセスポイント）名、パスワード、取得したPCのIPアドレス、ポート番号（デフォルトは8888）に書き換えてください。また、注意事項として、PCとPi:Co Classic3は、同じWi-FiのAPに接続してください。同じWi-FiのAPに接続しないと通信できません。

```C/C++
set_microros_wifi_transports("使用するWiFiのAP名", "Wi-Fiのパスワード", "PCのIPアドレス", 8888);
```

変更したらスケッチをビルドして書き込みます。

PC側

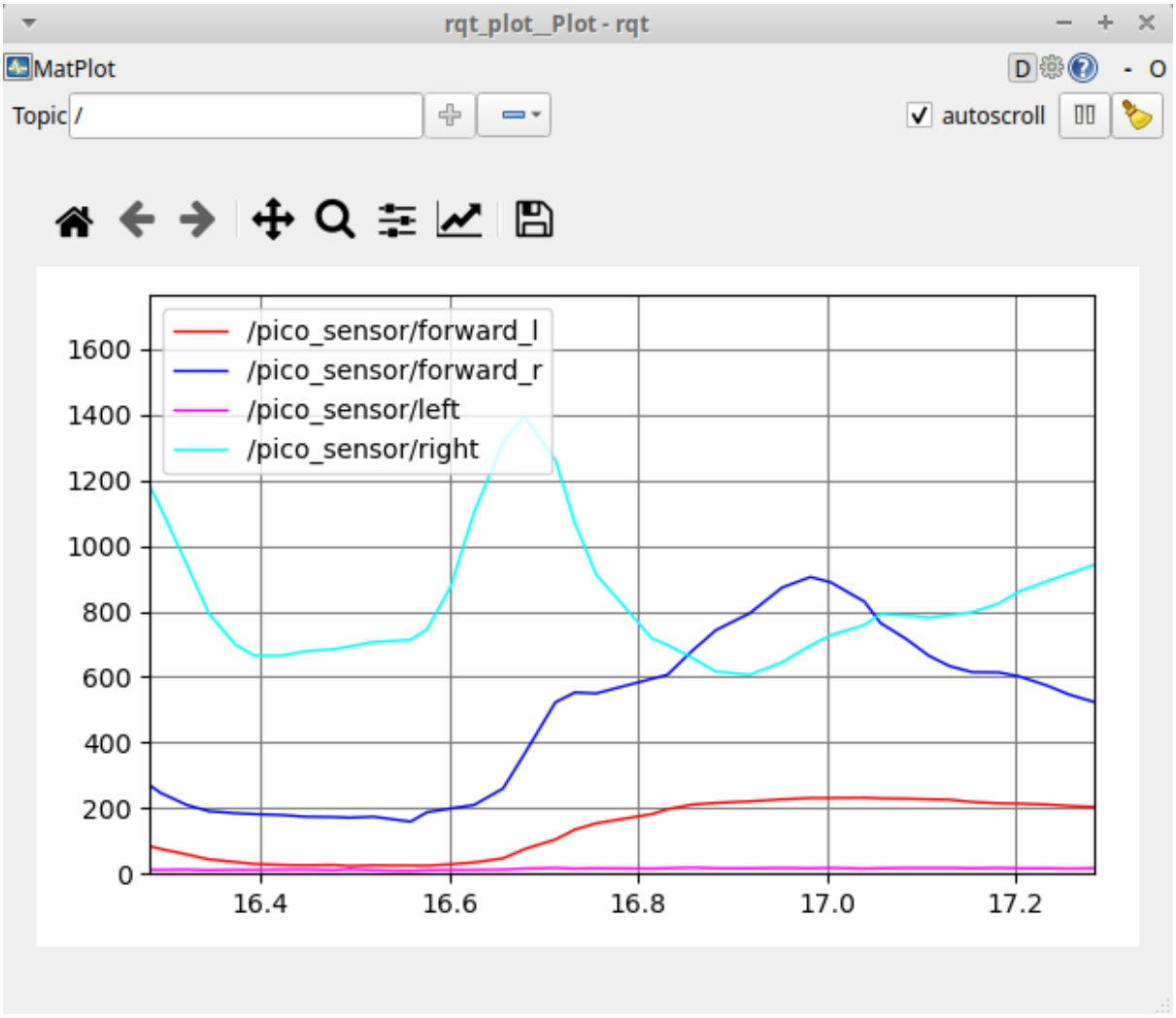
ターミナルを2つ起動し、以下のコマンドを実行します。
1つ目のターミナルではmicro-ROS-Agentを起動します。

```Unset
$ source ~/ros2_ws/install/setup.bash
$ ros2 run micro_ros_agent micro_ros_agent udp4 --port 8888
```

2つ目のターミナルではrqt_plotを起動します。

```Unset
$ source ~/ros2_ws/install/setup.bash
$ ros2 run rqt_plot rqt_plot topics /pico_sensor/forward_r /pico_sensor/forward_l /pico_sensor/right /pico_sensor/left
```

通信に成功すると、rqt_plotにPi:Co Classic3の壁センサ値が描画されます。壁センサに手をかざすと値が変化します。



rqt_plotでPi:Co Classic3の壁センサ値を表示

使用しているツール、OSSのバージョン

本書で使用するツールやOSSのバージョンは以下のとおりです。

本書で使用するツールやOSSのバージョン情報

ツールやOSS	バージョン	URL
Arduino IDE	2.1.0	https://www.arduino.cc/en/software
Arduino core for the ESP32	2.0.8	https://github.com/espressif/arduino-esp32/releases
micro_ros_arduino	v2.0.5_humble	https://github.com/micro-ROS/micro_ros_arduino/releases

製品保証

保証の内容

お客様にお買い上げいただきました本製品につき、株式会社アールティ（以下「弊社」といいます）の設計あるいは製作上の責任にて故障や不具合が生じた場合、下記に示す保証期間と条件により、無償で修理等します。

保証者の名称、所在地および電話番号

保証者は、本書の「お問い合わせ」に記載のとおりとします。なお、修理等の受付時間は、「お問い合わせ」に記載の受付時間とします。

保証期間

保証期間は、本製品を納入した日から起算して12ヶ月間とし、この期間を経過した場合、保証は終了とします。

保証の適用

1. この保証は、日本国内で販売し、使用される本製品に適用されます。海外に設置や移動した本製品は、この保証の対象となりません。

2. この保証は、本製品の本体についてのみ適用します。本製品のセットを構成する付属品については、この保証の対象外とします。

3. 本体の故障または不具合により生じた本製品の本体以外の故障、不具合、破損、滅失、損害（人的・物的損害、間接損害、特別損害、逸失利益等）については、本保証の対象外とします。

4. この保証は、標準仕様の製品に適用されます。特殊仕様および特記事項を含む特注仕様の内容は保証範囲外とします。

保証の除外事項

次の何れかの事項に該当する場合、保証は適用されません。

1. 本製品のマニュアル、装置添付ラベル、取扱説明書群（以下「マニュアル等」といいます）が定める手順、注意事項、安全事項、確認事項、動作方法等を順守しなかったことによる故障または不具合

2. マニュアル等に記載された稼働環境条件以外の条件のもとで稼働させたことに起因する故障または不具合

3. マニュアル等に記載された仕様（可搬重量、動作速度等）の限度・範囲を超える使用（お客様によるプログラムの改変、本体の改造等を含みます）に起因する故障または不具合

4. 経時変化による劣化・故障・不具合

5. 天災地変による故障または不具合

6. 結露、異常電圧、衝突、転倒、落下、公害等の事故による故障または不具合

7. 弊社または弊社が指定する業者以外による修理・整備に起因する故障または不具合

8. 前各号のほか、弊社の責に帰すことのできない事由により生じた故障または不具合

保証の態様

保証期間中に、マニュアル等に従い正常な使用状態で本製品が故障し、または不具合を起こし、お客様より弊社にその旨ご連絡をいただいた場合、弊社は、弊社の判断により、以下の何れかの措置を講じます。

(ア) 無償修理
お客様よりお送りいただいた現品につき修理、部品の交換等を行い、正常な状態に回復します。修理は、弊社または弊社が委託した業者が行うものとします。部品等にかかる費用は、弊社の負担とします。なお、修理のために交換された部品または本体の一部については、お返しできない場合がありますのでご了承ください。

(イ) 本製品の無償交換
お客様よりお送りいただいた現品につき修理不能と弊社が判断した場合、同等の製品と交換します。なお、この場合、現品の全部または一部をお返しできない場合がありますのでご了承ください。同等の製品をお客様へお送りする際の送料は、弊社の負担とします。

(ウ) 返金
弊社は、（ア）項に定める無償修理および（イ）項に定める無償交換に応じることができないと判断したとき、本製品の購入価格を上限として返金します。なお、返金の際の銀行振込等の手数料は、弊社の負担とします。

お客様の費用負担

次に掲げる費用は、お客様の負担とします。

1. 前条（ア）項に定める無償修理について、修理前の現品をお客様から弊社へお送りいただく際の梱包費用および送料。なお、お客様は、輸送に耐え得る梱包方法にて梱包するものとします。

2. 前条（イ）項に定める無償交換について、交換前の現品をお客様が弊社へお送りいただく際の梱包費用および送料。なお、お客様は、輸送に耐え得る梱包方法にて梱包するものとします。

3. 保証の除外事項に該当することが判明した場合または故障・不具合でないことが判明した場合の修理・交換サービス料金および返送の際の送料。弊社の点検・調査により、保証の除外事項に該当することが判明した場合、弊社は、お客様にその旨お伝えし、修理等の要否について確認します。要修理等とのご回答をお客様から得た場合、弊社は、別途お客様と合意した修理・交換サービス料金にて修理等を行います。 なお、本項に定める場合の現品の返送にかかる送料は、お客様負担とします。

保証を受けるための手続き

弊社は、故障・不具合の原因の究明、修理等の解析を迅速に行うため、お客様に下記のお手続きをお願いします。なお、修理期間は、現品到着日より約2週間とさせていただきますが、故障状況によってはさらに時間を要する場合がありますのでご了承ください。

1. 使用条件をできる限り詳細に明記した書面の提示
2. 故障状況をできる限り詳細に明記した書面の提示

お問い合わせ

本製品に関するお問い合せは、下記窓口までお申し付けください。最新の製品情報、会社情報等については、弊社ホームページをご覧ください。

〒101-0021
東京都千代田区外神田3-9-2 末広ビル3F
株式会社アールティ

URL　https://rt-net.jp/

TEL　03-6666-2566

E-mail　support@rt-net.jp （技術サポート）

sales@rt-net.jp （営業サポート）

受付時間　平日11:00-18:00（土日祝、夏季、年末年始はお休みです）

改訂履歴

発行日 (YY/MM/DD)	版数	改訂内容	編集者
23/6/26	1.0	初版	Masatake Aoki

Copyright・知的財産権について