



BT-AdpKit 説明書

Bluetooth - RS232C(CAT/CI-V)
技術適合対応 ESP32-DeviceKit 付属
BT - A d p (K i t) 説明書
Ver 2.00d



目次

BT-AdpKit 説明書	1
目次	2
はじめに	3
本体付属品	3
BT-AdpKit に関して.....	3
機器の接続・設定	4
外部接続・内部基板の説明	4
動作・ジャンパー設定の説明.....	6
A r d u i n o環境	7
E S P - I D F環境	16
Arduino・ESP-IDF での書き込み失敗の場合	22
ESP32-DeviceKit へのプログラム書込接続イメージ	25

はじめに

この度は、萬拵屋商品をお買い上げいただきましてありがとうございます。
ご使用前やご利用中にこの説明書をお読みいただき正しくご使用いただけますようお願いいたします。

本体付属品

* BT - A d p 本体	1 個
* ESP32-DeviceKit (秋月電子通商さん販売ボード)	1 個
* ジャンパーソケット	9 個
* ケースゴム足 (貼り付け)	4 個
* 説明書	1 冊
* サンプル等収録DVD	1 枚

BT-AdpKit に関して

ESP32-DeviceKit (技適対応) を利用した Bluetooth-RS232C 通信アダプターのキットになります。ご購入後ご自身で ESP32 チップに作成したプログラムを書き込んでいただく必要があります。ESP32-DeviceKit を取り付ける以外の基板上必要な部品、外部配線はすべて完成済みです。

BT - A d p (K i t) は基本的に SD - C N T ・ T C - A H 4 ・ T C - F C 4 等のコントローラ無線機接続 (R S 2 3 2 C ・ C I - V ・ C A T) コネクタを I C - 7 0 5 (アイコム社製無線機) に接続するためのアダプターとして開発しました。使い方はサンプルプロジェクト (A r d u i n o ・ E S P - I D F) を書き込んで頂いても使用可能ですし、自由に E S P 3 2 チップのプログラミングをして頂いて何かアイデアのあるユニットを作成して頂いても、もしくはサンプルプロジェクトを改造してもっと高機能なアダプターとして作成して頂いても使い方は自由です。回路図とサンプルプログラム等々から何かを作って頂ければ幸いです。また回路には I 2 C インターフェースのコネクタ、 I 2 C 仕様のシリアルROMのパターンも準備しています。またプログラムを作成していただく事で無線 Lan に対応した RS232C アダプターとしても利用可能と思われます。ジャンパーピン部分もプログラム作成によっては T T L レベルの I O ポートとして利用可能になります。
(J P 1 は基板内よりのピンが G N D です)

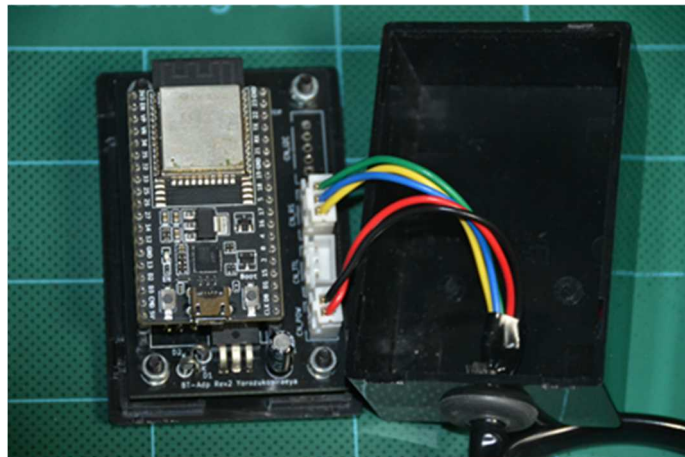
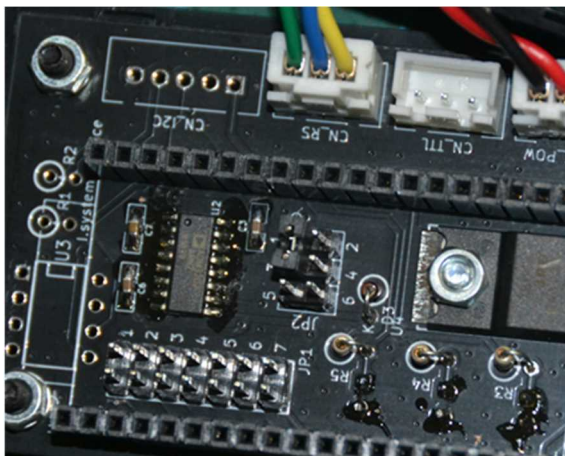
ご注意

ご使用におきましては電源 13.5V から 5V 生成のリニアレギュレーターが多少熱くなりますが、使用には問題ありません。ノイズ発生的にはスイッチング式より遙かに少ないと思われます。プログラム作成で、ジャンパーピン等利用して他のハード機能を追加される場合はレギュレータ熱損失にご注意をお願いします。

機器の接続・設定

外部接続・内部基板の説明

内部基板

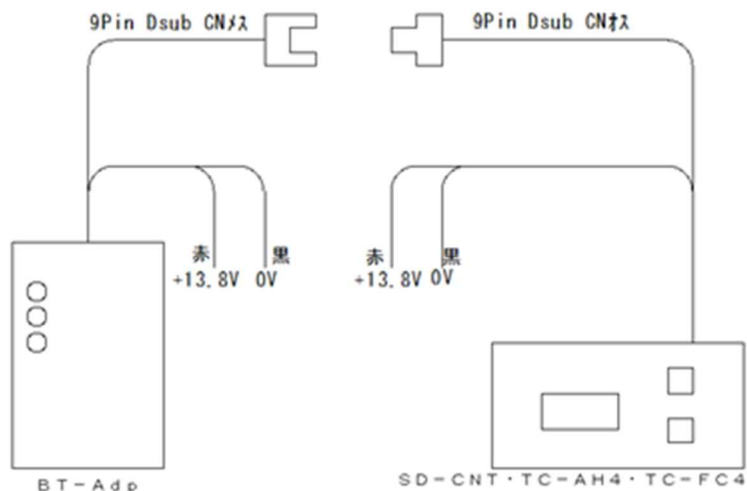


参考基板設定 (サンプルプロジェクト: ESP-IDF) 基板への ESP32-DeviceKit 取り付け状態
 アクセプタ動作、デバイス名(BT-ADP_A0)
 有線通信: 4800bps、8bit、stop1、None

上記の設定でサンプルプロジェクト (ESP-IDF) 書込でSD-CNT・TC-AH4・TC-FC4
 とIC-705が接続できます。

I2Cのコネクタ (CN-12C)、シリアルROM (U3)、I2Cプルアップ抵抗 (R1、R2) は実装されていません。

SD-CNT・TC-AH4・TC-FC4との接続



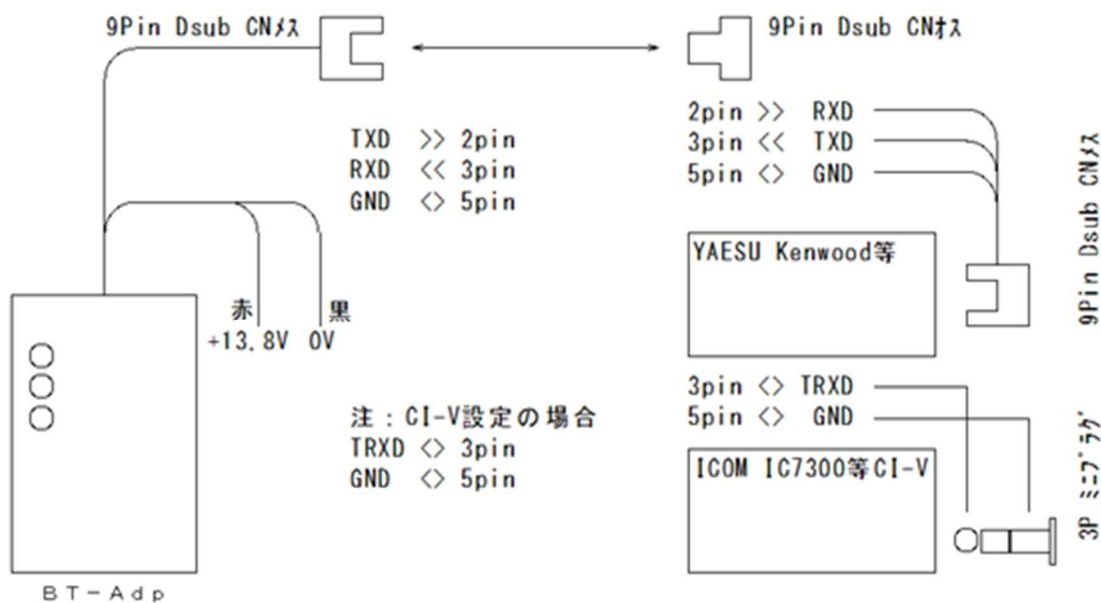
注: BT-Adp と SD-CNT 等との電源接続は赤 = +13.5V、黒 = 0V です。
 個々の機器はダイオードで保護していますが、どちらか逆の場合はコネク
 ターのGNDを経由して電流が流れ、どちらかの基板が損傷します。

BT-Adp が ESP-IDF でのサンプルプロジェクトの場合、BT-Adp 設定は JP1-1~4 はすべて OPEN で利用してくださ
 い。JP1-5 はいずれかでお使いください。Arduino サンプルの場合はジャンパー (JP1) は未サポートです。

BT-Adp 側の通信回路設定は JP2-3, 5 短絡です。(RS232C 仕様)

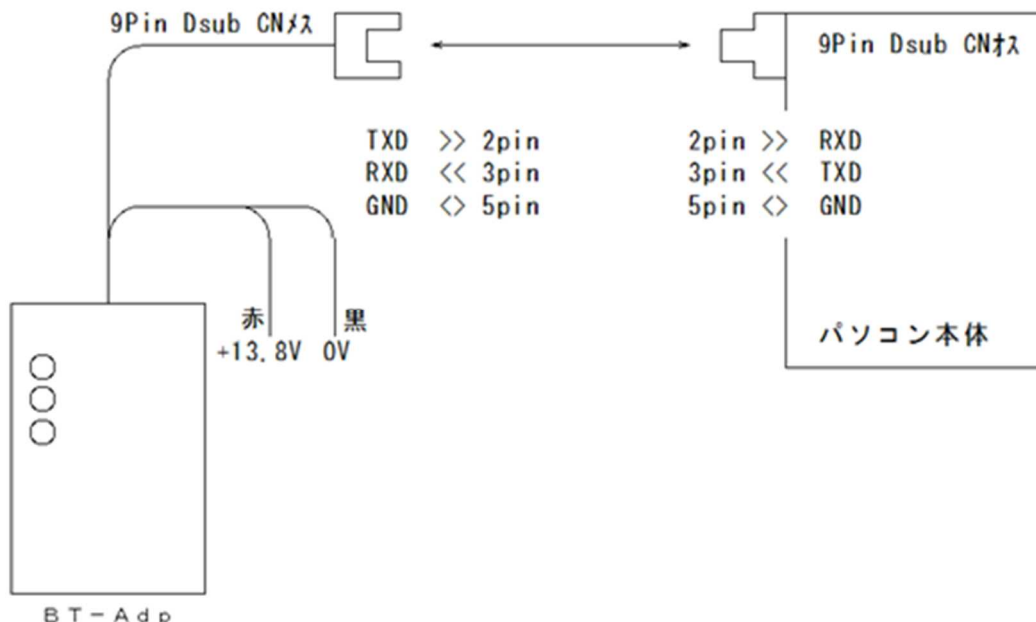
SD-CNT, TC-AH4, TC-FC4 (V3) 側の通信回路設定は JP2-3, 5 短絡です。(RS232C 仕様)

パソコン (HamRadioDelux 等) と無線機を接続する場合・・・ESP-IDFサンプル使用



注：無線機側のピン配置は無線機の説明書等を参照してください。
 パソコン側の Bluetooth は市販の物をご利用ください。
 BT-Adp のリグに対する通信仕様は Arduino サンプルでは未サポートです。
 (プログラム変更が必要です)

パソコン等のRS232CにBT-Adpを接続の場合 (イニシエータで使用)・・・ESP-IDFサンプル使用



注：Arduino サンプルではイニシエータ動作は未サポートです。

動作・ジャンパー設定の説明

動作

- 1 : 電源が投入で Connect (赤LED) が点滅します。
- 2 : IC-705 から接続されると Connect (赤LED) は点灯になります。
またパソコン等からの接続ではサービスでのCOMポートがOPENされると点灯に切り替わります。
- 3 : 通信が発生したとき「BT Recv」はBluetoothでの受信があったとき、
「RS Recv」はD-sub9ピンコネクタから受信があったときに点灯します。

通信インターフェースの設定

RS232C	: JP2 1-3 (CLOSE)
八重洲旧 CAT	: JP2 3-5 (CLOSE)
ICOM CI-V	: JP2 3-5(CLOSE), 4-6(CLOSE)

プログラムでの設定 (JP1はプログラムにより内容が変わります)

<Arduinoサンプルの場合>・・・JP1は未使用です。

- * RS232C 通信仕様・・・4800bps、8bit、stop1、None になっています。
- * 動作モード：アクセプタ (接続される側)
- * デバイス名：BT-ADP

<ESP-IDFサンプルの場合>

* 通信速度

JP1-1(OPEN)、JP1-2(OPEN)	> 4800bps
JP1-1(CLOSE)、JP1-2(OPEN)	> 9600bps
JP1-1(OPEN)、JP1-2(CLOSE)	> 38400bps
JP1-1(CLOSE)、JP1-2(CLOSE)	> 115200bps

* データビット

8bit 固定

* ストップビット

JP1-3(OPEN) > 1bit / JP1-3(CLOSE) > 2bit

* 動作モード：JP1-4(OPEN) > アクセプタ (接続される側)

JP1-4(CLOSE) > イニシエータ (接続する側)

* デバイス名 (アクセプタ時)

JP1-5(OPEN) > BT-ADP_A0

JP1-5(CLOSE) > BT-ADP_A1

* デバイス名 (イニシエータ時)

JP1-5(OPEN) > BT-ADP_I0

JP1-5(CLOSE) > BT-ADP_I1

「BT-ADP_I0」は電源 ON で「BT-ADP_A0」を検索して接続します。

「BT-ADP_I1」は電源 ON で「BT-ADP_A1」を検索して接続します。

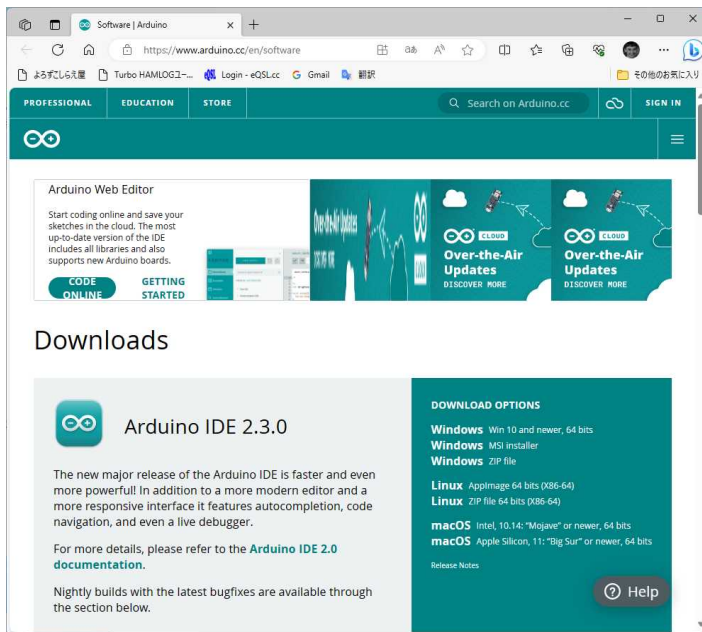
注：SD-CNT・TC-AH4・TC-FC4 以外でのご使用 (HamRadioDelux 等) で ICOM リグを接続される場合は CI-V でのエコーバック有効・無効の設定が必要な場合があります。

パソコン (HamRadioDelux) と IC-705 接続の場合はエコーバック有効でないで HamRadioDelux は接続 NG になります。

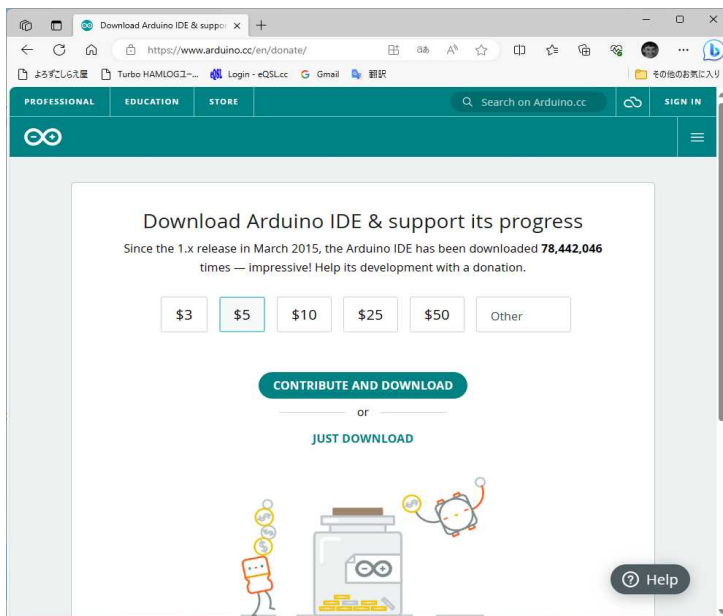
Arduino環境

環境のダウンロード

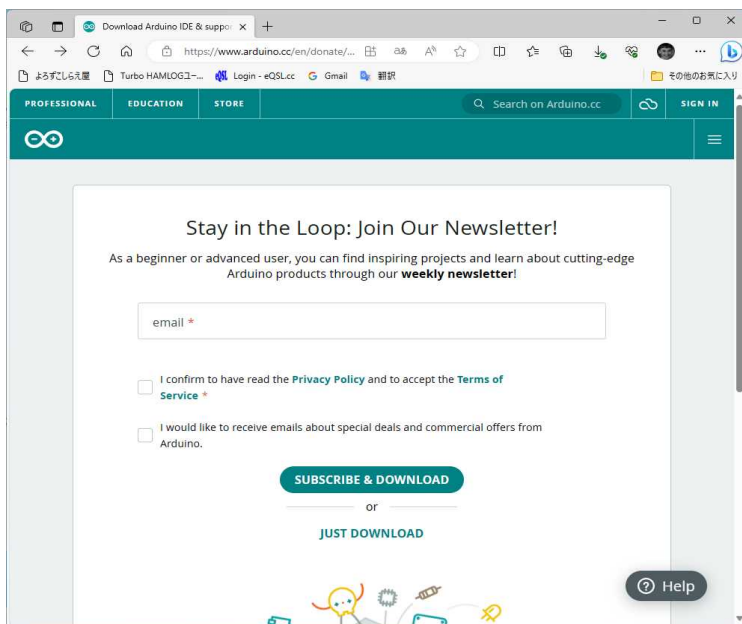
Arduinoのスケッチ環境はブラウザで「<https://www.arduino.cc/en/software>」にアクセスすると下記のページが表示されます。



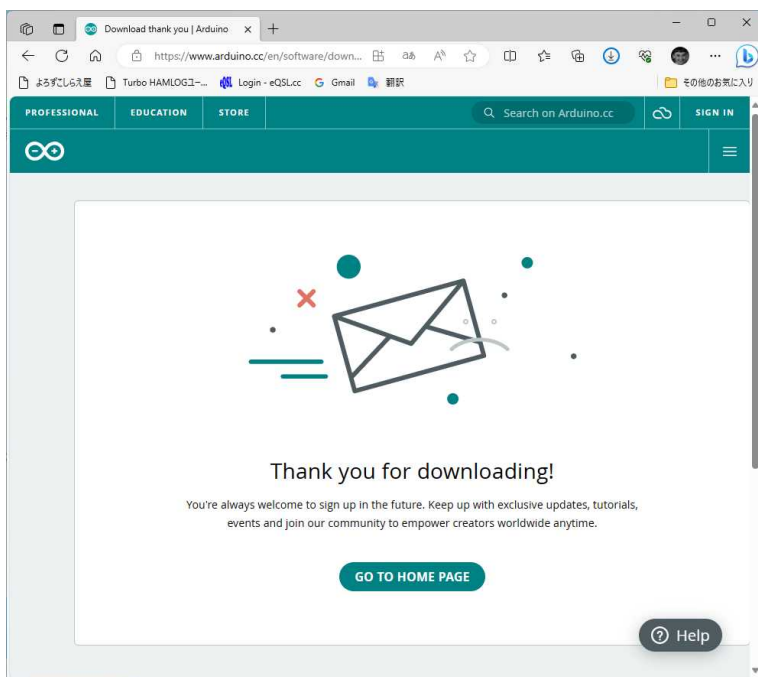
「Arduino IDE 2.3.0」をダウンロードします。
ダウンロードは「Windows MSI installer」がいいと思います。



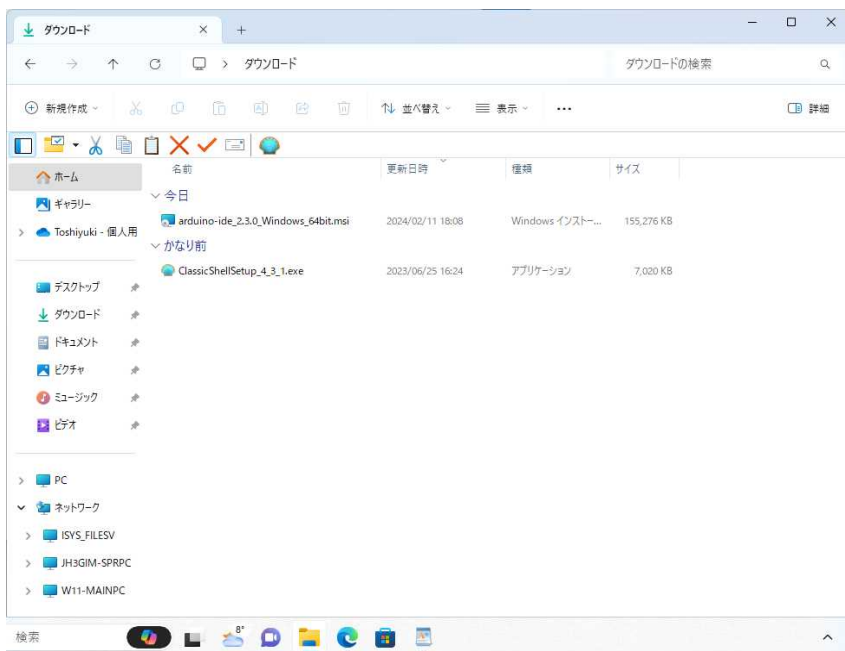
この画面で寄付をされる方は上の「CONTRIBUTE & DOWNLOAD」を、無料で使われる方は「JUST DOWNLOAD」をクリックしてください。



メールアドレスが求められますが下の「JUST DOWNLOAD」でOKです。



そうするとブラウザで「arduino-ide_2.3.0_Windows_64bit.msi」がダウンロードされま



パソコンのダウンロードフォルダーのネットからコピーされた「arduino-ide_2.3.0_Windows_64bit.msi」をクリックすることでArduinoのインストールが開始されます。インストールは過去のVerの様に同意とかの画面はなく勝手にインストールされ、デスクトップにアイコンが生成されます。

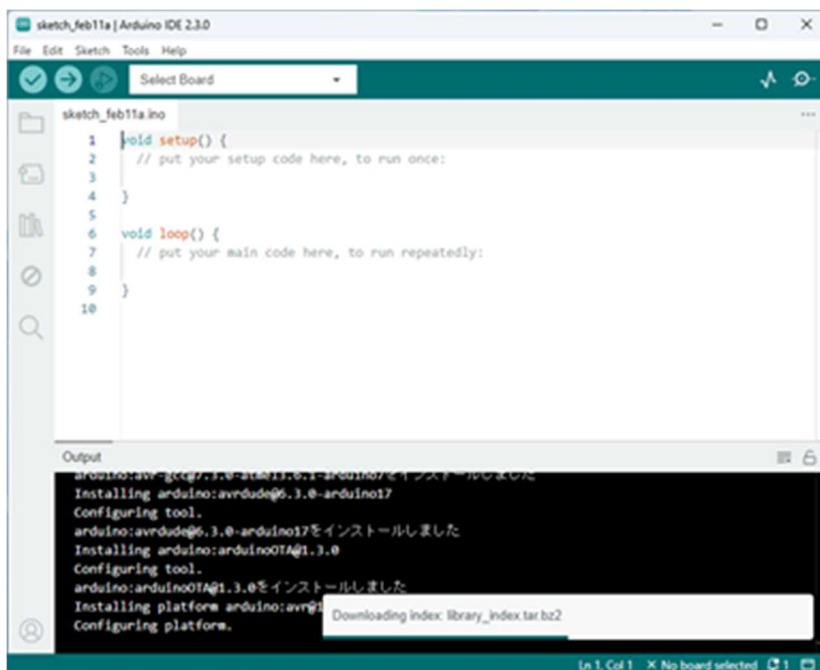


Arduino の起動

ウィンドウズ画面の「Arduino IDE」のアイコンが作成されますので、ダブルクリックで実行します。



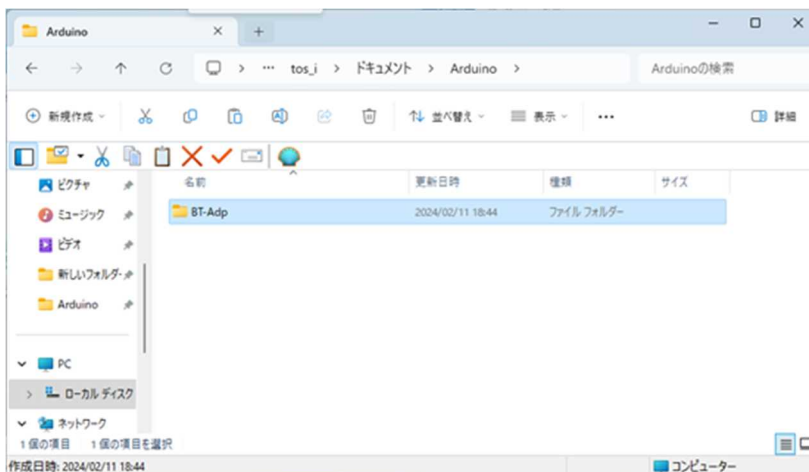
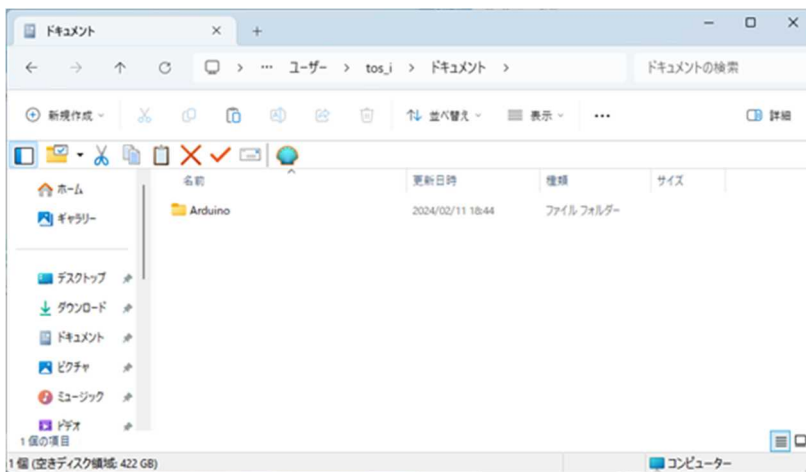
もしかするとこんな感じの画面がでますので、「許可」をクリックしてください。
・・・「ndsn-・・・」と「ArduinoIDE」の2つができました。
他にも USB 関連とインストールの画面があればインストールしてください。
その後下記の画面でスケッチの起動が完了です。



日本語には「File」>「Preferences」で次の画面での「Settings」タブにある「Language」で日本語を選択します。OKをクリックするとメニューは日本語に代わります。

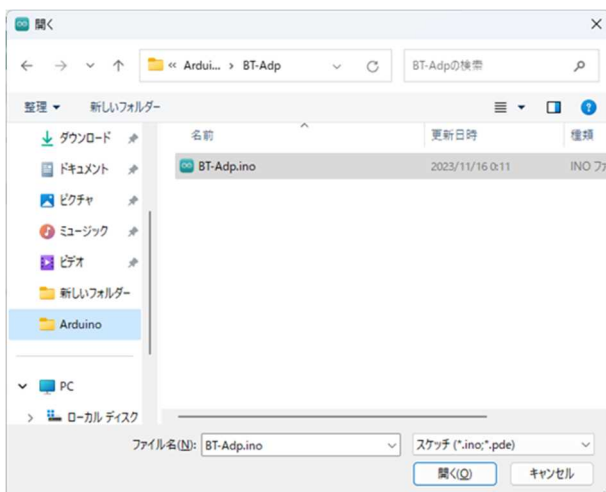
BT-Adp プロジェクト (スケッチ) のコピー

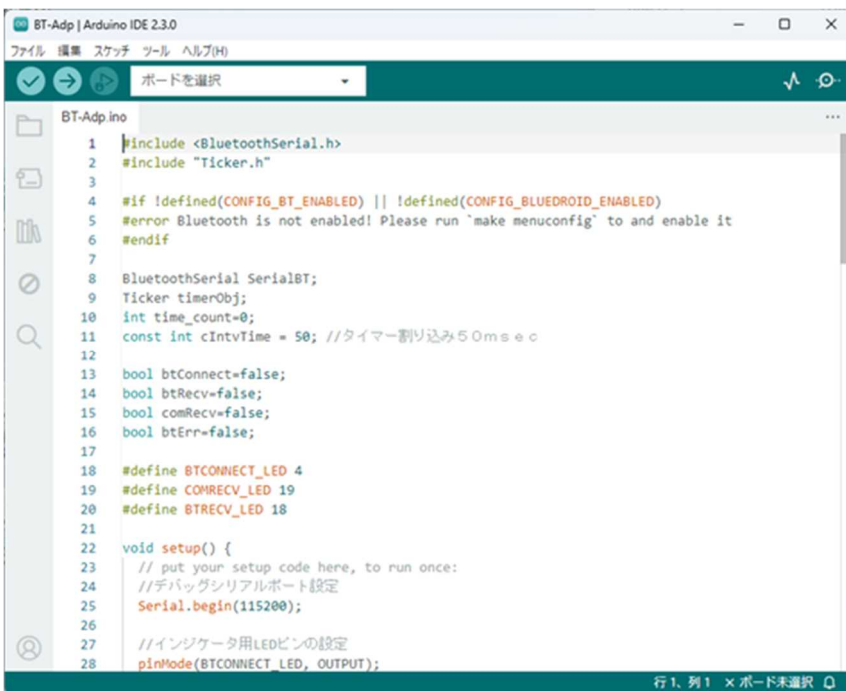
パソコンのドキュメントに「Arduino」フォルダーが作成されていますので、ここに付属 CD の「Arduino」フォルダーにある「BT-Adp」フォルダーをコピーします。



コンパイル・書き込み

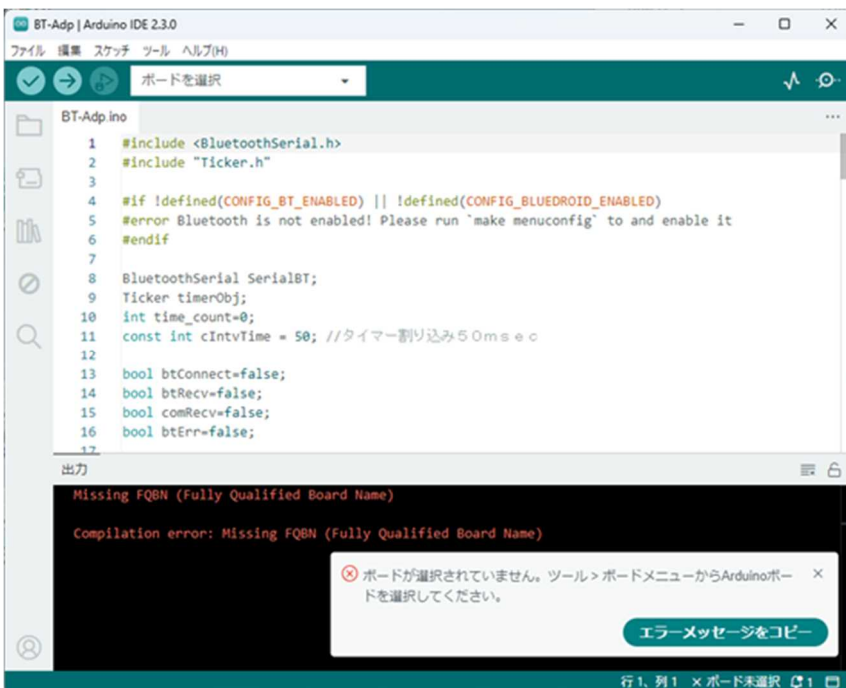
Arduino IDE の「ファイル」>「開く」で先にコピーした「BT-Adp」フォルダーの中の「BT-Adp.ino」を選択して開きます。





```
BT-Adp.ino
1  #include <BluetoothSerial.h>
2  #include "Ticker.h"
3
4  #if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
5  #error Bluetooth is not enabled! Please run `make menuconfig` to and enable it
6  #endif
7
8  BluetoothSerial SerialBT;
9  Ticker timerObj;
10 int time_count=0;
11 const int cIntvTime = 50; //タイマー割り込み50msec
12
13 bool btConnect=false;
14 bool btRecv=false;
15 bool comRecv=false;
16 bool btErr=false;
17
18 #define BTCONNECT_LED 4
19 #define COMRECV_LED 19
20 #define BTRECV_LED 18
21
22 void setup() {
23   // put your setup code here, to run once:
24   //デバッグシリアルポート設定
25   Serial.begin(115200);
26
27   //インジケータ用LEDピンの設定
28   pinMode(BTCONNECT_LED, OUTPUT);
```

この画面でスケッチの読み込みが完了です。ここでコンパイル実行するとまだ ESP32 の環境ができていないのでエラーが発生します。



```
BT-Adp.ino
1  #include <BluetoothSerial.h>
2  #include "Ticker.h"
3
4  #if !defined(CONFIG_BT_ENABLED) || !defined(CONFIG_BLUEDROID_ENABLED)
5  #error Bluetooth is not enabled! Please run `make menuconfig` to and enable it
6  #endif
7
8  BluetoothSerial SerialBT;
9  Ticker timerObj;
10 int time_count=0;
11 const int cIntvTime = 50; //タイマー割り込み50msec
12
13 bool btConnect=false;
14 bool btRecv=false;
15 bool comRecv=false;
16 bool btErr=false;
17
```

出力

Missing FQBN (Fully Qualified Board Name)

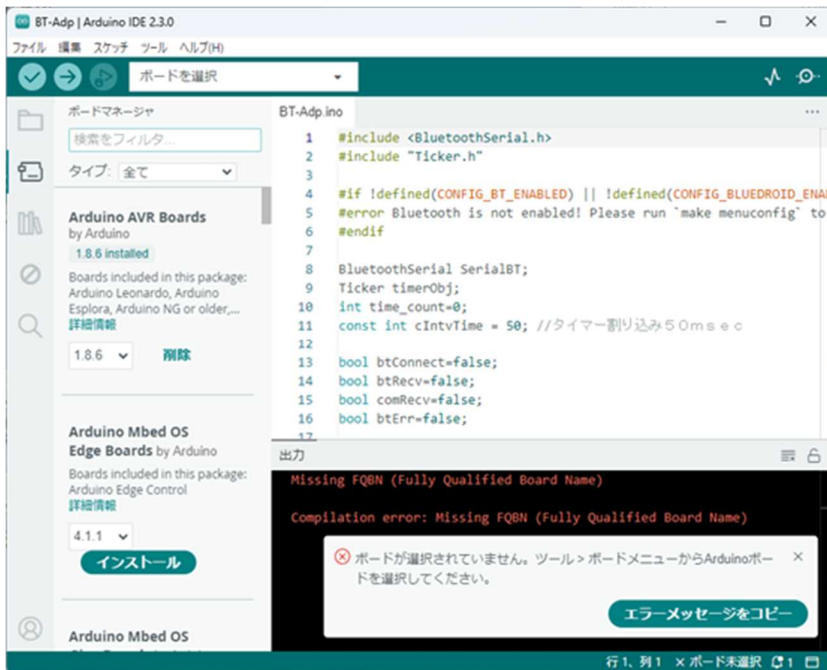
Compilation error: Missing FQBN (Fully Qualified Board Name)

✖ ボードが選択されていません。ツール>ボードメニューからArduinoボードを選択してください。

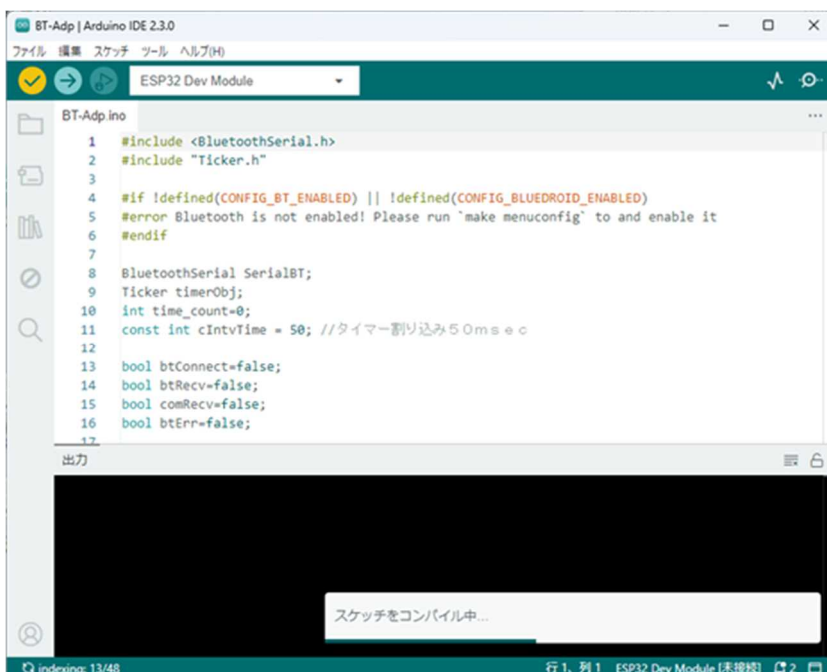
エラーメッセージをコピー

行 1, 列 1 × ボード未選択

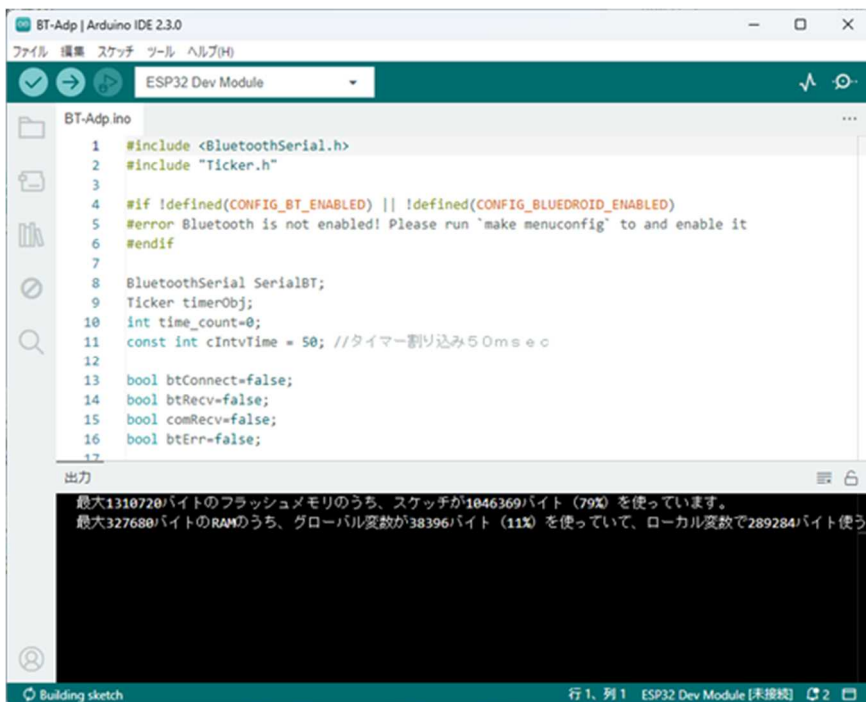
ESP32 環境を設定するには、「ツール」>「ボード」>「ボードマネージャ」を選択すると左側にボードマネージャが表示されます。



検索で「ESP」と入力しますと「esp32 by Espressif」が見つかりますので、これをインストールします。インストールが完了したら「ツール」>「ボード」>「esp32」から「ESP32 Device Module」を選びます。



これでコンパイルすると



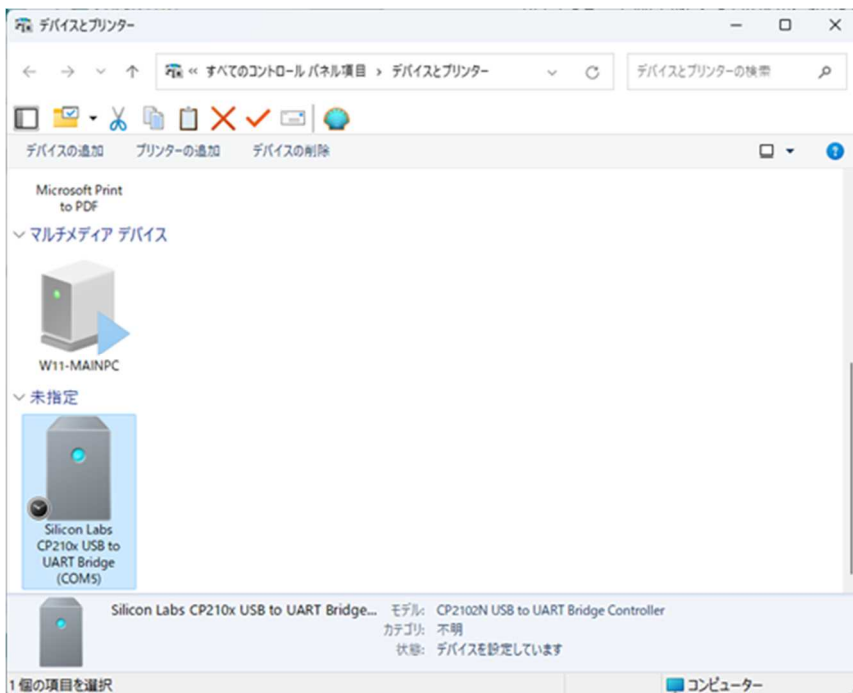
コンパイル完了します。

完了すると・・・画面下部にメモリ情報等が表示され、オブジェクトが作成されます。

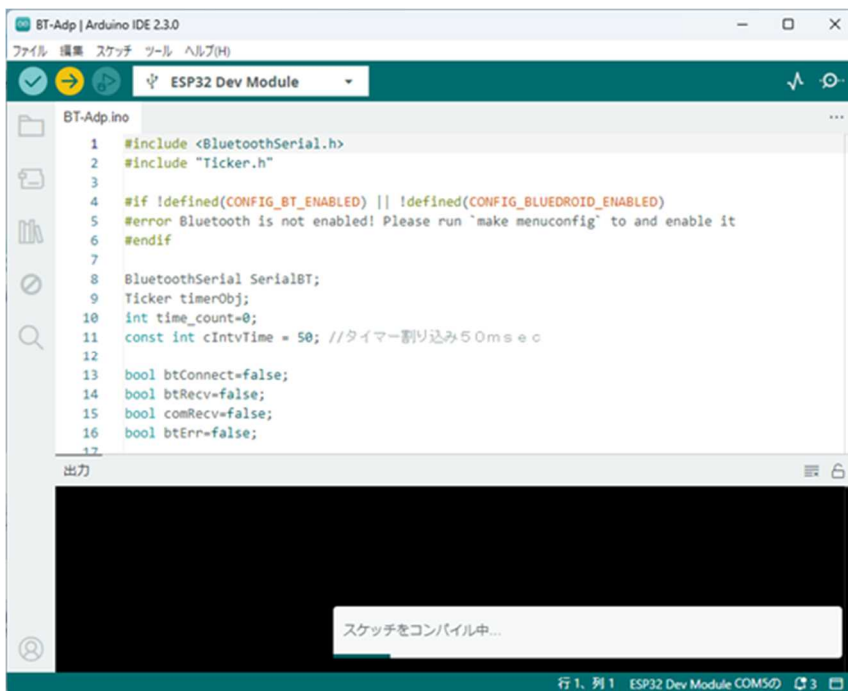
このあと、「ESP32-DeviceKit」にUSBケーブルを接続してパソコン接続すれば、下記の様にCOMポートとして認識されます。（「デバイスとプリンター」で確認します）

認識されない場合はデバイスドライバがインストールされていないのかもしれませんが。

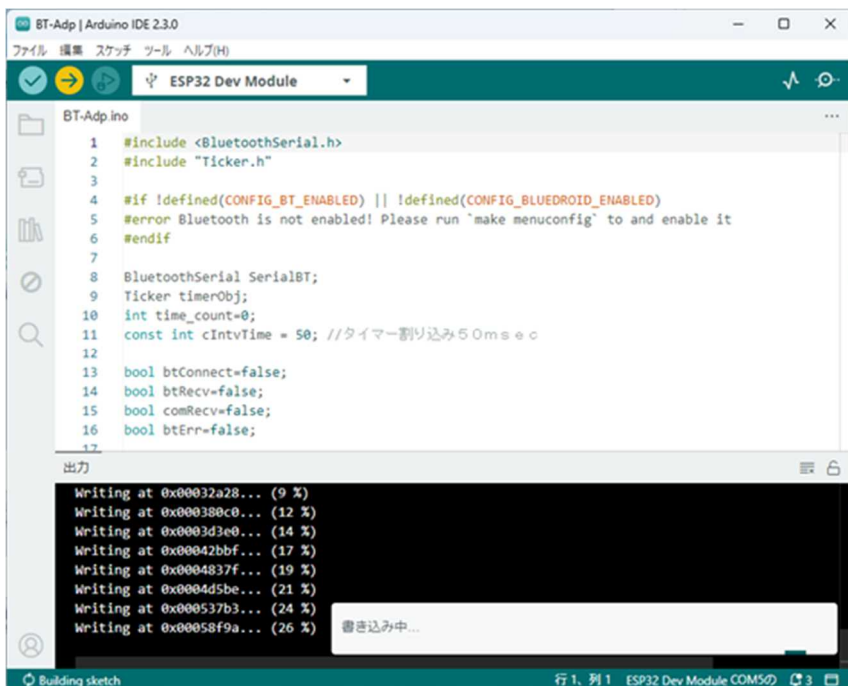
作成したWindows11の環境では「Silicon labs CP210x USB to UART Bridge(COM5)」と認識されました。



上記の確認では「COM5」になっていますので、スケッチの「ツール」>「シリアルポート」で「COM5」を選択しておいて、スケッチの2つ目のボタン「→」で書き込みを実行します。



再度コンパイルが実行され・・・その後書き込みの状態が表示されます。



書き込みが完了しましたら基板に戻して電源を投入すると赤いLEDが点滅し、IC-705で検索すると「BT-ADP」が見つかります。

「接続しますか」・・・で接続するとBT-Adpの赤LEDが点灯になり接続されます。

ESP-IDF環境

環境のダウンロード

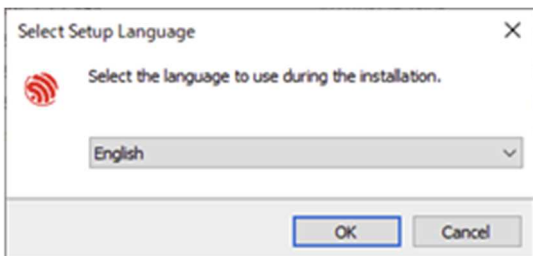
ESP-IDF環境はブラウザで「<https://dl.espressif.com/dl/esp-idf/>」にアクセスしますと、下記の画面が表示されます。



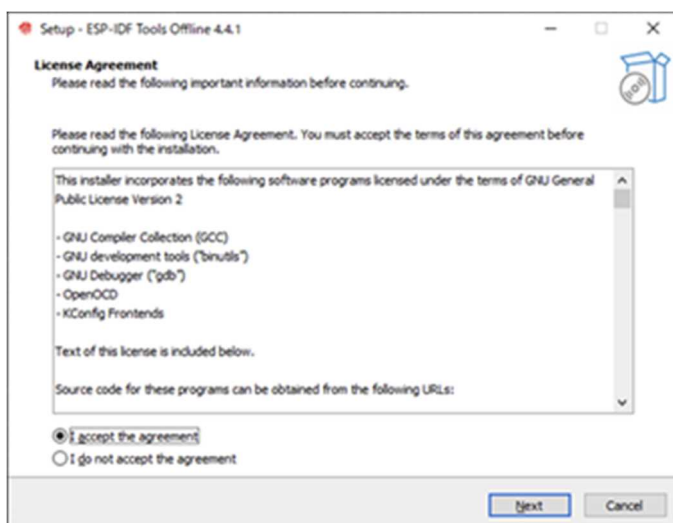
開発過程で使ったVerは4.4.1なので（新しいVerでも可能と思われませんが）少し下の方にスクロールしますと「Offline Installer v4.4.6」がありますのでこれをダウンロードします。ダウンロードが完了するとパソコンのダウンロードフォルダーに「esp-idf-tools-setup-offline-4.4.6.exe」がコピーされますので、これをダブルクリックで実行しますと下記のインストール画面が表示されます。

新しいVerで5.1.2などがありますが、コンパイルでエラーがでました・・・
Ver 4.4でコンパイル・書込をお願いします。

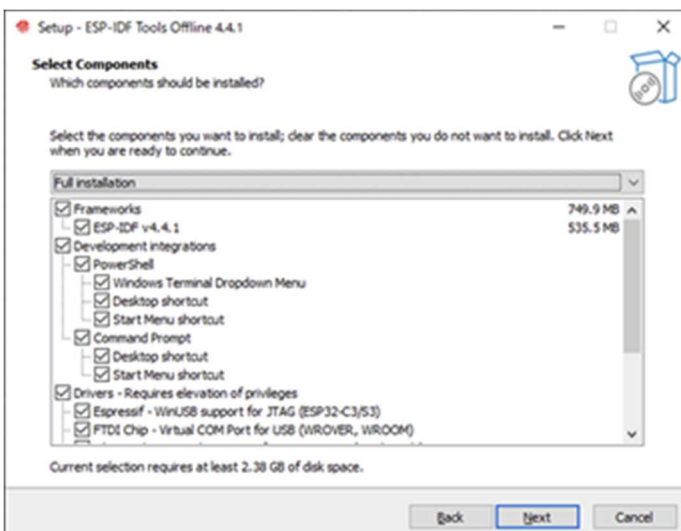
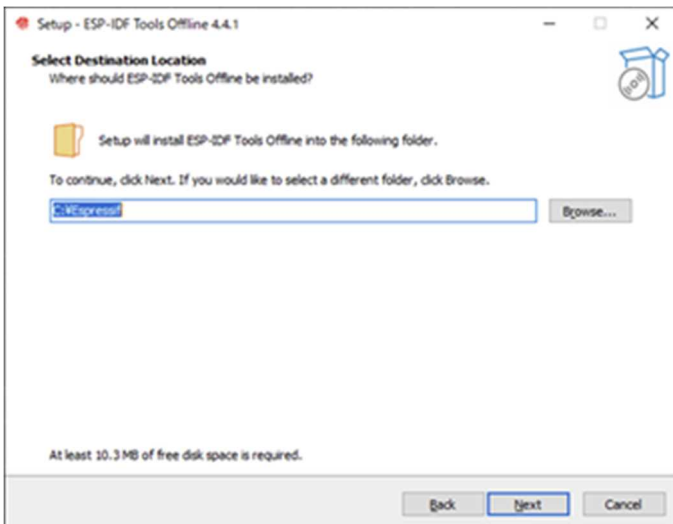
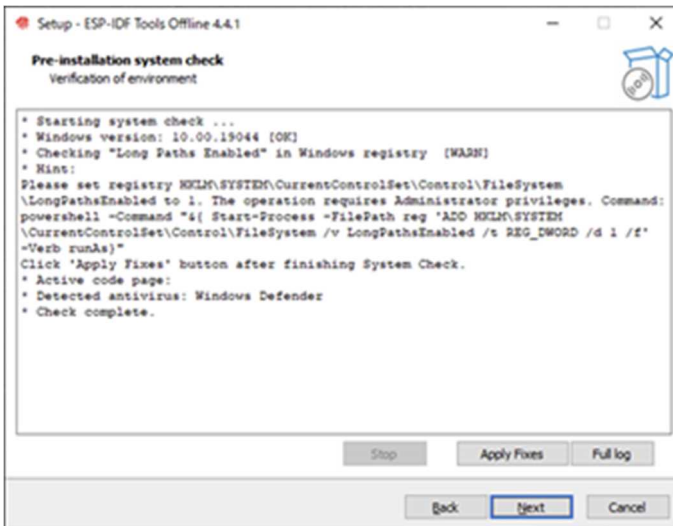
インストール



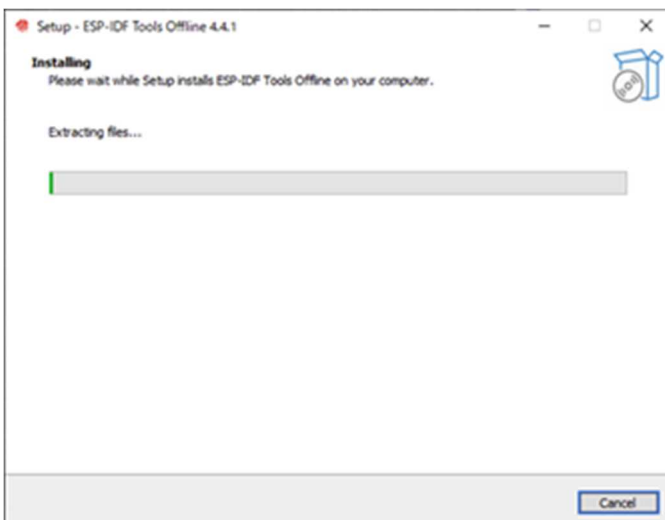
日本語がありませんので「English」を選択しておきます・・・「OK」をクリックすると



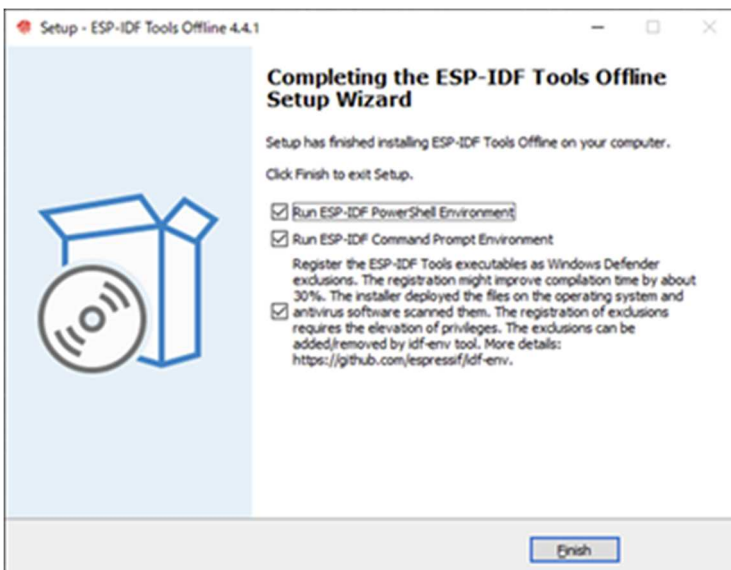
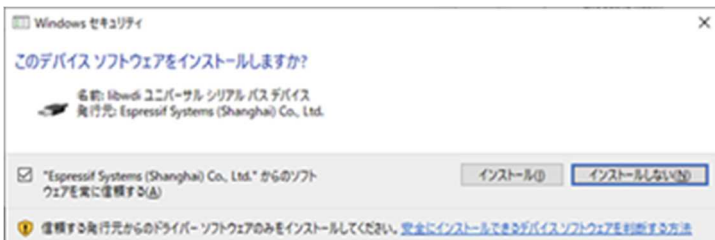
「I accept the...」を選択して「Next」・・・後はず～～と「Next」です



ここで「Install」をクリックするとインストールが始まります。



途中 USB ドライバー等のインストール確認がありますので「インストール」でドライバーも入れておきます。



この画面でインストール完了です。「Finish」をクリックすると Dos 窓で2つの環境が実行されます。どちらでも OK ですが「ESP-IDF 4.4 PowerShell」環境で作っています。

ESP-IDF の起動

```

added to PATH
.....
:YEspressif\frameworks\esp-idf-v4.4.1\components\esptool_py\esptool
:YEspressif\frameworks\esp-idf-v4.4.1\components\esp_update
:YEspressif\frameworks\esp-idf-v4.4.1\components\espcoredump
:YEspressif\frameworks\esp-idf-v4.4.1\components\partition_table
:YEspressif\tools\xtensa-esp32-elf\esp-2021r2-patch3-8.4.0\xtensa-esp32-elf\bin
:YEspressif\tools\xtensa-esp32s2-elf\esp-2021r2-patch3-8.4.0\xtensa-esp32s2-elf\bin
:YEspressif\tools\xtensa-esp32s3-elf\esp-2021r2-patch3-8.4.0\xtensa-esp32s3-elf\bin
:YEspressif\tools\triscv32-esp-elf\esp-2021r2-patch3-8.4.0\triscv32-esp-elf\bin
:YEspressif\tools\esp32ulp-elf\2.28.51-esp-20191205\esp32ulp-elf-binutils\bin
:YEspressif\tools\esp32s2ulp-elf\2.28.51-esp-20191205\esp32s2ulp-elf-binutils\bin
:YEspressif\tools\make\3.20.3\bin
:YEspressif\tools\openocd-esp32\v0.11.0-esp32-20211220\openocd-esp32\bin
:YEspressif\tools\minia\1.10.2\
:YEspressif\tools\idf.exe\1.0.3\
:YEspressif\tools\ocache\4.3\ocache-4.3-windows-64
:YEspressif\tools\idf\run\110.9\idf\run\11-0.9-win64
:YEspressif\frameworks\esp-idf-v4.4.1\tools
Checking if Python packages are up to date...
Python requirements from C:\E:\espressif\frameworks\esp-idf-v4.4.1\requirements.txt are satisfied.

Done! You can now compile ESP-IDF projects.
Go to the project directory and run:
idf.py build

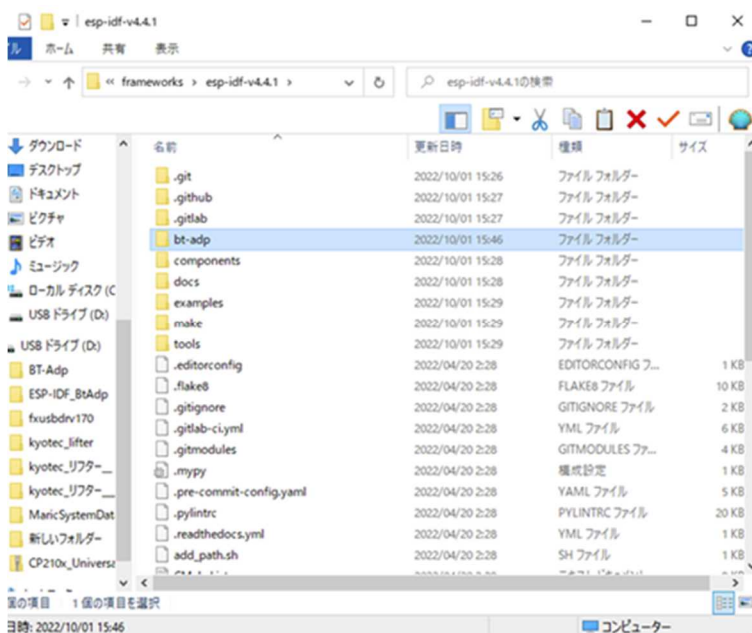
PS C:\E:\espressif\frameworks\esp-idf-v4.4.1>

```

上記が PowerShell 上で実行されている画面です。この状態でほぼ動作は可能と思われますが、ドライバー等の動作も考慮して、念のため再起動しておくほうがいいかもしれません。再起動後はデスクトップに作成されている「ESP-IDF 4.4 PowerShell」をダブルクリックすることで上記の画面が表示されます。

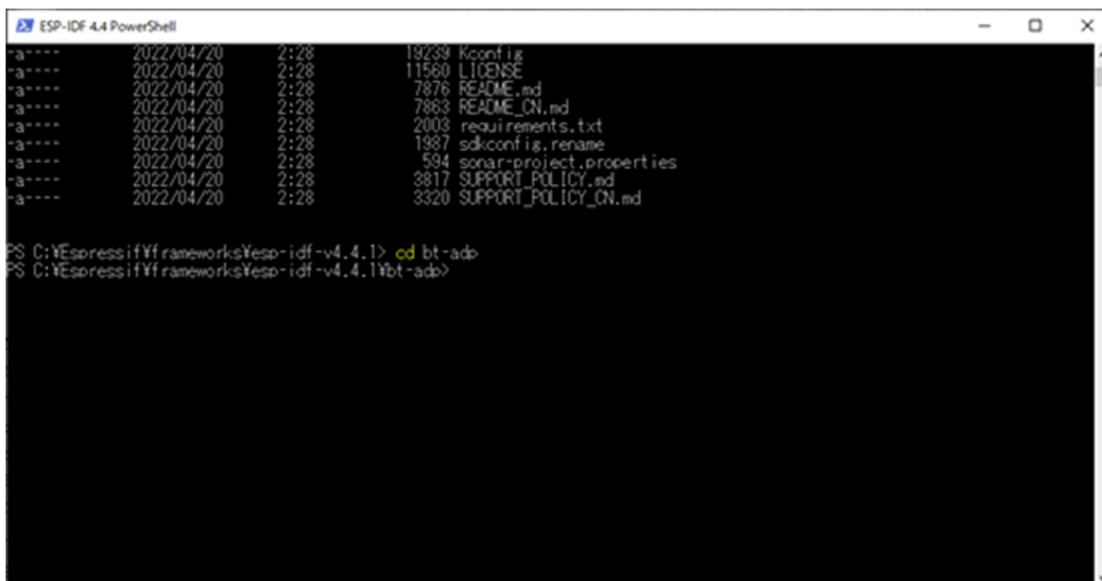
BT-Adp プロジェクト（フォルダー）のコピー

次に Esp-IDF の環境下に「bt-adp」のフォルダーをコピーしておきます。コピー先は「C:\E\espressif\frameworks」になります。ここには「examples」フォルダーもあり、このフォルダーにはいろいろなサンプルが収録されています。「BT-Adp」はこの「examples」>「bluetooth」>「bluedroid」>「classic_bt」にある「bt_spp_acceptor」と「bt_spp_initiator」から作成した物になります。下記の画面が「bt-adp」をコピーした状態です。



コンパイル・書き込み

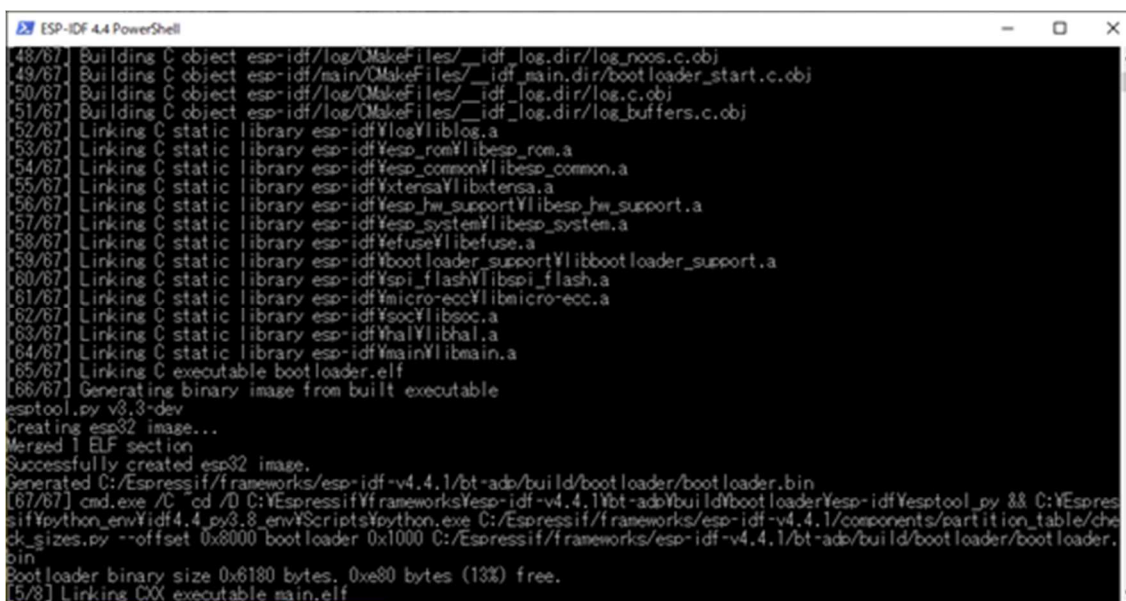
先の PowerShell 画面でこのコピーした「bt-adp」に入ります。
コマンドは「cd bt-adp」です。このコマンドで下記の様な画面になります。



```
ESP-IDF 4.4 PowerShell
-a----      2022/04/20      2:28      19239 Kconfig
-a----      2022/04/20      2:28      11560 LICENSE
-a----      2022/04/20      2:28      7876  README.md
-a----      2022/04/20      2:28      7883  README_CN.md
-a----      2022/04/20      2:28      2003  requirements.txt
-a----      2022/04/20      2:28      1987  sdkconfig.rename
-a----      2022/04/20      2:28         584 sonar-project.properties
-a----      2022/04/20      2:28      3817  SUPPORT_POLICY.md
-a----      2022/04/20      2:28      3320  SUPPORT_POLICY_CN.md

PS C:\Espressif\frameworks\esp-idf-v4.4.1> cd bt-adp
PS C:\Espressif\frameworks\esp-idf-v4.4.1\bt-adp>
```

この状態で「bt-adp」にはすでに大まかなオブジェクト（MakeConfig 等）は作成されていますので、PowerShell 上で「idf.py build」と入力してください。・・・「idf.py fullclean」を求められたらクリーンして再度ビルドしてください。

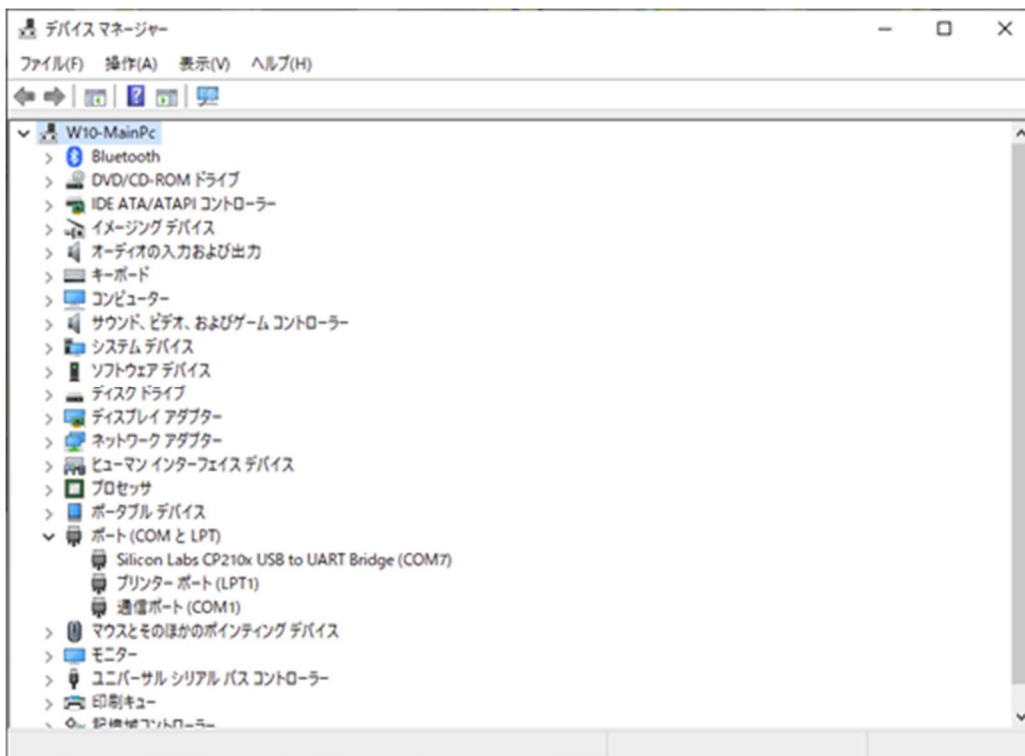


```
ESP-IDF 4.4 PowerShell
[48/67] Building C object esp-idf/log/Makefiles/_idf_log.dir/log_noos.c.obj
[49/67] Building C object esp-idf/main/Makefiles/_idf_main.dir/bootloader_start.c.obj
[50/67] Building C object esp-idf/log/Makefiles/_idf_log.dir/log.c.obj
[51/67] Building C object esp-idf/log/Makefiles/_idf_log.dir/log_buffers.c.obj
[52/67] Linking C static library esp-idf\log\liblog.a
[53/67] Linking C static library esp-idf\esp_rom\libesp_rom.a
[54/67] Linking C static library esp-idf\esp_common\libesp_common.a
[55/67] Linking C static library esp-idf\xtensa\libxtensa.a
[56/67] Linking C static library esp-idf\esp_hw_support\libesp_hw_support.a
[57/67] Linking C static library esp-idf\esp_system\libesp_system.a
[58/67] Linking C static library esp-idf\efuse\libefuse.a
[59/67] Linking C static library esp-idf\bootloader_support\libbootloader_support.a
[60/67] Linking C static library esp-idf\spi_flash\libspi_flash.a
[61/67] Linking C static library esp-idf\micro-ecc\libmicro-ecc.a
[62/67] Linking C static library esp-idf\soc\libsoc.a
[63/67] Linking C static library esp-idf\hal\libhal.a
[64/67] Linking C static library esp-idf\main\libmain.a
[65/67] Linking C executable bootloader.elf
[66/67] Generating binary image from built executable
esptool.py v3.3-dev
Creating esp32 image...
Merged ELF section
Successfully created esp32 image.
Generated C:/Espressif/frameworks/esp-idf-v4.4.1/bt-adp/build/bootloader/bootloader.bin
[67/67] cmd.exe /C cd /D C:\Espressif\frameworks\esp-idf-v4.4.1\bt-adp\build\bootloader\&& C:\Espressif\python_env\idf4.4_py3.8_env\Scripts\python.exe C:/Espressif/frameworks/esp-idf-v4.4.1/components/partition_table/check_sizes.py --offset 0x8000 bootloader 0x1000 C:/Espressif/frameworks/esp-idf-v4.4.1/bt-adp/build/bootloader/bootloader.bin
Bootloader binary size 0x6180 bytes, 0xe80 bytes (13%) free.
[5/8] Linking CXX executable main.elf
```

上記の様にビルドが実行され書き込むイメージが作成されます。・・・しばらく時間がかかります。
最後に「run idf.py -p (PORT)flash」と表示されれば完了です。

ビルド失敗する様でしたら「bt-adp」フォルダーの「build」フォルダーを削除してビルドしてみてください。

まず「ESP32-DeviceKit」を USB に接続して（USB ケーブルはご用意ください）COM ポートを確認しておきます。COM ポートのドライバーは最初のインストール時に組み込まれていますから、デバイスマネージャーで確認すれば下記の様に COM 番号が表示されます。

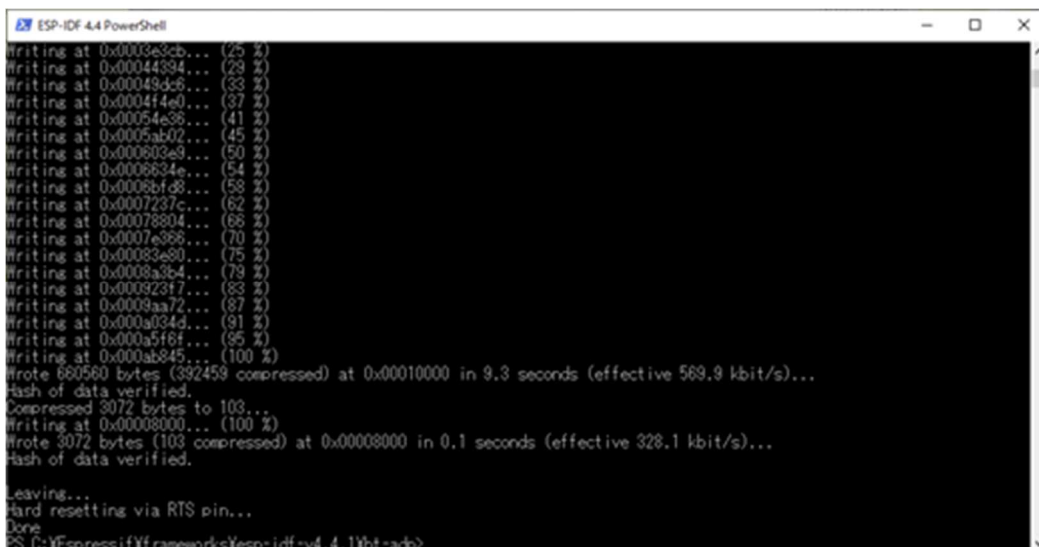


画面では COM7 に割り付けられていますが、パソコンの状況ではどの番号になるか不定です。この COM 番号「COM * (1桁)」もしくは「COM * * (2桁)」が確定すれば、PowerShell 上で「idf.py -p com7 flash」と入力してください。このコマンドの「com7」は確認した「com *」もしくは「com * *」に置き換えてください。

```
ESP-IDF 4.4 PowerShell
Successfully created esp32 image.
Generated C:/Espressif/frameworks/esp-idf-v4.4.1/bt-adp/build/main.bin
[8/8] cmd.exe /C cd /D C:/Espressif/frameworks/esp-idf-v4.4.1/bt-adp/build/main.bin
main.bin binary size 0xa1450 bytes. Smallest app partition is 0x100000 bytes. 0x5ebb0 bytes (37%) free.

Project build complete. To flash, run this command:
C:\Espressif\python_env\idf4.4_py3.8_ernv\Scripts\python.exe ..\components\esptool_py\esptool.py -p (PORT) -b 480800 --before default_reset --after hard_reset --chip esp32 write_flash --flash_mode dio --flash_size detect --flash_freq 40m 0x10000 bootloader/bootloader.bin 0x8000 build\partition_table\partition-table.bin 0x10000 build\main.bin
or run 'idf.py -p (PORT) flash'
PS C:\Espressif\frameworks\esp-idf-v4.4.1\bt-adp> idf.py -p com7 flash
Executing action: flash
Running ninja in directory c:\Espressif\frameworks\esp-idf-v4.4.1\bt-adp\build
Executing 'ninja flash...'
[1/5] cmd.exe /C cd /D C:/Espressif/frameworks/esp-idf-v4.4.1/bt-adp/build/main.bin
main.bin binary size 0xa1450 bytes. Smallest app partition is 0x100000 bytes. 0x5ebb0 bytes (37%) free.
[2/5] Performing build step for 'bootloader'
[1/1] cmd.exe /C cd /D C:/Espressif/frameworks/esp-idf-v4.4.1/bt-adp/build/bootloader\esp-idf\esptool_py && C:\Espressif\python_env\idf4.4_py3.8_ernv\Scripts\python.exe C:/Espressif/frameworks/esp-idf-v4.4.1/components/partition_table/check_sizes.py --offset 0x8000 bootloader 0x10000 C:/Espressif/frameworks/esp-idf-v4.4.1/bt-adp/build/bootloader/bootloader.bin
bootloader binary size 0x6180 bytes. 0xe80 bytes (13%) free.
[2/3] cmd.exe /C cd /D C:/Espressif/frameworks/esp-idf-v4.4.1/components/esptool_py/run_serial_tool.cmake
esptool.py esp32 -p com7 -b 480800 --before=default_reset --after=hard_reset write_flash --flash_mode dio --flash_freq 40m --flash_size 4M 0x10000 bootloader/bootloader.bin 0x10000 main.bin 0x8000 partition_table/partition-table.bin
esptool.py v3.3-dev
Serial port com7
Connecting...
Failed to get PID of a device on com7, using standard reset sequence.
```

上記がコマンド「idf.py -p com7 flash」を入力して実行した画面で、下記が書き込み完了した画面です。

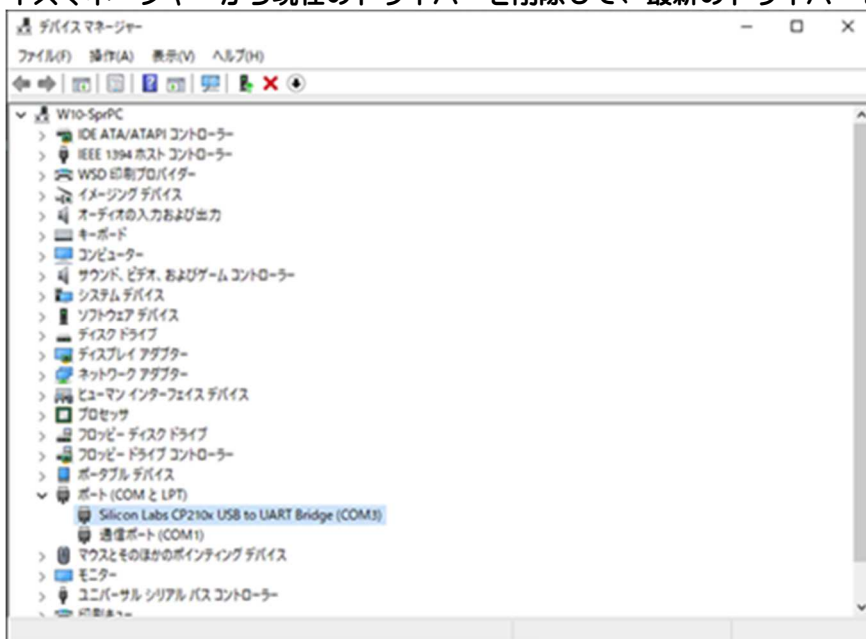


この状態で USB ケーブルを外して BT-Adp の JP1 はすべて解放・JP2 は 1 - 3 を短絡で ESP32-DeviceKit を差し込んで、電源を投入すれば赤 LED が点滅で接続待機になります。ESP-IDE の場合は先の Arduino の場合と違ってデバイス名は「BT-AP_A0」となります。

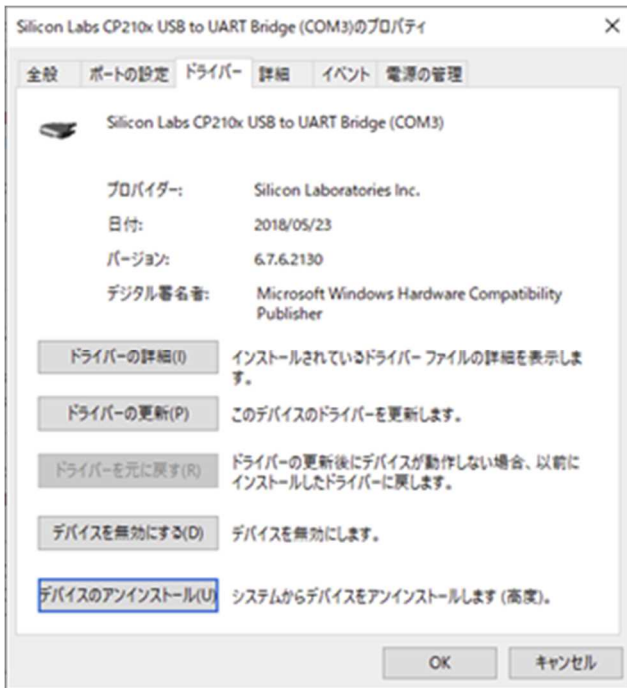
RS232C 側の設定は JP1 すべて解放で SD-CNT (TC-AH4、TC-FC4) に合わせて 4800bps、8bit、stop1 の ICOM 設定になります。他 JP1 の設定では通信速度が 9600・38400・115200、ストップビットが 1bit・2bit、機能でアクセプター（接続される側）・イニシエータ（接続する側）・接続先は BT-ADP_A0、BT-ADP_A1)、デバイス名（アクセプター>BT-ADP_A0、BT-ADP_A1・イニシエータ>BT-ADP_I0、BT-ADP_I1)の選択が出来る様にソースコードが書かれています。デバイス名の「BT-ADP_I0」は「BT-ADP_A0」に接続、「BT-ADP_I1」は「BT-ADP_A1」に接続します。他予備の設定で 2 個のジャンパー空きがあります。必要に応じてソースコードを変更してください。

Arduino・ESP-IDF での書き込み失敗の場合

もし下記の様に失敗等があれば・・・多分 USB のドライバーがうまく動作していないかもしれませんので、デバイスマネージャーから現在のドライバーを削除して、最新のドライバーをインストールしてみてください。



この項目を右クリックで「プロパティ」を開きます。



デバイスの「アンインストール」クリックで

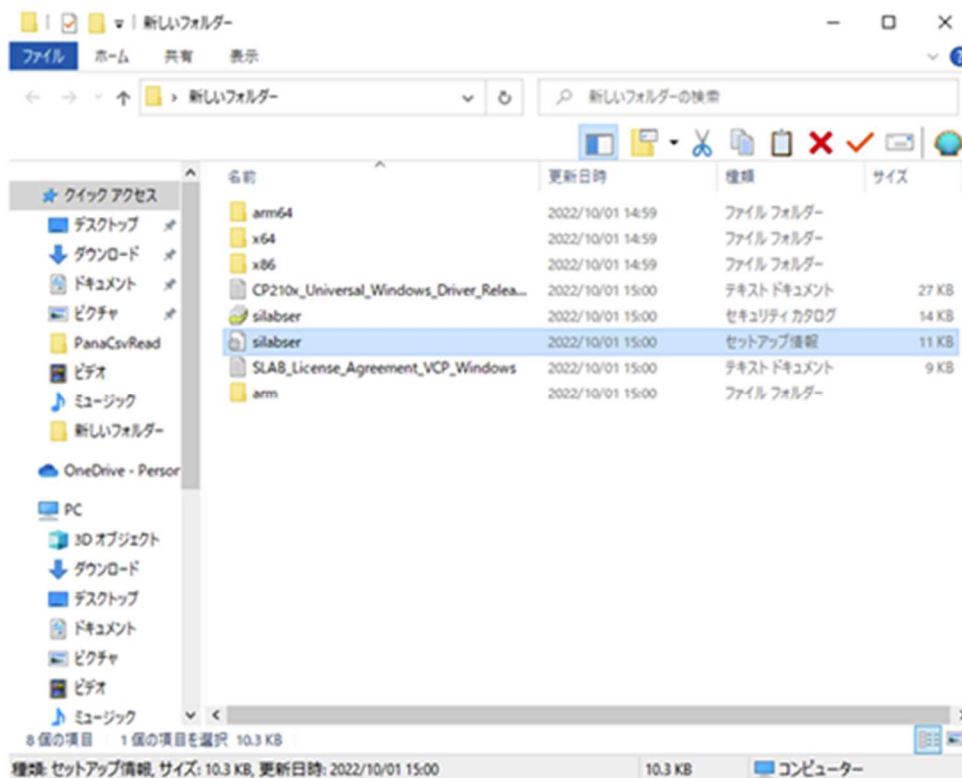


「このデバイスのソフトウェアを削除します」にチェックを入れて「アンインストール」クリックでドライバとデバイスは削除されます。

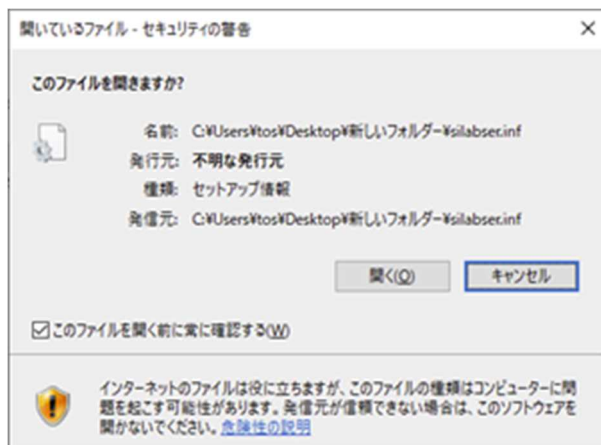
最新のドライバーは「<https://jp.silabs.com/developers/usb-to-uart-bridge-vcp-drivers?tab=downloads>」をブラウザで開いて頂くと



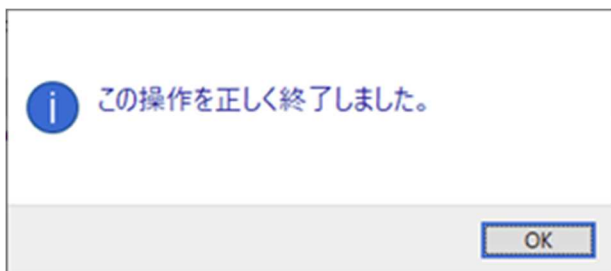
上記の画面で「CP210x Universal Windows Driver」をクリックするとパソコンの「ダウンロード」フォルダーに「CP210x_Universal_Windows_Driver.zip」がコピーされますので、これを開いて適当なフォルダーにzip内のファイル等をコピーします。



この中の「silabser」（セットアップ情報）を右クリックして、インストールをクリックすると新しいドライバーがインストールされます。



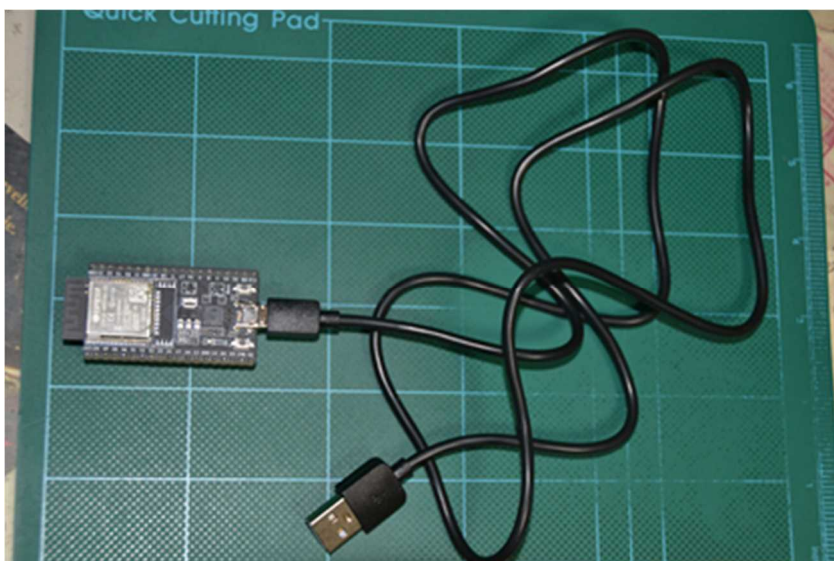
「開く」をクリックするとインストールが実行されます。Windowsでのインストール実行の許可が表示されましたら許可してください。



この画面で完了になります。

もう一度 ESP-IDF からの書き込みを実行してみてください。
(念のため Windows の再起動はした方がいいかもしれません)

ESP32-DeviceKit へのプログラム書込接続イメージ



この状態で USB をパソコンへ接続すれば OK です。
ESP32 側は「micro-B」の仕様です。

書き込み完了後 BT-Adp 基板に差し込んでください。

2022. 10. 10 Ver2. 00
キット形式で初版をリリース
2023. 04. 10 Ver2. 00a
ESP-IDF でビルド時失敗する場合の対応追記
2023. 04. 11 Ver2. 00b
基板への ESP32 装着状態等写真説明追記
2023. 04. 17 Ver2. 00c
Arduino 環境で「ボードの選択」説明を追記
2024. 02. 11 Ver2. 00d
説明書を更新
Arduino IDEでの記述に更新
ESP-IDFではVer 5. 1ではエラーの発生があることを確認