

robbit-esp 開発マニュアル

制作日：2025年9月

robbit-esp : 概要

- robbit-espはESP32-C3で制御する扱いやすいtwo-wheeled self-balancing robotである。

- 開発手順

robbit-espの組み立て



プログラム書き込み

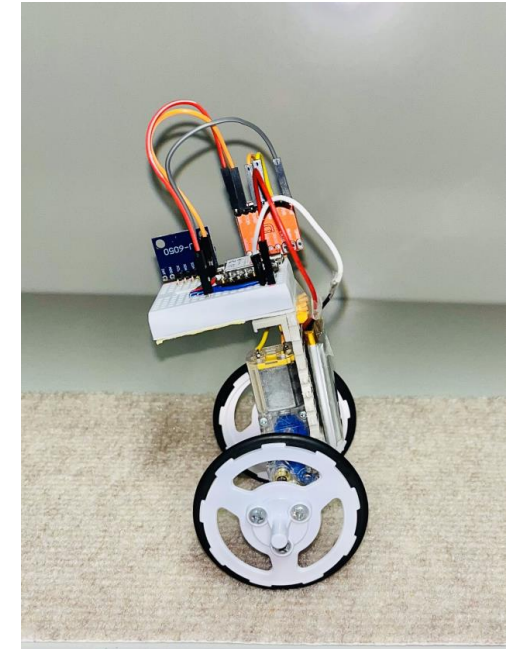
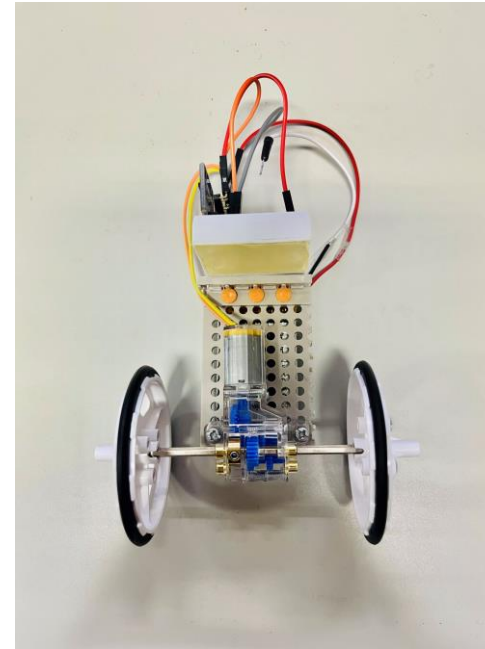


動作確認



パラメータチューニング

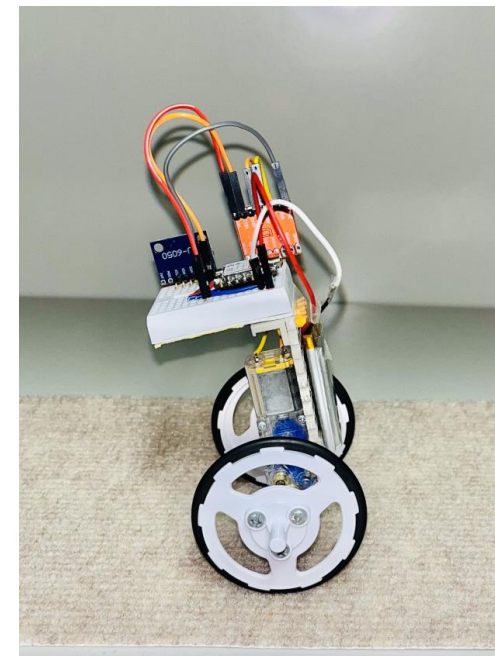
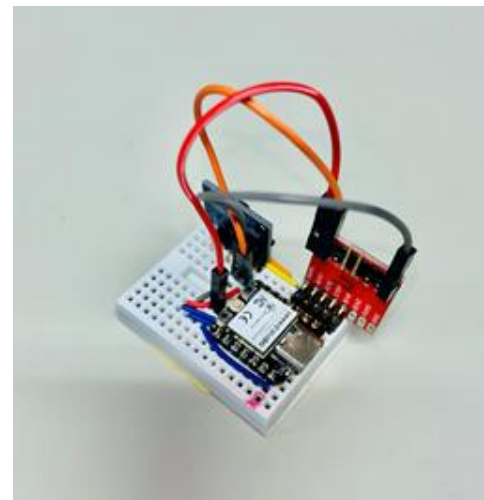
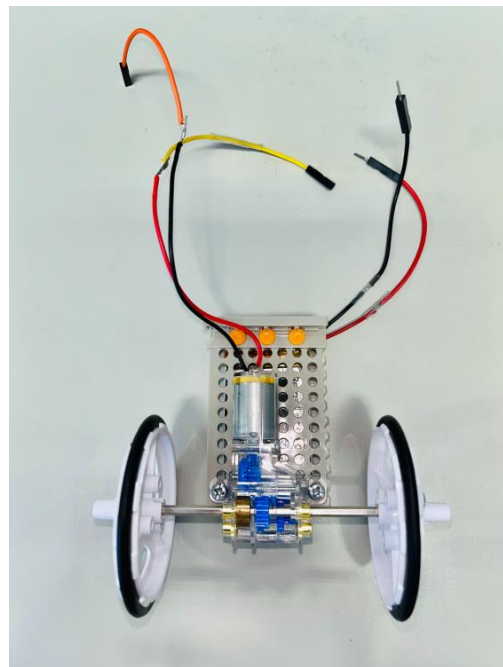
robbit-esp



robbit-espの組み立て：部品購入

■ robbit-espの組み立ては以下の手順で行う

部品購入 ➡ シャーシ作成 ➡ 制御モジュール作成 ➡ 接着



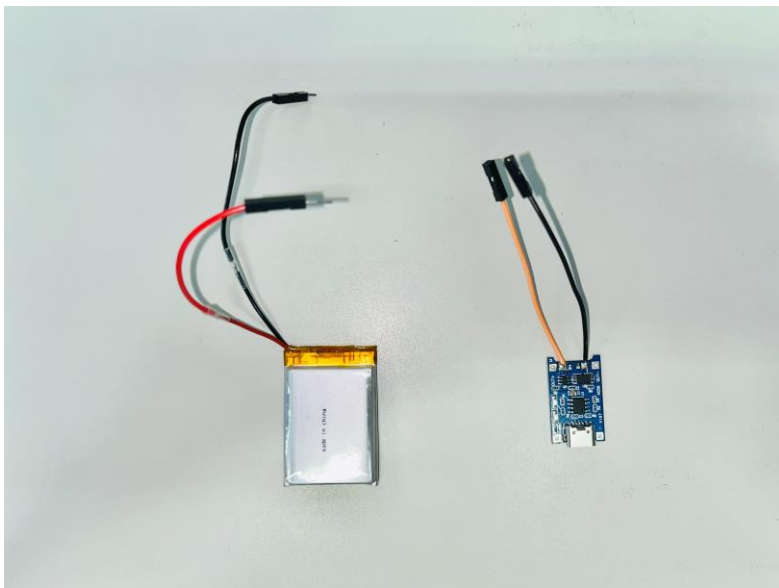
robbit-espの組み立て：部品購入

■ 下記の部品一覧にある部品を購入する

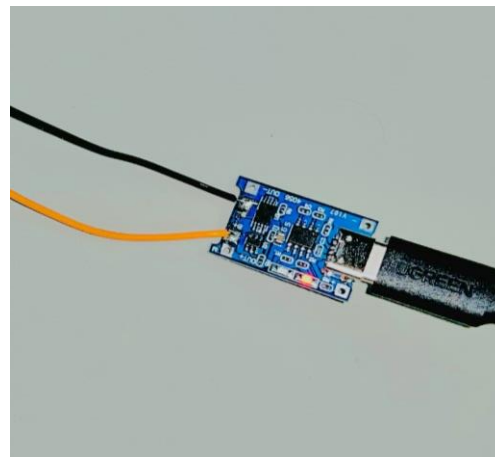
部品	名称	個数
マイクロコントローラ	XIAO ESP32-C3	1
センサ	MPU-6050	1
モータ	Mini Motor Standard Gearbox 70188	1
タイヤ	Slim Tire Set (55mm Dia.) 70193	1
モータドライバ	TB6612FNG	1
バッテリー	EEMB Lithium-Ion Battery 653042	1
プレート	Universal Plate Set 70157	1

robbit-espの組み立て：シャーシ作成

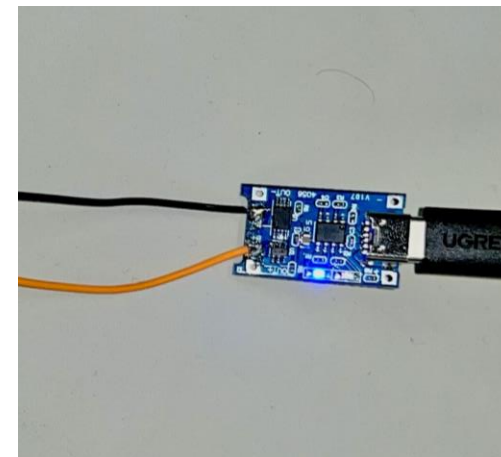
- バッテリーと電源モジュールを図のようにはんだ付けする
- はんだ付け後は、バッテリーを充電モジュールに接続する
- 充電モジュールの色でバッテリーの充電具合がわかる
(赤: 充電不足, 青: 充電完了)



はんだ付けの例



赤: 充電不足



青: 充電完了

充電モジュールのLED点灯

robbit-espの組み立て：シャーシ作成

- ユニバーサルプレートをはんだ付けする
- ユニバーサルプレートはこの後作る制御モジュールの大きさに合わせるとよい
- またユニバーサルプレートには下図のように、制御モジュールを接着する部分を用意しておく



切断後のユニバーサルプレート

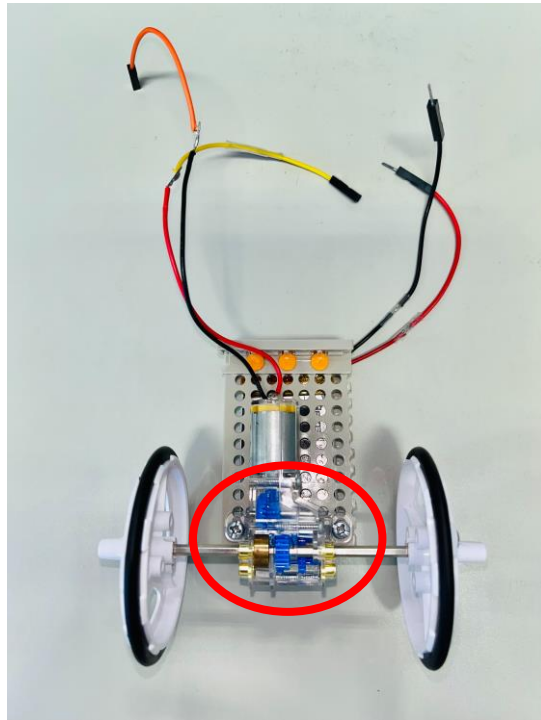
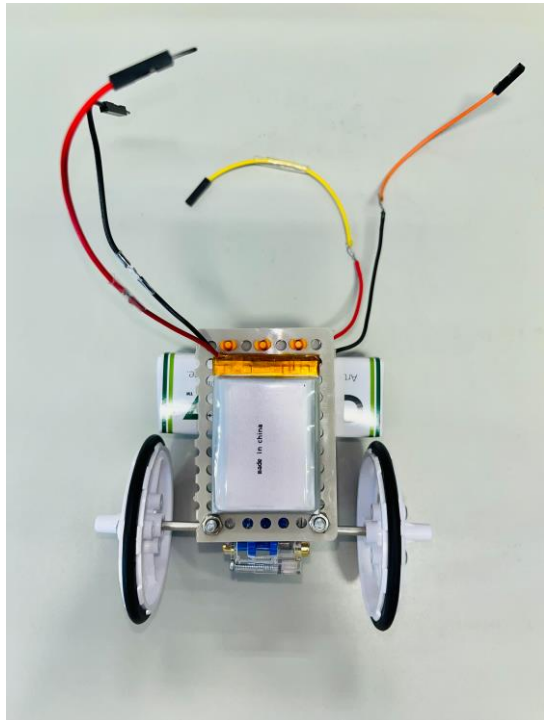


制御モジュールとの接続部分

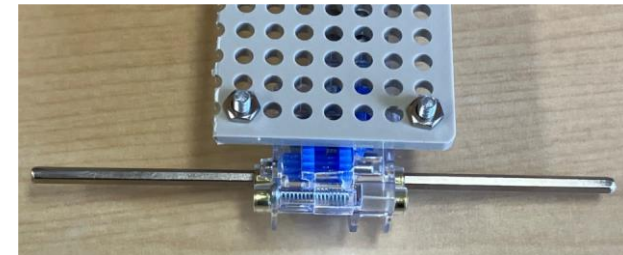
- TAMIYAのMini Motor Standard Gearbox 70188とSlim Tire Set (55mm Dia.) 70193を作成する。
- 次ページにギアボックス組み立てマニュアルを載せている。必要な部品や作業には黄色で色付けしている
- タイヤに関しては、最初は直径が55mmのタイヤを使用すると良い。
直径が小さいタイヤも同封されているが、制御が難しくなる場合がある。

robbitの組み立て：シャーシ作成

- モーターが完成したらタイヤを取り付ける
- 最後にユニバーサルプレートにバッテリーとモーターを取り付けるとシャーシが完成する。



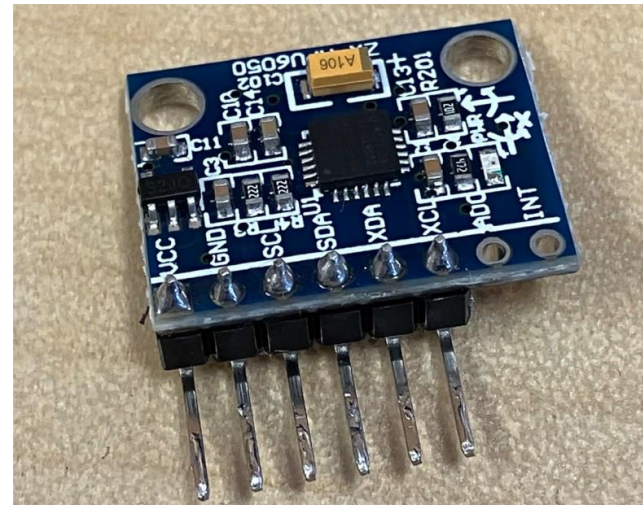
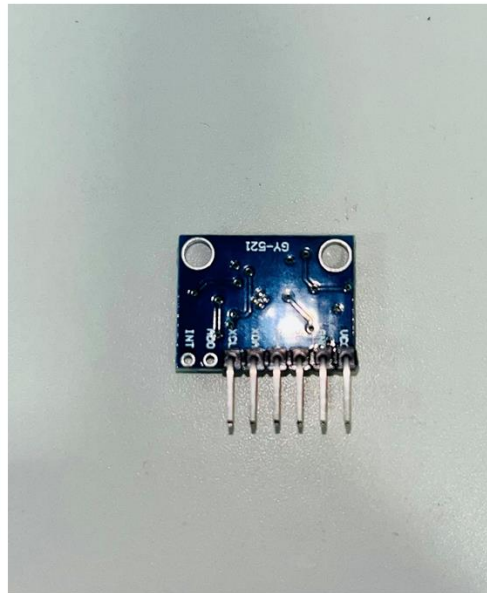
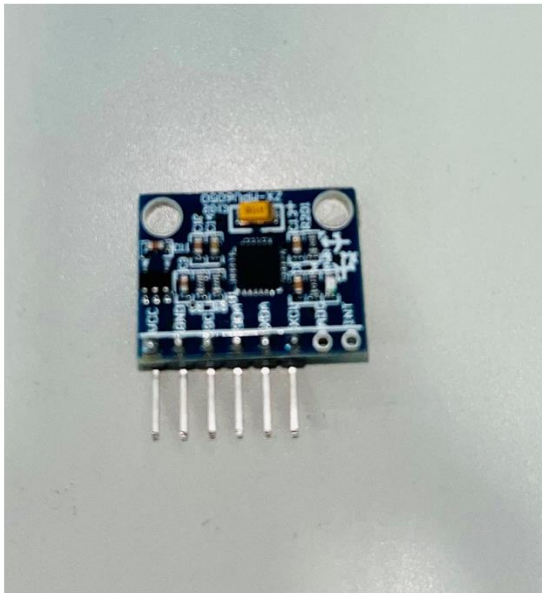
ネジでしっかり固定



シャーシの完成例

robbitの組み立て：制御モジュール作成

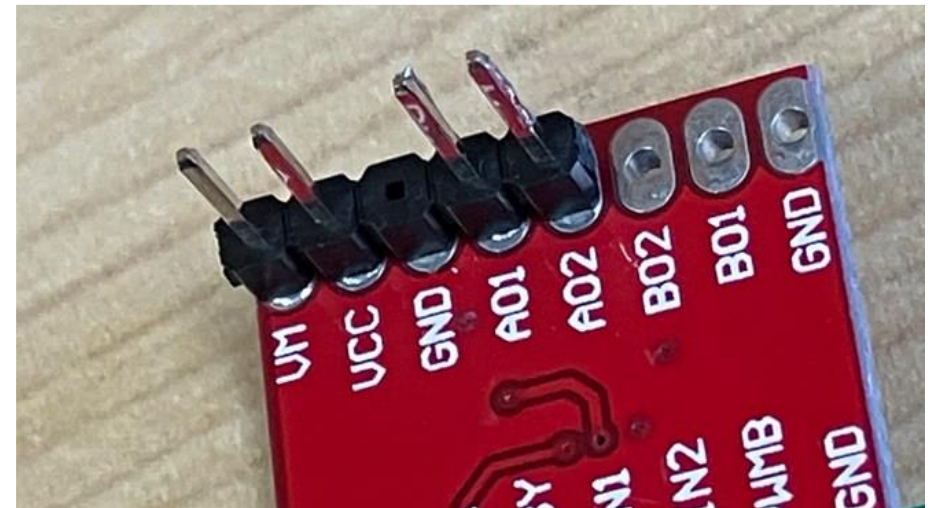
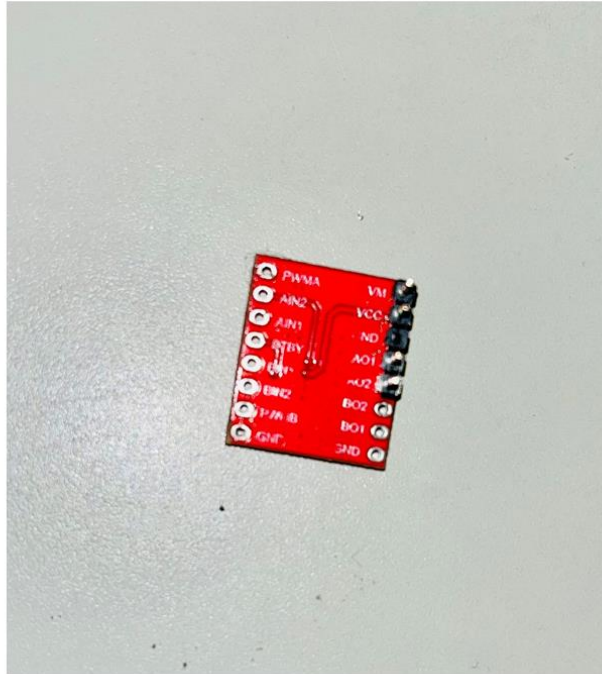
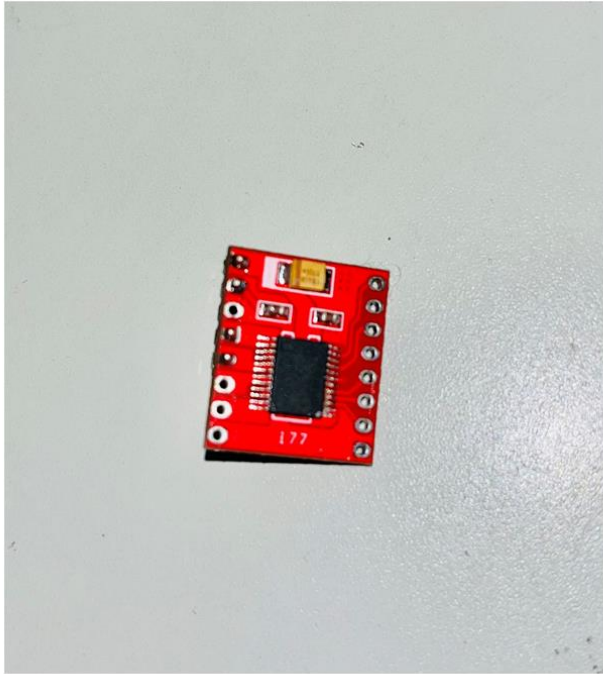
- センサ(MPU-6050)にピンヘッダをはんだ付けする。IMUモジュールはVCC, GND, SCL, SDAをつなげば動作するが、安定性向上のため接続部分にはすべてヘッダピンをはんだ付けすることを勧める。



はんだ付けの例

robbitの組み立て：制御モジュール作成

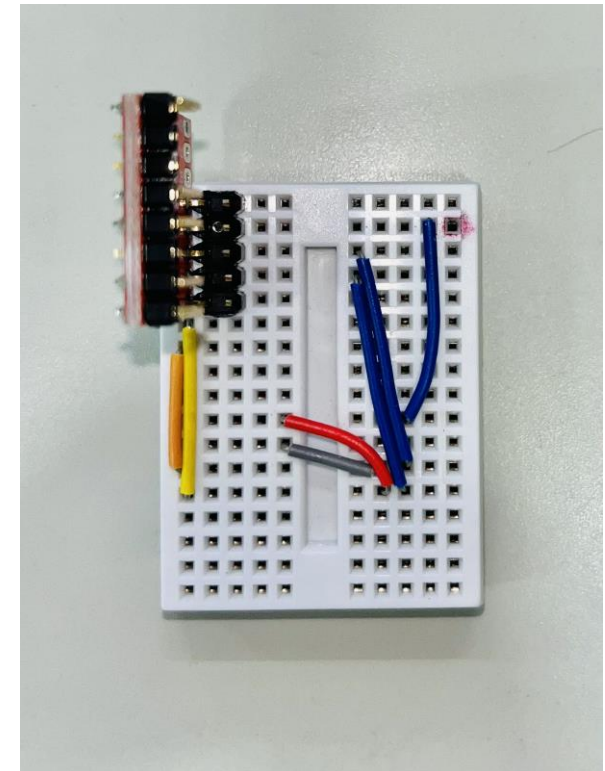
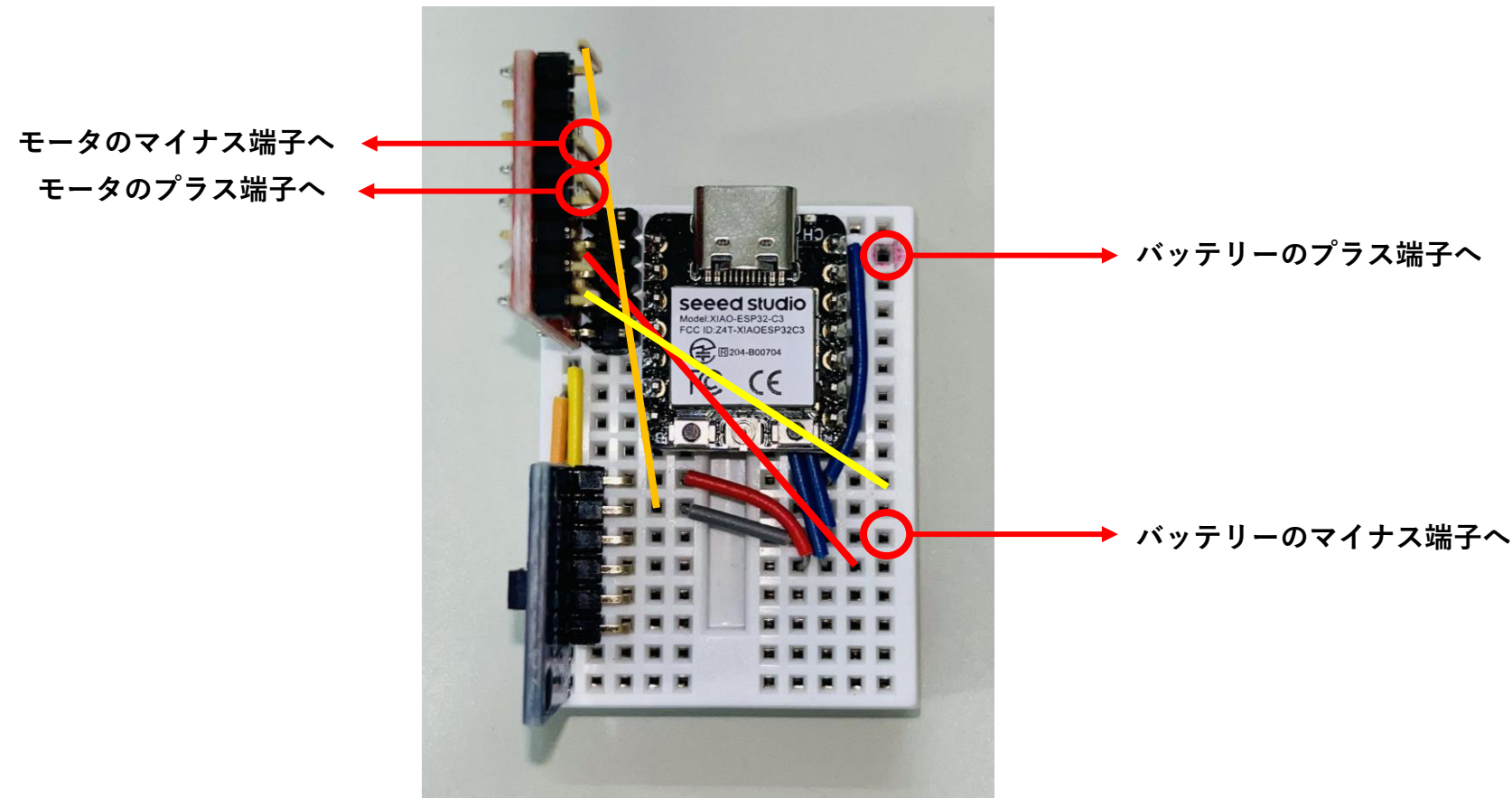
- モータドライバ(TB6612FNG)のVM, VCC, GND, A01, A02にはんだ付けをする
- ただし, GNDのピンは使用しないので取り除く



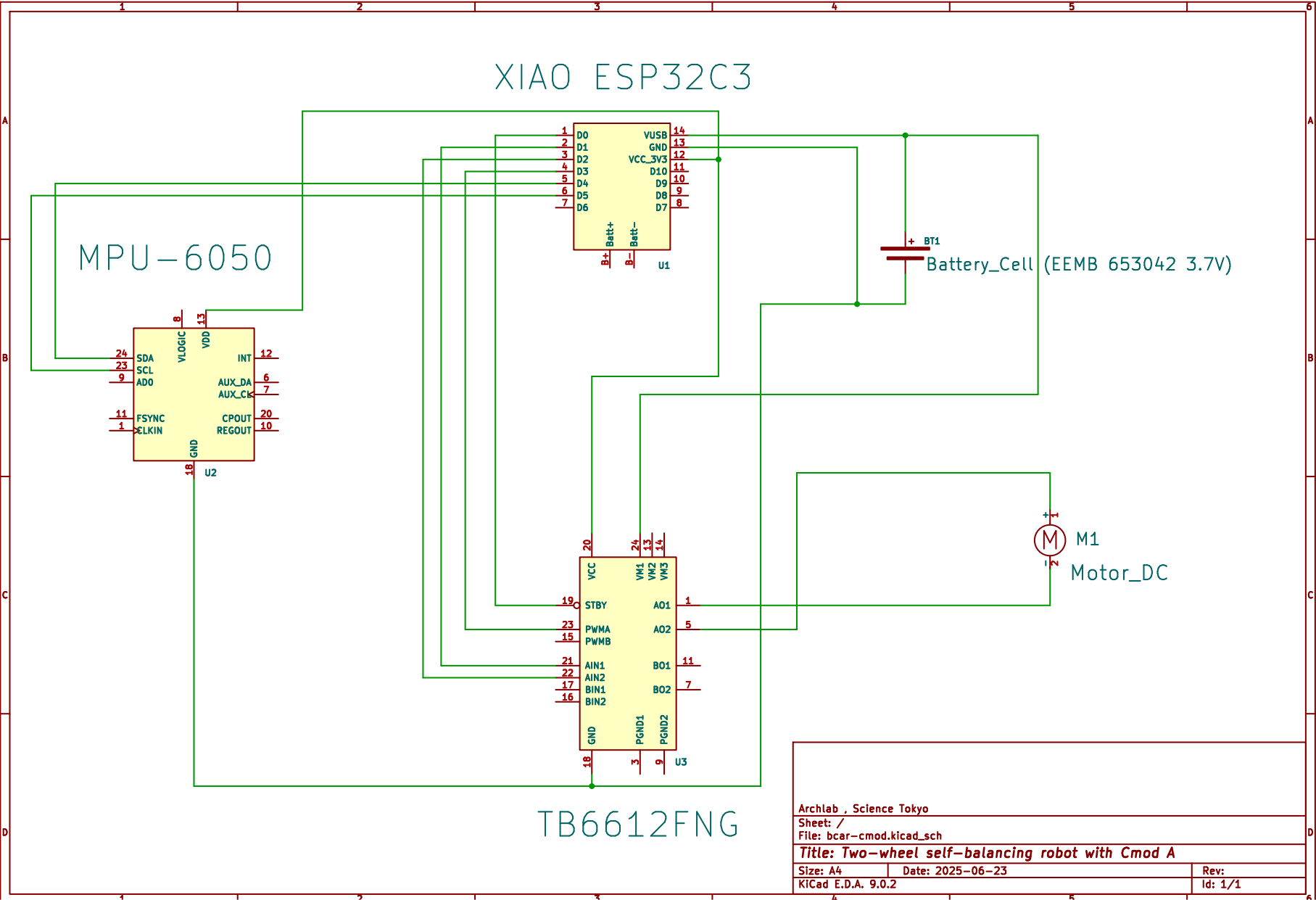
はんだ付けの例

robbit-espの組み立て：制御モジュール作成

- ブレットボードまたは、ユニバーサル基盤を用意する。
- 用意したブレットボードまたは基盤に回路図を参考に配線、はんだ付けをする。

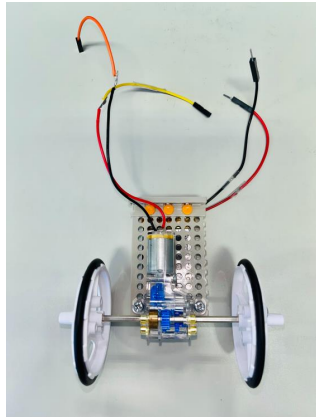


robbit-espの組み立て：制御モジュール作成

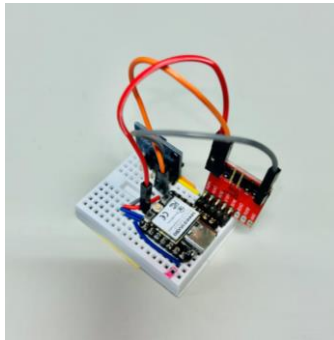


robbit-espの組み立て：制御モジュール作成

- 完成したシャーシと制御モジュールを両面テープ等で接着すると、robbit-espが完成する。

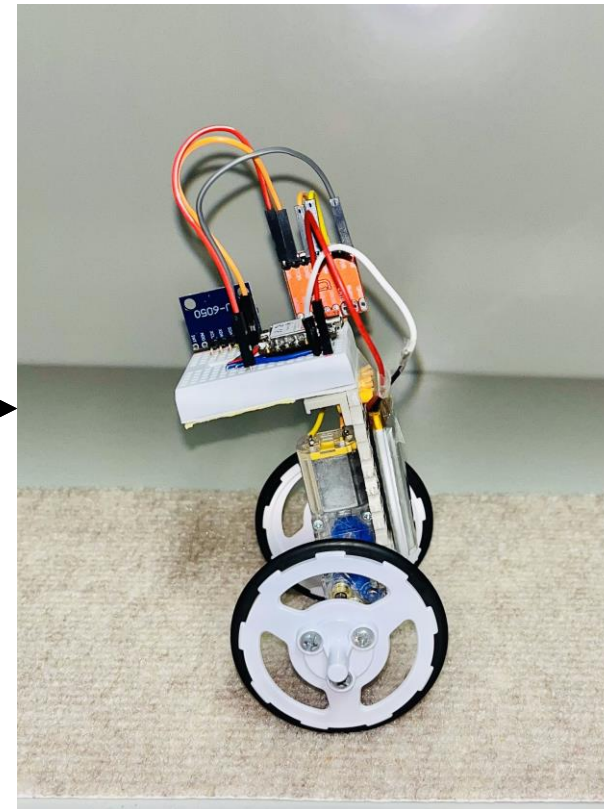


シャーシ



制御モジュール

接着



robbit-esp

プログラム書き込み: 環境構築

- 完成したrobbit-espにプログラムを書き込むにはArduino IDEを使用する
ダウンロードしていない場合はダウンロードする



Arduino IDE 2.3.6

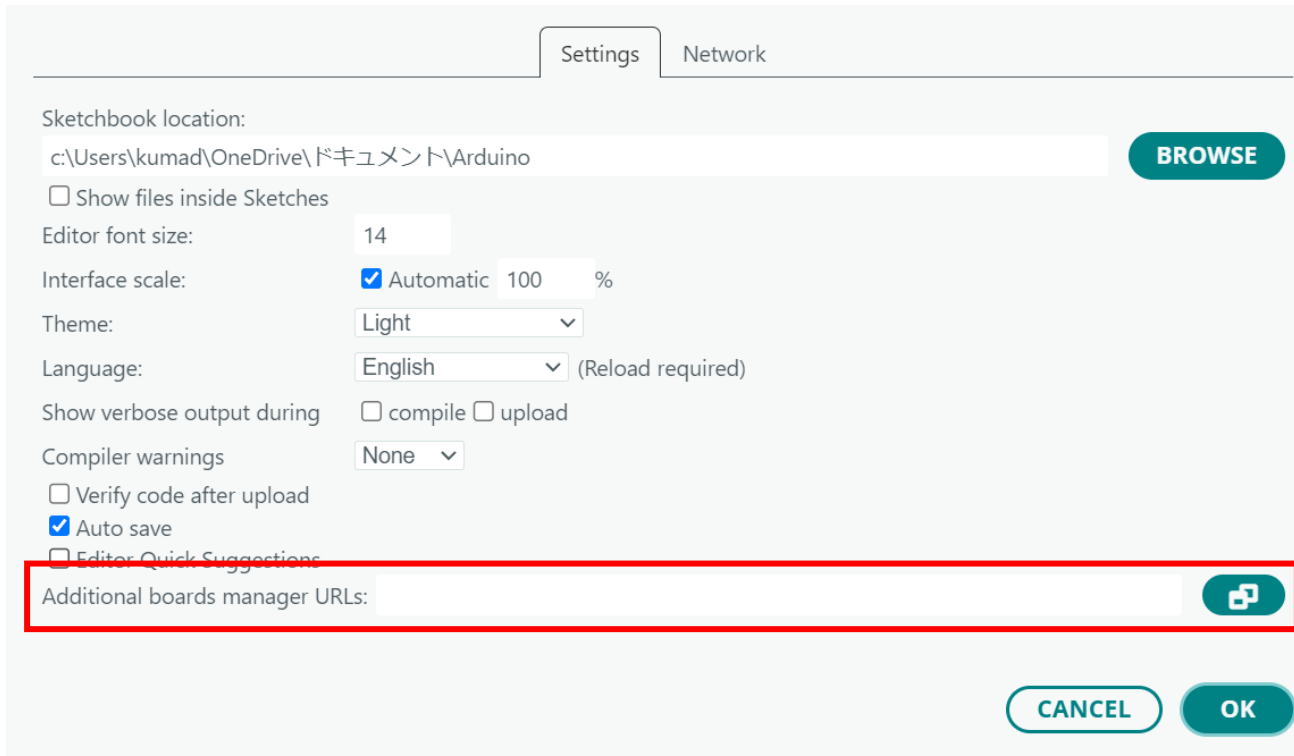
[Release notes](#)

- ダウンロードページ: <https://www.arduino.cc/en/software/>
- Arduino IDEのダウンロードが完了したら、githubのリポジトリにあるrobbit-esp.inoを開き、各種設定を行う。

プログラム書き込み: ボード設定

- `File` -> `Preference` -> `Setting` -> `Additional boards manager URLs`
に以下のURLを設定する

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



Settings Network

Sketchbook location:
c:\Users\kumad\OneDrive\ドキュメント\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: None

☐ Verify code after upload

☒ Auto save

☐ Editor Quick Suggestions

Additional boards manager URLs: **Copy**

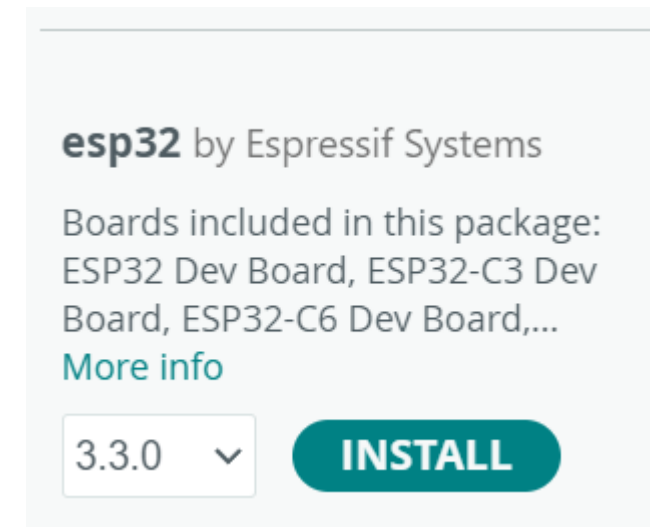
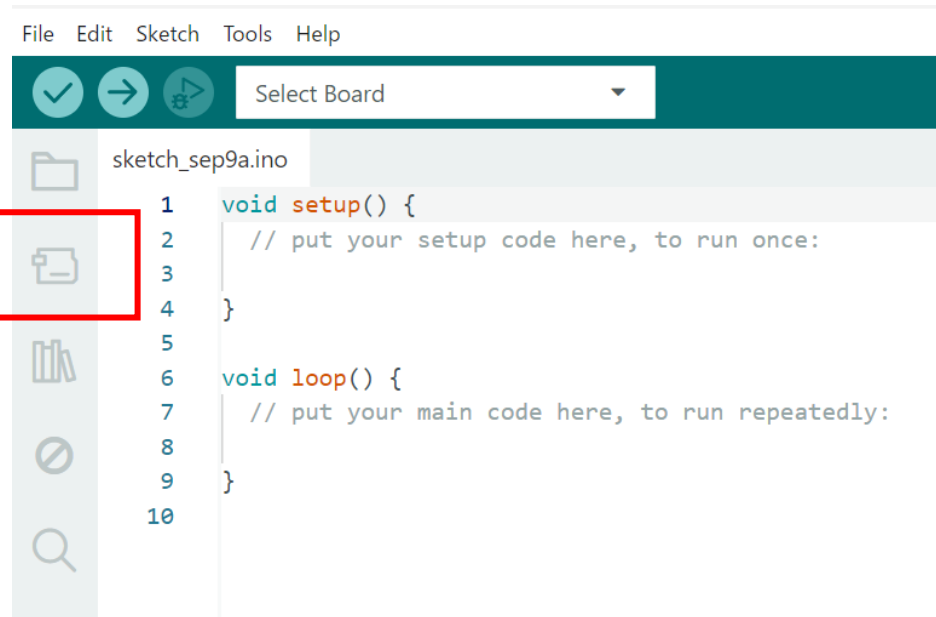
CANCEL OK

URLの記入場所

プログラム書き込み: ボード設定

- `Tools` -> `Board` -> `Board Manager` でBoard Managerを開き `esp32 by Espressif Systems` をダウンロードする
- `Tools` -> `Board` -> `esp32` -> `XIAO_ESP32C3` を選択する
- `Tools` -> `Port` でESP32-C3に対応するポート番号を設定する

ここからもボードマネージャーを開ける



インストールするボード

プログラム書き込み: ライブラリ設定

- `Tools` -> `Manage libraries` でライブラリマネージャーを開き以下のライブラリをインストールする

- MPU6050 by Electric Cats
- Madgwick by Arduino
- BluetoothSerial by Henry Abrahamsen

ここからもライブラリ
マネージャーを開ける



- Madgwickフィルタのライブラリ内にMadgwickAHRS.hというヘッダーファイルがあるので以下の関数を追加する

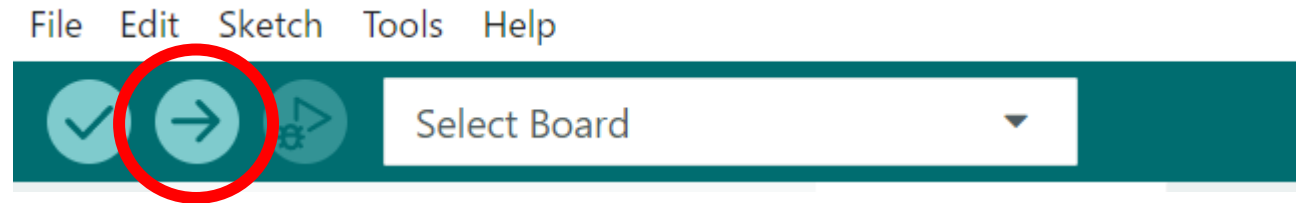
```
void setGain(float gain) { beta = gain; }
```

プログラム書き込み: 書き込み

■ 書き込む前に以下のことを確認する

- PCとESP32-Cが接続する
- `Tools` -> `Port` でESP32-C3に対応するポート番号を設定する
- `Tools` -> `Board` -> `esp32` -> `XIAO_ESP32C3` を選択する

■ 上記の内容をすべて確認出来たら画面左上のUploadマークを押すとコンパイルが行われた後、書き込みが行われる。



Uploadボタン

- 書き込みが終了したら、PCとESP32-C3の接続を解除し、robbitの電源を入れて動作を確認する。
- 動作確認はカーペットのようなある程度摩擦が生じる環境で行うと良い。
- 摩擦のない環境だと、その場で自立することが難しくなる。

動作確認の項目

- 手にもって、robbit-espを傾けた時に傾けた方向に車輪が回転するか
- 垂直状態から90度傾けた時に車輪の動きが止まるか(プログラムでは40度傾くと止まるようになっている)

- robbitのESP32-C3モデルはPCやスマートフォンとのBLE通信によりパラメータチューニングを行える。

■ PCの場合

- パラメータチューニングにはpythonライブラリのbleakを使用するため、仮想環境を作成する必要がある
- Pythonの仮想環境を作成した後、`pip install bleak`を実行しインストールを行う
-> 22ページへ

■ スマートフォンの場合

- BLE通信を行う専用のアプリをインストールする -> 25ページへ



BLE Scanner(Android, iOS両方に対応)

パラメータチューニング: BLE接続(PC)

- robbitを起動し、少し待ってからble_connect.pyを実行する。
- 実行すると、robbitとPCの無線での接続が行われ、成功すると下記の画面が出力される。
- この画面では、robbitに搭載するESp32-C3のサービスUUID(Service_uuid)、MACアドレス(MAC_adress)、Characteristic_uuidを表示している。
- Service_uuidとCharacteristic_uuidはinoファイルの冒頭で定義している値になっているはずである。

```
(venv) PS C:\Users\kumagai\ble-bcar> python .\ble_connect.py
connected 64:E8:33:89:F6:02
Service_uuid: 00001800-0000-1000-8000-00805f9b34fb
Service_uuid: 00001801-0000-1000-8000-00805f9b34fb
Service_uuid: 4fafc201-1fb5-459e-8fcc-c5c9c331914b
MAC_adress : 64:E8:33:89:F6:02
Characteristic_uuid : beb5483e-36e1-4688-b7f5-ea07361b26a8
input ("parameter number" "value") -> 
```

パラメータチューニング: 値変更(PC)

- パラメータの変更は`input ("parameter number" "value") -> `の後に以下のような形式で入力すると実行される。パラメータ番号については24ページ参照

"パラメータ番号" _ "値"
ただし、_ は半角スペース

- 下の画像はV_MAXというパラメータを200に変更した場合の例である。
変更が完了すると、変更結果が出力される

```
input ("parameter number" "value") -> 6 200  
write V_MAX = 200
```

パラメータチューニング: その他の命令

- 現状のパラメータを確認したい場合は、`input ("parameter number" "value")`
-> `に`r`を入力すると確認できる。

```
input ("parameter number" "value") -> r
*-----*
target : -12.10,
Kp : 900.00,
Ki : 3000.00,
Kd : 20.00,
pwm_base : 40.00,
V_MAX : 200.00
*-----*
```

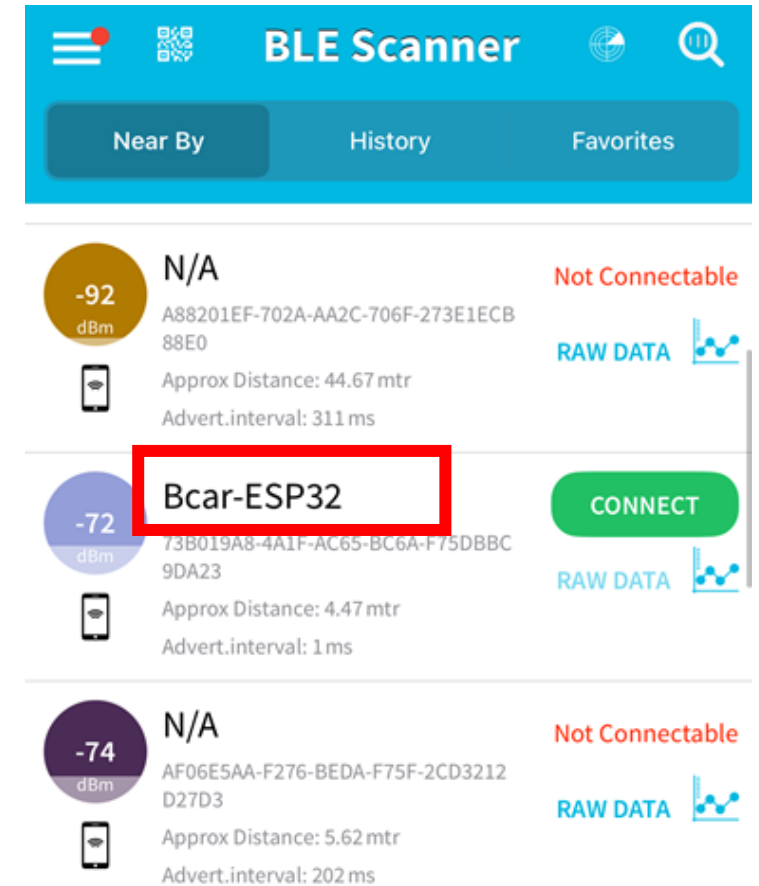
- robbitとのBLE接続を解除したい場合には、`input ("parameter number" "value")` -> `に`e`を入力すると確認できる。

パラメータチューニング: BLE接続(スマホ)

- Robbit-espを起動すると、BLE Scannerの画面にESP32-C3の接続先が現れる
- 接続端末の名前はinoファイル内の以下の部分で設定している
- 同時に複数のrobbit-espを動かす際には、それぞれ接続端末の名前を変更することを勧める

接続端末の名前設定

```
BLEDevice::init("Bcar-ESP32");  
BLEServer *pServer = BLEDevice::createServer();  
BLEService *pService = pServer->createService(SERVICE_UUID);
```

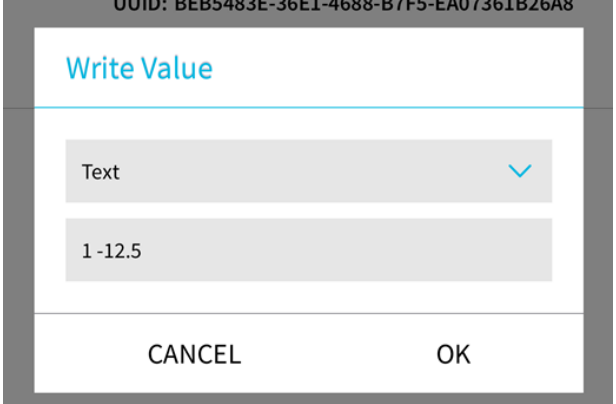


アプリの画面

パラメータチューニング: 値変更(スマホ)

- パラメータの変更は以下のような形式で入力すると実行される。
パラメータ番号については24ページ参照

"パラメータ番号" "値"
ただし、は半角スペース



The screenshot shows a mobile application interface with a dialog box titled "Write Value". At the top, there is a small text field containing a UUID: BEB5483E-36E1-4688-B7F5-EA07361B26A8. Below the title, there is a dropdown menu labeled "Text" with a blue checkmark icon. Underneath the dropdown, there is a text input field containing the value "1-12.5". At the bottom of the dialog, there are two buttons: "CANCEL" and "OK".

入力例

パラメータチューニング: その他の命令

- アプリ内でread処理を行うことですべてのパラメータを確認できる
- 表示形式はカンマ区切りで以下のように出力される。

TARGET, P_GAIN, I_GAIN, D_GAIN, PWM_Base, V_MAX

< Bcar-ESP32 DISCONNECT

Status CONNECTED

BONDED

CUSTOM SERVICE

^ 4FAFC201-1FB5-459E-8FCC-C5C9C331914B

PRIMARY SERVICE

CUSTOM CHARACTERISTIC R W

UUID: BEB5483E-36E1-4688-B7F5-EA07361B26A8

Properties: READ, WRITE

Value: -12.00, 900.00, 3000.00, 38.00, 40.00, 118.00

0x2D31322E30302C203930302E30302C20333030

Hex: 302E30302C2033382E30302C2034302E30302C2031382E30300A

Write Value: WRITE REQUEST



パラメーター値

- パラメータチューニングの際に利用するパラメータ番号とは、変更可能なパラメータに割り当てられている番号である

パラメータ名	パラメータ番号	役割
Target	1	車体の目標角度
P_gain	2	比例要素のゲイン
I_gain	3	積分要素のゲイン
D_gain	4	微分要素のゲイン
PWM_Base	5	PWM信号の増加値
V_max	6	PWM信号の最大値

- ループ周波数

ループ周波数とはmain.cpp内に記述されているwhileループの周波数を表す

main.cpp内の`LOOP_HZ`がループ周波数を表す

値を大きくすればその分whileループ内の処理も高速に行われる

単位はHzで記述する

```
#define LOOP_HZ      1000  // Hz of main loop
```

1000Hzに設定した例

- 積分要素の上限値

アンチwindアップ実装のため、積分要素の上限値を決定する

main.cppの`I_MAX`に値が設定してある

```
#define I_MAX      0.4  // Anti-windup
```

上限値を0.4に設定した例

パラメータチューニングの際にどの程度まで性能向上を目指すか、目標を考える
下記は一例である

■ 難易度：易しい

カーペットなどの摩擦のある領域で10分以上の自立を目指す

■ 難易度：難しい

摩擦の少ない領域でできるだけ長く自立させる

風や傾斜などの外乱を加える