STEM Duコントローラ RDC-ESP32 TYPE 1・2・3 スマホアプリ Dabble による無線コントローラの導入

九州国際大学
水井研究室

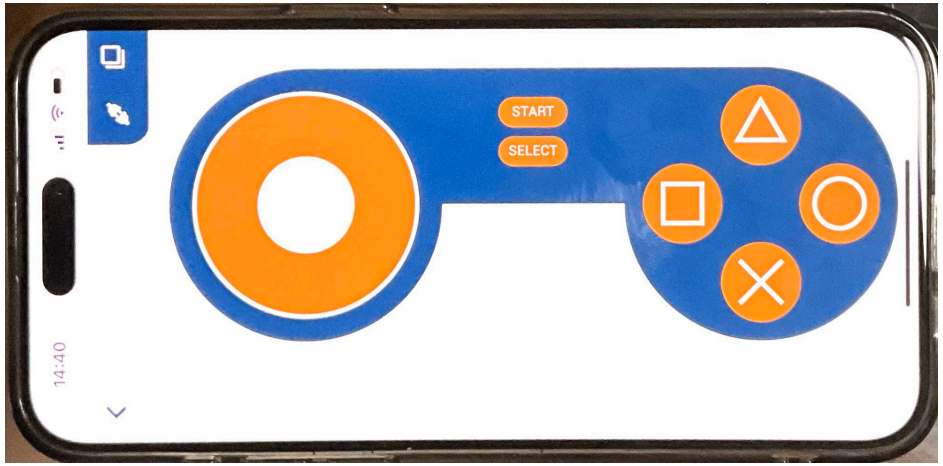
事前準備　STEM Duコントローラ RDC-ESP32R2　開発環境のインストール

参考：
JAPAN ROBOTECH webサイト（プログラミング環境をインストール）
https://sites.google.com/site/japanrobotech2/start

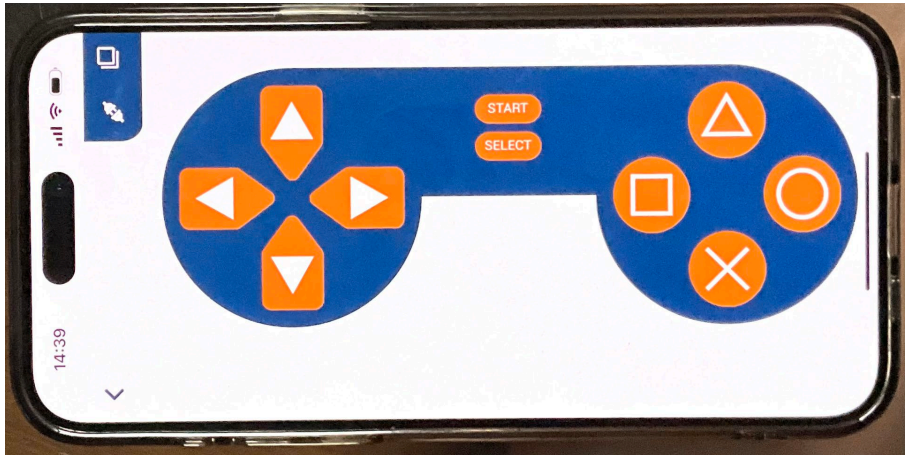
埼玉大学　教育学部　野村研究室（RDC-ESP32のボードマネージャをインストール）
http://neo.stem-edulab.org/page_20200820030156/page_20201016230943

スマホapp「Dabble」

スマートフォンの組み込み機能であるタッチパネルや，GPS・マイク・加速度計などのセンサーの出力値をBluetooth経由でESP32へ送信します．これにより，ハードウェアをコントロールできます．
また，Scratch & Arduinoと互換性のある専用プロジェクトも提供しており，実際にやってみて学ぶのに役立ちます．
今回はスマートフォンを，無線ゲームコントローラとして使います



写真：Joystic Mode

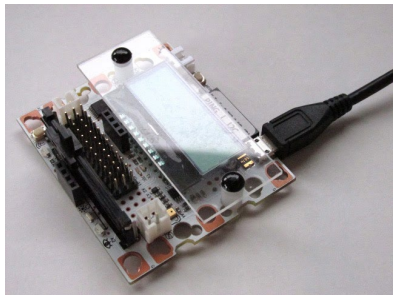


写真：Digital Mode

動作の流れ（イメージ）

目的：RDCとスマートフォンをBluetooth接続 → アプリ：Dabble で操作

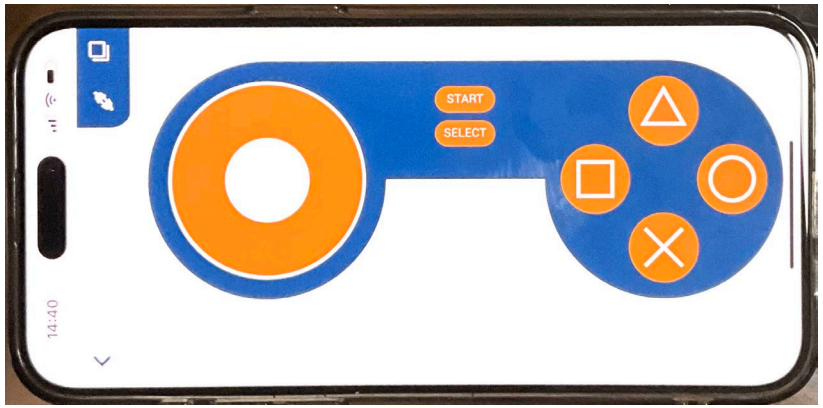
1．RDC開発環境Arduino IDEへ
DabbleESP32ライブラリ追加
→動作確認プログラムを準備



2．スマートフォンへ
app「Dabble」をインストール

3．RDCとスマートフォンを
Bluetooth接続

4．「Dabble」を起動
Gamepadを選択



5．「Dabble」のコントローラを操作すると，
操作に応じた入力値がBluetooth接続を
通じて，RDCへ出力され続けます

作業の流れ　目的：RDC-ESP32とスマートフォンを Bluetooth接続し，ゲームコントローラ

1．DabbleESP32ライブラリをArduino IDEに追加

（Arduino IDEと動作確認プログラムの準備）

2．Dabbleをスマートフォンへインストール

3．RDC-ESP32とスマートフォンをBluetooth接続

（スマートフォンのBluetooth設定から，RDC-ESP32を登録）

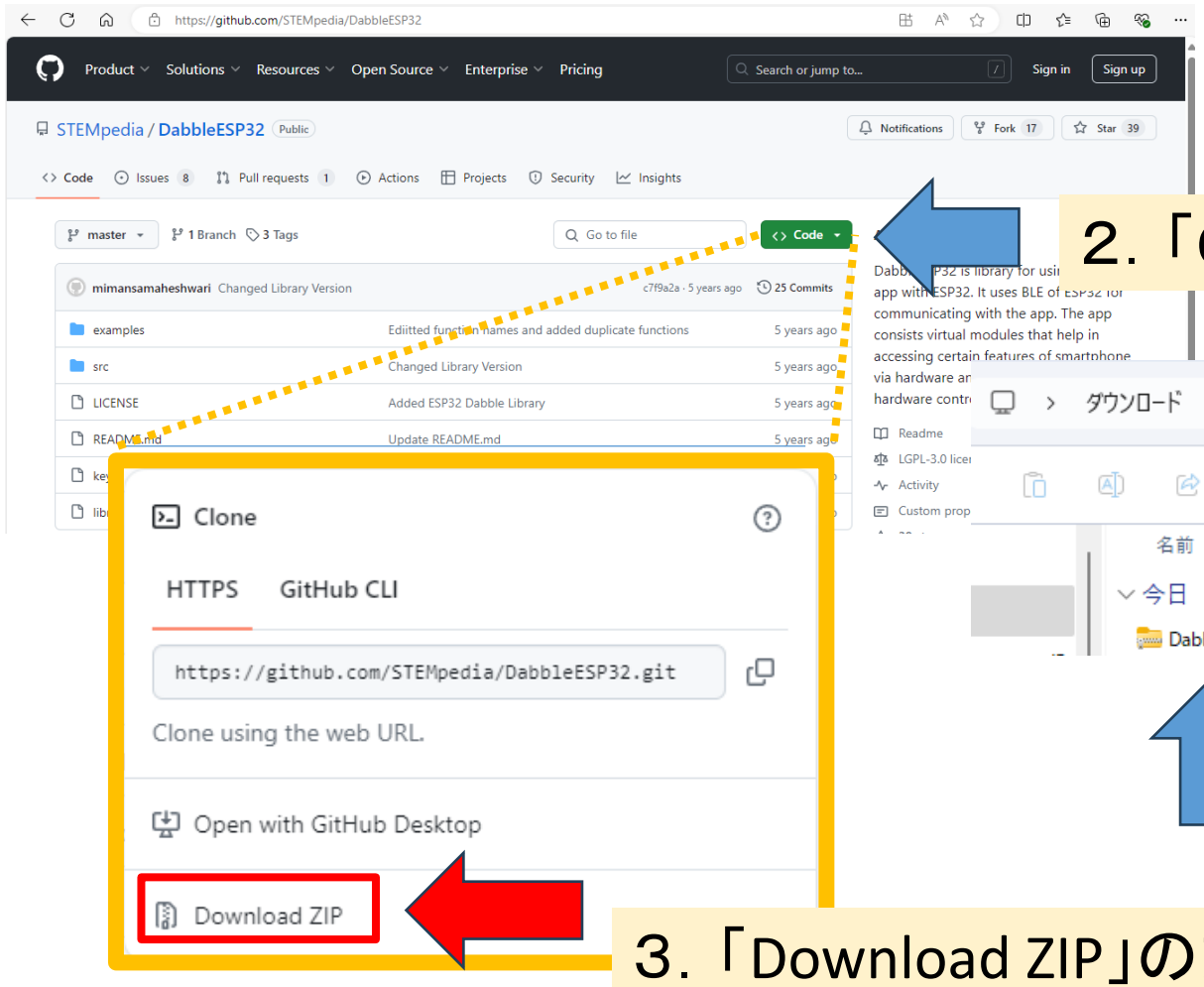
4．サンプルプログラムで動作確認

（コントローラの動作確認）

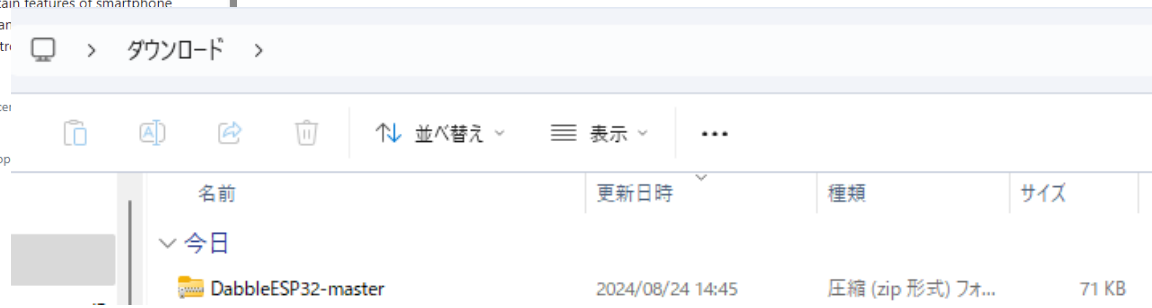
1. Dabbleesp32ライブラリをArduino IDEに追加

1-1. DabbleEsp32ライブラリの追加

1．次のURLを開く
https://github.com/STEMpedia/DabbleESP32



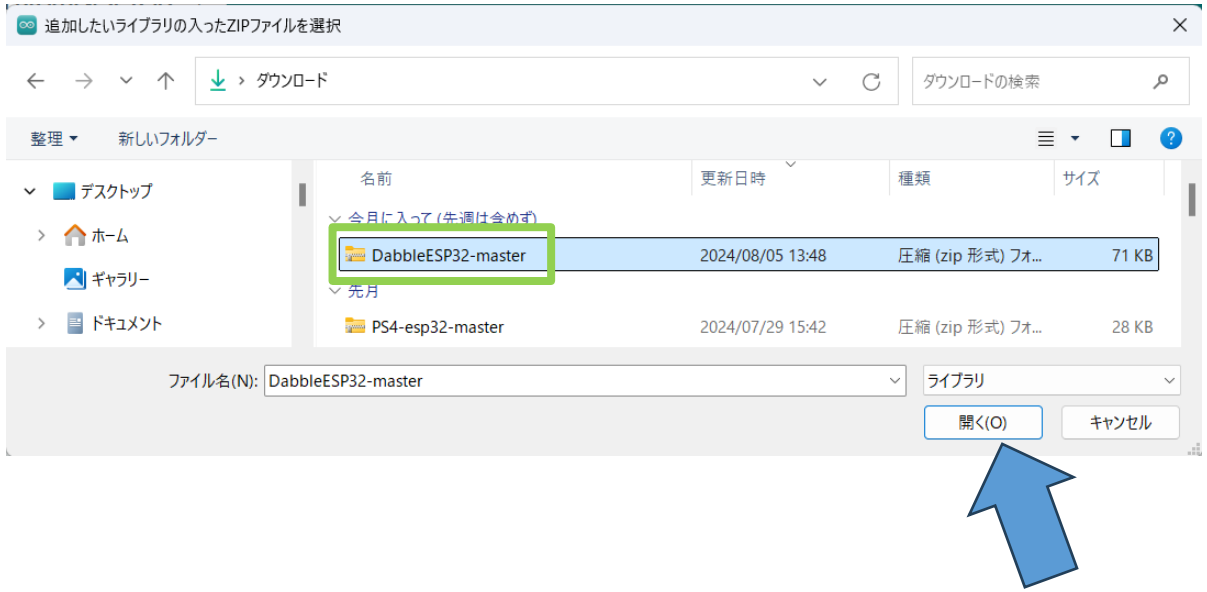
2．「Code▼」をクリック



ダウンロードのフォルダに，
「DabbleESP32-maste.zip」ファイルを確認

3．「Download ZIP」の文字をクリックし，
ファイル一式をダウンロード

1-2 DabbleEsp32ライブラリをArduino IDEにインストール



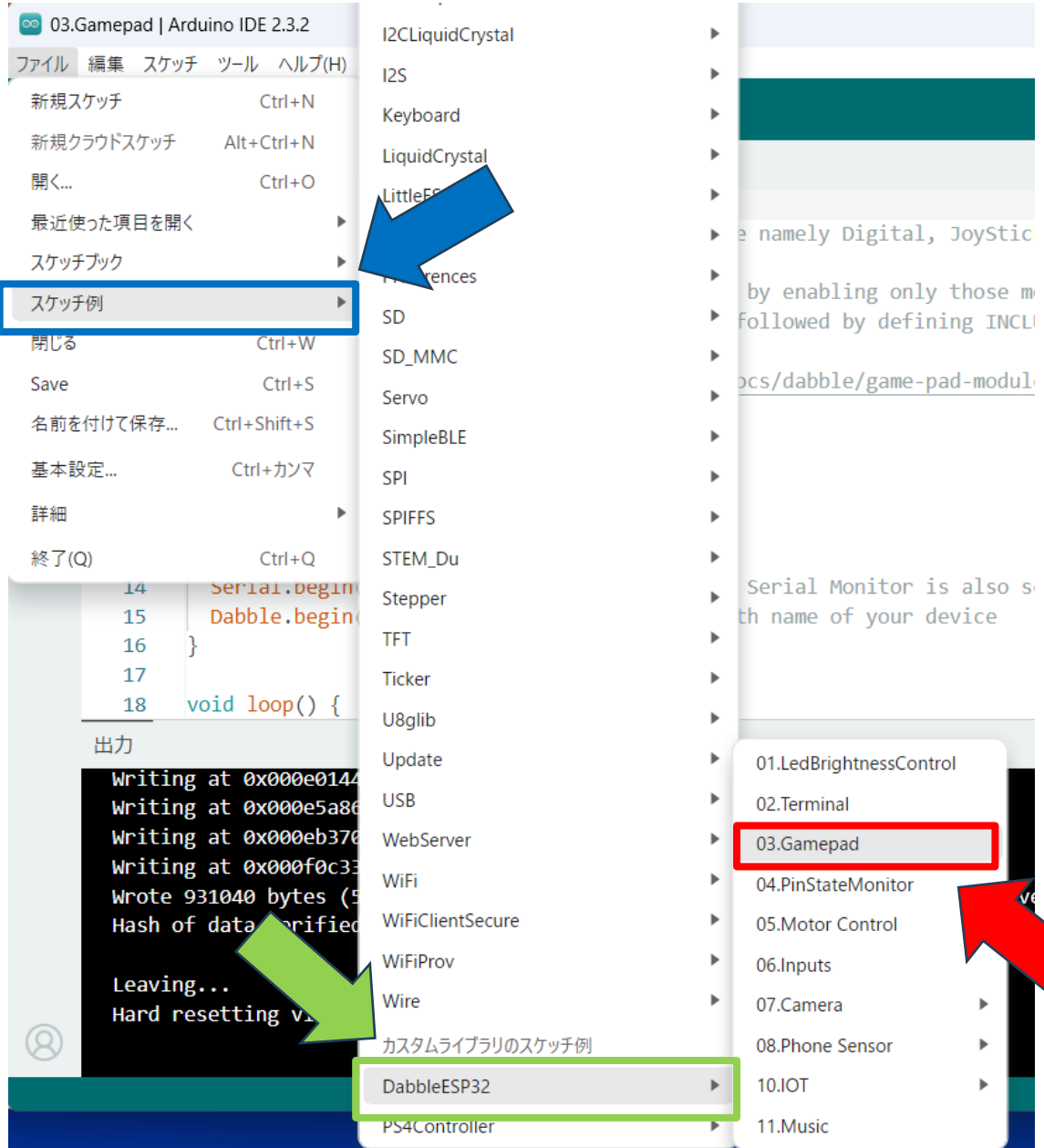
1. Arduino IDEを起動し
   スケッチ →
   ライブラリをインクルード →
   .ZIP形式のライブラリをインストール
   を選択

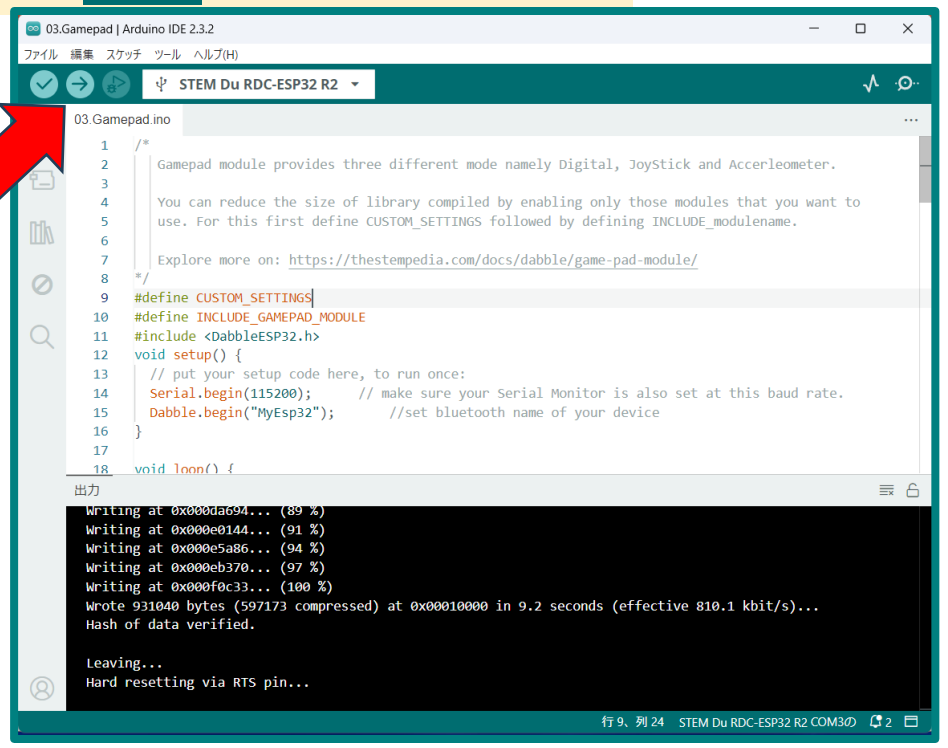
2. ダウンロードしたフォルダを選択し，
   先ほど入手した
   「 DabbleESP32-maste.zip」ファイルを
   を選択して，「開く」をクリック

1-3 サンプルプログラムの準備

1.Arduino IDEの「ファイル」から
→スケッチ例→DabbleESP32
→03.Gamepad
の順に選択てクリック

2.「書き込み」  をクリック

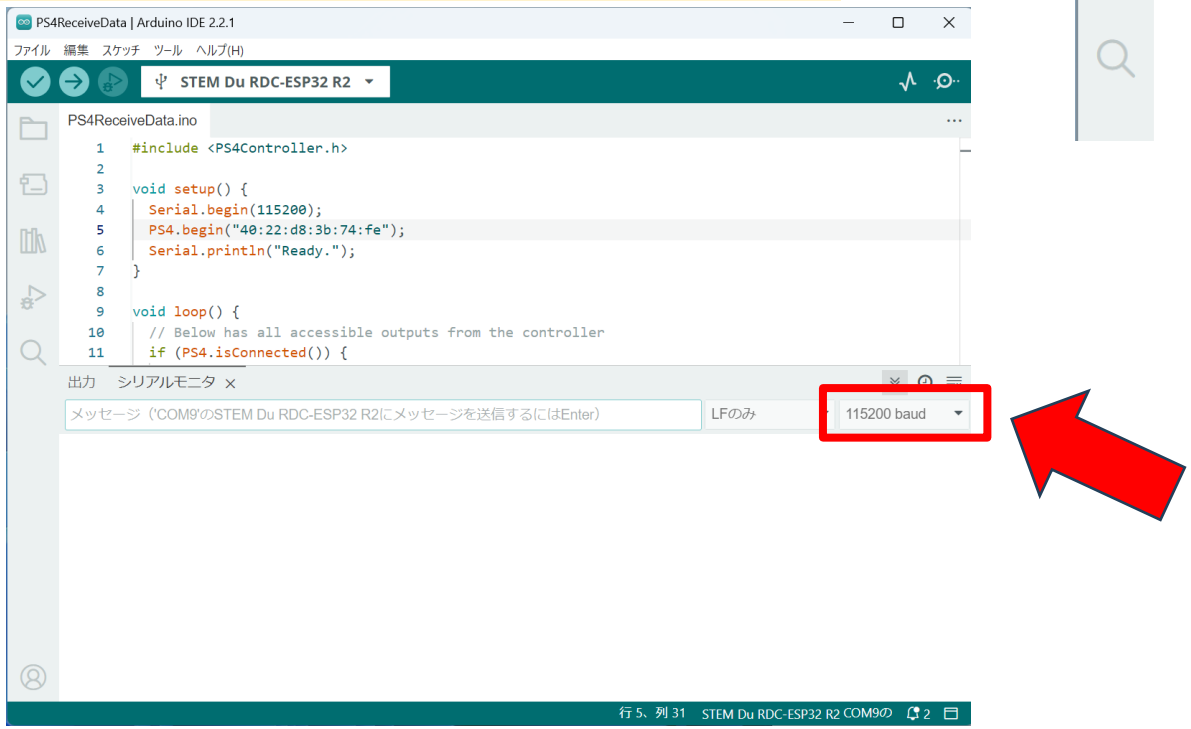




1–4. サンプルプログラムで動作確認

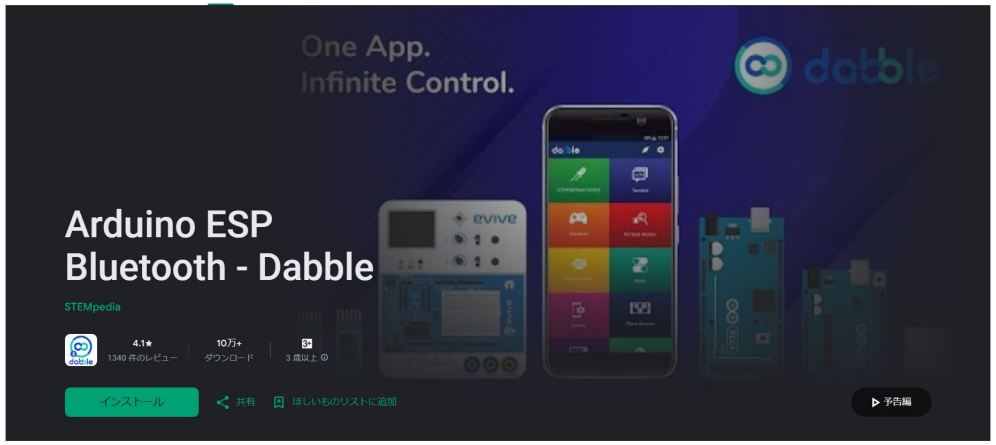
1.「ツール」→ シリアルモニタ
　を起動する

2.シリアルモニタの通信速度から
　「115200baud」を選択

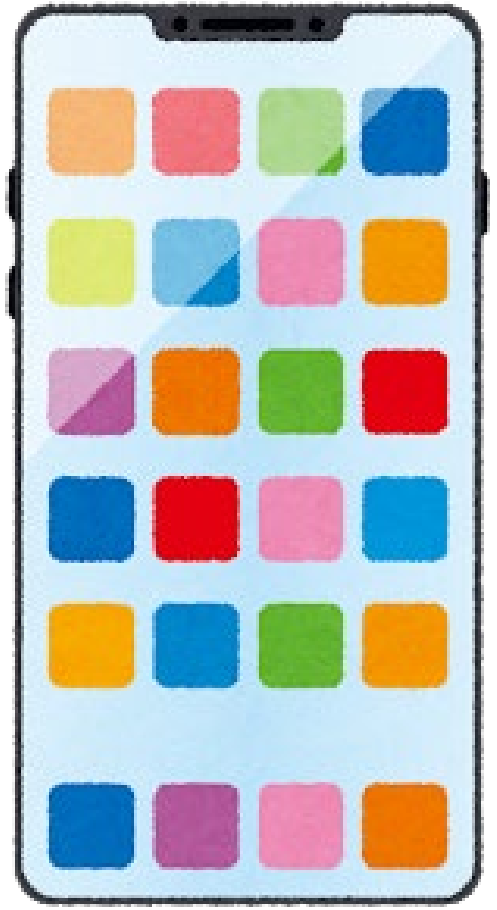


PCとRDCはこのままにして，
次の手順へ

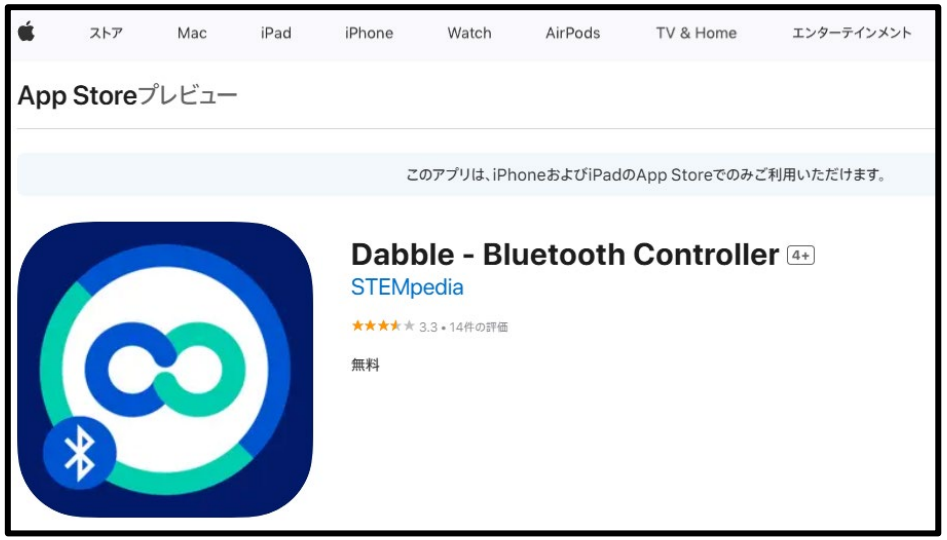
2. スマホapp「Dabble」をインストール



各URLを参考に，
スマートフォンへ「Dabble」をインストールする



https://play.google.com/store/apps/details?id=io.dabbleapp&hl=ja

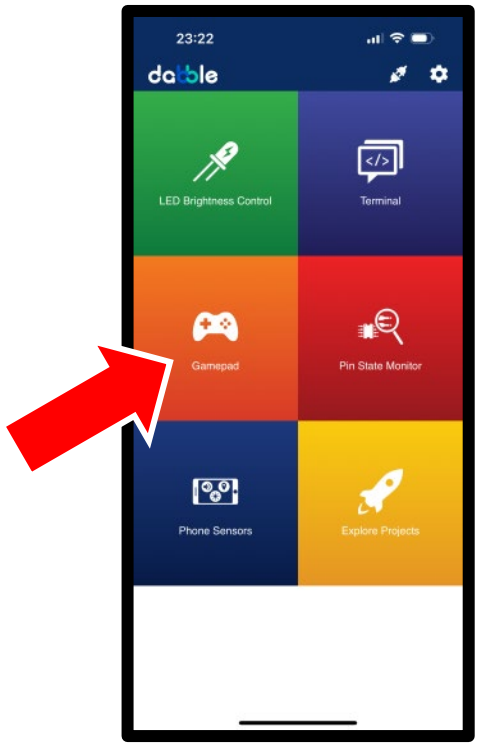


https://apps.apple.com/jp/app/dabble-bluetooth-controller/id1472734455

3. RDC-ESP32とスマートフォンをBluetooth接続

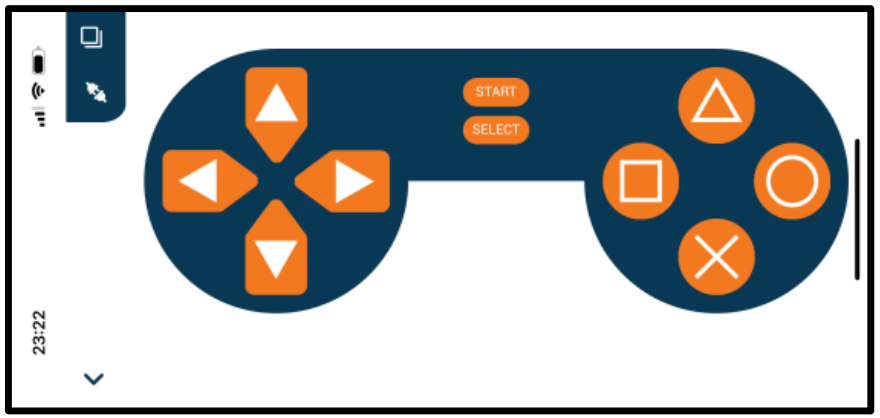
app「Dabble」を起動して，RDCとBluetooth接続を行う

1. スマートフォンへインストールした
「Dabble」を起動



2.「Gamepad」をタップ

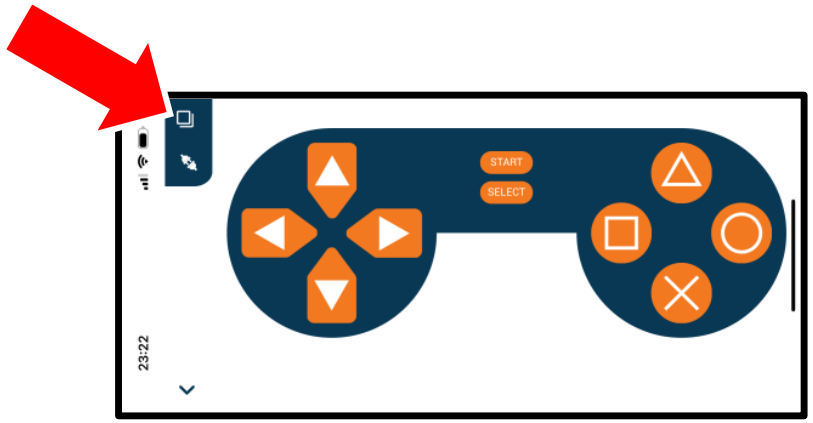
3. コントローラ画面への
切替わりを確認

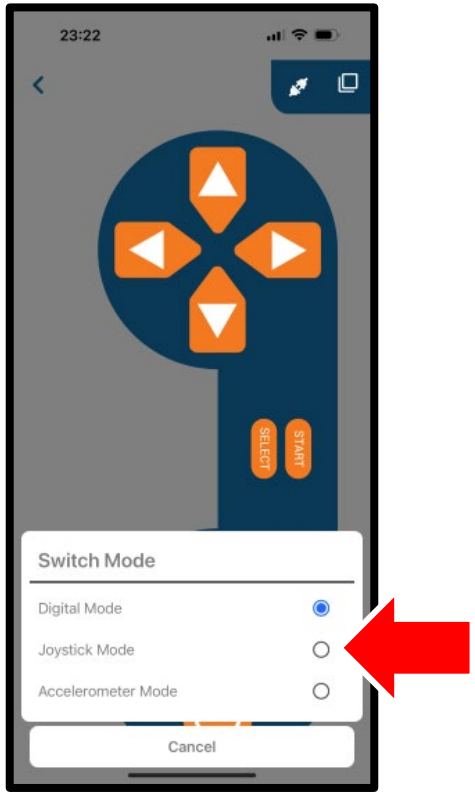


3. RDC-ESP32とスマートフォンをBluetooth接続

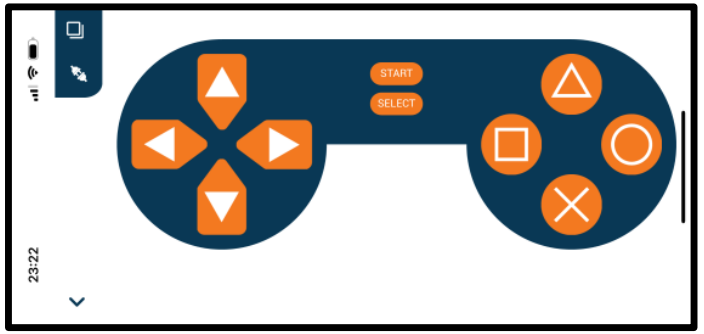
4. コントローラ「Swirch Mode」の切り替え

4.1　画面右上の  をタップ

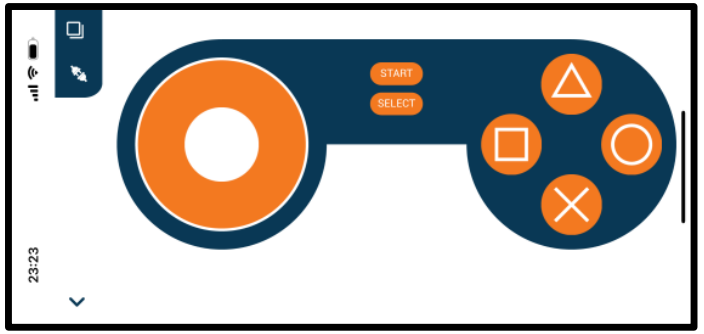




4.2　Switch Mode で「Joystick Mode」へ変更する



Digital Mode

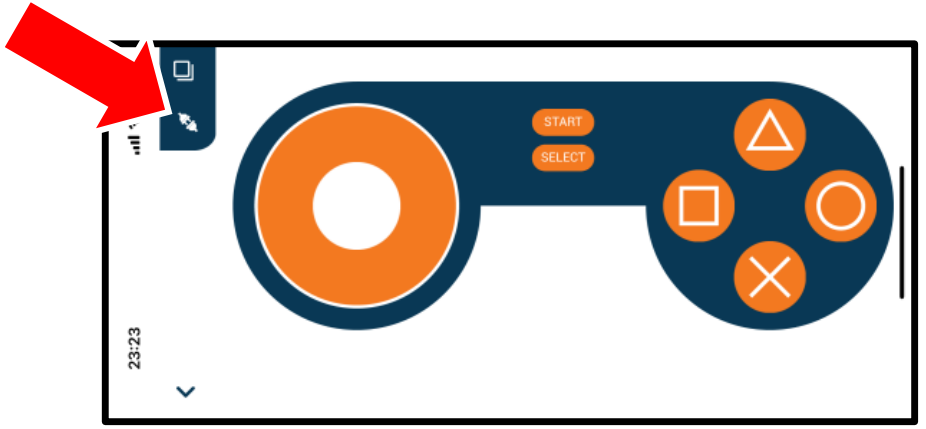


- Joystick Mode
  指で右の白丸を操作
- Accelerometer　Mode
  スマホを傾け白丸を操作

3．RDC-ESP32とスマートフォンをBluetooth接続

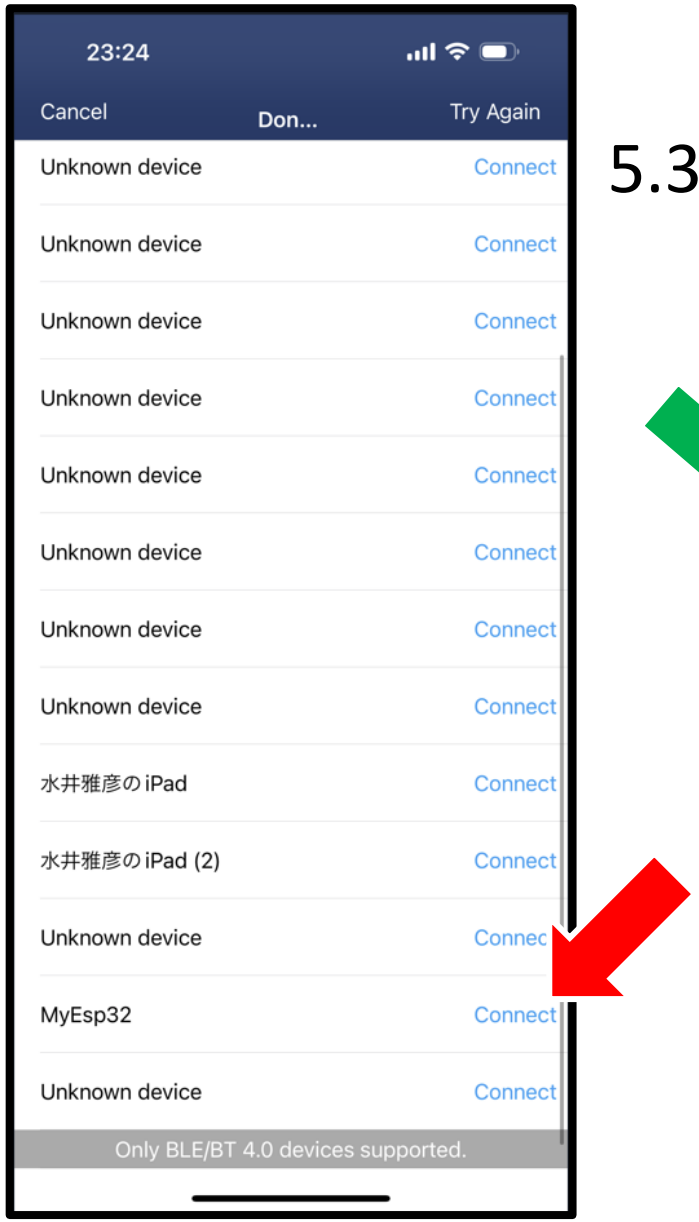
5．RDCと接続

5.1　画面右上の  をタップ

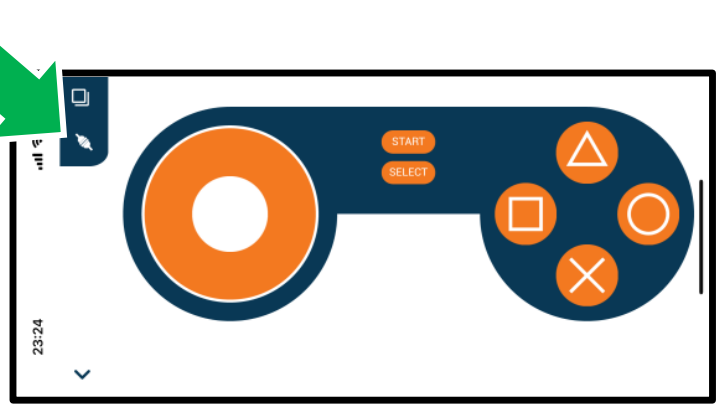


5.2.「MyEsp32」を探して「Connect」をタップ

※RDCの名称「MyEsp32」について後に解説



5.3　画面右上のアイコンが  になったことを確認



※5.3が成功すれば
Bluetooth接続は完了！

3．RDC−ESP32とスマートフォンをBluetooth接続

※RDCの名称「MyEsp32」を変更

教室などで何人かとBluetooth接続を行うとき，
自分のRDCと区別するために名称「MyEsp32」を変更しよう

1-3　サンプルプログラムの準備で，スケッチ「03.Gamepad.ino」を選択した

03.Gamepad.ino

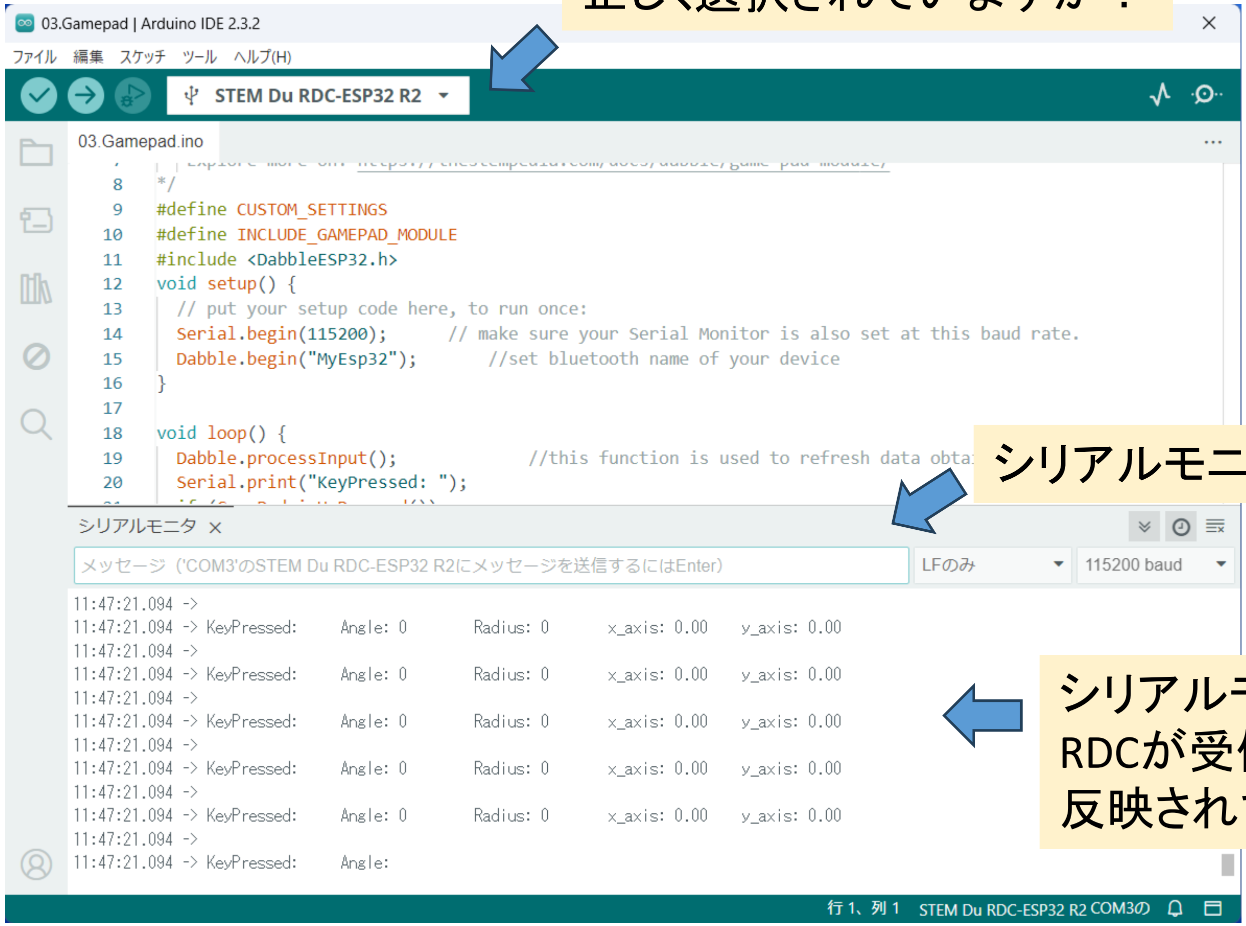
```
1   /*
2     Gamepad module provides three different mode namely Digital, JoyStick and Accerleometer.
3
4     You can reduce the size of library compiled by enabling only those modules that you want to
5     use. For this first define CUSTOM_SETTINGS followed by defining INCLUDE_modulename.
6
7     Explore more on: https://thestempedia.com/docs/dabble/game-pad-module/
8   */
9   #define CUSTOM_SETTINGS
10  #define INCLUDE_GAMEPAD_MODULE
11  #include <DabbleESP32.h>
12  void setup() {
13    // put your setup code here, to run once:
14    Serial.begin(115200);      // make sure your Serial Monitor is also set at this baud rate.
15    Dabble.begin("MyEsp32");       //set bluetooth name of your device
16  }
```

訳：Bluetooth接続での
あなたのデバイス名

15行目:Dabble.begin(“MyEsp32”);
この名称がBluetooth接続の相手を探すときに使われる
“MyEsp32”を　“MyEsp***” などに（***を出席番号など）
変更するとよい

4. サンプルプログラムで動作確認

正しく選択されていますか？



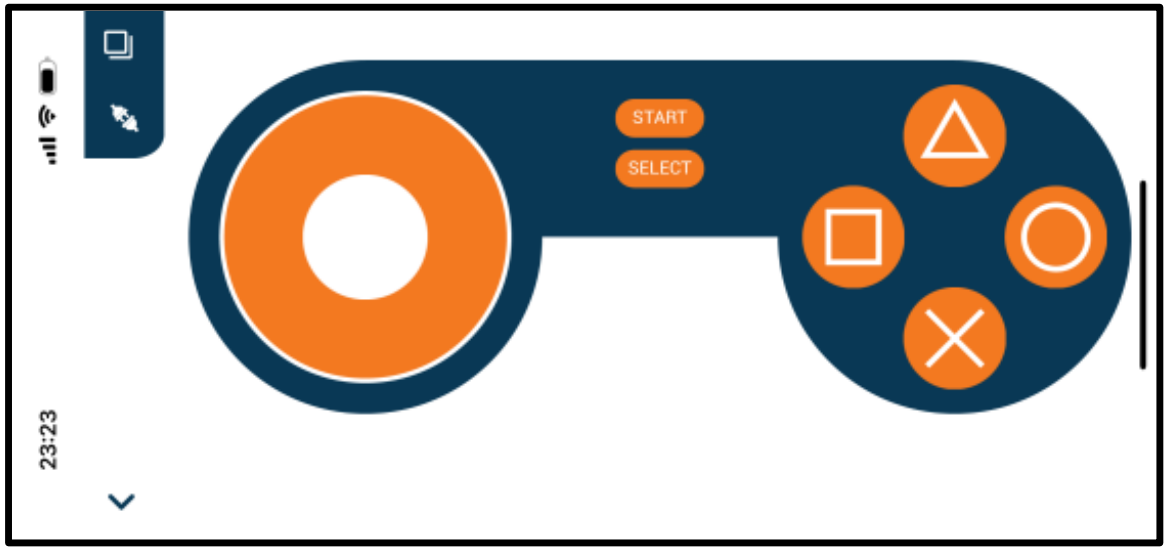
シリアルモニタを起動（「ツール」内）

シリアルモニタに，
RDCが受信したコントローラの入力値が
反映されていますか？

RDCをリセットしたら，再度app「Dabble」から接続する必要があります

4. サンプルプログラムで動作確認

4.1 ボタン入力の確認



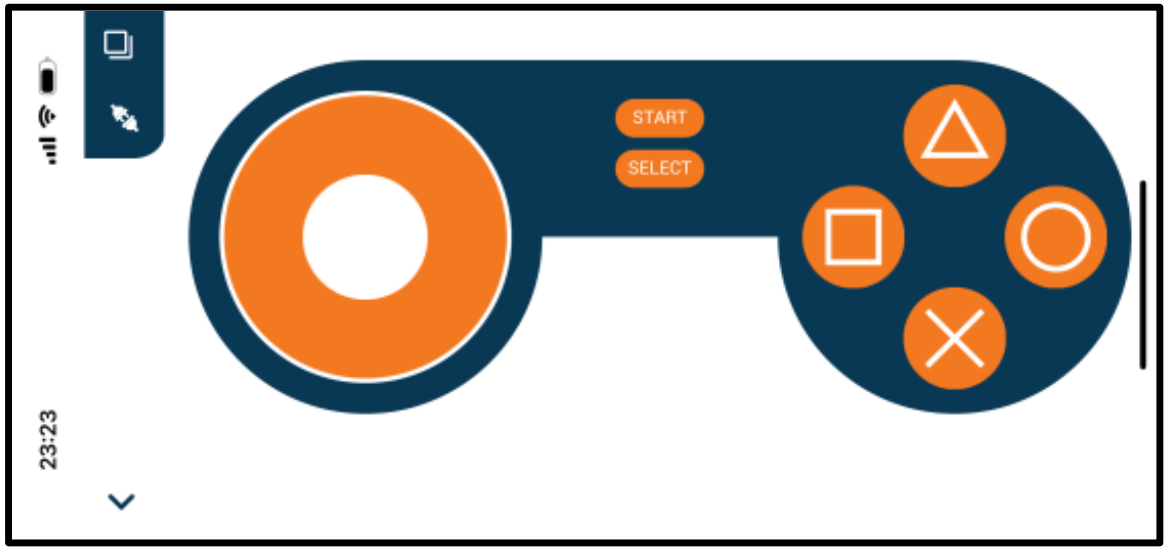
「□，△，○，×，START，SELECT」のボタンを押すと「KeyPressed」に続き入力値が表示されます

「△」のボタンを押すと
「KeyPressed: Triangle」と表示されているのが確認できます

```
12:04:50.879 -> KeyPressed: Triangle    Angle: 0        Radius: 0       x_axis: 0.00    y_axis: 0.00
12:04:50.912 ->
12:04:50.912 -> KeyPressed: Triangle    Angle: 0        Radius: 0       x_axis: 0.00    y_axis: 0.00
12:04:50.912 ->
```

4. サンプルプログラムで動作確認

4.2 アナログ入力の確認



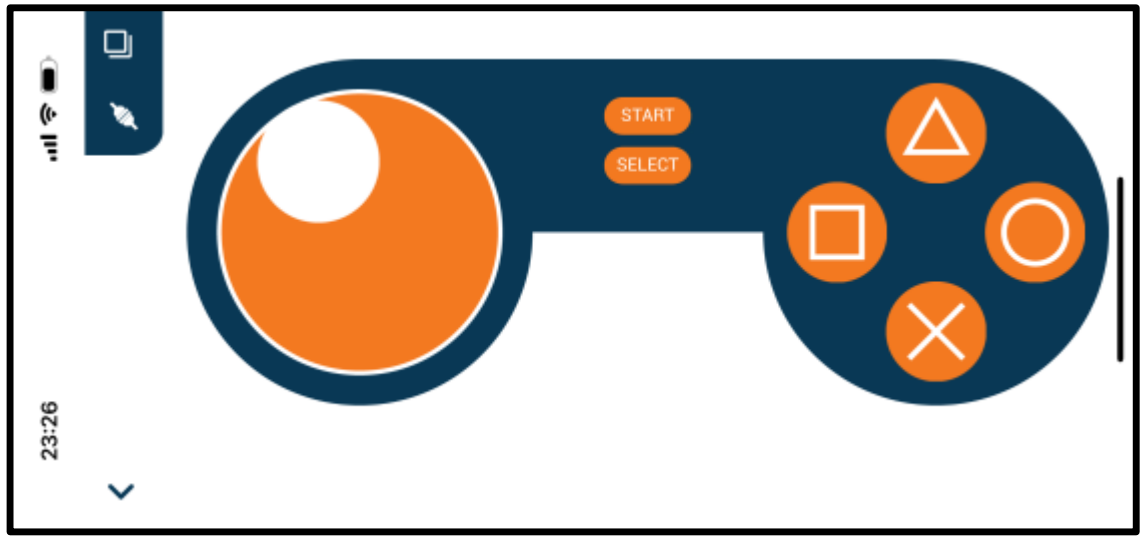
左の「白丸」を動かし，アナログ入力を行えます．
指を触れなければ，「白丸」は中心に戻ります

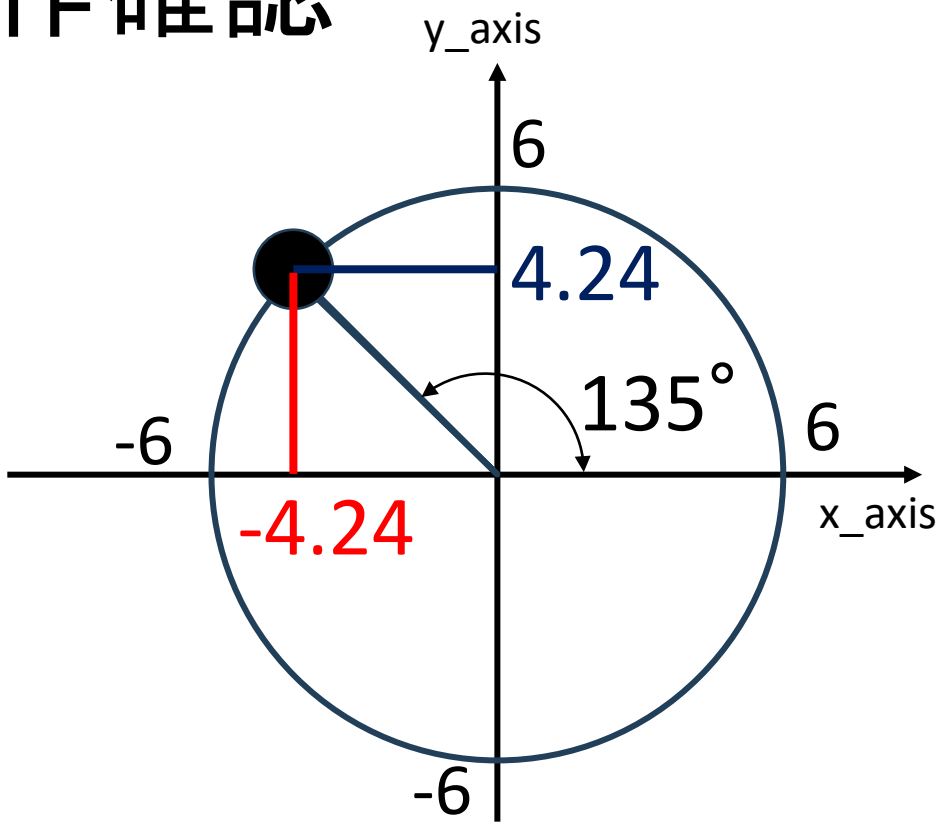
「白丸」が中心にある時の値を確認しましょう．
「Angle」は角度，「Radius」は円中心から「白丸」までの距離を表示

```
12:17:17.342 -> KeyPressed:     Angle: 0        Radius: 0       x_axis: 0.00    y_axis: 0.00
12:17:17.342 ->
12:17:17.342 -> KeyPressed:     Angle: 0        Radius: 0       x_axis: 0.00    y_axis: 0.00
12:17:17.342 ->
```

4. サンプルプログラムで動作確認

4.3 アナログ入力値の表示





Angleと Radius，x_axis と y_axis の2種類でアナログ入力値を表示

```
12:23:29.049 -> KeyPressed:     Angle: 135      Radius: 6       x_axis: -4.24   y_axis: 4.24
12:23:29.049 ->
12:23:29.049 -> KeyPressed:     Angle: 135      Radius: 6       x_axis: -4.24   y_axis: 4.24
12:23:29.049 ->
```

おわりに

最新のマニュアルやRDCを用いたサンプルプログラムを，
GitHubにて公開中
mmizui/RDC-ESP32-Tips (github.com)