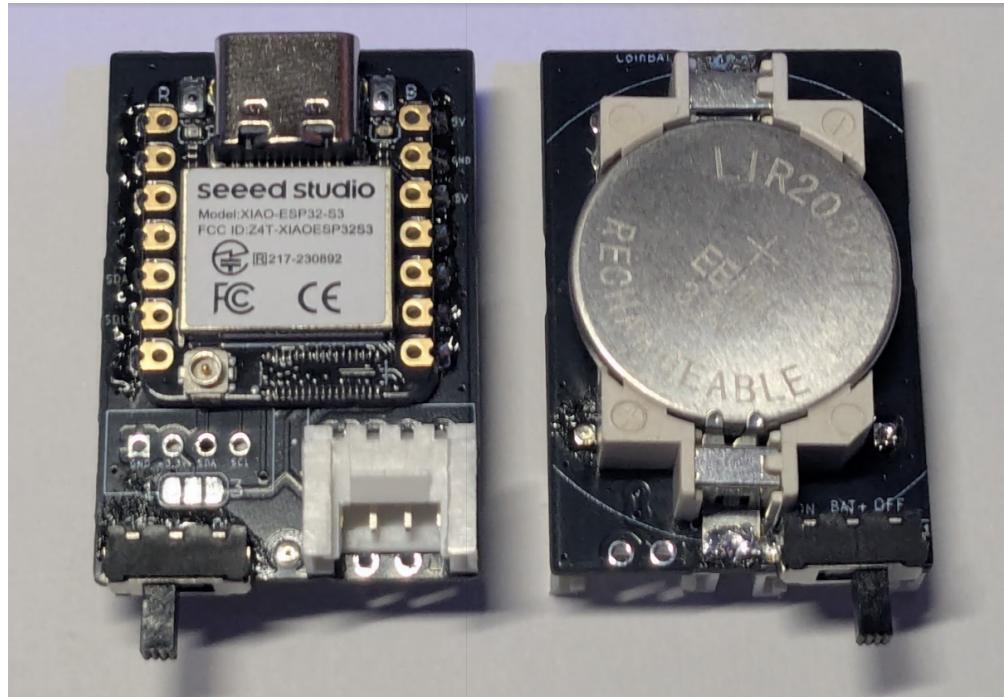


XIAO-Coin で始める IoT



やわらからじお

目次

1. はじめに
2. XIAO-Coin について
3. 部品とコスト
4. 組み立て手順
5. 開発環境の構築
6. センサー接続とプログラミング
7. 応用例とプロジェクト
8. トラブルシューティング
9. 接続図と PCB
10. 参考資料

はじめに

本書は、XIAO-Coin を使用した小型充電式無線マイコン、並びに MicroPython による簡易なプログラミング解説するガイドブックです。

対象読者

- 電子工作初心者ー中級者
- 小型デバイスでのプロトタイピングを行いたい方

本書で学べること

- XIAO-Coin の組み立て
- MicroPython によるプログラミング -各種センサーの活用方法

注意点

- **CR2032** 電池は使用できません
- 一般的なリチウム電池は使用できません
- **LIR2032/2450** 専用です（リチウムイオンコイン電池のみ）
- 半田付けに慣れている必要があります

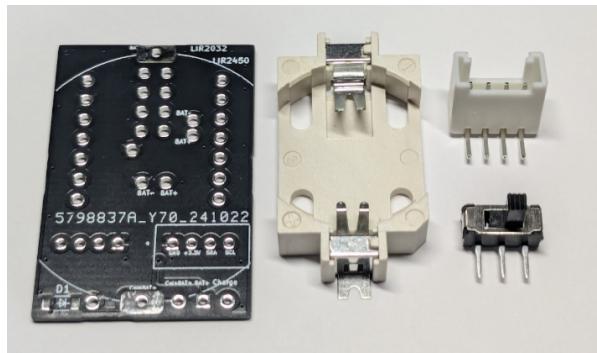
XIAO-Coinについて

概要

XIAO-Coinは、リチウムイオン充電機能のある Seeed Studio 製 XIAO とリチウムイオンコイン電池を組み合わせた、小型(38mm x 28mm)の無線マイコンボードです。

オンラインで基板セットを販売しています。GitHub で KiCAD ファイルを公開していますので、気に入った方は沢山創ったり、カスタマイズして是非ご利用ください。

スイッチサイエンス販売サイト: <https://www.switch-science.com/products/10032>



XIAO-Coin_ セット品

特徴

- XIAO の小ささと Lipo 充電端子を生かしたコインホルダー一体型無線マイコン
- Grove 端子で I2C センサ接続可能
- 充電機能内蔵のため各種コイン電池に対応：45mAh～120mAh (2032～2450 サイズ)

必要部品とコスト

必要部品一覧 (BOM)

- XIAO の単価は C3 が最も安く、630 円/個で試作できるため各種センサの動作評価に向いています。
- これだけ安ければ動作したセンサとセットでそのままにしておけるので、毎回ソフトやセンサを付け替える必要が無いのが便利です。

名称	価格	販売先	備考
XIAO ESP32 C3/C6/S3	630 円~810 円	Seeed	10 個購入時の値段
ボタン電池基板取付用ホルダー CH29-2032LF	50 円	秋月電子	
スライドスイッチ MK-12D13G4-B	10 円	LCSC	
Grove コネクター L 型 スルーホール	15 円	秋月電子	

XIAO 各種モデルの単価 (Seeed 社から 10 個購入時)

モデル	単価 (USD)	単価 (円) *	特徴
XIAO ESP32 C3	\$4.20	630 円	最安値、WiFi/Bluetooth 対応
XIAO ESP32 C6	\$4.90	735 円	WiFi 6 対応、Thread/Zigbee 対応
XIAO ESP32 S3	\$5.40	810 円	カメラ対応、AI 機能強化

* 為替レート: 1USD = 150 円で計算

推奨電池

EEMB LIR2032H 充電式バッテリー - 電圧: 3.7V - 容量: 70mAh - 価格: ¥1,309



LIR2032H 電池

組み立て手順

部品のハンダ付け手順

1. 基板に XIAO を半田付け
2. 裏面の充電端子をホール半田付け
3. 電源スイッチを半田付け
4. Grove 端子を半田付け
5. バッテリーホルダーを半田付け

注: 先にバッテリーホルダーを付けると、XIAO の充電端子を半田付けできないので注意。次ページで写真付きで説明します。

XIAO と基板の半田付け

- XIAO の片側の端子をピン固定し、反対側の端子を半田付けする。



XIAO と基板の半田付け

XIAO のはんだ付け

- 反対側のピンを外し、はんだ付けする。



XIAO と基板の半田付け __ 反対側

充電端子のホール半田付け

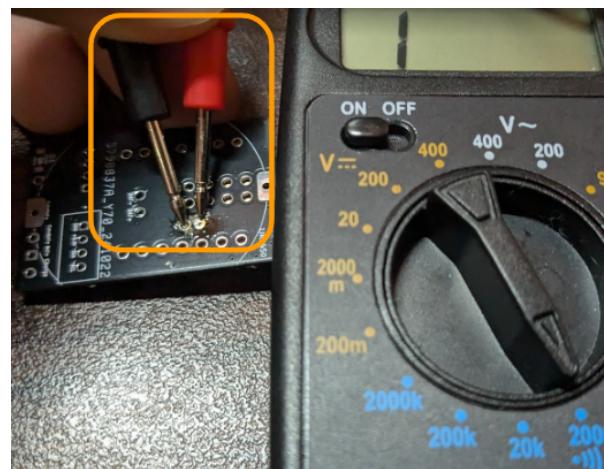
- 基板裏面の充電端子 (BAT+、BAT-) をスルーホール越しに半田付けする。(ホール半田付け)
- 2025年5月から基板をカットして半田付けしやすくしています。



充電端子のホール半田付け

充電端子のショート確認

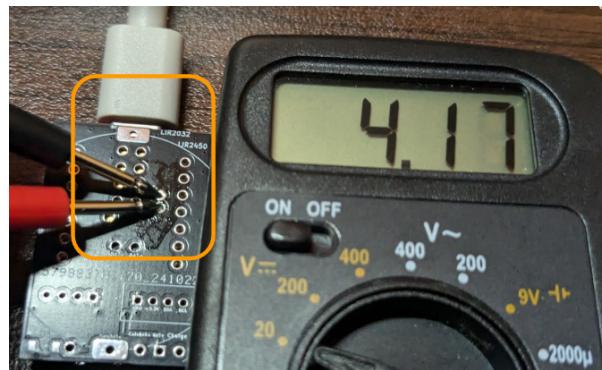
- BAT+ と BAT-がショートしていない事を念のため確認する。



充電端子のショート確認

電圧確認

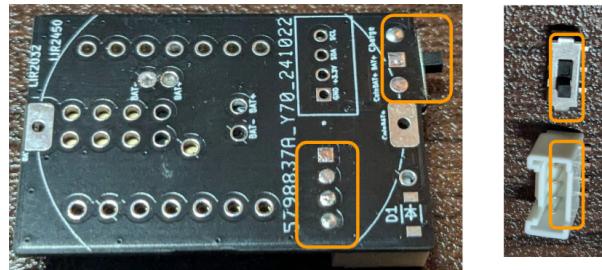
- USB から電源供給し、充電端子間 (BAT+ と BAT-) で 4.1~4.2V 程度の電圧があることを確認する。



充電端子 __ 電圧確認

電源スイッチと Grove コネクタの半田付け

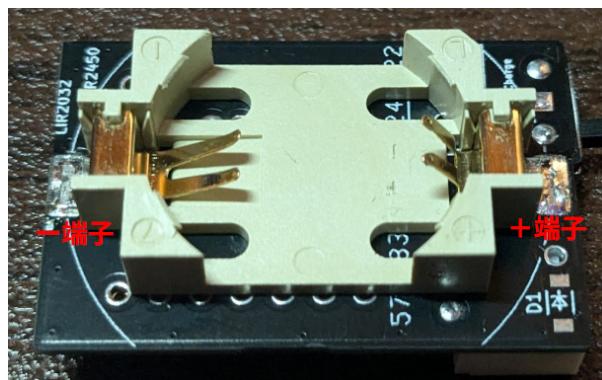
- 電源端子と Grove 端子のピンを切り 1mm 程度とする。
- 電源端子 →Grove 端子の順に半田付けする（低背部品の順に半田付けする）。
- ピンが長いとバッテリーホルダーが浮いてしまうため、適切な長さで且つ半田が隆起しないよう少量とする。



電源スイッチと Grove コネクタの半田付け

バッテリーホルダーの半田付け

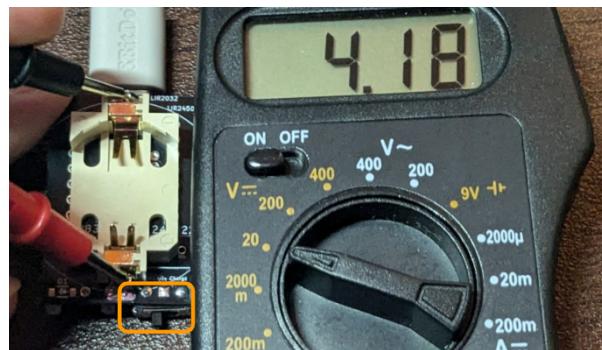
- バッテリーホルダーを半田付けする。バッテリー端子の+/-をしっかり確認し、+/-端子を間違えないに半田付けする。



バッテリーホルダーの半田付け

電圧確認

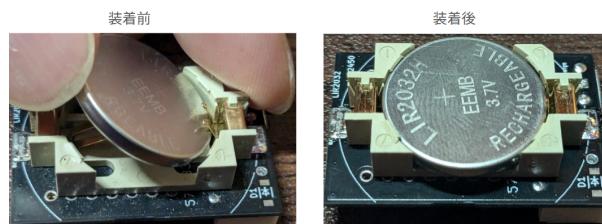
- 電源スイッチを中央側に切り替える事で XIAO とバッテリーが通電する。
- バッテリー端子の+端子と-端子間で電圧が 4.1-4.2V 程度ある事を確認する。



バッテリーホルダ_電圧確認

電池の装着

- コイン電池の+/-がショートしないよう気を付けて装着。

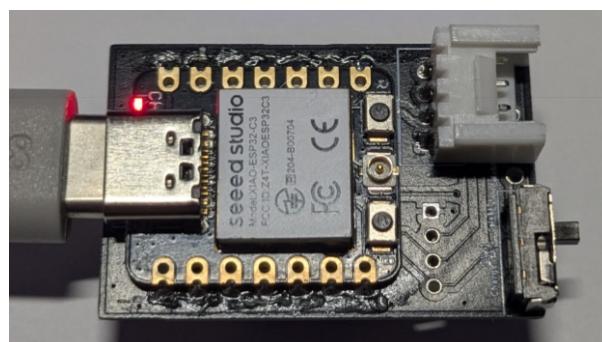


電池の装着

完成品

完成！

USB 接続し、スライドスイッチを中央側 にすると充電され、充電中は CHG LED が点滅します
好きなセンサを接続しましょう！



完成品

開発環境の構築

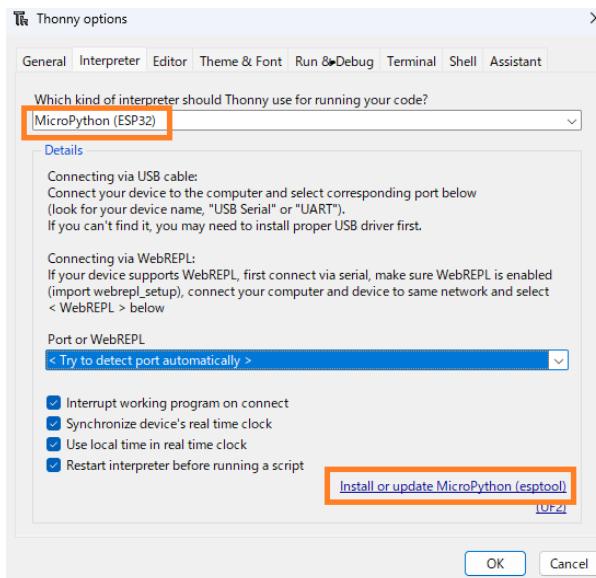
Thonny のインストールと設定

1. Thonny のインストール

1. Thonny 公式サイトからダウンロード
2. インストーラーを実行してセットアップ

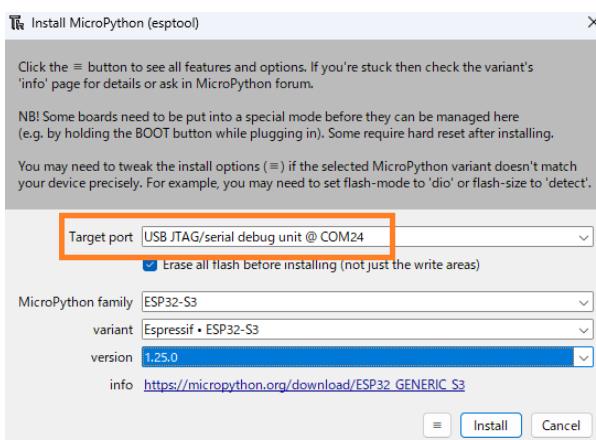
2. MicroPython ファームウェアの書き込み 下記 6 ステップを順に行ってください。

- Thonny メニュー「Tools」→「Options」
- 「MicroPython(ESP32)」を選択
- 「Install or update MicroPython(esptool)」を選択



Thonny 設定画面

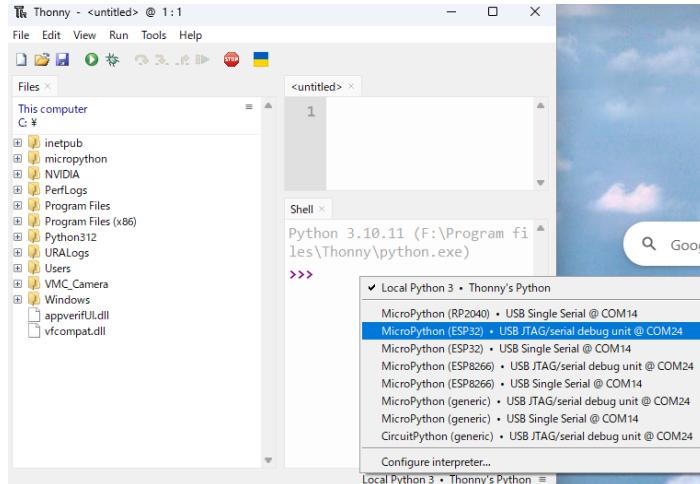
- XIAO ESP32 の Boot ボタンを押しながら USB 接続（ブートモードにする必要があります。）
- 適切な COM ポートを選択（USB/JTAG についているものを選びます）
- ESP32 用ファームウェアを選択して Install



ファームウェア書き込み

3. ESP32への接続

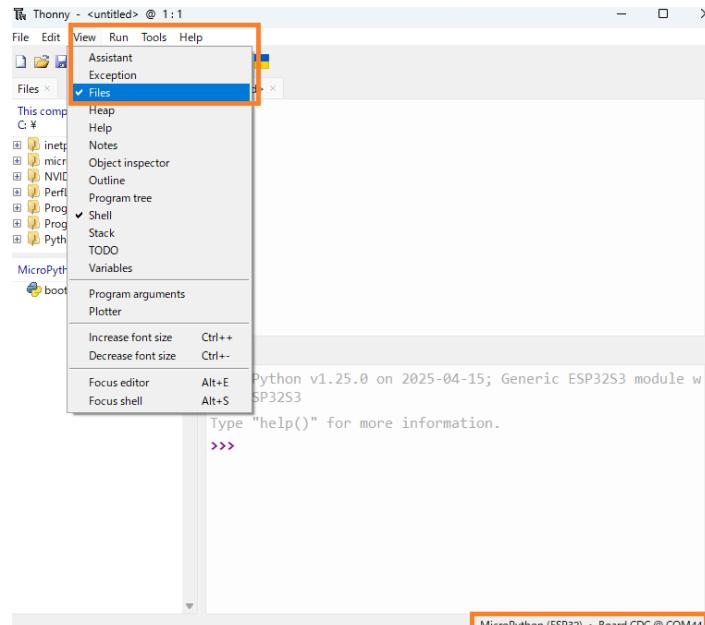
- XIAO ESP32 のリセットボタンを押すか USB 再接続した後、Thonny 右下の接続先を適切な COM ポートに変更します
- 「MicroPython(ESP32)」と表示されれば接続成功です



COM ポート選択

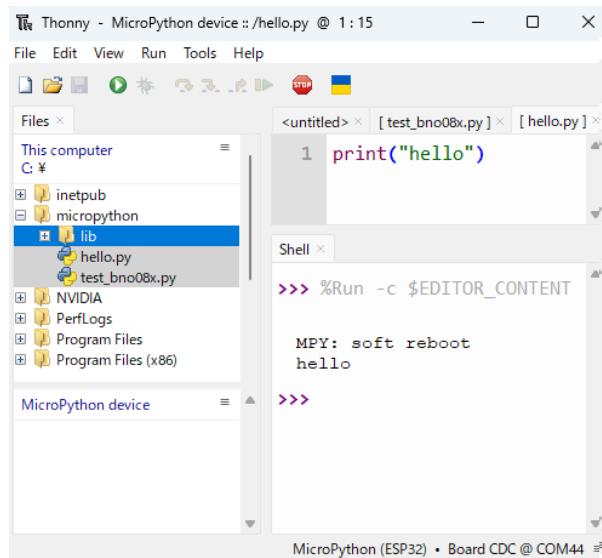
4. ファイルのアップロード

- 「View」→「Files」でファイルツリーを表示



ファイルビュー

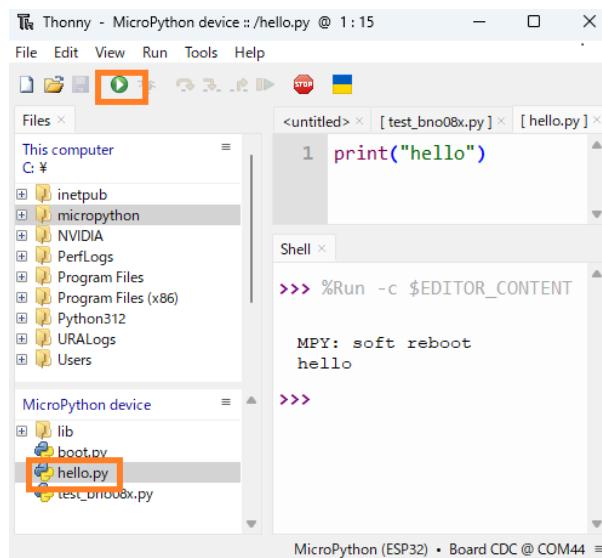
- micropython プロジェクトファイルを ESP32 にアップロード
 - Github のリポジトリから micropython フォルダをダウンロードし、C ドライブ直下に格納してからアップロードしてください <https://github.com/uecken/xiao-coin>



ファイルアップロード

5. テストプログラムの実行

- 再生ボタンを押し、Hello と表示されれば動作しています。



Hello.py の実行

センサー接続とプログラミング

BNO085 IMU センサーの使用

センサーの調達

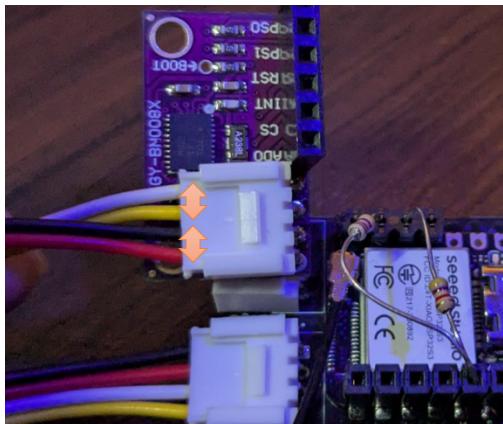
- BNO085 は AliExpress から購入すると安いです。
- 但し Grove コネクタを使う場合、3.3V-GND、SDA-SCL ピンをそれぞれ入れ替える必要があるので注意してください。
- URL: <https://ja.aliexpress.com/item/1005005902501032.html>



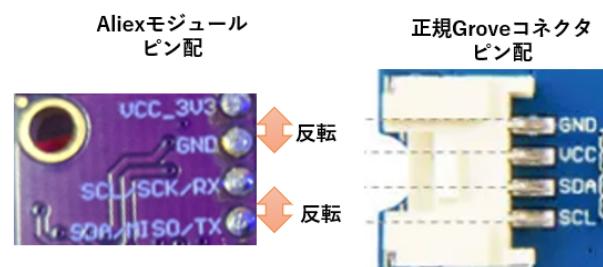
BNO085

配線方法

- 参考までに、Aliexpress のモジュールのピン配置、正規の Grove コネクタのピン配置を参考として示します。正規の Grove コネクタのピン配置に合わせるために、3.3V-GND、SDA-SCL ピンをそれぞれ入れ替える必要があります
- 一般的に I2C は SDA と SCL にプルアップ抵抗が必要です。通信が安定しない場合、3.3V と SDA、3.3V と SCL を繋ぐように $4.7\text{k}\Omega$ の抵抗を接続して下さい。



XIAO-BNO085_ 配線



Grove ピン配_Aliex モジュールピン配

サンプルプログラム

- サンプルプログラムは下記に格納しています。
- https://github.com/uecken/xiao-coin/blob/main/micropython/test_bno08x_simple.py

```
# BN008x MicroPython I2C Test program - Simplified version
# Basic sensor data display only

from machine import I2C, Pin
from utime import sleep_ms
from bno08x import *

# I2C ピン設定 (ESP32 モデル別)
I2C1_SDA = Pin(5) # XIAO-ESP32 S3:5, C6:22, C3:6
I2C1_SCL = Pin(6) # XIAO-ESP32 S3:6, C6:23, C3:7

# I2C 初期化
i2c1 = I2C(0, scl=I2C1_SCL, sda=I2C1_SDA, freq=400000, timeout=200000)

# BN0085 センサー初期化
bno = BN008X(i2c1, debug=False)
print("BN008x I2C connection : Done\n")

# センサー機能を有効化
bno.enable_feature(BNO_REPORT_ACCELEROMETER, 50)          # 加速度センサー 50ms 間隔
bno.enable_feature(BNO_REPORT_GYROSCOPE, 50)                # ジャイロスコープ 50ms 間隔
bno.enable_feature(BNO_REPORT_GAME_ROTATION_VECTOR, 50)    # ゲーム回転ベクトル 50ms 間隔
bno.enable_feature(BNO_REPORT_GRAVITY, 50)                  # 重力ベクトル 50ms 間隔

# オイラー角とクオータニオンの設定
bno.set_quaternion_euler_vector(BNO_REPORT_GAME_ROTATION_VECTOR)

print("BN008x sensors enabling : Done\n")
print("Starting sensor data display in 3 seconds...")
sleep_ms(3000)

# 初期キャリブレーション
bno.calibration()
print("Initial sensor calibration complete")
sleep_ms(1000)

# 姿勢の零点調整
bno.tare()
print("Tare operation complete - current orientation set as reference")
sleep_ms(1000)

print("\n===== Starting sensor data display =====\n")

# メインループ
count = 0
while True:
    count += 1

    print(f"===== Sensor Data Update #[count] =====")

    # 加速度データ（重力込み）
    accel_x, accel_y, accel_z = bno.acc
```

```

print(f"Acceleration\tX: {accel_x:+.3f}\tY: {accel_y:+.3f}\tZ: {accel_z:+.3f}\t{m/s²}")

# 重力ベクトル
grav_x, grav_y, grav_z = bno.gravity
print(f"Gravity\tX: {grav_x:+.3f}\tY: {grav_y:+.3f}\tZ: {grav_z:+.3f}\t{m/s²}")

# 線形加速度（重力補正済み）
linear_x = accel_x - grav_x
linear_y = accel_y - grav_y
linear_z = accel_z - grav_z
print(f"Linear Accel\tX: {linear_x:+.3f}\tY: {linear_y:+.3f}\tZ: {linear_z:+.3f}\t{m/s²}")

# ジャイロスコープ（角速度）
gyro_x, gyro_y, gyro_z = bno.gyro
print(f"Gyroscope\tX: {gyro_x:+.3f}\tY: {gyro_y:+.3f}\tZ: {gyro_z:+.3f}\t{rad/s}")

# オイラー角 (Roll, Pitch, Yaw)
roll, pitch, yaw = bno.euler
print(f"Euler Angle\tRoll: {roll:+.3f}\tPitch: {pitch:+.3f}\tYaw: {yaw:+.3f}\t{rad}")

# クォータニオン
quat_w, quat_x, quat_y, quat_z = bno.quaternion
print(f"Quaternion\tW: {quat_w:+.3f}\tX: {quat_x:+.3f}\tY: {quat_y:+.3f}\tZ: {quat_z:+.3f}\t")

print() # 空行

# 1 秒間隔で表示
sleep_ms(1000)

```

```

This computer
C:\inetpub\micropython\NVIDIA\PEMicro\PerfLogs\Program Files\Program Files (x86)\Python312\URALogs\Users\VMC_Camera\Windows\appverifU.dll\vfcompat.dll

MicroPython device
lib\hello.py\icm20948_esp32.py\icm20948_img_dmp3a.py\test_bno08x.py\test_bno08x.simple.py\test_icm20948_dmp.py\test_icm20948_dmp3a.py\test_icm20948_quaternion.py\test_icm20948_quaternion.py

<untitled> [ test_bno08x.py ] [ test_bno08x_simple.py ] 
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74

print(f"Acceleration\tX: {accel_x:+.3f}\tY: {accel_y:+.3f}\tZ: {accel_z:+.3f}\t{m/s²}")

# 重力ベクトル
grav_x, grav_y, grav_z = bno.gravity
print(f"Gravity\tX: {grav_x:+.3f}\tY: {grav_y:+.3f}\tZ: {grav_z:+.3f}\t{m/s²}")

# 線形加速度（重力補正済み）
linear_x = accel_x - grav_x
linear_y = accel_y - grav_y
linear_z = accel_z - grav_z
print(f"Linear Accel\tX: {linear_x:+.3f}\tY: {linear_y:+.3f}\tZ: {linear_z:+.3f}\t{m/s²}")

# ジャイロスコープ（角速度）
gyro_x, gyro_y, gyro_z = bno.gyro
print(f"Gyroscope\tX: {gyro_x:+.3f}\tY: {gyro_y:+.3f}\tZ: {gyro_z:+.3f}\t{rad/s}")

# オイラー角 (Roll, Pitch, Yaw)
roll, pitch, yaw = bno.euler
print(f"Euler Angle\tRoll: {roll:+.3f}\tPitch: {pitch:+.3f}\tYaw: {yaw:+.3f}\t{rad}")

# クォータニオン
quat_w, quat_x, quat_y, quat_z = bno.quaternion
print(f"Quaternion\tW: {quat_w:+.3f}\tX: {quat_x:+.3f}\tY: {quat_y:+.3f}\tZ: {quat_z:+.3f}\t")

print() # 空行

# 1 秒間隔で表示
sleep_ms(1000)

===== Sensor Data Update #42 =====
Acceleration X: -3.016 Y: -0.043 Z: -4.281 m/s²
Gravity X: -3.023 Y: -0.047 Z: -4.281 m/s²
Linear Accel X: +0.008 Y: +0.004 Z: +0.000 m/s²
Gyroscope X: +0.008 Y: -0.008 Z: +0.008 rad/s
Euler Angle Roll: -118.298 Pitch: +18.422 Yaw: -33.818 rad
Quaternion W: -0.787 X: +0.325 Y: -0.016 Z: +0.524

===== Sensor Data Update #43 =====
Acceleration X: -3.016 Y: -0.043 Z: -4.242 m/s²
Gravity X: -3.016 Y: -0.055 Z: -4.242 m/s²
Linear Accel X: +0.000 Y: +0.012 Z: +0.000 m/s²
Gyroscope X: +0.000 Y: +0.000 Z: -0.008 rad/s
Euler Angle Roll: -118.145 Pitch: +18.404 Yaw: -33.760 rad
Quaternion W: -0.786 X: +0.324 Y: -0.016 Z: +0.525

```

Thonny_BNO085_ プログラム実行結果

出力データの解説

```
===== Sensor Data Update #42 =====
Acceleration   X: -3.016   Y: -8.043   Z: -4.281   m/s2
Gravity        X: -3.023   Y: -8.047   Z: -4.281   m/s2
Linear Accel   X: +0.008   Y: +0.004   Z: +0.000   m/s2
Gyroscope      X: +0.008   Y: -0.008   Z: +0.008   rad/s
Euler Angle Roll: -118.298  Pitch: +18.422  Yaw: -33.818   rad
Quaternion     W: -0.787   X: +0.325   Y: -0.016   Z: +0.524
```

参考までに、重力加速度を除いた Linear Accel（線形加速度）が取れるので正しく使えばある程度の相対位置測位に利用できます。

- 意味: 重力を除いた純粋な加速度 (Acceleration - Gravity)
- 単位: m/s²
- 値の解釈:
 - X: +0.008 m/s² → ほぼゼロ（静止状態）
 - Y: +0.004 m/s² → ほぼゼロ（静止状態）
 - Z: +0.000 m/s² → 完全にゼロ（静止状態）
- 用途: 物体の実際の動きの検出、歩数計、振動検知

実用的な活用例 1. 水平器

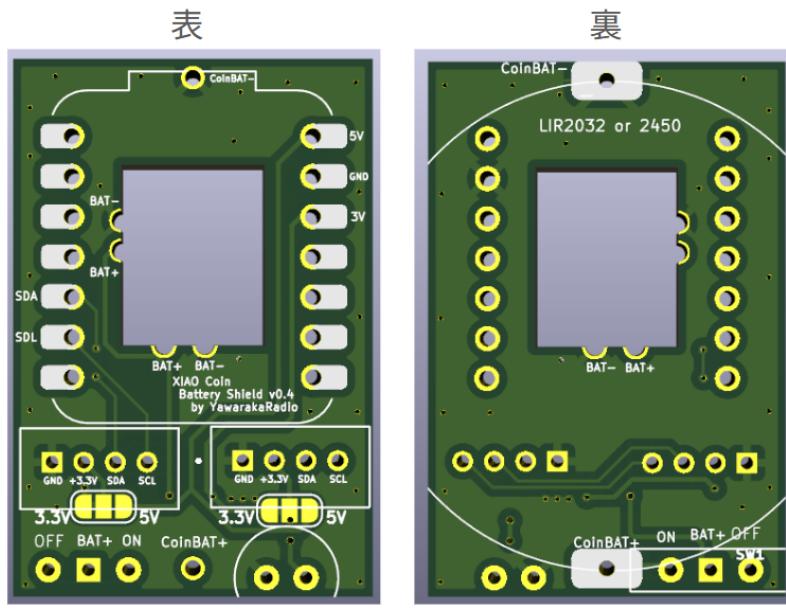
```
# Euler Angle の X, Y 成分で水平からの傾きを検出
tilt_x = roll * 180 / 3.14159 # ラジアンを度に変換
tilt_y = pitch * 180 / 3.14159
```

2. 動作検出

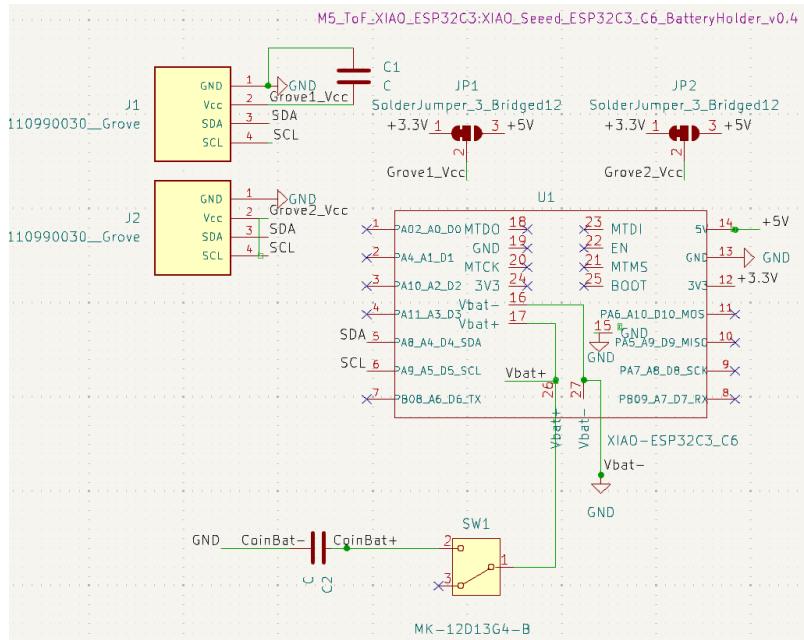
```
# Linear Accel の大きさで動きを検出
motion_magnitude = (linear_x**2 + linear_y**2 + linear_z**2)**0.5
if motion_magnitude > 0.5: # 閾値
    print("Motion detected!")
```

接続図と PCB

- ・ ソルダージャンパーは通常 3.3V に繋がっていますが、カットして 5V 側に半田付けする事で Grove の電源を 5V で駆動出来ます。
- ・ 中央を四角にカットしているので、XIAO の裏面の充電端子と半田付けしやすくなっています。
- ・ KiCAD ファイルを公開しているので、XIAO Coin を沢山つかったり、カスタマイズする際にご利用下さい。PR 頂けると嬉しいです。
 - KiCAD: https://github.com/uecken/xiao-coin/tree/main/XIAO_ButtonBatteryPCB_Public
- ・ Gerver データを JLCPCB にアップロードして発注すれば 5 枚 3\$ で約一週間で到着します。
 - Gerber URL



XIAO_PCB



XIAO_ 接続図

トラブルシューティング

- 充電されない: 電池の極性、充電端子の半田付けを確認してください
- 電圧が出ない: ショートの有無、スイッチの位置を確認してください
- I2C 通信エラー: プルアップ抵抗の追加や配線を確認してください

参考資料

- GitHub:** <https://github.com/uecken/xiao-coin/tree/main>

最新の更新内容 (2025/5/23)

1. XIAO と基板の充電端子を半田付けしやすくするため中央部分をカット (Castellated Hole 化)
2. Grove 端子の電源を Solder Jumper で 3.3V/5V 選択可能に
3. Grove 端子を表面にも追加可能に

連絡先・サポート

やわらからじお - Email: yawaraka.radio@gmail.com

