

---

# SunFounder PiPower

[www.sunfounder.com](http://www.sunfounder.com)

2023 年 10 月 11 日



# 目次

第 1 章	部品リスト	3
第 2 章	PiPower の組み立て方	5
第 3 章	機能一覧	13
3.1	詳細説明	14
3.2	バッテリーインジケータ	19
第 4 章	使い始め方	21
4.1	HassOS のインストール	21
4.2	Home Assistant に PiPower Pro を追加する	32
4.3	ダッシュボードの設定	39
4.4	コードエディターでカードを追加	41
4.5	PiPower Pro エンティティ	50
4.6	安全シャットダウンの設定	52
4.7	クーロンカウンタ (ベータ版)	62
4.8	カスタム開発	64
4.9	複数の PiPower Pro ユニット	65
4.10	IO 拡張	65
第 5 章	FAQ	67
5.1	PiPower Pro が動作しないのはなぜですか？	67
5.2	PiPower Pro はどのシングルボードコンピュータで使用できますか？	67

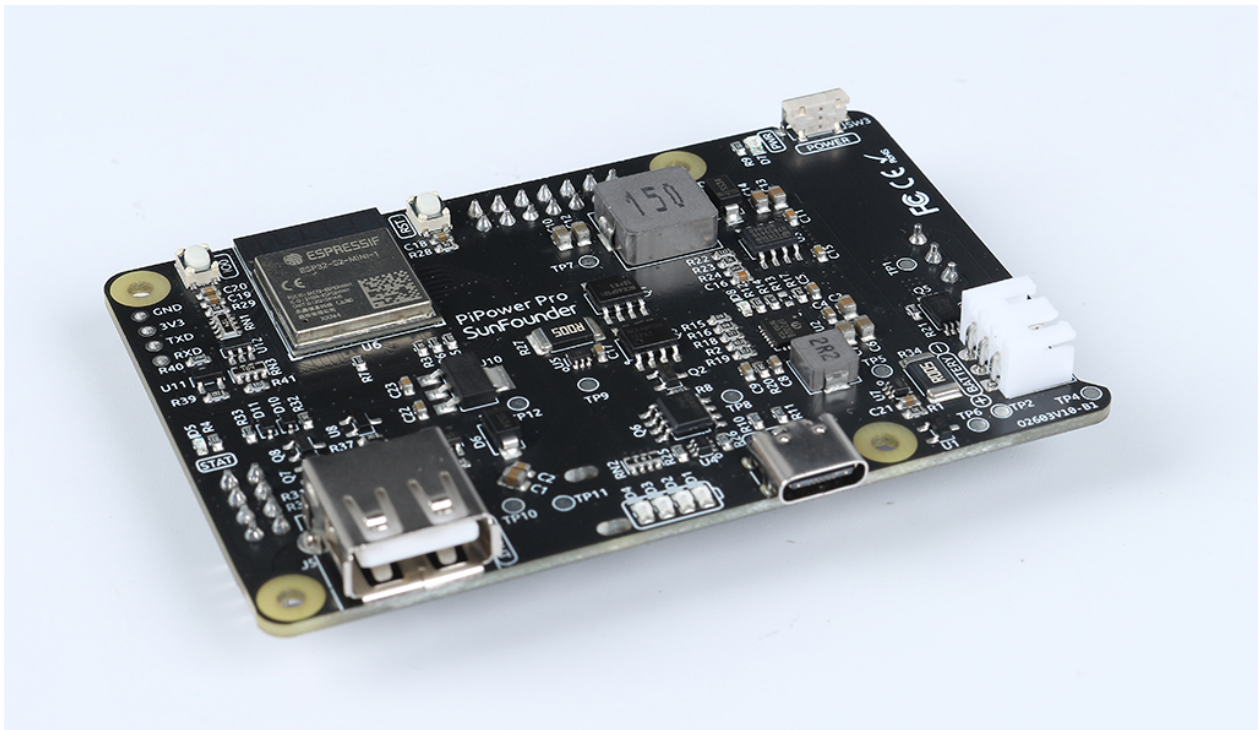


SunFounder PiPower をお選びいただき、ありがとうございます。

注釈: このドキュメントは以下の言語で利用可能です。

- 
- 
- 

ご希望の言語でドキュメントにアクセスするために、それぞれのリンクをクリックしてください。



## UPS とは何か？

Raspberry Pi のプロジェクトが常時電源を必要とする場合、主電源のみに依存するのは適切ではありません。地域によっては、頻繁に電源が落ちたり、サージが発生したりすることがあります。電源の変動は Raspberry Pi を破損させる可能性があり、電源が途絶えると、Raspberry Pi はすぐにシャットダウンします。したがって、安全にシャットダウンすることができず、SD カード上の全てのデータが失われるリスクが高まり、破壊される可能性が増加します。

そのため、無停電電源装置 (UPS) の使用が推奨されます。

UPS を使用すると、メイン電源からの電源遮断が発生した場合 (遮断 = 停電)、バッテリーや他の電源が引き継いでデバイスに電力を供給し続け、シャットダウンすることなく動作します。UPS は、非常電源として考えることが多いです。主電源が復旧した後、UPS は再充電され、次のトラブルに備えることができます。

### PiPower について

これが、最初に PiPower を設計した理由です。PiPower は、Raspberry Pi の第二の電源として使用できます。PiPower に USB-C メイン電源アダプタを接続すると、Raspberry Pi に直接電力を供給し、バッテリーを低電流で充電します。停電や USB-C メイン電源の切断が発生した場合、PiPower は Raspberry Pi をシームレスに電源供給できます。

PiPower は、5V/3A の電源供給を出力でき、様々な Raspberry Pi の使用状況に対応しています。4 つの電源インジケータがあり、各インジケータは電源の 25% を示し、Raspberry Pi の電源を入/切するスイッチも装備しています。

**警告:** バッテリーを初めて入れるときや、バッテリーを取り外して再度入れるときには、正常に動作しない場合があります。このような場合、Type C ケーブルを充電ポートに差し込んで、保護回路をオフにする必要があります。その後、バッテリーは正常に使用できます。

### PiPower Pro について

PiPower Pro は、PiPower を基盤に、ESP32 S2 モジュールを統合してモジュールのバッテリー電圧や電流状態、入出力電圧、電流をリアルタイムで監視できるようにしました。また、充電電流のインテリジェントな調整や、入力電源とバッテリー電源の間でのシームレスな切り替えを特長としており、連続した電源出力が確保されます。

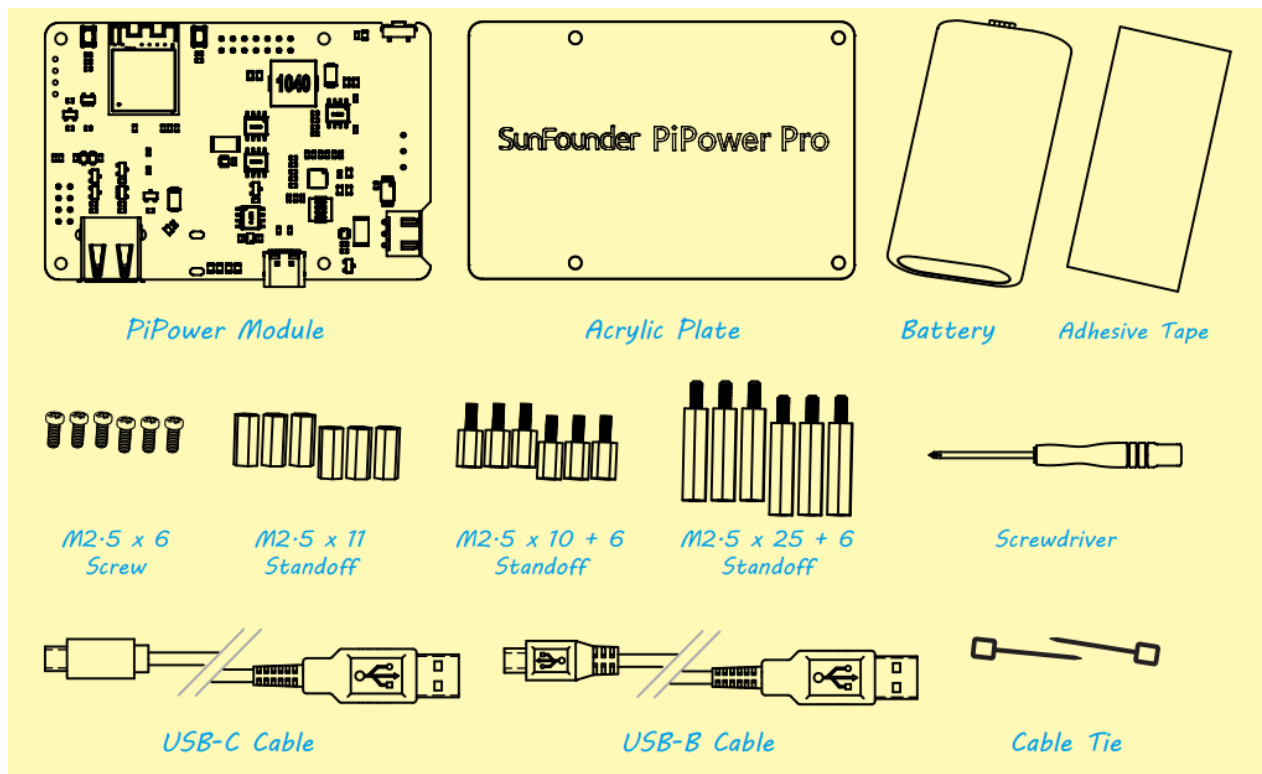
Home Assistant と統合することで、ユーザーはすべてのパラメータデータに簡単にアクセスし、スマートデバイスのシナリオの自動化を設定できます。

さらに、PiPower Pro は、サブデバイスのオン/オフ状態を制御するための外部 IO インターフェースを提供しています。オープンソースの ESPHome 設定を利用して、ユーザーは IO 機能をカスタマイズし、さらに多くのセンサーをシステムに追加することができます。

製品を使用中に何か質問があれば、[service@sunfounder.com](mailto:service@sunfounder.com) までメールをお送りください。できるだけ早くご返信いたします。

## 第 1 章

## 部品リスト







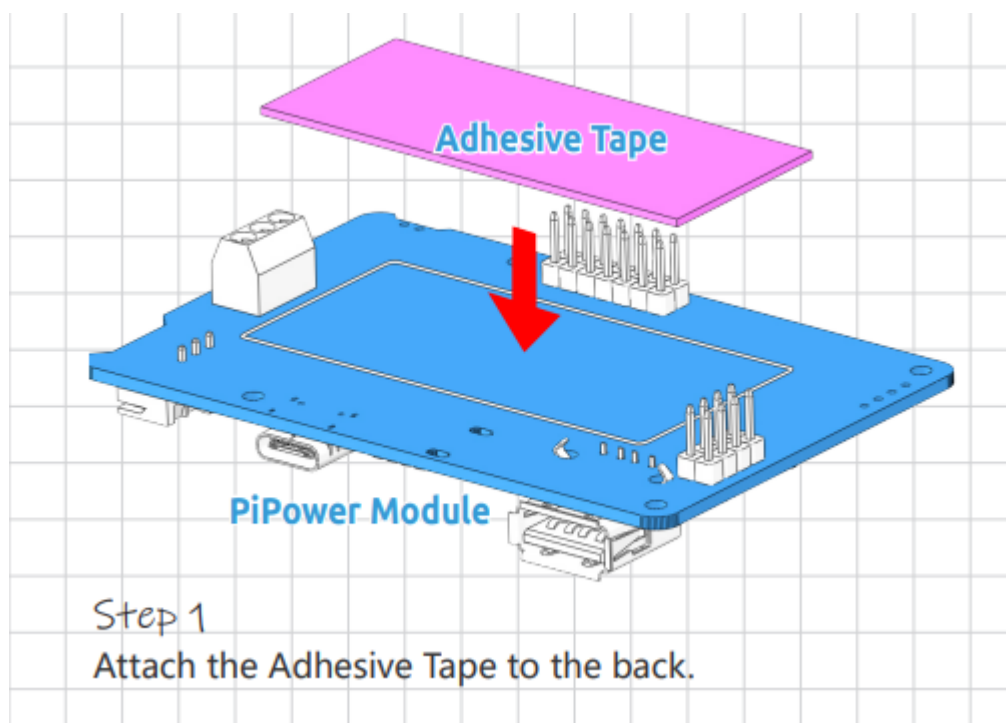
## 第 2 章

# PiPower の組み立て方

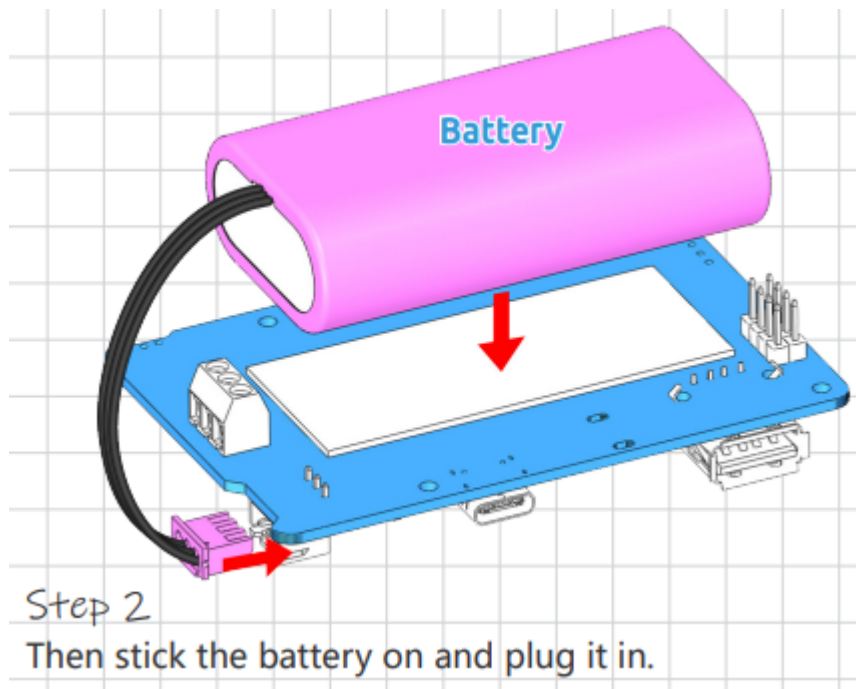
パッケージ内のコンポーネントに馴染んだ後、PiPower の組み立てを開始します。

次のステップでは、特にバッテリーの取り付け位置や透明アクリル裏蓋に注意しなければならない詳細が多くあります。

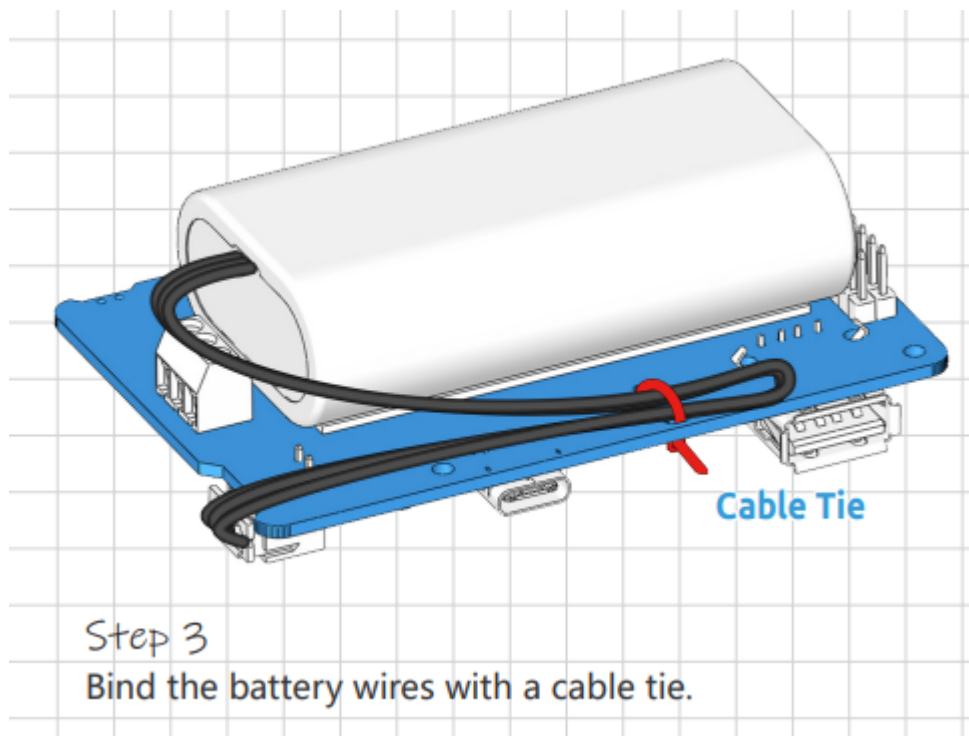
1. 裏側に粘着テープを貼り付けます。



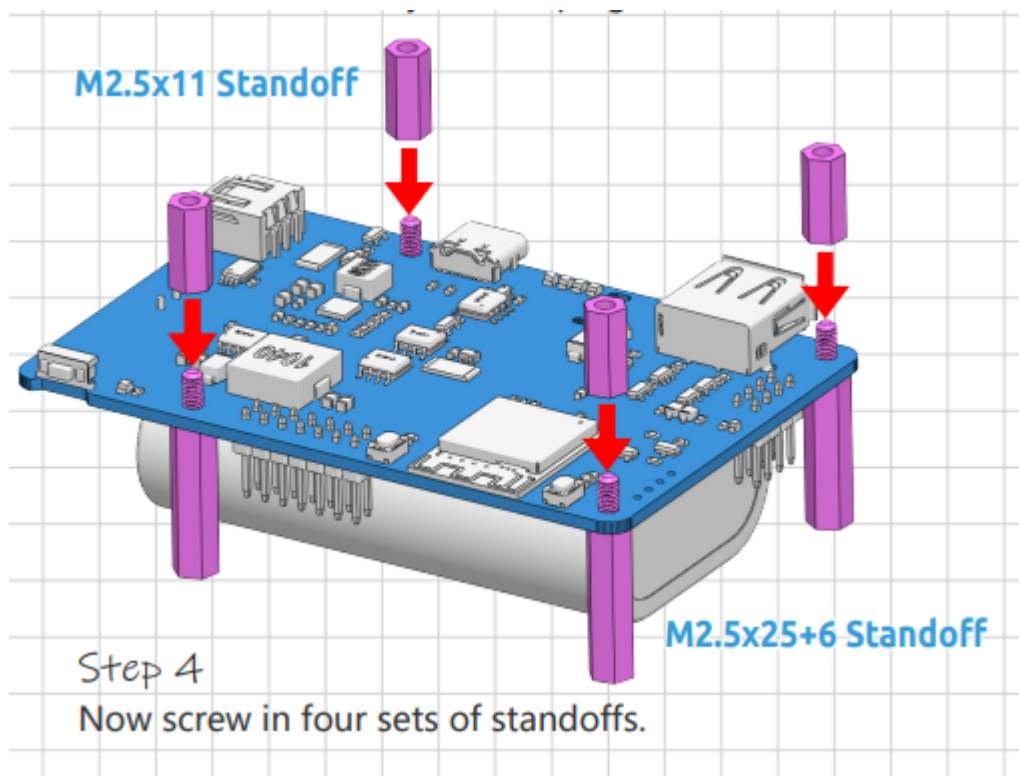
2. 次に、バッテリーを貼り付けて接続します。



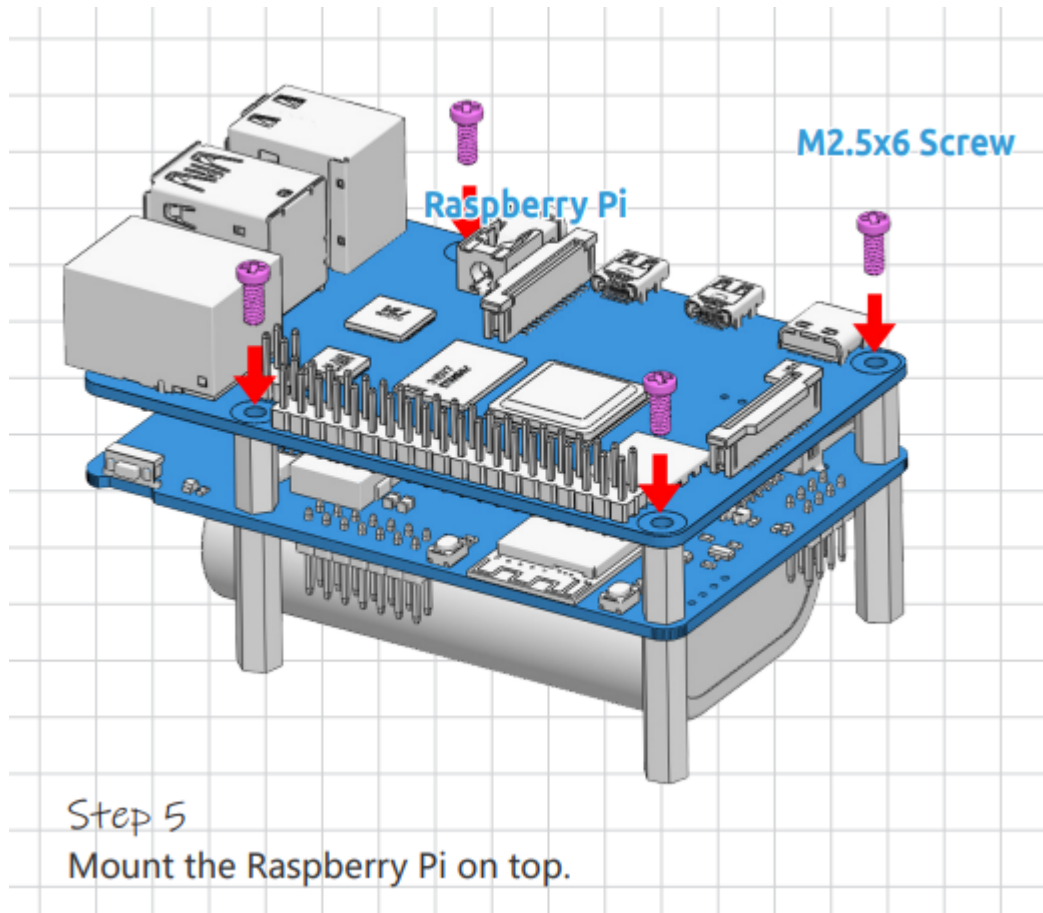
3. バッテリーのワイヤーをケーブルタイで束ねます。



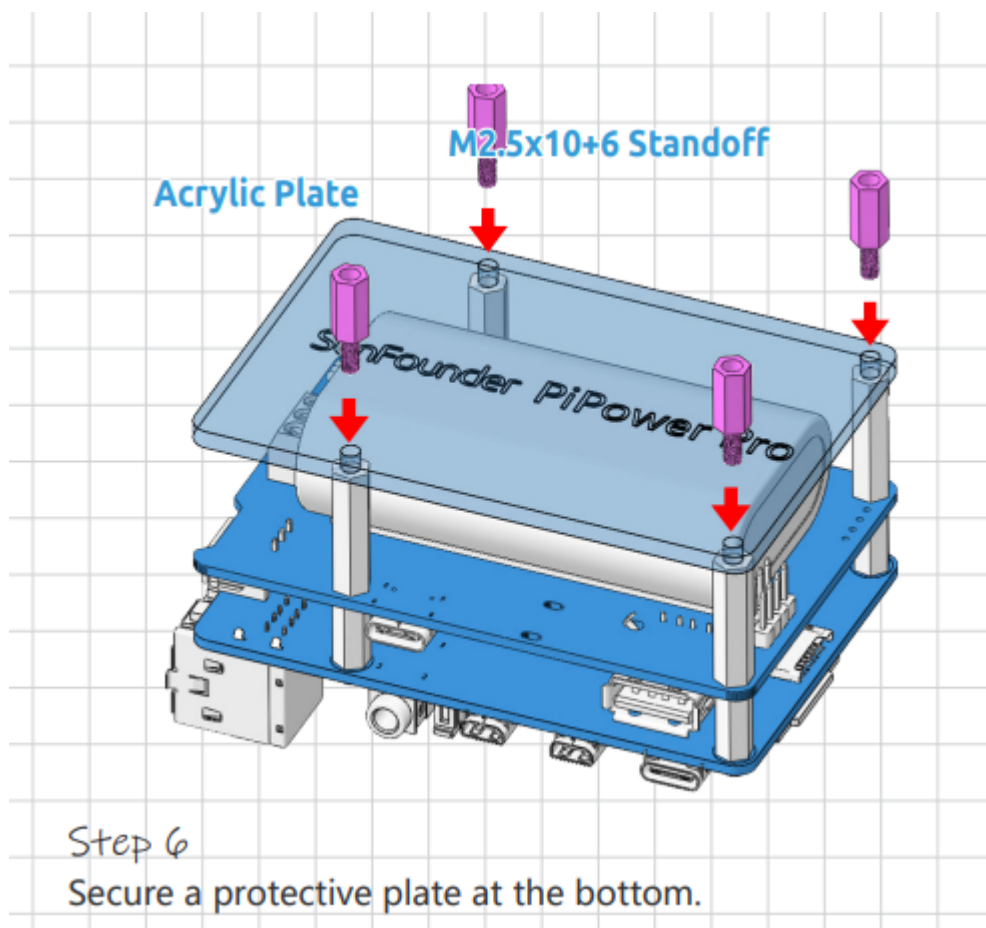
4. 4組のスペーサーを取り付けます。



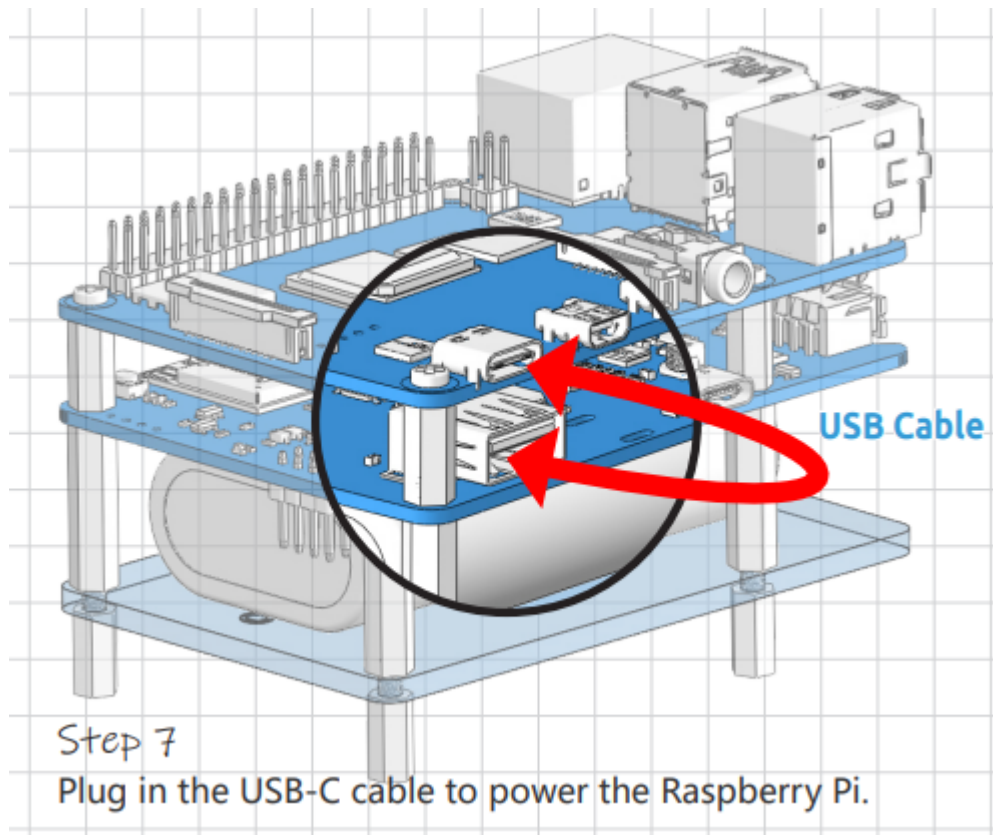
5. Raspberry Pi を上部に取り付けます。



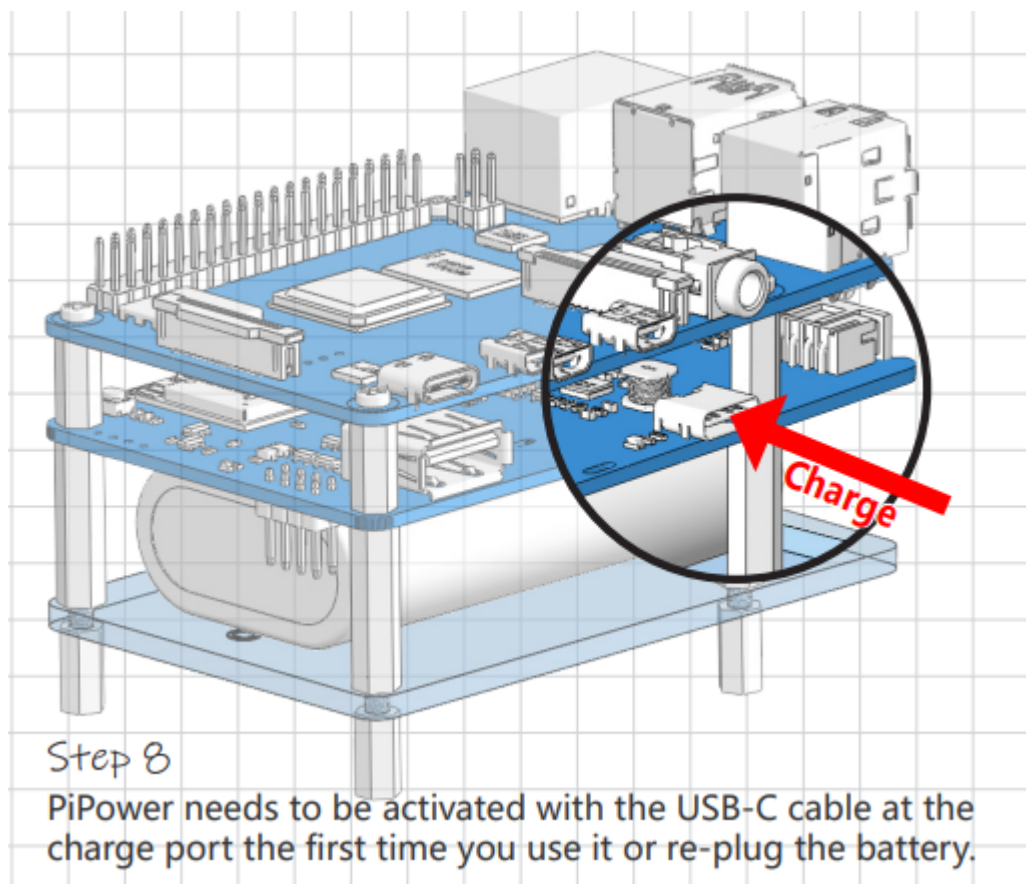
6. 下部に保護プレートを固定します。



7. USB-C ケーブルを接続して、Raspberry Pi に電力を供給します。



8. PiPower を初めて使用する際や、バッテリーを再度接続する際は、USB-C ケーブルを充電ポートに接続して起動する必要があります。



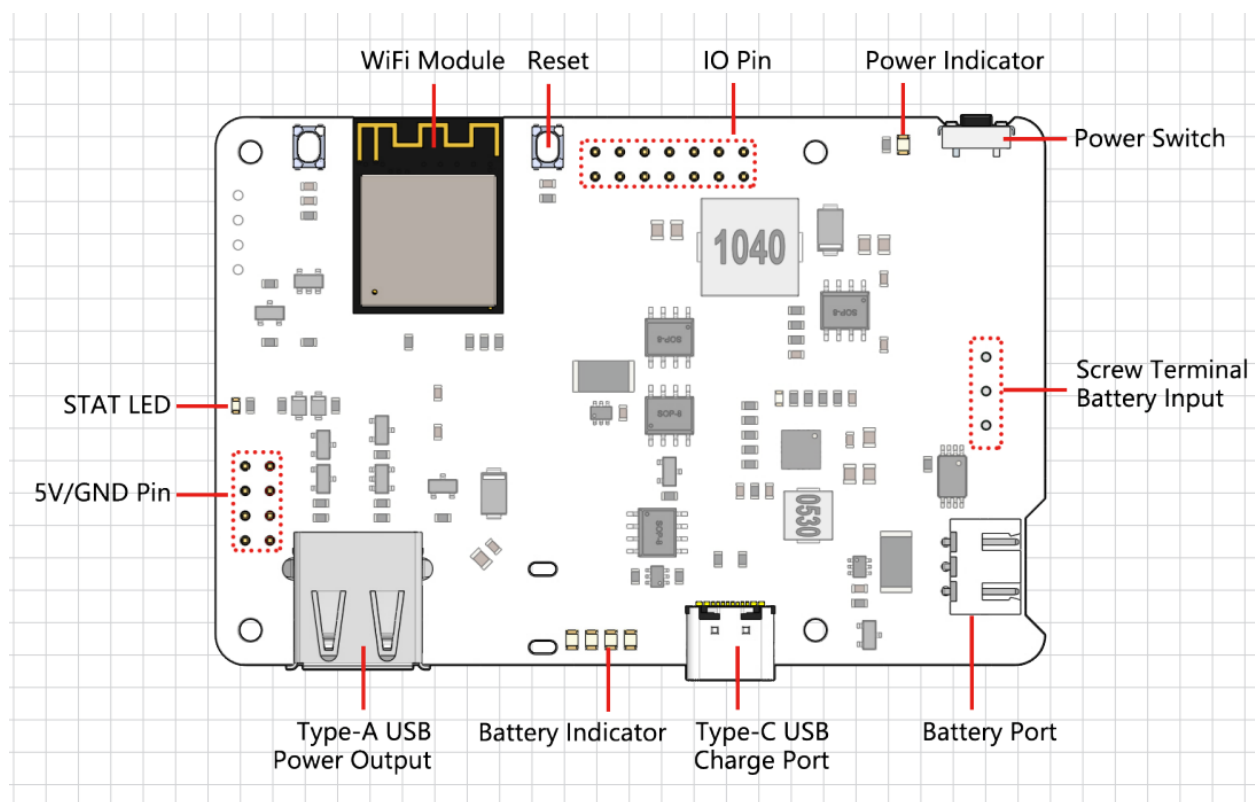
警告：バッテリーを初めて取り付ける時や、取り外して再度取り付ける時に、正常に動作しない場合があります。この場合、Type C ケーブルを充電ポートに接続して、保護回路をオフにする必要があります。その後、バッテリーは正常に使用できます。





## 第 3 章

# 機能一覧



- パススルー充電
- シャットダウン時の電流 :  $< 0.5\text{mA}$
- 入力 :
  - USB Type-C, 5V/3A
  - バッテリー入力
- 出力 :

- USB Type-A, 5V/3A
- 2x4P P2.54 ピンヘッダー
- 充電電力：5V/2A
- 搭載バッテリー
  - タイプ: 3.7V リチウムイオンバッテリー x 2
  - 容量: 2000mAh
  - コネクタ: PH2.0, 3P
- 過放電保護電圧：6.0V
- 過充電保護電圧：8.4V
- サイズ: 90mm x 60mm x 24.9mm
- 基板上的インジケータ
  - 1 x 充電インジケータ (CHG)
  - 1 x 電源インジケータ (PWR)
  - 4 x バッテリーインジケータ (D4 ~ D7)
- 基板上的電源スイッチ
- 基板上的 MCU ESP32 S2

### 3.1 詳細説明

#### STAT LED

ESP32 S2 のステータスを示すインジケータです。

- 消灯: ESP32 S2 の電源がオフ。
- ゆっくり点滅: ESP32 S2 の電源はオンですが、Wi-Fi は接続されていません。
- 点灯: ESP32 S2 の電源がオンで、Wi-Fi が接続されています。

---

注釈: 「ESP32 S2 がオフ」という状態は、USB Type C の電源が接続されている状況を指します。この状態では、ESP32 S2 は「電源がオフ」という技術的な意味でオフですが、完全にはシャットダウンしていません。電源 LED は ESP32 S2 の制御下で点灯し、一部の機能が稼働している場合があります。USB Type C の電源を取り外すと、ESP32 S2 は完全にシャットダウンします。

---

電源経路の切り替え

PiPower Pro は、最大出力保護を確保するために、自動的に電源経路を切り替える機能を統合しています。

- 1. 外部電源が接続されている場合、5V の出力は外部の USB Type C を介して直接供給され、スイッチでオフにすることができます。外部の電源は、入力電圧が 4.6V を超えることを確認しながら、最大電流でバッテリーを充電します。
- 2. 電源が切断される瞬間、システムはシームレスにバッテリー電源への出力に切り替わり、電源の中断中でもシステムが正常に動作することを確保します。
- 3. 外部電源が 4.6V 未満の場合、システムは外部デバイスの電力喪失を防ぐためにバッテリーバックアップ電源に自動的に切り替わります。

表 1 出力電源のロジック

スイッチ	外部電源	出力ステータス
オン	接続中	外部電源
オン	接続解除または 4.6V 以下の電圧	バッテリー電源
オフ	接続中	オフ
オフ	接続解除または 4.6V 以下の電圧	オフ

充電電力

電源がオンの状態では、入力電圧に基づいて充電電流が自動的に調整されます。

表 2 充電電流のロジック

スイッチ	充電電流
オン	入力電圧に基づいて調整
オフ	2A

- 1. スwitchが「オフ」の状態の場合、PiPower Pro は外部に電源を供給せず、充電電流は最大 2A に達し、迅速に充電することができます。0% から 100% までの充電時間は約 2 時間 10 分です。
- 2. 「オン」の状態では、PiPower Pro が外部に電源を供給する必要があり、外部の USB もバッテリーに電源を供給する必要があります。USB の電源供給の電圧が安定していることを確保するために、入力電圧に基づいて充電電流が調整され、電圧が 4.6V 以上に保たれるようにします。

過放電保護

シングルバッテリーの電圧が 3V 以下になると、バッテリー保護が作動し、バッテリーはこれ以上放電されません。

バッテリーが取り外されると、オンボードの過放電保護回路のメカニズムにより、電圧は低すぎると見なされ、保護回路が作動します。PiPower にバッテリーを再度接続すると、バッテリーは正常に動作しません。この場合、Type C ケーブルを充電ポートに差し込むことで保護回路をオフにし、バッテリーを通常通り使用できます。

### 過充電保護

バッテリーの合計電圧が 8.4V に達すると、充電は終了します。

### 充電バランス

2 つのバッテリーの電圧が等しくない場合、2 つのバッテリーの充電電流は自動的に調整され、2 つのバッテリーがバランスを取るようになります。

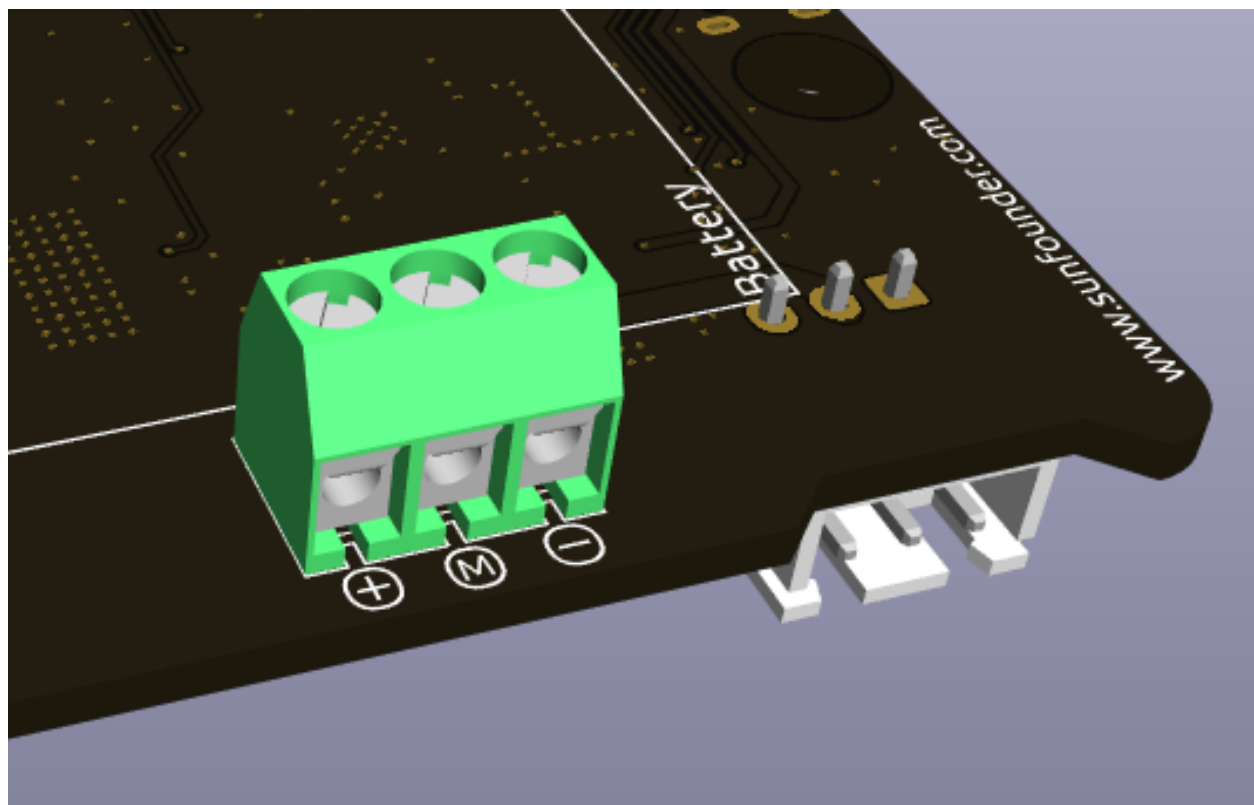
シングルバッテリーが 4.2V を超えると、電圧分配抵抗チャネルが導通し、バッテリーの充電電流が減少または放電されます。

### バッテリー

この製品には、2 つの 3.7V 18650 リチウムイオンバッテリーがシリーズ接続されており、XH2.54 3P コネクタを備え、定格容量は 2000mAh です。

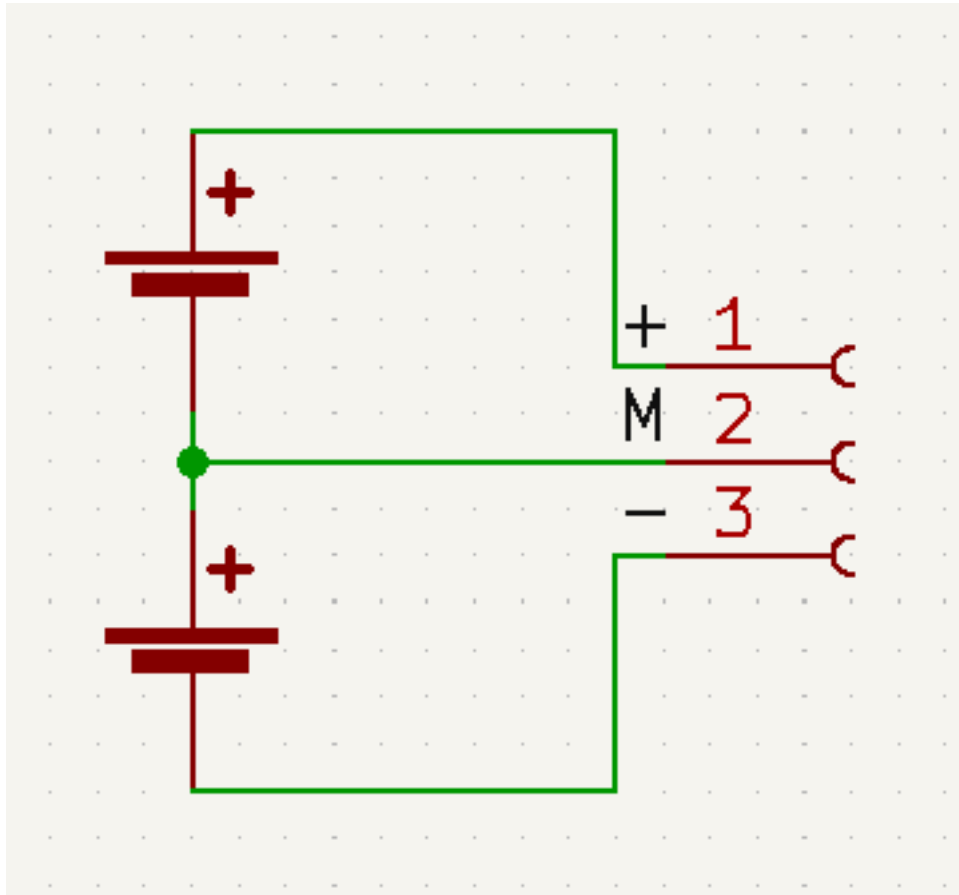
- 成分: リチウムイオン (Li-ion)
- 容量: 2000mAh、14.8Wh
- 重量: 90.8g
- セル: 2
- コネクタ: XH2.54 3P
- 過充電保護電圧: セルあたり 4.2V
- 過放電保護: 3V

### 外部バッテリー



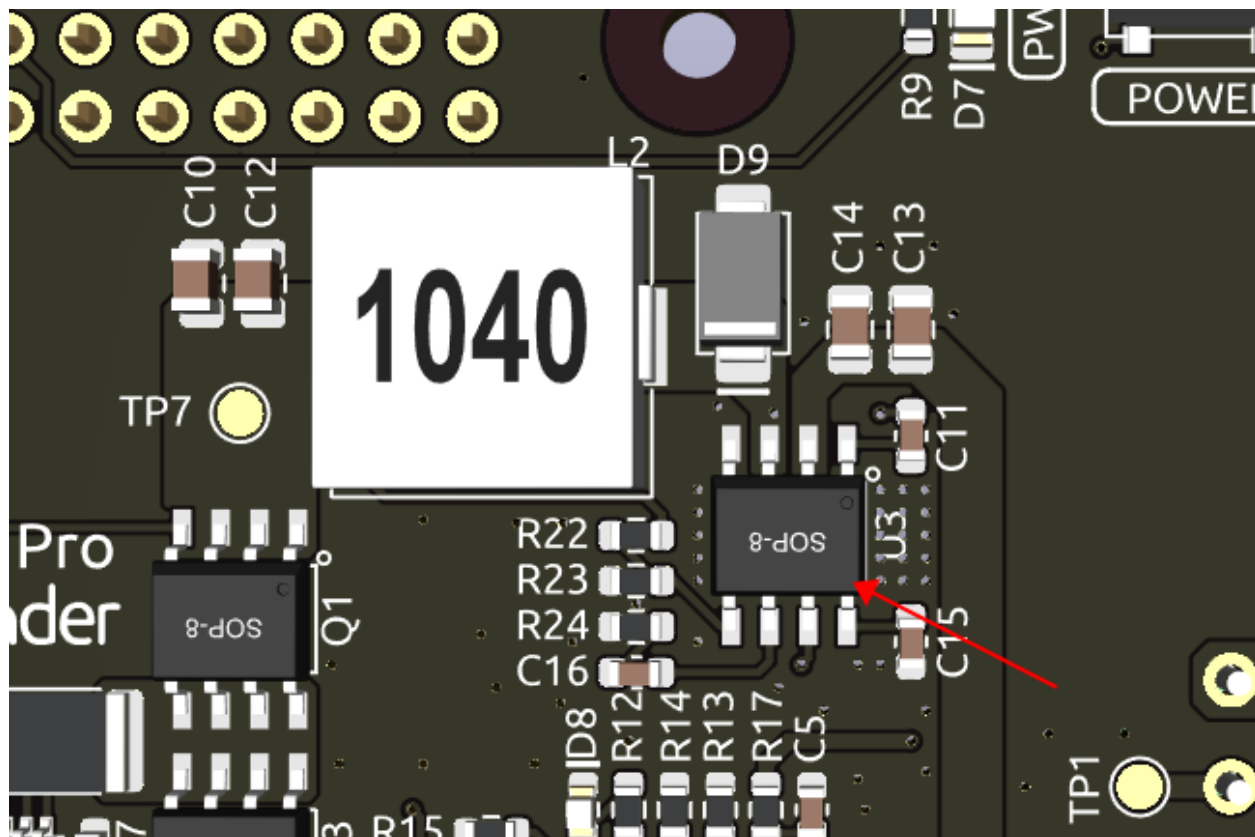
ねじ端子を使用して、自分のバッテリーを接続できます。デバイスは、3.7V のリチウムイオンまたはリチウムポリマーバッテリー 2 つのみをサポートしています。バッテリーには保護ボードが付いていることが望ましく、15W 以上の出力が確保されることが推奨されます。

警告：同時に外部バッテリーと付属のバッテリーを接続しないでください！



### 温度

出力電力が最大公称 5V/3A に達すると、DC-DC 降圧チップ U1 の温度は約 70-80℃ に上昇しますので、火傷を防ぐために触れないよう注意し、通気を確保してください。温度が DC-DC の保護温度 75℃ に達すると、DC-DC は過熱損傷を防ぐためにシャットダウンします。



### D8 LED

D8 LED は、IP2326 充電チップによって提供される充電ステータスインジケータです。当初、このライトは、充電状態とバッテリーの異常を示すために設計されました。しかし、充電出力で電流の流れがあるかどうかのみを検出できます。この出力電流は、DC-DC コンバータを介して 5V を出力するためにルーティングできます。簡単に言えば、入力電力が不足している場合、バッテリーが電源を補うと、LED は点灯し続け、誤解を招く可能性があります。ただし、LED はバッテリーが正常に機能しているかどうかを示すことができるため、残されました（バッテリーが挿入されていない場合、LED が点滅します）。

## 3.2 バッテリーインジケータ

バッテリーインジケータと電圧との関係は以下の通りです：

- 4 つの LED すべて点灯：電圧 > 7.7V
- 3 つの LED 点灯：電圧 > 7.2V
- 2 つの LED 点灯：電圧 > 6.7V
- 1 つの LED 点灯：電圧 > 6.4V
- 4 つの LED すべて消灯：電圧 < 6V、この時、バッテリーを充電する必要があります。





## 第 4 章

# 使い始め方

PiPower Pro は Home Assistant に統合できます。これを行うには、HassOS がインストールされた Raspberry Pi が必要です。セットアップのための下記のリンクをフォローしてください。

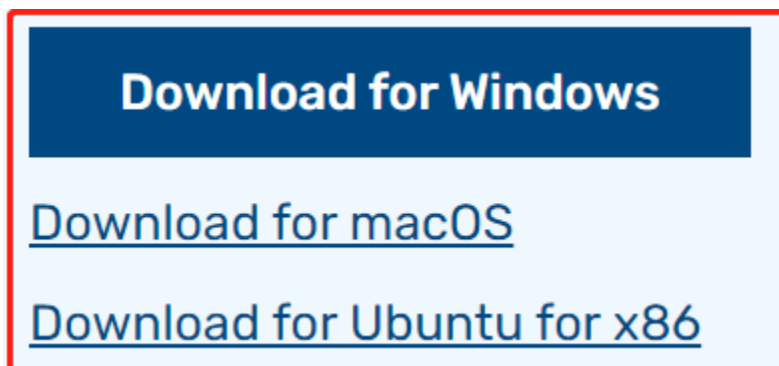
### 4.1 HassOS のインストール

このセクションでは、Raspberry Pi に Home Assistant オペレーティングシステムをインストールする手順を案内します。Raspberry Pi のシステムにある既存の内容がすべて失われることに注意してください。手順を進める前にデータのバックアップが重要です。

#### ステップ 1

Raspberry Pi は、Mac OS、Ubuntu 18.04、および Windows で動作するグラフィカルな SD カードライティングツールを開発しています。これは、イメージをダウンロードして SD カードに自動でインストールするため、多くのユーザーにとって最も簡単な選択です。

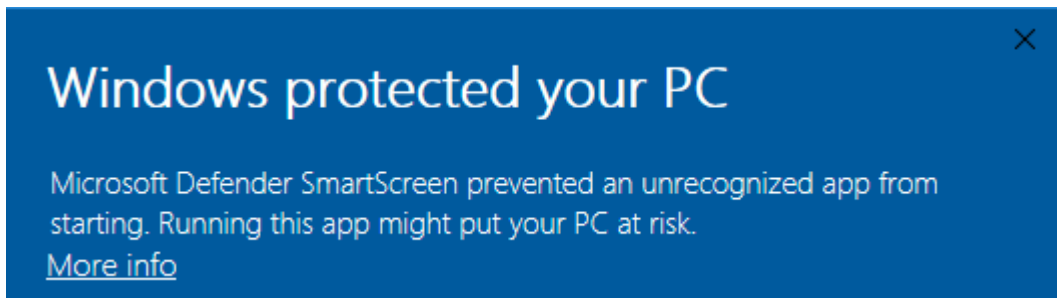
ダウンロードページ <https://www.raspberrypi.org/software/> にアクセスします。ご利用のオペレーティングシステムに対応する **Raspberry Pi Imager** のリンクをクリックし、ダウンロードが完了したらインストーラーを起動します。



#### ステップ 2

インストーラーを起動すると、オペレーティングシステムが実行をブロックしようとする場合があります。例として、Windows では次のようなメッセージが表示される場合があります。

このポップアップが表示された場合、**More info** をクリックしてから **Run anyway** をクリックし、Raspberry Pi Imager のインストール手順に従います。

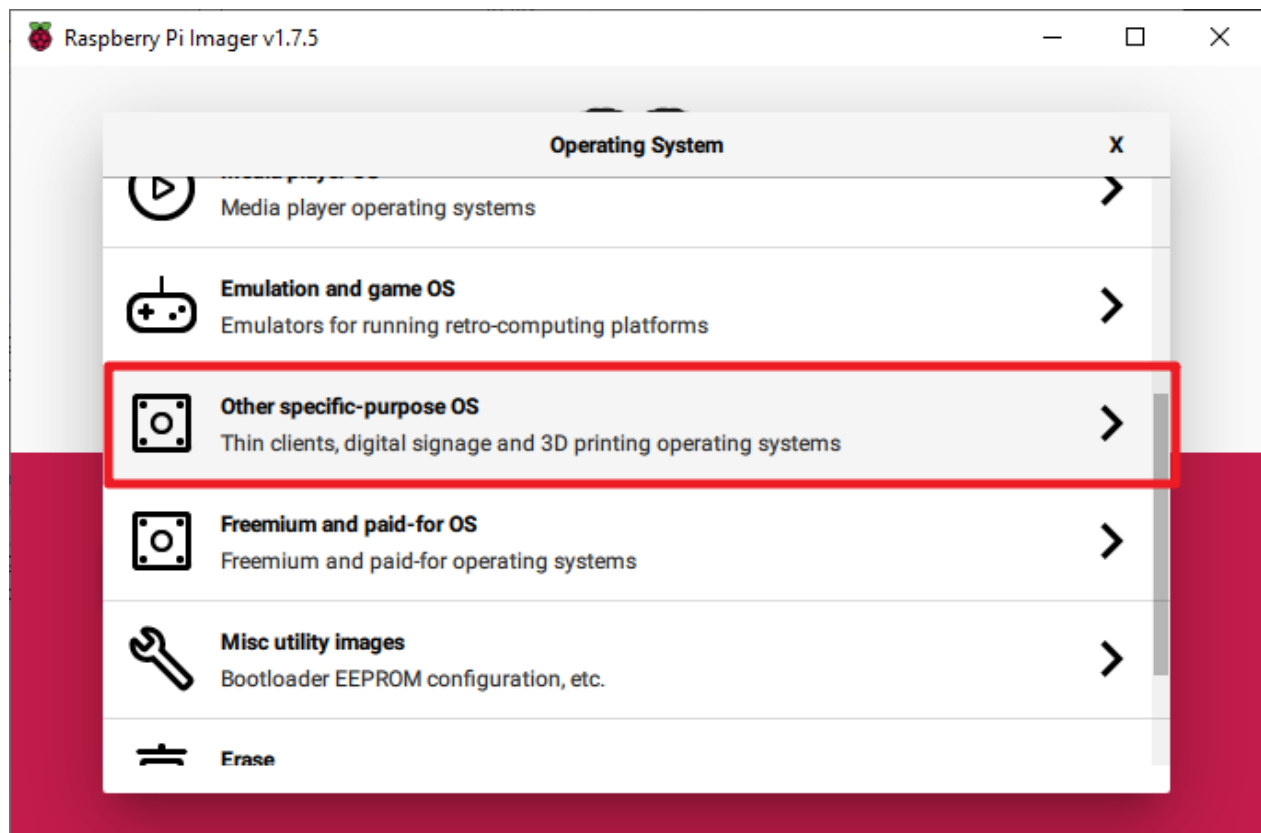


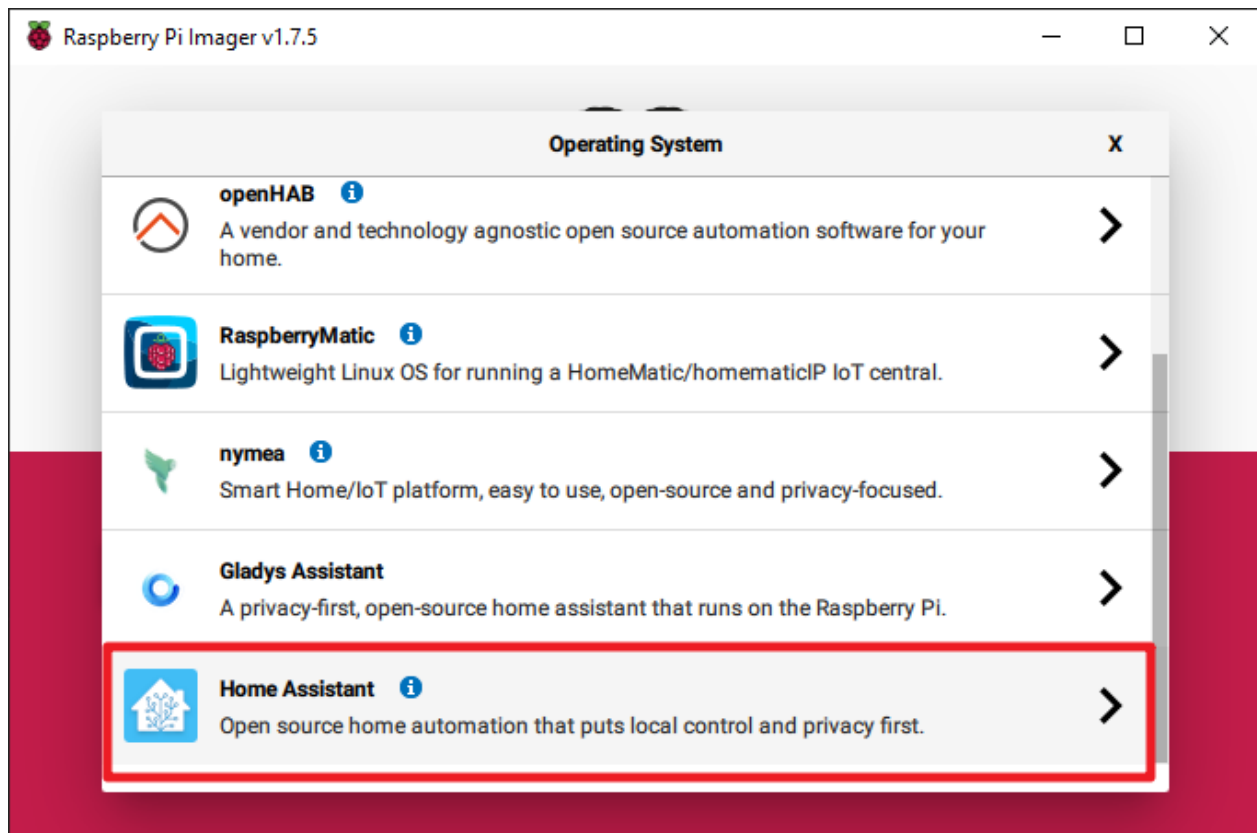
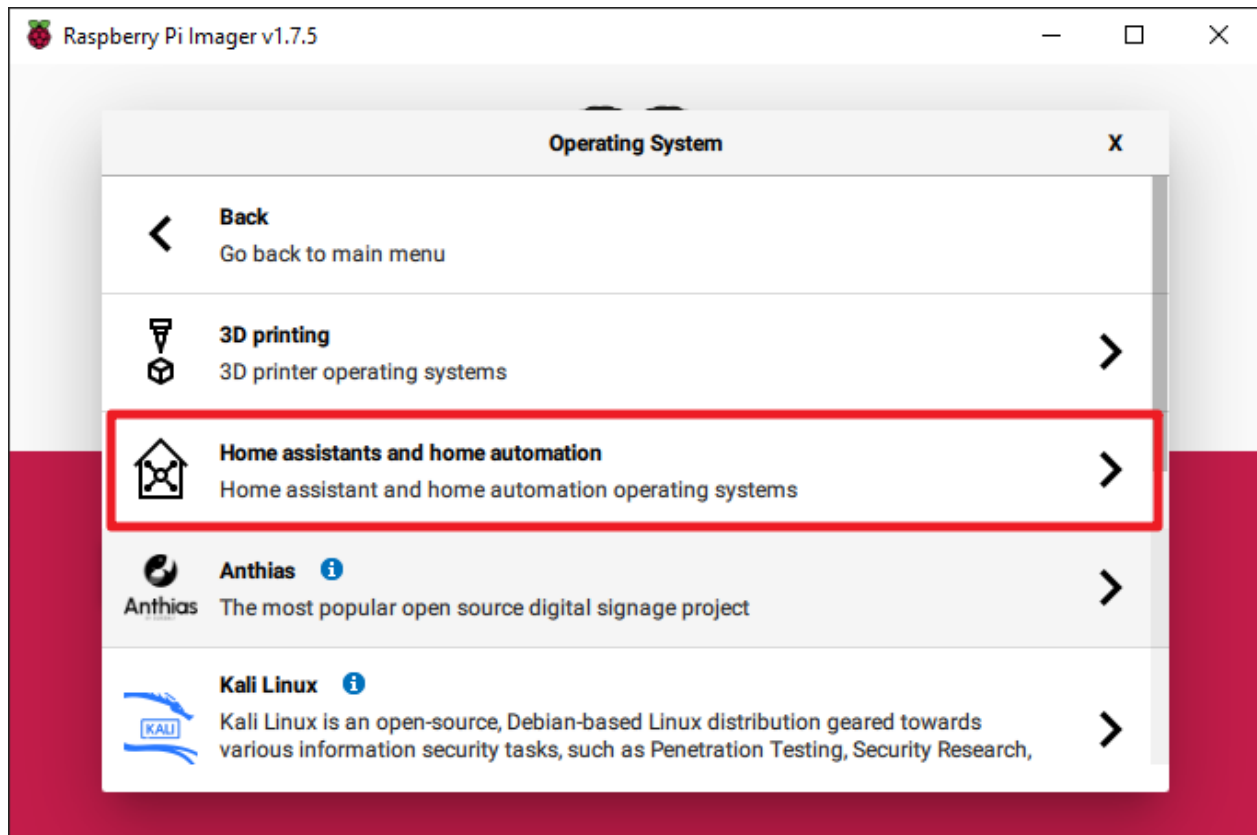
### ステップ 3

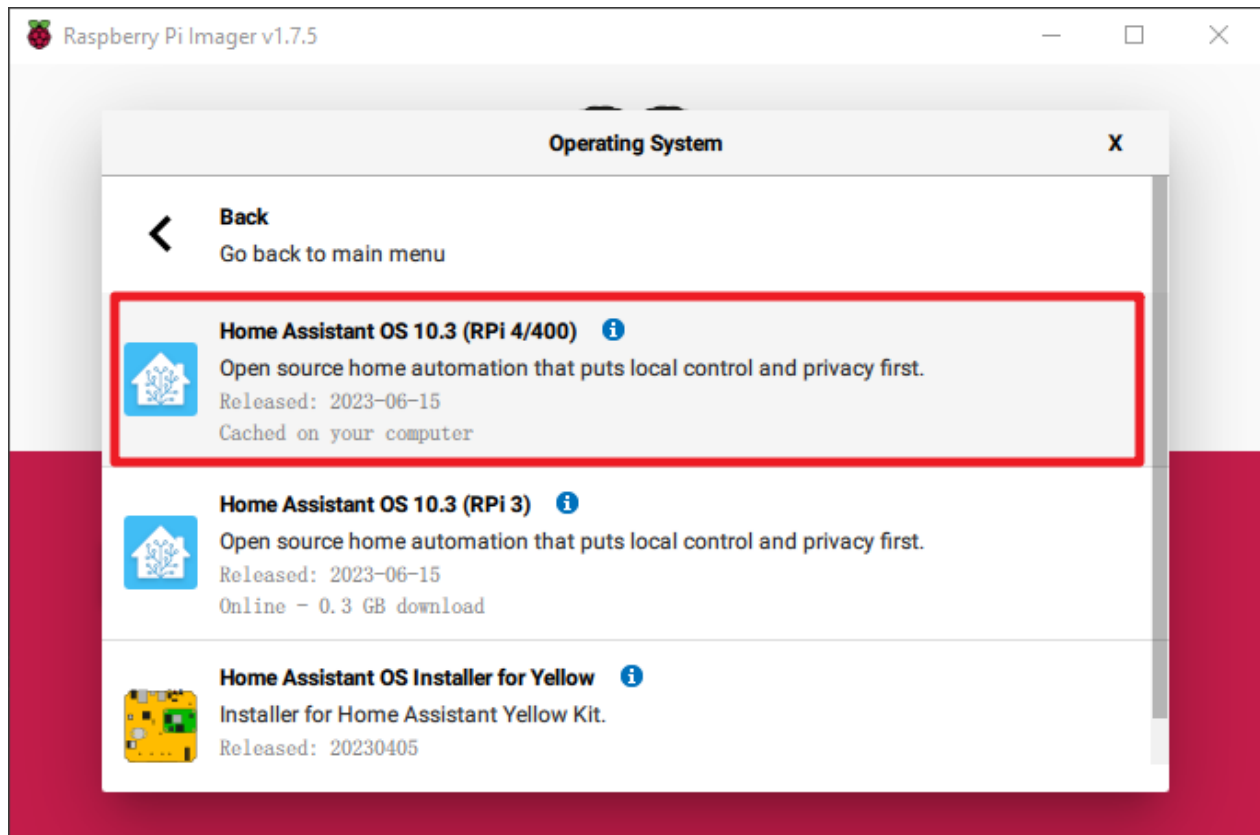
SD カードをコンピューターやラップトップの SD カードスロットに挿入します。

### ステップ 4

Raspberry Pi Imager で、インストールしたい OS とインストールする SD カードを選択します。

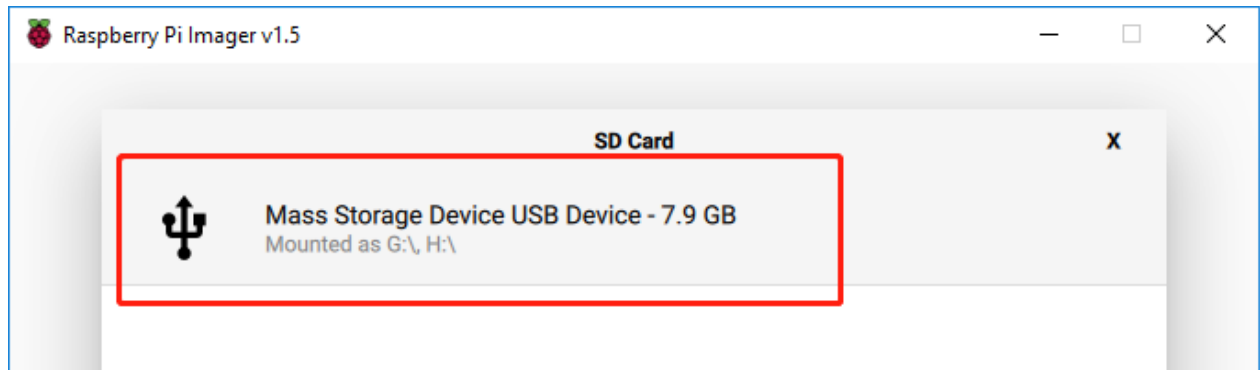






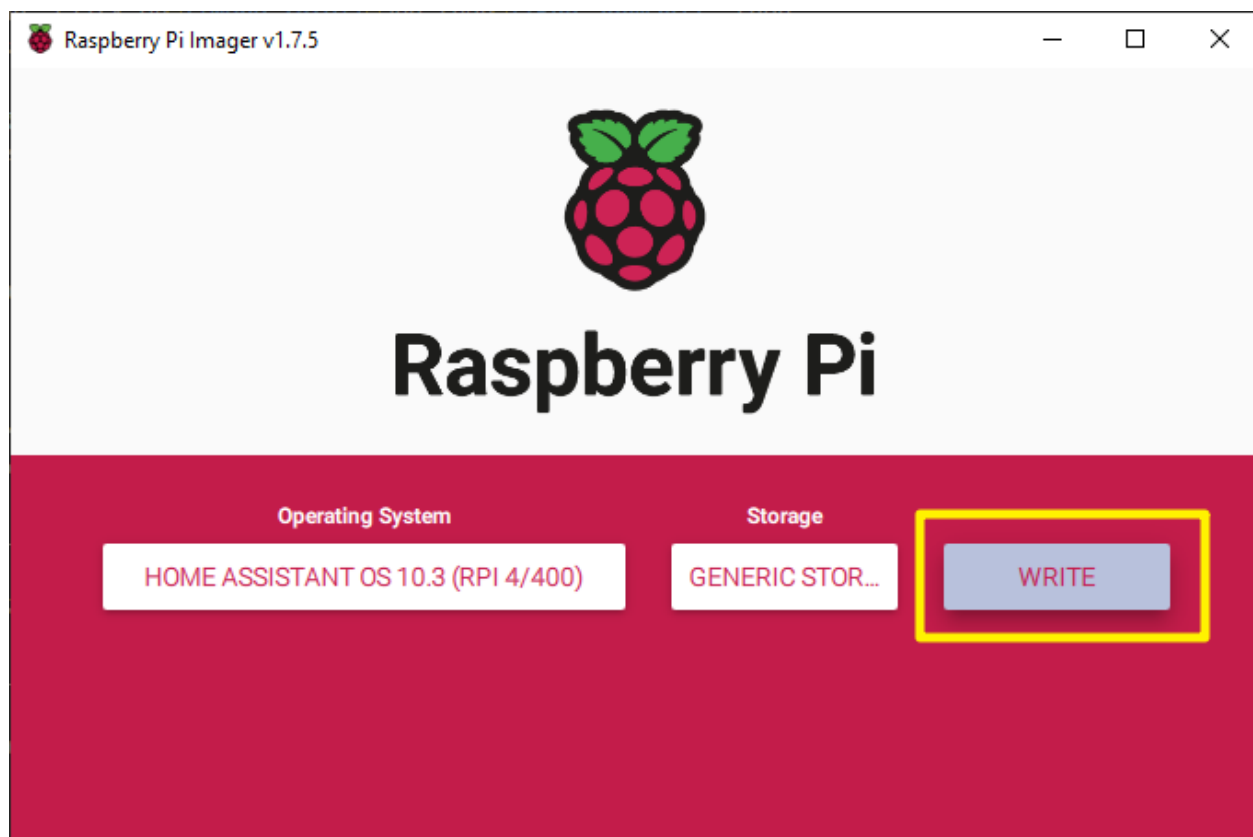
#### ステップ 5

使用する SD カードを選択します。



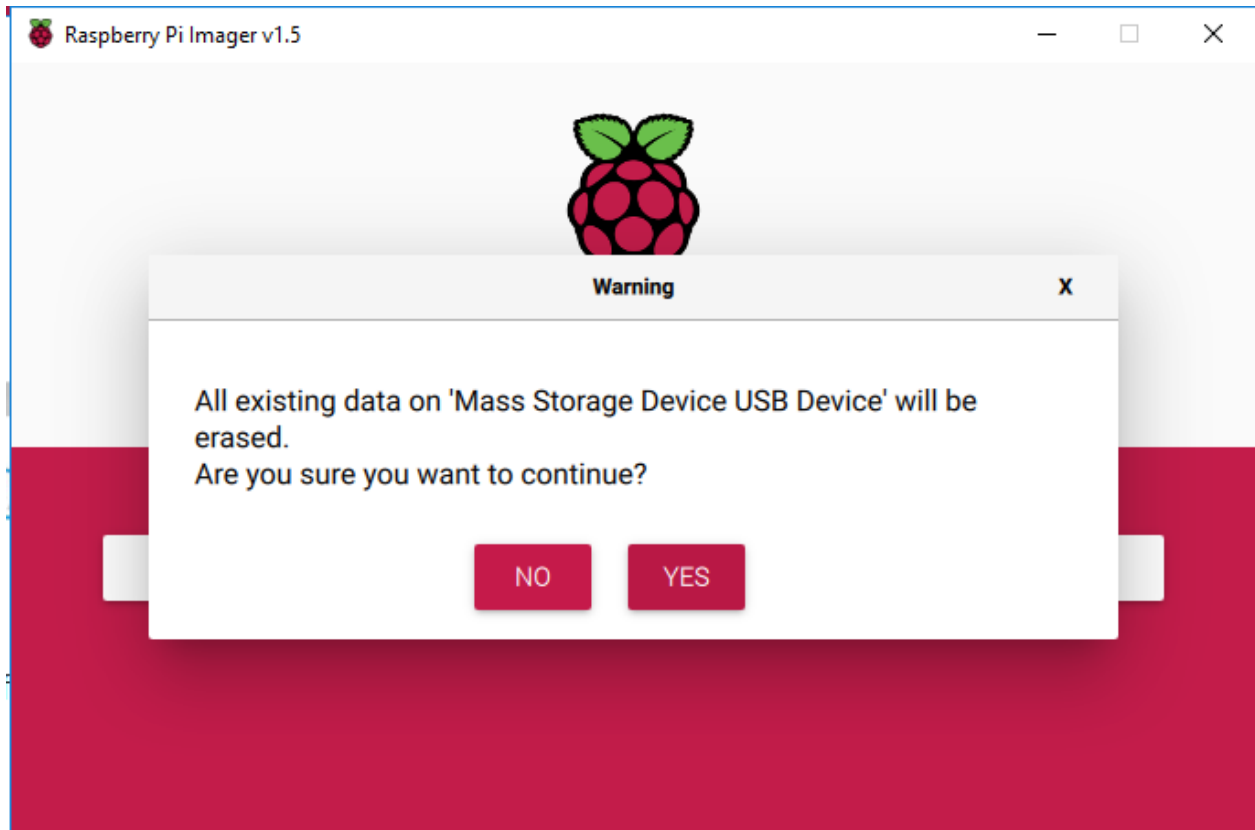
#### ステップ 6

WRITE ボタンをクリックします。



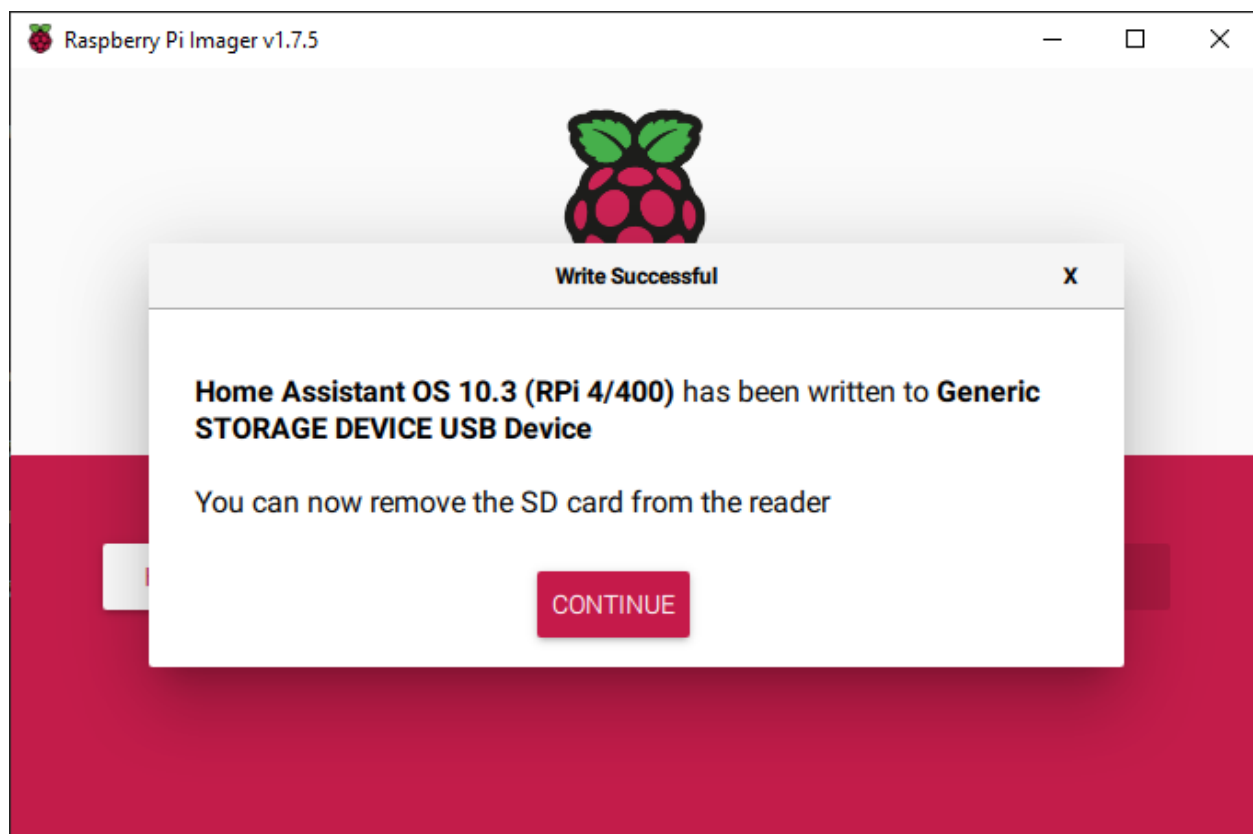
#### ステップ 7

SD カードにファイルがある場合は、それらのファイルをバックアップして永久に失うことを防ぎます。バックアップするファイルがない場合は、Yes をクリックします。



#### ステップ 8

しばらく待った後、書き込みが完了したことを示すウィンドウが表示されます。

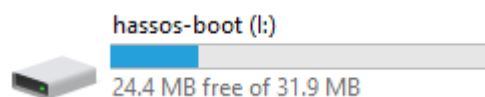


## ステップ 9

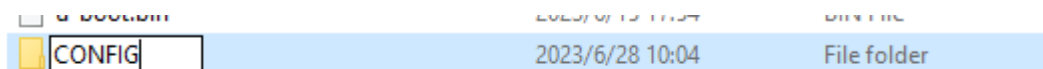
次に、Pironman の WiFi を設定します。

注釈: 有線接続を使用してネットワークにアクセスする予定の場合、このステップはスキップできます。

ファイルエクスプローラーを開き、Hassio-boot という名前の SD カードにアクセスします。



ルートパーティションに CONFIG という新しいフォルダを作成します。



CONFIG フォルダ内に network というフォルダを作成します。

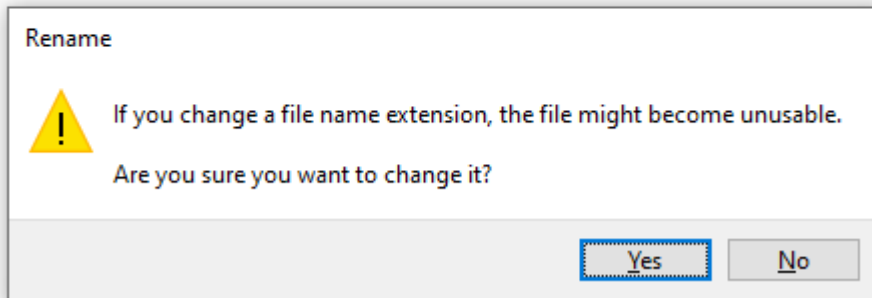
This PC > hassos-boot (I:) > CONFIG >

Name	Date modified	Type
modules	2023/6/28 10:10	File folder
network	2023/6/28 11:34	File folder

network フォルダ内に、拡張子なしで my-network という新しいテキストファイルを作成します。

This PC > hassos-boot (I:) > CONFIG > network

Name	Date modified	Type
my-network	2023/6/28 11:34	Text Docu



my-network ファイルに次のテキストを書き込み、MY\_SSID と MY\_WLAN\_SECRET\_KEY をご自身のネットワークの SSID とパスワードに置き換えます：

```
[connection]
id=my-network
uuid=72111c67-4a5d-4d5c-925e-f8ee26efb3c3
type=802-11-wireless

[802-11-wireless]
mode=infrastructure
ssid=MY_SSID
# Uncomment below if your SSID is not broadcasted
#hidden=true

[802-11-wireless-security]
auth-alg=open
key-mgmt=wpa-psk
psk=MY_WLAN_SECRET_KEY
```

(次のページに続く)



(前のページからの続き)

```
[ipv4]
method=auto

[ipv6]
addr-gen-mode=stable-privacy
method=auto
```

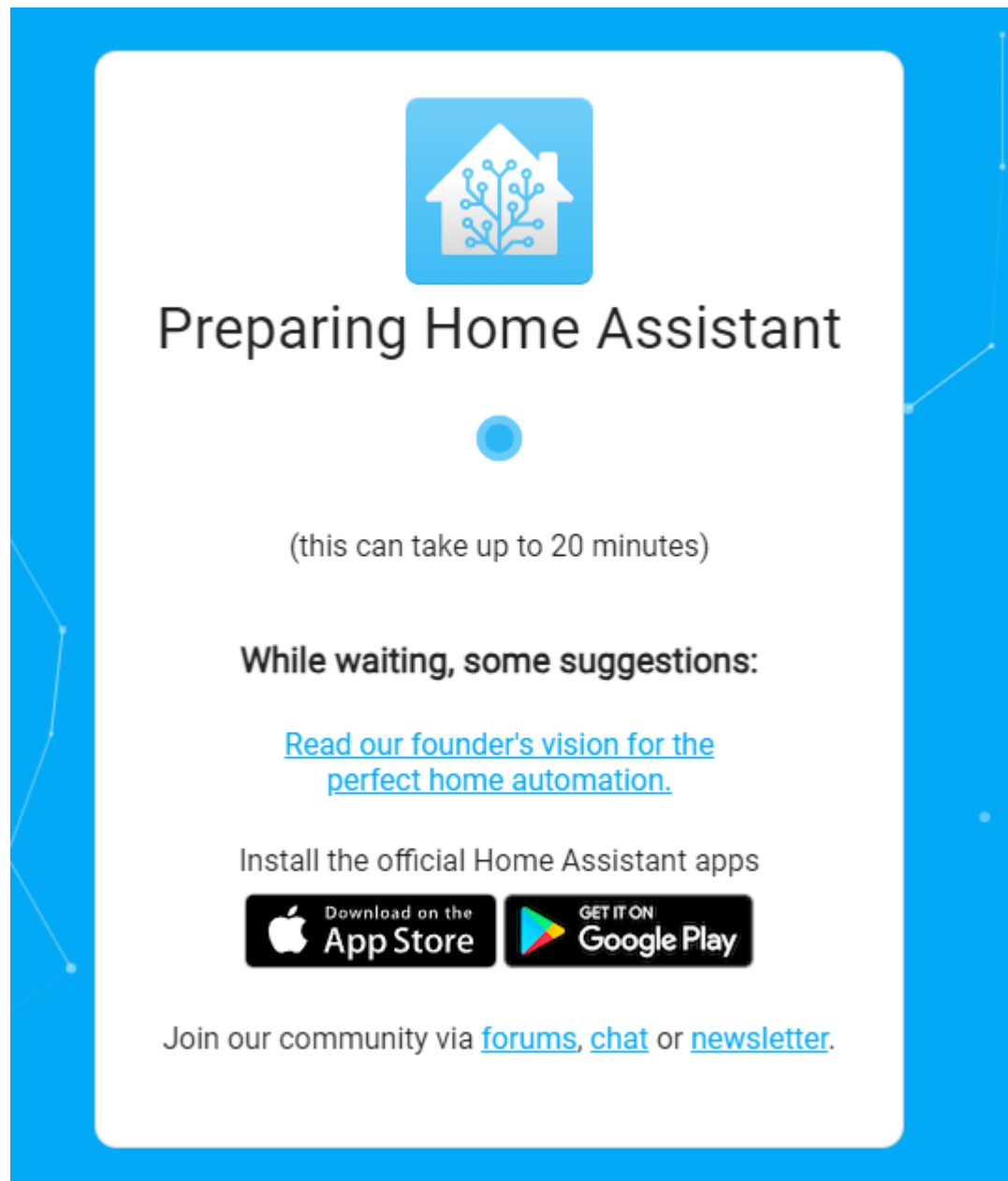
ファイルを保存して閉じます。

#### ステップ 10

microSD カードをコンピュータから取り出し、Raspberry Pi に挿入します。その後、電源（および必要な場合はイーサネットケーブル）を接続します。

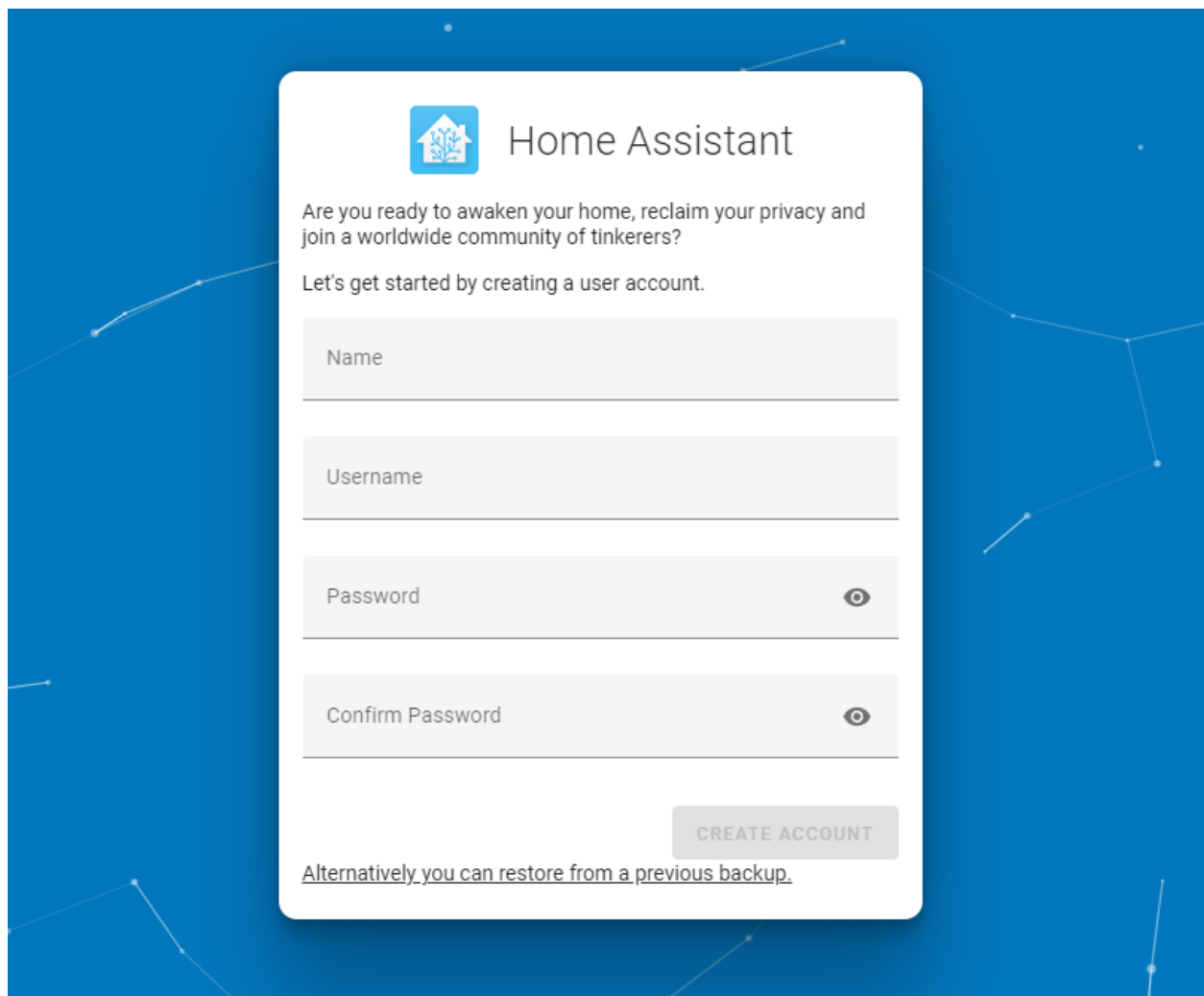
コンピュータに戻って `homeassistant.local:8123` に移動します。それが機能しない場合は、ルーターで IP アドレスを確認できます。

Home Assistant を初めて使用する際、初期設定を実行するためにしばらく待つ必要がある場合があります。



## ステップ 11

次に、最初のアカウントの作成を促されます。

The image shows the Home Assistant setup interface. It features a blue background with a white central card. At the top of the card is the Home Assistant logo (a house with a plant) and the text "Home Assistant". Below this, a message asks if the user is ready to awaken their home and join a community of tinkerers. It then prompts the user to create a user account. There are four input fields: "Name", "Username", "Password", and "Confirm Password". The "Password" and "Confirm Password" fields have eye icons to toggle visibility. A "CREATE ACCOUNT" button is located below the "Confirm Password" field. At the bottom of the card, there is a link that says "Alternatively you can restore from a previous backup.".

Home Assistant

Are you ready to awaken your home, reclaim your privacy and join a worldwide community of tinkerers?

Let's get started by creating a user account.

Name

Username

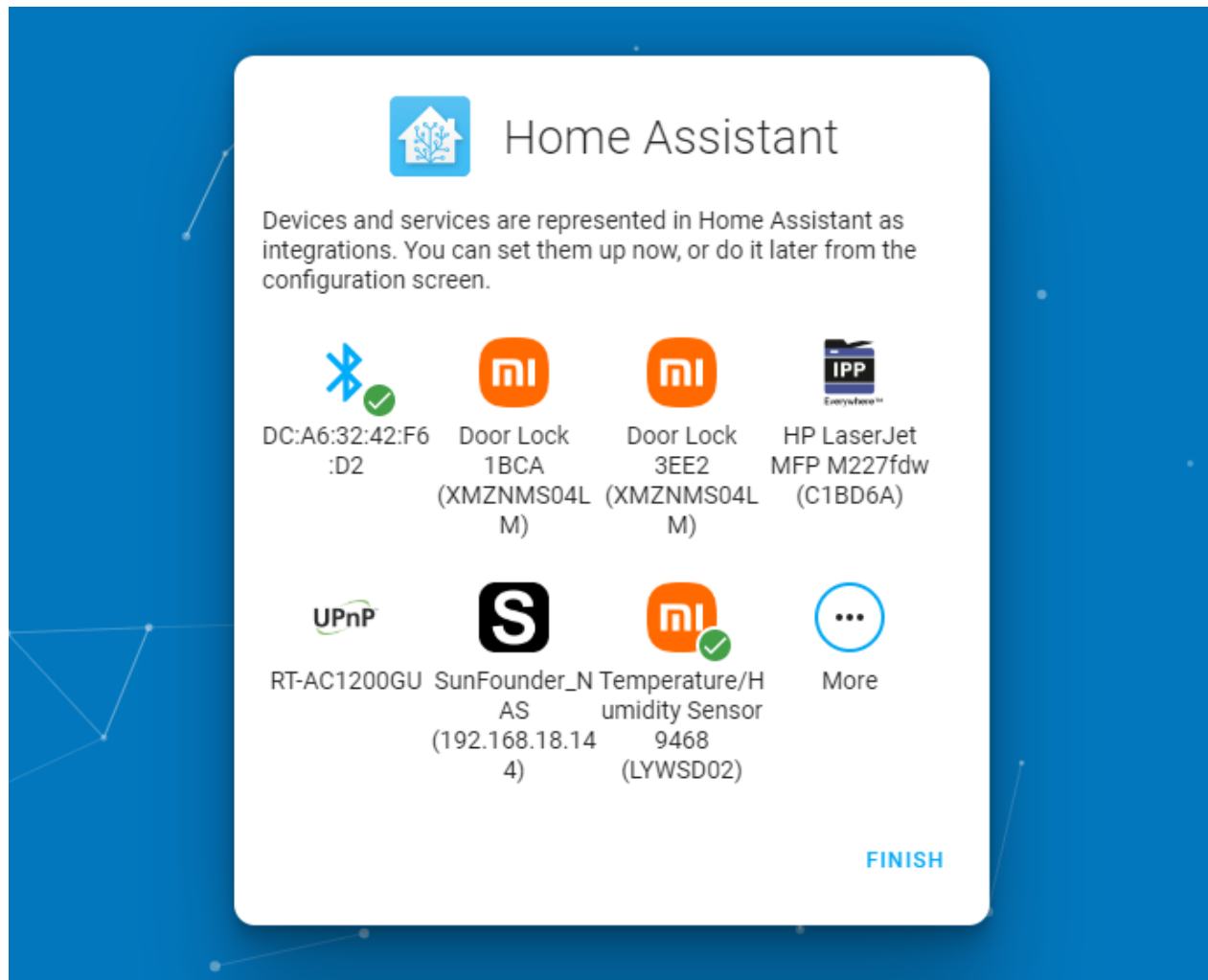
Password

Confirm Password

CREATE ACCOUNT

[Alternatively you can restore from a previous backup.](#)

システムは、検出されたデバイスのインストールを促しますが、今のところ、FINISH をクリックしてこれをスキップできます。



これで、Home Assistant のセットアップが完了しました。

注釈: すでに Home Assistant をお持ちの場合、この部分は無視してください。

Home Assistant での PiPower-Pro カードの設定:

## 4.2 Home Assistant に PiPower Pro を追加する

### ステップ 1

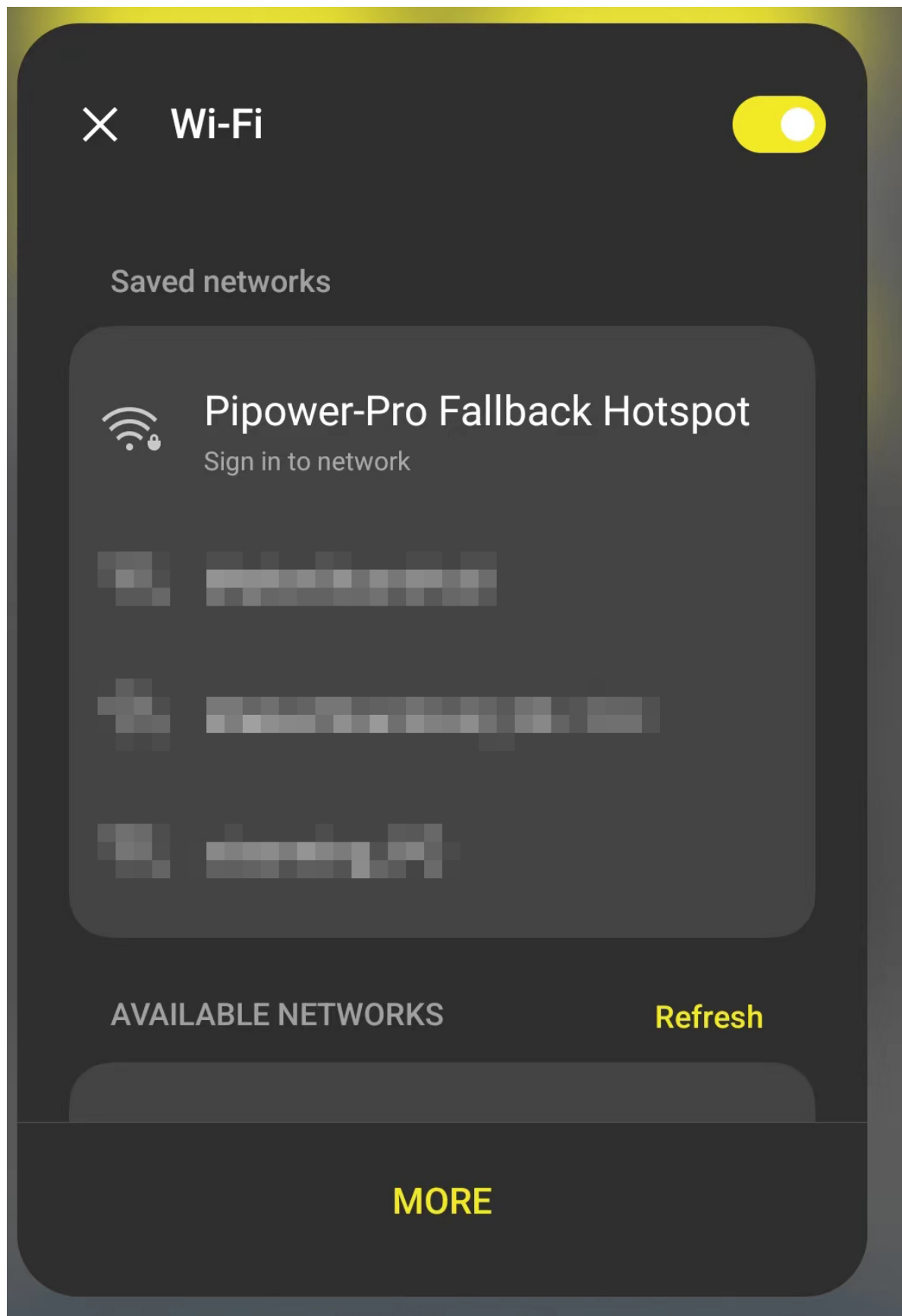
バッテリーを取り付けます。

### ステップ 2

全てのバッテリーインジケータが点灯するまで USB-C 充電器を接続します (これはバッテリーが完全に充電されていることを意味します)。電源ボタンを押して電源を入れます。

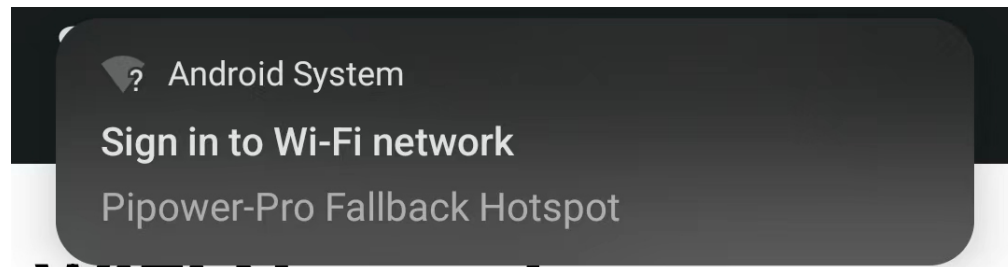
### ステップ 3

PiPower Pro のネットワークを設定します。携帯電話（または他のデバイス）で Wi-Fi を検索し、PiPower Pro Fallback Hotspot に接続します。パスワードは 12345678 です。



#### ステップ 4

接続が完了すると、携帯電話に設定ページが表示されます。ここで PiPower の Wi-Fi 設定を完了します。



# WiFi Networks: pipower-pro



## WiFi Settings

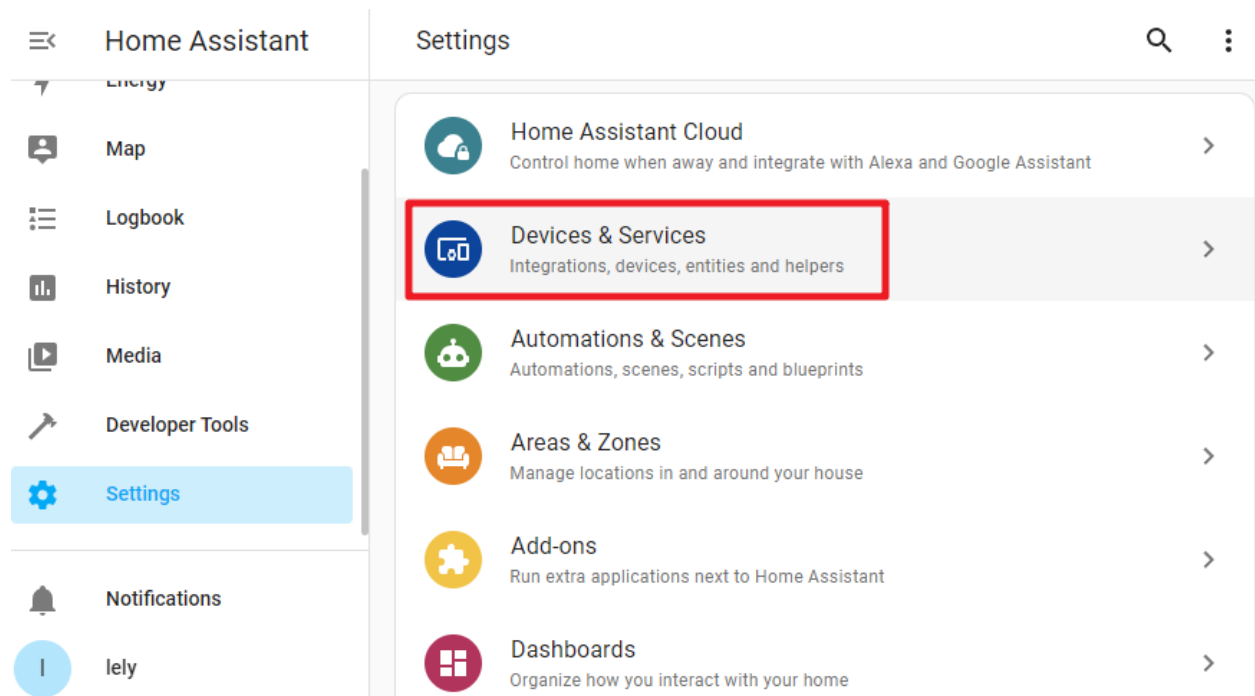
SSID
Password



設定ページが自動的に表示されない場合は、ブラウザを開いて **pipower-pro.local** を訪問してください。

#### ステップ 5

Home Assistant ページを開き、左のサイドバーから設定を選択し、デバイスとサービスを選択します。



#### ステップ 6


右下の + ADD INTEGRATION をクリックします。



#### ステップ 7

ESPHome を選択します。

Select brand ×

 Search for a brand name  
ESPHome ×

 ESPHome >

#### ステップ 8

pipower-pro.local を入力し、送信します。

ESPHome ? ×

Please enter connection settings of your [ESPHome](#) node.

Host\*  
pipower-pro.local

Port  
6053

SUBMIT

#### ステップ 9

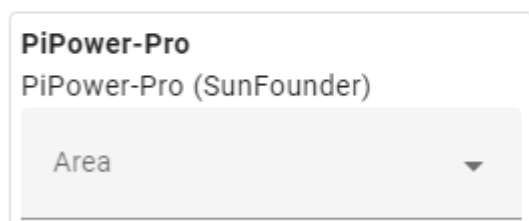
エリアを選択し、設定を完了します。

# Success!



Created configuration for PiPower-Pro.

We found the following devices:



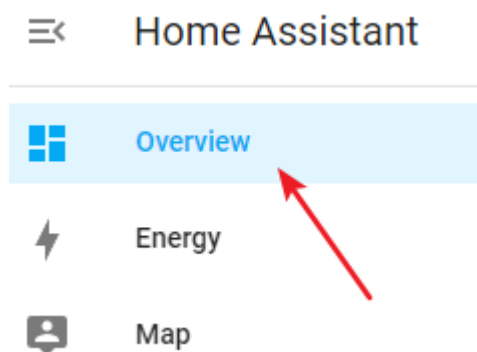
**FINISH**

## ステップ 10

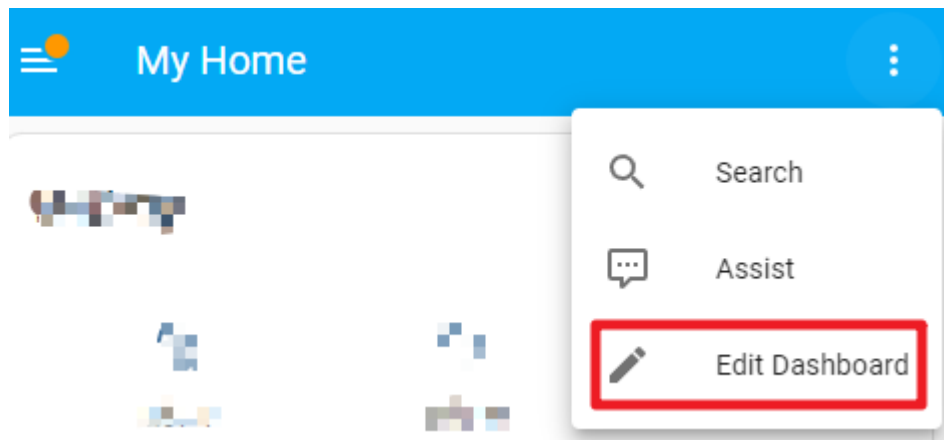
PiPower Pro を正常に追加しました。ダッシュボード上で必要な PiPower Pro の設定を追加することができます。

## 4.3 ダッシュボードの設定

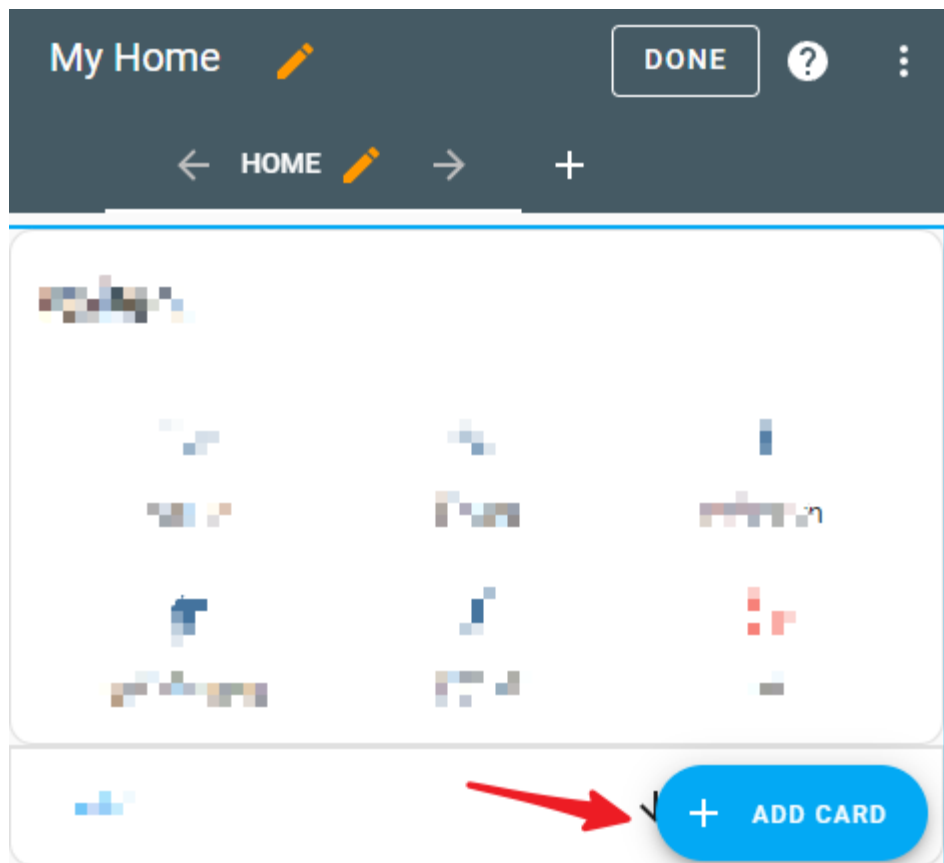
1. Home Assistant のページで、左のサイドバーにある **Overview** をクリックし、コントロールページに移動します。



2. メニューボタンをクリックして、**Edit Dashboard** を選択します。



3. 右下にあるボタンをクリックしてカードを追加し、オプションから希望のカードを選択し、必要に応じて設定を行い、保存します。



## 4.4 コードエディターでカードを追加

1. カードを追加した後、カードの yaml ファイルを手動で編集できます。カード編集ページで **SHOW CODE EDITOR** をクリックしてください。

× Alarm Panel Card Configuration ?

Entity\*  
No matching entities found

Name
Theme (optional)

Available States

- ☒ Arm home
- ☒ Arm away
- ☐ Arm night
- ☐ Arm vacation
- ☐ Custom bypass

[SHOW CODE EDITOR](#) [CANCEL](#) [SAVE](#)

**Invalid configuration**

```

type: alarm-panel
states:
  - arm_home
  - arm_away
entity: ''

```

2. その後、yaml ファイルを直接変更してください。私たちはいくつかの便利な PiPower Pro の設定を提供しています。以下の yaml コードをそのままボックスにコピーしてください。

× Alarm Panel Card Configuration ?

```

1 type: alarm-panel
2 states:
3   - arm_home
4   - arm_away
5 entity: ''
6

```

**Invalid configuration**

```

type: alarm-panel
states:
  - arm_home
  - arm_away
entity: ''

```

[SHOW VISUAL EDITOR](#) [CANCEL](#) [SAVE](#)

## × Vertical Stack Card Configuration



```
1 type: vertical-stack
2 cards:
3   - type: entities
4     entities:
5       - entity: switch.pipower_pro_output_switch
6       - entity: sensor.pipower_pro_output_source
7       - entity: binary_sensor.pipower_pro_external_power
8       - entity: sensor.pipower_pro_battery_voltage
9       - entity: sensor.pipower_pro_output_voltage
10    title: PiPower Pro
11    show_header_toggle: true
12    state_color: true
13  - square: true
14    type: grid
15    cards:
16      - type: gauge
17        entity: sensor.pipower_pro_battery_current
18        min: -2
19        max: 2
20        severity:
21          green: 0
22          yellow: 2
23          red: 2
24        needle: true
25        name: Battery Current
26      - type: gauge
27        entity: sensor.pipower_pro_output_current
28        min: 0
29        max: 3
30        severity:
31          green: 0
32          yellow: 2
33          red: 2.5
34        needle: true
35        name: Output Current
```

[SHOW VISUAL EDITOR](#) [CANCEL](#) [SAVE](#)

### PiPower Pro

☒

☒

PiPower-Pro Output Switch ☒

PiPower-Pro Output Source Battery

PiPower-Pro External Power Plugged in

PiPower-Pro Battery Voltage 5.87 V

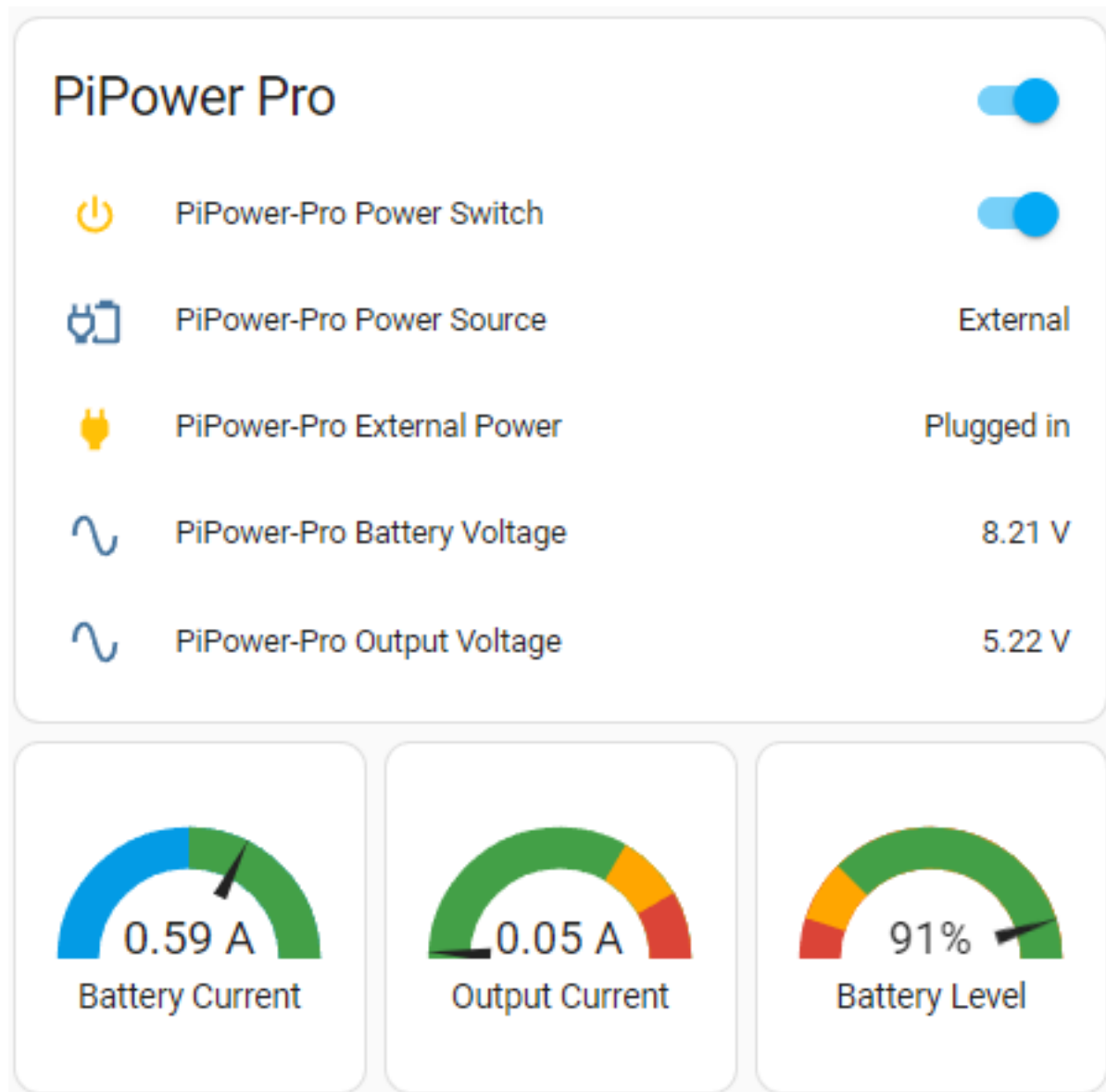
PiPower-Pro Output Voltage 4.11 V

0.00 A  
Battery Current

0.08 A  
Output Current

0%  
Battery Level

簡単な概要



```

type: vertical-stack
cards:
  - type: entities
    entities:
      - entity: switch.pipower_pro_output_switch
      - entity: sensor.pipower_pro_output_source
      - entity: binary_sensor.pipower_pro_external_power
      - entity: sensor.pipower_pro_battery_voltage
      - entity: sensor.pipower_pro_output_voltage
    title: PiPower Pro
    show_header_toggle: true

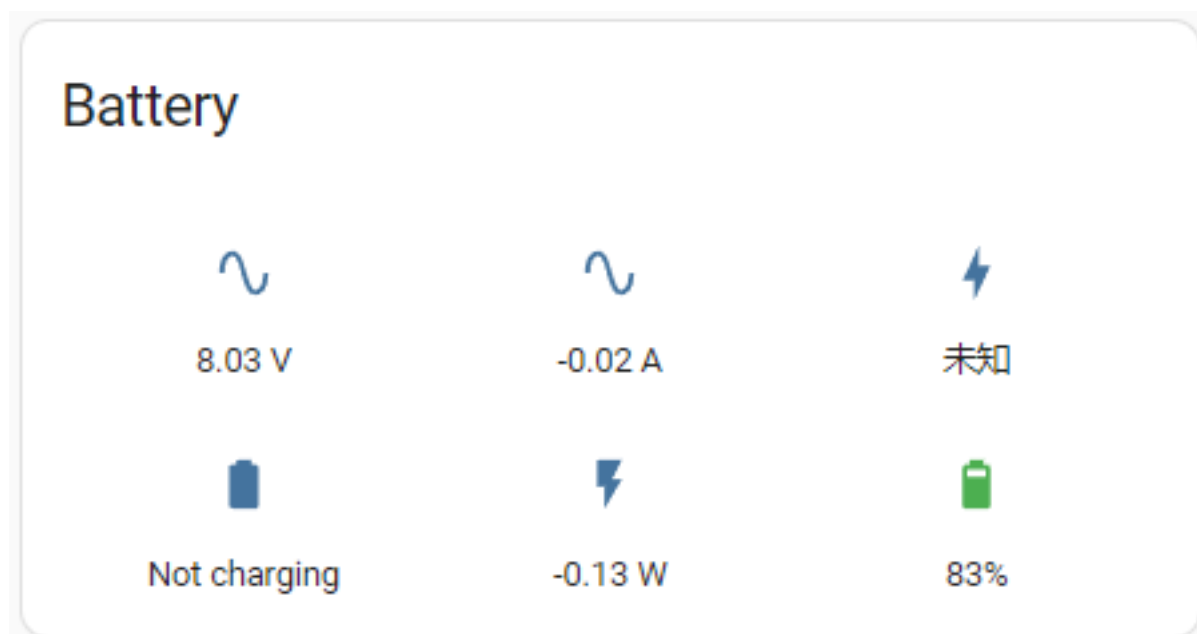
```

(次のページに続く)

```
state_color: true
- square: true
type: grid
cards:
  - type: gauge
    entity: sensor.pipower_pro_battery_current
    min: -2
    max: 2
    severity:
      green: 0
      yellow: 2
      red: 2
    needle: true
    name: Battery Current
  - type: gauge
    entity: sensor.pipower_pro_output_current
    min: 0
    max: 3
    severity:
      green: 0
      yellow: 2
      red: 2.5
    needle: true
    name: Output Current
  - type: gauge
    entity: sensor.pipower_pro_battery_level
    name: Battery Level
    min: 0
    max: 100
    severity:
      green: 25
      yellow: 10
      red: 0
    needle: true
columns: 3
```

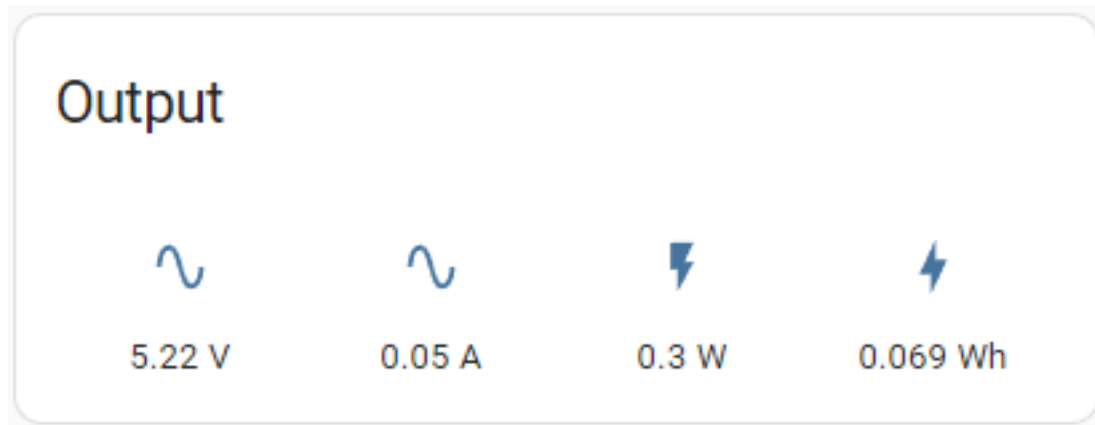
バッテリー情報





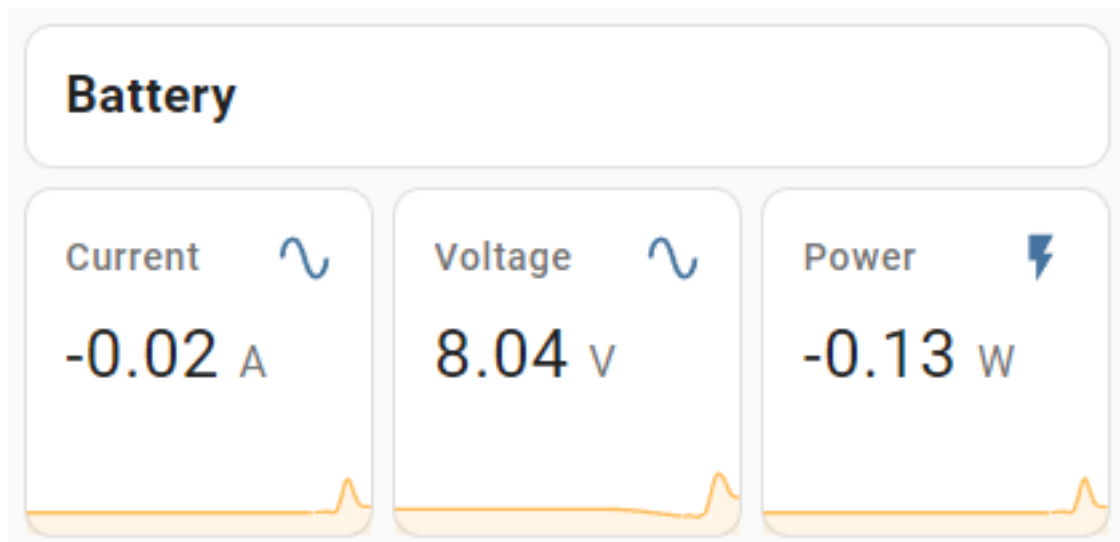
```
show_name: false
show_icon: true
show_state: true
type: glance
entities:
  - entity: sensor.pipower_pro_battery_voltage
  - entity: sensor.pipower_pro_battery_current
  - entity: sensor.pipower_pro_battery_capacity
  - entity: binary_sensor.pipower_pro_is_charging
  - entity: sensor.pipower_pro_battery_power
  - entity: sensor.pipower_pro_battery_level
title: Battery
columns: 3
```

出力情報



```
show_name: false
show_icon: true
show_state: true
type: glance
entities:
  - entity: sensor.pipower_pro_output_voltage
  - entity: sensor.pipower_pro_output_current
  - entity: sensor.pipower_pro_output_power
  - entity: sensor.pipower_pro_output_energy
title: Output
```

#### バッテリーチャート



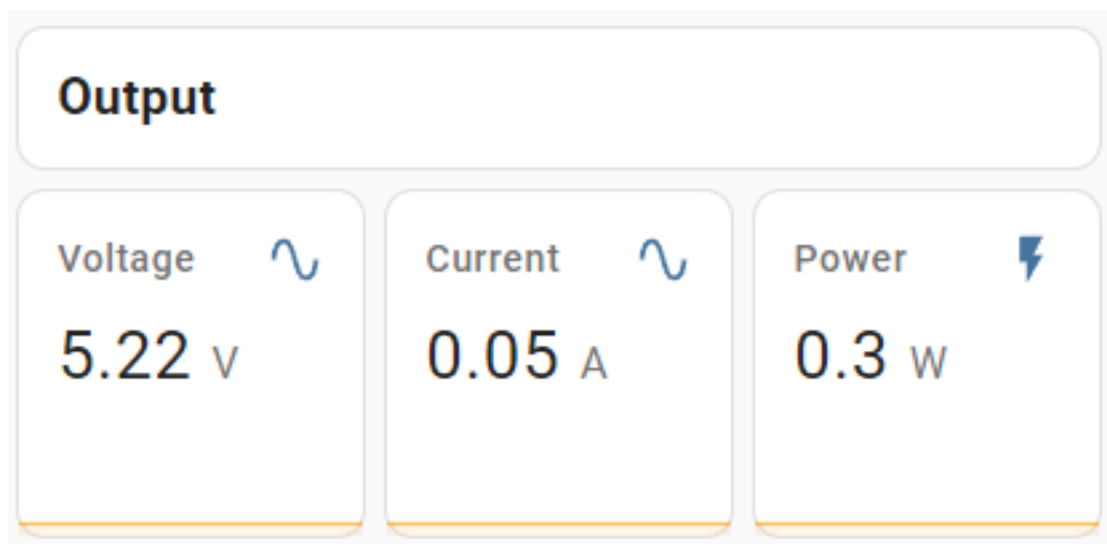
```
type: vertical-stack
cards:
  - type: markdown
```

(次のページに続く)

(前のページからの続き)

```
content: '## Battery'
- square: true
columns: 3
type: grid
cards:
  - hours_to_show: 12
    graph: line
    type: sensor
    entity: sensor.pipower_pro_battery_current
    detail: 2
    name: Current
  - hours_to_show: 12
    graph: line
    type: sensor
    entity: sensor.pipower_pro_battery_voltage
    detail: 2
    name: Voltage
  - hours_to_show: 12
    graph: line
    type: sensor
    entity: sensor.pipower_pro_battery_power
    detail: 2
    name: Power
```










出力チャート



```
type: vertical-stack
cards:
  - type: markdown
    content: '## Output'
  - square: true
    columns: 3
    type: grid
    cards:
      - hours_to_show: 12
        graph: line
        type: sensor
        entity: sensor.pipower_pro_output_voltage
        detail: 2
        name: Voltage
      - hours_to_show: 12
        graph: line
        type: sensor
        entity: sensor.pipower_pro_output_current
        detail: 2
        name: Current
      - hours_to_show: 12
        graph: line
        type: sensor
        entity: sensor.pipower_pro_output_power
        detail: 2
        name: Power
```

設定

## Settings

	PiPower Pro Power Source	Battery
	PiPower Pro External Output	<input type="checkbox"/>
	PiPower Pro Battery Output	<input type="checkbox"/>
	PiPower Pro ESP32 Power	<input checked="" type="checkbox"/>
	PiPower Pro Sub Device Power State	关闭
	PiPower Pro Sub Device Power	<input checked="" type="checkbox"/>
	PiPower Pro Version	v1.0.0
	PiPower Pro Battery Capacity Algorithm... Voltage Map	
	PiPower Pro Factory Reset	按下

type: entities

entities:

- entity: sensor.pipower\_pro\_input\_voltage
- entity: sensor.pipower\_pro\_output\_source
- entity: switch.pipower\_pro\_external\_output
- entity: switch.pipower\_pro\_battery\_output
- entity: switch.pipower\_pro\_esp32\_power
- entity: binary\_sensor.pipower\_pro\_sub\_device\_power\_state
- entity: switch.pipower\_pro\_sub\_device\_power

(次のページに続く)

(前のページからの続き)

```
- entity: sensor.pipower_pro_version
- entity: sensor.pipower_pro_battery_capacity_algorithm
- entity: button.pipower_pro_factory_reset
title: Settings
show_header_toggle: false
state_color: true
```

## 4.5 PiPower Pro エンティティ

Home Assistant に詳しく、カードのカスタマイズを自分で行いたい場合、使用できる PiPower Pro エンティティのリストは以下の通りです。

### 基本情報

- `binary_sensor.pipower_pro_battery_low` - バッテリー低下状態 (bool)
- `binary_sensor.pipower_pro_is_charging` - 充電状態 (V)

### スイッチ

- `switch.pipower_pro_battery_output` - バッテリー出力スイッチ (bool)
- `switch.pipower_pro_esp32_power` - ESP32 電源スイッチ (bool)
- `switch.pipower_pro_external_output` - 外部出力スイッチ (bool)

### 出力

- `sensor.pipower_pro_output_voltage` - 出力電圧 (V)
- `sensor.pipower_pro_output_current` - 出力電流 (A)
- `sensor.pipower_pro_output_power` - 出力電力 (W)
- `sensor.pipower_pro_output_energy` - 出力エネルギー (Wh) 全出力エネルギーの計算に使用される。サービス経由でリセット可能。詳細はすべてのサービスを参照

### バッテリー

- `sensor.pipower_pro_battery_voltage` - バッテリー電圧 (V)
- `sensor.pipower_pro_battery_current` - バッテリー電流、充電の場合は正、放電の場合は負 (A)
- `sensor.pipower_pro_battery_power` - バッテリー出力電力 (W)
- `sensor.pipower_pro_battery_capacity` - バッテリー容量 (mAh)

- `sensor.pipower_pro_battery_level` - バッテリーレベル (%)

#### 入力

- `sensor.pipower_pro_input_voltage` - 外部入力電圧 (V)

#### サブデバイス制御

- `switch.pipower_pro_sub_device_power` - サブデバイス電源制御信号 (bool)
- `binary_sensor.pipower_pro_sub_device_power_state` - サブデバイス電源状態 (bool)

#### その他

- `sensor.pipower_pro_battery_capacity_algorithm` - バッテリー容量アルゴリズム (String)
- `sensor.pipower_pro_power_source` - 現在の出力ソース: バッテリー/外部 (String)
- `sensor.pipower_pro_battery_factory_capacity` - バッテリー工場名義容量 (mAh)
- `binary_sensor.pipower_pro_external_power` - 外部入力状態 (bool)
- `button.pipower_pro_factory_reset` - 工場設定リセットボタン (bool)
- `update.pipower_pro_firmware` - ファームウェア更新
- `switch.pipower_pro_power_switch` - 出力スイッチ (bool)
- `sensor.pipower_pro_version` - PiPower Pro バージョン (String)

#### すべてのサービス

- `set_battery_factory_capacity` - バッテリー工場名義容量の変更 (capacity: int, mAh)、デフォルト 2000
- `enable_coulomb_count_beta` - クーロン計算アルゴリズムの有効化 (enable: bool)、デフォルトは false
- `reset_capacity` - 現在の容量を工場名義容量にリセット
- `reset_output_energy` - 出力エネルギーを 0 にリセット
- `set_edv2` - 放電終了電圧 2 の設定、放電キャリブレーション 2 のための電圧、デフォルト 6.8. クーロンカウンタの詳細を参照
- `set_edv1` - 放電終了電圧 1 の設定、放電キャリブレーション 1 のための電圧、デフォルト 6.5. クーロンカウンタの詳細を参照
- `set_edv0` - 放電終了電圧 0 の設定、放電キャリブレーション 0 のための電圧、デフォルト 6.2. クーロンカウンタの詳細を参照

- `set_rcv` - リセットキャリブレーションステータスのための電圧の設定、デフォルト 8.0. クーロンカウンタの詳細を参照
- `simulate_low_power` - 低電力トリガーシナリオのテストのための低電力シミュレーション

高度な使用方法:

## 4.6 安全シャットダウンの設定

PiPower Pro には、接続デバイスの電源ステータス（以下、サブデバイスと称します）を監視するための 2 つのピンが標準で設定されています。これにより、遠隔での電源オン、オフ、そしてバッテリーが低くなった際の自動安全シャットダウンが可能となります。

---

注釈: HassOS を実行しているホストが PiPower Pro のサブデバイスとして設定されている場合、そのホストがシャットダウンすると機能が失われ、遠隔電源オンは不可能となります。

---

- ピン 42 およびセンサーエンティティ `binary_sensor.pipower_pro_sub_device_power_state` はデバイスの現在の状態を検知します。
- ピン 41 およびエンティティ `switch.pipower_pro_sub_device_power` はサブデバイスの電源を制御します。

例として、PiPower Pro を Raspberry Pi の UPS 電源として利用し、その状態を監視し、外部電源が途絶え、バッテリー残量が低下した場合に安全にシャットダウンさせる方法を見ていきましょう。

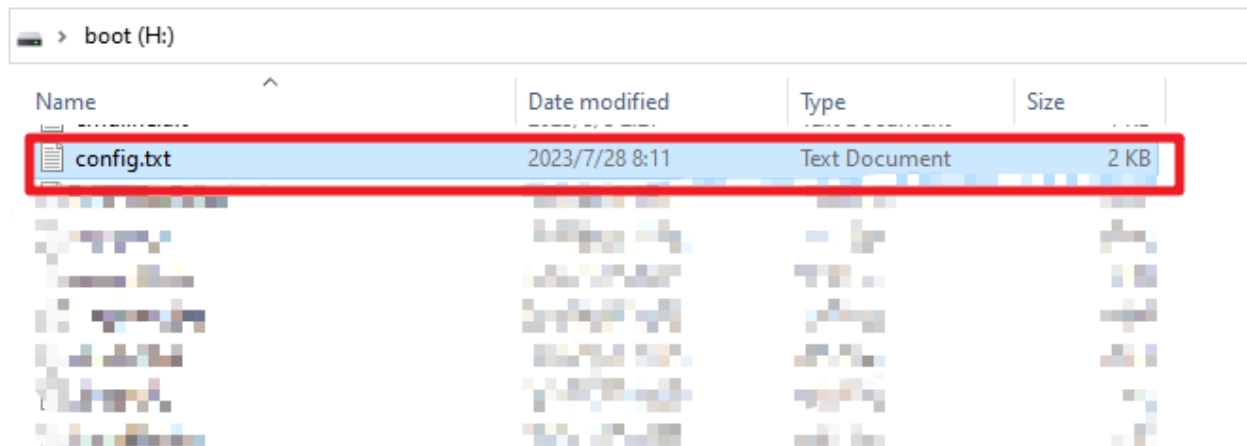
### ステップ 1

Raspberry Pi の設定を行います。

Raspberry Pi の 2 つのピンをそれぞれ **Power Status Signal Pin** と **Shutdown Signal Pin** に設定します。これは `devicetree` を通じて行います。

Raspberry Pi システムの SD カードを PC に挿入し、boot パーティションのルートディレクトリで `config.txt` を探します。





このファイルを開き、[all] の下に次の 2 行を追加します。

```
dtoverlay=gpio-poweroff,gpiopin=17  
dtoverlay=gpio-shutdown,gpio_pin=18
```

- gpio-poweroff は Raspberry Pi の電源のオン/オフ状態を示します。正しく設定すると、Raspberry Pi は電源がオンの際にこのピンをハイにし、オフの際にはローにします。
- gpio-shutdown は Raspberry Pi のシャットダウンを制御する信号です。正しく設定すると、このピンをローにすることで Raspberry Pi がシャットダウンを開始します。

## ステップ 2

- PiPower Pro のピン 42 を Raspberry Pi の gpio-poweroff ピン（こちらではピン 17）に接続します。
- PiPower Pro のピン 41 を Raspberry Pi の gpio-shutdown ピン（こちらではピン 18）に接続します。

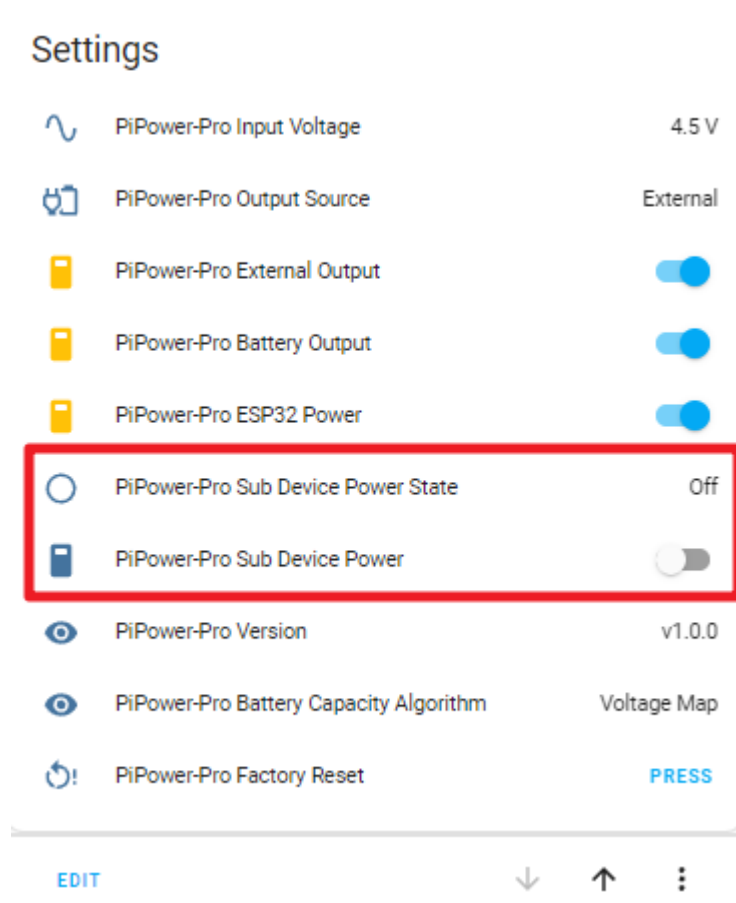
## ステップ 3

上記の 2 つの信号が正常に動作するか確認します。

ダッシュボードに以下の 2 つのエンティティを追加します：

- binary\_sensor.pipower\_pro\_sub\_device\_power\_state
- switch.pipower\_pro\_sub\_device\_power

**Settings** カードを追加する際（カードの追加方法は [コードエディターでカードを追加](#) を参照）、これらの 2 つのエンティティがそれぞれ PiPower-Pro Sub Device Power State および PiPower-Pro Sub Device Power として表示されます。



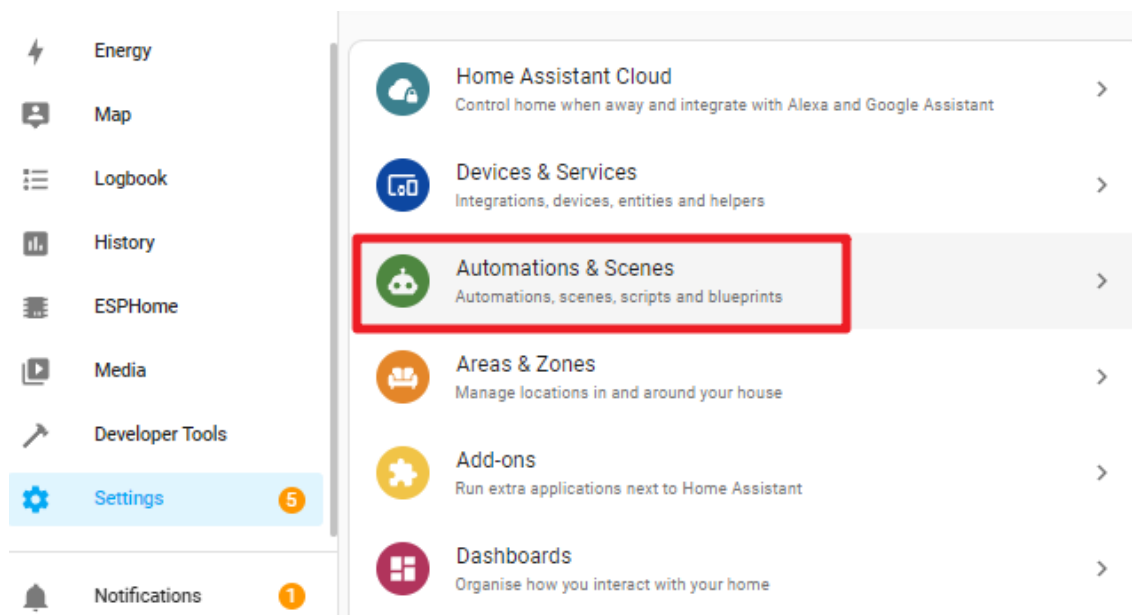
前者を使用して Raspberry Pi の動作状態を確認し、後者を使用して Raspberry Pi の電源をオフにすることができます。

注釈: PiPower-Pro サブデバイス電源 は Raspberry Pi をオフにするだけです。再度オンにするためには、Raspberry Pi に電源を供給する必要があります (例: **PiPower Pro** カードのメインスイッチをオンにする)。

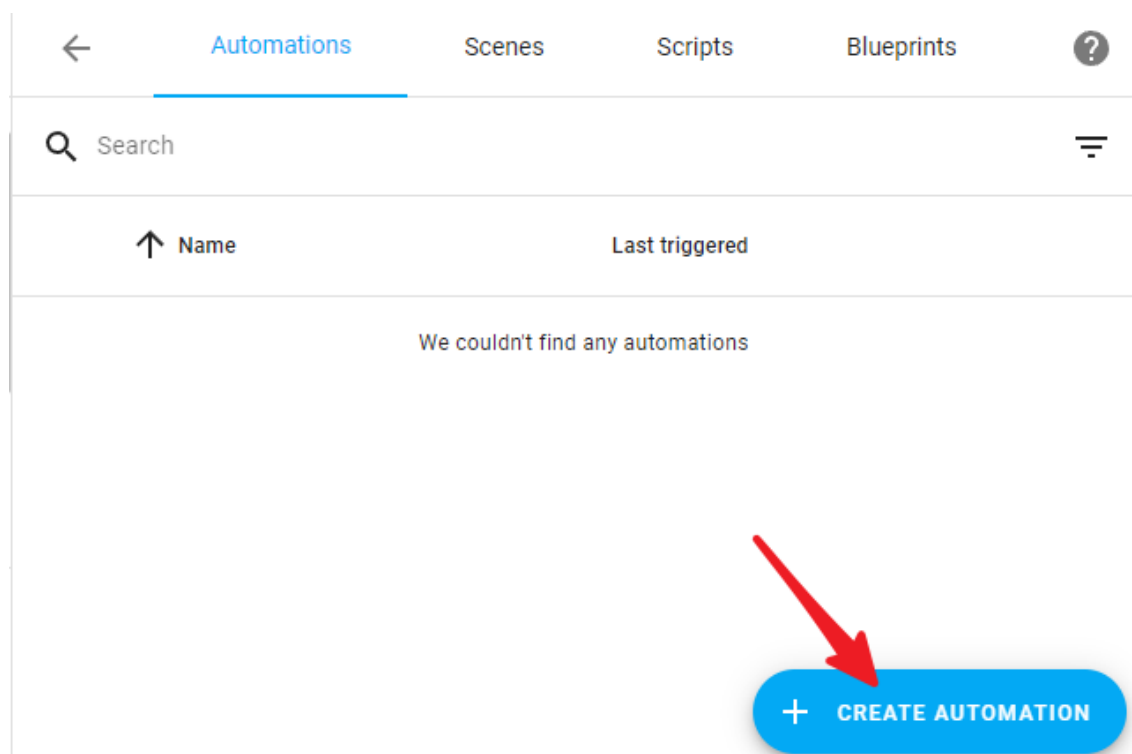
### ステップ 4

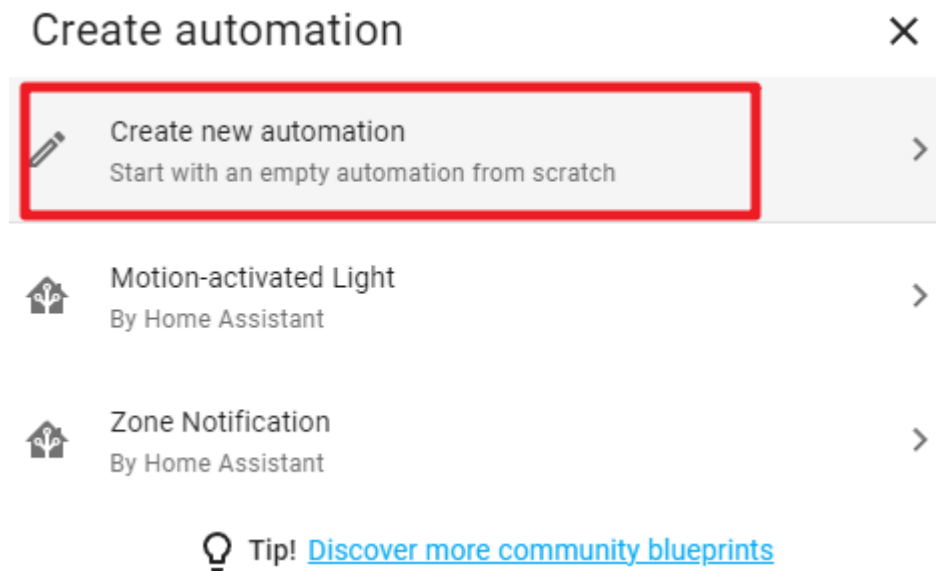
次に、PiPower Pro が Raspberry Pi を安全にシャットダウンできるように自動化を設定します：

1. Home Assistant の設定ページを開き、左側のサイドバーの「Settings」をクリックし、「Automations」を選択します。

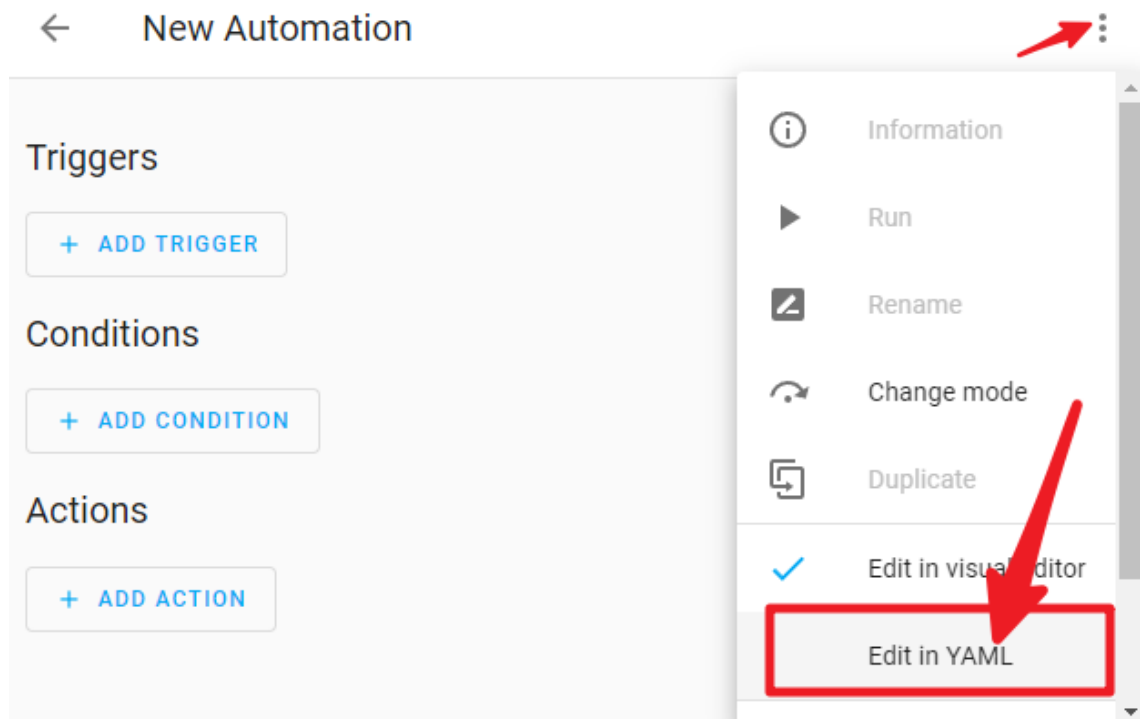


2. 新しい自動化を作成します。





3. 「Edit in YAML」をクリックします。



4. 既存のコードを以下のコードに置き換えます。

```
alias: Safe shutdown RPi
description: Turn off Raspberry Pi if no external power plug in and battery low
trigger:
  - platform: state
```

(次のページに続く)

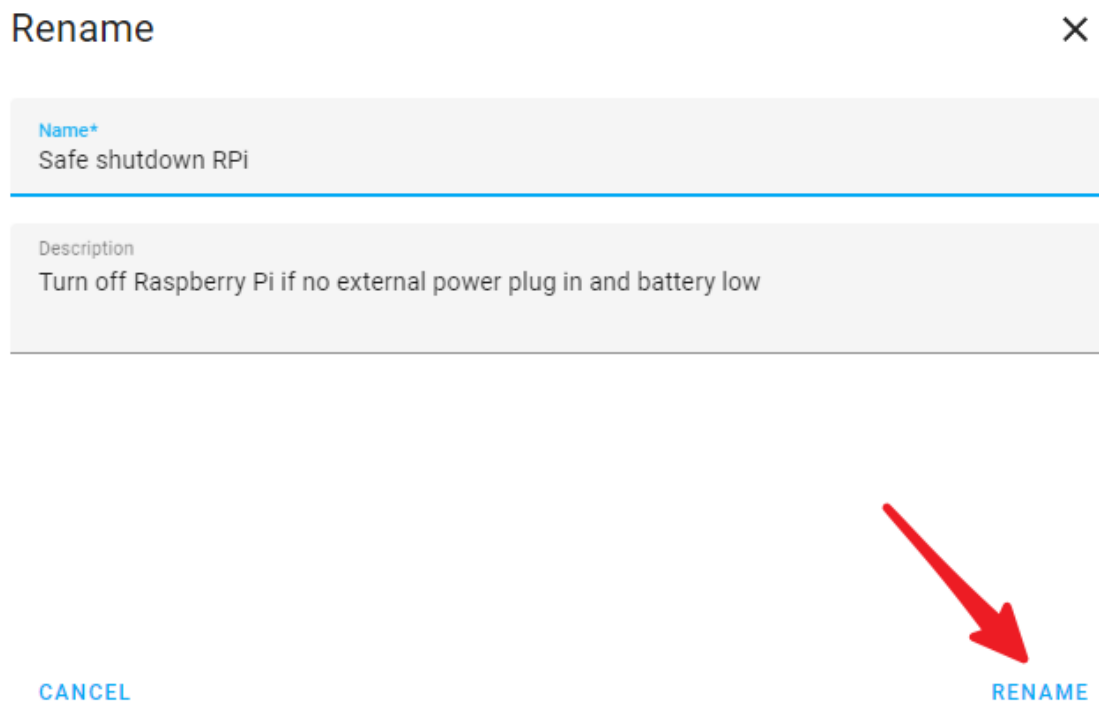
(前のページからの続き)

```
entity_id:
  - binary_sensor.pipower_pro_external_power
from: "on"
to: "off"
- platform: numeric_state
  entity_id: sensor.pipower_pro_a03846_battery_level
  below: 25
condition:
  - condition: and
    conditions:
      - condition: state
        entity_id: binary_sensor.pipower_pro_a03846_external_power
        state: "off"
  - condition: and
    conditions:
      - condition: state
        entity_id: switch.pipower_pro_sub_device_power
        state: "on"
action:
  - type: turn_off
    device_id: a0ee4e356c85c4f69f765ed72baad129
    entity_id: switch.pipower_pro_sub_device_power
    domain: switch
mode: single
```

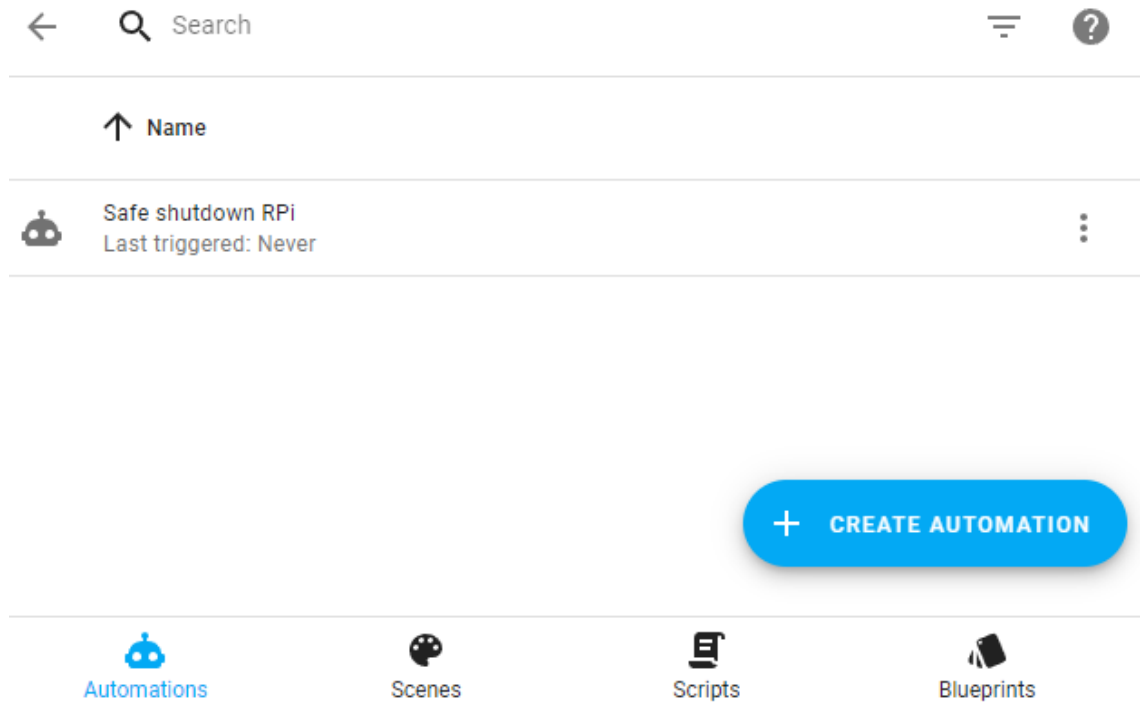
5. 「Save」をクリックします。



6. 「rename」をクリックします。



7. 1 つ前のレベルに戻ります。新しく設定された自動化が表示されるはずです。



注釈: さらにいくつかの自動化を作成する必要があります。それらを完了するために前のステップを参照してください。

#### 電力を節約

```
alias: Save Power
description: Turn off if raspberry pi power off
trigger:
  - platform: state
    entity_id:
      - binary_sensor.pipower_pro_sub_device_power_state
    from: "on"
    to: "off"
condition:
  - condition: state
    entity_id: switch.pipower_pro_sub_device_power
    state: "off"
action:
  - delay:
    hours: 0
    minutes: 0
```

(次のページに続く)

(前のページからの続き)

```
seconds: 2
milliseconds: 0
- type: turn_off
  device_id: a0ee4e356c85c4f69f765ed72baad129
  entity_id: switch.pipower_pro_a03846_power_switch
  domain: switch
- type: turn_off
  device_id: a0ee4e356c85c4f69f765ed72baad129
  entity_id: switch.pipower_pro_a03846_esp32_power
  domain: switch
mode: single
```

### RPi 電源オフ同期

```
alias: Sync Power Off RPi
description: Power Off Raspberry Pi is Switch Off
trigger:
  - platform: state
    entity_id:
      - switch.pipower_pro_a03846_power_switch
    from: "on"
    to: "off"
condition: []
action:
  - type: turn_off
    device_id: a0ee4e356c85c4f69f765ed72baad129
    entity_id: switch.pipower_pro_sub_device_power
    domain: switch
mode: single
```

### RPi 電源オン同期

```
alias: Sync Power On RPi
description: Power On Raspberry Pi is Switch On
trigger:
  - platform: state
    entity_id:
      - switch.pipower_pro_a03846_power_switch
    from: "off"
```

(次のページに続く)



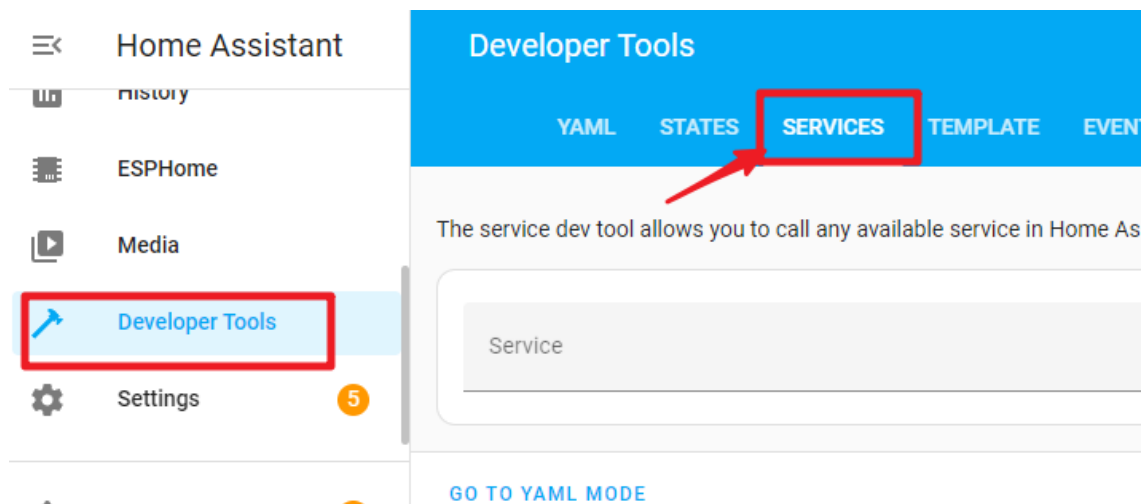
(前のページからの続き)

```
to: "on"
condition: []
action:
  - type: turn_on
    device_id: a0ee4e356c85c4f69f765ed72baad129
    entity_id: switch.pipower_pro_sub_device_power
    domain: switch
mode: single
```

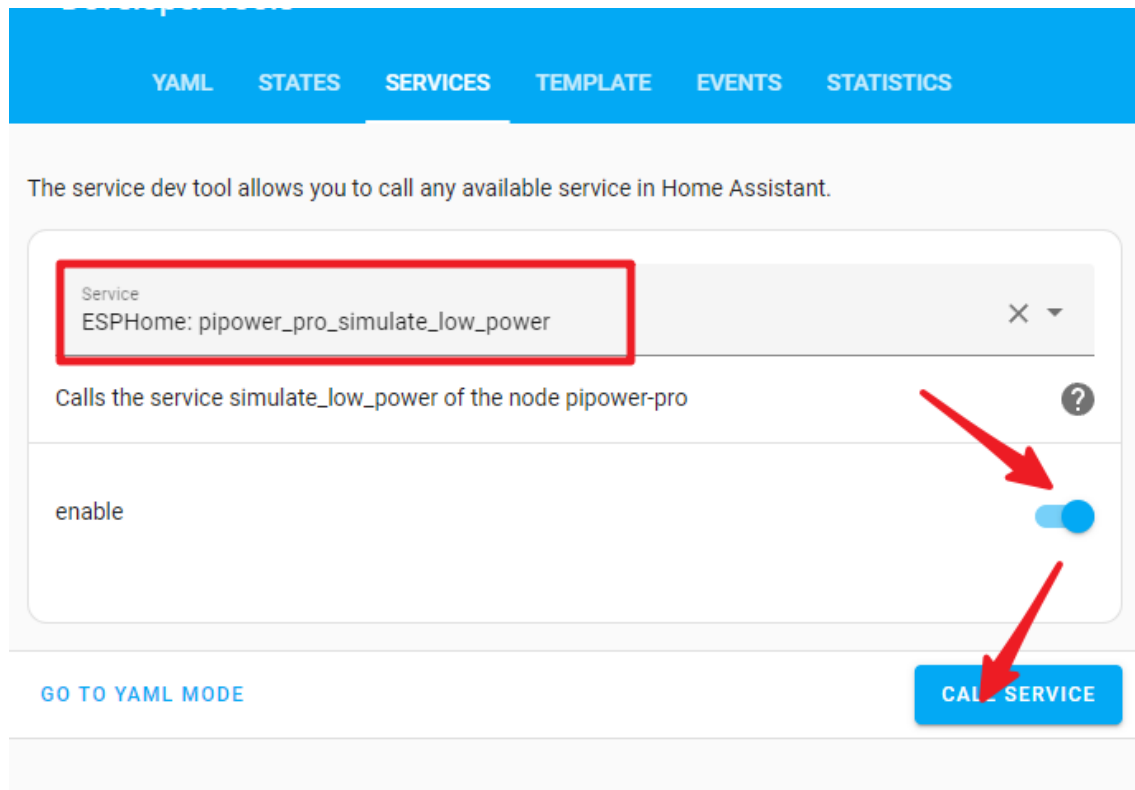
## ステップ 5

低電力状態のシミュレーションを使用してテストをトリガーします：

1. Developer Tools の SERVICES インターフェースを開きます。



2. ESPHome: `pipower_pro_simulate_low_power` を見つけて有効にし、「Call Service」ボタンをクリックします。



PiPower のバッテリーライトが消え、概要でバッテリーレベルが 10% に低下するのを確認できます。

Raspberry Pi はシャットダウンし、シャットダウンが完了すると 2 秒後に PiPower Pro の電源が切れ、PWR ライトが消えます。

## 4.7 クーロンカウンタ (ベータ版)

クーロンカウンタのアルゴリズムはバッテリー容量の計算の精度を向上させることができますが、現在はベータ段階にあり、大きな誤差が生じる可能性があります。十分な注意を払って使用してください。

クーロンカウンタを有効にする

1. Home Assistant のページに移動し、左のサイドバーで「Developer Tools」をクリックします。
2. Developer Tools のページで、「Services」タブを選択します。
3. サービスのリストから、ESPHome: pipower\_pro\_enable\_coulomb\_count\_beta を選択します。
4. enable\_coulomb\_count\_beta のスイッチをオンにします。
5. 下の「Call Service」ボタンをクリックします。
6. エンティティ sensor.pipower\_pro\_battery\_capacity\_algorithm で現在選択されているバッテリー容量のアルゴリズムを確認できます。

## アルゴリズム

クーロンカウンタのアルゴリズムは、毎秒バッテリーの電流と電圧の測定を積分してエネルギーを計算します。

$\text{Capacity} += \text{Voltage} * \text{Current}$

## マッチング

この積分によって計算される容量は、現在の時点からの充放電エネルギーのみです。実際のバッテリー容量と関連付けるためのマッチングプロセスが必要です。ここでのマッチング方法はシンプルです。PiPower Pro のデフォルトのバッテリー容量はバッテリーの名目容量、すなわち 2000mAh です。実際のバッテリー容量はこの値よりも小さいでしょう。バッテリーが充電される限り、容量は最大 2000mAh ( `set_battery_factory_capacity` のサービスを使用して変更可能 ) に設定されるため、バッテリーが完全に充電されたとき、容量値は実際のバッテリー容量の 2000mAh と一致し、積分計算値も実際のバッテリー容量値と一致します。

## 自動キャリブレーション

積分は誤差を蓄積する可能性があり、バッテリーが時間とともに使用されるとバッテリー容量は減少し、名目 2000mAh 容量に達しないかもしれません。したがって、バッテリー容量をキャリブレートするためのいくつかのキャリブレーション方法を使用する必要があります。

ここでは、補償放電終了電圧 (CEDV) キャリブレーション方法を使用します。CEDV キャリブレーション方法の原理は、放電終了時の電圧が比較的正確であり、この時の電圧曲線も最も急であることです。この電圧をキャリブレーションポイントとして使用するのには適切です。したがって、3 つの EDV ポイントを設定します : `edv2 ( 7 % )`、`edv1 ( 3 % )` および `edv0 ( 0 % )`。

これら 3 つのキャリブレーション電圧を設定した後、バッテリーがこれら 3 つのポイントに放電されたとき、PiPower Pro はバッテリーをキャリブレートします :  $\text{MaxCapacity} = \text{MaxCapacity} - \text{Capacity} + \text{MaxCapacity} * 7\%$ 。電圧の変動により同じポイントでの無制限のキャリブレーションを避けるため、充電が RCV ( Reset Calibration Voltage、デフォルト 8.0V ) に達する前にキャリブレーションは一度に制限されます。`edv2`、`edv1`、`edv0`、および `rcv` は Service サービスで設定できます。詳細は [PiPower Pro エンティティ](#) を参照してください。

## インジケータ

クーロンカウンタアルゴリズムが有効になると、バッテリーインジケータもクーロンカウンタモードに切り替わります。ただし、バッテリーレベルの読み取りが正しくない、またはバッテリーレベルがリセットされる可能性があります。

バッテリーインジケータと電力との関係は次のとおりです :

- 4 つの LED すべてが点灯 : 75 %
- 3 つの LED が点灯 : 50 %
- 2 つの LED が点灯 : 25 %
- 1 つの LED が点灯 : 10 %

- 4 つの LED すべてが消灯 : 0 %、バッテリーの充電が必要です。

## 4.8 カスタム開発

PiPower Pro の基本機能がご自身のニーズに十分でない場合、PiPower Pro 上でカスタム開発を行うことができます。

PiPower Pro のすべてのソフトウェアはオープンソースです。以下は、カスタム開発のための基本的なチュートリアルと準備です。

1 Home Assistant の開発者モードを開く。

- Home Assistant の管理ページを開きます。
- 左下の「Configuration」を選択します。

2 ESPHome をインストール。

- Home Assistant の管理ページを開きます。
- 左下の「Configuration」を選択します。
- 「Add-ons」を選択します。
- 「Add」ボタンをクリックします。
- "esphome"を検索します。
- 「Install」をクリックします。
- インストール後、「Start」をクリックします。
- 「Add to Sidebar」を選択します。

3 新しいデバイスを作成。

- サイドバーの「ESPhome」をクリックし、ESPHome 管理ページに入ります。
- 「New Device」を選択します。
- デバイス名を入力します、例 : "PiPower Pro"。
- 初回設定では、Wi-Fi のアカウントとパスワードも入力する必要があります。
- 「ESP32 S2」を選択します。
- 確認してインストールをスキップします。

4 新しいデバイスを設定。

- 作成したばかりのデバイスを選択し、「Edit」をクリックして YAML 編集ページに入ります。

b. 下部に PiPower Pro のテンプレートを追加します：

```
packages:
  remote_package: github://sunfounder/pipower-pro/pipower-pro-template.
  ↪yaml@main
```

c. 右上の「Install」をクリックして、PiPower Pro にインストールします。

4.9 複数の PiPower Pro ユニット

もし同じ Home Assistant 環境内で複数の PiPower Pro ユニットを使用する場合、YAML の設定を変更する必要があります。"esphome"の下に name\_add\_mac\_suffix: true を追加してください。

```
esphome:
  name: pipower-pro
  friendly_name: PiPower-Pro
  name_add_mac_suffix: true
```

4.10 IO 拡張

J4 は拡張用に使用されます。この IO は ESP32 S2 から提供されています。

表 1 IO 拡張

機能	ピン	ピン	機能
5V	5V	3V3	3V3
ADC,Touch,GPIO8	8	GND	Ground
ADC,Touch,GPIO9	9	10	GPIO10,Touch,ADC
ADC,DAC,GPIO18	18	36	GPIO36
GPIO37	37	38	GPIO38
GPIO39	39	40	GPIO40
GPIO41	41	42	GPIO42



## 第 5 章

# FAQ

### 5.1 PiPower Pro が動作しないのはなぜですか？

初めてバッテリーを入れるときや、バッテリーを外して再度取り付けるとき、バッテリーが正常に動作しない場合があります。

これは、バッテリーが取り外されたとき、オンボードの過放電保護回路のメカニズムにより、電圧が低すぎると判断され、保護回路が作動するためです。

この時、**Type C** ケーブルを充電ポートに差し込み、保護回路をリリースする必要があります。そうすれば、バッテリーは正常に使用できます。

### 5.2 PiPower Pro はどのシングルボードコンピューターで使用できますか？

PiPower Pro と互換性のあるシングルボードコンピューターは以下に示されています。

---

注釈：機能的に互換性があるとは、PiPower Pro で正常に電源供給できることを意味します。

---