

---

# SunFounder PiCar-S

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

2022 年 08 月 02 日



# 目次

第 1 章	PiCar-S について	1
1.1	部品一覧	2
1.2	前書き	10
1.3	車の組み立て	11
1.4	回路構築	24
1.5	ラズベリー パイを始めよう	28
1.6	サーボ構成	39
1.7	補正	45
1.8	車を装備する！	48
1.9	付録	75
1.10	ありがとうございました	85
第 2 章	著作権表示	87





## 第 1 章

# PiCar-S について

**PiCar-S** は、Raspberry Pi 世代 3 モデル B、世代 3 モデル B +、世代 4 モデル B で使用できるクールなスマートカーである。超音波障害物回避、ライトフォロア、ラインフォロアなどの 3 つのセンサーモジュールを搭載しているため、車の制御方法に関するプログラミングを良く学ぶことはできる。

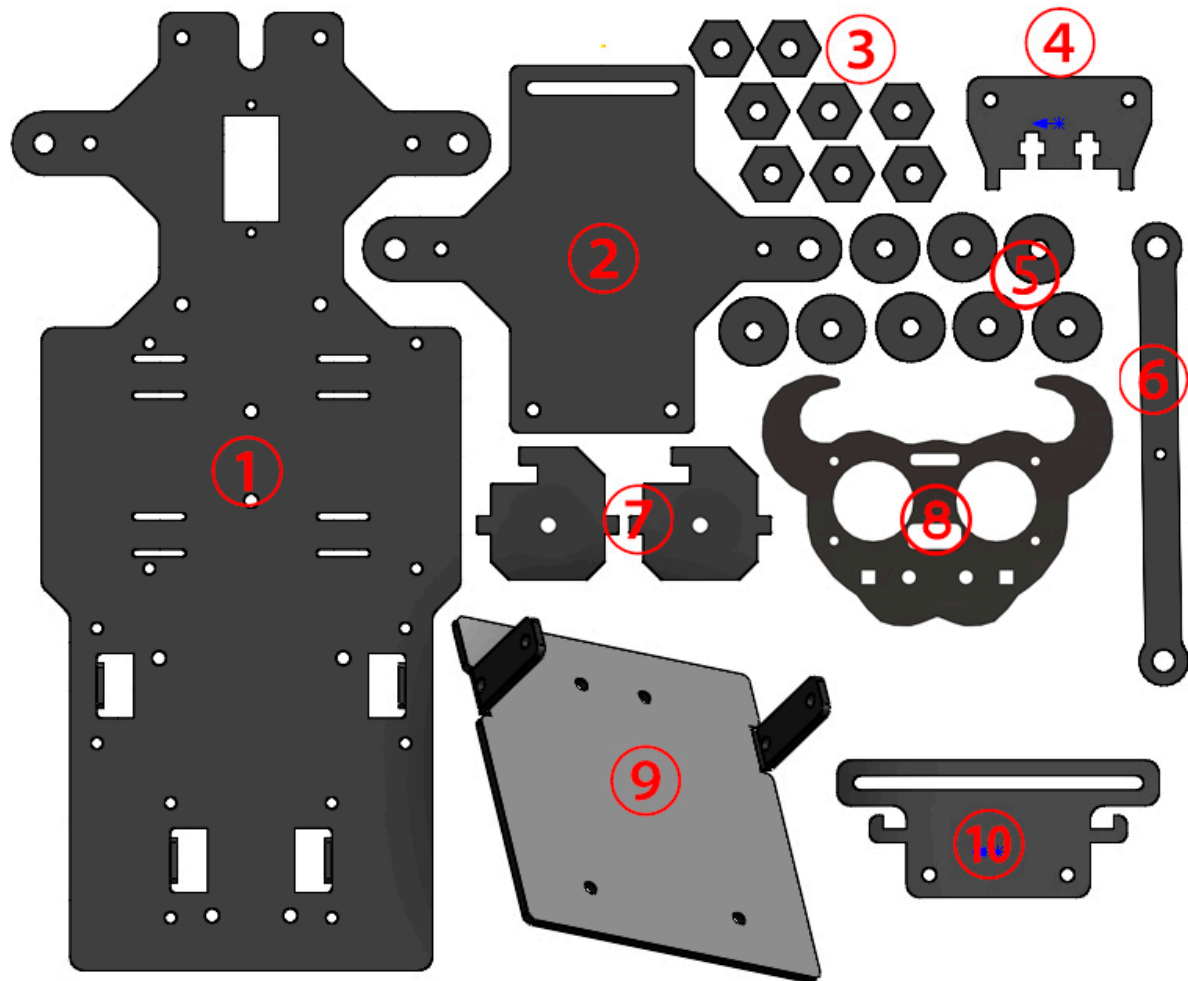
本取扱説明書では、ハードウェアとソフトウェアの両方を考え、説明、物理部品のイラストを使用して PiCar-S を組み立てる方法について説明する。これらのすべてがどのように機能するかを学ぶことを楽しむだろう。次のリンクをクリックすると、PDF ユーザーマニュアルを表示したり、コードのクローンを作成したりできます：[https://github.com/sunfounder/SunFounder\\_PiCar-S/tree/V3.0](https://github.com/sunfounder/SunFounder_PiCar-S/tree/V3.0)。

製品の関するご意見、またはマニュアルにない新しいプロジェクトを学習したい場合は、遠慮なくこちらまでご連絡ください。できるだけ早めにマニュアルに更新します。

サポートメールアドレスはこちらまで：[cs@sunfounder.com](mailto:cs@sunfounder.com)

## 1.1 部品一覧

### 1.1.1 構造プレート



1. 上部プレート x 1
2. 全部ハーフシャーシ x1
3. 六角前輪固定プレート x 8
4. 超音波サポート x1
5. ベアリングシールド x 8
6. ステアリングリンケージ x 1
7. ステアリングコネクター x2
8. 超音波コネクター x1

- 9. 後部ハーフシャーシ x1
- 10. センサーコネクター x1


















### 1.1.2 SunFounder SF006C サーボ x1










- 1. サーボ x1
- 2. 1 アームロッカーアーム x1
- 3. アームロッカーアーム x 1
- 4. 4 アームロッカーアーム x1
- 5. ロッカーアーム固定ネジ x 1
- 6. ロッカーアームねじ



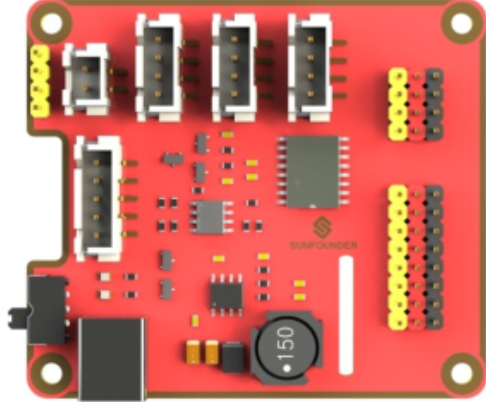
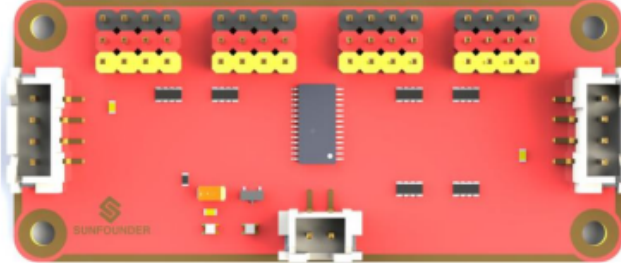
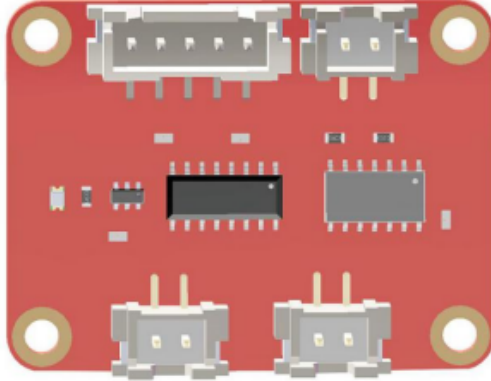
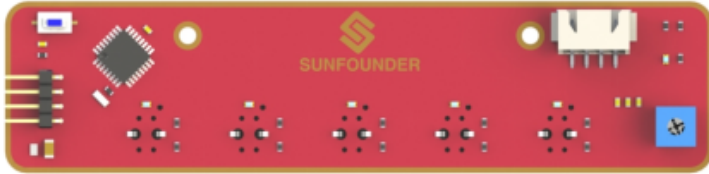

## 1.1.3 メカニカルファスナー

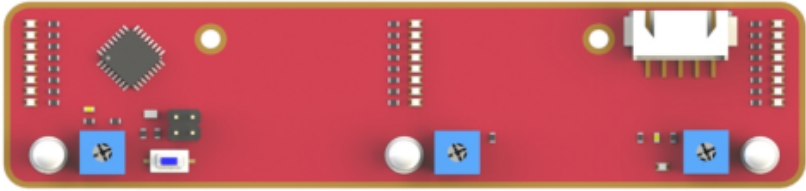
Name	Component	Qty.
M1.4x6 Screw		6
M1.5x4 Self-tapping Screw		3
M2x8 Screw		2
M2.5x6 Screw		4
M2.5x12 Screw		8
M3x8 Screw		8
M3x8 Countersunk Screw		2
M3x10 Screw		9
M3x25 Screw		4
M4x25 Screw		2
M1.4 Nut		6
M2 Nut		2
M2.5 Nut		12
M3 Nut		23
M4 Self-locking Nut		2
M2.5x8 Copper Standoff		8
M3x25 Copper Standoff		8

### 1.1.4 配線

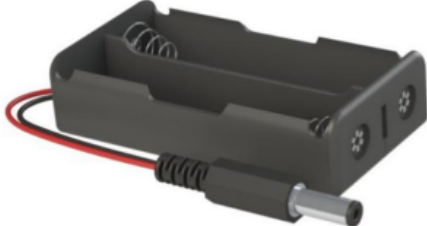
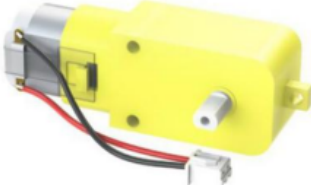



100mm 5-Pin Jumper Wire		1
50mm 4-Pin Jumper Wire		1
50mm 2-Pin Jumper Wire		1
100mm 2-Pin Jumper Wire		1
200mm 5-Pin Jumper Wire		1
200mm 4-Pin Jumper Wire		1
200mm 4-Pin Jumper Wire		1

## 1.1.5 PCB

Robot HATS		1
PCA9685 PWM Driver		1
Motor Driver Module		1
5-CH Line Follower Module		1
Ultrasonic Obstacle Avoidance Module		1





Light Follower Module		1
-----------------------	--	---

### 1.1.6 その他の部品

2x18650 Battery Holder		1
DC Gear Motor		2
Rear Wheel		2
Front Wheel		2
Ribbon (30cm)		1

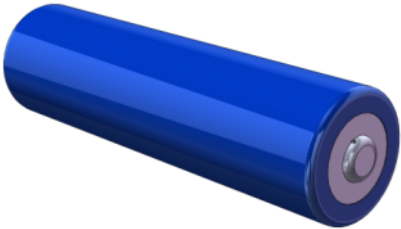


### 1.1.7 工具

Cross Screwdriver		1
Cross Socket Wrench		1
M2.5/M4 Small Wrench		1
M2/M3 Small Wrench		1

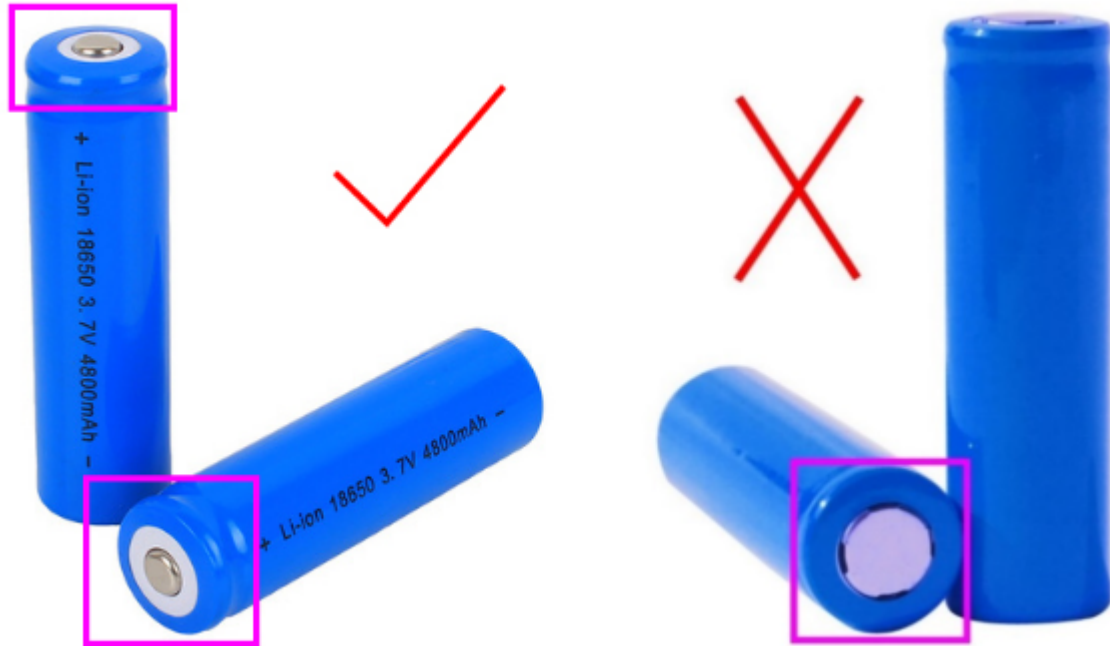
### 1.1.8 別途用意する必要な部品

このキットには以下の成分が含まれていません。

Component	Name	Qty.
	18650 3.7V Rechargeable Li-ion Battery	2

注釈: 保護ボードなしの 18650 バッテリーを使用することを推奨します。他のバッテリーを使用すると、保護ボードの過電流保護により、製品の電源が切れて動作を停止する可能性があります。

保護回路なしのバッテリーの場合は、バッテリーホルダーとの接続を確実にするために、アノードが膨らんでいる(正極が尖った)バッテリーを購入してください(以下を参照)。

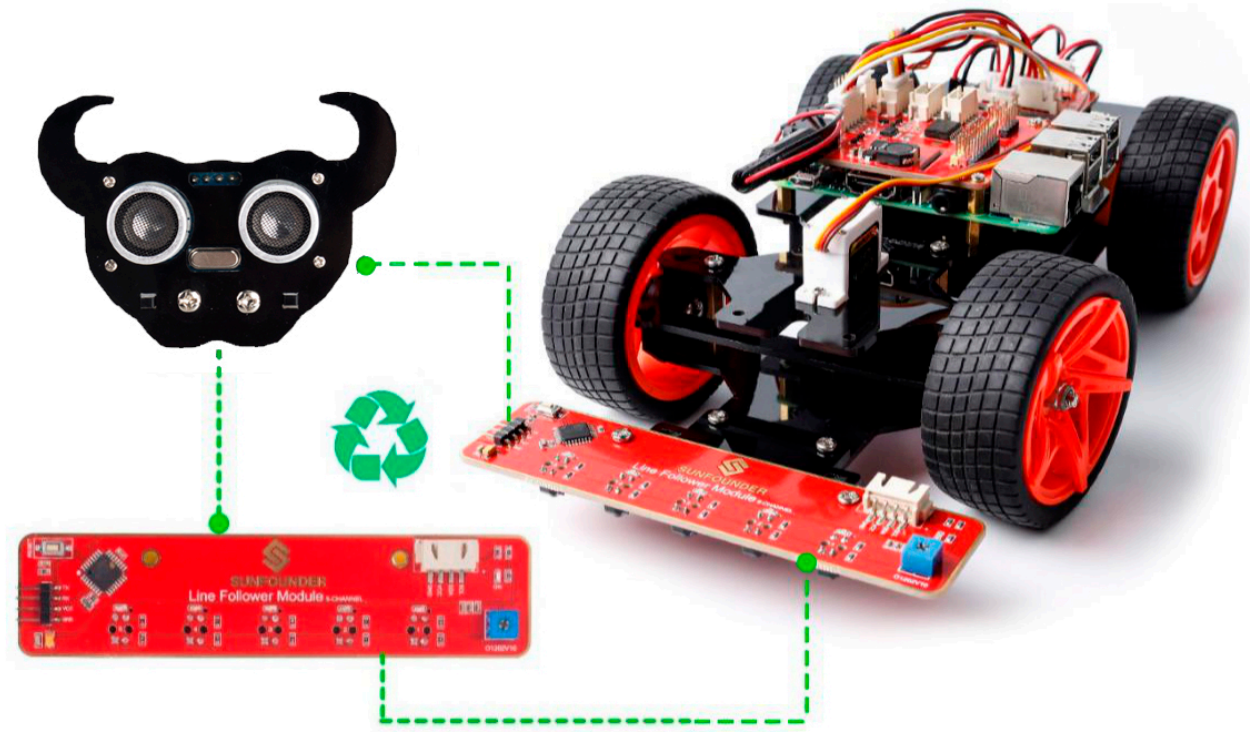


長時間使用するために、可能な限り最大電力のバッテリーを使用してください。

---

## 1.2 前書き

PiCar-S は Raspberry Pi に基づく スマートセンサー カーロボットで、ライトフォロワー、ラインフォロワー、超音波障害物回避を含む 3 つのセンサーモジュールを搭載している。これらのモジュールにより、このスマートカーはいくつかの簡単な自動動作が可能になる。したがって、これらのセンサーで車を制御するために、Python でのプログラミングについての基本知識を学ぶことができる。このスマートカーを作ろう！

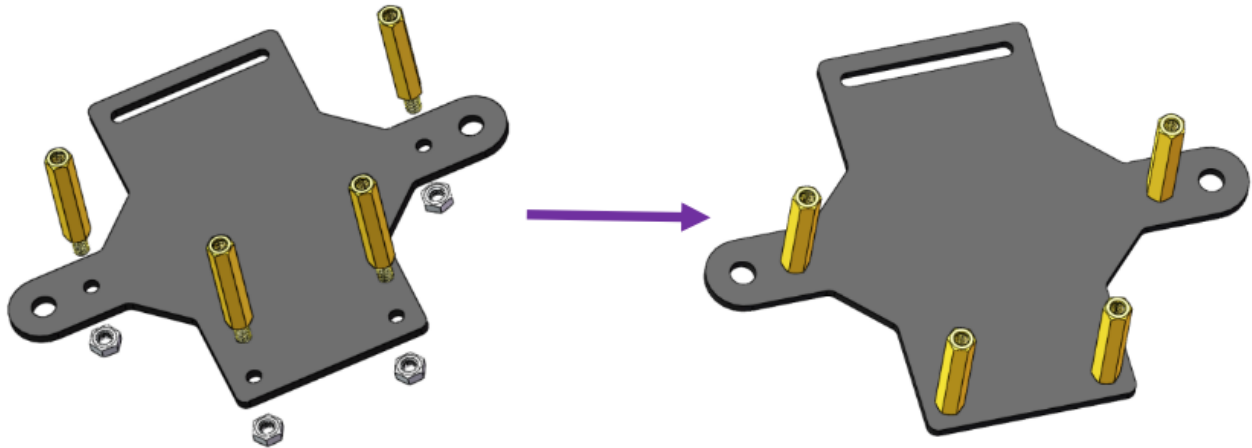


## 1.3 車の組み立て

箱を開けて多くの部品をチェックするとき、とても興奮しているか？ 忍耐を持って、気楽にやってください。次の手順を **CAREFUL** 読んでください。各ステップが終了したら、マニュアルの図に基づいて作業を再確認してください。心配しないで！いくつかの特定のステップでヒントが与えられる。チュートリアルに一步一步で従ってください。さて、もう騒ぎは無く、今から始めよう！

### 1.3.1 フロントハーフシャーシ

4 つの M3x25 銅製スタンドオフ と M3 ナット で前部ハーフシャーシを組み立てる：



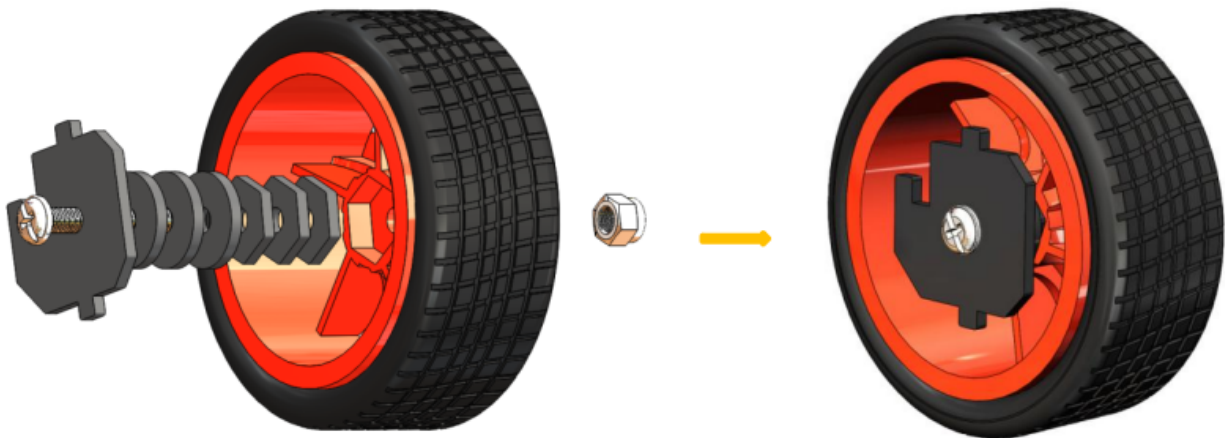
### 1.3.2 前輪

---

注釈: 組み立てる前に、ステアリングコネクタの方向に注意してください。

---

以下に示すように、M4x25 ネジ を一つの ステアリングコネクタ、3 つの ベアリングシールド、3 つの 六角前輪 固定プレート、と一つの 前輪 を通して M4 セルフロックナット (方向に注意してください) に挿入する：

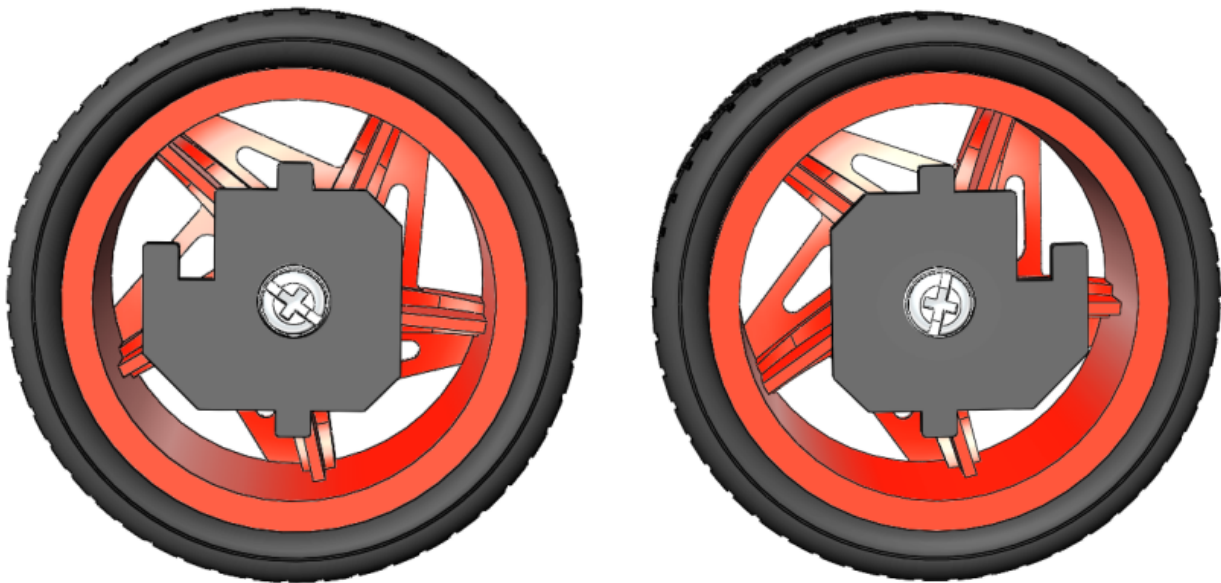


クロスソケットレンチを使用して M4 セルフロックナット を固定してからドライバーを使用して M4x25 ネジ を締める。



注釈: セルフロックナットはしっかりと締まっていることを確認してください。車輪とステアリングコネクタが動かなくなるまでネジを締め、次にネジを少し緩めて、ステアリングプレートだけが動くようにする。したがって、接続が緩すぎない場合、ホイールは柔軟に回転できる。

同じ方法で他の前輪を組み立てるが、車輪のステアリングコネクタは前のものと対称であることを覚えておいてください:



これで2つの前輪の組み立てが完了した。

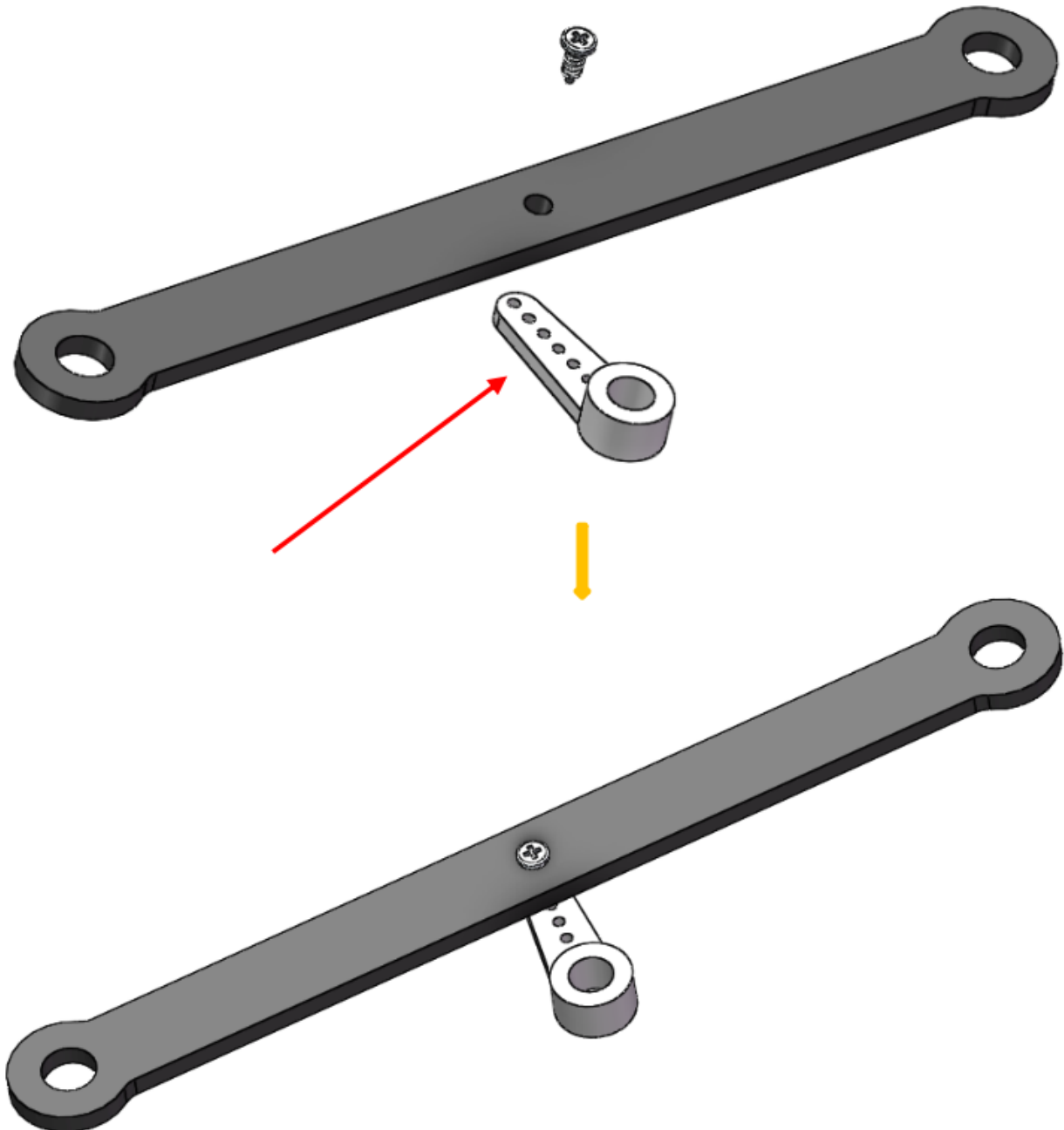
### 1.3.3 ステアリングパーツ

ステアリングリンケージと1アームロッカーアームを M1.5x4 セルフタッピングネジで 接続します。

---

注釈: ギアから最も遠いアームの最初の穴 (下の矢印で示されている) に挿入します。

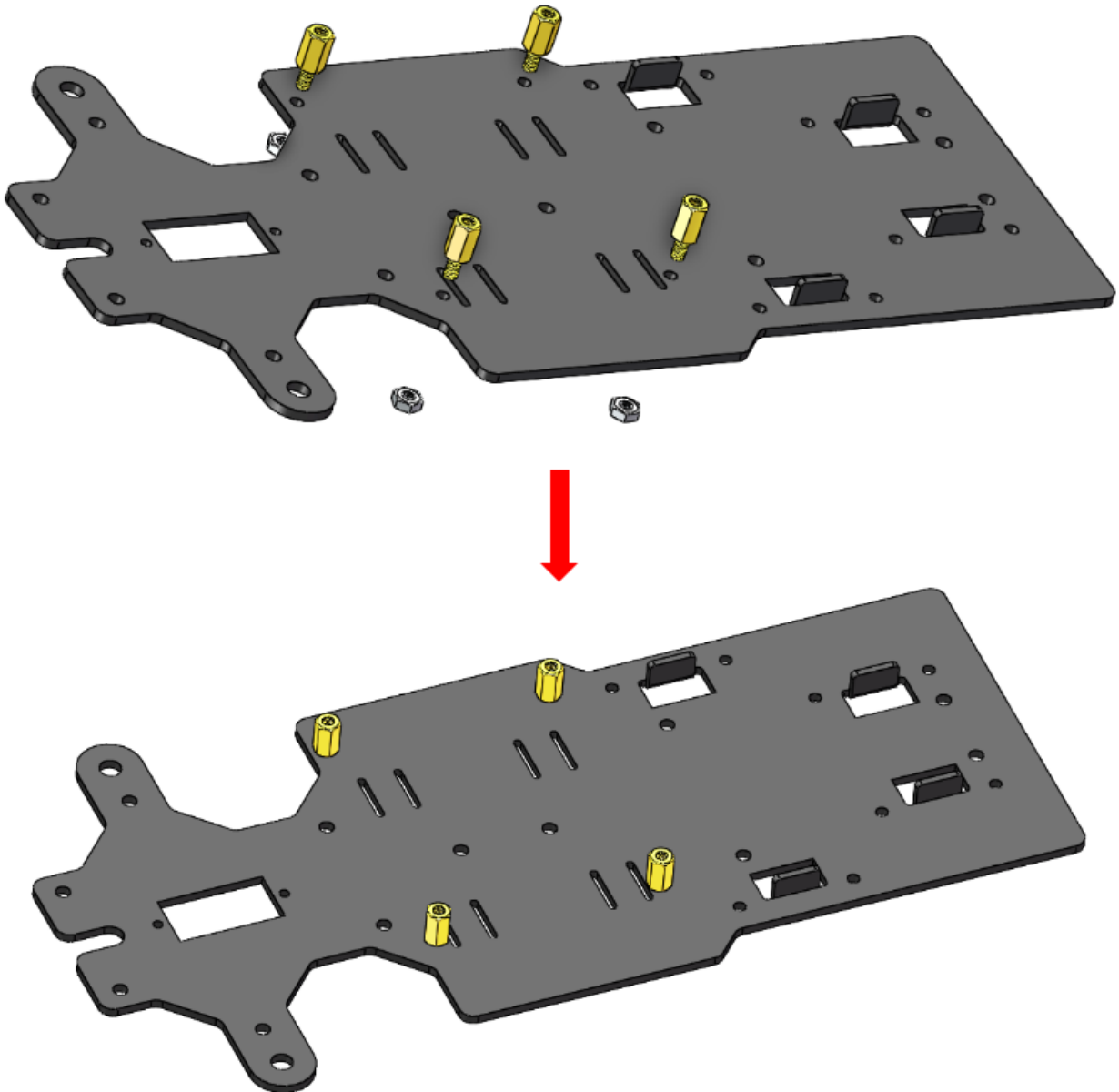
---



注釈: それらをしっかりと締めてから、ネジを少し緩め、ステアリングリンクageが柔軟に動くようにする。

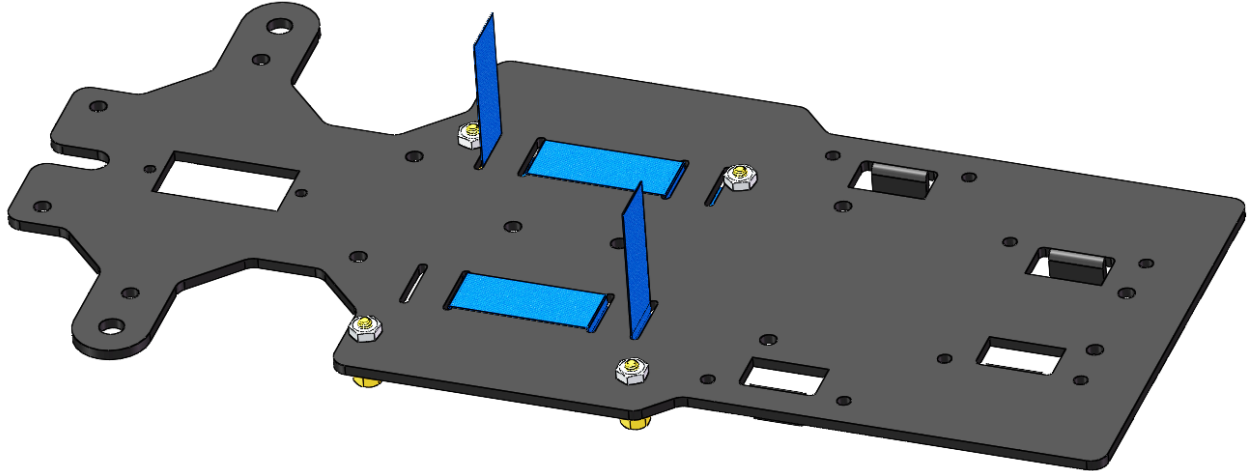
### 1.3.4 上部プレート

最初に M2.5x8 銅製スタンドオフ と M2.5 ナット を 上部プレート に取り付け。突き出た支柱が上に向くように注意してください。



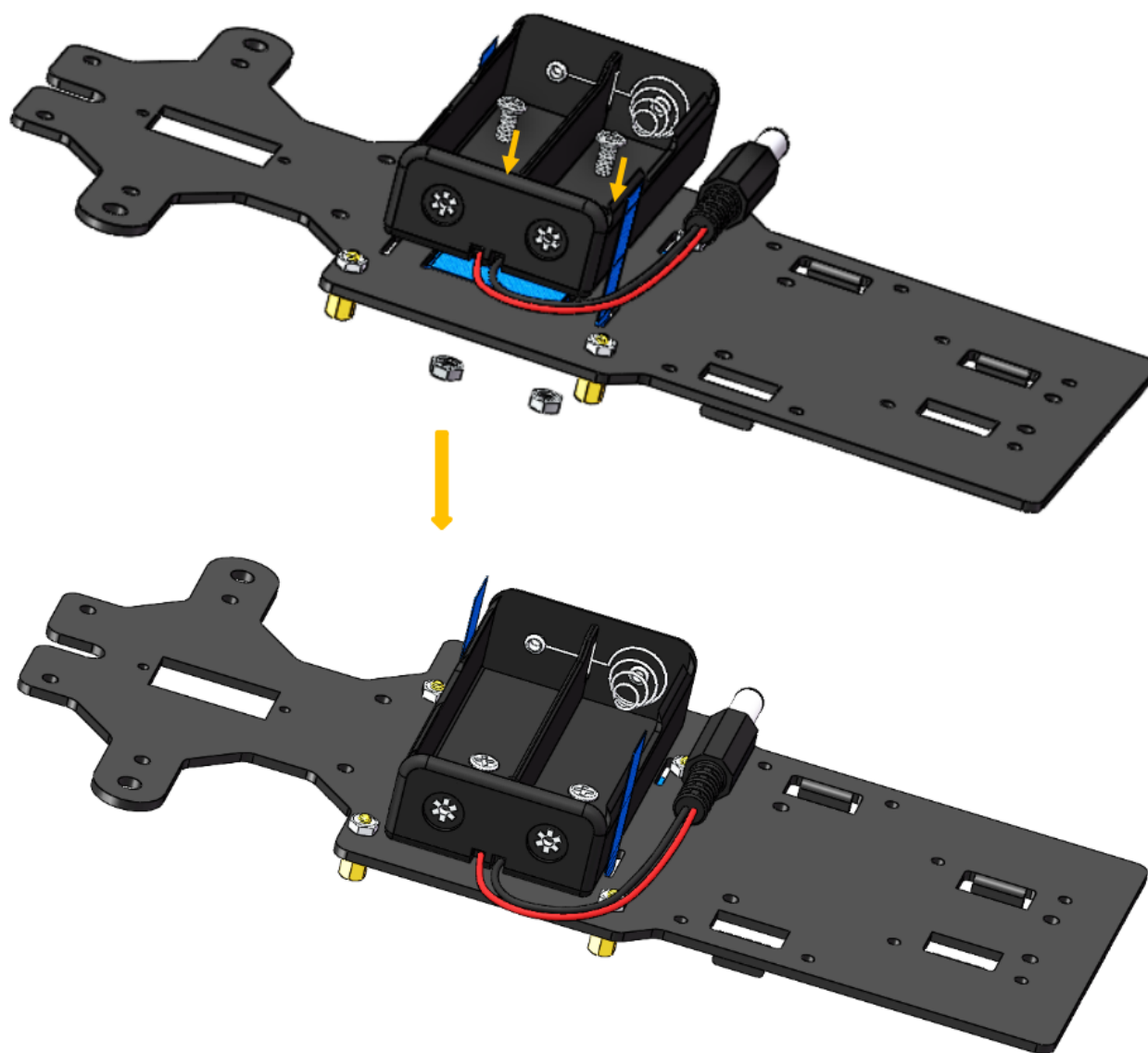
### 1.3.5 バッテリーホルダー

上部プレートを裏返しにする。リボン を半分にカットする。プレートの穴に通す。方向に注意して、バッテリーを後で簡単に取り外すことができるように、プレート的一方の端を長くしてください。



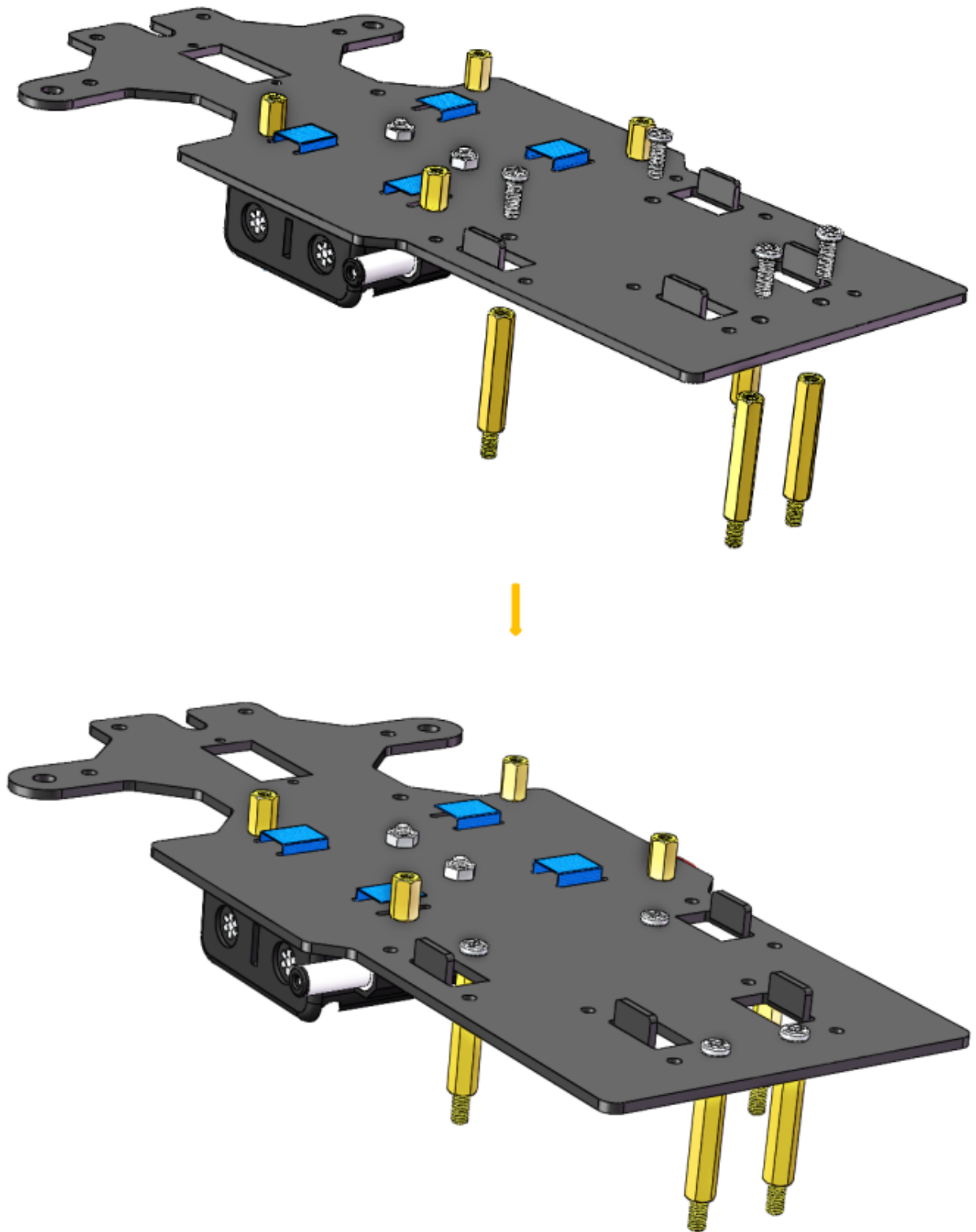
2 本の M3x8 皿ネジ と M3 ナット でバッテリーホルダーを固定する：バッテリーホルダーのワイヤーの方向に注意してください。





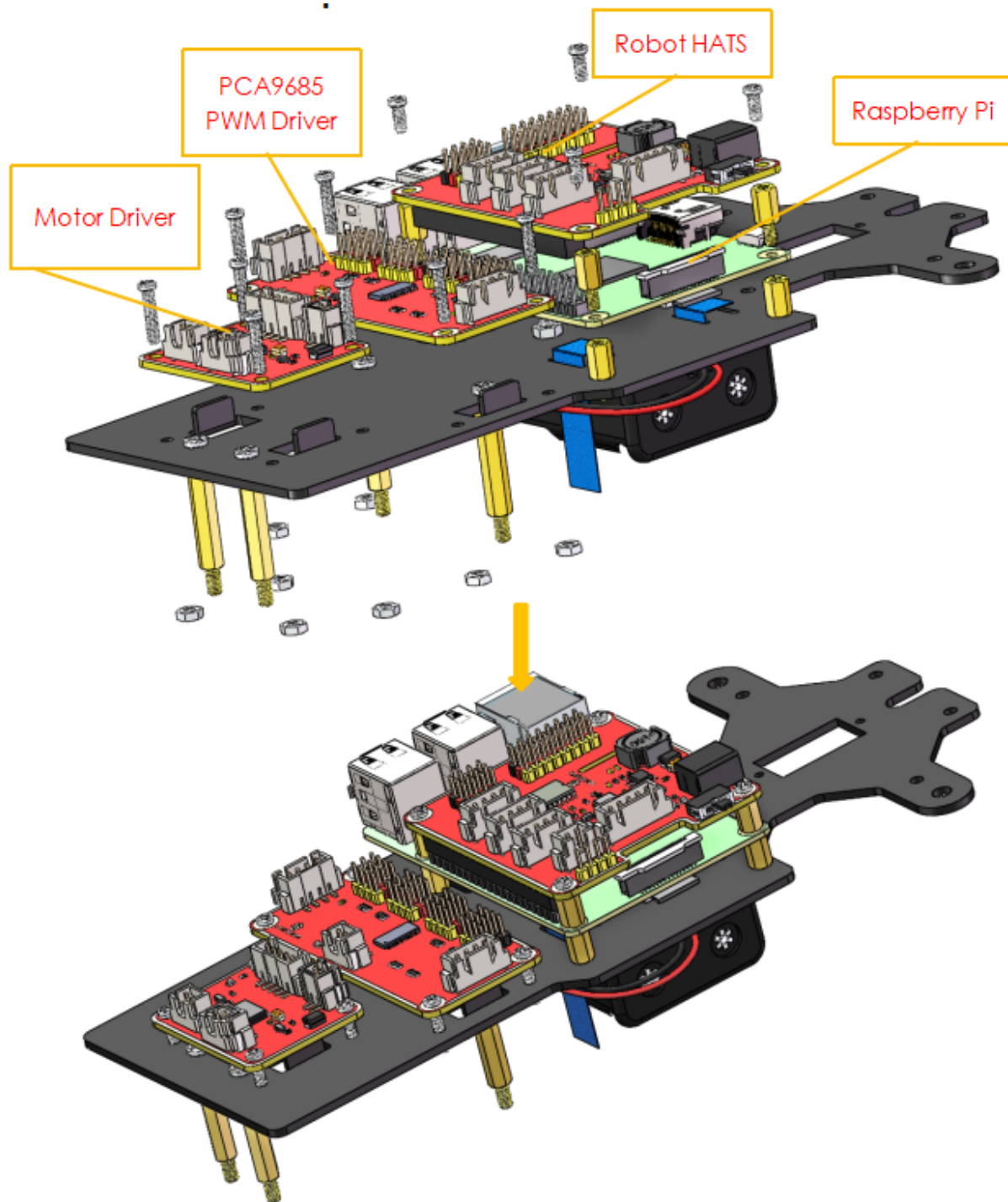
### 1.3.6 後輪（ネジ）

4本の M3x25 銅製スタンドオフ を備えた 4本の M3x8 ネジ を差し込む：



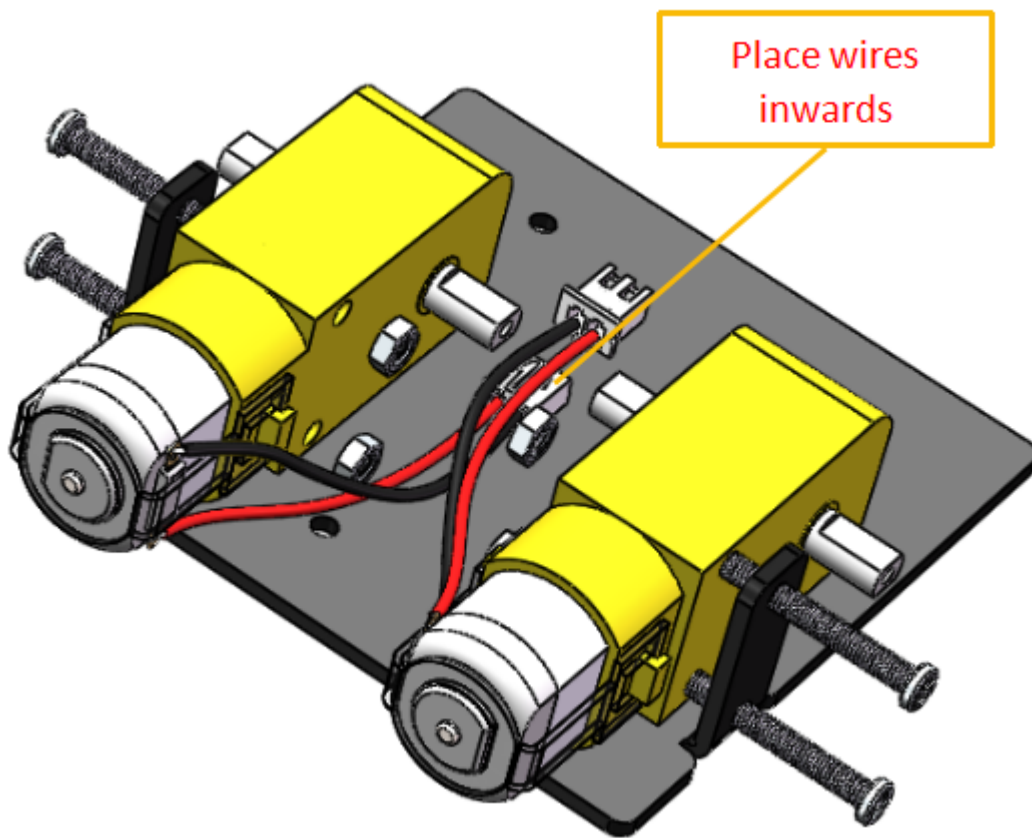
### 1.3.7 PCB 組み立て

- 1) **Raspberry Pi** (TF カード差込済み) を 8 本の **M2.5x8** 銅製スタンドオフ で組み立てから ロボット **HATS** を差し込む。
- 2) 4 本の **M2.5x6** ネジ で ロボット **HATS** を固定する。
- 3) **PCA9685 PWM** ドライバ ーと モータードライバ ーを 8 本の **M2.5x12** ネジと **M2.5** ナット で 下部プレート に固定する。

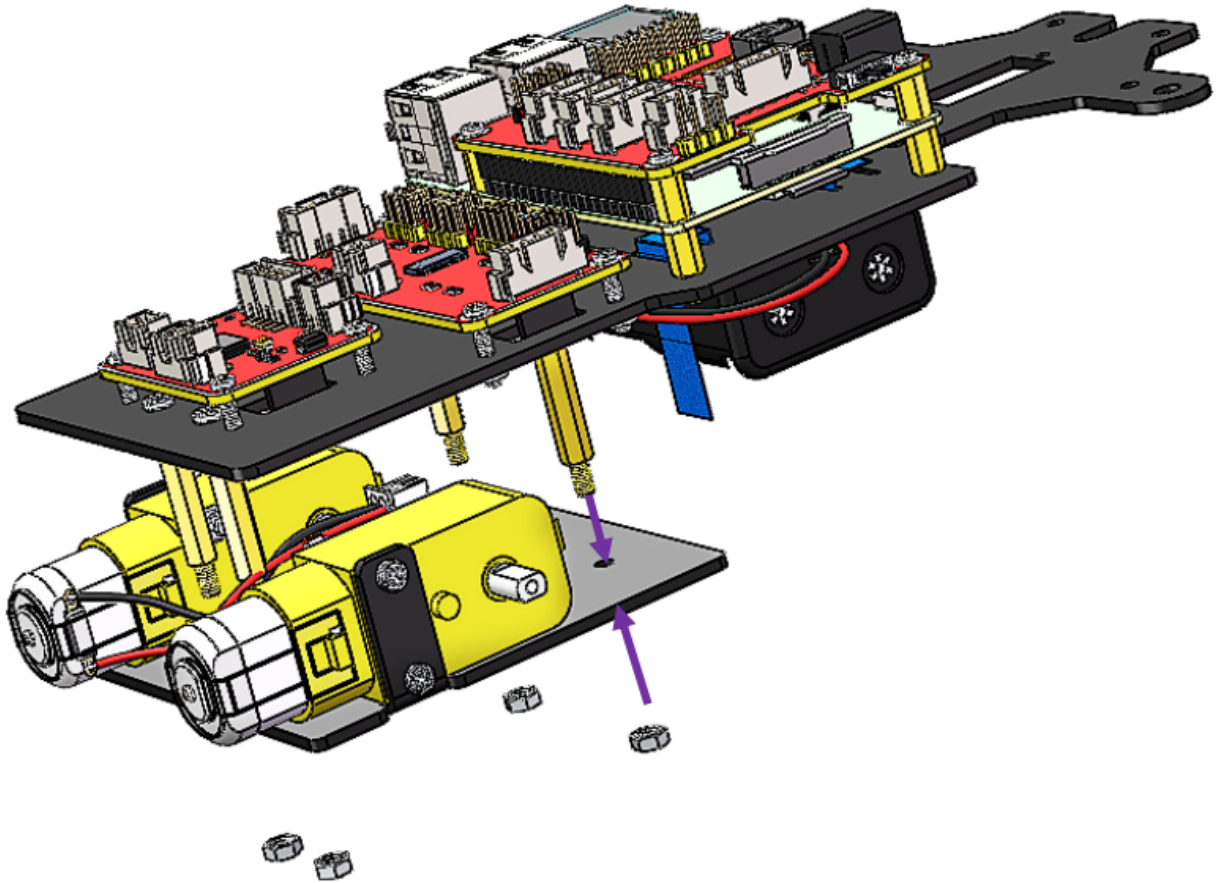


### 1.3.8 後輪（走行）

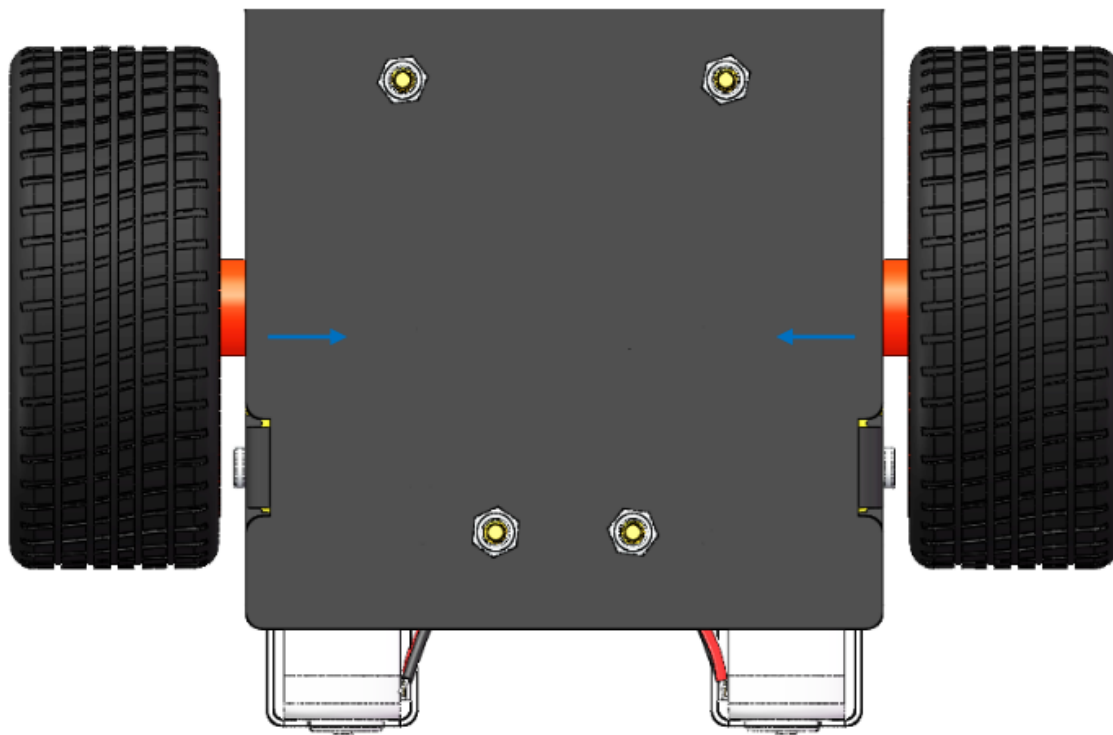
4 本の M3x25 ネジ と M3 ナット で 2 つのモーターを後部ハーフシャーシに組み立てる。モーターを配線で内側に配置するように注意してください。こうすると、回路を接続することを便利にする。



4 つの M3 ナットで後輪を組み立てる。

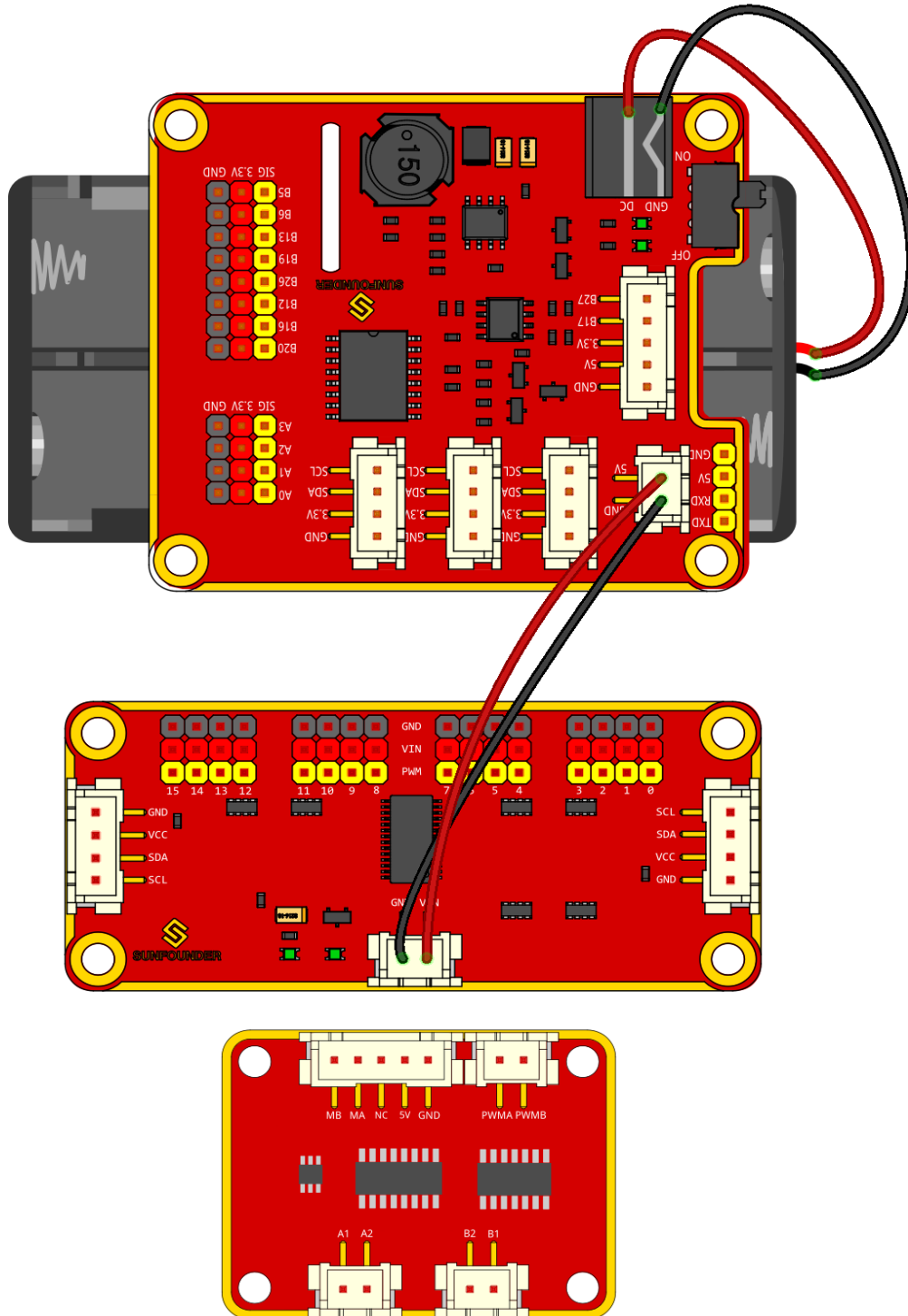


後輪 をモーターシャフトに合わせ、回転させてやさしく差し込む。



## 1.4 回路構築

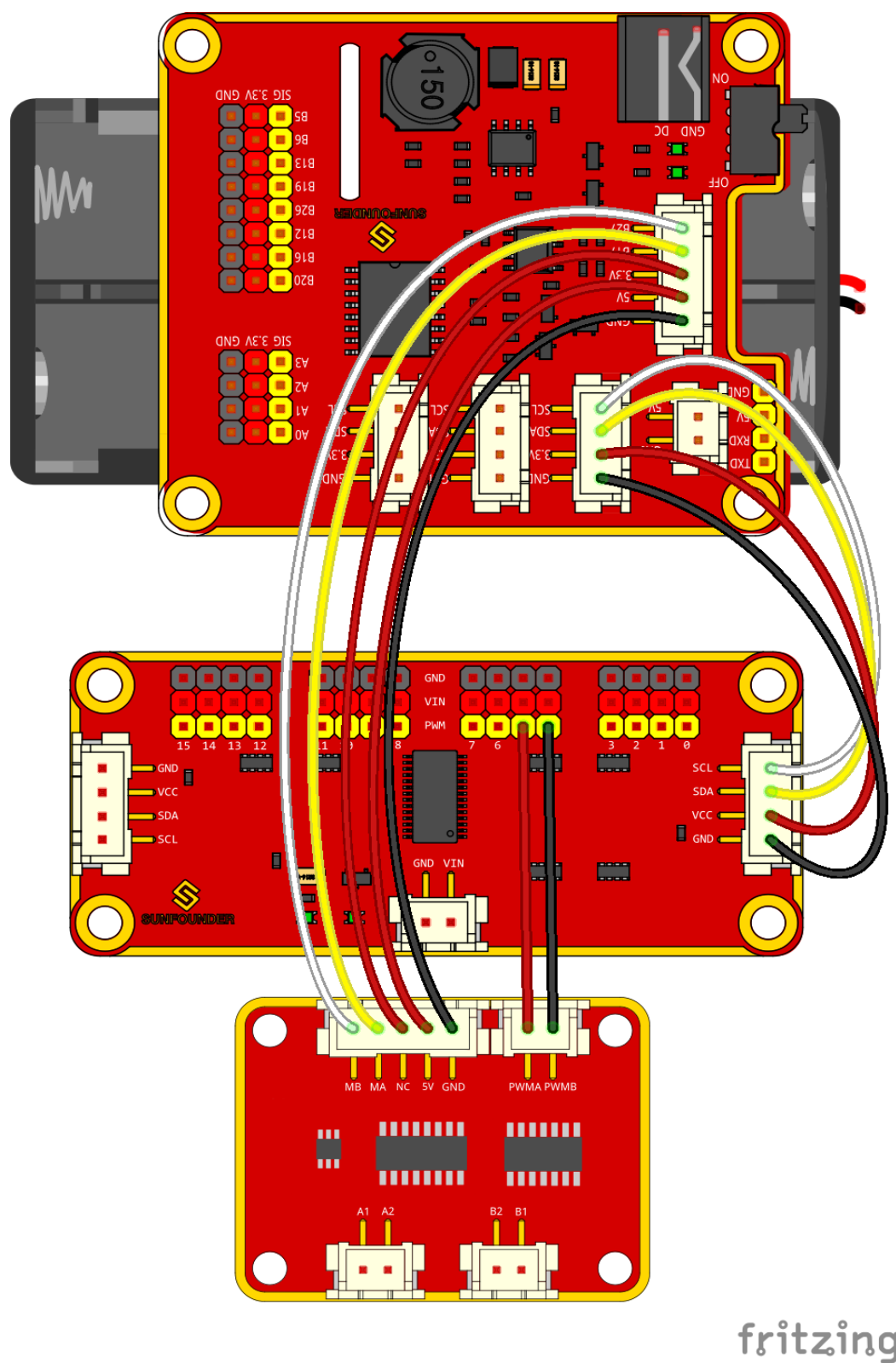
### 1.4.1 電源を接続する



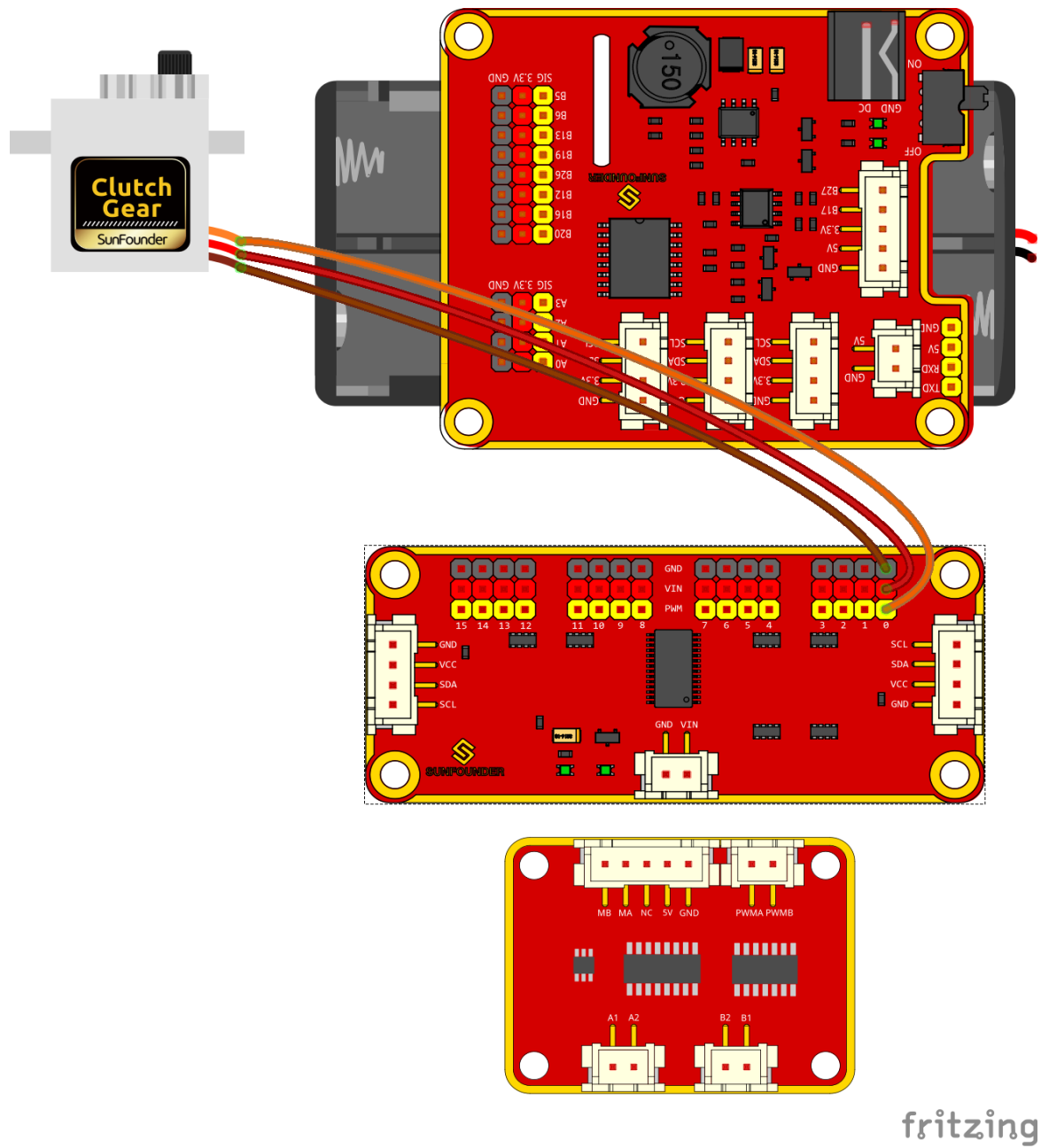
fritzing



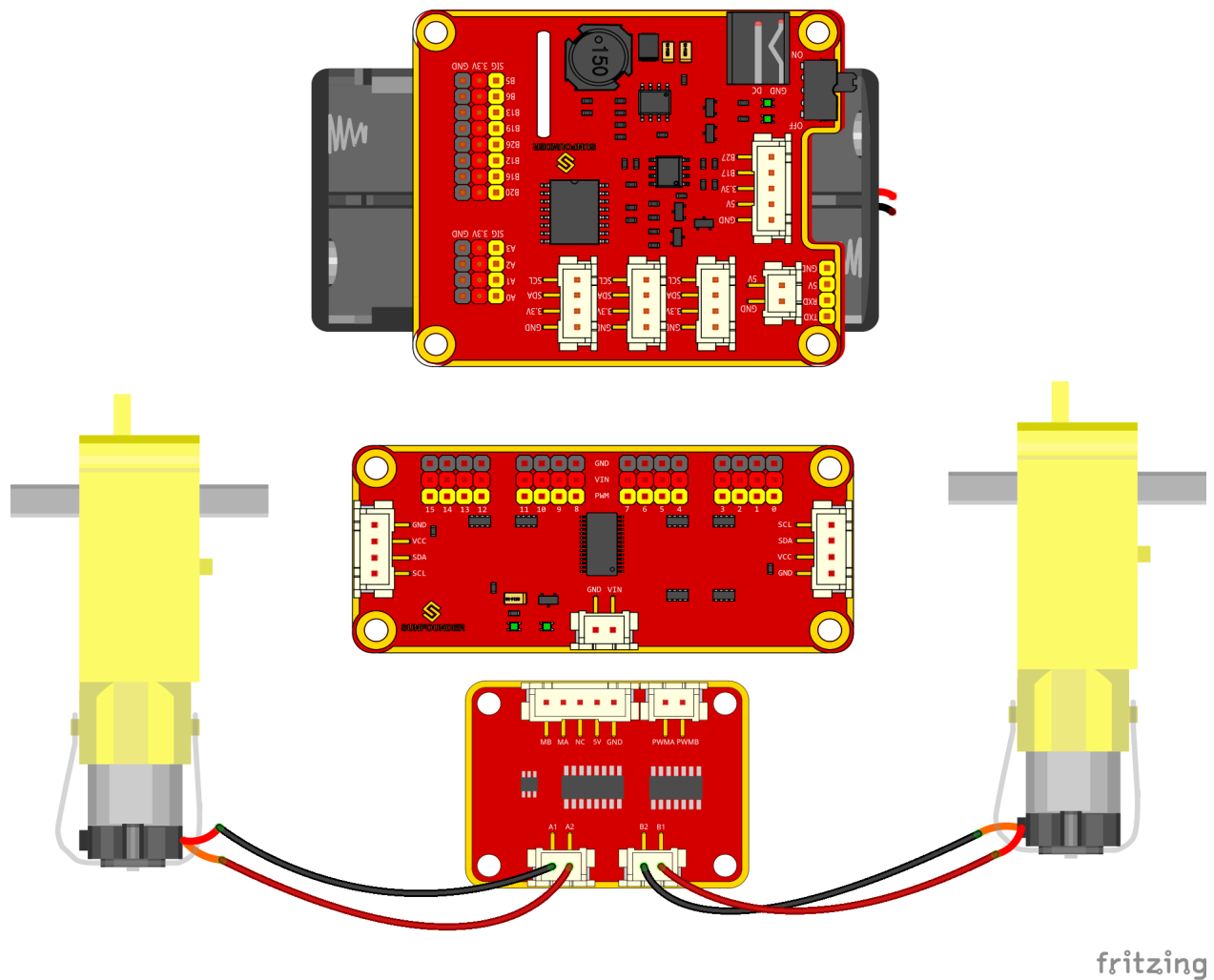
### 1.4.2 モジュールを接続する



### 1.4.3 サーボを接続する



### 1.4.4 モーターを接続する



完全な接続は次のように表示される。



注釈: ラズベリーパイがセットアップされている場合は、この部分をスキップして次の章に進むことができます。

### 1.5.1 オペレーティングシステムのインストール

必要なコンポーネント

任意のラズベリーパイ	パーソナルコンピュータ*1 台
マイクロ SD カード*1 枚	

#### ステップ 1

ラズベリーパイが開発したグラフィカルな SD カード書き込みツールは、Mac OS、Ubuntu 18.04、Windows で動作し、イメージをダウンロードして自動的に SD カードにインストールしてくれるので、ほとんどのユーザーにとって最も簡単なオプションです。

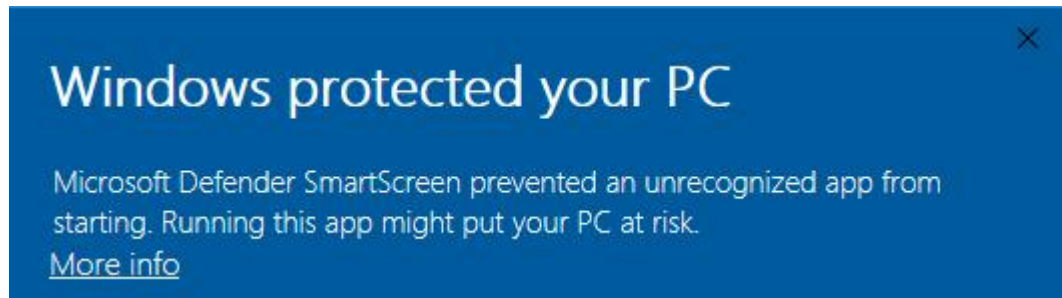
ダウンロードページにアクセスします。<https://www.raspberrypi.org/software/>。お使いのオペレーティングシステムに合ったラズベリーパイイメージャーのリンクをクリックし、ダウンロードが終了したら、それをクリックしてインストーラーを起動します。



#### 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。例えば、Windows では次のようなメッセージが表示されます。

これがポップアップされたら、順次に「More info」と「Run anyway」をクリックした後、指示に従ってラズベリーパイイメージャーをインストールします。

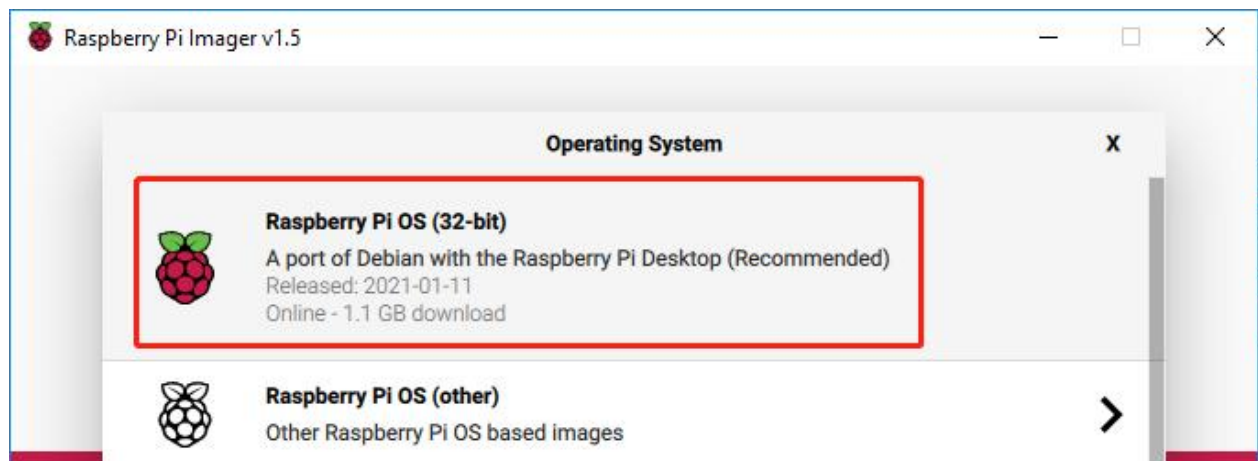


### ステップ 3

SD カードをパソコンやノートパソコンの SD カードスロットに挿入します。

### ステップ 4

ラズベリーパイイメージャーで、インストールしたいオペレーティングシステムと SD カードを選択します。

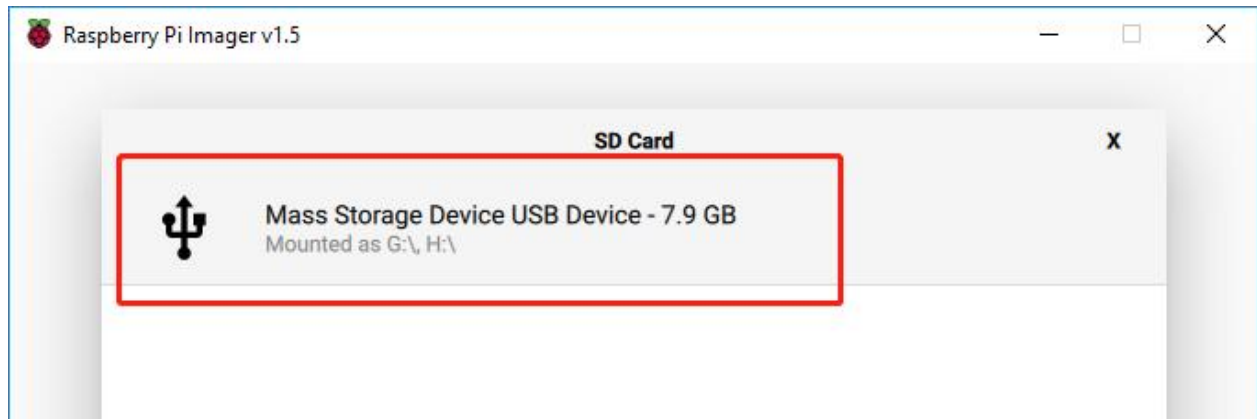


### 注釈:

- 1) 初回はインターネットに接続されている必要があります。
- 2) そのオペレーティングシステムは、将来のオフラインでの使用のために保存されます (ラストダウンロード. キャッシュ、C:/Users/yourname/AppData/Local/Raspberry Pi/Imager/cache, )。そのため、次にソフトを開いたときには、「リリース：日付、あなたのコンピュータにキャッシュされた」という表示になります。

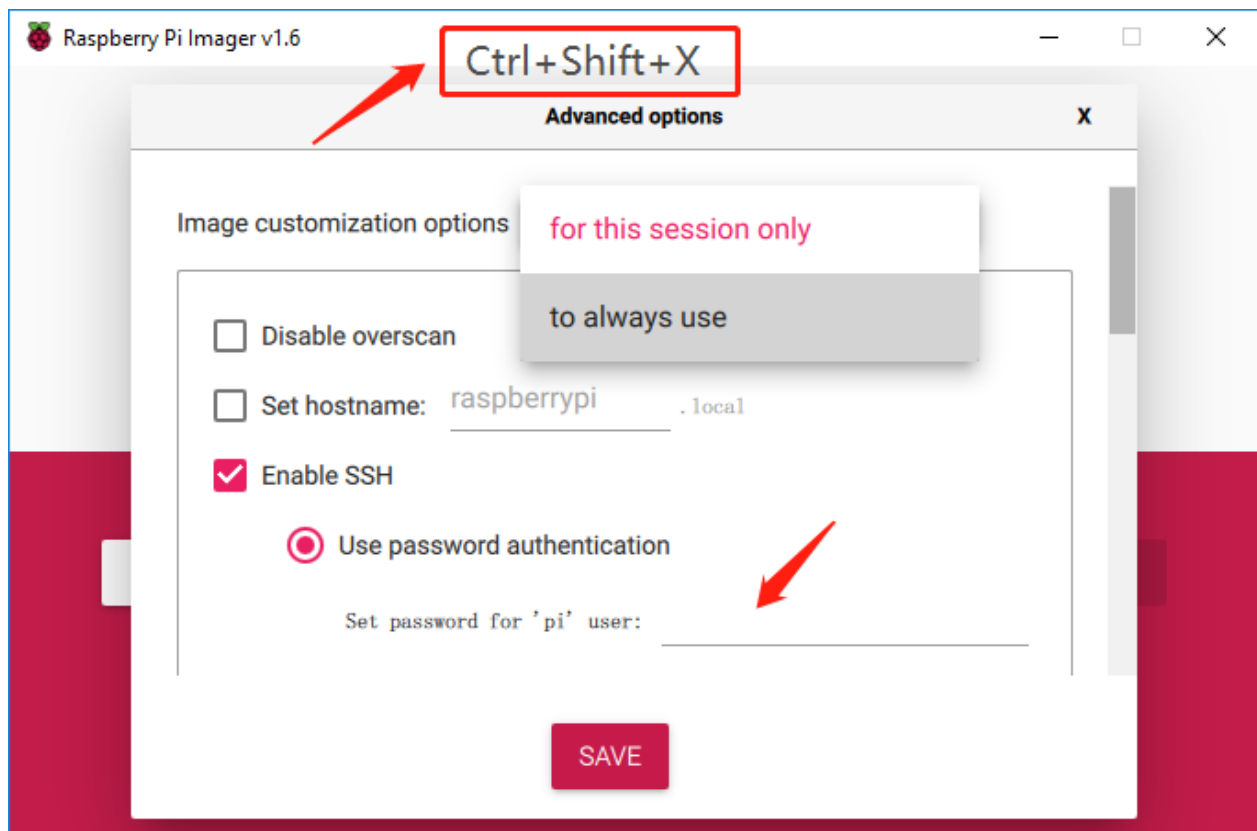
### ステップ 5

使用中の SD カードを選択します。



#### ステップ 6

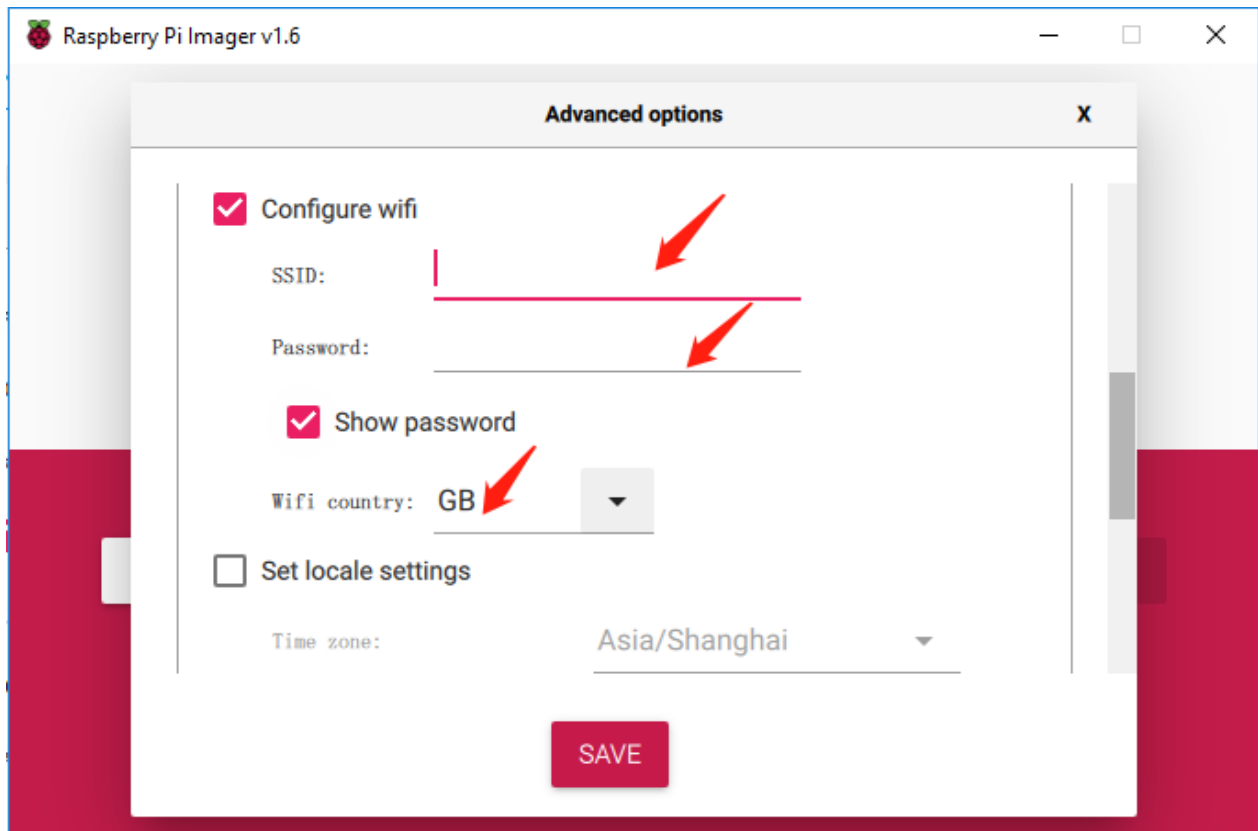
「Ctrl+Shift+X」を押すと、SSHの有効化と無線LANの設定を行うための「Advanced options」ページが開きます。この2つの項目は必ず設定する必要がありますが、その他の項目はあなたの選択次第です。このイメージカスタマイズオプションを常に使用するように選択することができます。



その後、下にスクロールして Wifi の設定を完了し、「SAVE」をクリックします。

注釈: **wifi country** は、ラズベリーパイを使用している国の `ISO/IEC alpha2 code` <[https://en.wikipedia.org/wiki/ISO\\_3166-1\\_alpha-2#Officially\\_assigned\\_code\\_elements](https://en.wikipedia.org/wiki/ISO_3166-1_alpha-2#Officially_assigned_code_elements)>`\_`の2文字のコードを設定してください、以下のリンク

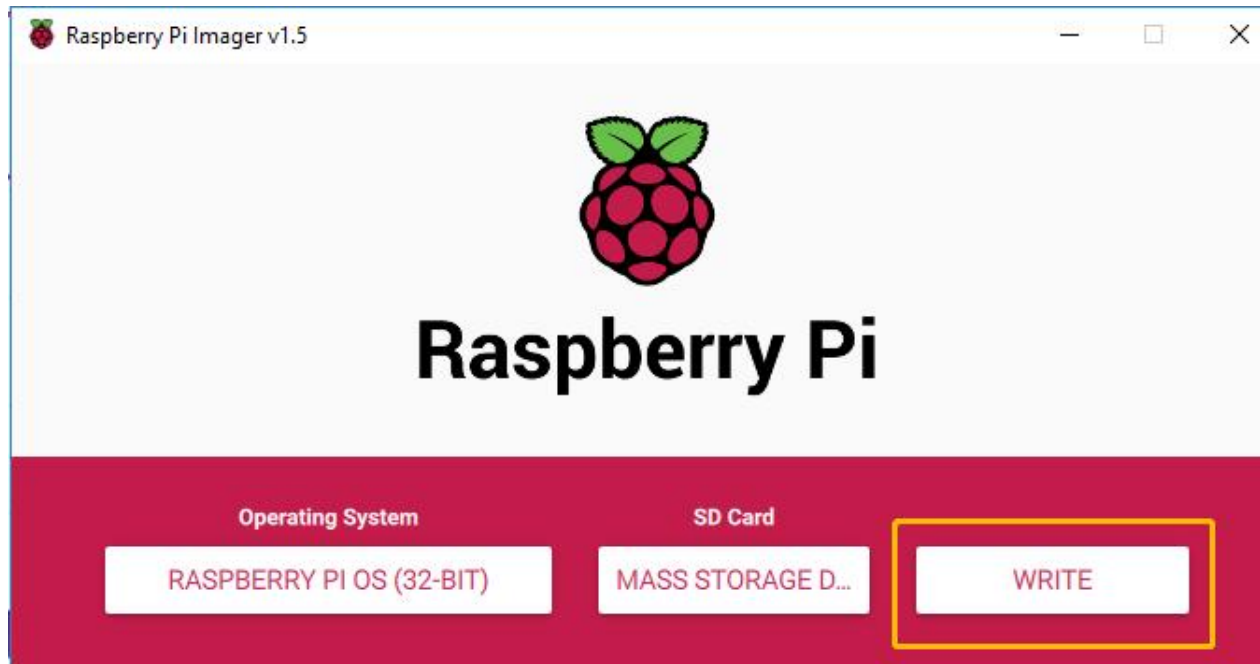
を参照してください : [https://en.wikipedia.org/wiki/ISO\\_3166-1\\_alpha-2#Officially\\_assigned\\_code\\_elements](https://en.wikipedia.org/wiki/ISO_3166-1_alpha-2#Officially_assigned_code_elements)。



#### ステップ 7

「書込み」ボタンをクリックします。





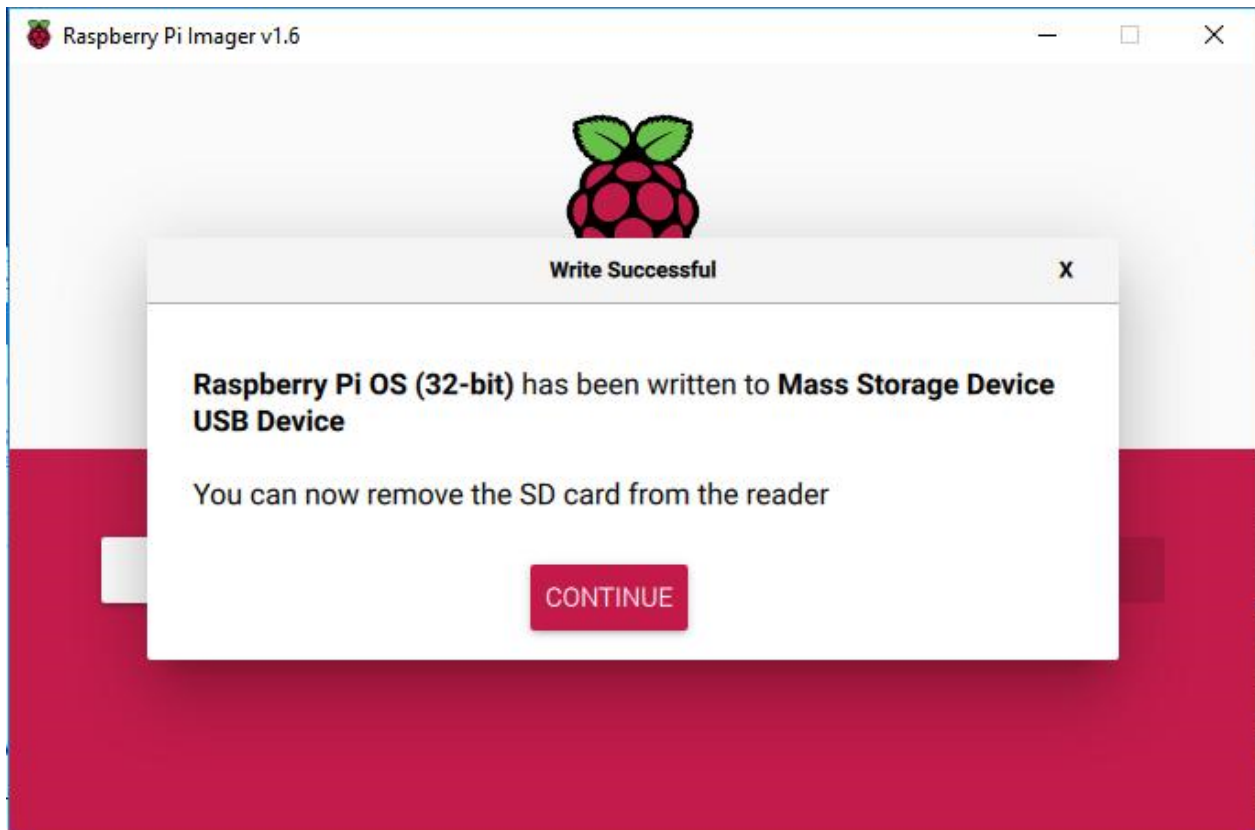
#### ステップ 8

SD カードに何らかのファイルが保存されている場合は、それらのファイルを永久に失わないようにするために、まずそれらのファイルをバックアップすることをお勧めします。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 9

一定時間を待った後、書き込み完了を表す以下のウィンドウが表示されます。



### 1.5.2 Raspberry Pi に電源を入れる

USB カードリーダーを取り出し、SD カードを RaspberryPi に挿入します。

完全に充電された 2 つの 18650 バッテリーをホルダーに入れ、バッテリーホルダーからのワイヤーを開発ボードに差し込み、スイッチをオフからオンに切り替えます。また、最初のテストに時間がかかるため、RaspberryPi の電源アダプターを使用してカーキットに電力を供給することをお勧めします。

### 1.5.3 IP アドレスを取得する

Raspberry Pi の電源を入れたら、その IP アドレスを取得する必要がある。IP アドレスを知る方法はたくさんあるが、そのうちの 2 つを以下のように示す。

#### 1. ルーター経由で確認する

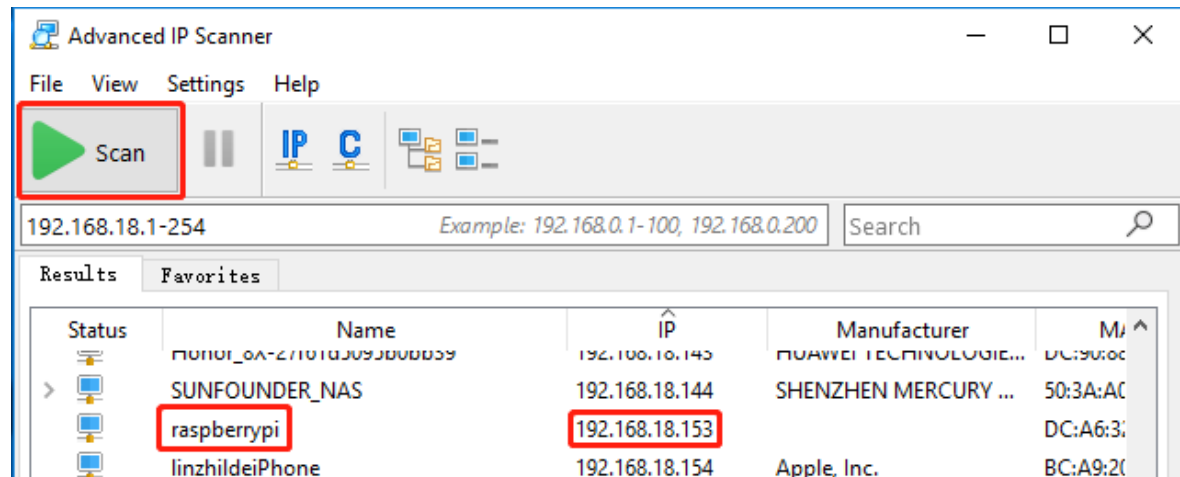
ルーター（ホームネットワークなど）にログインする権限がある場合は、ルーターの管理インターフェイスで Raspberry Pi に割り当てられたアドレスを確認できる。

システムのデフォルトのホスト名 - Raspberry Pi OS は **raspberrypi** であり、それを見つける必要がある。（ArchLinuxARM システムを使用している場合は、alarmpi を見つけてください。）

## 2. ネットワークセグメントスキャン

ネットワークスキャンを使用して、Raspberry Pi の IP アドレスを検索することもできる。ソフトウェア、**Advanced IP scanner**（Google からダウンロード）を使える。

「スキャン」をクリックすると、接続されているすべてのデバイスの名前が表示される。同様に、Raspberry Pi OS のデフォルトのホスト名は **raspberrypi** であり、今ホスト名とその IP を見つけてください。



### 1.5.4 SSH リモートコントロールを使用する

SSH を適用することにより、Raspberry Pi の Bash Shell を開くことができる。Bash は Linux の標準のデフォルトシェルである。シェル自体は、お客様と Unix/Linux をリンクするブリッジである「C」で書き込まれるプログラムである。さらに、必要な作業のほとんどを完了することに役立ち。

#### Linux または/ Mac OS X ユーザーの場合

##### ステップ 1

Applications->Utilities に入り、**Terminal** を見つけてから開く。

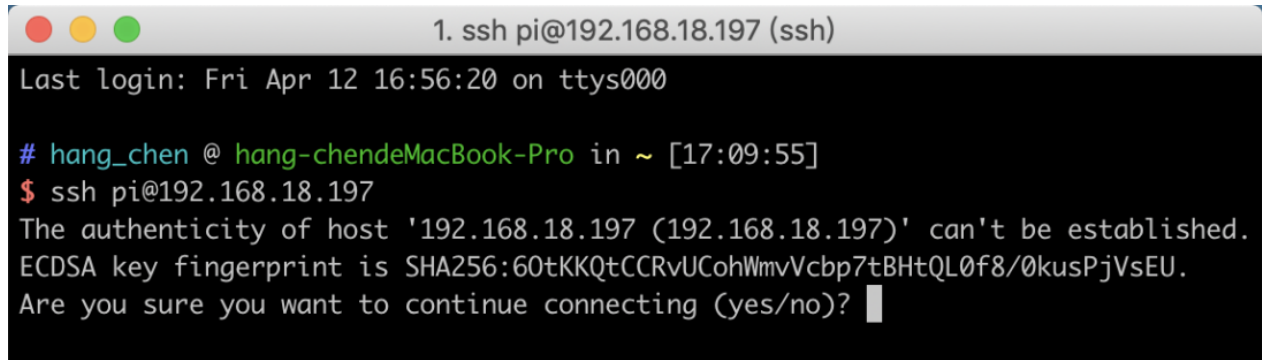
##### ステップ 2

`ssh pi@ip_address` と入力します。"pi" はユーザー名、"ip\_address" は IP アドレスです。例えば：

```
ssh pi@192.168.18.197
```

##### ステップ 3

"yes" を入力します。

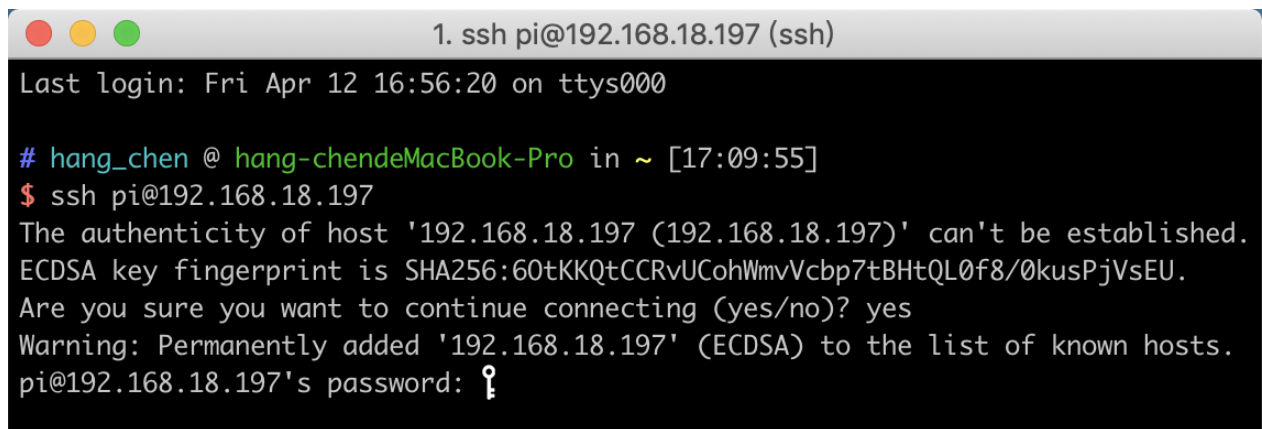
A terminal window titled "1. ssh pi@192.168.18.197 (ssh)" with standard macOS window controls. The terminal output shows a successful login for the user 'hang\_chen' on a MacBook-Pro. The prompt is '# hang\_chen @ hang-chendeMacBook-Pro in ~ [17:09:55]'. The user enters '\$ ssh pi@192.168.18.197'. The terminal displays a warning about the host's authenticity, showing the ECDSA key fingerprint: 'SHA256:60tKKQtCCRvUCohWmvVcbp7tBHtQL0f8/0kusPjVsEU'. It asks 'Are you sure you want to continue connecting (yes/no)?' with a cursor at the end.

```
1. ssh pi@192.168.18.197 (ssh)
Last login: Fri Apr 12 16:56:20 on ttys000

# hang_chen @ hang-chendeMacBook-Pro in ~ [17:09:55]
$ ssh pi@192.168.18.197
The authenticity of host '192.168.18.197 (192.168.18.197)' can't be established.
ECDSA key fingerprint is SHA256:60tKKQtCCRvUCohWmvVcbp7tBHtQL0f8/0kusPjVsEU.
Are you sure you want to continue connecting (yes/no)?
```

#### ステップ 4

デフォルトのパスワード「**raspberr**y」を入力する。

A terminal window titled "1. ssh pi@192.168.18.197 (ssh)" showing the continuation of the SSH session. The user has responded 'yes' to the authenticity warning. The terminal displays 'Warning: Permanently added '192.168.18.197' (ECDSA) to the list of known hosts.' and prompts for the password: 'pi@192.168.18.197's password:'. A cursor is visible after the prompt.

```
1. ssh pi@192.168.18.197 (ssh)
Last login: Fri Apr 12 16:56:20 on ttys000

# hang_chen @ hang-chendeMacBook-Pro in ~ [17:09:55]
$ ssh pi@192.168.18.197
The authenticity of host '192.168.18.197 (192.168.18.197)' can't be established.
ECDSA key fingerprint is SHA256:60tKKQtCCRvUCohWmvVcbp7tBHtQL0f8/0kusPjVsEU.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.18.197' (ECDSA) to the list of known hosts.
pi@192.168.18.197's password:
```

#### ステップ 5

これで、Raspberry Pi が接続され、次のステップに進む準備ができた。

```
1. pi@raspberrypi: ~ (ssh)
Last login: Fri Apr 12 16:56:20 on ttys000

# hang_chen @ hang-chendeMacBook-Pro in ~ [17:09:55]
$ ssh pi@192.168.18.197
The authenticity of host '192.168.18.197 (192.168.18.197)' can't be established.
ECDSA key fingerprint is SHA256:60tKKQtCCRVUCohWmvVcbp7tBHtQL0f8/0kusPjVsEU.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.18.197' (ECDSA) to the list of known hosts.
pi@192.168.18.197's password:
Linux raspberrypi 4.9.80-v7+ #1098 SMP Fri Mar 9 19:11:42 GMT 2018 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue May 21 07:29:46 2019 from 192.168.18.126

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set
a new password.

pi@raspberrypi:~ $
```

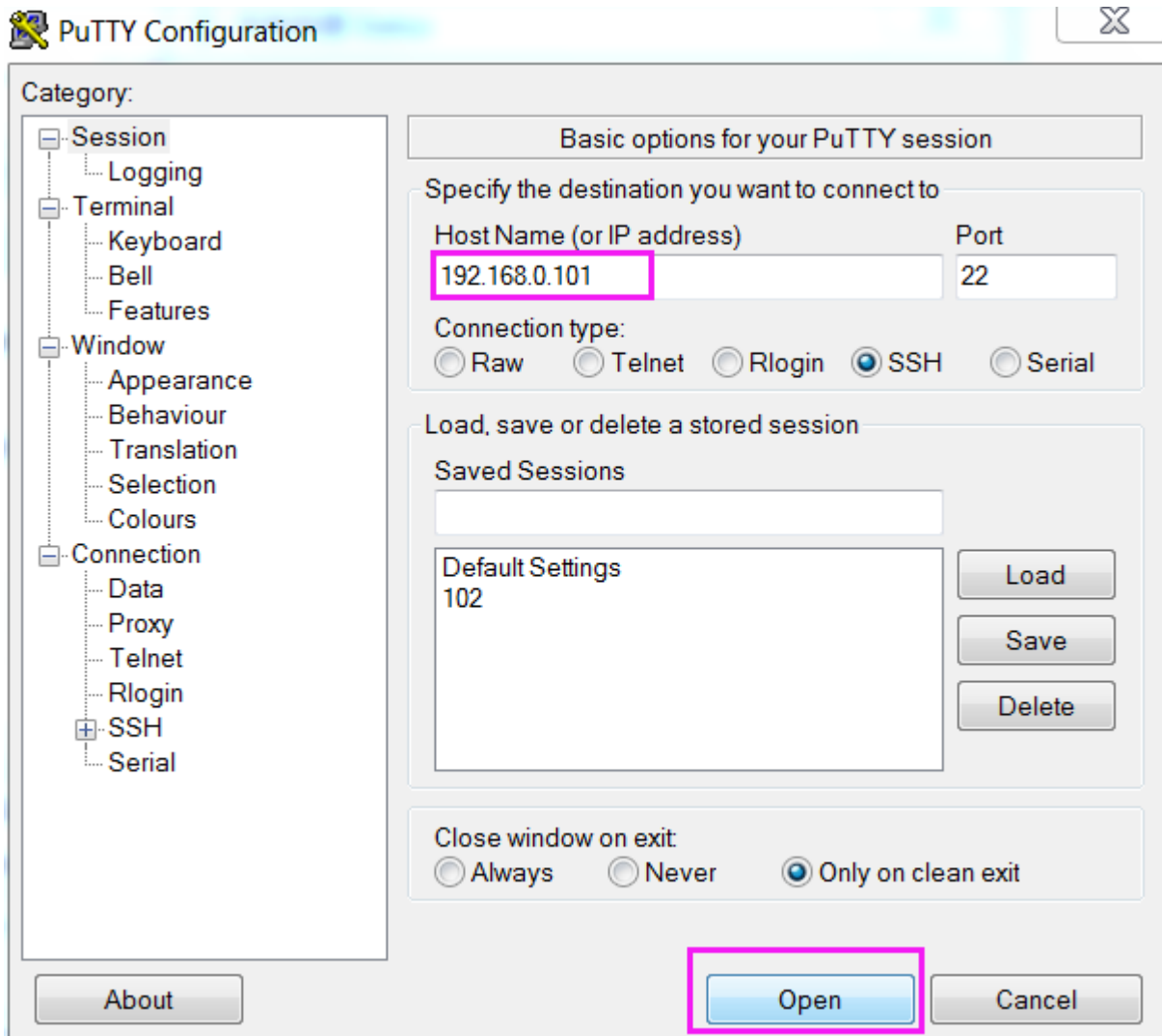
注釈: パスワードを入力すると、ウィンドウに文字が表示されないが、これは正常である。必要なのは、正しいパスコードを入力するだけである。

### Windows ユーザーの場合

Windows ユーザーの場合、いくつかのソフトウェアのアプリケーションで SSH を使用できます。ここでは、PuTTY をお勧めします (Google からダウンロードできます)。

#### ステップ 1

PuTTY をダウンロードする。PuTTY を開き、左側のツリー構造にある **Session** をクリックする。 **Host Name (or IP address)** の下のテキストボックスに RPi の IP アドレスを入力し、 **Port** に **22** (デフォルトでは 22) を入力する。



## ステップ 2

**Open** をクリックする。IP アドレスを使用して Raspberry Pi に初めてログインすると、安全上の指示が表示されることに注意してください。Yes をクリックする。

## ステップ 3

PuTTY ウィンドウに「**login as :**」と表示されたら、「**pi**」(RPi のユーザー名)とパスワード「**raspberrry**」(変更していない場合はデフォルトのパスワードである)を入力する。



```
pi@raspberrypi: ~
login as: pi
pi@192.168.0.234's password: raspberry

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Feb 21 02:54:55 2017
pi@raspberrypi:~ $
```

ここで、Raspberry Pi を接続し、次の手順を実行する。

## 1.6 サーボ構成

また、このキットで使用されるサーボはソフトウェアによって調整され、他のサーボほど物理的な固定点がないため、ここではソフトウェアを介してサーボを構成する。最初に、構成の前にソフトウェアの実装を完了してください。

---

注釈：本章では、バッテリーを取り付けて電源スイッチを ON にスライドすることを忘れないでください。

---

### 1.6.1 ソースコードを取得する

ソースコードは Github リポジトリにある。git clone でソースコードをダウンロードする：

```
cd /home/pi/
git clone --recursive https://github.com/sunfounder/SunFounder_PiCar-S.git -b V3.0
```

---

注釈：入力時に十分に注意してください。ユーザー名とパスワードの入力を求められた場合は、入力を間違えている可能性がある。それは発生した場合、Ctrl+C を押して終了し、再試行してください。

---

### 1.6.2 コードディレクトリに入る

```
cd ~/SunFounder_PiCar-S/  
ls
```

コードディレクトリに入ると、インストールスクリプトが表示される：

```
pi@raspberrypi:~ $ cd ~/SunFounder_PiCar-S/  
pi@raspberrypi:~/SunFounder_PiCar-S $ ls  
example  install_dependencies  LICENSE  maps  README.md  show  
pi@raspberrypi:~/SunFounder_PiCar-S $
```

### 1.6.3 スクリプトを介して環境をインストールする

インストールスクリプトを使用して、必要なすべてのソフトウェアと構成を実行することはできる。代わりに一步一步に実行する場合は、「付録 1：手動によるインストール」の手順に従ってください。

```
sudo ./install_dependencies
```

注釈:

1. インストールスクリプトは、必要な部品をインストールし、動作環境用に構成する。インストール中に Raspberry Pi がインターネットに接続されていることを確認してください。接続しない場合は、失敗する恐れがある。
2. インストールを完了した後、Raspberry Pi は再起動すると示す。再起動するには **yes** と入力してください。

### 1.6.4 サーボを 90 度に設定する

再起動後、次のコマンドを入力する：

```
picar
```

ここに 3 つのコマンドが表示される。

```
pi@raspberrypi:~ $ picar  
Usage:  picar [Command] [value]  
Commands:  
servo-install          Set 16 channel servos to 90 degree for installation  
front-wheel-test [chn] Test the steering servo connect to chn, chn default  
0  
rear-wheel-test        Test the rear wheel
```



最初の 1 つは、前輪を組み立てた後に使用する サーボ調整用の `servo-install` である。このコマンドを実行するとサーボが 90 度回転するため、ここではこのコマンドを使用する。

```
picar servo-install
```

```
pi@raspberrypi:~ $ picar servo-install
Servo now is set to 90 degree.
```

注釈：「`OSError: [Errno 121] Remote I/O error`」エラーメッセージが表示された場合は、`raspi-config` を開く：

```
sudo raspi-config
```

次に、**3 Interfacing Options** → **<P5 I2C>** → **<YES>** → **<OK>**を選択して I2C サービスを有効にします。キーボードの上、下、左、右のキーを使って選択できます、最後は`<Enter>`キーを押して確認します。

コードの実行後、ロッカーアームをサーボに差し込む。ロッカーアームが時計回りと反時計回りに回転し、特定の位置で停止する。サーボが良い状態にあると示す。以下の条件のいずれかがサーボに発生した場合、サーボは不良である：

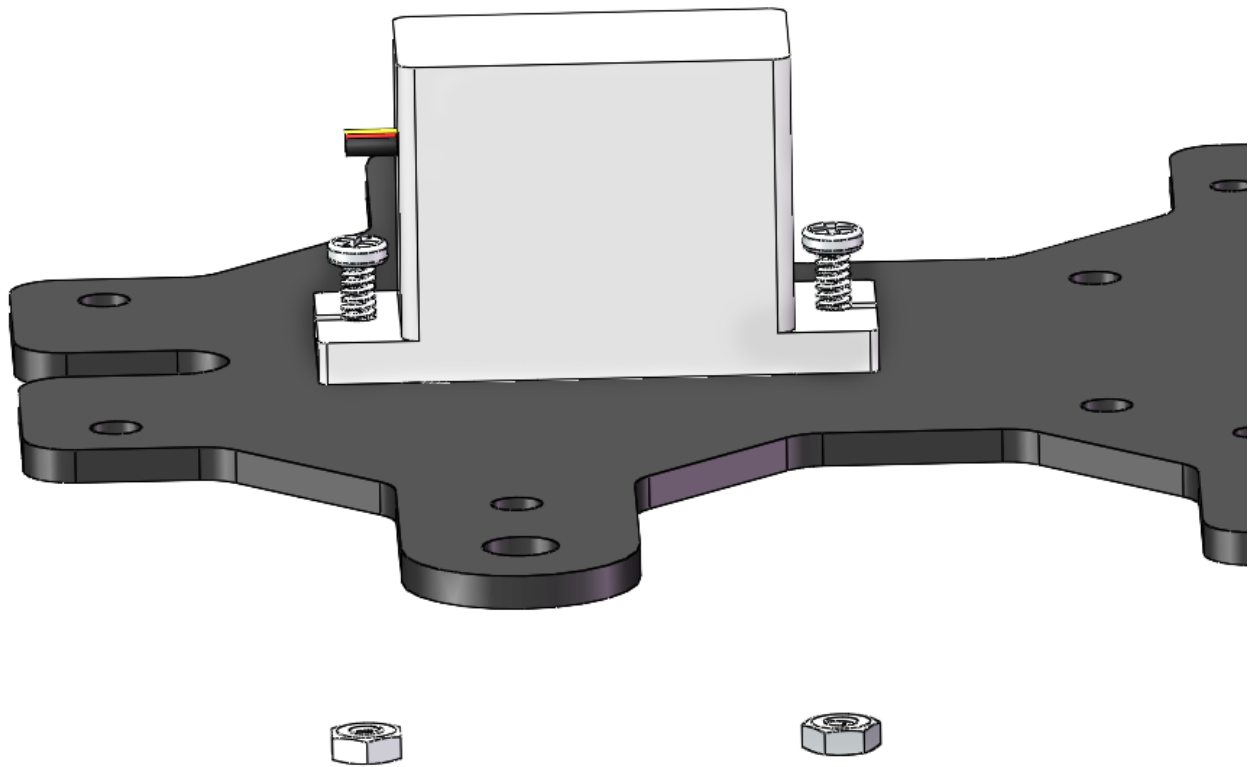
- 1) 雑音あり、熱い。
- 2) サーボラインを抜いてロッカーアームを回すと「カ」「カ」「カ」のように聞こえるか、またはギアの駆動音が鳴らさない。
- 3) ゆっくりで継続的に回してください。

上記の状況のいずれかが発生した場合は、[service@sunfounder.com](mailto:service@sunfounder.com) に送信してください。新品に交換してあげる。使用または組み立ての過程で破損した場合は、公式ウェブサイト <http://www.sunfounder.com> にて購入してください。

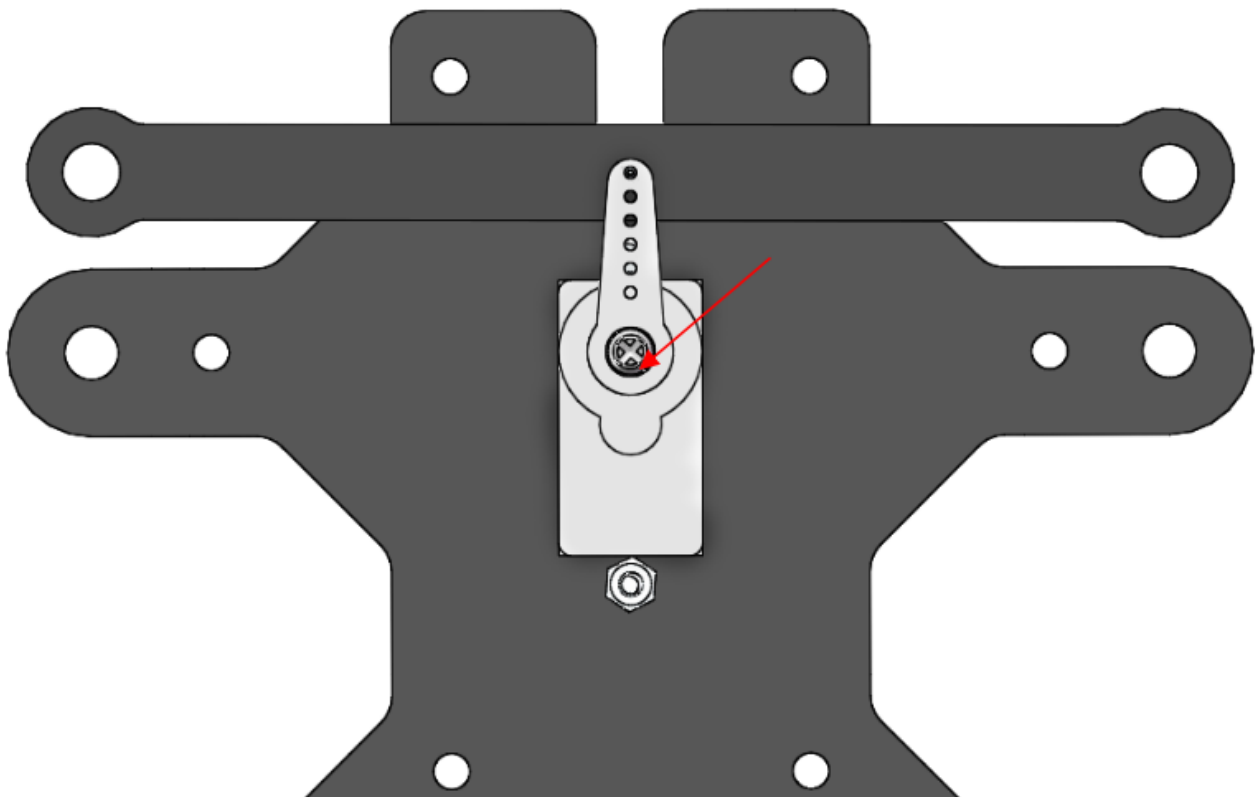
### 1.6.5 車の残り部分を組み立てる

警告：アセンブリのプロセス全体で、`servo-install` コマンドを実行し続けてください。

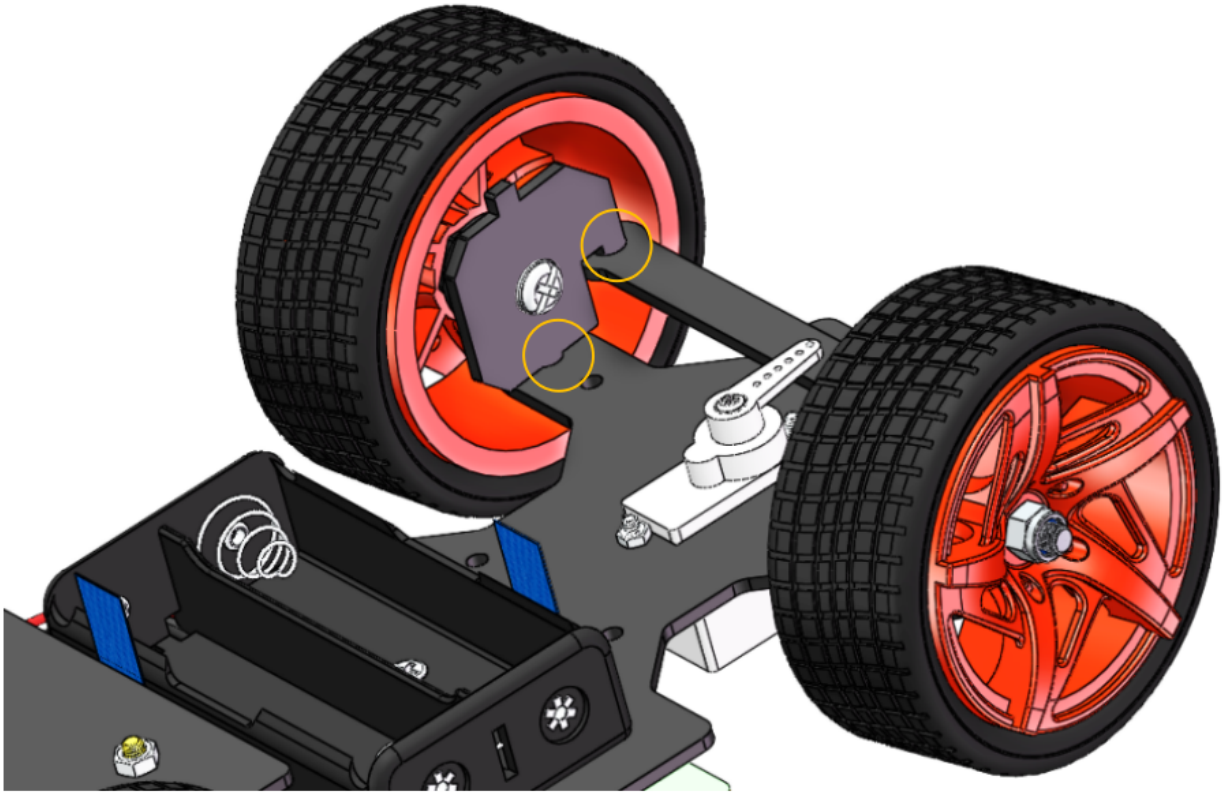
2 つの **M2x8 ネジ** と **M2 ナット** を使ってステアリングサーボを上部プレートに取り付ける（サーボワイヤーの方向に注意してください）：



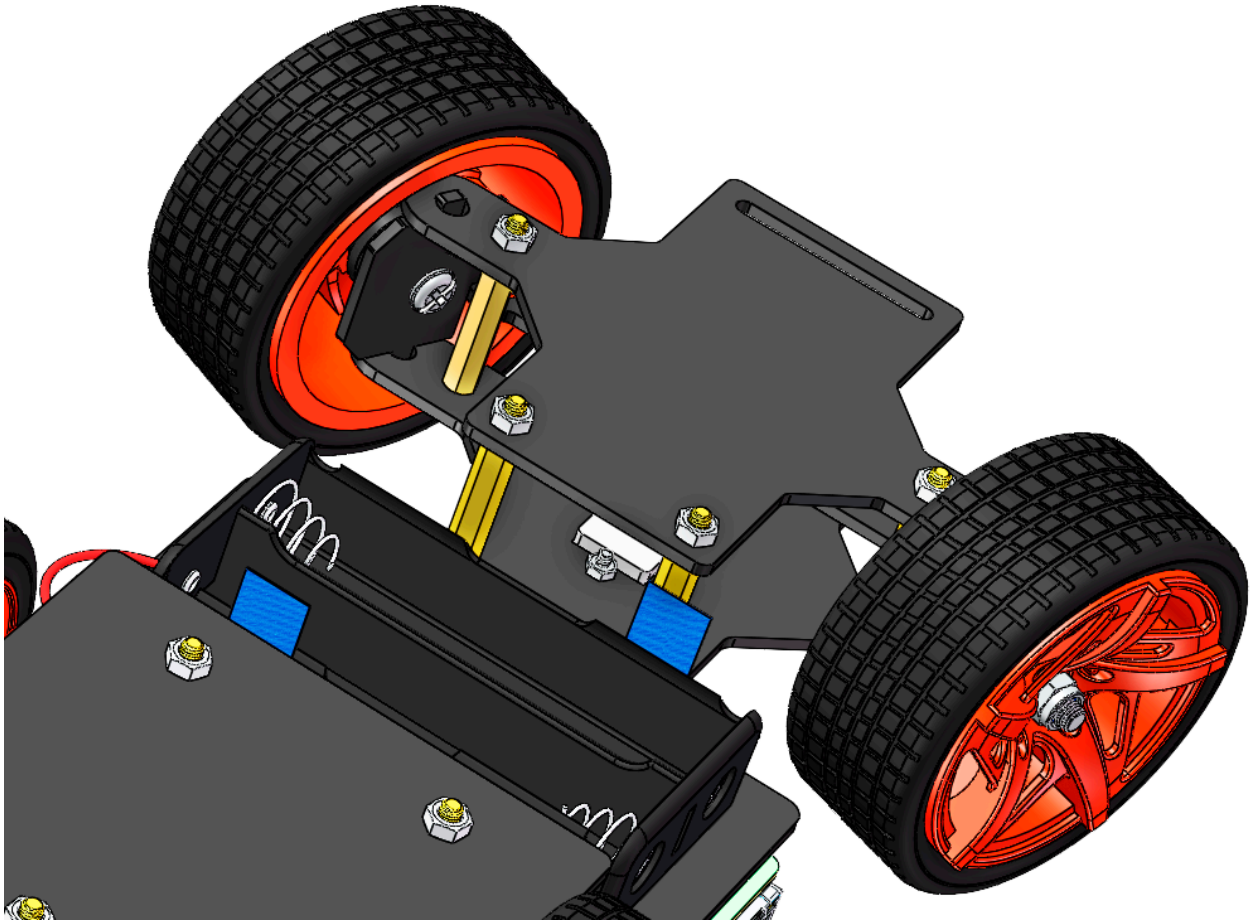
ロッカーアーム固定ネジ（最短）でステアリングリンケージをロッカーアームに接続する。



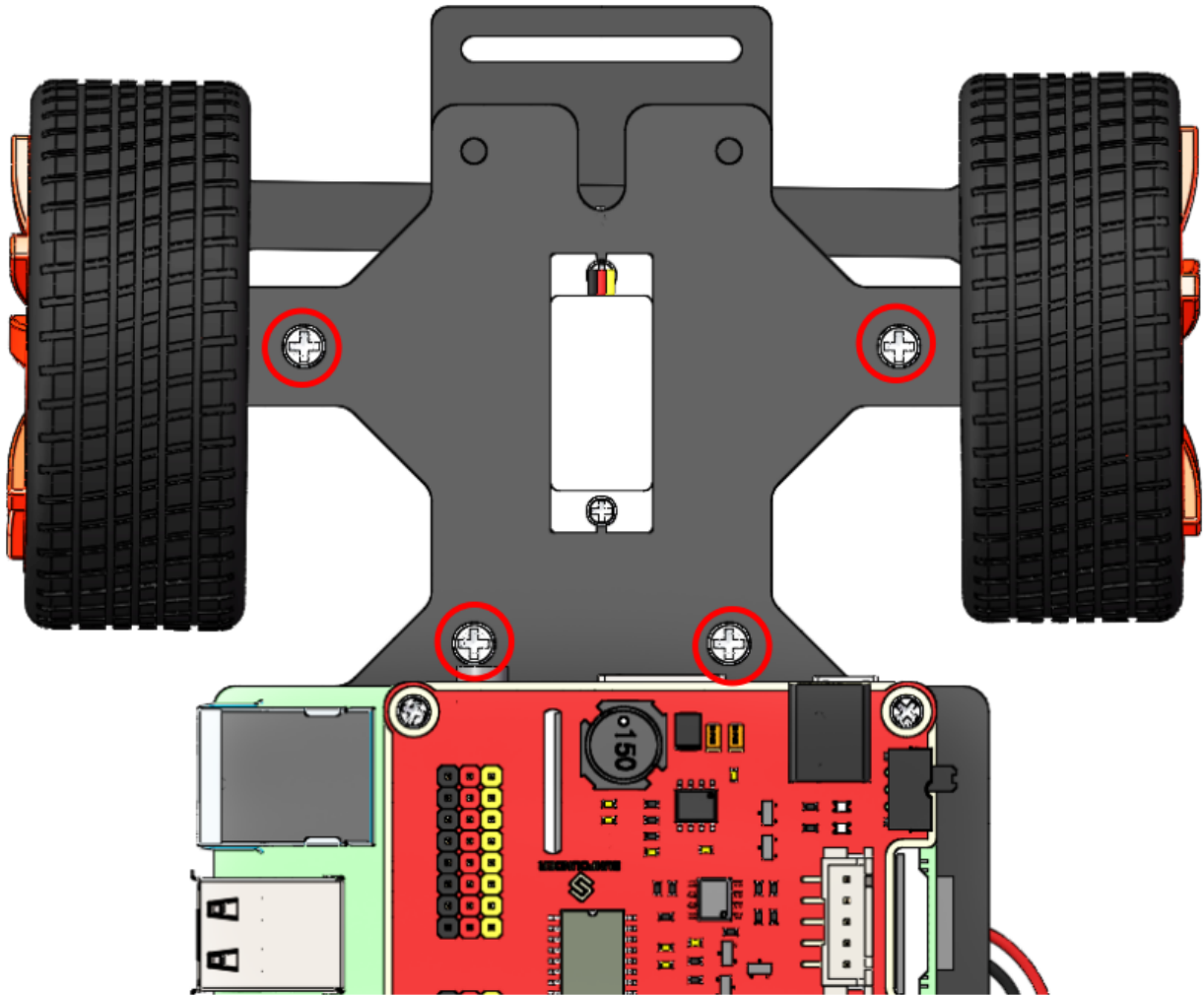
ホイールを上部プレートに入念に取り付ける。



次に、組み立てられたフロントハーフシャーシを、スタンドオフを穴に合わせている上部プレートに置く。



入念に持ち、上下を逆にして、4本の M3x8 ネジ でスタンドオフと上部プレートを固定する。



これで、アセンブリ全体が完了した。おめでとう！

## 1.7 補正

### 1.7.1 サーボを補正する

以前にサーボを 90 度に調整するコマンドを覚えているか？今回、他の 2 つのコマンドについて説明する。

2 番目のコマンド **front-wheel-test** は、組み立て後に前輪が柔軟に回転できるかどうかをテストするために使用される。このコマンドを実行すると、左右に駆動する。

```
picar front-wheel-test
```

```

pi@raspberrypi:~ $ picar front-wheel-test
('DEBUG "front_wheels.py":', 'Set debug off')
('DEBUG "front_wheels.py":', 'Set wheel debug off')
('DEBUG "Servo.py":', 'Set debug off')
turn_left
turn_straight
turn_right
turn_straight

```

前輪がまっすぐな状態では、前輪の方向が正確に前を向いていない場合がある。フロントシャーシの真ん中の線から明らかなずれがある場合は、サーボを組み立て直して、もう一度 `servo-install` を実行する。ほんの少しのずれ（約 0~15 度のような）であるならば、ソフトウェアによって調整することができる。

SunFounder\_PiCar/picar フォルダを開く：

```
cd /home/pi/SunFounder_PiCar/picar
```

```
sudo nano config
```

```

GNU nano 3.2                                config

# File based database

turning_offset = 0

forward_A = 0

forward_B = 0

```

エディターでフォルダの `config` ファイルを開く。いくつかのパラメーターが表示される。`turning_offset` の値は、前輪を調整するために使用される。デフォルトではその値は 0 となっている。前輪を少し右に回したい場合は、大きな値に変更してください。左寄りにするには、小さい値に設定する（負の数に設定できる）。

ただし、ホイールを過度に構成 **DO NOT** ください（-30~30 の推奨値）。そうしないと、サーボが動かなくなったり壊れたりする恐れがある。

Turning\_offset の値を変更した後、**Ctrl+O** を押して変更を保存し、**Ctrl+X** を押して終了する。`picar servo-install` コマンドを実行して、前輪のステータスを確認する。

```
picar servo-install
```

そして前輪が正面を向いていない場合は、ファイル `config` を数回編集する必要がある。通常 3~5 回程度調整する必要がある。前輪の補正が完了したら、後輪の補正に進む。

### 1.7.2 モーターを補正する

2つのDCモーターの配線はランダムであるため、モーターのVCCとGNDがホイールに逆に接続され、コードで構成されているように車輪が逆方向に回転すべきときに、正面に回転してしまう場合がある。したがって、後輪を交互に加速と減速を同時に行う3番目のコマンドを使用できる。

```
picar rear-wheel-test
```

```
pi@raspberrypi:~/SunFounder_PiCar/picar $ picar rear-wheel-test
('DEBUG "back_wheels.py":', 'Set debug off')
('DEBUG "TB6612.py":', 'Set debug off')
('DEBUG "TB6612.py":', 'Set debug off')
('DEBUG "PCA9685.py":', 'Set debug off')
('Forward, speed =', 0)
('Forward, speed =', 1)
('Forward, speed =', 2)
('Forward, speed =', 3)
('Forward, speed =', 4)
```

後輪の回転方向が画面と同じかどうかを確認してください。2つの車輪は2つのモーターによって別々に駆動されることに注意してください。一つが前方に回転し、もう一つが後方に回転する場合があります。その場合は、そのコマンドで逆回転する1つまたは両方のホイールを調整する必要があります。

```
cd /home/pi/SunFounder_PiCar/picar
```

```
sudo nano config
```

```
GNU nano 3.2 config
File based database
turning_offset = 0
forward_A = 0
forward_B = 0
```

**forward\_A** と **forward\_B** は2つのモーターのデフォルトの回転方向を変更する。値は0または1のみで、時計回りと反時計回りの回転を表す。デフォルトでは、両方のパラメーターで0となっている。したがって、車輪が逆回転する場合は、ホイールの対応するパラメーターを1に変更するだけである。

**Ctrl+O** を押して変更を保存し、**Ctrl+X** を押して終了する。

コマンド **picar Rear-wheel-test** を再度実行して、コマンドに従って後輪が回転しているかどうかを確認する。

```
picar rear-wheel-test
```

`config` を **PiCar-S** の下のディレクトリの **example** にコピーする。

```
cp config ~/SunFounder_PiCar-S/example
```

## 1.8 車を装備する！

センサーモジュールのない車は、視覚や聴覚のない人と同じように武装していないため、周囲の環境を感じない。だから我々がこれからすること、車を装備させて、周囲の物を検出できるようにすることである。次に、**PiCar** を **PiCar-S** に変える。

正確には **PiCar-S** とは何か？ -----データを収集して処理する機能を車に与えるいくつかのセンサーを **PiCar** に装備する。**PiCar** へのセンサーモジュールは、ゲームコンソールへのカートリッジである。それらはゲームの基本的なデザインに追加されるため、プレイが豊かになる。コードにも似ている。プロセッサは **SunFounder\_PiCar** を使用して車の動作を駆動し、さまざまなモジュール( **SunFounder\_Light\_Follower**、**SunFounder\_Line\_Follower**、**SunFounder\_Ultrasonic\_Avoidance** ) に対応するコードパッケージを読み出す。

以下の対応するモジュールの説明の配線に従って、希望のセンサーモジュールを組み立てる。トランスフォーマーを楽しんでください！





### 1.8.1 障害物回避

#### 動作原理

超音波障害物回避モジュールは収集したデータを検出して、障害物からの距離を計算できる Raspberry Pi に転送する。Pi は前輪と後輪の方向と回転を調整するコマンドを送信して、障害物がある場合は障害物から離れて PiCar-S を制御する。

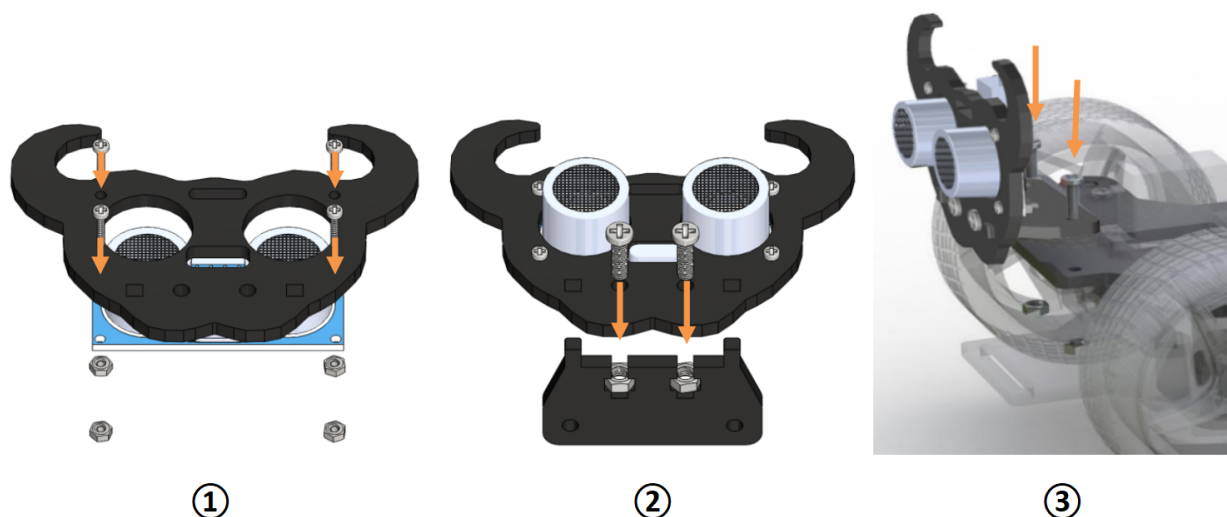
## 手順

### ステップ 1 組み立て

① 超音波モジュールと超音波コネクタを M1.4\*8 本のネジと M1.4 本のナットで接続します。

② 次に、超音波サポートに M3\*10 本のネジと M3 ナットで接続します。

③ 最後に、M3\*10 本のネジと M3 ナットでアッププレートに取り付けます。



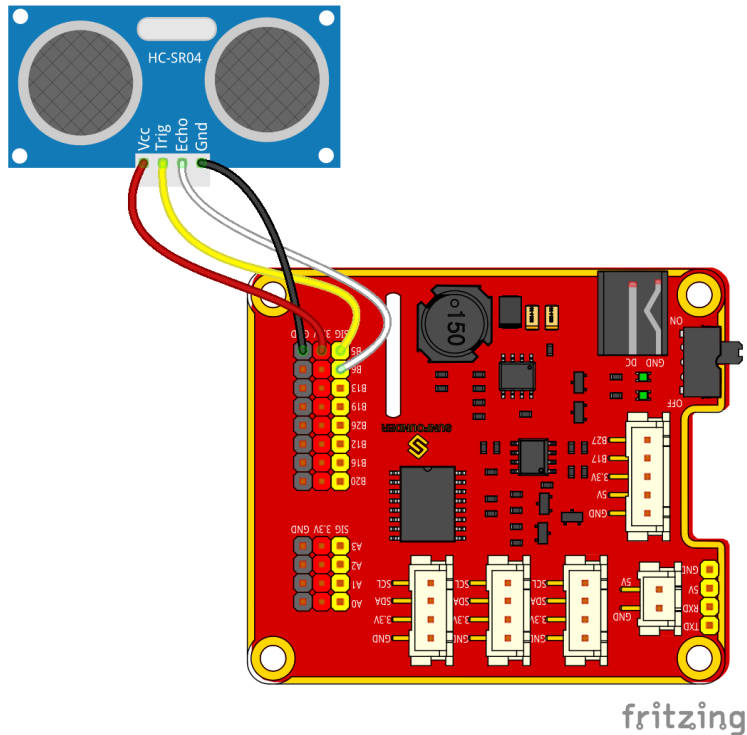
注意：指でナットをスロットに入れて下に保持する方が簡単です。

### ステップ 2 配線

以下に示すように、4 ピンのアンチリバースケーブルで超音波障害物回避装置をロボット HATS に接続する。

超音波モジュールは 5V または 3.3V の電源を持っていることがある。ここでは、3.3V 電源を与える。

Ultrasonic module	Robot HATS
Trig	B5
Echo	B6
VCC	3.3V
GND	GND



### ステップ3 テスト

適用する前に、まず超音波障害物回避モジュールをテストする。

```
cd ~/SunFounder_PiCar-S/example/
```

```
python3 test_ultrasonic_module.py
```

```
pi@raspberrypi:~/SunFounder_PiCar-S/example $ python3 test_ultrasonic_module.py
distance 4 cm
Less than 10
distance 4 cm
Less than 10
distance 4 cm
Less than 10
distance 4 cm
Less than 10
distance 4 cm
Less than 10
distance 5 cm
Less than 10
distance 6 cm
```

距離の測定がそれほど正確でない場合がある。大丈夫よ。この 25kHz の超音波モジュールは汎用のものではないが、約 30~40 度の水平検出範囲を持っている。したがって、測定された距離はそれほど正確ではないかもしれないが、その狭い範囲は障害物回避に便利である。さらに、Raspberry Pi はリアルタイムのオペレーティングシステムではないため、不正確な時間計算は距離測定の精度にも影響する。ただし、この超音波モジュールは障害物を回

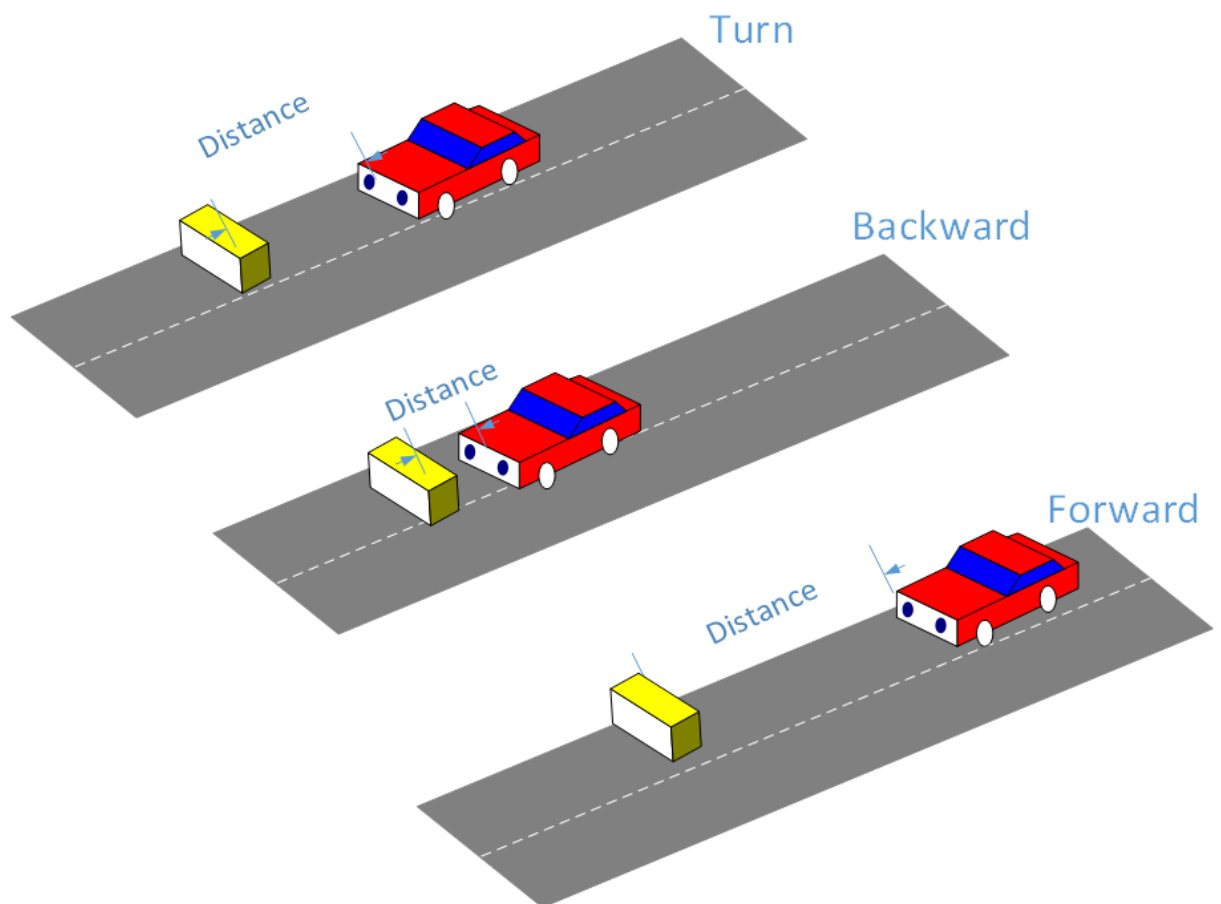
避することに十分である。

ステップ 4 旅に出よう！

これで、上記のテスト後の超音波モジュールの効果の概要が分かるだろう。超音波障害物回避のコードを実行してみよう。

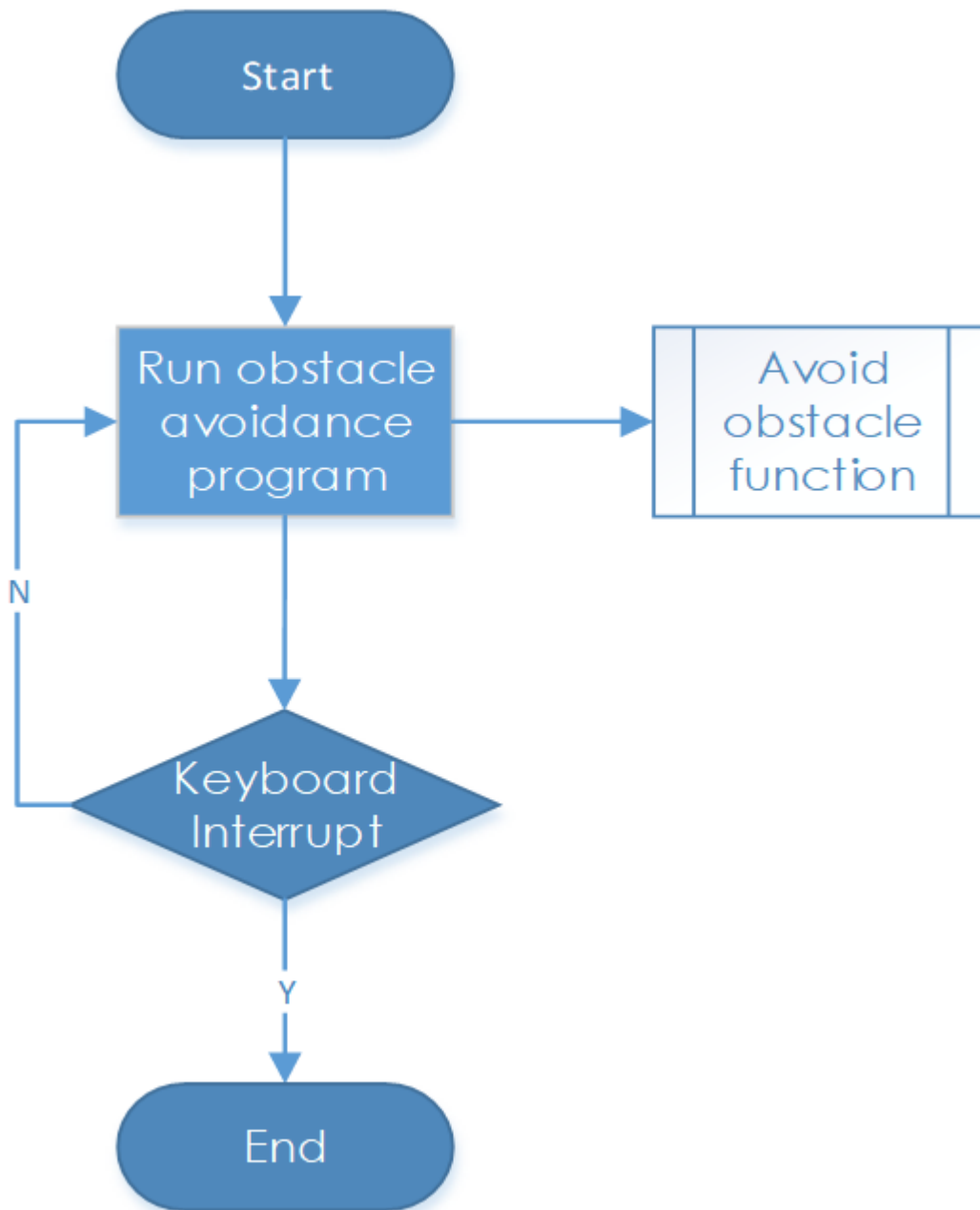
```
python3 ultra_sonic_avoid.py
```

今のところ、PiCar-S が起動した。車を地面に置くだけである。プログラムに従い、障害物を検出すると方向を転換する。障害物が近すぎる場合、後方に移動し、左/右に走行する。また、コードで障害物検出範囲のしきい値と後方に移動するしきい値を変更することもできる。



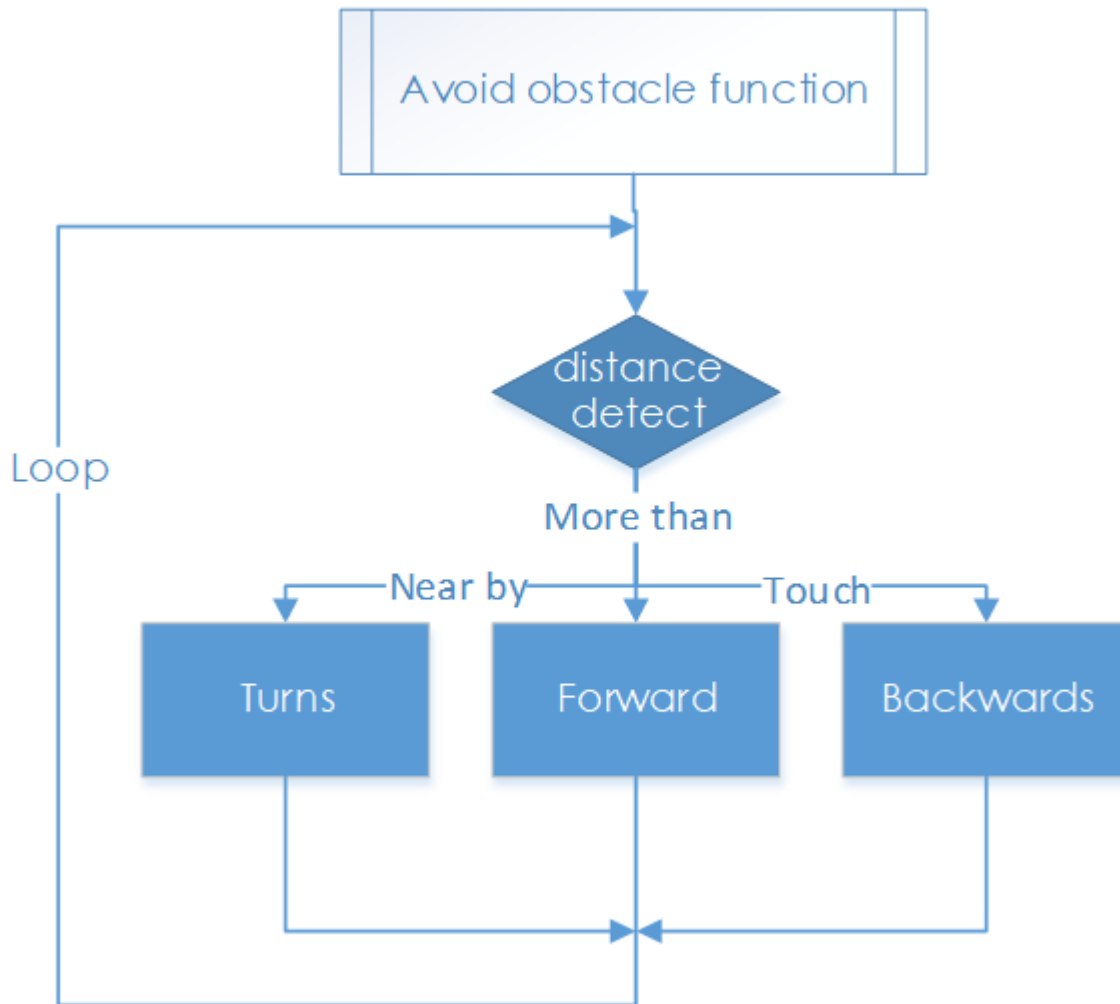
## ultra\_sonic\_avoid.py のコードの説明

## ワークフロー全体

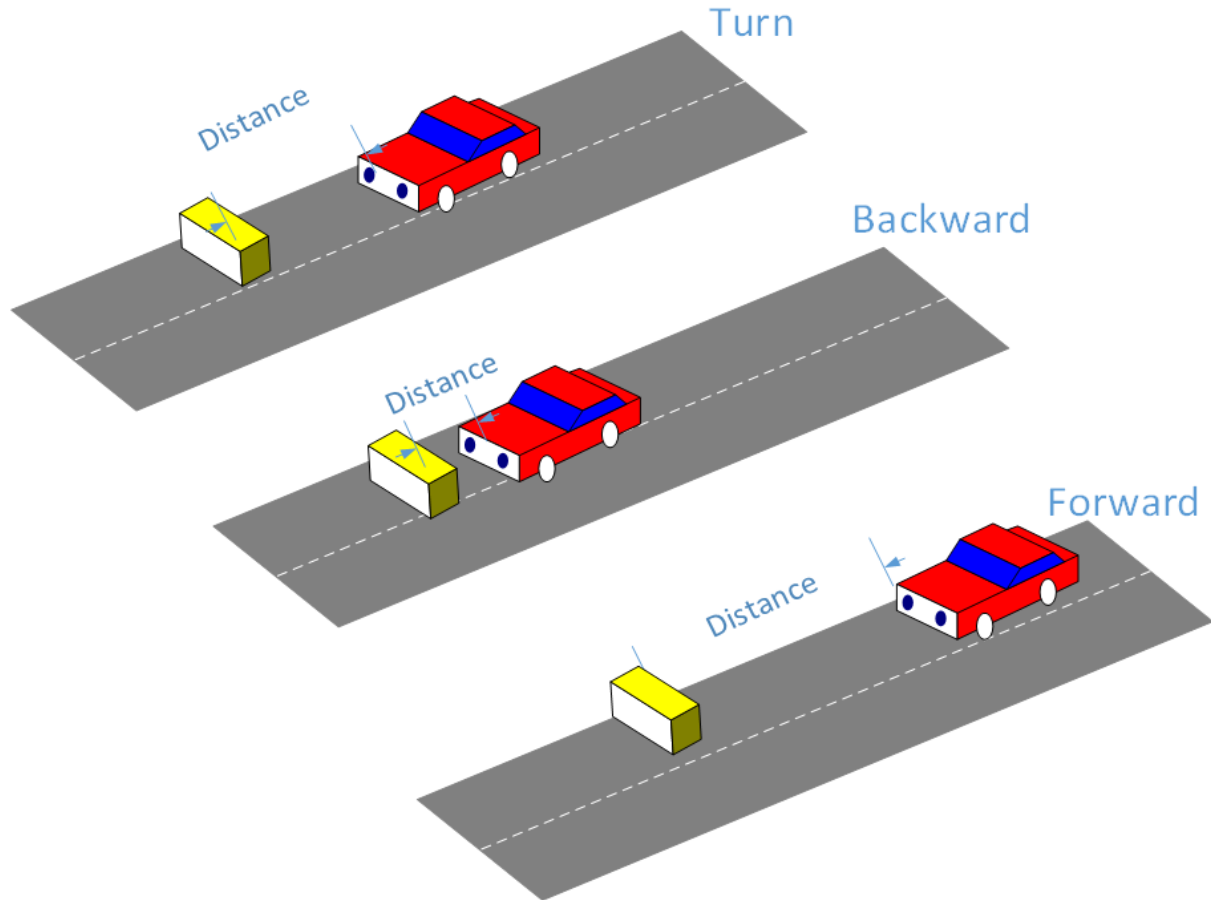


超音波モジュールはデジタル値、つまり高レベルまたは低レベルを返し、返された 2 つのレベル間の間隔時間を障害物までの距離に変換できる。したがって、ここではタイミングを求めるために Python で `time` モジュールを呼び出す。距離を計算する式は、超音波モジュールのドライバーに書き込まれる。メインプログラムは、対応するプログラムを呼び出して距離値を取得するだけである。

## 障害物回避機能のサブフロー



車が始動すると、障害物を検出して周期的に距離を測定したり、判断を下したり、行動を起こす。ここは三つのケースがある：障害物までの距離がしきい値と等しい場合、車は方向を変える。距離がしきい値を下回る場合、車は方向を変える前に後退する。距離がしきい値を超えると、続けて前進する。



### 機能説明

**ua = Ultra\_Sonic.UltraSonic\_Avoidance(17)**

Ultra\_Sonic モジュールで UltraSonic\_Avoidance クラスのオブジェクト **ua** を作成する。丸括弧内の数字は、モジュールの SIG が接続されているピン番号を表す初期パラメーターである。BCM 命名方法が適用されているため、Raspberry Pi の対応するピンは # 17 である。

**back\_distance** と **turn\_distance** の 2 つの定数は測距距離のしきい値を設定するためのものである。

**while()** loop

検出された距離が **back\_distance** より小さい場合、車は後方に移動する。距離は **back\_distance** と **turn\_distance** の間にある場合、車は方向を変える（前述のパラメータ **turning\_angle** で回転角度を設定でき、角度は正または負の数で、それぞれ左または右に曲ることができる。サーボの最大回転角度を考慮すると、回転角度は **-90 ~ 90 度**であることを注意してください。そうしないと、サーボが焼ける恐れがある。）検出された距離が **turn\_distance** より大きい場合、車は続けて前進する。

**bw.backward()**、後輪を後方に回転させる。**bw.forward()**、後輪を前方に回転させる。後輪駆動モジュール **back\_wheels** のこれら 2 つの機能は車輪の回転方向を設定するために使用される。

**bw.set\_speed(speed)**、**back\_wheels** の関数で、ホイールの回転速度を設定します。数値が大きいほど (0 ~ 100 の範囲内)、ホイールの回転が速くなります。

**fw.turn(angle)**、**back\_wheels** の関数で、回転角度を設定します。車がまっすぐ進むときの角度は 90 度です。数を減らして左に曲がり、数を増やして右に曲がります。

**fw.turn\_straight()**、前輪を直進する角度に戻す。

詳細：

**back\_distance** と **turn\_distance**

定数を変更して、車をオフに戻し、障害物回避中に希望の距離と角度で離れるようにしてください。

### 1.8.2 ライトフォロワー

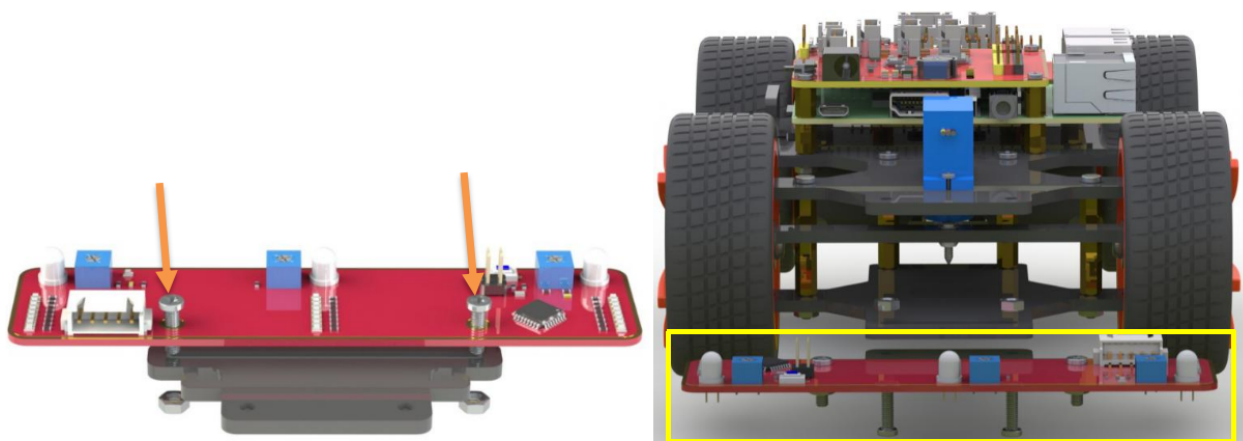
#### 動作原理

ライトフォロアモジュールは周囲の光源を検出し、データをプロセッサに転送する。プロセッサはデータを分析し、光源の方向を見つけるため、前輪と後輪の動きを制御してリソースにアプローチするコマンドを送信する。

#### 手順

##### ステップ 1 組み立て

M3\*10 ネジと M3 ナットを使用してライトフォロアをセンサーコネクタに接続し、2 つの M3\*10 ネジと 2 つの M3 ナットを使ってそれらを車に組み立てる。指でナットを下に持ってください。



##### ステップ 2 配線

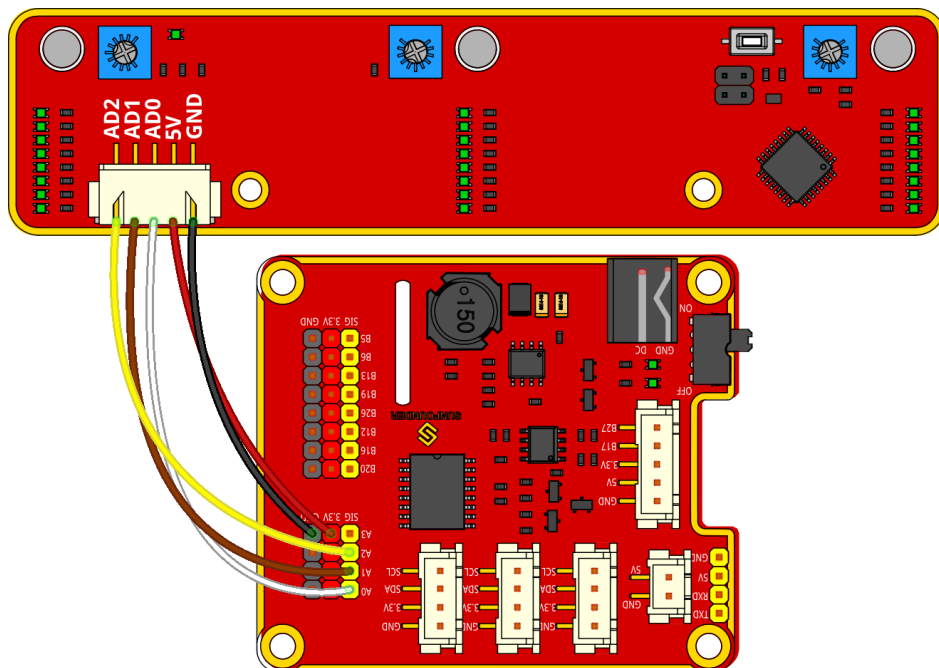
以下に示すように、5 ピンのアンチリバースケーブルでライトフォロアをロボット HATS に接続する。



Light Follower	Robot HATS
AD2	A2
AD1	A1
AD0	A0
5V	3.3V
GND	GND

注釈: なぜ 5V を 3.3 に接続するのか質問を持っているかもしれない。さて、ライトフォロア上の STM8 チップの動作電圧は 2.7-5.5V なので、ここで 3.3V に接続できる。5V を 5V に接続しないでください！ Robot HATS のすべてのアナログポートは 3.3V で動作する PCA8591 から導かれる。したがって、電圧が 3.3V ~ 5V の場合、出力値は常に 255 になるため、5A に接続すると PCA8591 が損傷する恐れがある。必ず **3.3V** に接続してください。

配線は次の通りである：



fritzing

### ステップ3 テスト

まずライトフォロワーをテストしよう。

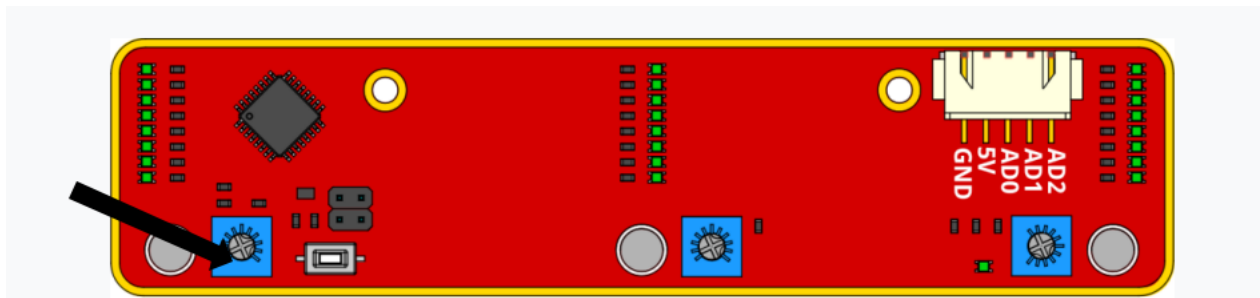
```
cd ~/SunFounder_PiCar-S/example/
```

```
python3 test_light_module.py
```

```
pi@raspberrypi:~/SunFounder_PiCar-S/example $ python3 test_light_module.py
a0 = 200      a1 = 220      a2 = 204
a0 = 201      a1 = 221      a2 = 204
a0 = 200      a1 = 220      a2 = 204
a0 = 200      a1 = 220      a2 = 204
a0 = 201      a1 = 220      a2 = 204
a0 = 201      a1 = 221      a2 = 205
a0 = 201      a1 = 221      a2 = 205
a0 = 201      a1 = 220      a2 = 204
a0 = 201      a1 = 220      a2 = 204
a0 = 201      a1 = 221      a2 = 204
a0 = 201      a1 = 221      a2 = 205
a0 = 201      a1 = 221      a2 = 204
```

フォトランジスタを懐中電灯の光点に当てる。光量を増やすと、より多くの LED が点灯し、出力値が減少する。

ここで、青色の調整可能な抵抗器を回転させて、同じ光の輝度で値を変更できる。最良のステータスは次のとおりである：



- 1) LED が 1 つしか点灯していない場合、出力値は 255 である。
- 2) ライトが最も明るく、すべての LED が点灯する場合、出力値は約 10 ~ 25 である。

ステップ 4 旅に出よう！

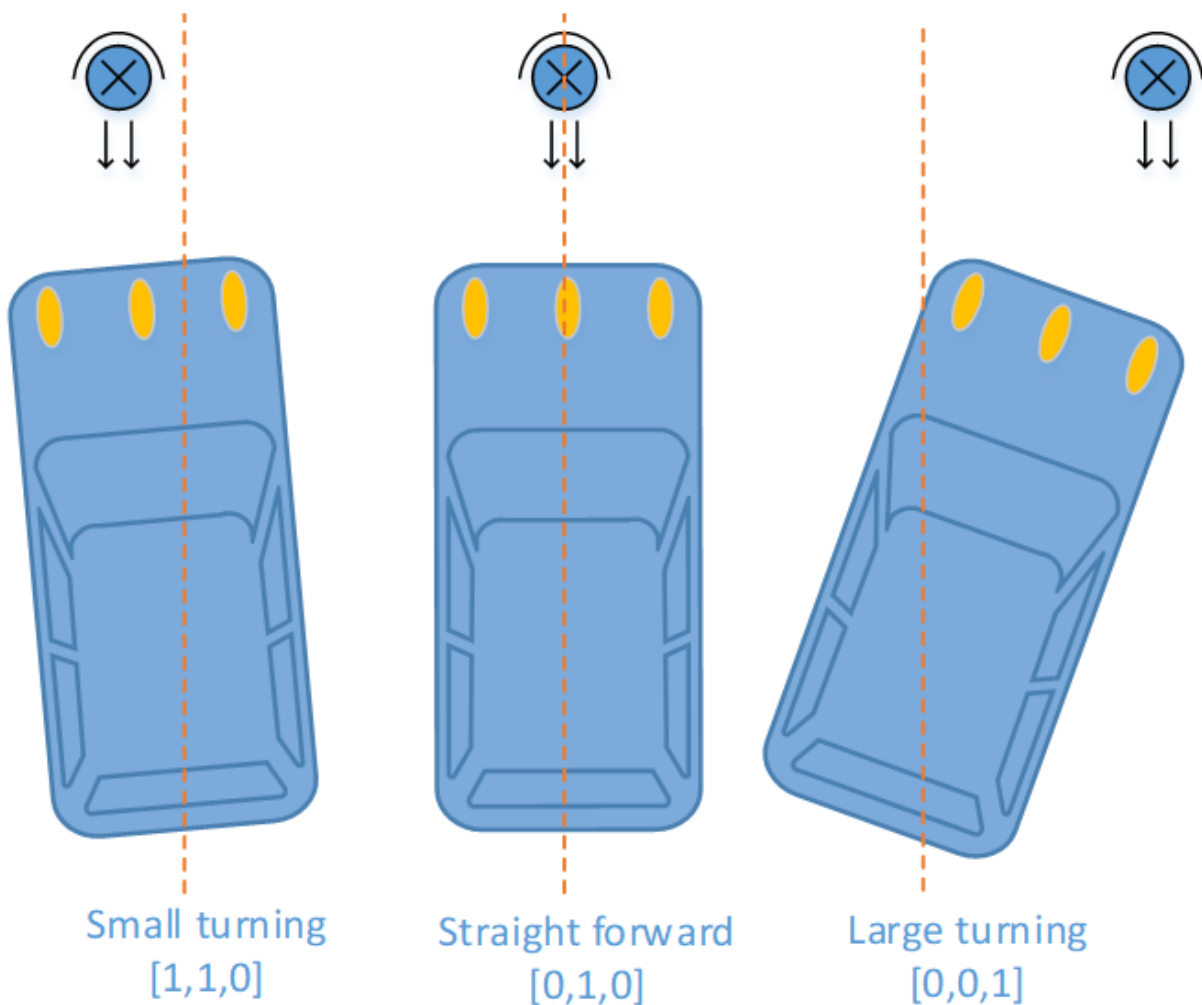
```
git submodule update --init
```

```
python3 light_follower.py
```

```
^Cpi@raspberrypi:~/SunFounder_PiCar-S/example $ python3 light_follower.py
DEBUG "front_wheels.py": Set debug off
DEBUG "front_wheels.py": Set wheel debug off
DEBUG "Servo.py": Set debug off
DEBUG "back_wheels.py": Set debug off
DEBUG "TB6612.py": Set debug off
DEBUG "TB6612.py": Set debug off
DEBUG "PCA9685.py": Set debug off
calibrating.....
calibrate 1
calibrate 2
calibrate 3
calibrate 4
```

上記のコードを実行すると、車は次の light following モードに入る。円を描くように右に曲がり続け、さまざまな方向の光の状態に関する情報を収集する。車をより広い場所に置いて待ってください。

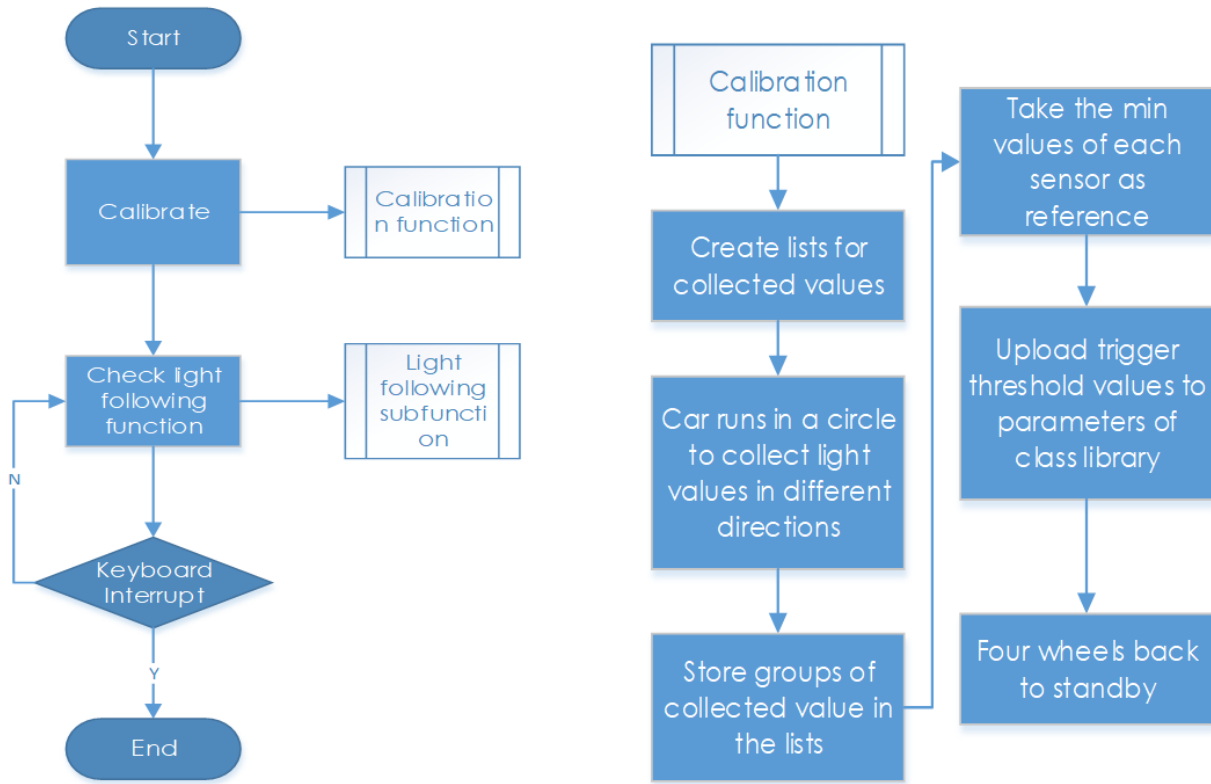
補正が完了すると、車は一時停止する。ライトフォロアモジュールの電灯を照らすと、車は移動するときにライトスポットに追従する。



## - light\_follower.py のコードの説明

### ワークフロー全体

環境の光条件が複雑なので、感光センサーは実際に使用する前に補正する必要がある。環境光の輝度の情報を収集する。光源が周囲より明るい場合のみ、車は光に追従できる。



ここでは、補正後のライトとメインプログラム内のライトフォローを含む 2 つの主要な関数/モジュールを書き込む。

### ライトフォロア補正機能のサブフロー

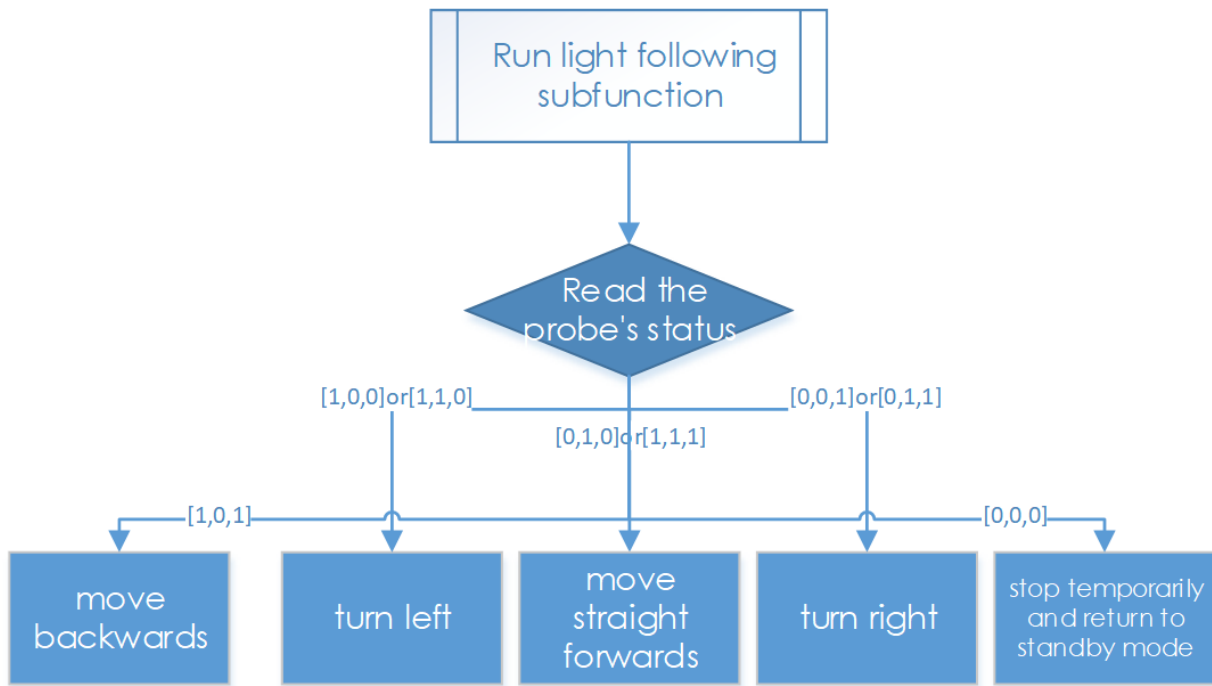
3 つの感光性コンポーネントを個別に構成する必要があるため、何回に収集された A0、A1、と A2 に値を保存する 3 つのリストを設定する。次に、最も明るい条件での出力アナログ値である最小値を選ぶ。

使用する光源は環境光よりもはるかに明るいいため、最も明るい条件での出力値を参照として使用してください。

さらに、しきい値を設定する必要がある-収集された光源の値と環境の値のギャップがしきい値を超えた場合、値を 0 または 1 に設定する。ここでは、[0,0,0] を使用して、トリガーされていないときの 3 つのフォトレジスタのステータスを表す。対応するフォトレジスタの検出値がしきい値よりも高い場合、「0」は「1」になる。したがって、3 つの要素のリストに従って車の関連アクションを設定できる。

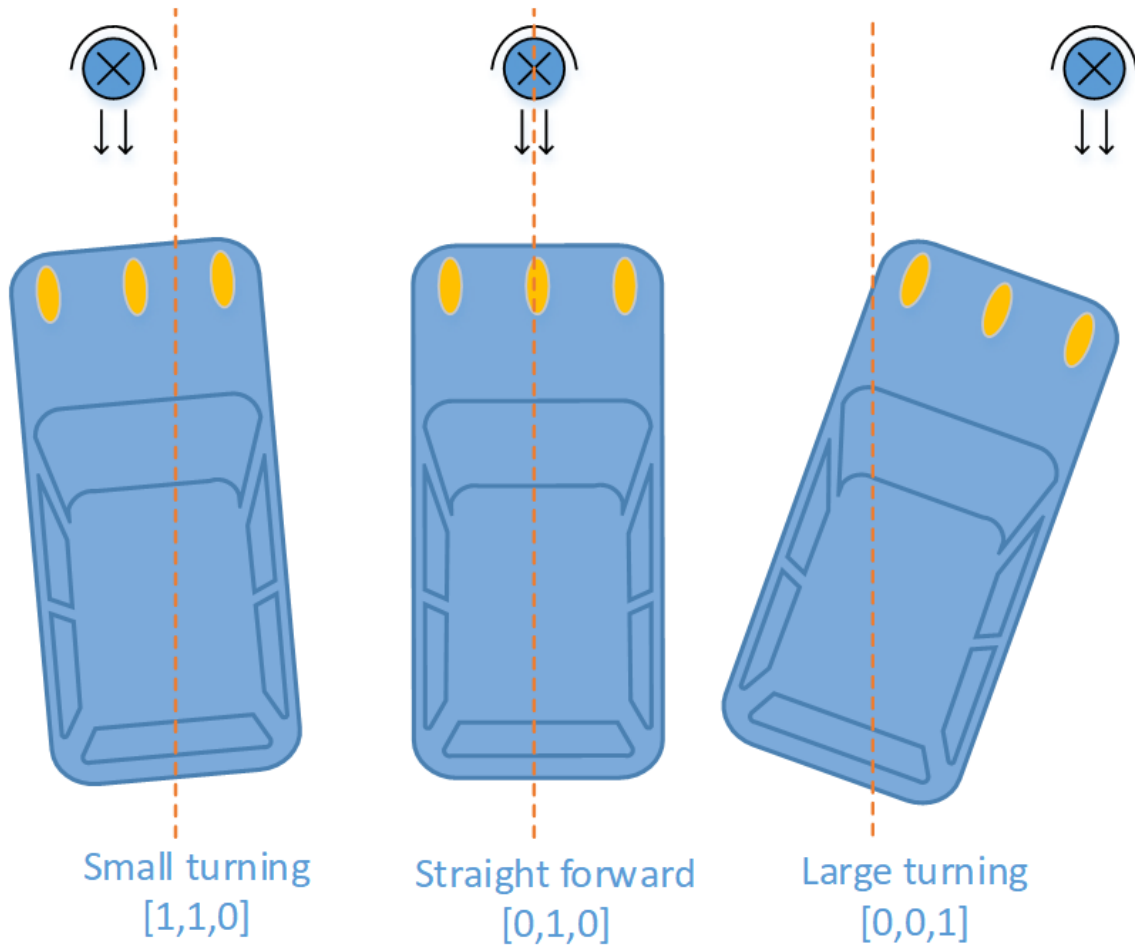
光が検出された場合、車は動き、それに追跡する。光が検出されない場合、車は一時的に停止し、旋回を続けて円を検出する。

## ライトフォロアー機能のサブフロー



ライトフォロワーには3つのフォトトランジスターが含まれるため、そのステータスリストは8つのステータスを表す3つの要素で構成される（順列と組み合わせに基づく）。ここでは、これらのステータスに関連する応答を設定する必要がある。

3つの要素は3つのプローブのステータスを示す：1は検出された光を表し、0は検出されなかったことを表す。たとえば、[1,0,0]は光が左側のプローブによってのみ検出されることを示し、つまり、光源が車の左側にあり、車の応答アクションを左折として設定する。[1,1,0]は光が左側と中央のプローブで検出されることを意味するため、その応答アクションも左折に設定する必要がある。対応するステータスに従って同様に右折するように設定する。光が検出されない場合、ステータスは[0,0,0]であるため、応答アクションを停止してスタンバイモードに戻るよう設定する。



ここでは、大角度と小角度の回転を区別するために、別の変数（ステアリング角度）を設定してください。ライトが左側の中心にある場合（ステータス [1,1,0]）、小角度の回転を適用する必要がある。ライトが左側の端にある場合（ステータス [1,0,0]）、大きな角度の回転を適用する。

#### 機能説明

コードを理解するには、上記のソフトウェアサブフローを参考にしてください。

以前にインポートされた `light_follower_module`、`front_wheels`、と `back_wheel` を含む 3 つの Python モジュールがコードに使われる。これらはこのキットのドライバーであり、それぞれライトフォロー、前輪と後輪となっている。

関連クラスはここで定義されている。モジュールを適用して使用すると、関連するクラスのオブジェクトが作成され、ハードウェアのさまざまな部分がクラスオブジェクトによる関数を呼び出すことによって駆動される。

たとえば、ライトフォローモジュールの場合、`If` という名前のオブジェクトを作成する：

```
If = Light_Follower.Light_Follower()
```

それからクラスオブジェクトによって関数を呼び出す。

```
A0 = lf.read_analog()[0]
```

この関数 `read_analog()` は、3 つのプロープの検出されたアナログ値を保存する 3 つの要素のリストを返す。ここでは、`A0=lf.read_analog()[0]`, `A1=lf.read_analog()[1]` と `A2=lf.read_analog()[2]` を使用して、戻り値の 3 つの要素を変数 A0-A2 に個別に保存する。

ここでは `for()` ループを 10 回使用する。つまり、補正モードで車が円を描くように走行すると、車はアナログ値を 10 回取得する。ここでは、最小値を参考している。より多くのサンプルが必要な場合は、ループの回数を増やしてください。

`env0_list.append(A0)` 関数によって、検出された値を各ループのリストに保存する。ループが終了すると、Python の組み込みリスト関数 `reference[0] = min(env0_list)` がリストの最小値を選ぶ。

```
lt_status_now = lf.read_flashlight()
```

この目的は 3 要素のリストを返すモジュールのステータスを読み取ることである。この機能は輝度が調節可能な懐中電灯によって引き起こされる問題を解決するために使用される。PWM 方式による輝度変化で繰り返し点滅するので、この機能をドライバーライブラリーに追加し、光源が急に消灯したり、ON/OFF の比率で輝度が変化したりしても、車が何度も動いたり止まったりしないようにする。

```
fw.turn(turning_angle)
```

前輪ステアリング機能。前輪がステアリングに適用されている場合、メインプログラムはこの関数を呼び出す。パラメータは回転角度となっている。

```
bw.forward()
```

```
bw.set_speed(forward_speed)
```

後輪には 2 つの機能が必要である。最初の関数は回転方向を順方向に制御する（逆方向の関数は `bw.backward()` である）。2 つ目は車輪の回転速度を設定すること。パラメータは速度の値である（範囲：0-100）。パラメータが大きければ大きいほど、ホイールの回転が速くなる。

### 1.8.3 ラインフォロー

#### 動作原理

ラインフォロワーは周囲の環境でラインを検出し、データをプロセッサに転送する。プロセッサはデータを分析し、前輪と後輪の動きを制御するコマンドを送信する。

### 手順

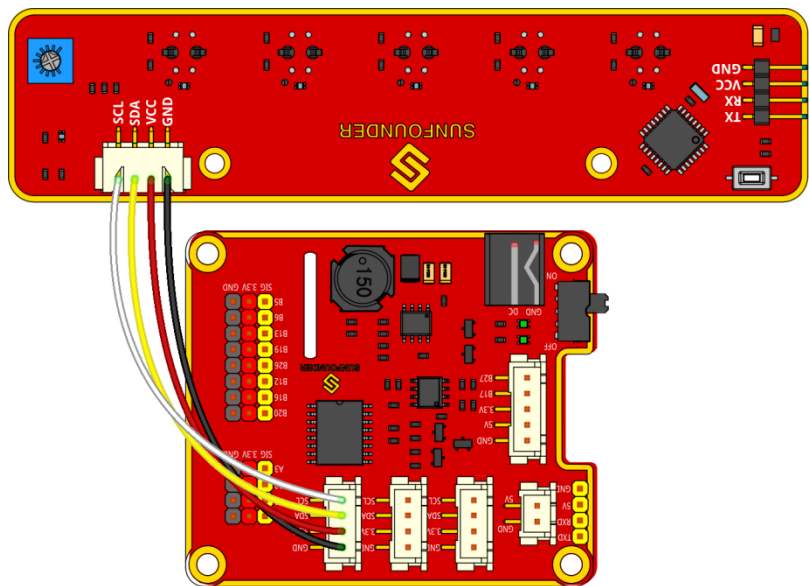
#### ステップ 1 組み立て

ライトフォロアを M3\*10 ネジ と M3 ナット でセンサーコネクタに接続し、2 つの M3\*10 ネジ と 2 つの M3 ナット で車に組み立てます。ナットを指で下に保持することをお勧めします。



#### ステップ 2 配線

以下に示すように、4 ピンのアンチリバースケーブルでラインフォロアモジュールをロボット HATS に接続する。



fritzing

#### ステップ 3 テスト

ディレクトリの例をチェックする：



```
cd ~/SunFounder_PiCar-S/example
```

i2c-tools を介して i2c デバイスが認識されているかどうかを確認する

```
sudo i2cdetect -y 1
```

```
pi@raspberrypi:~/SunFounder_PiCar-S/example $ cd ~/SunFounder_PiCar-S/example
pi@raspberrypi:~/SunFounder_PiCar-S/example $ sudo i2cdetect -y 1
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  -- 11  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
40: 40  --  --  --  --  --  --  -- 48  --  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
70: 70  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
pi@raspberrypi:~/SunFounder_PiCar-S/example $
```

11 がラインフォロワーの i2c アドレスであることが判明した。表示されない場合は、配線が正しくなく、Raspberry Pi との i2c 通信も失敗したことを示す。次のステップの前に配線を確認する必要がある。

テストコードを実行する。

```
python3 test_line_module.py
```

```
pi@raspberrypi:~/SunFounder_PiCar-S/example $ python3 test_line_module.py
[295, 297, 300, 297, 298]
[1, 1, -1, 1, 1]

[296, 297, 300, 297, 298]
[1, 1, -1, 1, 1]

[296, 297, 300, 297, 298]
[1, 1, -1, 1, 1]

[295, 297, 300, 297, 298]
[1, 1, -1, 1, 1]

[296, 296, 300, 297, 298]
[1, 1, -1, 1, 1]
```

注釈: ラインフォロワーモジュールをより適切に機能させるために、感度を調整してください。手順は次のとおりである:

モジュールを白い表面に置き、値を読み取る。白い表面に置き、値を読み取る。

差を計算し、最大値になるまで、次のモジュールのポテンショメーターを時計回りと反時計回りに回転させる。これでデバッグは完了した。

ステップ 4 実行を開始する！

ラインフォロワーコードを実行する

```
python3 line_follower.py
```

プログラムの実行が開始されると、補正のヒントが画面に表示される。まずは白い表面でモジュールを補正する。ラインフォロアの 5 つのプロープすべてを白いボードの上に配置する。完了した補正のプロンプトが数秒後に画面に表示される。次に、黒い線の補正に進む。また、開始のプロンプトが画面に表示され、すべてのプロープが黒い線の上に配置される。完了した補正のプロンプトが数秒後に画面に表示される。

モジュールの補正がすべて完了すると、車を走らせることができる。プロープ付きの PiCar-S をホワイトボードの黒い線の上に置くと、線自体に従って進む。

### ラインフォロアの追跡を作る方法

車が黒い線をたどるように追跡を作成するには、次の材料を準備しなければならない：

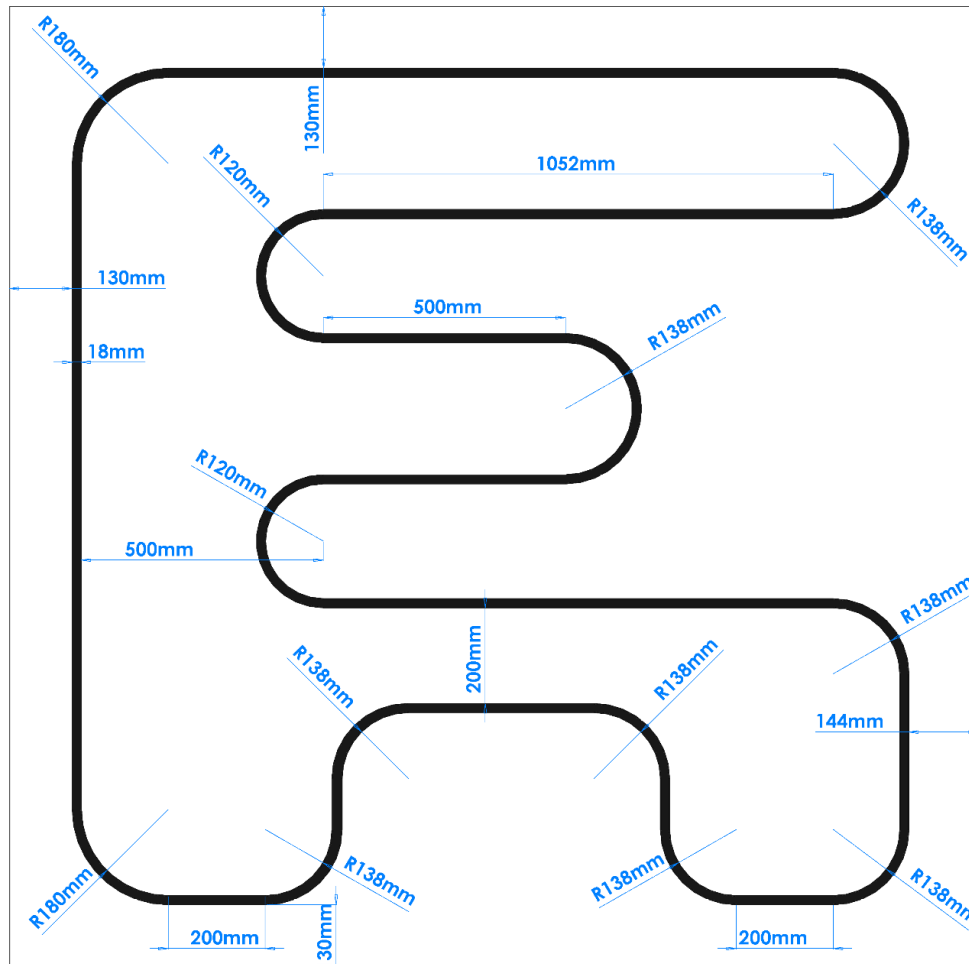
大きな紙、黒いテープのロール（黒い線）、ハードカードボード（サイズはトラックのサイズによって異なる）、または床や机などの平らな面。

1. 紙をハードボードに滑らかに広げ、ボードまたは平らな面に貼り付けます。
2. テープを紙に貼り付ける。

作成のルール：

1. 黒い線の幅：約 18～30mm、2 つのプロープ間の距離、2 つの隣接していないプロープの最小距離以下
2. 2 本の線の間隔：モジュール全体の幅である 125mm 以上で、2 本の線を同時に検出したときに車が混乱することを防ぐ。
3. 曲線の直径：138mm 以上。前輪が左または右に 45 度回転する場合、車が曲がる経路の半径はホイールベース（前輪の中心と後輪の中心間の距離）と等しくなる。カーブの半径が小さすぎると、車はカーブをスムーズに曲がったり、通過したりことはできない。

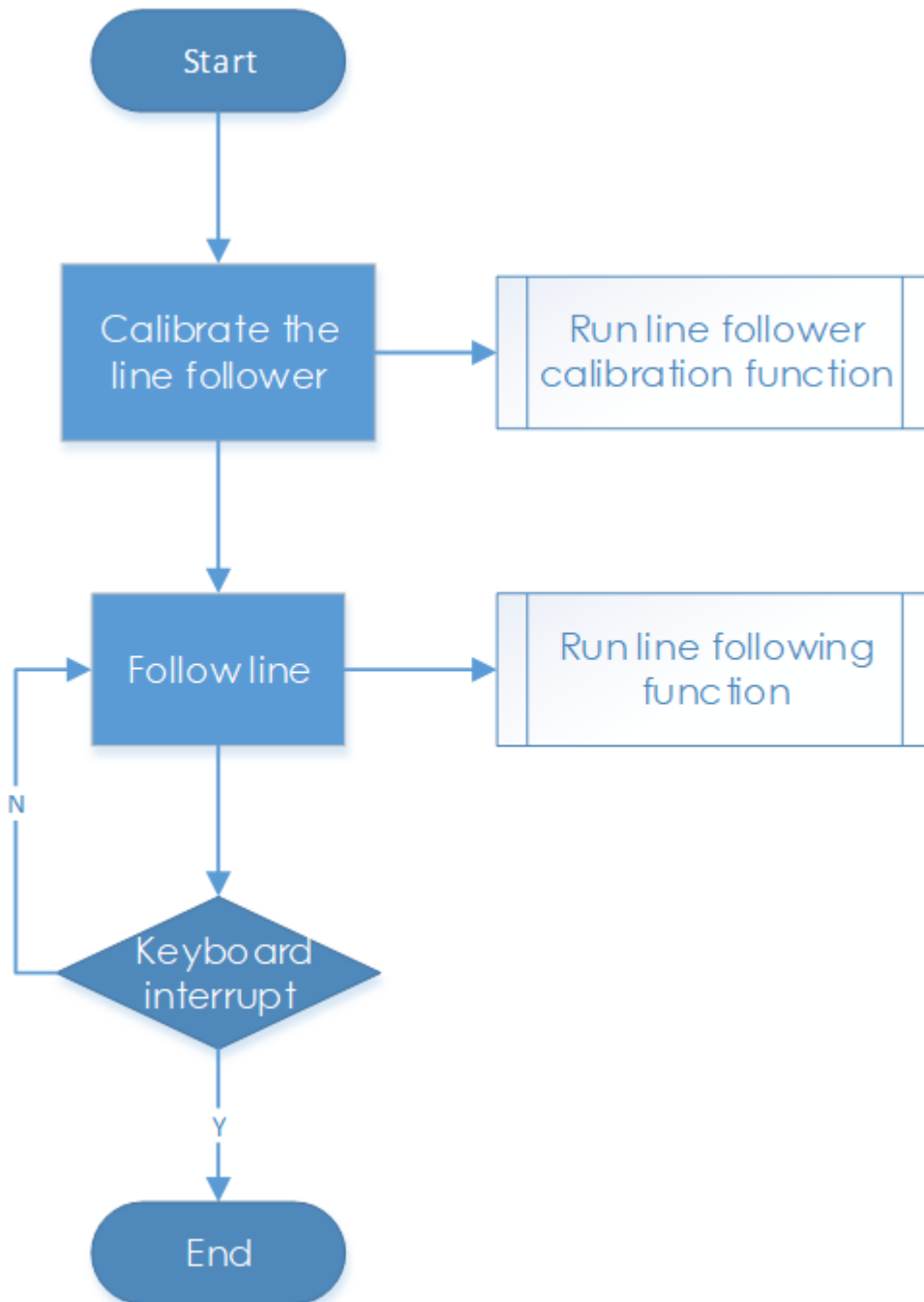
追跡のサンプルを以下に示す（元のマップファイルは [github](#) のフォルダー マップ の下にある）：



### line\_follower.py のコードの説明

#### ワークフロー全体

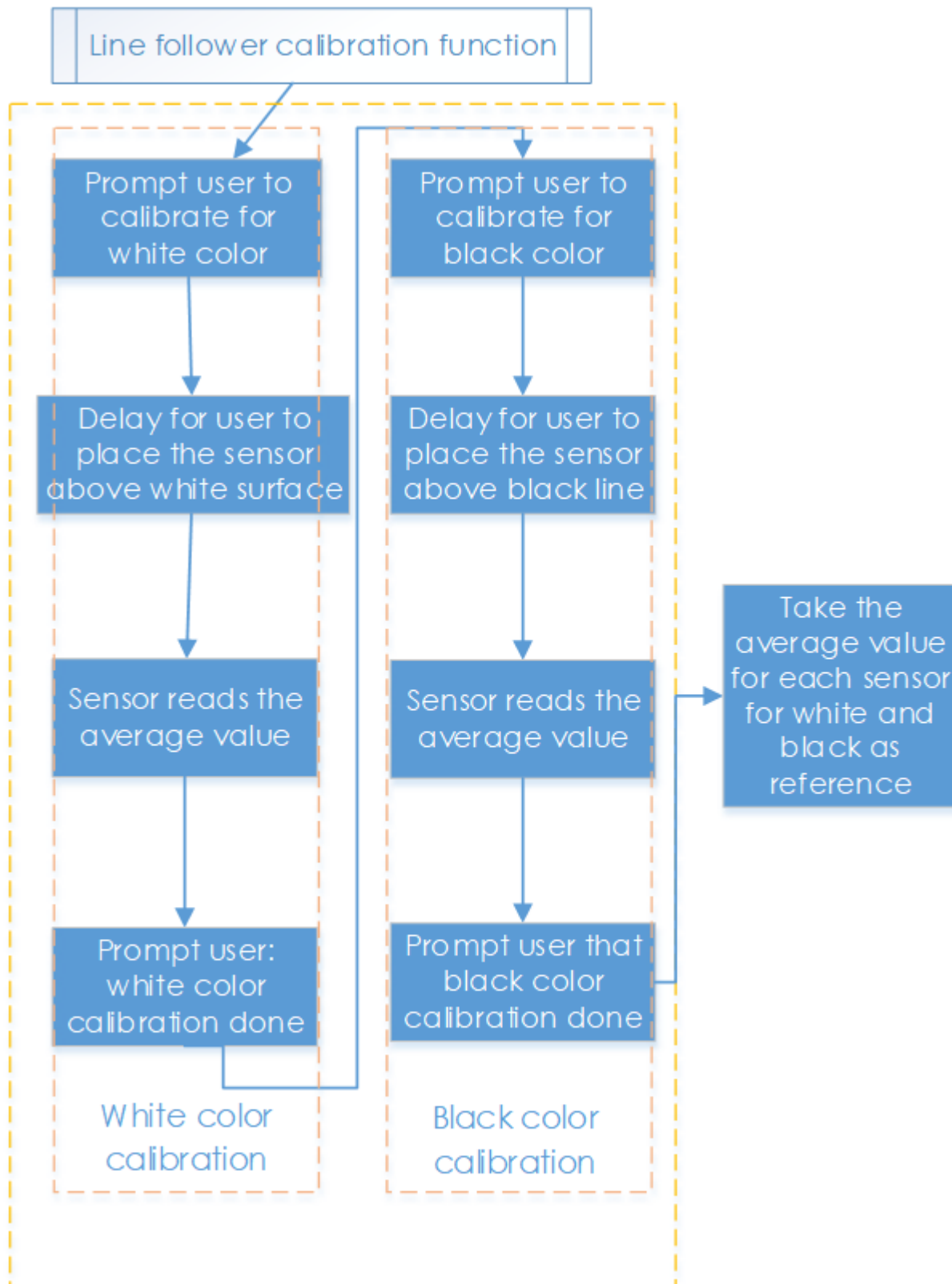
マイナスの環境要因の干渉を考慮して、実際に使用する前にラインフォロワーセンサーを補正しなければならない。



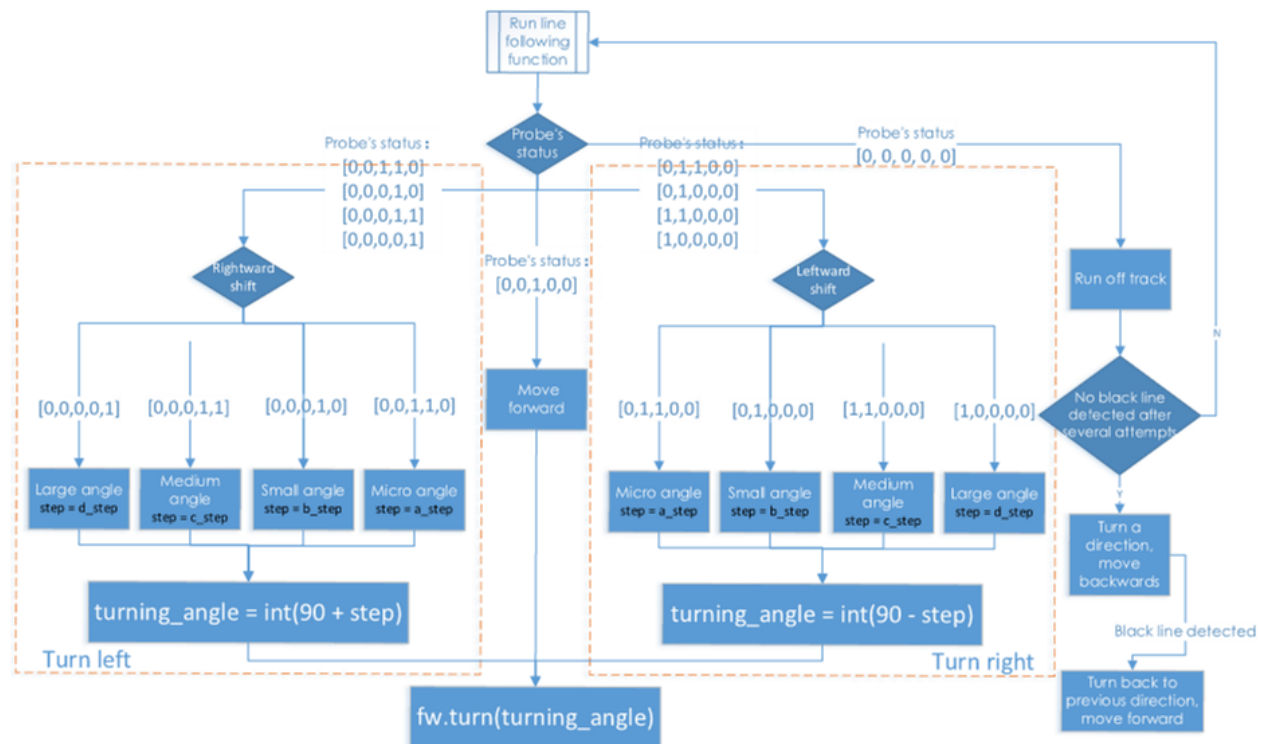
ここでは、ラインフォロワー補正とラインフォロイングを含む 2 つの主要な機能がメインプログラムに含まれている。

#### ラインフォロワー補正機能のサブフロー

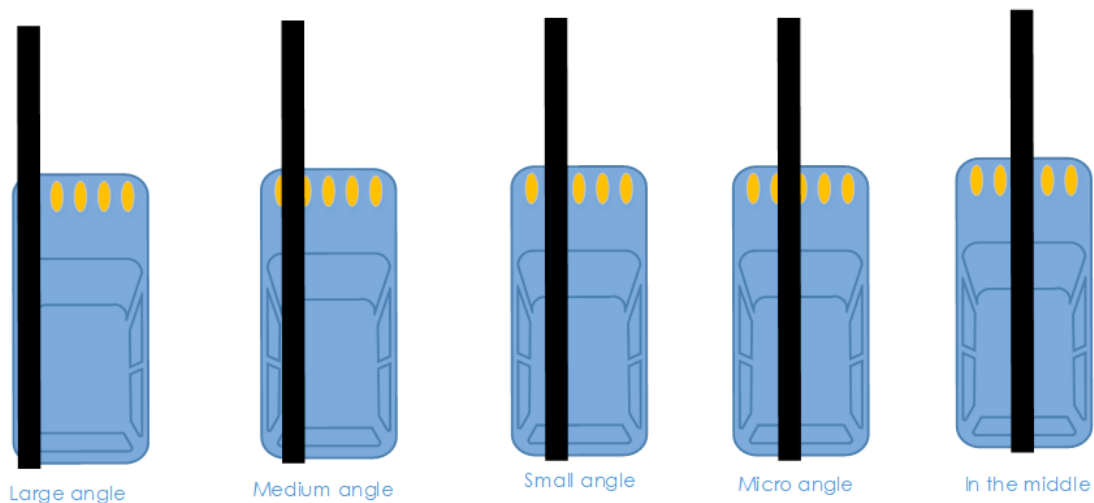
ラインフォロア構成を実行するときは、センサーの上限と下限に近い、白色、次に黒色から始める。それから黒と白の平均値を基準値として使用し、検出された値が基準よりも高い場合、白でなければならない。検出された値が参照よりも低い場合は、黒になる。5 つの検出器のステータスを 5 つの要素 [0,0,0,0,0] で示す。



## ライン追跡機能のサブフロー

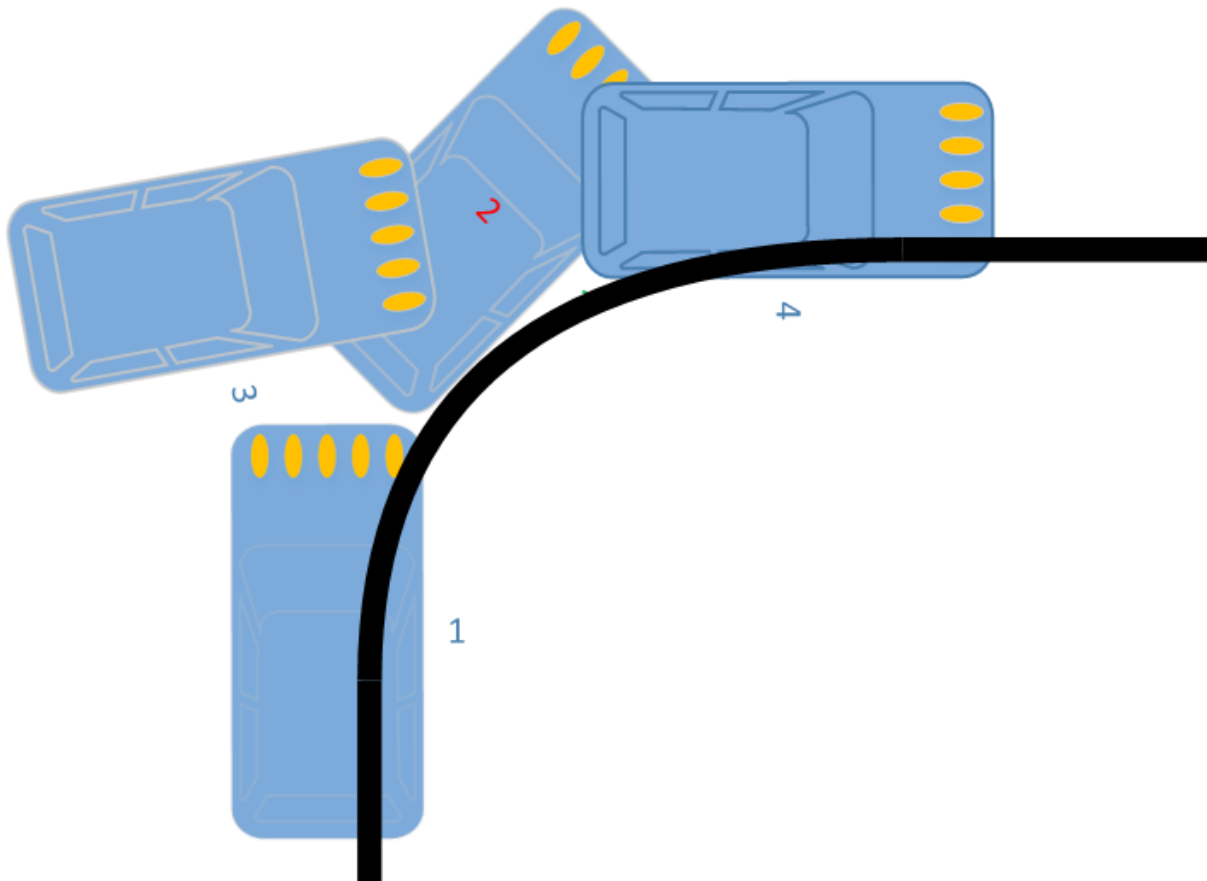


ラインフォロー機能では、プローブの検出結果に応じてサーボの回転角度を異なるレベルに設定する。車の前の線が小さなカーブとして検出された場合、車は小さな角度で曲がる。大きいと車は大きい角度で曲がる。したがって、ここでは、a\_step、b\_step、c\_step、d\_step の 4 つの角度回転定数を設定する。



もともと車が前進するとき、サーボは 90 度である。車を運転して左折するには、サーボを 90 度以上の角度にしなければならない。右折するには、サーボは 90 ステップの角度でなければならない。

特別な場合もある：車がコースから外れ、すべてのプローブが黒い線を検出できなくなった場合、以下のプログラムが実行される。



場合によっては、特にカーブの半径が非常に小さい方向 (1) に車が曲がる場合、車が軌道を使い果たして黒い線を検出できない場合がある (2)。このような場合に応答プログラムがないと、車は再び軌道を追うことができなくなる。したがって、車を反対方向に後退させるように応答プログラムを設定し (3)、次に黒い線が再び検出されるまで元の方向に戻り、前に進む (4)。

#### 機能説明

コードのロジックは、上記のフローチャートに示すとおりである。

インポートされた `SunFounder_Line_Follower`、`front_wheels`、`back_wheels` を含む 3 つの Python モジュールがコードで使用されている。これらはこのキットのドライバーであり、それぞれラインフォロワー、前輪と後輪となっている。

関連クラスはここで定義されている。モジュールを適用して使用すると、関連するクラスのオブジェクトが作成され、ハードウェアのさまざまな部分がクラスオブジェクトによる関数を呼び出すことによって駆動される。

次のモジュールと同様に、`If` 上という名前のオブジェクトを作成する：

**lf = Line\_Follower\_module.Line\_Follower(references=REFERENCES)**

パラメータは初期値であり、クラスオブジェクトを呼び出して関数を適用する。

**lf.read\_digital()**

この機能はすべてのプローブのアナログ信号を読み取り、それをデジタル信号に変換するために使用される。信号が参照よりも大きい場合、対応するパラメーターは 0 になり、基準より低い場合、パラメーターは 1 になる。プローブは 5 つあるため、5 つのパラメーターのリストを取得する。

**fw.turn(turning\_angle)**

前輪回転機能。メインプログラムは前輪を回す場合にこの関数を呼び出す。パラメータは回転角度となっている。

**bw.forward()**

**bw.set\_speed(forward\_speed)**

後輪には 2 つの機能が必要である。1 つは回転方向を順方向に制御する（後ろに回転するには、**bw.backward()**）。2 つ目は回転速度を設定する。パラメータは速度の値である（範囲 0 ~ 100）。パラメータが大きければ大きいほど、ホイールの回転が速くなる。

### 1.8.4 組合せ

したがって、このスマートカーは 3 つの異なる機能でスマートになる。しかし、1 つのセンサーモジュールだけでは十分ではないと思っているか？それらのセンサーモジュールを 1 つに組み合わせてみてください！ここでは、実験を示す-参照用の障害物回避によるライトフォロー。

ライトフォロアを使用して車を走行させると、ライトを追うときに障害物にぶつかる場合がある。また、車を後退させるのはあまり便利ではない（ただし、配列が [1,0,1] である場合、車を後退させると設定したが、車が動いていて、ライトが時々正確にできない場合があるため、これらの値を取得することは難しい）。だから、板紙や足で車を後退させることもできるので、それがとても簡単である。

この例のプログラムを以下で確認してください。

まず、車のライトフォロアモジュールと超音波障害物回避モジュールを組み立てる。

ssh 経由でコンピューターの Raspberry Pi にログインし、ディレクトリに入る

```
cd ~/SunFounder_PiCar-S/example
```

コードを実行する。

```
python3 light_with_obsavoidance.py
```



## 動作原理

障害物回避をライトフォローよりも優先順位を高く設定する。車の前に障害物がある場合、障害物から離れて軌道に戻る。そうでない場合、車はライトを追従する。

車のライトフォローと障害物回避はセンサーモジュールに依存するため、2つのセンサーのステータスを個別に読み取る2つの関数を設定し、それらの関数から返されるフラグに値を割り当てる：`state_light()` と `state_sonic()`。

関数 `state_sonic()` は、戻り値は `avoid_flag` である。

車が障害物に 接近している 場合は、`avoid_flag=2` を返す；

障害物に 近すぎる 場合は、`avoid_flag = 1` を返す；

前方に障害物が 検出されない 場合は、`avoid_flag = 0` を返す。

関数 `state_light()` では、戻り値は `light_flag` である。

ライトスポットが車の 前にある 場合、`light_flag = 0` を返す；

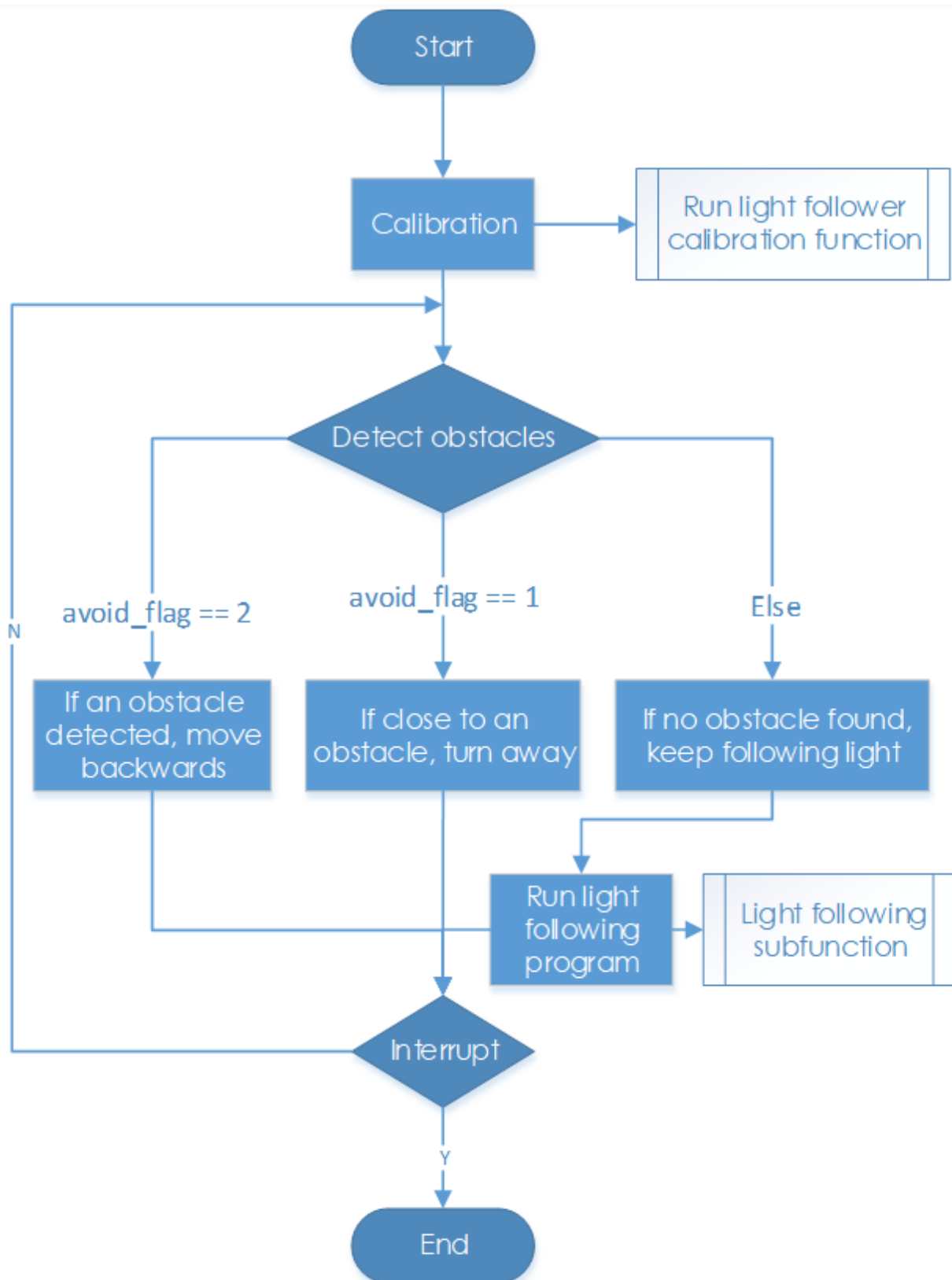
スポットが 右側にある 場合、`light_flag = 1` を返す；

スポットが 左側にある 場合、`light_flag = 2` を返す；

スポットが 後ろにある 場合、`light_flag = 3` を返す；

光点が検出されない 場合、`light_flag = 4` を返す。

メインプログラム `main()` は、`avoid_flag` と `light_flag` に従って対応するプログラムを実行し、`avoid_flag` は優先順位となっている。



## 1.9 付録

### 1.9.1 手動でインストールする

1. apt リストを更新する。

```
sudo apt-get update
```

2. python-smbus をインストールする。

```
sudo pip3 install smbus2
```

3. PiCar モジュールをインストールする。

```
cd~
git clone --recursive https://github.com/sunfounder/SunFounder_PiCar.git
cd SunFounder_PiCar
python3 setup.py install
```

4. I2C を作動させる。

/boot/config.txt ファイルを編集する

```
sudo nano /boot/config.txt
```

各行の前の "#" は、スケッチでは有効にならない以下の内容をコメント化するためのものである。I2C 構成部分もデフォルトでコメント化されている。ファイルの最後に次のコードを追加するか、または関連する行の先頭にあるポンド記号 "#" を削除する。どちらもいい。

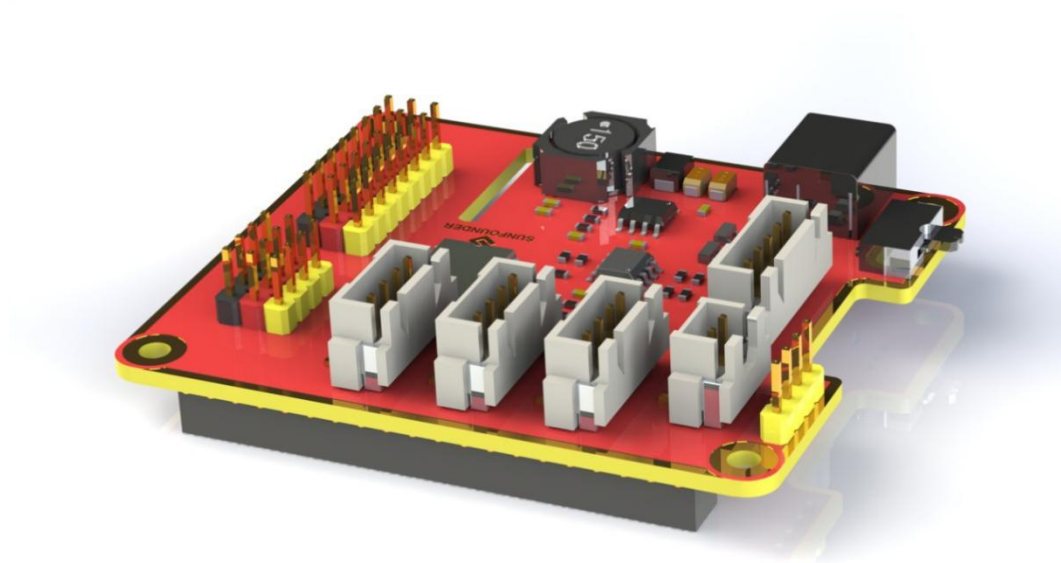
```
dtoverlay=i2c_arm=on
```

5. 再起動する。

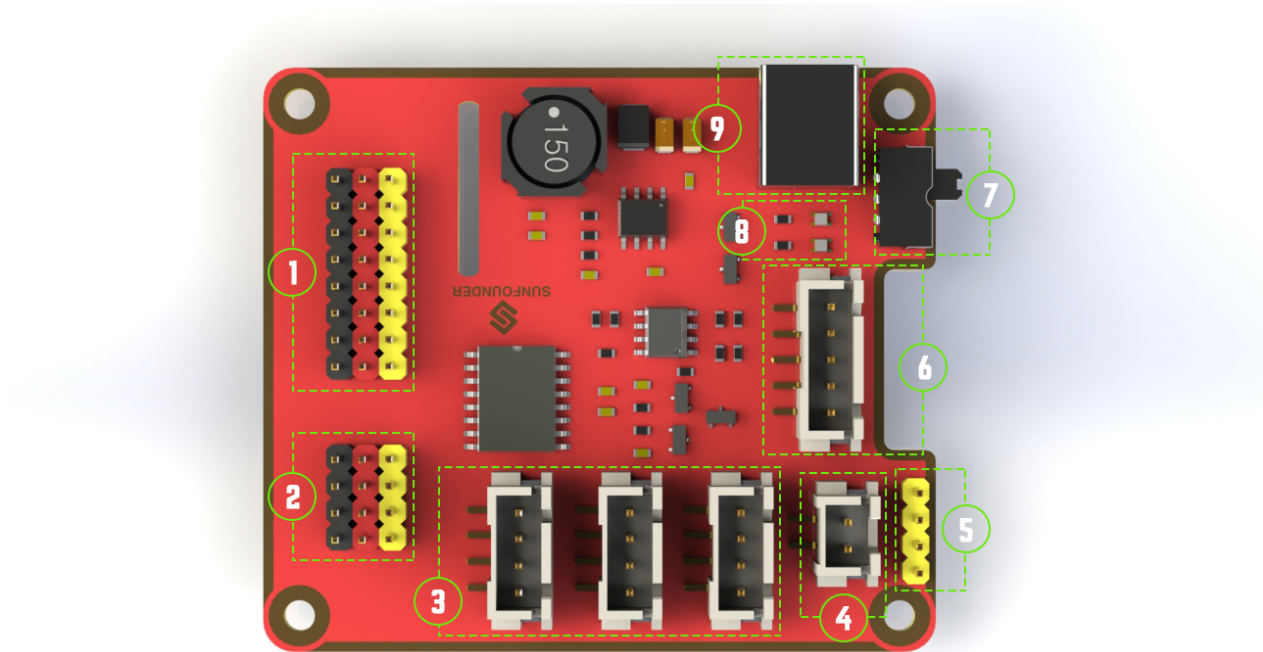
```
sudo reboot
```

### 1.9.2 モジュール

### ロボット HATS



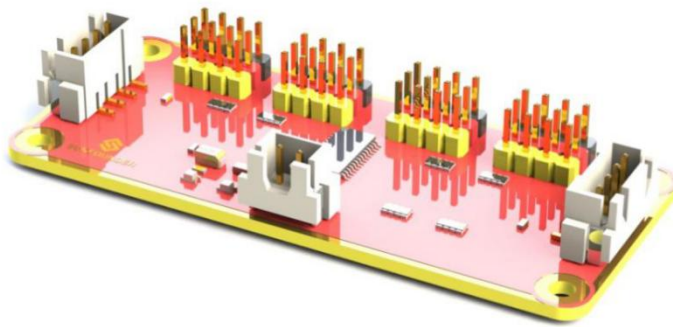
ロボット **HATS** は 40 ピン Raspberry Pi に特別に設計された HAT であり、Raspberry Pi 世代 3 モデル B、世代 3 モデル B +、世代 4 モデル B で動作し、GPIO ポートから Raspberry Pi に電力を供給する。HATS のルールに基づいた理想的なダイオードの設計により、USB ケーブルと DC ポートの両方を介して Raspberry Pi に電源を供給できるため、バッテリーの電力不足によって TF カードが損傷することを防ぐことができる。PCF8591 は I2C 通信とアドレス 0x48 を備えた ADC チップとして使用される。



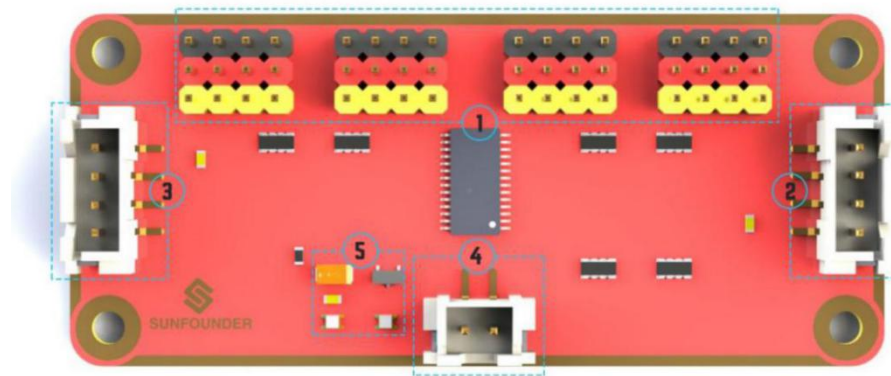
1. デジタルポート : 3 線式デジタルセンサーポート、信号電圧 : 3.3V、VCC 電圧 : 3.3V。

2. アナログポート：3 線 4 チャンネル 8 ビット ADC センサーポート、基準電圧：3.3V、VCC 電圧：3.3V。
3. I2C ポート：3.3V I2C バスポート
4. 5V 電源出力：PWM ドライバーへの 5V 電源出力。
5. UART ポート：4 線 UART ポートと 5V VCC は USB への SunFounder FTDI シリアルと完全に連携する。
6. モーター制御ポート：モーター用 5V、モーター MA と MB の方向制御、フローティングピン NC、モータードライバースタックとの連携。
7. スイッチ：電源スイッチ
8. 電源インジケータ：電圧を指示する-2 つのインジケータが点灯：> 7.9V。1 つのインジケータ：7.9V ~ 7.4V。インジケータ点灯なし：< 7.4V。バッテリーを保護するために、インジケータが点灯していない場合は、充電する際にこれを取り出してください。電源インジケータは単純なコンパレータ回路によって測定された電圧に依存する。負荷によっては検出電圧が通常より低下する場合があるので参考値としてご利用ください。
9. 電源ポート：5.5/2.1mm 標準 DC ポート、入力電圧：8.4 ~ 7.4V（制限された動作電圧：12V ~ 6V）。

## PCA9865



PCA9865 16 チャンネル 12 ビット I2C バス PWM ドライバー。独立した PWM 出力電力をサポートし、並列接続用の 4 線式 I2C ポート、PWM 出力用の区別された 3 色ポートをより簡単に利用できる。



1. PWM 出力ポート：3 色ポート、独立したパワー PWM 出力ポート、サーボへの直接接続。

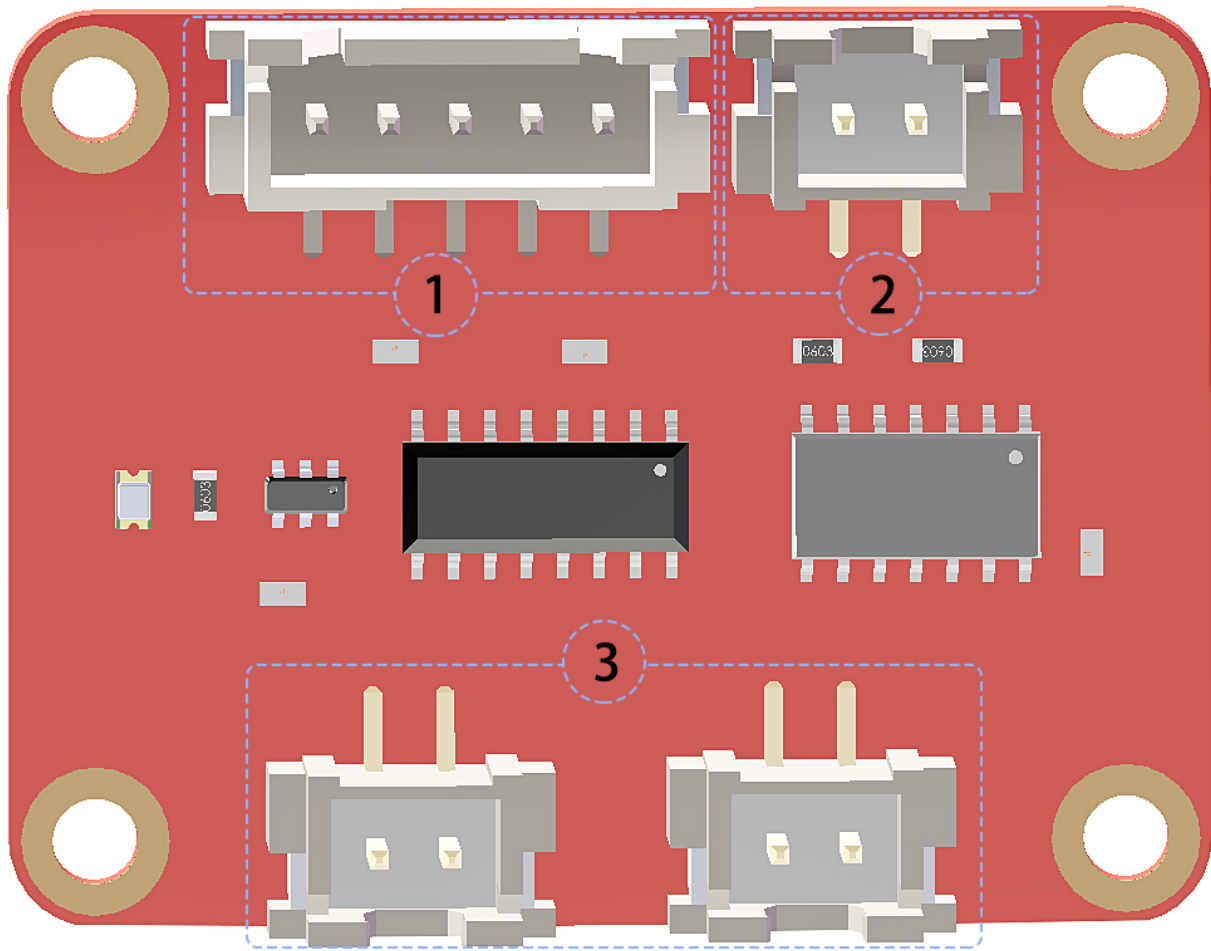
2&3. I2C ポート：4 線式 I2C ポートは並列で使用できる。3.3V/5.5V に対応

3. PWM 電源入力：最大 12V。

4. LED：チップと PWM 電源入力用電源インジケータ。

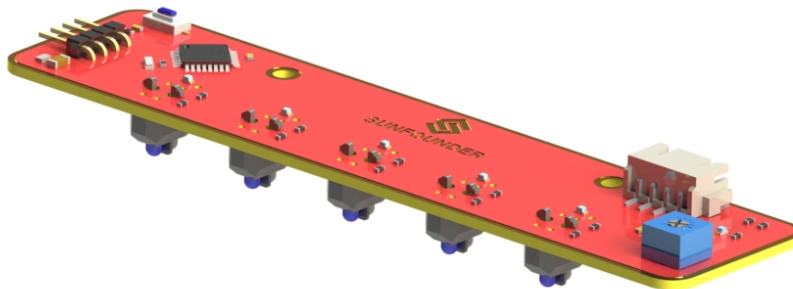
### モータードライバーモジュール

モータードライバーモジュールは低発熱の小型パッケージモータードライブである。



1. 電源とモーター制御ポート：チップとモーターに電力を供給し、モーターの方向を制御するためのピンが含まれている。
2. モーターの PWM 入力：2 つのモーターの速度を調整するための PWM 信号入力。
3. モーター出力ポート：2 つのモーターの出力ポート

### ラインフォロワーモジュール



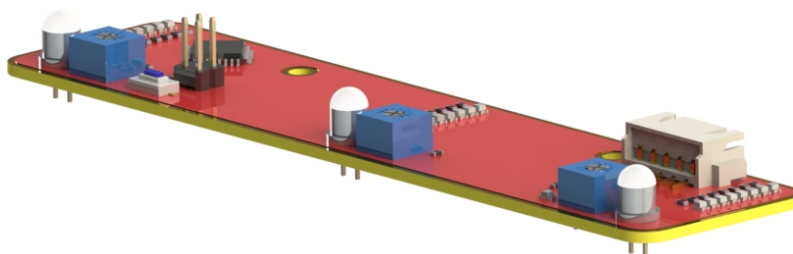
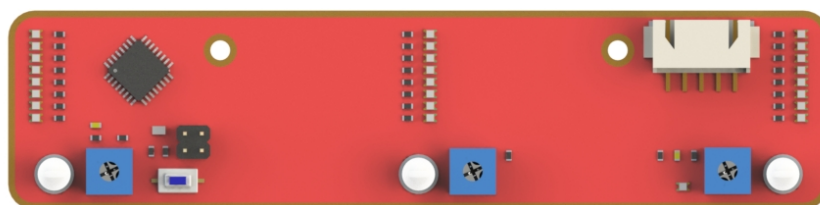
TCRT5000 赤外線光電スイッチは高送信電力赤外線フォトダイオードと高感度フォトリランジスタを採用している。これは、オブジェクトの IR 反射光の原理を適用することによって機能する。光は放射され、次に反射され、同期回路によって感知される。そして、光の強弱によって物体の有無を判断する。白黒の線を簡単に識別できる。

言い換えると、フォトリランジスタが黒と白のラインを通過するときの異なる伝導レベルにより、異なる出力電圧が生成される可能性がある。したがって、必要なのは、Atmega328 の AD コンバーターによってデータを収集し、I2C 通信を介してマスターコントロールボードにデータを送信することだけである。



このモジュールは5つのTRT5000 センサーを使用する赤外線追跡センサーである。TRT5000 の青色 LED は発光管であり、通電すると人間の目には見えない赤外光を発する。センサーの黒い部分は受信に使用される。内部の抵抗器の抵抗値は、受信した赤外線によって変化する。

#### ライトフォロワーモジュール

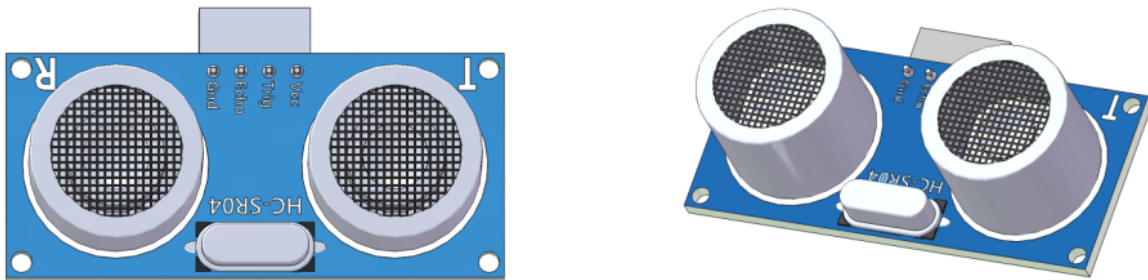


フォトトランジスタはフォトダイオードとも呼ばれ、光を電流に変換するデバイスである。フォトンがPNジャンクションで吸収されると、電流が生成される。逆電圧が印加されると、デバイスの逆電流は光の輝度とともに変化する。光が強いほど、逆電流が大きくなる。ほとんどのフォトトランジスタはこのように動作する。

HATS の ADC チップは 8 ビットのアナログ信号を受信して整数に変換し、信号を Raspberry Pi に転送する。Raspberry Pi はデータを分析して、最も明るい領域（光源）の方向を決定し、4 つの車輪のステアリングと動きを制御して光源に近づける。

この実験では、光に焦点を当てた懐中電灯が必用する。少なくとも、モジュールの 3 つのフォトトランジスタに同時に到達するために、トーチのスポットサイズは大きすぎではない。まあ、あなたは小さなスポットサイズを得るために車の近くに懐中電灯を照らすこともできる。

### 超音波障害物回避モジュール



モジュールには、前方の障害物までの距離を検出する超音波距離センサ HC-SR04 が搭載されています。ロボットが障害物を避けるためによく使われています。2 つの穴があるので、ロボットに簡単に取り付けることができます。4 ピンの逆戻り防止ケーブルが付属しているので、よりタイトで簡単に配線ができます。

HC-SR04 超音波距離センサは、2cm から 400cm までの距離を 3mm の範囲精度で非接触で測定することができます。HC-SR04 モジュールは、それぞれ超音波送信機、受信機、制御回路が含まれているため、HC-SR04 モジュールを使用する際には、Trig 端子と Echo 端子の接続に注意する必要があります。PiCar-S に装着すると、距離を計測し、前方に障害物があるかどうかを検知します。

### 原理

Trig に 10us の短いパルスを供給してレンジングを開始し、モジュールは 40 kHz で 8 サイクルの超音波バーストを送信し、Echo にエコーを上げる。エコーはパルス幅と距離に比例した距離オブジェクトである。トリガー信号を送信してからエコー信号を受信するまでの間に、Time Interval を通じて Range を計算できる。

式：

$$Range(m) = \frac{TimeInterval \times 340_{m/s}}{2}$$

または：

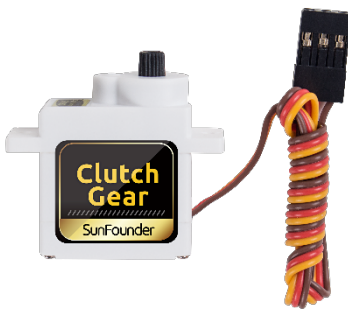
$$\text{Range (cm)} = \frac{\text{Time Interval}}{58}$$

または：

$$\text{Range(inchs)} = \frac{\text{Time Interval}}{148}$$

エコーへのトリガー信号を防ぐために、60ms 以上の測定サイクルを使用してください。

### SunFounder SF006C サーボ

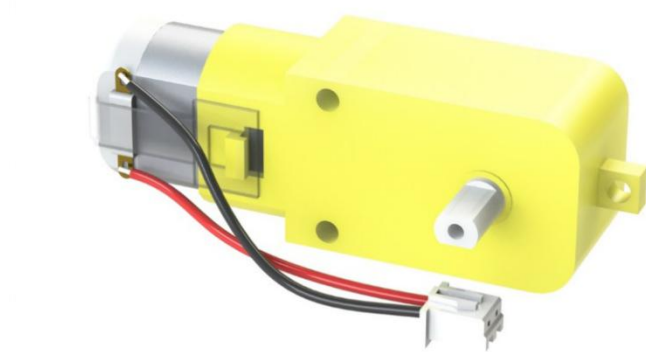


SunFounder SF0180 サーボは 180 度の 3 線式デジタルサーボである。60Hz の PWM 信号を利用し、物理制限はない。最大 180 度までの内部ソフトウェアのみによって制御される。

電気仕様：

Item	V = 4.8V	V = 6.0V
Consumption Current* (No Load)	≤50mA	≤60mA
Stall Current	≤550mA	≤650mA
Rated Torque	≥0.6 kgf·cm	≥0.7 kgf·cm
Max. Torque	≥1.4 kgf·cm	≥1.6 kgf·cm
No Load Speed	≤0.14sec/60°	≤0.12sec/60°

## DC ギアモーター



これは減速機付きの DC モーターである。以下のパラメーターを参照してください：

Motor	Model	F130SA-11200-38V
	Rated Voltage	4.5V-6V
	No-load Current	$\leq 80\text{mA}$
	No-load Speed	$10000 \pm 10\%$
Gear Reducer	Gear Ratio	1:48
	Speed (no-load)	$\approx 200\text{rpm}$ ( $\approx 180\text{rpm}$ in test)
	Current	$\leq 120\text{mA}$

## 著作権表示

このマニュアルのテキスト、画像とコードを含むがこれらに限定されないすべての内容は、SunFounder Company が所有している。関連する規制と著作権法に基づき、著者と関連する権利所有者の法的権利を侵害することなく、個人的な研究、調査、享楽、またはその他の非営利目的でのみ使用してください。許可なく営利目的でこれを使用する個人または組織については、会社は法的措置を取る権利を留保する。

## 1.10 ありがとうございました

私たちの製品を評価した評価者、チュートリアルのための提案を提供したベテラン、そして私たちをフォローしてサポートしてきたユーザーのおかげで。私たちにあなたの貴重な提案は、より良い製品を提供する私たちのモチベーションです!

特定の感謝

- レン・ヴィセソン
- カレン・ダニエル
- フアン・デラコスタ

さて、このアンケートに記入するのに少し時間を割いてもらえますか?

---

注釈: アンケートを送信した後、トップに戻って結果を確認してください。

---



## 第 2 章

# 著作権表示

このマニュアルのテキスト、画像とコードを含むがこれらに限定されないすべての内容は、SunFounder Company が所有している。関連する規制と著作権法に基づき、著者と関連する権利所有者の法的権利を侵害することなく、個人的な研究、調査、享楽、またはその他の非営利目的でのみ使用してください。許可なく営利目的でこれを使用する個人または組織については、会社は法的措置を取る権利を留保する。