SunFounder PiCar-V

www.sunfounder.com

2022 年 08 月 02 日

目次

第 1 章

PiCar-V について

PiCar-V には、広角 USB ウェブカメラ、配線が少なくシンプルな 3 つのまったく新しい回路基板、調整可能な構造が内蔵している一部のアクリルプレートや車を制御するほとんどすべてのプラットフォームに適した新しいコードが装備されている。

本書では、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-V を構築す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-V/tree/V3.0.

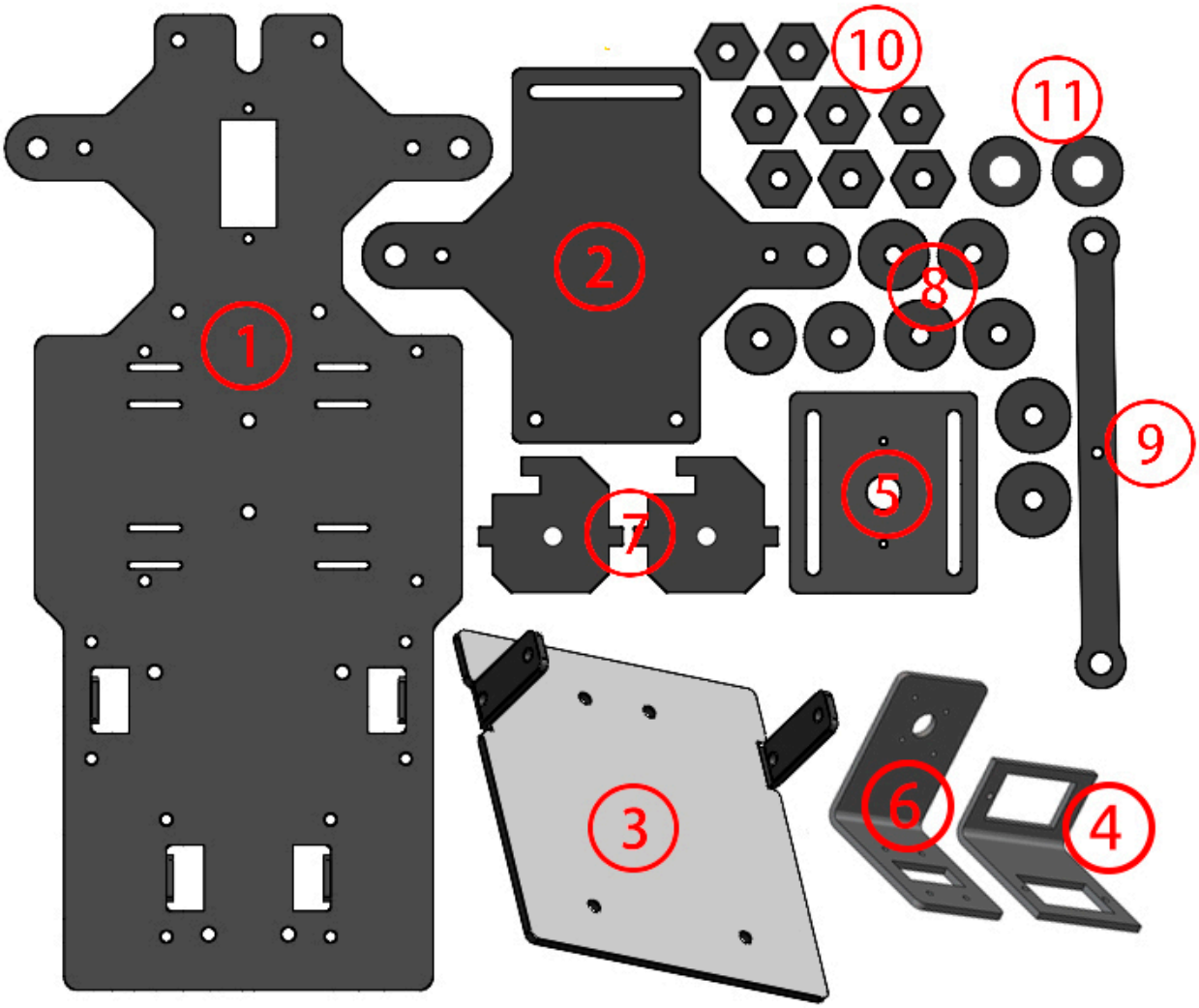
Github の上で弊社のページでお気軽いに問い合わせください。

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

1.1 部品一覧

1.1.1 プレート



1. 上部プレート x 1
2. 前部ハーフプレート× 1
3. 後部ハーフプレート× 1
4. パン/チルトプレート x 1
5. パンとチルトのベースプレート x 1
6. カメラマウントプレート x 1
7. サーボコネクタープレート x 2

8. ベアリングシールド x 8
9. サーボリンケージプレート x 1
10. 六角前輪固定プレート× 8
11. ガスケットプレート× 2

注釈: ベアリングシールドはガスケットプレートに似ているが、ガスケットプレートの口径はベアリングシールドよりも大きくなっている。

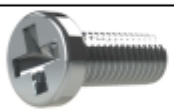
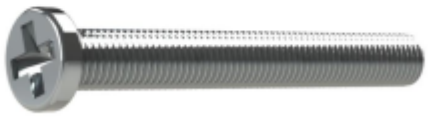
1.1.2 サーボ x 3

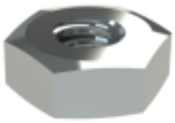
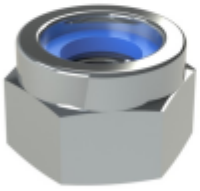
1. サーボ
2. 1 アームロッカーアーム
3. アームロッカーアーム
4. 4 アームロッカーアーム

5. ロッカーアーム固定ネジ

6. ロッカーアームねじ

1.1.3 メカニカルファスナー

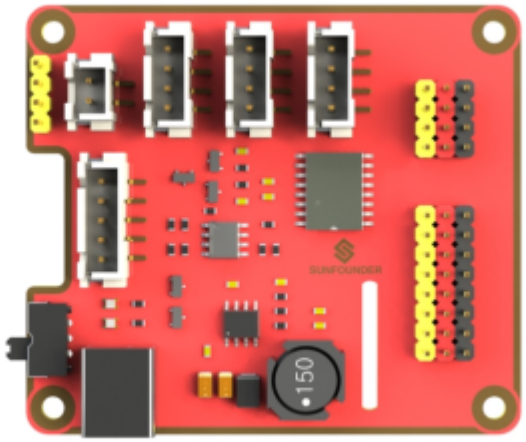
Name	Component	Qty.
M1.5x4 Self-tapping Screw		10
M2x8 Screw		6
M2.5x6 Screw		4
M2.5x12 Screw		8
M3x8 Screw		8
M3x8 Countersunk Screw		2
M3x10 Screw		7
M3x25 Screw		4
M4x25 Screw		2

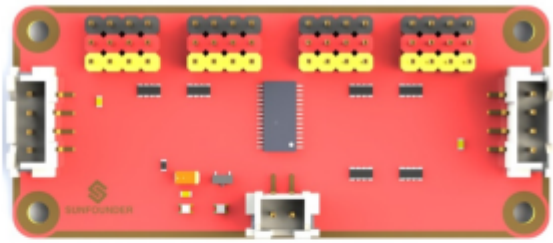
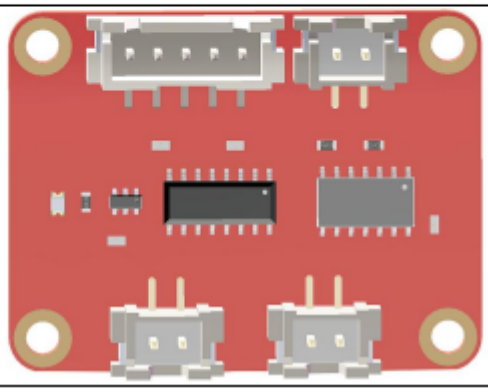
M2 Nut		6
M2.5 Nut		12
M3 Nut		21
M4 Self-locking Nut		2
M2.5x8 Copper Standoff		8
M3x25 Copper Standoff		8

1.1.4 配線

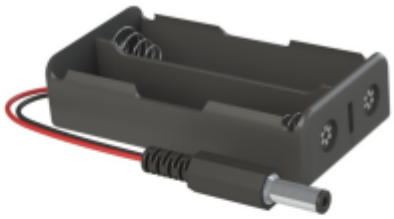
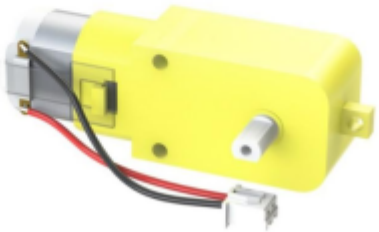
100mm HX2.54 5-Pin Jumper Wire		1
50mm HX-2.54 4-Pin Jumper Wire		1
50mm HX-2.54 2-Pin Jumper Wire		1
100mm HX-2.54 2-Pin Jumper Wire		1

1.1.5 PCB

Robot HATS		1

PCA9685 PWM Driver		1
Motor Driver Module		1

1.1.6 その他の部品

2x18650 Battery Holder		1
DC Gear Motor		2

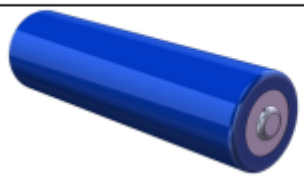
120° Wide-angle USB Camera		1
Rear Wheel		2
Front Wheel		2
Ribbon (30cm)		1

1.1.7 工具

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

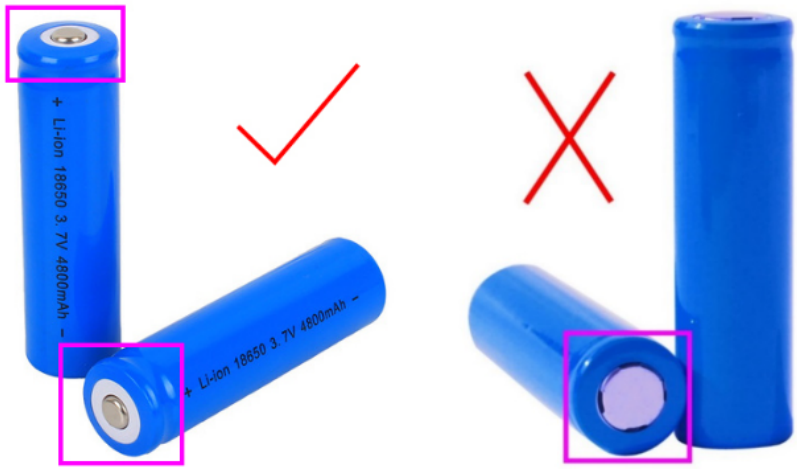
1.1.8 別途用意する必要な部品

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:

1. 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。
2. 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）。



3. 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-V は以前のスマートビデオカーキットに基づいて開発された。新しい PCA9685 PWM ドライバー、小型で高性能の DC モータードライバーモジュール、電源とインターフェイスが統合されたロボット HAT、より鮮明な視界を備えた広角カメラ、いくつかの優れたホイールとタイヤなどが搭載されている。詳しくは後で説明する。

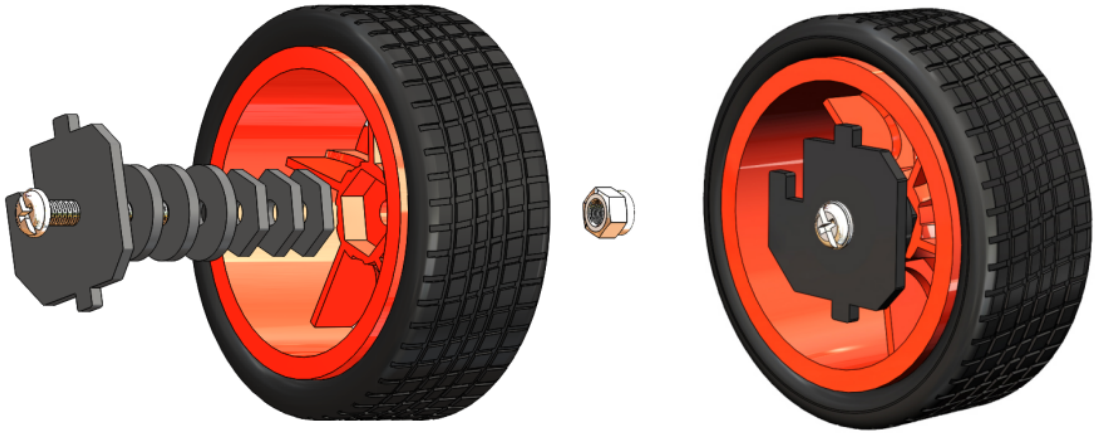
この新しいバージョンの基本的な機能は、PiCar-V をリモートで制御したり、キャプチャしたビデオデータをストリーミングしたりするなど、前のとほぼ同じである。それでは新規機能は何か？謎を解き明かそう！

1.3 車の組み立て

箱を開けてたくさんの部品を見るとワクワクしますか？あなたの忍耐を保ち、それを楽にしてください。次の手順の一部の詳細は注意深く観察する必要があることに注意してください。各ステップを終えた後、マニュアルの図に基づいて作業を再確認する必要があります。はじめましょう！

1.3.1 前輪

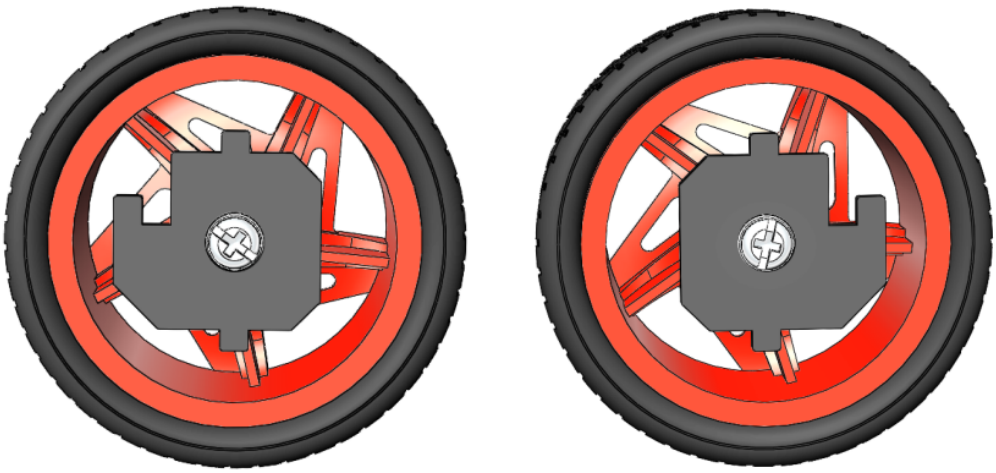
以下に示すように、**M4x25 ネジ** を一つの **ステアリングコネクタプレート** 、3 つの **ベアリングシールド** 、3 つの **六角前輪固定プレート** 、と一つの **前輪** を通して **M4 セルフロックナット** に挿入する：



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。

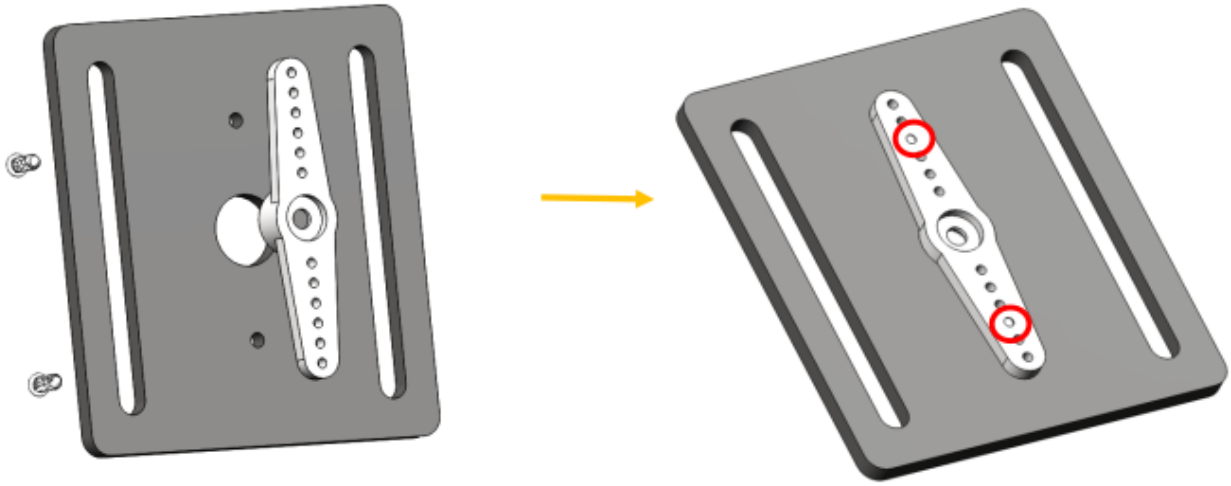
同じ方法で他の前輪を組み立てるが、車輪のステアリングコネクタプレートは前のものと対称であることを覚えておいてください。



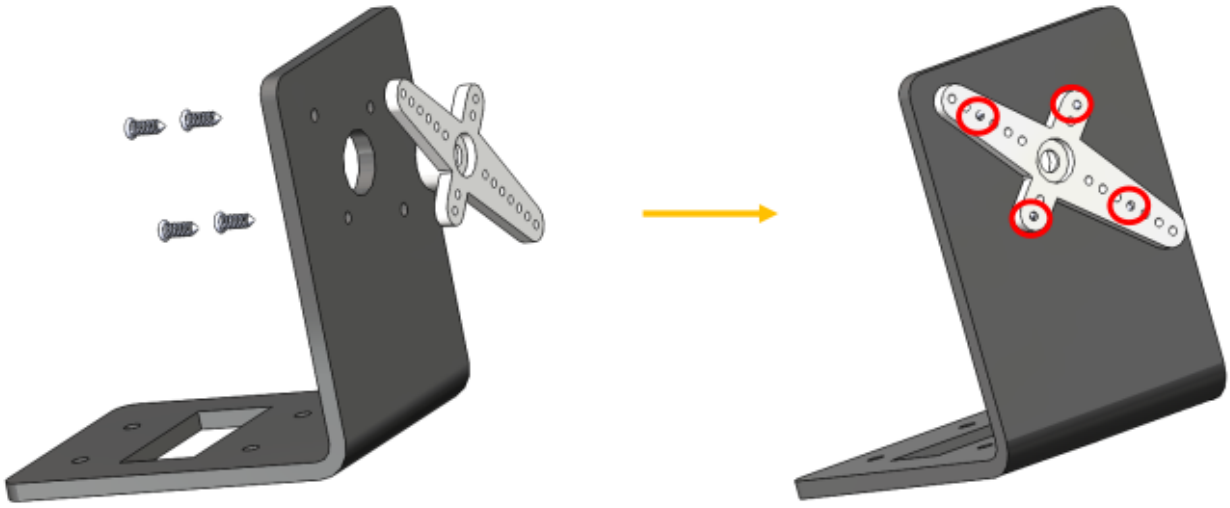
これで 2 つの前輪の組み立てが完了した。

1.3.2 パンとチルト

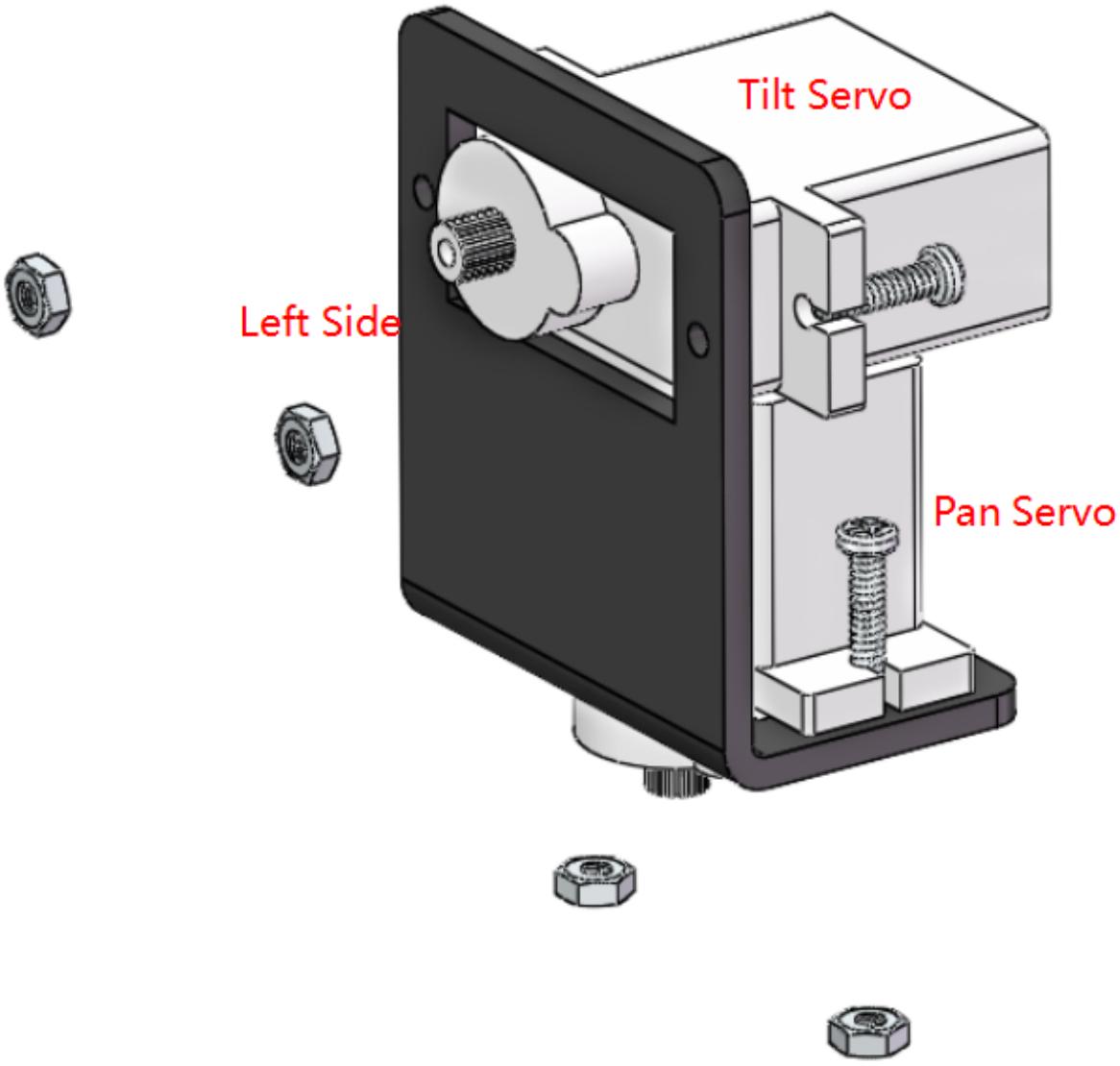
2 x アームロッカーアームを取出し、2M1.5 x 4 ネジを使って、パン-チルトベースプレートに取り付けます。(方向はに次のように)。



カメラマウントプレート にも同様の操作を行ってください。



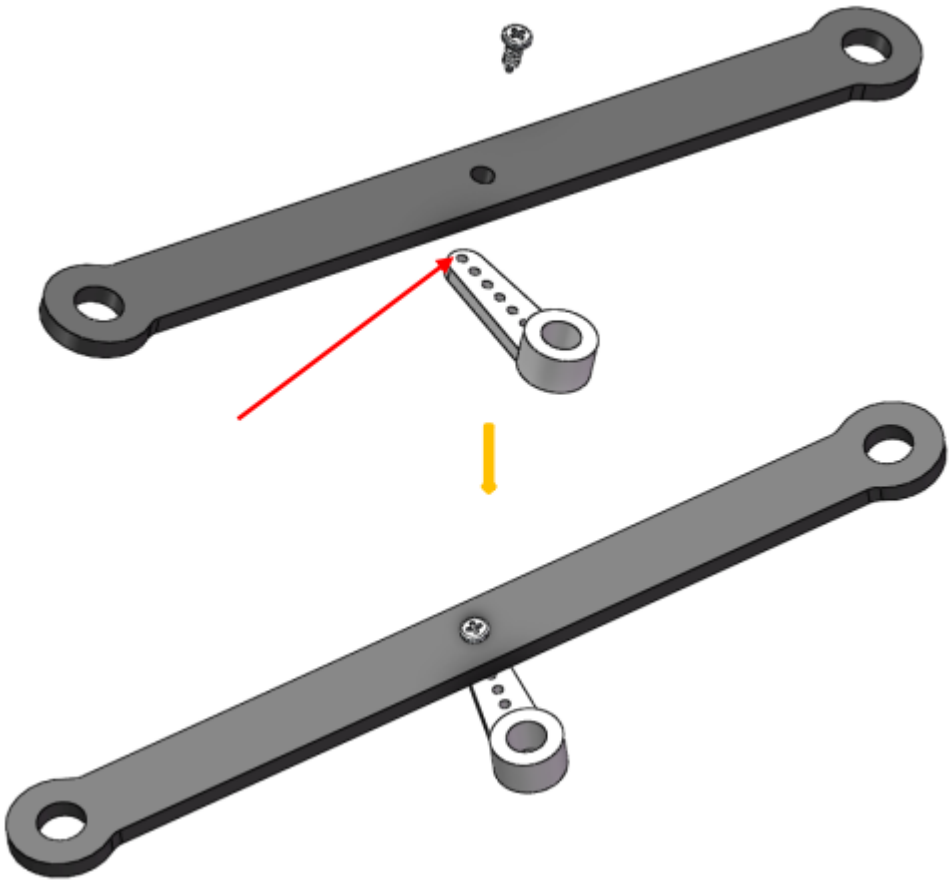
4 つの M2x8 ネジ と **M2 ナット** を使用して、2 つのサーボを **パン＆チルトプレート** に取り付ける（2 つのサーボシャフトは左側にある）。



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。

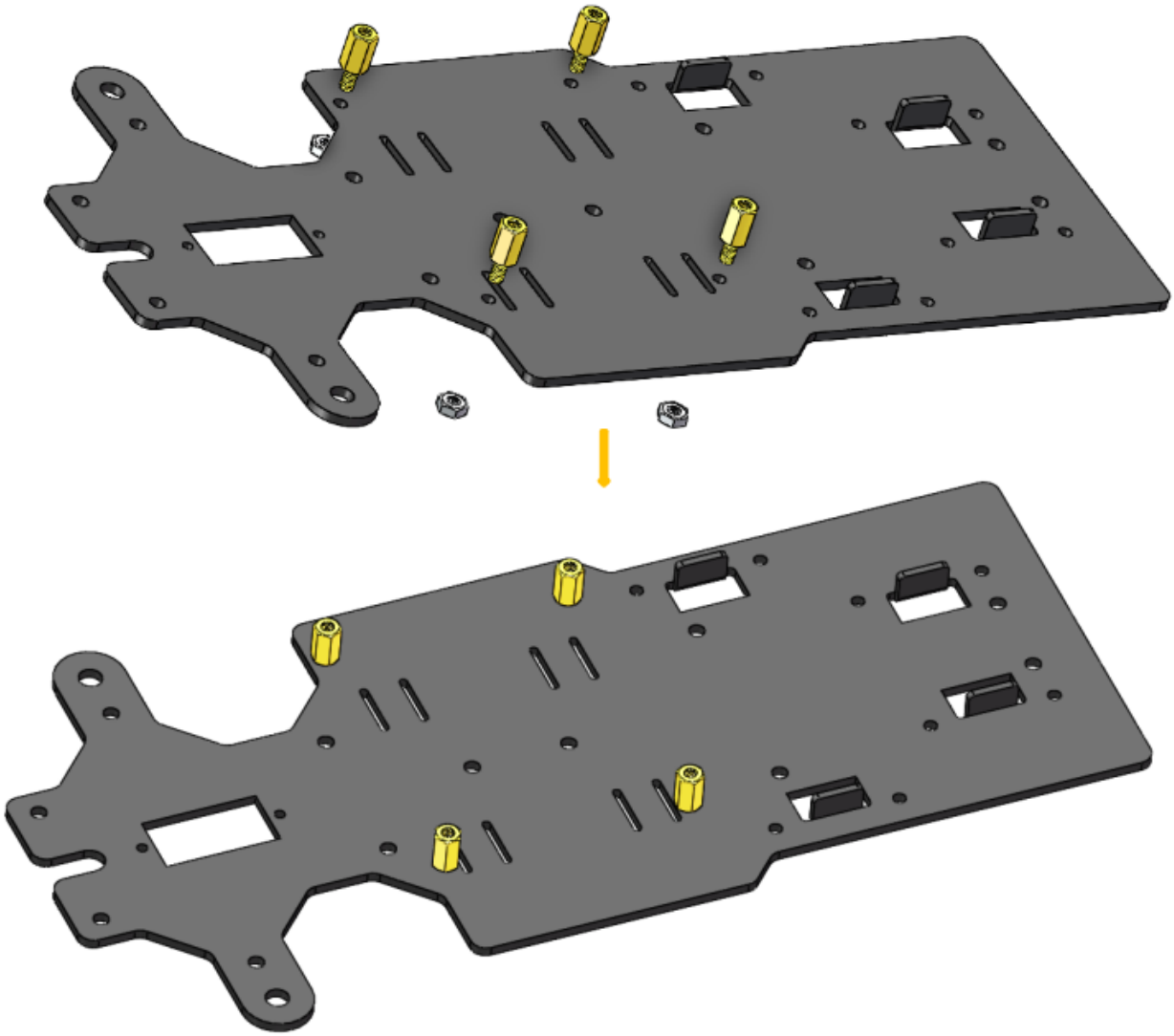
注釈: ギアから最も遠いアームの **最初** の穴（下の **矢印** で示されている）に挿入します。



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

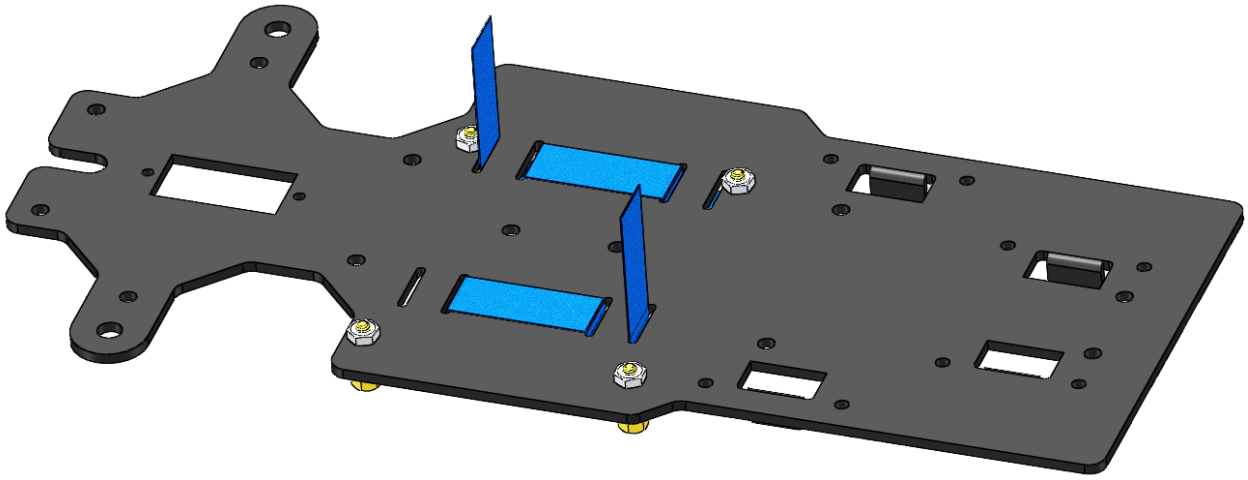
1.3.4 上部プレート

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** に取り付ける。突き出た支柱が上に向くように注意してください。

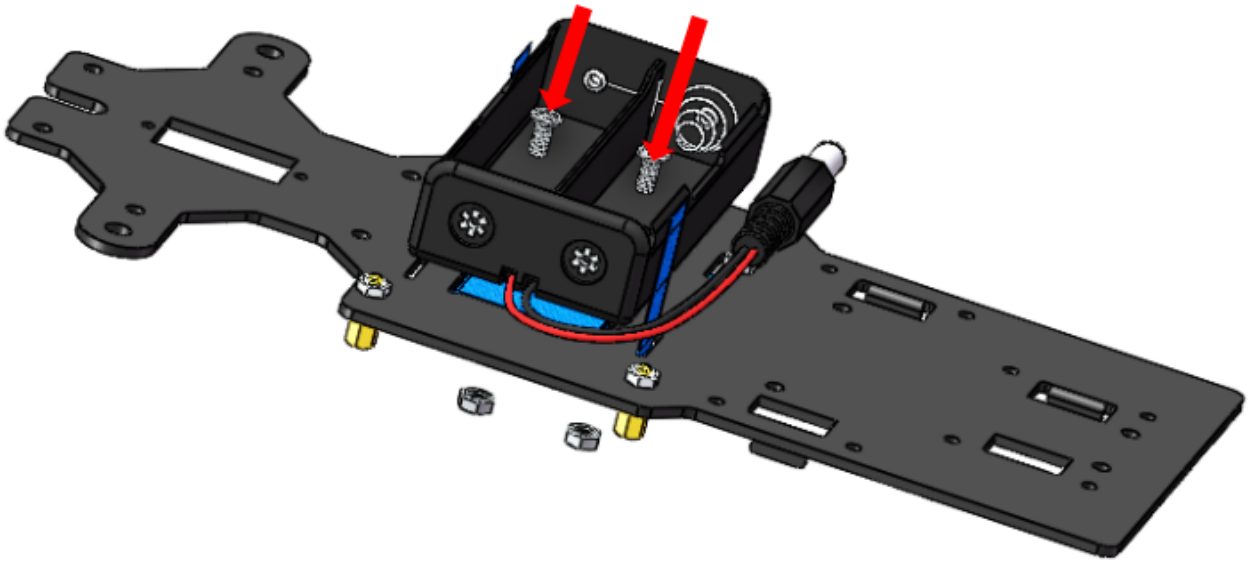


1.3.5 バッテリーホルダー

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、プレートの一方の端を長くしてください。

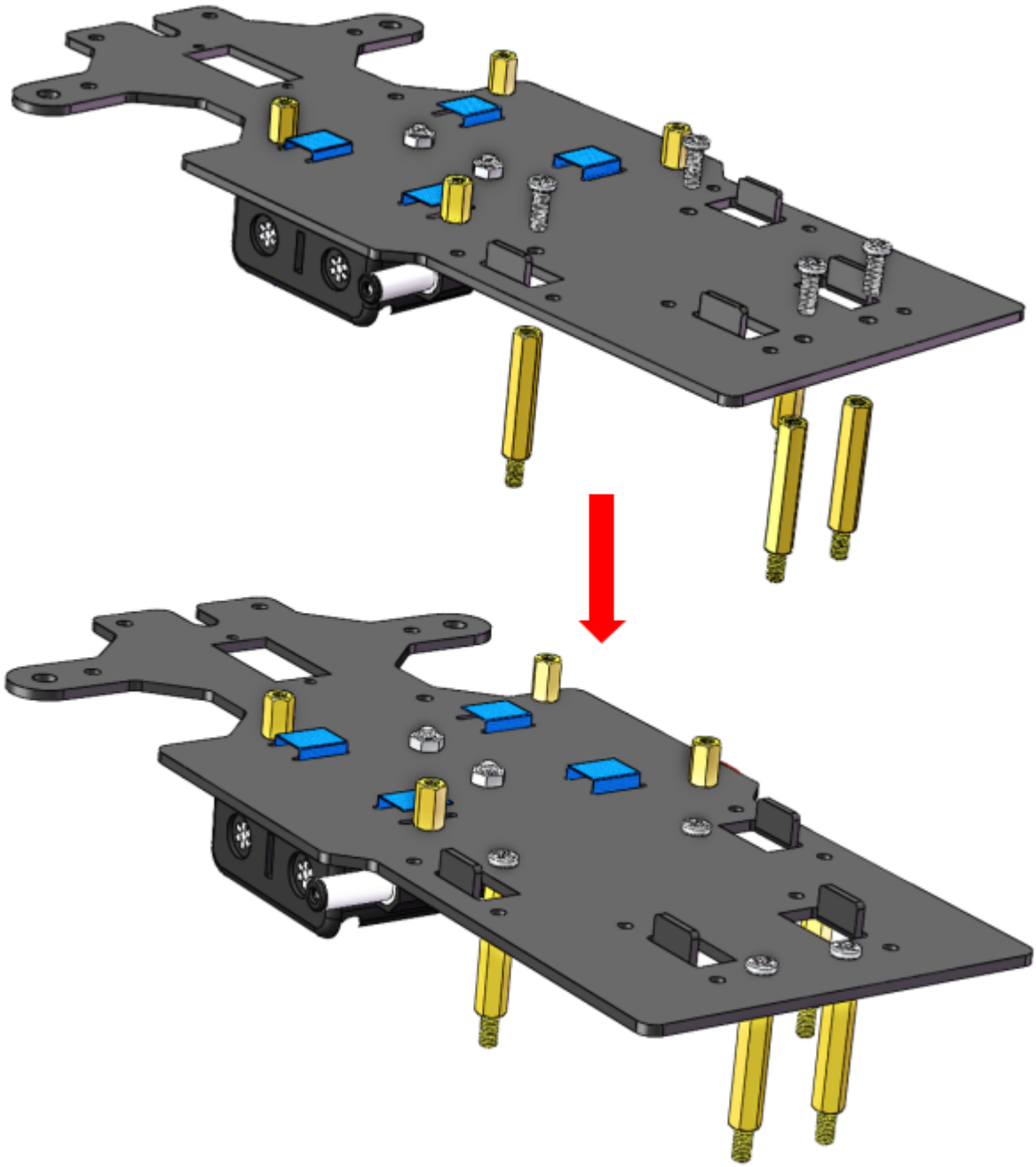


2 本の **M3x8 皿ネジ** と **M3 ナット** でバッテリーホルダーを固定する。バッテリーホルダーの配線の方向に注意を払ってください。



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。



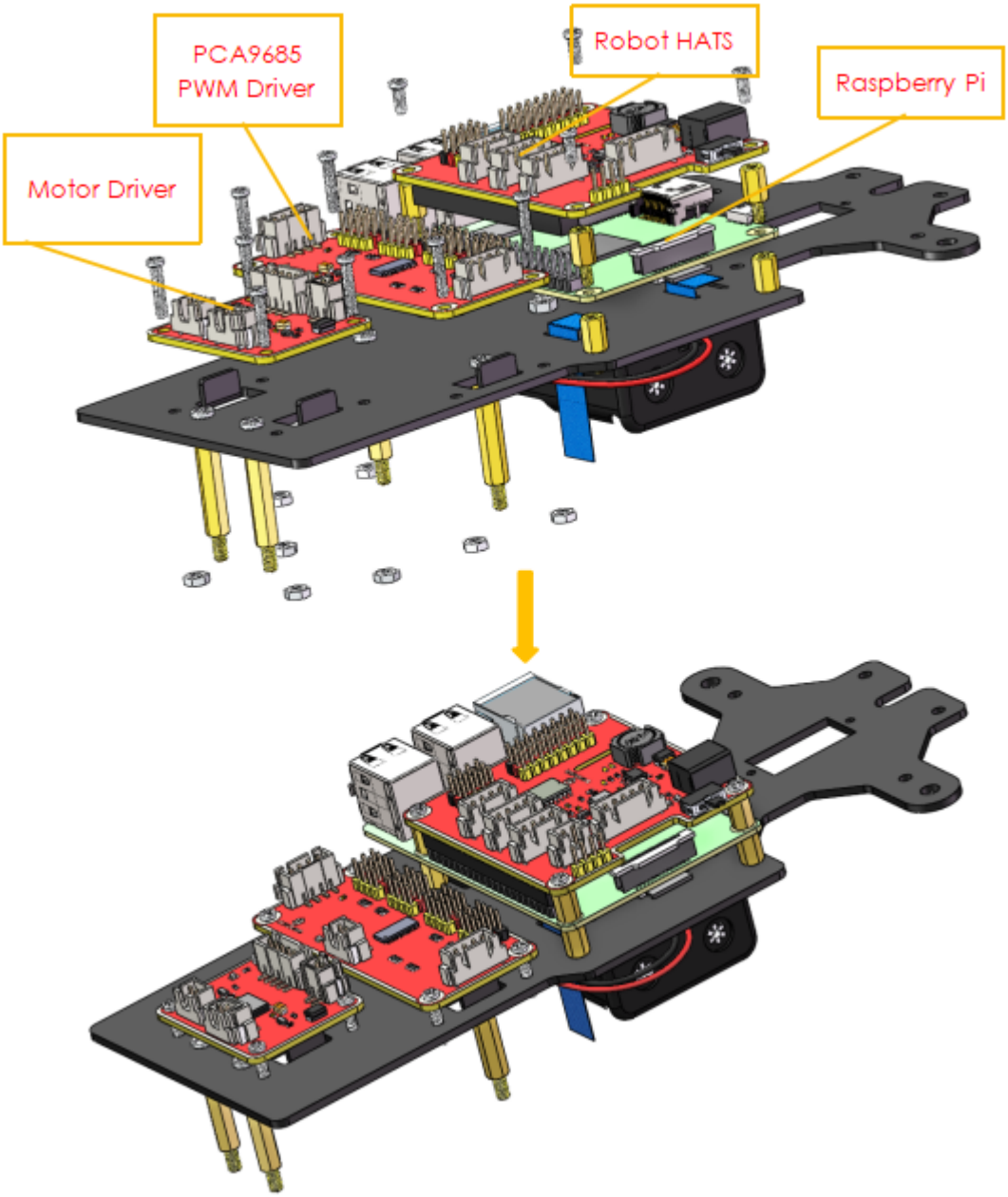
1.3.7 PCB 組み立て

1) **Raspberry Pi**（TF カード差込済み）を 8 つの **M2.5x8 シングルパス銅製スタンドオフ** で組み立てから **ロボット HATS** を差し込む。

2) 4 つの **M2.5x6 ねじ** で **ロボット HATS** を固定する。

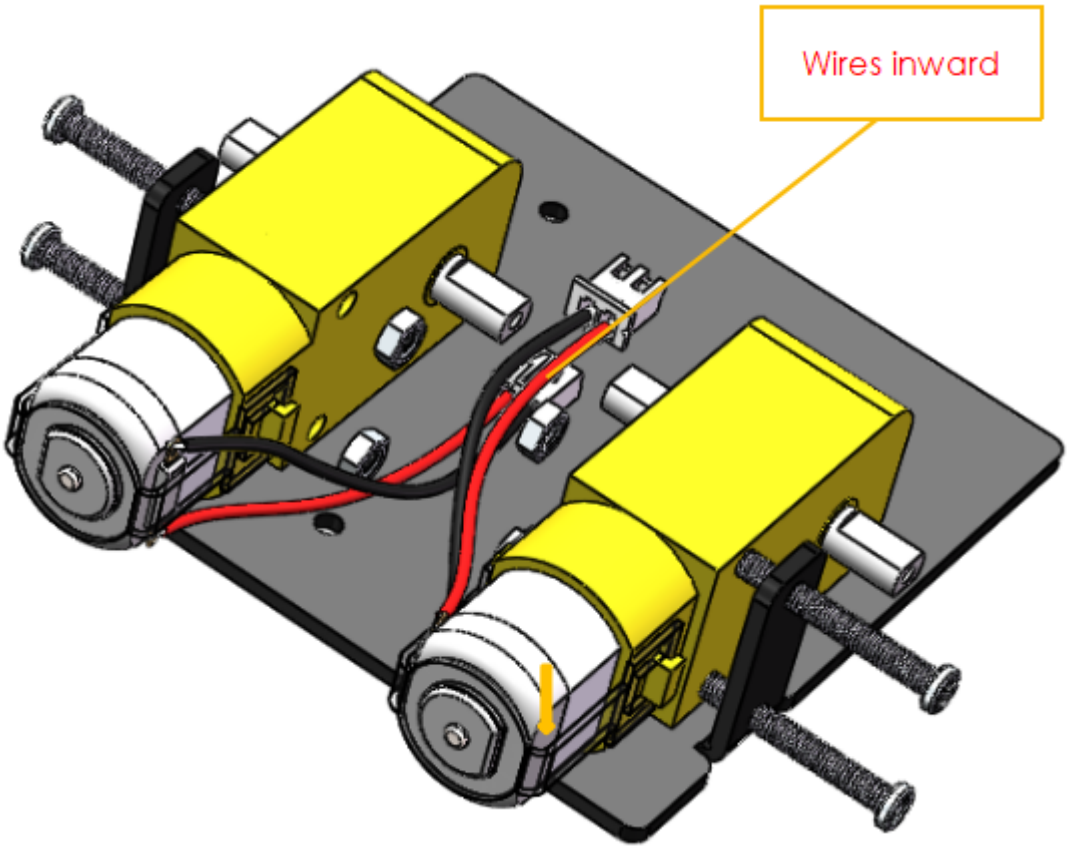
PCA9685 PWM ドライバー と **モータードライバ** ーを 8 つの **M2.5x12 ネジ** と **M2.5 ナット** で **下部プレート** に固

定する：

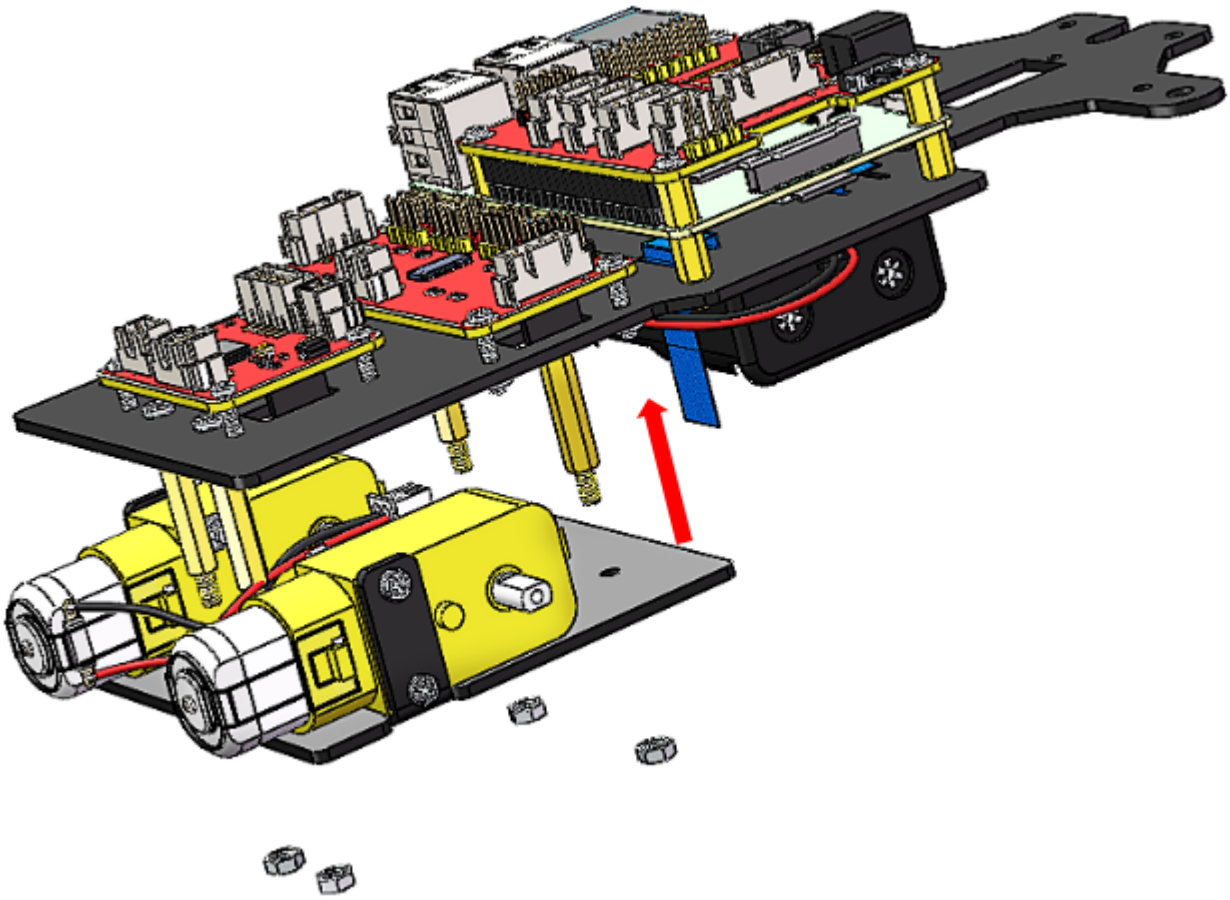


1.3.8 後輪の固定

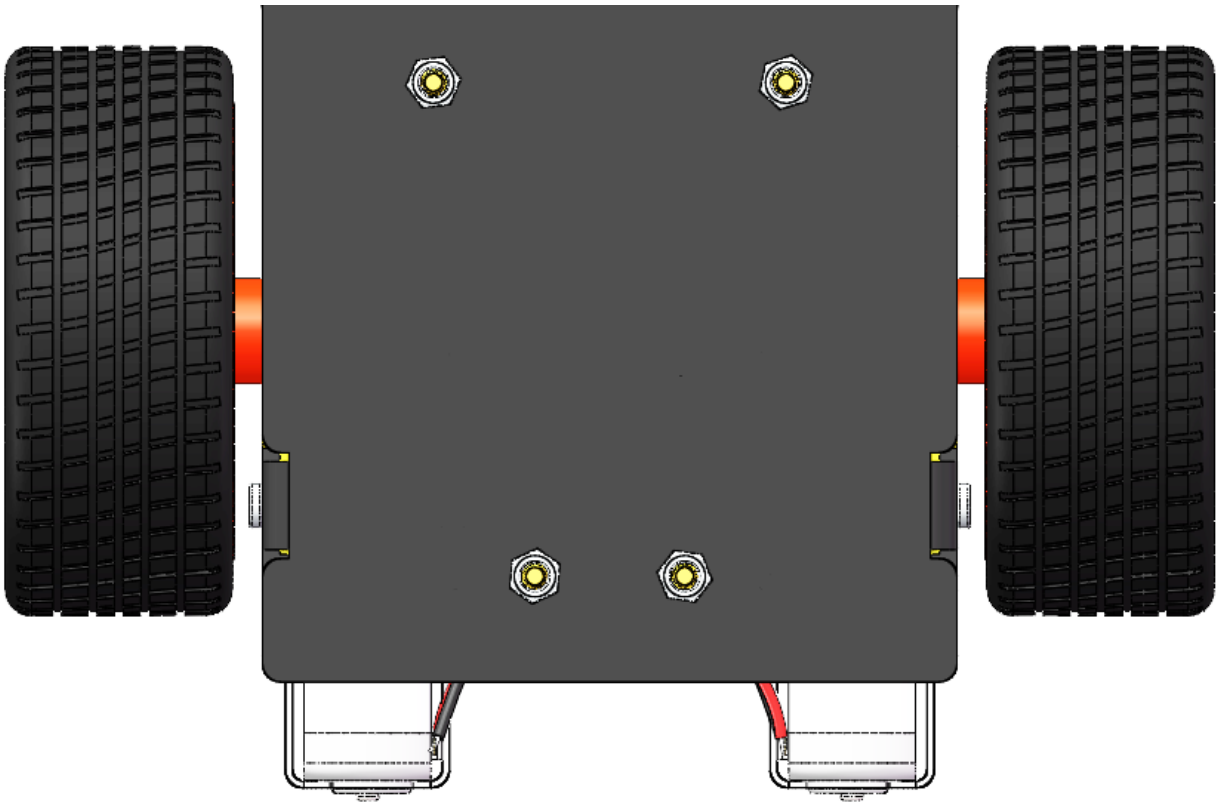
4 つの **M3x25 ネジ** と **M3 ナット** で 2 つのモーターを組み立てる。モーターを配線で内側に配置するように注意してください。こうすると、回路を接続することを便利にする。



4 つの M3 ナットで後輪を組み立てる。

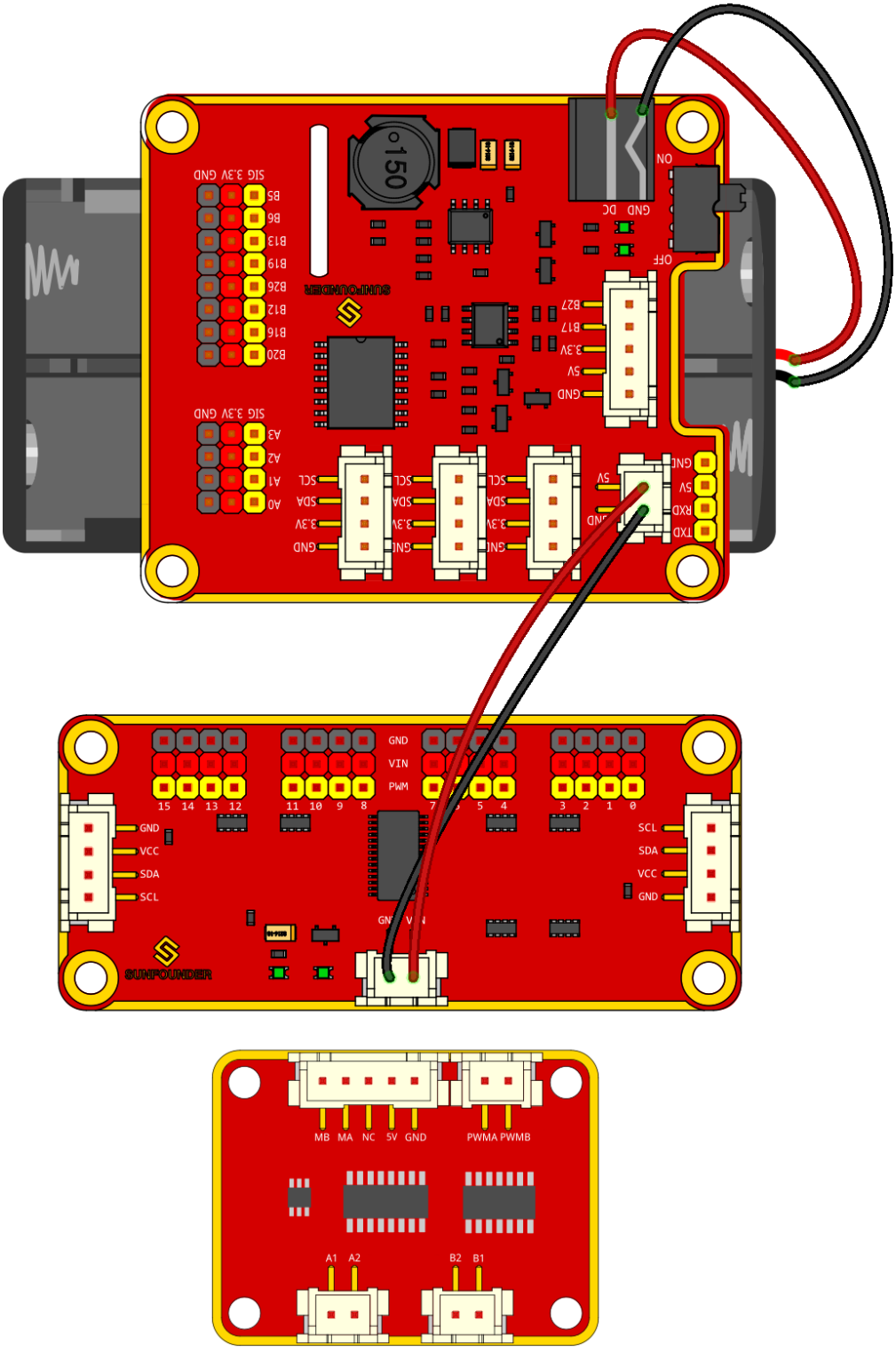


後輪 をモーターシャフトに合わせ、回転させてやさしく差し込む。

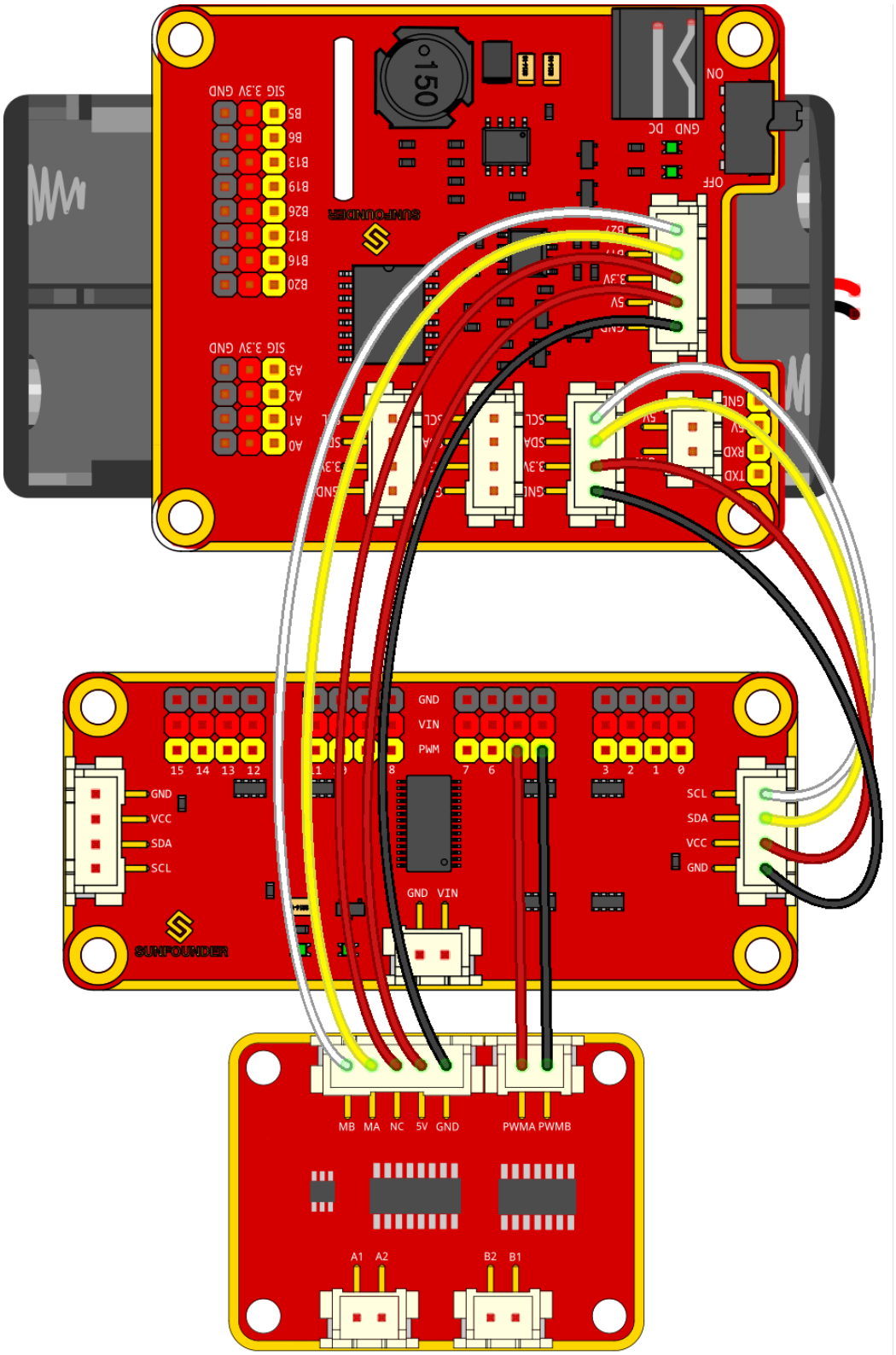


1.4 回路構築

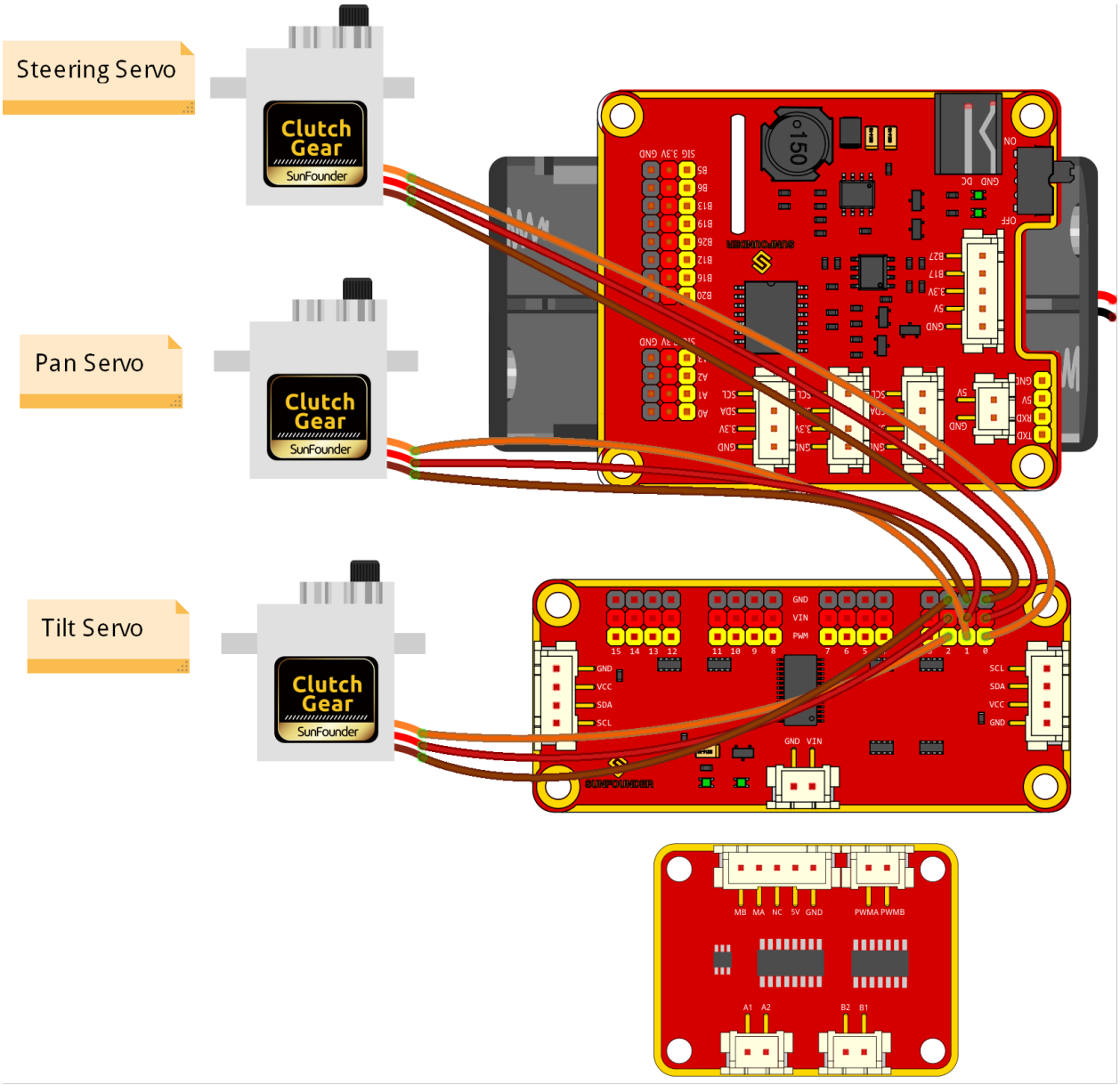
1.4.1 電源を接続する



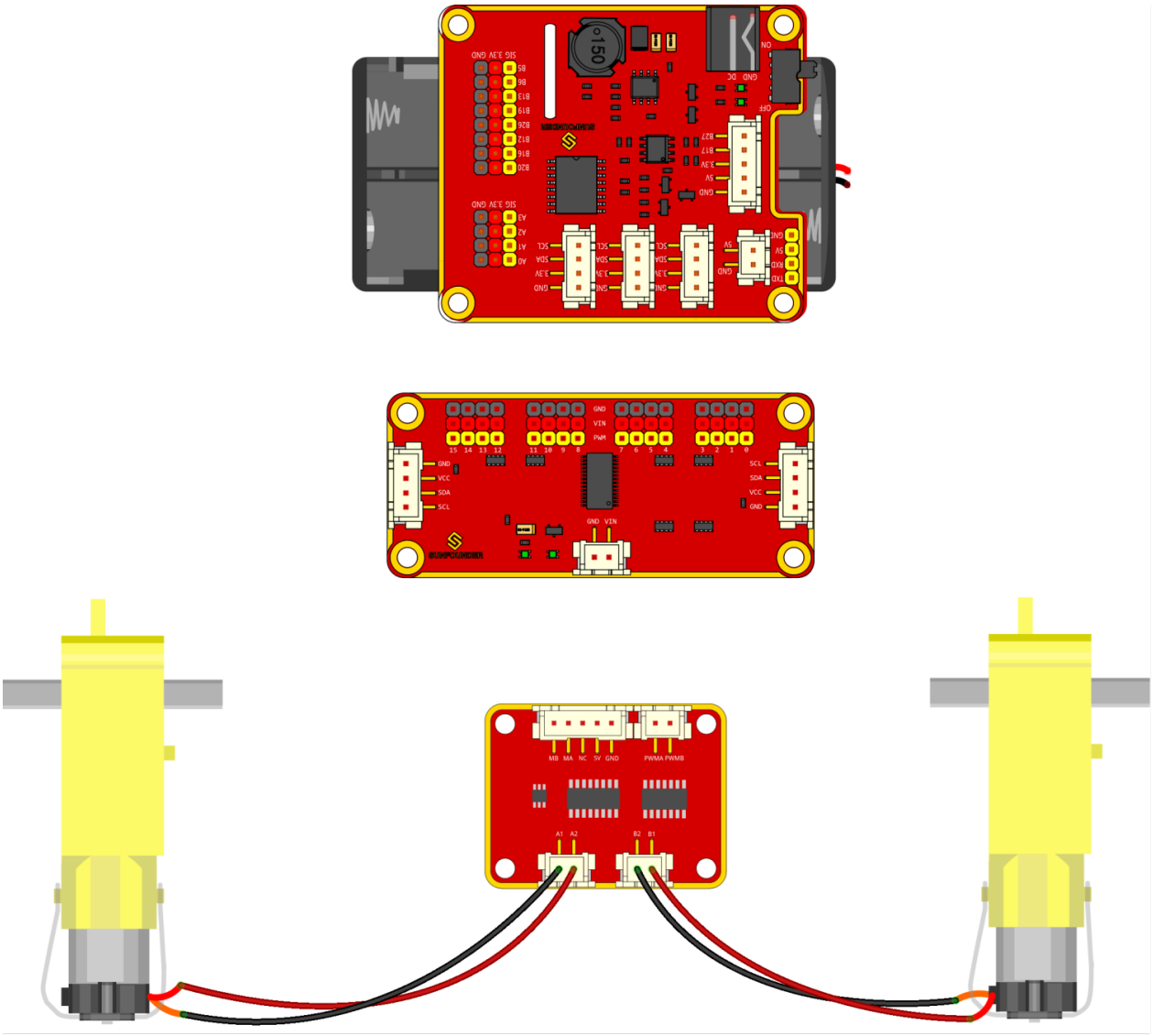
1.4.2 モジュールを接続する



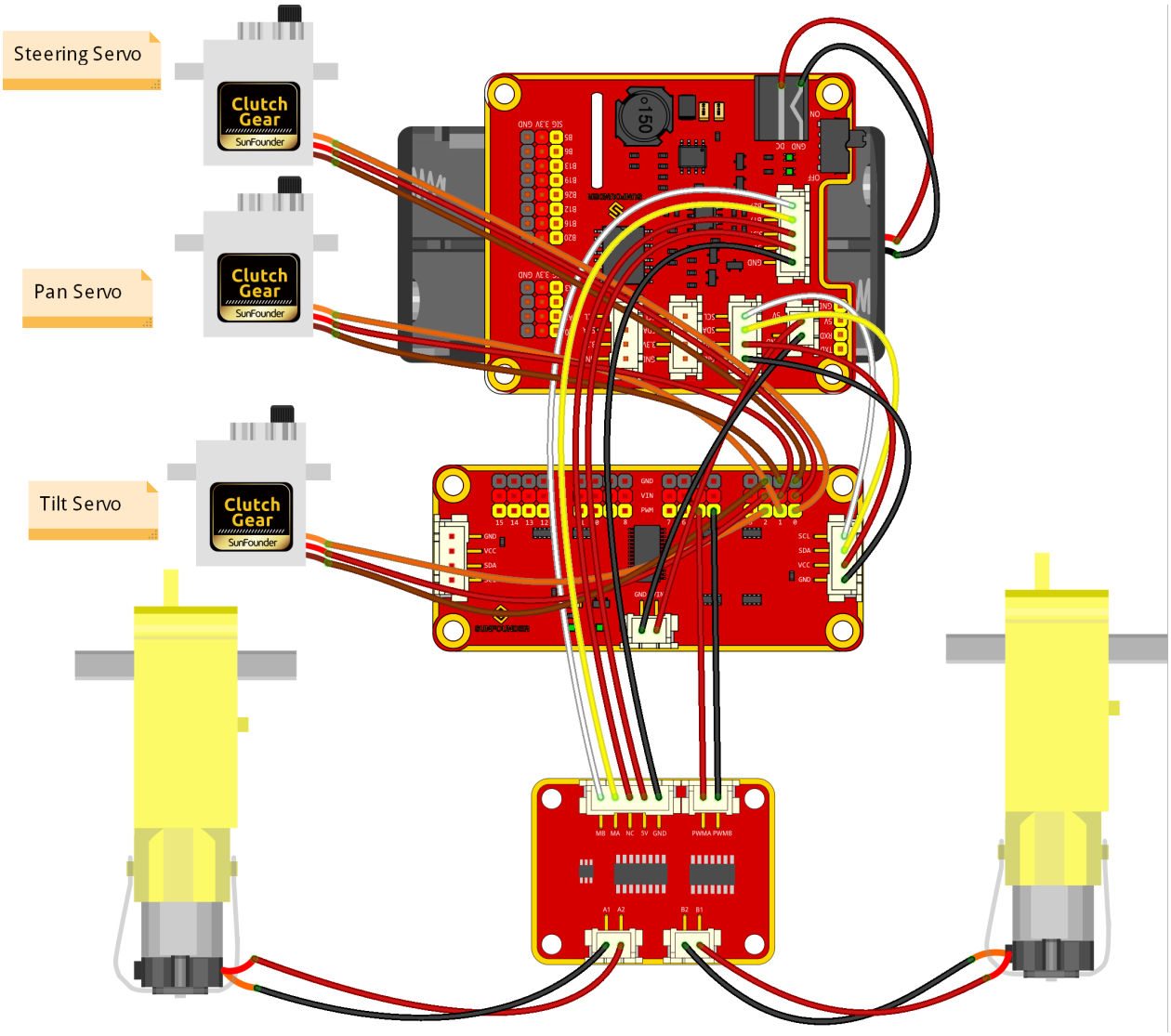
1.4.3 サーボを接続する



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。



1.5 ラズベリー パイを始めよう

この章では、まずラズベリーパイの起動方法を学びます。内容には、オペレーティングシステム、ラズベリーパイネットワークをインストールする方法とターミナルの開き方が含まれます。

注釈:

1. 完全なチュートリアルは、ラズベリーパイの公式サイト（https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up）で確認できます。
2. 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

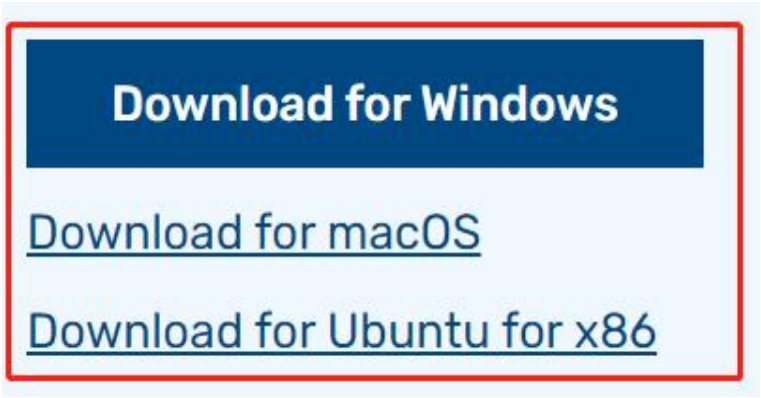
必要なコンポーネント

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。

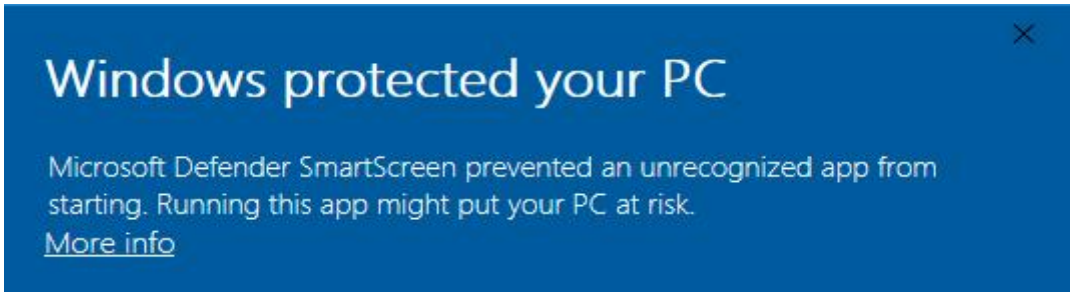
ダウンロードページにアクセスします。https://www.raspberrypi.org/software/。お使いのオペレーティングシステムに合ったラズベリーパイイメージャーのリンクをクリックし、ダウンロードが終了したら、それをクリックしてインストーラーを起動します。



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。

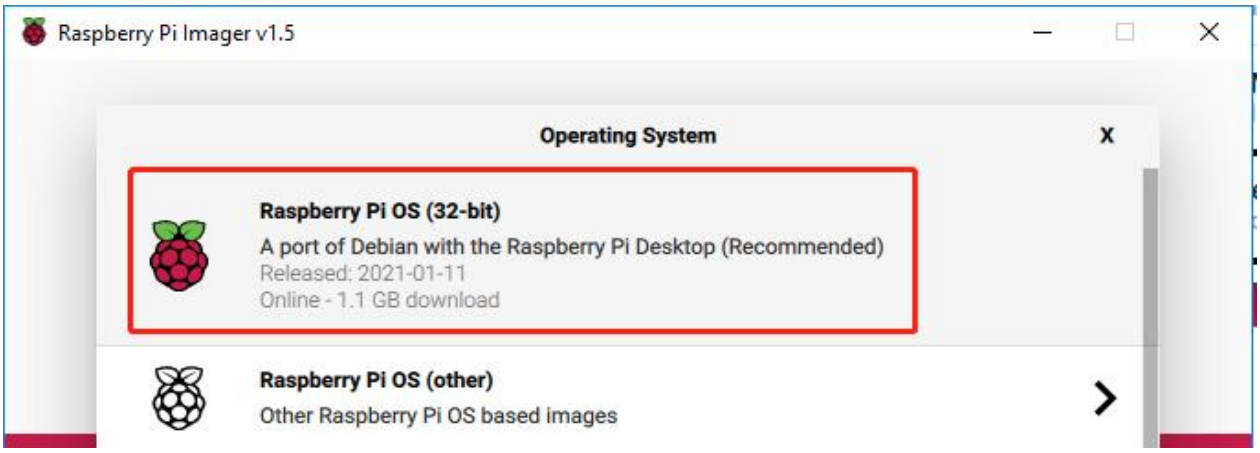


ステップ 3

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。



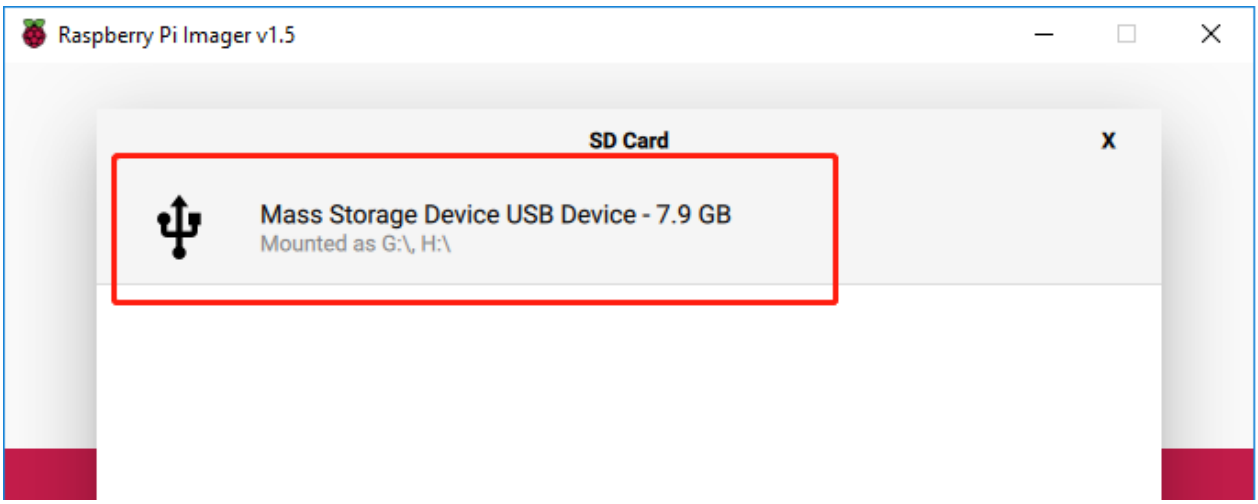
注釈:

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。

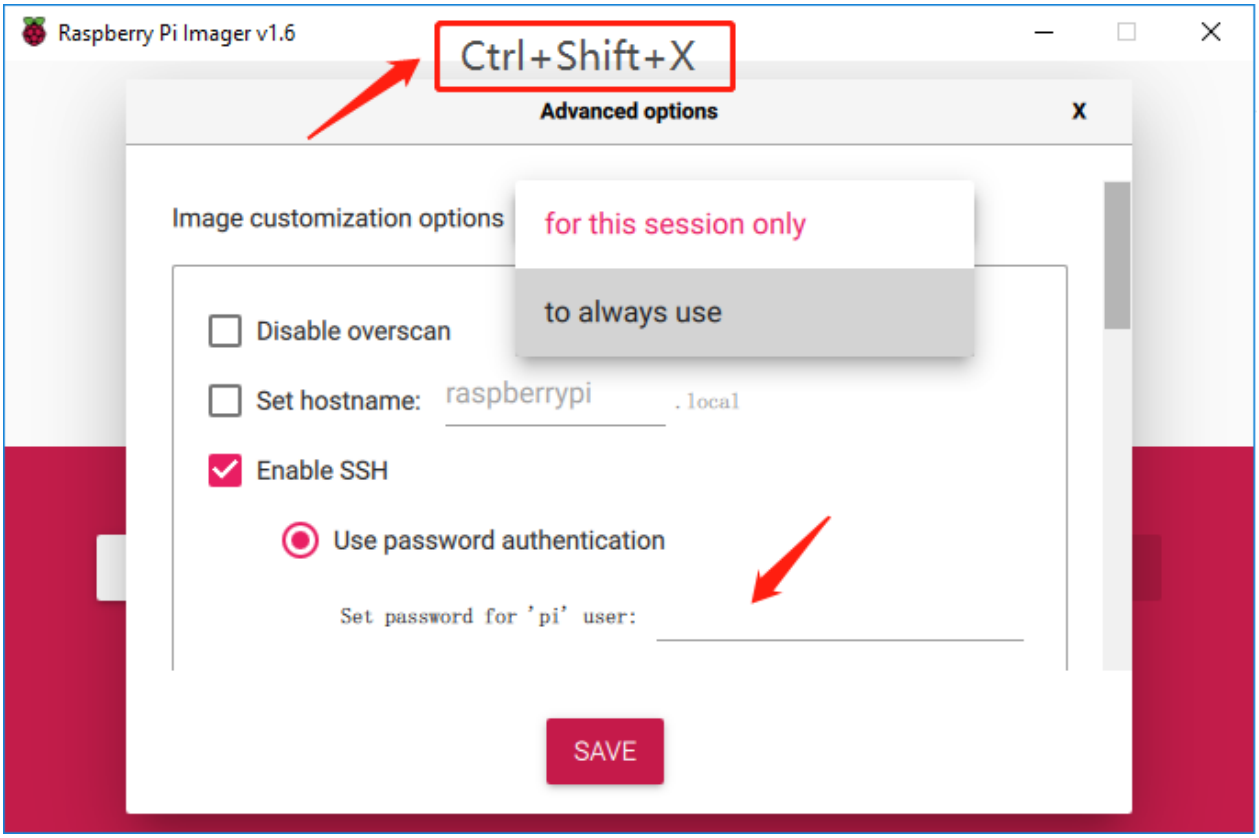
ステップ 5

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。

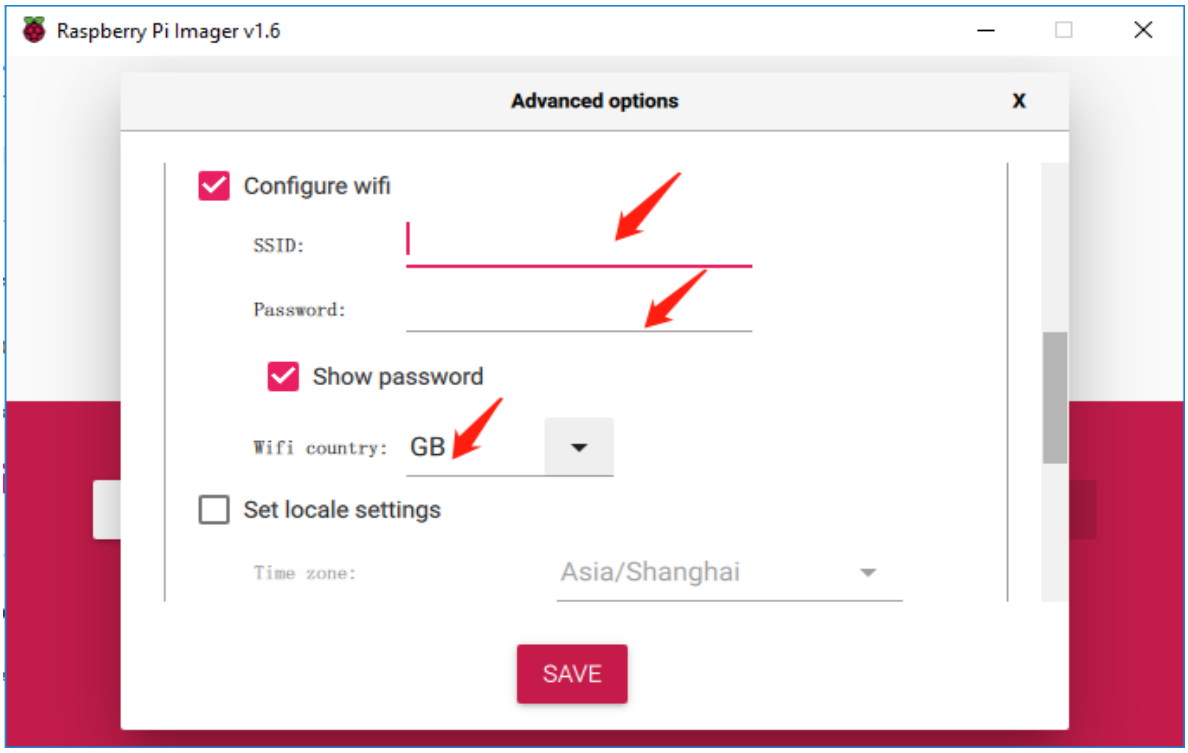


その後、下にスクロールして Wifi の設定を完了し、「**SAVE**」をクリックします。

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：

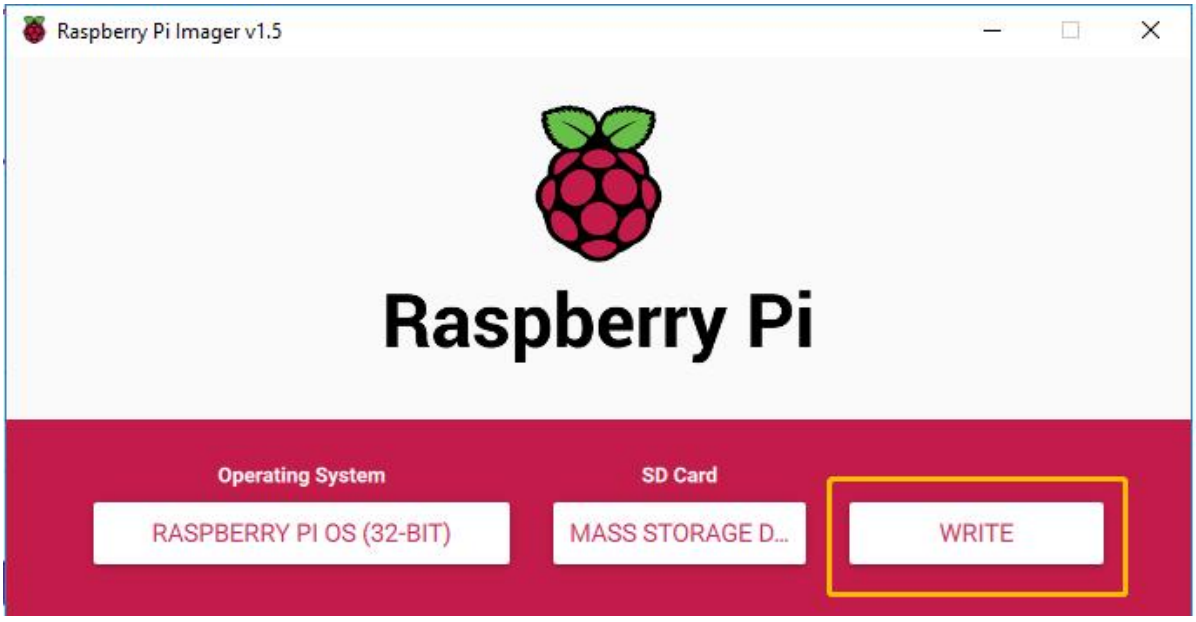
https://en.wikipedia.org/wiki/ISO_3166-1_alpha-2#Officially_assigned_code_elements

---



ステップ 7

「**WRITE**」ボタンをクリックします。

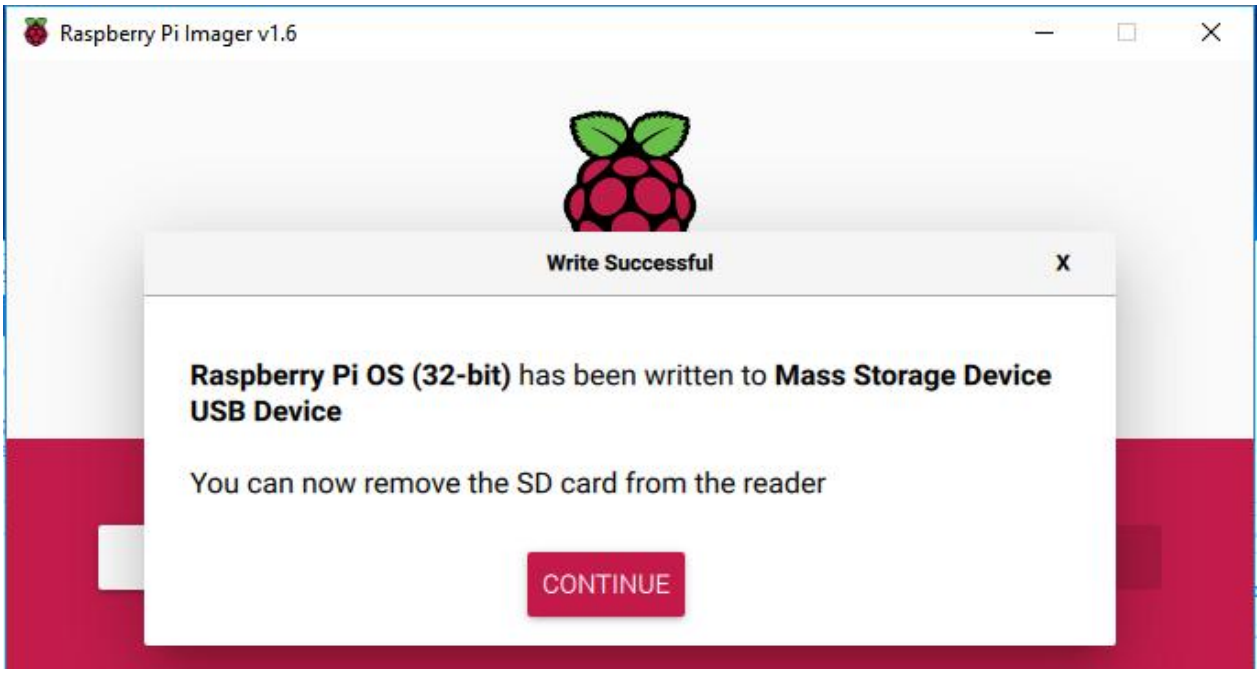


ステップ 8

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。



Raspberry Pi の電源を入れる

USB カードリーダーを接続し、マイクロ SD カードを Raspberry Pi に接続する。

フル充電された 2 つの 18650 バッテリーをホルダーに取り付け、バッテリーホルダーからのワイヤ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る。**最初のテストに時間がかかるため、Raspberry Pi の電源アダプターを使用して車に電源を供給することもお勧めします。**

IP アドレスを取得する

Raspberry Pi の電源を電源アダプターで入れたら、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

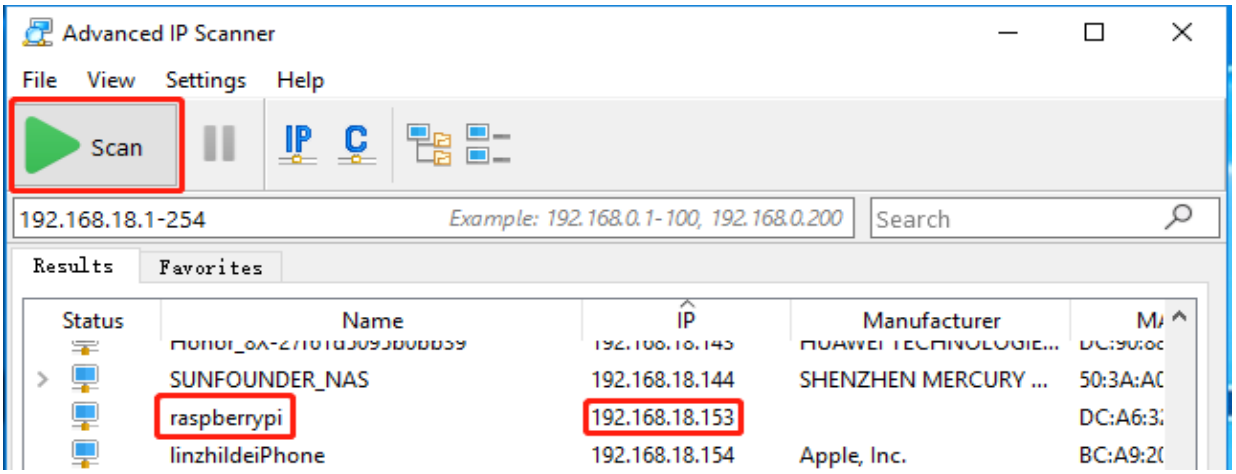
ルーター（ホームネットワークなど）にログインする権限がある場合は、ルーターの管理インターフェイスで Raspberry Pi に割り当てられたアドレスを確認できる。

システムのデフォルトのホスト名ー 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。

「**Scan**」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。



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。

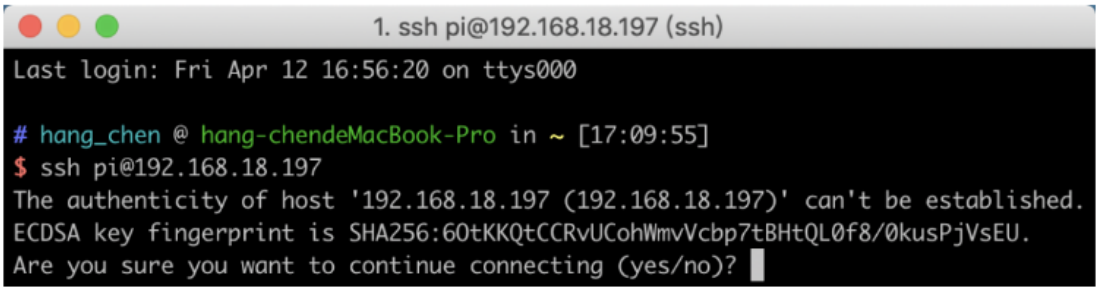
ステップ 2

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
```

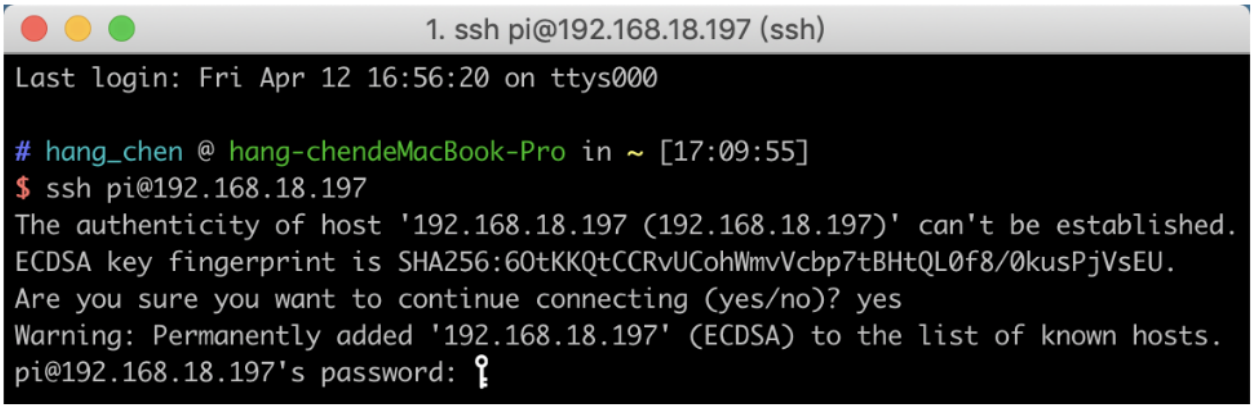
ステップ 3

"yes" を入力する。



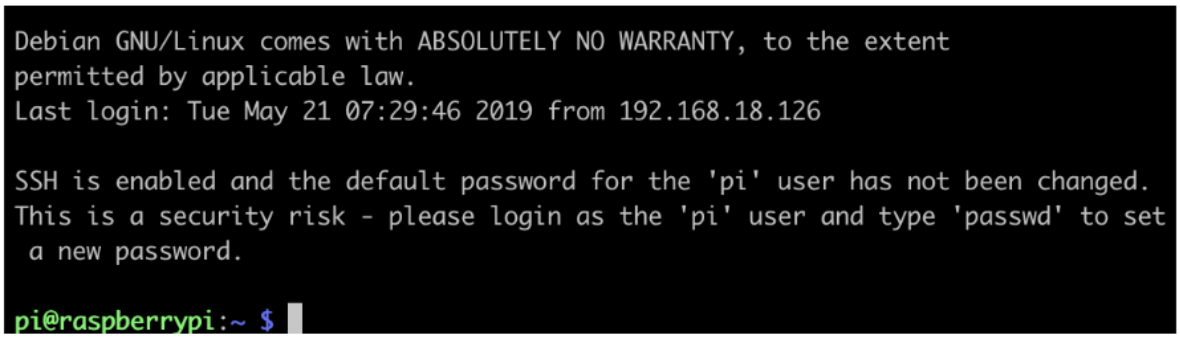
ステップ 4

デフォルトのパスワード「**raspberry**」を入力する。



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。



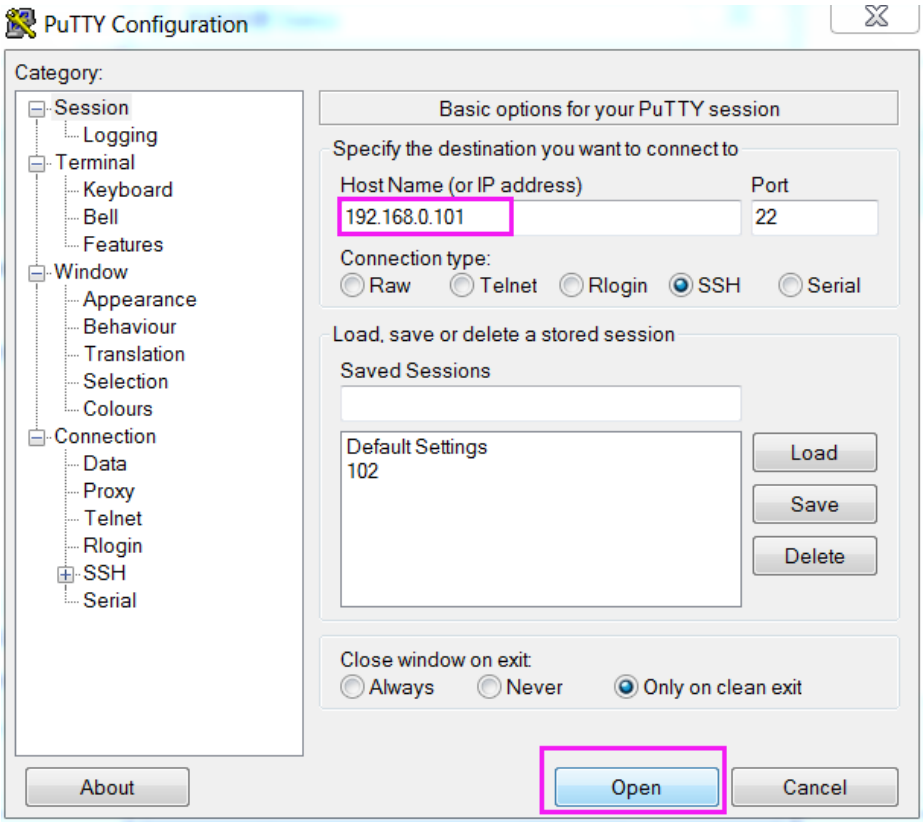
注釈: パスワードを入力すると、ウィンドウに文字が表示されないが、これは正常である。必要なのは、正しいパスコードを入力するだけである。

- **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 を使用できる。ここでは、**PuTTY**、を推薦する（Google からダウンロードできる）。

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」（または IP アドレス）** 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。

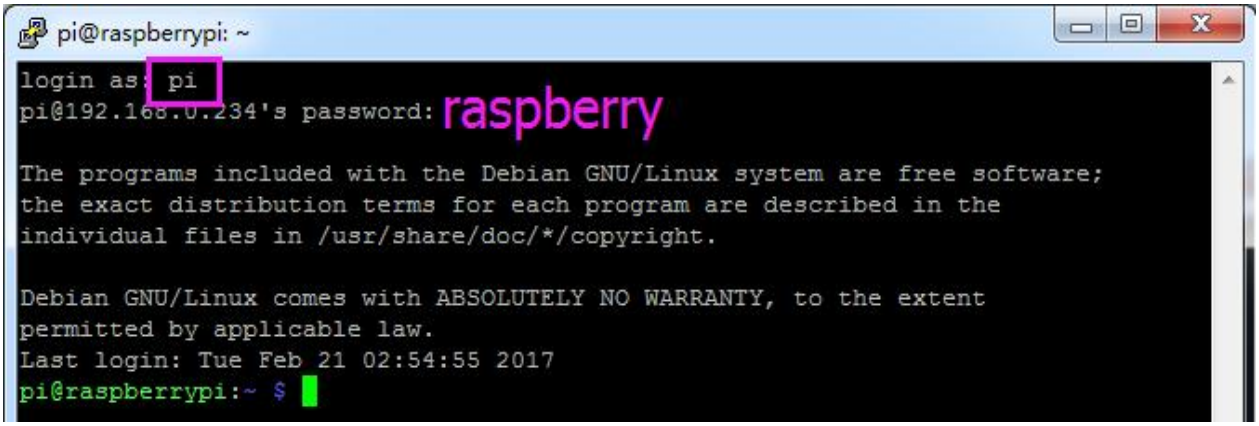


ステップ 2

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****「****raspberry**」（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。



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。

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。

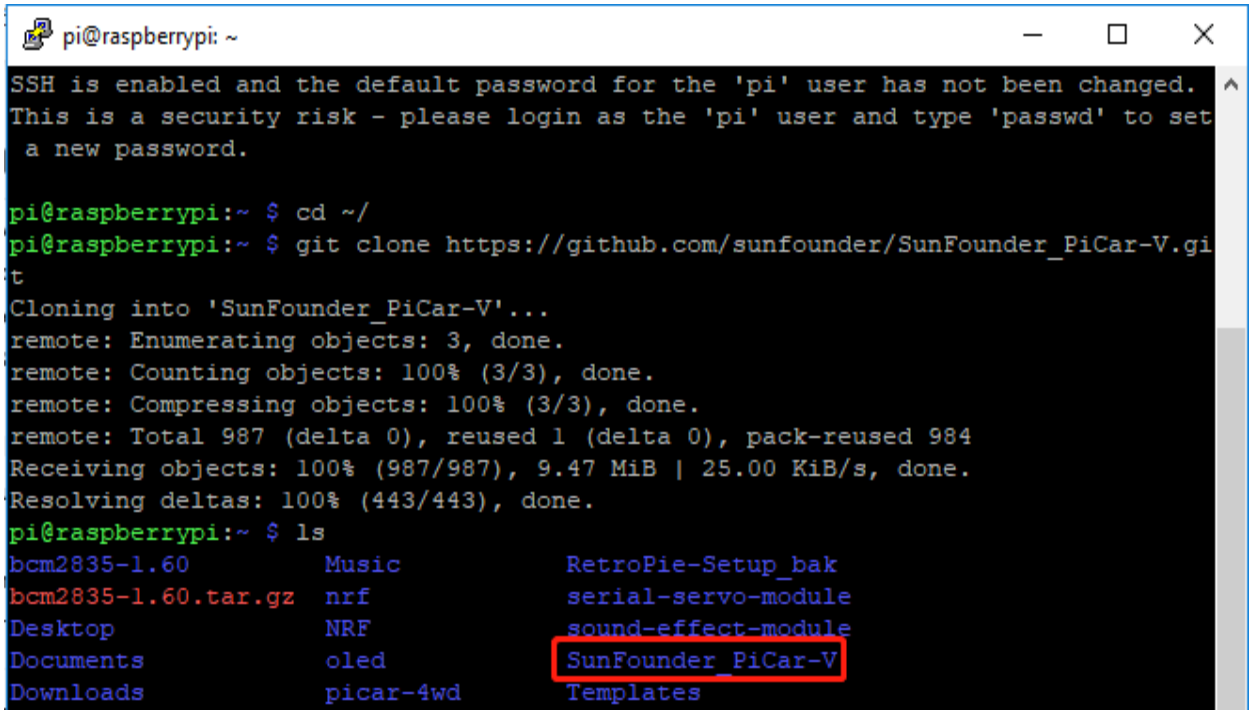
1.6.1 ソースコードを取得する

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 ~/
git clone https://github.com/sunfounder/SunFounder_PiCar-V -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。

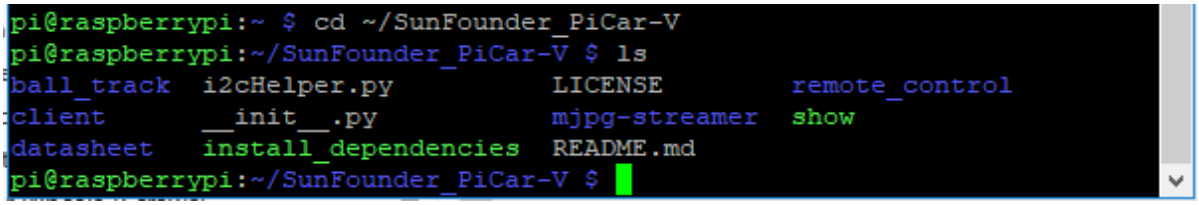
ls コマンドで確認すると、コードディレクトリ *SunFounder_PiCar-V* が表示される：



1.6.2 コードディレクトリに入る

```
cd ~/SunFounder_PiCar-V
```

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：



1.6.3 スクリプトを介して環境をインストールする

install_dependencies 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。代わりに一歩一歩に実行する場合は、「**付録 1：サーバーインストールスクリプトの機能**」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 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、picar ツールを実行する：

```
cd ~/SunFounder_PiCar-V
picar servo-install
```

注釈: If the "OSError: [Errno 121] Remote I/O error" error message appears, open raspi-config:

```
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。

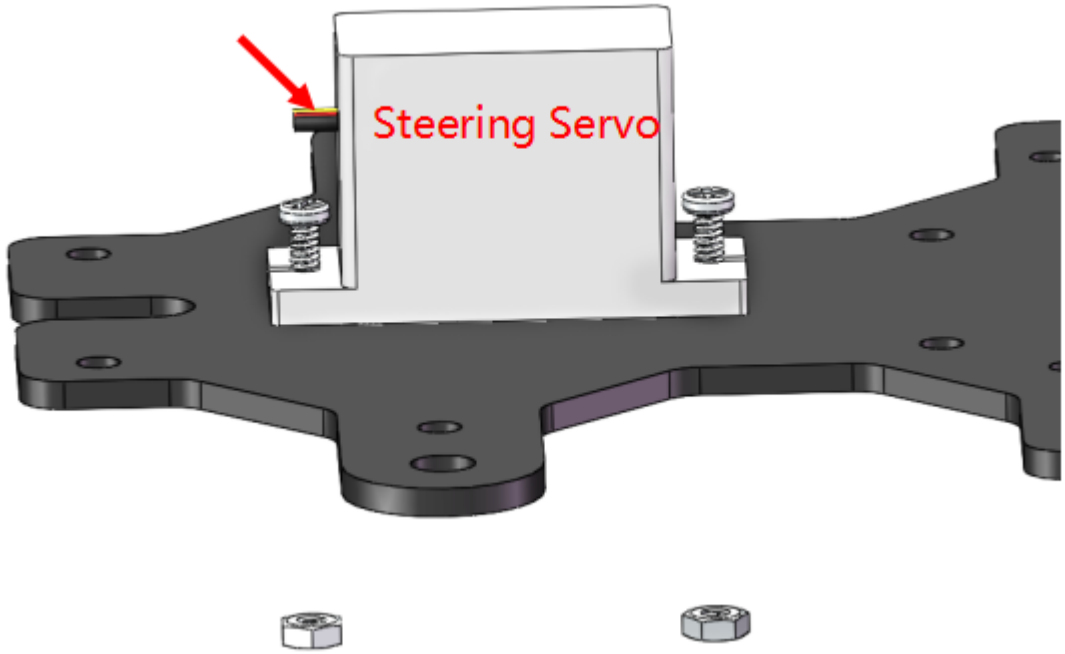
上記の状況のいずれかが発生した場合は、 service@sunfounder.com <mailto:support@sunfounder.com>`__に送信してください。新品に交換してあげる。使用または組み立ての過程で破損した場合は、公式ウェブサイト `www.sunfounder.com にて購入してください。

1.7 組み立てを続ける

> **警告:** バッテリーを取り付けて電源スイッチを ON にスライドさせてから、組み立てのプロセス全体でサーボインストールを実行し続けてください。

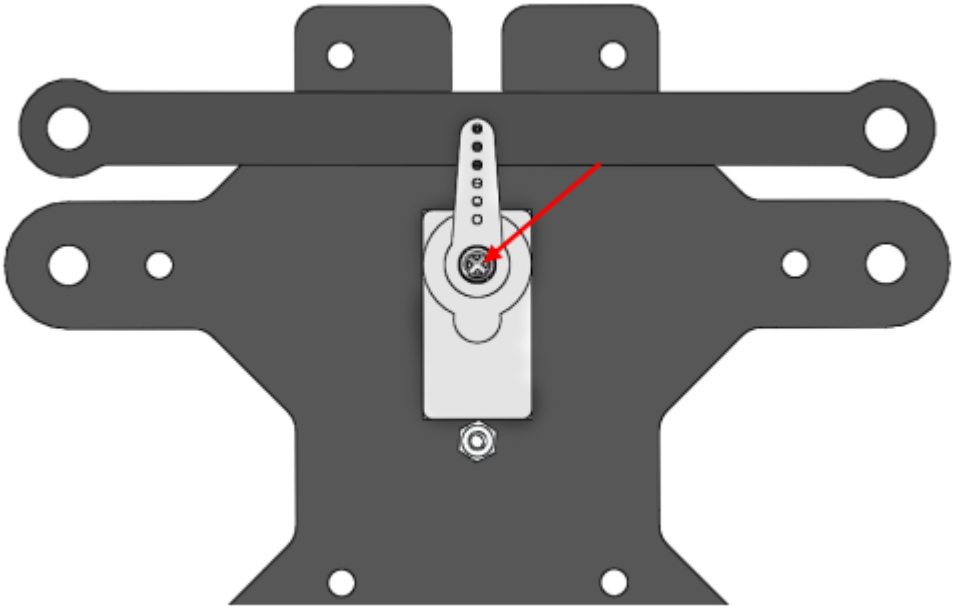
1.7.1 ステアリングサーボの組み立て

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）。

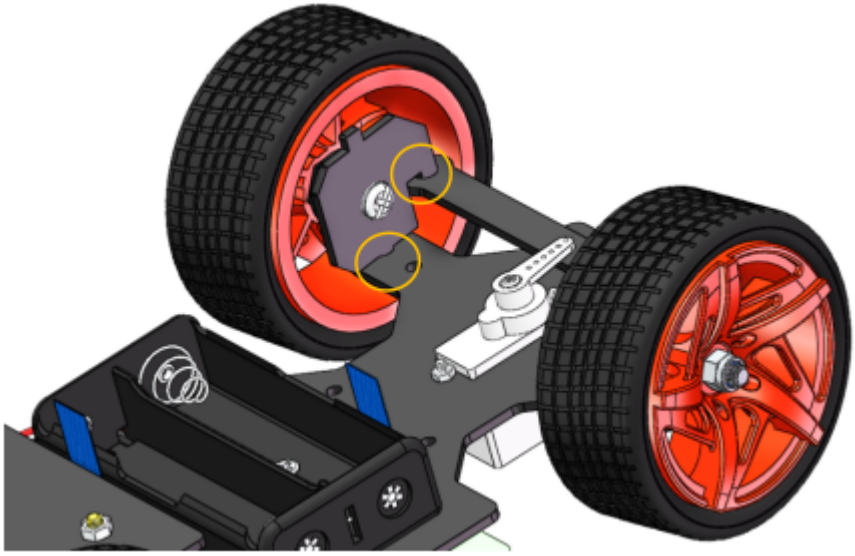


1.7.2 フロントハーフシャーシ

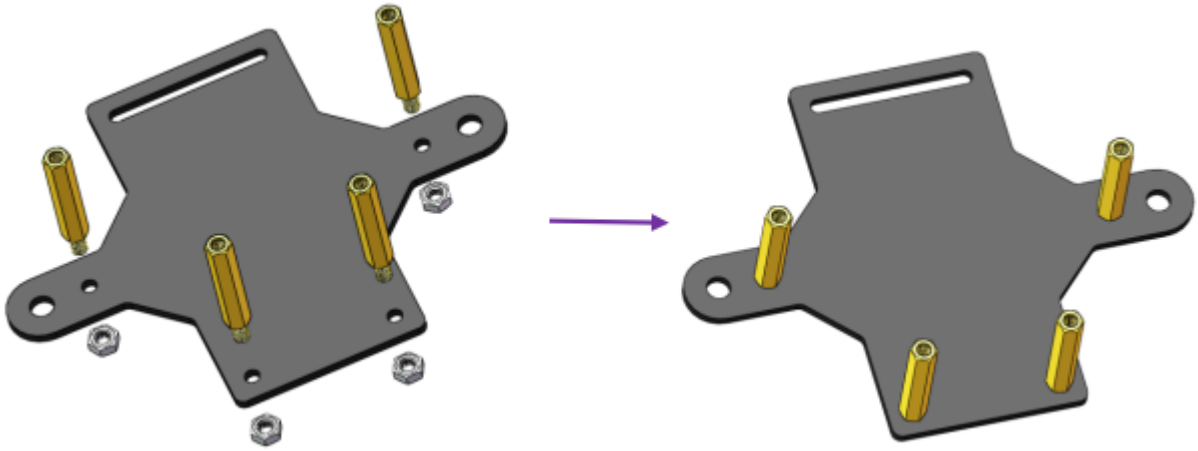
ロッカーアーム固定ネジ（最短）で **ステアリングリンケージ** をロッカーアームに接続する。



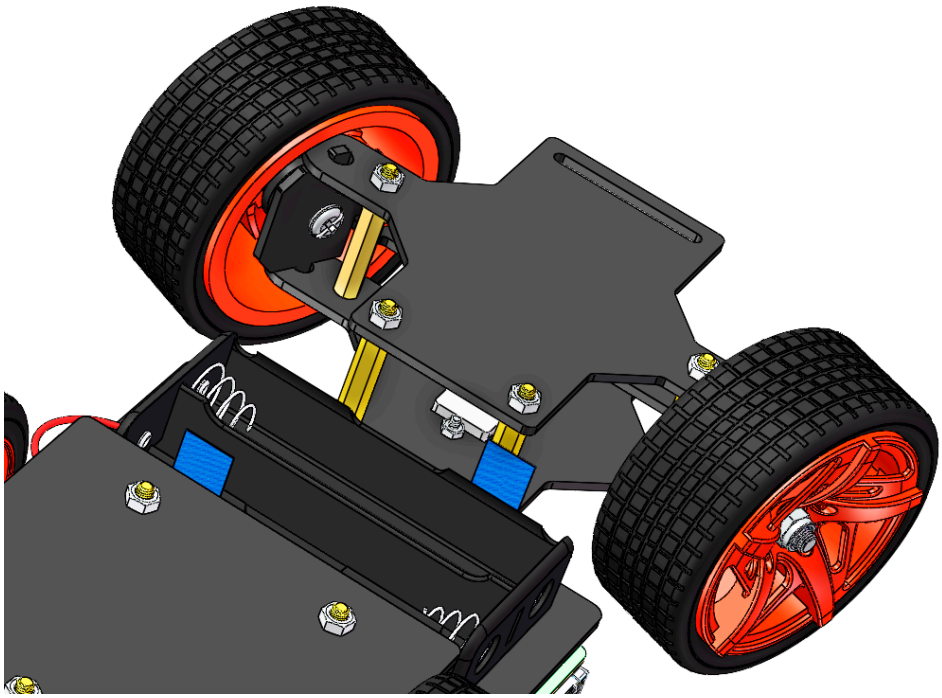
ホイールを上部プレートに入念に取り付ける。



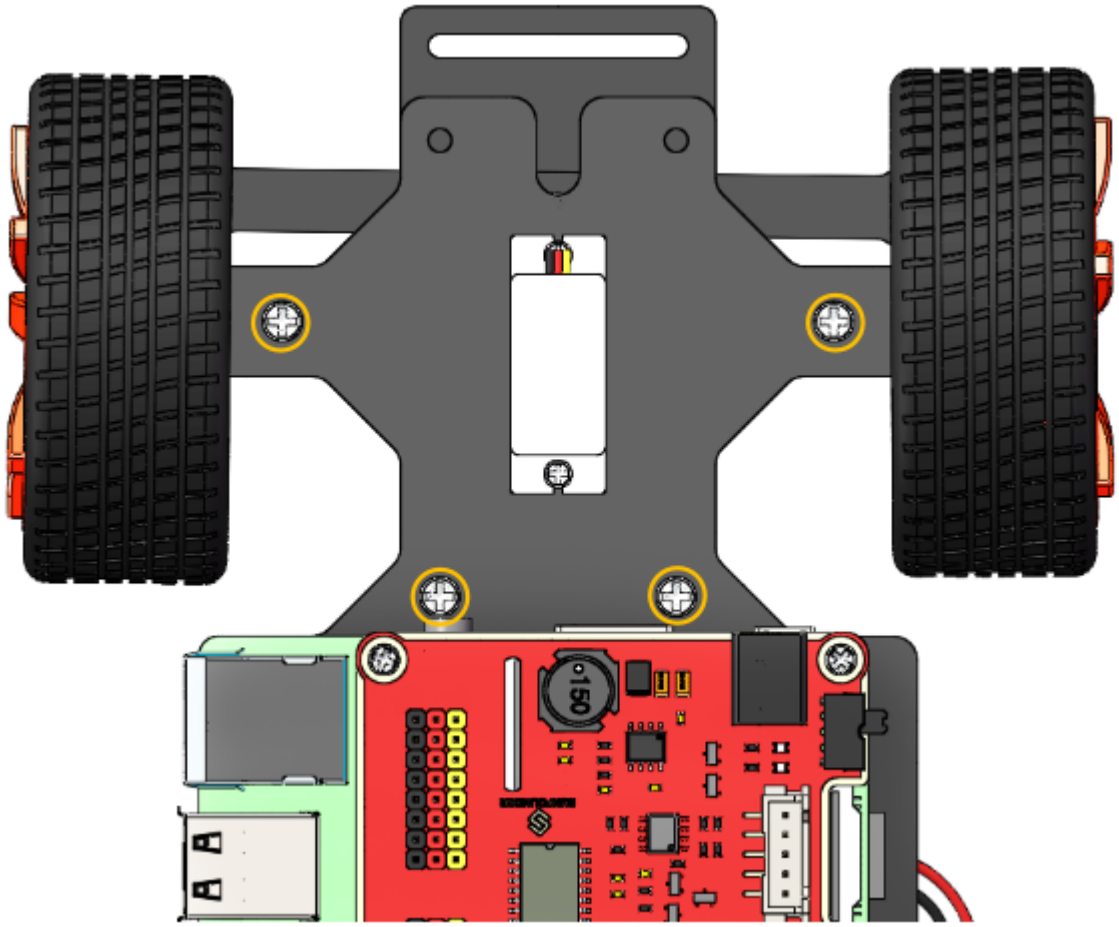
4 つの **M3x25 銅製スタンドオフ** と **M3 ナットでフロントハーフシャーシ** を組み立てる。



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。

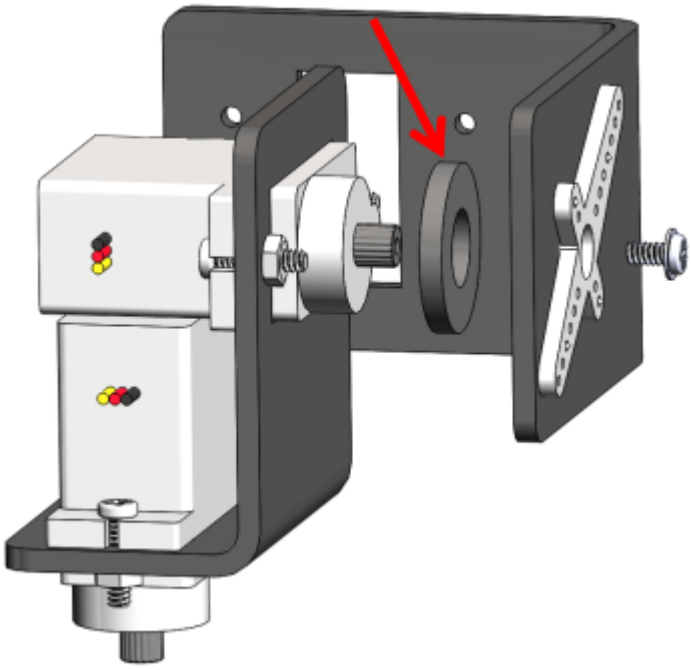


入念に持ち、上下を逆にして、4 本の **M3x8 ネジ** でスタンドオフと上部プレートを固定する。

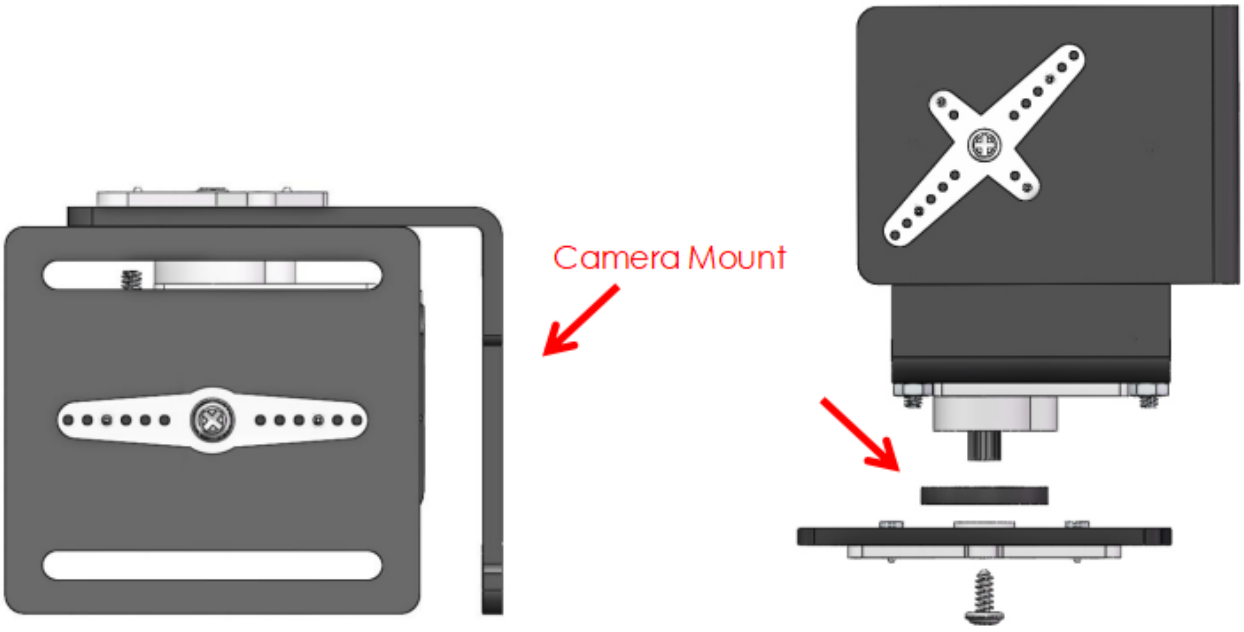


以下に示すように、組み立てられた **パン＆チルトプレート**、**ガスケットプレート** と **カメラマウントプレート** を **ロッカーアーム固定ネジ** で固定角度で組み立てる:

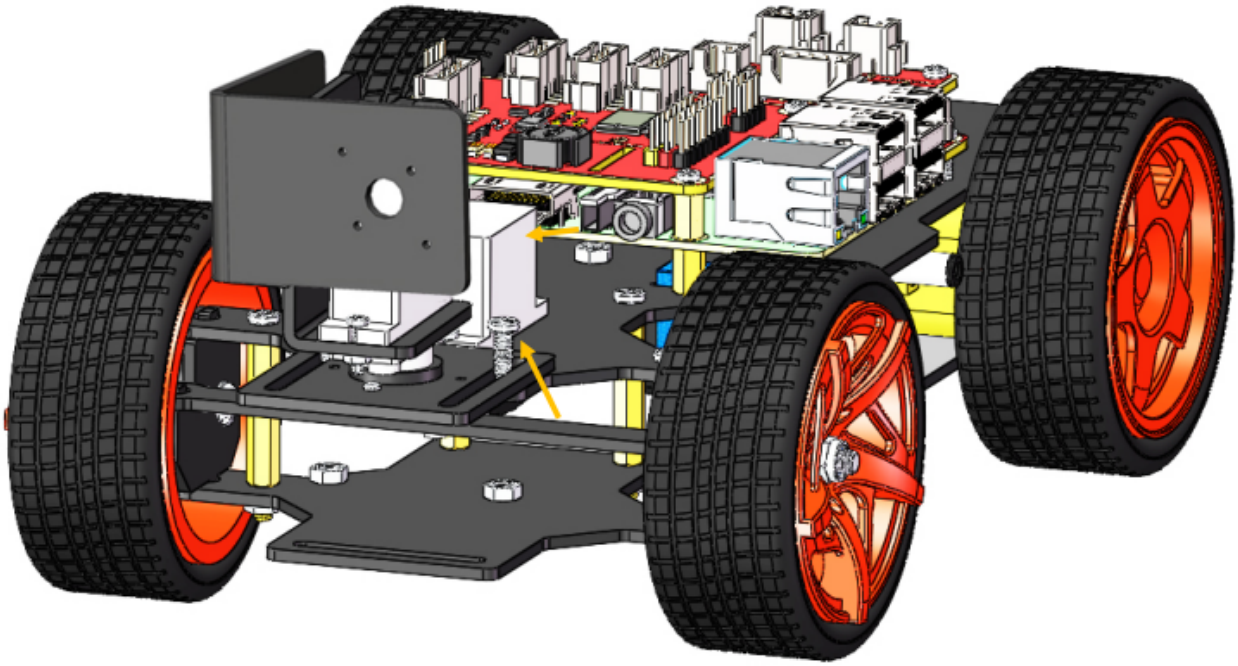
注釈: サーボが故障した場合、手でサーボを回さないでください。



同様に、サーボシャフトが 90 度に回転していることを確認する。次に、以下に示す角度で、**パン＆チルトベースプレート** を **ロッカーアーム固定ネジ** と **ガスケットプレート** で組み立てる。



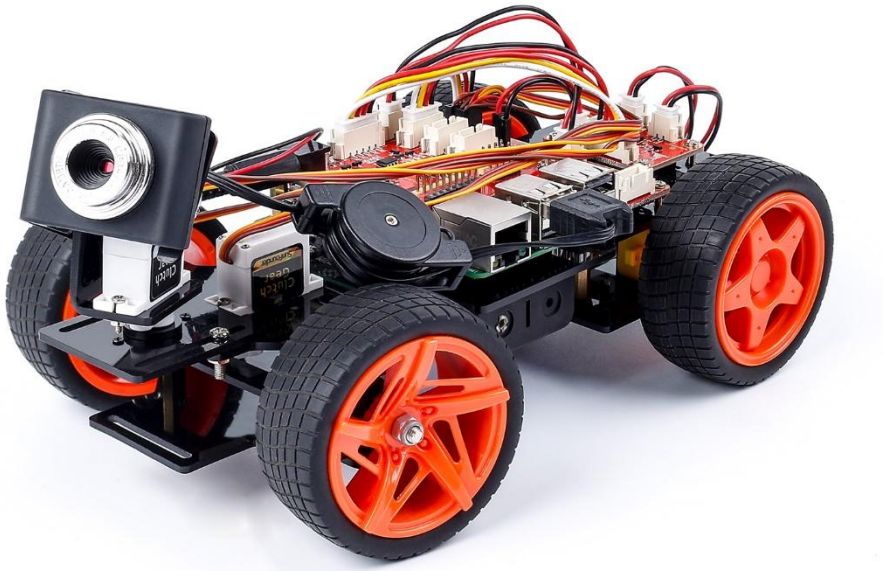
2 つの **M3x10 ネジ** と **M3 ナット** を使用して、**パン＆チルトプレート** を車に取り付ける。



1.7.3 カメラを組み立てる

カメラを取り出し、カメラマウントに鋏む。USB ケーブルを Raspberry Pi の USB ポートに接続する。

これで、アセンブリ全体が完了した。おめでとう！今すぐ車の電源を切ることができる！バッテリーを充電することを忘れないでください。



1.8 旅に出よう！

すべての手順が完了していて、機械的なアセンブリとソフトウェアのインストールに問題がないことを確認してください。

1.8.1 遠隔操作

これはこれから行うことである：

Raspberry Pi を **server** として使う。車を制御し、カメラでキャプチャした画像を送信するための API を使用して Web サーバーを実行する。

次に、PC、携帯電話、タブレットを **client** として、カメラから画像を取得し、Web サーバーの API を呼び出して車を制御します。

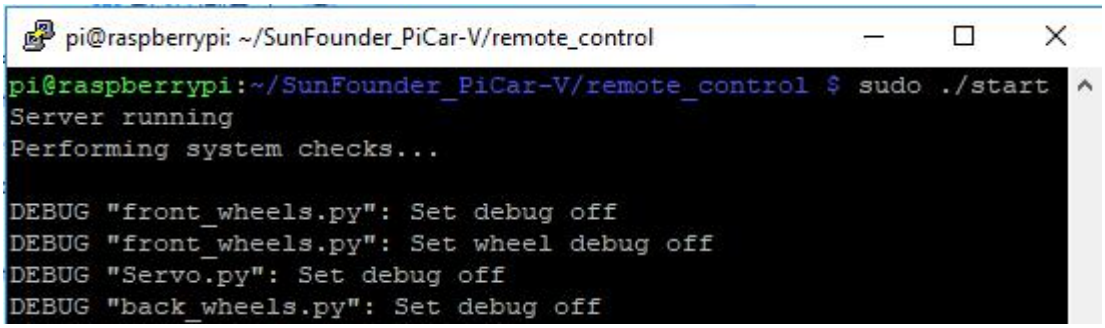
その後、再び車の電源を入れてください。最初のテストに時間がかかるため、Raspberry Pi の電源アダプターを使用して車に電源を供給することも勧める。

サーバーを実行する（Raspberry Pi での操作）

Raspberry Pi にリモートでログインする。**remote_control** ディレクトリの下で起動スクリプト **start** を実行して、Web サービスを開始する。

```
cd ~/SunFounder_PiCar-V/remote_control
python3 manage.py migrate
sudo ./start
```

スクリプトがサービスを有効にし、対応するデータが表示される。同時にハードウェアが初期化されるため、前輪に接続されているサーボとパン＆チルトが回転し、ハードウェアの初期化が完了したことを示す。

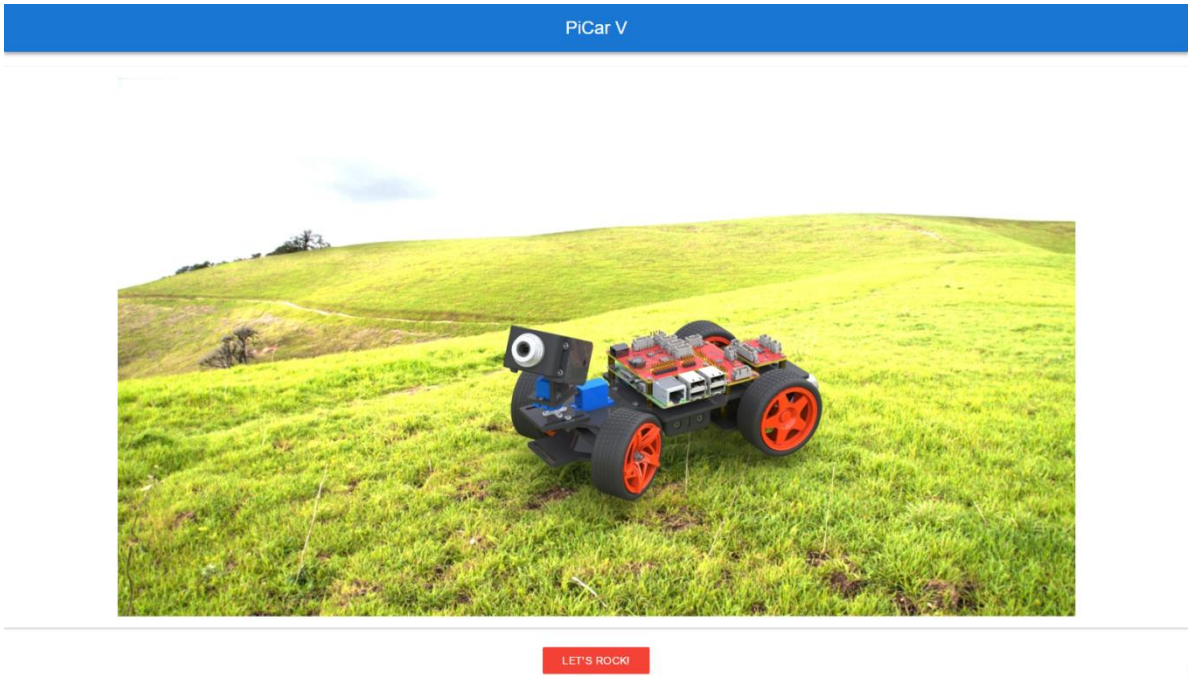


上記のような結果が得られれば、サーバーは準備できた。次に、クライアントを起動する。

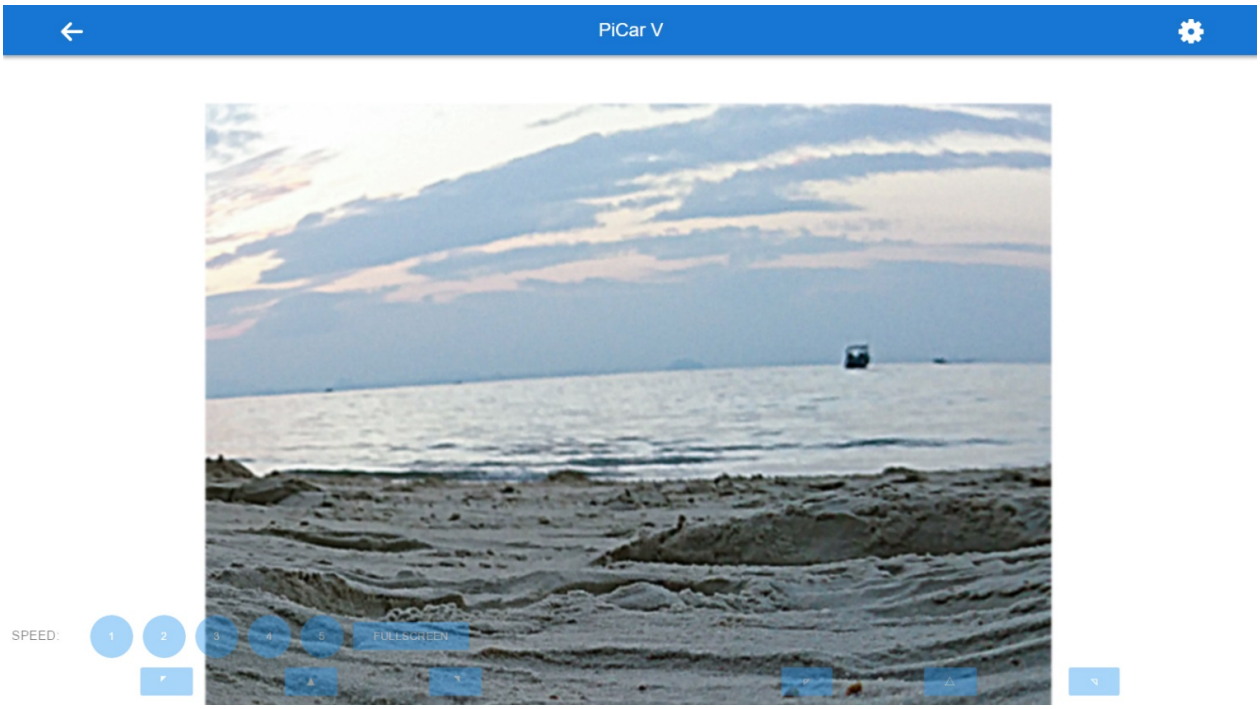
警告: クライアントの実行を停止するまで、サーバーをいつも動作させる。

クライアントの実行（PC での操作）

`http://<RPi_IP_address>:8000/` で車のサーバに訪問する。ウェルカムページが表示される：



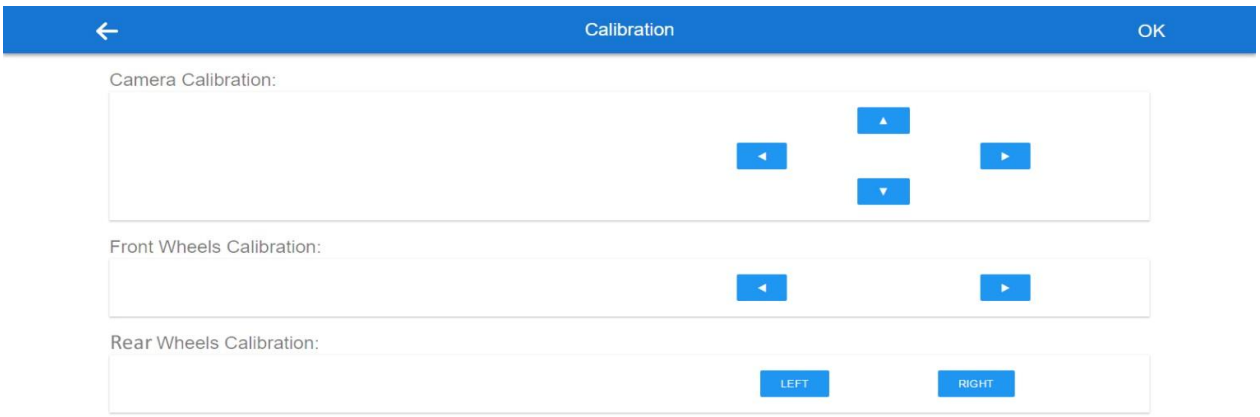
LET'S ROCK をクリックして、操作インターフェイスに入る。



このページでは、キーボードの **W**、**A**、**S**、**D** キーを押して、車を**前方**、**後方**、**左折**、**右折** に制御したり、矢印キーを押してカメラの移動を制御したりすることができ、数値 **1〜5** は速度レベルを変更する。

補正

もう一度 **FULLSCREEN** をクリックして、タイトルバーを表示する。次に、ページの右上隅にある設定ボタンをタップして、補正ページに進む。



補正には、**カメラ補正**、**前輪補正**、**後輪補正** の 3 つの部分が含まれている。

このページに入ると、車が前進する。そうでない場合は、**後輪補正** の **左**と **右** をクリックして、ホイールの角度を調整する。

注釈: ボタンを押すたびに角度が少しずつ変化する。時間あたりの変化を大きくするには、ボタンを長押ししてく

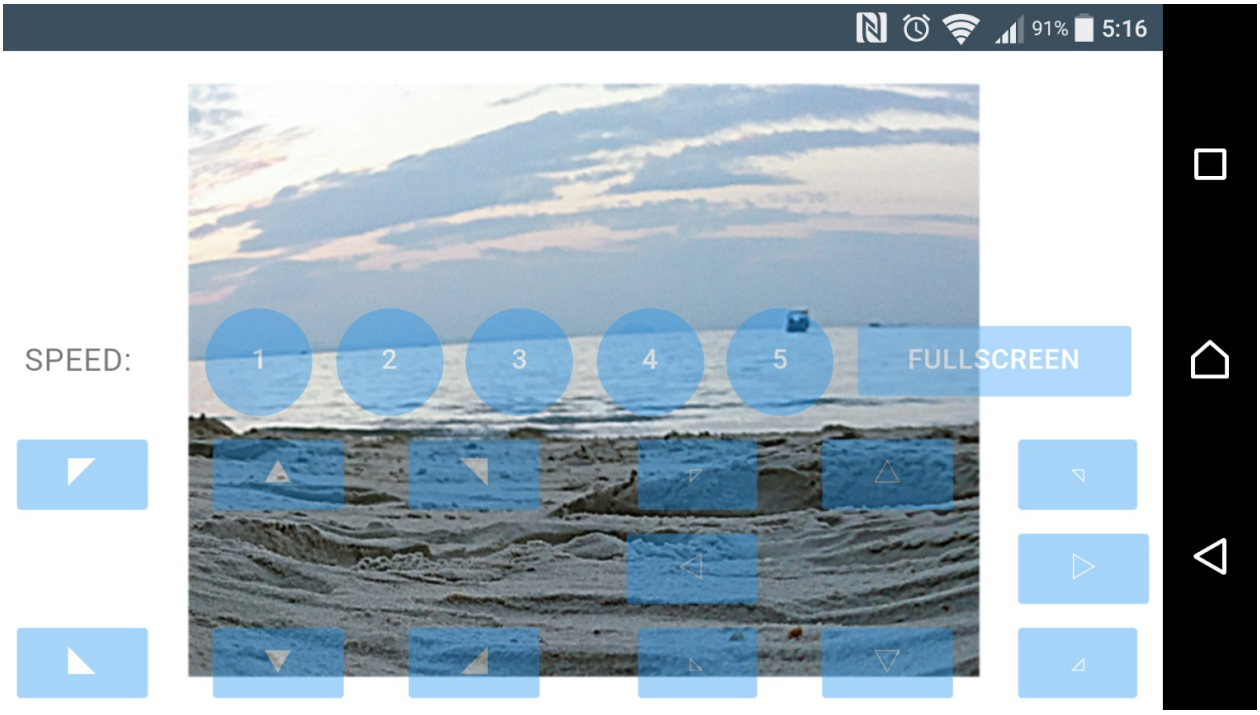
ださい。

すべての補正が完了したら、**OK** をクリックして結果を保存する。

携帯電話用

また、携帯電話の場合は、**FULLSCREEN** ボタンをタップすると、表示とパフォーマンスが向上する。次に、ページ上の 5 つの速度レベルのボタンをタップして速度を制御し、矢印ボタンを押して車の方向とパン＆チルトを制御する。ただし、**一度** にタブできるのは 1 つのタッチポイントのみである。

Android 操作システムのスマートフォンから次のスクリーンショットを取得する。



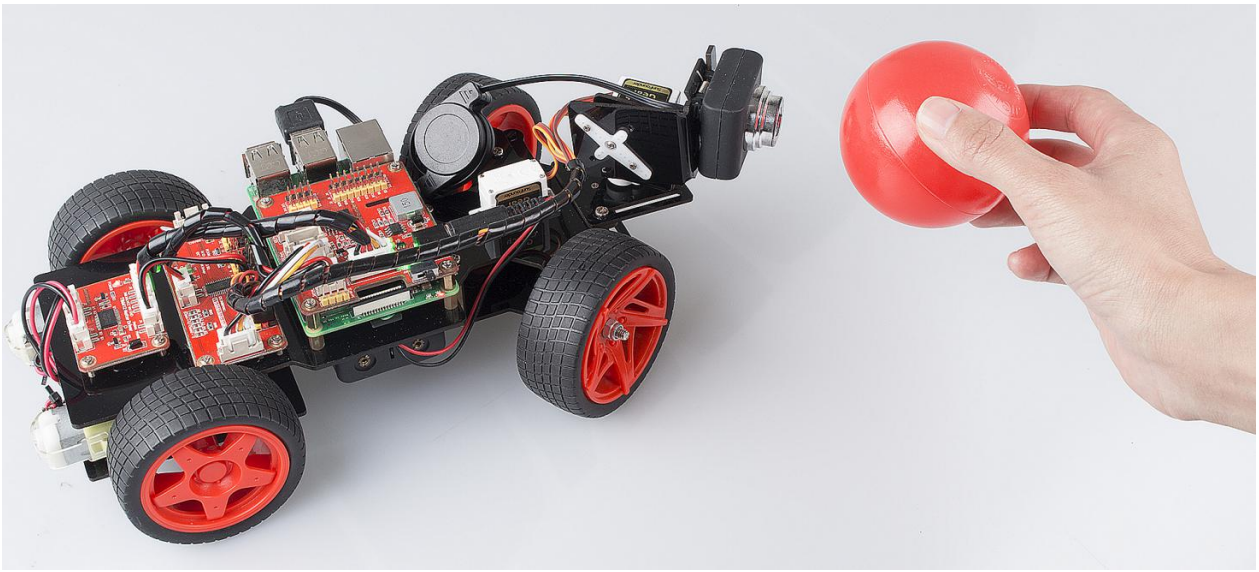
見た目は PC ほど良くないかもしれないが、以降の更新は **Github** で不定期にリリースされる可能性がある。リポジトリを使い、変更を加えたプルリクエストを送信してください。テスト後に問題がなければ、リクエストを知ることは楽しみである。

1.8.2 ボール_追跡装置

キーボードを使用して車の状態を制御するだけでなく、ボールトラッキングコードも作成した。

```
sudo apt-get install libatlas-base-dev libjasper-dev libqt4-test libwebp6 libtiff5␣
↪libopenexr23 libgstreamer1.0-0 libavcodec-dev libavformat-dev libswscale-dev␣
↪libqtgui4 -y
sudo pip3 install opencv-python==3.4.6.27
cd ~/SunFounder_PiCar-V/ball_track
python3 ball_tracker.py
```

コードが実行されたら、赤いボール（少なくともカメラの直径よりも大きい）を見つけ、カメラの前の 10〜50cm の範囲内に配置して、車があなたのボールを追跡するようにする。

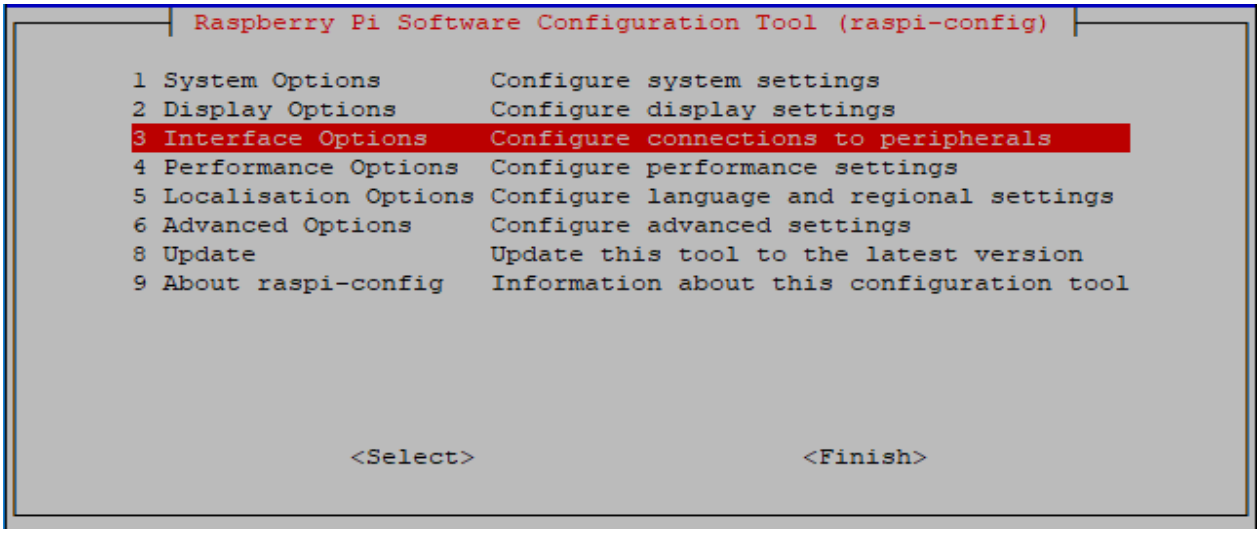


車があなたのボールを捕まえたかどうかを知りたい場合は、以下の手順を実行してください：

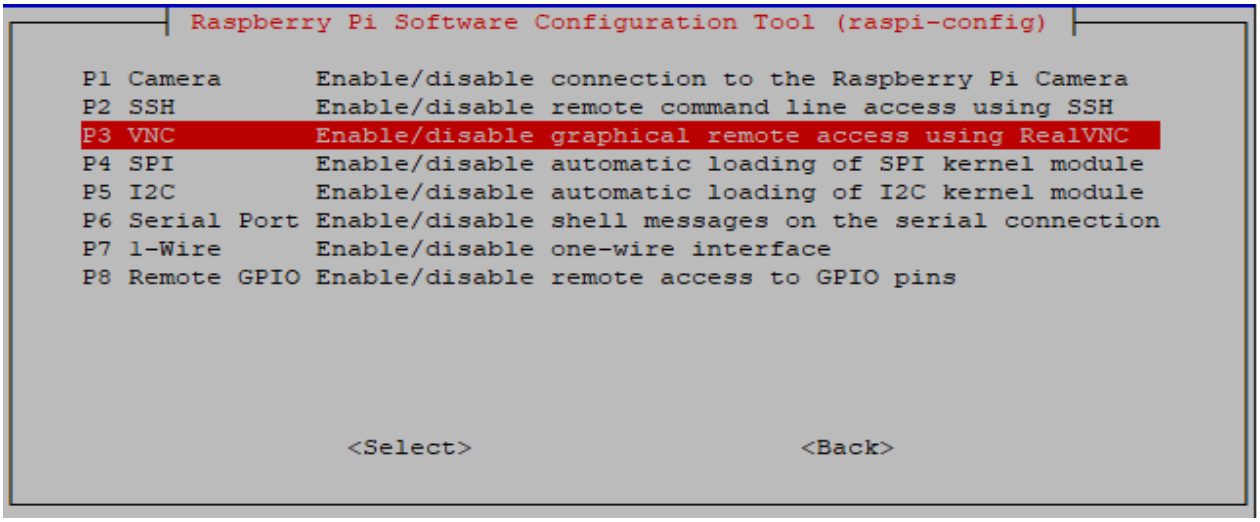
ステップ 1： Raspberry Pi で VNC を有効にする。

```
sudo raspi-config
```

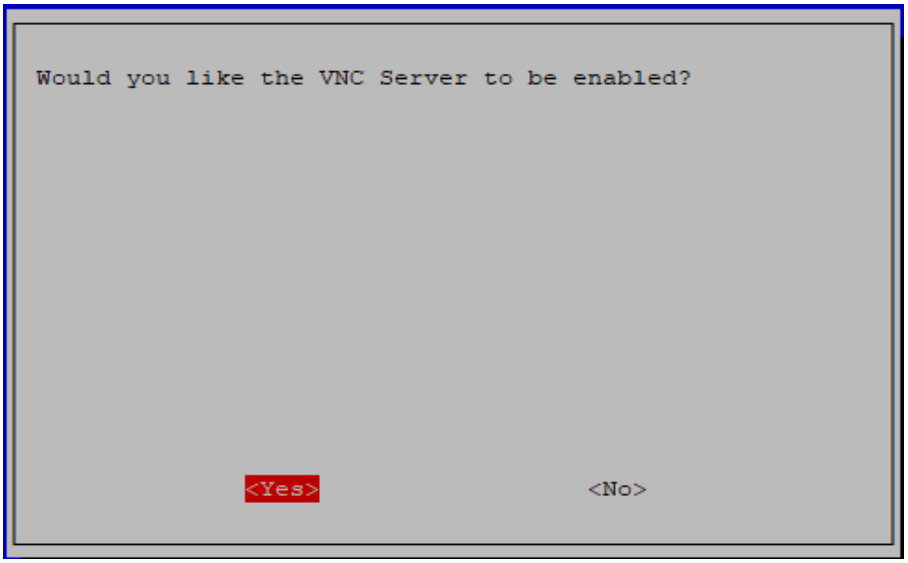
3 Interfacing Options を選択する。**Enter** キーを押してください。



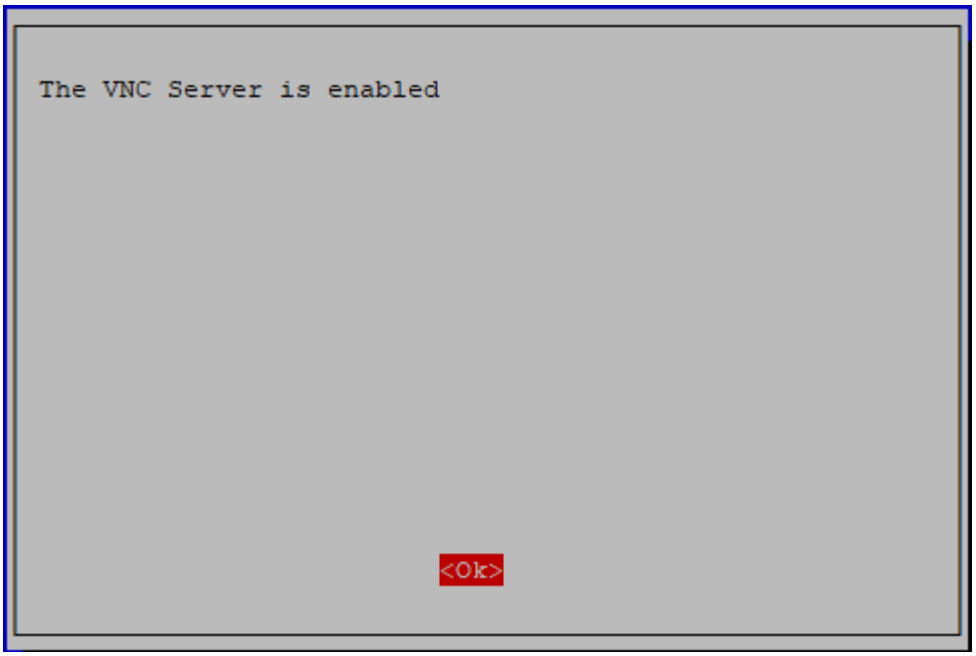
P3 VNC



キーボードの左矢印キーを押して、**はい** を選択します。



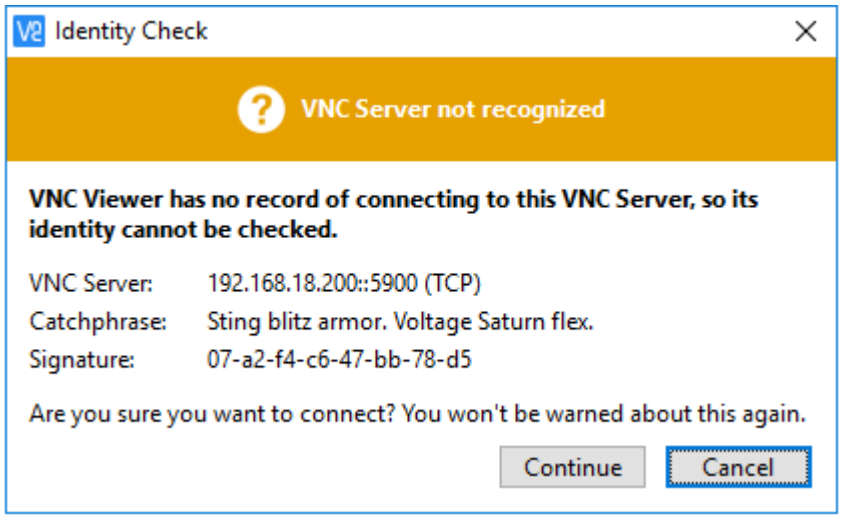
最後に **OK** -> **Finish** を選択して、構成を終了する。



ステップ 2： お使いのコンピューター用の VNC ビューアーをダウンロードしてインストールする。

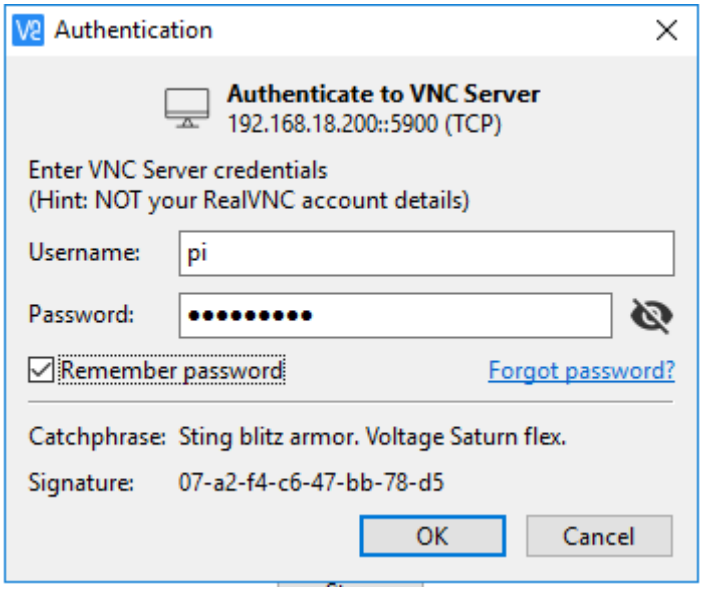
https://www.realvnc.com/en/connect/download/viewer/

注釈: Windows への実装中に次のプロンプトが表示された場合は、**Continue** をクリックしてインストールを続行してください。

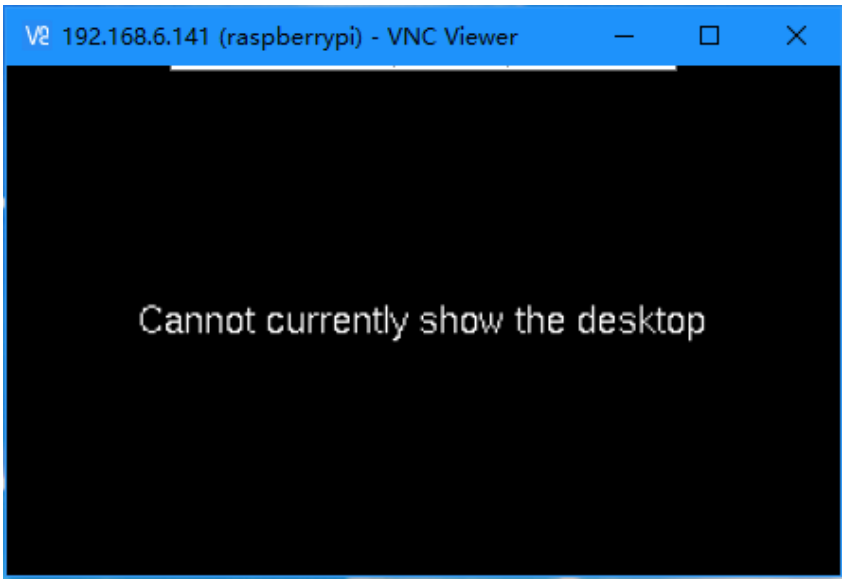


ステップ 3： インストールした VNC ビューアーを開き、Raspberry Pi の IP アドレスを入力して、Enter ボタンを押す。

ステップ 4： このページで Raspberry Pi のユーザー名とパスワードを入力し、デフォルトでは、別々は pi と raspberry である。次に OK をクリックする。



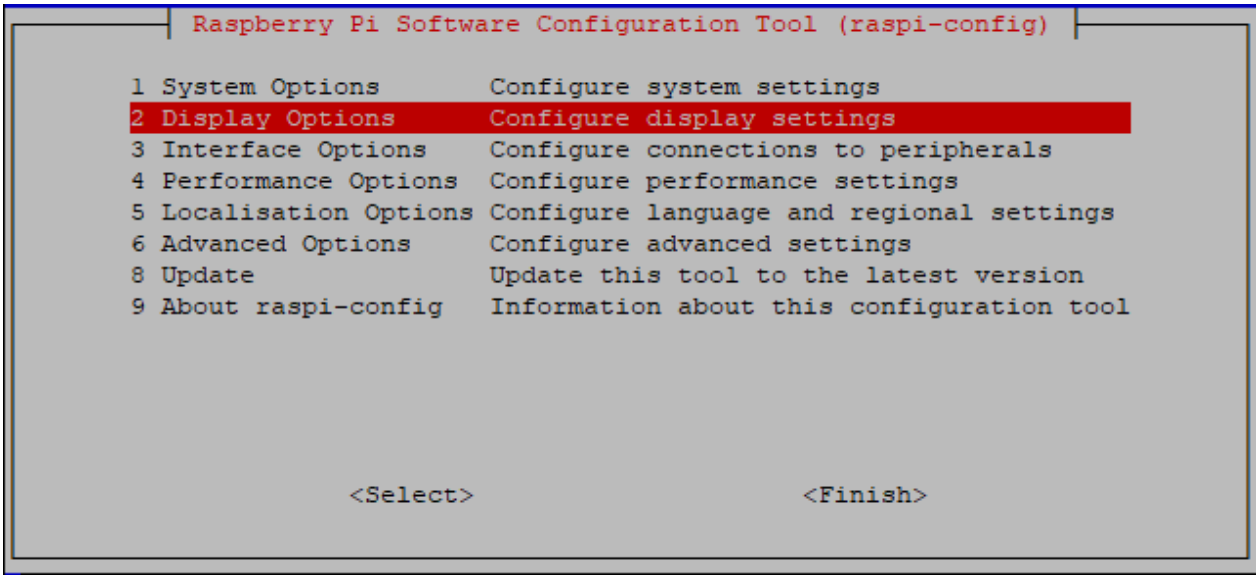
VNC Viewer でログインすると、画面が真っ暗になる場合があります。画面が正常に表示されている場合は、ステップ 5 に進みます。



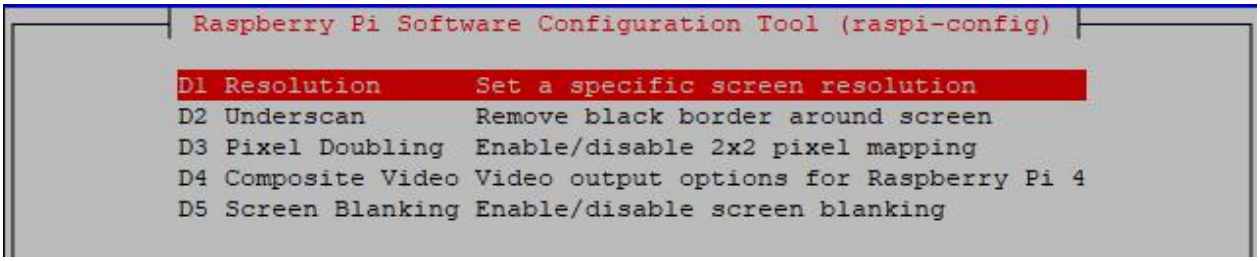
RaspberryPi の解像度を最高に変更します。

```
sudo raspi-config
```

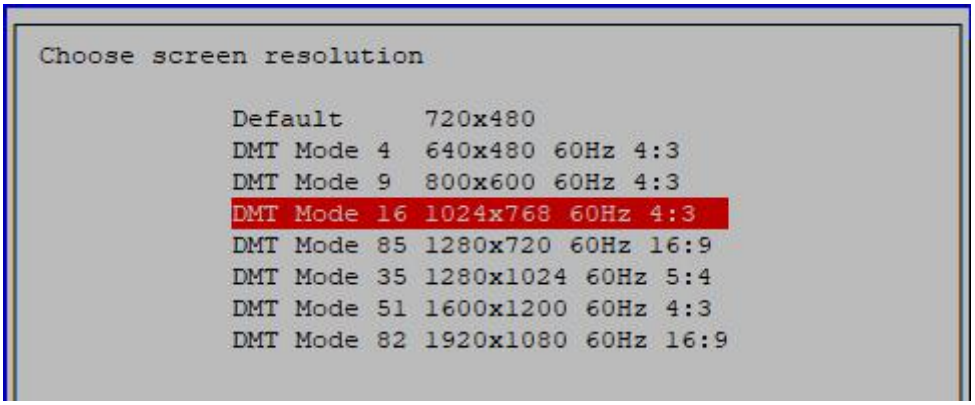
選択 **2 Display Options**。



選択 **D1 Resolution**。



デフォルトをモード 16 のような高解像度に変更します。次に **OK** をクリックする。



Raspberry Pi を再起動し、再度ログインすると、機能します。

```
sudo reboot
```

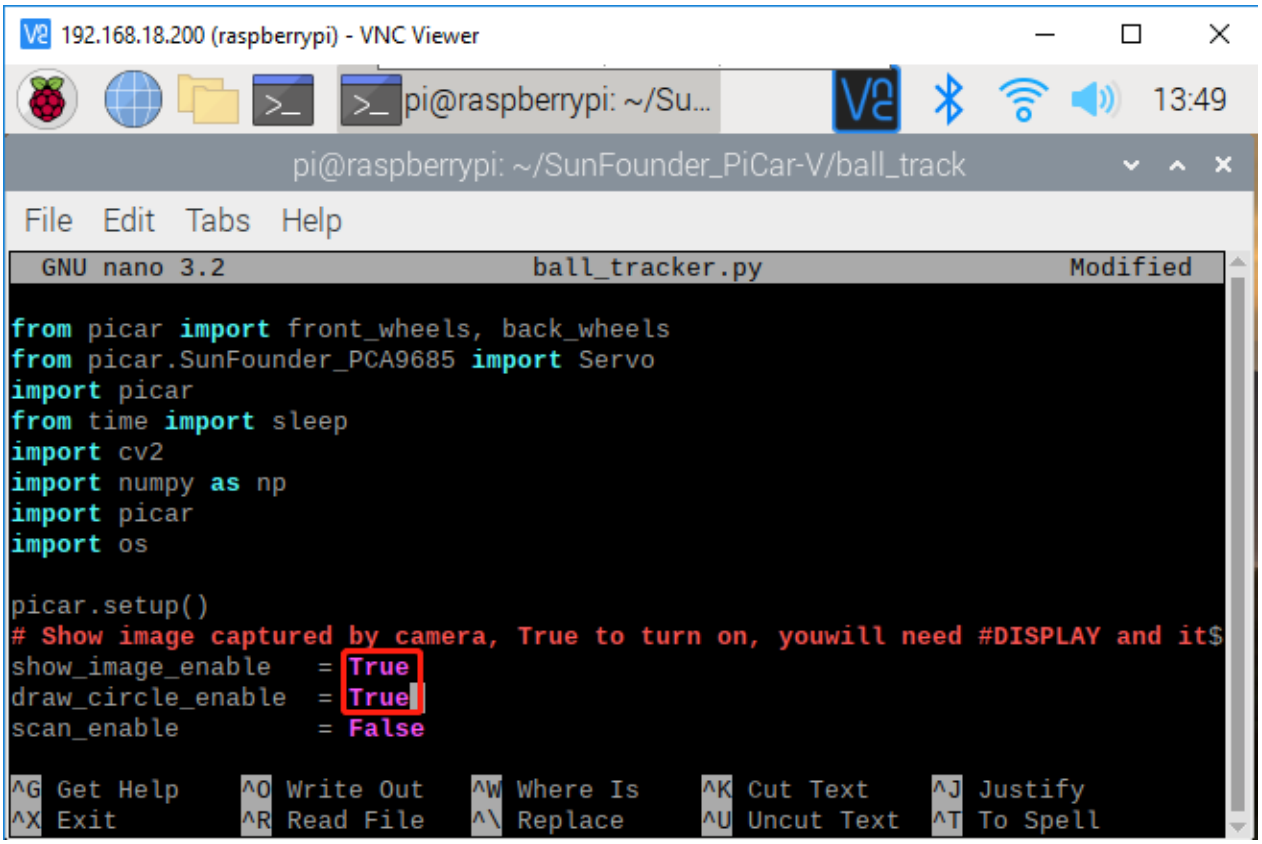
ステップ 5： 次に、Raspberry Pi デスクトップに入り、ターミナルをクリックし、次のコマンドを入力して、コード ball_tracker.py を開く。

```
cd ~/SunFounder_PiCar-V/ball_track
sudo nano ball_tracker.py
```

次に、コードを次のように修正する：

```
Show_image_enable = True
Draw_circle_enble = True
```

Ctrl+X と Y を押して変更を保存する。

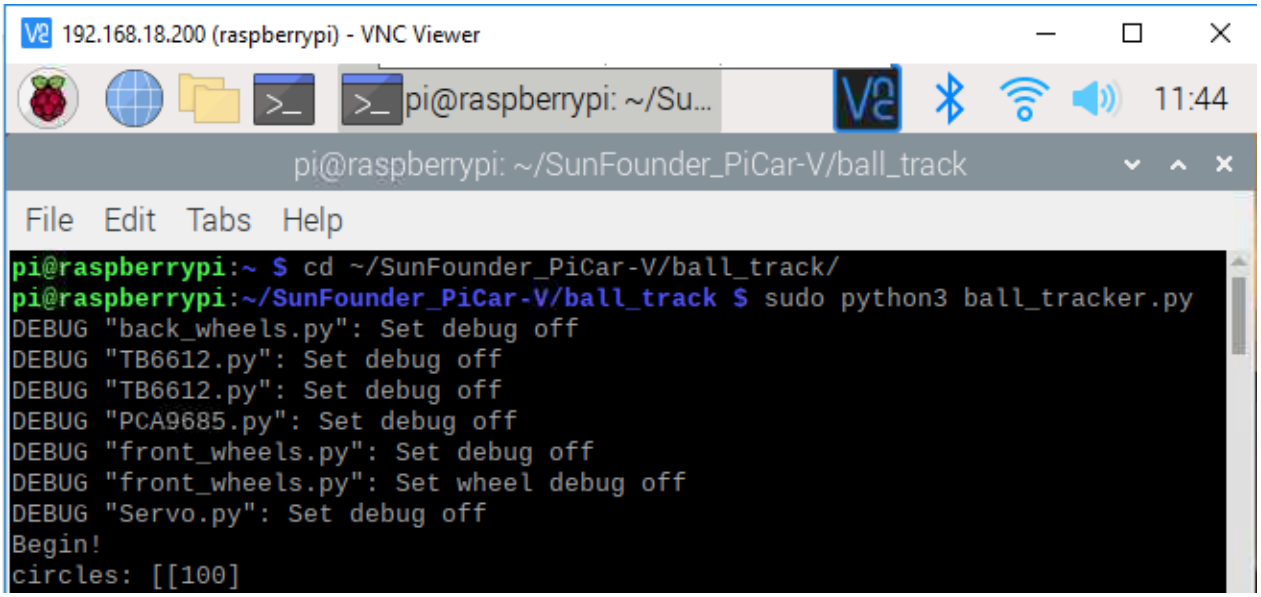


注釈: この手順を完了したら、VNC 経由で Raspberry Pi にログインするか、またはモニターを使用してください。そうでない場合、次の警告が出される：

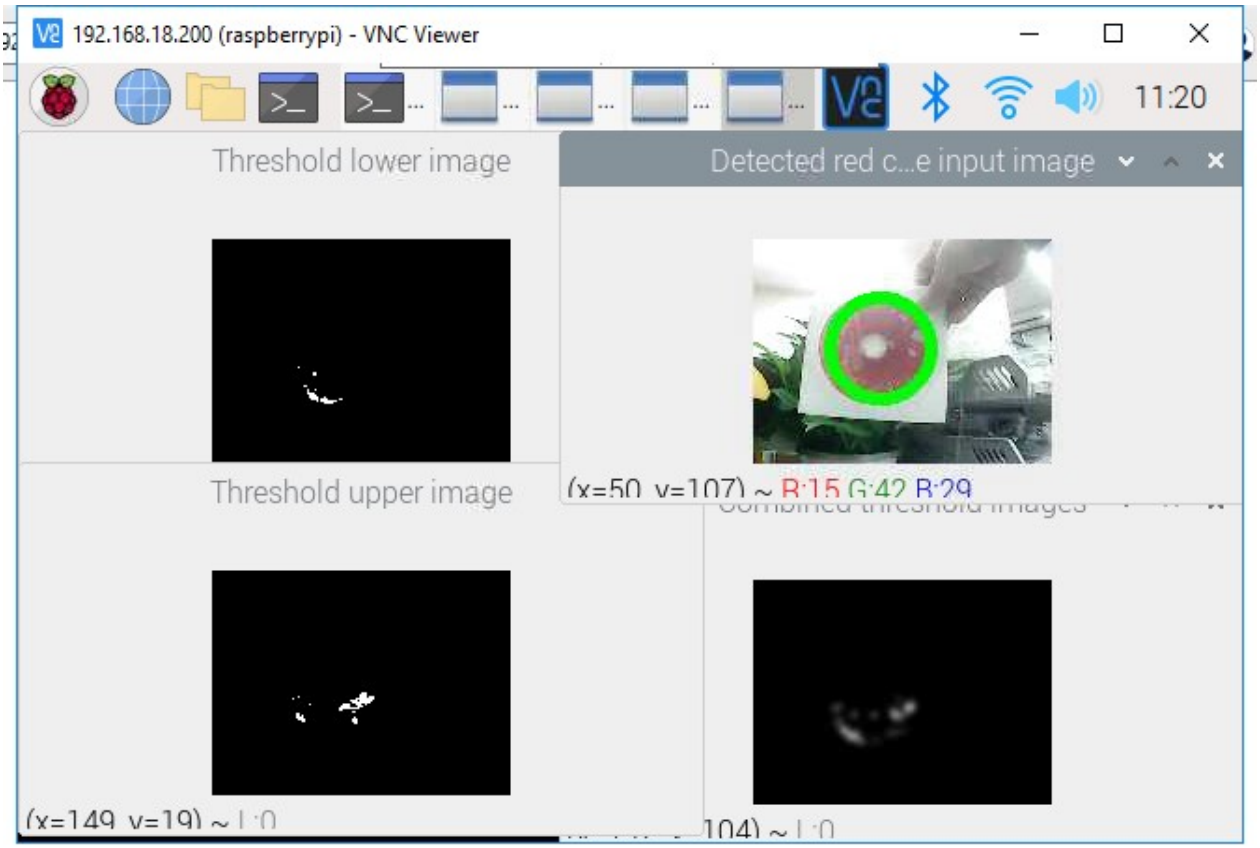
```
pi@raspberrypi:~/SunFounder_PiCar-V/ball_track $ python3 ball_tracker.py
Warning: Display not found, turn off "show_image_enable" and "draw_circle_enable"
```

ステップ 6： ball_tracker.py を実行する。

```
python3 ball_tracker.py
```



ステップ 7： コード ball_tracker.py を実行すると、これらの 4 つの画像が表示される。カメラの前に赤いボールを置くと、車が赤いボールを捕まえたことを示す緑の円が表示される。

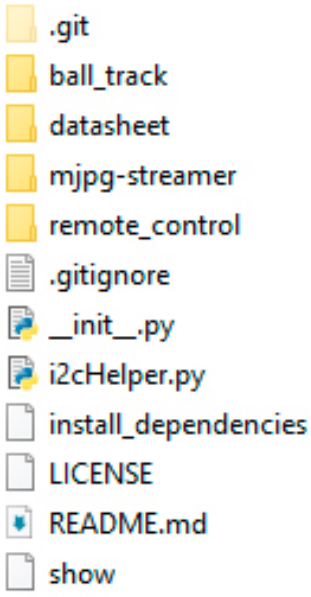


1.8.3 ファイル分析

/home/pi に 2 つのフォルダーがある：モーターとステアリングを制御するための **SunFounder_PiCar** と、パン/チルトとワイヤレス制御のための **SunFounder_PiCar-V** ここで PiCar-V のコードを調べる。

コードに含まれる内容と指示が多すぎるため、すべての詳細を説明しない。プログラムの Python 2、Python 3、Django、HTML、CSS の詳細については、関連する Web サイトにアクセスするか、または本を購入して自分で学習してください。次のパートでは、全体的な構造とプロセスについて簡単に説明する。

まず、コードフォルダー内のファイルを確認する：



- `.git` は Git リポジトリのすべての情報を確認するための隠しディレクトリで、リポジトリを作成すると、自動的に生成される。
- `Ball_track` は車に赤いボールを追跡させるために使用される。
- **データシート** はハードウェアモジュールで使用される一部のチップのマニュアルを保存する。
- `mjpg-streamer` はカメラのデータが転送されるオープンソースのストリーミングメディアライブラリである。
- `Remote_control` は Web サーバーのコードを保存するためのもので、API リクエストに基づいて車のモーターとサーボを制御する。
- `.gitignore` は Github リポジトリが同期されるときに無視されるファイルタイプのリクエストを記録する。
- Python パッケージの標準の必要なドキュメントであるプロジェクトを作成すると、`_init_.py` は自動的に生成されるため、そのままにしておいてください。

- i2cHelper.py は I2C 接続を構成したり検出したりするために Python 2 によって作成された Python スクリプトである。
- install_dependencies，簡単なインストールと環境設定のための実行可能な bash スクリプト。
- その名のとおり、LICENSE は GNU V2 ライセンスのテキストファイルである。
- README.md と show はステートメントとプロンプトに関する通常の情報を記録する。

サーバーコード

サーバーコードは Django 1.10 に基づいている（必要に応じて最新のリリースに適応する）。これに興味がある場合は、Django の Web サイト https://www.djangoproject.com/にアクセスするか。ここでは、Web サーバーがどのように機能するかを学習する。

注釈: バグを修正し、一部の関数の更新をリリースするために、Github でコードが不定期に更新される場合がある。したがって、ここのコードは初期バージョンである。以下の場合、Github リポジトリの更新を表示できる：

https://github.com/sunfounder/SunFounder_PiCar-V/tree/V3.0

コードディレクトリを開き、ls でファイルを確認します。

```
pi@raspberry:~ $ cd SunFounder_PiCar-V/
pi@raspberry:~/SunFounder_PiCar-V $ ls
datasheet   install_dependencies   mjpg-streamer   remote_control
client          i2cHelper.py       LICENSE         README.md show
```

remote_control は、Web サーバーのメインコードディレクトリである。cd remote_control でコードディレクトリを開き、ls によってファイルを確認する：

```
pi@raspberry:~/SunFounder_PiCar-V $ cd remote_control
pi@raspberry:~/SunFounder_PiCar-V/remote_control $ ls
db.sqlite3   manage.py   remote_control   start static
```

- Django プロジェクトを作成すると、db.sqlite3 が生成される。
- manage.py は Django プロジェクトのメインプログラムであり、プロジェクトの作成時に生成される。通常は変更する必要はない。
- remote_control には、メインコードファイルが含まれている。
- start は sudo python manage.py runserver 0.0.0.0:8000 を実行するために書かれた小さなスクリプトであり、便宜上、属性のあるインストールを使用したサーボ **インストール** である。
- static はいくつかの静的な画像を Web に保存するものである。

Django ウェブサーバーは通常、 sudo python manage.py runserver `` **の実行を通じて起動する。アドレス** 0.0.0.0:8000 **は、リスニングアドレスが** LAN **上のすべてのアドレスをカバーし、ポート番号が** 8000 **であることを意味する。ここでは、** ``remote_control フォルダーのコードに焦点を当てる。 cd remote_control を使用してディレクトリに入る：

```
pi@raspberry:~/SunFounder_PiCar-V/remote_control $ cd remote_control
pi@raspberry:~/SunFounder_PiCar-V/remote_control/remote_control $ ls
driver   \__init__.py   settings.py   templates urls.py   views.py   wsgi.py
```

- driver は車のドライバーを保存する。
- __init__.py は Python パッケージの標準的な必要なドキュメントである Django プロジェクトを作成すると自動的に生成される。そのままにしておいてください。
- settings.py は自動的に生成され、関連する設定を保存する。
- templates HTML 形式でウェブを格納するためのジャンゴアプリです。
- urls.py は自動的に生成され、コードに関連付ける URL を構成する。
- views.py は URL によって関連付けられるページコントロールのコードである。テンプレートを呼び出してページを表示し、ドライバーが車を制御する。
- wsgi.py は自動的に生成され、変更する必要はない。詳しくは Django の公式サイトをご覧ください。

これがコードの動作原理である。urls.py に自動的に関連付けられるメインプログラム manage.py を実行して、URL に応答する。Chrome のような Web ブラウザを実行して http://<rpi_ip_address>:<port> `` **または、クライアントを介して構成済みの** API **にアクセスすると、**``urls.py の関連付けにより、manage.py が views.py に変わる。次に、views.py はこれを処理し、テンプレートをブラウザに返す。さらに、ブラウザーで設定されたパラメーターに基づいて driver パッケージを呼び出し、車を制御する。

ここで、フォルダー driver を開き、以下を確認する：

```
pi@raspberry:~/SunFounder_PiCar-V/remote_control/remote_control $ cd driver/
pi@raspberry:~/SunFounder_PiCar-V/remote_control/remote_control/driver $ ls
camera.py   config   \__init__.py   stream.py
```

driver フォルダーには、主にパン＆チルトを制御するためのドライバーモジュールとカメラストリーマーが含まれている。

パン＆チルトを制御するための camera.py。

config は補正データを保存するために使用される。

__init__.py はパッケージの必須ファイルであり、そのままにしておいて下さい。

stream.py は MJPG-streamer に基づくビデオストリーミングサービスである。

表示するフォルダ templates を終了して開くと、以下が表示される：

```
pi@raspberry:~/SunFounder_PiCar-V/remote_control/remote_control/driver $ cd ../
pi@raspberry:~/SunFounder_PiCar-V/remote_control/remote_control $ cd templates/
admin.py    \__init__.py    models.py    tests.py
apps.py     migrations      templates    views.py
```

このフォルダーは、テンプレートを簡単に呼び出すために、`` manage.py startapp`` によって作成される。したがって、`` templates`` 以外のファイルは変更されていない。`templates` をもう一度開く：

```
pi@raspberry:~/SunFounder_PiCar-V/remote_control/remote_control/templates $ cd␣
↪templates
pi@raspberry:~/SunFounder_PiCar-V/remote_control/remote_control/templates/templates $␣
↪ls
base.html    cali.html    run.html
```

レイヤーで構成される 3 つの HTML ファイルがある。低レベルの `base.html` には、デフォルトでは、HTML の `` <head> **、レイアウト全体、とホームページのコンテンツが含まれている。表層：補正用の** ``cali.html` と、車を制御するための `run.html`。

1.8.4 高級段階

今、車は走行している！それを制御するか、または Windows、Linux、または Mac OS X でキャプチャされた画像を表示してみてください。

それでは次に何をすべきか？

はい、センサー！ Robot HATS には 8 つのデジタルチャネルと 4 つのアナログチャネルと 2 つの I2C ポートがある。さらに、車のヘッドのアクリル板にある 3mm のスロットにより、さまざまなセンサーの接続が可能になり、車がよりスマートになり、興味をそそられるようになる！

そして、コーディング！プログラミングに関する知識を身につけて、コードをコンパイルできる場合は、キット用に提供されているコードを変更して、車をよりすばらしいものにすることができる。たとえば、前輪と後輪を制御するコードは、SunFounder_PiCar フォルダで見つけで install-dependences によってプログラムをインストールした場合は、/home/pi ディレクトリにダウンロードされる。

ファイルの内容を確認してください：

```
cd ~/SunFounder_PiCar
ls
```

コードを変更したら、車の Github リポジトリのページに入り、フォークして改善に役立つだろう。また、問題を投稿して、Github ページでプルをリクエストすることもできる：

PiCar-V: https://github.com/sunfounder/SunFounder_PiCar-V/tree/V3.0

PiCar ドライバー：https://github.com/sunfounder/SunFounder_PiCar

もちろん、質問やアイデアがある場合は、service@sunfounder.com までお気軽に送信してください。

1.9 付録

1.9.1 付録 1：サーバーインストールスクリプトの機能

質問があるかもしれない：Raspberry Pi にサーバーをインストールすると、インストールスクリプトはどのように機能するか？ここで、詳細な手順を説明しよう。

1. pip をインストールする。

```
sudo apt-get install python-pip
```

2. pip を使って django をインストールする。

```
sudo pip install django
```

3. i2c-tools と python-smbus をインストールする。

```
sudo pip3 install smbus2
```

4. PiCar ドライバーをインストールする。

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

5. ソースコードをダウンロードしてください。

```
cd ~/
git clone https://github.com/sunfounder/SunFounder_PiCar-V -b V3.0
```

6. MJPG-Streamer ファイルをシステムディレクトリにコピーする。

```
cd ~/SunFounder_PiCar-V
sudo cp mjpg-streamer/mjpg_streamer /usr/local/bin
sudo cp mjpg-streamer/output_http.so /usr/local/lib/
sudo cp mjpg-streamer/input_file.so /usr/local/lib/
sudo cp mjpg-streamer/input_uvc.so /usr/local/lib/
sudo cp -R mjpg-streamer/www /usr/local/www
```

7. パスをエクスポートする。

```
export LD_LIBRARY_PATH=/usr/local/lib/ >> ~/.bashrc
export LD_LIBRARY_PATH=/usr/local/lib/ >> ~/.profile
source ~/.bashrc
```

8. I2C1 を有効にする。

/boot/config.txt ファイルを編集する：

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

最後に行を追加する：

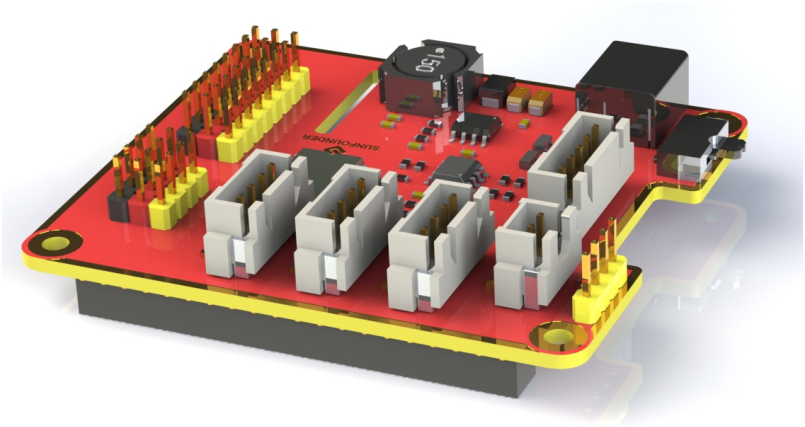
```
dtparam=i2c_arm=ons
```

9. 再起動する。

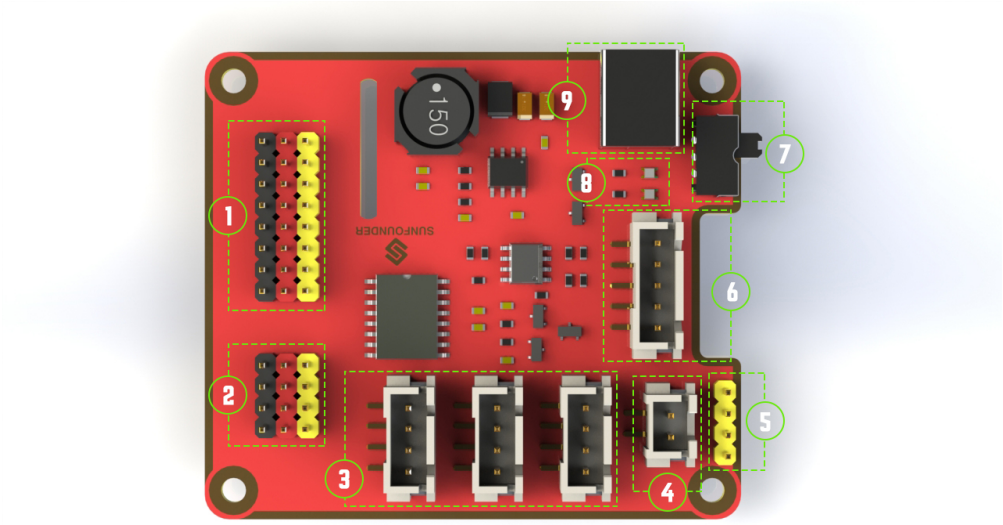
```
sudo reboot
```

1.9.2 付録 2：部品

ロボット HATS

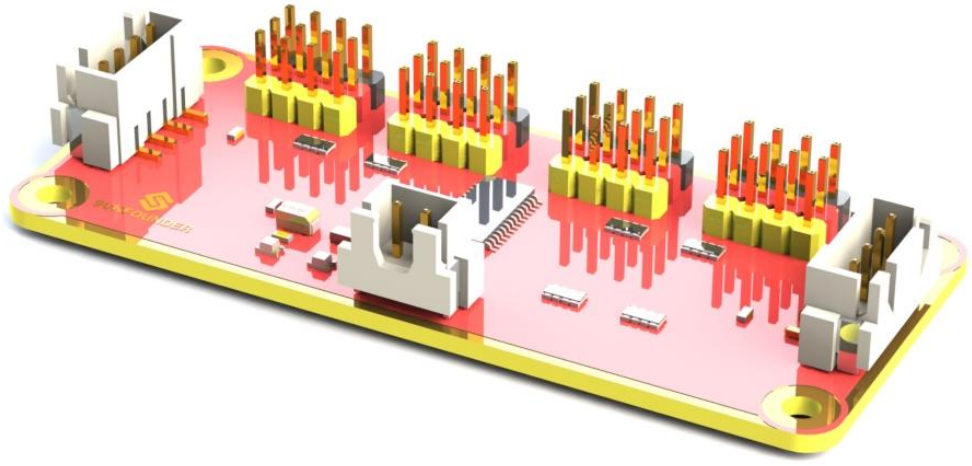


ロボット HATS は 40 ピン Raspberry Pi に特別に設計された HAT であり、Raspberry Pi モデル B +、世代 2 モデル B、世代 3 モデル B、世代 3 モデル B +、世代 4 モデル B で動作し、GPIO ポートから Raspberry Pi に電力を供給する。HATS のルールに基づいた理想的なダイオードの設計により、USB ケーブルと DC ポートの両方を介して Raspberry Pi に電源を供給できるため、バッテリーの電力不足によって TF カードが損傷することを防ぐことができる。PCF8591 は I2C 通信とアドレス 0x48 を備えた ADC チップとして使用される。

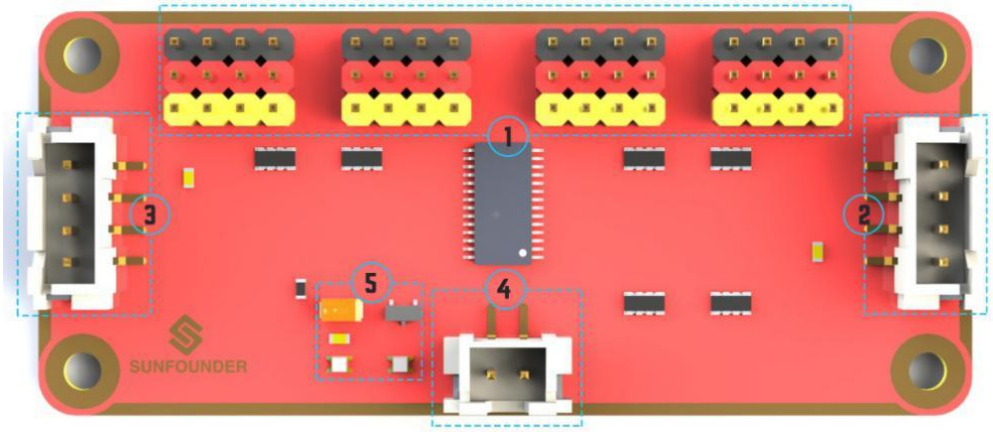


1. **デジタルポート**：3 線式デジタルセンサーポート、信号電圧：3.3V、VCC 電圧：3.3V。
2. **アナログポート**：3 線 4 チャネル 8 ビット ADC センサーポート、基準電圧：3.3V、VCC 電圧：3.3V。
3. **I2C ポート**：3.3V I2C バスポート
4. **5V 電源出力**：PWM ドライバーへの 5V 電源出力。
5. **UART ポート**：4 線 UART ポートと 5V VCC は USB への SunFounder FTDI シリアルと完全に連携する。
6. **モーター制御ポート**：モーター用 5V、モーター MA と MB の方向制御、フローティングピン NC、SunFounder モータードライバーモジュールとの連携。
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 つのインジケーター：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、インジケーター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



PCA9685 16 チャネル 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。

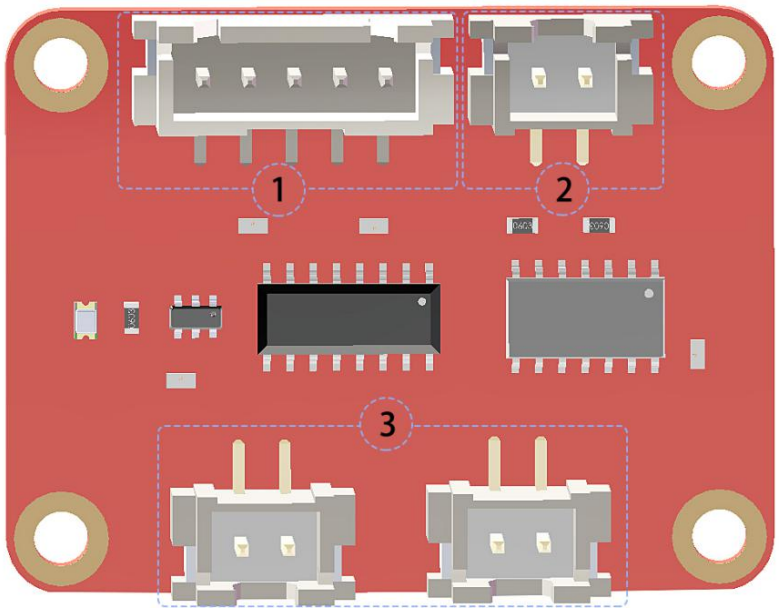
2 & 3. **I2C ポート**：4 線式 I2C ポートは並列で使用できる。3.3V/5.5V に対応。

4. **PWM 電源入力**：最大 12V。

5. **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。

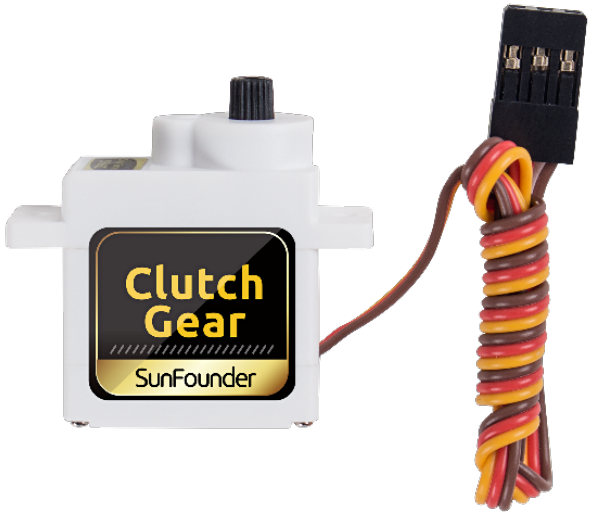


1. **電源とモーター制御ポート**：チップとモーターに電力を供給し、モーターの方向を制御するためのピンが含まれている。
2. **モーターの PWM 入力**：2 つのモーターの速度を調整するための PWM 信号入力。
3. **モーター出力ポート**：2 つのモーターの出力ポート。

USB ウェブカメラ

このカメラは 120 ° の広角をサポートするため、広く鮮明な視界が得られて、PiCar-V で使用するときのエクスペリエンスが向上する。

SunFounder SF006C サーボ

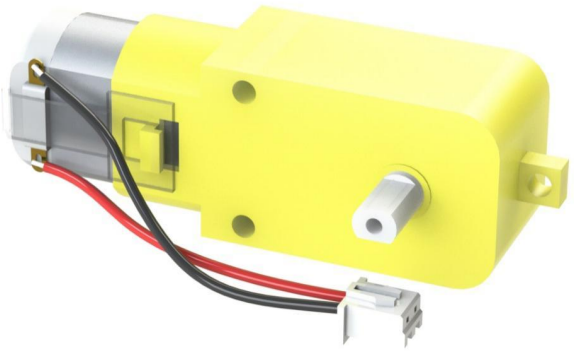


クラッチギアデジタルサーボの内部に DC コアモーターが搭載されており、一定の負荷をかけた後、ステアリングギアの減速装置は自動的にクラッチを切り、損傷や通常の負荷から製品を保護する。

パフォーマンスの機能：

アイテム	V = 4.8V	V = 6.0V
消費電流*（無負荷）	≦ 50mA	≦ 60mA
ストール電流	≦ 550mA	≦ 650mA
定格トルク	âL'ě0.6 kgfÂůcm	âL'ě0.7 kgfÂůcm
最大トルク	âL'ě1.4 kgf.cm	âL'ě1.6 kgf.cm
負荷速度なし	≦ 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	≤80mA
	No-load Speed	10000±10%
Gear Reducer	Gear Ratio	1:48
	Speed (no-load)	≈200rpm (≈180rmp in test)
	Current	≤120mA

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