SunFounder PiCar-X Kit

www.sunfounder.com

2024 年 03 月 05 日

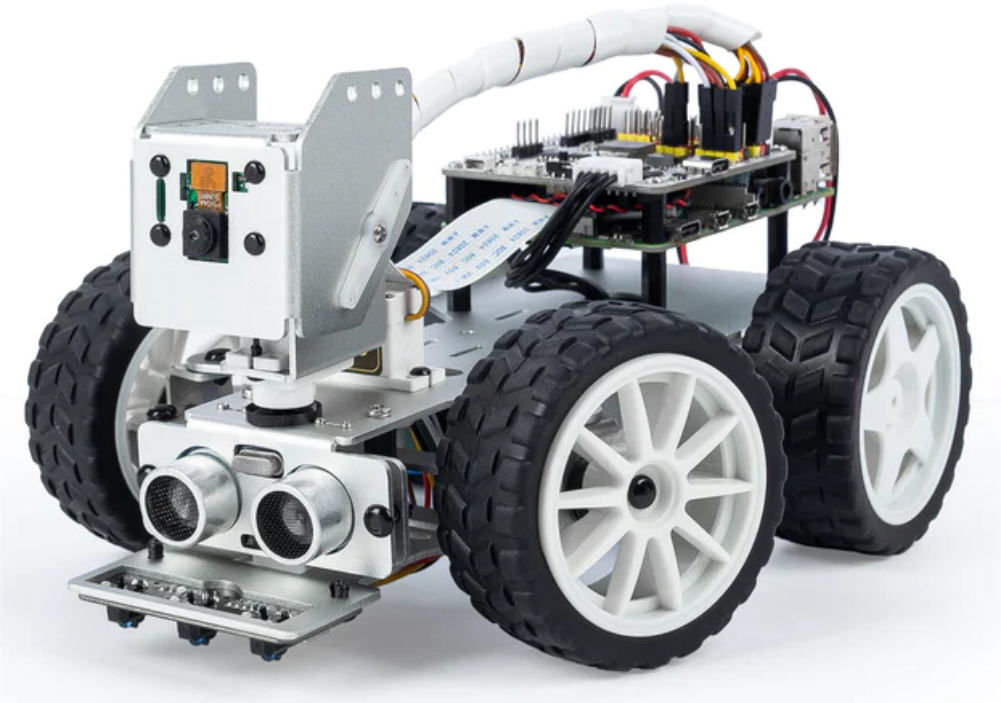
目次

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

注釈: このドキュメントは以下の言語で利用可能です。

-
-
-

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



PiCar-X は、Raspberry Pi を制御センターとして使用する AI 駆動の自動運転ロボットカーです。PiCar-X には 2 軸のカメラモジュール、超音波モジュール、ライン追跡モジュールが備わっており、色・顔・交通標識の検出、自動障害物回避、自動ライン追跡などの機能を提供できます。

PiCar-X は、Blockly と Python の 2 つのプログラム言語をサポートしています。どちらの言語でプログラムしても、Raspberry Pi の設定から関連するサンプルコードの実行までの詳細な手順が記載されています。

- ***Python* を楽しもう**
  - Python でのプログラミングを楽しむ方や、Python 言語を学びたい方のための章です。
  - PiCar-X を適切に動作させるためには、いくつかのライブラリを先にインストールする必要があります。
  - この章では、PiCar-X のための Raspberry Pi 設定とサンプルコードを提供しています。

  - APP - SunFounder Controller も提供されており、モバイルデバイスで PiCar-X をリモート制御することができます。

- *Ezblock* **を楽しむ**

  - このセクションでは、Scratch のような Blockly ベースの APP、Ezblock Studio を使用します。これを使用して、ブロックをドラッグ&ドロップして Picar-X を動かすことができます。
  - プログラミングする前に、提供されたオペレーティングシステムをプリインストールした Ezblock 環境で SD カードを再インストールする必要があります。このセクションには、新しいもしくは未使用の TF カードの使用を推奨します。
  - Ezblock Studio は、Mac、PC、Android を含むほぼすべてのデバイスで利用可能です。
  - 6 歳から 12 歳、プログラムのスキルがない、または Picar-X を迅速にテストしたい方には、Ezblock Studio がおすすめです。

目次

第 1 章

導入

1.1 自動運転車の歴史

自動運転車に関する実験は、少なくとも 1920 年代から行われています。1950 年代には前進が見られ、それ以降も研究が進行してきました。1980 年代に最初の自足型で真に自律した車が登場し、1984 年のカーネギーメロン大学の Navlab や ALV プロジェクト、1987 年のメルセデスベンツとドイツ連邦軍大学ミュンヘンの Eureka Prometheus プロジェクトなどがあります。1980 年代後半以降、多くの研究機関や大手自動車メーカーが、メルセデスベンツ、ゼネラルモーターズ、コンチネンタル・オートモーティブ・システムズ、Autoliv Inc.、ボッシュ、日産、トヨタ、アウディ、ボルボ、パルマ大学の Vislab、オックスフォード大学、グーグルなど、作動する自動運転車を開発してきました。2013 年 7 月、Vislab は BRAiVE をデモンストレーションし、公道を使用した混在交通のルートで自動的に動作しました。2019 年までに、アメリカの 29 州で自動車の公道運転が許可されています。

一部の UNECE 加盟国や EU 加盟国、イギリスを含む国々は、自動化された完全自動車に関連するルールや規制を制定しています。ヨーロッパでは、ベルギー、フランス、イタリア、イギリスの都市が、運転手のいない車のための輸送システムを運用する計画を進めており、ドイツ、オランダ、スペインでは公道でのロボット車のテストが既に許可されています。2020 年、イギリス、EU、日本では自動車の規制が既に進行中です。

- 参照： 自動運転車の歴史 - Wikipedia

今日、自動運転車は手元の技術革命として最も近いものです。一部の専門家は、2025 年までにレベル 4 の車が市場に登場する可能性が高いと予測しています。レベル 4 の車は、システムが正常に動作している限り、運転手が完全に注意を向ける必要がなくなります。

レベル 4 の参照：

- SAE 運転自動化のレベル ™
- ABI リサーチ：2025 年までに SAE レベル 3、4、5 の自動技術を搭載した 800 万台の車両が出荷されると予測

最近のソフトウェア（人工知能、機械学習）、ハードウェア（GPU、FPGA、加速度計など）、クラウドコンピューティングの急速な進展が、この技術革命を前進させています。

- 2010 年 10 月、イタリアの技術企業 **Vislab** が設計した無人トラックは、 イタリアから中国までの旅 に 3 ヶ月かかり、合計 8,077 マイルの距離を移動しました。
- 2015 年 4 月、 **Delphi Automotive** が設計した車は、 サンフランシスコからニューヨークまで 、3,400 マイルを移動し、その距離の 99 %をコンピュータ制御で完了しました。
- 2018 年 12 月、 **Alphabet** の **Waymo** は、 アリゾナでレベル 4 の自動運転タクシーサービスを開始 し、2008 年以降、無人運転車のテストを行っていました。ドライバーシートに誰も座っていない状態で、車両は 1 年以上運用され、1 千万マイル以上の距離を移動しました。
- 2020 年 10 月、 **Baidu** は、 北京での Apollo Robotaxi 自動運転タクシーサービスを完全に開始 しました。運転ルートは、地元の住宅地、商業地、レジャー施設、工業団地を網羅しており、完全な自動運転システムを提供しています。

しかし、毎日収集される大量のデータ、実際の運転記録やシミュレーションシナリオからのトレーニングデータを含めても、自動運転車の AI モデルの複雑さはまだ完全には満たされていません。

RAND の報告書 によれば、適切な自動学習レベルに到達するためには、信頼性を確立するために数百万マイル、あるいは数千億マイルのトレーニングデータが必要です。

ですので、自動運転車の未来は有望で興奮するものですが、技術が十分に成熟して、自動運転車市場に完全にアクセス可能になるまでには、まだ多くの開発年月が必要です。

新技術を迅速に成熟させるための確かな方法は、市場への参入要件を最小限にすることで、それを皆に簡単に利用可能にすることです。これが SunFounder が PiCar-X を立ち上げる動機です。

SunFounder の目標は、初心者や素人、または単に自動運転について学びたい人たちが、開発プロセスや技術、自動運転車の最新の革新について理解するのを助けることです。

1.2 PiCar-X について

PiCar-X は、Raspberry Pi プラットフォーム用の AI 制御自動運転ロボットカーであり、Raspberry Pi が制御センターとして機能します。PiCar-X の 2 軸カメラモジュール、超音波モジュール、ライン追跡モジュールは、色/顔/交通標識の検出、自動障害物回避、自動ライン追跡などの機能を提供できます。

SunFounder 設計の Robot HAT ボードを使用して、PiCar-X は左/右の駆動モーター、ステアリング用のサーボモーター、カメラのパン/チルト機能のサーボモーターを統合し、Robot HAT の ADC、PWM、デジタル I2C ピンをプリセットして、Raspberry Pi の標準機能への拡張を可能にします。スピーカーと Bluetooth チップは、Text-to-Speech、効果音、あるいはバックグラウンドミュージック機能のリモート制御のために、Robot HAT に組み込まれています。

PiCar-X のすべての機能、GPIO 制御、コンピュータビジョン、深い学習は、オープンソースの Python プログラム言語、OpenCV のコンピュータビジョンライブラリソフトウェア、Google の TensorFlow を使用して実装されています。他のソフトウェアも PiCar-X の機能を最適化するために含まれており、ユーザーにほぼ無限の学習環境を提供します。

1.3 深い学習とニューラルネットワーク

深い学習とニューラルネットワークについての詳細は、SunFounder が以下のリソースを推奨しています：

Machine Learning - Andrew Ng：このコースは、機械学習、データマイニング、統計的パターン認識に関する幅広い紹介を提供します。

Neural Networks and Deep Learning：この E-book は、ニューラルネットワーク、観測データからコンピュータに学習をさせる生物学的にインスパイアされたプログラミングパラダイム、およびニューラルネットワークの機械学習のための強力な一連の技術である深い学習をカバーしています。

Rethinking the Inception Architecture for Computer Vision：この高レベルのホワイトペーパーは、ユーザーが因子化された畳み込みと積極的な正則化を通じて、追加の計算を可能な限り効率的に利用する方法を探求しています。

第 2 章

組み立て手順

PiCar-X を組み立てる前に、すべての部品とコンポーネントが含まれていることを確認してください。もし不足や破損しているコンポーネントがあれば、可能な限り早く service@sunfounder.com まで SunFounder に直ちに連絡してください。

組み立て手順については、以下の PDF を参照して手順に従ってください：

[PDF]PiCar-X **の部品リストと組み立て**。

Raspberry Pi Zero W を PiCar-X に取り付ける

もしあなたのメインボードが Raspberry Pi Zero W である場合、以下は PiCar-X に取り付ける手順です。

その後、 **1:45** からのビデオ内の指示に従って組み立てを続けることができます。

組み立てチュートリアルビデオ（Raspberry Pi 4/3/1 モデル）

このビデオは、ロボットの組み立て手順を詳しく説明しています。

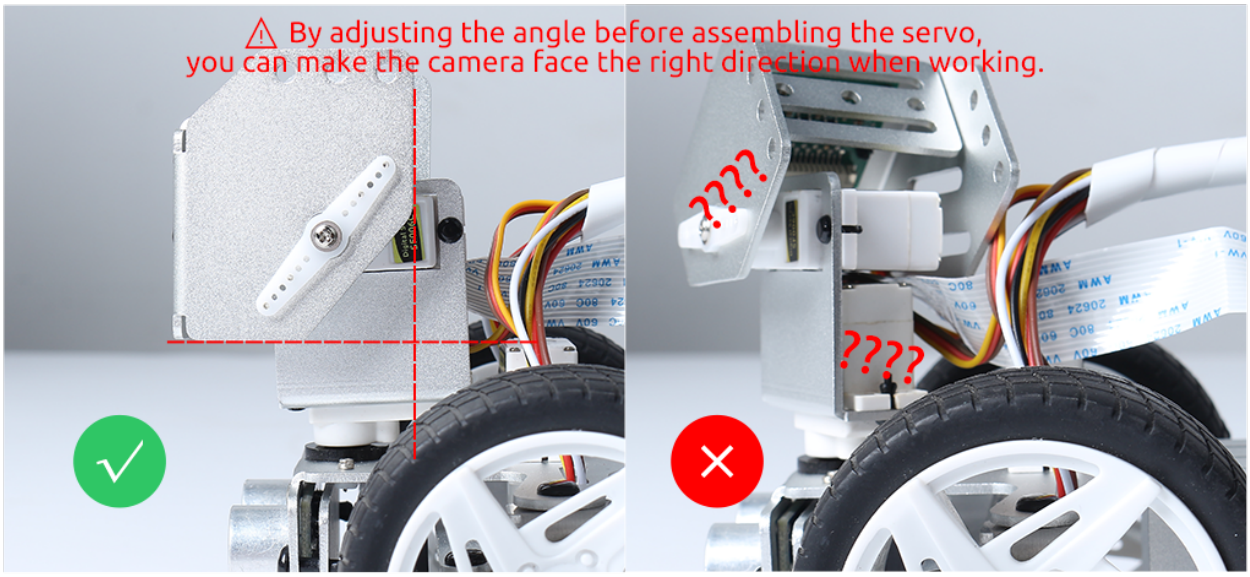
このチュートリアルで学べること：

- **準備**：必要な工具と部品を紹介し、組み立てを開始する前に万全の態勢を整えます。
- **組み立てステップ**：組み立ての各ステップを体系的にデモします。
- **ヒントと注意点**：プロセス全体を通じて、よくあるミスを避け、車がスムーズに動作するための重要なヒントとテクニックを共有します。
- **サーボのゼロリング**：各サーボを固定する前に、まずゼロリングが必要です。ゼロリングの手順は、最初に Raspberry Pi OS をインストールし、次に必要なモジュールをインストールし、その後スクリプトを実行して（すべての PWM ピンの角度を 0 に設定）。その後、サーボワイヤを接続してサーボをゼロリングします。

第 3 章

サーボの組み立てのための調整

サーボを組み立てる前に、角度はゼロに設定する必要があります。これはサーボモーターの動作範囲が限られているためで、角度をゼロ度に設定することで、サーボが電源投入時にその動作範囲を超えないように初期位置にあることを保証します。組み立て前にサーボがゼロ度に設定されていない場合、電源投入時にその動作範囲を超える可能性があり、サーボや接続されている機械システムにダメージを与える恐れがあります。したがって、角度をゼロに設定することは、サーボモーターの安全で正常な動作を保証するための重要な手順です。



Python のユーザーへ

Raspberry Pi OS のインストールとサーボの角度調整の完了には、*Python* **のクイックガイド** を参照してください。

Ezblock のユーザーへ

ezblock システムをインストールした後、P11 ピンを使用してサーボの角度を調整することができます。詳細は *EzBlock* **のクイックガイド** を参照してください。

第 4 章

Python を楽しもう

Python でプログラミングを学びたい初心者やビギナーの方には、基本的な Python プログラミングスキルと Raspberry Pi OS の知識が必要です。Raspberry Pi の設定を始めるには、Python に関するクイックガイドを参照してください：

4.1 Python のクイックガイド

このセクションでは、Raspberry Pi OS のインストール、Raspberry Pi への wifi 設定、Raspberry Pi へのリモートアクセス、対応するコードの実行方法について説明します。

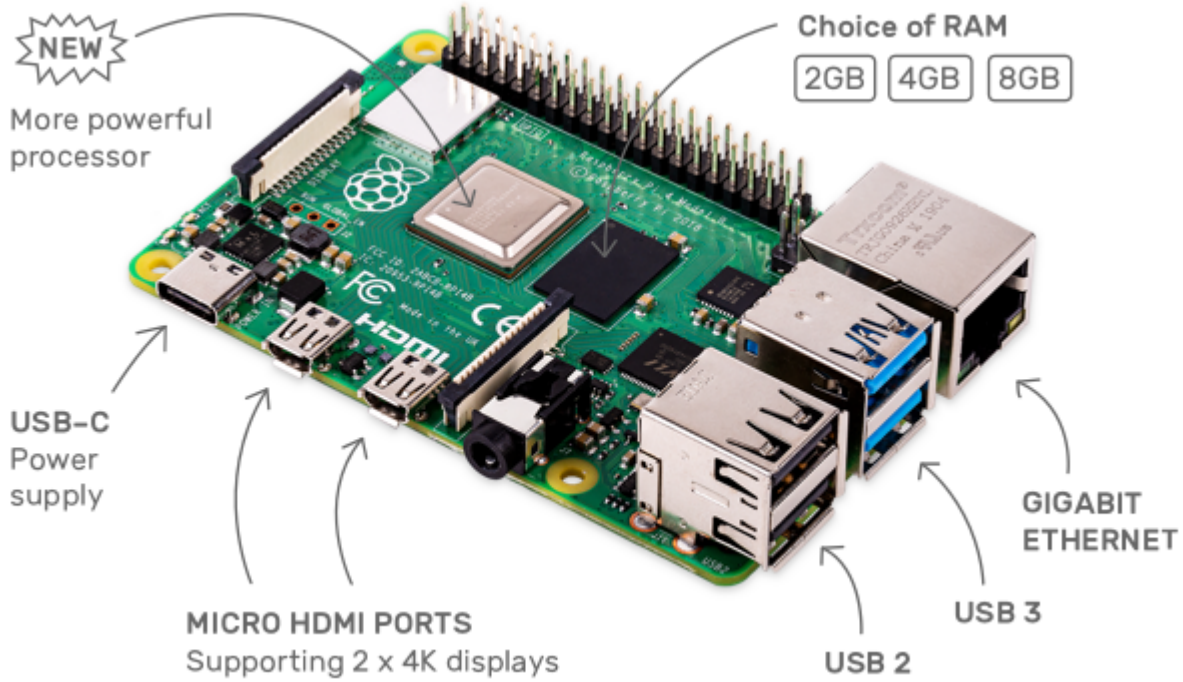
Raspberry Pi に精通しており、コマンドラインを正常に開くことができる場合、最初の 3 部分をスキップして、最後の部分を完了できます。

4.1.1 必要なものは何ですか？

必要なコンポーネント

Raspberry Pi

Raspberry Pi は、低コストでクレジットカードサイズのコンピュータで、コンピュータモニターやテレビに接続し、標準のキーボードとマウスを使用します。これは小さながらも能力のあるデバイスで、あらゆる年齢の人々がコンピューティングを探求し、Scratch や Python のような言語でプログラムを学ぶことを可能にします。



電源アダプタ

電源ソケットに接続するために、Raspberry Pi にはマイクロ USB ポート（多くの携帯電話に見られるものと同じ）があります。少なくとも 2.5 アンペアを提供する電源が必要です。

マイクロ SD カード

Raspberry Pi は、すべてのファイルと Raspberry Pi OS を保存するためにマイクロ SD カードを必要とします。少なくとも 8 GB の容量のマイクロ SD カードが必要です。

オプションのコンポーネント

スクリーン

Raspberry Pi のデスクトップ環境を表示するには、テレビ画面またはコンピュータモニターを使用する必要があります。スクリーンに内蔵スピーカーがある場合、Pi はそれらを介して音を再生します。

マウス＆キーボード

スクリーンを使用する場合、USB キーボードと USB マウスも必要です。

HDMI

Raspberry Pi には、ほとんどの現代のテレビやコンピュータモニターの HDMI ポートと互換性のある HDMI 出力ポートがあります。スクリーンが DVI または VGA ポートのみを持っている場合、適切な変換ラインを使用する必要があります。

ケース

Raspberry Pi をケースに入れることができます。これにより、デバイスを保護できます。

サウンドまたはイヤホン

Raspberry Pi には約 3.5 mm のオーディオポートが装備されており、スクリーンに内蔵スピーカーがない場合や、スクリーン操作がない場合に使用できます。

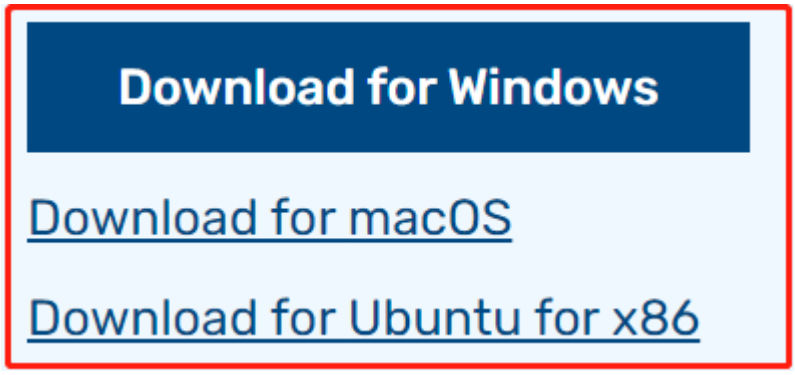
4.1.2 OS のインストール

必要なコンポーネント

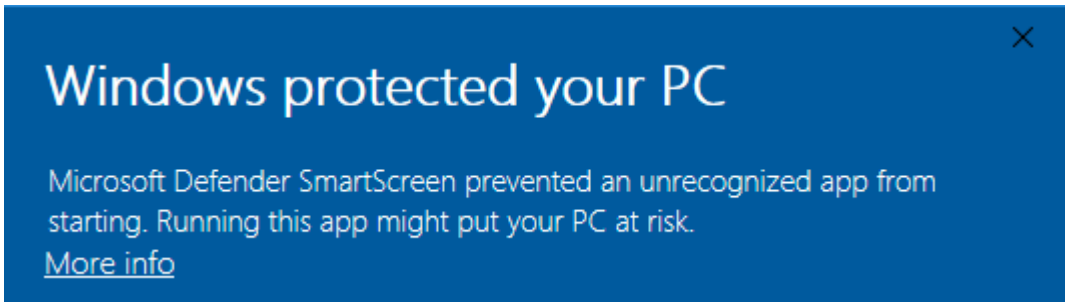
- Raspberry Pi 4B/Zero 2 w/3B 3B+/2B/Zero W
- 1 x パーソナルコンピュータ
- 1 x マイクロ SD カード

手順

1. Raspberry Pi のソフトウェアダウンロードページにアクセスします： Raspberry Pi Imager。ご使用のオペレーティングシステムに合わせた Imager バージョンを選択します。ダウンロード後、ファイルを開いてインストールを開始します。

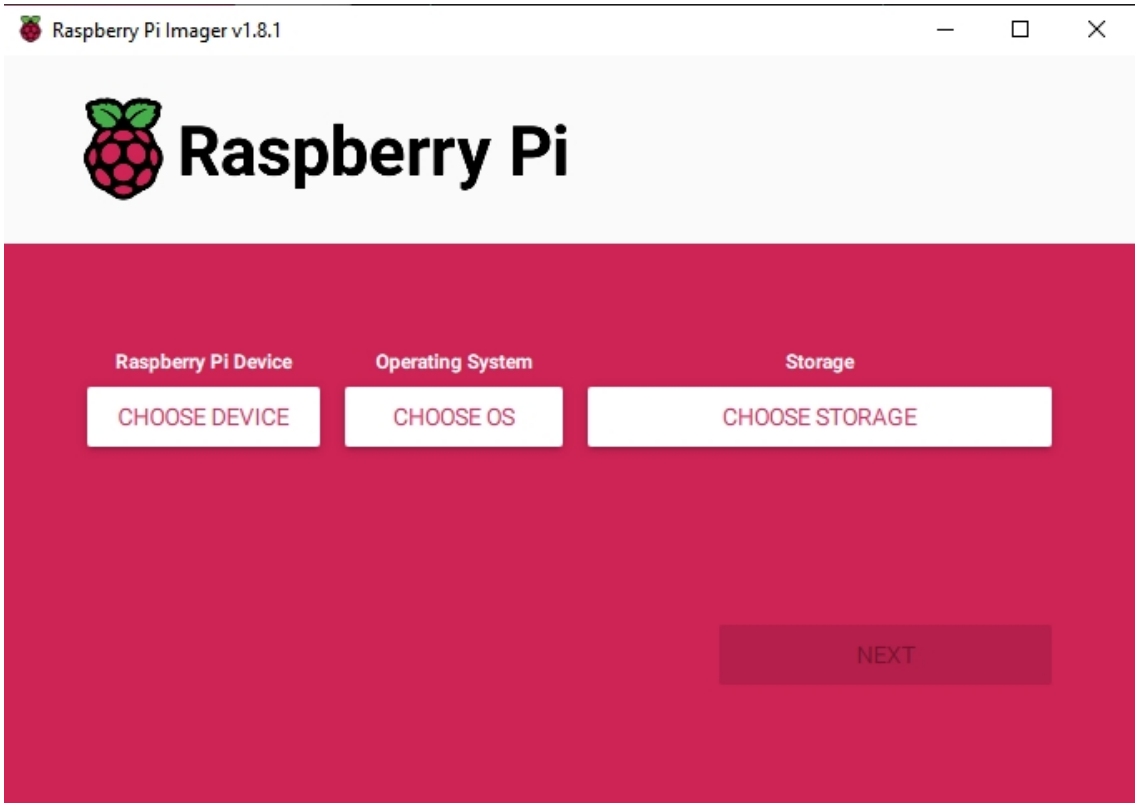


2. インストーラを起動すると、OS がセキュリティ警告を表示する場合があります。たとえば、Windows では警告メッセージが表示されることがあります。このような場合は、 **More info** を選択し、その後 **Run anyway** を選択します。画面の指示に従って Raspberry Pi Imager をインストールします。

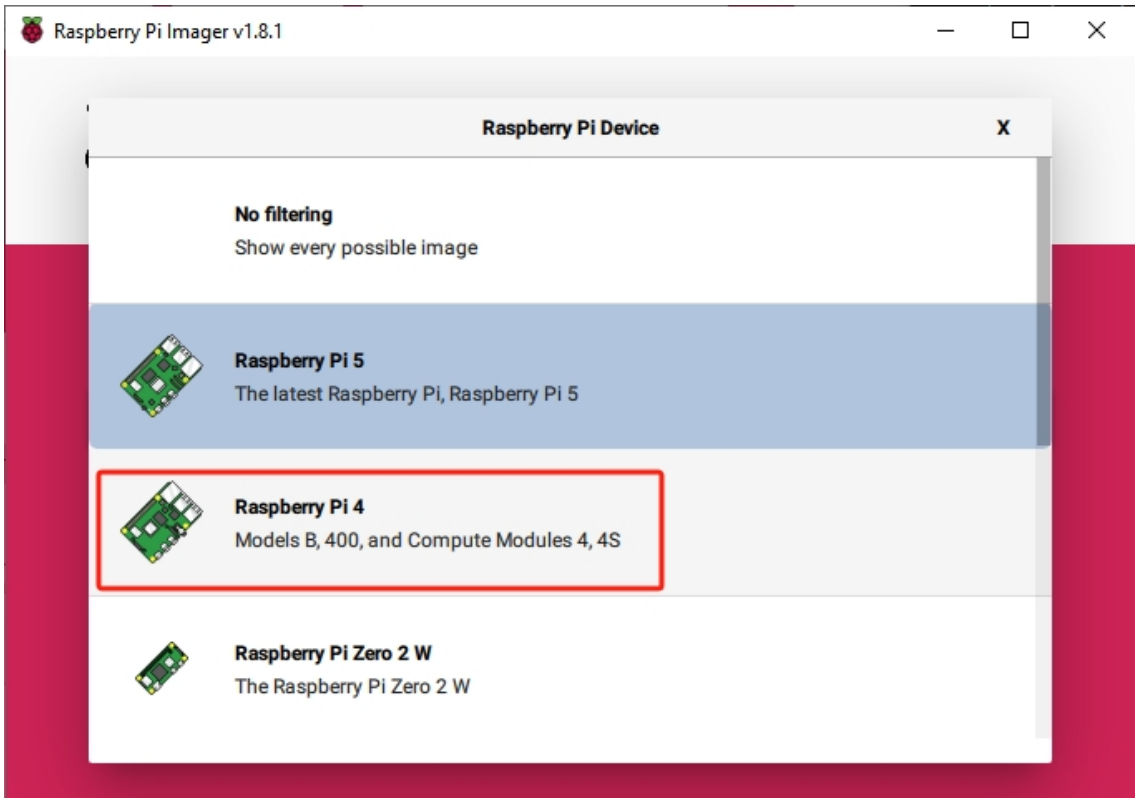


3. SD カードをコンピュータまたはラップトップの SD カードスロットに挿入します。

4. Raspberry Pi Imager アプリケーションをアイコンをクリックするか、端末で `rpi-imager` を実行して開きます。



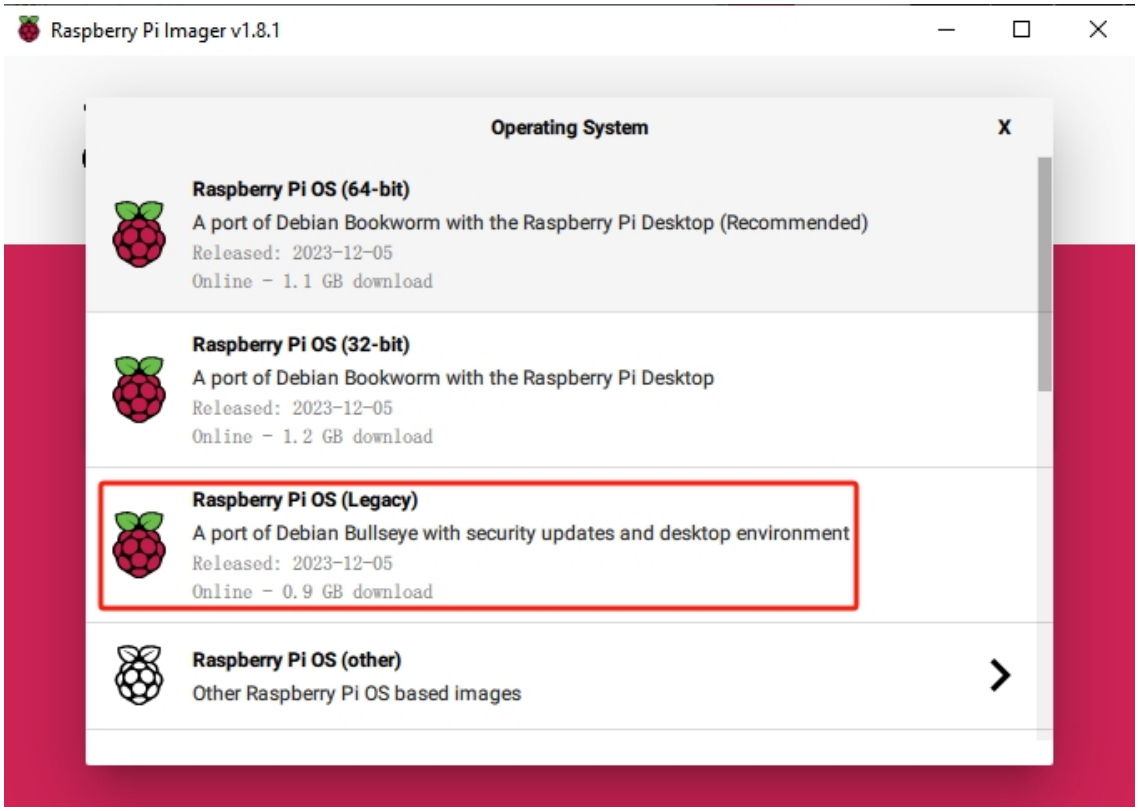
5. **CHOOSE DEVICE** をクリックし、リストから特定の Raspberry Pi モデルを選択します（注意：Raspberry Pi 5 は適用されません）。



6. **CHOOSE OS** を選択し、次に **Raspberry Pi OS (Legacy)** を選択します。

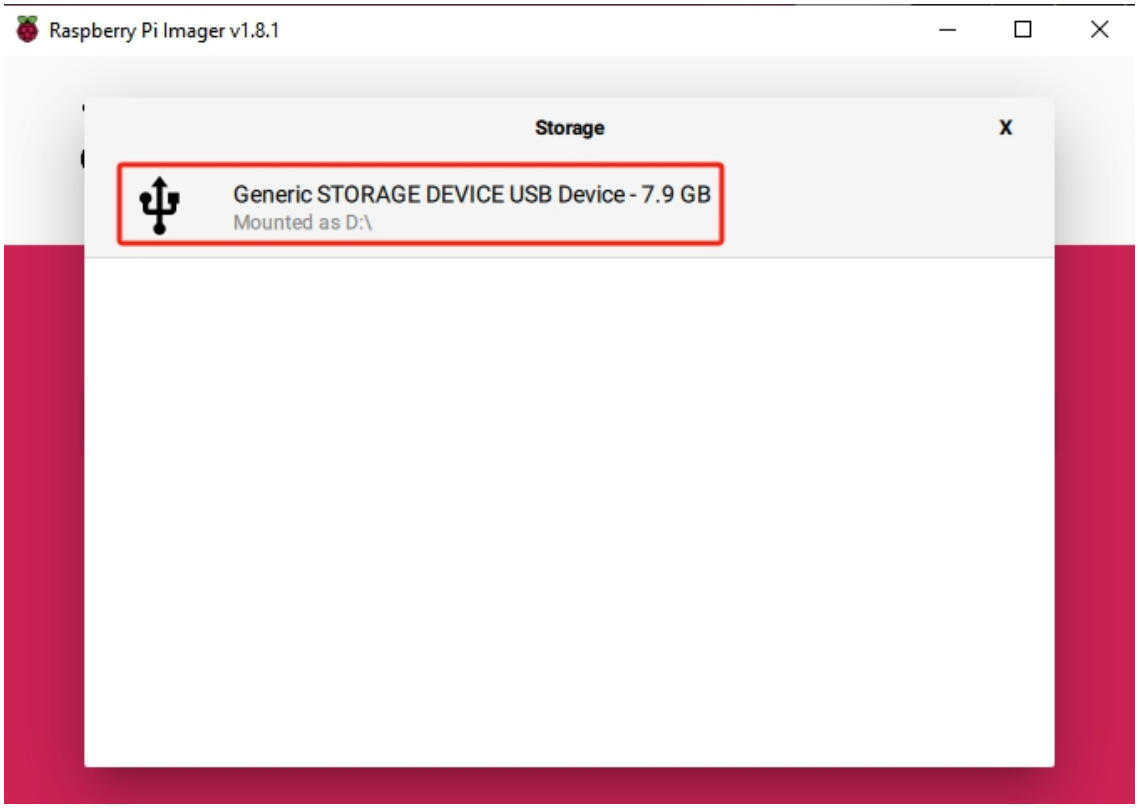
警告:

- スピーカーが動作しないため、 **Bookworm** バージョンをインストールしないでください。
- **Raspberry Pi OS (Legacy)** バージョン - **Debian Bullseye** をインストールする必要があります。

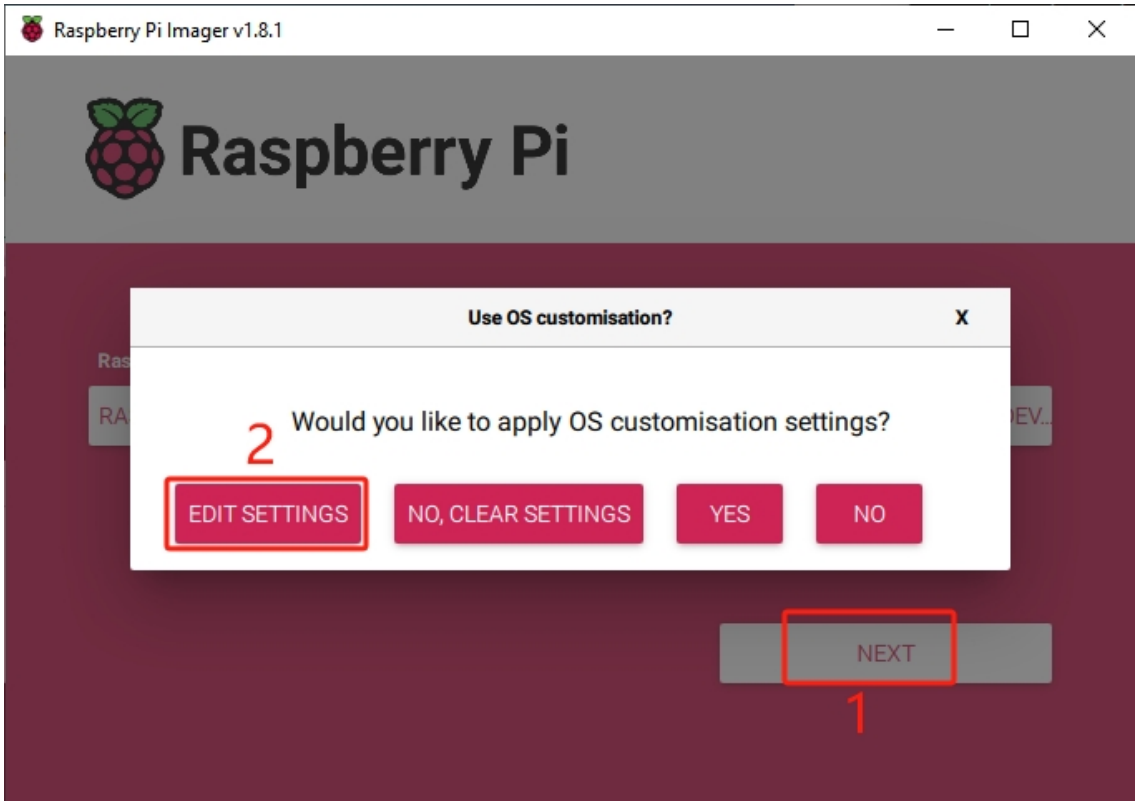


7. **Choose Storage** をクリックし、インストール用の正しいストレージデバイスを選択します。

注釈: 複数のストレージデバイスが接続されている場合は、特に正しいデバイスを選択してください。不確かな場合は他のデバイスを切断してください。

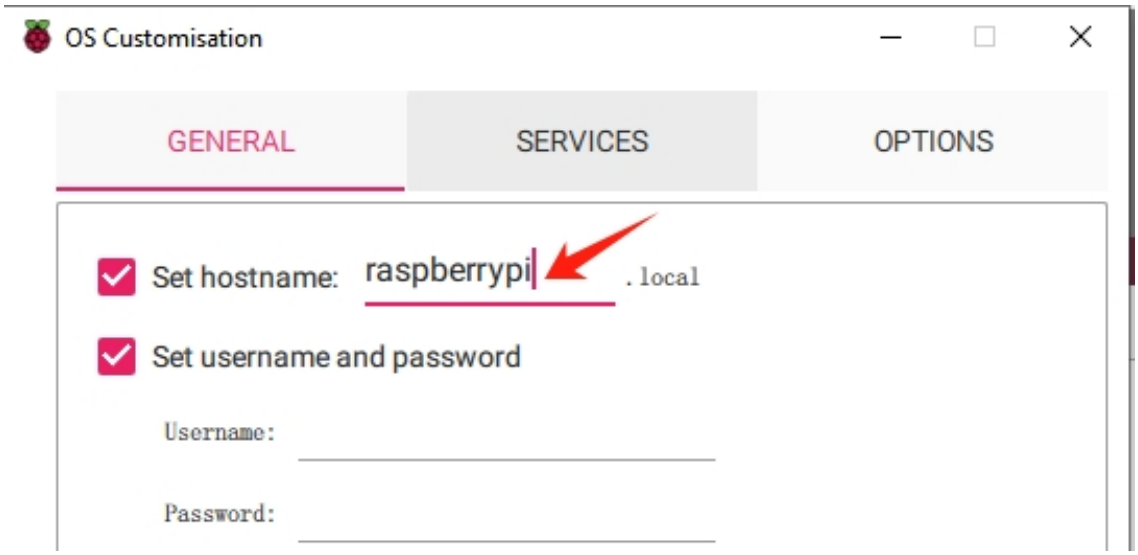


8. **NEXT** を押し、 **EDIT SETTINGS** を選択して OS の設定をカスタマイズします。



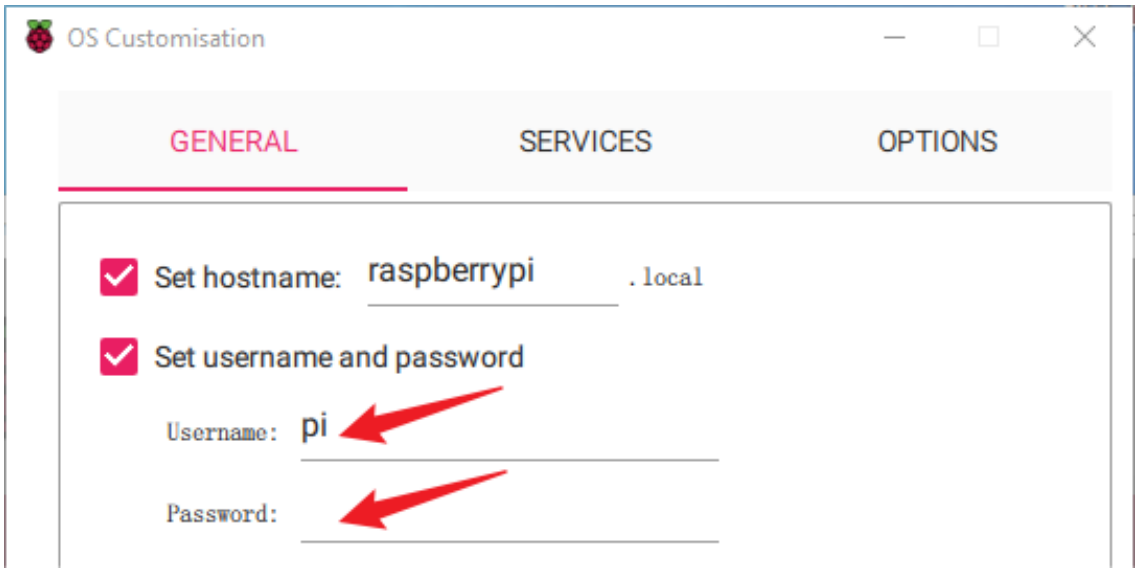
9. Raspberry Pi の **hostname** を設定します。

注釈: ホスト名は、Raspberry Pi がネットワーク上で自身を識別するために使用するものです。*<hostname>.local* または *<hostname>.lan* を使用して Pi に接続できます。



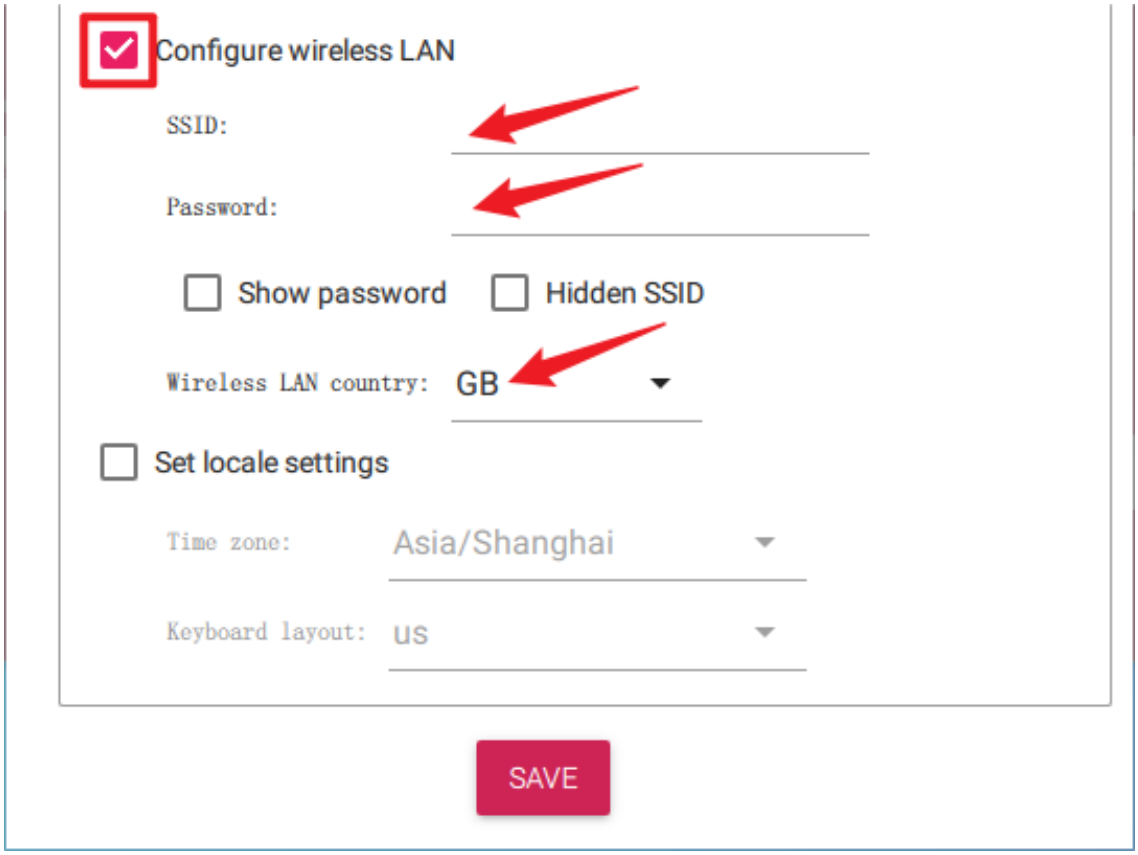
10. Raspberry Pi の管理者アカウント用に **Username** と **Password** を作成します。

注釈: Raspberry Pi にはデフォルトのパスワードがないため、セキュリティのためにユニークなユーザー名とパスワードを設定することが重要です。

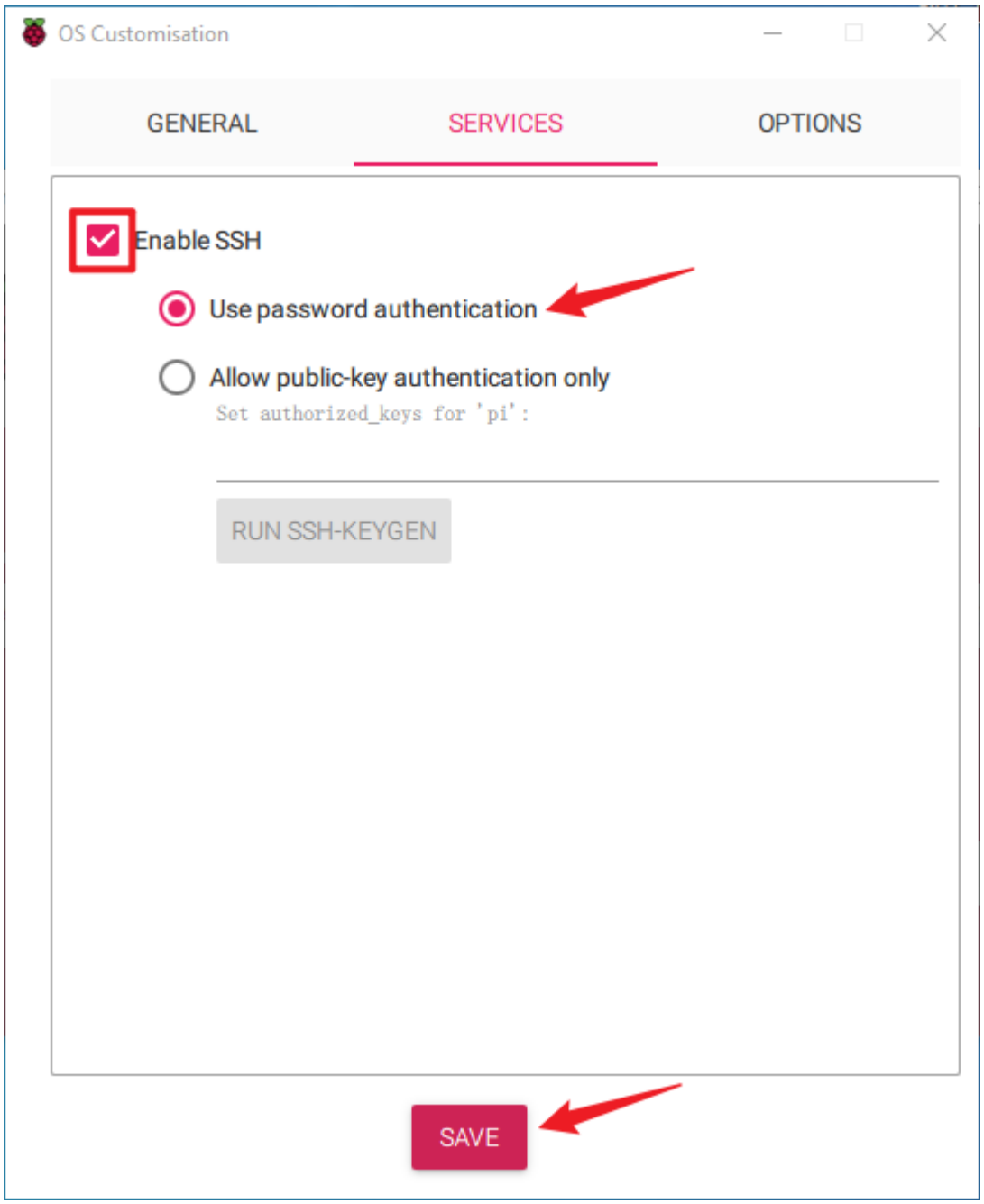


11. ワイヤレス LAN を設定するために、ネットワークの **SSID** と **Password** を入力します。

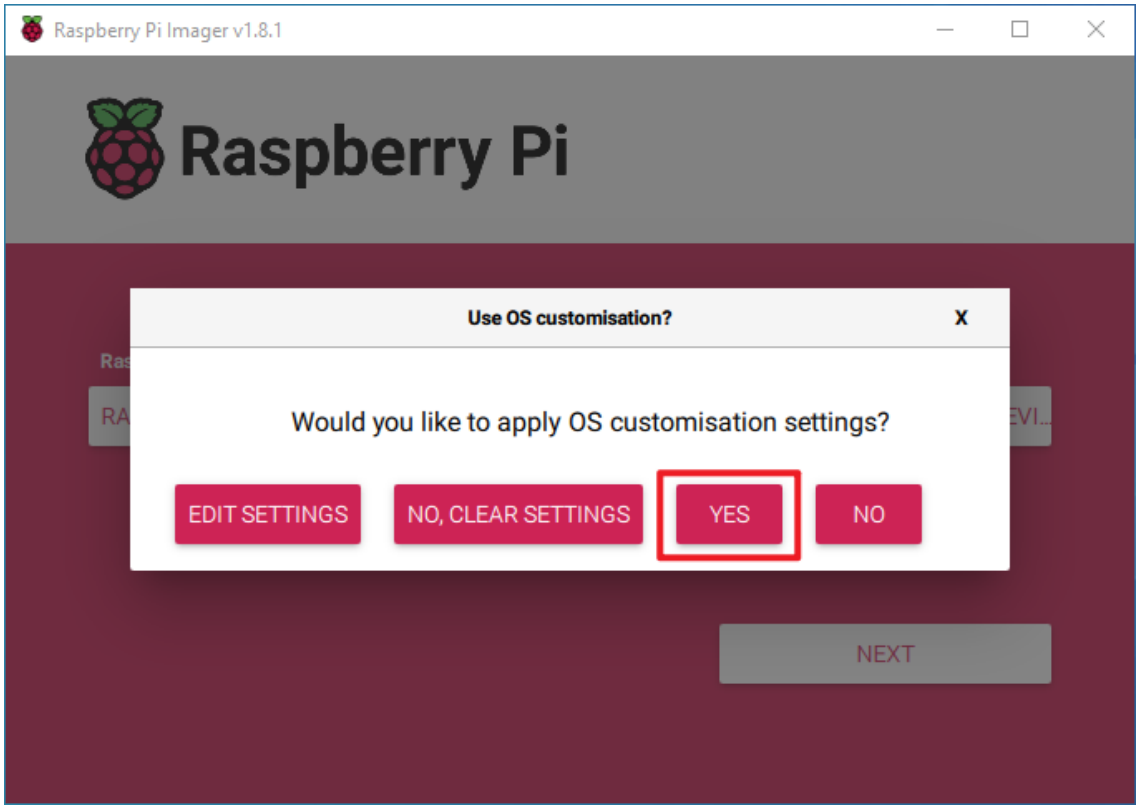
注釈: `Wireless LAN country` は、Raspberry Pi を使用している国の 2 文字の ISO/IEC alpha2 コードに設定する必要があります。



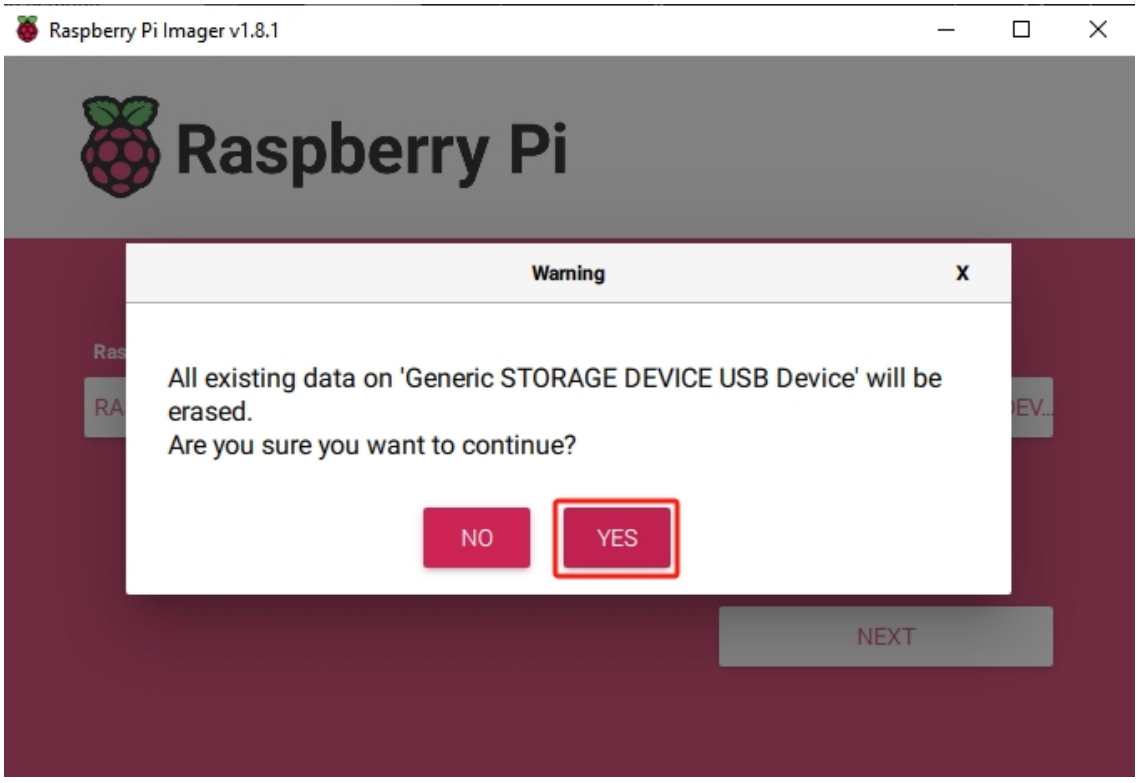
12. **SERVICES** をクリックし、パスワードベースのリモートアクセスのために **SSH** を有効にします。 **Save** をクリックすることを忘れないでください。



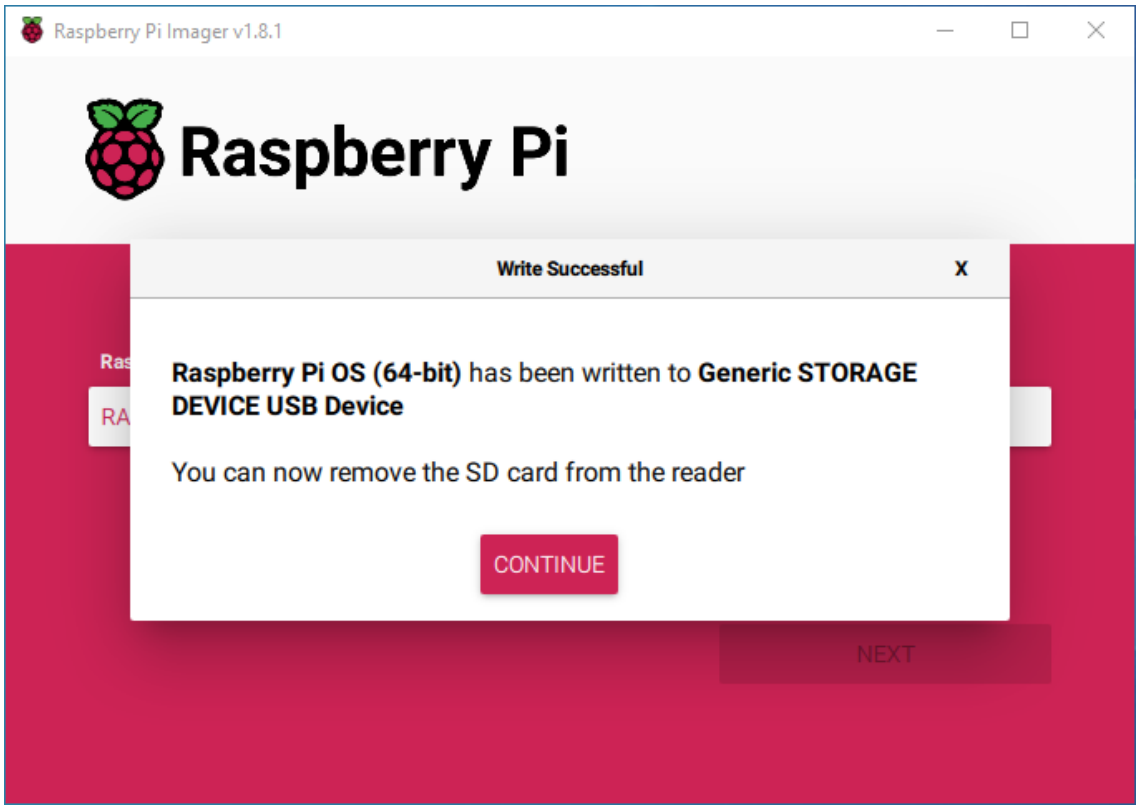
13. **Yes** をクリックして選択を確認します。



14. SD カードに既存のファイルがある場合は、データ損失を避けるためにバックアップを行ってください。バックアップが不要な場合は **Yes** をクリックして続行します。



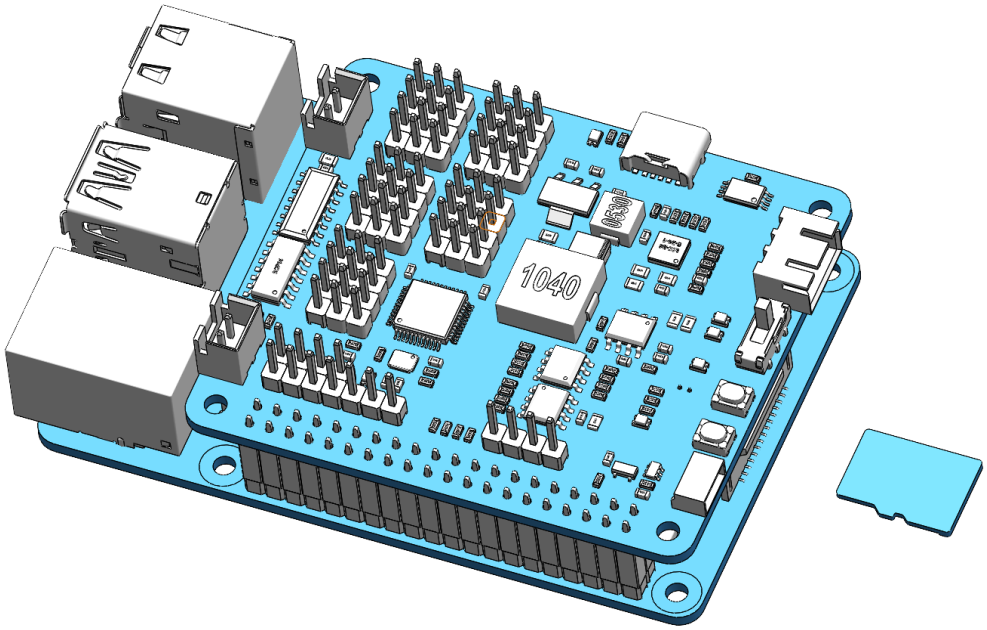
15. OS が SD カードに書き込まれるのを待ちます。完了すると、確認ウィンドウが表示されます。



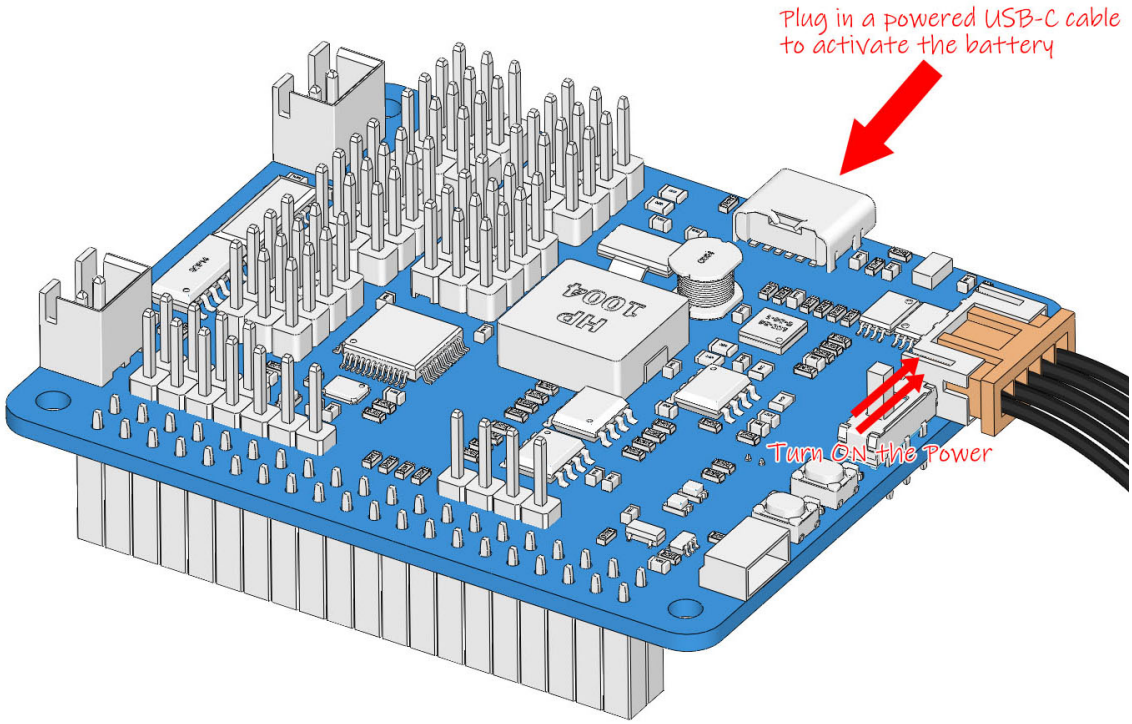
4.1.3 Raspberry Pi のセットアップ

Raspberry Pi の電源供給（重要）

1. Raspberry Pi OS でセットアップした SD カードを Raspberry Pi の下側にあるマイクロ SD カードスロットに挿入します。



2. 組み立て指示書に従って、バッテリーケーブルを挿入し、電源スイッチを ON にします。次に、USB-C ケーブルを挿入してバッテリーを起動します。1-2 分待つと、Raspberry Pi が正常に起動したことを示す音がします。



注釈: USB-C ケーブルを差し込んだままにしておくことをお勧めします。後続のソフトウェア設定プ

ロセスはかなり長いためです。

画面がある場合

注釈: ロボットにインストールされている Raspberry Pi ZERO は画面に接続するのが簡単ではありません。画面を使用せずにセットアップする方法を使用してください。

画面がある場合、Raspberry Pi での操作が簡単になります。

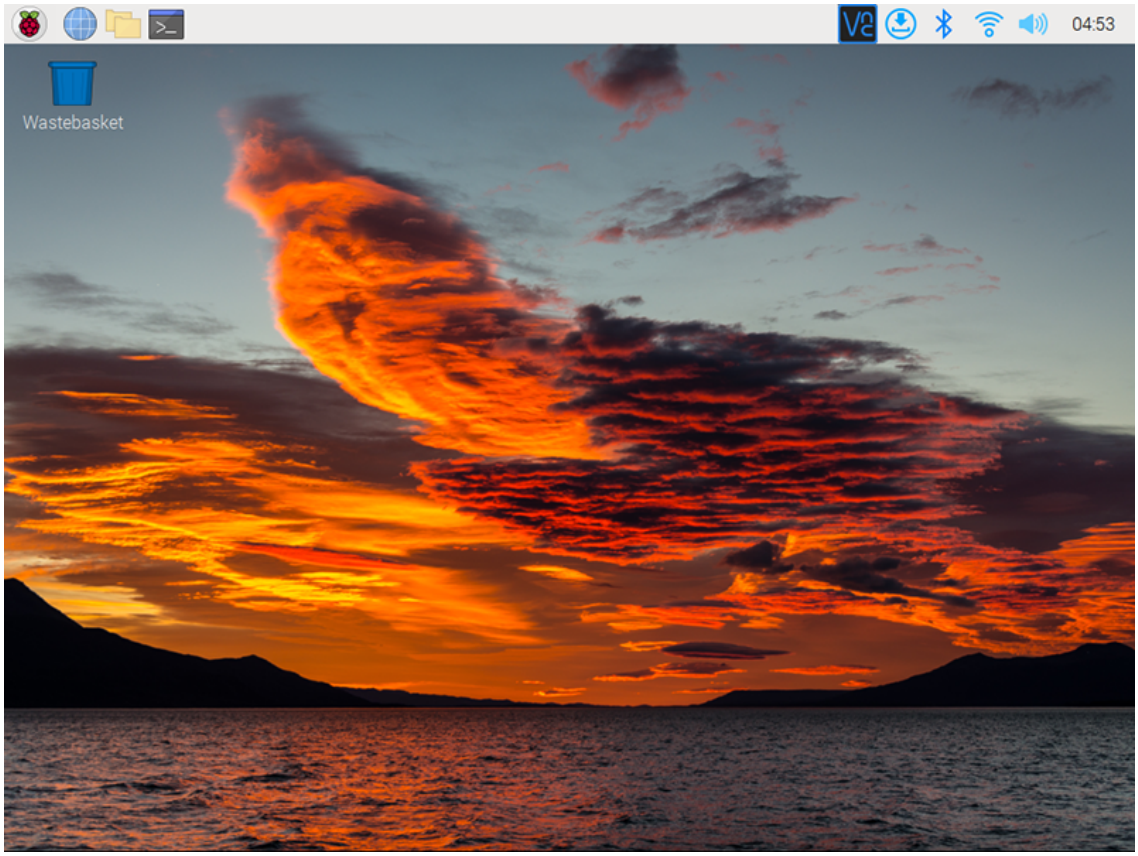
必要な部品

- 任意の Raspberry Pi
- 1 * 電源アダプター
- 1 * マイクロ SD カード
- 1 * 画面用の電源アダプター
- 1 * HDMI ケーブル
- 1 * 画面
- 1 * マウス
- 1 * キーボード

1. マウスとキーボードを接続します。
2. Raspberry Pi の HDMI ポートに画面を接続し、画面が壁のコンセントに接続され、オンになっていることを確認します。

   注釈: Raspberry Pi 4 を使用する場合、画面を HDMI0（電源入力ポートに最も近い）に接続する必要があります。

3. 電源アダプターを使用して Raspberry Pi に電源を供給します。数秒後、Raspberry Pi OS のデスクトップが表示されます。



画面がない場合

モニターがない場合は、リモートで Raspberry Pi にログインできます。

必要なコンポーネント

- – Raspberry Pi 4B/Zero 2 w/3B 3B+/2B/Zero W
- 1 * 電源アダプタ
- 1 * マイクロ SD カード

SSH コマンドを使用して Raspberry Pi の Bash シェルを開くことができます。Bash は Linux の標準デフォルトシェルです。シェル自体は、ユーザーが Unix/Linux を使用する際のコマンド（指示）です。必要なことのほとんどはシェルを通じて行うことができます。

コマンドウィンドウを使用して Raspberry Pi にアクセスすることに満足していない場合は、リモートデスクトップ機能を使用して、GUI を使用して Raspberry Pi 上のファイルを簡単に管理することもできます。

各システムの詳細なチュートリアルについては以下を参照してください。

Mac OS X ユーザー

Mac ユーザーの場合、コマンドラインからよりも VNC を使って直接 Raspberry Pi のデスクトップにアクセスする方が便利です。Raspberry Pi 側で VNC を有効にした後、Finder を介して設定されたアカウントのパスワードを入力することでアクセスできます。

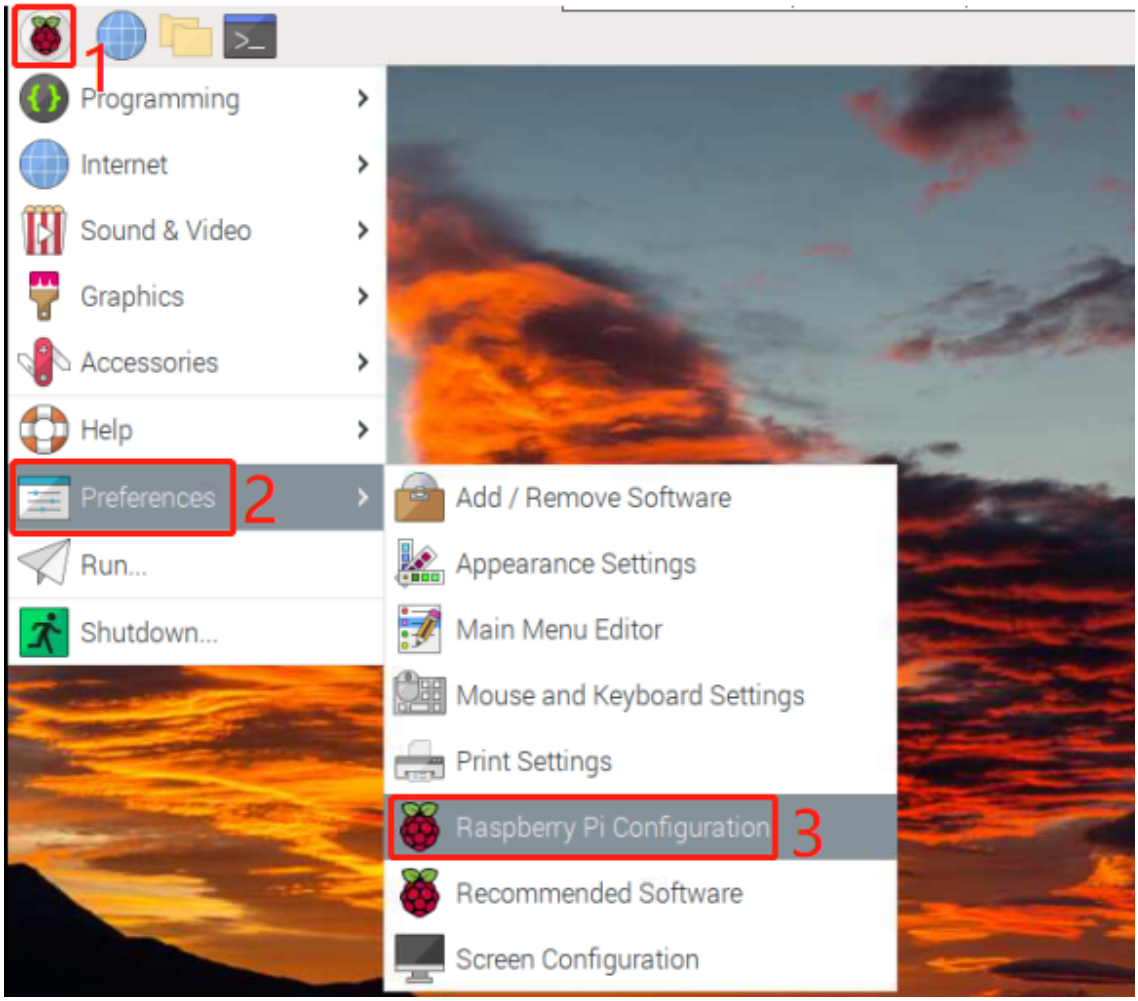
この方法では、Mac と Raspberry Pi 間の通信は暗号化されません。通信は自宅やビジネスネットワーク内で行われるため、保護されていなくても問題ありません。ただし、気になる場合は、 VNC® Viewer などの VNC アプリケーションをインストールすることができます。

一時的にモニター（テレビ）、マウス、キーボードを使用して、直接 Raspberry Pi のデスクトップを開いて VNC をセットアップできると便利です。そうでない場合でも問題ありません。SSH コマンドを使用して Raspberry Pi の Bash シェルを開き、そのコマンドを使用して VNC を設定することができます。

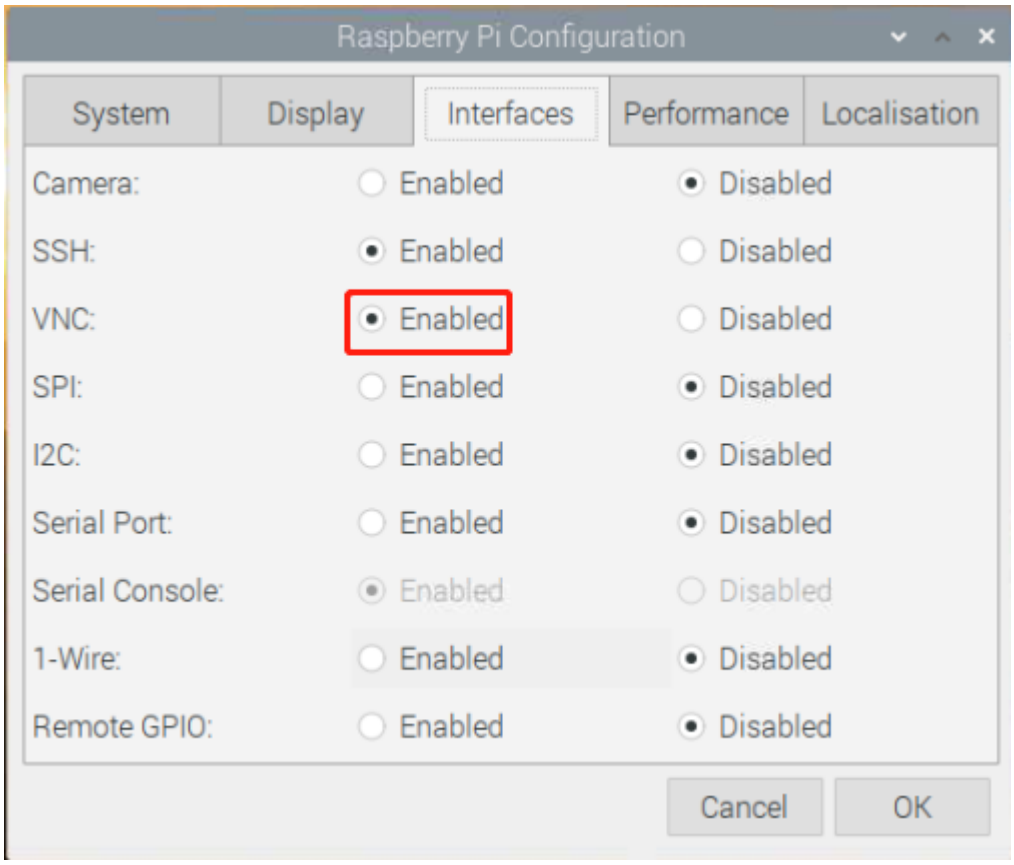
- **一時的にモニター（またはテレビ）を使用しますか？**
- **一時的なモニター（またはテレビ）がない場合**

一時的にモニター（またはテレビ）を使用しますか？

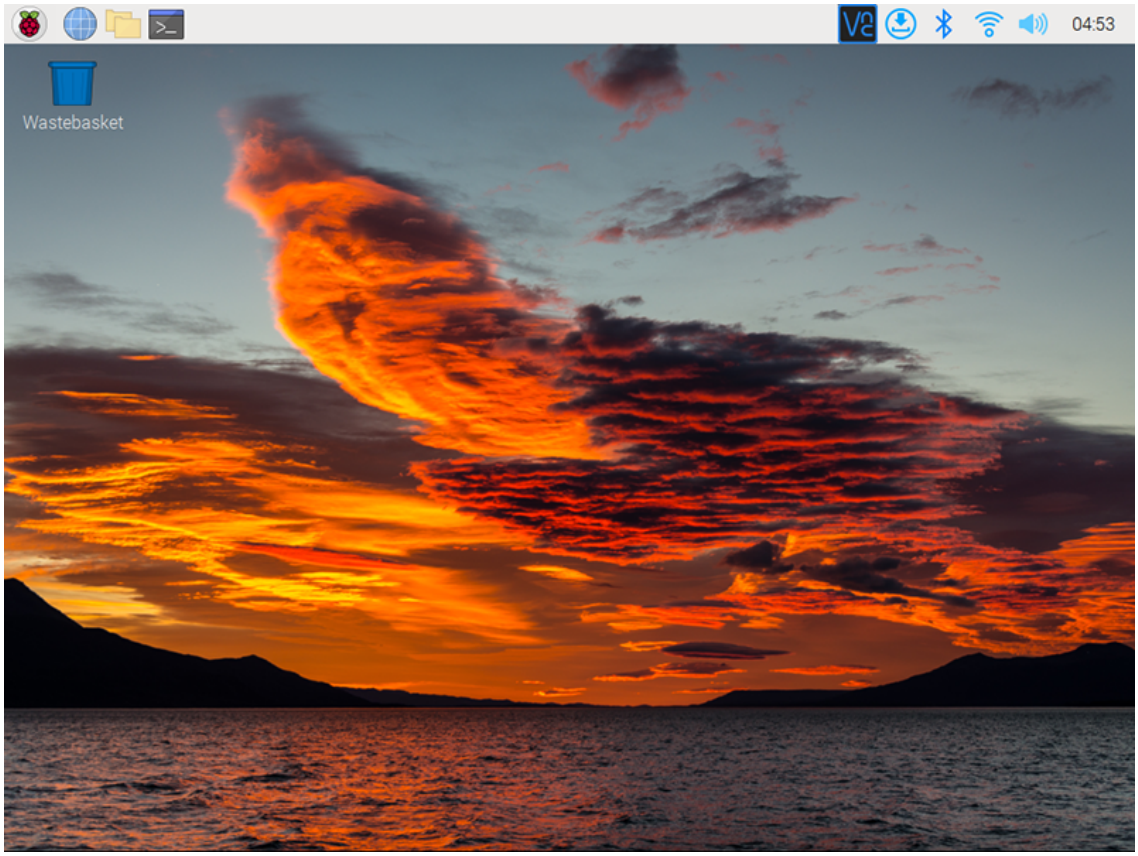
1. モニター（またはテレビ）、マウス、キーボードを Raspberry Pi に接続し、電源を入れます。図の数字に従ってメニューを選択します。



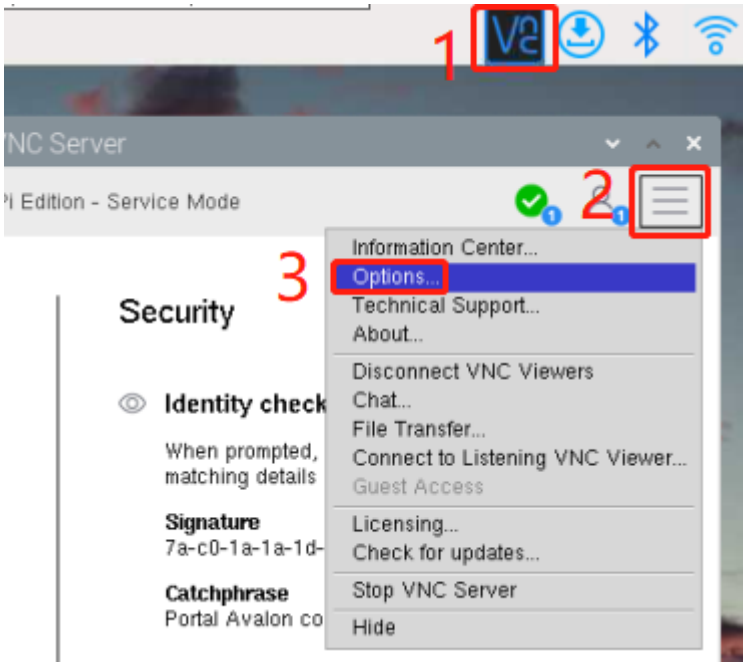
2. 次の画面が表示されます。 **Interfaces** タブで **VNC** を **Enabled** に設定し、 **OK** をクリックします。



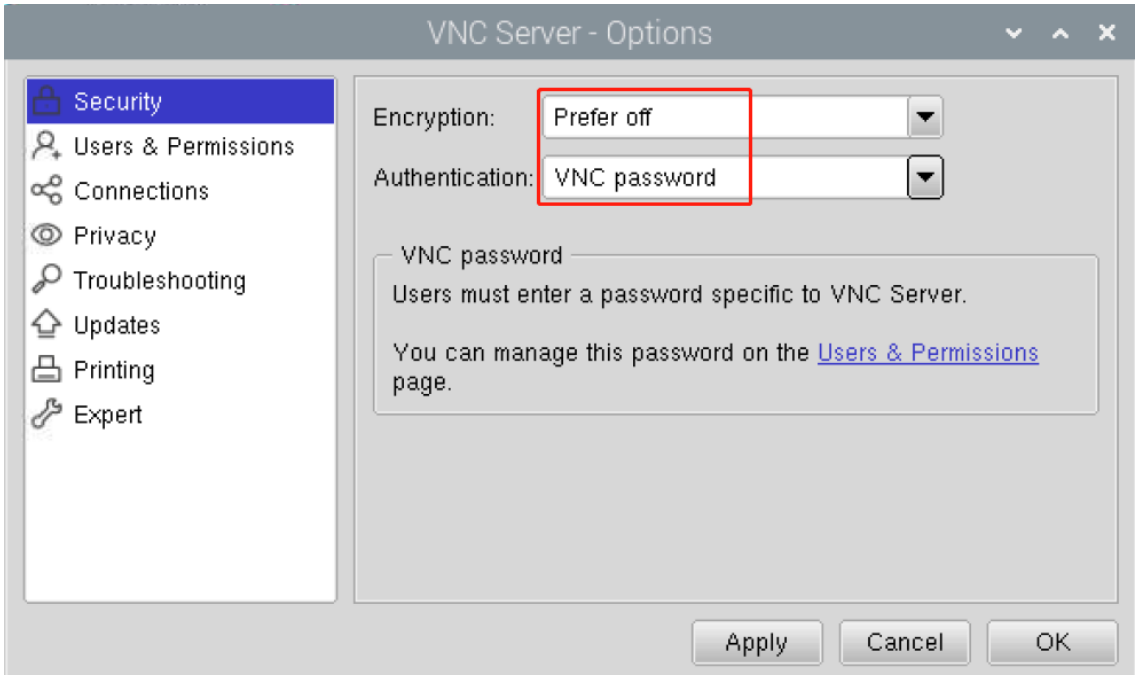
3. 画面の右上に VNC アイコンが表示され、VNC サーバーが起動します。



4. **VNC** アイコンをクリックして VNC サーバーウィンドウを開き、右上隅の **Menu** ボタンをクリックし、**Options** を選択します。

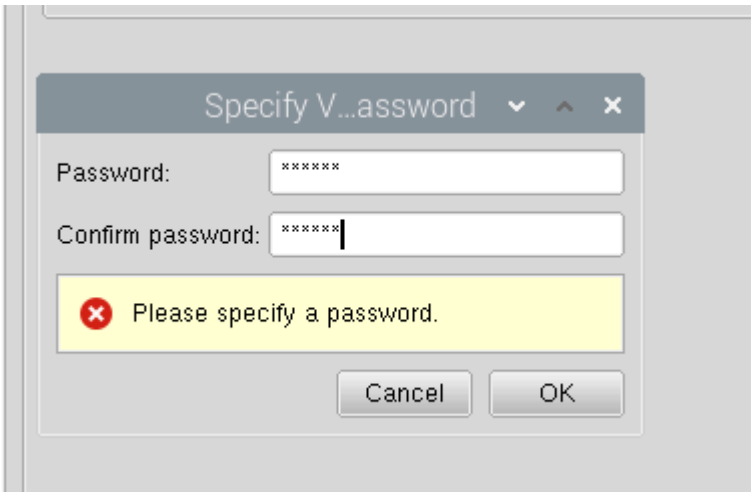


5. 次の画面が表示され、オプションを変更できます。



Encryption を **Prefer off**、 **Authentication** を **VNC password** に設定します。

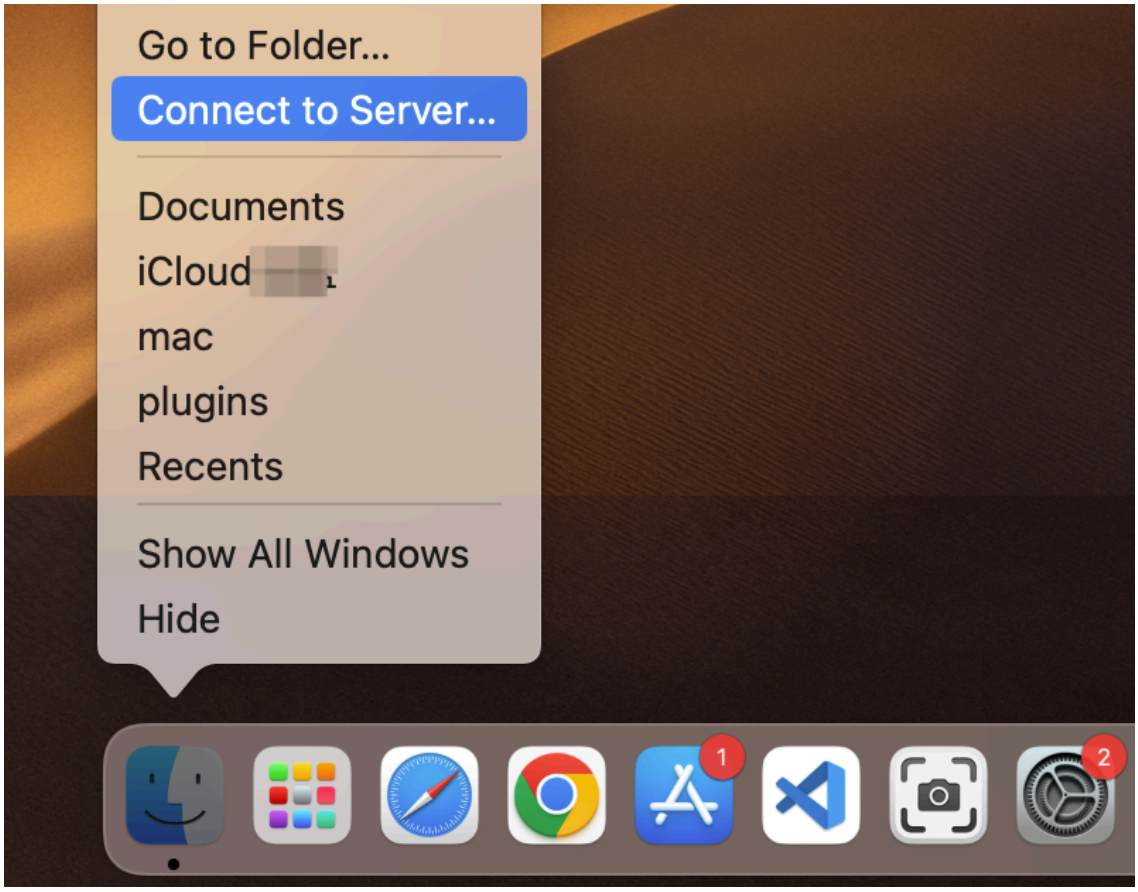
6. **OK** ボタンをクリックすると、パスワード入力画面が表示されます。Raspberry pi のパスワードと同じものを使用することも、異なるパスワードを設定することもできるので、入力して **OK** をクリックします。



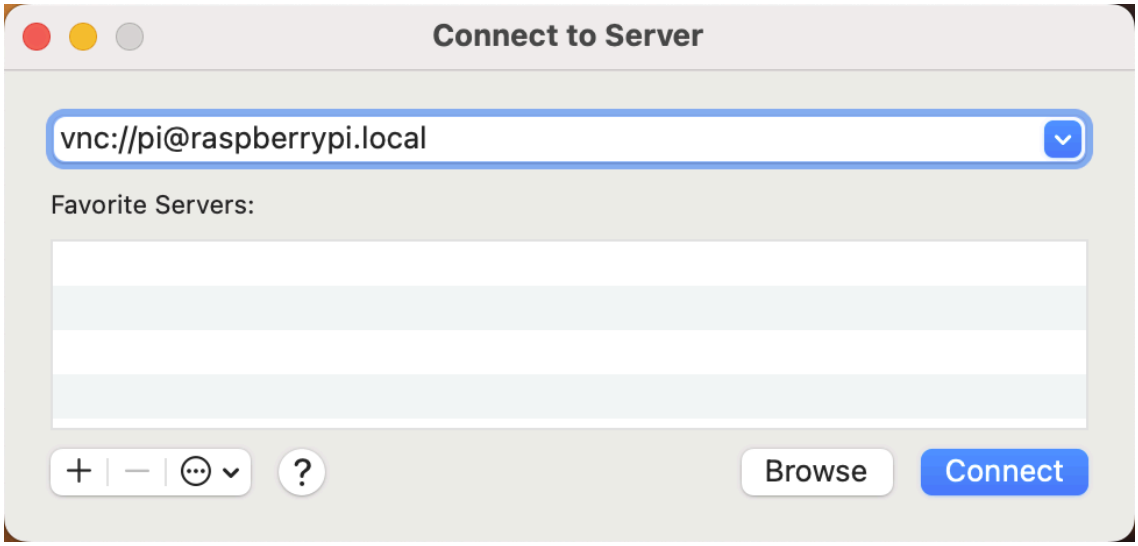
これで Mac からの接続が準備完了です。モニターを切断しても構いません。

ここからは Mac 側の操作になります。

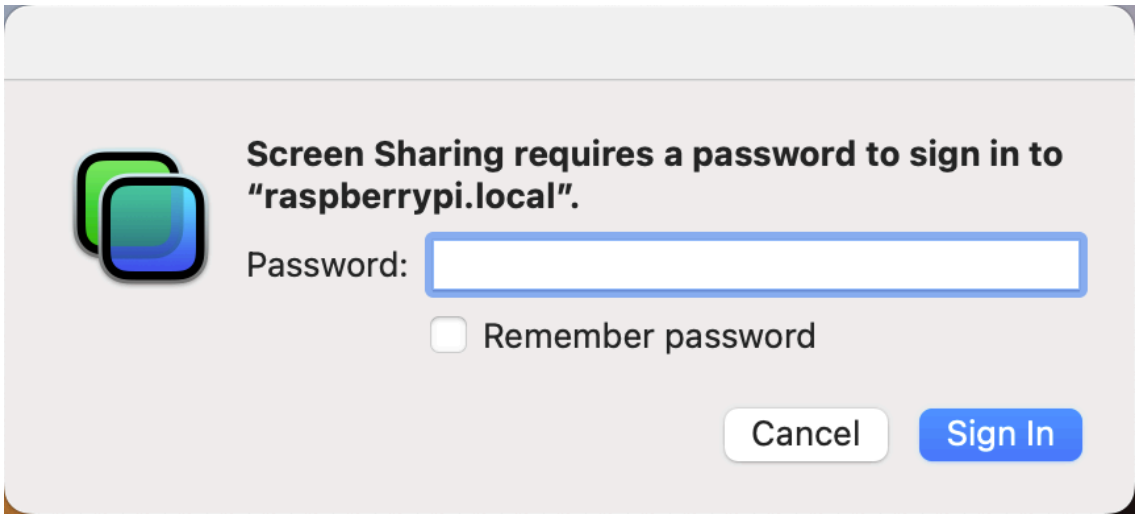
1. Finder のメニューから **Connect to Server** を選択します。右クリックで開くことができます。



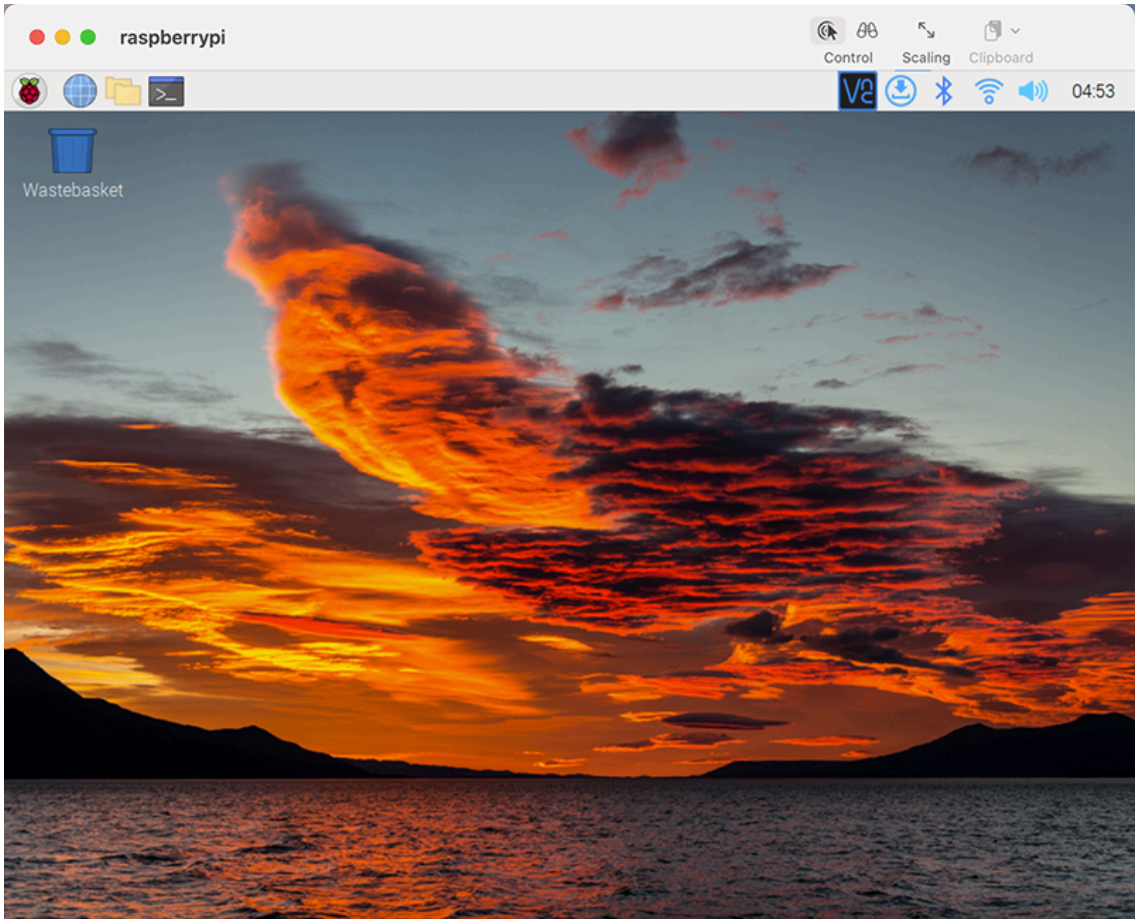
2. vnc://<username>@<hostname>.local``（または ``vnc://<username>@<IP address>）を入力します。入力した後、 **Connect** をクリックします。



3. パスワードの入力を求められるので、入力してください。



4. Raspberry pi のデスクトップが表示され、Mac からそのまま操作することができます。

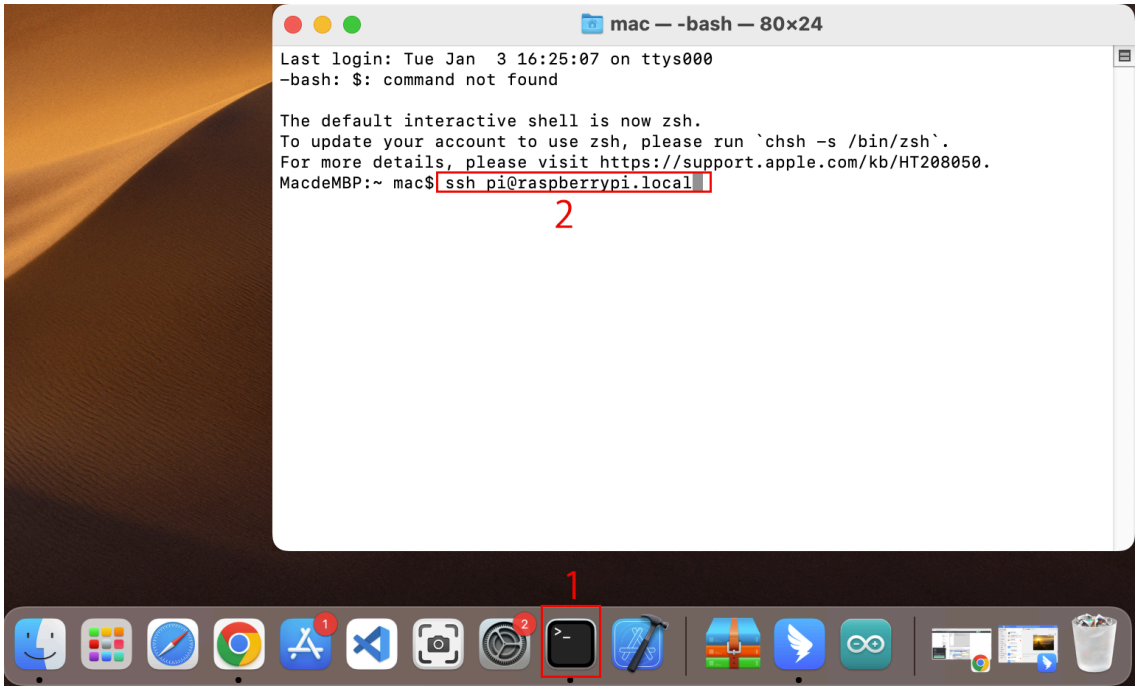


一時的なモニター（またはテレビ）がない場合

- SSH コマンドを適用して、Raspberry Pi の Bash シェルを開くことができます。
- Bash は Linux の標準デフォルトシェルです。
- シェル自体は、ユーザーが Unix/Linux を使用する際のコマンド（指示）です。
- 必要なことのほとんどはシェルを通じて行うことができます。
- Raspberry Pi 側の設定が完了した後、Mac の **Finder** から Raspberry Pi のデスクトップにアクセスできます。

1. ssh <username>@<hostname>.local と入力して Raspberry Pi に接続します。

```
ssh pi@raspberrypi.local
```



2. 最初にログインする際にのみ、以下のメッセージが表示されますので、 **yes** と入力します。

```
The authenticity of host 'raspberrypi.local␣
↪(2400:2410:2101:5800:635b:f0b6:2662:8cba)' can't be established.
ED25519 key fingerprint is SHA256:oo7x3ZSgAo032wD1tE8eW0fFM/kmewIvRwkBys6XRwg.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

3. Raspberry Pi のパスワードを入力します。入力されたパスワードは表示されませんので、間違えないよう注意してください。

```
pi@raspberrypi.local's password:
Linux raspberrypi 5.15.61-v8+ #1579 SMP PREEMPT Fri Aug 26 11:16:44 BST 2022␣
↪aarch64

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: Thu Sep 22 12:18:22 2022
pi@raspberrypi:~ $
```

4. Raspberry Pi に正常にログインできたら、次に VNC 経由で Mac からログインできるように設定します。最初のステップとして、以下のコマンドを実行してオペレーティングシステムを更新します。

```
sudo apt update
sudo apt upgrade
```

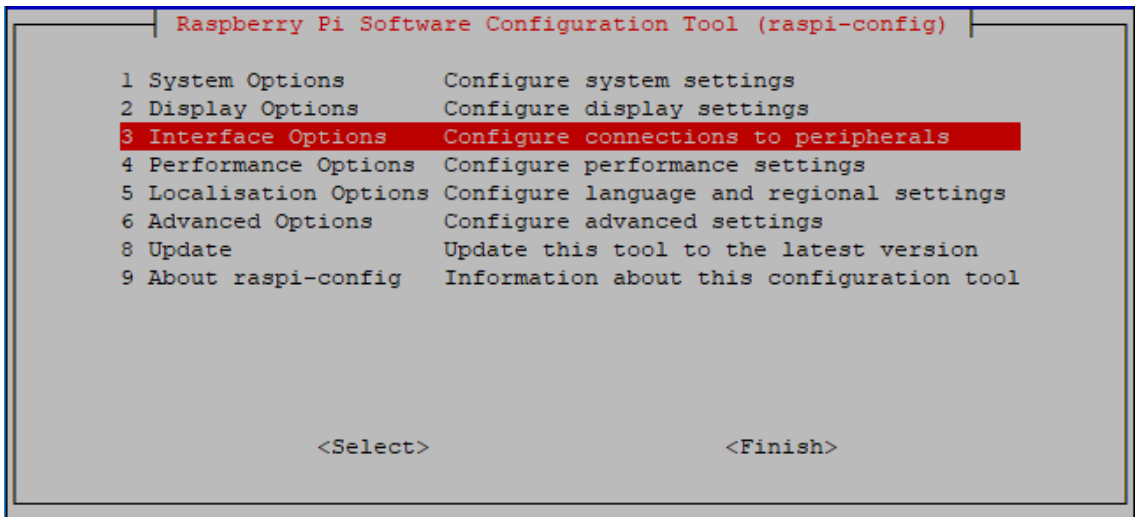
`Do you want to continue? [Y/n]` と表示されたら、 `Y` と入力してください。

更新には時間がかかることがあります。（その時の更新内容によります。）

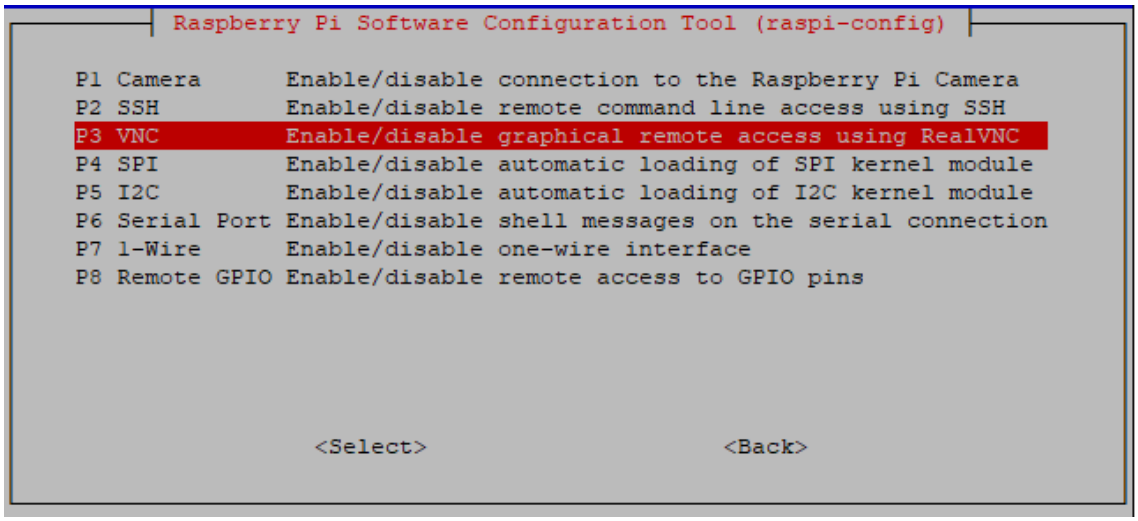
5. **VNC Server** を有効にするために、以下のコマンドを入力します。

```
sudo raspi-config
```

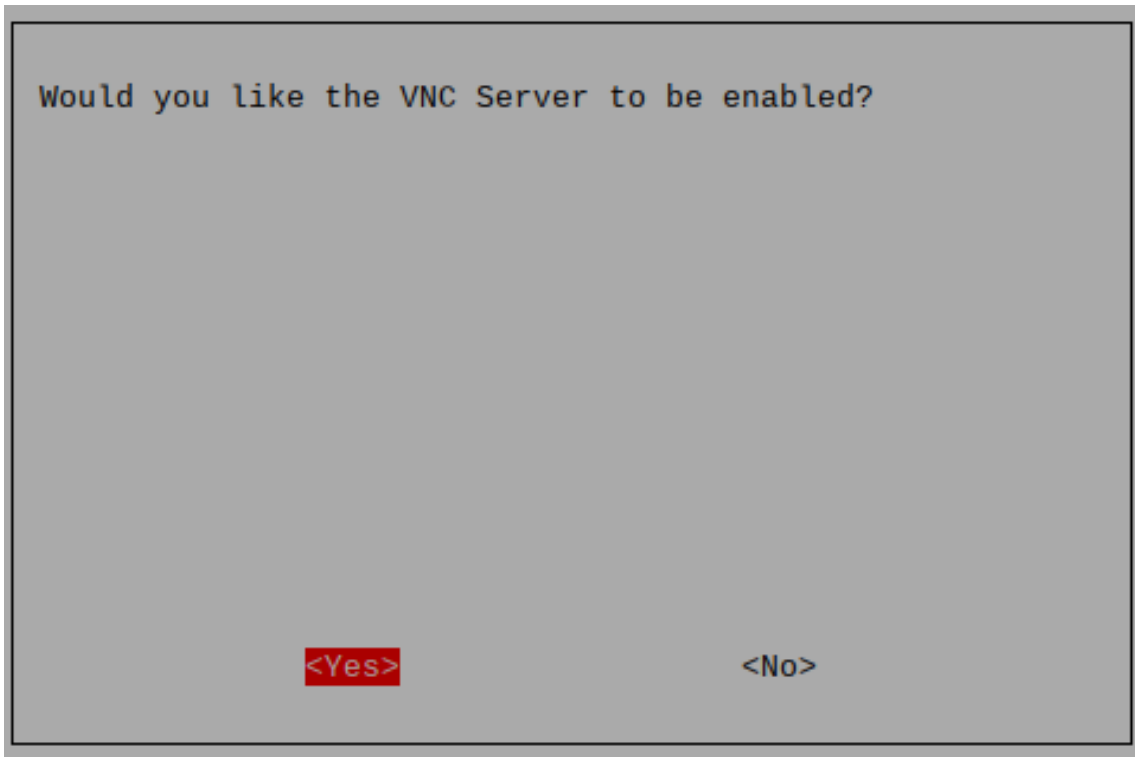
6. 次の画面が表示されます。キーボードの矢印キーを使って **Interface Options** を選択し、 **Enter** キーを押します。



7. 次に **VNC** を選択します。



8. キーボードの矢印キーで **<Yes>** -> **<OK>** -> **<Finish>** を選択して、設定を完了します。

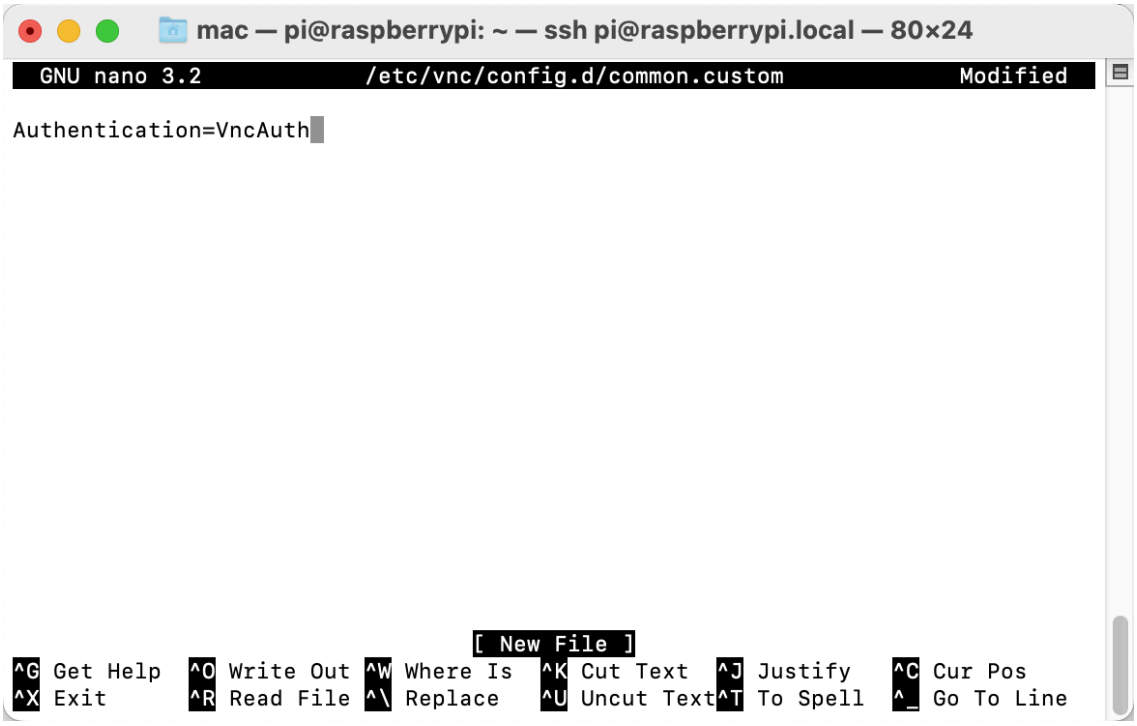


9. VNC サーバーが起動したので、Mac から接続するための設定を変更しましょう。

   コンピューター上のすべてのユーザーアカウントのすべてのプログラムのパラメータを指定するには、`/etc/vnc/config.d/common.custom` を作成します。

```
sudo nano /etc/vnc/config.d/common.custom
```

`Authentication=VncAuthenter` と入力した後、`Ctrl+X -> Y -> Enter` を押して保存して終了します。



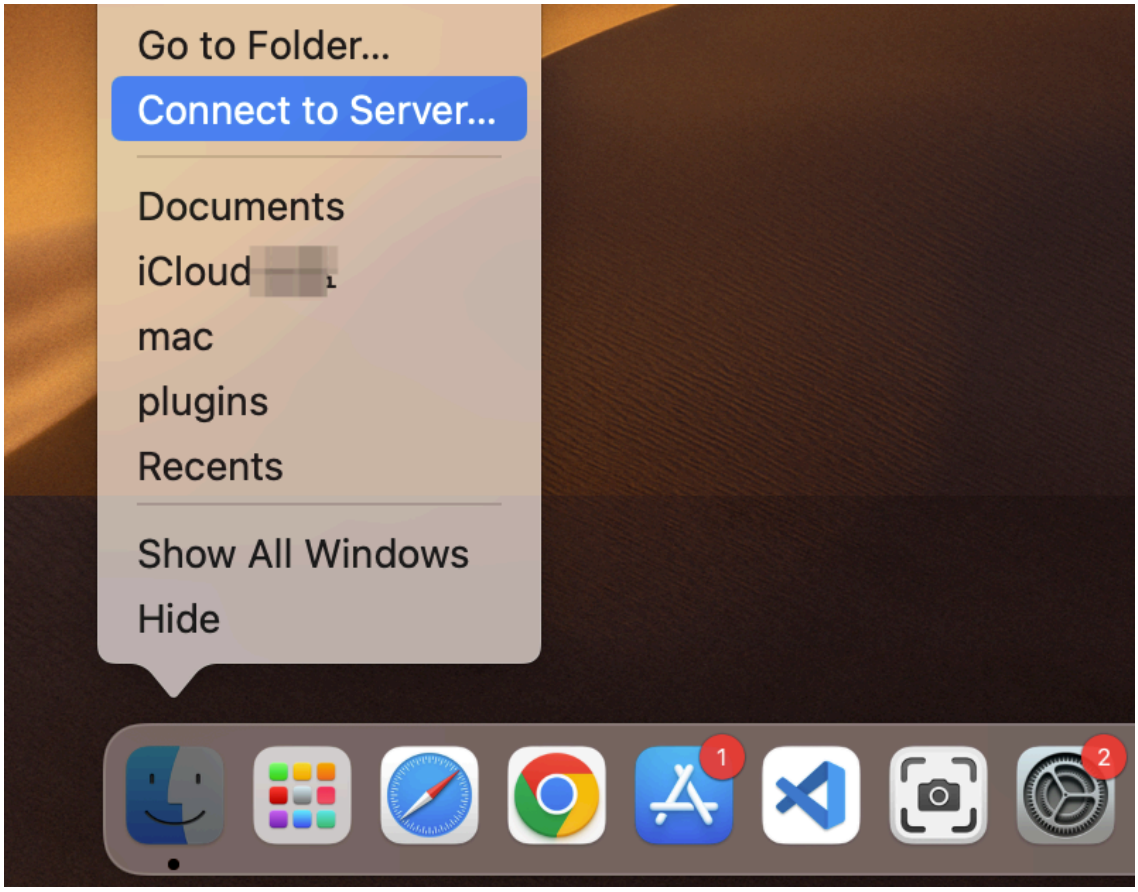
10. さらに、Mac から VNC 経由でログインするためのパスワードを設定します。Raspberry Pi のパスワードと同じものを使用することも、異なるパスワードを使用することもできます。

```
sudo vncpasswd -service
```

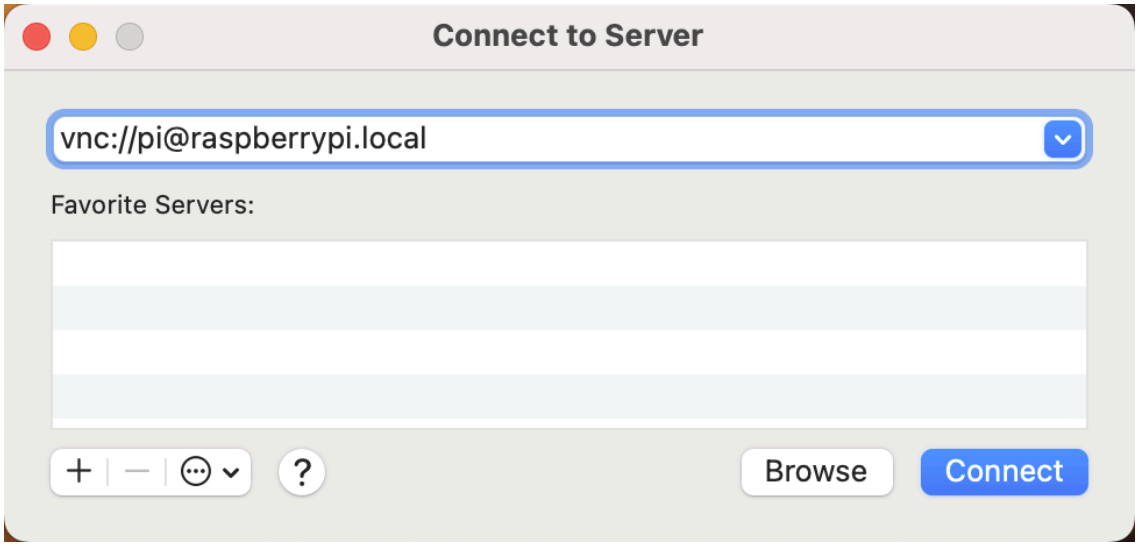
11. 設定が完了したら、Raspberry Pi を再起動して変更を適用します。

```
sudo sudo reboot
```

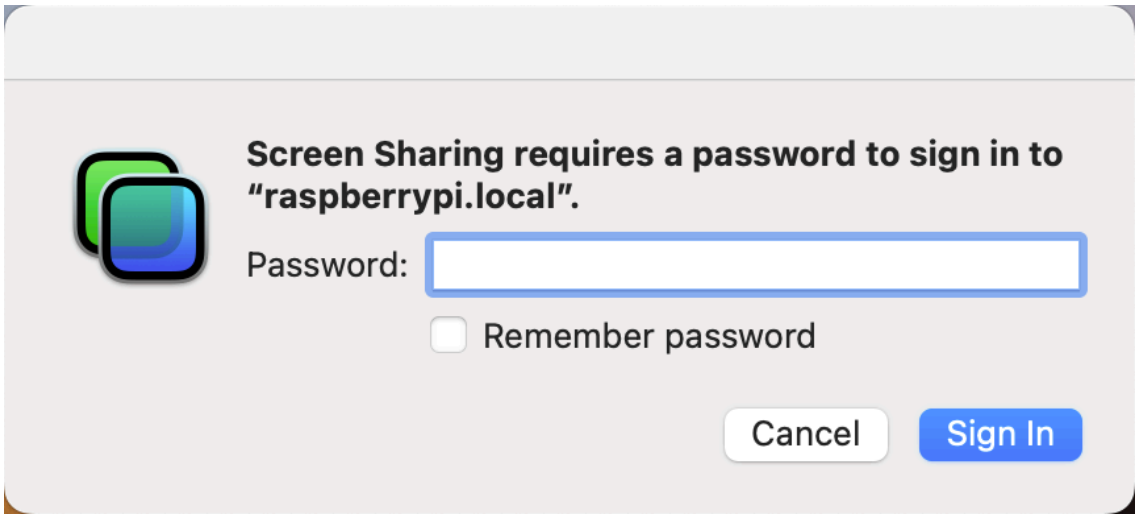
12. これで、右クリックで開くことができる **Finder** のメニューから **Connect to Server** を選択します。



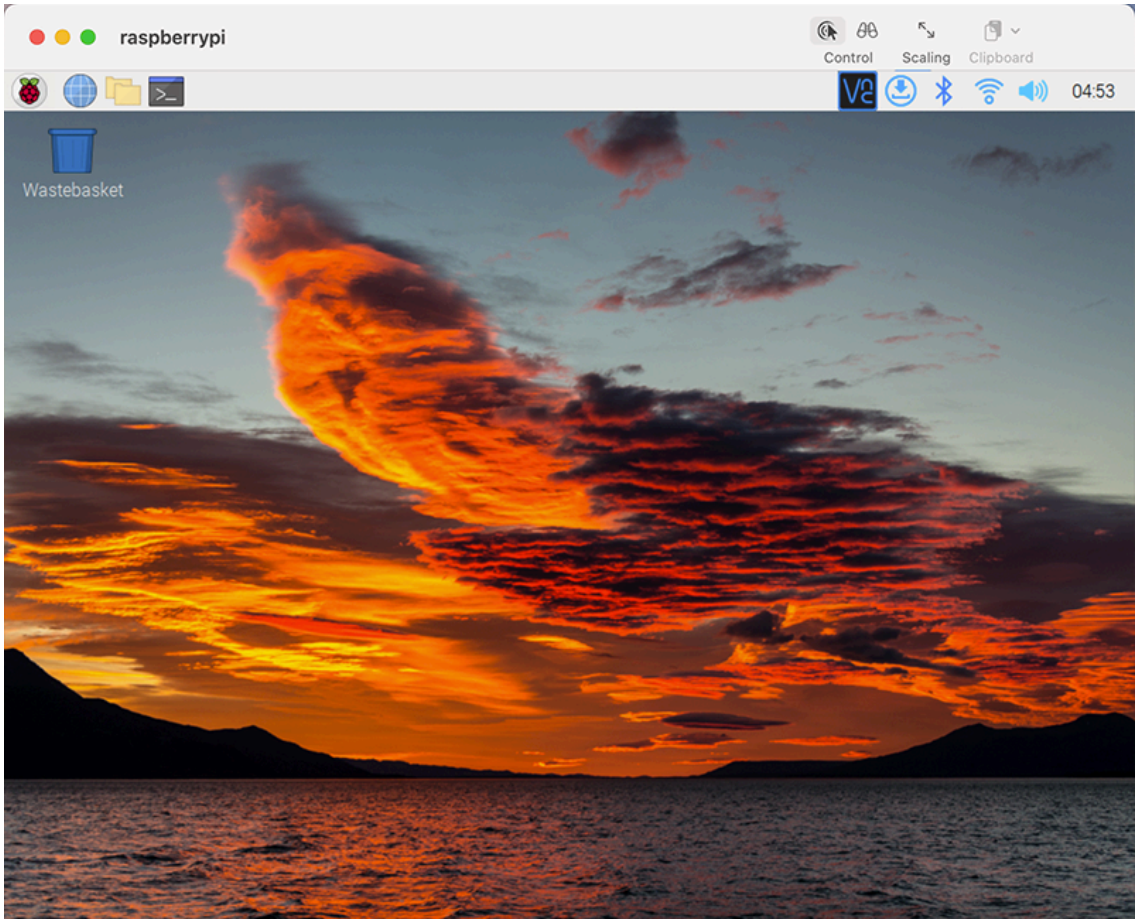
13. vnc://<username>@<hostname>.local``（または ``vnc://<username>@<IP address>）を入力します。入力した後、 **Connect** をクリックします。



14. パスワードの入力を求められるので、入力してください。



15. Raspberry Pi のデスクトップが表示され、Mac からそのまま操作できるようになります。

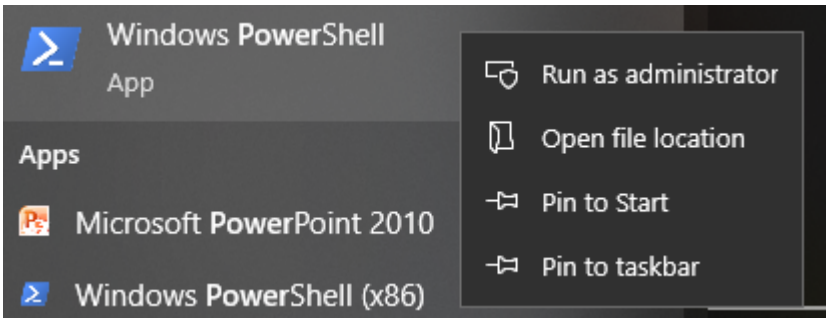


Windows ユーザー

リモートで Raspberry Pi にログイン

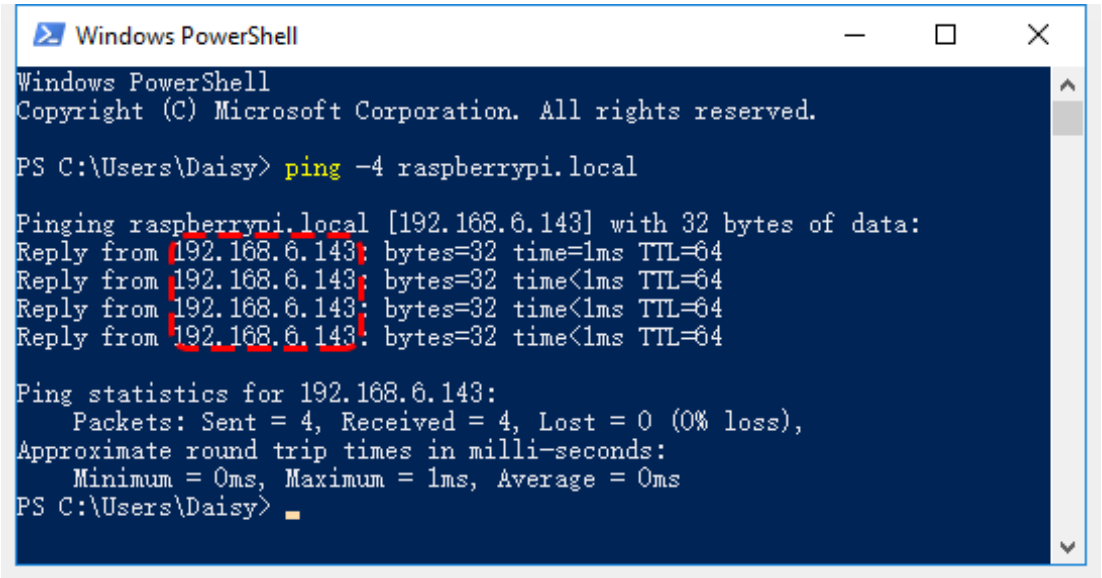
win10 を使用している場合、以下の方法でリモートで Raspberry Pi にログインできます。

1. Windows デスクトップの検索ボックスに `powershell` と入力し、 `Windows PowerShell` を右クリックして、表示されるメニューから `Run as administrator` を選択します。



2. `ping -4 <hostname>.local` と入力して、Raspberry Pi の IP アドレスを確認します。

```
ping -4 raspberrypi.local
```



上記のように、ネットワークに接続された後、Raspberry Pi の IP アドレスが表示されます。

- ターミナルが `Ping request could not find host pi.local. Please check the name and try again.` と表示された場合は、入力したホスト名が正しいか確認してください。
- それでも IP を取得できない場合は、Raspberry Pi のネットワークまたは WiFi 設定を確認してください。

3. これで、 ssh <username>@<hostname>.local （または ssh <username>@<IP address>）を使用して Raspberry Pi にログインできます。

```
ssh pi@raspberrypi.local
```

> **警告:** The term 'ssh' is not recognized as the name of a cmdlet... というプロンプトが表示された場合、
>
> システムが古く、ssh ツールがプリインストールされていないことを意味します。手動で *Powershell* **を使って** *OpenSSH* **をインストールする** を行う必要があります。
>
> または、*PuTTY* のようなサードパーティツールを使用することもできます。

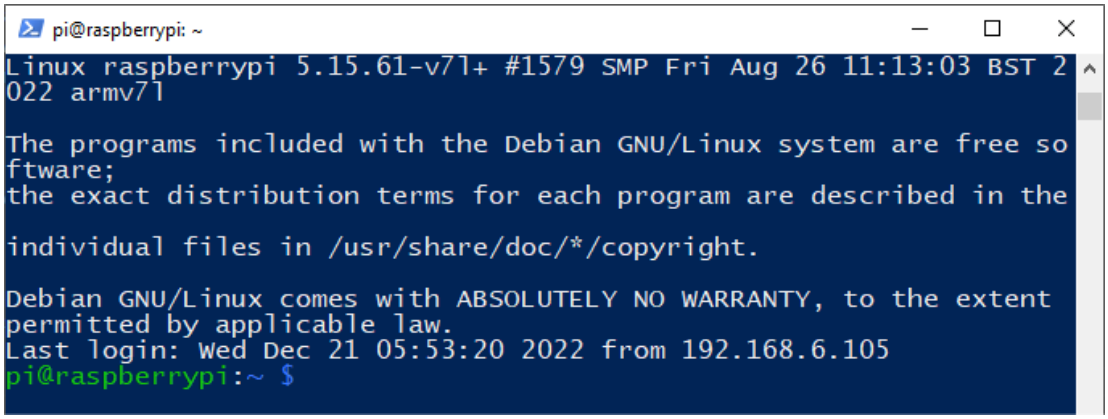
4. 最初にログインする際にのみ、以下のメッセージが表示されるので、 yes と入力します。

```
The authenticity of host 'raspberrypi.local␣
↪(2400:2410:2101:5800:635b:f0b6:2662:8cba)' can't be established.
ED25519 key fingerprint is SHA256:oo7x3ZSgAo032wD1tE8eW0fFM/kmewIvRwkBys6XRwg.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

5. 以前に設定したパスワードを入力します（私の場合は raspberry ）。

> **注釈:** パスワードを入力するとき、ウィンドウ上に文字が表示されないのは正常です。正しいパスワードを入力してください。

6. これで Raspberry Pi に接続され、次のステップに進む準備ができました。



リモートデスクトップ

コマンドウィンドウを使用して Raspberry Pi にアクセスすることに満足していない場合、リモートデスクトップ機能を使用して、GUI を使用して Raspberry Pi 上のファイルを簡単に管理することもできます。

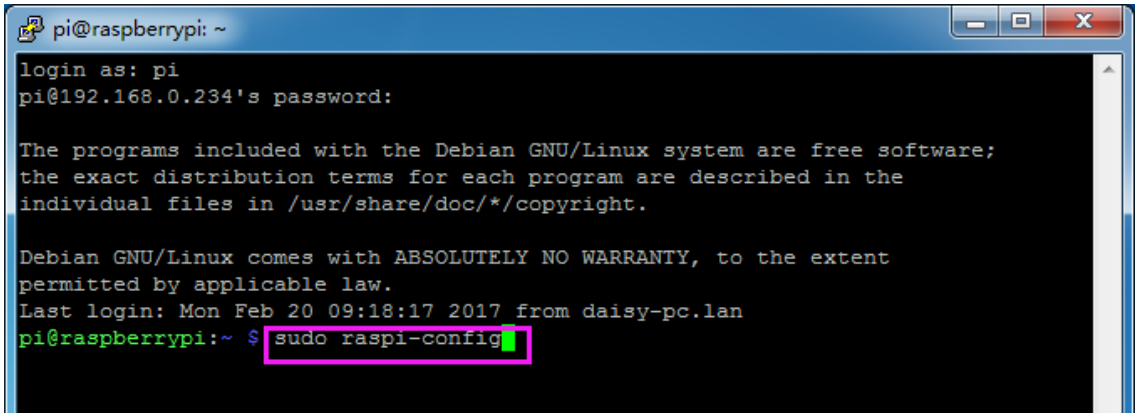
ここでは VNC® Viewer を使用します。

VNC サービスの有効化

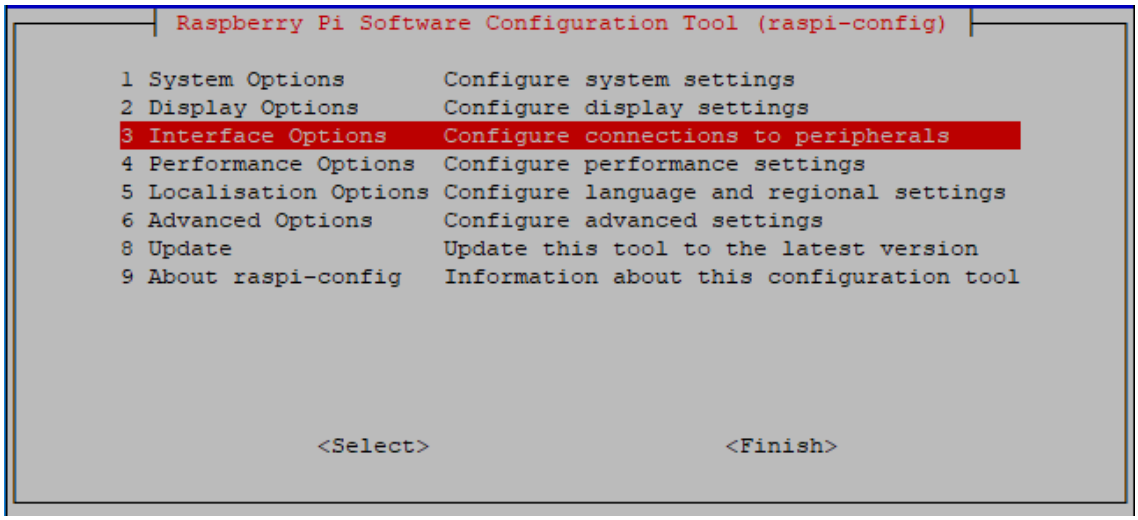
VNC サービスはシステムにインストールされています。デフォルトでは、VNC は無効になっています。config で有効にする必要があります。

1. 次のコマンドを入力します：

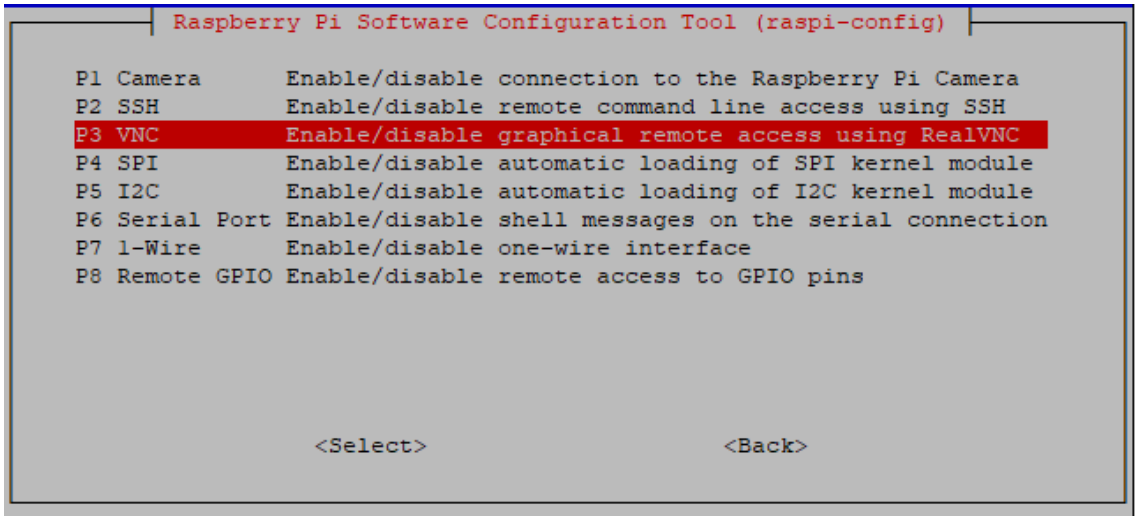
```
sudo raspi-config
```



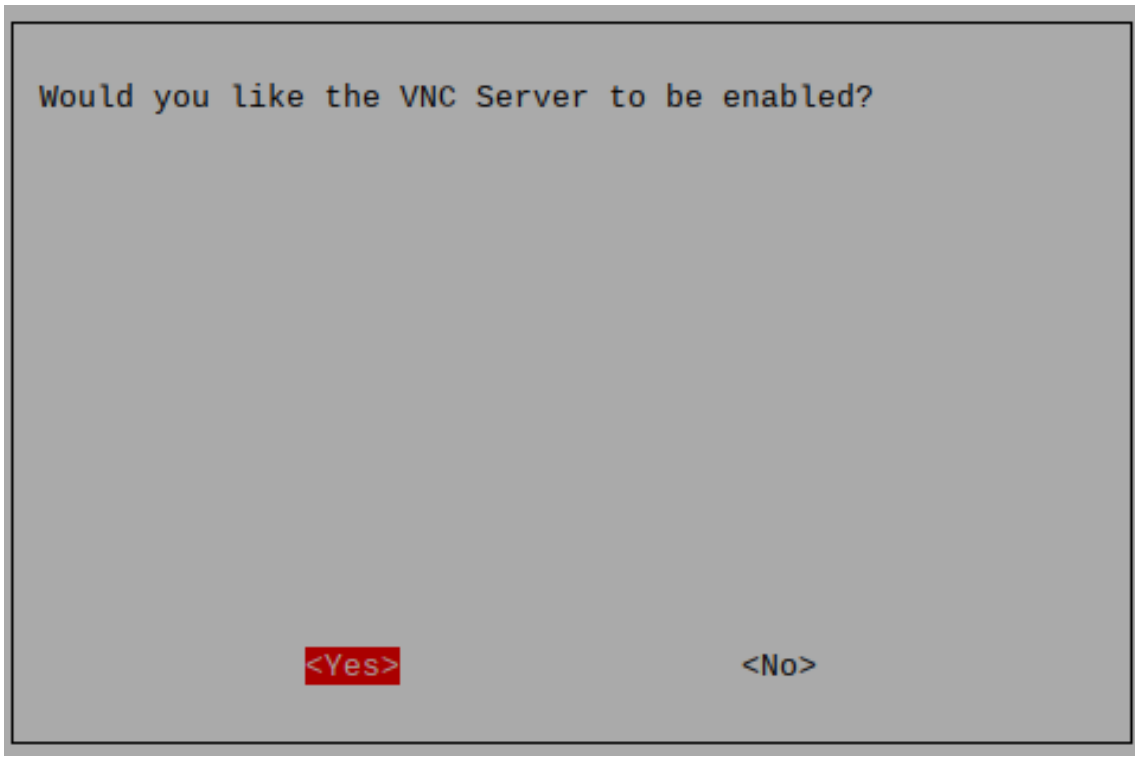
2. キーボードの下矢印キーを押して **3 Interfacing Options** を選択し、 **Enter** キーを押します。



3. 次に **VNC** を選択します。

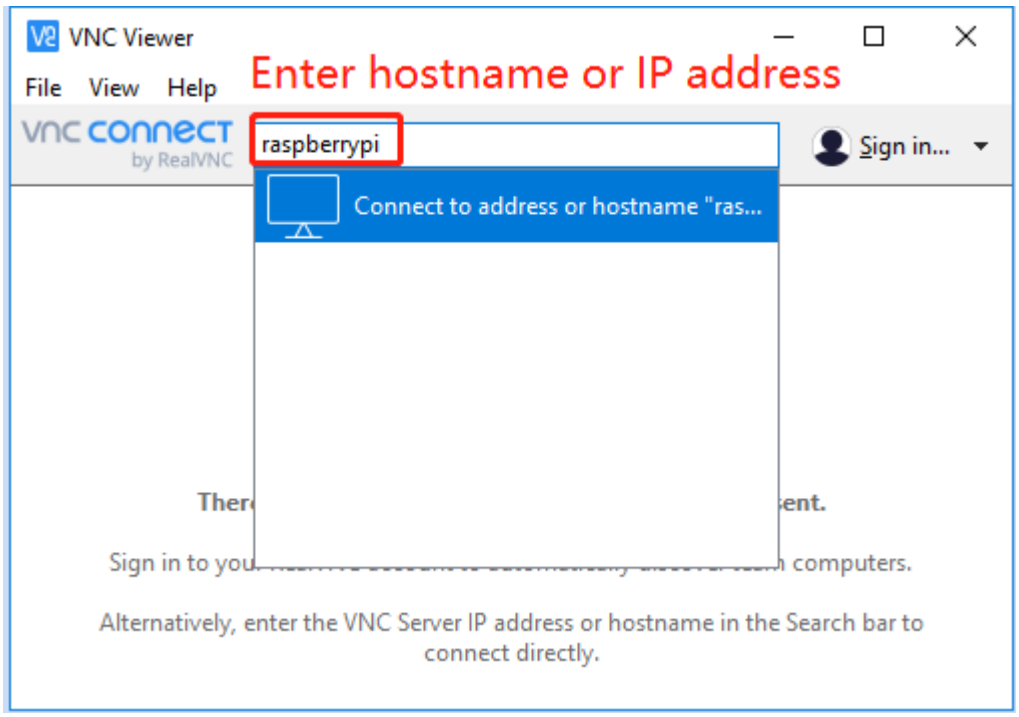


4. キーボードの矢印キーで **<Yes>** -> **<OK>** -> **<Finish>** を選択して、設定を完了します。

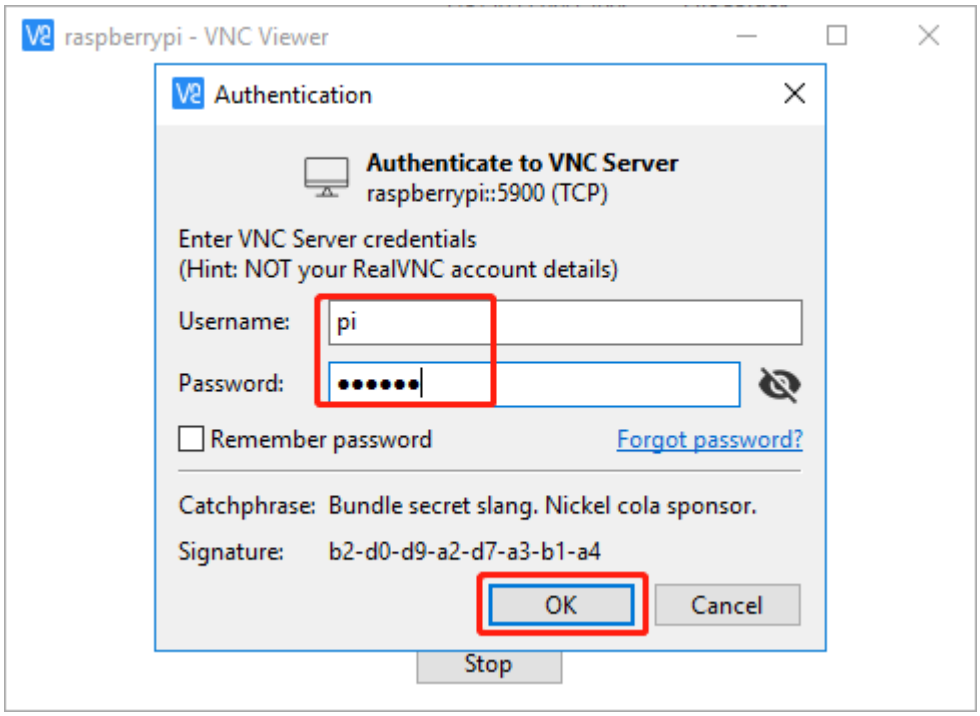


VNC でログイン

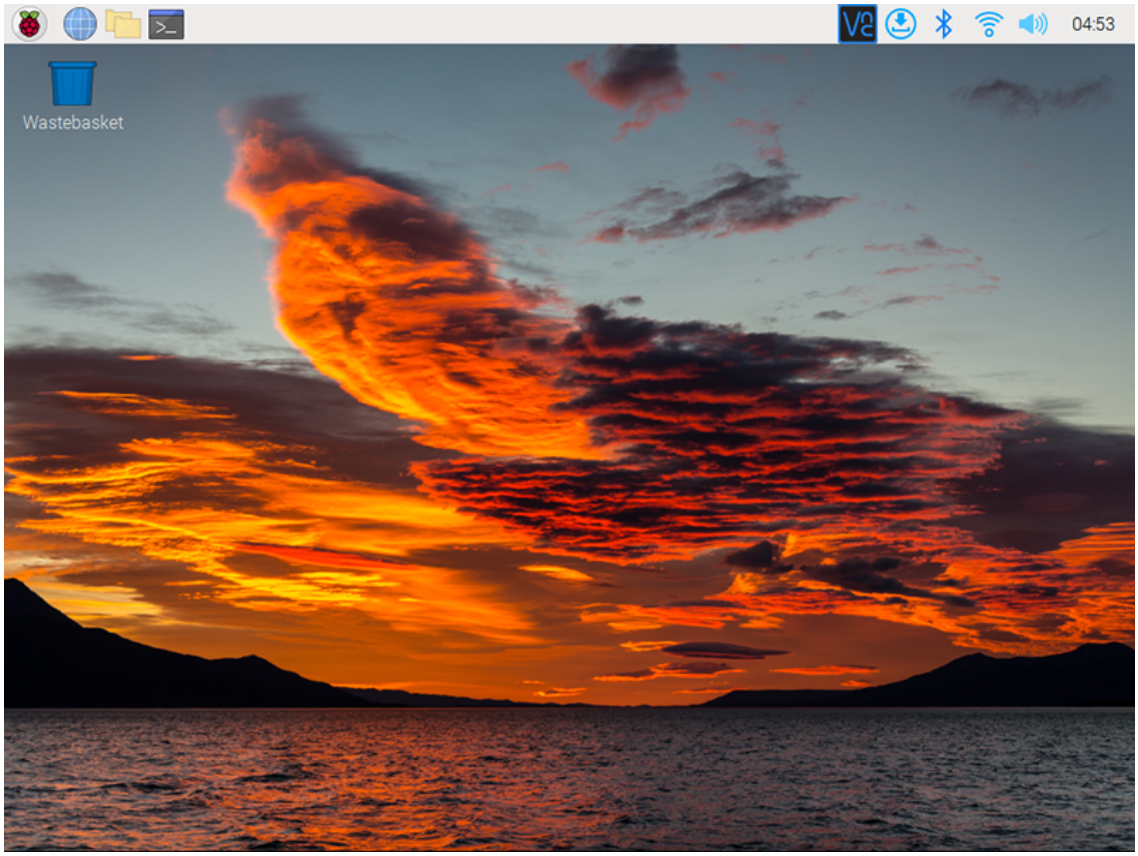
1. 個人用コンピューターに VNC Viewer をダウンロードしてインストールします。
2. インストールが完了したら、開いてホスト名または IP アドレスを入力して Enter キーを押します。



3. Raspberry Pi の名前とパスワードを入力した後、 **OK** をクリックします。

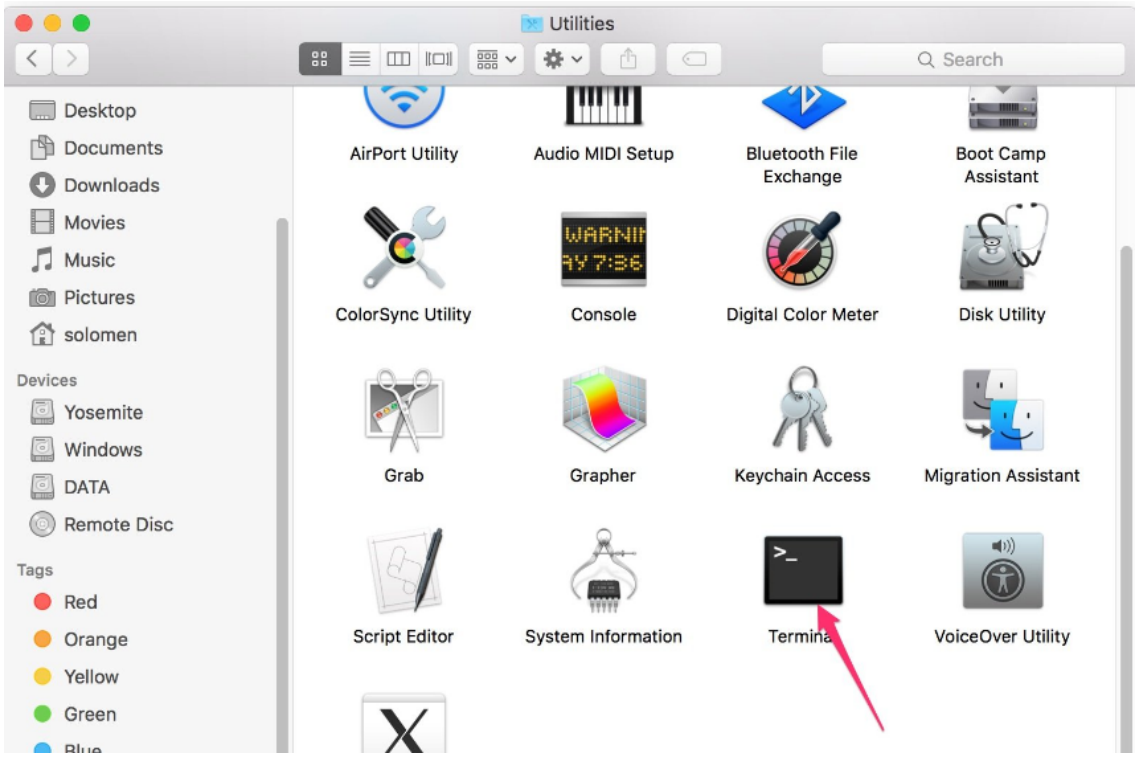


4. これで Raspberry Pi のデスクトップが表示されます。



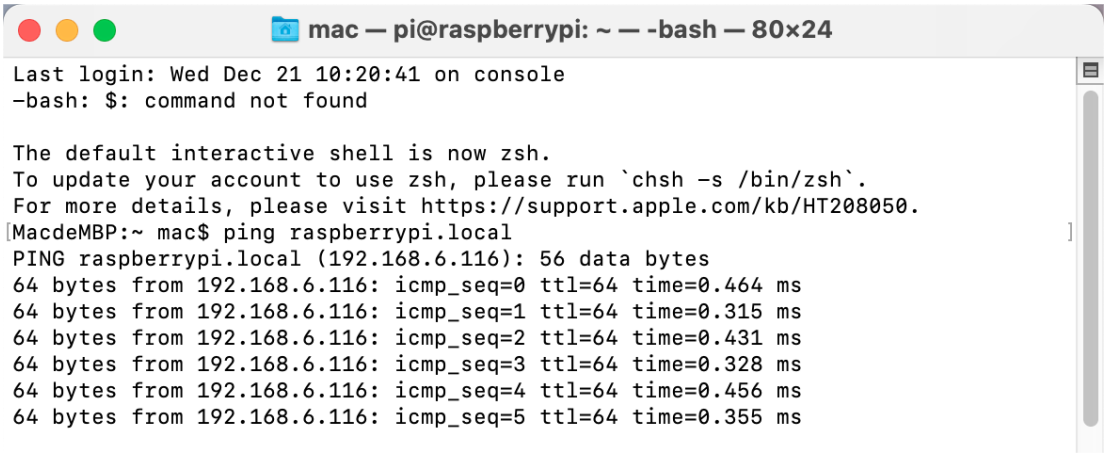
Linux /Unix ユーザー

1. **Applications** -> **Utilities** に移動し、 **Terminal** を探して開きます。



2. Raspberry Pi が同じネットワーク上にあるかを確認するために、 `ping <hostname>.local` と入力します。

```
ping raspberrypi.local
```



上記のように、ネットワークに接続された後、Raspberry Pi の IP アドレスが表示されます。

- ターミナルが `Ping request could not find host pi.local. Please check the name and try again.` と表示された場合、入力したホスト名が正しいか確認してください。
- それでも IP を取得できない場合は、Raspberry Pi のネットワークまたは WiFi 設定を確認してください。

3. `ssh <username>@<hostname>.local` （または `ssh <username>@<IP address>` ）と入力します。

```
ssh pi@raspberrypi.local
```

注釈: The term 'ssh' is not recognized as the name of a cmdlet... というプロンプトが表示された場合、

システムが古く、ssh ツールがプリインストールされていないことを意味します。手動で *Powershell* **を使って** *OpenSSH* **をインストールする** を行う必要があります。

または、 *PuTTY* のようなサードパーティツールを使用することもできます。

---

4. 最初にログインする際にのみ、以下のメッセージが表示されるので、 yes と入力します。

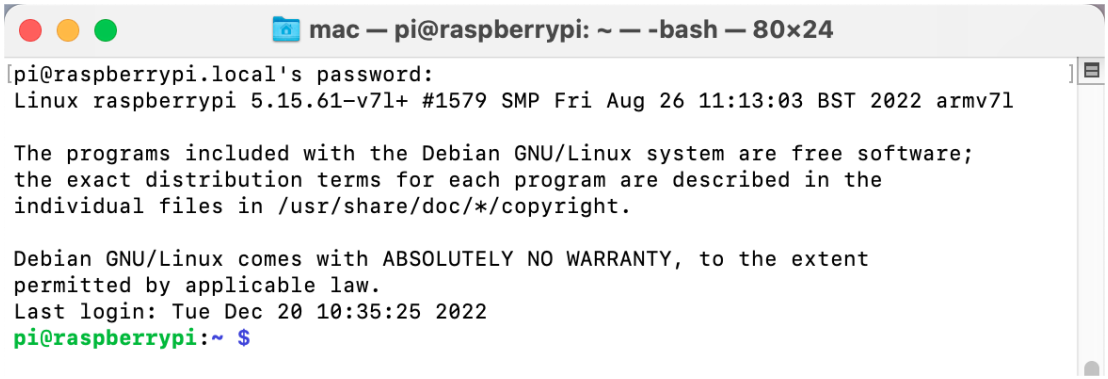
```
The authenticity of host 'raspberrypi.local␣
↪(2400:2410:2101:5800:635b:f0b6:2662:8cba)' can't be established.
ED25519 key fingerprint is SHA256:oo7x3ZSgAo032wD1tE8eW0fFM/kmewIvRwkBys6XRwg.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

5. 以前に設定したパスワードを入力します（私の場合は raspberry です）。

注釈: パスワードを入力するとき、ウィンドウ上に文字が表示されないのは正常です。正しいパスワードを入力してください。

---

6. これで Raspberry Pi に接続され、次のステップに進む準備ができました。



4.1.4 すべてのモジュールをインストールする（重要）

インターネットに接続していることを確認し、システムをアップデートしてください：

```
sudo apt update
sudo apt upgrade
```

注釈: Lite バージョンの OS をインストールする場合は、Python3 関連のパッケージをインストールする必要があります。

```
sudo apt install git python3-pip python3-setuptools python3-smbus
```

---

robot-hat をインストールします。

```
cd ~/
git clone -b v2.0 https://github.com/sunfounder/robot-hat.git
cd robot-hat
sudo python3 setup.py install
```

次に、 vilib モジュールをダウンロードしてインストールします。

```
cd ~/
git clone -b picamera2 https://github.com/sunfounder/vilib.git
cd vilib
sudo python3 install.py
```

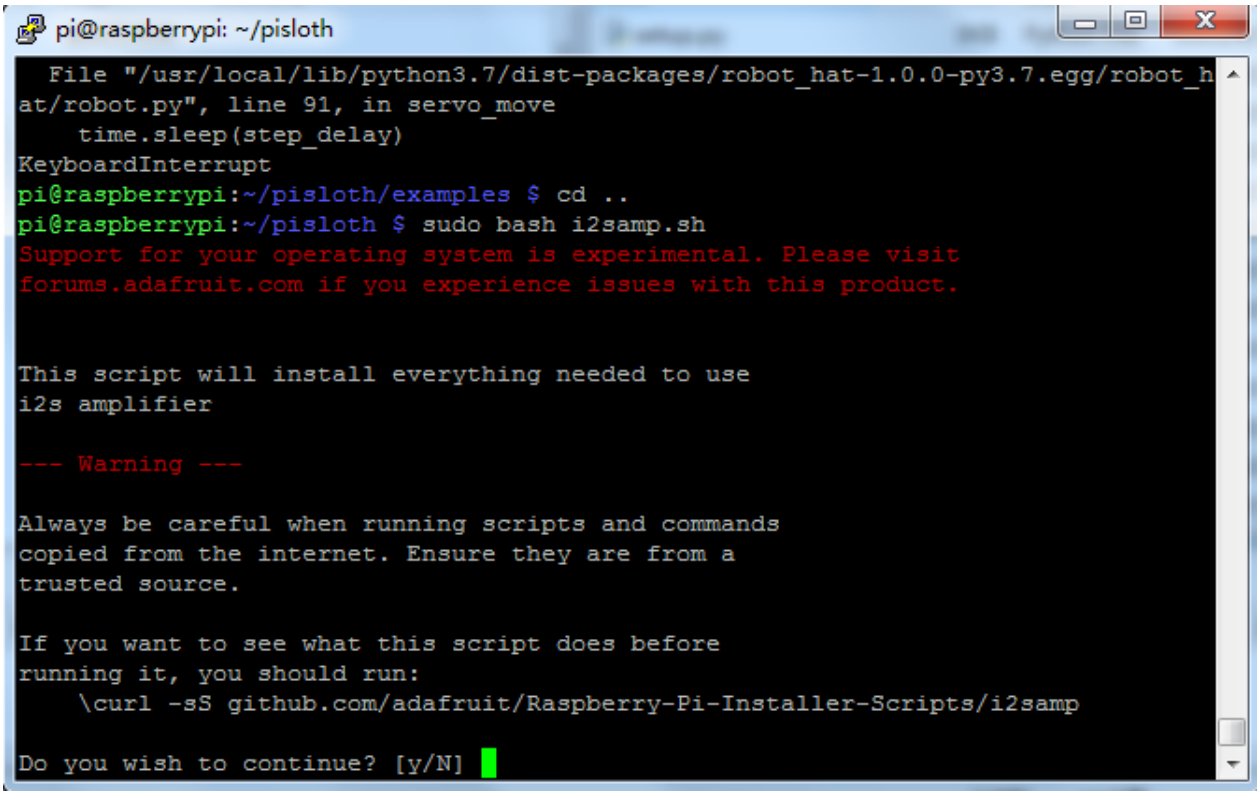
picar-x モジュールをダウンロードしてインストールします。

```
cd ~/
git clone -b v2.0 https://github.com/sunfounder/picar-x.git
cd picar-x
sudo python3 setup.py install
```

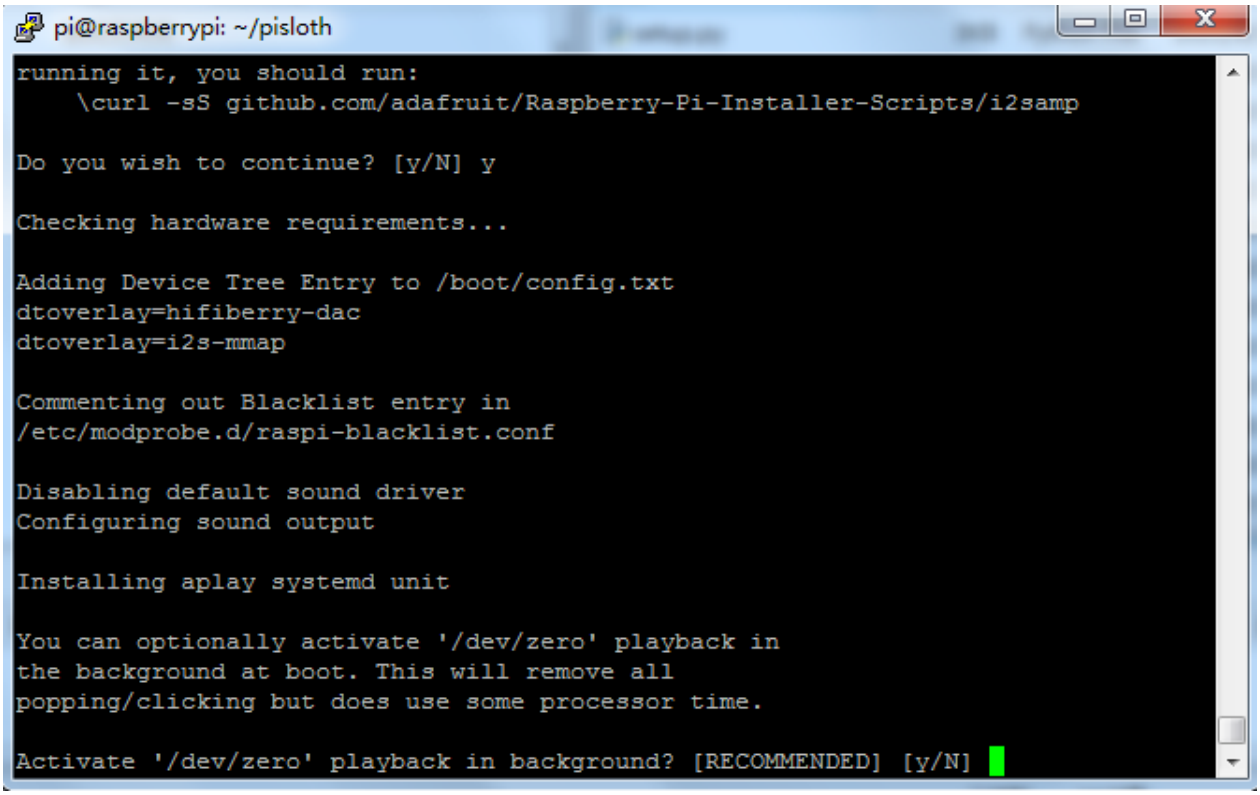
このステップには少し時間がかかりますので、ご patience ください。

最後に、i2s アンプに必要なコンポーネントをインストールするためのスクリプト i2samp.sh を実行する必要があります。そうしないと、picar-x に音が出ません。

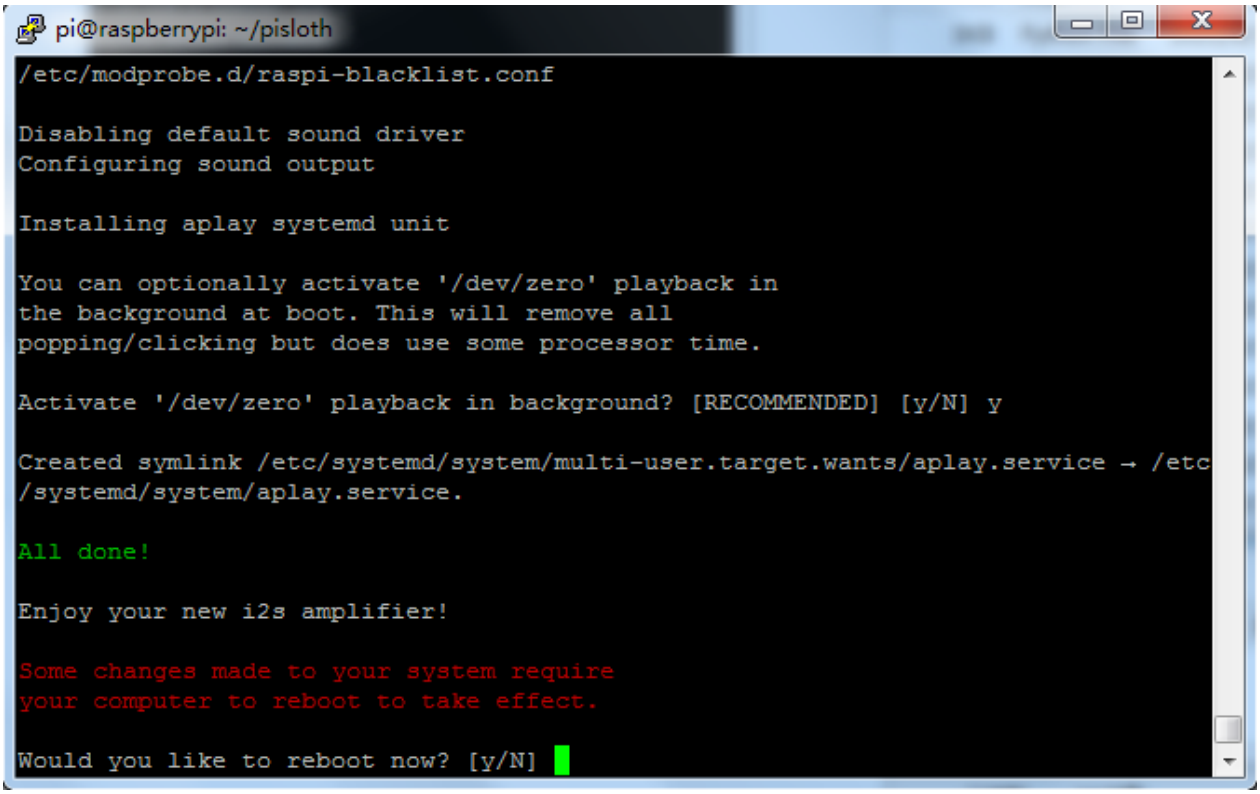
```
cd ~/picar-x
sudo bash i2samp.sh
```



スクリプトを続けて実行するために y と入力し、Enter キーを押します。



バックグラウンドで /dev/zero を実行するために y と入力し、Enter キーを押します。



Picar-X を再起動するために y と入力し、Enter キーを押します。

注釈: 再起動後に音が出ない場合は、i2samp.sh スクリプトを何度か実行する必要があるかもしれません。

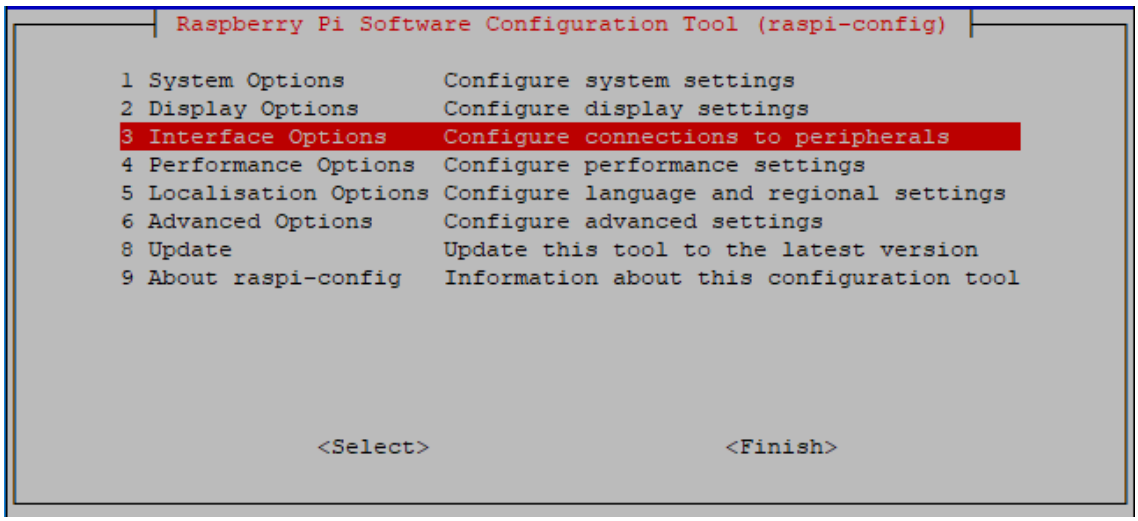
4.1.5 I2C インターフェースを有効にする（重要）

ここでは Raspberry Pi の I2C インターフェースを使用しますが、デフォルトでは無効になっているため、まず有効にする必要があります。

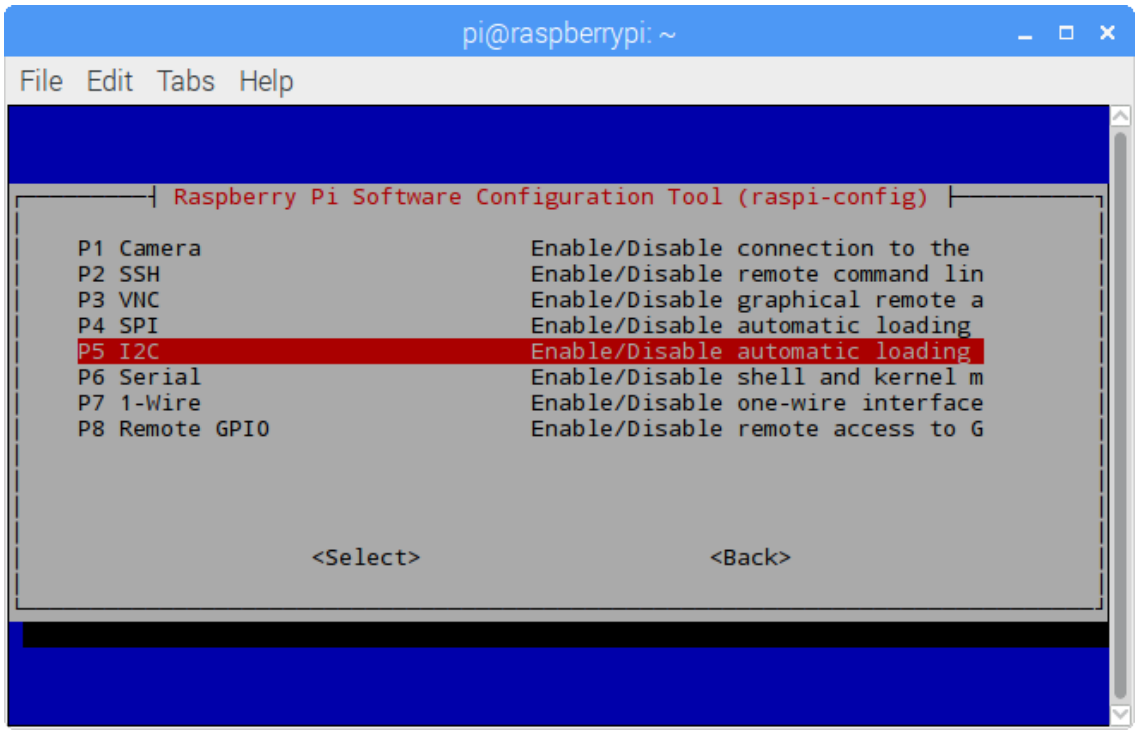
1. 次のコマンドを入力します：

```
sudo raspi-config
```

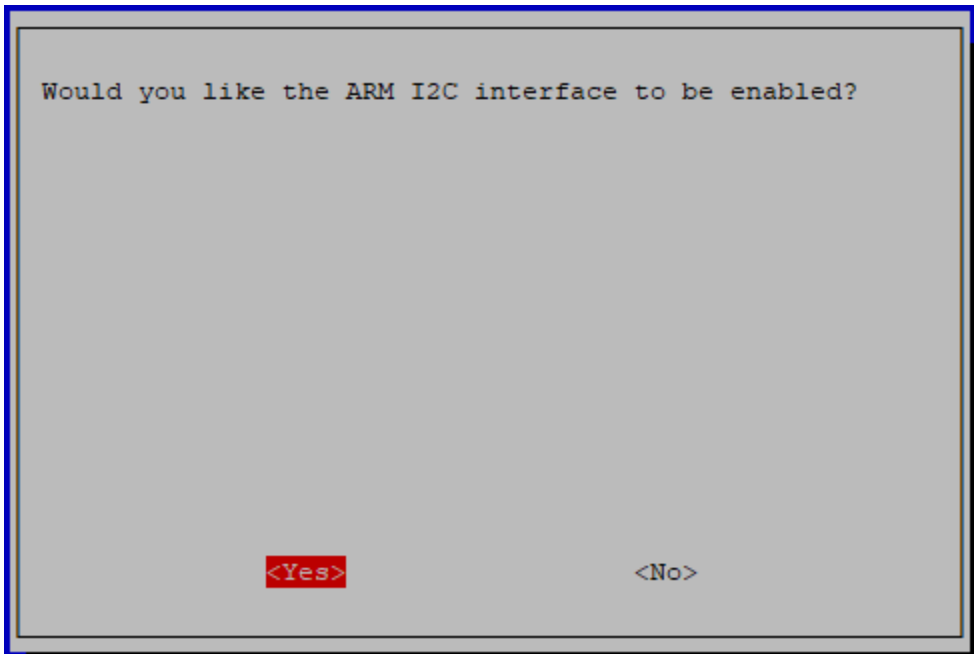
2. キーボードの下矢印キーを押して **Interfacing Options** を選択し、 **Enter** キーを押します。



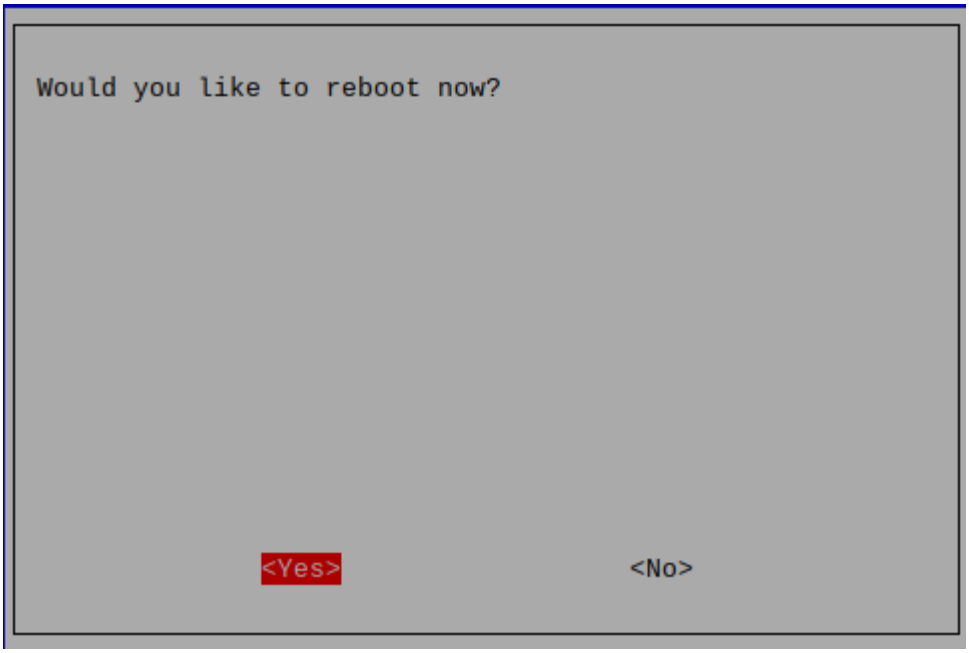
3. 次に **I2C** を選択します。



4. キーボードの矢印キーで **<Yes>** -> **<OK>** を選択して、I2C の設定を完了します。



5. **<Finish>** を選択した後、設定が有効になるために再起動が必要であることを示すポップアップが表示されます。 **<Yes>** を選択します。



4.1.6 サーボ調整（重要）

サーボの角度範囲は-90〜90 度ですが、工場で設定された角度はランダムで、0 ° かもしれませんし、45 ° かもしれません。このような角度で直接組み立てると、ロボットがコードを実行した後に混乱状態になったり、最悪の場合はサーボがブロックして焼損する原因となります。

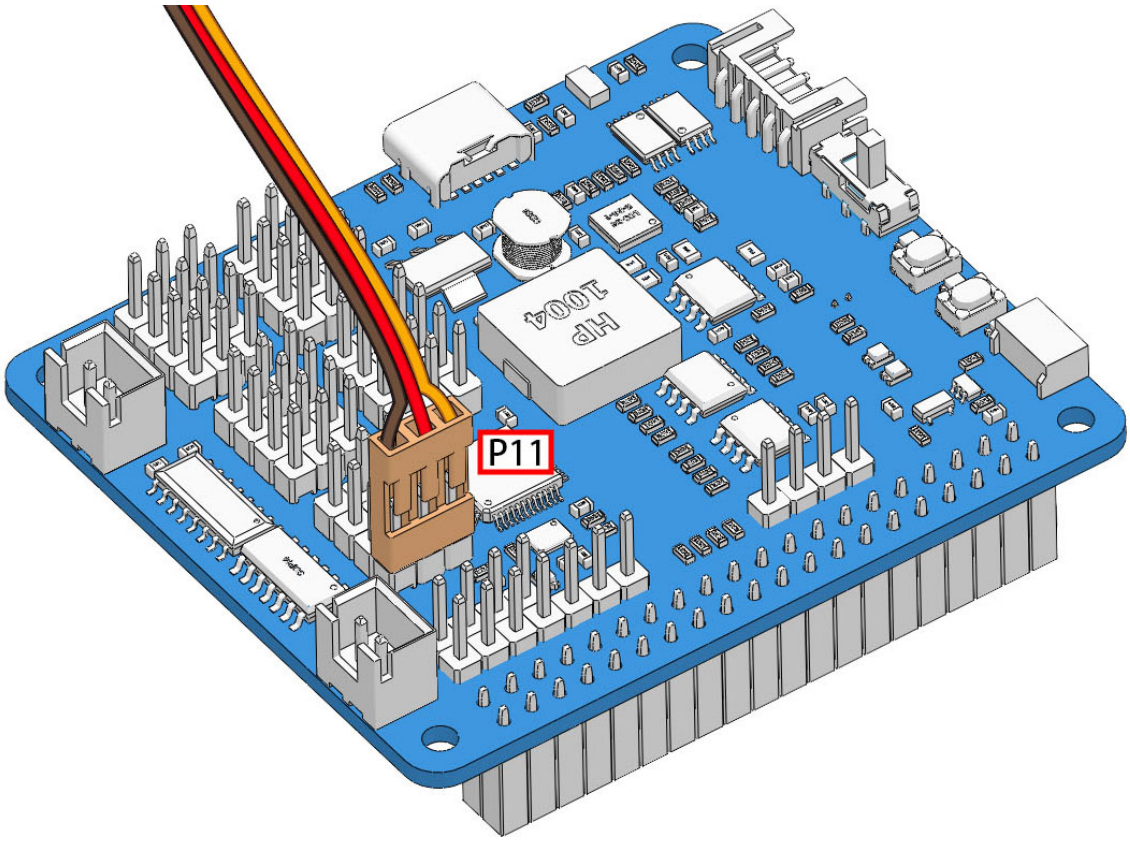
したがって、サーボの角度を 0 ° に設定してから取り付ける必要があります。そうすれば、どちらの方向に回転してもサーボの角度は中央になります。

1. サーボが正しく 0 ° に設定されていることを確認するために、まずサーボアームをサーボシャフトに挿入し、その後ロッカーアームを優しく異なる角度に回転させます。このサーボアームは、サーボが回転していることをはっきりと確認できるようにするためのものです。

2. 次に、 example/ フォルダ内の servo_zeroing.py を実行します。

```
cd ~/picar-x/example
sudo python3 servo_zeroing.py
```

3. 次に、以下のようにサーボケーブルを P11 ポートに接続します。同時にサーボアームが位置に回転するのを見ることができます（これが 0 ° の位置で、ランダムな位置であり、垂直または平行でない場合があります）。



4. 今、サーボアームを取り外し、サーボワイヤーが接続されたままにし、電源を切らないでください。その後、紙の指示に従って組み立てを続けます。

注釈:

- サーボネジで固定する前に、このサーボケーブルを抜かないでください。固定した後に抜くことができます。
- 電源が入っている状態でサーボを回転させないでください。サーボシャフトが正しい角度に挿入されていない場合は、サーボを取り出して再度挿入してください。
- 各サーボを組み立てる前に、P11 にサーボケーブルを接続し、電源を入れてその角度を 0 ° に設定する必要があります。

動画

組み立て動画の 6:25 から 8:48 には、この章の詳しいチュートリアルもあります。動画の指示に直接従うことができます。

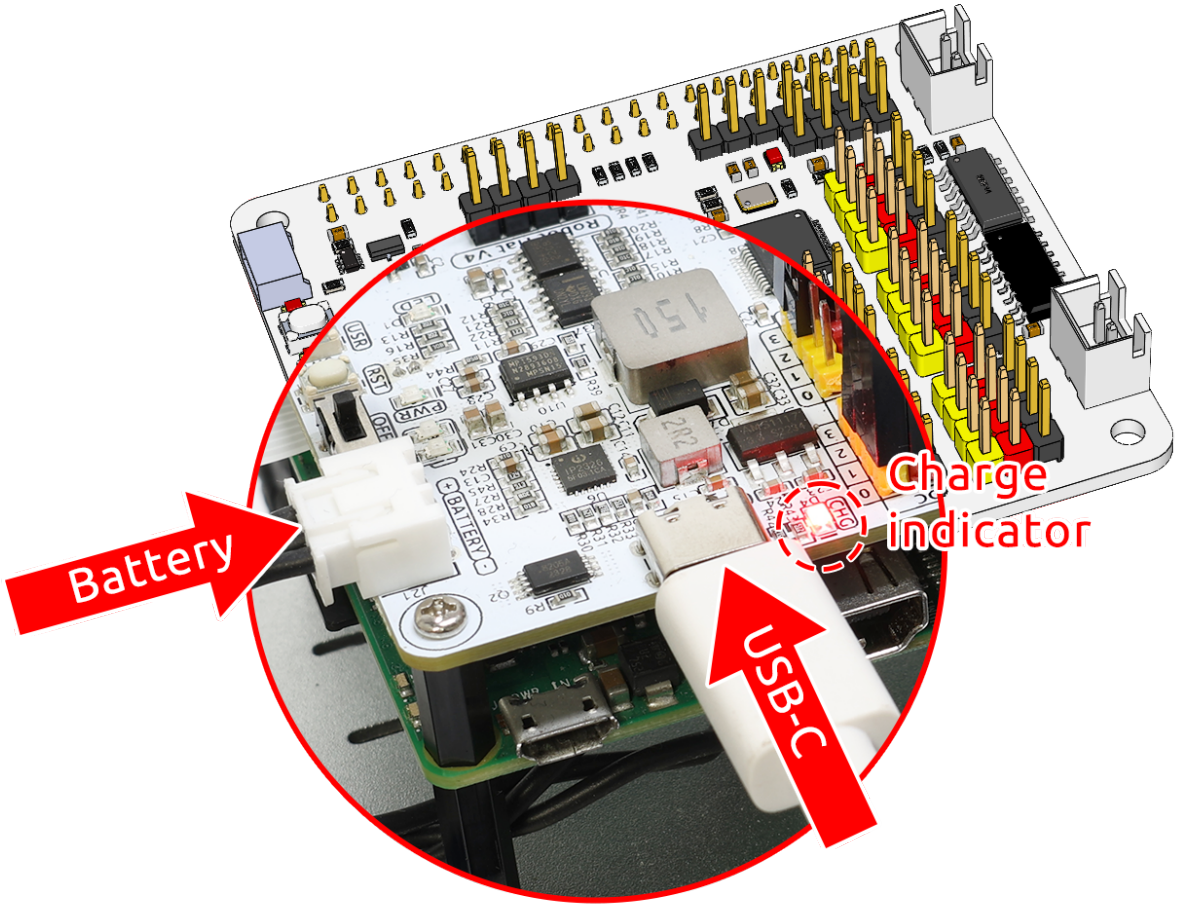
注釈:

- 6:25 から 8:48 のセクションのみをご覧ください。
- 動画の残りの部分は PiCar-X 2.0 についてのもので、お受け取りになった紙の組み立て指示と異なる組み立てステップが含まれている場合があります。正確さのために紙のマニュアルを参照してください。

4.2 起動 & 充電

4.2.1 充電

バッテリーケーブルを挿入してください。次に、バッテリーを充電するために USB-C ケーブルを挿入します。充電器はご自身で用意する必要があります。5V 3A の充電器をお勧めしますが、お使いのスマートフォンの充電器でも問題ありません。

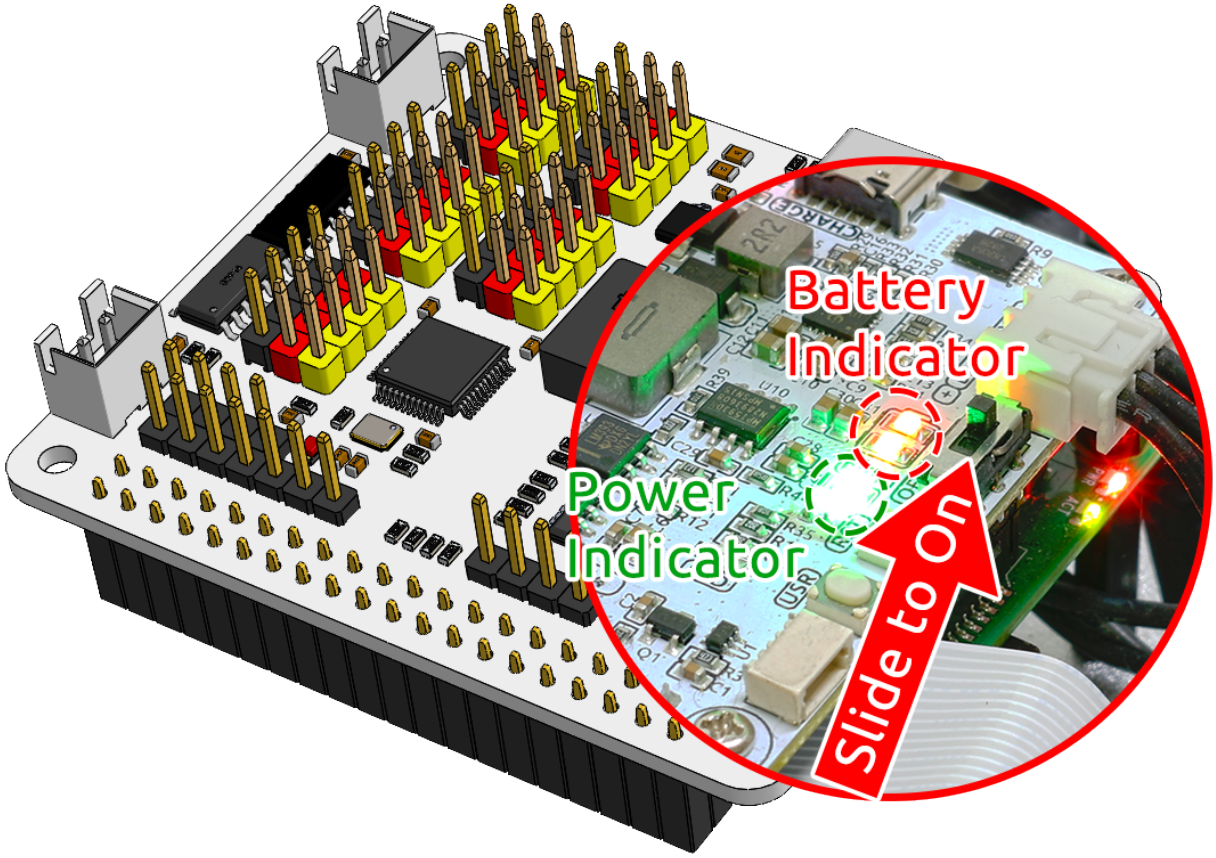


注釈: ロボットハットの Type-C ポートに外部の Type-C 電源を接続すると、すぐにバッテリーの充電が開始され、

赤いインジケーターライトが点灯します。バッテリーが完全に充電されると、赤いライトは自動的に消灯します。

4.2.2 起動

電源スイッチをオンにしてください。電源インジケーターライトとバッテリーレベルインジケーターライトが点灯します。



数秒待ってから、軽いビープ音が聞こえるはずです。これはラズベリーパイが正常に起動したことを示しています。

注釈: バッテリーレベルインジケーターライトが両方ともオフの場合は、バッテリーを充電してください。長時間のプログラミングやデバッグが必要な場合は、USB-C ケーブルを挿入してバッテリーを同時に充電し、ラズベリーパイを稼働させることができます。

4.2.3 18650 バッテリー

- VCC: バッテリーの正極端子です。ここには VCC と GND の 2 つのセットがあり、電流を増やし、抵抗を減らすためです。
- Middle: 2 つのセル間の電圧を均等にし、バッテリーを保護します。
- GND: バッテリーの負極端子です。

これは SunFounder が製作した 2 つの 18650 バッテリーから成るカスタムバッテリーパックで、容量は 2000mAh です。コネクタは XH2.54 3P で、シールドに挿入した後、直接充電できます。

特徴

- バッテリー充電：5V/2A
- バッテリー出力：5V/5A

- バッテリー容量：3.7V 2000mAh x 2
- バッテリー寿命：90 分
- バッテリー充電時間：130 分
- コネクタ：XH2.54 3P

PiCar-X の組み立てが完了したら、以下のプロジェクトを実行してみてください：

4.3 0. PiCar-X の校正

4.3.1 モーターとサーボの校正

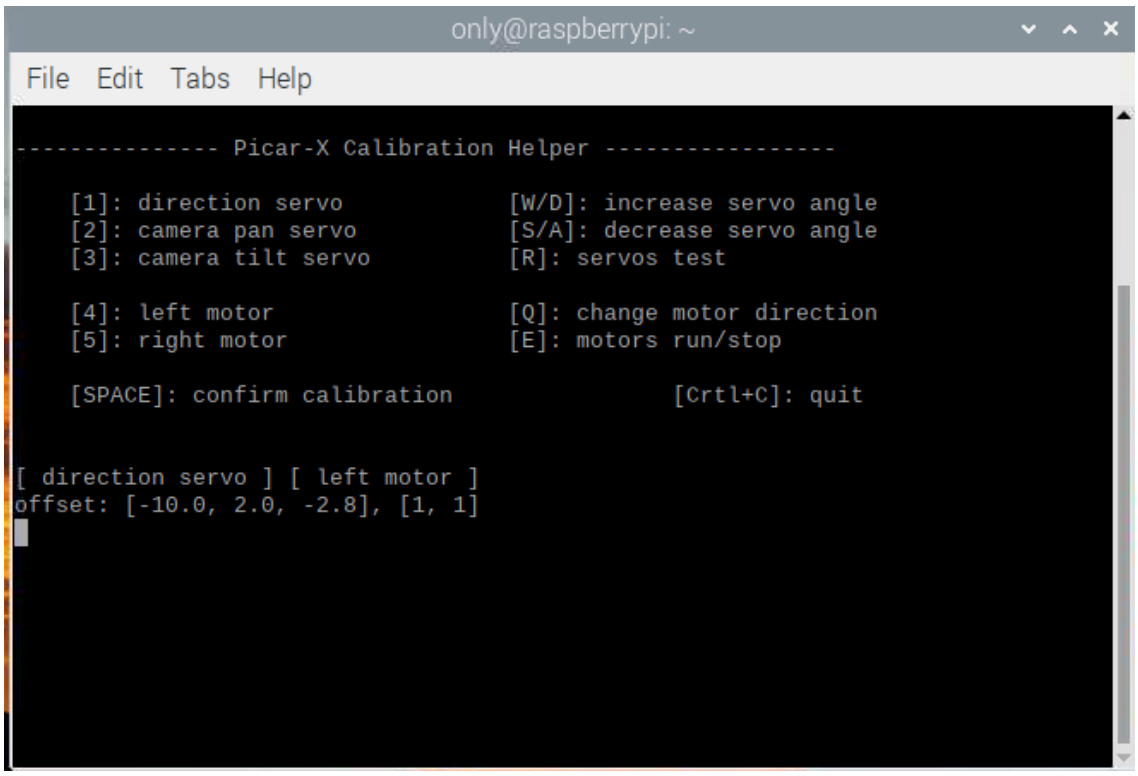
PiCar-X の取り付け中やサーボ自体の限界により、サーボの角度が多少傾くことがあるため、校正が可能です。

もちろん、組み立てが完璧で校正が不要だと思われる場合は、この章をスキップしても構いません。

1. `calibration.py` を実行します。

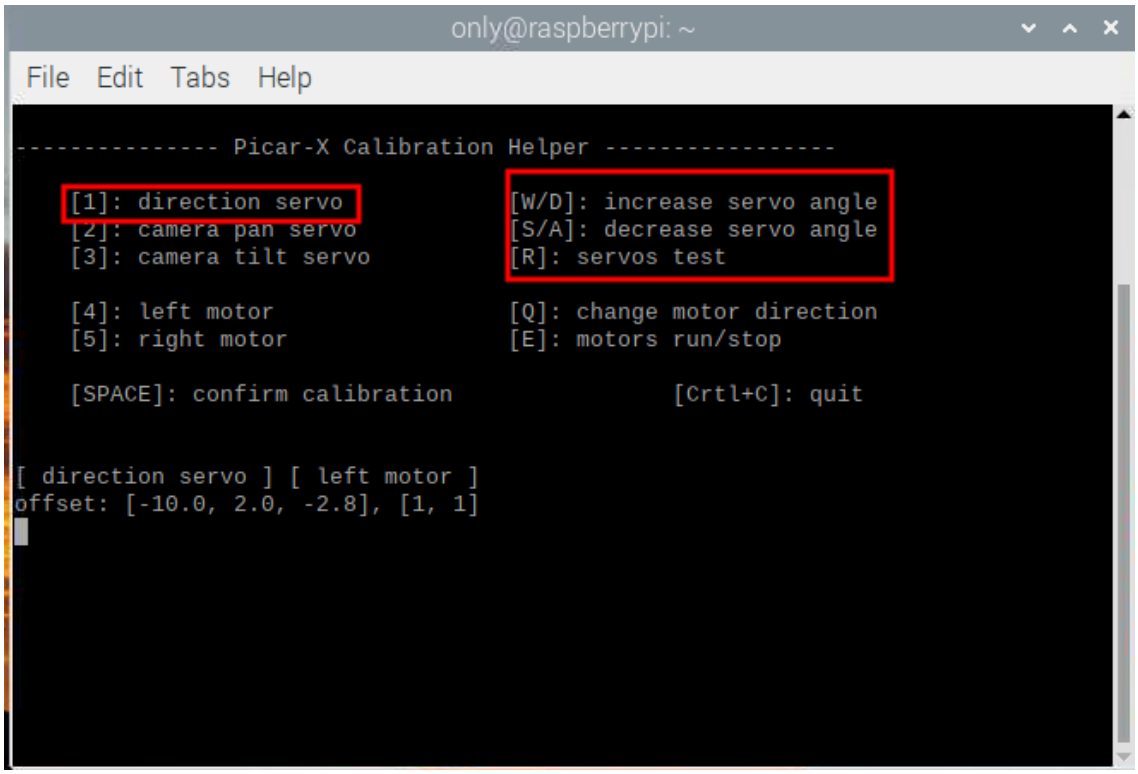
```
cd ~/picar-x/example/calibration
sudo python3 calibration.py
```

2. コードを実行すると、端末に以下のインターフェースが表示されます。

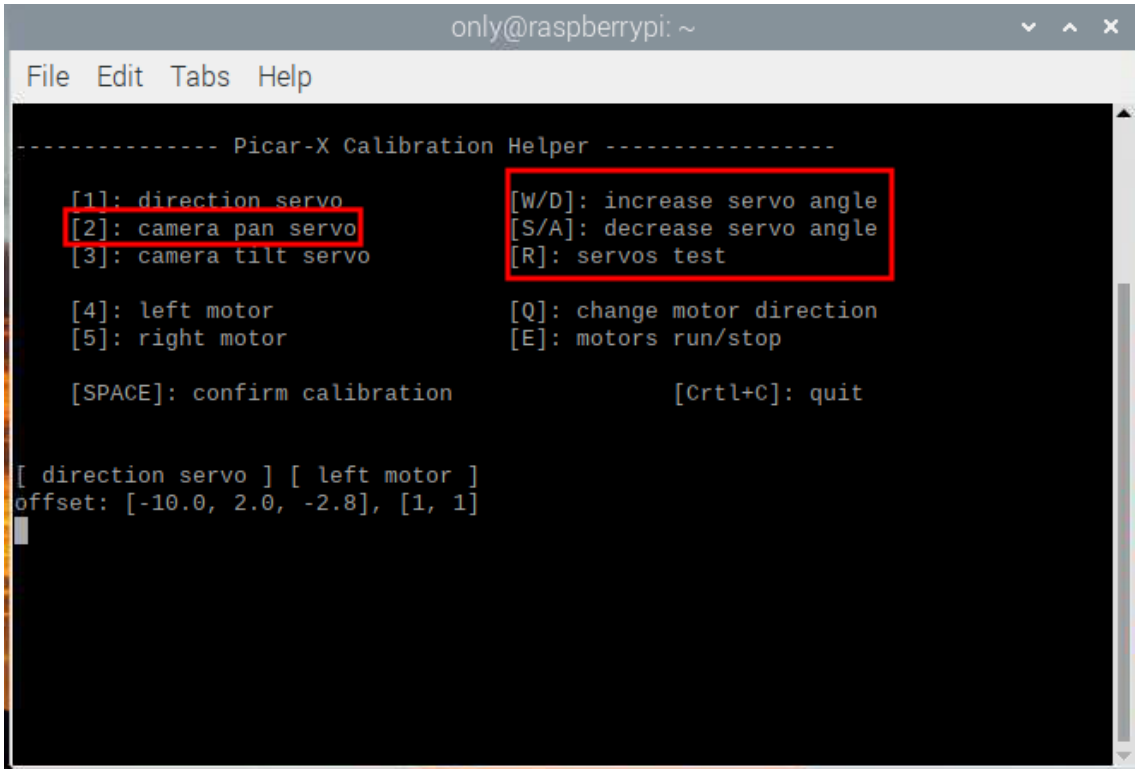


3. R キーは、3 つのサーボが正常に動作しているかど うかをテストするために使用します。 1、 2 、 3 キーでサーボを選択したら、 R キーを押してそのサーボをテストします。

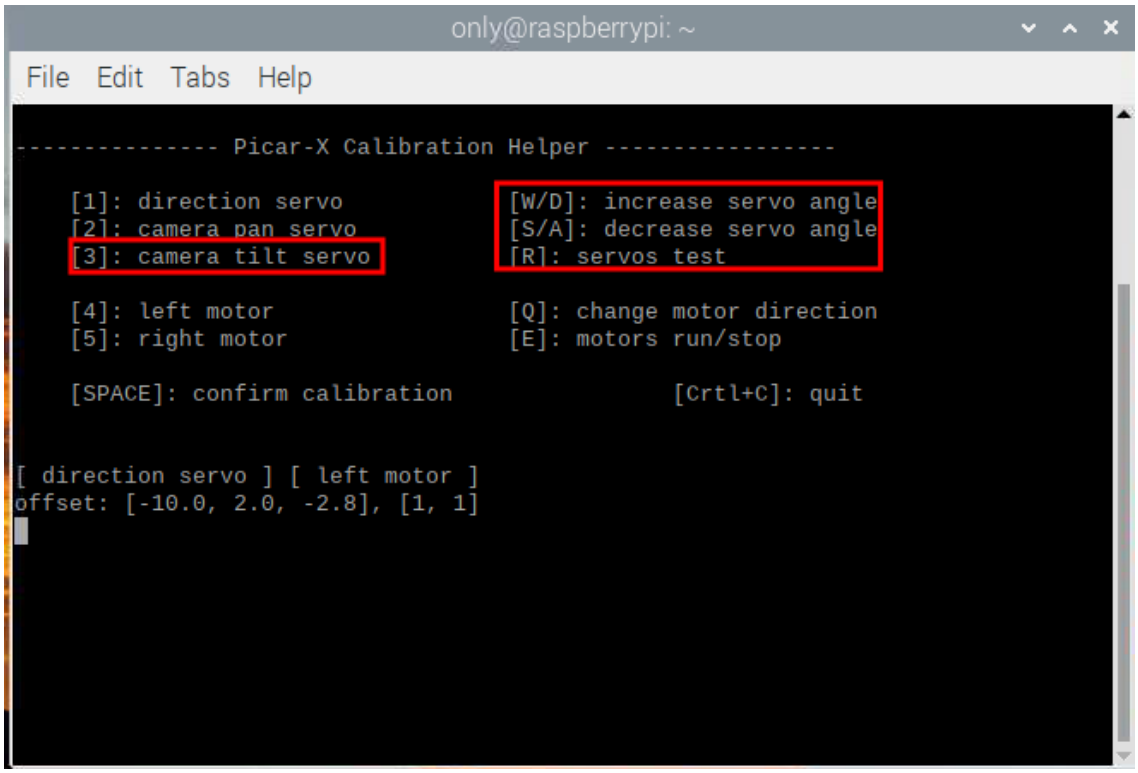
4. 数字キー 1 を押して前輪サーボを選択し、その後 W/S キーを押して前輪が左右に傾かずに可能な限り前を向くようにします。



5. 数字キー 2 を押して **Pan servo** を選択し、その後 W/S キーを押してパン/チルトプラットフォームが真っ直ぐ前を向き、左右に傾かないようにします。

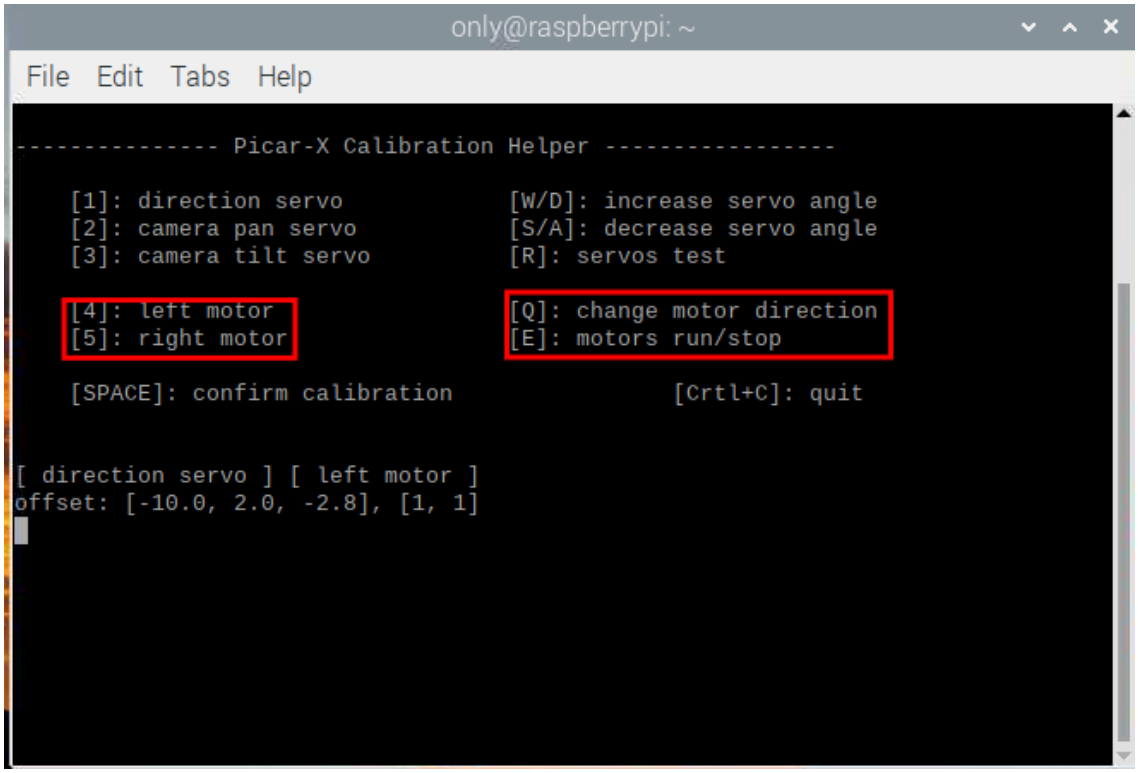


6. 数字キー 3 を押して **tilt servo** を選択し、その後 W/S キーを押してパン/チルトプラットフォームが真っ直ぐ前を向き、上下に傾かないようにします。

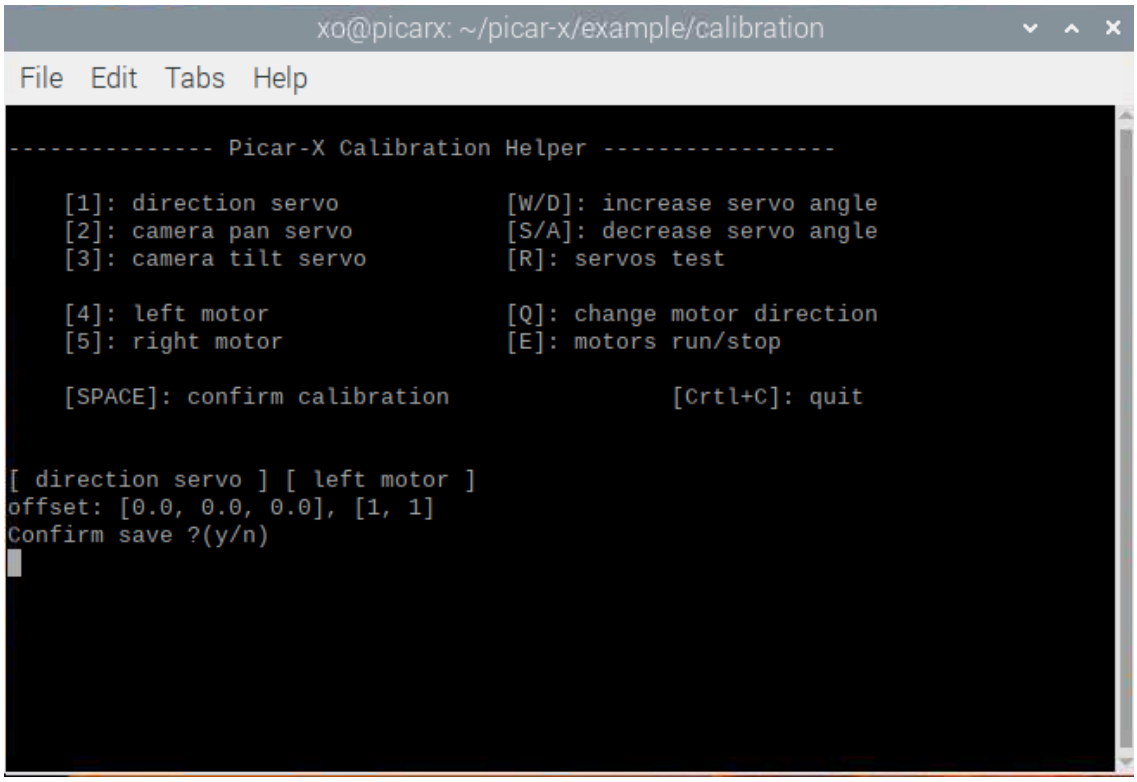


7. 取り付け中にモーターの配線が逆になっている可能性があるため、 E を押して車が正常に前進できるかどう

かをテストします。そうでない場合は、数字キー 4 と 5 を使用して左右のモーターを選択し、その後 Q キーを押して回転方向を校正します。



8. 校正が完了したら、 Spacebar を押して校正パラメータを保存します。確認のために y と入力するプロンプトが表示されたら、 Ctrl+C を押してプログラムを終了し、校正を完了します。



4.3.2 グレースケールモジュールの校正

環境条件や照明状況の違いにより、グレースケールモジュールのプリセットパラメータが最適でない場合があります。このプログラムを通じて設定を微調整し、より良い結果を得ることができます。

1. 明るい色の床に約 15cm の黒い電気テープを敷き、PiCar-X をテープの上に乗せます。このセットアップでは、グレースケールモジュールの中央センサーがテープの真上に、両サイドのセンサーがより明るい表面の上に来るようにします。

2. `grayscale_calibration.py` を実行します。

```
cd ~/picar-x/example/calibration
sudo python3 grayscale_calibration.py
```

3. コードを実行すると、端末に以下のインターフェースが表示されます。

```
<frozen importlib._bootstrap>:228: RuntimeWarning: Your system is neon capable but
 pygame was not built with support for it. The performance of some of your blits c
ould be adversely affected. Consider enabling compile time detection with environm
ent variables like PYGAME_DETECT_AVX2=1 if you are compiling without cross compila
tion.

        Picar-X Grayscale Module Reference
              Calibration Helper

 config_file: /opt/picar-x/picar-x.conf

 press [Q] to start line reference calibration,
 press [E] to start cliff reference calibration

 [SPACE]: confirm calibration          [Crtl+C]: quit

 ----------

current value: [854, 791, 922]
thresholds: [[56, 872], [82, 820], [83, 991]]
line reference: [1000.0, 1000.0, 1000.0]
cliff reference: [500.0, 500.0, 500.0]
```

4. 「Q」キーを押してグレースケール校正を開始します。すると PiCar-X が左右に小さな動きをします。このプロセス中に、3 つのセンサーがそれぞれ少なくとも一度は電気テープを横切るようにします。

5. また、「threshold value」セクションには 3 つのペアの大きく異なる値が表示され、一方で「line reference」にはそれぞれのペアの平均値となる 2 つの中間値が表示されます。

```
<frozen importlib._bootstrap>:228: RuntimeWarning: Your system is neon capable but
 pygame was not built with support for it. The performance of some of your blits c
ould be adversely affected. Consider enabling compile time detection with environm
ent variables like PYGAME_DETECT_AVX2=1 if you are compiling without cross compila
tion.

        Picar-X Grayscale Module Reference
              Calibration Helper

 config_file: /opt/picar-x/picar-x.conf

 press [Q] to start line reference calibration,
 press [E] to start cliff reference calibration

 [SPACE]: confirm calibration          [Crtl+C]: quit

Line reference auto calibration done.
Note that cliff reference values shou be less than line reference values.
current value: [860, 242, 314]
thresholds: [[44, 873], [60, 820], [63, 1112]]
line reference: [511, 517, 639]
cliff reference: [558, 555, 544]
```

6. 次に、PiCar-X を宙に浮かせたり（または崖の端に置いたり）して「E」キーを押します。すると、「cliff reference」の値もそれに応じて更新されます。

```
<frozen importlib._bootstrap>:228: RuntimeWarning: Your system is neon capable but
 pygame was not built with support for it. The performance of some of your blits c
ould be adversely affected. Consider enabling compile time detection with environm
ent variables like PYGAME_DETECT_AVX2=1 if you are compiling without cross compila
tion.

        Picar-X Grayscale Module Reference
              Calibration Helper

 config_file: /opt/picar-x/picar-x.conf

 press [Q] to start line reference calibration,
 press [E] to start cliff reference calibration

 [SPACE]: confirm calibration          [Crtl+C]: quit

Cliff reference auto calibration done.

current value: [4, 4, 4]
thresholds: [[0, 1517], [0, 1467], [0, 1519]]
line reference: [497, 472, 624]
cliff reference: [250, 238, 314]
```

7. すべての値が正確であることを確認したら、「space」キーを押してデータを保存します。その後、Ctrl+C を押してプログラムを終了できます。

4.4 1. PiCar-X を動かす

これは最初のプロジェクトです。PiCar-X の基本的な動きをテストしましょう。

コードの実行

```
cd ~/picar-x/example
sudo python3 1.move.py
```

このコードを実行すると、PiCar-X は前進し、S 字型に曲がり、停止して頭を振ります。

コード

注釈: 以下のコードは **変更/リセット/コピー/実行/停止** が可能です。しかし、それをする前に、picar-x/example のようなソースコードのパスに移動する必要があります。コードを変更した後、直接実行して効果を確認できます。

```
from picarx import Picarx
import time


if __name__ == "__main__":
    try:
        px = Picarx()
        px.forward(30)
        time.sleep(0.5)
        for angle in range(0,35):
            px.set_dir_servo_angle(angle)
            time.sleep(0.01)
        for angle in range(35,-35,-1):
            px.set_dir_servo_angle(angle)
            time.sleep(0.01)
        for angle in range(-35,0):
            px.set_dir_servo_angle(angle)
            time.sleep(0.01)
        px.forward(0)
        time.sleep(1)

        for angle in range(0,35):
            px.set_camera_servo1_angle(angle)
```

(次のページに続く)

(前のページからの続き)

```
            time.sleep(0.01)
        for angle in range(35,-35,-1):
            px.set_camera_servo1_angle(angle)
            time.sleep(0.01)
        for angle in range(-35,0):
            px.set_camera_servo1_angle(angle)
            time.sleep(0.01)
        for angle in range(0,35):
            px.set_camera_servo2_angle(angle)
            time.sleep(0.01)
        for angle in range(35,-35,-1):
            px.set_camera_servo2_angle(angle)
            time.sleep(0.01)
        for angle in range(-35,0):
            px.set_camera_servo2_angle(angle)
            time.sleep(0.01)

    finally:
        px.forward(0)
```

それはどのように機能するのですか？

PiCar-X の基本機能は、 picarx モジュールにあります。これは、ステアリングギアやホイールの制御に使用され、PiCar-X を前進させたり、S 字型に曲がらせたり、頭を振らせたりすることができます。

現在、PiCar-X の基本機能をサポートするライブラリがインポートされています。これらの行は、PiCar-X の動きを伴うすべての例に表示されます。

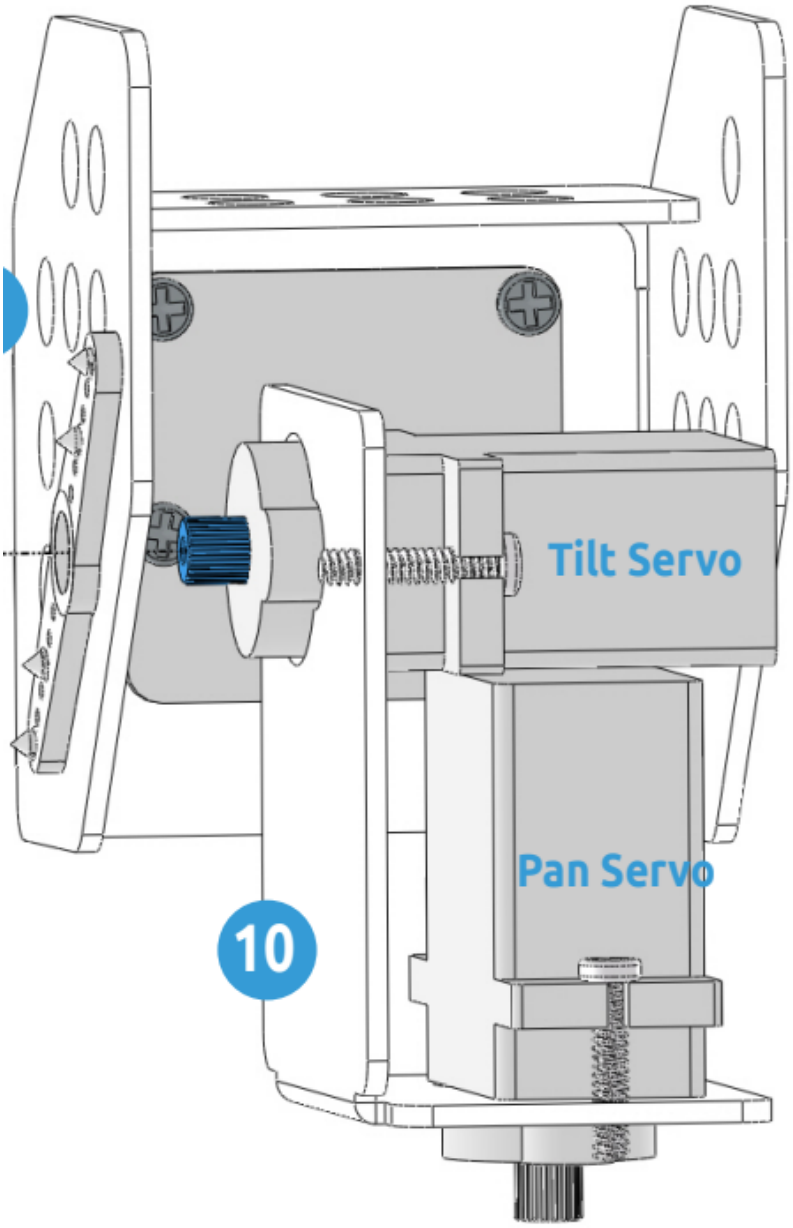
```
from picarx import Picarx
import time
```

次に、 for ループを使用する以下の関数は、PiCar-X を前進させ、方向を変え、カメラのパン/チルトを動かすために使用されます。

```
px.forward(speed)
px.set_dir_servo_angle(angle)
px.set_camera_servo1_angle(angle)
px.set_camera_servo2_angle(angle)
```

- forward()：PiCar-X に指定された speed で前進するよう命令します。
- set_dir_servo_angle：ステアリングサーボを特定の angle に回転させます。

- set_cam_pan_angle：パンサーボを特定の angle に回転させます。
- set_cam_tilt_angle：チルトサーボを特定の angle に回転させます。



4.5 2. キーボード制御

このプロジェクトでは、キーボードを使って PiCar-X をリモートで制御する方法を学びます。PiCar-X を前進、後退、左、右に動かすことができます。

コードの実行

```
cd ~/picar-x/example
sudo python3 2.keyboard_control.py
```

キーボードのキーを押して PiCar-X を制御しましょう！

- w: 前進
- a: 左に曲がる
- s: 後退
- d: 右に曲がる
- i: 頭を上げる
- k: 頭を下げる
- j: 頭を左に向ける
- l: 頭を右に向ける
- ctrl + c: 終了

コード

```
from picarx import Picarx
from time import sleep
import readchar

manual = '''
Press keys on keyboard to control PiCar-X!
    w: Forward
    a: Turn left
    s: Backward
    d: Turn right
    i: Head up
    k: Head down
    j: Turn head left
    l: Turn head right
```

(次のページに続く)

(前のページからの続き)

```
    ctrl + c: Press twice to exit the program
'''

def show_info():
    print("\033[H\033[J",end='')  # clear terminal windows
    print(manual)


if __name__ == "__main__":
    try:
        pan_angle = 0
        tilt_angle = 0
        px = Picarx()
        show_info()
        while True:
            key = readchar.readkey()
            key = key.lower()
            if key in('wsadikjl'):
                if 'w' == key:
                    px.set_dir_servo_angle(0)
                    px.forward(80)
                elif 's' == key:
                    px.set_dir_servo_angle(0)
                    px.backward(80)
                elif 'a' == key:
                    px.set_dir_servo_angle(-35)
                    px.forward(80)
                elif 'd' == key:
                    px.set_dir_servo_angle(35)
                    px.forward(80)
                elif 'i' == key:
                    tilt_angle+=5
                    if tilt_angle>35:
                        tilt_angle=35
                elif 'k' == key:
                    tilt_angle-=5
                    if tilt_angle<-35:
                        tilt_angle=-35
                elif 'l' == key:
```

(次のページに続く)

(前のページからの続き)

```
                    pan_angle+=5
                    if pan_angle>35:
                        pan_angle=35
                elif 'j' == key:
                    pan_angle-=5
                    if pan_angle<-35:
                        pan_angle=-35

                px.set_cam_tilt_angle(tilt_angle)
                px.set_cam_pan_angle(pan_angle)
                show_info()
                sleep(0.5)
                px.forward(0)

            elif key == readchar.key.CTRL_C:
                print("\n Quit")
                break

    finally:
        px.set_cam_tilt_angle(0)
        px.set_cam_pan_angle(0)
        px.set_dir_servo_angle(0)
        px.stop()
        sleep(.2)
```

どのように動作するのか？

PiCar-X は、読み取ったキーボードの文字に基づいて適切なアクションを行うべきです。`lower()` 関数は大文字を小文字に変換するため、文字の大文字・小文字に関わらず有効です。

```
while True:
    key = readchar.readkey()
    key = key.lower()
    if key in('wsadikjl'):
        if 'w' == key:
            pass
        elif 's' == key:
            pass
        elif 'a' == key:
            pass
```

(次のページに続く)

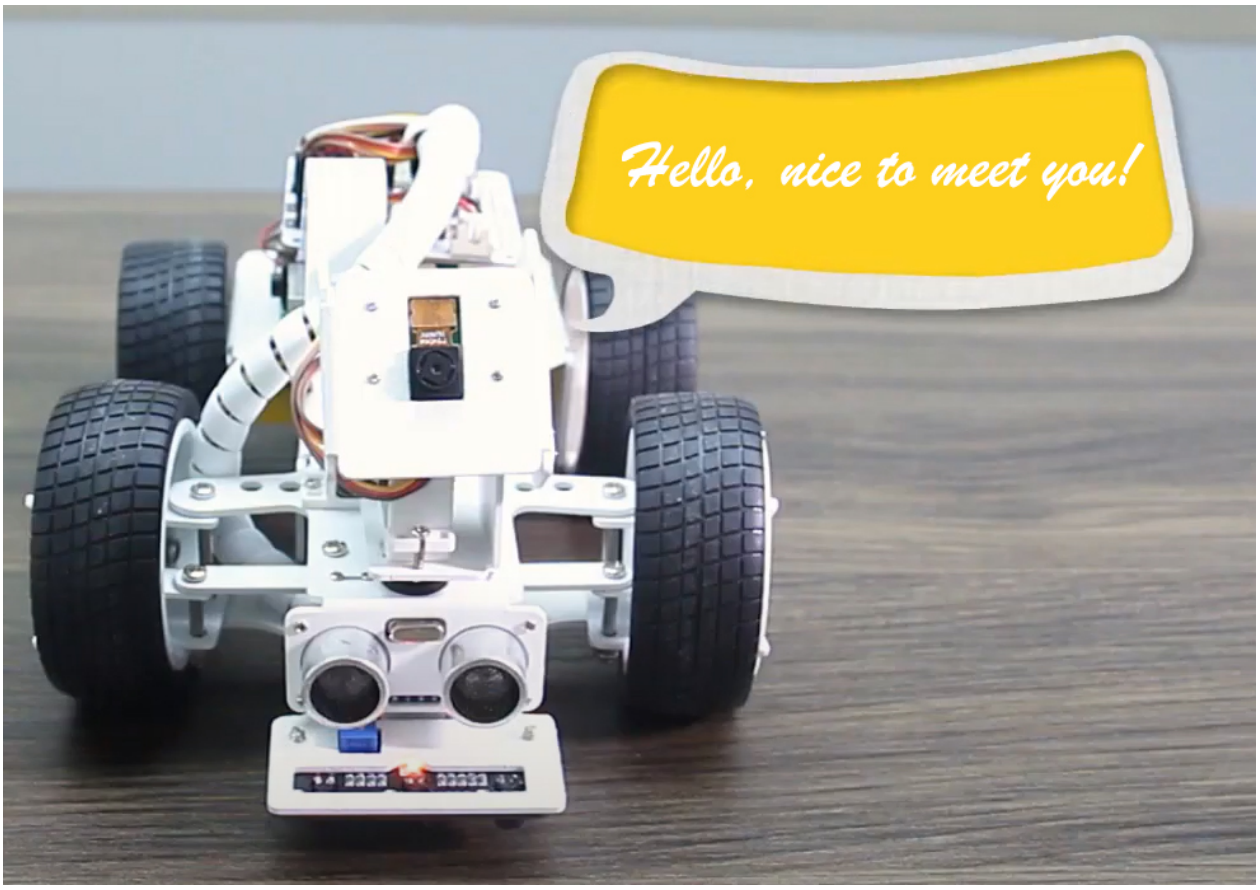
(前のページからの続き)

```
        elif 'd' == key:
            pass
        elif 'i' == key:
            pass
        elif 'k' == key:
            pass
        elif 'l' == key:
            pass
        elif 'j' == key:
            pass

    elif key == readchar.key.CTRL_C:
        print("\n Quit")
        break
```

4.6 3. テキストから音声へ & 効果音

この例では、PiCar-X（正確には Robot HAT）の音声効果を使用します。これは音楽、サウンド、テキストから音声への 3 つの部分から構成されています。

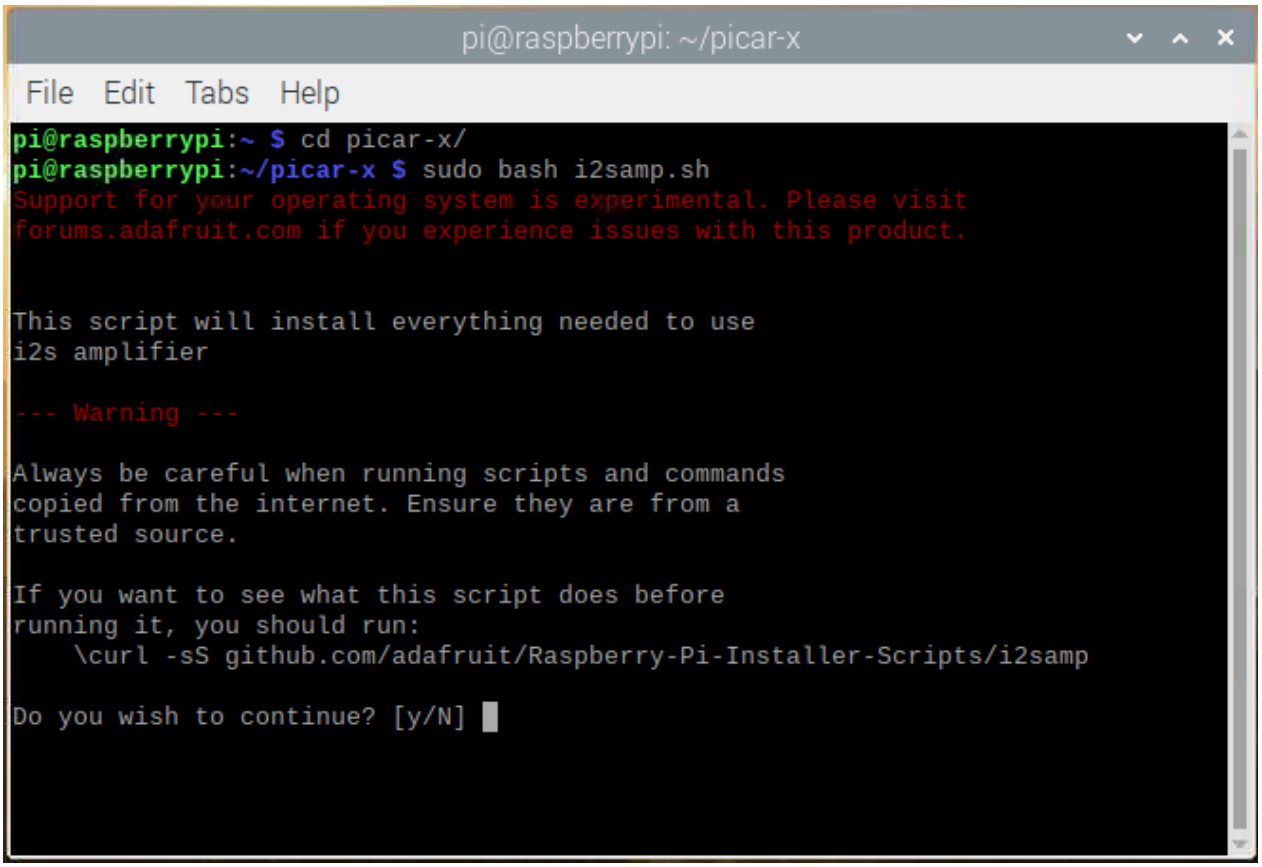


i2samp のインストール

テキストから音声への変換（TTS）と効果音の機能を使用する前に、まずスピーカーをアクティブにして、音を出せるようにしましょう。

picar-x フォルダ内で `i2samp.sh` を実行し、このスクリプトは i2s アンプを使用するために必要なものをすべてインストールします。

```
cd ~/picar-x
sudo bash i2samp.sh
```



いくつかの確認プロンプトが表示されますので、すべてのプロンプトに対して **Y** と回答してください。Raspberry Pi システムに変更が加えられた後、これらの変更が有効になるためにコンピュータを再起動する必要があります。

再起動後、再び `i2samp.sh` スクリプトを実行してアンプをテストします。スピーカーから音が正常に鳴る場合は、設定が完了しています。

コードの実行

```
cd ~/picar-x/example
sudo python3 3.tts_example.py
```

コードを実行した後、ターミナルに表示されたプロンプトに従って操作してください。

機能を呼び出すためにキーを入力してください！

- space: 効果音を再生（クラクション）
- c: スレッドで効果音を再生
- t: テキストを話す（ハローと言う）
- q: 音楽の再生/停止

コード

```
from time import sleep
from robot_hat import Music,TTS
import readchar

music = Music()
tts = TTS()

manual = '''
Input key to call the function!
    space: Play sound effect (Car horn)
    c: Play sound effect with threads
    t: Text to speak
    q: Play/Stop Music
'''

def main():
    print(manual)

    flag_bgm = False
    music.music_set_volume(20)
    tts.lang("en-US")


    while True:
        key = readchar.readkey()
        key = key.lower()
        if key == "q":
            flag_bgm = not flag_bgm
            if flag_bgm is True:
                music.music_play('../musics/slow-trail-Ahjay_Stelino.mp3')
            else:
                music.music_stop()

        elif key == readchar.key.SPACE:
            music.sound_play('../sounds/car-double-horn.wav')
            sleep(0.05)

        elif key == "c":
            music.sound_play_threading('../sounds/car-double-horn.wav')
```

(次のページに続く)

(前のページからの続き)

```
            sleep(0.05)

        elif key == "t":
            words = "Hello"
            tts.say(words)

if __name__ == "__main__":
    main()
```

どのように動作するのか？

背景音楽に関連する機能には以下のものがあります：

- `music = Music()`：オブジェクトを宣言。
- `music.music_set_volume(20)`：音量を設定します。範囲は 0〜100 です。
- `music.music_play('../musics/slow-trail-Ahjay_Stelino.mp3')`：音楽ファイルを再生します。ここでは `../musics` パス下の **slow-trail-Ahjay_Stelino.mp3** ファイルです。
- `music.music_stop()`：背景音楽の再生を停止します。

注釈: `musics` や `sounds` フォルダに異なる効果音や音楽を *Filezilla* **ソフトウェア** を通じて追加することができます。

効果音に関連する機能には以下のものがあります：

- `music = Music()`
- `music.sound_play('../sounds/car-double-horn.wav')`：効果音のファイルを再生します。
- `muisc.sound_play_threading('../sounds/car-double-horn.wav')`：メインスレッドを中断せずに新しいスレッドモードで効果音のファイルを再生します。

テキストから音声への機能は eSpeak ソフトウェアを使用して実装されています。

robot_hat の TTS モジュールをインポートし、テキストを音声に変換する機能をカプセル化します。

テキストから音声への関連機能には以下のものがあります：

- `tts = TTS()`
- `tts.say(words)`：テキストのオーディオ。
- `tts.lang("en-US")`：言語を設定します。

注釈: `lang("")` のパラメータに以下の文字を設定することで言語を設定します。

表 1　Language

zh-CN	Mandarin (Chinese)
en-US	English-United States
en-GB	English-United Kingdom
de-DE	Germany-Deutsch
es-ES	España-Español
fr-FR	France-Le français
it-IT	Italia-lingua italiana

4.7 4. 障害物回避

このプロジェクトでは、PiCar-X が前進しながら前方の障害物を検出し、障害物が近すぎる場合には前進の方向を変えます。

コードの実行

```
cd ~/picar-x/example
sudo python3 4.avoiding_obstacles.py
```

コードを実行すると、PiCar-X は前進します。

前方の障害物の距離が 20cm 以下であると検出すると、後退します。

20cm から 40cm の範囲内に障害物がある場合は、左に曲がります。

左に曲がった後の方向に障害物がないか、障害物の距離が 25cm 以上である場合は、引き続き前進します。

コード

注釈: 以下のコードを **変更/リセット/コピー/実行/停止** することができます。しかし、それをする前に、`picar-x/example` のようなソースコードのパスに移動する必要があります。コードを変更した後、直接実行して効果を確認できます。

```
from picarx import Picarx
import time

```

(次のページに続く)

(前のページからの続き)

```
POWER = 50
SafeDistance = 40   # > 40 safe
DangerDistance = 20 # > 20 && < 40 turn around,
                    # < 20 backward

def main():
    try:
        px = Picarx()
        # px = Picarx(ultrasonic_pins=['D2','D3']) # tring, echo

        while True:
            distance = round(px.ultrasonic.read(), 2)
            print("distance: ",distance)
            if distance >= SafeDistance:
                px.set_dir_servo_angle(0)
                px.forward(POWER)
            elif distance >= DangerDistance:
                px.set_dir_servo_angle(30)
                px.forward(POWER)
                time.sleep(0.1)
            else:
                px.set_dir_servo_angle(-30)
                px.backward(POWER)
                time.sleep(0.5)

    finally:
        px.forward(0)


if __name__ == "__main__":
    main()
```

どのように動作するのか？

- Picarx モジュールのインポートと定数の初期化:

  このコードのセクションでは、Picarx ロボットを制御するために不可欠な `picarx` モジュールから `Picarx` クラスをインポートします。後でスクリプト内で距離測定に基づいてロボットの動きを制御するために使用される `POWER`、 `SafeDistance`、 `DangerDistance` などの定数が定義されています。

```
from picarx import Picarx
import time

POWER = 50
SafeDistance = 40 # > 40 安全
DangerDistance = 20 # > 20 && < 40 旋回
# < 20 後退
```

- メイン関数の定義と超音波センサーの読み取り:

  main 関数は、Picarx ロボットが制御される場所です。 Picarx のインスタンスが作成され、ロボットの機能が活性化します。コードは無限ループに入り、超音波センサーからの距離を常に読み取ります。この距離はロボットの動きを決定するために使用されます。

```
def main():
try:
    px = Picarx()

    while True:
        distance = round(px.ultrasonic.read(), 2)
        # [残りのロジック]
```

- 距離に基づく動きのロジック:

  ロボットの動きは、超音波センサーから読み取った distance に基づいて制御されます。 distance が SafeDistance より大きい場合、ロボットは前進します。距離が DangerDistance と SafeDistance の間であれば、わずかに旋回して前進します。もし distance が DangerDistance 未満であれば、ロボットは逆方向に旋回しながら後退します。

```
if distance >= SafeDistance:
    px.set_dir_servo_angle(0)
    px.forward(POWER)
elif distance >= DangerDistance:
    px.set_dir_servo_angle(30)
    px.forward(POWER)
    time.sleep(0.1)
else:
    px.set_dir_servo_angle(-30)
    px.backward(POWER)
    time.sleep(0.5)
```

- 'finally' ブロックでの安全性とクリーンアップ:

try...finally ブロックは、中断またはエラーが発生した場合にロボットの動きを停止させることで安全性を確保します。これは、ロボットの制御不能な振る舞いを防ぐために重要な部分です。

```
try:
# [制御ロジック]
finally:
    px.forward(0)
```

- 実行エントリーポイント:

  標準的な Python エントリーポイント `if __name__ == "__main__":` が使用され、スクリプトがスタンドアロンプログラムとして実行されたときにメイン関数を実行します。

```
if __name__ == "main":
    main()
```

要約すると、このスクリプトは Picarx モジュールを使用してロボットを制御し、超音波センサーを利用して距離を測定します。ロボットの動きはこれらの測定値に基づいて適応され、finally ブロック内の安全メカニズムを通じて慎重な制御と安全な操作を保証します。

4.8 5. ライン追跡

このプロジェクトではグレースケールモジュールを使用して、PiCar-X を線に沿って前進させます。できるだけまっすぐで、あまり曲がっていない暗色のテープを使って線を作ります。PiCar-X が脱線した場合は、いくつかの実験が必要になるかもしれません。

コードの実行

```
cd ~/picar-x/example
sudo python3 5.minecart_plus.py
```

コードを実行すると、PiCar-X は線に沿って前進します。

コード

注釈: 以下のコードは **変更/リセット/コピー/実行/停止** することができます。しかし、それをする前に、`picar-x/example` のようなソースコードのパスに移動する必要があります。コードを変更した後、直接実行して効果を確認できます。

```
from picarx import Picarx
from time import sleep

px = Picarx()
# px = Picarx(grayscale_pins=['A0', 'A1', 'A2'])

# Please run ./calibration/grayscale_calibration.py to Auto calibrate grayscale values
# or manual modify reference value by follow code
# px.set_line_reference([1400, 1400, 1400])

current_state = None
px_power = 10
offset = 20
last_state = "stop"

def outHandle():
    global last_state, current_state
    if last_state == 'left':
        px.set_dir_servo_angle(-30)
        px.backward(10)
    elif last_state == 'right':
        px.set_dir_servo_angle(30)
        px.backward(10)
    while True:
        gm_val_list = px.get_grayscale_data()
        gm_state = get_status(gm_val_list)
        print("outHandle gm_val_list: %s, %s"%(gm_val_list, gm_state))
        currentSta = gm_state
        if currentSta != last_state:
            break
    sleep(0.001)

def get_status(val_list):
    _state = px.get_line_status(val_list)  # [bool, bool, bool], 0 means line, 1 means␣
↪background
    if _state == [0, 0, 0]:
        return 'stop'
    elif _state[1] == 1:
        return 'forward'
```

(次のページに続く)

(前のページからの続き)

```
    elif _state[0] == 1:
        return 'right'
    elif _state[2] == 1:
        return 'left'

if __name__=='__main__':
    try:
        while True:
            gm_val_list = px.get_grayscale_data()
            gm_state = get_status(gm_val_list)
            print("gm_val_list: %s, %s"%(gm_val_list, gm_state))

            if gm_state != "stop":
                last_state = gm_state

            if gm_state == 'forward':
                px.set_dir_servo_angle(0)
                px.forward(px_power)
            elif gm_state == 'left':
                px.set_dir_servo_angle(offset)
                px.forward(px_power)
            elif gm_state == 'right':
                px.set_dir_servo_angle(-offset)
                px.forward(px_power)
            else:
                outHandle()
    finally:
        px.stop()
        print("stop and exit")
        sleep(0.1)
```

どのように動作するのか？

この Python スクリプトは、グレースケールセンサーを使用して Picarx ロボットカーをナビゲーションします。主なコンポーネントは以下の通りです：

- インポートと初期化：

  このスクリプトは、ロボットカーを制御するための `Picarx` クラスと、遅延を追加するための time モジュールの `sleep` 関数をインポートします。

Picarx のインスタンスが作成され、特定のグレースケールセンサーピンでの代替初期化を示すコメント付きの行があります。

```
from picarx import Picarx
from time import sleep

px = Picarx()
```

- 設定とグローバル変数：

  current_state、 px_power、 offset、 last_state は、車の動きを追跡および制御するために使用されるグローバル変数です。 px_power はモーターのパワーを設定し、 offset はステアリング角度を調整するために使用されます。

```
current_state = None
px_power = 10
offset = 20
last_state = "stop"
```

- outHandle 関数：

  この関数は、車が「ラインアウト」のシナリオを処理する必要がある場合に呼び出されます。

  それは last_state に基づいて車の方向を調整し、新しい状態を決定するためにグレースケールセンサーの値をチェックします。

```
def outHandle():
    global last_state, current_state
    if last_state == 'left':
        px.set_dir_servo_angle(-30)
        px.backward(10)
    elif last_state == 'right':
        px.set_dir_servo_angle(30)
        px.backward(10)
    while True:
        gm_val_list = px.get_grayscale_data()
        gm_state = get_status(gm_val_list)
        print("outHandle gm_val_list: %s, %s"%(gm_val_list, gm_state))
        currentSta = gm_state
        if currentSta != last_state:
            break
    sleep(0.001)
```

- get_status 関数：

  この関数はグレースケールセンサーデータ（ val_list ）を解釈し、車のナビゲーション状態を決定します。

  車の状態は、どのセンサーがラインを検出するかに基づいて、 forward 、 left 、 right または stop になります。

```
def get_status(val_list):
    _state = px.get_line_status(val_list)  # [bool, bool, bool], 0 はライン、1 は背景を意味します
    if _state == [0, 0, 0]:
        return 'stop'
    elif _state[1] == 1:
        return 'forward'
    elif _state[0] == 1:
        return 'right'
    elif _state[2] == 1:
        return 'left'
```

- Main Loop:

  - while True ループは継続的にグレースケールデータをチェックし、それに応じて車の動きを調整します。

  - gm_state に応じて、ステアリング角度と動きの方向を設定します。

```
if __name__=='__main__':
    try:
        while True:
            gm_val_list = px.get_grayscale_data()
            gm_state = get_status(gm_val_list)
            print("gm_val_list: %s, %s"%(gm_val_list, gm_state))

            if gm_state != "stop":
                last_state = gm_state

            if gm_state == 'forward':
                px.set_dir_servo_angle(0)
                px.forward(px_power)
            elif gm_state == 'left':
                px.set_dir_servo_angle(offset)
```

(次のページに続く)

(前のページからの続き)

```
            px.forward(px_power)
        elif gm_state == 'right':
            px.set_dir_servo_angle(-offset)
            px.forward(px_power)
        else:
            outHandle()
```

- 安全性とクリーンアップ：

  `try...finally` ブロックは、スクリプトが中断または終了したときに車が停止することを保証します。

```
finally:
px.stop()
print("stop and exit")
sleep(0.1)
```

要約すると、このスクリプトはグレースケールセンサーを使用して Picarx ロボットカーをナビゲートします。センサーデータを継続的に読み取り、方向を決定し、それに応じて車の動きとステアリングを調整します。outHandle 関数は、車が大きくパスを調整する必要がある場合の追加ロジックを提供します。

4.9 6. 崖検出

PiCar-X に少し自己保護意識を与えて、自身のグレースケールモジュールを使用して崖からの突進を避けるようにしましょう。

この例では、車は休止状態になります。崖に押し出された場合、緊急に目覚め、後退し、「danger」と言います。

コードの実行

```
cd ~/picar-x/example
sudo python3 6.cliff_detection.py
```

コード

注釈: 以下のコードを **変更/リセット/コピー/実行/停止** することができます。しかし、それをする前に、`picar-x/example` のようなソースコードのパスに移動する必要があります。コードを変更した後、直接実行して効果を確認できます。

```
from picarx import Picarx
from time import sleep
from robot_hat import TTS

tts = TTS()
tts.lang("en-US")

px = Picarx()
# px = Picarx(grayscale_pins=['A0', 'A1', 'A2'])
# manual modify reference value
px.set_cliff_reference([200, 200, 200])

current_state = None
px_power = 10
offset = 20
last_state = "safe"

if __name__=='__main__':
    try:
        while True:
            gm_val_list = px.get_grayscale_data()
            gm_state = px.get_cliff_status(gm_val_list)
            # print("cliff status is:  %s"%gm_state)

            if gm_state is False:
                state = "safe"
                px.stop()
            else:
                state = "danger"
                px.backward(80)
                if last_state == "safe":
                    tts.say("danger")
                    sleep(0.1)
            last_state = state

    finally:
        px.stop()
        print("stop and exit")
        sleep(0.1)
```

どのように動作するのか？

崖を検出する機能は次のようになります：

- `get_grayscale_data()`：このメソッドは直接 3 つのセンサーの読み取り値を出力します。右から左に向かっています。エリアが明るいほど、得られる値が大きくなります。
- `get_cliff_status(gm_val_list)`：このメソッドは 3 つのプローブの読み取り値を比較し、結果を出力します。結果が真であれば、車の前方に崖があることが検出されます。

4.10 7. コンピュータービジョン

このプロジェクトでは、コンピュータービジョンの分野に正式に入ります！

コードの実行

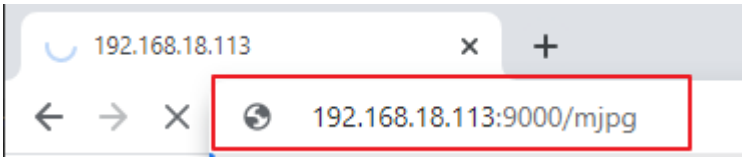
```
cd ~/picar-x/example
sudo python3 7.display.py
```

画像の表示

コードを実行すると、ターミナルに次のプロンプトが表示されます：

```
No desktop !
* Serving Flask app "vilib.vilib" (lazy loading)
* Environment: production
WARNING: Do not use the development server in a production environment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:9000/ (Press CTRL+C to quit)
```

次に、ブラウザで `http://<your IP>:9000/mjpg` にアクセスして、ビデオ画面を表示できます。例えば: `https://192.168.18.113:9000/mjpg`



プログラムを実行すると、最後に以下の情報が表示されます：

- 機能を呼び出すためにキーを入力してください！
- q: 写真を撮る
- 1: 色の検出 : 赤

- 2: 色の検出 : オレンジ
- 3: 色の検出 : 黄色
- 4: 色の検出 : 緑
- 5: 色の検出 : 青
- 6: 色の検出 : 紫
- 0: 色の検出をオフにする
- r: QR コードをスキャン
- f: 顔の検出をオン/オフに切り替える
- s: 検出されたオブジェクトの情報を表示

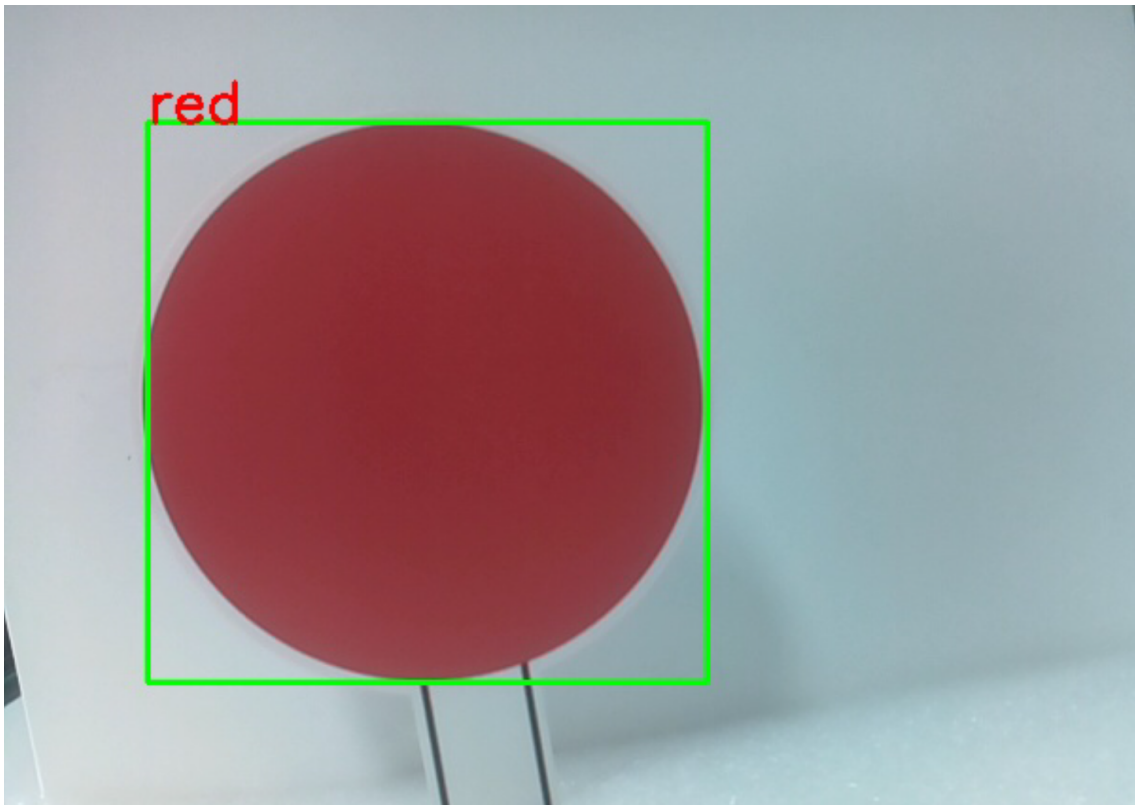
プロンプトに従って、対応する機能をアクティブにしてください。

- **写真を撮る**

  ターミナルで `q` と入力して Enter を押します。カメラが現在見ている画像が保存されます（色の検出機能がオンになっている場合は、保存された画像にマークボックスも表示されます）。Raspberry Pi の `/home/{username}/Pictures/` ディレクトリからこれらの写真を見ることができます。*Filezilla* **ソフトウェア** のようなツールを使用して、写真を PC に転送できます。

- **色の検出**

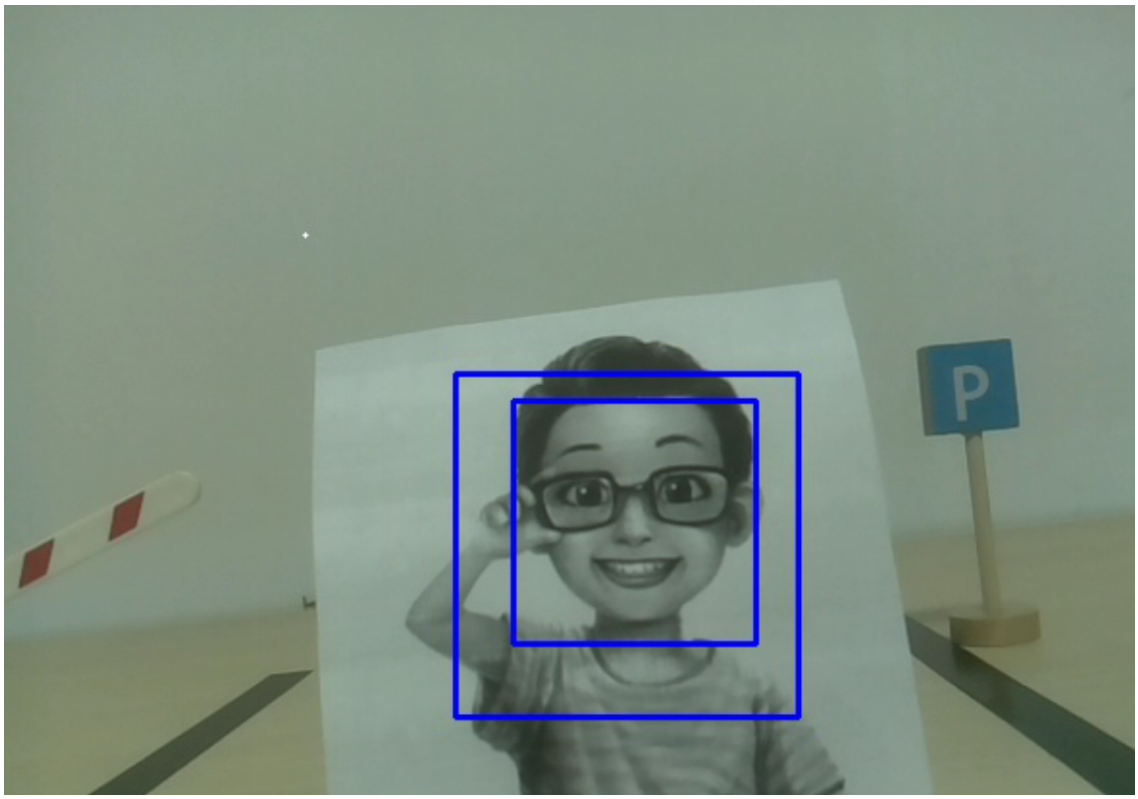
  `1〜6` の間の数字を入力すると、「赤、オレンジ、黄色、緑、青、紫」のうちの一つの色を検出します。`0` を入力すると、色の検出をオフにします。



注釈: 色の検出には PDF **カラーカード** をダウンロードして印刷することができます。

- **顔の検出**

  f と入力して顔の検出をオンにします。



- **QR コードの検出**

  r と入力して QR コード認識を開きます。QR コードが認識されるまで他の操作はできません。QR コードのデコード情報がターミナルに表示されます。

- **情報の表示**

  s と入力すると、ターミナルに顔の検出（および色の検出）対象の情報が表示されます。測定されたオブジェクトの中心座標（X、Y）とサイズ（幅、高さ）を含みます。

コード

```
from pydoc import text
from vilib import Vilib
from time import sleep, time, strftime, localtime
import threading
import readchar
import os

flag_face = False
flag_color = False
qr_code_flag = False

manual = '''
Input key to call the function!
    q: Take photo
    1: Color detect : red
```

(次のページに続く)

(前のページからの続き)

```
    2: Color detect : orange
    3: Color detect : yellow
    4: Color detect : green
    5: Color detect : blue
    6: Color detect : purple
    0: Switch off Color detect
    r: Scan the QR code
    f: Switch ON/OFF face detect
    s: Display detected object information
'''

color_list = ['close', 'red', 'orange', 'yellow',
        'green', 'blue', 'purple',
]

def face_detect(flag):
    print("Face Detect:" + str(flag))
    Vilib.face_detect_switch(flag)


def qrcode_detect():
    global qr_code_flag
    if qr_code_flag == True:
        Vilib.qrcode_detect_switch(True)
        print("Waitting for QR code")

    text = None
    while True:
        temp = Vilib.detect_obj_parameter['qr_data']
        if temp != "None" and temp != text:
            text = temp
            print('QR code:%s'%text)
        if qr_code_flag == False:
            break
        sleep(0.5)
    Vilib.qrcode_detect_switch(False)


def take_photo():
```

(次のページに続く)

(前のページからの続き)

```
    _time = strftime('%Y-%m-%d-%H-%M-%S',localtime(time()))
    name = 'photo_%s'%_time
    username = os.getlogin()

    path = f"/home/{username}/Pictures/"
    Vilib.take_photo(name, path)
    print('photo save as %s%s.jpg'%(path,name))


def object_show():
    global flag_color, flag_face

    if flag_color is True:
        if Vilib.detect_obj_parameter['color_n'] == 0:
            print('Color Detect: None')
        else:
            color_coodinate = (Vilib.detect_obj_parameter['color_x'],Vilib.detect_obj_
↪parameter['color_y'])
            color_size = (Vilib.detect_obj_parameter['color_w'],Vilib.detect_obj_
↪parameter['color_h'])
            print("[Color Detect] ","Coordinate:",color_coodinate,"Size",color_size)

    if flag_face is True:
        if Vilib.detect_obj_parameter['human_n'] == 0:
            print('Face Detect: None')
        else:
            human_coodinate = (Vilib.detect_obj_parameter['human_x'],Vilib.detect_obj_
↪parameter['human_y'])
            human_size = (Vilib.detect_obj_parameter['human_w'],Vilib.detect_obj_
↪parameter['human_h'])
            print("[Face Detect] ","Coordinate:",human_coodinate,"Size",human_size)


def main():
    global flag_face, flag_color, qr_code_flag
    qrcode_thread = None

    Vilib.camera_start(vflip=False,hflip=False)
    Vilib.display(local=True,web=True)
```

(次のページに続く)

(前のページからの続き)

```
    print(manual)

    while True:
        # readkey
        key = readchar.readkey()
        key = key.lower()
        # take photo
        if key == 'q':
            take_photo()
        # color detect
        elif key != '' and key in ('0123456'):  # '' in ('0123') -> True
            index = int(key)
            if index == 0:
                flag_color = False
                Vilib.color_detect('close')
            else:
                flag_color = True
                Vilib.color_detect(color_list[index]) # color_detect(color:str -> color_
↪name/close)
            print('Color detect : %s'%color_list[index])
        # face detection
        elif key =="f":
            flag_face = not flag_face
            face_detect(flag_face)
        # qrcode detection
        elif key =="r":
            qr_code_flag = not qr_code_flag
            if qr_code_flag == True:
                if qrcode_thread == None or not qrcode_thread.is_alive():
                    qrcode_thread = threading.Thread(target=qrcode_detect)
                    qrcode_thread.setDaemon(True)
                    qrcode_thread.start()
            else:
                if qrcode_thread != None and qrcode_thread.is_alive():
                # wait for thread to end
                    qrcode_thread.join()
                    print('QRcode Detect: close')
        # show detected object information
        elif key == "s":
```

(次のページに続く)

(前のページからの続き)

```
            object_show()

        sleep(0.5)


if __name__ == "__main__":
    main()
```

どのように動作するのか？

ここで注意するべき最初のことは、次の機能です。これらの 2 つの機能により、カメラを起動できます。

```
Vilib.camera_start()
Vilib.display()
```

「オブジェクト検出」に関連する機能：

- `Vilib.face_detect_switch(True)`：顔検出のオン/オフ切替
- `Vilib.color_detect(color)`：色検出について、一度に 1 色の検出のみ実行できます。入力できるパラメータは： `"red"`, `"orange"`, `"yellow"`, `"green"`, `"blue"`, `"purple"`
- `Vilib.color_detect_switch(False)`：色検出のオフ切替
- `Vilib.qrcode_detect_switch(False)`：QR コード検出のオン/オフ切替、QR コードのデコードデータを返します。
- `Vilib.gesture_detect_switch(False)`：ジェスチャー検出のオン/オフ切替
- `Vilib.traffic_sign_detect_switch(False)`：交通標識検出のオン/オフ切替

標的によって検出された情報は `detect_obj_parameter = Manager().dict()` 辞書に保存されます。

メインプログラムでは、次のように使用できます：

```
Vilib.detect_obj_parameter['color_x']
```

辞書のキーとその使い方は、次のリストに示されています：

- `color_x`：検出された色ブロックの中心座標の x 値、範囲は 0〜320
- `color_y`：検出された色ブロックの中心座標の y 値、範囲は 0〜240
- `color_w`：検出された色ブロックの幅、範囲は 0〜320
- `color_h`：検出された色ブロックの高さ、範囲は 0〜240

- color_n：検出された色パッチの数
- human_x：検出された人間の顔の中心座標の x 値、範囲は 0〜320
- human_y：検出された顔の中心座標の y 値、範囲は 0〜240
- human_w：検出された人間の顔の幅、範囲は 0〜320
- human_h：検出された顔の高さ、範囲は 0〜240
- human_n：検出された顔の数
- traffic_sign_x：検出された交通標識の中心座標の x 値、範囲は 0〜320
- traffic_sign_y：検出された交通標識の中心座標の y 値、範囲は 0〜240
- traffic_sign_w：検出された交通標識の幅、範囲は 0〜320
- traffic_sign_h：検出された交通標識の高さ、範囲は 0〜240
- traffic_sign_t：検出された交通標識の内容、値のリストは *['stop','right','left','forward']*
- gesture_x：検出されたジェスチャーの中心座標の x 値、範囲は 0〜320
- gesture_y：検出されたジェスチャーの中心座標の y 値、範囲は 0〜240
- gesture_w：検出されたジェスチャーの幅、範囲は 0〜320
- gesture_h：検出されたジェスチャーの高さ、範囲は 0〜240
- gesture_t：検出されたジェスチャーの内容、値のリストは *["paper","scissor","rock"]*
- qr_date：検出されている QR コードの内容
- qr_x：検出対象の QR コードの中心座標の x 値、範囲は 0〜320
- qr_y：検出対象の QR コードの中心座標の y 値、範囲は 0〜240
- qr_w：検出対象の QR コードの幅、範囲は 0〜320
- qr_h：検出対象の QR コードの高さ、範囲は 0〜320

4.11 8. あなたを見つめる

このプロジェクトは、 *7.* **コンピュータービジョン** プロジェクトに基づいており、顔検出アルゴリズムが追加されています。

カメラの前に現れると、顔を認識し、ジンバルを調整して顔をフレームの中心に保ちます。

`http://<your IP>:9000/mjpg` で画面を表示できます。

コードの実行

```
cd ~/picar-x/example
sudo python3 8.stare_at_you.py
```

コードが実行されると、車のカメラは常にあなたの顔を見つめ続けます。

コード

```
from picarx import Picarx
from time import sleep
from vilib import Vilib

px = Picarx()

def clamp_number(num,a,b):
    return max(min(num, max(a, b)), min(a, b))

def main():
    Vilib.camera_start()
    Vilib.display()
    Vilib.face_detect_switch(True)
    x_angle =0
    y_angle =0
    while True:
        if Vilib.detect_obj_parameter['human_n']!=0:
            coordinate_x = Vilib.detect_obj_parameter['human_x']
            coordinate_y = Vilib.detect_obj_parameter['human_y']

            # change the pan-tilt angle for track the object
            x_angle +=(coordinate_x*10/640)-5
            x_angle = clamp_number(x_angle,-35,35)
            px.set_cam_pan_angle(x_angle)

            y_angle -=(coordinate_y*10/480)-5
            y_angle = clamp_number(y_angle,-35,35)
            px.set_cam_tilt_angle(y_angle)

            sleep(0.05)

        else :
```

(次のページに続く)

(前のページからの続き)

```
            pass
            sleep(0.05)

if __name__ == "__main__":
    try:
    main()

    finally:
        px.stop()
        print("stop and exit")
        sleep(0.1)
```

どのように動作するのか？

while True の中のこれらのコード行により、カメラが顔を追いかけます。

```
while True:
    if Vilib.detect_obj_parameter['human_n']!=0:
        coordinate_x = Vilib.detect_obj_parameter['human_x']
        coordinate_y = Vilib.detect_obj_parameter['human_y']

        # change the pan-tilt angle for track the object
        x_angle +=(coordinate_x*10/640)-5
        x_angle = clamp_number(x_angle,-35,35)
        px.set_cam_pan_angle(x_angle)

        y_angle -=(coordinate_y*10/480)-5
        y_angle = clamp_number(y_angle,-35,35)
        px.set_cam_tilt_angle(y_angle)
```

1. 検出された人間の顔があるかどうかをチェックします

   ```
   Vilib.detect_obj_parameter['human_n'] != 0
   ```

2. 人間の顔が検出された場合、検出された顔の座標（ coordinate_x と coordinate_y ）を取得します。

3. 検出された顔の位置に基づいて新しいパンとチルト角度（ x_angle と y_angle ）を計算し、それらを調整して顔を追いかけます。

4. clamp_number 関数を使用してパンとチルト角度を指定された範囲内に制限します。

5. px.set_cam_pan_angle() と px.set_cam_tilt_angle() を使用してカメラのパンとチルト角度を設定

します。

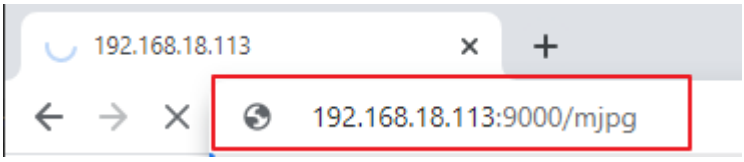
4.12 9. ビデオ録画

この例では、録画機能の使用方法を案内します。

コードの実行

```
cd ~/picar-x/example
sudo python3 9.record_video.py
```

コードを実行した後、ブラウザで http://<your IP>:9000/mjpg にアクセスして、ビデオ画面を表示できます。
例えば：http://192.168.18.113:9000/mjpg



キーボードのキーを押すことで、録画を停止または開始できます。

- q を押して録画を開始または一時停止/続行し、 e を押して録画を停止または保存します。
- プログラムを終了したい場合は、 ctrl+c を押してください。

コード

```
from time import sleep,strftime,localtime
from vilib import Vilib
import readchar
import os

manual = '''
Press keys on keyboard to control recording:
    Q: record/pause/continue
    E: stop
    Ctrl + C: Quit
'''

def print_overwrite(msg,  end='', flush=True):
    print('\r\033[2K', end='',flush=True)
    print(msg, end=end, flush=True)
```

(次のページに続く)

(前のページからの続き)

```
def main():
    rec_flag = 'stop' # start,pause,stop
    vname = None
    username = os.getlogin()

    Vilib.rec_video_set["path"] = f"/home/{username}/Videos/" # set path

    Vilib.camera_start(vflip=False,hflip=False)
    Vilib.display(local=True,web=True)
    sleep(0.8)  # wait for startup

    print(manual)
    while True:
        # read keyboard
        key = readchar.readkey()
        key = key.lower()
        # start,pause
        if key == 'q':
            key = None
            if rec_flag == 'stop':
                rec_flag = 'start'
                # set name
                vname = strftime("%Y-%m-%d-%H.%M.%S", localtime())
                Vilib.rec_video_set["name"] = vname
                # start record
                Vilib.rec_video_run()
                Vilib.rec_video_start()
                print_overwrite('rec start ...')
            elif rec_flag == 'start':
                rec_flag = 'pause'
                Vilib.rec_video_pause()
                print_overwrite('pause')
            elif rec_flag == 'pause':
                rec_flag = 'start'
                Vilib.rec_video_start()
                print_overwrite('continue')
        # stop
        elif key == 'e' and rec_flag != 'stop':
            key = None
```

(次のページに続く)

(前のページからの続き)

```
            rec_flag = 'stop'
            Vilib.rec_video_stop()
            print_overwrite("The video saved as %s%s.avi"%(Vilib.rec_video_set["path"],
↪vname),end='\n')
        # quit
        elif key == readchar.key.CTRL_C:
            Vilib.camera_close()
            print('\nquit')
            break

        sleep(0.1)

if __name__ == "__main__":
    main()
```

どのように動作するのか？

録画に関連する機能は以下の通りです：

- `Vilib.rec_video_run(video_name)`：ビデオの録画を開始するスレッドを開始します。 `video_name` はビデオファイルの名前で、文字列である必要があります。
- `Vilib.rec_video_start()`：ビデオ録画を開始または続行します。
- `Vilib.rec_video_pause()`：録画を一時停止します。
- `Vilib.rec_video_stop()`：録画を停止します。

`Vilib.rec_video_set["path"] = f"/home/{username}/Videos/"` はビデオファイルの保存場所を設定します。

4.13 10. ブルファイト

PiCar-X を怒れるブルにしましょう！カメラを使って赤い布を追いかけ、突進させます！

コードの実行

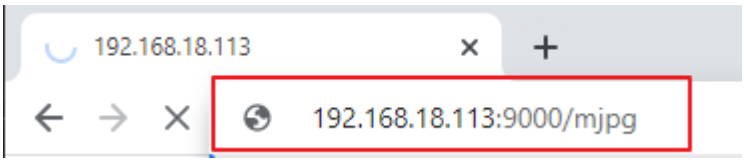
```
cd ~/picar-x/example
sudo python3 10.bull_fight.py
```

画像の表示

コードを実行すると、ターミナルに次のプロンプトが表示されます：

```
No desktop !
* Serving Flask app "vilib.vilib" (lazy loading)
* Environment: production
WARNING: Do not use the development server in a production environment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:9000/ (Press CTRL+C to quit)
```

次に、ブラウザで `http://<your IP>:9000/mjpg` にアクセスして、ビデオ画面を表示できます。例えば: `https://192.168.18.113:9000/mjpg`



コード

注釈: 以下のコードを **変更/リセット/コピー/実行/停止** することができます。しかし、それをする前に、`picar-x\examples` のようなソースコードのパスに移動する必要があります。コードを変更した後、直接実行して効果を確認できます。

```
from picarx import Picarx
from time import sleep
from vilib import Vilib

px = Picarx()

def clamp_number(num,a,b):
return max(min(num, max(a, b)), min(a, b))

def main():
    Vilib.camera_start()
    Vilib.display()
    Vilib.color_detect("red")
    speed = 50
    dir_angle=0
    x_angle =0
    y_angle =0
```

(次のページに続く)

(前のページからの続き)

```
    while True:
        if Vilib.detect_obj_parameter['color_n']!=0:
            coordinate_x = Vilib.detect_obj_parameter['color_x']
            coordinate_y = Vilib.detect_obj_parameter['color_y']

            # change the pan-tilt angle for track the object
            x_angle +=(coordinate_x*10/640)-5
            x_angle = clamp_number(x_angle,-35,35)
            px.set_cam_pan_angle(x_angle)

            y_angle -=(coordinate_y*10/480)-5
            y_angle = clamp_number(y_angle,-35,35)
            px.set_cam_tilt_angle(y_angle)

            # move
            # The movement direction will change slower than the pan/tilt direction
            # change to avoid confusion when the picture changes at high speed.
            if dir_angle > x_angle:
                dir_angle -= 1
            elif dir_angle < x_angle:
                dir_angle += 1
            px.set_dir_servo_angle(x_angle)
            px.forward(speed)
            sleep(0.05)

        else :
            px.forward(0)
            sleep(0.05)


if __name__ == "__main__":
    try:
    main()

    finally:
        px.stop()
        print("stop and exit")
        sleep(0.1)
```

どのように動作するのか？

この例の以下の 3 つの部分に注意が必要です：

1. メイン関数を定義する：
    - `Vilib.camera_start()` を使用してカメラを開始します。
    - `Vilib.display()` を使用してカメラフィードを表示します。
    - `Vilib.color_detect("red")` を使用して色検出を有効にし、ターゲット色を「赤」として指定します。
    - 変数を初期化：車の移動速度のための `speed`、車の移動方向角のための `dir_angle`、カメラのパン角度のための `x_angle`、カメラのチルト角度のための `y_angle`。
2. 赤色のオブジェクトを追跡するために継続的なループ（while True）に入る：
    - 検出された赤色のオブジェクトがあるかどうかをチェックします（ `Vilib.detect_obj_parameter['color_n'] != 0`）。
    - 赤色のオブジェクトが検出された場合、その座標（ `coordinate_x` と `coordinate_y`）を取得します。
    - 検出されたオブジェクトの位置に基づいて新しいパンとチルト角度（`x_angle` と `y_angle`）を計算し、それらを調整してオブジェクトを追いかけます。
    - `clamp_number` 関数を使用してパンとチルト角度を指定された範囲内に制限します。
    - `px.set_cam_pan_angle()` と `px.set_cam_tilt_angle()` を使用してカメラのパンとチルト角度を設定し、オブジェクトを視界に保ちます。
3. `dir_angle` と `x_angle` の差に基づいて車の動きを制御する：
    - `dir_angle` が `x_angle` より大きい場合、方向角を徐々に変更するために `dir_angle` を 1 減らします。
    - `dir_angle` が `x_angle` より小さい場合、`dir_angle` を 1 増やします。
    - `px.set_dir_servo_angle()` を使用して方向サーボ角度を設定し、車の車輪を適切に操縦します。
    - `px.forward(speed)` を使用して、指定された速度で車を前進させます。

4.14 11. ビデオカー

このプログラムは PiCar-X からの一人称視点を提供します！キーボードの WSAD キーを使用して移動方向を制御し、O と P で速度を調整します。

コードの実行

```
cd ~/picar-x/example
sudo python3 11.video_car.py
```

コードが実行されると、PiCar-X が撮影しているものを見て、次のキーを押すことで制御できます。

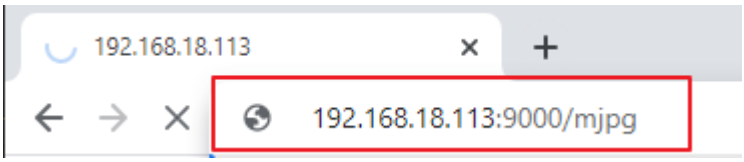
- O: 速度アップ
- P: 速度ダウン
- W: 前進
- S: 後進
- A: 左折
- D: 右折
- F: 停止
- T: 写真を撮る
- Ctrl+C: 終了

画像の表示

コードを実行すると、ターミナルに次のプロンプトが表示されます：

```
No desktop !
* Serving Flask app "vilib.vilib" (lazy loading)
* Environment: production
WARNING: Do not use the development server in a production environment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:9000/ (Press CTRL+C to quit)
```

次に、ブラウザで `http://<your IP>:9000/mjpg` にアクセスして、ビデオ画面を表示できます。例えば: `https://192.168.18.113:9000/mjpg`



コード

```
#!/usr/bin/env python3

from robot_hat.utils import reset_mcu
from picarx import Picarx
from vilib import Vilib
```

(次のページに続く)

(前のページからの続き)

```
from time import sleep, time, strftime, localtime
import readchar

import os
user = os.getlogin()
user_home = os.path.expanduser(f'~{user}')

reset_mcu()
sleep(0.2)

manual = '''
Press key to call the function(non-case sensitive):

    O: speed up
    P: speed down
    W: forward
    S: backward
    A: turn left
    D: turn right
    F: stop
    T: take photo

    ctrl + c: Press twice to exit the program
'''



px = Picarx()

def take_photo():
    _time = strftime('%Y-%m-%d-%H-%M-%S',localtime(time()))
    name = 'photo_%s'%_time
    path = f"{user_home}/Pictures/picar-x/"
    Vilib.take_photo(name, path)
    print('\nphoto save as %s%s.jpg'%(path,name))


def move(operate:str, speed):

    if operate == 'stop':
```

(次のページに続く)

(前のページからの続き)

```
        px.stop()
    else:
        if operate == 'forward':
            px.set_dir_servo_angle(0)
            px.forward(speed)
        elif operate == 'backward':
            px.set_dir_servo_angle(0)
            px.backward(speed)
        elif operate == 'turn left':
            px.set_dir_servo_angle(-30)
            px.forward(speed)
        elif operate == 'turn right':
            px.set_dir_servo_angle(30)
            px.forward(speed)



def main():
    speed = 0
    status = 'stop'

    Vilib.camera_start(vflip=False,hflip=False)
    Vilib.display(local=True,web=True)
    sleep(2)  # wait for startup
    print(manual)

    while True:
        print("\rstatus: %s , speed: %s    "%(status, speed), end='', flush=True)
        # readkey
        key = readchar.readkey().lower()
        # operation
        if key in ('wsadfop'):
            # throttle
            if key == 'o':
                if speed <=90:
                    speed += 10
            elif key == 'p':
                if speed >=10:
                    speed -= 10
```

(次のページに続く)

(前のページからの続き)

```
                if speed == 0:
                    status = 'stop'
            # direction
            elif key in ('wsad'):
                if speed == 0:
                    speed = 10
                if key == 'w':
                    # Speed limit when reversing,avoid instantaneous current too large
                    if status != 'forward' and speed > 60:
                        speed = 60
                    status = 'forward'
                elif key == 'a':
                    status = 'turn left'
                elif key == 's':
                    if status != 'backward' and speed > 60: # Speed limit when reversing
                        speed = 60
                    status = 'backward'
                elif key == 'd':
                    status = 'turn right'
            # stop
            elif key == 'f':
                status = 'stop'
            # move
            move(status, speed)
        # take photo
        elif key == 't':
            take_photo()
        # quit
        elif key == readchar.key.CTRL_C:
            print('\nquit ...')
            px.stop()
            Vilib.camera_close()
            break

        sleep(0.1)


if __name__ == "__main__":
    try:
```

(次のページに続く)

(前のページからの続き)

```
        main()
    except Exception as e:
        print("error:%s"%e)
    finally:
        px.stop()
        Vilib.camera_close()
```

4.15 12. 宝探し

部屋に迷路を用意し、6 つの角に 6 色の異なるカードを配置してください。次に、PiCar-X をコントロールして、これらの色カードを一つずつ探索しましょう！

注釈: 色検出用に PDF **カラーカード** をダウンロードして印刷することができます。

コードの実行

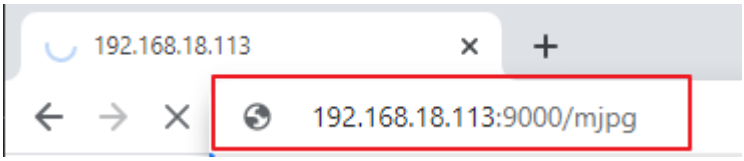
```
cd ~/picar-x/example
sudo python3 12.treasure_hunt.py
```

画像の表示

コードを実行すると、ターミナルに次のプロンプトが表示されます：

```
No desktop !
* Serving Flask app "vilib.vilib" (lazy loading)
* Environment: production
WARNING: Do not use the development server in a production environment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:9000/ (Press CTRL+C to quit)
```

次に、ブラウザで http://<your IP>:9000/mjpg にアクセスして、ビデオ画面を表示できます。例えば：http://192.168.18.113:9000/mjpg



コード

```
from picarx import Picarx
from time import sleep
from robot_hat import Music,TTS
from vilib import Vilib
import readchar
import random
import threading

px = Picarx()

music = Music()
tts = TTS()

manual = '''
Press keys on keyboard to control Picar-X!
    w: Forward
    a: Turn left
    s: Backward
    d: Turn right
    space: Say the target again
    ctrl+c: Quit
'''

color = "red"
color_list=["red","orange","yellow","green","blue","purple"]

def renew_color_detect():
    global color
    color = random.choice(color_list)
    Vilib.color_detect(color)
    tts.say("Look for " + color)

key = None
lock = threading.Lock()
def key_scan_thread():
    global key
    while True:
        key_temp = readchar.readkey()
        print('\r',end='')
```

(次のページに続く)

(前のページからの続き)

```
        with lock:
            key = key_temp.lower()
            if key == readchar.key.SPACE:
                key = 'space'
            elif key == readchar.key.CTRL_C:
                key = 'quit'
                break
        sleep(0.01)

def car_move(key):
    if 'w' == key:
        px.set_dir_servo_angle(0)
        px.forward(80)
    elif 's' == key:
        px.set_dir_servo_angle(0)
        px.backward(80)
    elif 'a' == key:
        px.set_dir_servo_angle(-30)
        px.forward(80)
    elif 'd' == key:
        px.set_dir_servo_angle(30)
        px.forward(80)


def main():
    global key
    Vilib.camera_start(vflip=False,hflip=False)
    Vilib.display(local=False,web=True)
    sleep(0.8)
    print(manual)

    sleep(1)
    _key_t = threading.Thread(target=key_scan_thread)
    _key_t.setDaemon(True)
    _key_t.start()

    tts.say("game start")
    sleep(0.05)
    renew_color_detect()
```

(次のページに続く)

(前のページからの続き)

```
    while True:

        if Vilib.detect_obj_parameter['color_n']!=0 and Vilib.detect_obj_parameter[
↪'color_w']>100:
            tts.say("will done")
            sleep(0.05)
            renew_color_detect()

        with lock:
            if key != None and key in ('wsad'):
                car_move(key)
                sleep(0.5)
                px.stop()
                key =  None
            elif key == 'space':
                tts.say("Look for " + color)
                key =  None
            elif key == 'quit':
                _key_t.join()
                print("\n\rQuit")
                break

        sleep(0.05)

if __name__ == "__main__":
    try:
        main()
    except KeyboardInterrupt:
        pass
    except Exception as e:
        print(f"ERROR: {e}")
    finally:
        Vilib.camera_close()
        px.stop()
        sleep(.2)
```

どのように動作するのか？

このコードの基本的なロジックを理解するために、以下の重要な部分に注目してください：

1. **初期化とインポート：**コードの最初にあるインポート文で使用されているライブラリを理解します。
2. **グローバル変数：**ターゲットの色とキーボード入力を追跡するためにコード全体で使用されるグローバル変数の定義。例えば `color` と `key` など。
3. `renew_color_detect()`：この関数はリストからランダムに色を選び、検出のターゲット色として設定します。また、選択された色をテキスト・トゥ・スピーチでアナウンスします。
4. `key_scan_thread()`：この関数は別のスレッドで実行され、継続的にキーボード入力をスキャンし、押されたキーで `key` 変数を更新します。スレッドセーフなアクセスのためにロックを使用します。
5. `car_move(key)`：この関数はキーボード入力（ `key` ）に基づいて PiCar-X の動きを制御します。ロボットの移動方向と速度を設定します。
6. `main()`：コードの全体的なロジックを統合する主要な機能です。以下を行います：
    - カメラを初期化し、カメラフィードを表示します。
    - キーボード入力をスキャンするための別のスレッドを作成します。
    - テキスト・トゥ・スピーチを使用してゲームの開始をアナウンスします。
    - 継続的なループに入ります：
        - 検出された色のオブジェクトをチェックし、有効なオブジェクトが検出された場合にはアクションをトリガーします。
        - キーボード入力を処理して、ロボットを制御し、ゲームと対話します。
    - ゲームの終了と、キーボード割り込みなどの例外を処理します。
    - カメラを閉じ、PiCar-X を停止することを確認します。

これらのコードの重要な部分を理解することで、PiCar-X ロボットがキーボード入力に応答し、カメラとオーディオ出力機能を使用して特定の色のオブジェクトを検出し、それと対話する基本的なロジックを把握できます。

4.16 13. アプリによる制御

SunFounder コントローラーは、Raspberry Pi/Pico ベースのロボットを制御するために使用されます。

このアプリには、ボタン、スイッチ、ジョイスティック、D パッド、スライダー、スロットルスライダーウィジェット、デジタルディスプレイ、超音波レーダー、グレースケール検出、スピードメーター入力ウィジェットが統合されています。

A-Q までの 17 エリアがあり、異なるウィジェットを配置して独自のコントローラーをカスタマイズできます。

さらに、このアプリケーションはライブビデオストリーミングサービスも提供しています。

このアプリを使用して PiCar-X コントローラーをカスタマイズしましょう。

どうやって？

1. sunfounder-controller モジュールをインストールします。

   最初に robot-hat、 vilib、 picar-x モジュールをインストールする必要があります。詳細は： **すべてのモジュールをインストールする（重要）** を参照してください。

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

2. コードを実行します。

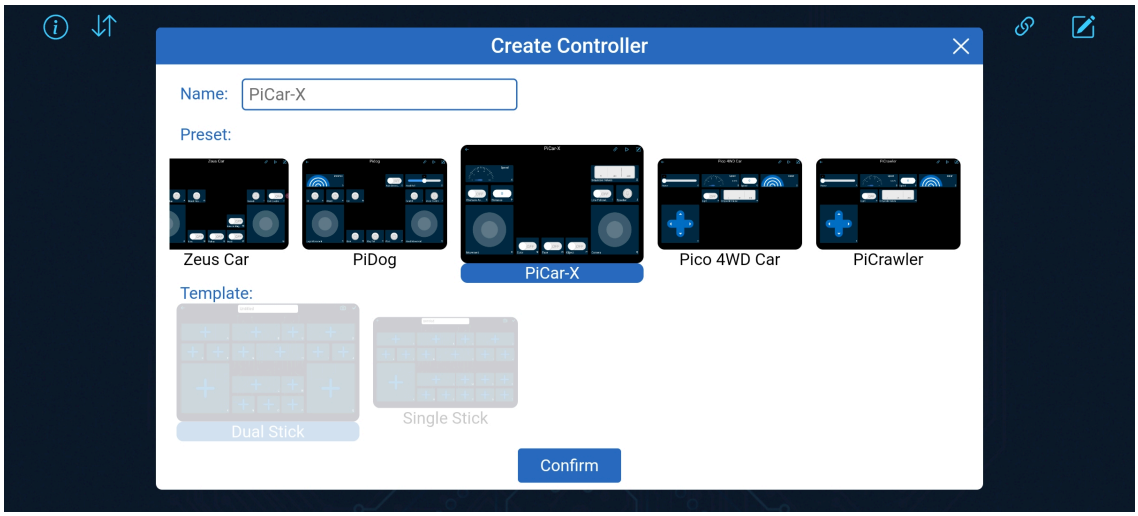
   ```
   cd ~/picar-x/example
   sudo python3 13.app_control.py
   ```

3. **APP Store(iOS)** または **Google Play(Android)** から SunFounder Controller をインストールします。

4. アプリを開き、新しいコントローラーを作成します。

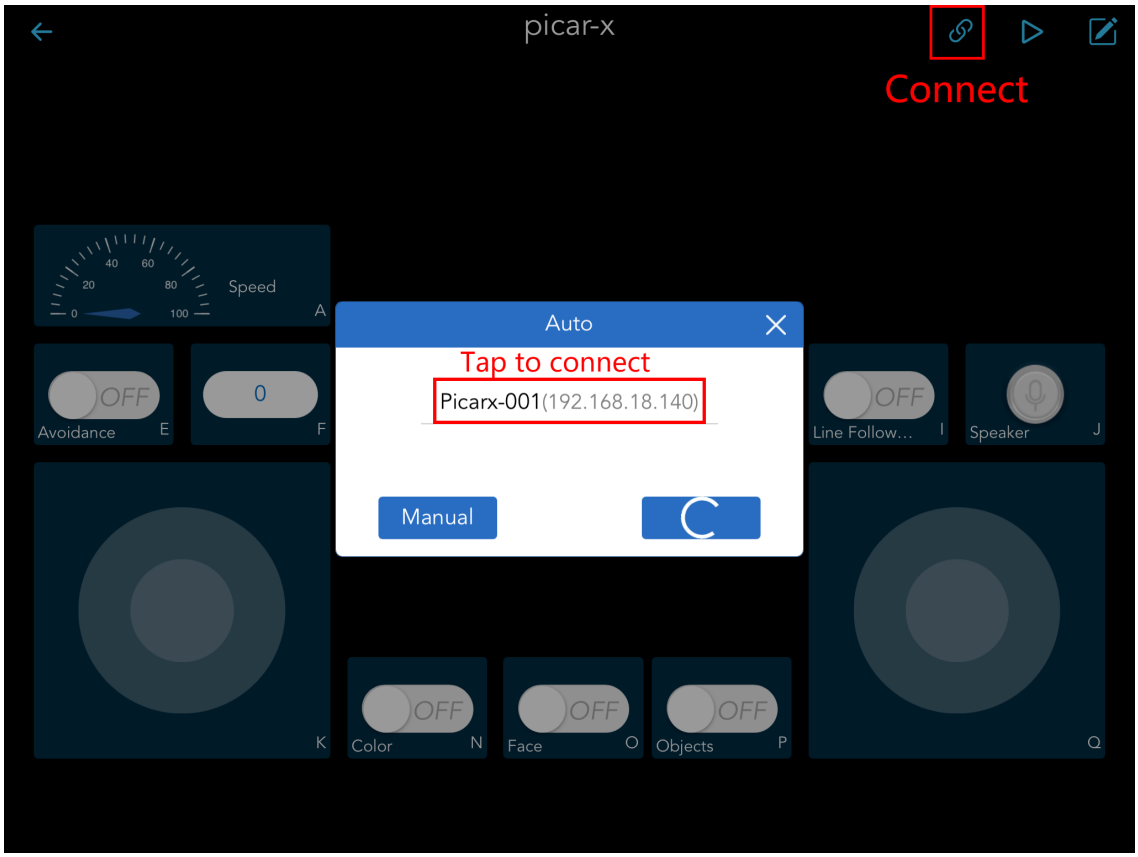
   SunFounder Controller アプリ内の + 記号をクリックして新しいコントローラーを作成します。

プリセットセクションには、いくつかの製品に対するプリセットコントローラーがあり、必要に応じて使用できます。ここでは、PiCar-X を選択します。

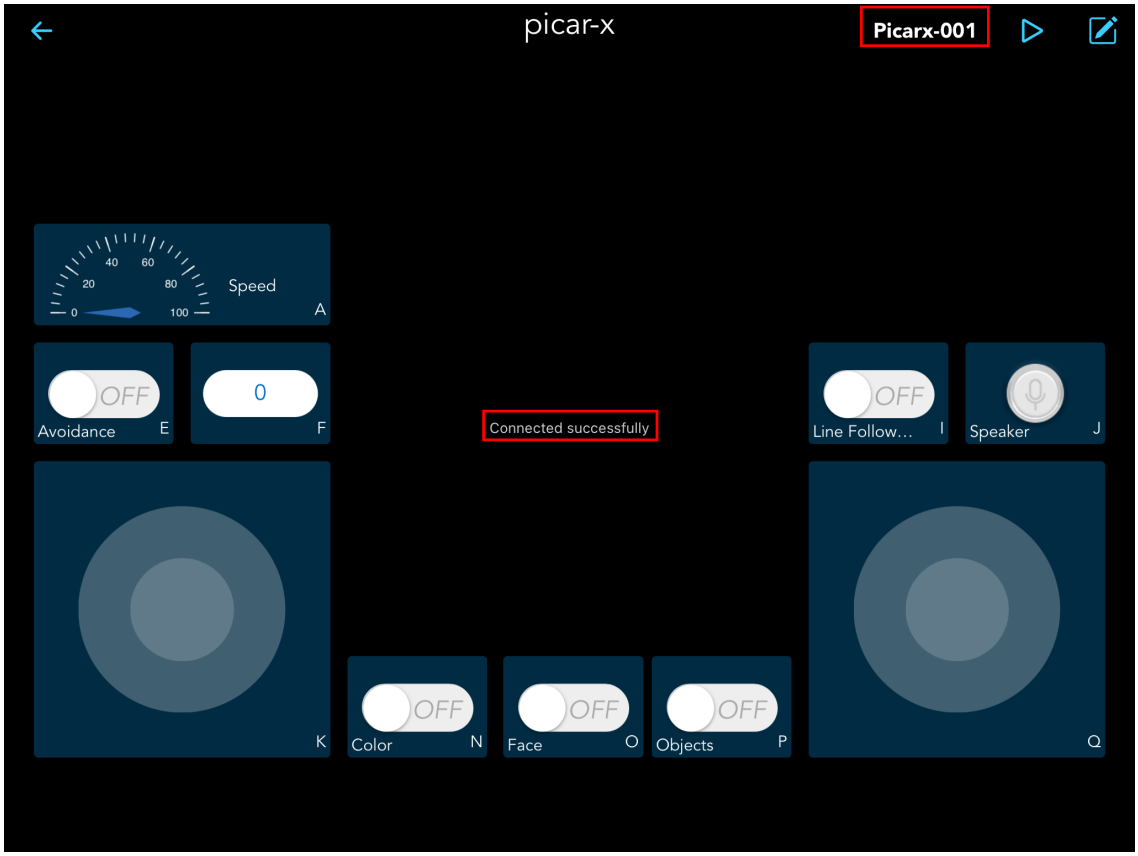


5. PiCar-x に接続する。

Connect ボタンをクリックすると、近くのロボットを自動的に検索します。その名前は `picarx_control.py` で定義されており、常に実行されている必要があります。

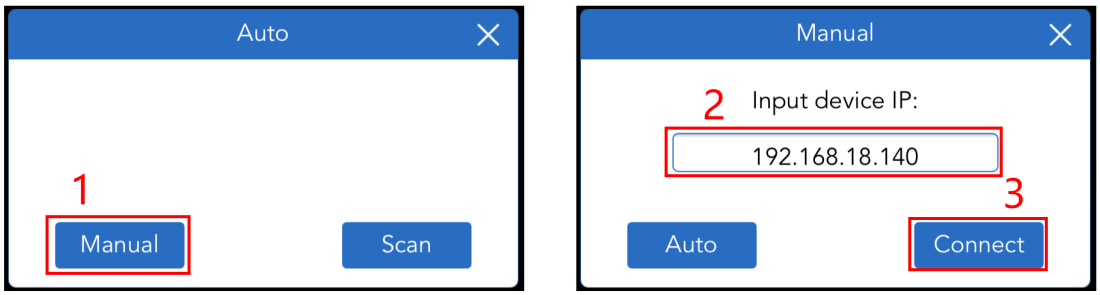


製品名をクリックすると「接続成功」というメッセージが表示され、製品名が右上に表示されます。



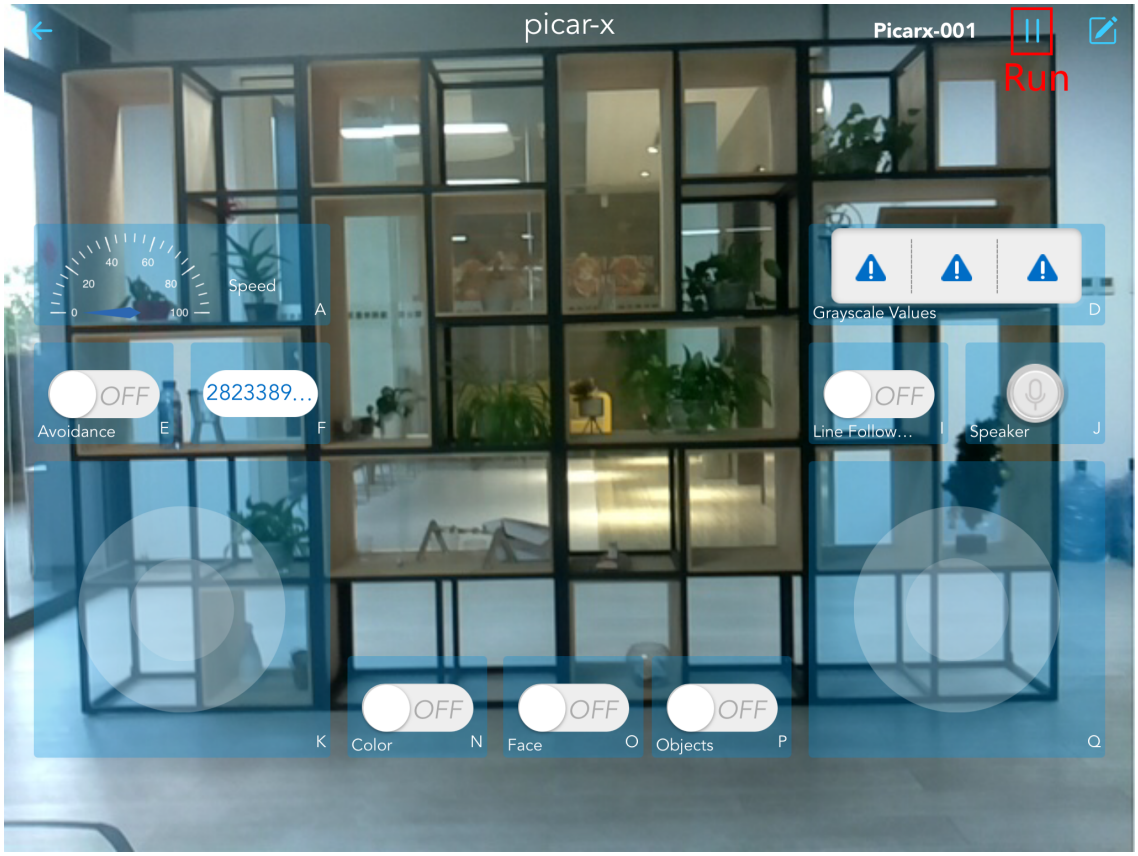
注釈:

- モバイルデバイスが PiCar-X と同じ LAN に接続されていることを確認する必要があります。
- 自動検索されない場合は、手動で IP を入力して接続することもできます。



6. このコントローラーを実行する。

   Run ボタンをクリックしてコントローラーを起動すると、車が撮影した映像が表示され、これらのウィジェットで PiCar-X を操作できます。



ウィジェットの機能は次のとおりです。

- **A**: 車の現在の速度を表示します。
- **E**: 障害物回避機能をオンにします。
- **I**: ラインフォロー機能をオンにします。
- **J**: 音声認識。このウィジェットを押して話し始め、離すと認識された音声が表示されます。`forward`、`backward`、`left`、`right` の 4 つのコマンドをコードに設定し、車を制御します。
- **K**: 車の前進、後進、左折、右折を制御します。
- **Q**: カメラ（頭部）を上下左右に動かします。
- **N**: 色認識機能をオンにします。
- **O**: 顔認識機能をオンにします。
- **P**: 物体認識機能をオンにし、約 90 種類の物体を認識できます。モデルのリストについては、こちらを参照してください： https://github.com/sunfounder/vilib/blob/master/workspace/coco_labels.txt。

第 5 章

Python ビデオコース

このビデオコースは、対話式学習に特化したオンラインの Python チュートリアルです。基本的な基盤をカバーする 3 つの入門ビデオが含まれており、Raspberry Pi のセットアップから始まり、PiCar-X の組み立て、必要なロボットモジュールのインストールまでを説明しています。この初期段階は、PiCar-X を使った様々なプロジェクトを始める前に、すべてが準備されていることを確認するためのものです。

導入部分に続いて、コースは 12 のプロジェクトビデオを特集しています。これらのプロジェクトは、PiCar-X の基本的な動きから始まり、キーボードコントロール、テキスト・トゥ・スピーチ（TTS）、障害物回避、ライン追跡、コンピュータビジョンへの応用まで、段階的にスキルを開発していきます。その後、複数の機能を組み合わせたより複雑なプロジェクトに進みます。オンラインチュートリアルからの例を実行する方法を教えるだけでなく、各トピックに追加の拡張を提供し、各機能の理解を深めることができます。

スタート

コースは、PiCar-X を操作するための基盤を築く 3 つの入門ビデオから始まります。これらのビデオは以下をカバーしています：

5.1 ビデオ A1：Raspberry Pi の始め方

これは PiCar-X のための最初のビデオです。

このビデオは、PiCar-X ロボットカーで使用するための Raspberry Pi のセットアップに関する包括的なチュートリアルを提供します。内容には以下が含まれます：

- PiCar-X の特徴。
- Raspberry Pi OS のイメージ作成方法。
- HDMI、PowerShell、リモートデスクトップ、SunFounder Create Agent を含むさまざまなプログラミング方法。
- 必要なロボットモジュールのインストール。

初心者の場合は、ステップバイステップで進めることをお勧めします。Raspberry Pi に詳しい場合は、ビデオの後半部分に従って必要なモジュールを直接インストールすることができます。

ビデオ

関連するオンラインチュートリアル

- *Python* **のクイックガイド**

5.2 ビデオ A2：PICAR-X の組み立て

このビデオでは、PICAR-X の組み立てに焦点を当てています。

チュートリアルの主な内容には以下が含まれます：

- **開梱**：キットコンポーネントの紹介、組み立て説明書、ツール、モジュール、モーターなど。
- **組み立てガイド**：Raspberry Pi にネジを取り付けることから始まり、ケーブルを接続し、ロボットハットを設定するまでの PICAR-X の組み立て手順の詳細。
- **サーボキャリブレーション**：サンファウンダーの Create Agent を使用してサーボをゼロ調整する。
- **モーターとバッテリーの取り付け**：フレームにモーターを取り付け、バッテリーを固定する。
- **センサーとカメラの取り付け**：グレースケールセンサー、超音波センサー、カメラを取り付けて接続する。
- **配線と最終セットアップ**：さまざまなコンポーネントのための配線を整理し接続し、最終チェックと配線の整理を行う。

このビデオは、組み立てプロセスの各ステップを通じてガイドするように設計されており、各コンポーネントがどのように一緒にフィットするかを徹底的に理解することを目的としています。

ビデオ

関連するオンラインチュートリアル

- **組み立て手順**

5.3 ビデオ A3：PiCar-X のキャリブレーション

このシリーズの 3 番目のビデオは、PiCar-X のキャリブレーションと Robot HAT の紹介に焦点を当てています。チュートリアルはいくつかの主要なセクションに分かれています：

- **グレースケールセンサーのキャリブレーション**：グレースケールセンサーにアクセスし、キャリブレーションする方法についての指示。

- **DC モーターとサーボのキャリブレーション**：方向と速度について DC モーターをキャリブレーションする手順、およびステアリングとカメラのパン/チルトサーボ。
- **スタートアップ時のスクリプト自動化**：Raspberry Pi のスタートアップ時にスクリプトを自動的に実行する方法。
- **Robot HAT の概要**：Robot HAT の詳細な概要、その特徴と機能を含む。

このチュートリアルは、PiCar-X を微調整し、Robot HAT の技術的な側面を理解したい人にとって重要です。

ビデオ

関連するオンラインチュートリアル

- *0. PiCar-X* **の校正**

プロジェクト

セットアップに続いて、コースは 12 のプロジェクトベースのビデオに深く潜り込んでいます。これらのビデオは、基本的な動きから始まり、より複雑なタスクに至るまで、PiCar-X の機能を段階的に高めていきます。これには以下が含まれます：

5.4 ビデオ 1：モーターの動きとステアリングコントロール

このビデオは PiCar-X シリーズの最初のプロジェクトチュートリアルとして、PiCar-X のモーターとステアリングサーボの制御方法に焦点を当てています。主な内容には以下が含まれます：

- **モーターの起動と停止**：PiCar-X の動きを制御する方法を示し、前進と後退を含みます。
- **モータースピードコントロール**：モーターの速度を増減する方法を示します。
- **ステアリングコントロール**：前輪ステアリングサーボを制御して左右に曲がる方法を教えます。
- **スクリプト実行**：Raspberry Pi が起動するときにプログラムを自動的に実行する方法を説明します。
- **Move.py コード解析**：モーターとステアリングを制御する背後にあるロジックについて、Move.py スクリプトの詳細な説明を提供します。
- **テストとデバッグ**：コードとハードウェアが正しく機能していることを確認するための実践的なテスト。

このビデオは、PiCar-X 初心者にとって優れた実践ガイドであり、モーターやステアリングの制御に関する詳細な指示を提供し、今後のプロジェクト開発のための確かな基盤を築きます。

ビデオ

関連するオンラインチュートリアル

- *1. PiCar-X* **を動かす**

5.5 ビデオ 2：キーボードを使用した PiCar-X の制御

このビデオチュートリアルでは、キーボードを使用して PiCar-X ロボットを制御する方法を学びます。内容は以下の通りです：

- **基本的な制御**：PiCar-X をキーボードコマンドで制御する方法を示します - `W` は前進、 `S` は後退、 `A` は左折、 `D` は右折。
- **カメラの傾斜とパン**：`I`（上）、`K`（下）、`J`（左）、`L`（右）を使用して、取り付けられたカメラの傾斜とパンを制御する方法を教えます。
- **コードの説明**：キーボード制御のための Python コードについて、ライブラリのインポートや制御ロジックを含め、詳細な説明を提供します。
- **コードの実行**：PiCar-X に接続することを含め、キーボード制御のための Python コードを実行する方法を示します。
- **制御の終了**： `Ctrl + C` を押すことでキーボード制御を終了し、ロボットをデフォルト状態に戻す方法を説明します。

このチュートリアルは、明確な指示とキーボードで PiCar-X ロボットを制御する実践的なデモンストレーションを提供するため、初心者やロボティクス愛好家に理想的です。

ビデオ

関連するオンラインチュートリアル

- 2. **キーボード制御**

5.6 ビデオ 3：テキスト・トゥ・スピーチ

このチュートリアルでは、PiCar-X ロボットのテキスト・トゥ・スピーチ機能について説明します：

- **ロボットのセットアップと初期動作**：ロボットの準備と初期動作を示します。これには停止や障害物検出が含まれます。
- **テキスト・トゥ・スピーチ機能**：ロボットがテキスト・トゥ・スピーチ技術を使用して事前定義されたフレーズを話したり、カウントダウンする様子を示します。
- **カスタムスクリプトのデモンストレーション**：ロボットが話したり、動いたり、障害物に反応したり、U ターンをしたりするカスタムスクリプトを実行します。
- **音楽の再生**：Python スクリプトを使用してロボットで音楽ファイルを再生する方法を教えます。

このレッスンは、PiCar-X ロボットにテキスト・トゥ・スピーチ機能を統合する方法について、実践的なデモンストレーションやコードの詳細を含む深いチュートリアルを提供します。

ビデオ

関連するオンラインチュートリアル

- *3.* **テキストから音声へ & 効果音**

5.7 ビデオ 4：超音波を使用した障害物回避

このビデオチュートリアルでは、PiCar-X ロボットにおける超音波センサーを使用した障害物回避について説明します：

- **超音波センサーの紹介**：距離を測定し、車の動きを制御するために超音波センサーを使う方法を説明します。
- **Python コードの詳細解説**：障害物回避のための Python コードについて、変数定義や条件付き動作ロジックを詳細に説明します。
- **カスタムスクリプトの作成と実行**：U ターンや障害物反応のためのカスタムスクリプトの作成と実行方法を示します。
- **実践的デモンストレーション**：異なる表面でのロボットの障害物回避能力をデモンストレーションします。
- **コードの保存と更新**：ロボット上でスクリプトを保存し更新する方法を説明します。

このレッスンは、PiCar-X ロボットで超音波センサーを基にした障害物回避を実装するための必須ガイドを提供します。

ビデオ

関連するオンラインチュートリアル

- *4.* **障害物回避**

5.8 ビデオ 5：グレースケール線追跡

このビデオチュートリアルでは、PiCar-X ロボットを使用したグレースケール線追跡について探求します：

- **線の検出**：グレースケールセンサーを使用して、ロボットが白い表面上の黒い線を検出し、線に沿って追跡または運転する方法を実演します。
- **Python コードの説明**：モジュールのインポート、オブジェクトの作成、線検出に応じて応答するロジックなど、線追跡に関連する Python コードについて説明します。
- **実践的デモンストレーション**：ロボットが異なる表面上で線を検出し、追跡するデモンストレーションを提供します。

- **トラブルシューティングと調整**：より良い線追跡パフォーマンスのためにコードを調整し修正する必要性について議論します。

このレッスンは、実践的なデモンストレーション、コードのウォークスルー、トラブルシューティングのヒントを含む、PiCar-X でのグレースケール線追跡の実装に関する包括的なガイドを提供します。

ビデオ

関連するオンラインチュートリアル

- *5.* **ライン追跡**

5.9 ビデオ 6：崖検出

このチュートリアルでは、PiCar-X ロボットで崖検出をプログラミングし活用するための重要な洞察を提供します。

- **導入**：表面反射を測定することで PiCar-X のグレースケールセンサーを使用した崖検出について説明します。
- **Python コードの詳細解説**：センサーの設定とプログラムロジックを詳述した崖検出のためのコードについてカバーします。
- **デモンストレーションとテスト**：ロボットが崖を検出し反応する様子を示し、コードの実行とテストの手順を説明します。
- **自動起動設定**：Raspberry Pi を設定して、起動時に崖検出スクリプトを実行する方法をガイドします。
- **実践的応用**：ロボットが異なる表面とエッジにどのように反応するかを示し、その崖検出能力を強調します。

ビデオ

関連するオンラインチュートリアル

- *6.* **崖検出**

5.10 ビデオ 7：PiCar-X コンピュータビジョン

このビデオチュートリアルは、PiCar-X のコンピュータビジョン機能に焦点を当てています：

- **コンピュータビジョンへの導入**：カメラを装備した PiCar-X を使用して、手の動き、指、色、顔を検出し、QR コードを読む方法を教えます。
- **リモートデスクトップと Python コードの実行**：VNC を使用して Raspberry Pi へのリモートデスクトップアクセスとコンピュータビジョンタスクのための Python コードの実行をデモンストレーションします。
- **色検出と写真撮影**：カメラコントロールを使用して異なる色を検出し写真を撮る方法を示します。

- **QR コードと顔検出**：QR コード読取りと顔検出機能の間で切り替える方法を説明します。
- **物体検出**：SunFounder の VI ライブラリからの組み込み機能を使用した物体検出について議論します。
- **ブラウザとモバイルでのビデオ視聴**：カメラフィードをブラウザや携帯電話にストリーミングする方法を教え、それらが PiCar-X と同じネットワークに接続されていることを確認します。
- **手検出**：VI ライブラリの手検出機能の使用と、対応する Python コードの実行についてカバーします。
- **コードの編集と実行**：さまざまなコンピュータビジョンタスクのための Python スクリプトを作成、編集、実行する方法をデモンストレーションします。

このレッスンでは、PiCar-X のコンピュータビジョン機能の探求を含む、色検出、QR コード読取り、顔検出、手の動き追跡の実践的デモンストレーションを提供する包括的なガイドを提供します。

ビデオ

関連するオンラインチュートリアル

- *7.* **コンピュータービジョン**

5.11 ビデオ 8：PiCar-X があなたをじっと見る

このチュートリアルでは、PiCar-X ロボットでカメラとサーボを使用してオブジェクト追跡を行う方法を教えます：

- **カメラ-サーボの統合**：動き追跡のためのカメラとサーボの接続をデモンストレーションします。
- **Python コードの概要**：PiCar-X のカメラ動作を制御するコードについて説明します。
- **カメラの起動とビデオ表示**：カメラを起動し、ビデオフィードを表示する方法を示します。
- **顔追跡**：ロボットが人間の顔を追跡し、カメラの角度を調整する方法の詳細を説明します。
- **デモンストレーション**：ロボットのカメラが動きに追随し調整するデモンストレーションを含みます。
- **ビデオストリーミング**：カメラフィードをブラウザや携帯電話にストリーミングする方法についてカバーします。

このレッスンでは、PiCar-X がそのカメラを使用してオブジェクトや顔に追随し焦点を合わせるための簡潔なガイドを提供します。

ビデオ

関連するオンラインチュートリアル

- *8.* **あなたを見つめる**

5.12 ビデオ 9：ビデオ録画

このチュートリアルでは、PiCar-X ロボットを使用してビデオを録画し管理する方法について、簡潔かつ詳細なガイドを提供します。

- **概要**：Raspberry Pi 自走型ロボットカーキットである PiCar-X を使用して、HD 1080p のビデオを録画する方法を教えます。
- **ドキュメントとコード**：ビデオ録画のための PiCar-X ドキュメントを案内し、録画コントロール（開始、一時停止、継続、停止）の Python スクリプトについて説明します。
- **リモートデスクトップ接続**：ビデオ録画のために Raspberry Pi にリモートで接続する方法をデモンストレーションします。
- **実践的録画デモンストレーション**：ビデオ録画スクリプトを実行し、リモートデスクトップを通じて操作する方法を示します。
- **ビデオ再生**：録画されたビデオを見つけて再生し、品質を確認する方法を説明します。
- **追加機能**：タイムラプスや簡単なキャプチャなどのより多くの録画オプションを提供する PiCamera2 GitHub リポジトリを紹介します。
- **シンプルなビデオスクリプト**：異なるフォーマットや設定で基本的なビデオ録画スクリプトを作成し実行する方法を強調します。

ビデオ

関連するオンラインチュートリアル

- *9.* **ビデオ録画**

5.13 ビデオ 10：PiCar-X でのブルファイト

このチュートリアルでは、カラー検出と動きに焦点を当てた「ブルファイト」ゲーム用の PiCar-X ロボットの使用について説明します：

- **ブルファイトのコンセプト**：PiCar-X のカメラを使用して赤色を追跡し、パンとチルトの制御を行います。
- **Python コードの概要**：色検出とロボットの動きの背後にあるプログラミングについて詳述します。
- **リモートデスクトップアクセス**：リモートデスクトップを介して PiCar-X のコードを管理する方法をデモンストレーションします。
- **カラートラッキングのメカニクス**：ロボットが赤い物体をどのように追跡し、それに応じて動きを調整するかを説明します。
- **デモンストレーション**：赤いターゲットを追求する PiCar-X のアクションを展示します。

このレッスンでは、カラートラッキングゲーム用に PiCar-X をプログラムする方法について、コードの説明とライブデモンストレーションを含む包括的なガイドを提供します。

ビデオ

関連するオンラインチュートリアル

- *10.* **ブルファイト**

5.14 ビデオ 11：ビデオカーとしての PiCar-X

このチュートリアルでは、PiCar-X をカメラ制御付きのビデオカーとして使用する方法を教えます：

- **制御メカニクス**：動きのためのキーボードコントロールを紹介します - 前進、後退、左、右、停止。
- **ビデオカーの機能**：PiCar-X を使用して運転し、速度調整、ターン、写真撮影に焦点を当てます。
- **Python コードの概要**：車とカメラを制御するための Python コードについて簡単に説明します。
- **リモートデスクトップとプログラム設定**：リモートデスクトップを介して車のプログラムをセットアップし、スタートアップ時に実行する方法を示します。
- **ライブデモンストレーション**：キーボード入力で PiCar-X を制御し、そのカメラを使用するデモンストレーションを含みます。
- **写真撮影と閲覧**：PiCar-X で撮影した写真をキャプチャし、閲覧する方法を教えます。

このレッスンでは、PiCar-X をビデオカーとして操作する方法について簡潔な概要を提供し、実践的な制御とカメラの使用に重点を置いています。

ビデオ

関連するオンラインチュートリアル

- *11.* **ビデオカー**

5.15 ビデオ 12：トレジャーハントゲーム

このチュートリアルは、PiCar-X を使用したプログラミングとロボティクスに関する魅力的で教育的な体験を提供します。

- **概要**：PiCar-X ロボットがランダムに選ばれた色を検出し、その方向へと進むトレジャーハントゲームをデモンストレーションします。
- **Python コード**：色検出、ロボットの動き制御、テキスト・トゥ・スピーチでのアナウンスに使用される Python コードの詳細な説明。

- **ゲームプレイ**：キーボード入力を使用してロボットを動かし、テキスト・トゥ・スピーチでアナウンスされるターゲット色を見つけて到達します。
- **実践的デモ**：赤、黄色、青などの異なる色を正確に識別し、向かっていくロボットのアクションを示します。
- **ゲーム終了の指示**：ロボットの停止とカメラのシャットダウンを含む、ゲームを安全に終了する方法について説明します。

ビデオ

関連するオンラインチュートリアル

- *12.* **宝探し**

第 6 章

Ezblock を楽しむ

初心者やビギナーのために、EzBlock は SunFounder が Raspberry Pi 用に提供しているソフトウェア開発プラットフォームです。EzBlock は、グラフィカル環境と Python 環境の 2 つのプログラミング環境を提供しています。

ほとんど全てのデバイスタイプ、Mac、PC、Android を含む、で利用可能です。

EzBlock のインストール、ダウンロード、使用を完了するためのチュートリアルは以下です。

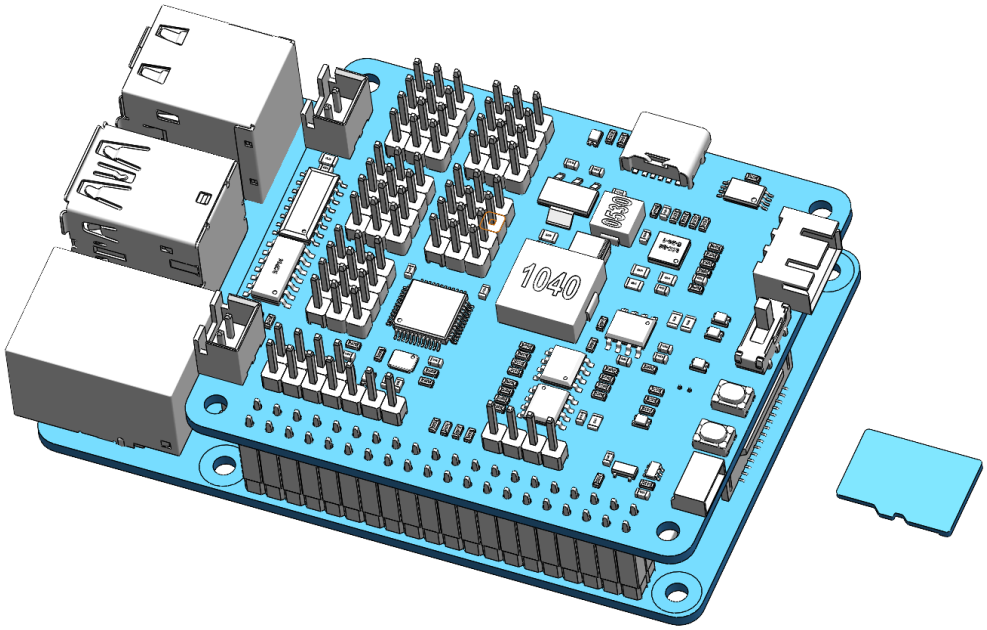
6.1 EzBlock のクイックガイド

サーボの角度範囲は-90〜90 度ですが、工場での設定角度はランダムで、0 ° であるかもしれませんし、45 ° であるかもしれません。このような角度でそのまま組み立てると、ロボットがコードを実行した後に混乱した状態になるか、もっと悪い場合はサーボがブロックして焼け出す原因となります。

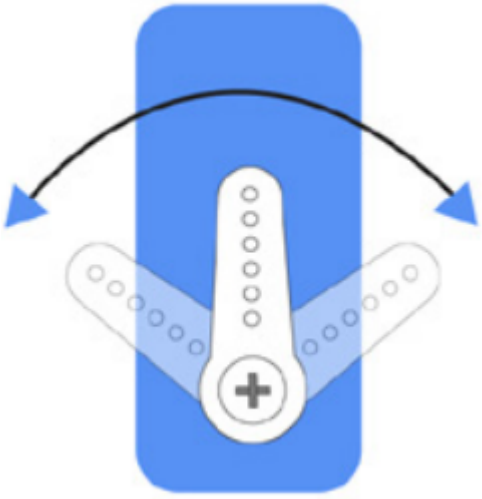
したがって、すべてのサーボの角度を 0 ° に設定してから取り付ける必要があります。これにより、サーボの角度が中央になり、どちらの方向に回転しても問題ありません。

1. まず、Install EzBlock OS (EzBlock の公式チュートリアル) を Micro SD カードにインストールしてください。インストールが完了したら、Raspberry Pi に挿入してください。

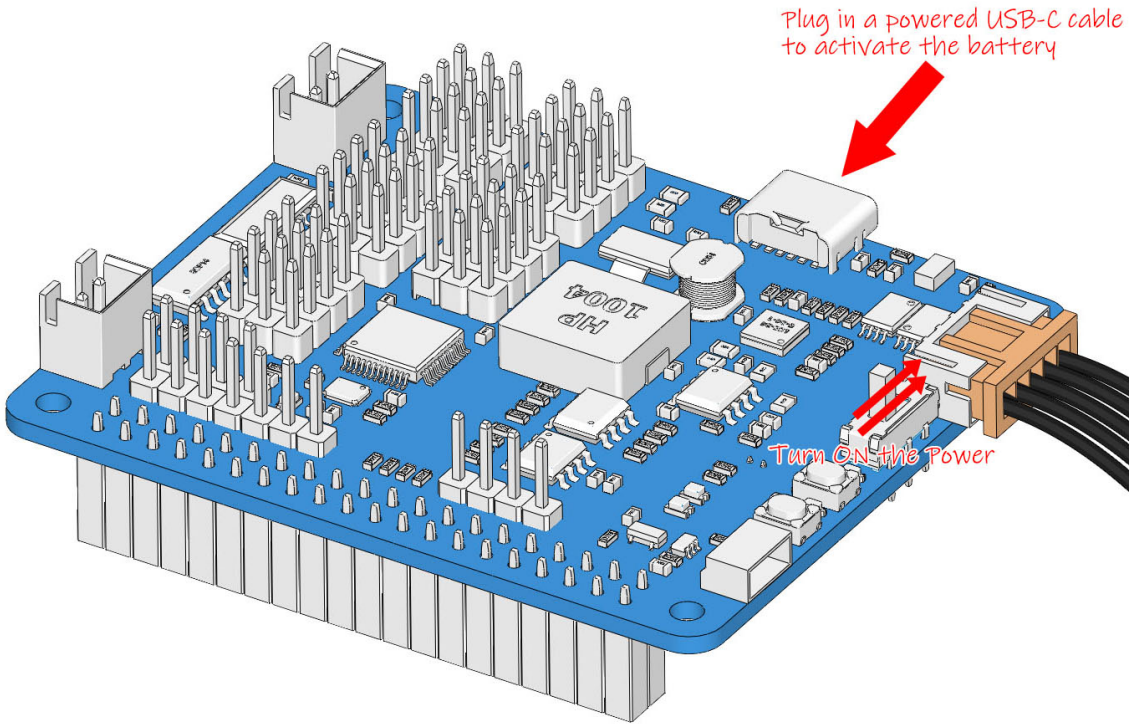
注釈: インストールが完了したら、このページに戻ってください。



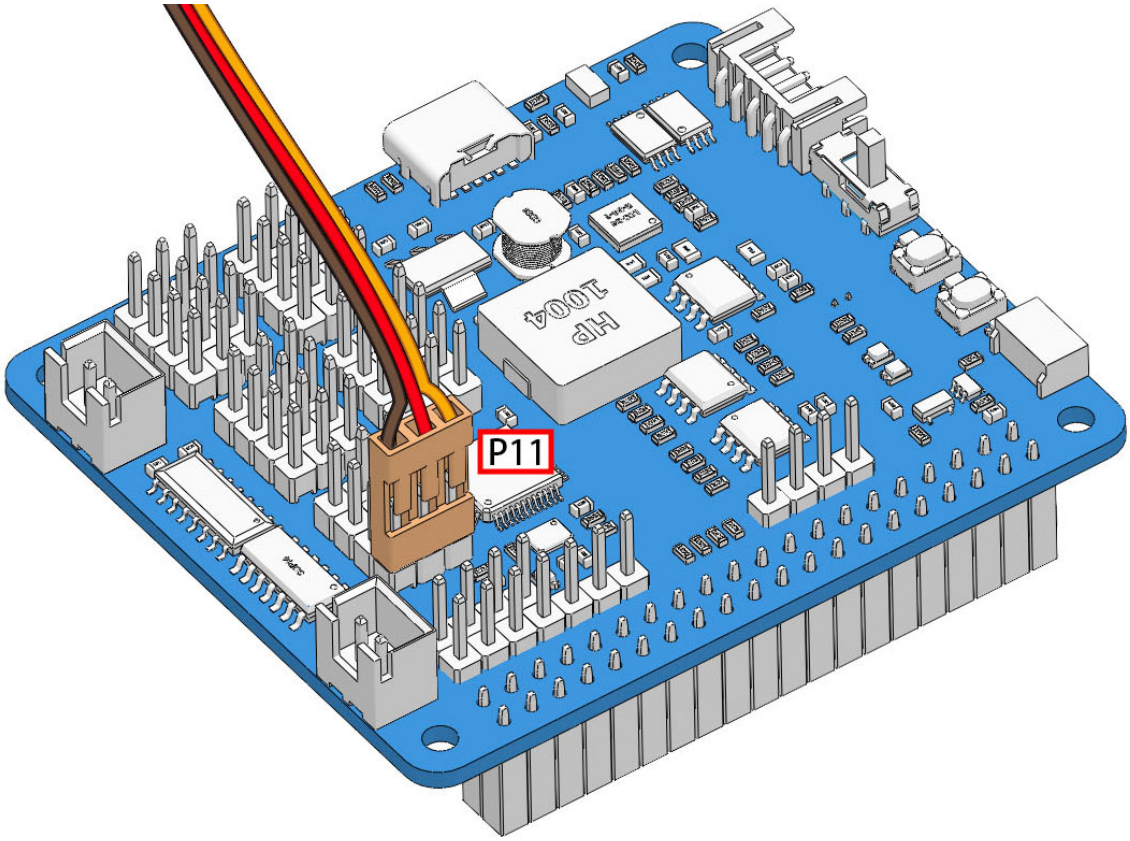
2. サーボが正確に 0 ° に設定されていることを確認するため、まずサーボアームをサーボシャフトに挿入し、ロッカーアームを別の角度にゆっくりと回転させます。このサーボアームは、サーボが回転していることをはっきりと確認するためのものです。



3. 組み立ての説明書に従い、バッテリーケーブルを挿入し、電源スイッチを ON にします。その後、電源を供給する USB-C ケーブルを挿入して、バッテリーをアクティブ化します。1-2 分待つと、Raspberry Pi が正常に起動したことを示す音がします。



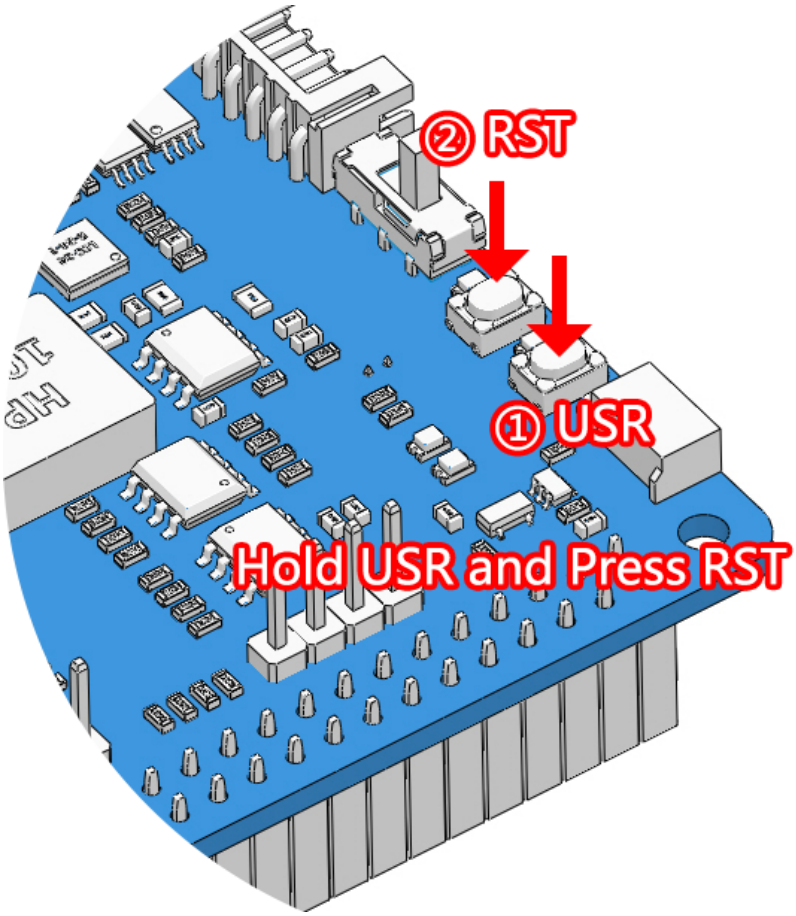
4. 次に、サーボケーブルを以下のように P11 ポートに接続します。



5. **USR** キーを押し続け、 **RST** キーを押すと、システム内のサーボゼロリングスクリプトが実行されます。

サーボアームが位置に回転するのを見たとき（これは 0 ° の位置であり、ランダムな位置であり、垂直または平行であるとは限りません）、それはプログラムが実行されたことを示しています。

注釈: この手順は一度だけ実行する必要があります。その後、他のサーボワイヤーを挿入するだけで、自動的にゼロになります。



6. さて、サーボアームを取り外し、サーボワイヤーが接続されたままにし、電源を切らないでください。その後、紙の組み立て説明書に従って組み立てを続けてください。

注釈:

- サーボをサーボネジで固定する前にこのサーボケーブルを抜かないでください。固定した後に抜くことができます。
- 電源が入っている状態でサーボを回転させないでください。ダメージの原因となります。もしサーボシャフトが間違った角度で挿入されていた場合、サーボを引き抜いて再度挿入してください。
- 各サーボを組み立てる前に、サーボケーブルを P11 に挿入し、電源を入れてその角度を 0 ° に設定する必要があります。

- あとで EzBlock APP でロボットにプログラムをダウンロードすると、このゼロリング機能は無効になります。

6.2 EzBlock Studio のインストールと設定

ロボットが組み立てられたら、基本的な操作を行う必要があります。

- Install EzBlock Studio：デバイスに EzBlock Studio をダウンロードしてインストールするか、Web ベースのバージョンを使用します。
- Connect the Product and EzBlock：Wi-Fi、Bluetooth を設定し、使用前にキャリブレーションを行います。
- Open and Run Examples：関連する例を直接表示または実行します。

注釈: PiCar-X に接続すると、キャリブレーションステップがあります。これは、取り付けプロセスの可能な偏差やサーボ自体の制限により、一部のサーボ角度がわずかに傾くことがあるためです。そのため、このステップでそれらをキャリブレーションできます。

しかし、組み立てが完璧でキャリブレーションが不要だと思う場合、このステップをスキップすることもできます。

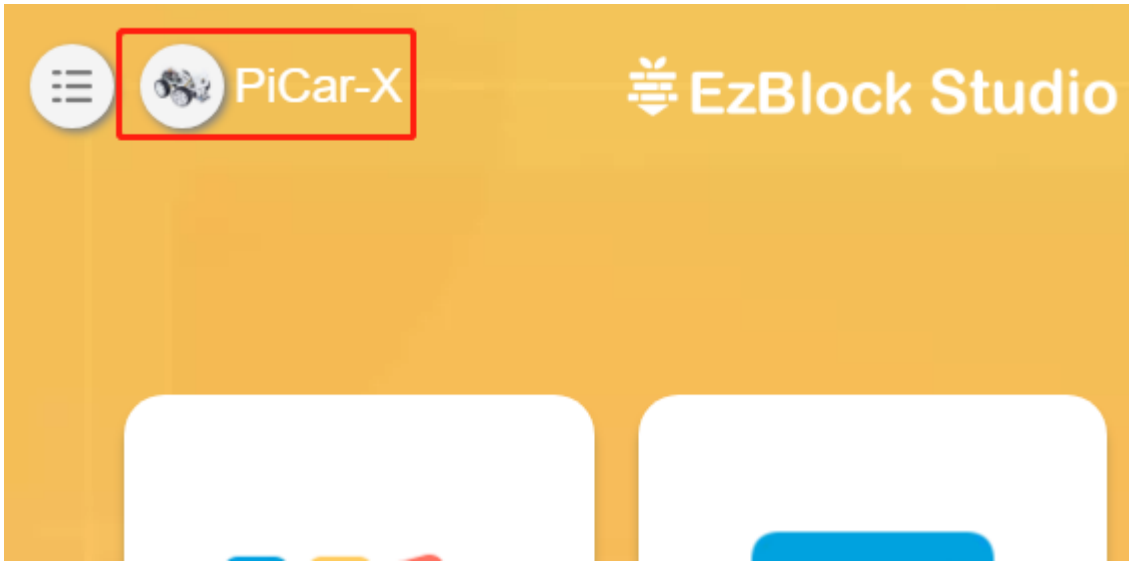
6.3 車のキャリブレーション

PiCar-X を接続した後、キャリブレーションの手順があります。これは、取り付けの過程での微妙なずれやサーボ自体の制約により、一部のサーボの角度が若干傾いている可能性があるためです。この手順でそれらを調整することができます。

しかし、組み立てが完璧でキャリブレーションが不要と感じる場合、この手順をスキップすることもできます。

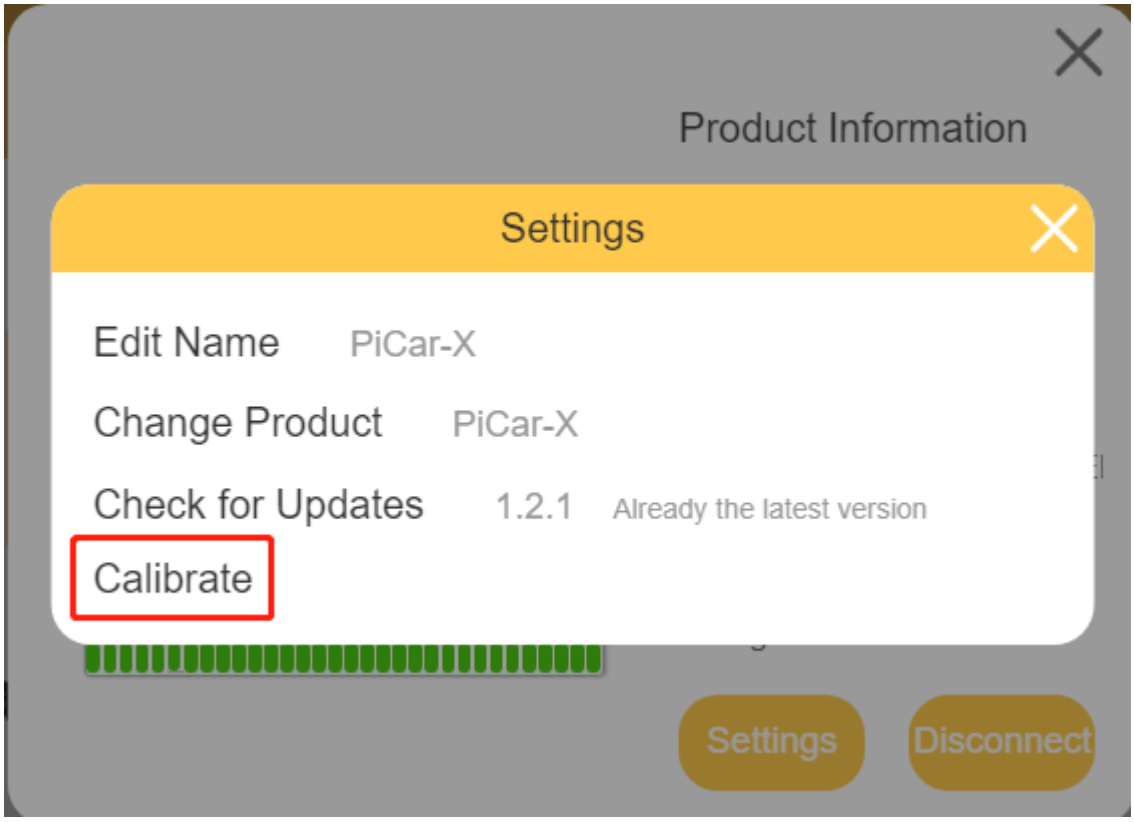
注釈: 使用中にロボットのキャリブレーションを再度行いたい場合は、以下の手順に従ってください。

1. 左上の接続アイコンをクリックして製品詳細ページを開きます。



2. **設定** ボタンをクリックします。

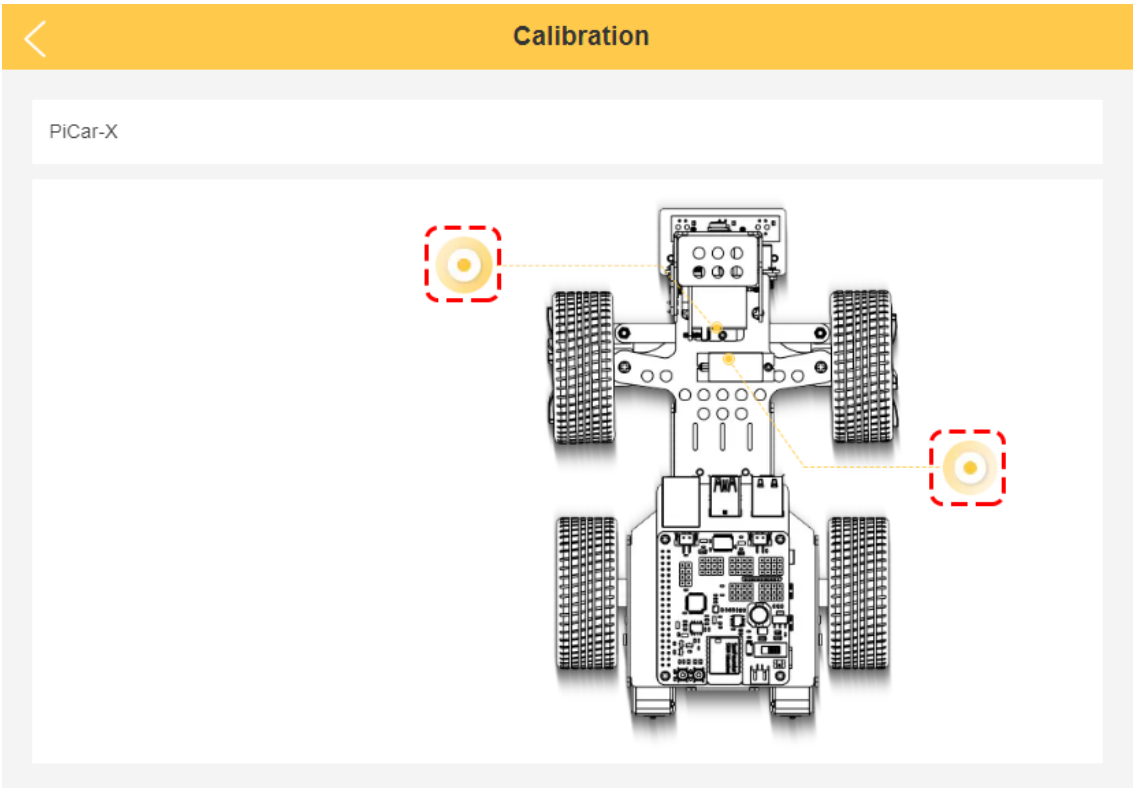
3. このページでは、製品名や製品タイプの変更、アプリのバージョンの確認、またはロボットのキャリブレーションができます。 **キャリブレート** をクリックすると、キャリブレーションページに移動します。



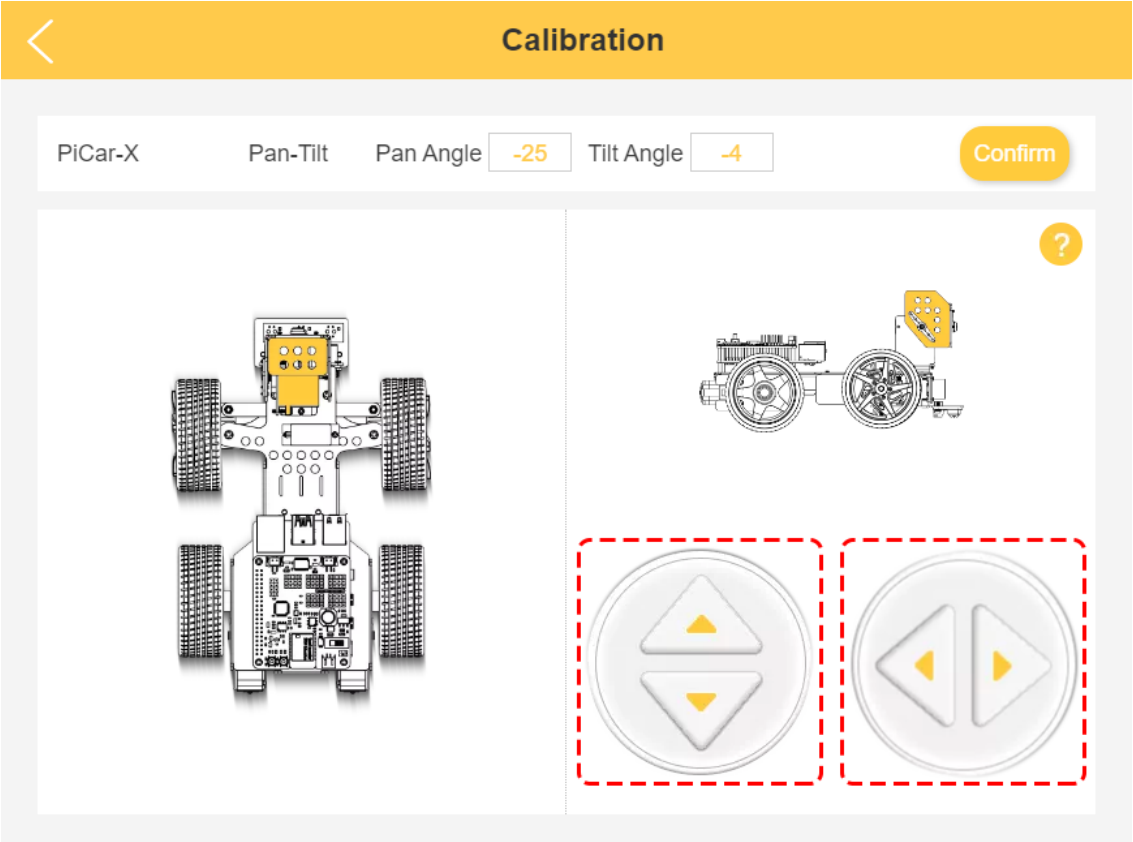
キャリブレーションの手順は次のとおりです：

1. キャリブレーションページに移動すると、キャリブレーションを行う場所を指示する 2 つのプロンプトポイントが表示されます。

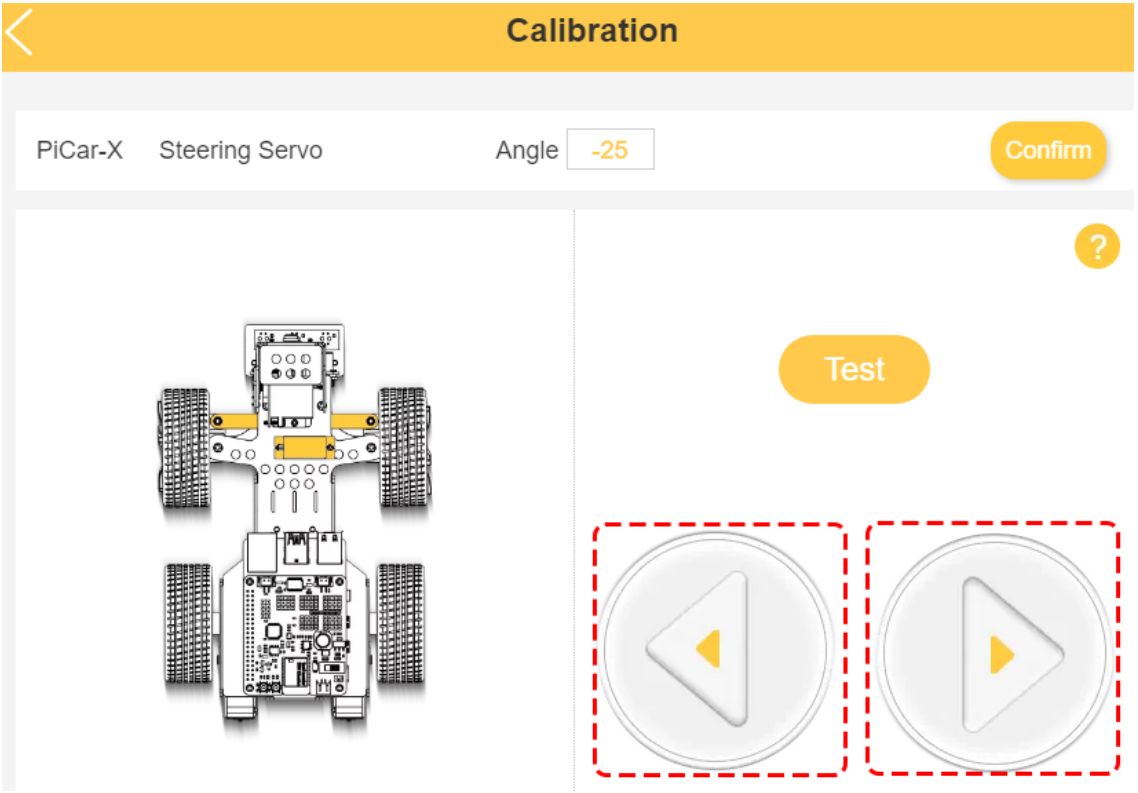
> **注釈:** キャリブレーションは微調整の過程です。ボタンを限界までクリックしても部品がまだ合っていない場合、部品を取り外し、再度組み立てることをおすすめします。



2. 左のプロンプトポイントをクリックして、PiCar-X のパン・チルト（カメラ部分）をキャリブレートします。右側の 2 組のボタンを使用して、パン・チルトの向きをゆっくりと調整し、その角度を確認することができます。調整が完了したら、 **確認** をクリックします。



3. 前輪の方向をキャリブレートするには、右のプロンプトポイントをクリックします。右側の 2 つのボタンを使用して、前輪が正面を向くように調整します。調整が終わったら、 **確認** をクリックします。

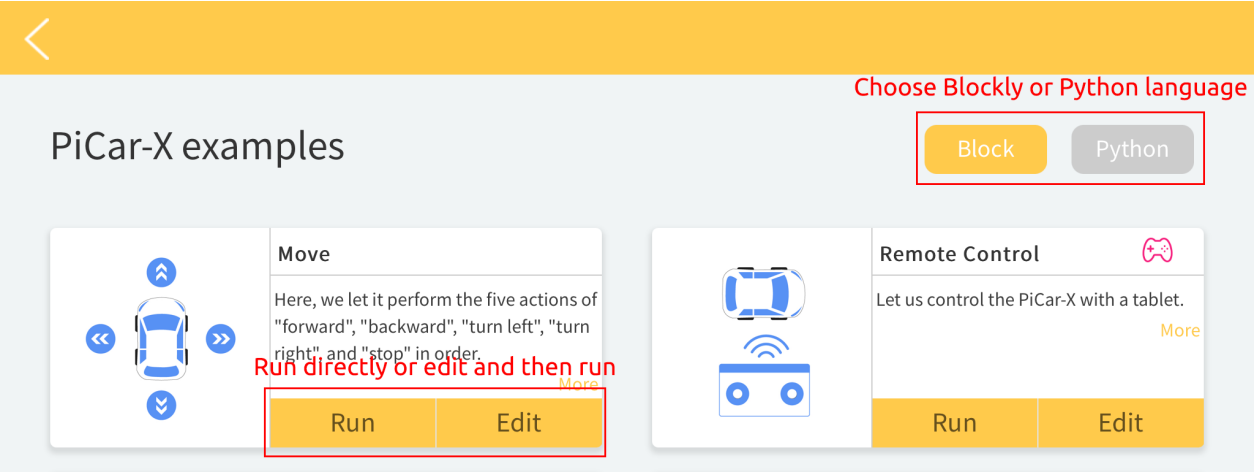


プロジェクト

このセクションは、PiCar-X の基本的なプログラミング機能から始まり、Ezblock Studio でより高度なプログラムを作成するまで続きます。各チュートリアルには新しい関数を紹介する TIPS が含まれており、ユーザーは対応するプログラムを記述することができます。例のセクションには、直接使用できる完全な参照コードもあります。例のセクションのコードを使用せずにプログラミングを試み、チャレンジを克服する楽しさを味わってください！

Ezblock のプロジェクトは全て Ezblock Studio の例ページにアップロードされています。例のページから、ユーザーはプログラムを直接実行することも、例を編集してユーザーの「My Projects」フォルダに保存することもできます。

例ページでは、Block 言語と Python 言語のどちらを選択するかをユーザーに選ばせます。このセクションのプロジェクトは Block 言語のみを説明していますので、Python コードの説明については、この ファイル を参照して、Python コードを理解してください。



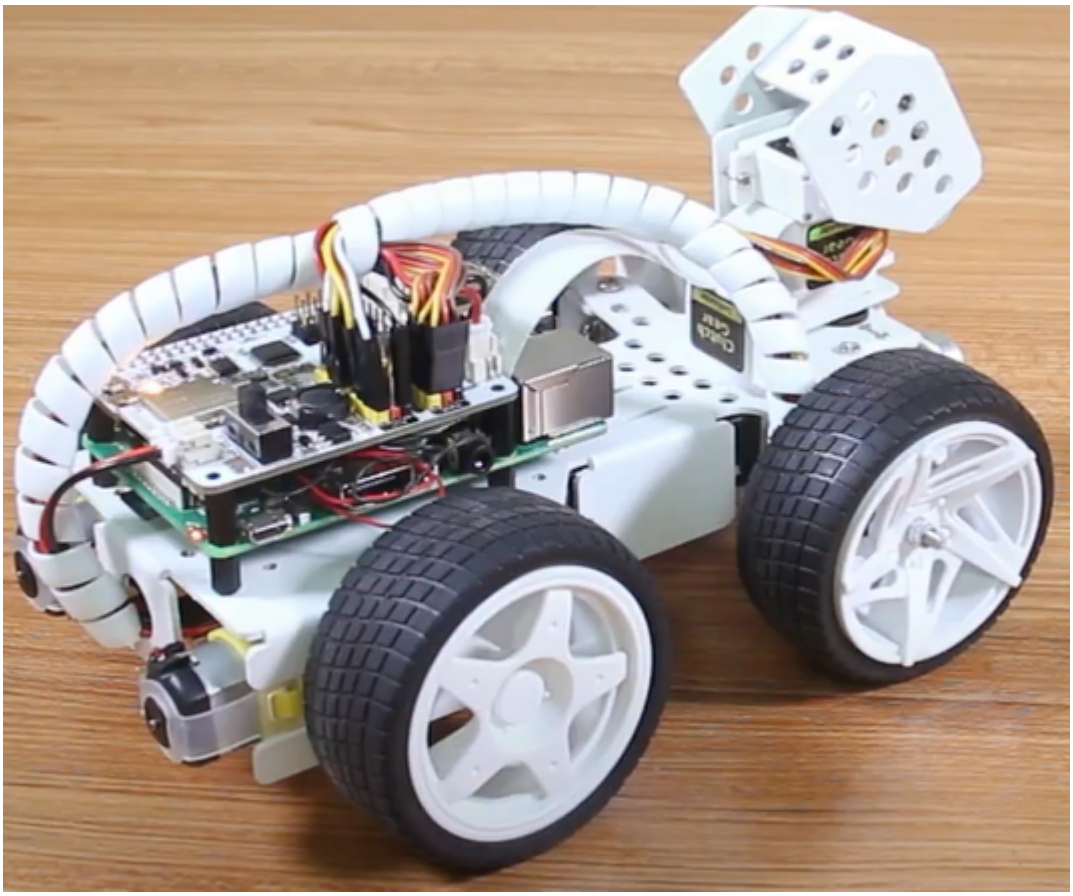
基本

6.4 移動

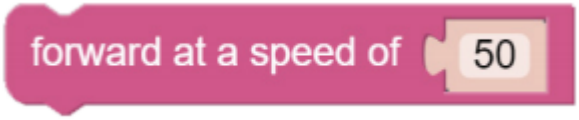
この初めてのプロジェクトでは、PiCar-X の動作をプログラムする方法を学びます。このプロジェクトでは、プログラムが PiCar-X に順番に「前進」「後退」「左折」「右折」「停止」の 5 つのアクションを実行するよう指示します。

Ezblock Studio の基本的な使い方を学ぶには、以下の 2 つのセクションを読み進めてください：

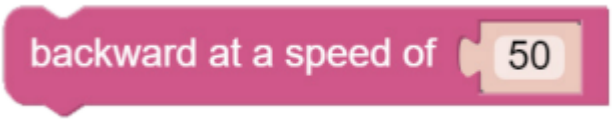
- How to Create a New Project?



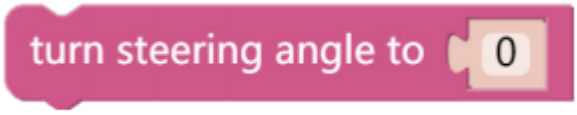
TIPS



このブロックを使うと、PiCar-X が利用可能なパワーのパーセンテージに基づいて前進します。以下の例では「50」はパワーの 50 %、つまり半速を意味します。



このブロックは、PiCar-X が利用可能なパワーのパーセンテージに基づいて後退します。



このブロックで前輪の向きを調整します。範囲は「-45」から「45」です。下の例の「-30」は、車輪が左に 30 ° 回

転することを意味します。

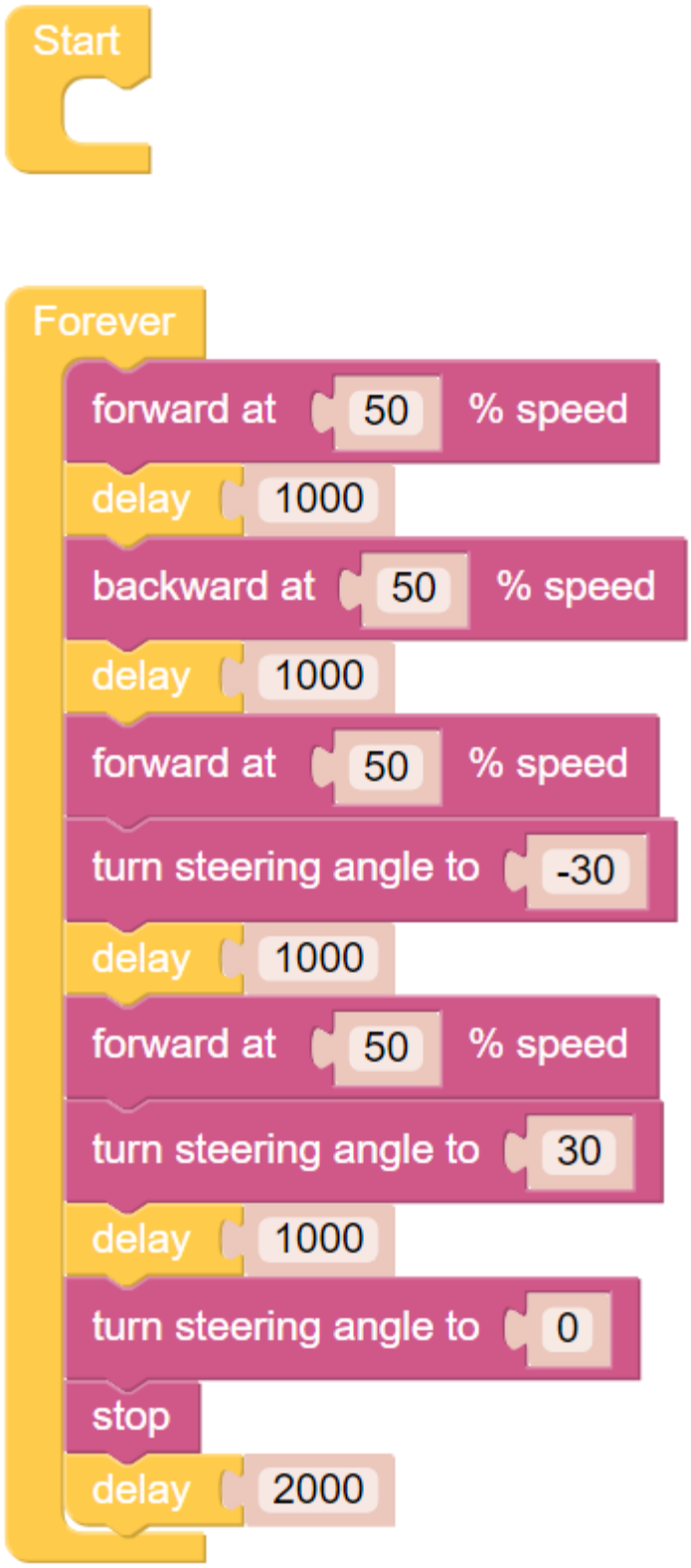
このブロックは、ミリ秒を基にコマンド間の時間的なブレークを引き起こします。以下の例では、PiCar-X は次のコマンドを実行する前に 1 秒（1000 ミリ秒）待機します。

このブロックで PiCar-X を完全に停止させます。

例

注釈:

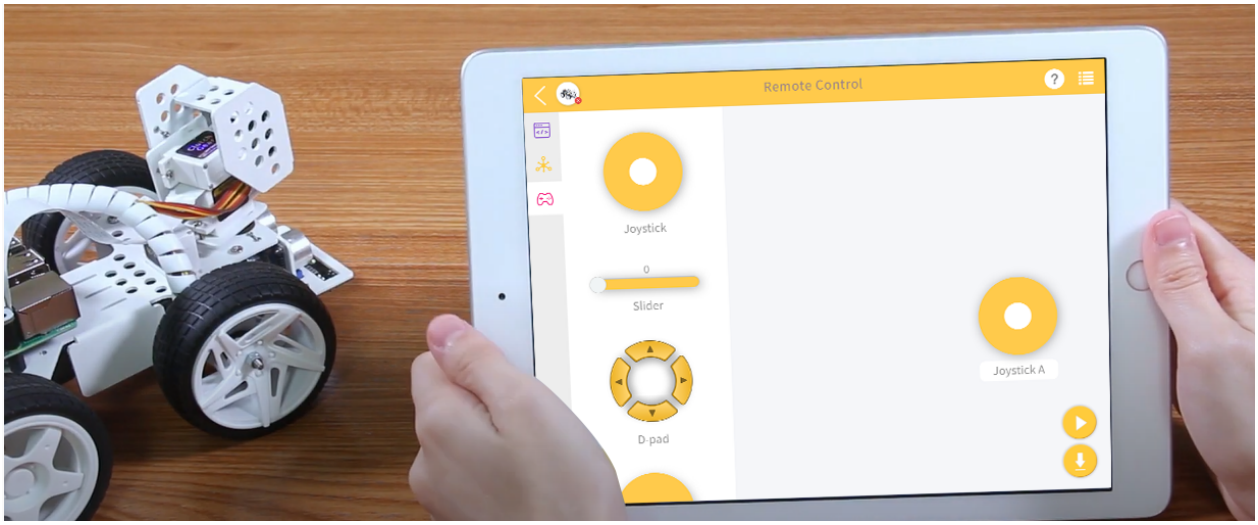
- 以下の画像に従ってプログラムを書くことができます。チュートリアルを参照してください: How to Create a New Project?。
- EzBlock Studio の **Examples** ページで同じ名前のコードを見つけ、 **Run** または **Edit** を直接クリックしてください。


Start
Forever
forward at 50 % speed
delay 1000
backward at 50 % speed
delay 1000
forward at 50 % speed
turn steering angle to -30
delay 1000
forward at 50 % speed
turn steering angle to 30
delay 1000
turn steering angle to 0
stop
delay 2000

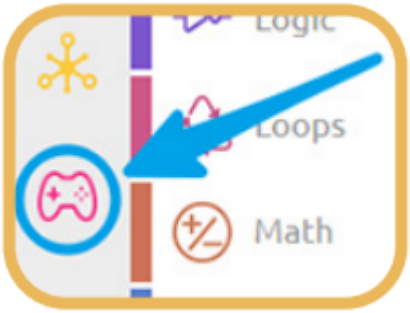
6.5 リモートコントロール

このプロジェクトでは、ジョイスティックウィジェットを使用して PiCar-X をリモートで制御する方法を学びます。注意: リモートコントロールページからジョイスティックウィジェットをドラッグアンドドロップした後、"Map"機能を使用してジョイスティックの X 軸と Y 軸の読み取りをキャリブレーションします。リモートコントロール機能の詳細については、以下のリンクを参照してください：

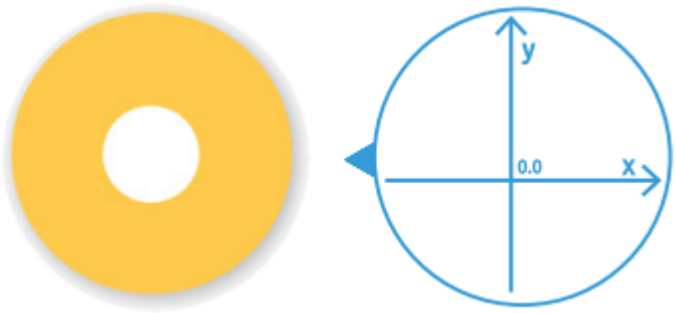
- How to Use the Remote Control Function?



TIPS

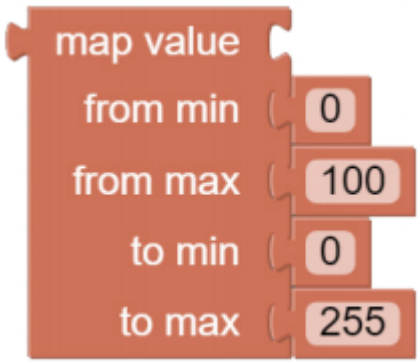


リモートコントロール機能を使用するには、メインページの左側からリモートコントロールページを開きます。



ジョイスティックをリモートコントロールページの中央領域にドラッグします。中央の白い点をトグルして、任意の方向にゆっくりとドラッグすると、(X,Y) 座標が生成されます。X 軸または Y 軸の範囲はデフォルトで"-100"から"100"に設定されています。ジョイスティックの最も左側に直接白い点をドラッグすると、X の値が"-100"、Y の値が"0"となります。

リモートコントロールページにウィジェットをドラッグアンドドロップすると、上記のブロックを持つ新しいカテゴリ-リモートが表示されます。このブロックは、リモートコントロールページでのジョイスティック値を読み取ります。ドロップダウンメニューをクリックして、Y 軸の読み取りに切り替えることができます。



マップ値ブロックは、ある範囲から別の範囲に数字をリマップすることができます。範囲が 0 から 100 に設定されており、マップ値が 50 の場合、それは範囲の 50% の位置、すなわち"50"です。範囲が 0 から 255 に設定されており、マップ値が 50 の場合、それは範囲の 50% の位置、すなわち"127.5"です。

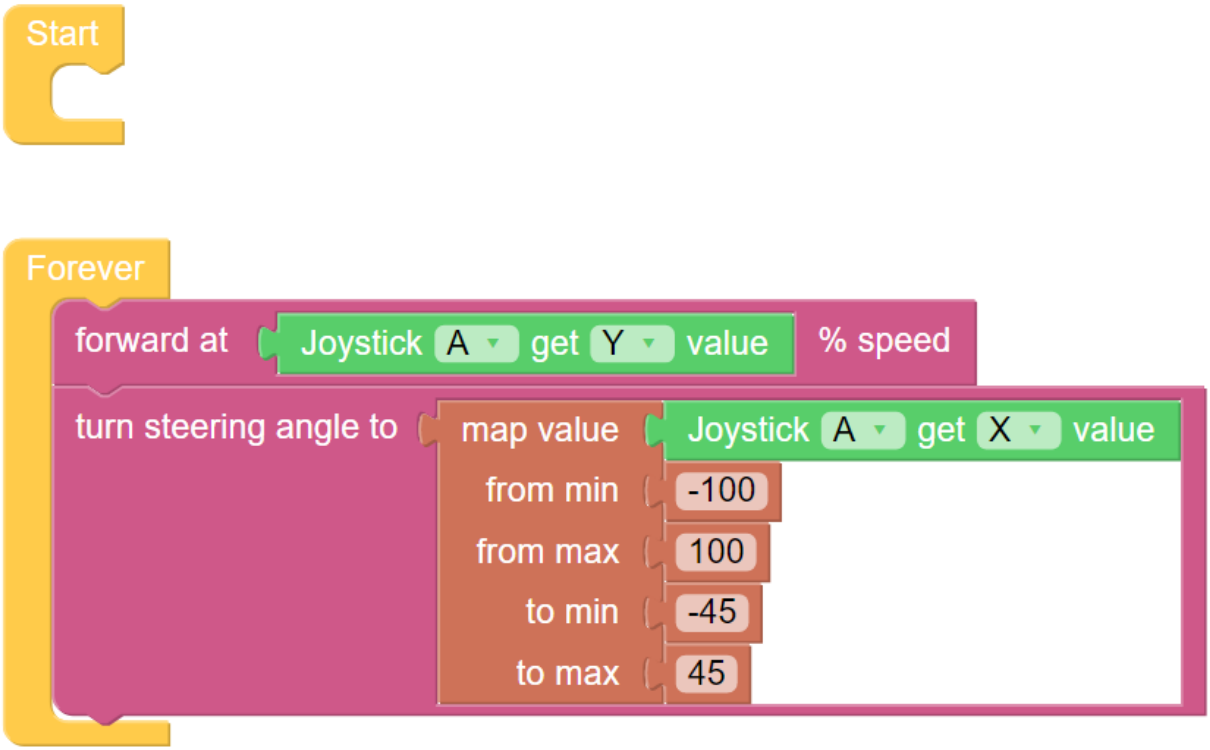
例

注釈:

- 以下の画像に従ってプログラムを書くことができます。チュートリアルを参照してください: How to Create

a New Project?。

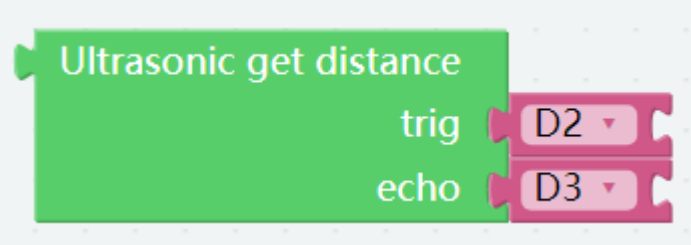
- EzBlock Studio の **Examples** ページで同じ名前のコードを見つけ、 **Run** または **Edit** を直接クリックしてください。



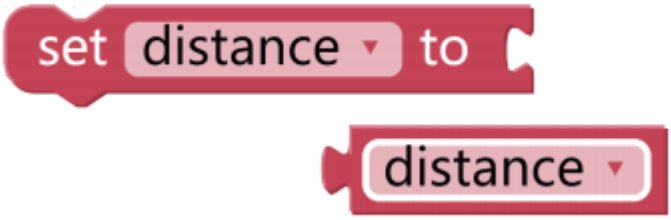
6.6 超音波モジュールのテスト

PiCar-X には、障害物回避や自動的な物体追尾の実験に使用できる組み込みの超音波センサーモジュールがあります。このレッスンでは、モジュールがセンチメートルでの距離（24 cm = 1 inch）を読み取り、結果を **Debug** ウィンドウで **Print** します。

TIPS



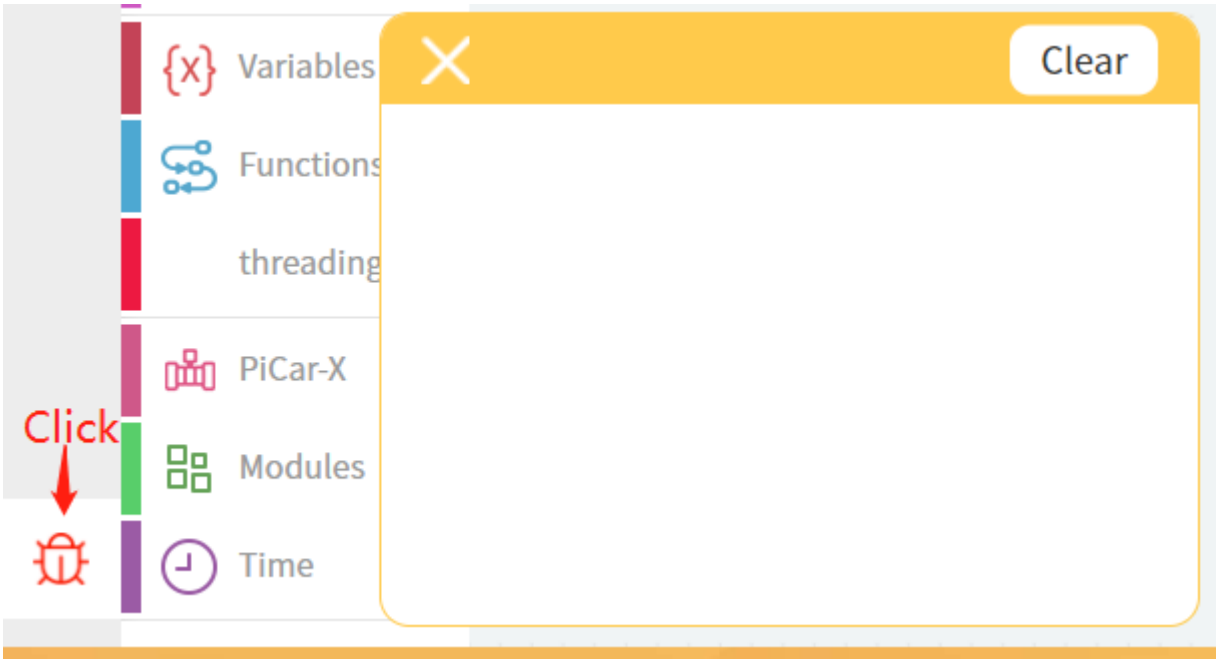
Ultrasonic get distance ブロックは、PiCar-X から直前の障害物までの距離を読み取ります。



このプログラムは **Variable** を使用して簡略化されています。例えば、プログラム内に複数の関数があり、それぞれが障害物までの距離を使用する必要がある場合、各関数が別々に同じ値を読み取るのではなく、 **Variable** を使用して各関数に同じ距離値を報告することができます。

Variables カテゴリの **Create variable...** ボタンをクリックし、ドロップダウン矢印を使用して " distance " という名前の変数を選択します。

Print 関数は、デバッグのために変数やテキストなどのデータを印刷することができます。

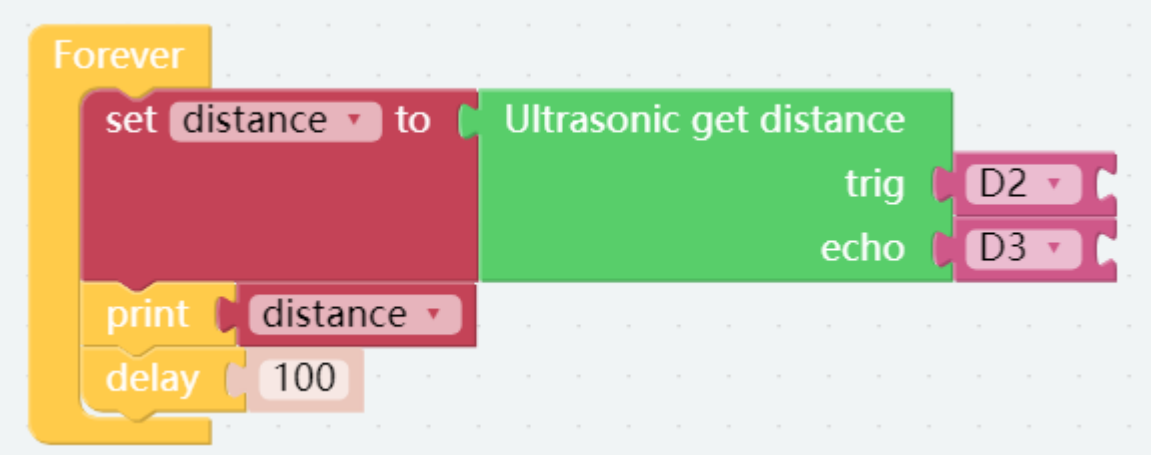


コードが実行中の場合、左下の角にある**Debug**アイコンをクリックしてデバッグモニターを有効にします。

例

注釈:

- 以下の画像に従ってプログラムを書くことができます。チュートリアルを参照してください: How to Create a New Project?。
- EzBlock Studio の **Examples** ページで同じ名前のコードを見つけ、 **Run** または **Edit** を直接クリックしてください。

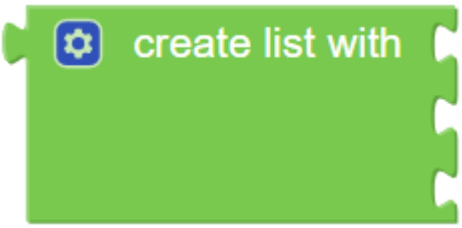


6.7 グレースケールモジュールのテスト

PiCar-X には、ラインフォロー、崖検出、および他の楽しい実験を実装するためのグレースケールモジュールが含まれています。グレースケールモジュールには、センサーが検出した色の濃度に応じて値を報告する 3 つの検出センサーがあります。例えば、純粋な黒の濃度を読み取るセンサーは “ 0 ” の値を返します。

TIPS

Grayscale module ブロックを使用して、センサーの値を読み取ります。上の例では、” A0 ” センサーは PiCar-X の最も左側のセンサーです。ドロップダウン矢印を使用して、センサーを “ A1 ”（中央のセンサー）または “ A2 ”（最も右側のセンサー）に変更します。

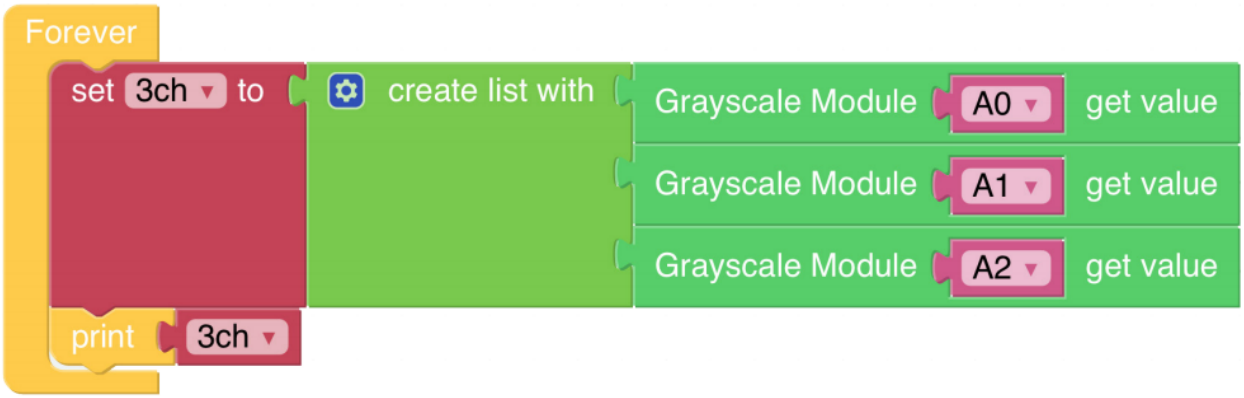


プログラムは、 **create list with** ブロックで簡略化されています。**List** は単一の **Variable** と同じように使用されますが、この場合、**Grayscale module** が 1 つ以上のセンサー値を報告するため、**List** は単一の **Variable** よりも効率的です。**create list with** ブロックは、各センサーのための別々の **Variables** を作成し、それらを List に入れます。

例

注釈:

- 以下の画像に従ってプログラムを書くことができます。チュートリアルを参照してください: How to Create a New Project?。
- EzBlock Studio の **Examples** ページで同じ名前のコードを見つけ、 **Run** または **Edit** を直接クリックしてください。



6.8 カラー検出

PiCar-X は、内蔵カメラを備えた自動運転車であり、Ezblock プログラムが物体検出および色認識コードを利用できるようになっています。このセクションでは、Ezblock を使用してカラー検出のプログラムを作成します。

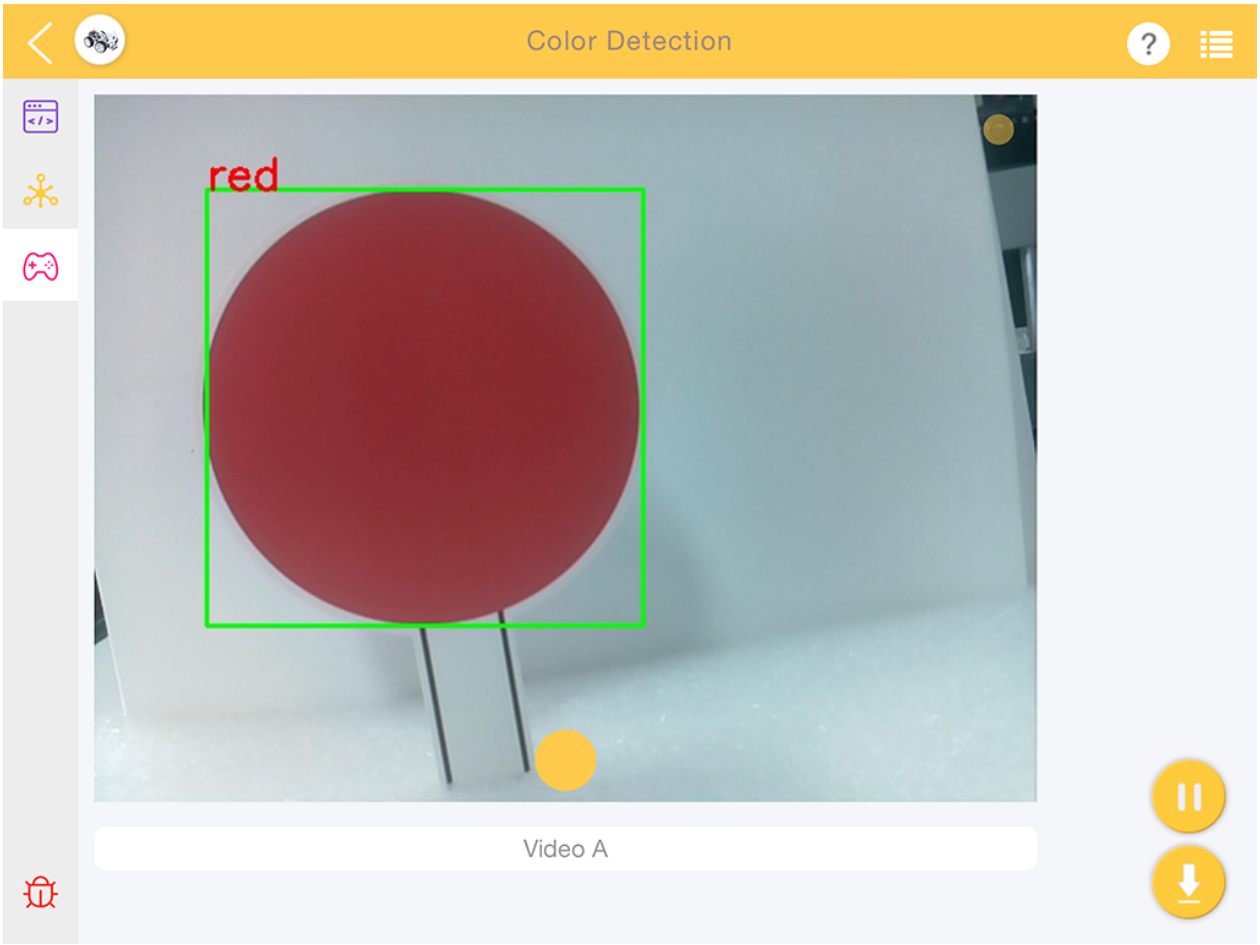
注釈: このセクションを試みる前に、Raspberry Pi カメラの FFC ケーブルが正しく、しっかりと接続されていることを確認してください。FCC ケーブルを確実に接続する方法の詳細な指示については、こちらを参照してください: **組み立て手順**。

このプログラムでは、最初に Ezblock に検出される色の色相-彩度-明度（HSV）範囲を指定し、次に OpenCV を使用して HSV 範囲内の色を処理して背景ノイズを除去し、最後に一致する色をボックス化します。

Ezblock には PiCar-X 用の 6 つのカラーモデルが含まれており、それらは「赤」「オレンジ」「黄」「緑」「青」「紫」となっています。次の PDF にはカラーカードが用意されており、カラープリンタで印刷する必要があります。

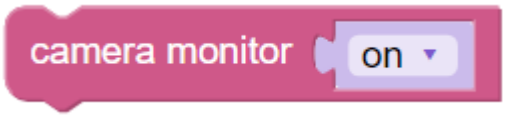
- [PDF] **カラーカード**

注釈: 印刷された色は、プリンターのトナーの違いや、たとえば褐色の紙といった印刷媒体のため、Ezblock のカラーモデルとは若干異なる色合いになる場合があります。これにより、色の認識精度が低下する可能性があります。

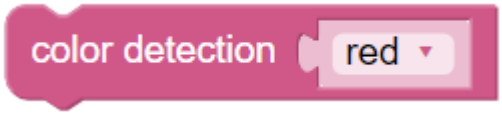


TIPS

リモートコントロールページからビデオウィジェットをドラッグすると、ビデオモニタが生成されます。ビデオウィジェットの使用方法の詳細については、こちらの Ezblock ビデオのチュートリアルを参照してください: How to Use the Video Function?。



カメラモニタ ブロックを **オン** に設定してビデオモニタを有効にします。注: **カメラモニタ** を **オフ** に設定すると、モニタは閉じますが、物体検出は利用可能です。

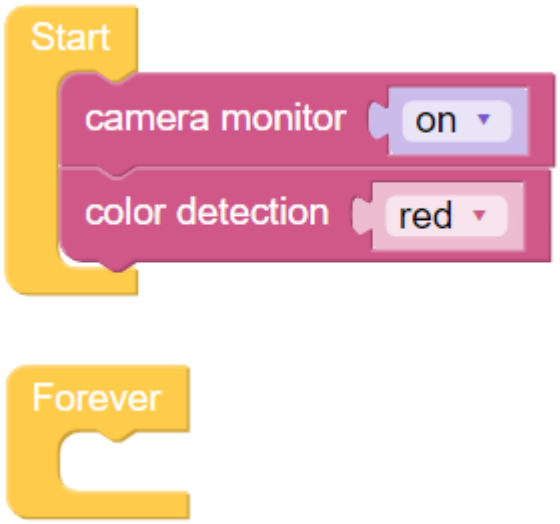


カラー検出 ブロックを使用してカラー検出を有効にします。注: 一度に 1 色のみを検出することができます。

例

注釈:

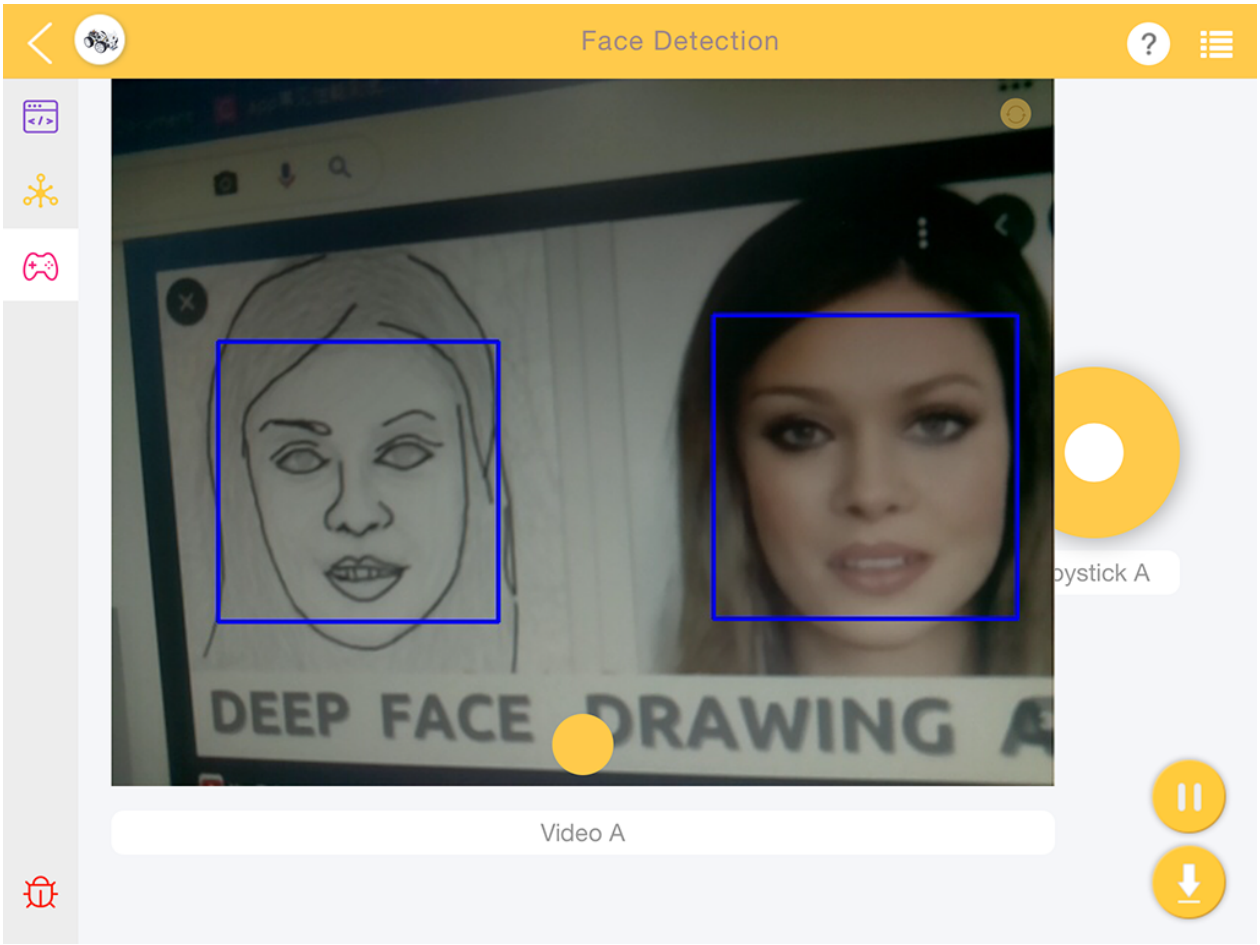
- 以下の画像に従ってプログラムを書くことができます。チュートリアルを参照してください: How to Create a New Project?。
- EzBlock Studio の **Examples** ページで同じ名前のコードを見つけ、 **Run** または **Edit** を直接クリックしてください。



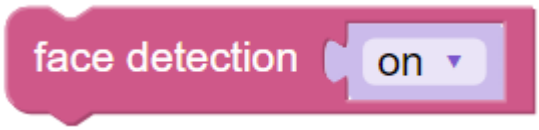
6.9 顔検出

カラー検出に加えて、PiCar-X には顔検出機能も含まれています。以下の例では、ジョイスティックウィジェットを使用してカメラの方向を調整し、デバッグモニタには検出された顔の数が表示されます。

ビデオウィジェットの使用方法の詳細については、こちらの Ezblock ビデオのチュートリアルを参照してください: How to Use the Video Function?。



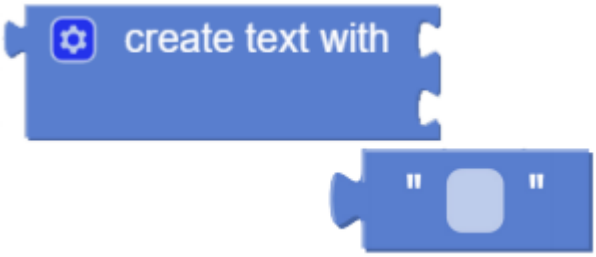
TIPS



顔検出 ウィジェットを **オン** に設定して、顔の検出を有効にします。

これらの 2 つのブロックは、 **リモートコントロール** のチュートリアルでの PiCar-X の運転と同様に、パン・チルトカメラの方向を調整するために使用されます。値が増加すると、カメラは右または上に回転し、値が減少すると、カメラは右または下に回転します。

画像検出の結果は、 **detected face** ブロックを通じて提供されます。ドロップダウンメニューオプションを使用して、画像検出機能からの座標、サイズ、または結果の数の間で選択します。

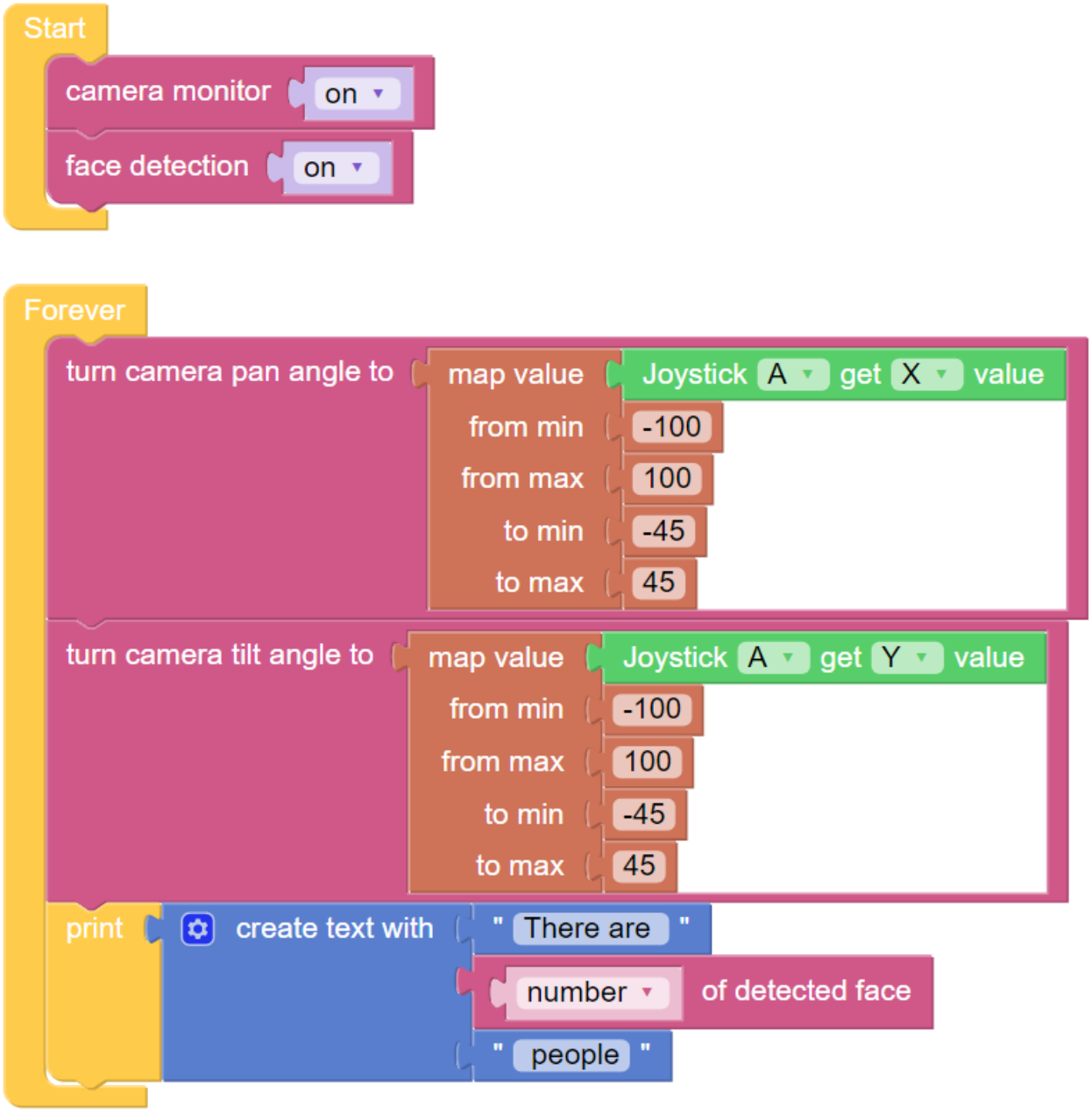


create text with ブロックを使用して、 **text** と **detected face** データの組み合わせを印刷します。

例

注釈:

- 以下の画像に従ってプログラムを書くことができます。チュートリアルを参照してください: How to Create a New Project?。
- EzBlock Studio の **Examples** ページで同じ名前のコードを見つけ、 **Run** または **Edit** を直接クリックしてください。



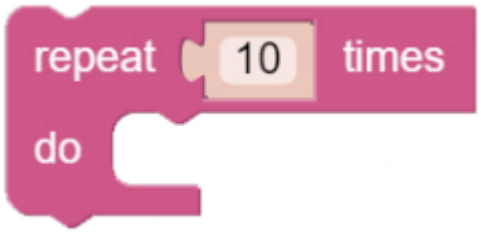
6.10 音効果

PiCar-X には音声実験に使用できる組み込みスピーカーがあります。Ezblock を使用すると、ユーザーはテキストを入力して PiCar-X に話させるか、特定の音効果を出すことができます。このチュートリアルでは、do/while 機能を使用して、3 秒間のカウントダウンの後に銃が発砲する音を PiCar-X が出します。

TIPS

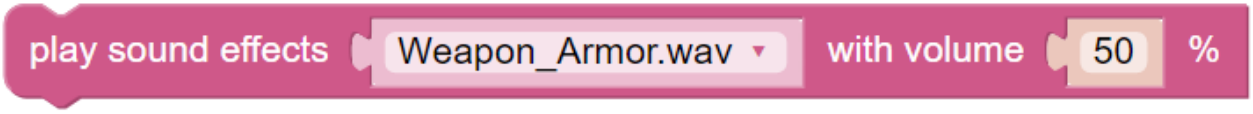
PiCar-X に文章を言わせるために、 **say** ブロックと **text** ブロックを使用します。 **say** ブロックはテキストや数字と一緒に使用することができます。

number ブロック。



repeat ブロックを使用すると、同じ文を繰り返し実行することができ、コードのサイズを縮小できます。

mathematical operation ブロックは、” + ”、” - ”、” x ”、” ÷ ” などの一般的な数学的な操作を実行することができます。

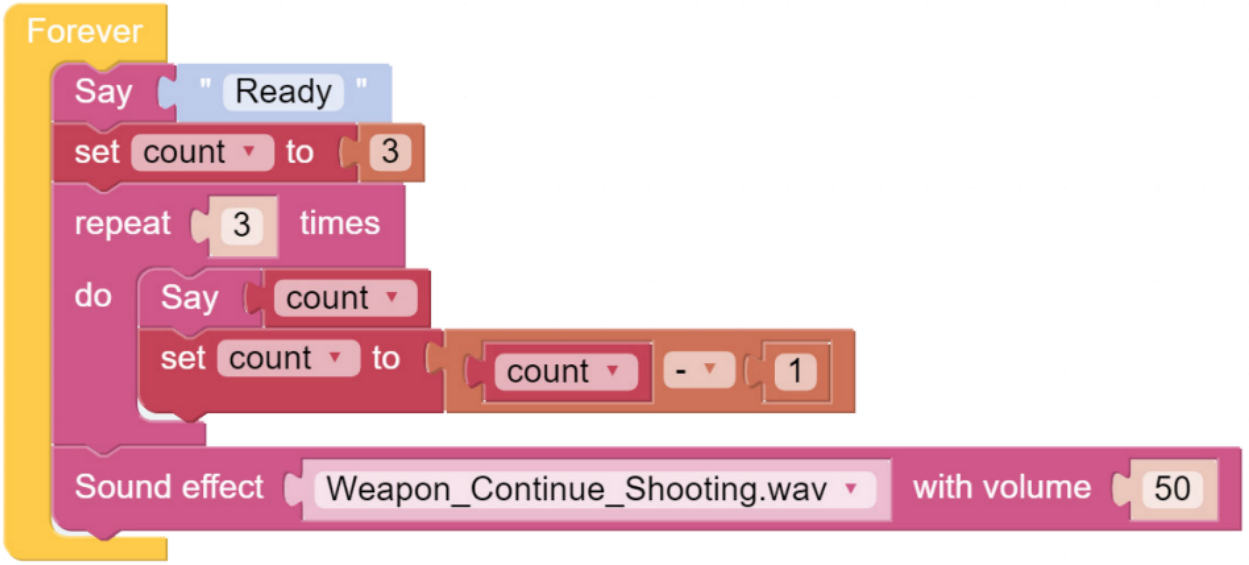


play **sound effects - with volume - %** ブロックには、サイレンの音、銃の音など、プリセットの音効果があります。音量の範囲は 0 から 100 まで設定することができます。

例

注釈:

- 以下の画像に従ってプログラムを書くことができます。チュートリアルを参照してください: How to Create a New Project?。
- EzBlock Studio の **Examples** ページで同じ名前のコードを見つけ、 **Run** または **Edit** を直接クリックしてください。



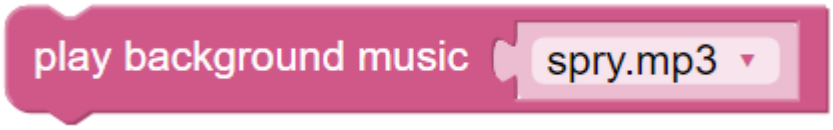
6.11 バックグラウンドミュージック

PiCar-X をプログラムして音効果やテキスト-音声 (TTS) を再生するだけでなく、PiCar-X はバックグラウンドミュージックも再生します。このプロジェクトでは、音楽の音量を調整するための **Slider** ウィジェットも使用します。

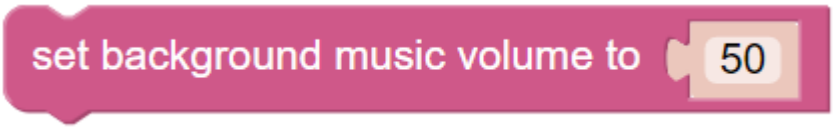
- How to Use the Remote Control Function?

Ezblocks のリモートコントロール機能の詳細なチュートリアルは、**リモートコントロール** チュートリアルを参照してください。

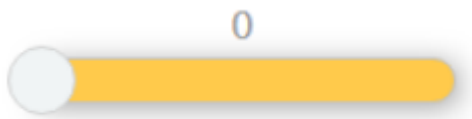
TIPS



play background music ブロックは **Start** 機能に追加する必要があります。ドロップダウンメニューを使用して、PiCar-X が再生する異なるバックグラウンドミュージックを選択します。



set background music volume to ブロックは、0 から 100 の範囲で音量を調整します。



Remote Control ページから **Slider** バーをドラッグして、音楽の音量を調整します。

slider [A] get value ブロックはスライダーの値を読み取ります。上の例では、スライダー 'A' が選択されています。複数のスライダーがある場合、ドロップダウンメニューを使用して適切なものを選択します。

例

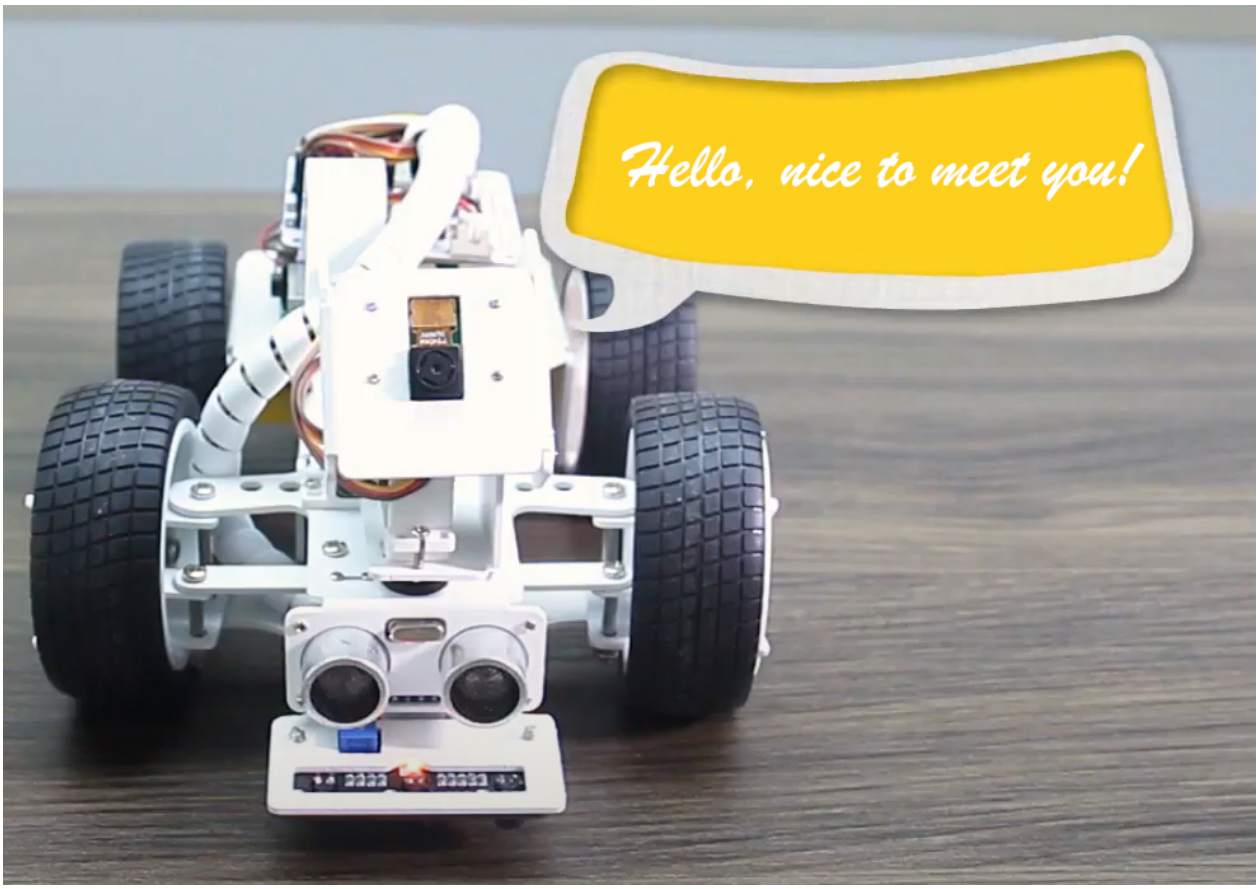
注釈:

- 以下の画像に従ってプログラムを書くことができます。チュートリアルを参照してください: How to Create a New Project?。
- EzBlock Studio の **Examples** ページで同じ名前のコードを見つけ、 **Run** または **Edit** を直接クリックしてください。

6.12 Hello

このプロジェクトでは、前述のプロジェクトからいくつかの機能を組み合わせます。PiCar-X の動きはリモートで制御され、PiCar のカメラも 2 つのジョイスティックコントローラを使用してリモートで制御されます。PiCar が誰かの顔を認識すると、礼儀正しく頷き、そして「Hello」と言います。

- How to Use the Video Function?
- How to Use the Remote Control Function?



ヒント

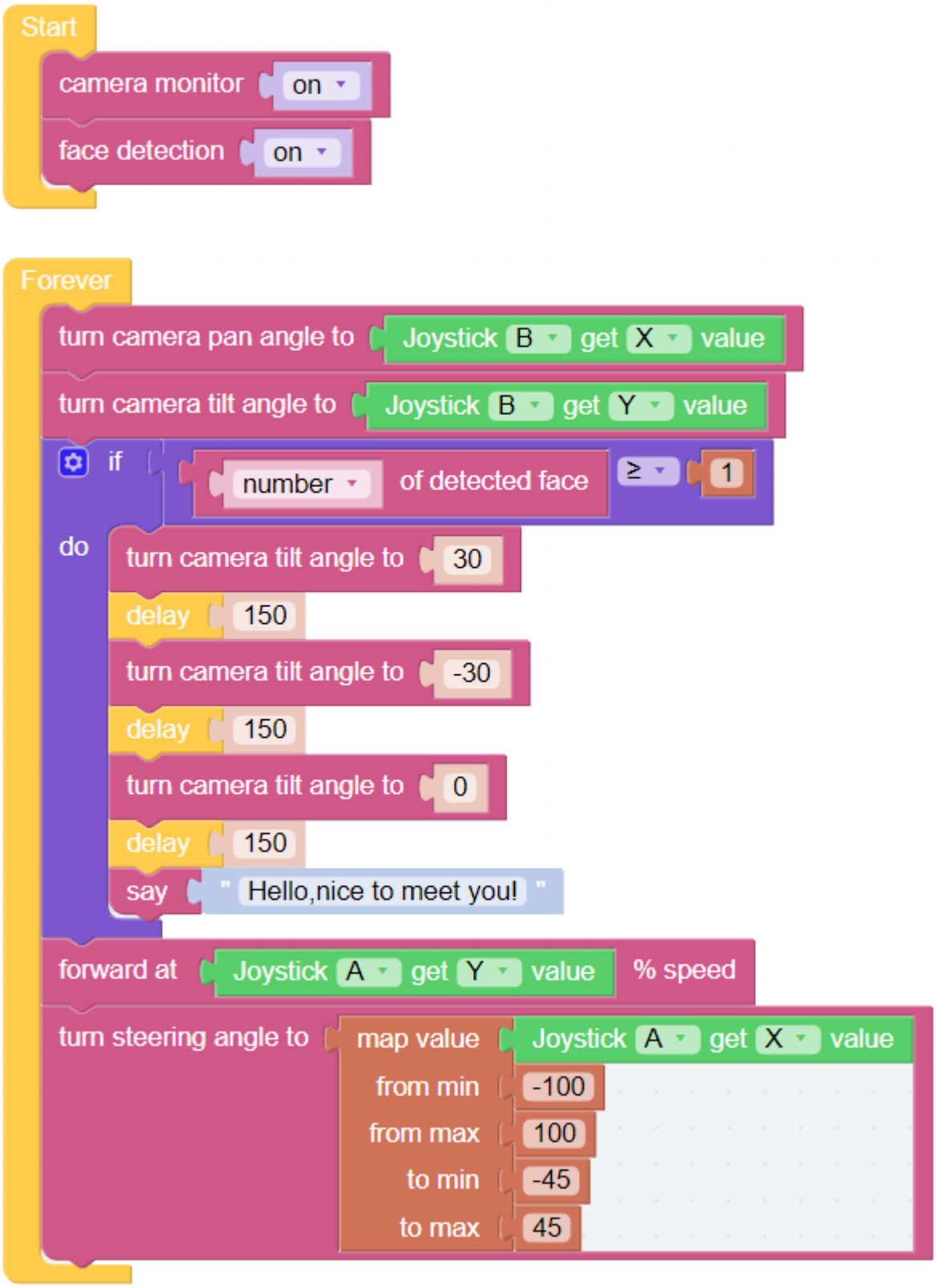
if do ブロックは、” if ” の条件判断が真の場合に礼儀正しく頷くために使用されます。

conditional statements ブロックは、**if do** ブロックと組み合わせて使用されます。条件は、”=”、”>”、”<”、” “、” “、または ” ≠ “ のいずれかであることができます。

例

注釈:

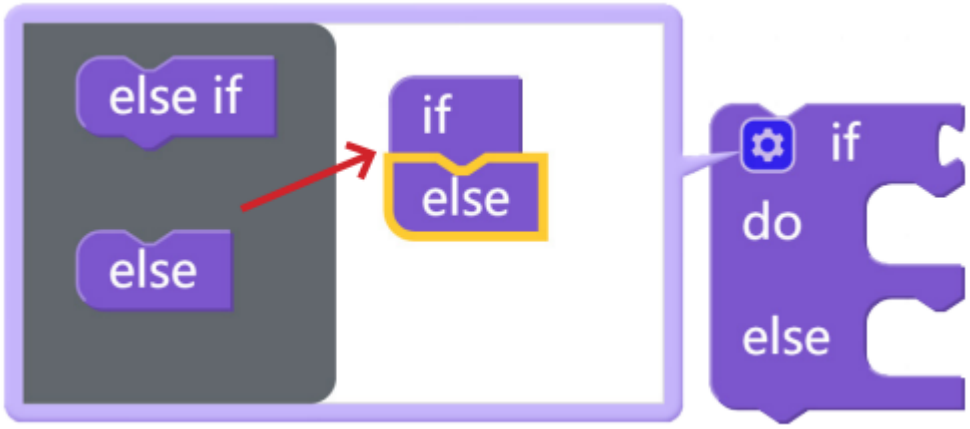
- 以下の画像に従ってプログラムを書くことができます。チュートリアルを参照してください: How to Create a New Project?。
- EzBlock Studio の **Examples** ページで同じ名前のコードを見つけ、 **Run** または **Edit** を直接クリックしてください。


Start
camera monitor on
face detection on
Forever
turn camera pan angle to Joystick B get X value
turn camera tilt angle to Joystick B get Y value
if number of detected face ≥ 1
do
turn camera tilt angle to 30
delay 150
turn camera tilt angle to -30
delay 150
turn camera tilt angle to 0
delay 150
say " Hello,nice to meet you! "
forward at Joystick A get Y value % speed
turn steering angle to map value Joystick A get X value
from min -100
from max 100
to min -45
to max 45

6.13 音楽カー

このプロジェクトでは、PiCar-X を音楽カーに変え、家の中を元気な音楽を流しながら移動させます。このプロジェクトでは、組み込まれた超音波センサーを使用して PiCar-X が壁にぶつからないようにする方法も示されます。

ヒント

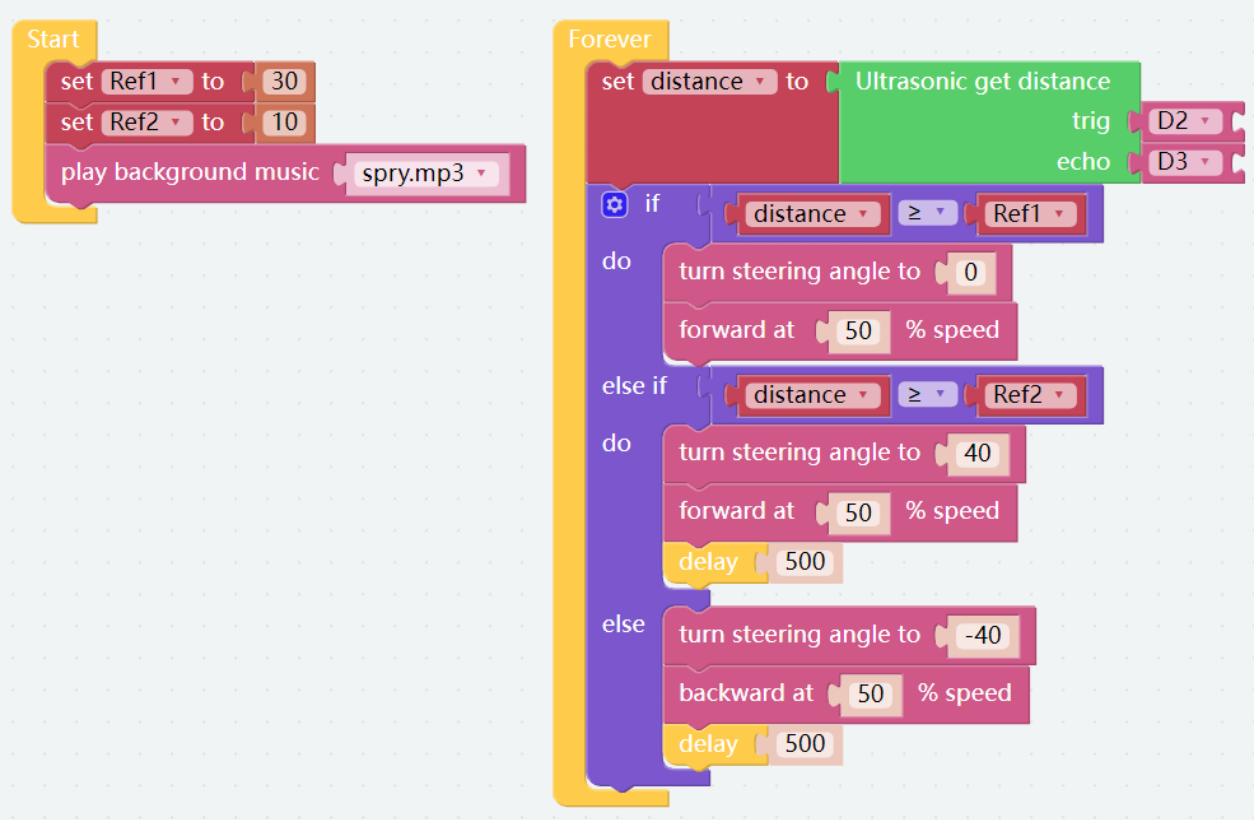


複数の条件判断を実装するには、シンプルな if do ブロックを if else do / else if do ブロックに変更します。これは、上記のように設定アイコンをクリックして行います。

例

注釈:

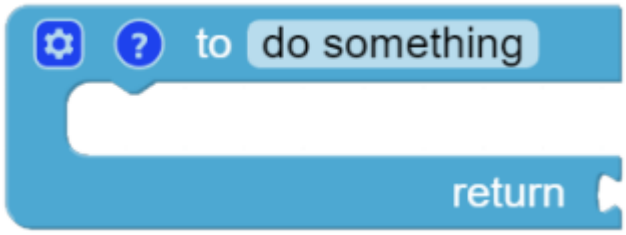
- 以下の画像に従ってプログラムを書くことができます。チュートリアルを参照してください: How to Create a New Project? 。
- EzBlock Studio の **Examples** ページで同じ名前のコードを見つけ、 **Run** または **Edit** を直接クリックしてください。



6.14 断崖検出

このプロジェクトでは、 **grayscale module** を使用して、PiCar-X が家の中を自由に移動している間に断崖から落ちないようにします。階段のある家には必須のプロジェクトです。

ヒント

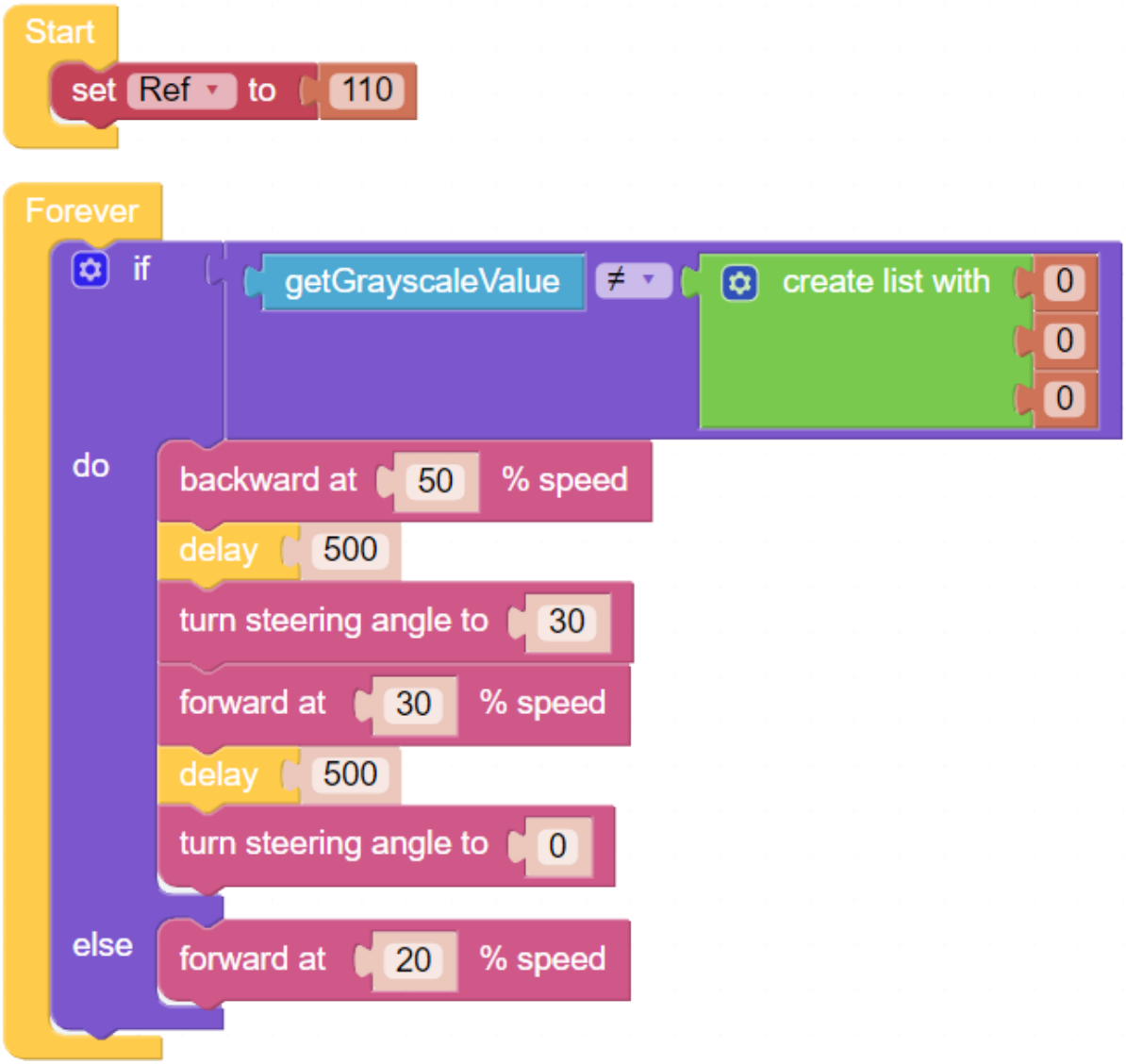


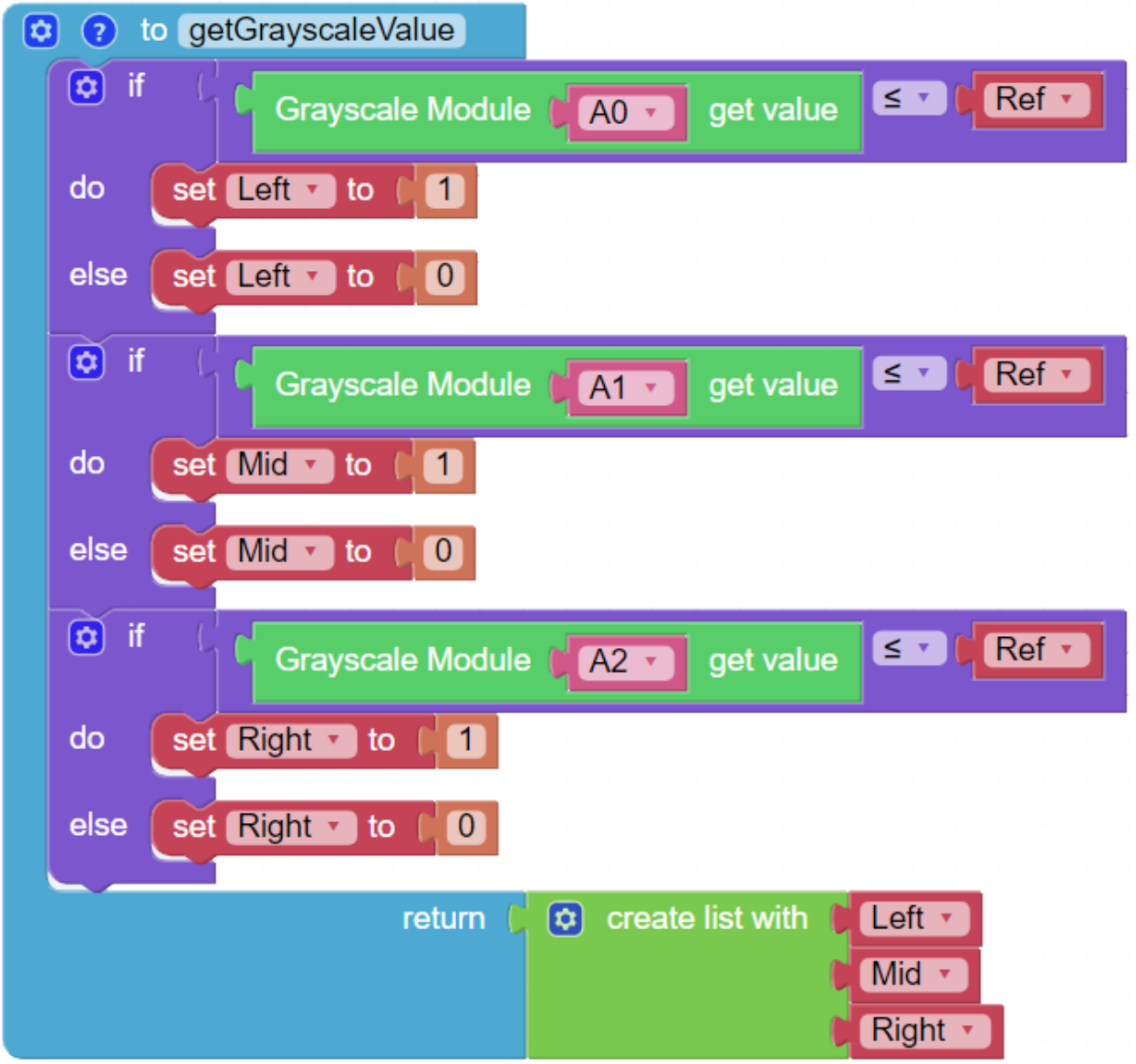
grayscale module は、同じ操作を複数回実行します。プログラムを単純化するため、このプロジェクトでは **do forever** ブロックに **list** 変数を返す **function** を紹介しています。

例

注釈:

- 以下の画像に従ってプログラムを書くことができます。チュートリアルを参照してください: How to Create a New Project?。
- EzBlock Studio の **Examples** ページで同じ名前のコードを見つけ、 **Run** または **Edit** を直接クリックしてください。





6.15 マインカート

マインカートプロジェクトを作成しましょう！このプロジェクトでは、グレースケールモジュールを使用して、PiCar-X をトラックに沿って前進させます。できるだけ直線的に、そしてあまり曲がっていないトラックを作るために、暗い色のテープを使って地面にトラックを作ります。PiCar-X が脱線する場合、実験が必要になるかもしれません。

トラックを移動するとき、グレースケールモジュールの左右のプローブは、明るい色の地面を検出し、中央のプローブはトラックを検出します。トラックに弧がある場合、センサーの左右のプローブは暗色のテープを検出し、その方向に車輪を回します。マインカートがトラックの端に達するか脱線すると、グレースケールモジュールは暗色のテープトラックを検出しなくなり、PiCar-X は停止します。

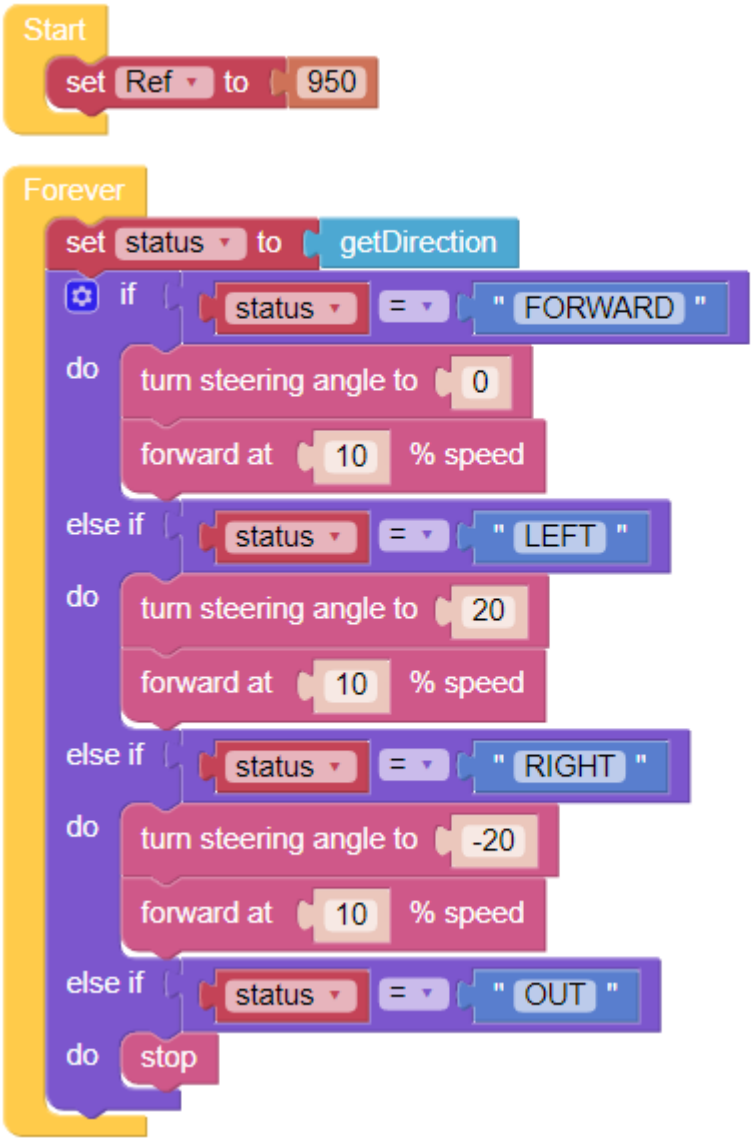
ヒント

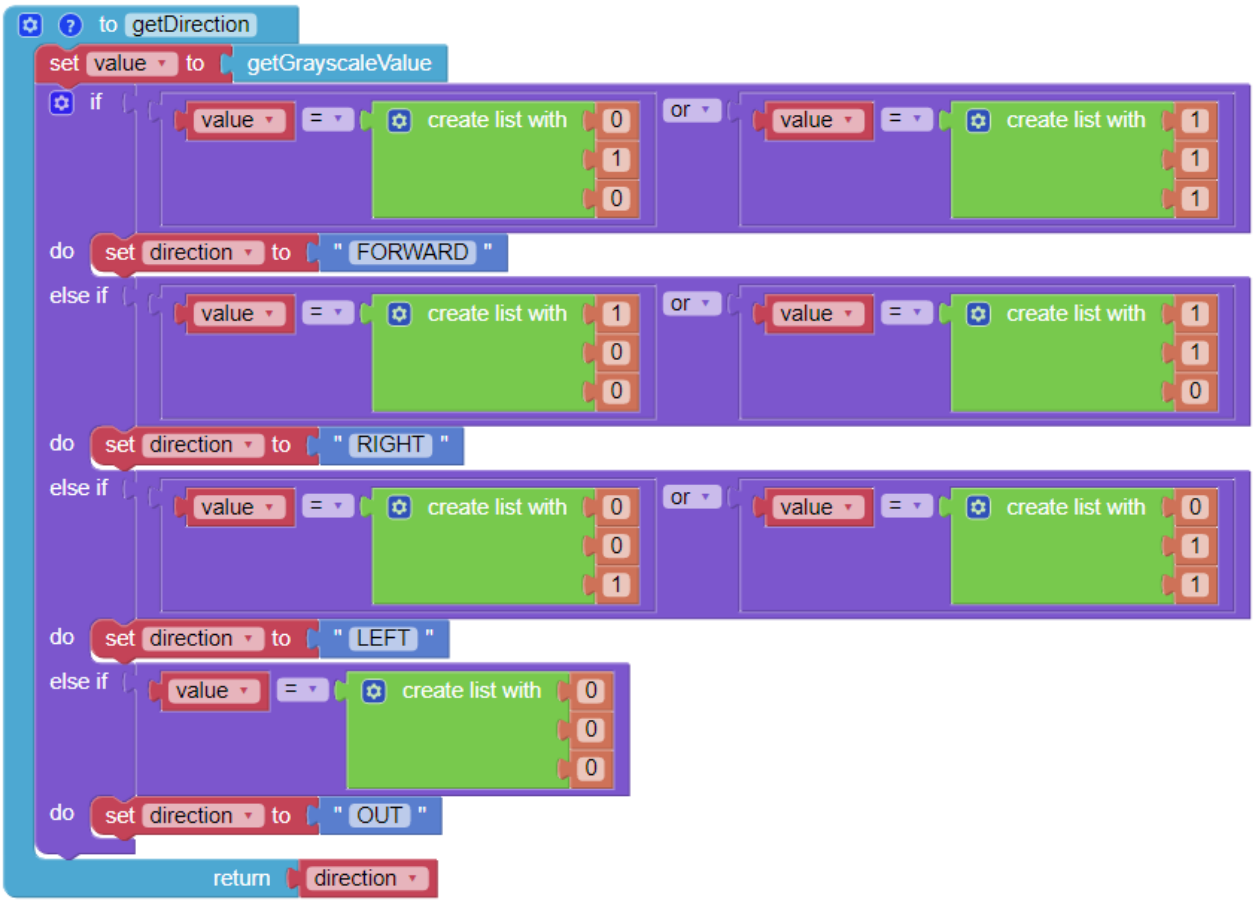
- **Set ref to ()** ブロックは、グレースケールのしきい値を設定するために使用されます。実際の状況に応じて変更する必要があります。白と黒の表面でのグレースケールモジュールの値を確認するために、 **グレースケールモジュールのテスト** を実行してみてください。このブロックにその中間値を記入します。

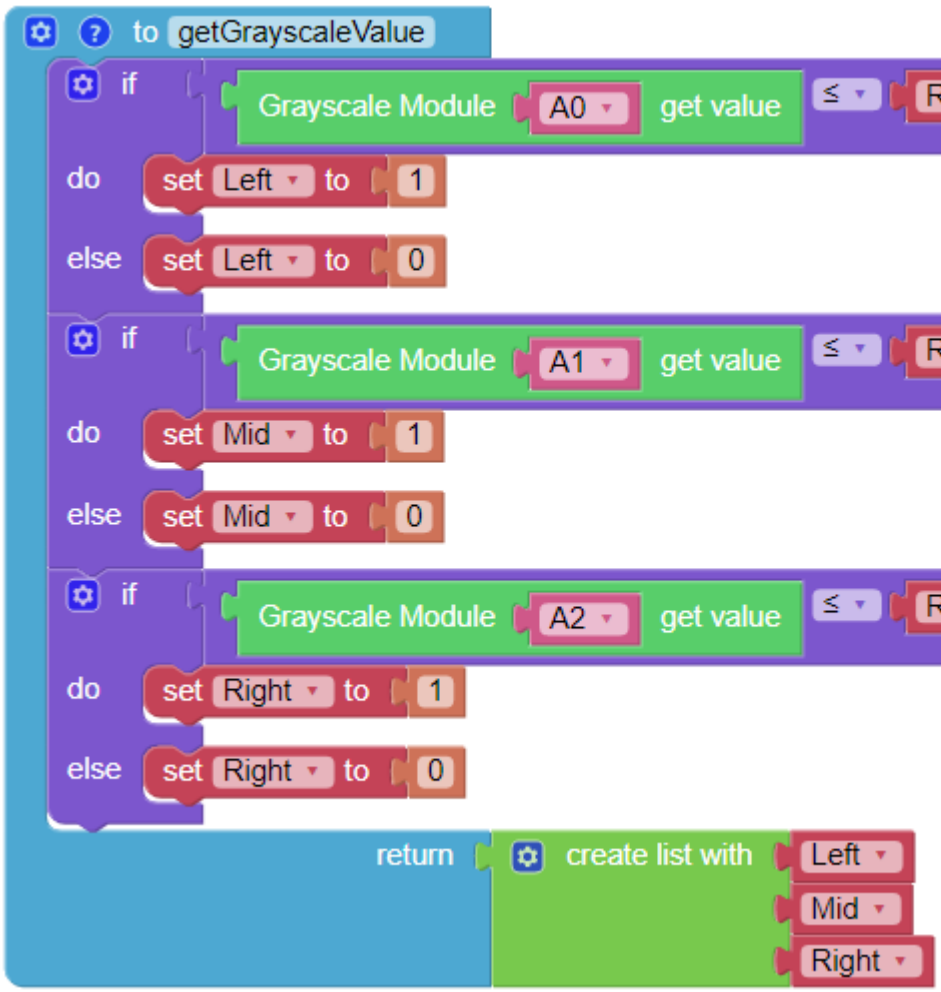
例

注釈:

- 以下の画像に従ってプログラムを書くことができます。チュートリアルを参照してください: How to Create a New Project?。
- EzBlock Studio の **Examples** ページで同じ名前のコードを見つけ、 **Run** または **Edit** を直接クリックしてください。

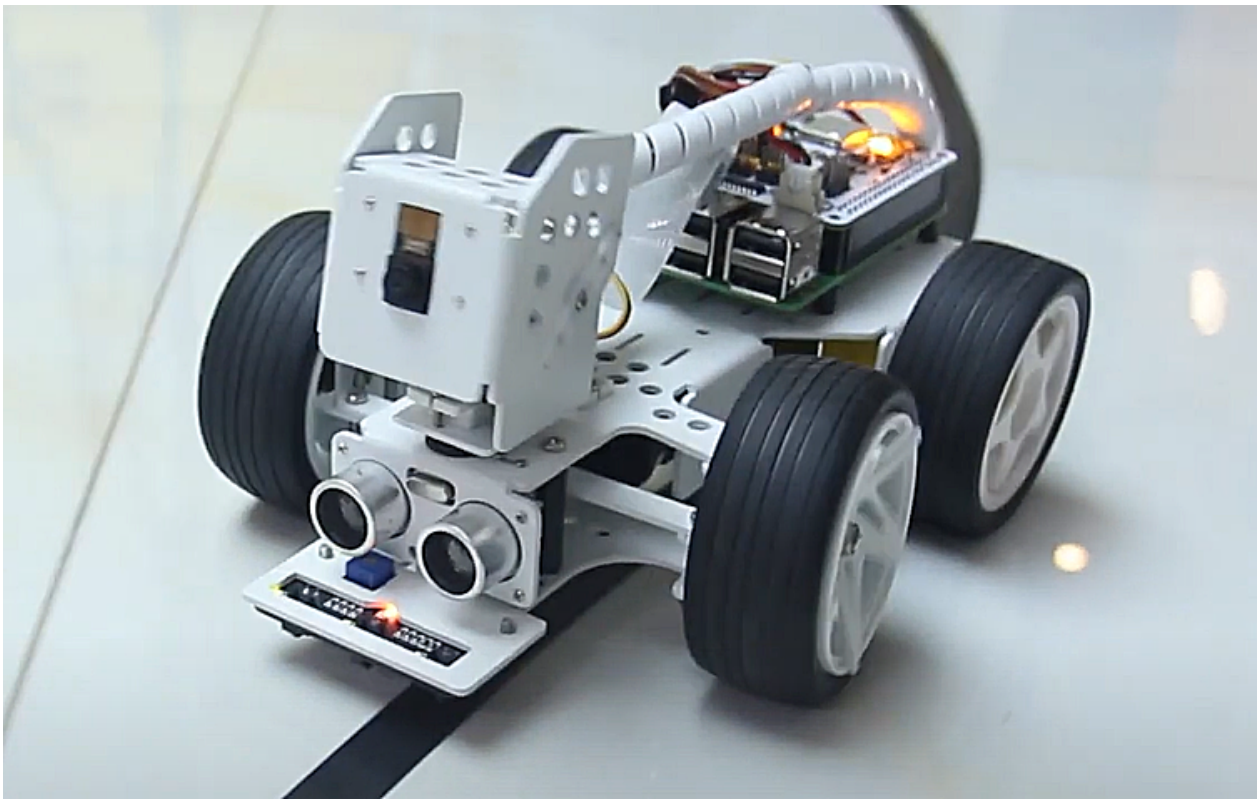

Start
set Ref to 950
Forever
set status to getDirection
if status = " FORWARD "
do
turn steering angle to 0
forward at 10 % speed
else if status = " LEFT "
do
turn steering angle to 20
forward at 10 % speed
else if status = " RIGHT "
do
turn steering angle to -20
forward at 10 % speed
else if status = " OUT "
do
stop


to getDirection
set value to getGrayscaleValue
if
value = create list with 0 1 0
or
value = create list with 1 1 1
do
set direction to " FORWARD "
else if
value = create list with 1 0 0
or
value = create list with 1 1 0
do
set direction to " RIGHT "
else if
value = create list with 0 0 1
or
value = create list with 0 1 1
do
set direction to " LEFT "
else if
value = create list with 0 0 0
do
set direction to " OUT "
return direction



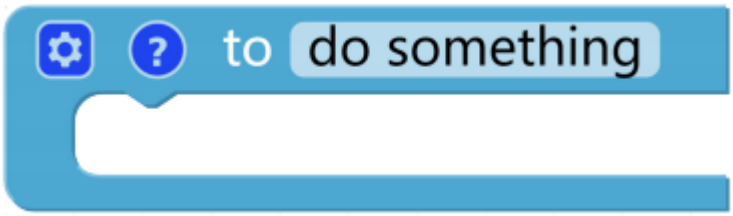
6.16 マインカートプラス

このプロジェクトでは、**マインカート** プロジェクトに脱線復旧が追加され、PiCar-X がさらに厳しいカーブから適応して復旧することができます。



ヒント

1. もう一つの **to do something** ブロックを使用して、PiCar-X が鋭いカーブからバックアップして復旧できるようにします。新しい **to do something** 関数は値を返さないことに注意してください。この関数は PiCar-X の方向を再設定するためだけに使用されます。



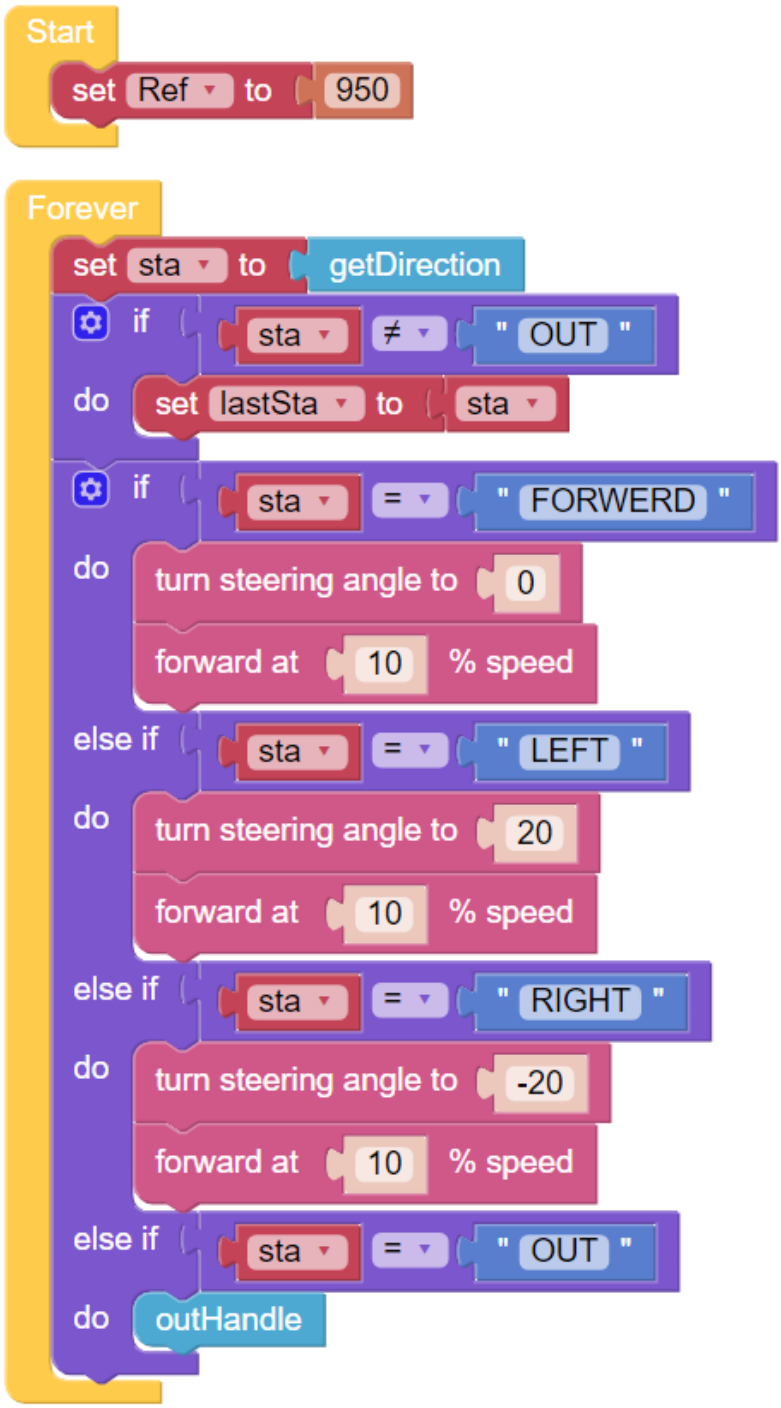
2. **Set ref to ()** ブロックは、グレースケールのしきい値を設定するために使用されます。実際の状況に応じて変更する必要があります。白と黒の表面でのグレースケールモジュールの値を確認するために、 **グレースケールモジュールのテスト** を実行してみてください。このブロックにその中間値を記入します。

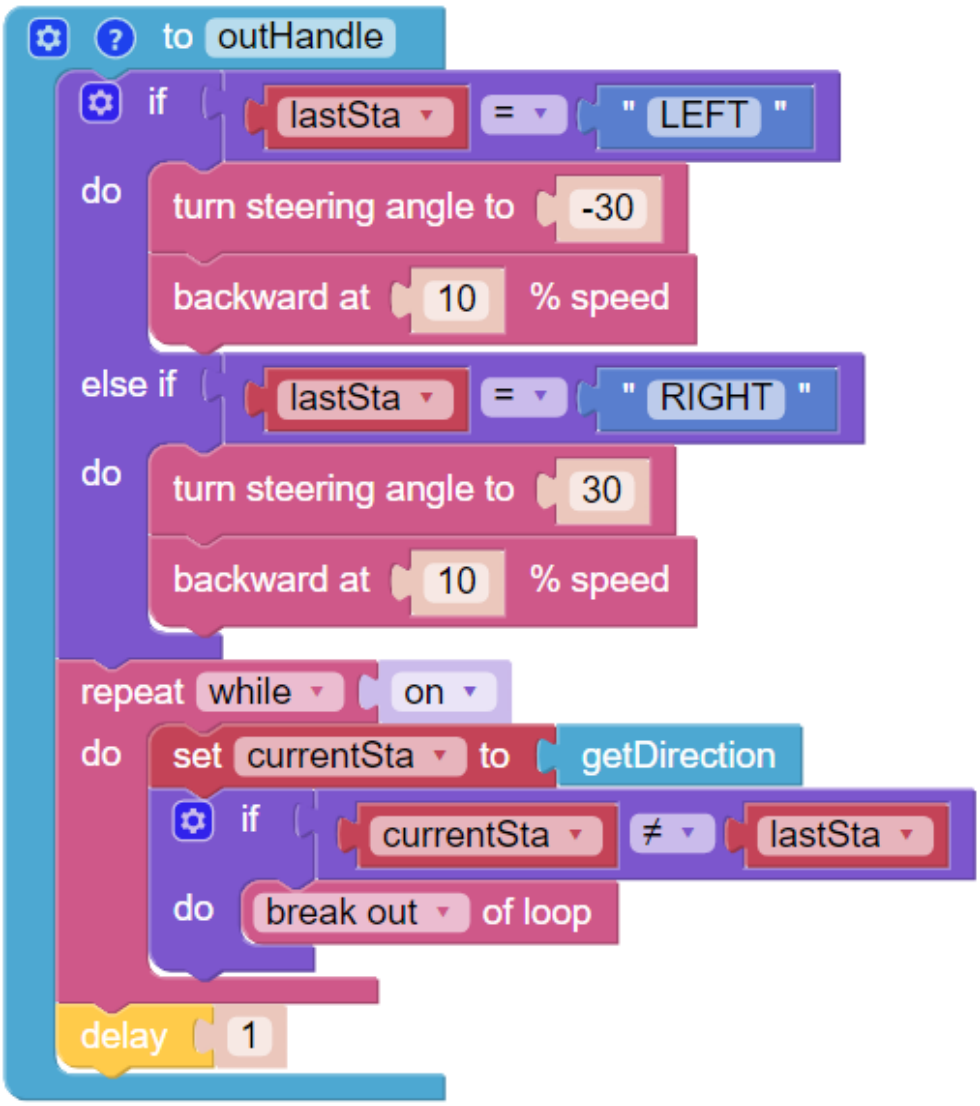
例

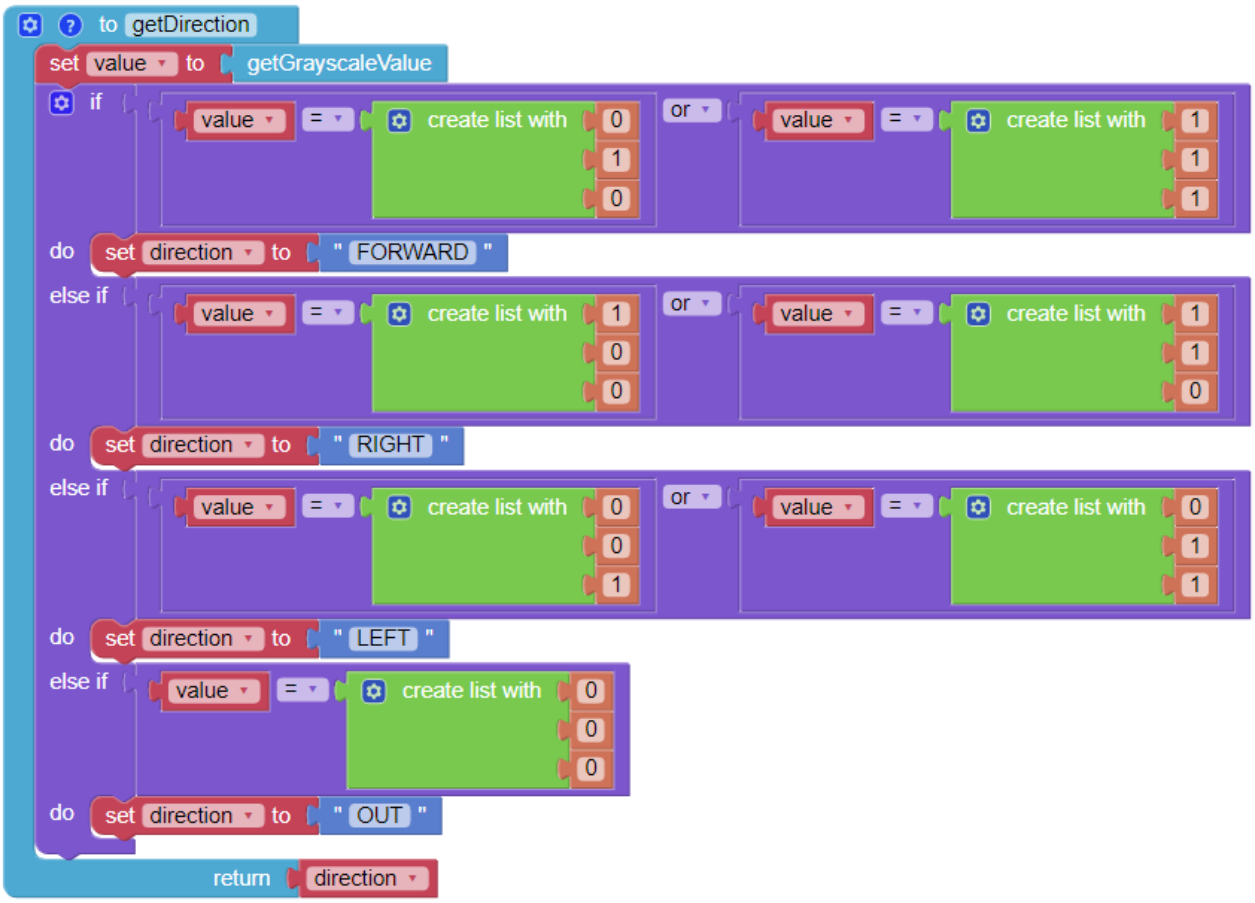
注釈:

- 以下の画像に従ってプログラムを書くことができます。チュートリアルを参照してください: How to Create a New Project?。

- EzBlock Studio の **Examples** ページで同じ名前のコードを見つけ、 **Run** または **Edit** を直接クリックしてください。




to outHandle
if lastSta = " LEFT "
do
turn steering angle to -30
backward at 10 % speed
else if lastSta = " RIGHT "
do
turn steering angle to 30
backward at 10 % speed
repeat while on
do
set currentSta to getDirection
if currentSta ≠ lastSta
do
break out of loop
delay 1


to getDirection
set value to getGrayscaleValue
if
value = create list with 0 1 0
or
value = create list with 1 1 1
do
set direction to " FORWARD "
else if
value = create list with 1 0 0
or
value = create list with 1 1 0
do
set direction to " RIGHT "
else if
value = create list with 0 0 1
or
value = create list with 0 1 1
do
set direction to " LEFT "
else if
value = create list with 0 0 0
do
set direction to " OUT "
return direction

6.17 闘牛

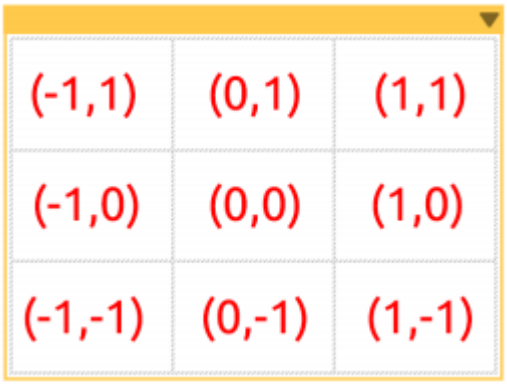
PiCar-X を怒った牛に変えてみましょう！赤い布、ハンカチなどを用意して、闘牛士になってください。PiCar-X が赤い布の後を追いかけるときは、当たらないように気をつけてください！

注釈: このプロジェクトは、先行するプロジェクトよりも高度です。PiCar-X は、カメラが赤い布を向いている方向に応じて本体の向きを自動的に調整するため、色検出機能を使用する必要があります。

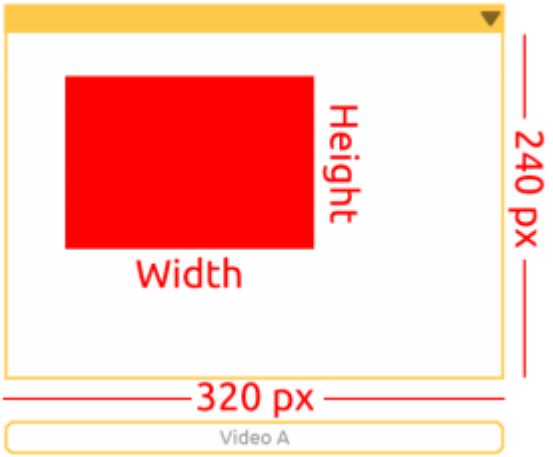
ヒント

Start ウィジェットに **color detection [red]** ブロックを追加して、PiCar-X が赤色の物体を探すようにします。

forever ループに **[width] of detected color** ブロックを追加して、入力を「オブジェクト検出」グリッドに変換します。



「オブジェクト検出」は、カメラ画像の中心点を基にして、検出された座標を (x, y) の値で出力します。画面は以下に示すような 3x3 のグリッドに分けられています。したがって、赤い布がカメラの画像の左上に保持されている場合、(x, y) の座標は (-1, 1) となります。

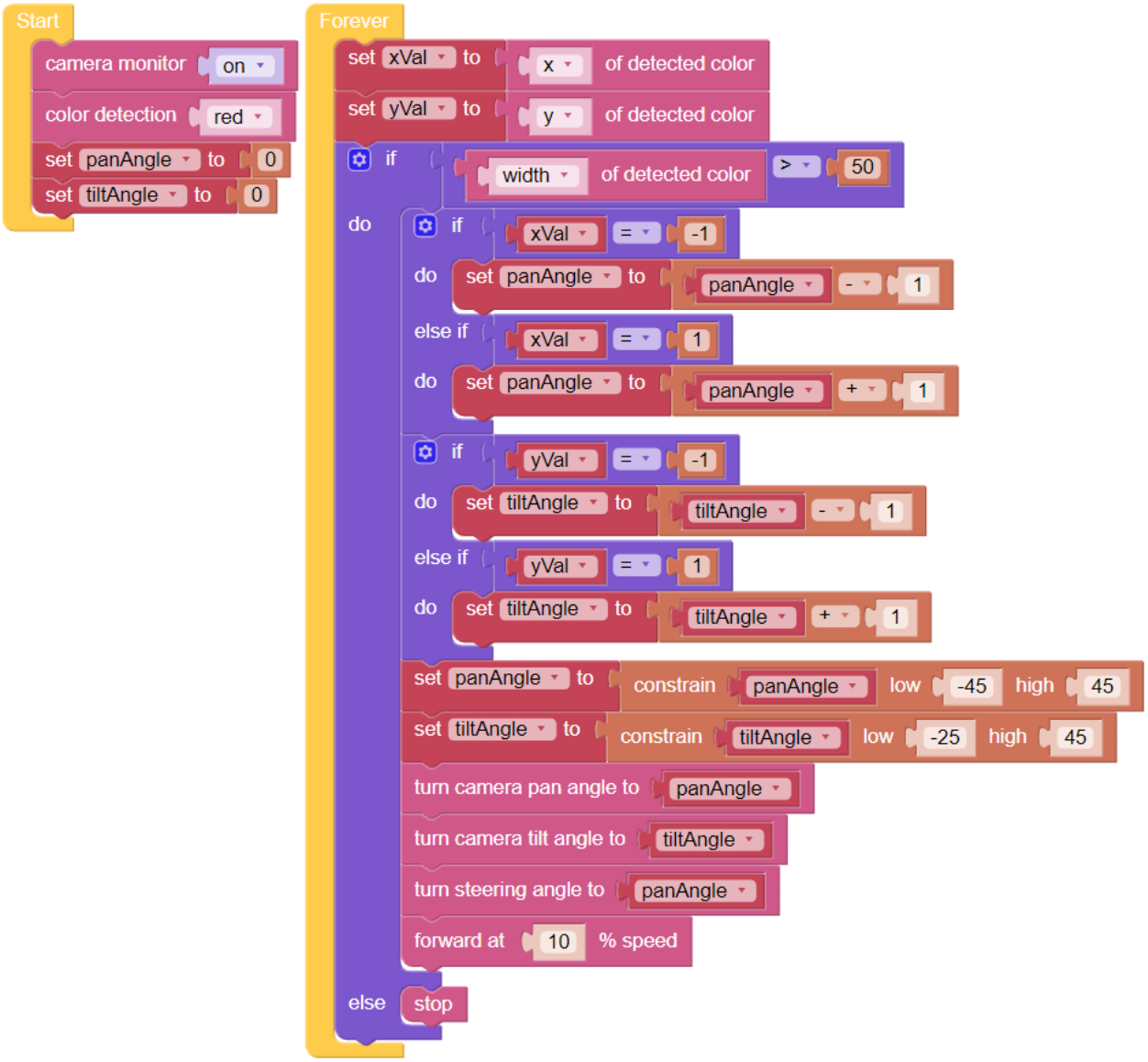


「オブジェクト検出」は、グラフィックの幅と高さを検出します。複数のターゲットが識別された場合、最大のターゲットの寸法が記録されます。

例

注釈:

- 以下の画像に従ってプログラムを書くことができます。チュートリアルを参照してください: How to Create a New Project?。
- EzBlock Studio の **Examples** ページで同じ名前のコードを見つけ、 **Run** または **Edit** を直接クリックしてください。



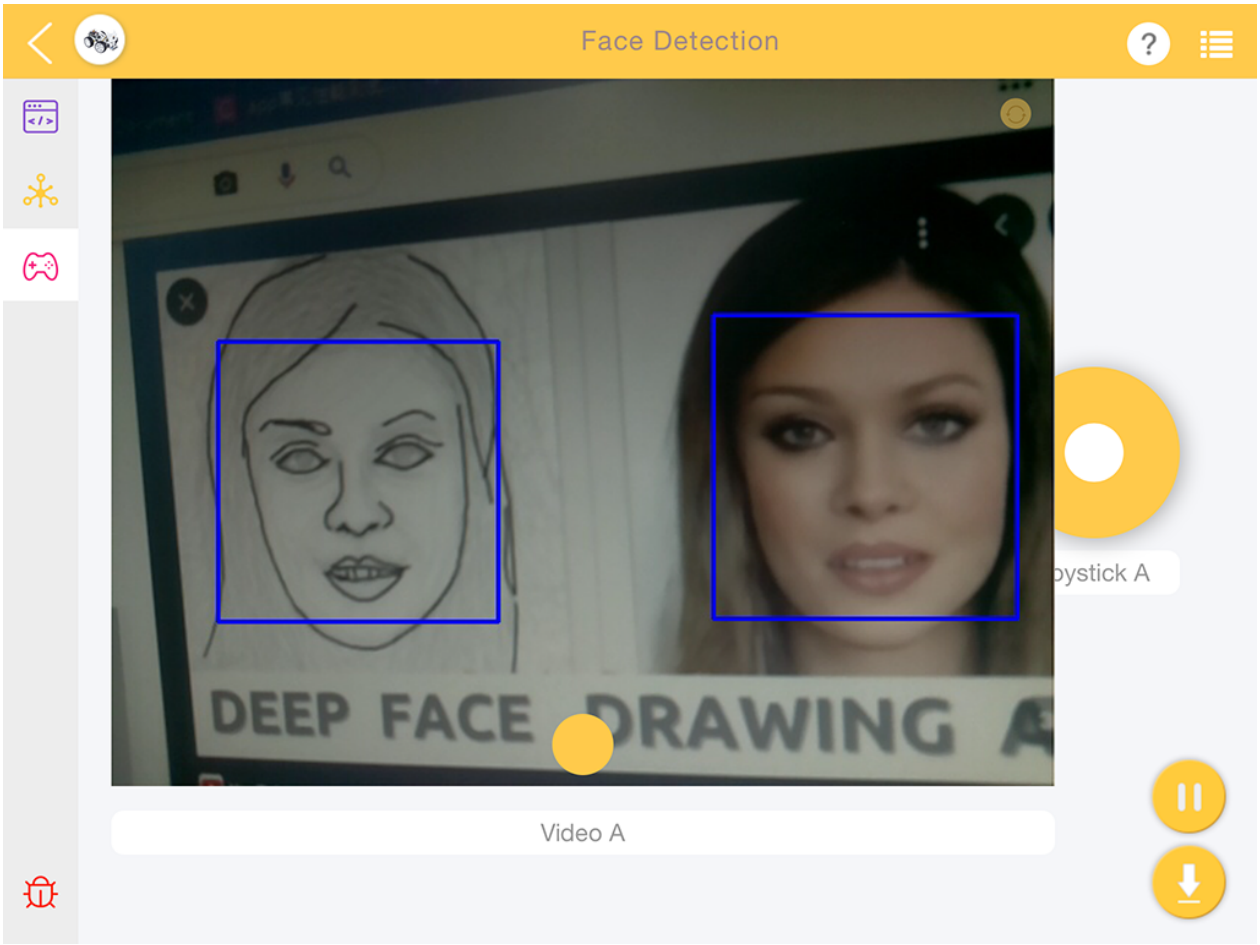
6.18 歩行者に注意

このプロジェクトでは、PiCar-X が道路の状況に基づいて適切な措置を実行するようにします。運転中、PiCar-X の進行方向に歩行者が検出された場合、完全に停止します。

プログラムを実行している間、PiCar-X の前に人物の写真を持ってください。ビデオモニターが人物の顔を検出し、PiCar-X が自動的に停止します。

運転安全プロトコルをシミュレートするために、 **[count]** の値を **if do else** ブロックに送信する判定手続きが作成されます。判定手続きは、人の顔を 10 回探します。顔が表示されると、 **[count]** は +1 増加します。 **[count]** が 3 より大きい場合、PiCar-X は動きを停止します。

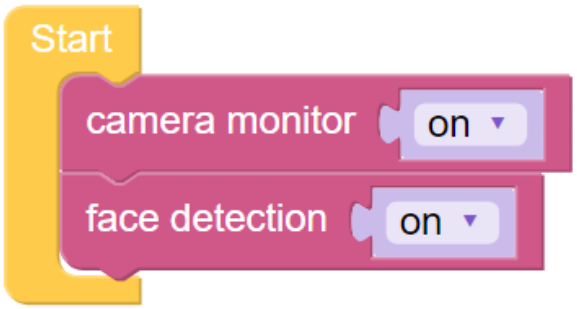
- How to Use the Remote Control Function?

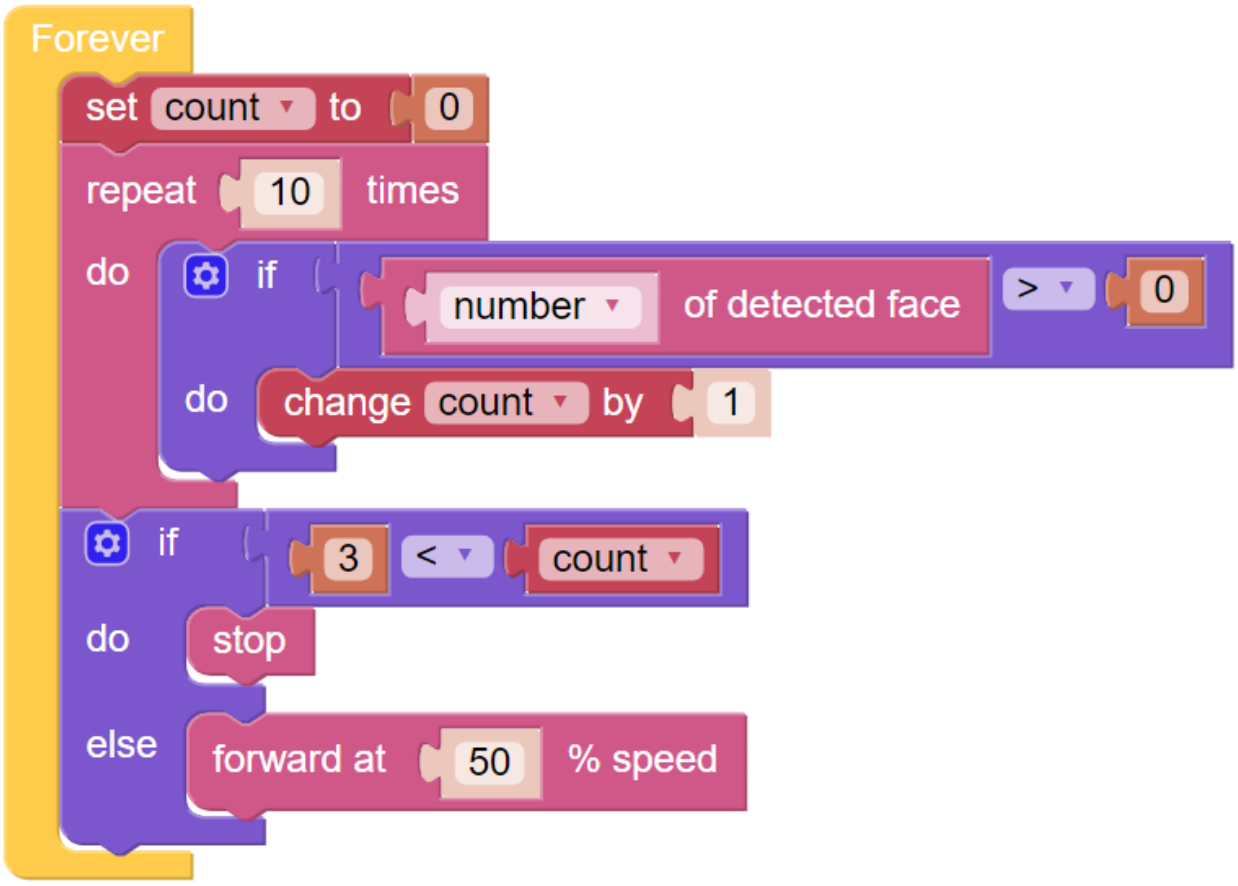


例

注釈:

- 以下の画像に従ってプログラムを書くことができます。チュートリアルを参照してください: How to Create a New Project?。
- EzBlock Studio の **Examples** ページで同じ名前のコードを見つけ、 **Run** または **Edit** を直接クリックしてください。





6.19 交通標識の検出

色や顔の検出に加えて、PiCar-X は交通標識の検出も行うことができます。

今回は、この交通標識の検出機能とライン追従機能を組み合わせてみましょう。PiCar-X にラインを追跡させ、その前に Stop サインを置くと、それは停止します。Forward サインを前に置くと、前進し続けます。

ヒント

1. PiCar は、以下の PDF に含まれる 4 つの異なる交通標識モデルを認識します。

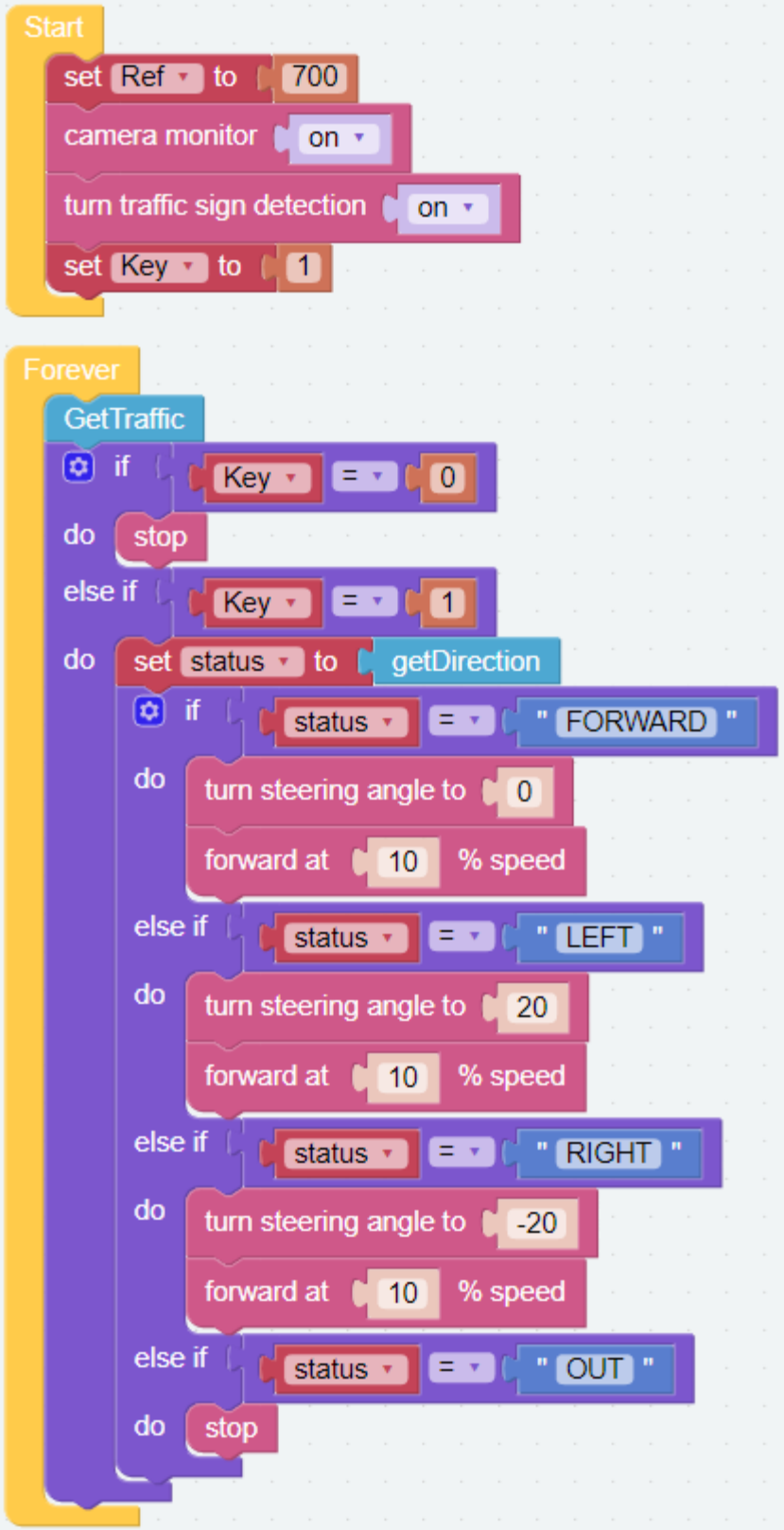
   - [PDF] **交通標識カード**

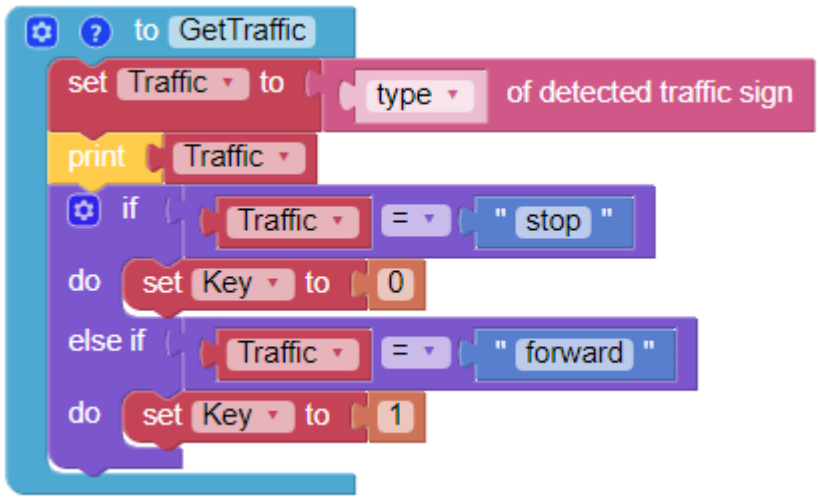
2. **Set ref to ()** ブロックは、グレースケールのしきい値を設定するために使用されます。実際の状況に応じてそれを変更する必要があります。**グレースケールモジュールのテスト** を実行して、白と黒の表面でのグレースケールモジュールの値を見ることができます。そして、その中間の値をこのブロックに入力します。

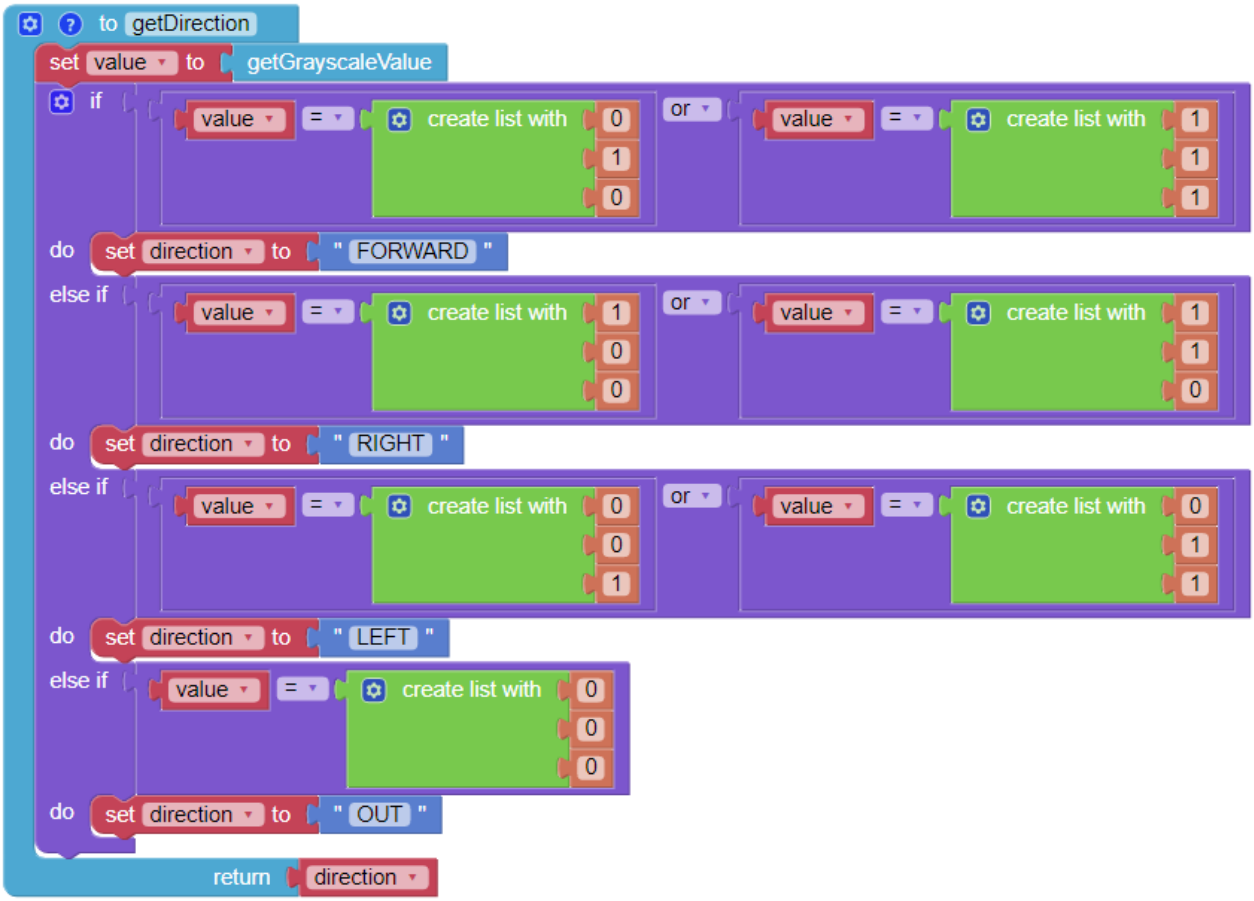
例

注釈:

- 以下の画像に従ってプログラムを書くことができます。チュートリアルを参照してください: How to Create a New Project?。
- EzBlock Studio の **Examples** ページで同じ名前のコードを見つけ、 **Run** または **Edit** を直接クリックしてください。


Start
set Ref to 700
camera monitor on
turn traffic sign detection on
set Key to 1
Forever
GetTraffic
if Key = 0
do stop
else if Key = 1
do set status to getDirection
if status = " FORWARD "
do turn steering angle to 0
forward at 10 % speed
else if status = " LEFT "
do turn steering angle to 20
forward at 10 % speed
else if status = " RIGHT "
do turn steering angle to -20
forward at 10 % speed
else if status = " OUT "
do stop


to GetTraffic
set Traffic to type of detected traffic sign
print Traffic
if Traffic = " stop "
do set Key to 0
else if Traffic = " forward "
do set Key to 1


to getDirection
set value to getGrayscaleValue
if value = create list with 0 1 0 or value = create list with 1 1 1
do set direction to " FORWARD "
else if value = create list with 1 0 0 or value = create list with 1 1 0
do set direction to " RIGHT "
else if value = create list with 0 0 1 or value = create list with 0 1 1
do set direction to " LEFT "
else if value = create list with 0 0 0
do set direction to " OUT "
return direction

6.20 オリエンティアリング

このプロジェクトは、リモートコントロール機能を使用して、PiCar-X を競技的なサバンジャーハントを通してガイドします！

まず、PiCar-X が通過できる障害物コース、迷路、または空の部屋を設置します。次に、ルートに沿ってランダムに 6 つのマーカーを配置し、PiCar-X が見つけるための各 6 つのマーカーにカラーカードを置きます。

PiCar-X の 6 つのカラーモデルは、赤、オレンジ、黄、緑、青、紫であり、下の PDF からカラープリンタで印刷する準備ができています。

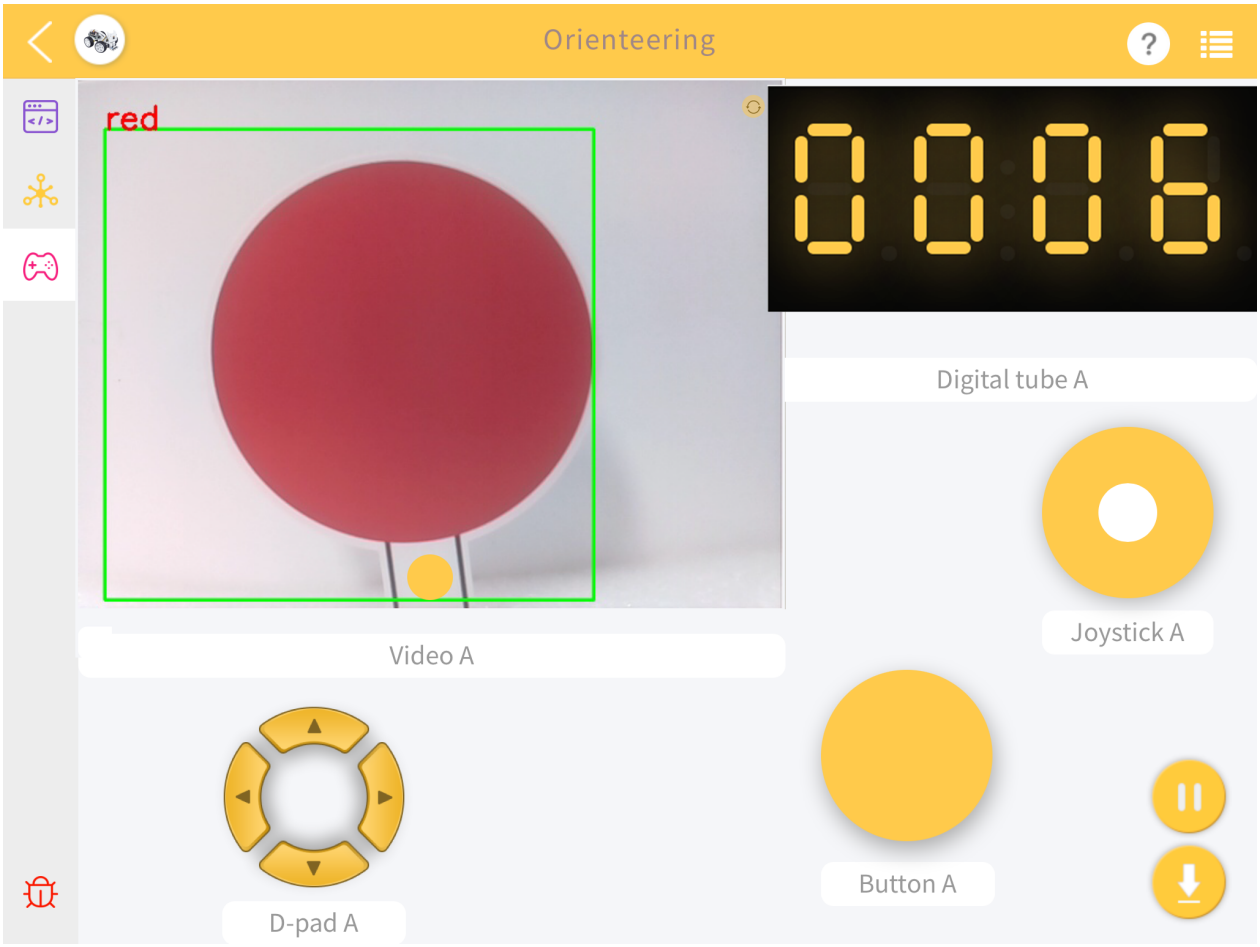
- [PDF] **カラーカード**

注釈: 印刷された色は、プリンタートナーの違いや、たとえばベージュ色の紙などの印刷媒体のため、Ezblock カラーモデルからわずかに異なる色合いを持つことがあります。これにより、色の認識があまり正確ではなくなる可能性があります。

PiCar-X は、ランダムな順序で 6 色のうち 3 色を見つけるようにプログラムされ、次に探す色をアナウンスするために TTS 機能を使用します。

目的は、PiCar-X ができるだけ短時間で 3 色のうちのそれぞれを見つけるのを助けることです。

フィールドの中央に PiCar-X を置き、リモートコントロールページのボタンをクリックしてゲームを開始します。

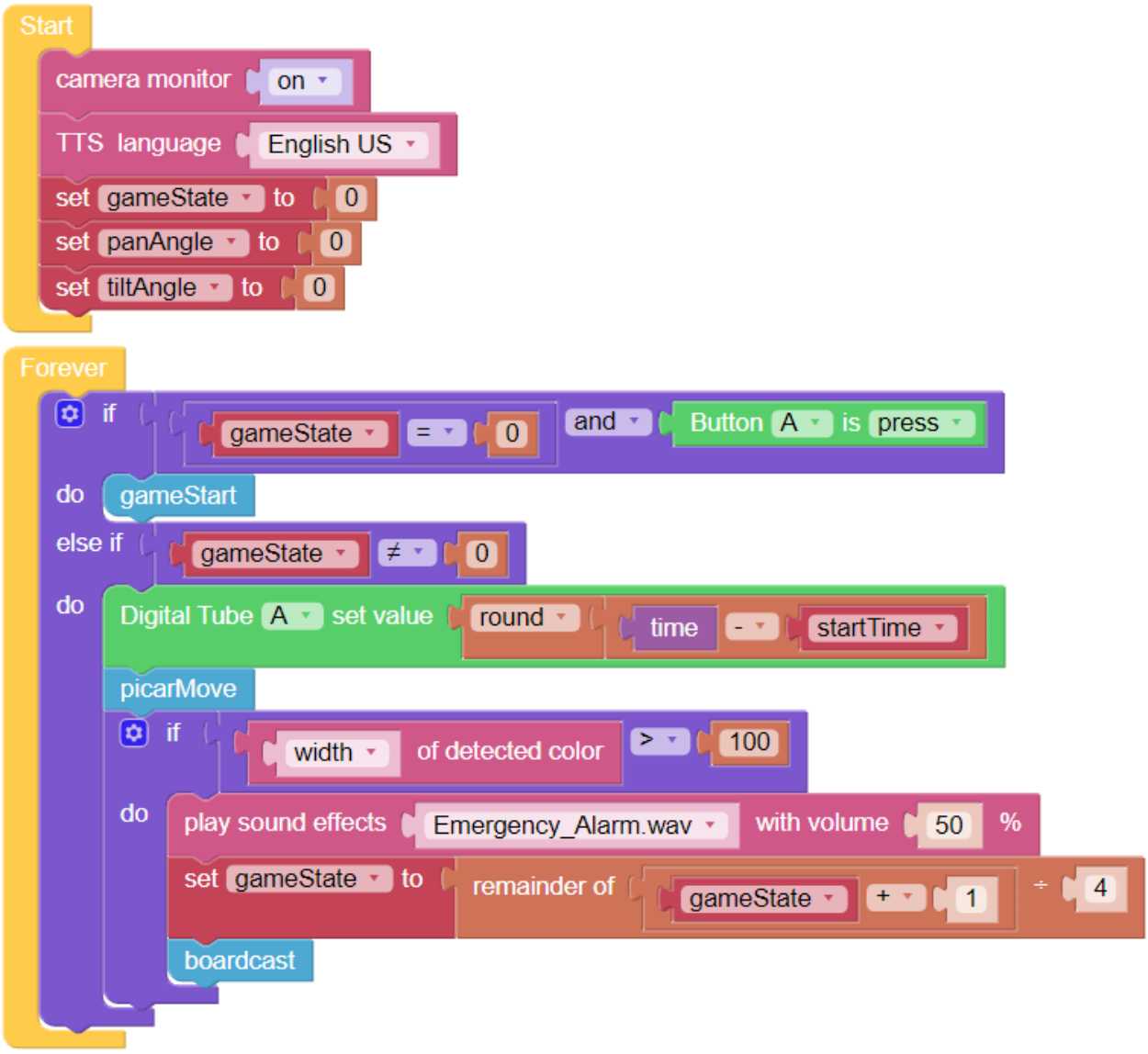


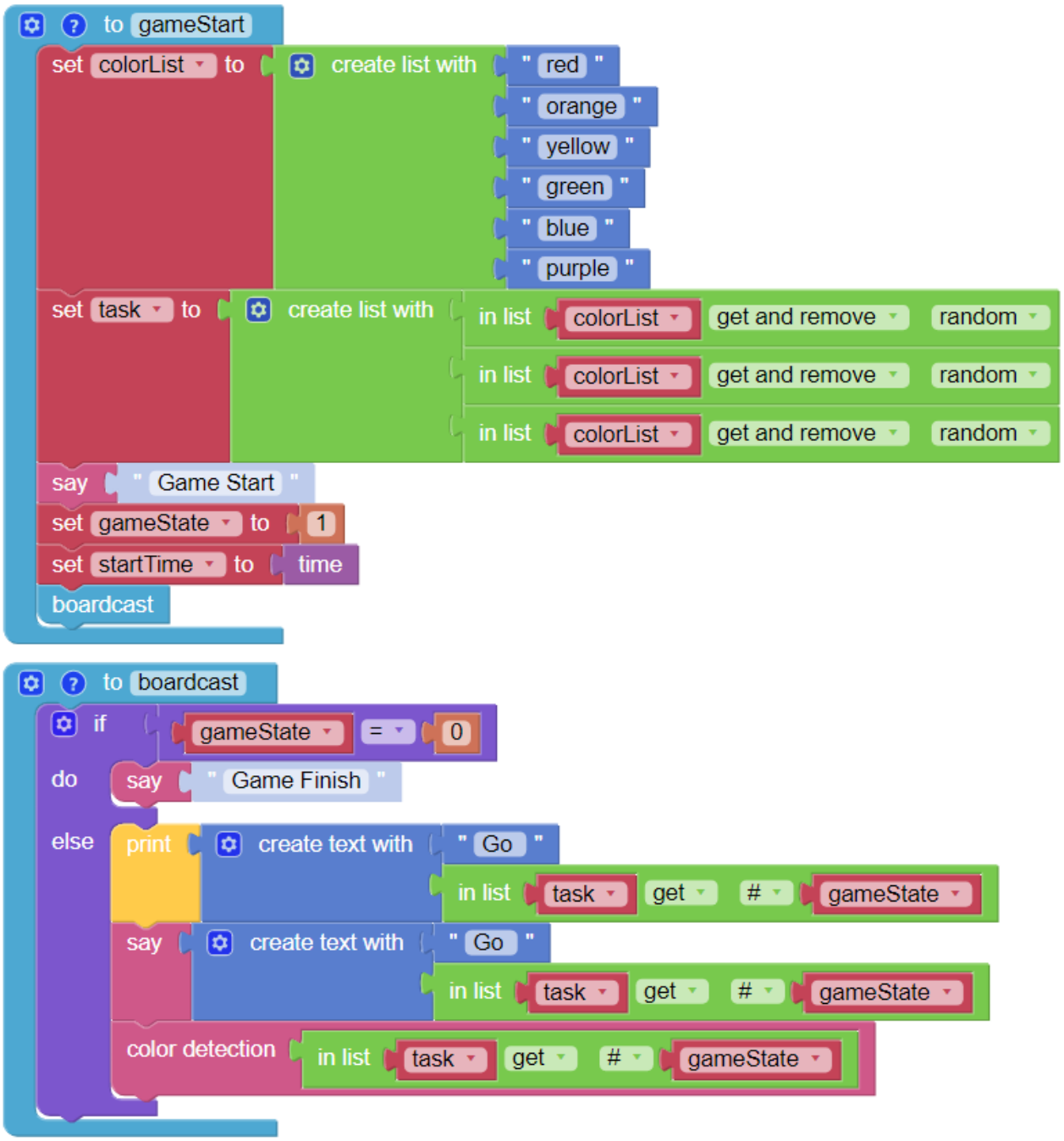
友達と交代でこのゲームをプレイして、PiCar-X が目的を最も速く達成するのを助けることができるのは誰かを見てみてください！

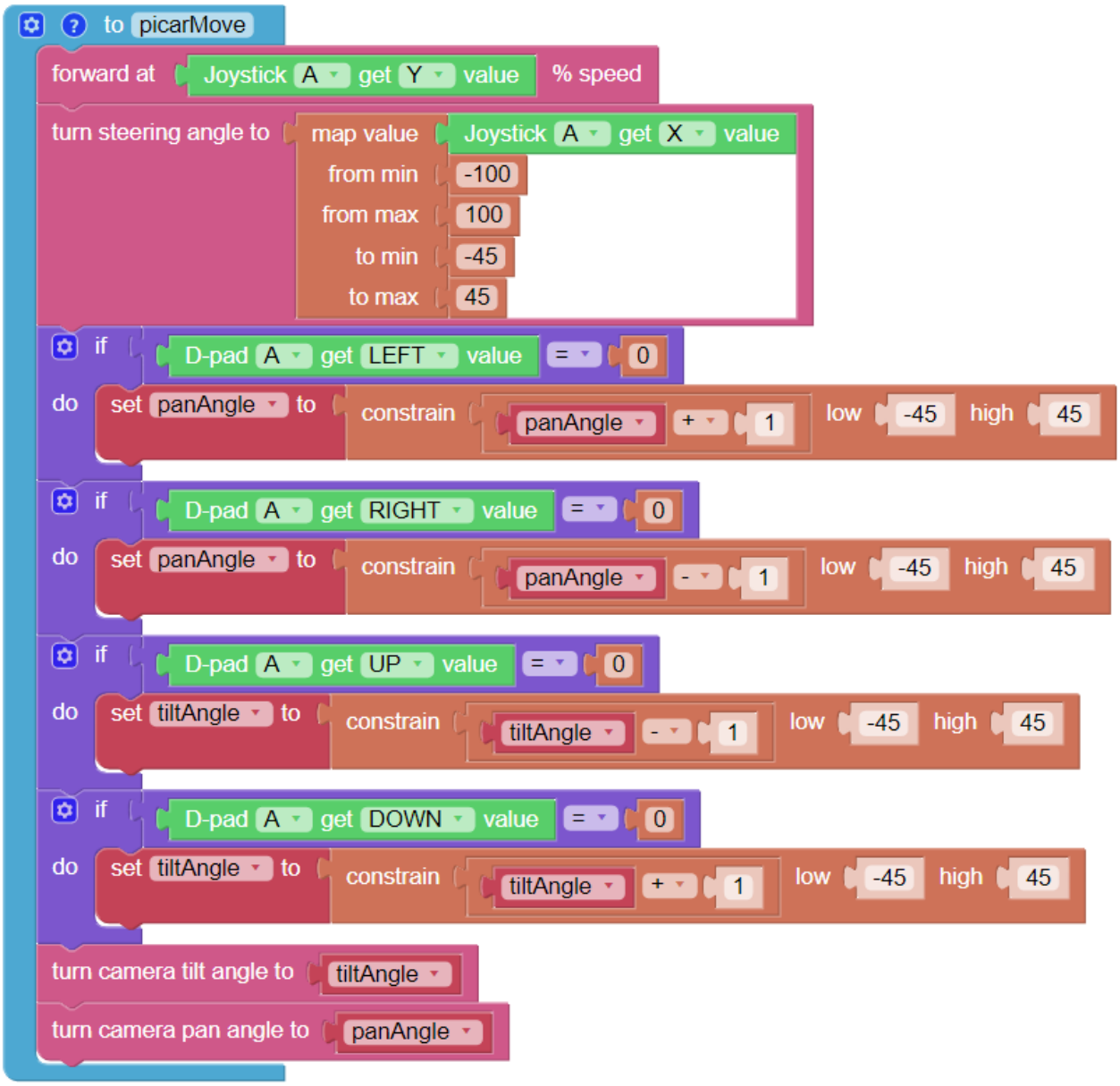
例

注釈:

- 以下の画像に従ってプログラムを書くことができます。チュートリアルを参照してください: How to Create a New Project?。
- EzBlock Studio の **Examples** ページで同じ名前のコードを見つけ、 **Run** または **Edit** を直接クリックしてください。


Start
camera monitor on
TTS language English US
set gameState to 0
set panAngle to 0
set tiltAngle to 0
Forever
if gameState = 0 and Button A is press
do gameStart
else if gameState ≠ 0
do Digital Tube A set value round time - startTime
picarMove
if width of detected color > 100
do play sound effects Emergency_Alarm.wav with volume 50 %
set gameState to remainder of gameState + 1 ÷ 4
boardcast


to gameStart
set colorList to create list with " red " " orange " " yellow " " green " " blue " " purple "
set task to create list with in list colorList get and remove random in list colorList get and remove random in list colorList get and remove random
say " Game Start "
set gameState to 1
set startTime to time
boardcast
to boardcast
if gameState = 0
do say " Game Finish "
else print create text with " Go " in list task get # gameState
say create text with " Go " in list task get # gameState
color detection in list task get # gameState


to picarMove
forward at Joystick A get Y value % speed
turn steering angle to map value Joystick A get X value
from min -100
from max 100
to min -45
to max 45
if D-pad A get LEFT value = 0
do set panAngle to constrain panAngle + 1 low -45 high 45
if D-pad A get RIGHT value = 0
do set panAngle to constrain panAngle - 1 low -45 high 45
if D-pad A get UP value = 0
do set tiltAngle to constrain tiltAngle - 1 low -45 high 45
if D-pad A get DOWN value = 0
do set tiltAngle to constrain tiltAngle + 1 low -45 high 45
turn camera tilt angle to tiltAngle
turn camera pan angle to panAngle

第 7 章

付録

7.1 Filezilla ソフトウェア

ファイル転送プロトコル（FTP）は、コンピュータネットワーク上のサーバーからクライアントへコンピュータファイルを転送するために使用される標準的な通信プロトコルです。

Filezilla はオープンソースソフトウェアで、FTP だけでなく、TLS 上の FTP（FTPS）や SFTP にも対応しています。Filezilla を使用して、ローカルファイル（写真や音声など）をラズベリーパイにアップロードしたり、ラズベリーパイからローカルにファイルをダウンロードすることができます。

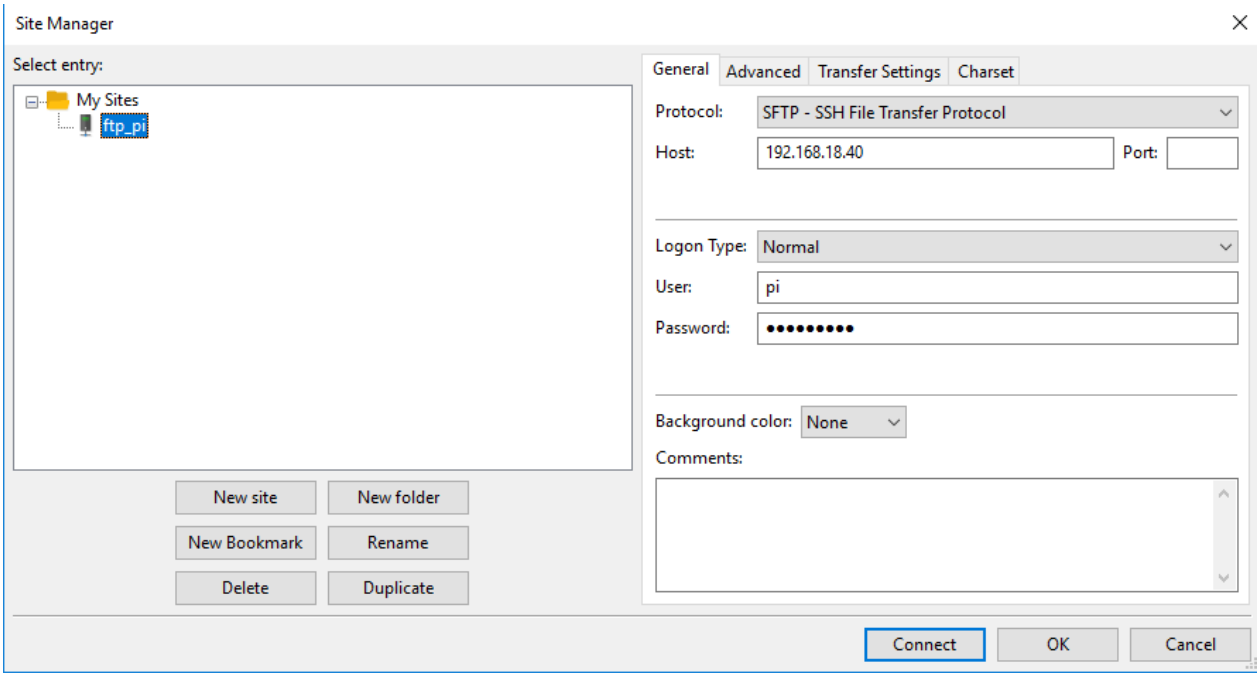
ステップ 1：Filezilla をダウンロードする。

Filezilla の公式ウェブサイト からクライアントをダウンロードしてください。Filezilla には非常に良いチュートリアルがありますので、こちらを参照してください： Documentation - Filezilla。

ステップ 2：ラズベリーパイに接続する

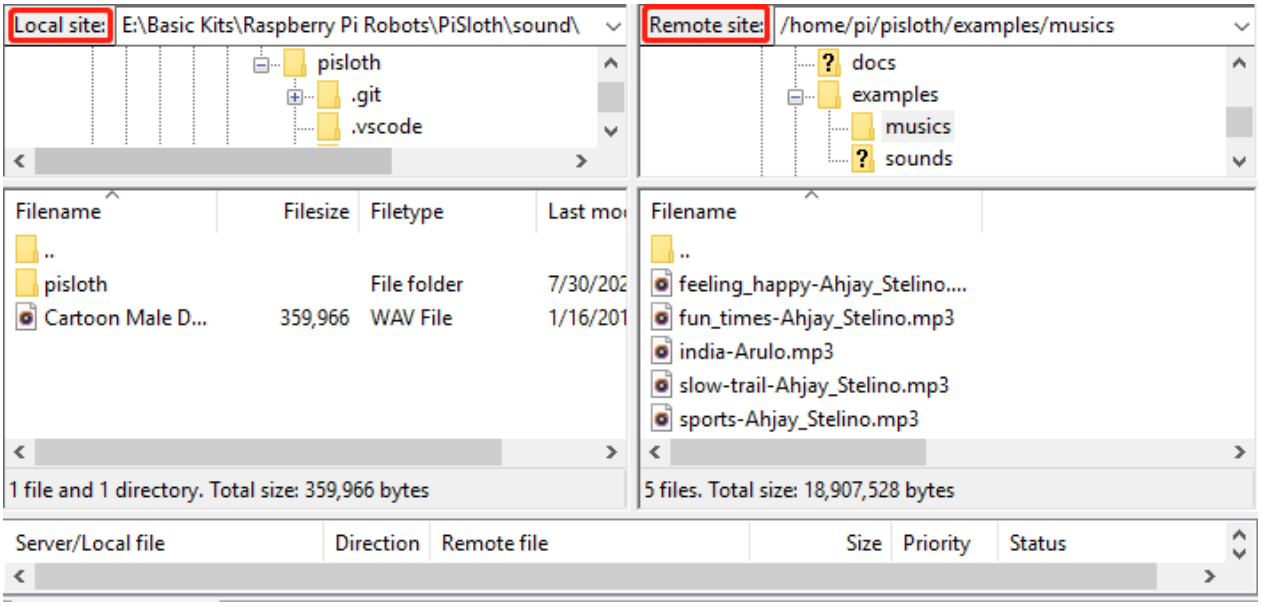
簡単なインストール後、開いて FTP サーバーに接続する。接続する方法は 3 つありますが、ここでは **Quick Connect** バーを使用します。 **hostname/IP**、 **username**、 **password**、 **port (22)** を入力し、 **Quick Connect** をクリックするか **Enter** を押してサーバーに接続します。

注釈: クイック接続は、ログイン情報をテストする良い方法です。恒久的なエントリーを作成したい場合は、成功したクイック接続後に **File-> Copy Current Connection to Site Manager** を選択し、名前を入力して **OK** をクリックします。次回は **File -> Site Manager** 内で以前に保存したサイトを選択することで接続できます。



ステップ 3：ファイルをアップロード/ダウンロードする。

ラズベリーパイにローカルファイルをドラッグアンドドロップでアップロードするか、ラズベリーパイ内のファイルをローカルにダウンロードします。



7.2 PuTTY

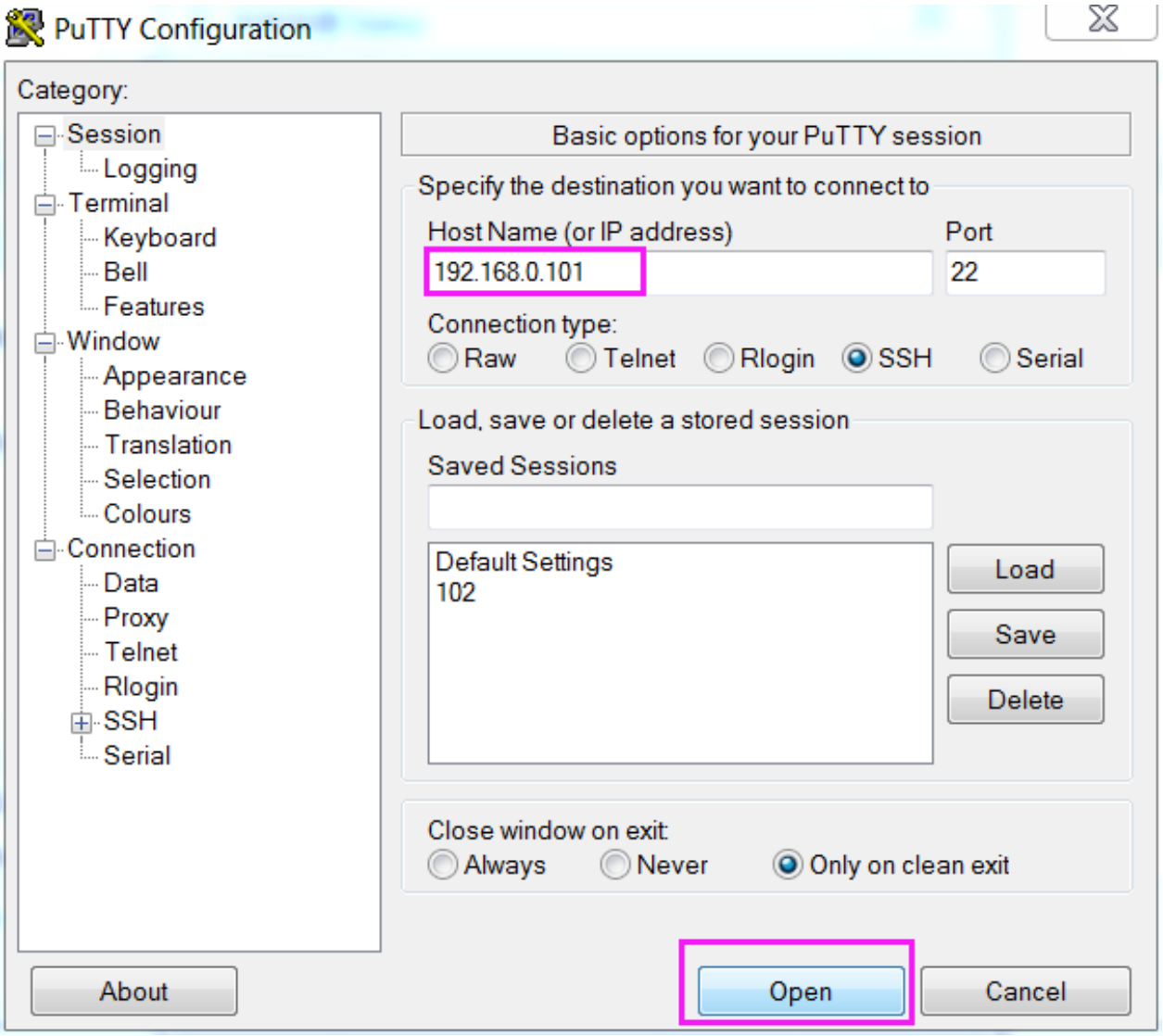
Windows ユーザーの場合、SSH アプリケーションをいくつか使用することができます。ここでは、PuTTY を推奨します。

ステップ 1

PuTTY をダウンロードします。

ステップ 2

PuTTY を開き、左側のツリー構造の **Session** をクリックします。 **Host Name (or IP address)** のテキストボックスに RPi の IP アドレスを、 **Port** の下には **22** を入力します（デフォルトでは 22 です）。



ステップ 3

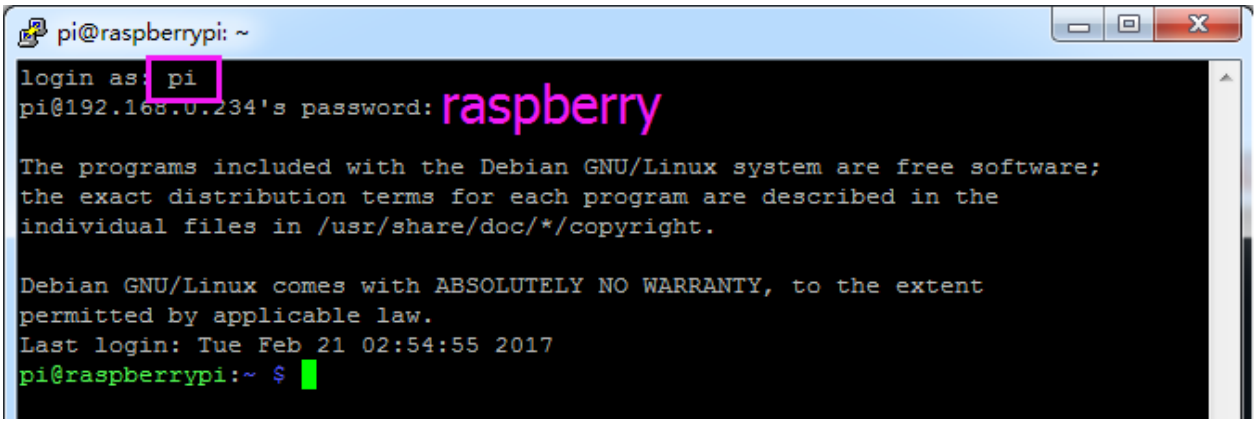
Open をクリックします。IP アドレスを使って Raspberry Pi に初めてログインする際、セキュリティリマインダーが表示されます。 **Yes** をクリックしてください。

ステップ 4

PuTTY ウィンドウが"**login as:**"と表示したら、"**pi**"（RPi のユーザー名）と **password**："raspberry"（変更していない場合のデフォルト）を入力します。

注釈: パスワードを入力するとき、ウィンドウ上に文字が表示されないのは正常です。正しいパスワードを入力することが必要です。

PuTTY の横に非アクティブと表示されている場合は、接続が切断されており、再接続する必要があります。



ステップ 5

これで Raspberry Pi に接続できました。次のステップを進める準備ができました。

7.3 IP アドレスの取得

Raspberry Pi が WI-FI に接続されたら、その IP アドレスを取得する必要があります。IP アドレスを知る方法はたくさんありますがそのうちの 2 つを以下の示します。

ただし最近であれば Raspberry pi の SD カードを作成した際にホスト名として付けた名前から簡単に接続できる事が多いので先ずはそちらを試してください。IP アドレス（数字）の代わりに SD カード作成時に指定したホスト名を使います。例： raspberrypi.local

場合によっては **ターミナル** や **Power Shell** の画面から `ping raspberrypi.local` と入力すると名前解決されて IP アドレスが表示されることもあります。

1. ルーターから調べる

ルーター（ホームネットワークなど）へのログイン権限があれば、ルーターの管理画面で Raspberry Pi に割り当てられたアドレスを確認できます。

Raspberry Pi OS のデフォルトのホスト名は **raspberrypi** ですのでそれを見つける必要があります。 (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・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ができます。 アプリの **Advanced IP scanner** などを利用できます。

対象 IP 範囲をスキャン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ます。 SD カード作成時にホスト名を変更していない場合、Raspberry Pi OS のデフォルトのホスト名は **raspberrypi** です。

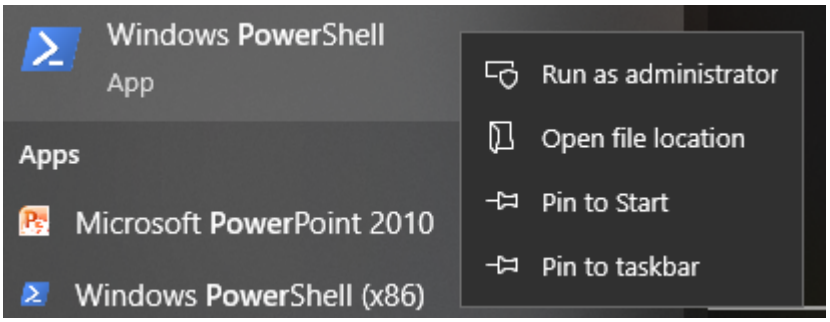
7.4 Powershell を使って OpenSSH をインストールする

ssh <username>@<hostname>.local （または ssh <username>@<IP address>）を使って Raspberry Pi に接続しようとすると、以下のエラーメッセージが表示されることがあります。

```
ssh: The term 'ssh' is not recognized as the name of a cmdlet, function, script␣
↪file, or operable program. Check the
spelling of the name, or if a path was included, verify that the path is correct␣
↪and try again.
```

これは、お使いのコンピュータシステムが古く、 OpenSSH がプリインストールされていないことを意味します。以下のチュートリアルに従って手動でインストールする必要があります。

1. Windows デスクトップの検索ボックスに powershell と入力し、 Windows PowerShell を右クリックして、表示されるメニューから Run as administrator を選択します。



2. 次のコマンドを使用して OpenSSH.Client をインストールします。

```
Add-WindowsCapability -Online -Name OpenSSH.Client~~~~0.0.1.0
```

3. インストール後、以下の出力が返されます。

```
Path          :
Online        : True
RestartNeeded : False
```

4. 次のコマンドを使用してインストールを確認します。

```
Get-WindowsCapability -Online | Where-Object Name -like 'OpenSSH*'
```

5. これで OpenSSH.Client が正常にインストールされたことがわかります。

```
Name  : OpenSSH.Client~~~~0.0.1.0
State : Installed
```

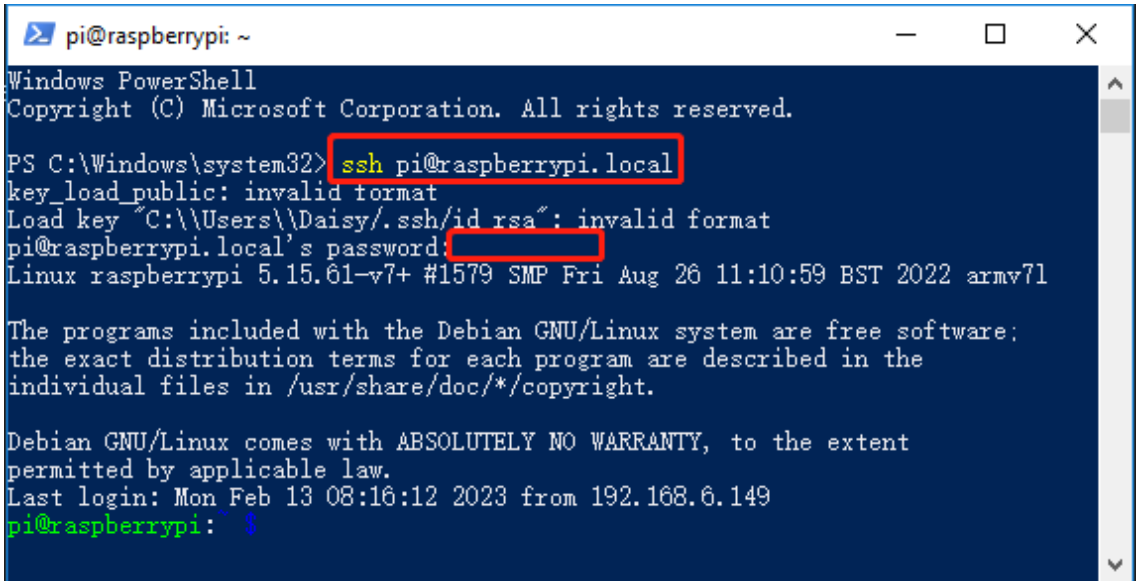
(次のページに続く)

(前のページからの続き)

```
Name   : OpenSSH.Server~~~~0.0.1.0
State  : NotPresent
```

警告: 上記のプロンプトが表示されない場合、Windows システムがまだ古いことを意味し、*PuTTY* のようなサードパーティの SSH ツールのインストールをお勧めします。

6. PowerShell を再起動し、管理者として実行し続けます。この時点で、 ssh コマンドを使用して Raspberry Pi にログインすることができ、以前に設定したパスワードを入力するように求められます。



7.5 バッテリーについて

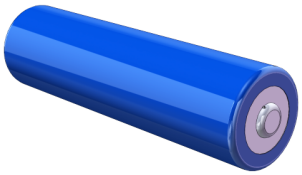
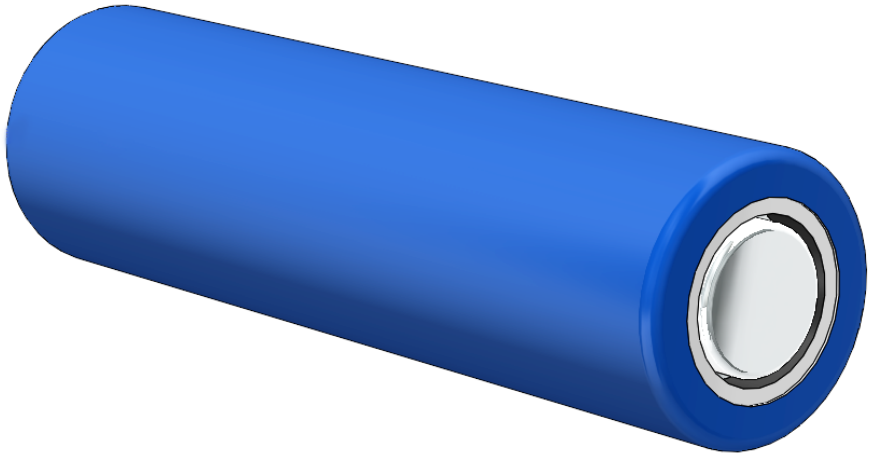
使用可能バッテリー

- 18650 リチウムイオンバッテリー（3.7V）充電可能タイプ
- ボタントップ（フラットトップでは無いもの）
- 保護回路の無いもの（国内で市販されているものはほとんどが保護回路付きなので購入時は注意してください。）
- 保護回路付きのものからご自分で保護回路を取り外して使用する事は自己責任となりますので注意が必要です。

注釈:

- バッテリーチャージャーは別途購入してください。
- ロボット・ハットのバッテリー表示用 LED が 2 つとも消灯したときはバッテリーがほとんどなくなっていますので、充電してください。なおリチウムイオンバッテリーは使い切ってから充電する必要はなく、むしろ継ぎ足し充電する方が寿命は伸びますので、遊んだ後は適時充電することをお勧めいたします。

ボタントップとは 18650 リチウムイオンバッテリーには＋電極が平らなものがあります。必ず通常の乾電池の様に＋電極が出っぱっているものを使用してください。また市販の 18650 リチウムイオンバッテリーには保護回路がついているものが多いので注意してください。保護回路付きのものからご自分で保護回路を取り外して使用する事は自己責任となりますので注意が必要です。

ボタントップ	フラットトップ
	

保護回路無し？

保護機能無しの 18650 バッテリーを使うことをお勧めしております。なぜなら保護機能により動作中のロボットの電源が切れてしまう事があるためです。また保護回路分バッテリー本体の長さが長くなり、ロボット付属のものはもとより現状の市販バッテリーホルダーに収まらなくなります。

バッテリーの容量は？

ロボットを長時間動作させるために、できるだけ大容量のバッテリーを使用してください。 3000mAh 以上の容量のバッテリーを購入することをお勧めします。

第 8 章

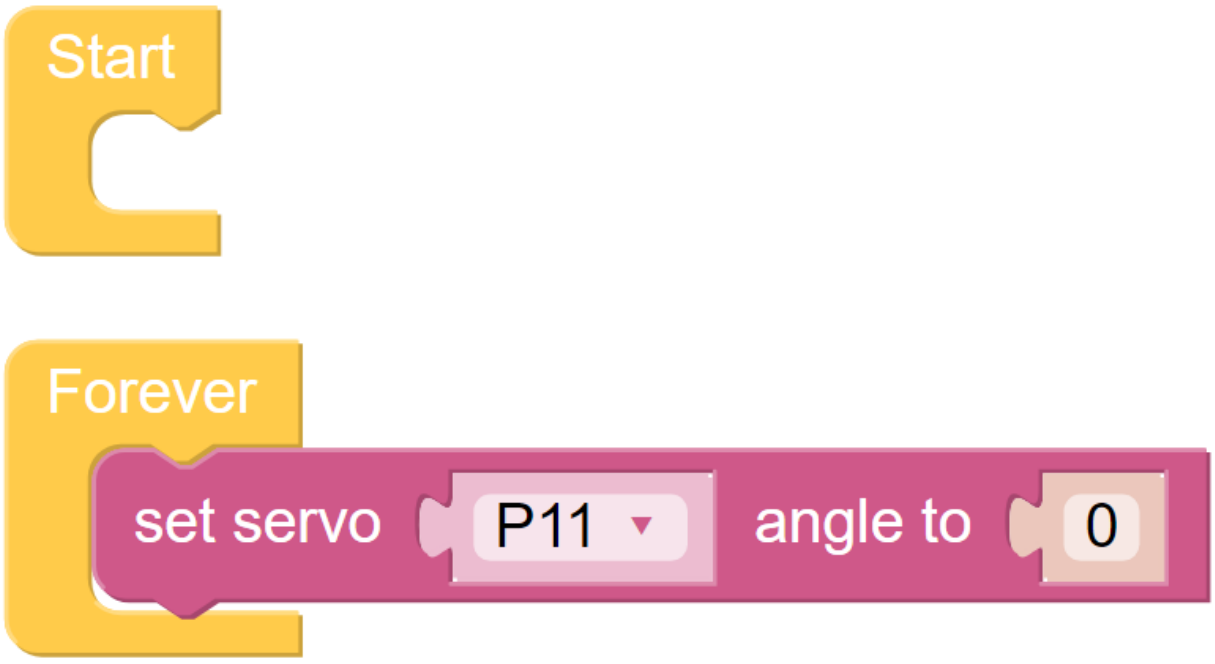
FAQ

8.1 Q1: Ezblock OS をインストールした後、サーボは 0 ° に回転しないのはなぜですか？

1) サーボのケーブルが正しく接続されているか、および Robot HAT の電源がオンになっているかを確認してください。

2) リセットボタンを押します。

3) Ezblock Studio でプログラムをすでに実行している場合、P11 のカスタムプログラムは利用できなくなります。以下の画像を参考にして、Ezblock Studio でサーボの角度を 0 に設定するプログラムを手動で書くことができます。



8.2 Q2: VNC を使用する際に、現在デスクトップを表示できないというメッセージが表示されますか？

ターミナルで ``sudo raspi-config``と入力して、解像度を変更してください。

8.3 Q3: サーボが突然中央位置に戻るのはなぜですか？

サーボが構造や他のオブジェクトによってブロックされ、目的の位置に到達できない場合、サーボは電流が過大となり焼き切れるのを防ぐために、電源オフ保護モードに入ります。

一定期間電源が遮断されていると、サーボに PWM 信号が与えられていない場合、サーボは自動的に元の位置に戻ります。

8.4 Q4: Robot HAT の詳細チュートリアルについて？

Robot HAT に関する包括的なチュートリアルはこちらで見ることができます。ハードウェアや API に関する情報も含まれています。

-

第９章

ありがとうございます

私たちの製品を評価してくださった評価者の皆様、チュートリアルの提案をしてくださったベテランの皆様、そして私たちをずっとフォローしサポートしてくださったユーザーの皆様に感謝します。皆様からの貴重な提案は、私たちがより良い製品を提供するための動機となっています！

特別な感謝

- レン・デイヴィスン
- カレン・ダニエル
- フアン・デラコスタ

少しだけお時間をいただき、このアンケートにご協力いただけますか？

注釈: アンケートを提出した後は、トップに戻って結果をご覧ください。

第 10 章

著作権通知