SunFounder picar-x

www.sunfounder.com

2024 年 03 月 05 日

目次

第 1 章　序章 5

1.1 自動運転車の歴史 5

1.2 PiCar-X について 7

1.3 深層学習とニューラル ネットワーク 8

第 2 章　部品リストと組み立て説明書 9

第 3 章　Robot HAT について 11

第 4 章　Python を楽しもう 15

4.1 Python に関するクイックガイド 15

4.1.1 必要なものは何ですか？ 15

4.1.2 OS のインストール 17

4.1.3 Raspberry Pi のセットアップ 26

4.1.4 すべてのモジュールをインストールする（重要） 51

4.1.5 I2C インターフェースを有効にする（重要） 53

4.1.6 サーボ調整（重要） 56

4.2 0. PiCar-X の校正 58

4.2.1 モーターとサーボの校正 58

4.2.2 グレースケールモジュールの校正 62

4.3 1. PiCar-X を動かす 65

4.4 2. キーボード制御 68

4.5 3. テキストから音声へ & 効果音 71

4.6 4. 障害物回避 76

4.7 5. ライン追跡 79

4.8 6. 崖検出 84

4.9 7. コンピュータービジョン 86

4.10 8. あなたを見つめる 95

4.11 9. ビデオ録画 98

4.12 10. ブルファイト 100

4.13 11. ビデオカー 103

4.14 12. 宝探し 108

4.15 13. アプリによる制御 112

第 5 章　Ezblock の利用に関して　117
5.1 EzBlock のクイックガイド 117
5.2 EzBlock Studio のインストールと設定 121
5.3 PiCar-X のキャリブレーション 121
5.4 ムーブ 127
5.5 リモコン・カー 131
5.6 超音波モジュール試験 133
5.7 グレースケール・モジュール・テスト 135
5.8 色検出 136
5.9 顔検出 139
5.10 音声読み上げ 142
5.11 バックグラウンド・ミュージック 144
5.12 ハロー！ 146
5.13 ミュージック・カー 149
5.14 崖検知 150
5.15 トロッコ 152
5.16 トロッコ＋ 155
5.17 闘牛 160
5.18 歩行者に注意 162
5.19 交通標識識別 164
5.20 オリエンテーリング・ゲーム 169

第 6 章　付録　175
6.1 Filezilla ソフトウェア 175
6.2 PuTTY 177
6.3 IP アドレスの取得 179
6.4 Powershell を使って OpenSSH をインストールする 180
6.5 バッテリーについて 181

第 7 章　FAQ　183

第 8 章　謝辞　185

第 9 章　Copyright Notice　187

私どもの PiCar-X をお選びいただき誠にありがとうございます。

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。

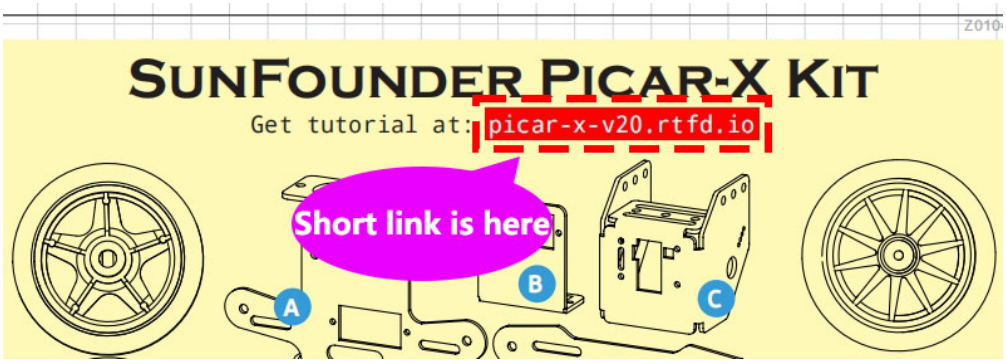
-
-
-

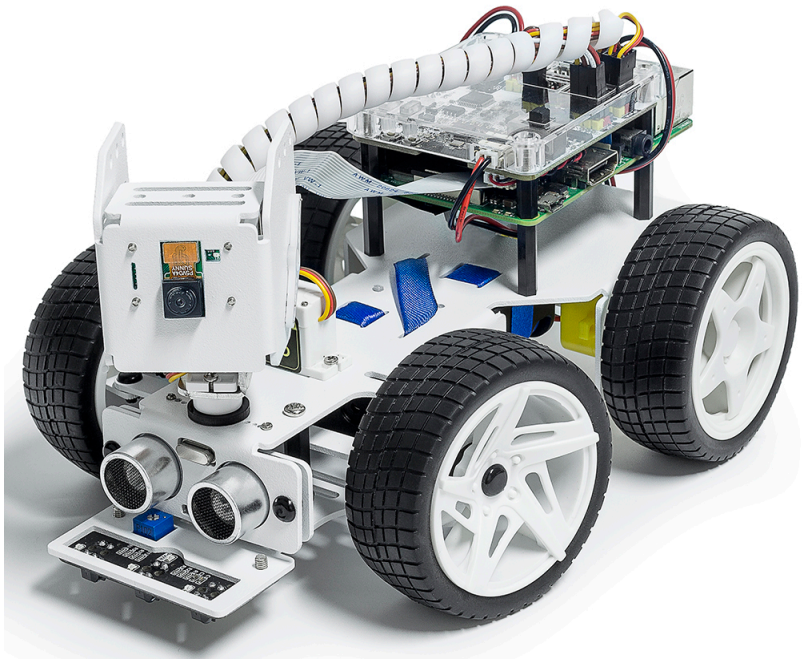
ご希望の言語でドキュメントにアクセスするには、該当のリンクをクリックしてください。

警告: PiCar-X には 2 つのバージョンがございます。各バージョンのオンラインチュートリアルのスクリプトは互換性がありませんので、ご注意ください。

適切なセットアップを確保するため、取扱説明書に記載されている短縮リンクを使用してバージョンを特定してください：

- リンクが"picar-x.rtfd.io"の場合は、このチュートリアルを続行してください。
- リンクが"picar-x-v20.rtfd.io"の場合は、 のチュートリアルをご参照ください。





PiCar-X は、Raspberry Pi プラットフォーム用の AI 駆動の自動運転ロボットカーで、Raspberry Pi がコントロール 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、およびライン トラッキング モジュールは、色/顔/交通標識検出の機能を利用できます。また自動障害物回避、自動ライントレースなども試せます。

PiCar-X には「Blockly（ビジュアルプログラミング）」と「Python」の 2 つのプログラミング言語が利用可能です。プログラミングの言語に関係なく、Raspberry Pi の構成から関連するサンプル コードの実行まで、すべてを説明する詳細な手順をご用意しました。プログラム経験のない方は手始めに「Blockly」でプログラムの基礎を学び、その後に本格的な「Python」を学ぶための基礎づくりができます。「Python」は最近注目を集めているプログラミング言語で特に AI 関連のモジュールが充実していると言われております。小学校高学年から社会人でプログラミングや AI に興味のある方まで幅広く、楽しく学ぶことができると信じております。

- ***Python* を楽しもう**
  - この章は、Python でのプログラミングを楽しむ人、または Python 言語を学びたい人向けです。
  - Picar-X を正しく動作させるには最初にいくつかのライブラリをインストールする必要があります。
  - この章では、PiCar-X の Raspberry Pi の構成とサンプル コードを提供しています。
  - アプリ - SunFounder コントローラーも提供しておりスマホやタブレットで PiCar-X をリモートコントロールすることもできます。
- ***Ezblock* の利用に関して**
  - このセクションでは、ビジュアルプログラミングのアプリである Ezblock Studio を使用します。Ezblock Studio は、小学校で習う Scratch と同様にブロックをドラッグ・アンド・ドロップして Picar-X を動かすことができます。

- プログラミングの前に Ezblock 環境で提供されるオペレーティングシステムを SD カードにインストールする必要があります。 このセクションでは新品または現在は使用していない SD カードを使用することをお勧めします。
- Ezblock Studio は、Mac、PC、Android などほぼすべてのタイプのデバイスで利用できます。
- Ezblock Studio は、6 歳から 12 歳（あるいはそれ以上の）プログラミング スキルがない人でも Picar-X で簡単に遊びたい（学びたい）人に適しています。

Content

第 1 章

序章

1.1 自動運転車の歴史

自動運転車の実験は少なくとも 1920 年代から行われてきました。有望な試験が 1950 年代に行われ、それ以来進展し続けています。1980 年代になると、1984 年のカーネギー メロン大学の Navlab と 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 プロジェクトによって最初の自立型自動運転車が登場しました。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、オートリブ、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、Google などの実用的な自動運転車を開発しています。2013 年 7 月、Vislab は一般に公開された混合交通ルートを自律的に移動する車両、BRAiVE のデモンストレーションを行いました。2019 年の時点で米国の 29 の州では公道での自動運転車を許可する法律がすでに可決されています。

一部の UNECE 加盟国、および英国を含む EU 加盟国は自動化および完全自動化された自動車に関連する規則および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、英国の都市が自動運転車の輸送システムを運用する計画を立てており、ドイツ、オランダ、スペインはすでに公共交通機関でのロボット車のテストを許可しています。2020 年には英国、EU、日本はすでに自動運転車の規制に向けて順調に進んでいます。

- 参考： 自動運転車の歴史 - ウィキペディア

現在、自動運転車は目前に迫った技術革命です。 一部の専門家は、2025 年までにレベル 4 の自動車が市場に参入する可能性が高いと予測しています。 レベル 4 の自動車ではシステムが正常に機能している限り、ドライバーは完全に別のことに注意を向けることができ、交通状況に注意を払う必要がなくなります。

レベル 4 リファレンス:

- Driving Automation の SAE レベル ™
- ABI Research は、2025 年に SAE レベル 3、4、および 5 の自律技術を搭載した 800 万台の車両が出荷されると予測しています。

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)、およびクラウド コンピューティングにおける最近の急速な進歩がこの自動運転の技術革命を推進させています。

- 2010 年 10 月、イタリアのテクノロジー企業 **Vislab** によって設計された無人トラックは、イタリアから中国まで 3 か月かけて走破し `travel from Italy to China, 合計走行距離は 8,077 マイルでした。
- 2015 年 4 月、 **Delphi Automotive** が設計した自動車が 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% をコンピューター制御の下で走破しました。
- 2018 年 12 月、 **Alphabet**' の **Waymo** は アリゾナ州でレベル 4 の自動運転タクシー サービスを開始しました。 アリゾナ州では 2008 年からすでに自動運転車のテストを行っており、運転席に誰もいない状態での車両が 1 年以上稼働し、既に 1,000 万マイル以上走行しました。
- 2020 年 10 月、**Baidu** 北京で Apollo Robotaxi 自動運転タクシー サービス で開始しました。 運転ルートは、地元の住宅、商業、レジャー、工業団地をカバーし、完全自動運転システムを提供しています。

しかしながら実際の運転記録やシミュレートされたシナリオからのトレーニングデータなど、毎日膨大な量のデータが収集されているにもかかわらず、自動運転車の AI モデルは完全には満たされていません。

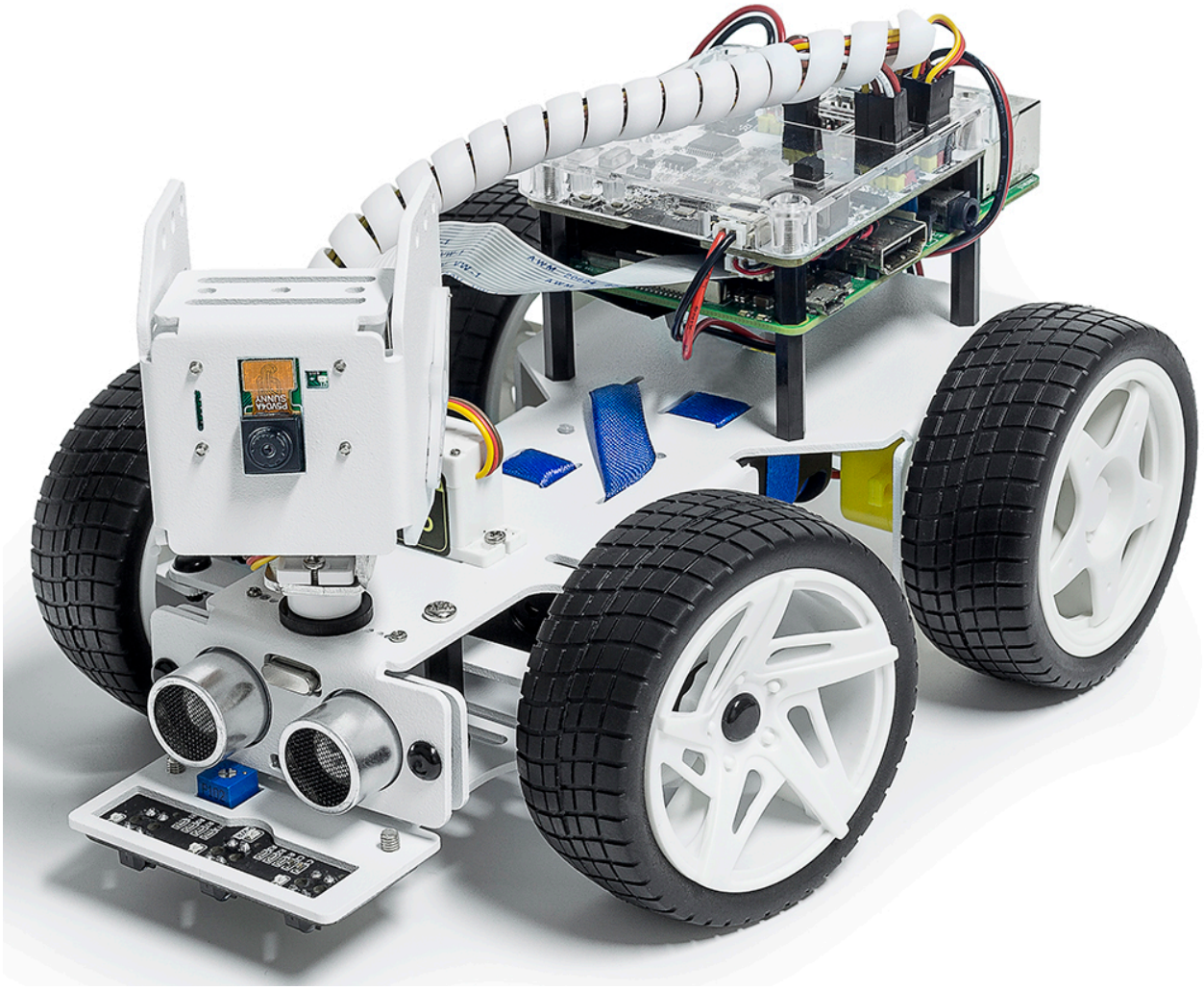
RAND のレポート によると適切なレベルの自律学習に到達するにはさらに数億マイル、場合によっては数千億マイルのトレーニング データが必要です。

したがって、完全な自動運転車の未来は有望で刺激的ですが、技術が成熟して全自動運転車が利用できるようになるまでにはまだ何年もの開発期間が必要です。

新しいテクノロジーを急速に成熟させるための実証済みの方法は、市場参入要件を最小限に抑えて、誰もが簡単にアクセスできるようにすることです。 これが PiCar-X を立ち上げた SunFounders の動機です。

SunFounders の目標は、初心者、初級者、および自動運転について学びたいだけの人を支援し、自動運転車の開発プロセス、技術、および最新のイノベーションを理解することです

1.2 PiCar-X について



PiCar-X は Raspberry Pi プラットフォーム用の AI 制御の自動運転ロボットカーで Raspberry Pi がコントロール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、およびライン・トラッキングモジュールは、色/顔/交通標識の検出、自動障害物回避、自動ライン・トラッキングなどの機能を提供できます。

SunFounder が設計した 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、およびカメラのパン/チルト機能を統合し、Robot HAT の ADC、PWM、およびデジタル I2C ピンを事前設定して Raspberry Pi の標準機能を拡張します。 スピーカーと Bluetooth チップの両方が Robot HAT に組み込

まれており、テキスト読み上げ、効果音、さらにはバックグラウンド ミュージック機能のリモートコントロールが可能です。

GPIO 制御、コンピューター ビジョン、ディープ ラーニングを含むすべての PiCar-X 機能は、オープンソースの Python プログラミング言語、OpenCV のコンピューター ビジョン ライブラリ ソフトウェア、およびディープ ラーニング フレームワーク用の Google の TensorFlow を通じて実装されます。 PiCar-X の機能を最適化するために他のソフトウェアが含まれており、ユーザーはほぼ無限の学習環境を利用できます。

1.3 深層学習とニューラル ネットワーク

深層学習とニューラル ネットワークの詳細については、SunFounder は次のリソースを推奨しています。

機械学習 - Andrew Ng：このコースでは、機械学習、データマイニング、統計的パターン認識について幅広く紹介しています。

ニューラル ネットワークとディープ ラーニング：この電子書籍ではコンピュータが観測データから学習できるようにする生物学に着想を得たプログラミング パラダイムであるニューラル ネットワークと、ニューラル ネットワークにおける機械学習のための強力な一連の手法であるディープ ラーニングの両方について説明されています。

コンピューター ビジョンのインセプション アーキテクチャの再考：この高レベルのホワイト ペーパーでは、因数分解された畳み込みと積極的な正則化によって追加の計算を可能な限り効率的に利用することで、ユーザーがネットワークをスケールアップできる方法について説明されています。

第 2 章

部品リストと組み立て説明書

PiCar-X を組み立てる前にすべての部品と部材が含まれていることを確認してください。 不足している、または破損している部品がある場合はすぐに SunFounder (service@sunfounder.com) に連絡して、問題をできるだけ早く解決してください。なお申し訳ないことにまれにネジなどの部品の袋のラベルが間違って貼られていることがありますので、中の部品と説明書の絵をよく見比べてください。

組み立て手順については次の PDF の手順書に従ってください。

`[PDF]Component List and Assembly of PiCar-X.`

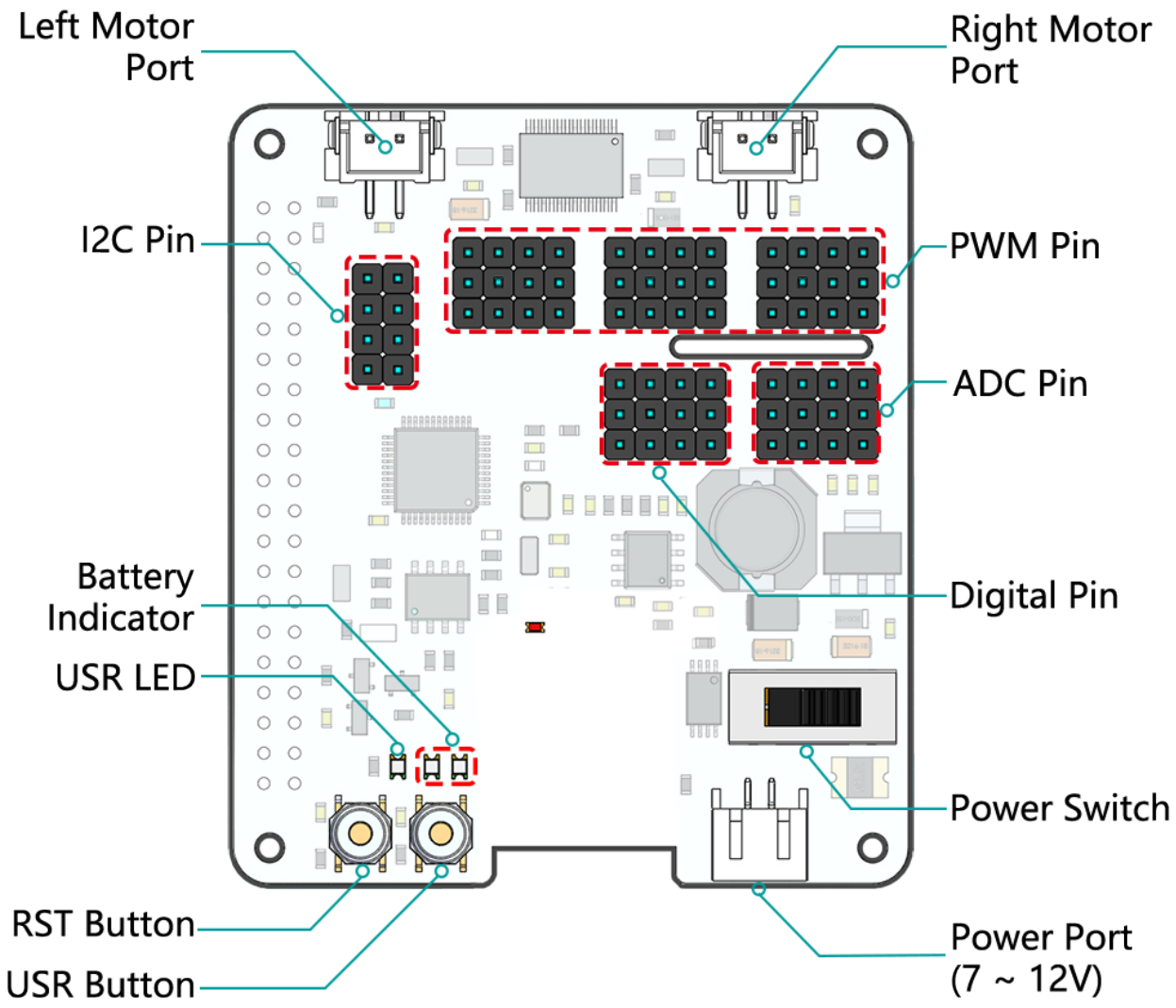
注釈: 組み立て後にロボット HAT を介してサーボの電源を入れた場合、手動でステアリング・ギアなどを無理に動かさないでください。サーボが損傷する可能性があります。

注釈:

1. 組み立てる前に 18650 リチウムイオンバッテリーを 2 本購入して完全に充電する必要があります。バッテリーについては **バッテリーについて** を参照してください。
2. ロボットハットを利用してバッテリーに充電できないので充電器も同時に購入する必要があります。

第 3 章

Robot HAT について



Left/Right Motor Port

- 2 チャンネル XH2.54 モーターポート。
- 左側のポートが GPIO 4 に、右側のポートが GPIO 5 に接続されています。

I2C Pin

- Raspberry Pi の 2 チャンネル I2C ピン。

PWM Pin

- 12 チャンネル PWM ピン、P0-P12。

ADC Pin

- 4 チャンネル ADC ピン、A0-A3。

Digital Pin

- 4 チャンネルデジタルピン、D0-D3。

Battery Indicator

- 7.8V 以上の電圧がある場合、2 つの LED が点灯します。
- 6.7V～7.8V の範囲で 1 つの LED が点灯します。
- 6.7V 以下では、全ての LED が消灯します。

USR LED

- プログラミングから利用できます。 (「1」を設定すると LED が点灯し、「0」を設定すると LED が消灯します。)

RST Button

- RST ボタンを短く押すと実行中のプログラムがリセットされます。

USR Button

- USR ボタンの機能はプログラで利用できます。 （押すと「0」が、離すと「1」が設定されます。）

Power Switch

- ロボット HAT の電源を ON/OFF します。
- 電源ポートに電源を接続すると、Raspberry Pi が起動します。ただし、ロボット HAT を有効にするために、電源スイッチを ON にする必要があります。

Power Port

- 7-12V PH2.0 2 ピン電源入力。
- Raspberry Pi と Robot HAT を同時に給電します。

注釈: 詳細については Robot HAT Documentation 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 に関するクイックガイド

このセクションでは、Raspberry Pi OS のインストール、Raspberry Pi への wifi の設定、Raspberry Pi へのリモートアクセスをして対応するコードを実行する方法を教えます。

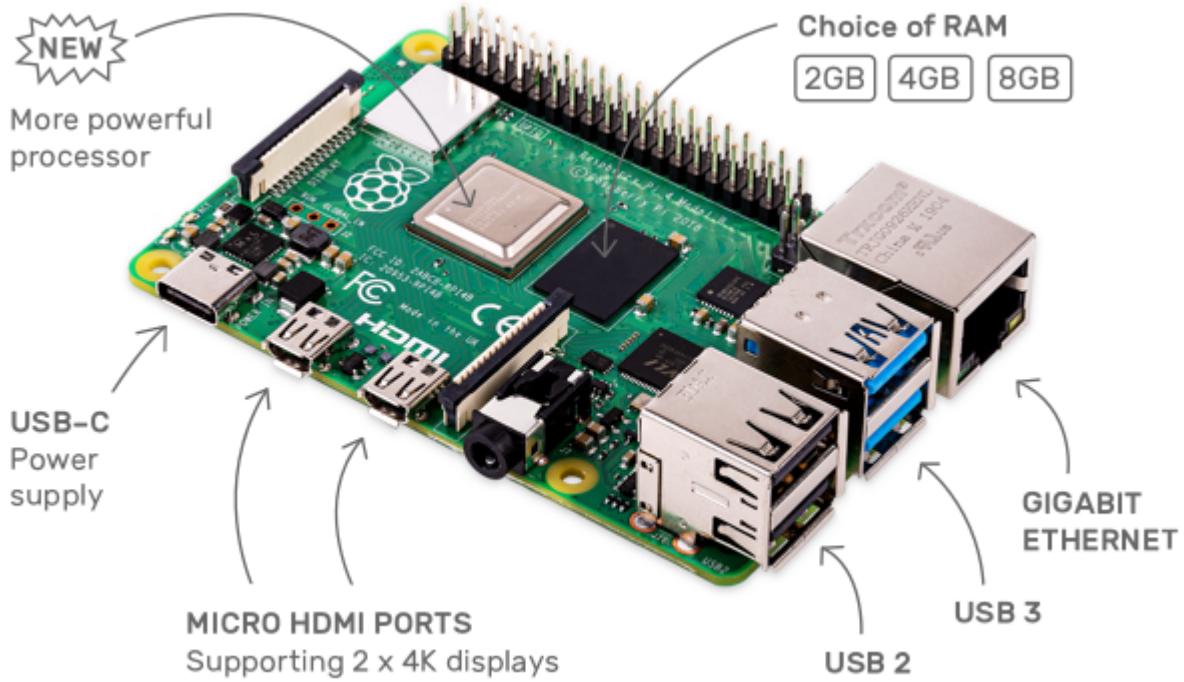
Raspberry Pi に慣れており、コマンドラインを開くことができる場合は、最初の 3 部分をスキップして、最後の部分だけを完了してください。

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。

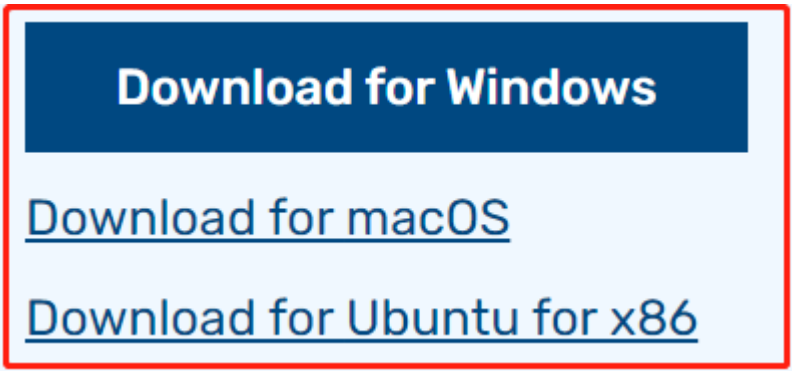
4.1.2 OS のインストール

必要なコンポーネント

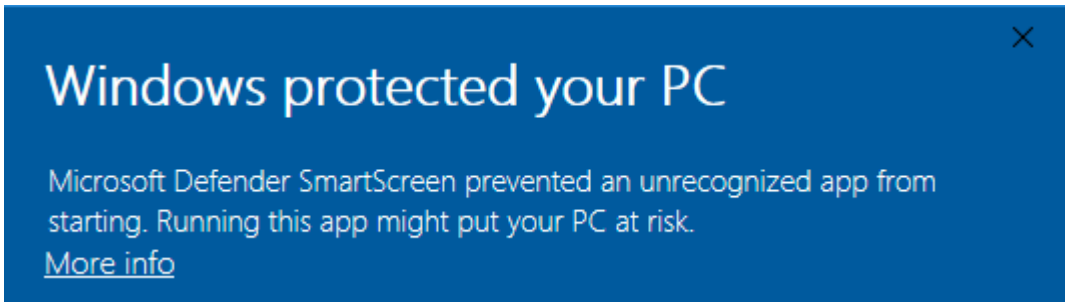
- Raspberry Pi 4B/Zero 2 w/3B 3B+/2B/Zero W
- 1 x パーソナルコンピュータ
- 1 x マイクロ SD カード

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。

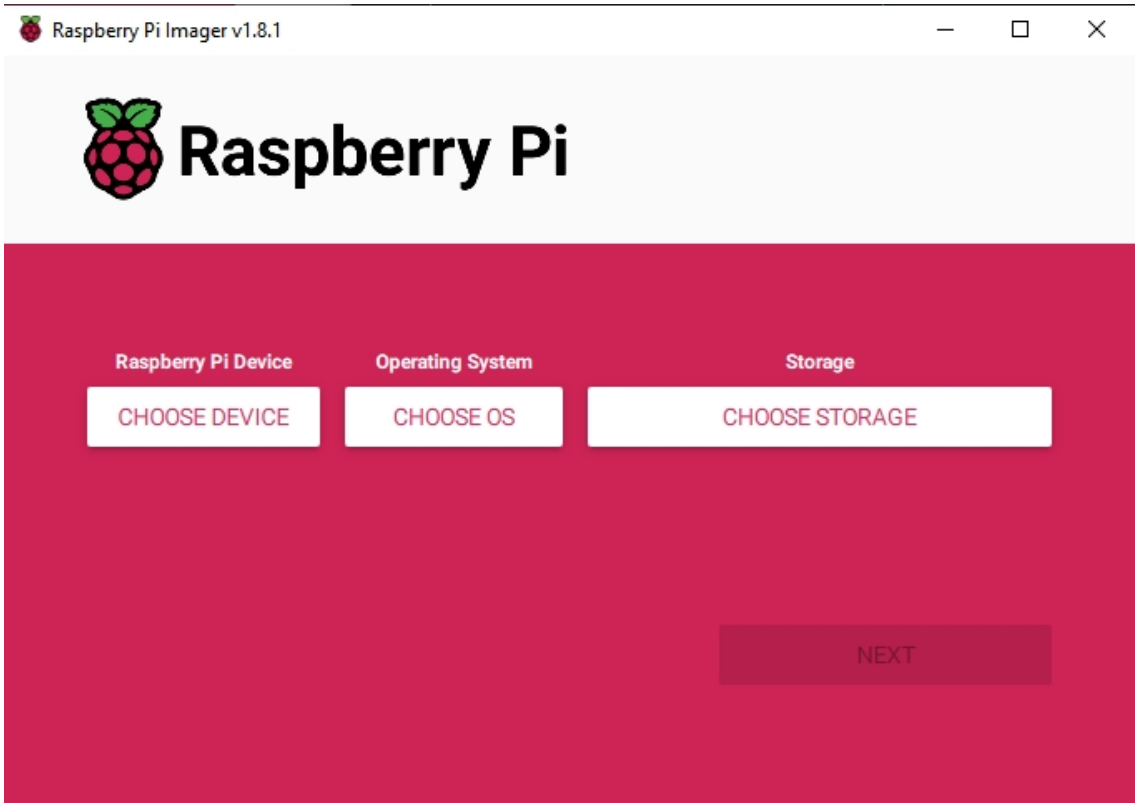


2. インストーラを起動すると、OS がセキュリティ警告を表示する場合があります。たとえば、Windows では警告メッセージが表示されることがあります。このような場合は、 **More info** を選択し、その後 **Run anyway** を選択します。画面の指示に従って Raspberry Pi Imager をインストールします。

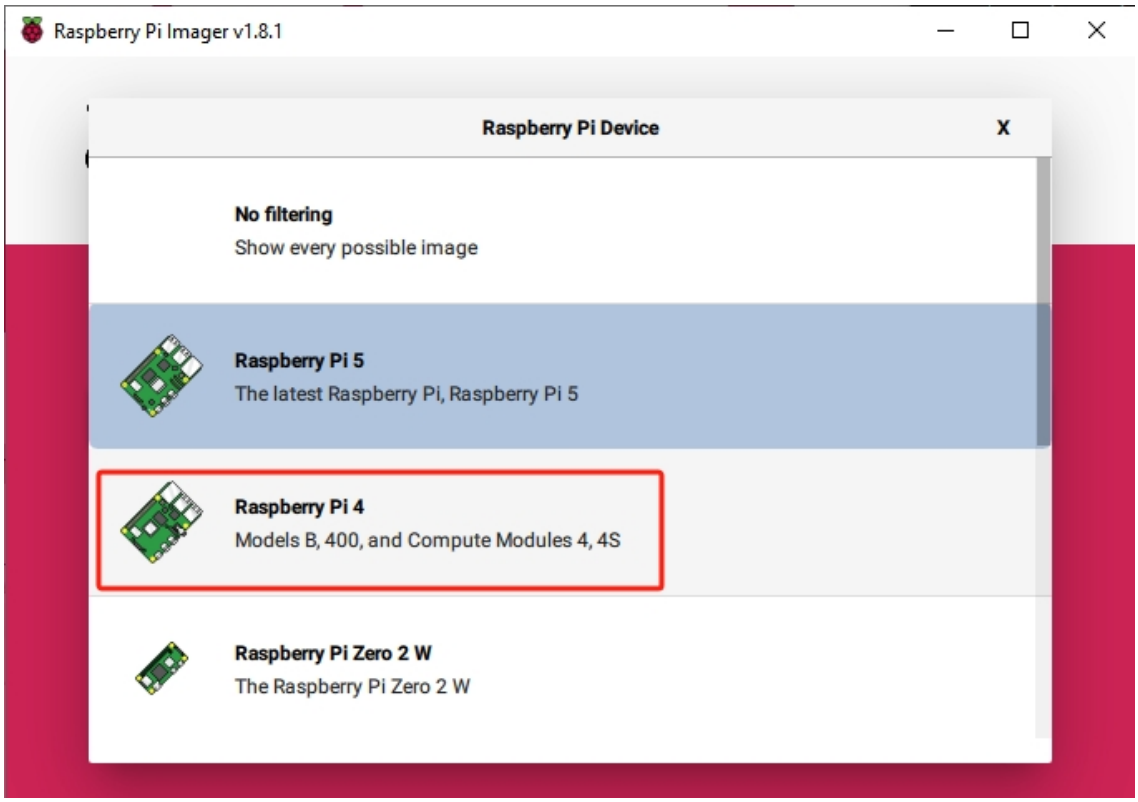


3. SD カードをコンピュータまたはラップトップの SD カードスロットに挿入します。

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。

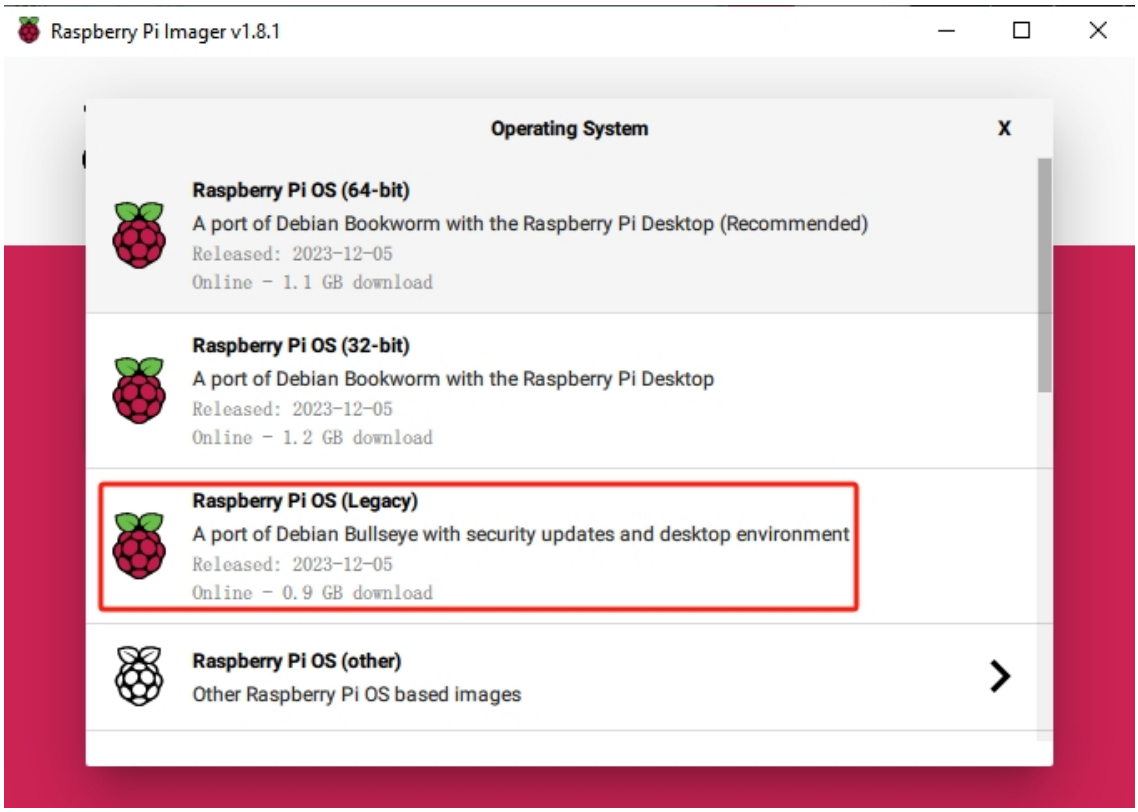


5. **CHOOSE DEVICE** をクリックし、リストから特定の Raspberry Pi モデルを選択します（注意：Raspberry Pi 5 は適用されません）。



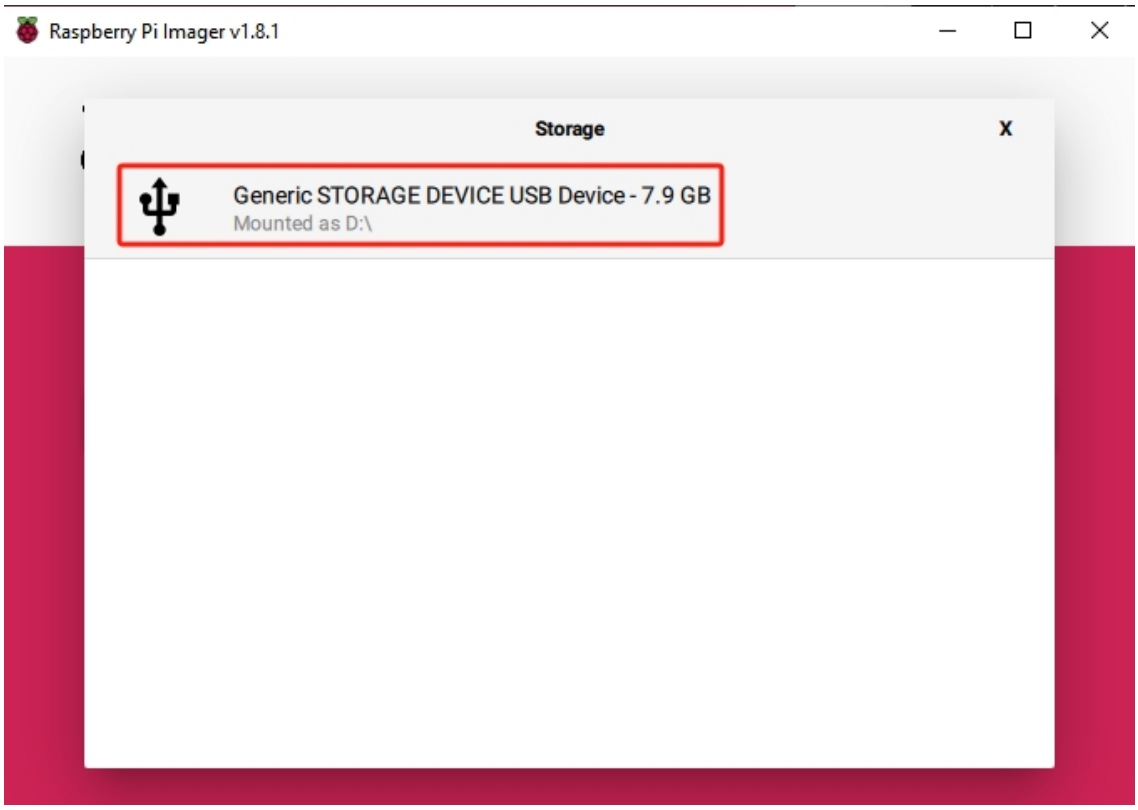
6. **CHOOSE OS** を選択し、次に **Raspberry Pi OS (Legacy)** を選択します。

> **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。

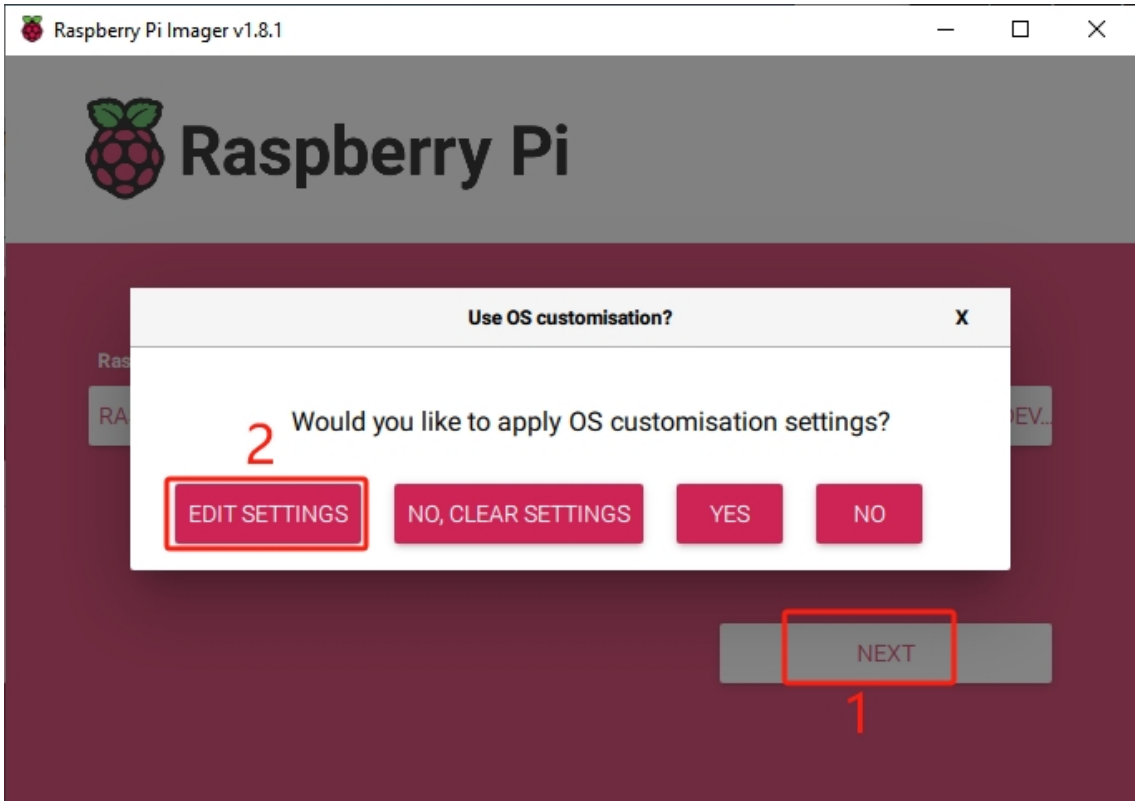


7. **Choose Storage** をクリックし、インストール用の正しいストレージデバイスを選択します。

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。

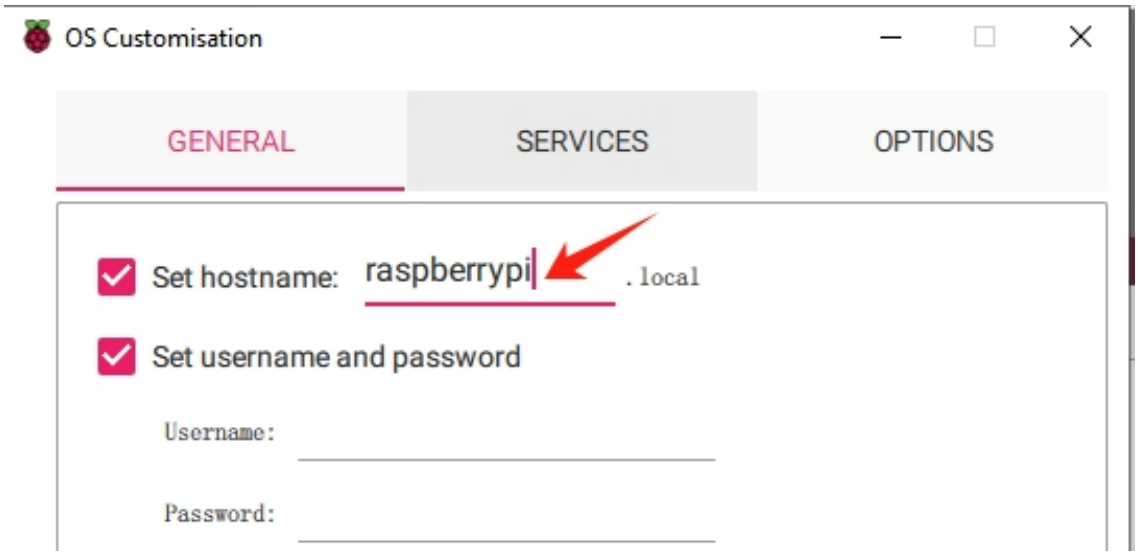


8. **NEXT** を押し、 **EDIT SETTINGS** を選択して OS の設定をカスタマイズします。



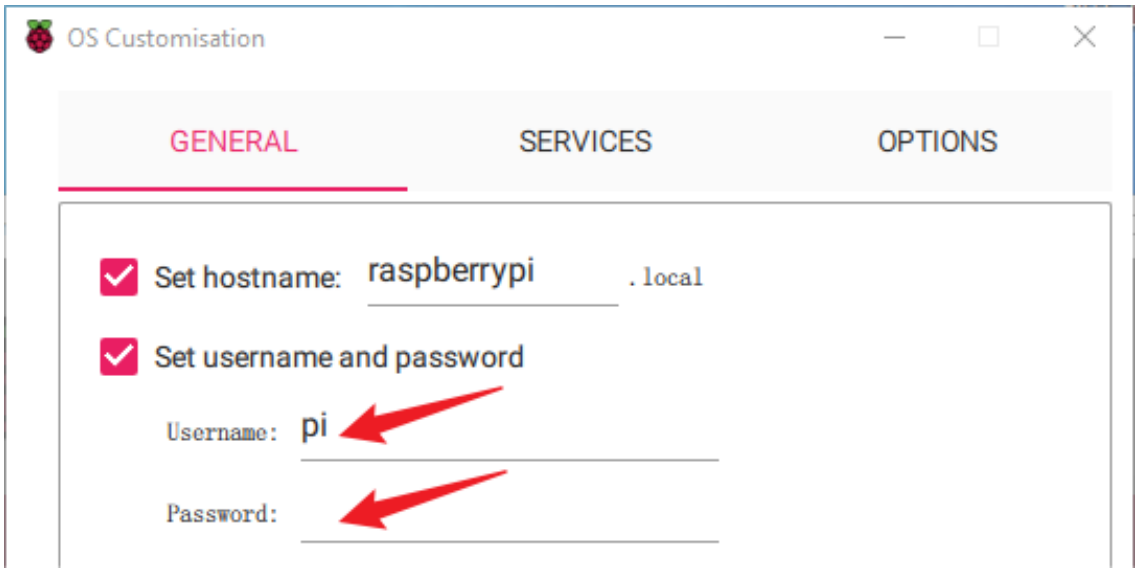
9. Raspberry Pi の **hostname** を設定します。

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。



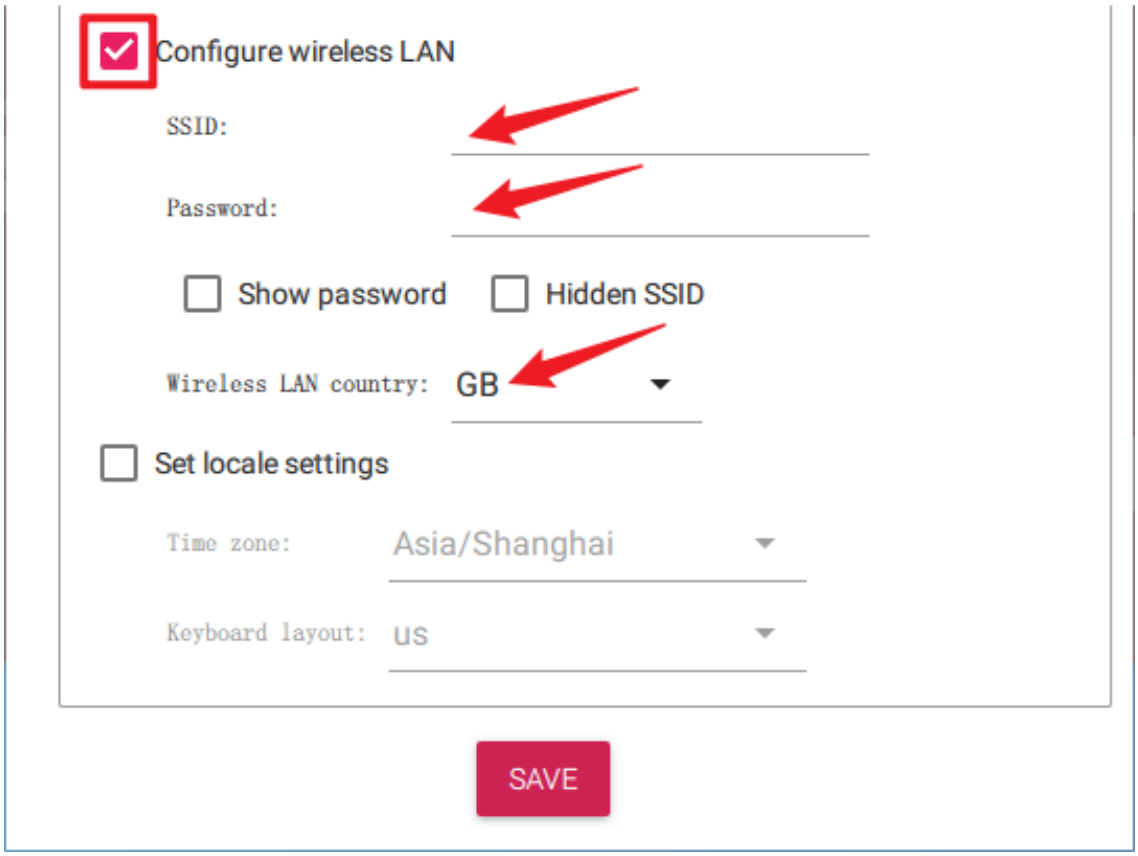
10. Raspberry Pi の管理者アカウント用に **Username** と **Password** を作成します。

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。

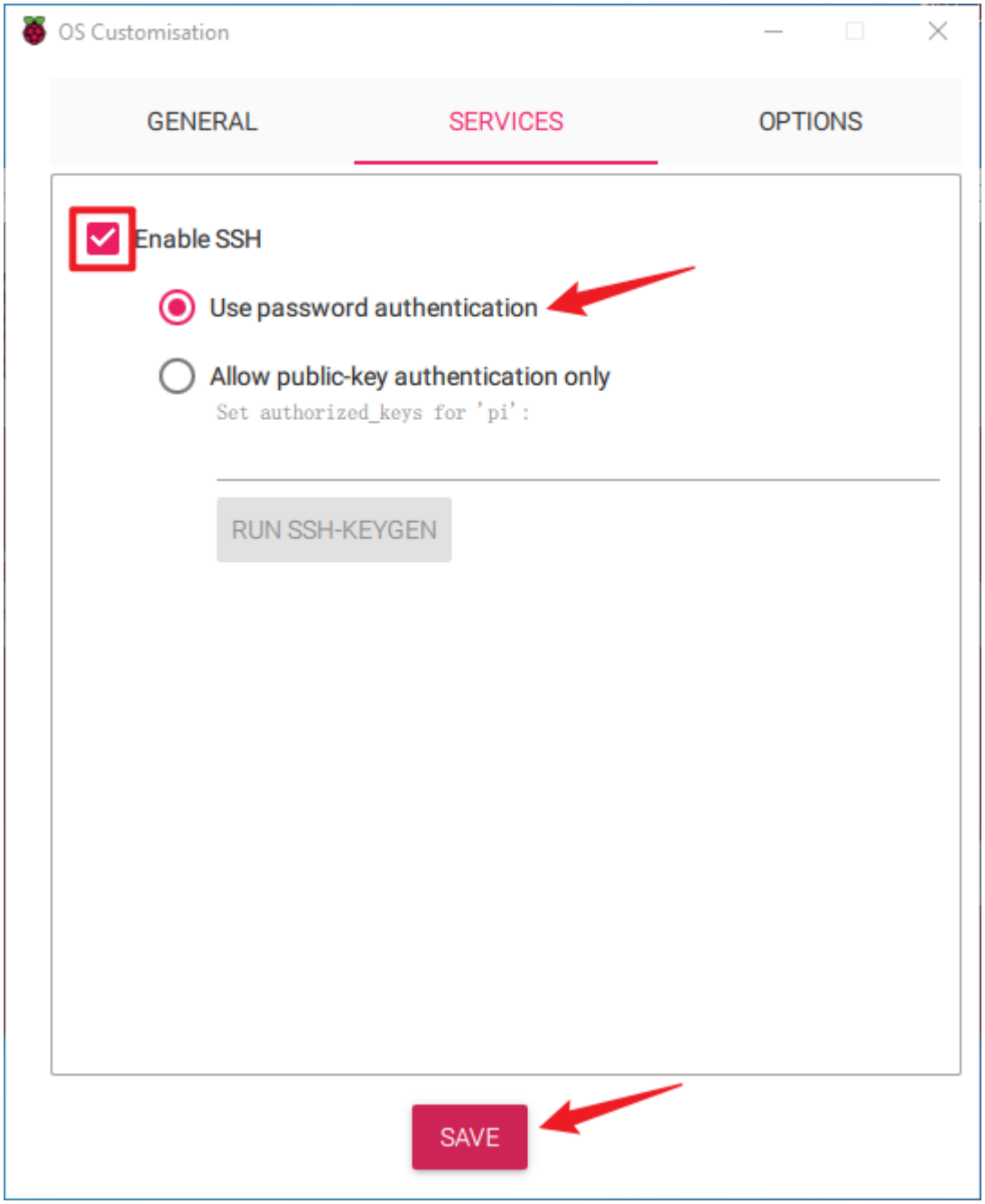


11. ワイヤレス LAN を設定するために、ネットワークの **SSID** と **Password** を入力します。

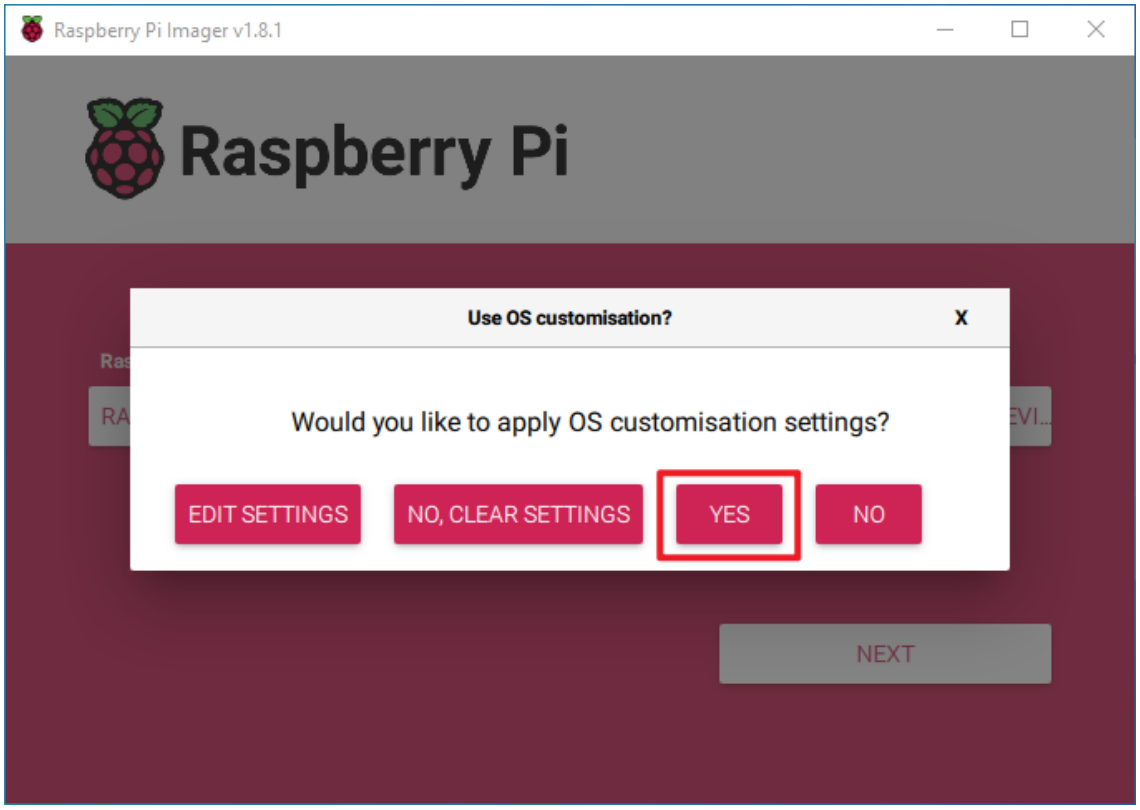
注釈: `Wireless LAN country` は、Raspberry Pi を使用している国の 2 文字の ISO/IEC alpha2 コードに設定する必要があります。



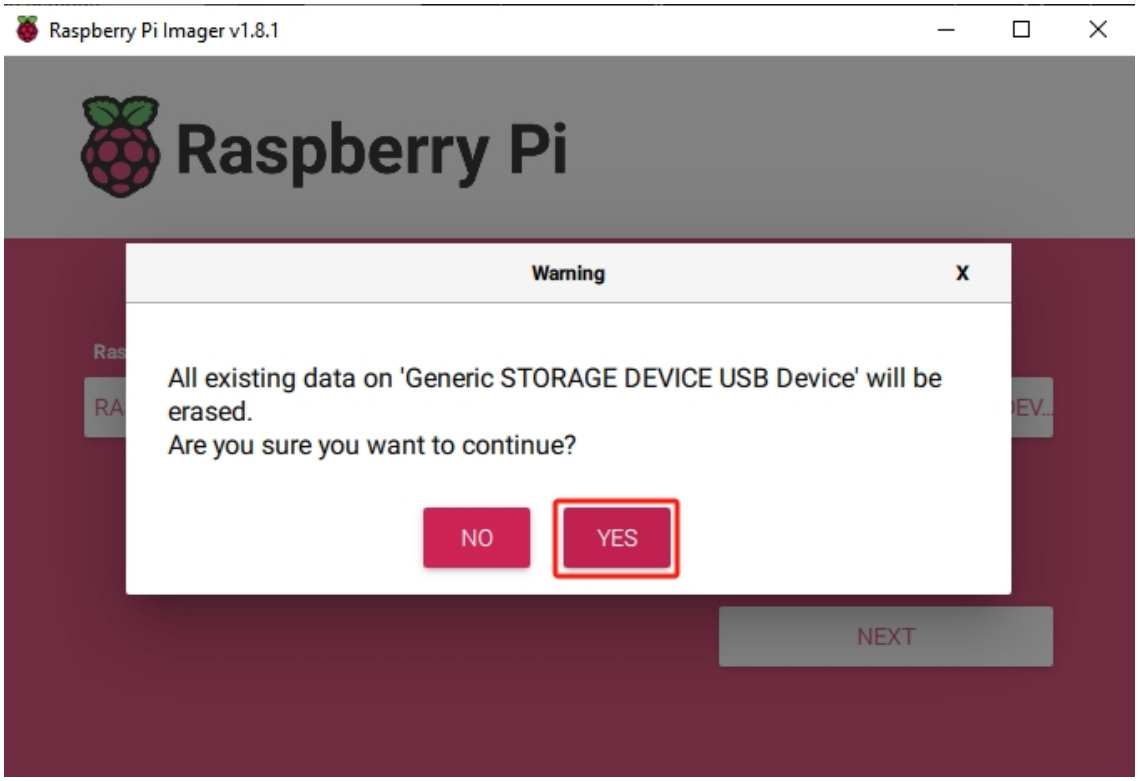
12. **SERVICES** をクリックし、パスワードベースのリモートアクセスのために **SSH** を有効にします。 **Save** をクリックすることを忘れないでください。



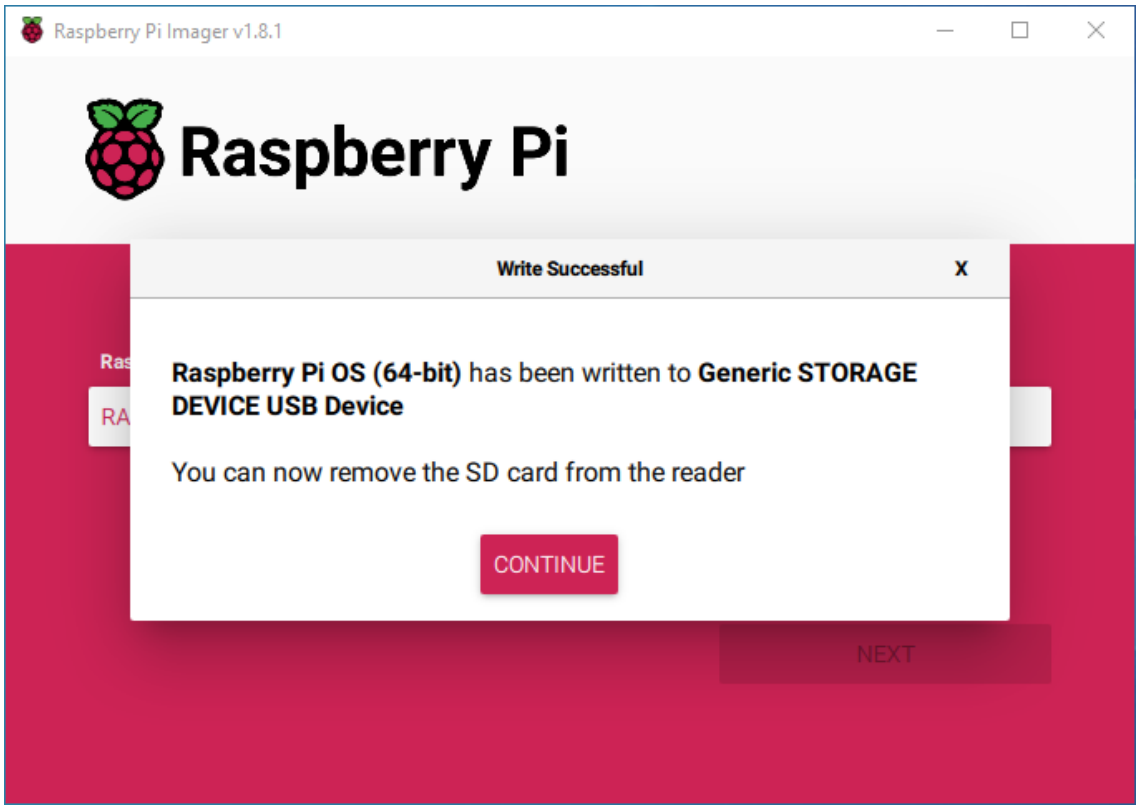
13. **Yes** をクリックして選択を確認します。



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。



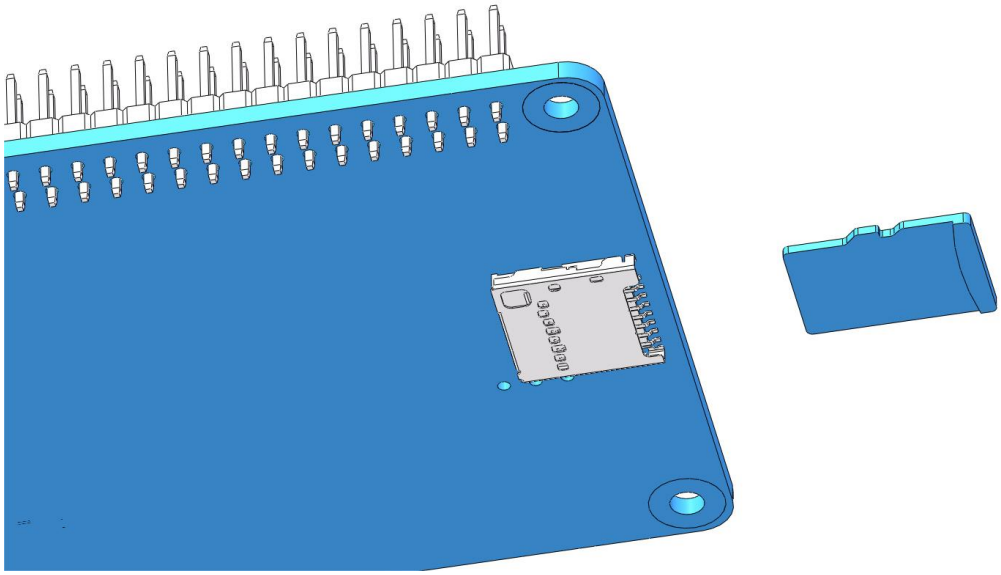
15. OS が SD カードに書き込まれるのを待ちます。完了すると、確認ウィンドウが表示されます。



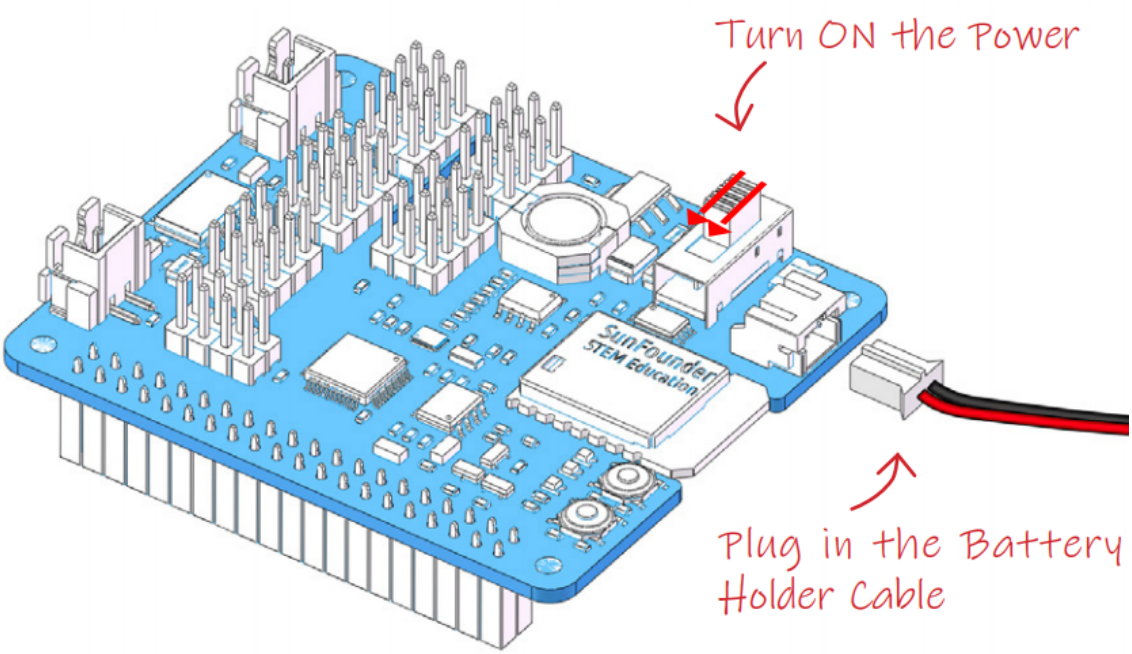
4.1.3 Raspberry Pi のセットアップ

Raspberry Pi 用の電源供給（重要）

1. Raspberry Pi OS がセットアップされた SD カードを、Raspberry Pi の下側にある microSD カードスロットに挿入します。



2. 組み立て説明に従い、バッテリーケーブルを挿入し、電源スイッチを入れます。次に、USB-C ケーブルを挿してバッテリーに電源を供給します。1〜2 分待つと、Raspberry Pi が正常に起動したことを示す音が聞こえます。



注釈: 後続のソフトウェアセットアッププロセスにかなりの時間がかかるため、USB-C ケーブルを差し込んだままにすることを推奨します。

画面をお持ちの場合

注釈: ロボットに搭載されている Raspberry Pi ZERO は画面に接続しにくいため、画面がない方法でセットアップしてください。

画面がある場合は、Raspberry Pi での操作が容易になります。

必要なコンポーネント

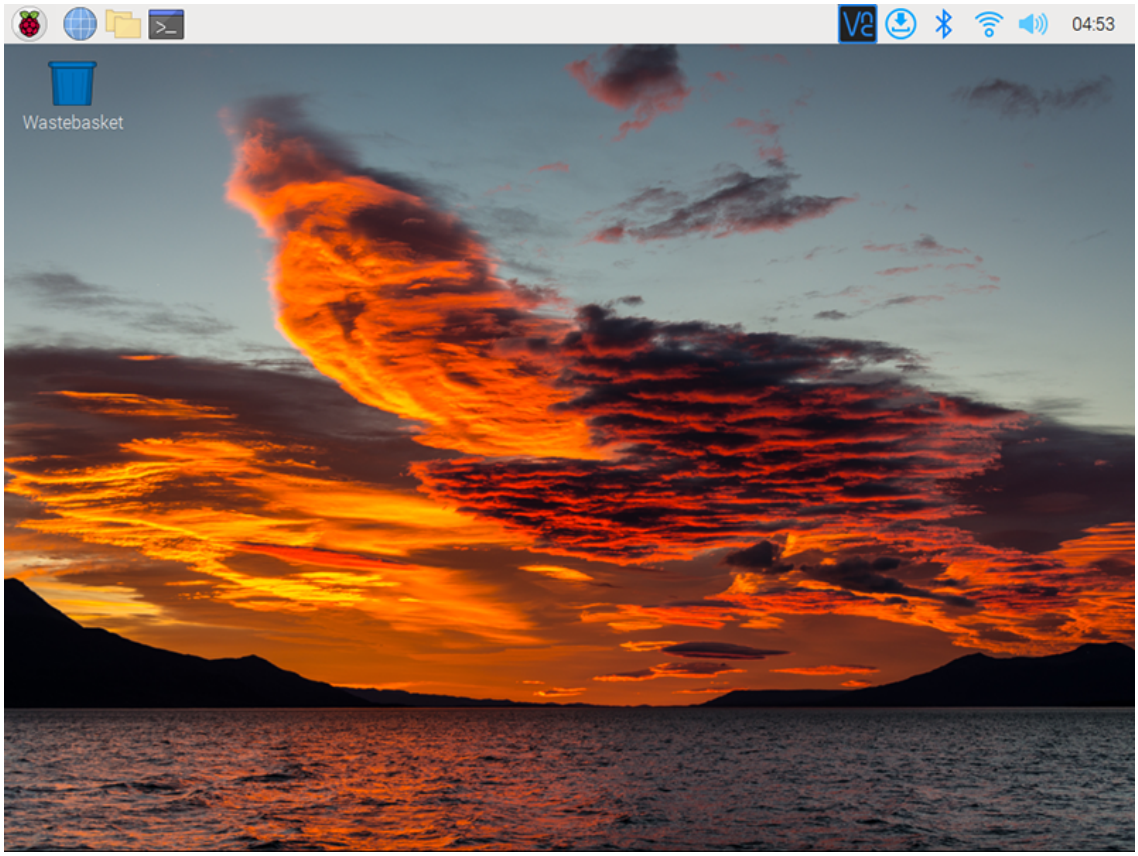
- Raspberry Pi 4B/3B 3B+/2B
- 1 * 電源アダプタ
- 1 * マイクロ SD カード
- 1 * スクリーン電源アダプタ
- 1 * HDMI ケーブル
- 1 * スクリーン
- 1 * マウス
- 1 * キーボード

1. マウスとキーボードを接続します。
2. スクリーンを Raspberry Pi の HDMI ポートに接続し、スクリーンがコンセントに差し込まれ、オンになっていることを確認します。

   注釈: Raspberry Pi 4 を使用する場合は、電源ポートに最も近い HDMI0 にスクリーンを接続する必要があります。

   ---

3. 電源アダプタを使用して 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。

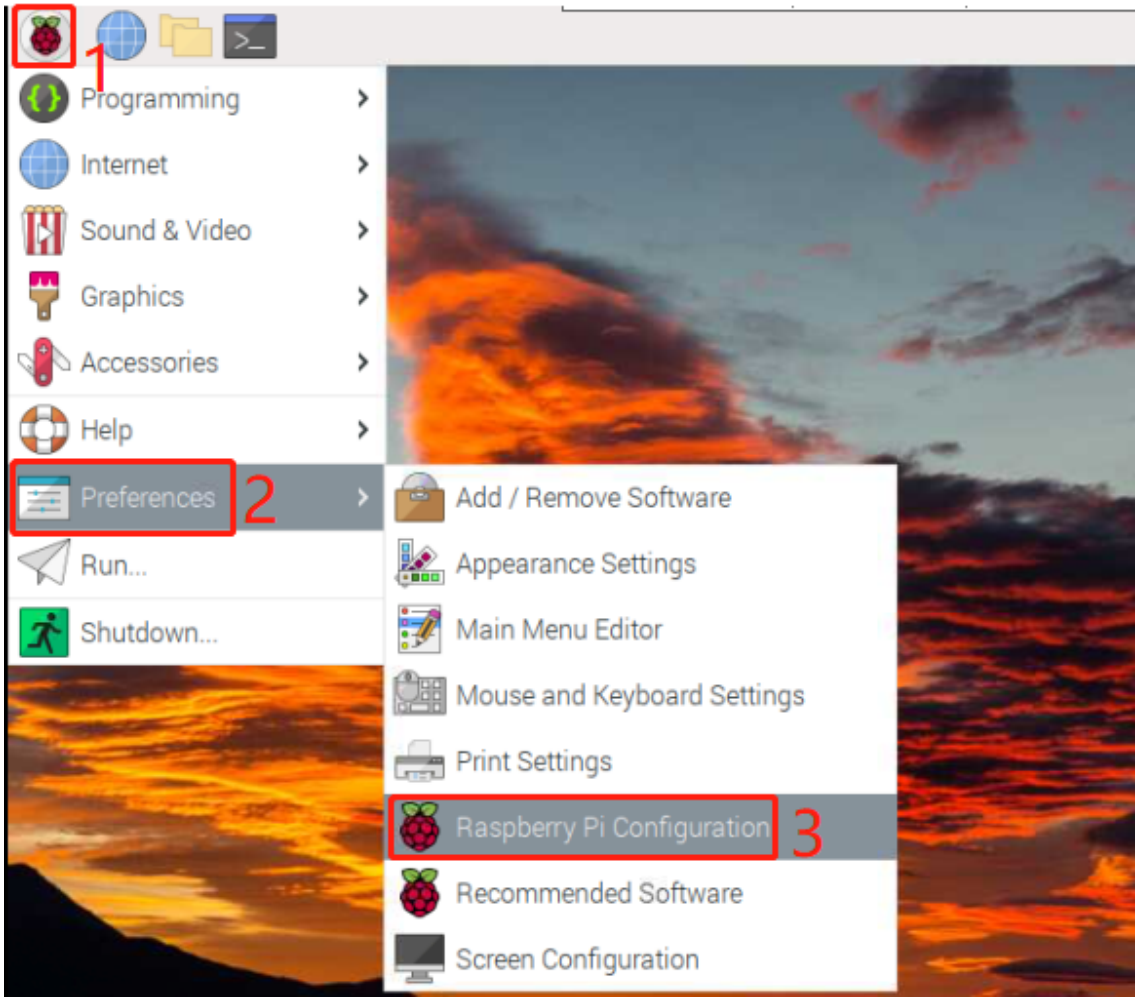
この方法では、Mac と Raspberry Pi 間の通信は暗号化されません。通信は自宅やビジネスネットワーク内で行われるため、保護されていなくても問題ありません。ただし、気になる場合は、 VNC® Viewer などの VNC アプリケーションをインストールすることができます。

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。

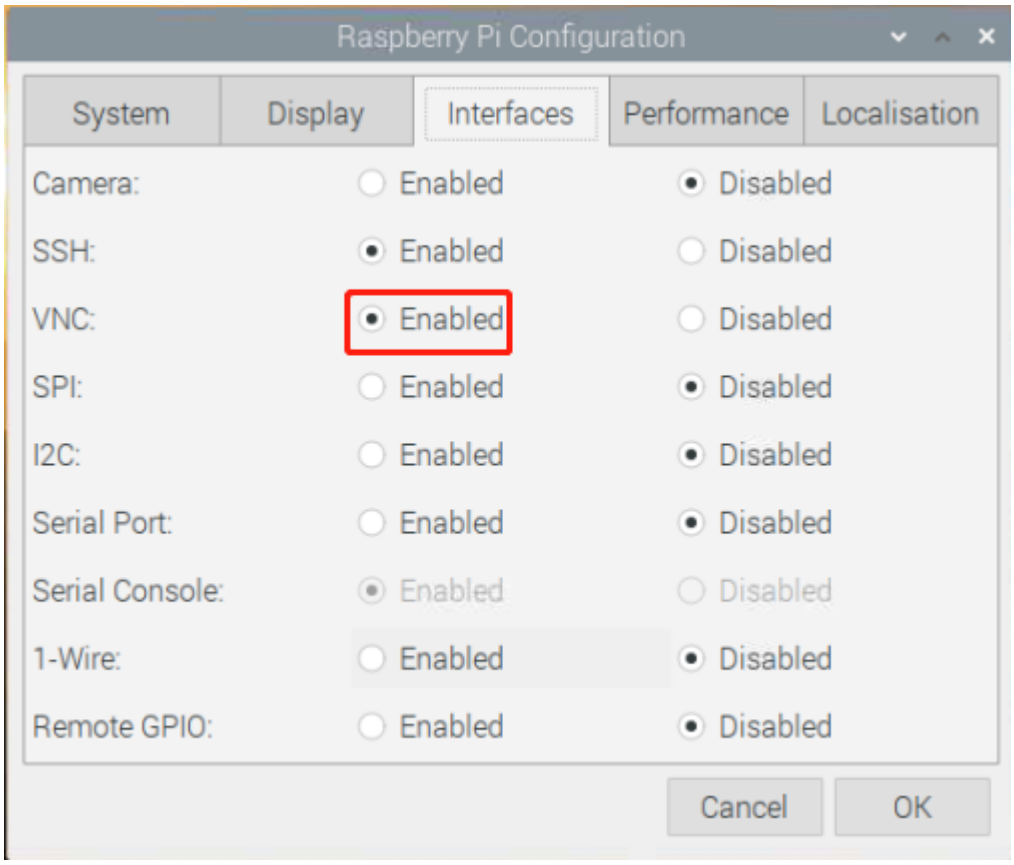
- **一時的にモニター（またはテレビ）を使用しますか？**
- **一時的なモニター（またはテレビ）がない場合**

一時的にモニター（またはテレビ）を使用しますか？

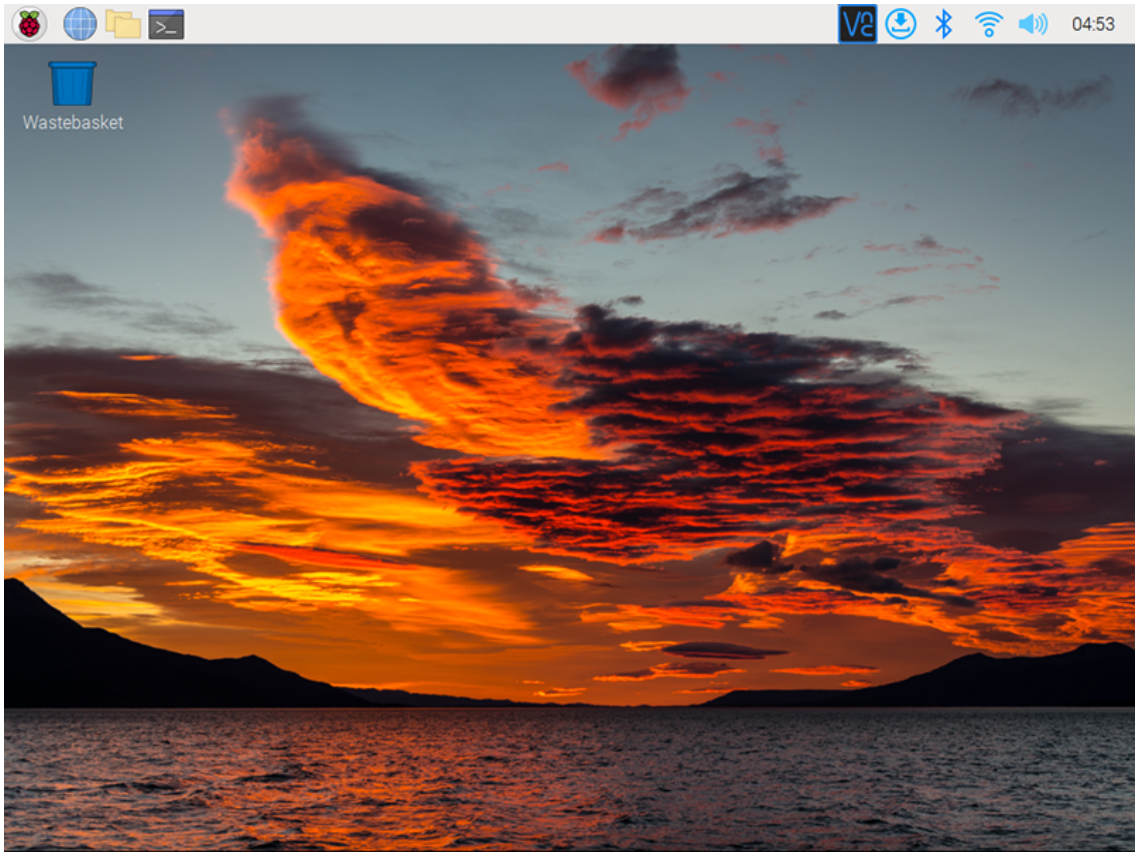
1. モニター（またはテレビ）、マウス、キーボードを Raspberry Pi に接続し、電源を入れます。図の数字に従ってメニューを選択します。



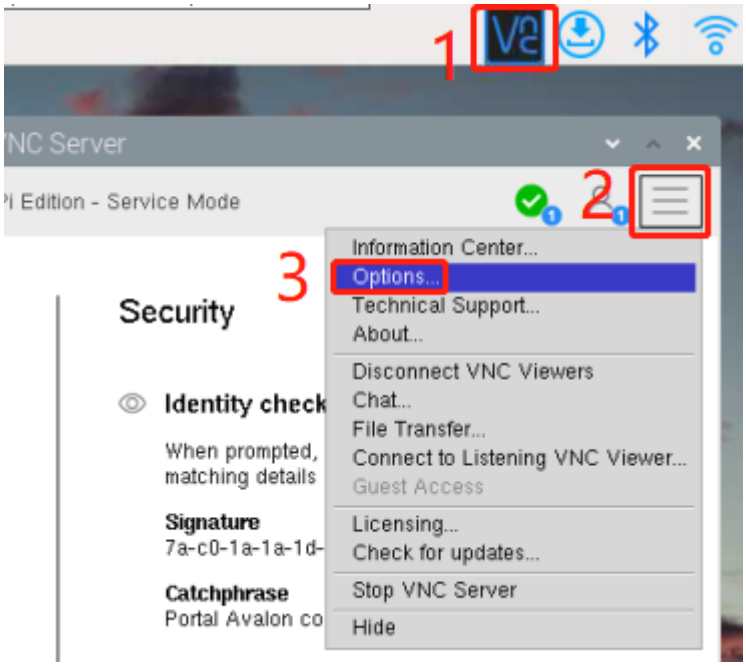
2. 次の画面が表示されます。 **Interfaces** タブで **VNC** を **Enabled** に設定し、 **OK** をクリックします。



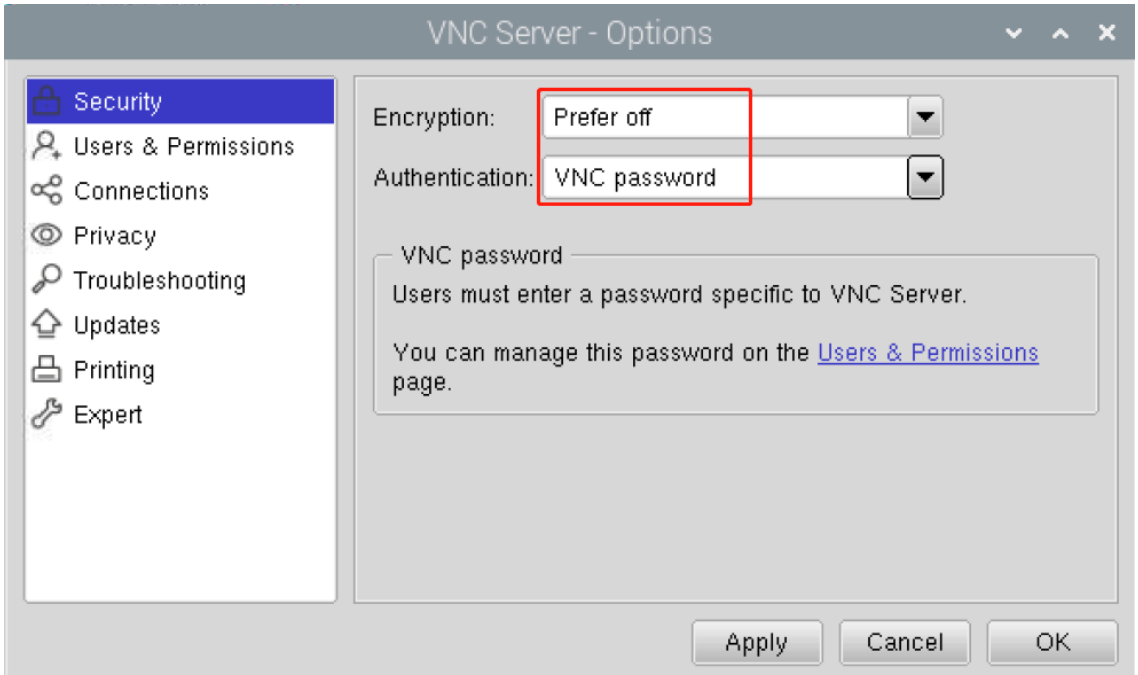
3. 画面の右上に VNC アイコンが表示され、VNC サーバーが起動します。



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。

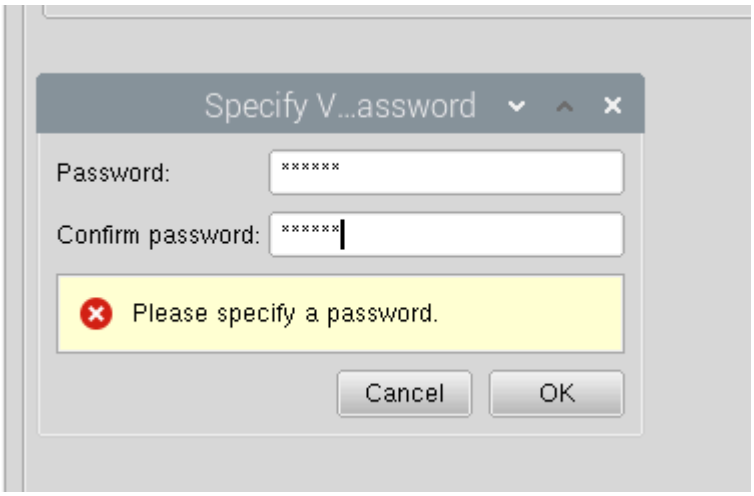


5. 次の画面が表示され、オプションを変更できます。



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

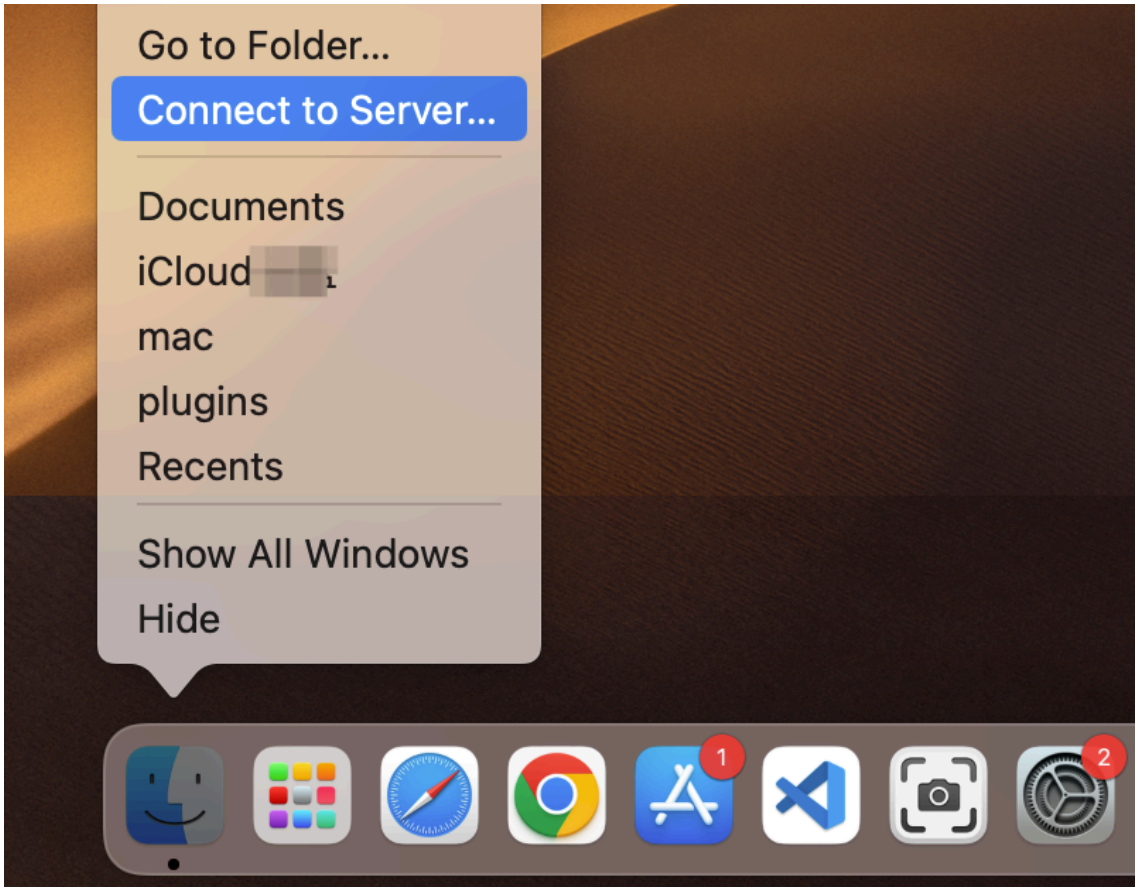
6. **OK** ボタンをクリックすると、パスワード入力画面が表示されます。Raspberry pi のパスワードと同じものを使用することも、異なるパスワードを設定することもできるので、入力して **OK** をクリックします。



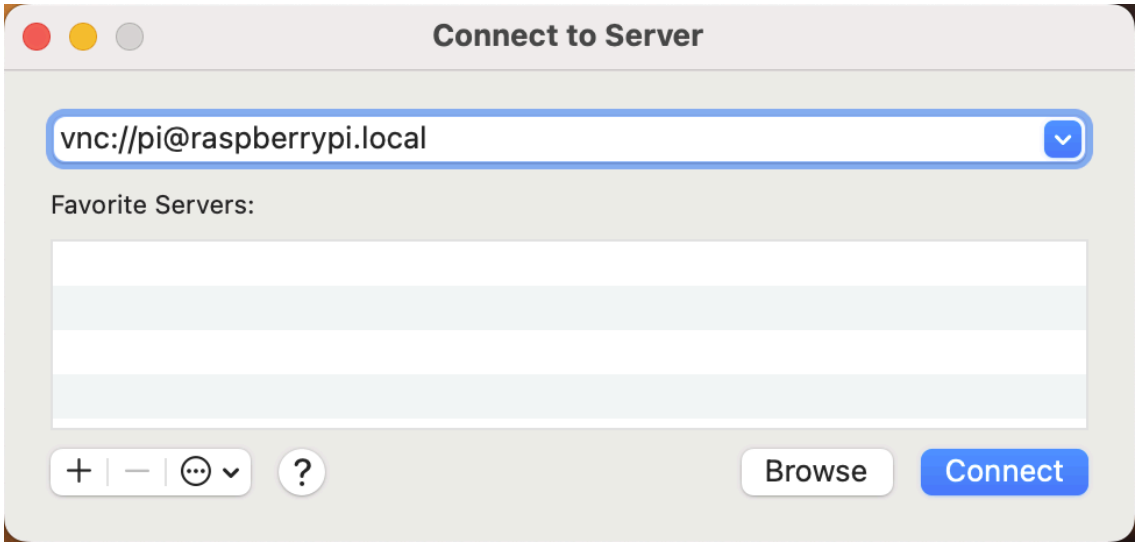
これで Mac からの接続が準備完了です。モニターを切断しても構いません。

ここからは Mac 側の操作になります。

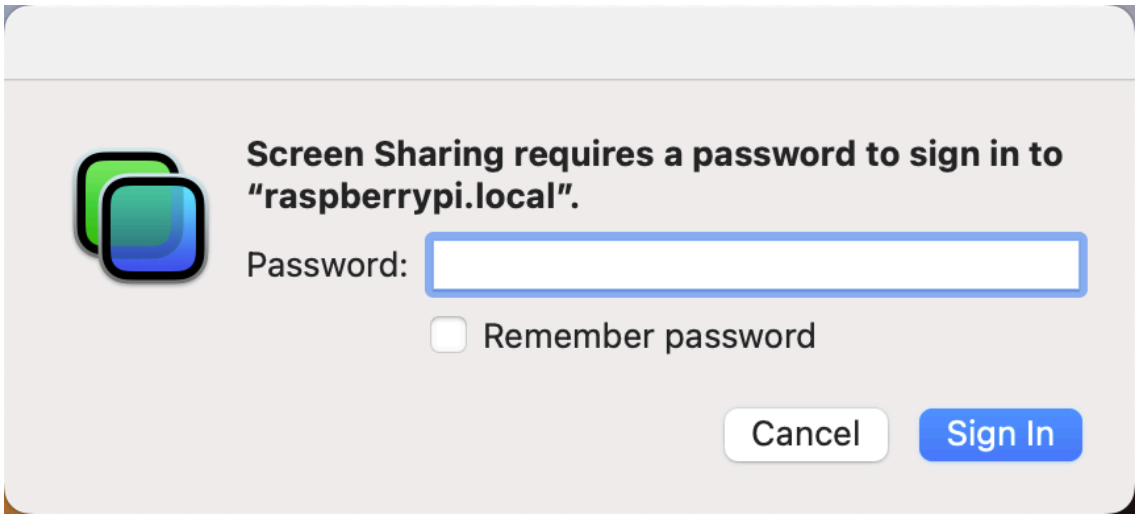
1. Finder のメニューから **Connect to Server** を選択します。右クリックで開くことができます。



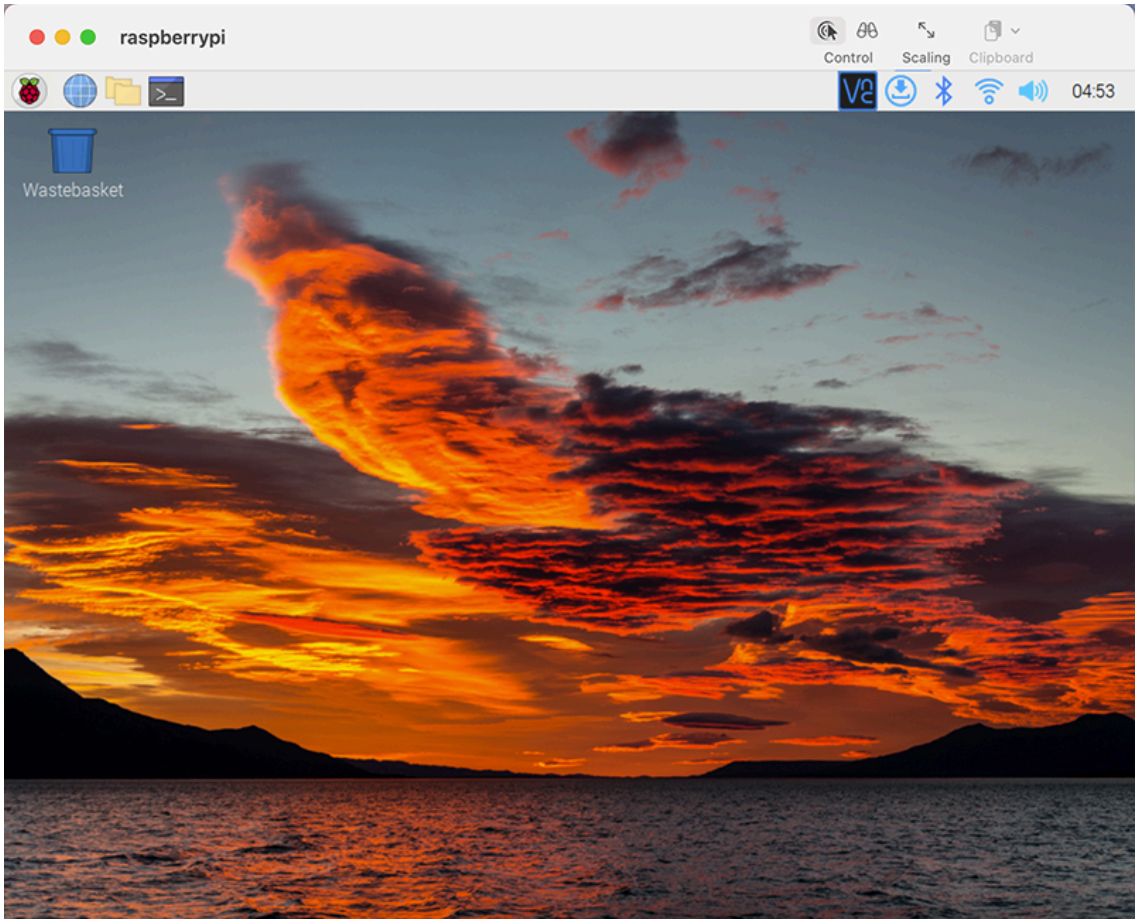
2. vnc://<username>@<hostname>.local``（または ``vnc://<username>@<IP address>）を入力します。入力した後、 **Connect** をクリックします。



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。

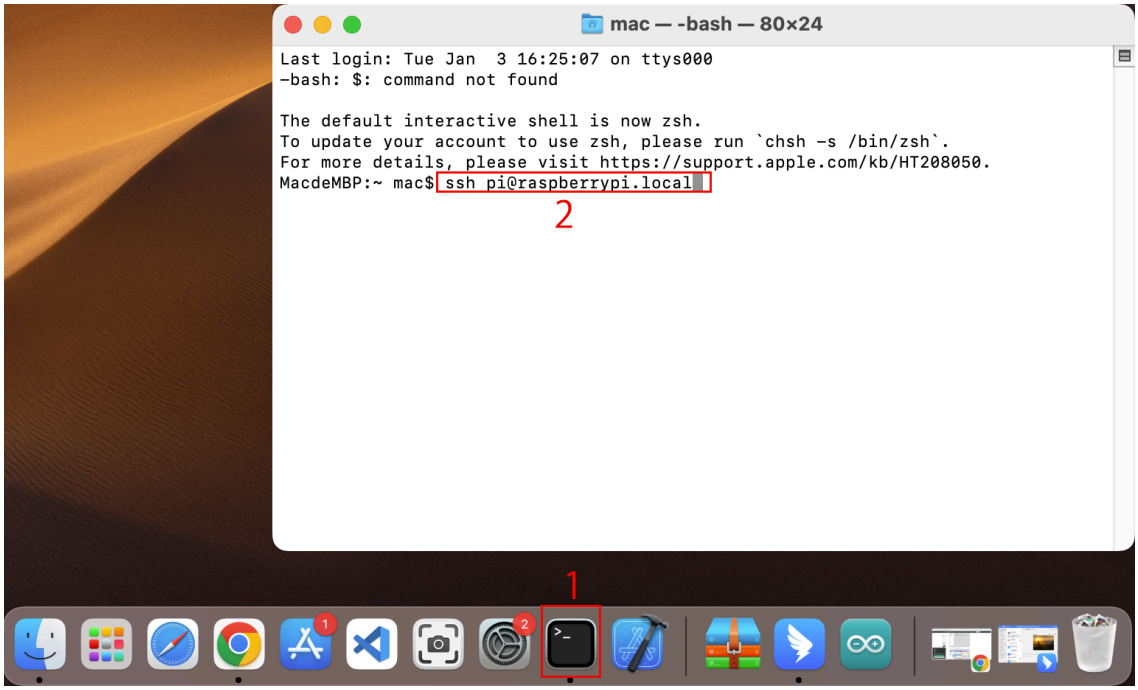


一時的なモニター（またはテレビ）がない場合

- 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
```

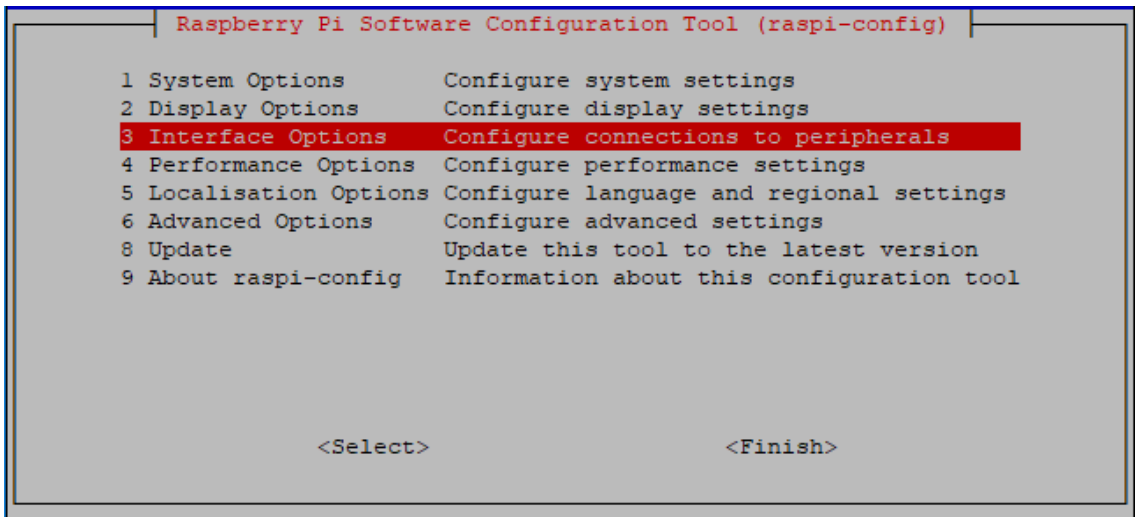
`Do you want to continue? [Y/n]` と表示されたら、 `Y` と入力してください。

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。)

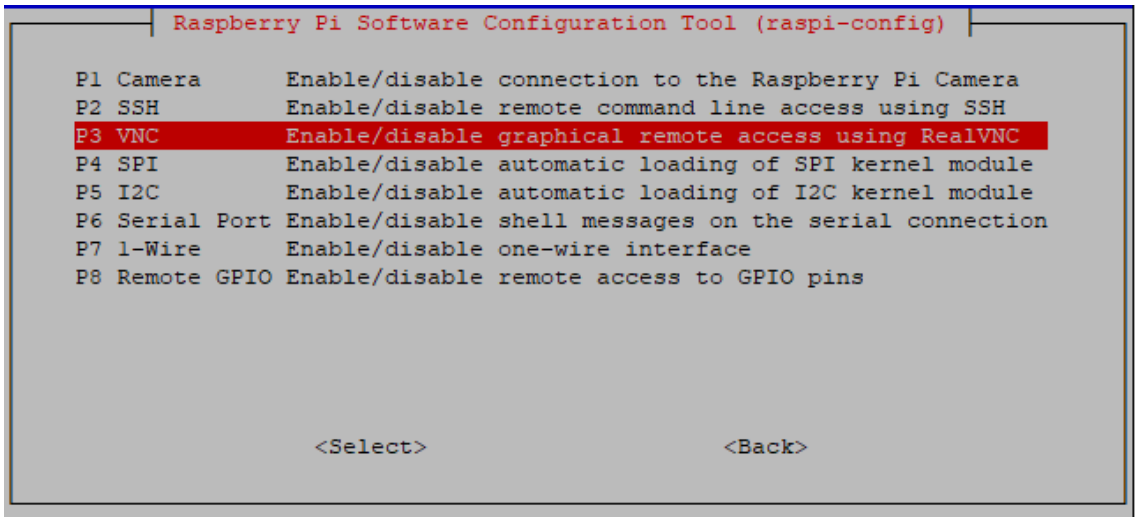
5. **VNC Server** を有効にするために、以下のコマンドを入力します。

```
sudo raspi-config
```

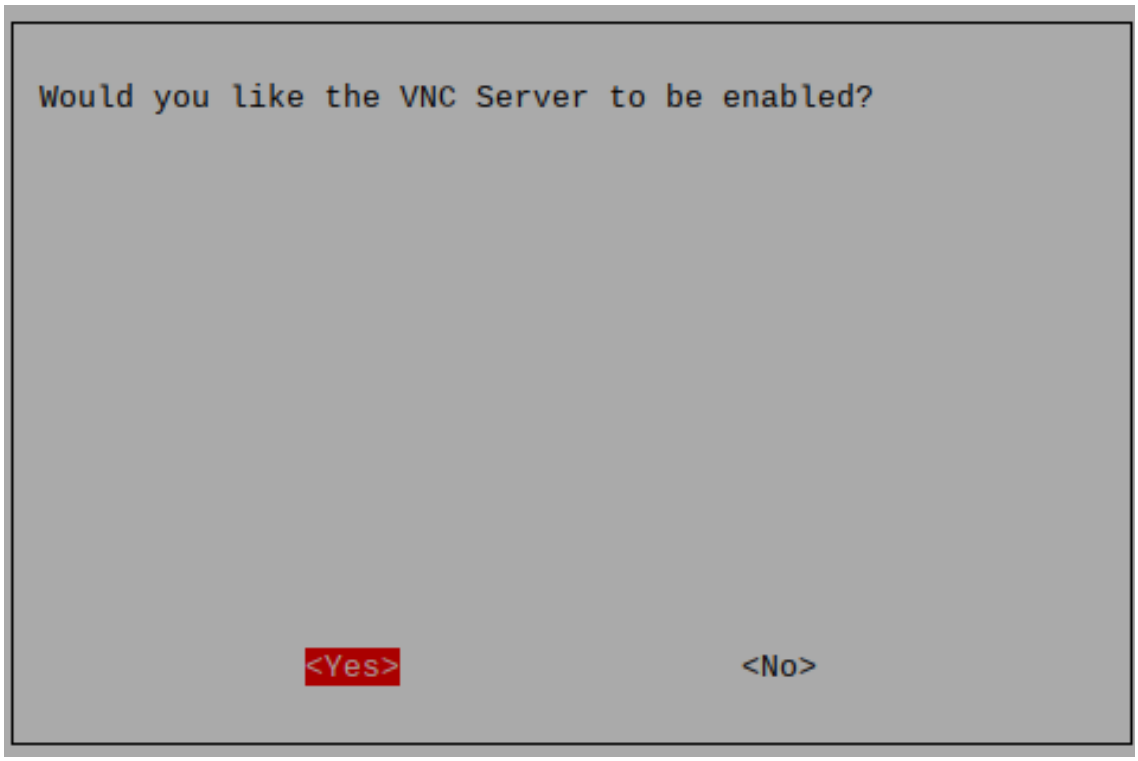
6. 次の画面が表示されます。キーボードの矢印キーを使って **Interface Options** を選択し、 **Enter** キーを押します。



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。

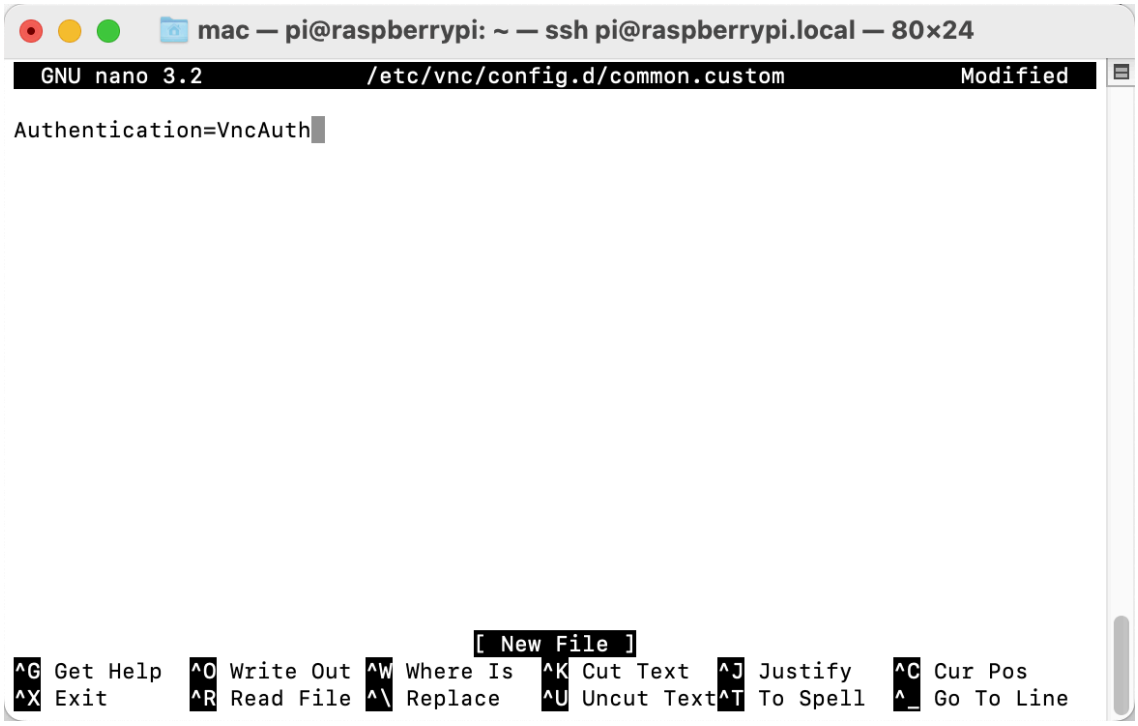


9. VNC サーバーが起動したので、Mac から接続するための設定を変更しましょう。

   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。



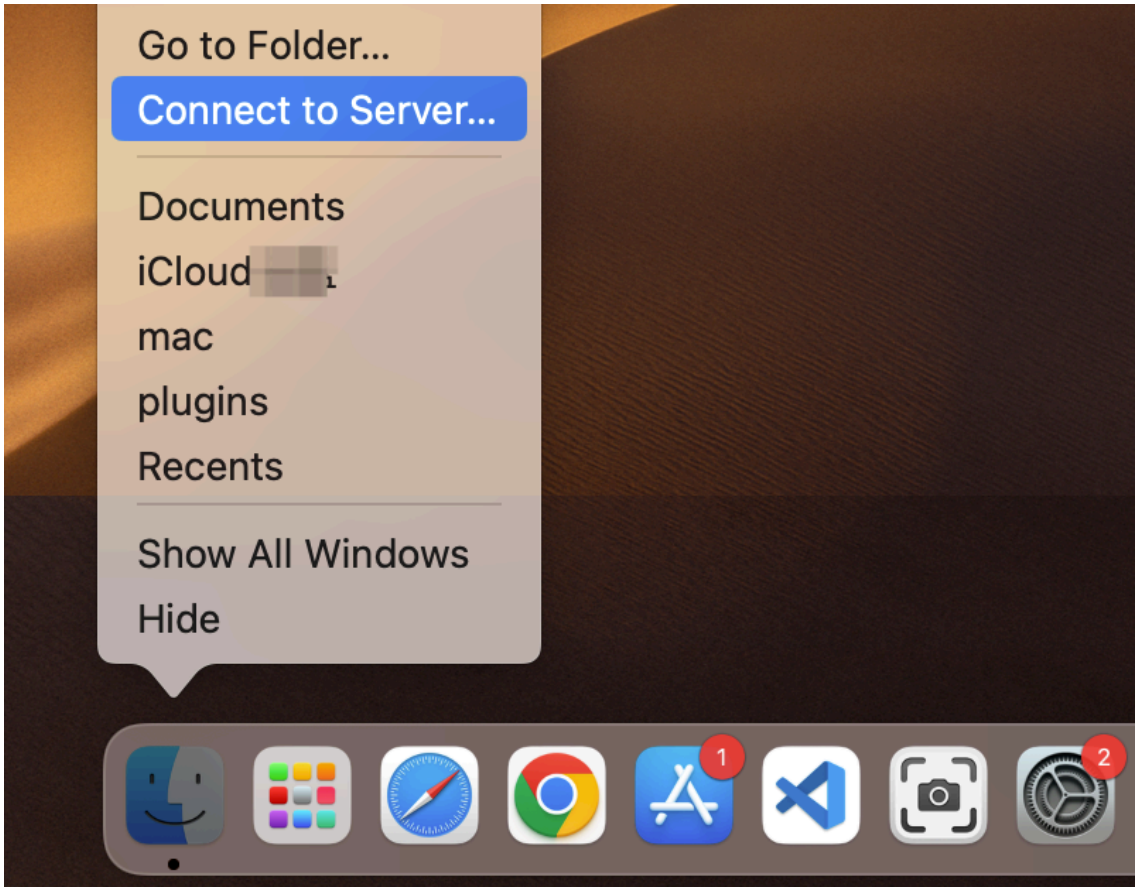
10. さらに、Mac から VNC 経由でログインするためのパスワードを設定します。Raspberry Pi のパスワードと同じものを使用することも、異なるパスワードを使用することもできます。

```
sudo vncpasswd -service
```

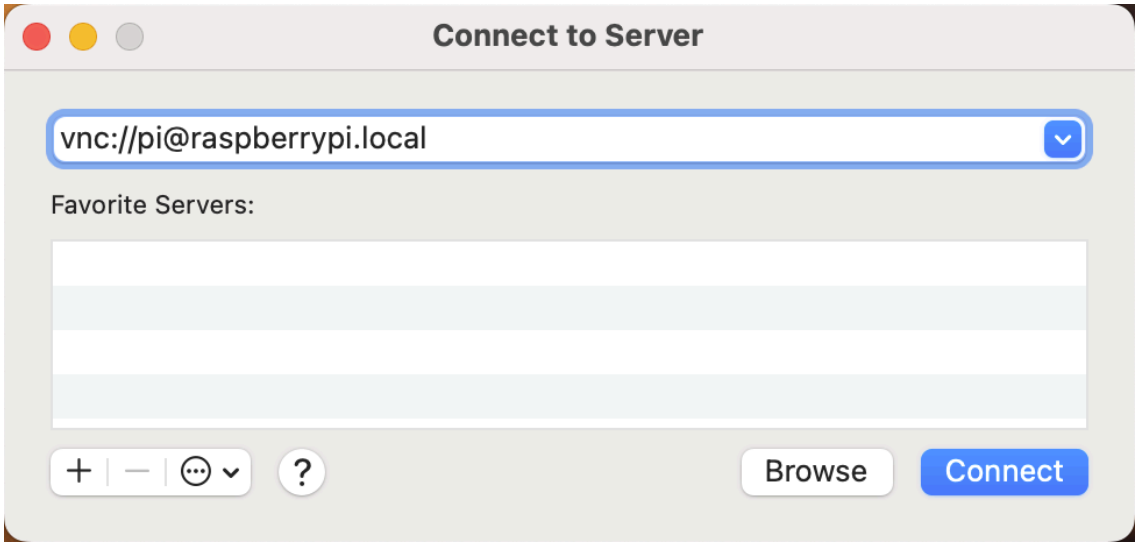
11. 設定が完了したら、Raspberry Pi を再起動して変更を適用します。

```
sudo sudo reboot
```

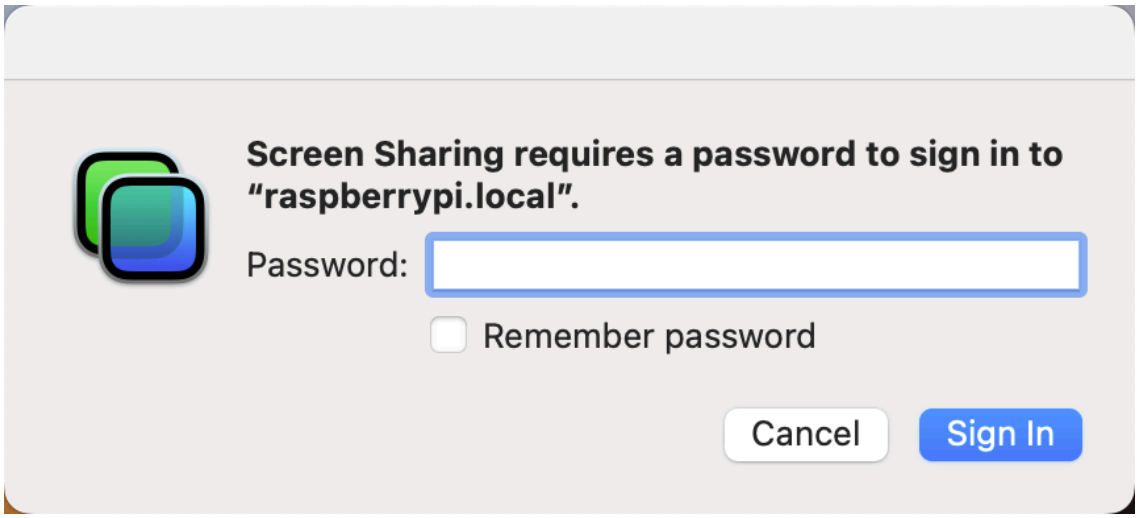
12. これで、右クリックで開くことができる **Finder** のメニューから **Connect to Server** を選択します。



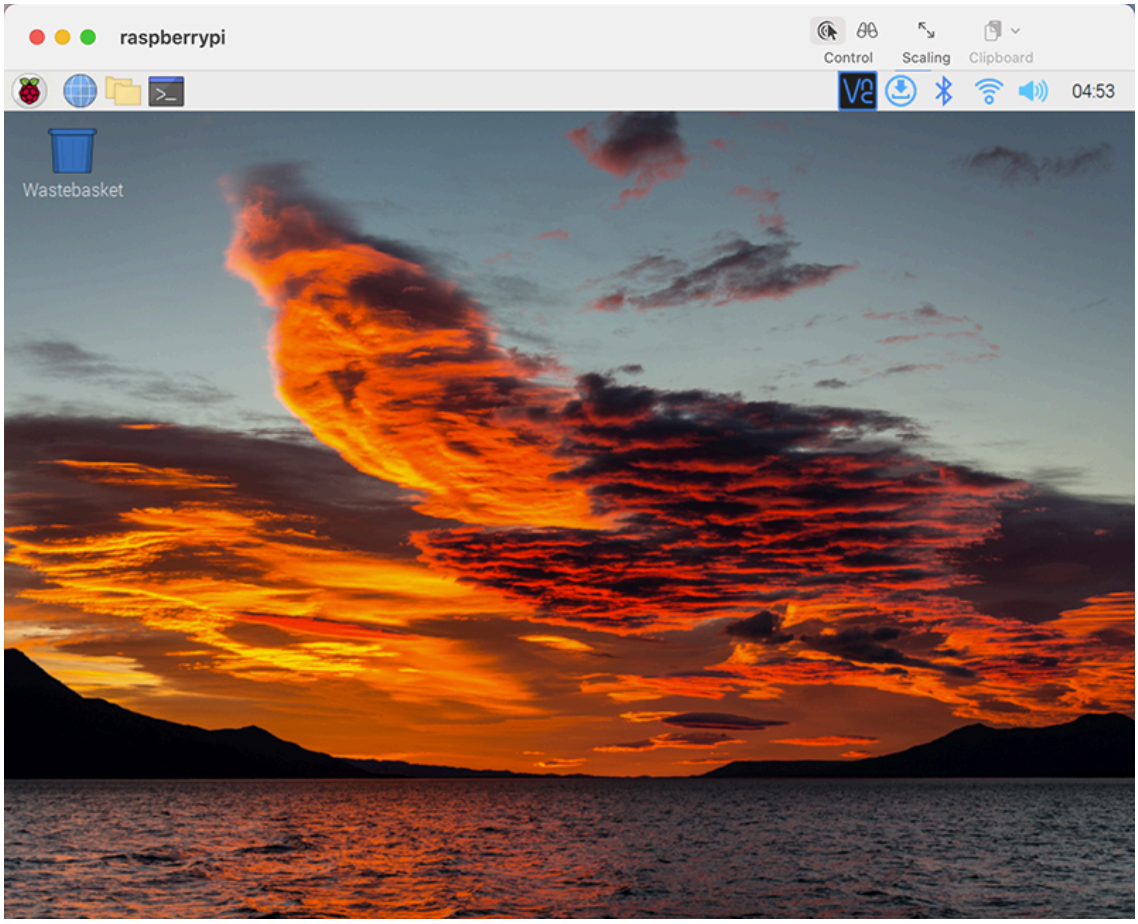
13. `vnc://<username>@<hostname>.local`` （または ``vnc://<username>@<IP address>）を入力します。入力した後、 **Connect** をクリックします。



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。

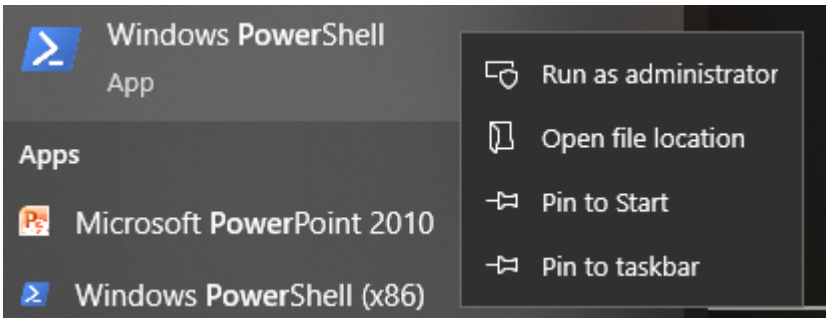


Windows ユーザー

リモートで Raspberry Pi にログイン

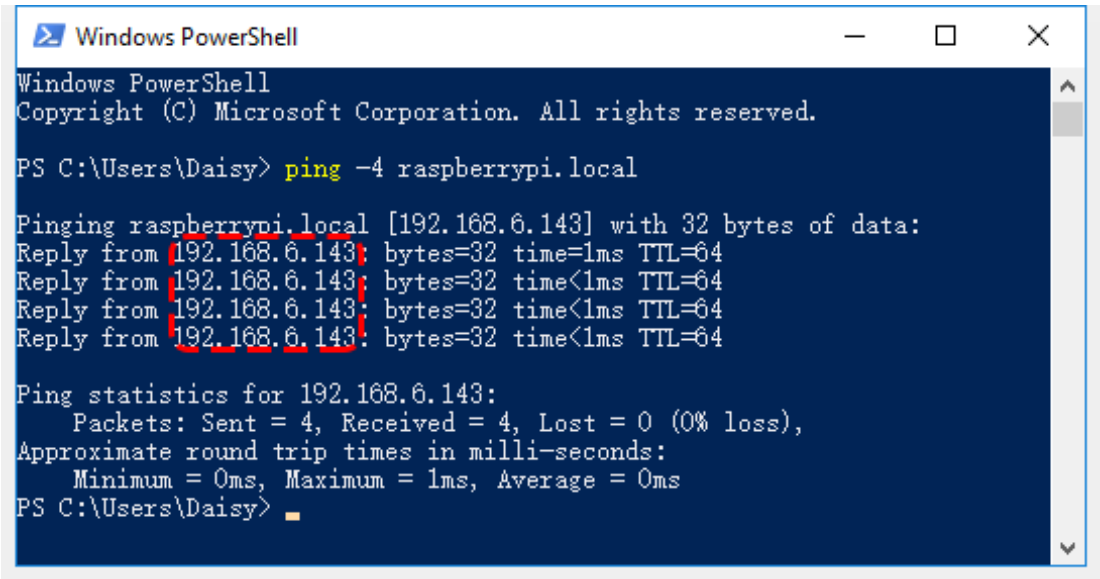
win10 を使用している場合、以下の方法でリモートで Raspberry Pi にログインできます。

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。

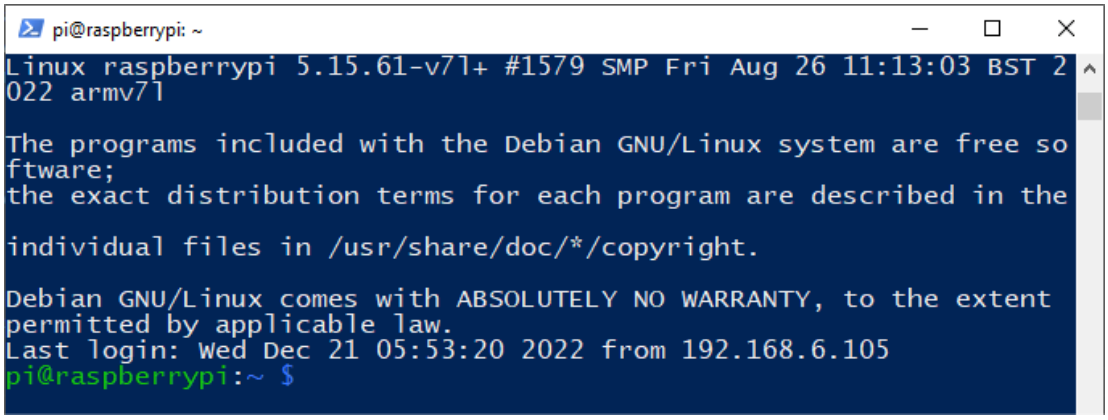
4. 最初にログインする際にのみ、以下のメッセージが表示されるので、 yes と入力します。

```
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。

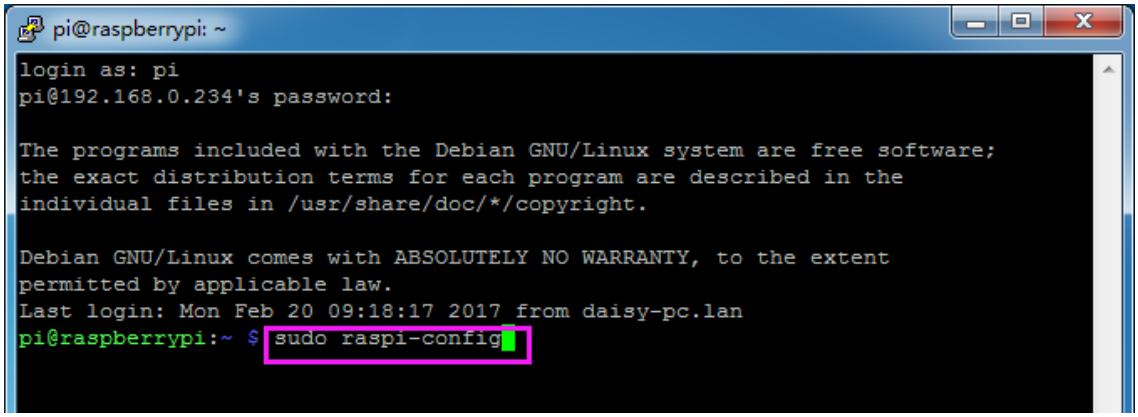
ここでは VNC® Viewer を使用します。

VNC サービスの有効化

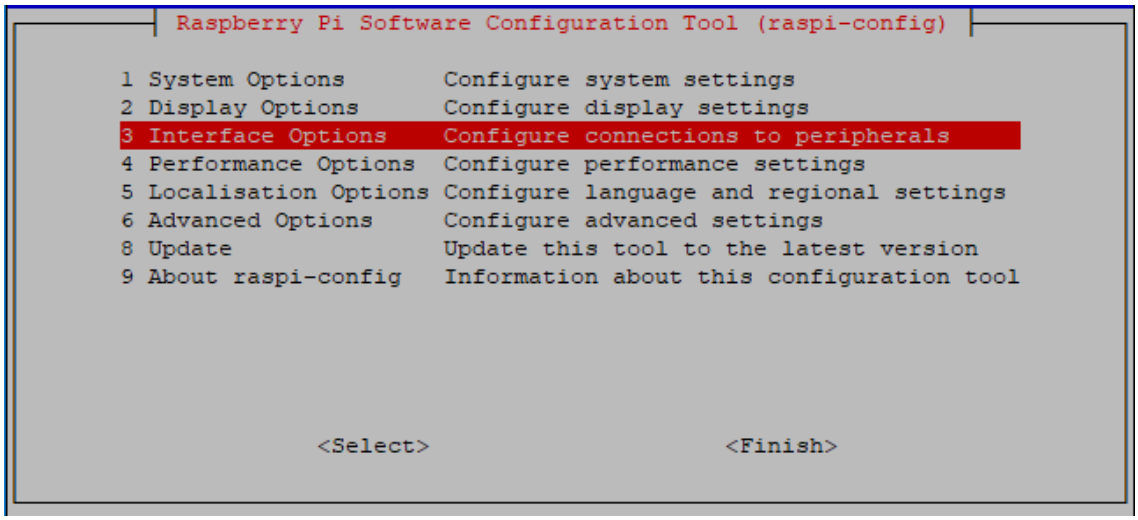
VNC サービスはシステムにインストールされています。デフォルトでは、VNC は無効になっています。config で有効にする必要があります。

1. 次のコマンドを入力します：

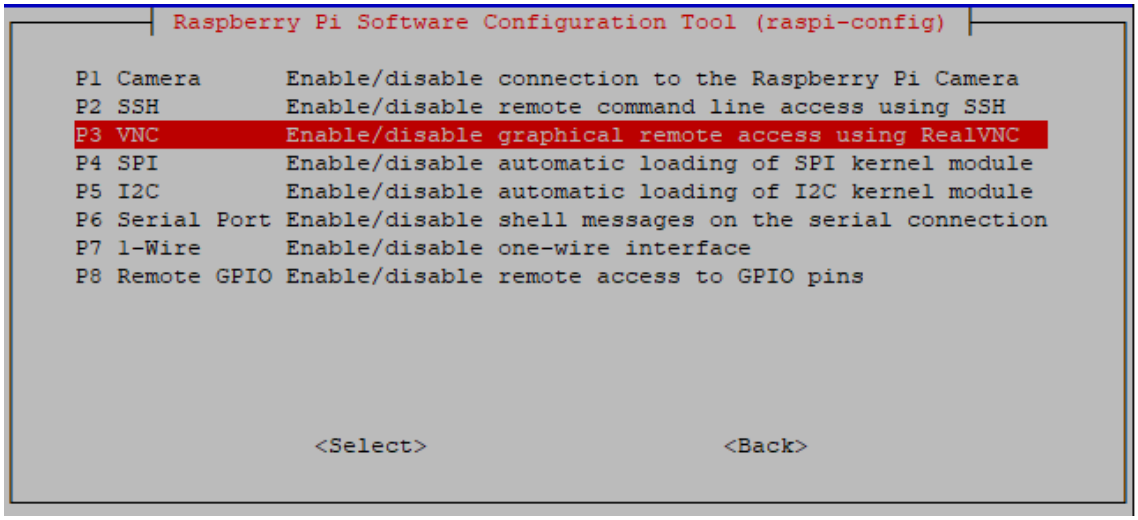
```
sudo raspi-config
```



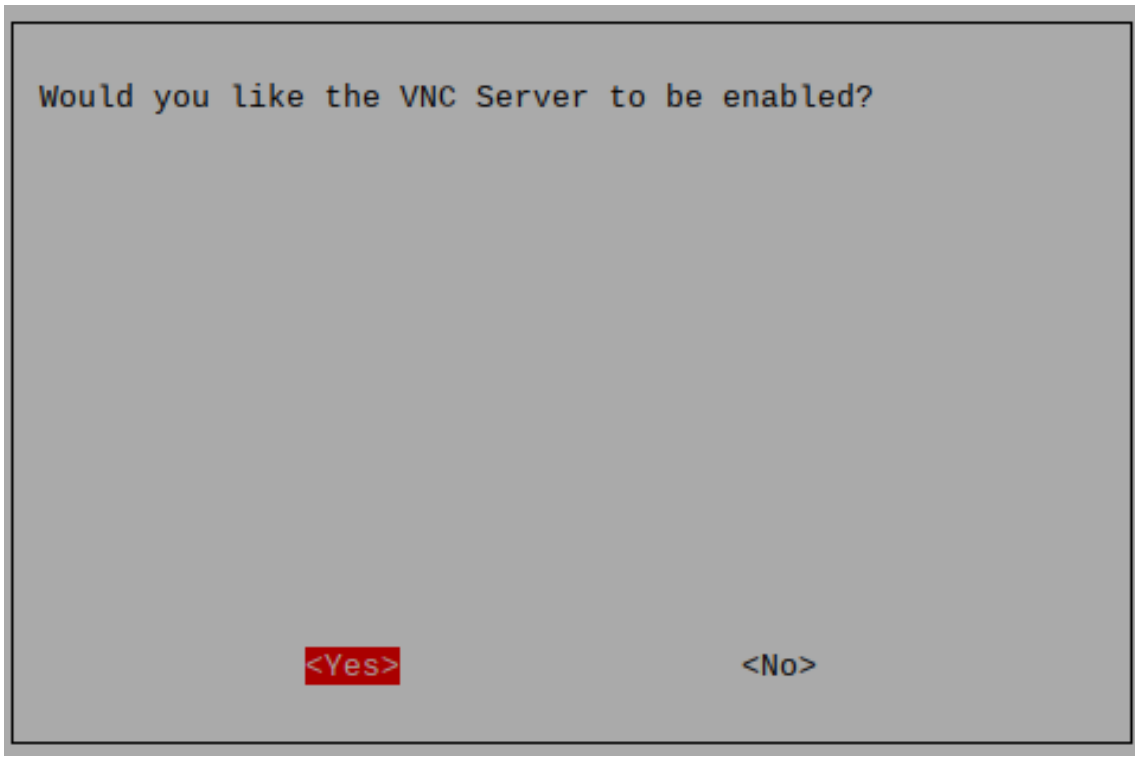
2. キーボードの下矢印キーを押して **3 Interfacing Options** を選択し、 **Enter** キーを押します。



3. 次に **VNC** を選択します。

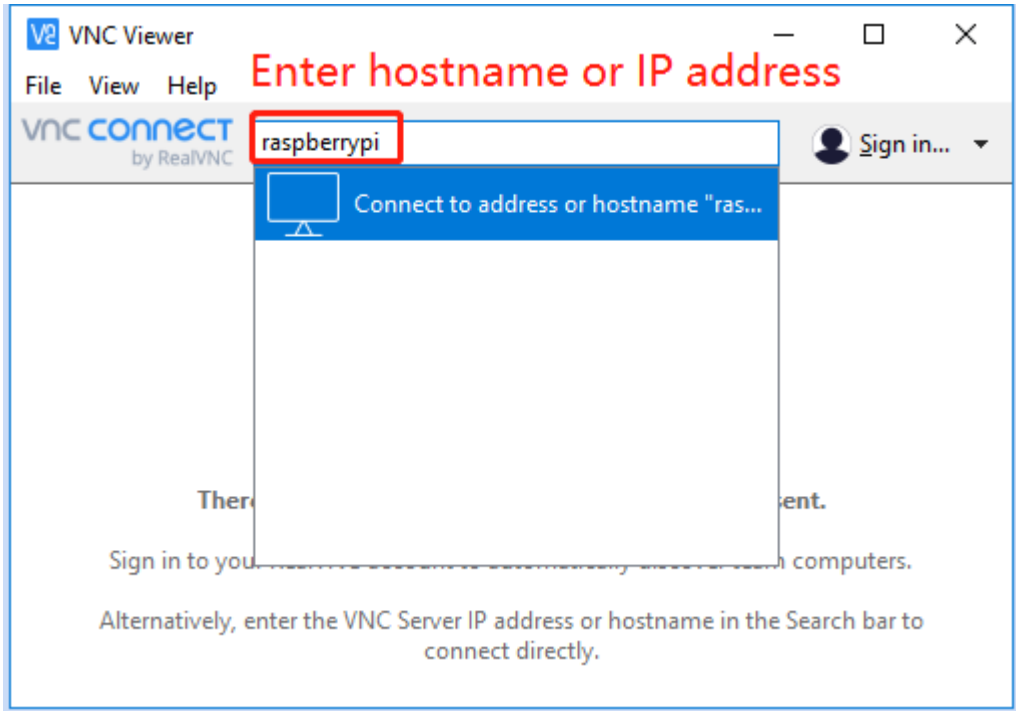


4. キーボードの矢印キーで **<Yes>** -> **<OK>** -> **<Finish>** を選択して、設定を完了します。

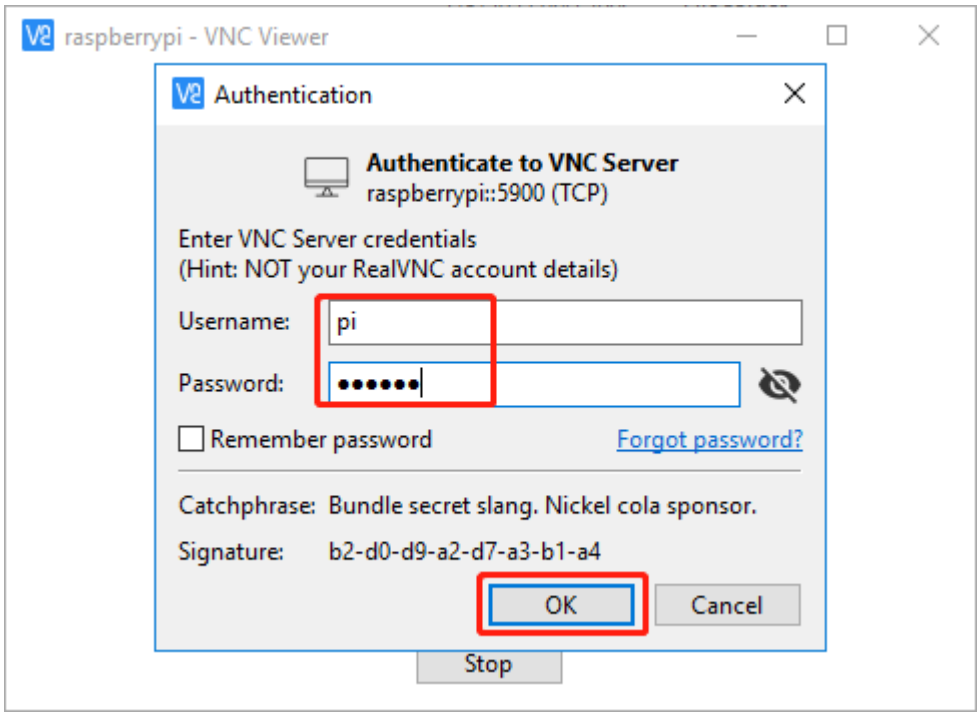


VNC でログイン

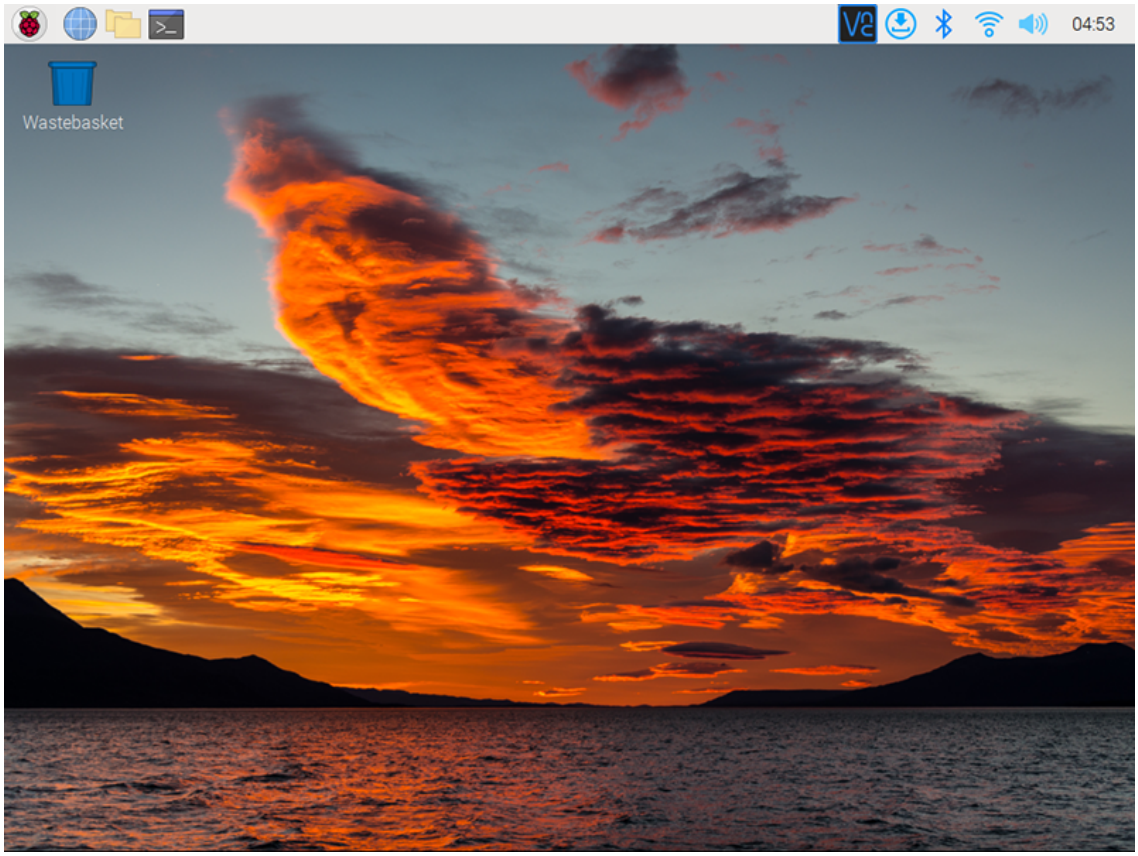
1. 個人用コンピューターに VNC Viewer をダウンロードしてインストールします。
2. インストールが完了したら、開いてホスト名または IP アドレスを入力して Enter キーを押します。



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。

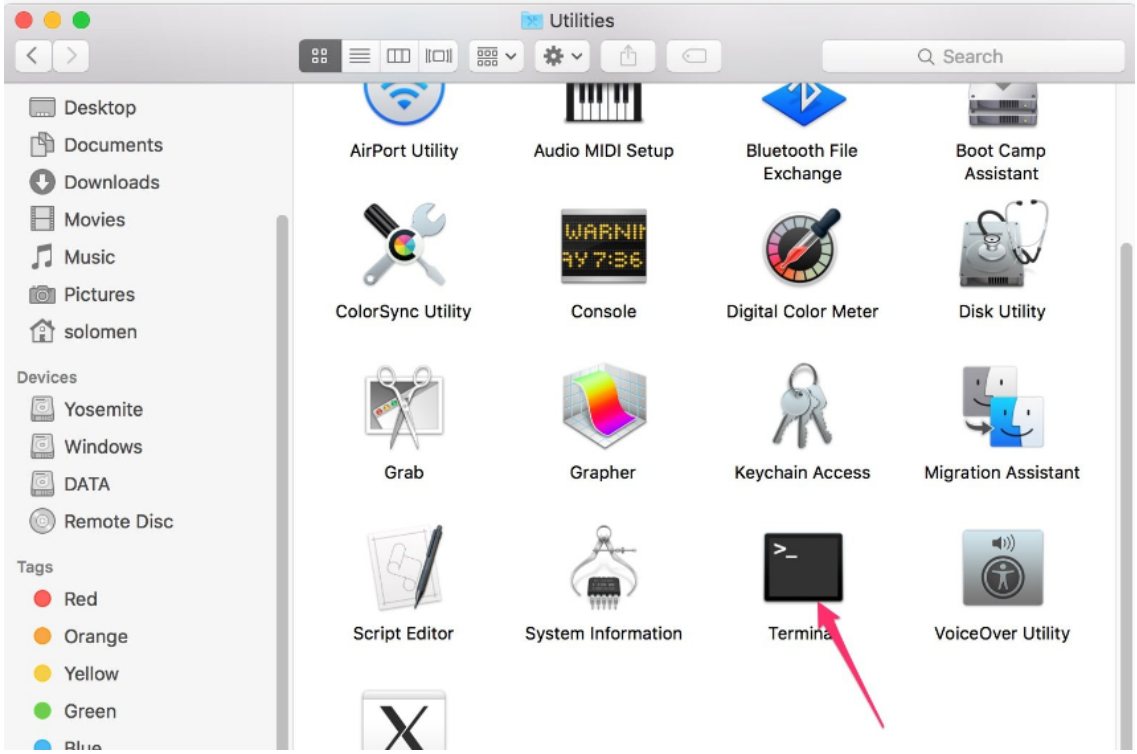


4. これで Raspberry Pi のデスクトップが表示されます。



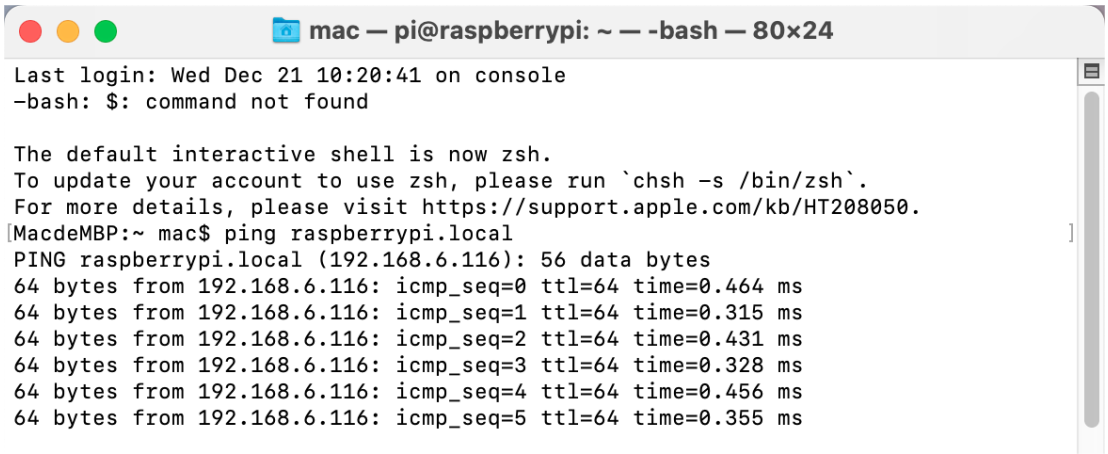
Linux /Unix ユーザー

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。

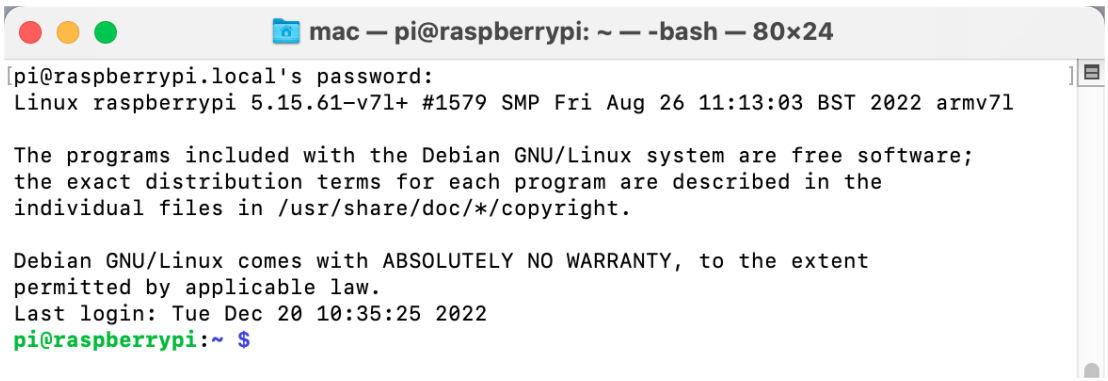
```
The authenticity of host 'raspberrypi.local␣
↪(2400:2410:2101:5800:635b:f0b6:2662:8cba)' can't be established.
ED25519 key fingerprint is SHA256:oo7x3ZSgAo032wD1tE8eW0fFM/kmewIvRwkBys6XRwg.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])?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

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。

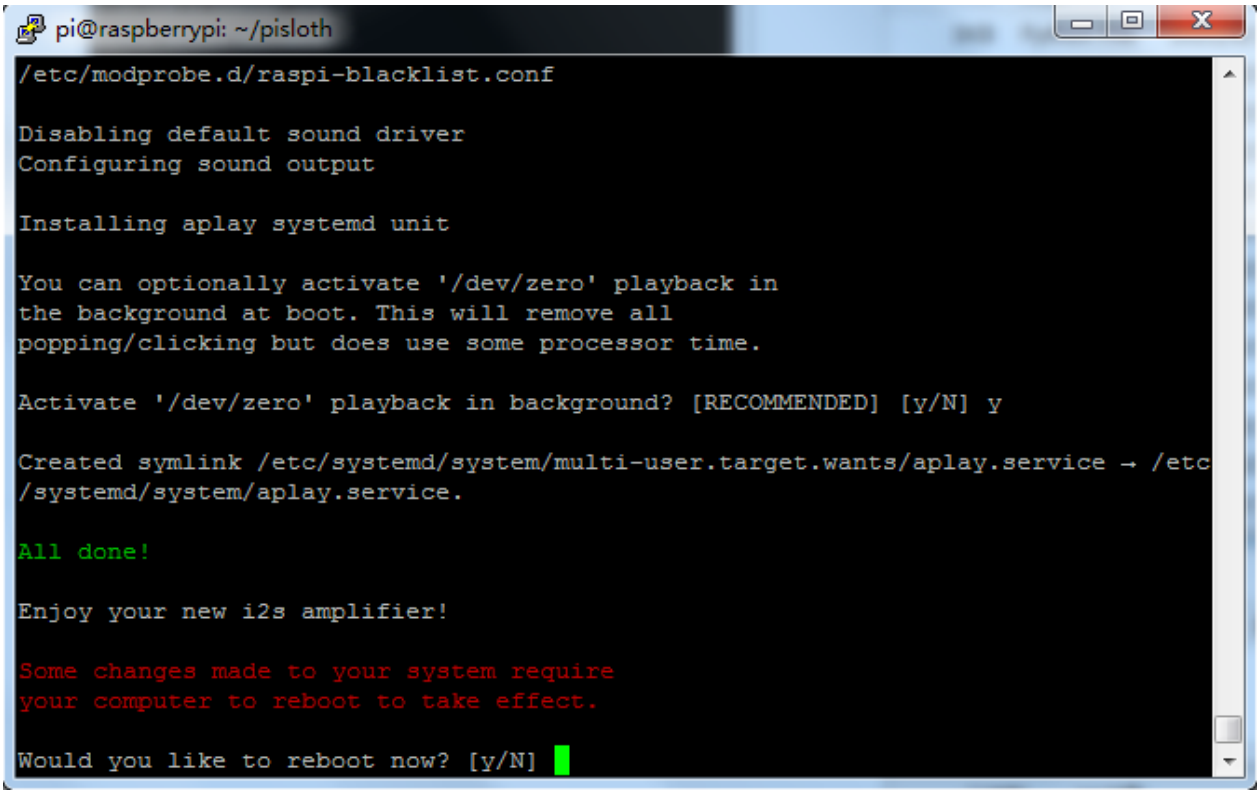
```
cd ~/picar-x
sudo bash i2samp.sh
```

```
pi@raspberrypi: ~/pisloth
  File "/usr/local/lib/python3.7/dist-packages/robot_hat-1.0.0-py3.7.egg/robot_h
at/robot.py", line 91, in servo_move
    time.sleep(step_delay)
KeyboardInterrupt
pi@raspberrypi:~/pisloth/examples $ cd ..
pi@raspberrypi:~/pisloth $ sudo bash i2samp.sh
Support for your operating system is experimental. Please visit
forums.adafruit.com if you experience issues with this product.


This script will install everything needed to use
i2s amplifier

--- Warning ---

Always be careful when running scripts and commands
copied from the internet. Ensure they are from a
trusted source.

If you want to see what this script does before
running it, you should run:
    \curl -sS github.com/adafruit/Raspberry-Pi-Installer-Scripts/i2samp

Do you wish to continue? [y/N]
```

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。

```
pi@raspberrypi: ~/pisloth
running it, you should run:
    \curl -sS github.com/adafruit/Raspberry-Pi-Installer-Scripts/i2samp

Do you wish to continue? [y/N] y

Checking hardware requirements...

Adding Device Tree Entry to /boot/config.txt
dtoverlay=hifiberry-dac
dtoverlay=i2s-mmap

Commenting out Blacklist entry in
/etc/modprobe.d/raspi-blacklist.conf

Disabling default sound driver
Configuring sound output

Installing aplay systemd unit

You can optionally activate '/dev/zero' playback in
the background at boot. This will remove all
popping/clicking but does use some processor time.

Activate '/dev/zero' playback in background? [RECOMMENDED] [y/N]
```

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。

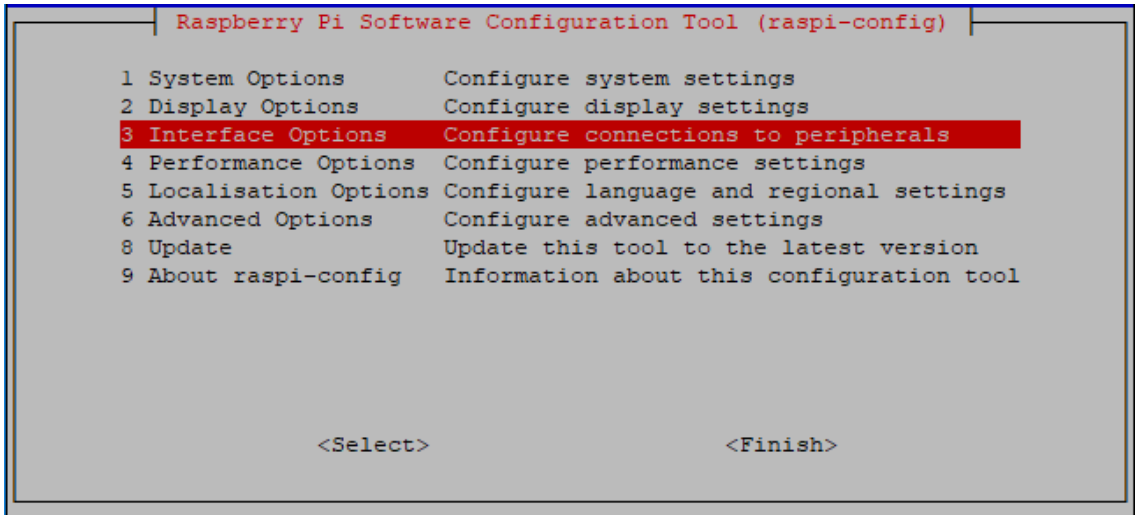
4.1.5 I2C インターフェースを有効にする（重要）

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。

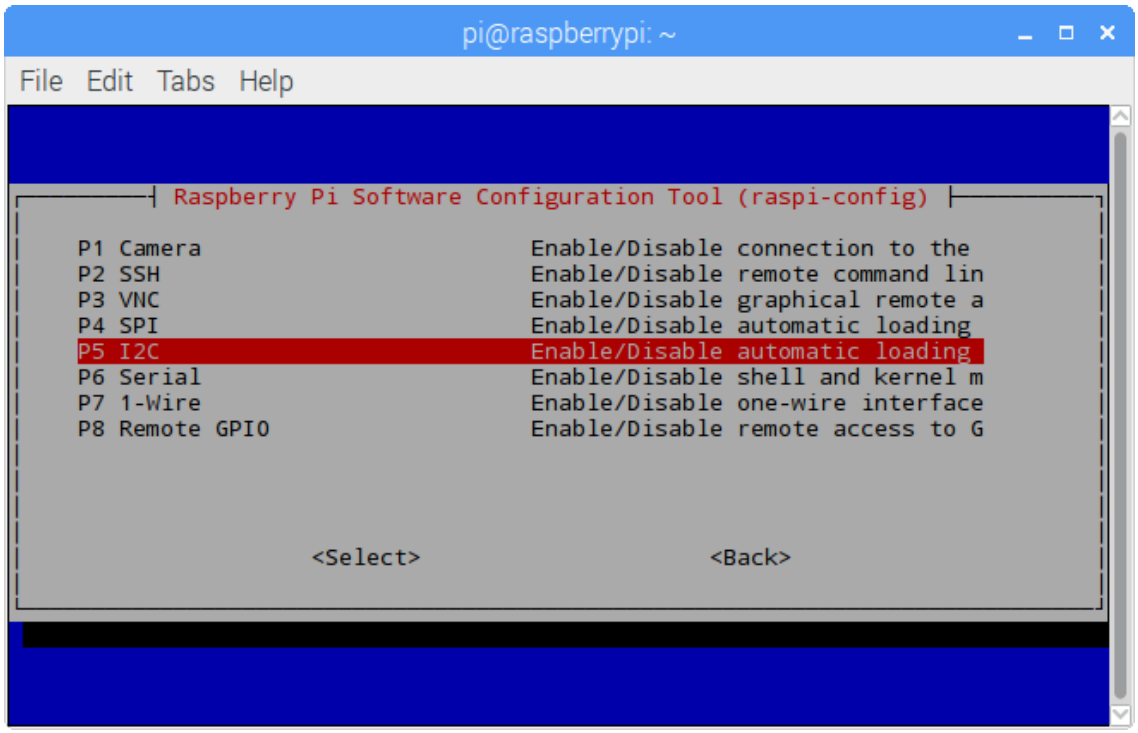
1. 次のコマンドを入力します：

```
sudo raspi-config
```

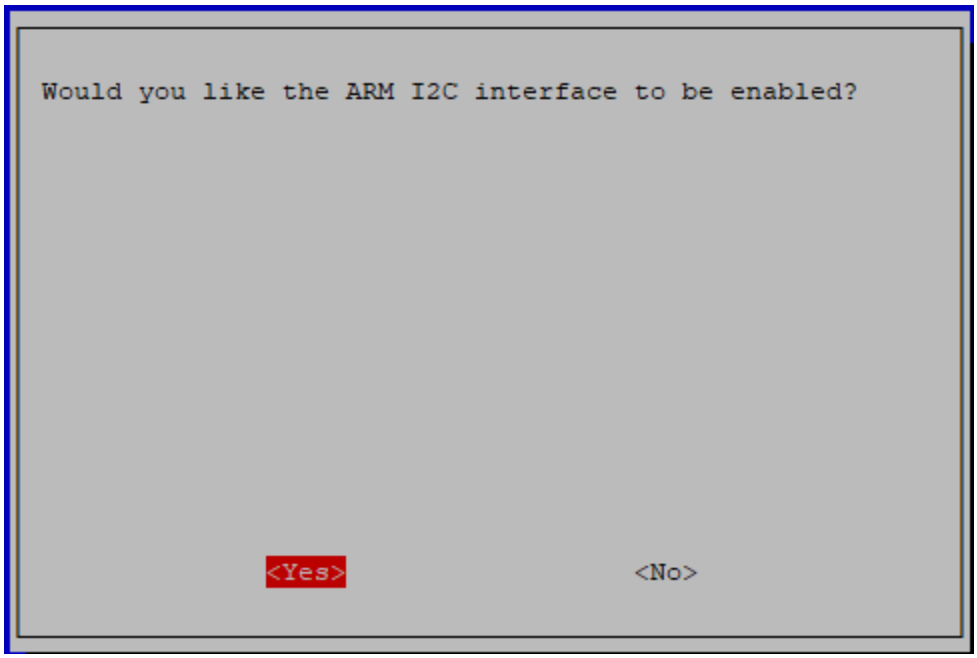
2. キーボードの下矢印キーを押して **Interfacing Options** を選択し、 **Enter** キーを押します。



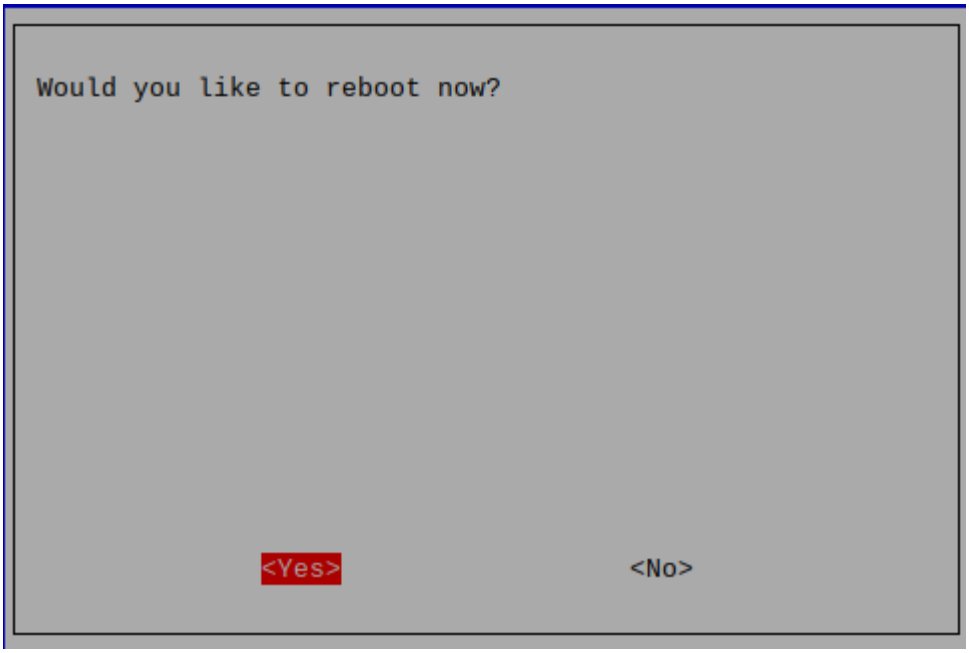
3. 次に **I2C** を選択します。



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。

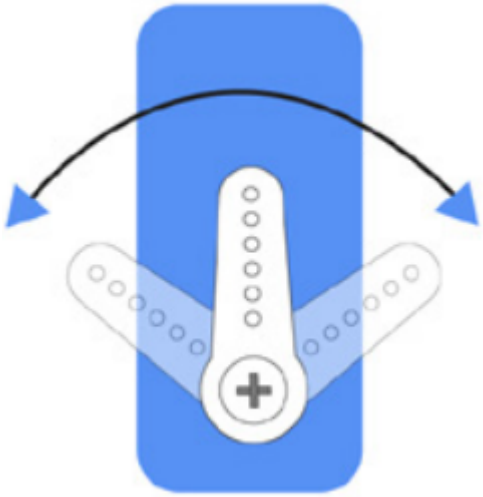


4.1.6 サーボ調整（重要）

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。

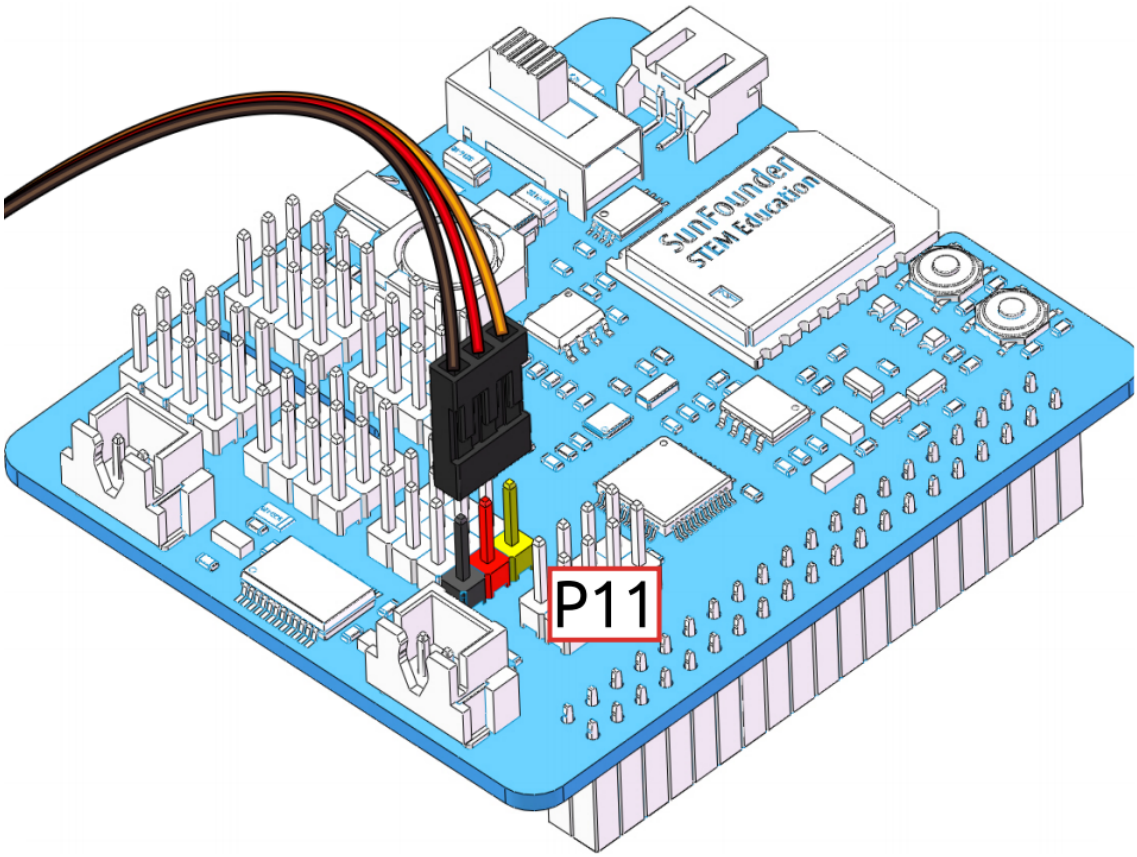
1. サーボが正しく 0 ° に設定されていることを確認するために、まずサーボアームをサーボシャフトに挿入し、その後ロッカーアームを優しく異なる角度に回転させます。このサーボアームは、サーボが回転していることをはっきりと確認できるようにするためのものです。



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。

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：

4.2 0. PiCar-X の校正

4.2.1 モーターとサーボの校正

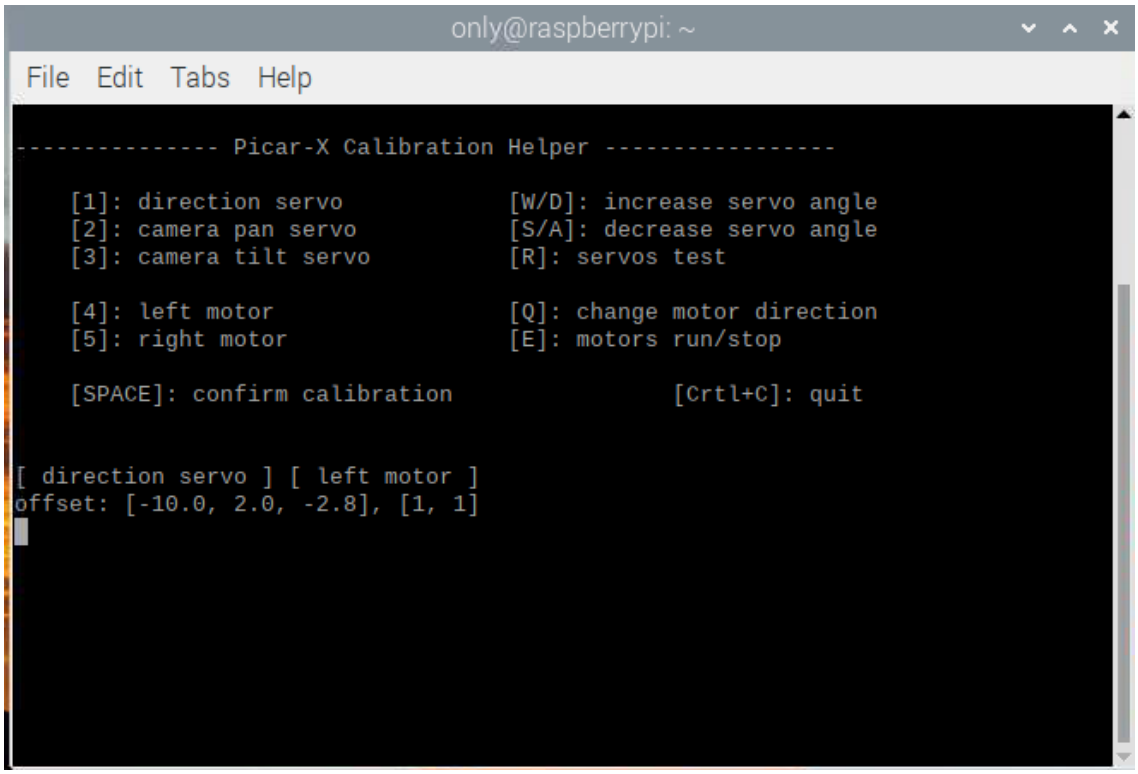
PiCar-X の取り付け中やサーボ自体の限界により、サーボの角度が多少傾くことがあるため、校正が可能です。

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。

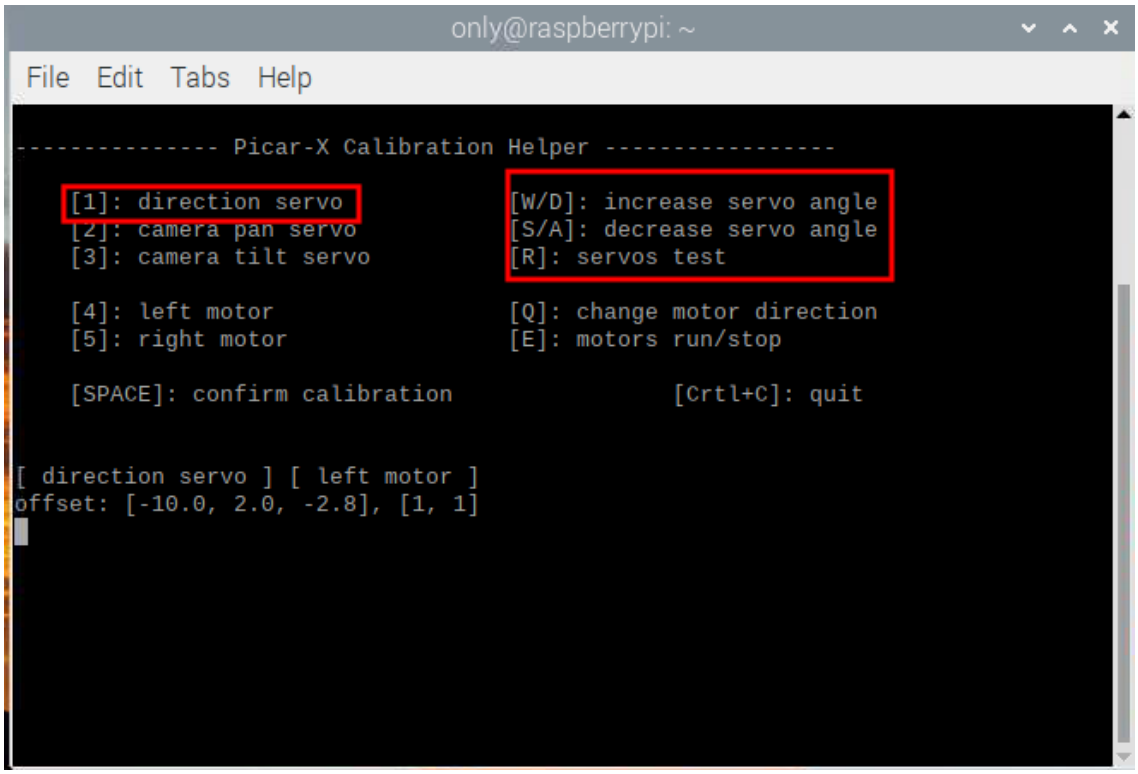
```
cd ~/picar-x/example/calibration
sudo python3 calibration.py
```

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。

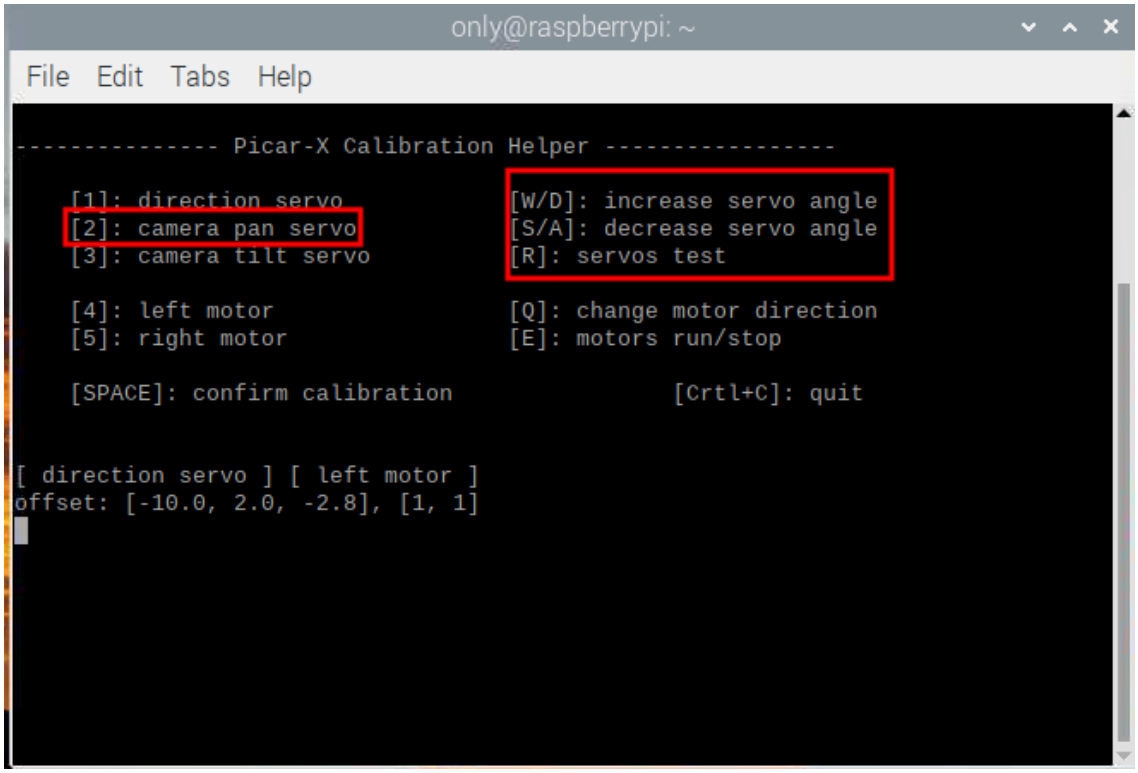


3. R キーは、3 つのサーボが正常に動作しているかど うかをテストするために使用します。 1、 2 、 3 キーでサーボを選択したら、 R キーを押してそのサーボをテストします。

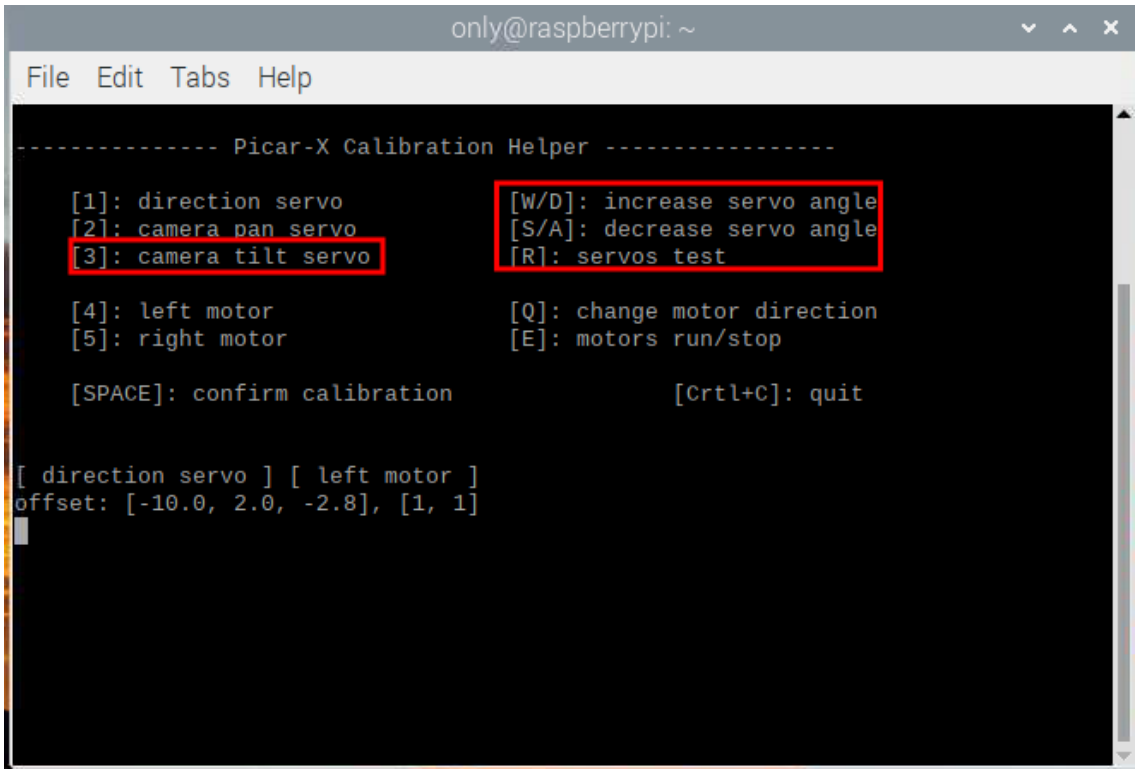
4. 数字キー 1 を押して前輪サーボを選択し、その後 W/S キーを押して前輪が左右に傾かずに可能な限り前を向くようにします。



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。

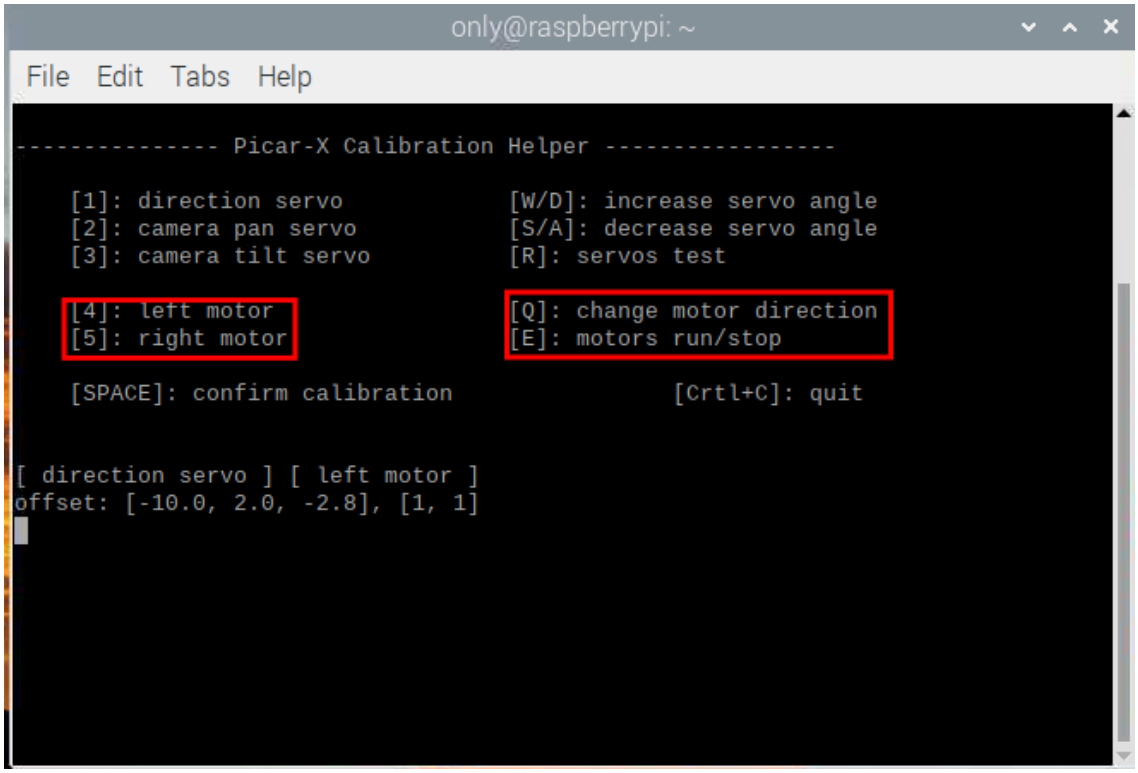


6. 数字キー 3 を押して **tilt servo** を選択し、その後 W/S キーを押してパン/チルトプラットフォームが真っ直ぐ前を向き、上下に傾かないようにします。

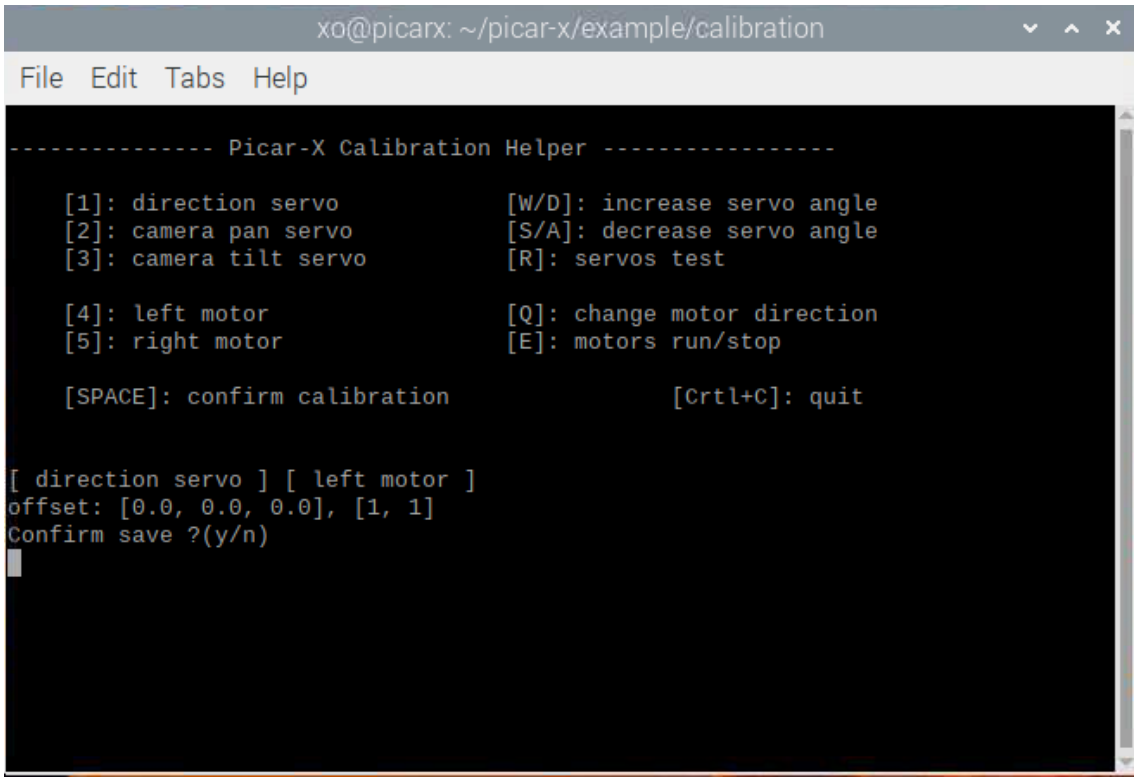


7. 取り付け中にモーターの配線が逆になっている可能性があるため、 E を押して車が正常に前進できるかどう

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。



4.2.2 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。

4.3 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。

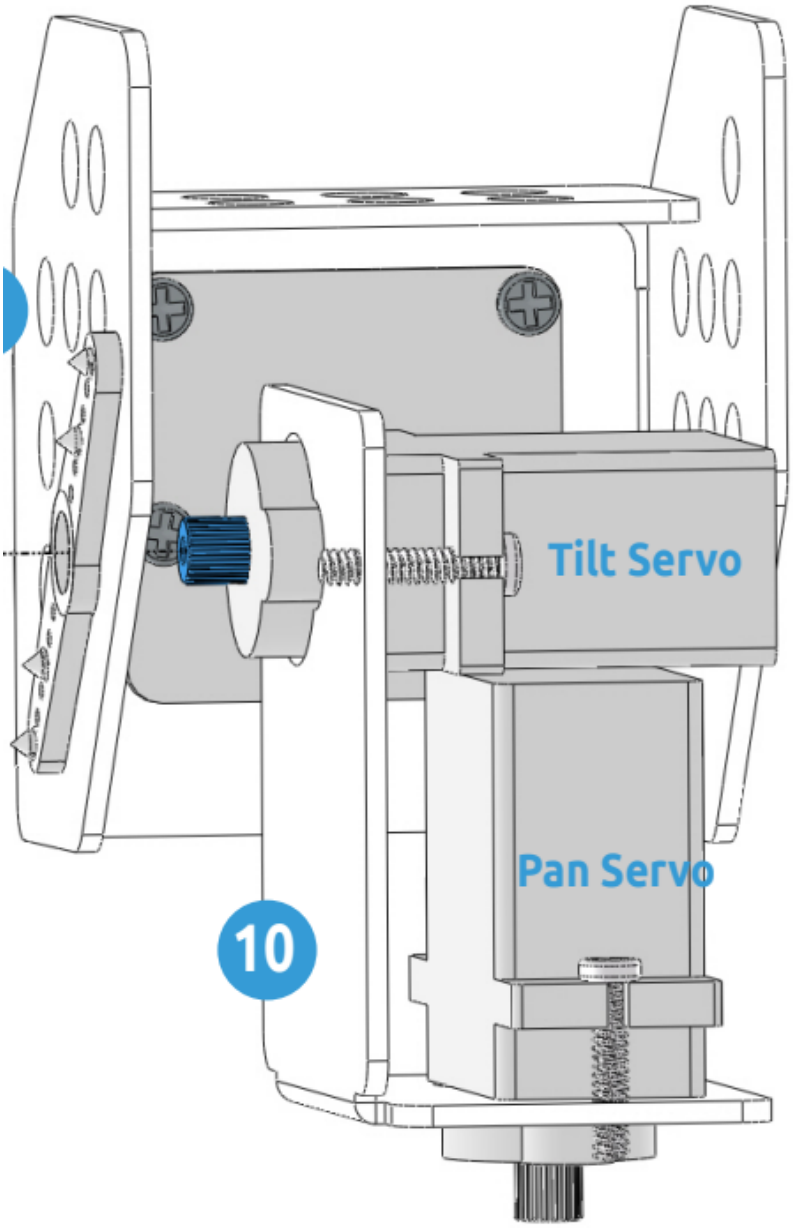
```
from picarx import Picarx
import time
```

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。



4.4 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
```

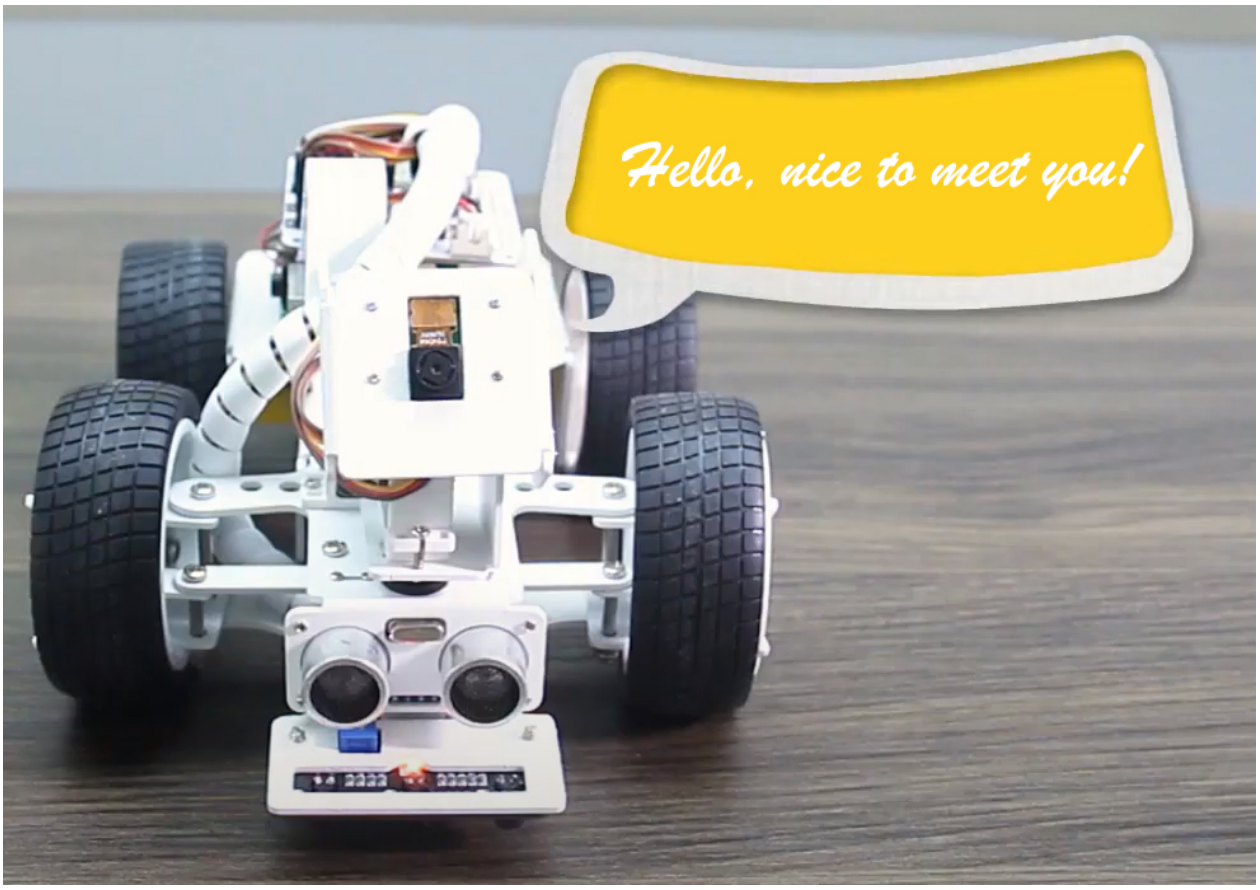
(次のページに続く)

(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
```

4.5 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。

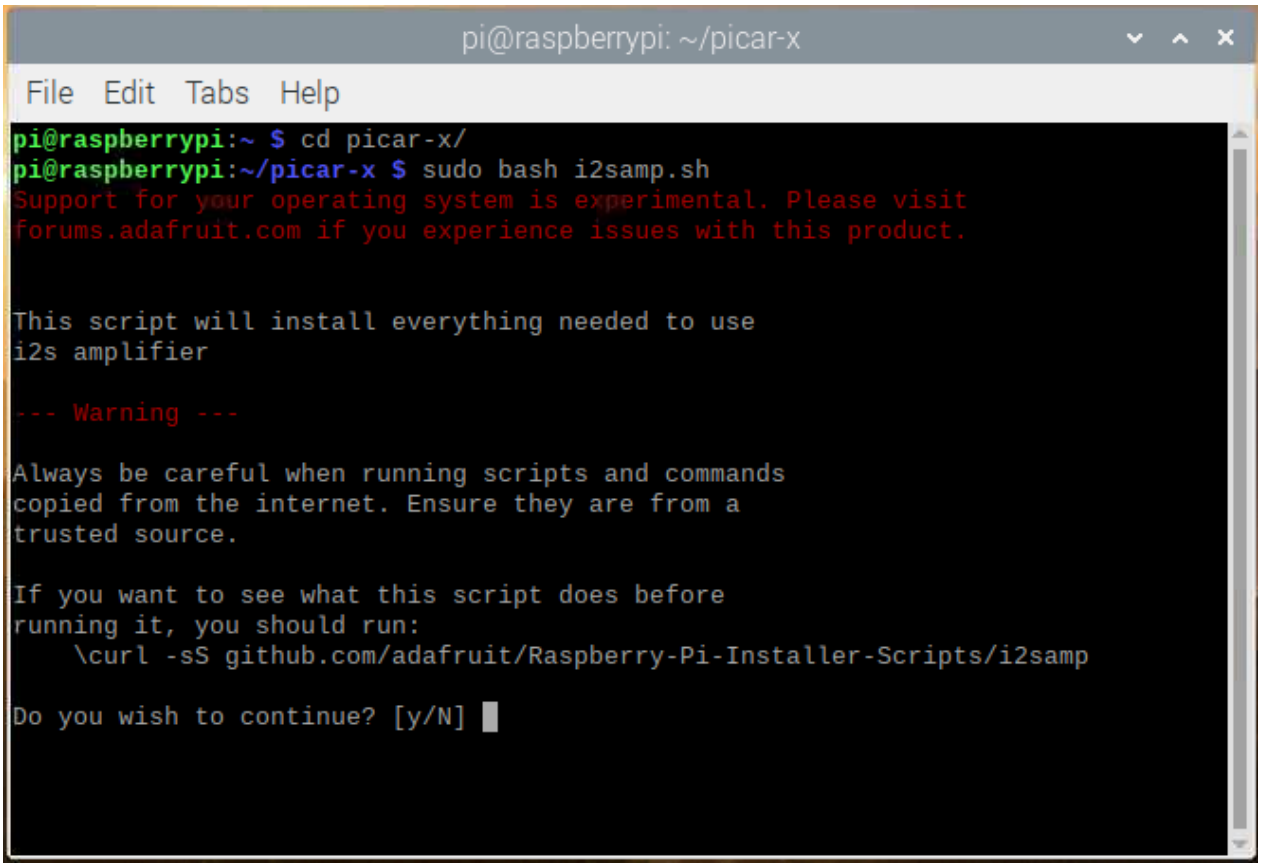


i2samp のインストール

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

4.6 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。

4.7 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。

4.8 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。

```python
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。

4.9 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

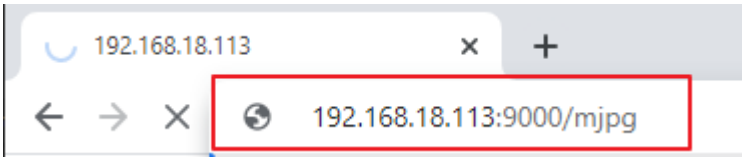
```
cd ~/picar-x/example
sudo python3 7.display.py
```

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

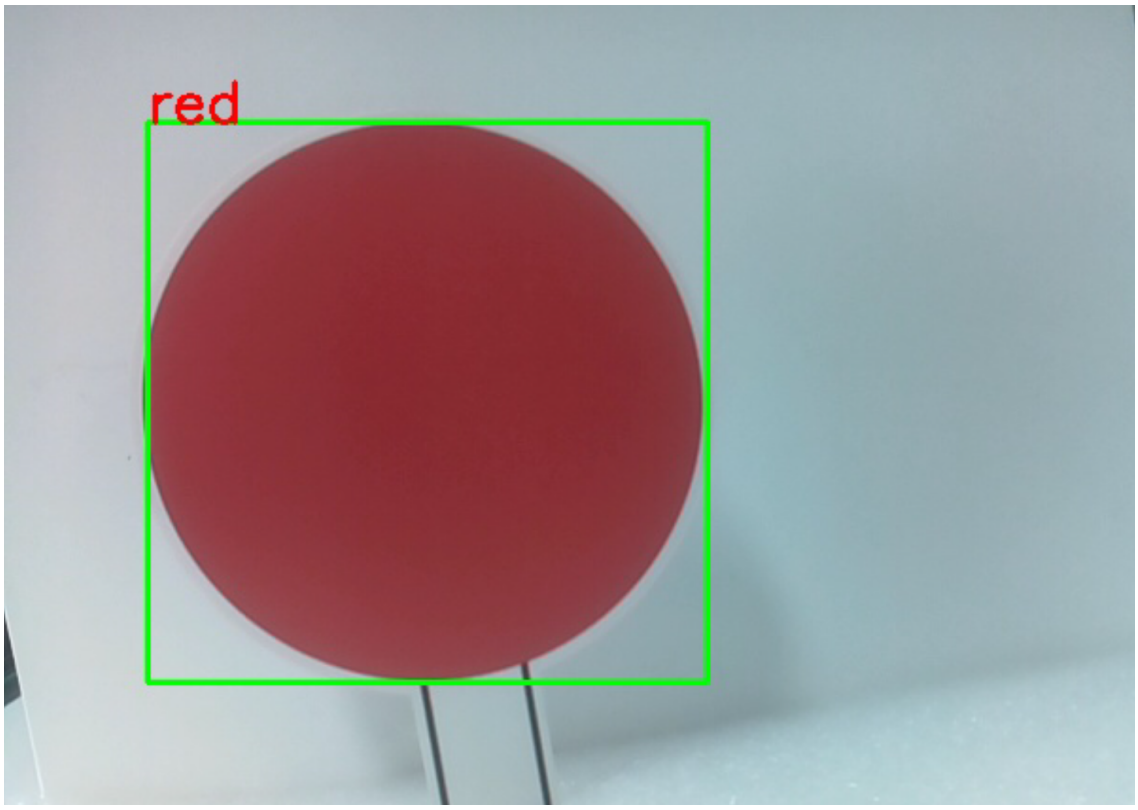
プロンプトに従って、対応する機能をアクティブにしてください。

- **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**

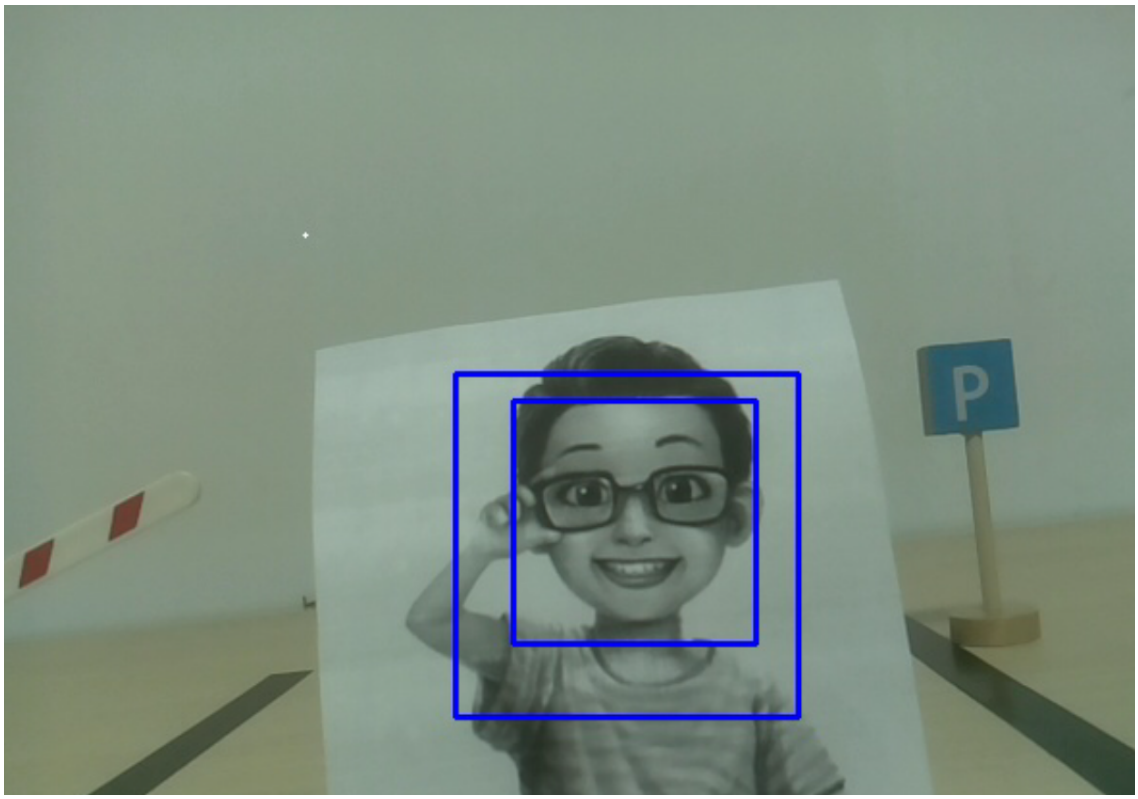
  `1〜6` の間の数字を入力すると、「赤、オレンジ、黄色、緑、青、紫」のうちの一つの色を検出します。`0` を入力すると、色の検出をオフにします。



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

4.10 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。

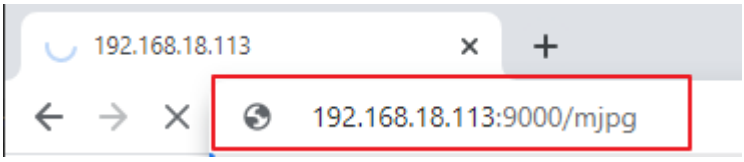
4.11 9. ビデオ録画

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。

4.12 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

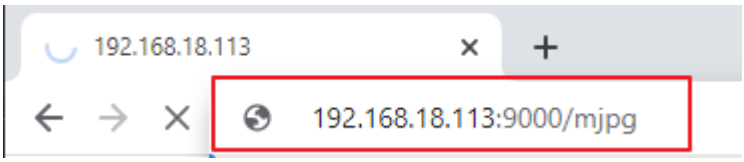
```
cd ~/picar-x/example
sudo python3 10.bull_fight.py
```

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。

4.13 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。

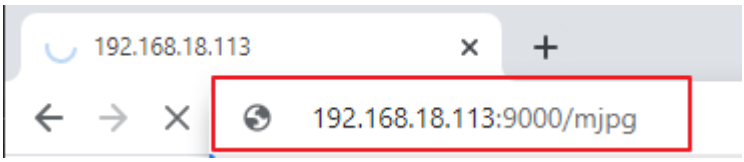
- O: 速度アップ
- P: 速度ダウン
- W: 前進
- S: 後進
- A: 左折
- D: 右折
- F: 停止
- T: 写真を撮る
- Ctrl+C: 終了

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()
```

4.14 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

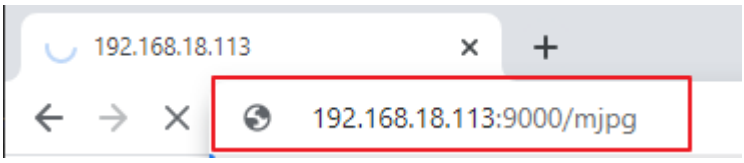
```
cd ~/picar-x/example
sudo python3 12.treasure_hunt.py
```

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。

4.15 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。

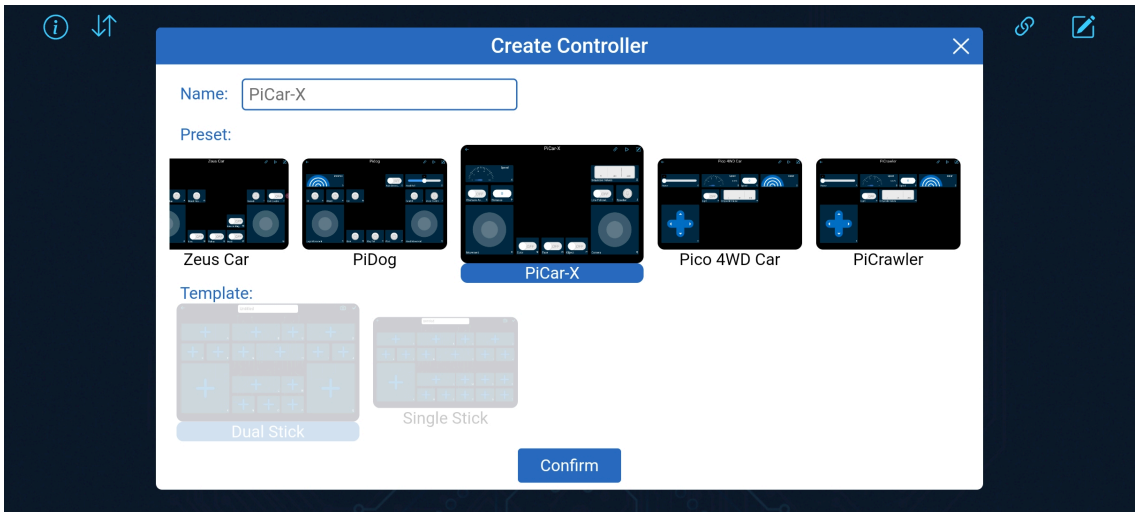
```
cd ~/picar-x/example
sudo python3 13.app_control.py
```

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。

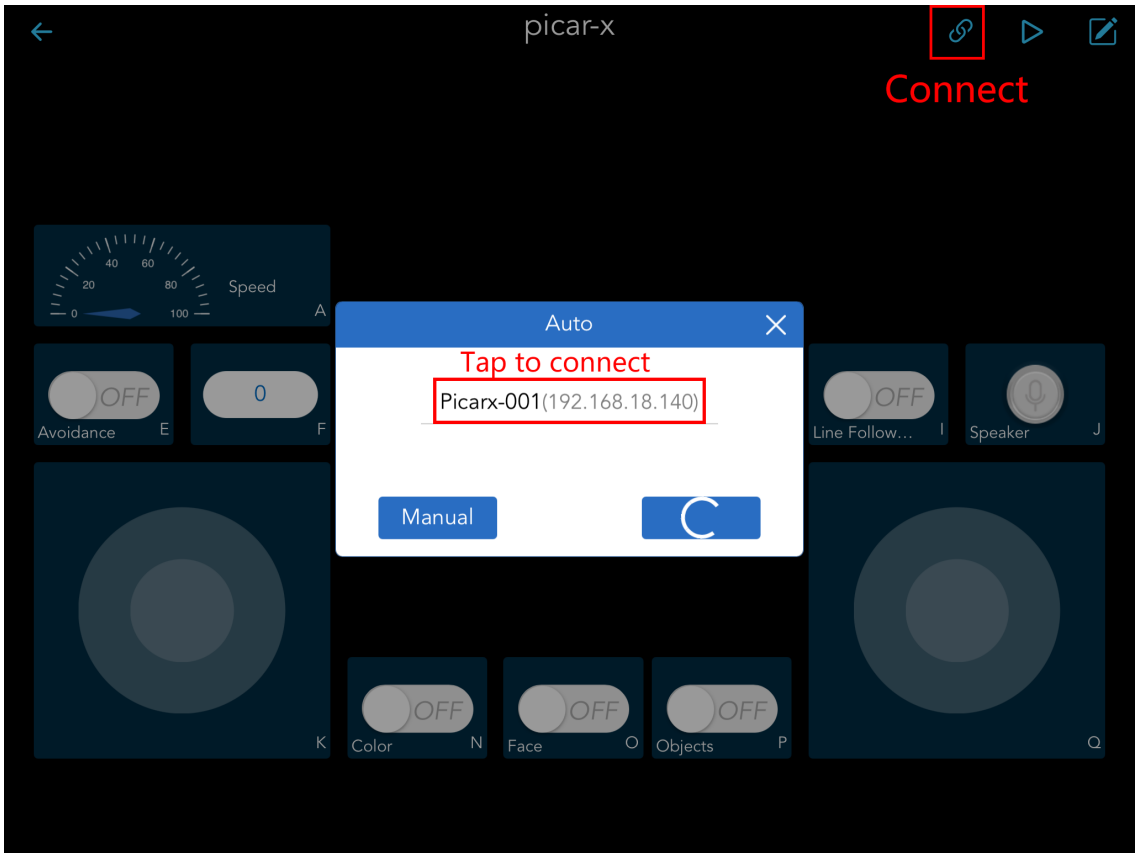
   SunFounder Controller アプリ内の + 記号をクリックして新しいコントローラーを作成します。

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。

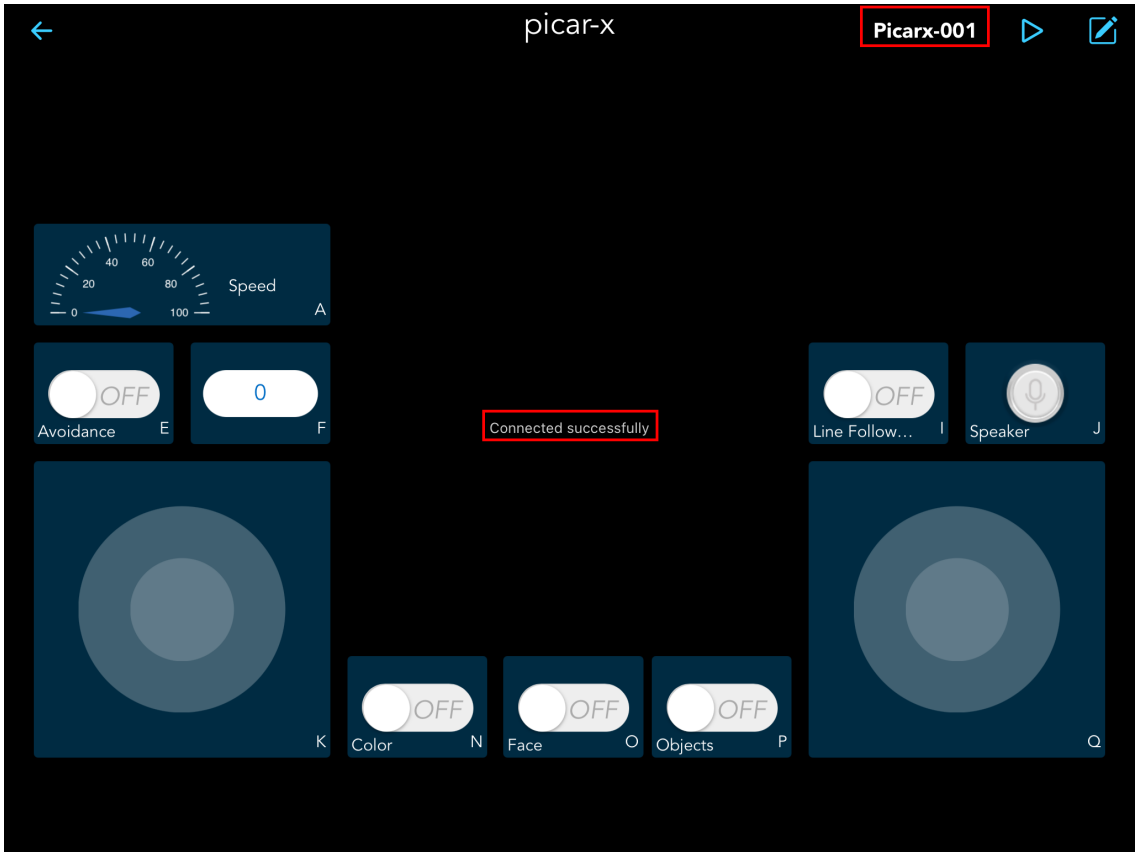


5. PiCar-x に接続する。

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。

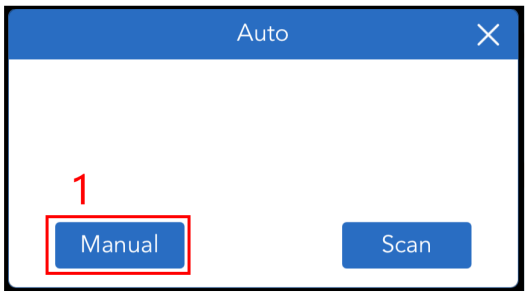


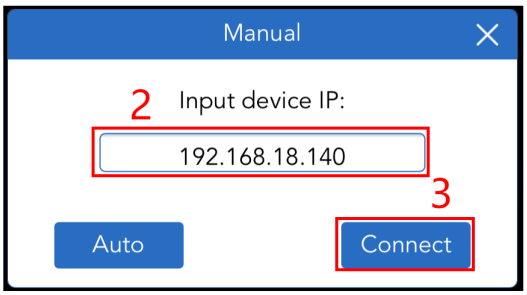
製品名をクリックすると「接続成功」というメッセージが表示され、製品名が右上に表示されます。



注釈:

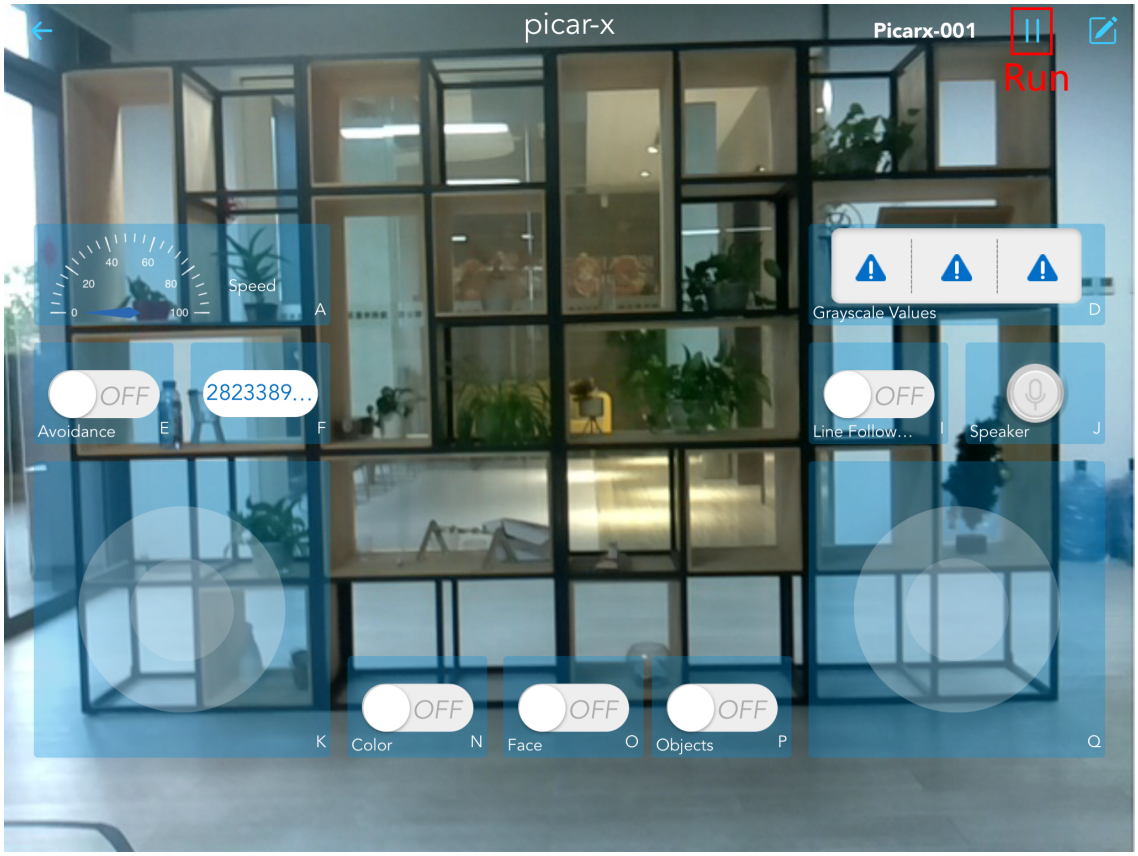
- モバイルデバイスが PiCar-X と同じ LAN に接続されていることを確認する必要があります。
- 自動検索されない場合は、手動で IP を入力して接続することもできます。





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

Ezblock の利用に関して

初心者および初級者向けの EzBlock は SunFounder が提供する Raspberry Pi 用のソフトウェア開発プラットフォームです。 Ezbock 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ます。

Mac PC、Windows PC、Android、iPad など、ほぼすべてのタイプのデバイスで利用できます。

これは EzBlock のインストール、ダウンロード、および使用するのに役立つチュートリアルです。

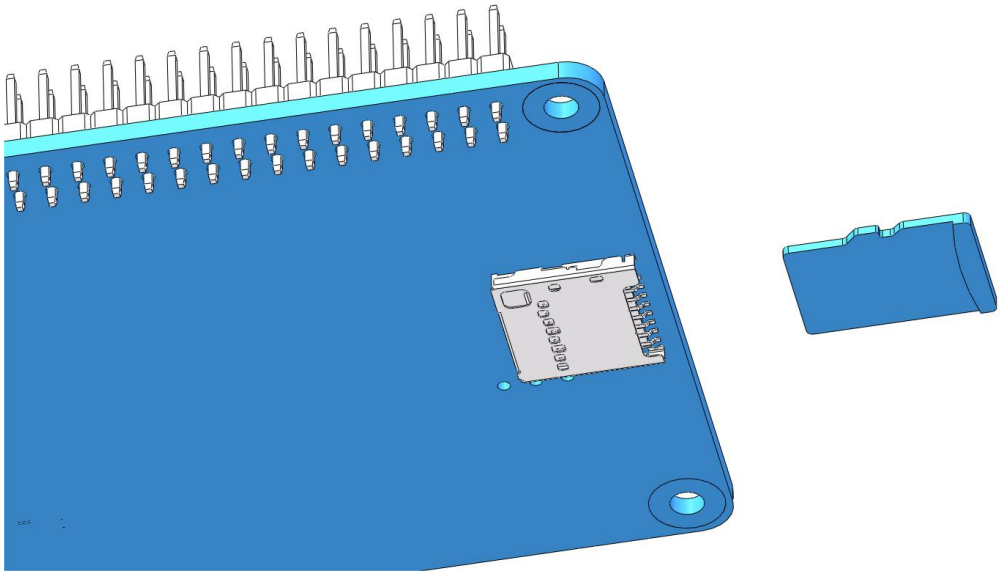
5.1 EzBlock のクイックガイド

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 ° であったり、45 ° であったりします。このような角度のまま組み立てると、ロボットがコードを実行した後の状態が混乱する可能性があります。さらに、サーボがブロックされて焼ける恐れもあります。

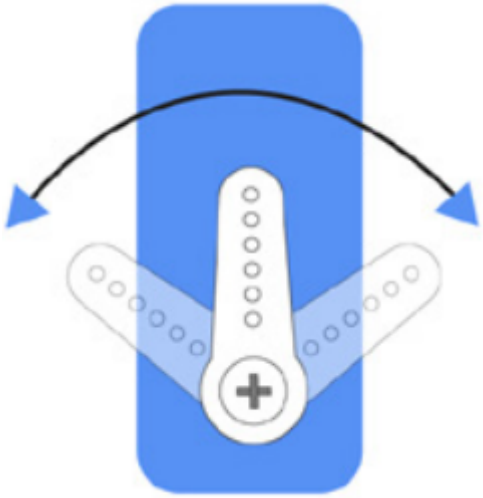
したがって、すべてのサーボの角度を 0 ° に設定してから取り付ける必要があります。これにより、サーボの角度は中央になり、どちらの方向にも回転します。

1. まず、Install EzBlock OS（EzBlock の専用チュートリアル）を Micro SD カードにインストールします。インストールが完了したら、Raspberry Pi に挿入します。

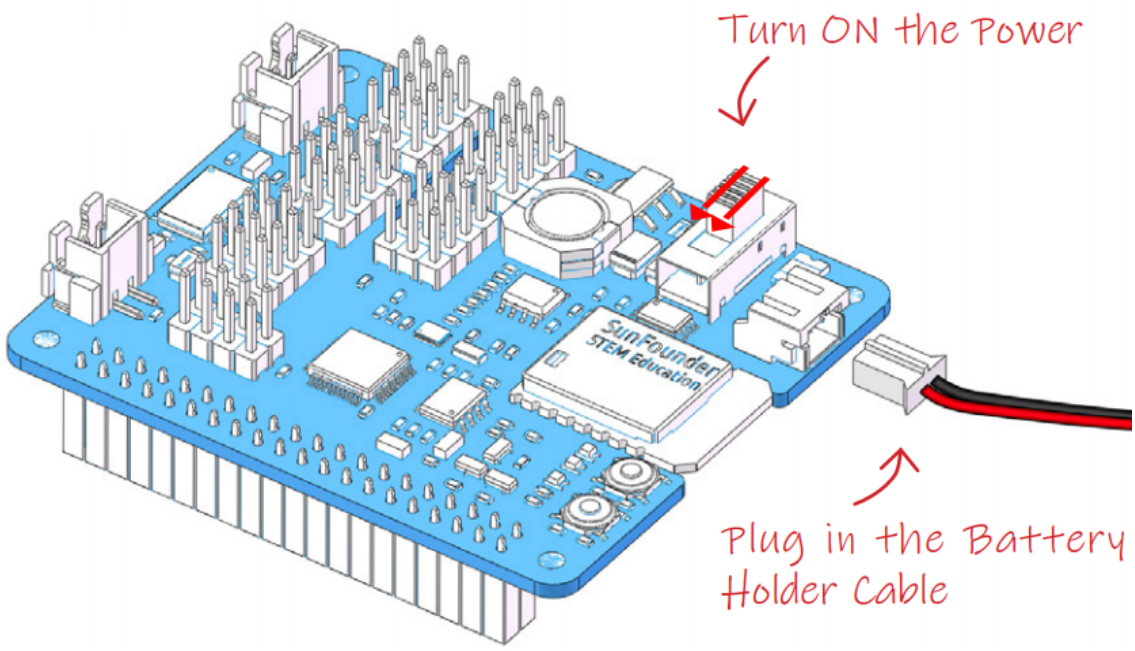
注釈: インストールが完了したら、このページに戻ってください。



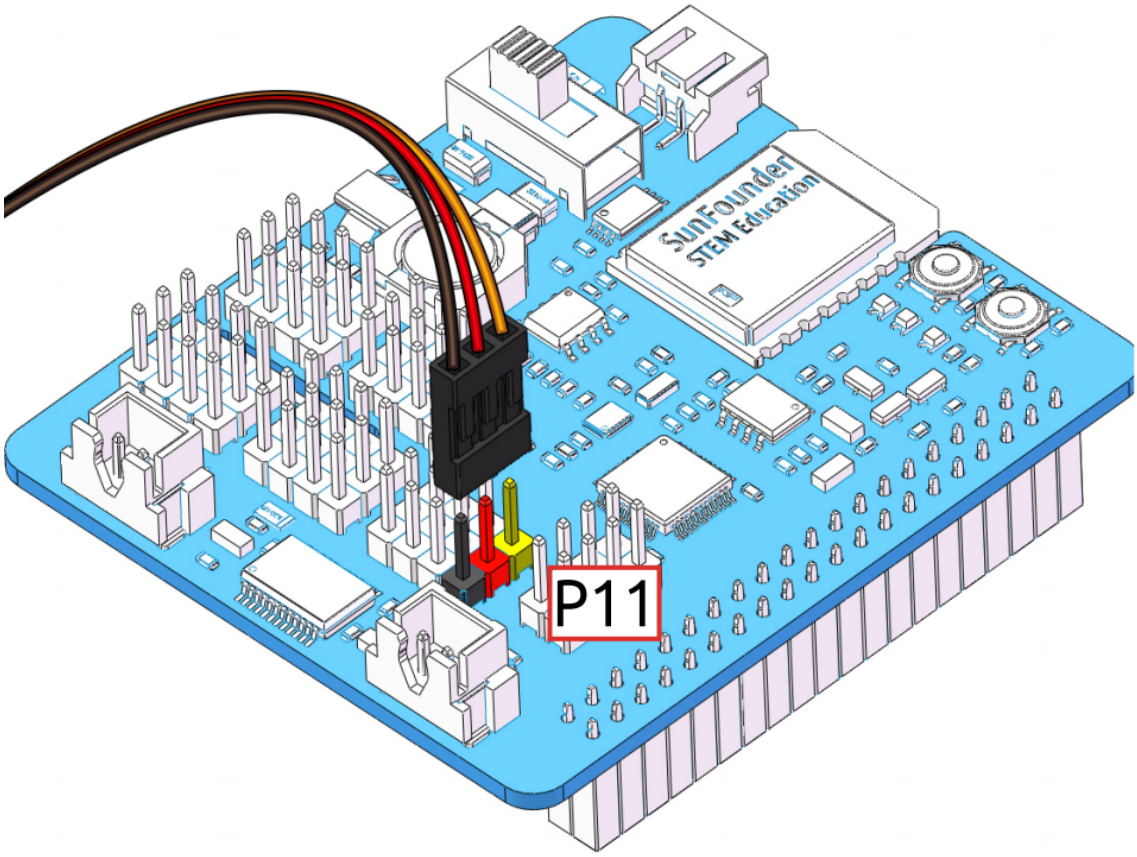
2. サーボが正しく 0 ° に設定されていることを確認するため、サーボアームをサーボシャフトに挿入し、ロッカーアームを別の角度にやさしく回転させます。このサーボアームは、サーボが回転しているのをはっきりと見るためのものです。



3. 組み立て説明書の指示に従い、バッテリーホルダーケーブルを挿入し、電源スイッチを ON にします。1-2 分待つと、Raspberry Pi が正常に起動したことを示す音が鳴ります。



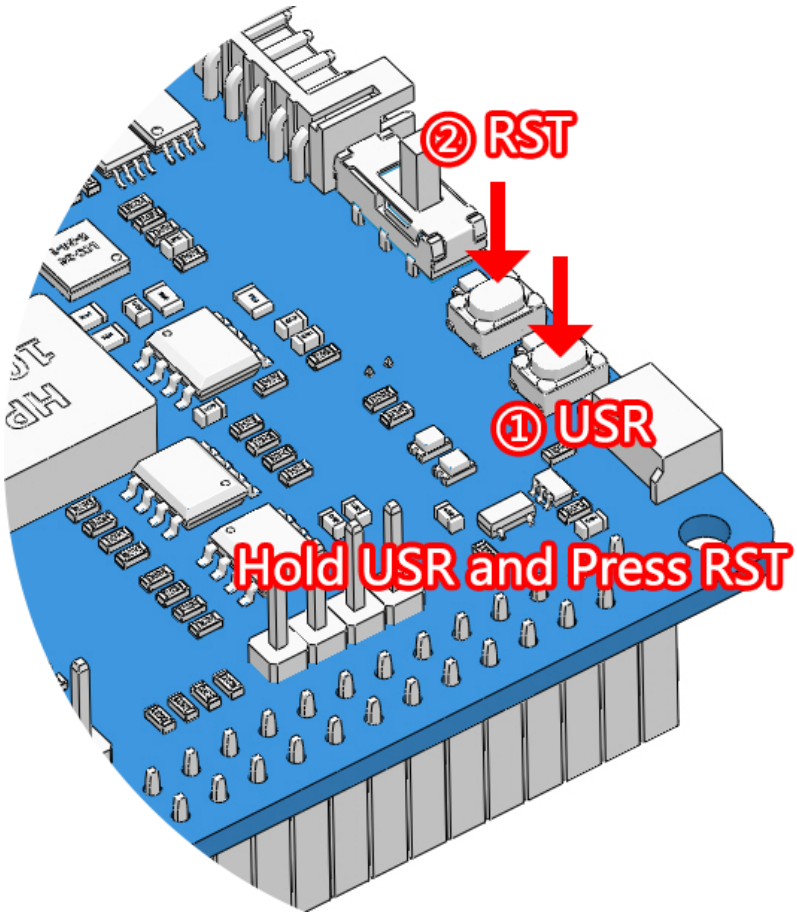
4. 次に、サーボケーブルを以下のように P11 ポートに接続します。



5. **USR** キーを押し続けたまま、 **RST** キーを押して、システム内のサーボゼロリングスクリプトを実行します。サーボアームが位置に回転するのを見ると（これは 0 ° の位置であり、ランダムな位置であり、垂直ま

たは平行でない場合があります。)、プログラムが実行されたことを示します。

注釈: このステップは一度だけ行う必要があります。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. これで、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て、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ます。

注釈:

- このサーボケーブルは、サーボをサーボネジで固定する前には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。損傷の原因となる可能性があります。サーボシャフトが間違った角度で挿入されている場合は、サーボを取り出して再挿入してください。
- 各サーボを組み立てる前に、サーボケーブルを P11 に接続し、電源を入れて角度を 0 ° に設定する必要があります。

- あとで EzBlock APP を使用して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。

5.2 EzBlock Studio のインストールと設定

ロボットの組み立てが終わると最初にいくつかの作業が必要になります。

- Install EzBlock Studio: タブレットやスマートフォンを使う場合は EzBlock Studio のダウンロードとインストールが必要になります。PC の場合ではウェブベースで作業可能です。
- Connect the Product and EzBlock: 利用前の Wi-Fi または Bluetooth の設定とキャリブレーション。
- Open and Run Examples: プログラムの例の表示と実行。

注釈: PiCar-X に接続後にはキャリブレーションが必要です。前に述べた様にわずかなずれ（誤差）が残っているためです。なお通常キャリブレーションは毎回行う必要はなく初めて接続した時だけ行えば十分です。ただ何らかの原因でずれが生じた際はもう一度行うことが可能です。

またもし組み立て時点で完璧に組めたという自信があるのであればこの作業を飛ばしても構いません。

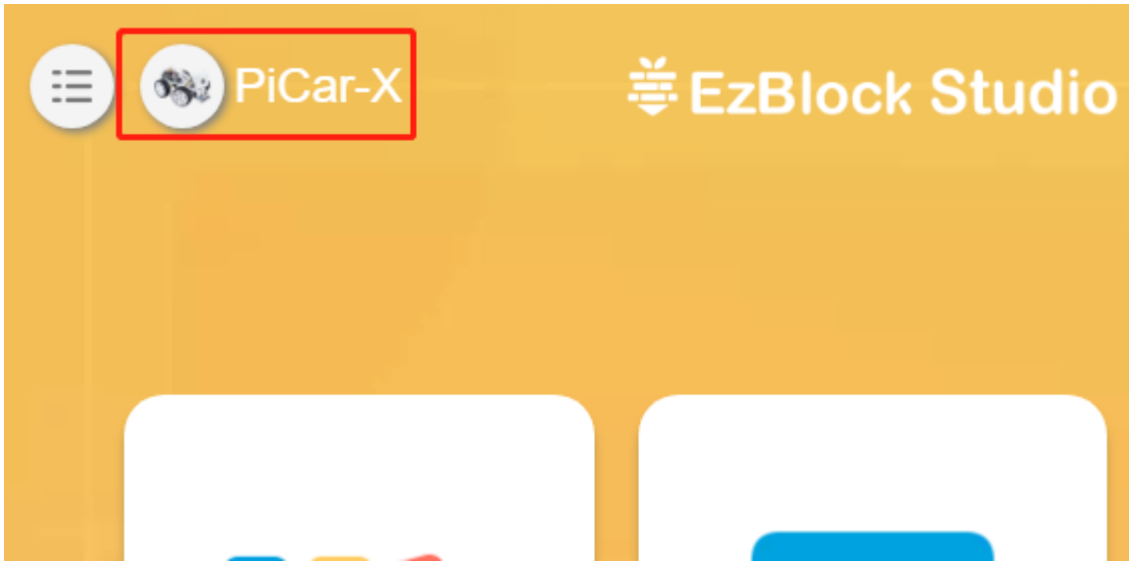
5.3 PiCar-X のキャリブレーション

PiCar-X を初めて接続した後、キャリブレーションの手順があります。PiCar-X の取り付け時の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いている可能性があるため調整することができます。

もちろん組み立てが完璧でキャリブレーションが必要ないと思われる場合はこの章をスキップできます。

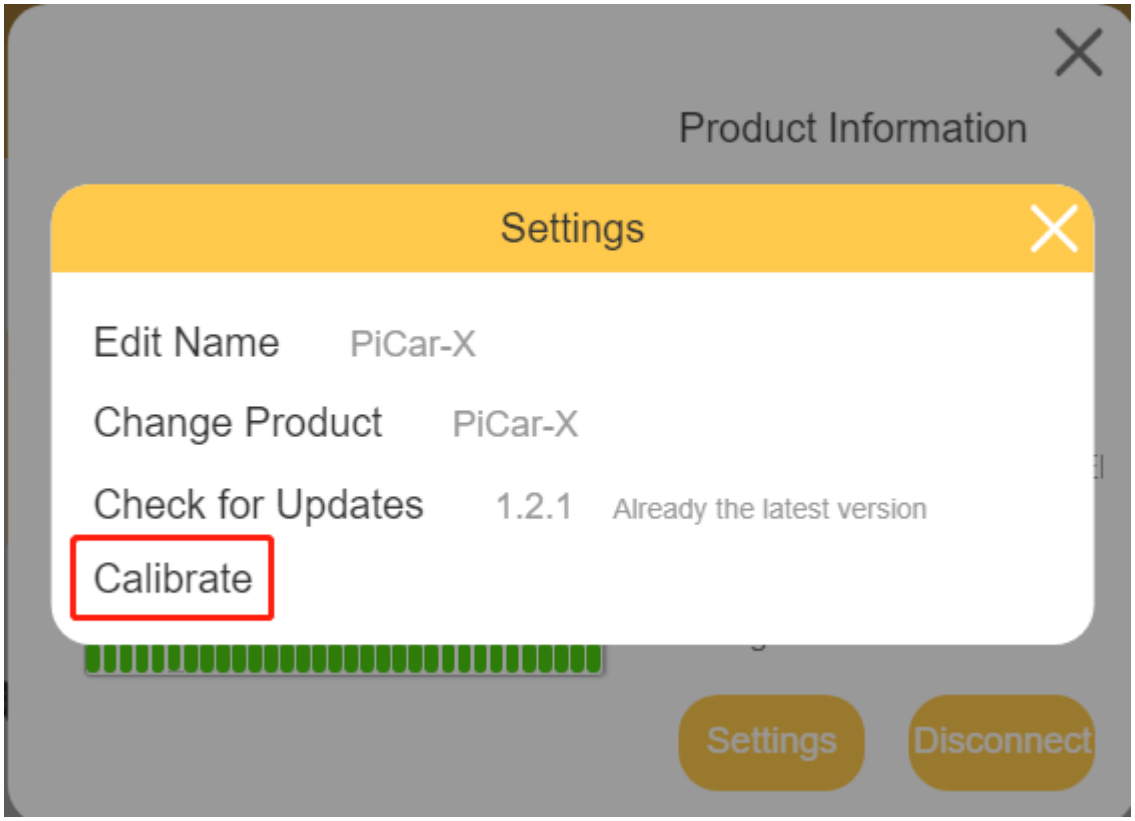
注釈: 使用中にロボットの再キャリブレーションを行う場合は、以下の手順に従ってください。

1. 左上のコネクトアイコンをクリックすると、商品の詳細ページが表示されます。



2. **Settings** ボタンをクリックします。

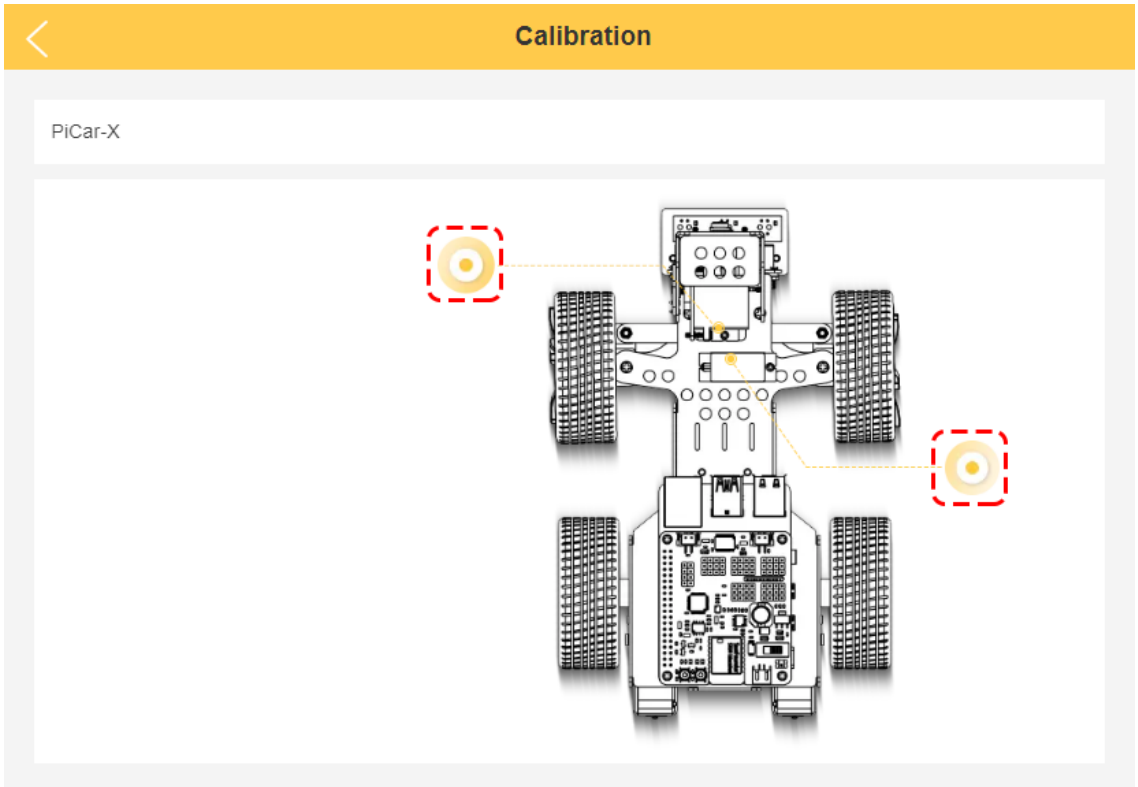
3. このページでは、製品名、製品タイプの変更、アプリのバージョン表示、ロボットのキャリブレーションを行うことができます。 **Calibrate** をクリックすると、キャリブレーションのページに移動します。



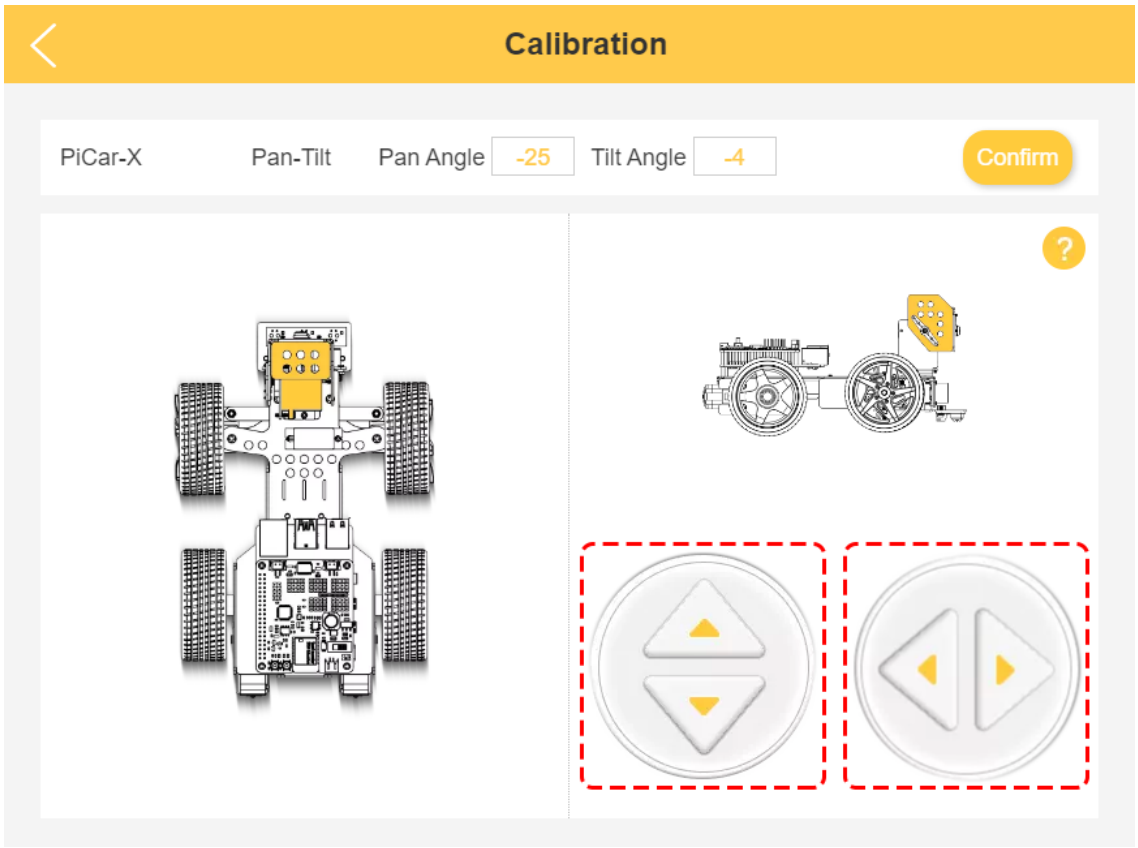
キャリブレーションの手順は以下の通りです:

1. キャリブレーションのページに移動すると、どこでキャリブレーションを行うかを示す 2 つのプロンプトポイントが表示されます。.

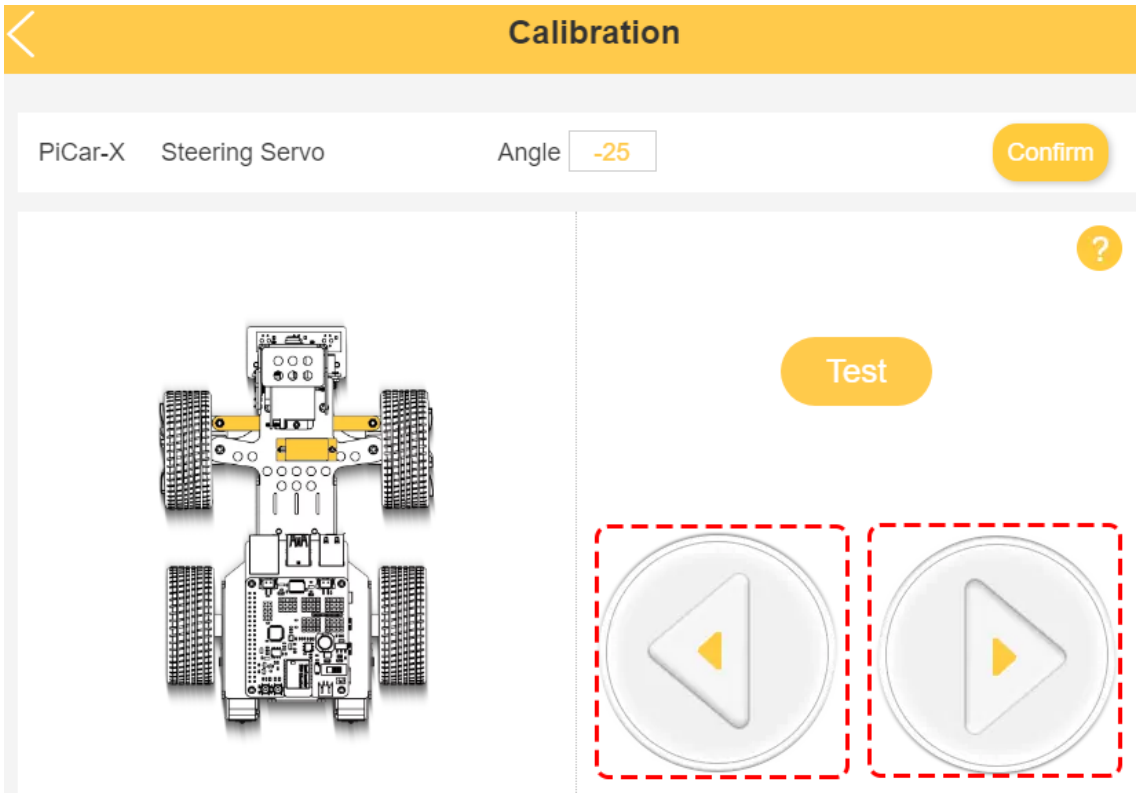
> **注釈:** キャリブレーションは、微小な調整作業です。ボタンを限界までクリックしてもパーツがズレている場合は、一度パーツを外して組み立て直すことをお勧めします。



2. 左側のプロンプトポイントをクリックすると、PiCar-X のパン・チルト（カメラ部）のキャリブレーションを行います。 右側の 2 組のボタンを使って、パン・チルトの向きをゆっくり調整したり、その角度を確認することもできます。 調整が完了したら、 **Confirm** をクリックします。



3. 前輪の向きをキャリブレーションするには、右側のプロンプトポイントをクリックします。右側の 2 つのボタンを使って、前輪がまっすぐ前を向くようにします。調整が完了したら、**Confirm** をクリックします。

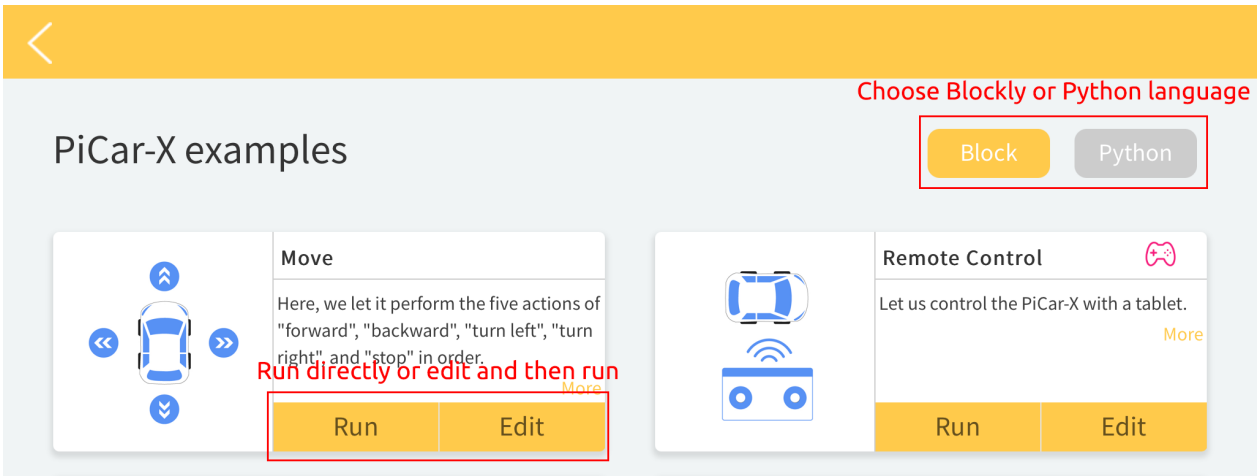


プロジェクト

このセクションは PiCar-X の基本的なプログラミング機能説明から始まり、Ezblock Studio でより高度なプログラムを作成するまで記述されています。各チュートリアルには新しい機能を紹介するティップスが含まれており、ユーザーは対応するプログラムを作成できます。直接使用できる完全な参照コードもサンプル セクションにあります。サンプル 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ることをお勧めします。課題を克服する楽しい経験をお楽しみください。

すべての Ezblock プロジェクトは Ezblock Studio のサンプル ページにアップロードされています。 [サンプル] ページから、ユーザーはプログラムを直接実行したり、サンプルを編集してユーザーの [マイ プロジェクト] フォルダーに保存したりできます。

例では、ユーザーはブロックまたは Python 言語を選択できます。このセクションのプロジェクトでは、ブロック言語についてのみ説明します。Python コードの説明についてはこの **ファイル** <*https://github.com/sunfounder/picar-x/blob/v2.0/docs/(EN)%20picarmini.md*> を参照してください。 Python コードを理解するのに役立ちます。



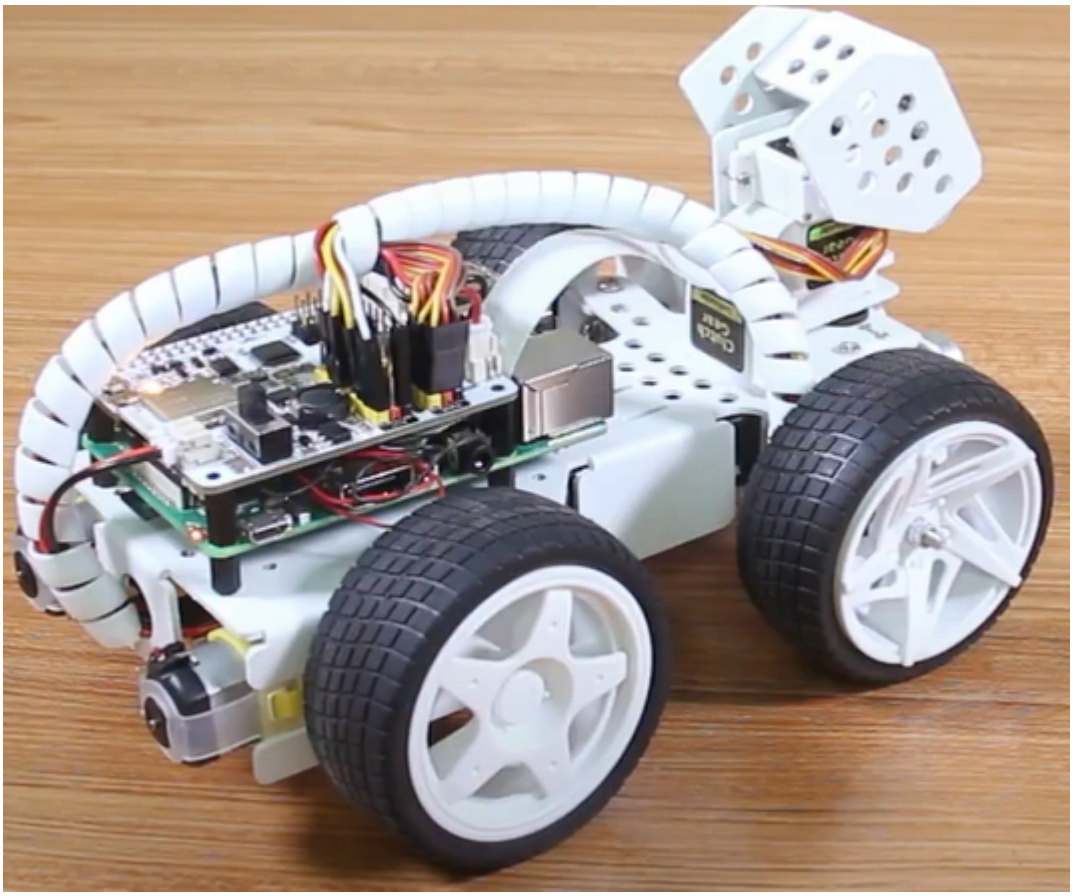
基本

5.4 ムーブ

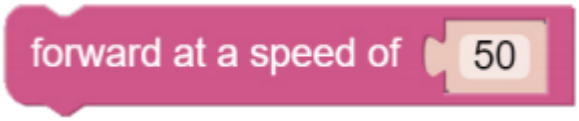
この最初のプロジェクトでは、PiCar-X の基本の動きをプログラムする方法を教えます。 このプロジェクトでは、プログラムは PiCar-X に「前進」、「後退」、「左折」、「右折」、「停止」の 5 つのアクションを順番に実行するよう指示します。

Ezblock Studio の基本的な使い方を学ぶには、次の 2 つの章をお読みください。

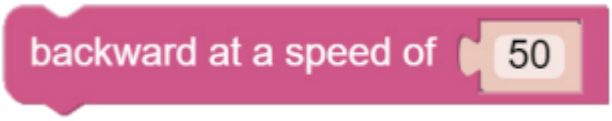
- How to Create a New Project?



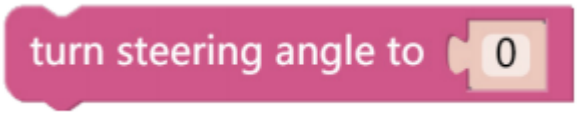
ティップス



このブロックはモーターにかける電圧のパーセンテージに基づいた速度で PiCar-X を前進させます。 以下の例では、「50」は出力の 50%、つまり半分の出力です。モーターの性質上あまり細かいコントロールはできず、大体 10% ずつくらいで差がわかる程度と思ってください。



このブロックは上記と同じようにして後退の速度を指定します。



このブロックは、前輪の向きを調整します。 範囲は「-45」～「45」です。 下の例では、「-30」は前輪が左に 30 ° 回転する (車が左に曲がる) ことを意味します。

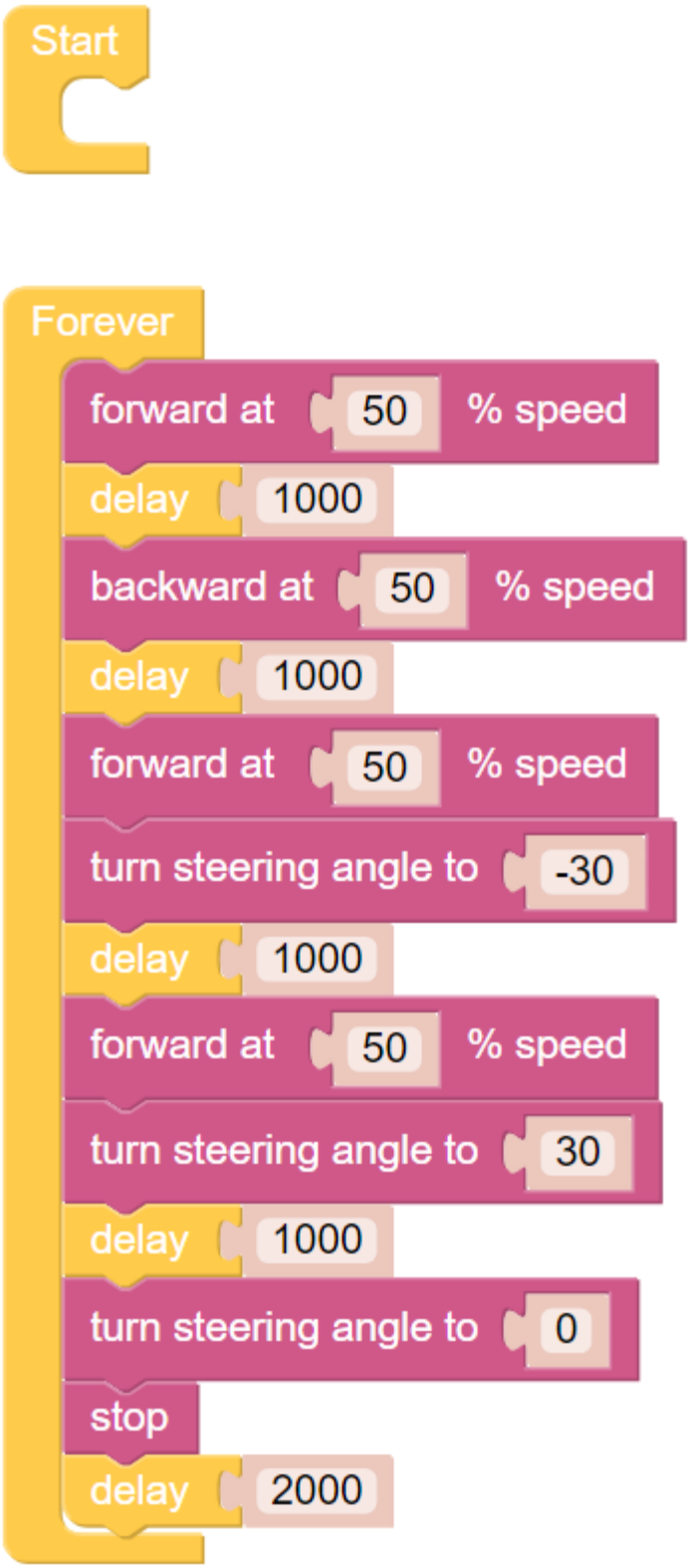
このブロックはプログラムの実行を遅らせる時間をミリ秒（1/1000 秒）単位に指定します。以下の例では、PiCar-X は次のコマンドを実行する前に 1 秒 (1000 ミリ秒) 待ちます。なおこの待ち時間の間前進など事前に設定したコマンドは有効のままですから例えば前進中にハンドルを切ったまましばらく進む（曲がってゆく）場合などに使います。

このブロックは PiCar-X を停止させます。

例：

注釈:

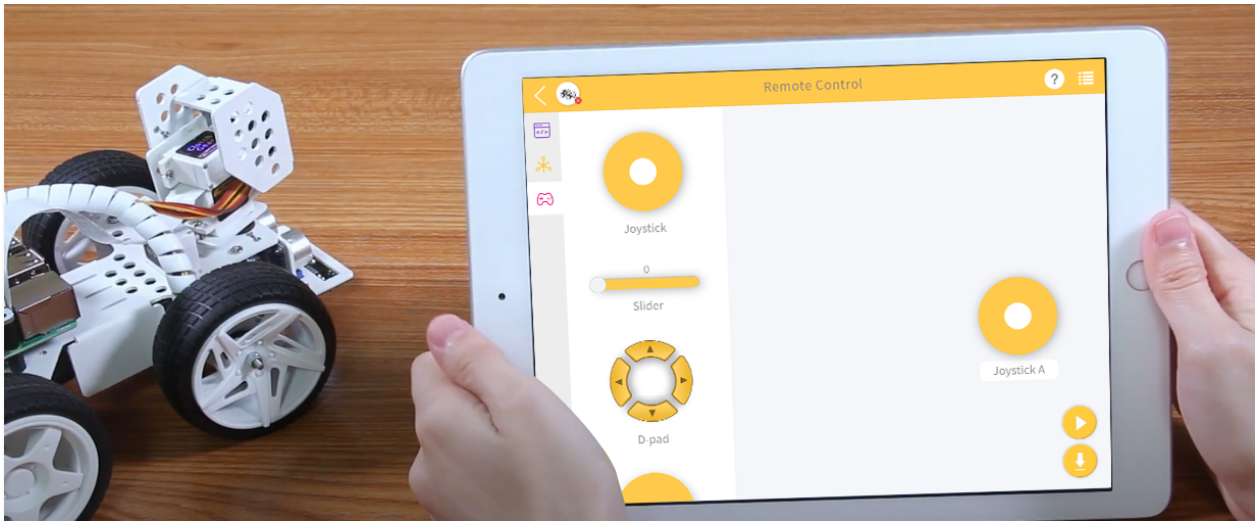
- 次の例を参考にしてプログラムを作成してください。またチュートリアルを参照してください。How to Create a New Project?.
- または EzBlock Studio の **Examples** 画面から「 **Move** 」を探し **Run** か **Edit** を直接クリックしてください。


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

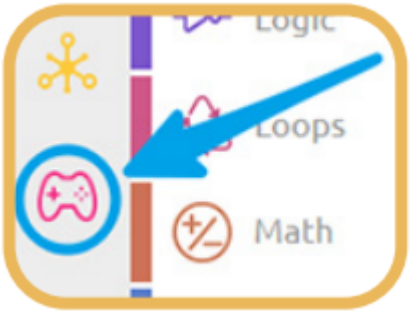
5.5 リモコン・カー

このプロジェクトでは画面上のジョイスティックを使用して PiCar-X をリモコン制御する方法を説明します。Note: リモコン画面からジョイスティックをドラッグ＆ドロップした後、「マップ」機能を使用して、ジョイスティックの X 軸と Y 軸の読み取り値を調整します。 リモコン機能の詳細については、次のリンクを参照してください。

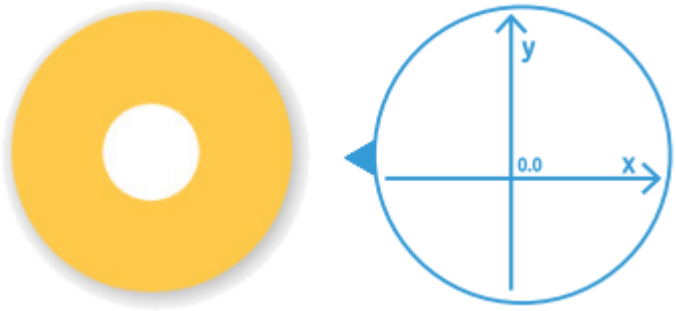
- How to Use the Remote Control Function?



ティップス



リモコン機能を使用するには、メインページの左側からリモコン画面を開きます。



ジョイスティックをリモコン画面の中央領域にドラッグします。 中央の白い点をマウスで任意の方向に軽くドラッグすると、(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。 ジョイスティックを左端にドラッグすると、X 値が「-100」になり、Y 値が「0」になります。

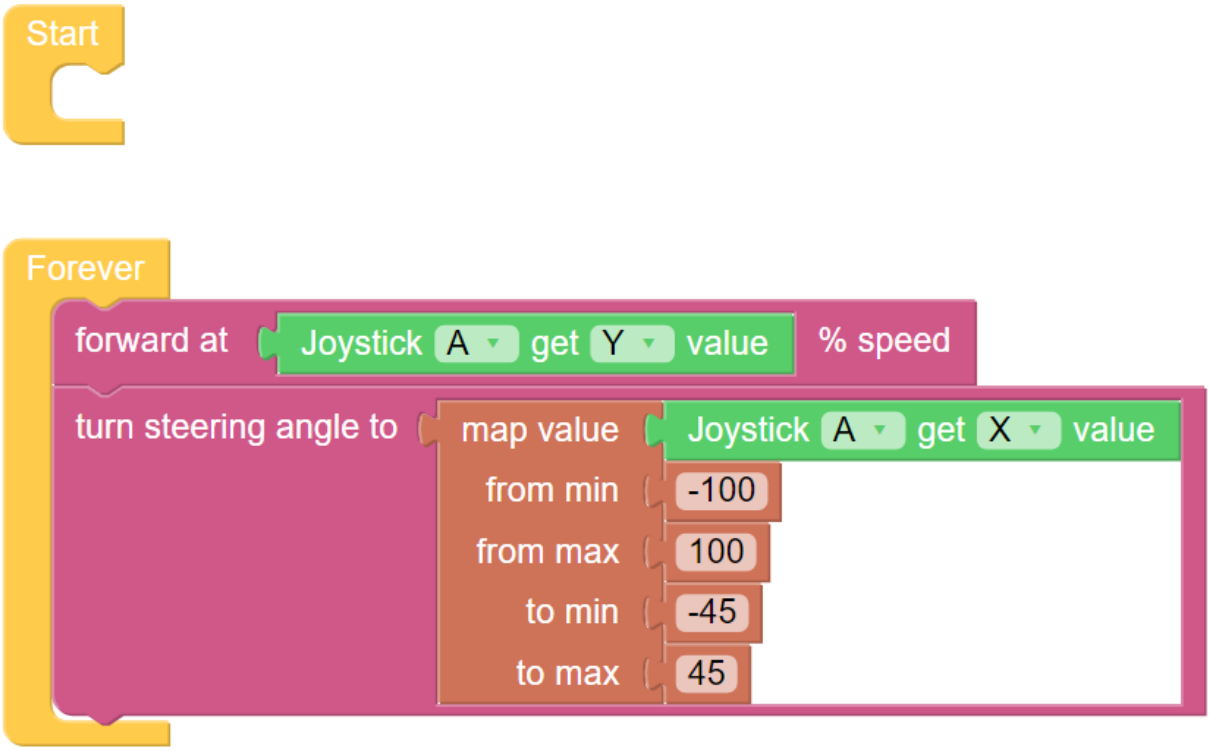
リモコン画面に部品を設定すると、上記のブロックを持つ新しいカテゴリのリモートが表示されます。このブロックは、リモコン画面のジョイスティックの値を読み取ります。 ドロップダウン メニューをクリックして、Y 軸の読み取り値に切り替えることができます。

マップ値ブロックは、数値をある範囲から別の範囲に再マップ（変換）できます。 上の例のように From の範囲が 0 ～ 100 に設定されて、to の範囲が 0 ～ 255 の場合、入力値（map value）が 50 の場合入力範囲の 50%、つまり「50」になります。 変換後の範囲が 0 ～ 255 に設定されていますので出力は 255 の 50%、つまり「127.5」になります。これにより下の例「turn steering angle to」のようにジョイスティックの入力値（-100～100）をハンドル（前輪）の向きの左 45 度～右 45 度（-45～45）に変換したりすることができます。

例：

注釈:

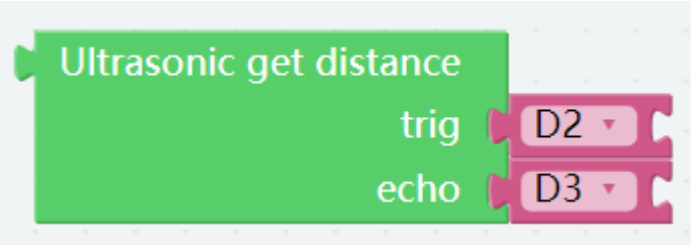
- 次の例を参考にしてプログラムを作成してください。またチュートリアルを参照してください。How to Create a New Project?.
- または EzBlock Studio の **Examples** 画面から「 **Remote Control** 」を探し **Run** か **Edit** を直接クリックしてください。



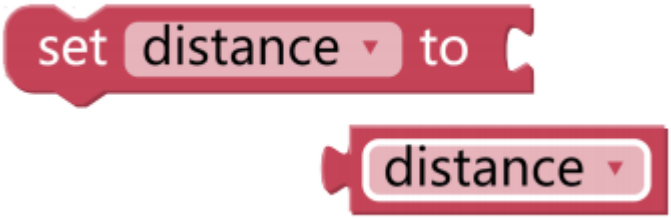
5.6 超音波モジュール試験

PiCar-X には障害物回避や自動物体追跡実験に使用できる超音波センサーが組み込まれています。 このプロジェクトではセンサーは距離をセンチメートル単位で読み取り、結果を **デバッグ** ウィンドウに **表示** します。

ティップス



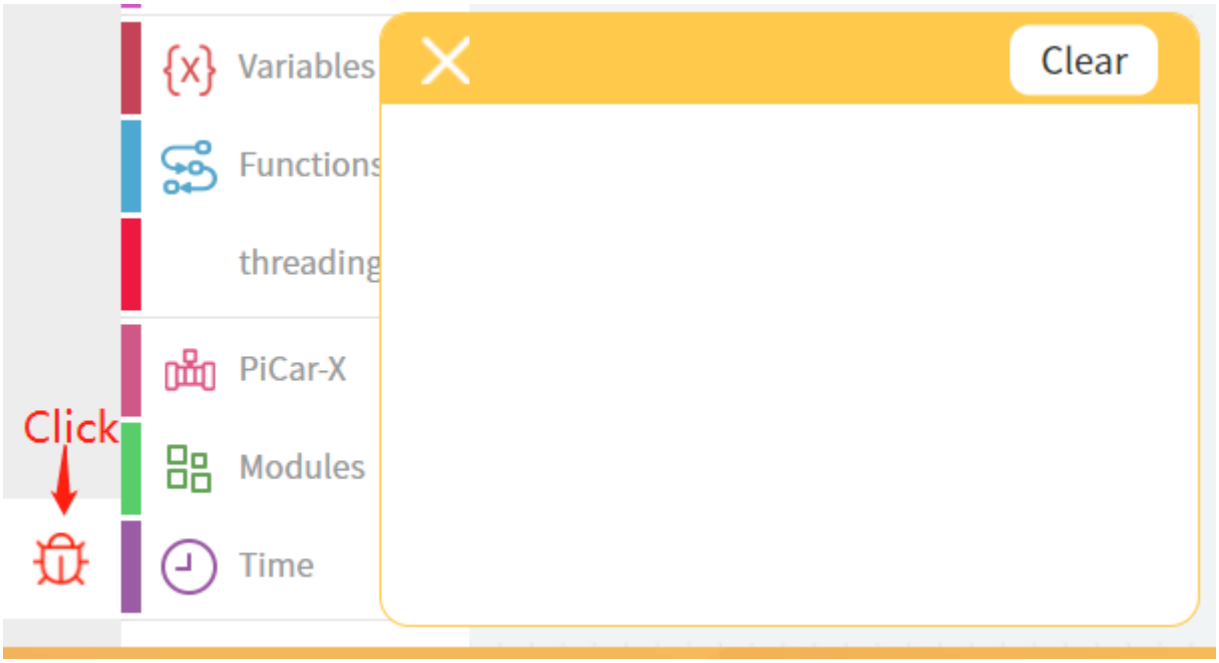
Ultrasonic get distance ブロックは、PiCar-X から前方の障害物までの距離を読み取ります。



このプログラムは **変数** の使い方を練習しましょう。たとえば、プログラム内に障害物までの距離を使用する必要がある複数の関数がある場合、 **変数** を使用して各関数が距離を個別に読み取る代わりに一度計測したデータを各関数で同じ値を利用できます。

Variables カテゴリの **Create variable...** ボタンをクリックし、名前の欄に例えば「 **distance** 」という変数の名前を入力します。

Print 関数はデバッグを容易にするために変数やテキストなどのデータを画面に表示できます。

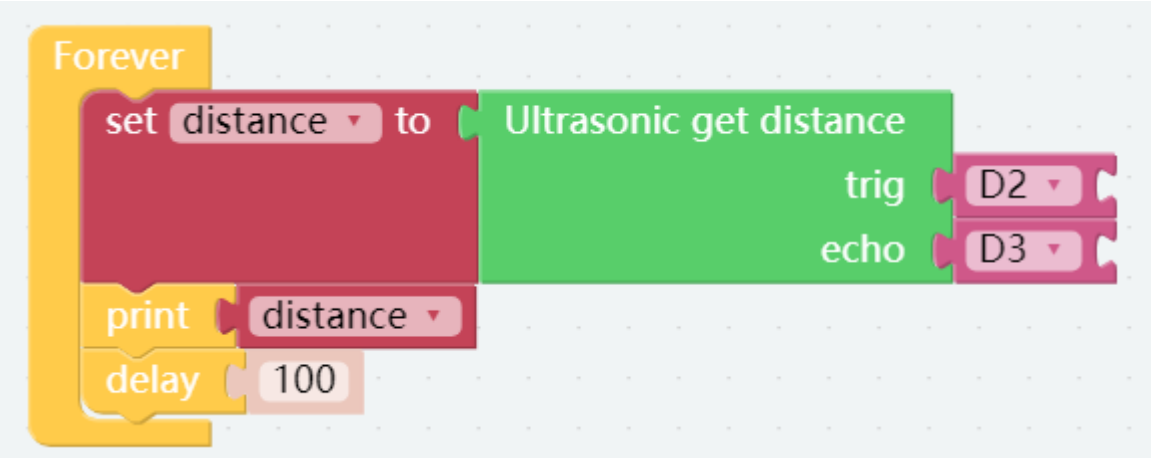


コード（プログラム）が実行されたら、左下隅にある **デバッグ** アイコンをクリックして、デバッグ モニターを有効にします。

例：

注釈:

- 次の例を参考にしてプログラムを作成してください。またチュートリアルを参照してください。How to Create a New Project?.
- または EzBlock Studio の **Examples** 画面から「 **Ultrasonic Sencor Test** 」を探し **Run** か **Edit** を直接クリックしてください。

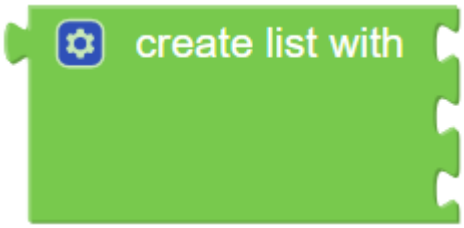


5.7 グレースケール・モジュール・テスト

PiCar-X にはライン追跡、崖検出、およびその他の楽しい実験を実装するためのグレースケール・モジュールが付いています。 グレースケール・モジュールには 3 つの検出センサーがあり、それぞれがセンサーによって検出された色の濃淡に従った値を返します。 たとえば純粋な黒を読み取るセンサーは値として「0」を返します。

ティップス

Grayscale module ブロックを使用してセンサー毎の値を読み取ります。 上記の例で「A0」センサーは PiCar-X の左端にあるセンサーです。 ドロップダウン矢印を使用してセンサーを「A1」(中央センサー) または「A2」(一番右のセンサー) に変更できます。

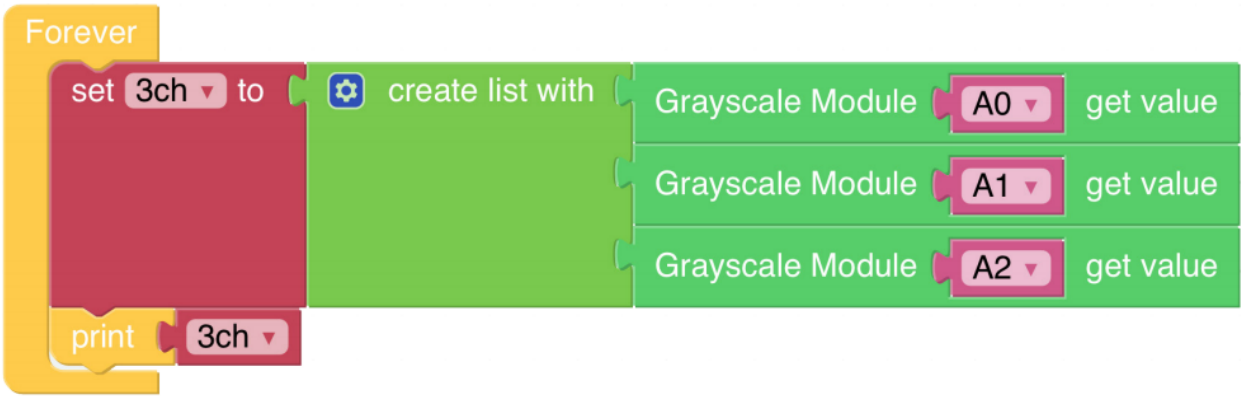


このプログラムは、 **create list with** ブロックによって簡略化されています。 **リスト** は単一の **変数** と同じように使用されるが、この場合、グレースケールモジュールは複数のセンサー値を報告するため、単一の **変数** よりもリストが効果的である。 **create list with** ブロックは、各センサーに対して個別の **変数** を作成し、リストに配置します。

例：

注釈:

- 次の例を参考にしてプログラムを作成してください。またチュートリアルを参照してください。：How to Create a New Project?.
- または EzBlock Studio の **Examples** 画面から「 **Grayscale Sensor Test** 」を探し **Run** か **Edit** を直接クリックしてください。



5.8 色検出

PiCar-X はカメラを内蔵していて Ezblock プログラムで物体検出と色認識コードを利用できます。 このセクションでは Ezblock を使用して色検出のプログラムを作成します。

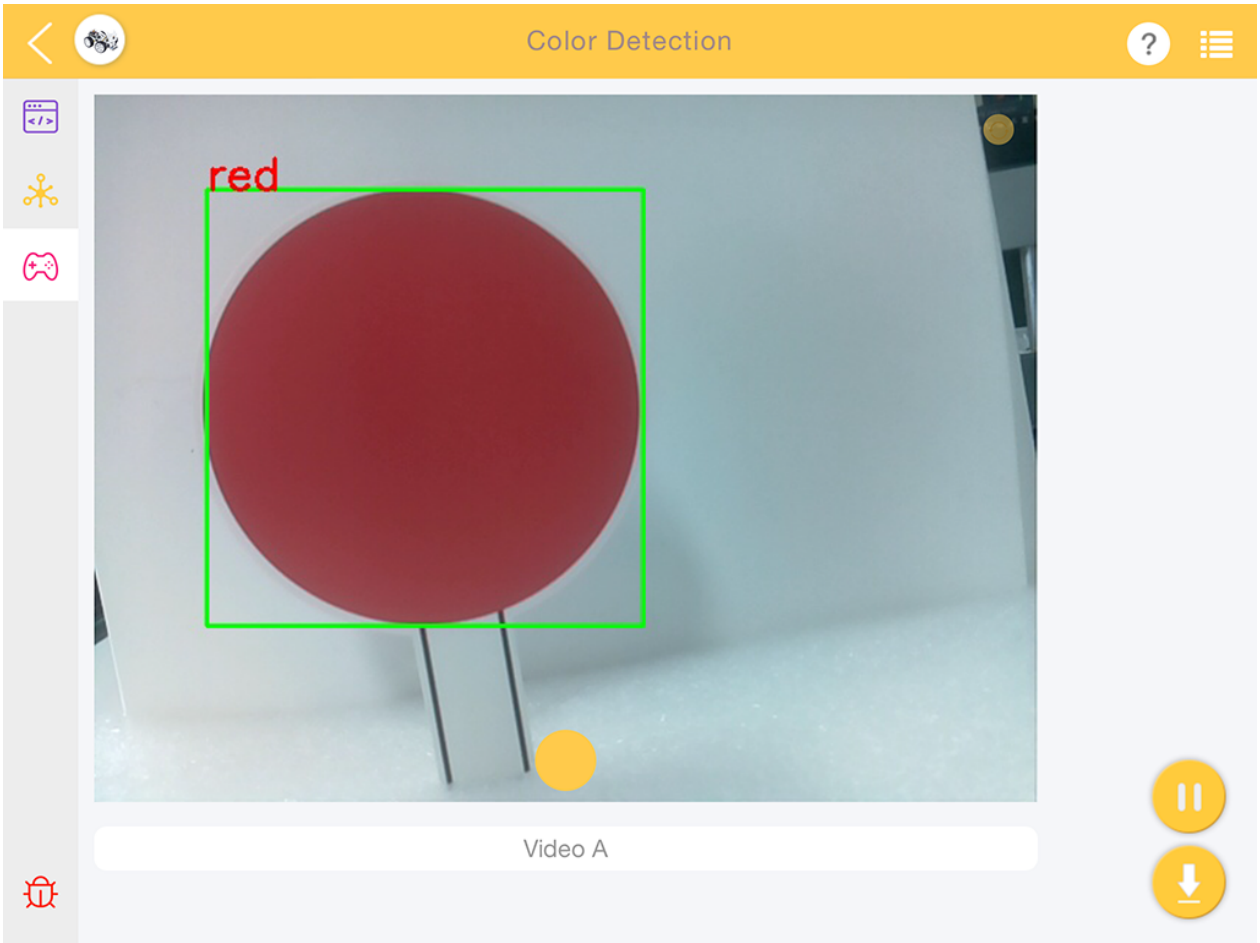
注釈: このセクションを試す前に Raspberry Pi カメラの FFC ケーブルが正しくしっかりと接続されていることを確認してください。 FCC ケーブルを確実に接続するための詳細な手順については、**部品リストと組み立て説明書** を参照してください。

このプログラムでは、Ezblock 最初に検出される色（の色相彩度値 (HSV) 空間範囲）を指定し、次に OpenCV を使用して（HSV 範囲内の）色を処理してバックグラウンド ノイズを除去し、最後に一致した色に関する情報をまとめます。

Ezblock には、PiCar-X 用の「赤」「オレンジ」「黄」「緑」「青」「紫」の 6 色モデルがあります。 次の PDF にカラーカードが用意されており、カラー プリンターで印刷する必要があります。

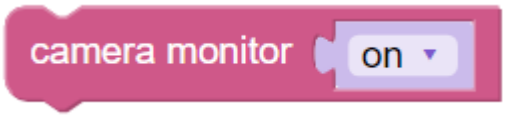
- `[PDF]Color Cards`

注釈: 印刷された色は、プリンターのトナーの違い、または印刷用紙のわずかな色の違いなどによって、Ezblock カラー モデルとはわずかに異なる色合いになる場合があります。 これにより、色認識の精度が低下する可能性があります。

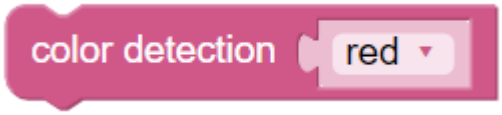


ティップス

Remote Control のページで。ビデオのモニター画面をドラッグしてリモコン画面に設置します。 ビデオ・モニターの使用方法の詳細については、こちらの Ezblock ビデオに関するチュートリアルを参照してください: How to Use the Video Function?。



camera monitor のブロックを **on** にすると **Remote Control** のページのモニター画面にカメラで捉えている映像を表示できます。 注: **camera monitor** を **off** にするとモニター画面への表示は行われませんが、色検出機能はそのまま動作しています。

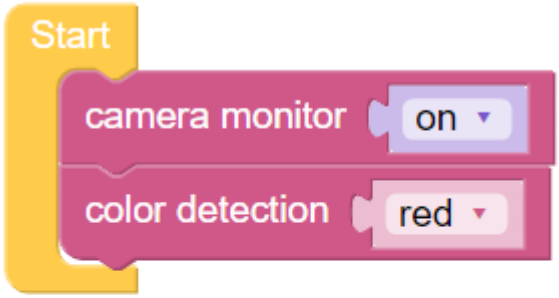


color detection ブロックを使用して、色検出を有効にします。 注: 一度に検出できる（設定できる）色は 1 つだけです。

例：

注釈:

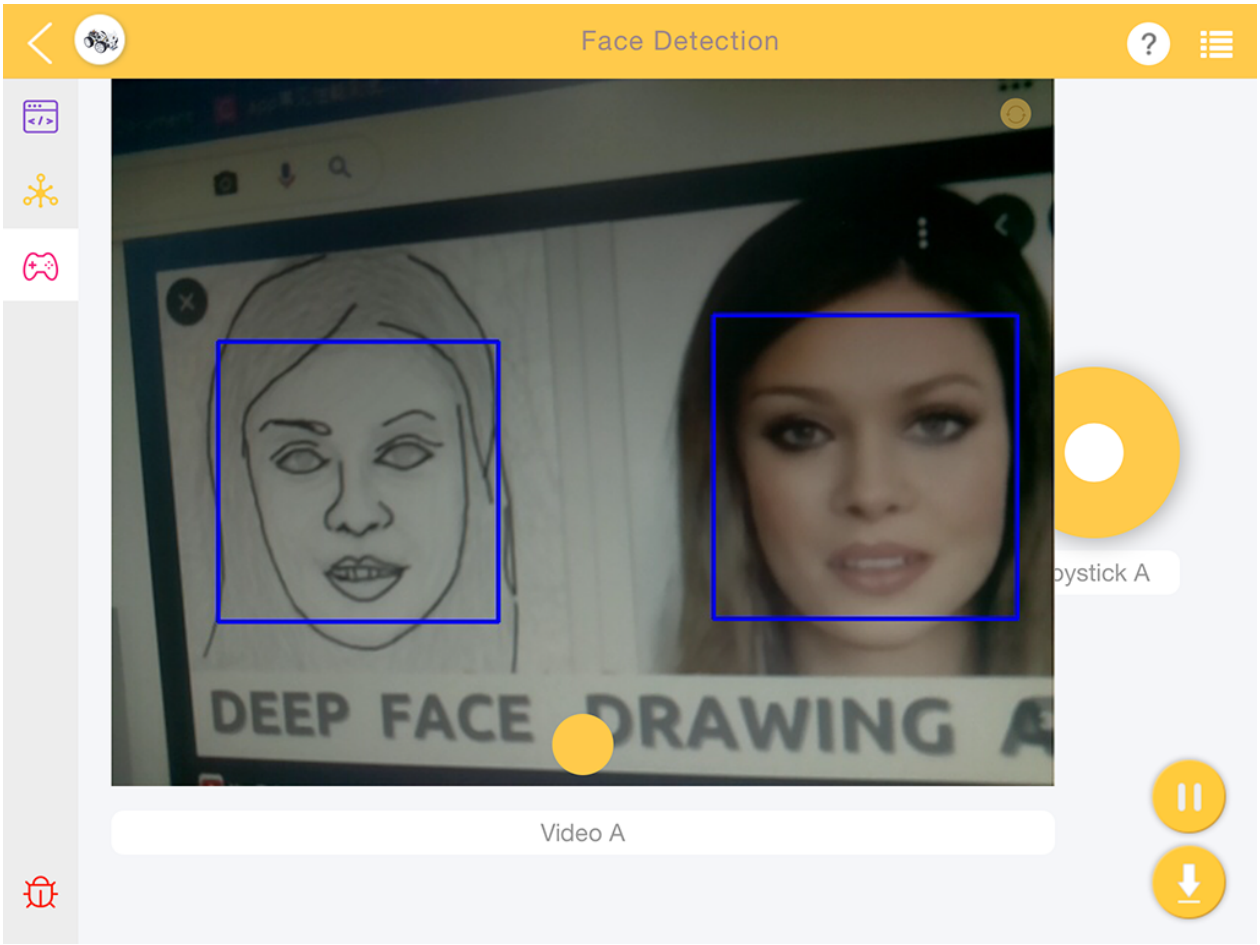
- 次の例を参考にしてプログラムを作成してください。またチュートリアルを参照してください。How to Create a New Project?.
- または EzBlock Studio の **Examples** 画面から「 **Color Detection** 」を探し **Run** か **Edit** を直接クリックしてください。



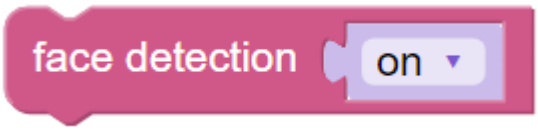
5.9 顔検出

PiCar-X には、色検出に加えて顔検出機能も搭載されています。 次の例では、リモートコントロール画面のジョイスティックを使用してカメラの方向を変え、カメラが捉えた人の顔の数をデバッグ・モニターに表示します。

ビデオ・モニターの使用方法の詳細については、こちらの Ezblock ビデオに関するチュートリアルを参照してください: How to Use the Video Function?。



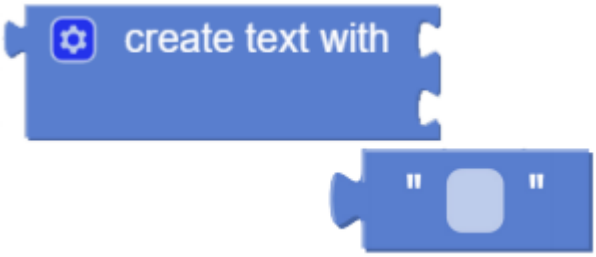
ティップス



face detection 機能を **on** に設定して、顔検出を有効にします。

以下の 2 つのブロックは、**リモコン・カー** のチュートリアルで PiCar-X のハンドル操作をするのと同様に、カメラのパン（左右）とチルト（上下）の向き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。

画像検出結果は **detected face** ブロックを介して提供されます。 ドロップダウンメニュー・オプションを使用して、画像検出関数からの結果の座標、サイズ、または数のいずれを読み出すかを選択します。

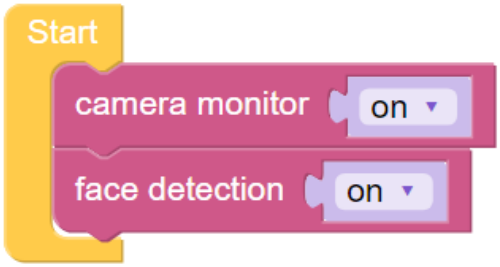


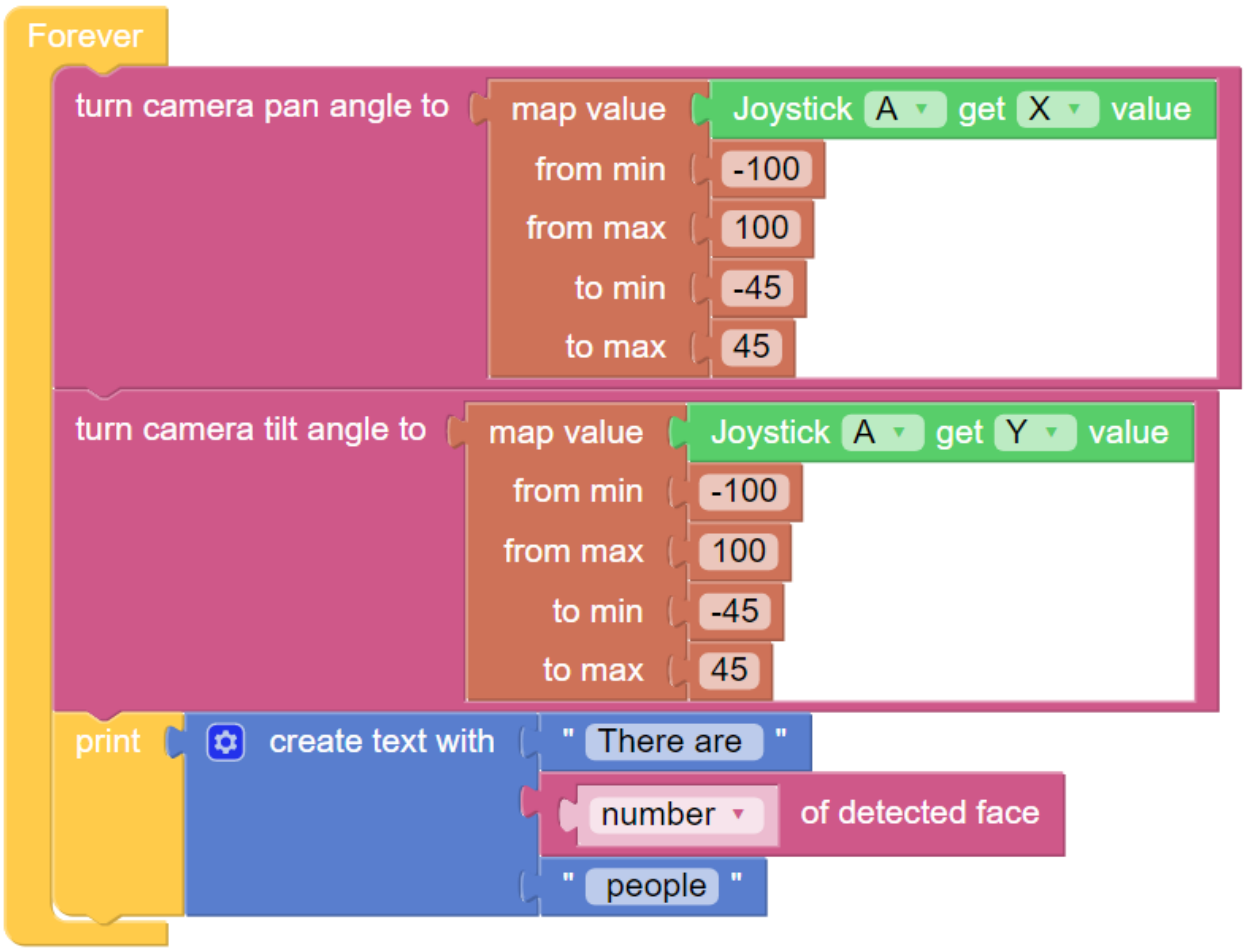
create text with ブロックを使用して、**テキスト** と **検出された顔** データの組み合わせを出力します。

例：

注釈:

- 次の例を参考にしてプログラムを作成してください。またチュートリアルを参照してください。How to Create a New Project?.
- または EzBlock Studio の **Examples** 画面から「 **Face Detection** 」を探し **Run** か **Edit** を直接クリックしてください。





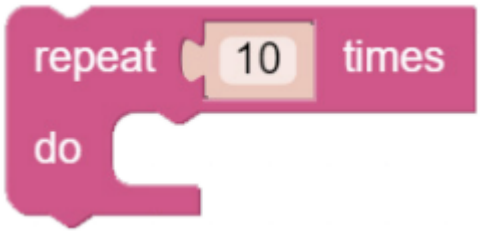
5.10 音声読み上げ

PiCar-X には、オーディオ実験に使用できるスピーカーが内蔵されています。 Ezblock を使用して文字を入力して PiCar-X に喋らせるたり特定の効果音を鳴らしたりできます。 このチュートリアルでは、PiCar-X は do/while 関数を使用して、3 秒のカウントダウン後に銃の発砲音を出します。

ティップス

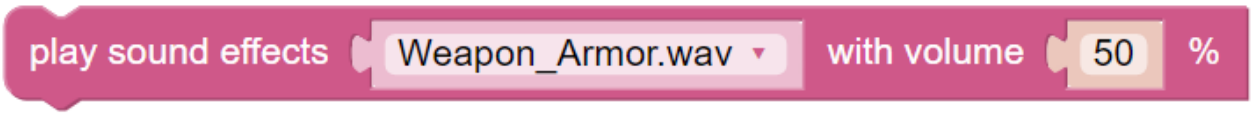
say ブロックと **text** ブロックを使用して、PiCar-X が言う文章を書きます。 **say** ブロックは文字または数字が使用できます。ただし英語になります。上手くローマ字で書くと日本語もそれなりに喋ります。

数字 ブロック



repeat ブロックを使用すると、同じブロックが繰り返し実行ためプログラムが簡素化できます。

数学演算 ブロックは、「+」、「-」、「x」、「÷」などの一般的な数学関数を実行できます。

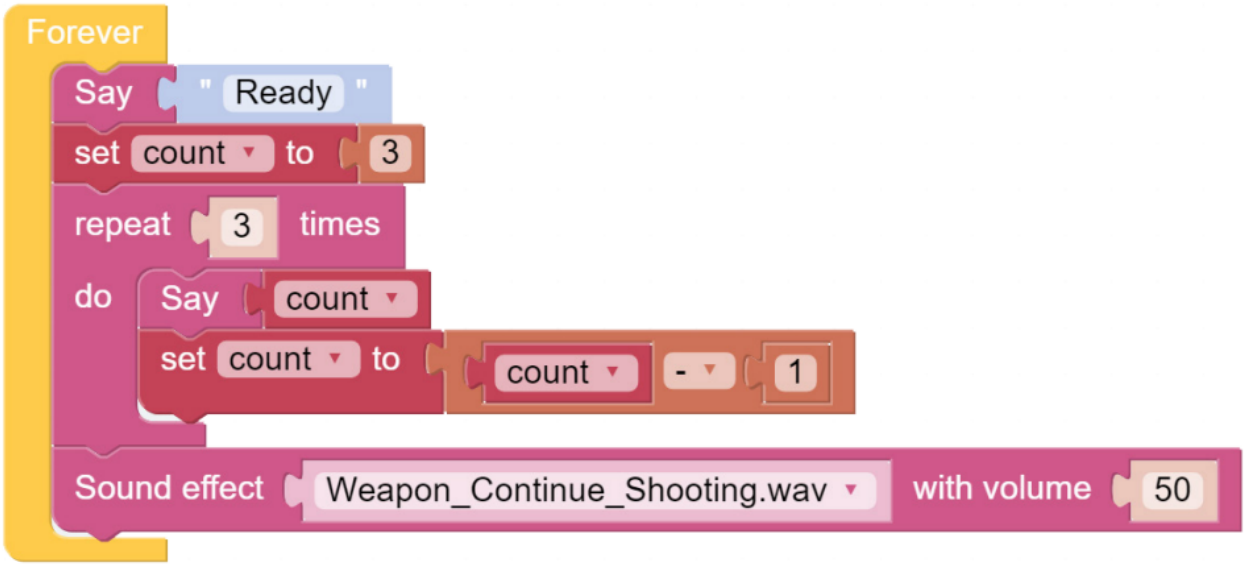


play sound Effects - with volume - % ブロックには、サイレンの音、銃の音などのサウンド効果があらかじめ設定されています。 音量の範囲は 0 から 100 まで設定できます。

例：

注釈:

- 次の例を参考にしてプログラムを作成してください。またチュートリアルを参照してください。How to Create a New Project?.
- または EzBlock Studio の **Examples** 画面から「 **Sound Effect** 」を探し **Run** か **Edit** を直接クリックしてください。



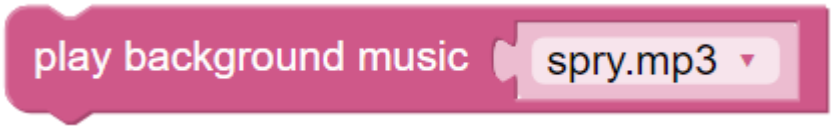
5.11 バックグラウンド・ミュージック

PiCar-X をプログラムして効果音やテキスト読み上げ (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、 **Slider** ウィジェットを使用して音楽の音量を調整出来ます。

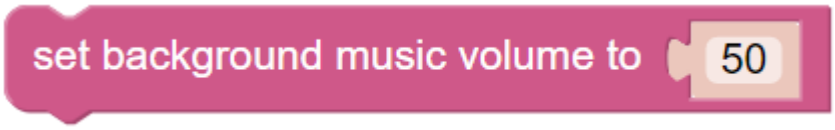
- How to Use the Remote Control Function?

Ezblocks リモート コントロール機能の詳細については、**リモコン・カー** チュートリアルを参照してください。

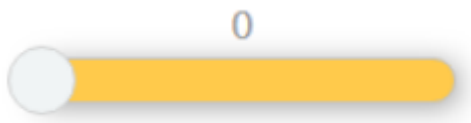
ティップス



play background music のブロックは **Start** ファンクションの中にセットしなければいけません。ドロップダウンメニューから PiCar-X が演奏する音楽を選択できます。実際に演奏が開始されるのは **set background music volume to** のブロックか **slider [A] get value** のブロックで音量が指定された後からになります。



set background music volume to のブロックで演奏する音楽の音量を決定します。 **Start** ファンクションの中でもその他の場所でもどちらにでもセットできます。設定できる値は０から１００までとなります。



Remote Control のページでこのスライドバーをドラッグしてリモコン画面に設置します。演奏している音楽の音量を自由に変更できます。

slider [A] get value ブロックは **Remote Control** のページで設定した **Slider** バーから演奏する音量を設定できます。 **Start** ファンクション以外のところで使用できます。ここで音量を調整して初めて音楽の演奏が始まります。

例：

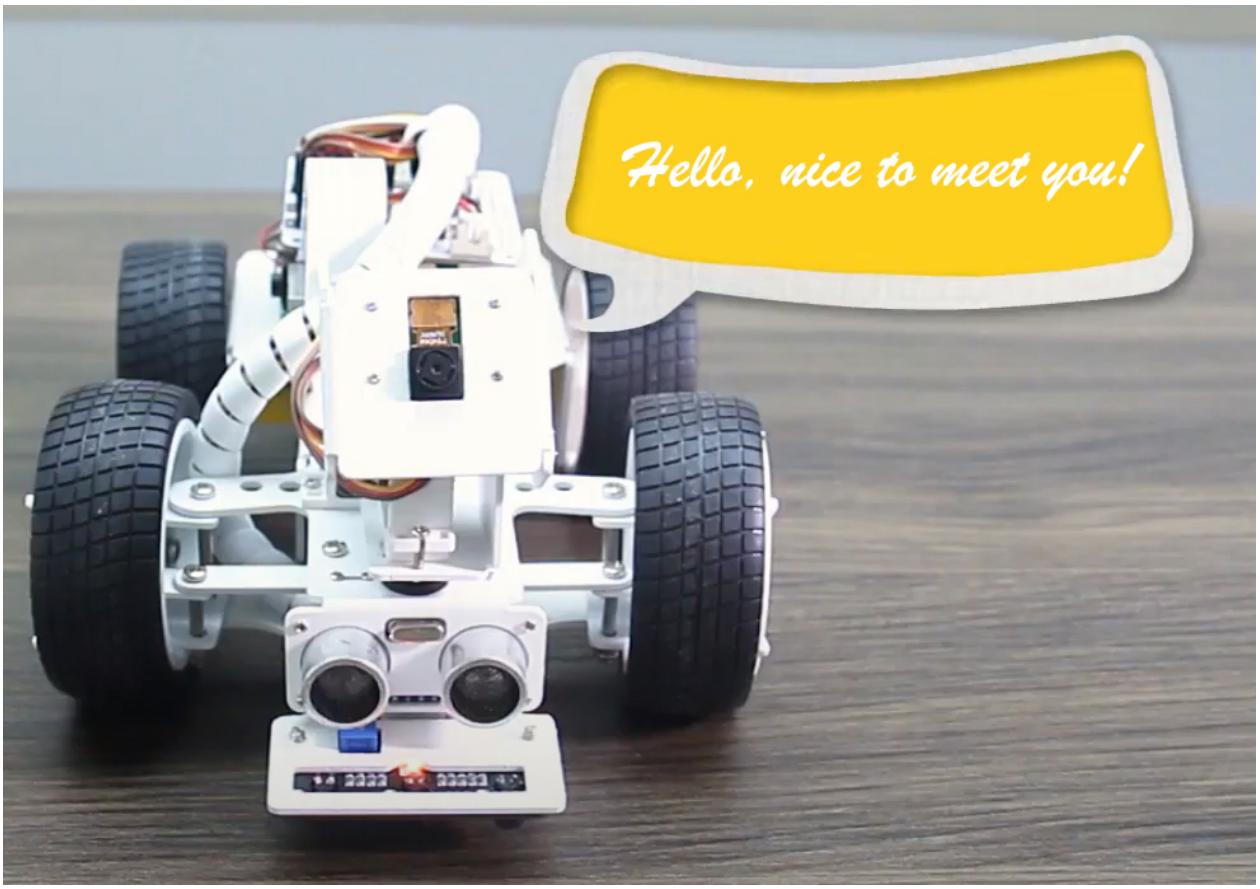
注釈:

- 以下の例を参考にしてプログラムしてください。また次のチュートリアルをご参照ください。How to Create a New Project?.
- または EzBlock Studio の **Examples** 画面から「 **Background Music** 」 を探し **Run** か **Edit** を直接クリックしてください。

5.12 ハロー！

このプロジェクトは、前のプロジェクトのいくつかの機能を組み合わせます。 PiCar-X の動きはリモコン操作され、カメラもジョイスティックを使用して向きを変えることができます。 PiCar-X が誰かの顔を見つけると丁寧にうなずいてから「ハロー!」と言います。

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



ティップス

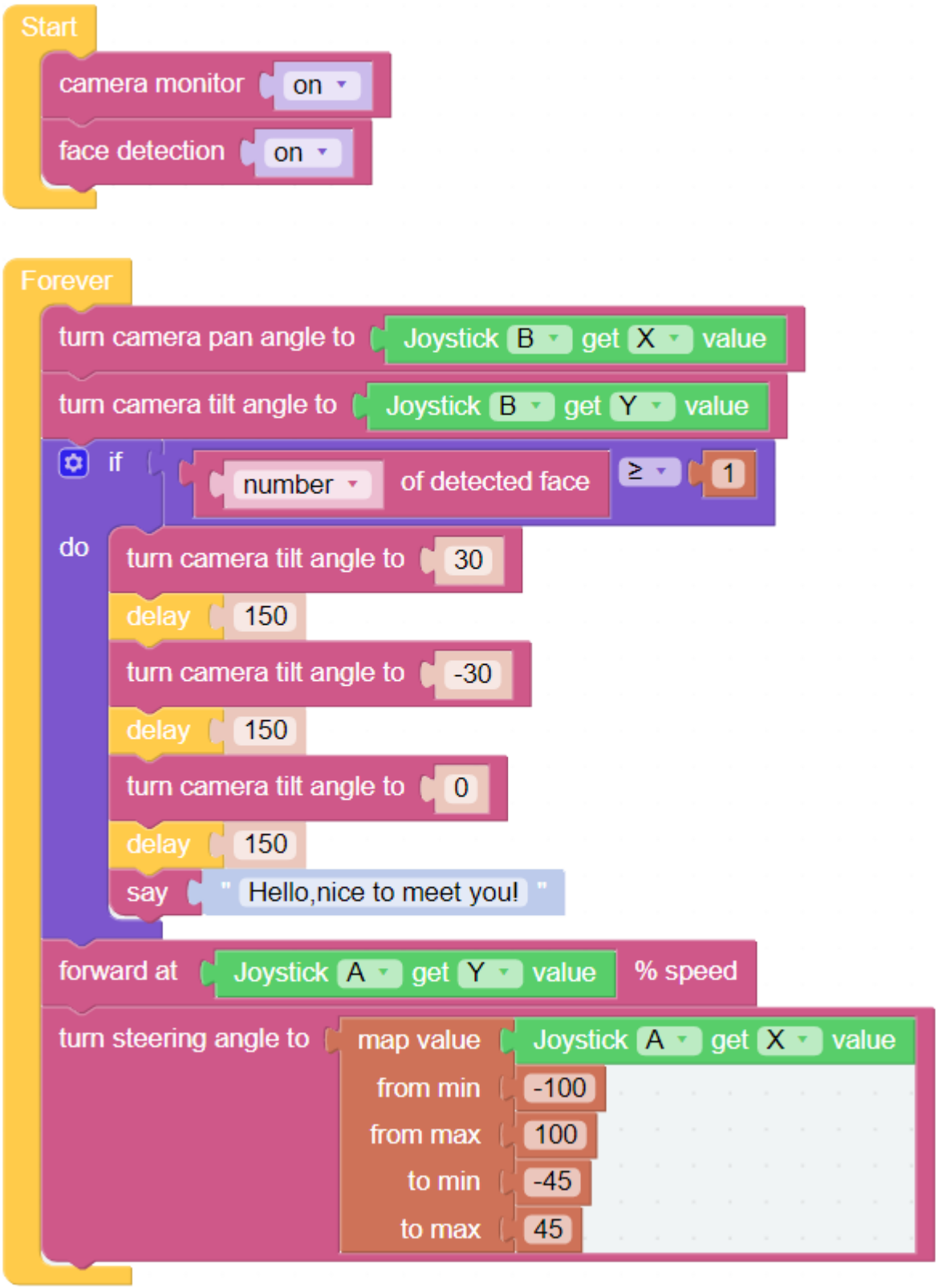
if do ブロックは、「if」の条件付き判断が true（正） になると丁寧にうなずくために使用されます。

条件ステートメント 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、「=」、「>」、「<」、「」、「」、「≠」のいずれかです。

例：

注釈:

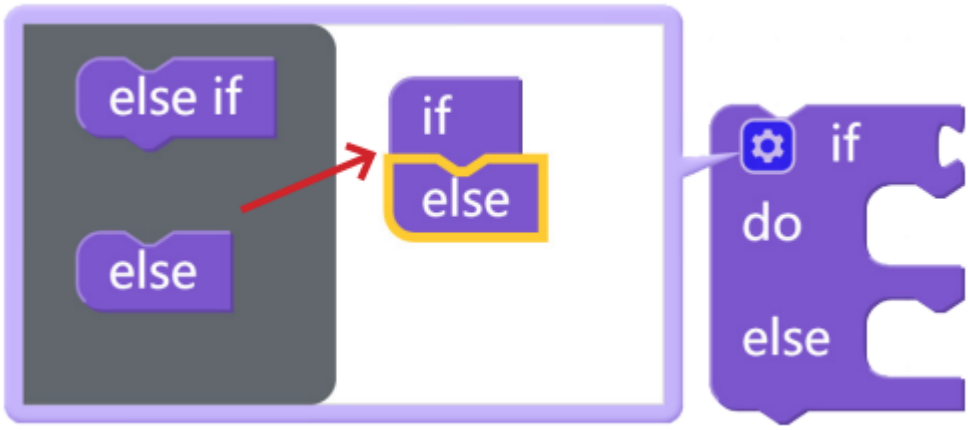
- 次の例を参考にしてプログラムを作成してください。またチュートリアルを参照してください。How to Create a New Project?.
- または EzBlock Studio の **Examples** 画面から「 **Say Hello** 」を探し **Run** か **Edit** を直接クリックしてください。


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

5.13 ミュージック・カー

このプロジェクトでは PiCar-X をミュージック・カーに変えて、家の中を楽しい音楽を奏でて回ります。 このプロジェクトでは、PiCar-X が内蔵の超音波センサーで距離を測って壁にぶつかるのを回避する方法も示します。

TIPS

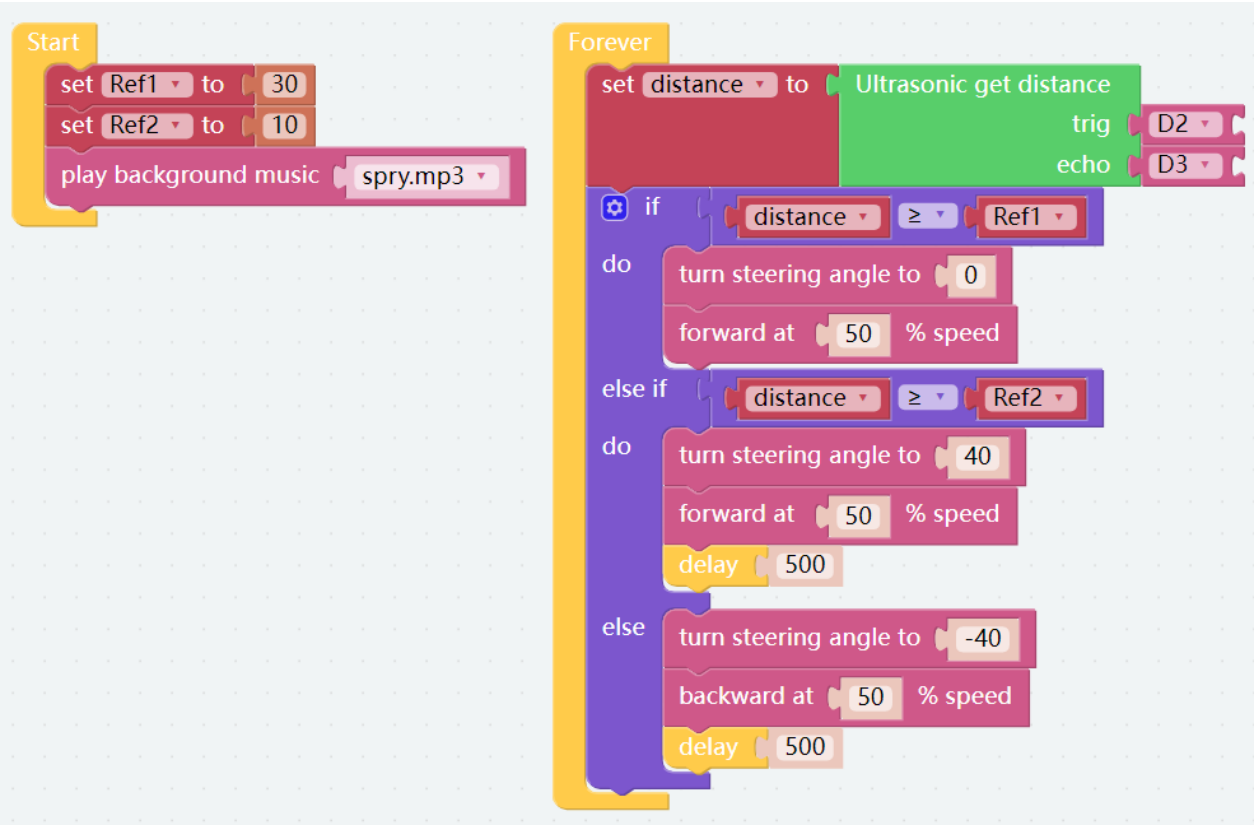


複数の条件判定を実装するには、単純な if do ブロックを if else do / else if do ブロックに変更します。 これは、上に示すように設定アイコン（ギアマーク）をクリックして行います。

例：

注釈:

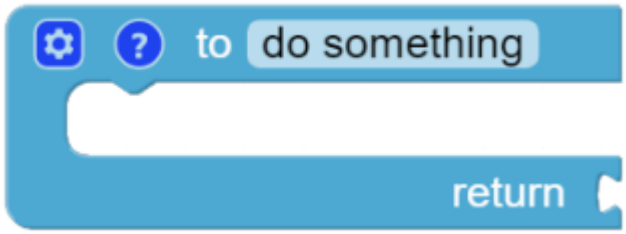
- 次の例を参考にしてプログラムを作成してください。またチュートリアルを参照してください。How to Create a New Project?.
- または EzBlock Studio の **Examples** 画面から「 **Music Car** 」を探し **Run** か **Edit** を直接クリックしてください。



5.14 崖検知

このプロジェクトでは **グレースケール モジュール** を使用して家の中を自由に動き回っている PiCar-X が崖から落ちるのを防ぎます。 階段のある家には欠かせないプロジェクトです。

ティップス

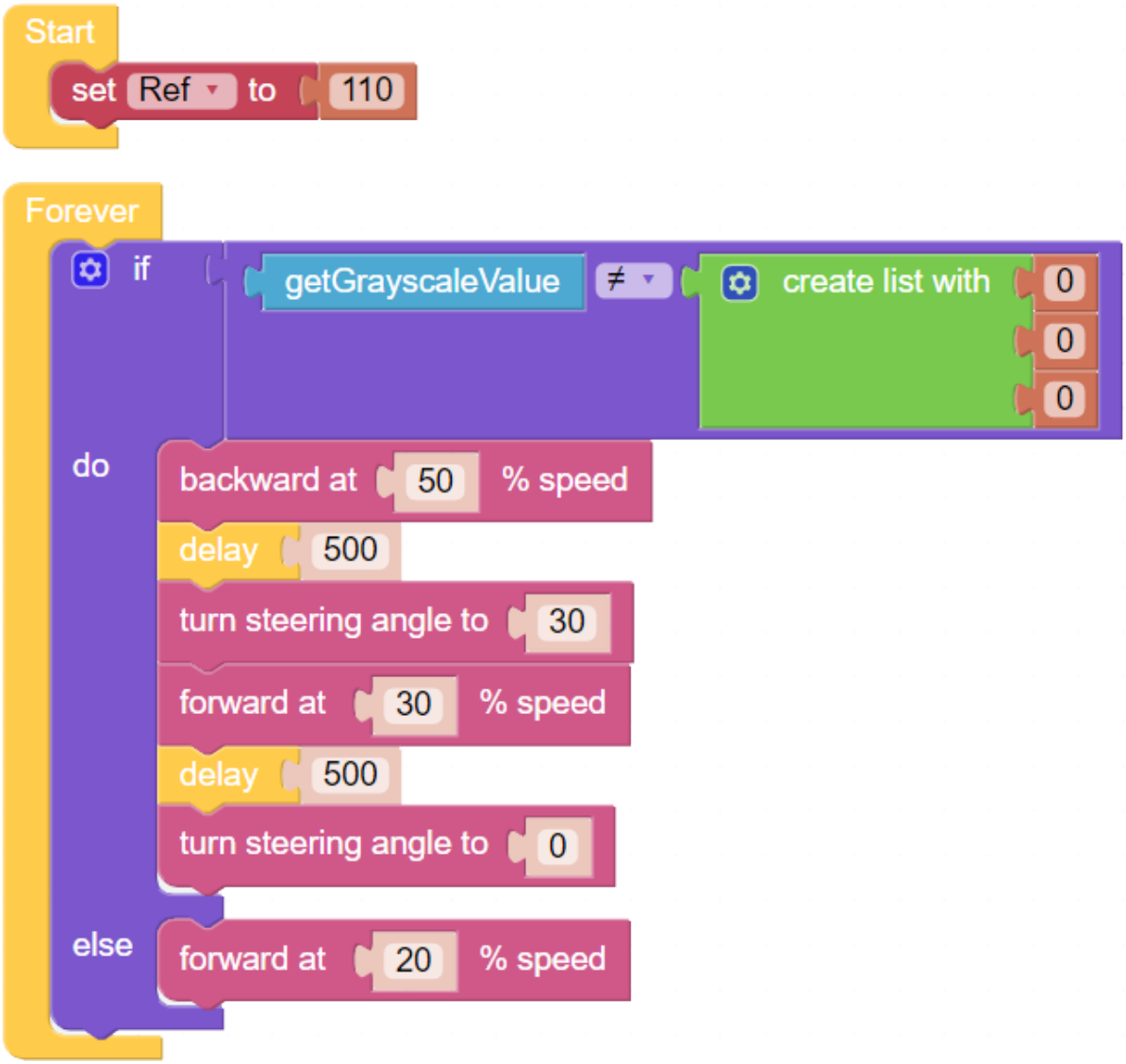


グレースケール モジュール は 3 個セットで付いているので同じ操作を複数回実行します。 またプログラムを簡素化するために、このプロジェクトでは **list** 変数を **do forever** ブロックに返す **関数** を導入しています。

例：

注釈:

- 次の例を参考にしてプログラムを作成してください。またチュートリアルを参照してください。How to Create a New Project?.
- または EzBlock Studio の **Examples** 画面から「 **Clife Detection** 」を探し **Run** か **Edit** を直接クリックしてください。



5.15 トロッコ

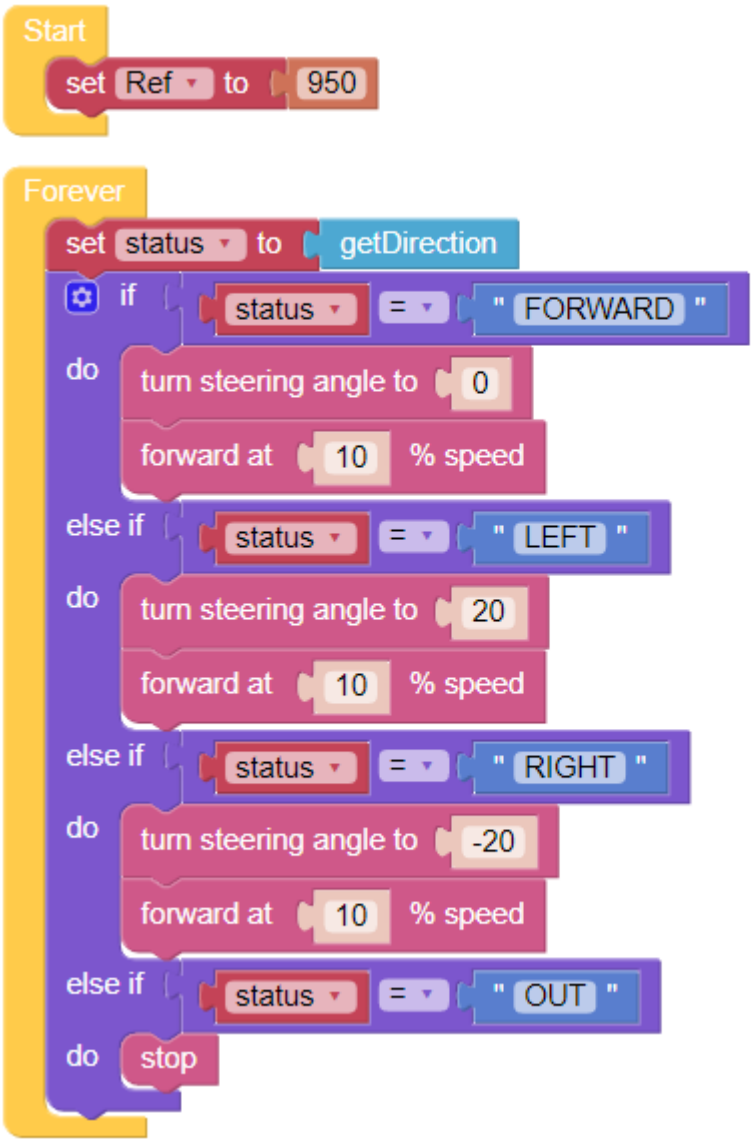
トロッコプロジェクトを作ろう！ このプロジェクトでは、グレースケール・モジュールを使用して、PiCar-X をトラック（：線路：黒い線）に沿って前進させます。暗い色のテープや白い模造紙に黒いマジックなどを利用してトラックを作ります。黒のビニールテープは艶があってあまり黒いとはみなされないので注意が必要です。トラックの曲線は曲げすぎないようにします。 PiCar-X が脱線した場合はカーブの具合など色々工夫してみてください。

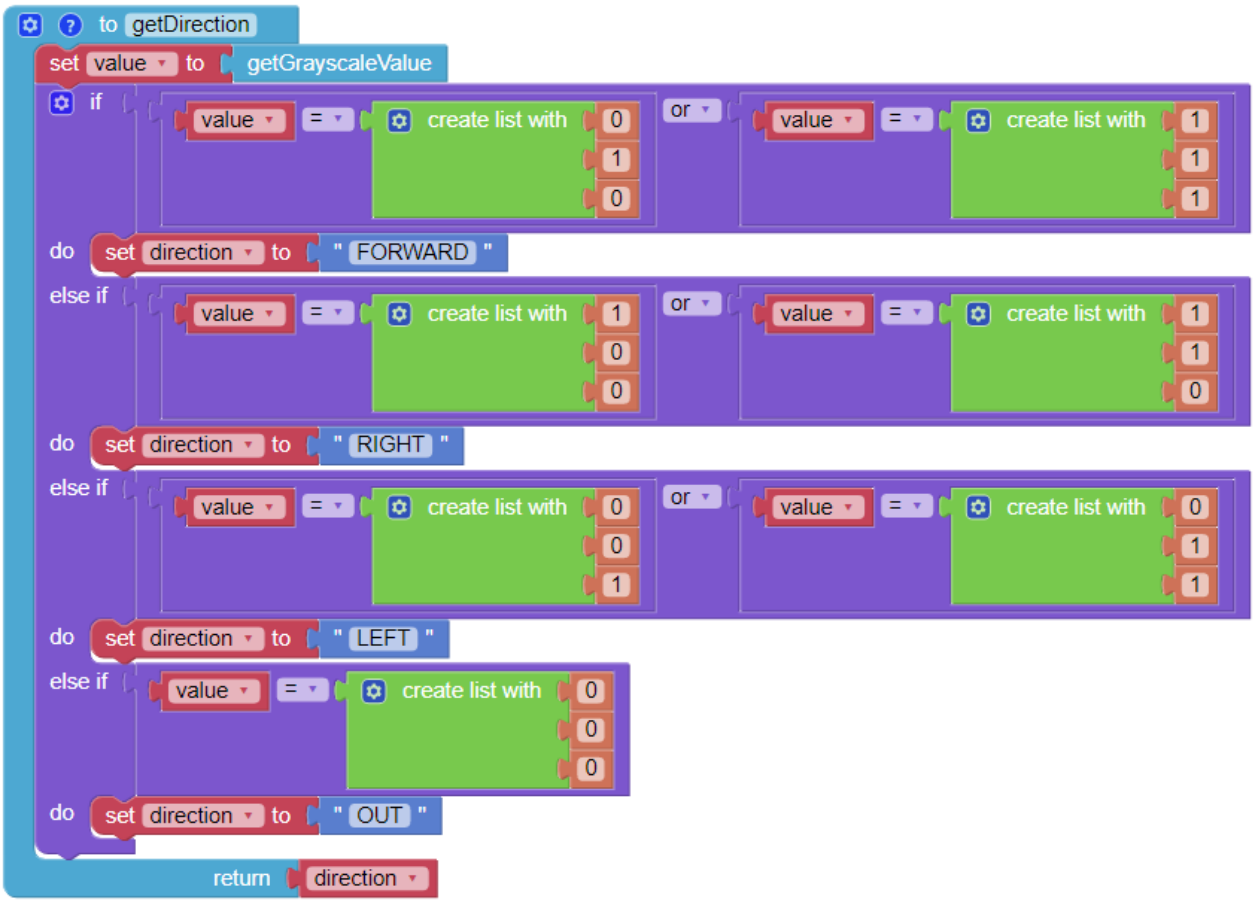
トラックに沿って前進するとグレースケール・モジュールの左右のセンサーが明るい色の地面を検出し、中央のセンサーが黒いトラックを検出します。トラックにカーブがある場合、センサーの左側または右側にあるセンサーが暗い色のテープを検出し、その方向に前輪を向けます。トロッコが線路の端に到達するか脱線すると、グレースケール・モジュールは黒色の線路を検出できなくなり、PiCar-X は停止します。

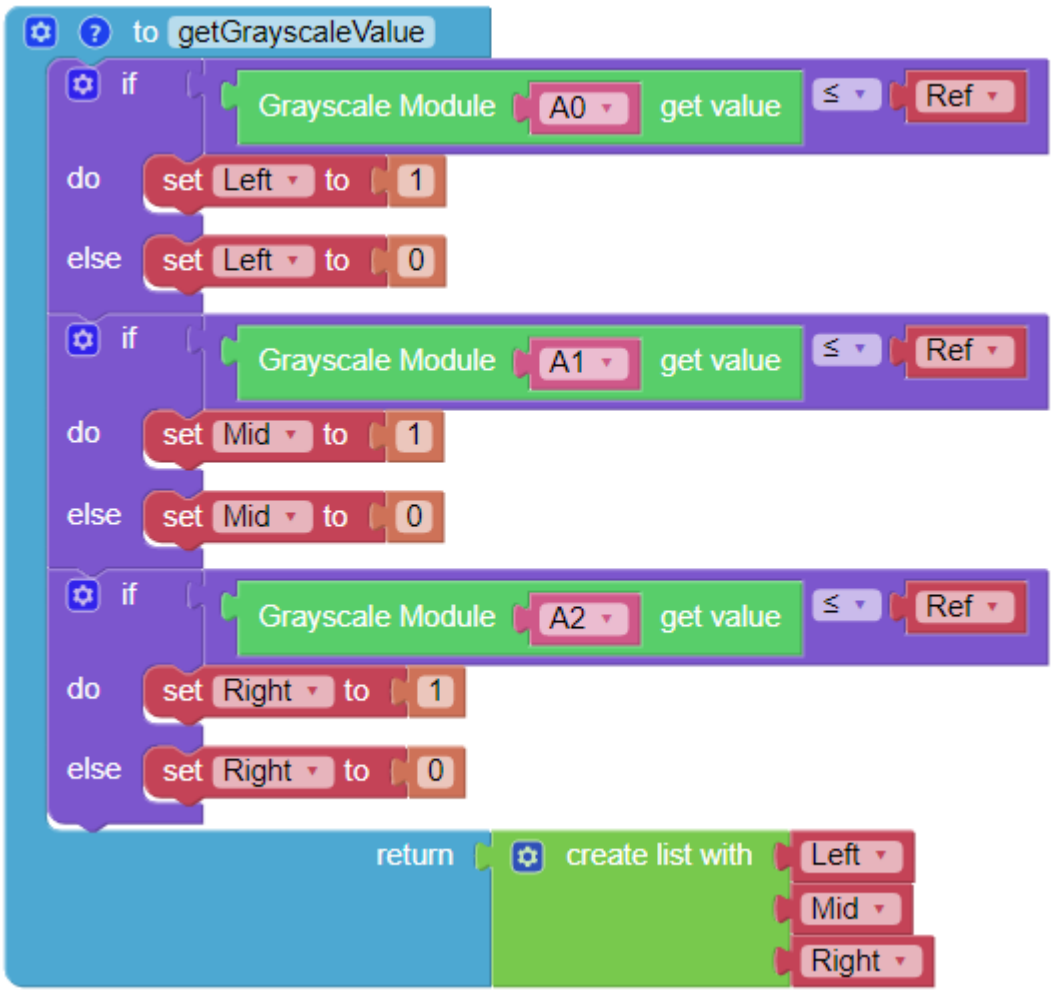
例：

注釈:

- 次の例を参考にしてプログラムを作成してください。またチュートリアルを参照してください。How to Create a New Project?.
- または EzBlock Studio の **Examples** 画面から「 **Minecart** 」を探し **Run** か **Edit** を直接クリックしてください。

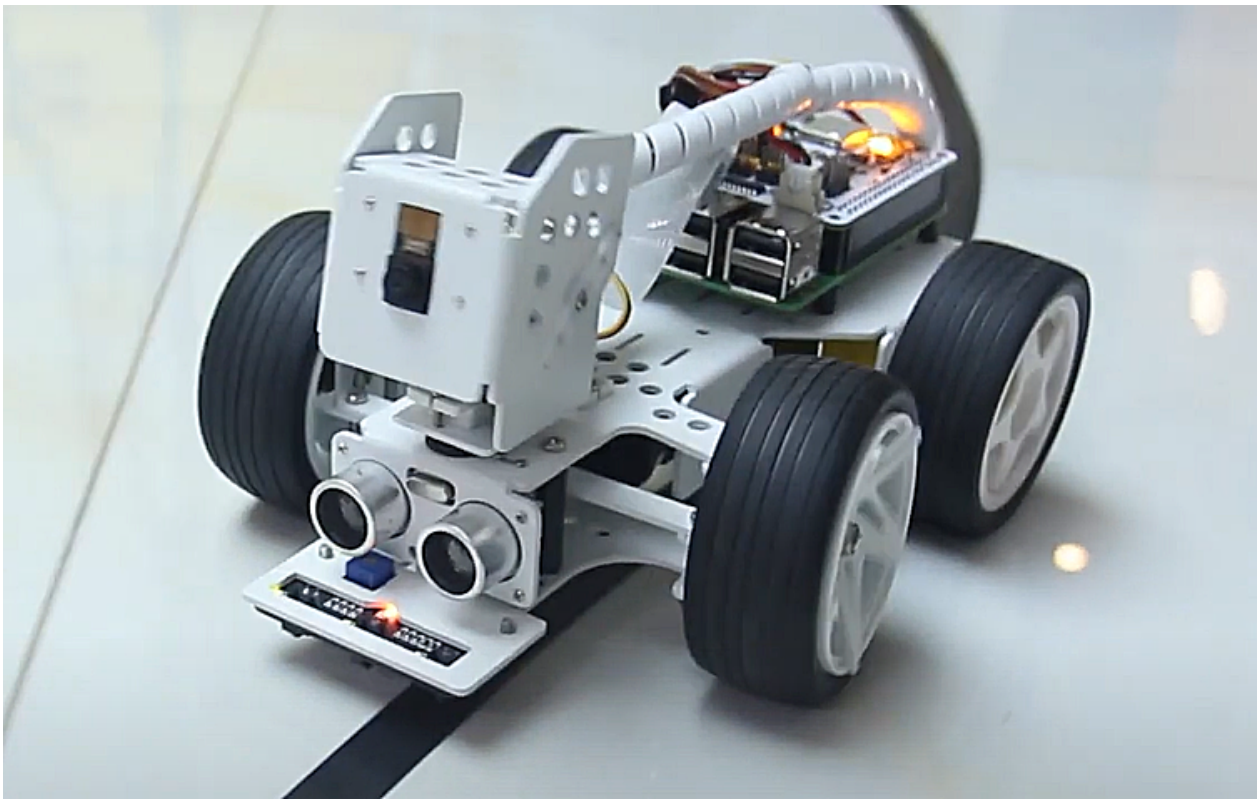



to getDirection
set value to getGrayscaleValue
if value = create list with 0 1 0 or value = create list with 1 1 1
do set direction to " FORWARD "
else if value = create list with 1 0 0 or value = create list with 1 1 0
do set direction to " RIGHT "
else if value = create list with 0 0 1 or value = create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

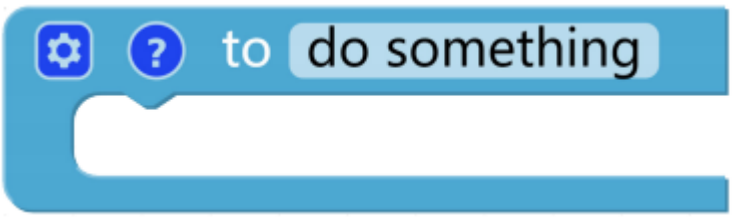


5.16 トロッコ+

このプロジェクトでは、脱線復旧機能が **トロッコ** に追加され、PiCar-X が進化してより厳しいカーブでも追従できるようになりました。



ティップス

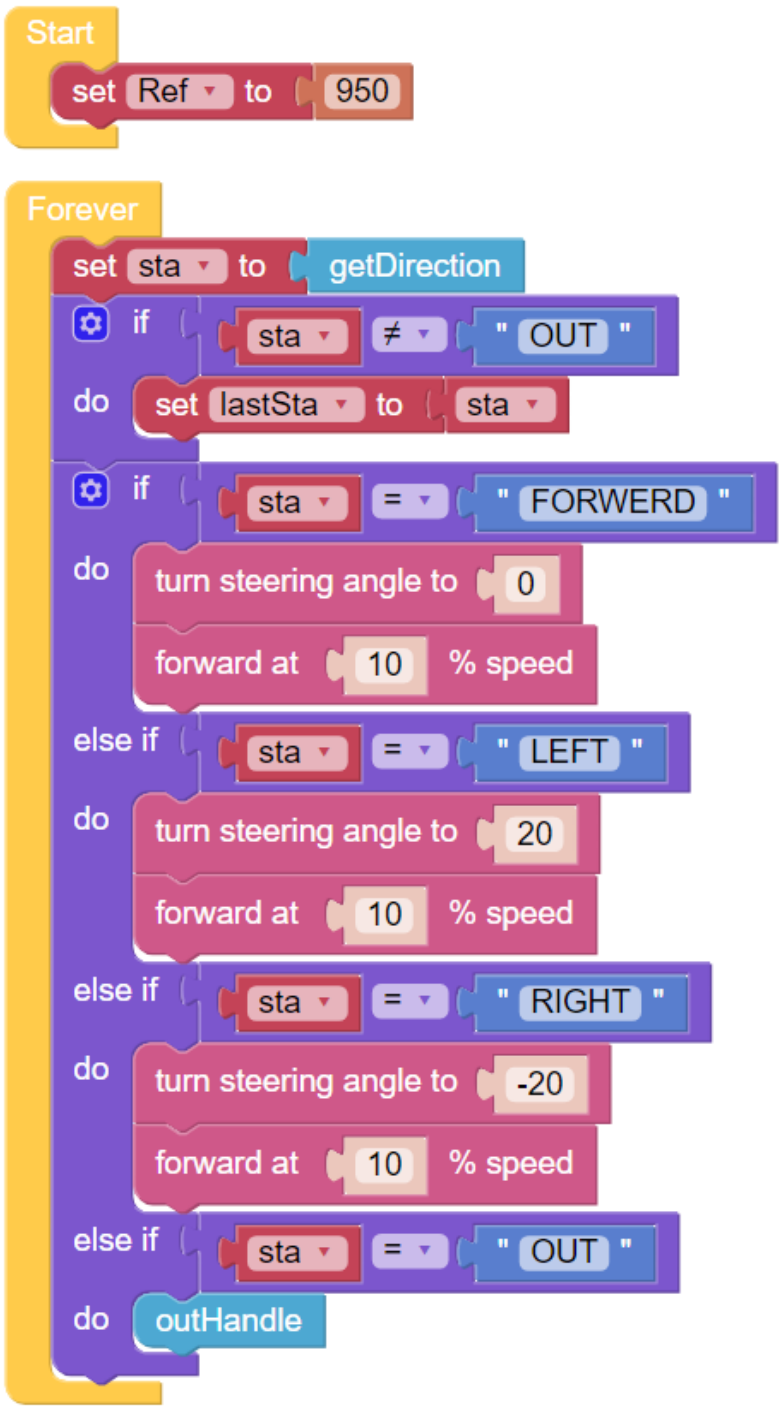


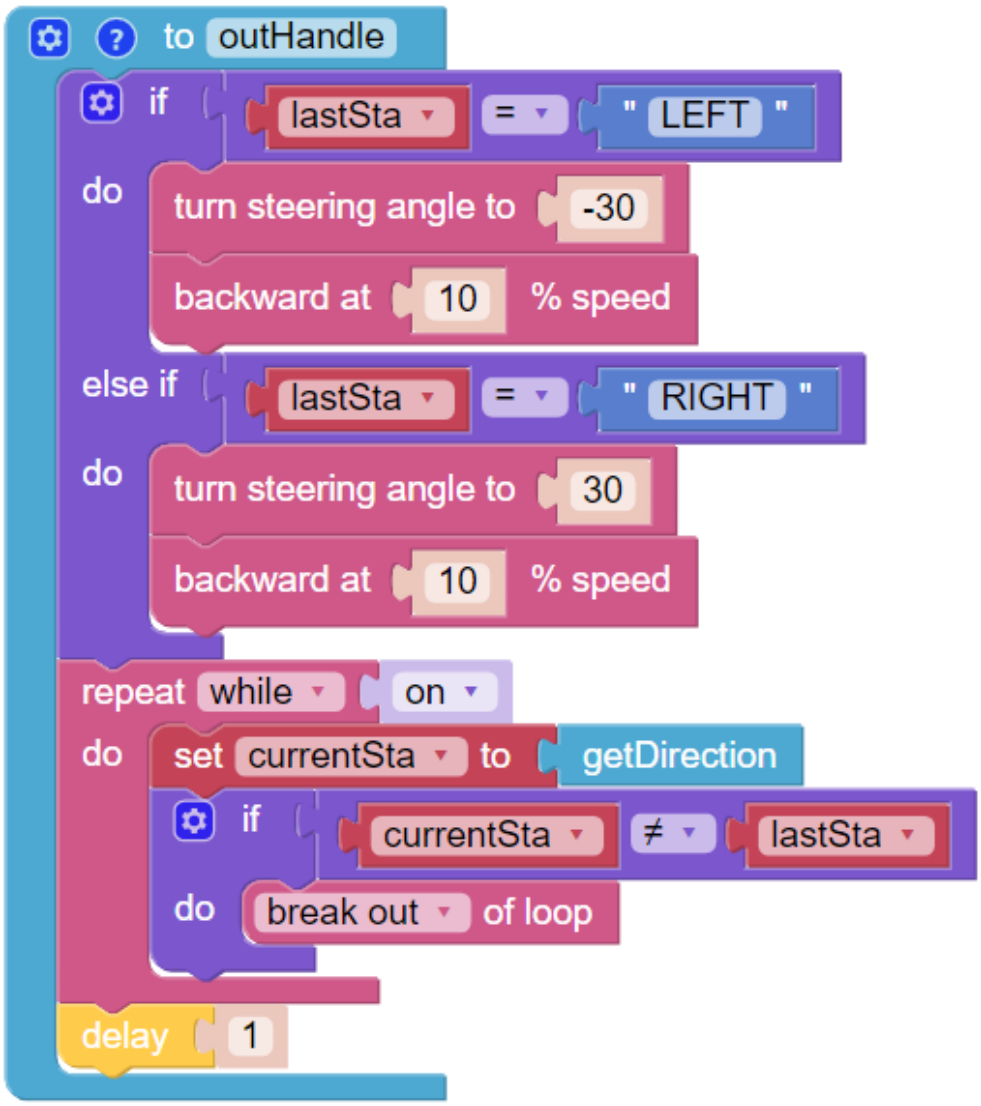
新たに **to do something** ブロックを追加してラインを逸脱しかけた時は PiCar-X が後退し、急なカーブでもラインを逸脱せずに追従できるようにします。新しい **to do something** 関数は値を返さず、PiCar-X の向きを変えるためだけに使用されることに注意してください。

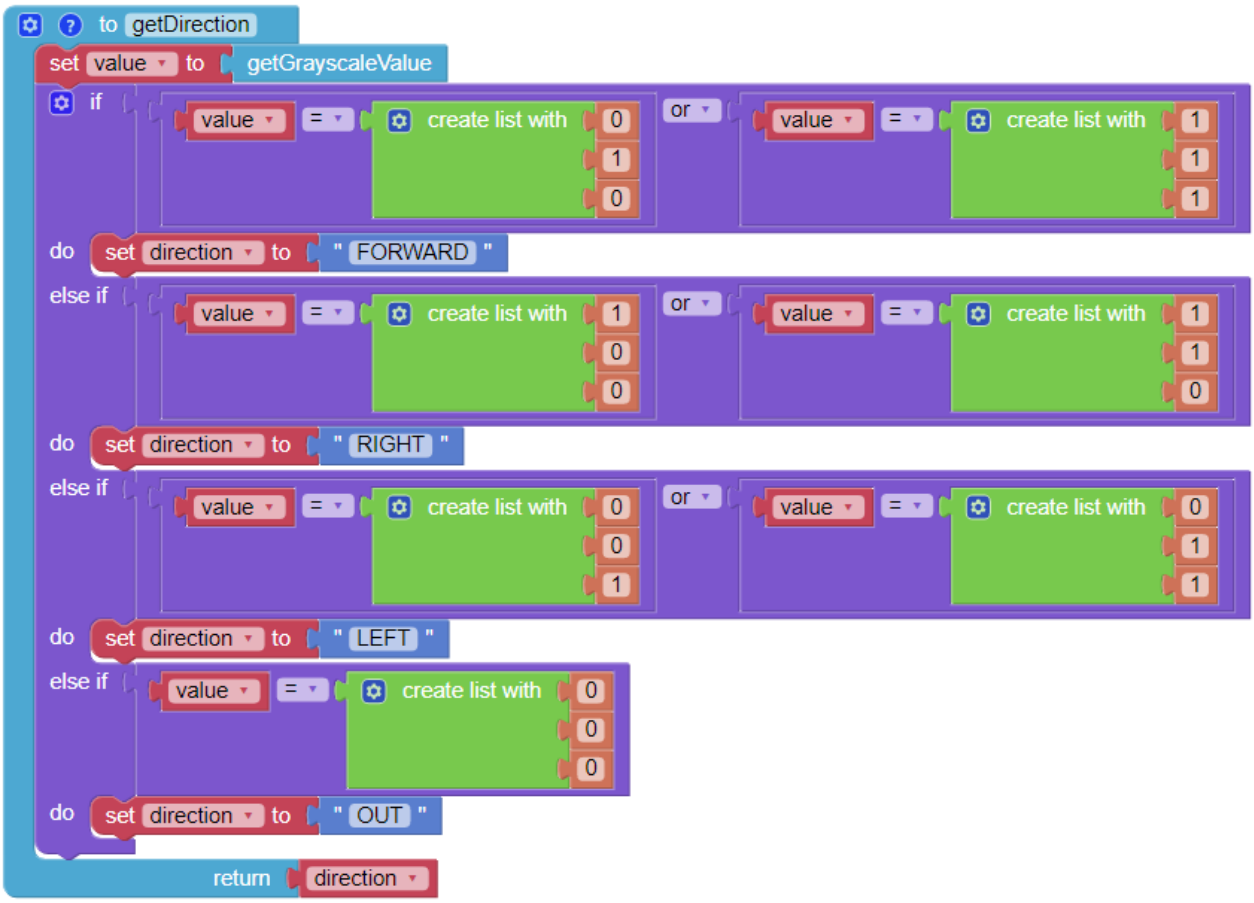
例：

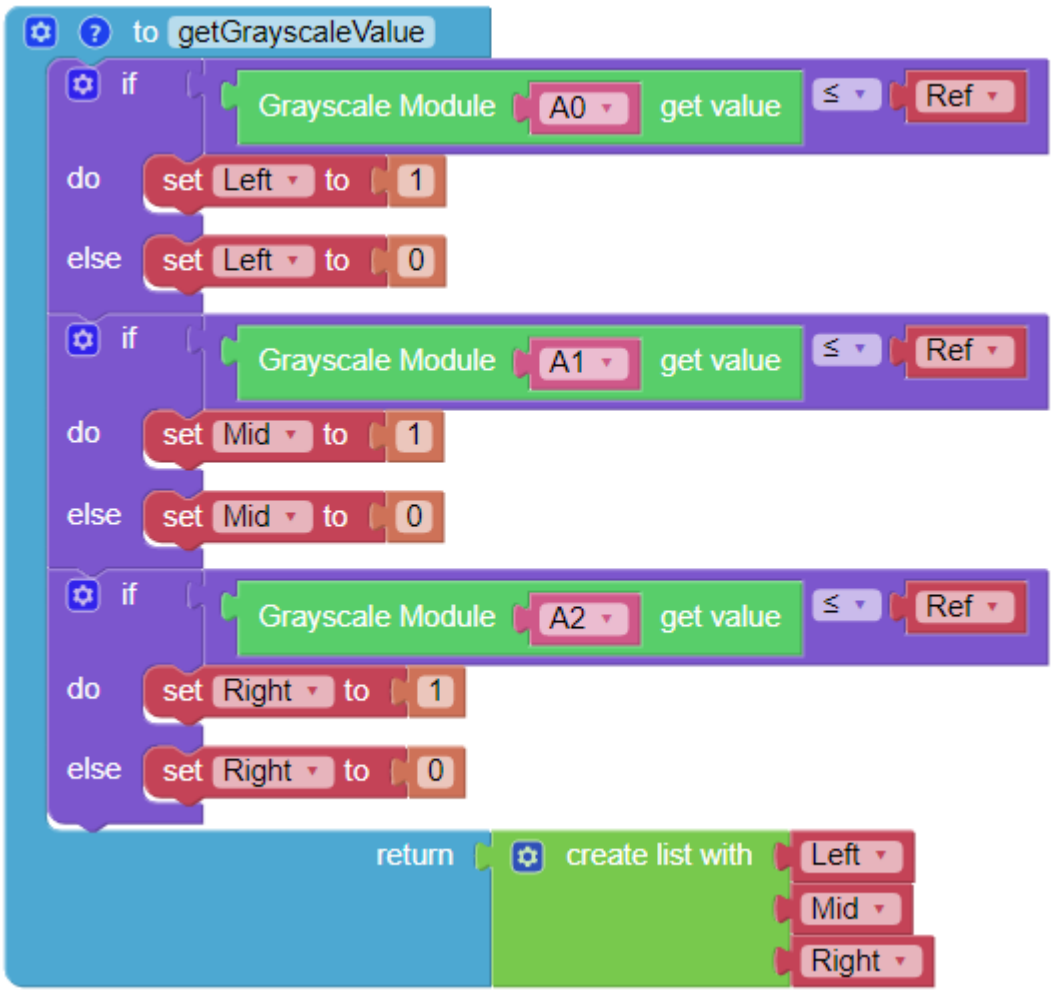
注釈:

- 次の例を参考にしてプログラムを作成してください。またチュートリアルを参照してください。How to Create a New Project?.
- または EzBlock Studio の **Examples** 画面から「 **Minecart Plus** 」を探し **Run** か **Edit** を直接クリックしてください。


Start
set Ref to 950
Forever
set sta to getDirection
if sta ≠ " OUT "
do set lastSta to sta
if sta = " FORWERD "
do turn steering angle to 0
forward at 10 % speed
else if sta = " LEFT "
do turn steering angle to 20
forward at 10 % speed
else if sta = " RIGHT "
do turn steering angle to -20
forward at 10 % speed
else if sta = " OUT "
do outHandle


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



5.17 闘牛

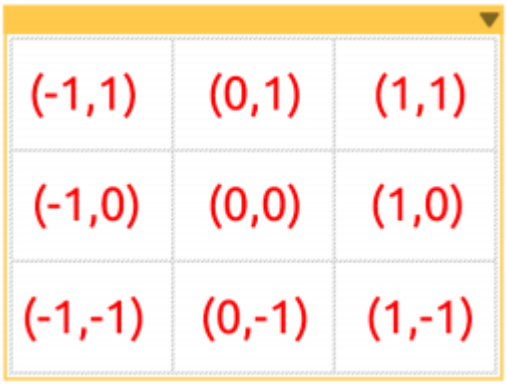
PiCar-X が怒った雄牛に変身！ ハンカチなどの赤い布を用意して、闘牛士になりきりましょう。 PiCar-X が赤い布を追うときは、ぶつからないように注意！

注釈: このプロジェクトは、先行するプロジェクトよりも進んでいます。 PiCar-X は、色検出機能を使用してカメラを赤い布に向け、さらにカメラが向いている方向に車体の向きを変えて追いかけます。

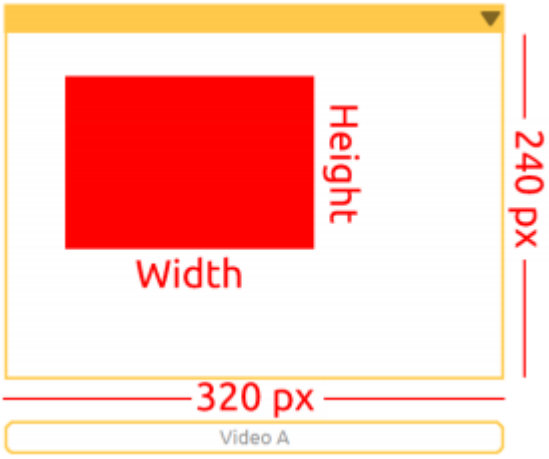
TIPS

color detection [red] ブロックを **Start** ブロックに追加して、PiCar-X のカメラが赤い色をターゲットとして探す

ようにすることから始めます。 **forever** ループ・ブロックで、検出された色の **[width] of detected color** ブロックを追加して、入力を **object detection** グリッドに変換します。



「object detection」は、カメラ画像の中心点に基づいて検出されたターゲットの座標を (x, y) 値で出力します。以下に示すように、画面は 3x3 グリッドに分割されます。したがって、赤い布がカメラの画像の左上にある場合、(x, y) 座標は (-1, 1) になります。

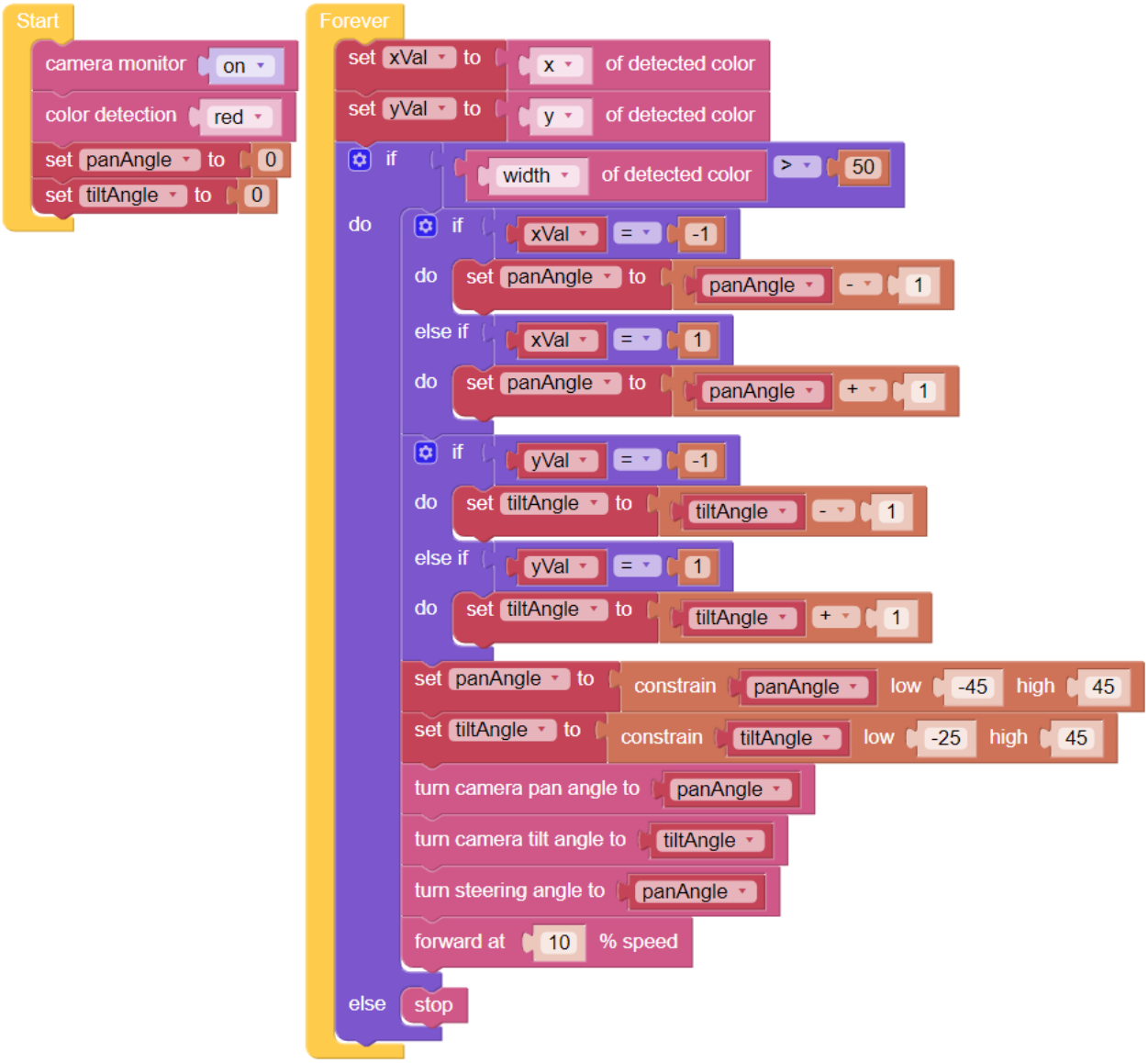


「object detection」は検出されたターゲットの幅と高さを検出します。複数のターゲットが識別された場合、最大のターゲットの寸法が記録されます。

例：

注釈:

- 次の例を参考にしてプログラムを作成してください。またチュートリアルを参照してください。: How to Create a New Project?.
- または EzBlock Studio の **Examples** 画面から「 **Bullfight** 」を探し **Run** か **Edit** を直接クリックしてください。



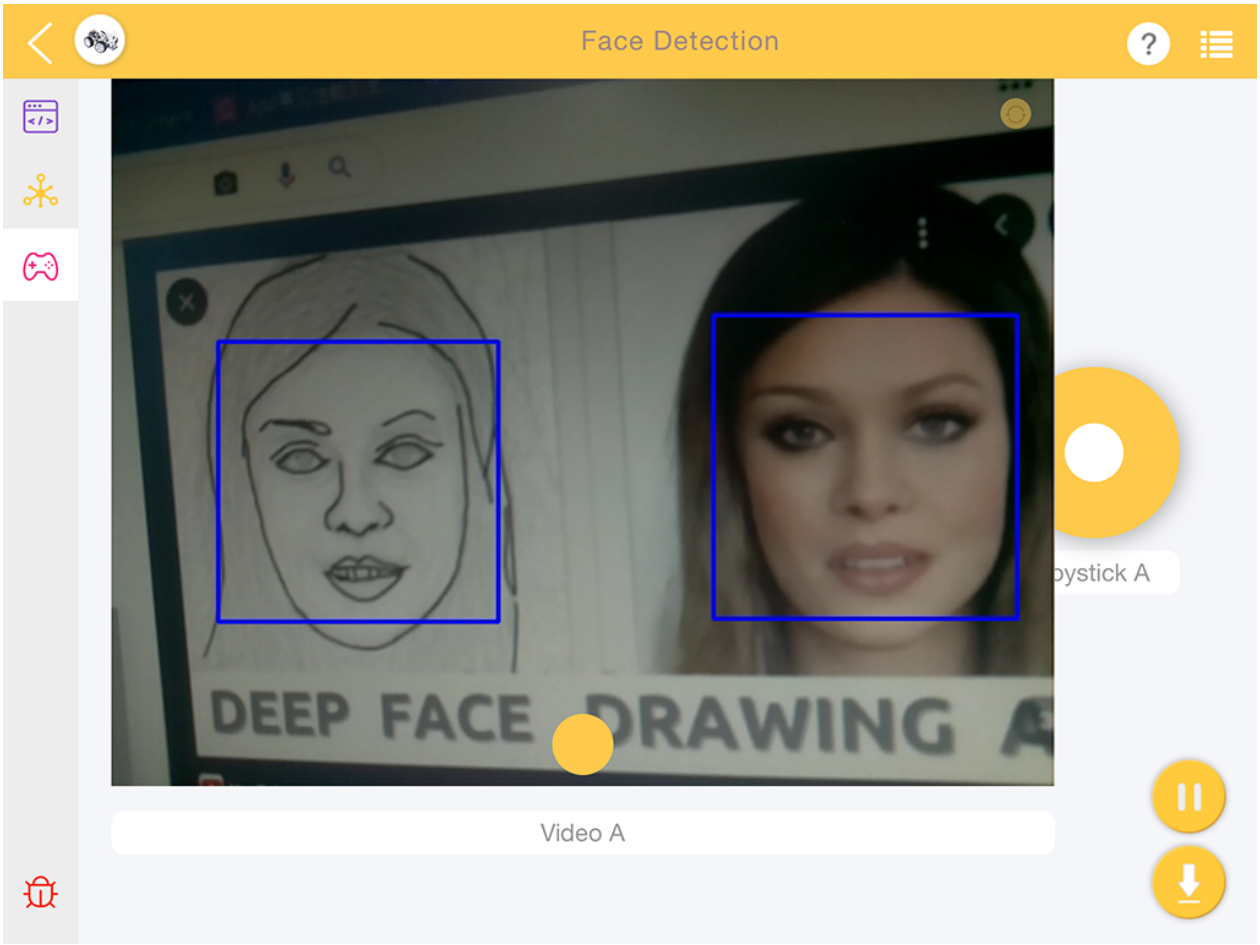
5.18 歩行者に注意

このプロジェクトは、PiCar-X が道路状況に基づいて適切な措置を実行するようにします。 走行中に歩行者を検知すると、PiCar-X は完全に停止します。

プログラムが実行されたら、PiCar-X の前に人物の写真をかざします。 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]** 値を **if do else** ブロックに送信する判断手順が作成されます。 判定手順は人間の顔を 10 回探し、顔が現れた場合は **[count]** を +1 増やします。 **[count]** が 3 より大きい場合、PiCar-X は動きを止めます。

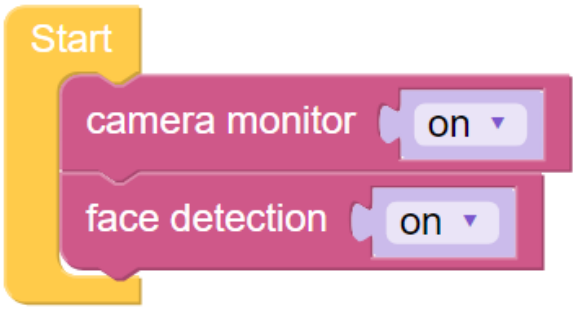
- How to Use the Remote Control Function?

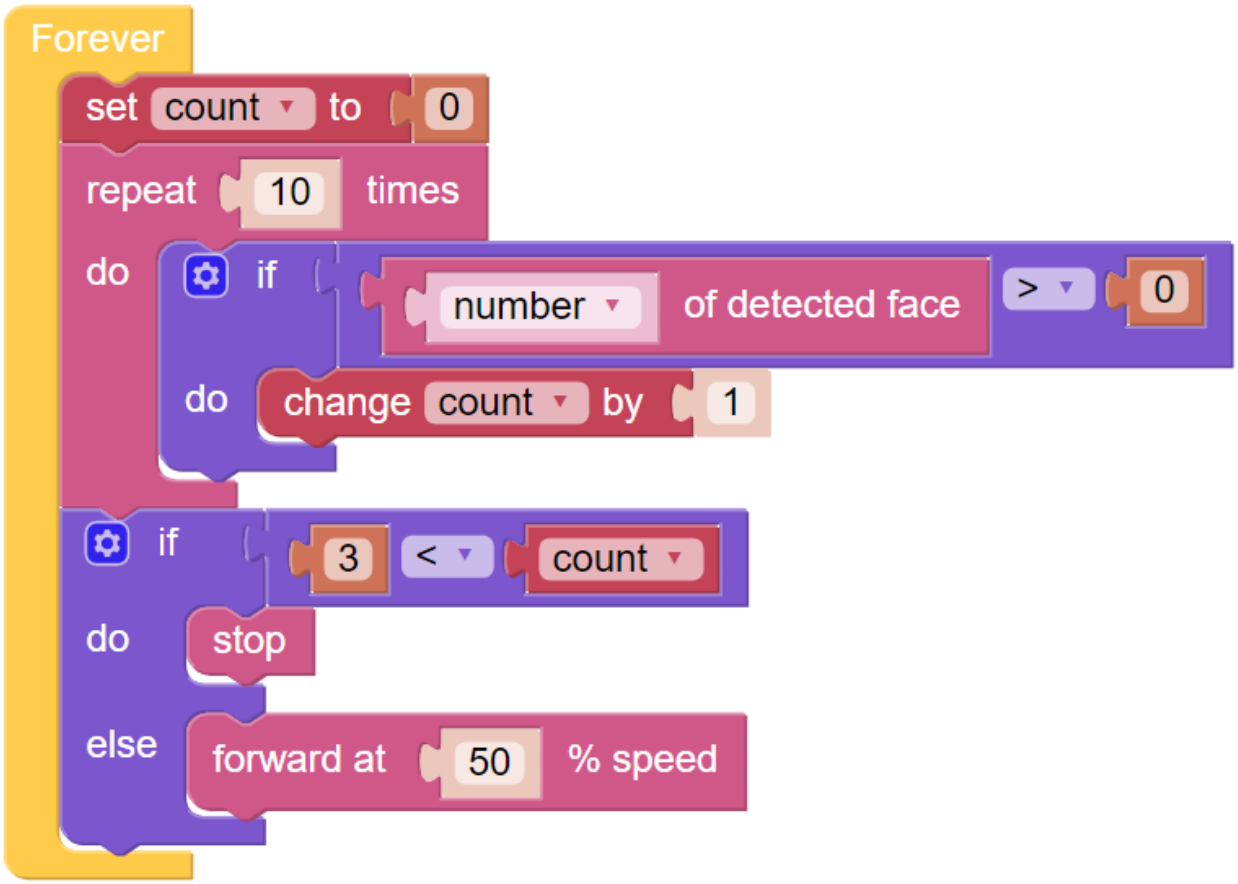


例：

注釈:

- 次の例を参考にしてプログラムを作成してください。またチュートリアルを参照してください。How to Create a New Project?
- または EzBlock Studio の **Examples** 画面から「 **Beware of Pedestrians** 」を探し **Run** か **Edit** を直接クリックしてください。





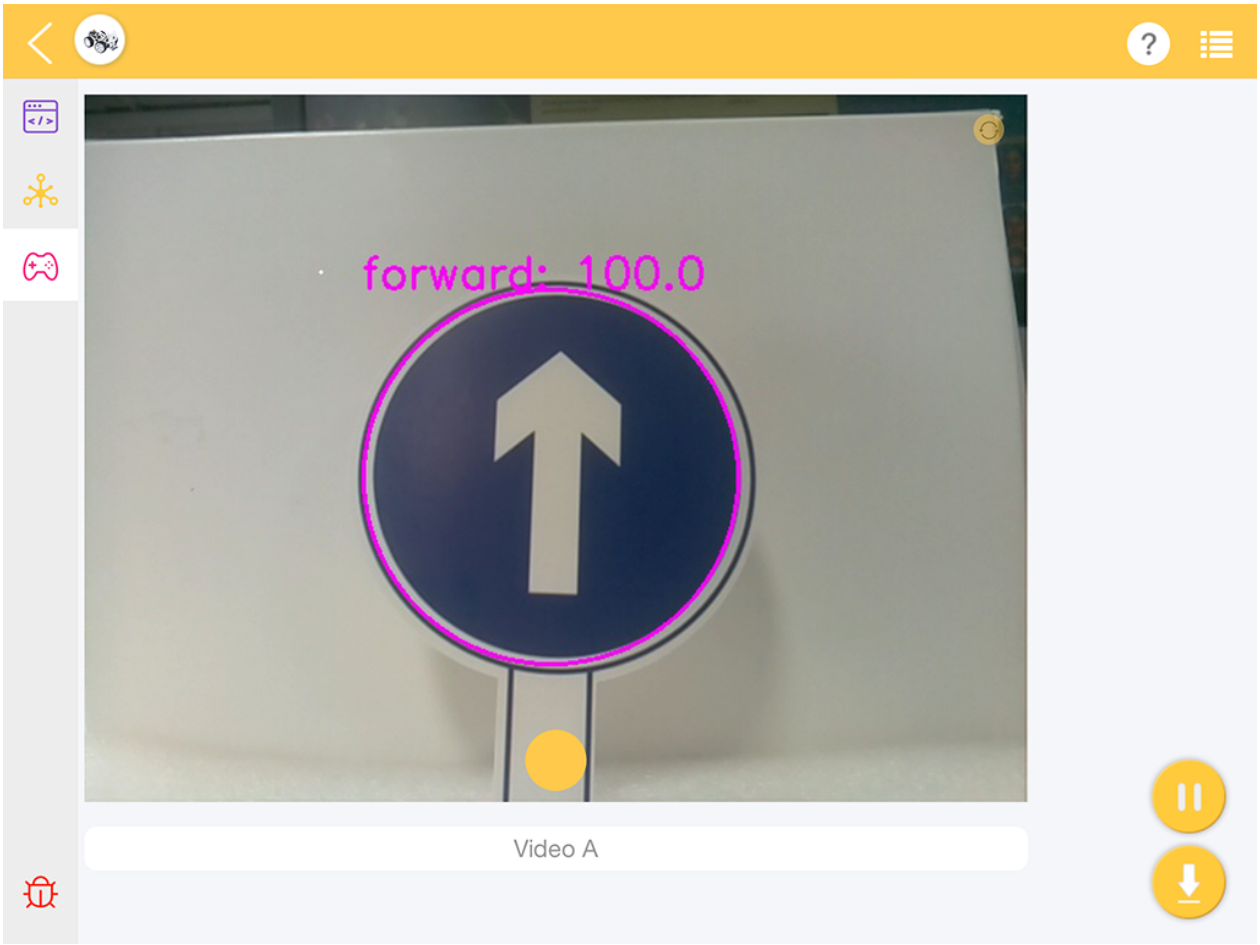
5.19 交通標識識別

このプロジェクトは、画像検出機能を使用して交通標識を探し、PiCar-X を標識の指示に従従わせるものです。**traffic sign detection [on]** ブロックは、以下の印刷可能な PDF に含まれている 4 つの異なる交通標識を認識します。PiCar-X が **STOP** 標識を検出すると停止します。**FORWARD** 標識は前進し、**LEFT** または **RIGHT** 矢印はその方向に曲がります。

- `[PDF]Traffic Sign Cards`

注釈: 印刷された色は、プリンターのトナーの違い、または印刷用紙のわずかな色の違いなどによって、Ezblock が設定した色とはわずかに異なる色合いになる場合があります。 これにより認識の精度が低下する可能性があります。

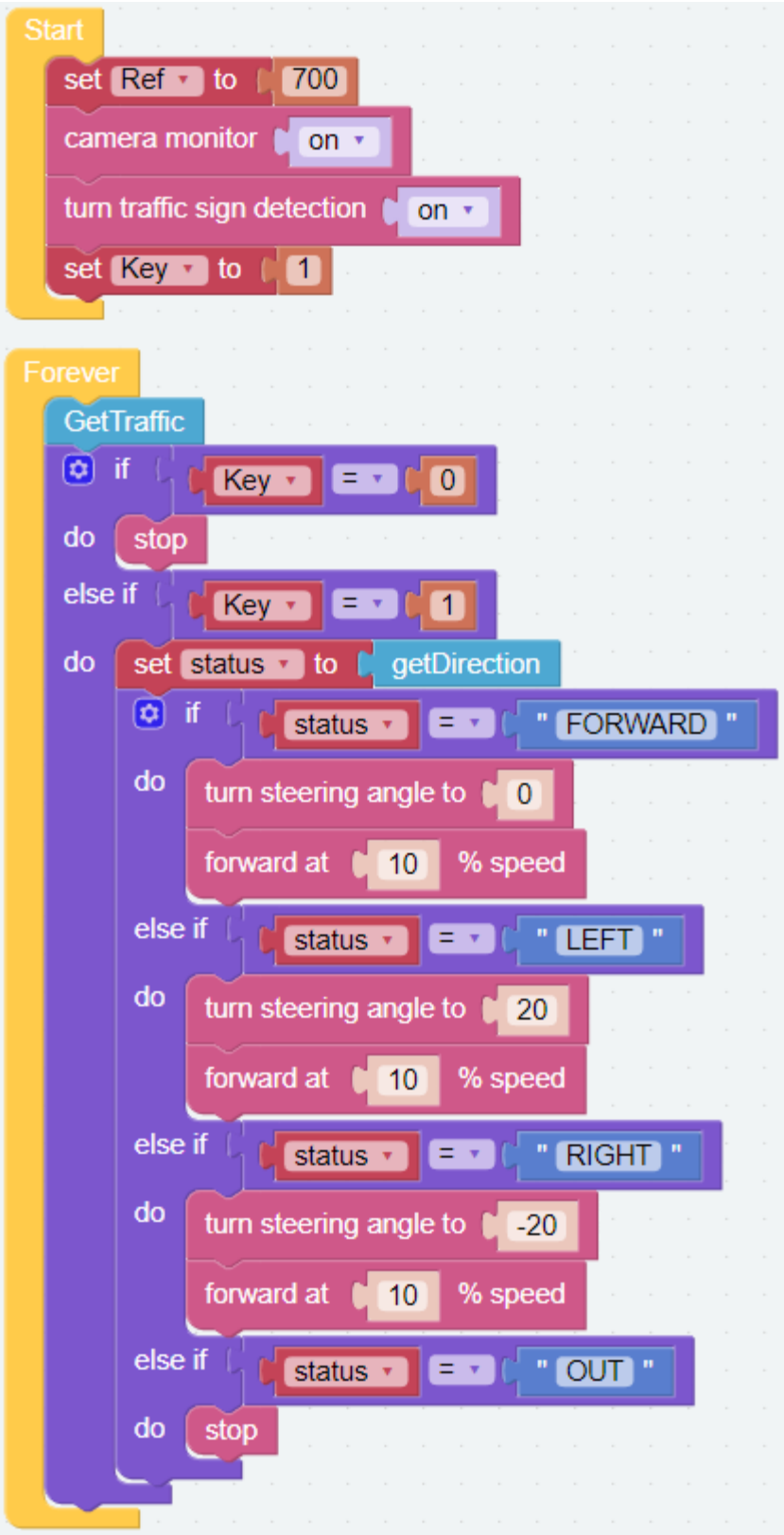
このプロジェクトは **トロッコ** に基づいていますが、グレースケール・センサーを使用する代わりにカメラによる交通標識検出の機能を使用します。 検出結果は、Ezblock Studio のビデオ モニターでも確認できます。

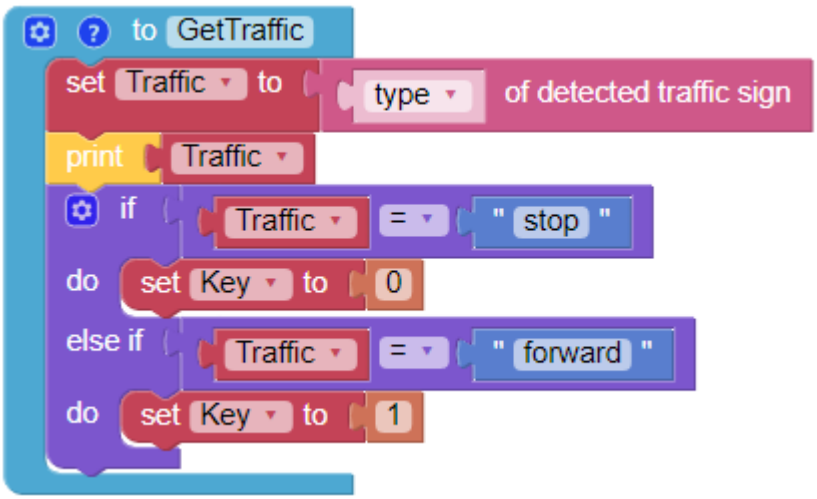


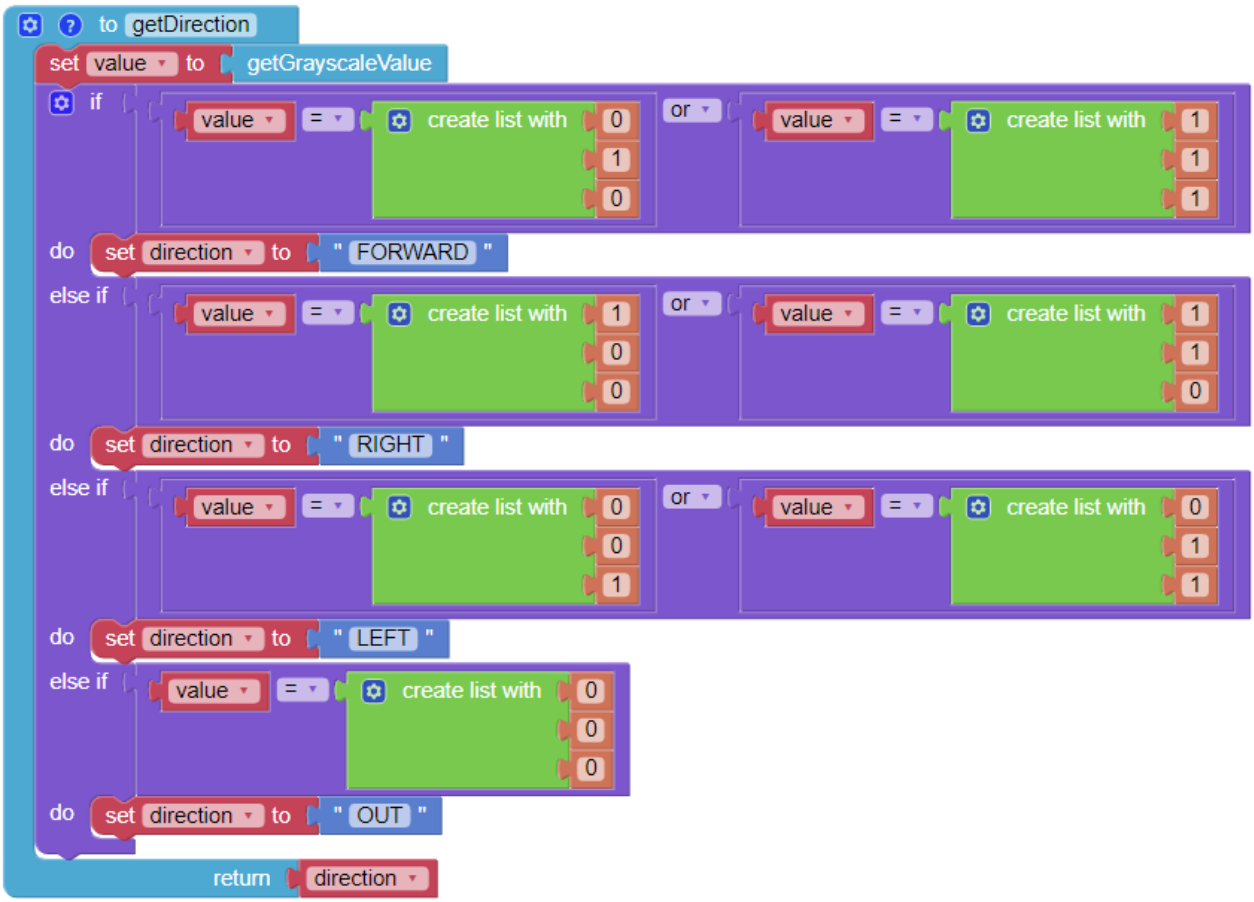
ティップス

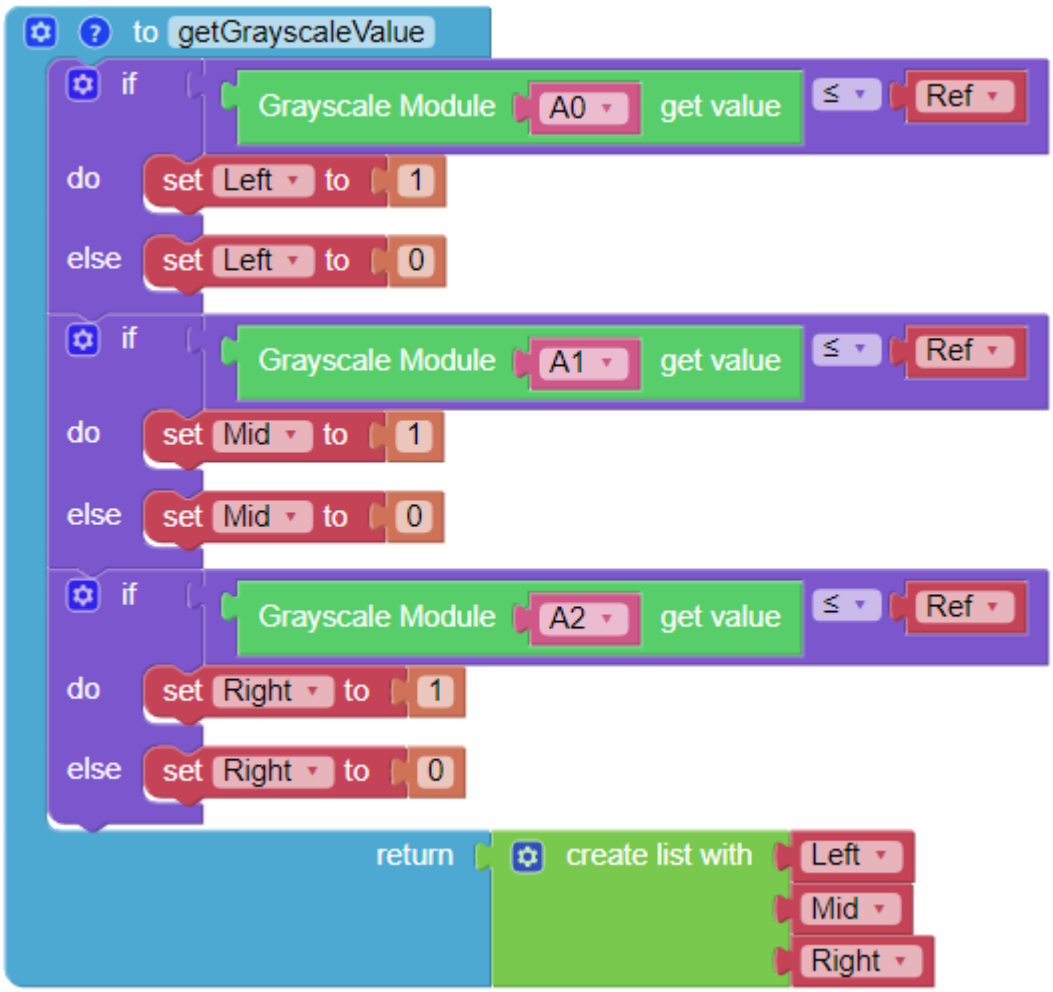
注釈:

- 次の例を参考にしてプログラムを作成してください。またチュートリアルを参照してください。How to Create a New Project?.
- または EzBlock Studio の **Examples** 画面から「 **Traffic Sign Detection** 」を探し **Run** か **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



5.20 オリエンテーリング・ゲーム

このプロジェクトはリモートコントロール機能を使って PiCar-X を動かして、借り物競争みたいに指定された色を探すゲームです。

まず、PiCar-X が通り抜けることができる障害物コース、迷路、または広めの平らな床がある部屋を設定します。次に、ルートに沿って 6 つのマーカーをランダムに配置し、PiCar-X が見つけられるように 6 つのマーカーのそれぞれにカラーカードを置きます。

６つのカラーカード：赤、オレンジ、黄色、緑、青、紫の６色となります。以下の PDF が用意されているのでカラープリンターから印刷してご使用ください。

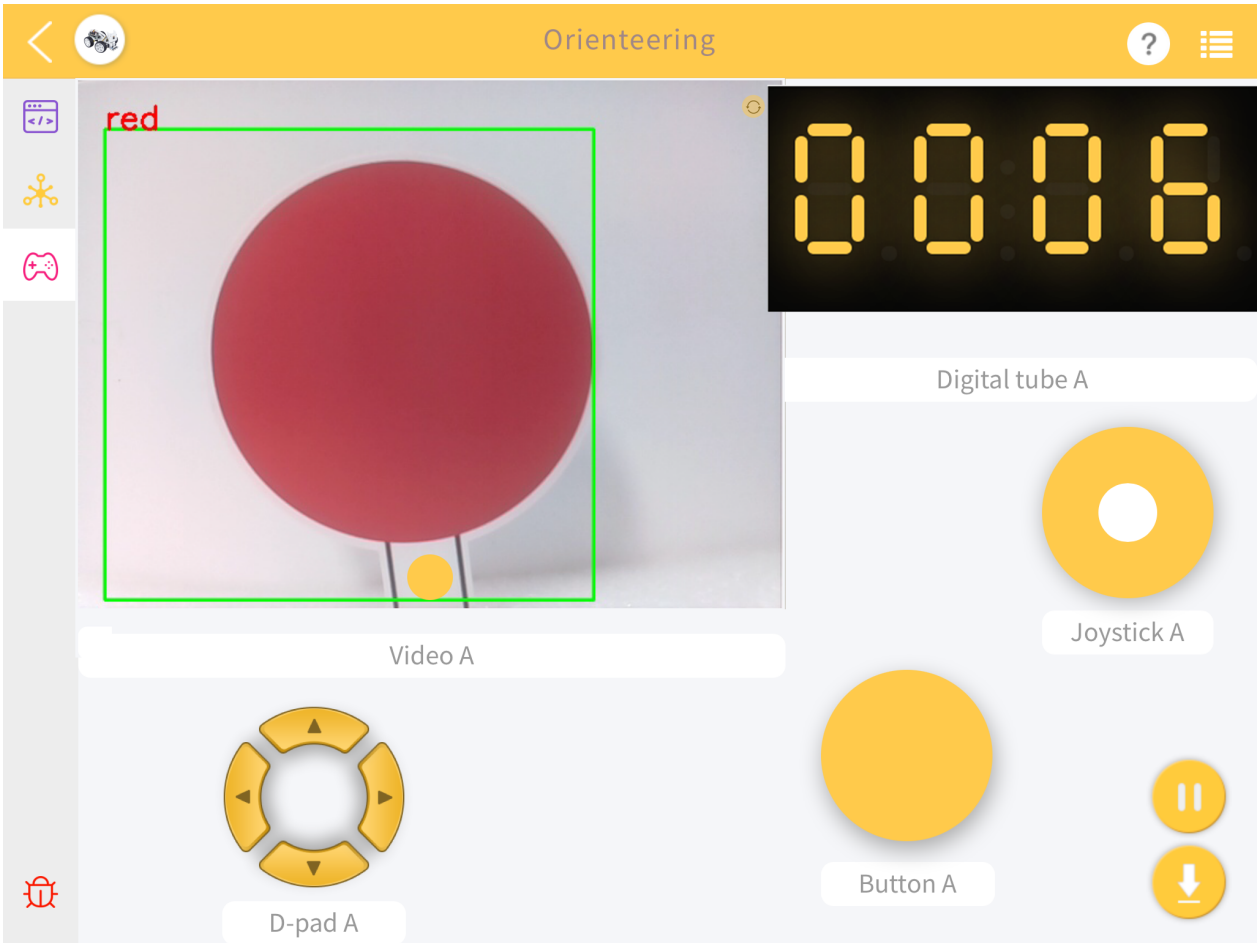
- `[PDF]Color Cards`

注釈: 印刷された色はプリンターや印刷する紙によって EzBlock のカラーモデルとは少し違ったものになります。少し色がついた紙などを使用した場合などでは色の認識率が下がります。

このゲームでは PiCar-X が 6 色の中からランダムに 3 色を選んで TTS 機能（文章読み上げ機能）で探すべき色を順番に伝えてきます。

そしてそれらの色を出来るだけ早く見つける様にします。 指定された色のカードにある程度近づくと音が鳴ってその色をクリアしたことが知らされて次の色が指定されます。

PiCar-X をスタート場所に決めた位置に置き、リモートコントロール画面のボタンを押してゲームを始めます。

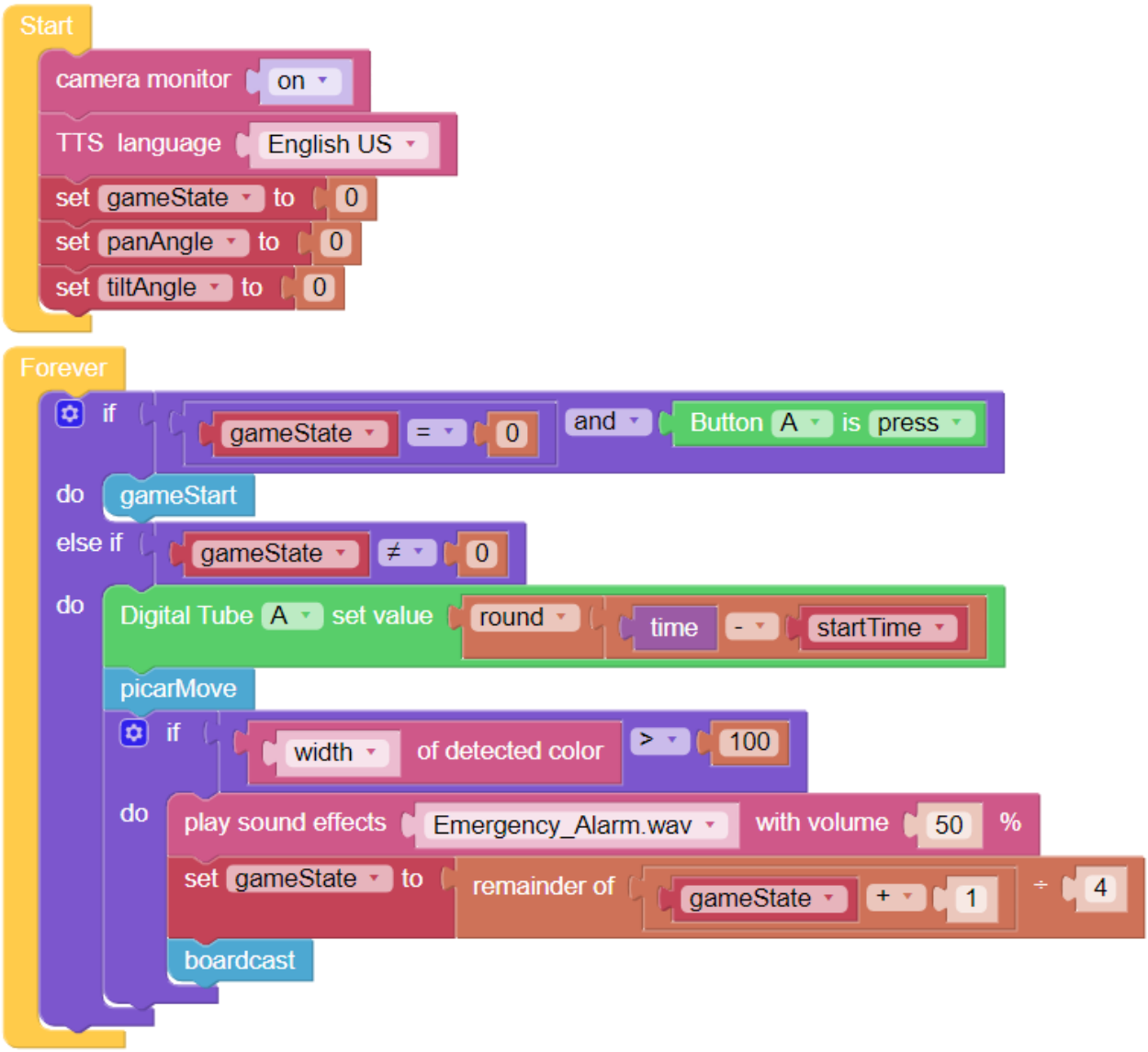


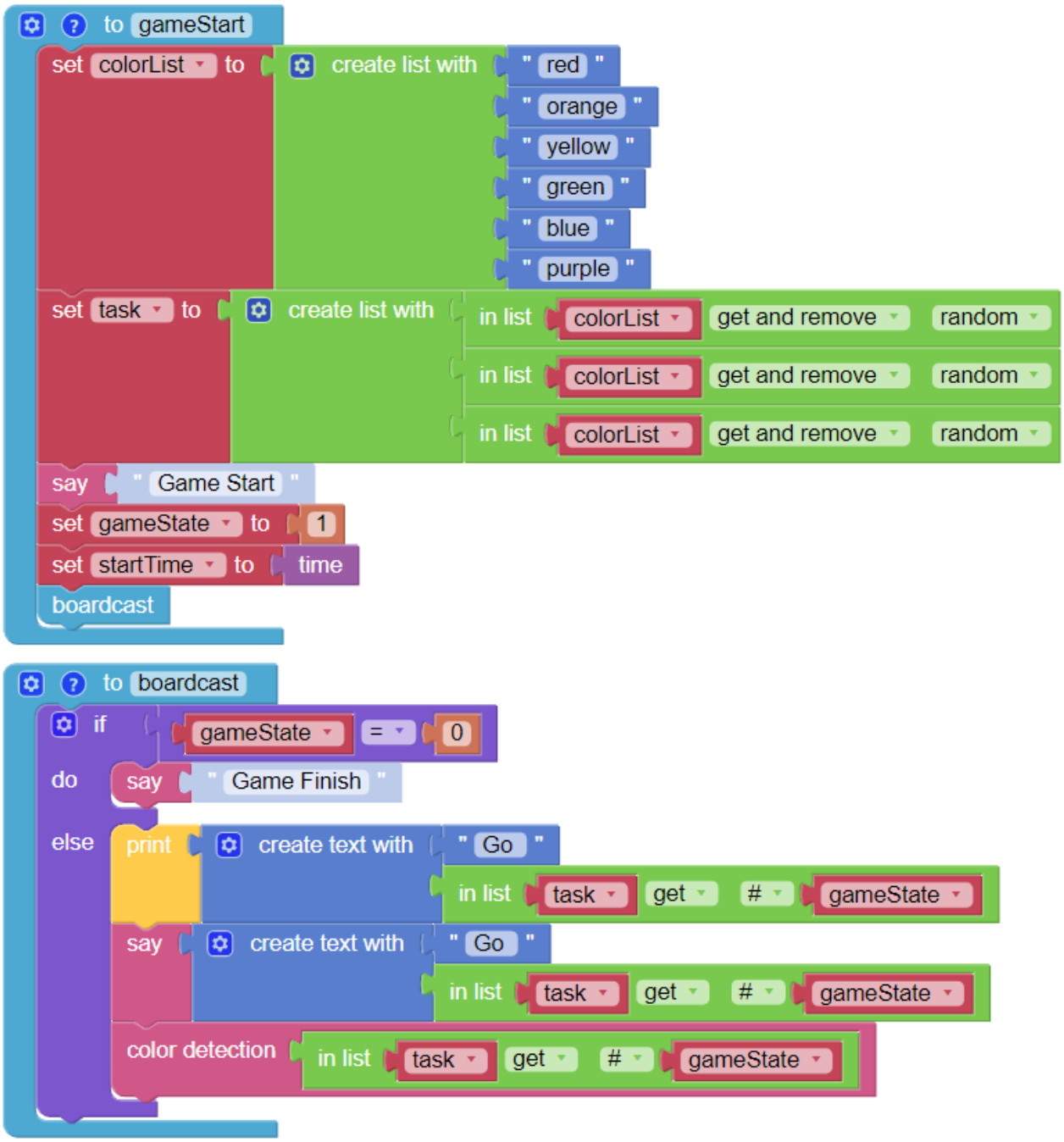
友人と交代で誰が一番上手く PiCar-X を操作し、一番早く目的のカードを探せるか競い合いましょう。

EXAMPLE

注釈:

- 次の例を参考にしてプログラムを作成してください。またチュートリアルを参照してください。: How to Create a New Project?.
- または EzBlock Studio の **Examples** 画面から「 **Orienteering** 」を探し **Run** か **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 6 章

付録

6.1 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。

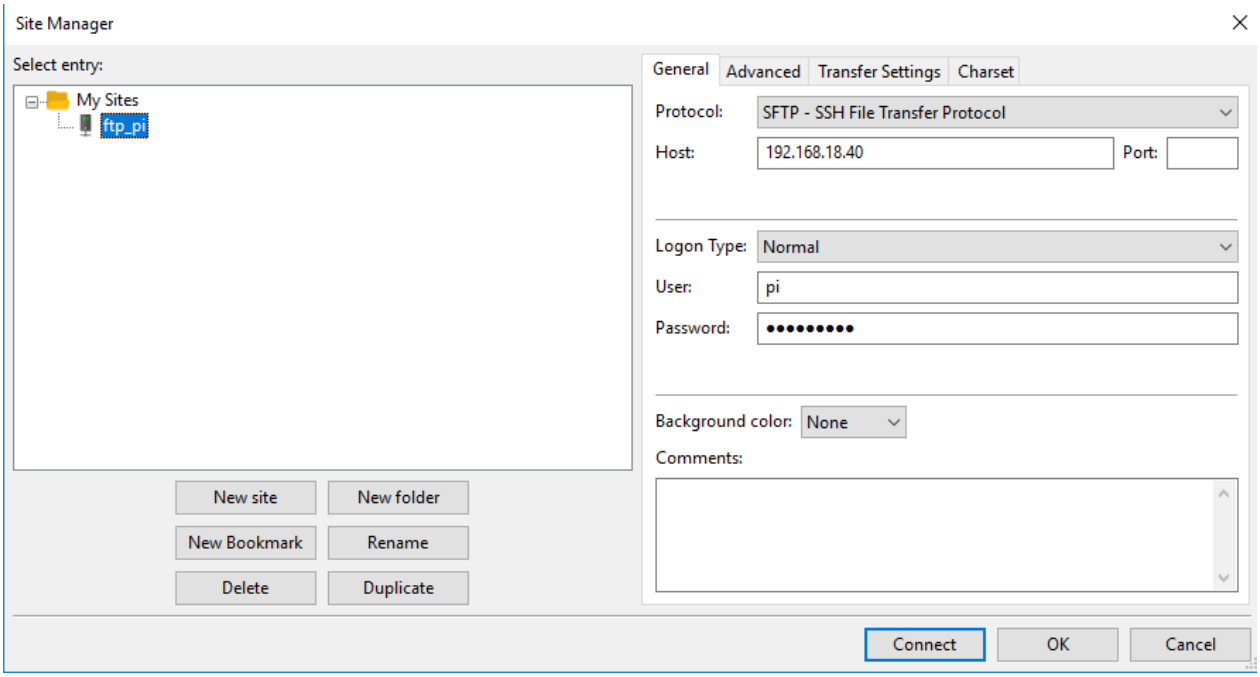
ステップ 1：Filezilla をダウンロードする。

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

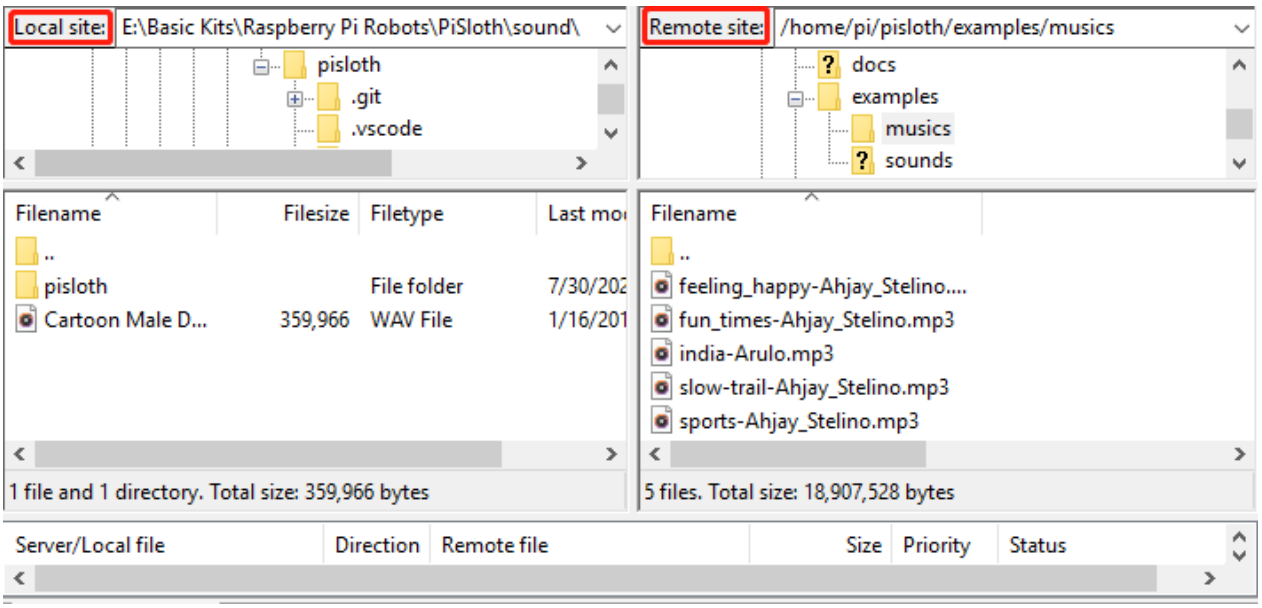
簡単なインストール後、開いて FTP サーバーに接続する。接続する方法は 3 つありますが、ここでは **Quick Connect** バーを使用します。 **hostname/IP**、 **username**、 **password**、 **port (22)** を入力し、 **Quick Connect** をクリックするか **Enter** を押してサーバーに接続します。

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。



6.2 PuTTY

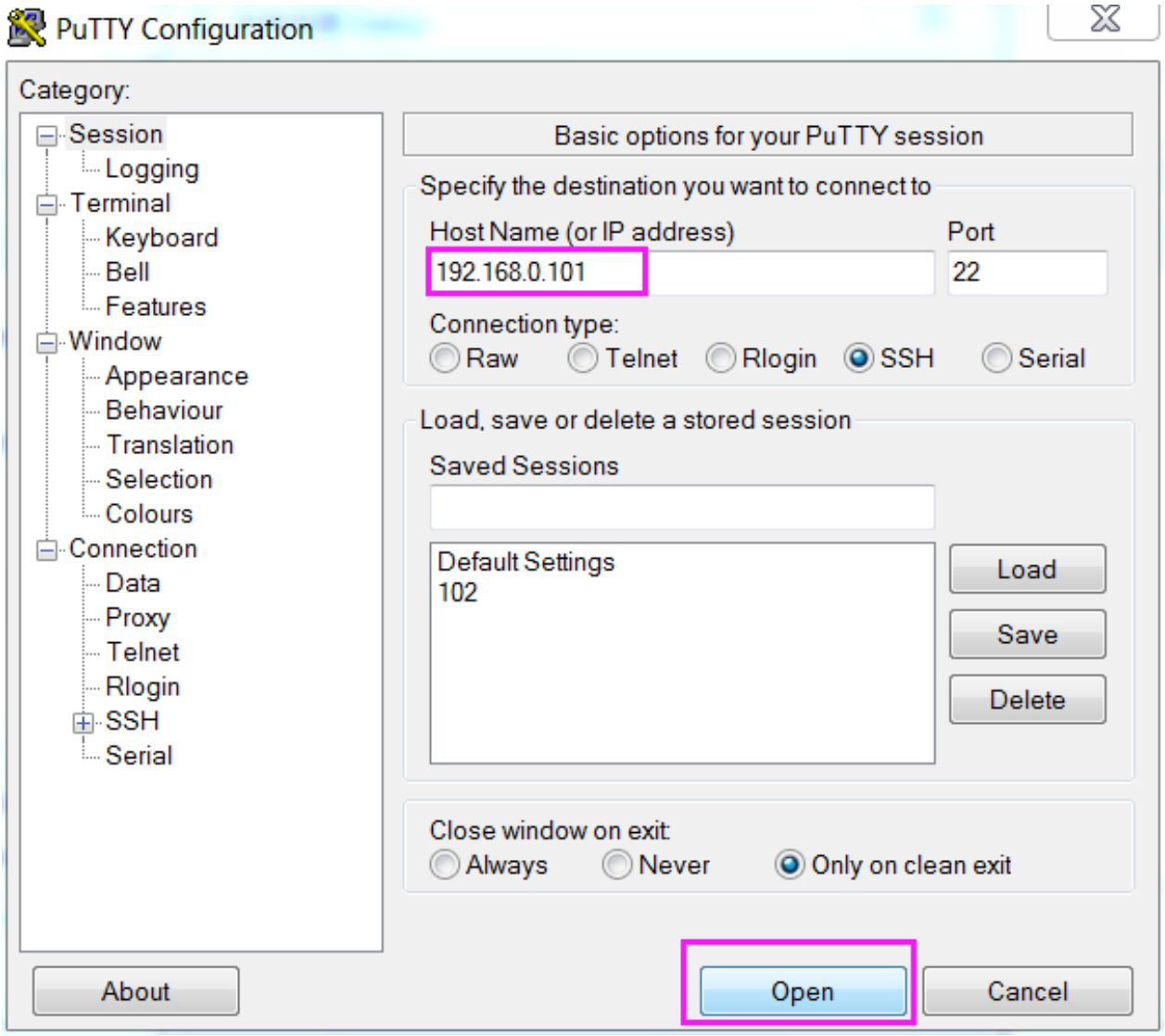
Windows ユーザーの場合、SSH アプリケーションをいくつか使用することができます。ここでは、PuTTY を推奨します。

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

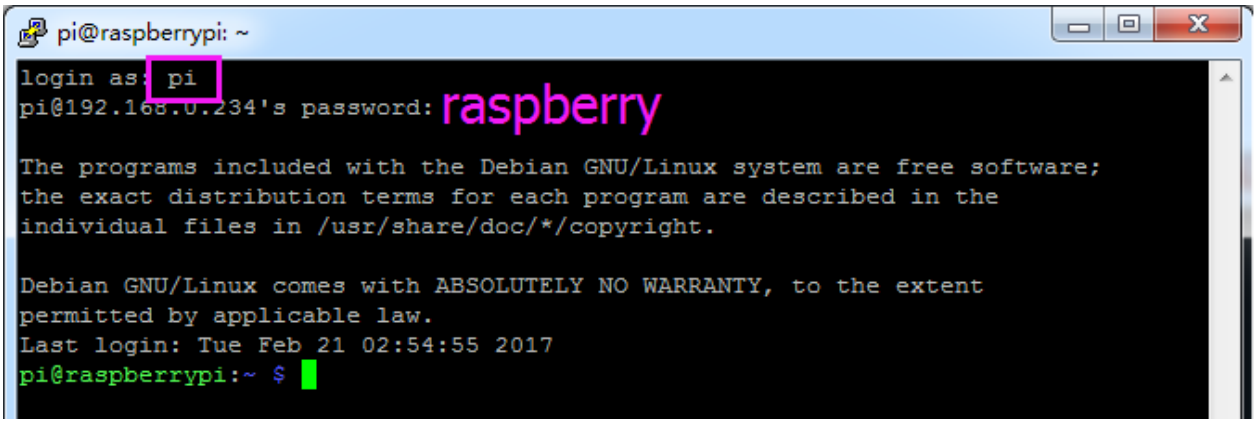
Open をクリックします。IP アドレスを使って Raspberry Pi に初めてログインする際、セキュリティリマインダーが表示されます。 **Yes** をクリックしてください。

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。

6.3 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。

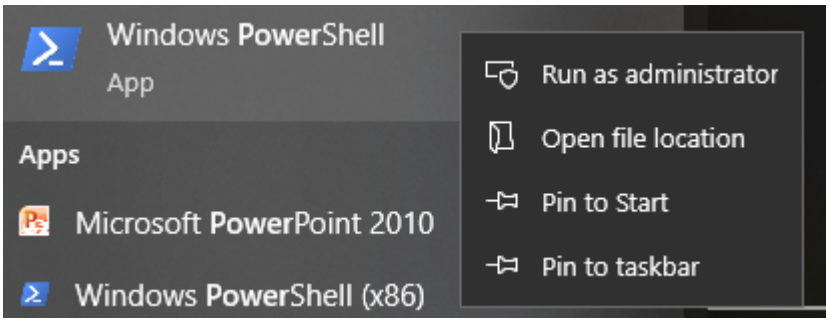
6.4 Powershell を使って OpenSSH をインストールする

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
```

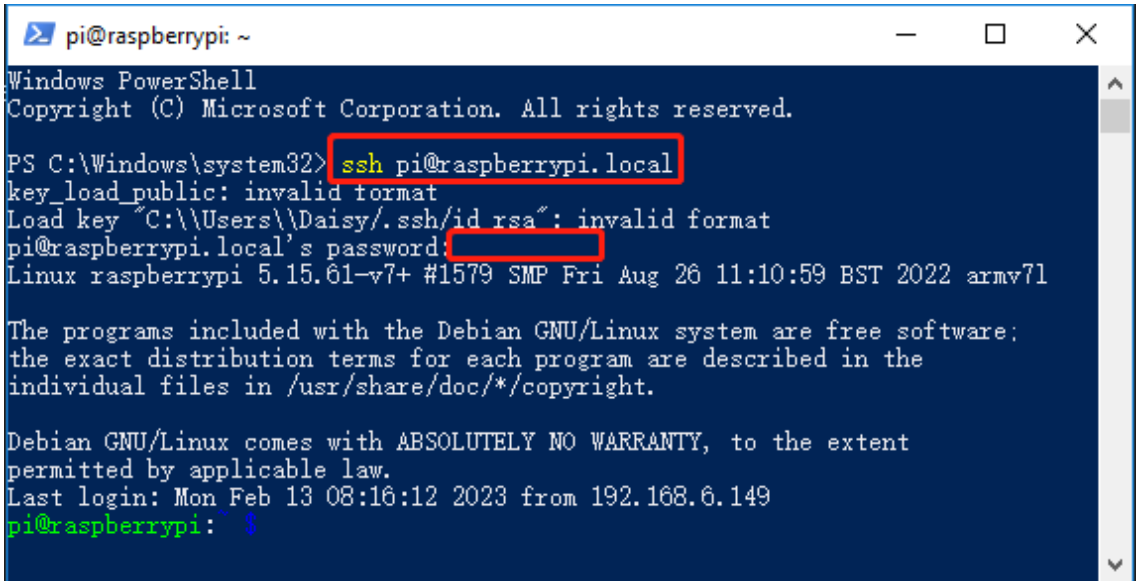
(次のページに続く)

(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。



6.5 バッテリーについて

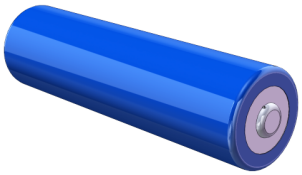
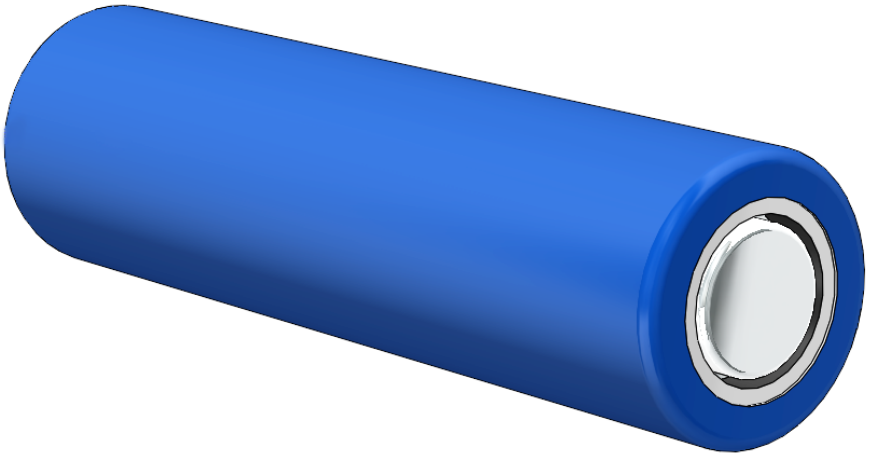
使用可能バッテリー

- 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 7 章

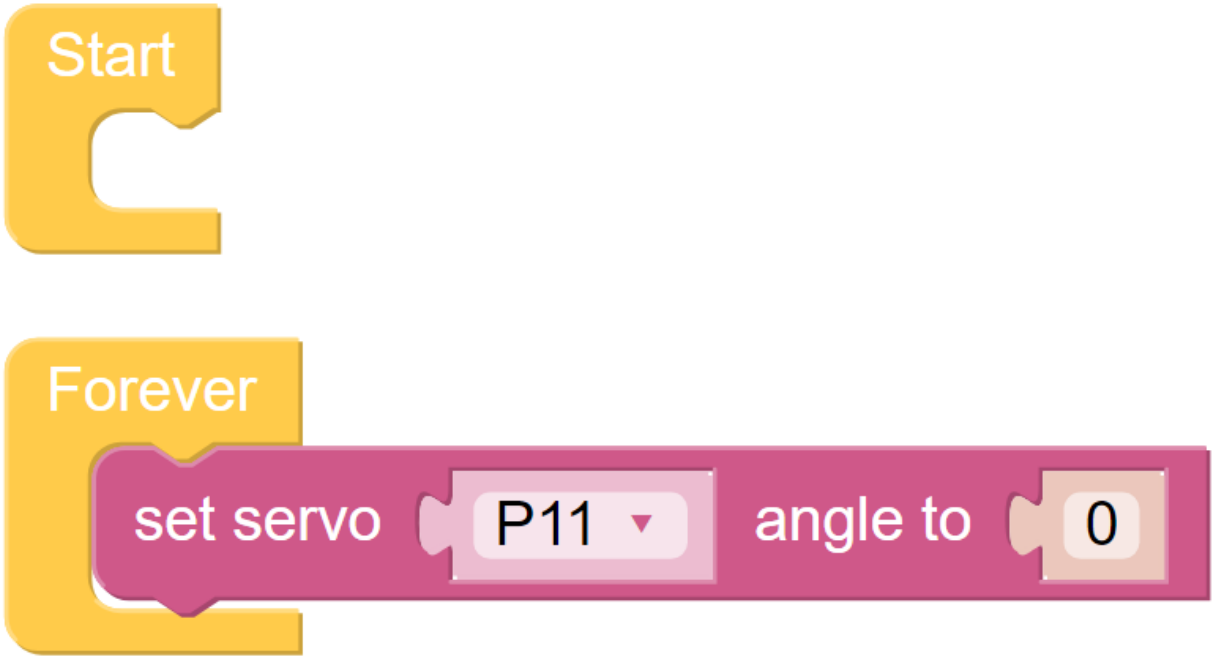
FAQ

Q1: Ezblock OS をインストール後、サーボが 0 ° に回らない？

1) サーボケーブルが正しく接続されているか、Robot HAT の電源が入っているかを確認してください。

2) Robot HAT の RST（リセット）ボタンを押してみてください。

3) Ezblock Studio で既にプログラムを実行している場合、Robot HAT 上のピン「P11」用のカスタムプログラムは使用できなくなります。 以下の図を参照してサーボ角度を 0 に設定するプログラムを Ezblock Studio で手動で作成できます。



Q2: VNC を使用しているときに「現在デスクトップを表示できない」というプロンプトが表示されますか?

ターミナルで `sudo raspi-config` と入力して解像度を変更します。

Q3: 理由もなくサーボが時々中間位置に戻るのはなぜですか?

サーボが何らかによってブロックされ（タイヤが本体にぶつかるなど）目的の位置に到達できない場合、過電流によるサーボの焼損を防ぐためにサーボは電源をオフにして保護モードに入ります。

電源が切れた後、サーボに PWM 信号が与えられない場合にサーボは自動的に元の位置に戻ります。

第 8 章

謝辞

私たちの製品を評価してくれた方々、チュートリアルに提案を提供してくれたベテランの方々、そして私たちをフォローしてサポートしてくれたユーザーに感謝します。私たちへの貴重なご意見はより良い製品を提供するための私たちのモチベーションとなっております!

特に以下の方々へ感謝いたします。

- Len Davisson
- Kalen Daniel
- Juan Delacosta

アンケートにご協力いただけると幸いです。

注釈: アンケート送信後にトップに戻って結果をご確認ください。

第 9 章

Copyright Notice

このマニュアルのすべての内容（テキスト、画像、コードを含むがこれらに限定されない）は SunFounder Company が所有しています。 著者および関連する権利所有者の法的権利を侵害することなく、関連する規制および著作権法の下で、個人的な研究、調査、娯楽、またはその他の非営利または非営利の目的でのみ使用する必要があります。これらを許可なく営利目的で使用する個人または組織については当社は法的措置を取る権利を留保します。