

第4版

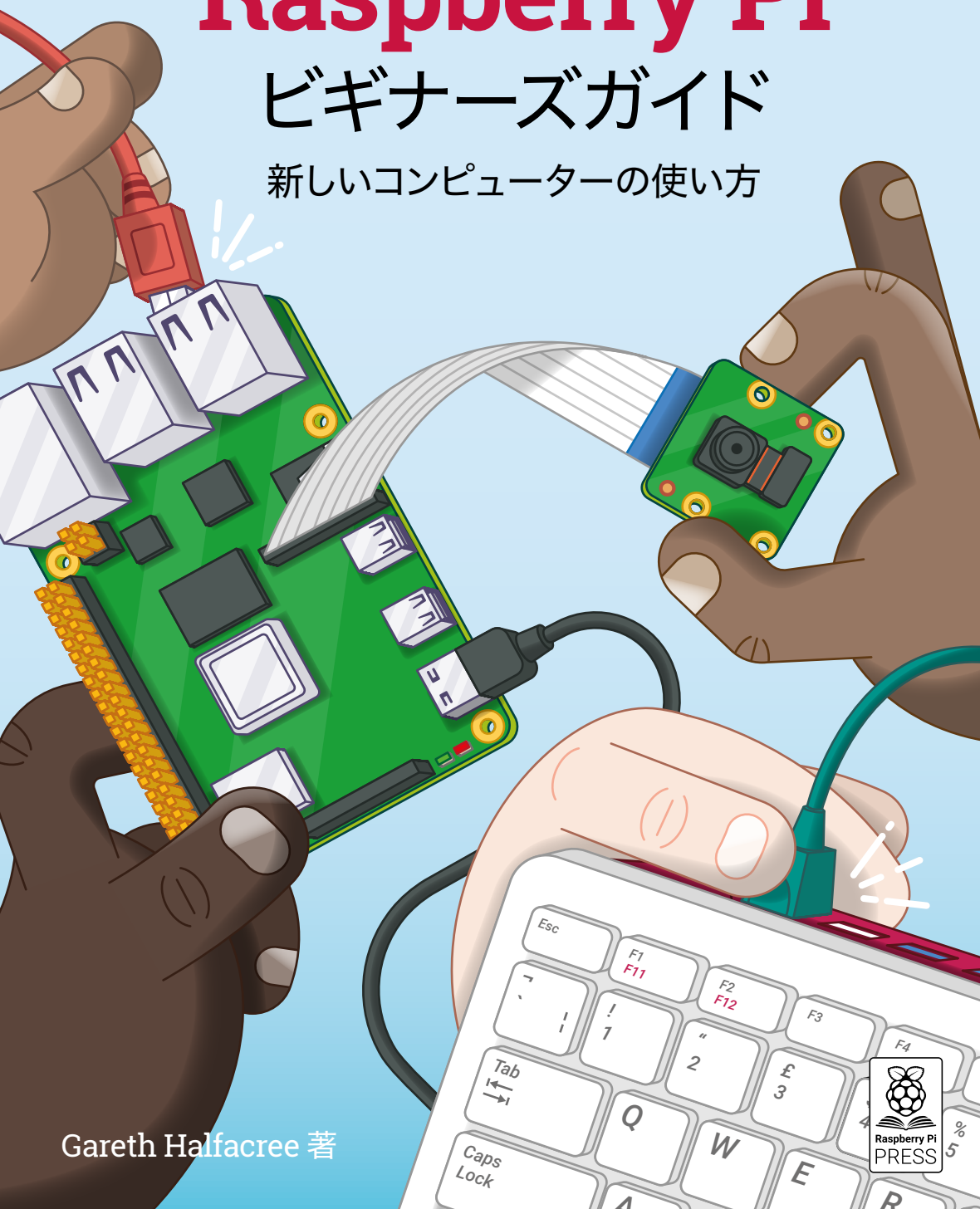
RASPBERRY PI 400 対応版

公式

Raspberry Pi

ビギナーズガイド

新しいコンピューターの使い方



Gareth Halfacree 著



公式
Raspberry Pi
ビギナーズガイド
新しいコンピューターの使い方



2020年初版、発行者: Raspberry Pi Trading Ltd (Maurice Wilkes Building,
St. John's Innovation Park, Cowley Road, Cambridge, CB4 0DS)

出版ディレクター: Russell Barnes ・ 編集者: Phil King
デザイン: Critical Media ・ イラスト: Sam Alder
CEO: Eben Upton

ISBN: 978-1-912047-81-9

発行者および関係者は、本書で言及または宣伝されている
商品、製品、またはサービスに関する不作為または過失に対して一切の責任を負わないものと
します。

特に明記されていない限り、本書の内容は Creative
Commons Attribution-NonCommercial-ShareAlike 3.0 Unported
(CC BY-NC-SA 3.0) ライセンスが適用されるものとします。

公式 Raspberry Pi ビギナーズガイドへようこそ

はじめに、みなさんに Raspberry Pi を気に入ってもらえることを期待しています。Raspberry Pi は手頃な価格のコンピューターで、コーディングの学習やロボットの制作、多様なプロジェクトの作成に使用することができます。Raspberry Pi は標準のボード型のもや、キーボードが統合された新しい Raspberry Pi 400 があります。

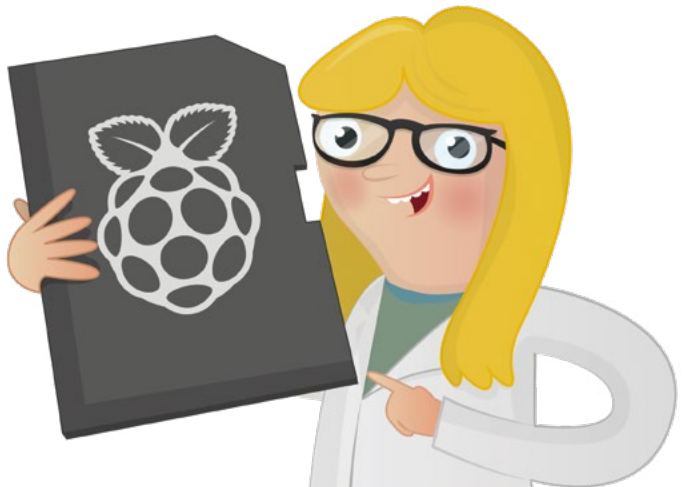
Raspberry Pi はインターネットを見たりゲームをプレイしたり、映画や音楽を楽しんだり、ふつうのコンピューターでできることが同じようにできます。ただし、Raspberry Pi にはふつうのコンピューターよりできることがもっとあります。

Raspberry Pi ではコンピューターの内部にアクセスすることができます。自分でオペレーティングシステムをセットアップして、ケーブルや回路を GPIO ピンに直接つなげられます。公式のオペレーティングシステムには、若い人たちへプログラミングを教えるために必要になる Scratch や Python をはじめとした主要なプログラミング言語が用意されています。

世界がこれまで以上にプログラマーを求めるなか、Raspberry Pi は新しい世代の人々のコンピュータサイエンスや技術への好奇心に火をつけています。

あらゆる世代の人々が、Raspberry Pi を使ってレトロゲーム機から気象観測所まであらゆるエキサイティングなプロジェクトを作っています。

もしゲームの作成、ロボットの制作、様々な素晴らしいプロジェクトの改造 (ハック) に興味を持ったなら、本書がそれら始めるための手助けになるでしょう。



著者について

Gareth Halfacree はフリーランスの技術ジャーナリスト兼ライターです。教育分野でシステム管理者を務めていたこともあります。オープンソースのソフトウェアとハードウェアに情熱を注いでいて、Raspberry Pi プラットフォームを早期に活用し、その機能と柔軟性についていくつかの出版物を執筆しています。詳細については、Twitter 上で **@ghalfacree** を検索するか、彼の Web サイト **freelance.halfacree.co.uk** にアクセスしてください。



目次

第 1 章:Raspberry Pi を知ろう	008
Raspberry Pi について詳しくご紹介します	
第 2 章:Raspberry Pi をセットアップしよう	022
Raspberry Pi を動かすために必要な周辺機器を接続します	
第 3 章:Raspberry Pi を使ってみよう	036
Raspberry Pi オペレーティングシステムについて学びます	
第 4 章:Scratch 3 を使ってプログラミングしてみよう	054
かんたんに学べるブロックベースの言語でコーディングを始めます	
第 5 章:Python を使ってプログラミングしてみよう	092
Python を使ってテキストベースのコーディングに取り組んでみます	
第 6 章:Scratch と Python を使って 物理的コンピューティングに挑戦しよう	120
Raspberry Pi の GPIO ピンに接続されている電子部品を制御します	
第 7 章:Sense HAT を使って物理的コンピューティングに挑戦しよう	152
アドオンボードを使ってセンサーやLEDマトリクスを使いこなします	
第 8 章:Raspberry Pi の Camera Module を使ってみよう	196
小さなカメラを使って高解像度な写真やビデオを撮影します	
付録	
付録 A: microSD カードに OS をインストールしてみよう	214
付録 B: ソフトウェアをインストール/アンインストールしてみよう	216
付録 C: コマンドラインインターフェイス	222
付録 D: 参考資料	228
付録 E: Raspberry Pi 構成ツール	234
付録 F: High Quality Camera をセットアップしてみよう	240
付録 G: Raspberry Pi の仕様	244
付録 H: Raspberry Pi の安全性とユーザーガイド	247

第1章

Raspberry Pi を知ろう

この章では、クレジットカードカードサイズの新しいコンピューター「Raspberry Pi」の構成について詳しく説明します。さまざまなコンポーネントとその機能について確認しましょう。



Raspberry Pi は、非常にユニークなデバイスです。小型で低コストですが、コンピューターとして完全に機能します。Web サイトを閲覧したり、ゲームをプレイしたり、プログラムの作成方法を学習したり、独自の回路やデバイスを作成したりなど、どのような用途であっても、Raspberry Pi (とそのコミュニティ) があなたをサポートします。

Raspberry Pi は、「シングルボードコンピューター」と呼ばれるデバイスです。シングルボードコンピューターは、デスクトップコンピューター、ノートパソコン、スマートフォンと同じように機能しますが、単一のプリント回路基板上で動作する点が異なります。多くのシングルボードコンピューターと同様に、Raspberry Pi も小型のデバイスです。クレジットカードとほぼ同じサイズですが、小型だからといって機能的に劣るというわけではありません。多くの電力を消費する大型のコンピューターでできることは、Raspberry Pi でもすべて実行することができます (ただし、処理速度については、必ずしも大型コンピューターと同じというわけではありません)。

Raspberry Pi ファミリーは、世界中で実践的なコンピューター教育を促進したいという願いから誕生しました。非営利の RaspberryPi Foundation を創出したメンバーが協力して、Raspberry Pi を開発しました。開発メンバーたちは、Raspberry Pi がこれほど人気が出るとは考えていませんでした。2012 年に製作した数千台はすぐに完売し、それから現在までに世界中で数百万台が販売されました。Raspberry Pi は、一般家庭、学校、オフィス、データセンター、工場だけでなく、自走式のボートや気球などでも利用されています。

最初の Model B が発売されてから、さまざまなモデルがリリースされてきましたが、どのモデルも、特定の用途に合わせて仕様や機能が向上しています。たとえば Raspberry Pi Zero ファミリーは、フルサイズの Raspberry Pi の小型版ですが、USB ポートと有線ネットワークポートを 1 つだけにするなど、いくつかの機能を削ることにより、非常にコンパクトなレイアウトと消費電力の低減を実現しました。

ただし、すべての Raspberry Pi モデルに共通していることがあります。それは、すべてのモデルに「互換性」があるということです。どのモデルで作成されたソフトウェアであっても、すべてのモデルで使用することができます。たとえば、最新バージョンの Raspberry Pi のオペレーティングシステムを、初期モデルの Model B のプロトタイプ版で稼働させることもできます。もちろん、実行速度は遅くなりますが、稼働することは間違いありません。

このガイドでは、Raspberry Pi 4 Model B と、Raspberry Pi の最新バージョンで最も強力なモデルである Raspberry Pi 400 について説明します。ただし、このガイドで説明する内容は他の Raspberry Pi モデルにもあてはまることなので、別のバージョンを使用する場合も心配する必要はありません。



Raspberry Pi 400

Raspberry Pi 400 の場合、キーボードケースに回路基板が組み込まれています。Raspberry Pi のすべてのコンポーネントを確認する場合は、これ以降の説明を読んでください。Raspberry Pi 400 について確認する場合は、20 ページに進んでください。



Raspberry Pi の詳しい説明

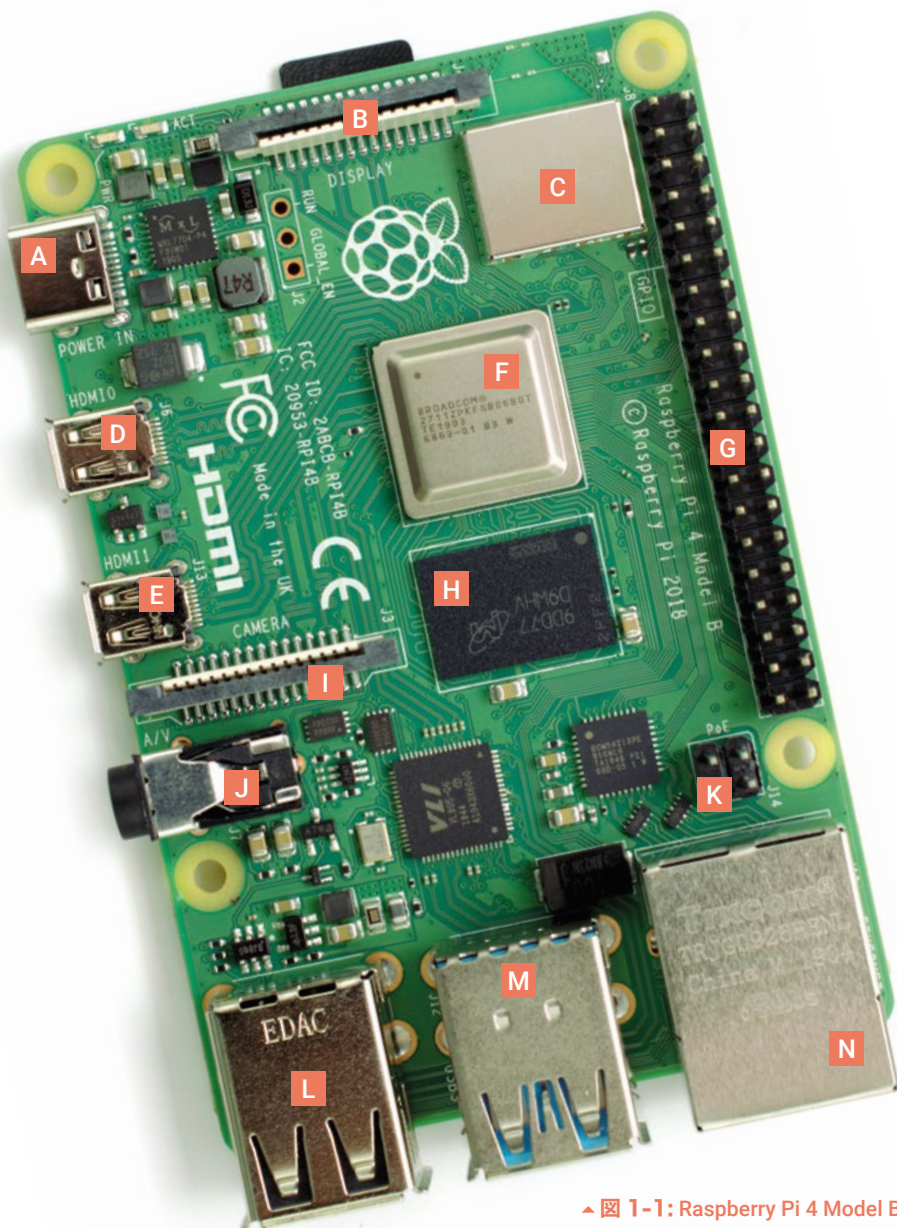
標準的な Raspberry Pi は、内部の仕組みが見えない従来のコンピューターとは異なり、すべてのコンポーネント、ポート、機能が見える構造になっています。必要に応じて、別売りのケースを購入してボードを保護することもできます。こうした構造により、コンピューターの各パーツの役割を理解することができます。また、周辺機器と呼ばれる外部パーツを接続した場合の仕組みについても、簡単に理解することができます。



- A USB Type-C 電源ポート
- B DSI ディスプレイポート
- C 無線/Bluetooth
- D micro HDMI 0 ポート
- E micro HDMI 1 ポート

- F システムオンチップ
- G GPIO
- H RAM
- I CSI カメラポート
- J 3.5mm AV ジャック

- K PoE
- L USB 2.0 ポート
- M USB 3.0 ポート
- N イーサネットポート



▲ 図 1-1: Raspberry Pi 4 Model B

図 1-1 は、上から見た Raspberry Pi 4 Model B の写真です。このガイドを参照しながら Raspberry Pi を使用する場合は、この写真と同じ向きで使用するようにしてください。向きが違っていると、GPIO ヘッダーなどを使用する場合に混乱する可能性があります (GPIO ヘッダーの詳細については、「第 6 章: Scratch と Python を使って物理的コンピューティングに挑戦しよう」を参照)。

小さなボードにいろいろなものが詰め込まれているように見えるかもしれませんが、Raspberry Pi の仕組みは非常にシンプルです。最初に、デバイスを起動する内部のコンポーネントを見てみましょう。

Raspberry Pi のコンポーネント

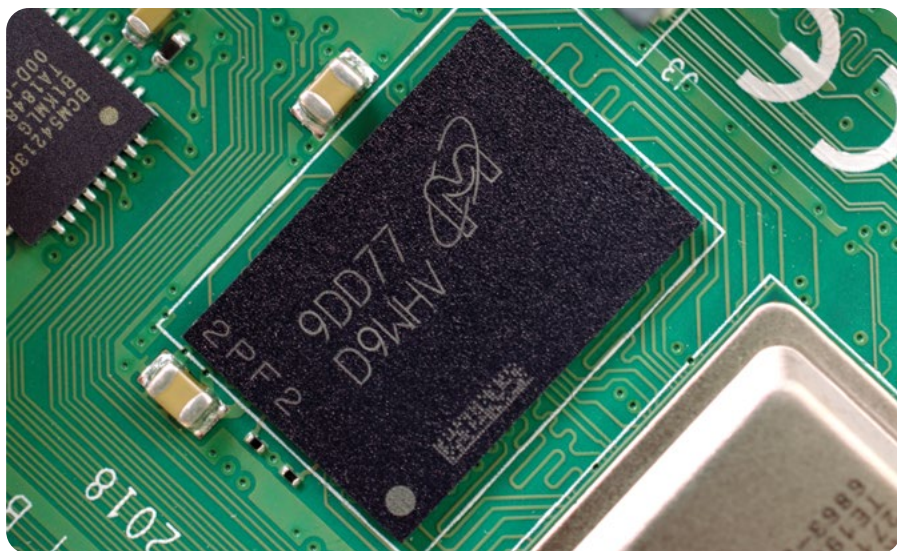
他のコンピューターと同様に、Raspberry Pi はさまざまなコンポーネントから構成されています。各コンポーネントには、コンピューターを稼働させるための役割があります。最初に、最も重要なコンポーネントを紹介します。それは、ボード上部の中央に配置されている「システムオンチップ (SoC)」です。金属製のカバーが取り付けられています (図 1-2)。



▲ 図 1-2: Raspberry Pi のシステムオンチップ (SoC)

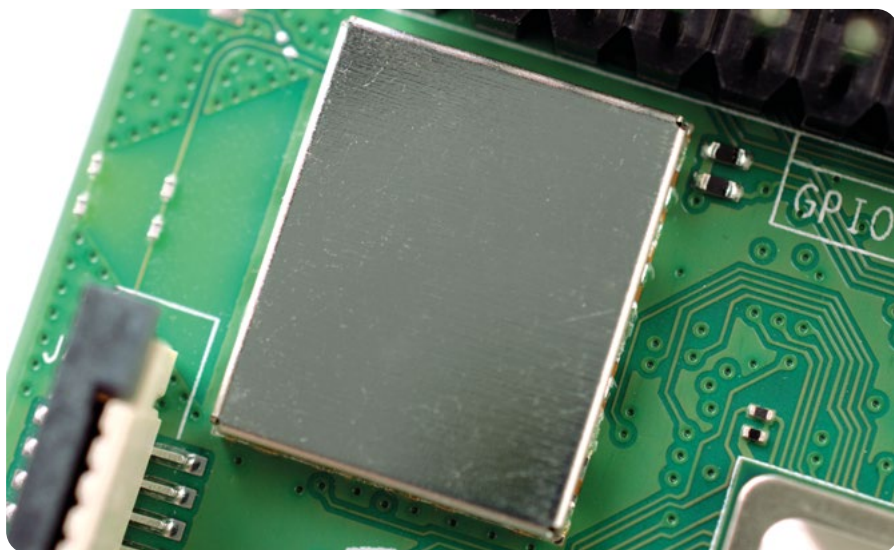
システムオンチップの金属カバーの内部には、集積回路と呼ばれるシリコンチップが搭載されています。このシリコンチップに、Raspberry Pi システムの核心部分が組み込まれています。たとえば、コンピューターの頭脳である中央処理装置 (CPU) や、画像を処理するためのグラフィックスプロセッシングユニット (GPU) などが組み込まれています。

CPU を稼働させるためにはメモリーが必要になりますが、SoC のすぐ下にある黒くて小さな四角形のチップ (プラスチックカバーが付いています) がメモリーです (図 1-3)。これが、Raspberry Pi のランダムアクセスメモリー (RAM) です。Raspberry Pi 上で行った操作は RAM に記録されます。保存操作を実行した場合のみ、作業内容が microSD カードに書き込まれます。Raspberry Pi の RAM は揮発性メモリーで、microSD カードは不揮発性メモリーです。Raspberry Pi の電源を切ると、揮発性 RAM 上の記録が消去されますが、不揮発性メモリーである microSD カードの記録が消去されることはありません。



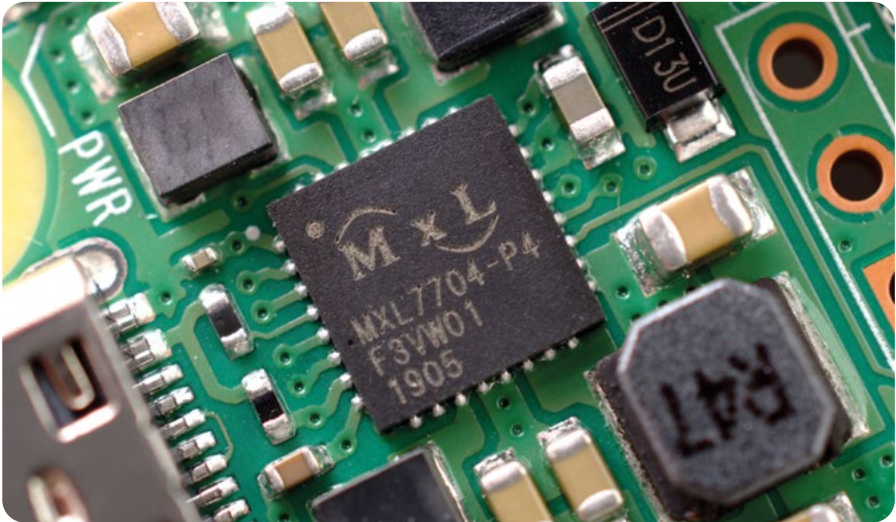
▲ 図 1-3: Raspberry Pi のランダムアクセスメモリー (RAM)

ボードの右上には、無線 コンポーネントを覆う金属製のカバーがあります (図 1-4)。これは、Raspberry Pi と無線デバイス間で通信を行うためのコンポーネントです。この無線コンポーネントは、2 つのコンポーネントとして機能します。1 つは、コンピューターネットワークに接続するための WiFi、もう 1 つは、マウスなどの周辺機器に接続して近くのスマートデバイス (センサーやスマートフォンなど) に対してデータを送受信するための Bluetooth です。



▲ 図 1-4: Raspberry Pi の無線コンポーネント

ボードの下部 (中央部の USB ポートの上) に、プラスチックカバーが付いた小さな黒いチップがあります。これは、4 つの USB ポートの動作を制御する USB コントローラー です。その横にさらに小さなチップがありますが、これは Raspberry Pi のイーサネットネットワークポートの動作を制御するネットワークコントローラー です。これらのチップよりもさらに小さなチップが、ボードの左上 (USB Type-C の電源コネクタの右横) にあります (図 1-5)。これは電源管理集積回路 (PMIC) と呼ばれるもので、この回路により、マイクロ USB ポートから流れてくる電力が、Raspberry Pi を稼働させるために必要な電力に変換されます。



▲ 図 1-5: Raspberry Pi の電源管理集積回路 (PMIC)

ここまでの説明は少し難しいかもしれませんが、ボード上の各コンポーネントの役割や配置を詳しく覚えなくても、Raspberry Pi を使用する際に問題はないため、心配する必要はありません。

Raspberry Pi のポート

Raspberry Pi には、様々なポートが付属しています。最初に、ボード下端の向かって左側と中央にある 4 つのユニバーサルシリアルバス (USB) ポート を見てみましょう (図 1-6)。これらのポートを使用して、キーボード、マウス、デジタルカメラ、フラッシュドライブなど、USB に対応した周辺機器を Raspberry Pi に接続することができます。技術的な話をすると、これらの USB ポートは種類が違います。内部に黒い部分がある方は USB 2.0 規格に準拠した USB 2.0 ポートで、内部に青い部分がある方は USB 3.0 規格に準拠した高速な USB 3.0 ポートです。



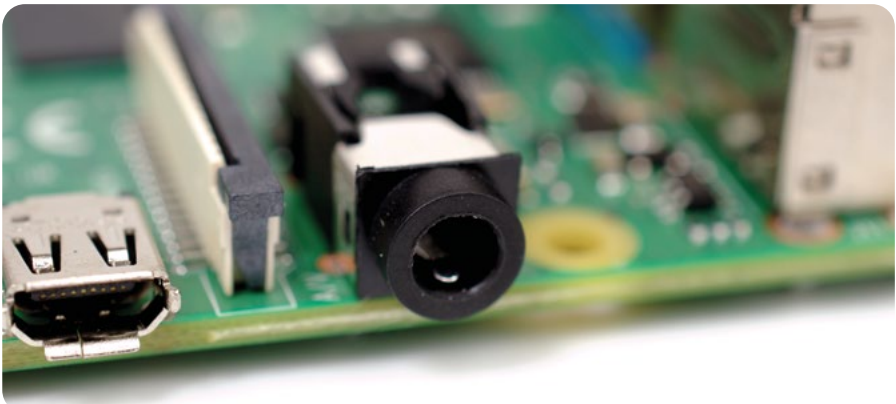
▲ 図 1-6: Raspberry Pi の USB ポート

USB ポートの右側にはイーサネットポート があります。これは、ネットワークポート ともいいます (図 1-7)。RJ45 と呼ばれるコネクタが端についたケーブルをこのポートに差し込むと、有線コンピュータネットワークに Raspberry Pi を接続することができます。イーサネットポートをよく見ると、ポートの下部に2つの LED があることがわかります。これらの LED により、接続状態を確認することができます。



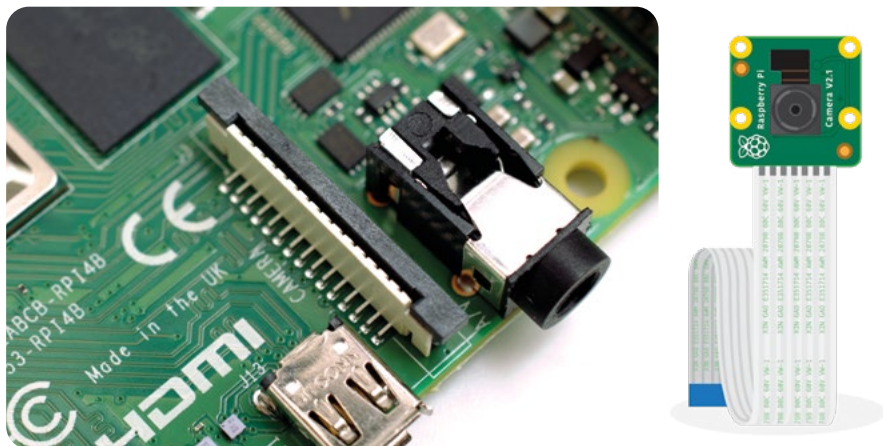
▲ 図 1-7: Raspberry Pi のイーサネットポート

ボードの左端 (USB ポートの上) には、3.5mm の AV (audio-visual) ジャック があります (図 1-8)。これはヘッドフォンジャック とも呼ばれ、このジャックにヘッドフォンを接続できますが、アンプスピーカーに接続すると、ヘッドフォンよりも音質が高くなります。この 3.5mm の AV ジャックには、隠れた機能があります。それは、チップ - リング - リング - スリーブ (TRRS) アダプターという特殊なケーブルを使用して、音声信号だけでなく映像信号も送信するという機能です。このジャックにプラグを差し込んで、コンポジット映像信号 に対応している TV、プロジェクター、ディスプレイに接続することができます。



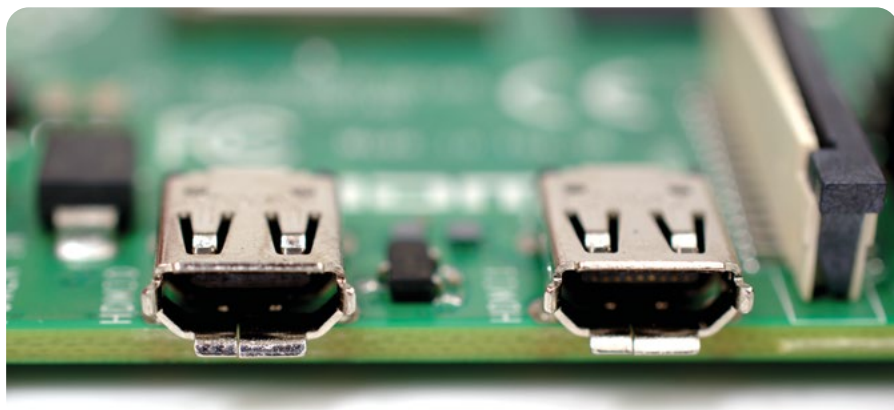
▲ 図 1-8: Raspberry Pi の 3.5mm AV ジャック

3.5 mm AV ジャックのすぐ上に、プラスチック製のフラップが付いた変わった形の引き上げ式カメラコネクタがあります。これは、カメラシリアルインターフェイス (CSI) とも呼ばれます (図 1-9)。このコネクタにより、Raspberry Pi 専用のカメラモジュールを使用することができます (このカメラモジュールについては、「第 8 章: Raspberry Pi の Camera Module を使ってみよう」で詳しく説明します)。



▲ 図 1-9: Raspberry Pi のカメラコネクタ

カメラコネクタの上 (ボードの左端) には、2 つの micro HDMI (High Definition Multimedia Interface) ポートがあります (図 1-10)。これらのポートは、ゲームコンソール、セットトップボックス、TV で見られるコネクタの小型版です。HDMIの正式名「高精細度マルチメディアインターフェイス」が表すとおり、HDMI ポートは、音声信号と映像信号の両方を非常に高い品質で送信します。これらのポートを使用して、Raspberry Pi を最大 2 台のディスプレイ装置 (コンピューターのモニター、TV、プロジェクター) に接続することができます。



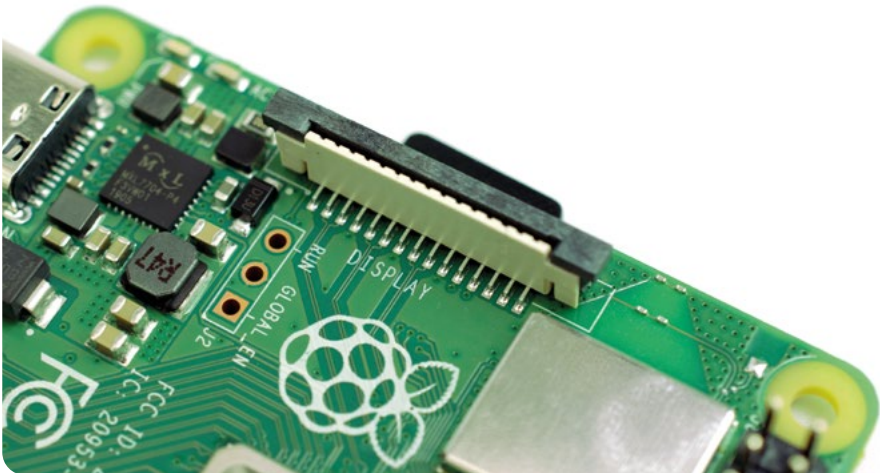
▲ 図 1-10: Raspberry Pi の micro HDMI ポート

HDMI ポートの上には、USB Type-C 電源ポートがあります (図 1-11)。このポートを使用して、Raspberry Pi を電源に接続します。USB Type-C ポートは、スマートフォンやタブレットなど、多くのポータブルデバイスに付いているポートです。一般的なモバイル充電器を使用して Raspberry Pi に電力を供給することもできますが、安定した動作のために、Raspberry Pi 専用の USB Type-C 電源を使用することをお勧めします。

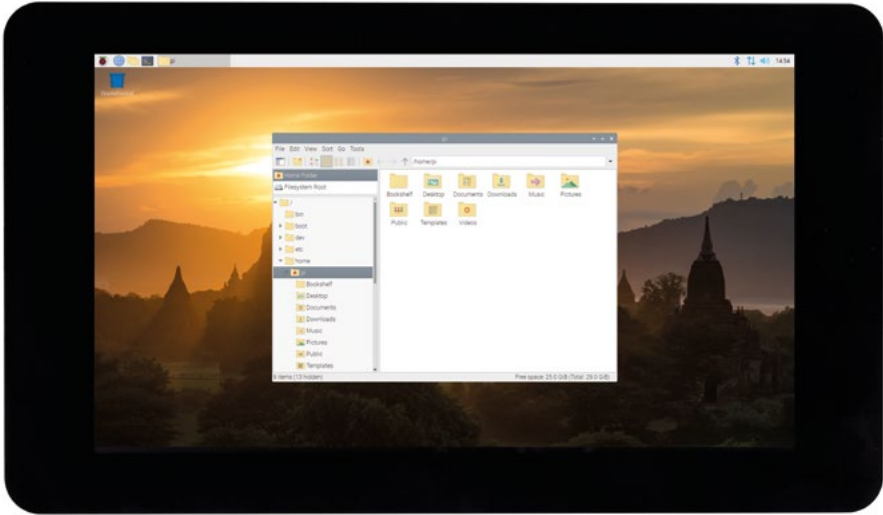


▲ 図 1-11: Raspberry Pi の USB Type-C 電源ポート

ボードの最上部にも、変わった形のコネクタがあります (図 1-12)。一目見ただけでは、カメラコネクタとまったく同じように見えます。しかし、実際の機能はまったく違います。これは、ディスプレイコネクタ またはディスプレイシリアルインターフェイス (DSI) と呼ばれるもので、Raspberry Pi のタッチディスプレイ専用設計されています (タッチディスプレイについては、図 1-13)。

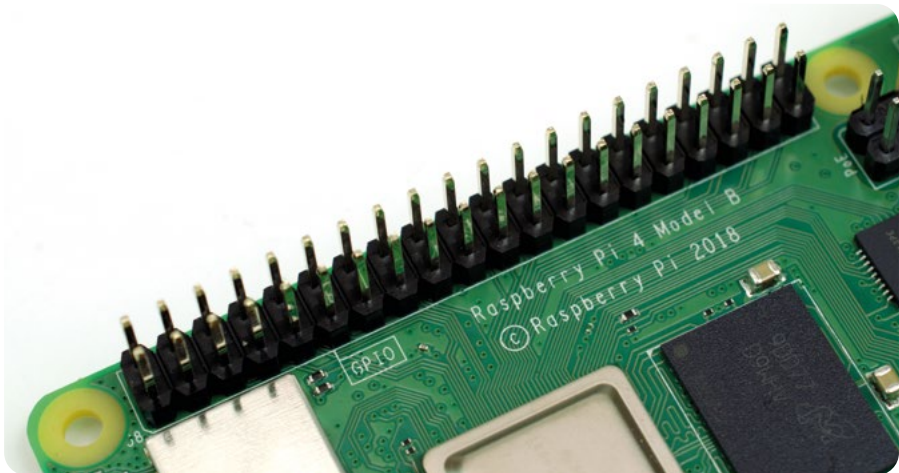


▲ 図 1-12: Raspberry Pi のディスプレイシリアルインターフェイス (DSI)



▲図 1-13: Raspberry Pi のタッチディスプレイ

ボードの右端には 40 本の金属ピンがあり、20 本のピンが 2 列に分かれて配置されています (図 1-14)。これは GPIO (汎用入出力) ヘッダーと呼ばれるものです。Raspberry Pi はこの GPIO ヘッダーを使用して、LED、ボタン、温度センサー、ジョイスティック、パルスレイトモニターなど、さまざまなハードウェアと通信を行います。GPIO ヘッダーについては、「第 6 章: Scratch と Python を使って物理的コンピューティングに挑戦しよう」で詳しく説明します。GPIO ヘッダーの左下には、4 本のピンが配置された小さなヘッダーがあります。これは、USB Type-C ではなくネットワーク接続から電力を受け取るために、オプションのアドオンである Power over Ethernet (PoE) HAT を接続する際に使用されるものです。



▲図 1-14: Raspberry Pi の GPIO ヘッダー

最後にもう 1 つだけ Raspberry Pi にポートがあるのですが、このポートを上から見ることはできません。ボードを裏返すと、ディスプレイコネクタの反対側に microSD カードコネクタ があることがわかります (図 1-15)。これは Raspberry Pi のストレージで、ここに挿入された microSD カードには自分で作成したファイルやインストールしたソフトウェア、Raspberry Pi の動作に必要なオペレーティングシステムがすべて保存されます。



▲ 図 1-15: Raspberry Pi の microSD カードコネクタ



▲ 図 1-16: Raspberry Pi 400 の一体型キーボード

Raspberry Pi 400

Raspberry Pi 400 のコンポーネントは Raspberry Pi 4 と同じで、それらをキーボードの筐体の中に組み込んであります。この構造によって、コンポーネントをキーボードの筐体で保護できるだけでなく、占有スペースを少なくして、机とケーブルをスッキリさせられるメリットがあります。

システムオンチップやメモリーなど、Raspberry Pi 400 は Raspberry Pi 4 と同じ主要コンポーネントで構成されています。これらのコンポーネントを外から見るとはできませんが、間違いなくキーボードに搭載されています。それでは、キーボードの各部を見ていきましょう (図 1-16)。キーボードの右上隅には、3 つの LED があります。Num Lock キーを押すと、左端の LED が点灯します。Num Lock キーを使用すると、通常のキーボードのテンキーと同様に、いくつかのキーを切り替えることができます。Caps Lock キーを押すと、中央の LED が点灯します。Caps Lock キーで、大文字と小文字を切り替えることができます。Raspberry Pi 400 の電源を入れると、右端の LED が点灯します。

Raspberry Pi 400 の背面にはポートがあります (図 1-17)。後ろから見て左端にあるポートは、GPIO ヘッダーです。これは 17 ページで説明したものと同一ヘッダーですが、位置が逆になっています。最初のピンであるピン 1 が右上に配置され、最後のピンであるピン 40 が左下に配置されています。GPIO ヘッダーについては、「第 6 章: Scratch と Python を使って物理的コンピューティングに挑戦しよう」で詳しく説明します。



▲ 図 1-17: Raspberry Pi 400 背面のポート

GPIO ヘッダーの横には、microSD カードスロットがあります。このスロットに microSD カードを挿入すると、そのカードが Raspberry Pi 400 のストレージとして機能します。オペレーティングシステム、アプリケーション、データが microSD カードに保存されます。Raspberry Pi 400 には、microSD カードが事前に挿入されています。カチッと音がするまでカードをゆっくり押し引き抜くと、カードを取り外すことができます。カードを元に戻す場合は、光沢のある金属部分を下に向けて、カチッと音がするまでゆっくりとスロットに挿入します。

microSD カードスロットの横にある 2 つのポートは、モニターや TV などのディスプレイ装置に接続するための micro HDMI ポートです。Raspberry Pi 4 と同様に、Raspberry Pi 400 でも最大 2 台のディスプレイ装置を接続することができます。micro HDMI ポートの横には、Raspberry Pi の電源アダプターもしくは互換性のある USB 電源に接続するための USB Type-C 電源ポートがあります。

その横にある 2 つの青いポートは、USB 3.0 ポートです。これらのポートにより、ソリッドステートドライブ (SSD)、メモリスティック、プリンターなどのデバイスに対して高速で通信することができます。その横にある白いポートは、USB 3.0 よりも低速な USB 2.0 ポートです。付属の Raspberry Pi マウスをこのポートに接続することができます。

最後のポートは、ギガビットイーサネットポートです。内蔵の Wi-Fi 無線ネットワークの代わりに、このポートにネットワークケーブルを接続すると、Raspberry Pi 400 をネットワークに接続することができます。Raspberry Pi 400 をネットワークに接続する方法については、「第 2 章: Raspberry Pi をセットアップしよう」で詳しく説明します。

第 2 章

Raspberry Pi を セットアップしよう

この章では、Raspberry Pi に重要なコンポーネントを接続して Raspberry Pi をセットアップする方法について説明します

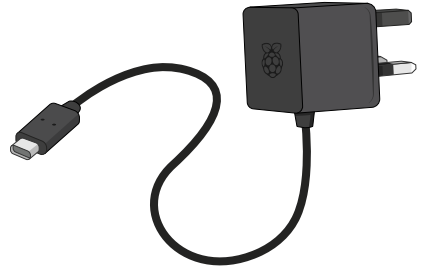


Raspberry Pi は、可能な限り簡単にセットアップできるように設計されていますが、ほかのコンピューターと同様に、周辺機器と呼ばれるさまざまな外部コンポーネントが必要になります。Raspberry Pi の場合、一般的な密閉型コンピュータとは異なり、簡単に回路基板を見ることができます。Raspberry Pi の回路基板を見ると、セットアップが複雑そうだと感じるかもしれませんが、心配する必要はありません。このガイドの手順に従えば、10 分もかからずに Raspberry Pi をセットアップして使用することができます。

Raspberry Pi デスクトップキットまたは Raspberry Pi 400 を購入した場合は、セットアップに必要なものがほぼすべて揃っています。準備する必要があるのは、セットトップボックス、ブルーレイプレイヤー、ゲームコンソールで使用するタイプの HDMI コネクターに対応したコンピューターモニターまたは TV だけです。コンピューターモニターや TV に接続すれば、Raspberry Pi の動作を確認することができます。

アクセサリーが付属していない Raspberry Pi を購入した場合は、コンピューターモニターまたは TV のほかに、以下のものがようになります。

- **USB 電源** - USB Type-C コネクタが付いている定格 3 アンペア (3 A) の 5V 電源を使用してください。Raspberry Pi の急速な電力需要の変化に対応できる Raspberry Pi 専用電源を使用することをお勧めします。



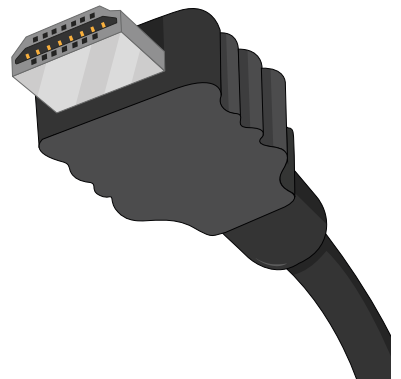
- **NOOBS がプリインストールされている microSD カード** - microSD カードは、Raspberry Pi のストレージとして機能します。システム上のすべてのファイルやソフトウェアだけでなく、オペレーティングシステム自体も microSD カードに保存されます。カード容量の最低要件は 8GB ですが、余裕を持って 16GB のカードを使用することをお勧めします。NOOBS (New Out-Of-Box Software: すぐに使用できる新しいソフトウェア) がプリインストールされているカードを使用すると、作業の手間を省くことができます。何もインストールされていないカードにオペレーティングシステム (OS) をインストールする方法については、**付録 A** を参照してください。



- **USB キーボードと USB マウス** - Raspberry Pi を操作する場合は、キーボードとマウスを使用します。USB コネクタが付いていれば、有線か無線にかかわらず、ほぼすべてのキーボードとマウスを Raspberry Pi で使用することができます。ただし、カラフルなライトが点灯するゲーミングキーボードの場合、消費電力が大きくなるため、動作が不安定になる可能性があります。



- **micro HDMI ケーブル** - このケーブルにより、音声と画像が Raspberry Pi から TV やモニターに送信されます。このケーブルの端部には、Raspberry Pi 用の micro HDMI コネクタがあり、反対側の端部には、ディスプレイ用のフルサイズの HDMI コネクタがあります。HDMI アダプターや標準的なフルサイズの HDMI ケーブルに対して micro HDMI コネクタを使用することもできます。HDMI ソケットが付いていないモニターを使用する場合は、DVI-D アダプター、DisplayPort アダプター、または VGA アダプター用の micro-HDMI を購入してください。コンポジットビデオ端子や SCART ソケットが付いている古いタイプの TV を使用する場合は、3.5mm の TRRS オーディオ/ビデオケーブルを使用してください。



Raspberry Piは、電流による短絡が発生する可能性のある金属面に置かない限り、ケースがなくても安全に使用することができます。ただし、オプションのケースを使用すれば、安全性がさらに高くなります。デスクトップキットには Raspberry Pi 専用ケースが付属していますが、サードパーティ製のケースを使用してもかまいません。

無線ネットワーク (WiFi ネットワーク) ではなく有線ネットワークで Raspberry Pi を使用する場合は、ネットワークケーブルも必要になります。ネットワークケーブルを使用する場合は、ケーブルの端部をネットワークのスイッチまたはルーターに接続する必要があります。Raspberry Pi に組み込まれている無線ネットワークを使用する場合は、ケーブルは必要ありませんが、無線ネットワークの名前、キー、またはパスフレーズを知っておく必要があります。



Raspberry Pi 400 のセットアップ

以下の手順は、Raspberry Pi4 または RaspberryPi ファミリーの別のペアボードメンバーをセットアップするための手順です。Raspberry Pi 400 のセットアップ手順は、32 ページ以降に記載されています。



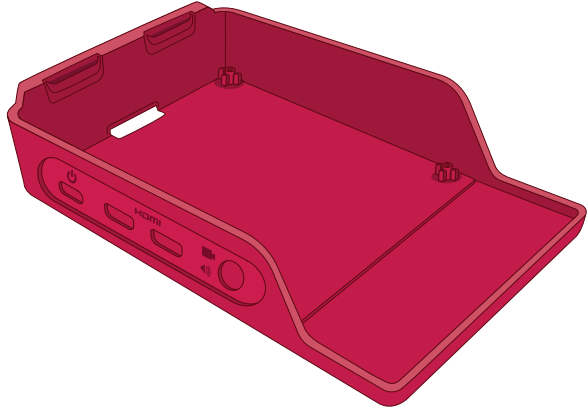
ハードウェアのセットアップ

最初に、Raspberry Pi を箱から取り出します。Raspberry Pi は堅牢なハードウェアですが、乱暴に扱えば破損することもあります。ボードを持つ場合は、平面ではなく端を持つようにしてください。その際、隆起した金属ピンの部分に特に注意してください。これらの金属ピンが曲がってしまうと、アドオンボードなどのハードウェアを追加できなくなるだけでなく、最悪の場合は短絡が発生して RaspberryPi が破損する可能性があります。

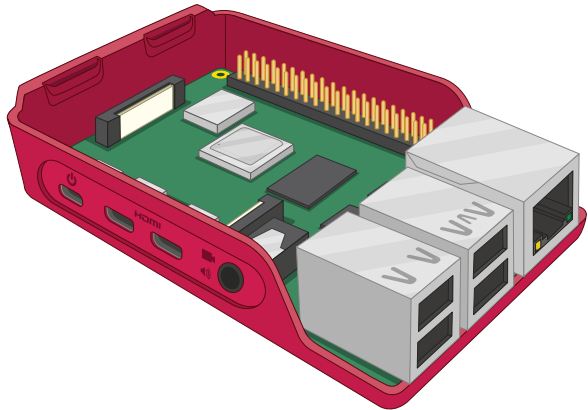
各ポートの場所と役割については、「**第 1 章: Raspberry Pi を知ろう**」を参照してください。ケースを組み立てる

最初に、Raspberry Pi をケースに収納します (ケースを使用する場合)。Raspberry Pi 専用ケースを使用する場合は、ケースを赤いベースの部分と白いカバーの部分に分割します。

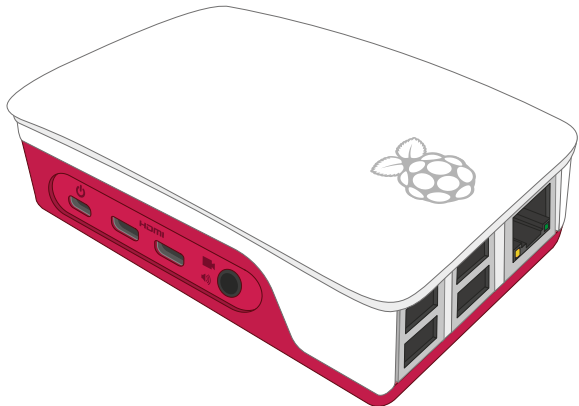
- 1 閉じている端部が左側、開いている端部が右側になるようにベースを置きます。



- 2 microSD カードが挿入されていない Raspberry Pi の USB ポートとイーサネットポートの部分を持って少し傾け、USB Type-C 電源ポート、2 つの micro HDMI ポート、3.5mm の AV ジャックがベース側部のそれぞれの開口部にはまるように押し込み、Raspberry Pi が水平になるまでゆっくりと下ろします。

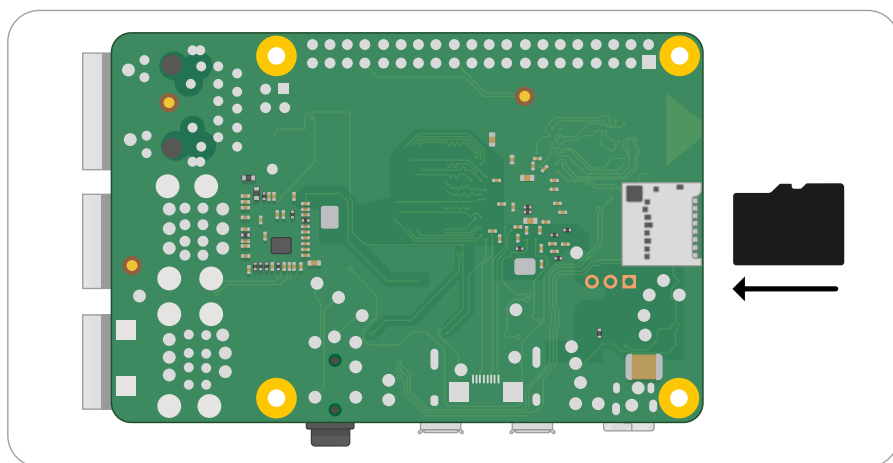


- 3 白いカバーを持ち、カバーの左側にある 2 つのクリップをベースの左側にある 2 つの穴 (microSD カードスロットの上部) にはめ込みます。位置が正しいことを確認し、カバーの右側 (USB ポートの上部) をカチッと音がするまで押し込みます。

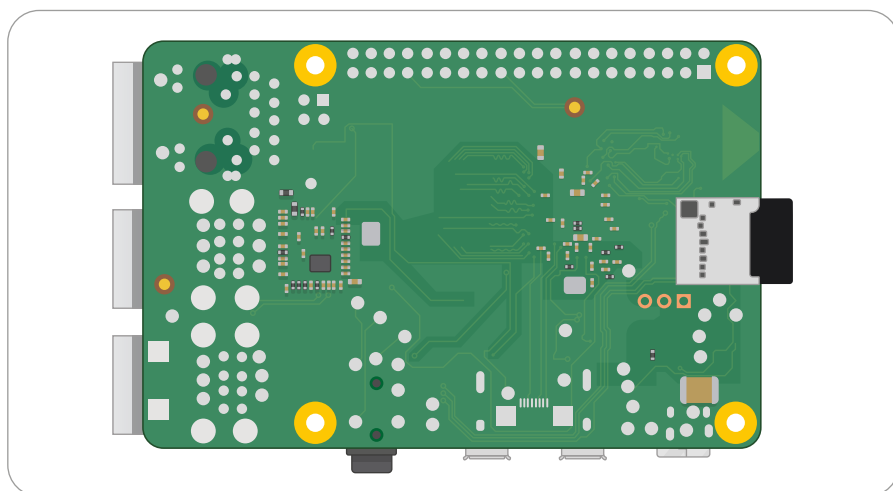


microSD カードを挿入する

Raspberry Pi のストレージとして機能する microSD カードを挿入するには、Raspberry Pi を裏返し、カードのラベルを Raspberry Pi の反対側に向けた状態で、カードを microSD カードスロットに挿入します。カードは一方方向でしか挿入できないようになっています。カードにあまり力をかけなくても、スロットに差し込むことができます。



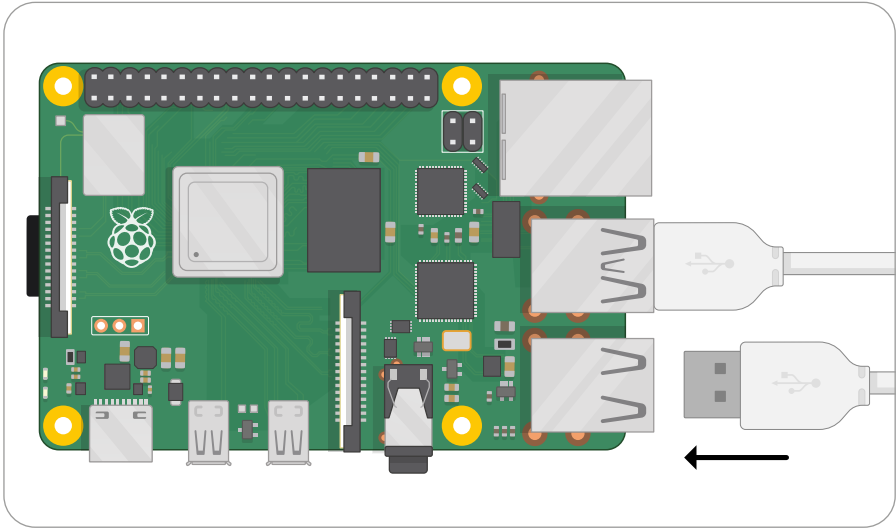
microSD カードをカードスロットに挿入すると、定位置で止まります。その際、カチッという音はしません。



カードを取り出す場合は、カードの端を持ってゆっくりと引き出してください。古いモデルの Raspberry Pi の場合は、最初にカードを軽く押してロックを解除する必要がありますが、Raspberry Pi 3 と 4 では、その必要はありません。

キーボードとマウスを接続する

キーボードの USB ケーブルを、Raspberry Pi の 4 つの USB ポート (USB 2.0 ポートと USB 3.0 ポート) のいずれかに接続します。Raspberry Pi 専用キーボードを使用する場合は、キーボードの背面にマウス用の USB ポートがあります。専用キーボードを使用しない場合は、マウスの USB ケーブルを Raspberry Pi の別の USB ポートに接続します。



キーボードとマウス用の USB コネクタは、あまり力をかけなくてもポートに接続することができます。少し力を入れてもコネクタをポートに接続できない場合は、コネクタの向きが間違っています。USB コネクタの向きが正しいことを確認してからポートに接続してください。

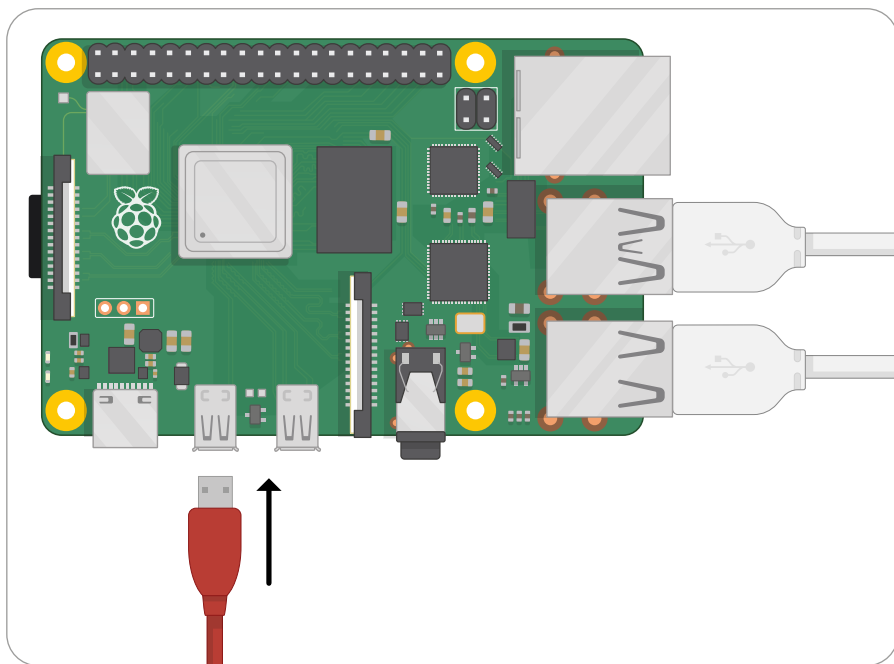


キーボードとマウス

キーボードとマウスは、計算処理において何を実行するかを Raspberry Pi に指示するための主要な手段です。ディスプレイは出力デバイスと呼ばれるのに対して、キーボードとマウスは入力デバイスと呼ばれます。

ディスプレイを接続する

micro HDMI ケーブルの小さな端部の方を、Raspberry Pi の USB Type-C ポートに近い方の micro HDMI ポートに差し込み、反対側の端部をディスプレイ (コンピューターのモニターまたは TV) に接続します。TV に複数の HDMI ポートがある場合は、そのポートの番号を確認し、TV の入力をその番号に切り替えて、Raspberry Pi のディスプレイになっているかどうかを確認します。ポート番号がわからなくても、心配する必要はありません。TV の入力を順に切り替えていけば、Raspberry Pi のディスプレイになっているポートが見つかります。

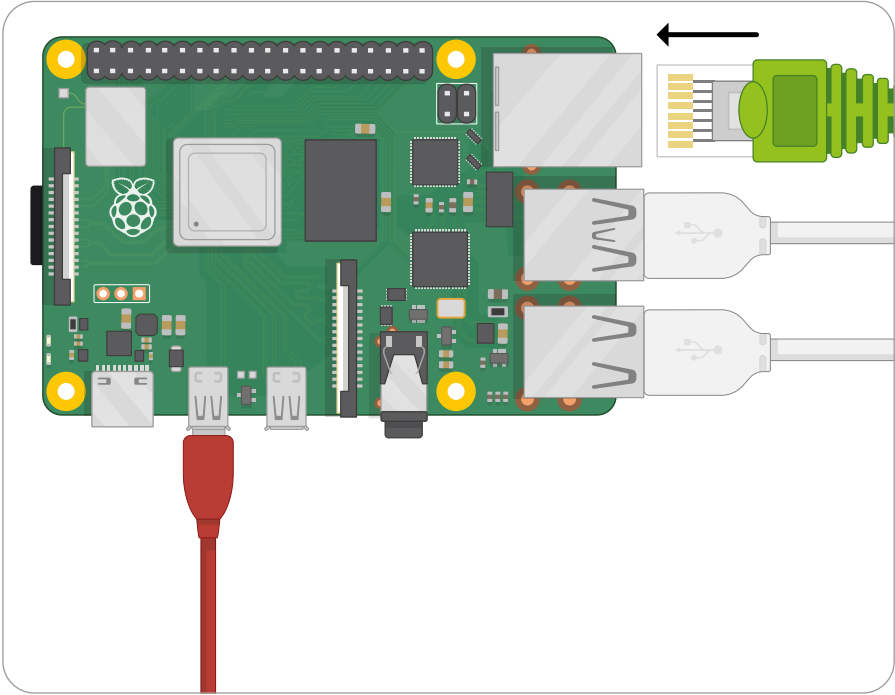


TV を接続

接続先の TV やモニターに HDMI コネクタが付いていない場合でも、Raspberry Pi を使用できないわけではありません。電子機器メーカーが製造している市販のアダプターケーブルを使用して Raspberry Pi の micro HDMI ポートを DVI-D アダプター、DisplayPort アダプター、または VGA アダプターに変換することにより、HDMI コネクタが付いていない TV やコンピューターモニターに Raspberry Pi を接続することができます。これらのアダプターを Raspberry Pi の micro HDMI ポートに接続し、適切なアダプターケーブルを TV やモニターに接続するだけです。接続先の TV やモニターにコンポジットビデオ端子または SCART 入力端子しかない場合は、3.5mm の TRRS アダプターケーブルと、コンポジットビデオ端子と SCART 端子をつなぐアダプターを購入して、3.5mm の AV ジャックに接続します。

ネットワークケーブルを接続する (オプション)

Raspberry Pi を有線ネットワークに接続する場合は、ネットワークケーブル (イーサネットケーブルとも呼ばれます) のプラスチッククリップを下に向けた状態で、カチッと音がするまで、ケーブルを Raspberry Pi のイーサネットポートに差し込みます。ケーブルを取り外す場合は、プラスチッククリップをプラグに向かって内側に押し込み、ケーブルをゆっくりとスライドさせて取り外します。

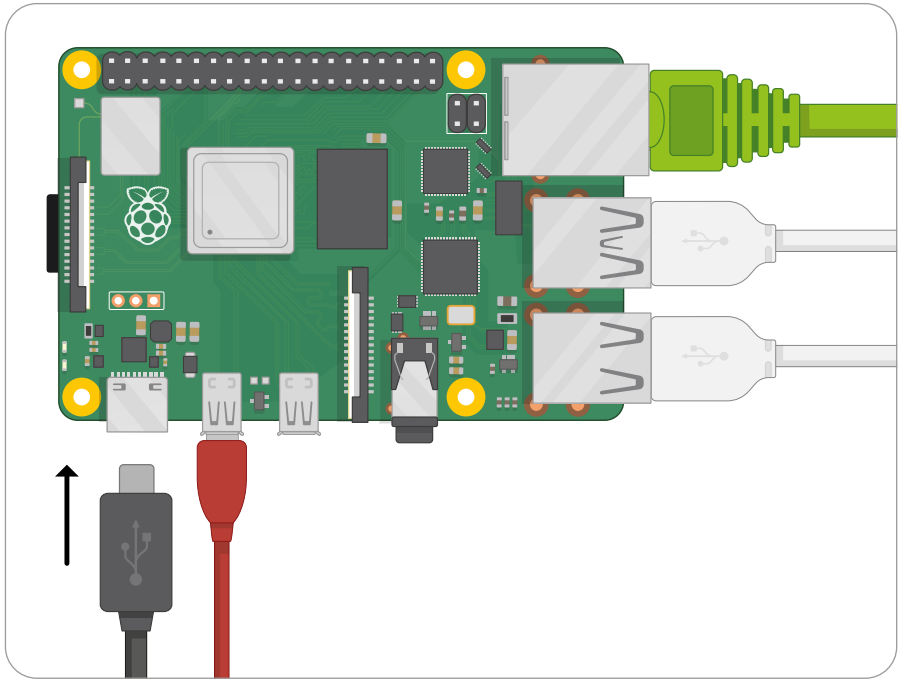


ネットワークケーブルの反対側の端部についても同様に、ネットワークハブ、ネットワークスイッチ、またはルーターに接続します。

電源を接続する

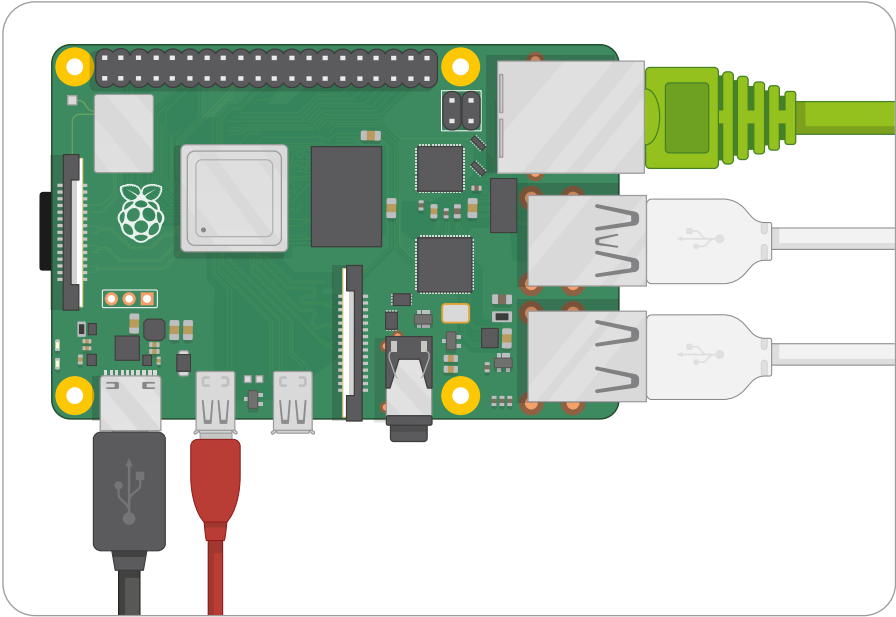
ハードウェアの最後のセットアップ手順として、Raspberry Pi を電源に接続します。この手順が完了すると、ソフトウェアのセットアップ準備が整います。Raspberry Pi には電源スイッチがないため、電源に接続するとすぐに通電します。

最初に、USB Type-C の電源ケーブルを Raspberry Pi の USB Type-C 電源ポートに接続します。電源ケーブルの向きはどちらでもかまいません。ゆっくりと電源ポートに差し込んでください。電源装置に取り外し可能なケーブルが接続されている場合は、そのケーブルの反対側の端部が電源装置の本体に接続されていることを確認してください。

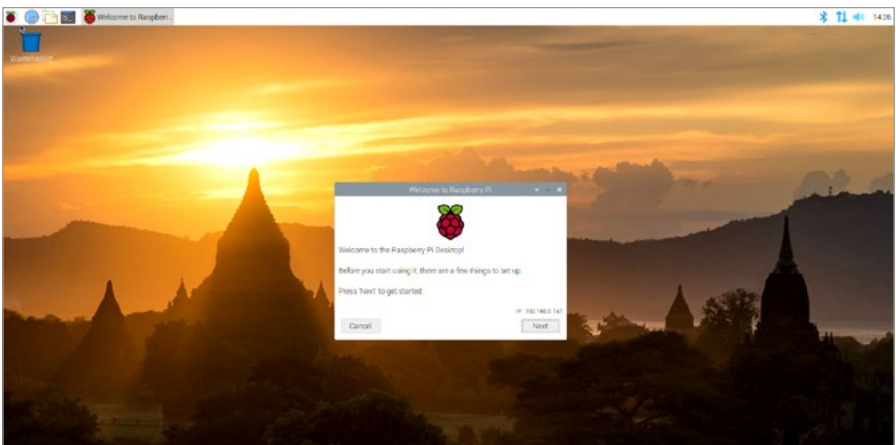


最後に、電源を主電源ソケットに接続してソケットのスイッチを入れます。すぐに Raspberry Pi が起動します。

これで、Raspberry Pi のセットアップが完了しました。



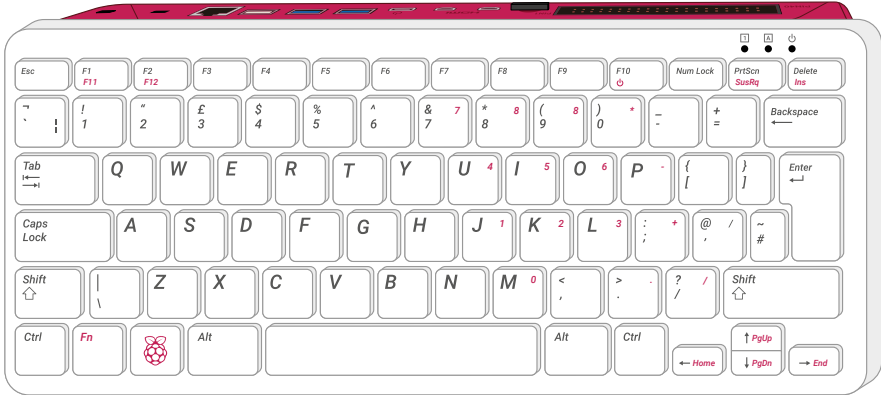
黒い画面の左上に 4 つの RaspberryPi ログが短時間表示されます。microSD カード用にソフトウェアのサイズが変更される場合は、青い画面が表示されることがあります。黒い画面が表示された場合は、数分間待ってください。Raspberry Pi を初めて起動すると、バックグラウンドでハウスキーピング処理が実行されます。しばらくすると、Raspberry Pi OS のデスクトップとセットアップウィザードが表示されます (図 2-1)。これで、オペレーティングシステムを設定する準備が整いました。これについては、オペレーティングシステムの背景については、「第3章: Raspberry Pi を使ってみよう」で詳しく説明します。



▲ 図 2-1: Raspberry Pi OS のデスクトップとセットアップウィザード

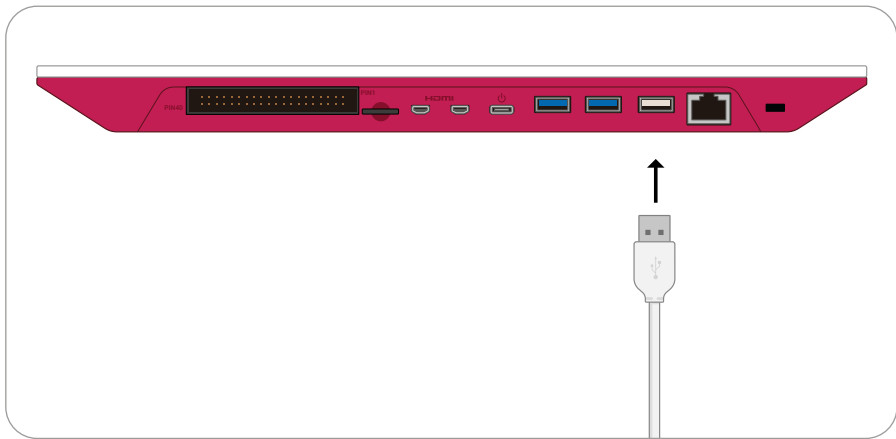
Raspberry Pi 400 のセットアップ

Raspberry Pi 4 とは異なり、Raspberry Pi 400 にはキーボードが内蔵されていて、microSD カードが事前に挿入されています。Raspberry Pi 400 を使用するには数本のケーブルを接続する必要がありますが、数分もあれば接続が完了します。



マウスを接続する

Raspberry Pi 400 のキーボードはすでに接続されているため、接続する必要のあるのはマウスだけです。マウスの端にある USB ケーブルを取り外し、Raspberry Pi 400 の背面にある3つのUSBポート (2.0 ポートと 3.0 ポート) のいずれかに差し込みます。2 つの高速ポート (USB 3.0 ポート) を空けておきたい場合は、白いポートにケーブルを差し込みます。

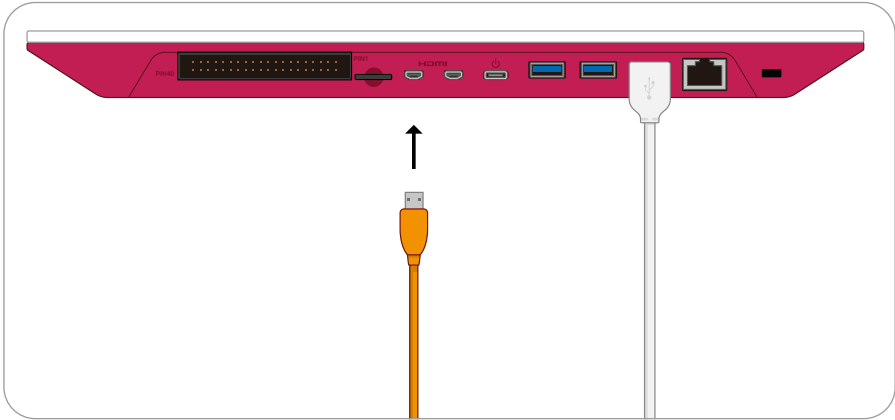


USB コネクタは、あまり力をかけなくてもポートに接続することができます。少し力を入れてもコネクタをポートに接続できない場合は、コネクタの向きが間違っています。USB コネクタの向きが正しいことを確認してからポートに接続してください。

ディスプレイを接続する

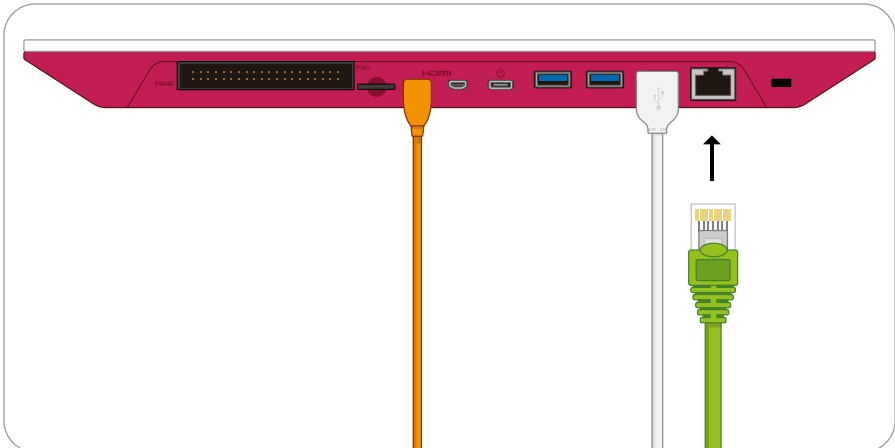
micro HDMI ケーブルの小さな端部の方を、Raspberry Pi 400 の microSD カードスロットに近い方の micro HDMI ポートに差し込み、反対側の端部をディスプレイ (コンピューターのモニターまたは TV) に接続します。TV に複数の HDMI ポートがある場合は、そのポートの番号を確認し、TV の入力をその番号に切り替えて、Raspberry Pi のディスプレイになっているかどうかを確認します。ポート番号がわからなくても、心配する必要はありません。

TV の入力を順に切り替えていけば、Raspberry Pi のディスプレイになっているポートが見つかります。



ネットワークケーブルを接続する (オプション)

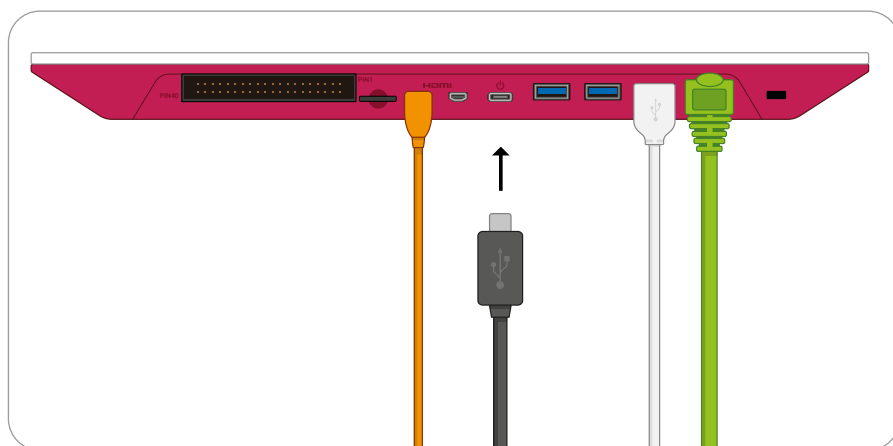
Raspberry Pi 400 を有線ネットワークに接続する場合は、ネットワークケーブル (イーサネットケーブルとも呼ばれます) のプラスチッククリップを上に向けた状態で、カチッと音がするまで、ケーブルを Raspberry Pi 400 のイーサネットポートに差し込みます。ケーブルを取り外す場合は、プラスチッククリップをプラグに向かって内側に押し込み、ケーブルをゆっくりとスライドさせて取り外します。



ネットワークケーブルの反対側の端部についても同様に、ネットワークハブ、ネットワークスイッチ、またはルーターに接続します。

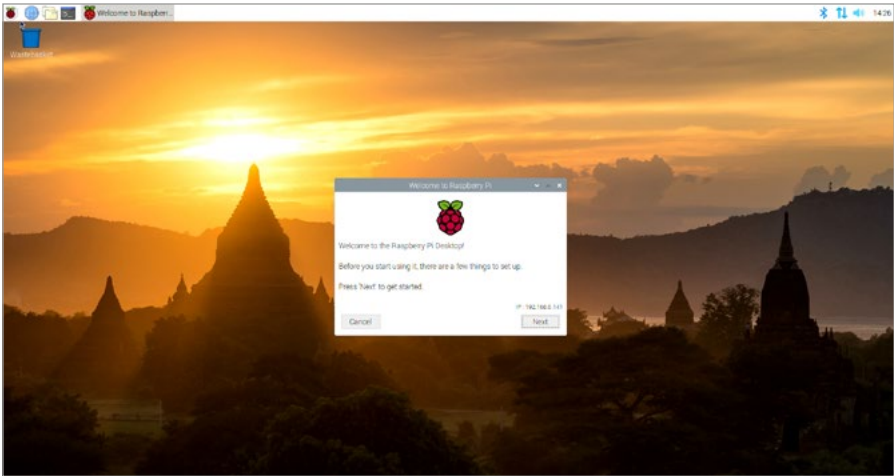
電源を接続する

ハードウェアの最後のセットアップ手順として、Raspberry Pi 400 を電源に接続します。この手順が完了すると、ソフトウェアのセットアップ準備が整います。Raspberry Pi 400 には電源スイッチがないため、電源に接続するとすぐに通電します。最初に、USB Type-C の電源ケーブルを Raspberry Pi の USB Type-C 電源ポートに接続します。電源ケーブルの向きはどちらでもかまいません。ゆっくりと電源ポートに差し込んでください。電源装置に取り外し可能なケーブルが接続されている場合は、そのケーブルの反対側の端部が電源装置の本体に接続されていることを確認してください。



最後に、電源を主電源ソケットに接続してソケットのスイッチを入れます。すぐに Raspberry Pi 400 が起動します。これで、Raspberry Pi 400 のセットアップが完了しました。

黒い画面の左上に 4 つの RaspberryPi ロゴが短時間表示されます。microSD カード用にソフトウェアのサイズが変更される場合は、青い画面が表示されることがあります。黒い画面が表示された場合は、数分間待ってください。Raspberry Pi を初めて起動すると、バックグラウンドでハウスキーピング処理が実行されます。しばらくすると、Raspberry Pi OS のデスクトップとセットアップウィザードが表示されます (図 2-2)。これで、オペレーティングシステムを設定する準備が整いました。これについては、オペレーティングシステムの背景については、「第3章: Raspberry Pi を使ってみよう」で詳しく説明します。

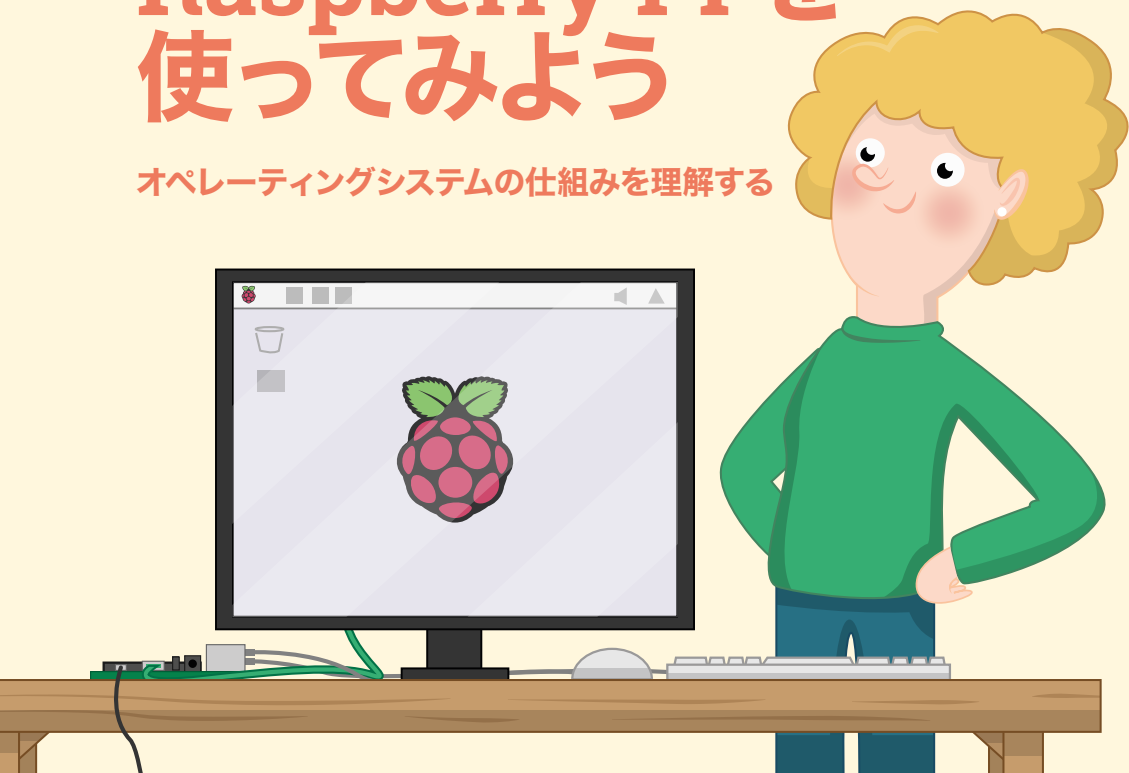


▲ 図 2-2: Raspberry Pi OS のデスクトップとセットアップウィザード

第3章

Raspberry Pi を使ってみよう

オペレーティングシステムの仕組みを理解する



Raspberry Pi では、さまざまなソフトウェアを稼働させることができます。コンピューターの核となるオペレーティングシステムもソフトウェアです。Raspberry Pi Foundation の公式オペレーティングシステムは、Raspberry Pi OS です。Debian Linux をベースとして開発されたこの OS は、Raspberry Pi 用にカスタマイズされていて、さまざまな機能がプリインストールされています。

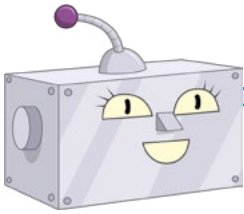
これまで Microsoft Windows や Apple macOS しか使ったことがなくても、心配する必要はありません。Raspberry Pi OS の動作は基本的に、ウィンドウ、アイコン、メニュー、ポインター (WIMP と総称します) の動作と同じなので、すぐに慣れることができます。バンドルされているいくつかのソフトウェアについては、次の章で説明します。

ウェルカムウィザード

初めて Raspberry Pi OS を起動すると、図 3-1 のようなウェルカムウィザードが表示されます。このウィザードに表示される手順に従い、用途に合わせて Raspberry Pi OS の設定 (構成と呼ぶ場合もあります) を変更することができます。



▲ 図 3-1: ウェルカムウィザード



ウィザードを閉じる

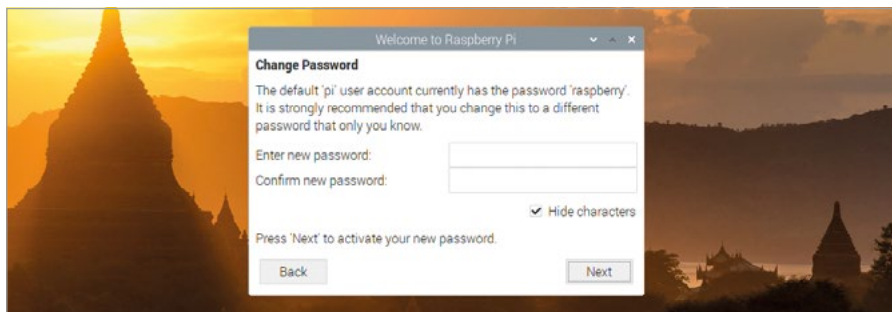
「Cancel」ボタンを押してウェルカムウィザードを閉じることもできますが、最初に表示される一連の情報を入力しないと、Raspberry Pi の一部の機能が動作しません (無線ネットワークなど)。

「Next」ボタンを押し、ドロップダウンボックスを順にクリックして、リストから国、言語、タイムゾーンを選択します (図 3-2)。US 配列のキーボードを使用する場合は、「Use US keyboard」チェックボックスをクリックします。デスクトップとプログラムを英語で表示する場合は、「Use English language」チェックボックスをクリックしてチェックマークを付けます。これらの操作が完了したら「Next」をクリックします。



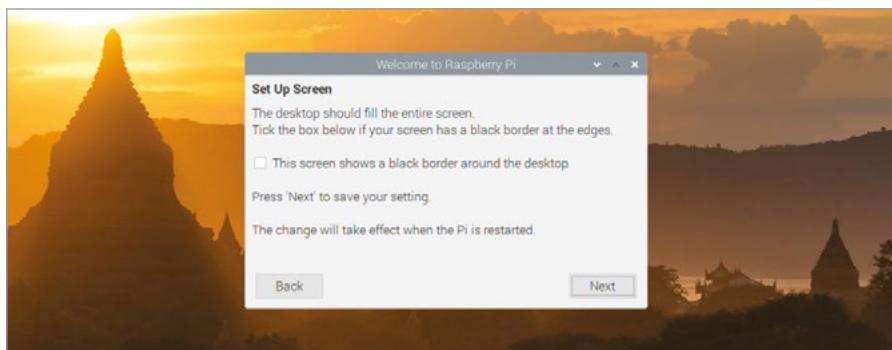
▲ 図 3-2: 言語などを選択する

次に、「pi」ユーザーのパスワードを変更するための画面が表示されます。デフォルトのパスワードは「raspberrry」ですが、セキュリティ上の理由により、新しいパスワードを設定することを強くお勧めします。新しいパスワードをボックスに入力します (図 3-3)。パスワードを入力したら「Next」をクリックします。



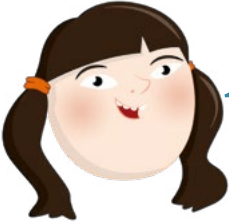
▲ 図 3-3: 新しいパスワードを設定する

次に、画面の周囲に黒い枠線を表示するかどうかを選択するための画面が表示されます (図 3-4)。Raspberry Pi のデスクトップ画面を TV やモニター全体に表示する場合は、チェックボックスを選択しないでください。Raspberry Pi のデスクトップ画面が TV やモニターよりも小さく、デスクトップ画面の周囲に黒い枠線を表示する場合は、チェックボックスをクリックしてください。操作が完了したら「Next」をクリックします。



▲ 図 3-4: チェックボックスはクリックしない (画面の周囲に黒い枠線を表示しない)

次に、リストから WiFi ネットワークを選択するための画面が表示されます (図 3-5)。マウスまたはキーボードを使用してネットワークリストをスクロールし、該当するネットワークを選択して「Next」をクリックします。無線ネットワークが保護されている場合 (実際に保護されている必要があります)、ネットワークのパスワード (事前共有キーと呼ばれることもあります) を入力するための画面が表示されます。このパスワードは通常、ルーターのカードまたはルーター本体に記載されています。「Next」をクリックすると、無線ネットワークに接続されます。無線ネットワークに接続しない場合は、「Skip」をクリックします。



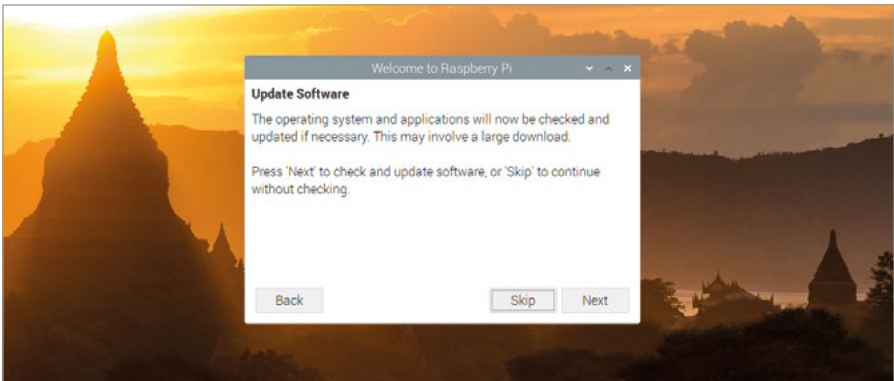
無線ネットワーク

組み込み無線ネットワークを使用できるのは、Raspberry Pi 3、Pi 4、Pi Zero W ファミリーのみです。これら以外の Raspberry Pi モデルで無線ネットワークを使用する場合は、USB WiFi アダプターが必要になります。



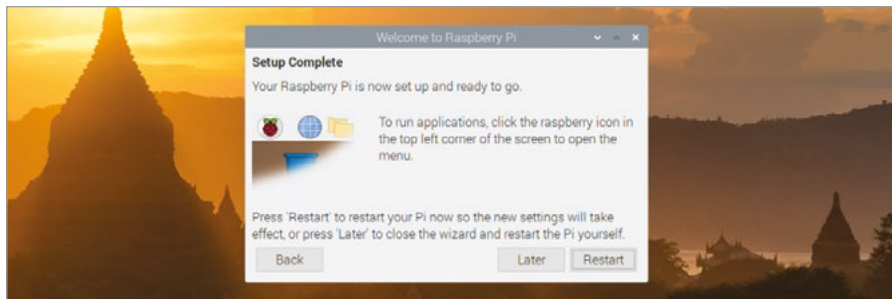
▲ 図 3-5: 無線ネットワークを選択する

次に、Raspberry Pi にインストールされているソフトウェア (Raspberry Pi OS を含む) のアップデートを確認してインストールするための画面が表示されます (図 3-6)。バグの修正、新機能の追加、パフォーマンスの改善を目的として、Raspberry Pi OS は定期的にアップデートされます。これらのアップデートをインストールする場合は「Next」をクリックし、インストールしない場合は「Skip」をクリックします。アップデートのインストールが完了するまで、数分かかる場合があります。アップデートのインストールが完了すると、「System is up to date」というメッセージを示すウィンドウが表示されます。このウィンドウが表示されたら、「OK」ボタンをクリックします。



▲ 図 3-6: アップデートを確認する

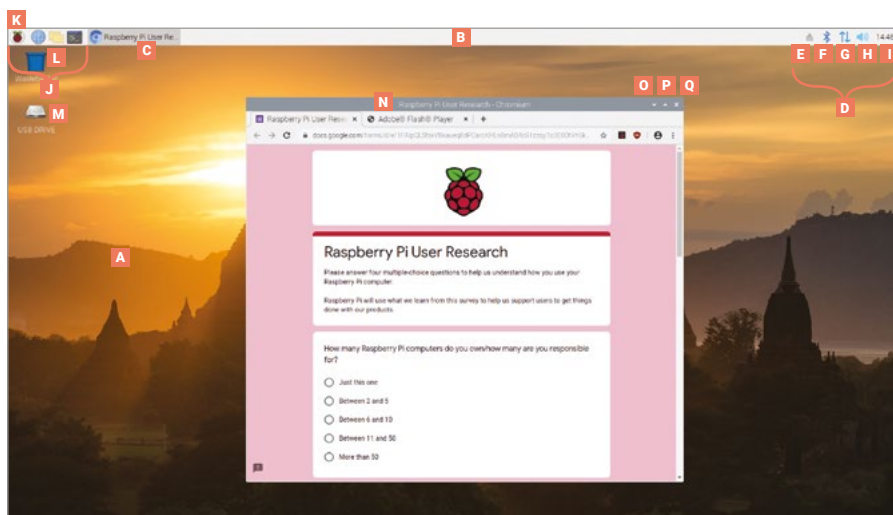
最後に、Raspberry Pi を再起動するための画面が表示されます (図 3-7)。ウェルカムウィザード上で行ったいくつかの設定変更は、Raspberry Pi を再起動するまで適用されません (再起動は「リポート」とも呼ばれます)。この画面が表示されたら、「Restart」ボタンを押して Raspberry Pi を再起動します。再起動後は、ウェルカムウィザードは表示されません。これで、Raspberry Pi を使用する準備が整いました。



▲ 図 3-7: Raspberry Pi を再起動する

デスクトップのナビゲーション

ほとんどの Raspberry Pi にインストールされている Raspberry Pi OS のバージョンは、「Raspberry Pi OS with desktop」というバージョンです。この OS のメインのグラフィカルユーザーインターフェイスは、図 3-8 のように表示されます。これがデスクトップ画面になります。このデスクトップ画面の大部分は、壁紙と呼ばれる画像で占められています (図 3-8 の A)。この壁紙に、各種のプログラムが表示されます。デスクトップ画面の上部には、タスクバー (B) が表示されます。このタスクバーで、各種のプログラムを読み込むことができます。読み込んだプログラムは、タスクバー内にタスク (C) として表示されます。

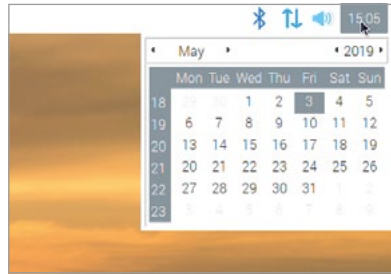


▲ 図 3-8: Raspberry Pi OS のデスクトップ画面

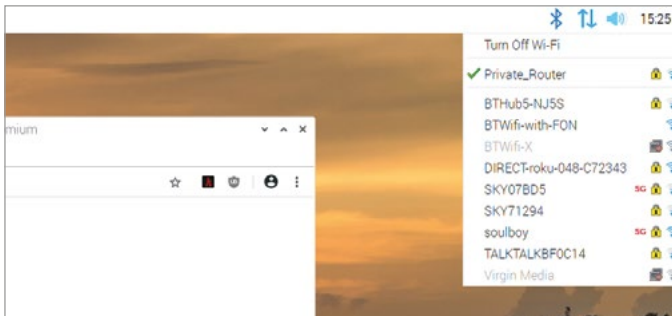
- | | | |
|-------------------------|---------------------------------------|-----------------------|
| A 壁紙 | H ボリュームアイコン | N ウィンドウのタイトルバー |
| B タスクバー | I 時計 | O 最小化アイコン |
| C タスク | J ランチャー | P 最大化アイコン |
| D システムトレイ | K メニューアイコン
(Raspberry アイコン) | Q 終了アイコン |
| E メディア取り外しアイコン | L ゴミ箱アイコン | |
| F Bluetooth アイコン | M リムーバブルドライブアイコン | |
| G ネットワークアイコン | | |

タスクバーの右側には、システムトレイ (D) が表示されます。USB メモリースティックなどのリムーバブルストレージが Raspberry Pi に接続されている場合、このリムーバブルメディアを表すアイコン (E) が表示されます。このアイコンをクリックすると、メディアを安全に取り外すことができます。右端には時計 (I) が表示されます。この時計をクリックすると、デジタルカレンダーが表示されます (図 3-9)。

▶ 図 3-9: デジタルカレンダー

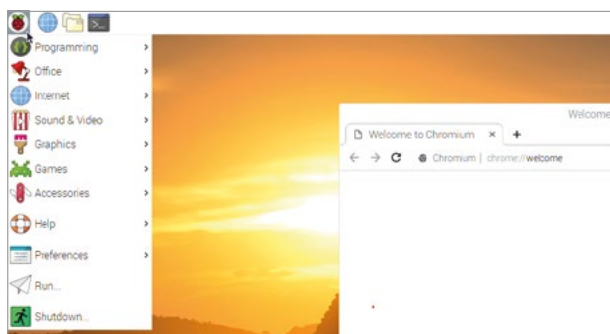


時計の横には、スピーカーアイコン (H) が表示されます。マウスの左ボタンでこのアイコンをクリックすると、Raspberry Pi の音量を調整することができます。マウスの右ボタンでこのアイコンをクリックすると、Raspberry Pi で使用する音量を調整することができます。スピーカーアイコンの横には、ネットワークアイコン (G) が表示されます。無線ネットワークに接続している場合は、無線信号の強度を示す一連のバーが表示されます。有線ネットワークに接続している場合は、2 つの矢印が表示されます。ネットワークアイコンをクリックすると、周囲の無線ネットワークが表示されます (図 3-10)。ネットワークアイコンの横にある Bluetooth アイコン (F) をクリックすると、周囲の Bluetooth デバイスに接続することができます。



◀ 図 3-10: 周辺の無線ネットワークの一覧表示

メニューバーの左側には、ランチャー (J) が表示されます。Raspberry Pi にインストールされているプログラムは、ランチャーとして表示されます。ショートカットアイコンとしてデスクトップ上に表示されるプログラムもあれば、メニュー内に表示されるプログラムもあります。左端の raspberry アイコン (K) をクリックすると、画面の左側にプログラムのメニューが表示されます (図 3-11)。

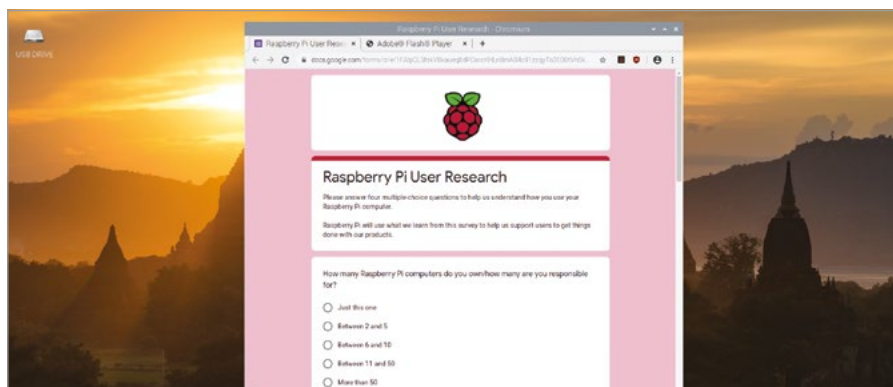


▲ 図 3-11: Raspberry Pi OS のプログラムメニュー

このメニューに表示されるプログラムは、カテゴリーに分類されて表示されます。カテゴリーの名前を見れば、そのカテゴリーに属するプログラムのタイプを判断することができます。たとえば「プログラミング」カテゴリーには、プログラムを作成するためのソフトウェアが表示され（プログラミングについては、「第 4 章: Scratch を使ってプログラミングしてみよう」で詳しく説明します）、「Games」カテゴリーには、各種のゲームソフトウェアが表示されます。このガイドでは、すべてのプログラムについて詳しくは説明しません。それぞれのプログラムを自由に試してみてください。

Chromium Web ブラウザー

Raspberry Pi の操作方法を覚えるため、最初に Chromium という Web ブラウザーをロードしてみましょう。タスクバーの左上隅にある raspberry アイコンをクリックしてメニューを表示し、マウスポインターで「Internet」カテゴリーを選択して「Chromium Web Browser」をクリックします（図 3-12）。これで、Chromium ブラウザーがロードされます。



▲ 図 3-12: Chromium Web ブラウザー

これまでに Google の Chrome ブラウザーを使ったことがあれば、Chromium にもすぐに慣れることができます。Chromium ブラウザーを使用すれば、動画の再生やゲームのプレイだけでなく、フォーラムサイトやチャットサイトで世界中のユーザーとやり取りすることができます。

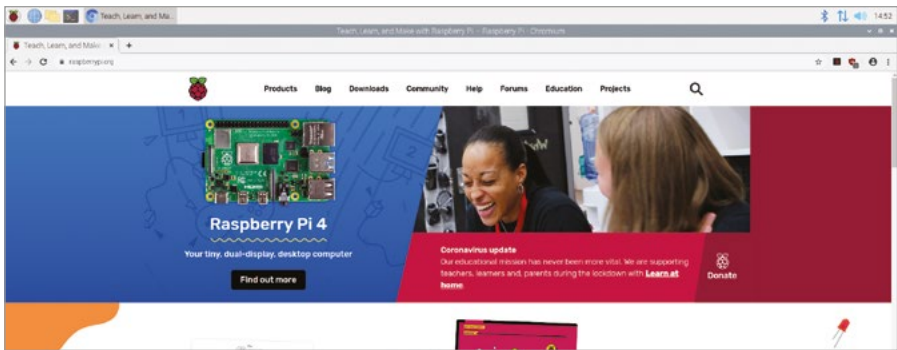
最初に、Chromium のウィンドウを最大化します。Chromium ウィンドウのタイトルバー (N) の右上には 3 つのアイコンがありますが、このうち中央の上矢印アイコン (P) をクリックします。このアイコンは、ウィンドウを最大化するためのアイコンです。このアイコンをクリックすると、ウィンドウが画面いっぱいに表示されます。最大化アイコンの左側には、最小化アイコン (O) が表示されます。このアイコンをクリックすると、ウィンドウが非表示になります (実際には、最小化された状態でタスクバーに表示されます)。タスクバーでこのウィンドウをクリックすると、元の状態に戻ります。最大化アイコンの右側には、「X」アイコン (Q) が表示されます。これは、ウィンドウを閉じるためのアイコンです。このアイコンをクリックすると、ウィンドウが終了します。



ウィンドウを閉じる前に必ず保存

作業内容を保存する前にウィンドウを閉じることはお勧めしません。多くのプログラムでは、終了ボタンをクリックしたときに作業内容を保存するように警告するメッセージが表示されますが、こうしたメッセージが表示されないプログラムもあります。

Chromium ウィンドウ上部のアドレスバー (URL を入力するための白いバー) の内部をクリックし、「www.raspberrypi.org」と入力して **ENTER** キーを押します。これにより、Raspberry Pi の Web サイトがブラウザに読み込まれます (図 3-13)。検索する用語をアドレスバーに入力することもできます。「Raspberry Pi」や「Raspberry Pi OS」などを検索してみてください。



▲ 図 3-13: Raspberry Pi の Web サイトを Chromium に読み込む

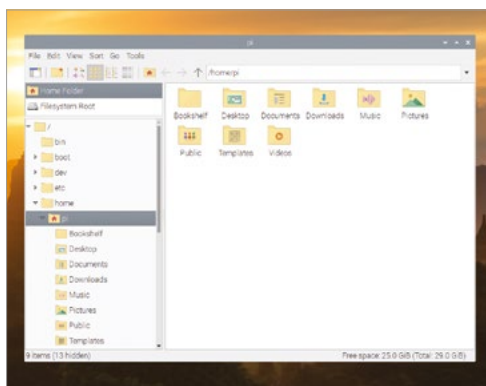
Chromium を初めて起動すると、ウィンドウの上部にいくつかのタブが表示される場合があります。別のタブに切り替える場合は、そのタブをクリックします。Chromium を終了せずにタブだけを閉じる場合は、そのタブの右端に表示されている「x」アイコンをクリックします。新しいタブを開く場合は、最後のタブの右側にある「+」アイコンをクリックするか、キーボードで **CTRL** キーと **T** キーを同時に押します。新しいタブを開くと、複数の Chromium ウィンドウを切り替える必要がなくなるため、1 つの Chromium ウィンドウで複数の Web サイトにアクセスする場合は、新しいタブを開くと便利です。

Chromium での操作が終了したら、ウィンドウ右上の「x」アイコンをクリックします。

ファイルマネージャ

プログラムファイル、ビデオファイル、画像ファイルなど、ファイルはすべて home ディレクトリに保存されます。home ディレクトリを表示するには、raspberrypi アイコンをクリックしてメニューを表示し、マウスポインターで「アクセサリ」を選択して「ファイルマネージャ」をクリックします (図 3-14)。これにより、ファイルマネージャが起動します。

▶ 図 3-14: ファイルマネージャ

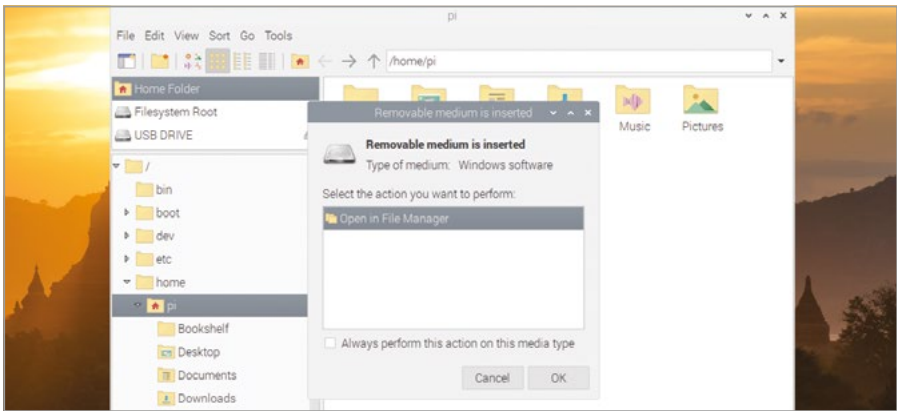


ファイルマネージャを使用すると、Raspberry Pi の microSD カードに保存されているファイルとフォルダーのみでなく (フォルダーはディレクトリとも呼ばれます)、Raspberry Pi の USB ポートに接続されているリムーバブルストレージデバイス (USB フラッシュドライブなど) に保存されているファイルとフォルダーも表示することができます。ファイルマネージャを初めて起動すると、自動的に home ディレクトリが表示されます。home ディレクトリには、フォルダーのほかにサブフォルダーも表示されます。メニューの場合と同様に、サブフォルダーもカテゴリ別に表示されます。主なサブフォルダーを以下に示します。

- **Bookshelf:**このサブフォルダーには、書籍や雑誌など、Raspberry Pi Press が発行する出版物のデジタルコピーが保存されます。このビギナーズガイド のコピーも、このサブフォルダーに保存されています。メニューの「Help」セクションの Bookshelf アプリケーションを使用して、これらのコピーを読むことができます。また、別のコピーをダウンロードすることもできます。
- **Desktop:**初めて Raspberry Pi OS を起動すると、このサブフォルダー内のファイルがデスクトップ画面に表示されます。よく使用するファイルをこのサブフォルダーに保存すると、デスクトップ画面ですぐにアクセスできるため、便利です。
- **Documents:**ユーザーが作成するほとんどのファイルは、このサブフォルダーに保存されます。
- **Downloads:**Chromium ブラウザーを使用してインターネットからファイルをダウンロードすると、そのファイルが自動的にこのサブフォルダーに保存されます。
- **Music:**このサブフォルダーには、Raspberry Pi で作成した音楽ファイルが保存されます。
- **Pictures:**このサブフォルダーには、画像ファイル が保存されます。
- **Public:**ユーザーが作成するほとんどのファイルはプライベートファイルになりますが、このサブフォルダーに保存されたファイルは、他の Raspberry Pi ユーザーも使用できるパブリックファイルになります。

- **Templates:**このサブフォルダーには、テンプレートが保存されます。テンプレートとは、基本的なレイアウトと構造だけが設定されている空白のドキュメントのことです。テンプレートはアプリケーションに付属していますが、自分で作成することもできます。
- **Videos:**このサブフォルダーには、ビデオが保存されます。ほとんどのビデオ再生プログラムは、最初にこのサブフォルダーを調べます。

ファイルマネージャウィンドウは、2つのペインから構成されています。左側のペインには、Raspberry Pi に存在するディレクトリが表示され、右側のペインには、左側のペインで選択したディレクトリ内のファイルとサブディレクトリが表示されます。リムーバブルストレージデバイスを Raspberry Pi の USB ポートに接続すると、そのデバイスをファイルマネージャで表示するかどうかを確認するポップアップウィンドウが表示されます (図 3-15)。このウィンドウで「OK」をクリックすると、リムーバブルストレージデバイスに保存されているファイルとディレクトリが表示されます。



▲ 図 3-15: リムーバブルストレージデバイスを接続する

Raspberry Pi の microSD カードとリムーバブルデバイス間で、ファイルを簡単にコピーすることができます。ファイルをコピーするには、home ディレクトリとリムーバブルデバイスをそれぞれ個別のファイルマネージャウィンドウで開き、一方のウィンドウで目的のファイルをマウスの左ボタンでクリックし、もう一方のウィンドウまでそのままマウスポインターを移動して、マウスの左ボタンから指を離します (図 3-16)。これは、ドラッグアンドドロップと呼ばれる操作です。

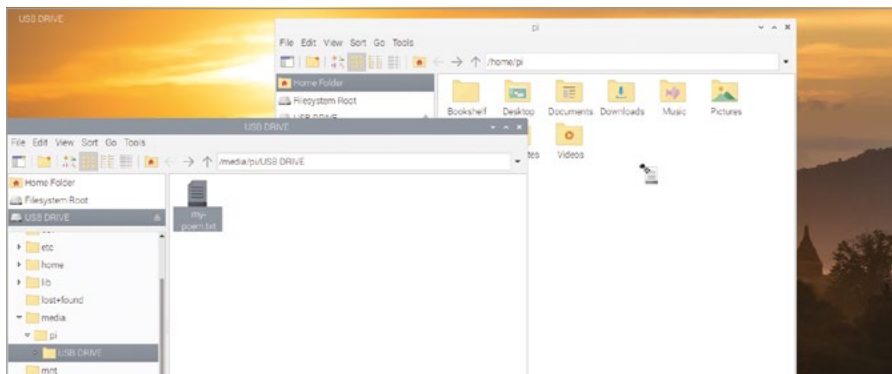
別の方法でファイルをコピーすることもできます。一方のウィンドウで目的のファイルを 1 回クリックし、「編集(E)」メニューを選択して「コピー(C)」をクリックします。次に、もう一方のウィンドウをクリックし、「編集(E)」メニューを選択して「貼り付け(P)」をクリックします。

「編集(E)」メニューには「カット(C)」オプションもあります。このオプションは「コピー(C)」オプションに似ていますが、「コピー」の場合はコピー元のファイルがそのまま残るのに対して、「カット」の場合はコピー元のファイルが削除される点が異なります。どちらのオプションも、キーボードショートカットで使用することができます。コピーの場合は **CTRL+C** キーを押し、カットの場合は **CTRL+X** キーを押します。ファイルを貼り付ける場合は、いずれも **CTRL+V** キーを押します。



キーボードショートカット

このガイドでは、CTRL+C などのキーボードショートカットが記載されていますが、これは、キーボードで CTRL キーを押しながら C キーを押すという意味です。



▲ 図 3-16: ファイルをドラッグアンドドロップする

操作が終了したら、ウィンドウの右上にある「x」アイコンをクリックしてファイルマネージャを終了します。複数のウィンドウを開いている場合は、すべてのウィンドウを閉じてください。リムーバブルストレージデバイスが Raspberry Pi に接続されている場合は、画面右上に表示されているリムーバブルデバイスのアイコンをクリックしてから、デバイスを取り外してください。



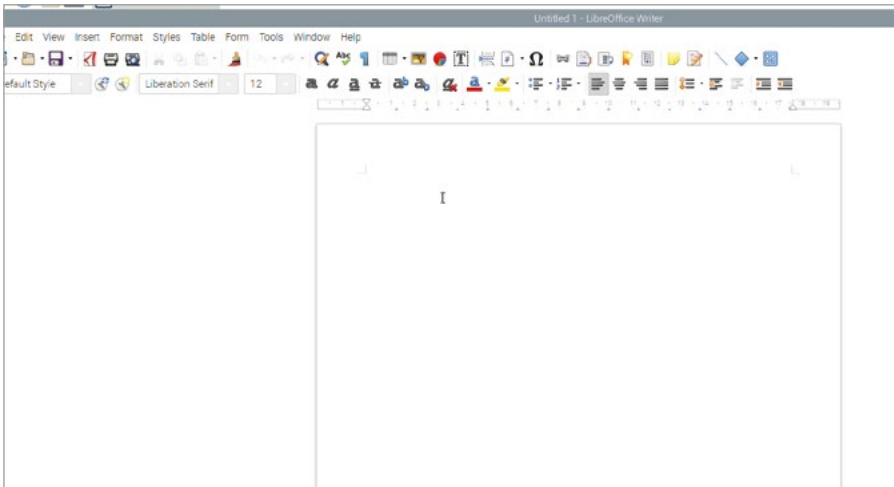
デバイスの取り外し

外部のストレージデバイスを取り外す場合は、必ず最初に取り外しボタンを押すようにしてください。ボタンを押さずにデバイスを取り外すと、デバイス内のファイルが破損して使用できなくなる可能性があります。

LibreOffice プロダクティビティスイート

Raspberry Pi には、LibreOffice スイートが付属しています。raspberrypi メニューアイコンをクリックし、マウスポインターを「オフィス」に移動して「LibreOffice Writer」をクリックしてください。これにより、LibreOffice の文書作成機能が起動します (図 3-17)。LibreOffice は、広く使用されているプロダクティビティスイートです (Microsoft Office や Google ドキュメントなどがプロダクティビティスイートです)。

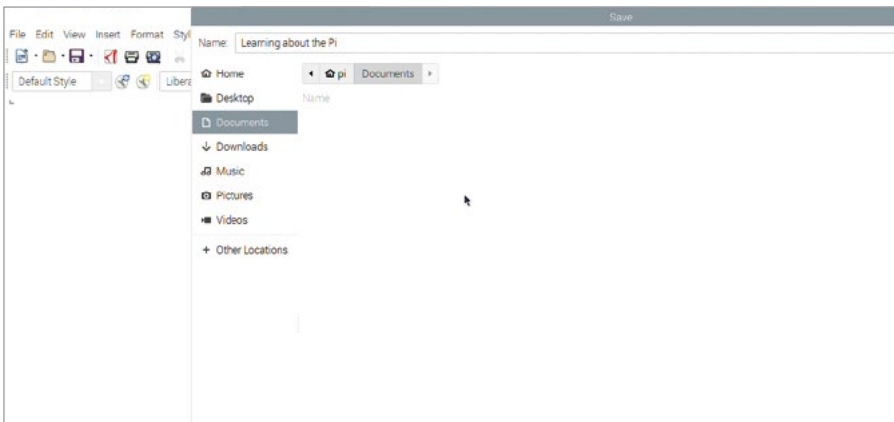
注: すべての Raspberry Pi OS イメージにデフォルトで LibreOffice がインストールされているわけではありません。インストールされていない場合は、49 ページの「Recommended Software ツール」を参照して、適切なツールをインストールしてください。



▲ 図 3-17: LibreOffice Writer プログラム

文書作成機能を使用すると、文書を作成できるだけでなく、文書の書式を簡単に設定することができます。たとえば、フォントのスタイル、色、サイズ、効果などを変更できるだけでなく、画像、グラフ、表などのコンテンツを挿入することもできます。また、文章の入力時にスペルミスや文法的な間違いがハイライト表示されるため（スペルミスは赤、文法的な間違いは緑で表示されます）、正しい文章を入力することができます。

最初に、Raspberry Pi とそのソフトウェアについてここまで学習した内容を入力してみましょう。ウィンドウの上部にあるさまざまなアイコンを使用して、文字を大きくしてみたり、文字の色を変えてみたりして、各アイコンの機能を確認してください。マウスポインターをアイコンの上に置くと、そのアイコンの機能を示す「ツールチップ」が表示されます。入力作業が終了したら、「File」メニューの「Save」オプションをクリックします（図 3-18）。作業ファイルに名前を付けて「Save」ボタンをクリックします。



▲ 図 3-18: ドキュメントを保存する



作業内容を保存する

作業の途中でも、定期的に作業内容を保存するようにしてください。こまめに保存すれば、作業中に停電になっても、少し作業をするだけで、すぐに元の状態に戻すことができます。

LibreOffice Writer は、LibreOffice プロダクティビティスイートのほんの一部分にすぎません。メニューの「オフィス」カテゴリには、LibreOffice Writer のほかにも以下のプログラムが表示されます。

- **LibreOffice Base:** これは、情報の検索、分析、保存を行うためのデータベースツールです。
- **LibreOffice Calc:** これは、数値の処理やグラフの作成を行うための表計算ツールです。
- **LibreOffice Draw:** これは、画像や図を作成するためのイラストレーションツールです。
- **LibreOffice Impress:** これは、スライドの作成とスライドショーの再生を行うためのプレゼンテーションツールです。
- **LibreOffice Math:** これは、正しい形式で数式を作成するための式エディターツールです。このツールで作成した数式は、別のドキュメント内で使用することもできます。

LibreOffice は、Raspberry Pi 以外のコンピューターやオペレーティングシステムでも使用することができます。LibreOffice を libreoffice.org から無料でダウンロードして、Microsoft Windows、Apple macOS、Linux コンピューターにインストールすることができます。

LibreOffice の詳しい使用方法を確認する場合は、「Help」メニューをクリックしてください。作業が終了したら、ウィンドウ右上の「x」アイコンをクリックして LibreOffice Writer を終了します。



ヘルプ情報

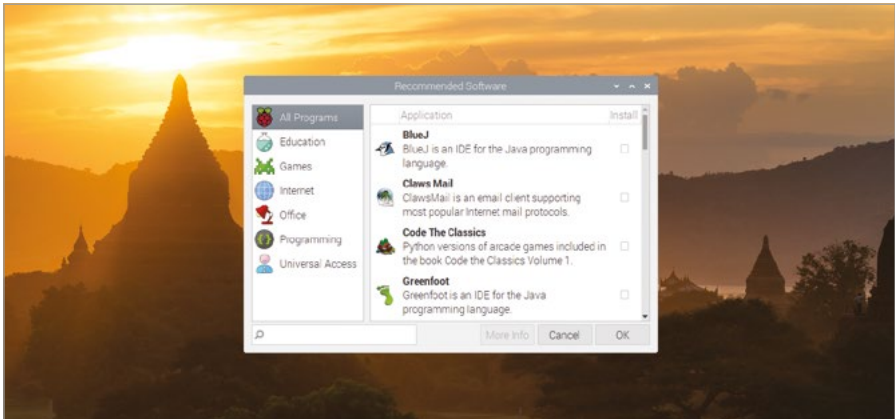
ほとんどのプログラムには、ヘルプメニューが用意されています。このメニューで、そのプログラムに関するさまざまなヘルプ情報を確認することができます。プログラムの操作方法がわからない場合は、ヘルプメニューを使ってみましょう。

Recommended Software ツール

Raspberry Pi OS にはさまざまなソフトウェアが付属していますが、これら以外のソフトウェアを使用することもできます。Recommended Software ツールで、推奨ソフトウェアを確認することができます。

Recommended Software ツールを使用する場合は、インターネットに接続する必要があります。Raspberry Pi をインターネットに接続して raspberry メニューアイコンをクリックし、マウスポインターを「設定」に移動して「Recommended Software」をクリックします。これにより、Recommended Software ツールが起動し、使用可能なソフトウェアに関する情報のダウンロードが開始されます。

数秒間待つと、互換性のあるソフトウェアパッケージが一覧表示されます (図 3-19)。raspberry メニューに表示されるソフトウェアと同様に、これらのソフトウェアパッケージもさまざまなカテゴリに分類されて表示されます。左側のペインでいずれかのカテゴリをクリックすると、そのカテゴリに属するソフトウェアが表示され、「All Programs」をクリックすると、すべてのソフトウェアが表示されます。

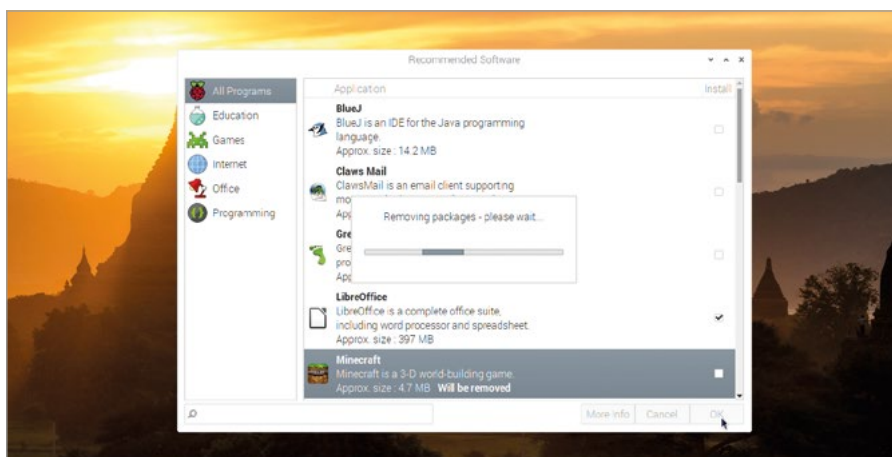


▲ 図 3-19: Recommended Software ツール

右端のチェックボックスにチェックマークが表示されているソフトウェアは、Raspberry Pi にすでにインストールされているソフトウェアです。チェックマークが付いていないソフトウェアについては、チェックボックスを選択してチェックマークを付けると、インストールすることができます。複数のソフトウェアにチェックマークを付けて一度にインストールすることができますが、推奨サイズより小さな microSD カードを使用している場合は、容量が不足する可能性があります。

同じ方法で、ソフトウェアをアンインストールすることができます。チェックボックスにチェックマークが付いているソフトウェアを探してチェックマークを外すと、そのソフトウェアがアンインストールされます。ソフトウェアを誤ってアンインストールした場合や、後でそのソフトウェアが必要になった場合は、もう一度チェックマークを付けると、そのソフトウェアをインストールすることができます。

インストールするソフトウェアにチェックマークを付け、アンインストールするソフトウェアのチェックマークを外したら、「OK」ボタンをクリックします (図 3-20)。これにより、インストールプロセスとアンインストールプロセスが開始されます。プロセスが完了すると、ダイアログボックスが表示されます。このダイアログボックスで「OK」をクリックすると、Recommended Software ツールが終了します。

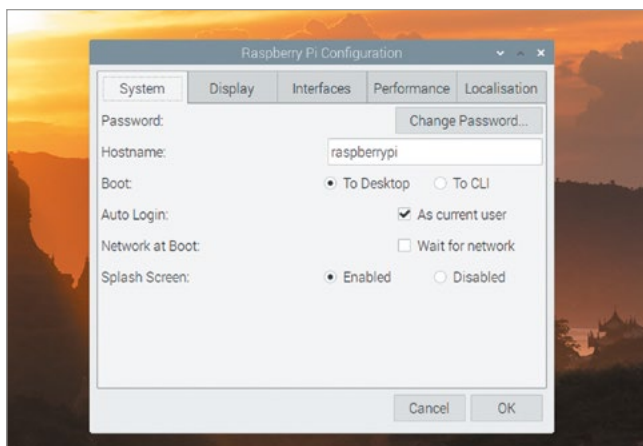


◀ 図 3-20: ソフトウェアをアンインストールする

ソフトウェアのインストールとアンインストールを行うためのもう 1 つのツールとして、「Add/Remove Software」ツールがあります。このツールは、Raspberry Pi OS メニューの「設定」カテゴリーに表示されます。このツールを使用すると、さまざまなソフトウェアを選択できますが、いずれも Raspberry Pi Foundation がテストを行っていないソフトウェアです。

Raspberry Pi Configuration ツール

この章で最後に紹介するツールは、Raspberry Pi Configuration というツールです。これは、この章の最初に紹介したウェルカムウィザードに似ていますが、ウェルカムウィザードよりも多くの OS 設定を変更することができます。raspberrypi アイコンをクリックしてマウスポインターを「設定」カテゴリーに移動し、「Raspberry Pi Configuration」をクリックします (図 3-21)。これにより、Raspberry Pi Configuration ツールが起動します。



◀ 図 3-21: Raspberry Pi Configuration ツール

このツールは、5 つのタブから構成されています。最初のタブは「システム」タブです。このタブを使用して、アカウントパスワードの変更や、ホスト名の設定（ローカルの無線ネットワークまたは有線ネットワークで Raspberry Pi が使用するホストの名前）など、各種の設定を変更することができます。ただし、ほとんどの設定については、変更する必要はありません。次のタブは「ディスプレイ」タブです。このタブでは、接続先の TV やモニターに合わせて、画面の表示設定を変更することができます。



さらに詳しい情報

この章に記載されている情報は、Raspberry Pi Configuration ツールに慣れるための概要的な情報です。各設定に関する詳しい情報については、「付録 E: Raspberry Pi Configuration ツール」を参照してください。

「インターフェイス」タブには、さまざまな設定が表示されます。これらの設定はすべて、初期値として無効になっています。Raspberry Pi カメラモジュールなど、新しいハードウェアを追加した場合に、このタブで設定を変更する必要があります（変更する必要があるかどうかについては、追加するハードウェアの製造元の指示に従ってください）。それ以外の場合は、このタブで設定を変更する必要はありません。ただし、次の場合は例外です。SSH -「セキュアシェル」が有効になります。SSH クライアントを使用して、ネットワーク上の別のコンピューターから Raspberry Pi にログインすることができます。VNC -「バーチャルネットワークコンピューティング」が有効になります。VNC クライアントを使用して、ネットワーク上の別のコンピューターから Raspberry Pi OS のデスクトップ画面の表示と制御を行うことができます。リモート GPIO - ネットワーク上の別のコンピューターから、Raspberry Pi の GPIO ピンを使用することができます（GPIO ピンについては、「第 6 章: Scratch と Python を使って物理的コンピューティングに挑戦しよう」で詳しく説明します）。

次のタブは「パフォーマンス」タブです。このタブでは、Raspberry Pi のグラフィックスプロセッシングユニット（GPU）で使用するメモリーの量を設定することができます。また、一部のモデルでは、オーバークロックというプロセスにより、Raspberry Pi のパフォーマンスを上げることができます。ただし、どうしても変更する必要がある場合を除き、このタブに表示される設定はそのままにしておくことをお勧めします。

最後のタブは「ローカライゼーション」タブです。このタブでは、ロケールを変更することができます。ロケールとは、Raspberry Pi OS で使用する言語、数値の表示形式、タイムゾーン、キーボードの配列、WiFi を使用する国などを制御するための設定のことです。ここでは、何も変更せずに「キャンセル」をクリックします。



注意

国により、WiFi 無線で使用できる周波数の規則が異なります。Raspberry Pi Configuration ツールで、無線 LAN の国を別の国（自分が住んでいない国）に変更すると、ネットワークに接続できなくなる場合があります。それだけでなく、無線の使用に関する法律に違反する可能性もあるので、注意してください。

シャットダウン

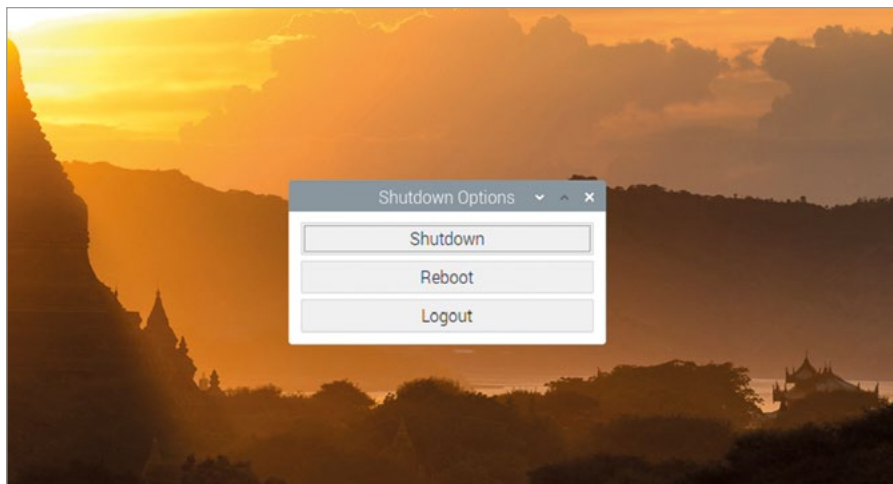
Raspberry Pi OS のデスクトップ画面について学習したところで、Raspberry Pi を安全にシャットダウンする方法を見ていきましょう。これは非常に重要な操作です。その他のコンピューターと同様に、Raspberry Pi の場合も、作業中のファイルが揮発性メモリー に保存されます。システムの電源を切ると、揮発性メモリーの内容が消去されます。ドキュメントを作成する場合は、定期的にドキュメントを保存するだけで問題ありません。保存したドキュメントファイルは、揮発性メモリーから不揮発性メモリー である microSD カードに保存されるため、データの欠落が発生することはありません。

しかし、オープンされているファイルは、作業中のファイルだけではなく、Raspberry Pi OS の稼働中は、多くのファイルがオープンされたままの状態になります。この状態で Raspberry Pi の電源を切ると、オペレーティングシステムが破損し、再インストールが必要になる可能性があります。

これを防ぐには、ファイルをすべて保存してから Raspberry Pi OS の電源を切る必要があります。この手順を、オペレーティングシステムのシャットダウン といいます。

OS をシャットダウンするには、デスクトップ画面左上の raspberry アイコンをクリックして「ログアウト」を選択します。3 つのオプションが表示されたウィンドウが開きます (図 3-22)。「Shutdown」、「Reboot」、「Logout」という 3 つのオプションが表示されます。これらのうち最もよく使用するのは、「Shutdown」オプションです。このオプションをクリックすると、稼働しているソフトウェアとオープンされているファイルがすべて終了してから、Raspberry Pi がシャットダウンされます。ディスプレイが真っ暗になってから数秒間待つと、緑色で点滅している Raspberry Pi のライトが消えます。この状態になったら、電源を切っても問題ありません。

Raspberry Pi の電源をもう一度入れる場合は、電源ケーブルを入れ直すか、コンセントの電源を切り替えます。



▲ 図 3-22: Raspberry Pi をシャットダウンする

「Reboot」オプションを選択すると、シャットダウンの場合と同様にすべてのプログラムとファイルが終了しますが、Raspberry Pi が再起動するという点が異なります。リブートの動作は、Raspberry Pi をシャットダウンして電源を切ってから、電源を入れ直すという動作と同じです。オペレーティングシステムの再起動が必要な変更を行った場合に（重要なソフトウェアをアップデートした場合など）、「Reboot」オプションを使用します。また、ソフトウェアで問題が発生すると（これをクラッシュといいます）Raspberry Pi OS の動作が不安定になるため、このような場合もリブートを実行する必要があります。

Raspberry Pi に複数のユーザーアカウントが存在する場合は、「Logout」を使用すると非常に便利です。ログアウトを実行すると、稼働中のプログラムがすべて終了し、ユーザー名とパスワードを入力するログイン画面が表示されます。間違っても「Logout」オプションをクリックした場合は、ユーザー名として「pi」と入力し、ウェルカムウィザードで作成したパスワードを入力すれば、元のアカウントに戻ることができます。



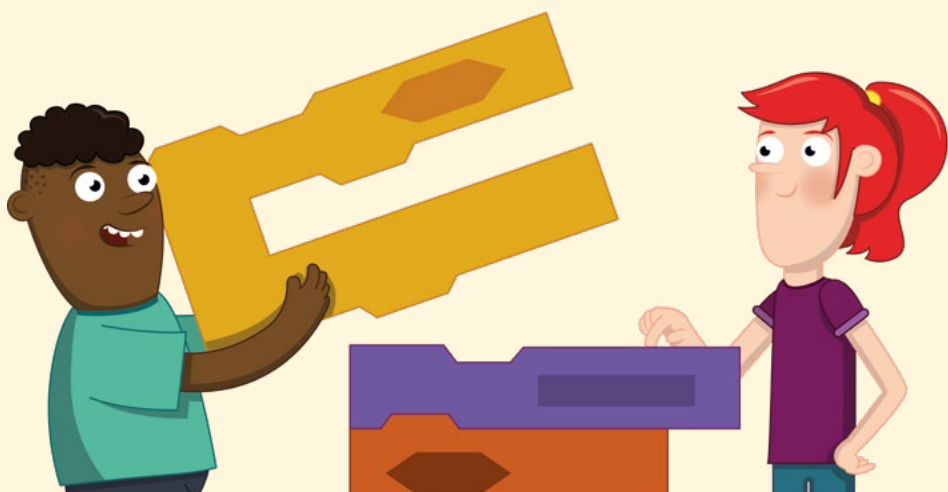
注意

シャットダウンする前に Raspberry Pi の電源ケーブルを抜かないようにしてください。シャットダウンの前に電源ケーブルを抜くと、オペレーティングシステムが破損したり、作成したファイルがなくなったりする可能性があります。

第4章

Scratch 3 を使って プログラミングしてみよう

この章では、ブロックベースのプログラミング言語である
Scratch を使用してコーディングを行う方法について学習します

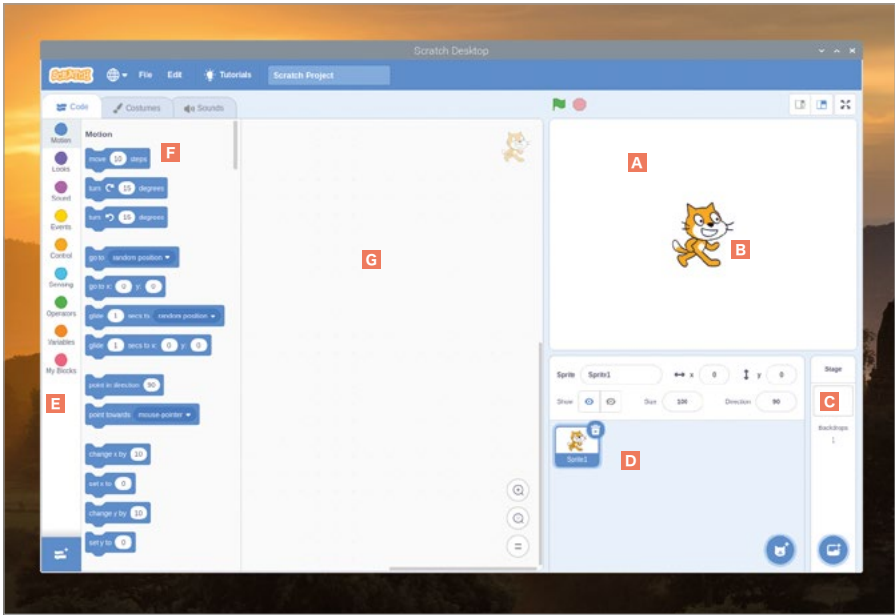


Raspberry Pi を使用して、頭に浮かんだアイデアを自分だけのプログラムとして形にしてみましょう。Raspberry Pi なら、これまでにプログラムを作ったことがあるかどうかに関係なく（プログラムを作成することを「プログラミング」や「コーディング」といいます）楽しみながらプログラミングを学習することができます。

ここでは、Scratch というビジュアルプログラミング言語を使用してコーディングを行います。マサチューセッツ工科大学 (MIT) が作ったプログラミング言語です。従来のプログラミング言語は、コンピューターが実行する処理をテキストで記述します。これは「ケーキを作るときに文章でレシピを書く」といった方法によく似ています。Scratch の場合は、ブロックを並べてプログラムを作成していきます。各ブロックにはあらかじめコードが記述されていて、ジグソーパズルのピースのように色分けされています。

初めてコーディングを行う場合は、年齢に関係なく、Scratch を使用することをお勧めします。外観はカラフルで親しみやすいものになっていますが、簡単なゲームやアニメーションから複雑なロボットプロジェクトまで、さまざまなものを開発できる強力な機能が用意されています。

Scratch 3 のインターフェイスの紹介



A ステージ領域 - スプライトは、舞台上で演じる役者のように、プログラムの指示に従ってステージ領域内を移動します。

B スプライト - Scratch で制御するキャラクターのことをスプライトといいます。

C ステージ用コントロール - ステージ用の各種コントロールを使用して、ステージを変更することができます (画像をステージの背景として追加するなど)。

D スプライトリスト - Scratch で作成したスプライトと Scratch に読み込んだスプライトは、すべてここに表示されます。

E ブロックパレット - プログラムで使用できるすべてのブロックが、ブロックパレットに表示されます。ブロックパレットは、カテゴリー別に分けられて表示されます。

Scratch のバージョン

このガイドの作成時点では、Raspberry Pi OS には Scratch 1、2、3 が付属しています。これらはすべて、Raspberry Pi OS メニューの「プログラミング」セクションに表示されます。このガイドでは、Scratch 3 について説明します。Scratch 3 を使用できるのは Raspberry Pi 4 だけであることに注意してください。また、Raspberry Pi Zero、Model A、A+、B、B+ では、Scratch 2 を使用することはできません。

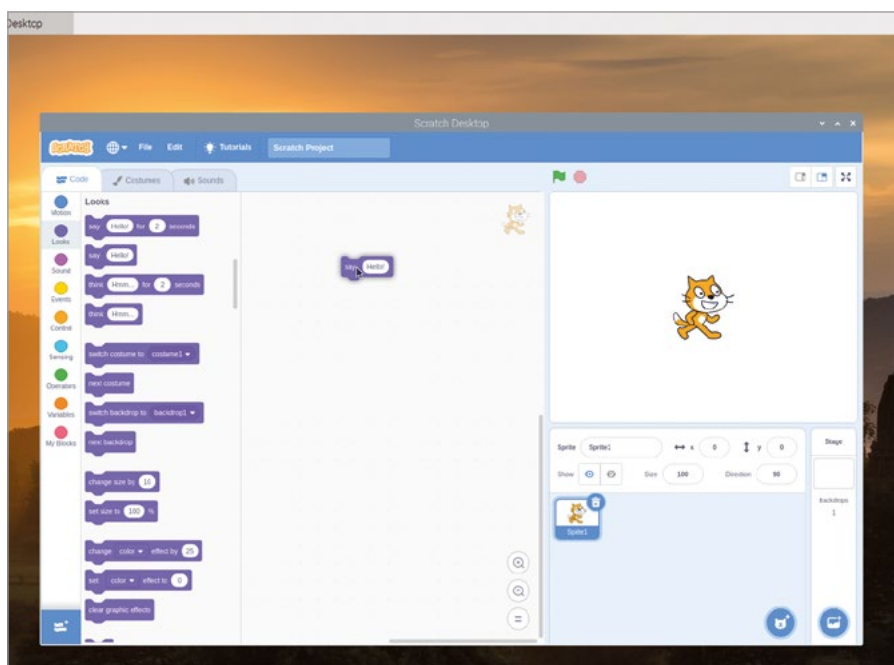
F ブロック - ブロックには、プログラムコードが事前に記述されています。ブロックを使用すると、プログラムを段階的に作成することができます。

G コード領域 - ブロックパレットのブロックをコード領域にドラッグアンドドロップして、プログラムを作成します。

Scratch で「こんにちは!」というプログラムを作成する

Scratch 3 の起動方法は、Raspberry Pi にインストールされている他のプログラムの場合と同じです。raspberrypi アイコンをクリックして Raspberry Pi OS メニューを表示し、カーソルを「プログラミング」セクションに移動して「Scratch 3」をクリックします。クリックしてから数秒後に、Scratch 3 のユーザーインターフェイスが表示されます。

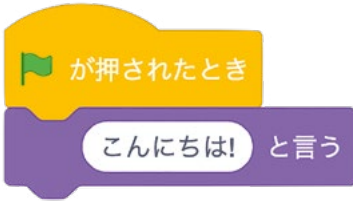
多くのプログラミング言語では、コードを記述してコンピューターに指示を出す必要がありますが、Scratch の場合は違います。最初に、Scratch ウィンドウ左側のブロックパレットで「見た目」カテゴリーをクリックします。これにより、「見た目」カテゴリーに属するブロックが紫で表示されます。「**こんにちは! と言う**」ブロックを探し、マウスの左ボタンをクリックしたまま、Scratch ウィンドウ中央のコード領域にこのブロックをドラッグしてマウスボタンから指を離します (図 4-1)。



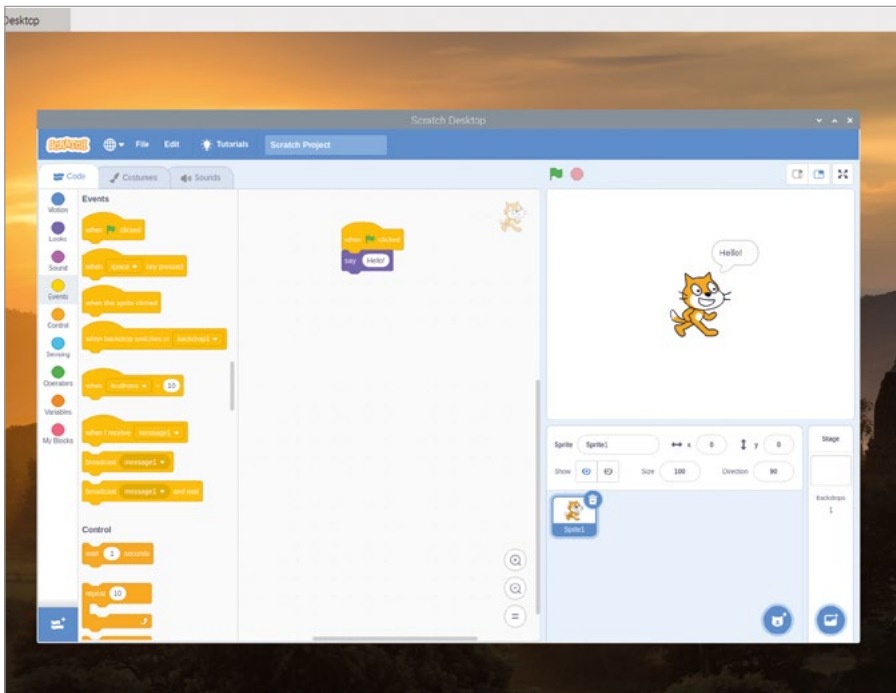
▲ 図 4-1: ブロックをコード領域にドラッグアンドドロップする

ここで、ブロックの形に注目してください。ブロックの上側がくぼんでいて、その分だけブロックの下側が突き出していることがわかります。ここからは、ジグソーパズルのピースのように、ブロックの上下に別のブロックをはめ込んでいきます。このプログラムでは、トリガー ブロックを配置します。

ブロックパレットの「イベント」カテゴリー (金色のカテゴリー) をクリックし、「**緑の旗が押されたとき**」ブロックをコード領域にドラッグします。このブロックはハット ブロックといいます。これが、トリガーブロックとして機能します。このブロックの突き出している部分を「**こんにちは! と言う**」ブロックのくぼんでいる部分にはめ込んで、マウスボタンから指を離します。正確にはめ込む必要はありません。ブロック同士が近くにあれば、自動的にはめ込まれます。自動的にはめ込まれない場合は、ブロックをクリックして位置を調整してください。

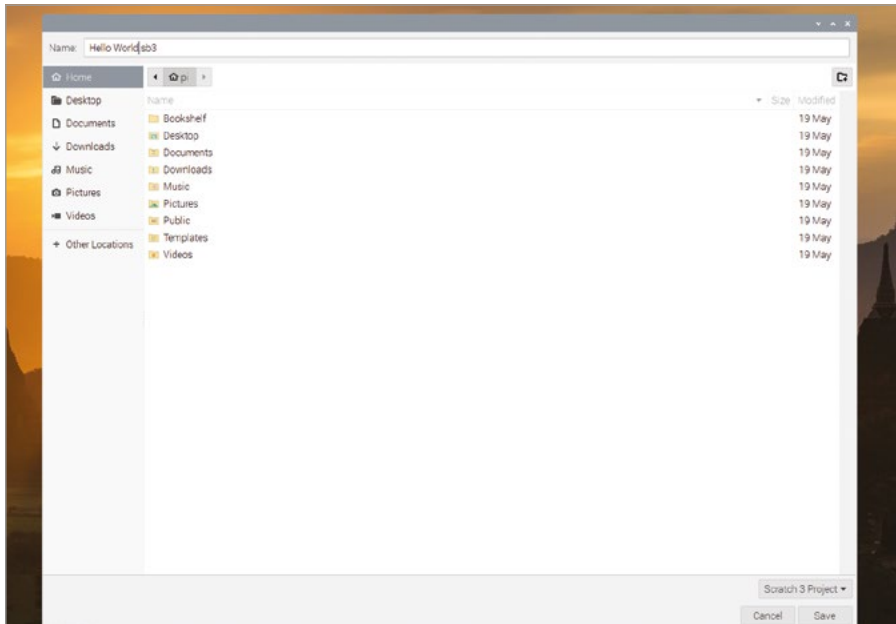


これでプログラムが完成しました。実際にプログラムを動かしてみましょう。これを、プログラムの実行 といいます。ステージ領域の左上にある緑色のフラグアイコンをクリックします。猫がステージ上で「こんにちは!」と言えば、最初のプログラムは成功です (図 4-2)。

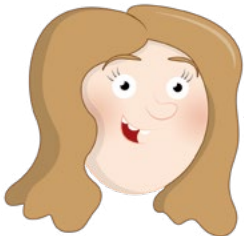


▲ 図 4-2: ステージ領域左上の緑のフラグアイコンをクリックする

ここで、プログラムに名前を付けて保存しましょう。「ファイル」メニューをクリックして「コンピューターに保存する」を選択します。次に、プログラム名を入力して保存ボタンをクリックします (図 4-3)。



▲ 図 4-3: プログラムにわかりやすい名前を付けて保存する



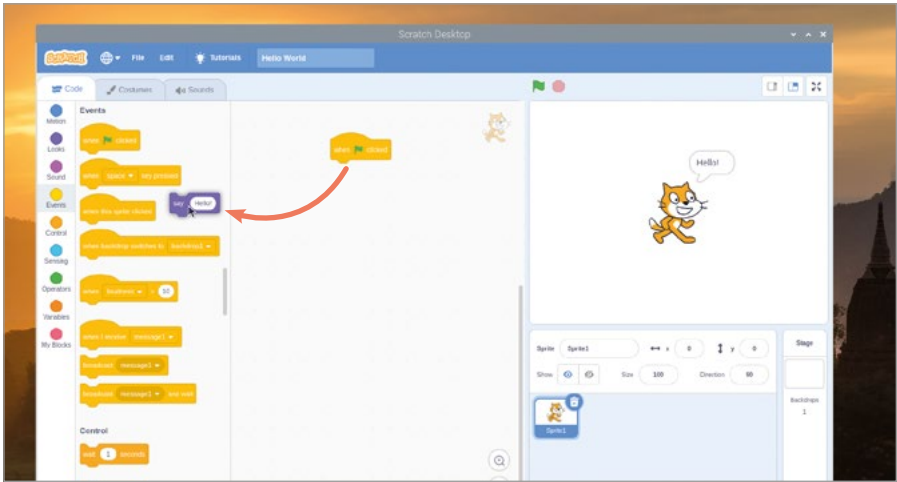
言葉を変えてみよう

Scratch には、値を編集できるブロックがあります。「こんにちは!」の部分をクリックして別の言葉を入力し、緑のフラグアイコンをクリックしてみましょう。どうなりましたか?

次のステップ: シーケンス処理

このプログラムにはブロックが 2 つしかないため、何度フラグをクリックしても、猫は「こんにちは!」としか言いません。もっと複雑な動作を実行するには、シーケンス処理 について理解する必要があります。コンピューターで実行するプログラムは、どんなに簡単なものでも、いくつかの手順を記述する必要があります。たとえるならば、料理のレシピのようなものです。それぞれの手順は、リアシーケンスと呼ばれる論理的な順序で実行されます。

最初に、「**こんにちは! と言う**」ブロックをクリックしてブロックパレットにドラッグします (図 4-4)。この操作により、ブロックがプログラムから削除され、トリガーブロックの「**緑の旗が押されたとき**」だけが残ります。



▲ 図 4-4: ブロックパレットにドラッグしてブロックを削除する

ブロックパレットの「動き」カテゴリーをクリックし、「10 歩動かす」ブロックをコード領域のトリガーブロックの下にドラッグします。「動き」という名前のとおり、このカテゴリーには、スプライト（この場合は猫）が向いている方向に何歩動かかを指定するためのブロックが用意されています。



さらに動作を追加して、シーケンスを作成してみましょう。ピンクの「音」パレットをクリックし、「終わるまで ニャー の音を鳴らす」ブロックを「10 歩動かす」ブロックの下にドラッグします。次に「動き」カテゴリーをクリックし、「10 歩動かす」ブロックを「...音を鳴らす」ブロックの下にドラッグします。「10 歩動かす」ブロックの「10」の部分をクリックし、「-10」に変更します（変更後のブロックは、「-10 歩動かす」になります）。



ステージ領域左上の緑のフラグアイコンをクリックして、プログラムを実行してみましょう。猫が鳴きながら右側に移動してから（スピーカーをオンにするかヘッドフォンを接続して、鳴き声を確認してください）、元の位置に戻るはずですが、フラグアイコンをもう一度クリックすると、猫は同じ動作を繰り返します。

これで、一連のシーケンスが完成しました。Scratch では、上から順に動作が 1 つずつ実行されます。Scratch では、シーケンス内の命令が一度に 1 つしか実行されないため、動作が非常に速くなります。ここで、「-10 歩動かす」ブロックを下にドラッグして「終わるまで ニャー の音を鳴らす」ブロックから離し、「終わるまで ニャー の音を鳴らす」ブロックをブロックパレットにドラッグします。次に、ブロックパレットの「ニャー の音を鳴らす」ブロックを「play sound Meow until done」ブロックがあった位置にドラッグし、「-10 歩動かす」ブロックを元の位置に戻します（以下の図のようになります）。



この状態で緑のフラグアイコンをクリックすると、猫が動かなくなります。実際には動いているのですが、動作が非常に速いため、動いていないように見えます。これは、音が鳴り終わる前に「ニャー の音を鳴らす」ブロックが次の動作に進んでしまうためです。Raspberry Pi の処理速度は非常に速いため、猫の動作を確認する前に次の動作が実行されるということです。「終わるまで ニャー の音を鳴らす」ブロックを使用すればこの動作を修正できますが、それ以外にも方法があります。ブロックパレットのオレンジ色の「制御」カテゴリーをクリックし、「1 秒待つ」ブロックを「ニャー の音を鳴らす」ブロックと「-10 歩動かす」ブロックの間にドラッグします。



この状態で緑のフラグアイコンをクリックすると、猫が右側に移動し、1 秒間止まってから、元の位置に戻ってきます。この動作は、遅延 と呼ばれます。シーケンスを構成する一連の命令の実行時間を制御する場合は、この遅延という動作が重要になります。



チャレンジ: 歩数を増やす



「... 歩動かす」ブロックをコード領域に追加したり、すでにコード領域にドラッグされている「... 歩動かす」ブロックの歩数を増やしたりしてみましょう。各ブロックで設定されている歩数が違う場合、猫の動きはどうなりますか? また、「ニャー」という音が鳴っているときに別の音を鳴らそうとするとどうなりますか?

ループ処理

これまでに作成したシーケンスは、1 回だけ実行されます。緑のフラグアイコンをクリックすると、猫が鳴きながら移動し、そこでプログラムが停止します。もう一度フラグアイコンをクリックするまで、プログラムは停止したままになります。しかし、Scratch にはループ というコントロールブロックが用意されているため、プログラムを繰り返して実行することができます。

ブロックパレットの「制御」カテゴリーをクリックし、「**ずっと**」ブロックを探します。このブロックを、「**緑の旗が押されたとき**」ブロックと「**10 歩動かす**」ブロックの間にドラッグします。



ここで、アルファベットの「C」のような形をした「ずっと」ブロックが、シーケンス内のすべてのブロックを囲むようにして自動的に伸びることに注意してください。この状態で緑のフラグアイコンをクリックすると、「**ずっと**」ブロックの機能がすぐにわかります。プログラムが 1 回の実行で終了するのではなく、何度も繰り返し実行されます。「ずっと」という言葉のとおり、ずっと実行されます。この動作のことを、プログラミング用語で無限ループ といいます (永久ループという場合もあります)。名前のとおり、終わりのないループ処理のことです。

猫の鳴き声はずっと聞こえて気になる場合は、緑のフラグアイコンの横にある赤い八角形アイコンをクリックすると、プログラムが停止します。ここで、違うループに変えてみましょう。上にある方の「10 歩動かす」ブロックを「が押されたとき」ブロックの下方向にドラッグすると、その下にあるすべてのブロックが「ずっと」ブロックから外れます。「ずっと」ブロックをブロックパレットにドラッグしてコード領域から削除し、代わりに「10 回繰り返す」ブロックを「が押されたとき」ブロックの下にドラッグします (以下の図のようになります)。



緑のフラグアイコンをクリックして、新しいプログラムを実行してみましょう。最初は、元のプログラムと同じように、シーケンス内の動作を何度も繰り返しているだけのように見えます。しかし、このプログラムの場合、一連の動作を 10 回繰り返すと、ループ処理が停止します。この動作のことを有限ループといいます。有限ループの場合、ループの終了条件を自分で定義することになります。プログラミングにおいて、ループ処理は非常に便利な処理です。特にゲームプログラムなどでは、無限ループ処理と有限ループ処理を多用します。



ループの回数を変えてみよう

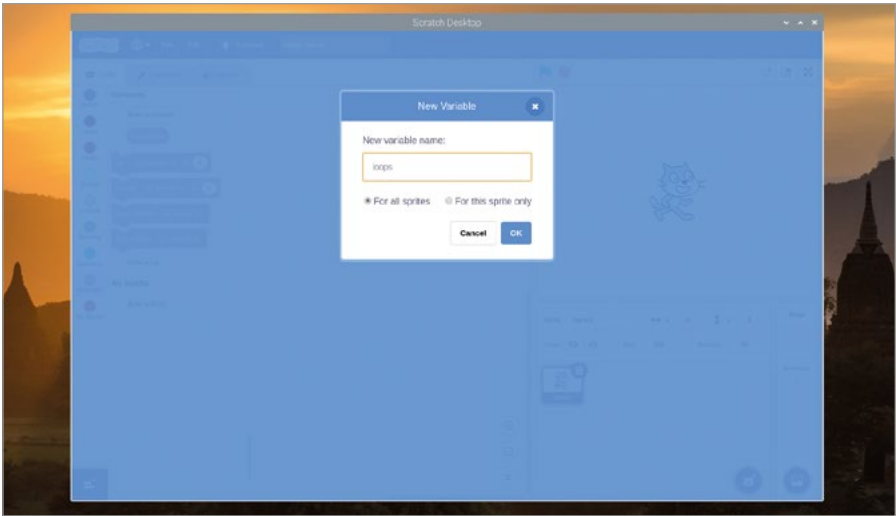
「... 回繰り返す」ブロックのループ回数を多くするとどうなりますか？ ループ回数を少なくするとどうなりますか？ ループの回数を 0 にするとどうなりますか？

変数と条件

Scratch プログラムのコーディングを本格的に開始する前に、2 つの重要な概念を理解しておく必要があります。それは、変数 と条件 です。変数と条件は、互いに密接に関係しています。「変数」はその名前のとおり、時間の経過やプログラムの制御によって変化する可能性がある値のことです。変数には、名前とそれを保管する値という 2 つの主要な特性があります。この値は、数値である必要はありません。数値やテキスト、true や false、完全な空 (null 値 といいます) などのいずれでもかまいません。

プログラミングで変数を使用すると、非常に便利です。たとえば、ゲームを考えてみましょう。ゲームでは、キャラクターの状態、物体の移動速度、現在プレイしているレベル、スコアなどの情報を追跡する必要があります。これらの情報は、プログラム内ではすべて変数として追跡されます。

最初に「ファイル」メニューをクリックし、「コンピューターに保存する」を選択してプログラムを保存します。すでにプログラムが保存されている場合は、プログラムを上書きするかどうかを確認するメッセージが表示されます。上書きを選択すると、前に保存したプログラムが新しいプログラムに置き換わります。次に、「ファイル」をクリックして「新規」を選択し、新しい空白のプロジェクトを開きます（「現在のプロジェクトの内容を置き換えますか」という内容のメッセージが表示された場合は「OK」をクリックします）。ブロックパレットで濃いオレンジ色の「変数」カテゴリーをクリックし、「変数を作る」ボタンをクリックします。変数名として「loops」を入力して「OK」をクリックします（図 4-5）。この操作により、ブロックパレットに一連の「ループ...」ブロックが表示されます。



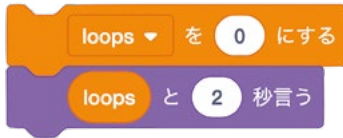
▲ 図 4-5: 新しい変数の名前を入力する

「loops を 0 にする」ブロックをコード領域にドラッグします。これにより、ループ変数の値が 0 に設定されます。これを初期化といいます。次に、ブロックパレットの「見た目」カテゴリーをクリックし、「こんにちは! と 2 秒言う」ブロックを「loops を 0 にする」ブロックの下にドラッグします。

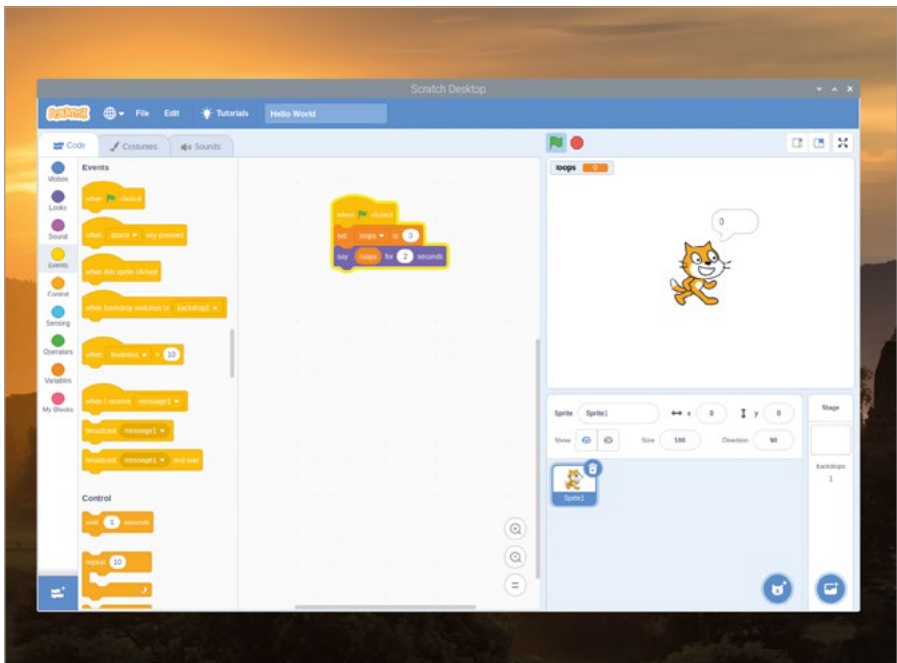


これまでに見てきたように、「こんにちは! と言う」ブロックを配置すると、このブロックに書かれている言葉を猫が話します。このブロックに自分で言葉を書き込む代わりに、変数を使用して言葉を入力することができます。ブロックパレットの「変数」カテゴリーをクリックし、リストの先頭に表示されて

いる、左側にチェックマークが付いている丸い「loops」ブロック (レポーターブロック といいます) を「こんにちは! と 2 秒言う」ブロックの「こんにちは!」の部分に重なるようにドラッグします。これにより、「loops と 2 秒言う」という新しいブロックができます。



ブロックパレットの「イベント」カテゴリーをクリックし、「**が押されたとき**」ブロックを一連のブロックの上にドラッグします。この状態で、ステージ領域左上の緑のフラグアイコンをクリックすると、猫の吹き出しに「0」が表示されます (図 4-6)。これは、loops 変数の値が「0」になっているためです。

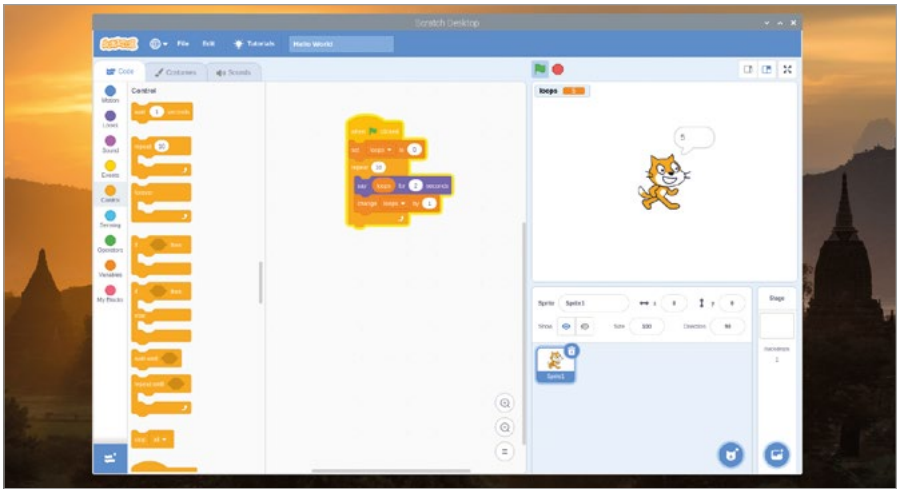


▲ 図 4-6: 猫の吹き出しに loops 変数の値が表示される

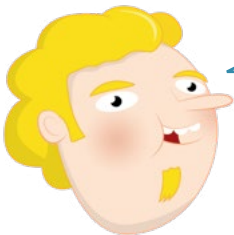
変数の値は変えることができます。ブロックパレットの「変数」カテゴリーをクリックし、「**loops を 1 ずつ変える**」ブロックをシーケンスの最下部にドラッグします。次に「制御」カテゴリーをクリックし、「**10 回繰り返す**」ブロックを「**loops を 0 にする**」ブロックのすぐ下にドラッグします。これにより、各ブロックが以下の図のように配置されます。



この状態で緑のフラグアイコンをクリックすると、カウントが 0 から 9 まで順に増えていきます。こうした動作になるのは、プログラムの変数を変更したためです。ループ処理が実行されるたびに、loops 変数の値が 1 ずつ増えていきます (図 4-7)。



▲ 図 4-7: ループ処理によってカウントが増えていく



ゼロからカウントする

ここで作成したループ処理は 10 回繰り返して実行されますが、猫が数えるカウントは「9」で止まります。これは、変数の初期値として「0」を設定しているためです。0 と 9 の間には、0 と 9 を含めて 10 個の数字があるため、カウントが「10」に達する前にプログラムが停止します。変数の初期値を 0 から 1 に変更すると、10 までカウントされるようになります。

変数を使用して、さらに複雑な動作を実行することができます。「loops と 2 秒言う」ブロックをドラッグして「10 回繰り返す」ブロックから離し、「10 回繰り返す」ブロックの下にドロップします。「10 回繰り返す」ブロックをブロックパレットにドラッグしてコード領域から削除します。次に、「... まで繰り返す」ブロックをコード領域にドラッグし、「loops を 0 にする」ブロックともう 1 つのブロックを囲むように配置します (以下の図のようになります)。次に、ブロックパレットの緑色の「演算」カテゴリでダイヤモンドの形をした「=」ブロックをクリックし、「... まで繰り返す」ブロック上にドラッグします。

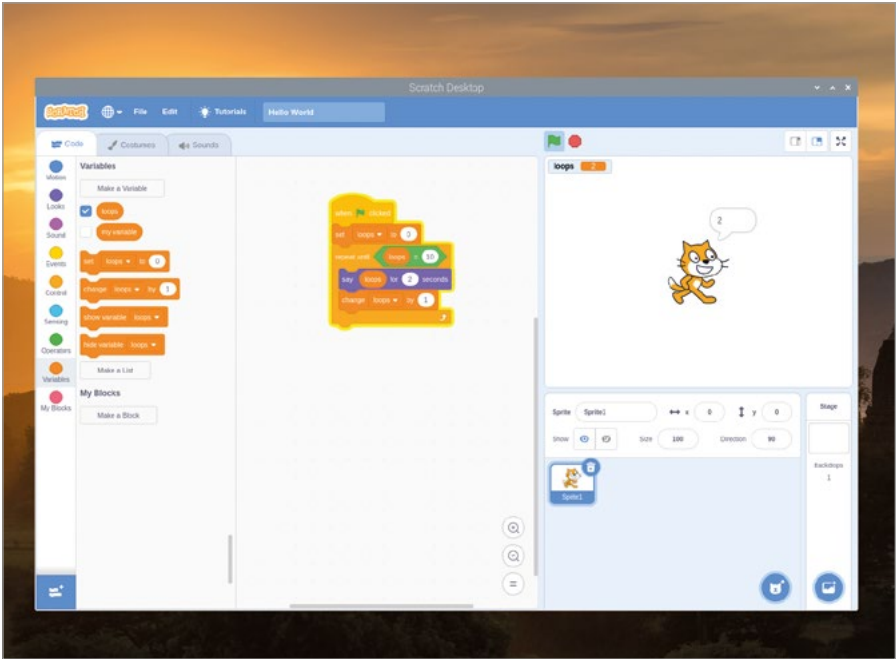


この演算ブロックにより、2 つの値を比較することができます (変数など)。「変数」カテゴリをクリックし、「loops」レポーターブロックを「=」ブロック内の空白のスペースに重なるようにドラッグします。次に、右側の「50」の部分をクリックして「10」を入力します。



この状態で、ステージ領域上部の緑のフラグアイコンをクリックすると、以前と同じ動作が実行されます (カウントが 0 から 9 まで増えて、プログラムが停止します - 図 4-8)。これは、「... まで繰り返す」ブロックと「10 回繰り返す」ブロックの動作がまったく同じであるためです。ただ

し、ループ回数自体をカウントするという方法と、loops 変数に対して設定された数値を比較するという方法には明確な違いがあります。このプログラムの場合、loops 変数の値が 10 に達するとプログラムが停止します。





▲ 図 4-8: 「...回繰り返す」ブロックで比較演算子を使用する

こうした動作を実行する演算子のことを比較演算子といいます。これは、2つの値を比較するための演算子です。ブロックパレットの「演算」カテゴリーをクリックすると、中央に「<」「>」「=」という記号が書かれたダイヤモンド型のブロックがあることがわかります。「=」だけでなく、「<」と「>」も比較演算子です。「<」の場合、左側の値が右側の値よりも小さい場合に動作が実行され、「>」の場合は、左側の値が右側の値よりも大きい場合に動作が実行されます。

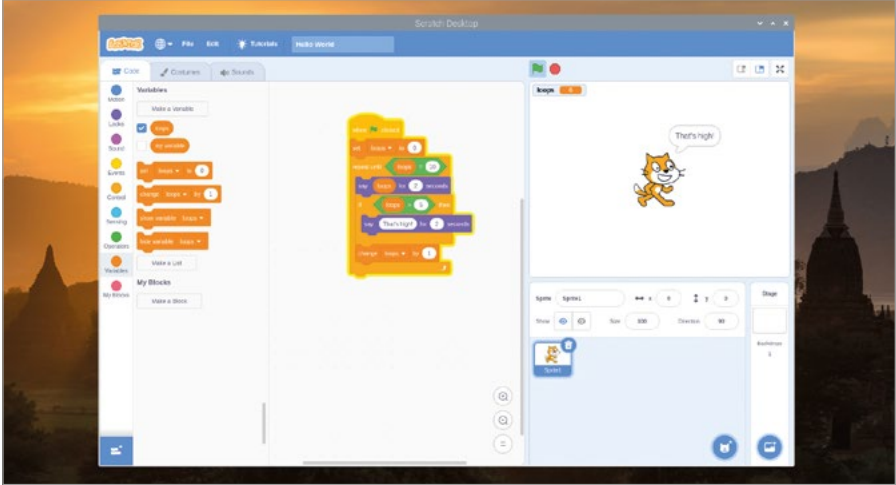
ブロックパレットの「制御」カテゴリーで「もし...なら」ブロックをクリックし、コード領域の「loops と 2秒言う」ブロックのすぐ下にドラッグします。その際、「loops を1ずつ変える」ブロックも自動的に「もし...なら」ブロックの一部として囲まれてしまうため、ブロックをドラッグして「もし...なら」ブロックの下に移動します。次に、ブロックパレットの「見た目」カテゴリーをクリックし、「こんにちは! と 2秒言う」ブロックを「もし...なら」ブロック内にドラッグします。次に、ブロックパレットの「演算」カテゴリーをクリックし、「=」ブロックを「もし...なら」ブロック内のダイヤモンド型のスペースに重ねるようにドラッグします。



「もし ... なら」ブロックは、条件付きブロックです。条件付きブロックの場合、特定の条件が満たされた場合にのみ、条件ブロック内のブロックで指定された動作が実行されます。ブロックパレットの「変数」カテゴリの「loops」レポーターブロックをクリックし、「」ブロックの左側の空いているスペースに重ねるようにドラッグして、右側の「50」という値を「5」に変更します。次に、「」ブロックの「こんにちは!」の部分をクリックして「これは大きな数字です!」に変更します。



この状態で、緑のフラグアイコンをクリックします。最初は、猫がゼロからカウントを読み上げていきます。これは、以前の動作と同じです。カウントが「6」を超えると「もし...なら」ブロックがトリガーされ、猫の吹き出しに「これは大きな数字です!」というテキストが表示されます(図 4-9)。おめでとうございます! これで変数と条件の使い方について理解しました!



▲ 図 4-9: カウントが 6 を超えると「これは大きな数字です!」が表示される



チャレンジ: 大きな数字と小さな数字



5 未満の数字について猫にコメントさせるには、どのようにプログラムを変更すればよいでしょうか? 大きな数字と小さな数字の両方で猫にコメントさせることはできるでしょうか?

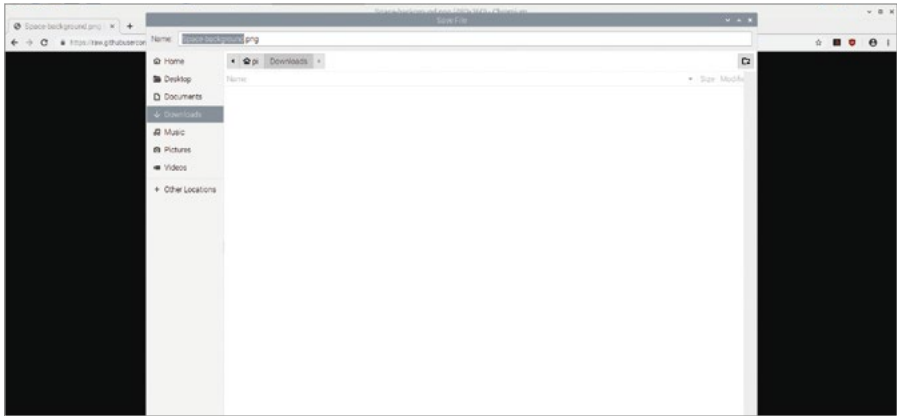
「もし...なら」ブロックを使うと、これらの処理を簡単に作成することができます。

プロジェクト 1: 宇宙飛行士の反応時間タイマー

Scratch の仕組みを理解したところで、もう少しインタラクティブなプログラムを作成してみましょう。ここでは、ESA の「ティム・ピーク」というイギリス人飛行士をモデルとして、国際宇宙ステーション用のタイマーを作成します。

これまで作成したプログラムを保存し、「ファイル」メニューの「新規」をクリックして新しいプロジェクト画面を開きます。最初に、「ファイル」メニューの「コンピューターに保存する」をクリックして、「Astronaut Reaction Timer」という名前前でプロジェクトを保存します。

このプロジェクトでは、2つの画像を使用します。1つはステージの背景画像で、もう1つはスプライトの画像です。これらの画像は、Scratchの組み込みリソースには含まれていません。これらの画像をダウンロードするには、raspberrypi アイコンをクリックして Raspberry Pi OS メニューを表示し、マウスポインターを「Internet」に移動して「Chromium Web Browser」をクリックします。ブラウザが起動したら、ブラウザのアドレスバーに「rpf.io/astronaut-backdrop」と入力して **ENTER** キーを押します。宇宙の画像を右クリックして「名前を付けて画像を保存」を選択し、「ダウンロード」フォルダーを選択して「保存」ボタンをクリックします (図 4-10)。次に、アドレスバーに戻って「rpf.io/astronaut-sprite」と入力して **ENTER** キーを押します。



▲ 図 4-10: 背景画像を保存する

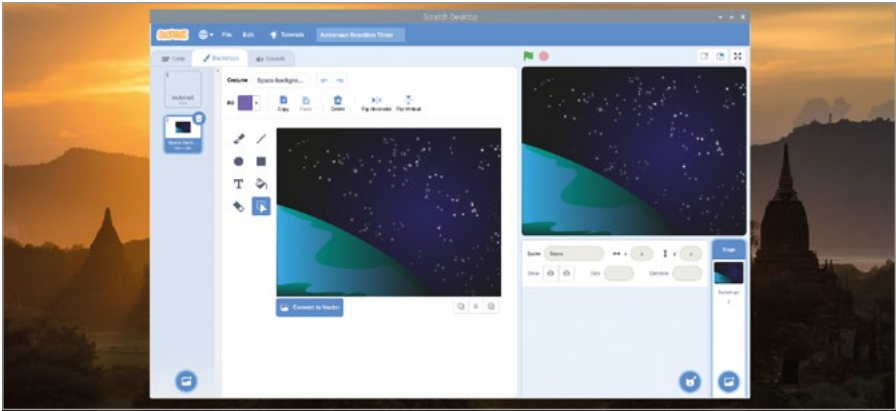
ティム・ピークの画像を右クリックして「名前を付けて画像を保存」を選択し、「ダウンロード」フォルダーを選択して「保存」ボタンをクリックします。これらの画像を保存したら、タスクバーを使用して Scratch 3 に戻ります。Chromium ブラウザーは、開いたままでも終了してもかまいません。



ユーザーインターフェイス

ここまでの学習で、Scratch 3 のユーザーインターフェイスにはだいぶ慣れてきたはずですが、ここからは、これまでに学習してきた知識を応用して、いくつかのプログラムを作っていきます。画面のどこに何があるのかを忘れた場合は、この章の最初にあるユーザーインターフェイスの画像を確認してください。

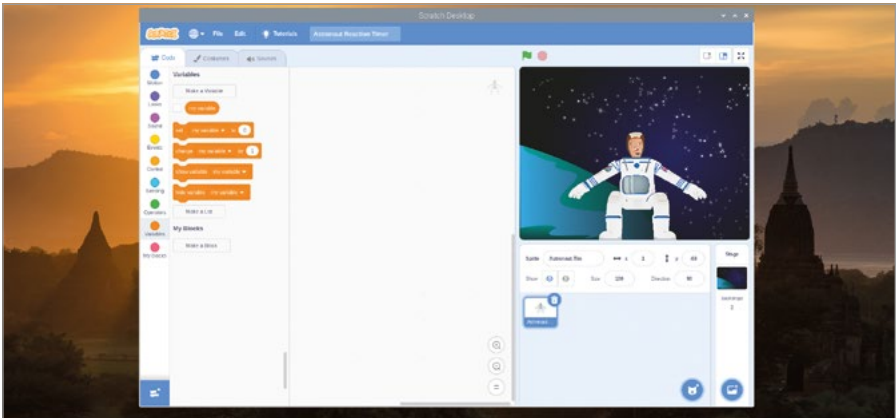
スプライトリストにある猫のスプライトを右クリックして「削除」を選択します。「背景を選ぶ」アイコン (🎨) にマウスポインターを置いて「背景をアップロード」アイコン (📁) をクリックします。次に、「ダウンロード」フォルダー内の **Space-background.png** ファイルを選択して「OK」をクリックします。空白の背景が宇宙の画像に代わり、コード領域にも同じ画像が表示されます (図 4-11)。この背景画像を編集することもできますが、ここでは、Scratch 3 ウィンドウの左上に表示されている「コード」タブをクリックします。



▲ 図 4-11: ステージ領域に宇宙の背景画像が表示される

「スプライトを選ぶ」アイコン (🐱) にマウスポインターを置き、リストの先頭に表示される「スプライトをアップロード」アイコン (📄) をクリックします。

「ダウンロード」フォルダーで **Astronaut-Tim.png** ファイルを選択して「OK」をクリックします。スプライトは、ステージ領域の中央に自動的に配置されますが、中央に配置されない場合もあります。その場合は、マウスを使ってスプライトの位置を中央下部に調整してください (図 4-12)。

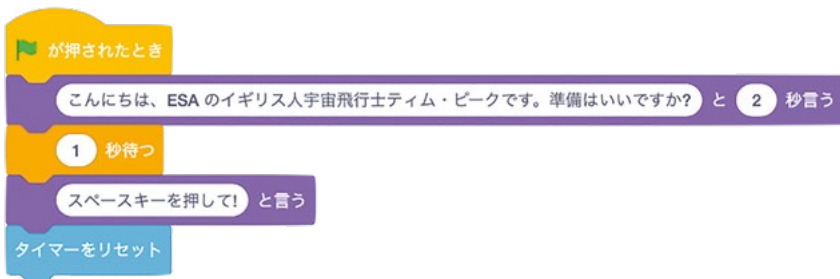


▲ 図 4-12: スプライトをステージ領域の中央下部にドラッグする

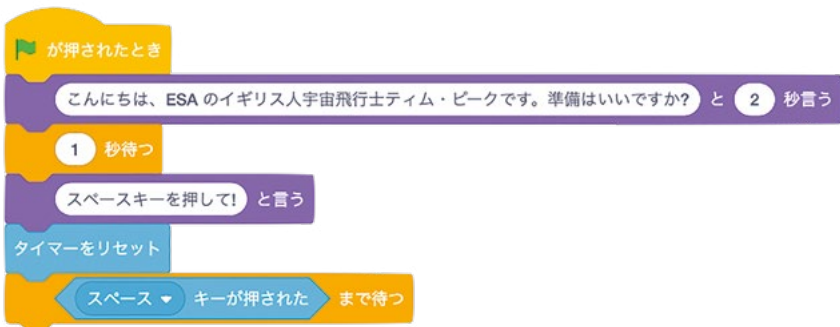
新しい背景とスプライトを配置したら、プログラムを作成していきましょう。最初に、「time」という変数を作成します。「変数を作る」ボタンをクリックして「time」と入力し、「すべてのスライプ用」オプションが選択されていることを確認して「OK」をクリックします。ステージ領域でスプライトをクリックし (スプライトペインのスプライトをクリックしてもかまいません)、「イベント」カテゴリの「**🚩 が押されたとき**」ブロックをコード領域にドラッグします。次に、「見た目」カテゴリの「**こんにちは! と 2 秒言う**」ブロックをコード領域にドラッグし、「こんにちは!」の部分で「こんにちは、ESA のイギリス人宇宙飛行士ティム・ピークです。準備はいいですか?」に変更します。



「制御」カテゴリの「1秒待つ」ブロックをコード領域にドラッグし、「見た目」カテゴリの「こんにちは! と言う」ブロックをコード領域にドラッグします。次に、「こんにちは!」を「スペースキーを押して!」に変更し、「調べる」カテゴリの「タイマーをリセット」ブロックをコード領域にドラッグします。このブロックは、タイミングを処理する特殊な変数を制御するためのブロックです。たとえば、ゲーム内での反応速度を計測する場合などに、このブロックを使用します。

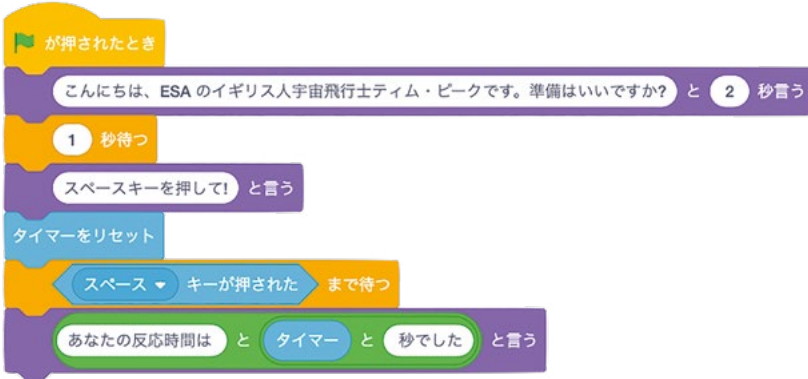


「制御」カテゴリの「...まで待つ」ブロックをコード領域に追加し、このブロックのダイヤモンド型のスペースに「調べる」カテゴリの「スペースキーが押された」ブロックをドラッグします。これにより、キーボードでスペースキーを押すまでプログラムが一時停止しますが、一時停止中もタイマーは稼働しているため、「スペースキーを押して!」というメッセージが表示されから実際にスペースキーが押されるまでの時間が、タイマーによって正確に測定されます。

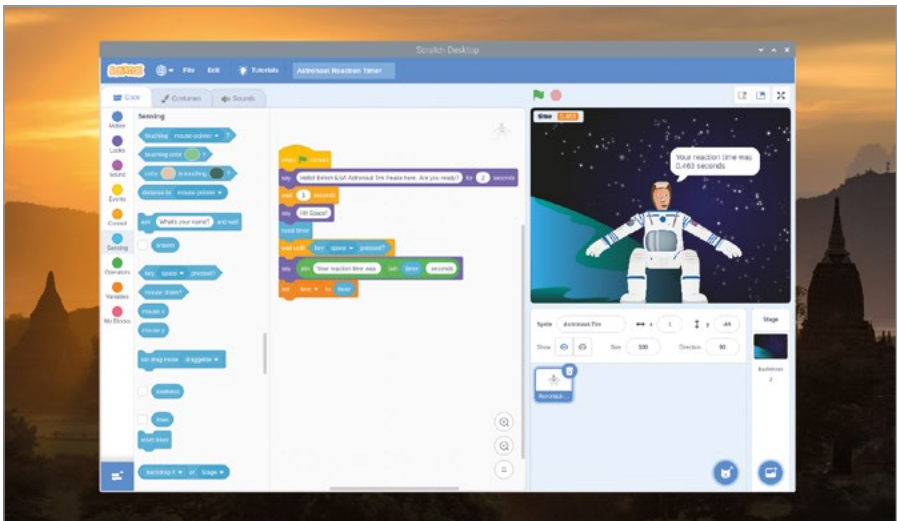


ここで、スペースキーを押すまでにどれくらいの時間がかかったのかを、わかりやすい方法で測定してみましょう。これを行うには、「演算」カテゴリの「りんごとバナナ」ブロックを使用する必要があります。このブロックを使用すると、2つの値（変数を含む）を組み合わせたことができます。これを結合といいます。

最初に、「こんにちは! と言う」ブロックを置き、「こんにちは!」の部分に「りんごとバナナ」演算ブロックをドラッグします。「りんご」の部分をクリックして「あなたの反応時間は」と入力し(「時間は」の後に空白を追加してください)、「りんごとバナナ」ブロックをコード領域上の「バナナ」の部分にドラッグします。次に、「調べる」カテゴリの「タイマー」レポートブロックをコード領域上の「りんご」の部分にドラッグし、「バナナ」の部分に「秒でした」と入力します(「秒でした」の前に空白を追加してください)。



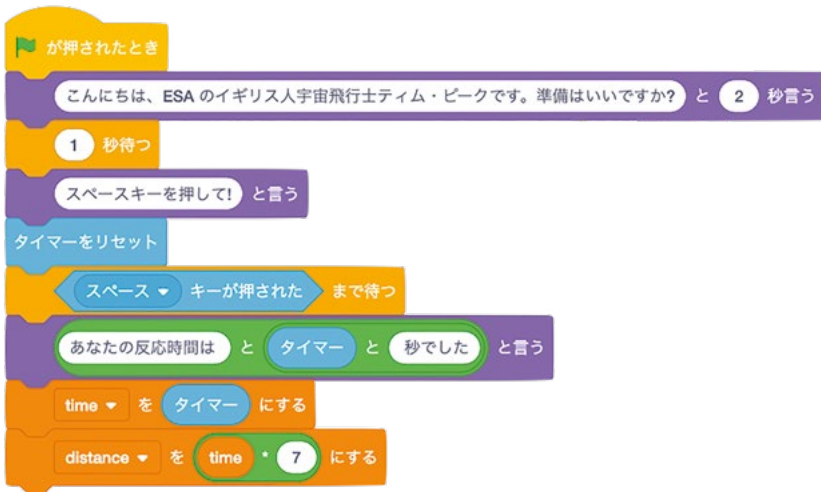
次に、「変数」カテゴリの「変数を 0 にする」ブロックをシーケンスの一番下にドラッグします。このブロックの変数名が「time」になっていることを確認し(なっていない場合は、下矢印をクリックして「time」を選択します)、右側の「0」の部分に「調べる」カテゴリの「タイマー」レポートブロックをドラッグします。これで、ゲームをテストする準備が整いました。ステージ領域左上の緑のフラグアイコンをクリックしてみましょう。フラグアイコンをクリックして「スペースキーを押して!」というメッセージが表示されたら、すぐにスペースキーを押します(図 4-13)。最高記録を目指して頑張ってください。



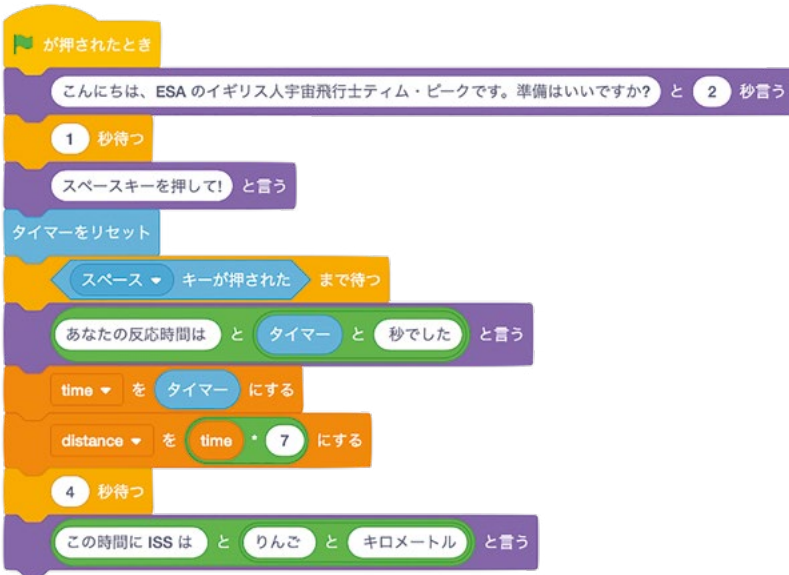
▲ 図 4-13: ゲームをプレイする

次に、このプログラムを拡張してみましょう。国際宇宙ステーションの速度は秒速約 7 キロメートルですが、あなたがスペースキーを押すまでの時間に国際宇宙ステーションがどれくらいの距離を進んだのかを計算してみましょう。最初に、「distance」という新しい変数を作成します。新しい変数を作成すると、「変数」カテゴリ内のブロックが自動的にその新しい変数名に変わりますが、すでにコード領域に追加されている「time」ブロックは変わらないことに注意してください。

「distance を 0 にする」ブロックをコード領域にドラッグし、このブロックの「0」の部分に「*」演算ブロックをドラッグします（これは掛け算用のブロックです）。この演算ブロックの左側の空白スペースに「time」レポートブロックをドラッグし、右側の空白スペースに「7」を入力します。これで、結合後のブロックが「distance を time * 7 にする」のようになります。このブロックにより、スペースキーを押すまでの時間が 7 倍されます。この値が、スペースキーを押すまでの間に国際宇宙ステーション (ISS) が進んだ距離になります。



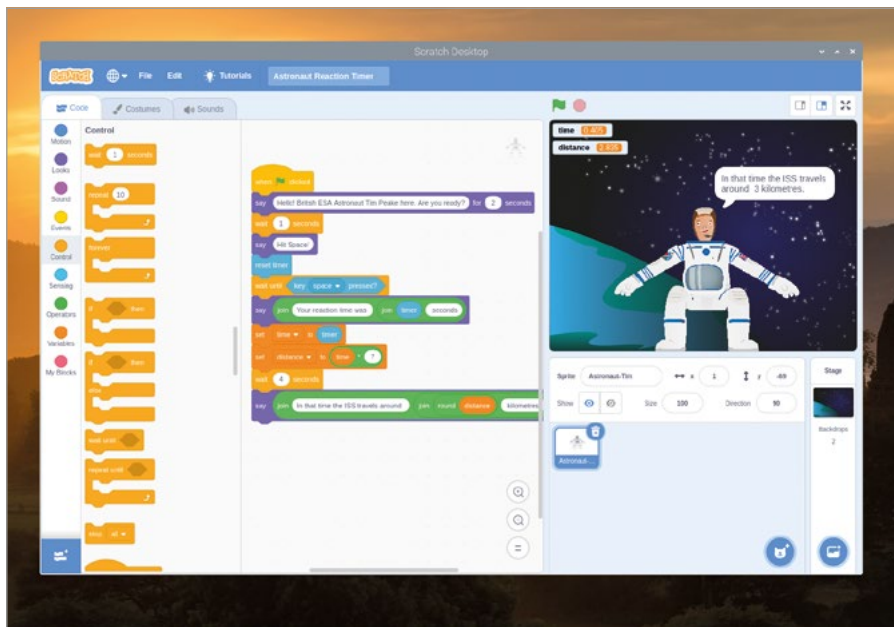
「1 秒待つ」ブロックをコード領域に追加し、「1」の部分を変更して「4」に変更します。次に、「こんにちは! と言う」ブロックをシーケンスの一番下にドラッグし、このブロックに 2 つの「りんごとバナナ」ブロックをドラッグします（この手順については、これまで説明しています）。次に、左側の「りんご」の部分に「この時間に ISS は」と入力し（「ISS は」の後に空白を追加してください）、「バナナ」の部分に「キロメートル進みました」と入力します（「キロメートル」の前に空白を追加してください）。



次に、中央の「りんご」の部分に「...を四捨五入」演算ブロックをドラッグし、この演算ブロックの空白部分に「distance」レポートブロックをドラッグします。この「...を四捨五入」ブロックにより、小数点以下の値が整数に四捨五入され、読みやすい値になります。



緑のフラグアイコンをクリックして、プログラムを実行してみましょう。スペースキーを押すまでの間に、ISS はどれくらいの距離を進むのでしょうか。作業が終わったら、必ずプログラムを保存するようにしてください。プログラムを保存しておく、次回はその続きから作業できるため、最初からプログラムを作り直す必要はありません。



▲ 図 4-14: ISS が進んだ距離がタイムの吹き出しに表示される



チャレンジ: 新しいスプライトを作ってみよう

宇宙飛行士以外にすばやい反射神経が必要になる職業は何でしょうか。その職業を表示するためのスプライトと背景を自分で描いてみましょう。

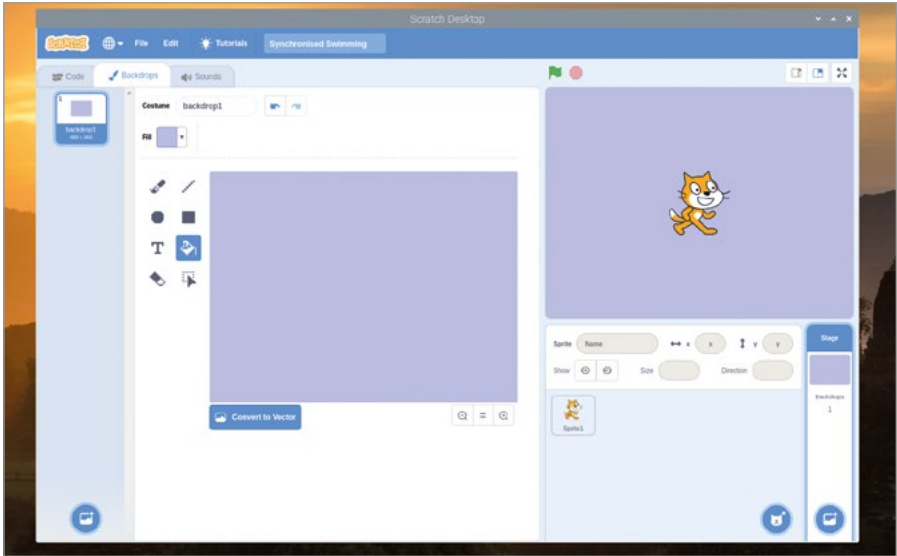
プロジェクト 2: シンクロナイズドスイミング

ほとんどのゲームで複数のボタンを使用しますが、これから作成するプロジェクトでも、キーボードの ← キーと → キーを使用します。

オンラインプロジェクト

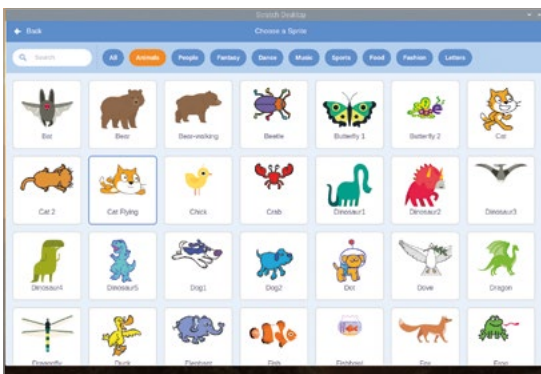
このプロジェクトは、オンライン (rpf.io/synchro-swimming) でも公開されています。

最初に、「Synchronised Swimming」という名前で新しいプロジェクトを作成します。ステージ管理セクション右側の「ステージ」をクリックし、ウィンドウ左上の「背景」タブをクリックします。次に、背景の下に表示されている「ビットマップに変換」ボタンをクリックします。「塗りつぶし」パレットで水色を選択し、塗りつぶしアイコン (🖌️) をクリックして背景内をクリックすると、水色に塗りつぶされた背景に変わります (図 4-15)。



▲ 図 4-15: 背景を水色で塗りつぶす

スプライトリストにある猫のスプライトを右クリックして「削除」を選択します。次に、「スプライトを選ぶ」アイコン (🐱) をクリックして、スプライトの一覧を表示します。「動物」カテゴリーをクリックして「Cat Flying」を選択します (図 4-16)。このスプライトを使用すると、面白いシンクロナイズドスイミングになりそうです。



▲ 図 4-16: ライブラリーでスプライトを選ぶ

スプライトをクリックし、2 つの「**スペース キーが押されたとき**」イベントブロックをコード領域にドラッグします。

最初のブロックの「スペース」の横にある下矢印をクリックし、オプションリストで「左向き矢印」を選択します。「動き」カテゴリーの「**15 度回す**」ブロックを「**スペース キーが押されたとき**」ブロックの下にドラッグします。次に、もう 1 つのイベントブロックで

「右向き矢印」を選択し、このブロックの下に「**C 15 度回す**」ブロックをドラッグします。



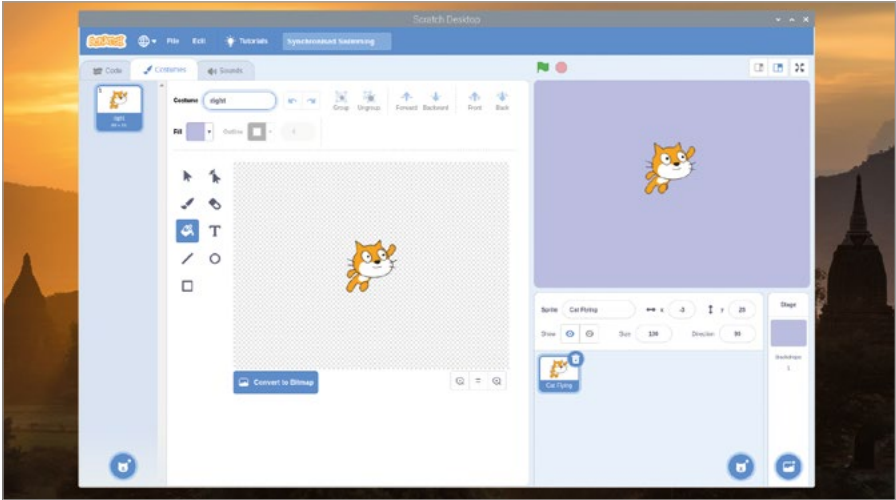
← キーと → キーを押して、プログラムの動作を確認してみましょう。それぞれの矢印キーで設定した動作に合わせて、猫が回転します。今回は、緑のフラグアイコンをクリックしなくてもプログラムを実行することができます。これは、プログラムが実行されていない状態でも、「...キーが押されたとき」イベントブロックが常にアクティブな状態になっているためです。

次に、上と同じ手順を 2 回繰り返します。ただし今回は、イベントブロックで「上向き矢印」と「下向き矢印」を選択し、これらのブロックの下に「**10 歩動かす**」ブロックと「**-10 歩動かす**」ブロックをドラッグします。これで、回転するだけでなく、前後に泳ぐこともできるようになりました。



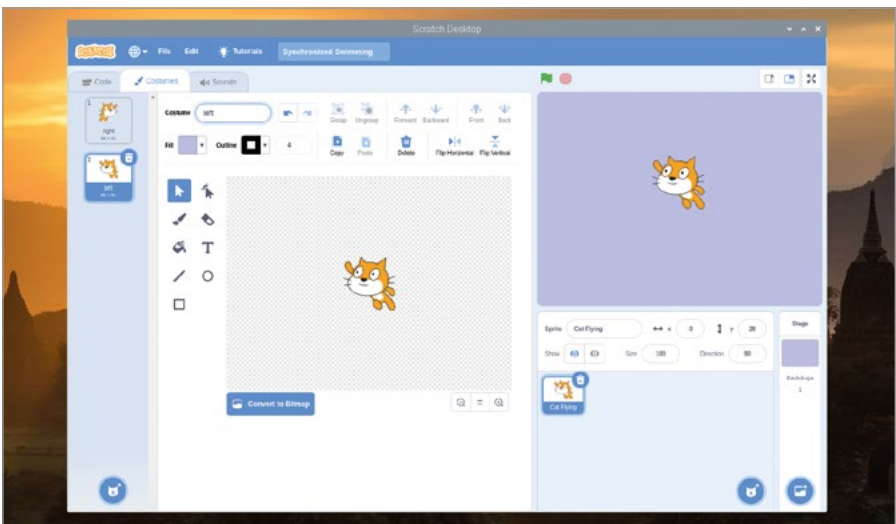
猫の動きをもっとリアルなものにするため、猫のコスチューム を変えてみましょう。猫をクリックし、ブロックパレットの上にある「コスチューム」タブをクリックします。「cat flying-a」というコスチュームをクリックし、このコスチュームの右上に表示されているゴミ箱アイコン (🗑️) をクリックします。これにより、「cat flying-a」というコスチュームが削除されます。次に、「cat flying-b」というコスチューム

をクリックし、「コスチューム」ボックスで「右」と入力します (図 4-17)。これが、このコスチュームの名前になります。



▲ 図 4-17: コスチュームの名前を「右」にする





新しく作成した「右」コスチュームを右クリックして「複製」を選択します。これにより、コスチュームのコピーが作成されます。このコピーをクリックして「選択」アイコン (🖱️) をクリックし、「左右反転」アイコン (🔄) を選択します。次に、コスチューム名として「左」と入力します (図 4-18)。これで、スプライト用の 2 つのコスチュームが完成しました。1 つは、猫が右を向いたときのコスチュームで、もう 1 つは、猫が左を向いたときのコスチュームです。



▲ 図 4-18: コスチュームを複製して左右反転し、「左」という名前を付ける

コスチューム領域の上にある「コード」タブをクリックし、「見た目」カテゴリーの 2 つの「コスチュームを 左 にする」ブロックを、「左向き矢印...」イベントブロックの下と「右向き矢印...」イベントブロックの下にそれぞれドラッグします。「右向き矢印...」イベントブロックにドラッグするブロックについては、「コスチュームを左にする」の「左」の部分を変更します（「コスチュームを 右 にする」）。この状態で矢印キーを押すと、猫が進行方向を向いて泳ぐようになります。



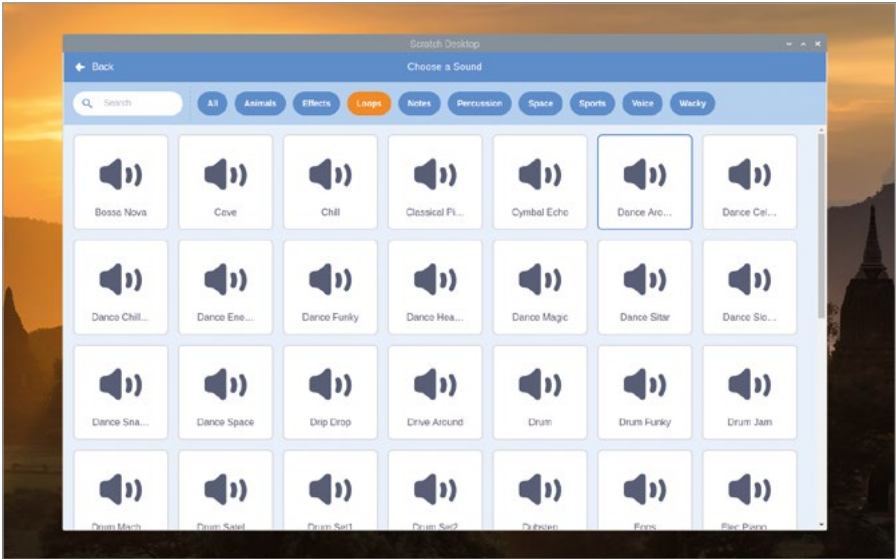
ここからは、猫の数を増やしていきましょう。そのためには、猫の位置をリセットする必要があります。「イベント」カテゴリの「が押されたとき」ブロックをコード領域に追加し、その下に「動き」カテゴリの「座標を 0、座標を 0 にする」ブロックと「90 度に向ける」ブロックをドラッグします(必要に応じてブロック内の値を変更してください)。この状態で緑のフラグアイコンをクリックすると、右を向いた猫がステージの中央に移動します。



次に、多くの猫をステージに追加してみましょう。「6 回繰り返す」ブロックをコード領域にドラッグし（「10」の部分を変更します）、このブロックの内側に「制御」カテゴリーの「自分自身のクローンを作る」ブロックをドラッグします。すべての猫が同じ方向に泳いでいけないように、「6 回繰り返す」ブロック内の「自分自身のクローンを作る」の上に「60 度回す」ブロックをドラッグし、「15」の部分を変更します。この状態で緑のフラグアイコンをクリックして上下左右の矢印キーを押すと、6 匹のネコが生き生きと泳ぎだします。



さらに臨場感を出すため、音楽を追加してみましょう。ブロックパレット上部の「音」タブをクリックし、画面左下の「音を選ぶ」アイコン (🔊) をクリックします。次に「ループ」カテゴリをクリックし、画面をスクロールしていずれかのサウンドを選択します (図 4-19)。各サウンドの右上にあるアイコンにマウスポインターを置くと、そのサウンドが再生されます。ここでは、「Dance Around」を選択します。音楽を選択したら、「コード」タブをクリックしてコード領域に戻ります。

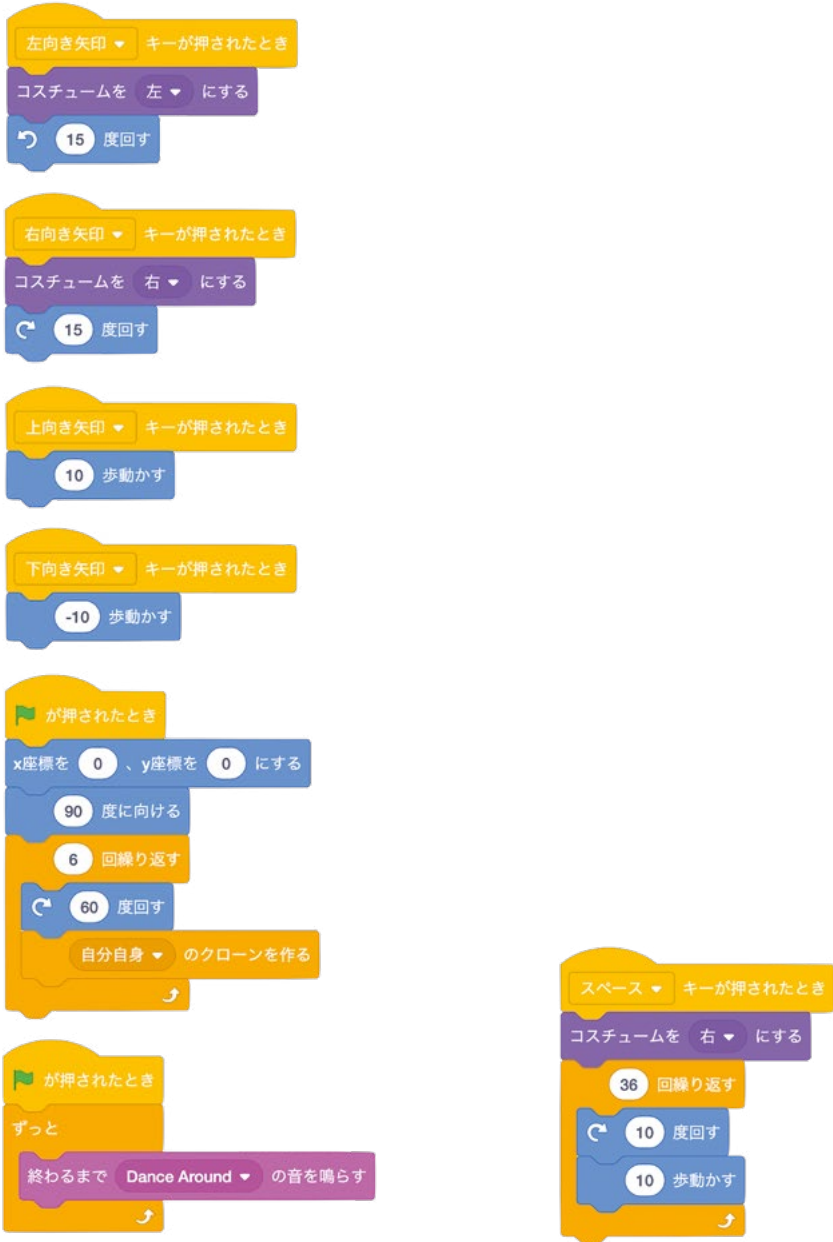


▲ 図 4-19: サウンドライブラリーでサウンドを選択する

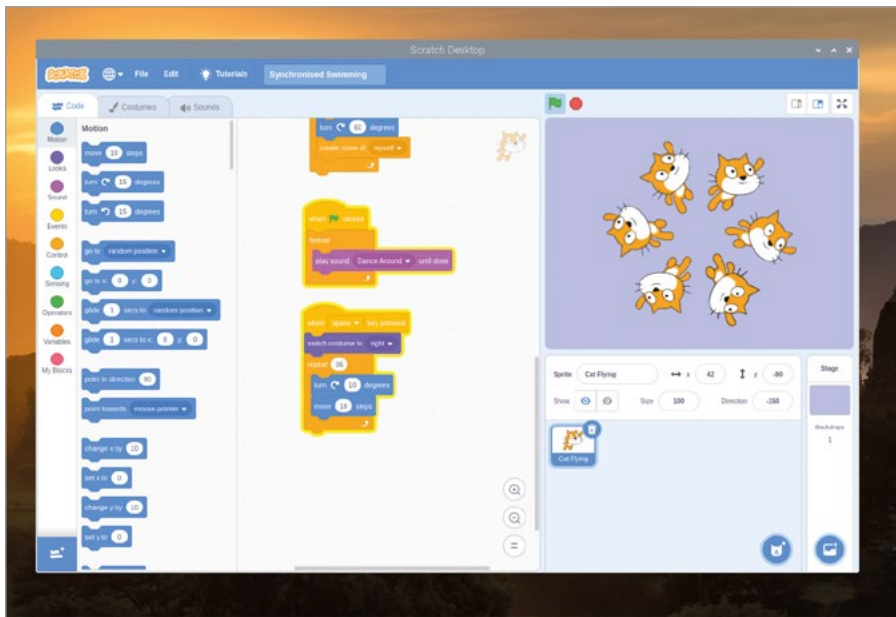
「イベント」カテゴリの「**緑の旗が押されたとき**」ブロックをコード領域に追加し、このブロックの下に「制御」カテゴリの「**ずっと**」ブロックをドラッグします。次に、この制御ブロックの内側に「**終わるまで dance の音を鳴らす**」ブロックをドラッグします。このブロックに、上の手順で選択したサウンドが表示されていることに注意してください。作業が完了したら、緑のフラグアイコンをクリックして、プログラムの動作を確認してみましょう。フラグアイコンの右側にある赤い八角形アイコンをクリックすると、プログラムが停止します。



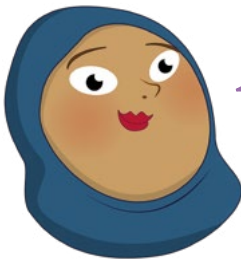
最後に、新しいイベントトリガーを追加して、このプログラムを完成させましょう。「イベント」カテゴリの「スペース キーが押されたとき」ブロックをコード領域に追加し、このブロックの下に「見た目」カテゴリの「コスチュームを 右 にする」ブロックをドラッグします。次に、その下に「36 回繰り返す」ブロックをドラッグし、「10」の部分を変更し、「36」に変更し、このブロックの内側に「10 度回す」ブロックと「10 歩動かす」ブロックをドラッグします。



緑のフラグアイコンをクリックしてプログラムを開始し、スペースキーを押して動作を確認してみましょう (図 4-20)。作業が終了したら、必ずプログラムを保存してください。



▲ 図 4-20: シンクロナイズドスイミングのルーティーンを完成させる



チャレンジ: 新しいルーティーンを作ってみよう ?

ループブロックを使用して、シンクロナイズドスイミングの新しいルーティーンを作ってみましょう。猫の数を増やしたり減らしたりするにはどうすればよいでしょうか？ キーボードの別のキーを使用して複数のルーティーンを追加することはできるでしょうか？

プロジェクト 3: アーチェリーゲーム

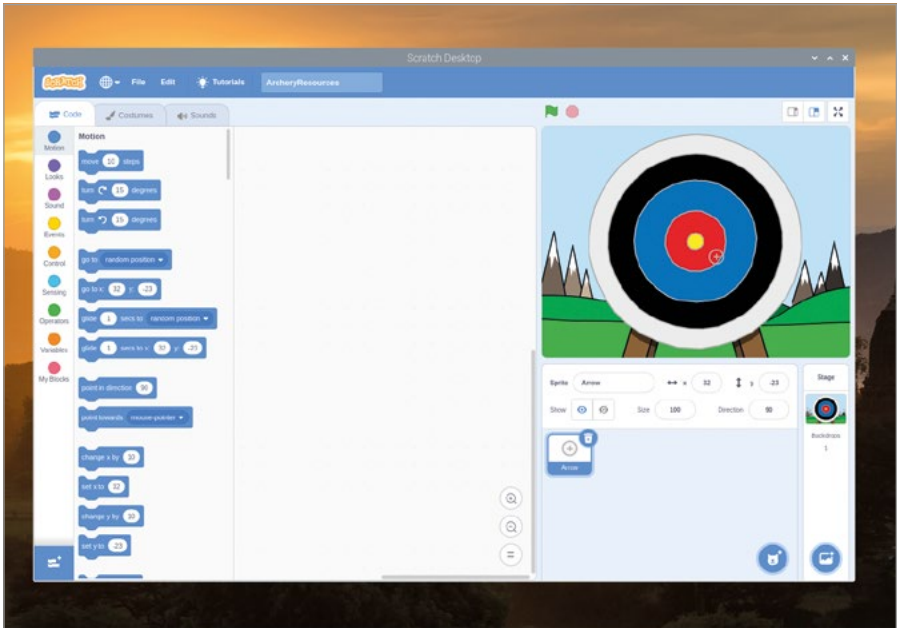
Scratch の操作にもだいぶ慣れてきたところで、少し複雑なアーチェリーゲームの作成に挑戦してみましょう。これは、不規則に動く弓と矢を使って標的を狙うというゲームです。

オンラインプロジェクト

このプロジェクトは、オンライン (rpf.io/archery) でも公開されています。

最初に、Chromium ブラウザーを開き、アドレスバーに「rpf.io/p/en/archery-go」と入力して ENTER キーを押します。ゲームのリソースが zip ファイルとしてダウンロードされたら、zip ファイルを右クリックして「ここに解凍」を選択してください。Scratch 3 に戻り、「ファイル」メニューをクリックして「コンピューターから読み込む」を選択します。次に、**ArcheryResources.sb3** を選択して「開く」ボタンをクリックします。現在のプロジェクトのコンテンツを置き換えるかどうかを確認するメッセージが

表示されます。まだ変更内容を保存していない場合は、「キャンセル」をクリックして変更内容を保存し、すでに保存されている場合は「OK」をクリックします。



▲ 図 4-21: アーチェリーゲーム用のリソースプロジェクト

読み込んだプロジェクトには背景とスプライトがすでに設定されていますが (図 4-21)、ゲームをプレイするためのコードは何もありませんので、あなた自身で追加していきましょう。最初に「**緑の旗が押されたとき**」ブロックを追加し、このブロックの下に「**メッセージ1を送る**」ブロックをつなげます。このブロックの「メッセージ1」の横にある下矢印をクリックして「新しいメッセージ」を選択し、メッセージ入力画面で「新しい矢」と入力して「OK」ボタンをクリックします。この時点で、ブロックは「**新しい矢を受け取ったとき**」のようになります。



「...を送る」ブロックを使用すると、プログラムの内部でメッセージをやり取りすることができます。実際に動作を実行するには、「**メッセージ1を受け取ったとき**」ブロックを追加する必要があります。このブロックについても、「メッセージ1」の下矢印をクリックして「新しい矢」を選択します (新しい矢を受け取ったとき)。今回は、リストに「新しい矢」が表示されるため、これを選択するだけです。メッセージの入力画面で「新しい矢」と入力する必要はありません。

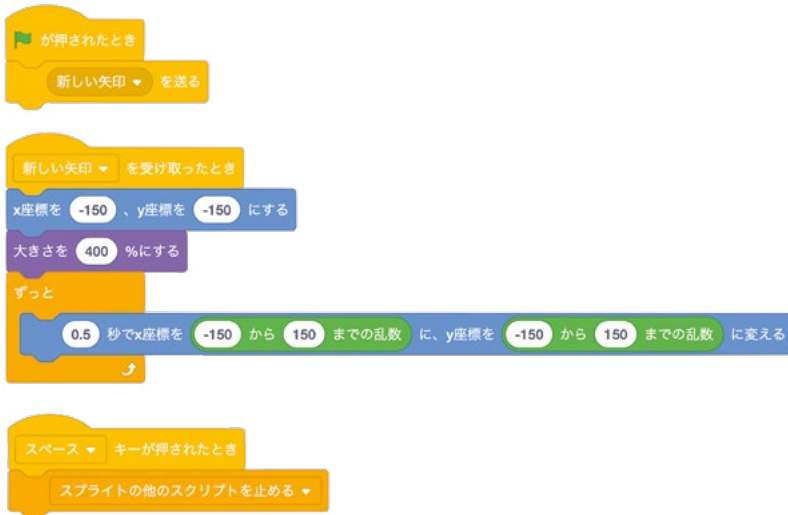
「新しい矢を受け取ったとき」ブロックの下に、「x座標を -150、y座標を -150 にする」ブロックと「大きさを 400 % にする」ブロックを追加します。これらのブロック内の値はデフォルト値ではないため、コード領域に追加してから、手動で変更する必要があります。この状態で緑のフラグアイコンをクリックすると、サイズが 4 倍になった矢がステージの左下に移動します。



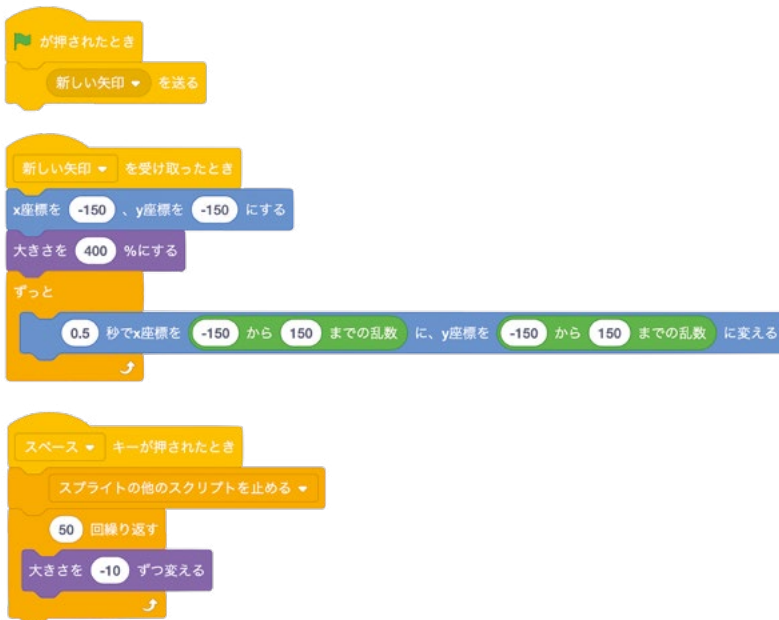
ゲームを難しくするために、弓を引いたときの揺れる動きのシミュレーションを追加してみましょう。「ずっと」ブロックをコード領域に追加し、このブロックの下に「1秒でx座標を -150 に、y座標を -150 にする」ブロックをドラッグします。左側の「0.5」という値を「1」に変更し、中央の値と右側の値にそれぞれ「-150 から 150 までの乱数」演算ブロックをドラッグして、各値を「-150」と「150」に変更します。これにより、矢がランダムな方向に移動するため（移動する距離もランダムです）、的を狙うのが難しくなります。



緑のフラグアイコンをクリックすると、矢が的全体をランダムに動き回ることがわかります。この時点では矢を的に向かって射るための方法がないので、この方法を追加していきましょう。「スペースキーが押されたとき」ブロックをコード領域に追加し、このブロックの下に「すべてを止める」制御ブロックをドラッグします。この制御ブロック内の下矢印をクリックし、「スプライトの他のスクリプトを止める」を選択します（「スプライトの他のスクリプトを止める」）。

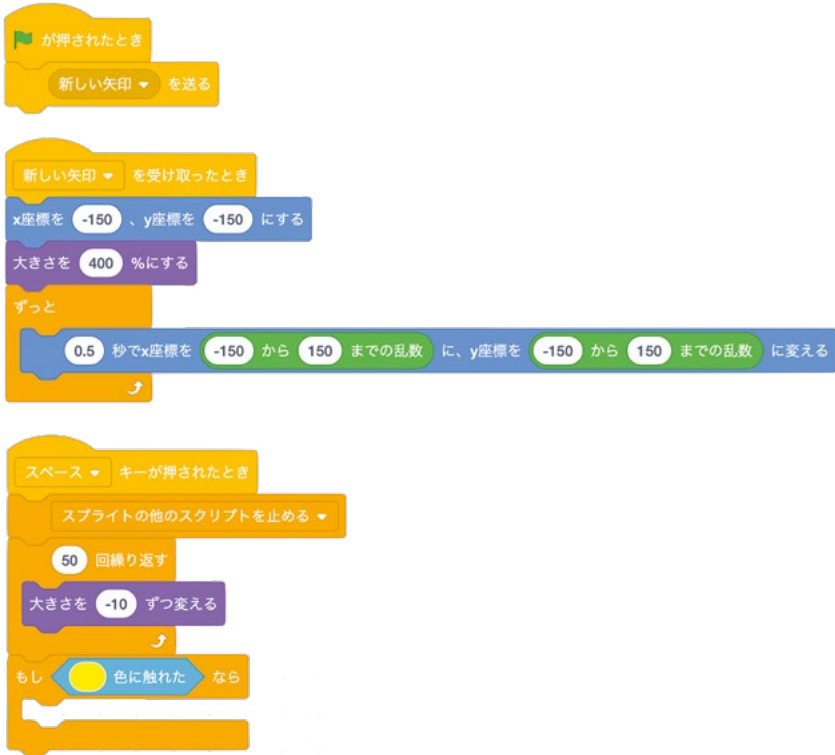


この状態で緑のフラグアイコンをクリックし、適当なタイミングでスペースキーを押すと、矢の動きが止まります。次に、矢が的に向かって飛んでいく動きを加えてみましょう。「50 回繰り返す」ブロックをコード領域に追加し、このブロックの下に「大きさを -10 ずつ変える」ブロックをドラッグして、緑のフラグアイコンをクリックします。矢がどんどん小さくなっていき、的に向かって飛んでいる様子が表現できました。



スコアを計算できるようにすると、ゲームがもっと楽しくなります。「50 回繰り返す」ブロックの下に「もし ... なら」ブロックをドラッグし、このブロックのダイヤモンド型のスペースに「調べる」カテゴリー

の「色に触れた」ブロックをドラッグします。次に、このブロックの色の部分をクリックして色オプションを表示し、このオプションの下部に表示されている アイコンをクリックして、ステージ領域に表示されている的の中心部 (黄色い部分) をクリックします。



矢が的の中心部に当たったことが音とスコアでわかるように、「cheer の音を鳴らす」ブロックと「200 ポイントと 2 秒言う」ブロックを「もし ... なら」ブロックの内側にドラッグします。最後に、「もし ... なら」ブロックの下に「新しい矢を受け取ったとき」ブロックをドラッグします。これで、プレイヤーが矢を放つたびに、次の新しい矢が表示されるようになります。では、緑のフラグアイコンをクリックしてゲームをプレイしてみましょう。矢がみごとに的の中心の黄色い部分に当たれば、大きな歓声があがり、200 ポイントを獲得することができます！



「付録 D:
参考資料」で紹介し
ている Scratch プロ
ジェクトも試してみま
しょう

このままでもゲームとしてプレイできますが、スコアを獲得するには少し難しすぎるかもしれません。この章で学習したスキルを応用して、このゲームを改良してみましょう。たとえば、的の中心部だけでなく、ほかの部分に矢が当たった場合もポイントを獲得できるようにすれば、もっとゲームが楽しくなります。赤い部分に当たった場合は 100 ポイント、青い部分に当たった場合は 50 ポイントなど、いろいろと工夫してみてください。



チャレンジ: 改良してみよう

このゲームをもっと簡単にするにはどうすればよいでしょうか？
このゲームをもっと難しくするにはどうすればよいでしょうか？
変数を使ってスコアを加算していくことはできるでしょうか？ 制限時間を決めてカウントダウンすることはできるでしょうか？

第5章

Python を使ってプログラミングしてみよう

Scratch について理解したところで、今度は Python を使ってテキストベースのコーディングを行ってみましょう。

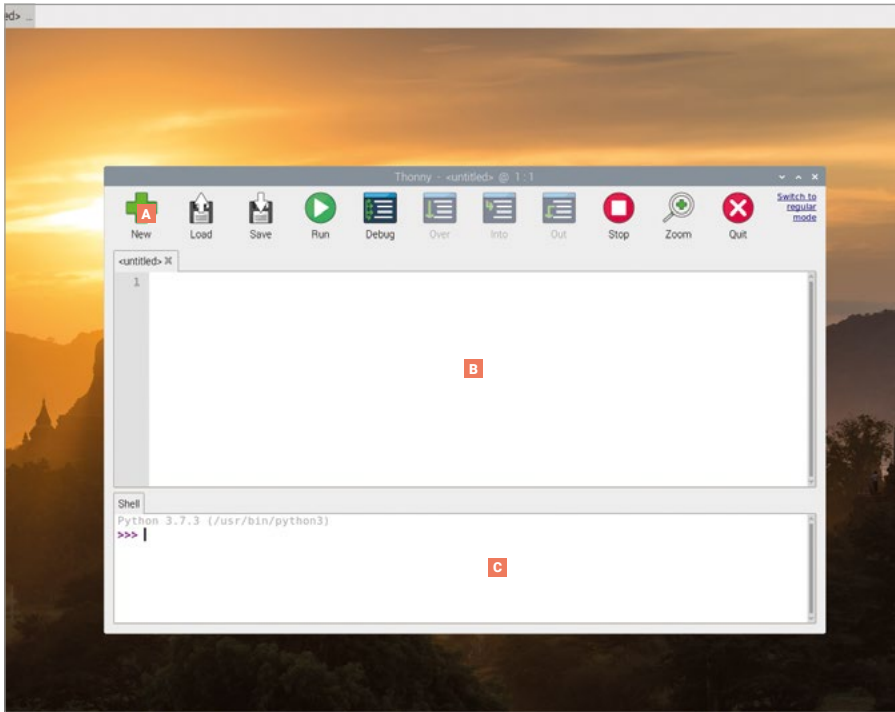


モンティ・パイソンというコメディグループにちなんで名付けられた Python は、ガイド・ヴァンロッサムが趣味のプロジェクトとして 1991 年に一般公開した後、幅広いプロジェクトを推進するプログラミング言語として成長し、多くの人々に愛されています。Scratch の視覚的な環境と異なり、Python はテキストベースです。シンプルな言語と特定のフォーマットを使用して命令を記述し、コンピューターで実行します。

Python は、すでに Scratch を使用しているユーザーの次のステップで、柔軟性が向上した、より「従来型」のプログラミング環境を提供します。ただし、学習が難しいわけではありません。少し練習すれば、簡単な計算から驚くほど複雑なゲームに至るまで、だれでも Python プログラムを作成できます。

この章は、「第 4 章 Scratch 3 を使ってプログラミングしてみよう」で紹介した用語と概念に基づいています。第 4 章の演習を完了していない場合は、まず第 4 章の演習を完了することをお勧めします。そうすることで、この章の内容が理解しやすくなります。

Thonny Python IDE の紹介



- A ツールバー** – Thonny の「Simple Mode」インターフェイスには、メニューとしてわかりやすいアイコンバーが用意されており、Python プログラムを作成、保存、読み込み、実行できるほか、さまざまな方法でテストを実施できます。
- B スクリプト領域** – スクリプト領域は Python プログラムを記述する場所です。プログラムを表示するメイン領域と、行番号を表示する細長い領域とに分かれています。
- C Python シェル** – Python シェルには個々の命令を入力できます。これらの命令は **ENTER** キーを押すとすぐに実行されます。実行中のプログラムに関する情報も表示します。

Thonny のバージョン

Thonny には 2 つのインターフェイスバージョンがあります。「Regular Mode」と、初心者向けの「Simple Mode」です。この章では「Simple Mode」を使用します。これは raspberry メニューの「プログラミング」セクションから Thonny を開くと、デフォルトで読み込まれます。

Python で「こんにちは!」というプログラムを作成する

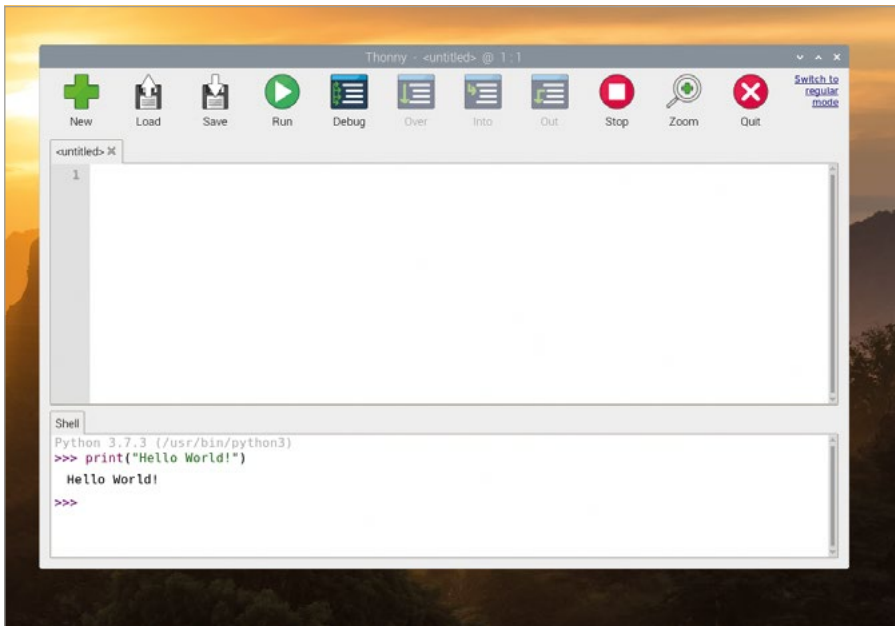
Raspberry Pi にプリインストールされている他のプログラムと同様、Thonny はメニューから利用できます。raspberrypi アイコンをクリックし、カーソルを「プログラミング」セクションに移動して、「Thonny Python IDE」をクリックします。クリックしてから数秒後、Thonny のユーザーインターフェイスが (デフォルトでは「Simple Mode」で) 表示されます。

Thonny は統合開発環境 (IDE) と呼ばれているパッケージです。難しそうな名前ですが、ソフトウェアの記述 (開発) に必要なあらゆるツールを 1 つのユーザーインターフェイス (環境) に集約する (統合) ことを表しています。利用可能な IDE は多数存在します。多くの異なるプログラミング言語をサポートするものもあれば、Thonny のように単一の言語をサポートするものもあります。

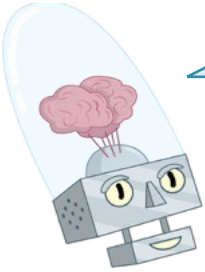
プログラムのベースとして視覚的な構成ブロックを提供する Scratch とは異なり、Python はすべてを記述するという点で、より従来型のプログラミング言語です。1 つ目のプログラムを作成しましょう。Thonny ウィンドウの下部にある Python のシェル領域をクリックし、以下の命令を入力して ENTER キーを押してください。

```
print("こんにちは!")
```

ENTER キーを押すと、プログラムがすぐに実行されます。Python はあなたが入力した「こんにちは!」というメッセージを同じシェル領域に表示します (図 5-1)。これはシェルが Python のインタプリタに直接指示しているためです。インタプリタはあなたの指示を確認し、その意味を解釈します。これは誰かと話をするように、あなたが会話を終わると反応し、あなたが次の話をするまで待機するので対話モードと呼ばれます。



▲ 図 5-1: Python のシェル領域に「こんにちは!」と表示されます




構文エラー

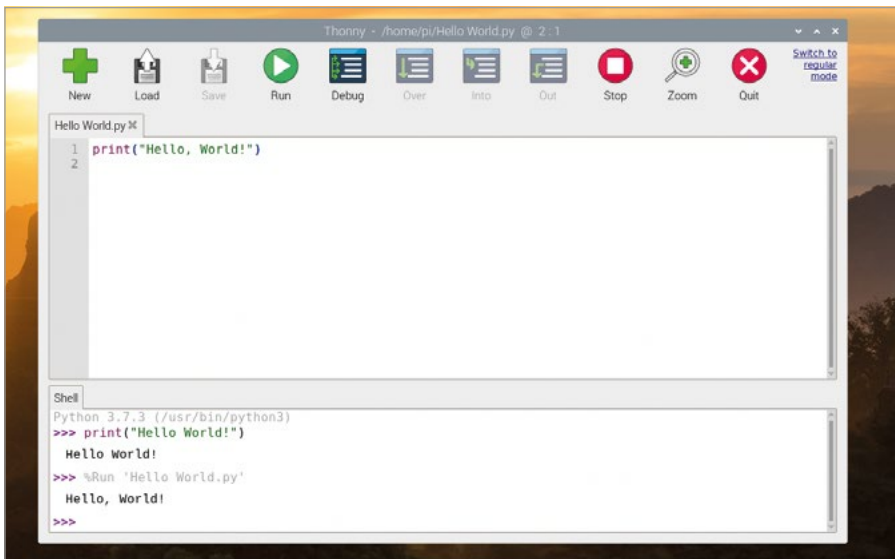
プログラムが実行されず、「syntax error」メッセージがシェル領域に出力される場合は、記述内容に誤りがあります。Python は命令の記述方法が厳密です。角かっこや引用符は漏れがないように正確に記述する必要があります。命令は大文字小文字の区別を含め正確に記述する必要があります。たとえば、「Print」ではなく「print」と記述しなければなりません。また、命令内に不要な記号を含めないようにする必要があります。これらの方法に従っていないと、プログラムは実行されません。もう一度命令を入力し、この本の命令と一致していることを確認してから **ENTER** キーを押してみてください。

ただし、Python を対話モードで使用しなければならないわけではありません。Thonny ウィンドウの中央にあるスクリプト領域をクリックし、プログラムをもう一度入力します。

```
print("こんにちは!")
```

ENTER キーを押すと、今度はスクリプト領域に新しい空白行のみが表示されます。この形式でプログラムを動作させるには、Thonny のツールバーで「Run」アイコン  をクリックする必要があります。クリックすると、最初にプログラムを保存するように求められます。「Hello World」などのわかりやすい名前を入力して、「Save」ボタンをクリックします。プログラムが保存されると、Python のシェル領域に 2 つのメッセージが表示されます (図 5-2)。

```
>>> %Run 'Hello World.py'
こんにちは!
```



▲ 図 5-2:簡単なプログラムの実行

1 つ目の行は Thonny の命令です。保存したプログラムを実行するように Python のインタープリターに指示しています。2 つ目の行はプログラムの出力です。あなたが Python に出力するように指示したメッセージが表示されます。これで、1 つ目の Python プログラムを作成し、対話モードとスクリプトモードの両方で実行しました。




チャレンジ: 新しいメッセージ



Python プログラムが出力するメッセージを変更できますか? メッセージをさらに追加する場合、対話モードとスクリプトモードのどちらを使用しますか? プログラムから角かっこや引用符を削除した後、プログラムをもう一度実行するとどうなりますか?

次のステップ: ループとコードのインデント

Scratch がジグソーパズルのようなブロックを使用して、プログラムのどの部分をどの部分と組み合わせるかを制御するのと同様、Python にもプログラムの実行シーケンスを制御する独自の方法があります。インデントです。Thonny のツールバーで「New」アイコン  をクリックし、新しいプログラムを作成しましょう。既存のプログラムは失われません。代わりに、Thonny はスクリプト領域の上に新しいタブを作成します。以下のように入力します。

```
print("ループ開始!")  
for i in range(10):
```

最初の行は、「こんにちは」プログラムと同様に、シェルに簡単なメッセージを出力します。2 つ目の行は有限ループを開始します。これは Scratch を使用する場合と同様に機能します。**i** カウンターがループに割り当てられ、一連の数値が与えられます。これらの数値は 0 から 10 未満の間で増加するよう **range** 命令によって指示されています。コロンの記号 (:) は、次の命令がループの一部であることを Python に指示します。

Scratch では、ループに含まれる命令は文字どおり C 型ブロック内に含まれていますが、Python ではインデントを使って表します。次の行は 4 つの空白スペースで始まります。Thonny では 2 行目の終わりに **ENTER** キーを押すと、これらのスペースが追加されます。

```
print("ループ回数", i)
```

空白スペースによってこの行は他の行よりも内側に配置されます。Python ではこのインデントによってループ外の命令とループ内の命令が区別されます。インデントされたコードは、ネストされたコードと呼ばれます。

3 行目の終わりに **ENTER** キーを押すと、Thonny は次の行がループの一部であると想定し、自動的にインデントします。これを削除するには、4 行目を入力する前に **BACKSPACE** キーを 1 回押します。

```
print("ループ終了!")
```

これで 4 行のプログラムが完成しました。最初の行はループの外側にあるため 1 回だけ実行されます。2 行目はループを設定します。3 行目はループ内にあり、ループがループするたびに 1 回実行されます。4 行目は再びループの外側にあります。

```
print("ループ開始!")  
for i in range(10):  
    print("ループ回数", i)  
print("ループ終了!")
```

「Run」アイコンをクリックし、プログラムを「Indentation」という名前で保存し、シェル領域に出力を表示します (図 5-3)。

ループ開始!

ループ回数 0

ループ回数 1

ループ回数 2

ループ回数 3

ループ回数 4

ループ回数 5

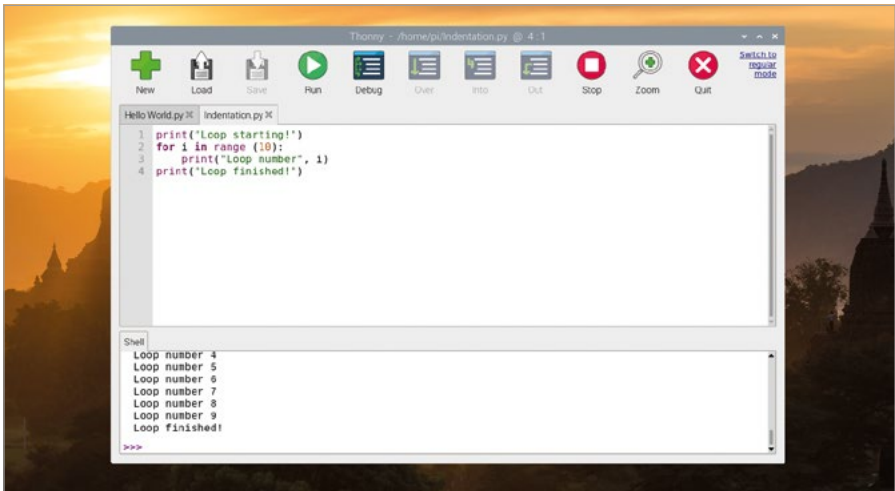
ループ回数 6

ループ回数 7

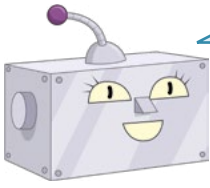
ループ回数 8

ループ回数 9

ループ終了!



▲ 図 5-3:ループの実行



ゼロからカウントする

Python はインデックスがゼロの言語です。つまり、1 からではなく 0 からカウントを開始します。そのため、プログラムでは 1 から 10 ではなく 0 から 9 の数値が出力されます。

必要に応じて、`range(10)` 命令を `range(1, 11)` またはその他の任意の数値に切り替えることで、この動作を変更することができます。

インデントは Python の重要な要素ですが、プログラムが期待どおりに機能しない原因にもなります。プログラムで問題の原因を特定する (デバッグと呼ばれます) ときは、必ずインデントを確認するようにしてください。特にループ内にループをネストしているときは注意してください。

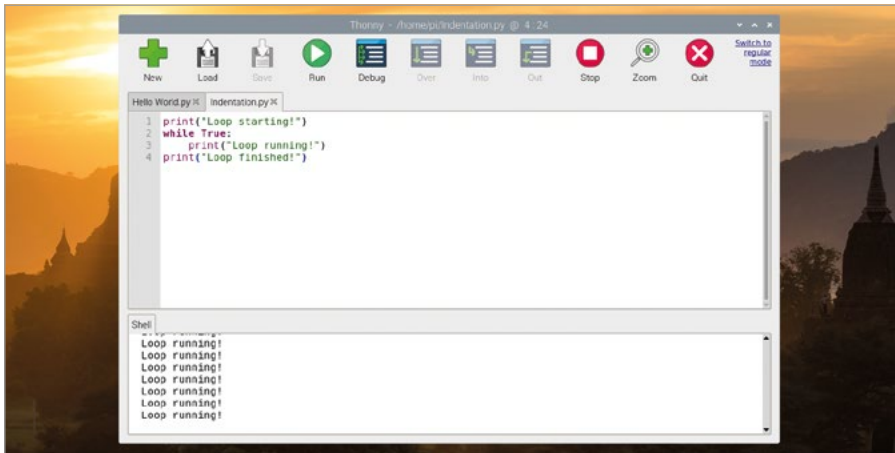
Python は永続的に実行される無限ループもサポートしています。プログラムを有限ループから無限ループに変更するには、2 行目を以下のように編集します。

```
while True:
```


「Run」アイコンをクリックすると、「**name 'i' is not defined**」というエラーが表示されます。これは変数 `i` を作成して値を割り当てた行を削除したためです。これを修正するには、変数を使用しないように 3 行目を編集します。

```
print("ループ実行中!")
```

「Run」アイコンをクリックすると、「ループ開始!」のメッセージがすぐに表示され、その後に「ループ実行中!」のメッセージが永続的に表示されます (図 5-4)。ループは無限に繰り返されるため、「ループ終了!」のメッセージは出力されません。Python が「ループ実行中!」のメッセージの出力を終了すると、ループの先頭に戻って再度出力が行われます。



▲ 図 5-4: プログラムを停止するまで無限ループは続きます


Thonny のツールバーで「Stop」アイコン  をクリックすると、プログラムが実行を停止します。これはプログラムの中断と呼ばれます。Python のシェル領域にメッセージが表示され、4 行目に到達することなくプログラムは停止します。



チャレンジ: ループをループする

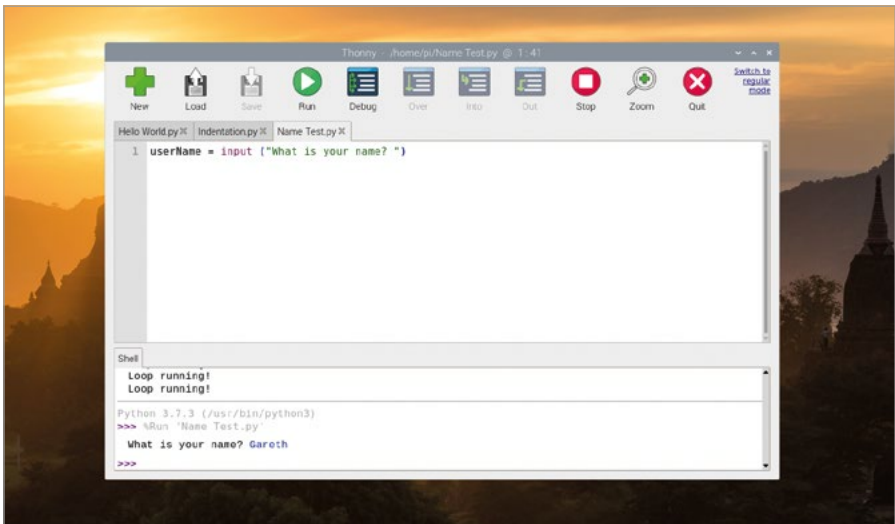
ループを再び有限ループに戻すことはできますか? プログラムに 2 つ目の有限ループを追加できますか? ループ内にループを追加するにはどうしますか? 追加するとどのように機能すると思いますか?

条件と変数

すべてのプログラミング言語と同様、変数の役割はループを制御することだけではありません。新しいプログラムを開始しましょう。Thonny のメニューで「New」アイコン  をクリックし、スクリプト領域に以下を入力してください。

```
userName = input ("あなたの名前は? ")
```

「Run」アイコンをクリックし、プログラムを「Name Test」という名前で保存します。あなたの名前の入力を求めるメッセージがシェル領域に表示されます。シェル領域に名前を入力し、ENTER キーを押します。これがプログラムの唯一の命令であるため、名前を入力しても何も起こりません (図 5-5)。変数に指定したデータを使って何か実行させる場合は、プログラムに追加の行を入力する必要があります。



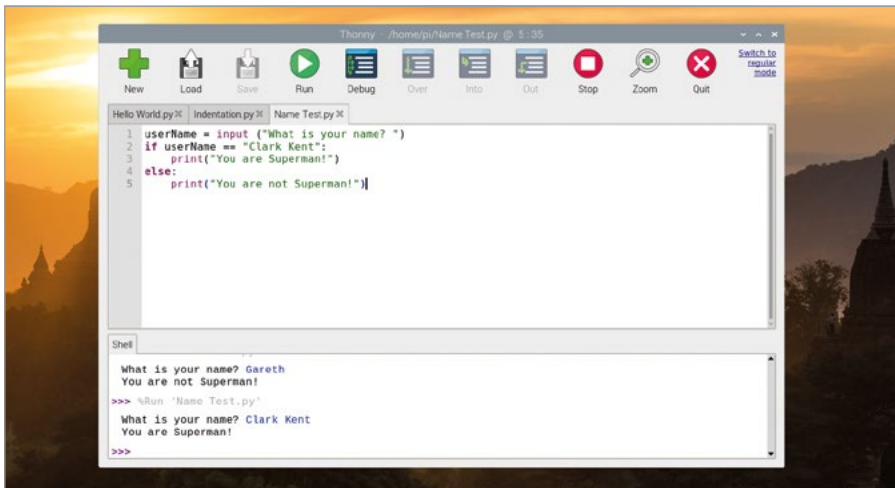
▲ 図 5-5: input 関数を使用すると、ユーザーにテキスト入力を求めることができます

名前を使用するプログラムを作成してみましょう。以下のように入力し、条件文を追加してください。

```
if userName == "Clark Kent":
    print("あなたはスーパーマンです!")
else:
    print("あなたはスーパーマンではありません!")
```

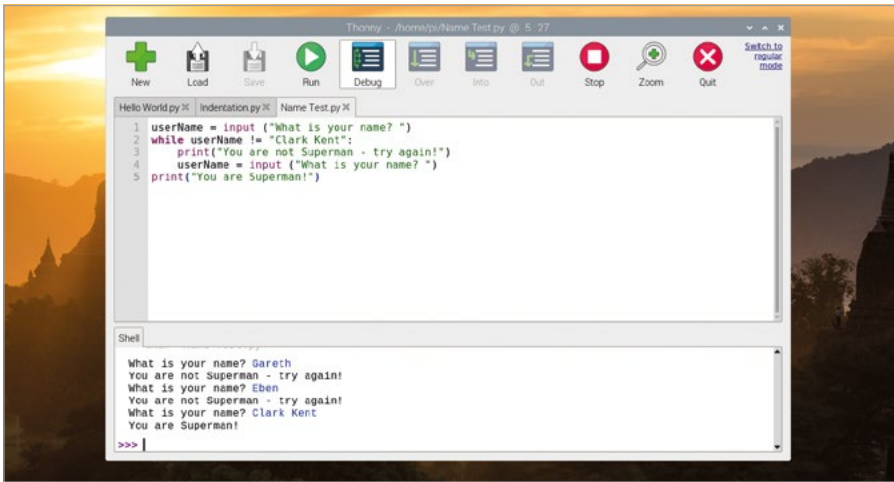
Thonny はコードのインデントが必要であることを確認すると自動的にインデントを追加しますが、どこでインデントを終了するかを特定することはできません。そのため、自分でスペースを削除する必要があります。

「Run」アイコンをクリックし、シェル領域に名前を入力します。あなたの名前が Clark Kent でない場合、「あなたはスーパーマンではありません!」というメッセージが表示されます。もう一度「Run」をクリックし、今度は「Clark Kent」と入力します。大文字と小文字の区別を含め (C と K は大文字にします)、プログラムに記述した名前と正確に一致するように入力してください。今回はあなたがスーパーマンであると認識されます (図 5-6)。

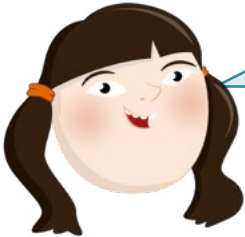


▲ 図 5-6: 人類を救うため世界中を飛び回っているのでは?

==記号は、`userName` 変数がプログラム内のテキスト (文字列と呼ばれます) と等しいかどうかを直接比較するように Python に指示します。このほかにも比較を行うことができます。>を使用すると、数値が別の数値より大きいかどうかを比較できます。<を使用すると、数値が別の数値より小さいかどうかを比較できます。=>を使用すると、数値が別の数値以上であるかどうかを比較できます。=<を使用すると、数値が別の数値以下であるかどうかを比較できます。!=も使用でき、数値が別の数値と等しくないことを意味します。つまり、==と逆の意味になります。これらの記号は、厳密には比較演算子と呼ばれます。



▲ 図 5-7: 「Clark Kent」と入力されるまで名前を質問し続けます



「=」と「==」の使用

変数を使用する場合は、`=`と`==`の違いを学習することが重要です。注：`=`は「変数をこの値と等しくする」ことを意味し、`==`は「変数がこの値と等しいかどうかを確認する」ことを意味します。これらを混同してプログラムを作成すると、プログラムが機能しない可能性があります。

比較演算子はループでも使用できます。2 行目から 5 行目を削除して以下を入力してください。

```
while userName != "Clark Kent":
    print("あなたはスーパーマンではありません - もう一度やり直してください!")
    userName = input("あなたの名前は? ")
    print("あなたはスーパーマンです!")
```

「Run」アイコンをもう一度クリックします。今度はあなたがスーパーマンであることが確認されるまで、プログラムは終了せずに名前を質問し続けます (図 5-7)。非常に単純なパスワードのように動作します。ループから抜け出すには、「Clark Kent」と入力するか、Thonny のツールバーで「Stop」アイコンをクリックします。おめでとうございます! これで、条件と比較演算子の使い方について理解しました!



チャレンジ: 質問を増やす

複数の質問を表示するようにプログラムを変更し、複数の変数に回答を保存できますか? 「第 4 章 Scratch 3 を使ってプログラミングしてみよう」で作成したプログラムと同様に、ユーザーが入力した数値が 5 より大きい小さいかを出力するプログラムを条件と比較演算子を使って作成できますか?



プロジェクト 1: タートルによる雪の結晶


Python がどのように機能するかを理解したところで、今度はグラフィックを操作してみましょう。タートルと呼ばれるツールを使用して雪の結晶を作成します。

オンラインプロジェクト

このプロジェクトは、オンライン (rpf.io/turtle-snowflakes) でも公開されています。



当初タートルは動物名のカメと同じ形をした物理的なロボットで、直線に移動し、方向転換し、ペンを上下するように設計されていました。デジタルバージョンのタートルでは、直線の描画の開始と停止を指示します。Logo 言語などの一部の言語と異なり、Python にはタートルツールが組み込まれていませんが、タートルの機能を提供するアドオンコードのライブラリーが付属しています。ライブラリーは Python の機能を拡張する新しい命令を追加するためのコードバンドルです。import コマンドを使用して、あなたの独自のプログラムにコードを取り込みます。

新しいプログラムを作成しましょう。「New」アイコン  をクリックし、以下のように入力してください。

```
import turtle
```

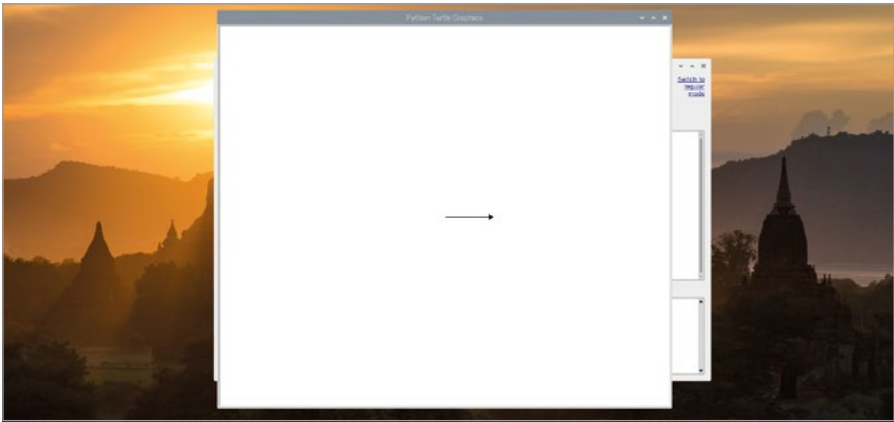
ライブラリーに含まれている命令を使用するときは、ライブラリー名、ピリオド、命令名の順に入力する必要があります。毎回入力するのは大変なので、代わりに短い変数名を割り当てることができます。1文字だけでも構いませんが、タートルにペット名を付けることにしましょう。以下のように入力します。

```
pat = turtle.Turtle()
```

プログラムをテストするには、タートルに何か指示する必要があります。以下のように入力してください。

```
pat.forward(100)
```

「Run」アイコンをクリックし、プログラムを「Turtle Snowflakes」という名前で保存します。プログラムが保存されると、「Turtle Graphics」という新しいウィンドウが表示され、プログラムの実行結果が表示されます。Pat (タートル) は 100 歩前に前進して直線を描いています (図 5-8)。



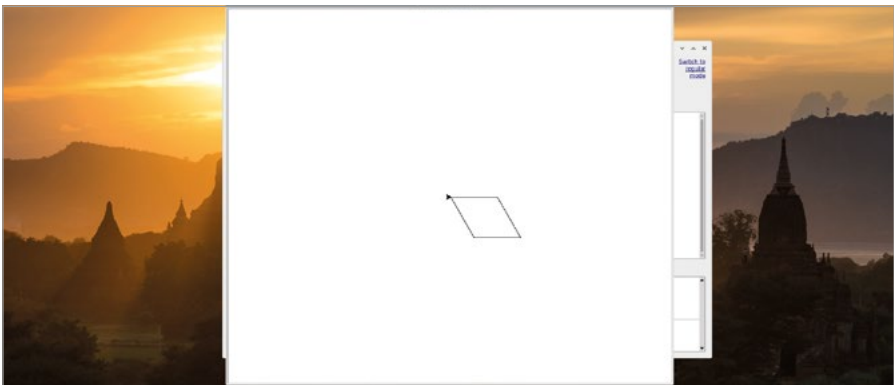
▲ 図 5-8: タートルが前進して直線を描きます

メインの Thonny ウィンドウに戻ります。このウィンドウが「Turtle Graphics」ウィンドウの後ろに隠れている場合は、「Turtle Graphics」ウィンドウの最小化ボタンをクリックするか、画面上部のタスクバーにある Thonny エントリーをクリックします。「Turtle Graphics」ウィンドウを閉じる場合は、「Stop」ボタンをクリックします。

手作業ですべての移動命令を入力するのは大変なので、3 行目を削除し、図形を作成するループを作成します。

```
for i in range(2):
    pat.forward(100)
    pat.right(60)
    pat.forward(100)
    pat.right(120)
```

プログラムを実行すると、Pat は平行四辺形を 1 つ描きます (図 5-9)。



▲ 図 5-9: 方向変換と移動を組み合わせることで図形を描くことができます

これを雪の結晶のような図形にするには、Thonny のメインウィンドウの「Stop」アイコンをクリックし、3 行目に以下の行を追加して、ループの外側にループを作成します。

```
for i in range(10):
```

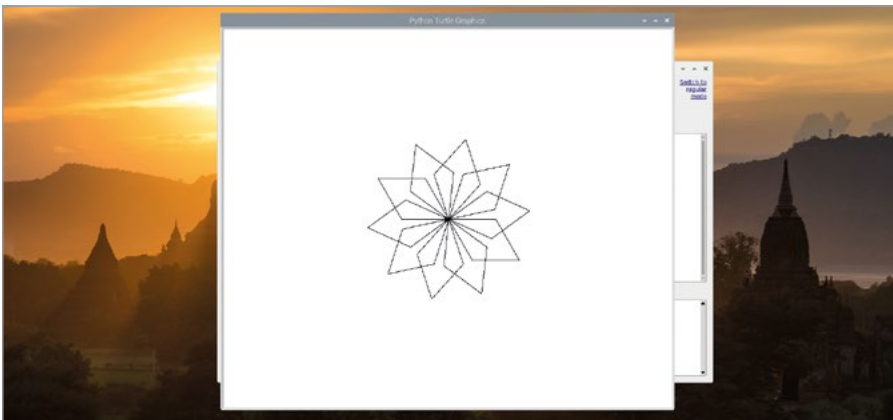
…プログラムの最後に以下を追加します。

```
pat.right(36)
```

既存のループが正しくインデントされていないため、プログラムはこのままでは実行されません。これを修正するには、既存のループの各行の先頭 (4 行目から 8 行目) をクリックし、**SPACE** キーを 4 回押してインデントを修正します。プログラムは以下のようになります。

```
import turtle
pat = turtle.Turtle()
for i in range(10):
    for i in range(2):
        pat.forward(100)
        pat.right(60)
        pat.forward(100)
        pat.right(120)
    pat.right(36)
```

「Run」アイコンをクリックし、タートルの動きを確認します。前回と同じように平行四辺形が描かれますが、1 つの平行四辺形が完了すると、36 度回転して別の平行四辺形が描かれます。この動作は画面に重なり合った平行四辺形が 10 個描かれるまで続き、雪の結晶のような図形が作成されます (図 5-10)。

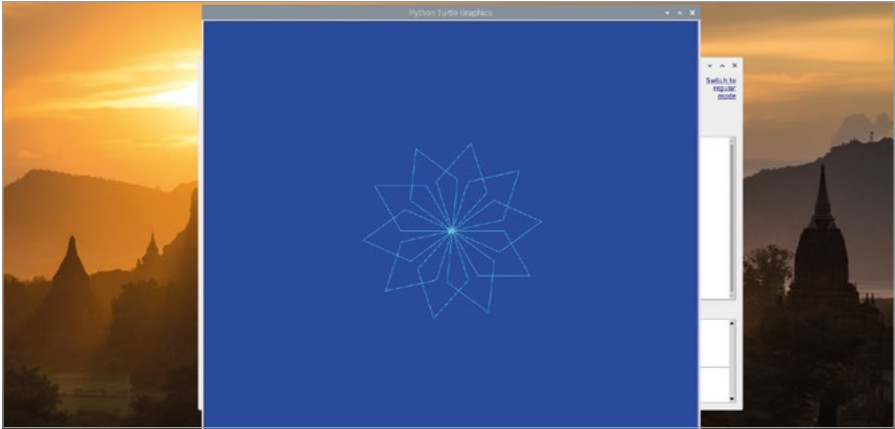


▲ 図 5-10: 図形の作成を繰り返すことでより複雑な形状を作成します

ロボットのタートルは大きな紙に単色で描画しますが、Python のシミュレーションタートルではさまざまな色を使用できます。3 行目と 4 行目に以下の行を新しく追加します。

```
turtle.Screen().bgcolor("blue")
pat.color("cyan")
```

プログラムを再度実行すると、新しいコードを実行した結果を確認できます。「Turtle Graphics」ウィンドウの背景色が青に変わり、雪の結晶がシアンで表示されます (図 5-11)。



▲ 図 5-11:背景と雪の結晶の色が変わります

random ライブラリーを使用して、リストからランダムに色を選択することもできます。プログラムの先頭に戻り、2 行目に以下を挿入します。

```
import random
```

4 行目で指定されている背景色を「blue」から「grey」に変更し、5 行目に新しい行を挿入して、「colours」という新しい変数を作成します。

```
colours = ["cyan", "purple", "white", "blue"]
```



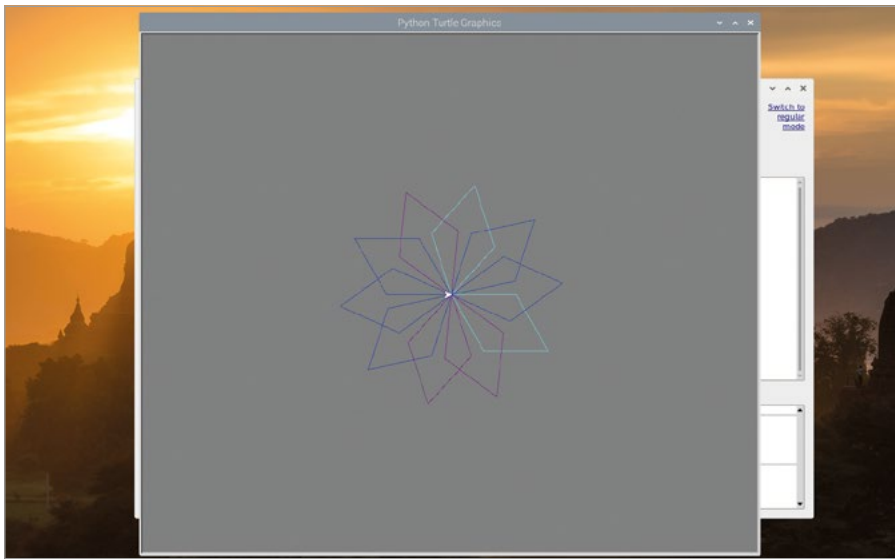
アメリカ英語のスペル

多くのプログラミング言語はアメリカ英語のスペルを使用しています。Python も例外ではありません。タートルのペンの色を変えるコマンドのスペルは `color` です。イギリス英語の `colour` を使用すると機能しません。ただし、変数には任意のスペルを使用できます。そのため、新しい変数 `colours` を呼び出して Python に理解させることができます。

このタイプの変数はリストと呼ばれ、角かっこを使用します。この場合、リストには雪の結晶を構成する並行四辺形の色を指定します。ただし、ループが繰り返されるたびに色を 1 つ選択するように Python に指示する必要があります。プログラムの最後に以下を入力します。上の行のように 4 つのスペースを使ってインデントし、外側のループの一部を形成するようにしてください。

```
pat.color(random.choice(colours))
```

「Run」アイコンをクリックすると、手裏剣にも似た雪の結晶の図形が再び描かれます。ただし、今度は個々の平行四辺形を描くときにリストから色がランダムに選択され、雪の結晶がマルチカラーで美しく描かれます (図 5-12)。



▲ 図 5-12: 平行四辺形にランダムな色が適用されます

手裏剣のような雪の結晶をより本物に近づけるため、**colours** リストのすぐ下の 6 行目に新しい行を追加し、以下のように入力します。

```
pat.penup()  
pat.forward(90)  
pat.left(45)  
pat.pendown()
```

ロボットのタートルを使用する場合は、**penup** 命令と **pendown** 命令によって紙面上で物理的なペンを移動させますが、仮想世界では線の描画の開始や停止をタートルに指示します。ただし今回は、ループを使用するのではなく関数を作成します。関数はいつでも呼び出すことができるコードセグメントです。関数を使用すると、独自の Python 命令を作成することもできます。

まず平行四辺形ベースの雪片を描くコードを削除しましょう。これは 10 行目の `pat.color("cyan")` 命令から 17 行目の `pat.right(36)` 命令までのすべての行が対象になります。`pat.color(random.choice(colours))` 命令は残しますが、行の先頭にハッシュ記号 (#) を追加します。これは命令のコメントアウトと呼ばれ、Python がこの命令を無視することを意味します。コメントを使用するとコードの説明を追加できます。これにより、後でコードを確認したり他のユーザーにコードを送信したりするときに、コードの理解が簡単になります。

「branch」という独自の関数を作成しましょう。`pat.pendown()`: の下の 10 行目に、以下の命令を入力します。

```
def branch():
```

この命令は `branch` 関数を定義します。`ENTER` キーを押すと、Thonny は関数の命令に自動的にインデントを追加します。以下のように入力します。インデントに注意してください。一部のコードはインデントが 3 段階深くなっています。

```
for i in range(3):
    for i in range(3):
        pat.forward(30)
        pat.backward(30)
        pat.right(45)
    pat.left(90)
    pat.backward(30)
    pat.left(45)
pat.right(90)
pat.forward(90)
```

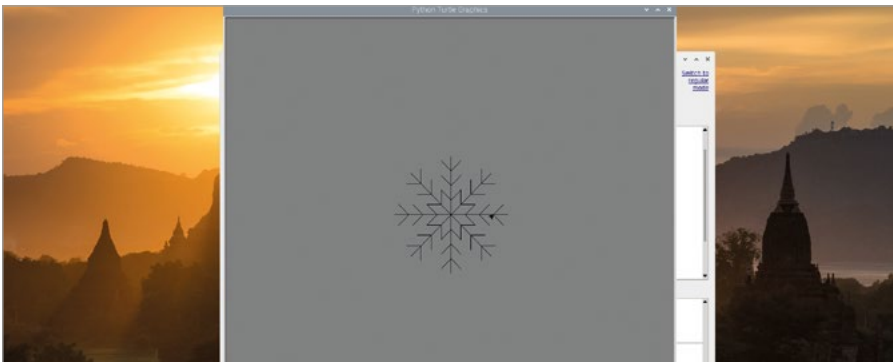
最後に、プログラムの末尾 (ただし上記でコメントアウトされている行の上) に新しいループを作成し、新しい関数を実行します (呼び出しと呼ばれます)。

```
for i in range(8):
    branch()
    pat.left(45)
```

完成したプログラムは以下のようになります。

```
import turtle
import random
pat = turtle.Turtle()
turtle.Screen().bgcolor("grey")
colours = ["cyan", "purple", "white", "blue"]
pat.penup()
pat.forward(90)
pat.left(45)
pat.pendown()
def branch():
    for i in range(3):
        for i in range(3):
            pat.forward(30)
            pat.backward(30)
            pat.right(45)
        pat.left(90)
        pat.backward(30)
        pat.left(45)
    pat.right(90)
    pat.forward(90)
for i in range(8):
    branch()
    pat.left(45)
# pat.color(random.choice(colours))
```

「Run」をクリックし、「Turtle Graphics」ウィンドウ上で Pat があなたの指示に従って描画する様子を確認します。これで、雪の結晶がより本物らしく描かれるようになりました (図 5-13)。



▲ 図 5-13: branch を追加することでより本物らしい雪の結晶を描くことができます



チャレンジ: 雪の結晶を描く



コメントアウトした指示を使用して、雪の結晶の雪片をさまざまな色で描くことができますか? 「snowflake」関数を作成し、この関数を使ってたくさんの雪の結晶を画面の上に描くことができますか? 雪の結晶のサイズと色をランダムに変えるようにプログラムを変更できますか?

プロジェクト 2: 恐怖の間違い探し

Python はタートルベースのグラフィックだけでなく写真やサウンドを処理することもできます。これらはゲームに利用すると非常に効果的です。ハロウィーンに最適な恐怖の間違い探しゲームを作成して、友達を驚かせましょう。

オンラインプロジェクト



このプロジェクトは、オンライン (rpf.io/scary-spot) でも公開されています。

このプロジェクトには 2 つの画像 (間違い探し用の画像とサブライズ用の「恐ろしい」画像) と 1 つのサウンドファイルが必要になります。raspberrry アイコンをクリックして Raspberry Pi OS のメニューを読み込み、「Internet」カテゴリーを選択して、Chromium Web ブラウザーをクリックします。ブラウザーが起動したら、アドレスバーに「rpf.io/spot-pic」と入力し、ENTER キーを押します。画像を右クリックして「名前を付けて画像を保存」をクリックし、左側のリストから Home フォルダーを選択して「保存」をクリックします。Chromium のアドレスバーをクリックし、「rpf.io/scary-pic」と入力して ENTER キーを押します。同様に、画像を右クリックして「名前を付けて画像を保存」をクリックし、Home フォルダーを選択して「保存」をクリックします。

サウンドファイルを取得するには、アドレスバーをクリックし、「rpf.io/scream」と入力して ENTER キーを押します。プレーヤーにリアルな驚きを与えるこのサウンドファイルは自動的に再生されますが、使用する前に保存する必要があります。小さなオーディオプレーヤーを右クリックして「名前を付けて保存」をクリックし、Home フォルダーを選択して「保存」をクリックします。Chromium ウィンドウを閉じます。

Thonny のツールバーで「New」アイコンをクリックし、新しいプロジェクトを開始します。前回と同様、Python の機能を拡張するライブラリーを使用します。Pygame ライブラリーは、名前が示すようにゲームを念頭に置いて作成されています。以下のように入力します。

```
import pygame
```

他のライブラリーのコードと、Pygame ライブラリーのサブセクションのコードも必要になります。以下のように入力し、これらをインポートします。


```
from pygame.locals import *
from time import sleep
from random import randrange
```

`from` 命令は `import` 命令と異なり、ライブラリー全体ではなく、ライブラリーの一部のみをインポートできます。次に、Pygame のセットアップを行います。これは初期化と呼ばれます。Pygame はプレイヤーのモニターまたはテレビの幅と高さを知る必要があります。これは解像度と呼ばれます。以下のように入力します。

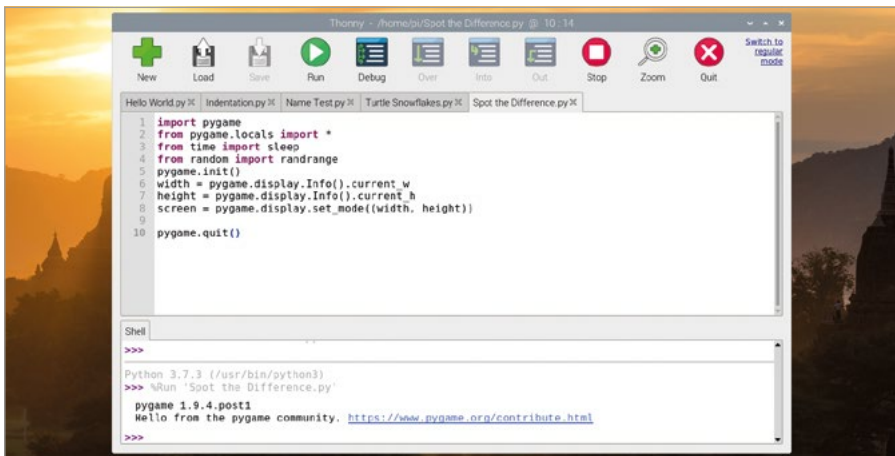
```
pygame.init()
width = pygame.display.Info().current_w
height = pygame.display.Info().current_h
```

Pygame の最後のセットアップ手順では Pygame のウィンドウを作成します。Pygame では「screen」と呼ばれます。以下のように入力します。

```
screen = pygame.display.set_mode((width, height))
```

```
pygame.quit()
```

中央の空白行に注意してください。ここにあなたのプログラムを作成します。その前に、「Run」アイコンをクリックし、プログラムを「Spot the Difference」という名前で保存して、以下の動作を確認します。Pygame がウィンドウを作成し、背景を黒で塗りつぶします。ウィンドウは停止の命令に到達するとすぐに非表示になります。シェル内の短いメッセージ (図 5-14) を除き、プログラムはほとんど完成していません。



▲ 図 5-14: プログラムは動作しますが、大した機能は実行できません

間違い探しの画像を表示するには、`pygame.quit()` の上の空白行に以下のように入力します。

```
difference = pygame.image.load('spot_the_diff.png')
```

画像が画面全体に表示されるようにするため、モニターまたはテレビの解像度に合わせて画像を拡大/縮小する必要があります。以下のように入力します。

```
difference = pygame.transform.scale(difference, (width, height))
```

画像がメモリーに保存されたので、実際に画像を画面に表示するよう Pygame に指示します。このプロセスはブリッティングまたはビットブロック転送と呼ばれます。以下のように入力します。

```
screen.blit(difference, (0, 0))
pygame.display.update()
```

最初の行によって画面の左上隅から画像のコピーが開始されます。2 行目は画面を再描画するよう Pygame に指示します。2 行目を入力しない場合、画像はメモリー内の正しい場所に配置されますが、画面上に表示されません。

「Run」アイコンをクリックすると、画像が一瞬だけ画面上に表示されます (図 5-15)。



▲ 図 5-15:あなたの間違い探し画像

画像の表示時間を長くするには、`pygame.quit()` のすぐ上に以下の行を追加します。

```
sleep(3)
```

もう一度「Run」をクリックすると、画像が画面上に表示される時間が長くなります。サプライズ画像を追加するには、`pygame.display.update()`: 行のすぐ下に以下のように入力します。

```
zombie = pygame.image.load('scary_face.png')
zombie = pygame.transform.scale(zombie, (width, height))
```

遅延を追加し、ゾンビ画像がすぐに表示されないようにします。

```
sleep(3)
```

画像を画面にプリットして更新し、プレーヤーに表示されるようにします。

```
screen.blit(zombie, (0,0))
pygame.display.update()
```

「Run」アイコンをクリックし、何が起こるか確認します。Pygame は間違い探しの画像を読み込みますが、3 秒後に恐ろしいゾンビの画像を表示します (図 5-16)。



▲ 図 5-16:恐ろしい画像が表示されます

遅延が常に 3 秒に設定されていると、次に何が起こるか予測されやすくなるかもしれませんが、`screen.blit(zombie, (0,0))` 行の上にある `sleep(3)` 行を以下のように変更しましょう。

```
sleep(randrange(5, 15))
```

これにより、5 から 15 の乱数が選択され、その時間だけプログラムが遅延するようになります。次に、`sleep` 命令のすぐ上に以下の行を追加して、`scream` サウンドファイルを読み込みましょう。

```
scream = pygame.mixer.Sound('scream.wav')
```

サウンドの再生が開始されるようにするには、**sleep** 命令の下に移動して、以下の行を追加します。これにより、恐ろしい画像がプレーヤーに表示される直前にサウンドの再生が開始されます。

```
scream.play()
```

最後に、**pygame.quit()** のすぐ上に以下の行を入力し、サウンドの再生の停止を Pygame に指示します。

```
scream.stop()
```

「Run」アイコンをクリックし、作成したゲームを確認してください。普通どおりに間違い探しゲームを楽しんでいると、身の毛がよだつような金切り声とともに恐ろしいゾンビの画像が表示されます。友達を驚かせましょう。サウンドの再生が始まる前にゾンビの画像が表示される場合は、**scream.play()** 命令の後 (**screen.blit** 命令の前) にわずかな遅延を追加することで解決できます。

```
sleep(0.4)
```

完成したプログラムは以下のようになります。

```
import pygame
from pygame.locals import *
from time import sleep
from random import randrange
pygame.init()
width = pygame.display.Info().current_w
height = pygame.display.Info().current_h
screen = pygame.display.set_mode((width, height))
difference = pygame.image.load('spot_the_diff.png')
difference = pygame.transform.scale(difference, (width, height))
screen.blit(difference, (0, 0))
pygame.display.update()
zombie = pygame.image.load('scary_face.png')
zombie = pygame.transform.scale(zombie, (width, height))
scream = pygame.mixer.Sound('scream.wav')
sleep(randrange(5, 15))
scream.play()
screen.blit(zombie, (0,0))
pygame.display.update()
sleep(3)
scream.stop()
pygame.quit()
```

あとは友達を招待して間違い探しをするだけです。スピーカーがオンになっていることを忘れず確認してください。



チャレンジ: 外観を変更する



クリスマスなどのイベントに合わせてゲームの画像を変更できますか? 独自の間違い探し画像や恐ろしい画像を (GIMP などのグラフィックエディターを使用して) 描くことはできますか? 間違い探しでユーザーがクリックした箇所が追跡されるようにし、ゲームの信ぴょう性を向上させることができますか?


プロジェクト 3: RPG 迷路

Python のコツをつかんだところで、今度は Pygame を使ってもう少し複雑なゲームを作成してみましょう。古典的なロールプレイングゲームに基づいて完全に機能するテキストベースの迷路ゲームです。テキストアドベンチャーまたはインタラクティブフィクションと呼ばれるこれらのゲームの歴史は、コンピューターがグラフィックスを処理できなかった時代までさかのぼります。当時のゲームファンは、想像上のグラフィックほど鮮やかなグラフィックはないと主張していました。

オンラインプロジェクト

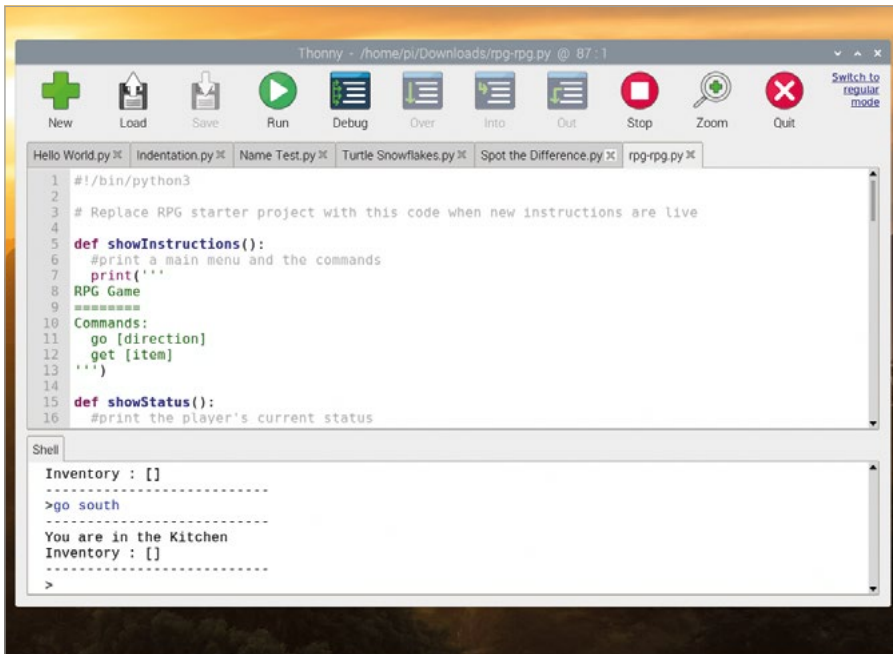
このプロジェクトは、オンライン (rpf.io/python-rpg) でも公開されています。

このプログラムはこの章の他のプログラムよりもかなり複雑になるため、途中で作成されているプログラムを使うことにしましょう。Chromium Web ブラウザーを開き、rpf.io/rpg-code のアドレスに移動します。

Chromium Web ブラウザーはプログラムのコードを Downloads フォルダーに自動的にダウンロードしますが、この種類 (Python プログラム) のファイルがコンピューターに損害を与える可能性がありますと警告します。信頼できる Raspberry Pi Foundation からファイルをダウンロードしたので心配ありません。画面の下部に表示される警告メッセージの「Keep」ボタンをクリックします。Thonny に戻って「Load」アイコン  をクリックします。Downloads フォルダーで **rpg-rpg.py** ファイルを見つけ、「Load」ボタンをクリックします。

「Run」アイコンをクリックし、テキストアドベンチャーの仕組みを理解します。ゲームの出力は Thonny ウィンドウの下部にあるシェル領域に表示されます。出力を読みやすくするため、最大化ボタンをクリックして Thonny ウィンドウを拡大してください。

現時点でゲームの内容は非常にシンプルです。2 つの部屋があり、オブジェクトはありません。プレイヤーは 2 つの部屋の 1 つ目の部屋である大広間からスタートします。キッチンに移動するには、「go south」と入力して **ENTER** キーを押します (図 5-17)。キッチンにいる場合は、「go north」と入力することで大広間に戻ることができます。「go west」や「go east」と入力することもできますが、これらの方向に部屋が存在しないため、エラーメッセージが表示されます。



▲ 図 5-17:現時点で部屋は 2 つしかありません

スクリプト領域でプログラムの 29 行目まで下にスクロールし、**rooms** という変数を見つけます。このタイプの変数は辞書と呼ばれ、ゲームに部屋、部屋の出口、出口がどの部屋につながっているかを指示します。

ゲームをより面白くするため、大広間の東にダイニングルームを作成して、部屋を追加しましょう。スクリプト領域で **rooms** 変数を見つけ、38 行目の } の後にコンマ記号 (,) を追加して拡張し、以下のように入力します (辞書では正確にインデントする必要はありません)。

```

'Dining Room' : {
    'west' : 'Hall'
}

```

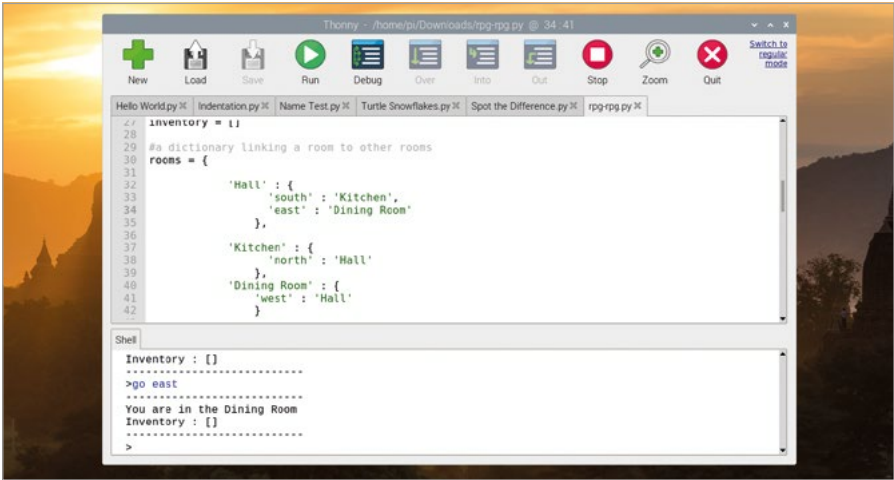
出口は自動的に作成されないため、大広間に新しい出口を作成する必要もあります。33 行目の末尾に移動してコンマを追加し、以下の行を追加します。

```

'east' : 'Dining Room'

```

「Run」アイコンをクリックし、新しい部屋を確認します。ホールにいる場合は、「go east」と入力すると、ダイニングルームに移動します (図 5-18)。ダイニングルームにいる場合は、「go west」と入力すると、大広間に移動します。これで、独自の部屋が作成されました。

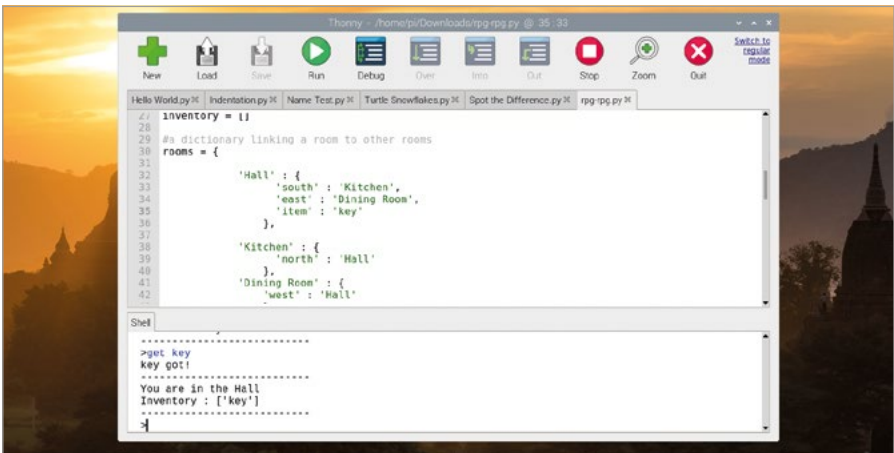


▲ 図 5-18:別の部屋が追加されました


ただし、空の部屋では面白くありません。部屋にアイテムを追加するには、その部屋の辞書を変更する必要があります。「Stop」アイコンをクリックし、プログラムを停止します。スクリプト領域で **Hall** の辞書を見つけ、**'east' : 'Dining Room'** 行の末尾にコンマを追加し、**ENTER** キーを押さずに以下の行を入力します。

'item' : 'key'

もう一度「Run」をクリックします。今回は新しいアイテムである鍵の表示について通知されます。「get key」(図 5-19) と入力すると、鍵を収集してアイテムリスト (インベントリーと呼ばれます) に追加できます。部屋から部屋へ移動するときはインベントリーもいっしょに移動します。



▲ 図 5-19:収集した鍵はインベントリーに追加されます

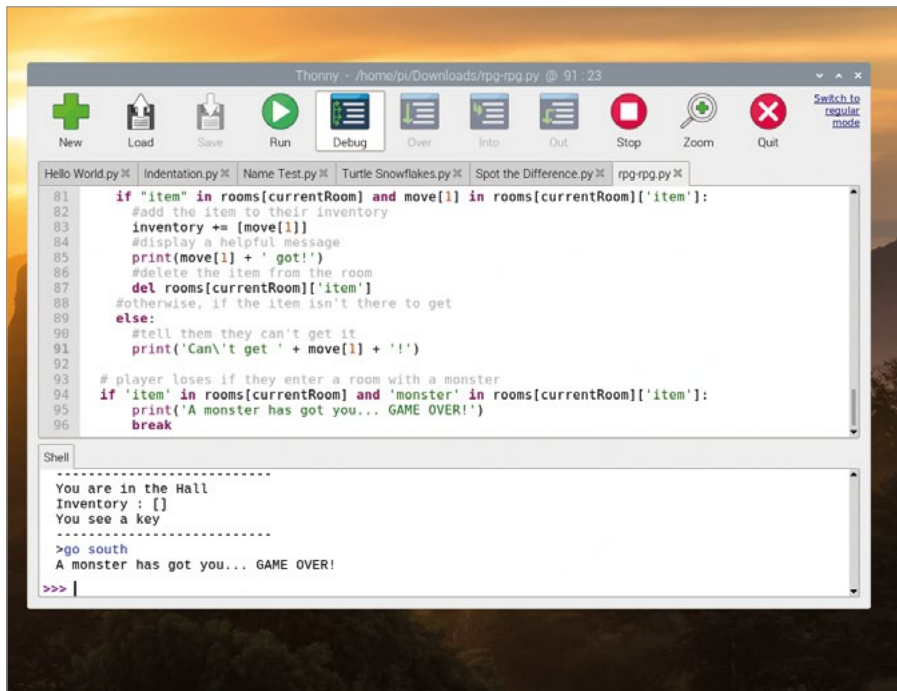
「Stop」アイコン  をクリックし、回避しなければならないモンスターを追加してゲームをより面白くしましょう。**Kitchen** の辞書を見つけ、「鍵」アイテムを追加した場合と同じ方法で「モンスター」アイテムを追加します。上の行の末尾にコンマを追加することを忘れないようにしてください。

```
'item' : 'monster'
```

モンスターがプレイヤーを攻撃できるようにするには、ゲームにロジックを追加する必要があります。スクリプト領域でプログラムの一番下までスクロールし、以下の行を追加します。ハッシュ記号でマークされたコメントも含めてください。コメントは後でプログラムを確認する場合にプログラムを理解するのに役立ちます。また、行は必ずインデントするようにしてください。

```
# player loses if they enter a room with a monster
if 'item' in rooms[currentRoom] and 'monster' in
rooms[currentRoom]['item']:
    print('モンスターに攻撃されました...GAME OVER!')
    break
```

「Run」をクリックし、キッチンに移動してみてください (図 5-20)。モンスターがあなたを攻撃します。



▲ 図 5-20:キッチンにモンスターがいるため、ネズミが出る心配はありません

このアドベンチャーをよりゲームらしくするには、アイテムや部屋を追加する必要があります。また、すべてのアイテムをインベントリーに収めて家を脱出すると、ゲームに「勝利」できるようにしましょう。ダイニングルームを作成した場合と同様に、別の部屋を追加します。今回はガーデンを作成します。ダイニングルームの辞書を使用して出口を追加します。上の行の末尾にコンマを追加することを忘れないようにしてください。

```
'south' : 'Garden'
```

次に、新しい部屋をメインの `rooms` の辞書に追加します。同様に、上の行の `}` の後にコンマを追加することを忘れないようにしてください。

```
'Garden' : {  
    'north' : 'Dining Room'  
}
```

ダイニングルームの辞書に「薬」オブジェクトを追加します。ここでも、上の行にコンマを追加することを忘れないようにしてください。

```
'item' : 'potion'
```

最後に、プログラムの一番下までスクロールして、プレイヤーがすべてのアイテムを所持しているかどうかを確認するロジックを追加します。所持している場合は、ゲームに勝利したことを伝えます。

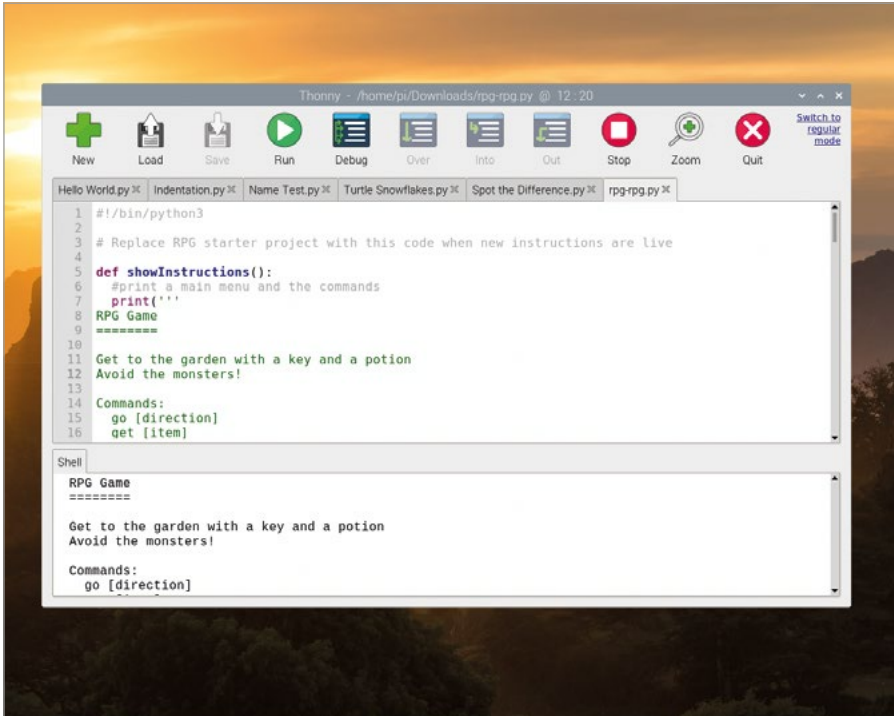
```
# player wins if they get to the garden with a key and a potion  
if currentRoom == 'Garden' and 'key' in inventory and  
'potion' in inventory:  
    print('家から脱出しました...あなたの勝ちです!')  
    break
```

「Run」をクリックします。鍵と薬を収集してガーデンに移動し、ゲームを終了します。キッチンにはモンスターがいるので入らないようにしてください。

最後に、ゲームを完了する方法をプレイヤーに指示することにしましょう。プログラムの一番上にスクロールし、`showInstructions()` 関数が定義されている場所に移動して、以下を追加します。

**鍵と薬を収集し、ガーデンに到達してください。
モンスターを回避してください!**

最後にもう一度ゲームを実行すると、ゲームの初めに新しい手順が表示されます (図 5-21)。これで、インタラクティブなテキストベースの迷路ゲームが作成されました。



```
1 #!/bin/python3
2
3 # Replace RPG starter project with this code when new instructions are live
4
5 def showInstructions():
6     #print a main menu and the commands
7     print('''
8     RPG Game
9     =====
10
11     Get to the garden with a key and a potion
12     Avoid the monsters!
13
14     Commands:
15     go [direction]
16     get [item]
```

```
RPG Game
=====

Get to the garden with a key and a potion
Avoid the monsters!

Commands:
go [direction]
```

▲ 図 5-21: プレイヤーに指示が表示されるようになりました



チャレンジ: ゲームを拡張する



部屋を追加してゲーム時間を長くできますか? モンスターから身を守るアイテムを追加できますか? モンスターを倒す武器をどのように追加しますか? 既存の部屋の階上または階下に部屋を追加し、階段で移動できるようにできますか?

第 6 章

Scratch と Python を使って物理的コンピューティングに挑戦しよう

コーディングは画面上の操作のみを意味するわけではありません。Raspberry Pi の GPIO ピンに接続されている電子部品を制御することもできます。



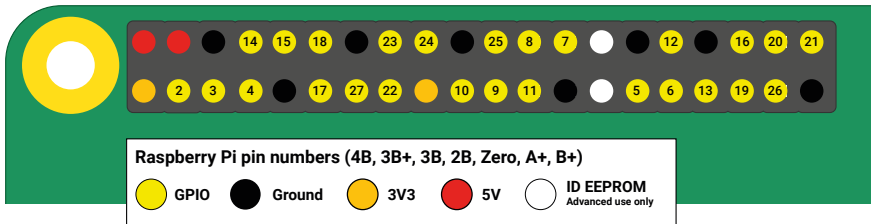
通常、「プログラミング」や「コーディング」と聞くと、人々はたいていソフトウェアのことを思い浮かべます。しかし、コーディングはソフトウェアを制御するだけではなく、ハードウェアを通じて現実の世界も制御できます。これを物理的コンピューティングと呼びます。

名前が示すように、物理的コンピューティングはプログラムを使用して現実世界の物事を制御します。その対象はソフトウェアよりもハードウェアです。物理的コンピューティングは、洗濯機のプログラムを設定するとき、プログラム可能なサーモスタットの温度を変更するとき、また安全に道路を横断するために信号機のボタンを押すときに使用されています。

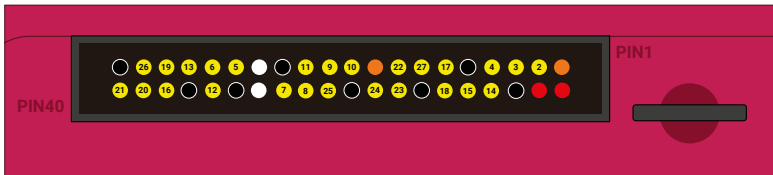
Raspberry Pi は物理的コンピューティングを学習するのに最適なデバイスです。汎用入出力 (GPIO) ヘッダーと呼ばれる主要機能を備えているからです。

GPIO ヘッダーの概要

GPIO (汎用入出力) ヘッダーは Raspberry Pi の回路基板の上端、または Raspberry Pi 400 の背面にあり、2 列の長い金属ピンのような外観をしています。LED やスイッチなどのハードウェアを Raspberry Pi に接続すると、作成したプログラムを使用してハードウェアを制御できます。ピンは入力と出力の両方に使用できます。



Raspberry Pi の GPIO ヘッダーは 40 本のオスピンで構成されています。物理的コンピューティングのプロジェクトに使用できるピンもあれば、電力を供給するピンもあります。Sense HAT などのアドオンハードウェアとの通信用に予約されているピンもあります (第 7 章を参照)。



Raspberry Pi 400 には同じ GPIO ヘッダーが搭載されており、ピンもすべて同じですが、他の Raspberry Pi モデルとは配置が逆になっています。この図は Raspberry Pi 400 の背面で GPIO ヘッダーを見た場合の外観を示しています。Raspberry Pi 400 の GPIO ヘッダーで接続を行うときは、必ず配線を再確認してください。ケースには「Pin 40」や「Pin 1」というラベルが付いていますが、配置に注意する必要があります。

GPIO 拡張機能

Raspberry Pi 400 の GPIO ヘッダーをそのまま使用してもまったく問題ありませんが、エクステンションを使用すると扱いやすくなります。エクステンションを使用すると、Raspberry Pi 400 の外にピンを出すことができます。つまり、いちいち背面を見なくても配線を確認したり調整したりできるようになります。互換性のあるエクステンションは、pimoroni.com の Black HAT Hack3r や adafruit.com の Pi T-Cobbler Plus などがあります。

エクステンションを購入する場合は、常に配線方法を確認するようにしてください。Pi T-Cobbler Plus などの一部の拡張機能は、GPIO ピンのレイアウトが異なります。配線する場合は、必ずにメーカーの指示に従ってください。

ピンの種類にはいくつかのカテゴリーがあり、それぞれ特定の機能があります。

3V3	3.3 ボルトの電力	常時オンの 3.3 V 電源。Raspberry Pi の内部の電圧と同じです
5V	5 ボルトの電力	常時オンの 5 V 電源。Raspberry Pi がマイクロ USB 電源コネクタから電源を取り込む際の電圧と同じです
Ground (GND)	0 ボルトのアース	アース接続。電源に接続された回路を完成させるために使用されます
GPIO XX	汎用入出力ピン番号「XX」	プログラムに使用できる GPIO ピン。2 から 27 までの番号で識別されます
ID EEPROM	予約済みの専用ピン	Hardware Attached on Top (HAT) などのアクセサリで使用するために予約されているピン

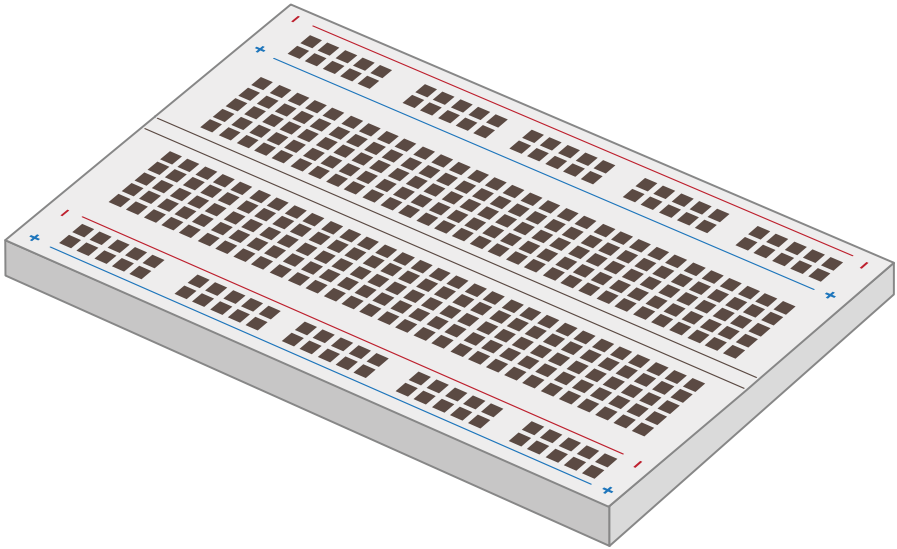
注意

Raspberry Pi の GPIO ヘッダーを使用すると、楽しみながら安全に物理的コンピューティングの実験ができますが、取り扱いには注意が必要です。ハードウェアを接続および切断するときは、ピンを曲げないように注意してください。プロジェクトで明示的に指示されていない限り、2 つのピンを（偶発的か意図的かを問わず）直接接続しないでください。短絡と呼ばれる現象が発生し、ピンによっては Raspberry Pi に致命的な損傷を与える可能性があります。



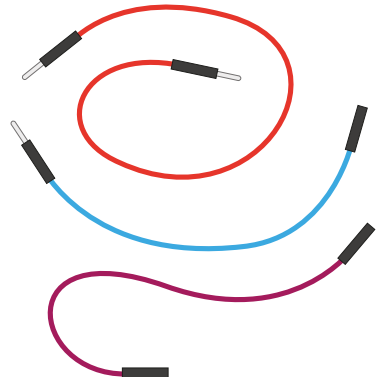
電子部品

GPIO ヘッダーは物理的コンピューティングを開始するために必要なコンポーネントの一部にすぎません。残りのコンポーネントは電子部品、つまり GPIO ヘッダーで制御する各種デバイスです。利用可能なコンポーネントは数千とありますが、ほとんどの GPIO プロジェクトは以下の一般的な部品を使用して作成します。

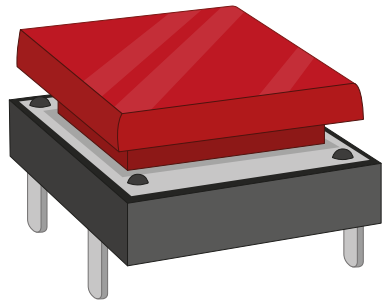


ブレッドボード (ソルダーレスブレッドボードとも呼ばれます) は物理的コンピューティングプロジェクトをととても簡単にしてくれます。ケーブルで接続する必要があるような個別のコンポーネントをブレッドボードに挿入すると、それらは下に隠された金属のレールを介して接続されます。多くのブレッドボードには配電セクションも含まれているため、回路を簡単に構築できます。物理的コンピューティングを開始するためにブレッドボードは必要ありませんが、あると非常に便利です。

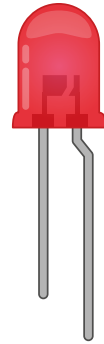
ジャンパー線 (ジャンパーリードとも呼ばれます) はコンポーネントを Raspberry Pi に接続します。ブレッドボードを使用しない場合は、コンポーネントを相互接続します。種類は 3 つあります。ブレッドボードを GPIO ピンに接続するオス - メス (M2F) ジャンパー線、ブレッドボードを使用しない場合に個々のコンポーネントを相互接続するメス - メス (F2F) ジャンパー線、ブレッドボードの一部を別の部分に接続するオス - オス (M2M) ジャンパー線です。プロジェクトによっては 3 種類すべてのジャンパー線が必要になります。ブレッドボードを使用する場合は通常、M2F ジャンパー線と M2M ジャンパー線のみ必要になります。



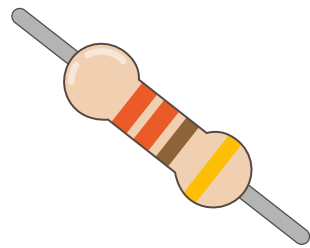
押しボタンスイッチ (モーメンタリスイッチとも呼ばれます) はゲームコンソールの制御に使用するスイッチです。一般的には 2 足か 4 足の押しボタンスイッチがあり、どちらのタイプも Raspberry Pi で動作します。押しボタンは入力デバイスです。押しボタンが押された場合にタスクを実行するようにプログラムで指示することができます。一般的なスイッチにはラッチスイッチもあります。押しボタンは押している間だけアクティブになりますが、ラッチスイッチは照明のスイッチと同様、一度切り替えるとアクティブになり、再度切り替えるまでアクティブなままになります。



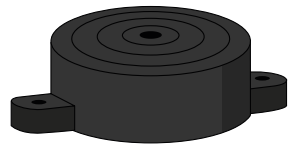
発光ダイオード (LED) は出力デバイスです。プログラムから直接制御します。LED はオンになると点灯します。洗濯機の電源が入ればなしになっていることを知らせる小さな LED から、部屋の照明といった大きな LED に至るまで、家のあらゆる場所で使用されています。LED はさまざまな形、色、サイズが提供されていますが、すべてが Raspberry Pi に適しているわけではありません。5V または 12V の電源用に設計されている LED は使用しないようにしてください。



抵抗器は電流の流れを制御するコンポーネントで、さまざまな値の抵抗器が提供されています。値はオーム (Ω) と呼ばれる単位を使用して表記されます。オームの値が高いほど、より多くの抵抗が供給されます。Raspberry Pi の物理的コンピューティングプロジェクトでは主に、LED が過剰な電流を引き込んで LED 自体や Raspberry Pi を損傷させるのを防ぐために使用されます。これには定格がおよそ 330Ω の抵抗器が必要になりますが、多くの電子部品店では利便性に配慮して、一般的に利用されているさまざまな値の抵抗器がセットになった便利なパックを販売しています。



圧電ブザー (通常はブザーと呼ばれます) も出力デバイスです。LED は光を生成しますが、ブザーは音を生成します。ブザーのプラスチックケースの中には一対の金属板が含まれています。アクティブになると、これらの金属板は互いに振動してブーという音を出力します。ブザーにはアクティブブザーとパッシブブザーの 2 種類があります。アクティブブザーの方が使いやすいので、アクティブブザーを入手するようにしてください。



その他の一般的な電気部品には、Raspberry Pi に接続する前に特別な制御ボードを必要とするモーター、動きを検出する赤外線センサー、天気の前測に使用する温度センサーや湿度センサー、光を検出することで逆 LED のように動作する光依存抵抗器 (LDR) (入力デバイス) などがあります。

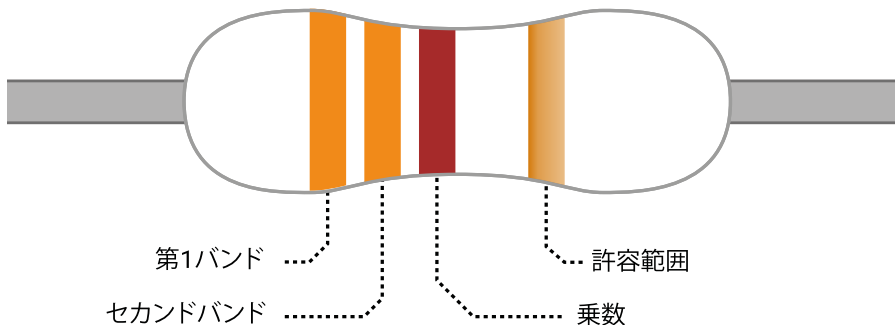
世界中の販売者は Raspberry Pi を使用した物理的コンピューティング向けのコンポーネントを提供しており、個別の部品として入手することも、作業を開始するために必要なものすべてが含まれているようなキットとして入手することもできます。販売者を見つけるには、rpf.io/products にアクセスして、「Raspberry Pi 4」をクリックしてください。住んでいる国や地域の Raspberry Pi パートナー (認定リセラー) のオンラインストア一覧が表示されます。

この章のプロジェクトを完了するには、少なくとも以下が必要になります。

- LED × 3: 赤、緑、黄色、または琥珀色
- 押しボタンスイッチ × 2
- アクティブブザー × 1
- オス - メス (M2F) ジャンパー線とメス - メス (F2F) ジャンパー線
- (オプション) ブレッドボードとオス - オス (M2M) ジャンパー線

抵抗器のカラーコードを読む

抵抗器の値はさまざまです。値がゼロの抵抗器は単なるケーブルのように機能し、足のような大きさの抵抗器は高い抵抗値を生成します。抵抗器の値が数字で印刷されていることはほとんどなく、代わりに抵抗器の本体の周りに色付きの線またはバンドとして印刷されている特別なコードを使用します。



	1st/2nd Band	Multiplier	Tolerance
黒	0	$\times 10^0$	-
褐色	1	$\times 10^1$	$\pm 1\%$
赤	2	$\times 10^2$	$\pm 2\%$
オレンジ	3	$\times 10^3$	-
黄	4	$\times 10^4$	-
緑	5	$\times 10^5$	$\pm 0.5\%$
青い	6	$\times 10^6$	$\pm 0.25\%$
バイオレット	7	$\times 10^7$	$\pm 0.1\%$
灰白	8	$\times 10^8$	$\pm 0.05\%$
白い	9	$\times 10^9$	-
ゴールド	-	$\times 10^{-1}$	$\pm 5\%$
銀	-	$\times 10^{-2}$	$\pm 10\%$
なし	-	-	$\pm 20\%$

抵抗器の値を読むには、バンドのグループが左側に、単独のバンドが右側に来るように抵抗器を配置します。最初のバンドの色を「第1・第2バンド」列で調べて、1桁目と2桁目の値を特定します。この例では橙色のバンドが2つあり、どちらも「3」を意味するため、合計は「33」になります。抵抗器にバンドのグループが3つではなく4つある場合は、3番目のバンドの値もメモします（バンドのグループが5/6つある抵抗器については rpf.io/5-6band を参照してください）。

グループ内の最後（3番目または4番目）のバンドの色を「乗数」列で調べます。これにより、現在の値に乗算する値を特定し、抵抗器の実際の値を計算できます。この例では「 $\times 10^1$ 」を意味する茶色のバンドが使用されています。難しく見えますが、これは単なる科学的記数法です。「 $\times 10^1$ 」は「値の

最後に 0 を 1 つ追加する」ことを意味します。青の「 $\times 106$ 」は「値の最後に 0 を 6 つ追加する」ことを意味します。

橙色のバンドから特定した 33 に、茶色のバンドで特定した 0 を追加すると、結果は 330 になります。これが抵抗器の値 (オーム単位) になります。右端の最後のバンドは抵抗器の許容差です。これは単純に定格値にどれだけ近いかを表します。安価な抵抗器には銀色のバンドが含まれているものや、最後のバンドが存在しないものがあります。銀色のバンドは値が定格から 10 パーセント上下する可能性があることを示し、最後のバンドが存在しない抵抗器は値が定格から 20 パーセント上下する可能性があることを示します。最も高価な抵抗器には灰色のバンドが含まれており、これは値が定格の 0.05 パーセント以内に収まることを示しています。趣味の一環としてプロジェクトを実行する場合、精度はそれほど重要になりません。通常は許容誤差が大きい抵抗器でも問題なく使用できます。

抵抗器の値が 1000 オーム (1000 Ω) を超える場合は通常、キロオーム (k Ω) を使用して表記されます。100 万オームを超える場合は、メガオーム (M Ω) を使用して表記されます。2200 Ω の抵抗器は 2.2 k Ω と表記されます。2200000 Ω の抵抗は 2.2 M Ω と表記されます。

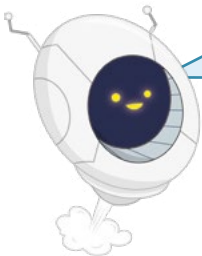


回答してみよう

100 Ω の抵抗器にはどのような色のバンドがありますか? 2.2 M Ω の抵抗器にはどのような色のバンドがありますか? 最も安価な抵抗器を見つけたい場合は、許容誤差を示すバンドの色が何色のものを探せばよいですか?

はじめての物理的コンピューティングプログラム「ハロー、LED!」

画面に「こんにちは!」と出力することがプログラミング言語の学習の第一歩であるのと同様に、LED を点灯させることが物理的コンピューティングの学習を開始する一般的な方法です。このプロジェクトでは、LED、330 オーム (330 Ω) の抵抗器 (または 330 Ω に近い抵抗器)、メス - メス (F2F) ジャンパー線が必要になります。

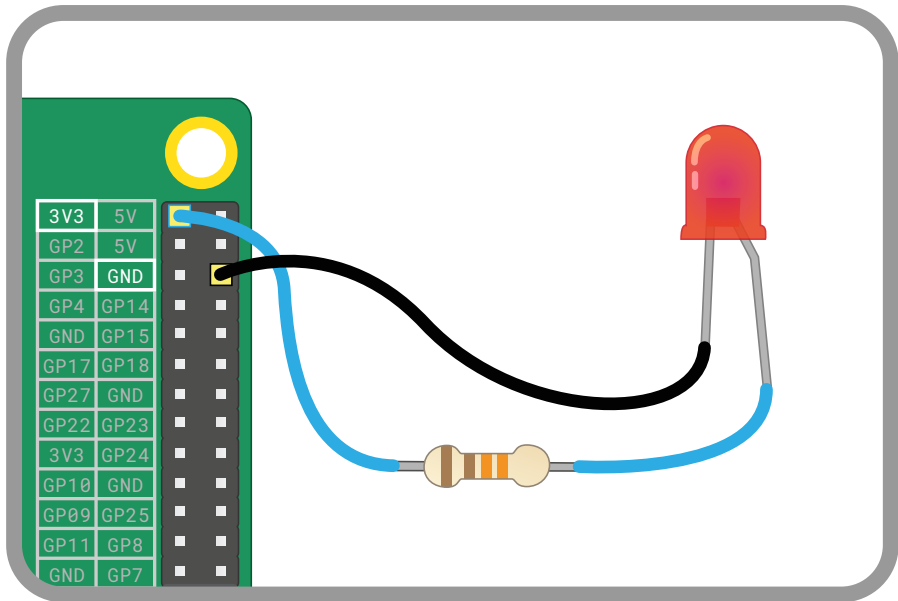


抵抗器は大事

抵抗器はこの回路に不可欠なコンポーネントです。LED が引き込むことができる電流量を制限することにより、Raspberry Pi と LED を保護します。これがないと、LED は過剰な電流を引き込んで LED 自体 (または Raspberry Pi) を焼き尽くす可能性があります。このような目的で使用する抵抗器を電流制限抵抗器と呼びます。必要となる抵抗器の正確な値は使用する LED によって異なりますが、330 Ω の抵抗器はほとんどの一般的な LED で機能します。値が高いほど LED は暗くなり、値が小さいほど LED は明るくなります。

LED に適切な値の抵抗器が内蔵されていることがわかっている場合を除き、電流制限抵抗器なしで LED を Raspberry Pi に接続しないようにしてください。

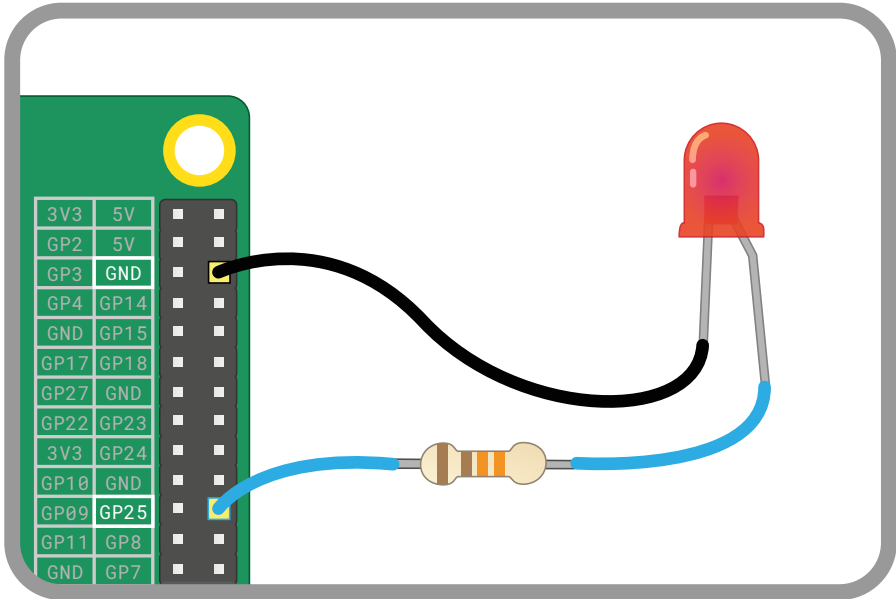
まず LED が機能するか確認します。Raspberry Pi の向きを変え、GPIO ヘッダーが右側縦方向に 2 つ並ぶようにします。330 Ω の抵抗器の一方の終端を最初の 3.3 V ピン (図 6-1 で「3V3」のラベルが付いているピン) に F2F ジャンパー線を使用して接続し、もう一方の終端を LED の長い方の足 (プラスまたは陽極) に、もう 1 つの F2F ジャンパー線を使用して接続します。LED の短い方の足 (マイナスまたは陰極) を最初のアースピン (図 6-1 で「GND」とラベル付けされているピン) に、最後の F2F ジャンパー線を使用して接続します。



▲ 図 6-1: LED を上記のピンに配線する - 抵抗器を忘れないようにしてください

Raspberry Pi がオンになっていれば LED は点灯します。点灯しない場合は、回路を再確認します。抵抗器の値が高すぎないこと、すべてのケーブルが適切に接続されていること、図と同じ適切な GPIO ピンを使用していることを確認してください。LED の足も確認します。LED の長い方の足は回路のプラス側に接続し、短い方の足はマイナス側に接続する必要があります。逆に接続すると、LED は機能しません。

LED が機能したら、プログラミングを開始します。3.3 V ピン (図 6-2 で「3V3」とラベル付けされているピン) からジャンパー線を外し、これを GPIO 25 ピン (図 6-2 で「GP25」とラベル付けされているピン) に接続します。LED がオフになりますが、これは正常な動作です。



▲ 図 6-2:「3V3」からケーブルを外して GPIO 25 ピンに接続する


これで、Scratch または Python で LED をオン/オフさせるプログラムを作成するための準備が整いました。

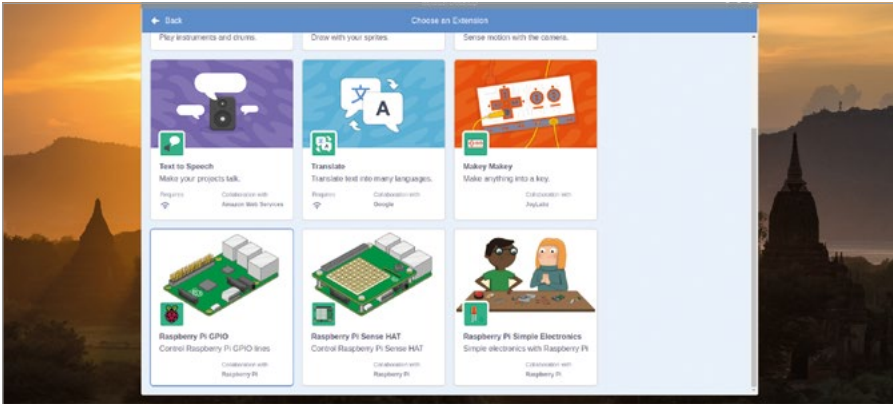


コーディング知識

この章のプロジェクトを実行するには、Scratch 3 と Thonny Python の統合開発環境 (IDE) を使用することに慣れている必要があります。まだ使用したことがない場合は、最初に「第 4 章 Scratch 3 を使ってプログラミングしてみよう」と、「第 5 章: Python を使ってプログラミングしてみよう」で説明されているプロジェクトを試してみてください。

Scratch の LED コントロール

Scratch 3 を起動し、「Add Extension」アイコン  をクリックします。下にスクロールし、「Raspberry Pi GPIO」拡張機能 (図 6-3) を探してクリックします。これにより、Scratch 3 から Raspberry Pi の GPIO ヘッダーを制御するために必要な各種ブロックが読み込まれます。新しいブロックがブロックパレットに表示されます。これらのブロックが必要な場合は、「Raspberry Pi GPIO」カテゴリーで利用できます。



▲ 図 6-3:「Raspberry Pi GPIO」拡張機能を Scratch 3 に追加する

「**が押されたとき**」イベントブロックをコード領域にドラッグした後、このブロックの下に「**set gpio to output high**」ブロックを配置します。使用するピンの番号を選択する必要があるため、小さい矢印をクリックしてドロップダウンを開き、「25」をクリックして GPIO 25 ピンを制御するよう Scratch に指示します。



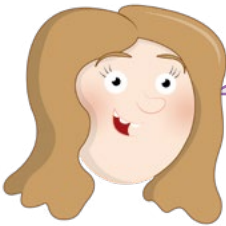
緑のフラグアイコンをクリックして、プログラムを実行してみましょう。LED が点灯すると、最初の物理的コンピューティングプロジェクトのプログラミングに成功したことになります。プログラムを停止するには、赤い八角形アイコンをクリックします。LED が点灯されたままになることに注目してください。これはプログラムで LED をオンにすることしか Raspberry Pi に指示していないためです。これが「**set gpio 25 to output high**」ブロックの「output high」の意味するところです。もう一度オフにするには、ブロックの最後にある下矢印をクリックし、リストから「low」を選択します。



緑のフラグアイコンをもう一度クリックすると、プログラムによって LED がオフになります。今回は「**ずっと**」制御ブロックといくつかの「**1秒待つ**」ブロックを追加して、LED を 1 秒ごとに点滅させるプログラムを作成してみましょう。



緑のフラグアイコンをクリックし、LEDを確認します。LEDが1秒間点灯した後、1秒間消灯し、再度1秒間点灯します。この動作は赤い八角形アイコンをクリックして停止させるまで繰り返されます。LEDがオン状態またはオフ状態のときに八角形アイコンをクリックするとどうなるか試してみてください。



チャレンジ: 変更してみよう

LEDの点灯時間を長くするにはプログラムをどのように変更すればよいですか? 消灯時間を長くするにはどうすればよいですか? LEDスイッチのオンとオフを確認できる最小遅延はどれくらいですか?

Python の LED コントロール

raspberrry メニューの「プログラミング」セクションから Thonny を起動します。「New」ボタンをクリックして新しいプロジェクトを開始し、「Save」をクリックして「Hello LED」という名前を付けて保存します。Python から GPIO ピンを使用するには、GPIO Zero という名前のライブラリが必要になります。このプロジェクトではライブラリの一部のみを使用して LED を制御します。Python のシェル領域に以下のように入力し、ライブラリーのこのセクションのみをインポートします。

```
from gpiozero import LED
```

次に、LED がどの GPIO ピンに接続されているかを GPIO Zero に指示する必要があります。以下のように入力します。

```
led = LED(25)
```

これらの 2 つの行を組み合わせることで、Python は Raspberry Pi の GPIO ピンに接続されている LED を制御できるようになります。また、どのピンを制御するかを Python に指示できます (回路に複数の LED が存在する場合は制御対象となる複数のピンを指示できます)。LED を実際に制御するには、以下のように入力します。

```
led.on()
```

LED を再びオフに切り替えるには、以下のように入力します。

```
led.off()
```

これで、Raspberry Pi の GPIO ピンを Python で制御できるようになりました。これらの 2 つの命令をもう一度入力してみてください。LED がすでにオフになっている場合、**led.off()** は何もしません。LED がすでにオンになっているときに「**led.on()**」と入力した場合も同様です。

プログラムらしくするには、スクリプト領域に以下のように入力します。

```
from gpiozero import LED
from time import sleep
```

```
led = LED(25)
```

```
while True:
    led.on()
    sleep(1)
    led.off()
    sleep(1)
```

このプログラムは **gpiozero** (GPIO Zero) ライブラリーから LED 関数を、**time** ライブラリーから **sleep** 関数をインポートし、無限ループを構築して、LED を 1 秒間オンにした後、1 秒間オフにする動作を繰り返します。「Run」ボタンをクリックして動作を確認します。LED が点滅を開始します。Scratch プログラムの場合と同様、LED がオン状態のときにオフ状態のときに「Stop」ボタンをクリックした場合の動作をメモします。



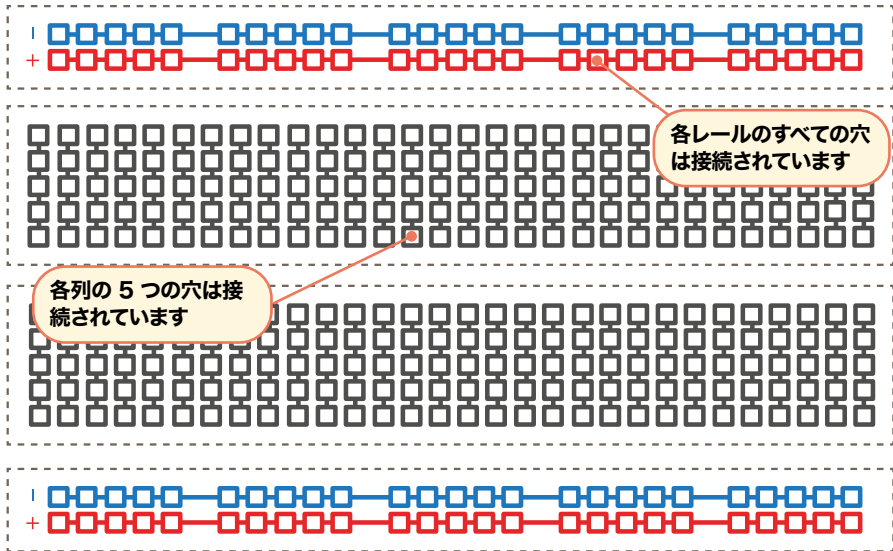
チャレンジ: 点灯時間を長くしてみよう



LED の点灯時間を長くするにはプログラムをどのように変更すればよいですか? 消灯時間を長くするにはどうすればよいですか? LED スwitch のオンとオフを確認できる最小遅延はどれくらいですか?

ブレッドボードを使用する

次のプロジェクトは、ブレッドボードにコンポーネントを配置して電気接続を行うと、はるかに簡単に完了できます。



ブレッドボードは穴で覆われています。穴はコンポーネントに合わせて 2.54 mm 間隔で配置されています。これらの穴の下には、これまで使用してきたジャンパー線のような機能を持つ金属片があります。これらはボード全体にわたって列を組んで並んでいます。ほとんどのボードには中央に割れ目があり、2 つに分割できるようになっています。多くのブレッドボードには上部に文字が記されており、両側には数字が記されています。これらによって特定の穴を見つけることができます。A1 は左上隅、B1 はその右側の穴、B2 はその 1 つ下の穴になります。A1 は隠れている金属片によって B1 に接続されていますが、ジャンパー線を追加しない限り、1 つの穴を 2 つの穴に接続しないようにしてください。

大きなブレッドボードにも両側に穴の列があります。通常は赤と黒、または赤と青の線が記載されています。これらはパワーレールです。配線を容易にするように設計されています。Raspberry Pi のアースピンからいずれかのパワーレール (通常は青または黒の線とマイナス記号が記載されています) に 1 本のケーブルを接続し、ブレッドボード上の多数のコンポーネントに対して共通のアースを提供できます。回路に 3.3 V または 5 V の電力が必要な場合も同様の操作を行うことができます。

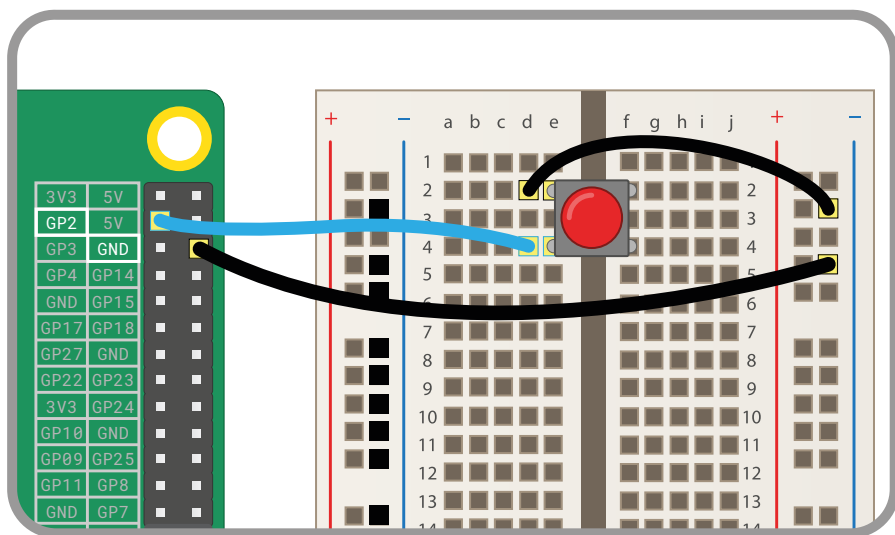
ブレッドボードに電子部品を追加するのは簡単です。リード (突き出ている金属部品) を穴に合わせて、部品が所定の位置に収まるまで静かに押します。必要な接続の数がブレッドボードで対応可能な接続の数を超える場合は、オス - オス (M2M) ジャンパー線を使用できます。ブレッドボードから Raspberry Pi への接続には、オス - メス (M2F) ジャンパー線を使用します。

ブレッドボードの 1 つの穴に複数の部品リードやジャンパー線を差し込まないようにしてください。注意: 中央の割れ目を除き、穴は列単位で接続されます。このため、A1 の部品リードは B1、C1、D1、E1 に追加するものすべてと電氣的に接続されます。

次のステップ: ボタンの読み取り

LED などの出力は別ですが、「GPIO」の「入出力」部分は、ピンを入力として使用することもできます。このプロジェクトでは、ブレッドボード、オス - オス (M2M) ジャンパー線、オス - メス (M2F) ジャンパー線、押しボタンスイッチが必要になります。ブレッドボードがない場合はメス - メス (F2F) ジャンパー線を使用できますが、誤って回路を壊さないようにボタンを押すのが非常に困難になります。

まずブレッドボードに押しボタンを追加します。押しボタンの足が 2 つしかない場合は、これらの足がブレッドボード上で異なる行番号の穴に挿入されていることを確認してください。足が 4 つある場合は押しボタンの向きを調整し、足がある方の側面がブレッドボードの行と平行になるようにし、足のない方の平らな側面が上部と下部に配置されるようにします。ブレッドボードのアース線を Raspberry Pi のアースピン (図 6-4 で「GND」と記載されているピン) に M2F ジャンパー線を使用して接続します。次に、押しボタンの一方の足を M2M ジャンパー線を使用してアース線に接続します。最後に、もう一方の足 (4 足スイッチを使用している場合はこの前に接続した足と同じ側にある足) を Raspberry Pi の GPIO 2 ピン (図 6-4 で「GP2」と記載されているピン) に M2F ジャンパー線を使用して接続します。



▲ 図 6-4:押しボタンを GPIO ピンに配線する

Scratch でボタンを読む

新しい Scratch プログラムを開始しましょう。「**緑の旗が押されたとき**」ブロックをコード領域にドラッグします。「**set gpio to input pulled high**」ブロックを接続し、ドロップダウンから「2」を選択して、押しボタンに使用した GPIO ピンと一致させます。



この状態で緑のフラグアイコンをクリックしても何も起こりません。ピンを入力として使用するよう Scratch に指示しましたが、その入力をどう処理するかは指示していないためです。「ずっと」ブロックを一番下にドラッグし、その内側に「もし ... なら ... でなければ」ブロックをドラッグします。「gpio is high?」ブロックを見つけて、「もし ... なら」部分のひし形のスペースにドラッグし、ドロップダウンで「2」を選択して、どの GPIO ピンを制御するかを指示します。「こんにちは! と 2 秒言う」ブロックを「でなければ」部分にドラッグし、「ボタンが押されました!」と表示されるように編集します。ブロックの「もし ... なら」部分は空白のままにしておきます。



これによって多くのことが実行されます。テストしてみましょう。緑のフラグアイコンをクリックし、ブレッドボード上のボタンを押します。ボタンが押されたことをスプライトが通知すると、GPIO ピンからの入力が正常に読み取られたこととなります。

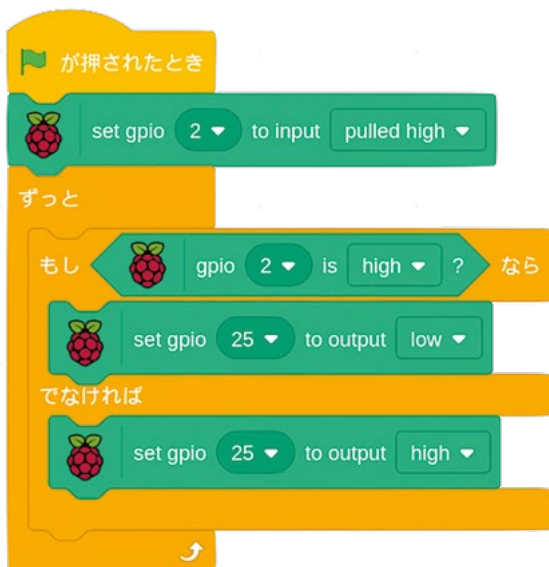
「もし gpio 2 is high? なら」部分が空白であることに気づかれているかもしれませんが、ボタンが実際に押されたときに実行されるコードは、押されるまでの間、「でなければ」部分にあります。わかりずらいかもしれません。ボタンを押すと実際に電圧は「high」になるのでしょうか。実はその逆です。Raspberry Pi の GPIO ピンは、入力として設定されている場合、通常は「high」またはオンになっています。ボタンを押すと「low」に引き下げられます。

回路をもう一度見てください。回路のプラスを提供する GPIO 2 ピンとアースピンにボタンがどのように接続されているか確認してください。ボタンを押すと、GPIO ピンの電圧がアースピンを介して「low」に引き下げられ、Scratch プログラムは「もし gpio 2 is high? なら」ブロック内のコードの実行を停止します。代わりに、「でなければ」部分のコードを実行します。

わかりずらい場合はこれだけ覚えておいてください。Raspberry Pi の GPIO ピンのボタンを押すと、ピンの電圧は高くなるのではなく、低くなります。

プログラムをさらに拡張するには、LED と抵抗器を回路に追加し直します。抵抗器を GPIO 25 ピンと LED の長い方の足に接続し、LED の短い方の足をブレッドボードのアース線に接続することに注意してください。

「ボタンが押されました! と 2 秒言う」ブロックをコード領域からブロックパレットにドラッグして削除し、「set gpio 25 to output high」ブロックに置き換えます。ドロップダウン矢印を使用して GPIO 番号を変更する必要があることに注意してください。「set gpio 25 to output low」ブロック (値を変更することに注意してください) を、「もし gpio 2 is high? なら」の現在空白になっている部分に追加します。



緑のフラグアイコンをクリックし、ボタンを押します。ボタンを押している間、LED が点灯します。ボタンを離すと、再び消灯します。これで、別の GPIO ピンからの入力に基づいて 1 つの GPIO ピンを制御することに成功しました。



チャレンジ: 点灯したままにしてみよう



ボタンを離しても LED が数秒間点灯し続けるようにするにはプログラムをどのように変更すればよいですか? ボタンを押していないときに LED をオンにし、押しているときにオフにするには、何を変更する必要がありますか?

Python でボタンを読む

Thonny の「New」ボタンをクリックして新しいプロジェクトを開始します。「Save」ボタンをクリックし、「**Button Input**」という名前を付けて保存します。ボタンの入力として GPIO ピンを使用することは、LED の出力としてピンを使用することと非常に似ていますが、GPIO Zero ライブラリーから別の関数をインポートする必要があります。スクリプト領域で以下のように入力します。

```
from gpiozero import Button
button = Button(2)
```

GPIO Zero が提供する `wait_for_press` 関数を使用すると、ボタンを押したときにコードが実行されるようになります。以下のように入力します。

```
button.wait_for_press()
print("ボタンが押されました!")
```

「Run」ボタンをクリックし、押しボタンスイッチを押します。Thonny ウィンドウの下部にある Python のシェルにメッセージが出力されると、GPIO ピンからの入力が正常に読み取られたこととなります。プログラムを再試行する場合は、「Run」ボタンをもう一度クリックする必要があります。プログラムにはループが含まれていないため、メッセージがシェルに出力されるとプログラムがすぐに終了するためです。

プログラムをさらに拡張するには、LED と抵抗器を回路に（まだ追加していない場合は）追加し直します。抵抗器を GPIO 25 ピンと LED の長い方の足に接続し、LED の短い方の足をブレッドボードのアース線に接続することに注意してください。

LED を制御してボタンを読むには、GPIO Zero ライブラリーから `Button` 関数と `LED` 関数の両方をインポートする必要があります。`time` ライブラリーの `sleep` 関数も必要になります。プログラムの先頭に戻り、以下の 2 行をプログラムの最初の 2 行になるように入力します。

```
from gpiozero import LED
from time import sleep
```

`button = Button(2)` 行の下で、以下のように入力します。

```
led = LED(25)
```

`print("ボタンが押されました!")` 行を削除し、以下に置き換えます。

```
led.on()
sleep(3)
led.off()
```


完成したプログラムは以下のようになります。

```
from gpiozero import LED
from time import sleep
from gpiozero import Button

button = Button(2)
led = LED(25)
button.wait_for_press()
led.on()
sleep(3)
led.off()
```

「Run」ボタンをクリックし、押しボタンスイッチを押します。LED が 3 秒間点灯した後、再びオフになって、プログラムが終了します。これで、Python でボタン入力を使用して、LED を制御することに成功しました。



チャレンジ: ループを追加してみよう



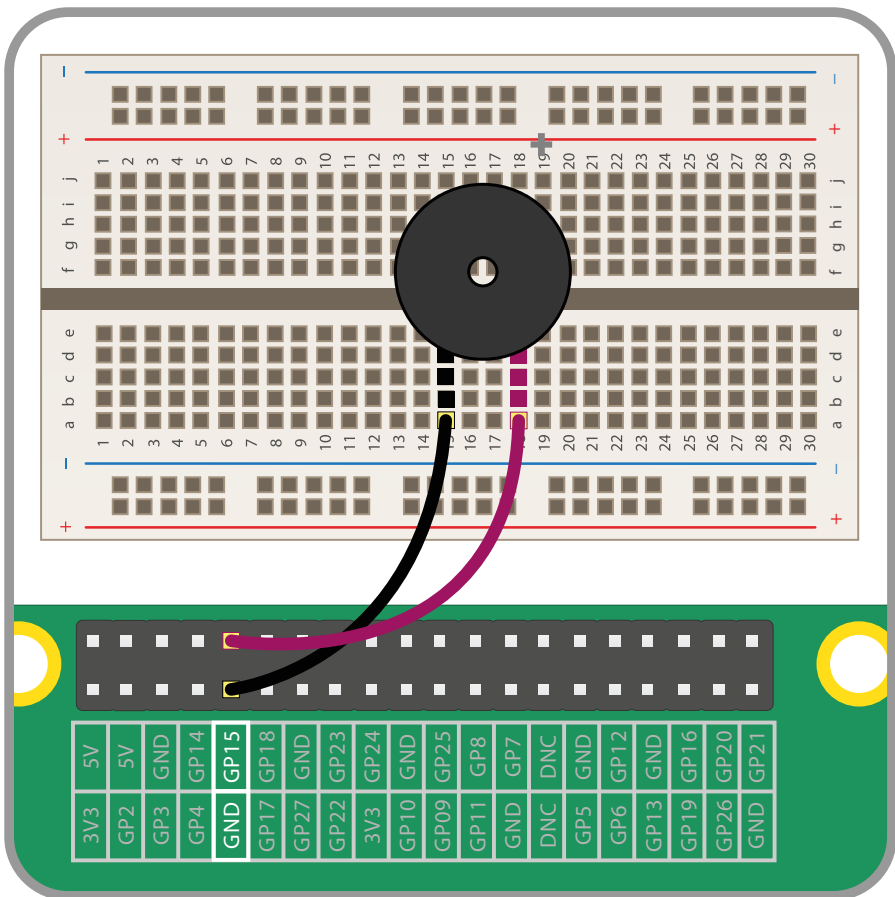
ボタンを 1 回押した後、プログラムを終了しないで、プログラムが繰り返されるようにするには、どのようにループを追加すればよいですか? ボタンを押していないときに LED をオンにし、押しているときにオフにするには、何を変更する必要がありますか?

音を出す: ブザーの制御

LED は優れた出力デバイスですが、目で確認する必要があります。ブザーを使用すると、部屋のどこにいても音で気づくことができます。このプロジェクトでは、ブレッドボード、オス - メス (M2F) ジャンパー線、アクティブブザーが必要になります。ブレッドボードがない場合は、代わりにメス - メス (F2F) ジャンパー線を使用してブザーを接続できます。

アクティブブザーは、回路とプログラミングにおいて LED とまったく同様に処理できます。LED 用に作成した回路を再利用しますが、LED はアクティブブザーと置き換え、抵抗器は外しておきます。ブザーが機能するには、より多くの電流が必要になるからです。ブレッドボードと M2F ジャンパー線を使用して、ブザーの一方の足を GPIO 15 ピン (図 6-5 で「GP15」とラベル付けされているピン) に接続し、もう一方の足をアースピン (図で「GND」とラベル付けされているピン) に接続します。

ブザーに足が 3 本ある場合は、マイナス記号 (-) が記載されている足をアースピンに接続し、「S」または「SIGNAL」と記載されている足を GPIO 15 に接続します。残りの足 (通常は真ん中の足) は 3.3 V ピン (「3V3」とラベル付けされているピン) に接続します。



▲ 図 6-5: ブザーを GPIO ピンに接続する

Scratch でブザーを制御する

LED を点滅させるプログラムと同じものを再作成します。ボタンを読み取るプロジェクトを作成する前に、LED を点滅させるプログラムを保存していた場合は、そのプログラムを読み込みます。*

「set gpio to output high」ブロックのドロップダウンを使用して「15」を選択し、Scratch が正しい GPIO ピンを制御するようにします。



緑のフラグアイコンをクリックすると、ブザーが鳴り始めます。1 秒間オンになった後、1 秒間オフになります。ブザーのクリック音しか聞こえない場合は、アクティブブザーではなくパッシブブザーが使用されています。アクティブブザーは急速に変化する信号（振動と呼ばれます）を生成してそれ自体で金属板を振動させますが、パッシブブザーには振動信号が必要になります。Scratch を使用してオンにすると、金属板は 1 回転して停止します。プログラムでピンをオンまたはオフに切り替えるまで、「カチッ」という音を生成し続けます。

赤い八角形アイコンをクリックするとブザーは停止しますが、音が出ていない状態のときにクリックするようにしてください。そうしないと、プログラムを再度実行するまでブザーが鳴り続けることになります。



チャレンジ: ブザー音を変更してみよう

ブザー音を短くするにはプログラムをどのように変更すればよいですか?ブザーをボタンで制御できるように回路を構築できますか?

Python でブザーを制御する

GPIO Zero ライブラリーを介してアクティブブザーを制御することは、オンとオフの状態があるという点で、LED を制御することとほとんど同じです。ただし、**buzzer** という別の関数が必要になります。Thonny で新しいプロジェクトに「**Buzzer**」という名前を付けて保存し、以下のように入力します。

```
from gpiozero import Buzzer
from time import sleep
```

LED の場合と同様、ブザーがどのピンに接続されているかを GPIO Zero に指示して、ブザーが制御されるようにする必要があります。以下のように入力します。

```
buzzer = Buzzer(15)
```

これ以降は、LED を制御するために作成したプログラムとほとんど同じです。唯一の違いは (GPIO ピンの番号が異なることを除き) **led** の代わりに **buzzer** を使用することです。以下のように入力します。

```
while True:
    buzzer.on()
    sleep(1)
    buzzer.off()
    sleep(1)
```

「Run」ボタンをクリックすると、ブザーが鳴り始めます。1 秒間オンになった後、1 秒間オフになります。アクティブブザーではなくパッシブブザーを使用している場合は、ブザーが連続的に鳴る代わりに、1 秒ごとに短いクリック音のみが聞こえます。パッシブブザーには急速に変化する信号を生成し、ブザー内の金属板を振動させるオシレーターが存在しないからです。

「Run」ボタンをクリックするとプログラムは終了しますが、ブザーの音が出ていない状態のときにクリックするようにしてください。そうしないと、プログラムを再度実行するまでブザーが鳴り続けることとなります。



チャレンジ: ブザー音を改良してみよう

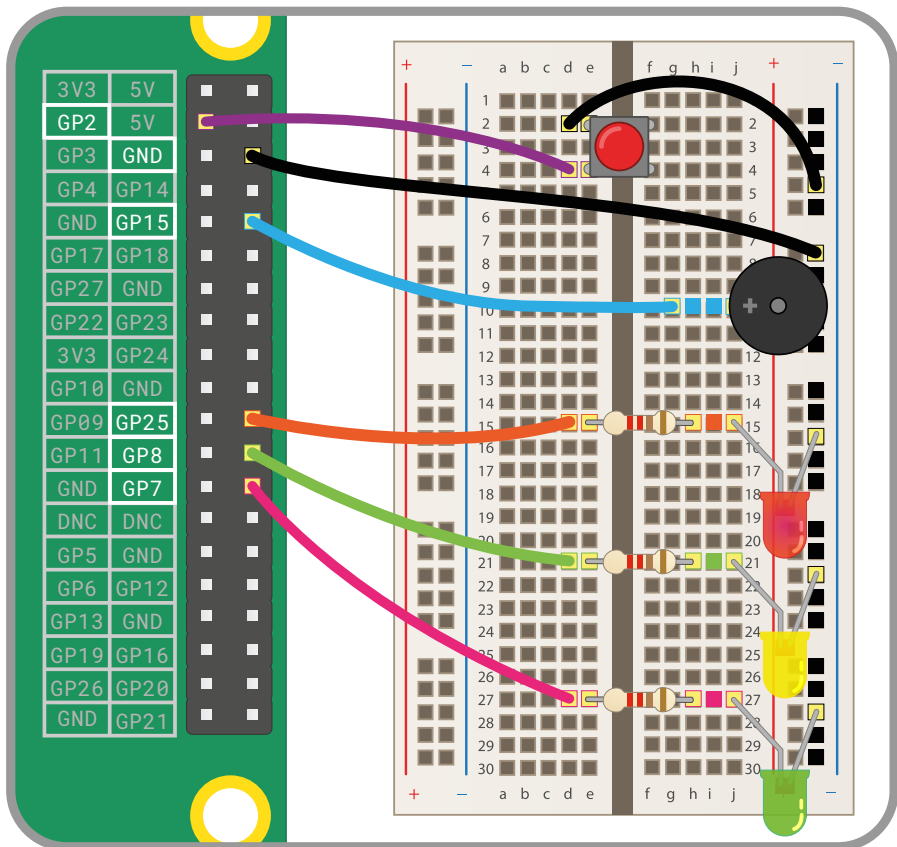
ブザー音を短くするにはプログラムをどのように変更すればよいですか?ブザーをボタンで制御できるように回路を構築できますか?




Scratch プロジェクト: 信号


ボタン、ブザー、LED を入力や出力として使用する方法を理解したところで、実用的なコンピューティングを構築してみましょう。横断歩道用の信号 (押しボタン付き) を作成します。このプロジェクトでは、ブレッドボード、LED (赤、黄、緑が 1 個ずつ)、330 Ω の抵抗器 (3 個)、ブザー、押しボタンスイッチ、オス - オス (M2M) ジャンパー線、オス - メス (M2F) ジャンパー線が必要になります。

まず回路を構築します (図 6-6)。ブザーを GPIO 15 ピン (図 6-6 で「GP15」とラベル付けされているピン) に接続します。赤の LED は GPIO 25 ピン (「GP25」)、黄の LED は GPIO 8 (「GP8」)、緑の LED は GPIO 7 (「GP7」)、スイッチは GPIO 2 (「GP2」) に接続します。330 Ω の抵抗器を GPIO ピンと LED の長い方の足の間に接続し、すべてのコンポーネントの 2 番目の足をブレッドボードのアース線に接続することに注意してください。最後に、アース線を Raspberry Pi のアースピン (「GND」とラベル付けされているピン) に接続して回路を完成させます。




▲ 図 6-6: 信号プロジェクトの配線図

新しい Scratch 3 プロジェクトを開いて、「 が押されたとき」ブロックをコード領域にドラッグします。次に、回路上で押しボタンスイッチに接続されている GPIO 2 ピンが出力ではなく入力であることを Scratch に指示する必要があります。ブロックパレットの「Raspberry Pi GPIO」カテゴ

リーから「set gpio to input pulled high」ブロックをドラッグし、「が押されたとき」ブロックの下に配置します。「0」の横に表示されている下矢印をクリックし、ドロップダウンリストから「2」を選択します。

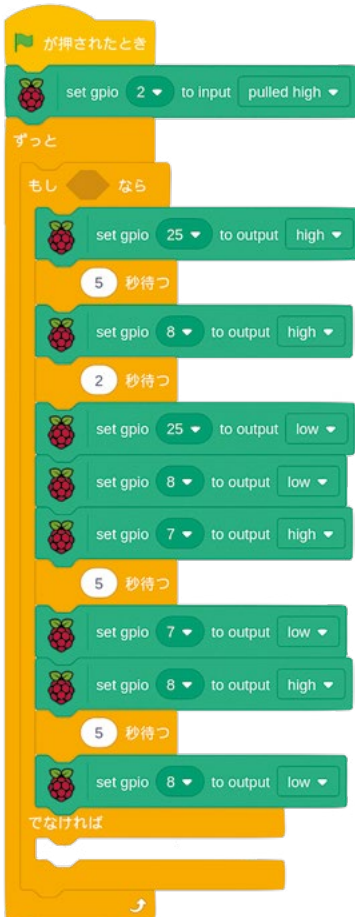


次に、信号シーケンスを作成する必要があります。「ずっと」ブロックをプログラムにドラッグし、空白の部分にブロックを挿入して、LED 信号のオン/オフが一定のパターンで切り替わるようにします。どの GPIO ピンにどのコンポーネントを接続しているか忘れないようにしてください。ピン 25 を使用しているときは赤の LED、ピン 8 をしているときは黄の LED、ピン 7 をしているときは緑の LED を使用していることとなります。

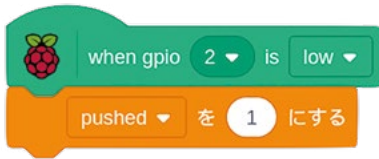


緑のフラグアイコンをクリックし、LEDを確認します。最初に赤が点灯し、次に赤と黄の両方が点灯し、次に緑が点灯し、最後に黄が点灯します。赤のLEDが再度点灯し、このシーケンスがもう一度繰り返されます。この点灯パターンは英国の信号と同じです。必要に応じて、他の国の信号に合わせてシーケンスを編集できます。

横断歩道をシミュレートするには、プログラムでボタンの押下を監視する必要があります。プログラムを現在実行している場合は、赤い八角形アイコンをクリックしてプログラムを停止します。「もし...なら...でなければ」ブロックをスクリプト領域にドラッグし、「ずっと」ブロックの真下に接続して、「if then」セクションに信号のシーケンスを設定します。ひし形のスペースは空白のままにしておきます。

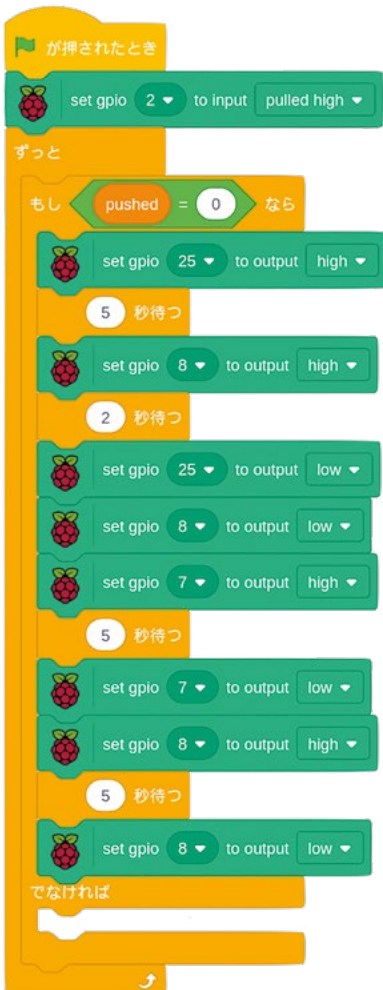


実際の横断歩道ではボタンを押してもライトはすぐに赤に変わりません。赤のライトはシーケンスの順番が来るまで待機します。この動作をプログラムに組み込むには、「when gpio is low」ブロックをコード領域にドラッグし、ドロップダウンリストから「2」を選択します。その下に「pushed を 1 にする」ブロックをドラッグします。



このブロックの組み合わせはボタンの押下を監視し、ボタンが押されると「pushed」変数を 1 に設定します。このように変数を設定すると、ボタンが押されたことを記憶させることができます。ボタンが押されてもただちにアクションをトリガーしない場合に便利です。

元の一連のブロックに戻り、「もし...なら」ブロックを見つけます。「」演算子ブロックを「もし...なら」ブロックのひし形スペースにドラッグし、「pushed」レポーターブロックを最初の空白スペースにドラッグします。ブロックの右側に「0」と入力し、「50」を上書きします。



緑のフラグアイコンをクリックし、信号がシーケンスに
 従って点灯するのを確認します。押しボタンスイッチを押
 します。最初は何も起きていないように見えますが、シー
 ケンスの終わりに達すると、黄の LED のみ点灯します。
 信号はしばらくするとオフになり、そのままの状態が続き
 ます。これは「pushed」変数を指定しているためです。

あとは、ライトをオフにすること以外の機能を横断歩
 道のボタンに追加するだけです。メインとなる一連のブ
 ロックで「**でなければ**」ブロックを見つけ、その中に「**set gpio 25 to output high**」ブロックをドラッグします。
 赤い LED が接続されているピンと一致するように、デ
 フォルトの GPIO ピン番号を変更することに注意してく
 ださい。

その下（「**でなければ**」内）にブザーのパターンを作
 成します。「**10 回繰り返す**」ブロックをドラッグし、このブ
 ロックに「**set gpio 15 to output high**」、「**0.2 秒待つ**」、
 「**set gpio 15 to output low**」、「**0.2 秒待つ**」の各ブ
 ロックを挿入して、GPIO ピンの値をブザーコンポーネン
 トのピンの値と一致するように変更します。

最後に、「**10 回繰り返す**」ブロック
 の下（「**でなければ**」ブロック内）に「**set gpio 25 to output low**」ブロックと
 「**pushed を 0 にする**」ブロックを追加します。最後の
 ブロックはボタンの押下を格納する変数をリセットし、
 ブザーのシーケンスが永遠に繰り返されないようににし
 ます。

緑のフラグアイコンをクリックし、ブレッドボード上の
 スイッチを押します。シーケンスが完了すると、赤いライ
 トが点灯してブザーが鳴り、安全に横断できることを歩
 行者に知らせます。数秒経つとブザーは停止し、信号の
 シーケンスが再開され、次にボタンが押されるのを待機
 します。

これで、横断歩道の信号を独自にプログラムすること
 に成功しました。

```

  が押されたとき
  set gpio 2 to input pulled high
  ずっと
  もし pushed = 0 なら
    set gpio 25 to output high
    5 秒待つ
    set gpio 8 to output high
    2 秒待つ
    set gpio 25 to output low
    set gpio 8 to output low
    set gpio 7 to output high
    5 秒待つ
    set gpio 7 to output low
    set gpio 8 to output high
    5 秒待つ
    set gpio 8 to output low
  でなければ
    set gpio 25 to output high
    10 回繰り返す
    set gpio 15 to output high
    0.2 秒待つ
    set gpio 15 to output low
    0.2 秒待つ
    set gpio 25 to output low
    pushed を 0 にする
  when gpio 2 is low
  pushed を 1 にする
  
```



チャレンジ: 改良してみよう

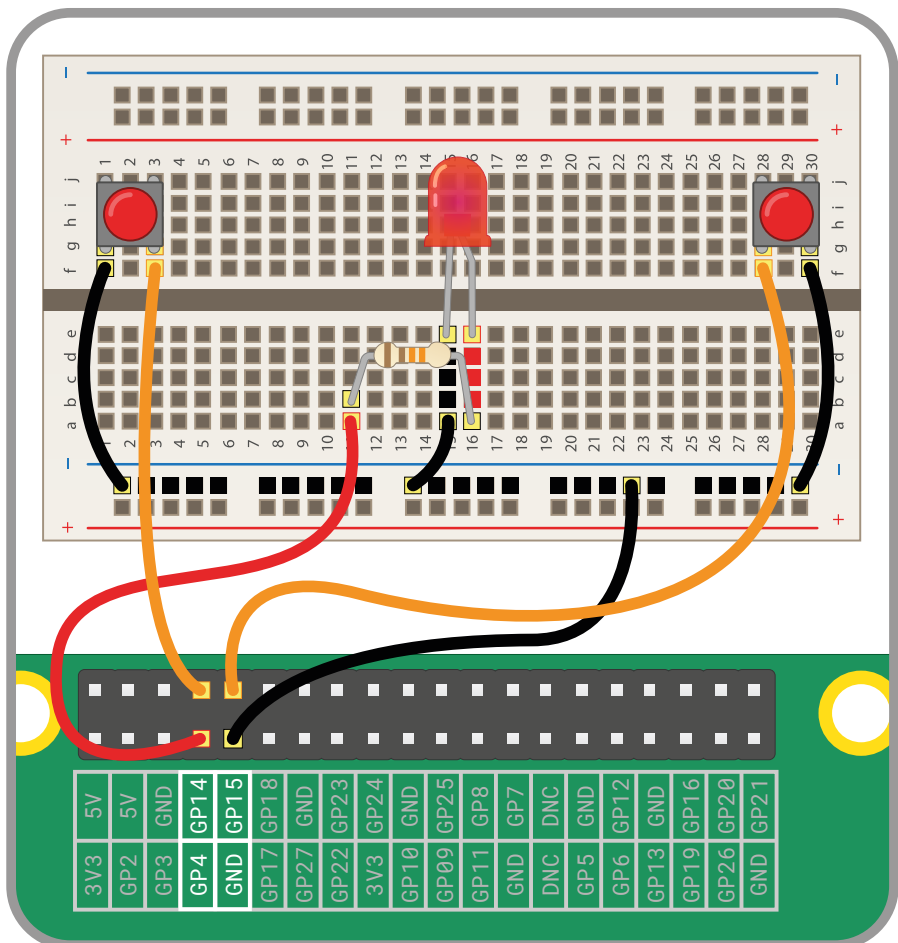
歩行者が横断する時間が長くなるようにプログラムを変更
 できますか?他の国の信号パターンに関する情報を見つけ、
 その信号パターンに合わせてプログラムを再作成できま
 すか?LED の明るさを下げることはできますか?



Python プロジェクト: クイック反応ゲーム

ボタンと LED を入力や出力として使用する方法を理解したところで、実用的なコンピューティングを構築してみましょう。2 人用のクイック反応ゲームです。どちらが早く反応するかを確認できるように設計します。このプロジェクトでは、ブレッドボード、LED、330 Ω の抵抗器、押しボタンスイッチ (2 個)、オス - メス (M2F) ジャンパー線、オス - オス (M2M) ジャンパー線が必要になります。

まず回路を構築します (図 6-7)。ブレッドボードの左側にある最初のスイッチを GPIO 14 ピン (図 6-7 で「GP14」とラベル付けされているピン) に接続し、ブレッドボードの右側にある 2 番目のスイッチを GPIO 15 ピン (「GP15」とラベル付けされているピン) に接続します。LED の長い方の足は 330 Ω の抵抗器に接続し、抵抗器は Raspberry Pi の GPIO 4 ピン (「GP4」とラベル付けされているピン) に接続します。すべてのコンポーネントの 2 番目の足はブレッドボードのアース線に接続します。最後に、アース線を Raspberry Pi のアースピン (「GND」とラベル付けされているピン) に接続します。



▲ 図 6-7: クイック反応ゲームの配線図

Thonny で新しいプロジェクトに「**Reaction Game**」という名前を付けて保存します。GPIO Zero ライブラリーの **LED** 関数と **button** 関数、および time ライブラリーの **sleep** 関数を使用します。時間を節約するため、2 つの GPIO Zero 関数を別々の行でインポートする代わりに、これらの関数をコンマ記号 (,) で区切って一緒にインポートします。スクリプト領域で以下のように入力します。

```
from gpiozero import LED, Button
from time import sleep
```

これまでと同様、2 つのボタンと LED がどのピンに接続されているかを GPIO Zero に指示する必要があります。以下のように入力します。

```
led = LED(4)
right_button = Button(15)
left_button = Button(14)
```

次に、LED のオンとオフを切り替える命令を追加して、LED が正しく機能しているか確認できるようにします。

```
led.on()
sleep(5)
led.off()
```

「Run」ボタンをクリックします。LED が 5 秒間点灯した後オフになり、プログラムが終了します。ただし、反応ゲームの目的を考えると、毎回 5 秒後に LED が消灯するのでは、プレイヤーが LED の動作を簡単に予測できてしまいます。**from time import sleep** 行の後に以下の行を追加します。

```
from random import uniform
```

random ライブラリーは名前が示すように乱数を生成することを可能にします (ここでは一様分布を使用します。「[rpf.io/uniform](#)」を参照)。**sleep(5)** 行を見つけて以下のように変更します。

```
sleep(uniform(5, 10))
```

もう一度「Run」ボタンをクリックします。今回は LED の点灯時間が 5 秒間から 10 秒間の範囲でランダムに変わります。LED が消灯するまでの時間を確認し、「Run」ボタンを何度かクリックしてください。実行するたびに点灯時間が変わり、プログラムの動作が予測しにくくなったことがわかります。

ボタンを各プレイヤーのトリガーに変えるには、関数を追加する必要があります。プログラムの最後に移動し、以下のように入力します。

```
def pressed(button):
    print(str(button.pin.number) + " がゲームに勝ちました")
```

Python はインデントを使ってどの行が関数かを認識することに注意してください。Thonny は 2 行目を自動的にインデントします。最後に以下の 2 行を追加し、ボタンを押しているプレーヤーを検出します。インデントしないように注意してください。インデントすると、Python はこれらの行を関数の一部として処理します。

```
right_button.when_pressed = pressed
left_button.when_pressed = pressed
```

プログラムを実行します。今回は LED が消灯した後、すぐに 2 つのボタンのいずれかを押してみてください。最初のボタンが押されたことを示すメッセージが、Thonny ウィンドウの下部にある Python のシェルに出力されます。どちらのボタンを押してもそのたびにメッセージが表示されます。また、メッセージにはわかりやすいボタンの名前ではなく、ピン番号が使用されています。

これを修正するため、プレイヤーに名前を尋ねることにします。`from random import uniform` 行の下に、以下のように入力します。

```
left_name = input("左側のプレイヤーの名前 ")
right_name = input("右側のプレイヤーの名前 ")
```

関数に戻り、`print(str(button.pin.number) + " がゲームに勝ちました")` 行を以下で置き換えます。

```
if button.pin.number == 14:
    print(left_name + " がゲームに勝ちました")
else:
    print(right_name + " がゲームに勝ちました")
```

「Run」ボタンをクリックし、Python のシェル領域に両方のプレイヤーの名前を入力します。LED が消灯した後にできるだけ早くボタンを押すと、ピン番号ではなくプレイヤー名が出力されます。

ボタンを押すたびにゲームに勝ったというメッセージが表示される問題を修正するには、`sys` (system の略語) ライブラリーの `exit` 関数を新たに追加します。最後の `import` 行の下で、以下のように入力します。

```
from os import _exit
```

関数の最後にある `print(right_name + " がゲームに勝ちました")` 行の下で、以下のように入力します。

```
    _exit(0)
```

ここではインデントが重要になります。`_exit(0)` は 4 つのスペースを使用してインデントし、2 行上にある `else:` およびその 2 行上にある `if` と文頭を揃える必要があります。この命令は、最初のボタンが押された後、プログラムを停止するように Python に指示します。つまり、プレイヤーが 2 番目にボタンを押しても何も発生しません。

完成したプログラムは以下のようになります。

```
from gpiozero import LED, Button
from time import sleep
from random import uniform
from os import _exit

left_name = input("左側のプレイヤーの名前 ")
right_name = input("右側のプレイヤーの名前 ")
led = LED(4)
right_button = Button(15)
left_button = Button(14)

led.on()
sleep(uniform(5, 10))
led.off()

def pressed(button):
    if button.pin.number == 14:
        print(left_name + " がゲームに勝ちました")
    else:
        print(right_name + " がゲームに勝ちました")
    _exit(0)

right_button.when_pressed = pressed
left_button.when_pressed = pressed
```

「Run」ボタンをクリックしてプレイヤーの名前を入力し、LED が消灯するのを待つと、勝利したプレイヤーの名前が表示されます。Python 自体からのメッセージも表示されます。**Backend terminated or disconnected .Use 'Stop/Restart' to restart ...**これは Python が `_exit(0)` コマンドを受信してプログラムを停止したことを意味しますが、「Stop」アイコンをクリックしてプログラムを完全に終了し、次のラウンドに備える必要があることも意味します (図 6-8)。

```

Thonny - /home/pi/Downloads/Reaction Game.py @ 24:35
New Load Save Run Debug Over Info Out Stop Zoom Quit

Reaction Game.py ✕
9 right_button = Button(15)
10 left_button = Button(14)
11
12 led.on()
13 sleep(uniform(5, 10))
14 led.off()
15
16 def pressed(button):
17     if button.pin_number == 14:
18         print(left_name + " won the game")
19     else:
20         print(right_name + " won the game")
21     _exit(0)
22
23 right_button.when_pressed = pressed
24 left_button.when_pressed = pressed

Shell
>>> %Run 'Reaction Game.py'
Left player name is Gareth
Right player name is Eben
>>> Gareth won the game

Backend terminated or disconnected. Use 'Stop/Restart' to restart ...

```

▲ 図 6-8:勝者が決まったらプログラムを停止する必要があります

これで、独自の物理的ゲームの作成に成功しました。



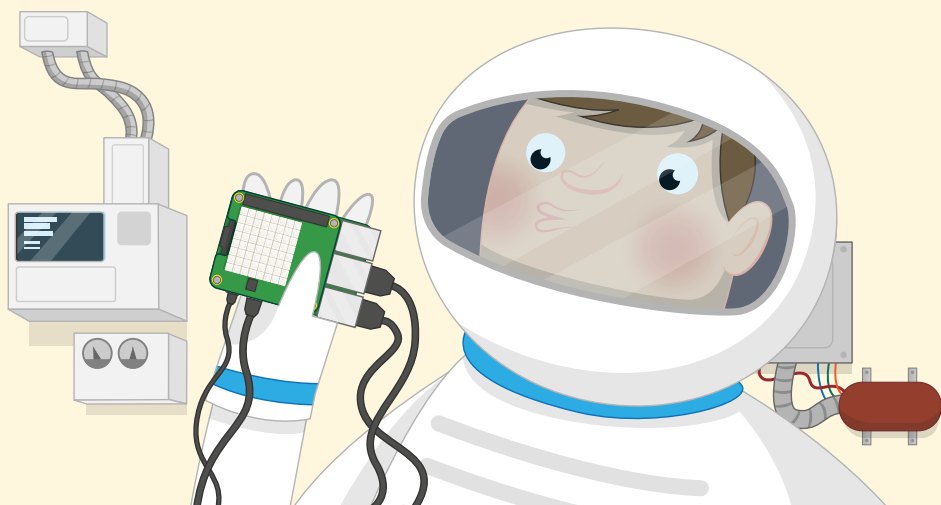
チャレンジ: ゲームを改良してみよう

ゲームが継続的に実行されるようにループを追加できますか?最初に `_exit(0)` 命令を削除することに注意してください。ゲームを複数回競った場合にだれが勝利したかわかるようにスコアカウンターを追加できますか?ライトの消灯に反応するまでの時間がわかるようにタイマーを追加できますか?

第7章

Sense HAT を使って 物理的コンピューテ ィングに挑戦しよう

国際宇宙ステーションで使用されている Sense HAT は、センサーと LED マトリックスディスプレイを備えた Raspberry Pi 用の多機能アドオンボードです。



Raspberry Pi では、Hardware Attached on Top (HAT) という特殊なアドオンボードを使用することができます。Raspberry Pi の HAT には、マイク、ライト、電子リレー、スクリーンなど、さまざまなものを追加できますが、その中でも非常に特徴的なのが Sense HAT と呼ばれる HAT です。

Sense HAT は、「Astro Pi プロジェクト」という宇宙ミッション用に特別に設計された HAT です。Raspberry Pi Foundation、イギリス宇宙局、欧州宇宙機関が参加したこの共同プロジェクトでは、オービタル・サイエンシズ社が開発したシグナスという無人宇宙補給機に Raspberry Pi のボードと Sense HAT が乗せられて、国際宇宙ステーションに運ばれました。安全に地球上空の軌道に乗った Sense HAT は、それ以来、ヨーロッパ全域の子どもたちが書いたコードを実行し、科学的な実験を行っています（宇宙飛行士たちは、Sense HAT を「Ed」と「Izzy」という愛称で呼んでいます）。

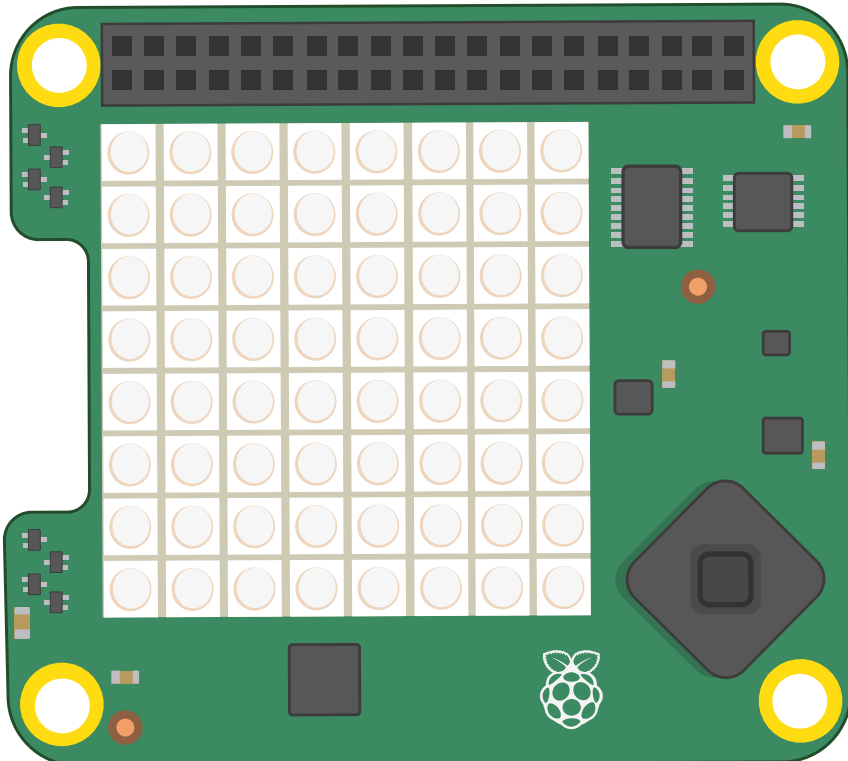
Ed と Izzy は地球から遠く離れた宇宙空間で活躍していますが、同じタイプの Sense HAT ハードウェアは、Raspberry Pi のすべての販売店で購入することができます。また、Sense HAT を今すぐ購入しなくても、ソフトウェア上で Sense HAT のシミュレーションを実行することができます。

エミュレーターについて

この章の説明は、Raspberry Pi の GPIO ヘッダーに Sense HAT を取り付けて使用することを前提としていますが、Sense HAT を使用しない場合は、「Sense HAT を取り付ける」というセクションを飛ばして、プログラムの作成にチャレンジしてもかまいません。Sense HAT がなくても、エミュレーターが Sense HAT の代わりに機能します。

Sense HAT の概要

Sense HAT は、Raspberry Pi 用の強力な多機能なアドオンボードです。Sense HAT には、数百万の色を表現できる 64 個 (8 x 8) のプログラマブル LED (赤、緑、青の LED。赤、緑、青の組み合わせを、その頭文字をとって RGB といいます) だけでなく、5 方向のジョイスティックコントローラーと 6 個のオンボードセンサーも搭載されています。



ジャイロスコopセンサー: 時間の経過に伴う角度の変化 (角速度 といいます) を感知するためのセンサーです。地球の重力の方向を追跡することにより、角速度が測定されます (重力とは、地球の中心に向かって物体が引き下げられる力のことです)。わかりやすく言うと、Sense HAT を回転させたときに、地球の表面に対してどれくらいの速度で回転しているかを計測するのが、ジャイロスコopセンサーということです。

加速度計: このセンサーはジャイロスコopセンサーに似ていますが、地球の重力に対する角度を検知するのではなく、複数の方向における加速度を測定するためのセンサーです。ジャイロスコopセンサーと加速度計のデータを組み合わせると、Sense HAT の方向と動作を追跡することができます。

磁力計: これは、磁場の強さを測定するためのセンサーです。地球の自然磁場を測定して磁北方向を判断することにより、Sense HAT の動作を追跡することもできます。このセンサーを使用して、金属製の物体や電界を検知することもできます。これら 3 つのセンサー (ジャイロスコopセンサー、加速度計、磁力計) は、すべて同じチップに組み込まれています。Sense HAT の回路基板上に配置されているこのチップには、「ACCEL/GYRO/MAG」というラベルが付いています。

湿度センサー: これは、空気中に含まれる水蒸気の種類 (相対湿度 といいます) を測定するためのセンサーです。相対湿度の範囲は、0% (空気中にまったく水蒸気が含まれていない状態) から 100% (空気が水蒸気で完全に飽和している状態) です。湿度データを使用して、いつ雨が降り出すかを予測することができます。

気圧センサー: これは、気圧を測定するためのセンサーです (気圧計 ともいいます)。「気圧」という用語は天気予報などでよく聞きますが、どういう意味なのかよくわからない人もいるでしょう。たとえば、高い山に登ると次第に空気が薄くなって息が苦しくなります。このように、海面から離れるほど空気が薄くなり、気圧が下がっていきます。気圧計は、こうした気圧の変化を測定するためのセンサーです。

温度センサー: これは、Sense HAT 周辺の温度を測定するためのセンサーです。この温度は、Sense HAT 自体の温度によっても影響を受けます。たとえば Sense HAT をケースで覆った場合、Sense HAT 周辺の温度は、Sense HAT 自体が放出する熱によって通常よりも高くなる場合があります。Sense HAT には、個別の温度センサーはありません。Sense HAT の温度センサーは、湿度センサーと気圧センサー内に組み込まれています。プログラムで、湿度センサーと気圧センサーのいずれか 1 つだけを使用することも、両方のセンサーを使用することもできます。



Raspberry Pi 400 の Sense HAT

Sense HAT は Raspberry Pi 400 と互換性があるため、Raspberry Pi 400 背面の GPIO ヘッダーに直接挿入することができますが、挿入すると LED があなたとは反対側の向きになり、上下逆さの状態になってしまいます。

これに対処するには、GPIO 用の延長ケーブルまたは拡張ボードが必要になります。対応する拡張ボードとしては、Pimoroni 社の Black HAT Hack3r シリーズがあります。Sense HAT を Black HAT Hack3r ボード本体とともに使用することも、Black HAT Hack3r ボードに付属している 40 ピンリボンケーブルを延長ケーブルとして使用することもできます。延長ケーブルや拡張ボードを使用する場合は、製造元のマニュアルを参照して、正しい方法で使用してください。

Sense HAT を取り付ける

Sense HAT ハードウェアを購入した場合は、最初にパッケージを開けて、必要な部品 (Sense HAT 本体、4 つの金属製またはプラスチック製のスペーサー、8 本のネジ) が揃っているかどうかを確認します。何本かの金属ピンが差し込まれた黒いプラスチック片が付属している場合があります (Raspberry Pi の GPIO ピンに似ています)、その場合は、ピンを上に向けた状態で、カチッという音がするまでプラスチック片を Sense HAT の下部に押し込みます。

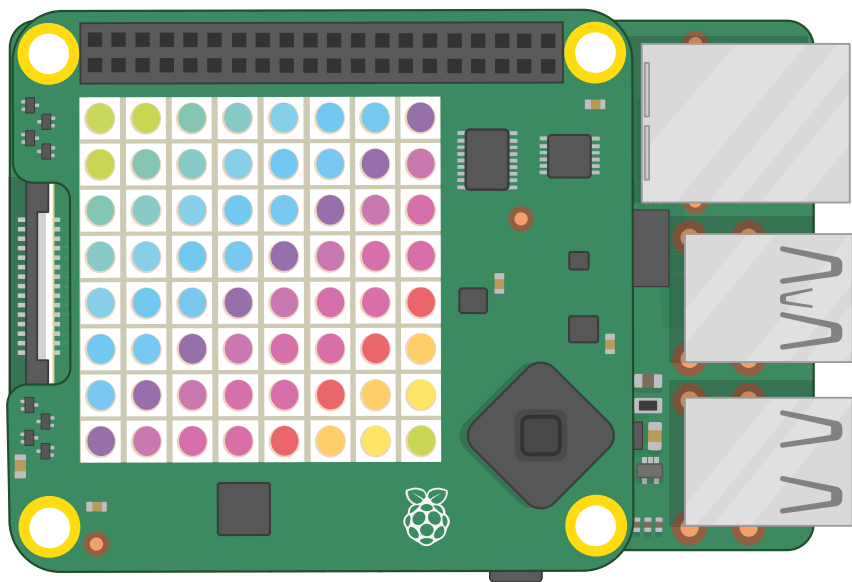
スペーサーは、ジョイスティックを使用したときに Sense HAT が曲がらないようにするために付属しています。スペーサーを取り付けなくても Sense HAT は機能しますが、スペーサーを取り付けると、Sense HAT、Raspberry Pi、GPIO ヘッダーが破損しないように保護することができます。

注意

Hardware Attached on Top (HAT) モジュールを GPIO ヘッダーに差し込んだり取り外したりする場合は、必ず Raspberry Pi の電源を切るようにしてください。HAT モジュールを GPIO ヘッダーに差し込む場合は、HAT モジュールを平らにして、GPIO ヘッダーのピンに合わせてゆっくりと差し込んでください。

Raspberry Pi の四隅にある取り付け穴に、4 本のネジを上に向けて下から押し入れ、これらのネジの上にスペーサーを取り付けます。次に、Sense HAT を Raspberry Pi の GPIO ヘッダーまで押し下げます。GPIO ピンに合わせて、できるだけ平らになるように押し下げてください。最後に、取り付け穴に差し込んだ 4 本のネジをスペーサーに固定します。Sense HAT が正しく取り付けられていれば、ジョイスティックを操作しても Sense HAT が曲がったりぐらついたりすることはありません。

Raspberry Pi の電源を入れると、Sense HAT の LED がレインボーパターンで点灯し (図 7-1)、その後消灯します。これで、Sense HAT の取り付けが完了しました。



▲ 図 7-1: 初めて電源を入れると、LED がレインボーパターンで点灯する

Sense HAT を取り外す場合は、上部のネジを外して (小さなドライバーが必要になる場合があります) HAT を上方向に取り外してから、スペーサーを取り外してください。HAT は非常にしっかりと固定されているため、取り外すときに GPIO ヘッダーを曲げたりしないように注意してください。

Sense HAT を使ってみよう

初めて Sense HAT を操作すると、LED マトリクスディスプレイにウェルカムメッセージが表示されます。最初にウェルカムメッセージが表示されるのは、どのプログラミングプロジェクトでも同じです。Sense HAT エミュレーターを使用する場合は、Raspberry Pi OS メニューアイコンをクリックして「プログラミング」カテゴリを選択し、「Sense HAT Emulator」をクリックします。

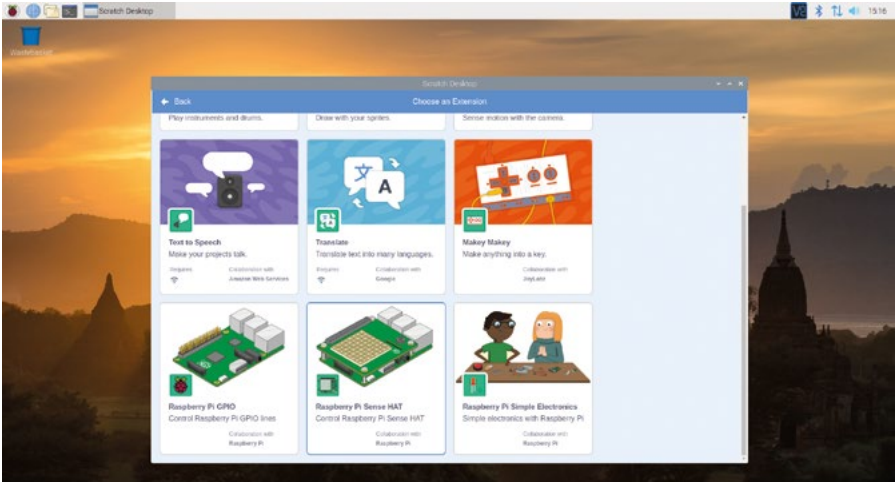


プログラミングに関する知識について

この章の説明は、Scratch 3 または Python (あるいはその両方) と Thonny の統合開発環境 (IDE) に関する知識があることを前提としています。これらに関する知識がない場合は、「第 4 章: Scratch を使ってプログラミングしてみよう」または「第 5 章: Python を使ってプログラミングしてみよう」(あるいはその両方) を最初に読んでください。

Scratch を起動する

Raspberry Pi OS メニューから Scratch 3 を起動します。次に、Scratch ウィンドウの左下にある「Add Extension」ボタンをクリックし、「Raspberry Pi Sense HAT」拡張機能をクリックします（図 7-2）。これにより、LED マトリックスディスプレイをはじめとする Sense HAT のさまざまな機能を制御するための各種ブロックが読み込まれます。これらのブロックは、Raspberry Pi の「Sense HAT」カテゴリーに表示されます。

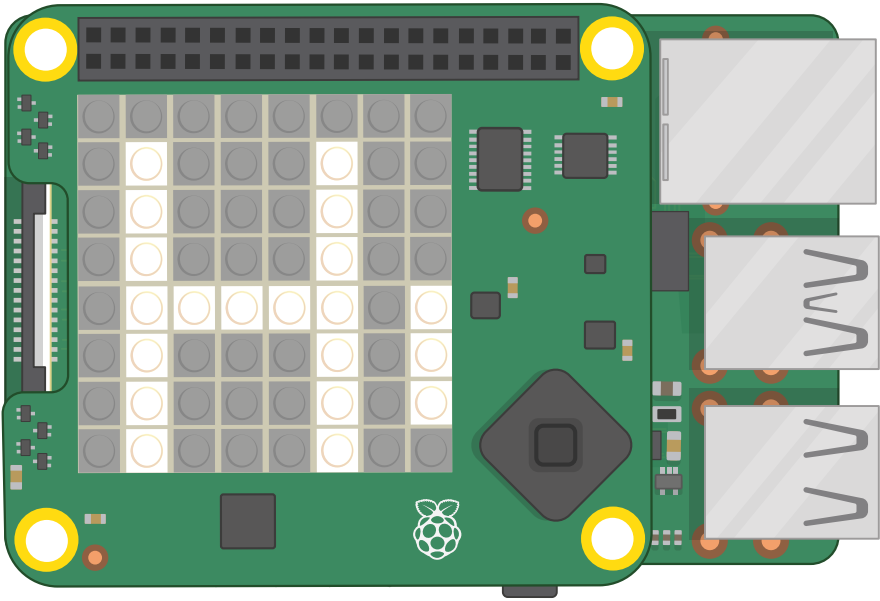


▲ 図 7-2: Raspberry Pi の Sense HAT 拡張機能を Scratch 3 に追加する

最初に、「**が押されたとき**」イベントブロックをスクリプト領域にドラッグし、そのすぐ下に「**display text Hello!**」ブロックをドラッグします。次に、ブロック内のテキストを「**display text Hello, World!**」に変更します。

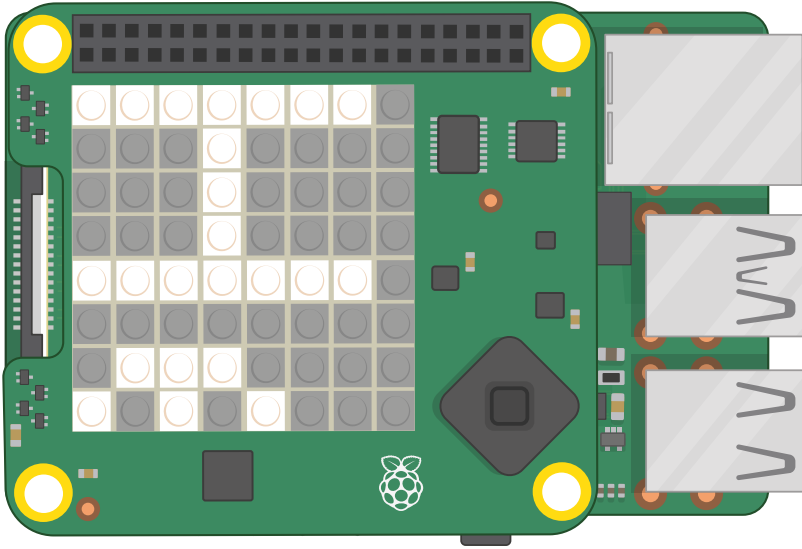


ステージ領域の緑のフラグアイコンをクリックして、Sense HAT (または Sense HAT エミュレータ) を確認してみましょう。Sense HAT の LED マトリックスディスプレイに「Hello, World!」というメッセージが 1 文字ずつゆっくりとスクロール表示されます（図 7-3）。うまく表示されたら成功です。



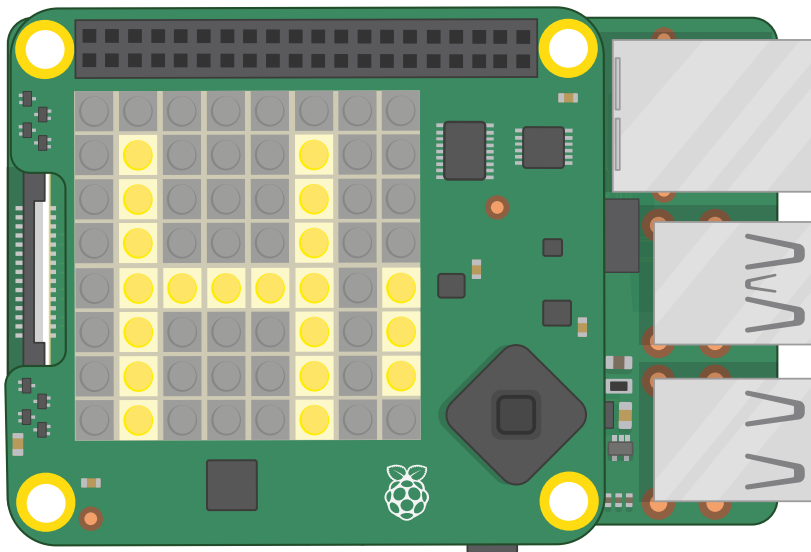
▲ 図 7-3: メッセージが 1 文字ずつ LED マトリクスディスプレイに表示される

これで、簡単なメッセージをスクロール表示できるようになりました。次に、このメッセージの表示方法を変更してみましょう。表示するメッセージを変更するだけでなく、メッセージの表示方向を変更することもできます。ブロックパレット上の「**set rotation to 0 degrees**」ブロックを「**が押されたとき**」ブロックと「**display text Hello, World!**」ブロックの間にドラッグし、「0」の横に表示されている下矢印をクリックして「90」に変更します。緑のフラグアイコンをクリックすると、前と同じメッセージが表示されますが、左から右に向かって表示されるのではなく、下から上に向かって表示されます (図 7-4)。



▲ 図 7-4: メッセージが縦方向にスクロール表示される

次に、メッセージの色を変えてみましょう。値を「90」から「0」に戻し、「set rotation to 0 degrees」ブロックと「display text Hello, World!」ブロックの間に「set colour」ブロックをドラッグします。次に、「set colour to」ブロックの最後に表示されている色の部分をクリックして Scratch のカラーパレットを表示し、明るい黄色を選択します。この状態で緑のフラグアイコンをクリックすると、メッセージが明るい黄色で表示されます (図 7-5)。

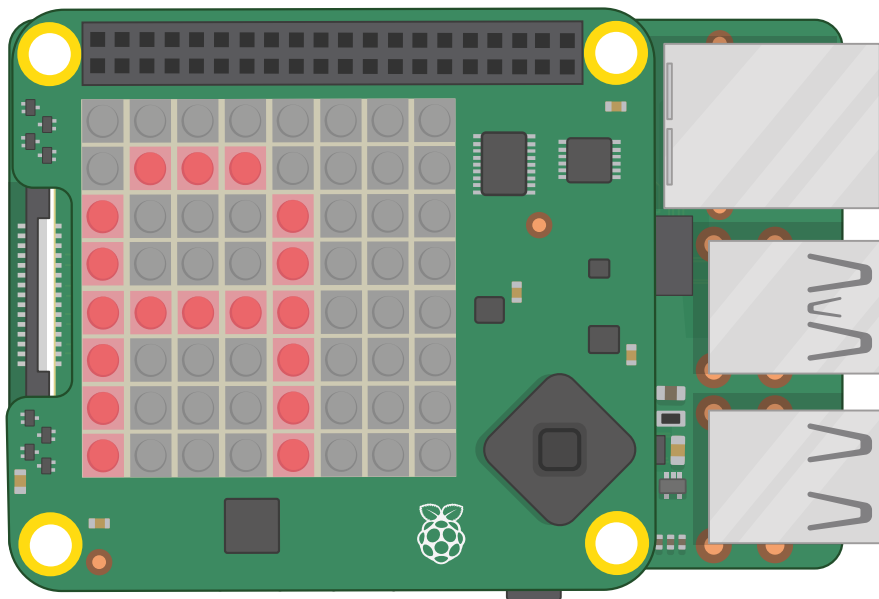


▲ 図 7-5: メッセージの色を変える

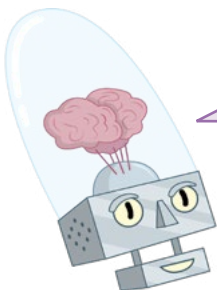
次に、「set colour to 黄」ブロックと「display text Hello, World!」ブロックの間に「set background」ブロックをドラッグし、「set colour to」ブロックの色の部分をクリックしてカラーパレットをもう一度表示します。

今回は、メッセージ以外の場所（背景といいます）にある LED の色を変えてみます。カラーパレットで青を選択して緑のフラグアイコンをクリックすると、文字が明るい黄色で表示され、背景が青で表示されます。いくつか色を変えてみて、好きな色の組み合わせを試してみましょう。色の組み合わせを間違えると、メッセージが見にくくなるので注意してください。

ここまではメッセージ全体をスクロール表示してきましたが、文字を個別に表示することもできます。「display text」ブロックをスクリプト領域外にドラッグして削除し、代わりに「display character A」ブロックをスクリプト領域にドラッグします。この状態で緑のフラグアイコンをクリックすると、「A」という文字だけが表示されます。別の動作を指示しないかぎり、この文字はスクロールすることも消えることもなく、表示されたままになります。このブロックには、「display text」ブロックで設定した色がそのまま適用されます。この色を赤に変えてみましょう（図 7-6）。



▲ 図 7-6: 1 文字だけ表示する



チャレンジ: メッセージを繰り返し表示する ?

ループ処理の知識を活用して、メッセージをのスクロール表示を繰り返してみましょう。また、1 文字ごとに色を変えてメッセージを表示してみましょう。

Python を起動する

raspberry メニューアイコンをクリックして「プログラミング」カテゴリを選択し、「Thonny」をクリックします。Sense HAT エミュレーターに重なる形で Thonny ウィンドウが表示されている場合は、エミュレーターまたはウィンドウ上部の青いタイトルバーでマウスボタンをクリックしたまま、エミュレーターとウィンドウの両方が表示される位置までドラッグします。



Python コード行の変更

物理的なハードウェアとしての Sense HAT 用に記述された Python コードは、1 行を変更するだけで、Sense HAT エミュレーター上で実行できるようになります（逆の場合も同様です）。Sense HAT エミュレーター上で Python コードを実行する場合は、この章に記載されている `sense_hat import SenseHat` という行を、すべて `from sense_emu import SenseHat` に変更します。この行を元に戻すと、物理的な Sense HAT ハードウェア上で Python コードが実行されるようになります。

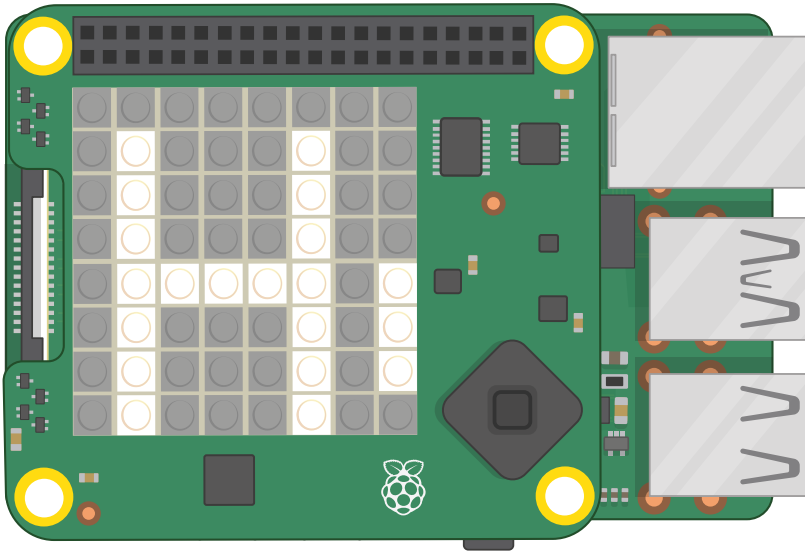
Python プログラムで Sense HAT または Sense HAT エミュレーターを使用する場合は、Sense HAT ライブラリーをインポートする必要があります。そのためには、スクリプト領域で以下のように入力します。Sense HAT エミュレーターを使用する場合は、`sense_hat` の部分を `sense_emu` に変えてください。

```
from sense_hat import SenseHat
sense = SenseHat()
```

Sense HAT ライブラリーには、メッセージを取得して LED マトリックスディスプレイでスクロール表示できる形式に変換するための「`show_message ()`」というシンプルな関数が用意されています。以下のように入力します。

```
sense.show_message("Hello, World!")
```

プログラムを「Hello Sense HAT」という名前で保存して「Run」ボタンをクリックします。Sense HAT の LED マトリックスディスプレイに、「Hello, World!」というメッセージが 1 文字ずつゆっくりとスクロール表示されます（図 7-7）。このメッセージが表示されれば、このプログラムは成功です。



▲ 図 7-7: メッセージが 1 文字ずつ LED マトリックスディスプレイに表示される

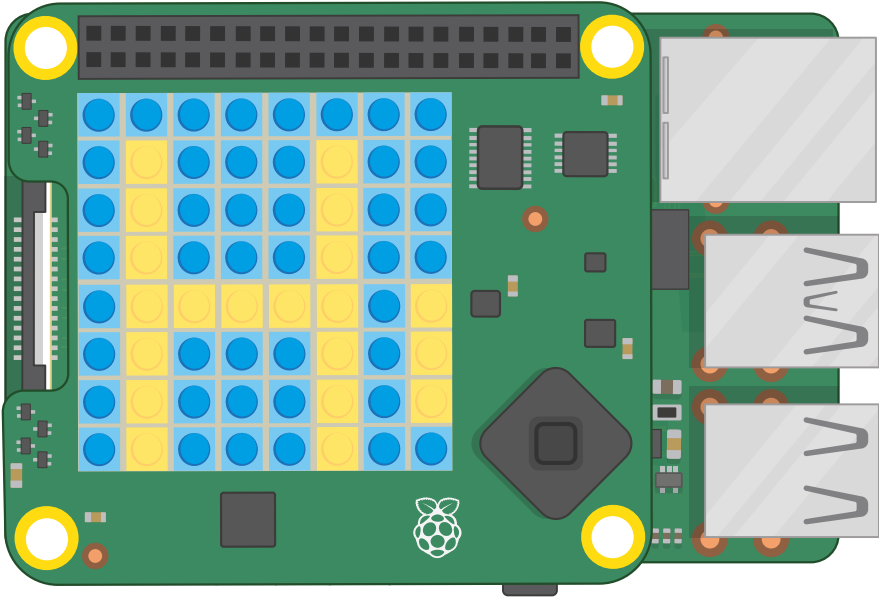
`show_message()` 関数でできることは、これだけではありません。プログラムに戻り、最後の行を以下のように変更します。

```
sense.show_message("Hello, World!", text_colour=(255, 255, 0),  
back_colour=(0, 0, 255), scroll_speed=(0.05))
```

コンマ (,) で区切られている各コード部分のことを、パラメーターといいます。パラメーターにより、`show_message()` 関数のさまざまな動作を制御することができます。たとえば `scroll_speed=()` は、メッセージのスクロール速度を変更するためのパラメーターです。ここで指定されている「0.05」という値ではおよそ2倍の速度になります。値を大きくすると、速度は遅くなります。

`text_colour=()` は、テキストの色を設定するためのパラメーターで、`back_colour=()` は、背景の色を設定するためのパラメーターです (どちらのパラメーターも、アメリカ英語の「color」ではなくイギリス英語の「colour」で記述しますが、ほとんどの Python コードはアメリカ英語のスペルで記述します)。ただし、これらのパラメーターで色の名前 (「black」や「red」など) を指定することはできません。3 つの数値を使用して、色を指定する必要があります。最初の数値は、その色に含まれる赤の量を指定するための値です。0 から 255 までの範囲(値が大きくなるほど赤の量が多くなります)で任意の数値を指定します。2 番目の数値は緑の量を指定するための値で、最後の数値は青の量を指定するための値です。これらの色の組み合わせは、赤 (Red)、緑 (Green)、青 (Blue) のそれぞれの頭文字をとって RGB と呼ばれます。

「Run」アイコンをクリックして Sense HAT の LED マトリックスディスプレイを確認してみましょう。これまでよりもメッセージのスクロール速度が上がり、青い背景と明るい黄色の文字が表示されます (図 7-8)。パラメーターの値をいろいろと変えてみて、自分に合った表示速度と色の組み合わせを見つけてください。



▲ 図 7-8: メッセージと背景の色を変える

変数を作成して、わかりやすい色の名前を付けることができます。これを行うには、`sense.show_message()` 行の上に以下のコードを追加します。

```
yellow = (255, 255, 0)
blue = (0, 0, 255)
```

次に、`sense.show_message()` 行を以下のように変更します。

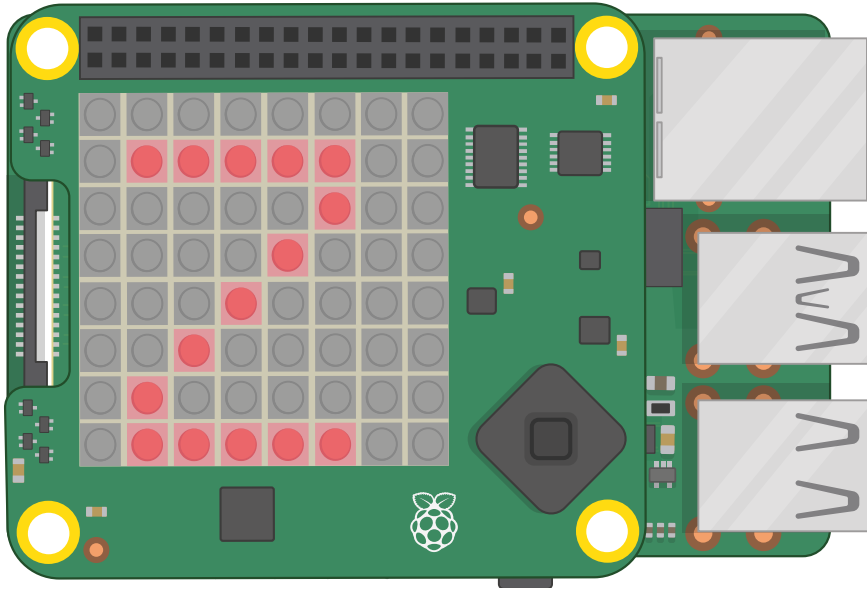
```
sense.show_message("Hello, World!", text_colour=(yellow), back_
colour=(blue), scroll_speed=(0.05))
```

この状態でもう一度「Run」アイコンをクリックすると、前と同じように青い背景と黄色のメッセージが表示されます。しかし今回は、コード内で「yellow」と「blue」という変数が使用されているため、コードを一目見ただけでどんな色が設定されているのかがすぐにわかります。必要な数の色を変数として作成することができます。たとえば赤の場合は、「255, 0, 0」、白の場合は、「255, 255, 255」、黒の場合は、「0, 0, 0」などのように指定します。

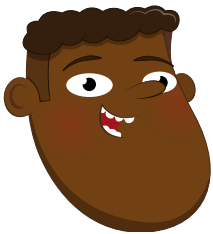
メッセージ全体をスクロール表示だけでなく、文字を個別に表示することもできます。`sense.show_message()` 行をすべて削除し、代わりに以下のように入力します。

```
sense.show_letter("Z")
```

この状態で「Run」をクリックすると、Sense HAT の LED マトリックスディスプレイに「Z」という文字が表示されます。この場合、自動的にスクロール表示されるメッセージとは異なり、ディスプレイには「Z」が表示されたままになります。`sense.show_message()` 関数で指定したパラメーターを `sense.show_letter()` 関数でも使用して、文字の色を赤に変えてみましょう (図 7-9)。



▲ 図 7-9: 1 文字だけ表示する



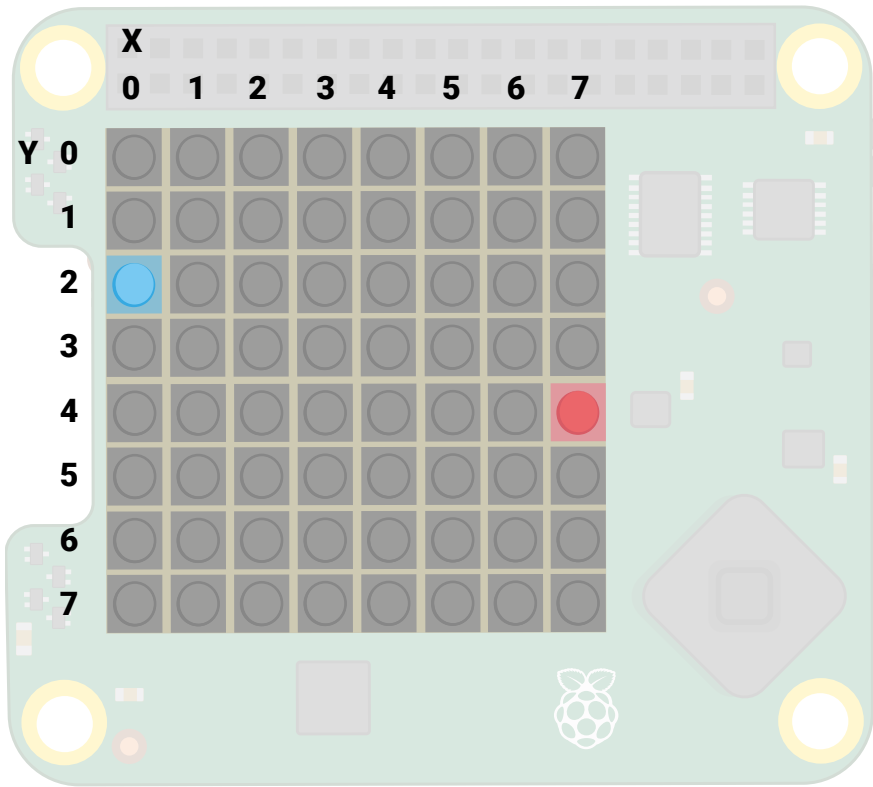
チャレンジ: メッセージを繰り返し表示する ?

ループ処理の知識を活用して、メッセージをのスクロール表示を繰り返してみましょう。また、1 文字ごとに色を変えてメッセージを表示してみましょう。どれくらいの速度でメッセージを表示できますか？

次のステップ: LED で絵を描く

Sense HAT の LED マトリックスディスプレイには、メッセージだけでなく絵 (パターン) を表示することもできます。それぞれの LED を画像内の 1 つのピクセルとして操作できるため、絵だけでなくアニメーションも作成することができます (ピクセルは、*picture element* の略称です)。

ただし、LED で絵を描くには、それぞれの LED の設定を変更するための手段が必要になります。これを行うには、Sense HAT の LED マトリックスディスプレイのレイアウトを理解し、それぞれの LED のオンとオフを正しく切り替えるためのプログラムを作成する必要があります。



▲ 図 7-10: LED マトリックスディスプレイのレイアウト (座標系)

ディスプレイの各行には 8 つの LED が配置されています。行は全部で 8 行あるため、LED の合計数は 64 個になります (図 7-10)。LED を数えるときは、(Python を含むほとんどのプログラミング言語では) 0 から始めて 7 で終わる数え方をします。最初の LED はディスプレイの左上隅にある LED で、最後の LED は右下隅にある LED です。マトリックスディスプレイ上の特定の LED を指定する場合は、行番号と列番号を使用します。この位置のことを、LED の座標 といいます。たとえば、上の図に表示されている青い LED の座標は「0, 2」で、赤い LED の座標は「7, 4」になります。これらの座標の最初の数値は、マトリックスを横方向に走る X 軸の値で、次の数値は、マトリックスを縦方向に走る Y 軸の値になります。

ディスプレイにどんな絵を描くか考えるときは、まず方眼紙に手描きしてみるとイメージしやすくなります。あるいは LibreOffice Calc などのスプレッドシートを使用しても良いでしょう。

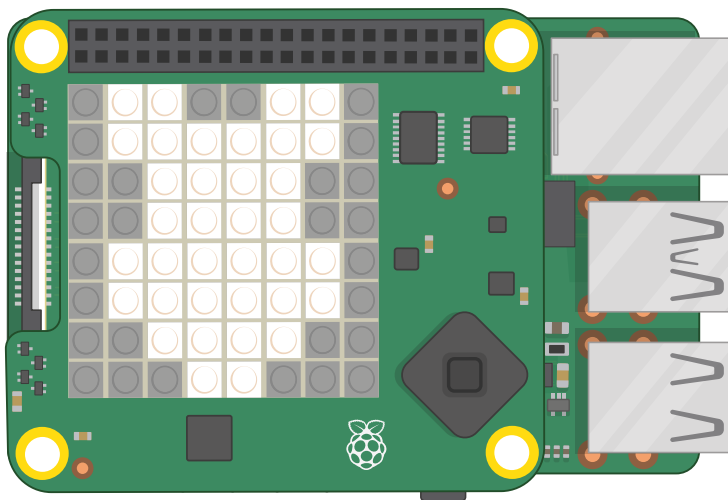
Scratch で絵を描く

これまでに作成したプログラムを保存してから、Scratch で新しいプロジェクト画面を開きます。この章の最初に起動した Scratch 3 が開いたままになっている場合は、Raspberry Pi の Sense HAT 拡張機能がすでに読み込まれているため、特に操作は必要ありません。Scratch 3 を終了してから再起動した場合は、「Add Extension」ボタンをクリックして Sense HAT 拡張機能を読み込む

ください。「**が押されたとき**」イベントブロックをコード領域にドラッグし、このブロックの下に「**set background**」ブロックと「**set colour**」ブロックをドラッグします。背景色が黒、それ以外の部分の色が白になるように、両方のブロックを編集します。黒を指定するには、明るさのスライダーと彩度のスライダーを 0 の位置までスライドし、白を指定するには、明るさのスライダーを 100、彩度のスライダーを 0 の位置までそれぞれスライドします。Scratch では、Sense HAT プログラムで最後に選択した色がそのまま使用されるため、同じプログラムで別の色を使用する場合は、最初にその色を設定する必要があります (別のプログラムで選択した色が適用されることはありません)。次に、プログラムの一番下に「**display ラズベリー**」ブロックをドラッグします。



この状態で緑のフラグアイコンをクリックすると、HAT の LED がラズベリーの形で点灯します (図 7-11)。



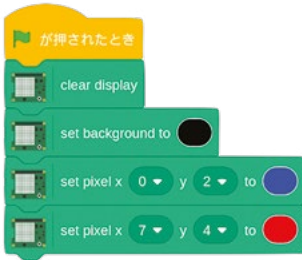
▲ 図 7-11: 白の LED がラズベリーの形で点灯する (LED を直視しないように!)

このラズベリーの形は事前に設定されたものですが、この形を変えることもできます。ブロック内のラズベリーアイコンの横にある下矢印をクリックすると、描画モードがオンになります。パターン上の任意の LED をクリックして、LED のオンとオフを切り替えることができます。下部の 2 つのボタンを使用すると、すべての LED をオンまたはオフにできます。いろいろなパターンを作成して、Sense HAT でどのように表示されるかを試してみましょ。また、「set background to」ブロックと「set colour to」ブロックを使用して、背景の色とパターンの色を変えてみましょう。

次に、**が押されたとき** ブロック以外のブロックをブロックパレットにドラッグして削除し、このブロックの下に「**clear display**」ブロックをドラッグします。この状態で緑のフラグアイコンをクリックすると、すべての LED がオフになります。

絵を描くには、それぞれの LED (ピクセル) で異なる色を設定する必要があります。これを行うには、編集した「**display ラズベリー**」ブロックを「**set colour**」ブロックに接続するか、各ピクセルを個別に設定します。

このセクションの最初で紹介した LED マトリックス (赤い LED と青い LED が 1 つずつ点灯しているマトリックス) を作成するには、先頭の「**clear display**」ブロックを残し、その下に「**set background**」ブロックをドラッグします。次に、「**set background**」ブロックで色を黒に変更し、その下に 2 つの「**set pixel x 0 y 0**」ブロックをドラッグします。最後に、これら 2 つのブロックを以下のように設定します。

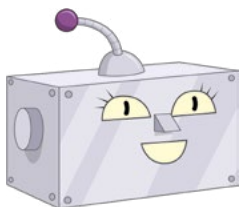


この状態で緑のフラグアイコンをクリックすると、**図 7-10** と同じ配置で LED が点灯します。これで、LED を個別に制御できるようになりました。

次に、「set pixel」ブロックを以下のように設定します。



緑のフラグアイコンをクリックする前に、LED マトリックスで LED がどのように点灯するかを予想して、その予想が正しいかどうかを確認してください。



チャレンジ: 新しいデザイン

新しいデザインを作ってみましょう。最初の方眼紙などを使って、デザインを描いてみましょう。好きな色を使って自由にデザインしてください。



Python で絵を描く

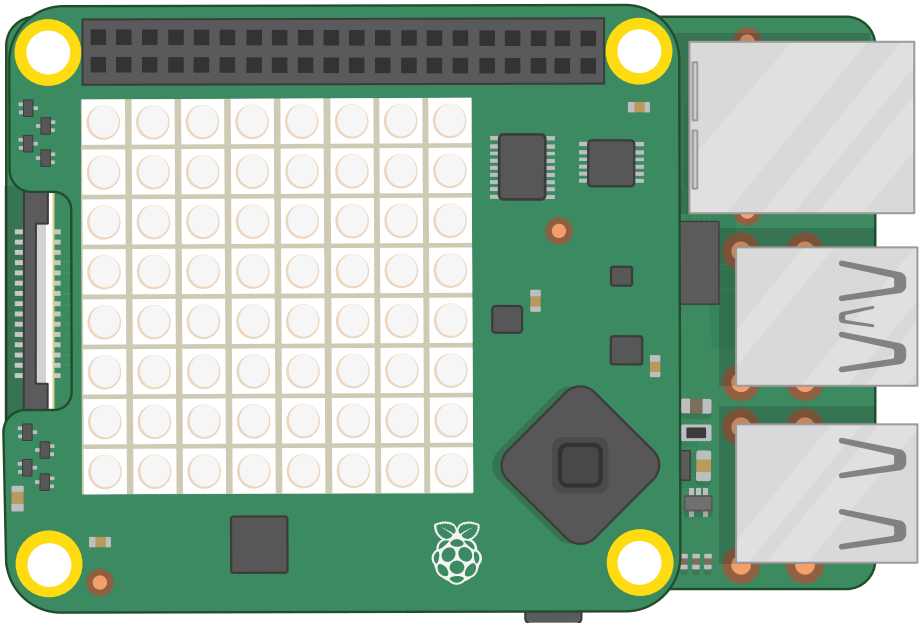
Thonny で新しいプログラムを開いて「Sense HAT Drawing」という名前で保存し、以下のように入力します (エミュレーターを使用する場合は、`sense_hat` の部分を `sense_emu` に変えてください)。

```
from sense_hat import SenseHat
sense = SenseHat()
```

Sense HAT を使用するには、これらのコード行を両方とも入力する必要があることに注意してください。次に、以下のように入力します。

```
sense.clear(255, 255, 255)
```

この状態で「Run」アイコンをクリックすると、すべての LED が明るい白で点灯します (図 7-12)。その際、目を傷める可能性があるため、LED を直視しないでください。

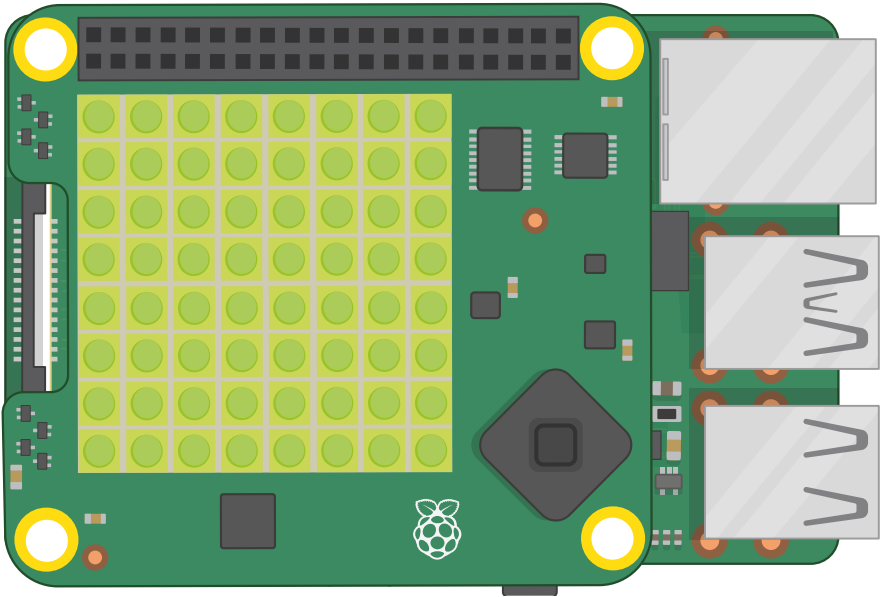


▲ 図 7-12: すべての LED が明るい白で点灯する (LED を直視しないこと)

`sense.clear()` は、以前にプログラミングされた LED の設定をクリアするための関数です。この関数では RGB カラーパラメーターを指定できるため、好きな色で LED を点灯させることができます。たとえば、以下のように入力します。

```
sense.clear(0, 255, 0)
```

この状態で「Run」をクリックすると、すべての LED が明るい緑で点灯します (図 7-13)。このほかにも、いくつかの色を試してみてください。「Hello World」プログラムで作成した色の変数を使用すると、コードが読みやすくなります。

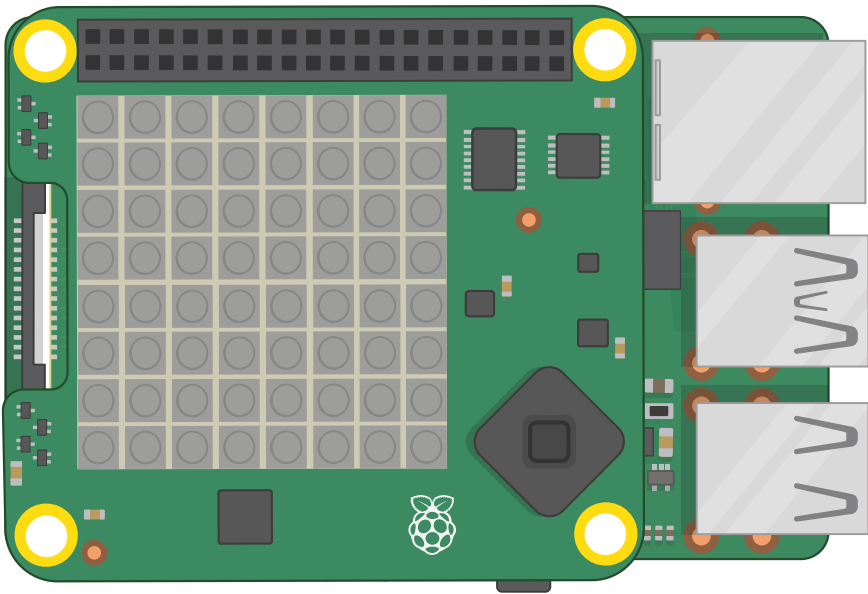


▲ 図 7-13: すべての LED が明るい緑で点灯する

LED をクリアするには、黒の RGB 値を使用する必要があります。黒を指定する場合は、赤、青、緑の値をそれぞれ「0」に設定する必要がありますが、もっと簡単な方法があります。それは、以下のように入力する方法です。

`sense.clear()`

このように入力すると、すべての LED が消灯します (図 7-14)。これは、`sense.clear()` 関数のカッコ内に何も値が指定されていないためです。プログラム内で LED を完全にクリアする必要がある場合は、この関数を使用すると便利です。



▲ 図 7-14: `sense.clear` 関数を使用してすべての LED をオフにする

このセクションの最初で紹介した LED マトリックス (赤い LED と青い LED が 1 つずつ点灯しているマトリックス) を作成するには、`sense.clear()` の下に以下のコード行を追加します。

```
sense.set_pixel(0, 2, (0, 0, 255))
sense.set_pixel(7, 4, (255, 0, 0))
```

最初の 2 つの数値の組み合わせ (0, 2 と 7, 4) は、マトリックス上のピクセルの位置を表しています。左側の数値が X 軸 (横方向) で、右側の数値が Y 軸 (縦方向) になります。その次のかっこで括られた値は、ピクセルの色を表す RGB 値です。この状態で「Run」ボタンをクリックすると、図 7-10 と同じ色と配置で LED が点灯します。

次に、これらのコード行を削除して、以下のように入力します。

```
sense.set_pixel(2, 2, (0, 0, 255))
sense.set_pixel(4, 2, (0, 0, 255))
sense.set_pixel(3, 4, (100, 0, 0))
sense.set_pixel(1, 5, (255, 0, 0))
sense.set_pixel(2, 6, (255, 0, 0))
sense.set_pixel(3, 6, (255, 0, 0))
sense.set_pixel(4, 6, (255, 0, 0))
sense.set_pixel(5, 5, (255, 0, 0))
```

「Run」をクリックする前に、コードで指定された値と LED マトリックスを見比べて、何が表示されるか予想してみましょう。さあ、「Run」をクリックして、予想が正しいかどうかを確認しましょう。

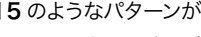
ここまでは、`set_pixel()` 関数を個別に使用して描画する方法を説明しましたが、この方法は時間がかかります。ここでは、複数のピクセルをまとめて変更する方法について説明します。`set_pixel()` 行をすべて削除して、以下のように入力します。

```
g = (0, 255, 0)
b = (0, 0, 0)
```

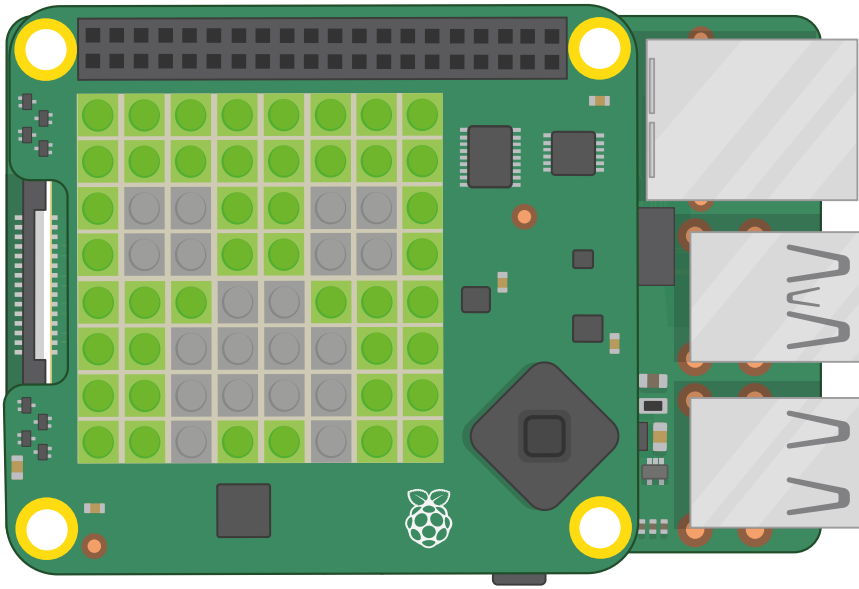
```
creeper_pixels = [
    g, g, g, g, g, g, g, g,
    g, g, g, g, g, g, g, g,
    g, b, b, g, g, b, b, g,
    g, b, b, g, g, b, b, g,
    g, g, g, b, b, g, g, g,
    g, g, b, b, b, b, g, g,
    g, g, b, b, b, b, g, g,
    g, g, b, g, g, b, g, g
]
```

```
sense.set_pixels(creeper_pixels)
```

たくさん書きましたが、「Run」をクリックすると小さなクリーパーが確認できるでしょう。最初の 2 行は、2 つの色（緑と黒）を指定する変数を作成するための行です。コードの読み書きを簡単にするため、緑は「g」、黒は「b」という 1 文字の変数にしています。

その下のコードブロックは、LED マトリックス上の 64 個すべてのピクセルを設定するためのコードブロックです。それぞれの値をコンマ (,) で区切り、全体を角カッコで囲んでいます。このコードブロックでは、数値の組み合わせではなく、プログラムの先頭で定義した「g」と「b」という変数を使用して色を指定しています。これにより、 7-15 のようなパターンが表示されます。

最後に、`sense.set_pixels(creeper_pixels)` の行 `sense.set_pixels()` 関数が変数を受け取ってマトリックス全体を一度に描画します。ピクセルごとに描画しようとするよりもずっと簡単です！



▲ 図 7-15: マトリックス上でパターンを描画する

絵を回転させたり反転させたりすることもできます。たとえば、Sense HAT の取り付け方法に合わせて絵を回転したり、簡単なアニメーションを作成したりすることができます。

ここでは、`creeper_pixels` 変数を編集して、図 7-15 のパターンの「左目」を閉じてみましょう。4つの「b」ピクセルを置き換えるために、3 行目の 2 目と 3 目、続けて 4 行目の 2 目と 3 目をそれぞれ「g」に変更します。

```
creeper_pixels = [
    g, g, g, g, g, g, g, g,
    g, g, g, g, g, g, g, g,
    g, g, g, g, g, b, b, g,
    g, g, g, g, g, b, b, g,
    g, g, g, b, b, g, g, g,
    g, g, b, b, b, b, g, g,
    g, g, b, b, b, b, g, g,
    g, g, b, g, g, b, g, g
]
```

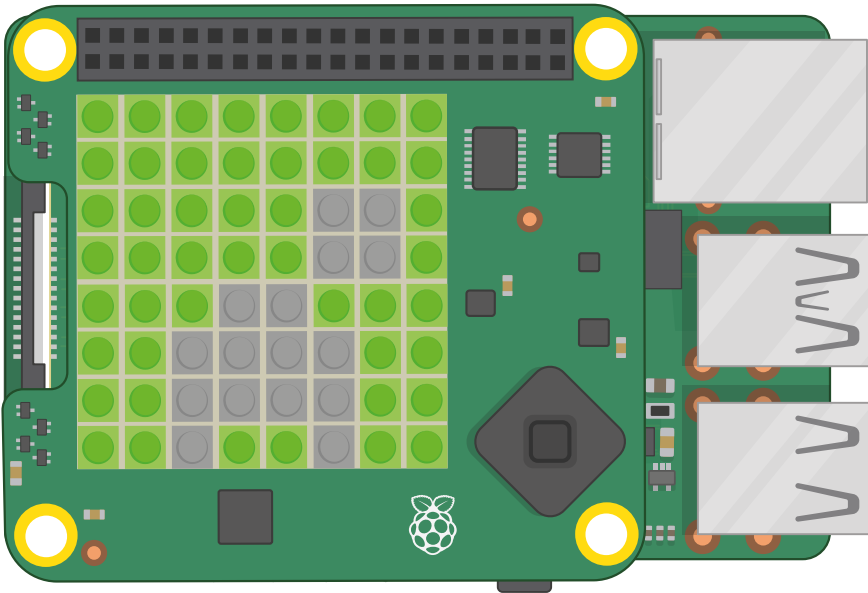
「Run」をクリックすると、クリーパーの左目の部分が閉じた状態になります (図 7-16)。次に、アニメーションを作成してみましょう。プログラムの先頭に以下の行を追加します。

```
from time import sleep
```

次に、プログラムの最後に以下の行を入力します。

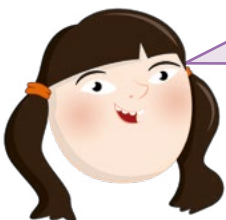
```
while True:  
    sleep(1)  
    sense.flip_h()
```

「Run」をクリックしてみましょう。クリックするたびに、クリーパーの目が開いたり閉じたりします。



▲ 図 7-16: 2 フレームの簡単なアニメーション

`flip_h()` は、絵を横方向に反転させるための関数です。縦方向に反転させたい場合は、`sense.flip_h()` 関数の代わりに `sense.flip_v()` 関数を使用します。また、`sense.set_rotation(90)` を使って、絵を 0、90、180、270 度に回転させることもできます。絵を回転させたい角度に応じて数字を変えます。この関数を使って、クリーパーを点滅させる代わりにいろいろな角度に回転させてみましょう。



チャレンジ: 新しいデザイン

新しいパターンを作ってみましょう。最初の方眼紙などを使って、何を変数にするかを考えながら、パターンを描いてみましょう。好きな色を使って自由にデザインしてください。ヒント: 一度使用した変数は、変更して繰り返し使用することができます。



センサーを使用する

Sense HAT には、いくつかの便利なセンサーが付属しています。これらのセンサーを使用して温度や加速度などのデータを測定し、プログラム内でそのデータを活用することができます。



センサーのエミュレーション

Sense HAT エミュレーターを使用している場合は、慣性測定センサーと環境測定センサーのシミュレーション機能を有効にする必要があります。これを行うには、エミュレーターで「Edit」>「Preferences」をクリックし、シミュレーション機能を選択します。次に、同じメニューの「Orientation Scale」で「180°..360°|0°..180°」を選択し、エミュレーター上の数値が Scratch と Python で表示される数値と一致していることを確認してから「Close」ボタンをクリックします。

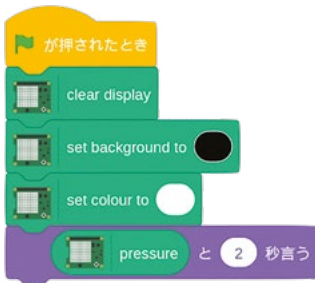
環境測定用センサー

気圧センサー、湿度センサー、温度センサーは、すべて環境測定用センサーです。これらのセンサーにより、Sense HAT の周辺環境のデータを測定することができます。

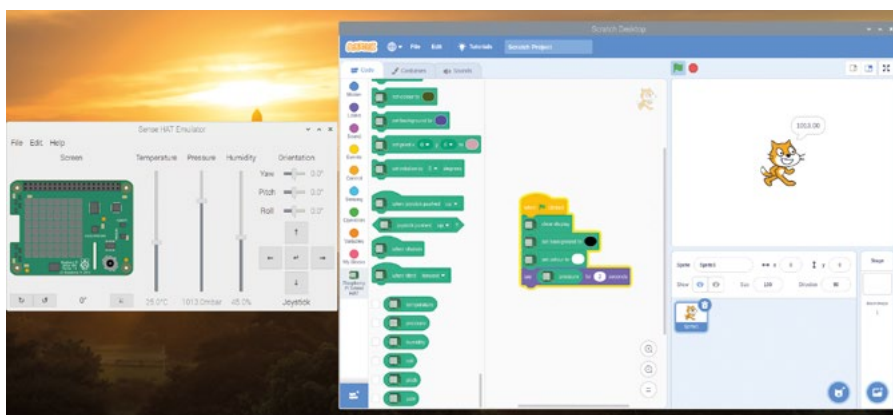
Scratch で環境データを測定する

Scratch で新しいプログラムを開き、「Add Extension」ボタンをクリックして Sense HAT 拡張機能を読み込みます（まだ読み込まれていない場合）。コード領域に「**が押されたとき**」イベントブロックをドラッグし、その下に「**clear display**」ブロックと「**set background to 黒**」ブロックをドラッグします。次に「**set colour to 白**」ブロックをドラッグし、明るさのスライダーと彩度のスライダーを使用して色を設定します。プログラムを作成する場合は、以前のプログラムで設定した色が表示されることがないように、必ず最初に色を設定するようにしてください。

次に、「見た目」カテゴリの「**こんにちは! と 2 秒言う**」ブロックをプログラムの最後に追加し、「Sense HAT」カテゴリの「**pressure**」ブロックを、「**こんにちは! と 2 秒言う**」ブロックの「Hello!」の部分に重なるようにドラッグします。



この状態で緑のフラグアイコンをクリックすると、気圧センサーから読み取られた現在の気圧データがミリバール 単位で表示されます。この気圧データは 2 秒後に消えます。次に、Sense HAT に息を吹きかけてから（エミュレーターを使用している場合は、「Pressure」スライダーを上方向にスライドさせてから）緑のフラグアイコンをクリックすると、前よりも高い値の気圧データが表示されます（図 7-17）。



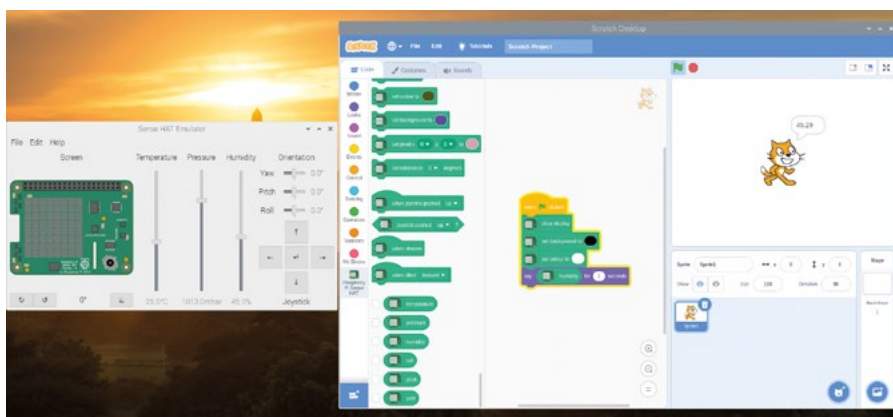
▲ 図 7-17: 気圧センサーの測定データを表示する



値の変更

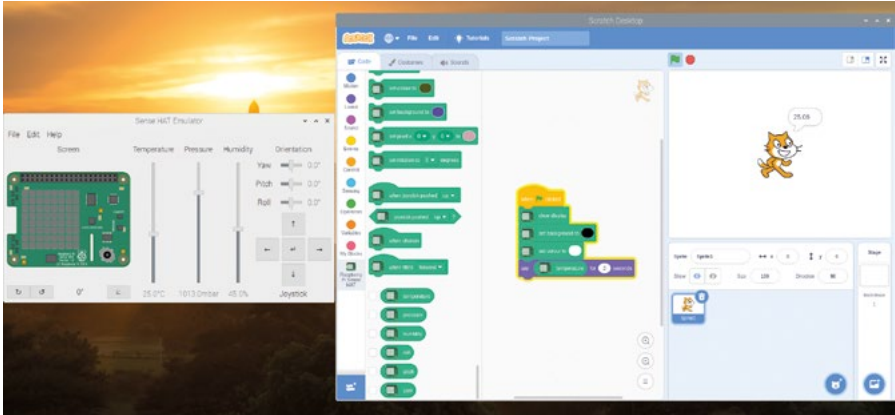
Sense HAT エミュレーターを使用している場合は、スライダーとボタンを使用して、各センサーの値を変更することができます。気圧センサーのスライダーを下にスライドしてから、緑のフラグアイコンをもう一度クリックしてみましょう。

次に、湿度センサーに切り替えてみましょう。「**pressure**」ブロックを削除して、代わりに「**humidity**」ブロックをコード領域にドラッグします。この状態で緑のフラグアイコンをクリックすると、室内の現在の相対湿度が表示されます。ここでも、Sense HAT に息を吹きかけてみましょう（エミュレーターを使用している場合は、「Humidity」スライダーを上方向にスライドしてみましょう）。前よりもずっと高い値の湿度データが表示されるはずです（図 7-18）。



▲ 図 7-18: 湿度センサーの測定データを表示する

温度センサーの場合も、手順は同様です。「humidity」ブロックを削除し、代わりに「temperature」ブロックをコード領域にドラッグします。この状態で緑のフラグアイコンをクリックすると、現在の温度が摂氏で表示されます(図 7-19)。ただし、必ずしも、室内の温度と正確に一致している温度が表示されるわけではありません。Raspberry Pi の稼働中は常に熱が発生するため、Sense HAT 本体と各センサーの温度も高くなり、その結果として、室温よりも高い温度が表示されることがあります。



▲ 図 7-19: 温度センサーの測定データを表示する



チャレンジ: スクロールとループ

それぞれのセンサーの測定データを順に読み取り、そのデータがステージ領域ではなく LED マトリックス上に表示されるようにプログラムを変更してみましょう。また、常に最新の測定データが表示されるように、ループ処理を追加してみましょう。

Python で環境データを測定する

センサーの測定データを取得するには、Thonny で新しいプログラムを開き、そのプログラムを「Sense HAT Sensors」という名前で保存します。次に、Sense HAT を使用するために欠かせない以下の文をスクリプト領域に入力します(エミュレーターを使用する場合は、sense_hat の部分を sense_emu に変えてください)。

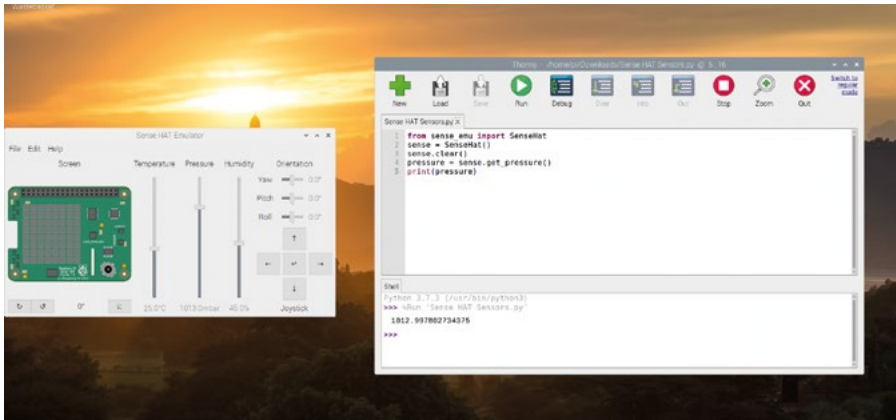
```
from sense_hat import SenseHat
sense = SenseHat()
sense.clear()
```

また、最後に実行したプログラムの内容が表示されたままにならないように、プログラムの先頭には `sense.clear()` を含めておくといいでしょう。

気圧センサーの測定データを取得するには、以下のように入力します。

```
pressure = sense.get_pressure()
print(pressure)
```

この状態で「Run」をクリックすると、Thonny ウィンドウ下部の「Shell」領域に数値が表示されます。この数値は、気圧センサーで測定された気圧（単位はミリバール）を示す数値です（図 7-20）。Sense HAT に息を吹きかけてから（エミュレーターを使用している場合は、「Pressure」スライダーを上方向にスライドしてから）「Run」アイコンをクリックすると、前よりも高い数値が表示されます。



▲ 図 7-20: Sense HAT で測定された気圧データを表示する



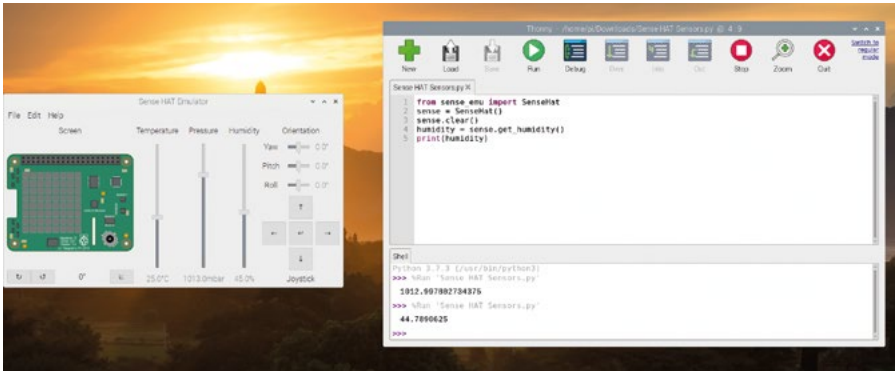
値の変更

Sense HAT エミュレーターを使用している場合は、スライダーとボタンを使用して、各センサーの値を変更することができます。たとえば、気圧センサーのスライダーを下にスライドしてから、「Run」をクリックしてみましょう。

次に、湿度センサーに切り替えてみましょう。最後の 2 行を削除し、以下のコードを入力します。

```
humidity = sense.get_humidity()
print(humidity)
```

この状態で「Run」をクリックすると、室内の現在の相対湿度がパーセント単位で「Shell」領域に表示されます。ここでも、Sense HAT に息を吹きかけてみましょう（エミュレーターを使用している場合は、「Humidity」スライダーを上方向にスライドしてみましょう）。前よりもずっと高い値の湿度データが表示されるはず（図 7-21）。

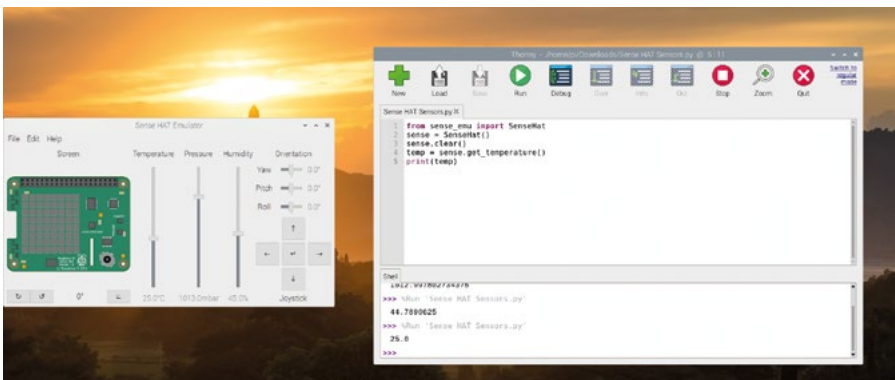


▲ 図 7-21: 湿度センサーの測定データを表示する

温度センサーを使用する場合は、最後の 2 行を削除し、以下のコードを入力します。

```
temp = sense.get_temperature()
print(temp)
```

この状態で「Run」をクリックすると、現在の温度が摂氏で表示されます (図 7-22)。ただし、必ずしも、室内の温度と正確に一致している温度が表示されるわけではありません。Raspberry Pi の稼働中は常に熱が発生するため、Sense HAT 本体と各センサーの温度も高くなり、その結果として、室温よりも高い温度が表示されることがあります。



▲ 図 7-22: 現在の温度を表示する

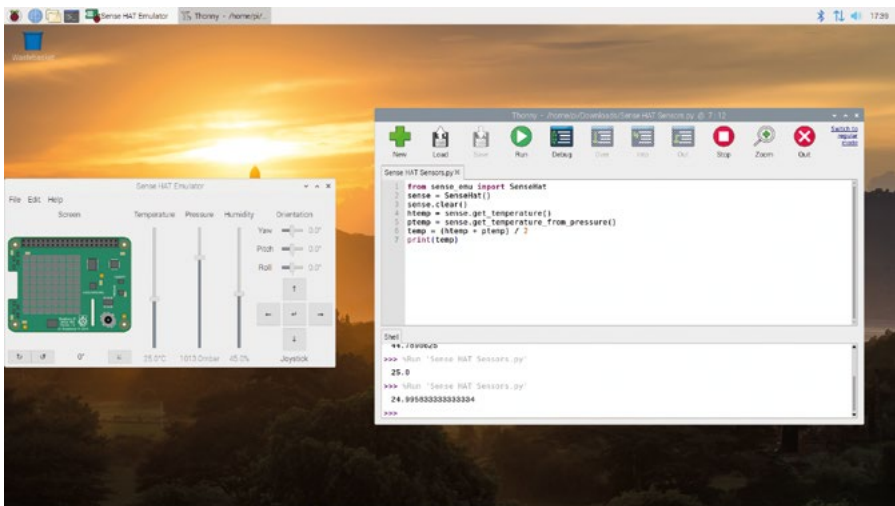
通常は、湿度センサー側に組み込まれている温度センサーの温度が返されますが、もし気圧センサー側に組み込まれた温度センサーの温度を取得したい場合は、`sense.get_temperature_from_pressure()` というコードを使用します。2 つの温度センサーから読み取って平均値を出すこともできるため、片方の温度センサーの値だけを使用する場合と比べて、より正確な値にできる可能性があります。最後の 2 行を削除し、以下のコードを入力しましょう。


```

htemp = sense.get_temperature()
ptemp = sense.get_temperature_from_pressure()
temp = (htemp + ptemp) / 2
print(temp)

```

この状態で「Run」アイコンをクリックすると、「Shell」パネルに数値が表示されます (図 7-23)。これが、両方の温度センサーから読み取った測定値を足して 2 で割った平均値です。エミュレーターを使用している場合は、湿度センサーに組み込まれている温度センサーの測定値、気圧センサーに組み込まれている温度センサーの測定値、両方の温度センサーの平均値が、すべて同じ値になります。



▲ 図 7-23: 両方の温度センサーの平均値を表示する



チャレンジ: スクロールとループ

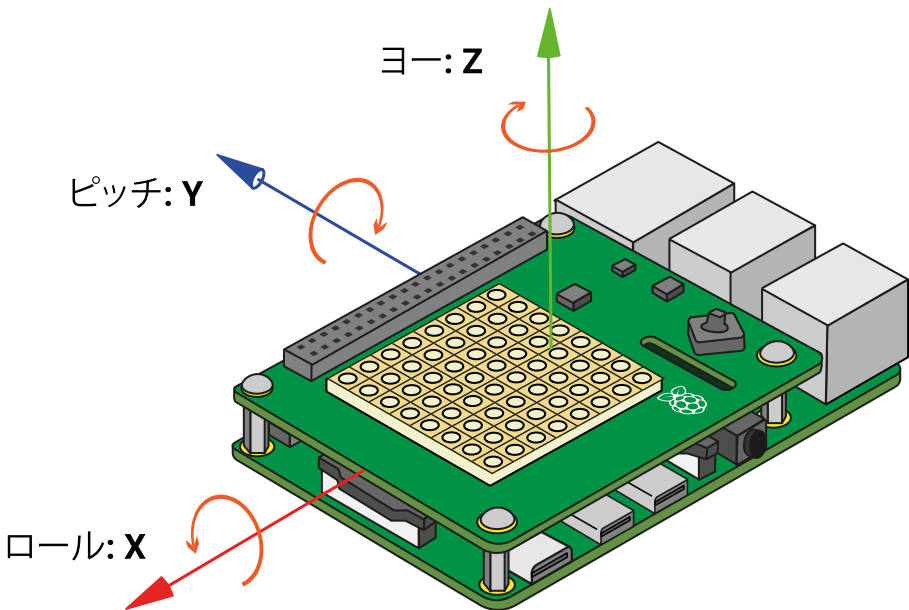


それぞれのセンサーの測定データを順に読み取り、そのデータが「Shell」領域ではなく LED マトリックス上に表示されるようにプログラムを変更してみましょう。また、常に最新の測定データが表示されるように、ループ処理を追加してみましょう。

慣性データの測定

Sense HAT のジャイロスコープセンサー、加速度計、磁力計は、それぞれが連携して慣性測定装置 (IMU) として機能します。これらのセンサーでは、温度センサーなどの環境測定用センサーと同様に、周辺環境に関するデータが測定されます。たとえば磁力計の場合、磁場の強さに関するデータが測定されます。これらのセンサーは通常、Sense HAT 本体の動作に関するデータを測定する目的で使用されます。複数のセンサーを組み合わせたものが IMU になります。各センサーから個別に取得した測定データを使用できるプログラム言語もあれば、各センサーの測定値を組み合わせただけで使用できないプログラム言語もあります。こうしたプログラム言語の場合に、IMU を使用する必要があります。

IMU について理解する前に、空間軸の仕組みについて理解する必要があります。Sense HAT と Raspberry Pi は、X 軸 (左右方向の軸)、Y 軸 (前後方向の軸)、Z 軸 (上下方向の軸) という 3 つの空間軸に沿って移動することができます (図 7-24)。また、これら 3 つの軸を基準として Sense HAT と Raspberry Pi を回転させることもできます。その場合、空間軸の名前が変わります。X 軸を基準として回転させる場合は「ロール」、Y 軸を基準として回転させる場合は「ピッチ」、Z 軸を基準として回転させる場合は「ヨー」という名前になります。Y 軸に沿って Sense HAT を回転させる場合はピッチを調整し、X 軸に沿って回転させる場合はロールを調整し、Z 軸に沿って回転させる場合はヨーを調整することになります。ピッチ、ロール、ヨーは、飛行機の動きを考えると理解しやすくなります。飛行機が離陸する場合は機首を上げ、着陸する場合は機首を下げますが、この動作が「ピッチ」に該当します。飛行機が旋回する場合は機体を左右に傾斜させますが、この動作が「ロール」に該当します。水平な状態を保ったまま飛行機の進行方向を変える場合はラダー (方向舵) を操作しますが (自動車のステアリングを操作する場合に似ています)、この動作が「ヨー」に該当します。



▲ 図 7-24: Sense HAT の IMU の空間軸

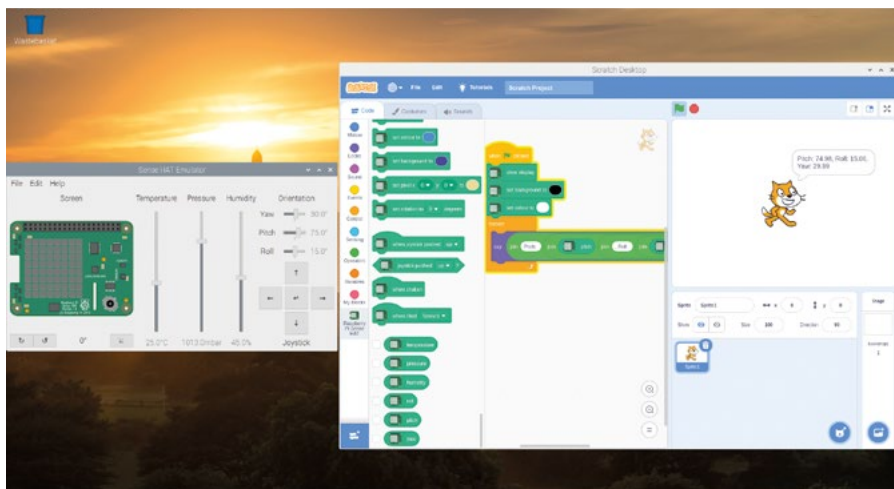
Scratch で慣性データを測定する

Scratch で新しいプログラムを開き、「Add Extension」ボタンをクリックして Sense HAT 拡張機能を読み込みます (まだ読み込まれていない場合)。コード領域に「**緑の旗が押されたとき**」イベントブロックをドラッグし、その下に「**clear display**」ブロックをドラッグします。次に、「**set background to 黒**」ブロックと「**set colour to 白**」をドラッグして、色を設定します。

次に、「**ずっと**」ブロックを一番下にドラッグし、このブロック内に「**こんにちは! と言う**」ブロックを配置します。IMU の 3 つの軸 (ピッチ、ロール、ヨー) で測定データを表示するには、「**と**」演算子ブロックとそれに対応する Sense HAT ブロックを追加する必要があります。画面上での表示を見やすくするため、以下のようにコンマとスペースを指定してください。



緑のフラグアイコンをクリックし、Sense HAT と Raspberry Pi をゆっくりと動かしてみましょう (ケーブルが外れないように注意してください)。X 軸、Y 軸、Z 軸に沿って Sense HAT を傾けると、その傾斜に従って、ピッチ、ロール、ヨーの値が表示されます (図 7-25)。



▲ 図 7-25: ピッチ、ロール、ヨーの値を表示する

Python で慣性データを測定する

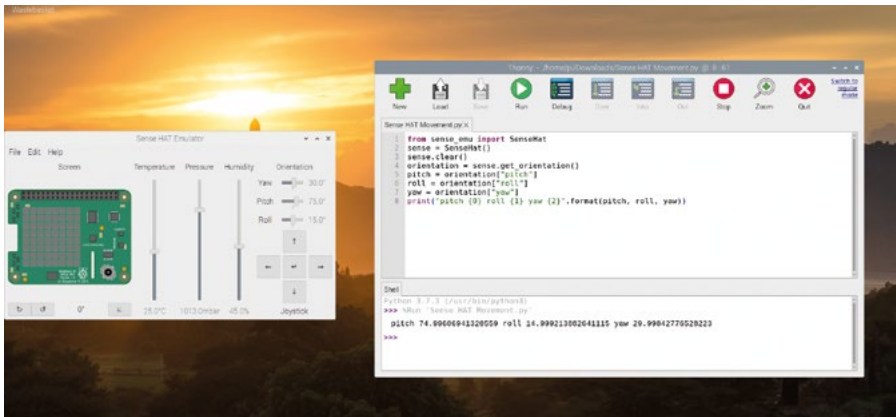
Thonny で新しいプログラムを開き、そのプログラムを「Sense HAT Movement」という名前で保存します。これまでと同じように、以下のコード行を入力します（エミュレーターを使用する場合は、sense_hat の部分を sense_emu に変えてください）。

```
from sense_hat import SenseHat
sense = SenseHat()
sense.clear()
```

IMU の測定データを使用し、3 つの軸に基づいて Sense HAT の現在の方向を判断するには、以下のように入力します。

```
orientation = sense.get_orientation()
pitch = orientation["pitch"]
roll = orientation["roll"]
yaw = orientation["yaw"]
print("pitch {0} roll {1} yaw {2}".format(pitch, roll, yaw))
```

この状態で「Run」をクリックすると、Sense HAT の方向が 3 つの軸に分かれて表示されます（図 7-26）。Sense HAT の傾きを変えて「Run」をクリックすると、その傾きを示す新しい数値が表示されます。



▲ 図 7-26: Sense HAT のピッチ、ロール、ヨーの値を表示する

IMU では、Sense HAT の方向を測定するだけでなく、Sense HAT の動作を測定することもできます。動作に関する正確なデータを取得するには、ループ処理で IMU の測定データを頻繁に読み取る必要があります。方向に関するデータの場合は、測定データを 1 回取得するだけで済みますが、動作に関するデータの場合は、その動作に合わせてデータを繰り返し取得する必要があります。sense.clear() 関数の下にあるコードをすべて削除し、以下のコードを入力します。

while True:

```
acceleration = sense.get_accelerometer_raw()
x = acceleration["x"]
y = acceleration["y"]
z = acceleration["z"]
```

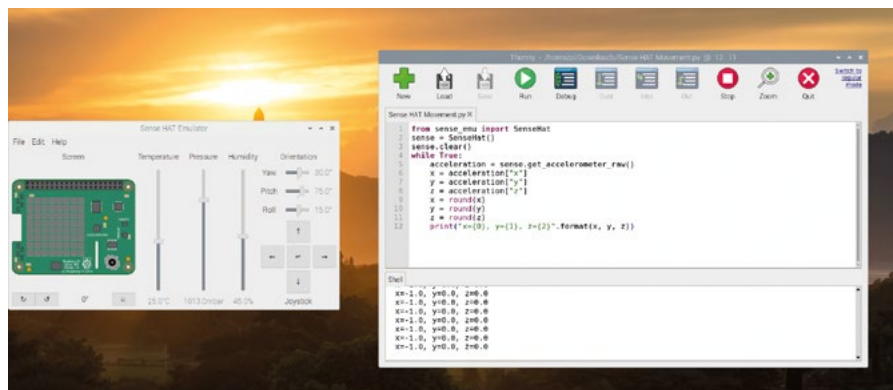
このコードにより、加速度計の測定データを 3 つの空間軸に分けて格納するための変数が作成されます (X 軸、Y 軸、Z 軸)。加速度計の測定データはそのままでは読みにくいため、以下のように入力します。これにより、測定値が最も近い整数に四捨五入されるので、データが読みやすくなります。

```
x = round(x)
y = round(y)
z = round(z)
```

最後に、以下のように入力して 3 つの値を表示します。

```
print("x={0}, y={1}, z={2}".format(x, y, z))
```

この状態で「Run」をクリックすると、加速度計の測定データが「Shell」領域に表示されます (図 7-27)。このプログラムの場合、前に作成したプログラムとは異なり、測定値が連続して表示されます。この表示を停止するには、赤い停止アイコンをクリックします。



▲ 図 7-27: 加速度計の測定値が、最も近い整数に四捨五入して表示される

Sense HAT が静止しているにもかかわらず、3 つの軸のいずれかの加速度値 (Raspberry Pi が机の上で水平に置かれている場合は Z 軸の加速度) が 1G として表示されることに注意してください (1G の「G」は重力という意味です)。この値が表示されるのは、加速度計が地球の重力 (地球の中心に向かって物体が引き寄せられる力) を検知しているためです。机の上に乗っているものを机からどかすと床に向かって落ちていきますが、これが地球の重力です。

プログラムの実行中に、Sense HAT が接続された Raspberry Pi を持ち上げて、ゆっくりと回

転させてみてください (ケーブルが外れないように注意してください)。Raspberry Pi のネットワークポートと USB ポートを下に向けると、Z 軸の値が 0G、X 軸の値が 1G になります。HDMI ポートと電源ポートを下に向けると、Y 軸の値が 1G になります。HDMI ポートを上に向けると、Y 軸の値が -1G になります。

地球の重力は約 1G ですが、この値と 3 つの空間軸の測定値を組み合わせることで、Raspberry Pi がどの方向を向いているのかを判断することができます。また、Raspberry Pi の動きを検知することもできます。Sense HAT が接続された Raspberry Pi をゆっくりと振動させて、加速度計の測定値を確認してみましょう。強く振動させるほど、加速度計の測定値が高くなるのがわかります。

`sense.get_accelerometer_raw()` の引数を使用すると、IMU を構成する加速度計以外の 2 つのセンサー (ジャイロスコープセンサーと磁力計) がオフになり、加速度計で測定されたデータだけを表示することができます。同様に、ジャイロスコープセンサーの測定データだけを表示することも、磁力計の測定データだけを表示することもできます。

`acceleration = sense.get_accelerometer_raw()` という行を以下の通り変更しましょう。

```
orientation = sense.get_gyroscope_raw()
```

この行の下にある 3 行すべてで、`acceleration` の部分を `orientation` に変更します。この状態で「Run」をクリックすると、3 つの軸のすべてについて、Sense HAT の方向を示す値が表示されます (これらの値は、最も近い整数に四捨五入された状態で表示されます)。前のプログラムでは、IMU を構成する 3 つのセンサーの測定データが表示されていましたが、このプログラムでは、ジャイロスコープセンサーの測定データだけが表示されます。たとえば、移動するロボットの背面に Sense HAT を取り付けてその方向を測定する場合や、強い磁場の近くで Sense HAT を使用する場合などは、ジャイロスコープセンサーの測定データだけを表示すると、ロボットの動作や磁場に関する測定データを考慮する必要がなくなるため、混乱することなくデータを読むことができ便利です。

赤い停止アイコンをクリックしてプログラムを停止してください。次に、磁力計を使用してみましょう。最初の 4 行を残してすべてのコードを削除し、`while True` 行の下に以下のコードを追加します。

```
north = sense.get_compass()  
print(north)
```

この状態でプログラムを実行すると、磁北の方向を示す値が「Shell」領域に連続して表示されます。Sense HAT をゆっくりと回転させると、磁北の方向を基準として、Sense HAT の方向を示す値が変化します。これは、コンパスと同じ動作です。磁石がある場合は (冷蔵庫に貼るマグネットでもかまいません)、Sense HAT の周囲で磁石を動かして、磁力計の測定値がどのように変化するかを確認してください。



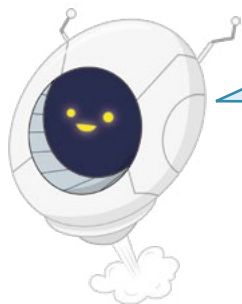
チャレンジ: 自動回転

LED マトリックスと慣性測定センサーについて学習した知識を活用して、Sense HAT の方向に応じてパターンを自動的に回転するプログラムを作ってみましょう。



ジョイスティックコントロール

Sense HAT の右下隅にあるジョイスティックは小さいですが、非常に強力な機能を持っています。前後左右という 4 方向からの入力データだけでなく、上からの入力データ (押しボタンスイッチを押した場合など) もあるため、全部で 5 方向の入力データを認識できます。




注意

Sense HAT のジョイスティックを使用する場合は、この章の最初で説明したように、必ずスペーサーを取り付けてください。スペーサーを取り付けずにジョイスティックを操作すると、Sense HAT ボードが曲がるため、Sense HAT だけでなく Raspberry Pi の GPIO ヘッダーも損傷する可能性があります。



Scratch のジョイスティックコントロール

Raspberry Pi の Sense HAT 拡張機能が読み込まれた状態で、Scratch で新しいプログラムを開きます。スクリプト領域に「が押されたとき」イベントブロックをドラッグし、その下に「**clear display**」ブロックをドラッグします。次に、「**set background to 黒**」ブロックと「**set colour to 白**」ブロックをドラッグして、色を設定します。

Scratch では、Sense HAT のジョイスティックはキーボードのカーソルキーにマップされます。ジョイスティックを上を押すと、Scratch はキーボードで上矢印キーが押されたと認識し、ジョイスティックを下を押すと、キーボードで下矢印キーが押されたと認識します。同様に、ジョイスティックを左に押した場合は左矢印キー、右に押した場合は右矢印キーの動作として認識されます。また、押しボタンスイッチのようにジョイスティックを上から押すと、Scratch はキーボードで **ENTER** キーが押されたと認識します。



注意

物理的な Sense HAT ハードウェアでのみ、ジョイスティックコントロールを使用することができます。Sense HAT エミュレーターを使用している場合は、ジョイスティックの代わりに、キーボード上の対応するキーを使用する必要があります。



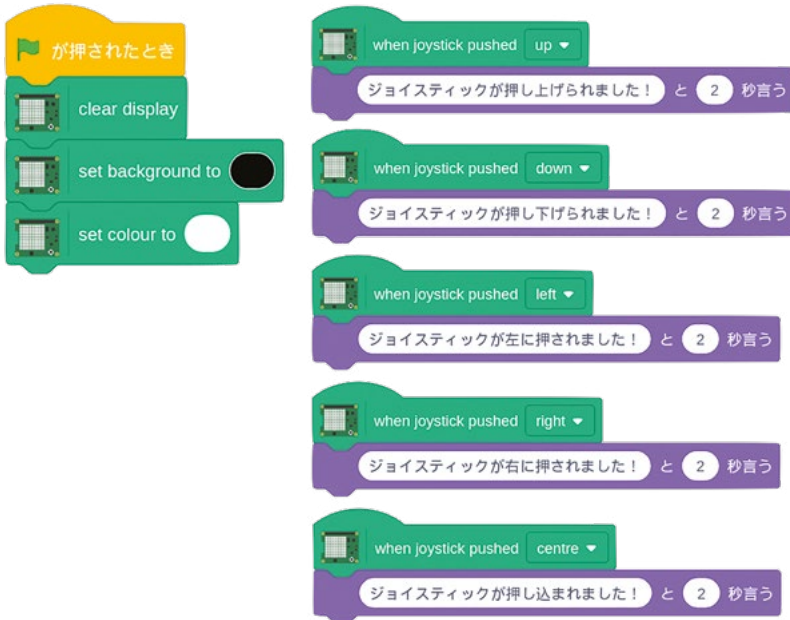
「when joystick pushed up」ブロックをコード領域にドラッグし、その下に「こんにちは! と 2 秒言う」ブロックをドラッグします。



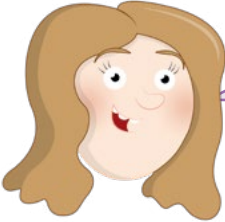
この状態でジョイスティックを上を押すと、ステージ上の猫が「Hello!」と言います。

次に、「こんにちは! と 2 秒言う」ブロックの内容を「

ジョイスティックが上に押されました! と 2 秒言う」に変更します。同様に、「イベント」カテゴリーのブロックと「見た目」カテゴリーのブロックを組み合わせ、ジョイスティックの 5 方向 (前後左右と上からのプッシュ) の動作を認識するためのプログラムを作成します。



ジョイスティックを自由に操作して、設定したメッセージが表示されるかどうかを確認してみましょう。



最後のチャレンジ:

Sense HAT のジョイスティックを使用して、Scratch のステージ領域でスプライトを制御してみましょう。たとえば、別のスプライトがステージ領域に登場するたびに、LED で楽しいメッセージを表示してみましょう。



Python のジョイスティックコントロール

Thonny で新しいプログラムを開き、そのプログラムを「Sense HAT Joystick」という名前で保存します。これまでと同じように、Sense HAT のセットアップと LED マトリックスのクリアを行うための 3 行のコードを最初に入力します。

```
from sense_hat import SenseHat
sense = SenseHat()
sense.clear()
```

次に、無限ループを追加します。

```
while True:
```

そして、Sense HAT のジョイスティックからの入力データを待機するために以下のコードを入力します。Thonny が行の先頭にインデントを自動で挿入してくれるでしょう。

```
    for event in sense.stick.get_events():
```

最後に、以下の行を入力します（この行の先頭にも、自動的にインデントが挿入されます）。これは、ジョイスティックが操作された場合に処理を実行するためのコードです。

```
        print(event.direction, event.action)
```

「Run」をクリックし、ジョイスティックを自由に操作してみましょう。ジョイスティックを押した方向（前後左右と上からのプッシュ）が、「Shell」領域に表示されます。

ジョイスティックをいずれかの方向に 1 回押すと **pressed** というイベントが発生し、ジョイスティックを元の位置に戻すと **released** というイベントが発生します。これらのイベントをプログラム内で使用することができます。たとえば、ゲーム内でキャラクターを登場させる場合に、いずれかの方向にジョイスティックを押したときに、その方向にキャラクターを移動させ、ジョイスティックを元の位置に戻したときにキャラクターの移動を停止する、などの動作をプログラミングすることができます。

また、「for ループ」文を使用するだけでなく、ジョイスティックを使用して関数を呼び出すこともできます。`sense.clear()` の下にあるコードをすべて削除し、以下のように入力します。

```
def red():
    sense.clear(255, 0, 0)

def blue():
    sense.clear(0, 0, 255)

def green():
    sense.clear(0, 255, 0)

def yellow():
    sense.clear(255, 255, 0)
```

これらの関数は、Sense HAT の LED マトリックス全体を単色 (赤、青、緑、黄) に変更するための関数です。これらの関数を使用すると、非常に簡単にプログラムを機能させることができます。これらの関数を実際に呼び出すには、ジョイスティックの操作に合わせてどの関数を呼び出すのかをプログラム内で指定する必要があります。ここでは、以下のように入力します。

```
sense.stick.direction_up = red
sense.stick.direction_down = blue
sense.stick.direction_left = green
sense.stick.direction_right = yellow
sense.stick.direction_middle = sense.clear
```

最後に、ジョイスティックからの入力データを監視するために、無限ループ (メインループ といいます) を追加する必要があります。ループ処理を追加しないと、プログラムが 1 回実行されただけで終了してしまいます。以下の 2 行を入力します。

```
while True:
    pass
```

「Run」をクリックして、ジョイスティックを自由に操作してみましょう。LED の色が次々に変わっていくのがわかります。LED をオフにするには、ジョイスティックを押しボタンのように上から押します。上のコード行の最後の行 (`middle` が記述されている行) により、すべての LED をオフにするための `sense.clear()` 関数が呼び出されます。これで、ジョイスティックからの入力データをキャプチャできるようになりました。



最後のチャレンジ:

画面にパターンを表示する方法について学習した知識を活用して、ジョイスティックを押した方向に画像を回転させるプログラムを作ってみましょう。ジョイスティックを上から押したときに他の画像に切り替えるようなプログラムを作ってみましょう。



Scratch プロジェクト: Sense HAT で線香花火を作る

このプロジェクトでは、これまでに学習した Sense HAT の知識をすべて活用して、室温に応じて点灯速度が変化する電気線香花火を作ってみましょう。室温が低くなるほど点灯速度が上がり、室温が高くなるほど点灯速度が下がる線香花火です。

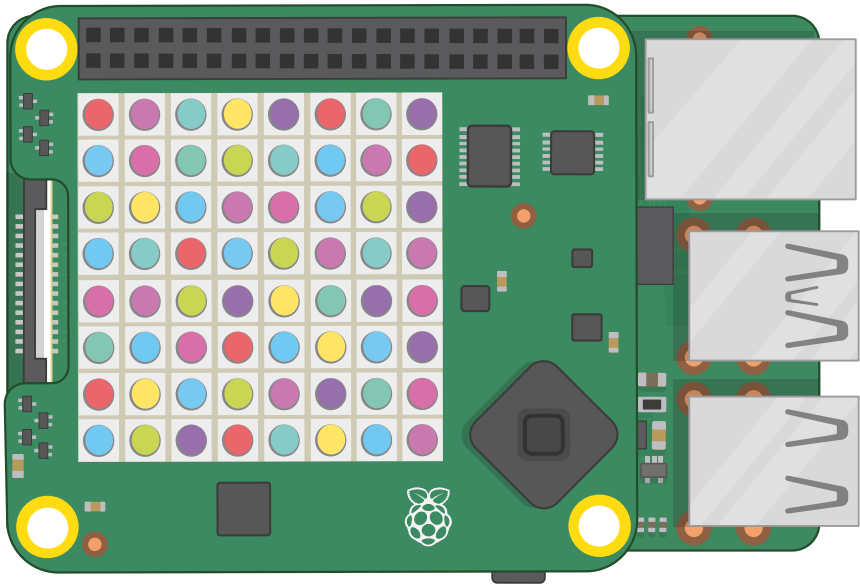
Raspberry Pi の Sense HAT 拡張機能が読み込まれた状態で、新しい Scratch プロジェクトを開きます。これまでと同じように、コード領域に「**旗が押されたとき**」ブロックをドラッグし、その下に「**clear display**」ブロックをドラッグします。次に、「**set background to 黒**」ブロックと「**set colour to 白**」ブロックをドラッグして、色を設定します。

最初に、簡単な線香花火を作成してみましょう。コード領域に「**ずっと**」ブロックをドラッグし、このブロック内に「**set pixel x 0 y 0 to 色**」ブロックを配置します。ここでは、事前に設定されている数値をそのまま使用するのではなく、「**1 から 10 までの乱数**」演算ブロックを使用して X 座標、Y 座標、色セクションをそれぞれ設定します。

「乱数」ブロックの「1」と「10」という数値は変更する必要があります。「**set pixel**」ブロックの最初の 2 つは、LED マトリックスのピクセルの X 座標と Y 座標に対応する数値です。そのため、「1」を「0」に、「10」を「7」に変更します。変更後のブロックは、「**0 から 7 までの乱数**」のようになります。次に、各ピクセルの色も同様に変更します。カラーセクターで色を選択すると、その色がスクリプト領域に直接表示されますが、内部的には色は数値で表され、直接数値を指定することもできます。それでは、最後の「乱数」ブロックを「0 から 16777215 までの乱数」のように編集します。最後の「乱数」ブロックを「**0 から 16777215 までの乱数**」のように編集します。



この状態で緑のフラグアイコンをクリックすると、Sense HAT の LED がランダムなカラーパターンで点灯します (図 7-28)。これで、電気線香花火が完成しました。



▲ 図 7-28: ランダムなカラーパターンで点灯する LED

次に、室温に応じて花火の点灯速度を変えてみましょう。最初に、「set pixel」ブロック内の「ずっと」ブロックの下に「1秒待つ」ブロックをドラッグします。「wait 1 seconds」の「1」の部分に重なるように「●/●」除算演算子ブロックをドラッグし、分母の部分に「10」を入力します。最後に、除算演算子ブロックの分子の部分に重なるように「temperature」ブロックをドラッグします。



この状態で緑のフラグアイコンをクリックすると、室温が非常に低くないかぎり、花火の点灯速度が前よりもずっと遅くなります。これは、花火の点灯条件として温度を追加したためです。このプログラムでは、現在の室温を 10 で割った値が、LED が次に点灯するまでの秒数になります。そのため、室温が 20°C の場合は 2 秒後に LED が点灯し、室温が 10°C の場合は 1 秒後に LED が点灯します。室温が 10°C 未満の場合は、1 秒経過しないうちに LED が点灯することになります。

室温が 0°C 未満の場合 (水が凍結する温度)、プログラムは 0 秒未満に LED を点灯しようとしても、これはタイムマシンでも発明しないかぎり物理的に不可能なことなので、0 秒を設定した場合と同じ動作になります。これで、Sense HAT の各種の機能をプログラムで活用できるようになりました。

Python プロジェクト: Sense HAT でトライコーダーを作る

このプロジェクトでは、これまでに学習した Sense HAT の知識をすべて活用して、SF ファンにとってはおなじみのトライコーダーを作ってみましょう (トライコーダーとは、さまざまなセンサーが組み込まれた架空の携帯分析装置のことです)。

Thonny で新しいプロジェクトを開いて「Tricorder」という名前で作成し、これまでと同じように以下のコードを入力します。

```
from sense_hat import SenseHat
sense = SenseHat()
sense.clear()
```

次に、Sense HAT の各センサー用の関数を定義します。最初に、慣性測定装置 (IMU) 用の関数を入力します。

```
def orientation():
    orientation = sense.get_orientation()
    pitch = orientation["pitch"]
    roll = orientation["roll"]
    yaw = orientation["yaw"]
```

ここでは、IMU の測定データを LED で 1 文字ずつスクロール表示するため、何桁もある小数点がスクロール表示されないように、小数点以下の部分を四捨五入します。ただし、整数に四捨五入するのではなく、小数点以下 1 桁の数値に四捨五入します。

```
pitch = round(pitch, 1)
roll = round(roll, 1)
yaw = round(yaw, 1)
```

最後に、トライコーダーがハンドヘルドデバイス (モニターやテレビに接続する必要がない装置のこと) として機能するように、IMU の測定データが LED でスクロール表示されるように設定する必要があります。

```
sense.show_message("Pitch {0}, Roll {1}, Yaw {2}".
format(pitch, roll, yaw))
```

これで、IMU で測定された方向データを読み取って表示するための関数が完成しました。他のセンサーについても、同じような関数を作成する必要があります。最初に、温度センサー用の関数を作成します。

```
def temperature():
    temp = sense.get_temperature()
    temp = round(temp, 1)
    sense.show_message("Temperature: %s degrees Celsius" % temp)
```

最後の温度センサーの測定データを LED に出力する「show_message」の行に注目してください。「%s」という文字は「プレースホルダー」と呼ばれるもので、プレースホルダーの部分は `temp` 変数の値に置き換わります。プレースホルダーの前後に「Temperature:」や「degrees Celsius」というテキストラベルをそれぞれ指定すると、LED に表示される測定データがわかりやすくなります。

次に、湿度センサー用の関数を定義します。

```
def humidity():
    humidity = sense.get_humidity()
    humidity = round(humidity, 1)
    sense.show_message("Humidity: %s percent" % humidity)
```

次に、気圧センサー用の関数を定義します。

```
def pressure():
    pressure = sense.get_pressure()
    pressure = round(pressure, 1)
    sense.show_message("Pressure: %s millibars" % pressure)
```

最後に、磁力計用の関数を定義します。

```
def compass():
    for i in range(0, 10):
        north = sense.get_compass()
        north = round(north, 1)
        sense.show_message("North: %s degrees" % north)
```

この関数の `for` ループ文により、磁力計の測定データが 10 回読み込まれます。データを繰り返し読み取ることにより、データの精度が高くなります。表示される測定データにばらつきがある場合は、データの読み取り回数を 20 回、30 回、100 回と増やしていくと、データの精度が上がります。

これで、Sense HAT の各センサーから測定データを読み取り、そのデータを LED でスクロール表示するための 5 つの関数が完成しました。ただし、どのセンサーの測定データを表示するのかを決定する方法が必要です。ここでは、その方法としてジョイスティックを使ってみましょう。

以下のように入力します。

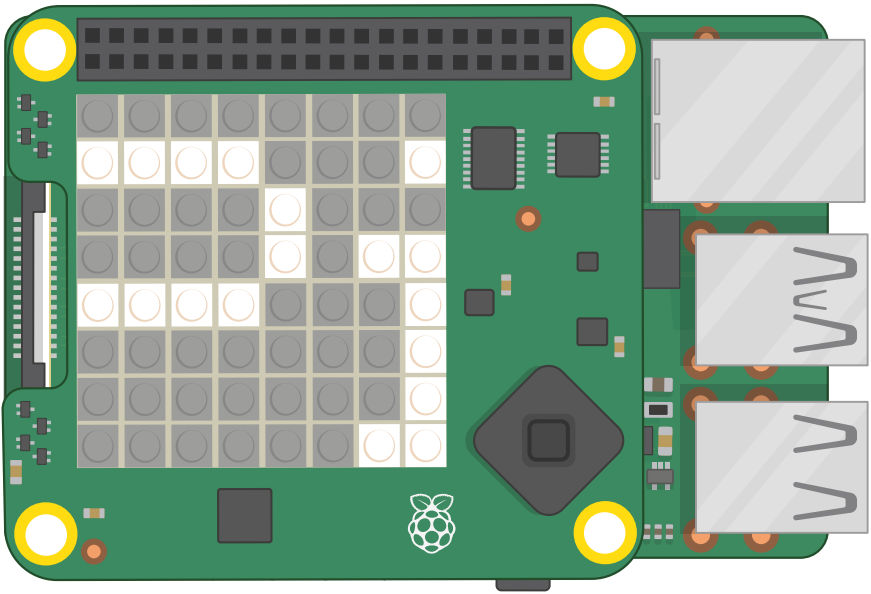

```
sense.stick.direction_up = orientation
sense.stick.direction_right = temperature
sense.stick.direction_down = compass
sense.stick.direction_left = humidity
sense.stick.direction_middle = pressure
```

以上のコードによってジョイスティックの 5 つの方向に各センサーが割り当てられます。ジョイスティックを上を押すと、方向センサーの測定データが読み取られ、下を押すと、磁力計の測定データが読み取られます。同様に、ジョイスティックを左に押した場合は湿度センサー、右に押した場合は温度センサー、上から押し込んだ場合は気圧センサーの測定データがそれぞれ読み取られます。

最後に、ジョイスティックからの入力データを待機するためのメインループを作成する必要があります。プログラムの最後に、以下のコードを追加します。

```
while True:
    pass
```

「Run」をクリックして、ジョイスティックをいずれかの方向に押ししてみてください。その方向に割り当てられているセンサーの測定データが LED にスクロール表示されます (図 7-29)。測定データのスクロール表示が終了したら、ジョイスティックを別の方向に押し違うセンサーの測定データを表示してみましょう。これで、ハンドヘルドタイプのトライコーダーが完成しました。



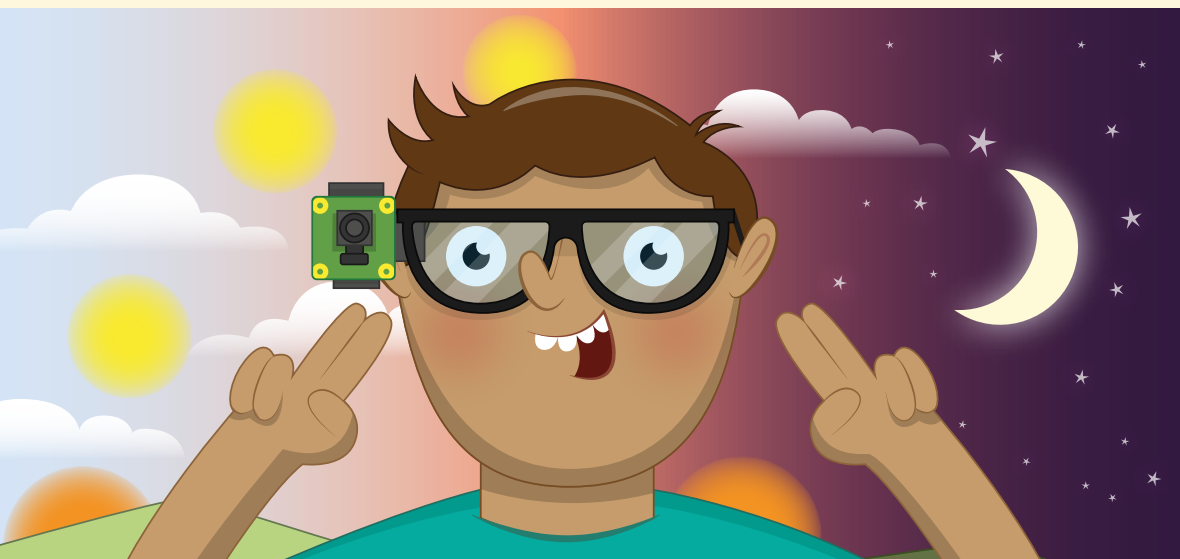
▲ 図 7-29: 各センサーの測定データが LED にスクロール表示される

「付録 D: 参考資料」では、いくつかの Sense HAT プロジェクトを紹介しています。そちらも確認してください。

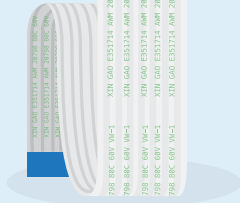
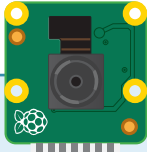
第 8 章

Raspberry Pi の Camera Module を 使ってみよう

High Quality Camera というカメラモジュールを Raspberry Pi に接続すると、解像度の高い写真やビデオを撮影して、すばらしいコンピュータービジョンプロジェクトを作成することができます



目に見えるプロジェクト (ロボット工学の分野ではコンピュータービジョンプロジェクトとい
います) を作成したいと考えたことがあるならば、Raspberry Pi のオプションカメラモジ
ュールや新しい High Quality Camera (HQ カメラ) を使ってプロジェクトを作成して
みましょう。細いリボンケーブルが付属している小さな正方形の回路基板であるカメラモジュールと
HQ カメラは、Raspberry Pi のカメラシリアルインターフェイス (CSI) ポートに接続して使用します
(Raspberry Pi 400 では、カメラモジュールは使用できません)。このカメラモジュールで、高解像度
の静止画像信号と動画信号を生成することができます。これらの信号は、そのまま使用することも、独
自のプログラムに取り込んで使用することもできます。



カメラの種類

Raspberry Pi のカメラには、標準のカメラモジュール、NoIR カメラモジュール、高品質カメラ (HQ カメラ) モジュールという 3 つの種類があります。明るい場所で通常の画像や動画を撮影する場合は標準のカメラモジュールを使用し、特殊なレンズを使用して品質の高い画像を撮影する場合は HQ カメラモジュールを使用します。まったく光が差し込まない暗闇で赤外線を光源として撮影する場合は、赤外線フィルターがない NoIR カメラモジュールを使用します (赤外線は英語で IR と表記します。IR フィルターがないカメラなので、NoIR カメラといいます)。暗い場所や夜間に撮影を行う場合は (巣箱の中で撮影する場合や、夜間に撮影する場合など)、NoIR カメラと赤外線の光源が必要になります。

Raspberry Pi の標準カメラモジュールと NoIR カメラモジュールは、ソニーの IMX219 というイメージセンサーがベースになっています。このイメージセンサーは 8 メガピクセルセンサーと呼ばれるもので、最大 800 万ピクセルの写真を撮影することができます。この場合、幅が最大 3280 ピクセル、高さが最大 2464 ピクセルの画像がキャプチャされることになります。カメラモジュールでキャプチャできるのは、静止画像だけではありません。フル HD 解像度で毎秒 30 フレーム (30 fps) の速度でビデオ映像をキャプチャすることもできます。解像度を下げてフレームレートを上げることにより、ビデオ内での動作をよりスムーズにしたり、スローモーション効果を作成したりすることができます。たとえば、720 ピクセルのビデオ映像の場合は、フレームレートを 60fps に設定します。480 ピクセルのビデオ画像 (VGA) の場合は、フレームレートを最大 90fps まで設定することができます。

HQ カメラでは、12.3 メガピクセルのソニー IMX477 センサーが使用されています。これは、標準カメラモジュールと NoIR カメラモジュールのセンサーのピクセル値よりも大きいので、より多くの光を集めることができ、その結果として画像の品質が高くなります。ただし、カメラモジュールとは異なり、HQ カメラにはレンズが付属していないため、写真やビデオを撮影することはできません。C マウントまたは CS マウントが付属しているレンズであれば、好きなレンズを使用することができます。別のマウントが付いているレンズを使用する場合は、C マウントアダプターまたは CS マウントアダプターを取り付ける必要があります。レンズを取り付ける方法については、「付録 F:高品質カメラをセットアップしてみよう」を参照してください。



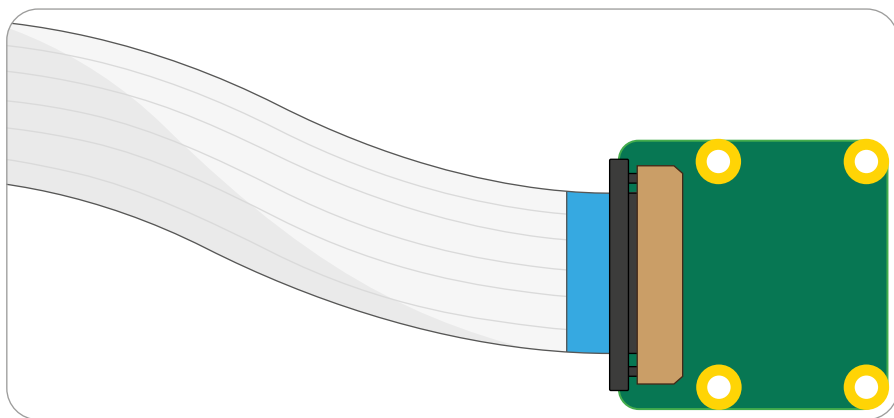
Raspberry Pi 400

Raspberry Pi のカメラモジュールを Raspberry Pi 400 で使用することはできません。USB タイプの Web カメラならば使用できますが、Raspberry Pi OS に組み込まれている Raspberry Pi カメラモジュール専用ソフトウェアツールを USB タイプの Web カメラで使用することはできません。

カメラを取り付ける

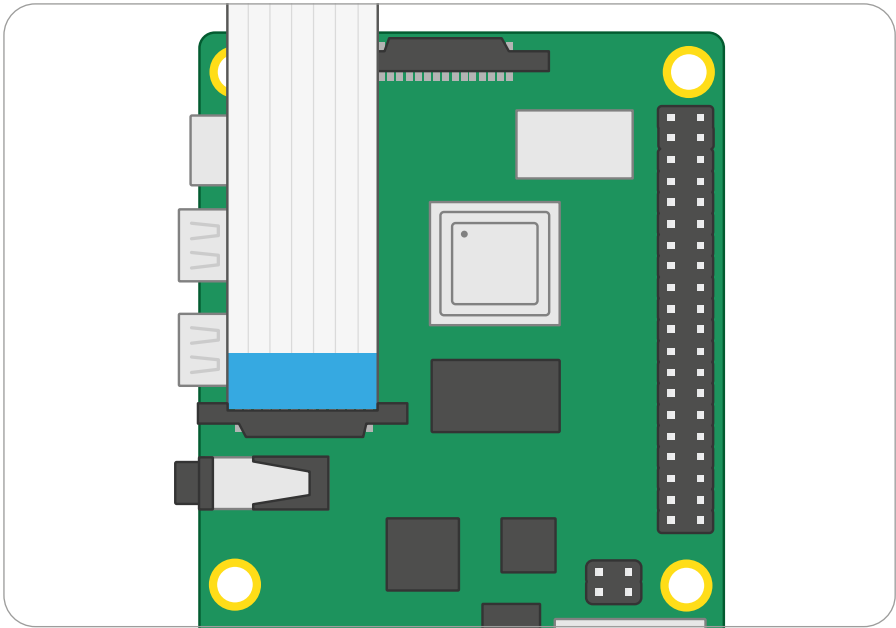
ほかのハードウェアアドオンと同様に、カメラモジュールと HQ カメラを Raspberry Pi に接続したり Raspberry Pi から取り外したりする場合は、必ず電源をオフにして電源ケーブルを抜いてください。Raspberry Pi の電源がオンになっている場合は、raspberrypi メニューで「Shutdown」を選択し、電源がオフになったことを確認してから電源ケーブルを抜いてください。

ほとんどの場合、カメラモジュールまたは HQ カメラには、付属のリボンケーブルがすでに接続されています。接続されていない場合は、センサーが下になるようにカメラボードを裏返して、平らなプラスチックコネクタを探してください。次に、このコネクタの突き出た部分に指の爪を慎重に引っ掛けて、コネクタをゆっくりと引き出します。リボンケーブルの銀色の端部を下に、青いプラスチックの部分を上に向けて、引き出したコネクタのフラップの下にリボンケーブルを差し込み、カチッと音がするまでゆっくりとフラップを押し込みます (図 8-1)。ケーブルのどちらの端部を差し込んでかまいません。ケーブルがまっすぐになっていれば、ケーブルが正しく差し込まれています。ケーブルが正しく差し込まれている場合、軽く引っ張っても抜けることはありません。正しく差し込まれていない場合は、フラップを引き上げてもう一度差し込んでください。



▲ 図 8-1:リボンケーブルをカメラモジュールに接続する

ケーブルのもう一方の端部も同じ方法で接続します。最初に、Raspberry Pi のカメラポート (または CSI ポート) の位置を確認し、ポートのフラップをゆっくりと引き上げます。Raspberry Pi がケースに収納されている場合は、最初にケースを取り外すと、ポートの位置を簡単に確認することができます。Raspberry Pi の HDMI ポートを手前に向けた状態で、リボンケーブルの銀色の端部を左側、青いプラスチックの部分を上に向けて、フラップの下にリボンケーブルを差し込み、フラップをゆっくりと元の位置まで押し込みます (図 8-2)。ケーブルがまっすぐになっていれば、ケーブルが正しく差し込まれています。ケーブルが正しく差し込まれている場合、軽く引っ張っても抜けることはありません。正しく差し込まれていない場合は、フラップを引き上げてもう一度差し込んでください。



▲ 図 8-2: Raspberry Pi のカメラポート/CSI ポートにリボンケーブルを接続する

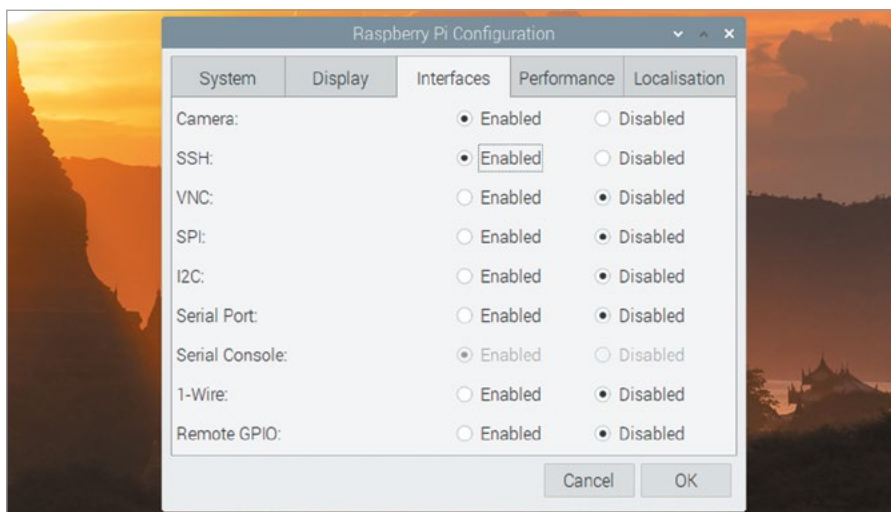
カメラモジュールには、製造時、出荷時、取り付け時に誤ってレンズに傷がつかないように、レンズを保護するための小さな青いプラスチック片が付属しています。カメラを利用する時にはレンズに気をつけてプラスチック片を取り外します。



フォーカスの調整

カメラモジュールには通常、レンズのフォーカスを調整するためのプラスチック製の小さなホイールが付属しています。通常は、フォーカスの設定を変更する必要はありませんが、非常に近い距離で撮影を行う場合は、このホイールをレンズ上でスライドさせてゆっくりひねると、レンズのフォーカスを手動で調整することができます。HQ カメラでフォーカスを調整する方法については、付録 F を参照してください。

Raspberry Pi の電源を入れ、Raspberry Pi OS を起動します。カメラを使用する前に、カメラが接続されていることを Raspberry Pi に認識させる必要があります。これを行うには、raspberrypi アイコンメニューを開いて「設定」カテゴリーを選択し、「Raspberry Pi の構成」をクリックします。構成ツールが起動したら「インターフェイス」タブをクリックし、リスト内に「カメラ」という項目が表示されていることを確認します。次に、「有効」オプションの左側にある丸いラジオボタンをクリックします(図 8-3)。これで、カメラがオンになります。この状態で「OK」をクリックすると、Raspberry Pi の再起動確認画面が表示されますので「OK」をクリックして再起動します。再起動完了が完了すればカメラを使う準備は完了です。

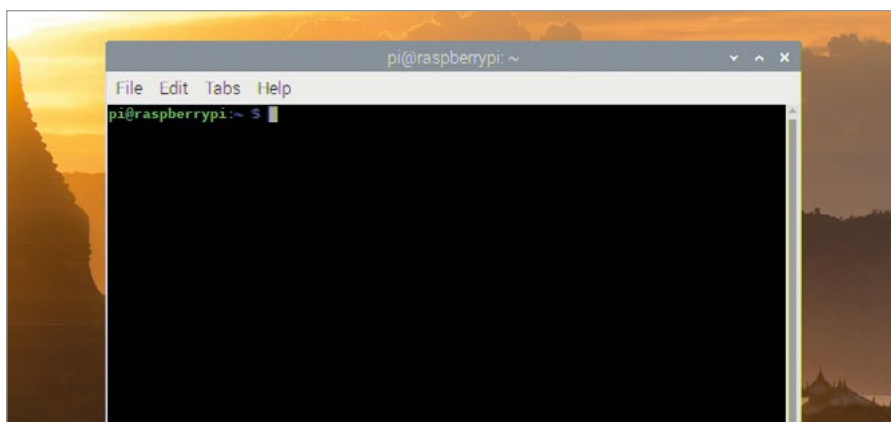


▲ 図 8-3: 「Raspberry Pi の構成」ツールでカメラをオンにする

カメラをテストする

raspistill ツールを使用すると、カメラモジュール/HQ カメラが正しく取り付けられているかどうか、「Raspberry Pi の構成」ツールでカメラインターフェイスが有効になっているかどうかを確認することができます。この **raspistill** ツールは、ビデオ用の **raspivid** ツールとともに、Raspberry Pi のコマンドラインインターフェイス (CLI) を使用してカメラの画像をキャプチャするように設計されています。

これまで見てきたプログラムとは異なり、**raspistill** ツールはメニューには表示されません。**raspistill** ツールを起動するには、**raspberry** アイコンをクリックしてメニューを表示し、「アクセサリ」カテゴリーを選択して「Terminal」をクリックします。緑と青の文字が表示された黒いウィンドウが表示されます (図 8-4)。これはターミナルと呼ばれるウィンドウで、このウィンドウからコマンドラインインターフェイスにアクセスすることができます。

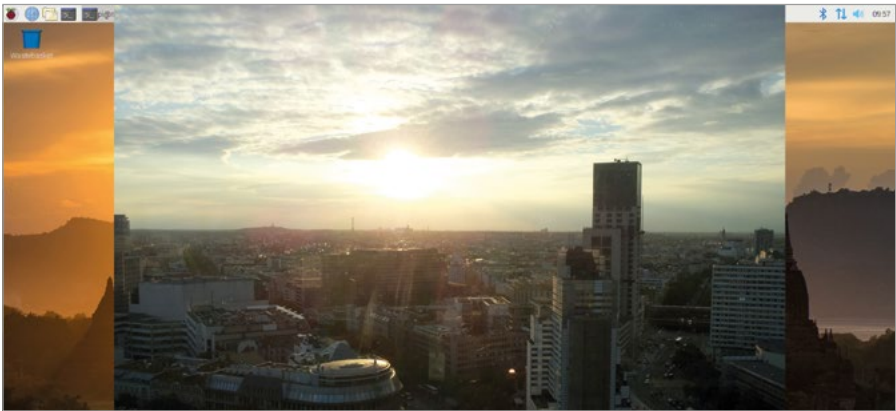


▲ 図 8-4: コマンドを入力するためのターミナルウィンドウを開く

カメラをテストするために、ターミナルウィンドウで以下のコマンドを入力してみましょう。

```
raspistill -o test.jpg
```

このコマンドを入力して **ENTER** キーを押すと、カメラのレンズから見える大きな画像が画面に表示されます (図 8-5)。これはライブプレビューと呼ばれる画像で、デフォルトで 5 秒間表示されます。5 秒が経過すると、1 枚の静止画像がキャプチャされ、home フォルダ内に「test.jpg」という名前で保存されます。別の画像をキャプチャする場合は、同じコマンドをもう一度入力します。ただし、出力ファイル名 (-o フラグの後の「test.jpg」の部分) を変更してください。このファイル名を変更しないと、最初に保存した画像が上書きされます。



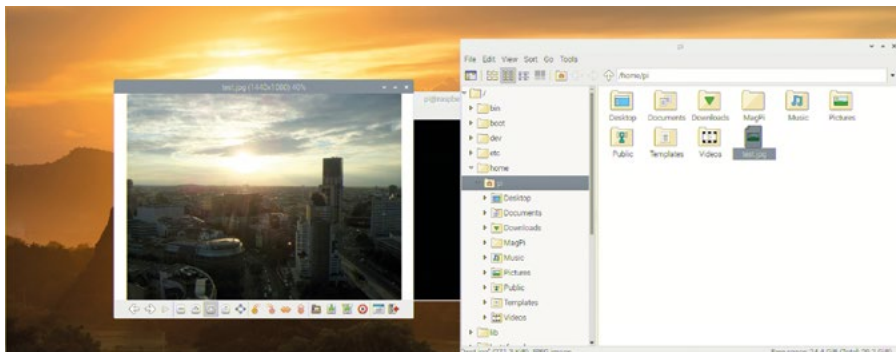
▲ 図 8-5:カメラのレンズから見たライブプレビュー

ライブプレビューの上下が逆になっている場合は、`raspistill` ツールを使用して、カメラの位置が逆になっていることを Raspberry Pi に認識させる必要があります。Raspberry Pi は、カメラモジュールの下端からリボンケーブルが出てくるように設計されていますが、一部のサードパーティ製カメラマウントアクセサリの場合、カメラモジュールの側部や上部からリボンケーブルが出てくるように設計されています。その場合は `-rot` スイッチを使用して、画像を回転させることができます (90 度、180 度、または 270 度)。カメラの上端からリボンケーブルが出ている場合は、以下のコマンドを実行します。

```
raspistill -rot 180 -o test.jpg
```

カメラの右端からリボンケーブルが出ている場合は、上のコマンドの「180」の部分で「90」に変更し、カメラの左端からリボンケーブルが出ている場合は、「270」に変更します。元のキャプチャ角度が間違っている場合は、`-rot` スイッチを使用して正しい角度に修正してください。

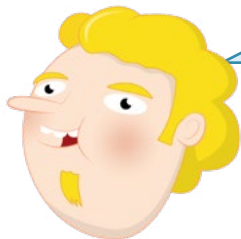
カメラで撮影した画像を表示するには、raspberrypi メニューの「アクセサリ」カテゴリで「ファイルマネージャ」を選択します。上で撮影した「test.jpg」という画像は、home/pi フォルダに保存されています。ファイルリストで test.jpg ファイルをダブルクリックすると、このファイルに保存されている画像が画像ビューアーに表示されます (図 8-6)。この画像をメールに添付したり、ブラウザを使用して Web サイトにアップロードしたり、外部のストレージデバイスにドラッグしたりすることができます。



▲ 図 8-6:キャプチャした画像を表示する

picamera について

picamera という Python 用の便利なライブラリーを使用すると、非常に柔軟な方法でカメラモジュールや HQ カメラを制御することができます。picamera により、カメラのプレビュー機能、画像撮影機能、ビデオ撮影機能を詳細に管理し、自分で作成したプログラムにこれらの機能を取り込むことができます。それだけでなく、自分で作成したプログラムを、GPIO Zero ライブラリー内の GPIO モジュールを使用するプログラムと組み合わせることもできます。



Python プログラミング

この章に記載されているプロジェクトを作成するには、Python というプログラミング言語、Thonny IDE、Raspberry Pi の GPIO ピンに関する知識が必要になります。まだこれらの知識がない場合は、最初に「第 5 章: Python を使ってプログラミングしてみよう」と「第 6 章: Scratch と Python を使って物理的なコンピューティングに挑戦しよう」を読んでください。

ターミナルウィンドウの右上に表示されている「X」をクリックしてウィンドウを終了し、raspberrypi メニューの「プログラミング」カテゴリで「Thonny」を選択します。次に、新しいプロジェクトを「Camera」という名前で保存し、スクリプト領域に以下のコードを入力します。これにより、プログラムに必要なライブラリーのインポートが開始されます。

```
from picamera import PiCamera
from time import sleep
camera = PiCamera()
```

最後の行で `camera` 関数を指定することにより、カメラモジュールまたは HQ カメラを制御することができます。カメラを制御するには、以下のように入力します。

```
camera.start_preview()
sleep(10)
camera.stop_preview()
```

「Run」をクリックすると、デスクトップが非表示になり、カメラのレンズから見える画像のプレビューが画面いっぱいに表示されます (図 8-7)。カメラを動かしたり、レンズの前で手を振ったりすると、その動作が画面上に表示されます。10 秒が経過するとプレビュー画面が終了し、プログラムも終了します。ただし、`raspistill` ツールで画像をプレビュー表示した場合とは異なり、プログラムの終了後に画像が保存されることはありません。



▲ 図 8-7:カメラのレンズから見たフルスクリーンのライブプレビュー

プレビューの方向が間違っている場合は、画像を正しい方向に回転させてください。`camera = PiCamera()` 行の下に、以下のように入力します。

```
camera.rotation = 180
```

プレビューの上下が逆になっている場合は、このように入力すると正しく表示されます。`raspistill` の場合と同様に、`camera.rotation` 関数を使用しても、画像を回転させることができます (90 度、180 度、または 270 度)。カメラモジュールのどの部分からケーブルが出ているかにより (右端、上端、左端)、回転角度を調整してください。間違った方向で画像やビデオをキャプチャしないように、プログラムの先頭に `camera.rotation` 関数を指定するようにしてください。

静止画像をキャプチャする

写真をキャプチャするには、ライブプレビューを表示するだけではなく、プログラムを変更する必要があります。最初に、プレビューの表示時間を短くしてみましょう。**sleep(10)** という行を、以下のように変更します。

```
sleep(5)
```

この行のすぐ下に、以下の行を追加します。

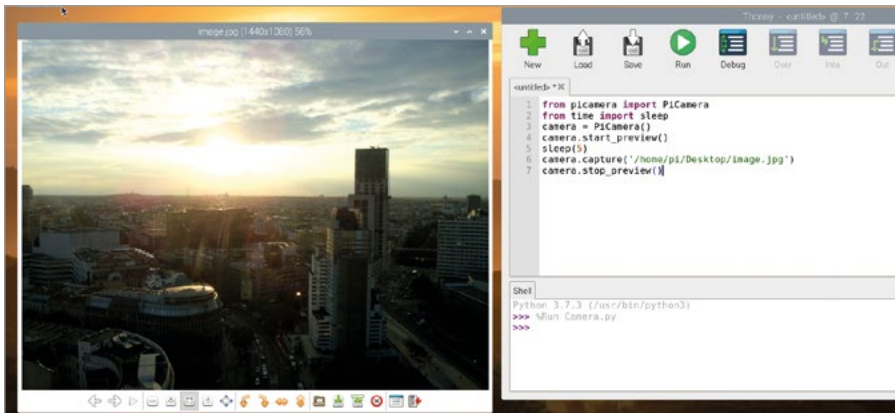
```
camera.capture('/home/pi/Desktop/image.jpg')
```



カメラの設定の調整にかかる時間

カメラがプレビューモードになっている場合、画像や動画の解析が実行されます。その際、最適な品質を確保するためにカメラの設定を調整する必要がありますかどうかを確認され、必要に応じて調整処理が実行されます。非常に暗い場所や非常に明るい場所でカメラを使用する場合などに、この調整処理が実行されます。調整処理の実行中にプレビュー画面を表示することはできませんが、調整処理が完了すると、すぐに鮮明なプレビュー画面が表示されます。この調整処理の時間を確保するため、2 秒以上の時間をプログラム内で指定してから、画像をキャプチャするようにしてください。

camera.capture 関数を使用して静止画像を保存することができますが、そのためには、その画像の名前と保存場所を指定する必要があります。ここでは、デスクトップに画像を保存します。ゴミ箱アイコンのすぐ下に、この画像のアイコンが表示されます。Thonny ウィンドウが邪魔な場合は、ウィンドウのタイトルバーをクリックして、別の場所にドラッグしてください。キャプチャした画像が保存されているファイルをダブルクリックすると、その画像が表示されます (図 8-8)。これで、カメラのプログラムが作成されました。



▲ 図 8-8: キャプチャした画像を表示する

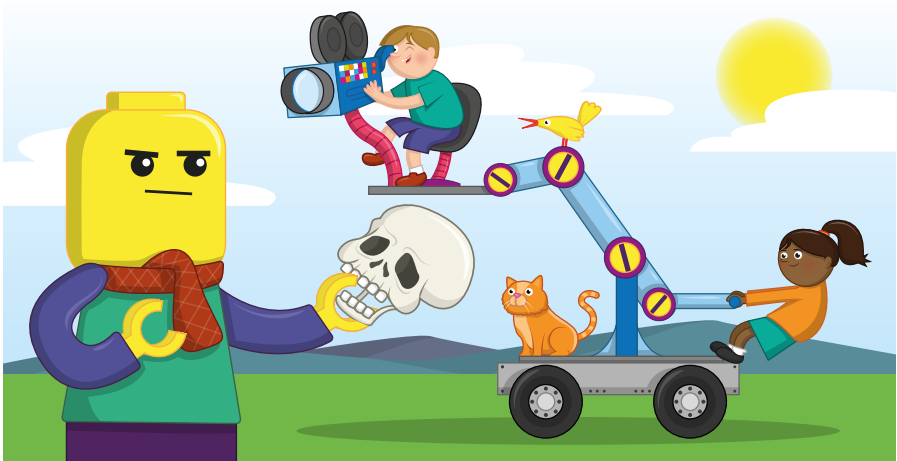
動画をキャプチャする

静止画像だけでなく、動画をキャプチャすることもできます。`camera.start_preview()` 行と `camera.stop_preview()` 行の間にあるコードをすべて削除し、`camera.start_preview()` 行の下に以下のコードを入力します。

```
camera.start_recording('/home/pi/Desktop/video.h264')
sleep(10)
camera.stop_recording()
```

「Run」をクリックすると、写真撮影と同じようにカメラのプレビューが表示されますが、今回はデスクトップ上に動画ファイルが保存されます。今回はプレビューが 10 秒間表示されるので、この時間にカメラの前で少し身体を動かしてみましょう。10 秒が経過したら、デスクトップ上のビデオファイルを開いてください。

デスクトップ上の `video.h264` ファイルをダブルクリックするだけで、動画を再生することができます。カメラの前で身体を動かした様子が動画として再生されます。動画の再生が終了すると、ターミナルウィンドウに簡単なメッセージが表示され、再生ソフトウェアが終了します。これで、Raspberry Pi のカメラモジュールまたは HQ カメラを使用して動画をキャプチャできるようになりました。



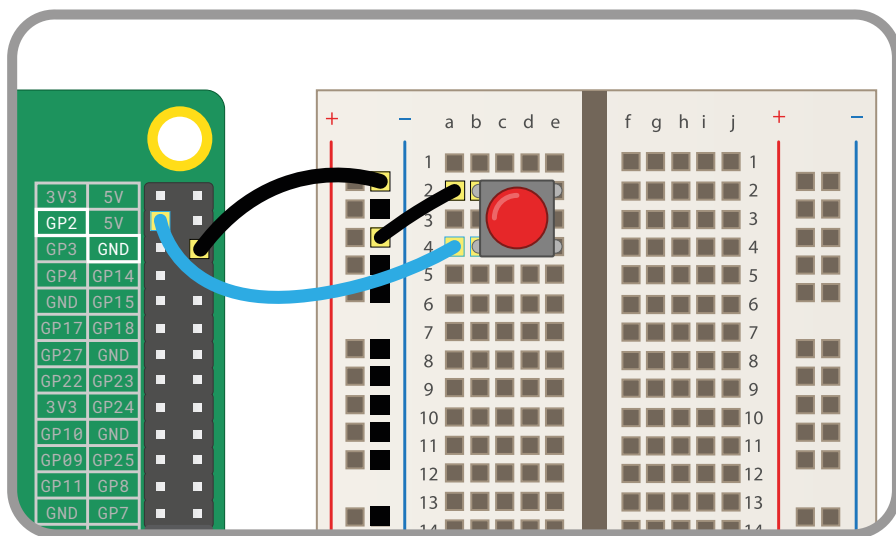
ボタンを押すと動作が止まるアニメーションを作成する

この章で学習した内容と、「第 6 章: Scratch と Python を使って物理的コンピューティングに挑戦しよう」で学習した内容 (Raspberry Pi の GPIO ヘッダーにハードウェアを接続する方法) を応用して、ボタンを押すとストップモーションになるアニメーションを作ってみましょう。

ストップモーションアニメを作成するには、動きのある人や動物やクルマなど (これらを「オブジェクト」と総称します) を表現するための多くの静止画像が必要になります。オブジェクトが写った、多くの画像を速いスピードで次々に表示することで、画像内のオブジェクトが動き回っているように見えます。速い動作にすることも、ゆっくりとした動作にすることもできます。

ここで作成するアニメーションでは、押しボタンスイッチ、ブレッドボード、オス - オス (M2M) ジャンパー線、オス - メス (M2F) ジャンパー線が必要になります。ブレッドボードがない場合は、メス - メス (F2F) ケーブルを使用してスイッチを接続できますが、プッシュ操作が少し難しくなります。これらのコンポーネントについて詳しくは、「第 6 章: Scratch と Python を使って物理的なコンピューティングに挑戦しよう」を参照してください。アニメーションに登場させるオブジェクトも必要になりますが、粘土で作った動物、おもちゃのクルマ、人形、フィギュアなど、何でもかまいません。

最初に、回路を作成します。押しボタンをブレッドボードに追加し、M2F ジャンパー線を使用して、ブレッドボードのアース線を Raspberry Pi のアースピン (図 8-9 で「GND」と記載されているピン) に接続します。次に、M2M ジャンパー線を使用して、押しボタンスイッチの一方の脚をブレッドボード上のアース線に接続し、M2F ジャンパー線を使用して、押しボタンスイッチのもう一方の脚を GPIO ピン 2 (図 8-9 で「GP2」と記載されているピン) に接続します。



▲ 図 8-9:押しボタンスイッチを GPIO ピンに接続するための配線図

Thonny で、新しいプロジェクトに「**Stop Motion**」という名前を付けて保存します。次に、カメラと GPIO ポートを使用するために必要なライブラリーのインポートと設定を行います。以下のコードを入力してください。

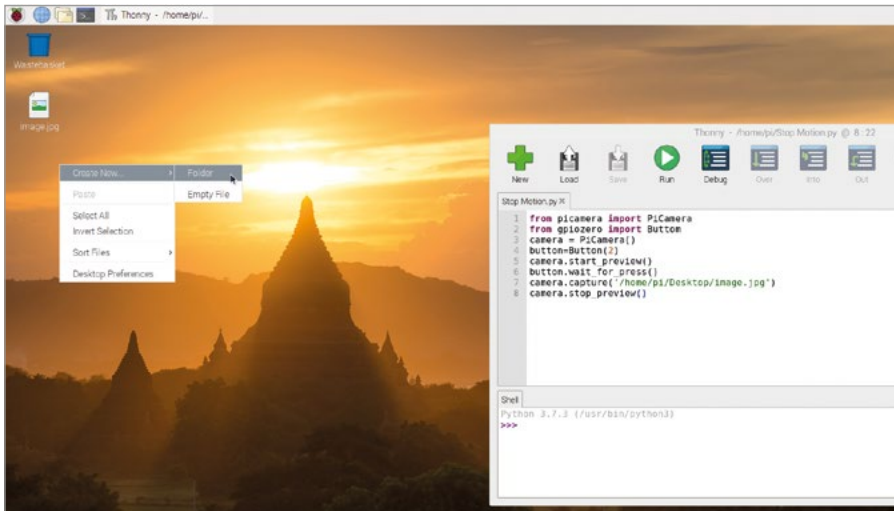
```
from picamera import PiCamera
from gpiozero import Button
camera = PiCamera()
button = Button(2)
```


次に、以下のコードを入力します。

```
camera.start_preview()
button.wait_for_press()
camera.capture('/home/pi/Desktop/image.jpg')
camera.stop_preview()
```

この状態で「Run」をクリックすると、カメラのレンズに映っているシーンのプレビューが表示されます。このプレビューは、押しボタンスイッチを押すまで画面に表示されたままになります。押しボタンスイッチを押すと、画像がデスクトップに保存されてからプレビューが終了します。画像は **image.jpg** というファイルに保存されます。このファイルをダブルクリックして、プログラムが正しく動作するかどうかを確認してください。

ストップモーションアニメを作成するには、多くの静止画像を使用して動きを表現する必要があります。これらの画像をすべてデスクトップに保存すると、デスクトップがごちゃごちゃして見にくくなるため、専用のフォルダーを作成する必要があります。デスクトップ上の空いている場所を右クリックし、「Create New」>「Folder」の順に選択します (図 8-10)。フォルダー名として「**animation**」と入力し (すべて小文字で入力します)、「OK」ボタンをクリックします。



▲ 図 8-10: キャプチャした画像を保存するための新しいフォルダーを作成する

アニメーション用の画像をキャプチャするたびにプログラムを再起動するのは面倒なので、ループ処理を実行するようにプログラムを変更しましょう。ただし、前の章で作成したループ処理とは異なり、ここで作成するループ処理の場合は、プログラムを適切なタイミングで終了する方法が必要になります。たとえば、プレビューの表示中にプログラムを終了すると、デスクトップ画面が表示されなくなります。そのためには、**try** と **except** という 2 つの特殊な命令を使用します。

最初に、`camera.start_preview()` の下にあるコードをすべて削除し、以下のように入力します。

```
frame = 1
```

このコードにより、`frame` という新しい変数が作成されます。この変数を使用して、現在のフレーム番号が保管されます。この変数を作成しないと、押しボタンスイッチを押すたびに、前回保存した画像が新しい画像で上書きされることになります。

次に、以下のように入力します。これが、ループ処理の先頭になります。

```
while True:
    try:
```

`try` 命令の内部に記述されたコードにより、画像がキャプチャされます。以下のように入力してください。

```
        button.wait_for_press()
        camera.capture('/home/pi/Desktop/animation/frame%03d.jpg' % frame)
        frame += 1
```

これらの 3 行のコードには、いくつかの便利な仕組みが組み込まれています。たとえば「%03d」の部分は、数値を取得して先頭にゼロを付加し、3 桁の数値に変換するためのコードです。「1」は「001」、「2」は「002」、「10」は「010」にそれぞれ変換されます。このコードにより、ファイルの順序を正しく管理し、すでに保存されているファイルの上書きを避けることができます。

2 行目の最後に記述されている「% `frame`」により、`frame` 変数の数値がファイル名で使用されます。3 行目の「`frame += 1`」により、`frame` 変数の値が 1 ずつ増えていくため、ファイル名が重複することはありません。初めてボタンを押すと、`frame` 変数の値が 1 から 2 になり、次にボタンを押すと 2 から 3 になり、その後も同様に 1 ずつ増えていきます。

この時点では、写真の撮影が終了すると、カメラのプレビューが表示されたままの状態でプログラムが終了します。カメラのプレビューを終了してからプログラムを終了させるためには、`try` 命令に対して `except` を記述する必要があります。以下のように入力しますが、`try` セクションと同じインデントレベルで入力してください。こうすることにより、`except` と `try` は別の命令であるということを Python が判断できるようになります。

```
        except KeyboardInterrupt:
            camera.stop_preview()
            break
```

最終的なプログラムは以下ようになります。

```

from picamera import PiCamera
from time import sleep
from gpiozero import Button
camera = PiCamera()
button = Button(2)
camera.start_preview()
frame = 1
while True:
    try:
        button.wait_for_press()
        camera.capture('/home/pi/Desktop/animation/frame%03d.jpg'
% frame)
        frame += 1
    except KeyboardInterrupt:
        camera.stop_preview()
        break

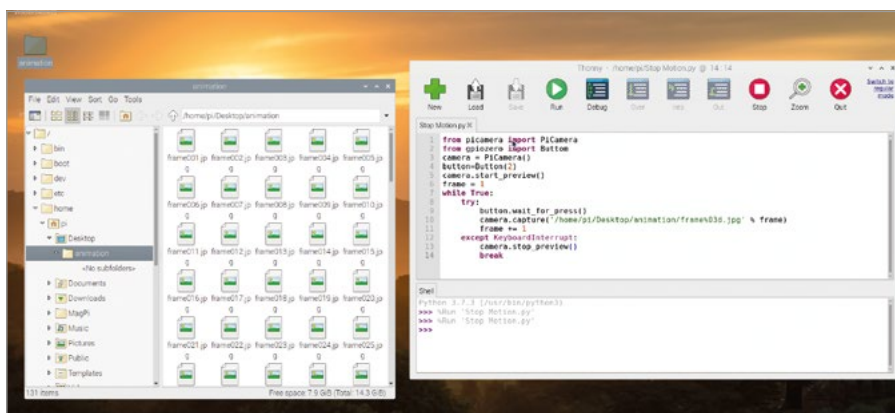
```

この状態で「Run」をクリックしてみましょう。次に、キーボードの **CTRL + C** キーを押して終了します。**CTRL** キーを押したまま **C** キーを押し、**C** キーを離してから **CTRL** キーを離すと失敗せずに入力できます。これらのキーは、プログラムに対する割り込みキーとして機能します。これらのキーを押すと、プログラムの処理が停止します。**except KeyboardInterrupt:** 行を記述しないと、カメラのプレビューが画面に表示されたままプログラムが終了しますが、この **except** 行を記述すると、**except** 行の内部に記述されているコードが実行されてからプログラムが終了します（この場合は、カメラのプレビューが終了してからプログラムが終了することになります）。

これで、ストップモーションアニメをキャプチャするための準備が整いました。アニメーションを作成するオブジェクトの撮影位置にカメラモジュールまたは HQ カメラを置き、動かないように固定します。カメラが動いてしまうと、きれいなアニメーションにはなりません。オブジェクトを撮影位置に置いて「Run」をクリックすると、プログラムが開始されます。プレビュー画面でオブジェクトの配置を確認し、押しボタンスイッチを押します。これで、最初のフレームがキャプチャされます。

次に、オブジェクトを少しでも移動し、もう一度押しボタンスイッチを押します。これで、2 番目のフレームがキャプチャされます。フレーム間でのオブジェクトの移動距離が短いほど、スムーズなアニメーションになります。アニメーションが完成するまで、この作業を繰り返します。キャプチャするフレームの数が多いほど、アニメーションの時間が長くなります。

作業が完了したら、**CTRL + C** キーを押してプログラムを終了し、デスクトップ上の **animation** フォルダをダブルクリックして、キャプチャしたフレーム（画像）を一覧表示します（図 8-11）。どのように撮影されているか、いくつか画像をダブルクリックして確認してみましょう。



▲ 図 8-11: フォルダ内に保存されたキャプチャ画像

この時点では、個別の静止画像がフォルダ内に保存されているだけで、まだアニメーションにはなっていません。アニメーションを作成するには、これらの静止画像を動画に変換する必要があります。これを行うには、**raspberrypi** アイコンをクリックしてメニューを表示し、「アクセサリ」カテゴリーを選択して「Terminal」をクリックします。これにより、Raspberry Pi に対するコマンドを入力するためのコマンドラインインターフェイスが起動します。コマンドラインインターフェイスについては、付録 C で詳しく説明します。ターミナルがロードされたら、以下のように入力して「animation」フォルダに移動します。

```
cd Desktop/animation
```

「Desktop」の「D」は、大文字で入力してください。これは、Raspberry Pi OS では大文字と小文字が区別されるためです。そのため、コマンドやフォルダ名を入力する場合は、大文字と小文字を正確に入力する必要があります。次に、以下のように入力します。

```
ffmpeg -i frame%03d.jpg -r 10 animation.h264
```

このコマンドにより、**ffmpeg** というプログラムが呼び出され、animation フォルダ内の静止画像が **animation.h264** という動画に変換されます。ffmpeg プログラムがインストールされていない場合は、**sudo apt-get install ffmpeg** コマンドを実行してインストールしてください。動画に変換する静止画像の数により、このプログラムが完了するまでの時間が異なります（数分かかる場合もあります）。プログラムが完了すると、ターミナルのプロンプト画面が再表示されます。

動画を再生するには、animation フォルダ内の **animation.h264** ファイルをダブルクリックします。または、ターミナルのプロンプト画面で以下のように入力して動画を再生することもできます。

```
omxplayer animation.h264
```

動画の読み込みが完了すると、ストップモーションアニメの再生が開始されます。これで、アニメーションが完成しました。

アニメーションの動きが速すぎたり遅すぎたりする場合は、`ffmpeg` コマンドの `-r 10` の「10」の部分を変更します。これは、フレームレートを指定するための値です。フレームレートとは、動画内で1秒間に再生される静止画像の数のことです。この値を小さくすると、アニメーションの動きがぎくしゃくした感じになり、大きな値を指定すると、アニメーションの動きがスムーズになります。ただし、大きな値を指定すると、アニメーションの全体的な再生時間が短くなります。

動画の調整が完了したら、その動画をデスクトップから Videos フォルダにドラッグして保存します。Videos フォルダに保存しないと、次にプログラムを実行したときに、デスクトップ上の動画ファイルが上書きされることとなります。

カメラの詳細設定

Raspberry Pi のカメラモジュールまたは HQ カメラをより詳細に設定する必要がある場合は、Python の `picamera` ライブラリーを使用します。このライブラリーには、さまざまな設定が用意されています。これらの設定とそのデフォルト値については、これ以降で詳しく説明します。

`camera.awb_mode = 'auto'`

カメラの自動ホワイトバランスモードを指定する場合は、この設定を使用します。指定できるモードは次のとおりです：**`off`**、**`auto`**、**`sunlight`**、**`cloudy`**、**`shade`**、**`tungsten`**、**`fluorescent`**、**`incandescent`**、**`flash`**、**`horizon`**。動画の色が少し青い場合や黄色い場合は、別のモードを試してみてください。

`camera.brightness = 50`

カメラ画像の明るさを指定する場合は、この設定を使用します。0 から 100 までの範囲内で明るさを指定することができます (0 が最も暗く、100 が最も明るくなります)。

`camera.color_effects = None`

カメラ内で使用されているカラー効果を変更する場合は、この設定を使用します。通常、この設定を変更する必要はありませんが、2つの数値を指定すると、色の記録方法を変更することができます。たとえば (**`128, 128`**) と指定すると、白黒の画像を作成することができます。

`camera.contrast = 0`

画像のコントラストを指定する場合は、この設定を使用します。大きな数値を指定すると、明暗がはっきりとした画像になり、小さな数値を指定すると、明暗がぼやけた画像になります。-100 から 100 までの範囲内で、任意の数値を指定することができます (-100 が最小コントラスト、100 が最大コントラストになります)。

`camera.crop = (0.0, 0.0, 1.0, 1.0)`

画像をトリミングする場合は、この設定を使用します。トリミングとは、画像の側面や上部などを切り取って、画像の必要な部分だけをキャプチャするという操作のことです。この設定では、画像の X 座標、Y 座標、幅、高さをそれぞれ数値で指定します。デフォルト設定の場合は、画像全体がキャプチャされます。たとえば、画像の幅 (1.0) と高さ (1.0) をそれぞれ -0.5 と 0.5 に変更して、画像がどのように表示されるかを確認してください。

camera.exposure_compensation = 0

カメラの露出補正を行う場合は、この設定を使用します。露出を補正することにより、各画像でキャプチャされる光の量を手で制御することができます。この設定を使用すると、明るさの設定を変更する場合は異なり、カメラ自体を制御することができます。-25 から 25 までの範囲内で、任意の数値を指定することができます (-25 が最も暗く、25 が最も明るくなります)。

camera.exposure_mode = 'auto'

露出モードを指定する場合は、この設定を使用します。露出モードにより、カメラモジュールまたは HQ カメラでの画像の露出方法が決まります。指定できるモードは次のとおりです: **off**、**auto**、**night**、**backlight**、**spotlight**、**sports**、**snow**、**beach**、**verylong**、**fixedfps**、**antishake**、**fireworks**

camera.framerate = 30

1 秒間の動画を作成するためにキャプチャする画像の数を指定する場合は、この設定を使用します。この画像の数のことを、フレームレートといいます。フレームレートの数値を大きくすると動画の動きがスムーズになりますが、使用する画像の数が多くなるため、より多くのストレージスペースが必要になります。フレームレートを高くする場合は、解像度を下げる必要があります。解像度を指定する場合は、**camera.resolution** という設定を使用します。

camera.hflip = False

画像を水平軸 (X 軸) を基準として反転する場合は、この設定を使用します。**True** を指定すると、画像が水平方向に反転します。

camera.image_effect = 'none'

特定の画像効果を動画ストリームに適用する場合は、この設定を使用します。適用した画像効果は、プレビュー画面だけでなく、保存した画像と動画にも表示されます。指定できる効果は次のとおりです: **blur**、**cartoon**、**colorbalance**、**colorpoint**、**colorswap**、**deinterlace1**、**deinterlace2**、**denoise**、**emboss**、**film**、**gpen**、**hatch**、**negative**、**none**、**oilpaint**、**pa stel**、**posterise**、**saturation**、**sketch**、**solarize**、**washedout**、**watercolor**

camera.ISO = 0

カメラの ISO 感度を変更する場合は、この設定を使用します。ISO 感度とは、光に対するカメラの感度のことです。デフォルト設定の場合、周辺の光量に応じて、カメラの ISO 感度が自動的に調整されます。ISO 感度の値として、100、200、320、400、500、640、800 のいずれかを指定することができます。ISO 感度の値を大きくすると、光量が少ない環境でのカメラの性能が高くなりますが、キャプチャする画像や動画の画質が粗くなります。

camera.meter_mode = 'average'

カメラの露出の設定時に光量を指定する場合は、この設定を使用します。デフォルト値は「average」ですが、この場合、画像全体で光量が平均化されます。デフォルト値以外にも、**backlit**、**matrix**、**spot** を指定することができます。

camera.resolution = (1920, 1080)

キャプチャした画像または動画の解像度 (幅と高さ) を指定する場合は、この設定を使用します。2 つの数値で、幅と高さを指定します。解像度の値を小さくすると、画像を保管するためのスペースが少なくなるため、フレームレートを高くすることができます。解像度の値を大きくすると、画像の品質が高くなりますが、画像を保管するためにより多くのスペースが必要になります。

camera.rotation = 0

画像を回転する場合は、この設定を使用します。画像の回転角度として、0 ~ 90、180、270 を指定することができます。たとえば、下部からリボンケーブルが出てくるようにカメラを配置できない場合に、この設定を使用してください。

camera.saturation = 0

画像の彩度 (色の鮮やかさ) を調整する場合は、この設定を使用します。-100 から 100 までの範囲内で、任意の値を指定することができます。

camera.sharpness = 0

画像の鮮明度を調整する場合は、この設定を使用します。-100 から 100 までの範囲内で、任意の値を指定することができます。

camera.shutter_speed = 0

画像や動画をキャプチャするときのシャッター速度を調整する場合は、この設定を使用します。シャッター速度は、マイクロ秒単位で設定することができます。暗い場所ではシャッター速度を下げ、明るい場所ではシャッター速度を上げると、適切な露出で画像をキャプチャすることができます。通常は、デフォルト値 (自動調整) をそのまま使用します。

camera.vflip = False

画像を垂直軸 (Y 軸) を基準として反転する場合は、この設定を使用します。**True** を指定すると、画像が垂直方向に反転します。

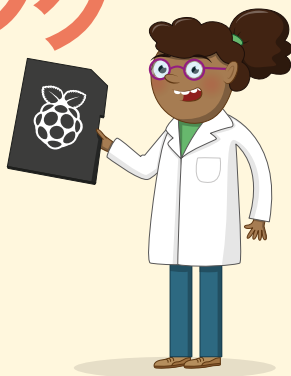
camera.video_stabilization = False

動画の揺れを抑える場合は、この設定を使用します。**True** を指定すると、動画の揺れを抑えることができます。カメラモジュールまたは HQ カメラをロボットに取り付けて撮影する場合や、カメラモジュールまたは HQ カメラを持ち歩きながら撮影する場合は、この設定を使用すると、カメラの揺れを抑えることができます。

これらの設定と、このガイドに記載されていない設定の詳細については、[picamera.readthedocs.io](https://readthedocs.io) を参照してください。

付録 A

microSD カードに オペレーティング システムを インストール してみよう



NOOBS が事前にインストールされた microSD カードを購入すると、Raspberry Pi OS (以前は Raspbian と呼ばれていました) を簡単にインストールすることができます (NOOBS とは、デフォルト設定のまますぐに使用できるソフトウェアのことです)。この microSD カードは、Raspberry Pi のすべての販売店で購入することができます。また、以下の手順に従い、Raspberry Pi Imager を使用して、何も書き込まれていない microSD カードにオペレーティングシステムを手動でインストールすることもできます (すでにデータが書き込まれている microSD カードに上書きでインストールすることもできます)。

注意

NOOBS が事前にインストールされている microSD カードを購入した場合は、カードを Raspberry Pi に差し込むだけで、それ以外の操作は必要ありません。何も書き込まれていない microSD カードに OS をインストールする場合と、すでにファイルが書き込まれている microSD カードに OS をインストールする場合に、以下の手順を実行してください。すでにファイルが書き込まれている microSD カードに対して以下の手順を実行すると、カード内のファイルがすべて上書きされてしまうため、事前にバックアップを作成してください。

Raspberry Pi Imager をダウンロードする

Debian をベースとして開発された Raspberry Pi OS は、Raspberry Pi の公式オペレーティングシステムです。Raspberry Pi の microSD カードに Raspberry Pi OS をインストールする最も簡単な方法は、[rpf.io/downloads](https://www.raspberrypi.org/documentation/installation/installing-images/README.md) から Raspberry Pi Imager ツールをダウンロードし、このツールを使用してインストールを行うという方法です。このガイドではこの方法をお勧めしますが、NOOBS を使用してオペレーティングシステムをインストールしてもかまいません (NOOBS も、[rpf.io/downloads](https://www.raspberrypi.org/documentation/installation/installing-images/README.md) からダウンロードすることができます)。

Raspberry Pi Imager アプリケーションは、Windows コンピューター、macOS コンピューター、Ubuntu Linux コンピューターに対応しています。使用しているコンピューターに対応したバージョンのアプリケーションをダウンロードしてください。

macOS コンピューターの場合は、ダウンロードした DMG ファイルをダブルクリックします。「App Store と確認済みの開発元からのアプリケーションを許可」オプションを使用してダウンロードしたアプリケーションを実行する場合、「セキュリティとプライバシー」設定を変更しなければならないことがあります。次に、Raspberry Pi Imager アイコンを Applications フォルダにドラッグします。Windows コンピューターの場合は、ダウンロードした EXE ファイルをダブルクリックします。プロンプト画面が表示されたら「Yes」ボタンを選択します。これにより、アプリケーションを実行できるようになります。次に「Install」ボタンをクリックして、インストールを開始します。

OS を microSD カードに書き込む

microSD カードを Windows コンピューターまたは Mac コンピューターに差し込みます。使用しているコンピューターにカードリーダーが付属していない場合は、microSD カード用の USB アダプターが必要になります。microSD カードを事前にフォーマット（初期化）しておく必要はありません。

次に、Raspberry Pi Imager アプリケーションを起動し、「Choose OS」ボタンをクリックしてインストールするオペレーティングシステムを選択します。先頭には、標準の Raspberry Pi OS が表示されます。簡易版の Lite バージョンや、お勧めのソフトウェアがすべて事前にインストールされている Full バージョンをインストールする場合は、「Raspberry Pi OS (other)」を選択します。LibreELEC、Ubuntu Core、Ubuntu Server をインストールするためのオプションもあります（LibreELEC をインストールする場合は、使用する Raspberry Pi モデルのバージョンを選択する必要があります）。



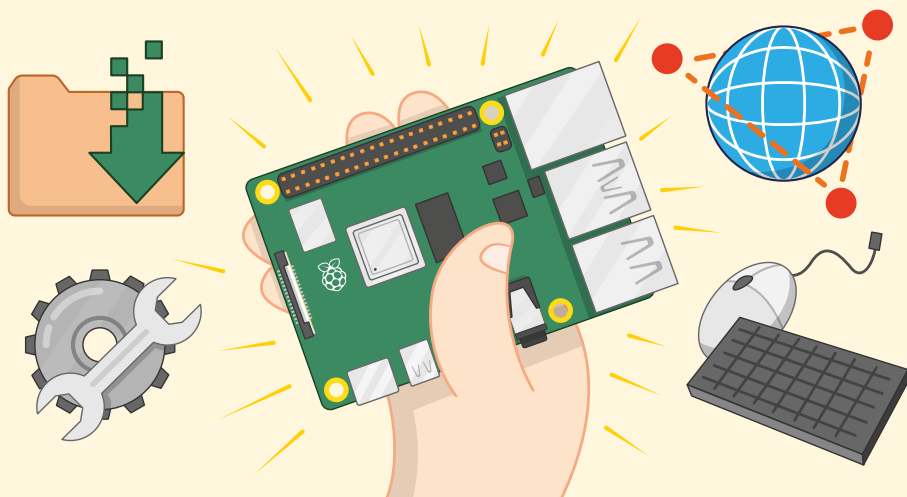
注: Lakka などの OS をインストールする場合は、その OS のイメージファイルを対象の Web サイトからダウンロードし、Raspberry Pi Imager で「Use Custom」オプションを選択します。

次に、インストールする OS を選択して「Choose SD card」ボタンをクリックし、インストール先の microSD カードを選択します（通常は、インストール先の microSD カードが 1 つだけ表示されます）。

最後に「Write」ボタンをクリックします。これにより、選択した OS がカードに書き込まれ、検証処理が実行されます。書き込み処理と検証処理が完了したら、microSD カードを取り外してかまいません。取り外したカードを Raspberry Pi に差し込んで Raspberry Pi を起動すると、カードにインストールされたオペレーティングシステムが稼働します。

付録 B

ソフトウェアをインストール/アンインストールしてみよう



Raspberry Pi OS には、製造元の Raspberry Pi Foundation が厳選した人気の高いソフトウェアパッケージが組み込まれていますが、Raspberry Pi で使用できるのはこれらのパッケージだけではありません。以下で説明する手順を実行すると、追加のソフトウェアを選択してインストールしたり、アンインストールしたりすることができます。追加のソフトウェアをインストールすると、Raspberry Pi の機能を拡張することができます。

ここで説明する手順を実行するには、「第 3 章: Raspberry Pi を使ってみよう」に記載されている推奨ソフトウェアツールの使用方法を理解する必要があります。第 3 章をまだ読んでいない場合は、最初に第 3 章を読んでその内容を理解してください。

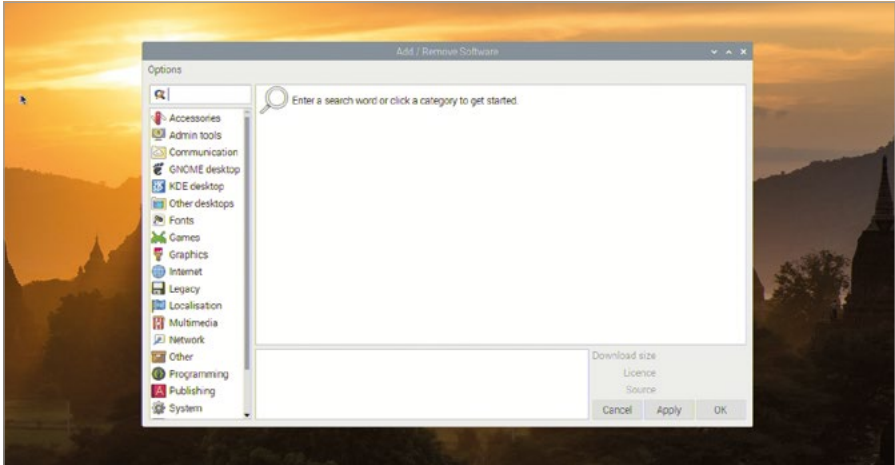


カードの容量

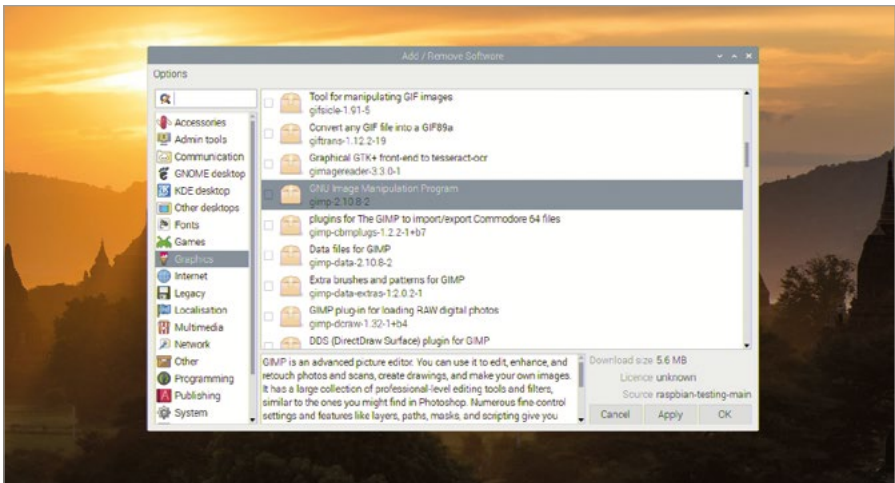
Raspberry Pi に追加のソフトウェアをインストールすると、microSD カードの空き容量がその分だけ少なくなります。容量が 16GB 以上のカードを使用すると、多くのソフトウェアをインストールすることができます。使用するカードが Raspberry Pi に対応しているかどうかを確認する方法については、rpf.io/sdcardlist を参照してください。

使用可能なソフトウェアを表示する

Raspberry Pi OS で使用できるソフトウェアパッケージの一覧を表示するには、ソフトウェアリポジトリを使用します。raspberrypi アイコンをクリックしてメニューを表示し、「設定」カテゴリを選択して「Add/Remove Software」をクリックしてください。少し待つと、「Add/Remove Software」ウィンドウが表示されます。



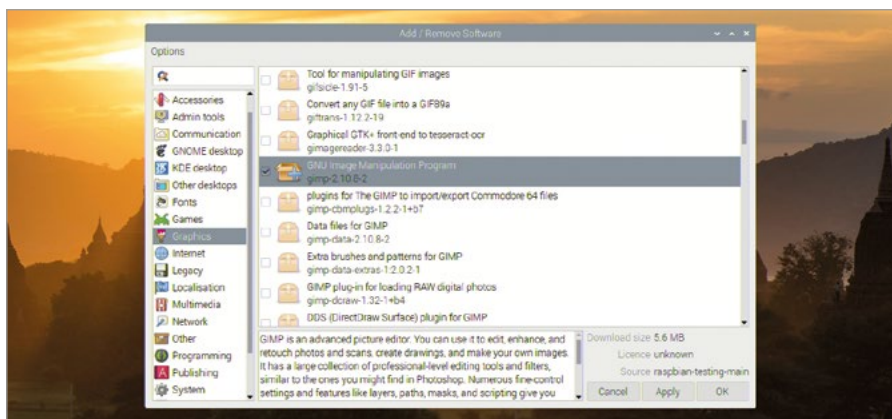
「Add/Remove Software」ウィンドウの左側に、カテゴリが一覧表示されます。これらのカテゴリは、raspberrypi アイコンをクリックしたときにメインメニューに表示されるカテゴリと同じものです。いずれかのカテゴリをクリックすると、そのカテゴリに属するソフトウェアが一覧表示されます。または、ウィンドウ左上のボックスに「テキストエディター」や「ゲーム」などの検索語を入力してもかまいません。検索語に一致するソフトウェアパッケージが表示されます。いずれかのパッケージをクリックすると、そのパッケージに関する詳細情報がウィンドウの下部に表示されます。



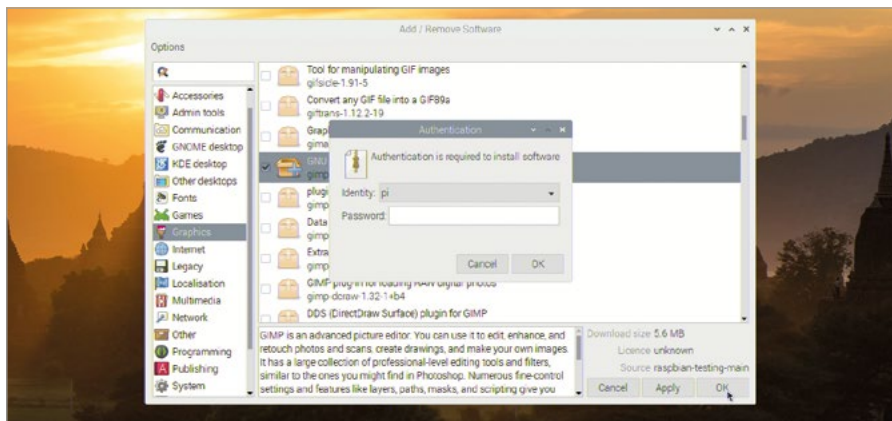
選択したカテゴリーに多数のソフトウェアパッケージが含まれている場合は、「Add/Remove Software」画面に一覧が表示されるまでに多少時間がかかることがあります。

ソフトウェアをインストールする

インストールするパッケージを選択するには、そのパッケージの横に表示されているボックスをクリックしてチェックマークを付けます。一度に複数のパッケージをインストールすることができます。インストールするパッケージのボックスをすべてクリックしてください。選択したパッケージには「+」記号が表示されます。この記号は、そのパッケージがインストールされるということを表しています。

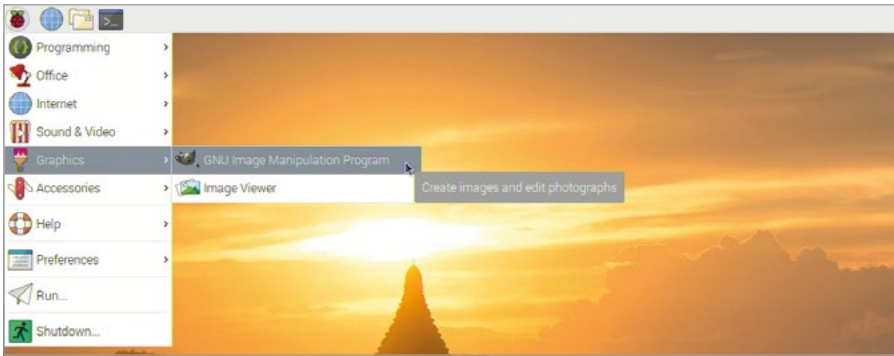


選択したパッケージを確認したら、「OK」ボタンまたは「Apply」ボタンをクリックします。「OK」ボタンをクリックすると、ソフトウェアのインストール後に「Add/Remove Software」ウィンドウが終了しますが、「Apply」ボタンをクリックすると、ソフトウェアのインストール後も「Add/Remove Software」ウィンドウが表示されたままになります。「OK」ボタンまたは「Apply」ボタンをクリックすると、パスワードの入力画面が表示されます。これは、自分以外のだれかがソフトウェアパッケージのインストールやアンインストールを行うことを防ぐためです。



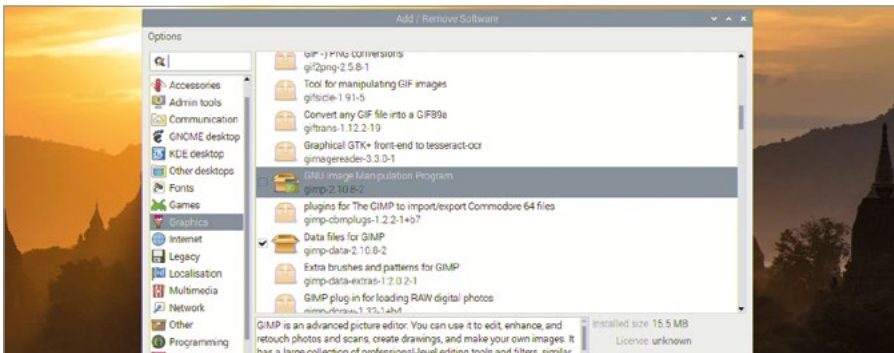
パッケージを 1 つだけインストールしたにもかかわらず、そのパッケージと一緒にいくつかのパッケージがインストールされる場合があります。これを、依存関係 といいます。たとえば、ゲームソフトウェアをインストールする場合、ゲーム用のサウンドエフェクトパッケージや Web サーバー上で使用されるデータベースなどが一緒にインストールされることがあります。

ソフトウェアパッケージのインストールが完了したら、raspberrry アイコンをクリックしてメニューを表示し、目的のカテゴリーにそのソフトウェアパッケージが表示されることを確認してください。メインメニューには、常に「Add/Remove Software」ウィンドウと同じカテゴリーが表示されるわけではありません。たとえば、一部のソフトウェアについては、メインメニュー上に何もエントリーが表示されません。こうしたソフトウェアはコマンドラインソフトウェア と呼ばれ、ターミナル上で実行する必要があります。コマンドラインとターミナルの詳細については、「付録 C: コマンドラインインターフェイス」を参照してください。



ソフトウェアをアンインストールする

削除するソフトウェアパッケージを選択するには (ソフトウェアを削除することをアンインストール といいます)、目的のパッケージを探し (検索機能を使用すると便利です)、そのパッケージの横にあるボックスをクリックしてチェックマークを外します。一度に複数のパッケージをアンインストールすることができます。アンインストールするパッケージのボックスをすべてクリックしてください。選択したパッケージには、小さなゴミ箱アイコンが表示されます。このアイコンは、そのパッケージがアンインストールされることを表しています。



インストールの場合と同様に、「OK」または「Apply」をクリックします。これにより、選択したソフトウェアパッケージのアンインストールが開始されます。これまでの手順でパスワードを入力していない場合は、「OK」または「Apply」をクリックしたときにパスワードの入力画面が表示されます。また、アンインストールするソフトウェアパッケージに依存するパッケージ（依存関係パッケージ）も削除するかどうかを確認するための画面が表示される場合もあります。ソフトウェアのアンインストールが完了すると、そのソフトウェアが raspberry アイコンのメニューに表示されなくなります。ただし、そのソフトウェアを使用して作成されたファイル（グラフィックパッケージ用の画像ファイルや、ゲーム用に保存されたファイルなど）については、削除されることはありません。

注意

Raspberry Pi OS にインストールされているソフトウェアパッケージは、Raspberry Pi を実行するために必要なソフトウェアパッケージも含めて、すべて「Add/Remove Software」ウィンドウに表示されます。これらのパッケージは削除できますが、誤って削除しないように、本当に不要かどうか確認できない場合は、パッケージをアンインストールしないようにしてください。必要なパッケージを誤ってアンインストールしてしまった場合は、「第 2 章: Raspberry Pi をセットアップしよう」の説明に従って Raspberry Pi OS をもう一度インストールするか、付録 A の説明に従ってオペレーティングシステムを再インストールしてください。

付録 C コマンドラインイン ターフェイス



Raspberry Pi にインストールされているほとんどのソフトウェアはデスクトップ上で管理することができますが、いくつかのソフトウェアについては、ターミナルと呼ばれるアプリケーションでコマンドラインインターフェイス (CLI) というテキストベースのインターフェイスを使用してアクセスする必要があります。ほとんどの場合、CLI を使用する必要はありませんが、この付録では、CLI の基礎的な知識について説明します。

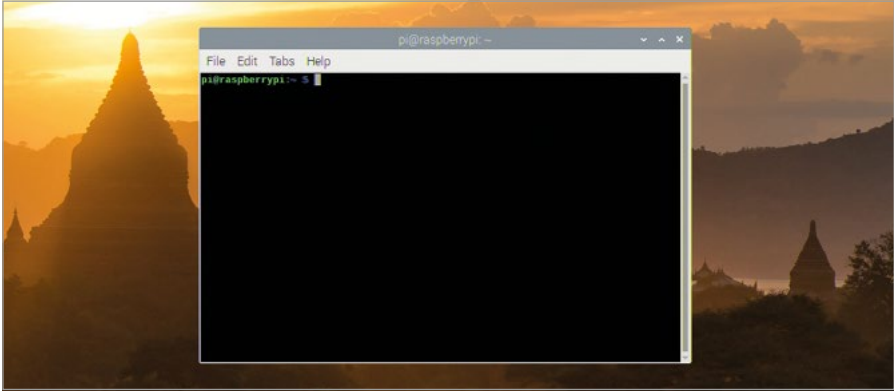


詳しい情報について

この付録では、Linux コマンドラインインターフェイスについて詳しくは説明しません。詳しい情報については、Web ブラウザーで rpf.io/terminal にアクセスしてください。

ターミナルを起動する

CLI にアクセスするには、仮想テレタイプ端末 (VTY 端末) を読み込むためのソフトウェアパッケージであるターミナルを使用する必要があります。VTY 端末とは、初期のコンピューターが登場した頃に使われていた用語で、当時は、現在のようなキーボードとモニターではなく、大型の電気機械式タイプライターを使用してコマンドを発行していました。ターミナルパッケージを起動するには、raspberry アイコンをクリックしてメニューを表示し、「アクセサリ」カテゴリーを選択して「Terminal」をクリックします。



ターミナルウィンドウは他のウィンドウと同様に、デスクトップ上でドラッグしたり、サイズを変更したり、最大化したり、最小化したりすることができます。また、ウィンドウ内の文字のサイズを変更することもできます。これを行うには、「Edit」メニューをクリックして「Zoom In」または「Zoom Out」を選択するか、キーボードで **CTRL** キーを押しながら「+」記号または「-」記号を押します。

プロンプト

ターミナルを起動すると、ターミナルウィンドウの先頭にプロンプトが表示されます。Raspberry Pi OS が稼働している Raspberry Pi の場合は、以下のようなプロンプトが表示されます。

```
pi@raspberrypi:~$
```

プロンプトの先頭部分の「**pi**」は、ユーザー名を表しています。**@** の後ろにある「**raspberrypi**」の部分は、コンピューターのホスト名を表しています (デフォルト値は「**raspberrypi**」です)。「**:**」の後ろにある「**~**」の部分は、チルダといえます。チルダは、ホームディレクトリを参照するための簡易的な方法で、現在の作業ディレクトリ (CWD) を表しています。最後の「**\$**」の部分は、現在ログインしているユーザーが非特権ユーザーであることを表しています。非特権ユーザーとは、ソフトウェアのインストールやアンインストールなどを実行する場合にパスワードを入力しなければならないユーザーのことです。

ディレクトリを移動する

ターミナルウィンドウで以下のように入力して **ENTER** キーを押してみましょう。

```
cd Desktop
```

プロンプトが以下のように変わります。

```
pi@raspberrypi:~/Desktop $
```

これは、現在の作業ディレクトリが変更されたことを示しています。「cd Desktop」を入力する前の作業ディレクトリはホームディレクトリ (~) でしたが、入力後は、ホームディレクトリ内の **Desktop** というサブディレクトリに変わりました。このように、現在の作業ディレクトリを変更する場合は、**cd** コマンドを使用します（「cd」は、change directory の省略形です）。



大文字と小文字の区別

Raspberry Pi OS のコマンドラインインターフェイスでは、大文字と小文字が区別されます。そのため、コマンドやファイル名を入力する場合は、大文字と小文字を正しく入力する必要があります。上記のコマンドを入力したときに「no such file or directory」というメッセージが表示された場合は、「Desktop」の「D」が大文字になっているかどうかを確認してください。

ホームディレクトリに戻る方法は 4 つあります。それぞれの方法について説明します（いずれの場合も、ホームディレクトリに戻ってからまた **Desktop** サブディレクトリに移動します）。最初の方法は、以下のように入力する方法です。

```
cd ..
```

「..」の部分は、ショートカットを表しています。この場合、「現在のディレクトリから 1 つ上の階層のディレクトリに移動する」という意味になります。こうした上位階層ディレクトリのことを親ディレクトリといいます。**Desktop** サブディレクトリの上位階層ディレクトリはホームディレクトリなので、「cd ..」と入力するとホームディレクトリに移動することになります。もう一度 **Desktop** サブディレクトリに移動し、2 番目の方法として以下のように入力します。

```
cd ~
```

「~」の部分は、「ホームディレクトリに移動する」という意味になります。「cd ..」の場合は、現在の作業ディレクトリからその親ディレクトリに移動するだけなので、1 つ上の階層のディレクトリがホームディレクトリではない場合、直接ホームディレクトリに移動することはできませんが、「cd ~」の場合は、どのディレクトリからでも直接ホームディレクトリに移動することができるため、「cd ..」コマンドよりも簡単です。しかし、「cd ~」コマンドよりもさらに簡単な方法があります。それは、以下のように入力する方法です。

```
cd
```

「cd」だけを入力すると、デフォルトでホームディレクトリに移動します。最後の方法は、以下のように入力する方法です。

```
cd /home/pi
```

この方法では、絶対パス というパスを入力します。絶対パスを指定すると、現在の作業ディレクトリがどのディレクトリであっても、指定したパスのディレクトリに移動することができます。そのため、「**cd**」や「**cd ~**」の場合と同様に、絶対パスを指定した場合もホームディレクトリに直接移動できますが、最後にユーザー名を指定する必要があります。

ファイルの処理

次に、ファイルの操作方法を練習してみましょう。**Desktop** ディレクトリに移動して以下のコマンドを入力してください。

```
touch Test
```

デスクトップに **Test** というファイルが表示されます。**touch** コマンドは通常、ファイルの日時情報を更新する場合に使用しますが、新しいファイルを作成することもできます（上記の場合は、「Test」というファイルが新しく作成されます）。

次に、以下のように入力します。

```
cp Test Test2
```

デスクトップに **Test2** というファイルが表示されます。これは、元のファイルをコピー するためのコマンドです。ファイル名以外は、すべて元のファイルと同じものが作成されます。このファイルを削除するには、以下のように入力します。

```
rm Test2
```

これは、ファイルを削除 するためのコマンドです。削除されたファイルは、デスクトップに表示されなくなります。

注意

rm コマンドを実行してファイルを削除した場合、削除したファイルがゴミ箱に保管されるグラフィカルなファイルマネージャとは異なり、ファイルが完全に削除されます。そのため、**rm** コマンドを実行する場合は注意してください。

次に、以下のように入力します。

```
mv Test Test2
```

これは、ファイルを移動 するためのコマンドです。元の **Test** ファイルがデスクトップから消えて、代わりに **Test2** ファイルが表示されます。このように、**mv** コマンドでファイル名を変更することができます。

Desktop ディレクトリ以外のディレクトリ内のファイルを表示することもできます。以下のように入力します。

```
ls
```

これは、現在の作業ディレクトリの内容を一覧表示するためのコマンドです。「ls」の後ろにディレクトリ名を指定すると、そのディレクトリの内容が表示されます。また、各種のスイッチを指定することにより、隠しファイルやファイルサイズなど、さまざまな詳細情報も表示することができます。たとえば、以下のように入力します。

```
ls -larth
```

「larth」の部分が、ls コマンドのスイッチになります。l スwitchを指定すると、出力情報が縦方向に一覧表示されます。a スwitchを指定すると、隠しディレクトリや隠しファイルを含め、すべてのディレクトリとファイルが表示されます。r スwitchを指定すると、通常とは逆の順序で出力情報が表示されます。t スwitchを指定すると、更新時刻の順にファイルが表示されます。r スwitchと組み合わせて指定すると、最も古いファイルが先頭に表示され、最も新しいファイルが最後に表示されます。h スwitchを指定すると、ファイルサイズが読みやすい形式で表示されます。

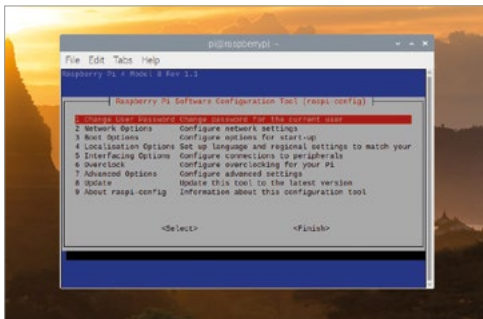
プログラムを実行する

プログラムの中には、コマンドラインインターフェイスでしか実行できないプログラムもあれば、グラフィカルインターフェイスとコマンドラインインターフェイスの両方で実行できるプログラムもあります。たとえば Raspberry Pi Software Configuration Tool は、グラフィカルインターフェイスとコマンドラインインターフェイスの両方で実行することができます。このツールは通常、raspberry アイコンメニューから起動します。

以下のように入力します。

```
raspi-config
```

「このツールは root ユーザーとして実行する必要があります」という内容のエラーメッセージが表示されます。root ユーザーとは、Raspberry Pi のスーパーユーザーアカウントを持っているユーザーのことです。以下のように入力すると、root ユーザーとしてツールを実行することができます。



```
sudo raspi-config
```

このコマンドの「sudo」の部分は、ユーザーの切り替えを実行するという意味です。これにより、root ユーザーとしてコマンドが実行されることになります。

sudo コマンドを実行する必要があるのは、ソフトウェアのインストールやアンインストール、システム設定の変更など、レベルの高い権限が必要な処理を実行する場合だけです。そうした権限が必要ないプログラム（ゲームなど）を実行する場合は、**sudo** コマンドを使用しないでください。

TAB キーを 2 回押して「Finish」を選択し、次に **ENTER** キーを押すと、Raspberry Pi Software Configuration Tool が終了し、コマンドラインインターフェイスに戻ります。最後に、以下のように入力します。

```
exit
```

このコマンドを実行すると、コマンドラインインターフェイスセッションが終了し、ターミナルアプリケーションが停止します。

TTY を使用する

コマンドラインインターフェイスを使用するための方法は、ターミナルアプリケーションだけではありません。テレタイプ (TTY) というアプリケーションに切り替えて、コマンドラインインターフェイスを使用することもできます。キーボードで **CTRL** キーと **ALT** キーを押しながら **F2** キーを押すと、「tty2」に切り替わります。

```
Raspbian GNU/Linux 9 raspberrypi tty2
raspberrypi login:
```

ユーザー名とパスワードをもう一度入力すると、ターミナルアプリケーションの場合と同じようにコマンドラインインターフェイスを使用できるようになります。メインのデスクトップインターフェイスが使用できない場合は、TTY を使用すると便利です。

TTY からデスクトップ画面に戻る場合は、**CTRL + ALT** キーを押しながら **F7** キーを押します。**CTRL + ALT + F2** キーをもう一度押すと、「tty2」に戻ることができます。コマンドラインインターフェイスには、最後に実行したコマンドが表示されます。

ここでは、TTY からデスクトップ画面に戻る前に以下のように入力します。

```
exit
```

次に **CTRL + ALT + F7** キーを押して、デスクトップ画面に戻ります。TTY からデスクトップ画面に戻る前に「exit」コマンドを実行する理由は、キーボードにアクセスできる人なら、だれでも TTY に切り替えることができるからです。その場合、あなたがログインしているときに、あなたのパスワードを知らない別の人 TTY に切り替えることができます。「exit」コマンドを実行すれば、こうしたことは起こりません。

これで、Raspberry Pi OS のコマンドラインインターフェイスの基礎をマスターしました。

付録D 参考資料



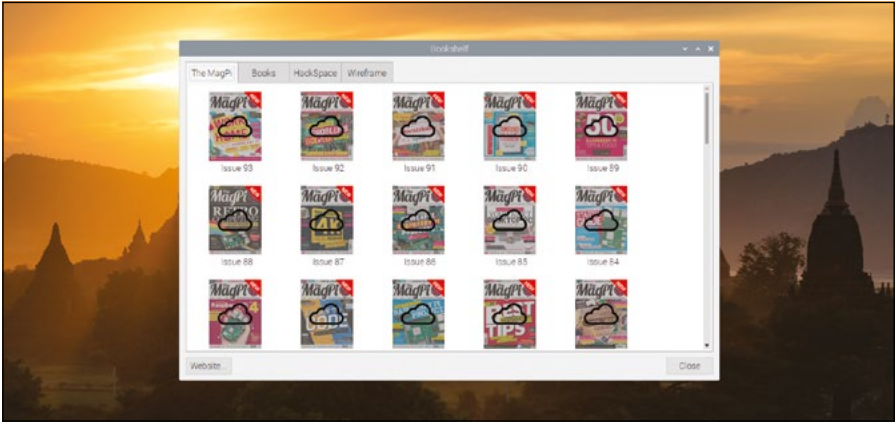
公 式 Raspberry Pi ビギナーズガイド は、Raspberry Pi を初めて使用する人を対象としているため、Raspberry Pi のすべてについて記載されているわけではありません。世界中の多くの人たちが Raspberry Pi を使用して、ゲーム、センシングアプリケーション、ロボット工学、人工知能など、さまざまな分野でプロジェクト開発を行っています。こうした事例を参考にすると、いろいろなヒントが見つかります。

この付録では、Raspberry Pi に関するさまざまな参考資料を紹介します。これらの資料を参照して、このビギナーズガイド で学習した知識とスキルをさらに伸ばしてください。

Bookshelf

▶ Raspberry メニューで「Help」>「Bookshelf」を選択

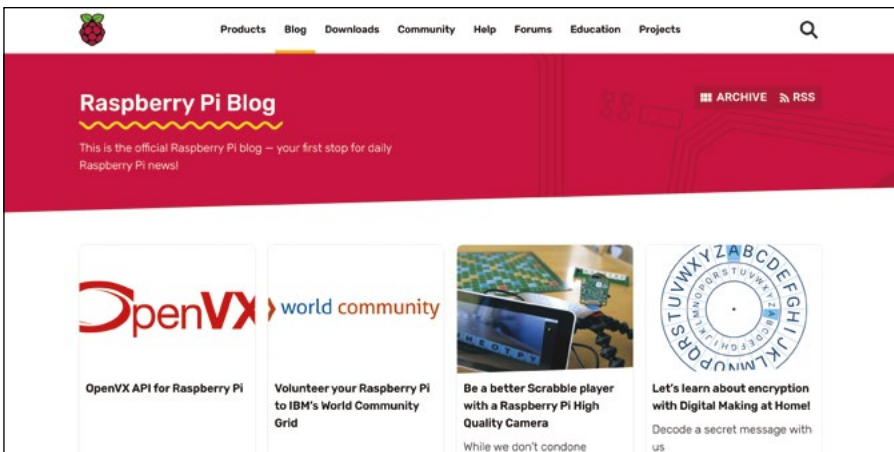
Bookshelf は、Raspberry Pi OS に付属しているアプリケーションです。Raspberry Pi Press が発行する出版物のデジタル版を Bookshelf からダウンロードして読むことができます（この Raspberry Pi ビギナーズガイド も、Bookshelf からダウンロードすることができます）。raspberrypi メニューアイコンをクリックし、「Help」>「Bookshelf」を選択するだけで、さまざまな出版物を表示することができます。また、すべて無料でダウンロードすることができます。



Raspberry Pi のブログ

▶ rpf.io/blog

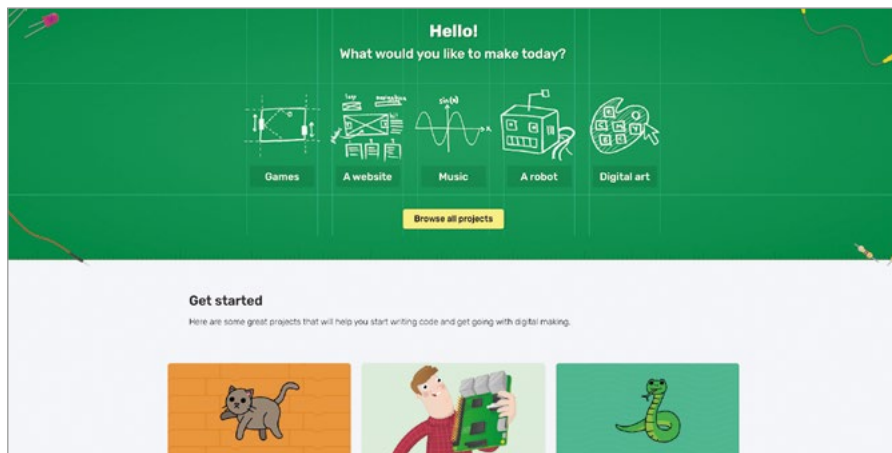
Raspberry Pi に関する最新情報を確認する場合は、最初に公式ブログをチェックすることをお勧めします。公式ブログでは、新しくリリースされたハードウェア、学習教材、お勧めのプロジェクト、キャンペーンなど、さまざまな情報が公開されています。公式ブログで Raspberry Pi の最新情報をチェックしてください。



Raspberry Pi のプロジェクト

▶ rpf.io/projects

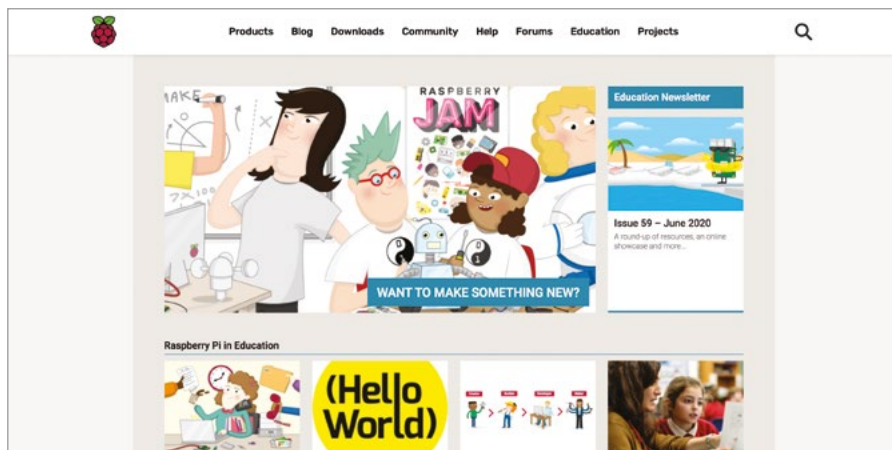
Raspberry Pi の公式プロジェクトサイトでは、ゲーム開発、作曲、Web サイト制作、Raspberry Pi をベースとしたロボット開発など、さまざまな分野のプロジェクト開発を詳しく確認することができます。多くのプロジェクトは、多言語対応になっています。まったくの初心者から上級者まで、あらゆるレベルに対応したプロジェクトが揃っています。



Raspberry Pi の学習サイト

▶ rpf.io/education

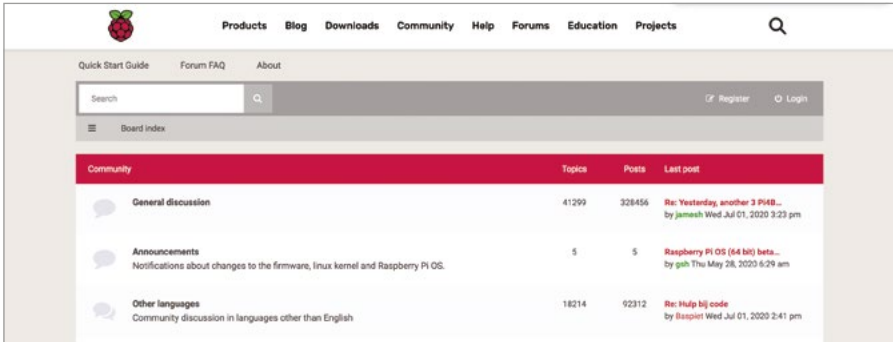
Raspberry Pi の公式学習サイトには、Raspberry Pi に関するニュースレター、オンライントレーニング、各種プロジェクトなど、Raspberry Pi を学習するためのさまざまなコンテンツが用意されています。また、Picademy トレーニングプログラム、Code Club と CoderDojo のボランティアが管理するコーディングプログラム、グローバルな Raspberry Jam イベントへのリンクも用意されています。



Raspberry Pi フォーラム

▶ rpf.io/forums

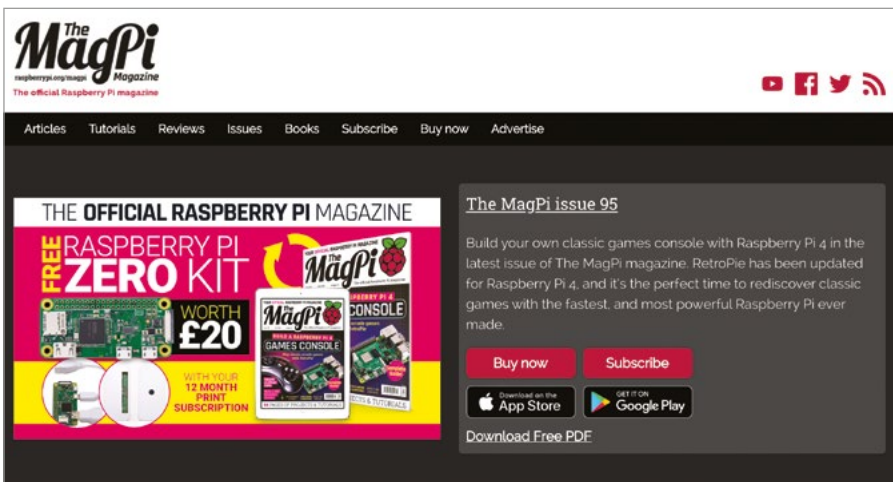
Raspberry Pi フォーラムは、多くの Raspberry Pi ユーザーが集まってチャットをするための場所です。初心者が直面する問題から高度な技術に関する話題まで、さまざまなテーマでチャットが行われています。チャットで雑談をすることもあり、気軽に参加できます。



雑誌「The MagPi」

▶ magpi.cc

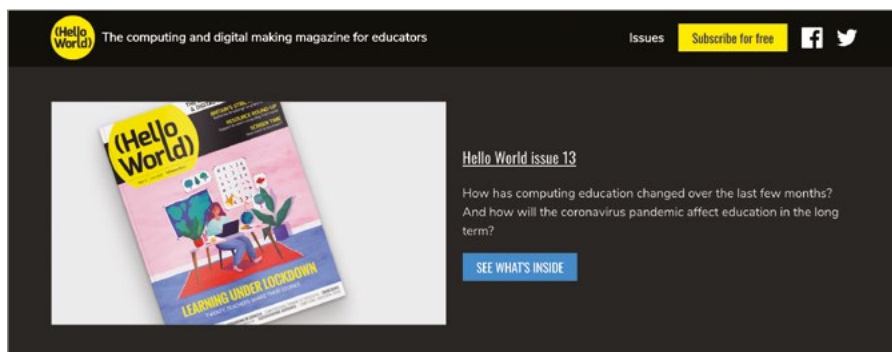
「The MagPi」は、Raspberry Pi の公式月刊誌です。この雑誌では、チュートリアル、ガイド情報、レビュー、ニュースなど、Raspberry Pi に関するさまざまな情報を紹介しています。世界中の多くの Raspberry Pi ユーザーが、この雑誌の制作に協力しています。この雑誌は、多くの書店やスーパーマーケットで購入することができます。また、クリエイティブコモンズライセンスに従い、デジタル版を無料でダウンロードすることもできます。The MagPi 誌だけでなく、さまざまなトピックに関する雑誌や書籍も発行されています。これらの雑誌や書籍は、印刷物として購入することも、無料のデジタル版としてダウンロードすることもできます。



雑誌「Hello World」

▶ helloworld.cc

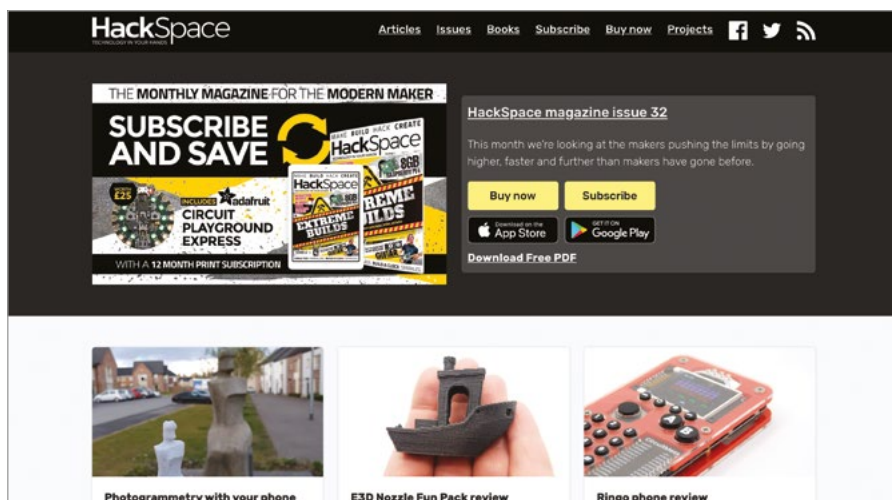
年に3回発行される「Hello World」誌は、イギリス在住の教師、ボランティア、図書館職員のための無料の雑誌です。ただし、クリエイティブコモンズライセンスに従い、だれでも無料のデジタル版をダウンロードすることができます。また、印刷物として購入することもできます。



雑誌「HackSpace」

▶ hsmag.cc

HackSpace は、The MagPi 誌よりも幅広い読者層を対象とした雑誌です。ハードウェアとソフトウェアのレビュー、チュートリアル、インタビューなど、メーカー向けのさまざまな情報が紹介されています。Raspberry Pi に関する知識をベースとして、さらに多くの知識を学習したいと考えているならば、HackSpace 誌を読んでみることをお勧めします。この雑誌は、書店やスーパーマーケットで購入することも、無料のデジタル版をダウンロードすることもできます。



付録 E

Raspberry Pi 構成ツール



Raspberry Pi Configuration ツールは、プログラム上のインターフェイス設定やネットワーク上での管理設定など、Raspberry Pi でさまざまな設定を行うための強力なパッケージです。ただし、初心者にとっては少し難しい設定もあるため、この付録では、それぞれの設定についてその目的を含めて順に説明します。

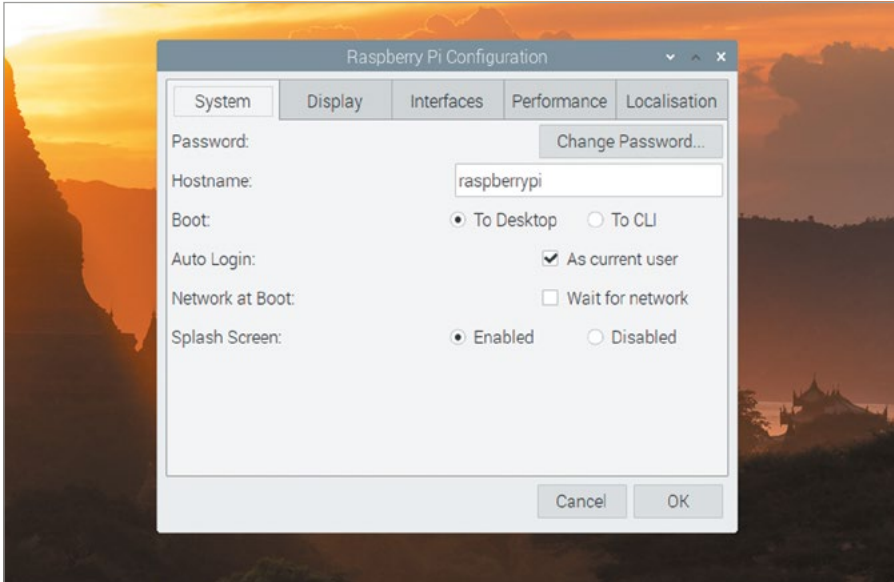
Raspberry Pi Configuration ツールは、raspberrypi アイコンメニューの「設定」カテゴリから起動します。**raspi-config** コマンドを使用して、コマンドラインインターフェイス (ターミナルアプリケーション) から起動することもできます。コマンドラインから起動する場合と、メニュー画面から起動する場合のレイアウトは異なります (それぞれ、別のカテゴリにオプションが表示されます)。この付録では、メニュー画面から起動する場合について説明します。

注意

どうしても変更する必要がある場合を除き、Raspberry Pi Configuration ツールの設定はそのままにしておくことをお勧めします。オーディオ HAT やカメラモジュールなど、新しいハードウェアを Raspberry Pi に追加する場合は、画面に表示される説明に従って設定を変更してください。それ以外の場合は、通常、デフォルト設定をそのまま使用してください。

「システム」タブ

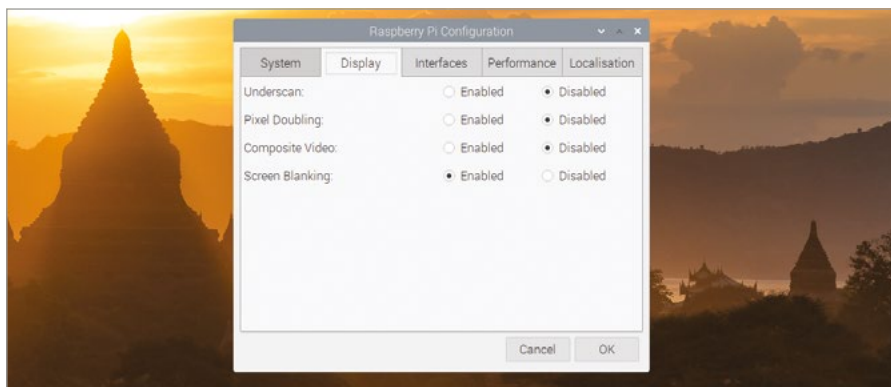
「システム」タブには、Raspberry Pi OS のさまざまなシステム設定を制御するためのオプションが表示されます。



- **パスワード:**現在のユーザーアカウントの新しいパスワードを設定する場合は、「パスワードを変更」ボタンをクリックします。デフォルトのユーザーアカウント名は「pi」です。
- **ホスト名:**これは、ネットワーク上の Raspberry Pi の識別名です。同じネットワーク上で複数の Raspberry Pi を使用する場合は、互いに重複しないホスト名を付ける必要があります。
- **ブート:**これを「デスクトップ」に設定すると（これがデフォルト値です）、Raspberry Pi OS のデスクトップ画面が表示され、「CLI」に設定すると、「付録 C: コマンドラインインターフェイス」で説明したコマンドラインインターフェイス画面が表示されます。
- **自動ログイン:**「現在のユーザーとしてログインする」を選択すると（デフォルトで選択されています）、ユーザー名とパスワードを入力しなくても、Raspberry Pi OS のデスクトップ画面が自動的に表示されます。
- **ネットワークブート:**「Wait for network」を選択すると、ネットワークに接続されるまで Raspberry Pi OS は起動しません。
- **スプラッシュ画面:**これを「有効」に設定すると（これがデフォルト設定です）、Raspberry Pi のブートメッセージの代わりに、グラフィカルなスプラッシュ画面が表示されます。

「Display」タブ

「Display」タブには、画面の表示方法に関する設定オプションが表示されます。



■ **オーバースキャン:**これは、Raspberry Pi のビデオ出力の周囲に黒い枠を表示するかどうかを指定するためのオプションです (多くのテレビ画面のフレームを補正するためのオプション)。黒枠を表示せずにディスプレイいっぱいにデスクトップ画面を表示する場合は「無効」を選択し、黒枠を表示したままにする場合は「有効」を選択します。

■ **Pixel Doubling:**サイズの小さな高解像度ディスプレイを使用する場合は、この「Pixel Doubling」オプションを有効にすると、画面上の表示が大きくなって見やすくなります。

■ **Composite Video:**これは、チップ - リング - リング - スリーブ (TRRS) アダプターとともにコンポジットオーディオビデオ (コンポジット AV) 端子を使用する場合に、その端子で使用可能な複合ビデオ出力を制御するためのオプションです。HDMI の代わりにコンポジットビデオ出力を使用する場合は、このオプションを「有効」に設定し、使用しない場合は「無効」のままにしてください。

■ **Screen Blanking:**これは、スクリーンブランキング機能 (数分後にディスプレイをオフにする機能) の有効と無効を切り替えるためのオプションです。

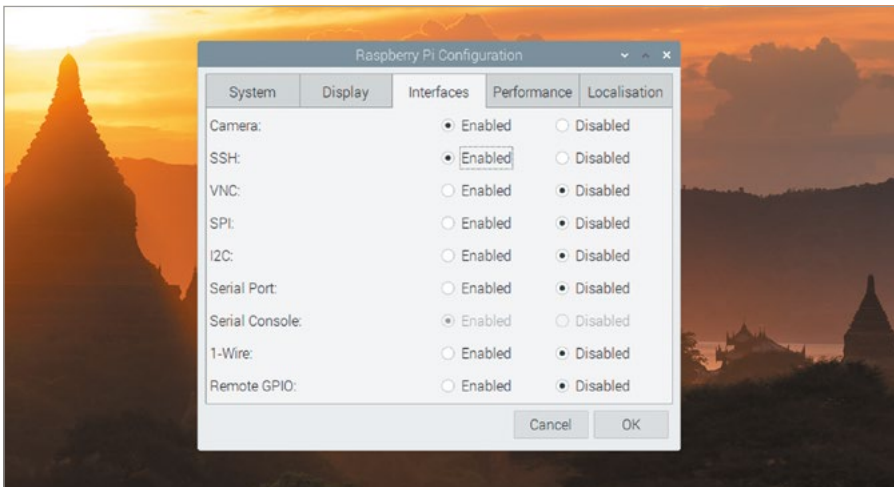
「インターフェイス」タブ

「インターフェイス」タブには、Raspberry Pi のハードウェアインターフェイスを制御するための設定オプションが表示されます。

■ **カメラ:**これは、Raspberry Pi のカメラモジュールで使用するカメラシリアルインターフェイス (CSI) の有効と無効を切り替えるためのオプションです。

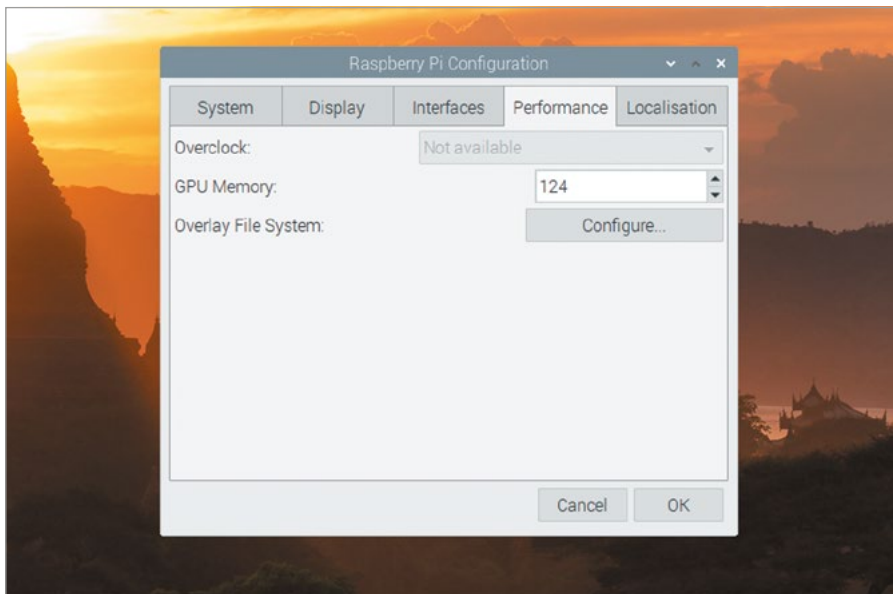
■ **SSH:**これは、セキュアシェル (SSH) インターフェイスの有効と無効を切り替えるためのオプションです。SSH クライアントで SSH インターフェイスを使用すると、同じネットワーク上の別のコンピューターから Raspberry Pi のコマンドラインインターフェイスを起動することができます。

- **VNC:** これは、仮想ネットワークコンピューティング (VNC) インターフェイスの有効と無効を切り替えるためのオプションです。VNC クライアントで VNC インターフェイスを使用すると、同じネットワーク上の別のコンピューターから Raspberry Pi のデスクトップ画面を表示することができます。
- **SPI:** これは、シリアルペリフェラルインターフェイス (SPI) の有効と無効を切り替えるためのオプションです。このインターフェイスにより、GPIO ピンに接続される特定のハードウェアアドオンを制御することができます。
- **I2C:** これは、Inter-Integrated Circuit (I²C) インターフェイスの有効と無効を切り替えるためのオプションです。このインターフェイスにより、GPIO ピンに接続される特定のハードウェアアドオンを制御することができます。
- **Serial Port:** これは、GPIO ピンで使用する Raspberry Pi のシリアルポートの有効と無効を切り替えるためのオプションです。
- **Serial Console:** これは、シリアルコンソールの有効と無効を切り替えるためのオプションです。シリアルコンソールとは、シリアルポート上で使用するコマンドラインインターフェイスのことです。上記の「Serial Port」オプションが「有効」になっている場合に、このオプションを使用することができます。
- **1-Wire:** これは、1-Wire インターフェイスの有効と無効を切り替えるためのオプションです。このインターフェイスにより、GPIO ピンに接続される特定のハードウェアアドオンを制御することができます。
- **リモート GPIO:** これは、ネットワークサービスの有効と無効を切り替えるためのオプションです。GPIO Zero ライブラリーでこのネットワークサービスを使用すると、同じネットワーク上の別のコンピューターから Raspberry Pi の GPIO ピンを制御することができます。リモート GPIO の詳細については、gpiozero.readthedocs.io を参照してください。



「パフォーマンス」タブ

「パフォーマンス」タブには、メモリーの使用量と Raspberry Pi のプロセッサの処理速度を制御するための設定オプションが表示されます。



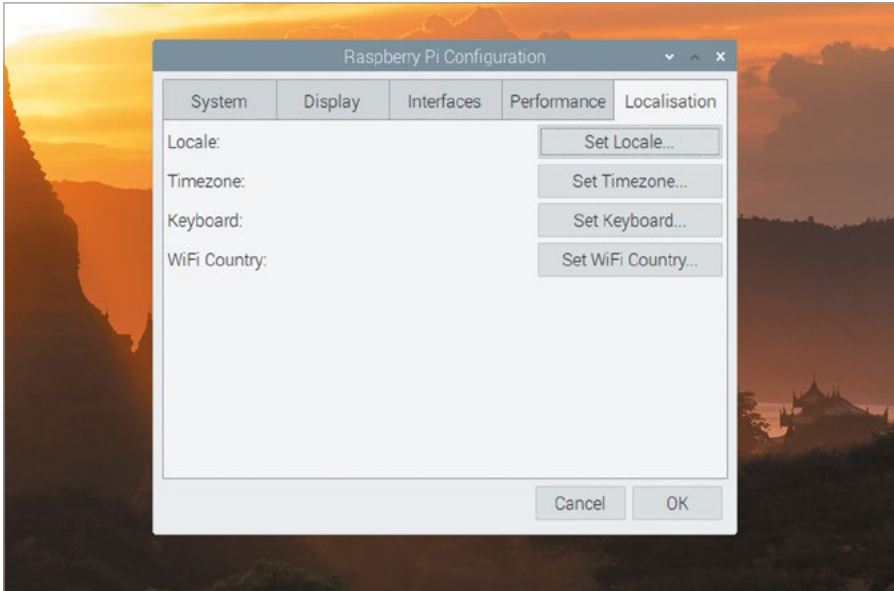
■ **オーバークロック(O):** Raspberry Pi のパフォーマンスを向上させるための設定を選択することができます。ただし、パフォーマンスを向上させると、電力消費量が増加したり、Raspberry Pi 本体で熱が発生したり、Raspberry Pi の全体的な製品寿命が短くなったりします。このオプションは、すべての Raspberry Pi モデルで使用できるわけではありません。

■ **GPU メモリー(G):** Raspberry Pi のグラフィックプロセッサで使用するメモリーの量を指定することができます。デフォルト値よりも大きな値を指定すると、複雑な 3D レンダリング処理と一般的な GPU (GPGPU) タスクのパフォーマンスが向上します (ただし、Raspberry Pi OS で使用できるメモリーの量が少なくなります)。デフォルト値よりも小さな値を指定すると、メモリーの使用量が大きいタスクのパフォーマンスが向上します (ただし、3D レンダリング処理、カメラの操作、ビデオの再生機能のパフォーマンスが低下します。場合によっては、これらが機能しなくなることもあります)。

■ **Overlay File System:** これは、mircoSD カードではなく仮想 RAM ディスクだけに変更内容が書き込まれるように、Raspberry Pi のファイルシステムをロックするためのオプションです。これにより、Raspberry Pi を再起動するたびに、mircoSD カードをクリーンな状態に戻すことができます。

「ローカリゼーション」タブ

「ローカリゼーション」タブには、キーボードのレイアウトに関する設定オプションなど、Raspberry Pi を使用する地域に関する設定オプションが表示されます。



- **ロケール:** ロケールに関するシステム設定（言語、国、文字セットなど）を選択することができます。ここで言語を変更した場合、翻訳の対象となるアプリケーションで表示される言語だけが変更されることに注意してください。
- **タイムゾーン:** 該当する地域のタイムゾーンを選択することができます（最初に地域を選択し、次に都市を選択します）。Raspberry Pi がネットワークに接続されているにもかかわらず、間違った時刻が表示される場合は、正しくないタイムゾーンが選択されています。
- **キーボード:** キーボードの種類、言語、レイアウトを選択することができます。キーボードで間違った文字や記号が入力される場合は、ここで修正することができます。
- **無線 LAN の国:** WiFi を使用する国を指定することができます。必ず、Raspberry Pi を使用する国を指定してください。間違った国を選択すると、近くの WiFi アクセスポイントに接続できなくなる場合があります。また、放送法に違反する可能性があります。ここで国を指定してから、WiFi を使用するようにしてください。

付録 F

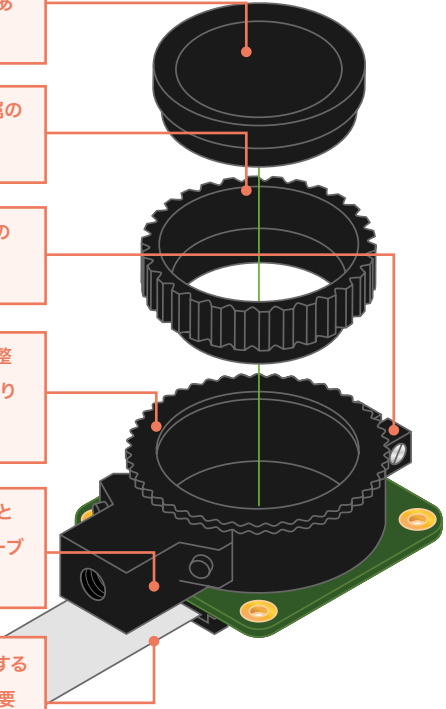
High Quality Camera をセットアップしてみよう

High Quality Camera (HQ カメラ) を使用すると、標準のカメラモジュールよりも高い解像度で画像をキャプチャすることができます。標準のカメラモジュールにはレンズが付属していますが、HQ カメラには付属していません。そのため、レンズを購入する必要があります。HQ カメラでは、標準の C マウントレンズと CS マウントレンズを使用することができます。ここでは、6mm のレンズと 16mm のレンズの取り付け方法について説明します。

6mm の CS マウントレンズ

HQ カメラでは、低価格の 6mm レンズを使用することができます。このレンズは、基本的な写真撮影に適しています。また、非常に近い距離で被写体にフォーカスを合わせることができるため、小さな被写体を撮影する場合にも適しています。





カメラセンサーはほごりの影響を受けやすいため、レンズを取り外す場合は、このダストキャップを取り付ける必要があります

C マウントレンズを HQ カメラに取り付ける場合は、付属の C-CS アダプターを使用してください

このネジを使用して、バックフォーカス調整リングを所定の位置に固定します

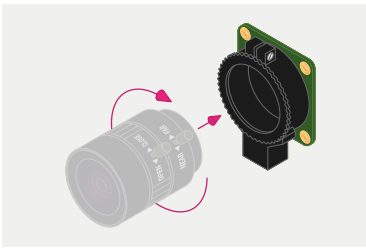
このリングを使用して、固定焦点レンズのフォーカスを調整したり、焦点調整可能レンズのフォーカス範囲を変更したりすることができます

この部分を使用して、カメラを標準の三脚に取り付けることができます (カメラを三脚に取り付けるときに、リボンケーブルが破損しないように注意してください)

HQ カメラには、Raspberry Pi のカメラポートに接続するための 20cm のリボンケーブルが付属していますが、必要に応じてより長いケーブルを使用することもできます

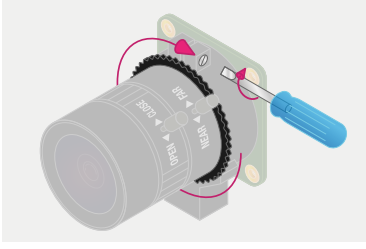
01 レンズを取り付ける

6mm レンズは CS マウント用のレンズであるため、C-CS アダプターは必要ありません (C マウントレンズを使用する場合は、上の図のように C-CS アダプターを取り付ける必要があります)。CS マウントレンズに C-CS アダプターを取り付けると、焦点を正しく合わせることができなくなります。バックフォーカス調整リングに完全に入るまで、レンズを時計回りに回転させます。



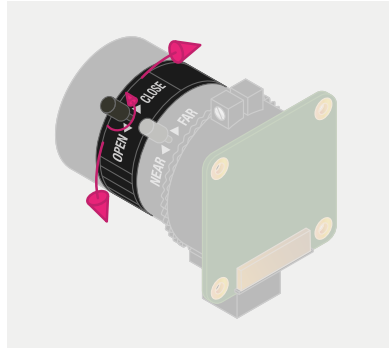
02 バックフォーカス調整リングと固定ネジ

バックフォーカスの距離ができるだけ短くなるように、バックフォーカス調整リングをいっぱいまでねじ込む必要があります。付属のバックフォーカス固定ネジを使用して、バックフォーカス調整リングを固定してください。



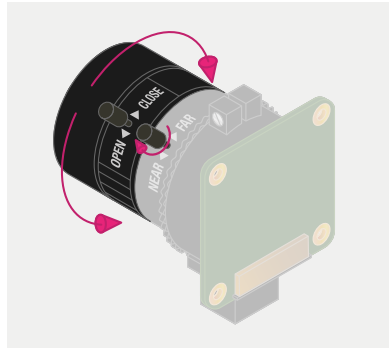
03 絞り

絞りを調整するには、レンズを自分の身体と反対側に向けてカメラを持ちます。カメラから最も遠い外側のリングをしっかりと持ちながら、中央のリング（絞り）を回します。画像の明るさを下げる場合は、絞りを時計回りに回し、画像の明るさを上げる場合は、絞りを反時計回りに回します。明るさの調整が完了したら、レンズの側面にあるネジを締めて絞りを固定します。



04 フォーカス

最初に、「NEAR ◀▶ FAR」というラベルが付いている内側のフォーカスリングをネジで固定します。次に、レンズを自分の身体と反対側に向けてカメラを持ちます。レンズの外側の2つのリングを持ち、画像の焦点が合うまで2つのリングを時計回りに4〜5回ほど回します。近くの被写体にフォーカスを合わせる場合は、外側の2つのリングを時計回りに回し、遠くの被写体にフォーカスを合わせる場合は、反時計回りに回します。必要な場合は、調整後のフォーカスに合わせて、もう一度絞りを調整してください。



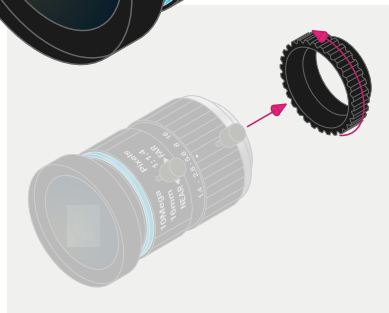
16mm の C マウントレンズ

16 mm レンズを使用すると、6 mm レンズよりも高品質の画像を撮影することができます。16 mm レンズは 6 mm レンズよりも画角が狭いため、遠くの被写体を撮影する場合に適しています。



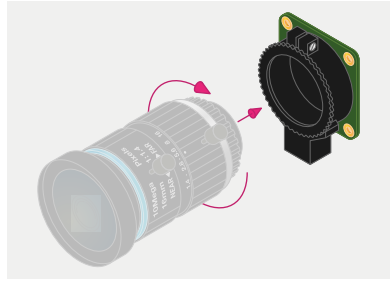
01 C-CS アダプターを取り付ける

HQ カメラに付属している C-CS アダプターを 16 mm レンズに取り付けます。16 mm レンズは C マウント用のレンズで、6 mm レンズよりもバックフォーカスの距離が長いので、アダプターが必要になります。



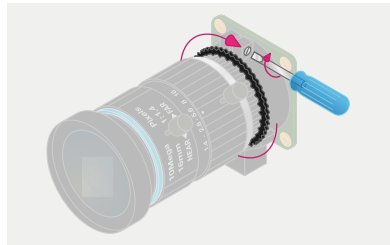
02 レンズをカメラに取り付ける

バックフォーカス調整リングに完全に入るまで、16mm レンズと C-CS アダプターを時計回りに回転させます。



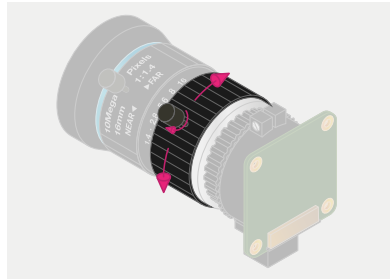
03 バックフォーカス調整リングと固定ネジ

バックフォーカス調整リングをいっぱいまでねじ込みます。付属のバックフォーカス固定ネジを使用して、バックフォーカス調整リングを固定してください。



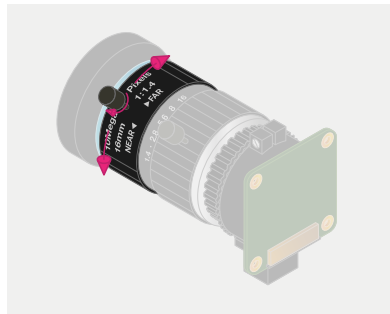
04 絞り

絞りを調整するには、レンズを自分の身体と反対側に向けてカメラを持ちます。カメラをしっかりと持ったまま、カメラに最も近いリング（最も手前のリング）を回します。画像の明るさを下げる場合は、絞りを時計回りに回し、画像の明るさを上げる場合は、絞りを反時計回りに回します。明るさの調整が完了したら、レンズの側面にあるネジを締めて絞りを固定します。



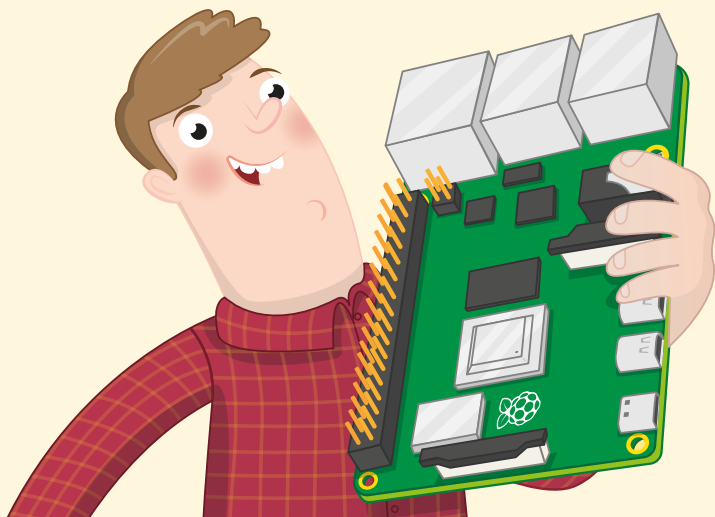
05 フォーカス

フォーカスを調整するには、レンズを自分の身体と反対側に向けてカメラを持ちます。近くの被写体にフォーカスを合わせる場合は、「NEAR ◀ ▶ FAR」というラベルが付いているフォーカスリングを反時計回りに回し、遠くの被写体にフォーカスを合わせる場合は、時計回りに回します。必要な場合は、調整後のフォーカスに合わせて、もう一度絞りを調整してください。



付録 G

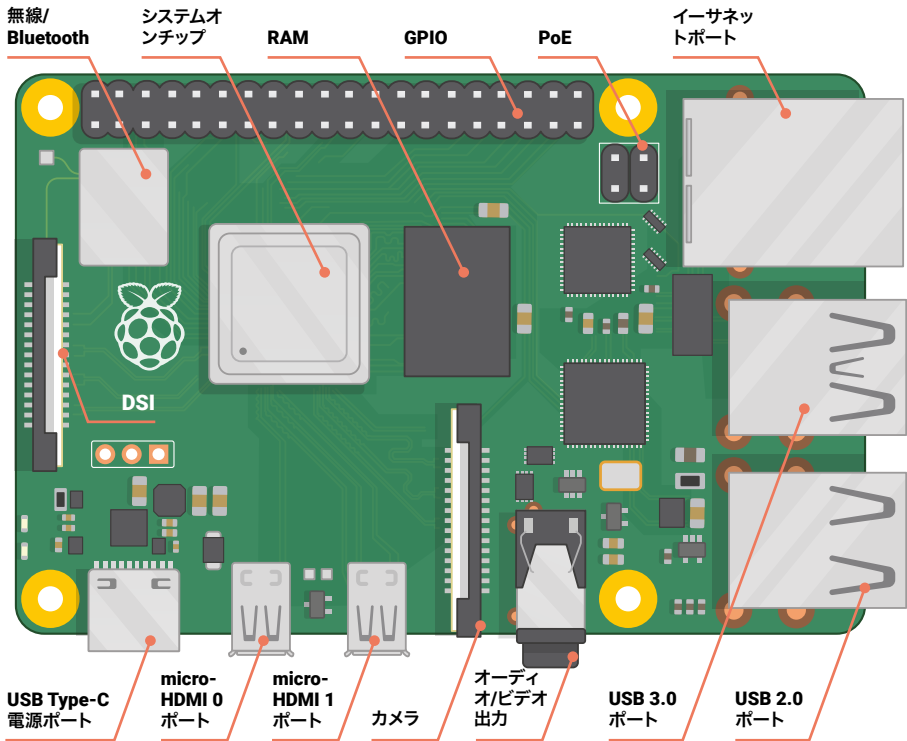
Raspberry Pi の仕様



コンピュータを構成するさまざまなコンポーネントと機能のことを「仕様」と総称します。2台のコンピュータの仕様を確認すれば、それぞれのコンピュータを比較することができます。最初は慣れていないため、コンピュータの仕様を見と難しく感じるかもしれません。仕様を詳しく理解しなくても Raspberry Pi を使うことに問題はありますが、技術的な情報に興味を持っている人のために、ここでは Raspberry Pi の仕様について説明します。

Raspberry Pi 4 Model B と Raspberry Pi 400 のシステムオンチップは、Broadcom BCM2711B0 です。Raspberry Pi 4 を近くで見ると、金属カバーの部分に「Broadcom」という文字が印字されていることがわかります。このシステムオンチップは、4 つの 64 ビット ARM Cortex-A72 中央処理装置 (CPU) コアから構成されており、各 CPU コアは 1.5GHz または 1.8GHz (1 秒あたり 15 億サイクルまたは 18 億サイクル) で稼働します。ビデオや 3D レンダリング (ゲームなど) の処理を実行する Broadcom VideoCore VI (6) のグラフィック処理装置 (GPU) は、500MHz (1 秒あたり 5 億サイクル) で稼働します。

Raspberry Pi のシステムオンチップは、2GB、4GB、または 8GB (GB とはギガバイトのことで、1GB は 10 億バイトです) の LPDDR4 (低電力ダブルデータレート 4) RAM (ランダムアクセスメモリー) に接続されます (Raspberry Pi 400 の場合は 4GB の LPDDR4 RAM に接続されます)。LPDDR4 RAM は、3200MHz (1 秒あたり 32 億サイクル) で稼働します。このメモリーは、中央処理装置とグラフィック処理装置間で共有されます。microSD カードスロットでは、最大 512 GB (5,120 億バイト) のストレージデバイスを使用することができます。



イーサネットポートでは、最大 1 ギガビット (1000Mbps、1000-Base-T) 接続がサポートされています。無線接続の場合は、2.4GHz と 5GHz の周波数帯域で稼働する 802.11ac WiFi ネットワーク、Bluetooth 5.0 接続、Bluetooth Low Energy (BLE) 接続がサポートされています。

以下に、Raspberry Pi 4 の仕様を項目別に示します。

- **CPU:**1.5GHz の 64 ビットクアドコア ARM Cortex-A72
- **GPU:**500MHz の VideoCore VI
- **RAM:**1GB、2GB、または 4GB の LPDDR4
- **ネットワーク:** ギガビットイーサネット、デュアルバンド 802.11ac、Bluetooth 5.0、Bluetooth Low Energy
- **オーディオ/ビデオ出力:**3.5mm アナログ AV ジャック、micro-HDMI 2.0 x 2
- **周辺機器の接続:** USB 2.0 ポート x 2、USB 3.0 ポート x 2、カメラシリアルインターフェイス (CSI)、ディスプレイシリアルインターフェイス (DSI)
- **ストレージ:** microSD (最大 512GB)
- **電源:**5 ボルト/3 アンペア (USB Type-C)
- **その他:**40 ピン GPIO ヘッダー、Power over Ethernet に対応 (追加のハードウェアが必要)



Raspberry Pi 400 の仕様を以下に示します。

- **CPU:** 1.8GHz の 64 ビットクアッドコア ARM Cortex-A72
- **GPU:** 500MHz の VideoCore VI
- **RAM:** 4GB の LPDDR4
- **ネットワーク:** ギガビットイーサネット、デュアルバンド 802.11ac、Bluetooth 5.0、Bluetooth Low Energy
- **オーディオ/ビデオ出力:** micro-HDMI 2.0 x 2
- **周辺機器の接続:** USB 2.0 ポート x 1、USB 3.0 ポート x 2
- **ストレージ:** 16GB の microSD (最大 512GB)
- **電源:** 5 ボルト/3 アンペア (USB Type-C)
- **その他:** 40 ピン GPIO ヘッダー

付録 H

Raspberry Pi の 安全性とユーザーガイド



Raspberry Pi

設計・供給元
Raspberry Pi Trading Ltd
Maurice Wilkes Building
Cowley Road
Cambridge
CB4 0DS
UK
www.raspberrypi.org

Raspberry Pi の規制準拠と安全情報

Raspberry Pi 4 Model B
FCC ID:2ABCB-RPI4B
IC ID:20953-RPI4B

Raspberry Pi 400
FCC ID:2ABCB-RPI400
IC ID:20953-RPI400

重要:本体を電源に接続する前に、以下のサイトで安全情報を確認してください:

www.raspberrypi.org/safety



注意:以下のサイトで、がんと生殖障害に関する注意情報を確認してください:

www.P65Warnings.ca.gov.

規制と認証に関する情報は、すべて以下のサイトで参照することができます:

www.raspberrypi.org/compliance



IFETEL:2019LAB-ANCE4957
Raspberry Pi 4 Model B の認証番号



TRA 登録番号
ER73381/19
Raspberry Pi 4 Model B の認証番号



CCA019LP1120T2

Raspberry Pi 4 Model B の認証番号



NTC
承認済みタイプ:
番号: ESD-GEC-1920098C
Raspberry Pi 4 Model B の認証番号



TA-2019/750 APPROVED
Raspberry Pi 4 Model B の認証番号



このガイドに記載されている HDMI、HDMI High-Definition Multimedia Interface、HDMI ロゴは、米国とその他の国における HDMI Licensing Administrator, Inc. の商標または登録商標です。



公式 Raspberry Pi ビギナーズガイド

Raspberry Pi は、小型で高性能な英国製コンピューターです。スマートフォンと同じ技術をベースとして設計されている Raspberry Pi を使用して、コーディングを学習したり、コンピューターの仕組みを調べたり、独自のプロジェクトを作成したりすることができます。このガイドを読めば、Raspberry Pi の操作が非常に簡単であることがわかります。

このガイドでは、以下の項目について説明しています。

- > Raspberry Pi をセットアップしてオペレーティングシステムをインストールし、Raspberry Pi の使用を開始する。
- > プログラミング言語として Scratch 3 と Python を使用し、ステップバイステップガイドを参照しながらプロジェクトのコーディングを開始する。
- > 電子部品を接続し、独自のプロジェクトを作成する。

raspberrypi.org

価格: 10 ポンド



9 781912 047819

