スタートガイド
教育版レゴ® マインドストーム®
EV3 MicroPython

バージョン 1.0.0

目次

1 インストール 2
2 プログラムの作成と実行 8
3 EV3 BRICK – EV3 プログラミングブロック 16
4 EV3 デバイス – EV3 モータとセンサ 20
5 Parameters – Parameters と定数 29
6 ツール–タイミングとデータロギング 37
7 ロボティクス–ロボティクスモジュール 38
8 信号と単位 40
9 ロボットエデュケーター 43
10 カラーソーター 45
11 ロボットアーム H25 49
Python モジュールインデックス 53
索引 54

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

このガイドでは、レゴ® マインドストーム® ev3 ロボット用の MicroPython プログラムの作成を開始する方法について説明します。これには、2 つの手順があります。

- インストール:まず、コンピューターと EV3 ブロックを準備し、必要なツールを揃えてインストールします。また、EV3 ブロックをオン/オフに切り替えたり、画面上のメニューをナビゲートする方法についても学びます。
- プログラムの作成と実行:次に、プログラムを作成して、EV3 ブロックにダウンロードする方法について説明します。また、作成したプログラムを、コンピューターまたは EV3 ブロックから起動する方法についても学習します。

最初のデモプログラムの実行を終えると、サンプルプログラムを試し、独自のプログラムの開発を開始する準備が整います。

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

章1

インストール

このページでは、プログラミングを開始するために必要なすべての情報を収集し、インストールする手順について説明します。

1.1 必要なものは？

開始するには、次のものが必要です。

- Windows 10 または Mac OS コンピューター
- インターネットアクセスと管理者アクセス権限

  これは、インストール時にのみ必要です。後でプログラムを作成して実行する場合、特別なアクセス権限は必要ありません。
- MicroSD カード

  必要なカードの最小容量は 4GB、最大容量は 32GB です。このタイプの microSD カードは microSDHC としても知られています。アプリケーションパフォーマンスクラス A1 のカードの使用をお勧めします。
- コンピューターの microSD カードスロットまたはカードリーダー

  コンピューターに (micro) SD カードスロットがない場合は、外付の USB (マイクロ) SD カードリーダーを使用できます。
- EV3 セットに付属のミニ USB ケーブル

この装置の一般的な構成は図 1.1 に要約されています。

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

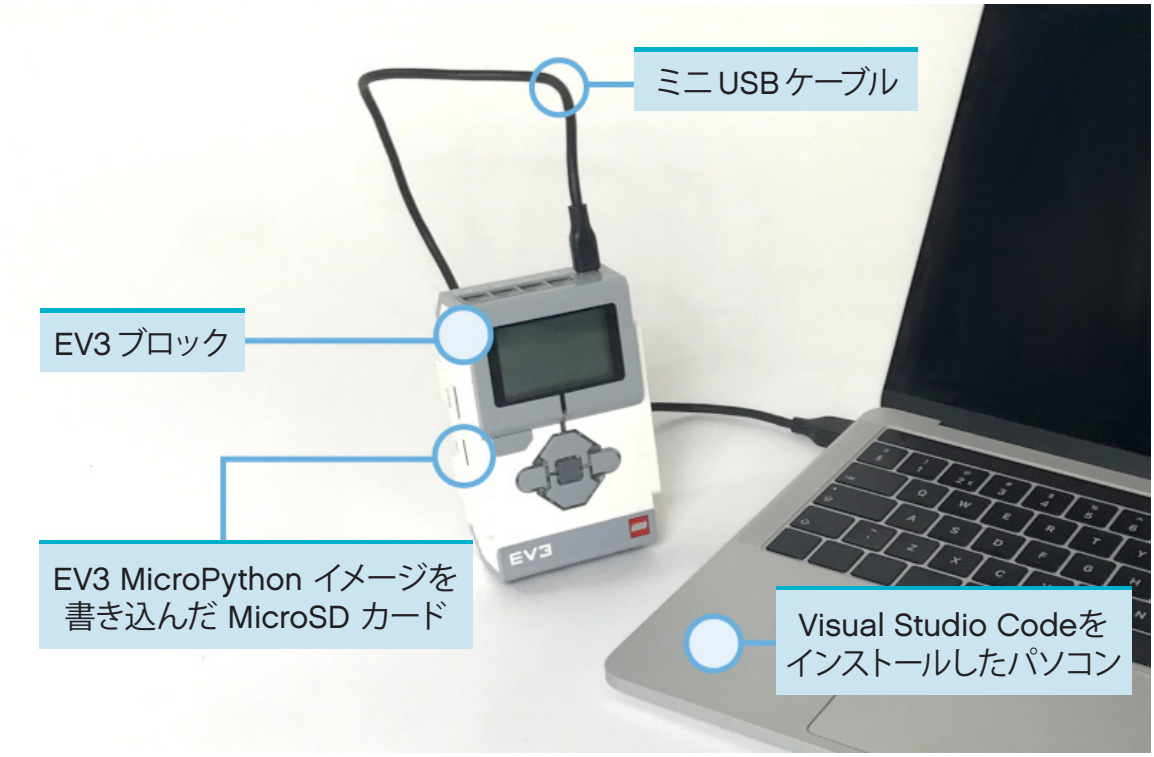


図 1.1:セットアップの概要

1.2 コンピューターの準備

Visual Studio コードを使用して、MicroPython プログラムを記述します。次の手順に従って、このアプリケーションをダウンロードし、インストールし、設定を行います。

1. Visual Studio Code をダウンロードします。
2. 画面の指示に従って、アプリケーションをインストールします。
3. Visual Studio Code を起動します。
4. [拡張機能] タブを開きます。
5. 図 1.2 のとおり、EV3 MicroPython 拡張機能をインストールします。

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

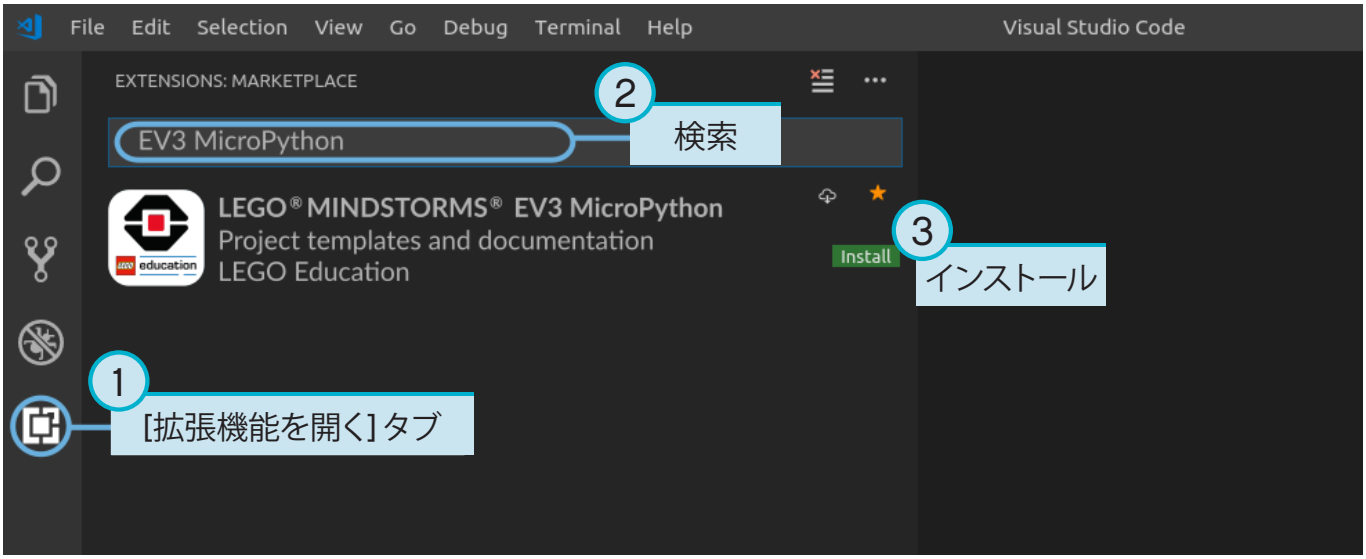


図 1.2: Visual Studio Codeマーケットプレイスから拡張機能をインストールする

1.3 microSD カードの準備

EV3 ブロックで MicroPython プログラムを実行するために必要なツールを、microSD カードにインストールする方法を学習します。

MicroSD カードに、削除したくないファイルが含まれている場合は、先に、保存しておくコンテンツのバックアップを作成してください。以前の MicroPython プログラムをバックアップする方法は、必要に応じて、EV3 のファイル管理を参照してください。

このプロセスでは、以前の MicroPython プログラムを含む、microSD カードのすべてのデータが消去されます。

MicroSD カードに MicroPython ツールをインストールするには:

1. EV3 MicroPython microSD カードイメージをダウンロードして、わかりやすい場所に保存します。このファイルは約 360 MB です。ファイルを解凍する必要はありません。
2. Etcherなどの microSD カードの書き込みツールをダウンロードし、インストールします。
3. MicroSD カードを、コンピューターまたはカードリーダーに挿入します。
4. 書き込みツールを起動し、画面の手順に従って、ダウンロードしたファイルをインストールします。Etcher を使用する場合は、図 1.3 に示すように、以下の手順に従います。
   a. ダウンロードした EV3 MicroPython microSD カードイメージファイルを選択します。
   b. MicroSD カードを選択します。デバイスがmicroSD カードに対応しており、サイズが容量内であることを確認してください。
   c. 上書きプロセスを開始します。これは数分程度かかる場合があります。上書きプロセスが完了するまで、カードを取り出さないでください。

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

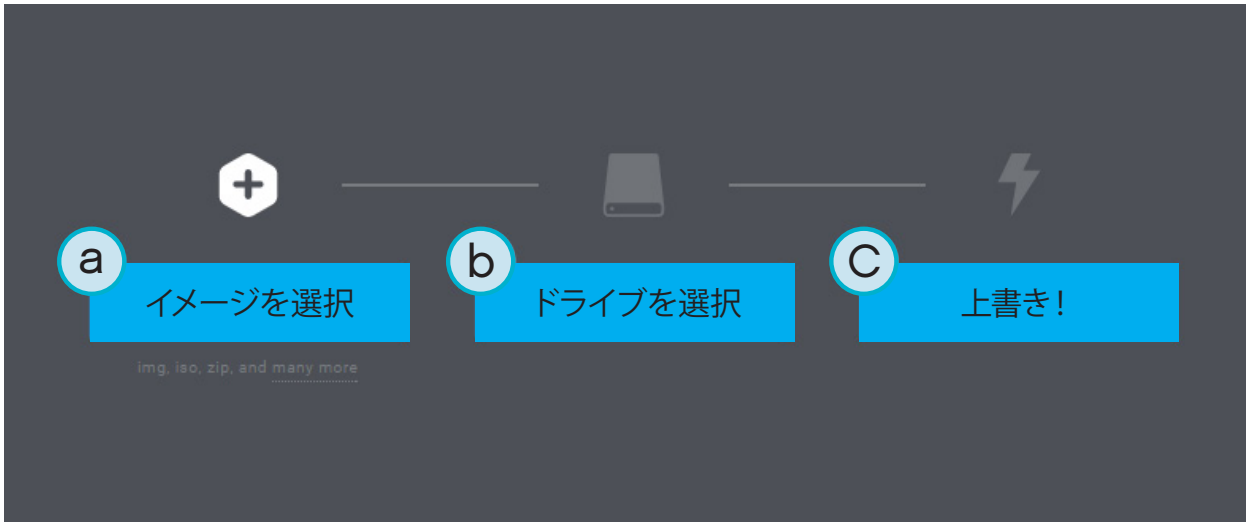


図 1.3: Etcherを使って EV3 MicroPython microSD カードイメージを上書きする

1.4 microSD カードを更新する

MicroSD カードを更新するには、上記のリンクから新しいイメージファイルをダウンロードし、上記の手順で microSD カードに上書きを行います。保存したい MicroPython プログラムは、すべてのバックアップしておいてください。

手順の前に、MicroSD カードの内容を消去する必要はありません。これは、新しいイメージファイルを上書きするときに自動的に行われます。

1.5 EV3 ブロックの使用

EV3 ブロックが起動していないことを確認します。図1.4 のとおり、容易した microSD カードを EV3 ブロックの microSD カードスロットに挿入します。

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

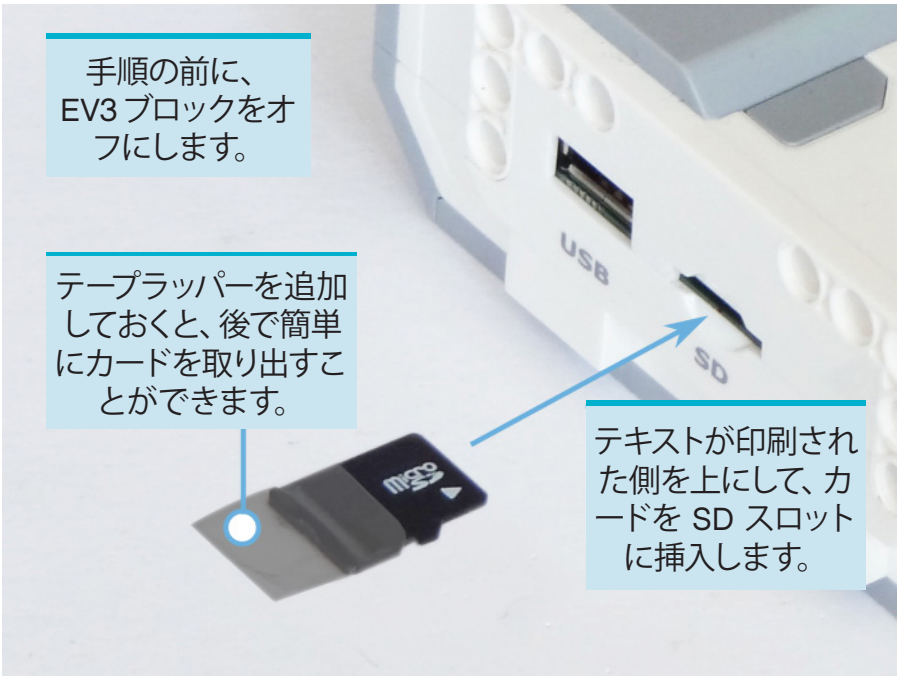


図 1.4: 上書き済みの microSD カードを EV3 ブロックに挿入する

1.5.1 EV3 ブロックのオン・オフを切り替える

濃いグレーの[センター]ボタンを押して、EV3 ブロックをオンにします。

ブートプロセスは数分程度かかることがあります。起動中は、EV3 ブロックのステータスライトがオレンジ色に変わって断続的に点滅し、EV3 画面には多くのテキストが表示されます。ステータスランプが緑に変わると、EV3 ブロックを使用することができるようになっています。

EV3 ブロックをオフにするには、図1.5 に示すように、[戻る] ボタンから [シャットダウン] メニューを開き、[センター] ボタンを使って [電源オフ] を選択します。

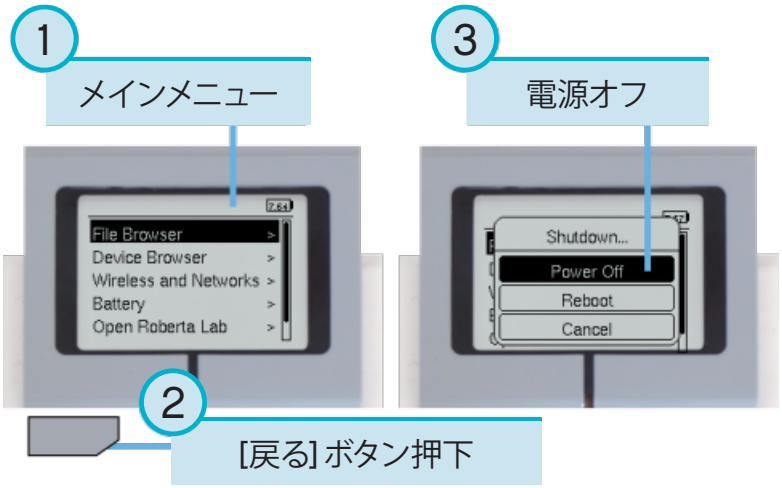


図 1.5: EV3 ブロックをオフにする

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

1.5.2 モータとセンサ値の表示

プログラムを実行していないときには、図1.6に示すように、デバイスのブラウザを使って、モータとセンサの値を表示することができます。

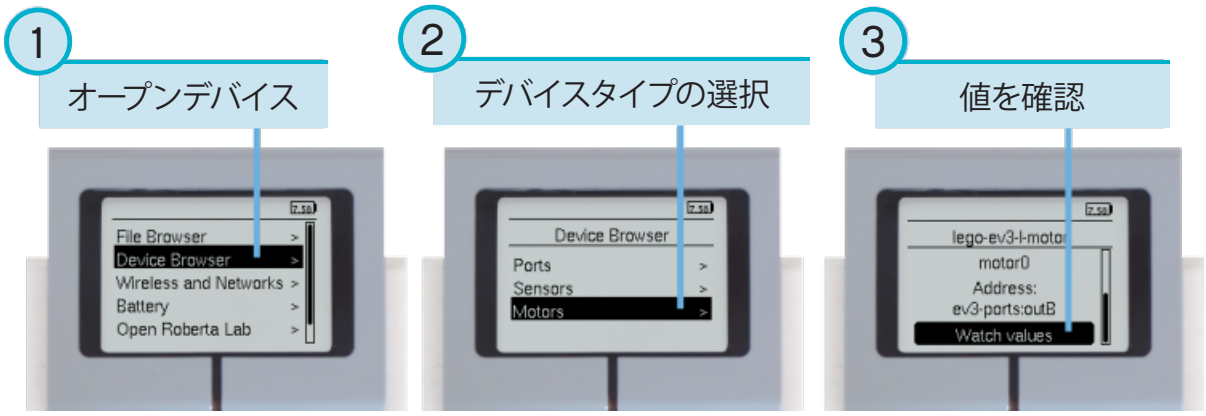


図 1.6:モータとセンサの値の表示

1.5.3 元のファームウェアに戻す

前に使っていたレゴ® ファームウェアやレゴ・プログラムの状態にいつでも戻すことができます。これを行うには:

1. 上の図のように EV3 ブロックをオフにします。
2. 画面とブロックのステータスライトが消えるまで待ちます。
3. MicroSD カードを取り出します。
4. EV3 をオンにします。

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

章2

プログラムの作成と実行

コンピューターと EV3 ブロックのセットアップが完了したら、次に、プログラムの作成を開始します。

プログラムの作成と管理を容易にするため、まず、EV3 ロボットの MicroPython プロジェクトやプログラムが、どのように保存されているかを簡単に見てみましょう。

図 2.1 に示すように、プログラムはプロジェクトフォルダに保存されています。プロジェクトフォルダは、コンピューター上のディレクトリです。このフォルダにはメインプログラム (**main.py**) や、その他オプションとして使うスクリプトやファイルが含まれています。このプロジェクトフォルダとそのすべての内容は、メインプログラムを実行する EV3 ブロックにコピーされます。

このページでは、プロジェクトを作成する方法と、作成したプロジェクトを EV3 ブロックに転送する方法を説明します。

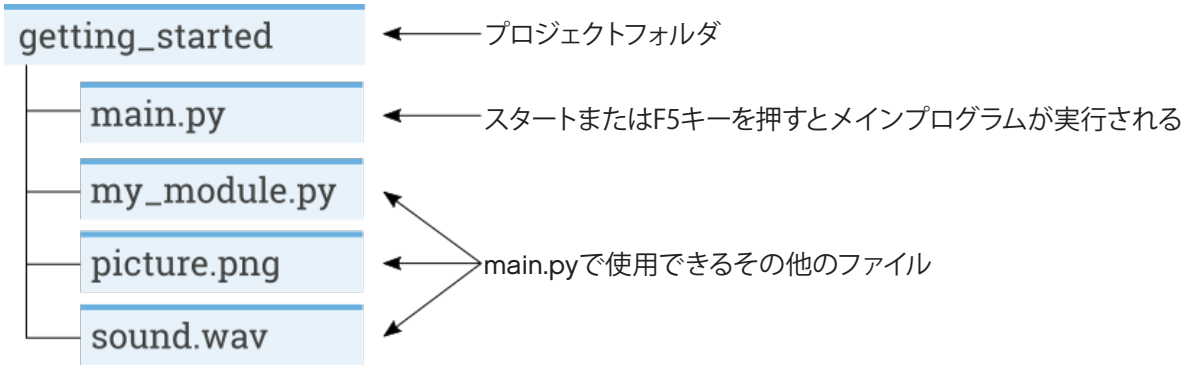


図 2.1: プロジェクトには、main.py と呼ばれるプログラムと、サウンドや MicroPython モジュールなどの、オプションのリソースが含まれています。

2.1 新しいプロジェクトを作成する

新しいプロジェクトを作成するには、図 2.2 に示すように[EV3 MicroPython タブ]を開き、[新規プロジェクトを作成] をクリックします。テキストフィールドが表示されます。このフィールドに、プロジェクト名を入力し、[Enter] キーを押します。プロンプトが表示されたら、このプログラムの場所を選択し、[フォルダの選択]をクリックして確認します。

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

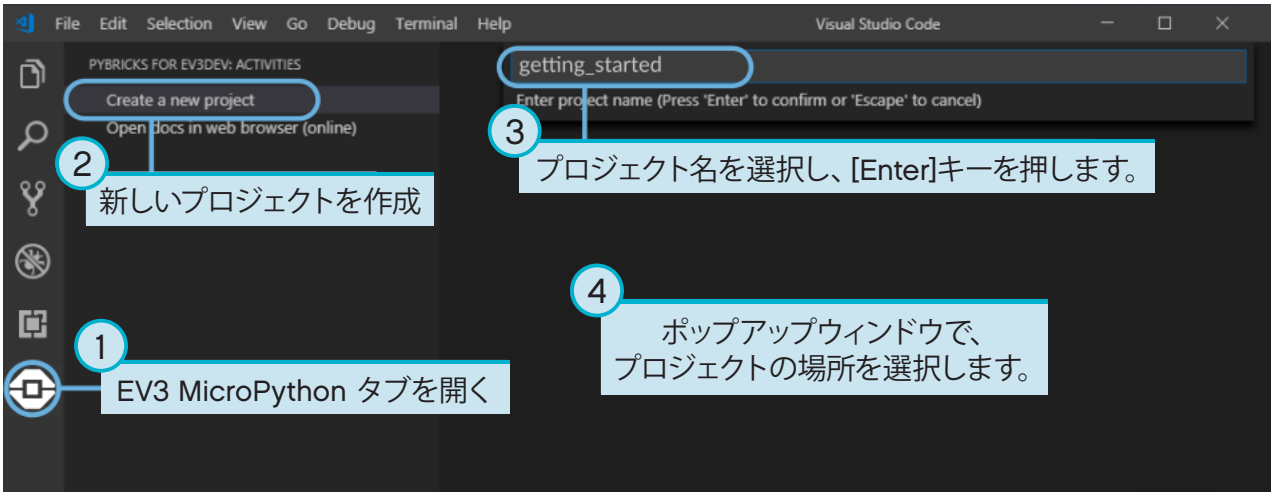


図 2.2: 新しいプロジェクトを作成この例ではプロジェクト名は getting_started となっていますが、
実際に作成するときは好きな名前を選択できます。

新しいプロジェクトを作成した時点で、既にmain.py という名前のファイルが含まれています。このファイルの内容を表示したり変更するには、図 2.3 に示すとおり、ファイルをファイルブラウザで開きます。ここで、プログラムを作成します。

MicroPython プログラミングを初めて使用する場合は、既存のコードはそのままにしておき、コードを追加していくことをお勧めします。

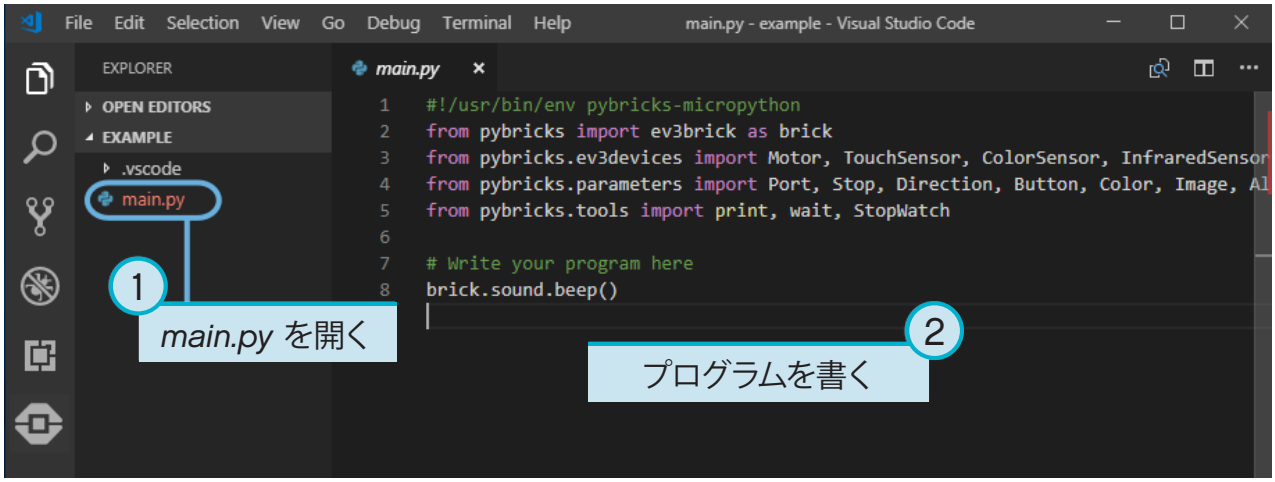


図 2.3: Defaultの main.py プログラムを開きます。

2.2 既存のプロジェクトを開く

以前に作成したプロジェクトを開くには、図2.4に示すように[ファイル]をクリックして[フォルダを開く]をクリックします。次に、以前に作成したプロジェクトフォルダに移動し、[OK]をクリックします。メニューオプションの[最近開いた項目]を使用して、最近使用したプロジェクトを開くこともできます。

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

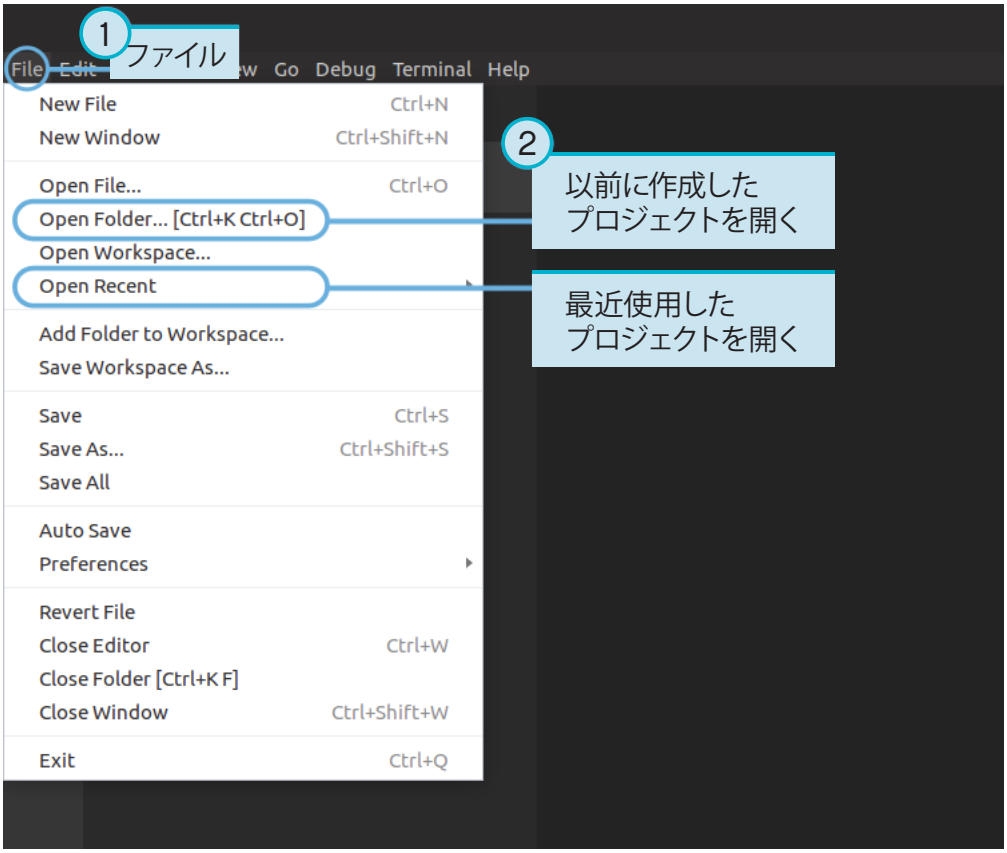


図 2.4: 以前に作成したプロジェクトを開きます。

2.3 Visual Studio コードを使用して EV3 ブロックに接続する

コードを EV3 ブロックに転送できるようにするには、まず、EV3 ブロックを、コンピューターとミニ USB ケーブルで接続し、Visual Studio コードで接続設定を行う必要があります。これを行うには:

- EV3 ブロックを起動します。
- EV3 ブロックをコンピューターとミニ USB ケーブルで接続します。
- 図 2.5 に示すように、USB 接続を設定します。

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

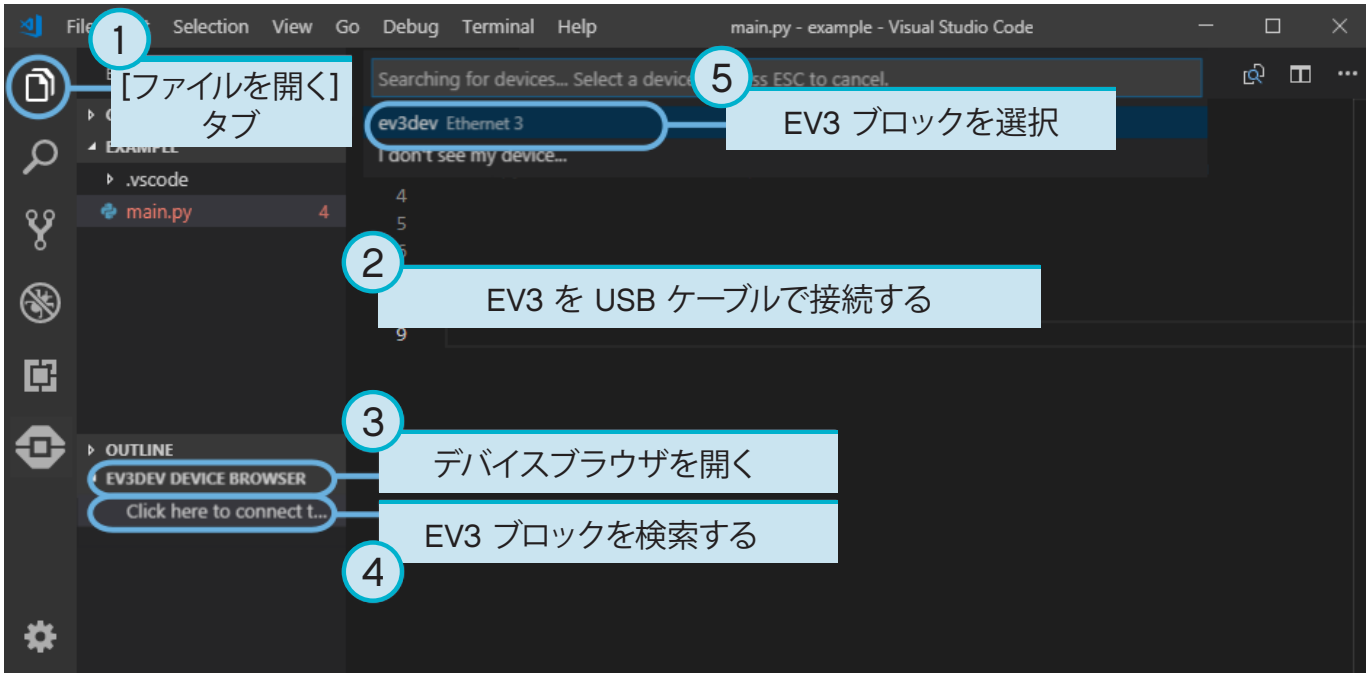


図 2.5: コンピューターと EV3 ブロック間の USB 接続を設定する。

2.4 プログラムのダウンロードと実行

F5 キーを押すと、プログラムを実行できます。または、図2.6に示すように、[デバッグ] タブに移動し、緑の [スタート] 矢印をクリックして手動で開始することもできます。

プログラムが起動したら、必要に応じて、ポップアップツールバーからプログラムを停止することができます。
また、EV3 ブロックの [戻る] ボタンで、いつでもプログラムを停止することができます。

プログラムが *print* コマンドを使用して出力を生成する場合は、出力ウィンドウに表示されます。

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

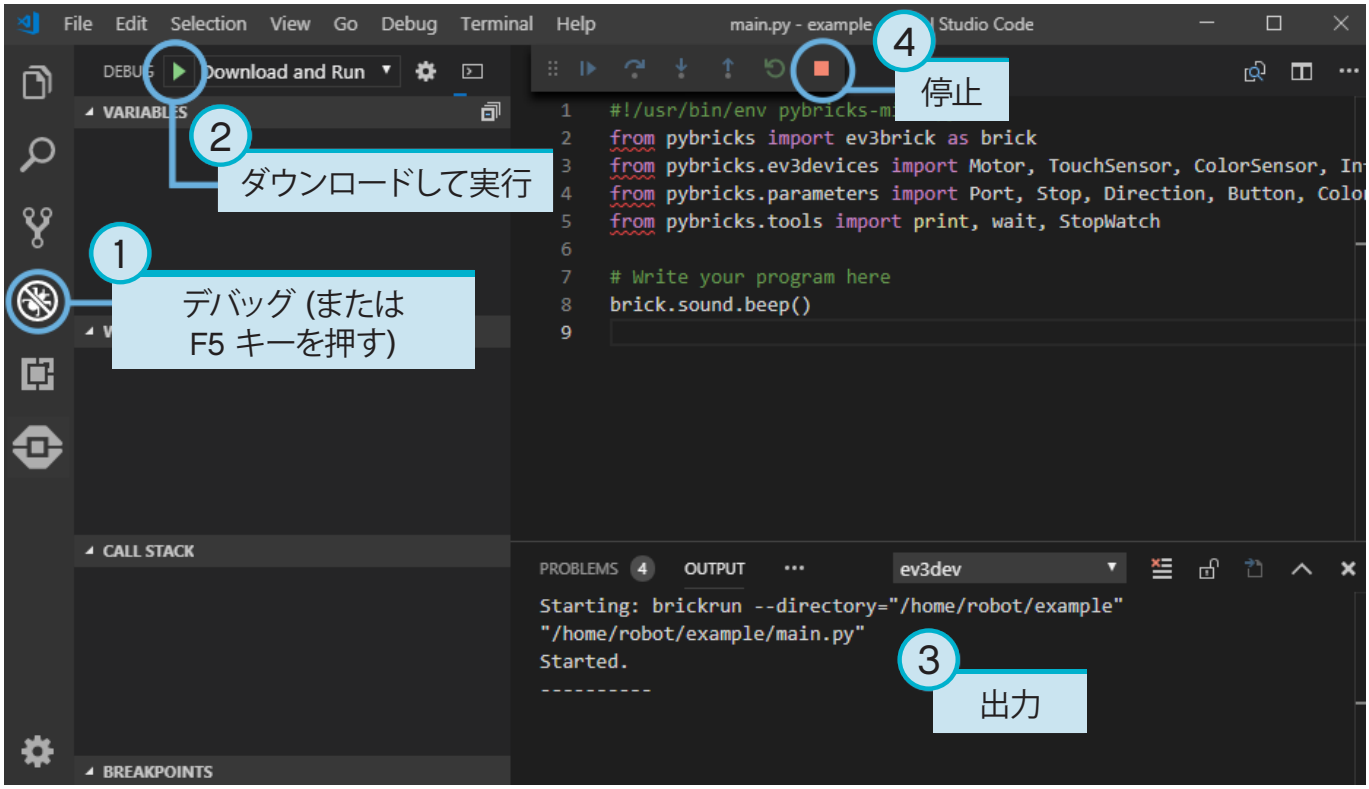


図 2.6: プログラムの実行。

2.5 サンプルプログラムの拡張

基本的なコードテンプレートが実行できたら、プログラムを展開し、モータを動かせるようになります。
図 2.7 に示すように、まず、Lモータを EV3 ブロックのポート B に接続します。

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

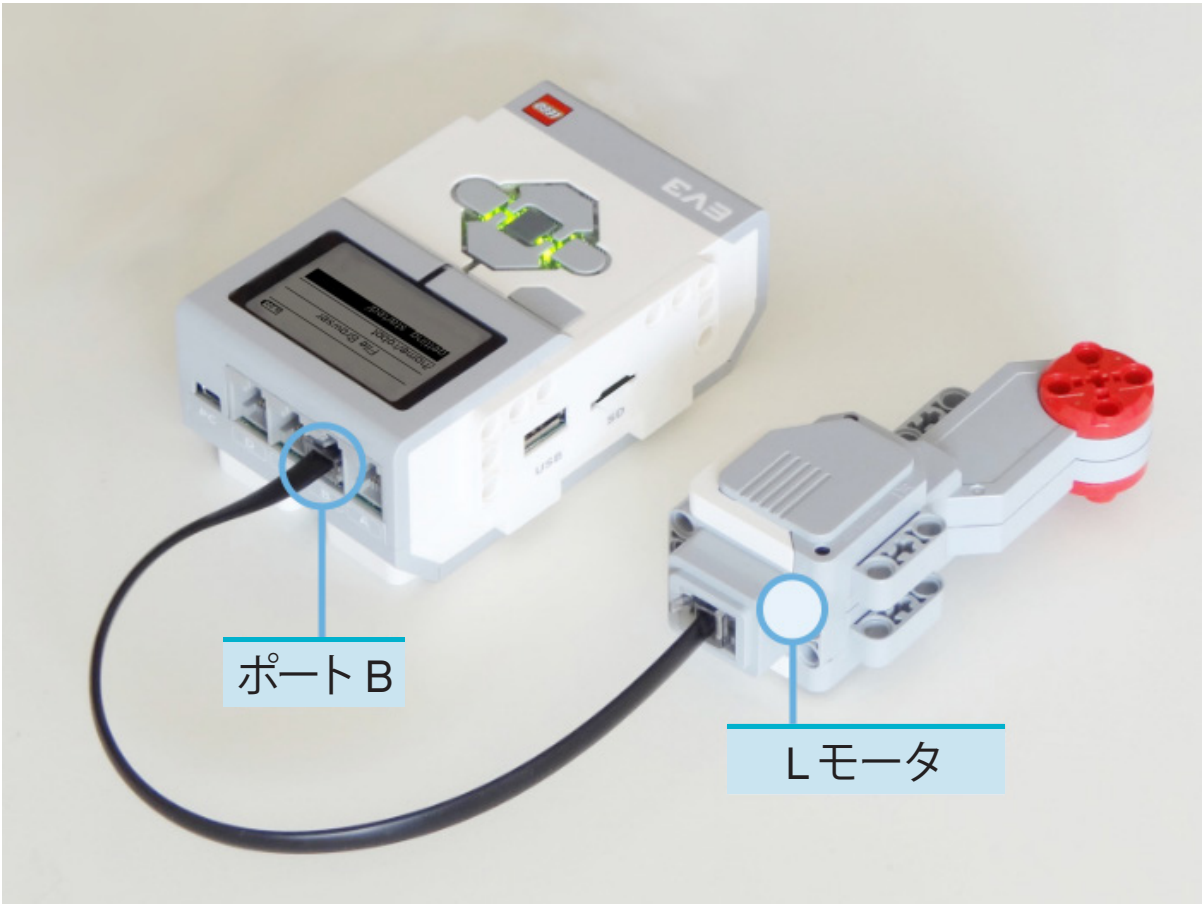


図 2.7:ポート B にLモータが接続された EV3 ブロック。

次に、 *main.py*を、次のように編集します。

```
#!/usr/bin/env pybricks-micropython

from pybricks import ev3brick as brick
from pybricks.ev3devices import Motor
from pybricks.parameters import Port

# Play a sound.
brick.sound.beep()

# Initialize a motor at port B.
test_motor = Motor(Port.B)

# Run the motor up to 500 degrees per second. To a target angle of 90 degrees.
test_motor.run_target(500, 90)

# Play another beep sound.
# This time with a higher pitch(1000 Hz) and longer duration(500 ms).
brick.sound.beep(1000, 500)
```

このプログラムは、ロボットのビープ音を鳴らし、モータを回転させ、高い音程のトーンでもう一度ビープ音を鳴らすようになっています。
プログラムを実行して、プログラムのとおりに動作することを確認します。

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

2.6 EV3 ブロックでのファイル管理

EV3 ブロックにプロジェクトをダウンロードした後、図2.8 に示すように、保存されているプログラムを、デバイスブラウザで実行、削除、またはバックアップすることができます。

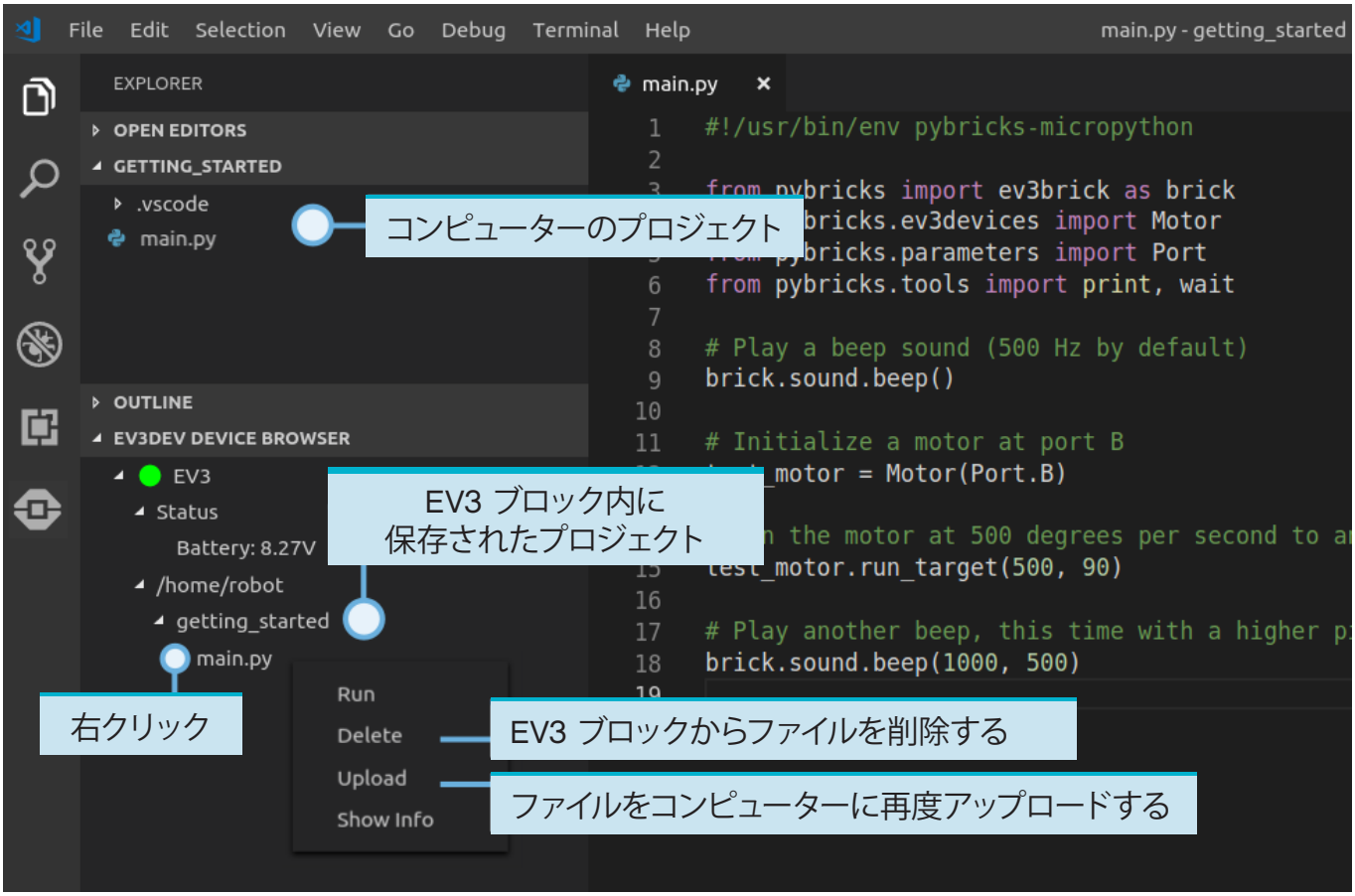


図 2.8:Ev3 デバイスブラウザで EV3 ブロック上のファイルを管理します。

2.7 コンピューターを使わずにプログラムを実行する

EV3 ブロックから、以前にダウンロードしたプログラムを直接実行することもできます。

これを行うには、EV3 画面の*ファイルブラウザ*でプログラムを検索し、[センター]ボタンキーを押して、図 2.9 に示すとおり、プログラムを起動します。

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

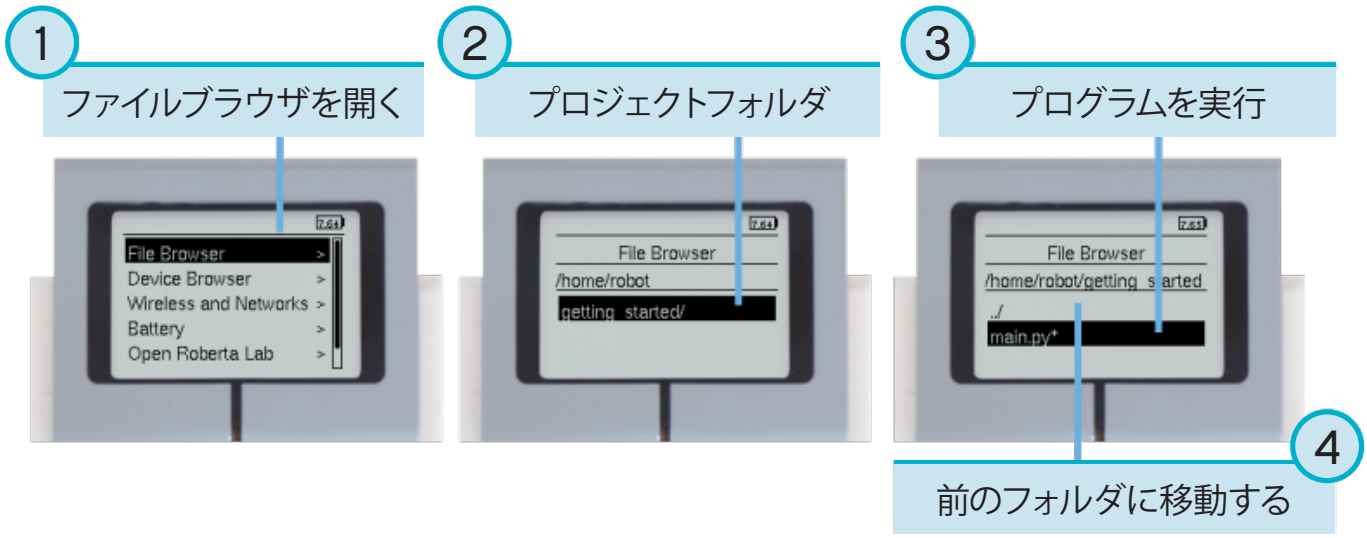


図 2.9: EV3 ブロックのボタンを使ってプログラムを起動します。

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

章3

EV3 BRICK – EV3 プログラマブルブロック

レゴ® マインドストーム® EV3

3.1 ボタン

buttons()

EV3 ブロックで、どのボタンが押されているかを確認します。

リターン押されたボタンのリスト。

戻り値 *Button* のの種類の一覧

例：

```
# Do something if the left button is pressed
if Button.LEFT in brick.buttons():
    print("The left button is pressed.")
```

```
# Wait until any of the buttons are pressed
while not any(brick.buttons()):
    wait(10)

# Wait until all buttons are released
while any(brick.buttons()):
    wait(10)
```

3.2 ライト

light(*color*)

ブロックのライトの色を設定します。

Parameters **color**(Color) – ライトの色。ライトをオフにするには Color.BLACK を選択するか、または None を選択します。

例：

```
# Make the light red
brick.light(Color.RED)

# Turn the light off
brick.light(None)
```

3.3 サウンド

classmethod sound.**beep**(*frequency=500, duration=100, volume=30*)
ビープ音やトーンを鳴らします。

Parameters

- **frequency(周波数)**(*frequency:Hz*) – ビープ音の周波数 (デフォルト: 500)。
- **duration(持続時間)**(*time: ms(時間：ミリ秒)*) – ビープ音の持続時間 (デフォルト: 100).
- **volume(音量)**(*percent(パーセント): %*) – ビープ音の音量 (デフォルト: 30).

例：

```
# A simple beep
brick.sound.beep()

# A high pitch(1500 Hz) for one second(1000 ms) at 50% volume
brick.sound.beep(1500, 1000, 50)
```

classmethod sound.**beeps**(*number*)
Defaultのいくつかのビープ音を、短い間隔で再生します。

Parameters number (*int*)– ビープ音の数です。

例：

```
# Make 5 simple beeps
brick.sound.beeps(5)
```

classmethod sound.**file**(*file__name, volume=100*)
音声ファイルを再生します。

Parameters

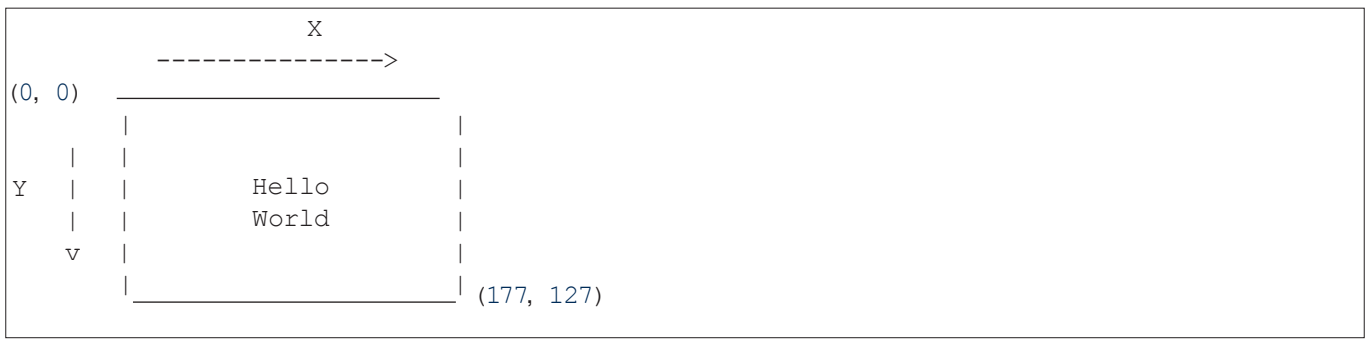
- **file _ name**(*str*) – 拡張子を含むサウンドファイルへのパス。
- **volume(音量)**(*percentage(パーセント): %*) – サウンドの音量(デフォルト: 100).

例：

```
# Play one of the built-in sounds
brick.sound.file(SoundFile.HELLO)

# Play a sound file from your project folder
brick.sound.file('mysound.wav')
```

3.4 ディスプレイ



LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

classmethod display.**clear**()
画面上のすべての文字や数字をクリアします。

classmethod display.**text**(*text, coordinate=None*)
テキストを表示します。

Parameters

- **text**(*str*) – 表示するテキスト。
- **coordinate**(*tuple*)(座標タプル) – (x, y) タプルを調整します。これは、最初の文字の左上隅にあります。座標が指定されていない場合は、次の行にプリントされます。

例：

```
# Clear the display
brick.display.clear()

# Print ``Hello`` near the middle of the screen
brick.display.text("Hello",(60, 50))

# Print ``World`` directly underneath it
brick.display.text("World")
```

classmethod display.**image**(*file__name, alignment=Align.CENTER, coordinate=None, clear=True)*
イメージファイルを表示します。

配置を指定するには、alignment(整列)を使用するか、coordinate(座標)を指定します。
ただし、両方を使って配置を指定することはできません。

Parameters

- **file _ name**(*str*) – 画像ファイルへのパス。絶対パス、またはプロジェクトフォルダに対する相対パスを利用できます。
- **alignment(整列)**(Align) – 画像を配置する場所 (*デフォルト*: Align.CENTER)。(中央揃え)
- **coordinate**(*tuple*)(座標タプル) – (x, y) タプルを調整します。
  画像の左上隅(*デフォルト*: None)(なし)
- **clear**(*bool*) – 画像を表示する前に画面をクリアするかどうかを指定します (*デフォルト*: True)(真)

例：

```
# Show a built-in image of two eyes looking upward
brick.display.image(ImageFile.UP)

# Display a custom image from your project folder
brick.display.image('pybricks.png')

# Display a custom image at the top right of the screen, without clearing
# the screen first
brick.display.image('arrow.png', Align.TOP_RIGHT, clear=False)
```

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

3.5 バッテリー

classmethod battery.**voltage**()
バッテリーの電圧の値を取得します。

リターン Battery voltage (バッテリー電圧)。

戻り値 *voltage(電圧): mV*

例:

```
# Play a warning sound when the battery voltage
# is below 7 Volt (7000 mV = 7V)
if brick.battery.voltage() < 7000:
    brick.sound.beep()
```

classmethod battery.**current**()
バッテリーによって供給された電流の値を取得します。

リターン Battery current (バッテリー電流)。

戻り値 *current: mA*

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

章 4

EV3 デバイス – EV3 モータとセンサ

レゴ®マインドストーム® EV3 モータとセンサ。

4.1 モータ

class Motor(*port, direction=Direction.CLOCKWISE, gears=None*)

レゴ® マインドストーム® EV3 MとLモータ。

エレメント 99455/6148292 または 95658/6148278、次のセットに含まれています:

- 31313:レゴ® マインドストーム® EV3 (2013)
- 45544:レゴ® マインドストーム® エデュケーション EV3 基本セット(2013)
- 45503 または 45502:セパレートパーツ (2013)

Parameters

- **port(ポート)** (Port) – モーターが接続されているポート。
- **direction(方向)** (Direction) – 正の速度方向 (*デフォルト*: Direction.CLOCKWISE).
- **gears(ギヤ)** (リスト) – モーターにリンクされたギヤのリスト (*デフォルト*: None).

  例えば: [12, 36] は、12 歯と 36 歯のギヤを使うギヤトレインを表します。
  図の比率の例を参照してください。

  [[12, 36], [20, 16, 40]]など、複数のギヤトレインのリストを使用します。

  ギヤトレインを指定すると、自動的にすべてのモータコマンドと設定が調整され、結果として得られるギヤ比が考慮されます。選択したギヤの数に関係なく、モータの方向は変わりません。

  たとえば、gears=[12,36],の場合、ギヤ比は 3 です。これは出力が機械的に 3 倍遅くなることを意味します。これを補償するために、モータコマンドを入力すると、モータは自動的に 3 倍の速さで 3 倍の距離を回転します。したがって、run_angle (200,90)を選択すると、メカニズムの出力は90度で200度/秒に変わります。

  以下の同じことがドキュメントにも当てはまります。「モータ角度」または「モータ速度」を示す場合、これを「メカニズムの出力角」および「メカニズム出力速度」として読み換えられます。これはギヤ比が自動的に考慮されるためです。

  gears 設定は、回転センサが付いているモータでのみ利用できます。

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

例：

```
# Initialize a motor(by default this means clockwise, without any gears).
example_motor = Motor(Port.A)

# Initialize a motor where positive speed values should go counterclockwise
right_motor = Motor(Port.B, Direction.COUNTERCLOCKWISE)

# Initialize a motor with a gear train
robot_arm = Motor(Port.C, Direction.CLOCKWISE, [12, 36])
```

回転センサのないモータの場合

dc(*duty*)
モータのデューティサイクルを設定します。

パラメータ duty(percentage: %) – デューティサイクル (-100.0 から 100)。

例：

```
# Set the motor duty cycle to 75%.
example_motor.duty(75)
```

回転センサ付きのモータの場合

angle()
ロボットの回転角度の値を取得します。

リターンモータの回転角度

戻り値 *angle: deg*

reset _ angle(*angle*)
モータの累積回転角をリセットします。

パラメータ angle(*angle: deg*) – 角度をリセットする値を指定します。

speed()
モータの速度 (角速度) の値を取得します。

リターンモータ回転速度

戻り値*回転速度: 度/秒 (rotational speed: deg/s)*

stop(停止)(*stop_type=Stop.COAST*)
モータを停止させる 。

パラメータstop _ type(Stop(停止)) – 惰力走行、ブレーキ、またはホールドのいずれか
(*デフォルト*: *Stop.COAST*),

run(実行)(*速度*)
モータを一定速度 (角速度) で走行させます。

モータは指定された速度で加速し、負荷下でも、デューティサイクルが自動的に調節され、一定した速度を保ちます。この走行は、モータに新しいコマンドを入力するか、プログラムが停止するまで、バックグラウンドで続行されます。

パラメータ speed(速度)(*回転速度: 度/秒*) – モーターの速度。

run _ time(*speed, time, stop_type=Stop.COAST, wait=True*)
所定の時間、一定の速度 (角速度) でモータを動かします。

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

モータは指定された速度で加速し、負荷下でも、デューティサイクルが自動的に調節され、一定した速度を保ちます。指定された時間後にぴったり停止するように減速が行われます。

Parameters

- **speed(速度)**(*回転速度: 度/秒*) – モーターの速度。
- **time(時間)**(*時間: ミリ秒*) – 操作の期間。
- **stop _ type** (`Stop`) – 停止した後に惰性走行、ブレーキ、またはホールドするかどうか (*Default:* `Stop.COAST`).
- **wait** (*bool*) – プログラムの残りの部分を続けて実行する前に、前の操作が完了するまで待ちます (*Default*: `True`)。これは、プログラムが指定された時間待機することを意味します。

run _ angle (*speed, rotation_angle, stop_type=Stop.COAST, wait=True*)

所定の角度で一定速度 (角速度) でモータを動かします。

モータは指定された速度で加速し、負荷下でも、デューティサイクルが自動的に調節され、一定した速度を保ちます。モータが所定の角度を走行した後で、ちょうど停止するように減速を行います。

Parameters

- **speed(速度)**(*回転速度: 度/秒*) – モーターの速度。
- **rotation _ angle** (*angle: deg*) – モーターを回転させる角度。
- **stop _ type**(`stop(停止)`) – 停止した後に惰性走行、ブレーキ、またはホールドするかどうか (*Default:* `Stop.COAST`).
- **wait** (*bool*) – プログラムの残りの部分を続けて実行する前に、前の操作が完了するまで待ちます (*Default:* `True`)。これは、モータが正確に指定された要求された角度だけ移動するまで、プログラムが待機することを意味します。

run _ target (*speed, target_angle, stop_type=Stop.COAST, wait=True*)

モータを一定の速度 (角速度) で、入力された目標角度になるまで動かします。

モータは指定された速度で加速し、負荷下でも、デューティサイクルが自動的に調節され、一定した速度を保ちます。入力された目標角度でちょうど停止するように減速を行います。

回転方向は、目標角度によって自動的に選択されます。

Parameters

- **speed(速度)**(*rotational speed: deg/s (回転速度: 度/秒)*) – モータの絶対速度。方向は目標角度に基づいて自動的に選択されます:指定した速度が正でも負でも、違いはありません。
- **target _ angle**(*angle:deg*) – 現在の角度に関係なく、モーターを回転させる目標角度。
- **stop _ type**(`stop(停止)`) – 停止した後に惰性走行、ブレーキ、またはホールドするかどうか (*Default:* `Stop.COAST`).
- **wait** (*bool*) – プログラムの残りの部分を続けて実行する前に、前の操作が完了するまで待ちます (*Default*: `True`)。これは、モータが目標角度に到達するまで、プログラムが待機することを意味します。

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

回転センサ付きモータの高度な方法

track_target(目標追跡)(*target_angle(目標角度)*)
時間がたつにつれて変化する目標角度を追跡します。

この関数は *run_target()* と非常によく似ていますが、速さと加速度の設定は無視され、可能な限り速く目標角度まで移動します。このコマンドでは、target_angle(目標角度)の変化の速さと加速度を選択することで、スピードと加速を調整できます。

この方法は、高速ループでモータの目標が連続的に変化する場合に役立ちます。

パラメータ target_angle(*angle:deg*)-モーターを回転させる目標角度。

例:

```
# Initialize motor and timer
from math import sin
motor = Motor(Port.A)
watch = StopWatch()
amplitude = 90

# In a fast loop, compute a reference angle
# and make the motor track it.
while True:

    # Get the time in seconds
    seconds = watch.time()/1000
    # Compute a reference angle. This produces
    # a sine wave that makes the motor move
    # smoothly between -90 and +90 degrees.
    angle_now = sin(seconds)*amplitude
    # Make the motor track the given angle
    motor.track_target(angle_now)
```

stalled()
現在モータがストールしているかどうかを確認します。

最大トルクでも動けなくなると、モータはストールします。たとえば、何かがモータをブロックしている場合や、メカニズムが物理的にそれ以上回転できない場合などがあてはまります。

具体的には、PID コントローラによって計算されたデューティサイクルが、最大に達し(したがって duty (デューティ) = duty_limit)、かつモータが最小速度に達することができない状態(したがって、speed(速度) < stall_speed)が、少なくとも stall_time(ストール持続時間)間続くと、モータがストールしたと判定されます。

ストール検知の感度の調整を行うときは、*set_dc_settings()* と *set_pid_settings()* を使って、duty_limit(デューティ限界), stall_speed(ストール速度), stall_time(ストール持続時間)をそれぞれ変更できます。

リターンモータがストールしている場合はTrueを返し、そうでない場合は Falseを返します。

戻り値 bool

run_until_stalled (*speed, stop_type=Stop.COAST, duty_limit=default*)
モータを一定速度 (角速度) でストールするまで動かします。最大トルクでも動けなくなると、モータはストールしているとみなされます。正確な定義は、*stalled()* を参照してください。

Duty_limit 引数を使用すると、この操作中にモータのトルクを一時的に制限できます。これは、ギヤ付きまたはレバー機構に、フルモータトルクを適用しないようにするときに便利です。

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

Parameters

- **speed(速度)**(*回転速度: 度/秒*) – モーターの速度。
- **stop _ type**(stop(停止)) – 停止した後に惰性走行、ブレーキ、またはホールドするかどうか (*Default: Stop.COAST*).
- **duty _ limit** (*percent: %*) – 相対トルク制限値。この制限は *set _ dc _ settings()* と同じように機能しますが、この場合の制限は一時的なものであり、コマンドを完了した後は、以前の値に戻されます。

set _ dc _ settings (*duty_limit, duty_offset*)
設定を変更すると、*dc()* コマンドの動作を調整できます。これは、バックグラウンドで *dc()* メソッドを使用するすべての run commandsにも影響します。

Parameters

- **duty _ limit** (*percent(パーセント): %*) – 後続のモータコマンドの相対トルクを制限しますこれにより、後続のモータコマンドに適用される最大デューティサイクルが設定されます。これにより、最大トルク出力が、絶対最大ストールトルクのパーセンテージまで減少します。これは、モータのフルトルクを、ギヤ機構またはレバー機構に適用したり、LEGO®トレインが意図せずにフルスピードで回転しないようにする場合に便利です。(*Default*:100).
- **duty _ offset** (percent: %) – *dc()*を使用する場合に指定される最小デューティサイクル。これにより、小さなフィード漸進トルクが加わり、非常に低いデューティサイクル値でも、モータが動くようになります。独自のフィードバックコントローラを作成するときに便利です *(Default: 0).*

set _ run _ settings (*max_speed, acceleration*)
すべての run コマンドに対して、モータの最大速度と加速/減速を設定します。

これは、モータに入力されるrun, run _ time, run _ angle, run _ target, run _ until _ stalled コマンドに適用されます。各モータのDefaultのParametersも参照してください。

Parameters

- **max _ speed**(*rotational speed: deg/s(回転速度: 度/秒)*) – モーターコマンドの間のモーターの最高速度。
- **acceleration**(加速度) (*rotational acceleration(回転加速度): deg/s/s(度/秒/秒)*) – 目標速度に向けた加速度と停止に向けた減速。これは正の値にする必要があります。モータは必要に応じて減速の符号を自動的に変更します。

例:

```
# Set the maximum speed to 200 deg/s and acceleration to 400 deg/s/s.
example_motor.set_run_settings(200, 400)

# Make the motor run for 5 seconds. Even though the speed argument is 300
# deg/s in this example, the motor will move at only 200 deg/s because of
# the settings above.
example_motor.run_time(300, 5000)
```

set _ pid _ settings (*kp, ki, kd, tight_loop_limit, angle_tolerance, speed_tolerance, stall_speed, stall_time*)
位置と速度のコントローラの設定を設定します。各モータの pid とDefaultのParametersも参照してください。

Parameters

- **kp** (*int*) – プロポーショナルポジション(および積分速度)制御定数。
- **ki** (*int*) – 積分位置制御定数。
- **kd** (*int*) – デリバティブポジション(および比例速度)制御定数。

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

- **tight_loop_limit** (*time: ms(時間：ミリ秒)*) – 前のコマンドを開始したあとで、この間隔内に実行コマンドを実行すると、コントローラは、オペレーターが直接速度を操作しようとしていると判断します。これは、アクセラレータの設定を無視し、すぐに run コマンドで指定された速度の追跡を開始することを意味します。これは高速ループで、モータをスムーズに加速させるのではなく、ライントレースロボットなどを使用して、迅速に反応させたい場合に便利です。
- **angle_tolerance** (*angle:deg*) – 動作が完了したと見なされる前に、目標角度から許容される偏差を指定します。
- **speed_tolerance** (*rotational speed: deg/s*) (回転速度: 度/秒) – モーションが完了したと見なされる前に、ゼロ速度からの偏差が許容されます。
- **stall_speed**(*rotational speed: deg/s*)(回転速度: 度/秒) – *ストール ()* を参照してください。
- **stall_time** (*time: ms*)(時間: ミリ秒) – *ストール ()* を参照してください。

4.2 センサ

4.2.1 タッチセンサ

class TouchSensor (*port*)
レゴ®マインドストーム® EV3 タッチセンサ。

エレメント 95648/6138404、含まれるセット:

- 31313:レゴ® マインドストーム® EV3 (2013)
- 45544:レゴ® マインドストーム® エデュケーション EV3 基本セット(2013)
- 45507:セパレートパーツ (2013)

パラメーターport(ポート)(Port) – センサーが接続されているポート。

センサが押されています()
センサが押されていないか確認してください。

リターンセンサが押されている場合は True、押されていない場合は False を返します。

戻り値 bool

4.2.2 カラーセンサ

class ColorSensor(*port*)
レゴ® マインドストーム® EV3 カラーセンサ。

エレメント95650/6128869 、含まれるセット:

- 31313:レゴ® マインドストーム® EV3 (2013)
- 45544:レゴ® マインドストーム® エデュケーション EV3 基本セット(2013)
- 45506:セパレートパーツ (2013)

パラメーター port(ポート)(Port) – センサーが接続されているポート。

color()
表面のカラーを測定します。

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

リターン `Color.BLACK`, `Color.BLUE`, `Color.GREEN`, `Color.YELLOW`, `Color.RED`, `Color.WHITE`, `Color.BROWN` または `None` のいずれかです

戻り値 `Color`(色)、または 色が検出できない場合は、`None` を返します。

ambient ()
周辺の光の強さを測定します。

リターン周囲光の強さを0 (暗い) ～ 100 (明るい) の範囲で返します。

戻り値 *percentage (パーセンテージ): %*

反射 ()
赤色のライトを使用して表面反射を測定します。

リターン 0(反射なし) から 100 (高反射) までの範囲の反射を返します。

戻り値 *percentage (パーセンテージ): %*

rgb ()
赤、緑、青の光を使用して表面の反射を測定します。

リターン赤、緑、青の光の反射を返し、それぞれの範囲は 0.0 (反射なし) ～ 100.0 (高反射) になっています。

戻り値 3つの*パーセンテージのタプル*

4.2.3 赤外線センサとビーコン

{**class InfraredSensor** (*port*)
レゴ® マインドストーム® EV3 赤外線センサとビーコン。

エレメント 95654/6132629 および 72156/6127283。次のセットに含まれています:

- 31313:レゴ® マインドストーム® EV3 (2013)
- 45509と 45508:セパレートパーツ (2013)

パラメーター **port(ポート)**(`Port`) – センサーが接続されているポート。

distance ()
赤外線を使用して、センサとオブジェクト間の相対距離を測定します。

リターン 0 (最も近い) から 100 (最も遠い) までの相対距離をます。

戻り値 *relative distance 相対距離: %*

beacon (*channel*)
リモコンと赤外線センサ間の相対距離と角度を測定します。

Parameters **channel(チャンネル)**(*int*) – リモートのチャネル番号。

リターンリモコンと赤外線センサの間の相対距離 (0 ～ 100) とおおよその角度 (-75 ～75 度) のタプルを返します。

戻り値 (*相対距離: %、angle: deg*) またはリモートが検出されない場合 (`None`)。

ボタン (*チャンネル*)
赤外線リモコンのボタンが押されていることを確認します。

Parameters **channel(チャンネル)**(*int*) – リモートのチャネル番号。

リターン指定したチャネルのリモコンで押されたボタンの一覧を返します。

戻り値 *Button*の*一覧*

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

4.2.4 超音波センサ

class UltrasonicSensor (*port*)
レゴの® マインドストーム® EV3 超音波センサ。

エレメント 95652/6138403、含まれるセット:

- 45544:レゴ® マインドストーム® エデュケーション EV3 基本セット(2013)
- 45504:セパレートパーツ (2013)

パラメーター port(ポート) (Port)– センサーが接続されているポート。

distance(距離)(*silent=False*)
超音波音波を使用して、センサと物体との間の距離を測定します。

Parameters silent (サイレント) (*bool*) – 距離を測定した後、センサをオフにするには、True を選択します。

センサをオンのままにするには、False を選択します (*Default*)。

silent = True を選択すると、計測を行う場合を除き、センサは超音波を発信しません。この場合、他の超音波センサとの干渉は低減されますが、センサをオフにするときには、毎回約300 ミリ秒が必要です。

リターン距離(ミリメートル) を返します。

戻り値 *distance(距離): mm*

存在()
超音波を検出することにより、他の超音波センサの存在を確認します。

他の超音波センサがサイレントモードで動作している場合は、他のセンサが測定を行っている間のみ、存在を検出することができます。

リターン超音波が検出された場合は True を返し、そうでない場合は False を返します。

戻り値 bool

4.2.5 ジャイロセンサ

class GyroSensor (*port, direction=Direction.CLOCKWISE*)
レゴ® マインドストーム® EV3 ジャイロセンサ。

エレメント99380/6138411 、含まれるセット:

- 45544:レゴ® マインドストーム® エデュケーションEV3 基本セット(2013)
- 45505:セパレートパーツ (2013)

Parameters

- **ポート**(Port) – センサが接続されているポート。
- **方向** (Direction)(方向) – センサ上の赤い点正の回転方向 (*Default*:Direction.CLOCKWISE).

speed ()
センサの速度 (角速度) を取得します。

リターンセンサ角速度を返します。

戻り値 *回転速度: 度/秒*

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

angle ()
センサの累積角度を取得します。

リターン 回転角度を返します。

戻り値 *angle: 度*

reset_angle (*angle*)
センサの回転角度を希望の値に設定します。

Parameters **angle(角度)** (*angle: deg*) (角度：度) – 角度をリセットしたときの値を指定します。

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

章5

PARAMETERS – PARAMETERSと定数

Pybricks API の定数Parameters/引数。

class Port

EV3 プログラマブルブロックのポート。

Motor ports

A

B

C

D

Sensor ports:

S1

S2

S3

S4

class Direction

正の速度値の場合の回転方向: 時計回りまたは反時計回り。

CLOCKWISE

正の速度の値では、モータが時計回りに動きます。

COUNTERCLOCKWISE

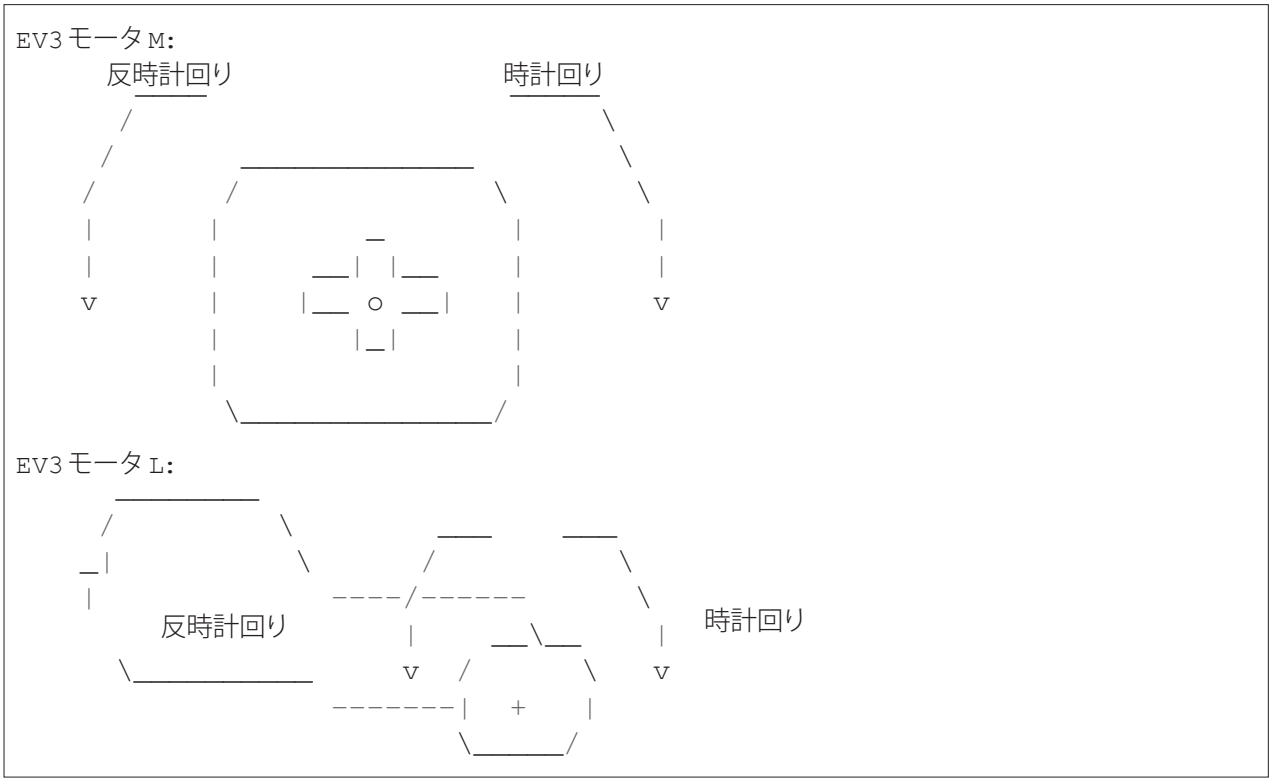
正の速度の値を指定すると、モータが反時計回りに動きます。

すべてのモータについて、回転の向きは、時計を見るのと同じように、シャフトを見た方向から定義されます。

NXT または EV3 モータの場合は、モータの赤/オレンジのシャフトが右下にあることを確認してください。

Parameter	正の速度	負の速度
Direction.CLOCKWISE	時計回り	反時計回り
Direction.COUNTERCLOCKWISE	反時計回り	時計回り

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.



class Stop

モータが停止した後の動作: 惰性走行、ブレーキ、またはホールド。

COAST

モータを自由に回転させます。

BRAKE

小さな外力に対して、受動的に抵抗します。

HOLD

モータの制御を続け、指示された角度で保持します。このコマンドは、エンコーダー付きモータでのみ利用可能です。

Stop タイプは、停止した後の動作に対する抵抗を定義します。

Parameter	抵抗	物理的意味
Stop.COAST	低	摩擦
Stop.BRAKE	中	摩擦 + 動きと反対のトルク
Stop.HOLD	高	摩擦 +コマンド角度を保持するトルク

class Color

光または表面の色。

BLACK

BLUE

GREEN

YELLOW

RED

WHITE

BROWN

ORANGE

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

PURPLE

class Button
ブロックやリモコンのボタン:

LEFT_DOWN

DOWN

RIGHT_DOWN

LEFT

CENTER

RIGHT

LEFT_UP

UP

BEACON

RIGHT_UP

LEFT__UP	UP/BEACON	RIGHT__UP
LEFT	CENTER	RIGHT
LEFT__DOWN	DOWN	RIGHT__DOWN

class Align
ディスプレイ上の画像の位置合わせ。

BOTTOM_LEFT

BOTTOM

BOTTOM_RIGHT

LEFT

CENTER

RIGHT

TOP_LEFT

TOP

TOP_RIGHT

class ImageFile
標準の EV3 画像へのパス。

Information

RIGHT

FORWARD

ACCEPT

QUESTION_MARK

STOP_1

LEFT

DECLINE

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

THUMBS_DOWN

BACKWARD

NO_GO

WARNING

STOP_2

THUMBS_UP

レゴ®

EV3

EV3_ICON

オブジェクト

TARGET

目

BOTTOM_RIGHT

BOTTOM_LEFT

EVIL

CRAZY_2

KNOCKED_OUT

PINCHED_RIGHT

WINKING

DIZZY

DOWN

TIRED_MIDDLE

MIDDLE_RIGHT

SLEEPING

MIDDLE_LEFT

TIRED_RIGHT

PINCHED_LEFT

PINCHED_MIDDLE

CRAZY_1

NEUTRAL

AWAKE

UP

TIRED_LEFT

ANGRY

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

class SoundFile

EV3 標準サウンドへのパス。

表現

SHOUTING

CHEERING

CRYING

OUCH

LAUGHING_2

SNEEZING

SMACK

BOING

BOO

UH_OH

SNORING

KUNG_FU

FANFARE

CRUNCHING

MAGIC_WAND

LAUGHING_1

情報

LEFT

BACKWARDS

RIGHT

OBJECT

COLOR

FLASHING

ERROR

ERROR_ALARM

DOWN

FORWARD

ACTIVATE

SEARCHING

TOUCH

UP

ANALYZE

STOP

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

DETECTED
TURN
START
通信
MORNING
EV3
GO
GOOD_JOB
OKEY_DOKEY
GOOD
NO
THANK_YOU
YES
GAME_OVER
OKAY
SORRY
BRAVO
GOODBYE
HI
HELLO
MINDSTORMS
LEGO
FANTASTIC
動作
SPEED_IDLE
SPEED_DOWN
SPEED_UP
色
BROWN
GREEN
BLACK
WHITE
RED
BLUE
YELLOW
機械

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

TICK_TACK

HORN_1

BACKING_ALERT

MOTOR_IDLE

AIR_RELEASE

AIRBRAKE

RATCHET

MOTOR_STOP

HORN_2

LASER

SONAR

MOTOR_START

動物

INSECT_BUZZ_2

ELEPHANT_CALL

SNAKE_HISS

DOG_BARK_2

DOG_WHINE

INSECT_BUZZ_1

DOG_SNIFF

T_REX_ROAR

INSECT_CHIRP

DOG_GROWL

SNAKE_RATTLE

DOG_BARK_1

CAT_PURR

数字

ZERO

ONE

TWO

THREE

FOUR

FIVE

SIX

SEVEN

EIGHT

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

NINE

TEN

システム

READY

CONFIRM

GENERAL_ALERT

CLICK

OVERPOWER

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

章6

ツール – タイミングとデータロギング

タイミングとデータロギングを行う一般的なツール。

print (*value, ..., sep, end, file, flush*)

端末またはストリームに値をプリントします。

例：

```
```

wait (*time*)

指定した時間だけ、ユーザープログラムを一時停止します。

Parameters time(時間) (*time (時間): ms (ミリ秒)*)-待機時間を表します。

class StopWatch

ストップウォッチで、時間間隔を測定できます。携帯電話のストップウォッチ機能に似ています。

time ()

ストップウォッチの現在の時刻を取得します。

リターン経過時間を返します。

戻り値 *time (時間): ms (ミリ秒)*

pause ()

ストップウォッチを一時停止します。

resume ()

ストップウォッチを再開します。

reset ()

ストップウォッチの時間を0にリセットします。

実行状態は影響を受けません:

- 一時停止していた場合、一時停止されたままになります(ただし、現在は 0 になっています)。
- 実行中の場合は、実行されたままになります(ただし、再び 0 から開始します)。

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

章7

ロボティクス – ロボティクスモジュール

Pybricks API 用のロボティクスモジュールです。

class DriveBase (*left_motor, right_motor, wheel_diameter, axle_track*)

2つの駆動車輪とオプションのホイールキャスター付きのロボット車両を表すクラス。

Parameters

- **left_motor**(Motor) – 左側の車輪を駆動するモータ。
- **right_motor**(Motor) – 右側の車輪を駆動するモータ。
- **wheel_diameter***(dimension: mm)* – 車輪の直径。
- **axle_track***(dimension: mm)* – 2 つの車輪の中間点の間の距離。

例:

```
# Initialize two motors and a drive base
left = Motor(Port.B)
right = Motor(Port.C)
robot = DriveBase(left, right, 56, 114)
```

drive (*speed, steering*)

指定された速度と回転率で運転を開始します。これらの値は、両方のロボットの車輪の間の中心点で測定を行います。

Parameters

- **speed(速度)**(*speed(速度): mm/秒*) – ロボットの前進速度。
- **steering(ステアリング)**(*rotational speed (回転速度): deg/s (度/秒)*) – ロボットの回転率。

例:

```
# Initialize two motors and a drive base
left = Motor(Port.B)
right = Motor(Port.C)
robot = DriveBase(left, right, 56, 114)

# Initialize a sensor
sensor = UltrasonicSensor(Port.S4)

# Drive forward until an object is detected
robot.drive(100, 0)
while sensor.distance() > 500:
    wait(10)
robot.stop()
```

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

drive _ time(*speed (速度), steering (ステアリング), time (時間)*)
指定された速度と回転率で一定時間作動してから、停止します。

Parameters

- **speed(速度)** (*speed (速度): mm/秒*) – ロボットの前進速度。
- **steering(ステアリング)** (*rotational speed (回転速度): 度/秒*) – ロボットの回転率。
- **time(時間)** (*time(時間): ms (ミリ秒)*) – 操作の長さ。

例:

```
# Drive forward at 100 mm/s for two seconds
robot.drive(100, 0, 2000)

# Turn at 45 deg/s for three seconds
robot.drive(0, 45, 3000)
```

stop(停止) (*stop_type=Stop.COAST*)
ロボットを停止します。

Parameters **stop _ type**(Stop) – 惰性走行ブレーキ、ホールドのいずれか (*Default*: *Stop.COAST*)。

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

章 8

信号と単位

多くのコマンドでは、既知の物理量で引数を指定できます。このページでは、各数量とその単位の概要を示します。

8.1 時間 time: ms

すべての時間と期間の値は、ミリ秒 (ms) 単位で測定されます。

たとえば、*run_time(実行時間)*を使用した動作の継続時間、*wait(待機)*時間の値の戻り値は*StopWatch*はミリ秒単位で指定します。

8.2 角度 angle: deg

すべての角度は、度 (deg) で測定されます。一回転は、360 度に相当します。

たとえば、*Motor*または*GyroSensor*の角度の値は度で表されます。

8.3 回転速度 rotational speed: deg/s

回転速度 (角速度) は、何かが回転する速さを表し、1 秒あたりの度数 (deg/s (度/秒)) で表されます。

たとえば、*Motor*または*GyroSensor*の回転速度の値は、1秒あたりの度数で表されます。

プログラムでは、1秒あたりの度数の使用をお勧めしています。次の表を使用すると、一般的に使用される単位の変換を行うことができます。

	deg/s	rpm
1 deg/s =	1	1/6=0.167
1 rpm =	6	1

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

8.4 距離 distance: mm

距離は、ミリメートル (mm) 単位が使用できるが使用できる場合は、この単位で表されます。

たとえば、*UltrasonicSensor*の距離の値はミリメートルで測定されます。

プログラムでは、ミリメートルの使用をお勧めしています。次の表を使用すると、一般的に使用される単位の変換を行うことができます。

	mm	cm	inch
1 mm =	1	0.1	0.0394
1 cm =	10	1	0.394
1 inch =	25.4	2.54	1

8.5 dimention: mm

寸法も、距離と同じように、ミリメートル (mm) 単位が使用できる場合は、この単位で表現されます。

たとえば、車輪の直径は、ミリメートルで測定されます。

8.6 相対距離 relative distance: %

距離測定によっては、特定の単位で正確な値が得られない場合がありますが、これらは、非常に近い範囲 (0%) から非常に遠い (100%) まで、さまざまな範囲に広がっています。これらは相対距離と呼ばれます。

たとえば、 *InfraredSensor*の距離値は相対距離です。

8.7 速度 speed: mm/s

リニア速度はミリメートル/秒 (mm/秒) で表されます。

たとえば、ロボット車両の速度は mm/秒で表されます。

8.8 回転速度 rotational acceleration: deg/s/s

回転加速度、または *angular acceleration (角加速度)*は、回転速度の変化の速さを表します。これは1秒間の毎秒の変化 (deg/s/s(度/秒/秒))として表されます。これはまた、一般的に *deg/s (度/秒)²*としても表記されます。

たとえば、*Motor*の回転加速度の設定を調整し、一定速度のセットポイントにどれだけ速く到達するか変更することができます。

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

8.9 パーセンテージ percentage: %

特定の単位を持たない信号もありますが、これらは最小値 (0%)から最大値 (100%) に及びます。
特にパーセンテージで表される値には *relative distances(相対距離)* があります。

たとえば、サウンドの *volume(音量)* の範囲は 0% ～ 100% です。

8.10 frequency (周波数): Hz

音の周波数はヘルツ (Hz) で表されます。

たとえば、*beep(ビープ音)* の周波数を選択すると、音程を変えることができます。

8.11 voltage (電圧): mV

電圧はミリボルト (mV) で表されます。

例えば、*battery(バッテリー)* の電圧を確認することができます。

8.12 current (電流): mA

電流はミリアンペア (mA) で表されます。

例えば、*battery(バッテリー)* から供給される電流を確認することができます。

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

章9

ロボットエデュケーター

この例では、ロボットエデュケーター(図 9.1) を、障害物を認識するまで走行させます。その後、バックして振り向くと、再び走行を開始します。

ロボットエデュケーターの組み立て説明書は、レゴ®エデュケーションのウェブサイトにあります。

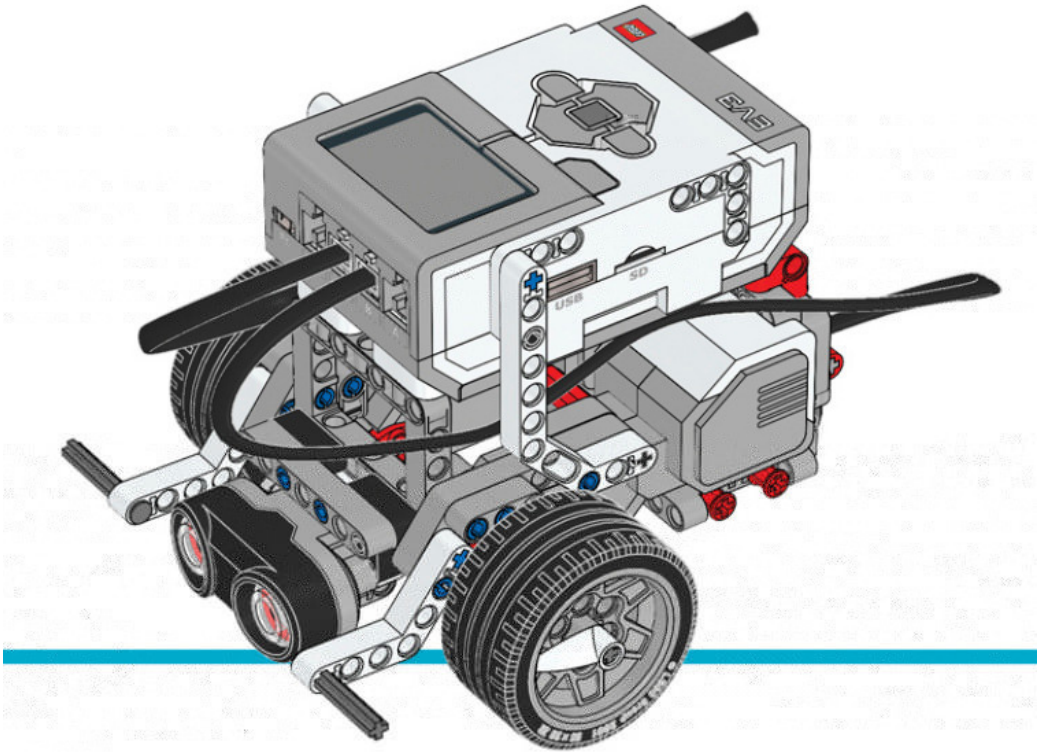


図 9.1: 超音波センサ付きロボット エデュケーター。

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

```
#!/usr/bin/env pybricks-micropython

from pybricks import ev3brick as brick
from pybricks.ev3devices import Motor, TouchSensor, ColorSensor
from pybricks.parameters import Port, Button, Color, ImageFile, SoundFile
from pybricks.tools import wait
from pybricks.robotics import DriveBase

# Play a sound.
brick.sound.beep()

# Initialize the Ultrasonic Sensor. It is used to detect
# obstacles as the robot drives around.
obstacle_sensor = UltrasonicSensor(Port.S4)

# Initialize two motors with default settings on Port B and Port C.
# These will be the left and right motors of the drive base.
left_motor = Motor(Port.B)
right_motor = Motor(Port.C)

# The wheel diameter of the Robot Educator is 56 millimeters.
wheel_diameter = 56

# The axle track is the distance between the centers of each of the wheels.
# For the Robot Educator this is 114 millimeters.
axle_track = 114

# The DriveBase is composed of two motors, with a wheel on each motor.
# The wheel_diameter and axle_track values are used to make the motors
# move at the correct speed when you give a motor command.
robot = DriveBase(left_motor, right_motor, wheel_diameter, axle_track)

# The following loop makes the robot drive forward until it detects an
# obstacle. Then it backs up and turns around. It keeps on doing this
# until you stop the program.
while True:
    # Begin driving forward at 200 millimeters per second.
    robot.drive(200, 0)

    # Wait until an obstacle is detected. This is done by repeatedly
    # doing nothing(waiting for 10 milliseconds) while the measured
    # distance is still greater than 300 mm.
    while obstacle_sensor.distance() > 300:
        wait(10)

    # Drive backward at 100 millimeters per second. Keep going for 2 seconds.
    robot.drive_time(-100, 0, 2000)

    # Turn around at 60 degrees per second, around the midpoint between
    # the wheels. Keep going for 2 seconds.
    robot.drive_time(0, 60, 2000)
```

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

章10

カラーソーター

このカラーソーターのサンプルプログラム(図10.1)では、カラーセンサを使用して、色付きのテクニックビームをスキャンできます。

色付きのビームを1つずつスキャンし、トレーに加えます。ビープ音で、色を登録したことを確認します。トレーがいっぱいになったときに、または中央のボタンを押すと、ロボットによってテクニックブロックの色別の分類が開始されます。

レゴ®エデュケーションのウェブサイトで、カラーソーターの組み立て手順を確認できます。

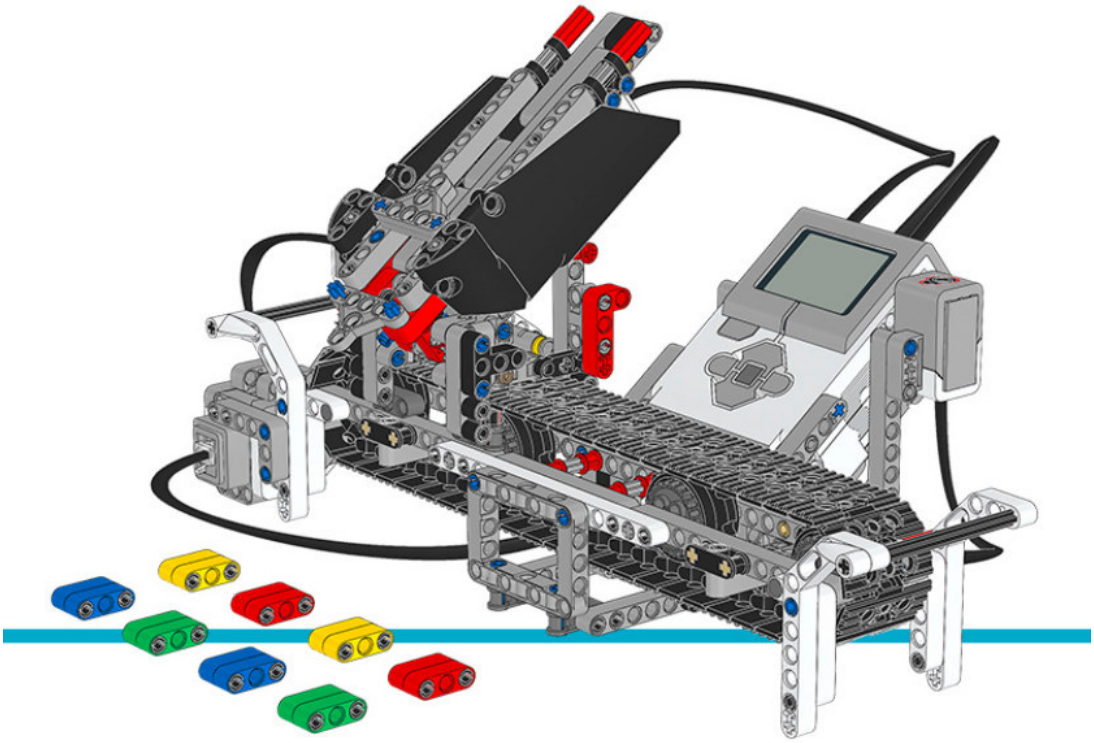


図 10.1: カラーソーター

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

```
#!/usr/bin/env pybricks-micropython

from pybricks import ev3brick as brick
from pybricks.ev3devices import Motor, TouchSensor, ColorSensor
from pybricks.parameters import Port, Button, Color, ImageFile, SoundFile
from pybricks.tools import wait

# The colored objects are either red, green, blue, or yellow.
POSSIBLE_COLORS =(Color.RED, Color.GREEN, Color.BLUE, Color.YELLOW)

# Initialize the motors that drive the conveyor belt and eject the objects.
belt_motor = Motor(Port.D)
feed_motor = Motor(Port.A)

# Initialize the Touch Sensor. It is used to detect when the belt motor
# has moved the sorter module all the way to the left.
touch_sensor = TouchSensor(Port.S1)

# Initialize the Color Sensor. It is used to detect the color of the objects.
color_sensor = ColorSensor(Port.S3)

# This is the main loop. It waits for you to scan and insert 8 colored objects.
# Then it sorts them by color. Then the process starts over and you can scan
# and insert the next set of colored objects.
while True:
    # Get the feed motor in the correct starting position.
    # This is done by running the motor forward until it stalls. This
    # means that it cannot move any further. From this end point, the motor
    # rotates backward by 180 degrees. Then it is in the starting position.
    feed_motor.run_until_stalled(120)
    feed_motor.run_angle(450, -180)

    # Get the conveyor belt motor in the correct starting position.
    # This is done by first running the belt motor backward until the
    # touch sensor becomes pressed. Then the motor stops, and the the angle is
    # reset to zero. This means that when it rotates backward to zero later
    # on, it returns to this starting position.
    belt_motor.run(-500)
    while not touch_sensor.pressed():
        pass
    belt_motor.stop()
    wait(1000)
    belt_motor.reset_angle(0)

    # Clear all the contents from the display.
    brick.display.clear()

    # When we scan the objects, we store all the color numbers in a list.
    # We start with an empty list. It will grow as we add colors to it.
    color_list = []

    # This loop scans the colors of the objects. It repeats until 8 objects
    # are scanned and placed in the chute. This is done by repeating the loop
    # while the length of the list is still less than 8.
    while len(color_list) < 8:
        # Show an arrow that points to the color sensor.
        brick.display.image(ImageFile.RIGHT)
```

(次のページに進む)

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

(前ページの続き)

```
    # Show how many colored objects we have already scanned.
    brick.display.text(len(color_list))

    # Wait for the center button to be pressed or a color to be scanned.
    while True:
        # Store True if the center button is pressed or False if not.
        pressed = Button.CENTER in brick.buttons()
        # Store the color measured by the Color Sensor.
        color = color_sensor.color()
        # If the center button is pressed or a color is detected,
        # break out of the loop.
        if pressed or color in POSSIBLE_COLORS:
            break

    if pressed:
        # If the button was pressed, end the loop early.
        # We will no longer wait for any remaining objects
        # to be scanned and added to the chute.
        break
    else:
        # Otherwise, a color was scanned.
        # So we add(append) it to the list.
        brick.sound.beep(1000, 100, 100)
        color_list.append(color)

        # We don't want to register the same color once more if we're
        # still looking at the same object. So before we continue, we
        # wait until the sensor no longer sees the object.
        while color_sensor.color() in POSSIBLE_COLORS:
            pass
        brick.sound.beep(2000, 100, 100)

        # Show an arrow pointing to the center button,
        # to ask if we are done.
        brick.display.image(ImageFile.BACKWARD)
        wait(2000)

# Play a sound and show an image to indicate that we are done scanning.
brick.sound.file(SoundFile.READY)
brick.display.image(ImageFile.EV3)

# Now sort the bricks according the list of colors that we stored.
# We do this by going over each color in the list in a loop.
for color in color_list:

    # Wait for one second between each sorting action.
    wait(1000)

    # Run the conveyor belt motor to the right position based on the color.
    if color == Color.BLUE:
        brick.sound.file(SoundFile.BLUE)
        belt_motor.run_target(500, 10)
    elif color == Color.GREEN:
    brick.sound.file(SoundFile.GREEN)
        belt_motor.run_target(500, 132)
```

(次のページに進む)

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

(前ページの続き)

```
    elif color == Color.YELLOW:
        brick.sound.file(SoundFile.YELLOW)
        belt_motor.run_target(500, 360)
    elif color == Color.RED:
        brick.sound.file(SoundFile.RED)
        belt_motor.run_target(500, 530)

    # Now that the conveyor belt is in the correct position,
    # eject the colored object.
    feed_motor.run_angle(1500, 90)
    feed_motor.run_angle(1500, -90)
```

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

章 11

ロボットアーム H25

このプログラム例では、ロボット (図 11.1) が、ブラックホイールハブのスタックを永久に移動します。最初にロボットアームが初期化され、次にハブの移動が開始されます。

レゴ®エデュケーションウェブサイトでロボットの組み立て説明書を見ることができます。

ヒント：ロボットを組み立てるときには、microSD カードが簡単にアクセスできるように EV3 ブロックの向きを逆にします。

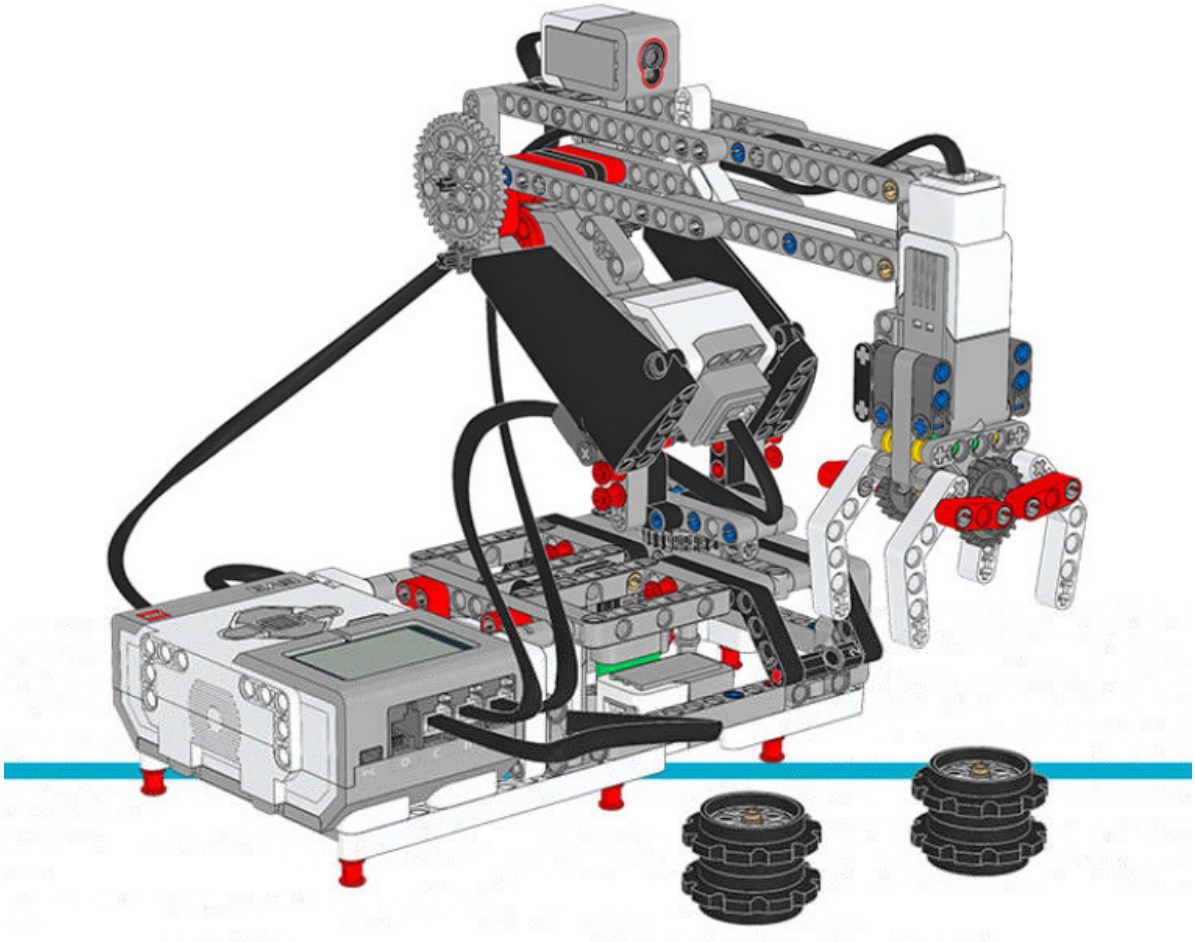


図 11.1: ロボットアーム H25

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

```
#!/usr/bin/env pybricks-micropython

from pybricks import ev3brick as brick
from pybricks.ev3devices import Motor, TouchSensor, ColorSensor
from pybricks.parameters import Port, Stop, Direction
from pybricks.tools import wait

# Configure the gripper motor on Port A with default settings.
gripper_motor = Motor(Port.A)

# Configure the elbow motor. It has an 8-teeth and a 40-teeth gear
# connected to it. We would like positive speed values to make the
# arm go upward. This corresponds to counterclockwise rotation
# of the motor.
elbow_motor = Motor(Port.B, Direction.COUNTERCLOCKWISE, [8, 40])

# Configure the motor that rotates the base. It has a 12-teeth and a
# 36-teeth gear connected to it. We would like positive speed values
# to make the arm go away from the Touch Sensor. This corresponds
# to counterclockwise rotation of the motor.
base_motor = Motor(Port.C, Direction.COUNTERCLOCKWISE, [12, 36])

# Limit the elbow and base accelerations. This results in
# very smooth motion. Like an industrial robot.
elbow_motor.set_run_settings(60, 120)
base_motor.set_run_settings(60, 120)

# Set up the Touch Sensor. It acts as an end-switch in the base
# of the robot arm. It defines the starting point of the base.
base_switch = TouchSensor(Port.S1)

# Set up the Color Sensor. This sensor detects when the elbow
# is in the starting position. This is when the sensor sees the
# white beam up close.
elbow_sensor = ColorSensor(Port.S3)

# Initialize the elbow. First make it go down for one second.
# Then make it go upwards slowly(15 degrees per second) until
# the Color Sensor detects the white beam. Then reset the motor
# angle to make this the zero point. Finally, hold the motor
# in place so it does not move.
elbow_motor.run_time(-30, 1000)
elbow_motor.run(15)
while elbow_sensor.reflection() < 32:
    wait(10)
elbow_motor.reset_angle(0)
elbow_motor.stop(Stop.HOLD)

# Initialize the base. First rotate it until the Touch Sensor
# in the base is pressed. Reset the motor angle to make this
# the zero point. Then hold the motor in place so it does not move.
base_motor.run(-60)
while not base_switch.pressed():
    wait(10)
base_motor.reset_angle(0)
base_motor.stop(Stop.HOLD)
```

(次のページに進む)

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

(前ページの続き)

```
# Initialize the gripper. First rotate the motor until it stalls.
# Stalling means that it cannot move any further. This position
# corresponds to the closed position. Then rotate the motor
# by 90 degrees such that the gripper is open.
gripper_motor.run_until_stalled(200, Stop.COAST, 50)
gripper_motor.reset_angle(0)
gripper_motor.run_target(200, -90)

def robot_pick(position):
    # This function makes the robot base rotate to the indicated
    # position. There it lowers the elbow, closes the gripper, and
    # raises the elbow to pick up the object.

    # Rotate to the pick-up position.
    base_motor.run_target(60, position, Stop.HOLD)
    # Lower the arm.
    elbow_motor.run_target(60, -40)
    # Close the gripper to grab the wheel stack.
    gripper_motor.run_until_stalled(200, Stop.HOLD, 50)
    # Raise the arm to lift the wheel stack.
    elbow_motor.run_target(60, 0, Stop.HOLD)

def robot_release(position):
    # This function makes the robot base rotate to the indicated
    # position. There it lowers the elbow, opens the gripper to
    # release the object. Then it raises its arm again.

    # Rotate to the drop-off position.
    base_motor.run_target(60, position, Stop.HOLD)
    # Lower the arm to put the wheel stack on the ground.
    elbow_motor.run_target(60, -40)
    # Open the gripper to release the wheel stack.
    gripper_motor.run_target(200, -90)
    # Raise the arm.
    elbow_motor.run_target(60, 0, Stop.HOLD)

# Play three beeps to indicate that the initialization is complete.
brick.sound.beeps(3)

# Define the three destinations for picking up and moving the wheel stacks.
LEFT = 160
MIDDLE = 100
RIGHT = 40

# This is the main part of the program. It is a loop that repeats endlessly.
#
# First, the robot moves the object on the left towards the middle.
# Second, the robot moves the object on the right towards the left.
# Finally, the robot moves the object that is now in the middle, to the right.
#
# Now we have a wheel stack on the left and on the right as before, but they
# have switched places. Then the loop repeats to do this over and over.
while True:
```

(次のページに進む)

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

(前ページの続き)

```
    # Move a wheel stack from the left to the middle.
    robot_pick(LEFT)
    robot_release(MIDDLE)

    # Move a wheel stack from the right to the left.
    robot_pick(RIGHT)
    robot_release(LEFT)

    # Move a wheel stack from the middle to the right.
    robot_pick(MIDDLE)
    robot_release(RIGHT)
```

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

PYTHON モジュールインデックス

e
ev3brick, 16
ev3devices, 20

p
parameters, 29

r
robotics, 38

t
tools, 37

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

索引

A

`Align` (*class in parameters*), 31
`Align.BOTTOM` (*in module parameters*), 31
`Align.BOTTOM_LEFT` (*in module parameters*), 31
`Align.BOTTOM_RIGHT` (*in module parameters*), 31
`Align.CENTER` (*in module parameters*), 31
`Align.LEFT` (*in module parameters*), 31
`Align.RIGHT` (*in module parameters*), 31
`Align.TOP` (*in module parameters*), 31
`Align.TOP_LEFT` (*in module parameters*), 31
`Align.TOP_RIGHT` (*in module parameters*), 31
`ambient` () (*ColorSensor method*), 26
`angle` () (*GyroSensor method*), 27
`angle` () (*Motor method*), 21

B

`beacon` () (*InfraredSensor method*), 26
`beep` () (*ev3brick.sound class method*), 17
`beeps` () (*ev3brick.sound class method*), 17
`Button` (*class in parameters*), 31
`Button.BEACON` (*in module parameters*), 31
`Button.CENTER` (*in module parameters*), 31
`Button.DOWN` (*in module parameters*), 31
`Button.LEFT` (*in module parameters*), 31
`Button.LEFT_DOWN` (*in module parameters*), 31
`Button.LEFT_UP` (*in module parameters*), 31
`Button.RIGHT` (*in module parameters*), 31
`Button.RIGHT_DOWN` (*in module parameters*), 31
`Button.RIGHT_UP` (*in module parameters*), 31
`Button.UP` (*in module parameters*), 31
`buttons` () (*in module ev3brick*), 16
`buttons` () (*InfraredSensor method*), 26

C

`clear` () (*ev3brick.display class method*), 18
`Color` (*class in parameters*), 30
`color` () (*ColorSensor method*), 25
`Color.BLACK` (*in module parameters*), 30
`Color.BLUE` (*in module parameters*), 30
`Color.BROWN` (*in module parameters*), 30
`Color.GREEN` (*in module parameters*), 30
`Color.ORANGE` (*in module parameters*), 30
`Color.PURPLE` (*in module parameters*), 30
`Color.RED` (*in module parameters*), 30
`Color.WHITE` (*in module parameters*), 30
`Color.YELLOW` (*in module parameters*), 30
`ColorSensor` (*class in ev3devices*), 25
`current` () (*ev3brick.battery class method*), 19

D

`dc` () (*Motor method*), 21
`Direction` (*class in parameters*), 29
`Direction.CLOCKWISE` (*in module parameters*), 29
`Direction.COUNTERCLOCKWISE` (*in module parameters*), 29
`distance` () (*InfraredSensor method*), 26
`distance` () (*UltrasonicSensor method*), 27
`drive` () (*DriveBase method*), 38
`drive_time` () (*DriveBase method*), 38
`DriveBase` (*class in robotics*), 38

E

`ev3brick` (*module*), 16
`ev3devices` (*module*), 20

F

`file` () (*ev3brick.sound class method*), 17

G

`GyroSensor` (*class in ev3devices*), 27

I

`image` () (*ev3brick.display class method*), 18
`ImageFile` (*class in parameters*), 31
`ImageFile.ACCEPT` (*in module parameters*), 31
`ImageFile.ANGRY` (*in module parameters*), 32
`ImageFile.AWAKE` (*in module parameters*), 32
`ImageFile.BACKWARD` (*in module parameters*), 32

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

ImageFile.BOTTOM_LEFT (*in module parameters*), 32
ImageFile.BOTTOM_RIGHT (*in module parameters*), 32
ImageFile.CRAZY_1 (*in module parameters*), 32
ImageFile.CRAZY_2 (*in module parameters*), 32
ImageFile.DECLINE (*in module parameters*), 31
ImageFile.DIZZY (*in module parameters*), 32
ImageFile.DOWN (*in module parameters*), 32
ImageFile.EV3 (*in module parameters*), 32
ImageFile.EV3_ICON (*in module parameters*), 32
ImageFile.EVIL (*in module parameters*), 32
ImageFile.FORWARD (*in module parameters*), 31
ImageFile.KNOCKED_OUT (*in module parameters*), 32
ImageFile.LEFT (*in module parameters*), 31
ImageFile.MIDDLE_LEFT (*in module parameters*), 32
ImageFile.MIDDLE_RIGHT (*in module parameters*), 32
ImageFile.NEUTRAL (*in module parameters*), 32
ImageFile.NO_GO (*in module parameters*), 32
ImageFile.PINCHED_LEFT (*in module parameters*), 32
ImageFile.PINCHED_MIDDLE (*in module parameters*), 32
ImageFile.PINCHED_RIGHT (*in module parameters*), 32
ImageFile.QUESTION_MARK (*in module parameters*), 31
ImageFile.RIGHT (*in module parameters*), 31
ImageFile.SLEEPING (*in module parameters*), 32
ImageFile.STOP_1 (*in module parameters*), 31
ImageFile.STOP_2 (*in module parameters*), 32
ImageFile.TARGET (*in module parameters*), 32
ImageFile.THUMBS_DOWN (*in module parameters*), 31
ImageFile.THUMBS_UP (*in module parameters*), 32
ImageFile.TIRED_LEFT (*in module parameters*), 32
ImageFile.TIRED_MIDDLE (*in module parameters*), 32
ImageFile.TIRED_RIGHT (*in module parameters*), 32
ImageFile.UP (*in module parameters*), 32
ImageFile.WARNING (*in module parameters*), 32
ImageFile.WINKING (*in module parameters*), 32
InfraredSensor (*class in ev3devices*), 26

L

light () (*in module ev3brick*), 16

M

Motor (*class in ev3devices*), 20

P

parameters (*module*), 29
pause () (*StopWatch method*), 37
Port (*class in parameters*), 29
Port.A (*in module parameters*), 29
Port.B (*in module parameters*), 29
Port.C (*in module parameters*), 29
Port.D (*in module parameters*), 29
Port.S1 (*in module parameters*), 29
Port.S2 (*in module parameters*), 29
Port.S3 (*in module parameters*), 29
Port.S4 (*in module parameters*), 29
presence () (*UltrasonicSensor method*), 27
pressed () (*TouchSensor method*), 25
print () (*in module tools*), 37

R

reflection () (*ColorSensor method*), 26
reset () (*StopWatch method*), 37
reset_angle () (*GyroSensor method*), 28
reset_angle () (*Motor method*), 21
resume () (*StopWatch method*), 37
rgb () (*ColorSensor method*), 26
robotics (*module*), 38
run () (*Motor method*), 21
run_angle () (*Motor method*), 22
run_target () (*Motor method*), 22
run_time () (*Motor method*), 21
run_until_stalled () (*Motor method*), 23

S

set_dc_settings () (*Motor method*), 24
set_pid_settings () (*Motor method*), 24
set_run_settings () (*Motor method*), 24
SoundFile (*class in parameters*), 32
SoundFile.ACTIVATE (*in module parameters*), 33
SoundFile.AIR_RELEASE (*in module parameters*), 35
SoundFile.AIRBRAKE (*in module parameters*), 35
SoundFile.ANALYZE (*in module parameters*), 33
SoundFile.BACKING_ALERT (*in module parameters*), 35
SoundFile.BACKWARDS (*in module parameters*), 33
SoundFile.BLACK (*in module parameters*), 34
SoundFile.BLUE (*in module parameters*), 34
SoundFile.BOING (*in module parameters*), 33
SoundFile.BOO (*in module parameters*), 33
SoundFile.BRAVO (*in module parameters*), 34
SoundFile.BROWN (*in module parameters*), 34
SoundFile.CAT_PURR (*in module parameters*), 35
SoundFile.CHEERING (*in module parameters*), 33
SoundFile.CLICK (*in module parameters*), 36
SoundFile.COLOR (*in module parameters*), 33

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

SoundFile.CONFIRM (*in module parameters*), 36
SoundFile.CRUNCHING (*in module parameters*), 33
SoundFile.CRYING (*in module parameters*), 33
SoundFile.DETECTED (*in module parameters*), 33
SoundFile.DOG_BARK_1 (*in module parameters*), 35
SoundFile.DOG_BARK_2 (*in module parameters*), 35
SoundFile.DOG_GROWL (*in module parameters*), 35
SoundFile.DOG_SNIFF (*in module parameters*), 35
SoundFile.DOG_WHINE (*in module parameters*), 35
SoundFile.DOWN (*in module parameters*), 33
SoundFile.EIGHT (*in module parameters*), 35
SoundFile.ELEPHANT_CALL (*in module parameters*), 35
SoundFile.ERROR (*in module parameters*), 33
SoundFile.ERROR_ALARM (*in module parameters*), 33
SoundFile.EV3 (*in module parameters*), 34
SoundFile.FANFARE (*in module parameters*), 33
SoundFile.FANTASTIC (*in module parameters*), 34
SoundFile.FIVE (*in module parameters*), 35
SoundFile.FLASHING (*in module parameters*), 33
SoundFile.FORWARD (*in module parameters*), 33
SoundFile.FOUR (*in module parameters*), 35
SoundFile.GAME_OVER (*in module parameters*), 34
SoundFile.GENERAL_ALERT (*in module parameters*), 36
SoundFile.GO (*in module parameters*), 34
SoundFile.GOOD (*in module parameters*), 34
SoundFile.GOOD_JOB (*in module parameters*), 34
SoundFile.GOODBYE (*in module parameters*), 34
SoundFile.GREEN (*in module parameters*), 34
SoundFile.HELLO (*in module parameters*), 34
SoundFile.HI (*in module parameters*), 34
SoundFile.HORN_1 (*in module parameters*), 35
SoundFile.HORN_2 (*in module parameters*), 35
SoundFile.INSECT_BUZZ_1 (*in module parameters*), 35
SoundFile.INSECT_BUZZ_2 (*in module parameters*), 35
SoundFile.INSECT_CHIRP (*in module parameters*), 35
SoundFile.KUNG_FU (*in module parameters*), 33
SoundFile.LASER (*in module parameters*), 35
SoundFile.LAUGHING_1 (*in module parameters*), 33
SoundFile.LAUGHING_2 (*in module parameters*), 33
SoundFile.LEFT (*in module parameters*), 33
SoundFile.LEGO (*in module parameters*), 34
SoundFile.MAGIC_WAND (*in module parameters*), 33
SoundFile.MINDSTORMS (*in module parameters*), 34
SoundFile.MORNING (*in module parameters*), 34
SoundFile.MOTOR_IDLE (*in module parameters*), 35
SoundFile.MOTOR_START (*in module parameters*), 35
SoundFile.MOTOR_STOP (*in module parameters*), 35
SoundFile.NINE (*in module parameters*), 35
SoundFile.NO (*in module parameters*), 34
SoundFile.OBJECT (*in module parameters*), 33
SoundFile.OKAY (*in module parameters*), 34
SoundFile.OKEY_DOKEY (*in module parameters*), 34
SoundFile.ONE (*in module parameters*), 35
SoundFile.OUCH (*in module parameters*), 33
SoundFile.OVERPOWER (*in module parameters*), 36
SoundFile.RATCHET (*in module parameters*), 35
SoundFile.READY (*in module parameters*), 36
SoundFile.RED (*in module parameters*), 34
SoundFile.RIGHT (*in module parameters*), 33
SoundFile.SEARCHING (*in module parameters*), 33
SoundFile.SEVEN (*in module parameters*), 35
SoundFile.SHOUTING (*in module parameters*), 33
SoundFile.SIX (*in module parameters*), 35
SoundFile.SMACK (*in module parameters*), 33
SoundFile.SNAKE_HISS (*in module parameters*), 35
SoundFile.SNAKE_RATTLE (*in module parameters*), 35
SoundFile.SNEEZING (*in module parameters*), 33
SoundFile.SNORING (*in module parameters*), 33
SoundFile.SONAR (*in module parameters*), 35
SoundFile.SORRY (*in module parameters*), 34
SoundFile.SPEED_DOWN (*in module parameters*), 34
SoundFile.SPEED_IDLE (*in module parameters*), 34
SoundFile.SPEED_UP (*in module parameters*), 34
SoundFile.START (*in module parameters*), 34
SoundFile.STOP (*in module parameters*), 33
SoundFile.T_REX_ROAR (*in module parameters*), 35
SoundFile.TEN (*in module parameters*), 36
SoundFile.THANK_YOU (*in module parameters*), 34
SoundFile.THREE (*in module parameters*), 35
SoundFile.TICK_TACK (*in module parameters*), 34
SoundFile.TOUCH (*in module parameters*), 33
SoundFile.TURN (*in module parameters*), 34
SoundFile.TWO (*in module parameters*), 35
SoundFile.UH_OH (*in module parameters*), 33
SoundFile.UP (*in module parameters*), 33
SoundFile.WHITE (*in module parameters*), 34
SoundFile.YELLOW (*in module parameters*), 34
SoundFile.YES (*in module parameters*), 34
SoundFile.ZERO (*in module parameters*), 35

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.

speed () (*GyroSensor method*), 27
speed () (*Motor method*), 21
stalled () (*Motor method*), 23
Stop (*class in parameters*), 30
stop () (*DriveBase method*), 39
stop () (*Motor method*), 21
Stop.BRAKE (*in module parameters*), 30
Stop.COAST (*in module parameters*), 30
Stop.HOLD (*in module parameters*), 30
StopWatch (*class in tools*), 37

T

text () (*ev3brick.display class method*), 18
time () (*StopWatch method*), 37
tools (*module*), 37
TouchSensor (*class in ev3devices*), 25
track_target () (*Motor method*), 23

U

UltrasonicSensor (*class in ev3devices*), 27

V

voltage () (*ev3brick.battery class method*), 19

W

wait () (*in module tools*), 37

LEGO, the LEGO logo and MINDSTORMS are trademarks of the LEGO Group.
©2019 The LEGO Group.