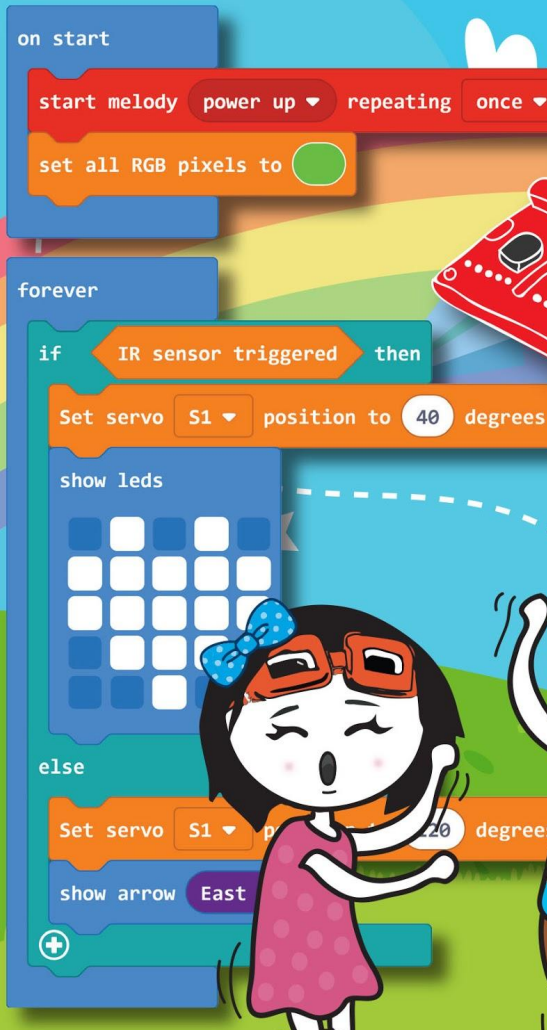
Cytron Technologies

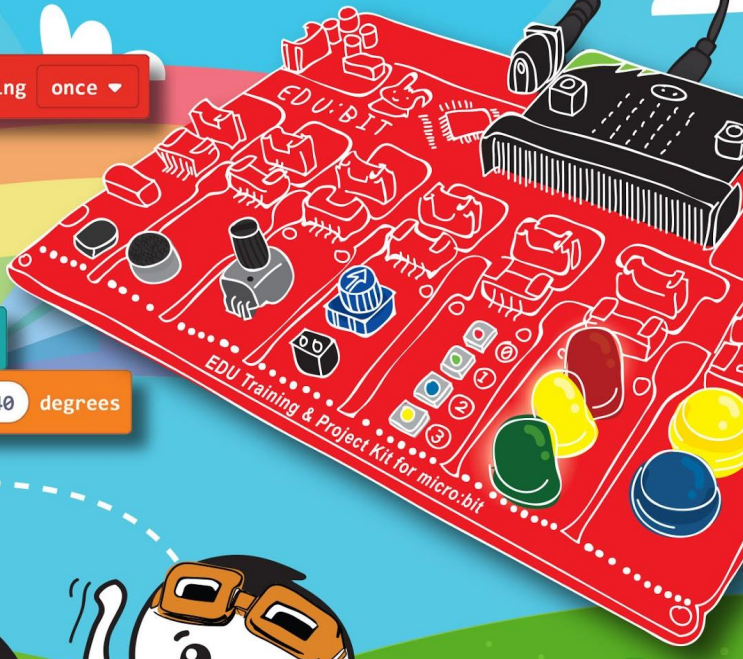
CLASSROOM STEAM EDUCATION

EDU:BIT 本 v1.0
(日本語訳)

Explore STEM & Coding with

EDU:BIT





RERO EDUTEAM @ CYTRON より

さんへ

（生徒名）

「Micro:bit（マイクロビット）」って知っていますか？子供たちが楽しく簡単にプログラミングを学べるよう、イギリスで開発された小さなマイコンボードで、世界中で販売されています。

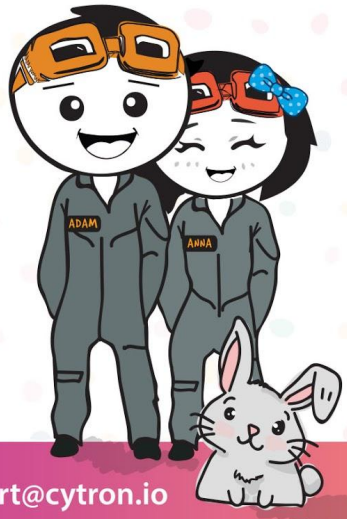
Cytronのエンジニアはこれを改良し、プログラミングを少しずつ学べるよう、EDU:BITボードを開発しました。音楽を再生する**Music Bit**（ピエゾブザー／オーディオジャック内蔵）、音を感知する**Sound Bit**（信号機ビット）、アナログ信号を制御する**Potentio Bit**（ポテンシオビット）、ヒトやモノの動きを感知する**IR Bit**（IRビット）、RGB値を制御して様々な色を表示する**RGB Bit**（RGB ビット）、赤、黄、緑LEDの**Traffic Light Bit**（信号機ビット）、また、**Button Bit**という大きなボタンも備わっています。**DC Motor**と**Servo Motor**（サーボモーター）も付属しています。プログラミングに最適な学習用素材です。

さっそく始めてみましょう。各チャプターでは、じゃんけん・ヘビとはしご・鬼ごっこ・ツイスター・「サイモンさんは言いました」ゲームなど、子供にお馴染みのゲームを取り上げ、それらを体験しながら学習を進めていきます。ガイドに従って一歩一歩、他の生徒とゲームを楽しんでください。プログラムを変更することで、ゲーム内容をより面白くすることも可能です。

各チャプターの最後に練習問題があり、アプリケーションの作り方を復習・応用していきます。やってみて、分からないことがあったら、いつでも相談してください。

準備はいいですか？
楽しい学習体験の始まりです!

アダムとアンナより



EDU:BITを使った STEM及びプログラミング学習用 トレーニング＆プロジェクトキット

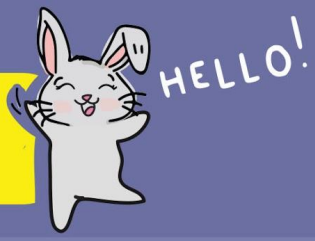
著者：Cheryl Ng, SC Lim & Adrian Teo
イラスト：Suhana Oazmi

翻訳者 : Yasuo Mitani

2020

発行者 Cytron Technologies

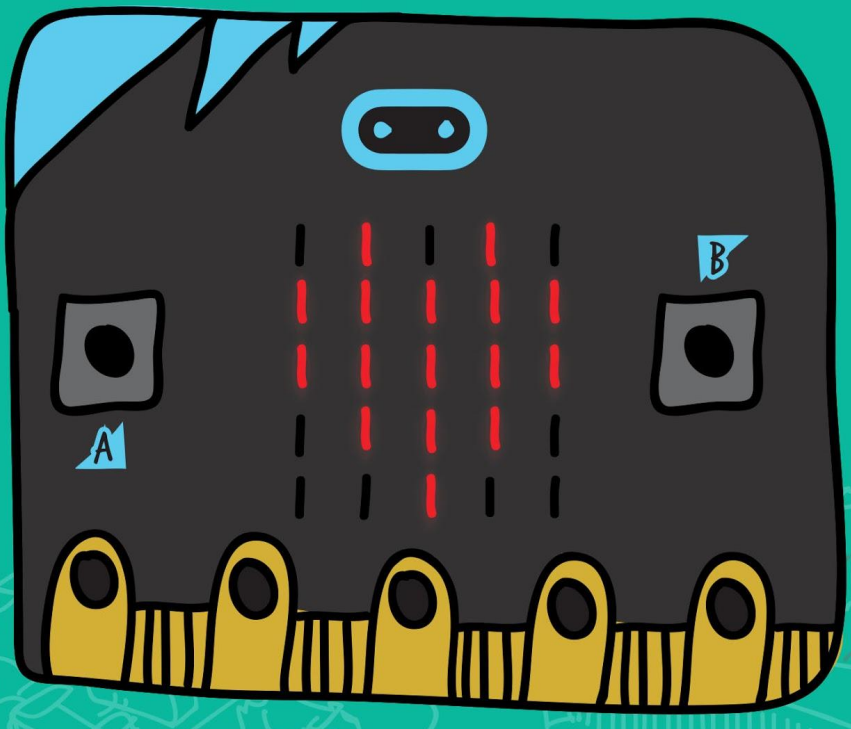
もくじ

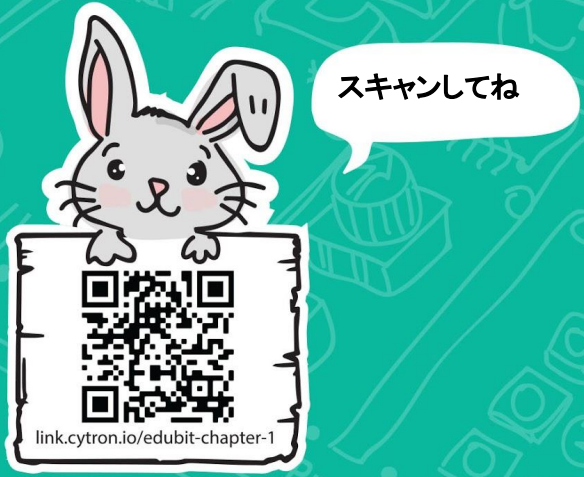


チャプター 1

＞ハローワールド

micro:bitのLEDを光らせよう

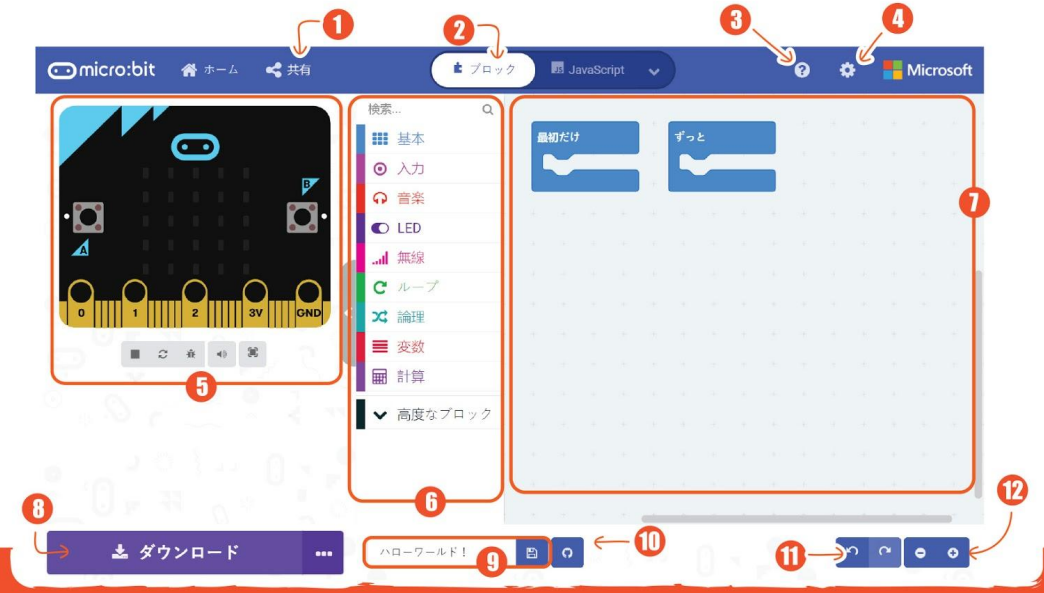




作ってみよう！

Step 1 ブラウザーで次のURLにアクセスします。http://makedcode.microbit.org
「新しいプロジェクト」をクリック：プロジェクト名を入力し、**「作成」**をクリック。

Microsoft MakeCodeのエディター画面が表示されます。EDU:BIT プログラミングは、ブロックをドラッグ &ドロップで組み立てていくだけなので、とても簡単です。

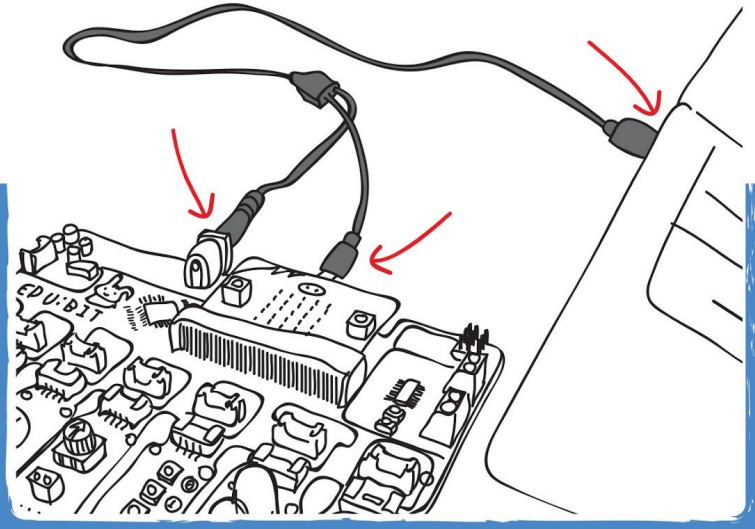


1) プログラムを他の人と共有するときに使います。
2) プログラミング方法をBlocks, JavaScript, Pythonから選択できます。
3) ヘルプメニューを開きます。
4) 拡張機能の追加などの設定ができます。
5) **シミュレーター**- プログラムがmicro:bit上でどう動作するかを、確認できます。
6) **ツールボックス**- プログラムに使うブロックが種類別にまとめられています。ブロックの色は種類ごとに同じです。
7) **ワークスペース**- ここでブロックを組み立ててプログラムを作成します。
8) プログラムをパソコンにダウンロードします。
9) プログラムに名前をつけます。
10) GitHubリポジトリを作成します。
11) 操作を戻したり、また進めたりします。
12) 表示を拡大したり縮小したりします。

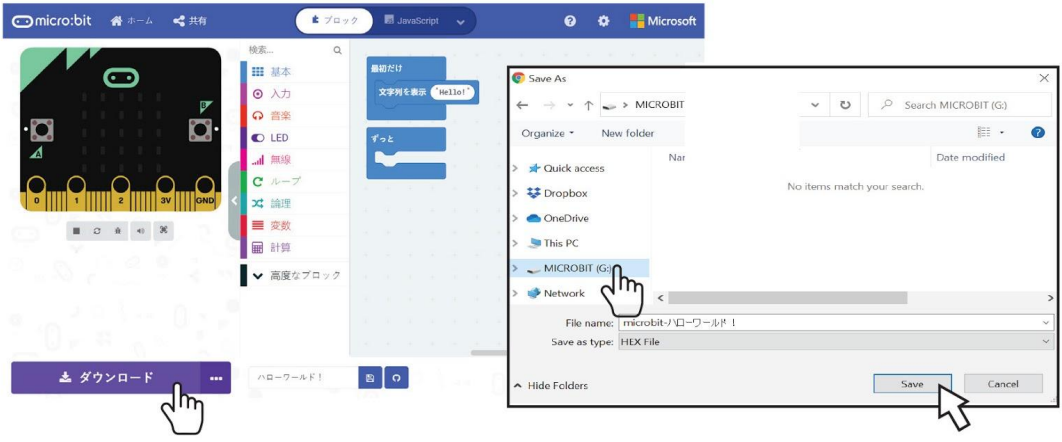
Step 2 [基本]をクリックし、[文字列を表示 _]ブロックを選択。

Step 3 [文字列を表示 _]ブロックを[最初だけ]ブロックのへこみ部分に合わせます。

Step 4 EDU:BITとPCは以下の図のように、USBケーブルで接続します。EDU:BITを起動するには電源ボタンを ONにしてください。



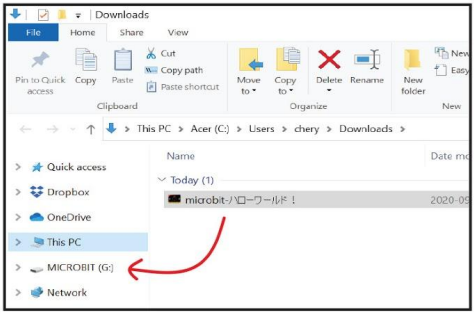
Step 4 **[ダウンロード]**ボタンをクリックするとポップアップウィンドウが開きます。MICROBITドライブに保存してください。
「ダウンロード完了」と表示されたらウィンドウを閉じてください。



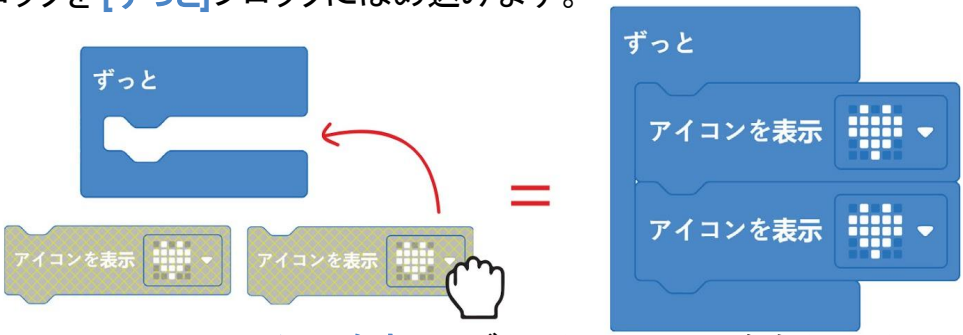
このようにプログラムをコピーする作業をフラッシングというのよ。コピー中はmicro:bitの後ろのオレンジ色の LEDがピカピカ光って、コピーが終わるとプログラムが自動的に実行されるのよ。

ノート！

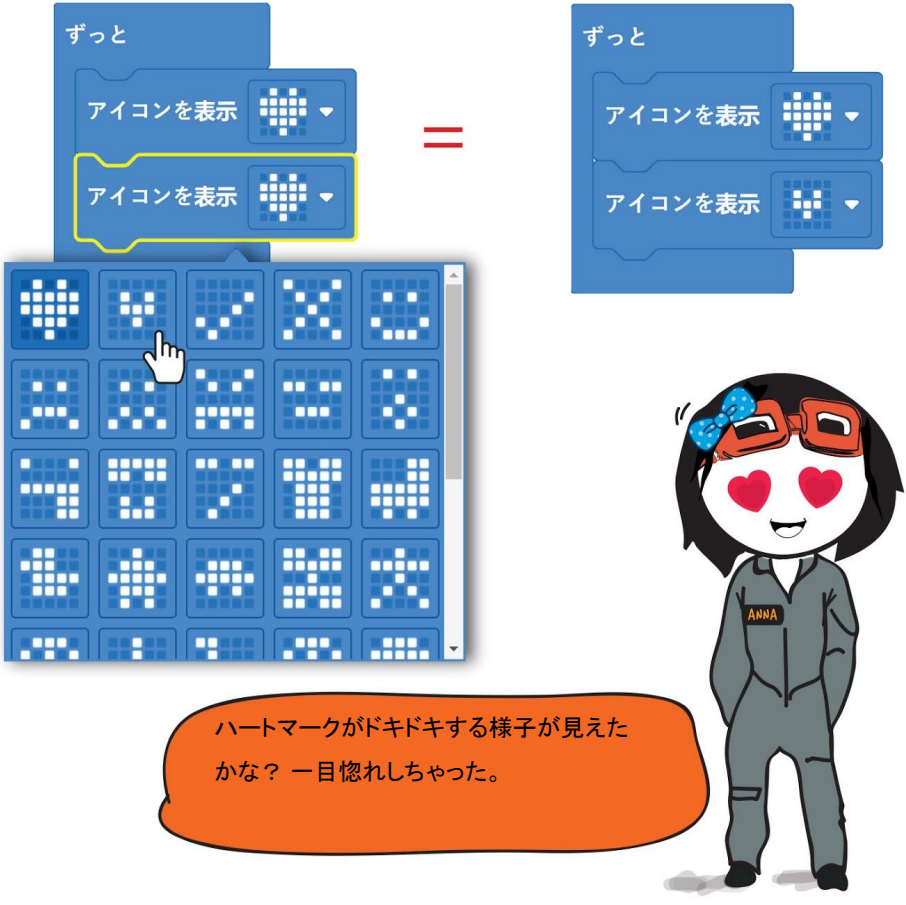
ポップアップウィンドウが表示されない場合は、ブラウザが所定のダウンロード先にファイルを自動的にダウンロードしたことを意味します。ウィンドウの下部に表示されるダウンロードした　.hexファイルを右クリックし、"フォルダに表示 "を選択します。 ダウンロードした　"micromit-xxxx.hex "ファイルをクリックして、フラッシュドライブにファイルをコピーするように、 MICROBITドライブにドラッグしてください。



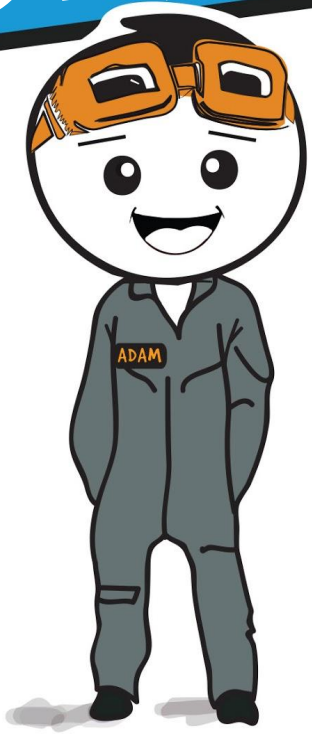
Step 6 [基本]をクリックし、[アイコンを表示_]ブロックをクリックします。これを繰り返して、別の[アイコンを表示_]ブロックを追加します。[アイコンの表示_] ブロックを [ずっと]ブロックにはめ込みます。



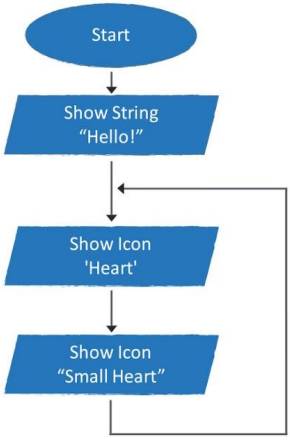
Step 7 2つ目の[アイコンを表示_]ブロックのアイコンを左クリックし、ポップアップウィンドウから「小さなハート」のデザインを選択します。そのプログラムをEDU:BITにフラッシングしてください。

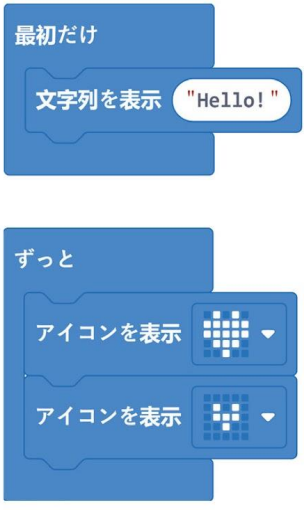


プログラムの かいせつ

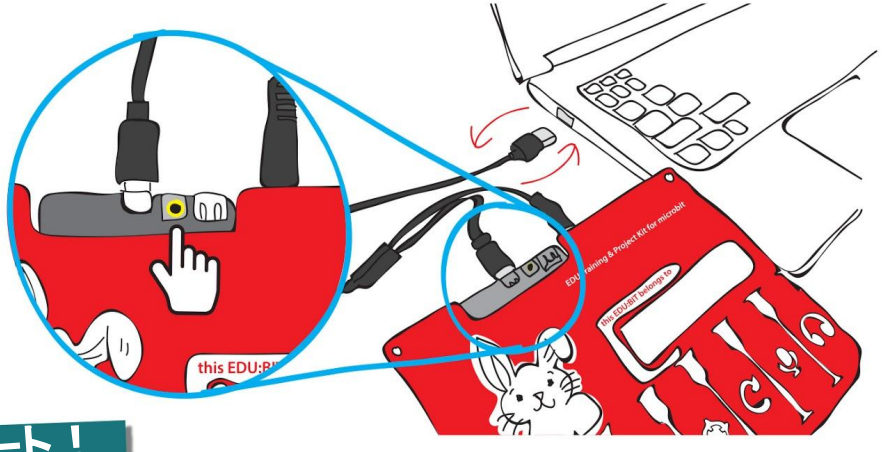


「Hello!」は1回しかスクロール表示されないけど、ハートのアイコンは何回も表示されるのに 気付いたかな？ なぜか分かるかな？





[最初だけ] ブロックは、起動時にプログラムを実行します (最初だけ)。
[ずっと] ブロックは、コードを何度も何度も実行します (ずっと)。



ノート！

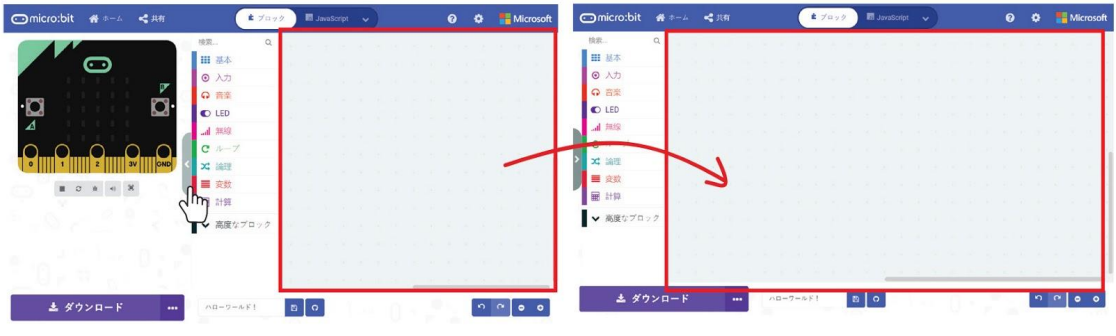
プログラムを再実行したい場合は、RESETボタンを押すか、USBケーブルを抜き差ししてください。

アドバイス #1

ブロックを削除したい場合は、削除したいブロックを、ツールボックスまでドラッグ &ドロップすれば OK です。削除するときはツールボックスがゴミ箱表示になります。または、ブロックを右クリックして「ブロックの削除」を選択することもできます。

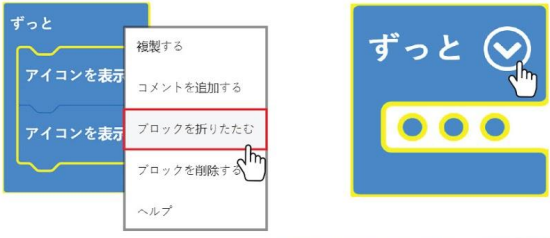
アドバイス #2

シミュレーターを使用しない場合、タブをクリックすることで、ワークスペースが広くなります。



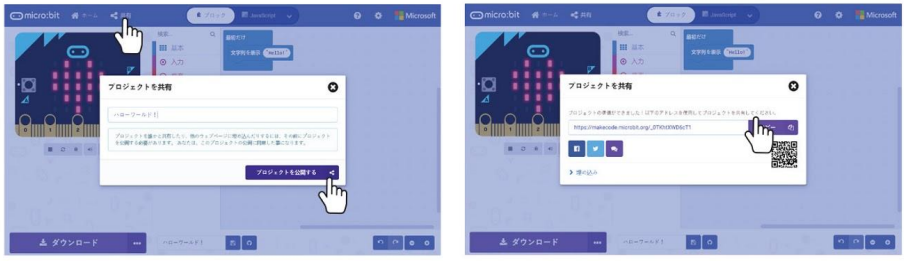
アドバイス #3

ブロックの内部を右クリックして「ブロックを折りたたむ」を選択すると、ブロックを折りたたむことができます。折りたたまれたブロックを展開するには、　アイコンをクリックします。



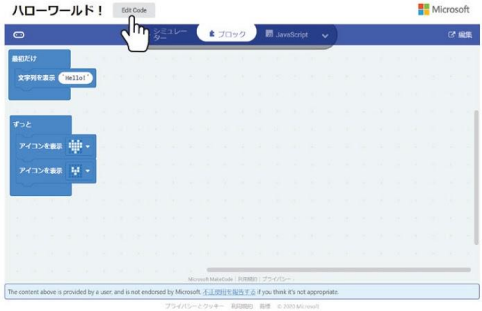
アドバイス #4

プロジェクトの URLを先生や友達に送ることで、プログラムを共有することができます。 [共有]をクリックし、ポップアップウィンドウの [プロジェクトを公開する]をクリックします。すると、あなたのプロジェクトの URLが表示されますので、[コピー]ボタンをクリックしてプロジェクトの URLをコピーしてください。



アドバイス #5

共有されたプロジェクトの URLを先生や友達がブラウザで開くと、次のようなページが表示されます。プログラムを参照するだけでなく、[Edit Code]ボタンでプログラムを編集することもできます。



アドバイス #6

デバイスをペアリングすることで、ワンクリックでダウンロードが可能です。歯車のアイコンをクリックして、「デバイスのペアリング」を選択します。

ノート！

micro:bitに最新のファームウェアを搭載し、EdgeまたはChromeブラウザを使用する必要があります。ファームウェアをアップデートする必要がある場合は、こちらの説明を参照してください。
http://microbit.org/get-started/user-guide/firmware/

EDU:BITがPCに接続されていることを確認し、ポップアップウィンドウの**[デバイスを接続する]**ボタンをクリックします。次に、リストから BBC micro:bit CMSIS-DAPまたはDAPLink CMSIS-DAPを選択し、**[接続]**ボタンをクリックします。

お使いのデバイスをペアリングした後、**[ダウンロード]**ボタンをクリックすると、EDU:BITに直接プログラムをフラッシングすることができます。試してみてください。

デバイスのペアリングが上手く行かないときは、
http://makecode.microbit.org/device/usb/webusb/troubleshoot
を見てね。

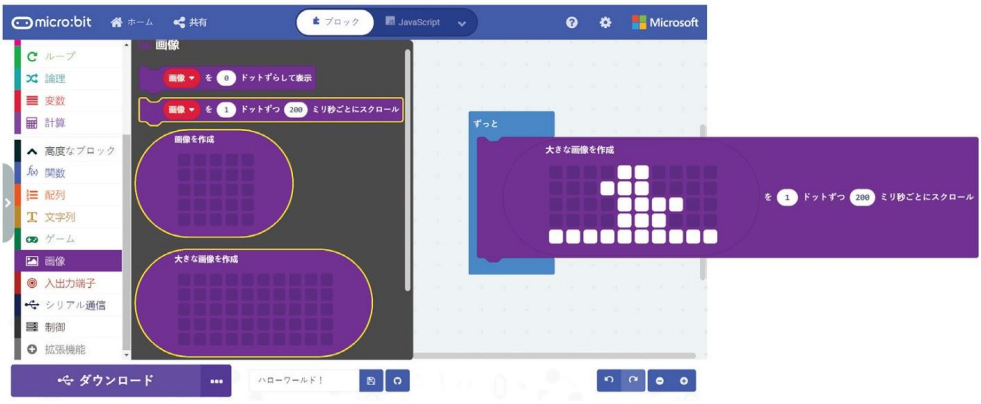
他のブロックも使ってみよう

#1 **[LED画面に表示]**ブロックを使って自分のアイコンをデザインしたり、**[数を表示]**ブロックを使って数字を表示したりすることができます。

ミリ秒(ms)は1,000分の1秒のことね。

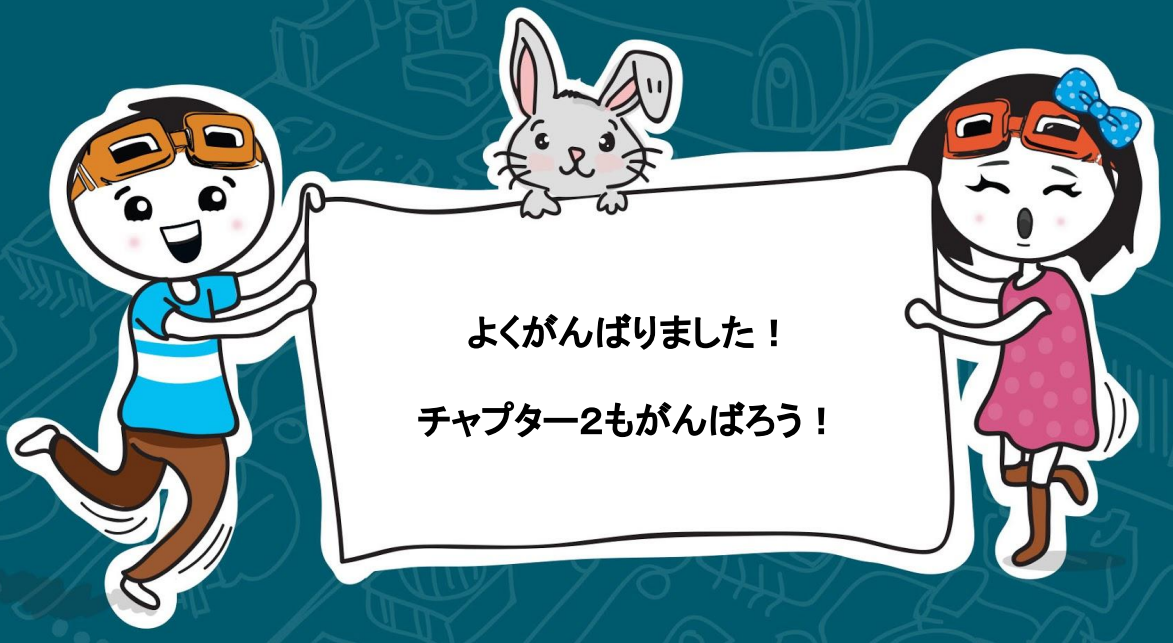
#2 プログラムの実行速度を遅くするためには、**[一時停止（ミリ秒）_]**ブロックを追加します。設定したミリ秒 (ms)間、プログラムを一時停止します。

#3 LED上で画像をスクロールさせるには、**高度なブロック**配下の**[画像]**から、**[_を_ドットずつ_ミリ秒ごとにスクロール]**ブロックを選択し、**[画像を作成]**または**[大きな画像を作成]**と一緒に使用します。



練習問題

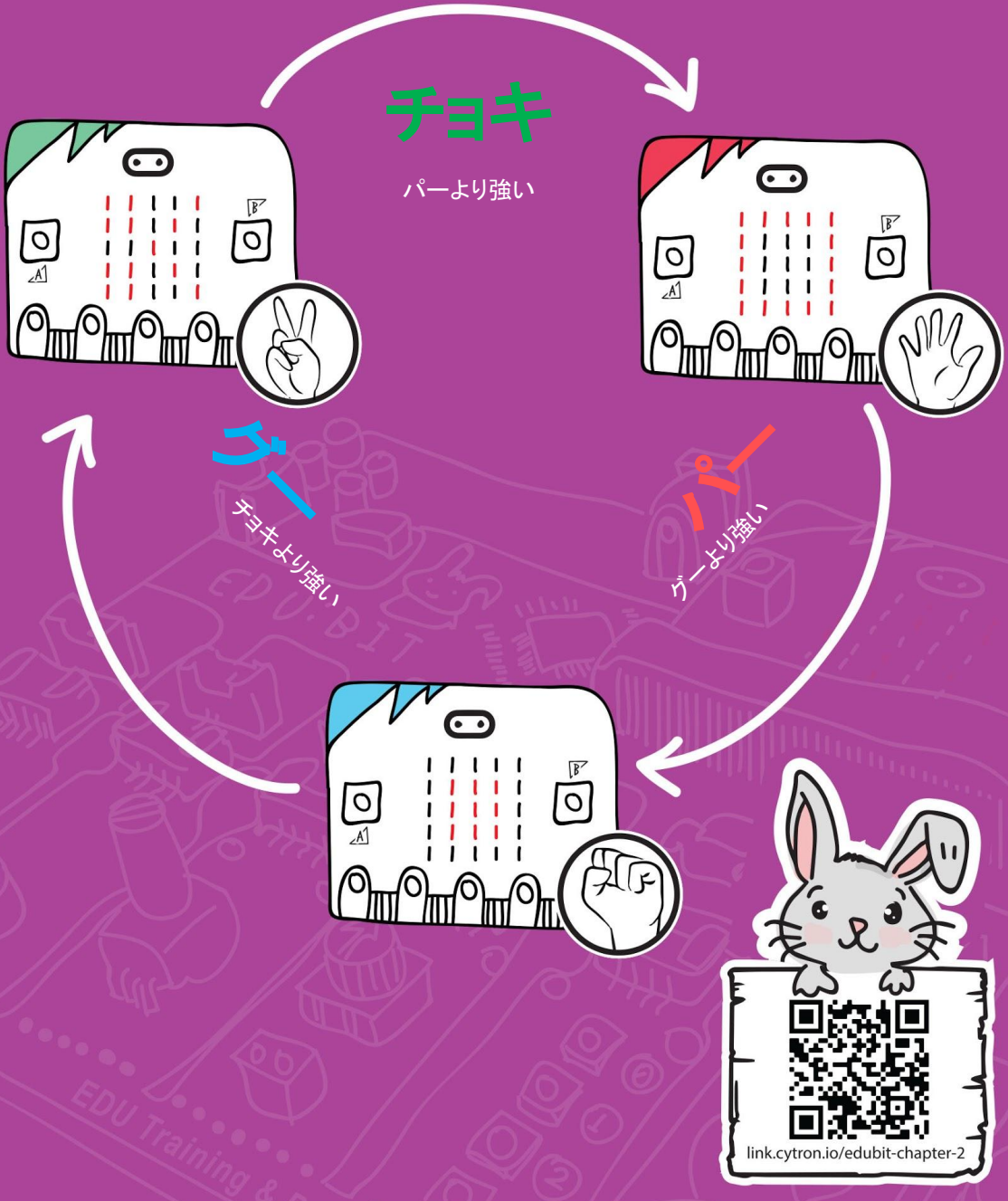
ECU:BITにデジタルアナウンス機能をプログラムしてみよう	
最初だけ	クラス名の他に自由なアニメーションを表示してみよう。
ずっと	今日の日付と何かクラスへのメッセージを表示させつづけてみよう。



チャプター 2

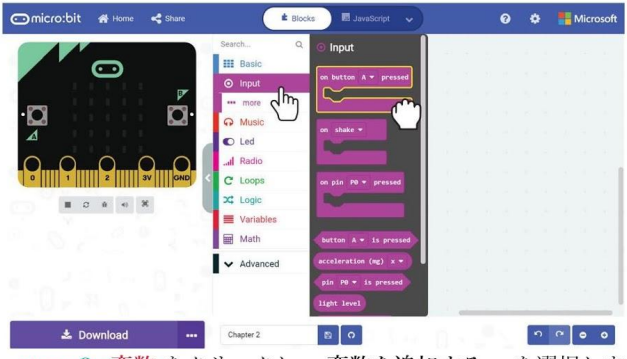
じゃんけん大会で遊ぼう！

micro:bitのボタン操作とButton Bit

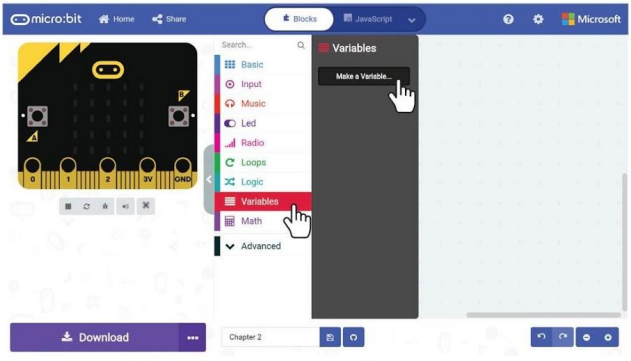


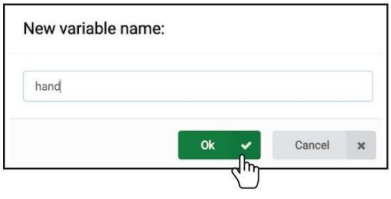
作ってみよう！

Step 1 https://makecode.microbit.org/ にアクセスして（既に MakeCode Editor を使用している場合はホームアイコンをクリック）、新しいプロジェクトを作成（さくせい）してください。[入力]から[ボタン_が押されたとき]ブロックを選択します。

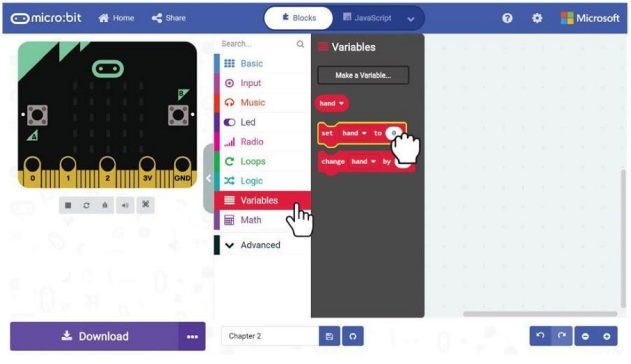


Step 2 [変数]をクリックし、[変数を追加する...]を選択します。
ポップアップウィンドウに「**hand**」と入力して OK をクリックします。

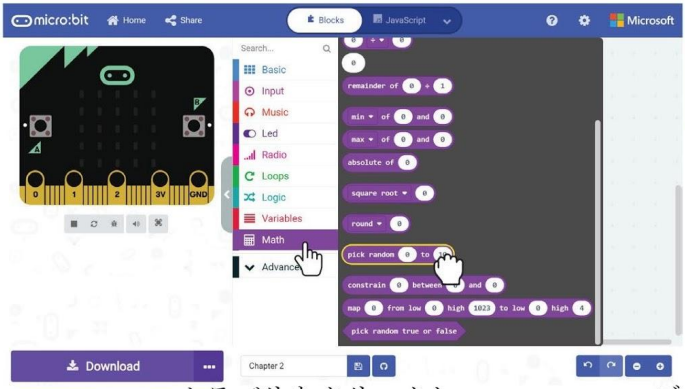




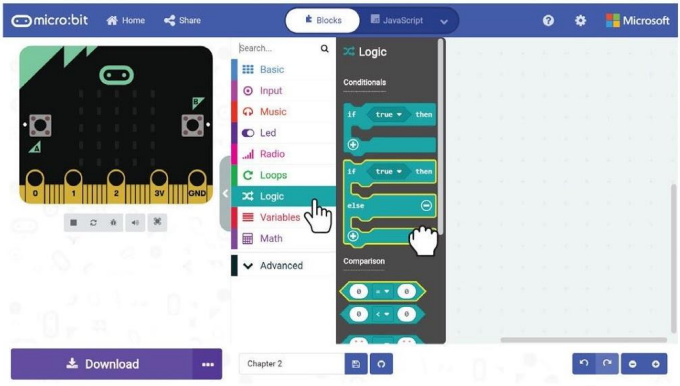
Step 3 [変数]をクリックし、[変数_を_にする]ブロックを選択。[ボタン A が押されたとき]ブロックにはめ込（こ）みます。

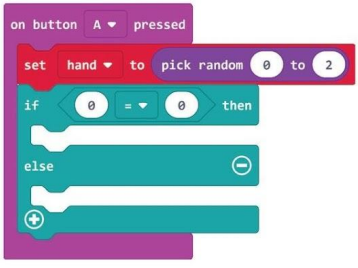


Step 4 **[計算]**をクリックし、**[pick random_ to_]**ブロックを選択します。数字の 10 を **2** に変更します。

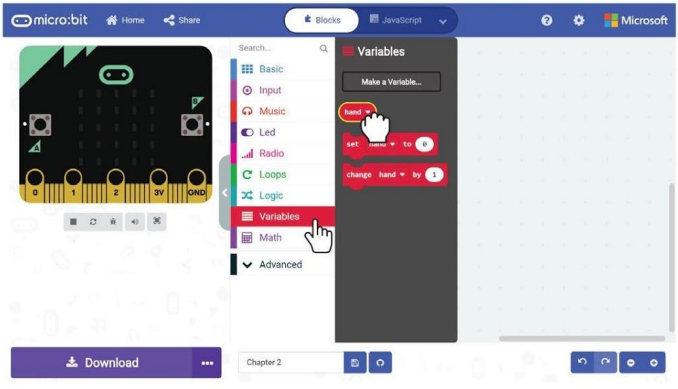


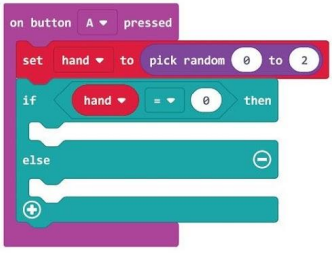
Step 5 **[logic]**カテゴリをクリックし、**[if-then-else]**ブロックと**[_=_]**比較ブロックを選択します。比較ブロックを「**if**」の所（ところ）にはめ込みます。





Step 6 **[Variables]**をクリックし、**[hand]**ブロックを選択。ブロックを比較ブロックにはめ込みます。

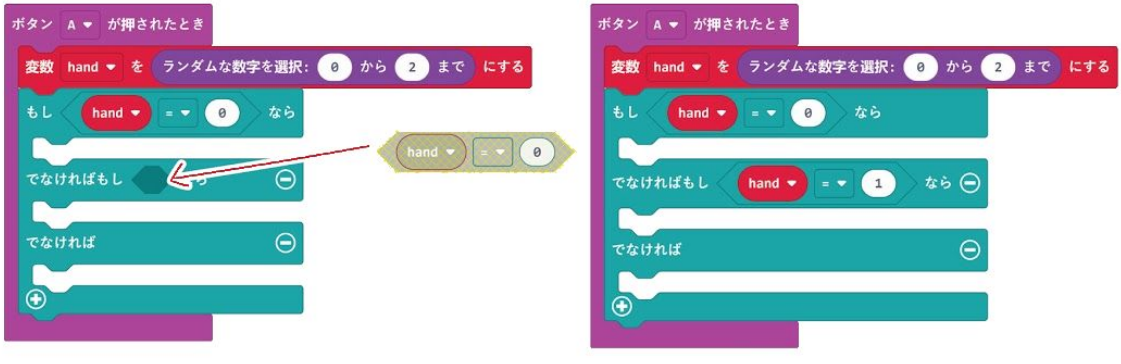




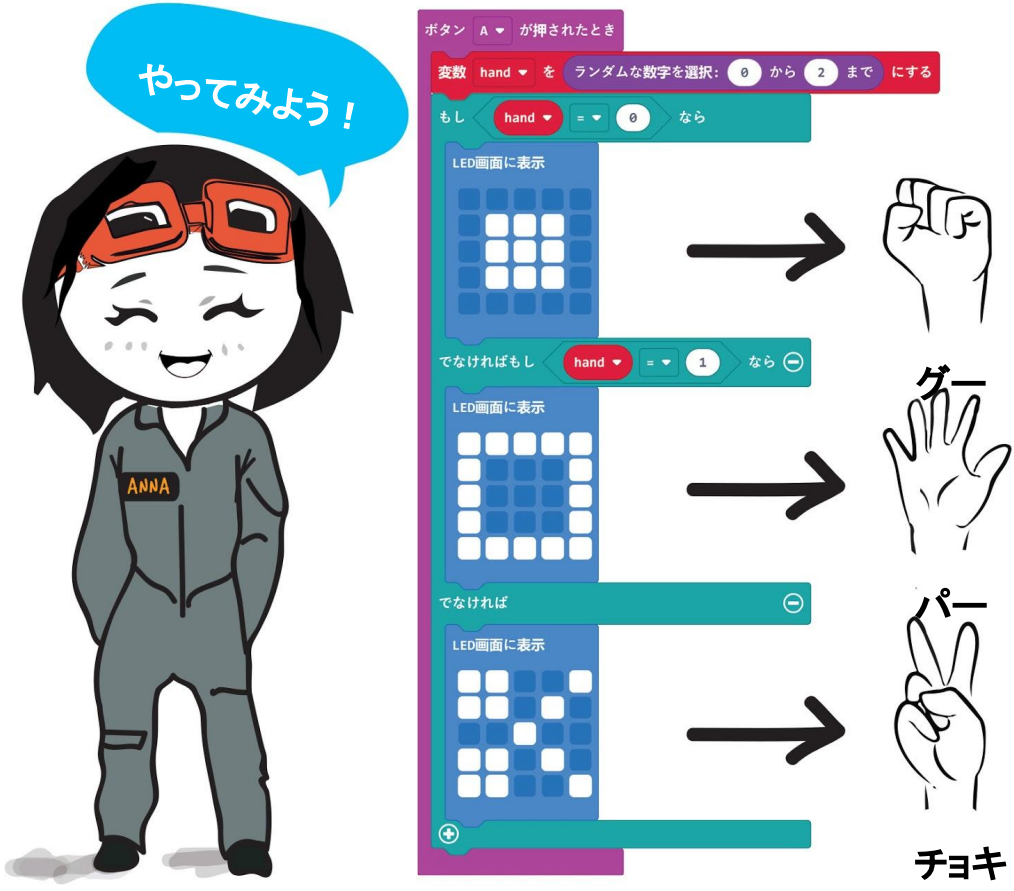
Step 7  アイコンをクリックして、「もし」ブロックに**[でなければ]**条件を追加します。

Step 8 比較ブロックを右クリックし、「**複製する**」を選択します。

Step 9 複製されたブロックを「でなければ」にはめ込み、数字の 0を**1**に変更します



Step 10 **[基本]**の**[LED画面に表示]**ブロックを「**もし**」「**でなければもし**」「でなければ」に追加しましょう。**[LED画面に表示]**ブロック内のボックスをクリックして、以下の例のような画像を作りましょう。



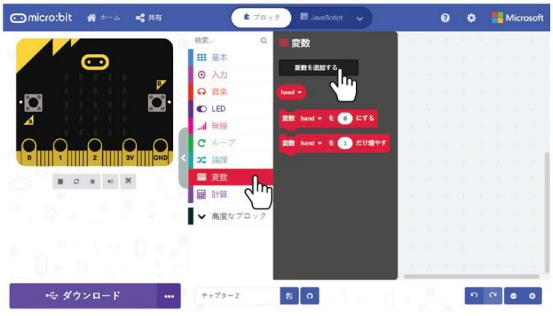
EDU:BITにプログラムをフラッシングして、お友達とジャンケンをしてみよう。micro:bitのボタンAを押すか、黄色のボタンを押すと、LED「グー」「パー」「チョキ」のいずれかが表示されるよ。

カラム：プロジェクトを保存したい場合は、保存ボタンをクリックして、PC上のフォルダに保存してください。

各プレイヤーにライフを3つ割り当てて、このゲームを面白くしよう。'lives'という名前の新しい変数を作って、以下のブロックを追加するんだ。

Step 11 [入力]から[ボタン_が押された時]ブロックを選択します。ブロックを複製し、それぞれの設定を「ボタンB」「ボタンA+B」に変更します。

Step 12 [変数]から[**変数を追加...**]を選択します。ポップアップウィンドウに「**lives**」と入力し、「OK」をクリックします。

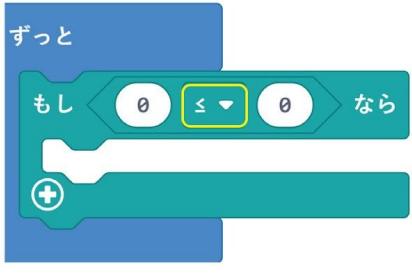


Step 13 [変数]から[変数_を_にする]ブロックを選択し、[基本]:[最初だけ]ブロックにはめ込みます。変数を「**lives**」に設定し、値を**3**に変更します。

Step 14 **[変数]**から**[変数_を_だけ増やす]**ブロックを選択し、**[ボタン_が押された時]**ブロックにはめ込みます。変数を '**lives**'に設定し、値を**-1**に変更します。

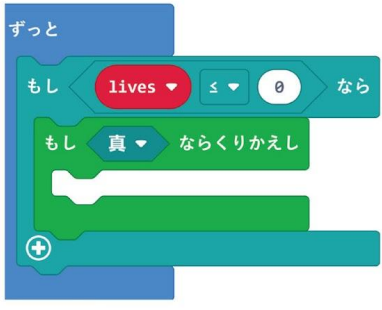
Step 15 **[基本]**をクリックし、**[数を表示]**ブロックを選択します。それを複製して、**[ボタン_が押された時]**ブロックの両方にはめ込みます。

Step 16 **[論理]**をクリックして、**[もし-なら]**ブロックと**[_ = _]**ブロックをプログラムに追加します。これらを**[基本]**：**[ずっと]**ブロックにはめ込み、符号を'≤'に変更します。

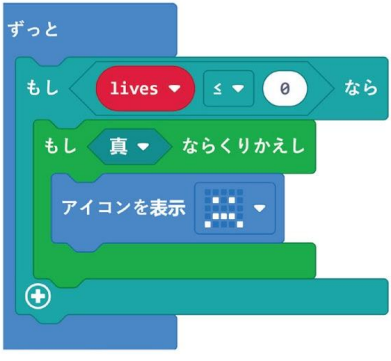


Step 17 [変数]カテゴリをクリックし、[lives]ブロックを選択します。複製して、[数を表示]ブロックと[_ = _]ブロックの左側にはめ込みます。

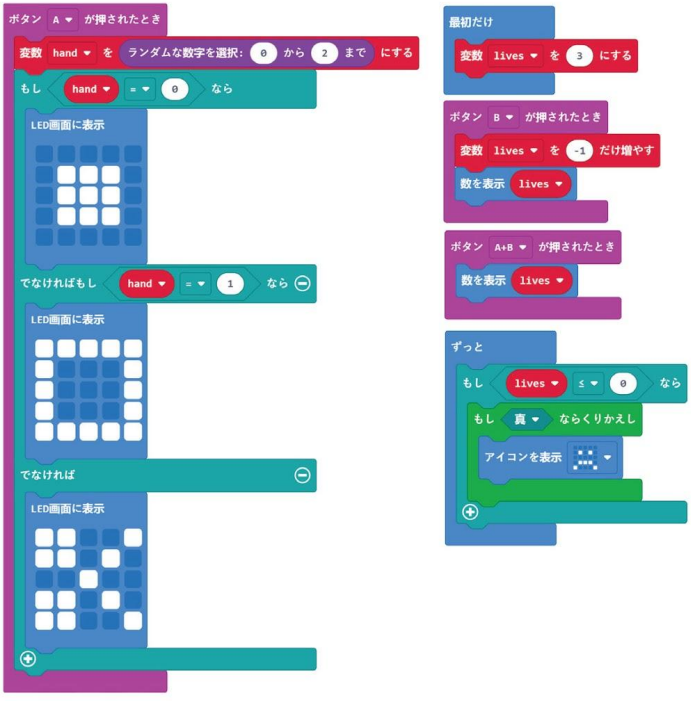
Step 18 [ループ]から[もし_ならくりかえし]ブロックを選択し、[もし-なら]ブロックにはめ込みます。



ステップ19 [基本]から[アイコンを表示]ブロックを選択します。[もし-ならくりかえし]ブロックにはめ込み、アイコンを「 **かなしい顔** 」に変更します。



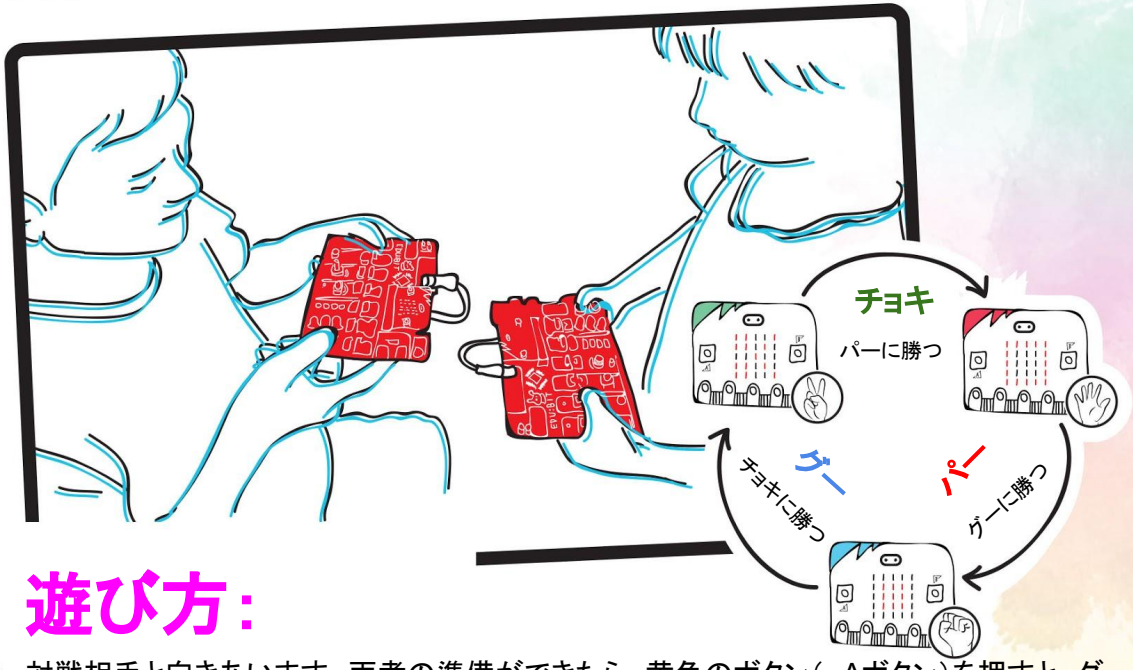
Step 20 これがプログラムの完成形です。これを EDU:BITにフラッシングします。お友達とジャンケン大会で遊んで、ジャンケン王者を決定しましょう。



ここでヒント。
すべてのブロックは色分けされているの。ツールボックスから、同じ色を探して、必要なブロックを見つけてみてね。
または、検索ボックスにキーワードを入力して探すこともできるわよ。

遊んでみよう！

ジャンケン大会 - 応用編



遊び方：

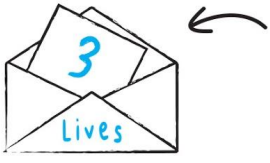
- 対戦相手と向きあいます。両者の準備ができたら、黄色のボタン（ Aボタン）を押すと、グー、パー、チョキのいずれかが表示されます。
- 結果を見比べて勝敗を確認しましょう。
- 負けた場合、自分の EDU:BITの青いボタン（ボタン B）を1回押してライフをマイナス 1します。
- 黄色と青色のボタンを同時に押すと、残りのライフ数を確認できます。
- 3回負けた場合はゲームオーバーとなり、そのプレイヤーの EDU:BITはかなしい顔を表示します。

ノート！

- **勝負をやり直す場合は、リセットする必要があります。**
- **勝負相手がいない場合でも、シミュレーターと対戦可能です。**

プログラムのかいせつ

コンピュータプログラミングでは、変数と呼ばれるものを使って、プログラム実行中の情報や値を格納します。変数は、文字が書かれた紙が入ったラベル付きの封筒のようなものだと考えてください。紙を取り出して、新しい文字を書いた別の紙に置き換えることができます。先ほどのプログラムでは、"lives "という変数を作成し、初期値として 3を代入しました。



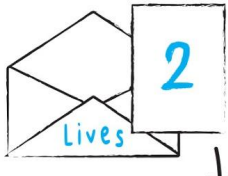
紙に書かれた3が文字です。

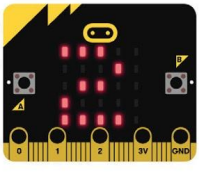
'Lives'と書かれた封筒は変数になります。

そして、ボタンBが押されるたびに値を -1していきます。

ボタンBを押すと、「3」と書かれた紙を封筒から取り出し、「2」(つまり3 - 1 = 2)と書かれた別の紙と交換します。

ボタン A+B を同時に押すと、LED に変数の現在値が表示されます。



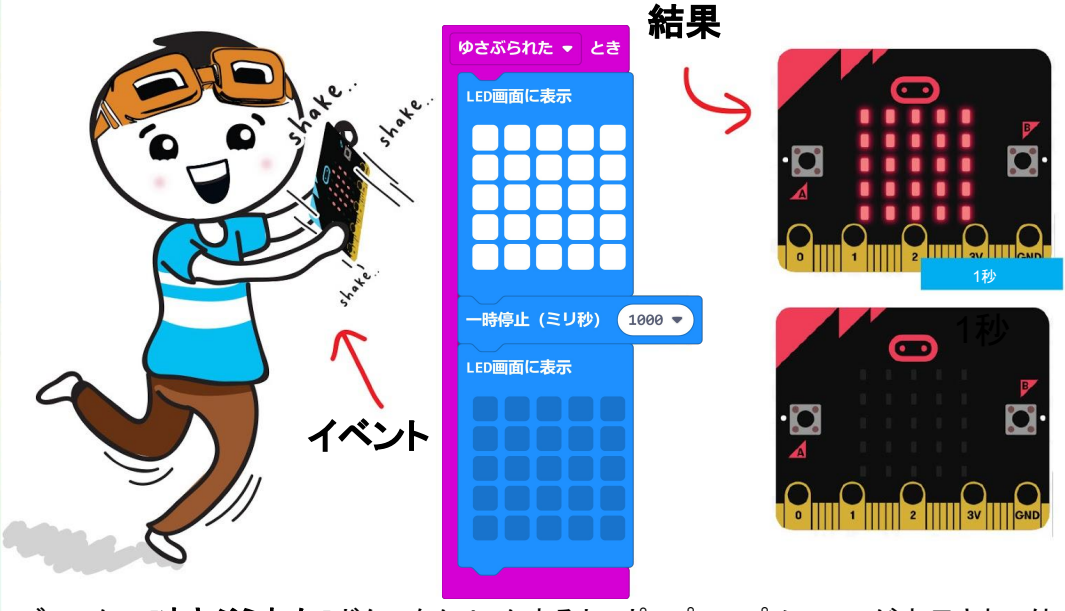


A+Bボタンを押したら、封筒の中に入っている紙を取り出し、そこに書かれている文字を「参照(読む)」します。

他のブロックも使ってみよう

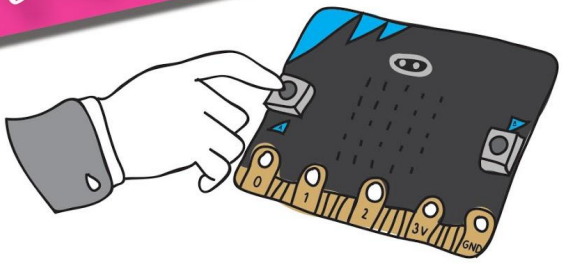
[ボタン_が押されたとき]ブロックの他にも、ツールボックスの**[入力]**から他のブロックをイベントベースのプログラミングに使用することができます。ユーザがボタンを押したり、ボードを振ったりしたときの動作を「イベント」と呼びます。

例えば、以下のコードを実行すると、ボードを振る度にLEDが1秒間点灯します。試してみましょう。

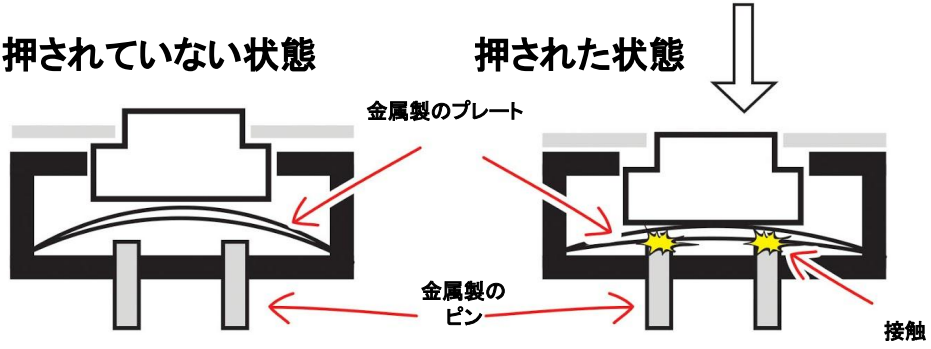


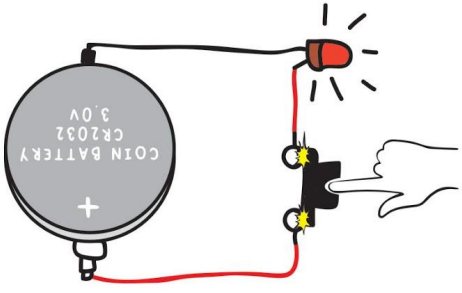
ブロックの**[ゆさぶられた]**ボタンをクリックすると、ポップアップメニューが表示され、他の選択肢が表示されます。EDU:BITをプログラムして、これらの異なるアイコンを表示するようにしてみてください。色々やって、楽しんでください！

面白い事実！



プッシュボタンは、押されたか押されていないかの2つの状態のみが判定可能な入力デバイスまたはスイッチです。

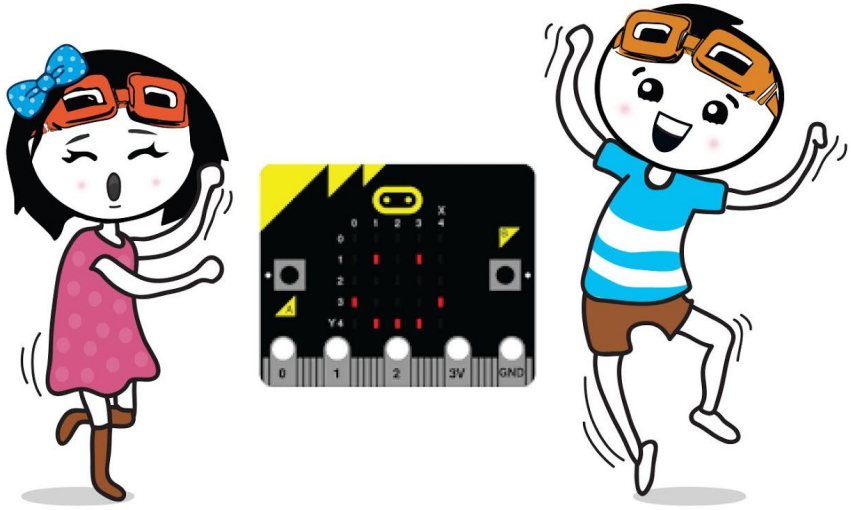




プッシュボタンを押すと電気回路がつながってし、LEDが点灯します！プッシュボタンを離すとどうなるでしょう？

黒、灰色、緑、白ボタンは通常、オン/オフ機能のために使用されるのよ。赤は緊急停止のためよ。

練習問題



EDU:BITに生徒の出席簿機能プログラムしてみましょう。女子がクラスに出席したときは Aボタンを、男子はBボタンを押してください。	
最初だけ	笑顔を表示させましょう。変数 'Girl'と'Boy'には0を設定します。
ボタンAが押されたとき（黄色ボタン）	変数 Girlの値を+1します。
ボタンBが押されたとき（青色ボタン）	変数 Boyの値を+1します。
ボタンAとBが同時に押されたとき	以下の情報をLEDにスクロール表示します。 Total=(Girl+Boy) ; Girl=(Girl) ; Boy=(Boy)

チャプター 3

音楽で遊ぼう！

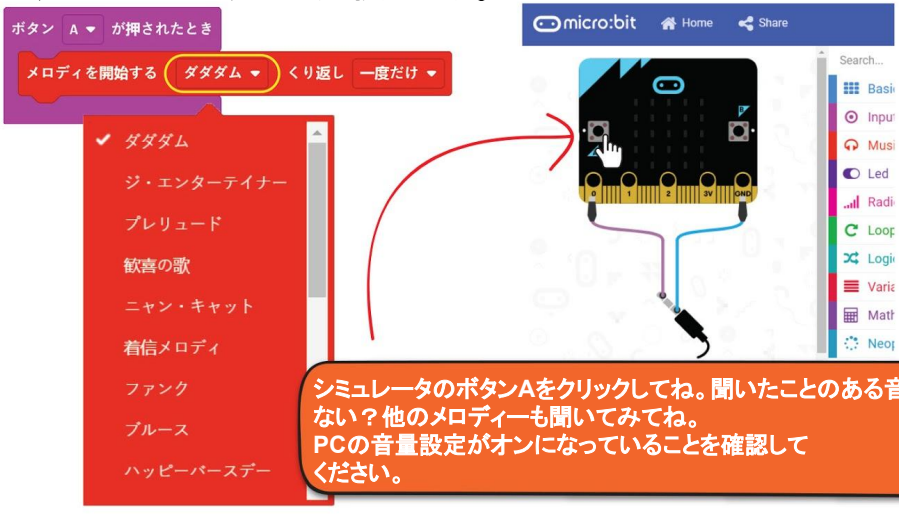
Music Bit (ピエゾブザー + オーディオジャック)

作ってみよう！

Step 1 新規プロジェクトを作成します。ツールボックスの**[入力]**から**[ボタン_が押されたとき]**ブロックを選択します。

Step 2 ツールボックスの **[音楽]**から**[メロディを開始する _くり返し_]**ブロックを選択します。

Step 3 **[ダダダム]**をクリックして、ドロップダウンリストから**'ハッピーバースデー'**を選択します。



プリセットされた音楽の他に、オリジナルの曲作ってEDU:BITにプログラムすることもできるよ。キャッチーな曲を流してみよう。

Step 4 ツールボックスの**[入力]**から**[ボタン_が押されたとき]**ブロックを選択します。ボタン"**B**"を選択します。

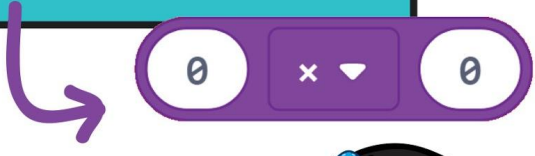
Step 5 ツールボックスの**[音楽]**から**[音を鳴らす 高さ(Hz)_長さ_]**ブロックを選択します。

Step 6 **[音を鳴らす 高さ(Hz)_長さ_]**ブロックを右クリックし、「複製する」をクリックします。**[音を鳴らす 高さ(Hz)_長さ_]**ブロックが5つになるまで繰り返して、**[ボタン_が押されたとき]**ブロックにはめ込みます。

Step 7 以下のサンプルプログラムに従って、
[音を鳴らす 高さ（Hz）_長さ_]ブロックの'**高さ（Hz）**'と'**長さ**'を選択してください。

ノート！

紫のブロックはツールボックスの[計算]から選択してください。

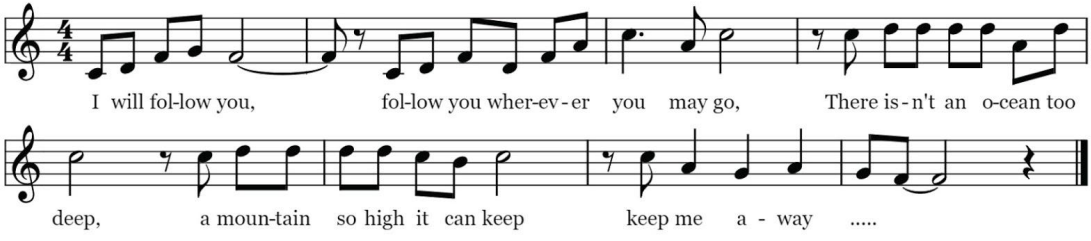


シミュレータのBボタンをクリックしてみてね。どんな曲が流れるかな？

シミュレーターを時々確認しながら、プログラムが正しいかどうかをチェックするといいよ。

Step 8 **[音を鳴らす 高さ（Hz）_長さ_]** ブロックを追加し、
'高さ（Hz）' と **'長さ'** を変更して、プログラムを作っていきます。高さ（ Hz）と長さについては、次のページを参照してください。

I Will Follow You



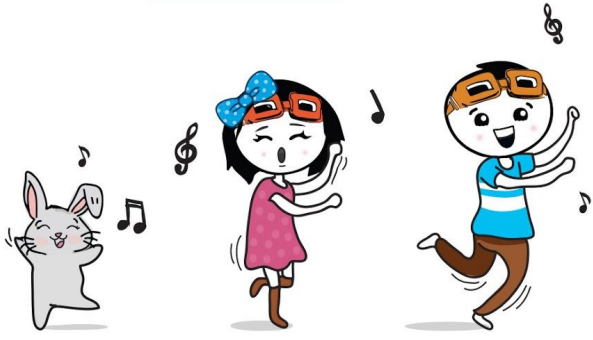
I will fol-low you,

Fol-low you wher-ev-er you may go,

There is n't an o-cean too deep

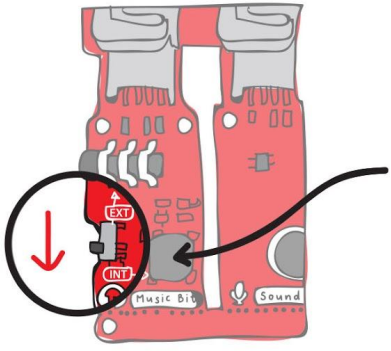
A moun-tain so high it can keep,

Keep me a-way … …



Step 9 完成したプログラムをEDU:BITにフラッシングしてください。

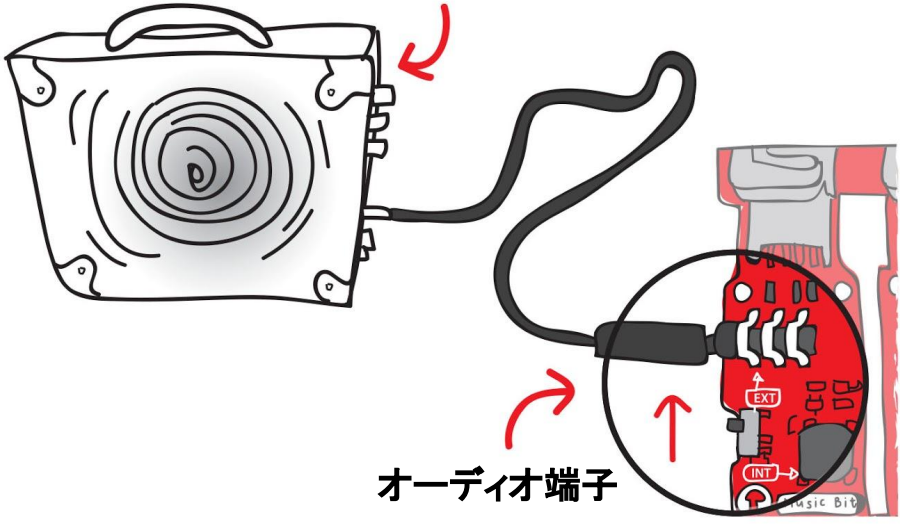
EDU:BITの青いボタン(Bボタン)を押すと、"I Will Follow You "という曲が流れるわよ。その前に、EDU:BITの電源を入れ、ピエゾブザーのスイッチをINT(内部)にしておいてね。



ピエゾブザー

また、EDU:BITのオーディオ端子に外部スピーカーやヘッドフォンを接続することもできるよ。スイッチをEXT(外部)にスライドさせてね。

スピーカー



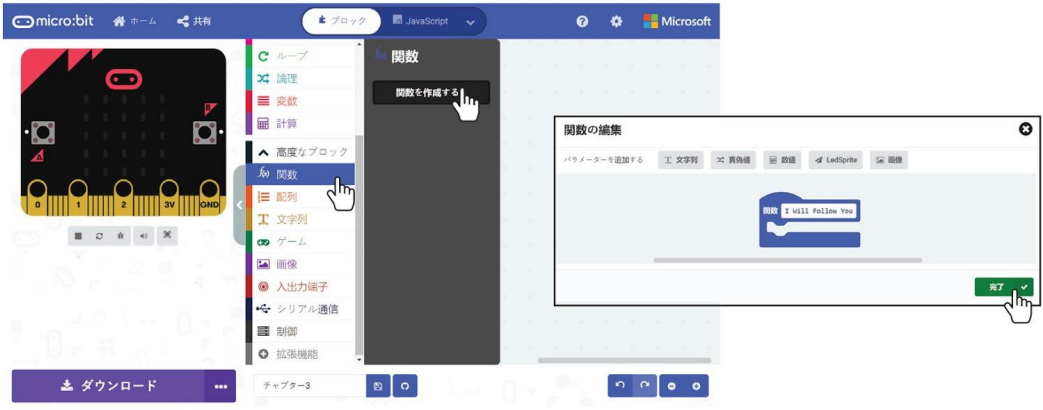
オーディオ端子

音楽が小さすぎる？それとも大きすぎるかしら？[音量を設定する_]ブロックを追加して、0から255（最大音量）まで音の大きさを変えることができるわよ。

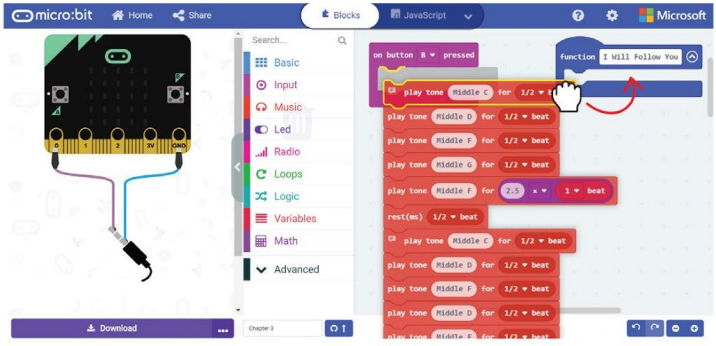
Step 10 ツールボックスから**[音楽]**をクリックし、**[音量を設定する_]**ブロックを選択します。**[最初だけ]**ブロックにはめ込み、音量を**200**に変更します。

ノート！「I Will Follow You」など、ある機能を実行するプログラムを関数として定義することが出来ます。関数は命令や手続きのセットです。一度関数を定義してしまえば、同じブロックを何度も書くことなく、複数の場所で使用することができます。

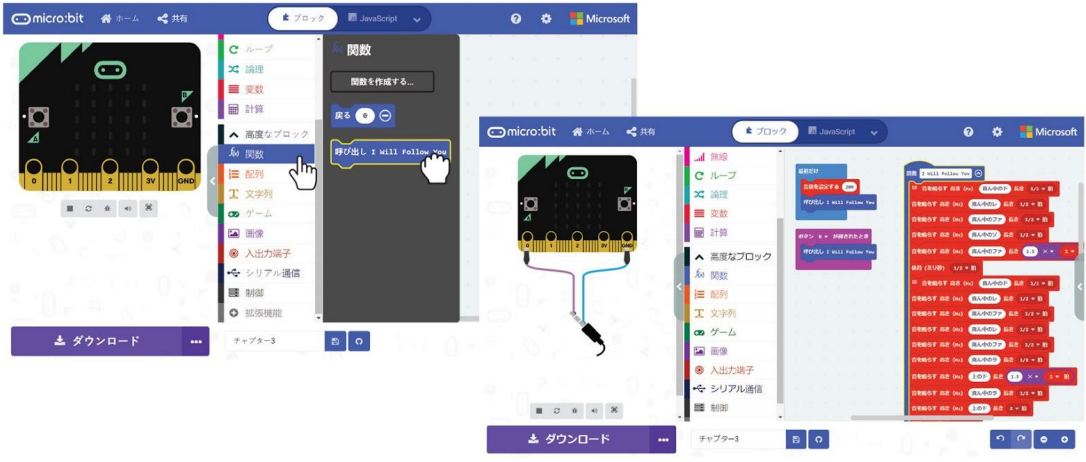
Step 11 ツールボックスから**[高度なブロック]**をクリックし、**[関数]**を選択します。**[関数を作成する...]**をクリックし、ポップアップウィンドウでdoSomethingの名前を**'I Will Follow You'**に変更します。「**完了**」をクリックします。



Step 12 エディタ上に **[関数 I Will Follow You]** ブロックが表示されます。**[ボタンBが押されたとき]**ブロックの一番上のブロックをクリックして、すべてのブロックを **[関数 I Will Follow You]**にドラッグしてはめ込みます。

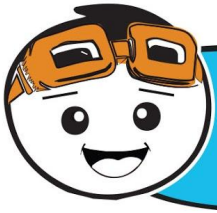


Step 13 ツールボックスから **[関数]**をクリックし、
[呼び出し I Will Follow You]ブロックを選択して、複製します。
[呼び出し I Will Follow You]ブロックを **[最初だけ]** ブロックと
[ボタンBが押されたとき] ブロックにはめ込みます。以下はサンプルプログラムです。

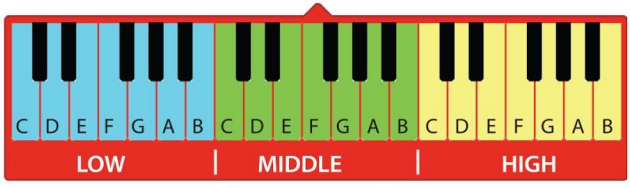


Step 14 完成したプログラムをEDU:BITにフラッシングして、音楽を聞いてみてください。

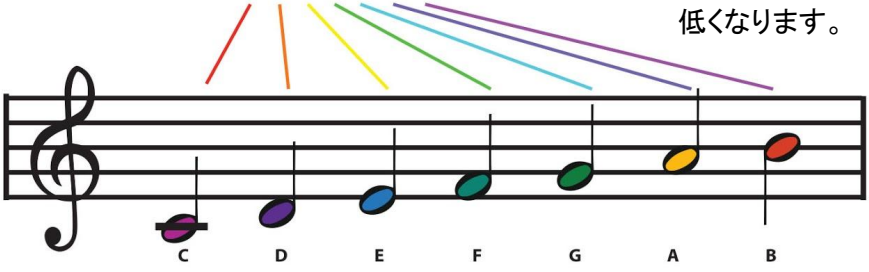
プログラムのかいせつ



楽譜の読み方を知っていれば、他の曲を作ってEDU:BITにプログラムすることもできるんだ。ここでは、譜面を「読み解く」するための簡単な方法を教えるね。



五線上の音符の位置（5本の水平線）によって、どの音を弾くかが決まります。五線上の音符の位置が高ければ高いほど音程や周波数が高くなり、逆に低ければ低いほど音程や周波数が低くなります。



Sign	Rest	Relative Length	Duration
𝅝	𝄻	Whole Note	4 beats
𝅗𝅥	𝄼	Half Note	2 beats
♩	𝄽 or 𝄾	Quarter Note	1 beat
♪	𝄿	Eighth Note	1/2 beat
𝅘𝅥𝅯	𝅀	Sixteenth Note	1/4 beat

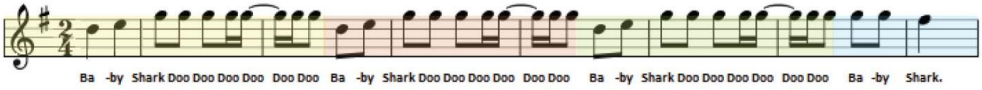
音符の種類によって、音が流される時間が変わります。

プログラムの　かいせつ

手がかりを使って、次の曲を読み取れるかしら？

Baby Shark

♩= 115



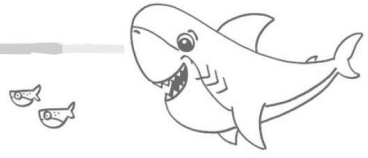
Line 1	Ba	-by	Shark	doo	doo,	doo	doo	doo	doo
Note	High D	High E	High G		High G	High G	High G		High G
Beat	1		1/2	1/2	1/2		1/2	1/4	1/2

Line 2	Ba	-by	Shark	doo	doo,	doo	doo	doo	doo
Note	High D		High G	High G	High G		High G	High G	High G
Beat	1/2	1/2	1/2		1/2	1/4	1/2		1/2

Repeat the same for Line 3

Line 4	Ba	-by	Shark
Note		High G	High F#
Beat	1/2	1/2	

Program EDU:BIT to play Baby Shark tune when the yellow button (Button A) and blue button (Button B) are pressed at the same time.



ノート！

[音量を設定する_]ブロックを使って音量を調整してください。

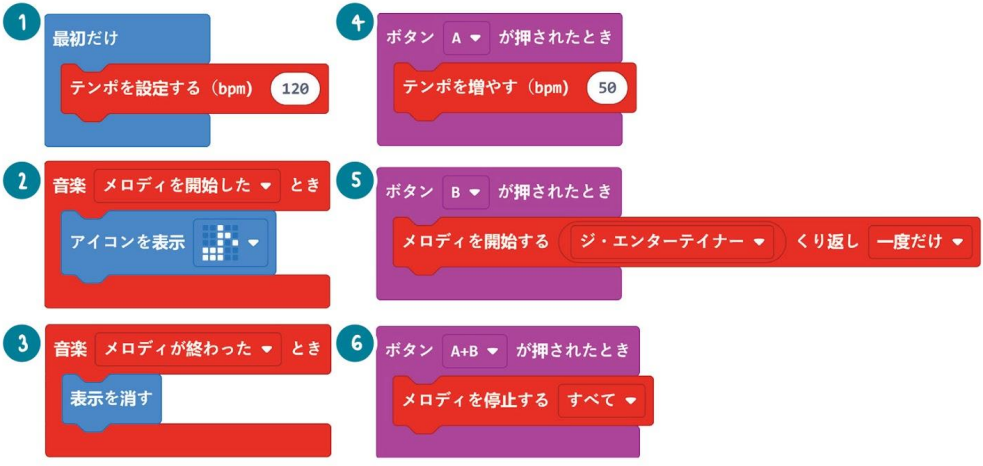
他のブロックも使ってみよう

#1 **[テンポを設定する（bpm）_]** ブロックを使って曲のペースを設定することができます。bpm（1分あたりのテンポ）が高いほど、曲は速く、リズミカルになります。テンポを変更するには、
[テンポを増やす（bpm）_]ブロックを使用します。

#2 現在再生中のメロディを停止させるには、**[メロディを停止する_]**ブロックを使用します。

#3 **[音楽_とき]**と設定条件(メロディを開始した、メロディを終了した等)をを使って、プログラムを実行するイベントにすることもできます。

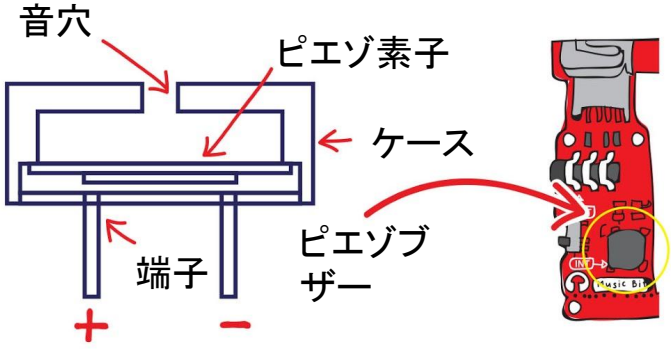
サンプルプログラム



1. このプログラムでは、テンポは 120bpmに初期設定されています。
2. 曲が始まると、LEDに音符のアイコンが表示されます。
3. 曲が終了すると、LEDがすべて消えます。
4. ボタンAを押すたびにテンポが 50bpmずつ上がります。
5. ボタンBを押すと「entertainer」が再生されます。
6. ボタンA+Bを同時押しすると曲が停止します。

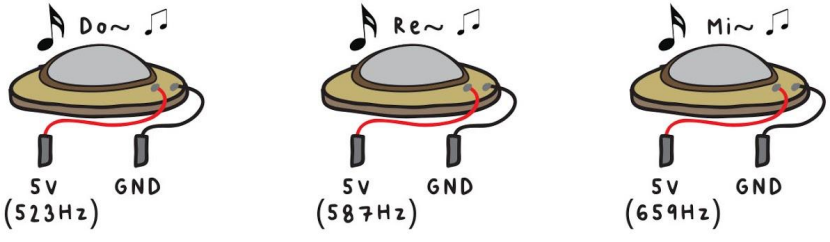
面白い事実！

ピエゾブザーは、電気信号が流れたときにピエゾ素子の一部を振動させて音を出すものです。



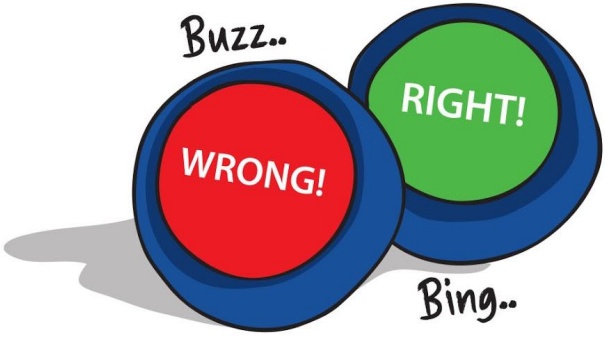
電気信号の周波数を変化させることで振動の速度が変化するため、**ピエゾブザー**は異なる音を出すことができます。

ピエゾ素子



人間の耳は、20Hz～20,000Hzの音を聞き取ることができるの。20Hz以下の音は低周波音と言って、20,000Hz以上の音は超音波っていうの。

練習問題



正解/不正解を知らせるゲームショーブザー機能を EDU:BITにプログラムしましょう。	
最初だけ	笑顔を表示します。
ボタンAが押されたとき （黄色ボタン）	✔アイコンを表示し、”パワーアップ”を一度だけ流します。
ボタンBが押されたとき （青色ボタン）	✘アイコンを表示し、”ワワワワー”を一度だけ流します。
ボタンAとBが同時に押されたとき	LEDを消します。

チャプター 4

勝つか、負けるか、引き分けか

Traffic Light Bit

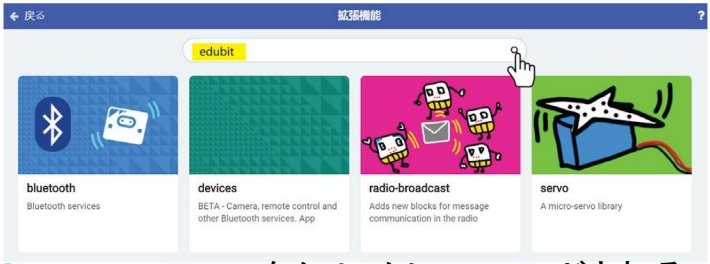
スキャンしてね

EDU:BITに赤、黄、緑のLEDが見えるかしら？これは信号機ビットというのよ。これを使ってプログラムするには、 MakeCodeエディターにEDU:BIT拡張機能を追加する必要があるの。拡張機能とは、EDU:BITをプログラムするためにエディタに追加するカスタムブロックのようなものね。

作ってみよう！

Step 1 MakeCodeエディターで新しいプロジェクトを作成します。　アイコンをクリックし、「拡張機能」を選択します。 *拡張機能の追加にはインターネット接続が必要です。

Step 2 検索ボックスに”**edubit**”と入力し、エンターキーを押します。

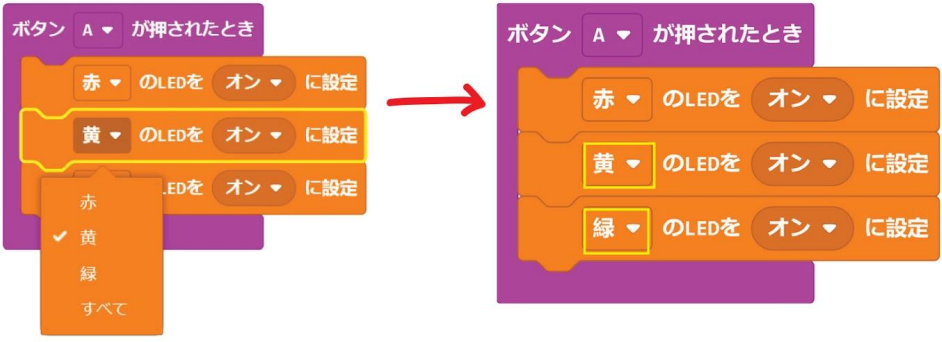


Step 3 **'edubit'**をクリックして、ロードされるのを待ちます。ロードが終わると、MakeCodeエディターのツールボックスに新しい機能が追加されます。

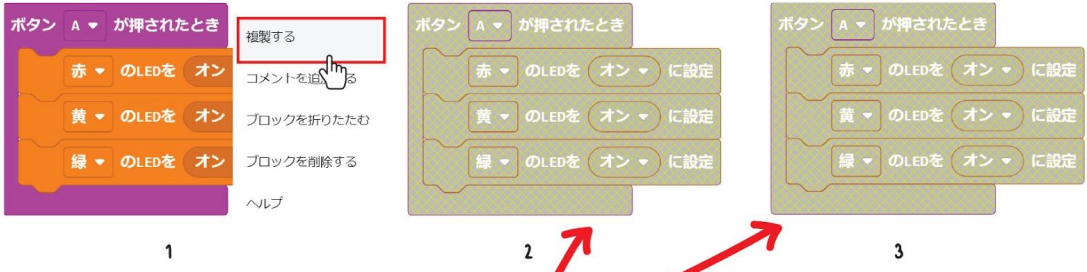
Step 4 ツールボックスから **[入力]**をクリックし、**[ボタン_ が押されたとき]**ブロックを選択します。

Step 5 ツールボックスから **[信号機ビット]**をクリックし、**[_のLEDを_に設定]**ブロックを選択します。**[_のLEDを_に設定]**ブロックを右クリックし、「複製する」をクリックして、3つの**[_のLEDを_に設定]**ブロックを作ります。それらのブロックを **[ボタンAが押されたとき]**にはめ込みます。

Step 6 色を上から順に**赤**、**黄色**、**緑**の順に設定します。



Step 7 **[ボタン_が押されたとき]**ブロックを右クリックして「複製する」を選択します。これを繰り返して、同じブロックを 3セット作ります。



*これらのブロックは無効化されています。これは**[ボタン_が押されたとき]**ブロックが複数存在するためです。

Step 8 2 番目と 3 番目の**[ボタン_が押されたとき]**ブロックの「A」をそれぞれ**「B」**と**「A+B」**に変更します。

Step 9 以下の図にしたがって、LEDをオンからオフに変更します。

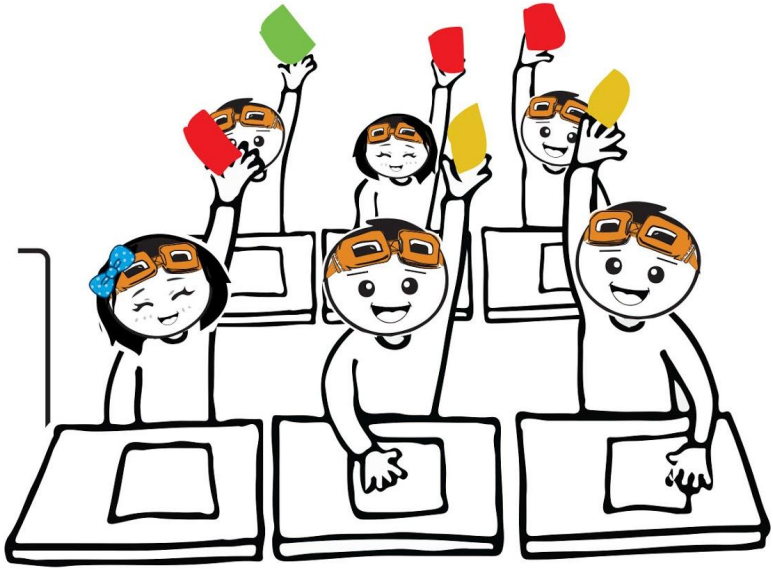
Step 10 EDU:BITにプログラムをフラッシングして、Aボタンと押したとき、Bボタンを押したとき、A+Bボタンを同時押ししたときでどうなるか、観察してみましょう。

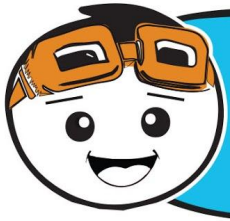
出してくれ！

立入禁止

入っていいよ

すごーい！これで、EDU:BITを自分の信号機として使えるようになったね。ほかにも使い道が思いつくかな？





LED（発光ダイオード）はデジタル出力装置の一つの例なんだ。LEDには、オンとオフの2つの状態しかなくて、オンは一般的に1、オフは0を意味するんだ。

EDU:BITにタイマーインジケータ機能をプログラムすることもできるのよ。以下のサンプルコードを参考にしてね。

EDU:BITを振るとタイマーが始まります。

タイマーがスタートしたことを知らせる音。

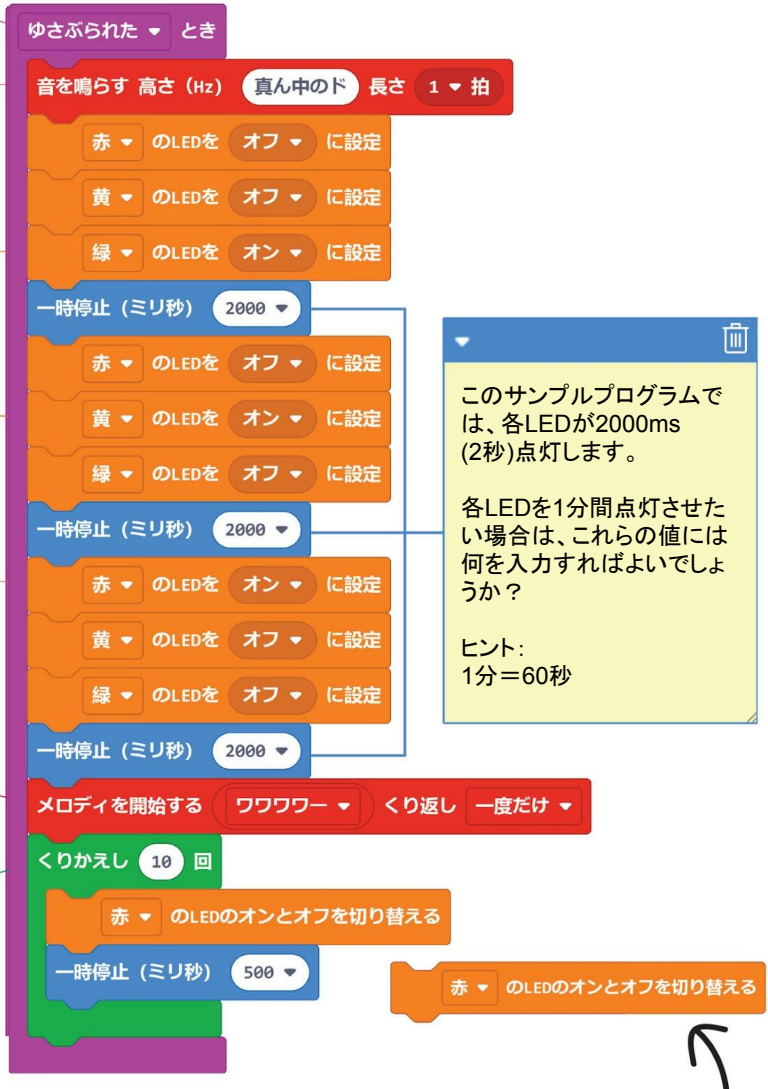
緑色のLEDが点灯する。

黄色のLEDが点灯する。

赤色のLEDが点灯する。

時間切れになると、「ワワワワー」と鳴ります。

赤色LEDを10回点滅させます。



このサンプルプログラムでは、各LEDが2000ms(2秒)点灯します。

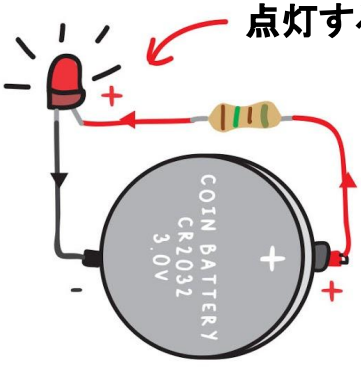
各LEDを1分間点灯させたい場合は、これらの値には何を入力すればよいでしょうか？

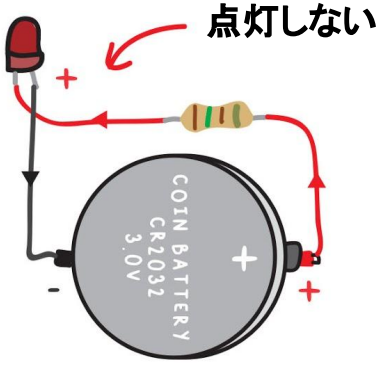
ヒント：
1分＝60秒

LEDの状態をオンからオフ、オフからオン。これを繰り返すことによって、LEDが点滅するように見せることができるんだ。

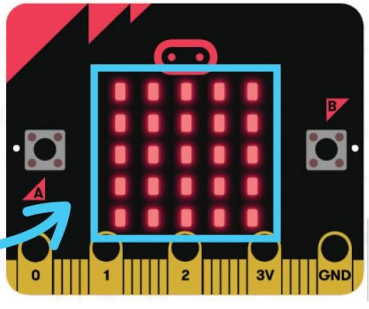
面白い事実！

発光ダイオード（LED）とは、電気から光を作り出す半導体デバイスのことです。プラス端子とマイナス端子の2つの端子を持っています。LEDのプラスとマイナスを正しく接続し、電流を流すと発光します。





micro:bitで使用されるLEDは表面実装技術(SMT)に基づいており、非常に小さくすることができます。



micro:bit以外にも、EDU:BIT基板上には41個のSMT LEDがあるのよ。全部みつけられるかしら？

練習問題

EDU:BITに「チャレード」などのゲーム用スコアカウンターやタイマー機能をプログラムしてみましょう。 ※Charade（シャレード）～身振り手振りで言葉あて(英語)をするジェスチャーゲーム	
最初だけ	変数 Team A = 0 変数 Team B = 0
ボタンAが押されたとき （黄色ボタン）	チームAのスコアを＋1します。 チームAの現在のスコアを表示します。
ボタンBが押されたとき （青色ボタン）	チームBのスコアを＋1します。 チームBの現在のスコアを表示します。
ボタンAとBが同時に押されたとき	チームAとBのスコアをスクロール表示します。
ゆさぶられたとき	1分間タイマーをスタートさせます。**緑色のLED**（30秒）、次に**黄色のLED**（20秒）、最後に**赤色のLED**（10秒）を点灯。タイムアップしたら「ワワワワー」を鳴らしながら、**赤色LED**を10回点滅させます。

ヒント:
2つの変数を作り、それぞれにチーム Aとチームという名前をつけよう。

遊んでみよう！

Win, Lose or Draw~

遊び方：

- チームAとチームBの2チームに分かれましょう。
- チームAのメンバー1人がカードを1枚選んでゲームスタートです。カードに書かれた文字を読んだ後、EDU:BITを振ってタイマー（1分）をスタートさせます。
- カードの内容をボードに絵いて、チームメンバーに当ててもらいます。喋ったり、身振り手振りをしたりしてはいけません。
- 時間切れまでにチームメンバーの誰かが正解した場合、 Aボタンまたは黄色のボタンを押します。チームAに1ポイントが与えられます。
- チームAが不正解で、逆にチーム Bが正解したらチーム Bに1ポイントGETです。
- 両チームが交代で、時間切れまでゲームを続けていきます。
- 最も多くのポイントを獲得したチームが勝利です。

スキャン

ノート！

ここをスキャンして、印刷可能なカードをダウンロードしてください。絵を描けない場合は、シャレードに挑戦してみてください。絵を描く代わりに、ジェスチャーを使ってメンバーにヒントを出します。自由に楽しんでください。

チャプター 5

IR デジタルサイコロを転がしてみよう

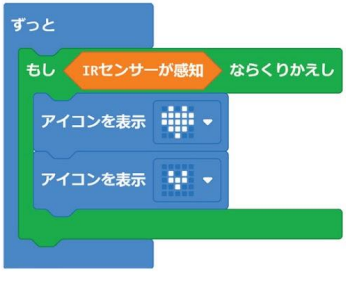
IR Bit

作ってみよう！

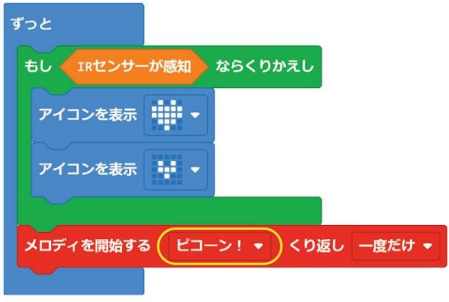
Step 1 MakeCodeエディターで新規プロジェクトを作成し、EDU:BIT拡張機能を追加します(ページ40参照)。ツールボックスから**[ループ]**をクリック。**[もし_ならくりかえし]**ブロックを選択し、**[ずっと]**ブロックにはめ込みます。

Step 2 ツールボックスから**[IR Bit]**をクリックし、**[IRセンサーが感知]**ブロックを選択。**[もし_ならくりかえし]**ブロックにはめ込みます。

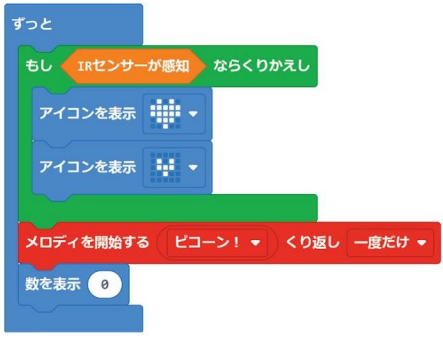
Step 3 ツールボックスの**[基本]**をクリックし、**[アイコンを表示_]**ブロックを2つ追加してください。片方のアイコンを「小さいハート」に変更し、**[もし_ならくりかえし]**ブロックにはめ込みます。



Step 4 ツールボックスから[音楽]をクリックし、
[メロディを開始する_くり返し_]ブロックを選択します。メロディを
”ダダダム”から”**ピコーン！**”に変更します。

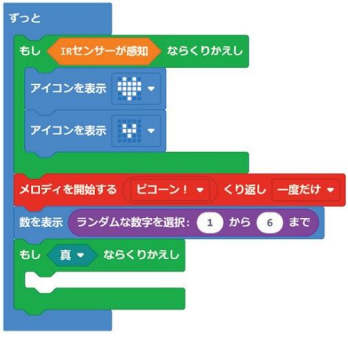


Step 5 ツールボックスから[基本]をクリックし、[数を表示_]ブロックを選択します。

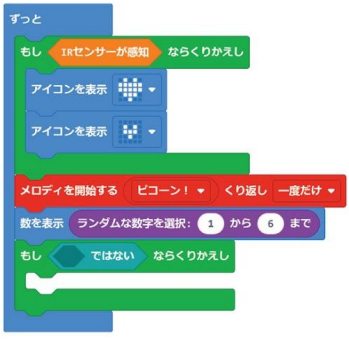


Step 6 ツールボックスから[計算]をクリックし、[ランダムな数字を選択_から_まで]ブロックを選択。数字を**1**と**6**にします。

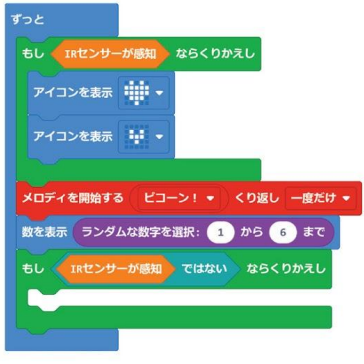
Step 7 ツールボックスから**[ループ]**をクリックし、**[もし_ならくりかえし]**ブロックを選択します。



Step 8 ツールボックスから**[論理]**をクリックし、真偽値**[_ではない]**を選択し、**[もし_ならくりかえし]**の条件部にはめ込みます。



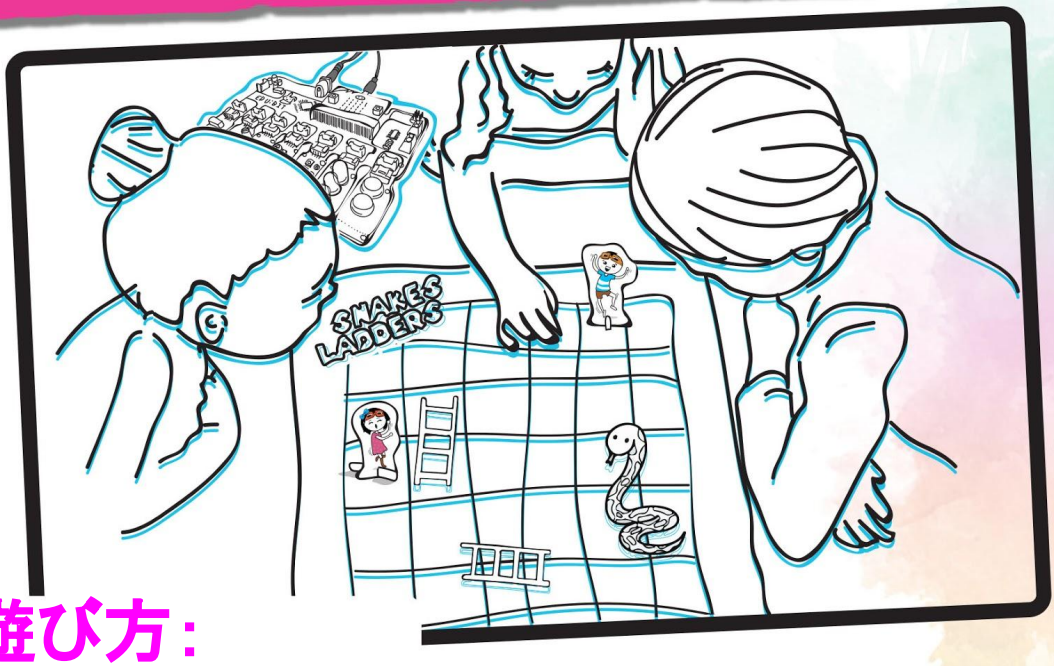
Step 9 ツールボックスから**[IR Bit]**をクリックし、**[IRセンサーが感知]**ブロックを選択。**[_ではない]**ブロックにはめ込みます。



Step 10 EDU:BITにプログラムをフラッシングします。

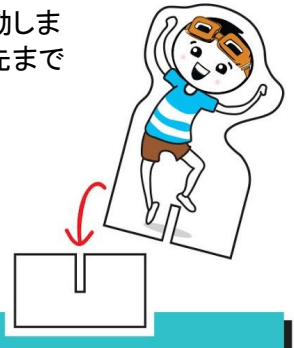
遊んでみよう！

ヘビとはしごゲーム



遊び方：

- 各プレイヤーは 1枚のキャラクターを選び、「 Start Here」と書かれたスペースに置きます。
- プレイヤーは、交互に「サイコロを振り」ます。 IR ビット上に手のひらをかざします。ハートのアニメーションが表示されたら、手のひらを離します。
- LEDに表示されている数（ 1～6 の間）分、キャラクターを前方に移動させます。
- キャラクターがはしごの下のマスに来た場合は、はしごの上に移動します。キャラクターがヘビの頭のマスに来た場合は、ヘビの尻尾の先まで移動します。
- 最初に100に到達したプレイヤーが勝ちです。楽しんでください。



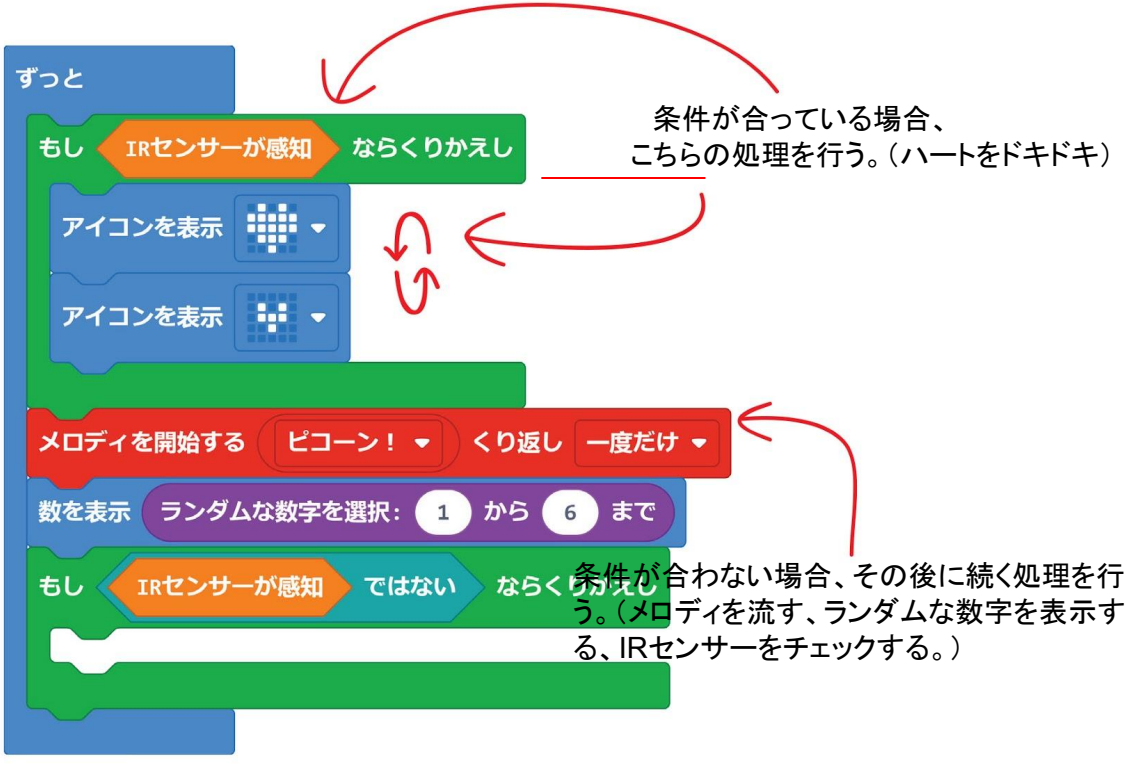
ノート！

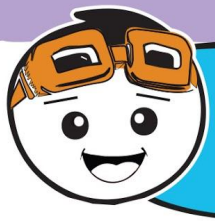
ヘビとはしごのゲームボードとキャラクターがキットに入っています。キャラクターとスタンドを取り出して、組み立ててください。

プログラムの かいせつ

さっきのプログラムでは、[ループ]の[もし_ならくりかえし]ブロックを使ったね。ループの仕組みを知っているかな？

プログラムが **[もし_ならくり返し]** ブロックに来ると、条件をチェックします。条件が合っている間（真の場合）、プログラムは **[もし_ならくり返し]** ブロック内のプログラムを実行します。条件が合わなくなると（偽の場合）と、プログラムはループを終了し、次のプログラムを実行します。

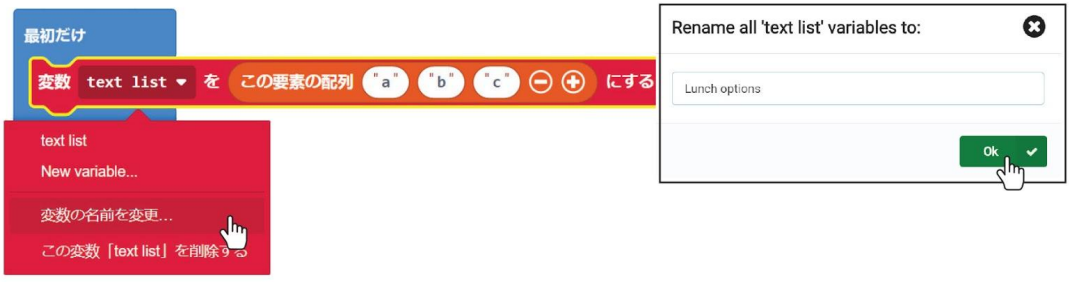




EDU:BITをデジタルサイコロとして使う以外にも、どれを選ぶか決められないときに、選択肢から選ばせるプログラムに修正することもできるよ。例えば、今日のランチは何をにしようかな？

Step 11 **[高度なブロック]**:**[配列]**から**[変数_をこの要素の配列_ _ _にする]**ブロックを選択し、**[基本]**:**[最初だけ]**にはめ込みます。

Step 12 **[text list]** blockを選択し、**"変数の名前を変更"**選択します。ポップアップウィンドウに**'Lunch options'**と入力し、「OK」をクリックします。

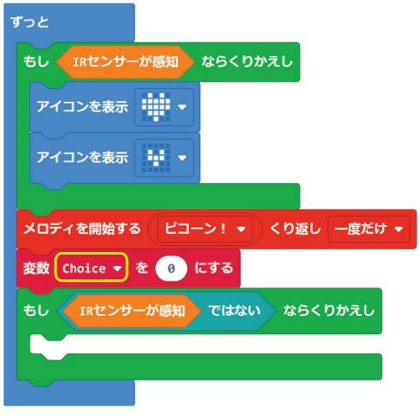


Step 13 配列を1つずつクリックしてランチメニューを入力します。

Step 14 ツールボックスから**[変数]**をクリックして、
という変数を作ってください。

Step 15 **[数を表示 ランダムな数字を選択_から_まで]**ブロックを右クリックし、
「ブロックを削除する」を選択してください。

Step 16 ツールボックスから**[変数]**をクリックし、**[変数_を_にする]**
ブロックを選択します。**[メロディを開始する_くり返し]**と**[もし_ならくりかえし]**ブロックの間にはめ込みます。
変数名を**'Choice'**にします。



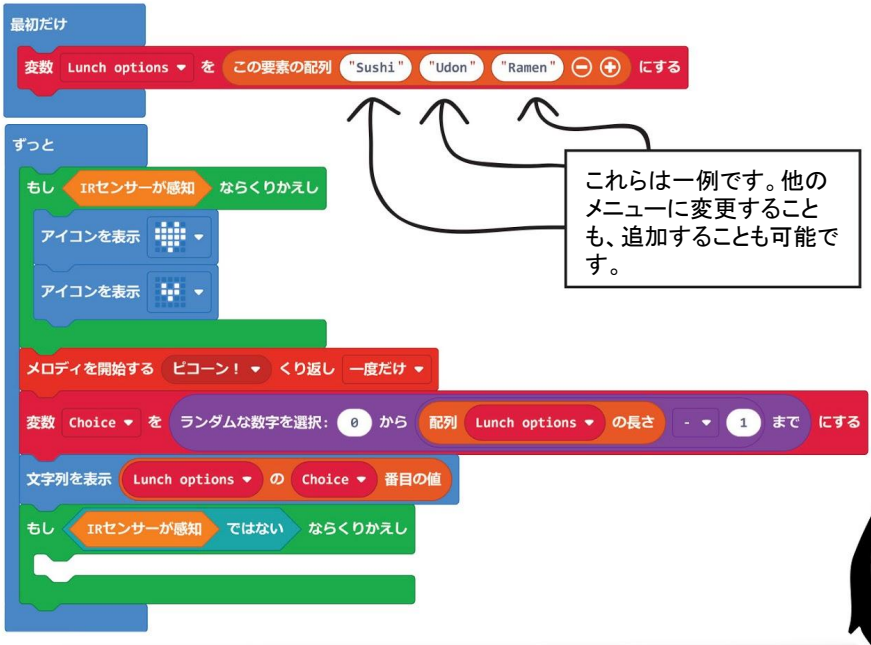
Step 17 ツールボックスから**[計算]**をクリックし、
[ランダムな数字を選択_から_まで]と**[_-_]**ブロックを追加します。最後の数字は**1**にします。

Step 18 ツールボックスから**[基本]**をクリックし、**[文字列を表示]**ブロックを選択します。

Step 19 ツールボックスから**[高度なブロック]**：**[配列]**をクリックし、
[配列_長さ]と**[_の_番目の値]**ブロックを選択します。

Step 20 **[list]**をクリックし、両方のブロックの変数を**Lunch options'**に変更します。ツールボックスから**[変数]**をクリックし、**[Choice]**ブロックを選択します。これを**[_の_番目の値]**ブロックにはめ込みます。

「今日のランチは何にしようかな？」プログラムの完成です。

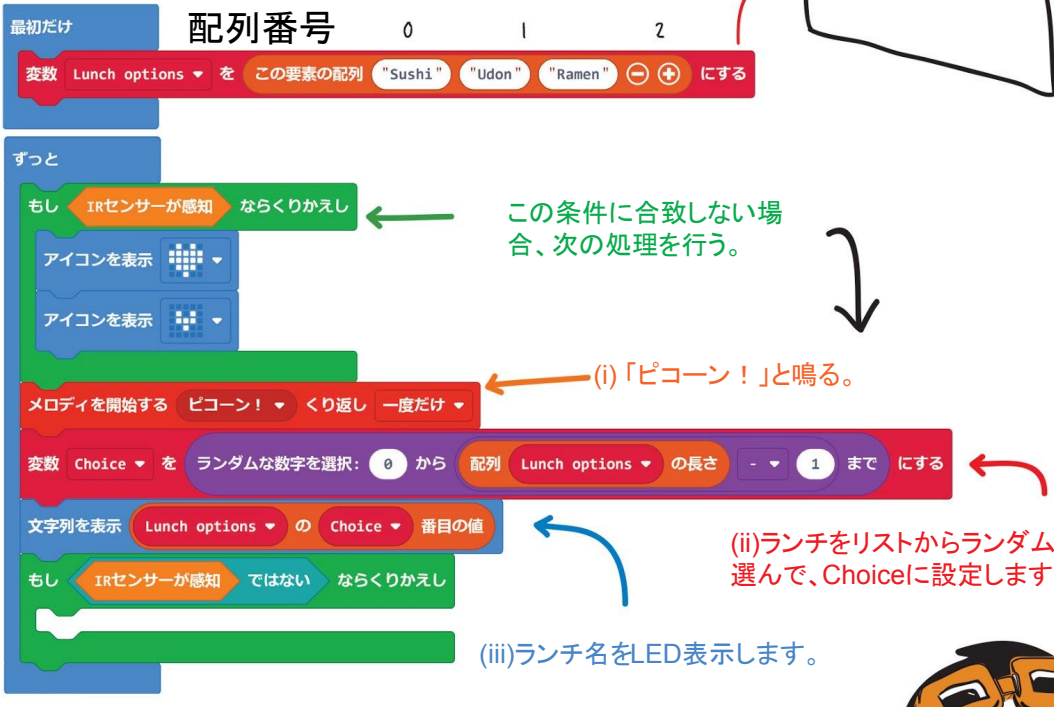


次に何を食べようか迷ったときは、IR Bitに手をかざして、手を離すことで、EDU:BITに決めてもらうことができるわよ。また、プログラムを修正することで、友達と遊ぶゲームを作ることもできるわよ。どこを変更すればいいかわかるかしら？

プログラムのかいせつ

配列は、関連する変数のリストまたは集まりです。複数に分割されたフォルダのようなもので、それぞれに情報を保存します。配列は、リストを追加したり削除したりする必要があるときに、プログラムを簡単に変更できるようにする為に使用します。

このプログラム例では、3つの要素を持つ配列を作成し、これを「Lunch options」と名付けています。そして、各メニューを簡単に編集することができます。また、　　ボタンをクリックするだけで、メニューを追加したり、減らしたりすることもできます。



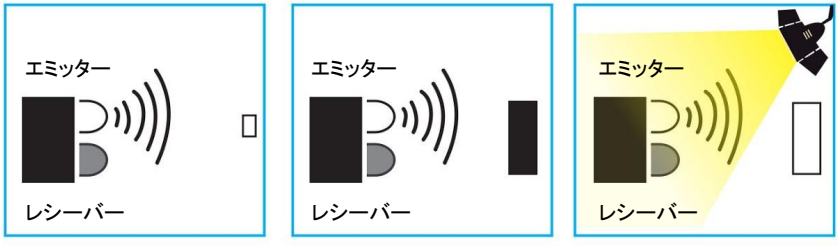
プログラミングでは、1から数えるのではなく、0から数えるようにしているんだ。だから、「炒飯」の配列番号は 0で、最後の「ナシレマ」は3番目だけど、配列番号は 2になるんだ。

面白い事実！

赤外線センサー(IR)は、一般的に人やモノの動きを検出するために使用される電子機器です。IRセンサーは、エミッタ(IR LED)とレシーバ(光検出器)の2つの部分から構成されています。

どうやって動いているの？

IR LED は、センサの前にある物体から反射する赤外線を放射します。反射光の量がしきい値より大きい場合、IR Bitは「感知」されます。感知されると、IR Bit の LED表示器が点灯します。
物体がない場合や、遠すぎる場合は、赤外線はレシーバに反射しません。そのため、IR ビットは感知しません。
また、以下のような場合には、IRセンサーが動作しないことがあります。



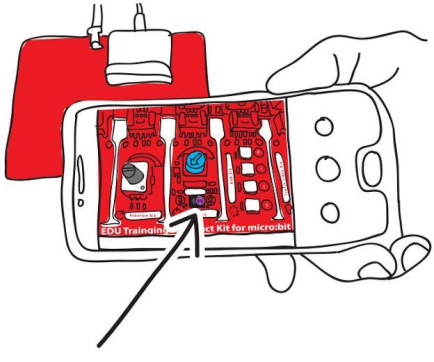
物体が小さすぎる。

物体の表面が黒または濃い色をしている。

光が邪魔している。

知ってた？

赤外線は肉眼では見えませんが、スマホのカメラだと赤外線を見ることができます。

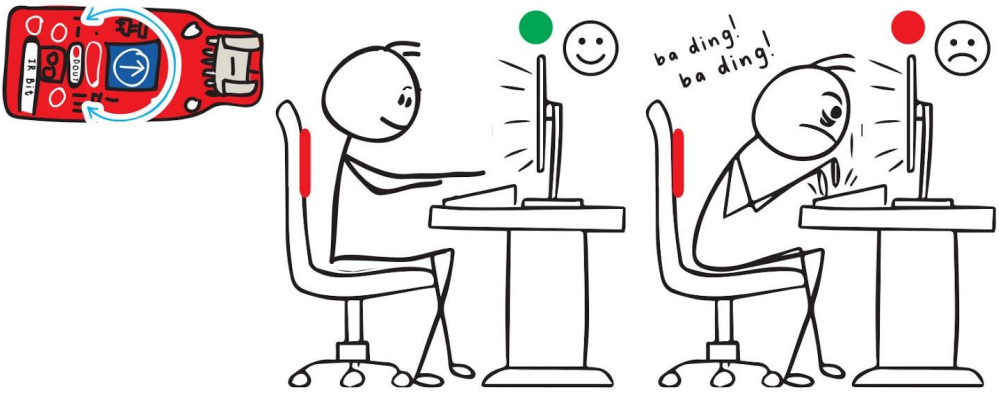


練習問題

EDU:BITに「姿勢を正す機能」をプログラムしてみましょう。	
最初だけ	“Mind your posture:”という文字列をスクロール表示します。
IRセンサーが感知したとき	LEDに笑顔を表示し、緑色の LEDを点灯させます。
IRセンサーが感知しなかったとき	「ピコーン！」と音を一回鳴らします。LEDに悲しい顔を表示し、赤色の LEDを点灯させます。

使い方：

EDU:BITを図のように椅子の背もたれに取り付けます。正しい姿勢で座ってください。LEDが点灯するまで、IR Bitの青いノブを調整します（IR Bitが背中を検知します）。この作業をキャリブレーションといいます。

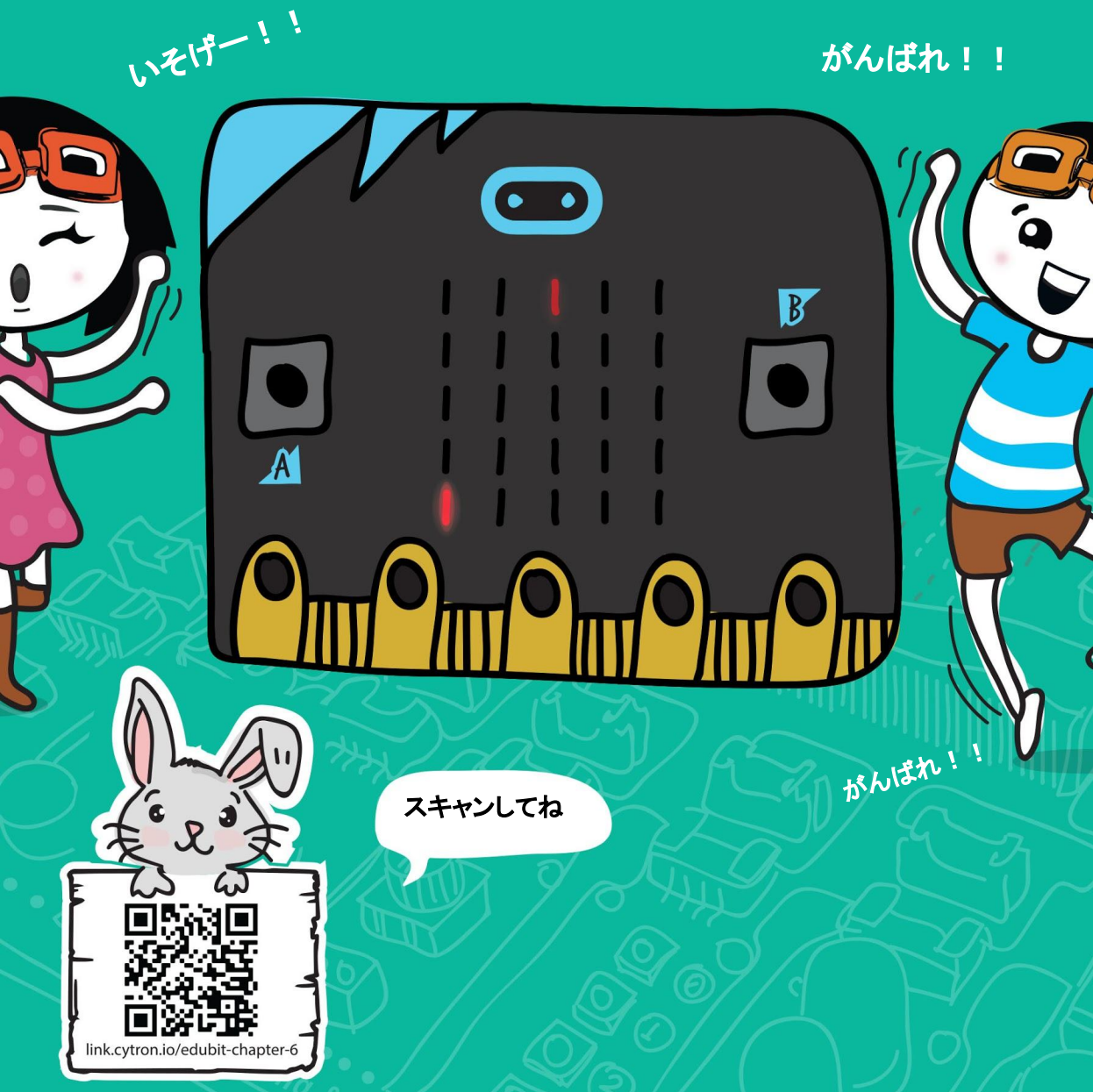


服の色によっては、IRビットを調整する必要があるわ。なぜか分かるかしら？

チャプター 6

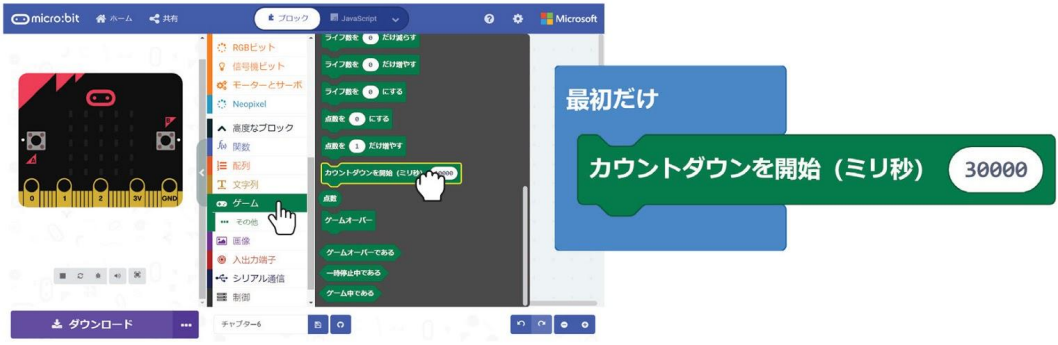
鬼ごっこ

Potentio Bit

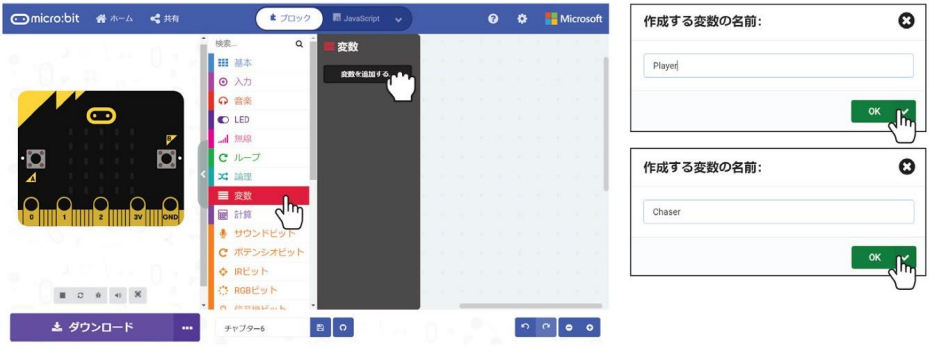


作ってみよう！

Step 1 MakeCodeエディターで新規プロジェクトを作成し、EDU:BIT拡張機能を追加します (ページ40参照)。ツールボックスの**[高度なブロック]**から**[ゲーム]**をクリックします。**[カウントダウンを開始（ms）_]**ブロックを選択し、**[最初だけ]**ブロックにはめ込みます。値は **30000**にします。



Step 2 ツールボックスの**[変数]**から**[変数を追加する..]**をクリックします。ポップアップウィンドウに**'Player'**と入力し、「OK」をクリックします。同じ手順で、**'Chaser'**という変数を追加します。



Step 3 ツールボックスから**[変数]**をクリックし、**[変数_を_にする]**ブロックを選択します。**[変数_を_にする]**ブロックを複製し、**[最初だけ]**ブロックにはめ込み、変数をそれぞれ、**'Chaser'**と**'Player'**にします。

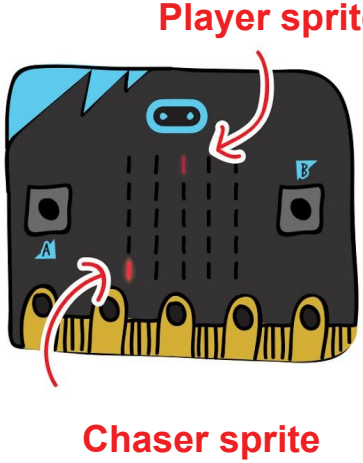
Step 4 ツールボックスの**[高度なブロック]**から**[ゲーム]**をクリックし、**[スプライトを作成 X:_ Y:_]**ブロックを選択します。ブロックを複製し、**[変数_を_にする]**ブロックにはめ込みます。**'Chaser'**の値を**x: 0 y:5**、**'Player'**の値を**x: 2 y:0**に変更します。

Step 5 ツールボックスの**[高度なブロック]**から**[ゲーム]**をクリックし、**[_方向転換_に_°]**ブロックを選択します。**[最初だけ]**ブロックにはめ込み、値を**'Player'**、角度を**90**に設定します。

Step 6 ツールボックスの**[高度なブロック]**:**[ゲーム]**をクリックし、**[_の_に_を設定する]**ブロックを選択します。変数名を**'Player'**、**'x'**を**'明るさ'**、明るさの値を**50**に設定します。

Step 7 ツールボックスの**[入力]**をクリックし、
[ボタン_が押されたとき]ブロックを選択します。ブロックを複製し、
2つ目の**[ボタン_が押されたとき]**ブロックの値を'B'に変更します。

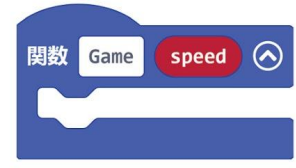
Step 8 ツールボックスの**[ゲーム]**をクリックし、**[_を_ドット進める]**
ブロックを選択します。ブロックを複製し、**[ボタン_が押されたとき]**にはめ込みます。両方のブロックの変数名を**'Player'**に変更し、値を**-1**（Aボタンが押されたとき）、**1**（Bボタンが押されたとき）に設定します。



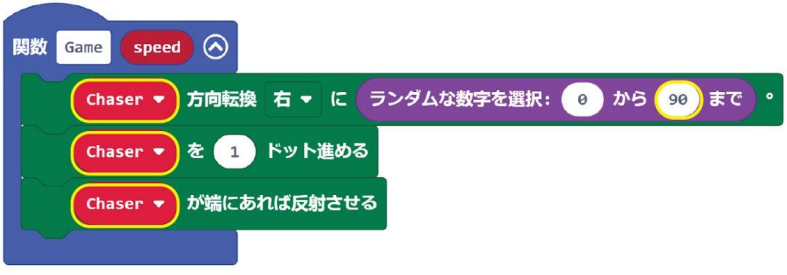
EDU:BITにプログラムをフラッシングしよう。青いボタン（Bボタン）を押すと、うす暗いLEDが下に向かって動くのが分かるかな？それがPlayerのスプライトだよ。スプライトとは、自分でコントロールできる「生き物みたいな小さなLED」だよ。
じゃ、黄色のボタン（Aボタン）を押すとどうなると思う？

Step 9 ツールボックスの**[高度なブロック]**をクリックし、**[関数]**から**[関数を作成する...]**を選択します。関数の編集画面でdoSomething'を**'Game'**に変更します。次に**[数値]**をクリックしてパラメータを追加し、'数値'を**'speed'**に変更して、**「完了」**してください。

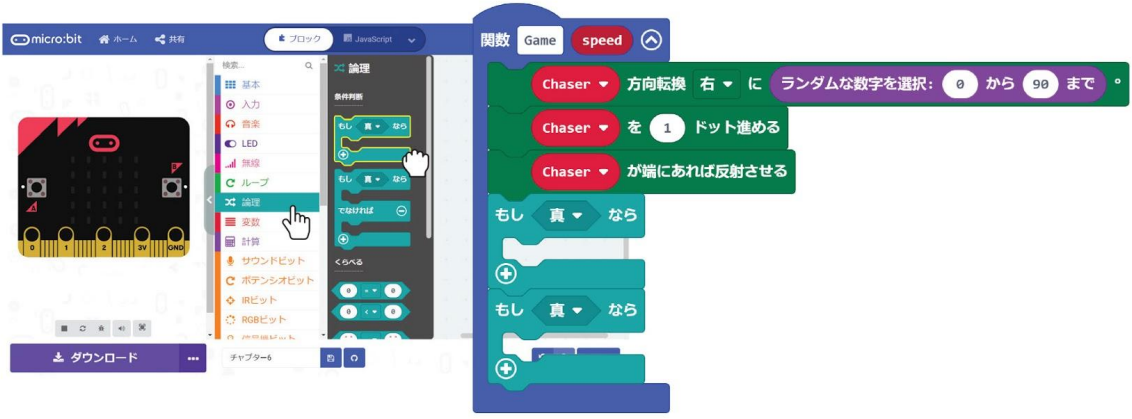
この関数ブロックが表示されます。



Step 10 続けて、**[高度なブロック]**:**[ゲーム]**と**[計算]**からブロックを選択し、以下の図のようにプログラムしてください。変数名は**'Chaser'**に、値は**90**に設定してください。



Step 11 [論理]から[もし-なら]ブロックを2つ、プログラムに追加します。



Step 12 [高度なブロック]：[ゲーム]と[計算]からブロックを選択し、以下の図のようにプログラムしてください。

がんばれ！

Step 13 両方の**[sprite]**ブロックをクリックして、**'Chaser'**に変更してください。ツールボックスから**[変数]**をクリックし、**[Player]**ブロックを選択します。
[_が他のスプライト_にさわっている]ブロックの空欄にはめ込みます。

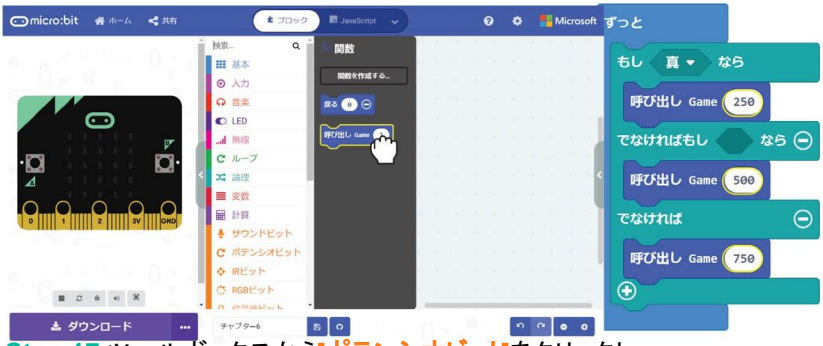
Step 14 ツールボックスの**[基本]**をクリックして、**[一時停止（ミリ秒）_]**ブロックを選択します。プログラムに追加してください。関数ブロックの**[speed]**をドラッグ＆ドロップで**[一時停止（ミリ秒）_]**の空欄にはめ込みます。

鬼がプレイヤーにさわったらに音を鳴らしてゲームを盛り上げましょ。どのブロックをどこに追加すればいいか、分かるかしら？

ゲームの難易度をたかくしてみよう！

Step 15 ツールボックスの **[論理]**をクリックし、**[もし-なら-でなければ]**ブロックを選択します。ブロックを**[ずっと]**ブロックにはめ込みます。条件を増やすには（＋）ボタンを押してください。

Step 16 ツールボックスから**[関数]**をクリックし、**[呼び出し ゲーム_]**ブロックを選択します。ブロックを複製し、**[もし-なら-でなければ]**ブロックにはめ込みます。**[呼び出し ゲーム_]**ブロックの値を**250**、**500**、**750**の順に変更します。



Step 17 ツールボックスから**[ポテンショビット]**をクリックし、**[ポテンショメータ値>_]**ブロックを選択します。ブロックを複製し、**[もし-なら-でなければ]**ブロックにはめ込みます。最初のブロックの値を**800**、2番目のブロックの値を**400**に変更します。

これが完成したプログラムです。

アニメーションを表示し、30秒のタイマーを開始します。

ChaserとPlayerを指定した座標上に表示します。

Playerのスプライトを90度回転させる。

Playerの明るさを薄暗くしてChaserと見分けがつくようにしています。

```
最初だけ
カウントダウンを開始（ミリ秒） 30000
変数 Chaser を スプライトを作成 X: 0 Y: 5 にする
変数 Player を スプライトを作成 X: 2 Y: 0 にする
Player 方向転換 右 に 90 °
Player の 明るさ に 50 を設定する
```

AボタンとBボタンを押したときにPlayerが一つ上または一つ下に移動するよう設定します。

```
ボタン A が押されたとき
Player を -1 ドット進める

ボタン B が押されたとき
Player を 1 ドット進める
```

Chaserのスプライトを制御する関数。

チェイサーの角度を0～90°の間で右方向にランダム設定し、1ステップ移動します。チェイサーが端に触れると逆方向に移動開始します。

チェイサーのスプライトがエッジに触れるたびに変数 "score "に1点加点します。

Chaser がPlayer触ったとき、「ワワワワー」と音を鳴らし、ゲームオーバーのアニメーションとスコアを表示します。

変数 "speed "に設定したミリ秒の間、プログラムを一時停止させます。

```
関数 Game speed
Chaser 方向転換 右 に ランダムな数字を選択: 0 から 90 まで °
Chaser を 1 ドット進める
Chaser が端にあれば反射させる
もし Chaser が端にある なら
点数を 点数 + 1 にする
もし Chaser が他のスプライト Player にさわっている なら
ゲームオーバー
一時停止（ミリ秒） speed
```

ずっとポテンシオビットを監視します。

もし、ポテンショメータ値が800より高い場合は、関数"ゲーム "を呼び出し、変数 "speed "を250msに設定します。

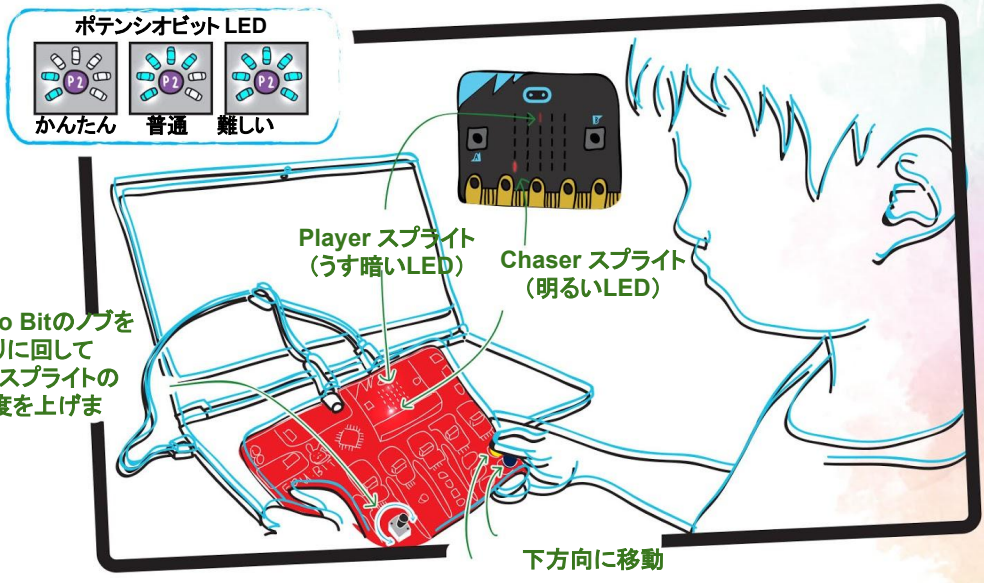
でなければもし、ポテンショメータ値が400よりも高い場合は、関数"ゲーム "を呼び出し、変数 "speed "を500msに設定します。

でなければ 関数"ゲーム "を呼び出し、変数 "speed "を750msに設定します。

```
ずっと
もし ポテンショメータ値 > 800 なら
呼び出し Game 250
でなければもし ポテンショメータ値 > 400 なら
呼び出し Game 500
でなければ
呼び出し Game 750
```

遊んでみよう！

鬼ごっこ



遊び方：

- 電源を入れると、Chaserのスプライトがランダム方向に動き出します。
- Playerのスプライトを上下に動かし、Chaserに当たらないようにしましょう。黄色ボタン（Aボタン）で上、青色ボタン（Bボタン）で下方向に移動します。
- PlayerとChaserがぶつかるか、30秒経過するとゲームオーバーです。
- Chaserが端にぶつかる毎に、1点ゲットです。一番点数の高いPlayerが勝ちです。楽しみましょう！

ヒント！

***1 Chaserの速さを上げると、Chaserが端にぶつかる回数も多くなるので、30秒以内に高い点数を取りやすくなります。**
***2 ゲームオーバー後、Aボタン+Bボタンの同時押しで、再スタートすることができます。これは[Game]ブロックに組み込まれた機能です。**

プログラムのかいせつ

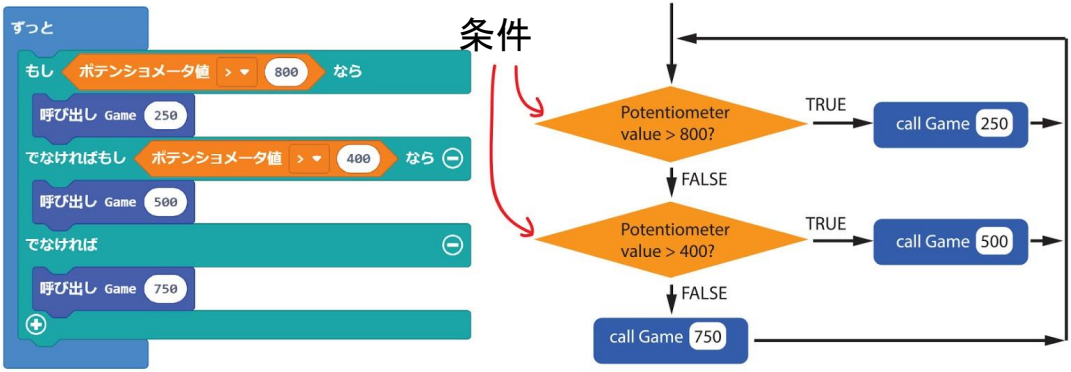
プログラミングにおいては、条件文を使って処理の流れを決めて行きます。MakeCodeにおいては、**[論理]**から**[もし-なら]**か**[もし-なら-でなければ]**ブロックを使って条件文を作ります。もし、式の条件が合っていれば、その中の文を実行します。そうでなければ、次のブロックの条件文を参照します。

もし、この条件が合っていれば
（例. ChaserがPlayerに触ったら）

そのときは、この処理を行う。
（ワワワワーと音を鳴らし、ゲームオーバーを表示する。）

複数の条件がある場合、プログラムは上から順に条件を評価し、最初に条件が一致したプログラムを実行します。このように、条件が上にある条件は、下にある条件よりも優先度が高くなります。

例えば、このゲームプログラムは、あらかじめ設定されたしきい値とポテンショメータ値を比較することで、Chaserの速さを決定します。



もし、ポテンショメータ値が 800より大きいとき、関数 Gameを呼び出します。（このとき、変数speedは250msに設定されます。）
でなければもし、ポテンショメータ値が 400より大きいとき、関数 Gameを呼び出します。（変数speedは500msに設定されます。）
でなければ、関数Gameを呼び出します。（変数 speedは750msに設定されます。）

他のブロックも使ってみよう

#1 [基本]:[数を表示_]ブロックと[ポテンシオビット]:[ポテンショメータ値]ブロックを使い、ポテンショメータ値を表示します。

#2 ポテンショメータは0 から 1023 までの値を返します。**[高度なブロック]** :**[入出力端子]**の**[数値をマップする_ 元の下限_ 元の上限_ 結果の下限_ 結果の上限_]**ブロックを使用して、読み込んだ数値をある意味を持った数値の範囲にマッピングすることができます。

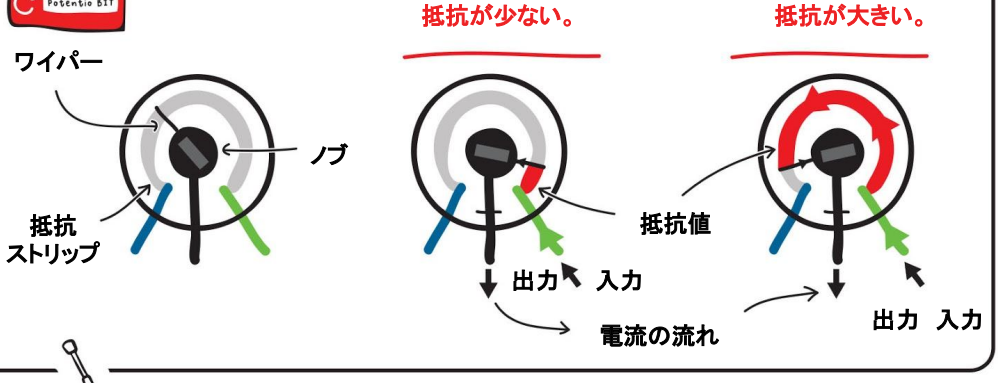
#3 マッピングブロックは小数点以下の数値（例： 1.68、3.998）を返します。数値を四捨五入するには、**[計算]**:**[小数点以下四捨五入 _]**ブロックを使用します。

#4 Potentio Bitから読み取った値を0～7の範囲にマッピングするサンプルプログラムを以下に示します。値は四捨五入されて、LEDに表示されます。

面白い事実！

ポテンショメータは、ポットにあるようなノブやスライダーを使って、抵抗値を調整できる可変抵抗器です。

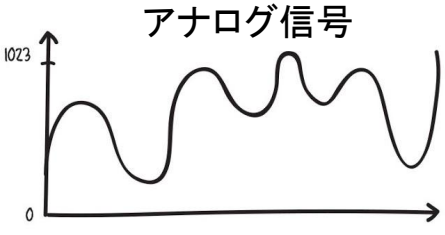
ポテンショメータが 10,000Ωの場合、ワイパーの位置を変えることで抵抗値を0～10,000Ωにすることができます。

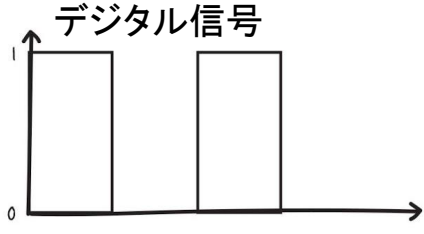


使用例

- スピーカーの音量
- ラジオの周波数制御
- 給湯器の温度調節

EDU:BITのポテンショメータはアナログ入力装置の一種です。電位を測定し、測定した電圧（0V～3.3V）を0～1023の整数値に変換します。

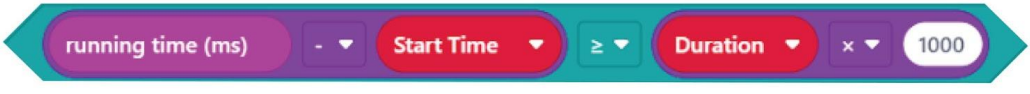




練習問題

EDU:BITにタイマー機能をプログラムしてみましょう。 Potentio Bitを使用して時間（60秒間）を調整し、Aボタンでタイマーを作動させ、Bボタンでリセットします。	
最初だけ	Modeを0に設定します。
Aボタン（黄色ボタン）が押されたとき	Modeを1に設定します。 Start Timeを稼働時間（ミリ秒）に設定します。 笑顔のアイコンを表示します。
Bボタン（青色ボタン）が押されたとき	Modeを0に設定します。
ずっと	Modeをチェックし続けます。 • Modeが0の場合は、ポテンショメータから読み取った値を 0～60の範囲にマッピングして四捨五入し、LEDに時間を表示します。 • Modeが1の場合は、条件「 稼働時間（ミリ秒）- Start Time ＞ (Duration x 1000)」をチェックし、合致しているときは「ワワワワー」と鳴らして、Modeを2にします。 • Modeが上記以外の場合、悲しい顔のアイコンを表示します。

ヒント:
ヒント#1 変数が3種類必要よ。(Mode、Start Time、Duration)
ヒント#2 稼働時間（ミリ秒）ブロックはツールボックスの[入力]にあるわよ。
ヒント#3 条件文を使って、タイマーが終了したかを判定してね。



チャプター 7

拍手を聞こう！

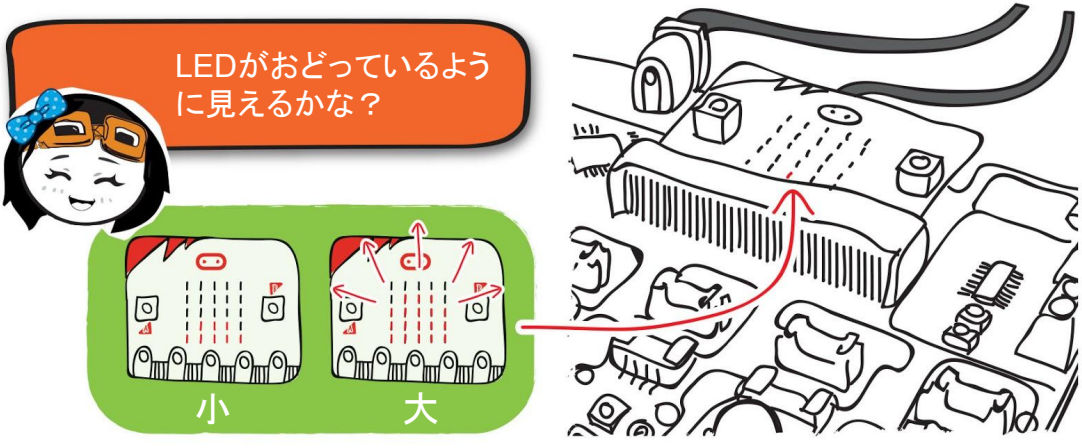
Sound Bit

作ってみよう！

Step 1 MakeCodeエディターで新規プロジェクトを作成し、EDU:BIT拡張機能を追加します(ページ40参照)。ツールボックスの **[LED]**をクリックし、**[棒グラフを表示する 値_最大値_]**ブロックを選択します。ブロックを **[ずっと]**ブロックにはめ込みます。

Step 2 ツールボックスの **[サウンドビット]**をクリックし、**[音量]**ブロックを選択します。ブロックを **[棒グラフを表示する 値_最大値_]**ブロックにはめ込み、2つ目の値を0から**1023**に変更します。

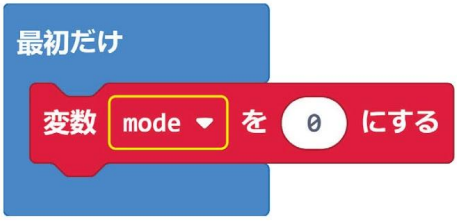
Step 3 プログラムをEDU:BITにフラッシングします。手を叩いたり、指を弾いたりしながら、LEDを観察してみましょう。



じゃあ、EDU:BITに拍手機能を作ってみよう

Step 4 新規プロジェクトを作成し、EDU:BIT拡張機能を追加します。 **[変数]**をクリックし、**[変数を追加する...]**を選択します。ポップアップウィンドウに **'mode'**と入力し「OK」を押します。同じやり方で変数を2つ追加し、それぞれ、 **'loudness'**と**'highest'**に変更します。

Step 5 **[変数]**から**[変数_を_にする]**ブロックを選択し、**[最初だけ]**ブロックにはめ込みます。変数名を 'mode'に変更します。



Step 6 **[入力]**から**[ボタン_が押されたとき]**ブロックを選択します。

Step 7 [変数]から[変数_を_にする]ブロックを選択し、[ボタンAが押されたとき]ブロックにはめ込みます。最初のブロックの変数を'mode'、値を1にします。2つ目のブロックは'highest'とデフォルト値の0に設定します。

Step 8 [IRビット]から[IRセンサーが感知したとき]ブロックを選択します。

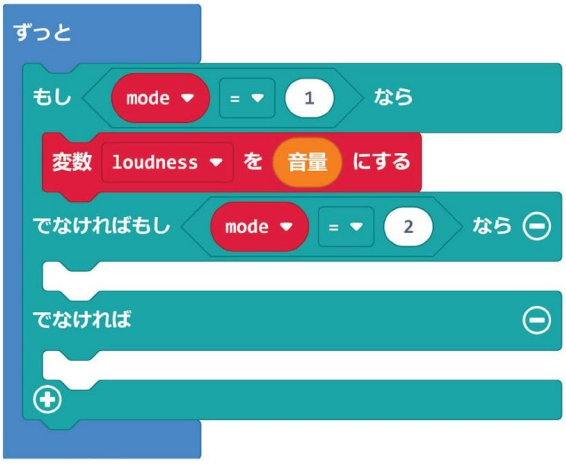
Step 9 [変数]から[変数_を_にする]ブロックを選択し、[IRセンサーが感知したとき]ブロックにはめ込みます。変数名を'mode'、値を2に設定します。

Step 10 **[論理]**から**[もし-なら-でなければ]**ブロックを選択し、**[ずっと]**ブロックにはめ込みます。プラスボタンを押して条件を追加します。

Step 11 **[論理]**から**[_=_]**ブロックを選択します。ブロックを複製し、**[もし-なら-でなければ]**ブロックにはめ込みます。

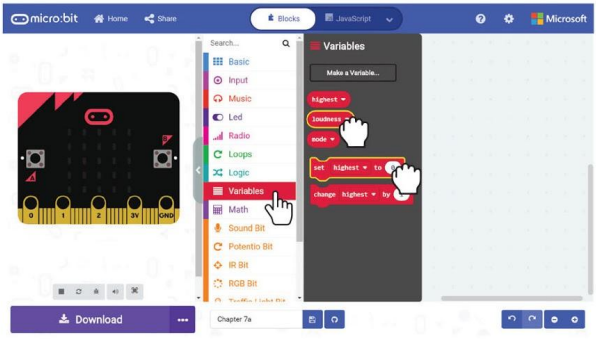
Step 12 **[変数]**から**[mode]**を選択し、条件ブロックの左辺にはめ込みます。右辺の値はそれぞれ **1**、**2**に設定します。

Step 13 [変数]から[変数_を_にする]ブロックを選択し、[もし-なら-でなければ]ブロックの1つ目の空欄にはめ込みます。変数を'loudness'に変更し、[サウンドビット]の[音量]ブロックを値にはめ込みます。



Step 14 [論理]から[もし-なら]ブロックと[_=_]ブロックを選択します。=は>に変更します。[変数]から[loudness]と[highest]を選択し、比較用ブロックにはめ込みます。

Step 15 [変数]から[変数_を_にする]ブロックを選択し、[もし_なら]ブロックにはめ込みます。[変数]から[loudness]ブロックを選択し、値に設定します。



Step 16 [LED]から[棒グラフを表示する値_最大値_]ブロックを選択します。[変数]から[loudness]ブロックを選択し、[棒グラフを表示する値_最大値_]ブロックにはめ込みます。値は**1023**に変更します。

Step 17 [基本]から[数を表示_]ブロックを選択し、
[もし-なら-でなければ]ブロックの2つ目の空欄にはめ込みます。
[変数]から[highest]ブロックを選択し、[数を表示_]ブロックにはめ込みます。

Step 18 [基本]から[アイコンを表示_]ブロックを選択し、[もし-なら-でなければ]ブロックの最後の空欄にはめ込みます。

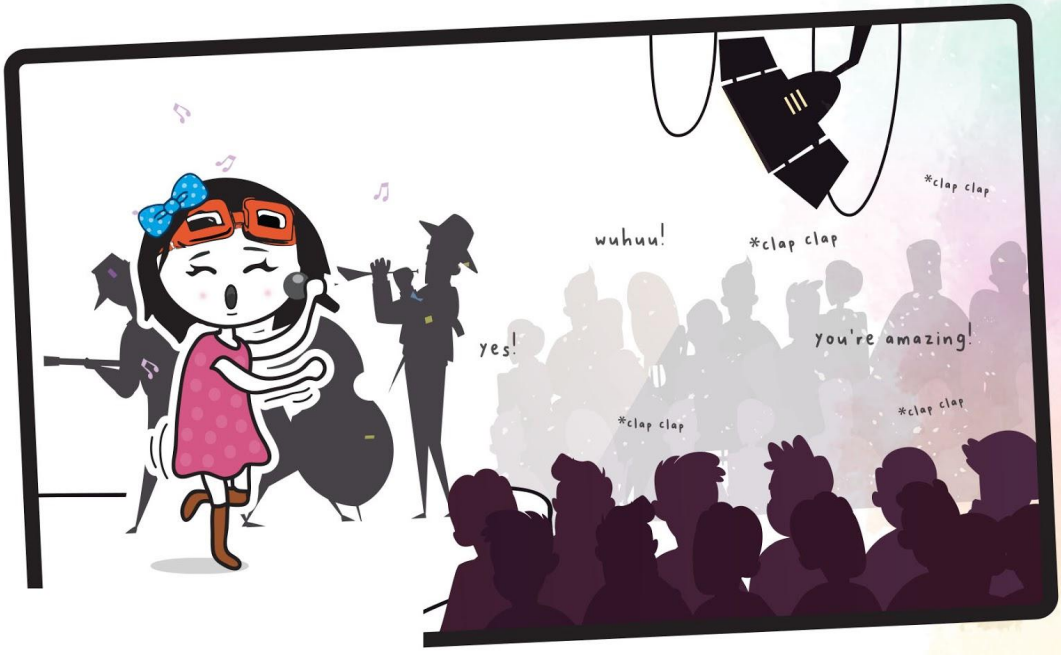
完成した
プログラムです。

がんばれ！

Step 19 EDU:BITにフラッシングすると、タレントタイムショーに使える、拍手出力メーター機能の完成です。

遊んでみよう！

拍手を聞こう！



遊び方：

- 個人、ペア、またはチームで、歌ったり、踊ったり、冗談を言ったりするパフォーマンスをする準備をしましょう。
- 準備ができたら、交代でパフォーマンスを披露していきましょう。観客は、パフォーマンスの後に拍手をします。面白ければ面白いほど大きな拍手をおくりましょう。
- 手拍子が鳴り止んだら、IR Bitを感知させて、スコア（録音された最大音量）をスクロール表示します。
- 次のパフォーマンスを始める前に、Aボタンを押してスコアをリセットします。
- 最も大きな拍手を受けた個人、ペア、チームが勝ちです。楽しんでください！

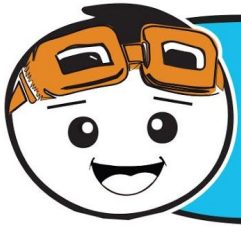
プログラムのかいせつ

1つのプログラム内に複数の処理ある場合、処理を実行するイベントトリガーを使用して1つの処理から別の処理に切り替えます。プログラムをスムーズに実行するために、ずっとループを使用して現在のモードを常にチェックし、対応するブロックを実行します。

最初だけmodeを0に設定。（スタンバイmode）

常に現在のmodeをチェックし、modeに対応した処理（ブロックの集まり）を実行します。

プログラム実行中に、あるmodeから別のmodeに変更するためのイベントトリガーブロック。

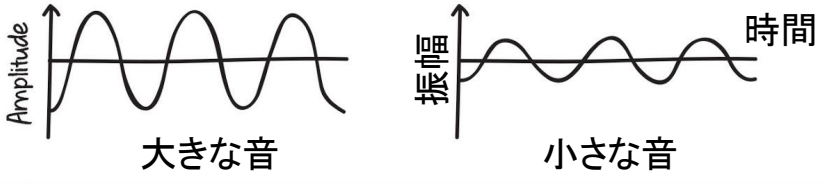


modeや処理を追加したいときは、[ゆさぶられたとき]や[音量が超えた時]などをイベントトリガーに使うこともできるよ。また、[もし-なら-でなければ]ブロックの（＋）ボタンを押すことで、条件を追加することもできるよ。

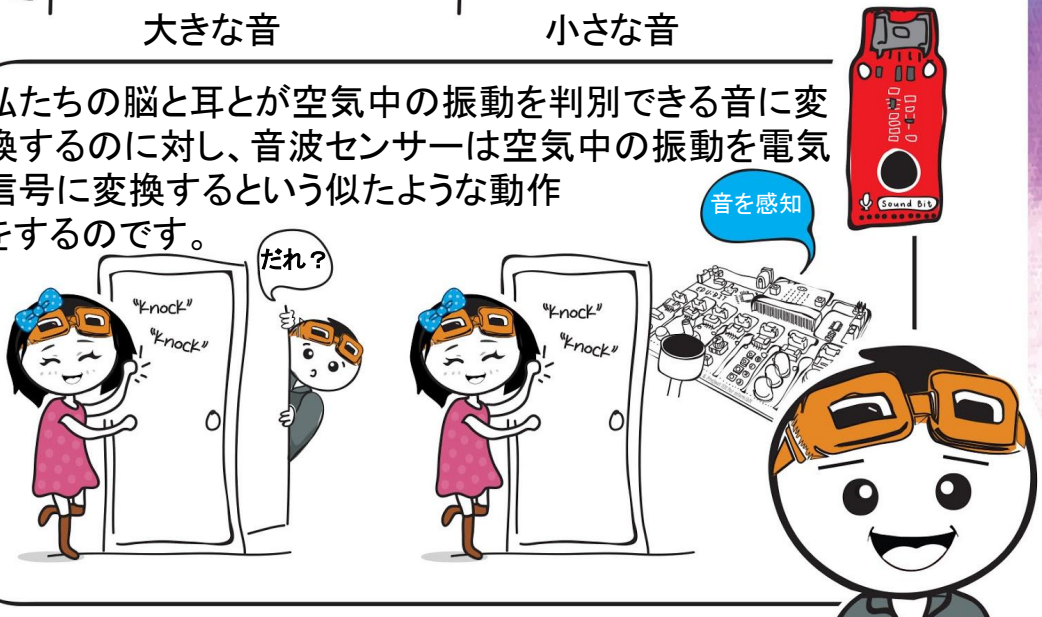
面白い事実！

音は物体が振動することで発生し、例えばドラムを叩くと音が発生します。その振動によって周囲の空気分子（媒体）が振動し、音波が発生します。

音波センサは、音波の強さ（音の大きさ）を検出して電気信号に変換するモジュールです。



私たちの脳と耳とが空気中の振動を判別できる音に変換するのに対し、音波センサーは空気中の振動を電気信号に変換するという似たような動作をするのです。



使用例

- 防犯ベルの音探知機
- 音に反応する照明
- 聴覚ベビーモニター

僕たち人間は外の音を全部聞き取ることができると思う？　どうして？

練習問題

EDU:BITに教室のノイズモニターをプログラムしてみましょう。
Traffic Light BitのLEDを使用して、音量を表示します。

音のレベル	音のレベルの範囲	Traffic Light BitのLED
うるさい	()～1023	赤色LED
少しうるさい	()～()	黄色LED
ちょうどいい	0～()	緑色LED

ヒント：

ヒント#1 それぞれの音のレベルのしきい値を前もって決めておく必要があります。
ヒント#2 モニタリングを安定させるには、一定の間隔で音のレベルの値の平均値を取得します。

簡単すぎたら、レベルアップにチャレンジしてみてください。
しきい値がポテンショメータの値に対して相対的になるようにプログラムを修正してください。

チャプター 8

ぐるぐる回そう！

DC Motor

作ってみよう！

Step 1 MakeCodeエディターで新規プロジェクトを作成し、EDU:BIT拡張機能を追加します(ページ40参照)。**[入力]**から**[ボタン_が押されたとき]**を選択します。

Step 2 **[モーターとサーボ]**から**[モーターを動かす__速さ_]**ブロックと**[モーターを止める_]**ブロックを選択します。速さの値は**80**に設定します。

Step 3 **[基本]**から**[一時停止（ミリ秒）_]**ブロックを選択し、**[モーターとサーボ]**から**[モーターを動かす__速さ_]**ブロックと**[モーターを止める_]**ブロックの間にはめ込みます。

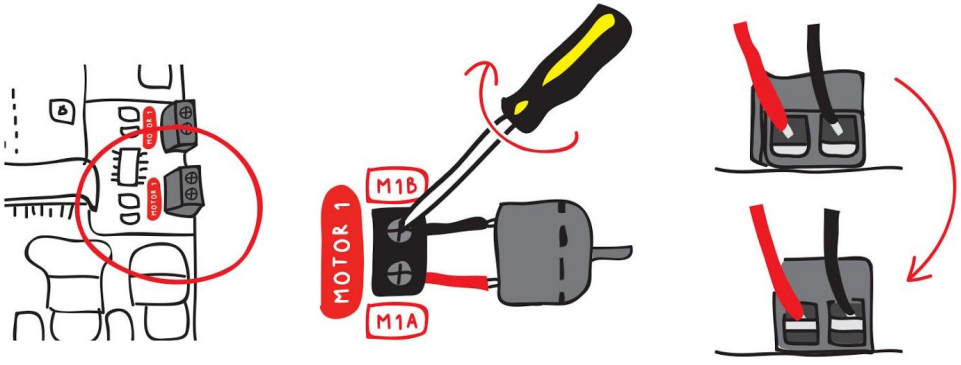
Step 4 **[計算]**から**[ランダムな数字を選択：_から_まで]**ブロックを選択します。**[一時停止（ミリ秒）_]**ブロックにはめ込み、値をそれぞれ **200**と**1000** に変更します。

Step 5 **[音楽]**から**[メロディを開始する_くり返し_]**ブロックを選択し、メロディを**"ピコーン！"**に変更します。（好きなメロディでも構いません。）

Step 6 完成したプログラムを EDU:BITにフラッシングします

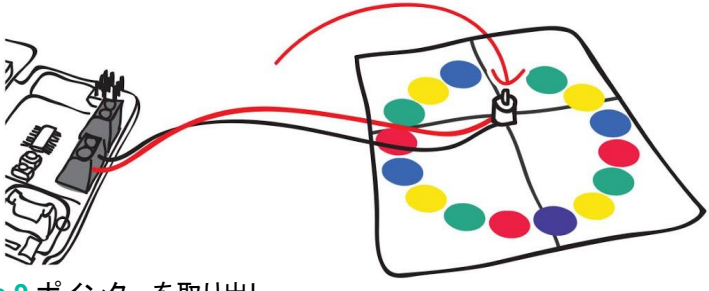
このプログラムを使って、ランダムスピナー（回転機）が必要なゲームを作ることができるわよ。モーターは動き出すと回転を始め、ランダムな時間が過ぎると止まるの。

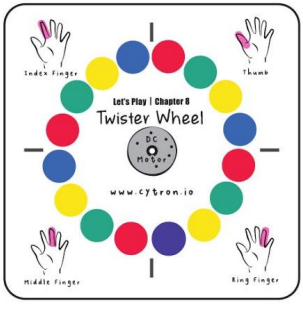
Step 7 MOTOR1端子にDCモータを接続してください。-(i) 導線を差し込みます。(ii) 付属のドライバーを使用してネジを締め、固定させます。



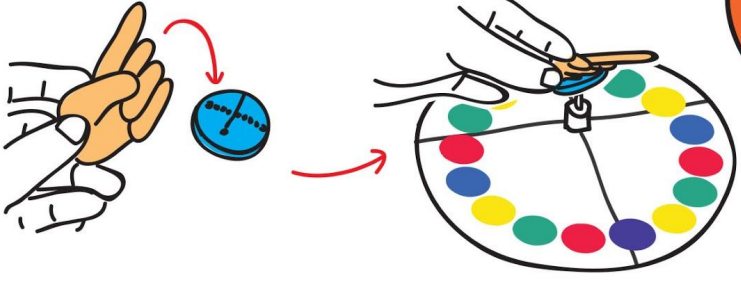
黄色ボタン（Aボタン）を押してテストしてみよう。もしモーターが回らなかったら、配線が正しく接続されているか、電源が入っているかを確認してみてね。

Step 8 DCモーターの導線をツイスターホイールの中心に取り付けるには、両面テープや熱接着剤などを使用します（図参照）。





Step 9 ポインターを取り出し、
接着剤でプラスチック製のディスクに取り付けます。
その後、モーターシャフトの所定の位置にディスクを固定します。



ツイスターホイール、ポインター、ゲームマップは付属品に含まれているからね。

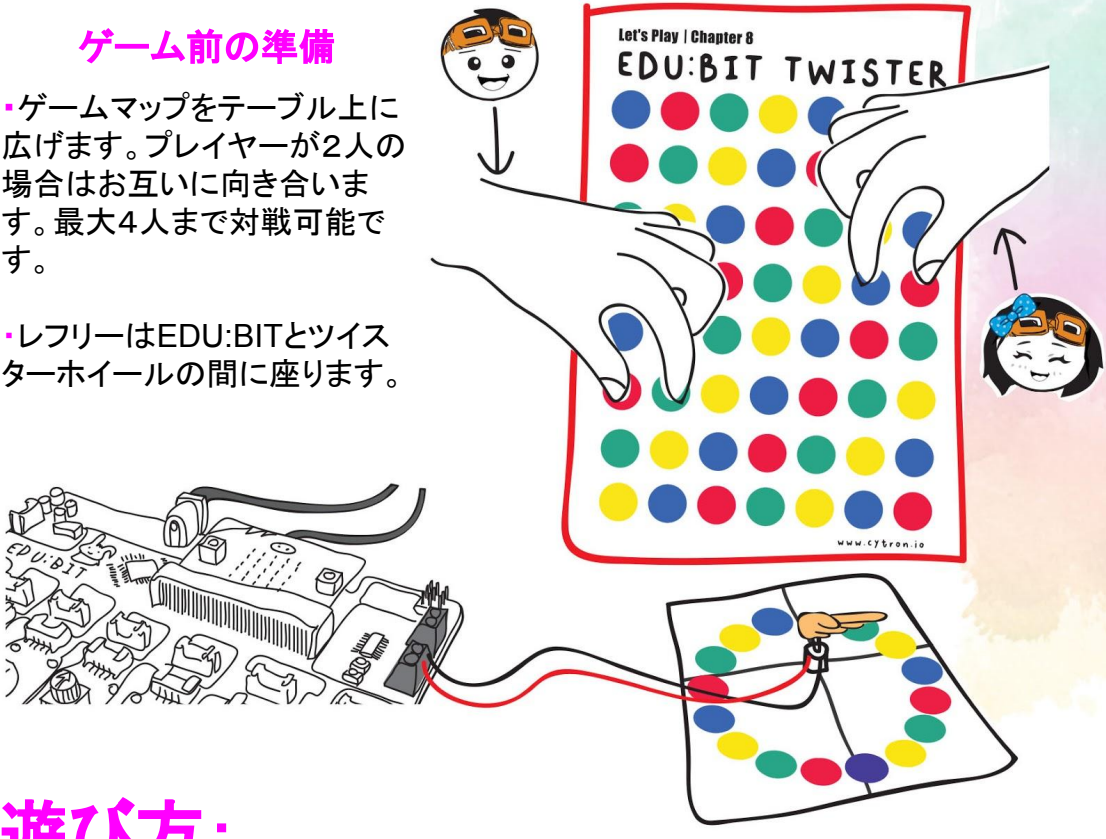
遊んでみよう！

ぐるぐる回そう！

ゲーム前の準備

・ゲームマップをテーブル上に広げます。プレイヤーが2人の場合はお互いに向き合います。最大4人まで対戦可能です。

・レフリーはEDU:BITとツイスターホイールの間に座ります。



遊び方：

このゲームでは、レフリーの指示に従って、プレイヤーは交代で、ゲームマップ上の色のついた円に指を置いていきます。

レフリーの役割は、黄色のボタン（またはAボタン）を押してポインターを回転させ、ポインターが指している指と色をコールすることです。例えば、「人差し指、赤」と。
自分の番になったら、レフリーが指示した指を指定した色のついた円に置きます。すでに同じ色に指があるときは、別の場所に移動させます。

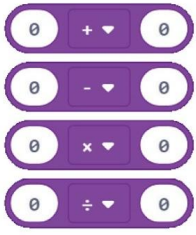
指置きに失敗したらゲーム退場です。

最後まで残ったプレーヤーが勝ちです。

他のブロックも使ってみよう

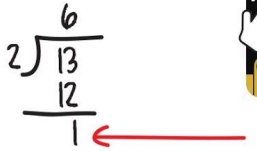
[計算]を使って、変数に計算結果を設定することができます。

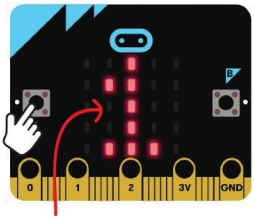
#1 足し算、引き算、掛け算、割り算を行うには、以下のブロックを使います。



#2 [_を_で割ったあまり]を使えば、割り算で割り切れないとき、余った数を計算することができます。

例:





余り

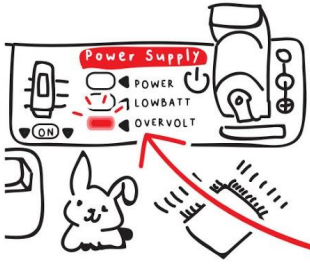
#3 [_を_で割ったあまり]を使えば、数字が偶数か奇数かを判別することができます。単純に2で割った余りが”1”の場合は”奇数”、余りが”0”の場合は”偶数”です。やってみましょう！

Aボタンを一番最初に押し、EDU:BIT上に　”odd”と表示され、次に押したときは”even”と表示されるのに気づいたかな？ Aボタンを99回押したらどうなるかな？

面白い事実！

一般的に**DCモータ**と呼ばれる直流モータは、電気エネルギーを機械のエネルギーに変換する回転式の機械です。

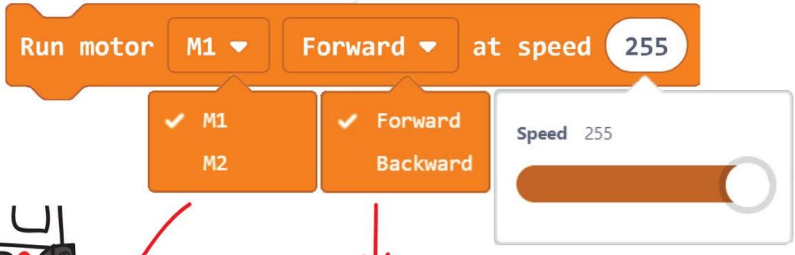
直流モータを回転させるためには、入力電圧を調整する必要があります。
入力電圧を調整することで、回転速度を制御することができます。入力電圧が高いほど、モーターの回転速度が速くなります。EDU:BITキットのDCモーターの推奨電圧は3.6V～6Vです。

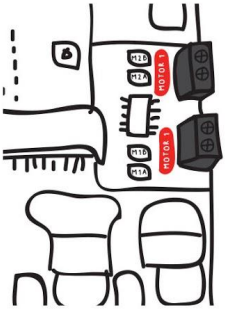


注意！
推奨電圧を超える高電圧をモーターに加えすぎると、モーターの寿命が短くなります。

過電圧表示器

以下のブロックを使って、DCモーターの回転方向や回転スピードを変えることができます。





回転方向

EDU:BIT基板にはDCモータ端子が2つあります。いずれかを選択してください。

0から255までの相対的な値です。大きい値ほど、モーター速度が上がります。

EDU:BITにはモーターテスト回路が内蔵されているのよ。白いボタン（M1A、M1B、M2A、M2Bとラベル表示）を押して、接続が確実に行われているかどうか、モーターが正常に動作しているかどうかを確認してね。

練習問題

EDU:BITに、音量に反応し、Potentio Bitで速度が制御される、ファン機能をプログラムしましょう。

最初だけ	ハートのアイコンを表示します。 （お気に入りのアイコンでもOKです。） 変数Modeの値を0に設定します。
音量が超えたとき > _ (しきい値)	変数Modeの値を+1します。
ずっと	変数'Speed'をポテンショメータ値とマッピングさせます。元の下限を0、元の上限を1023、結果の下限を0、結果の上限を255とします。 常にModeの値をチェックします。 ・Modeが偶数のとき、モーター M1を停止します。 ・Modeが奇数のとき、モーター M1を動作させます。(モーターの速さは、ポテンショメータ値とマッピングさせた変数'Speed'によります。)

ヒント

ヒント#1： モーターを作動させたり停止させたりするには、音量トリガー（例.しきい値）を決めます。

ヒント#2： ModeとSpeedの2つの変数を作ります。

ヒント#3： ファンの羽根をモーターシャフトに取り付け、プログラムを実行します。モーターが回転しているときに空気が吹いている感じがしない場合は、プログラムで回転方向を変更させます。

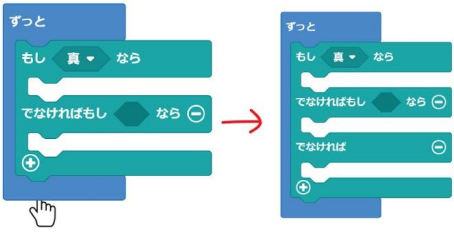
チャプター 9

シュート...ゴール！

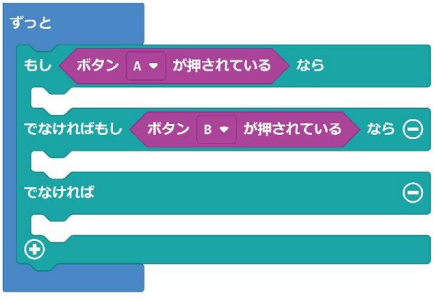
Servo Motor

作ってみよう！

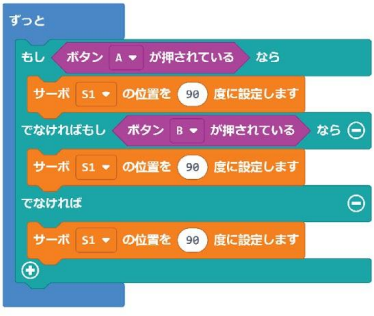
Step 1 MakeCodeエディターで新規プロジェクトを作成し、EDU:BIT拡張機能を追加します(ページ40参照)。**[論理]**から**[もし-なら-でなければ]**ブロックを選択し、[ずっと]ブロックにはめ込みます。＋ボタンを押して条件文を追加してください。



Step 2 **[入力]**から**[ボタン_が押されている]**ブロックを選択します。ブロックを複製し、**[もし-なら-でなければ]**ブロックの条件欄にはめ込みます。2つ目のブロックの値を**'Button B'**に変更してください。

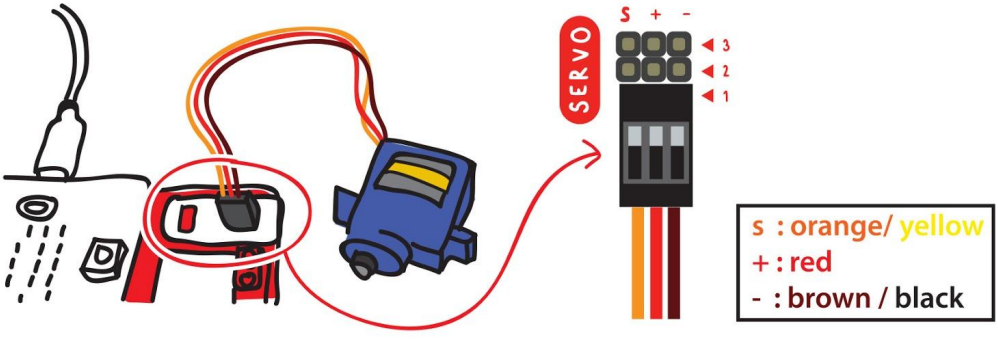


Step 3 **[モーターとサーボ]**から**[サーボ_の位置を_度に設定します]**ブロックを選択します。ブロックを複製し、**[もし-でなければもし-でなければ]**ブロックの各空欄にはめ込みます。

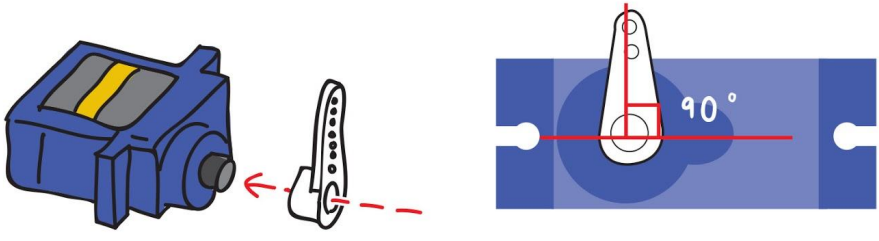


Step 4 1つ目と2つ目のブロックの値をそれぞれ、**30**と**150**に変更します。EDU:BITにフラッシングしてください。

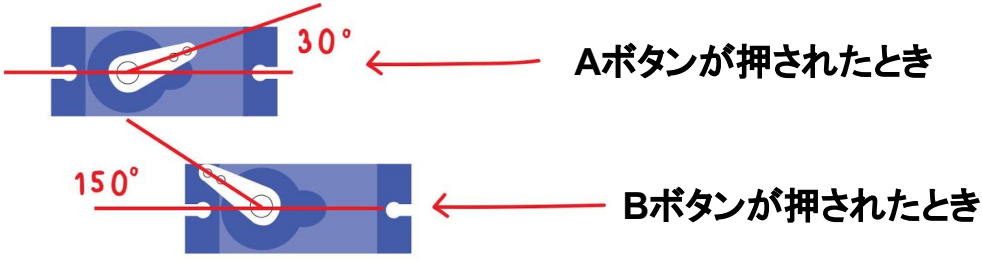
Step 5 サーボモーターのケーブルを下図のように、Servo Port1に接続してください。



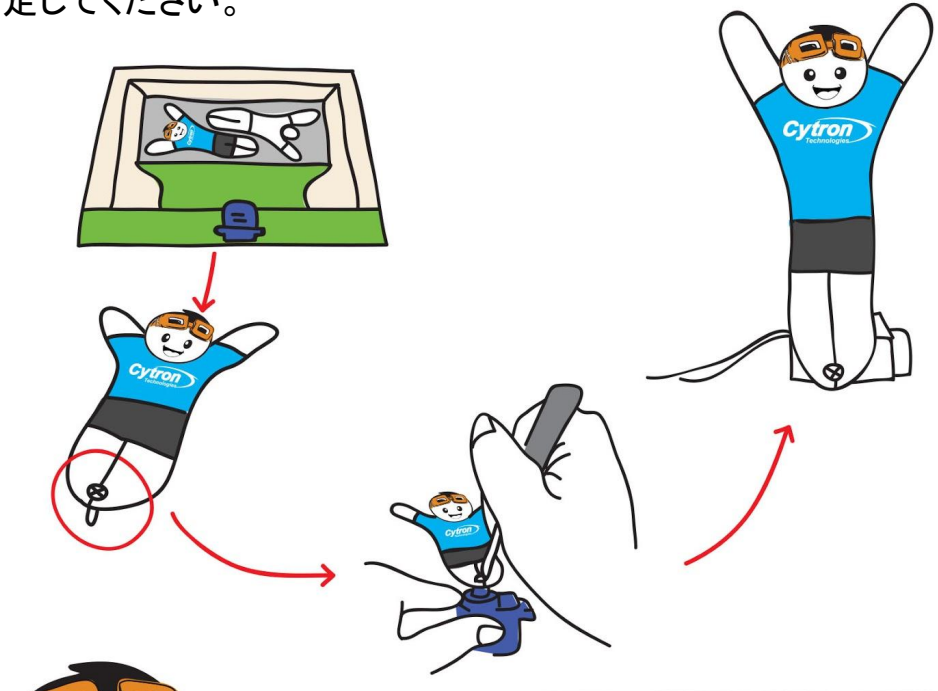
Step 6 EDU:BITの電源を入れ、下図のように、サーボアームをサーボモーターの軸に取り付けます。

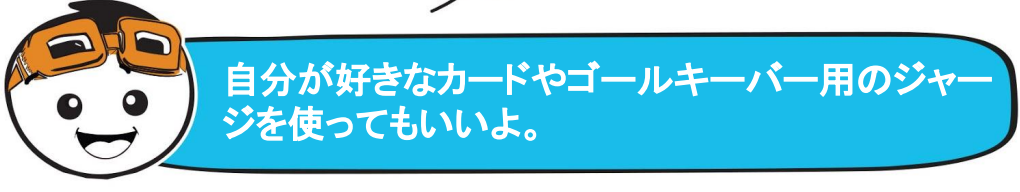


Step 7 AボタンとBボタンを押して、テストしてみてください。

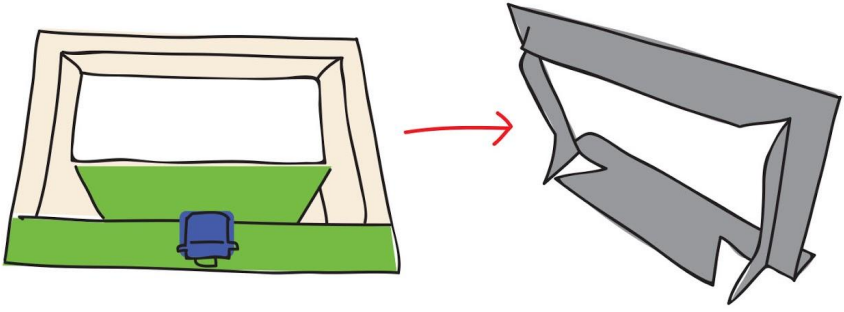


Step 8 付属品からゴールキーパーを取り出します。付属のドライバーとネジを使って、ゴールキーパーをサーボアームに固定します。両面テープや熱接着剤等を使用して、ゴールキーパーをサーボアームにしっかりと固定してください。

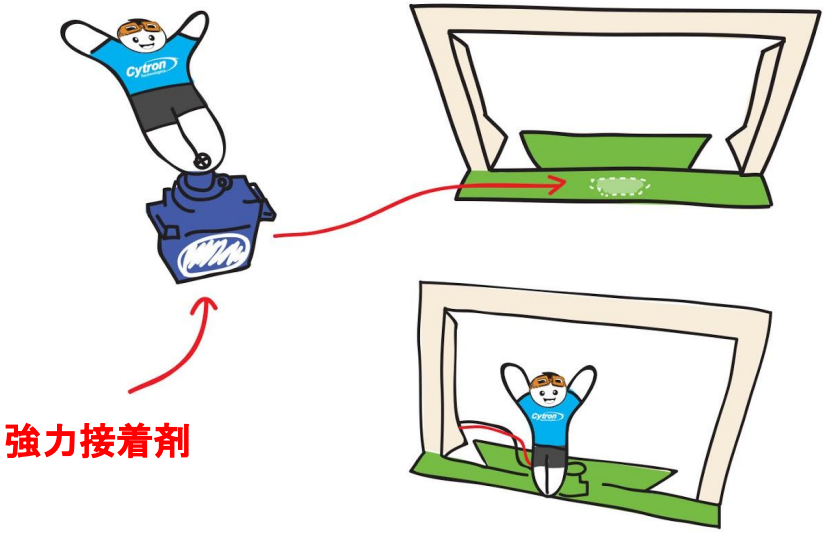




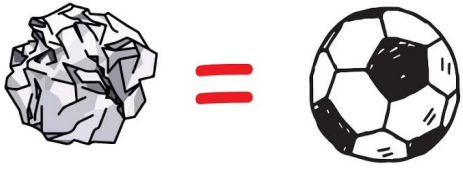
Step 9 ゴールポストを取り出し、下図のように設置します。



Step 10 両面テープや熱接着剤などを使用して、サーボモーターを下図の位置にしっかりと固定してください。

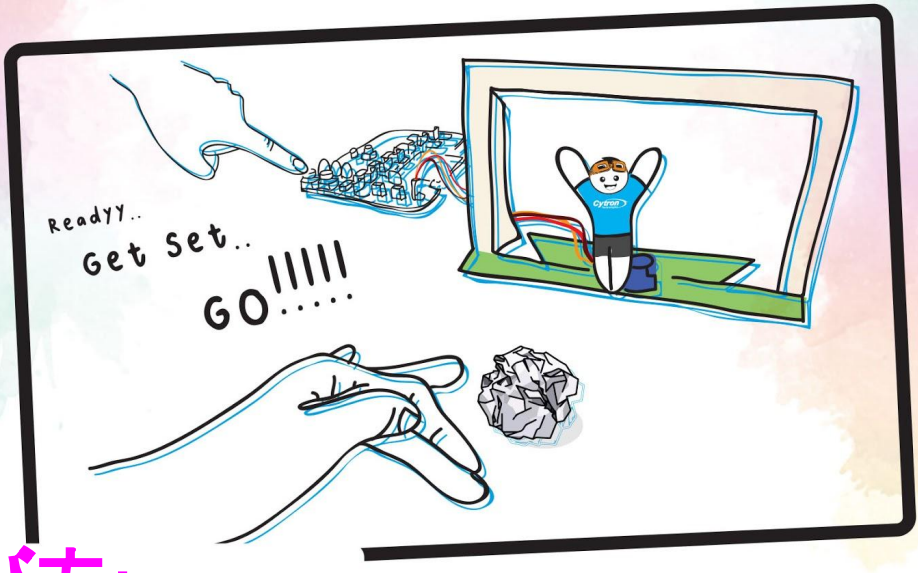


Step 11 適当な紙を丸めてサッカーボールの代わりにします。楽しい PK 戦の用意は、これで完了です。準備は OK？



遊んでみよう！

シュート...ゴール！



遊び方：

ゴールポストを設置し、ペナルティマークをマークします。（ゴールから3m程度。年齢に合わせて調整してください）

参加者はキーパーとキッカーを交代で行います。

キッカーは、ゴールに向かって指でボールを弾きます。

キーパーはシュートを防ぎます。黄色ボタン（Aボタン）で左に移動、右ボタン（Bボタン）で右に移動します。

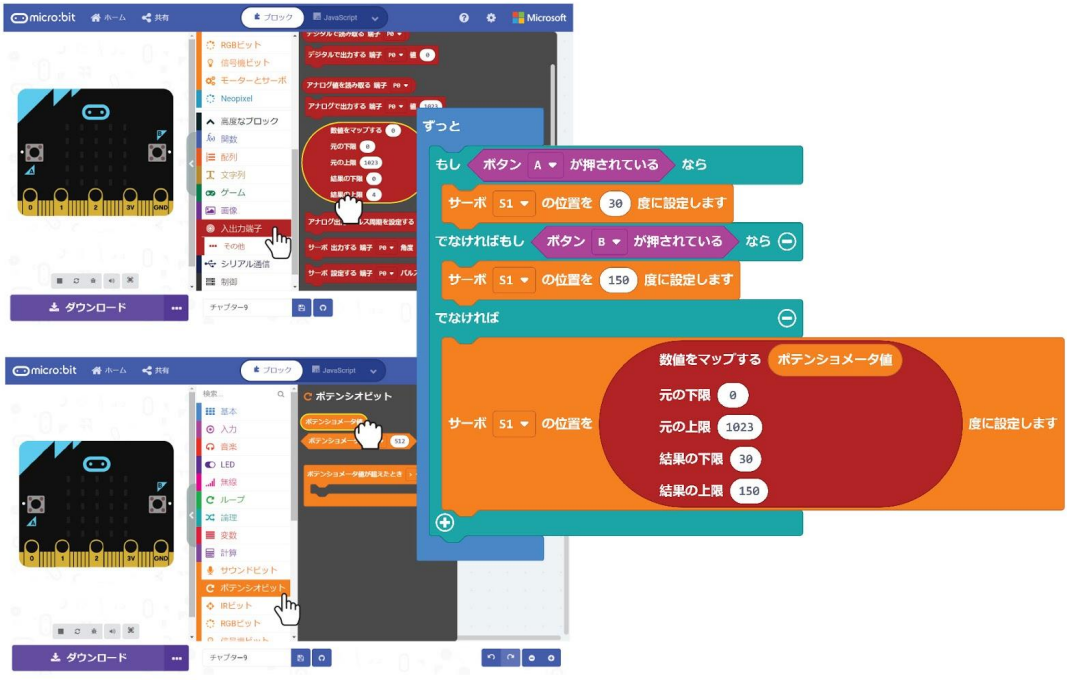
キッカーはシュートを5本、放ちます。得点の多いプレイヤーが勝ちです。

知ってる？サッカーの試合では、試合終了時にスコアが同点になって、延長戦でも引き分けの場合、勝者を決める PK戦が行われるんだ。PK戦では、相手チームのゴールキーパーだけが守っているゴールに向かって、ペナルティマークから 5本のシュートを蹴るんだよ。そして、ゴールを決めた本数が多いチームが勝つんだ。

さっきのプログラムでは、ボタンを押してゴールキーパーを左右に動かしたけど、Potentio Bitを使ってゴールキーパーを動かすこともできるんだ。

Step 12 **[高度なブロック]**の**[入出力端子]**から**[数値をマップする_元の下限_元の上限_結果の下限_結果の上限_]**ブロックを選択します。

Step 13 **[ポテンシオビット]**から**[ポテンショメータ値]**ブロックを選択し、**[数値をマップする_]**の値にはめ込みます。4つ目と5つ目の値は、それぞれ、30、150に設定します。



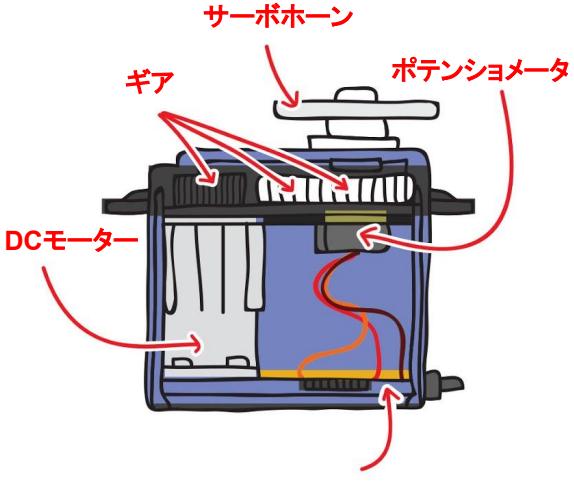
Step 14 EDU:BITにプログラムをフラッシングします。これで、Potentio Bitを使ってゴールキーパーを動かすことができます。試してみてください！

PKの練習をしたい場合は、「練習モード」というプログラムを作って、ゴールキーパーを左右に動かし続けみよう！

面白い事実！

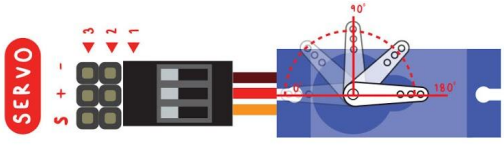
EDU:BITキットに搭載されている **サーボモーター** は、RC（ラジコン）サーボとも呼ばれ、ラジコンや小型ロボットを制御するために広く使われています。

サーボモーターは、電源 (+)、グラウンド(-)、制御信号による3線式システムを使用しています。一般的には、DCモーター、ギア、ポテンショメータ（位置センサー）、制御回路で構成されています。



サーボモーターは、電源 (+)、グラウンド(-)、制御信号による3線式システムを使用しています。一般的には、DCモーター、ギア、ポテンショメータ（位置センサー）、制御回路で構成されています。

連続回転するDCモーターとは異なり、サーボモーターの場合、回転を0～180°の範囲で自由に制御することができます。



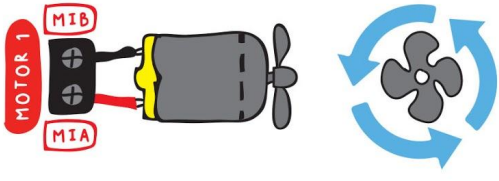
0-180°回転

サーボモーター

VS

DC モーター

0-360°回転



練習問題

EDU:BITにメトロノーム機能をプログラムしてみましょう。電源を入れたら、サーボモーター付属のポインターが一定のテンポで左右に振れるようにします。テンポは Potentio Bitが制御します。黄色ボタン（Aボタン）が
押されたとき、現在のテンポを表示します。
（例.120bpm）

ヒント

ヒント#1 TempoとDelayという2つの変数を作ります。
ヒント#2 メトロノームのテンポは一般的に40～200bpmです。
ヒント#3 60bpm(beat per minute)の場合、メトロノームが
1秒の間に60回、左右どちらかに振れます。(1秒に1回)
ヒント#4 テンポ(Tempo)が速ければ速いほど、遅延(Delay)が
小さくなります。

メトロノームとは、一定の間隔でクリック音やその他のサウンドを生成する装置です。通常は1分あたりのビート数（BPM）が用いられます。音楽家はこのデバイスを使用して、定期的なテンポで演奏の練習をします。同期した視覚運動（振り子の揺れやライトの点滅など）メトロノームの内に含まれます。

- ウィキペディア -

チャプター 10

マスターマインド、プログラムを見破れる？

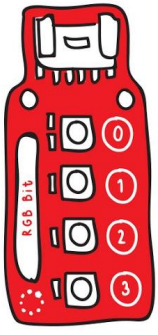
RGB Bit

作ってみよう！

Step 1 MakeCodeエディターで新規プロジェクトを作成し、EDU:BIT拡張機能を追加します。**[RGBビット]**から**[RGBピクセル _に設定_]**ブロックを選択します。

Step 2 **[RGBピクセル _に設定_]**ブロックを複製し、同じブロックを 4個作ります。

Step 3 作ったブロックを**[最初だけ]**ブロックにはめ込み、RGB ピクセル番号を0、**1**、**2**、**3**の順になるよう変更します。

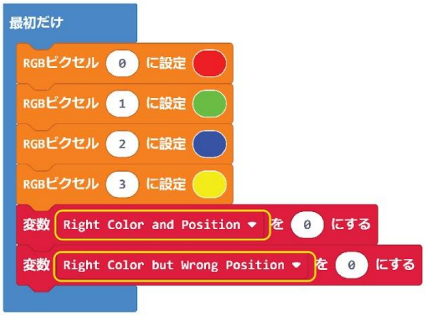


RGB Bitには4個のLEDがあり、それぞれに個別の番号（0～3）が割り振られています。この番号を使ってLEDを制御しましょう。

Step 4 作ったプログラムをEDU:BITにフラッシングします。

Step 5 変数”**Right Color and Position**”（色と位置も正解）と変数”**Right Color but Wrong Position**”（色は正解、位置は不正解）を作ります。

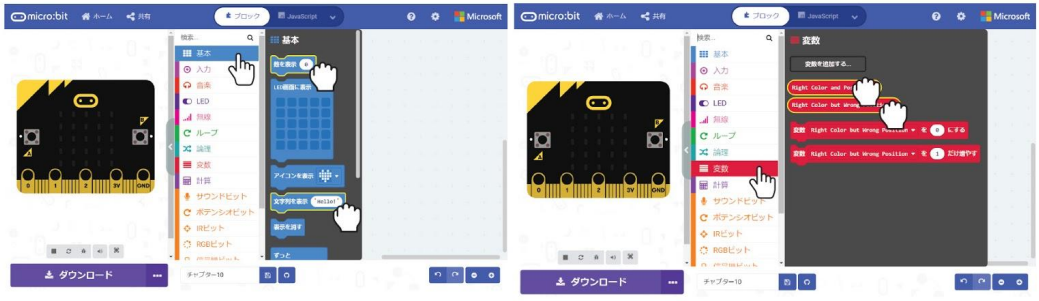
Step 6 [変数]から[変数_を_にする]ブロックを選択します。ブロックを複製し、[最初だけ]ブロックにはめ込みます。変数をそれぞれ、”Right Color and Position”と”Right Color but Wrong Position”に変更します。



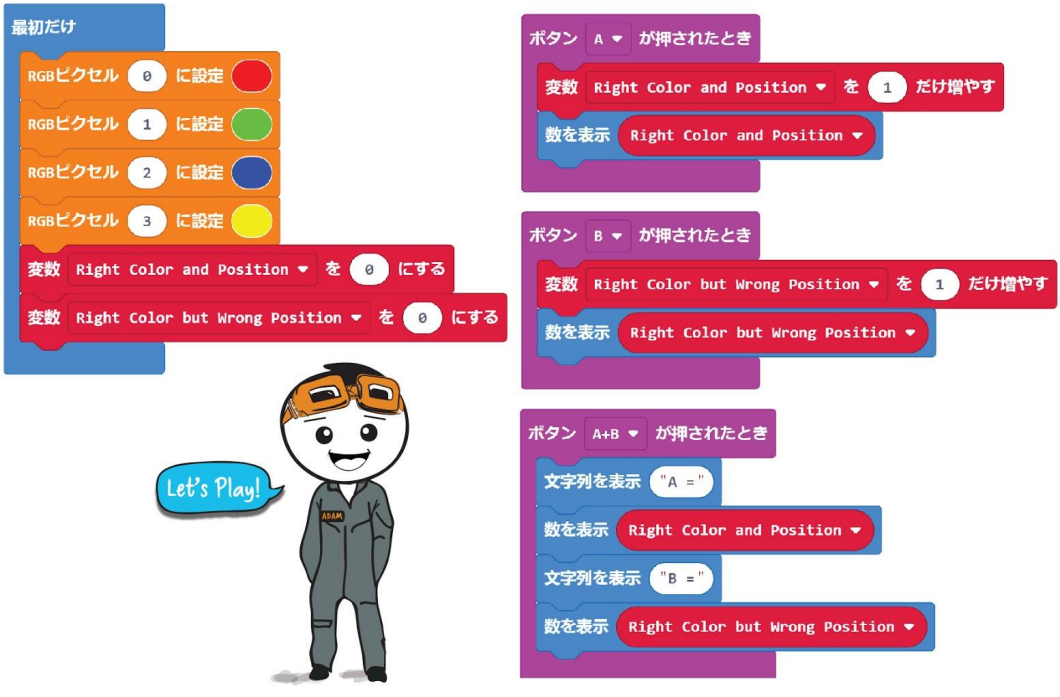
Step 7 [入力]から[ボタン_が押されたとき]ブロックを選択します。ブロックを複製し、2つ目の値を**B**、3つ目の値を**A+B**に変更します。

Step 8 [変数]から[変数_を_だけ増やす]ブロックを選択します。複製し、[ボタンAが押されたとき]ブロックと[ボタンBが押されたとき]ブロックにはめ込みます。変数をそれぞれ、”Right Color and Position”と”Right Color but Wrong Position”に変更します。

Step 9 [基本]から[数を表示 _]ブロックと[文字列を表示 _]ブロックを選択します。[変数]から[Right Color and Position]ブロックと[Right Color but Wrong Position]ブロックを選択します。

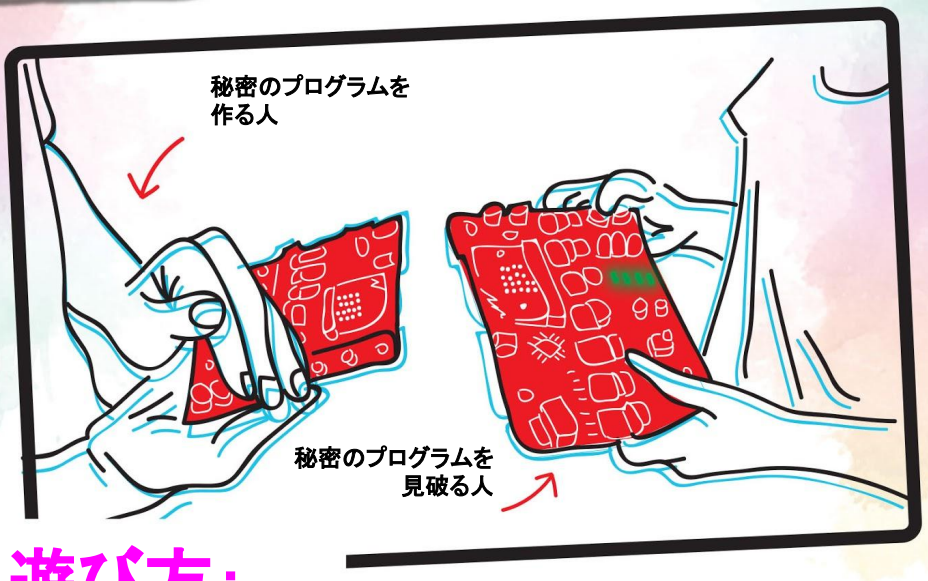


Step 10 [文字列を表示]ブロックの”Hello”を”**A=**”と”**B=**”に変更します。完成したプログラムがこちらです。



遊んでみよう！

マスターマインド、プログラムを見破れる？



遊び方：

①プレイヤー1（秘密のプログラムを作る人）は、4つのLEDを適当な順番で点灯させる秘密のプログラムを作ります。最初だけ、色は　と　にしましょう。LEDの色を手で隠して、対戦相手に見せないようにしましょう。

②プレーヤー2（秘密のプログラムを見破る人）は、プレーヤー1のプログラムを見破ります。自分のRGB LEDを点灯させてプレーヤー1に見せます。

③プレーヤー1は、黄色ボタン（Aボタン）または青色ボタン（Bボタン）を押して、”色と位置が正解”の数と”色は正解、位置は不正解”の数を表示します。

④プレーヤー2は黄色ボタンと青色ボタンの同時押しで、これまでに正解した数を確認します。

⑤プレーヤー2が色と位置を全部正解するまで②と③を10回くり返します。

⑥プレーヤー1とプレーヤー2とで役割を交代します。

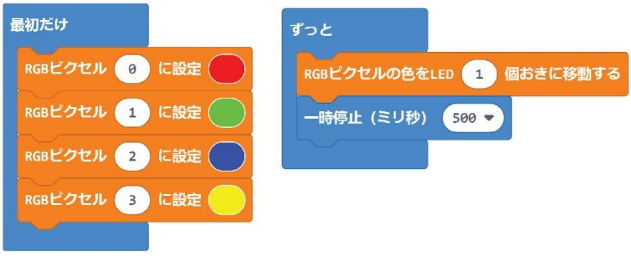
勝敗

プレーヤー2が色と位置を全て正解したら勝ちです。不正解の場合は、プレーヤー1の勝ちとなります。

他のブロックも使ってみよう

#1 **[RGBピクセル_に設定_]**ブロックに**[RGBピクセルをレインボーパターンに設定]**ブロックに置き換えると、LEDが虹のように点灯するよ。

#2 **[RGBピクセルの色をLED_個おきに移動する]**ブロックを**[ずっと]**ブロックの中で使い、**[一時停止（ミリ秒）_]**ブロックを使ってプログラムの実行速度を遅くすることで、ランニングライト効果を見ることができます。こちらが、サンプルプログラムです。



#3 **[RGBピクセルの色をLED_個分消す]**ブロックを**[ずっと]**ブロックの中で使い、**[一時停止（ミリ秒）_]**ブロックを使ってプログラムの実行速度を遅くすることで、LEDが1つずつ消えていく表示にすることができます。こちらが、サンプルプログラムです。

設定されている値を正数から負数に変えると、LEDが点灯／消灯する方向を変えることができるわよ。

面白い事実！

図工の時間で、赤、黄、青の 3原色を習ったと思います。それらの色を混ぜ合わせると、下図のような色になります。

ただし、テレビやコンピュータの画面など、光を使って色を表現する機器では、RGBカラーモデルが使用されます。

上の図では、赤（R）、緑（G）、青（B）の3つの光を組み合わせて、白色の光を作り出しています。

練習問題

EDU:BITに記憶力テスト機能をプログラムしてみましょう。

テスト方法

- EDU:BITを左に傾けると、RGB BitのLEDが数秒間、いずれかの色で点灯します。
- LEDが消えるまで観察し、消えた後、点灯した色の名前と順番を答えます。
- Bボタンを押して、RGB BitのLEDを先ほどと同じように点灯させてください。答え合わせをします。
- 正解の場合、黄色ボタン（Aボタン）を押して、自分の得点を表示します。不正解ならゲームオーバーです。
- Potentio Bitのノブを回して、LEDの点灯時間を変えることにより、ゲームの難易度を調整します。
- 得点の高い参加者が勝ちです。

ヒント：
ヒント#1 変数Score（スコア）と変数 Pattern（パターン）を作ること。
ヒント#2 パターン毎に、どの LEDを何色で光らせるかを前もって、決めておくこと。色の種類やパターンを増やしてゲームを難しくすることができるわよ。その逆もできるので、小さい子には色の種類やパターンを減らして易しくしてあげてね。

ボーナスチャプター

LEDを使った「サイモンさんは言いました」

ラジオ通信

link.cytron.io/edubit-bonus-chapter

どんなコミュニケーションでも、少なくとも送信者と受信者という存在が必要です。このゲームでは、無線放送の信号を送受信して通信するので、EDU:BITが2つ必要です。

作ってみよう！

Step 1 MakeCodeエディターで新規プロジェクトを作成し、EDU:BIT拡張機能を追加します。**[無線]**から**[無線のグループを設定 _]**ブロックを選択し、**[最初だけ]**ブロックにはめ込みます。

Step 2 **[基本]**から**[アイコンを表示_]**ブロックを選択し、プログラムに追加します。

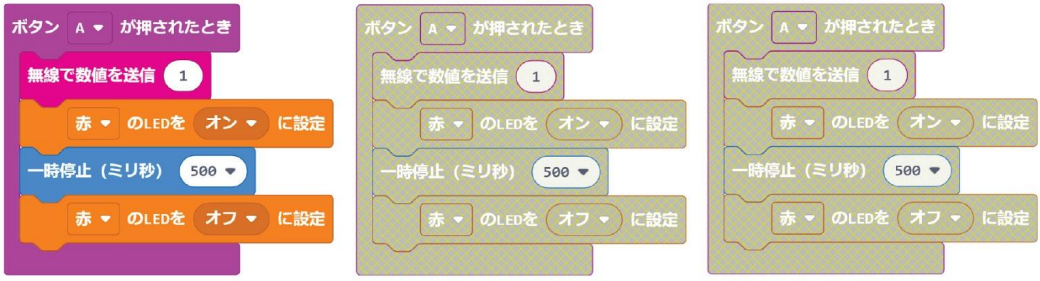
Step 3 **[入力]**から**[ボタン_が押されたとき]**ブロックを選択します。

Step 4 **[無線]**から**[無線で数値を送信]**ブロックを選択します。値を**1**に変更します。

Step 5 **[信号機ビット]**から**[_のLEDを_に設定]**ブロックを選択します。複製し、**[ボタン_が押されたとき]**ブロックにはめ込みます。2つ目のブロックの値を'オフ'に変更します。

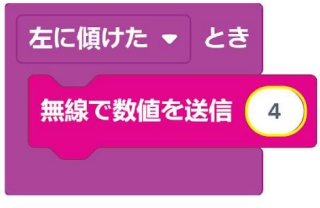
Step 6 **[基本]**から**[一時停止（ミリ秒）_]**を選択し、**[_のLEDを_に設定]**ブロックの間にはめ込みます。値を **500**に変更します。

Step 7 **[ボタン_が押されたとき]**ブロックを右クリックし、複製します。同じブロックの集まりを3セット作ります。



Step 8 複製したブロックのボタン、数値、色を下図のように変更します。

Step 9 **[入力]**から**[_とき]**ブロックを選択し、値を**'左に傾けた'**に変更します。**[無線]**から**[無線で数値を送信]**ブロックを選択します。数値を**4**に変更します。



Step 10 **[無線]**から**[無線で受信したとき receivedString]**ブロックを選択します。

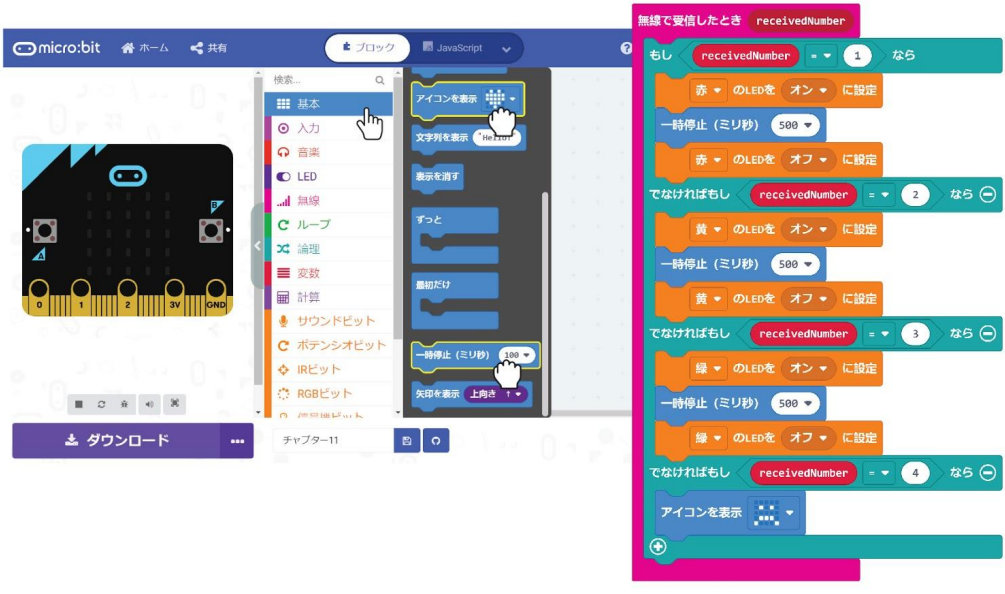
Step 11 **[論理]**から**[もし-なら-でなければ]**ブロックを選択します。

ボタンをクリックし、**[でなければもし]**条件を3つ作ります。　ボタンをクリックし、**[でなければ]**条件を削除します。**[論理]**から**[_=_]**ブロックを選択し、[もし]と[でなければもし]ブロックにはめ込みます。

Step 12 変数**'receivedString'**を下図のように各条件の左辺にドラッグ &ドロップではめ込みます。右辺の値をそれぞれ、**1**, **2**, **3**, **4**に変更します。

Step 13 **[信号機ビット]**から**[_のLEDを_に設定]**ブロックを選択します。ブロックを複製し、条件文の最初の 3つの空欄にはめ込みます。色とオン／オフを下図のように設定します。

Step 14 **[基本]**から**[一時停止（ミリ秒）_]**ブロックを選択します。複製し、**[_のLEDを_に設定]**ブロックの間にはめ込み、値を**500**にします。**[基本]**：**[アイコンを表示_]**ブロックを選択し、最後の[でなければもし]の空欄にはめ込みます。アイコンは“かなしい顔”を設定します。



Step 15 **[音楽]**から**[メロディを開始する _くり返し_]**ブロックを選択し、プログラムに追加します。メロディは ”ワワワワー”に設定します。（または、好きなメロディを選択してください。）

下図が完成したプログラムです。

放送信号を送受信するには、EDU:BIT同士が同じ無線グループに属する必要があります。これは、0～255の間の任意の数を設定できます。

これらはイベントブロックです。実行されると、EDU:BITは、同じ無線グループに属するEDU:BITに対応するコマンド(1、2、3または4)を送信します。

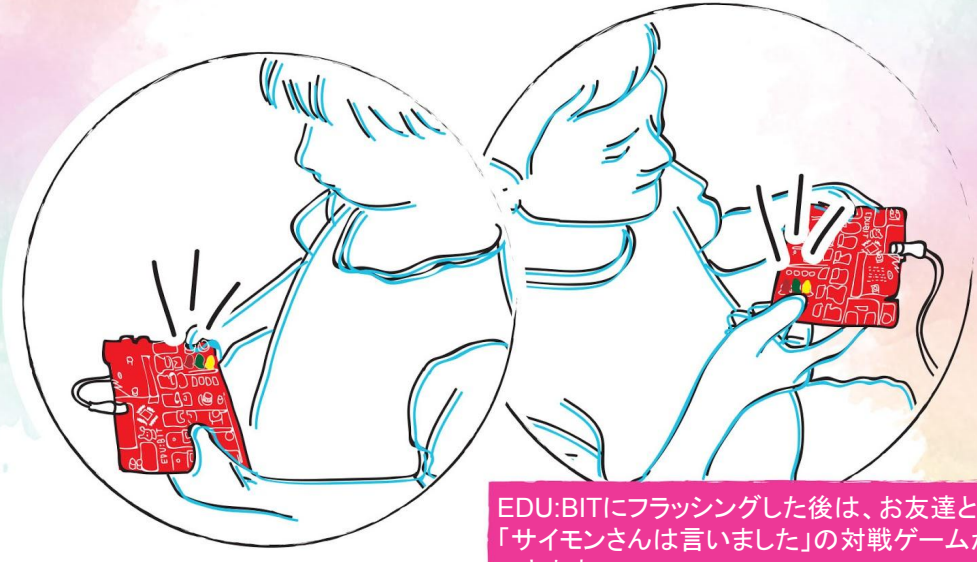
コマンドを受信すると、対応するプログラムを実行します。

Step 16 完成したプログラムを、お友達同士で EDU:BITにフラッシングしてください。

自分と友達のEDU:BITの電源入れると、友達の EDU:BITにラジオ信号を送ってLEDを点灯させることができるんだ。友達からラジオ信号を受け取って、自分の EDU:BITのLEDを点灯することもできるよ。

遊んでみよう！

LEDを使った、「サイモンさんは言いました。」



EDU:BITにフラッシングした後は、お友達と「サイモンさんは言いました」の対戦ゲームができます。

遊び方：

プレイヤーは交代で”サイモンさん”になります。自分の順番になったら、赤、黄色、緑のLEDを点灯させます。

- プレイヤー1がTraffic Light Bitを一つ点灯させてゲームスタートです。
- プレイヤー2は点灯したLEDを見て、自分のLEDを同じように点灯させます。
- 次はプレーヤー2が自分のTraffic Light Bitを点灯させます。プレーヤー 1はそれを見て、自分のLEDを同じように点灯させます。
- 対戦相手が点灯するLEDを間違えたら、自分の EDU:BITを傾けて、ゲームを終了させます。
- 間違えたプレイヤーは、リセットしてゲームを再スタートさせます。

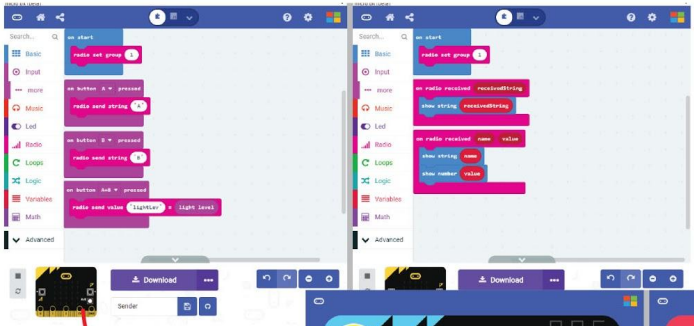
このゲームは最初は簡単ですが、ターンごとに難しくなります。LEDを注意深く観察する必要があります。記憶力を鍛える楽しいゲームです。

他のブロックも使ってみよう

#1 **[無線で文字列を送信"_"]**ブロックを使えば、数値だけでなく文字を送信できます。この場合、**[無線で受信したとき receivedString]**を使って文字を受信します。送受信できる最大文字数は19文字です。

#2 **[無線で送信 " "=_]**ブロックと**[無線で受信したとき name value]**ブロックを使えば文字と数字を同時に送受信できます。最大文字列は8文字です。

EDU:BITが2つ以上なくても、makecode/multiにアクセスすることで、ラジオ通信をテストすることができるよ。送信者と受信用のプログラムを作って、結果をシミュレーターで確認してみよう。



クリックすると全画面表示

送信者

受信者

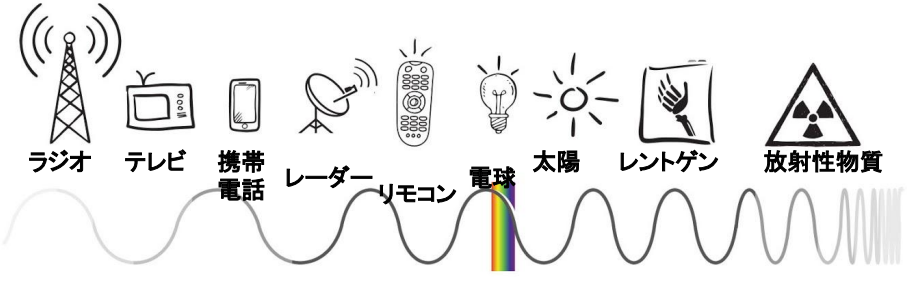
送信者がAボタンを押すと、ラジオ信号を受信した受信者は「A」という文字を表示するの。A+Bボタンを同時押しした場合は、どうなるか分かる？

面白い事実！

EDU:BITの裏にラジオとBluetooth通信機が内蔵されています。

アンテナは、信号を電磁波として送信するもので、テレビやラジオ放送、衛星放送などにもよく使われています。

電磁スペクトル



ノート！

EDU:BIT同士でラジオ信号を送受信するには、ラジオグループを同じに設定しておく必要があります。

練習問題

授業の答えを返すプログラムを作ってみましょう。

使い方

生徒が先生に文字列を送信すると、先生の EDU:BITにはメッセージが表示されます。数字を送信すると、先生の Traffic Light Bitが点灯します。その意味は、以下の通りです。

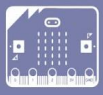
受信した数字	点灯するLED	意味
1	赤 	A / いいえ / 不正解
2	黄色 	B / 多分 / 分からない
3	緑 	C / はい / 正解

生徒は自分の名前を先生の EDU:BITに送信し、
番号（1，2，3）を送信してLEDを点灯させます。

Aボタンを押して1を送信。
Bボタンを押して2を送信。
A+Bボタンを同時に押して 3を送信。

ちょっとしたヒントだけど、生徒それぞれ2～3文字のニックネームを付けると、メッセージがスクロール表示される時間を減らすことができるわよ。

学んだこと



- メッセージとアニメーションをLEDに表示するプログラム。
- ダウンロード、保存、MakeCodeの.hexファイルの出力と編集方法

- イベントベースプログラミング。
- 変数の作り方と使い方。

- Misic Bitのピエゾブザーを使ってメロディを鳴らす方法。
- 関数の作り方と使い方。
- 楽譜の読み方。

- Traffic Light Bitの点灯、消灯、点滅。
- 拡張機能の使い方。

- IR Bitを使ってモノを感知するプログラム。
- ループを使ったプログラム。
- 配列の作り方と使い方。

- ポテンシオビットを使ったアナログ信号の受信プログラム。
- アナログ信号のマッピング。
- 論理ブロックを使った条件文プログラム。

- Sound Bitを使って音を感知するプログラム。
- イベントトリガーによってモードを切り替える方法。

- DC Motorの回転方法とスピード調整プログラム。
- 計算ブロックを使った算術演算。

- サーボモーターの使い方- 角度調整プログラム。

- RGB Bitの点灯方法と色、パターンの調整プログラム。

- ラジオ信号の送受信プログラム。

学習したところにチェックを入れ、そうでないところは、復習しましょう。

RERO EDUTEAM @ CYTRON より

よくできました！！！

これで全チャプタークリアです。MakeCode Editorでプログラムすることを学びました。子供に人気のあるゲームを選びましたが、楽しく遊んでいただけましたでしょうか。授業用の手軽なアプリケーションを作ることができたら、とても嬉しいです。

もう、micro:bitとEDU:BITの機能で何ができるのか、よく理解しているはずです。でも待って...実はそれぞれの機能を分解することもできるんです。知ってました？

余裕があれば「分解」してみてください。一度メインボードから外してしまえば、様々な機能を使って新しいプロジェクトを作ることができます。付属のプラグアンドプレイケーブルを使って接続する必要があります。

これからは、あなた自身で新しいゲームやアプリケーションを作る番です。どんな素晴らしいプロジェクトを思いつくのか、私達も楽しみにしてます。

是非、あなたの作った作品を私達と共有してください。メールやフェイスブック、あなたからのご連絡を心待ちにしております。

アダムとアンナより


Bit..
Bit....