＜目次＞

＜関連 Web サイト＞

小中学生のためのプログラミング教室

https://u-manabi.net/ild-pkouza/

＊学習ガイド関連の資料は、上記サイトの「プログラミングー学習ガイド」にある。

Ⅰ. micro:bit とタブレット端末

Ⅰ-1　micro:bit の特徴

micro:bit は、イギリス BBC（英国放送協会）が開発し、Micro:bit 教育財団が 7 年生（11～12 歳）の生徒を対象に無料配布した手のひらサイズの安価なコンピュータです。

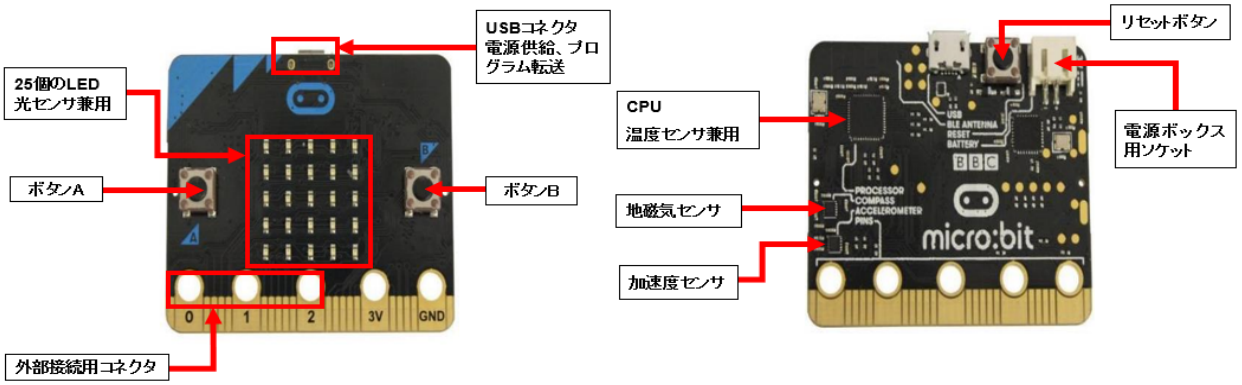


図 Ⅰ-1　micro:bit(実習で利用する micro:bit)

micro:bit のハードウエア機能としては、

・25 個の LED（表示、センサ)、光、温度、加速度計などのセンサ

・プログラムができるスイッチボタン（2 個）

・Bluetooth による無線通信

・物理的に接続するための端子

などがあります。さらに、以下のような特徴があります。

・ビジュアル言語で、簡単な操作で利用できる

・シミュレータがついている

・JavaScript、Python に自動変換できる

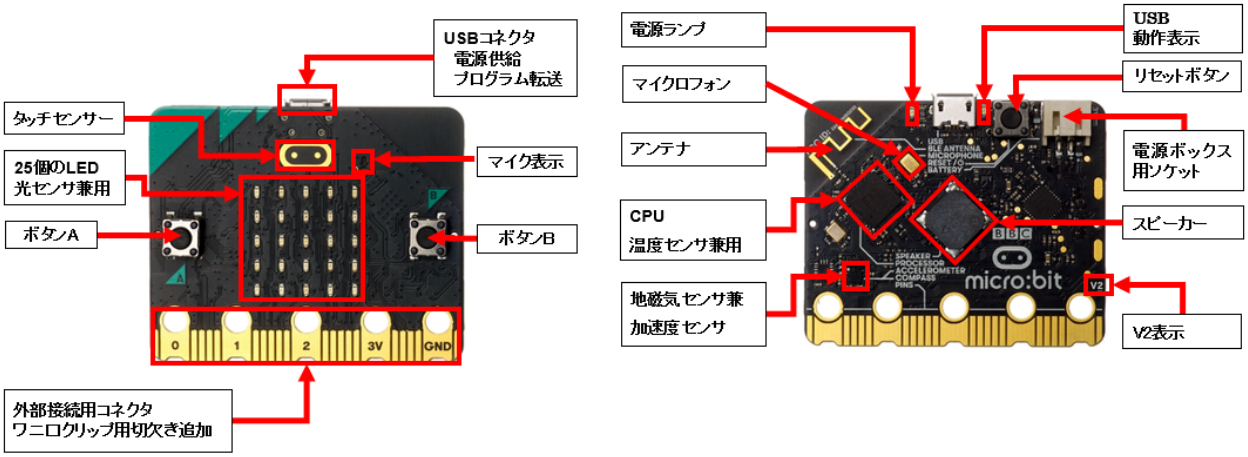


図 Ⅰ-2　micro:bit V2（2021 年 8 月以降、販売されているバージョン）

【参考資料】
https://microbit.org/ja/new-microbit/
https://tech.microbit.org/hardware/

I-2　iPad での micro:bit の利用

iPad の「App Store」から、micro:bit を検索して(図 I-3)、インストールします。一度、インストールされていれば、「開く」をクリックすると、micro:bit のアプリのメニューが表示されます（図 I-4)。micro:bit のアプリのメニューには、5 つの項目があり、ここでは、上の 3 つの項目を利用します。右上の「ヘルプ」をクリックすると、このメニューの説明や注意事項が書かれています。

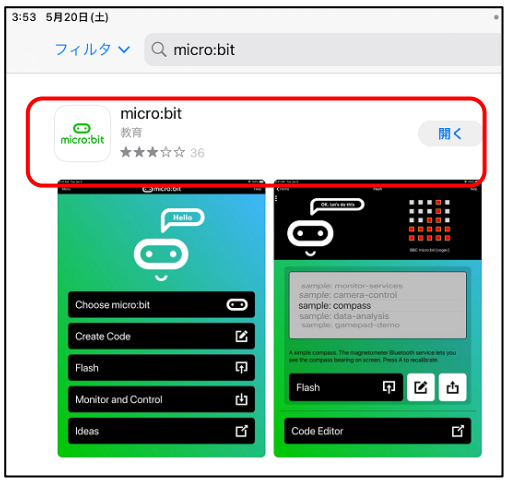


図 I-3　micro:bit のアプリ

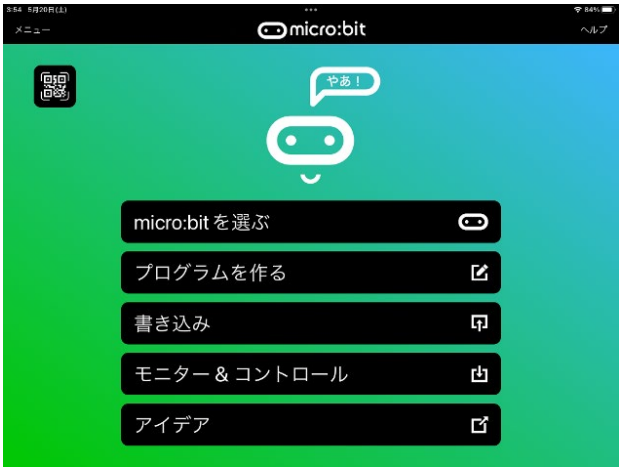


図 I-4　micro:bit のアプリのメニュー

次に、micro:bit を、iPad の Bluetooth 機能を利用して接続するために「ペアリング」を行います。

注）最初に、iPad の「設定」-「Bluetooth」で、オンになっていることを確認しておく。

図 I-4 の micro:bit のアプリのメニューから「micro:bit を選ぶ」をタップすると、現在、iPad に接続されている micro:bit があれば、名前（5 文字、パターン）が表示されます(図 I-5)。ここでは、新しい micro:bit を接続するので、図 I-5 の一番下のメニュー「新しい micro:bit をペアリング」をタップします。すると、図 I-6 のペアリングモードになります。

【ステップ I：ペアリングの準備】

micro:bit の A と B ボタン(表側)を押したまま、リセットボタン(裏側)を押します。一旦、micro:bit の LED(25 個)がすべて点灯し、そのあと「パターン」が表示されます。micro:bit に「パターン」が表示されたら、「次」をタップします。

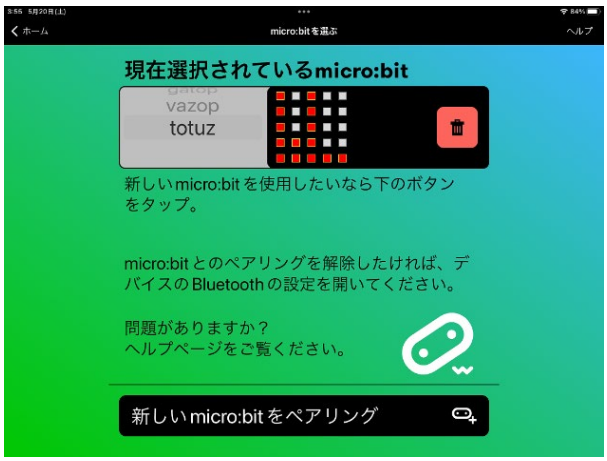


図 I-5　micro:bit のペアリング

図 I-6　ペアリングモード

【ステップ 2：パターンの入力】

次に、図 I-7 のようなパターン入力の画面が表示されれば、micro:bit の LED に表示されているパターンと同じように入力し(図 I-8 は入力済み)、 ペアリングの準備が完了します。A ボタンを押して、「次へ」でタップします。

注）LED の各列のパターンの一番上の□をクリックすれば、その下のパターンは、すべて入力されます。

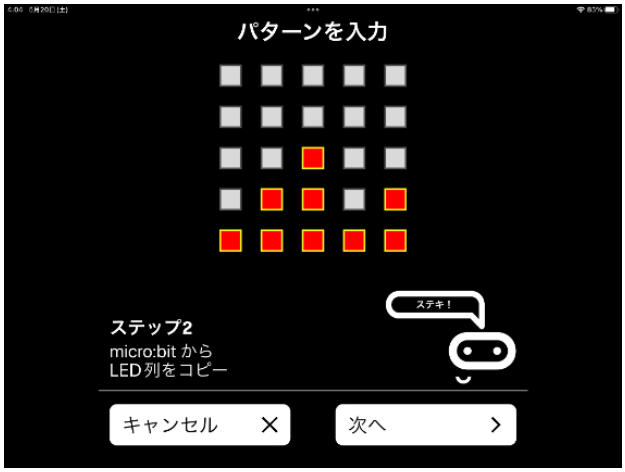


図 I-7　パターンの入力

図 I-8　ペアリングの準備完了

【ステップ 3：ペアリングの完了】

ペアリングに成功すれば、iPad は図 I-9 に示す画面になり、micro:bit にはチェックマーク（✓）が表示されます。ペアリングの完了操作として、micro:bit のリセットボタンを押してから「OK」をタップします。新しくペアリングされた micro:bit の名称（ここでは、「zoguv」）とパターンが、現在選択されている micro:bit として表示されます（図 I-10)。

図 I-9　ペアリングの完了

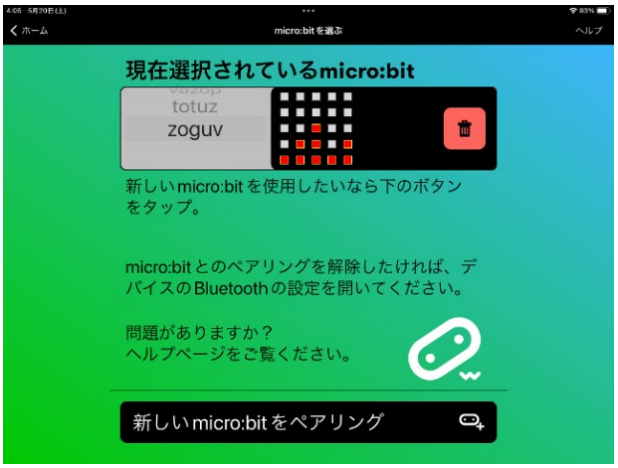


図 I-10　選択されたされた micro:bit

I-3　エディタによるプログラムの作成

図 I-4 の micro:bit のアプリのメニューから、「プログラムを作る」をタップすると、図 I-11 のような画面（ホーム）が表示されます。ここで、「新しいプロジェクト」をタップすると、「プロジェクトを作成する」ダイアログで、プロジェクト（プログラム）の名前（ここでは、prei1-1）をつけて、「作成」をタップします。すると、micro:bit 用のエディタ（MakeCode）やシミュレータの機能があるシミュレータ画面が表示されます(図 I-12)。

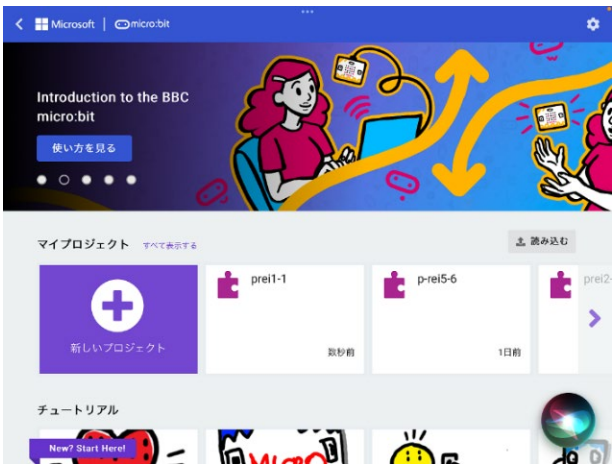


図 I-11　新しいプロジェクト

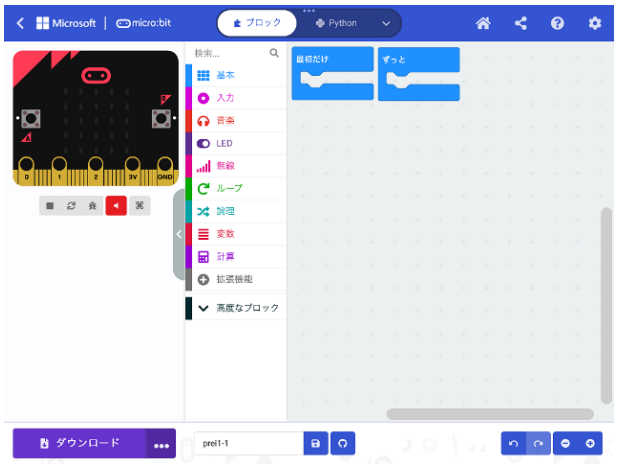


図 I-12　シミュレータ画面

図 I-13　シミュレータ画面の名称

注）iPad の画面の向きを、横から縦にすると、シミュレータ画面の表示位置が変わる。

新しいプロジェクトを作成すると、プログラミングエリアには、「最初だけ」ブロックと「ずっと」ブロックが、最初に置かれています。図 I-13 では、不要なブロックである「ずっと」ブロックは、ツールボックスへ、ドラッグ＆ドロップして削除しています。

そして、ツールボックスの「基本」をタップし、「LED 画面に表示」ブロックを、ドラッグ＆ドロップで、プログラミングエリアに移動し、「最初だけ」ブロックにつなげます。LED をタップ（光の ON/OFF が切り替わる）して、ハート形に見えるように LED を ON にしています。

このように、図 I-13 は、「最初だけ」ブロックを利用して、ハートを表示しているものです（プログラム　prei1-1)。「ずっと」ブロックを利用して、ツールボックスから「基本」-「一時停止」や「表示を消す」を選択して、ハートの点滅も確認しておきましょう（プログラム prei1-2)。

Windows からは、下記の Web サイトにアクセスすれば、「MakeCode エディタ」のホーム（図 I-11 に同じ）画面が表示されます。

https://makecode.microbit.org/

シミュレータ画面の名称は、図 I-13 に示す通りで、それぞれの概要は、以下の通りです。

［ツールボックス］

基本、入力、音楽、LED、無線、ループ、論理、変数、計算、そして、高度なブロックがあり、それぞれのツールをクリックすると、利用できるブロックが表示される。

[プログラミングエリア]

ツールボックスで選択したブロックをエリア内にドロップすることによって、プログラムが作成できる。「最初だけ」「ずっと」のブロックが、最初に置かれている。

[ホーム]

「ホーム」を選択すると、新しいプロジェクトの場合には、名前を付けることができる。最初に名前をつけていれば、最初の画面に戻る。

注）最初に、名前をつけていない場合は、「題名未設定」となる。

[ブロック]

「ブロック」を「JavaScript」や「Python」に切り替えることによって、「ブロック」で書かれたプログラムをそれぞれの言語で表示することができる。

［ダウンロード］

「ダウンロード」では、プログラムを micro:bit に書き込み(ダウンロード)することができる。また、「…」を開くと、ダウンロードのオプションを指定できる。「FD のアイコン」では、プロジェクト名のついたプログラムをファイルとして、指定した場所に保存することができる。

[シミュレータ]

micro:bit の画面の下には、プログラムを四角ボタン（■）で停止、三角ボタン（▶）で開始できる。そのほか、再起動、デバッグモードの切り替えなどができる。

I-4　プログラムの micro:bit へダウンロード

図 I-4 の micro:bit のアプリのメニューから「書き込み」をタップすると、図 I-14 の画面が表示されます。なお、図 I-13 の画面で、「保存」（FD のアイコン）をタップしても、図 I-14 の画面が表示されます。

注）書き込み先の micro:bit は、あらかじめペアリングモード（A+B を押した状態で、micro:bit のリセットボタンを押す）にしておきます。micro:bit へダウンロードする際には、毎回行います。

次に、プログラム名（ここでは、prei1-1）を選択して、「書き込み」をタップします(図 I-14)。書き込みを行う micro:bit（ここでは、「zoguv」）を検索し(図 I-15)、見つからない場合は、ペアリングモードのリセットを行います(図 I-16)。

「続行」を選択すると、「通信中・・・」と表示され、少ししてから書き込みが終われば、図 I-17 の終了画面が表示されます。

図 I-14　書き込みの画面

図 I-15　micr:bit の検索

図 I-16　ペアリングモード

図 I-17　書き込みの終了

I-5　外部ファイルの読み込み

ホーム画面の右にある「読み込む」(図 I-18 の赤での囲み)を選択すると、図 I-18 に示すようなダイアログが表示されるので、左の「ファイルを読み込む・・・」の箇所をタップすると、図 I-19(a)に示すように、ファイルを選択するダイアログが表示されます。図 I-19(a)の「ファイルを選択」の箇所をタップすると図 I-19(b)が表示されるので、上から 3 つ目の「ファイルを選択」の箇所をアップします。

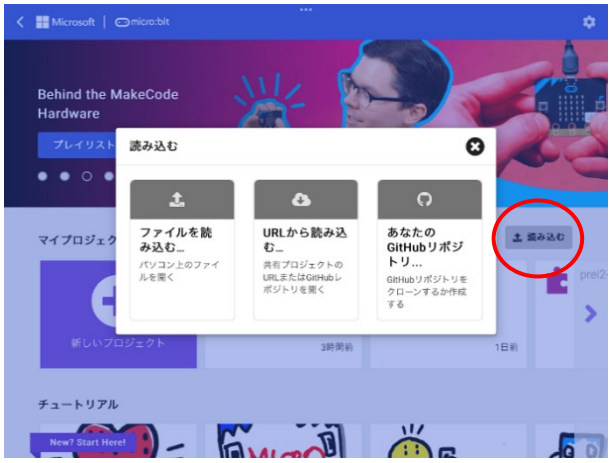


図 I-18　ファイルの読み込み

(a)

(b)

図 I-19　ファイルの選択

次に、ファイルが書き込まれている場所と読み込むファイルを指定します。図 I-20 は、左のメニューにある「iCloud Drive」「この iPad 内」「Google ドライブ」「OneDrive」の中から、「OneDrive」をタップしたときの画面です。この「OneDrive」の例では「ファイル」「ライブラリ」「共有アイテム」の 3 つのフォルダがあり、「ファイル」フォルダをタップすると、保存されたプログラムのファイル名が表示されています(図 I-21)。この中から、読み込みたいファイルをタップすると、図 I-19 (a) に、ファイル名が表示されます。

注）ファイルが書き込まれている場所（ここでは、OneDrive）が読み込める状態になっているものとしています。

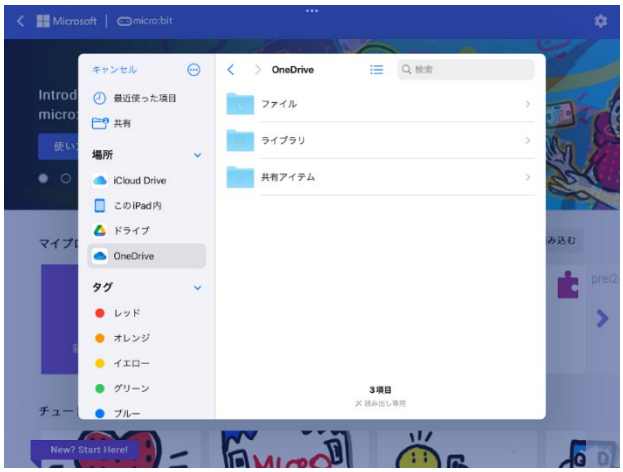


図 I-20　ファイルの格納場所

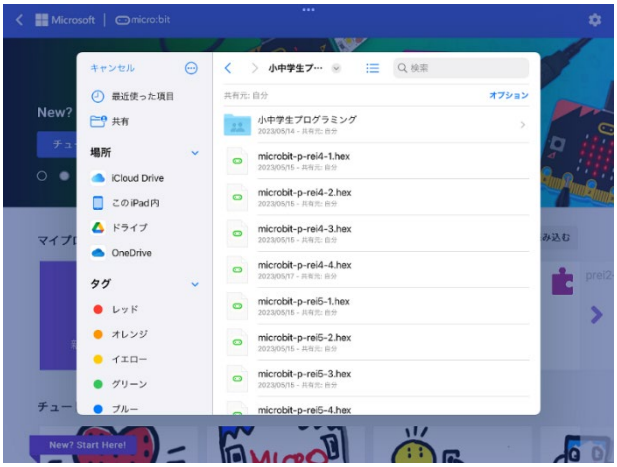


図 I-21　保存されているプログラムファイル

２．プログラムの基本と応用

2-1　プログラムの基本（順次構造）

【例題 2-1】（プログラム prei2-1）

３つのアイコン（ハート、小さいダイアモンド、しかく）を表示してみよう。

「ずっと」ブロック

＊「基本」－「アイコン表示」（ハート）、「基本」－「一時停止」

＊「基本」－「アイコン表示」（小さダイアモンド）、「基本」－「一時停止」

＊「基本」－「アイコン表示」（しかく）、「基本」－「一時停止」、「基本」－「表示を消す」

注）「最初だけ」「ずっと」ブロック、「基本」などは、テーマ１の「エディタによるプログラム作成」を参照

2-2　プログラムの基本（反復構造）

【例題 2-2】（プログラム prei2-2）

アイコンを３つ（小さいダイアモンド、はさみ、しかく）を繰り返し５回表示してみよう

「最初だけ」ブロック

＊「基本」－「アイコン表示」の「はさみ」を選択する

＊「ループ」－「くりかえし」　数値０を５にする

じゃんけんの「グー」「チョキ」「パー」をこの３つのアイコンで表示する。

【例題 2-3】（プログラム prei2-3）

「変数　カウンター」を利用して、表示してみよう。

「最初だけ」ブロック

＊「ループ」の「変数 カウンター ～　くりかえす」

数値 0 を 4 にする

注）変数「カウンター」は、「0 から始まり、

「0，1，2，3，4」の 5 回繰り返す。

注）変数は数値や文字を入れるための領域

反復構造は、繰り返し構造ともいう。

2-3　プログラムの基本（分岐構造）

【例題 2-4】（プログラム prei2-4）

乱数を利用して、変数 c に 0 から 1 の数値を入れる。c＝0 のときは、「グー」を表示し、そうでないとき（c=1）は、「パー」を表示してみよう。

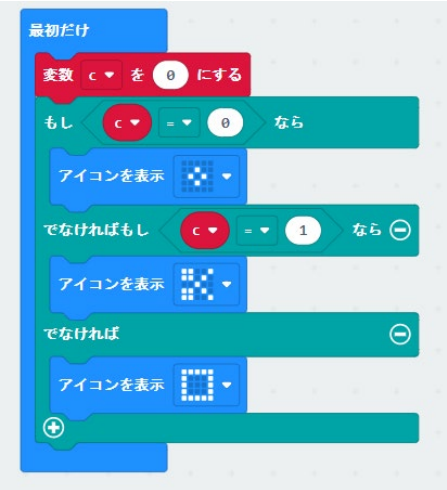
「ずっと」ブロック

＊「変数」-「変数を追加する」　変数 c にする

＊「計算」-「0～10 までの乱数」　数値 10 を 1 にする（乱数は 0、1 のいずれか）

【例題 2-5】（プログラム prei2-5）

「グー」「チョキ」「パー」を表示するプログラムを作成し、変数 c の値を、3 つの分岐を確かめてみよう。



「最初だけ」ブロック

＊「変数」-「変数を追加する」　変数 c にする

＊「変数」-「変数 c を 0 にする」

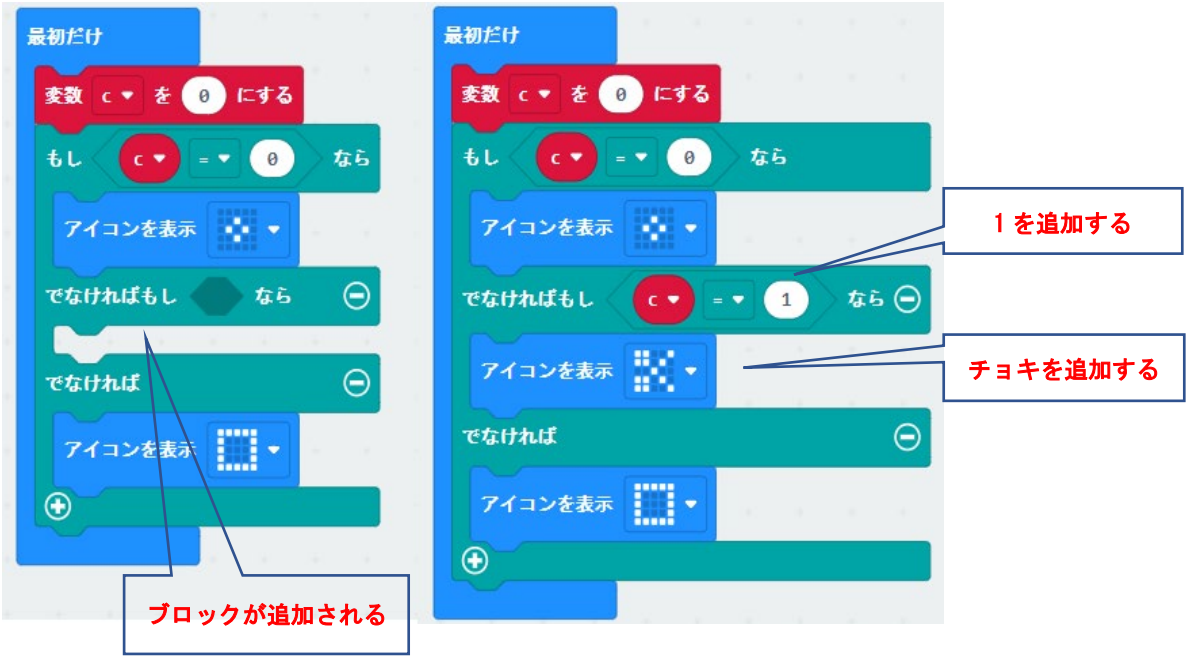
＊「論理」-「条件判断」（もし なら～　でなければ～）で、

「⊕」をクリックするとブロックが追加される

＊「でなければもし　なら」の箇所、

「 c　＝▼　1 」　を追加する

「チョキ」を追加する

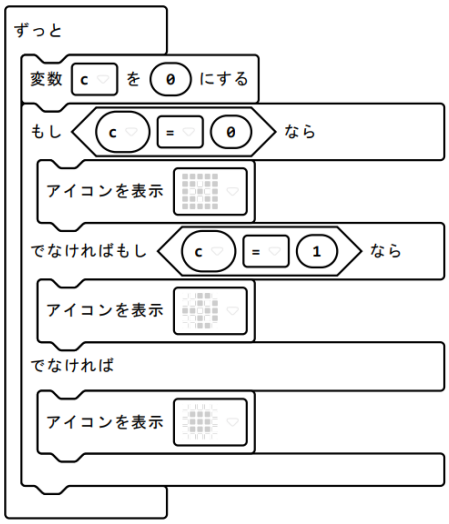


分岐構造は、選択構造ともいう。
順次構造、反復構造(繰り返し構造)、分岐構造(選択構造)
をプログラムの基本構造という。

＜プログラムの印刷＞

エディタの画面(右上端)にある歯車（⚙）から「印刷」を選択する。少し経過して「プログラムを印刷する」の画面が表示され、ブロック（もしくは、右図のように修正された形）のプログラムを印刷できる。

注）付録Ⅰの「入力用拡張ブロック」を利用しているプログラムは印刷されない。



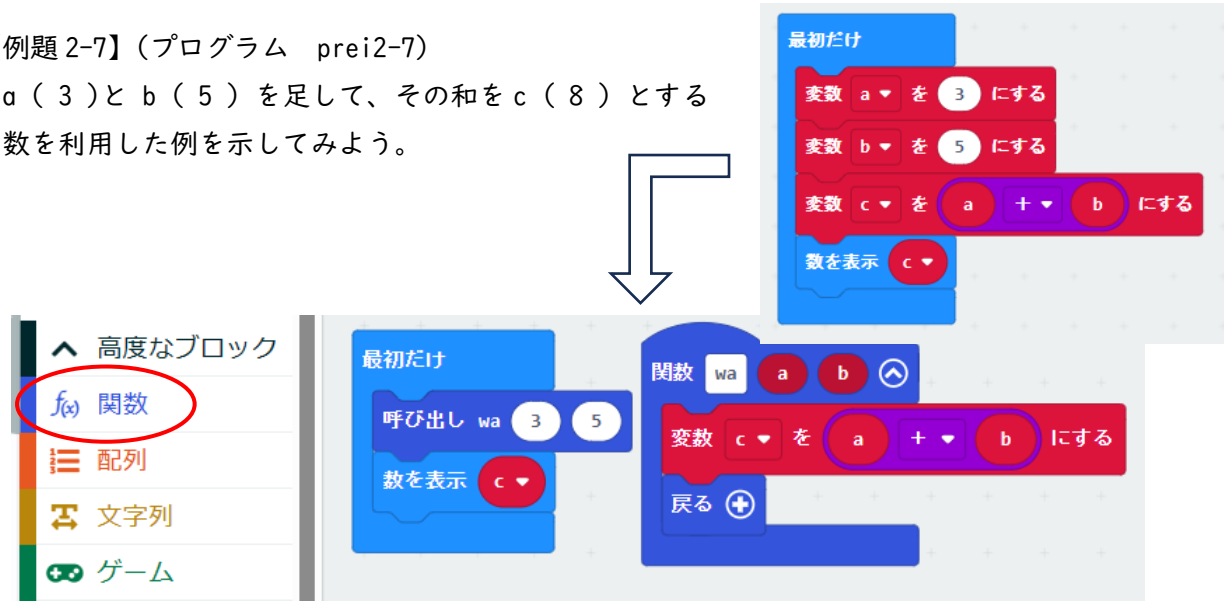
注）付録Ⅰの「入力用拡張部ブロック」を利用しているプログラムは印刷されない。

2-4　プログラムの応用（関数）

　関数は、一定の処理をまとめたもので、プログラムの中で必要に応じて呼び出される。関数には、プログラミング言語に初めから用意されている組み込み関数(例えば、乱数)とプログラムの作成者が用意するユーザ定義関数がある。ここでは、後者のユーザ定義関数を関数という。

【例題 2-7】（プログラム　prei2-7）

　a（ 3 ）と b（ 5 ）を足して、その和を c（ 8 ）とする関数を利用した例を示してみよう。



＜関数の作成手順＞

＊micro:bit のツールボックスの「高度なブロック」－「関数」を選択する

＊「関数」－「関数を作成する…」が表示される

＊「関数を作成する」を選択すると、「関数の編集」ダイアログが表示される

＊関数の名称（doSomething の箇所）を記入する

＊引数があれば、パラメーターを追加する(例えば、数値)を選択して記入する

＜関数の引数と戻り値＞

　micro:bit では、関数(wa)は、上に示した図のようなブロックでかける。

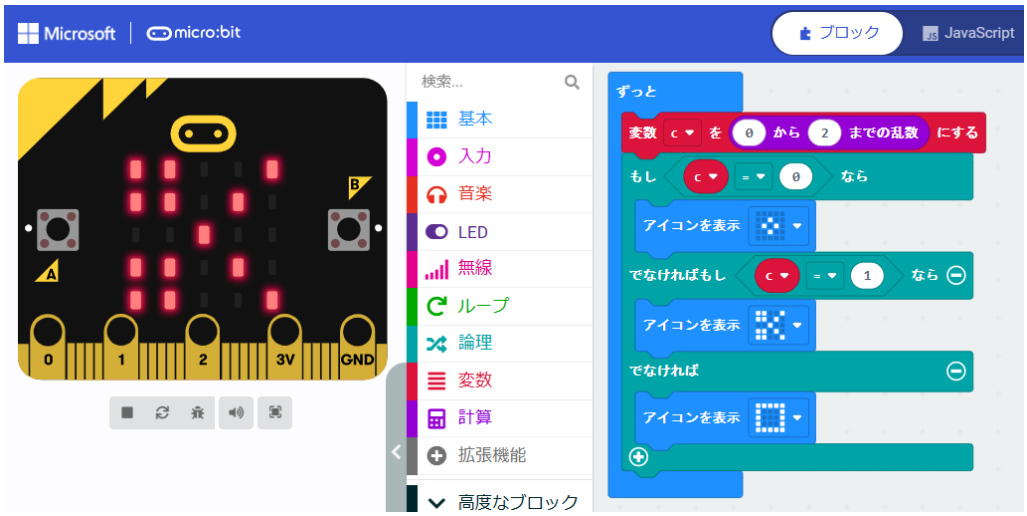
「関数」ブロックの「wa」を関数名、変数 a、b は関数に引き渡されるパラメータ（この例では数値）で引数という。micro:bit では、関数(wa)は、上に示した図のようなブロックでかける。この例では、「戻る」に値がなく変数 c の値として返しているので、「戻る」のブロックはなくてもよい。関数側で計算した結果の値を「戻る」で返す場合は、必要になる。

4．コンピュータとじゃんけん

4-1　「グー」「チョキ」「パー」の表示

【例題 4-1】（プログラム prei4-1）

0～2 の乱数を利用して、アイコンで「グー」（数値 0）、「チョキ」（数値 1）、「パー」（数値 2）を表示してみよう（例題 2-5 参照）。



「ずっと」ブロック

＊「変数」－「変数を追加する」　変数 c にする

＊「計算」－「0～10 までの乱数」　数値 10 を 2 にする（乱数は 0、1、2 のいずれか）

4-2　コンピュータの手

【例題 4-2】（プログラム prei4-2）

コンピュータの手を 5 回繰り返して、表示してみよう。

「最初だけ」ブロック

＊「基本」－「アイコン表示」で、「グー」を表示する

＊「カウンター」（0～4）で、5 回繰り返す

＊「基本」－「数を表示」で、カウンターの数値（回数）を表示する

＊自分の番のときは、「矢印を表示」で「右向き」（→）」を表示して、2 秒待つ

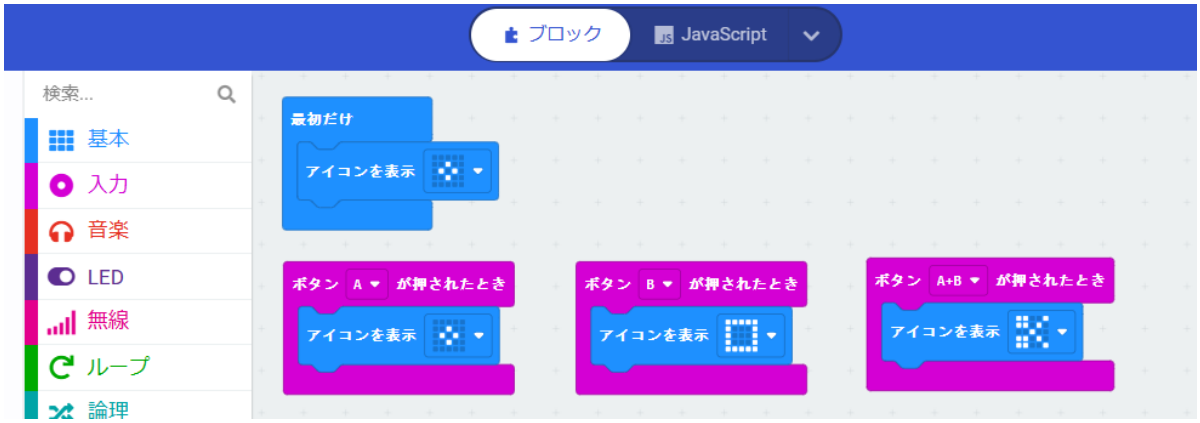
＊コンピュータのときは、「文字列を表示」で「Com」と表示する

＊例題 4-1 のプログラムを加え、「グー」か「チョキ」か「パー」をランダムに表示する

4-3　自分の手

【例題 4-3】（プログラム prei4-3）

スイッチボタンを利用して、自分の手をアイコンで表示してみよう。ただし、A ボタンは「グー」、B ボタンは「パー」、A+B ボタンは「チョキ」とする。



「最初だけ」ブロック

＊「基本」－「アイコンを表示」で「グー」を表示する

＊「入力」－「ボタン A が押されたとき」を選択する

A ボタンでは「グー」、B ボタンでは「パー」、A+B ボタンでは「チョキ」とする。

4-4　「グー」「チョキ」「パー」を出す回数（コンピュータ）

【例題 4-4】（プログラム prei4-4）

コンピュータの手を 100 回実施したとき、「グー」「チョキ」「パー」の出す回数を調べるプログラムを考えてみよう。

実際は、乱数の出す数値(0,1,2)の回数を調べるプログラムである。

例題 4-1 のプログラムで、「グー」「チョキ」「パー」を表示している個所を変数 k0、k1、k2 に、カウントしている。

＜発展＞プログラミング言語

micro:bit では、プログラミング言語へ自動変換できる。下のプログラムは、プログラミング言語「Python」を選択し、ブロックで作成したプログラムを表示したものである（数を表示、文字列を表示の箇所は除く）。ブロックで、c、k0、k1、k2 の変数の初期値を「0」にしていない場合でも、自動的に「0」が設定されている。

「Python」で、プログラムを記述する場合は、つぎの手順で行う。

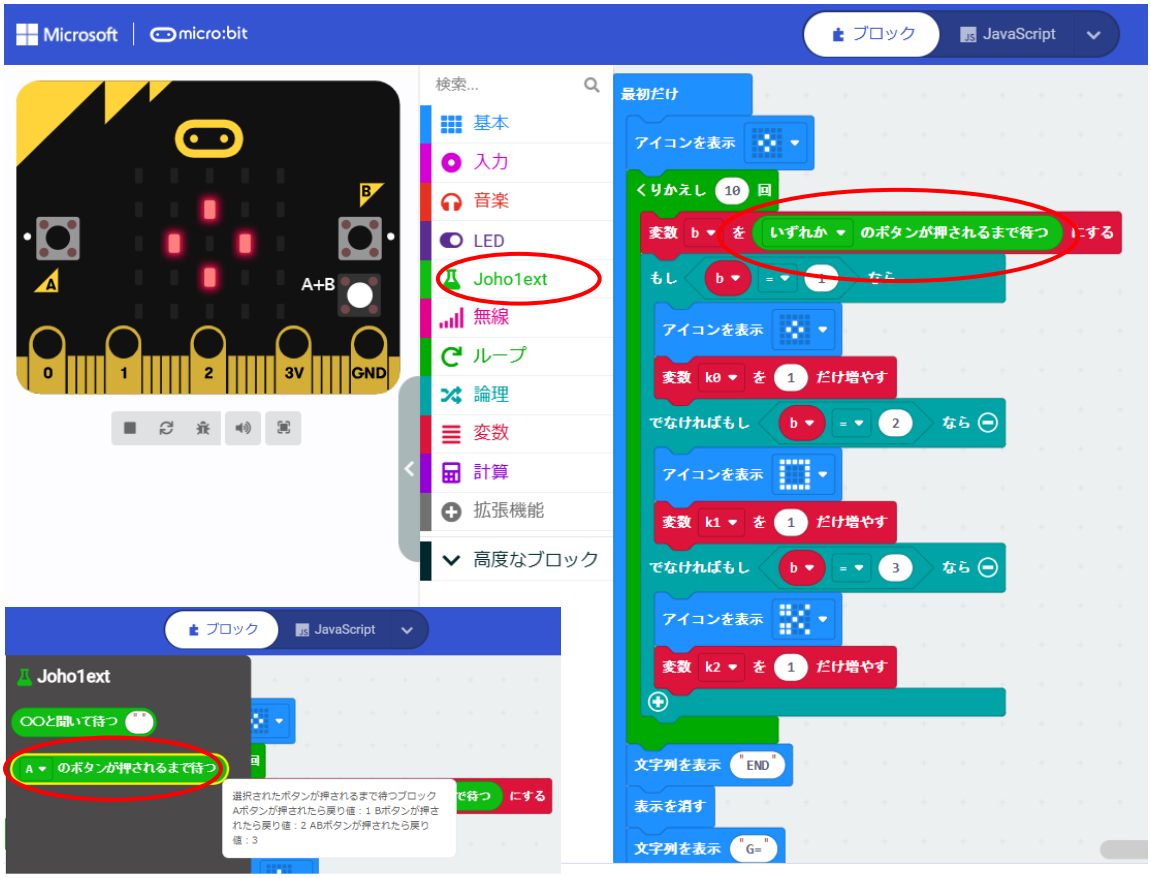
・新しいプロジェクトを作成し、「Python」を選択する
・この Python のプログラムの 12 行目までを入力する。
　注）行の番号は不要である。空白は、「Tab」であける。
・エラーがでれば修正し、エラーがでなくれば、「ブロック」で表示する
・「ブロック」で、「回数」の表示結果を見やすいように作成する。

```
1   c = 0
2   k0 = 0
3   k1 = 0
4   k2 = 0
5   for カウンター in range(100):
6       c = randint(0, 2)
7       if c == 0:
8           k0 += 1
9       elif c == 1:
10          k1 += 1
11      else:
12          k2 += 1
```

4-5　「グー」「チョキ」「パー」の出す回数(自分)

【例題 4-5】（プログラム prei4-5）

自分の手を 10 回実施したとき、「グー」「チョキ」「パー」の出す回数を調べるプログラムを作成してみよう。入力用拡張ブロック（「いずれかの　ボタンが押されるまで待つ」）（付録 1）を利用する。



「最初だけ」ブロック
＊最初、「グー」を表示する
＊繰り返し 10 回のループ内は、4-4 のプログラムと同じように、「グー」の回数の変数 k0、「チョキ」の回数の変数 k1、「パー」の回数の変数 k2 としている
＊回数の表示は、4-4 のプログラムと同じである

＜入力用拡張ブロック（いずれかのボタンが押されたまで待つ）＞
戻り値は、１、２、３に設定されており、このプログラムでの「グー」「チョキ」「パー」との対応は、下記の通りである。
A ボタンが押されたときの戻り値　　：1（グー）
B ボタンが押されたときの戻り値　　：2（パー）
A＋B ボタンが押されたときの戻り値　：3（チョキ）

4-6　じゃんけんの自動判定（コンピュータとの対戦）
【例題 4-6】（プログラム　prei4-6）
じゃんけんの自動判定（コンピュータとの対戦）するプログラムを作成してみよう。
例題 4-2 を参考にしてコンピュータの出す手の関数(Com)、例題 4-5 を参考にして自分の出す手の関数(You)を作成する。また、関数　判定を作成し、勝ったときは「うれしい顔」、負けたときは「悲しい顔」、引き分け（あいこ）のときは「困った顔」のアイコンで表示する。

＜勝敗の判定表＞
A、B の 2 人のじゃんけんの勝敗は、下表に示す通りで、「a-b+3」を 3 で割った余りでの値で判断する。例題 4-5 の A（You）、B（Com）の場合

引き分け：0　　A(You)の勝ち：2　　B(Com)の勝ち：1

種類	数値	A	B	判定	（A-B）の値	（A-B+3）/ 3 の余り
グー	0	0	0	引分け	0	0
		0	1	A	-1	2
		0	2	B	-2	1
チョキ	1	1	0	B	1	1
		1	1	引分け	0	0
		1	2	A	-1	2
パー	2	2	0	A	2	2
		2	1	B	1	1
		2	2	引分け	0	0

＜参考文献＞
高橋、喜家村、稲川：micro:bit で学ぶプログラミング、pp.14-15、pp.41-43、コロナ社(2019)

関数は、以下のように作成する(例題 2-7 参照)。

＊micro:bit のツールボックスの「高度なブロック」－「関数」を選択する

＊「関数」－「関数を作成する…」を選択すると、「関数の編集」ダイアログが表示される

＊「関数の編集」で、まず、引数のない関数 2 つの関数「Com」と「You」を作成する。

「関数　Com」

例題 4-2 のプログラムの後半部分と同じである。

「関数　You」

例題 4-5 のプログラムの後半部分と同じである。このプログラムでは、「入力用拡張ブロック（いずれかのボタンが押されたまで待つ)」を利用している。

「ずっと」ブロック

＊「You」と表示し、「呼び出し　You」で、関数を呼び出す

＊「Com」と表示し、「呼び出し　Com」で、関数を呼び出す

＊「呼び出し　判定」で、関数を呼び出す

「関数　判定」

　判定式の余りを求める式の箇所は、以下のように作成する。

＊「計算」－「remainder of 0 / 1」を選択し、後ろの「1」を 3 にする

＊「計算」の「0 - 0」、「変数」の a と b から、「a － b」を作成する

＊さらに、「計算」の「0 + 0」から「0 ＋ 3」を作成し、「0」に「a － b」を重ねる

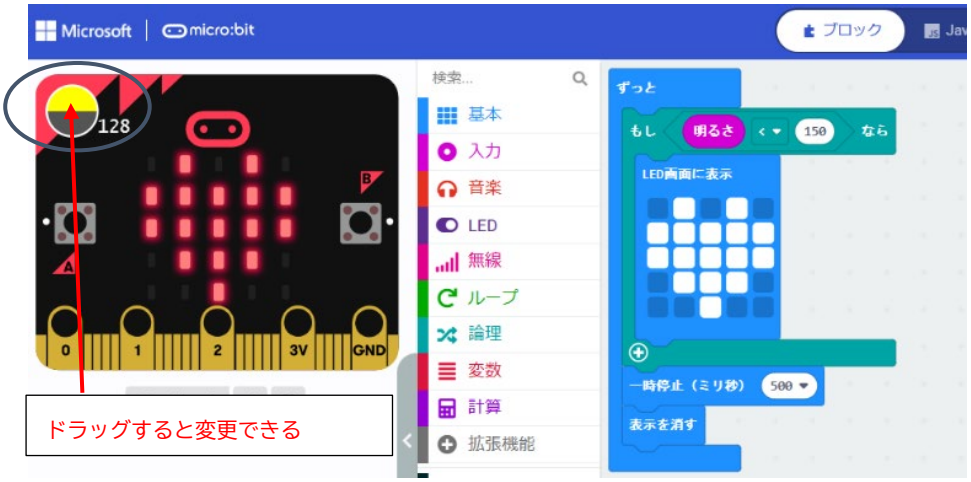
＊作成した「a － b + 3」を「remainder of 0 / 3」の「0」に重ねる

６．カラーLED の点灯と制御

6-1　光センサによる LED の点灯

【例題 6-1】（プログラム prei6-1）

光センサで明るさを調べて、暗ければ、アイコンの「ハート」を点滅してみよう。



「ずっと」ブロック

＊「論理」－「条件判断」で、「もし　なら～」を選択する

＊「論理」－「くらべる」で、「0 < = 0」　を選択する

＊「入力」－「明るさ」を重ねる。数値 0 を 150 にする　シミュレータの明るさの数値は 128

【例題 6-2】（プログラム prei6-2）

光センサで明るさを調べて、暗ければ、LED を点灯してみよう。

「ずっと」ブロック

＊「論理」－「条件判断」で、「もし　なら～」を選択する

＊「論理」－「くらべる」で、「0 < = 0」を選択する

＊「入力」－「明るさ」を重ねる。数値 0 を 125 にする

＊「高度なブロック」で「入出力端子」選択し、

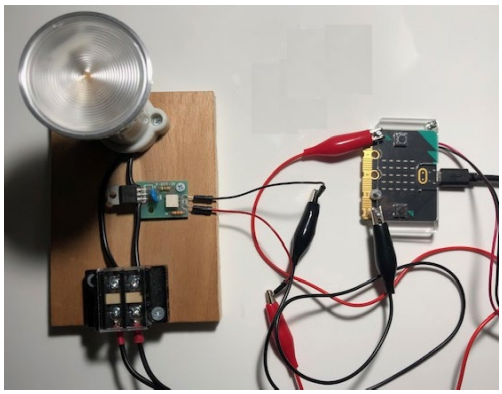
「デジタルで出力する 端子 P0 ▼ 値」を選択、数値は、1 と 0 にする

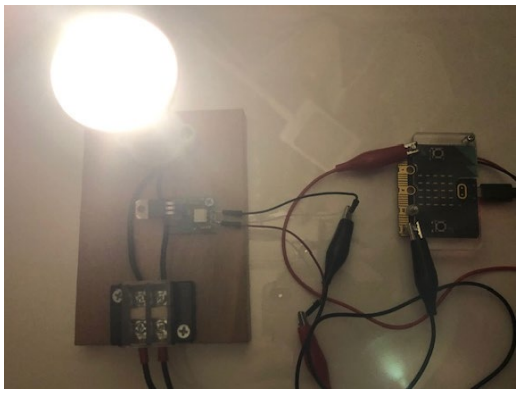
注)図のシミュレータでは、明るさの数値が 75 で、デジタル端子 P0 の出力は 1 になっている

＜100V　LED 電球の制御＞

例題 6-2 に示したように、micro:bit から LED の点灯を行うことがきる。さらに、100V の LED 電球も micro:bit で点灯を行えるが、そのためには、制御するためのリレー回路（SSR：Solid State Relay）が必要である。下の写真は、例題 6-2 のプログラムに、100V の LED 電球(回路自作)をつないだものである。

左は部屋の蛍光灯が ON（LED 電球：OFF)、右は部屋の蛍光灯が OFF（LED 電球：ON）の状態である。





6-2　スイッチボタンによる LED の点灯

【例題 6-3】（プログラム prei6-3）

A ボタンを押すと、LED が点灯し、もう 1 回押すと、消灯するプログラムを作成してみよう。

「最初だけ」ブロック

＊「変数」－「変数を追加」で、変数は、s にする

＊「変数」－「変数 s を 0 にする」

「ボタン A」

＊「論理」－「条件判断」で、「もし　なら～」を選択し、「s =▼ 0」を重ねる

＊「もし」ブロックでは、

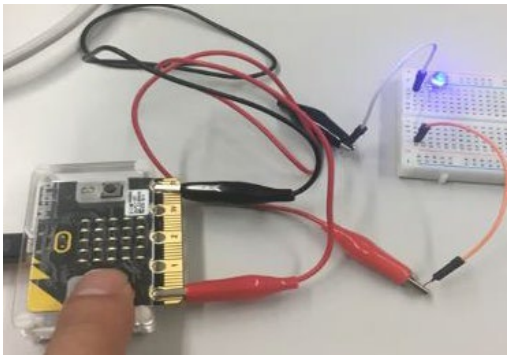
「デジタルで出力する 端子 P0 ▼ 値」の数値は 1

「変数」－「変数 s を 0 にする」で、数値は 1 にする

＊「なら～」ブロックでは、

「デジタルで出力する 端子 P0 ▼ 値」の数値は 0

「変数」－「変数 s を 0 にする」で、数値は 0 にする



＜参考文献＞

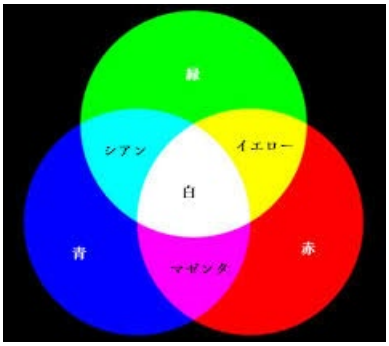
高橋、喜家村、稲川：micro:bit で学ぶプログラミング、pp.35-36、p.38、コロナ社(2019)

6-3　フルカラーLED

【Neopixel】

Neopixel は、１つのセルごとに赤（R)、緑（G)、青（B）の３つの LED と制御回路が入っており、フルカラーで光らせることができる LED の集合体である。

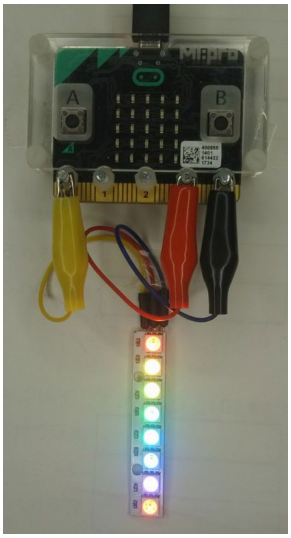
フルカラーは、色の明るさを、赤(R)、緑(G)、青(B)が、それぞれ 0～255 の 256 段階で選べるので、256*256*256=16,777,216 色の表現が可能となる。なお、R:255、G:255、B:255 では白、R:0、G:0、B:0 では黒になる。

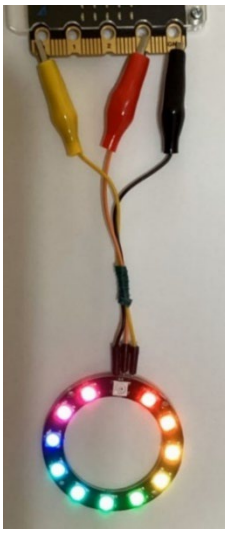


光の３原色　R(赤) G(緑) B(青)

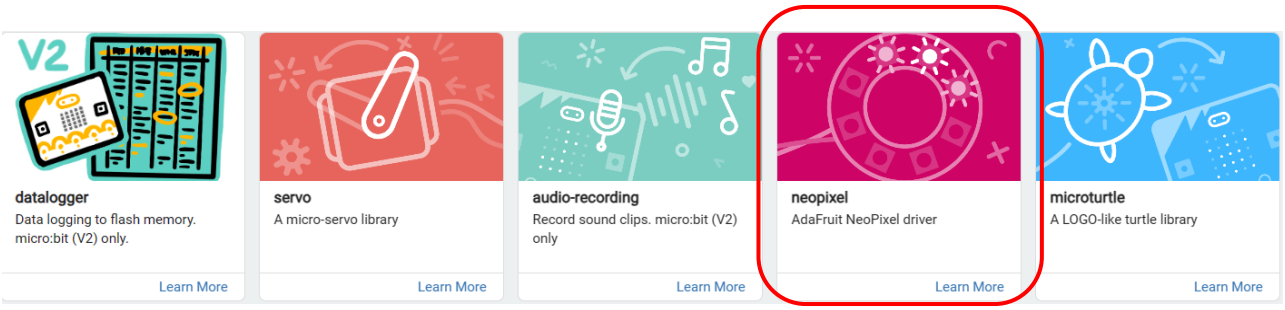
【micro:bit と Neopixel の接続】

micro:bit と Neopixel を右図のように接続する（黄色は P0 端子、赤は 3V 端子、黒は GND 端子)。右図の Neopixel は、８個の LED が棒状（Stick 型）に接続された製品で、その他にも、Neopixel には、リング状の製品がある。





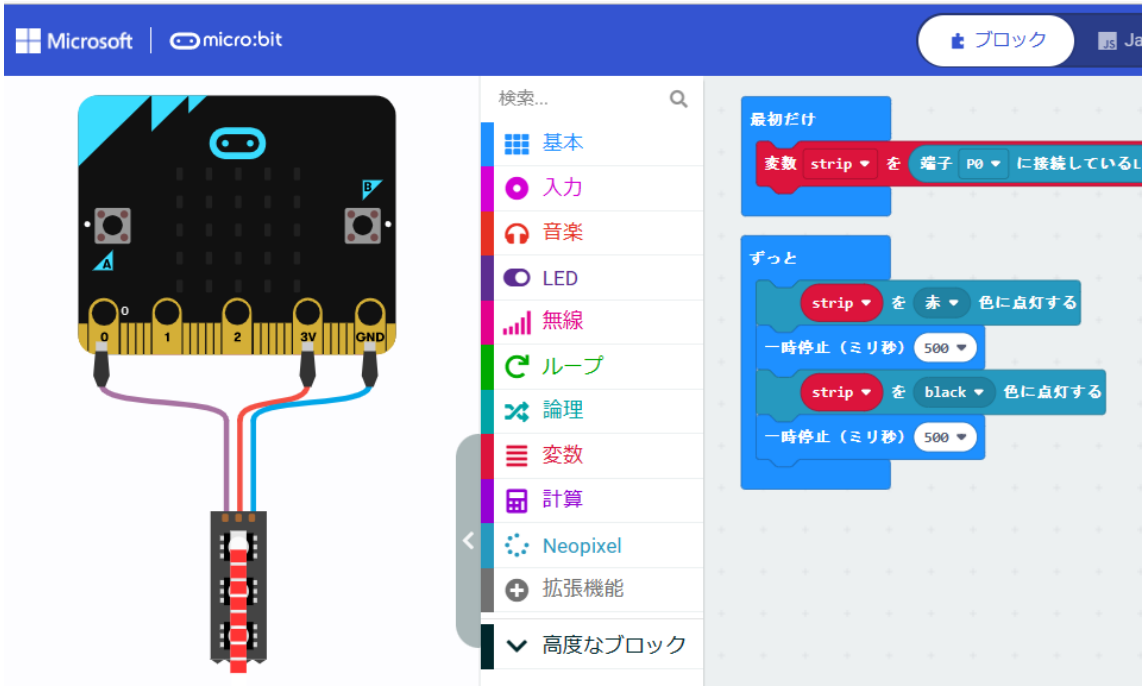
Neopixel を利用するには、ライブラリが必要である。ツールボックスの下にある「拡張機能」をクリックすると、拡張機能の一覧が表示されるので、「Neopixel」を選択する。ツールボックスの「計算」の下に、「Neopixel」のブロックが追加される。



6-4　Neopixel の点滅

【例題 6-4】（プログラム　prei6-4）

Neopixel(Stick 型)を赤色で点滅してみよう。



「最初だけ」ブロック

＊「Neopixel」－「変数 strip を 端子 P0 に接続している LED　個…　にする」

　ここで、LED 24 個→　8 個」に変更しておく

「ずっと」ブロック

＊「Neopixel」－「strip　赤色に点灯する」　色は、赤色のままにしておく

＊「Neopixel」－「strip　black 色に点灯する」　black では、消灯になる

【例題 6-5】（プログラム prei6-5）

「Neopixel」で、「RGB（赤　緑　青)」色に点灯する」に変更し、フルカラーで表現してみよう。

右図では、シアン（水色に近い青緑色）になる。

【例題 6-6】（プログラム prei6-6）

Neopixel を 8 色で点灯させてみよう。

「最初だけ」ブロック

例題 6-4 に同じ（図は、省略）

「ずっと」ブロック

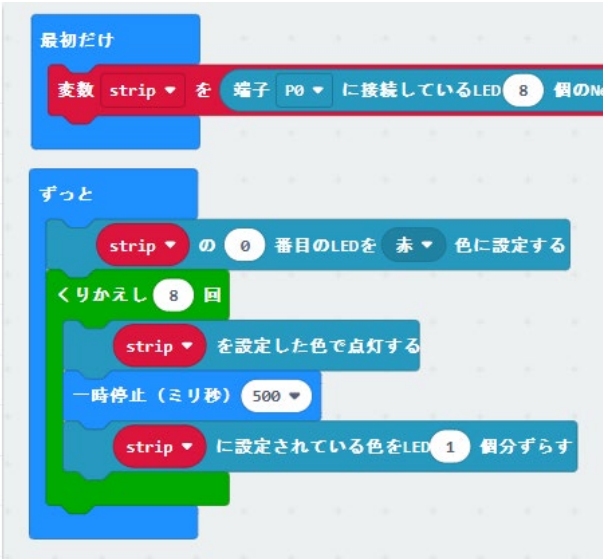
＊「Neopixel」－「strip の 0 番目　色に設定する」　0～7 番目を図の色（赤・・・紫）にする

＊「Neopixel」－「strip で設定した色で点灯する」

6-5　Neopixel の点灯（色の上下移動）

【例題 6-7】（プログラム prei6-7）

Neopixel を 8 色で点灯させ、上下移動してみよう。



「最初だけ」ブロック

例題 6-4 に同じ（図は、省略）

「ずっと」ブロック

＊「Neopixel」－「strip の 0 番目 赤 色に設定する」

＊「ループ」－「くりかえし 8 回」にする

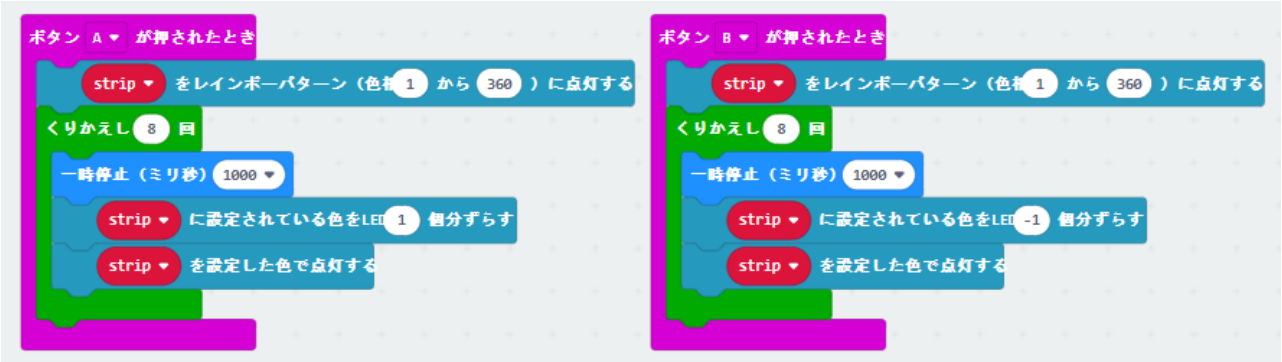
＊「Neopixel」－「strip で設定した色で点灯する」

＊「Neopixel」－「strip 設定されている色を 1 個ずらす」

6-6　レインボーパターンの点灯

【例題 6-8】（プログラム prei6-8）

　Neopixel のライブラリの中には、「レインボーパターンに点灯する」という命令がある。レインボーパターンを使って、虹色で光らせてみよう。



「最初だけ」ブロック

　例題 6-4 に同じ（図は、省略）

「ボタン A」

＊「レインボーパターン」(色相 1 から 360)に点灯する」を選択する

＊下へ移動させるために、「strip に設定されている色を 1 個ずらす」を選択する

「ボタン B」

＊「レインボーパターン」(色相 1 から 360)に点灯する」を選択する

＊上へ移動させるために、「strip に設定されている色を－1 個ずらす」を選択する

付録Ⅰ　入力用拡張ブロック

micro:bit を利用する際に、他のプログラミング言語（特にスクラッチ）との互換性及びデータの入力方法を統一的にするために入力用の拡張ブロックを用意している。

入力用拡張ブロック（「Joholext」）には、つぎの２つのブロックがある。

・「〇〇と聞いて待つ“　”」

数値を０から９まで変更できるブロックである。A ボタンを押すごとに、数値が０から１ずつ増え、B ボタンを押すと確定する。

・「A　ボタンが押されるまでまつ」（A、B、A+B　ボタン）

ボタンが選択されるまで待つブロックである。A ボタンを押されると「１」、B ボタンを押されると「2」、A＋B ボタンを押される「3」が返される。

注）「Joholext」の「Johol」は、高校の共通教科情報科の「情報 I」を想定して名付けている。

＜拡張ブロックの追加手順＞

＊「ブロック」から「JavaScript」　にする

＊エクスプローラーの「▽」（左下の囲み箇所）を選び、「＋」からカスタムブロックを追加する

＊新しいファイルを追加しますか？　→　「Joholext」とする

＊「// ここにコードを追加します」　→　入力用拡張ブロックのテキストを貼り付ける

注）入力用拡張ブロックのテキストは、下記の Web に掲載されている。

＜参考資料＞　喜家村 奨：初等・中等教育におけるプログラミングの指導、入力用拡張ブロック

https://u-manabi.net/microbit/kaken/

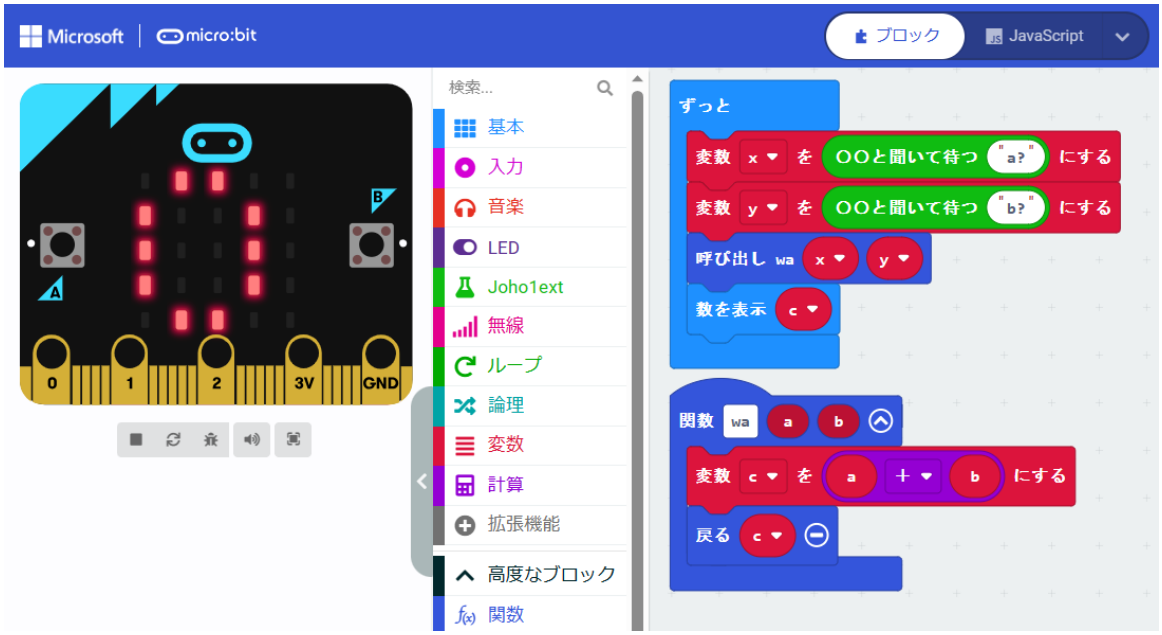
＊ブロックに戻す　→　「Joholext」が追加されている

＜入力用拡張ブロックの利用例＞

利用例を(1)、(2)に示す。

(1)「○○と聞いて待つ“　”」

例題 2-7 に示した関数(wa)では、a、b の数値は、プログラム内で固定されているが、値を変更して入力したい場合は、つぎの例のように、拡張ブロック（「○○と聞いて待つ“　”」）を利用する。



「ずっと」ブロック

＊「a？」と表示し、a の値が入力されるまで待ち、入力されたら x に代入する

＊「b？」と表示し、b の値が入力されるまで待ち、入力されたら y に代入する

(2)「A　ボタンが押されるまでまつ」

micro:bit には、「入力」に「A　が押されたとき」というブロックがあるが、このブロックを利用しない場合は、つぎの例のように、拡張ブロック（「A　ボタンが押されるまでまつ」）を利用する。この例では、A、B、A+B のボタンを利用し、A ボタン（戻り値 1）のときは、うれしい顔、B ボタン（戻り値 2）のときは、かなしい顔、A+B ボタン（戻り値 3）のときは、困り顔のアイコンを表示する。