## Ⅰ. micro:bit のプログラム

### I-I　「最初だけ」ブロックの利用（ハートの表示）（プログラム prei1-I）


「最初だけ」ブロック

＊「基本」-「LED 画面に表示」

### I-2　「ずっと」ブロックの利用（ハートの点滅）（プログラム prei1-2）


「ずっと」ブロック

＊「基本」-「一時停止」「表示を消す」

## ２．プログラムの基本（順次構造、反復構造）

### 2-1　順次構造（プログラム prei2-1）


「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」-「表示を消す」

### 2-2　反復構造（繰り返し 5 回）（プログラム prei2-2）


「最初だけ」ブロック

＊「基本」-「アイコン表示」の「はさみ」を選択

　じゃんけんの「グー」「チョキ」「パー」表示とする。

＊「ループ」－「くりかえし」　数値の 0→ 5

反復構造は、繰り返し構造ともいう。

別解（カウンター利用）（繰り返し５回）（プログラム prei2-3）


「最初だけ」ブロック

＊「ループ」の「変数 カウンター ～ くりかえす」 数値の 0→ 4

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

３．プログラムの基本（分岐構造）

3-1 分岐構造（分岐２つ）（プログラム prei3-1）


「最初だけ」ブロック

＊「変数」-「変数を追加する」 → 「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～）

＊「論理」-「くらべる」-「0 ＝▽ 0」 → 「c ＝▽ 0」 にする

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。

3-2　分岐構造（分岐２つ、乱数の利用）（プログラム prei3-2）


「ずっと」ブロック

＊「変数」-「変数を追加する」　→　「c」にする

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

＜参考＞　micro:bit の Python への自動変換プログラム

(1) 反復構造（prei2－3）

```
for カウンター in range(5):
    basic.show_icon(IconNames.SMALL_DIAMOND)
    basic.pause(500)
    basic.show_icon(IconNames.SCISSORS)
    basic.pause(500)
    basic.show_icon(IconNames.SQUARE)
    basic.clear_screen()
```

(2) 分岐構造（prei3－1）

```
c = 0
if c == 0:
    basic.show_icon(IconNames.SMALL_DIAMOND)
else:
    basic.show_icon(IconNames.SQUARE)
```

## 4．LED の点灯

### 4-1　光センサによる「ハート」の点滅（プログラム p-rei4-1）

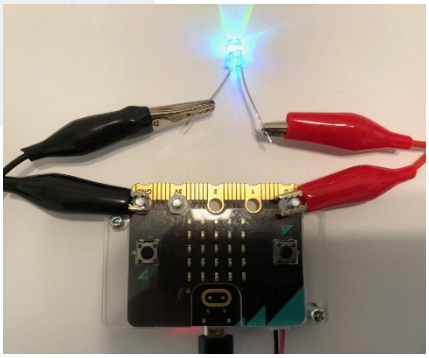

「ずっと」ブロック

＊「論理」－「条件判断」（もし　なら～）

＊「論理」－「くらべる」「0 < = 0」

＊「入力」－「明るさ」を重ねる。あとの数値 0→　150、シミュレータの明るさの数値は 128

### 4-2　光センサによる LED の点灯（プログラム p-rei4-2）

「ずっと」ブロック

＊「論理」－「条件判断」（もし　なら～）

＊「論理」－「くらべる」「0 < = 0」

＊「入力」－「明るさ」を重ねる。あとの数値 0→　125

＊「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

＊シミュレータの明るさの数値は 75、デジタル端子 P0 の出力は 1

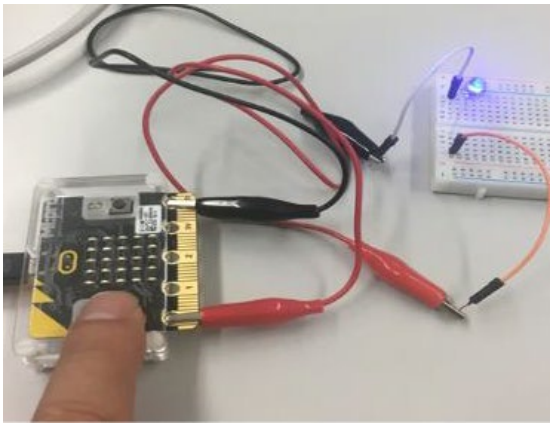
4-3　スイッチによる LED の点灯（プログラム p-rei4-3）

「最初だけ」ブロック
＊「変数」－「変数を追加」　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」（もし　なら～）
＊「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」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」　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)

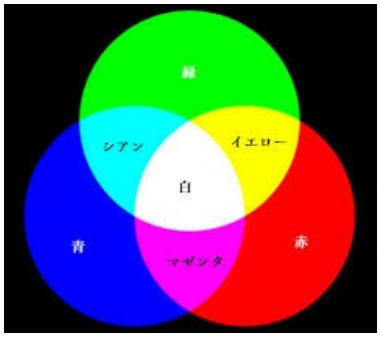
## ５．フルカラーLED の制御

【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 の集合体です。

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ります。

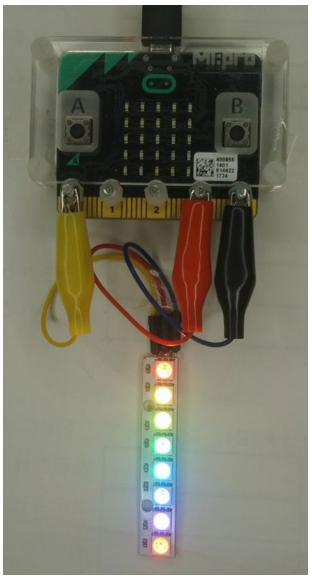
なお、R:255、G:255、B:255 では白、R:0、G:0、B:0 では黒になります。

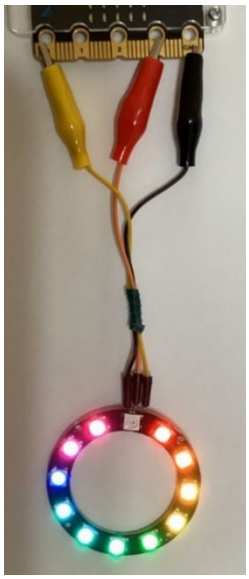



光の３原色　R(赤) G(緑) B(青)

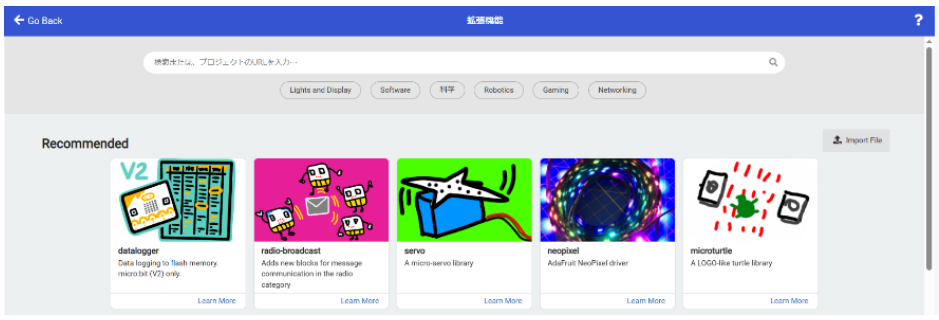
【micro:bit と Neopixel の接続】

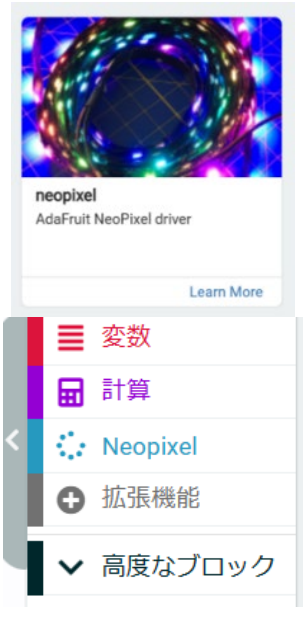
micro:bit と Neopixel を右図のように接続します（黄色は P0 端子,赤は 3V 端子,黒は GND 端子)。右図の Neopixel は、８個の LED が棒状（Stick 型）に接続された製品です。その他にも、Neopixel には、リング状の製品があります。



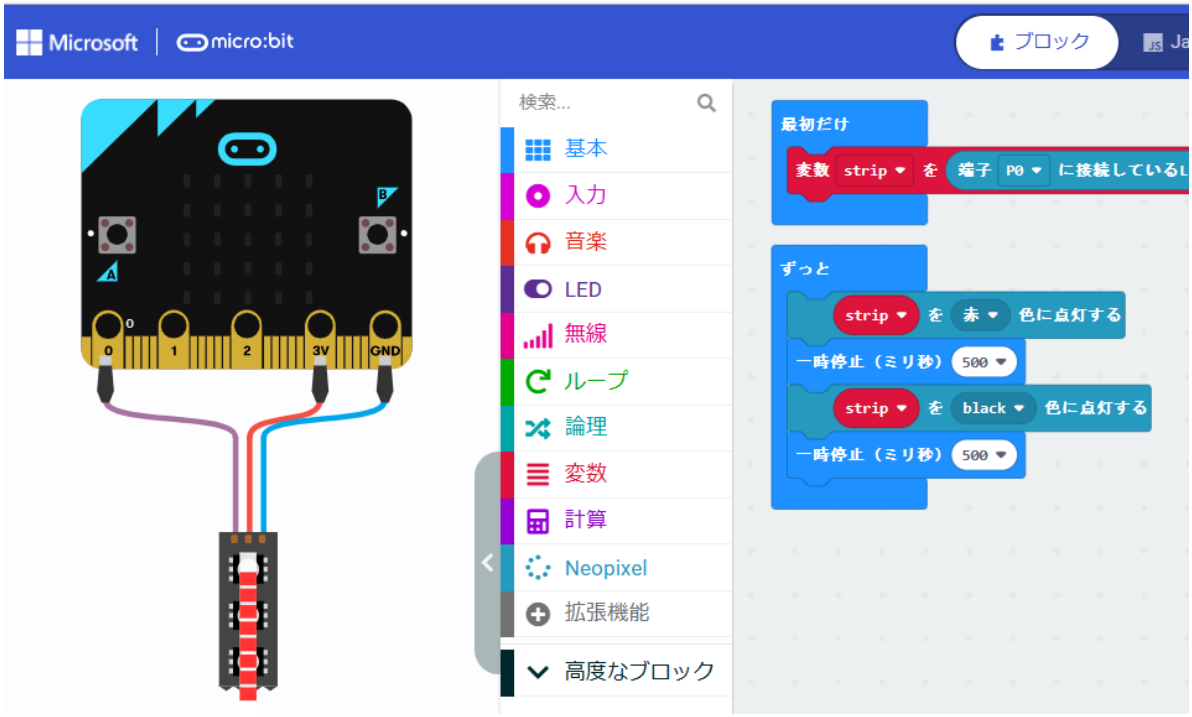


Neopixel を利用するには、ライブラリが必要です。ライブラリは、ツールボックスの下にある「⊕拡張機能」をクリックすると、拡張機能の一覧が表示されるので、「Neopixel」を選択する。ツールボックスの「計算」の下に、「Neopixel」のブロックが追加されます。



5-1　Neopixel の点滅(赤色の点滅)（プログラム p-rei5-1）





「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

＜参考＞（プログラム p-rei5-2）

　「Neopixel」で、「RGB（赤　緑　青)」色に点灯する」に変更すると、色はフルカラーで表現できる。下図では、シアン（水色に近い青緑色）になる。

5-2　Neopixel の点滅（8 色の点灯）（プログラム p-rei5-3）


「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」

5-3　Neopixel の点滅（色の上下移動）（プログラム p-rei5-4）

「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　回」で、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」


＜2 色の上下移動）＞（プログラム p-rei5-5）

　A ボタンで、赤色が上から下へ、B ボタンで、緑色が下から上へ移動するように変更する。

「A ボタン」

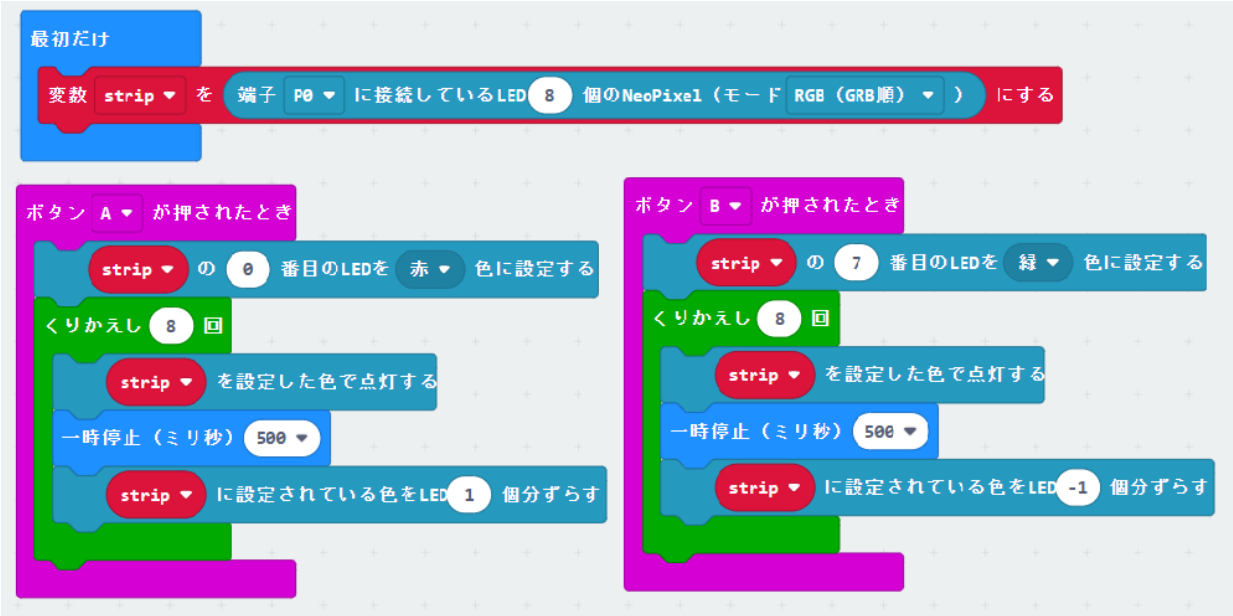
＊前の例題の「ずっと」ブロックを変更する。

「B ボタン」

＊「strip の 0 番目　緑 色に設定する」

＊「strip で設定した色で点灯する」

＊「strip 設定されている色を LED -1 個ずらす」とする




＜参考＞ エレベータのシミュレーション（プログラム p-rei5-6）

１階から８階までの建物で、エレベータで移動できるものする。次のような場合、Neopixel で表示するプログラムを考えてみよう。

・エレベータは、最初は１階（もしくは、８階）にある。

・１階で A ボタンを押すと、１階から上へ移動して、５階で止まる。

・８階で B ボタンを押すと、８階から下へ移動して、３階で止まる。

・上行きの移動は、赤で表示し、下行きの移動は、緑で表示する。