



Unit 7: TI-RGB Array

Skill Builder 1: それらを照らす

このレッスンでは、TI-RGB Array上の16個のLEDをまとめて(一度に)、および個別に(一度に1つずつ)制御する方法を学習します。

Note: このユニットでTI-RGB Arrayのlight(ライト)を再現するため使われるシミュレーションは、デモンストレーション目的でのみ使用されます。このツールはTI製品ではなく、TIから購入や配布はされません。

Caution: ライトが速く点滅することは、一部の学生にとっては邪魔になることがあるため、**sleep()**ステートメントを使って速度を少し遅くすることをお勧めします。

目標

- すべてのLEDを点灯し、ループを使って同時に点滅
- 別のループを使ってLEDを一度に1つずつ点灯および消灯



TI-RGB Array



背面



ブレッドボードポート

TI-RGB Arrayは16色のLEDとコントローラチップを備えた回路基板であり、短い4線ケーブルが付属しています。TI-Innovator Hubのブレッドボード(BB)ポートを使って、TI-Innovator Hubに接続します。回路基板の背面にある配線手順に従って、回路基板をTI-Innovator Hubに接続します。また、TI-Innovator HubはTI-Nspire CX IIに接続します。

1. Hub Projectテンプレートを使って新規のPythonプログラムを開始します。

menu > TI Hub > Add Output Device(メニュー>TI Hub>出力デバイス追加)を押して、TI-RGB Arrayを選択します。

varプレースホルダー(仮に確保した場所)の代わりに、任意の変数名を入力します(右図参照)。私たちは**cb** (circuit board(回路基板)用)としました。

```

1.1 1.2 1.3 *Unit 7 Py... RAD 1/19
*u7sb1.py
#=====
from ti_hub import *
from math import *
from time import sleep
from ti_plotlib import text_at,cls
from ti_system import get_key
#=====
cb=rgb_array()

```

2. ライトを一度に点灯させるには、プログラムの次の行に変数名の後にピリオドまたは小数点を入力します。

cb.

エディタに**rgb_array()**クラスで使えるすべてのメソッド(関数)を示すダイアログボックスがポップアップ表示されます。変数**cb**はそのクラス(オブジェクト)のインスタンスであり、これらのクラスメソッドのいずれかを使います。

```

1.1 1.2 1.3 *Unit 7 Py... RAD 10/19
*u7sb1.py
#=====
from ti_hub import *
from set(led_position,red,green,blue)
from set_all(red,green,blue)
from all_off()
from pattern(value)
from measurement()
cb=rgb_array()
cb.

```

図のように、**set_all(red, green, blue)**を選択します。

3. ステートメントは、変数名、ピリオド、ポップアップリストから選択した関数で構成されます。

Pythonメニューの他の多くのコマンドと同様、このコマンドには3つのインラインプロンプト(**red**, **green**, **blue**)が含まれ、それぞれに許可された値(0~255)を示すツールチップがあります。

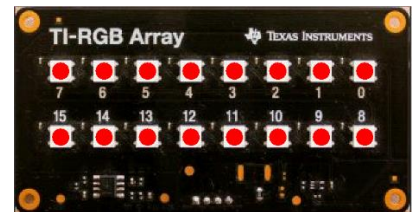
```

1.1 1.2 1.3 *Unit 7 Py_ray RAD 10/19
*u7sb1.py
#=====
from ti_hub import *
from math import *
from time import sleep
from ti_plotlib import text_at,cls
from ti_system import get_key
#=====
cb=rgb_array()
cb.set_all(red,green,blue)
0-255
    
```

4. 3色の値を選択して入力します。

プログラムを実行して、16個のLEDすべてが自分の色で点灯することを確認します。

プログラムが終了した後も、LEDは点灯したままであることを注意します。



5. LEDをオフにするには、次のステートメントを使います。

cb.all_off()

変数名とピリオドをもう一度入力し、リストから**all_off()**を選択します。

そして、点灯と消灯の間に**sleep(2)**(秒)を追加します。そうしないと、何も表示されません。

ここでプログラムを実行すると、LEDは2秒間点灯したままになります。

```

1.1 1.2 1.3 *Unit 7 Py_ray RAD 14/22
*u7sb1.py
from math import *
from time import sleep
from ti_plotlib import text_at,cls
from ti_system import get_key
#=====
cb=rgb_array()
cb.set_all(255,0,0)
sleep(2)
cb.all_off()
    
```

6. LED制御ステートメントをforループに入れて、数回点滅させます。それらがオフにされた後、別の**sleep()**を追加します。物事を少しスピードアップするためsleep時間を調整したいかもしれません。ループブロック内のすべてのステートメントをインデント(字下げ)してください。

Note: コンストラクタステートメント**cb = rgb_array()**をループブロックに含めないでください。一度だけ定義する必要があります!

続行する前に、プログラムを実行してテストします。

```

1.1 1.2 1.3 *Unit 7 Py_ray RAD 14/16
*u7sb1.py
from math import *
from time import sleep
from ti_plotlib import text_at,cls
from ti_system import get_key
#=====
cb=rgb_array()
for i in range(10):
    cb.set_all(255,0,0)
    sleep(1)
    cb.all_off()
    sleep(1)
    
```

7. プログラムがすべてのLEDを一度に点滅させたときが成功です。

つぎに、内部ループを使ってLEDを一度に1つずつ制御しましょう。



(demo1.1.gif)

8. **for i...**ループの下に、内側のループを追加します：**for j in range(16):**
 4つのループブロックステートメントをすべてインデント(字下げ)して、内側のループに適用されるようにします。
 ステートメント**cb.set_all(...)**を削除しますが、空白行は残します。その代わりに、次を入力します。

cb.

そして、**set(led_position, red, green, blue)**を選択します。

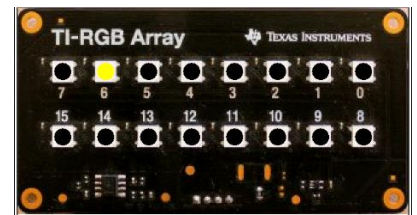
内側のループ変数**j**を**led_position**として使い、色の値を入力します。
 スリープ値と外側のループ**range()**を変更して、処理を少し高速化します。

9. プログラムを実行します。これで、16個のLEDが一度に1つずつ3回点灯します。

ドキュメントを保存することを忘れないでください。

```

1.1 1.2 1.3 *Unit 7 Py...ray RAD 15/17
* u7sb1.py
from time import sleep
from ti_plotlib import text_at,cls
from ti_system import get_key
#=====
cb=rgb_array()
for i in range(3):
    for j in range(16):
        cb.set_all(j,255,255,0)
        sleep(1)
        cb.all_off()
        sleep(1)
  
```



(demo1.2.gif)