Unit 7: The TI-RGB Array — Skill Builder 1: Light Them Up

In this lesson, you will learn to control the 16 LEDs on the TI-RGB Array both collectively (all at once) and individually (one at a time).

Note: The simulation that is used in this unit to replicate the lights on the TI-RGB Array is used for demonstration purposes only. This tool is not a TI product and is not available for purchase or distribution by TI.

Caution: Rapidly flashing lights may be disturbing for some students, so it is wise to use **sleep()** *statements to slow things down a bit.*

Objectives:

- Light up ALL LEDs and make them blink in unison using a loop
- Use another loop to light up and turn off the LEDs one at a time

TI-RGB Array

back

breadboard ports

The TI-RGB Array is a circuit board with 16 color LEDs and a controller chip and comes with a short 4-wire cable. It connects to the TI-Innovator Hub using the breadboard (BB) ports on the TI-Innovator Hub. Follow the wiring instructions on the back of the circuit board to connect it to the TI-Innovator Hub; connect the TI-Innovator Hub to your TI-Nspire CX II.

1. Start a new Python program using the Hub Project template.

   Press **menu > TI Hub > Add Output Device** and select the **TI-RGB Array**.

   In place of the *var* placeholder, type any variable name. We used **cb** (for 'circuit board').

2. To get the lights to light up all at once, on the next line of your program type your variable name followed by a period or decimal point.

   cb.

   A dialog box pops up in the editor showing all the methods (functions) that are available to the ***rgb_array()*** *class. Your variable (***cb***) is an instance of that class (an object) and can use any of these class methods.*

   Select **set_all(red, green, blue)**, as shown.

3. Your statement consists of your variable name, a period, and the function you selected from the pop-up list.
   As with a lot of other commands in the Python menus, this one contains three inline prompts (**red**, **green**, and blue) with a tool tip on each indicating the allowed values (0-255).

4. Choose and enter values for the three colors.

   Run your program and see that all 16 LEDs light up in your color.
   Notice that the LEDs remain lit even after the program ends.

5. To turn the LEDs off, use the statement:

   cb.all_off()

   Type your variable name and a period again and select **all_off()** from the list. And, add a **sleep(2)** (seconds) between the lighting up and the shutting off; otherwise, you won't see anything!

   When you run the program now, the LEDs stay lit for two seconds.

6. Put your LED control statements into a **for** loop to make them blink on and off several times. Add another **sleep()** after they are turned off. You may want to adjust the sleep times to speed things up a bit. Be sure to indent all statements in the loop block.

   Note: *Do not* include the *constructor* statement **cb = rgb_array()** in the loop block. It only needs to be defined once!

   Run your program to test it before you continue.

7. If your program blinks all LEDs at once, then you are successful. Now let's control the LEDs one at a time with an inner loop.

(demo1.1.gif)

8. Below the **for i…** loop, add an *inner* loop: **for j in range(16):**
   Indent all four loop block statements so that they now apply to the *inner* loop.
   Delete the statement **cb.set_all(…)**, but leave the blank line. In its place, type
   cb.
   and select **set(led_position, red, green, blue)**.

   Use the inner loop variable **j** as the led_position and enter your color values.
   Change the sleep values and the outer loop **range()** to speed things up a bit.

9. Run your program. Now your 16 LEDs light up one at a time for 3 times.

   Remember to save your document.

(demo1.2.gif)