



Unit 7: The TI-RGB Array

Application: Smart Lights

In this application, you will control the number of LEDs lit on the TI-RGB Array using the TI-Innovator Hub's brightness sensor.

Objectives:

- Use the brightness sensor to control the TI-RGB Array
- Adjust the brightness range to suit the TI-RGB Array
- Make sure that all 16 LEDs are impacted by the brightness

Smart Lights

As the room darkens, the lights in the room get brighter. Imagine a 'smart home' with no light switches! Write a program that monitors the brightness and turns on more or less LEDs, as necessary.



Teacher Tip: One of the challenges in this project is getting ALL the LEDs to react to some brightness value, including the end conditions: all on and all off. Your students may have to adjust the **brightness.range()** values to suit your lighting conditions. A smartphone flashlight will be helpful to control brightness.

1. As in the previous lessons, begin this Python Hub Project by importing the TI-RGB Array module and use the **r = rgb_array()** constructor and the **while** loop to terminate the program with **[clear]**.

```

r = rgb_array()
while not escape():
  ♦ ♦

```





2. To use the brightness sensor, you must first import the **Brightness** module. Press **[math] ti_hub... Hub Built-in devices Brightness**

import brightns

Then, before the **while** loop, set the **brightness.range()** to match the number of LEDs on the TI-RGB Array board that could be lit: 0 to 16.

brightness.range(0,16)

by pressing **[math] Brightness...**

Use the range (**0, 16**) because this is the range of the number of LEDs to light up on the board.

The maximum value the sensor will produce is 16. Is the minimum 0?

```

EDITOR: RGBD
PROGRAM LINE 0007
# Hub Project
from ti_system import *
from time import *
from rgb_arr import *
r=rgb_array()
import brightns
brightness.range(0,16)
while not escape():
**

```

Teacher Tip: A value of 0 is hard to achieve on the sensor. The next step converts the brightness value to an integer.

3. In the **while** block, start by reading the **brightness.measurement()** and store the value in a variable (**b**).

◆ ◆ **b = brightness.measurement()**

The function produces a floating-point number (float, decimal). Convert it to an integer value using:

◆ ◆ **b = int(bright)**

or combine the two statements into one operation:

◆ ◆ **b = int(brightness.measurement())**

Recall that **int()** is found on **<Fns...> Type**

4. To test your program: add **disp_at()** statement found on **[math] ti_hub Commands:**

◆ ◆ **disp_at(6, str(b), "left")**

Recall that you need **str(b)** because the **disp_at()** function requires a string to display. You can either type **str()** or get it from **<Fns...> Type**.

Run the program to ensure that all seventeen values (0...16) do appear. If not, then adjust the **range()** so that they do. Try using an artificial light source such as a flashlight or the 'flashlight' feature on a smartphone.

You may want to add **disp_clr()** before the loop (not shown).

```

EDITOR: RGBD
PROGRAM LINE 0011
# Hub Project
from ti_system import *
from time import *
from rgb_arr import *
r=rgb_array()
import brightns
brightness.range(0,16)
while not escape():
**b=brightness.measurement()
**b=int(b)
**

```

```

EDITOR: RGBD
PROGRAM LINE 0011
from ti_system import *
from time import *
from rgb_arr import *
r=rgb_array()
import brightns
brightness.range(0,16)
while not escape():
**b=brightness.measurement()
**b=int(b)
**disp_at(6,str(b),"left")
**

```



5. Since 0 is the darkest value and 16 is the brightest, we want the number of LEDs lit to be the *opposite*: when bright = 0, there should be 16 LEDs lit and when bright is 16, there should be 0 LEDs lit.

Write an expression for the number of lights (**l**) in terms of brightness (**b**).

$$l = ???$$

```
EDITOR: RGBD
PROGRAM LINE 0012
from time import *
from rgb_arr import *
r=rgb_array()
import brightns
brightns.range(0,16)
while not escape():
    b=brightns.measurement()
    b=int(b)
    disp_at(6,str(b),"left")
    l=???
```

6. It's possible that all LEDs should be off:

```
◆◆ if l == 0:
◆◆◆◆ r.all_off()
◆◆ else:
◆◆◆◆
```

```
EDITOR: RGBD
PROGRAM LINE 0016
import brightns
brightns.range(0,16)
while not escape():
    b=brightns.measurement()
    b=int(b)
    disp_at(6,str(b),"left")
    l=???
    if l==0:
        r.all_off()
    else:
        -
```

7. We want *all* the LEDs to be affected by the brightness so we will use a **for** loop to control the state of every LED every time. The variable **l** is a deciding factor when turning an LED on or off:

```
for i in range(1,17):
```

(Remember that the value 17 is not processed by the loop. The variable i takes on the values from 1 to 16 thus representing the 16 LEDs.)

```
EDITOR: RGBD
PROGRAM LINE 0017
brightns.range(0,16)
while not escape():
    b=brightns.measurement()
    b=int(b)
    disp_at(6,str(b),"left")
    l=???
    if l==0:
        r.all_off()
    else:
        for i in range(17):
            -
```

8. Complete the program by adding an **if...else...** statement to tell the TI-Innovator Hub which LEDs are on and which ones are off.

Hint: If **l** is 1, then you want to turn on LED 0. When **l** is 16, you want to turn on all LEDs (#0 to #15). Use the color (255,255,255) to get a bright white light.

Remember to turn **all** the LEDs **off** just before the end of the program.





Teacher Tip: The complete program (**bold** text is not provided in the lesson):

```
r = rgb_array()
brightness.range (0,16)
disp_clr()
while not escape():
    b = brightns.measurement()
    b = int(bright)
    disp_at(7,str(b),"left")
    l = 16 - b
    if l == 0:
        r.all_off()
    else
        for i in range (1,17) :
            if i <= l
                r.set(i-1,255,255,255)
            else:
                r.set(i-1,0,0,0)
cb.all_off()
```

Tip: Connect a power supply to the TI-Innovator Hub and use **r==rgb_array** (“as lamp”) for much brighter LEDs.

For a greater challenge: Instead of each LED being lit or not, have the program *gradually* brighten or darken the next LED as needed. Use a float for **b** and **l** and use the decimal part to gradually light up an LED.