



Unit 3: Brightness, if and while with the TI-Innovator™ Hub

Skill Builder 3: Brightness and Color

In this lesson, you will use the brightness sensor to control the color LED.

Objectives:

- Use **brightness.range()** to change the brightness scale
- Use the brightness value to light up the color LED
- Investigate numeric transformations

Unlike the TI-Innovator Hub light (the red LED), the color LED can vary in brightness. You will now use the brightness sensor to control that LED.



1. Make another copy of your brightness metering program (or your brightness and light program from the last lesson – they are almost the same).

Since the three color channels of the color LED only permit values from 0 to 255, set the brightness range from 0 to 255.

brightness.range(0, 255)

The brightness value **b** can now be used as values for the three color channels...perhaps.

2. Add a statement after the **brightness.measurement()** statement to light up the color LED using the variable **b** for all three channels:

color.rgb(b,b,b)

Run your program.

```

1.1 1.2 1.3 *Unit 3 Pyt...ile RAD 12/21
u3sb3b.py
from ti_system import get_key
#=====
cls()
text_at(13,"Press [esc] to end","center")

brightness.range(0,255)

while get_key() != "esc":
    b=brightness.measurement()
    
```

```

1.4 1.5 1.6 *Unit 3 Pyt...ile RAD 13/25
u3sb3b.py
from ti_system import get_key
#=====
cls()
text_at(13,"Press [esc] to end","center")

brightness.range(0,255)

while get_key() != "esc":
    b=brightness.measurement()
    color.rgb(b,b,b)
    
```

Teacher Tip: At this point, the brighter the room, the brighter the LED. This is backwards.



- Notice that the LED gets *brighter* as the light level increases. This is backwards! The darker the room, the brighter the light should be. Change the variable **b** (between the `brightness.measurement()` and the `color.rgb()`) to make the LED bright for low values and dim for large values. (See `b=???` in the screen to the right.)

Try it yourself before proceeding to the next step.

```

1.2 1.3 1.4 *Unit 3 Pyt...ile RAD 17/21
u3sb3b.py
text_at(13,"Press [esc] to end","center")
brightness.range(0,255)
while get_key() != "esc":
    b=brightness.measurement()
    b= ???
    color.rgb(b,b,b)
    text_at(7,"brightness = "+str(b),"left")
  
```

Teacher Tip: Making a linear transformation of a variable is common in programming.

- Here's one that works:

$$b=255 - b$$

When **b** is 0, the expression `255-b` changes the value of **b** to 255; when **b** is 255, the expression `255-b` changes the value of **b** to 0.

You may have to move your `text_at()` statement in the program to display the original value of **b** and not the transformed value.

Can you modify the program to produce other colors besides white?

When the program ends, the color LED may remain on. Add a statement at the end of the loop (not indented) to turn the color LED off.

```

1.2 1.3 1.4 *Unit 3 Pyt...ile RAD 15/21
u3sb3b.py
text_at(13,"Press [esc] to end","center")
brightness.range(0,255)
while get_key() != "esc":
    b=brightness.measurement()
    b= 255-b
    color.rgb(b,b,b)
    text_at(7,"brightness = "+str(b),"left")
  
```

Teacher Tip: New to programming? Statements similar to `b=255-b` may look incorrect but are very common in programming. The expression on the right is evaluated first, then the result of the calculation is stored in the variable on the left. The value of **b** on the right is *different* from the value on the left.

One way of using different colors is to only use two channels. Red and green make yellow: `color.rgb(b, b, 0)`. Or change between red and green using `color.rgb(b,255-b,0)`.