Basic Patterns

An interesting thing to do with these lights is to make them behave in a pattern. In this section, you'll learn how to make the lights perform some basic patterns, and then you are free to create your own pattern.

Challenge 1:

Blink Alternating Lights

A basic pattern is to blink every other light. Using what you've learned about loops, if-statements, and the modulus operator, add code toblink_alternating_lights() to blink every other light. Test it in the usual manner.

Challenge 2:

Fading the Lights

So far, you have learned how to blinks lights and change their color. Another neat feature of these lights is their ability to fade. To fade all the lights from off to on (or vice versa), you slowly adjust the brightness. Take a look at the given function fade_brightness(). This code for this function is *almost* complete, but not quite. Uncomment the given code, and then modify it so that the lights fade from off to on. Test in the usual manner. (Recall that you can fade individual lights by changing their color value.)

Note that setBrightness() will scale all colors to a max value, which will prevent the colors from appearing faded. Your lights are currently set to a default value of 64 in loop(), so that the FastLED colors will appear more vibrant. You may change this as you wish for your project.

Now that you can fade the lights, try:

- changing the color of the bulbs, so that they fade in a color of your choosing
- making the bulbs fade more quickly (There are TWO ways to do this. What are they?)

Challenge 3:

Chasing Lights

One of the most recognizable patterns is that of chasing lights. In this pattern, lights light up one after another in order so that it appears the light is moving down the strand.

For this task, implement a chase pattern inside the provided function chase().

Challenge 4:

Chasing Lights using leds_scroll()

Above you completed the chase pattern in the typical way, but this framework actually has special support for patterns such as chase. The leds data structure has not only an element for each light but also has an element for an extra light at the end of the string (spot 25). Additionally, it provides a function,leds_scroll() that will automatically scroll the pattern for you---by shifting the characteristics of bulb x to bulb x-1.

To use these features, set your desired characteristics to the fictional bulb leds[NUM_LEDS] and then call leds_scroll(). You'll see that characteristic propagate down the lights.

Once you have played with leds_scroll() some, implement a chase with alternating colors in the given stub functionchase_two_colors_with_scroll(). This task is harder than the ones you have had previously, so be certain to carefully consider what you need to do *before* you begin coding.

Challenge 5:

Create a Pattern of Your Choosing

Congratulations! You have completed all the given tasks related to controlling the lights and reached Competition Category One. Now you should create a pattern of your own design.

To implement your pattern, you may use the project files already found in the sketch or add new functions to the tests/tests.h files. Whichever you choose, be certain to call the appropriate functions from the loop() function in project_framework.

After creating a pattern of your own design, you may choose to move on to either dancing lights or displaying a message, which are also showcase categories, or you may show off your pattern in the patterns category. Whichever you choose, be sure to follow the showcase rules!

Towards the end of the week, you may want to eliminate the startup test pattern performed by the software. If you need our help to do so, just ask! We're always happy to help.

Source