Controlling Lights

Software

Now that the hardware is set up, let's take a look at the software. Software is the programs that tell the hardware what to do. We've already seen an example of a piece of software that controls the Arduino board: the Blink program. Now let's begin looking at software to control the lights through the Arduino board.

To begin, you'll first need to download the First Bytes framework into your home directory. The full framework is in another directory in this drive.

Then, using the terminal from which you opened the Arduino IDE, type the following commands, which will unzip and expand the archive you just downloaded:

```
gunzip firstbytes_framework.tar.gz
tar -xvf firstbytes_framework.tar
```

Next, we need to open the Project Illuminate framework. From within the Arduino IDE, select *File->Open*. From there, navigate to your home directory and click on the firstbytes folder. Within that directory, you'll find a project_framework folder. Double click on that and and then again onproject_framework.ino. The Arduino sketchbook called project_framework should be opened.

This framework already consists of a few files (indicated by the tabs across the top), and you may add your own as you progress in the project. The pre-existing files are briefly described here, and we'll look at some of them more in-depth below.

- project_framework: this file contains the setup() and loop() functions that we also saw in the Blink example. The setup() function is complete as provided (for now, as you know more you may want to change it), but you'll be modifying the loop() function.
- lights/lights.h: these files are a header file (lights.h) and source file (lights). These files contain the data structures that provide access to the lights and functions that will enable some basic capabilities. You'll want to use the functions, but you should not need to change any of the code in these files.
- tests/tests.h: these are the files where you will begin programming. They contain the stub code for many functions that will help you learn to control the lights yourself.
- project/project.h: these are the files where you will put your code for your project, once you finish the lights tutorial.

Some important things to note before we begin programming:

- The framework depends on functions and data structures defined in the FastLED library, which is included in the lib directory of your framework. We will only be using some of its capabilities, though.
- The framework is set up so that you make changes to the lights in the leds data structure but those changes are not reflected in the actual lights immediately. Instead, when you are ready for your changes to be reflected in the lights, you call the function FastLED.show(), which calls the show() function in the FastLED library, and will mirror your changes into the actual lights.
- While FastLED.show() updates the lights with your changes, we often want the changes we've made to appear for some amount of time before the lights change again. To help us do that, the FastLED library provides a function called delay, which you call in the exact same manner as show(), with one addition: the delay function takes one argument. That argument tells the program how long it should wait *in milliseconds* before proceeding to the next instruction. For example, if you would like to delay for one second, use the command:

- FastLED.delay(1000)
- delay() also calls show(), so you don't have to call one and then the other each time you want to update your lights. (Why might you want to show() without providing a delay?)
- Each light is represented by a struct called CRGB. This struct contains fields red, green, and blue. The color and brightness of each light is controlled by setting the values of those fields. The potential values for each field range from 0-255, where 0 represents no color and 255 represents maximum color. Using variations of these numbers, you can create many colors with a range of brightnesses. For instance, a red value of 255 and green and blue values of 0 result in the brightest red possible, where a red value of 64 and green and blue values of 0 is red at about 1/3 of maximum brightness. White is represented as 255, 255, 255, and black, which is the same as off, is represented as 0, 0, 0.
- The FastLED library also provides many predefined colors for you. We have created a sorted list for you, or you may consult the list in lib/pixeltypes.hline 464, which is in your project_framework directory. The colors in the file can be accessed by prepending CRGB:: to the name, such as CRGB::Aquamarine.
- The brightness of the light set as a whole is controlled through the function FastLED.setBrightness(), which sets the brightness for the entire strand of lights. Brightness values range from 0 to 255. Beware, though, as the 255 value results in very bright lights and some colors appear faded. To get some idea of how the colors appear at brightness level 64, execute the ColorPalette example in the FastLED library, which is found inproject_framework/lib/examples/ColorPalette. You can launch it by either opening the .ino file in that directory through the Arduino IDE or by navigating to that directory using the Navigator (double click on the hard drive labeled File System on your desktop).

Challenge 1:

Now, let's look at the tests file. At the top of this file, there is a function called blink_light_0_example, which is provided for you. Take a careful look at that function and convince yourself that you understand its contents. You'll also notice that this function is called from the loop() function inside theproject_framework file. When you are ready, test the example by uploading it to the Arduino board (use the arrow pointing to the right as you did to blink the LED). The lights will first perform some tests where they alternate PaleVioletRed and SteelBlue, and then the code in the loop() function will begin to execute. In this case, that code is our blink_light_0_example().

If the code works properly, you should notice light 0 blinking white. (Notice that this is light **0** and not light **1**!) Now, change the code to do the following things:

- Blink a color other than white. Remember that pixeltypes.h has pre-defined colors. You are also welcome to define your own. You can set the colors individually using code such as the code below, which changes the color of light 0:

```
Leds[0].red = 112;
Leds[0].green = 75;
Leds[0].blue = 23;
```

- Blink at a different brightness level.
- Blink at either a faster or slower frequency.

Challenge 2:

For our next test, let's blink both the first and last light. Recall that we have 25 lights. If the first light is number 0, what do you suppose the last light is?

Challenge 3:

In your C++ tutorial, you learned how to use loops. Using a loop, complete the blink_all_lights() function so that every light blinks. Once again, test it by commenting out any calls to other functions in the loop() function in project_framework and uncommenting the call to blink_all_lights() in that same function.

Source