A quick overview of the FastLED library and the various pieces/components of it

In addition, there is API level documentation at http://fastled.io/docs/3.1/modules.html

Supported Chipsets

The core of the library is providing support to a number of LED chipsets that are out there.

SPI Chipsets

(See the [chipset reference] wiki page for a more complete list with more information on each chip)

SPI based chipsets are usually 4 wires - data, clock, power, and ground. The advantage to these led chipsets is they can be cheap, the SPI data protocol does well over distance, and with some chipsets, you can get really high data transfer rates.

- LPD8806 - a workhorse of an SPI based chipset, allows for high speed writing of data (we've pushed over 20Mbps to these chips!).
- WS2801 - a lower speed SPI based chipset, data rate is limited to 1Mbps, but they're cheap
- SM16716 - another SPI based chipset, somewhat strange protocol.
- Total Control Lighting - Also known as P9813, this is an SPI chipset that is starting to see more use. (Support for this library has been written, just needs testing, now!)
- APA102 - another SPI based chipset, looks promising, packaging similar to the WS2812, but potentially much higher data rates (w/FastLED2.1)

3-Wire Chipsets

Three wire led pixels are becoming quite popular. Having only data, ground, and power lines, they're a bit more compact than the SPI based chipsets (even further, the WS2812B's combine the led controller chip and the led in a single package!).

- Neopixel - Aka the WS2811 (or WS2812, or WS2812B), these seem to be the rgb leds of choice for many people right now.

- TM1809/TM1804 - A common chipset for a while on aliexpress, has a similar protocol to the WS2811 and friends.
- TM1803 - Sold by radio shack
- UCS1903/UCS1903B/UCS1904/UCS2903 - Another 3 wire chipset, slower data protocol than the others, which were already slow. Not terribly common.
- GW6205 (w/FastLED2.1)
- LPD1886 - a 3 wire chipset that is 12-bits per pixel instead of the usually 7/8-bit per pixel most of the other chipsets seen are so far (w/FastLED2.1)

Not yet supported

Of course, progress always marches forward, and there's a variety of new led chipsets on the horizon for the library to support. Here's a taste of what's coming:

- AS1130
- TLC5940
- TM1829 - a 3 wire chipset that also allows dynamically changing the base line power usage!
- TM1812
- D3001/CY3001 - investigating this chipset.

Deprecated chipsets

The older version of this library, FastSPI_LED, supported a number of chipsets that required work on the host MCU to manage PWM. For a combination of reasons, including these chipsets going away, as well as a desire to get away from having the library rely on timer based code, those chipsets are no longer supported:

- *595 shift register chips
- HL1606 - spi based for on and off, PWM was managed on the host
- LPD6803 - spi based, did PWM on chip, but required host MCU to strobe clock line to drive PWM

Supported Platforms

One of the goals of this version of the library is to lay the groundwork for making it more easily portable to a wide variety of platforms. At the moment, the library has only been tested and is known to work with AVR and ARM based MCUs that are nominally arduino-compatable. Namely, if the code is built/pushed using some variation of the Arduino application with stock compilers, the library should work. Future versions of the library will support a wider range of platforms and compilers.

- Arduino - Pretty much all the official arduino platforms should be supported at this point, including the Due (as of FastLED2.1) and the Yún (as of FastLED 3.0.3) and the Zero (as of FastLED 3.1).
- Adafruit Flora/Gemma & Trinket - ATtiny based chipsets for wearable projects

- PJRC Teensy 2 & 3 - avr and arm based project boards from PJRC, arduino comparable, but with a variety of extra goodies on them. The teensy3 and teensy 3.1 is one of our favorite boards to dev against! The Teensy LC is supported as of FastLED 3.1
- RFDuino (as of FastLED 3.1)
- SparkCore and Photon
- ESP8266 - using the arduino board definitions from http://arduino.esp8266.com/stable/package_esp8266com_index.json

Some upcoming platforms:

- LPC8XX - this port is mostly done, it needs some polish (and unwinding from a contract job) to put in the library
- NRF52 - M4F with bluetooth on board - the 51822's bigger sibling!

Quick Examples

How easy is the library to use? Here's a quick example providing some blinking code:

```
#include "FastLED.h"

CRGB leds[1];
void setup() { FastLED.addLeds<NEOPIXEL, 6>(leds, 1); }
void loop() {
        leds[0] = CRGB::White; FastLED.show(); delay(30);
        leds[0] = CRGB::Black; FastLED.show(); delay(30);
}
```

HSV Library

Unsatisfied with the state of a lot of the HSV and color wheel libraries, we put a lot of work into providing an HSV to RGB conversion library that is fast and adjusted for human color perception. Why use the HSV color space? It's a bit easier to navigate and provide transitions between colors than using RGB. When defining colors with RGB you're mixing the Red, Green, and Blue color values. When using HSV, instead, you're defining the hue of the color (that is, where it is on the color wheel), how saturated it is, and how bright it is. For example, here's some simple code that will cycle through the colors of a rainbow:

```
#include "FastLED.h"
CRGB leds[60];
void setup() { FastLED.addLeds<NEOPIXEL, 6>(leds, 60); }
void loop() {
  static uint8_t hue = 0;
  FastLED.showColor(CHSV(hue++, 255, 255));
```

```
    delay(10);
  }
```

Math Library

When doing LED programming, a lot of times you want to do math on the rgb or hsv values to help provide your transitions in brightness and color. However, the AVR/arduino platform isn't exactly known for the fastest math out there. To help out with this, the library provides a number of math functions tuned for 8-bit operations, including scaling functions, fast sin/cos functions, fast random number generators, and interpolation and memory management functions.

Low level device access

Finally, to do a lot of the magic in writing to LEDs, the library has some generalized classes to provide high speed, flexible access to pins and SPI hardware. While write only at the moment, this code can help you access pins and devices quicker than the stock arduino libraries do, even moreso once read support is added to the pin and spi libraries.