// Use if you want to force the software SPI subsystem to be used for some reason (generally, you don't)

// #define FORCE\_SOFTWARE\_SPI

// Use if you want to force non-accelerated pin access (hint: you really don't, it breaks lots of things)

// #define FORCE\_SOFTWARE\_SPI

// #define FORCE\_SOFTWARE\_PINS

#include "FastLED.h"

///////////////////////////////////////////////////////////////////////////////////////////

//

// Move a white dot along the strip of leds. This program simply shows how to configure the leds,

// and then how to turn a single pixel white and then off, moving down the line of pixels.

//

// How many leds are in the strip?

#define NUM\_LEDS 31

// Data pin that led data will be written out over

#define DATA\_PIN 3

// Clock pin only needed for SPI based chipsets when not using hardware SPI

//#define CLOCK\_PIN 8

// This is an array of leds. One item for each led in your strip.

CRGB leds[NUM\_LEDS];

// This function sets up the ledsand tells the controller about them

void setup() {

// sanity check delay - allows reprogramming if accidently blowing power w/leds

delay(2000);

// Uncomment one of the following lines for your leds arrangement.

// FastLED.addLeds<TM1803, DATA\_PIN, RGB>(leds, NUM\_LEDS);

// FastLED.addLeds<TM1804, DATA\_PIN, RGB>(leds, NUM\_LEDS);

// FastLED.addLeds<TM1809, DATA\_PIN, RGB>(leds, NUM\_LEDS);

// FastLED.addLeds<WS2811, DATA\_PIN, RGB>(leds, NUM\_LEDS);

// FastLED.addLeds<WS2812, DATA\_PIN, RGB>(leds, NUM\_LEDS);

// FastLED.addLeds<WS2812B, DATA\_PIN, RGB>(leds, NUM\_LEDS);

FastLED.addLeds<NEOPIXEL, DATA\_PIN>(leds, NUM\_LEDS);

// FastLED.addLeds<WS2811\_400, DATA\_PIN, RGB>(leds, NUM\_LEDS);

// FastLED.addLeds<GW6205, DATA\_PIN, RGB>(leds, NUM\_LEDS);

// FastLED.addLeds<GW6205\_400, DATA\_PIN, RGB>(leds, NUM\_LEDS);

// FastLED.addLeds<UCS1903, DATA\_PIN, RGB>(leds, NUM\_LEDS);

// FastLED.addLeds<UCS1903B, DATA\_PIN, RGB>(leds, NUM\_LEDS);

// FastLED.addLeds<WS2801, RGB>(leds, NUM\_LEDS);

// FastLED.addLeds<SM16716, RGB>(leds, NUM\_LEDS);

// FastLED.addLeds<LPD8806, RGB>(leds, NUM\_LEDS);

// FastLED.addLeds<P9813, RGB>(leds, NUM\_LEDS);

// FastLED.addLeds<WS2801, DATA\_PIN, CLOCK\_PIN, RGB>(leds, NUM\_LEDS);

// FastLED.addLeds<SM16716, DATA\_PIN, CLOCK\_PIN, RGB>(leds, NUM\_LEDS);

// FastLED.addLeds<LPD8806, DATA\_PIN, CLOCK\_PIN, RGB>(leds, NUM\_LEDS);

}

// This function runs over and over, and is where you do the magic to light

// your leds.

void loop() {

// Move a single white led

// add the behind LED to be orange

int orangeLed = 1;

int redLed = 2;

int greenLed = 3;

int blueLed = 4;

for(int whiteLed = 0; whiteLed < NUM\_LEDS; whiteLed = whiteLed + 1) {

// Turn our current led on to white, then show the leds

leds[whiteLed] = CRGB::White;

leds[orangeLed] = CRGB::Orange;

leds[redLed] = CRGB::Red;

leds[greenLed] = CRGB::Green;

leds[blueLed] = CRGB::Blue;

// Show the leds (only one of which is set to white, from above)

FastLED.show();

// Wait a little bit

delay(100);

// Turn our current led back to black for the next loop around

leds[whiteLed] = CRGB::Black;

if(orangeLed < NUM\_LEDS) {

orangeLed++;

} else {

orangeLed = 0;

}

if(redLed < NUM\_LEDS) {

redLed++;

} else {

redLed = 0;

}

if(greenLed < NUM\_LEDS) {

greenLed++;

} else {

greenLed = 0;

}

if(blueLed < NUM\_LEDS) {

blueLed++;

} else {

blueLed = 0;

}

}

}