#include <Adafruit\_NeoPixel.h>

#ifdef \_\_AVR\_\_

 #include <avr/power.h>

#endif

#define PIN 7

// Parameter 1 = for number of pixels in strip

// Parameter 2 = Arduino pin number

// Parameter 3 = pixel type flags, add together as needed:

// NEO\_KHZ800 800 KHz bitstream (most NeoPixel products w/WS2812 LEDs)

// NEO\_KHZ400 400 KHz (classic 'v1' (not v2) FLORA pixels, WS2811 drivers)

// NEO\_GRB Pixels are wired for GRB bitstream (most NeoPixel products)

// NEO\_RGB Pixels are wired for RGB bitstream (v1 FLORA pixels, not v2)

// NEO\_RGBW Pixels are wired for RGBW bitstream (NeoPixel RGBW products)

Adafruit\_NeoPixel strip = Adafruit\_NeoPixel(240, PIN, NEO\_GRB + NEO\_KHZ800);

// IMPORTANT: To reduce NeoPixel burnout risk, add 1000 uF capacitor across

// pixel power leads, add 300 - 500 Ohm resistor on first pixel's data input

// and minimize distance between Arduino and first pixel. Avoid connecting

// on a live circuit...if you must, connect GND first.

void setup() {

 // This is for Trinket 5V 16MHz, you can remove these three lines if you are not using a Trinket

 #if defined (\_\_AVR\_ATtiny85\_\_)

 if (F\_CPU == 16000000) clock\_prescale\_set(clock\_div\_1);

 #endif

 // End of trinket special code

 strip.begin();

 strip.setBrightness(50);

 strip.show(); // Initialize all pixels to 'off'

}

void loop() {

 // Some example procedures showing how to display to the pixels:

 colorWipe(strip.Color(255, 10, 20), 50); // Red

 colorWipe(strip.Color(50, 255, 60), 50); // Green

 colorWipe(strip.Color(0, 0, 255), 50); // Blue

//colorWipe(strip.Color(0, 0, 0, 255), 50); // White RGBW

 // Send a theater pixel chase in...

 theaterChase(strip.Color(127, 127, 127), 50); // White

 theaterChase(strip.Color(127, 20, 0), 50); // Red

 theaterChase(strip.Color(50, 0, 127), 50); // Blue

 rainbow(20);

 rainbowCycle(20);

 theaterChaseRainbow(50);

}

// Fill the dots one after the other with a color

void colorWipe(uint32\_t c, uint8\_t wait) {

 for(uint16\_t i=0; i<strip.numPixels(); i++) {

 strip.setPixelColor(i, c);

 strip.show();

 delay(wait);

 }

}

void rainbow(uint8\_t wait) {

 uint16\_t i, j;

 for(j=0; j<256; j++) {

 for(i=0; i<strip.numPixels(); i++) {

 strip.setPixelColor(i, Wheel((i+j) & 255));

 }

 strip.show();

 delay(wait);

 }

}

// Slightly different, this makes the rainbow equally distributed throughout

void rainbowCycle(uint8\_t wait) {

 uint16\_t i, j;

 for(j=0; j<256\*5; j++) { // 5 cycles of all colors on wheel

 for(i=0; i< strip.numPixels(); i++) {

 strip.setPixelColor(i, Wheel(((i \* 256 / strip.numPixels()) + j) & 255));

 }

 strip.show();

 delay(wait);

 }

}

//Theatre-style crawling lights.

void theaterChase(uint32\_t c, uint8\_t wait) {

 for (int j=0; j<10; j++) { //do 10 cycles of chasing

 for (int q=0; q < 3; q++) {

 for (uint16\_t i=0; i < strip.numPixels(); i=i+3) {

 strip.setPixelColor(i+q, c); //turn every third pixel on

 }

 strip.show();

 delay(wait);

 for (uint16\_t i=0; i < strip.numPixels(); i=i+3) {

 strip.setPixelColor(i+q, 0); //turn every third pixel off

 }

 }

 }

}

//Theatre-style crawling lights with rainbow effect

void theaterChaseRainbow(uint8\_t wait) {

 for (int j=0; j < 256; j++) { // cycle all 256 colors in the wheel

 for (int q=0; q < 3; q++) {

 for (uint16\_t i=0; i < strip.numPixels(); i=i+3) {

 strip.setPixelColor(i+q, Wheel( (i+j) % 255)); //turn every third pixel on

 }

 strip.show();

 delay(wait);

 for (uint16\_t i=0; i < strip.numPixels(); i=i+3) {

 strip.setPixelColor(i+q, 0); //turn every third pixel off

 }

 }

 }

}

// Input a value 0 to 255 to get a color value.

// The colours are a transition r - g - b - back to r.

uint32\_t Wheel(byte WheelPos) {

 WheelPos = 255 - WheelPos;

 if(WheelPos < 85) {

 return strip.Color(255 - WheelPos \* 3, 0, WheelPos \* 3);

 }

 if(WheelPos < 170) {

 WheelPos -= 85;

 return strip.Color(0, WheelPos \* 3, 255 - WheelPos \* 3);

 }

 WheelPos -= 170;

 return strip.Color(WheelPos \* 3, 255 - WheelPos \* 3, 0);

}