**Understanding the Code**

This code is designed to control **NeoPixel LEDs** arranged in a **7-segment display format**, allowing it to display numbers (0-9) and perform animations.

**Key Components of the Code**

**1. Library and Constants**

#include <Adafruit\_NeoPixel.h>

* This library is used to control NeoPixel LED strips.

#define PIXELS\_PER\_SEGMENT 3 // Number of LEDs per segment

#define PIXELS\_DIGITS 1 // Number of digits in the display

#define PIXELS\_PIN D10 // The GPIO pin connected to the NeoPixel strip

* The **7-segment display** is made of **7 segments**, each segment containing **3 LEDs**.
* The number of **digits** (7-segment displays) is set to **1**.

Adafruit\_NeoPixel strip = Adafruit\_NeoPixel(PIXELS\_PER\_SEGMENT \* 7 \* PIXELS\_DIGITS, PIXELS\_PIN, NEO\_GRB + NEO\_KHZ800);

* Creates a NeoPixel strip object.
* The total number of LEDs = **(3 LEDs per segment) × (7 segments per digit) × (1 digit) = 21 LEDs**.
* The NeoPixels use the **GRB color format** and run at **800 kHz**.

**2. 7-Segment Pixel Mapping**

/\*

 a

 f b

 g

 e c

 d

\*/

* This defines how the **7-segment display** is structured.

byte segments[7] = {

 0b0000100, // Segment e

 0b0001000, // Segment d

 0b0010000, // Segment c

 0b0100000, // Segment b

 0b1000000, // Segment a

 0b0000010, // Segment f

 0b0000001 // Segment g

};

* Defines which **bit corresponds to which segment**.
* Each bit represents whether a segment is **ON (1)** or **OFF (0)**.

byte digits[10] = {

 0b1111110, // 0

 0b0011000, // 1

 0b1101101, // 2

 0b0111101, // 3

 0b0011011, // 4

 0b0110111, // 5

 0b1110111, // 6

 0b0011100, // 7

 0b1111111, // 8

 0b0011111 // 9

};

* Each number (0-9) is defined by **which segments are lit**.

**3. Clearing the Display**

void clearDisplay() {

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

 strip.setPixelColor(i, strip.Color(0, 0, 0)); // Turn off all LEDs

 }

 strip.show(); // Update LEDs

}

* Turns **all pixels off**.

**4. Setup and Loop**

void setup() {

 strip.begin(); // Initialize the NeoPixel strip

}

* Starts the NeoPixel strip.

void loop() {

 disp\_Digits(1000); // Show digits 0-9 with 1 second delay

}

* Calls disp\_Digits(), which **cycles through numbers 0-9**.

**5. Displaying Digits and Segments**

**Displaying All Segments**

void disp\_Seg(int wait) {

 clearDisplay();

 for (int d = 0; d < 5; d++) { // Cycle through all segments 5 times

 for (int i = 6; i > 0; i--) { // Iterate through each segment

 for (int n = 0; n < PIXELS\_DIGITS; n++) { // Apply to each digit

 writeSegment(n, i);

 }

 strip.show();

 delay(wait);

 }

 }

}

* Turns each **segment** on/off one by one.

**Displaying Numbers 0-9**

void disp\_Digits(int wait) {

 clearDisplay();

 for (int i = 0; i < 10; i++) { // Iterate through numbers 0-9

 for (int n = 0; n < PIXELS\_DIGITS; n++) { // Apply to each digit

 writeDigit(n, i);

 }

 strip.show();

 delay(wait);

 }

}

* Turns on the segments needed to display each **number**.

**Counting Up and Down**

void disp\_CountUP(int num, int wait) {

 clearDisplay();

 for (int i = 0; i <= num; i++) {

 writeDigit(0, (i / 100) % 10); // Hundreds place

 writeDigit(1, (i / 10) % 10); // Tens place

 writeDigit(2, (i / 1) % 10); // Ones place

 strip.show();

 delay(wait);

 }

}

* **Counts up** from 0 to num (e.g., 0 → 1 → 2 → …).
* Supports **multi-digit** displays.

void disp\_CountDOWN(int num, int wait) {

 clearDisplay();

 for (int i = num; i >= 0; i--) {

 writeDigit(0, (i / 100) % 10);

 writeDigit(1, (i / 10) % 10);

 writeDigit(2, (i / 1) % 10);

 strip.show();

 delay(wait);

 }

}

* **Counts down** instead.

**6. Writing a Digit to the Display**

void writeDigit(int index, int val) {

 byte digit = digits[val]; // Get bit pattern for digit

 for (int i = 6; i >= 0; i--) { // Iterate through 7 segments

 int offset = index \* (PIXELS\_PER\_SEGMENT \* 7) + i \* PIXELS\_PER\_SEGMENT;

 uint32\_t color;

 if (digit & 0x01 != 0) { // If segment should be ON

 if (val == 1) color = strip.Color(50, 0, 0); // Red

 if (val == 2) color = strip.Color(50, 50, 0); // Yellow

 if (val == 3) color = strip.Color(50, 0, 50); // Purple

 if (val == 4) color = strip.Color(0, 50, 0); // Green

 if (val == 5) color = strip.Color(0, 50, 50); // Cyan

 if (val == 6) color = strip.Color(0, 0, 50); // Blue

 if (val == 7) color = strip.Color(50, 25, 0); // Orange

 if (val == 8) color = strip.Color(25, 5, 75); // Custom

 if (val == 9) color = strip.Color(75, 25, 5); // Custom

 if (val == 0) color = strip.Color(5, 75, 25); // Custom

 }

 else color = strip.Color(0, 0, 0); // OFF

 for (int j = offset; j < offset + PIXELS\_PER\_SEGMENT; j++) {

 strip.setPixelColor(j, color); // Set segment color

 }

 digit = digit >> 1; // Move to next segment

 }

}

* Uses **bitwise operations** to determine which segments to turn **on/off**.

**Summary**

* The code **controls a NeoPixel-based 7-segment display**.
* It **cycles through numbers (0-9)**, counts **up/down**, and has **animations**.
* Uses **bitwise operations** to map segments.
* Different **colors are assigned** for each number.