```
/**
 * Copyright (c) 2020 Raspberry Pi (Trading) Ltd.
 *
 * SPDX-License-Identifier: BSD-3-Clause
 */

#include <stdio.h>
#include <stdlib.h>

#include "pico/stdlib.h"
#include "hardware/pio.h"
#include "hardware/clocks.h"
#include "ws2812.pio.h"

#define IS_RGBW true
#define NUM_PIXELS 150

#define neopixel_power 11
```

Includes all the necessary header files & libraries for the code to run & defines the constants.

NOTE: neopixel_power = 11 was added by me because the power to the LED is connected to GPIO 11 for our board.

```
#ifdef PICO_DEFAULT_WS2812_PIN
#define WS2812_PIN PICO_DEFAULT_WS2812_PIN
#else
// default to pin 2 if the board doesn't have a default WS2812 pin defined
#define WS2812_PIN 12
#endif
```

Defines if there is a default LED pin, use that, if not change it to 12 because connected to GPIO 12 on our board

```
static inline void put_pixel(uint32_t pixel_grb) {
    pio_sm_put_blocking(pio0, 0, pixel_grb << 8u);
}

static inline uint32_t urgb_u32(uint8_t r, uint8_t g, uint8_t b) {
    return
            ((uint32_t) (r) << 8) |
            ((uint32_t) (g) << 16) |
            (uint32_t) (b);
}
```

Put pixel takes a 32 bit value & writes it into FIFO.

```
void pattern_snakes(uint len, uint t) {
    for (uint i = 0; i < len; ++i) {
        uint x = (i + (t >> 1)) % 64;
        if (x < 10)
            put_pixel(urgb_u32(0xff, 0, 0));
        else if (x >= 15 && x < 25)
            put_pixel(urgb_u32(0, 0xff, 0));
        else if (x >= 30 && x < 40)
            put_pixel(urgb_u32(0, 0, 0xff));
        else
            put_pixel(0);
    }
}
```

This is just an LED pattern

```
void pattern_random(uint len, uint t) {
    if (t % 8)
```

```
        return;
    for (int i = 0; i < len; ++i)
        put_pixel(rand());
}
```

This function has different colorful patterns assigned randomly to put_pixel ()

```
void pattern_sparkle(uint len, uint t) {
    if (t % 8)
        return;
    for (int i = 0; i < len; ++i)
        put_pixel(rand() % 16 ? 0 : 0xffffffff);
}

void pattern_greys(uint len, uint t) {
    int max = 100; // let's not draw too much current!
    t %= max;
    for (int i = 0; i < len; ++i) {
        put_pixel(t * 0x10101);
        if (++t >= max) t = 0;
    }
}
```

These are just different LED patterns.

```
typedef void (*pattern)(uint len, uint t);
const struct {
    pattern pat;
    const char *name;
} pattern_table[] = {
        {pattern_snakes,  "Snakes!"},
        {pattern_random,  "Random data"},
        {pattern_sparkle, "Sparkles"},
        {pattern_greys,   "Greys"},
};
```

This creates a type called pattern for a pointer to a function that takes 2 arguments. It then creates a struct pattern table with different patterns

```
int main() {
    gpio_init(neopixel_power); -1
    gpio_set_dir(neopixel_power, GPIO_OUT); 2
    gpio_put(neopixel_power,1); -3
```

By default the LED power pin on our board is set to 0. This initializes & sets it as an output & value to 1.

```
    //set_sys_clock_48();
    stdio_init_all(); -4
    printf("WS2812 Smoke Test, using pin %d", WS2812_PIN); -5

    // todo get free sm
    PIO pio = pio0; -6
    int sm = 0; -7
    uint offset = pio_add_program(pio, &ws2812_program); -8

    ws2812_program_init(pio, sm, offset, WS2812_PIN, 800000, IS_RGBW); -9

    int t = 0; -21
    while (1) {
        int pat = rand() % count_of(pattern_table); 22
        int dir = (rand() >> 30) & 1 ? 1 : -1; 23
        puts(pattern_table[pat].name); — 24
```

Loads the program onto a PIO, configures the SM & cycles through the patterns randomly.

↳ goes to pio.h file

```
puts(dir == 1 ? "(forward)" : "(backward)");     —25
for (int i = 0; i < 1000; ++i) {
    pattern_table[pat].pat(NUM_PIXELS, t);     —26
    sleep_ms(10);     —27
    t += dir;     —28
}
```