```c
/**
 * 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

#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        connected to GPIO12
#define WS2812_Power_PIN 11  Power supply connected to GPIO11

#endif

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) |    Shift the color to RGB
            (uint32_t) (b);
}

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));    Red
        else if (x >= 15 && x < 25)
            put_pixel(urgb_u32(0, 0xff, 0));    Green
        else if (x >= 30 && x < 40)
            put_pixel(urgb_u32(0, 0, 0xff));    Blue
        else
            put_pixel(0);
    }
}

void pattern_random(uint len, uint t) {
    if (t % 8)
        return;                                 output a random color
    for (int i = 0; i < len; ++i)
        put_pixel(rand());
}

void pattern_sparkle(uint len, uint t) {
    if (t % 8)
        return;
```

```
62      for (int i = 0; i < len; ++i)
63          put_pixel(rand() % 16 ? 0 : 0xffffffff);
64  }
65
66  void pattern_greys(uint len, uint t) {
67      int max = 100; // let's not draw too much current!
68      t %= max;
69      for (int i = 0; i < len; ++i) {
70          put_pixel(t * 0x10101);
71          if (++t >= max) t = 0;
72      }
73  }
74
75  typedef void (*pattern)(uint len, uint t);
76  const struct {
77      pattern pat;
78      const char *name;
79  } pattern_table[] = {
80          {pattern_snakes,  "Snakes!"},
81          {pattern_random,  "Random data"},
82          {pattern_sparkle, "Sparkles"},
83          {pattern_greys,   "Greys"},
84  };
85
86  int main() {
87      const uint LED_PIN = WS2812_Power_PIN;
88      gpio_init(LED_PIN);
89      gpio_set_dir(LED_PIN, GPIO_OUT);
90      gpio_put(LED_PIN, 1);
91
92      //set_sys_clock_48();
93      stdio_init_all();
94      printf("WS2812 Smoke Test, using pin %d", WS2812_PIN);
95
96      // todo get free sm
97      PIO pio = pio0;
98      int sm = 0;
99      uint offset = pio_add_program(pio, &ws2812_program);
100
101     ws2812_program_init(pio, sm, offset, WS2812_PIN, 800000, IS_RGBW);
102
103     int t = 0;
104     while (1) {
105         int pat = rand() % count_of(pattern_table);
106         int dir = (rand() >> 30) & 1 ? 1 : -1;
107         puts(pattern_table[pat].name);
108         puts(dir == 1 ? "(forward)" : "(backward)");
109         for (int i = 0; i < 1000; ++i) {
110             pattern_table[pat].pat(NUM_PIXELS, t);
111             sleep_ms(10);
112             t += dir;
113         }
114     }
115 }
116
```

make the light sparkle

output the grey color

output different patterns

Initiate the GPIO

```
// -------------------------------------------------- //
// This file is autogenerated by pioasm; do not edit! //
// -------------------------------------------------- //

#pragma once

#if !PICO_NO_HARDWARE
#include "hardware/pio.h"
#endif

// ------ //
// ws2812 //
//        //

#define ws2812_wrap_target 0
#define ws2812_wrap 3

#define ws2812_T1 2
#define ws2812_T2 5
#define ws2812_T3 3

static const uint16_t ws2812_program_instructions[] = {
            //     .wrap_target
    0x6221, //  0: out    x, 1            side 0 [2]
    0x1123, //  1: jmp    !x, 3           side 1 [1]
    0x1400, //  2: jmp    0               side 1 [4]
    0xa442, //  3: nop                    side 0 [4]
            //     .wrap
};

#if !PICO_NO_HARDWARE
static const struct pio_program ws2812_program = {
    .instructions = ws2812_program_instructions,
    .length = 4,
    .origin = -1,
};

static inline pio_sm_config ws2812_program_get_default_config(uint offset) {
    pio_sm_config c = pio_get_default_sm_config();
    sm_config_set_wrap(&c, offset + ws2812_wrap_target, offset + ws2812_wrap);
    sm_config_set_sideset(&c, 1, false, false);
    return c;
}

#include "hardware/clocks.h"
static inline void ws2812_program_init(PIO pio, uint sm, uint offset, uint pin, float
freq, bool rgbw) {
    pio_gpio_init(pio, pin);
    pio_sm_set_consecutive_pindirs(pio, sm, pin, 1, true);
    pio_sm_config c = ws2812_program_get_default_config(offset);
    sm_config_set_sideset_pins(&c, pin);
    sm_config_set_out_shift(&c, false, true, rgbw ? 32 : 24);
    sm_config_set_fifo_join(&c, PIO_FIFO_JOIN_TX);
    int cycles_per_bit = ws2812_T1 + ws2812_T2 + ws2812_T3;
    float div = clock_get_hz(clk_sys) / (freq * cycles_per_bit);
    sm_config_set_clkdiv(&c, div);
    pio_sm_init(pio, sm, offset, &c);
    pio_sm_set_enabled(pio, sm, true);
}

#endif
```

```
61
62  // ---------- //
63  // ws2812_parallel //
64  // ---------- //
65
66  #define ws2812_parallel_wrap_target 0
67  #define ws2812_parallel_wrap 3
68
69  #define ws2812_parallel_T1 2
70  #define ws2812_parallel_T2 5
71  #define ws2812_parallel_T3 3
72
73  static const uint16_t ws2812_parallel_program_instructions[] = {
74              //     .wrap_target
75      0x6020, //  0: out     x, 32
76      0xa10b, //  1: mov     pins, !null             [1]
77      0xa401, //  2: mov     pins, x                 [4]
78      0xa103, //  3: mov     pins, null              [1]
79              //     .wrap
80  };
81
82  #if !PICO_NO_HARDWARE
83  static const struct pio_program ws2812_parallel_program = {
84      .instructions = ws2812_parallel_program_instructions,
85      .length = 4,
86      .origin = -1,
87  };
88
89  static inline pio_sm_config ws2812_parallel_program_get_default_config(uint offset) {
90      pio_sm_config c = pio_get_default_sm_config();
91      sm_config_set_wrap(&c, offset + ws2812_parallel_wrap_target, offset +
    ws2812_parallel_wrap);
92      return c;
93  }
94
95  #include "hardware/clocks.h"
96  static inline void ws2812_parallel_program_init(PIO pio, uint sm, uint offset, uint
    pin_base, uint pin_count, float freq) {
97      for(uint i=pin_base; i<pin_base+pin_count; i++) {
98          pio_gpio_init(pio, i);
99      }
100     pio_sm_set_consecutive_pindirs(pio, sm, pin_base, pin_count, true);
101     pio_sm_config c = ws2812_parallel_program_get_default_config(offset);
102     sm_config_set_out_shift(&c, true, true, 32);
103     sm_config_set_out_pins(&c, pin_base, pin_count);
104     sm_config_set_set_pins(&c, pin_base, pin_count);
105     sm_config_set_fifo_join(&c, PIO_FIFO_JOIN_TX);
106     int cycles_per_bit = ws2812_parallel_T1 + ws2812_parallel_T2 + ws2812_parallel_T3;
107     float div = clock_get_hz(clk_sys) / (freq * cycles_per_bit);
108     sm_config_set_clkdiv(&c, div);
109     pio_sm_init(pio, sm, offset, &c);
110     pio_sm_set_enabled(pio, sm, true);
111 }
112
113 #endif
114
115
```